Journal of Computational Design and Engineering Vol. 1, No. 3 (2014) 153~160
www.jcde.org

An alternative method for smartphone input using AR markers

Yuna Kang[1] and Soonhung Han[2,*]

[1] *1st R&D Institute-3, Agency for Defense Development, 488 Bugyuseong-daero, Yuseong-gu, Daejeon 305-152, Republic of Korea*
[2] *Department of Ocean System Engineering, Korean Advanced Institute for Science and Technology, 291 Daehak-ro, Yuseong-gu, Daejeon 305-701, Republic of Korea*

(Manuscript Received January 8, 2014; Revised March 17, 2014; Accepted March 18, 2014)

Abstract

As smartphones came into wide use recently, it has become increasingly popular not only among young people, but among middle-aged people as well. Most smartphones adopt capacitive full touch screen, so touch commands are made by fingers unlike the PDAs in the past that use touch pens. In this case, a significant portion of the smartphone's screen is blocked by the finger so it is impossible to see the screens around the finger touching the screen; this causes difficulties in making precise inputs. To solve this problem, this research proposes a method of using simple AR markers to improve the interface of smartphones. A marker is placed in front of the smartphone camera. Then, the camera image of the marker is analyzed to determine the position of the marker as the position of the mouse cursor. This method can enable click, double-click, drag-and-drop used in PCs as well as touch, slide, long-touch-input in smartphones. Through this research, smartphone inputs can be made more precise and simple, and show the possibility of the application of a new concept of smartphone interface.

Keywords: Smartphone; Augmented reality (AR); Marker; Interface; Human-computer interaction (HCI)

1. Introduction

1.1 Current states

In recent years, smartphones, a type of cell phone that perform more advanced functions like that of a PC, are widely available. Most of the smartphones have adopted the full-touch screen for convenient browsing, such as the PDAs in the past. Generally, touch screens can be categorized into 2 types. First type is the resistive type which recognizes pressure; another type is the capacitive type which detects changes in the electric current on the screen. PDAs adopted resistive type full-touch screen, but smartphones released recently use capacitive full-touch screen. The capacitive type can be operated and scrolled smoother than the resistive type [1]. However, the capacitive touch screens can only detect specific types of touch pens made for smartphones and cannot detect gloves, nails or general touch pens. In addition, because of decrease in mobility, people do not generally use touch pens for smartphones.

Also many problems have occurred in practice because the capacitive type is more sensitive than the resistive type. Representatively, when a user input characters with the soft keyboard (QWERTY), errors occur most frequently. As shown in Figure 1, the thumb covers several buttons when a user touches one button. The user cannot see where the thumb is located exactly, so many typing errors occur. Another common error occurs at mobile pages (see Figure 2). Currently, many web sites are providing mobile versions of web sites for smartphones. The old web pages for PCs are too large to be seen in a smartphone, so the user needs to zoom into the page to see a specific region of the page. The mobile pages are made to fit the screen size of smartphones, so the user can see

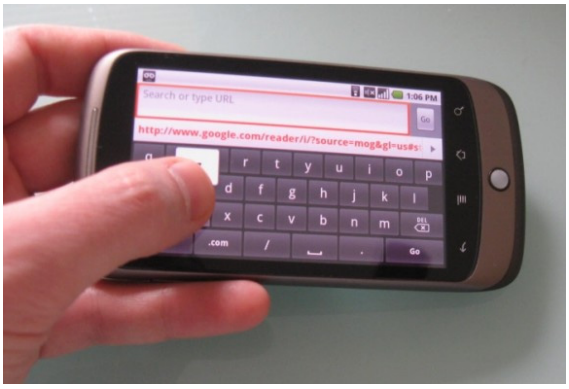


Figure 1. The user cannot see buttons covered by the thumb on QWERTY soft keyboard. [2, 3].

