Realsense based system for tracking Hand Pose*

1st Nilesh
Mechanical Engineering Department
Indian Institute of Technology Jodhpur
Jodhpur, India
Email:- nilesh.2@iitj.ac.in

2nd Dhruv Gupta
Materials Engineering Department
Indian Institute of Technology Jodhpur
Jodhpur, India
Email:- gupta.77@iitj.ac.in

3rd Riby Abraham Boby
Mechanical Engineering Department
Indian Institute of Technology Jodhpur
Jodhpur, India
Email:- riby@iitj.ac.in

4th Suril Shah
Mechanical Engineering Department
Indian Institute of Technology Jodhpur
Jodhpur, India
Email:- surilshah@iitj.ac.in

***Abstract*—This article focuses on the implementation details of a project based on the Human Machine Interface(HMI) device. Computer vision techniques are widely used in developing human-machine interfaces. We developed a hand tracking pose system using Intel's RealSense Depth Camera d415. The system is designed to accurately capture and track the movements of the human hand and gestures in real-time, enabling intuitive and natural interaction with virtual environments using Mediapipe model. The camera calibration is done to capture projected screen accurately. Aim of the project will be to develop a interaction with projected screen more accurately and a cheaper system (compared to Boby et al. 2017).**

It was developed to teach alphabets to children. We will make compact and efficient version (compared to existing versions) of the human machine interface using Intel Realsense camera D415. The human hand is an incredible gift that enables us to interact with the world around us. However, for some individuals, hand mobility may be impaired due to disabilities. This is where we thought to modify the project and present a solution in this project.

I. Introduction

The goal of HMI development is to create interfaces that are intuitive, easy to use, and efficient, and that provide users with the information they need to operate the system effectively.
3-D camera based interactive touch-screen are becoming more interestingly. One of the main benefits is that it eliminates the need for auxiliary devices like glasses or trackers, which can be uncomfortable for the benefit of the users. Those with impairments or mobility challenges may find touch-less screens useful because they can operate the device without using their hands. In order to develop HMI an interactive touch-less screen with 3-D camera on it can be very useful in various fields. It can also be used in education to make learning process more interactive and fun for students. A large touch screen can be very expansive, so we uses a 3-D Camera with a projector to make it interactive at a low cost, which can be easily transported. This motivates us to develop a touch-less screen. In this article we will be defining our work on developing an interactive touch-screen game with the help of Intel Realsense depth camera and the projector. This article focuses on the technique used for camera calibration, taking input from the Intel Realsense camera, identifying human hand from the input feed, integrating the projected screen with human hand, creating a game by which children can learn letters. The calibration technique is automated, it don't need any human intervention. We extended work of not only using hand gestures but a person with hand disability(he must have at-least one finger working correctly) can also interact with screen.

II. System-Setup

Figure 1A depicts the skeleton of the system quite accurately. The sensor Intel RealSense D415 is the main part of the system, and it has a depth measuring range of 0.3m to 5m with a field of view of 65° 40°. It is accurate to within 2% at 2 metres and runs at a smooth 30 frames per second. A USB cord connects this sensor to PC. A standard classroom projector was utilised to project a screen on the wall and demonstrating the functionality of the device. For the sensor to work properly, it must be placed to one side of the projector in a way that allows it to see the user fingers, palm and screen.

