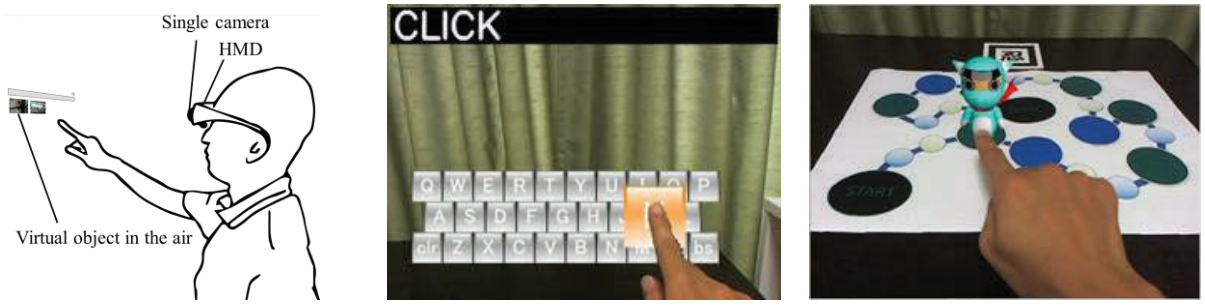
A Natural Click Interface for AR Systems with a Single Camera

Atsushi Sugiura*
University of Yamanashi

Masahiro Toyoura†
University of Yamanashi

Xiaoyang Mao‡
University of Yamanashi



(a) Configuration of the prototype system.

(b) Typing on a virtual keyboard.

(c) Operating a virtual object by clicking a position in the real world.

Figure 1: Overview of the proposed interface. A user wears a head-mounted display with a single camera. He clicks virtual buttons in the air with his finger.

ABSTRACT

Clicking on a virtual object is the most fundamental and important interaction in augmented reality (AR). However, existing AR systems do not support natural click interfaces, because head-mounted displays with only one camera are usually adopted to realize augmented reality and it is difficult to recognize an arbitrary gesture without accurate depth information. For the ease of detection, some systems force users to make unintuitive gestures, such as pinching with the thumb and forefinger. This paper presents a new natural click interface for AR systems. Through a study investigating how users intuitively click virtual objects in AR systems, we found that the speed and acceleration of fingertips provide cues for detecting click gestures. Based on our findings, we developed a new technique for recognizing natural click gestures with a single camera by focusing on temporal differentials between adjacent frames. We further validated the effectiveness of the recognition algorithm and the usability of our new interface through experiments.

Keywords: Wearable system, Augmented reality, Gesture recognition, Mobile application.

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems Artificial, augmented, and virtual realities; I. 2. 10 [Artificial intelligence]: Vision and Scene Understanding Motion

1 INTRODUCTION

In this paper, we propose a natural input interface for AR systems. Figure 1(a) shows the configuration of a prototype system with the proposed interface. It consists of a HMD display and one single camera. The camera is assumed to be installed around the user's eyes and to face outwards along the view direction. A user can use his hand to directly operate a virtual object in the augmented environment shown on the HMD screen. Figure 1(b) is the HMD screen snapshot of the user typing on a virtual keyboard with his finger. Figure 1(c) demonstrates the application of the new interface to a game in which the user can move a virtual object by clicking a position in the real world, which is captured by the camera.

With the widespread adoption of smartphones and other camera-installed mobile devices, AR technology has become an important part of many practical applications and services. Google announced that the long-awaited Google GLASS [1] will become available for sale in early 2014, which is expected to further boost the dissemination of AR devices. Google GLASS provides a see-through display in front of an eye and a camera facing outwards, capturing the real world. However, it relies on voice recognition and a small touch panel input for command execution. A more natural interface for such an AR system would be a gesture-based interface, which would allow the user to interact with a virtual object in a similar way as he or she interacts with objects with using his or her hands in the real world. However, even with the most state-of-the-art AR systems, this kind of true direct manipulation, which is considered vital for the seamless connection between the virtual world shown on the display and the real world captured by the camera, has not been realized. A major factor preventing existing AR systems from supporting gesture-based direct manipulation is the difficulty of recognizing an arbitrary gesture with a single camera, which cannot capture accurate depth information. Although several projects developing wearable display devices with stereo-camera [17] have been reported recently, developing a single-camera-based AR system remains important, especially considering the population of AR applications on compact mobile devices, such as cellphones.

In this work, we aim to recognize natural hand gestures with a single camera. In particular, we focus on click gestures. In a traditional graphical interface, a click refers to the action of placing the cursor on a target and then pressing a button on the mouse to select an object or execute a command. It is the most essential operation. Here, we define a click operation in a VR system as an intuitive gesture a user would make to select a virtual object or execute a graphically represented command. Through a

* email: g12dhl02@yamanashi.ac.jp
† email: mtoyoura@yamanashi.ac.jp
‡ email: mao@yamanashi.ac.jp

study investigating how subjects perform when they are told to "click" virtual objects with their fingers without any instructions or training, we found that the velocity and acceleration of fingers provide useful cues for detecting click gestures. Based on our findings, we defined a new motion-based model for intuitive click gestures and developed a novel technique for recognizing such gestures with a single camera. Our technique does not assume controllable illuminations or an accurate 3D capturing of hand. Therefore, it can be implemented in any AR system, including those using mobile phones and head-mounted displays (HMDs).

The major contributions of this paper can be summarized as follows:

- Design a novel study for investigating what an intuitive click gesture in an AR system is.
- Introduce a new motion state transition model for recognizing the click gestures.
- Implement two new algorithms for detecting the click gestures in an AR system based on the motion of a finger.
- Conduct experiments for evaluating the new click interfaces in terms of click gesture recognition performance and intuitiveness.

The remainder of the paper is organized as follows: Section 2 introduces the related works. Section 3 describes the study. Sections 4 and 5 present the technical details of the proposed click interface. Section 6 describes the implementation issues and the experiments for evaluating the proposed techniques. Section 7 concludes the paper.

