Evaluation of gaze-added target selection methods suitable for general GUIs

A. Monden and K. Matsumoto
Graduate School of Information Science,
Nara Institute of Science and Technology,
8916-5, Takayama, Ikoma, Nara 630-0101, Japan
E-mail: akito-m@is.aist-nara.ac.jp E-mail: matumoto@is.aist-nara.ac.jp

M. Yamato
Red Hat Japan, Inc., Shinkanda Building 3F,
2-15-2, Sotokanda, Chiyodaku, Tokyo 101-0021, Japan
E-mail: jet@gyve.org

Abstract: This paper examines three gaze-added methods, the Auto, Manual, and SemiAuto, that have a potential to increase the efficiency of target selection operations in general GUI environments such as MS-Windows and Mac-OS. These three methods employ the human's eye gaze and a hand (mouse operation) together to enable users to select a target even if the jittery motions of an eye and the measurement error of an eye-tracking device have occurred. The result of the experiment under an environment in which small targets are placed in a narrow layout showed that the operational efficiency with the SemiAuto method was the best among three methods; and, the SemiAuto method was the same or faster than the mouse-only operation without increasing errors greatly. Especially concerning the discontinuous selection situation (a target selection whose cursor position is randomly located), the operation with the SemiAuto method was about 31% faster than the mouse-only operation.

Keywords: human-computer interaction; user interface design; eye-gaze interface; eye tracking device; desktop environment; mouse operation; input device; usability.

Reference to this paper should be made as follows: Monden, A., Matsumoto, K. and Yamato, M. (2005) 'Evaluation of gaze-added target selection methods suitable for general GUIs', *Int. J. Computer Applications in Technology*, Vol. 24, No. 1, pp.17–24.

Biographical notes: Akito Monden received the BE degree (1994) in electrical engineering from Nagoya University, Japan, and the ME degree (1996) and DE degree (1998) in information science from Nara Institute of Science and Technology, Japan. He is currently Assistant Professor in the Graduate School of Information Science at Nara Institute of Science and Technology. His research interests include human-computer interaction, human factors in software development, and protection of intellectual properties of software. He is a member of the ACM, IEEE, the Institute of Electronics, Information and Communication Engineers (IEICE), the Information Processing Society of Japan (IPSJ), and the Japan Society for Software Science and Technology (JSSST).

Ken-ichi Matsumoto received the BE, ME, and PhD degrees in information and computer sciences from Osaka University, Japan, in 1985, 1987, and 1990, respectively. He is currently professor in the Graduate School of Information Science at Nara Institute of Science and Technology, Japan. His research interests include software measurement environment and software productivity. He is a member of the ACM, IEEE, the Institute of Electronics, Information and Communication Engineers (IEICE) and Information Processing Society of Japan (IPSJ).

Masatake Yamato received the BE degree (1997) in information engineering from Ritsumeikan University, Japan, and the ME degree (1999) and DE degree (2002) in information science from Nara Institute of Science and Technology, Japan. He is currently

working for Red Hat Japan, Inc. His research interests include eye-gaze interface and open source software development. He is a member of the IEEE and Institute of Electronics, Information and Communication Engineers (IEICE).

1 INTRODUCTION

In recent years, user interfaces employing the humans' eye gaze as an input have become a popular topic among HCI researchers (Salvucci, 1999; Velichkovsky and Hansen, 1996; Vertegaal, 1999; Ward. and Mackay, 2002). Such interfaces, called eye gaze interfaces, are categorired into two types: (1) Gaze-centered interface, and (2) Gaze-added interface (Salvucci and Anderson, 2000). In Gaze-centered interface, all of the user's operations are done by eye-gaze only (Hansen et al., 1995; Ohno, 1998; Partala et al., 2001). The user of such interfaces only needs to look at the GUI for selecting buttons and/or menu items, zooming in/out a window, scrolling a window, and so on (Ebisawa et al., 1996; Goldberg et al., 1995; Yamato et al., 1999). Although most interfaces of this type are less efficient than conventional interfaces (using a mouse and a keyboard), they are useful not only for the person who has physical handicaps in his/her hand, but also for the person who does not have enough working space for operating a mouse and a keyboard to use the computer. On the other hand, gaze-added interface employs both an eye and a hand together for GUI operations to make the interface more efficient than conventional interfaces (Salvucci and Anderson, 2000; Zhai et al., 1999). For example, moving a cursor onto a target with eye gaze is much faster than the conventional mouse-only operation. Since the speed of eye movements are about 350 degree/s to 500 degree/s, it takes only about 150 ms to move the gazing point from a corner to the opposite corner of 21 in. size display (Ohno, 1999).

