Improving Selection of Off-Screen Targets with Hopping

Pourang Irani
Computer Science Department
University of Manitoba
Winnipeg, Manitoba, Canada
irani@cs.umanitoba.ca

Carl Gutwin
Computer Science Department
University of Saskatchewan
Saskatoon, Saskatchewan, Canada
gutwin@cs.usask.ca

Xing Dong Yang
Computer Science Department
University of Alberta
Edmonton, Alberta, Canada
xingdong@cs.ualberta.ca

ABSTRACT

Many systems provide the user with a limited viewport of a larger graphical workspace. In these systems, the user often needs to find and select targets that are in the workspace, but not visible in the current view. Standard methods for navigating to the off-screen targets include scrolling, panning, and zooming; however, these are laborious when users cannot see a target's direction or distance. Techniques such as halos can provide awareness of targets, but actually getting to the target is still slow with standard navigation. To improve off-screen target selection, we developed a new technique called *hop*, which combines halos with a teleportation mechanism that shows proxies of distant objects. Hop provides both awareness of off-screen targets and fast navigation to the target context. A study showed that users are significantly faster at selecting off-screen targets with hopping than with two-level zooming or grab-and-drag panning, and it is clear that hop will be faster than either halos or proxy-based techniques (like drag-and-pop or vacuum filtering) by themselves. Hop both improves on halo-based navigation and extends the value of proxies to small-screen environments.

Author Keywords

Navigation, graphical workspaces, off-screen targets, halo, vacuum filtering, drag-and-pop, proxy targets.

ACM Classification Keywords

H5.2 [User Interfaces]: Interaction styles.

INTRODUCTION

Designers of visual applications are commonly challenged with the task of adequately displaying all the information required by users in the available viewing space. As a result, many applications – such as map browsers, graphical editing programs, or visualization systems – present a graphical workspace that is considerably larger than the screen. In these systems, only a small subset of the information is displayed in the viewport, and a large quantity of information resides outside the viewing region.

This situation can exist for any size display, but it affects small-screen devices such as handhelds and PDAs more strongly (Figure 1). As display space is reduced, even small visual datasets will be partly located outside the view. For these devices, techniques are required for retrieving, inspecting, and manipulating off-screen content.

Figure 1. Large visual workspace with several candidate targets (R=Restaurants; M=Metro Stations), with PDA viewport superimposed.

On most platforms, zooming, panning, and scrolling are the most common navigation tools available for accessing off-screen content. However, these techniques often require considerable navigational effort from the user [20]. Interactive focus+context and overview+detail views, such as fisheye or radar views, have also been considered as ways to facilitate the presentation of large content on small displays [22, 24]. While these techniques adjust the presentation and layout of large information spaces for limited viewports, they introduce additional interactive and cognitive costs (e.g., [10]).

For the task of selecting off-screen targets, no current method is able to satisfy all of the following design goals:

- *off-screen object awareness*: users should be able to stay aware of the presence and locations of potential targets that are off-screen;

- *minimal navigation*: users should be able to move to off-screen targets without much more effort than what is required for on-screen objects;
- *context visibility*: users should be able to determine and make use of the environment surrounding a target;
- *full-scale view*: users should be able to see potential targets in enough detail to determine if they are important.

To better address the problem of selecting off-screen targets, we designed and evaluated a new technique, called *hop* (from 'halo+proxy'). Hop uses object halos to provide awareness of off-screen targets [1], a proxy technique to bring targets close to the user's cursor [2, 6], and a teleportation mechanism to transport the user to the location and context of the target. To test the effectiveness of hop, we carried out an experiment that compared the performance of hopping with zooming and panning in an off-screen targeting task. The study showed that hopping is significantly and substantially faster than either two-level zooming or grab-and-drag panning. Hop demonstrates again that halos are valuable, and also shows that proxy-based selection techniques can be successfully used in small-screen environments.

In the following sections we review related literature in visual workspaces and selection techniques, describe the design of hop in more detail, and report on the methods and results of the study. We then consider ways that hop can be used to improve current systems, and discuss the underlying reasons for hop's success.

RELATED WORK
Several areas of previous work are relevant to the new technique: 2D navigation methods, visualization methods for off-screen objects, and proxy and portal methods for accessing distant objects.

2D Navigation Methods
Scrolling, Panning, Zooming
Most mainstream applications allow users to scroll, pan or zoom to view off-screen content. To see off-screen content, scrolling interfaces provide widgets such as scroll bars or scroll-rings [19, 25]; however, scrolling still requires considerable effort to get to off-screen locations. Several improvements to basic scrolling have been studied, including integration of scrolling into physical devices such as keyboards and mice [12, 28], rate-based scrolling [28] that maps the displacement of an input device to the scrolling velocity [12], and speed-dependent automatic zooming (SDAZ) [14] to reduce the motion-blur encountered at high scroll speeds [14, 9]. All scrolling interfaces, however, require that the workspace can only be inspected linearly. As a result, users must spend more time when objects are further away.

Panning allows the user to view off-screen content by moving the workspace under a fixed viewport [15]. A panning operation is defined by a click-drag-and-release to shift a subset of the workspace into view. This form of interaction limits the amount of displacement that takes place at each pan operation. Studies [15, 16] have compared different panning methods for a variety of tasks. One result [15] shows that for touch-based systems, panning the document into view was better than dragging the viewport around the workspace. Similar to scrolling, panning presents off-screen content linearly; as a result, users must perform multiple pan operations to locate distant items.

