A depth camera-based system to enable touch-less interaction using hand gestures

Conference Paper · August 2021

DOI: 10.1109/NIR52917.2021.9666090

6 authors, including:

Ruslan Damindarov
Innopolis University
1 PUBLICATION 4 CITATIONS
SEE PROFILE

Cham An Fam
Innopolis University
1 PUBLICATION 4 CITATIONS
SEE PROFILE

Riby Abraham Boby
Indian Institute of Technology Madras
30 PUBLICATIONS 152 CITATIONS
SEE PROFILE

Muhammad Fahim
Queen's University Belfast
9 PUBLICATIONS 18 CITATIONS
SEE PROFILE

A depth camera-based system to enable touch-less interaction using hand gestures

R. Damindarov[1], C. A. Fam[1], R. A. Boby[1], M. Fahim[2], A. Klimchik[1], T. Matsumaru[3]

***Abstract*—Touch-less interaction is an active area of research due to its numerous applications ranging from medical systems to gaming. In this paper we propose a novel interactive system based on RealSense camera to track the pose of hand and to recognize gestures. Our approach is based on MediaPipe model introduced by Google to detect the landmarks of the hand. The system utilizes the camera calibration information and user actions to get the geometry of the deployed environment, which shows high accuracy in measurement of hand position. The touch-less interaction is demonstrated using a graphical user interface for painting.**

I. Introduction

In the recent years, the interest related to 3D camera based interactive touch screens is increasing tremendously. One of the primary reasons is that it does not require to wear auxiliary tools, such as glasses or trackers which may cause discomfort for the users [1]. The most common interfaces based on keyboard and mouse, are not suitable for all people, especially elderly or small children [2]. However, over the last year with prevalence of infectious disease all over the world, another requirement has propped up. The user may not want to perform the actual touch with the screen because of the environment or sanitary reasons. This motivates the development of "touch-less interactive screens". In our work we changed the action of touching a screen to pointing the finger towards the screen and using hand motions and gestures to register the process of making a choice. The designing of "touch-less interactive screens" would definitely attract much interest, especially in this pandemic period. The task of these devices may be different and it depends on the goal of designing them. For example, they may detect the hand, fingers or even objects held in hand. This work discusses a mechanism of tracking the hand pose using depth camera [3]. It is expected that these kind of systems will be better in terms of ability to recognize multiple finger based gestures and accuracy, in comparison with systems based on alternative sensors [4], [5] that were used earlier.

[1]Innopolis University, Universitetskaya 1, Tatarstan, Russia; e-mail: `r.damindarov@innopolis.university`, `c.fam@innopolis.university`, `ribyab@gmail.com`, `a.klimchik@innopolis.ru`.

[2]School of Electronics, Electrical Engineering and Computer Science, Queen's University Belfast, Northern Ireland; e-mail: `M.Fahim@qub.ac.uk`

[3]Graduate School of Information Production Systems, Waseda University, Kitakyushu, Japan;e-mail: `matsumaru@waseda.jp`

978-1-6654-2407-3/21/$31.00 ©2021 IEEE

The major focus of this article is the following:

- Interaction with a screen without touch using only hand gestures utilizing finger poses detected using deep learning based approach;
- use of a compact and accurate depth camera to detect interaction and technological aspects for building a demonstrative model.

The structure of this article is as follows: section II discusses about the existing methods of creating interactive systems. The details of the setup are described in section III. Section IV reveals the methods used in the study. Later, the experimental results and discussions are presented in sections V and VI. Section VII concludes the article.

II. Related Work

An earlier work considered using Kinect V2 as a depth camera for projectable interactive surfaces for children's education purposes. It was named as Image-Projective Desktop Varnamala Trainer (IDVT). Sharma et. al. [4] pointed out the portability of camera-projector systems, which means that they can work at any surface. It can be a floor, table or some screen. In addition, it can be easily integrated with the existing projector based systems, commonly available at schools. However, one should notice that the speed of detection is rather slow, since the accuracy was very low compared to the RealSense sensor that is used in this study. Also there are significant errors in computing the distance between the screen to be touched and the real position. Finally, Kinect is bulky and it is not suitable for many interactive/robotic applications which require compact sensors.

