Fingertip Detection Approach on Depth Image Sequences for Interactive Projection System

Conference Paper · October 2016

DOI: 10.1109/ETCM.2016.7750827

6 authors, including:

Roger Granda
Escuela Superior Politécnica del Litoral (ESPOL)
8 PUBLICATIONS **100** CITATIONS
SEE PROFILE

Arturo E Cadena
Escuela Superior Politécnica del Litoral (ESPOL)
10 PUBLICATIONS **38** CITATIONS
SEE PROFILE

Rubén Carvajal
Escuela Superior Politécnica del Litoral (ESPOL)
1 PUBLICATION **5** CITATIONS
SEE PROFILE

Katherine Chiluiza
Escuela Superior Politécnica del Litoral (ESPOL)
55 PUBLICATIONS **682** CITATIONS
SEE PROFILE

Fingertip Detection Approach on Depth Image Sequences for Interactive Projection System

Arturo Cadena*, Rubén Carvajal*, Bruno Guamán*, Roger Granda*, Enrique Peláez†, Katherine Chiluiza†
*Information Technology Center
Escuela Superior Politécnica del Litoral - ESPOL
Guayaquil, Ecuador
{arturo.cadena, ruben.carvajal, bguaman, roger.granda}@cti.espol.edu.ec
†Escuela Superior Politécnica del Litoral - ESPOL
Guayaquil,Ecuador
{epelaez, kchilui}@espol.edu.ec

***Abstract*—This study presents a vision-based approach for fingertip tracking on multi-touch tabletop which combines infrared and depth image processing. This approach intends to tackle two main issues on tabletop interaction: improve the performance for real-time applications and increase fingertip detection accuracy. A prototype using this fingertip tracking method was implemented with a depth and infrared camera. This approach processes the user's arm, hands and fingertips images using depth-space constraints, as well as clustering. Fingertip positions are accurately corrected using additional infrared information. Quantitative results show high accuracy of fingertip detection, with lower error rates compared to previous studies. Also, increased capabilities for real-time multi-user interaction are further demonstrated through a set of response time tests.**

1. Introduction

In recent years, there has been a growing tendency for including interactive tabletop systems within classrooms [1], [2]. The introduction of such technologies in classrooms to support co-located group-work activities has proven to be effective by decreasing social loafing [3], enhancing synergy in groups [4], and providing awareness to educators about student's contributions [5]. Academic institutions have started to acquire commercial interactive tabletops to be used in classrooms. Some examples are: Microsoft Surface Hub [6], Mimio Interactive Whiteboard [7], etc. However, the high cost of these solutions can be a barrier for their widespread use.

The availability of better quality and low range cost of optical sensors is enabling the construction of low-cost vision-based interactive tabletops [8]. These solutions allow a more natural and marker-less interaction by using commercial depth-cameras that have been the backbone technology to construct experimental applications [9], [10]. Moreover, due to continuous improvements in optical technologies, such as depth cameras, it is possible to capture more accurate information of 3D world. Some research has been done on the potential of these devices to capture more detailed information of hands interaction [11] [12]. However, there is still need for in-depth studies about tabletop implementations using depth cameras to support user-friendly gestures, as reported by [13].

Despite the advantages of using low-cost depth-cameras, there are still some issues to be addressed for improving interactive tabletop implementation. One issue is the inaccuracy at detecting fingertip positions. Since, after depth segmentation, the resulting hand contours are prone to erosion, which in turn leads to assign fingertip positions incorrectly [11]. On the other hand, real-time performance is another major issue which must be considered. Some research have overcome inaccuracy problems by analyzing the 3D image directly [14]. However, such approach leads to demand high computational costs, therefore not being able to perform on a real-time multiple-user settings.

The main contribution of this work is: a validated approach to provide an accurate fingertip detection for tabletop interaction by restoring eroded depth contours using infrared information. Additionally, the proposed system is suitable for performing real-time multi-user applications, since the frame rate is not affected when multiple users interact simultaneously. Nevertheless, it worth mentioning that the proposed approach could not deal with occlusion problems, since the analysis is performed on the image resulting from depth segmentation using deterministic techniques. Additionally, a tabletop prototype was implemented for testing this approach with up to five hands simultaneously.

This paper is structured as follows: Section 2 presents a summary of related work in this field; section 3 describes the proposed approach for finger tracking. Section 4 presents an evaluation of the proposed solution. Section 5 includes results of this work, conclusions and some reflections for future research are presented in section 6.

