Wedge: Clutter-Free Visualization of Off-Screen Locations

Sean Gustafson
University of Manitoba
Winnipeg, MB, Canada
sean@gustaf.ca

Patrick Baudisch
Microsoft Research
Redmond, WA, USA
baudisch@microsoft.com

Carl Gutwin
University of Saskatchewan
Saskatoon, SK, Canada
gutwin@cs.usask.ca

Pourang Irani
University of Manitoba
Winnipeg, MB, Canada
irani@cs.umanitoba.ca

ABSTRACT

To overcome display limitations of small-screen devices, researchers have proposed techniques that point users to objects located off-screen. Arrow-based techniques such as *City Lights* convey only direction. *Halo* conveys direction and distance, but is susceptible to clutter resulting from overlapping halos. We present *Wedge*, a visualization technique that conveys direction and distance, yet avoids overlap and clutter. Wedge represents each off-screen location using an acute isosceles triangle: the tip coincides with the off-screen locations, and the two corners are located on-screen. A wedge conveys location awareness primarily by means of its two *legs* pointing towards the target. Wedges avoid overlap programmatically by repelling each other, causing them to rotate until overlap is resolved. As a result, wedges can be applied to numbers and configurations of targets that would lead to clutter if visualized using halos. We report on a user study comparing Wedge and Halo for three off-screen tasks. Participants were significantly more accurate when using Wedge than when using Halo.

Author Keywords: Visualization, peripheral awareness, off-screen, small screens, spatial cognition, maps.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces – Graphical user interfaces.

INTRODUCTION

When viewing large graphical documents on a small screen device, objects of interest are often located off-screen. In the case of a map for example, some of the locations required to plan a route might be invisible. Spatial cognition tasks that are comparably easy when all relevant locations are visible can now become difficult.

Researchers have proposed several techniques for visualizing off-screen objects. Arrow-based techniques such as *City Lights* [12] place dashes or arrows at the edge of the screen to indicate the direction towards an off-screen location. *Halo* [1] surrounds off-screen locations with rings just large enough to intrude onto the screen. Unlike City Lights and other arrow-based techniques, Halo allows users to infer the exact location of off-screen targets, thereby conveying direction and distance. As a result, Halo outperforms arrow-based techniques when distance matters [1].

Halo's performance advantages, however, were tested with no or little overlap between the rings [1]. Larger numbers of targets as well as targets located in the same direction can cause halo arcs to overlap, as acknowledged by Baudisch and Rosenholtz. Figure 1a illustrates how 8 targets—only three more than tested by Baudisch and Rosenholtz—can already lead to substantial clutter. Arcs blend together, reducing the main strength of Halo, which is the pop-out effect of small-radius arcs among less-curved arcs. Note that overlap gets worse if multiple targets are located in the same direction, as commonly happens when the view is panned and off-screen targets gather along an edge or a corner.

Figure 1: (a) The problem: Halo arcs point users to off-screen targets, but overlapping arcs are hard to interpret. (b) *Wedges* point to the same off-screen locations; since they avoid overlap, the display remains intelligible.

In this paper, we present a visualization technique we call *Wedge*. As suggested by the three triangles overlaid onto

Figure 1b, wedges have the shape of an acute isosceles triangle. Each wedge represents an off-screen location: the tip coincides with the off-screen target, and the two corners are located on screen. A wedge conveys location awareness primarily through the two *legs* pointing towards the target. This allows users to triangulate the target location. Wedge therefore offers the same location functionality as Halo. However, each wedge offers two additional degrees of freedom (rotation and aperture) and in combination with the layout algorithm we present in this paper, this allows the wedges to avoid each other and thus overlap and clutter.

In the following sections, we review related literature in visual workspaces and selection techniques. We then present an experimental comparison between Wedge and Halo that found significant benefits for Wedge in terms of error rates and user preference.

RELATED WORK

The Wedge visualization is related to general techniques for showing off-screen content, such as overviews and fisheye views; and also related to off-screen visualization techniques, such as contextual views and Halo. Halo and Wedge are both based on the theory of amodal completion.

Overview+detail views show the workspace in miniature

Overview+detail techniques present a miniature view of the entire workspace in a separate window. The overview window can be displayed next to the detail view or as an overlapping inset window, while the main display shows a zoomed-in view. Users move the detail view either by panning or by dragging a viewfinder in the overview. Zoom lenses such as DragMag [24] may be considered overview+detail technique, except that the larger window shows the overview and the smaller inset window shows detail. The representation of a document in an overview is typically produced by simple geometric zooming; some systems, however, use a special representation designed to preserve and highlight specific features (e.g., [11], see also semantic zooming [3]).

Overview+detail views have been shown to be effective [16], but they impose additional cognitive processing on users by requiring them to reorient themselves when switching between views [2]. Additionally, overview windows require additional space, or, if overlaid onto the detail view, occlude part of the context in the main window.

Focus+context techniques

Focus+context techniques such as fisheye views [18] eliminate the need for multiple windows by presenting a distorted view of the entire workspace. They provide a smooth transition between an enlarged focus region and the surrounding context [5]. The drawback with many focus+context views is that they can make tasks that require targeting or revisitation more difficult [9, 23], and the distortion caused by fisheye views can degrade performance in tasks that have a clear spatial component.

Contextual views are space-efficient fisheye views

Contextual views [12] are generally derived from fisheye techniques. While traditional fisheye views typically convey a distorted, yet complete view of the periphery, contextual views tend to represent only objects of interest, represent these objects using abstract shapes (*proxies*), and then overlay these proxies onto screen space. Consequently, these techniques can only be used if the semantic information about objects and locations is available, which may not always be the case.

Arrows pointing into off-screen space appear in a number of different contexts such as on maps, documents, and more commonly in games, such as Nintendo's 1990 small screen game *Tecmo Bowl*.