Another approach suggested using the scanning laser range finder [5] in the tracking system. The advantage was the portability and the possibility to use screens of different sizes. The measurement accuracy was high and a large variety of applications could be developed using it [6]–[9]. As a drawback, only 2D measurements can be made, so a wide variety of hand gestures can not be directly detected. Additionally, it is comparatively expensive and the use of laser light may not be the suitable choice for applications involving human subjects. Both earlier existing systems modeled interaction as an actual touch on the surface.

Alternatively, Red, Green, Blue (RGB) camera was used to build such devices [10]–[12]. But the approach may be challenging when there is possibility of large variation in illumination in the background that may be unsuitable for customized situations. A better approach is to combine both

RGB data and depth information using depth cameras. There are other alternative sensors, for example, time of flight camera which is also being applied in [13]. The most interesting development is a system that uses both visible information and depth data [14], [15]. However, in these cases, there is no possibility of incorporating gestures i.e., the concept of "touch-less interaction", since all of the approaches were conceptualized before the start of the pandemic all over the world. It may be particularly noted that the already existing SDK [16], [17] for RGBD sensors detects mainly the skeleton points and can not be used directly for detecting the fingers and determining gestures. Therefore, in the earlier implementations the movement of the complete arm (rather than the finger) was used for interaction.

The interactive devices discussed earlier were meant mainly for large sized screens created using a projector. The earliest version named Interactive Desktop Arm Trainer (IDAT) was used for rehabilitation of stroke patients using games [9]. The later versions of IDVT were used in educating children through games which is commonly referred to as edutainment. However, these versions used only few keys (maximum of 5) and had not used the full capability of the first version wherein the interaction could be done all over the screen that was displayed.

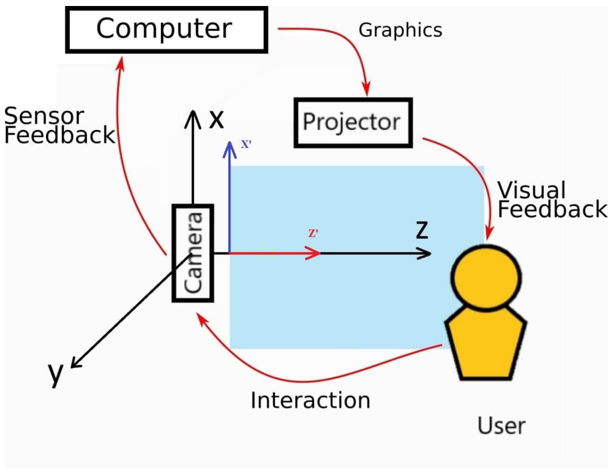
According to a survey of the existing literature the following aspects have been observed:

- Majority of existing systems focus on touch based interaction and they are not suitable for the current epidemiological situation;
- qualitative hand gestures using fingers could not be detected;
- existing systems were limited by less accuracy causing errors in detection, lack of on-board processing leading to slow speed of processing, etc.

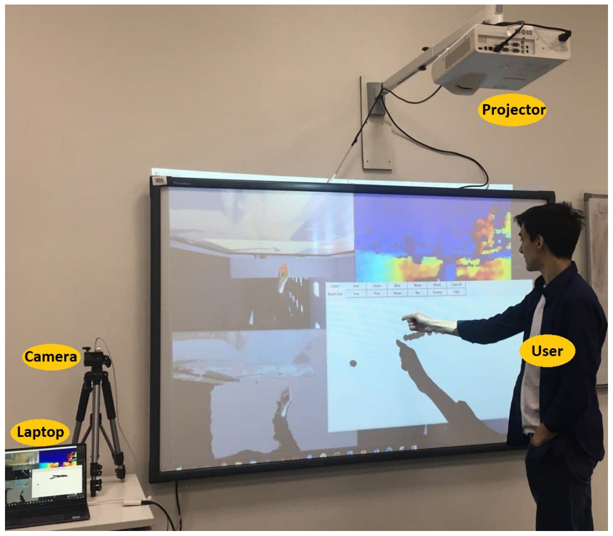
The proposed system is an attempt to tide over these problems/issues in the existing systems.

