Article

Adaptive Kinetic Scrolling: Kinetic Scrolling for Large Datasets on Mobile Devices

Jongwook Jeong, Neunghoe Kim and Hoh Peter In *

Department of Computer Science, Korea University, Seoul 02841, Korea; sique@korea.ac.kr (J.J.);
nunghoi@korea.ac.kr (N.K.)

* Correspondence: hoh_in@korea.ac.kr; Tel.: +82-2-3290-3206

Received: 22 August 2018; Accepted: 19 October 2018; Published: 23 October 2018

Featured Application: The proposed approach can help users to access target information more rapidly through the scroll widget in mobile devices. It would be used for mobile application with scrolling (e.g., music application, contact application).

Abstract: Scrolling is a frequently used Graphical User Interface widget that enables users to interact with a large amount of data using a limited viewport. However, if excessive data is included in the scroll, users are required to spend a substantial amount of time and effort to find the required information. In this paper, we present adaptive kinetic scrolling (AKS), a technique based on kinetic scrolling by which users can access target information more rapidly on mobile devices. Based on the user's behavior, AKS detects situations when the user intends to access certain information that may be distant from the current viewport. At this point, AKS amplifies the speed of kinetic scrolling. Furthermore, the scrolling speed adapts according to the size of the remaining data to be scrolled. The more data that the scrolling widget contains, the more rapidly it scrolls so that the user can quickly reach the target. Kinetic scrolling is frequently used in scrolling widgets, and with AKS, users can save time and energy wasted on repetitive meaningless scrolling. We conducted a user study and verified that the proposed scrolling technique enables users to access target information more rapidly, particularly when there is a large dataset to navigate.

Keywords: graphical user interfaces; mobile devices; activity-centered design; scrolling interface

1. Introduction

The storing and handling of large amounts of data on mobile devices is rather common at present [1]. People use their mobile devices every day, interacting with contact applications that store thousands of phone numbers or music applications that manage numerous songs. Scrolling is a typical widget that permits users to interact with a large amount of data, even on the limited viewport of a mobile device. Scrolling is a traditional Graphical User Interface (GUI) function familiar to users and is commonly used in desktop environments (e.g., web browsers and document viewers) [2].

However, scrolling is sometimes inconvenient to users. When there is excessive data to navigate, users are compelled to undergo an iterative, time-consuming scrolling process to access the necessary information. Moreover, when the viewport is small, such as on a mobile device, users generally experience inconvenience. For this reason, navigation and browsing have been considered important challenges in human–computer interaction design for mobile devices [3]. With the increasing popularity of mobile applications, scrolling in mobile devices has become considerably typical. At the same time, overcoming the inconvenience of scrolling through a large dataset has also become a considerably important challenge.

Consequently, numerous studies have been conducted to solve this problem in a desktop environment. Several studies [4,5] have suggested substantially effective scrolling techniques or widgets, whereas others [6–8] have conducted research on input facilities (i.e., track wheels, joysticks, etc.) to improve scrolling. However, these solutions are not appropriate for scrolling on mobile devices. Because of limitations in size and weight, which are intended for portability, input facilities on a mobile device are limited [3,9]. Practically, all mobile devices, particularly smartphones, are designed as touch-based systems, which differ in the manner in which they interact with users compared with desktop environments [10]. Some studies have attempted to solve this problem through additional GUI widgets such as a search function [11]. However, in certain cases, implementing an additional GUI on a mobile device for scrolling is inadequate because the screen is comparatively small [3,10,12].

We attempted to solve this problem by developing an enhanced kinetic scrolling technique based on users' behaviors and scrolling environments. Kinetic scrolling is a popular technique whereby a scroll widget can automatically continue scrolling for a short time after the finger is lifted from the screen. This technique helps users navigate information more comfortably. At present, Google's Android and Apple's iOS provide developers with a scrolling GUI widget that supports kinetic scrolling in the native development environment. More recently, numerous applications on mobile devices are using kinetic scrolling to automatically support viewport sliding. Moreover, an effective scrolling technique in the mobile device is an interesting topic for many specialized environments. For example, in the automotive industry, interacting with the large amounts of data on a mobile device while driving is an important challenge, wherein scrolling is considered as one of the typical interactions [13,14]. In this study, we present adaptive kinetic scrolling (AKS), a scrolling technique that permits rapid access to target information in an ordered large dataset based on the user's scrolling actions and the amount of data to be scrolled. We alleviate the user's trouble by amplifying the automatic scrolling feature of kinetic scrolling. We boosted the speed of automatic scrolling when users scroll to access target information that is located far away. Based on a user's scrolling actions, AKS determines their intent. Moreover, it amplifies the speed of kinetic scrolling on each occasion according to the dataset and current scroll position. The more data the scrolling widget contains, the more rapidly it scrolls so that the user can quickly reach the target. With available kinetic scrolling methods, the automatic scrolling speed is determined entirely through a touch-based gesture. Our main contribution is to permit users to access the target data more swiftly when scrolling on a mobile device.

