Available online at www.sciencedirect.com

ScienceDirect

Procedia Manufacturing 3 (2015) 5381 – 5388

Procedia MANUFACTURING

6th International Conference on Applied Human Factors and Ergonomics (AHFE 2015) and the Affiliated Conferences, AHFE 2015

Finger click detection using a depth camera

Hani Karam*, Jiro Tanaka

Department of Computer Science, University of Tsukuba,1-1-1 Tennodai, Tsukuba, 305-8577 Ibaraki, Japan

Abstract

Selection mechanisms are an essential tool for designating elements in a User Interface. The advances in Natural User Interfaces promoted the selection mechanisms from devices such as the mouse, to more natural methods such as hand gestures. Recently, depth sensors became widely available, and paved the way for new methods of hand gestures recognition. In this paper, we present Depth Click, a new approach of finger click detection using a depth camera. Our proposed method provides natural interactions in terms of selection mechanisms, and goes beyond that in allowing the clicks to be used for actions other than selection. Our technique proved to be fast, reliable and flexible. This paper explains the algorithm we used to detect the finger clicks in detail. We explain about some preliminary experiments for evaluating the accuracy of our approach, as well as its detection rate. Finally, we discuss the results of the evaluation.

*Keywords:*Natural User Interface; Finger click detection; Human Computer Interface; Depth camera

1. Introduction

A User Interface (UI) is seldom made from just one item; it usually consists of several elements, each one having its own use. To operate such an interface, a user has to be able to designate those elements distinctly, hence the need for selection mechanisms. In traditional Graphical User Interfaces (GUIs), users point to items (buttons, icons, etc…) by using an input device, usually a mouse, and select a given item by performing a click operation. However, in the recent years, with the advances in Natural User Interface (NUI), many researches are focusing on using hand gestures as a means for user interaction, as it has been shown that gestures tend to be an intuitive and natural way to

* Hani Karam. Tel.: +81 29-853-5165; fax: +81 29-853-5165.
*E-mail address:*hani@iplab.cs.tsukuba.ac.jp

doi:10.1016/j.promfg.2015.07.653

communicate with a computer [3], as well as that gestures are effective means of interaction with the everyday desktop [1]. Many gesture-based selection mechanisms have been proposed [12, 14, 15, 17], and while some of them are natural, most of them require the user to perform unintuitive gestures, or are limited to a very basic selection command.

In this paper, we present Depth Click (Fig. 1), a new approach for finger click detection using a depth camera. Our method allows us to use a click not just as a selection mechanism, but as a way to start and maintain many other different gestures as well.

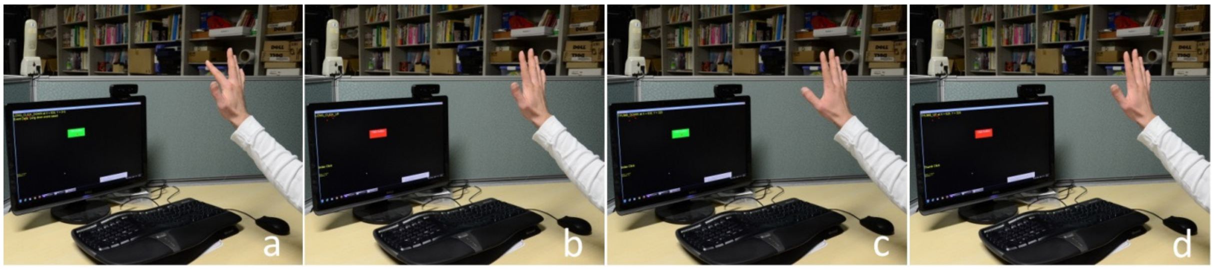

Fig. 1. Example of interactions: (a) and (b) depict an index-click; (c) and (d) show a thumb-click.

2. Related work

Gesture based interactions have been the subject of research for over three decades. It has been shown that gestures can be a good candidate for controlling a computer [1, 2]. Wachs et al. [3] point out that gestures are useful for computer interaction since they are the most primary and expressive form of human communication. They also state that gestures should be designed to be intuitive for users; that is, a gesture should be related to the operation it is associated with. Gesture data fetching is an important procedure in hand gesture recognition, and the used methods can be mainly divided into two categories [4]: glove-based and computer vision based. Data gloves can be used to precisely acquire the gesture's data, but they often encumber the user with tethers and setup time. However, computer vision based recognition is nonintrusive and untethered, thus freeing the user from donning any special hardware or apparatus to enable him to interact with the system, which gives rise to a more natural interaction [5]. This complies with the "Come as you are" Human-Computer Interaction (HCI) design [6]. For those reasons, we have opted for the vision-based approach.