The purpose of this paper is to establish an efficient gaze-added interface suitable for general GUIs such as MS-Windows, Mac-OS, and X-Window, etc., while conventional gaze-added interfaces use specialired GUIs where huge GUI items (icons, menu items, and buttons) are put in a wide layout (Silbert and Jacob, 2000). Since most computer users are commonly using application software on general GUIs, realiring an eye gaze device in general GUIs will be much more useful for many computer users. In this paper, the 'general GUI' is regarded to have the following characteristics:

(C1) the size of each target (GUI item) is 1 cm^2 or smaller
(C2) the distance between targets is 0 cm or greater (0 cm distance means targets are verging on each other).

Since there are many small icons and buttons scattered on the general GUIs, two problems arise in realiring a gaze-added interface in general GUI environments as follows:

(p1) *The measurement error of the eye-tracking device*

The measurement accuracy of present eye tracking devices is not high enough to keep pointing at a very small target. In general, the accuracy is about 0.3 degree to 1.0 degree as the angle of view. In the case that a user sits 50 cm away from a computer display, the error on the display is about 0.5 to 1.7 cm (Ohno, 1999; Yamato et al., 2000). Considering that the size of a typical GUI button used in general GUIs like MS-Windows is about 1 cm^2, the accuracy of the eye-tracking device is not high enough.

(p2) *Jittery motions of the user's eye*

Even if a user thinks that he/she is staring at a certain point on the display, actually it is not true because his/her eye makes jittery motions in the range of 0.4 cm^2 in the case a user sits 50 cm away from a computer display (Yamato et al., 2000). This makes it difficult for users to point exactly at a very small GUI button with an eye.

In this paper, we conduct an experimental evaluation to compare the efficiency of the three target selection methods (Auto, Manual, and SemiAuto) that have the potential to overcome the above problems, p1 and p2, under the GUI environment, while satisfying C1 and C2. As a basis of the three methods, this paper also illustrates the Basic method. Brief overviews of the four methods are below:

0 *Basic method*

The user's eye gaze is used for moving a cursor onto a target; and, the user's hand is used for clicking a mouse button so that the target is activated (selected). Hence, in order to select a target, the user has simply to look at it and click the mouse button. This method is a traditional way to incorporate the eye and the hand together in doing the target selection (Jacob, 1990).

1 *Auto method*

This method enables the user to select a nearest target even if the gazing point is not upon the target. When a user clicks a mouse button, the cursor will automatically jump onto the nearest target and the target will be immediately selected.

2 *Manual method*

This method enables the user to switch the input device from the eye to the mouse. Hence, the user can move the cursor roughly by an eye, and then manually move it onto the target precisely by a mouse operation.

3 *SemiAuto method*

This method combines the Auto and the Manual method. The user can switch the input device from the eye to the mouse anytime he/she wanted, and select the nearest target by clicking the mouse button.

Among the three gaze-added methods (Auto, Manual, and SemiAuto), the SemiAuto method is newly proposed in this paper. On the other hand, the idea of the Auto method was previously mentioned by Salvucci and Anderson (2000), and also in our early researches (Yamato et al., 2000;

Yamato et al., 2000). Also, a similar method as the Manual method has been proposed by Zhai et al. (1999). However, in previous studies, these methods were not evaluated in the general GUI environments that satisfy C1 and C2. In our evaluation, we prepared a GUI satisfying C1 and C2; and, compared these methods with mouse-only operations.

The reminder of this paper first illustrates the details of the above four gaze-added methods (Section 2). Next, it describes an experiment to evaluate the methods in general GUI environments (Section 3). Then we introduce some related work (Section 4); and in the end, the conclusion and future topics will be shown (Section 5).

2 GAZE-ADDED TARGET SELECTION METHODS

2.1 Basic method

The Basic method allows users to use their eye and hand together. In this method, a target selection operation is divided into two operations: a move operation (moving the cursor onto the target), and an act (= activation or action) operation (pressing or releasing a mouse button to activate the target). An eye tracking device is assigned to the move operation, and a mouse button is assigned to the act operation (Figure 1). Under this policy, in order to select a target, for example an icon, a user has simply to look at the icon and click the mouse button.