2. Related Work

Recent research [15] shows that hand-tracking systems commonly use depth cameras for hand segmentation. In

978-1-5090-1629-7/16/$31.00

tabletop-systems, 3D raw images are restricted to a particular depth range above the table resulting on a binary image containing hand contours within the defined range. Several works have defined the distance range to contain uniquely fingertips which are represented as blobs on the binary image and considered as touch points [10], [16]. Following the same approach, the work of [17] has included a technique for extracting other hand interaction features, such as handedness, user identification and the hand position. Nevertheless, these approach force users to position their hands at a fixed height and posture, contributing to the detriment of user interaction. In addition, many of these systems do not allow hands resting on tabletops, which may lead to wrong fingertips identification.

Other approaches such as [18], [19] have developed more sophisticated algorithms which analyze depth features directly from the 3D source by using: hand morphological constraints, particle filters, geodesic distance, among other 3D features. These studies report accurate fingertip detection at a variety of hand gestures and overcoming occlusion problems. Unfortunately, as most of them have been designed to recognize hand gestures over the air, they do not support a direct interaction on a physical surface. Additionally, such depth features analysis leads to a high computational cost, resulting in low frame rates, which may impact negatively the real time user interaction [20]. Additionally, some researchers have extended the blob tracking approach by defining a greater depth range that includes not just the fingertips but the whole arms of users for further analysis. Therefore, the arm contours extracted from the depth segmentation are evaluated utilizing several 2D features, such as convexity defects, contour simplification, k-curvature algorithm, among others to detect the fingertips position [21], [22]. Hence, a deeper 2D feature analysis is performed while maintaining lower computational costs than direct 3D feature analysis.

Dsensingni [9] for example simplifies the hand contour into a polygon representation, from which each angle within the finger vertexes is evaluated to be classified as a fingertip. However, such contour simplification may lead to information loss, especially when fingers are not stretched enough for being completely visible to the depth camera, which is located above. Similarly, the work proposed by [23] performs a k-curvature analysis on points of the original hand contour obtained from a depth image; k-curvature analysis has demonstrated a good performance in hand gestures.

Nonetheless, both of the previuos approaches extract information of the hand-contours obtained from the depth image, which is prone to erosion. This phenomena is caused by the operational error presented in depth cameras as well as the use of morphological operations performed in order to reduce noise. The proposed approach aims to solve the erosion problem by combining infrared and depth information. Thus, the infrared image is used to restore the information loss in the hand contours obtained from the depth image. As a result a more accurate fingertip detection was achieved. Additionally, by using this approach users are not forced to stretch their fingers to get their fingertips accurately positioned.

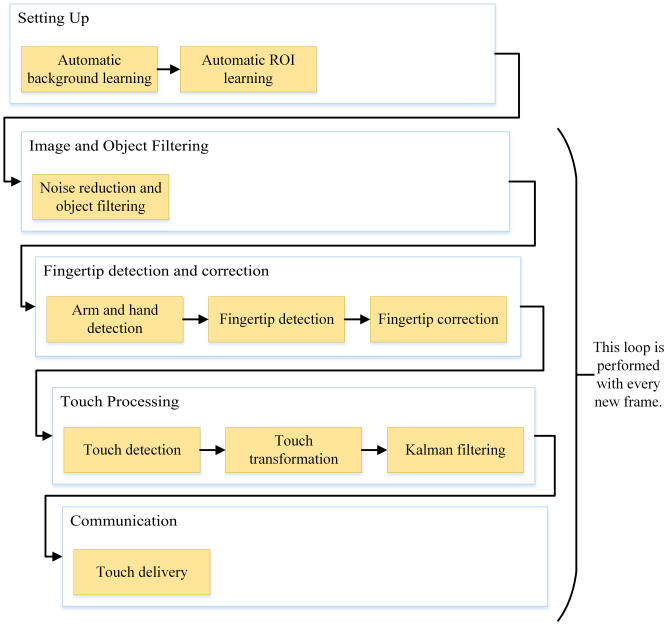

Figure 1. Overview of proposed system

3. Proposed approach

This section describes the approach followed to obtain, process, and track fingertips using depth and infrared images. This technique is developed in five stages: setup; image filtering; fingertip analysis and correction; touch analysis and correction; and, touch delivery (see Fig.1).