Chu et al. used hand gestures to take self-portraits [12]. They used an onscreen "hover button interface" as a selection mechanism, where they would hover with a finger to trigger the camera's shutter. But because of the noise of detection and cluster background, they have to keep the finger for 1 second over the button. While this reduces the amount of errors, it introduces a waiting time in the interaction. And since time is used as a trigger mechanism, the user has a limited interval during which he can change his mind. [15] and [16] used a crossing technique that was originally developed for pen-based interfaces: the crossing event occurs when the cursor intersected with the boundary of a graphical object. Nakamura et al. [17] improved on that by introducing "double-crossing" in which a crossing motion occurs twice in a short time interval. This helped reduce errors since unintended double-crossing occurs much less frequently than unintended crossing. While effective, these crossing techniques are limited to a basic selection operation. Another approach used in [14] consists of using "index pointing, thumb up" gesture to perform a selection. However, this gesture doesn't feel intuitive as a selection technique, and contradicts the idea presented in [3]. Kulshreshth et al. introduced a 3D gesture menu selection approach based on finger counting [18] in which all the menu items are numbered and the user has to extend a corresponding number of fingers to select a given item. It was shown to be a viable option for 3D menu selection tasks, especially that it can work in different layouts. Nonetheless, items have to be numbered, and the fingers will be used for counting, and are thus dedicated to this operation and cannot be used for other different tasks. A finger clicking interface for Augmented Reality systems was presented in [19], since it feels more natural for a user to interact with a virtual object in a similar way he uses his hands to interact with objects in the real world. The limitation of this approach is that the click is used just as a selection mechanism. Another restriction of most of the above mentioned methods is that once they are started, the user cannot cancel them, that is, if a user unintentionally selects an item, and realizes that he made a mistake, he cannot have a change of mind.

3. System overview

In this section we will discuss the approach we took, the click detection algorithm, the hardware we used, as well the implementation of the algorithm along with a prototype.

3.1. Approach

In the recent years, advancements have made depth sensors widely available as commercial products, such as the Microsoft Kinect and SoftKineticDepthSense. Unlike RGB (color) cameras, depth cameras add an additional dimension in depth maps and thus allow data to be captured in 3D. 3D gesture recognition methods can be mainly divided into two parts: machine learning and heuristics-based [7]. In the case of machine learning recognition, important features are extracted from the data and are then used as input for a classification algorithm. In addition, training is needed to make the classifier more robust and to maximize accuracy. Conversely, in a heuristic based approach, no machine learning algorithms are used. A simple heuristic recognizer for a “jump” action would check the difference between the current head’s position and its normal position: if the difference is greater than a given value, then a jump is recorded. While machine learning based recognition is more commonly used [8, 9, 10, 11], we have opted for heuristics-based recognition.

A simple heuristics-based solution for detecting an index-click would be to check when the distance of the index finger to the camera is smaller than the distances of the other fingers. However, this approach is flawed in the fact that if the hand is tilted inwards, the distance of the index to the camera will never be smaller than that of the middle, ring and small fingers’. In our previous research [13], we had introduced the “Three Fingers Clicking Gesture” to detect a click through a depth camera. We used the thumb, the middle finger and the palm center to define a plane, and we computed the angle between this plane and the vector formed by the palm center and index fingertip. Whenever the angle crossed a given threshold, a click was generated (Fig. 2).

