LayerFish: Bimanual Layering with a Fisheye In-Place

Andrew M. Webb[1], **Andruid Kerne**[1], **Zach Brown**[1], **Jun-Hyun Kim**[2], **Elizabeth Kellogg**[1]
[1]Interface Ecology Lab, Deptartment of Computer Science and Engineering
[2]Department of Landscape Architecture and Urban Planning
Texas A&M University, College Station, Texas, United States
{andrew, andruid, zach, elizabeth}@ecologylab.net, jhkim@arch.tamu.edu

ABSTRACT
We introduce, LayerFish, a bimanual interaction technique for layering overlapping content in large 2D design spaces. Designers compose visual elements, often producing overlap and requiring layering in the z-dimension. A common approach in design tools is to provide a scene index, as an ordered list of layered elements. Scene indexes become difficult, in terms of physical effort and human cognition, to deal with when working with hundreds of elements. LayerFish renders an in-place scene index as a fisheye to reduce demands on effort and attention. Bimanual gestures support selection, scrolling, and manipulation. We evaluated layering task performance with LayerFish in comparison to a traditional scene index. Findings indicate that the fisheye reduces time to find an element and selection reduces layering time when elements do not heavily overlap.

Author Keywords
visual design; scene index; pen+touch; bimanual interaction

ACM Classification Keywords
H.5.2. Information Interfaces and Presentation: User Interfaces - Graphical User Interfaces

INTRODUCTION
Designers work with many visual elements, from 10s to 100s, in large 2D spaces. Elements overlap, requiring reordering of the visual stack, or *layering*, to place some elements in front of or behind others. As the number of overlapping elements increases in scale, the complexity of layering also increases. This requires more expressive forms of interaction than the common "bring to front" and "send to back" contextual menu options provided in tools such as Powerpoint.

Design tools, such as Adobe Illustrator, support complex layering through a *scene index*—an ordered list showing the visual stacking order of elements. Each element in the 2D design space is represented in the scene index by a *correspondent* thumbnail. Correspondents can be reordered to adjust the visual stacking of elements. They can also be selected to constrain direct manipulation to only corresponding elements, even those occluded by other elements. As the number of elements increases, the scene index must afford scrolling. Scrolling becomes tedious when working with hundreds of elements. Further, the index is often located out of the user's visual focus, requiring her to split attention between the design space and the index.

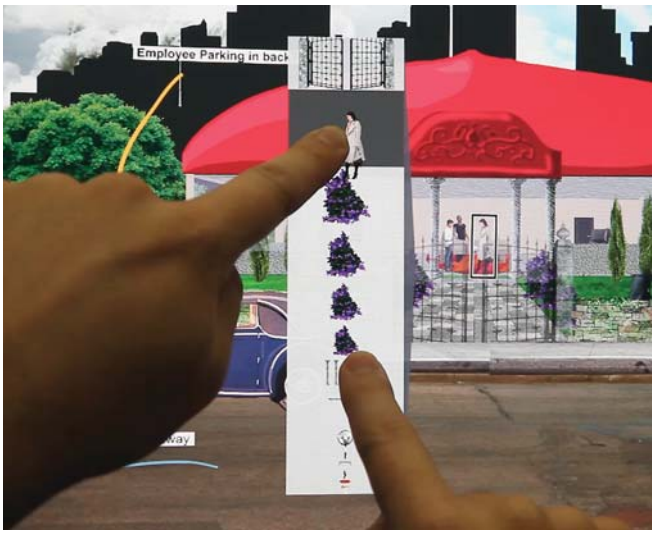

Figure 1. Example of bimanual interaction in LayerFish. The left hand touch has a correspondent selected and held in place, while the right hand drags to scroll the fisheye scene index.

Prior work developed alternative techniques that provide *in-place* layering, addressing the split attention shortcomings of the scene index [11, 19]. However, these techniques are designed for working with a small number of elements, and will not scale well in design spaces with hundreds of elements.

LayerFish is a pen+touch interaction technique to support layering and manipulating overlapping content in a 2D design space on desktop and large collaborative surfaces (see Figure 1). LayerFish addresses issues of scene index scale and split attention through a fisheye visualization [6] and in-place positioning of the scene index at the user's point of focus. The fisheye distorts the visual space, decreasing the sizes of correspondents away from a focus element, and so enables more to be visible than a typical scene index. However, interaction issues arise, as the spatial distortion disrupts layering and scrolling operations. We develop bimanual techniques to address these issues, which keep the focus element fixed while the scene index reorders layers and scrolls. We hypoth-

ISS'16, November 6–9, 2016, Niagara Falls, ON, Canada.

ACM ISBN 978-1-4503-4248-3/16/11 ...$15.00
DOI: http://dx.doi.org/10.1145/2992154.2992171

esize that the fisheye will reduce time to find desired correspondents, since the spatial distortion would help scrolling over large distances.

This paper begins with a discussion of prior work. Next, we present LayerFish. We follow with an evaluation comparing performance of LayerFish—when working with a large number of elements—to a traditional scene index. We discuss findings and derive implications for design.

RELATED WORK

Various layering techniques exist. Commercial design tools typically use either pop-up menus or scene indexes. HCI researchers have developed alternatives, investigating how spatial arrangement [19, 7, 10, 17] and sensing modalities [4, 11] can support interacting with dense and occluded content.

Commercial Design Tools

Many commercial design tools employ a contextual menu (e.g. Microsoft Powerpoint). The contextual menu includes several commands, not all of which are for accessing occluded content. In mouse-based interfaces, the contextual menu is activated by right clicking an element. The user must navigate a hierarchical menu and find the command to bring an element forward or backward. In touch-based interfaces, the contextual menu is activated by touching an element. Commands appear in a menu bar. The contextual menu approach requires the user to perform input on an element, which may be occluded, making it challenging to activate.

Alternatively, a scene index provides an ordered list of visual layers, which can be rearranged to adjust the z-ordering of elements. As the number of layers grows, the index becomes large. It can become difficult to find a layer. Thumbnails for each layer are used to support recognition. The thumbnails are all the same size, making it difficult to differentiate similar elements of different sizes [19]. LayerFish addresses issues of scale through a fisheye visualization and user selection.

HCI Researchers

Ramos et al. explored techniques that maintain the shape and size of elements, while creating spatial separation between layers, enabling novice users to interact with occluded content [19]. LayerFish addresses issues of scale while also maintaining element shape.