The remainder of this paper is structured as follows: we briefly review related research on the topic. Subsequently, we present the proposed technique in detail. We then describe the experiment conducted to verify the validity of our proposed method. On the basis of this, the results and applicability of our experiments are discussed. The paper concludes by summarizing the contributions of this study and future work on our proposed method.

2. Related Works

Scrolling is commonly used in a variety of environments including mobile devices and desktop web browsers. Various studies [4–7,15–18] have been conducted on the characteristics of scrolling. Among these studies, several methods for improving scrolling have been investigated. In this section, related works are discussed. First, reported scrolling techniques are considered. Several of these studies are designed with an approach similar to what we have employed. Subsequently, related research and techniques that apply to kinetic scrolling are covered, considering that our research is based on this type of scrolling.

2.1. Techniques in Scrolling

A technique [11] was previously proposed to support searching using the scroll bars in a document. In this study, the author proposed multiple scroll bars that indicate the location of the result of searching for text. The author expands the functionality of scrollbars to help users explore search results without changing the presentation of the document. However, the study was designed for

a desktop environment, and mobile devices do not explicitly display a fixed scroll bar. Moreover, displaying additional GUIs may be inappropriate for mobile devices owing to the limited screen size.

Ishak and Feiner presented a content-aware type of scrolling [4], which varies the direction and speed of scrolling according to the content of the document in order to improve user convenience. When users read multi-column documents or scroll through text search results, the direction of the scroll varies according to their manipulation technique. Moreover, depending on the importance of the content, the scrolling speed is adjusted to be either slow or fast. Such a method of adjusting the scrolling speed according to the content is similar to our research approach. However, the aforementioned technique solely adjusts scrolling speed for special tasks, such as moving to the next column in a multi-column document or scrolling through the results after a text search. On the other hand, our proposed technique adjusts the kinetic scrolling speed according to the amount of content when the user intends to access remote data.

Cockburn et al. proposed an improved scrolling technique for traversing a long document [19]; their approach is similar to the current approach. The study presented a gain function that changes the scrolling speed according to the length of the document to be traversed. Our approach is similar in that the speed of the scrolling is changed dynamically according to the environment. The difference in our study is that the proposed AKS is designed for a mobile environment and amplifying scrolling speed is applied only for kinetic movement. The study in [16] is based on the scroll wheel of a mouse or scrolling devices, such as trackpads, but is not suitable for a touch-based interface. In a touch-based scrolling interface, the movement of the viewport accurately reflects the traversed distance of the finger. When the finger is dragged but the viewport moves further than the distance moved by the finger through the gain function, the user may be perplexed. For this reason, the application of this approach in mobile devices can be problematic.

2.2. Kinetic Scrolling Techniques

Juho et al. conducted research to adjust kinetic scroll according to content when navigating a web page [15]. Accordingly, the friction in the kinetic scroll changes according to the user's degree of interest in the content. For example, when an item with a high degree of interest is displayed in the viewport, high friction is applied to the scrolling, which requires the user to focus. Although the goal of this research differs from the goal of our research, the approach of manipulating the kinetic scroll speed is similar to that of our research. One of the major differences between our research and the study in [15], regarding technical details, is that the application of their technique requires additional preprocessed data.

Baglioni et al. presented an automatic scrolling technique, which is sustained according to a flick-and-brake gesture [16]. This approach permits users to regulate automatic scrolling based on the pressure of their finger, allowing the users to navigate information with fewer actions. Our research focuses on the users' rapid access to specific information in large datasets and differs in enabling users to focus on navigating information using fewer actions.

3. Adaptive Kinetic Scrolling

In this section, we will discuss the motivation and technical details of our approach. Below, we first describe the motivation of our technique. Then, we explain when AKS is activated. Subsequently, we describe how AKS determines the amplification of the kinetic scrolling speed.

3.1. Motivation

The problem that we attempt to solve is to reduce the time and energy expended by users in repeated scrolling for accessing certain data that is far away. For example, consider the case of finding the phone number of a person whose name starts with Z in a contact application that orders contacts alphabetically. In this case, the user repeats scrolling several times to shift the viewport to a remote location regardless of the visual information that emerges from the scroll [20]. We designed AKS based

on such behaviors of users. AKS amplifies the scrolling speed when users access specific data located at a distance. By amplifying the scrolling speed, the time spent in meaningless scrolling is reduced and users will be able to conveniently access distant information marginally more rapidly with this approach. Accordingly, we attempted to solve the problem by amplifying the kinetic scrolling speed, which is automatically performed after the user releases the finger.

Moreover, in our technique, the degree of amplification of the kinetic scrolling speed depends on the instantaneous scrolling environment. The purpose of our proposed technique is to make it possible for users to accomplish tasks faster. The required task time depends on the size of the dataset being scrolled. If there were a large amount of data in the scroll, it would be more effective to make scrolling more rapid. Conversely, to scroll through a small dataset, the scrolling speed need not be over-amplified. Therefore, our technique changes the degree of amplifying adaptively depending on the amount of remaining data to be navigated through.

