KinectTouch: Accuracy Test for a Very Low-Cost 2.5D Multitouch Tracking System

Andreas Dippon *Gudrun Klinker*
Fachgebiet Augmented Reality
Technische Universität München
Boltzmannstr. 3, D-85748 Garching, Germany
{dippona, klinker}@in.tum.de

ABSTRACT

We present a simple solution for a touch detection system on any display at a very low cost. By using a Microsoft Kinect, we can detect fingers and objects on and above a display. We conducted a user study in order to evaluate the accuracy of this system compared to the accuracy of a capacitive touch monitor. The results show, that the system can't compete with an integrated system yet, but works well enough to be used as a touch detection system for large displays.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors

Keywords: Multitouch, Depth-sensing cameras, Object tracking, Interactive Surface.

INTRODUCTION

While multitouch is everywhere nowadays, touch-enabling instrumented hardware is still very expensive, especially for large displays. By using a very low-cost depth camera like the Microsoft Kinect [1], it is possible to turn any display into a touch screen without instrumentation. Through the depth information, fingers, hands, objects and gestures can be detected on and above a display. One drawback is that the moment of the contact with the display can't be determined as precisely as with other touch technologies. Another important aspect which we focus on in this paper is the accuracy of the detected touches compared to an embedded technique such as capacitive touch sensing.

First we will review related work, before describing the implemented tracking technique. Then the user evaluation and its results will be described. Afterwards we present some ideas for improvements to the system, followed by a short discussion.

ITS 2011, November 13-16, Kobe, Japan. Copyright 2011 ACM 978-1-4503-0871-7/11/11....$10.00.

RELATED WORK

Being one of the first who introduced the idea to integrate a touch detection and displaying system into a work desk [12], Wellner paved the way for future ideas in this field. Then it turned quiet around large interactive touchscreens. A decade later Dietz presented the DiamondTouch [3], a large multitouch screen which could distinguish different users. While this was already a very intriguing system, it was very expensive to get one and therefore not affordable as an office desk. The breakthrough came when Han presented his realization of a multitouch display using Frustrated Total Internal Reflection [5]. While this technique was available at a very low cost, the needed instrumentation of the display was quite complicated and time consuming. An easier technique, that we also use in this work was presented by Wilson [13]. In his work he explains the basics of a technique to sense fingers on any surface using a depth camera. While he already suggested to use it to track fingers on a display he didn't execute comparisons to other available touch technologies.

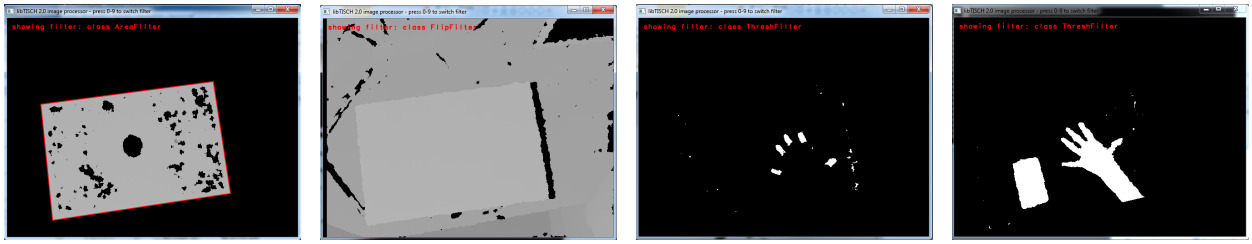
Additionally to touch detection, interaction above the surface comes more and more into focus. Recent contributions in this field came from Takeoka et al. [10] and Moeller et al. [8]. They both suggest using several layers of infrared LEDs or laser planes to detect gestures above a display. While the systems work very well, again they have to be integrated on the front of the display. Furthermore the needed hardware increases with the size of the display and the tracking volume.

FINGER AND OBJECT TRACKING

In order to detect touches on a display we use the depth image of a Microsoft Kinect, with a technique similar to [13]. To this end we place a Kinect above a display, facing downwards (see fig.3), save the depth values of the background and subtract them from the camera image in each frame to get the distance between hands, objects, fingers, etc. and the display. By defining an area of interest and thresholding the distance values of the subtracted image, we can track everything within a virtual cuboid above the display. We used the libTISCH library by Echtler[4] for our implementation of this algorithm, which is detailed in the following part.

