From the Lab to the World: Lessons from Extending a Pointing Technique for Real-World Use

Alex Jansen
The Information School
DUB Group
University of Washington
Seattle, WA 98195
ajansen7@uw.edu

Leah Findlater
The Information School
DUB Group
University of Washington
Seattle, WA 98195
leahkf@uw.edu

Jacob O. Wobbrock
The Information School
DUB Group
University of Washington
Seattle, WA 98195
wobbrock@uw.edu

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.
ACM 978-1-4503-0268-5/11/05.

Abstract
We present the *Pointing Magnifier* as a case study for understanding the issues and challenges of deploying lab-validated pointing facilitation techniques into the real world. The *Pointing Magnifier* works by magnifying the contents of an area cursor to allow for selection in a magnified visual and motor space. The technique has been shown in prior lab studies to be effective at reducing the need for fine pointing for motor-impaired users. We highlight key design and technical challenges in bringing the technique, and such techniques in general, from the lab to the field.

ACM Classification Keywords
H5.2 [Information interfaces and presentation]: User Interfaces—*Input devices and strategies*. K4.2. Computers and society: Social issues—*assistive technologies for persons with disabilities*.

General Terms
Design, Human Factors

Introduction
The quest to improve target acquisition has received significant attention in HCI over the years. Novel designs have included techniques to offer near-optimal "direct pointing" performance [4], attempts to beat

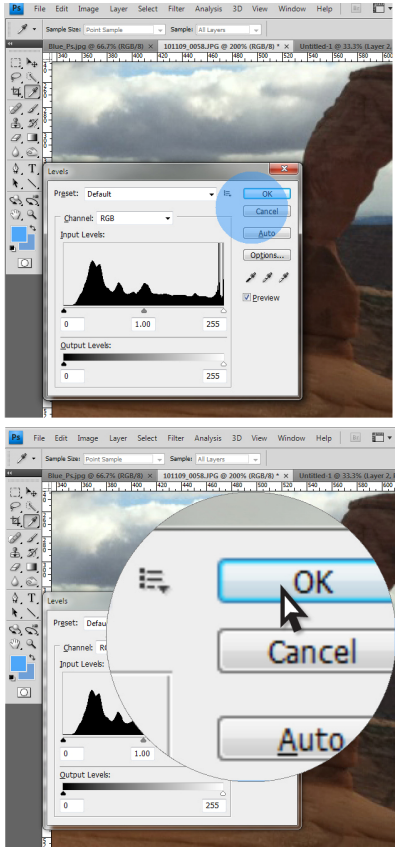

Figure 1. The Pointing Magnifier being moved over a dialog in Photoshop. The Pointing Magnifier appears as a blue area cursor (top). Once activated with a click, the Pointing Magnifier locks in place and magnifies the contents of the screen under the circular area cursor, which can be sized to suit user abilities (bottom).

Fitts' law through transformation of targets or the cursor (e.g., [5,9,10,13]), and techniques to support users with motor impairments (e.g., [1,3,7,11,12,13]). While many techniques improve speed and accuracy in controlled experiments, they are rarely deployed outside of the laboratory or designed to support the full range of interaction necessary for real-world use. To create pointing techniques that will be widely adopted, we need to understand how techniques function within the complexities of real applications and the context of real user priorities.

As a case study for understanding the challenges of deploying pointing techniques for real-world use, we present the *Pointing Magnifier*[1], our substantial extension of an existing pointing technique, the *Visual-Motor Magnifier* [3], from a controlled lab setting. First introduced by Findlater et al., the *Visual-Motor Magnifier* [3] eases target selection for users with motor impairments by reducing the need for fine, precise pointing. It combines an area cursor with visual and motor magnification (Figure 1). A lab evaluation with 12 motor-impaired participants showed the *Visual-Motor Magnifier* improved performance and user satisfaction over a standard point cursor. However, the original design and evaluation only focused on one aspect of pointing: left-click. In real-world use, the mouse must support a wide range of other behaviors.