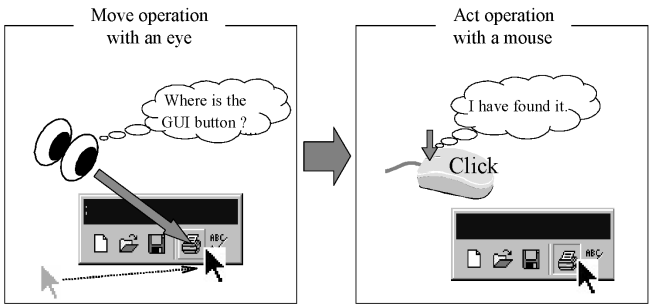

Figure 1 *Basic method*

Since moving the cursor with an eye is much faster than with a mouse, the eye tracking device is suitable to the move operation. On the other hand, on account of problems p1 and p2 described in Section 1, the eye-tracking device is not suitable to the act operation; and, the mouse is more preferable for this purpose.

2.2 Auto method

In this method, the nearest target from the place where the user is looking (gazing point) is automatically selected when the user performed an act operation. Even if the gazing point is out of an area of the target because of the measurement error and/or jittery motions, the nearest target from the user's gazing point will be selected (Figure 2). This method is useful especially in the case when the target is very small. In this method, the closest target is automatically highlighted so that the user can know which target is the closest. By using the Auto method, the user does not need to look at the target precisely. The user can roughly look at the target then click the mouse button to select it.

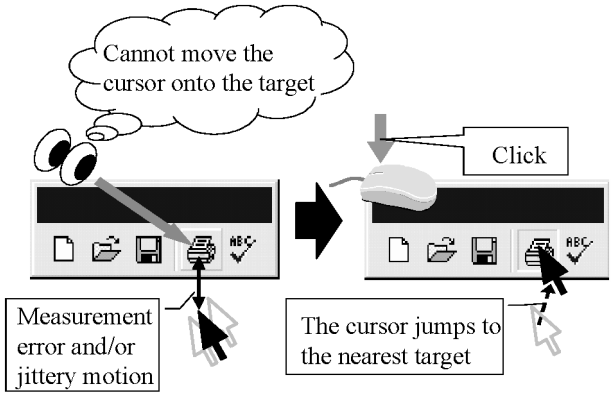

Figure 2 *Auto method*

2.3 Manual method

In this method, the user basically moves the cursor with the eye; however, the input device is switched from the eye-tracking device to the mouse when the user moves the mouse with his/her hand. The user can use the eye-tracking device for rough cursor movement and can use the mouse for delicate adjustments (Figure 3). The behaviour of the cursor in the manual method is same as the Basic method if the user did not move the mouse. If the user found that it is difficult to move the cursor onto the target because of the measurement error and/or jittery motions, the user can manually switch the input device into the mouse. If the user switches the input device properly, the selection misses may be reduced. In this method, after the user did an act operation by clicking a mouse button, the input device automatically switches back to the eye-tracking device so that the user can become ready for selecting the next target.

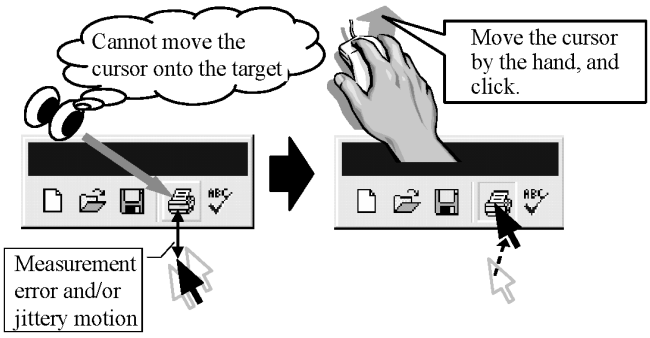

Figure 3 *Manual method*

2.4 SemiAuto method

This method is a combination of the Auto method and the Manual method (Figure 4). Just as the Auto method, the nearest target from the cursor is automatically selected when

the user clicks a mouse button. Also, just as the Manual method, the input device is switched from the eye tracking device to the mouse when the user moves the mouse. The user of the SemiAuto method can roughly point at the target either with the eye or the mouse then click the mouse button to select it.