2 Related Works

The creation of a natural user interface using a single camera is an active research topic in the fields of human computer interaction and computer vision. Projects on tangible interfaces [2-3], tabletop interfaces [4-5], projector-camera systems [6-7], Kinect [8] and the iPad [9] have attracted a great deal of attention. Especially in the case of tabletop interfaces, bare-hand gesture recognition from camera-captured images is often employed as a tool for inputting commands. Attempts to extract hand regions from captured images always face the challenges of self-occlusion, unpredictable illumination, cluttered backgrounds, blurred images, and so on. Attracted by the ease and robustness of recognition, conventional systems employ hand-shape-based command inputs [10] or pinching gesture detection [11], which unfortunately force users to perform predefined gestures rather than intuitive ones.

Although hand gesture recognition is one of classic problems in computer vision, it is still under active research. A survey of state-of-the-art hand recognition techniques can be found in [15]. The difficulty of hand recognition comes from 1) a deformable and flexible object with multiple joints having a high DOF (degree of freedom), 2) self-occlusion caused having by many joints, and 3) skin region extraction under fluctuating illumination.

The problem of estimating a high-DOF hand posture can be solved using textured gloves. Wang et al. [13] introduced a color glove and tried to estimate the posture and position of fingers from the texture of the glove. This work requires wearing textured gloves, which may limit the range of applications.

Self-occlusion caused by multiple joints can be addressed using multiple cameras. Multiple cameras can also reconstruct the 3D shape of a hand, which provides more information about the posture and position of the hand. Lee et al. [12] implemented a system supporting the interaction between virtual objects and a hand. In the system, skin color regions are first extracted from captured images. The stereo cameras provide 3D hand regions and fingertip positions. By computing the collision between the virtual objects and the line defined by using the gravity point and the fingertip position, the object the user is interacting with can be detected. Recently, another research group has begun developing a wearable display device [17] that combines a depth sensor [8] and an HMD to support hand gestures in AR systems. These multiple camera systems require calibration in advance. Furthermore, such systems tend to be bulky in size.

Skin region estimation under fluctuating illumination can be solved by progressively updating observed skin color. Kölsch et al. [18] addressed the accuracy of skin region extraction based on the idea of AR applications. In AR applications, cluttered backgrounds and objects of skin-like color are often observed. Their method can robustly extract hand regions, even in such an environment. The same group has also proposed shape-based hand region extraction [19-20]. Their main targets were static gesture recognition and hand position estimation for a hand. We adopt their intelligent skin color updating technique to detect dynamic clicking gestures.

In this paper, we propose an interaction system with no gloves, a single camera, and dynamic click detection. We identified a unique motion of the fingertip when performing click gestures through a study and developed a robust recognition technique that involves detecting such a motion with a single camera.

3 User Study

We have conducted a study to investigate what a natural click gesture in an AR system is. Twelve subjects of varying ages (four in their 30s, six in their 20s, and one in his or her teens) and varying levels of computer skill participated in the experiment. The subjects sat on a chair, wearing a video-see-through HMD (Vuzix WRAP AR920). The resolution of the monitor is 800x600. It has a dual-channel output, but we only used a single-channel output. The same image is displayed on both monitors. An additional Logicool QCam Pro 9000 camera is installed on the HMD, which captures the video of the operating scene at a resolution of 800x600. For the task, the subjects were asked to "click" each of the virtual buttons once with the pointing finger of their dominant hand (all twelve subjects were right-handed), without any detailed instructions or training. To investigate how 3D positions and the orientation of buttons may affect click gestures, we used two sets of virtual buttons. As shown in Figure 2(a), the first set of buttons consisted of five buttons in a cross-shaped layout, and they were oriented so as to be parallel to the XY plane. The second set, as shown in Figure 2(b), was oriented in the depth (Z) direction. During the operation, the finger is always displayed in front of the virtual buttons. No other visual, aural, or haptic feedbacks were provided for the interaction between the finger and a button. And no haptic feedback forced the subject to stop the movement of their fingers in the air, although most of the subjects commented that it would have been better to provide some feedback indicating that the click was completed. They also found it difficult to place their fingers on a target button in the HMD display.

(a) Buttons parallel to XY plane. (b) Buttons along Z direction.

Figure 2: The arrangement of virtual buttons.

(a) Type 1: Pull and push.

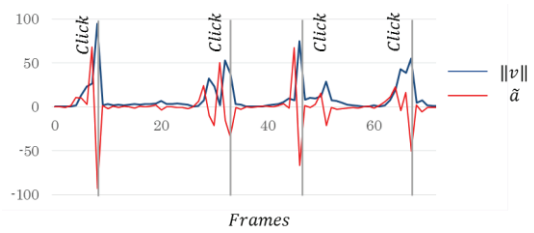
(b) Type 2: Push and pull.

Figure 3: Click gesture by users.

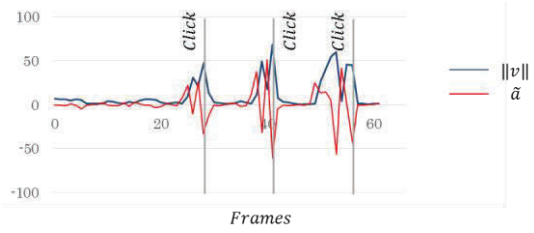
To track the 3D trajectory, velocity, and acceleration of the fingertip, we installed to capture the 3D position of fingertips with a LeapMotion Controller [6] from LEAP Inc. on a table in front of the seat of a user. However, due to the limited observation range, with the radius of 50 centimetres from LeapMotion, we could only obtain the motion data for six out of the twelve subjects. The other six subjects performed partially out of the range during this experiment.

