Screen Control System Based on Fingertip Tracking : AirFlick

YANG-KEUN AHN, KWANG-SOON CHOI, YOUNG-CHOONG PARK
Korea Electronics Technology Institute
121-835, 8th Floor, #1599, Sangam-Dong, Mapo-Gu, Seoul
REPUBLIC OF KOREA
ykahn@keti.re.kr

Abstract: - This paper suggests a method of extracting fingertips from 3D information obtained from single-depth cameras in a smart-device environment, and for controlling the screens without directly touching them. This method extracts fingertips from hand images, and controls the screen by reading the fingertips'tracking information. First, we extract hand areas in real time by utilizing depth information, and remove the noise by preprocessing the extracted hand areas to obtain the hand area through labeling. Next, we extract prospective fingertips from the areas and obtain final fingertips after a verification process. Fingertip movement information is indicated as a graph, and movement at a pace exceeding a certain critical level generates flick events. Finally, the key codes obtained from such events are relayed to application contents to control the screen.

Key-Words: - Finger Tracking, Fingertip Detection, Air Touch, Screen Control System, Finger Touch

1 Introduction

It has become very easy to encounter devices equipped with touch screens on a daily basis in modern society, ranging from small smart phones to larger monitors and laptops. It is, however, still difficult to directly use touch screens in some restrictive situations, such as when fingertips are covered with water or food. It is also difficult for users to directly control touch screens as the size of displays grows larger.

In order to address such problems, research has recently been conducted to recognize a user's fingertips, not on the touch screen, but within a spatial dimension, and then to understand certain gestures. Such a spatial touch method, based on vision technology utilizing a high-speed camera attached to a mobile device, was suggested by the lab of Ishikawa Oku at the University of Tokyo, but it was found to require not only high-performance camera equipment to track fingertips, but also a process of initialization [1]. Takeoka et al. suggested extracting fingertips by using the Z-touch method [2], but this method failed to accurately estimate the continuous depth values of fingertips due to the issue of processing speed. Tsukada et al. layered two panels in an effort to enhance accuracy, only to increase the cost of equipment [3].

This paper introduces a screen-controlling system based on the recognition of fingertip tracking information. Chapter 2 suggests the method of extracting a simple and robust group of fingertip candidates, and Chapter 3 discusses the method of extracting accurate fingertips from the candidate group through exception handling and verification. Chapter 4 describes an experimental example of recognition rates of gestures that control the screen without physical contact, and then applies the results to application programs, based on the movement tracking information of the extracted fingertips. Chapter 5 provides the conclusion of this study.

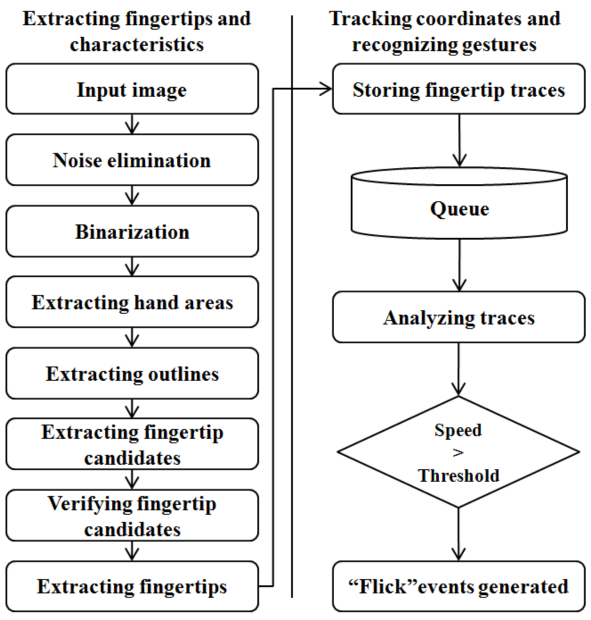

Fig. 1 Flow chart of system control

The system suggested in this study operates in two different parts: the part of extracting fingertips and characteristics and the part of tracking coordinates and recognizing gestures. Fig. 1 provides the flow chart for the suggested system's control.

