

Double-Crossing: A New Interaction Technique for Hand Gesture Interfaces

Takashi Nakamura, Shin Takahashi, and Jiro Tanaka

Department of Computer Science
Graduate School of Systems and Information Engineering
University of Tsukuba
1-1-1 Ten-noudai, Tskuba, Ibaraki 305-8573, Japan
{takashi, shin, jiro}@iplab.cs.tsukuba.ac.jp

Abstract. We propose double-crossing, a technique for interacting with large displays using hand gestures. This technique extends crossing, which was developed for a pen-based interface, and enables the ease of use by incorporating easy hand gestures such as finger movements.

Keywords: Crossing, Hand gesture, Large display.

1 Introduction

Hand gesture recognition is a promising approach for remote interaction with large displays. Several existing interaction techniques are based on hand gestures [9, 10]; Fig. 1 shows one such method. Complex hand gestures, however, are often difficult to recognize, are prone to errors, and tend to require expensive devices such as the dataglove. Even the problem of using hand movements to control a simple pointer presents difficulties because of unavoidable hand tremors.



Fig. 1. Example of remote interaction with large display

To cope with this problem, we propose a simple new interaction technique, called double-crossing, which extends to hand gestures the crossing technique that was originally developed for pen-based interfaces[1, 2]. Double-crossing can be implemented using an ordinary web camera and an LED placed on the user's fingertip. The system

tracks the position of the user's hand accurately enough to create a robust interface. We additionally designed a set of widgets for double-crossing interaction. By combining these widgets, we can easily implement customized hand gesture interfaces for different applications.

The rest of this paper is organized as follows: In section 2, we introduce the double-crossing interaction. In section 3, we describe the Hand-Bin prototype system. In section 4, we discuss the evaluation of the technique. In section 6, we summarize the paper.

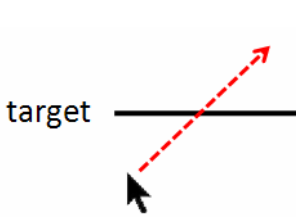


Fig. 2. (single) crossing

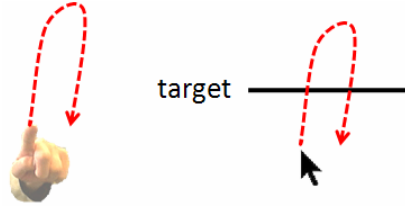


Fig. 3. (left) Hand Movement for double-crossing
(right) Movement of pointer and double-crossing

2 Double-Crossing

One problem encountered in hand gesture interfaces is that it is difficult to stabilize the user's hand at a fixed position. It is therefore hard to move a pointer to follow the hand's position and to select or click at the intended location accurately. To solve this problem, a pen stroke is used instead of clicking [7, 11]. In particular, we focus on the crossing technique, in which a crossing bar motion is made to trigger a command (see Fig. 2).

In our hand gesture interface, the hand movement of crossing is recognized by the camera so that it can be used to press a button or select a menu item in the GUI. To implement this idea, however, we must cope with unintentional hand movements that may be recognized as crossing. We hence propose to use a double-crossing action, which is a crossing bar motion that occurs twice during a short time interval. Since unintentional double-crossing occurs much less frequently than unintended crossing, we will show that double-crossing is effective as a basic interaction primitive for hand gesture-based graphical user interfaces. Double-crossing is a local finger gesture that moves a finger up-and-down, left-and-right; this action is simple enough that the user does not need to train the system (see Fig. 3).

3 Hand-Bin

We developed a prototype interface based on double-crossing, called Hand-Bin. Hand-Bin is a widget overlaid onto the Windows GUI (see Fig. 4). It consists of three types of components: click icon at the center of Fig. 5 (see Fig. 7 for details), menu icons around the click icon (see Fig. 8 for details), and toggling bars (two bars at the top-left of Fig. 5). The Hand-Bin widget always follows the pointer.