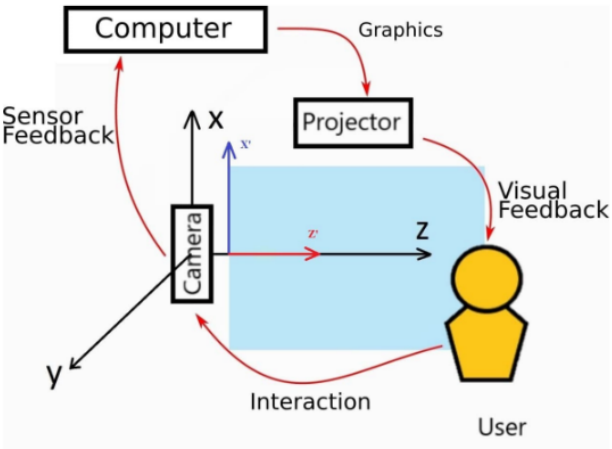

Fig. 1. (A) Setup configuration

The software running on computer acts as an interface between the Intel Sensor and the projector. It can identify the hand and follow its 3D movement. Python was chosen because it is popular and can be used with many kinds of hardware. As it is built on top of open source libraries, the current release will be more accessible and cost-effective for the entire user base and development team. Since on-board processing is an option, it's quick and compatible with a wide range of devices. The suggested approach requires less specialized operating systems, specialized hardware, expensive sensors, etc.

III. METHOD

The working of Intel Realsense camera and camera calibration technique has been studied. The hardware component of Intel RealSense consists of a depth sensor, which uses an infrared light projector and a stereo camera to capture depth information. For calibration, image processing technique is used with the help of openCV.

We have explained our method in several sections: A) Screen Capture and calibration B) Hand recognition C) Finger Recognition D) Calibration E) Working of game

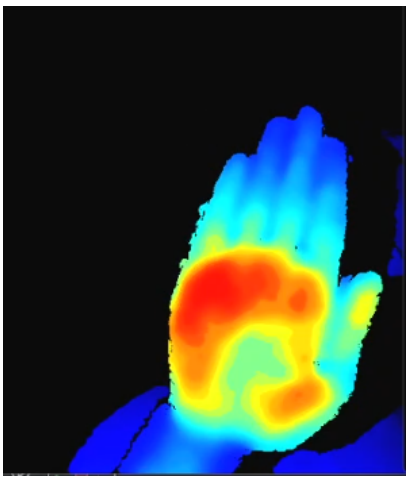

Fig. 2. Depth image

A) Screen Capture

A program was developed to get real time image, depth data using the Intel realsense sensor and store images. The 1st step to develop full game was calibration of screen with camera. As game start the camera initialize and captures images, from there it extract out the region of screen and remove other things from images, this is explained in calibration part and then we have to track hand movement in space and modify it for disable person to get hand's coordinates from the sensor. For this we are using Mediapipe library. Using co-ordinates of landmarks of hand we project them over screen and match the pixel values and apply condition to play the game on the projected screen by using distance between screen and hand with setting a threshold value of 10cm.

Algorithm:-

Import modules
# Define the frame width and height
h = 480
w = int(r*h), r is standard screen ratio = 1.33333
% Camera Initialization
dc = DepthCamera()
While loop(1):

1. Depth_frame & Color_frame = Get_frame(dc)
2. show_image(color_frame)
3. Save image - cv2.imwrite('image")

Processed for calibration

Fig. 3. Save images

B) Hand recognition

Google created the well-known machine learning framework called Mediapipe, which comes with a number of pre-trained models for computer vision tasks including hand tracking and position estimation. BlazePalm is a real-time hand detector machine learning algorithm it forms the basis of Mediapipe's hand tracking concept. BlazePalm employs a deep neural network to identify hands in input images and determine the bounding box of each hand. Following a hand detection by BlazePalm, the cropped hand image is sent to the Hand Landmark Model, a different deep neural network. The 21 hand landmarks (finger tips, knuckles, and wrist) in the image are estimated using this model's 3D coordinates. The x, y, and z coordinates of the 21 landmarks can be used to track the hand's movement and pose in real time.