2 Preprocessing

In the preprocessing stage, hand areas are extracted from input images (Fig. 2, (a)). A median filter is applied in order to remove noise from the input images. Pixel information that exists within a certain area is extracted from the pixels closest in depth images from which noise was eliminated, and then the extracted areas are binarized (Fig. 2, (b)). Later only hand areas are extracted through labeling (Fig. 2, (c)).

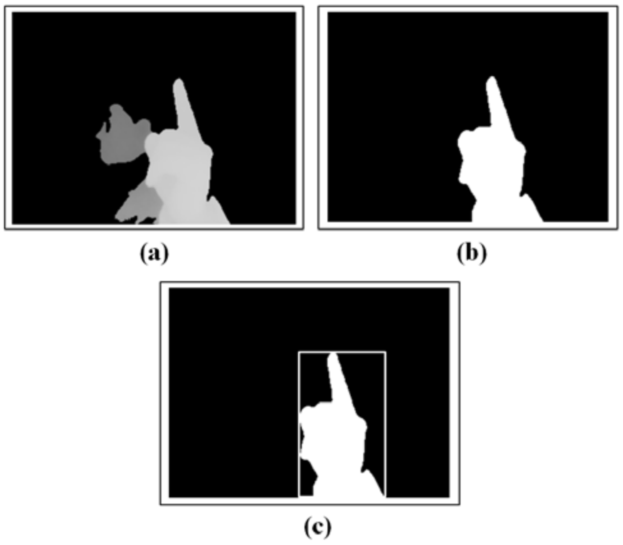

Fig. 2 Process of extracting hand areas (a) Depth image (b) Binarization after extracting close areas (c) Extracting hand areas through labeling

3 Screen Controlling Method through Fingertips Tracking

This chapter describes the method of extracting fingertips from extracted hand-area information, and controlling the screen by means of fingertip trace information.

3.1 Extracting Fingertip Candidates

We explore pixel information from the left of the Y axis to the right side of the extracted hand area. The X and Y coordinates of the first point met becomes the fingertip. This is indicated with the dotted-line arrow in (a) of Fig. 3. But if the fingertip is tilted to the left or right as shown in (b) of Fig. 3, the top pixel is mistakenly recognized instead of the actual fingertip.

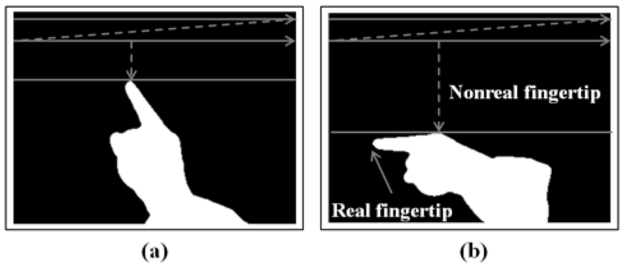

Fig. 3 Process of extracting fingertip candidates (a) Process of exploring fingertips (b) Example of mistaken recognition and actual fingertip

In order to address this problem, the ratio of width to height of hand areas is used, based on the fact that, during a natural gesture, the fingertip is bound to head the left or upper side. In this case, the width ratio of the hand area becomes larger than the height ratio. Fig. 4, (a) shows the standby gesture, with which the height becomes larger than the width in general. In addition, (b) displays the width-height ratio when the fingertip tilts to the left, while (c) shows the ratio when the fingertip tilts to the right. As indicated, the width is larger than the height.

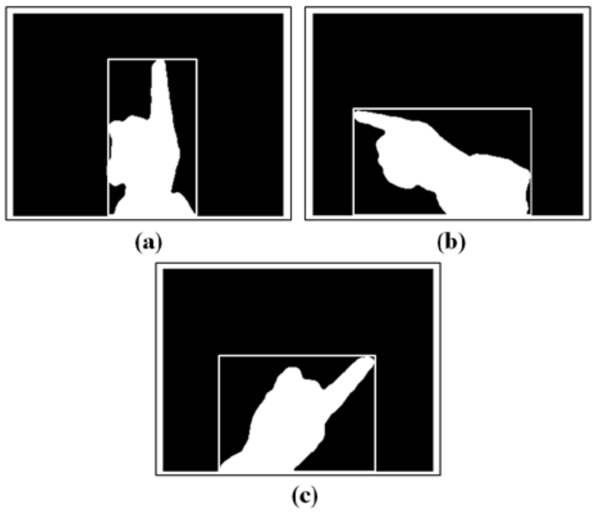

Fig. 4 Width-height ratio of the hand area (a) Standby (b) Tilted to the left (c) Tilted to the right