*Corresponding author. Tel.: +82-42-350-3040, Fax : +82-42-350-3210
E-mail address: shhan@kaist.ac.kr

doi: 10.7315/JCDE.2014.015

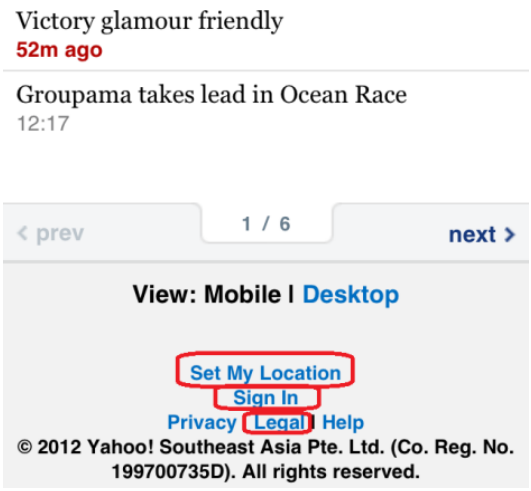


Figure 2. Example of input errors on mobile pages [3].

Figure 3. Example of using a touch pen for capacitive screen: It also covers the screen partially [3].

the pages without zooming in/out. However, some but tons in the page are located very close to each other and the mobile pages don't support commands for zooming in/out, so the user cannot press the desired button correctly.

There have been some software approaches to solve this problem. Currently, the operating system for smartphones provides an enlarged picture of the selected button when the user presses it. Also, in the game application, several calibration algorithms are applied to correct the input which is close to the exact input. However, these approaches are limited to input on soft-keyboard or specific games which have correction the algorithm. It is hard to select the desired input at once among many small buttons even for people who are familiar with smartphones. Moreover, if the user's fingers are dry or have hardened skin, or if the user is wearing gloves, the static electricity doesn't flow; in that case, even a simple input signal for smartphones does not occur. Touch pens for capacitive screen can solve these problems partially, but it isn't a permanent solution to the problem because the touch pen also partly covers the screen (see Figure 3).

In this study, to solve this problem, we propose a way to improve the smartphone interface using a simple marker which is often used when constructing augmented reality. First, previous studies similar to this study will be introduced and compared, and the proposed method will be described in detail. Finally, we will show some results of the implementation and some experiments which can determine the efficiency of this method.

2. Related works

Smartphones became popular only a few years ago, but the user interface of mobile devices using camera images has been developed since the past.

Some studies recognize the gestures of a user's hands [4-6] or a user's face [7, 8] from the camera images and these gestures were recognized and used as the input for PCs. There are many studies on gesture recognition in real-time for PC already. However, these methods require a long computation time in the image processing step, so it is difficult to apply to the smartphone environment. So in some researches for mobile devices, the system recognizes only shaking or tilting motions of a mobile device from camera images as the inputs [9-11].

In other researches, to solve the time-consuming problem of image processing, markers were used. It is fixed in real space and the user moves the camera to obtain the relative position of the marker for making input commands for a mobile device. Hansen et al. [12] located a marker on the floor, and moved the camera attached on the mobile device in the x, y, z direction.

On the other hand, Park and Moon [13] placed markers on fingers and the dummy which looks like a mobile device, and enabled the user to see the mobile device through a HMD. Their study is focused on the tangible interaction, but it is similar to our research in the point that input commands are made by camera images of the marker on the finger.

Hachet et al. [14] also suggests a navigation method for large-scale maps in mobile devices using a 3-colored target plate. Their study is similar to our research in the point which uses a fixed device and a moving target. However, the goal of Hachet's study is limited to the rotation and zoom-in/out of maps.