City Lights are "space-efficient fisheye techniques" [12]. Unlike its arrow-based predecessors, it also conveys the size of off-screen objects by projecting these objects onto the display window's edge, so that each off-screen object results in a line along the window border. City Lights also offer an abstract and coarse representation of object distance by giving lines one of two colors, each representing a specific distance range.

EdgeRadar [7] extends on City Lights by improving its notion of distance. EdgeRadar reserves a border along the screen edge to represent off-screen space. Replacing City Light's color coding, EdgeRadar represents distances as distances by compressing them proportionally into the border. EdgeRadar was shown to be useful for tracking moving objects [7].

All contextual view techniques have in common that they use a symbolic or distorted representation of distance. In order to interpret distance, users therefore need a legend explaining how distance cues map to actual distance.

Halo conveys location, but not direction and distance

Baudisch and Rosenholtz introduced Halo [1] to improve on the limited distance awareness of arrow-based techniques. Unlike its predecessors, Halo does *not* attempt to convey location by conveying direction and distance—Halo instead conveys location directly using a partially-out-of-the-frame approach, known in cinematography [13]. As a result, Halo's notion of distance is scale-independent and thereby overcomes the need for an explanation of any cue-to-distance mapping.

Using a slightly different set of tasks, Burigat et al. [4] compared Halo to scaled and stretched arrows that encoded distance as size and length of arrows, respectively. They reproduced Baudisch and Rosenholtz's finding that Halo improved performance when precise distance was required, and also found that scaled and stretched arrows were faster and more accurate than Halo in an off-screen target ordering task.

Techniques and applications that use Halo include *Perspective Cursor*, where Halo keeps users aware of mouse pointers traveling between screens [15]. In a study conducted by

Rohs and Essl, participants selected off-screen targets in a one-handed navigation task on handheld devices more efficiently when using Halo than when using zooming [17]. In the commercial world, Halo has been employed to show map objects on PDAs, and in the user interfaces of interactive applications, such as *Second Life*.

Two projects use oval halos to provide users with awareness of off-screen objects. In an experimental system for panning and zooming large trees, Nekrasovski et al. use an oval halo to obtain an aspect ratio more suitable for their screen [16]. *Hop* uses oval halos in an attempt to reduce overlap and clutter [10]. However, the authors suggest that the distortion impacted distance awareness and that oval halos were not sufficiently accurate for locating objects off-screen. We performed a small study to empirically verify this claim and found there was a significant effect for error amount ($t(5)=9.335$, $p=0.001$), with Halo ($M=162.81$, $SD=60.72$) receiving lower error amounts than Ovals ($M=334.84$, $SD=44.00$). Based on these insights, we discarded the oval approach and did not include it in our experimental comparison.

Perceptual theories guiding our design

The theoretical underpinning of both Halo and Wedge is the theory of *amodal perception* or *amodal completion*. It suggests that the human visual system will complete parts of an object even when it is only partially visible [6, 19]. Amodal perception is rooted in evolutionary and ecological adaptation of our visual system and allows humans to recognize partially occluded objects in their environment [20].

The ability to amodally complete objects works also for visual displays. The image in Figure 2a, for example, triggers the perception of a circle occluded by a square.

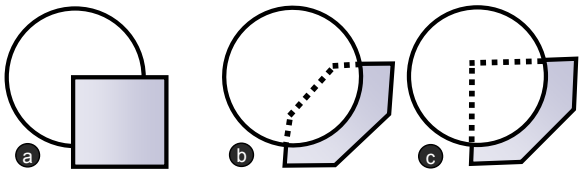

Figure 2: (a) Processes of amodal completion lead the visual system to infer a full circle covered by a square. (b) Figure completion based on *symmetry/regularity*, global process. (c) Figure completion based on *continuation*, local process

Gestaltists attribute object completion to our innate ability to identify complete objects or wholes out of parts [6, 8]. *Global* models hypothesize that the perceptual system tends to adopt the most regular or symmetrical solutions [21] (Figure 2b) and might explain why halos are perceived as complete wholes based on a portion of their visible arcs. *Local* models suggest that the visual system completes the occluded part by connecting the extension of the visible contours [21] (Figure 2c). In these models, good continuation and simplicity are the prevailing principles. The design of Wedge is therefore based on local models.

Several studies show that amodal completion occurs rapidly: from 100 msecs to 300 msecs [14]. Shore and Enns [22] have recently demonstrated that shape completion time also depends on the size of the occluded region. Our design of Wedge reflects this by making the on-screen portion of a wedge sufficiently visible and proportional to the "occluded" off-screen portion.

THE WEDGE

In this section, we describe the design of the Wedge visualization and the three main goals of the design.

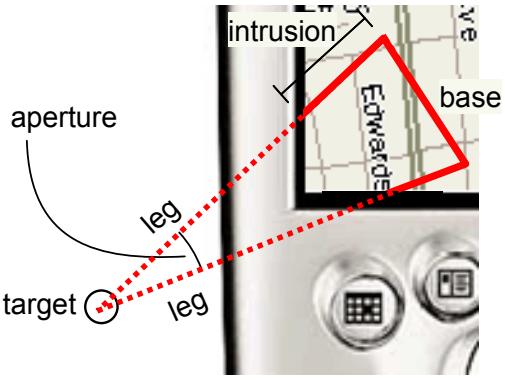

Figure 3: Each wedge consists of two legs and a base. The legs are separated by an angle we call the *aperture.*

As shown in Figure 3, each wedge consists of three line segments: two *legs* of equal length and one terminating line called the *base*. The legs are the key element. In order to locate the off-screen object referred to by a wedge, users visually trace the legs, extrapolate them across the display edge, and estimate where they intersect. The intersection point is the location of the off-screen object.

The base connects the legs, which plays an equally important role. On a screen with multiple wedges it is the bases that allow users to pair up legs, thus preventing users from tracing a pair of legs each of which belonging to a different wedge. To function properly, the bases of two or more wedges should overlap as little as possible; we will explain how we prevent this from happening when we discuss the wedge layout algorithm.