By means of the width-height ratio suggested, the fingertip is extracted in the following manner. When it is tilted to the left, the X and Y coordinates of the far left pixel are designated as the fingertip;

and when it is tilted to the right, those of the far right pixel are designated as the fingertip. The direction to which the hand is tilted is determined by the location of the Y coordinate in the far left and right pixels. In Fig. 5 (a), the Y coordinate in the left is located higher than the Y coordinate in the right. In contrast, in Fig. 5 (b) the right-side Y coordinate is higher than the Y coordinate in the right. Although it does not belong to the scope of normal gestures, if the left-side fingertip in Fig. 5 (c) is extracted in accordance with the above algorithm, the right-side elbow is mistakenly recognized as a fingertip.

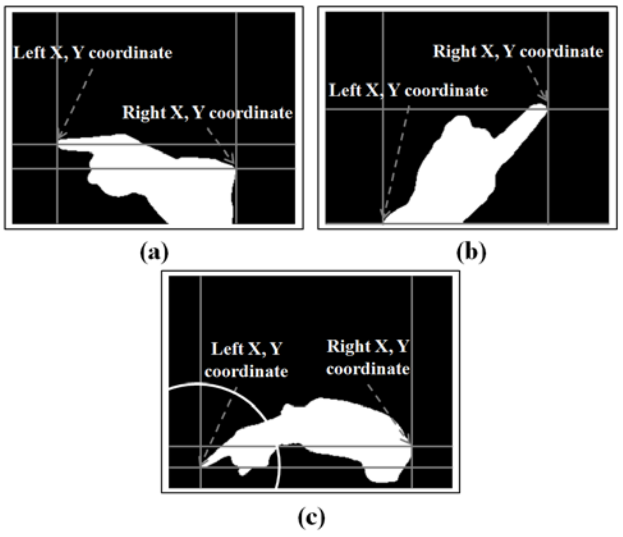

Fig. 5 Process of extracting fingertip candidates when tilted to the left or right (a) Tilted to the left (b) Titled to the right (c) Exception handling of abnormal gestures

In order the resolve the problem, the radius of the fingertip is set, and any movement beyond such boundary is handled as an exception in order to prevent the fingertip's coordinates from getting away. As shown in Fig. 5 (c), since the fingertip's coordinates stay within the circle and the right elbow is outside the circle, the problem of fingertips going off can be resolved.

3.2 Fingertip Verification

The previous chapter showed how to verify fingertip candidates. In this chapter, fingertip candidates are verified to confirm whether they are real fingertips. Finger 6 (a) shows the normal hand image from which the fingertip is to be extracted. Fig. 6 (b) is not the normal hand image and should be handled as an exception, so as not to extract a fingertip from this image.

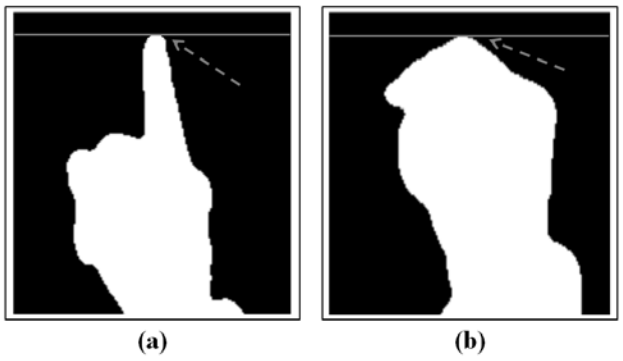

Fig. 6 Fingertips of normal and abnormal poses (a) Fingertip of the normal pose (b) Fingertip of the abnormal pose