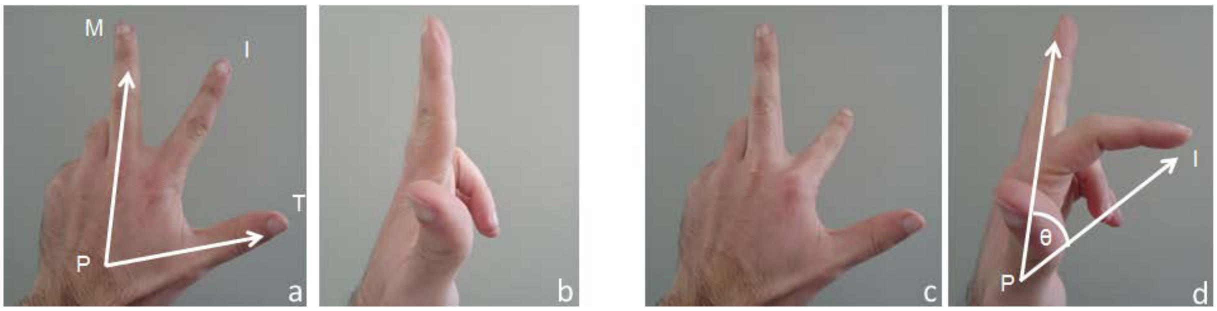

Fig. 2. The Three Fingers Clicking Gesture

To perform the gesture, we assume that only the thumb, index and middle fingers are held open; the index is then bent forward to initiate a click. The method does not need any training, and proved to be viable as a way of click detection; and since the palm is being used as a reference vis-à-vis the finger, this method had the advantage of being detectable even if the hand moved, as the reference is moving along with it. Using this approach, the gesture was also detectable even if the camera was behind the hand. Another advantage was that the selection mechanism was not affected by hovering time: the user can keep his hand above the target as much as desired, and the target will only be selected once a click has been performed. A possible usage scenario for this is the display of tooltips as long as the hand is hovering the target. Nonetheless, most users complained about the difficulty of maintaining this gesture.

3.2. Click detection algorithm

In our current approach, we computed the complement of the angle between the vector composed from the palm center and the index fingertip, and the normal vector protruding from the palm center. This method showed an ergonomic amelioration in the fact that users do not need to hold open three fingers anymore, and the neutral position became a much more comfortable open palm. Another enhancement was on the practical side: now that the thumb and middle finger were not used to generate a plane anymore, and that the ring and small finger where not closed anymore, we were able to detect a click with the individual fingers, and so a click was not linked to the index finger alone. This led to the ability of performing a "multi-click", that is, multiple fingers being clicked down simultaneously (the same way multi-touch is used with touchscreens).

Furthermore, a click has now been split into a "Click Down" (the finger crosses the threshold) and "Click Up" (the finger goes back to its original position). A click was hereby defined as a Click Down, followed by a Click Up, over the same interface target. We have empirically determined that an angle of 12 degrees is a viable threshold.

Separating a click into two different events opened up new possibilities in the detection. An example isthe "change of mind": assuming that a user performs a Click Down over a button, then realizes that it was the wrong button, all he has to do, while maintaining the Click Down posture, is move his hand so that the cursor exits the target. Upon releasing the click (performing a Click Up) with the cursor outside the target, no click is detected and thus allowing the user to cancel the event.

Moreover, we were able to detect a new event, a "Long Click", generated when the user keeps his finger in a Click Down posture over a given period of time (we have used 500 milliseconds in our tests). Once the user releases the click, the long click event is terminated. While we did not use any checking on the number of frames regarding a regular click's detection, we have imposed a duration constraint on the long click to make it more stable: after a long Click Down, the user has to keep the Click Up posture for 10 frames, that is, 333 milliseconds running at 30 fps (we discuss this further in the evaluation part). A possible application of this event is dragging and dropping targets. We were also able to detect a "Double-click" whenever two regular clicks were performed within an interval of time that is less than a given threshold (we used a 1000 millisecond threshold in our examples).

3.3. Hardware

To implement our prototype, we used a Creative Senz3D depth sensing camera, which uses Time-of-Flight technology, capturing videos at 30 fps. The resolutions are 640 x 480 for RGB and 320 x 240 for depth. The camera was placed on top of a 23" monitor with Full HD resolution, facing the user, tilted down approximately 10 degrees. The prototype has been implemented on a computer equipped with an Intel Core i5 3.2 GHz CPU and 4GB of RAM.

3.4. Prototype

We used the Intel Perceptual Computing SDK 2013 to retrieve the 3D positions (camera space coordinates) of the fingers and palm center, as well as the normal vector protruding from the palm. We used those points to construct the various vectors described in section 3.2. We also used the SDK to retrieve the 2D palm center position in the depth image (image space coordinates), which was extrapolated to a 1920 x 1080 resolution and then used as a pointing cursor that indicates the position of the hand. Onscreen rendering was implemented in SFML, and the entire prototype was written in C++.

