sensors

Article

Virtual Touch Sensor Using a Depth Camera

Dong-seok Lee and Soon-kak Kwon *

Department of Computer Software Engineering, Dong-eui University, Busan 47340, Korea; ulsan333@gmail.com
* Correspondence: skkwon@deu.ac.kr; Tel.: +82-51-897-1727

Received: 24 December 2018; Accepted: 18 February 2019; Published: 20 February 2019

Abstract: In this paper, a virtual touch sensor using a depth camera is proposed. Touch regions are detected by finding each region that consists of pixels within a certain distance from a touch surface. Touch points are detected by finding a pixel in each touch region whose neighboring pixels are closest to surfaces. A touch path error due to noise in the depth picture is corrected through a filter that introduces a weight parameter in order to respond quickly even with sudden changes. The virtual touch sensor is implemented by using the proposed methods for the touch point detection and the touch path correction. In the virtual touch sensor, a touch surface and pixels of a depth picture can be regarded as a virtual touch panel and virtual touch units, respectively. The virtual touch sensor can be applied to wide fields of touch interfaces. Results of simulations show the implementation of the touch-pen interface using the virtual touch sensor.

Keywords: virtual touch sensor; depth camera; depth picture; touch interface

1. Introduction

Convenient interface methods have been actively researched since the popularization of Internet of Things (IoT) devices. Touch interface is the most successful interface to replace existing interface devices such as keyboard and mouse. Touch interface is easy to learn regardless of gender or age. The interface speed using touch interface is twice as fast as conventional physical interfaces [1].

Currently, capacitive sensing methods [2–7] are the most used touch detection methods for touch interface. Capacitive sensing detects touch by measuring capacitances of electrodes. When an object such as a finger touches a panel that applies capacitive sensing, the amount of current that flows through electrodes mounted on the panel is changed. Capacitive sensing has the highest touch detection accuracy and speed among touch sensing methods. However, the price of a device using capacitive sensing increases as a panel size increases, since the touch interface device should be combined with the panel. Instead of the capacitive sensing method, methods using an infrared sensor [8–10] and an ultrasonic sensor [11–13] can be used to detect touch on a large screen. In the touch detection methods using infrared sensors, infrared emitter sensors attached to the screen emit infrared rays and then infrared detection sensors on the opposite side detect the infrared rays. When touch occurs, a touch object blocks the infrared rays, so the infrared detection sensors cannot detect the infrared rays. Touch detection methods using ultrasonic sensors are similar to the methods using infrared sensors. The touch detection methods using infrared or ultrasonic sensors have a limitation as the shape of the screen has to be flat.

In order to detect touch on a non-flat panel, pictures captured by cameras instead of various sensors can be used. Touch detection methods detect an area of the touch object such as a fingertip in captured pictures for the touch region. The touch point is defined as a pixel that has a highest probability for touching in the touch region. Research by Malik [14] captures pictures from two cameras, then detects a fingertip touching a surface by comparing the pictures. The method used by Sugita [15] detects changes of the color of the fingernail when it touches the surface. However, it is difficult to obtain distance information through color pictures. Therefore, the accuracy of color picture methods is remarkably lower than touch detection methods using sensors. In order to obtain distance information depth pictures, which are captured by a depth camera, can be used. In a depth picture, pixel values are set to distances from a depth camera. Studies on applications of depth pictures have been researched in various fields such as face recognition [16–18], simultaneous localization and mapping [19,20], object tracking [21–25], and people tracking [26–28]. Interface methods using depth pictures mainly use gesture recognition. Ren [29] proposed Finger–Earth Mover's Distance, which is a method to measure differences between the hand shapes in depth pictures by applying Earth Mover's Distance [30], in order to recognize finger gesture. Biswas [31] proposes a gesture recognition method that classifies differences between adjacent frames of a depth video using a support vector machine. Li [32] introduces contour tracing and finger vector calculation to recognize finger gesture. Methods for touch detection using depth pictures have also been studied. Harrison [33] proposes a method where each fingertip is detected by measuring variations of depth values in the vertical or horizontal direction in a depth picture. Touch is detected by measuring differences in depth values between the finger and neighboring areas. Wilson [34] proposes a touch detection method of comparing depth values between each touch object and surfaces. However, these proposed methods are only for touch region detection. Studies for touch point detection in depth pictures are insufficient. It is also necessary to study methods of correcting touch path errors, which are caused by depth picture noise.

In this paper, methods for touch point detection and touch path correction are proposed. Each touch region is detected as a region approaching a certain distance from surfaces, similar to Wilson's work [34]. Touch points are detected with pixels closest to surfaces, but other points may also be detected because of depth picture noise. Instead of selecting the closest point to the surfaces, each touch point is detected by finding a pixel in each touch region whose neighboring pixels are closest to surfaces. In touch point tracking, touch path errors occur due to depth picture noise and motion blur. The touch path errors are corrected using a filter similar to the Kalman filter. In the touch path correction using the Kalman filter, there is the disadvantage of a slow response in case of a sudden change of touch point movement. However, the proposed filter solves this problem by introducing a weight parameter that is related to a difference between predicted and measured positions of a touch point. A touch-pen interface is also implemented by the virtual touch sensor.

Surfaces which the depth camera captures and pixels in pictures are set as a virtual touch panel and virtual touch detection units, respectively. Conventional touch interface devices need a screen coupled with sensing devices, while the virtual touch sensor requires only the depth camera as a sensing device. Therefore, the virtual touch sensor is a screen-independent device.

2. Characteristics of Noise in a Depth Picture

Noise in a depth picture occurs due to measurement errors of the depth camera device. This noise causes degrading performance in object feature detection in a depth picture. Figure 1 shows changes in depth values according to frame flow at a certain pixel. The depth values with errors are continuously measured, but an actual depth value is the most frequent. In addition, an average of the depth values is close to the actual depth value.