Herrlich et al. developed a touch technique for selecting occluded content [10]. The user resolves selection ambiguity by touching proxies arranged radially around the point of interaction. Handle Flags is an in-place technique for selecting overlapping ink strokes without complex lasso selections [7]. For LayerFish, ther user needs to not only select the element to be layered, but also nearby element(s) that it will be layered above or below. Therefore, we use lasso for selecting a region rather than one of these precise selection approaches.

Pen and multi-touch sensing technologies support additional input parameters that can be used to access and manipulate occluded content. Davidson and Han used touch pressure and points of contact to define layering gestures [4]. Hinckley et al. combined touch to select an element with pen tilt to reveal occluded elements [11]. These techniques are designed for working with a few overlapping elements.

Leithinger and Haller address occlusion issues of context menus on cluttered tabletops through user-drawn arrangements of menu items [17]. LayerFish's in-place scene index raises occlusion issues that could be mitigated through user-drawn arrangement. However, new interaction issues arise as scrolling the scene index and re-ordering correspondents involves two dimensions instead of one.

LAYERFISH

LayerFish is a bimanual interaction technique designed for desktop and large surfaces, including tabletops, where both hands are free to interact on the surface. We found, through discussions with visual designers, that these form-factors are commonly used when working with many layers. Smaller form factors are also used, but tended to involve fewer layers.

The LayerFish technique consists of a selection and activation gesture, a fisheye scene index, and interactions for layering elements and manipulating occluded content. For brevity and simplicity, we will describe our technique in terms of a right-handed user, where the *right* hand is their *preferred* hand and the *left* hand is their *non-preferred* hand. For left-handed users, interactions with LayerFish are mirrored, so that roles of preferred and non-preferred hand are consistent.

Selection and Activation

Using LayerFish begins with selecting elements on which to operate, followed by an activation gesture. Selection specifies the elements active in LayerFish. Unlike a traditional scene index that contains all elements, selection enables the user to work with a subset of elements, and so addresses scaling issues for human attention. To distinguish between normal selection for manipulating a group of elements versus selection for LayerFish, we use an activation gesture. LayerFish supports two bimanual input modalities for selection and activation, pen+touch and multi-touch.

Pen+Touch

When using the pen, LayerFish supports lasso selection of elements. LayerFish is activated with a Lasso'n'Cross gesture [1]. A lasso selection with the pen is followed by a vertical slash across the center of the circled selection (see Figure 2). We adhere to Hinckley et al.'s recommendation that the pen by itself creates ink, but when combined with touch performs alternative operations [12]. In order to use the pen as a selection tool, a modifier is needed. We provide an edge-constrained chorded gesture area for the left hand, called the *Hotpad*. Edge-constrained gestures with the non-preferred hand support efficient modal switching [9]. The Hotpad appears in the bottom corner on the left-hand side. Holding one finger down on the Hotpad modifies the pen to function as a lasso selection tool. After the user performs a Lasso'n'Cross gesture with a vertical slice, LayerFish's fisheye scene index appears adjacent to the bounding box of the selection area.

Multi-touch

Touch input is less precise than pen due to the fat finger problem [13]. We suspected this imprecision would make lasso

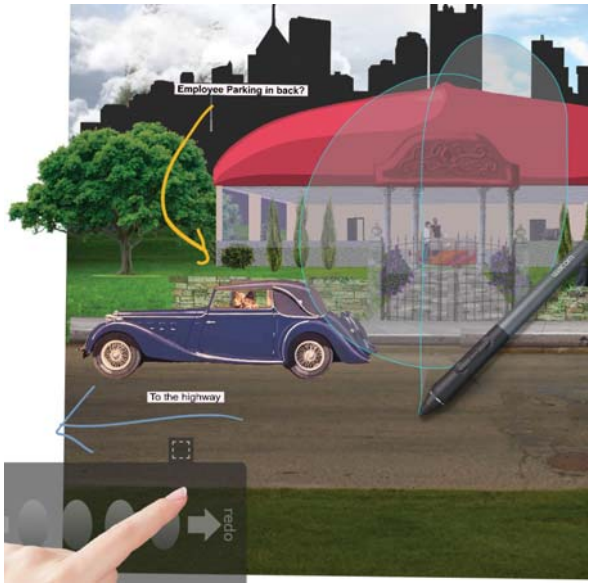

Figure 2. While holding a left-hand touch on the Hotpad, the user performs a Lasso'n'Cross gesture with a pen, selecting elements with a lasso, followed by a vertical slash to activate LayerFish.

selection involving dense overlapping content difficult with touch. We instead support rectangular selection with touch. Again, a left hand touch on the HotPad defines right-hand actions as selection. With a right-hand touch, the user drags a rectangular selection area that defines what elements LayerFish will select. Once the desired selection area is defined, with the left hand touch still down, the user lifts up the right hand and makes a vertical slash through the rectangular selection. The slash activates LayerFish's fisheye scene index with the elements within the selection bounds.

Fisheye Scene Index of Layered Elements

Upon activation, correspondents, representing the subset of selected elements, appear in an ordered list. Correspondents are ordered from top-most selected element to bottom-most.

Fisheye Visualization

A fisheye visualization is used to represent the list of correspondents (see Figure 3). Fisheyes are a focus+context technique that use spatial distortion to give larger visual emphasize to a focus while providing a smaller peripheral view of contextual details. In the case of LayerFish, the fisheye allows the user to see more correspondents at once than with a traditional scene index. The spatial distortion also makes the effect of scrolling non-linear, enabling direct-touch scrolling over larger distances than with a traditional scene index.

The fisheye visualization defines a focus, initially the top correspondent in LayerFish. The focus is given a fixed size (120 x 80 pixels at 94 dpi). Correspondents get progressively smaller as you move away from the focus. Pressing down with pen or touch on a correspondent makes it the focus of the fisheye. Through pen or touch drags, the scene index is scrolled, keeping the focus positioned under the drag input.

LayerFish supports inertial scrolling, a common technique used in touch interfaces for browsing large lists. With a quick

Figure 3. Example of fisheye scene index in LayerFish. The focus of the fisheye is on the correspondent of the woman in a coat. Correspondents get smaller in size further from the focus.

flick, the fisheye scrolls based upon the velocity and direction of the flick, decelerating with time. We employ a form of speed-coupled flattening during inertial scrolling to address target acquisition issues [8]. That is, when the velocity of inertial scrolling exceeds a threshold, we flatten the fisheye scene index, making all correspondents equal in size. As the velocity falls back below the threshold, we redraw the fisheye with a focus on the top-most or bottom-most visible correspondent in the direction of the scroll.