Byun and Kim's research [15], which has the most similar purpose to our research, proposed a method using the image processing technique for making input commands to the small keyboard of mobile devices. In their study, a virtual point is chosen from the camera image, and the 2D position of this point on real space is tracked when the camera is being moved around. As shown in Figure 4, the user can move the mobile device to match the chosen point to the desired location, and then click the switch button once to make an input event. Byun and Kim's study is most similar to our research in the approach that the acquired position from the camera is considered to be the position of the mouse pointer, but the most interesting characteristic is that the recognized

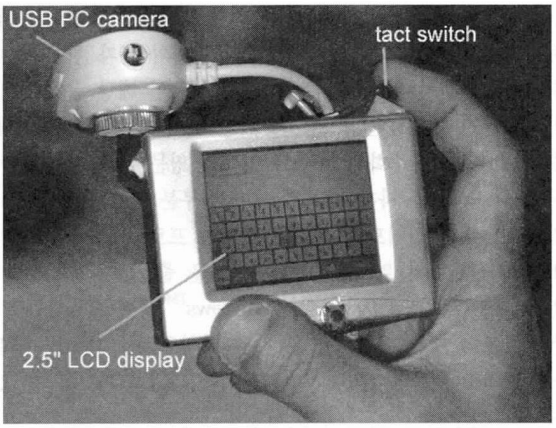


Figure 4. An interface for a mobile device [15].

point doesn't exist in the real world. Therefore, the absence of actual markers would be an advantage, but if the environment has dynamic objects, the stability cannot be guaranteed.

Table 1 shows the comparison with the related researches and this study. In Hachet's study [14] and Byun's study [15], the manipulation is very simple. Only a marker or a mobile device can be moved in 3D space for making input commands but Byun's study [15] requires a special device with a tact switch and the stability is low in dynamic environments. On the other hand, Hachet's study [14] has high stability for dynamic environments because it adopted a marker for the interface, but its target is the navigation on the map so the system supported only simple operations. Park's study [13] was focused to test digital mockup models, so the most detailed operations are available; however, the most complicated manipulation is required for the same reason.

There have been many studies which improve mobile interface using markers or the camera, but most of these studies have adopted the method of moving the mobile device near the fixed marker or the method of using a large pattern. However, if the mobile device is moved to enter a command, it's uncomfortable to see the display for the user. Also, because the smartphone should be available to use while carrying around, the method of using big patterns and fixed markers cannot be seen as a convenient method. To overcome this issue, this study adopts the method of using a small marker which can be attached to a finger or an object, and proposes the method which can make accurate inputs while keeping high mobility of mobile devices.

3. Interface using an AR marker

Augmented reality (AR) refers to a mixture environment of real environment and virtual objects [16].

There are many ways for accurate matching be-tween the real environment and the virtual environment in real-time. Among these, the method using an AR marker (marker-based tracking) is the most simple and robust way to get tracking results [17]. This study proposes the method where the position of an AR marker is used as the position of a mouse pointer on the PC.

Table 1. Comparison table for existing studies and this research.

	Hachet et al. [5]	Park et al. [14]	Byeon et al. [2]	This research
Simplicity of manipulation	High	Low	High	Medium
Stability of screen	High	High	Low	High
Stability for dynamic environment	High	Medium	Low	High
Possibility of detailed operation	Low	High	High	High

The input event on touch pad can be analyzed in two ways:

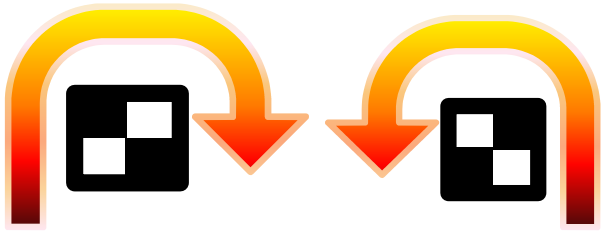
- The event which occurs when a user presses or releases a finger on the screen (OnPress / OnRelease)
- The event which occurs when a user's finger is moving on the screen (OnMove)

We Through these two inputs, most common functionalities of a mouse on PCs can be implemented. For example, a quick event of OnPress → OnRelease can be a "click" event, and the "click" event performed twice can be a "double-click" event. If OnMove event occurs between the OnPress event and the OnRelease event, it can be a "drag" event. Also, if the time interval between the OnPress event and the OnRelease event is long, it can be a "Long-Press" event which is often used in touch pads.