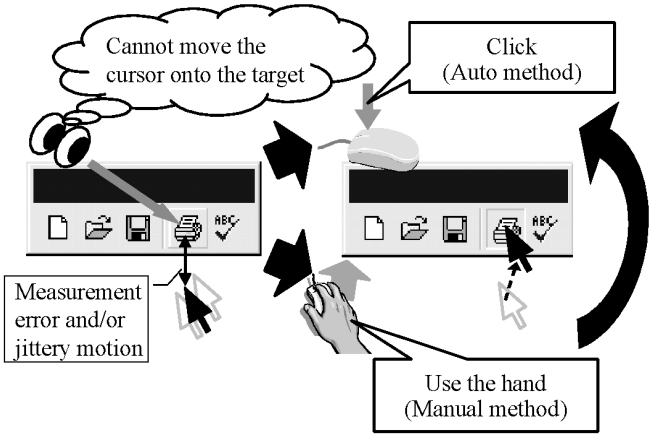

Figure 4 *SemiAuto method*

3 EXPERIMENT

The purpose of this experiment is to evaluate the efficiency of the three target selection methods (Auto, Manual, and SemiAuto) in the general GUI environment.

3.1 Design of experiment

3.1.1 Target selection methods

In this experiment we used four target selection methods:

1 Mouse-only operation
2 Auto method
3 Manual method
4 SemiAuto method

3.1.2 Task

Nine square targets, whose sizes are 1 cm^2, are put in a GUI window on the desktop of MS-Windows (Figure 5). As a single task of this experiment, each subject is asked to select a highlighted target out of nine targets 50 times. The highlighted point will change randomly after selecting a target. Subjects are told to select a target quickly and accurately. In case a subject made a miss in selecting a target, the window beeps as an alert and the subject must re-select the correct target. Here we regard it as a miss when the subject did an act operation on a target that is not highlighted or the subject did it outside of the target area.

We prepared two kinds of tasks for the mouse-only operation supposing two different situations: (1) The user selects different targets continuously (we call this *continuous selection situation*), and (2) The user selects a target after a non-target selection task, such as text input with a keyboard (we call this *discontinuous selection situation*). In the latter situation, the user must at first find a cursor on GUI before starting to select a target. In order to simulate this situation, we prepared a GUI that automatically resets the cursor position randomly after the user selected a target. This cursor position resetting was not used in the continuous selection situation. In the other three methods (Auto, Manual, and SemiAuto) we did not reset the cursor position because the cursor is basically set to the user's gazing point and it is meaningless to reset the cursor position.

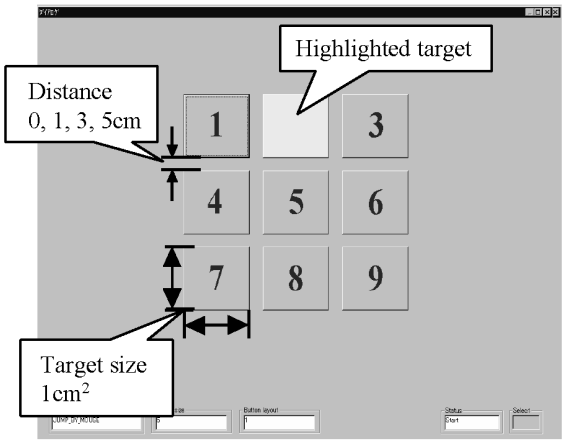

Figure 5 *GUI layout*

3.1.3 Target size and layout

The size of the target is 1 cm^2. Four kinds of target layout patterns are prepared by varying the distance between the targets: 0 cm, 1 cm, 3 cm, and 5 cm (Figure 5).

3.1.4 Subjects

The subjects are four graduate students and one faculty member of Nara Institute of Science and Technology. All five subjects usually use MS-Windows and are familiar with the operations using a mouse.

3.1.5 Environment

We used a 21 in. display (CRT). The resolution of the display is 1024 × 768 pixels, and the size of the display area is about 30 cm in height and 40 cm in width. The distance between the subjects' head and the display is about 50 cm. The subjects' computer is a DELL Dimension XPS R450 (Pentium II 450 MHz) and the Operating System is MS-Windows98.