III. System Setup

The basic structure of the system is illustrated in the Fig. 1a. The origin of a coordinate system is located in the middle of the sensor. The most important component of the system is the sensor Intel RealSense D435 with depth measurement range of 0.3 m to 3 m, and field of view 87° × 58°. It works at speed of 90 fps and the error is less than 2% at 2 m (for comparison, the the field of view of RGB camera is 69°× 42° and it operates at 30 fps). The sensor interfaces with the computer using a USB cable. There is no constraint on the type of the projector. To demonstrate the working system, an ordinary projector (Panasonic PT-TW340) was used to display a screen on a suitable surface. The camera was mounted on the side from the screen at the height 1.3 m and at the distance of 0.8 m. The sensor has to be positioned at the side of the projector in such a fashion that fingers as well as the palm of the person who interacts with it are clearly visible for the sensor. The size of used region is 1050 mm × 620 mm. Mutual configuration of sensor and the projector needs to be as shown



(a)



(b)

Fig. 1: The setup configuration: (a) schematic diagram of setup with a coordinate system (blue and red axes refer to the screen coordinate system, black axes – to the camera coordinate system), (b) real experimental setup.

in Fig. 1b. The program running on a personal computer acts as an interface between the depth camera and the projector. It detects the hand and tracks its 3D position.

After evaluating the performance and limitations of the earlier versions of a touch based interactive device [4], [5] some adaptations were made. The earlier implementation was based on VC++ [9] and .NET [4], [5]. However, Python was used in this version as it is widely used and can be used with external hardware. The earlier version was dependent on the Microsoft Kinect SDK, Visual Studio programming environments and etc. which are proprietary. The current version is based on the open source libraries which will ensure easier and cheaper access for the user community and developers. Since there is possibility of on-board processing, it is quite fast and can be used with most of the available hardware. Earlier versions had limitations due to the hardware requirements like high computing power with dedicated USB 3.0 connections, etc. Even the most recently developed interactive device Azure Kinect DK [18] has specialized hardware and software requirements which restrict its use in the long run. It is not easy to run such devices on hardware like Raspberry Pi which have restricted computing power. This is an important consideration for deploying the product on larger scales and was experienced while developing the earlier versions. In conclusion, the proposed version is less dependent on specialized operating systems, specialized hardware, expensive sensors, etc.

IV. Proposed method

In this section we clarify the proposed approach and introduce the used algorithms. In section IV-A we discuss how gesture recognition can be incorporated, section IV-B describes calibration steps, section IV-C contains the information about graphical user interface being used.

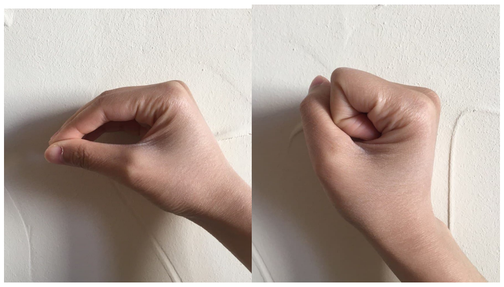

(a) Gesture 1 (b) Gesture 2

Fig. 2: Possible gestures for mouse click action.

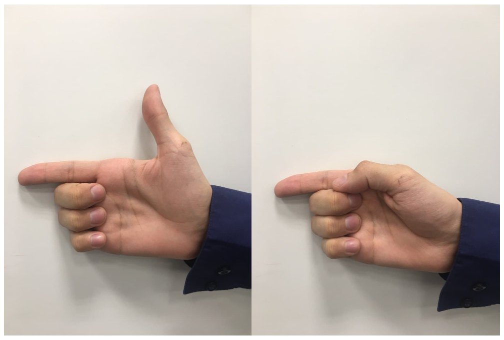

(a) Positioning only (b) Mouse click

Fig. 3: Gesture for mouse click action in the proposed work.

A. Gesture Recognition

One of the popular techniques for extracting parts of the body is skin detection and it is especially suitable for RGB cameras. However, this method cannot be used to find the fingers and their relative orientations, which is important for gesture based interaction. Therefore, an alternate method based on hand landmark is proposed. The Software development Kit: MediaPipe contains a robust model for hand tracking and detection of hand landmarks which has been trained on more than 30000 images [19]. It works based on Single Shot Detector model which implies that the hand landmarks are obtained from a single image itself [20]. This method is more stable and it is less dependent on the illumination in comparison to the skin detection based method. Tracking the landmarks of the hand (there are 21 in the model) allows to add gestures of interest. The choice of the proper gesture is an important step as correctly chosen gesture may improve the accuracy of touch detection. Selecting the gestures on the Fig. 2 will cause the hand occlusions although these gestures can be used easily. These gestures would perfectly satisfy if the camera was placed above the interactive screen. Thus, the positioning of the cursor and mouse click action is assumed to have occurred when the user makes the gestures as shown