The above is a type of a closed-adaptive system, in which the system responds to changes in the operating environment [21]. The adaptive user interface is among the worthwhile approaches that can improve usability [22,23] in a mobile environment [24]. It is one of the important aspects of usability that facilitates task performance [22,23]. Moreover, scrolling is considered as one of the most important interactions in perspective of usability of mobile applications [25,26]. With AKS, improvements in usability are expected by making it possible for users to accomplish scrolling tasks faster.

We have designed a mechanism to amplify the scrolling speed based on kinetic scrolling. Moreover, we propose a method for detecting attempts by a user to access target data that is far away from the current location, based on behavior. This technical detail is discussed in the next part of this section.

3.2. Adaptive Kinetic Scrolling Trigger

As mentioned above, our proposed scrolling technique amplifies the speed of kinetic scrolling to rapidly access specific data that is far away from the current position of the viewport. Therefore, AKS is only triggered in specific situations. Because AKS is a technique that amplifies the scrolling speed, the possibility of an overshoot while scrolling may increase. To prevent this type of side effect, a method to detect the user's intention to shift to target data distant from the current viewport is proposed. We defined a trigger condition to activate AKS to allow for this process.

In this study, we estimate the target distance based on the actions of the user. According to available research [20], users apply an "open-loop" action that does not depend on visual feedback to acquire a target rapidly when the target distance from the current position is considerable. In other words, users will scroll very rapidly without relying on sensory information when the distance from the target is sufficiently large. We use these user behaviors to define the condition in which the scrolling speed is amplified. In addition, we use the clutch time to read the intention of the users. Clutching is an action in which the users lift their finger from the touch screen and reposition it in the air. This clutch time indicates the time interval between scrolling actions. With a touch-based device, the clutch time is a reliable indicator of the intended scrolling velocity, with users increasing their gesture frequency to express a desire for faster scrolling [27]. In other words, a short clutch time implies that the user is repeatedly scrolling rapidly.

In summary, if a user wishes to scroll rapidly, the clutch time will be short; we will use this indicator to determine the case in which the user assumes that the target data is far away from the present position. Figure 1 shows the AKS as a state machine with a trigger condition.

Figure 1. State machine for adaptive kinetic scrolling (AKS). The top part (A) (gray-boxed) represents the available kinetic scroll as a state machine and the bottom part (B) (blue-boxed) represents a state machine added newly for AKS.

In the “idle” state, where there is no input, when the user touches the screen for scrolling, it enters the “ready” state. Subsequently, the screen shifts according to the drag, and it is in the “scroll” state. When the user releases the finger from the screen after the scrolling is completed, auto scrolling is performed owing to the kinetic effect. Then, the speed decreases over time, and the screen enters the “idle” state again.

AKS overrides the available “autoscroll” state under certain conditions. Our approach is designed to rapidly access the target information, assuming that the user knows the approximate location of the information. We designed AKS to activate when the clutch time observed is below the threshold (constant C in Figure 1), that is, when the user attempts to scroll very rapidly (see (B) in Figure 1). This constant C is determined empirically. The user’s cognition and action are affected by various external factors such as the size of the device’s viewport, GUI, and contents. Therefore, the threshold must be empirically determined for each environment. In Section 4.3, we discuss the procedure for determining the threshold value in more detail.

In addition, such user characteristics are relevant only if the user is familiar with the data or if the data is organized in a standard format (such as in form of alphabetical or numerical ordering) [20]. Our study covers only this type of environment. If the user is unfamiliar with the scrolling data, the observed target-acquisition time becomes linear with the target distance because all actions are performed based on sensory feedback. In this case, it is not feasible to apply rapid scrolling.

3.3. Adaptive Kinetic Scrolling Speed

Kinetic scrolling is a technique developed using inertial effects and friction, and the scrolling speed is dependent on these two physical concepts. Inertia is caused by the flicking or sliding gestures input by the user. Flicking [18,28] is a throw-like gesture, which is converted into an inertial effect wherein friction is constantly applied to the scroll. However, with the proposed AKS technique, the inertial effect is determined by both the input gesture of the user and the situation. Figure 2 shows factors other than the user input that affect AKS.

Figure 2. Remaining data determining scrolling speed. In available kinetic scrolling methods, the speed of automatic scrolling is determined only by the input gesture regardless of the situation. In AKS, we further adjust the speed of automatic scrolling by taking into consideration the remaining data.

According to the proposed solution, if the amount of remaining data is large, the speed is higher than that of standard kinetic scrolling; similarly, if the amount of remaining data is small, the speed is adjusted marginally. Using this mechanism, the initial speed of kinetic scrolling is determined dynamically on each occasion.

Kinetic scrolling speed exhibits an exponential regression curve based on the frictional effect. Scrolling velocity decelerated by friction is implemented in the form of an exponential decay factor [27]. The variable denotes the elapsed time, and the constants depend on the mobile device and Operating System. Figure 3 presents a comparison of the kinetic scrolling speed with and without AKS.

Figure 3. Kinetic scrolling speed for different data sizes, compared with standard kinetic scrolling.

