Drag-and-Pop and Drag-and-Pick: techniques for accessing remote screen content on touch- and pen-operated systems

ABSTRACT

Drag-and-pop and drag-and-pick are interaction techniques designed for users of pen- and touchoperated display systems. They provide users with access to screen content that would otherwise be impossible or hard to reach, e.g., because it is located behind a bezel or far away from the user. Drag-and-pop is an extension of traditional drag-and-drop. As the user starts dragging an icon towards some target icon, drag-and-pop responds by temporarily moving potential target icons towards the user's current cursor location, thereby allowing the user to interact with these icons using comparably small hand movements. Drag-and-Pick extends the drag-and-pop interaction style such that it allows activating icons, e.g., to open folders or launch applications. In this paper, we report the results of a user study comparing drag-and-pop with traditional drag-and-drop on a 15' (4.50m) wide interactive display wall. Participants where able to file icons up to 3.7 times faster when using the drag-and-pop interface.

KEYWORDS:

Drag-and-drop, drag-and-pick, interaction technique, pen input, touchscreen, heterogeneous display.

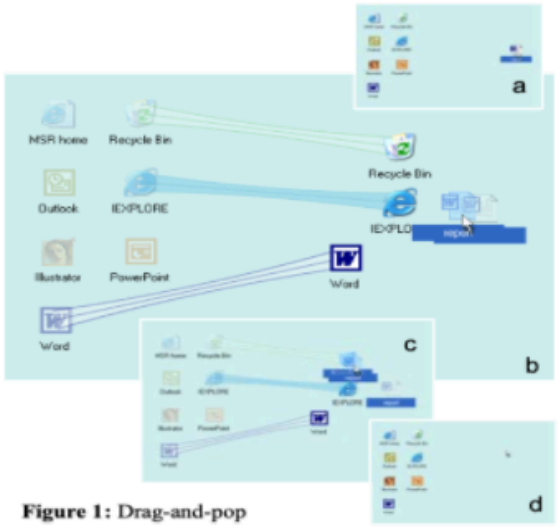

Figure 1: Drag-and-pop

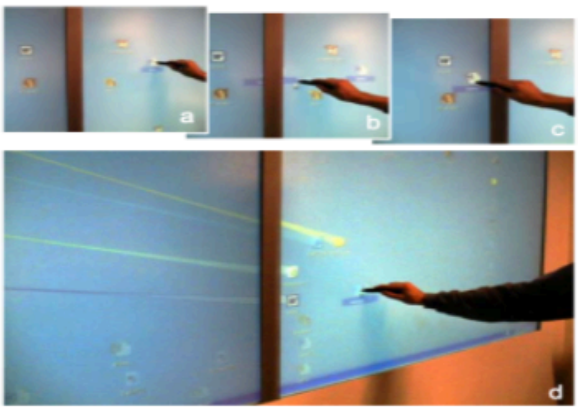

Figure 2: (a-c) Traditional drag-and-drop: Dragging an icon across the bezel requires the user to drop the icon half way across the bezel and pick it up at the other side (d) Drag-and-pop temporarily brings matching target icons to the current pen location, allowing the user to file icons without having to cross the bezel.

Drag-and-pick

modifies the drag-and-pop interaction concept such that it allows activating icons, e.g., to open a folder or to launch a program. While drag-and-pop is initiated by the user dragging an icon, drag-and-pick starts with the user performing a drag interaction on empty screen space. The system's response to this drag interaction is similar to drag-and-pop, but with two differences. First, all icons located in the direction of the drag motion will pop up, not only those of compatible type (Figure 3). Second, as the user drags the mouse cursor over one of the targets and releases the mouse button, the folder, file, or application associated with the icon is activated as if it had been double clicked. Figure 4 shows how this allows users to use the pen for launching an application, the icon of which is located on a monitor not supporting pen input. In principle, drag-and-pick can be applied to any type of widget, e.g., any buttons and menus located on a non-pen accessible monitor. In this paper, however, we will focus on the manipulation of icons.f

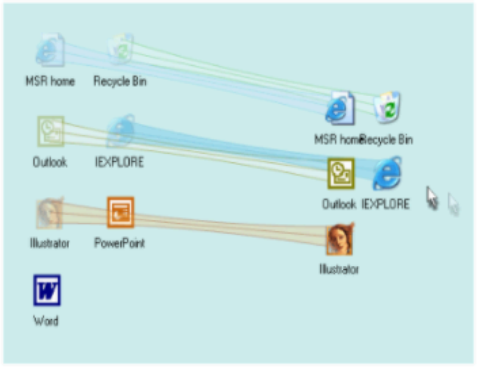

Figure 3: Drag-and-pick makes *all* icons in the direction of the mouse motion come to the cursor.

Figure 4: Drag-and-pick allows users to temporarily move icons from an external monitor to the tablet where the user can interact with them using the pen.