The setup stage is performed only once, when the process is launched. This stage is intended for learning the tabletop environment, as well as for delimiting the tabletop interaction area. From the second to fifth stage a set of actions are taken for noise removing, fingertip detection, touch processing, and touch delivering. A more detailed description of each stage is presented in the following subsections.

3.1. Setup

As a first step, an automatic background learning process is performed using the initial depth frames provided by the depth camera. From these frames an arithmetic mean is obtained for each depth pixel, resulting in a model of the background. This model is used as a reference for a depth background subtraction performed in the next stages.

During the second step, two regions of interest (ROI), one inside the other, are automatically created. The inner ROI aims to limit user's space interaction. Whereas, the outer ROI is for filtering out irrelevant objects outside its boundary, to avoid unnecessary information processing. Furthermore, outer and inner ROIs are used to identify hand and arm regions on the remaining objects, as described in the next section.

3.2. Image and Object Filtering

This stage is intended for filtering noise and potential non-hand objects. Initially, a binary image is obtained as a result of the depth background subtraction. In our experience, the resulting binary image usually showed salt and pepper noise. Which is removed using morphological operations: a single erosion and dilation with a rectangle mask. Next, the resulting objects are filtered by a contour size based in an upper and lower threshold. The remaining objects must have contour points outside and inside the inner ROI to be considered for further analysis.

3.3. Fingertip Detection and Correction

At this stage fingertip detection is performed by filtering out the image from the previous stage, which is explained as follows.

3.3.1. Arm and Hand Detection. As a first step, a contour analysis is performed on the resulting objects, which are assumed to be arm-hand objects. Thus, the contours obtained are called arm-hand contours.

Arm-hand contours are approximated to 2D polygons; which vertexes are clustered using a K-means algorithm. The K-means algorithm, in our case, is performed with K = 2, which allows the separation of vertexes in two clusters. The generated clusters are then evaluated to be classified as arm or hand regions. The cluster with more points inside the inner ROI are identified as hand, whereas the other cluster is labeled as arm.

3.3.2. Fingertip Detection. The fingertip detection algorithm is composed by two methods, one for finding the farthest points of the hand contour respect the center of mass of the arm-hand contour; and the other for performing contour curvature analysis based on a k-curvature algorithm [24]. These methods working together allows the detection of at least one fingertip even if the hand is closed or the fingers are joined together, as shown in Figure. 5.

The first method tries to find the farthest points of the hand (*Pf*, see fig. 2) with respect to the center of mass of the arm-hand. For doing this, consider the following reference points: point of the arm-hand contour (*Pi*), arm-hand contour center of mass (*CM*) and center of the cluster corresponding to the hand (*CH*). The search of *Pf* from the arm-hand contour is an iterative process, where every point of the arm-hand contour *Pi* is evaluated. Thus, an Euclidean distance between *Pi-CM* and *Pi-CH* is calculated. A candidate point *Pi* must satisfies the condition: distance *Pi-CH* less than distance *Pi-CM*. From the *Pi* candidate points, the point with the greatest *Pi-CM* distance is selected as the farthest point of the hand contour respect the center of mass *Pf*.

The second method is based on a k-curvature algorithm. Initially this algorithm takes three points for evaluation: point of the arm-hand contour (*Pi*), displaced forward *k* places *Pi* point (*Pi+k*) and displaced backward *k* places *Pi* point (*Pi-k*). The angle formed by the vectors *Pi-Pi+k* and *Pi-Pi-k* is calculated, if the angle is less than 70 °, a potential fingertip has been found, which is called *Ptest*. With the found point *Pi*(*Ptest*), a new point is calculated, which is located at the middle of a vector formed by the points *Pi+k* and *Pi-k*, called *Pmid*. Then, the Euclidean distance between *CM-Pmid*, *CM-Ptest* and *CH-Ptest* is calculated. A candidate point *Ptest* must satisfies the following condition to be a fingertip: distance *CM-Pmid* less than *CM-Ptest* and distance *CH-Ptest* less than *CM-Ptest*.

