School of Computer Science and Engineering

HCI GAME USING HAND GESTURES

Human Computer Interaction

CSE4015

Slot: A1

19BCE2460	**Sagar Sethu**
19BCI0274	**Heet Thakrar**
19BCE2266	**Yoshita Sai Madamala**

Under the guidance of,

Prof. Arun Kumar S

Vellore Institute of Technology

2020-21

Abstract:

- ### Aim:

  To create a human computer interaction video game which can be played with the help of hand gestures. Gesture recognition based interactions, provide a more realistic and immersive interaction compared to traditional peripherals.

- ### Objective:

  Primary objective of the game is to provide a fun and interesting way of playing a video game. The game controls are much more natural and adaptable than the usual keyboard controls. Using hand gestures also promote physical activity which improves health and general wellbeing as opposed to traditional keyboard controls.

- ### Proposed Methodology:

  The player would play the game with the help of his/her fist or palm. When the palm is gestured up, the character moves up and vice versa. The computer's front camera is the sensor which detects motion.

- ### Expected Outcome:

  Designing a game where random obstacles would be created for the player who must dodge them in order to

achieve a high score. There will be a timer when the game starts and it stops when the player collides with any of the obstacles. This game will be different from others due to the interesting and easy method of controlling the character in the game. The game is made to be user friendly, so that users from all sections of society will be able to easily play the game without any prior practice or learning of controls as the controls are the natural human movements.

Introduction:

I. Background and motivation of the project:

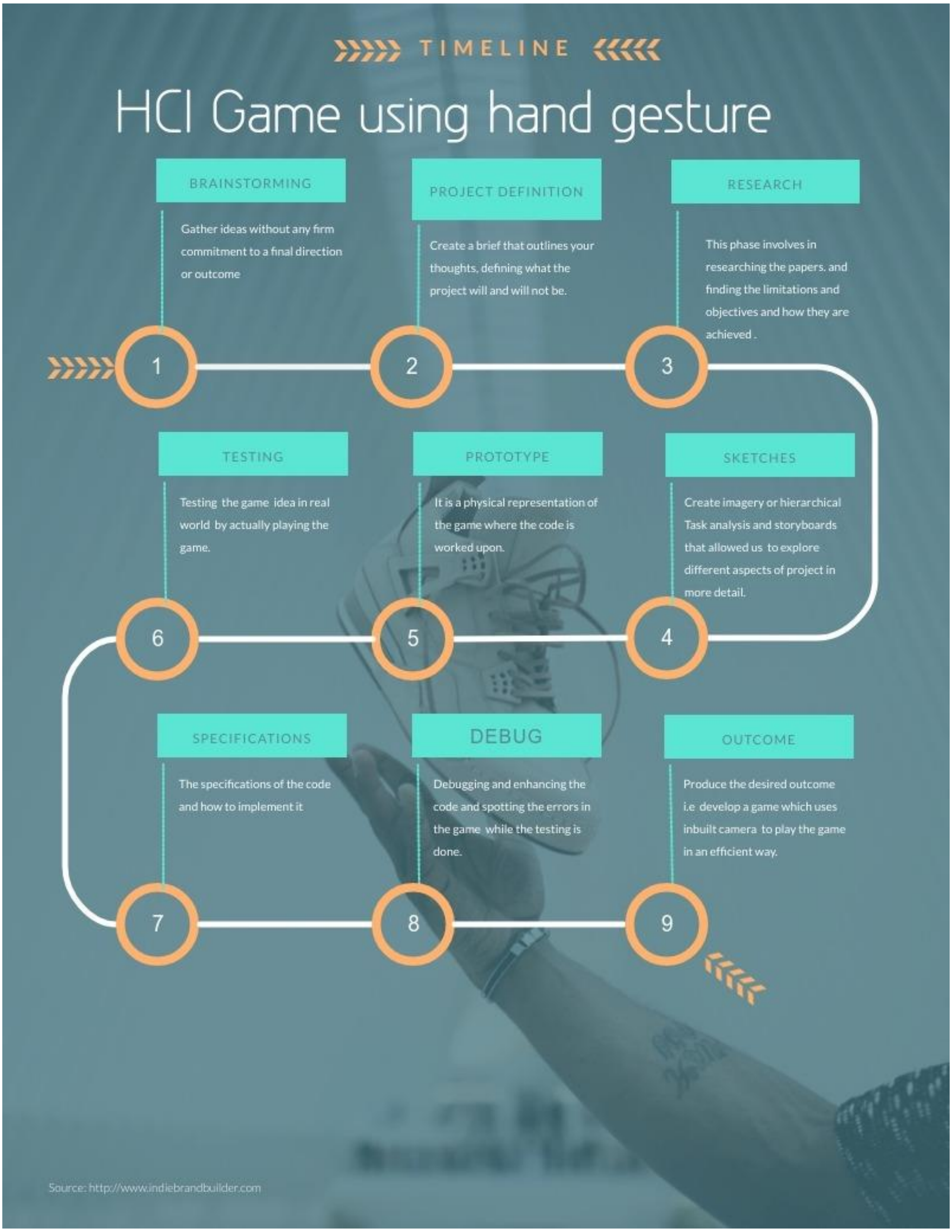
Hand gesture recognition systems provide users an enhanced interaction experience as it integrates the virtual and the real world object. Gesture recognition based interactions, provide a more realistic and immersive interaction compared to traditional peripherals. The gesture based interaction interface showcased here can be applied towards many applications like virtual reality, communication techniques and Games. The focus of our project is on games as the application domain for this interaction method.