In our model, the adjusted initial velocity (v) is calculated as:

$$v = v_0 \times (\log_A(\frac{RD}{B} + C)) \quad (1)$$

The variable RD represents the amount of the remaining data, and the constants A , B , and C can be optimized according to environmental factors such as the screen size of the device and type of information. Because user interfaces should be designed with consideration to the task and size of the touch screen and not applied universally across all devices [29].

Because the speed of kinetic scrolling is represented by an exponential decay, multiplying it by a monotonically non-decreasing positive function can produce an amplifying effect. Excessive speed amplification can adversely affect user experience; therefore, we have proposed a logarithm-based amplification function.

4. Experiment

We conducted a user study to examine the effect of our proposed scrolling technique on the scrolling task. There are three research questions to our experiment:

- **RQ. 1** How effective is AKS in reducing task time compared to standard kinetic scrolling?
- **RQ. 2** How effective is AKS in reducing the number of scrolling actions compared to standard kinetic scrolling?
- **RQ. 3** Does AKS affect the occurrence of overshoots?

The first and second questions investigate the effectiveness of our technique, whereas the third question verifies the side effects. We have developed a test application for these research questions. The application automatically records the task time, number of scrolling actions, and number of overshoots.

4.1. Participants

We recruited eight participants (males) with a mean age of 26.6 (standard deviation = 3.04). All the participants reported that they had used a smartphone for over three years. All the participants were university or graduate students.

4.2. User Study Application

Each test participant was required to use the scrolling method on a mobile device to find a specific name on a list. Figure 4 shows a screen capture of the experimental application. This application was implemented to provide a list of names when scrolling. Each name on the list was generated randomly to prevent a learning effect, and the target name that the users were requested to search for was displayed at the top of the screen (shown as A in Figure 4). The items in the list ((B) in Figure 4) were organized in alphabetical order, and the users were requested to click on the target name to complete the task. This GUI was designed with reference to the contact application of the mobile phone and rendered in a straightforward manner for the users' instinctive use. Moreover, as it is not a familiar dataset that users generally use, we placed the target data explicitly at the top (A) (see Figure 4) so as to enable remembering it during the experiment. For this experiment, we conducted a pilot study to determine the triggering threshold (clutch time). The pilot study is described in Section 4.5.

Figure 4. Screen capture of experimental application. In the experiment, the items on the list were written in Korean; however, in this figure, they have been replaced by sample data written in English for better understanding. (A) is a target data that the users were requested to search. (B) is a scroll list filled with names which alphabetical ordered.

Before conducting the test, the users completed two sessions using a standard scroll as a training task. We conducted training tasks without recording the results to permit the users to understand the process of the specified tasks. During the experiment, the test application used a script to record the task completion time, the number of scrolling actions required to complete the task, and the number of overshoots. And at the end of each task, the test moderator checks the collected data.

In this experiment, we applied the following values to the constants of Equation (1) presented in previous section.

$$v = v_0 \times (\log_{10}(\frac{RD}{20} + 10)) \quad (2)$$

In this configuration, when the remaining data (RD) is 1000 items, the speed is amplified by approximately 1.8, and when the target distance is 500 items, the speed is amplified by approximately 1.5.

4.3. User Study Design

We attempted to evaluate the effectiveness of the AKS using the A/B test. We measured the task time, number of scrolling actions, and overshoot when using AKS/Standard Scroll and compared them. Four participants used the standard scroll, and four used AKS.

Table 1 shows the settings for the user study sessions. The list is filled with 1000 items, and the location of the specific data (target distance) to be searched during each session is progressively increased to 200, 400, 600, and 800 items away from the start position of the viewport. For example, in the first session of the test, a list with 1000 names is created, and the target name is the 200th item. The test participants performed the tasks in a standard kinetic scrolling environment and under conditions similar to that of AKS. Each participant performed the tasks four times, which amounts to a total of 16 scrolling tasks for each conditions and distance. The task conducted each target distance four times in sequence (i.e., distance 200×4 times, distance 400×4 times, and so on). The task order was the same for the scrolling type.

Table 1. User study session for participants.

Scrolling Type	Target Distance	Repetition	Total
Standard	200, 400, 600, 800	4 times for each distance	16 times testing for each condition
AKS	200, 400, 600, 800	4 times for each distance	16 times testing for each condition

4.4. Study Environment

We implemented scrolling by applying a JavaScript-based open-source kinetic scrolling library [30]. During all the experimental sessions, the users used an Apple iPhone 6 in portrait mode. This device has a 4.7-inch display with a resolution of 1334×750 . The study was performed in regulated laboratory environment. Moreover, when an external problem occurred in a test session (i.e., wireless network delay, device freezing, etc.), we discarded the invalid data and requested a redoing of the test session.

4.5. Determining Threshold for Trigger

We conducted the following study to determine the threshold value of the clutch time for triggering the AKS: The participants completed a task to find a specific string data in an alphabetically ordered list on the mobile device, and we measured the target distance and clutch time. The study environment was similar to that of the previous experiment. In addition, all the users participating in the pilot study completed two training tasks as well as the earlier experiment. In this study, five participants in the experiment repeated a given task three times each. Figure 5 shows the measured results with a scatter plot. Note that none of the participants overlapped with the users in the pilot study.