As a demo, we implemented a simple image viewer. We have defined 5 gestures: index-click, thumb-click, click-and-drag, click-and-zoom and click-and-rotate, the latter 3 using an index-click as a way to initiate and terminate their respective interactions. The thumb-click is used to toggle commands between drag, zoom and rotate operations. In drag mode, users can perform a Click Down with the index finger and drag an image left or right; the next image will be displayed upon releasing the click. In zoom mode, the users execute an indexClick Down and bring their hand closer to the screen to zoom in, or pull their hand away from the screen to zoom out, and release the click once they wish to stop zooming. To rotate an image, the users also perform an indexClick Down then rotate their hand (similar to a waving motion). The image rotates accordingly, until the click is released (Fig. 3).

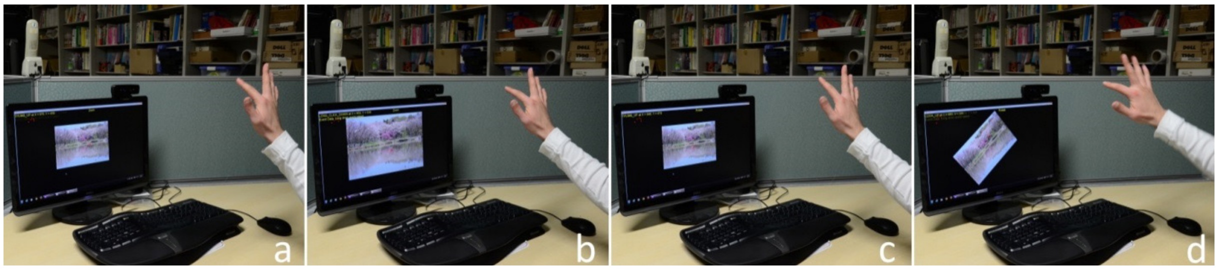

Fig. 3. The user clicks (a) and pushes his hand closer to the camera to zoom in (b); to rotate, he clicks (c) and turns his hand (d)

4. Evaluation

We proposed the following three hypotheses: (H1) Depth Click can help users complete a given task in a faster way, (H2) Depth Click is less error prone than some other proposed selection mechanisms and (H3) Depth Click is more natural that those mechanisms. To evaluate our system, 10 participants (5 males, 5 females) aged between 20 and 36 were recruited; 5 among them are computer scientists/engineers. All of them are highly familiar with computers, 2 of them have experience in gestural interaction, and 1 is left-handed. After explaining the process of the experiments to the participants, they were given 2 minutes to practice the different interactions (index-click, index double-click, thumb-click, index long-click). They then completed two different experiments, and were asked to fill out a questionnaire.

4.1. First experiment: comparison with other types of selection mechanisms

In this experiment, 9 blue buttons, sized 100 x 100 pixels, were arranged in 3 rows of 3 buttons each, separated by 100 pixels horizontally and vertically (Fig. 4). The buttons were numbered from 1 to 9. An additional "Start" button was also added. After selecting the Start button, a random number between 1 and 9 is displayed at the top of the screen, and the participants had to select the corresponding button. Upon any selection, a newrandom number is shown. However, selecting a button with a number different than the one that was shown counted as an error. 20 selections were required. The number of errors and the completion time were saved. The participants were required to repeat this experiment 4 times, once per selection mechanism (80 selections per participant, for a total of 800 selections).

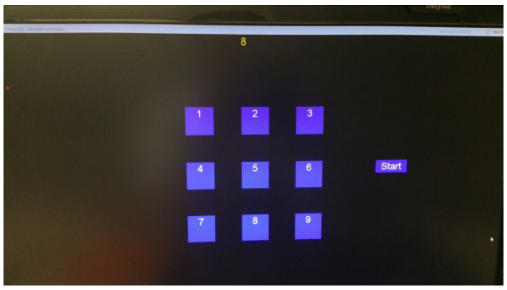

Fig. 4. 9 buttons, plus the Start button at the right. A number at the top of the screen showed the button that should be selected.

A cursor showed the position of the palm center. Whenever the cursor hovered above a button, that button's color changed to red to indicate hovering to the user. In the case of Depth Click and Closed Hand,a hovered button's color flashed to green upon a positive selection. The different selection mechanisms used in this experiment were:

- Depth Click: our method (Click Down, then Click Up over the same button).