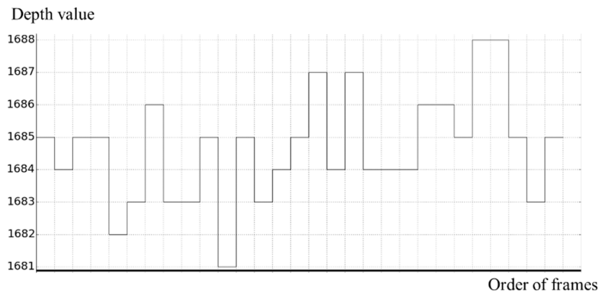


Figure 1. Noises according to frame flow for one depth pixel in depth picture.

Shape distortion of a moving object in depth pictures also occurs. Shape distortion is caused by motion blur [35]. Motion blur occurs when the capturing speed of a camera is slower than the speed of the object movement. Figure 2 shows the shape distortion in object detection due to motion blur.

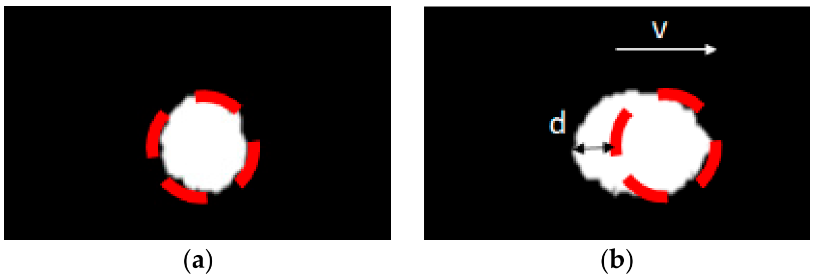


Figure 2. Shape distortion due to motion blur: (**a**) stopping object; (**b**) moving object.

3. Virtual Touch Sensor Using Depth Camera

In this paper, a virtual touch sensor using a depth camera is implemented. Surfaces captured by the depth camera are defined as a virtual panel. Pixels of each captured picture are defined as virtual touch units. Figure 3 shows the virtual touch sensor.

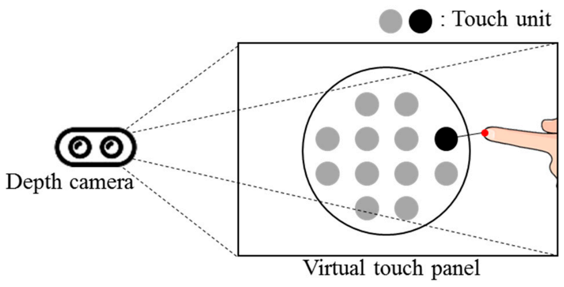


Figure 3. Virtual touch sensor using depth camera.

3.1. Touch Region Detection

In order to detect touch objects using a depth camera, it is necessary to obtain the depth values of a virtual touch panel. Several pictures are used in order to obtain the depth values without noise. The depth values of the virtual panel can be obtained from an average value or a mode value of each pixel in the several pictures. A method using the average values is fast to calculate and simple to implement. However, an incorrect depth value of the virtual touch panel can be obtained if the number of accumulated pictures is not sufficient. A method using the mode values is slow, because a sort for the accumulated depth values in each position is required, however this method can obtain the most accurate depth values of the virtual touch panel. Figure 4 shows methods for obtaining the depth values of the virtual touch panel at a certain pixel using the average and mode values.

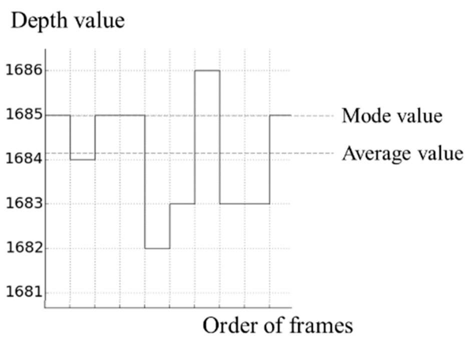
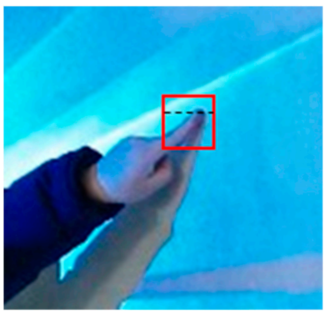


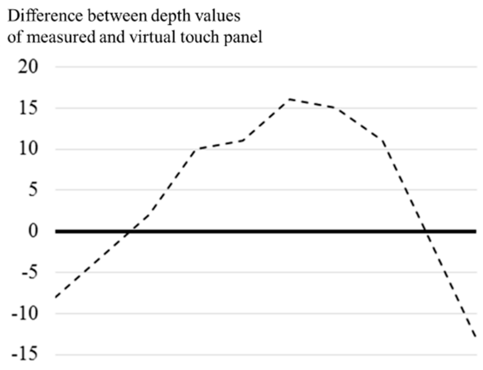
Figure 4. Methods for obtaining depth value of virtual touch panel.

The depth values of the virtual touch panel are compared with depth values of each captured depth picture in order to detect touch regions. Depth values of touch objects are different from the depth values of the virtual touch panel, as shown in Figure 5. The touch objects are detected using differences between the depth values of the touch object and the stored depth values for the virtual touch panel as shown in Figure 6b. To consider depth picture noise, regions composed of pixels which satisfy the following equation is set to the touch regions:

$$o_{x,y} = \begin{cases} 1, & if\ T_l < b_{x,y} - d_{x,y} < T_u \\ 0, & otherwise \end{cases}, \tag{1}$$

where $b_{x,y}$ and $d_{x,y}$ are depth values of the virtual touch panel and the captured picture in position (x, y), respectively, and $o_{x,y}$ is a value in position (x, y) of a binarization picture, which is a picture for touch region detection. A result for touch region detection is shown in Figure 6c. In touch region detection, regions which are not touch regions may also be detected because of noise. To solve this problem, each detected region whose size is less than S_{min} should be regarded as part of the virtual touch panel. S_{min} is determined by considering the resolution of the depth picture. Figure 6d shows the final result of touch region detection by removing noise.