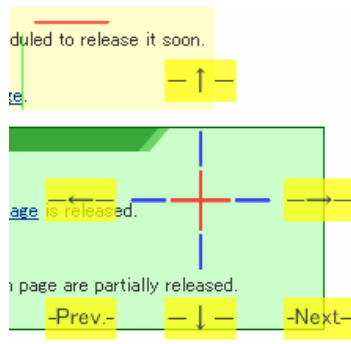


Fig. 4. Hand-Bin overlayed on Windows GUI

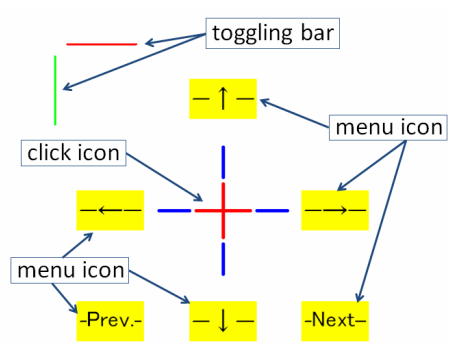


Fig. 5. Detail of Hand-Bin

The widget shown in Fig. 5 is an example of Hand-Bin interface designed for Internet browsing. We can easily design other widgets for other applications by arranging menu icons and the click icon for specific applications.

Fig. 6 shows an implementation of the Hand-Bin interface. The user has an LED placed on a fingertip, and the system detects the position of the LED by using images captured from the web camera. The system moves the pointer and detects double-crossing using the captured movements.

3.1 Click Icon

As shown in Fig. 7, the click icon consists of two parts: the cross hairs at the center and four click bars around the cross hairs. The Hand-Bin click event emulates a mouse left-button click occurring at the center of the cross hairs when the user double-crosses one of the click bars. The cross hairs can be moved by pushing them with the pointer. Because of hand tremors, it is difficult to fix the pointer at a single position, so the cross hairs do not move directly with the pointer; this allows the user more precisely to move the cross hairs to an intended location.

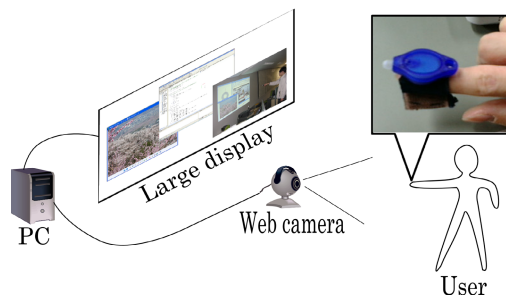


Fig. 6. System setting

3.2 Menu Icons

Menu icons are buttons in the Hand-Bin interface that trigger commands such as scrolling a page and switching tabs. Fig. 5 shows the Hand-Bin interface for Internet browsing. The up and down arrow buttons are scrolling buttons, the left and right arrow buttons are used for switching tabs, and the remaining two are the “previous page” and “next page” buttons in the web browser.

By double-crossing the invoking bar at the middle of the button, the user can execute the command assigned to the button. Although an invoking bar appears to be divided into two parts by the label, it is in fact one horizontal bar at the middle of the button. When invoking one command repeatedly, the second and following executions are invoked each by only one crossing of the invoking bar. The user can thereby quickly repeat commands. One double-crossing of the scrolling button, for example, scrolls the web page slightly, but the repeated up-and-down finger gesture enacted on the scroll button executes the scroll command repeatedly so the user can scroll the web page to the intended position. This repeated mode of action ends when the crossing does not occur for some time period.

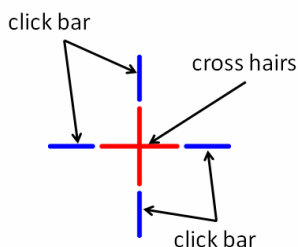


Fig. 7. Click icon

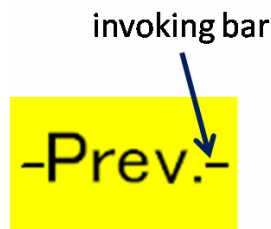


Fig. 8. Menu icon

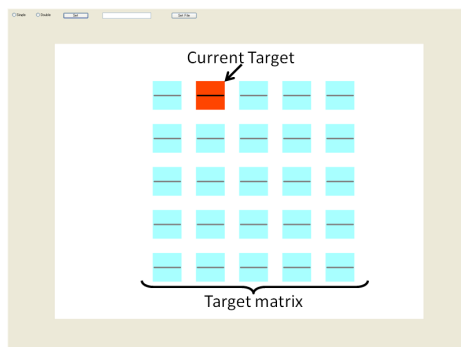


Fig. 9. The screen of the experiment

3.3 Toggling the Display of the Widget

The display of the click icon and the menu icons is toggled by double-crossing one of two toggling bars at the top-left of the Hand-Bin widget. The upper horizontal bar is

used to toggle the display of the click icon, and the lower vertical bar is used to toggle the display of the menu icons. The click icon and the menu icons are not normally displayed, leaving only the two toggling bars at the top-left.