- Double Cross: the cursor has to hover a button, exit the button’s boundaries, then hover the same button again. Hovering an unintended target only once was not counted as an error.
- Cross: the cursor hovers a button. An unintended cross was counted as an error.
- Closed Hand: the user hovers a button, then closes his hand to select.

4.2. Second experiment: detection test

The participants were asked to perform 20 clicks over a single button sized 200 x 100 pixels. They were required to repeat this experiment 3 times (60 clicks per participant, for a total of 600 clicks), each time using a different click (index-click, thumb-click and double-click). Upon completion, the system saved the number of detected clicks for each type. A human observer was counting the actual number of clicks performed by the participants, and asked them to stop whenever they reached 20.

4.3. Questionnaire

After completing the experiments, the participants were asked to fill the following questionnaire, to which they could respond on a five point Likert scale (-2 = Strongly Disagree, 2 = Strongly Agree): (1) Is an Index-click easy to perform? (2) Is a Thumb-click easy to perform? (3) Is a Double-click easy to perform? (4) Does our method feel more natural than other selection mechanisms? (5) Does a Long Click feel more natural than a Closed Hand for performing dragging actions?

In the additional “What is your preferred selection mechanism?” question, the participants had to choose an answer out of the following 4 possibilities: Cross, Double Cross, Click, Closed Hand. The participants were also offered the opportunity to enter their reason, and overall comments, freely.

4.4. Results

We compared the different results using a t-test. In the first experiment, the mean time to perform 20 selections using Depth Click is 55.45 seconds (S.D 12.58). While it is slightly faster than Double Cross (mean 57.2, S.D 12.61): T(18) = -0.3, p = 0.76, the difference is not significant. Our method was slower than Cross (mean 44.4, S.D5.54): T(12) = 2.54 , $p < 0.05$, showing that the difference is considerable. Finally, Depth Click is also slightly faster than Closed Hand (mean 58.21, S.D 12.61): T(18) = -0.48, p = 0.63, but the difference is not significant.

Our technique had a success rate of 18.2 (91%) (S.D 1.39), higher than Double Cross (mean 16.8 (84%), S.D 1.87): T(17) = 1.89, p = 0.07. It also had a success rate higher than Cross (mean 15.6 (78%), S.D 2.36): T(15) = 2.99, $p < 0.05$, and higher than Closed Hand (mean 17.6 (88%), S.D 2.95): T(13) = 0.58, p = 0.57.

In the second experiment, out of 20 clicks, the average numbers of detected Index-clicks, Thumb-clicks and Double-clicks were respectively 17.3 (86.5%, S.D 2.21), 17.3 (86.5%, S.D 3.02) and 17.5 (87.5%, S.D 3.2). Fig. 5 shows the results of both experiments.

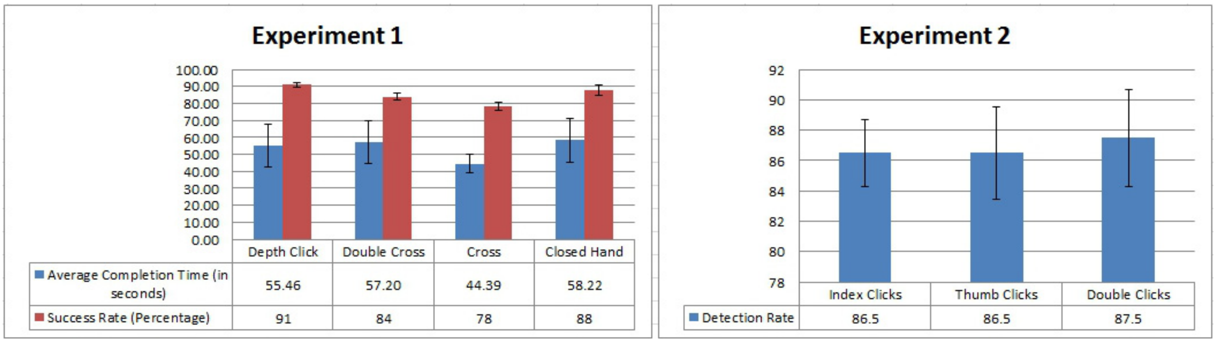

Fig. 5. The results of Experiments 1 (completion time and accuracy) and 2 (detection)

As for the questionnaire, all the averages were above neutral. Most people agreed that an Index-click is easy to perform (mean 1.2, S.D 0.63) and many agreed that a Thumb-click is easy to do (mean 0.6, S.D 0.84), same for a Double-click (mean 0.9, S.D 0.73). Most participants strongly agreed that a click is more natural than the other proposed selection mechanisms (mean 1.4, S.D 0.69), and many felt that a Long Click is more natural for dragging (mean 0.7, S.D 1.15). 8 out of 10 participants (80%) chose clicking as their preferred selection mechanism.