Layering Interactions

The user adjusts an element's position in the visual stack by first selecting and then dragging and dropping, with pen or touch, its correspondent up or down the scene index. A correspondent is selected by briefly pressing down on it. The selected correspondent enlarges slightly, giving the effect of popping out of LayerFish. A rectangular placeholder highlights the current position of a selected correspondent in the scene index (Figure 4). As the selected correspondent is dragged up or down, the placeholder moves up and down the list accordingly, swapping positions with correspondents. The fisheye redraws, maintaining focus on the placeholder. When the user drops the selected correspondent, it is placed back in the scene index in the position of the placeholder.

Bimanual Scrolling

Drag and drop layering only supports layering above or below what is currently visible in the fisheye. As user the drags a selected correspondent, the focus changes, providing less space to visualize correspondents in the direction of the drag. To overcome this issue, the user can scroll the scene index without changing the focus, using a bimanual gesture. With a correspondent selected using the right hand, the user can drag the scene index using the left hand, causing the fisheye to scroll. The selected correspondent remains fixed below the right hand, while its position in the scene index changes as correspondents above or below it are scrolled up or down.

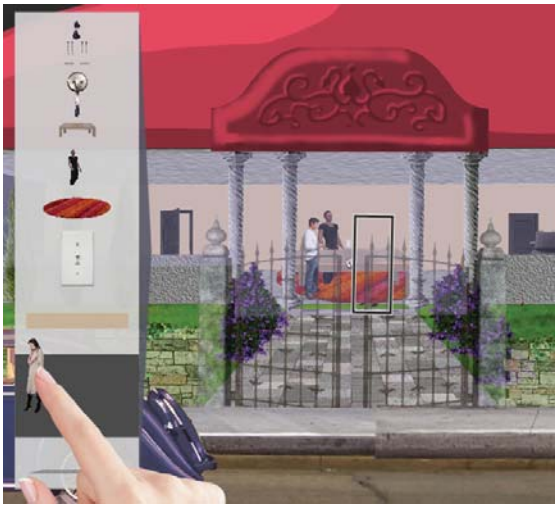

Figure 4. The correspondent for the woman in a coat is selected. A gray rectangle indicates the current position of the selected correspondent in the scene index. A rectangular outline in the design space shows the location of the element represented by the selected correspondent, even though the element is hidden behind the beige wall (the correspondent directly above in the scene index).

The user can scroll the fisheye scene index at a faster rate by dragging up or down with a left hand touch, beyond the bounds of LayerFish. This initiates automatic scrolling. The further away from LayerFish that the user drags the left hand touch, the faster the automatic scrolling happens.

EVALUATION DESIGN

To evaluate the effectiveness of LayerFish for adjusting the visual stacking order of overlapping content, we designed a repeated measures (within-subjects) evaluation comparing LayerFish with a traditional scene index, which we call *Sidebar*. We compare differences in task time between LayerFish and Sidebar. We vary several independent variables, including the number of elements and overlap density, to investigate in which situations LayerFish performs better than Sidebar.

Apparatus

The study apparatus consisted of a pre-questionnaire, two tutorial videos, 12 training tasks, and 48 study tasks. Participants began by answering the pre-questionnaire, which collected demographic information and details about their experiences using visual design tools and layering. Then, a tutorial video explained the task that participants would perform and demonstrated how to use either LayerFish or Sidebar. Next, participants performed 6 training tasks, followed by 24 study tasks using either LayerFish or Sidebar, depending on which tutorial video was shown. After completing the tasks, a second tutorial video demonstrated how to use the other technique, either LayerFish or Sidebar, that the participant had not yet used. Again, the participants performed 6 training tasks, followed by 24 study tasks. The study concluded with a post-questionnaire. Study sessions lasted approximately 30 minutes. Participants were given a $15 Amazon gift card.

Task

Each task consisted of a visual arrangement of overlapping elements (see Figures 6 and 7). Each element was a dancing figure from the artworks of Keith Haring [5], providing playfully engaging visuals with low complexity. The number of elements and how much they overlap was specified by independent variables (see *Independent Variables* section). The objective of each task was to layer a *task element* so that it resides between two goal elements in the z-dimension. The task and goal elements were uniquely colored with special fill patterns to make them easily distinguishable from the other elements (see Figure 5). The two goal elements always resided near the top of the visual stack, while the goal element resided near the middle or bottom. Performing a task consisted of three stages: (1) accessing the layering technique; (2) finding the task element; and (3) layering the task element between the two goal elements.

Figure 5. Yellow-gridded task element in the center, and two goal state elements (blue-dotted and red-striped) on left and right.

Sidebar

Sidebar represents a typical scene index used in visual design tools. In the study, Sidebar appears on the preferred hand side of the screen and contains correspondents for all elements (see Figure 6). Using a single touch drag, Sidebar is scrolled. The participant adjusts the layering of an element by dragging out the correspondent horizontally. The correspondent is removed from the scene index and appears under the participants dragging touch. Dragging the correspondent above or below the presented scene index will cause scroll the scene index in the dragged direction, beyond the initially visible subset of elements. Dropping the dragged correspondent back onto the scene index enables the participant to layer an element higher or lower in the visual stack. Dropping the correspondent outside the scene index cancels the operation, returning the correspondent to its original position.

Hardware Setup

The study workstation consisted of a Windows 8.1 PC with an Intel i7-5960X processor, 16GB RAM, and an NVIDIA GeForce GTX 680 Ti graphics card. Pen and touch input was supported through a Wacom Cintiq 24HD Touch display (1920 x 1200 pixels). Two cameras, one overhead and one on the right, captured participants' gestural interactions.

Independent Variables

Tasks involved 4 independent variables: Technique, Overlap Density, Number of Elements, and Layering Distance.

Technique

Technique was either LayerFish or Sidebar.

Overlap Density

Overlap Density is either *Low* or *High*. For Low Overlap Density, the task and goal elements only overlap with a few other elements (see Figure 7). This represents common layering tasks, such as when designing slides for a Powerpoint presentation. For High Overlap Density, all elements overlap. This represents the worst possible case for layering tasks.