3.4 Movement of the Hand-Bin Interface

The Hand-Bin widget is always displayed near the pointer; when the pointer moves away from the Hand-Bin widget, the widget moves to follow. In addition, we developed two methods by which the user can finely adjust the position of the cross hairs. One method uses the boundary region and the other uses the cross hairs click icon.

3.4.1 Movement Using the Boundary Region

The boundary region in the Hand-Bin widget is implemented to follow the pointer, so that the pointer always occupies a boundary region. When the pointer crosses over the boundary region, the Hand-Bin widget moves according to new coordinate with respect to direction. This method could look like pulling the boundary region and the Hand-Bin widget.

3.4.2 Movement Using the Cross Hairs

The Hand-Bin widget can be moved by placing the cross hairs of the click icon in the boundary region, with the intent being to move the cross hairs to a place where a user wants to click. In this method, the Hand-Bin widget is moved when the pointer meets the cross hairs, moving as if the cross hairs were pushed by the pointer.

3.5 Feedback

It is important to give the user feedback to register which operation is being carried out, and Hand-Bin uses sound for feedback. Because allocating a different sound to each possible operation might cause confusion when the number of sounds being emitted increases, we use only one kind of sound for all operations. The issue of proper feedback generally deserves further consideration.

4 Evaluation

4.1 Double-Crossing Experiment

4.1.1 Methods

We prepared 5 x 5 targets such as in Fig. 9. The sizes of the targets were 100 pixels by 100 pixels, with 50 pixels between targets. The experiment was performed with 10 users. Nine of the users were not familiar with interaction using hand gestures, whereas one user was well versed in hand gesture interfaces because he had conducted research in this area.

In the experiment, users selected specified targets using crossing. The specified target was the red target in Fig. 9. We used two crossing techniques: the existing single-crossing technique and the proposed double-crossing technique.

For each crossing technique, we proceeded as follows. First, users practiced 10 times before attempting five sets. In each set, they were given 20 targets. We measured the time taken and the error rate of object selection.

When a target that was not the specified target was selected, we labeled it as an error. The error rate was calculated as (number of trials in which a wrong target was selected)/(total number of trials), and ranged from zero to one. If a user crossed once over a wrong target, it was counted as an error in single-crossing, but was not counted as an error in double-crossing.

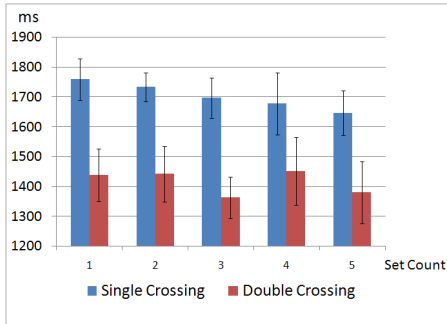


Fig. 10. Averages of selection time

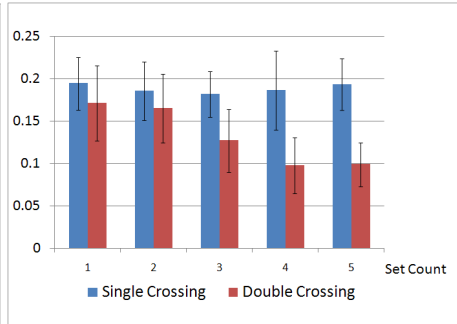


Fig. 11. Averages of error rate

4.1.2 Results

Fig. 10 and Fig. 11 show the selection time and the error rate for each set, both averaged over the 10 users.

Selection time: When we compared selection time, double-crossing was faster than single-crossing, with a difference of more than 150 ms. The average selection times for the 10 users were approximately 1702 ms in single-crossing and approximately 1414 ms in double-crossing. With single-crossing, the selection time became shorter as the number of sets increased, whereas double-crossing did not yield any such difference in selection times as the number of sets increased.