Figure 5. Relationship between target distance and clutch time.

When the target distance is between 200 and 800, the clutch time measured is approximately 100–200 ms. This is because when the user assumes the current position of the scroll to be far enough away from the target, they scroll rapidly and repeatedly to omit the unnecessary parts. Conversely, if the target distance is near and below a certain distance, the clutch time will increase significantly. The reason for this is that people are aware that they are close to the target and perform the action by examining the change by scrolling. The results from our study were similar to those in [27].

Based on these measurements, we have determined the threshold for triggering AKS. To determine a threshold, we set up the following two conditions:

1. Is AKS not triggered when the target is near?
2. Is AKS triggered when the target is far?

Condition 1 is intended to prevent side effects, and it implies a type of false-positive case. Condition 2 implies precision. These two conditions are in conflict. For example, if you set the threshold excessively low, it can cause side effects if AKS is triggered too easily and the target is in the vicinity. However, if the threshold is unnecessarily large, AKS will not function, and the effect will be lost.

In our study, we set the threshold value to 250 ms. We achieved this using the clutch time values exhibited by the users at a certain distance. In our study, users recognized in most cases that the target is close when the target distance was less than 100. Simultaneously, the clutch time increased rapidly (see Figure 5). With this value, over 95% of scrolling satisfied condition 1 and over 90% of scrolling satisfied condition 2.

We observed a notable point related to condition 2. Several users scrolled slowly regardless of the target distance. This is shown in Figure 5; it is evident at the points with clutch times of 300 ms or more regardless of distance. Therefore, it was not feasible to set a threshold that completely satisfied condition 2, and for our experiment, we set the threshold value marginally conservatively.

The values may have varied marginally depending on each user's preference and language, the GUI of the list, and the device used. For example, the speed of recognition may vary depending on whether the information in the scrolling list is numeric, text, or a certain language. GUI configuration and text size can also affect users. Therefore, this value should be measured for each system to be applied.

4.6. Implementation

We implemented AKS for the user study and in-depth analysis. In this section, we will present and explain the pseudocode based on the callback GUI function frequently used in mobile web or mobile application frameworks. The proposed code is shown below (see Box 1). Comparing with the statement machine of AKS, which is presented in Figure 1, will aid in clearer understanding. This code snippet was written by referring to the structure of the source implemented in [30].

Box 1. Pseudocode Script for AKS.

```
var offset //variable for measuring scrolling distance
var velocity //variable for velocity of scrolling
var timestamp //variable for timestamp to calculate scrolling velocity
var remainingData //variable for size of remaining data
var clutchtimeThreshold = C
//Constant for triggering the AKS

function tap() {
  //tap() is a callback function that is called when the screen is touched.

  var velocity = 0
  timestamp = Date.now();
    //Date.now() means the 'get current timestamp'
    offset = currentPos()
}

function release(){
  //release() is a callback function that is called when the finger is released.

  var delta = offset - currentPos()
  var elapsed = timestamp - Date.now()
  velocity = delta/elapsed
  remainingData = screen.Size() - currentPos()

  if(velocity > clutchtimeThreshold){
    AKS(delta, elapsed, remainingData)
  }else{
    kineticScroll(delta, elapsed)
  }
}
```

The *tap* function is called when the user taps a finger on the screen to start scrolling. At this time, the timestamp and current position at which scrolling starts are recorded in the *timestamp* and *offset* variables, respectively. Thereafter, the *release* function is called when the user releases their finger, which implies stoppage of scrolling. In the release function, the velocity is calculated based on the movement distance (*delta*) and the consumption time (*elapsed*) of the scroll just performed. At this time, AKS is activated if the scrolling speed is above the *clutchtimeThreshold* (constant C in Figure 3). In AKS, the speed of autoscrolling is amplified as shown in (Figure 1). Moreover, the parameter that determines the amplification is *remainingData*, which implies the amount of item in the scroll to be navigated. In our implementation, the remaining data in the scroll was measured in the following manner. First, we calculate the relative position of the current viewport to the total scrollable page length. Thereafter, the number of items displayed in the viewport was multiplied by the ratio. In this way, we can measure the remaining scrollable items. If you use the APIs provided by Android applications, you can measure the number of items left without additional calculations and calculate them more easily. AKS is a widget that operates using only the user's behavior and GUI environment without additional data preprocessing. It can be conveniently applied to the part corresponding to the view of the Model-View-Controller (MVC) pattern with only a simple implementation.

5. Results and Discussion

5.1. Results

We obtained the task time, number of scrolling actions, and number of overshoots from the user study. We calculated the average for each condition at the end of all test sessions and compared the results of AKS with those of standard scrolling to answer the research questions.

RQ. 1 How effective is AKS in reducing task time compared to standard kinetic scrolling?

First, we verified that using AKS aids users in accessing required information more rapidly (see Figure 6). When using AKS, the average task time was reduced by approximately 14–25%. We observed the effect of AKS to be significant, particularly when the target distance is long. In the case of a task with a target distance of 800 items, the average task time of AKS was reduced by approximately 25% compared to the average task time of standard scrolling. It was also verified that the task time was reduced by approximately 23% when the target distance was 600 items. As a result, AKS appears to be useful for reducing the time required for repetitive scrolling actions to access specific data.