Implementation

The image of the camera is encoded as a 11-Bit greyscale image where the values represent the depth data. The first filter we apply to the image is an area filter where the area of interest can be selected (see fig.1(a)). This can be done on the



(a) Areafilter (b) Background Image (c) Threshold Image: Fingers (d) Threshold Image: Objects

Figure 1: Camera Images

fly by drawing corner points for a polygon on the camera image. It is also possible to define several areas of interest. The next step is the application of a background filter. Because the camera does not work properly on a reflective display (see fig.1(a)), the display has to be covered by a thin non-reflective material (e.g. paper) during the initialization of the background image to get a reliable depth value for the screen surface (see fig.1(b)). The cover can be removed after the initialization. Most of the noise occurring due to the reflective display is filtered out by subsequent filters. Afterwards a threshold filter is applied, which discards all data outside a minimum and a maximum distance to the background values and saves the remaining data as white areas in a B/W image (see fig.1(c)). This filter simulates taking a thin slice above the display. A more detailed description can be found in [13]. After applying a lowpass filter to get rid of remaining noise, the blobs can be extracted in the final image. Object and shadow recognition can be done by a second filter chain whereby the minimum and maximum values of the threshold filter have to be adjusted (see hand and mobile phone in fig.1(d)). When two filter chains are used to detect fingers and objects, the detected fingers are automatically associated with according hands. Therefore fingers of different hands can be distinguished.

(a) Yaw: finger parallel to the vertical edge of the display (b) Pitch: finger between 15 and 60 degrees

Figure 2: Yaw and Pitch

ACCURACY STUDY

We conducted a user study in order to get an idea of the accuracy of this touch method compared to capacitive touch recognition. In this study, the positions of the touches generated by the Kinect system on a display were compared to the touch points generated by the touch display itself. The setup consisted of a 3M Multi-Touch Display M2256PW [2]

Figure 3: Setup: Kinect above a multitouch display

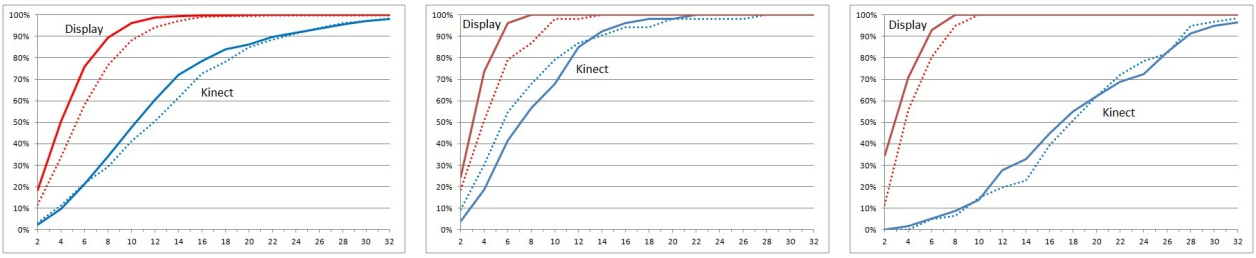
and a Microsoft Kinect. The display was used in a horizontal position with the Kinect mounted 0.75 meters above the display, facing downwards (see fig.3). During the tests, touches of the display and touches through the Kinect were recorded simultaneously. The center of each blob in the Kinect image was used as the touch position for the Kinect touch system.

Calibration

First the users had to calibrate both systems at the same time, by pressing four red dots in the corners (shifted 50 pixel inwards in x- and y-direction) of the display for a few seconds each. With this calibration method we ensure, that there is no default offset between the two touch inputs.

Test

Afterwards the users had to touch small white crosses (10*10 pixels) which were shown subsequently at random positions on the screen. The users were instructed to use the index fin-



(a) Average accuracy over all users (b) User with best accuracy (c) User with worst accuracy

Figure 4: Accuracy diagrams

ger of their dominant hand during the whole evaluation. The yaw of the finger always had to be 0 degrees, so that the finger was parallel to the vertical edge of the screen (see fig.2(a)). They also had to keep their index finger in a pitch between 15 and 60 degrees when touching the display (see fig.2(b)). If a higher pitch is used, the hand covers the index finger in the depth image. On touching a white cross, which is triggered by the touch event of the display, the current touch coordinates of both systems were recorded as well as the position of the shown cross. Afterwards the white cross was replaced by a new one at a different random position. The users had to do two rounds of more than 50 touches each. In the first round they were instructed to try to be as accurate as they can. In the second round they should try to be as fast as they can, in order to get data which better resembles the daily usage of the interface. Hereby they didn't have to keep the yaw at 0 degrees, but still using the index finger of their dominant hand and a pitch between 15 and 60 degrees.