The base may be a straight line as shown in Figure 3 or it can be an arc with its center point located at the off-screen location. Both form factors have benefits and drawbacks. The angles produced by the straight base can serve as an additional cue reinforcing distance. The curved base, in contrast, offers a distance cue by means of circle completion, as introduced by Halo. In the case of a curved base, however, vertex angles do not provide any additional cues, as they are always 90 degrees. In the following sections, we focus on the straight base version; in the user study we used a curved base.

Each wedge offers three degrees of freedom

The key aspect that distinguished Wedge from its predecessors is its three degrees of freedom. As shown in Figure 4, we can change the (a) rotation, (b) aperture, and (c) intrusion of a wedge, while it still points to the same location.

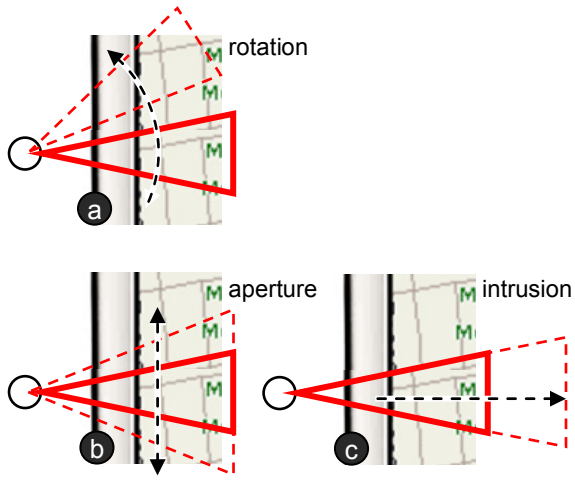

Figure 4: While continuing to point at the same location, each wedge offers three degrees of freedom: (a) rotation, (b) aperture, and (c) intrusion.

The three goals for "spending" degrees of freedom

The three degrees of freedom of each wedge can be used for three different purposes:

1. To avoid overlap with another wedge (see Figure 5). To resolve the overlap of the two wedges, we can either rotate the upper wedge upward (Figure 5b), reduce its aperture (Figure 5c), or reduce its intrusion (Figure 5d).

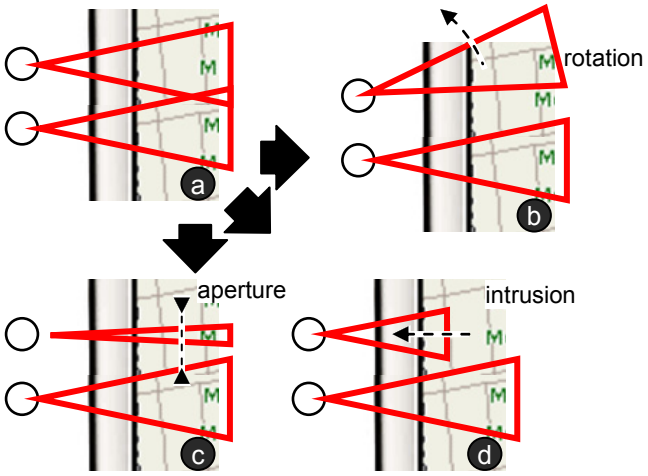

Figure 5: (a) Overlap between two wedges can be resolved by adjusting (b) wedge rotation, (c) aperture, or (d) intrusion.

2. To maximize the location accuracy communicated by the wedge. The goal of each wedge is to allow users to accurately locate the respective off-screen targets. A look at Figure 5 suggests that some of these wedges work better than others. The thin wedge in Figure 5c, for example, might not work as well as the rotated wedge in Figure 5b.

To get a better understanding of this issue and to provide the language for describing the layout algorithm, we introduce the notion of an off-screen *orbital* (we derive the term from chemistry, where a molecular orbital is a region in which an electron may be found in a molecule—basically a space with a probability distribution).

Figure 6 illustrates the concept. While we tend to think of legs as the first part of a *line* pointing towards the off-screen target, there is a certain amount of uncertainty about the angle. As a result, the shape emerging from a leg is not a line, but a cone. We call them *beams*. The intersection of the two beams is where the user would expect to find the off-screen target—this is the *orbital*. In reality beams and orbitals have a fuzzy perimeter, but for the sake of simplicity we illustrate them as solids.

The size of the orbital depends on two factors: *beam spread* and *intersection angle*.

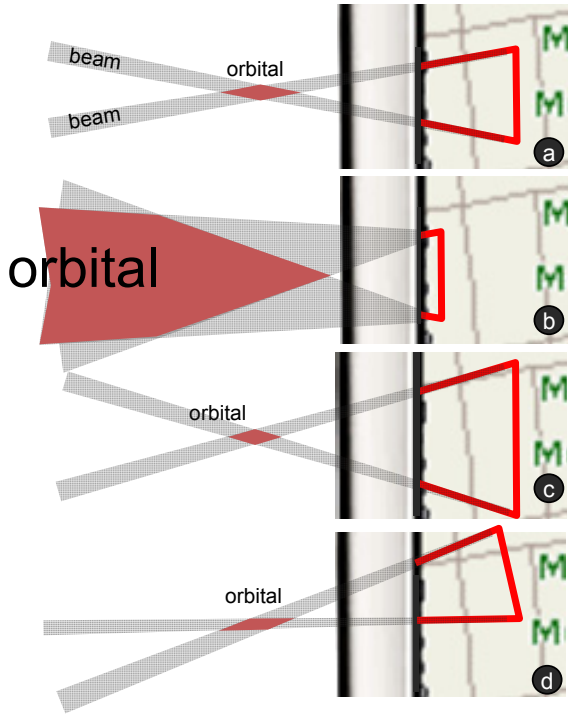

Figure 6: (a) The orbital is the area where users expect the target to be located. Its shape and size are determined by the intersection between the two beams emerging from each leg. Changing (b) intrusion, (c) aperture, and (d) rotation of the wedge affects size and shape of the orbital.