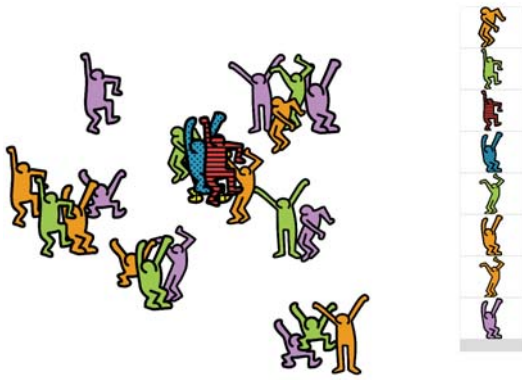

Figure 6. Study task example with Sidebar and Low Overlap Density.

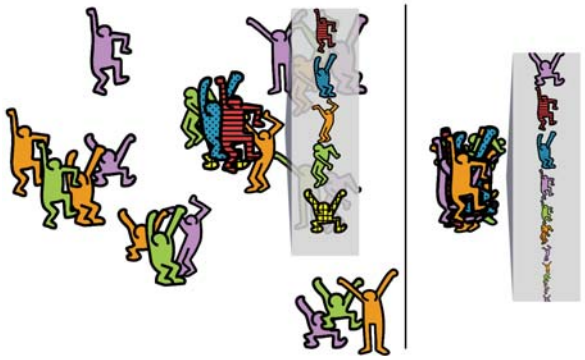

Figure 7. Low Overlap vs. High Overlap Density with LayerFish.

High Overlap Density focuses investigation on how LayerFish's fisheye visualization compares with a traditional scene index when the numbers of correspondents are the same. In Low Overlap Density tasks, LayerFish can be used to select a subset of elements. In High Overlap Density tasks, all elements overlap, so the user is unable to select a subset using LayerFish. As a result, both techniques present an equal number of correspondents, where the primary difference is how the correspondents are presented, either with fisheye or in a traditional scene index.

Number of Elements

The Number of Elements is either *25* or *100*. Through informal needs and requirements gathering sessions, Landscape Architecture students reported often creating design representations with around 100 layers. In the most extreme cases, all these layers overlapped. Thus, the condition with 100 elements and High overlap, represents a worst possible case scenario. While, 25 elements and High overlap represents the more common problematic scenario. One of our principle hypotheses is that LayerFish will scale better than Sidebar, as the number of elements increases.

Layering Distance (Relative)

Relative Layering Distance is the number of elements between the task element and the two goal elements at the start of the task. We use two possibilities for Layering Distance, *Medium* and *Far*. The absolute magnitude of these distances, in elements, varies directly with the Number of Elements. When Number of Elements is *25*, *Medium* is 7 and *Far* is 19. When Number of Elements is *100*, *Medium* is 44 and *Far* is 94. We vary Layering Distances so that participants do not know where the task element is located in each trial. Using two fixed distances for each Number of Elements reduces variability, and so facilitates comparison across techniques.

Tasks: Independent Variable Conditions

The 4 independent variables, each with 2 levels, produce 16 different conditions. Participants performed a total of 60 tasks, of which 12 were training and 48 were timed trials. For the timed trials, each condition was performed by each participant 3 times ($16 \times 3 = 48$).

Dependent Variables: Time Metrics

Layering tasks with scene indexes involve first finding a desired correspondent, and then adjusting that correspondent's position in the scene index. We wondered how a fisheye scene index would compare with a traditional scene index for this components of a layering task. Thus, we derive three time metrics for analyzing participants performance: Total Task Time, Find Time, and Layering Time. *Total Task Time* measures how long it took participants to perform the task in entirety. *Find Time* measures how long it took participants to find the task element that must be layered. Find Time is calculated as the time between the first touch down on the scene index (to scroll) to the first touch down on the task element. *Layering Time* measures how long it took participants upon finding the task element to reorder it to the goal state. Layering Time is calculated as the time after selection until the task element is positioned between the two goal elements.

We note that Find Time + Layering Time ≠ Total Task Time. Total Task Time includes additional time for selecting and activating LayerFish and for accessing Sidebar. We do not include a separate comparison of this additional time since LayerFish is expected to always be slower, because selection and activation requires a longer input sequence than the single touch required for accessing Sidebar.

PARTICIPANTS AND RESULTS

We recruited 47 participants (12 female) between 19 and 33 years old (22 mean). Participants were university students, primarily from Computer Science. We specifically invited participants from design-centric courses in Computer Science and Landscape Architecture to recruit participants with extensive experience working with layers. A majority (37) had worked with design tools at least once. Before investigating with expert visual designers, we sought to evaluate LayerFish with a more accessible population, as an intial usability study. Future work will examine expert visual designers.

We conducted a 2 (Technique) x 2 (Overlap Density) x 2 (Number of Elements) x 2 (Layering Distance) x 3 (trials) repeated measures analysis of variance (RM-ANOVA) on Total Task Time, Find Time, and Layering Time metrics for *All Participants*, the set of 47 participants. We performed pairwise comparison for each condition (see Table 1). Further, we examine a subset of All Participants identified as Experienced Layering Participants. We observe no order effects. Since a task was only completed when the task element was layered between the two goal elements, there is no measure

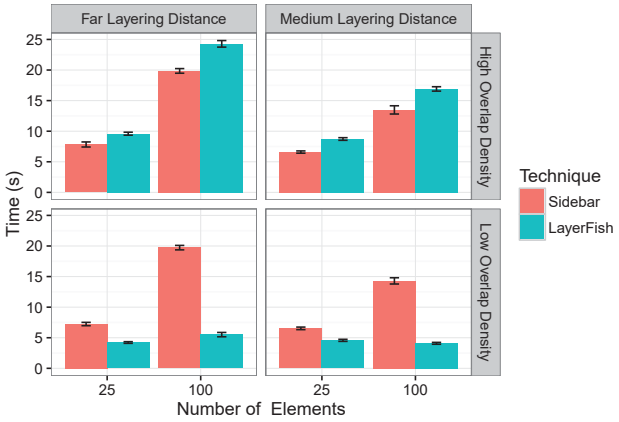

Figure 8. Mean Total Task Time for All Participants.

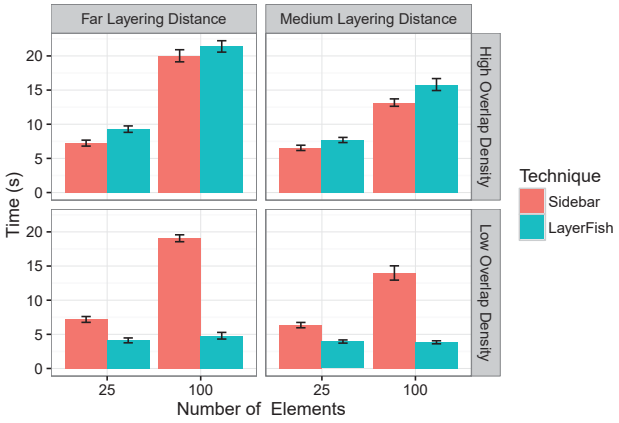

Figure 9. Mean Total Task Time for Experienced Layering Participants.