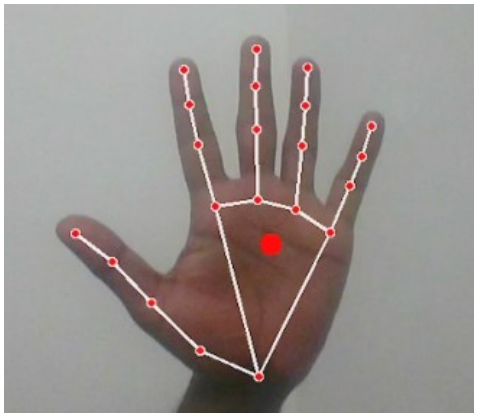

Fig. 4. Hand Landmarks

C) Finger recognition

In this algorithm we are detecting the finger used by user to interact with the system in real time, user can use any single finger or complete hand(all finger open) to interact with the screen and this algorithm will automatically detect the hand gesture and will act accordingly to give precise selection on the screen. If user uses complete hand then the average of the coordinates of the landmark 0, 5, 9, 13, 17 is taken and the landmark of tip of finger if single finger is used for selection. The algorithm is fully developed by us without support from any article or paper from internet. We have observed that for every finger there are 4 landmarks 1st, 2nd, 3rd, and 4th and if we consider the distance between them, we will get a relation between open and closed finger. A finger is open if distance between (1st, 3rd) less than (1st, 4th) or else it is closed.

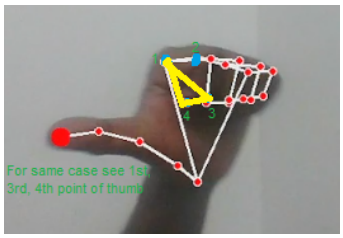

Fig. 5. Algorithm with Image

The algorithm is as follows:-

```
Algorithm:-
1. LandmarList = [Get using ML model]
2. Define distance fn
3. Define average fn
4. Define finger_pointing fn:-
      List = LandmarkList[1, 5, 9, 13, 17]
      for i in List:
          a = distance between 1st, 3rd, 4th point of every finger

      for i in length(a):
          if distance between (1, 3) > (1, 4) point on every finger:
              That finger is used by user
          if more than two fingers are open take average
```

Fig. 6. Algorithm for finger detection

D) Calibration

The input feed that we get will have the projected screen and its neighboring environment. We have to find the pixel coordinates of the projected screen in the video input that we get from the realsense camera to define the precise region, where the screen is present in the input feed by the camera. We will take the pixel coordinates of the corner of the projected screen. The game window is rectangular in shape, but its projection will be distorted because the wall and projector will not be at right angle precisely, resulting in the extension of one side of the projected screen. The realsense camera will also be kept at some obtuse angle with respect to the projected screen, which will lead to extension of the lower side of the projected screen and compression of the upper side(figure 7). So as the result of these factors the image of the game screen

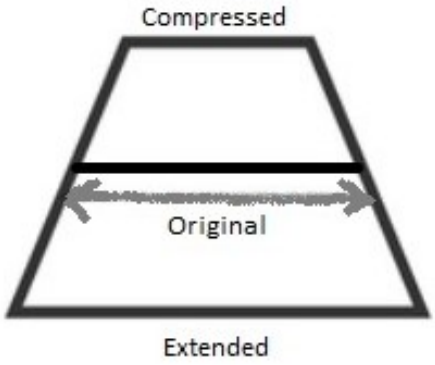

Fig. 7. screen distortion

received by camera as input will be a quadrilateral rather than a rectangle. We have to find the coordinates of the vertex of the quadrilateral for the precise touch control. For achieving this we used image processing technique. Our main aim is to isolate the game screen in the input by camera, so for that, as a calibration step, we had projected a red, blue and green game screen one after the other and used image processing and color recognition to find the exact coordinates of the vertex of the quadrilateral. We had used opencv for that. First we have captured an image from the input feed of the camera, the input image is converted to HSV format as opencv uses this format.two ranges, one upper and one lower is defined for every color(red, blue, green) in HSV format. These two ranges correspond to the light and dark shade of the set color. These ranges are used to define a mask over the original HSV image. Two masks are defined for two ranges. Masks have the set color region as white and the other color region as black. We merged the two masks to get a combination of both. As a result we get a grayscale image whose set color region is white and others are black. This image may contain some white region other than screen as they may be present in the neighboring environment, which is captured by the camera, so it needs a filtering process that removes other unnecessary white regions which will definitely pose a problem while further processing. For that we have identified the largest white area present in the image and preserved it and rest other regions are converted to black. For that we have used contours in opencv to identify that region, created another mask with small regions as black, this gives us an grayscale image with only one white region. These images are then refined to make the boundaries between black and white clear, using filters in opencv with suitable parameters. Then we used the 'findcontours' feature of opencv and approxPlyDP to find the coordinates of the corner of the quadrilateral. This process is repeated for each of the three