As an eye-tracking device, we used the EMR-NC, which was developed jointly by NAC Co. and NTT (Figure 6). In order to record the eye-gaze, the users wore a lightweight triangular frame. The EMR-NC detects the triangular frame by using two cameras, one on either side of the display, and controls an active mirror to track their eye position. This mechanism makes it possible to track the users' eye marks with a high accuracy, of about 0.3 degrees, and it allows the users to move their heads freely. The eye mark data is recorded 30 times per second (30 Hz) and sent to the user's computer in real-time.

Figure 6 *Eye tracking device EMR-NC*

3.1.6 Procedure

Before executing a task, the subjects are asked to thoroughly train in each method (including mouse-only operation). Therefore, the results of the experiment is supposed to have very little influence by the level of experience for each task.

Each subject performs 20 tasks in total: 5 target selection methods (Mouse-only (continuous selection situation), Mouse-only (discontinuous selection situation), Auto, Manual, and SemiAuto) in four different target layouts (distance between targets = 0 cm, 1 cm, 3 cm, and 5 cm).

In order to lessen the influence of the learning effect in each task, we decided the execution order of the tasks be based on the following policy.

- the execution orders of target selection methods are varied in each subject
- the execution orders of target layouts in tasks are varied in each subject.

3.2 Result of experiment

3.2.1 Comparison among gaze-added methods

3.2.1.1 Task completion time

Figure 7 shows the relation between the average target-selection time and the distance between the targets. The Auto method showed a good performance level when the distance between the targets is wide (more than 3 cm), however, the performance went poorly in narrowly distanced layouts. Especially concerning the 0 cm distance layout, the Auto method required more than twice as much time as the Manual and the SemiAuto method did (significant by $p < 0.05$). Since the targets are verging on each other in the 0 cm distance layout, it is often difficult for subjects to move the cursor onto or near the highlighted target.

In the Manual method, the average selection time was short in the 0 cm distance layout; however, it became longer as the distance got wider. Especially in the 5 cm distance layout, the Manual method required about 1.5 times as much time as the Auto and the SemiAuto method did ($p < 0.05$). One possible interpretation for this result is that the subjects needed extra time to find a highlighted target and to switch the input device from the eye to the mouse in the case of the wider layout.

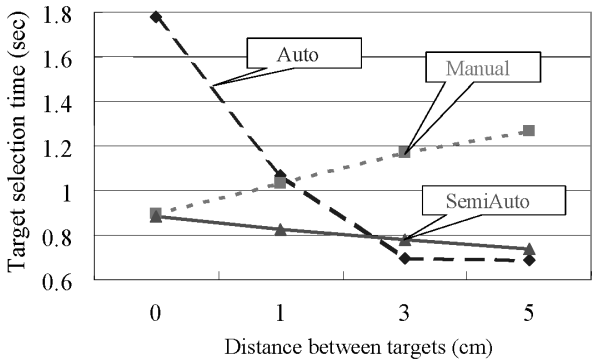

Figure 7 *Comparison among gaze-added methods in selection time*

In the SemiAuto method, the average selection time was relatively low and was stable in every target layout. In the 0 cm distance layout, the selection time was almost same as the Manual method. In the 1 cm distance layout, the Semi-Auto method required less time than the Auto and the Manual method. In the 3 cm and 5 cm distance layout, the SemiAuto method required a little more time than the Auto method.

Considering that the distance between the targets (icons, menu items and buttons) on the general GUI is quite various, the SemiAuto method is preferable for the general GUI environment since this method showed a relatively good performance in every target layout.

3.2.1.2 Misses

Figure 8 shows the relation between the average number of misses per one target selection and the distance between the targets. There was no significant difference between the Manual method and the SemiAuto method. On the other hand, the misses in the Auto method were increased as the distance became narrower. Especially in the 0 cm distance layout, the misses in the Auto method is more than 10 times greater than that of the Manual and the SemiAuto method ($p < 0.05$).

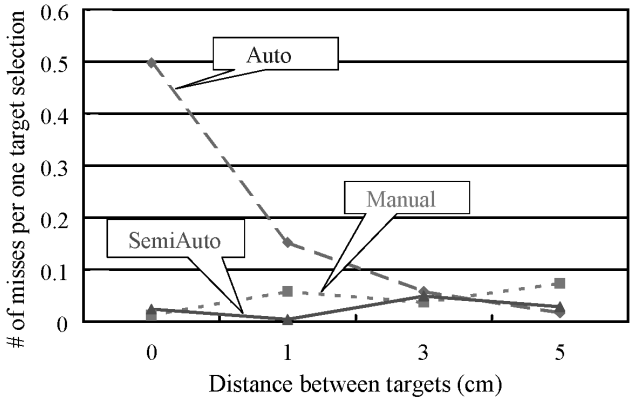

Figure 8 *Comparison among gaze-added methods in the number of misses*

Considering that the targets on the general GUI are sometimes verging on each other, the Auto method is not feasible for the practical use because of excess misses.