To understand the common patterns involved in a click gesture, we conducted a post-task interview with each subject. Each subject was presented with the video of himself/herself performing the task and asked to explain in detail the motion of his/her finger. Based on the post-task interview and the data from LeapMotion, we observed the following facts to be common to all or the majority of subjects:

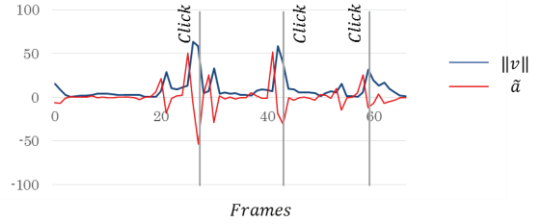
1. Based on the post-task interview, we found the subjects trying to confirm that their fingertips were on top of a button before clicking the button. The LeapMotion video also confirmed this statement. Also see the supplemental movie to confirm the gestures.
2. Based on the data tracked by LeapMotion, we found that the depth at which subjects tried to click the buttons varied by subject. Based on the post-task interview, we confirmed that this is because the subjects could not perceive the correct position of the buttons in terms of depth.
3. From the LeapMotion video, we observed that click gestures are similar to but more exaggerated than a general tapping gesture. In the interview, most subjects commented that because there was no haptic feedback for touching a virtual button, they tried to represent the click with an exaggerated gesture. Specifically, they had to stop their fingers to represent the click because there was no haptic feedback upon touching a button. Ten subjects first raised their fingers up, pushed down quickly, and then stopped on the button suddenly (Figure 3(a)). Two subjects pushed their fingers down slowly first and then raised them quickly (Figure 3(b)).
4. By analyzing the data from LeapMotion, we found that the click gesture described in (3), including its 3D motions, varies by subject, as well as by the position of the buttons. In other words, it includes the motion not only in the Z direction but also in the X and Y directions, depending on the relative position of the button and the hand.



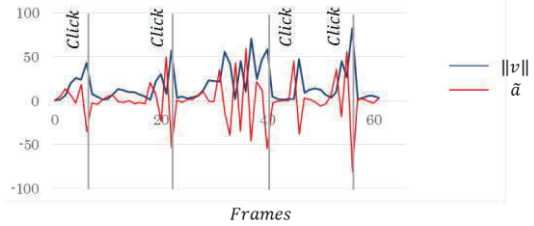
(a) Fingertip's speed and its differential for Subject A.



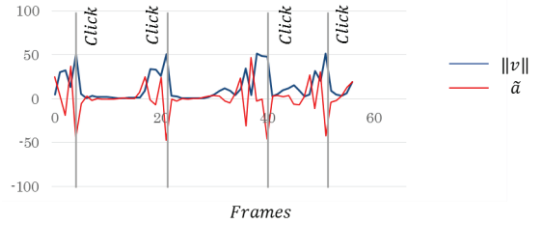
(b) Fingertip's speed and its differential for Subject B.



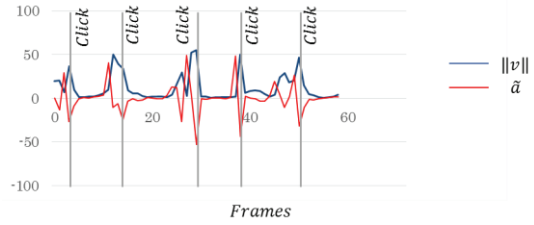
(c) Fingertip's speed and its differential for Subject C.



(d) Fingertip's speed and its differential for Subject D.



(e) Fingertip's speed and its differential for Subject E.



(f) Fingertip's speed and its differential for Subject F.

Figure 4: Fingertip's speed and its differential in click gesture, as tracked with LeapMotion [6].

Facts 1 and 2 suggest the necessity of providing some kind of feedback to the user regarding the pressing of a particular virtual button. As described in Section 1, we provide such feedback by changing the size and color of the button pressed.

Fact 3 suggests that the speed of the fingertip is an important cue for detecting clicks. Figure 4 shows the speed (magnitude of velocity) $\|v\|$ and the differential of speed $\tilde{a}$ for the six subjects' fingertips as tracked by LeapMotion. We observed that, when a click occurred, the differential of speed increased first and then dropped down drastically. We also observed a large peak in speed around the moment of the click. Such a characteristic motion within the gesture will help to distinguish it from other kinds of finger motions. Fact 1 suggests that there was usually a pointing gesture before the clicking gesture. Combining the detection of this pointing gesture has the potential to improve the performance of click gesture detection. Finally, Fact 4 suggests that we should consider the velocity in 3D when computing the speed.

4 Hand Area Extraction and Fingertip Position Estimation

Hand area extraction is necessary both for providing visual feedback to users and for detecting the click gesture. To provide a natural click interface, we should render the operating scene as if the user is operating an object with his/her hand in the real world. For this purpose, we extract the hand area from the image captured by the camera and render it in front of the virtual objects. To track the motion of the fingertip, we need a further estimate of the position of the fingertip.

4.1 Hand area extraction

We extract hand area by assuming it to be the region with skin color. Numerous other projects have already been performed regarding skin color detection and hand region extraction. Nevertheless, they are still problems in such cases. For example, illumination is often uncontrollable, such as in the outdoors or a dark place. Because developing sophisticated skin area detection is not our main focus, we employ a classic method, a Gaussian mixture skin color model [21], for extracting hand regions. The use of more recent tracking-based methods [18] or shape-based methods [20] would improve the accuracy of extracted hand regions. Note that computational cost should remain low when installing such methods into our system.

Assuming skin color can be represented with a single Gaussian model in HSV color space given a representative skin color $S = (h_s, s_s, v_s)$, we compute the distance of each pixel from S and extract all pixels with a distance smaller than a given threshold. In the current implementation, we measure each component of HSV separately. Assuming the thresholds for H, S, and V $to\ be\ h_{th}, s_{th}, v_{th}$, respectively, a pixel $P = (h_p, s_p, v_p)$ is detected to be of skin color if it satisfies all of the following three conditions:

$$h_s - h_{th} \le h_p \le h_s + h_{th} \quad (1)$$

$$s_s - s_{th} \le s_p \le s_s + s_{th} \quad (2)$$

$$v_s - v_{th} \le v_p \le v_s + v_{th} \quad (3)$$

The hand area is then detected as the largest connected component of the extracted skin-colored pixels.