Error rate: The error rate of double-crossing was lower than that of single-crossing in all sets. The average error rates over the 10 users were approximately 0.188 for single-crossing and approximately 0.132 for double-crossing. With single-crossing, the error rate did not change substantially as the number of sets increased. In contrast, the error rate for double-crossing decreased greatly as the number of sets increased: the error rate on the first set was approximately 0.171, whereas that on the fifth set was approximately 0.099.

4.1.3 Discussion

The selection time and error rate for double crossing were better than those for single-crossing. It was possible for users to move the pointer to the specified target in a straight line in the case of double-crossing, as in Fig. 12(a). In contrast, with single-crossing, the users often had to make a detour so as not to select other targets, as in Fig. 12(b) (c). These likely accounts for the higher selection time for single-crossing than for double-crossing.

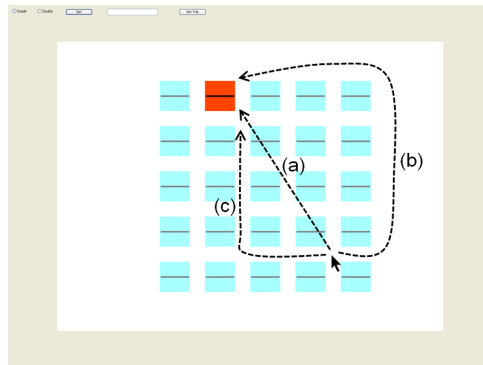


Fig. 12. How to move the pointer to the specified target

Moreover, movements such as those in Fig. 12(c) require users to execute fine motions. As compared to a mouse, however, the pointer wobbles easily when moving by hand gestures, so users often crossed unintended targets. As a result, the error rate of single-crossing was higher than that of double-crossing.

In double-crossing, the selection time did not change substantially although the error rate decreased greatly as the number of sets increased. We observed that users did not change the way in which they moved the pointer to the specified target. Some users, however, changed the manner of double-crossing as the number of sets increased. They moved their fingers too little at first and crossed the same target multiple times, often generating an error. However, they learned how to move the pointer for double-crossing as the number of sets increased and ceased to do erroneous crossing. This caused the error rate for double-crossing to decrease gradually.

4.2 Realistic Hand-Bin Evaluation

In addition to the evaluation experiment, we allowed various people to use a prototype Hand-Bin interface with double-crossing. We found that people who used the Hand-Bin interface for the first time were able to use it without discomfort.

We discovered one problem in particular: many users felt that the click operation using the click icon was odd. This might possibly be explained by the fact that users had to complete two steps to click: first, to move the cross hairs to the place where they wanted to click; and second, to execute the click operation by using the click bar. It was difficult for users to move the pointer precisely by hand when moving the cross hairs to the desired location. It will therefore be necessary to devise a clicking method that does not use the click icon. We propose several possible solutions here.

4.2.1 Improvement of the Click Operation

A simple improvement to the click operation would be to execute the click at the location of the pointer whenever it stays in one place for a fixed duration (see Fig. 13(a)). With hand movements, however, it is difficult for users to fix the pointer in one location. Moreover, because this technique is slow to execute, it will not be an effective method.

A click could also be executed at a target when it is enclosed by the pointer. If, for example, a user wants to click some targets or links, he or she can enclose them as in Fig. 13(b) by moving the pointer with hand gestures. The click operation is executed at the center of the enclosed place (marked by an 'x' in Fig. 13(b))

Another option is to execute a click wherever the user draws an 'x', for example, moving the pointer as in Fig. 13(c). The click operation is executed at the intersection generated by the trajectory of the pointer (given by the 'x' in Fig. 13(c)).

In both of these techniques, the click execution and the specification of the click location are accomplished simultaneously. The influence of the user's unintentional hand movements will be small because these techniques use moving gestures as opposed to still hand poses.

