Button++: Designing Risk-aware Smart Buttons

Eunji Park
Graduate School of Culture Technology, KAIST
Daejeon, Republic of Korea
tracy1829@kaist.ac.kr

Hyunju Kim
Graduate School of Culture Technology, KAIST
Daejeon, Republic of Korea
fullmoon83@kaist.ac.kr

Byungjoo Lee
Graduate School of Culture Technology, KAIST
Daejeon, Republic of Korea
byungjoo.lee@kaist.ac.kr

CHI'18 Extended Abstracts, April 21–26, 2018, Montreal, QC, Canada
ACM 978-1-4503-5621-3/18/04.
https://doi.org/10.1145/3170427.3188645

Abstract
Buttons are the most commonly used input devices. So far the goal of the designers was to provide a passive button that can accept user input as easily as possible. Therefore, based on Fitts' law, they maximize the size of the button and make the distance closer. This paper proposes Button++, a novel method to design smart buttons that actively judge user's movement risk and selectively trigger input. Based on the latest model of *moving target selection*, Button++ tracks the user's submovement just before the click and infers the expected error rate that can occur if the user repeatedly clicks with the same movement. This allows designers to make buttons that actively respond to the amount of risk in the user's input movement.

Author Keywords
Button design; smart buttons; moving target selection; submovements; pointing; Fitts' law; temporal pointing.

ACM Classification Keywords
H.5.m [Information interfaces and presentation (e.g., HCI)]: Miscellaneous

Introduction
With the advent of graphical user interfaces (GUI), buttons have become the most widely used input device in Human-Computer Interaction (HCI). Designing better buttons is

important for usability in many areas, such as typing, gaming, and playing music [12, 9, 5, 14, 10]. Designers have mainly worked on improving the activation area [15], activation point [12, 5, 9], and location of the buttons [4]. The ultimate goal of previous studies were to provide a passive button that can accept user input as easily as possible.

Button inputs, however, cannot be undone unless a separate cancellation function is implemented in the system. This feature of button input can cause irreversible problems in critical tasks. However, it is difficult to determine whether or not an already-pressed button input is an error. User errors can only be defined by knowing the user's original intent, but in most cases the intent is not explicitly transmitted to the computer. For this reason, researchers have been trying to guess the user's intentions from a few visible indicators of user behavior.

For example, when a user performs a pointing task, the input points are distributed near the center of the target. From the variance of the input points, it is possible to understand how precisely that person wanted to press [17]. However, because the variance of an input point is a statistical value obtained by analyzing multiple button inputs, it cannot be determined for individual button clicks.

In this study, we propose Button++, a smart button that can determine how much precision (or risk) a user is willing to take for a single click. Based on the latest model of *moving target selection* [10], Button++ tracks the user's submovement just before the click and infers the expected error rate if the user repeatedly clicks with the same level of precision. In a pilot study to verify this, we conducted a time-limited pointing task. Button++ allows rejection of triggered events from high-risk input movements, thereby preventing button input errors in critical situations.

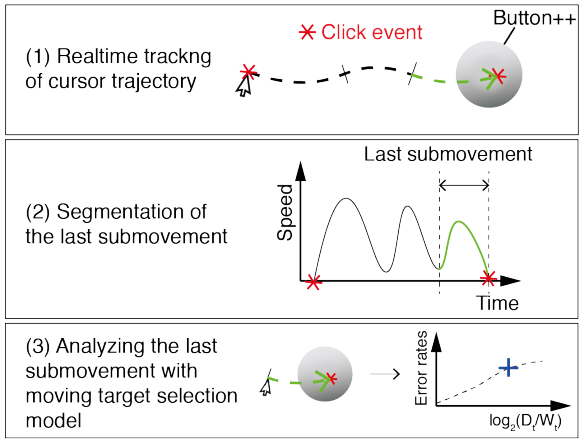

Figure 1: Overview of the technique: Button++ logs the cursor trajectory until the user clicks the button and segments it into submovements. The final submovement is then analyzed through the *moving target selection* model, from which Button++ can infer the risk taken by the user for each click. In this case, risk refers to *expected error rates* of the pointing, which depends on the user's bias in the speed-accuracy trade-off.