(**a**)



(**b**)

Figure 5. Difference between depth values of measured and virtual touch panel in position of touch: (**a**) original picture; (**b**) difference of depth values from virtual touch panel.

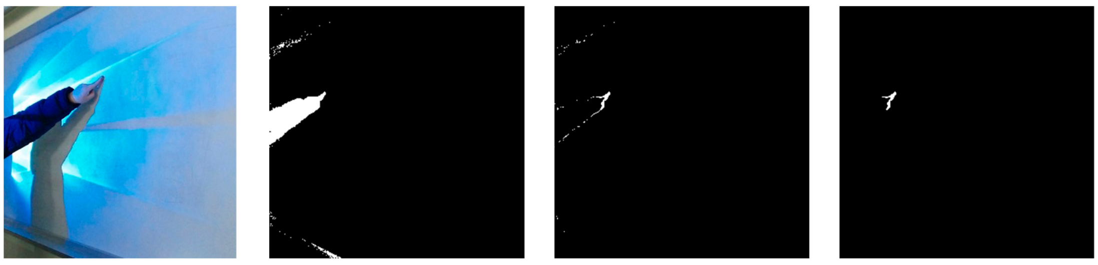

(**a**) (**b**) (**c**) (**d**)

Figure 6. Touch region detection: (**a**) original picture; (**b**) touch object detection; (**c**) touch region detection with noise; (**d**) touch region detection without noise.

3.2. Touch Point Detection

For each touch point a certain pixel where the touch occurs is at the edge of each touch region. To detect touch points, pixels whose distance from the edge of the bounding box of the touch region is within D_s are set to a search region. Pixels satisfying the following equation in the search region are detected:

$$b_{x,y} - d_{x,y} < T_t, \tag{2}$$

where T_t means a threshold for touch point detection. Pixels without accurate touch points may also be detected through Equation (2). In order to detect an accurate touch point, neighboring pixels should also satisfy Equation (2). If pixels satisfy Equation (2) at the position of the colored box in one of the 3×3 block patterns in Figure 7, a center pixel of the block is determined as a touch point.

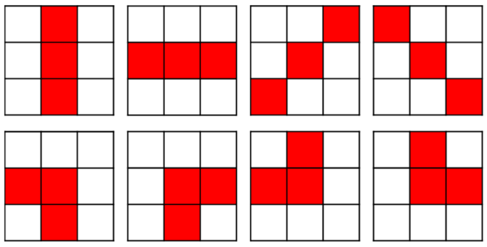
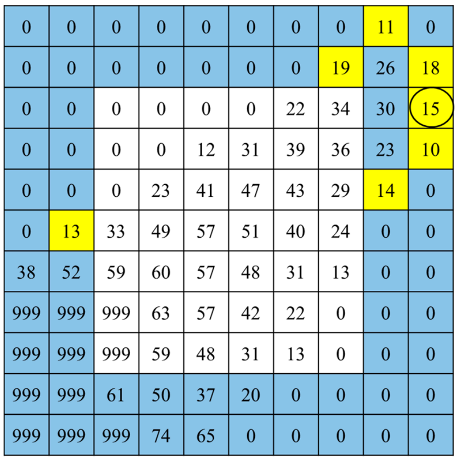


Figure 7. The 3×3 block patterns for touch point detection.

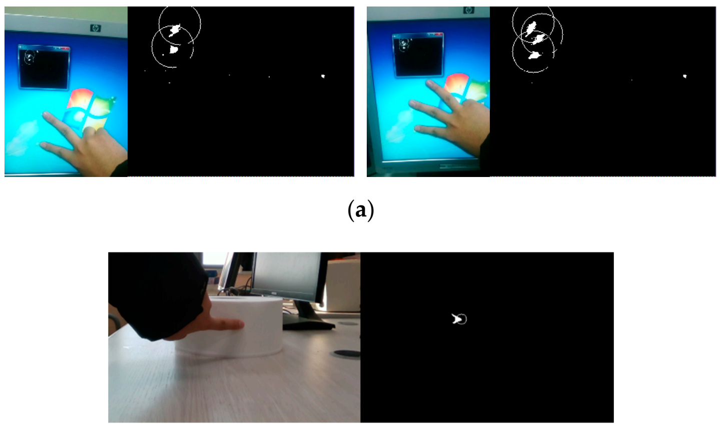
Figure 8 shows depth values in the bounding box of a touch region in Figure 6. Colored boxes are search areas and yellow boxes are pixels satisfying Equation (2) when D_s and T_t are set to 2 and 20, respectively. A circled pixel and neighboring pixels in a vertical direction satisfy Equation (2), so that this pixel is detected as a touch point.



0	0	0	0	0	0	0	0	11	0
0	0	0	0	0	0	0	19	26	18
0	0	0	0	0	0	22	34	30	15
0	0	0	0	12	31	39	36	23	10
0	0	0	23	41	47	43	29	14	0
0	13	33	49	57	51	40	24	0	0
38	52	59	60	57	48	31	13	0	0
999	999	999	63	57	42	22	0	0	0
999	999	999	59	48	31	13	0	0	0
999	999	61	50	37	20	0	0	0	0
999	999	999	74	65	0	0	0	0	0

Figure 8. Touch point detection for Figure 6.

Figure 9 shows the implementation of a virtual touch sensor through the proposed touch point detection. The virtual touch sensor can detect touch even if a virtual touch panel is a curved surface.