In this research, two methods will be introduced as the way to create the two events explained above on touch pads; first is the method of using the orientation of the AR marker only (position and rotation), and the other is the method of using the position of the marker and touch commands. The 2D positions of the marker from camera images can be considered as 2D positions of the mouse pointer in each method. Two methods were implemented individually.

3.1 Method A: Using a marker only

In the first method, the position and the orientation of the marker is only considered as a way of creating the input event in real-time. As in Figures 5(a) and (b), the system recognizes the symmetrical shape of the marker, and the OnPress/ OnRelease events occur depending on the change in the angle of the detected marker. Figure 5(a) can be recognized as an OnPress event, and Figure 5(b) can be recognized



(a) OnPress (b) OnRelease

Figure 5. The method creating OnPress and OnRelease event.

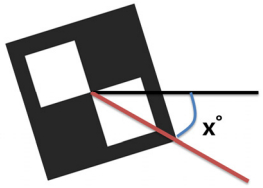


Figure 6. The rotation value of the marker [2].

as an OnRelease event. In practice, the marker doesn't need to rotate 90° fully; the recognition error can be increased if the precise angle is needed to make an event, so the system only detects the sign (positive/negative) of angles. If the rotation value x in Figure 6 is changed from positive to negative, OnPress event would occur, and if x is changed from negative to positive, OnRelease event would occur.

Figure 7 shows the overall flow chart of the method using the orientation of a marker. The system recognizes the position of the marker in the camera images. After determining the position and rotation of the marker, the system projects the 3D position of the marker onto the 2D screen of the smartphone to use as a mouse pointer. The angle of the marker can be checked in real-time to see whether the sign of the rotation value is changed; if the sign is changed, an OnPress or OnRelease event occurs depending on the change in the orientation and the color of the mouse pointer.

Figure 8 shows an example of the operation for click action with the rotation of a marker. The position of the marker located on the nail becomes the position of the mouse pointer

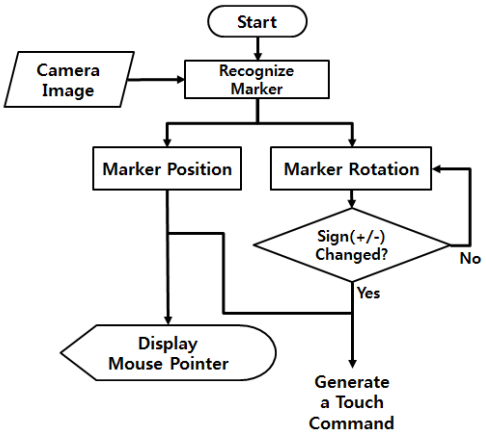


Figure 7. Flow chart of the method A.

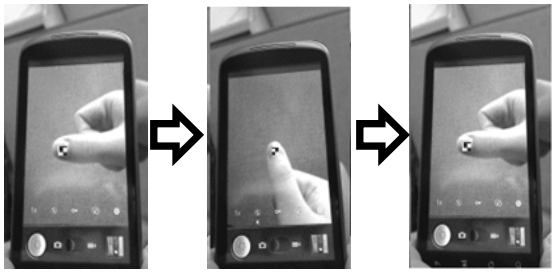


Figure 8. The click action of the method A.

Figure 9. Display of a mouse pointer from the marker.

(see Figure 9) and a series of actions of the marker rotating 90 degrees and returning to the origin, can be used like the click operation.

3.2 Method B: Using the position of a marker and touch commands

The method to recognize the rotation of a marker has no problem theoretically, but it needs a lot of training for the user to rotate the marker while keeping the exact location in practice. So we propose an alternate method which is the method of using both the position of a marker and touch commands on the screen to compensate for this problem. If the user places the marker on the desired 2D position, and then touches the screen, the input command occurs on the position of the marker, regardless of the location of the touch command.