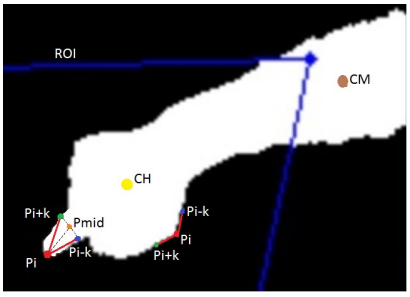

Figure 2. Points used in Fingertip Detection

3.3.3. Fingertip Correction. One disadvantage of the k-curvature algorithm is its accuracy, mainly because of the gaps observed on the contour points of the original obtained arm-hand contours, which causes the detected fingertip to be shifted respect the real fingertip. Another problem, as shown in Figure 3, when using depth channel images, there might be erosion of the fingers when they are closed to the table, resulting in a fingertip miss-placed compared to the real-finger location, which in turn affect the user interaction.

The first step to correct the above mentioned problems is generating a new arm-hand contour with no gaps between the points of the contours. These gaps create imprecision when calculating the position of *Pmid* because the length of the contour segment between *Pi* and points *Pi+k* and *Pi-k* might be different, or even more *Pmid* could be positioned outside of the detected finger. The gaps between the points of the arm-hand contour are completed using a linear equation defined from two consecutive points of the contour separated by a distance grater than 2 pixels. Then using the regenerated arm-hand contour, the farthest point of the finger respect to the point *Pmid* is found.

To correct the erosion problem, a frame of the infrared camera is used. For this frame a Gaussian Filter and Canny operator is applied to get the contours of the arm and hands. Using the corrected fingertip point *Ptest*, a ROI around this point is generated. Empirically, it was found that under conditions described on the evaluation section an optimal adjustment for region of interest was 10 x 10 pixels. This method uses three points for the analysis: *Pmid*, *Ptest* and a new point called *PtestIR*. This new point is iterated from all contour points inside the aforementioned ROI. Then, the angle between the segments *Ptest-Pmid* and *Ptest-PtestIR* is calculated. If a point *PtestIR* generates an angle close to 180°, it is selected to correct the fingertip position of

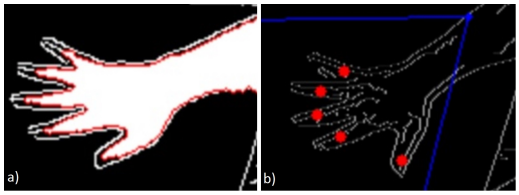

Figure 3. a) Segmented Hand Erosion from Depth Channel and external contour from Infrared Channel. b) Contour from Infrared Channel with fingertips without correction.

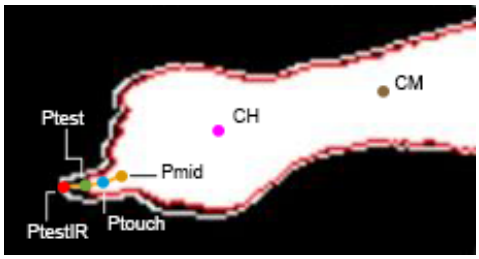

Figure 4. Fingertip position correction

Ptest. An example of fingertip position correction is shown in Figure 4.

3.4. Touch Processing

In this stage the touch points are identified using the fingertip points extracted from the last stage.

3.4.1. Touch Detection. For the touch detection technique, a new point called *Ptouch* is defined. This point contains a depth measure of the finger location with respect to tabletop surface. *Ptouch* is located on the middle of a straight line between the *Ptest* and *Pmid*. The distance between the table and *Ptouch* is obtained subtracting the raw pixel depth value of *Ptouch* from the learned background. To minimize the effect of the noise, an average of the depth measure from the points to the left, right, up and down of *Ptouch*, including this last one is obtained, and named as *Prom*.

In experimental tests it was found that touch events were lost frequently due to the noise of the depth image. To correct this, a hysteresis process was used. When the *Prom* depth value is less than the lower threshold, the state of touch is activated. But, if the *Prom* depth value is greater than the upper threshold, the state of touch is deactivated.

3.4.2. Touch transformation and Kalman filtering. In this step, touch coordinates are transformed from depth-camera space to a projected area. The resulting touch points, from the last stage, pass through a perspective transformation process using prior-known positions of the projected corners.

The final stage is used for delivering the resulting touch points to a client application. That is, touch-events are created and sent to a client application using a multi-touch protocol, once this stage is completed the process returns to the second stage to begin a new cycle.