(**a**)

(**b**)

Figure 9. Touch point detection: (**a**) for flat surface; (**b**) for curved surface.

3.3. Touch Path Correction

In tracking touch points detected by the proposed methods, touch path errors occur from noise and motion blur as shown in Figure 10.

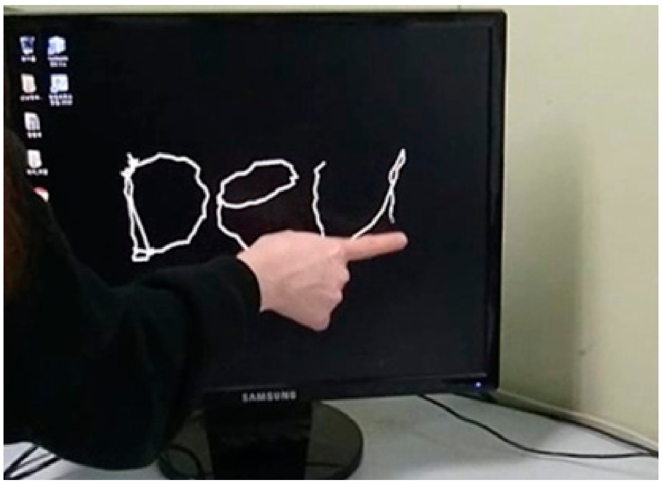


Figure 10. Touch path error.

A proposed filter for correcting touch path errors is similar to the Kalman filter that corrects a measured state through a prediction step, but it introduces a weight parameter in order to respond quickly even with sudden changes. The weight parameter is related to a difference between the predicted and measured positions of a touch point. Figure 11 shows the proposed filter.

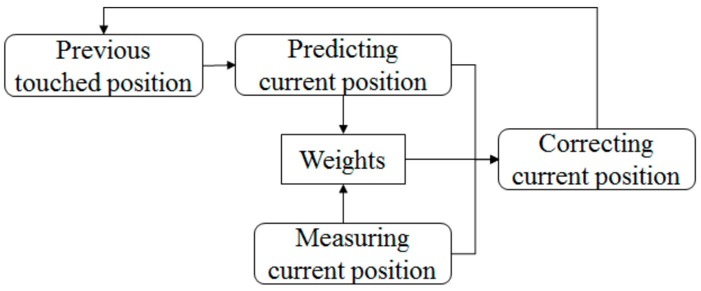

Previous touched position
Predicting current position
Weights
Correcting current position
Measuring current position

Figure 11. Flowchart of proposed filtering.

The position of a touch point in *n*th frame can be predicted by using a position and a speed in a previous frame as follows:

$$\begin{array}{c} \mathbf{p}_p(n) = \mathbf{p}(n-1) + \mathbf{v}(n-1) \\ \mathbf{v}(n) \equiv \mathbf{p}(n) + \mathbf{p}(n-1) \end{array}, \tag{3}$$

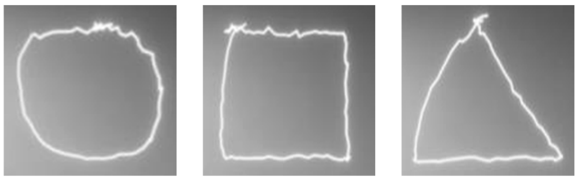
where $\mathbf{p}_p(n)$, $\mathbf{p}(n)$, and $\mathbf{v}(n)$ mean a predicted position, a corrected position, and a speed of a touch point in *n*th frame, respectively. A weight parameter α is calculated by using the difference between a measured and a predicted position as follows:

$$\alpha = \begin{cases} \frac{1}{T_f} e(n), & if\ e(n) \le T_f \\ 1, & otherwise \end{cases}, \tag{4}$$

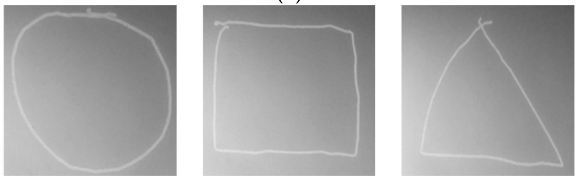
where T_f is a filter threshold and $e(n)$ is a pixel distance between $\mathbf{p}_m(n)$ and $\mathbf{p}_p(n)$. $\mathbf{p}_p(n)$ is completely ignored when $e(n)$ is greater than T_f. $\mathbf{p}(n)$ is calculated as follows:

$$\mathbf{p}(n) = \alpha \mathbf{p}_m(n) + (1-\alpha)\mathbf{p}_p(n). \tag{5}$$

Figure 12 shows a touch path correction using the proposed filtering.



(**a**)



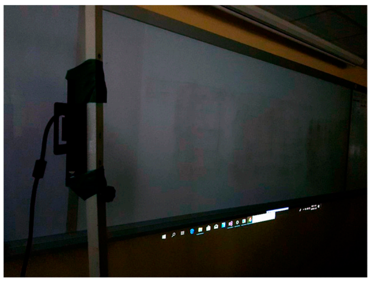
(**b**)

Figure 12. Touch path correction: (**a**) before touch path correction; (**b**) after touch path correction.

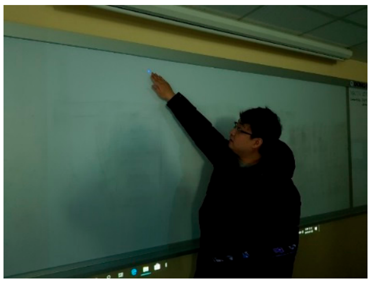
3.4. Implementation of Touch-Pen Interface

In a touch interface using captured pictures, the coordinate system in a captured picture is different from the screen coordinate system. Therefore, it is necessary to match the coordinates of captured pictures with screen coordinates. Homography transformation can match coordinates between the screen and the picture. Four pairs consisting of matching coordinates in the screen and in the captured picture are required for homography transformation. Each pair is obtained by displaying each dot on the screen and touching it. A touch interface according to the touch position can be implemented after matching the coordinates of the screen and the picture.