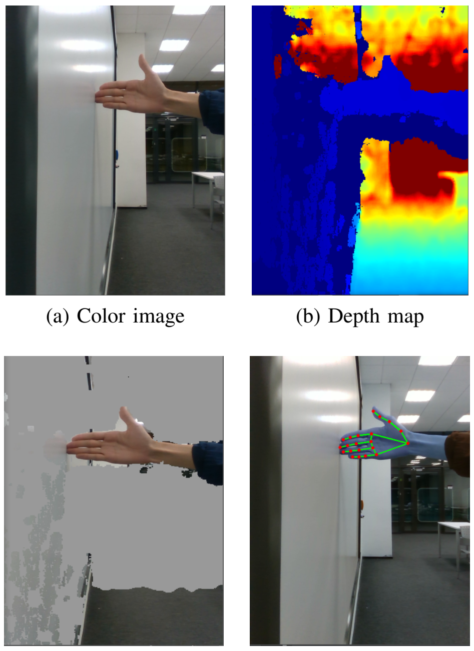

(a) Color image (b) Depth map (c) Image + Depth map (d) Hand landmarks

Fig. 4: Sequence of image conversions for detecting hand landmarks.

Algorithm 1: Detection of mouse click for the method based on the hand landmarks

```
1 Function tryClickGest(dist, threshold):
      Input: Distance between index finger and thumb
             dist (see Fig. 3), threshold distance
             threshold
      Output: Detection if click happened or not
2     if $dist < threshold$ then
3         mouse clicked;
4     else
5         mouse released;
```

in the Fig. 3. Though there are multiple gestures that can be added, this particular one was selected owing to the ease of performing, and the possibility to avoid occlusions that may happen when the camera image contains collinear fingers. These gestures are defined by the specified distance between the fingers of interest. The steps of image processing are illustrated in the Fig. 4. Algorithm 1 describes the process of detecting mouse click in more details.

B. Calibration

The initial step is calibration of the setup [21]. It is used for determining the transformation from the camera coordinate system to the projected screen coordinate system. Once it is done, the sensor feedback can be used to determine the region of the screen that is chosen by the user. Subsequently the visual feedback can be provided at the same location as well. An accurate calibration step is essential for determining the hand position, the choice of the person and for providing the visual feedback at the correct location. The method is described in Algorithm 2.

The calibration steps are the following:

1) Program launch and determination of camera resolution and intrinsic parameters;
2) Initialization of calibration mode, the program waits for hand detection;
3) Recording the coordinates of 4 points using gestures;
4) Transformation of pixel coordinates into the camera coordinate systems.

Algorithm 2: Brief algorithm of screen calibration

```
1  Function screenCalibration(res, intrinsic):
       Input: Camera resolution $res$, intrinsic parameters $intrinsic$
       Output: Calibrated points and size of the screen
2      camera initialization;
3      while number of calibrated points less than 4 do
4          detect the hand position;
5          if touch gesture then
6              write the position into the array;
7          else
8              continue;
9  transform point pixel coordinates into camera
     coordinates using $res$ and $intrinsic$;
10 obtain calibrated points;
11 obtain the size of the screen;
```

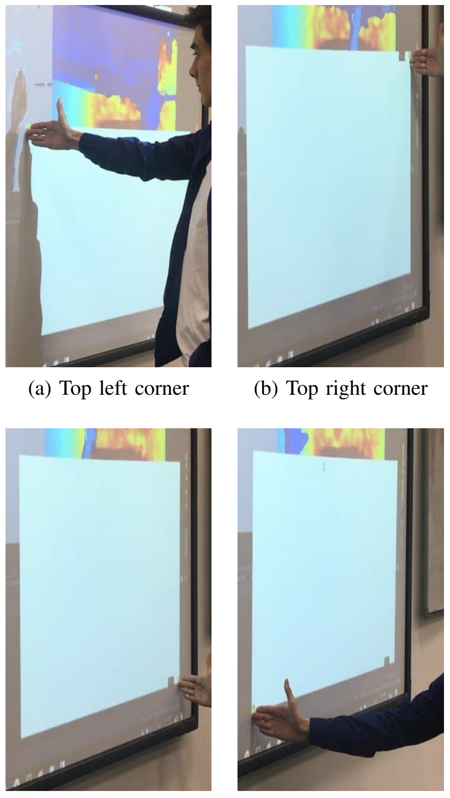
During calibration the user places the finger above the locations indicated on the screen. Later, using the gesture depicted in Fig. 3, the point is saved. The image formation in the camera is represented by:

$$s\mathbf{u} = \mathbf{K}\mathbf{x}, \text{ where } \mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \quad (1)$$