colors and the averaged value of the result is taken, it reduces the chances of error and gives us a precise result. Then we will store the depth value of all the coordinates in the screen region using the realsense camera. Initially we were taking it for each coordinate but later to reduce time, we took data for one out of three pixel and approximated the remaining on the basis of measured ones.

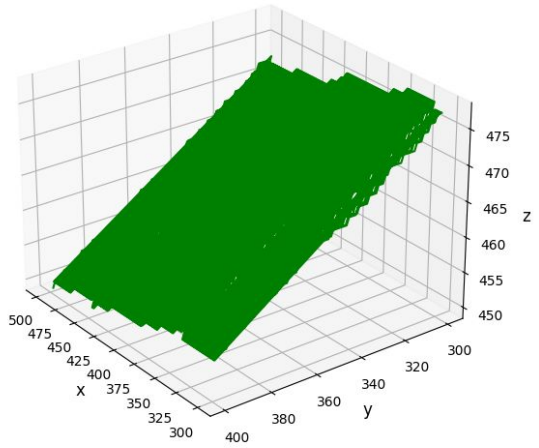

Fig. 8. Result

Figure 7 gives us depth data with respect to pixel coordinates for the required region. This will be used further for detecting the touch. As a result of this we have an auto calibration technique, it doesn't need any human intervention. This calibration technique is based on the fact that the camera and the projected screen will not be displaced from their position. If they are displaced during or after calibration it will not produce precise results and will require re-calibration.It will require less than a minute for calibration. Now we have the region where the game screen is present in the input feed from the camera. The game has six regions where we have different letter placed. So to find the coordinate range where the letter block lies, we had processed these four coordinates to find the coordinate range of each letter block using simple mathematics based codes.

E) Working of game

After the calibration is completed our game starts(figure 8), it announces which letter to choose and the user can respond accordingly. If the user selects the correct letter it provides another puzzle otherwise it will ask the user to repeat. Now to identify when the user touched the block, we will match the depth value of the finger to the depth value of the game screen which we had stored earlier, if they are close enough we will consider the user is touching it.

Fig. 9. Game interface

IV. RESULTS

The Intel Realsense Camera d415 is an old version also it's depth sensors does not work properly due to which the accuracy has been decreased. We find out that the depth sensor doesn't work in frame region (x - 0, 100) and (y - 0, 50). So we need to look up frame and calibrate accordingly. The method of hand tracking using Media Pipe is very accurate and fast technique. As this model is trained over 30000 images. As we modified the program for disabled person, the accuracy in case of finger detection is about 70%. Figure 8 shows use accuracy of screen we detected. The height difference of a plane is about 25px. The calibration technique is working fine with an accuracy of 80%. The accuracy of the calibration system largely depends on the wall color on which we are projecting. If the wall is a very bright color it may lead to some inaccuracy while calibration, though, generally projection is done only on light coloured walls or on the white projection screen. The accuracy of the touch system largely depends on the lighting conditions of the room and the brightness of the projector. A very bright projector may lead to inaccuracy as the IR of the camera will not be able to work properly.