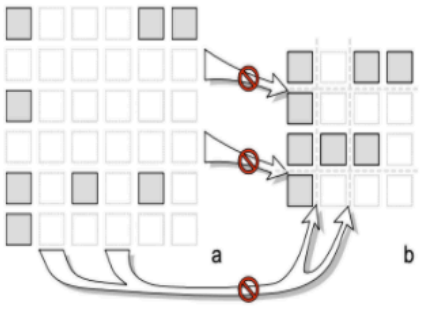

Figure 5: Drag-and-pop computes tip icon layouts (a) by snapping icons to a grid and then (b) removing empty rows and columns.

After the tip icon layout has been computed, drag-and-pop positions it on the screen such that the center of the layout's bounding box is located at the direct extension of the user's current mouse motion. The distance of the tip icon cluster to the user's current cursor position is configurable. For inexperienced users, we got best results with distances of around 100 pixels; shorter distances made these users likely to overshoot the cluster. For more experienced users, we were able to reduce the distance to values around 30 pixels, which allowed these users to operate drag-and-pop with less effort, in a more "menu-like" fashion. In order to reduce visual interference between tip icons and icons on the desktop, drag-and-pop diminishes desktop icons while tip icons are visible. This may obscure the nearby icons on the desktop making it hard to access to.

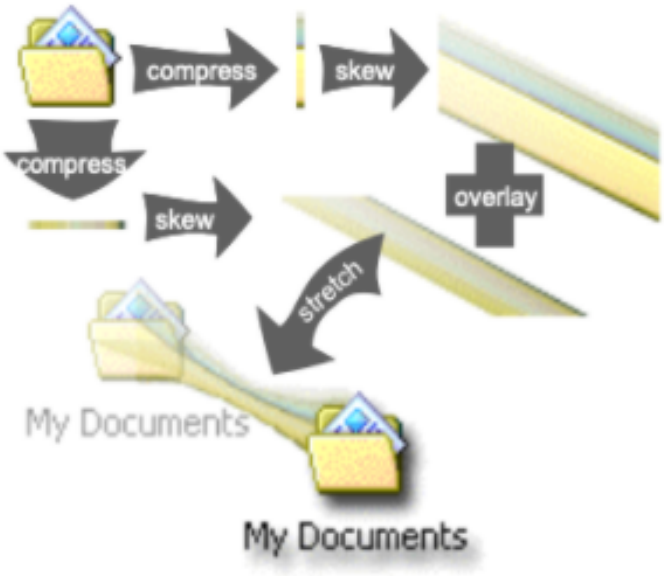

Figure 6: The motion blur textures on the rubber bands that connect tip icons with their bases are made by overlaying skewed copies of that icon.

Aborting Drag-and-Pop Interactions

As soon as tip icons and rubber bands are shown on the screen, drag-and-pop waits for the user to acquire one of the tip icons to complete the ongoing drag-and-pop or drag-and-pick interaction. There are two cases, however, in which users will want to abort the interaction without acquiring a tip icon. The first case is when the user dragged the mouse at a wrong angle so that the desired target icon did not pop up. In this case, the user may either drop the icon and try again or complete the interaction as a regular drag-and-drop interaction, i.e., by dropping the icon onto the target icon's base instead. The other case occurs if the user is intending to perform a regular mouse drag operation, for example to rearrange icons on the desktop or to capture a set of icons using a lasso operation. For these cases, drag-and-pop allows users to terminate tip icons onthe- fly and to complete the interaction without dragand- pop/pick. To abort, users have to move the mouse cursor away from the tip icon cluster while still keeping the mouse depressed. This can be done by overshooting the cluster or by changing mouse direction. In particular, this allows users to switch to the normal drag-and-drop and lasso-select functionality by introducing a simple zigzag gesture into

their cursor path. The zigzag contains at least one motion segment moving away from the tip icons, thus terminating tip icons as soon as they appear. The algorithm: the tip icon cluster is kept alive as long as at least one of the following three rules is successful. The first rule checks whether the mouse cursor has moved closer to the center of at least one of the icons in the tip icon cluster. This rule makes sure that the cluster does not disappear while users approach their targets. The second rule checks if the cursor is in the direct vicinity of an icon. This rule provides tolerance against users overshooting a tip icon while acquiring it. The third and last rule keeps the cluster alive if the cursor is stationary or if it is moving backwards very slowly (up to 5 pxl/frame). This rule makes drag-and-pop insensitive to jitter. Figure 7 illustrates the resulting behavior.

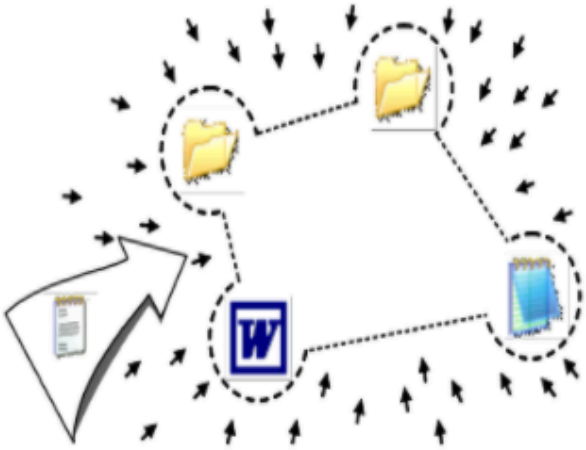

Figure 7: The tip icon cluster is kept alive as long as the user moves towards the cluster (arrows) or inside the convex hull surrounding the cluster (dashed).