The spread of each beam depends on the length of the leg it emerges from. The same way that a rifle fires more accurately than a pistol, long legs (Figure 6a) resulting from deeper intrusion result in thinner beams than the shorter legs of a wedge with shallow intrusion (Figure 6b). Note that the orbital of a wedge is infinite if the outside edges of the two beams diverge, as is the case in Figure 6b. In this case a wedge provides users with an estimate of the minimum target distance, but not with an estimate for the maximum distance.

The angle under which the two beams intersect depends on the aperture separating the wedge legs. Increasing the aperture of a wedge (Figure 6c) generally leads to a larger angle, resulting in a shorter orbital.

Rotating a wedge decreases the spread of one beam at the expense of increasing the spread of the other. This results in a skewed orbital (Figure 6d).

3. To serve as an additional cue or proxy for distance. Any of the three degrees of freedom can also be used to communicate a certain target property by convention, similar to how City Lights uses two colors to distinguish between two distance ranges. An easily-interpreted distance cue is important for tasks not involving absolute distances, such as determining which of two targets is closer.

The primary goal of wedge design is to achieve maximum accuracy, which means to minimize orbital size. On the other hand, it is clear that overlap and clutter have a huge affect on the readability of wedges, ultimately impacting accuracy more than any other factor. When designing the wedge layout algorithm we therefore prioritize as follows: (1) avoiding overlap, (2) maximizing location accuracy, and (3) providing an additional distance cue. This prioritization is not strict, since we must still provide a balance of the three goals described above.

THE WEDGE LAYOUT ALGORITHM

The layout algorithm determines an acceptable layout of wedges by manipulating intrusion depth, wedge aperture, and rotation, as described in the following sections.

Intrusion

We considered three primary options for mapping distance to intrusion: (a) constant intrusion, (b) shorter intrusion for longer distances, and (c) longer intrusion for longer distances (Figure 7). While constant intrusion led to increased overlap between wedge outlines (Figure 7a), the other two mappings naturally reduced overlap. These two mappings also had the potential to serve as an additional distance cue.

Figure 7: Two overlapping wedges shown in each of the three options for leg length: (a) constant leg length; (b) longer legs for closer targets; (c) shorter legs for closer targets.

We chose the directly proportional mapping (Figure 7c), because it minimized orbital sizes (it avoids cases of the type shown in (Figure 6b) and therefore increased accuracy, as explained in the previous section. As a positive side effect, it allowed us to nest wedges in some cases as shown in Figure 7c.

We calculate wedge intrusion using a non-linear function that gradually levels off for very distant objects. This ensured that the intrusion depth did not grow in an unbounded fashion, which would occlude the main display. We used a logarithmic function to control leg length:

$$leg = dist + \ln\left(\frac{(dist + 20)}{12}\right) \times 10$$

where *leg* is the length of each leg in pixels and *dist* is the distance in pixels between target and the edge of the screen. The additive constant of 20 pixels assures a minimum intrusion for close targets.

Aperture

The aperture of each wedge is mapped linearly to target distance using the following linear function:

$$aperture = (5 + dist \times 0.3) / leg$$

where *aperture* is the angle (in radians) separating the legs (Figure 3), *dist* is the distance of the target to the edge of the screen (in pixels), and *leg* is the overall length of each leg (in pixels). This mapping serves as the primary cue for target distance. The constants were the result of several pilot studies and balance orbital size and risk of overlap: larger apertures would have led to smaller orbitals, but at the expense of a significantly increased risk of overlap.

Corners

Screen corners have traditionally been a challenge for all contextual views because they represent a large proportion of off-screen space [1, 12]. At the same time, they offer less space for the proxy representing the target. The halo arc in the bottom left corner of Figure 1, for example, is cropped, reducing its accuracy substantially [1].

The additional degrees of freedom offered by wedges help alleviate this problem, yet this case still requires additional attention. Wedges for extremely distant objects would be cropped by the edges of the screen when displayed in the corners. To alleviate this, the leg length function increases the legs only up to 20 pixels. If the new intrusion was still insufficient to show wedge legs, the algorithm would decrease the aperture to the point of making the wedge fit in the corner. As a result of this, wedges always showed legs in the corners. The distances used in the experiment were rarely large enough for this to occur.

Rotation

Rotation is the primary means of avoiding wedge overlap. Figure 8 shows a cluster of wedges before and after resolving overlap using rotation. Fortunately, rotation has little impact on intrusion and aperture, so it does not affect the distance cues conveyed by intrusion and aperture.

Rotation is computed using a simple iterative algorithm. It is computationally inexpensive and offers real-time performance for maps with up five overlapping wedges.

Initially, wedges located along a screen edge are placed perpendicular to that edge. If a wedge is near a corner, the algorithm places it such that there is an equal amount of on-screen space on either side of the wedge. Next, the algorithm iterates to resolve overlap. In each step, the algorithm traverses all wedges on screen in clockwise order (according to the location of their base centers, not by target loca-

tion). The algorithm rotates wedges away, by a small amount, from neighbors with which it overlaps. This can propagate overlap to neighboring wedges and is resolved through repetition as shown in Figure 9. If there is no solution the algorithm will terminate after a fixed number of iterations, leaving wedges with as little overlap as possible.

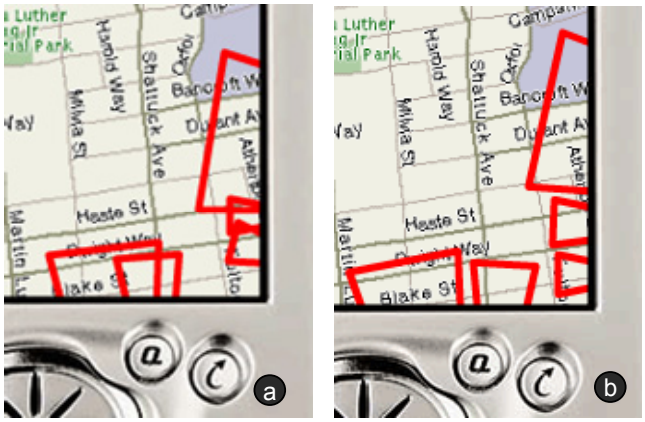

Figure 8: A cluster of wedges (a) before and (b) after applying the wedge overlap algorithm.