Zooming is an effective navigation method that provides multiple perspectives of the workspace. Zoom techniques show that being 'off-screen' is only relative to a particular zoom level, and that any amount of the workspace can be brought into view, albeit at the cost of detail. Unlike scrolling or panning, zooming allows users to view off-screen content in a non-linear fashion (far-away objects can be inspected before those that are close). Pad++ [5] facilitates zooming in and out of a workspace using multiple scaling factors. Overviews that result from zooming-out provide *awareness of off-screen content* to users. These overviews perform better than regular scrolling systems [16]. However, to find a particular off-screen object from a set of candidates, the user may have to perform multiple zoom operations.

Overview+detail and Focus+context
Methods like overview+detail views, fisheye views, and chunking are also designed to facilitate interaction with large workspaces on limited displays. Unlike traditional techniques, these techniques work by modifying the representation of the workspace.

Overview+detail techniques [4, 18] present a condensed overview of the workspace. The user can expand subsets of the information space when required. In many implementations, multiple windows are used for presenting overviews separately from the details. In these systems additional interaction overhead is required to manage the windows. With overviews, the user has to initiate a series of inspections by first locating the off-screen content of interest in the overview and then examining the details to determine if this is the content of interest. Nevertheless, this strategy can be effective: for example, one study showed that users perform equally well with overview+detail presentation as with zooming and panning for spatial cognition tasks [3].

Focus+context views such as fisheyes [22] present a distorted view of the workspace. The most relevant information is magnified while less important material is reduced so that the entire workspace can fit into the viewport. Results of one study [11] show that fisheye views can be as good as traditional interfaces for steering tasks. However, fisheye views present usability problems in targeting and memorability, and performance in reading large documents with fisheyes is worse than with overview+details [13]. Furthermore, the distortion caused

by fisheye views degrades tasks relying on spatial cognition or short-term recall. The distorted views can also make it difficult to inspect the details of target items.

Recently, chunking methods have been developed to break large content space into manageable and viewable portions. Flipzoom [7] is a technique that segments the entire workspace into viewable units. At any given time only one segment of the workspace is visible. This technique has been used for viewing web pages [8] or for viewing large documents in limited viewing spaces. Zonezoom [21] is another technique that segments the workspace into regions. It uses smooth in/out transitions of the segments to allow the user to view details of off-screen content. As with the overview+detail methods, chunking requires that the user be able to interpret the overview representations and recreate the relationships between the details and the whole. Overall, the additional complexity in interacting with overview+details, fisheye and chunking techniques may in some cases outweigh the benefits they offer.

The design of hop was influenced by two general classes of techniques: off-screen visualization and proxies. These are discussed in detail next.

Off-Screen Object Visualization

Halo [1] is a visualization technique that shows the distance and location of off-screen objects. Halo was built on a well-known principle in cinematography referred to as the *partially-out-of-the-frame* technique. Based on this technique, viewers get a feeling for the presence of a prop outside the scene and can recreate its characteristics based on the portion in view.

With halo, objects outside the viewport are surrounded by rings that are large enough that a portion of each ring is visible on the edge of the viewport. From the visible portion, users can infer the location of the object and the distance from the viewing space. Halos have been compared to the typical arrow visuals used in video games and maps to point to objects off-screen [1]. Halos improved performance by 16% to 33% for most tasks in comparison to simple arrows with numeric distance information [1].

City lights [27] are built on the same principles as halos. City lights visualize off-screen objects by placing rectangular blocks on the edges of the viewport. Additionally, city lights use visual cues such as color, shape and size to provide information about the physical properties of the off-screen objects, or other abstract information such as the degree of interest.

Halos and city lights are successful techniques for pointing users to the presence of off-screen objects. However, on their own they do not assist the user in navigating to the object for inspection or manipulation.

Proxies and Portals

The development of large screens and multi-display systems has produced a new series of interaction methods that bring distant objects closer to the user's interaction space. These techniques rely on *proxies* – temporary duplicates of the object that allow actions on the original – or on a *portal* that gives the user a window into a remote area of the workspace.

Drag-and-pop [2] is a proxy-based technique that creates local copies of objects that are located far away on a large display. The user makes a simple gesture and any distant object located within a +/- 30 degree arc are brought toward the user's cursor in the form of a proxy. The user can then interact with the proxies as they would with the original object. This significantly reduces the physical movement required for a user to interact with remote objects. Drag-and-pop showed significant savings in the time to select distant objects in comparison to conventional dragging. However, drag-and-pop is limited in the number of proxies it can provide to the user, in its ability to allow multiple operations, and in allowing the user to control which objects are rendered as proxies.

Vacuum filtering [6] was designed to overcome some of the limitations of the drag-and-pop approach. As in drag-and-pop, the underlying principle behind the vacuum is to bring distant objects closer to the user. In the vacuum, the user triggers the proxies by initiating a mouse down and drag operation. This creates an arc of influence (or vacuum) that 'pulls' proxies of distant objects towards the cursor. The vacuum shrinks the size of the proxies to maintain the relative layout of objects.

Both techniques have several shortcomings with respect to off-screen targeting:

- *size and number of proxies:* in drag-and-pop, the number of proxies are limited so there is no overlap. Although vacuum does not have a limit on the number of proxies, it must shrink the proxies in order to maintain the relative distances between the original objects. As a result, it can become difficult to view object details when many objects are 'vacuumed.'

- *off-screen object awareness:* both techniques are designed for large screen displays. As a result, they do not currently include any means for showing the presence of off-screen objects.

- *arc of influence:* the arc of influence created by the vacuum attracts objects in a larger radius as the vacuum gets closer to the edge of the workspace. This approach could 'vacuum in' large numbers of off-screen objects that may not be of interest to the user.

- *context visibility*: in the proxy approach, the details surrounding the original object are not available. While the vacuum maintains the relative distance between objects, elements around the original objects are not visible (such as the underlying map in Figure 1). The user still has to move to the distant location to inspect the surroundings of the objects.