Our system provides a calibration tool that allows a user to interactively specify an initial representative skin color and adjust the threshold using the initial frame captured in the assumed environment. During runtime, the hand area extracted from the previous frame is used as the training data for the next frame. When the hand region is not detected in a frame, the system abandons and resets the training data. Therefore, if the user is not satisfied with the result of the hand region extraction, he/she can let the algorithm restart from the current frame simply by removing his/her hand from the camera view once and then putting it back. We detect the skin-colored pixels by computing their Mahalanobis distance from the average color of the hand area in the previous frame. Assuming that $\mu = (\mu_h, \mu_s, \mu_v)$ is the average and Σ is the variance-covariance matrix of the color for the hand area extracted in the previous frame, a pixel P of the current frame satisfying the following condition is detected to be of skin color:

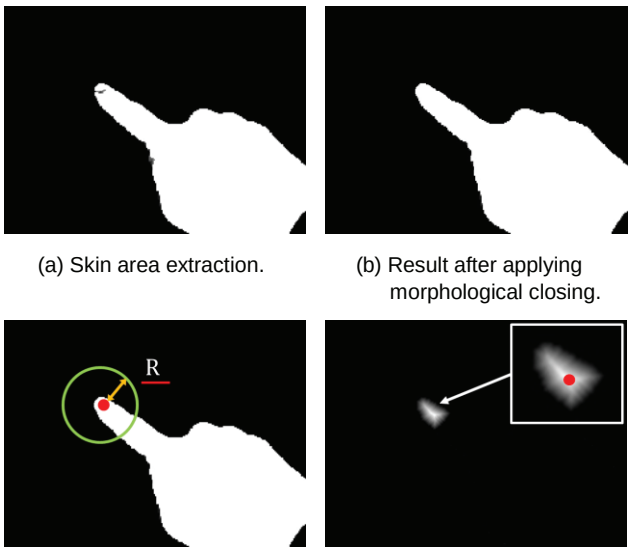
$$(P - \mu)^T \Sigma^{-1} (P - \mu) \le d_{th} \quad (4)$$

The threshold d_{th} is empirically set to 0.5 in our experiment. Using the previous frame as the training data makes the algorithm more robust considering the dynamic changes in lighting conditions.

As shown in Figure 5(a), depending on the lighting conditions, the claw area may not be successfully detected and appear as a hole in the hand area. We fill this potential hole by applying morphological closing operations to the extracted skin-colored area (Figure 5(b)).

4.2 Fingertip extraction

As shown in Figure 5(b), assuming the top-left corner of the captured image as the origin of coordinates, we take the pixel with the smallest y in the hand region as the temporarily assumed fingertip. Such an assumption is rational because the hand is expected to come from the bottom. Then, we draw a circle with the temporarily assumed fingertip as the center and R (given in advance) as the radius. We next perform a distance transformation for the finger region enclosed in the circle (Figure 5(c)). As a result, the pixels on the boundary of the finger region receive 0 as their distance value, and regarding the pixels inside the region, the farther they are from the boundary, the larger distance value they receive. The position of the fingertip is finally estimated as the peak of the parabola fitted curve for the distance values, as shown in Figure 5(d).



(a) Skin area extraction. (b) Result after applying morphological closing.

(c) Fingertip area extraction (d) Fingertip position detection.

Figure 5: Hand area detection and fingertip position estimation.

5 Proposed Virtual Click Interface

5.1 Motion-Based Click Model

Based on the observations from the study, we have recognized click gestures by analyzing the motion of the fingertip. We first classify the motion of a fingertip into one of the following four states:

STILL: stop at a position
MOVE: move at a normal speed.
FAST: move quickly
Sudden SD: slow down suddenly

Then, a click gesture can be modeled as the state transition diagram in Figure 6 shows. To perform a click, the user starts by confirming the pointing at a virtual object (STILL or MOVE), quickly raises his/her finger (FAST), and then pushes toward the object. Finally, the speed drops suddenly (Sudden SD) before stopping at the object (STILL).

State transitions can be detected by monitoring the fingertip's speed $\|v\|$ (magnitude of velocity) and the differential of the speed $\tilde{a}$ based on the diagram shown in Figure 7. The transition from STILL to MOVE can be detected by checking whether the current speed is above a given threshold. If both $\|v\|$ and $\tilde{a}$ become large, then a transition from MOVE to FAST has probably occurred. A Sudden SD is detected by checking whether $\tilde{a}$ is smaller than a given negative value. Although the current recognition algorithm relies on choosing an appropriate threshold, it is relative easy to find a robust threshold because $\|v\|$and $\tilde{a}$ show large peaks around the moment of the click, as confirmed in the study. As mentioned in Section 6, we have implemented a calibration tool for adapting the thresholds to individual users.

From the primary study, we have learned that the most characteristic feature of the click motion is the Sudden SD state, which distinguishes the click from all other motions of the finger. Therefore, we also implemented a simpler but efficient algorithm that recognizes a click gesture simply by detecting the Sudden SD state.

Figure 6: The state transition of the click motion.

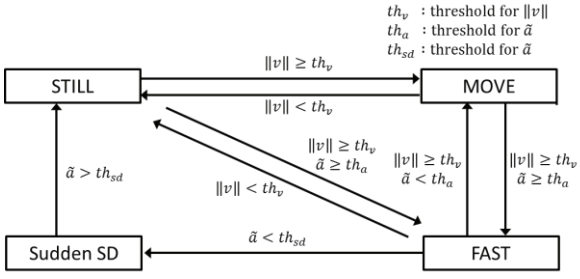


Figure 7: State transition diagram for detecting states of motion

5.2 Motion detection

To recognize the click gesture using the state transition diagram given in Section 5.1, we must compute the speed and the differential of the speed of the fingertip. As observed in the study, a click gesture is a 3D motion that includes the movement not only in the XY plane but also along the Z (depth) direction. However, because we assume single-camera-based AR systems, it is impossible to track the motion in the depth direction directly. To solve the problem, we use the change in object size due to perspective projection as the cue to estimate the change in the depth direction. In particular, we use the change in finger width to approximate the Z component of the speed and the differential of the speed.