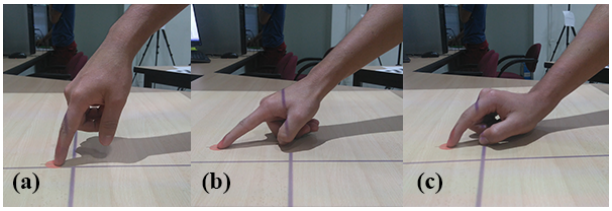

Figure 5. Finger pointing at the table different postures. (a) Pointing downwards using one finger; (b) Pointing stretching the finger; (c) Pointing at a relaxed finger posture

4. Evaluation

To validate the proposed approach, a prototype was constructed using a depth camera and an infrared sensor (See Figure 6). Both sensors operate at a frame rate of 30 fps. The depth camera is located at 0.80m above the table and pointing to the same. The dimensions of the tabletop interaction area are 0.54m x 0.86m; this area is generated with a pico-projector. Additioally, the implemented algorithm runs over a PC with an Intel Corei7 processor at 2.00 GHz, 8GB RAM and nVidia GeForce GT540M graphic card with 2 GB RAM.

Three different tests were deployed: one for measuring the fingertip tracking accuracy, the second to measure the system's touch delay, and the last one for measuring the operational frame rate of the system. Ten users were selected to participate in all the tests. The users were asked to use any variation of only the gestures shown in Figure 5 on every test.

4.1. Finger-tip Estimated Position Error

The evaluation plan is similar to the accuracy test performed by [17]. For this study, the evaluation is composed of two test where cameras were located at different height. Location of infrared and depth camera where 0.80m above the table for the fist test, and 1.20m for the second test. An application with a 6x6 grid of points developed under Unity framework was used in both test. The users were asked to touch 16 randomly-selected points on the tabletop projected area for 2 seconds. To show the accuracy of the system, the average error of fingertip position was calculated. This error was calculated as the difference between one of the randomly selected grid-point point and the system's estimated fingertip position, where the user touched the surface. As a result, 7476 samples were collected for analysis for each test.

4.2. Touch Delay

User's performance of tasks decreases when latency increases on direct-touch screens [25]. Computer-vision based solutions usually have higher touch delays compared to other solutions (resistive, capacitive or surface acoustic waves). In this study, touch delay was measured in two ways: an initial-touch delay, and an updated-touch delay.

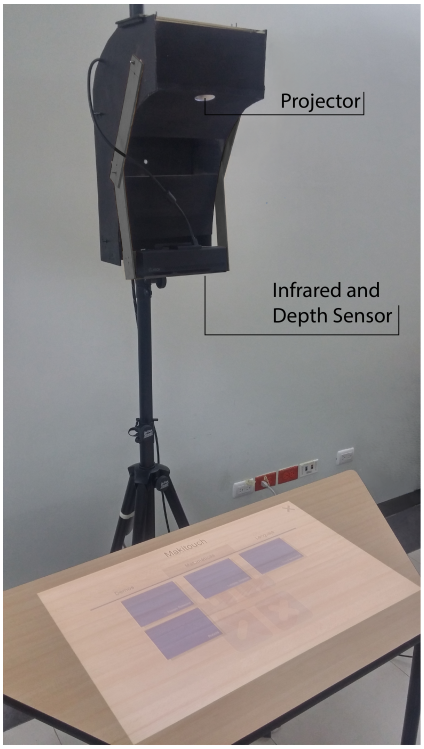

Figure 6. Picture of the tabletop implementation

Both measurements were performed using a high frame-rate camera at 75fps.

4.2.1. Initial Touch Delay. This measurement refers to the amount of time the system takes to recognize a touch event immediately after a fingertip is touching the tabletop projected area. To capture the initial-touch delay, users were asked to touch a random point 5 times on the projected area of the tabletop. As a result, 50 samples were collected for further analysis.

4.2.2. Update-Touch Delay. This measurement refers to the amount of time the system takes to update a touch-point position compared to the real-finger position, when a finger is moving. The updated-touch delay was measured asking users to move one finger towards one specific point on the screen. Each user had to repeat this experiment 5 times. As a result 50 samples were collected for further analysis.

4.3. Operational Frame rate

This measurement refers to the frame rate achieved when increasing the number of hands interacting with the system. Five individual test lasting one minute were carried out touching randomly at tabletop screen. The first test started with one hand; the subsequent tests considered two up to five hands.