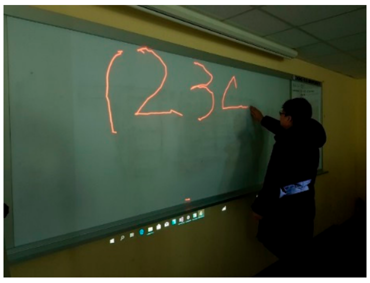
A touch-pen interface that draws lines following touch paths is implemented using the virtual touch sensor. Scenarios of the touch-pen interface are as follows: (1) A depth camera is placed in a position where it can capture the whole of the virtual touch panel; (2) four points are displayed sequentially on a surface of the virtual touch panel and a user touches each displayed point; (3) when the user drags the virtual touch panel, a line is drawn along dragged paths. Figure 13 shows a sequence of the touch-pen interface scenarios.



(a)



(b)



(c)

Figure 13. Implementation of touch-pen using virtual touch sensor: (**a**) placing depth camera; (**b**) calibration of virtual touch sensor; (**c**) implementation of touch-pen.

3.5. Limitations

The virtual touch sensor has an advantage that the size or surface type of the virtual touch panel does not affect touch detection. However, the virtual touch sensor has limitations as touch detection performance is dependent on specifications of the depth camera and touch detection on a dynamic virtual touch panel is difficult. This limitation is caused by using pictures captured by a camera. A detection interval for touch cannot exceed the frame rate of a depth camera. If the frame rate is 30 Hz, a detection interval cannot be less than about 33 ms. The precision for touch detection depends on the resolution of the depth camera.

The proposed virtual touch sensor can only detect touch if a virtual touch panel is not changed, because touch detection uses the pre-stored depth values of the virtual touch panel. A step that modifies the pre-stored depth values if a virtual touch panel is changed can be introduced to detect touch on a dynamic virtual touch panel. However, updating changes to the stored depth values after obtaining the depth values of a changed virtual touch panel take a lot of time. In order to detect touch in a dynamic touch panel in real time, a method for the classification of a virtual touch panel and virtual touch objects without previously stored depth values needs to be studied further.

4. Simulation Results

In order to measure the performance of the virtual touch sensor, we used Xtion Pro Live, which is manufactured by ASUS from Taiwan, as a depth camera. A resolution of depth pictures is specified as 320×240. For the virtual touch panel we used a screen with a width and height of 2.2 m and 1.7 m, respectively. We set the default angle and distance between the camera and the virtual touch panel as 30° and 1.5 m, respectively.

We measured the accuracy of the proposed touch-pen interface. A fixed position is touched 20 times. Each position detected as a touch point is displayed in the virtual touch panel as a dot. Each touch position error is measured as a distance between the touched and displayed position. S_{min}, T_l, T_u, D_s, and T_t, which are parameters of the virtual touch sensor, are set as 30, 3, 30, 4, and 10, respectively.

The average touch point detection time using the virtual touch sensor is measured as 20 ms. However, the actual touch detection interval cannot exceed the frame rate of the camera. Therefore, the detection time of the virtual touch sensor is slower than conventional touch detection devices using capacitive sensing. These have a touch point detection time of 10 ms in the case of self-capacitance or 6 ms in the case of mutual-capacitance [7].

In obtaining depth values of the virtual touch panel, touch position errors according to methods and a number of accumulated pictures is shown in Figure 14. Touch position errors when obtaining depth values using average values are large when the number of accumulated pictures is less than about 200. An improvement of touch accuracy doesn‘t appear if the number of accumulated pictures are more than about 300.

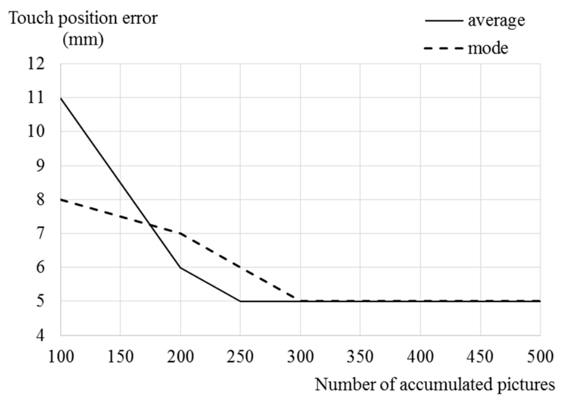


Figure 14. Touch position error according to obtaining methods of depth values of virtual touch panel and the number of accumulated pictures.

Table 1 shows success rates of touch region detection and position errors of touch points according to T_l. Some touch regions are not detected due to depth picture noise when T_l is less than 3. On the other hand, touch position errors are increased as T_l is more than 4 because some parts of the object regions are detected as the virtual touch panel.

Table 1. Success rate and position errors according to T_l in touch region detection.

T_l	Success Rate of Touch Region Detection (%)	Position Error of Touch Point(mm)
1	0	-
2	75	8
3	100	5
4	100	6
5	100	7

Table 2 shows position errors of touch points according to T_t. The position errors increase linearly as T_t increases. The number of detected pixels increases as T_t increases, so it is hard to detect an accurate touch point.

Table 2. Position errors according to T_t in touch region detection.

T_t	Position Error of Touch Point (mm)
1	3
2	5
3	7
4	9
5	11

Table 3 shows position errors according to an angle between the virtual touch panel and the depth camera. Touch point detection is most accurate when the angle is set to 30°. It is difficult to detect an accurate touch point when the angle is too small, because an area of the virtual touch panel occupied in a captured picture becomes too small. In contrast, errors in touch region detection increase when the angle is large. The accuracy of touch point detection increases as the distance is closer.

Table 3. Position errors according to distance and angle between camera and screen.