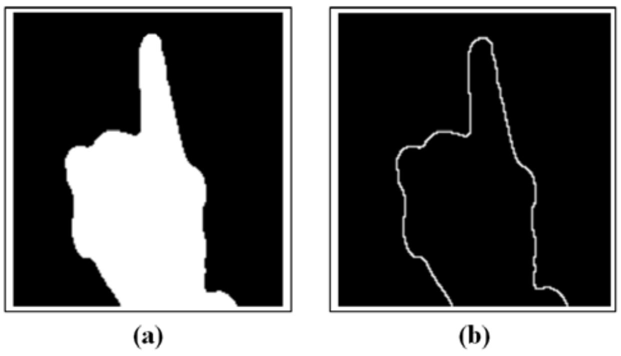

Fig. 7 Converting to binarized and outlined images (a) Binarized image (b) Outlined image

In order to verify if the extracted candidates are real fingertips, a curvature-based algorithm for fingertip-extracting is used. In order to apply this algorithm, the binarized image in Fig. 7 (a) is converted to the outlined image shown in Fig. 7 (b). When straight lines are drawn to the two adjoining pixels within a certain scope from fingertip candidates, it becomes possible to obtain an angle with the candidates. Fig. 8 (a) shows the example of a fingertip angle of the normal hand image, and (b) shows the fingertip angle of the abnormal hand image. As shown in Fig. 8, the angle of the normal hand image is smaller than that of the abnormal image. As a means to ensure higher accuracy, this paper verifies fingertips by utilizing the accumulated total of angle values of a series of continuous pixels as shown in the example of Fig. 8 (c), instead of the angle data of one adjacent pixel.

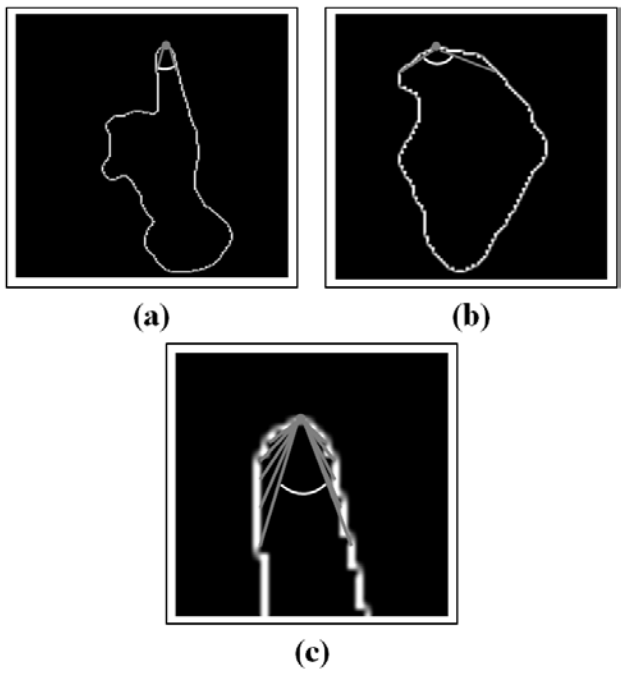
Fig. 8 Process of curvature-based fingertip verification (a) Normal fingertip angle (b) Abnormal fingertip angle (c) Example of accumulated fingertip angles

4 Experiment Results and Application Programs

The experiment was conducted with a computer, equipped with a CPU with Intel(R) Core(TM) i7-2600K 3.4GHz and a 8Gbyte memory. The depth camera DS325[4] was connected to the computer at 60FPS, and the software processing time is provided in Table 1 below. The entire arithmetic operation from image input to gesture recognition took 16ms, which means at least 60 frames per second of real-time operation.

Table 1 Processing Time

Type	Processing time (msec)
Acquiring depth images	5
Extracting hand areas	3
Extracting and verifying fingertips	7
Wait	1
Total	16

In order to verify the arguments proposed in this paper, we tested the left-flick and right-flick gestures involving moving a fingertip fast to either the left or right. In the experiment, a total of 10 users made 1,000 flick gestures, with each user making 50 left and right gestures at random.