Gestures, particularly hand gestures are also faster and possibly could be more accurate than using the keyboard – mouse combination of peripherals. The non-touch system is a modern method of computer-interface technology capable of revolutionizing human-computer interaction.

II. Project Novelty:

Our hand gestures are easy to use and learn for first time users. The hand gesture recognition system can be used in any game which has the same key binds. No need to alter or have different implementation for these games. It also provides a more fun and exciting way to play games as well as help the disabled people who have issues with utilizing the traditional methods of input. This form of human – computer interaction definitely is improving every day and has a lot of potential in the future for development and usage in everyday life. Key advantage here is that games can be played in a virtual space through the same movements in the real world, without installation of special controllers.

Project Timeline:



Related Literatures:

Paper Title	Gesture Recognition using Microsoft Kinect
Year of publishing	2014
Author names	K. K. Biswas
Objective	A method to recognize human gestures using a Kinect® depth camera.
Technique used	• The depth sensor consists of an infrared laser projector combined with a monochrome CMOS sensor • The depth map is visualized here using colour gradients from white (near) to blue (far) **II. FEATURE EXTRACTION** • CLAP: Clapping • CALL: Hand gesture to call someone • GREET: Greeting with folded hands • WAVE: Waving hand • NO: Shaking head sideways – "NO" • YES: Tilting head up and down – "YES" • CLASP: Hands clasped behind head • REST: Chin resting on Hand ***Pre – processing*** • Isolate the human making the gestures from the background scene. This is done by background subtraction from the depth image of the scene. This was done by using auto thresholding *B. Features from ROI* • *A region of interest (ROI) is created by placing a 14x14 grid on the extracted foreground. The gesture is parameterized using depth variation and motion information content of each cell of the grid.* *C. Training and Testing* • Each frame of the video was represented by a row of the matrix. The columns represent the feature points
Limitations	• Study is limited to minimalistic movements. • The method is not compute intensive, as very few calculations are involved to extract the features • The accuracy of the results could be improved by making use of the skin colour information of the colour camera. The method is not compute intensive, as very few calculations

Paper Title	Depth Camera Based Hand Gesture Recognition and its Applications in Human-Computer-Interaction
Year of publishing	2011
Author names	Zhou Ren, Jingjing Meng, Junsong Yuan
Objective	• Compare the performance in terms of speed and accuracy between FEMD and traditional corresponding based shape matching algorithm, Shape Context. • Introduce several HCI applications built on top of an accurate and robust hand gesture recognition system based on FEMD.
Technique used	**Vision-based hand gesture recognition methods can be classified into two categories:** • ***Machine Learning based approaches***: For a dynamic gesture, by treating it as the output of a stochastic process, the hand gesture recognition can be addressed based on statistical modelling, such as PCA, HMMs • **The second category is *Rule based approaches*:** Rule based approaches consist of a set of pre-encoded rules between feature inputs, which are applicable for both dynamic gestures and static gestures. ***Hand Detection*** ***Gesture Recognition*** **Performance Comparison** • compare the mean accuracy and mean running time between FEMD based hand gesture recognition system and Shape Context shape matching algorithm **Applications** • ***Sudoku game****: The user selects a square by hovering his hand over it and pushes once. He/she then commands a number to be filled into the square by performing the corresponding hand gesture*
Limitations	• traditional hand gesture recognition methods all applied restrictions on the user or environment because of the limitations of the optical sensors • Use of 3D features is limited to a few studies. Structured light was used to acquire 3D depth data. • Because of the limitations of the optical sensors, the quality of the captured images is sensitive to lighting conditions and cluttered backgrounds, usually not able to detect and track the hands robustly, which largely affects the performance of hand gesture recognition.