The overall flow chart of the method using the position of a marker and touch commands is shown in Figure 10. The system recognizes the marker from the camera image and the position of it, but not the rotation of the marker. The system checks the touch input in real-time, and if there is an input signal, the system intercepts the signal and sends an input command to the current position of the mouse pointer (the 2D position of the marker). Figure 11 shows an example of the "click" command using both the position of a marker and touch commands. Same as the method of using the orientation of a marker only, the position of a marker becomes the position of the mouse pointer.

This method is similar to the method using conventional

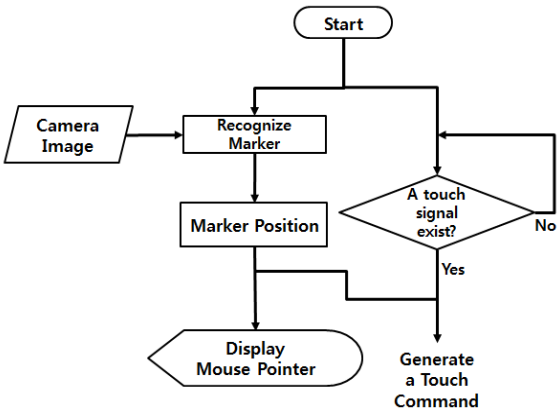
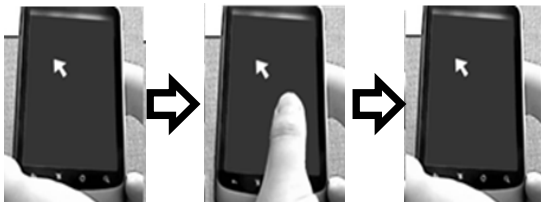


Figure 10. Flow chart of the method B.



(a) OnPress (b) OnRelease

Figure 11. The click action of the method B.

touch, but a more correct input can be made by this method since the finger does not cover up the actual input position. Also, this approach is more accurate and easy compared to the method using the orientation of a marker only, but for users in situations that are impossible to use capacitive touch screens, this method isn't available.

4. Implementation and experiments

4.1 Implementation

This system has been implemented on Google Nexus One, and its operating system is Android Gingerbread (Android 2.3.3). For implementing this system on the Android environment, several "views" need to be superimposed such as the camera preview, input buttons, and the mouse pointer. Especially, the camera preview and the mouse pointer should be updated in real-time repeatedly, so the delays can be caused on main thread by lack of resources. To alleviate these problems, SurfaceView in Android OS is used to run on the background resources for the real-time job.

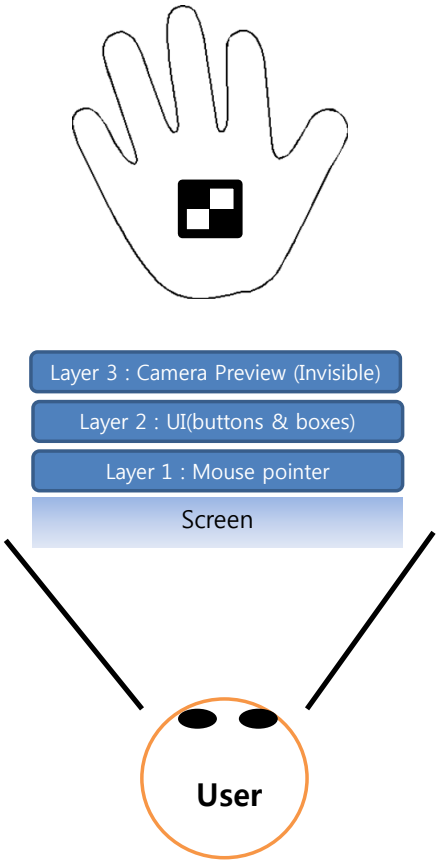


Figure 12. Structure of the implemented system.