Figure 6. Time required for scrolling tasks.

The results in Figure 6 show that AKS can help users to access target information faster. However, given that the scale of our experiments is limited, additional statistical analysis is needed. To verify these results, we conducted a *t*-test to confirm that there is a significant difference between the task time of the AKS and the standard scroll. First, we conducted the Shapiro-Wilk test (significance level = 0.05) to check the normality of the collected data group for each target distance. Through this test, we have confirmed that each dataset is normally distributed. Subsequently, we performed a *t*-test (significance level = 0.05) using the same datasets. Table 2 shows the standard deviation and *p*-value for each target distance. Based on the calculated *p*-value, the difference in task time between the two scrolling types is statistically significant.

Table 2. Standard deviation (SD) and *p*-value of task time.

Target Distance	Standard Deviation		<i>p</i> -Value
	Standard Scroll	AKS	
200	1.324	0.836	<0.001
400	1.184	1.074	<0.001
600	1.896	1.472	<0.001
800	1.571	2.119	<0.001

RQ. 2 How effective is AKS in reducing the number of scrolling actions compared to standard kinetic scrolling? In addition to the task time, we expected that the scrolling action performed by the user would be reduced when AKS was used. This is because scrolling becomes more rapid through AKS; therefore, the number of actions required to reach the target data can be reduced. The number of actions were reduced in general. When using AKS, the user performed approximately 2–10% fewer actions (see Figure 7). Similar to the task time, the larger the target distance, the more effective AKS appeared to be. When the data to be accessed is not far away, significant results could not be observed when applying AKS. For example, if the target distance is 200 or 400, only minor improvements are exhibited. However, when the target distance was 600 or 800, the effect was significantly higher. This tendency was also verified in **RQ. 1**. Moreover, compared to the task time, the improvement in the number of actions appears relatively marginal.

Figure 7. Number of scrolling actions required for scrolling tasks.

Like in **RQ. 1**, a *t*-test was performed using the collected data to verify the results. We used the Shapiro–Wilk test to check the normality of the collected datasets (significance level = 0.05). Using this test, we confirmed that all datasets were normally distributed. Furthermore, a *t*-test (significance level = 0.05) was performed to verify the differences in number of actions for two scrolling types. Table 3 shows the standard deviation and *p*-value for each target distance.

When the target distance is 200 or 400, the *p*-value is approximately 0.4. This means that the difference between the two groups is not statistically significant. In other words, AKS cannot guarantee a reduction in the number of scrolling actions in this case. These results imply that AKS does not contribute to reducing the number of actions when the target distance is not sufficient. As mentioned earlier, there was no significant difference in the number of actions when the target distance was short, and the *t*-test confirmed that the difference is not valid. However, the *t*-test result also shows that AKS can help the user reach the target information with fewer scrolling actions when the target distance is 600 or 800.

Table 3. Standard deviation (SD) and *p*-value of number of scrolling actions.

Target Distance	Standard Deviation		<i>p</i> -Value
	Standard Scroll	AKS	
200	1.652	1.825	0.422
400	1.914	2.219	0.438
600	4.747	3.225	0.021
800	3.286	3.540	<0.001

RQ. 3 Does AKS affect the occurrence of overshoots? Finally, we analyzed whether the application of AKS by users causes higher overshoot or not. This question is to ensure that the increase in the scrolling speed by AKS occurs without side effects when the user performs tasks. We obtained positive results in terms of overshooting by AKS. Overshoots were observed seven times with standard scrolling and ten times with AKS (see Figure 8). For these test sessions, seven and ten overshoots are not significantly different. These results indicate that there is only a marginal difference in the likelihood of an overshoot when AKS is used. The scrolling patterns applied were marginally different for each user, and the effectiveness of the proposed method was dependent on such patterns. A few users scrolled carefully; therefore, in these cases, AKS was triggered fewer times. Vice versa, for the few users with very rapid and bold scrolling, AKS was used more frequently. In conclusion, in most cases, we observed that the proposed AKS method is effective. The effects of AKS have been verified on the basis of experimental results, and a few of the considerations required while applying AKS are discussed in the next section.

Figure 8. Number of overshoots that occurred during scrolling tasks.

5.2. Discussion

We verified the usefulness of AKS through the results of the user study. AKS can aid users in accessing a faraway target through faster scrolling and in a more efficient manner. In this section, we will discuss a few points to be considered while applying our proposed system. We describe the case of applying AKS in the perspective of the characteristics of the system, the types of data used, the user behavior, and the costs to be incurred.

First, as mentioned in Section 3, AKS is effective when a user is familiar with the data or if the data is organized in a standard format. In other words, the user must estimate the location of the target data. It is not appropriate to apply AKS if the user is unfamiliar with the item being scrolled or cannot predict the location of the specific item. For example, AKS is not appropriate for a news application that changes the position of an item over time. In such systems, it is not necessary to consider a use case in which a user rapidly accesses specific information. Before applying AKS, it is necessary to first understand the characteristics of such a system.