Specifically, two major challenges need to be addressed in extending a pointing technique such as the *Visual-Motor Magnifier* [3] for real-world use. Lab evaluations most often only consider left clicks, rather than the full range of interaction (e.g., right-clicks, double clicks, dragging, scrolling, menu operation). Many techniques are also *target aware*, meaning they need access to the location and size of targets on the screen. To encourage HCI researchers to address these challenges and expand pointing technique research to the field, we survey existing pointing technique evaluations, outline technical challenges and design requirements of creating novel deployable pointing techniques, and ground the discussion in our experience extending the *Pointing Magnifier* to support general pointing needs.

[1] The *Pointing Magnifier* can be found online at: http://depts.washington.edu/aimgroup/proj/ptgmag/

Pointing Technique Design and Evaluation

We classify existing pointing techniques on what we consider to be the most salient characteristics when moving from constrained lab settings to general deployments. We also highlight previous evaluations, focusing on field deployments.

Level of refinement. When techniques are evaluated in constrained laboratory test-beds (Figure 2), they do not require the same level of refinement or depth as techniques deployed in the field. Lab evaluations overwhelmingly study only single click interaction (e.g.,[1,4,5,6,7,9,12,13]). One exception is the lab evaluation of Steady Clicks, which incorporated single clicks and drags [11]. Dramatically improving single click speed and accuracy shows a technique has potential, but without supporting right click, double click, drag, or scroll, it is of limited use outside the lab.

Knowledge of target location and size. A major reason why pointing techniques may not make the transition to the real-world despite promising lab results is that they often require awareness of target location and size. Table 1 categorizes some popular

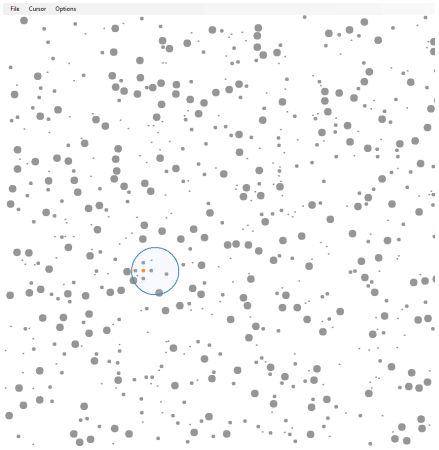

Figure 2. A screenshot taken from the test bed used for the lab evaluation of the enhanced area cursors.

techniques as *target-aware* or *target-agnostic* (i.e., requiring no knowledge of targets [12]). Major operating systems, such as Microsoft Windows and Mac OS X, do not expose information about target location and size in arbitrary software applications, so techniques such as the *Bubble Cursor* [4,8], *Click-and-Cross Cursor* [3], and *Sticky Icons* [1] have been restricted to laboratory studies in which a target-aware interface can be simulated. While these techniques have been tested extensively in laboratory settings, their effectiveness in real use is unknown. A creative, though limited exception, is a web browser-based implementation of the *Bubble Cursor* [8]. By accessing a webpage's document object model (DOM), the JavaScript bookmarklet can retrieve the location of links (targets) on the page.

While current operating systems inhibit target-aware pointing techniques, emerging technologies [2] may enable target-aware pointing techniques through reverse engineering user interfaces from their drawn pixels to identify targets. While implementing the *Bubble Cursor* [8] in a web browser is a start, we are on the cusp of being able to deploy target-aware pointing techniques in the real-world. Understanding the necessary refinements for extending pointing techniques to real applications is becoming increasingly important and is the focus of the work presented here.