Results

Data was recorded from 13 computer scientists (4 female, 9 male), aged between 23 and 53. We calculated the mean offset and the spread (around the mean offset) of the Kinect touches and the display touches for each user. In the first round, which focused on accuracy, the average spread was 5.91mm for the Kinect touches and 2.20mm for the display touches. In the second round, which focused on speed, the average spread was 6.36mm for the Kinect touches and 2.99mm for the display touches. Figure 5 shows a comparison of the spread of all users. The distribution of the touches around the white crosses for each user was also recorded during the test. By using this distribution we calculated the accuracy of hitting a circular virtual button which is centered on the mean offset of each user. The accuracy is therefore only dependent on the spread, which compensates offsets of different users. Figure 4(a) shows the diagram of the average accuracy over all users. The red lines show the accuracy for the display touches, the blue lines for the Kinect touches. The continuous lines show the first round and the dotted lines the second round of touches. In order to be able to compare the results to related work[11, 6], we calculated the minimum button sizes for 95% reliability. A virtual button which is hit by 95% of the touches measured by the Kinect would need to have a diameter of 28mm (28mm fast round). For the display touches the diameter would need to be 10mm (14mm fast round). Figure 4(b) shows the result of the best participant. For this participant, a button with a diameter of 16mm

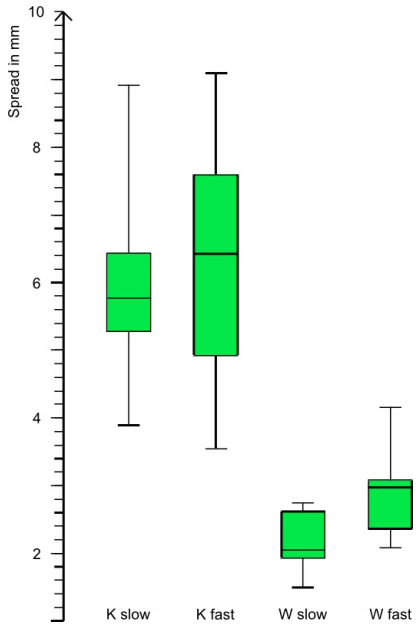
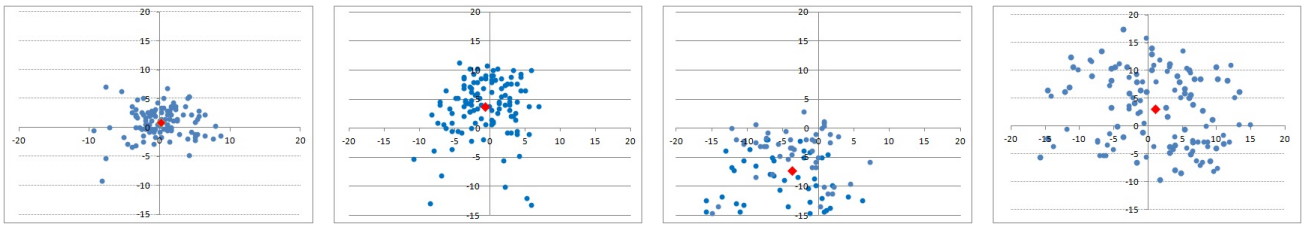


Figure 5: Spread of all users

(20mm fast round) for the Kinect touches and 6mm (10mm fast round) for the display touches would be sufficient. The worst result can be seen in fig.4(c). The main reason for this bad result is probably a bad calibration of the Kinect touches, as the display touches showed an average result. The distributions for the Kinect touches (slow and fast) of four users can be seen in fig.6. The axis are in millimeters and the origin resembles the targets. The blue dots show the position of the Kinect touch points and the red diamond shows the mean offset of each user.