As a conclusion of the results from the target selection time and the number of misses, the SemiAuto method is the most preferable for the general GUI environments among the three gaze-added methods.

3.2.2 Comparison with mouse-only operations

In this section we compare the SemiAuto method, which is the most preferable one among the three gaze-added methods, with mouse-only operations.

3.2.2.1 Task completion time

Figure 9 shows the relation between the average target-selection time and the distance between the targets. In the case of discontinuous selection situation, the SemiAuto method was about 31% faster than the mouse-only operation ($p < 0.05$). One of the reasons for this result is that while the subject needed to find the mouse cursor in the mouse-only operation, the subject did not need it in the SemiAuto method.

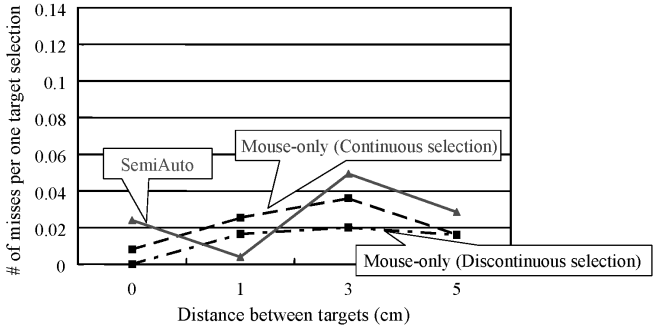

Figure 9 *Comparison between SemiAuto and mouse-only operations in selection time*

In the case of continuous target selection situation, in the 5 cm and 3 cm distance layout, the required time in the SemiAuto method was shorter than that in the mouse-only operation ($p < 0.05$). In the 1cm distance layout, there were no significant differences between the SemiAuto method and the mouse-only operation. Only in the 0 cm distance layout, the SemiAuto method required about 0.2 s longer time than the mouse-only operation ($p < 0.05$).

Except for the special situation that the user has to continuously select targets that are verging on each other (e.g., selecting GUI buttons on calculator software), the operational efficiency with the SemiAuto method is the same or faster than the mouse-only operation. Especially in the discontinuous selection situation, the operation with the SemiAuto method is much faster than the mouse-only operation.

3.2.2.2 Misses

Figure 10 shows the relation between the average number of misses per one target selection and the distance between the targets. We found that only a few misses were made in both the SemiAuto method and the mouse-only operations (less than 0.05 times in one target selection).

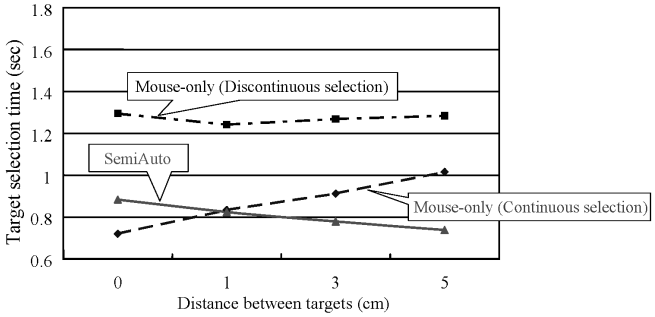

Figure 10 *Comparison between SemiAuto and mouse-only operations in the number of misses*

In almost all the target layout patterns, there were no great differences in the misses between the SemiAuto method and the mouse-only operations. In the case of continuous selection situation, there were no significant differences in all the layout patterns. On the other hand, in the case of discontinuous selection situation, only in the 0 cm and 3 cm distance layout, the misses in the SemiAuto method were greater than the mouse-only operation ($p < 0.05$).

As a result, compared with the mouse-only operation, the user of the SemiAuto method can select a target without increasing the amount of misses greatly.