Relationship to the native cursor. Pointing techniques that only modify the speed or placement of the native cursor are relatively easy to deploy for general use in comparison to techniques that modify the cursor itself or the shape or placement of targets (see classification in Table 1). The former class of techniques has a *direct* relationship to the native

	Target-aware	**Target-agnostic**
Direct Pointing	Sticky Icons [1] Force Field [1] Object Pointing [5] Ninja Cursor [9]	Angle Mouse [12] PointAssist [7] Steady Clicks [11]
Indirect Pointing	Bubble Cursor [4,8] Area Cursor [13] Visual-Motor Magnifier [3] Click-and-Cross [3]	Pointing Magnifier Fisheye Views [6]

Table 1. Classifying pointing techniques based on target-awareness and direct vs. indirect pointing. More challenging techniques to deploy in the field are darker.

cursor. The *Angle Mouse* [12] and *PointAssist* [7] are two examples. Both track mouse movement and lower mouse gain when pointing difficulty is detected. The techniques do not visually change the interface, so they do not need to provide accommodations for dragging, double-clicking, or other actions; making them relatively easy to transition from the lab to the field. Both have been shown to improve user performance in the lab and are available for public download[2].

In contrast, pointing techniques that have an *indirect* relationship with the native pointer, such as *Bubble Cursor* [4], *Click-and-Cross Cursor* [3], and *Visual-Motor Magnifier* [3], must intercept, interpret, and modify any mouse input before passing an action (e.g., click) to underlying applications. Techniques that fall

[2] The Angle mouse can be found online at: http://depts.washington.edu/aimgroup/proj/angle/

PointAssist can be found online at: http://www.cs.uiowa.edu/~hourcade/projects/pointassist/

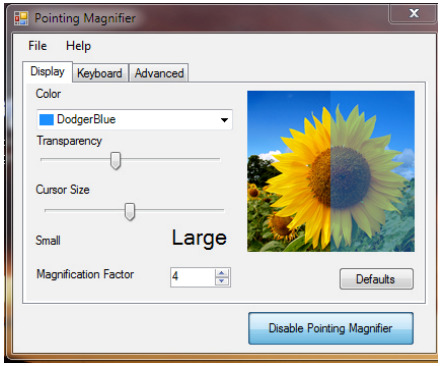

Figure 3. The control panel for the Pointing Magnifier includes options for setting color, transparency, size, and magnification factor.

into this class present greater technical and design challenges in deploying to the real world, and will likely need to explicitly accommodate all aspects of input and properly simulate interaction for the user.

Case Study: *Pointing Magnifier*

In extending the *Visual-Motor Magnifier* [3] to become the general-purpose *Pointing Magnifier*, we addressed several challenges and iteratively designed solutions to support interactions beyond single left-click.

The *Pointing Magnifier* is a two-stage pointing technique (Figure 1). During the first stage the user controls an area cursor of arbitrary size depending on their pointing accuracy [13]. To interact with a target, the user places the area cursor over that target and activates the magnifier by clicking any mouse button. This pins the circular cursor in place and causes everything under it to be magnified. The native mouse pointer then appears and the user interacts with magnified targets. Upon completion of an action (e.g., clicking, dragging), the pointer's location is transposed to its corresponding position in unmagnified space. Finally, the *Pointing Magnifier* returns to its original size and the user resumes control of the area cursor. To cancel out of the magnified state, the user can click anywhere outside of the magnified area. Importantly, although screen magnifiers have existed for a long time, they only change the visual space of the display, not the motor space. The *Pointing Magnifier* changes both, making pointing easier for people with motor impairments.

Design Decisions

Interactions for all pointing input had to be refined in extending the *Pointing Magnifier* to the real-world. The process involved iteratively refining each of the interactions, and gathering feedback from our own experiences and three additional users not familiar with the project. One member of our group also used the *Pointing Magnifier* as his primary cursor for two days near the end of development in an attempt to evaluate features and uncover unaddressed issues. In this section we explore the design decisions we made.

CONTROL PANEL AND KEYBOARD SHORTCUTS. A control panel allows users to change the *Pointing Magnifier*'s color, transparency level, area cursor size, and its magnification factor (Figure 3). Keyboard shortcuts can be customized to quickly enable/disable the *Pointing Magnifier*. User preferences are saved on closing the application.

TARGET-AGNOSTIC INTERACTION. The *Visual-Motor Magnifier* [3] used an inset *Bubble Cursor* [4] when magnified for selections. Since the Bubble Cursor requires awareness of targets, we had to replace it with a standard point cursor for the *Pointing Magnifier*.