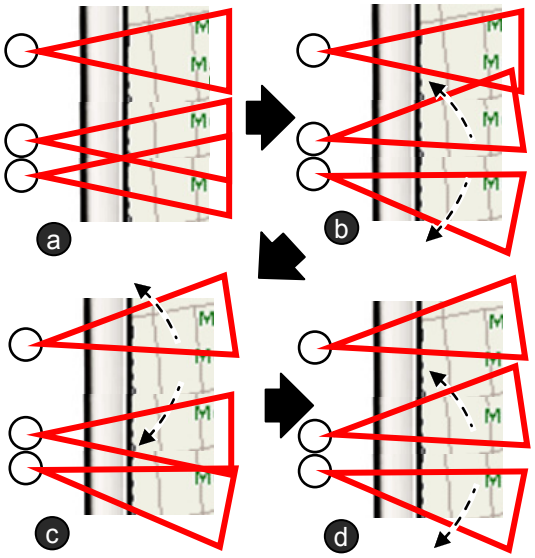

Figure 9: Iterations of the wedge layout algorithm: (a) Two wedges overlap, (b) they are rotated away causing new overlap, (c) new overlap is resolved, creating yet another overlap, and (d) all overlap is avoided.

USER STUDY

The objective of the study was to compare the effectiveness of Wedge with the commonly used Halo. We hypothesized that Wedge will be more accurate than Halo, primarily when they represent objects that get mapped to the corner of the display. We were also interested in identifying the effects of each of these techniques in high density layouts.

Participants

18 participants (3 female, 15 male) between the ages of 18 and 30 were recruited from an undergraduate computer science program. Participants were given course credit in exchange for their participation. None of the participants were familiar with either Halo or Wedge, and all had normal or corrected-to-normal vision. After the data was collected, two participants were removed due to extremely high error rates in all conditions, leaving 16 participants in the analysis.

Apparatus and Materials

The experiment was conducted using custom software written in Adobe Flash that simulated a handheld PDA. A simulated PDA screen was shown in the center of a 19" monitor at slightly larger than real-life size. Participants interacted with the simulated PDA via a standard mouse interface.

Experimental Conditions

Visualization

We compared performance with two types of visualization: *Halo* and *Wedge*. Halo was implemented using the original code written by Baudisch and Rosenholtz [1]. Wedge was implemented exactly as described above except that each wedge base was curved (as shown in Figure 10) instead of straight.

We controlled the total on-screen line length between the two conditions. This was done by choosing functions for wedge aperture and leg length such that the overall average on-screen line length for every target used in the study were equal (75 pixels for both Halo and Wedge).

Density

To explore whether overlap affects the visualizations, we tested two different organizations of targets. In *sparse* conditions, the targets were organized such that there were minimal overlapping halos. In *dense* conditions, the five targets were organized so that all of the on-screen visualizations were packed into a smaller area. The sparse conditions were programmatically converted to dense conditions by *folding* the display (once for Avoid and Closest tasks, twice for Locate task) such that each target was placed onto the same side of the display at the exact position as they were on the other side. As a result, the dense condition simulated the amount of clutter that would be equivalent to 20 (Locate task) or 10 (other tasks) off-screen objects. This procedure ensured that sparse and dense maps were comparable.

Position: Corners vs. Sides of the display

A second issue that can also affect clutter and density is whether or not the on-screen visualizations are placed in the *corner* or on the *side* of the screen. As discussed above, corners provide less space for halos, and cause additional arc overlap.

Tasks

The study used three tasks from Baudisch and Rosenholtz's [1] comparison of Halo to arrows. The tasks are illustrated in Figure 10. The Locate task directly assessed the accuracy of each visualization, while the two other tasks were secondary tasks and looked at how the visualizations could be used in realistic problem solving.

Locate. The users clicked in the off-screen space at the expected location of the off-screen targets indicated by each of the two visualizations. Users located targets in any order and the system automatically picked the closest match.

Avoid. As "ambulance dispatcher," the user was to select the hospital farthest from traffic jams. Each map contained indicators of five on- or off-screen traffic jams, and three blue cross-shaped icons representing hospitals.

Closest. Each map contained a blue car icon and five red wedges/arcs representing restaurants. The user's task was to click on the halo/wedge corresponding to the off-screen location closest to the car.

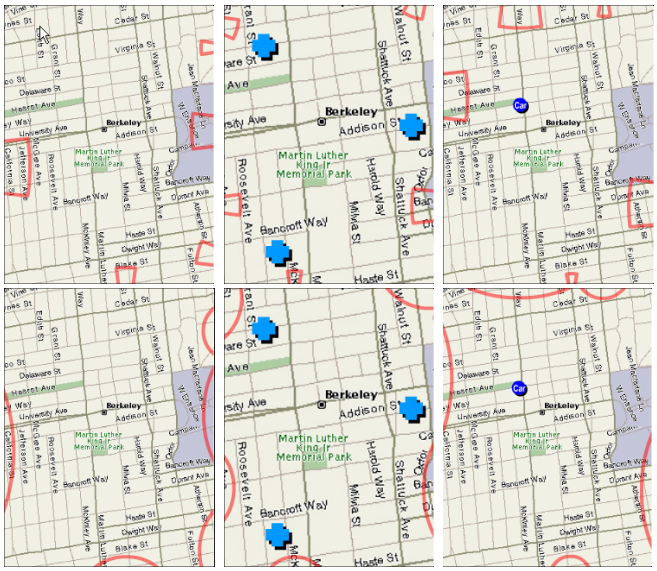

Figure 10. Experimental tasks, showing PDA screen only, with Wedge (above) and Halo (below).