Paper Title	**Design and implementation of a flexible hand gesture command interface for games based on computer vision**
Year of publishing	2009
Author names	João L. Bernardes Ricardo Nakamura Romero Tori
Objective	Describes a command interface for games based on hand gestures defined by postures, movement and location.
Technique used	• Gestures2Go's primary requisite is to allow their use to issue commands. Issuing commands is a very important task in most games, usually accomplished by pressing buttons or keys. **3. HCI and Game-specific requisites** • Gesture-based interfaces are almost always "invisible" to the user • It consists of an abstract framework that divides the system in modules and defines the interface between these modules and, currently, of a single, simple implementation of this framework. **4.1 The Abstract Framework** • *G2gGesture* is responsible for the gesture model, while *G2gAnalysis* and *G2gRecognition* define the interfaces for the classes that will implement gesture analysis and recognition • *G2gFeatureCol* is a collection of *G2gFeature* objects. ***4.2 Implementation*** • *The requirement analysis pointed that an implementation of the abstract framework described above specifically for games characteristics:* • *minimum need for setup,* • *low processing demand* • *a high number of possible gestures* • *tolerance to variations in the execution of gestures* • *allow multimodal interaction* • *Make development of games using gestures as easy as possible.* **5. Tests and Results** • The first priority was to verify the posture analysis and recognition strategy
Limitations	• The segmentation needs to be improved, the less robust part of the system and causes frequent and noticeable errors under some lighting conditions. • Methods often use depth data to aid in segmentation. Other methods do not require a known model for the hand but only track its position, not the contour, which is necessary for Gestures2Go. • Using depth data is another planned improvement to the system, both to the segmentation and to allow a greater number of postures, such as pointing postures • Formal usability tests must be conducted to determine whether the interaction techniques using Gestures2Go in a MMO are effective in the context of games.

Paper Title	**An Interaction Educational Computer Game Framework Using Hand Gesture Recognition**
Year of publishing	2012
Author names	Kai Zhang, Ying Zhai , Hon Wai Leong , Shengming Wang1
Objective	Ease the game development so that developers can easily use it to produce educational computer games for end-users.
Technique used	**1. Hand Gesture Vocabulary** • Our framework targets to be used to produce the educational computer games with hand gesture interaction for preschool children. Hence we first need to define a hand gesture vocabulary for this purpose. • We establish a hand gesture vocabulary closely related to preschool children education, so that children can answer simple questions using these hand gestures **2. Framework Overview** • Design a game authoring tool named authoring GUI in which the game script and game component are created. **3.Hand Gesture Recognition** • The recognition component is responsible for gaining images from camera, segmenting hand gestures from images, and finally recognizing hand gestures. • *Segmentation* • *Hand Gesture Feature Extraction* • *Recognition* **4.. Experiments on Recognition Rate** • Experiments to test our hand gesture recognition rate for five types of hand gestures. For each type of gestures we do 50 times to recognize them to find the average recognition rate
Limitations	• improve the design of the framework and implement the whole framework • how to enhance the authoring GUI of the framework in order to lower the entrance difficulty of using the framework • Build more power of the content libraries of the framework.