Portals into remote spaces are an alternative to proxies. Portals behave like windows that facilitate content viewing

in the original workspace. Frisbees [17] was designed as a telescope into the remote viewing space. The technique provides controls to move objects between various workspaces and manipulate them locally. Similarly, WinCuts [26] allows users to remotely interact with defined regions of existing windows. Unlike proxy-based methods, portals facilitate the viewing of the context around remote objects. However, additional operations such as zooming in and out of the portal are necessary to view the context. Significant overhead results from inspecting objects that are off-screen, which makes portal-based tools less suitable for lightweight tasks that involve inspecting remote objects and then returning to work on nearby content.

THE HOP (HALO+PROXY) TECHNIQUE

Hop is an interaction technique that enables quick access to off-screen objects. Hop works by providing awareness of off-screen targets, by bringing target proxies close to the user, and by transporting the user to the context of the target. The design of hop was driven by, and satisfies, the design principles outlined in the introduction. We describe the design of hop below in terms of its three components.

Halos: Awareness of off-screen objects

Hop adapts and enhances the halo [1] representation to satisfy the *off-screen object awareness* requirement. Halos are drawn using elliptical and circular lines from each off-screen object (see Figure 2); the result in the viewport is that an arc segment is visible on the edge of the screen for each off-screen item.

Our initial implementation of the halo showed a problem with the technique: when the number and distance of off-screen targets increases, halos overlap and become cluttered at the edge of the screen. This problem led to a slight improvement to save space along the edges of the screen. Objects that are directly north, east, south, and west of the viewport are represented using ellipses, which reduces the amount of overlapping along the edges when compared to circular rings. The idea of using ellipses is from Baudisch and Rozenholtz, who speculate in [1] that ovals could better convey off-screen distances than circles. Off-screen objects in the corner regions are indicated by circular halos.

Laser Beam: Invoking proxies

The second component of the hop technique is the *laser beam*. Hop uses a moveable 'beam' line to trigger the creation of proxies from the off-screen objects. Our version of the laser beam is a small improvement over other mechanisms (such as that used in the vacuum) that have a wider invocation range. Hop's laser beam interacts only with halos on the screen's edge. Our modification allows the user to more precisely select objects they would like to inspect.

The laser beam is invoked by clicking the mouse on the background and dragging the cursor toward an edge. The distance traveled by the cursor is indicated by the *circle of movement*, the center of which is located at the mouse-

down position. The circle of movement is later used for laying out the proxies (explained below). The laser beam is drawn from the center of the circle of movement up to the edge of the screen (see Figure 2).

The user then moves the cursor in a radial fashion, and the laser beam travels until it intersects a halo. For each beam-halo intersection, a proxy is created and placed near the circle of movement (details on layout are given below). Proxies remain opaque for one second, and then begin fading away. In the current hop system, proxies disappear completely after five seconds. Locations occupied by previous proxies are made available for any new proxy in that region.

At any point after creating the proxies, the user can release the mouse button and select a proxy, which teleports the user to the off-screen object.
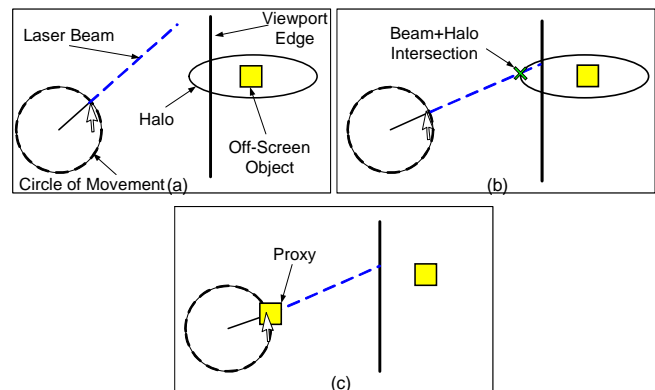

Figure 2. Invoking proxies with the laser beam: a) beam is created; b) beam intersects halo; c) proxy created.

Teleporting: Moving to the off-screen object

The final operation in hop is to move the viewport to the off-screen object. Clicking on a proxy invokes a 400ms-long animated transition from the current location to the location of the off-screen item. We added the animation after noticing that users would lose their orientation in a rapid movement toward the object.
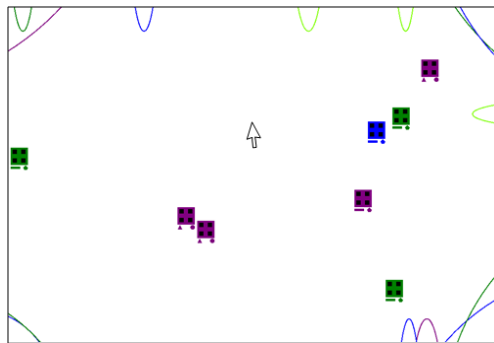
Moving the viewport to the location of the off-screen target ensures that the local environment of the item can be inspected (satisfying the *context visibility* requirement introduced earlier). The local environment is critical in applications such as the mapping system shown in Figure 1. Moving the viewport also ensures that items are displayed in their original size so that details can be inspected (satisfying the *full-scale view* requirement).