SINGLE CLICKS. Single clicks are the most basic function supported by the *Pointing Magnifier.* Pressing any button (not just the primary button) will activate the magnified state. From there, pressing any button (e.g, left, right) will perform its respective function. As soon as the user presses down on a button, the *Pointing Magnifier* leaves the magnified state and the area cursor is placed at the translated location of the button press.

DOUBLE CLICKS. Two clicks must occur within a fixed distance and amount of time to be a double click. After a click occurs in the magnified state, the *Pointing*

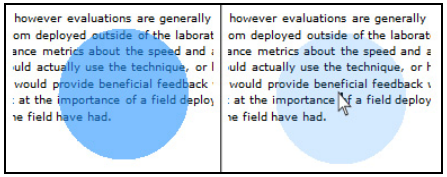

Figure 5. The Pointing Magnifier in its normal state (Left). The Pointing Magnifier while mouse events pass through to underlying windows. Notice how the magnifier is lighter on the right as well as the native cursor being visible.

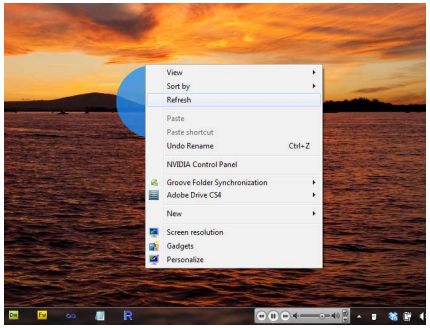

Figure 4. The Pointing Magnifier in unmagnified state behind a pop-up menu. While the Pointing Magnifier is still present and moves with the cursor, the point cursor behaves normally on the pop-up menu.

Magnifier returns to its unmagnified state but allows mouse events to be passed through to the underlying application until after the double-click time limit is reached. This delay prevents the *Pointing Magnifier* from being reactivated on the second click when a double click was intended. The ability to click through the *Pointing Magnifier* is visually portrayed to the user by increasing the opacity of the area cursor and displaying the native cursor at its center (Figure 4).

DRAGGING. When the mouse button is pressed down in the magnified state, the *Pointing Magnifier* leaves the magnified state, allowing clicks to pass through to the underlying application. However, as long as the mouse button remains pressed, mouse events will continue to pass through to underlying applications, allowing dragging to occur. Once the mouse button is released, the *Pointing Magnifier* resets.

MOUSE WHEEL. Scrolling the mouse wheel will not activate the *Pointing Magnifier* as no button is clicked. Mouse wheel events are passed directly through to the underlying application.

Implementation and Challenges

The *Pointing Magnifier* runs on Microsoft Windows and is a standalone application implemented using C# .NET 2.0 that we envision some users may set as a startup program. It runs as a top-level semi-transparent window, moving with and hiding the native mouse cursor. To magnify, the pixels on the screen underneath the area cursor are scraped and a static magnified bitmap image is created. Mouse events are captured and processed by the *Pointing Magnifier* using a low level mouse hook. In unmagnified state, button clicks activate the *Pointing Magnifier* and are not passed to the underlying application. In magnified state, interactions are translated by the *Pointing Magnifier* from magnified to unmagnified space before being simulated for the underlying application. To support the simulation, the underlying application must be set as the active and focused window (necessary for input to be handled properly) and to have mouse capture. Setting mouse capture allows the underlying application to continue receiving mouse movements for dragging actions and other specific behavior; for example, Microsoft Word requires mouse capture to display hover dialogs.

Limitations of the Pointing Magnifier

Mouse-over and hover state are not supported by the *Pointing Magnifier* because mouse position can only be passed to underlying applications when a click occurs. When activated, the *Pointing Magnifier* takes mouse capture from other applications. However, dialogs like pop-up and dropdown menus close when they lose mouse capture. In those situations, the window is drawn on top of the *Pointing Magnifier* (Figure 5), which at least allows the user to interact with it directly, even if magnification is temporarily not possible.