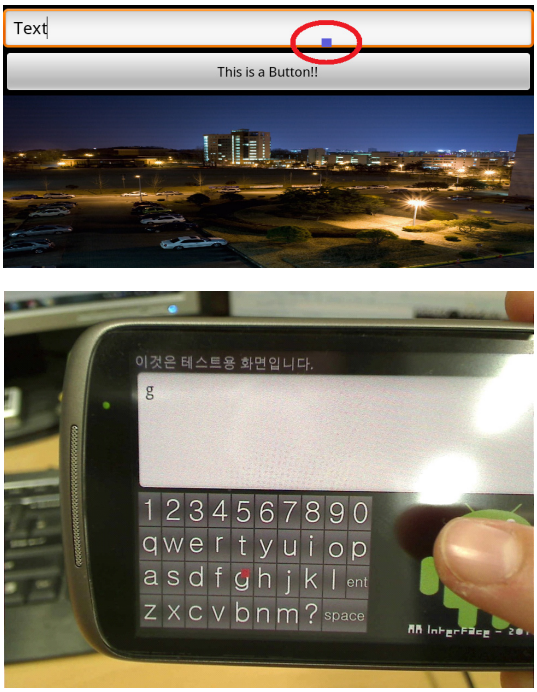


Figure 13. An example of implementation.

For recognizing markers, an open library, 'andAR' which is the Android version of ARToolkit (ARToolkit is the most popular library for tracking markers.) is used. The andAR library obtains the actual position of the marker relative to the position of the camera in the 3D coordinates, so we obtained the 2D coordinates of the mouse pointer by projecting the 3D coordinates on the 2D screen. The system shows a square-shaped mouse pointer on the obtained 2D coordinates.

The screen configuration is shown in Figure 12. The camera preview is located at the hindmost, and some buttons, input boxes and background images are located in front of the camera preview. The mouse pointer is displayed at the top, so the user can see which part will be selected. In Figure 12, an example of the actual implementation is shown. The camera preview is hidden for preventing confusions. The square-shaped mouse pointer is shown in purple on the OnRelease condition, and shown in pink on the OnPress condition. Figure 13 also shows an implementing result on the real device.

As explained above, "click", "long press", and "drag" functions were implemented using default touch input events (OnPress, OnRelease). Figure 14 shows how to use this system.

4.2 Result of the experiments

For verification, we compared the results of four methods; using the conventional finger touch method, using a capacitive touch pen, and using the two methods that were proposed.

Figure 14. How to use the system.

We used a 3M MHP-1000S model as the capacitive touch pen and a sticker marker 7 mm * 7 mm in size.

- Method 0 – Conventional touch method
- Method 1 – Using a capacitive touch pen
- Method 2 – Using the position and the rotation of a marker
- Method 3 – Using the position of a marker and touch commands

The testers are graduate students in their twenties who have used smartphones for more than 2 years. The four types of test case set were tested for each Method 0, 1, 2, and 3. Each test case sets are shown in Table 2. Only input accuracy is tested without considering input speed in these tests. The testers made drag and drop commands in given directions as accurately as possible for each method, and then the system calculated L2 error distance between the actual input points and given input points for each test.

Table 3 and Figure 15 show the average values of errors for each test case and each method. Method 0 (touch by a finger) has a large error compared to others. Especially when the user drags in the invisible direction covered by the hand – like the situation where a right-handed user drags from left to right (case 1, 4), a significant error occurs on the endpoint. In contrast, when the user drags from top to bottom as in case 3, the error in the starting point is similar to the end point, because the hand does not cover the endpoint. This result shows that the input accuracy is greatly affected by whether the target is covered by the user's hands or not when performing a touch input.

Method 1 (using a capacitive touch pen) is showing a better result than method 0. The result is not affected by targets and directions, but every endpoint has a large error compared to the starting points. The result shows that the error when using a touch pen is smaller than using fingers - but it also causes the coverage problem on the target point.

Table 2. Test cases for the input test [2, 3].

	Start point	End point
Case 1	(200,100)	(600,300)
Case 2	(600,100)	(200,300)
Case 3	(400,100)	(400,300)
Case 4	(200,200)	(600,200)

Table 3. Results: average values of distance error from the input test.