The sequence of actions for a complete hop is depicted in Figure 3.
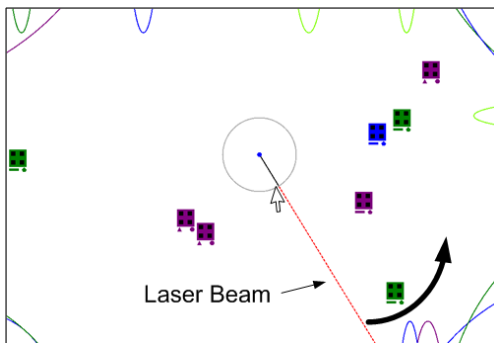
Layout of Proxies

As with similar interaction tools that invoke proxies (such as the vacuum, drag-and-pick, and tractor beam), a common design challenge is the layout of proxies for rapid interaction. The initial design of hop laid out the proxies in
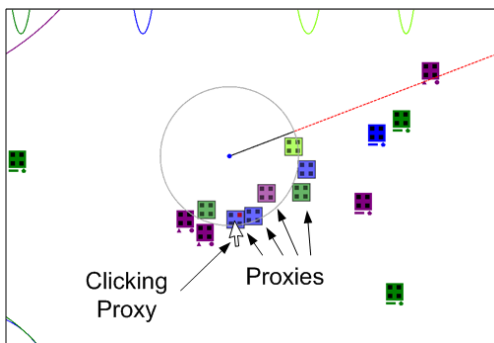
a linear order, but this layout caused the user to reach over further distances if the desired proxy was invoked later in the movement. To reduce the distance between the cursor and any proxy, we use a radial layout for the proxies.
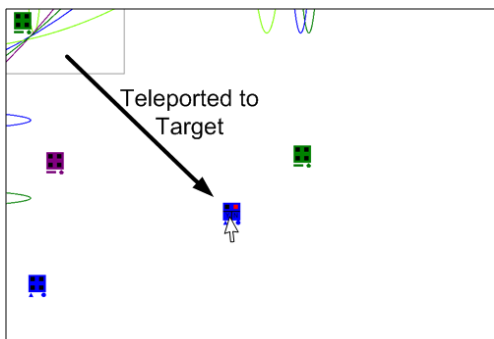

(a) Original View

(b) Laser Beam moving counter-clockwise

(c) Proxies are invoked once the laser beam intersects the halos; user clicks on a proxy

(d) Selecting a proxy teleports the user; final view after teleporting.

Figure 3. Sequence of actions in a hop.

The algorithm initially segments off-screen objects into four regions, based on the mouse-down location. Each region in the viewport will host proxies from the same off-screen region. Proxies are placed on the circle of movement created by dragging away from the original mouse-down location. The layout algorithm avoids overlaps by placing proxies either to the left or right of any existing objects. When the circle of movement becomes full, additional proxies are laid out on the next-larger concentric circle. This process continues until all proxies are drawn. The space from faded proxies is reused for new ones. The algorithm did not preserve the remote layout of the objects since that would require shrinking the proxies (as in [6]).

STUDY METHODS

We carried out a user study to evaluate whether hopping assists people in selecting off-screen targets, by comparing it to zooming and panning techniques.

Participants and apparatus

Thirteen paid volunteers (10 male, 3 female) were recruited from a local university. All users were frequent users of mouse-and-windows based systems (at least 12 hours per week). Participants had a variety of experience with zooming and panning techniques: five had more than three years' experience with both, two were experienced with zooming but not panning, and six were not experienced with either technique. Participants stated they had used zooming and panning in map systems and image editors. None were familiar with halos, proxy techniques, or hopping.

Participants performed the experiment on a P4 Windows XP PC running a custom .NET application. The display was a 17" monitor set to 1280×1024 resolution.

Tasks

The system presented two-dimensional target selection tasks in several different distance and density conditions (see Figure 4). The task involved consecutive selection of 10 targets in the presence of distracters. Participants were required to click a sequence of off-screen objects which were designated as targets.

Targets were rendered as 32×32 pixel squares. A target was differentiated from a distracter by the presence of a red square in the upper right corner of the object (Figure 5). The targets appeared sequentially; after clicking on one target, the next target appeared somewhere in the workspace, and the participant began to look for it.

To simulate the importance of contextual information that is an important part of real-world systems, we added a decision to the task that could only be made correctly by observing local information. In addition to the presence of a red square, participants were required to correctly identify 'true' and 'false' targets. A 'true target' was accompanied by a triangle and circle landmark (Figure 5) which consisted of the two shapes drawn below the target. 'False

targets' were targets that contained the red square but showed a different landmark (see Figure 5). Upon locating a target, the participant was asked to click on the label ('Y' or 'N') to indicate whether the target was true or false. Half of the targets in each block were 'true targets.'

15 objects, 600 pixels   15 objects, 1200 pixels

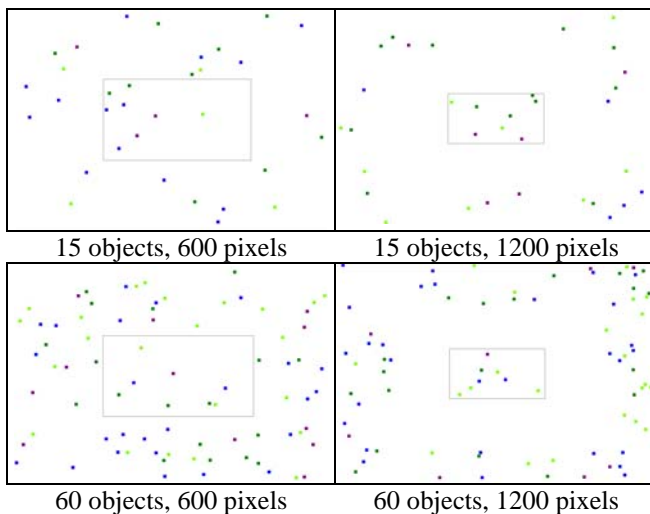60 objects, 600 pixels   60 objects, 1200 pixels

Figure 4. Example workspaces from study for two different distance and density conditions; each dot is either a target or a distracter. Starting viewport rectangle is shown in grey.