AKS is a data-independent system that does not require additional preprocessing. Moreover, as we have observed in the result of RQ. 3, side effects will not occur with our method. We have identified the case where the target data is far away, based on users' scrolling behaviors, in order to prevent side effects. Based on this design, AKS is triggered only when the user intends to rapidly access specific information. Therefore, the application of AKS is effective regardless of the type of data.

AKS yielded particularly meaningful results when users wished to access data farther away, as was the tendency observed in the experimental results. This is because AKS functions when the user scrolls rapidly and repeatedly without visual feedback to access faraway data. These results indicate

that AKS will be most effective if the dataset that the user needs to navigate through is large and if the user frequently accesses faraway data.

Applying AKS is not expensive from the implementation perspective. However, one aspect to consider is that as mentioned earlier, the clutch time value used to trigger the action is affected by the environment, and this value can be obtained through a heuristic process. Although this is not a complicated task, it is a labor-intensive process that developers and testers are required to perform manually. It would be effective to compare this cost with the positive outcomes of applying AKS when enhancing the system.

6. Conclusions

We introduced AKS, a novel scrolling technique that adaptively amplifies the speed of kinetic scrolling depending on the size of the dataset to be navigated. AKS aids users in accessing target data more rapidly when scrolling in a mobile application. Users generally consume a large amount of time and energy owing to repetitive and meaningless scrolling, and we contributed to improve this situation. There have been studies based on these problems; however, most of them have been conducted in desktop environments. As mobile environments are smaller and based on a touch interface, it is infeasible to apply those studies to mobile applications.

AKS is designed to be triggered based on user behavior. We used the clutch time as a trigger to determine the situation in which the user wishes to access specific information that is located far away from the current position, in order to prevent side effects. Further, our method determines the degree of increase in speed adaptively according to each situation.

We conducted a user study in which we compared the proposed AKS method with standard kinetic scrolling and verified the effectiveness of our technique. Based on the results of the experiment, the proposed technique is likely to be useful for rapidly accessing specific data. If users are required to scroll through a large amount of data on a mobile device, where it is challenging to support additional GUIs or functions, AKS will aid users.

In future work, we plan to study kinetic scrolling techniques based on factors additional to the amount of data, such as viewport size, type of information, and size of text. We will also conduct a large user study with participants across all age groups and genders.

Author Contributions: J.J. has performed the works of conceptualization, methodology, performed the experiments and writing—original draft preparation. N.K. supervised the experiments design and performed writing—review. H.P.I. supervised the research and reviewed the paper.

Funding: This research was supported by the MIST (Ministry of Science and ICT), Korea, under the National Program for Excellence in SW (2015-0-00936) supervised by the IITP (Institute for Information & communications Technology Promotion) and supported by National IT Industry Promotion Agency grant funded by the Ministry of Science and ICT and Ministry of Health and Welfare (NO. C1202-18-1001, Development Project of The Precision Medicine Hospital Information System (P-HIS)).

Acknowledgments: The authors would like thank to Nuri Park for supporting the experiment.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ma, Q.; Cui, T.; Zheng, J.; Zhang, S.; Zhou, W. Mobile phone changing prediction based on large-scale user behavioral data. In Proceedings of the 19th International Symposium on Wireless Personal Multimedia Communications (WPMC), Shenzhen, China, 13–16 November 2016; pp. 91–96.
2. Hinckley, K.; Jacob, R.J.K.; Ware, C. Input/output devices and interaction techniques. In *Computer Science Handbook*, 2nd ed.; Chapman and Hall/CRC: Boca Raton, FL, USA, 2004; pp. 20:1–20:32.
3. Huang, K.Y. Challenges in human-computer interaction design for mobile devices. In Proceedings of the World Congress on Engineering and Computer Science, San Francisco, CA, USA, 20–22 October 2009; pp. 236–241.