Denoting the position of the fingertip on the XY plane as (x_t, y_t) and the width of the finger as w_t for a frame t, the approximate speed $\|v_t\|$ and the differential of speed $\tilde{a}_t$ of the fingertip for frame t are computed as follows:

$$\|v_t\| = \|(x_t, y_t, w_t) - (x_{t-1}, y_{t-1}, w_{t-1})\| \quad (5)$$

$$\tilde{a}_t = \|v_t\| - \|v_{t-1}\| \quad (6)$$

Note that $\|v_t\|$ and $\tilde{a}_t$ are calculated as the displacement and its differential in a unit of time. Because the acceleration, as the secondary differential of position, is always a positive value and there is no way to distinguish "suddenly slow down" from "suddenly speed up" simply by computing the acceleration, we compute $\tilde{a}_t$ as the differential of speed $\|v_t\|$ instead of the differential of velocity vector v_t and use the sign of $\tilde{a}_t$ to distinguish "suddenly slow down" from "suddenly speed up." "Suddenly slow down" is indicated by a small negative $\tilde{a}_t$. As shown in Figure 8, w_t is computed as the distance between the two intersections of the finger region boundary and a circle centered on the fingertip (x_t, y_t). To make the algorithm more robust, we compute the average of the distance obtained by using five circles of different radii.

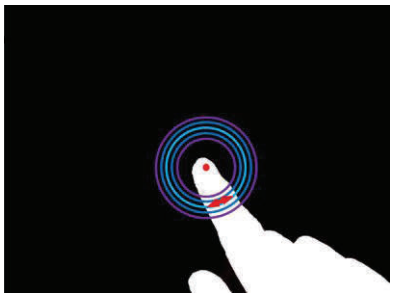


Figure 8: Estimation of finger width.

5.3 Hand and Fingertip Detection

Without haptic feedback, it is very difficult for users to perceive the relative position between their fingertips and virtual objects. As observed in the study, a subject tries to confirm that his/her fingertip is on the top of a button before performing the click. Therefore, providing some kind of feedback to notify the user of whether an object is ready for clicking is very important.

In traditional GUI, changing color is a commonly used approach to providing visual feedback about pointing at an object. In their gaze-based system, Majaranta et al. [23] proposed to change the color of a button when it was gazed at to provide the visual feedback that supports effective text input. Terajima et al. [5] succeeded in providing visual feedback for the touching of a virtual keyboard by changing the size of key buttons. We also employed changing the color and size of a virtual button when it was pointed at, as shown in Figure 9.

Pointing is detected by checking whether the position of the fingertip (x_t,y_t) is within the area of a virtual object for a certain period. Too long a period annoys users attempting to push a button, while too short a period causes a false-positive clickable

state for the button, which leads to the Midas touch problem. We empirically set the period to four frames, or 266msec in 15fps, in our current implementation.

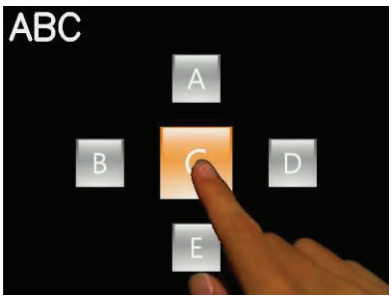


Figure 9: Visual feedback for pointing.

6 EXPERIMENTS

6.1 Implementation

To verify the effectiveness of proposed click interface, we have implemented a prototype system, as depicted in Figure 1. An HMD (Wrap920AR, Vuzix Cooporation) was connected to a laptop PC (OS : Windows 8, CPU: Core i5, CPU: 2.5GHz, MM: 4GB). The HMD originally possessed two VGA (640x480) USB cameras. We used one of the cameras for the experiment. The resolution of the displays was SVGA (800x600).

Two click gesture recognition algorithms are implemented. The first algorithm recognizes a click gesture by detecting a sudden drop-off in speed before stopping at a pointed-at virtual object. The second algorithm uses the state transition diagram based on the click model shown in Figure 7. When using the state-diagram-based method, choosing appropriate thresholds is especially crucial to achieving a high success rate. We have implemented a calibration tool for adapting the thresholds to individual users. Figure 10 shows a screenshot of the tool. The user is asked to click a button a few times, and the system computes the distribution of $\|v\|$ and $\tilde{a}$ and automatically finds the best thresholds for the user.

Figure 10: Calibration tool for adapting thresholds to individual users.

6.2 Evaluation

We have tested the effectiveness of the proposed visual feedback and the click gesture detection technique via subject studies. To evaluate the intuitiveness of the proposed click interface, children and seniors were also invited as subjects.

Two sets of virtual buttons are used in the experiment. Because the motion of the finger may vary according to the position relative to the virtual object, the first set of buttons, which is the same as the one used in the study, is designed to test the relative position factor among a virtual object, the finger and the camera. As shown in Figure 11(a), we use five virtual buttons in a cross-shaped layout for inputting the letters A~E. The size of each button is 80×80 pixels on the screen. The button for the letter C is placed in the center of the screen. The distance from the top and bottom buttons to the center button is 140 pixels, and the distance from the left and right buttons to the center button is 160 pixels. The second set of virtual buttons is designed to investigate whether the proposed detection algorithm is effective even when the virtual buttons are very close to one another. When the buttons are placed very close to one another, a part of the fingertip may overlap with the adjacent buttons and thus may affect the performance of click gesture detection. We expect that the visual feedback involving color and size will not only help user to point at the button more precisely but can also improve the performance of gesture detection. As shown in Figure 11(b), as a potential application, we designed a virtual calculator consisting of 18 squared buttons of 60*60 pixels and two rectangular buttons, "0" and "=," of 60*125 pixels. The distances in both the horizontal and vertical directions between the centers of the two adjacent buttons are 125 pixels. The color of a button changes to orange when a pointing is detected. The color changes to red, and the size changes to 1.5 times the original size if a click is detected.