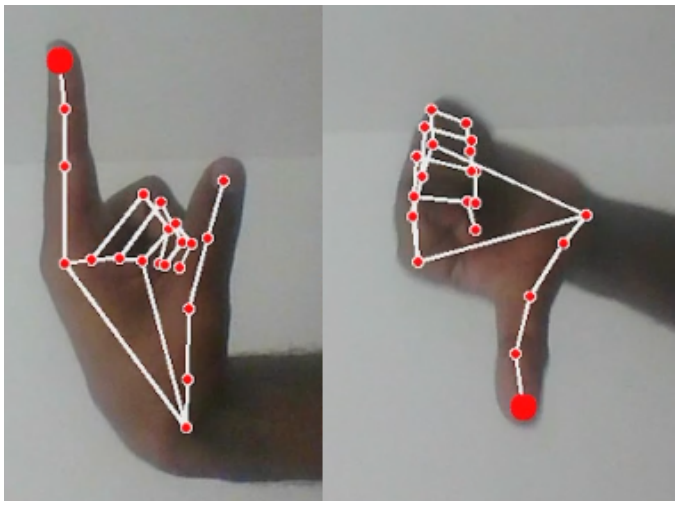

Fig. 10. Result(Finger Recognition)

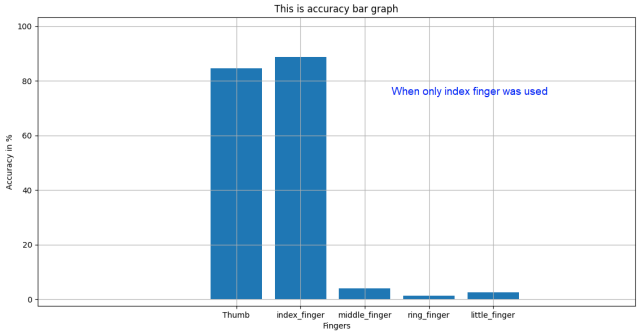

Fig. 11. Result(Finger Recognition accuracy)

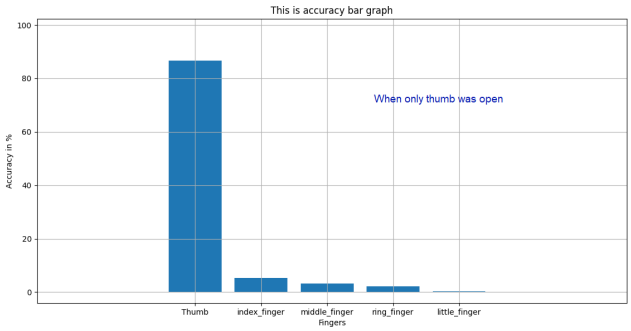

Fig. 12. Result(Finger Recognition accuracy)

Rest results and Videos are also uploaded on git hub. Source code for this project is available on GitHub: **Link**

V. Conclusion

We had developed a virtual touch screen with the help of Intel realsense depth camera, that helped us in playing the word game. This simple game, integrated with this technique will be an innovative and very interactive way to teach kids words and it can be extended to any language. With the help of the Intel depth camera, hand and finger detection, auto-calibration its use cases can be extended to other practical applications which require virtual touch screens or large, costly touch screens, which can be replaced by this to save cost. The proposed method gives us a unique way to define the hand gestures and enables for use of disabled person. Its distinguishing features over similar systems include touch-less interaction, object detection, and emotion recognition precision and speed with the use of delicate finger motions and a sensor.

VI. References

- "A depth camera-based system to enable touch-less interaction using hand gestures" by R Damindarov, CA Fam, RA Boby, M Fahim, A Klimchik, T Matsumaru
- "Human-Robot Interaction Using RealSense Camera" by S. Saha, et al. This paper presents a system that uses the Intel RealSense Camera for hand tracking and gesture recognition in human-robot interaction scenarios.
- "Projectable interactive surface using microsoft kinect v2: Recovering information from coarse data to detect touch" by Prayag Sharma, Ravi Prakash Joshi, Riby Abraham Boby, Subir Kumar Saha, Takafumi Matsumaru
- "MediaPipe a Open souce ML model by Google Link
- Color recognition link
- Image identification using openCV link