IMPROVEMENTS

Regarding the previously mentioned bad result, the first useful improvement would be to prolong the calibration phase in order to reduce calibration errors. This would decrease the error rates of the Kinect touches as well as the display touches. Another improvement for the position of the Kinect touches could be achieved by using a finger tracking algorithm on the object/shadow sensing image (see fig.1(d)). With this tracking method, a better approximation of the touch position can be utilized as pointed out in [7]. Further enhancement of the Kinect touch positions can be achieved by using information about the yaw and the pitch of the fin-



(a) Best result (b) Average result with low offset (c) Average result with big offset (d) Worst result

Figure 6: Distribution diagrams

ger as supposed in [9]. Additional improvement could be done by using information about the position of the hand and the user, which can be estimated by the users arm above the surface.

DISCUSSION

One important issue concerns our calibration method, which was designed to compare the accuracy of users with the two systems. By calibrating the display and the Kinect for each user, we wanted to make sure, that different finger shapes and touch techniques were taken into account. This was done, because we wanted to get results of how accurate a user can be with the Kinect, compared to a normal touch screen. As this calibration method showed some bad effects on the results, we would recommend to perform a more sophisticated calibration in future studies.

Another important issue is the question of how good the system would perform in daily use. Therefore the second round of tests was performed, where the users had to be as fast as possible and were not required to care about the yaw of their finger. The results showed, that the system is not accurate enough to be used on small (e.g. smartphones) or medium (e.g. tablets) devices. Yet for large surfaces (e.g. tables, large screens) it should be considered as a cheap and acceptable alternative to other techniques, withouth the need of instrumenting the surfaces.

CONCLUSION

In this work we used a Kinect to detect objects and touches on an uninstrumented display in order to compare its accuracy to a capacitive touch display. The comparison showed, that the system can't compete with an integrated system yet. Nevertheless we believe, that this alternative to expensive technologies works well enough to be used as a touch detection system for monitors, especially for large screens, as the costs for the tracking system won't increase. We also suggested several techniques, which can be used to improve the position data of tracked fingers.

ACKNOWLEDGMENTS

We thank Florian Echtler and Norbert Wiedermann for their contributions to this work.

REFERENCES

1. `http://www.xbox.com/kinect`. last accessed June 27, 2011.
2. `http://solutions.3m.com/wps/portal/3M/en_US/TouchSystems/TouchScreen/Solutions/MultiTouch/M2256PW/`. last accessed June 27, 2011.
3. P. Dietz and D. Leigh. DiamondTouch: a multi-user touch technology. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 219–226, 2001.
4. F. Echtler and G. Klinker. A multitouch software architecture. In *Proceedings of the 5th Nordic Conference on Human-Computer Interaction: Using Bridges (NordiCHI '08)*, pages 463–466, October 2008.
5. Jefferson Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118, 2005.
6. Christian Holz and Patrick Baudisch. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In *Proceedings of the 28th international conference on Human factors in computing systems*, CHI '10, pages 581–590, New York, NY, USA, 2010. ACM.
7. Christian Holz and Patrick Baudisch. Understanding touch. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 2501–2510, New York, NY, USA, 2011. ACM.
8. Jon Moeller, Andruid Kerne, and Sashikanth Damaraju. Zerotouch: a zero-thickness optical multi-touch force field. In *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems*, CHI EA '11, pages 1165–1170, New York, NY, USA, 2011. ACM.
9. Simon Rogers, John Williamson, Craig Stewart, and Roderick Murray-Smith. Anglepose: robust, precise capacitive touch tracking via 3d orientation estimation. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, pages 2575–2584, New York, NY, USA, 2011. ACM.
10. Yoshiki Takeoka, Takashi Miyaki, and Jun Rekimoto. Z-touch: an infrastructure for 3d gesture interaction in the proximity of tabletop surfaces. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 91–94, New York, NY, USA, 2010. ACM.
11. Feng Wang and Xiangshi Ren. Empirical evaluation for finger input properties in multi-touch interaction. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 1063–1072, New York, NY, USA, 2009. ACM.
12. Pierre Wellner. The digitaldesk calculator: tangible manipulation on a desk top display. In *Proceedings of the 4th annual ACM symposium on User interface software and technology*, UIST '91, pages 27–33, New York, NY, USA, 1991. ACM.
13. Andrew D. Wilson. Using a depth camera as a touch sensor. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 69–72, New York, NY, USA, 2010. ACM.