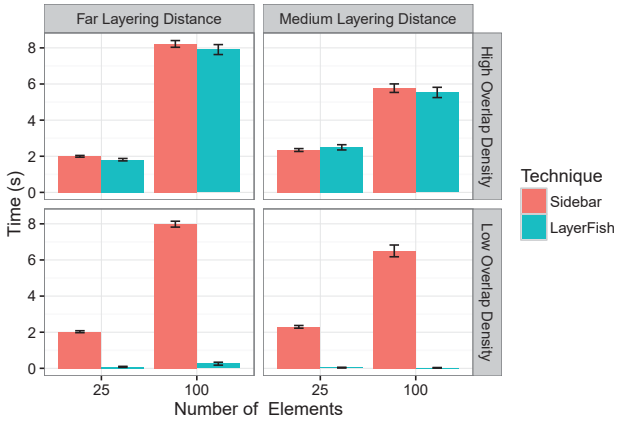

Figure 10. Mean Find Time for All Participants.

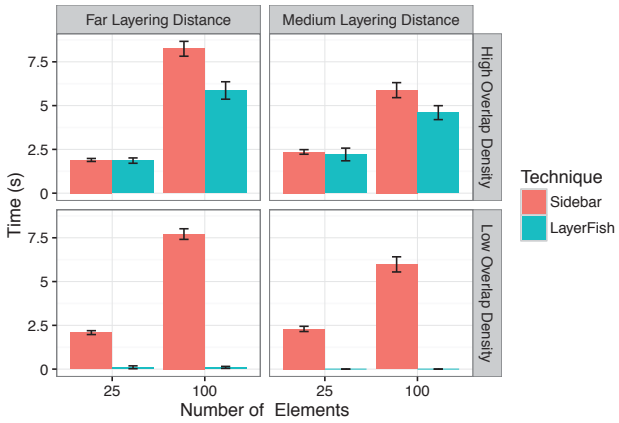

Figure 11. Mean Find Time for Experienced Layering Participants.

of errors. A post-questionnaire provides subjective experience report data.

Experienced Layering Participants

Through observations of participants during study sessions and variances in the data, we suspected that the participants more experienced with layering would performed tasks faster. From the All Participants set, we identify 12 Experienced Layering Participants based upon responses to two questions in the pre-questionnaires:

- How often do you use visual design tools?
- On average, how many layers do your projects contain?

These 12 participants responded that they often or frequently use visual design tools and typically work with at least 10 layers. The other 37 participants rarely used visual design tools, and on average, worked with 5 or less layers. For each time metric, we specifically compare times for these experienced users between the two techniques.

Total Task Time: All Participants

Total Task Time begins when a participant first touches and ends when the task is completed correctly. For LayerFish, this includes selection and activation time. Total Task Time using LayerFish was less than Sidebar with Low Overlap Density, but greater than with High Overlap Density (see Figure 8). Differences in means were statistically significant in all conditions (Table 1, A1-8). We observed a main effect for each independent variable: Technique ($F_{1,34} = 95.862$, $p < 0.001$), Overlap Density ($F_{1,34} = 918.01$, $p < 0.001$), Number of Elements ($F_{1,34} = 1938.316$, $p < 0.001$), and Layering Distance ($F_{1,34} = 154.541$, $p < 0.001$).

Total Task Time: Experienced Layer Participants

Experienced Layering Participants were on average faster for Total Task Time when compared with All Participants, particularly for LayerFish. In Low Overlap tasks, these participants were faster with LayerFish (see Figure 9). In High Overlap tasks, these participants were faster with Sidebar, but only some cases were statistically significant (Table 1, E6 and E7).

Find Time: All Participants

In Low Overlap Density tasks, participants were faster at finding the task element with LayerFish (Figure 10, Table 1 A1-4). In High Overlap Density tasks, participants averaged faster Find Time with LayerFish in most conditions. These results were statistically significant when Layering Distance is Far (Table 1, A6 and A8). We observed a main effect

Table 1. Pairwise comparison of Total Task Time, Find Time, and Layering Time between the two Techniques for each combination of Overlap Density, Number of Elements, Layering Distance. Rows A1-A8 (white background) show mean times with statistical significance for All Participants. Rows E1-E8 (gray background) show mean times with statistical significance for Experienced Layering Participants, a subset of All Participants.

				Total Task Time (s)			*Find Time (s)*			*Layering Time (s)*		
	Overlap Density	**Number of Elements**	**Layering Distance**	**Layer Fish**	**Sidebar**	$\frac{p <}{F =}$	**Layer Fish**	**Sidebar**	$\frac{p <}{F =}$	**Layer Fish**	**Sidebar**	$\frac{p <}{F =}$
A1	Low	25	Medium	4.6	6.5	$\frac{.001}{98.5}$	.036	2.3	$\frac{.001}{810.8}$	1.61	4.22	$\frac{.001}{145.7}$
E1	Low	25	Medium	4.01	6.35	$\frac{.001}{25.44}$	0	2.1	$\frac{.001}{187.3}$	1.08	4.00	$\frac{.001}{60.62}$
A2	Low	25	Far	4.2	7.2	$\frac{.001}{53.7}$	.081	2.03	$\frac{.001}{1036}$	1.48	5.21	$\frac{.001}{189.8}$
E2	Low	25	Far	4.12	7.18	$\frac{.001}{28.42}$	0	2.09	$\frac{.001}{155.6}$	1.14	5.10	$\frac{.001}{69.75}$
A3	Low	100	Medium	4.1	14.3	$\frac{.001}{283}$	.03	6.5	$\frac{.001}{397.1}$	1.43	7.79	$\frac{.001}{163.7}$
E3	Low	100	Medium	3.84	14.39	$\frac{.001}{71.66}$	.03	5.43	$\frac{.001}{201.8}$	1.03	8.03	$\frac{.001}{59.55}$
A4	Low	100	Far	5.5	19.7	$\frac{.001}{797}$	.10	7.98	$\frac{.001}{1858}$	2.23	11.75	$\frac{.001}{556.1}$
E4	Low	100	Far	4.80	19.16	$\frac{.001}{388.8}$	.07	7.38	$\frac{.001}{437.6}$	1.24	11.36	$\frac{.001}{414.7}$
A5	High	25	Medium	8.7	6.6	$\frac{.001}{50.2}$	2.33	2.34	$\frac{.97}{.002}$	3.77	4.25	$\frac{.20}{5.07}$
E5	High	25	Medium	7.6	6.54	$\frac{.10}{3.25}$	2.2	2.35	$\frac{.65}{.212}$	3.22	4.19	$\frac{.05}{3.63}$
A6	High	25	Far	9.6	7.8	$\frac{.01}{7.51}$	1.8	2.0	$\frac{.02}{5.67}$	5.10	5.83	$\frac{.11}{2.59}$
E6	High	25	Far	9.28	7.23	$\frac{.001}{25.44}$	1.9	1.9	$\frac{.83}{.05}$	3.97	5.34	$\frac{.047}{4.15}$
A7	High	100	Medium	16.9	12.9	$\frac{.001}{20.2}$	5.53	5.71	$\frac{.54}{.39}$	8.41	7.16	$\frac{.24}{1.44}$
E7	High	100	Medium	15.6	13.17	$\frac{.05}{4.22}$	4.6	5.9	$\frac{.10}{2.49}$	7.12	7.5	$\frac{.70}{.209}$
A8	High	100	Far	23.2	19.9	$\frac{.001}{23.3}$	7.46	8.22	$\frac{.03}{4.1}$	13.39	11.64	$\frac{.044}{4.10}$
E8	High	100	Far	21.39	20.0	$\frac{.50}{.477}$	5.86	8.25	$\frac{.017}{6.18}$	12. 69	11.78	$\frac{.61}{.268}$