Additionally, the camera intrinsic matrix is denoted by $\mathbf{K}$, image coordinates are represented by $\mathbf{u} \equiv s[u, v, 1]^\intercal$, scaling is represented by s and real world coordinates are represented by $\mathbf{x} \equiv [x, y, z, 1]^\intercal$. Substituting the depth information (i.e. z coordinate) in (1) the coordinates can be calculated as

$$x = \frac{(u - c_x)z}{f_x} \text{ and } y = \frac{(v - c_y)z}{f_y} \quad (2)$$

For each of the four locations represented in Fig. 5, the 3D coordinates are stored. Using this data, the relative pose of the projected screen and the sensor coordinate systems are determined. After this step the interaction of the user can be interpreted and suitable visual feedback actions can be initiated.


(a) Top left corner (b) Top right corner

(c) Bottom right corner (d) Bottom left corner

Fig. 5: Calibration by positioning hand at different extremities of the screen.

C. Interface for Painting Action

For demonstrating the functionality of the system a Graphical User Interface (GUI) for painting was implemented. In this application we track the activity of the users hand such that drawings can be created. The GUI is shown in the Fig. 6. Depending on the user's preference, the color and the size of the output point may be changed by using the gesture mentioned earlier. The algorithm of painter application is described in Algorithm 3. It is worth noting that the calculated coordinates x and z refer to the screen coordinate system (blue and red arrows in the Fig. 1a).

Algorithm 3: Realization of painter game

```
   Input: Camera resolution $res$, intrinsic parameters $intrinsic$
   Output: Updated interface
1  Camera initialization;
2  while not exit do
3      Detect hand position using MediaPipe detector;
4      Transform obtained coordinates into camera coordinates using $res$ and $intrinsic$;
5      Display mouse pointer to the calculated $x$ and $z$ coordinates;
6      Check for mouse click using Algorithm 1;
7      Render updated interface;
```

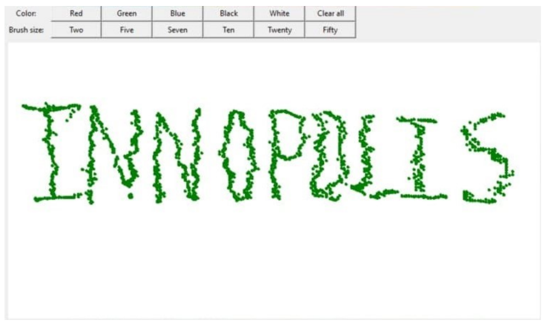


Fig. 6: Graphical User Interface of the painter game with an example of drawn figure. The first row is the set of color buttons, and the second row is the set of brush sizes.

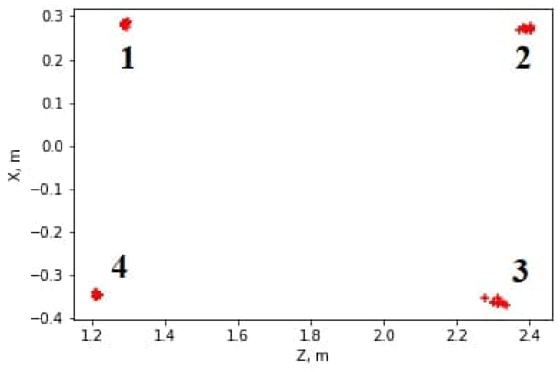
V. Experiments

Multiple experiments were conducted to determine the performance of the proposed method.