(a) Character input buttons. (b) Virtual Calculator.

Figure 11: Arrangement of virtual buttons for subject study.

6.2.1 Click gesture recognition

We tested the click gesture recognition algorithms based on sudden speed drop-off detection (Test A) and the state transition model (Test B) and compared the results of the two algorithms. Twenty subjects, including nine males and eleven females in their teens, 20s, and 30s, participated in both tests. To eliminate the learning effect, they were divided into two groups of the same size. The subjects in the first group participated in Test A first and then Test B, while with was reversed in the second group.

For each test, a subject was asked to perform five trials for the character inputting task and ten trials for the calculation task. For each trail of the character inputting task, the subject was asked to input ten characters. For each trial of the calculation task, the subject was asked to input an equation, such as "123+423=", to add, subtract, multiply, or divide two 3-digit numbers. Before starting the trial, each subject was allowed to practice for 1~2 minutes. The reason we allowed the users to practice is that it was the first time most subjects had worn an HMD. As a future project, we plan to use a different task that can help user to get used to the HMD without any learning effect regarding the proposed interface.

6.2.1.1 Test A: by detecting sudden speed drop-off

Performance data in terms of precision (ratio of true clicks over all detected clicks), recall (ratio of detected clicks over all the true clicks), and F-measure for the 20 subjects are presented in Figures 12 and 13. The performance statistics are given in Table 1. As we can see, the averages of the three measures are all above 93%. There are no significant differences between the two tasks. The false detections were mainly caused by the failure of skin area

detection, which results in incorrect fingertip position. In the experiment, we observed that slow clicking gestures tend to be missed.

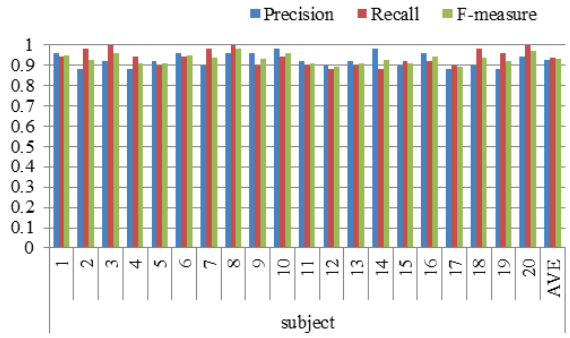


Figure 12: Performance of click gesture detection via sudden speed drop-off of fingertips for the character input task.

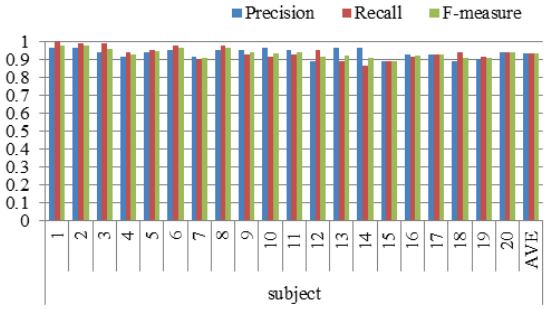


Figure 13: Performance of click gesture detection via sudden speed drop-off of fingertips for the calculation task.

Table 1: Statistics on the performance of click gesture detection via sudden speed drop-off of fingertips.

	Character			Calculator		
	Precision	Recall	F-measure	Precision	Recall	F-measure
AVE	0.93	0.94	0.93	0.93	0.93	0.93
MAX	0.98	1.00	0.98	0.96	1.00	0.98
MIN	0.88	0.88	0.89	0.89	0.86	0.89

6.2.1.2 Test B: by using the state transition model

The results and statistics are shown in Figures 14 and 15 and Table 2. Compared to the results of detecting sudden speed-drop off, all three measures went down. For the character input task, one subject's precision was below 80%. We conducted a T-test to compare the results of the two detection algorithms and found that there were significant differences at P=0.01 for precision, recall, and F-measure.

That is, the method of detecting the sudden speed drop-off outperforms the method using the state transition diagram. The main reason for this is that the latter uses multiple thresholds and it is difficult to find the best values for all the thresholds. However, the state transition diagram is a more general model and can be easily extended to other gestures. We are now improving the implementation of state transition detection by using statistical learning.

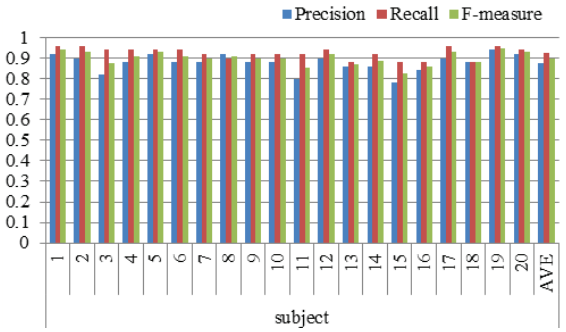


Figure 14: Performance of click gesture detection with the state transition model for the character input task.

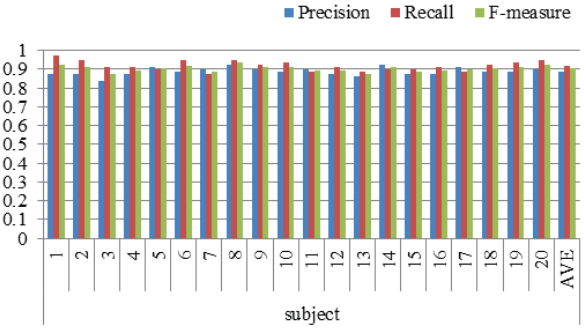


Figure 15: Performance of click gesture detection with the state transition model for the calculation test.