for all factors: Technique ($F_{1,34}$ = 1412.692, $p < 0.001$), Overlap Density ($F_{1,34}$ = 416.6, $p < 0.001$), Number of Elements ($F_{1,34}$ = 1175.654, $p < 0.001$), and Layering Distance ($F_{1,34}$ = 31.753, $p < 0.001$).

In Low Overlap Density tasks, Find Time for LayerFish can be zero seconds (Table 1, A1-4, E1-4). Find Time is intended to measure the scrolling time required to find the task element. Find Time is measured as the time between the first touch down within a technique and the first touch down on the task element. With LayerFish, these two touches may be the same touch, as selection allows the participant to see the task element without scrolling.

Find Time: Experienced Layering Participants

The Experienced Layering Participants were faster at finding the task element on average using LayerFish than Sidebar (see Figure 11). This result was statistically significant for all Low Overlap Density tasks and when Number of Elements was 100 and Layering Distance was Bottom for High Overlap Density tasks (Table 1, E1-4, E8). These participants also were on average faster at finding the task element than the average for All Participants.

Layering Time: All Participants

Layering Time measures long it takes to adjust the visual stack position of the task element so that it is between the two goal elements. The metric is derived by calculating the

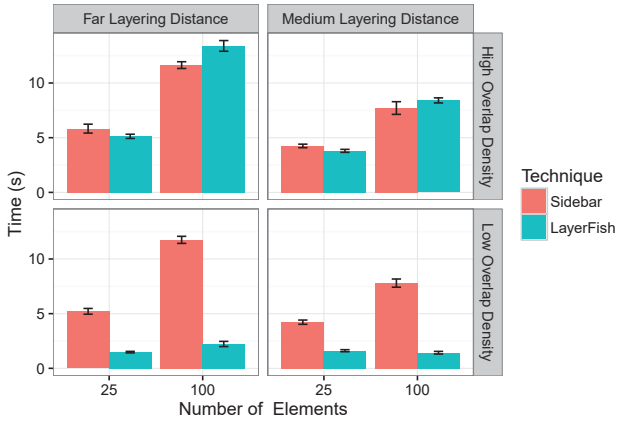

Figure 12. Mean Layering Time for All Participants.

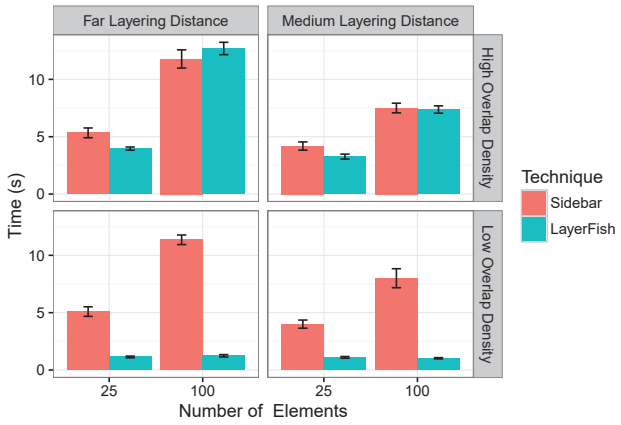

Figure 13. Mean Layering Time for Experienced Layering Participants.

difference in time between the selection of the task element and the finishing of the task. In Low Overlap Density tasks, participants had faster Layering Times when using LayerFish (see Figure 12). This result is statistically significant for all Low Overlap Density tasks (Table 1, A1-4). In High Overlap Density tasks, participants were faster with LayerFish when the Number of Elements was 25, but slower with LayerFish when the Number of Elements was 100. This result was statistically significant only in tasks with 100 elements and the Layering Distance is Far (Table 1, A8). We observed a main effect for all factors: Technique ($F_{1,34} = 247.403$, $p < 0.001$), Overlap Density ($F_{1,34} = 471.168$, $p < 0.001$), Number of Elements ($F_{1,34} = 1105.93$, $p < 0.001$), and Layering Distance ($F_{1,34} = 140.914$, $p < 0.001$).

Layering Time: Experienced Layering Participants

Experienced Layering Participants had faster Layering Times for Low Overlap Density tasks when using LayerFish (Figure 13). These results were statistically significant (Table 1, A1-4). In High Overlap Density tasks, these participants were faster with LayerFish in all cases, except when the Number of Elements was 100 and the Layering Distance was Far (E8). The results were statistically significant only when the Number of Elements was 25 (E5-6).

Experience Reports

Post-questionnaire responses provide reports of subjective experiences (see Figure 14). A majority of participants agreed that LayerFish was easy to use ($\chi^2 = 30.13$, $p < 0.001$), and they felt faster with LayerFish for both a few elements ($\chi^2 = 21.62$, $p < 0.003$) and many elements ($\chi^2 = 10.55$, $p < 0.05$). Participants agreed that LayerFish required less effort with only a few elements ($\chi^2 = 9.49$, $p < 0.05$). However, with many elements, only a slight majority agreed that LayerFish required less effort. This result was not statistically significant ($\chi^2 = 6.94$, $p < 0.14$). Participants agreed that they would use LayerFish in visual design tools like Adobe Photoshop ($\chi^2 = 20.55$, $p < 0.004$).