4. Ishak, E.; Feiner, S. Content-aware scrolling. In Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology (UIST 2006), Montreux, Switzerland, 15–18 October 2006; pp. 155–158.
5. Smith, G.M.; Schraefel, M.C. The radial scroll tool: scrolling support for stylus- or touch-based document navigation. In Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (UIST 2004), Santa Fe, NM, USA, 24–27 October 2004; pp. 53–56.
6. Zhai, S.; Smith, B.A.; Selker, T. Improving browsing performance: A study of four input devices for scrolling and pointing tasks. In Proceedings of the INTERACT'97, Sydney, Australia, 14–18 July 1997; pp. 286–293.
7. Hinckley, K.; Cutrell, E.; Bathiche, S.; Muss, T. Quantitative analysis of scrolling techniques. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Minneapolis, MN, USA, 20–25 April 2002; pp. 65–72.
8. Zhang, X.; Liu, X.; Yuan, S.M.; Lin, S.F. Eye Tracking Based Control System for Natural Human-Computer Interaction. *Comput. Intell. Neurosci.* **2017**, *2017*, 1–9. [[CrossRef](#)] [[PubMed](#)]
9. Dunlop, M.; Brewster, S. The challenge of mobile devices for human computer interaction. *Pers. Comput.* **2002**, *6*, 235–236. [[CrossRef](#)]
10. Nielsen, J.; Budiu, R. *Mobile Usability*, 2nd ed.; New Riders: San Francisco, CA, USA, 2012.
11. Björk, S. The ScrollSearcher Technique: Using Scrollbars to Explore Search Results. *INTERACT* **2001**, *1*, 745–746.
12. Wasserman, A.I. Software engineering issues for mobile application development. In Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, Santa Fe, NM, USA, 7–11 November 2010; pp. 397–400.
13. Louveton, N.; McCall, R.; Koenig, V.; Avanesov, T.; Engel, T. Driving while using a smartphone-based mobility application: Evaluating the impact of three multi-choice user interfaces on visual-manual distraction. *Appl. Ergon.* **2016**, *54*, 196–204. [[CrossRef](#)] [[PubMed](#)]
14. Ng, A.; Brewster, S. An Evaluation of Touch and Pressure-Based Scrolling and Haptic Feedback for In-Car Touchscreens. In Proceedings of the 9th ACM International Conference on Automotive User Interfaces and Interactive Vehicular Applications (AutomotiveUI 2017), Oldenburg, Germany, 24–27 September 2016; pp. 11–20.
15. Juho, K.; Amy, Z.X.; Jihee, K.; Robert, M.C.; Krzysztof, G.Z. Content-aware kinetic scrolling for supporting web page navigation. In Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST 2014), Honolulu, HI, USA, 5–8 October 2014; pp. 123–127.
16. Baglioni, M.; Sylvain, M.; Eric, L.; Yves, G. Flick-and-brake: finger control over inertial/sustained scroll motion. In Proceedings of the CHI 2011 Extended Abstracts on Human Factors in Computing Systems, Vancouver, BC, Canada, 7–12 May 2011; pp. 2281–2286.
17. Eunyhe, Y.; Geehyuk, L. Slack-scroll: sharing sliding operations among scrolling and other GUI functions. In Proceedings of the 27th Annual ACM Symposium on User interface Software and Technology (UIST 2014), Honolulu, HI, USA, 5–8 October 2014; pp. 105–106.
18. Tu, H.; Wang, F.; Tian, F.; Ren, X. A comparison of flick and ring document scrolling in touch-based mobile phones. In Proceedings of the 10th Asia Pacific conference on computer human interaction, Matsue, Japan, 28–31 August 2012; pp. 29–34.
19. Cockburn, A.; Philip, Q.; Carl, G.; Stephen, F. Improving scrolling devices with document length dependent gain. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Austin, TX, USA, 5–10 May 2012; pp. 267–276.
20. Cockburn, A.; Carl, G. A predictive model of human performance with scrolling and hierarchical lists. *Hum. Comput. Interact.* **2009**, *24*, 273–314. [[CrossRef](#)]
21. Oreizy, P.; Gorlick, M.M.; Taylor, R.N. An architecture-based approach to self-adaptive software. *IEEE Intell. Syst. Appl.* **1999**, *14*, 54–62. [[CrossRef](#)]
22. Usability 101: Introduction to Usability. Available online: <https://www.nngroup.com/articles/usability-101-introduction-to-usability> (accessed on 20 October 2017).
23. ISO. *Ergonomic Requirements for Office Work with Visual Display Terminals (VDT)*, ISO 9241-11: Part 11: Guidance on Usability; ISO ICS: 13.180; ISO: Geneva, Switzerland, 1998; p. 22.
24. Wesson, J.L.; Singh, A.; Tonder, B.V. Can adaptive interfaces improve the usability of mobile applications? *Hum. Comput. Interact.* **2010**, 187–198.

25. Chintapalli, V.V.S.M.; Tao, W.; Meng, Z.; Zhang, K.; Kong, J.; Ge, Y. A comparative study of spreadsheet applications on mobile devices. *Mob. Inf. Syst.* **2016**, 1–10. [[CrossRef](#)]
26. Punchoojit, L.; Hongwarittorn, N. Usability Studies on Mobile User Interface Design Patterns: A Systematic Literature Review. *Adv. Hum. Comput. Interact.* **2017**, 1–22. [[CrossRef](#)]
27. Quinn, P.; Sylvain, M.; Andy, C. Touch scrolling transfer functions. In Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST 2013), St. Andrews, UK, 8–11 October 2013; pp. 61–70.
28. Reetz, A.; Carl, G.; Tadeuze, S.; Miguel, N.; Sriram, S. Superflick: A natural and efficient technique for long-distance object placement on digital tables. In Proceedings of the Graphics Interface, Quebec, QC, Canada, 7–9 June 2006; pp. 163–170.
29. Jennings, A.; Ryser, S.; Drews, F. Touch screen devices and the effectiveness of user interface methods. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting, San Diego, CA, USA, 30 September–4 October 2013; pp. 1648–1652.
30. Github-Ariya/Kinetic: Kinetic Scrolling with JavaScript. Available online: <https://github.com/ariya/kinetic> (accessed on 20 October 2017).

© 2018 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).