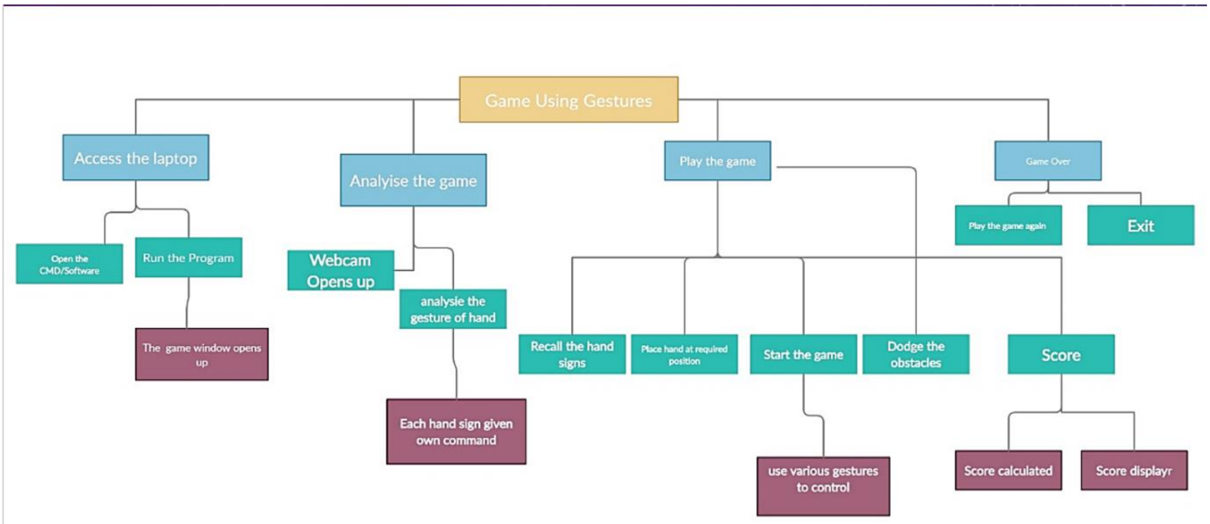
Paper Title	**Real time Hand Gesture Recognition using a Range Camera**
Year of publishing	2009
Author names	Zhi Li, Ray Jarvis
Objective	A real time hand gesture recognition system. The approach uses a range camera to capture the depth data
Technique used	1. **Range Camera** A 3-D range camera is employed as our apparatus. It delivers the depth data of objects in its view at every pixel at a high frame rate 1. **Depth data processing**: there is significant amount of noise in both depth and intensity data from the sensor 2. **Error regions with high velocity** **2.Hand segmentation and trajectory extraction:** • Skin colour information is traditionally used to locate and segment hand region **3.Hand gesture recognition** • **Hand shape analysis:** • A database needs to be established where a large set of images containing various hand patters are recorded with known labels in the training stage. **4.3D hand trajectory recognition** • A hand trajectory is also expressive in an interaction. Intentions are naturally expressed by movement rather than static hand object. **5.Experiment and results** • The rotation of the hand both around horizontal and vertical axes may make the appearances of the hand different. • Difficult to achieve viewpoint independent performance because of self-occlusion from the single view, the slight rotation in space should not affect the recognition output
Limitations	• limited resolution of the sensor chip in depth data processing • The corresponding hand position in the range camera in order to find its depth value. • Imposes extra burden of alignment between the web camera and the range camera. • The colour based method may locate the hand position incorrectly. • If two persons appear on the screen it causes error. • Overexposure and underexposure will also results in errors in the depth data.

Proposed Methodology (Framework):

➢ Proposed Work:

The main domain of the project lies in developing a game with the application of gesture recognition system. The usage of hand gestures to promote virtual activity as one does in real world, results in the main advantage that the game can be played in a virtual space with enhanced interaction much better than conventional peripherals.

➢ Method and approaches:



When the user runs the program the game in command prompt window opens up and accesses the webcam of the user. Once the game starts, the player needs only to move his hand to make his game character move and dodge the obstacles and score points.

If the user fails to dodge any obstacle the timer stops and displays the time and total score with a message "GAME OVER". The user is at his will to begin the game again or to quit the game.

Tools Used for Development

- PyCharm python IDE – This ODE is chosen over the rest because it is cross-platform and for accessing smart built-in developer tools, scientific tools and customizability.
- Webcam drivers – It is used to recognize the webcam associated with that device and for proper functioning of the webcam.
- Python libraries:

  >*Tensorflow:* TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow. It was created and is maintained by Google and released under the Apache 2.0 open source license. The API is nominally for the Python programming language, although there is access to the underlying C++ API.

  >*OpenCv:* The given library is used to access the webcam when the code is run for gesture recognition.

  >*Multiprocessing:* Multiprocessing is a package that supports spawning processes using an API similar to the threading module. The multiprocessing package offers

both local and remote concurrency, effectively side-stepping the Global Interpreter Lock by using subprocesses instead of threads.

>*Cv2:* OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

>*Nes py:* nes-py is an NES emulator and OpenAI Gym interface for MacOS, Linux, and Windows based on the SimpleNES emulator.

➢ Metrics and Measurement:

Importance of this project-

The proposed project has a promising future of the forth coming pro-digital era as currently the game focuses on breaking the norms of current gaming industries by hand gesture recognition system.

The method chosen i.e. hand gesture recognition would provide better indulging gaming due to better physical enhancement and a more realistic interaction between the user and the interface.

The project requires no large space for playing and proves its adaptability and is highly user friendly nature. Thus, the game can even be played in a virtual space by the player.