Angle(°)	Distance between Camera and Screen (m)				
	1.5	2	2.5	3	3.5
10	16	17	18	18	19
15	12	13	13	14	14
20	8	8	10	12	12
25	5	6	9	10	10
30	5	6	6	7	7
35	6	6	6	7	7
40	7	7	8	8	9
45	8	9	9	10	13
50	9	11	13	15	17
55	10	12	13	16	18

In order to measure the performance of correcting touch paths by using the proposed touch path correction, the proposed correction method is compared with the Kalman filter and a low-pass filter. The low-pass filter corrects touch paths through the following linear equation:

$$\mathbf{p}(n) = (\mathbf{p}_m(n) + \mathbf{p}(n-1))/2. \quad (6)$$

Equation (6) means that this filter corrects a position as an average between a measured position in a current frame and a corrected position in a previous frame. Figure 15 shows the responses of filters to touch point movements. The low-pass filter removes part of the changes in touch point movement unconditionally. The Kalman filter is more responsive to changes than the low-pass filter, but it still tends to eliminate large changes. In contrast, the proposed filter responds quickly even if large change occurs.

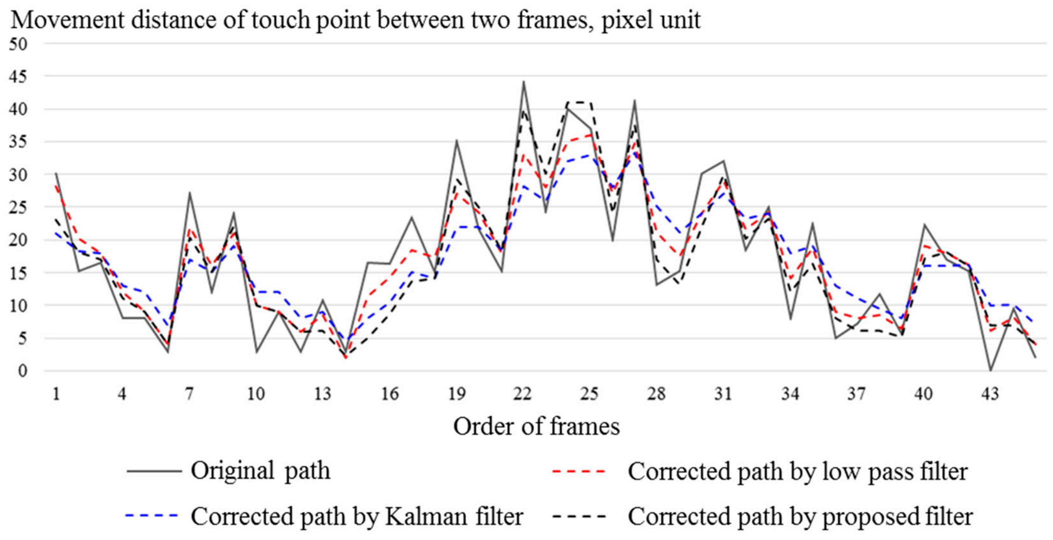


Figure 15. Responses of filters to touch movements.

We drew a shape as shown in Figure 16a, consisting of repetitive vertical and horizontal lines by using the touch-pen interface. Then we measured angles θ_1, θ_2 as shown in Figure 16b, between horizontal and vertical axes from a drawn line, respectively. A smaller angle is selected from two angles. A selected angle can be considered as an amount of touch path error. The angle errors converge to 0 as the path correction is accurate.

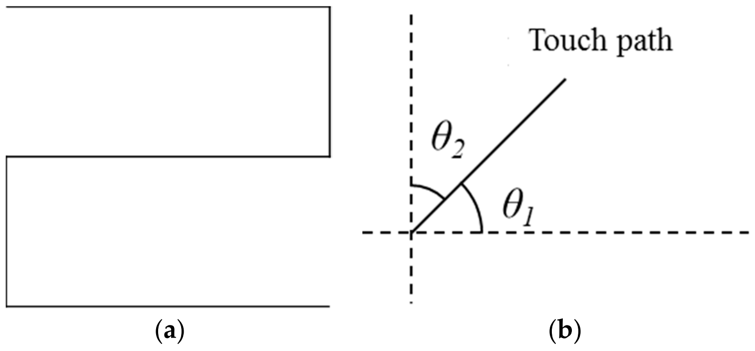


Figure 16. Measuring performance of proposed touch path filtering: (**a**) shape for measuring performance of path correction; (**b**) method for measuring touch path error.

Average angle errors of the original path, corrected paths by low-pass filter, Kalman filter, and proposed filter are 12.688°, 10.792°, 6.937°, and 6.813°, respectively. Figure 17 shows the results of correcting the touch path.

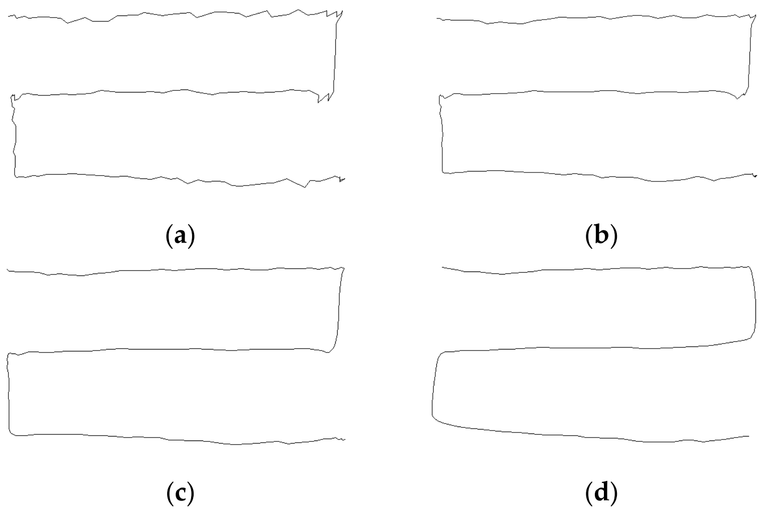


Figure 17. Path correction results according to filtering methods: (**a**) original touch paths; (**b**) correcting by low-pass filter; (**c**) correcting by Kalman filter; (**d**) correcting by proposed method.