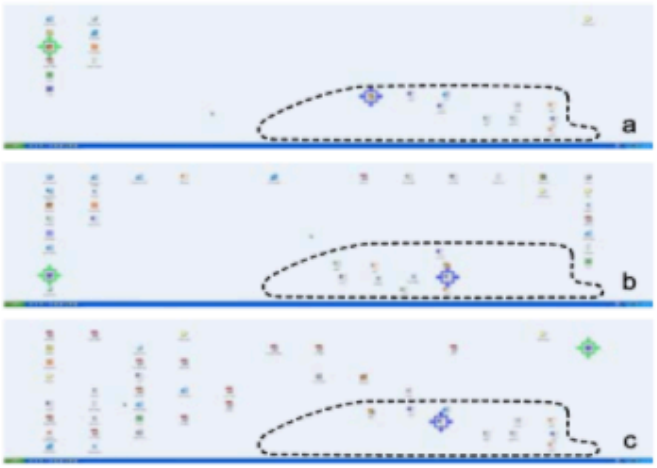

Figure 8: The (a) sparse, (b) frame, and (c) cluttered desktop layouts used in the study. The dashed line indicates the space reserved for the icons users had to file. Boxes around icons indicate icon to be filed and target.

Method

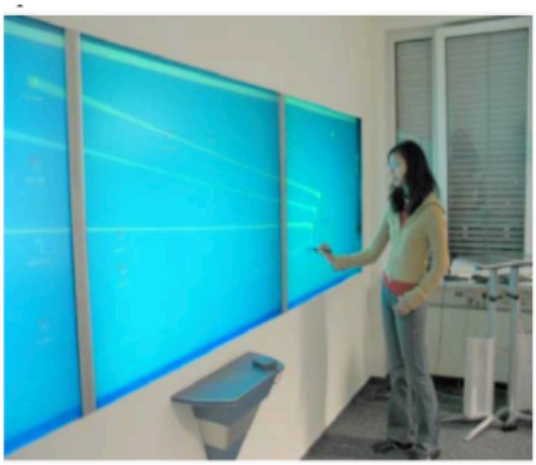

Figure 9: DynaWall setup used in user study

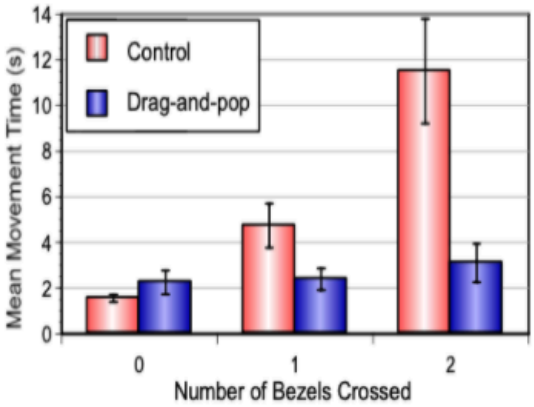

Figure 10: Mean movement time for control and drag-and-pop interfaces (± SEM).

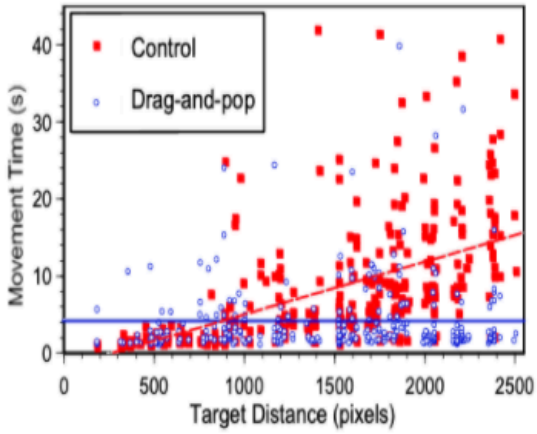

Figure 11: Movement time vs. target distance.

Conclusions and Future Work

The substantial time-savings found in the user study confirm our expectations. Although when used within a single screen unit drag-and-pop does not seem to by faster than traditional drag and drop (first pair of bars in Figure 10; drag-and-pop's capability of bridging distance to the target seems to

be nullified by the need for re-orientation), its advantages on very large screens and its capability of bridging across display units are apparent. On the usability side, we were glad to see that participants had no trouble learning how to use the technique and that they described the technique as understandable and predictable. The single biggest shortcoming, the target selection, is the subjects of current work. In addition to the changes described above, we consider dropping the notion of a fixed target sector size and replace it with a mechanism that adjusts the sector size dynamically based on the number of matching targets. Given the recent advent of commercially available tablet computers, our next step will be to explore how drag-and-pop and especially drag-andpick can help tablet computer users work with external monitors. While this paper focused on icons, we plan to explore ways of operating menus, sliders, and entire applications using the techniques described in this article.