4 RELATED WORK

Recently, several gaze-added interfaces for target selection have been proposed. Zhai et al. (1999) proposed an efficient method called MAGIC pointing, which employs a user's eye and hand together in selecting a target. Their method is similar to our Manual method in that the user can switch the input device from the eye to the mouse. Our SemiAuto method is different from the MAGIC in that the nearest target from the cursor is selected by an act operation even if the cursor is not on the target. Also, the environment for the experimental evaluation is different. They evaluated the efficiency of the MAGIC in a GUI environment whose targets were very small (20 pixels), however, the distance between targets in their environment was wider than 6 cm. Our experiment is different from theirs in that the distance between the targets is narrower (0 cm, 1 cm, 3 cm, and 5 cm), which is aimed to evaluate the methods in a general GUI environment. In addition, the number of misses have not been evaluated for MAGIC pointing.

Salvucci and Anderson (2000) proposed a method for selecting menu items also employing a user's eye and hand together. In their method, the move operation is done with an eye and the act operation is done with a keyboard. Their method, called intelligent gaze-added interface, utilises a probabilistic algorithm and a user model to interpret the gaze focus, while it alleviates the problems of measurement errors and jittery motions. Even in the case

when a user cannot move the cursor onto a menu item because of measurement errors and/or jittery motions, the system enables the user to select the item by judging the context of user's current task and by mapping gazes to the item that the user is likely selecting. However, from the viewpoint of pre-venting selection misses, it is difficult to adopt their method in general GUIs because it is difficult to build a valid user model in our daily GUI environment, which has so many application software and GUI items scattered on the desktop. In addition, their method has not been evaluated in the general GUI environment in which small targets are placed in a narrow layout.

Gaze-centered interfaces for target selection have also been proposed. Silbert and Jacob (2000) proposed and evaluated a method that enables users to select a target by gazing it for a moment (called dwell method). Furthermore, Hansen et al. (1995) used a special icon called 'Eyecon' to reduce the selection misses. Partala et al. (2001) proposed a method that enables users to selects a target by knitting eyebrows. Shaw et al. (1990) proposed a method for selecting a target by eye blink. These methods are very useful in a situation that the user cannot use his/her hands for operating a mouse and/or a keyboard. Although some of these methods showed a better performance level than conventional interfaces (using a mouse and a keyboard) in specially designed GUI environments, it is difficult to adopt these methods in general GUIs because of measurement errors and/or jittery motions.

5 CONCLUSION

In this paper, we examined three gaze-added methods, the Auto, Manual, and SemiAuto, that have a potential to increase the efficiency of target selection operations in general GUI environments. Through a controlled experiment under an environment in which small targets are placed in a narrow layout, we found that our SemiAuto method was the same or faster than the mouse-only operation without increasing errors greatly. Especially in the discontinuous selection situation (a target selection whose cursor position is randomly located), SemiAuto method was about 31% faster than the mouse-only operation.

Although past researches have proposed several gaze-added interfaces, most of them are evaluated only in a specialired GUI environment in which large targets are placed in a wide layout. Since most of computer users are usually using general GUIs such as MS-Windows and Mac-OS, an efficient eye-gaze interface suitable for general GUIs is needed. Our results showed that our SemiAuto method is a strong candidate that can increase the efficiency of GUI operations in our daily computer life.

One of our future projects is about controlling the appearance of the cursor. In our experiment, the cursor was always displayed on the user's gazing point; however, this may annoy the user. One of the solutions considered is to let the cursor appear only if it is near a target. Another solution was proposed by Zhai et al. (1999). In their method, the cursor will appear only in case the user touched and started to move a mouse.

Another future work is to conduct further experiments in more realistic settings, e.g., selecting desktop objects in MS-Windows and/or selecting icons on MS-Office, etc. The palmtop computers are also our target to introduce the eye-gaze. The Dasher is one of the applications that allow computer users to use eye-gaze in palmtop GUIs for the text input (Ward and Mackay, 2002). We believe our SemiAuto method is also applicable to the palmtop computers because many of them have a trackball pointer in place of a mouse pointer and the SemiAuto method is suitable for selecting the small icons of the palmtop desktop environments.

We believe incorporating an eye into the present GUI environments will bring us more useful, comfortable, and efficient interfaces in the near future. On the one hand eye-tracking devices are getting to be more accurate, cheaper and more familiar to us year-by-year; and, on the other hand conventional input devices, such as a keyboard, a mouse, a trackball, etc., have been used for a long time and will continue being used in the near future.