Figures 18 and 19 show average angle errors according to T_f. A correction effect is more clear as T_f increases. However, more shape distortion occurs at positions where movements of touch points suddenly change as T_f increases as shown in Figure 19.

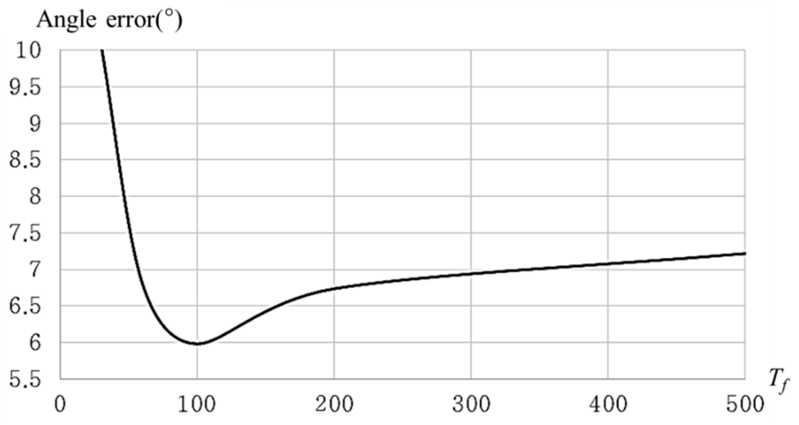


Figure 18. Angle error according to T_f.

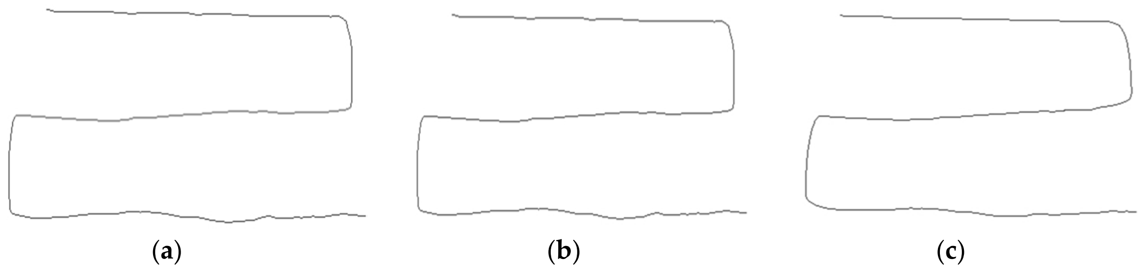


Figure 19. Shape distortion according to T_f: (**a**) T_f = 30; (**b**) T_f = 100; (**c**) T_f = 200.

5. Conclusions

In this paper, a virtual touch sensor was implemented by using a depth camera. The virtual touch sensor showed the most accuracy when T_l, T_u, the distance between a virtual touch panel and the depth camera and an angle were set as 3, 30, 1.5 m, and 30°, respectively. A touch-pen interface using the virtual touch sensor was also implemented. The virtual touch sensor is expected to solve the problem of applying touch interfaces to a large display, which is a disadvantage of existing physical touch methods. Conventional touch detection sensors can only use a flat surface as a touch panel, while the virtual touch sensor can also use a curved surface. The virtual touch sensor has the advantage that it is cheaper than conventional physical touch sensors. We expect the virtual touch sensor to be applicable to other fields such as a motion recognition.

Author Contributions: Conceptualization, D.-s.L. and S.-k.K.; software, D.-s.L.; writing—original draft preparation, D.-s.L. and S.-k.K.; supervision, S.-k.K.

Funding: This research was supported by The Leading Human Resource Training Program of Regional New Industry through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (No. 2018043621), and supported by the BB21+ Project in 2018.