Related Work

The task of pressing a button necessarily involves pointing movement. The pointing task can be divided into *spatial pointing*, which selects a target in space, and *temporal pointing*, which selects a target that exists in time [12]. Here Button++ calculates the risk of the user's movement to the former task.

Many studies have been conducted to understand the dispersion of input points in spatial pointing [16, 3]. From an information theoretical perspective, they attempted to explain the speed-accuracy trade-off, which is the relationship between the time taken for pointing and the variance of the resulting input point. As users take higher risks, the point-

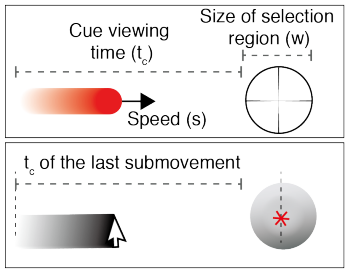

Figure 2: The last submovement that occurred just before the user clicked the button can be considered as a process of moving target selection. Using the latest model, Button++ accurately predicts the expected error rate of the last submovement.

ing time decreases, but the variance of the input points increases. Thus, if the expected variance of input points can be found for each click, it can be used as an indicator of the amount of risk a user takes.

However, the variance of the input points is a statistical value that is difficult to determine from a single pointing movement. One study [16] has attempted to express the expected variance of button inputs as a function of each pointing time, through Fitts' law. However, it is difficult to track the pointing time in a real environment [2], and their model does not take into account the independent effect of the target width.

Unlike previous studies, Button++ calculates the expected error rate of a single click based on kinematic measurements rather than statistics. Interpreting the cursor trajectory through a kinematic measure enables a deeper understanding of the phenomenon than the information theoretical viewpoint [7, 8, 11]. In particular, Button++ can reflect the effect of button size on the prediction of error rate. Button++ is also deeply related to techniques such as The Bubble Lens [13], which attempted to improve the usability of the button input by measuring the kinematics of the cursor trajectory.

Working Principles of Button++

Button++ logs the trajectory that a user moves to click the button. Immediately after the click, the trajectory is divided into several submovements by analyzing its speed profile. And from the kinematic properties of the last submovement just prior to the click, Button++ can infer the expected error rate.

More specifically, Button++ analyzes the last submovement through the recently published model of *moving target selection* [10]. In the moving target selection task, users must perform input when a moving target reaches a selection region. In this task, the target moves quickly and the user simply presses the button without moving. When the clicks that occur while the target is within the selection region are defined as successful trials, the error rate (E) can be expressed as:

$$E = 1 - \frac{1}{2}\left[erf(\frac{(1-c_\mu)}{c_\sigma\sqrt{2}} \cdot \frac{W_t}{D_t}) + erf(\frac{c_\mu}{c_\sigma\sqrt{2}} \cdot \frac{W_t}{D_t})\right]$$

$$\text{where,} \quad D_t = (\frac{1}{e^{\nu t_c} - 1} + \delta), \quad ID_t = log_2(D_t/W_t)$$

Here, t_c is the time allowed for the user to observe the movement of the target before reaching the selection region (cue viewing time), ν is the rate at which the user accepts information from the target, and δ is the baseline noise of the user's timing precision. All the values except t_c are determined through experiments and the values reported in the previous study can be used [10]. W_t is the time it takes for the target to pass through the selection region. For example, W_t is w/s when a target of speed s passes through a selection region of size w (see Figure 2).

The process from the last submovement to the click is considered to be a moving target selection task given to the user. In this case, the cursor is the moving target and the button becomes the selection region. t_c is the duration of the submovement and W_t is determined from the average speed of the submovement and the size of the button (see Figure 2). Finally the ID_t can be obtained through D_t and W_t, which is the index of difficulty of the last submovement when considered as a moving target selection task. At this time, submovement with high ID_t value is a risky input leading to a high error rate. With the individual ID values measured from each trial, Button++ can filter out risky input attempts.

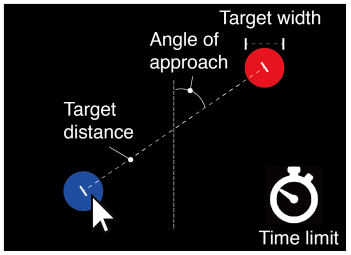

Figure 3: Participants in the pilot study performed a time limited pointing task.