REFERENCES

Ebisawa, Y., Ohtani, Y. and Sugioka, A. (1996) 'Proposal of a zoom and focus control method using an ultrasonic distance-meter for video-based eye-gaze detection under free-head conditions', *Proc. IEEE Engineering in Medicine and Biology Society* (*CD ROM Edition*), IEEE Computer Society.

Goldberg, J.H. and Schryver, J.C. (1995) 'Eye-gaze Determination of User Intent at the Computer Interface', in Findlay, J.M., Walker, R. and Kentridge, R.W. (Eds.): *Eye Movement Research*, Elsevier Science, Vol. 6, pp.491–502.

Hansen, J.P., Andersen, A.W. and Roed, P. (1995) 'Eye-gaze control of multimedia systems', in Anzai, Y., Ogawa, K. and Mori, H. (Eds.): *Symbiosis of Human and Artifact, Proc. 6th International Conference on Human Computer Interaction*, Elsevier Science, Vol. 20A, pp.37–42.

Jacob, R.J.K. (1990) 'What you look at is what you get: eye movement-based interaction techniques', *Proc. Human Factors in Computing System Conference* (*CHI'90*), ACM Press, pp.11–18.

Ohno, T. (1998) 'Features of eye gaze interface for selection tasks', *Proc. 3rd Asia Pacific Computer Human Interaction* (*APCHI'98*), IEEE Computer Society, pp.176–181.

Ohno, T. (1999) 'Quick menu selection task with eye mark', *IPSJ Journal*, Information Processing Society of Japan, Vol. 40, No. 2, pp.602–612.

Partala, T., Aula, A. and Surakka, V. (2001) 'Combined voluntary gaze direction and facial muscle activity as a new pointing technique', *Proc. Human-Computer Interaction* (*INTERACT'01*), IOS Press, pp.100–107.

Salvucci, D.D. (1999) 'Inferring intent in eye-based interfaces: tracing eye movements with process models', *Proc. Human Factors in Computing System Conference* (*CHI'99*), ACM Press, pp.254–261.

Salvucci, D.D. and Anderson, J. (2000) 'Intelligent gaze-added interface', *Proc. Human Factors in Computing System Conference* (*CHI'2000*), ACM Press, pp.273–280.

Shaw, R., Crisman, E., Loomis, A. and Laszewski, Z. (1990) 'The eye wink control interface: using the computer to provide the severely disabled with increased flexibility and comfort', *Proc. 3rd IEEE Symposium on Computer-Based Medical Systems*, IEEE Computer Society, pp.105–111.

Silbert, L.E. and Jacob, R. (2000) 'Evaluation of eye gaze interaction', *Proc. Human Factors in Computing System Conference*, ACM Press, pp.281–288.

Velichkovsky, B.M. and Hansen, J.P. (1996) 'New technological windows into mind: there is more in eyes and brains for human-computer interaction', *Proc. Human Factors in Computing System Conference* (*CHI'96*), ACM Press, pp.494–503.

Vertegaal, R. (1999) 'The gaze groupware system: mediating joint attention in multiparty communication and collaboration', *Proc. Human Factors in Computing System Conference* (*CHI'99*), ACM Press, pp.254–261.

Ward, D.J. and Mackay, D.J.C. (2002) 'Fast hands-free writing by gaze direction', *Nature*, Vol. 418, p.838.

Yamato, M., Monden, A., Matsumoto, K., Inoue, K. and Torii, K. (2000) 'Quick button selection with eye gazing for general GUI environments', 16th IFIP World Computer Congress, *Proc. International Conference on Software: Theory and Practice* (*ICS2000*), Publishing House of Electronics Industry, pp.712–719.

Yamato, M., Monden, A., Matsumoto, K., Inoue, K., Torii, K. (2000) 'Button selection for general GUIs using eye and hand together, *Proc. 5th International Working Conference on Advanced Visual Interfaces* (*AVI2000*), ACM Press, pp.270–273.

Yamato, M., Monden, A., Takada, Y., Matsumoto, K. and Torii, K. (1999) 'Scrolling the text windows by looking', *IPSJ Journal*, Information Processing Society of Japan, Vol. 40, No. 2, pp.613–622.

Zhai, S., Morimoto, C. and Ihde, S. (1999) 'Manual and gaze input cascaded pointing', *Proc. Human Factors in Computing System Conference* (*CHI'99*), ACM Press, pp.246–253.