4.5. Discussion

Depth Click was faster than Double Cross and Closed Hands; however, the difference in completion time was not significant. It also had a higher success rate than those methods, but nonetheless, the difference was not considerable. We can say that H1 and H2 were proven for those cases, albeit not in a notable way. Moreover, while slower than Cross (we attribute the speed of Cross to its high error rate), our method's error rate was significantly lower than that of Cross, so H2 was proven in this case too. Furthermore, with detection rates higher than 86%, we can say that the method is robust and reliable.

As per the answers of the questionnaire, we can state that clicking is more natural than the other selection mechanisms, and it was the preferred one, thus proving H3.

One of the problems that we faced was the fingers detection. Since we were using the camera's SDK to retrieve the fingers' positions, the detection of a click was constrained by the camera's abilities to detect the fingers. The camera proved to be sensitive to strong ambient light, and in many cases, the hand would stop being detected in many frames, even if the user did not move his hand. Since our method relies on having a Click Down and Click Up being performed over the target, if the palm center's detection fails during the Click Up gesture, it would be considered as a cancelled click, and thus not detected. The detection problem especially affected the Long Click: while being pressed down, the finger might not be detected and thus the interaction would end. To enhance the detection rate, we required a Click Up to be held for 10 frames (333 milliseconds at 30fps) after a Long Click.

Depth Click inherited all of the Three Fingers Clicking Gesture's advantages (section 3.1). In the following table, we present additional advantages of our approach that could not be quantified in experiments:

Table 1. Depth Click's possibilities compared with other techniques

	Depth Click	Double Cross	Cross	Closed Hand	Hover
Detection not limited by time	Yes	Not applicable	Not applicable	Yes	No
Allows a change of mind	Yes	Yes	No	No	Partial
Allows a "long press"	Yes	No	No	Yes	No
Unaffected by the interface layout	Yes	Partial	No	Yes	Partial
Fingers independency (multi-click)	Yes	No	No	No	No
Can be used for actions other than selection	Yes	No	No	No	No
Detectable without a target	Yes	No	No	Yes	No

Regarding the last entry (Detectable without a target), even though we specified that a Click Down and Click Up should be executed over the target in order to be detected, our method still allows us to detect a click in general, even if there is no target. For example, in our prototype, a user can change commands by using a Thumb-click anywhere, unlike dragging which has to be performed over the image.

We have also found that there is no direct correlation between having experience with gestural interfaces and completion speed, since those who previously used gestural interfaces were not the fastest to complete the tasks. However, we have found that, in general, those who were majoring in Computer Science / Engineering were the fastest to complete the tasks.

4.6. Possible applications

An interesting application would be the manipulation of mobile phones, should they be equipped with depth cameras in the future (such as the Nokia McLaren prototype). Another possible area of applications would be with Head Mounted Displays (HMD), such as Atheer Labs' AiR Smart Glasses which are equipped with a depth camera. Since Depth Click can be detected even if the camera is behind the hand, this makes it an ideal candidate for HMDs.

5. Conclusion and future work

We have presented Depth Click, a novel heuristics-based approach for detecting finger clicks using a depth camera without the need for training. Our method has proven to be reliable, natural, and allows users to cancel a click, thus reducing the possibility of unintended selections. Our approach is also very flexible, enabling clicks to be detected by different fingers independently, as well as enabling double-clicks to be detected. It can also be used for actions other than selection, thus expanding the possibilities of its output when mixed with other hand movements.

In our future work, we would like to enhance the detection rate by taking into consideration the hand openness. We also would like to compare it with machine-learning based approaches.