Table 2 Flick event recognition rates

User	Left (50)	Right (50)	Total (%)
1	50	50	100
2	50	48	98
3	50	50	100
4	49	49	98
5	50	48	98
6	50	50	100
7	49	50	99
8	50	50	100
9	50	49	99
10	50	49	99
Total	498	493	99.1

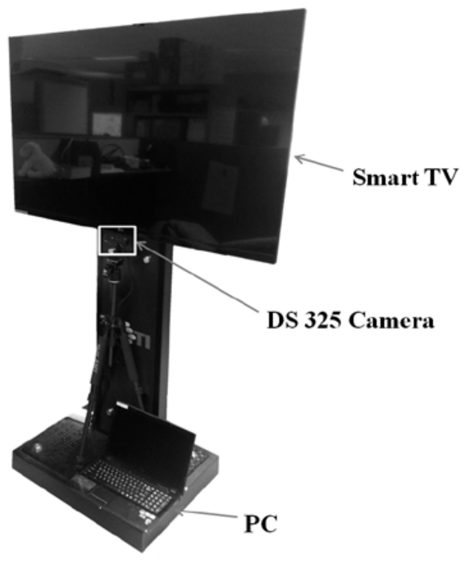

Fig. 9 System hardware organization

The results are provided in Table 2, which shows that left-flick gestures have a slightly higher recognition rate than right flicks. This is related to the higher degree of freedom of leftward gestures. In addition, no left gestures were mistaken as right gestures, or the vice versa, but in some cases users'flick gestures were not recognized at all. This is a result of user gestures failing to reach the critical level, but did not affect the actual screen controlling.

Fig. 9 displays the hardware organization of the AirFlick system. The smart TV shows the screen control, and the DS 325 camera is installed on the lower part of the smart TV to face upward by about 30 degrees to capture a user's gestures. The analysis of gestures is conducted by the computer connected to the camera, and the results are sent to the smart TV through the HDMI cable. Fig. 10 shows how the AirFlick system proposed in this paper is actually applied to application programs.

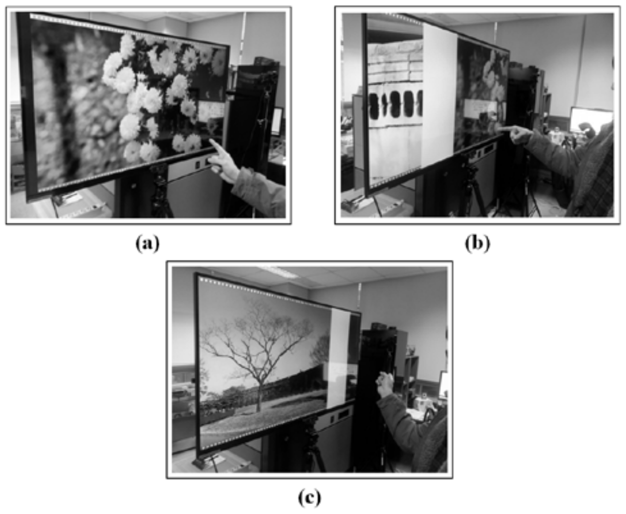

Fig. 10 AirFlick system (a) Standby (b) Flick to the left (c) Flick to the right

5 Conclusion

This paper suggested a method of extracting fingertips from 3D information obtained from single-depth cameras in a smart-device environment, and for controlling screens without directly touching them. By means of the preprocessing technique proposed in this paper, we extracted hand areas in a robust manner before extracting a simple and robust group of fingertip candidates and verifying fingertips based on curvature. Using information on the extracted fingertips, we were able to control screens without physically contacting them and observed at least 99% of recognition rate for gestures of finger licking to the left and right.

References:

[1] Y. Hirobe, T.Niikura, Y. Watanabe, T. Komuro, M. Ishikawa, “Vision-based Input Interface for Mobile Devices with High-speed Fingertip Tracking,” Adj. Proc. ACM UIST 2009, pp. 7-8.

[2] Y. Takeoka et al., “Z-touch: an infrastructure for 3d gesture interaction in the proximity of tabletop surfaces,” Proceedings of ITS’10, 2010.

[3] Y. Tsukada et al., “Layerd touch panel: the input device with touch layers,” Proceedings of CHI’02, 2002, pp. 584-585.

[4] http://www.softkinetic.com