Targets were randomly distributed to eight relative compass directions (E, NE, N, NW, W, SW, S, SE) outside the boundaries of the screen. To ensure that participants performed a minimal amount of navigation, the next target in the list of targets did not appear in the regions adjacent to the region containing the previous target. This ensured that two sequential targets were not on the same screen so that the participant would have to perform at least one navigation operation to locate the next target.
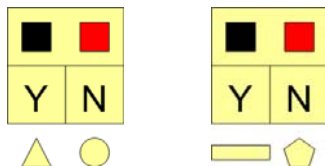

Figure 5. Targets, showing red square at top right. True target (with triangle and circle landmark) is shown at left; false target is shown at right.

Experimental conditions

The study involved three factors: navigation technique, off-screen distance, and density.

Navigation Techniques

The main focus of our research was based on evaluating the performance of hopping against different techniques for off-screen targeting tasks. We chose a two-level zoom and a grab-and-drag panning as comparison techniques; these two were chosen because they represent the most common techniques seen in current applications (other possibilities are considered in the discussion). A pilot study also included scrolling as a comparison technique, but initial data showed that scrolling was consistently and significantly outperformed by the other three techniques.

- *Zooming*. A two-level zoom was used in the study. Two-level zoom provides users with either an overview of the entire information space, or a fully-zoomed-in view. Users can switch between the views by clicking the right mouse button. In the zoomed-out view, all objects were visible, but details of the targets could not be seen. To examine an object and determine whether it is a target, users had to zoom in by moving their mouse near the object and clicking the right mouse button.

- *Panning*. This technique implemented the typical panning technique with a small improvement. The viewer moves the viewport by a mouse down-slide-and-release operation. The improvement takes into account the momentum of the pan operation to slide the viewport additional pixels in the direction of movement. The system shows the objects in actual size at all times.

- *Hopping*. Hop was implemented as described above: users click the mouse on the background and drag to create a circle of movement. They then sweep the laser beam across one or more halos to create proxies on the circle, and may at any time select one of the proxies to teleport to that object's real location. Participants had to navigate to the distant location in order to see landmarks and click the 'Y' or 'N' buttons on the target.

Off-Screen Distance

To determine whether performance varies with the distance of objects beyond the edge of the screen, we tested two distances. The *short range* positioned the objects at 600 pixels beyond the edge of the screen. The *long range* positioned objects randomly between 600 pixels and 1200 pixels beyond the edge of the screen.

Density

In most targeting tasks, the time to locate a target typically increases with the number of objects or the density of the information space. To determine whether density affected performance, three density values were used: *few* (15 objects), *some* (30 objects) and *many* (60 objects). Ten of the off-screen objects were used as targets; in addition to the off-screen objects, eight distracter objects were always displayed within the starting viewport.

Experimental Design

The study used a 3×2×3 within-participants factorial design. The factors were:
- Navigation technique: zoom, pan, hop
- Off-Screen Distance: 600, 1200 pixels.
- Density: 15, 30, 60 objects.

Interaction technique was fully counterbalanced using a Latin square; the other two factors were always presented in increasing order (i.e., from smaller to larger distance, and from smaller to larger densities). Within each condition, participants carried out 2 blocks of 10 off-screen targeting

trials. Each block consisted of trials for all combinations of Technique × Off-Screen Distance × Density.

With 12 participants, 3 navigation techniques, 2 distances, 3 densities, and 20 trials per condition, the system recorded a total of 4320 trials. The study system collected completion times and error information for each target. An error consisted of clicking 'N' on 'true target' and 'Y' on a 'false target'. Participants filled out a brief questionnaire asking them about their preferences at the end of the experiment.

Procedure

Participants were randomly assigned to one of the six order groups. Prior to starting the experiment, participants were given a short warm-up session (10 trials per technique) to practice off-screen target selection with the different interaction styles. Upon completing the 10 practice trials, all participants indicated they were comfortable with all three systems. Participants then completed the off-screen targeting tasks and were allowed to take breaks between the two blocks and between techniques. After all conditions for a session were complete, participants were asked to indicate the technique that was easiest and the technique for which they felt they performed the fastest.

RESULTS

Completion time

Effects of Navigation Technique
A repeated measures ANOVA showed a significant main effect of *navigation technique* ($F_{2,36}$=45.46, p<0.001). As shown in Figure 6, overall mean times for hop were fastest (5.83 secs) followed by zoom (12.52 secs) and pan (14.38 secs). Post-hoc pairwise t-tests showed that hop was significantly faster than zoom and pan (both p<0.05) but did not show any significant difference between zoom and pan. The performance of each navigation technique in each of the distance and density conditions is analyzed below.
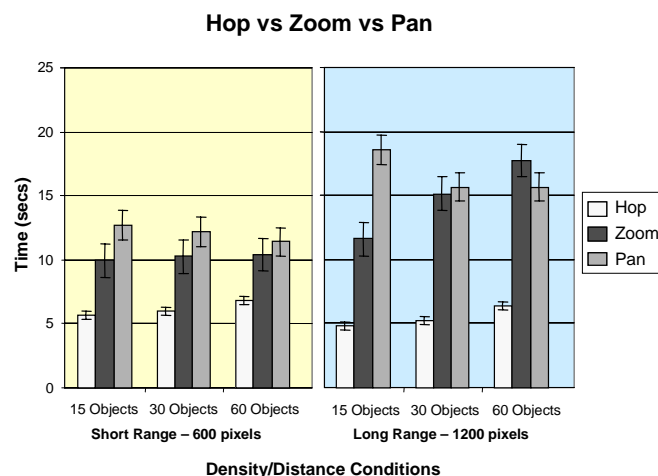

Hop vs Zoom vs Pan

Figure 6. Mean completion time for all conditions. Error bars show one standard error from the mean.