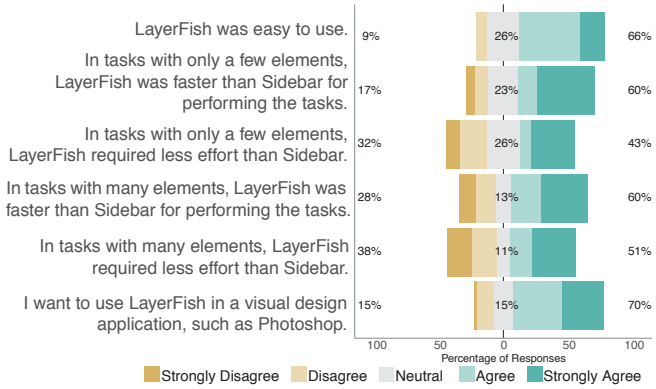

Figure 14. Results from post-questionnaire responses. Percentages show breakdowns participants' agreement (right), neutrality (middle), or disagreement (left) for each statement.

DISCUSSION

We hypothesized that our independent variables, which differentiate situations that users are likely to experience when working with overlapping content, would yield interesting findings. We observe significant differences between the two techniques for Overlap Density, Number of Elements, and Layering Distance. The most significant differences occur with Overlap Density. LayerFish was faster than Sidebar for Low Overlap tasks and slower in High Overlap tasks. We suspect that LayerFish requires additional learning time, as participants gain understanding of how interactions affect the fisheye representation. We also observed task performance differences for Experienced Layer Participants that indicate benefits of LayerFish for more expert users.

Low Overlap Density

In Low Overlap Density tasks, mean Total Task Time, Find Time, and Layering Time with LayerFish were all shorter than with Sidebar. Low Overlap Density means that the task element and goal elements only overlap with a few other elements. Selection in LayerFish allows the participant to engage only those few elements that overlap, unlike with Sidebar, where all elements are present in the scene index. Not surprisingly, this capability improves layering performance. After selection with LayerFish, the task element was immediately visible and could be quickly layered between the two goal elements, which were also visible. LayerFish does not require scrolling in this case, while Sidebar does.

LayerFish does not present a fisheye visualization in these low overlap tasks, instead using the common scene index representation. Ramos et al. suggested (but did not investigate) that an in-place scene index palette would improve layering times [19]. We have shown that this is true with at least 25 elements. We suspect that with fewer elements, the performance of a global scene index, like Sidebar, will be comparable to a contextual technique, like LayerFish. There exists a threshold in the number of elements, below which the selection and activation time for a contextual technique exceeds the time to move to the side to access a global technique.

High Overlap Density

We sought to understand the effects of the fisheye visualization on layering performance, by including a task condition, High Overlap, in which both LayerFish and Sidebar include a similar number of elements.

The Total Task Time results show that LayerFish is slower than Sidebar in high overlap tasks. LayerFish requires a selection gesture to activate. High Overlap tasks require that participants select (nearly) all elements using LayerFish. In this case, the selection takes time but doesn't help the user. When we further breakdown task time, for Find Time, we find that LayerFish is faster than Sidebar when there are 100 elements and Layering Distance is Far (Table 1, A8, E8). We suspect the fisheye's spatial distortion allows participants to scroll longer distances in a single touch gesture. As Layering Distance increases, so too does the difference in Find Time between LayerFish and Sidebar.

Meanwhile, for Layering Time with All Participants, LayerFish was slower when Number of Elements was 100 and Layering Distance was Far (Table 1, A8). When we look just at Experienced Layering Participants, LayerFish was faster when the Number of Elements was 25 (E5-6). We observe a larger variance in Layering Times when Layering Distance is Far, particularly for LayerFish. Bimanual scrolling was new to participants. We suspect that it may require more training for users to become comfortable at using it effectively.

Experienced Layer Participants

We observe a difference in the performance of the Experienced Layering Participants. Using LayerFish, these participants were faster than the average for All Participants. Yet, while using Sidebar, they performed, on average, similarly to All Participants. These performance differences for LayerFish are most evident in the Find Time metric with High Overlap tasks, particularly with 100 elements. In these cases, Experienced Layering Participants' mean Find Time is almost two seconds faster than the mean time for All Participants. The faster performance exhibited by Experienced Layering Particpants with LayerFish indicates that LayerFish has performance benefits for visual designers and others who work with many layers of overlapping elements. These participants seem more adept at learning to use LayerFish effectively. Novice layer users may also see improved performance with greater training and experience with layering.

The other significant difference was in Layering Time for High Overlap tasks with 25 elements. In these tasks, Experienced Layering Participants were faster using LayerFish than Sidebar. This matches the observed results for All Participants, except that, unlike for all, the differences for Experienced Layering Participants are statistically significant. All participants used bimanual scrolling in LayerFish to complete tasks with dense overlap. Again, bimanual scrolling is a new interaction, different from what participants are use to doing with touch devices. These results indicate that Experienced Layering Participants were among the quickest to learn how to effectively use the technique, emphasizing the benefits of LayerFish for visual designers who work with many layers.

Participant Experiences

Participants reported positive experiences using LayerFish. Interestingly, despite that participants were actually slower (in Total Task Time) using LayerFish with many elements, many participants felt that they were faster. When we exclude selection time for LayerFish, most participants were faster using LayerFish. This may have been part of participants' consideration when responding. Participants were not in agreement that LayerFish requires less effort than Sidebar when working with many elements. Since LayerFish requires additional touches for selection and bimanual scrolling, it is not surprising that participants felt it didn't require less effort.

IMPLICATIONS FOR DESIGN

We use the findings from the evaluation to derive implications for the design of pen+touch and multi-touch design tools that support layering operations.

Design tools with a scene index would benefit from transitioning to a fisheye visualization as the number of elements in the design space grows large. We saw benefits with LayerFish when the design space contained at least 25 elements, even when all elements overlapped. LayerFish was faster than Sidebar for Find Time in High Overlap tasks when Layering Distance was Far. We suspect that the spatial distortion of the fisheye allowed users to scroll longer distances more quickly. Prior work has used semantic zooming [18] as a form of spatial scaling to support speed-dependent scrolling rates in document browsing tasks [15]. We have shown that a fisheye view, as an alternate form of spatial scaling, can also improve scrolling performance in layering tasks with a scene index.