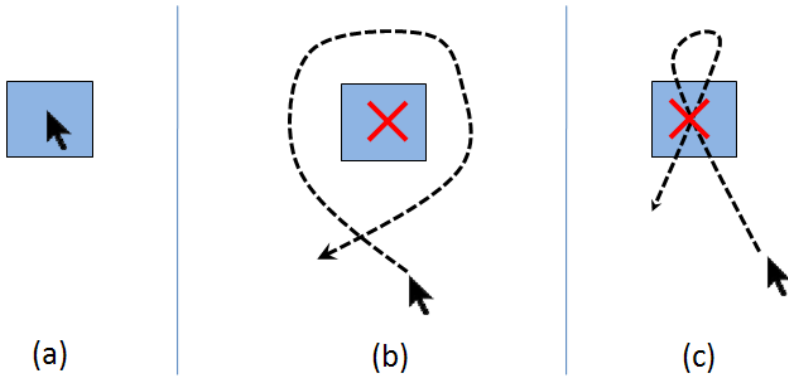


Fig. 13. A variety of click techniques: (a) immobilizing the pointer on the target, (b) enclosing the target using the pointer, (c) making an 'x' over the target

5 Related Work

Sören et al. [8] proposed an interaction technique that used hand gestures and FlowMenu [5, 4] to operate consumer electronics remotely. They used the direction of the finger for interactions and to operate FlowMenu. This method, however, takes a substantial amount of time, and it is difficult to use a continuous direction of one's finger to interact.

Dhawale[3] devised a technique for interacting with displays using only hand gestures. In this system, interaction with the display was accomplished by detecting the movement and inclination of the hand in three dimensions. Although the user can easily figure out the positional relation between the pointer and the hand, i.e., the position of the hand is projected on the display, operation becomes complex because gestures are tracked in three dimensions. Moreover, this technique requires a large horizontal space, which makes interaction difficult in the presence of obstacles such as a desk. This technique is not well suited to operation at a distance.

Hisamatsu et al.[6] used a laser pointer, crossing, and encircling to enable interaction. This technique executes a specific operation when the trajectory of the laser pointer crosses the edge of the screen or encircles an object. We have drawn

upon these techniques in our work. When using hand gestures, however, it is very difficult to move the pointer to a distant location instantaneously and directly as when using laser pointer, which has an on/off switch. We have designed the crossing technique with this problem in mind.

6 Summary

We proposed a hand gesture interface based on double-crossing for remote interaction with large display systems. We also described a prototype system called Hand-Bin. A Hand-Bin widget consists of a click icon and menu icons that can be tailored to specific applications. Hand-Bin works with the common Windows GUI and can be implemented with only a low-cost web camera and an LED, yet it works robustly in a real usage environment. We plan to conduct further formal evaluation of the usability of the Hand-Bin system and to improve the Hand-Bin system with intelligent cross hairs that recognize web pages.

References

1. Accot, J., Zhai, S.: More than dotting the i's - foundation for crossing-based interfaces. In: CHI 2002, pp. 73–80 (2002)
2. Apitz, G., Guimbretiere, F.: CrossY: A Crossing-Based Drawing Application. In: UIST 2004, pp. 3–12 (2004)
3. Dhawale, P., Masoodian, M., Rogers, B.: Bare-Hand 3D Gesture Input to Interactive Systems. In: ChinZ 2006, pp. 25–32 (2006)
4. Guimbretiere, F., Martin, A., Winograd, T.: Benefits of merging command selection and direct manipulation. *ACM Trans. Comput.-Hum. Interact.* 12(3), 460–476 (2005)
5. Guimbretiere, F., Winograd, T.: FlowMenu: Combining command, text, and data entry. In: UIST2000, pp. 213–216 (2000)
6. Hisamatsu, T., Shizuki, B., Takahashi, S., Tanaka, J.: A novel click-free interaction technique for large-screen interfaces. In: APCHI 2006 (2006)
7. Kurtenbach, G., Buxton, W.: The limits of expert performance using hierarchic marking menus, CHI 1993, pp. 482–487 (1993).
8. Lenman, S., Bretzner, L., Thuresson, B.: Using Marking Menus to Develop Command Sets for Computer Vision Based Hand Gesture Interfaces. In: NordiCHI, pp. 239–242 (2002)
9. Malik, S., Ranjan, A., Balakrishnan, R.: Interacting with Large Displays from a Distance with Vision-Trackd Multi-Finger Gestural Input. In: UIST 2005, pp. 43–52 (2005)
10. Vogel, D., Balakrishnan, R.: Distant Freehand Pointing and Clicking on Very Large, High Resolution Displays. In: UIST 2005, pp. 33–42 (2005)
11. Zhao, S., Agrawala, M., Hinckley, K.: Zone and Polygon Menus: Using Relative Position to Increase the Breadth of Multi-Stroke Marking Menus. In: CHI 2006, pp. 1077–1086 (2006)