Effects of Distance
As expected, *off-screen distance* of objects also had a significant main effect on completion time ($F_{1,24}$=12.42, p<0.01). A significant Technique × Distance interaction ($F_{2,24}$=25.65, p<0.001) was also present. Post-hoc pairwise comparisons showed that hop was significantly faster than zoom and pan (p<0.001 for both) for both distances. The results also show that zoom is significantly faster than pan (p<0.05) at off-screen distance of 600 pixels. At the off-screen distance of 1200 pixels, average zoom performance is slower than pan, but this difference is not significant.

We also analyzed the effect of *distance* for each technique. Interestingly, at the off-screen distance of 1200 pixels, average performance time with hop is 5.48 secs compared to 6.18 secs at the 600 pixel level. This difference is not significant. The results show that users were significantly faster ($F_{1,12}$=30.091, p<0.001) with zooming at a distance of 600 pixels (10.213 secs) than at a distance of 1200 pixels (14.836 secs). As expected, users were also significantly faster ($F_{1,12}$=39.498, p<0.001) with panning at the 600 pixel distance (12.11 secs) than at the 1200 distance (16.65 secs).

Effects of Density
Surprisingly, the results do not reveal main significant effect of *density* on performance time ($F_{2,24}$=3.24, p=0.057). However, a significant Technique × Density interaction was present ($F_{4,48}$=10.068, p<0.001). Post-hoc pairwise comparisons showed that hop was significantly faster than zoom and pan (p<0.001 for both) for all densities. However, zoom is only significantly better than pan at the 15-object density but not the others.

We also analyzed separately the effect of density on each technique. With the hop technique, participants are significantly faster at levels of 15 and 30 objects compared to 60 objects (p<0.001 for both). With the zoom technique we also observe significant main effects of density. Post-hoc pairwise comparisons show that participants are significantly faster with 15 objects than with 60 objects (p<0.001), but there is no significant difference between 15 and 30 objects or 30 and 60 objects. Interestingly, participants performed equally well on all three densities at the 600 pixel off-screen distance but were significantly slower between densities at the 1200 pixel level.

With the pan technique, the results show significant main effect of *density* ($F_{2,24}$=7.564, p=0.003). Pairwise comparisons show significant difference between 15 and 30 objects (p=0.039) and between 15 and 60 objects (p=0.020) but not between 30 and 60 objects. Surprisingly, participants were faster with panning as the number of objects increased. As discussed below, one explanation is that participants became disoriented in sparse workspaces.

Errors

An error was recorded if the user selected the incorrect type of target. The results showed that users were 99.99% accurate with panning and zooming and 99.98% accurate

with hop. The difference is not significant and a statistical test was not performed. One explanation for slightly lesser accuracy with hop was that on certain trials a few users aimed at the target during the teleportation. This resulted in recording incorrect clicks because the scene was animated.

User Preferences

Twelve out of thirteen participants indicated that they felt they were fastest with hop; one felt that zoom was fastest. All thirteen participants found hop to be the easiest to use.

Observations of Navigation Patterns

Watching users carry out the task showed clear navigation patterns for the different techniques. With zooming interfaces, participants started each trial by a zoom-out operation, followed by a zoom-in operation on the largest cluster in the scene. From this initial inspection, users performed clockwise or counter-clockwise zoom-out/zoom-in operations until the target was located. With panning, participants panned vertically or horizontally to initially bring the off-screen targets into view. Subsequent panning operations were performed, either clockwise or counter-clockwise, until the target was located.

Participants applied two distinct navigation patterns with hop; one pattern was used at the beginning of the experiment and the second pattern was used in subsequent trials until the end of the experiment. In the early trials, a participant would invoke the laser beam and aim it toward individual halos. All users would begin and complete a sweeping movement of the laser in proximity of a halo. If the resulting proxy represented a target of interest, the user performed the teleportation; otherwise the user continued in small steps to invoke individual proxies. However further into the experiment, we noticed that participants performed wider sweep operations with the laser beam. This action invoked many proxies simultaneously. The user scanned the proxies on the screen and then initiated the teleportation. In these later trials, the center of the laser beam sweep was usually closer to the edge of the screen than in earlier trials.

DISCUSSION

The user study provides evidence that a combination of halos and proxies is an efficient way to find and select off-screen targets. The main findings were:

- Selection times with hop were approximately half of what they were with either zoom or pan;
- Performance with hop remained constant regardless of changes in the distance of off-screen objects;
- Selection time with the zooming interface increased both with the number of objects and with object distance;
- Performance with panning improved as the number of objects increased.

In the following sections we consider reasons for these results, we discuss how hop will generalize in real-world systems, and summarize the main lessons for designers.

Reasons for our findings

The main reason for the performance of hop is the reduction in the number of navigation actions that users need for completing the task. Since the first part of the task is to find and evaluate targets, bringing all desired targets towards the user greatly reduces the amount of navigation work required. A hop involves one set of actions – a click-and-drag to create the circle of movement, another drag to bring proxies into range, and a visual search and selection to choose a proxy. In contrast, searching for a target in both zoom and pan requires multiple actions. We recorded the number of actions of these types: each trial with hop required about 1.3 operations as described above, each trial with zoom required about 16 operations, and each trial with pan required about 21. The high correlation between the number of navigations and performance times shows that the amount of navigation is a significant factor in off-screen targeting. This result confirms the *minimal navigation* design principle defined earlier.