Implementation
Technically, Button++ requires implementation of two functions: (1) trajectory logging, (2) submovement segmentation. First, trajectory logging is implemented through `Libpointing` library [1]. The library allows us to get the raw input value of any pointing device. In this implementation, all trajectories of the cursor are logged, but in actual deployment, it is sufficient to only track the last submovement immediately before the click. Second, the submovement was segmented by applying the `persistence1d` [6] algorithm to the speed profile of the logged cursor trajectory. For further details of the submovement segmentation, please refer to the previous study [11].

Pilot Study

In order to see if Button++ can accurately predict the expected error rate for each button click, we designed and conducted a pointing experiment according to `ISO 9241-9` standard.

Participants: Six students from the local university (2 males, 4 females) were recruited. The average age of participants was 27.2 years (σ=3.97). All the participants were right-handed.

Design: The experiment followed a 2×3×6 within-subject design with three independent variables: *target width*, *target distance* and *time limit*. The levels were the following:

- Target width: 4.8 and 8.4 mm
- Target distance: 48, 144, and 240 mm
- Time limit: 300, 400, 500, 600, 700, and 800 ms

Twenty angle of approaches were tested for each *target width-target distance* condition. As a result, participants performed 120 pointing trials for each *time limit* condition. A *time limit* condition did not change to the next condition until all corresponding width-distance conditions had been completed. The *time limit* conditions are given in random order, and within a *time limit* condition, the *target width* and *target distance* are given in random order. In the end, 4320 input events from 6 participants were logged.

Task: Participants had to select two circular targets on the screen. After clicking on the blue target, clicking on the red target ended the trial (see Figure 3). If the participant did not click on the red target within the given time limit after clicking on the blue target, the red target disappeared. Even if the red target was disappeared due to time limit violation, participants had to click to go to the next trial. If the participant clicked inside the red target (or the disappeared red target), the trial was considered successful. Participants were asked to make pointing as quickly and accurately as possible.

Apparatus: The application was implemented on a 3 GHz desktop computer (`Mac mini`, 10.13.1 High Sierra). A 27-inch LED monitor (`LG 27UD69P`) was used and the resolution of the task screen was 2560×1440 pixels. Pointing device was two-button wired optical mouse (`Samsung SNJ-B138`) with a resolution of 400 DPI. The cursor was a standard arrowhead pointing to the upper left.

Procedure: Participants performed the task with the same posture as they were using the computer. They sat on a regular office chair and the monitor was installed at the participant's eye level. Before the experiment, experimenter briefly introduced the task to the participants. A practice session was given until participants were accustomed to the task.

Result and Discussion: D_t and W_t were calculated from the t_c value and average speed value of the last submovement of each trial. At this time, other parameters were used

as they were measured in the previous study (ν=20.2, δ=0.366) [10]. Finally ID_t values and success indicators were obtained for each of 4320 trials. To calculate the error rate, we binned the indicators of success from a certain number of ID_t values.

The R^2 value of the model fit for the error rate binned from each of the 350 ID_t values was 0.949. The R^2 value for the error rate binned from 25 ID_t values was 0.823 (see Figure 4). This shows that Button++ can obtain the expected error rate with high precision for individual click events. This is in contrast to the previous model [16], which achieved a lower R^2 value (0.846) even after maximum averaging. This difference occurs because only the moving target selection model can consider the effect of the target width on the error rate.

Future Work

The study proposed Button++, a novel methodology for designing smart buttons that can infer expected error rates for a single button click. Future study will reveal more about the various benefits that Button++ can bring to its users.

Acknowledgement

This research was supported the by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2017R1C1B2002101).