Future Work

We intend to conduct a field study in which we have a group of people with motor impairments use the *Pointing Magnifier* for 3-5 days on their home computers. We will also collect daily questionnaire feedback and conduct interviews to elicit the benefits and challenges not observable in a performance-based lab study. As the *Visual-Motor Magnifier* [3] was already evaluated in a controlled experiment, gathering performance data such as speed and error rate is not our primary concern. Furthermore, logged data is much

more difficult to interpret from the field than in the lab because we do not know a user's intent. Nonetheless, we will add logging to the *Pointing Magnifier* so that we can gather data about context and frequency of use. We will use our findings to improve the design and further refine features of the *Pointing Magnifier*.

Conclusion

While pointing techniques are common in HCI, they are often tested only in laboratory settings. We categorized pointing techniques in an attempt to determine which techniques could feasibly be deployed in the real-world and to identify the challenges in doing so. We presented the *Pointing Magnifier* as a case study for the challenges in extending a lab-validated, indirect, target-agnostic pointing technique to the real-world. The *Pointing Magnifier* has been shown in a lab study to be effective at supporting target acquisition for users with motor impairments. Substantially extending this technique for real-world use will allow the technique to be available to many more people, and our ability to learn from it will be greatly expanded.

Acknowledgments

This work was supported in part by the National Science Foundation under grant IIS-0811063.

References

[1] Ahlström, D., Hitz, M., and Leitner, G. An evaluation of sticky and force enhanced targets in multi target situations. *Proc. NordiCHI '06*, ACM (2006), 58-67.

[2] Dixon, M. and Fogarty, J. Prefab: implementing advanced behaviors using pixel-based reverse engineering of interface structure. *Proc. CHI '10*, ACM (2010), 1525-1534.

[3] Findlater, L., Jansen, A., Shinohara, K., Dixon, M., Kamb, P., Rakita, J., Wobbrock, J.O. Enhanced area cursors: reducing fine pointing demands for people with motor impairments. *Proc. UIST '10*, ACM (2010), 153-162.

[4] Grossman, T. and Balakrishnan, R. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. *Proc. CHI '05*, ACM (2005), 281-290.

[5] Guiard, Y., Blanch, R., and Beaudouin-Lafon, M. Object pointing: a complement to bitmap pointing in GUIs. *Proc. GI 2004*, Canadian Human-Computer Communications Society (2004), 9-16.

[6] Gutwin, C. Improving focus targeting in interactive fisheye views. *Proc. CHI '02*, ACM (2002), 267-274.

[7] Hourcade, J.P., Nguyen, C.M., Perry, K.B., and Denburg, N.L. Pointassist for older adults: analyzing sub-movement characteristics to aid in pointing tasks. *Proc. CHI '10*, ACM (2010), 1115-1124.

[8] Knightley, S. Bubble cursor example/prototype. http://stuartk.co.uk/bubble-cursor/.

[9] Kobayashi, M. and Igarashi, T. Ninja cursors: using multiple cursors to assist target acquisition on large screens. *Proc. CHI '08*, ACM (2008), 949-958.

[10] McGuffin, M.J. and Balakrishnan, R. Fitts' law and expanding targets: Experimental studies and designs for user interfaces. *ACM TOCHI. 12*, 4 (2005), 388-422.

[11] Trewin, S., Keates, S., and Moffatt, K. Developing steady clicks:: a method of cursor assistance for people with motor impairments. *Proc. ASSETS '06*, ACM (2006), 26-33.

[12] Wobbrock, J.O., Fogarty, J., Liu, S., Kimuro, S., and Harada, S. The angle mouse: target-agnostic dynamic gain adjustment based on angular deviation. *Proc. CHI '09*, ACM (2009), 1401-1410.

[13] Worden, A., Walker, N., Bharat, K., and Hudson, S. Making computers easier for older adults to use: area cursors and sticky icons. *Proc. CHI '97*, ACM (1997), 266-271.