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

References

1. Kin, K.; Agrawala, M.; DeRose, T. Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. In Proceedings of the Graphics Interface 2009, Kelowna, BC, Canada, 25–27 May 2009; pp. 119–124.
2. Lee, S.K.; Buxton, W.; Smith, K.C. A multi-touch three dimensional touch-sensitive tablet. *ACM SIGCHI Bull.* **1985**, *16*, 21–25. [CrossRef]
3. Walker, G. A review of technologies for sensing contact location on the surface of a display. *J. Soc. Inf. Disp.* **2012**, *20*, 413–440. [CrossRef]
4. Barrett, G.; Omote, R. Projected-capacitive touch technology. *Inf. Disp.* **2010**, *26*, 16–21. [CrossRef]
5. Yang, I.S.; Kwon, O.K. A touch controller using differential sensing method for on-cell capacitive touch screen panel systems. *IEEE Trans. Consum. Electron.* **2011**, *57*, 1027–1032. [CrossRef]
6. Bhalla, M.R.; Bhalla, A.V. Comparative study of various touchscreen technologies. *Int. J. Comput. Appl.* **2010**, *6*, 12–18. [CrossRef]
7. Touch Technology Brief: Projected Capacitive Technology. 3M Company. Available online: http://multimedia.3m.com/mws/media/788463O (accessed on 30 January 2019).
8. Soni, V.; Patel, M.; Narde, R.S. An interactive infrared sensor based multi-touch panel. *Int. J. Sci. Res. Publ.* **2013**, *3*, 610–623.
9. Monnai, Y.; Hasegawa, K.; Fujiwara, M.; Yoshino, K.; Inoue, S.; Shinoda, H. HaptoMime: mid-air haptic interaction with a floating virtual screen. In Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology, Honolulu, HI, USA, 5–8 October 2014; pp. 663–667.
10. Lee, Y.; Omkaram, I.; Park, J.; Kim, H.S.; Kyung, K.U.; Park, W.; Kim, S. Aα-Si:H thin-film phototransistor for a near-infrared touch sensor. *IEEE Electron Device Lett.* **2015**, *36*, 41–43. [CrossRef]
11. Nonaka, H.; Da-te, T. Ultrasonic position measurement and its applications to human interface. *IEEE Trans. Instrum. Meas.* **1995**, *44*, 771–774. [CrossRef]
12. Firouzi, K.; Nikoozadeh, A.; Carver, T.E.; Khuri-Yakub, B.P.T. Lamb wave multitouch ultrasonic touchscreen. *IEEE Trans. Ultrason. Ferroelectr. Freq. Control.* **2016**, *63*, 2174–2186. [CrossRef] [PubMed]
13. Quaegebeur, N.; Masson, P.; Beaudet, N.; Sarret, P. Touchscreen surface based on interaction of ultrasonic guided waves with a contact impedance. *IEEE Sens. J.* **2016**, *16*, 3564–3571. [CrossRef]
14. Malik, S.; Laszlo, J. Visual touchpad: A two-handed gestural input device. In Proceedings of the 6th International Conference on Multimodal Interfaces, State College, PA, USA, 13–15 October 2004; pp. 289–296.
15. Sugita, N.; Iwai, D.; Sato, K. Touch sensing by image analysis of fingernail. In Proceedings of the SICE Annual Conference, Tokyo, Japan, 20–22 August 2008; pp. 1520–1525.
16. Fanelli, G.; Dantone, M.; Van Gool, L. Real time 3D face alignment with random forests-based active appearance models. In Proceedings of the IEEE International Conference and Workshops on Automatic Face and Gesture Recognition, Shanghai, China, 22–26 April 2013; pp. 1–8.
17. Dantone, M.; Gall, J.; Fanelli, G.; Van Gool, L. Real-time facial feature detection using conditional regression forests. In Proceedings of the IEEE International Conference and Workshops on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 2578–2585.

18. Min, R.; Kose, N.; Dugelay, J.L. KinectFaceDB: A Kinect Database for Face Recognition Systems. *IEEE Trans. Man Cybern. Syst.* **2014**, *44*, 1534–1548. [CrossRef]
19. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A benchmark for the evaluation of RGB-D SLAM systems. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura, Portugal, 7–12 October 2012; pp. 573–580.
20. Pomerleau, F.; Magnenat, S.; Colas, F.; Liu, M.; Siegwart, R. Tracking a depth camera: Parameter exploration for fast ICP. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; pp. 3824–3829.
21. Siddiqui, M.; Medioni, G. Human pose estimation from a single view point, real-time range sensor. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, San Francisco, CA, USA, 13–18 June 2010; pp. 1–8.
22. Munoz-Salinas, R.; Medina-Carnicer, R.; Madrid-Cuevas, F.J.; Carmona-Poyato, A. Depth silhouettes for gesture recognition. *Pattern Recognit. Lett.* **2008**, *29*, 319–329. [CrossRef]
23. Suryanarayan, P.; Subramanian, A.; Mandalapu, D. Dynamic hand pose recognition using depth data. In Proceedings of the 20th International Conference on Pattern Recognition, Istanbul, Turkey, 23–26 August 2010; pp. 3105–3108.
24. Preis, J.; Kessel, M.; Werner, M.; Linnhoff-Popien, C.L. Gait recognition with Kinect. In Proceedings of the Workshop on Kinect in Pervasive Computing, Newcastle, UK, 18 June 2012; pp. P1–P4.
25. Song, S.; Xiao, J. Tracking revisited using RGBD camera: Unified benchmark and baselines. In Proceedings of the IEEE International Conference on Computer Vision, Sydney, Australia, 1–8 December 2013; pp. 233–240.
26. Sung, J.; Ponce, C.; Selman, B.; Saxena, A. Human activity detection from RGBD images. *Plan Act. Intent Recognit.* **2011**, *64*, 47–55.
27. Spinello, L.; Arras, K.O. People detection in RGB-D data. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; pp. 3838–3843.
28. Luber, M.; Spinello, L.; Arras, K.O. People tracking in RGB-D data with on-line boosted target models. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; pp. 3844–3849.
29. Ren, Z.; Yuan, J.; Meng, J.; Zhang, Z. Robust part-based hand gesture recognition using kinect sensor. *IEEE Trans. Multimedia* **2013**, *15*, 1110–1120. [CrossRef]
30. Rubner, Y.; Tomasi, C.; Guibas, L.J. The earth mover's distance as a metric for image retrieval. *Int. J. Comput. Vision* **2000**, *40*, 99–121. [CrossRef]
31. Biswas, K.K.; Basu, S.K. Gesture recognition using microsoft kinect®. In Proceedings of the 5th International Conference on Automation, Robotics and Applications, Wellington, New Zealand, 6–8 December 2011; pp. 100–103.
32. Li, Y. Hand gesture recognition using Kinect. In Proceedings of the 2012 IEEE International Conference on Computer Science and Automation Engineering, Beijing, China, 22–24 June 2012; pp. 196–199.
33. Harrison, C.; Benko, H.; Wilson, A.D. OmniTouch: wearable multitouch interaction everywhere. In Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, Santa Barbara, CA, USA, 16–19 October 2011; pp. 441–450.
34. Wilson, A.D. Using a depth camera as a touch sensor. In Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces, Saarbrücken, Germany, 7–10 November 2010; pp. 69–72.
35. Nayar, S.K.; Ben-Ezra, M. Motion-based motion deblurring. *IEEE Trans. Pattern Anal. Mach. Intell.* **2004**, *26*, 689–698. [CrossRef] [PubMed]