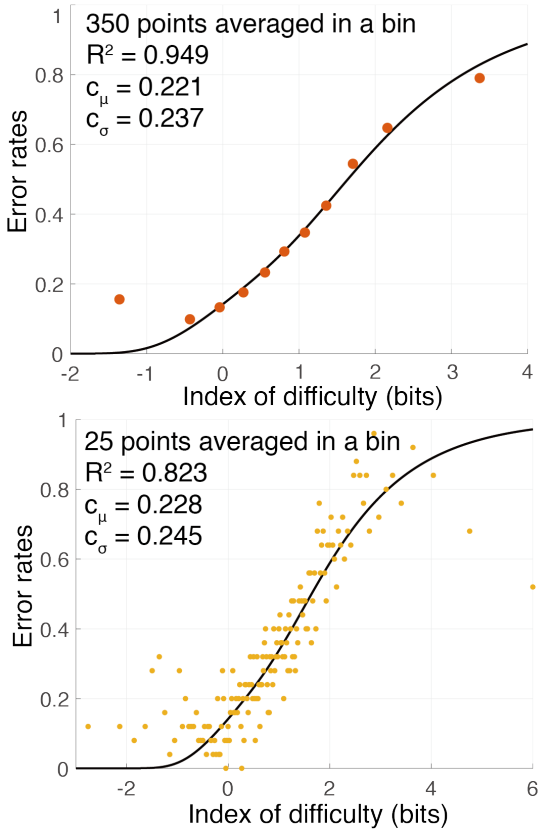

Figure 4: Results from the pilot study: Button++ did very well explain the experimental measurements, even though we did not remove the pointing outliers at all.

REFERENCES

1. Géry Casiez and Nicolas Roussel. 2011. No more bricolage!: methods and tools to characterize, replicate and compare pointing transfer functions. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 603–614.
2. Olivier Chapuis, Renaud Blanch, and Michel Beaudouin-Lafon. 2007. Fitts' law in the wild: A field study of aimed movements. (2007).
3. Yves Guiard and Olivier Rioul. 2015. A mathematical description of the speed/accuracy trade-off of aimed

movement. In *Proceedings of the 2015 British HCI Conference*. ACM, 91–100.

4. Andreas Karrenbauer and Antti Oulasvirta. 2014. Improvements to keyboard optimization with integer programming. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 621–626.
5. Sunjun Kim, Byungjoo Lee, and Antti Oulasvirta. 2018. Impact Activation Improves Rapid Button Pressing. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, to appear.
6. Y Kozlov and T Weinkauf. 2015. Persistence1D: Extracting and filtering minima and maxima of 1d functions. *h ttp://people. mpi-inf. mpg. de/ weinkauf/notes/persistence1d. html, accessed* (2015), 11–01.
7. Byungjoo Lee and Hyunwoo Bang. 2013. A kinematic analysis of directional effects on mouse control. *Ergonomics* 56, 11 (2013), 1754–1765.
8. Byungjoo Lee and Hyunwoo Bang. 2015. A mouse with two optical sensors that eliminates coordinate disturbance during skilled strokes. *Human–Computer Interaction* 30, 2 (2015), 122–155.
9. Byungjoo Lee, Qiao Deng, Eve Hoggan, and Antti Oulasvirta. 2017. Boxer: a multimodal collision technique for virtual objects. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*. ACM, 252–260.
10. Byungjoo Lee, Sunjun Kim, and Antti Oulasvirta. 2018. Moving Target Selection: A Cue Integration Model. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, to appear.
11. Byungjoo Lee, Mathieu Nancel, and Antti Oulasvirta. 2016. AutoGain: Adapting Gain Functions by Optimizing Submovement Efficiency. *arXiv preprint arXiv:1611.08154* (2016).
12. Byungjoo Lee and Antti Oulasvirta. 2016. Modelling error rates in temporal pointing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. ACM, 1857–1868.
13. Martez E Mott and Jacob O Wobbrock. 2014. Beating the bubble: using kinematic triggering in the bubble lens for acquiring small, dense targets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 733–742.
14. Antti Oulasvirta, Sunjun Kim, and Byungjoo Lee. 2018. Neuromechanics of a Button Press. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, to appear.
15. Daryl Weir, Henning Pohl, Simon Rogers, Keith Vertanen, and Per Ola Kristensson. 2014. Uncertain text entry on mobile devices. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 2307–2316.
16. Jacob O Wobbrock, Edward Cutrell, Susumu Harada, and I Scott MacKenzie. 2008. An error model for pointing based on Fitts' law. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 1613–1622.
17. Shumin Zhai, Jing Kong, and Xiangshi Ren. 2004. Speed–accuracy tradeoff in FittsâĂŹ law tasksâĂŤon the equivalency of actual and nominal pointing precision. *International journal of human-computer studies* 61, 6 (2004), 823–856.