Due to the involvement of physical activities the user will have better health and will not lose his physique because of playing, in the long run.

- *Comparison with Existing Models/Methods-*

Existing methods or models currently which are played on laptop majorly uses keyboard and mouse. Gestures provide the user with a new form of interaction that mirrors their experience in the real world. They feel natural and require neither interruption nor an additional device. Furthermore, they do not limit the user to a single point of input, but instead offer various forms of interaction.

The keyboard, mouse etc. lack the sensitivity desired in required application. Eventually the researchers working the area of Human Computer Interaction made a common emphasis to design and develop the user interfaces capable enough fulfil the intended performance criteria desired in the dynamic environment.

And the games we have displayed have easier movements to grasp onto and learn. It would take less time to get habituated to them and play efficiently. Palm open and close are the easiest movements one can grasp whereas other methods have very difficult gesture options. We have divided the screen into parts for the user to interact in an efficient way.

- *Algorithm Enhancement-*

1. Blue Colour Detection

- Camera Settings: In order to perform runtime operations, the device's webcamera is used. To capture a video, we need to create a Video Capture object. Its argument can be either the device index or the name of a video file. Device index is just the number to specify which camera. Normally one camera will be connected, so we simply pass 0. You can select the second camera by passing 1 and so on. After that, you can capture frame-by-frame. But at the end, don't forget to release the capture. Moreover if anyone wants to apply this colour detection technique on any image it can be done with little modifications in the code.

- Capturing frames: The infinite loop is used so that the web camera captures the frames in every instance and is open during the entire course of the program.
  After capturing the live stream frame by frame we are converting each frame in BGR colour space (the default one) to HSV colour space. There are more than 150 colour-space conversion methods available in OpenCV. But we will look into only two which are most widely used ones, BGR to Gray and BGR to HSV. For colour conversion, we use the function cv2.cvtColor (input_image, flag) where flag determines the type of conversion. For BGR to HSV, we use the flag cv2.COLOR_BGR2HSV. Now we know how to convert BGR image to HSV, we can use this to extract a coloured object. In HSV, it is easier to represent a colour than RGB colour-space.

In specifying the range, we have specified the range of blue colour. Whereas you can enter the range of any colour you wish.

- Masking technique: The mask is basically creating some specific region of the image following certain rules. Here we are creating a mask that comprises of an object in blue colour. After that bitwise and used on the input image and the threshold image so that only the blue coloured objects are highlighted and stored in res. We then display the frame, res and mask on 3 separate windows using imshow function.
- Display the frame: As imshow () is a function of HighGui it is required to call waitKey regularly, in order to process its event loop. The function waitKey () waits for key event for a “delay” (here, 5 milliseconds). If you don’t call waitKey, HighGui cannot process windows events like redraw, resizing, input event etc. So just call it, even with a 1ms delay.
- Key Press: As far as the colour is matched with our specified range of Blue colour, PressKey(0x11) is called and key is pressed, and thread is kept to sleep for 1 ms. And when, colour isn’t recognise then, ReleaseKey(0x11) is called, which releases the key pressed.
- Summarizing the process:
  1. Take each frame of the video.
  2. Convert each frame from BGR to HSV colour-space.
  3. Threshold the HSV image for a range of blue colour.
  4. Press corresponding Key (4 and 6), if colour is in Range [{110, 40, 40}, {130, 255, 255}]

2. Palm Recognition

- Skin Colour Recognition: In order to detect the skin from the video, we need to find out the character of the skin. Detecting skin-coloured pixels, although seems a straightforward easy task, has proven to be quite a challenging task in images that are captured under complex unconstrained imaging conditions. So we developed a method based on the colour feature for most human. The formula is shown as below.

$$R>85$$
$$R-B>10$$
$$R-G>10$$

Based on such criterion, we can efficiently segment the skin from the background, which can be considered to be human part. Then we can convert the image into binary image. The results are shown as figure.

Fig. Converted Image

- Preprocessing: Due to the nature of image, the hand region may have holes and cracks, which will definitely affect the accuracy of hand gesture. Usually the binary image will be noisy, so image preprocessing is necessary, which fills the holes. There are two main methods for digital image restoration, texture based method and non-textured-based method. In our algorithm, the diffusion coefficients are defined according to the distance and direction between the damaged pixel and its neighbourhood pixel.