A second result that requires explanation is that panning was faster in higher-density conditions. One reason is that larger densities helped participants stay oriented and aware of screen boundaries. Participants became disoriented when locating an off-screen target with panning, a problem that was more acute when several panning operations did not reveal any objects. The lower concentration of distracters in small densities typically resulted in users panning multiple times to locate clusters. With larger densities, these distracters themselves gave information about direction of travel and of the boundary of the workspace. With a denser scene, the chances of the user going astray during a pan operation was less likely than in a sparse scene.

Limitations to the hop technique

Although hop was by far the most effective technique in our study, there are certain limits to the technique.

- *Clutter.* Halos add visual information to the screen, and even with our modifications to the original halo presentation, large numbers of objects will occlude the screen edges, reducing usable space. Hop can still work in cluttered situations, although the user has reduced control over which proxies are created. The clutter can be reduced by replacing halos with glyphs. The radius of the glyph could indicate the distance of the off-screen object. When two or more off-screen objects are in proximity, concentric circles could be inserted in the glyph, as in [1].
- *The nature of the inspection task.* In our study, the hop proxies provided the right amount of detail to determine whether objects were legitimate targets. However, other tasks may require information that is not provided by our current proxies. For example, if contextual information is important in deciding on targets, a hop user may need several teleportations before finding the target.
- *Large-scale context.* Hop, like all proxy techniques, takes targets out of context in order to bring them closer. Techniques that preserve awareness of the overall

context, such as zooming, may allow users to better decide on which targets are most likely to be legitimate.

- *Getting lost*. The teleportation that is inherent to hop may cause some users to lose track of where they are in the overall space. We believe that this problem could easily be remedied by the addition of an inset overview that shows the user's current location in the workspace.

Using hop in real-world applications

The major strength of hop is that it allows the user to navigate quickly to an interest area off-screen to inspect whether information in that region is important. However, there are several issues that must be considered if it is to work in real-world tasks.

Filtering. Hop depends on a filter applied by the user that specifies which objects will be inspected. Once the filter is applied, hop can display the halos for all the objects off-screen. The remainder of the interaction technique would behave similarly as in the experimental setup.

Drag-and-drop to off-screen locations. Many applications will require that users be able to take objects to the off-screen location. Hop can be adapted for this task; since the proxies stay on the screen for a short time after they appear, the user could drag an object to the proxy to initiate the action. If the visible lifespan of the proxy is too short, users could also drag the proxy onto the real object; the proxy would 'capture' the object to be moved, and the teleportation would begin.

Small-screen devices. The hop technique can be adapted for small viewports such as those available on PDAs or larger mobile phones. However, as with other proxy-based techniques, the number of proxies that can be displayed simultaneously is limited. One approach would be to reduce the time span for which a proxy is visible, allowing other proxies to be shown. Another strategy would be to resize the proxies as has been done with other systems (e.g., [6]).

Alternative Designs for Hop

Several alternative designs can improve various aspects of hop. Augmenting halos (as in [27]) to show information about the object is one possible extension to hop. This design would require that users learn a new mapping from information to visual dimension and does not allow for the amount of detail possible with a proxy. However, enhanced halos could help the user decide which proxies to invoke.

An alternative to the laser beam would be to allow the user to select or cross halos with a pointing device. This technique could work in many situations; however, crossing several halos could be difficult, and dragging a stylus around a screen edge is difficult on devices that do not have a raised frame around the screen. Crossing also requires that the user move away from the original area of interest. Additionally, interacting with a laser beam has benefits over crossing for devices with jog-dial controls. In these cases the user can press the jog-dial to invoke the laser, then spin the jog-dial to rotate the laser and press the dial to

select a target. However, for a wider range of flexibility, crossing can be used as an adjunct to the laser.

Comparing hop to other navigation techniques

As stated above, we compared hop to zoom and pan because these are the most common techniques in current applications. There are, however, a number of modifications that could be made to zooming or panning to improve their performance, and there are other navigation techniques that could be tested.

Panning with halos. As shown by Baudisch and Rosenholtz [1], halos improve performance over ordinary panning. However, halos alone do not reduce the time required to pan the view to the target, and therefore we believe that the addition of proxy teleportation in hopping would still outperform this technique.

Focus+context views. Fisheye displays or moveable-magnifier views could allow users to inspect potential targets more quickly than was possible with our implementation of zooming. We plan to test hopping against these techniques in the future; however, both fisheyes and magnifiers introduce additional problems that may reduce their efficacy. For example, fisheye views can make targeting more difficult, and moveable magnifiers can occlude much of the context, particularly on small screens.

Lessons for Practitioners

We believe there are several lessons designers of large visual workspaces might find valuable from our findings:

- In large workspaces that necessitate off-screen targeting, designers should consider hopping as an alternative to zooming or panning.
- Bringing potential targets towards the user (i.e. proxies) assists navigation on small screens as much as it does on large displays, but should be coupled on the small screen with awareness of off-screen content.
- Off-screen targeting techniques should be designed using the minimal navigation principle to reduce overall navigation time.
- For short off-screen distances and sparse datasets, zoom-based interfaces should outperform panning.

CONCLUSION

Many applications provide large visual workspaces with small viewports, resulting in off-screen content. No current navigation technique is able to meet all of the design goals for selecting off-screen targets. We introduced a new technique – *hop* – to address this problem. Hop allows users to quickly and easily navigate to a region outside the viewport; it uses halos to provide an awareness of objects outside the view, a 'laser beam' to create proxies of specific off-screen objects, and a teleportation mechanism that takes the user to the remote location.

We carried out a user study to compare hop to two mainstream navigation techniques, grab-and-move pan and two-level zoom. Users were able to select off-screen targets

in half the time that was needed by either of the other two techniques. In addition, all users preferred using hop. Our results underscore the value of both halos and proxies, and show that these techniques can be successfully combined.