Procedure and Design

The study used a 2x2x2 factorial design with three factors:

- Visualization (*Halo* or *Wedge*)
- Density (*dense* or *sparse* target clustering)
- Position (*corner* or *side* of the screen)

Participants were shown the study system and were given a brief demonstration of both Halo and Wedge. They then carried out the three tasks: there were four training maps at the start of the task, and then 16 (for Locate) or 32 (for Closest, Avoid) test maps. The order of tasks and display conditions were fully counterbalanced. After the session, participants were asked to state which visualization type they preferred for each task. The study system collected error and completion time data.

RESULTS

We organize our results by the three tasks carried out in the study. Within each task we consider the effects of each of the three factors (visualization type, density, and position) on error and completion time. Note that in all analyses, subject was included as a random factor. Tables 1 and 2 show summary means for all measures and tasks.

	Halo	Wedge
Locate	45.3 pixels (19.0)	35.6 pixels (18.9)
Avoid	32.6% (18.6%)	28.3% (22.4%)
Closest	38.3% (21.2%)	34.6% (21.1%)

Table 1. Summary of error results (s.d.).

	Halo	Wedge
Locate	2.64 sec (1.70)	2.39 sec (1.10)
Avoid	4.69 sec (2.71)	4.54 sec (2.16)
Closest	5.35 sec (4.79)	5.45 sec (4.12)

Table 2. Summary of completion times (s.d.).

Task 1: Locate the off-screen location

The first task asked participants to click on the locations of the off-screen objects indicated by each wedge or halo on the screen. We gathered data about error amount and completion time to locate each of the five targets. The error amount was the Euclidian distance from the guessed position to the target's position.

Locate Task: Error Amount

Figure 11 shows the error amounts for Halo and Wedge in all conditions (dense and sparse; corner and side). We carried out a 2x2x2 ANOVA (Visualization x Density x Position) to test for differences. We found main effects of all three factors: for Visualization, $F_{1,15}=5.86$, $p=0.029$; for Density, $F_{1,15}=6.76$, $p=0.02$; for Position, $F_{1,15}=121.39$, $p<0.001$.

As can be seen from Figure 11 larger errors were seen in corner trials (mean 51 pixels) than in side trials (mean 30 pixels). There were also larger errors in dense configurations (mean 43) than sparse configurations (mean 38). The overall difference between visualizations was about 10 pixels (Halo mean 45.3 pixels; Wedge mean 35.6 pixels).

In addition, there was a significant interaction between Visualization and Position ($F_{1,15}=15.36$, $p=0.001$). As shown in Figure 11, the difference between visualization types is considerably larger in corners than on the sides of the screen, which supports our hypothesis that the reduced space in corners causes additional problems for Halo interpretation. There was no interaction between Visualization and Density ($F_{1,15}=0.67$, $p=0.43$).

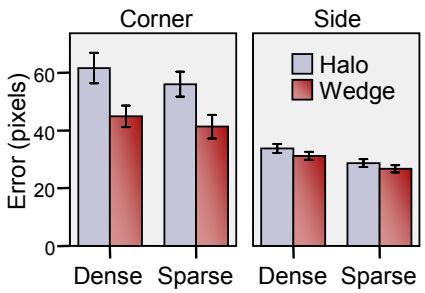

Figure 11. Locate task mean error amount by visualization type, density, and position. Error bars indicate standard error.

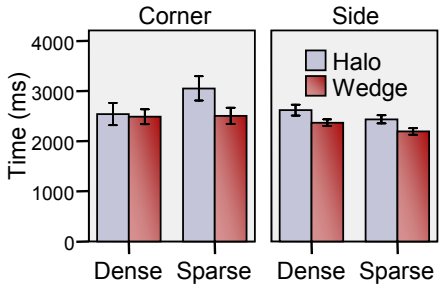

Figure 12. Locate task mean completion time. Error bars indicate standard error.

Locate Task: Completion Time

We also tested for differences in time to complete the task (see Figure 12). A 2x2x2 ANOVA showed no significant effects of Visualization ($F_{1,15}$=1.20, p=0.29), Position ($F_{1,15}$=4.54, p=0.05), or Density ($F_{1,15}$=0.65, p=0.43). There was a significant interaction between Density and Position ($F_{1,15}$=5.97, p=0.027), but no interactions with Visualization.

Task 2: Avoid the traffic jam

The second task asked participants to select one of three on-screen objects that was furthest from a set of off-screen objects. We gathered error rate and completion time data.

Avoid Task: Error Rate

Figure 13 shows error rates for the different visualizations, densities, and positions. A 2x2x2 ANOVA did not show any effects of Visualization ($F_{1,15}$=2.55, p=0.13), Position ($F_{1,15}$=2.38, p=0.14), or Density ($F_{1,15}$=0.58, p=0.46). In addition, there were no interactions between any factors.

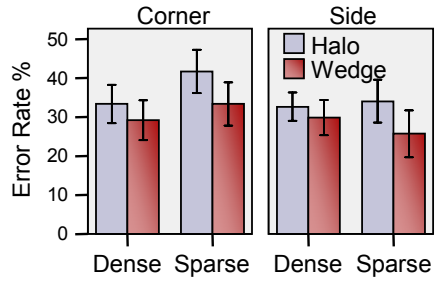

Figure 13. Avoid task mean error rate. Error bars indicate standard error.

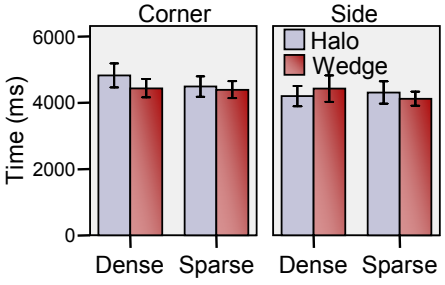

Figure 14. Avoid task mean completion time. Error bars indicate standard error.