		Method 0	Method 1	Method 2	Method 3
Case 1	Start	10.04	13.26	8.94	2.76
	End	27.35	14.54	8.30	3.81
Case 2	Start	27.22	11.46	8.09	4.45
	End	16.10	16.79	5.68	2.30
Case 3	Start	21.92	8.95	6.88	3.21
	End	22.42	19.36	5.91	2.35
Case 4	Start	14.16	15.52	8.74	2.59
	End	29.06	17.54	9.29	2.51
Total		21.03	14.68	7.73	3.00

On the other hand, method 2 and 3 show that results are not affected by the drag directions. However, method 2 shows inaccurate results compared to method 3 because of rotating errors of the marker.

There can be errors from the input of the directions or various situations, but method 2 shows 2.7 times less error than that of method 0, and 1.9 times less error than that of method 1 in these tests. Also, method 3 has 7 times less error than that of method 0, and 4.9 times less error than that of method 1.

4.3 Analysis of the result

This experiment is focused on testing the accuracy of inputs to show that the conventional input method has low accuracy and also needs much more time to make an accurate input. However, in situations where accurate inputs aren't needed, the user can make a faster input with the conventional method, so additional experiments are required, taking into consideration the time consumption and the accuracy of inputs to validate the availability of our method in the future.

When operating this system on smartphones, the camera module in the smartphone is used so the short battery duration may be a problem. To alleviate this problem, "accurate input mode" can be toggled for saving unnecessary battery consumption in this system.

Because the image processing technique is used for recognizing a marker, the darkness of the environment or shadows can affect the result. In this research, we use a simple square

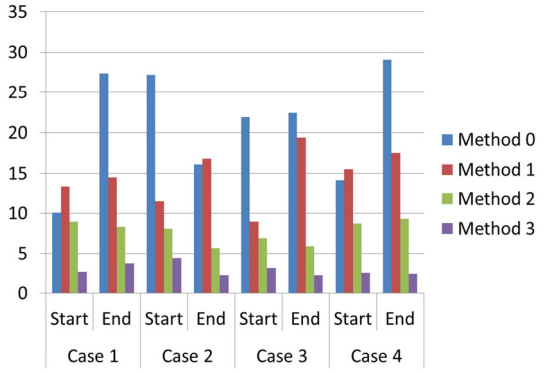


Figure 15. Graph of the results.

marker, so it's more robust than tracking a natural marker. Also, the most common problem in recognizing a square marker is the tracking failure according to the angle of a marker. However, in this research, it can be assumed that the marker is always placed in the front of the camera in parallel so the recognition error can be decreased. Also, the flash of the smartphone can be a solution for the dark environment. In that case, the recognition failure hardly occurs, but because it can accelerate the battery consumption, it is not appropriate for long term usage.

5. Conclusions

In this research, a new input interface for smartphones sing AR markers was proposed. We proposed two methods for reducing input errors on smartphones, and checked the possibility of accurate control on smartphones through the experiment.

When making capacitive touch commands is difficult or impossible, or when the user's fingers are relatively large, making touch inputs will be easier using the interface that adopted the methods explained above. Also, some applications that require detail works, such as Photoshop (editing pictures), sketching, painting, and making notes, will be easier to use. More applications that require detailed controls can be developed, and AR shooter games can be supported from these methods. In the future, the recognition rate, availability and efficiency of our methods on the applications which require detailed inputs should be tested.

In the method using the orientation of a marker only, there is a disadvantage when the marker is rotated on the target point, but if the marker that is attached to the portable object like a pen, an eraser, or a key ring is rotated, the user would be able to use this interface more easily.

Both methods of this research adopted image processing technique for recognizing a marker. If this interface operates with some heavy applications at the same time, delay can occur because image processing uses a lot of resources. These problems can be solved to apply a more robust and lightweight algorithm for the marker recognition.

In addition, this interface was developed in an integral system with one application, so the user cannot use it in other applications currently. In the Android OS, an independent application cannot access another application. Hence, in the future, we will provide this interface on OS-level, and then the user will be able to use this interface in other applications as a type of soft keyboard on smartphones.