5. Results

In this section, results from finger-tip estimated position error and delay tests are presented. Additionally, the operational frame rate of the system is reported. The outcomes of the finger-tip estimated position error test are the following: For the first test (0.80m depth-and-infrared camera) , results show that the average fingertip estimated position error was 0.53cm (st-dev: 0.37cm, max: 2.43cm, min: 0.01cm). For the second test (1.20m depth-and-infrared camera), fingertip estimated position error was 0.94cm (st-dev: 0.65 cm, max: 6.40cm, min: 0.01).

The average, maximum and minimum for the initial-touch delays were (in milliseconds): 105.5, 160.0, and 53.3, respectively. In general, the time it takes for recognizing a fingertip touching the tabletop surface is about 0.10 seconds. Moreover, for the updated-touch delay, the average, maximum and minimum were 193.5, 226.7 and 160.0, respectively. In this case, the delay is about 0.19 seconds; hence, it can be noticed that the updated-touch delay was almost twice as the initial-touch delay.

The Operational Frame Rate of the system was 30 fps for tests conditions with one, two, three and four-hands. Under the five-hands condition, 29 fps was the observed value. Thus, overall system performance works at a frame rate of 29.8 fps in average. Thereby, the frame rate values obtained at every measurement, indicate that increasing the number of hands do not affect significantly the performance of the system.

6. Discussion and Future Work

An approach for fingertip detection was proposed and implemented. This approach uses mixed information of depth and infrared images. Test for accuracy showed an average error of about 0.53cm for fingertip detection for 0.80m of depth and infrared camera. These errors are lower than the errors reported in other studies like the ones of [17] 0.8cm average, and [26] 0.59cm average.

As can be seen from the frame rate experiments, the obtained values were better or comparable to results presented by [9], [23] for one hand condition. Moreover, the performance of the implemented system was not affected when multiple hands were interacting simultaneously. Nonetheless, the results from the tests are not possible to be compared with other studies, since to our knowledge there are no research reporting frame rate measurements for multiple hand conditions.

Delay of initial recognition of fingertips on tabletop surfaces is low, which can be translated as rapid response. Despite the system achieved a good frame rate of 29 fps, the updated touch position had a delay of 0.2 sec. This delay was caused primarily by the Kalman filter delay. In order to overcome this delay an adjustment of the Kalman filter, it is necessary to explore a better model for tracking the hand movement. However, further research will be needed to handle better the randomness of the hand movement on touch interactions.

It could be noticed that the implemented prototype enabled users to rest their hands on tabletop surface, as can be seen in Figure 5c; which is nearly impossible on blob-based hand tracking. However, more research is needed to solve some interaction problems. All technical specifications described in the proposed approach were only validated on physical configurations mentioned on previous sections. Furthermore, this approach does not solve issues when occlusion of fingers occurs. Future versions of this work will include probabilistic 3D hand-tracking analysis to overcome this problem and to support interaction with more hand-gestures.

Acknowledgments

We would like to thank Banco de Ideas for their financial support for constructing the prototype for testing this technology, an initiative promoted by Senescyt - Ecuador.

References

[1] A. Kharrufa, R. Martinez-Maldonado, J. Kay, and P. Olivier, "Extending tabletop application design to the classroom," in *Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces*. ACM, 2013, pp. 115–124.

[2] A. C. Evans, J. O. Wobbrock, and K. Davis, "Modeling collaboration patterns on an interactive tabletop in a classroom setting," in *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. ACM, 2016, pp. 860–871.

[3] S. Buisine, G. Besacier, A. Aoussat, and F. Vernier, "How do interactive tabletop systems influence collaboration?" *Computers in Human Behavior*, vol. 28, no. 1, pp. 49 – 59, 2012.

[4] C. Anslow, P. Campos, L. Grisoni, and A. Lucero, "Collaboration meets interactive surfaces (cmis): Walls, tables, mobiles, and wearables," in *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*. ACM, 2015, pp. 479–483.

[5] R. Martínez, A. Collins, J. Kay, and K. Yacef, "Who did what? who said that?: Collaid: An environment for capturing traces of collaborative learning at the tabletop," in *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ser. ITS '11. New York, NY, USA: ACM, 2011, pp. 172–181.