Avoid Task: Completion Time

A 2x2x2 ANOVA showed no effects of any of the three factors on task completion time (Visualization $F_{1,15}$=0.18, p=0.68; Density $F_{1,15}$=2.09, p=0.17; Position $F_{1,15}$=1.58, p=0.23), and no interactions between any factors.

Task 3: Find the closest restaurant

The third task asked participants to select the closest off-screen object to an on-screen icon. Again, we gathered error rate and completion time data.

Closest Task: Error Rate

Figure 15 shows error rates for the different visualizations, densities, and positions. A 2x2x2 ANOVA showed significant main effects of Position ($F_{1,15}$=76.6, p<0.001), but not of Visualization ($F_{1,15}$=1.24, p=0.28) or Density ($F_{1,15}$=0.12, p=0.73). There was a significant interaction between Density and Position ($F_{1,15}$=7.33, p=0.016), but no interactions with Visualization.

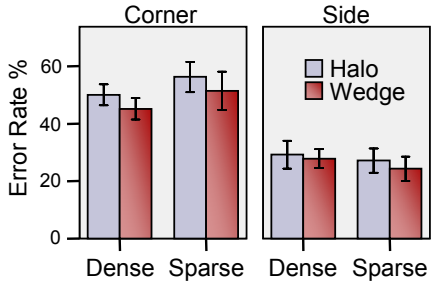

Figure 15. Closest task mean error rate. Error bars indicate standard error.

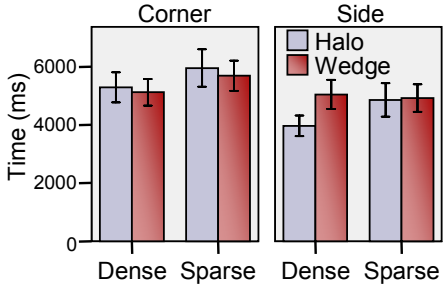

Figure 16. Closest task mean completion time. Error bars indicate standard error.

Closest Task: Completion Time

A 2x2x2 ANOVA showed significant main effects of Position ($F_{1,15}$=5.24, p=0.037), but did not show effects of Visualization ($F_{1,15}$=0.10, p=0.76) or Density ($F_{1,15}$=2.89, p=0.11). There was, however, a significant interaction between Visualization and Density ($F_{1,15}$=6.60, p=0.021).

Overall Preferences

After the experiment the participants were asked to state which visualization they preferred for each task. Table 3 summarizes the subjective preferences. In the Locate and Avoid task there was a clear preference for Wedge; in the Closest task, however, Halo was preferred.

	Wedge	Halo	No Preference
Locate	10	5	1
Avoid	10	5	1
Closest	6	8	2

Table 3: The number of participants who preferred each visualization technique for the three tasks.

Participant Comments

Comments made during the trial suggested reasons for the advantages for Wedge over Halo. One user said, "I found that when the rings overlap it is almost impossible to tell which is the right ring. Wedges just seem natural." And another stated, "overlapping rings made it very confusing at times. Directional wedges helped a lot, and they also seem to take up less space. More information meant less thinking with the wedges."

Participant's comments also provided some insight into the reasons why Halo was preferred for the Closest task – that the difference between distant and close off-screen objects was easier to determine with Halo, since there is a large visual difference in this case. One participant stated that, "the sizes of the arcs did not require too much calculation or thinking to spot the smallest ring."

DISCUSSION

Our hypotheses were that Wedge would be more accurate than Halo, and that this effect would be stronger in corners. The Locate task provides evidence in support of both hypotheses: accuracy with Wedge was significantly higher than Halo, and the difference was larger in corners. We did not find any interaction between Visualization and Density for the Locate task, however; it appears that people can successfully tease out the level of overlap seen in our tasks, although we plan to study higher levels of clutter in future studies.

In the Avoid and Closest tasks where people had to make use of this accuracy, we did not find significant differences between Wedge and Halo (even though Wedge was on average 13% and 10% more accurate in these tasks). Part of the reason for the lack of difference is that these tasks involved strategy more than the Locate task; therefore, it is possible that strategy choice overshadowed the beneficial effects of Wedge that were seen in the Locate task. In addition, the Closest task revealed an advantage for Halo (the large visual difference between distant and close objects) that we had not considered. Subjective results reinforce these findings – Wedge was strongly preferred for the Locate and Avoid tasks, in which Halo has several problems and few advantages.

Overall, our results confirmed our hypotheses and show the benefits of the new visualization. These benefits are more pronounced when off-screen objects are clustered into corners, where wedges allow users to triangulate the location of off-screen objects more precisely. We believe that Wedge's overlap-avoidance algorithm aids users in determining direction and distance. While we chose a brute-force approach for the layout algorithm, we will soon look into using proper optimization techniques to fit and lay out the wedges in the limited display space.

To successfully complete these tasks, it appears that participants employ different strategies. It is clear that for the Locate task, participants are extrapolating the full shape of the wedge and halo to locate the off-screen object. In this task, we reason that the visual shape of the wedge more clearly shows the shape completion process needed to perform the Locate task. In the case of the Avoid and Closest tasks, users have to rely primarily on distance cues. As we see from our results, the distance cues in Wedge are as good as those provided by Halo, and in some cases even better. In our algorithms we maintained the aperture of each wedge directly proportional to the distance of the target. In future work we need to investigate the effects of variable aperture size and intrusion depths, particularly for objects in the corners. We will also look at new designs that can better show large differences in distance, as Halo was able to do in the Closest task.