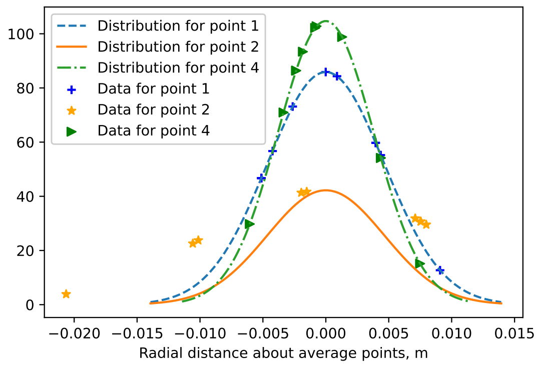
A. Calibration

For estimating the repeatability of the results and the difference between real and computed size of the screen ten calibration experiments were performed. Additionally, experimental situations of "losing" a hand from the frame and its re-appearance is also considered. Apart from this, the change of illumination has also been considered. Figure 7 shows the measured values of the corner points of the screen and the respective distributions with zero mean[1]. The x and z coordinates are taken with respect to middle left part of the drawing screen. The real size of the screen is 1050 mm $\times$ 620 mm. The obtained size of the screen and corresponding errors are presented in the Table I. The values of mean and standard deviation for the measured points are also shown in the Table II.

[1]The computed data shows that Point 3 at the right bottom corner has outliers as visible in Fig. 7a, therefore, its distribution was not included in Fig. 7b.



(a) Measurements used to calibrate the screen



(b) Distribution of measurements shown in Fig. 7a

Fig. 7: Results of multiple experiments with the coordinates obtained relative to the camera reference frame.

TABLE I: Screen size error estimation

Real size, mm^2	Obtained size, mm^2	Relative error of the width,%	Relative error of the height,%
1050 $\times$ 620	1106 $\times$ 627	5.06	1.11

TABLE II: Mean and standard deviation for calibrated points

Point	Mean, mm	Standard deviation, mm
1	1320.15	4.65
2	2410.87	9.45
3	2340.98	15.16
4	1261.44	3.81

B. Accuracy of Detection

In order to estimate the effectiveness of touch detection 200 touches for each method were performed by several participants. They were asked to move hands at different speeds of motions and at different angles with respect to the camera. The method using skin detection showed 71% of accuracy while the method based on hand landmarks detection showed 92.5%.

C. Detection Time

Besides the comparison of detection accuracy, it will be interesting to compare how fast each of the proposed methods detect the hand. For this task the time differences between methods based on skin detection and hand landmarks were computed. Total number of hand detections is 374. The experiments showed that MediaPipe model works faster than an alternative system based on skin tone detection and the average time difference is 18 ms, meanwhile the average absolute values are 53 ms for skin-detection approach and 35 ms for hand landmarks detection approach respectively. And for the user difference in 18 ms will be noticed especially in the usage of touch and touch-less screens. Modern manufacturing companies are trying to reduce the latency for more comfortable use of devices. Therefore, we believe that the study of this parameter plays an important role in the development of a real-time systems.

VI. Discussion

The experimental results gave some insights into the performance of the proposed approaches. The performance of calibration showed that the standard deviation in measurements are within 10 mm (except for one point). The maximum error of 15 mm was observed at the farthest point and minimum error of 3.8 mm was observed at the point closest to the sensor. For calibration the relative error of the width and height are 5.06% and 1.11% respectively. The results are small compared to the large size of the screen and are better than the results from the method proposed in [4] wherein the variation was much higher (greater than 10 mm). Also there is some error in calibration (2-3 cm), which is acceptable for our implementation once again due to the large screen size. The permissible level depends on the application that is developed. The earlier existing system [4] involved interaction with only 5 keys on the screen. However, in the case of painting GUI the interaction is possible throughout the whole screen, requiring high accuracy. The time taken to detect the hand is less than 1 second in case the tracking is lost. Considering average values, hand landmark based model could detect the hand faster than the skin tone based method by 18 ms.

With regards to the accuracy of detection, the hand landmark based detection performed better. The drawback of skin detection is that it cannot detect all skin tones. However, there may be degradation in the performance of the method based on hand landmark if there is occlusion of the fingers.

In the proposed application, i.e. touch detection, only one particular gesture was utilized and the results are corresponding to this gesture. However, it is possible to use multiple other gestures as well according to the application, user community, etc. In this case detailed experiments may be required for quantifying the accuracy in positioning.