References

[1] J. Segen , S. Kumar, Look ma, no mouse!, Communications of the ACM, v.43 n.7, 2000, pp. 102–109

[2] C. von Hardenberg , F. Bérard, Bare-hand human-computer interaction, Proceedings of the 2001 workshop on Perceptive user interfaces, ACM, 2001, Orlando, Florida, pp. 1–8

[3] J. P. Wachs , M. Kölsch , H. Stern , Y. Edan, Vision-based hand-gesture applications, Communications of the ACM, v.54 n.2, February 2011

[4] P. Garg., N. Aggarwal, and S. Sofat, Vision Based Hand Gesture Recognition. Engineering and Technology 49, 3 (2009), 972–977

[5] Y. Boussemart , F. Rioux , F. Rudzicz , M. Wozniewski , J. R. Cooperstock, A framework for 3D visualisation and manipulation in an immersive space using an untethered bimanual gestural interface, Proceedings of the ACM symposium on Virtual reality software and technology, ACM, Hong Kong, 2004, pp. 162–165

[6] J. Triesch, C. Von Der Malsburg. Robotic gesture recognition by cue combination, Informatik'98. Springer Berlin Heidelberg, 1998, pp. 223-232

[7] J. LaViola, Jr., An introduction to 3D gestural interfaces, Proceedings of SIGGRAPH '14 ACM SIGGRAPH 2014 Courses, Article No. 25 (2014)

[8] F. Dominio, M. Donadeo, G. Marin, P. Zanuttigh, G. M. Cortelazzo , Hand Gesture Recognition with Depth Data, ARTEMIS '13 Proceedings of the 4th ACM/IEEE international workshop on Analysis and retrieval of tracked events and motion in imagery stream, Barcelona, 2013, pp. 9–16

[9] K. Lai, J. Konrad, P. Ishwar , A gesture-driven computer interface using Kinect, Image Analysis and Interpretation (SSIAI), 2012 IEEE Southwest Symposium on, IEEE, Santa Fe, New Mexico, 2012, pp. 185–188

[10] K. K. Biswas, S. K. Basu, Gesture Recognition using Microsoft Kinect, Automation, Robotics and Applications (ICARA), 2011 5th International Conference on. IEEE, 2011, pp. 100–103

[11] U. Lee, J. Tanaka, Finger Identification and Hand Gesture Recognition Techniques for Natural User Interface , APCHI '13 Proceedings of the 11th Asia Pacific Conference on Computer Human Interaction, ACM, Bangalore, 2013, pp. 274–279

[12] S. Chu , J. Tanaka, Hand gesture for taking self portrait, Proceedings of the 14th international conference on Human-computer interaction: interaction techniques and environments, Springer, Orlando, Florida, pp. 238–247

[13] H. Karam, J. Tanaka, Two-Handed Interactive Menu: An Application of Asymmetric Bimanual Gestures and Depth Based Selection Techniques, Proceeding of 16th International Conference, HCI International 2014, Springer, Heraklion, Crete, 2014, pp. 187–198

[14] J. C. Lévesque, D. Laurendeau, M. Mokhtari, An Asymmetric Bimanual Gestural Interface for Immersive Virtual Environments, 5th International Conference, VAMR 2013, Held as Part of HCI International 2013, Springer, Las Vegas, 2013, pp. 192–201

[15] J. Accot, S. Zhai, More than dotting the i's - foundation for crossing-based interfaces, CHI '02 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, Minneapolis, 2002, pp. 73–80

[16] G. Apitz, F. Guimbretiere, CrossY: A Crossing-Based Drawing Application, UIST '04 Proceedings of the 17th annual ACM symposium on User interface software and technology, ACM, Santa Fe, 2004, pp. 3–12

[17] T. Nakamura , S. Takahashi , J. Tanaka, Double-Crossing: A New Interaction Technique for Hand Gesture Interfaces, Proceedings of the 8th Asia-Pacific conference on Computer-Human Interaction, Springer, Seoul, 2008, pp. 292–300

[18] A. Kulshreshth, J. J. LaViola, Jr, Exploring the usefulness of finger-based 3D gesture menu selection, CHI '14 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, Toronto, 2014, pp. 1093-1102

[19] A. Sugiura, M. Toyoura, X. Mao, A natural click interface for AR systems with a single camera, GI '14 Proceedings of the 2014 Graphics Interface Conference, CHCCS, Montreal, 2014, pp. 67-75