Table 2: Statistics on the performance of click gesture detection with the state transition model

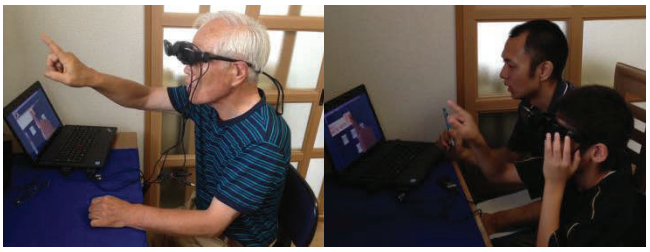
	Character			Calculator		
	Precision	Recall	F-measure	Precision	Recall	F-measure
AVE	0.88	0.93	0.90	0.89	0.92	0.90
MAX	0.94	0.96	0.95	0.93	0.98	0.94
MIN	0.78	0.88	0.83	0.84	0.88	0.87

6.3 Evaluation of the Intuitiveness of the Proposed Interface

To evaluate the intuitiveness of the proposed click interface, we invited seven senior subjects between 60 and 70 years old and six teenagers. None of them were familiar with the computer environment. They were asked to perform three trials for the character input task and five trials for the calculation task.

Figure 16 shows snapshots of the experiment. The performance and the statistics for all subjects are shown in Figure 17, Figure 18, Table 3, and Table 4. Only the sudden-speed-drop-off-based algorithm was tested. The performance of the seniors was lower than that of youth group. As shown by the T-test, there was a significant difference between the senior group and the youth group, and between the teenaged groups and youth group, with p=0.01, but there was no significant difference between the senior group and the teenager group.

Through interviews, we confirmed that it was very easy for senior subjects to learn the interface, even without any computer experience. The visual feedback also contributed largely to the usability of the interface. Some senior subjects reported that they felt like they were pushing a button in the real world and that the highlighting of a button with a different color made them feel the moment of pushing a button.



(a) A subject in his 70s. (b) A teenaged subject

Figure 16: Experimental environment.

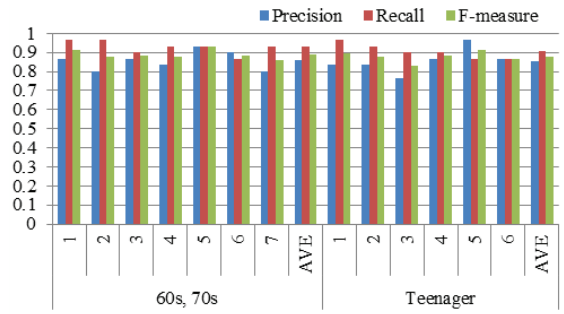


Figure 17: Click gesture detection performance for the senior and teenaged groups for the character input task (via detecting sudden speed drop-off).

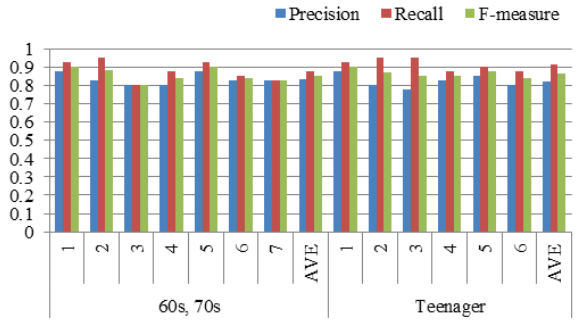


Figure 18: Click gesture detection performance for the senior and teenaged groups for the character input task (via detecting sudden speed drop-off).

Table 3: Statistics on click gesture detection performance for senior subjects.

	Character			Calculator		
	Precision	Recall	F-measure	Precision	Recall	F-measure
AVE	0.86	0.93	0.89	0.83	0.88	0.85
MAX	0.93	0.97	0.93	0.88	0.95	0.90
MIN	0.80	0.87	0.86	0.80	0.80	0.80

Table 4: Statistics on click gesture detection performance for teenaged subjects.

	Character			Calculator		
	Precision	Recall	F-measure	Precision	Recall	F-measure
AVE	0.86	0.91	0.88	0.82	0.91	0.86
MAX	0.97	0.97	0.91	0.88	0.95	0.90
MIN	0.77	0.87	0.83	0.78	0.88	0.84

DISCUSSION

Several subjects reported that the virtual buttons looked like real buttons. Clicking the buttons was fun and enjoyable. Several teenaged subjects became enthusiastic during the experiment because it felt like playing a game. Their comments support the idea that the proposed interface is intuitive for the subjects. On the other hand, several senior subjects claimed that clicking in the air made them tired. Therefore, the current click interface may not be suitable for use over an extended period of time.

Wearing an HMD was a new experience for all the subjects except for a few student colleagues. Nevertheless, most of those subjects did not have any particular difficulties in clicking the buttons displayed in front of them. The buttons were on the HMD coordinates, which makes it difficult to place a real finger on the virtual buttons. The subjects avoided this situation by adjusting their heads during the experiment. The problem can be solved by fixing the buttons to the world coordinates of the environment, which can be realized by using natural feature tracking and creating a 3D environment map.

7 CONCLUDING REMARKS

We have presented a novel click interface for AR systems with a single camera. With the new interface, a user can click an object in an AR environment in the same way he/she interacts with objects in the real world with his/her finger. A primary study was first conducted to build a model for a natural click gesture for AR systems. The effectiveness of the proposed gesture recognition algorithm, as well as the intuitiveness of the interface, was evaluated through subject studies.

The click is the most essential operation of interactive systems, so our system has a large variety of applications. For example, we can allow a user to type on a virtual keyboard or play a game with his/her, finger as shown in Figures 1(b) and (c), respectively. In Figure 1(c), our technique allows the user to click the positions on the papers, which are real objects captured by camera. Because we use a single camera, our technique can be used for building various user-friendly AR systems on compact devices. For example, a doctor can retrieve information about a patient without touching the screen of an iPad during an operation. Another example is that one can select the menu on a cell phone without touching the screen when one's hand is not clean.