VII. Conclusions

In this work the RealSense camera-based interactive system is designed and implemented. The method is based on hand landmark detection. This method has many advantages compared with the alternative method based on skin tone. It has less dependence on illumination and skin tone variations. Furthermore, the results of experiments show that the proposed method is rather robust and it is more sensitive to the hand motions rather than skin detection method (i.e., 75% against 92.5%). It is more stable and possible to be used as an alternative for mouse movements. The proposed method gives the opportunity to invoke different gestures for interacting with the screen without performing the actual touch. It enables the touch-less interaction in contrast to the touch based interaction envisaged by the earlier versions [4], [5]. Compared to these systems the uniqueness of the system is that it is possible to interact in a touch-less fashion, detect different kind of fine gestures involving fingers and use of sensor with higher accuracy and speed. A demonstration of the working model is available at https://youtu.be/B13jlNQG7s0.

References

[1] L.-H. Lee and P. Hui, "Interaction methods for smart glasses: A survey," *IEEE access*, vol. 6, pp. 28 712–28 732, 2018.

[2] C. Stephanidis and M. Antona, *Universal Access in Human-Computer Interaction: Aging and Assistive Environments: 8th International Conference, UAHCI 2014, Held as Part of HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014, Proceedings, Part III.* Springer, 2014, vol. 8515.

[3] Intel, "Depth camera d435," https://www.intelrealsense.com/depth-camera-d435/, 2021.

[4] P. Sharma, R. Joshi, R. A. 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.

[5] R. A. Boby, R. Prakash, S. Saha, T. Matsumaru, P. Sharma, and S. Jaitly, "Calibration and statistical techniques for building an interactive screen for learning of alphabets by children," in *International Journal of Advanced Robotic Systems*, vol. 14(3), 2017.

[6] T. Matsumaru and K. Akai, "Step-on interface on mobile robot to operate by stepping on projected button," *The Open Automation and Control Systems Journal*, vol. 2, pp. 85–95, 2009.

[7] T. Matsumaru, "A characteristics measurement of two-dimensional range scanner and its application," *The Open Automation and Control Systems Journal*, vol. 2, pp. 21–30, 2009.

[8] T. Matsumaru, Y. Jian, and Y. Liu, "Image-projective Desktop Arm Trainer IDAT for therapy," *The 22nd IEEE International Symposium on Robot and Human Interactive Communication (IEEE RO-MAN 2013)*, pp. 501–506, 2013.

[9] Y. Liu, Y. Jiang, and T. Matsumaru, "Development of image-projective desktop arm trainer, IDAT," *IEEE/SICE International Symposium on System Integration (SII)*, pp. 355–360, 2012.

[10] A. Agarwal, S. Izadi, M. Chandraker, and A. Blake, "High precision multi-touch sensing on surfaces using overhead cameras," in *Horizontal Interactive Human-Computer Systems, 2007. TABLETOP'07. Second Annual IEEE International Workshop on*, 2007, pp. 197–200.

[11] J. Dai and R. Chung, "Making any planar surface into a touch-sensitive display by a mere projector and camera," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, 2012, pp. 35–42.

[12] J. Hu, G. Li, X. Xie, Z. Lv, and Z. Wang, "Bare-fingers touch detection by the button's distortion in a projector–camera system," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 4, pp. 566–575, 2014.

[13] L. Zhang and T. Matsumaru, "Near-field touch interface using time-of-flight camera," *Journal of Robotics and Mechatronics*, vol. 28, no. 5, pp. 759–775, 2016.

[14] K. Horiuchi and T. Matsumaru, "Short range fingertip pointing operation interface by depth camera," in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO).* IEEE, 2018, pp. 132–137.

[15] A. Cadena, R. Carvajal, B. Guamán, R. Granda, E. Peláez, and K. Chiluiza, "Fingertip detection approach on depth image sequences for interactive projection system," in *2016 IEEE Ecuador Technical Chapters Meeting (ETCM)*. IEEE, 2016, pp. 1–6.
[16] Microsoft, "Download azure kinect body tracking sdk," https://docs.microsoft.com/en-us/azure/kinect-dk/body-sdk-download.
[17] Intel, "Skeleton tracking sdk," https://www.intelrealsense.com/skeleton-tracking/.
[18] Microsoft, "Azure kinect," https://azure.microsoft.com/en-us/services/kinect-dk/, 2021.
[19] Google, "Mediapipe hands." [Online]. Available: https://google.github.io/mediapipe/solutions/hands.html
[20] Google, "On-device, real-time hand tracking with mediapipe," https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html, 2019.
[21] B. Huang, Y. Tang, S. Ozdemir, and H. Ling, "A fast and flexible projector-camera calibration system," *IEEE Transactions on Automation Science and Engineering*, pp. 1–15, 2020.