Acknowledgments

This research was supported by Development of Integration and Automation Technology for Nuclear Plant Lifecycle Management grant funded by the Korea Government Ministry of Knowledge Economy (2011T100200145).

This work was supported by the Human Resources Development Program (No. 20134030200300) of the Korea Institute of Energy Technology Evaluation and Planning (KETEP) Grant funded by the Korea Government Ministry of Trade, Industry and Energy.

References

[1] Wikipedia [Internet]. c2013 [cited 2013 Sep 14]. Available from: http://www.wikipedia.org/

[2] Kang YA, Han SH. Improvement of smartphone interface using AR marker. Transaction of the Society of CAD/CAM Engineers. 2011; 16(5): 361-369.

[3] Kang YA, Han SH. Improvement of smartphone interface using an AR marker. In: Proceedings of the 11th International Conference on Virtual Reality Continuum and its Applications in Industry; 2012 Dec; NTU, Singapore.

[4] Endo Y, Tada M, Mochimaru M. Reconstructing individual hand models from motion capture data. Journal of Computational Design and Engineering. 2014; 1(1): 1-12.

[5] Huh SJ, Lee SW. A hierarchical Bayesian network for real-time continuous hand gesture recognition. Journal of Korea Institute of Information Scientists and Engineers: Softwares and Applications. 2009; 36(12): 967-1039.

[6] Starner T, Pentland A. Real-time American sign language recognition from video using hidden Markov models. MIT Media Lab., MIT, Cambridge, MA, 1995; Tech. Rep. TR-375, p. 1195-1207.

[7] De Silva LC, Aizawa K, Hatori M. Detection and tracking of facial feature by using edge pixel counting and deformable circular template matching. IEICE Transactions on Information and System. 1995; E78-D(9): 1195-1207.

[8] Oh ST, Jun BH. Head gesture recognition using facial pose states and automata technique. Journal of Korea Institute of Information Scientists and Engineers: Softwares and Applications. 2001; 28(12): 947-954.

[9] Bondarchuk V, Jung IR, Kim CS, Koh SJ. Implementation of mobile user interface control using camera captured information. In: Proceedings of Summer Conference of the Institute of Electronics Engineers of Korea; 2009 Jul 8-10; Jeju, Korea; p. 873-874.

[10] Lee CS, Cheon SY, Sohn MG, Lee SH. Hand gesture interface using mobile camera devices. Journal of Korea Institute of Information Scientists and Engineers: Computing Practices and Letters. 2010; 16(5): 621-625.

[11] Rohs M, Zweifel P. A conceptual framework for camera phone-based interaction techniques, Proceedings of the Third International Conference on Pervasive Computing; 2005 May 08-13; Munich, Germany; p. 171-189.

[12] Hansen TR, Eriksson E, Lykke-Olesen A. Mixed interaction space: designing for camera based interaction with mobile devices. In: International Conference of Human-Computer Interaction; 2005 Apr; Portland, OR.

[13] Park HJ, Moon HC. AR-based tangible interaction using a finger fixture for digital handheld products. Transaction of the Society of CAD/CAM Engineers. 2011; 16(1): 1-10.

[14] Hachet M, Pouderoux J, Guitton P. A camera-based interface for interaction with mobile handheld computers. In: Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games; 2005 Apr 3-6; Washington D.C., USA; p. 65-72.

[15] Byun JH, Kim MS. Tangible interaction: application for a new interface method for mobile device - Focused on development of virtual keyboard using camera input. Journal of Korean Society of Design Science. 2004; 17(3): 441-448.

[16] Lee JH, Han SH. Application of mixed reality with safety sign panel for a manufacturing simulation. In: Proceeding of the Society of CAD/CAM Conference; 2007 Jan 31-Feb 2; Pyungchang, Gangwondo; p. 582-588.

[17] Lee JH, Han SH, Cheon SU. Recognition of safety sign panel for mixed reality application in a factory layout planning. Transaction of the Society of CAD/CAM Engineers. 2009; 14(1): 42-49.