As a future project, we would like to design a more elaborate experimental setup to test the effectiveness of our technique for different interface variables. One promising approach is to employ some standard benchmark systems, such as an AR variation of the FittsStudy software (http://depts.washington.edu/aimgroup/proj/fittsstudy).

The currently implemented skin area defection algorithm may fail to detect the hand area correctly if the background consists of objects of skin-like color. Because the accuracy of the extracted fingertip position is highly dependent on the accuracy of skin region extraction, we need to employ more robust techniques. Recent hand posture estimation methods may contribute to the solution of this problem. Making proper assumptions about the environment, such as use in front of a wall, could be a more practical solution. Currently, we detect the click gesture by either detecting the sudden slowdown or the state transition of the motion. By combing the detection of pointing gestures, we can confirm high performance for both algorithms in the evaluation tests and avoiding the Midas touch problem. However, more experiments are required before we can make broad claims about the effectiveness of the interface. For example, when the users become more familiar with the interface, they may move their fingers more quickly and need less time to confirm pointing. In that case, it may become difficult to distinguish the click gesture

from other quick movements. We are now improving the robustness of the gesture detection by constructing a probability model for the state transition of the click model. Currently, our technique only applies to the click gesture performed with a single finger. It will be another interesting future direction to extend the technique to recognizing other kinds of gestures.

Wearing an HMD degrades the intuitiveness of the system. The HMD did not fit well on a small child's head in the experiment. An optical-see-through HMD may provide another solution to enhance the intuitiveness of the system, but it may cause difficulty in registering the positions of a real finger and virtual buttons. We plan to explore the possibility of installing a gaze tracker on an optical-see-through HMD to register the position of a real finger and virtual buttons.

ACKNOWLEDGMENT

We are deeply grateful to Prof. Kwan-liu Ma, Dr. Lichan Hong and anonymous reviewers for the numerous valuable comments and constructive suggestions. This work was supported by JSPS KAKENHI Grant Number 25730120 and 25540045.

REFERENCES

[1] GoogleGlass, Google, http://www.google.com/glass/start/.

[2] L. Taehee, H. Tobias. Handy AR: Markerless Inspection of Augmented Reality Objects Using Fingertip Tracking. *IEEE Wearable Computers, 2007 11th IEEE International Symposium on*, pages 83-90. 2007.

[3] S. Hashimoto, A. Ishida, M. Inami and T. Igarashi. TouchMe: Direct Manipulation for Robot Based on Augmented Reality. *The 21st International Conference on Artificial Reality and Telexistence, Proceedings of ICAT2011*. 2011.

[4] T. Murase, A. Moteki, N. Ozawa, N. Hara, T. Nakai, K. Fujimoto. Gesture Keyboard Requiring Only One Camera. *Proceedings of the 24th annual ACM symposium adjunct on User interface software and technology*, pages 9-10. 2011.

[5] K. Terajima, T. Komuro and M. Ishikawa. Fast finger tracking system for in-air typing interface. *In Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 3739-3744. 2009.

[6] C. Harrison, H. Benko and A. D. Wilson. OmniTouch: Wearable Multitouch Interaction Everywhere. *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 441-450. 2011.

[7] C. Colombo, A. D. Bimbo and A. Valli. Visual capture and understanding of hand pointing actions in a 3-D environment, *Man, and Cybernetics, Part B: Cybernetics. IEEE Transactions on*, volume 33, number 4, pages 677-686. 2003.

[8] Kinect, Microsoft.

[9] iPad, Apple, http://store.apple.com/us.

[10] H. Kim, G. Albuquerque, S. Havemann and W. D. Fellner. Tangible 3D: Immersive 3D Modeling through Hand Gesture Interaction. Proceedings of the 11th Eurographics conference on Virtual Environments, pages 191-199. 2005.

[11] A. Wilson, Robust. Vision-Based Detection of Pinching for One and Two-Handed Input. *UIST*. 2006.

[12] M. Lee, R. Green, and M. Billinghurst. 3D Natural Hand Interaction for AR Applications. *Image and Vision Computing New Zealand 23rd International Conference*, pages.1-6. 2008.

[13] R. Y. Wang, and J. Popovic. Real-Time Hand-Tracking with a Color Glove. *Journal of ACM Transaction on Graphics*, volume 28, number 3. 2009.

[14] A. Akl. A Novel Accelerometer-Based Gesture Recognition System. *Signal Processing, IEEE Transactions on*, volume 59, number 12, pages 6197-6205. 2011.

[15] A. Chaudhary, J. L. Raheja, K. Das, & S. Raheja. Intelligent Approaches to interact with Machines using Hand Gesture Recognition in Natural way: A Survey. *International Journal of Computer Science & Engineering Survey*, volume 2, number 1, pages 122-133. 2011.

[16] Leap Motion, Inc, https://www.leapmotion.com/.

[17] The Only Fully Augmented Reality Glasses, http://www.spaceglasses.com/.

[18] M. Kölsch, M. Turk. Fast 2D Hand Tracking with Flocks of Features and Multi-Cue Integration. *IEEE Workshop on Real-Time Vision for Human-Computer Interaction*, 2004.

[19] M. Kölsch, M. Turk. Analysis of Rotational Robustness of Hand Detection with a Viola-Jones Detector. International Conference on Pattern Recognition, volume 3, papes 107-110. 2004.

[20] T.Lee, T.Höllerer. Initializing Markerless Tracking Using a Simple Hand Gesture. *International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007.

[21] M. J. Jones and J. M. Rehg. Statistical color models with application to skin detection, *CVPR*, pp. 1274-1280, 1999.

[22] R.J.K. Jacob. The Use of Eye Movements in Human Computer Interaction Techniques: What You Look at is What You Get, ACM Transactions of Information Systems, Vol.9, No.2, pp.152-169, 1991.

[23] P. Majaranta, A. Aula, K.J. Raiha, Effects of Feedback on Eye Typing with a Short Dwell Time. ETRA, pp.139-146, 2004.