[6] (2015) Microsoft surface hub. [Online]. Available: https://www.microsoft.com/microsoft-surface-hub/en-us

[7] (2015) Mimioteach interactive system. [Online]. Available: http://www.mimio.com/en-LA/Products/MimioTeach-Interactive-Whiteboard.aspx

[8] R. X. Granda, V. Echeverría, K. Chiluiza, and M. Wong-Villacrés, "Supporting the assessment of collaborative design activities in multi-tabletop classrooms," in *Computer Aided System Engineering (APCASE), 2015 Asia-Pacific Conference on*. IEEE, 2015, pp. 270–275.

[9] F. Klompmaker, K. Nebe, and A. Fast, "dsensingni: a framework for advanced tangible interaction using a depth camera," in *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*. ACM, 2012, pp. 217–224.

[10] A. D. Wilson, "Using a depth camera as a touch sensor," in *ACM international conference on interactive tabletops and surfaces*. ACM, 2010, pp. 69–72.

[11] M. Shekow and L. Oppermann, "On maximum geometric finger-tip recognition distance using depth sensors," *22nd International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, WSCG 2014, Communication Papers Proceedings-in co-operation with EUROGRAPHICS Association*, pp. 83–89, 2015.

[12] J. Steward, D. Lichti, J. Chow, R. Ferber, and S. Osis, "Performance assessment and calibration of the kinect 2.0 time-of-flight range camera for use in motion capture applications," *FIG Working Week 2015*, pp. 1–14, 2015.

[13] P. Sharma, R. Joshi, R. Boby, S. Saha, and T. Matsumaru, "Projectable interactive surface using microsoft kinect v2: Recovering information from coarse data to detect touch," in *2015 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2015, pp. 795–800.

[14] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect." in *BmVC*, vol. 1, no. 2, 2011, p. 3.

[15] H. Hasan and S. Abdul-Kareem, "Human–computer interaction using vision-based hand gesture recognition systems: a survey," *Neural Computing and Applications*, vol. 25, no. 2, pp. 251–261, 2014.

[16] Z. Pan, Y. Li, M. Zhang, C. Sun, K. Guo, X. Tang, and S. Z. Zhou, "A real-time multi-cue hand tracking algorithm based on computer vision," in *2010 IEEE Virtual Reality Conference (VR)*. Ieee, 2010, pp. 219–222.

[17] S. Murugappana, N. E. Vinayaka, and K. Ramania, "Extended multi-touch: Recovering touch posture, handedness, and user identity using a depth camera," *UIST12*, p. 487, 2012.

[18] H. Hasan and S. Abdul-Kareem, "Humancomputer interaction using vision-based hand gesture recognition systems: A survey," *Neural Computing and Applications*, vol. 25, no. 2, pp. 251–261, 2014.

[19] D. D. Nguyen, T. C. Pham, and J. W. Jeon, "Fingertip detection with morphology and geometric calculation," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 1460–1465.

[20] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, D. Freedman, P. Kohli, E. Krupka, A. Fitzgibbon, and S. Izadi, "Accurate, robust, and flexible real-time hand tracking," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI '15. New York, NY, USA: ACM, 2015, pp. 3633–3642.

[21] J. MacRitchie and A. P. McPherson, "Integrating optical finger motion tracking with surface touch events," *Frontiers in psychology*, vol. 6, 2015.

[22] K. Kim, J. Kim, J. Choi, J. Kim, and S. Lee, "Depth camera-based 3d hand gesture controls with immersive tactile feedback for natural mid-air gesture interactions," *Sensors*, vol. 15, no. 1, pp. 1022–1046, 2015.

[23] A. M. Genest, C. Gutwin, A. Tang, M. Kalyn, and Z. Ivkovic, "Kinectarms: a toolkit for capturing and displaying arm embodiments in distributed tabletop groupware," in *Proceedings of the 2013 conference on Computer supported cooperative work*. ACM, 2013, pp. 157–166.

[24] F. T. Cerezo, "3d hand and finger recognition using kinect," in *The 19th ACM Conference on Computer and Communications Security (CCS2012)*, 2012.

[25] R. Jota, A. Ng, P. Dietz, and D. Wigdor, "How fast is fast enough?: A study of the effects of latency in direct-touch pointing tasks," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '13. New York, NY, USA: ACM, 2013, pp. 2291–2300.

[26] A. Dippon and G. Klinker, "Kinecttouch: Accuracy test for a very low-cost 2.5d multitouch tracking system," in *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ser. ITS '11. New York, NY, USA: ACM, 2011, pp. 49–52.