Design tools should support both in-place (i.e. LayerFish) and always-present (i.e. Sidebar) techniques for layering overlapping content. As the number of elements in a design grows large, the traditional scene index requires more effort and attention to effectively layer elements. An in-place technique, which enables selection to filter elements the user wishes to layer, can improve layering task performance time. However, when there are many elements that highly overlap, the user is unable to effectively select only the desired elements, reducing the benefits of selection. Additionally, when there are only a few elements, all are directly visible in the scene index, and scrolling is not required. In these cases, we recommend a traditional scene index that transitions to a fisheye when the number of elements grows large (at least 25).

Multi-touch scene indexes should support bimanual scrolling, particularly for tools in which users typically create 10 or

more layers. Bimanual interactions have been shown to improve performance of compound tasks, in which each hand can adjust different parameters of interaction [16]. In LayerFish, the right hand layers a selected correspondent, while the left hand scrolls the fisheye scene index. Unimanual approaches require dragging the selected correspondent to the top or bottom of the scene index, requiring more total touch distance travel than with our bimanual technique.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation, under grant IIS-1247126, and Wacom Technology Corporation. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not reflect the views of the NSF or Wacom.

CONCLUSION

We present LayerFish, a new bimanual interaction technique for layering overlapping content. With a bimanual gesture, the user selects a subset of elements within a larger design space. When filtering out elements with selection, layering operations take less time than with a traditional scene index. A fisheye scene index increases the number of visible correspondents, while reducing time to find a desired element. Bimanual scrolling allows a selected correspondent, manipulated with the right hand, to remain as the focus of the fisheye while the left hand scrolls the view.

We apply a fisheye visualization to a traditional scene index. Prior work has investigated various advantages and disadvantages of fisheyes in numerous contexts [3]. Our findings show that fisheyes can be faster for finding the desired correspondent in a large scene index. One of the primary challenges with fisheyes is target acquisition, since targets are continuously moving due to changes in the fisheye focus. One solution is to use speed-coupled flattening to remove the fisheye effects as panning or scrolling velocities increase [8]. We employ speed-coupled flattening with inertial scrolling in LayerFish. For target acquisition with hierarchical structures of text, Fisheye Menus [2] are slower than traditional hierarchical menus [14]. Our present investigation focuses on flat scene index structures with thumbnail images. Future work can investigate hierarchical scene indexes.

In order to focus investigation on layering performance, participants were not able to zoom and pan the design space. The High Density Overlap tasks where all elements overlap completely, while representing extreme scenarios, would gain little benefit from zooming and panning. However, we presently use LayerFish in a design environment with an infinite, zoomable canvas. Future work will look at how a zoomable space affects selection and layering performance with LayerFish.

Designers are accustomed to working with their hands. While working with physical media, visual designers generate new ideas by exploring different arrangements of elements, moving them around and sketching over and amidst these elements. Future work will investigate LayerFish in architecture contexts, where designers create complex visual representations with overlapping content. We hypothesize that visual designers would benefit from interactive tools that allow them to directly gesture with their hands to explore a space of ideas.

REFERENCES

1. Agarawala, A., and Balakrishnan, R. Keepin' it real: Pushing the desktop metaphor with physics, piles and the pen. In *Proc. ACM CHI* (2006), 1283–1292.
2. Bederson, B. B. Fisheye menus. In *Proc. ACM UIST* (2000), 217–225.
3. Cockburn, A., Karlson, A., and Bederson, B. B. A review of overview+detail, zooming, and focus+context interfaces. *ACM Comput. Surv. 41*, 1 (Jan. 2009).
4. Davidson, P. L., and Han, J. Y. Extending 2d object arrangement with pressure-sensitive layering cues. In *Proc. ACM UIST* (2008), 87–90.
5. Deitch, J., Geiss, S., and Gruen, J. *Keith Haring*. Rizzoli International Publications, 2014.
6. Furnas, G. W. Generalized fisheye views. *SIGCHI Bull. 17*, 4 (Apr. 1986), 16–23.
7. Grossman, T., Baudisch, P., and Hinckley, K. Handle flags: Efficient and flexible selections for inking applications. In *Proc. Graphics Interface* (2009).
8. Gutwin, C. Improving focus targeting in interactive fisheye views. In *Proc. ACM CHI* (2002), 267–274.
9. Hamilton, W., Kerne, A., and Robbins, T. High-performance pen + touch modality interactions: A real-time strategy game esports context. In *Proc. ACM UIST* (2012), 309–318.
10. Herrlich, M., Walther-Franks, B., Schröder-Kroll, R., Holthusen, J., and Malaka, R. Proxy-based selection for occluded and dynamic objects. In *Proc. of International Conference on Smart Graphics* (2011), 142–145.
11. Hinckley, K., Chen, X. A., and Benko, H. Motion and context sensing techniques for pen computing. In *Proceedings of Graphics Interface 2013* (2013), 71–78.
12. Hinckley, K., Yatani, K., Pahud, M., Coddington, N., Rodenhouse, J., Wilson, A., Benko, H., and Buxton, B. Pen + touch = new tools. In *Proc. ACM UIST* (2010).
13. Holz, C., and Baudisch, P. Understanding touch. In *Proc. ACM CHI* (2011), 2501–2510.
14. Hornbæk, K., and Hertzum, M. Untangling the usability of fisheye menus. *ACM Trans. Comput.-Hum. Interact. 14*, 2 (Aug. 2007).
15. Igarashi, T., and Hinckley, K. Speed-dependent automatic zooming for browsing large documents. In *Proc. ACM UIST* (2000), 139–148.
16. Kabbash, P., Buxton, W., and Sellen, A. Two-handed input in a compound task. In *Proc. ACM CHI* (1994).
17. Leithinger, D., and Haller, M. Improving menu interaction for cluttered tabletop setups with user-drawn path menus. In *Proc. ACM Tabletop* (Oct 2007).
18. Perlin, K., and Fox, D. Pad: an alternative approach to the computer interface. In *Proc. SIGGRAPH* (1993).
19. Ramos, G., Robertson, G., Czerwinski, M., Tan, D., Baudisch, P., Hinckley, K., and Agrawala, M. Tumble! splat! helping users access and manipulate occluded content in 2d drawings. In *Proc. of ACM AVI* (2006).