Based on the results of our study, we propose the following recommendations to designers:

- *Use Wedge*. Off-screen object information should be displayed using Wedge as the primary visualization technique: it offers significant improvements over Halo.
- *Reduce overlap*. Designers should reduce overlap in any visualization of off-screen objects, as overlap leads to reduced accuracy and greater difficulty identifying objects.
- *Rotation is better than overlap*. None of our participants were concerned about the rotation of the wedges, although several comments were received about the difficulty of the overlapping Halos. Therefore, we believe that rotation should be chosen over either cropping or overlap for off-screen visualizations.
- *Corners need special attention.* Our results confirm that designers need to pay significant attention to the design of off-screen visualizations so that they work equally well in the display corners.
- *Striking a balance.* Designers need to strike a fine balance in selecting parameters for off-screen visualizations to avoid as much overlap as possible, maximize location accuracy and to serve as a cue to distance.

CONCLUSIONS

We introduced a new off-screen visualization technique, Wedge, which reduces the amount of overlap on the display. We investigated the design space for this new technique; Wedge optimizes three valuable design principles that aid users in reducing interpretation costs and increasing accuracy. The wedge layout algorithm was designed to strike a balance between multiple factors; i.e., avoid overlap, provide accurate location information, and provide good distance cues. Wedge reduces clutter and is less prone to problems of corner-based clustering.

We carried out a study that showed significant accuracy advantages for the Wedge over the Halo; in addition, we found that Halo and Wedge provide equally good cues to

distance information. In future iterations of Wedge we will augment distance cues, and we will test the visualization with higher levels of clutter and other realistic tasks. Overall, our results indicate that Wedge is a simple but effective off-screen visualization technique that can enhance the utility of any application that relies on it.

ACKNOWLEDGEMENTS

We thank all of the study participants for their efforts, Colin Ouellette for administering the many pilot studies, and the reviewers for their valuable comments. This research was partially funded by NSERC.

REFERENCES

1. Baudisch, P. and Rosenholtz, R. (2003). Halo: A technique for visualizing off-screen locations. *Proc. CHI 2003*, 481–488.
2. 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 2002*, 259–266.
3. Bederson, B. B., Hollan, J. D., Perlin, K., Meyer, J., Bacon, D., and Furnas, G. (1996). Pad++: A zoomable graphical sketchpad for exploring alternate interface physics*, Visual Languages and Computation*, 7, 3–31.
4. Burigat S., Chittaro L., and Gabrielli S. (2006). Visualizing locations of off-screen objects on mobile devices: A comparative evaluation of three approaches, *Proc. MobileHCI 2006*, 239-246.
5. Carpendale, M. S. T. and Montagnese, C. (2001). A framework for unifying presentation space. *Proc. UIST 2001*, 61–70.
6. Elder, J. and Zucker, S. (1993). The effect of contour closure on the rapid discrimination of two-dimensional shapes. *Vision Research*, 33(7), 981–991.
7. Gustafson, S. and Irani, P. (2007). Comparing visualizations for tracking off-screen moving targets. *Proc. CHI 2007,* 2399–2404.
8. Guttman, S. E., and Kellman, P. J. (2002). Do spatial factors influence the microgenesis of illusory contours? *Journal of Vision*, 2, 355a.
9. Hornbæk, K. and Frøkjær, E. (2001). Reading of electronic documents: the usability of linear, fisheye, and overview+detail interfaces. *Proc. CHI 2001*, 293–300.
10. Irani, P., Gutwin, C., and Yang, X. D. (2006). Improving selection of off-screen targets with hopping. *Proc. CHI 2006,* 299–308.
11. Lam, H. and Baudisch, P. (2005). Summary Thumbnails: Readable overviews for small screen web browsers. *Proc. CHI 2005*, 681–690.
12. Mackinlay, J. D., Good, L., Zellweger, P. T., Stefik, M., and Baudisch, P. (2003). City Lights: Contextual views in minimal space. *Proc. CHI 2003*, 838–839.
13. Marsh, T. and Wright, P. (2000) Using cinematography conventions to inform guidelines for the design and evaluation of virtual off-screen space. *Proc. AAAI 2000 Spring Symp. Ser. Smart Graphics*, 123–127.
14. Murray, R., Sekuler, A., and Bennett, P. (2001). Time course of amodal completion revealed by a shape discrimination task. *Psychonomic Bulletin*, 8, 713–720.
15. Nacenta, M., Subramanian, S., Sallam, S., Champoux, B., and Gutwin, C. (2006). Perspective Cursor: Perspective-based interaction for multi-display environments. *Proc. CHI 2006,* 289–298.
16. Nekrasovski, D., Bodnar, A., McGrenere, J., Guimbretière, F., and Munzner, T. (2006). An evaluation of pan & zoom and rubber sheet navigation with and without an overview. *Proc. CHI 2006*, 11–20.
17. Rohs, M. and Essl, G. (2006). Which one is better? – Information navigation techniques for spatially aware handheld displays. *Proc. ICMI 2006,* 100–107.
18. Sarkar, M. and Brown, M. (1992). Graphical fisheye views of graphs. *Proc. CHI 1992*, 83–91.
19. Sekuler, A. and Murray, R. (2001). Amodal completion: A case studying grouping. In T. Shipley & P. Kellman (Eds.), *From Fragments to Objects: Segmentation and Grouping in Vision,* New York: Elsevier, 265–293.
20. Sekuler, A. and Palmer, S. (1992). Perception of partly occluded objects: A microgenetic analysis. *Journal of Experimental Psychology: General,* 121, 95–111.
21. Sekuler, A., Palmer, S., and Flynn, C. (1994). Local and global processes in visual completion. *Psychological Science,* 5, 260–267.
22. Shore, D. and Enns, T. (1997). Shape completion time depends on the size of the occluded region. *Experimental Psychology: Human Perception and Performance,* 23, 980–998.
23. Skopik, A. and Gutwin, C. (2005). Improving revisitation in fisheye views with visit wear. *Proc. CHI 2005*, 771–780.
24. Ware, C. and Lewis, M. (1995). The DragMag image magnifier. *Proc. CHI 1995*, 407–408.