In future work, we are planning to continue studying hop in a variety of settings. First, we plan to test the technique in a more realistic task, in which the background data (e.g., the map) is important for deciding on targets. Second, we plan to investigate hopping with mobile devices and dynamic (e.g., moving) off-screen objects. We also want to test hop in tasks that involve other types of off-screen navigation, such as spatial comparisons between elements in different locations. Last, we plan to investigate extensions to hop, such as adding read wear [23] to halos, to assist the user in keeping track of areas that have already been inspected.

ACKNOWLEDGMENTS

REFERENCES

1. Baudisch, P. and Rosenholtz, R. (2003). Halo: a technique for visualizing off-screen objects. *Proc. CHI'03*, 481-488.
2. Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., Bederson, B., and Zierlinger, A. (2003). Drag-and-pop and drag-and-pick: techniques for accessing remote screen content on touch- and pen-operated systems. *Proc. Interact'03*, 57-64.
3. Baudisch, P., Good, N., Bellotti, V., and Schraedley, P. (2002). Keeping things in context: a comparative evaluation of focus plus context screens, overviews, and zooming. *Proc. CHI'02*, 259-266.
4. Baudisch, P., Xie, X., Wang, C., and Ma, W. (2004). Collapse-to-zoom: viewing web pages on small screen devices by interactively removing irrelevant content. *Proc. UIST '04*, 91-94.
5. Bederson, B. B. and Hollan, J. D. (1994). Pad++: a zooming graphical interface for exploring alternate interface physics. *Proc. UIST '94*, 17-26.
6. Bezerianos, A. and Balakrishnan, R. (2005). The vacuum: facilitating the manipulation of distant objects. *Proc. CHI '05*, 361-370.
7. Björk, S. (2000). Hierarchical flip zooming: enabling parallel exploration of hierarchical visualizations. *Proc. AVI '00.* 232-237.
8. Björk, S., Holmquist, L. E., Redström, J., Bretan, I., Danielsson, R., Karlgren, J., and Franzén, K. (1999). WEST: a Web browser for small terminals. *Proc. UIST '99.* 187-196.
9. Cockburn, A. and Savage, J. (2003). Comparing Speed-Dependent Automatic Zooming with Traditional Scroll, Pan and Zoom Methods. *Proc. CHI'03*, 87-102.
10. Gutwin, C. (2002). Improving Focus Targeting in Interactive Fisheye Views, *Proc. CHI'02*, 267-274.

11. Gutwin, C. and Fedak, C. (2004). Interacting with big interfaces on small screens: a comparison of fisheye, zoom, and panning techniques. *Proc. Graphics Interface '04*, 145-152.
12. Hinckley, K., Cutrell, E., Bathiche, S., and Muss, T. (2002). Quantitative analysis of scrolling techniques. *Proc. CHI '02*, 65-72.
13. Hornbæk, K. and Frøkjær, E. (2001). Reading of electronic documents: the usability of linear, fisheye, and overview+detail interfaces. *Proc. CHI '01*, 293-300.
14. Igarashi, T. and Hinckley, K. (2000). Speed-dependent automatic zooming for browsing large documents. *Proc. UIST '00*, 139-148.
15. Johnson, J. A. (1995). A comparison of user interfaces for panning on a touch-controlled display. *Proc. CHI'95*, 218-225.
16. Kaptelinin, V. (1995). A comparison of four navigation techniques in a 2D browsing task. *Proc. CHI'95 Extended Abstracts*, 282-283.
17. Khan, A., Fitzmaurice, G., Almeida, D., Burtnyk, N., and Kurtenbach, G. (2004). A remote control interface for large displays. *Proc. UIST'04*, 127-136.
18. Lam, H. and Baudisch, P. (2005). Summary thumbnails: readable overviews for small screen web browsers. *Proc. CHI '05*, 681-690.
19. Moscovich, T. and Hughes, J. F. (2004). Navigating documents with the virtual scroll ring. *Proc. UIST '04*, 57-60.
20. O'Hara, K. and Sellen, A. (1997). A comparison of reading paper and on-line documents. *Proc. CHI'97*, 335-342.
21. Robbins, D., Cutrell, E., Sarin, R., & Horvitz, E. (2004). ZoneZoom: map navigation for smartphones with recursive view segmentation. *Proc. AVI'04*, 231-234.
22. Sarkar, M., and Brown, M. (1992). Graphical Fisheye Views of Graphs. *Proc. ACM CHI '92*, 83-91.
23. Skopik, A. and Gutwin, C. (2005). Improving revisitation in fisheye views with visit wear. *Proceedings CHI'05*, 771-780.
24. Smith, R.B., Hixon, R., and Horan, B. (1998). Supporting Flexible Roles in a Shared Space. *Proc. CSCW '98*, 197-206.
25. Smith, G. M. and Schraefel, M. C. (2004). The radial scroll tool: scrolling support for stylus- or touch-based document navigation. *Proc. UIST '04*, 53-56.
26. Tan, D., Meyers, B., Czerwinski, M. (2004). WinCuts: manipulating arbitrary window regions for more effective use of screen space. *Proc. CHI'04*, 1525-1528.
27. Zellweger, P. T., Mackinlay, J. D., Good, L., Stefik, M., and Baudisch, P. (2003). City lights: contextual views in minimal space. *Proc. CHI'03*, 838-839.
28. Zhai, S., Smith, B., and Selker, T. (1997) Improving Browsing Performance: a study of four input devices for scrolling and pointing tasks. *Proc. IFIP HCI'97*, 286-293.