- Contour Extraction and Hand Region Segmentation: After we remove the noise in the image, we need to extract contours. We consider each point cluster as a contour. Among these contours, there is only one contour which represents the hand region. Furthermore, hand region and face region are the largest two contours. Based on such fact, the problem of finding the hand from the contours becomes the problem of separating hands from faces. So, we made a specific Box inside which hand is placed by user and is detected.
- Gesture Recognition: the four different scale spatial pyramids are pooled to get the size of 1/4, 1/8, 1/16, 1/32 feature map respectively. So the different scale features can be captured. Then, global average pooling is used to obtain the weights of global abstract features as channel dimensions at lower levels. Finally, the final probability score of each class is obtained by using fully connected layer and softmax.

$$a = \sqrt{(end[0] - start[0])^2 + (end[1] - start[1])^2}$$
$$b = \sqrt{(far[0] - start[0])^2 + (far[1] - start[1])^2}$$
$$c = \sqrt{(end[0] - far[0])^2 + (end[1] - far[1])^2}$$

$$angle = \cos^{-1}\left[\frac{b^2 + c^2 - a^2}{2bc}\right]$$

If $angle \leq 90^\circ$, and $count\_defects \geq 4$, then the Recognized image is Open Hand. And this condition is not satisfied then the Recognized Image is Closed Hand.

- <u>Summarizing the process</u>:
  1. Skin Colour Detection
  2. Contour Extraction and Hand Region Segmentation
  3. Image Preprocessing
  4. Gesture Recognition and comparison
  5. Recognized Gesture (Open or Close Hand)

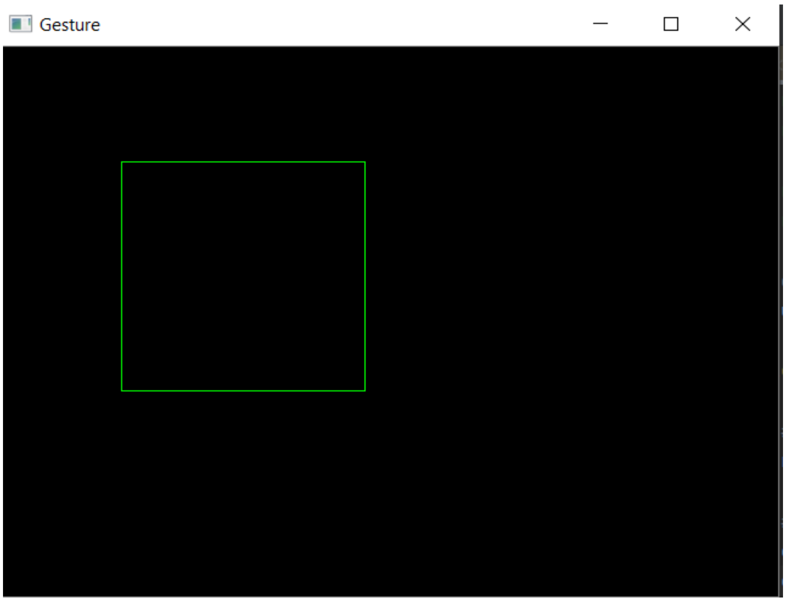
-><u>Overview organization of the proposed work (Context Diagram):</u>

The developed game will create randomised obstacles that the player needs to dodge in order to score points and beat his last highest score.

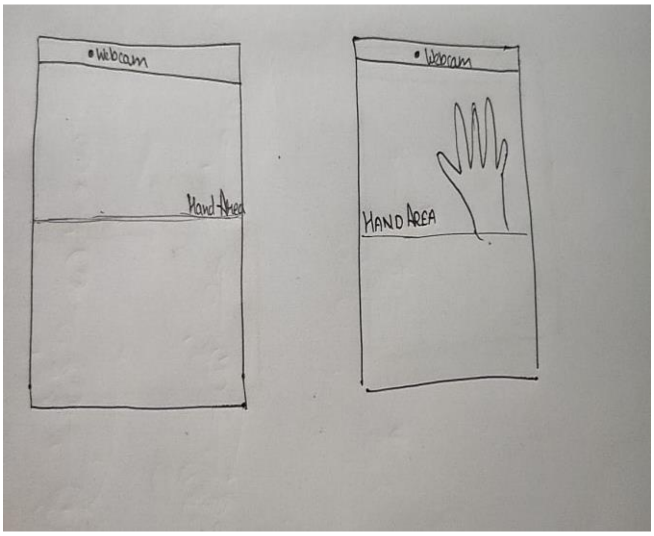
The game will also have a timer indicating the time which starts from the moment the player starts the game and would stop if the he fails to dodge any obstacle.

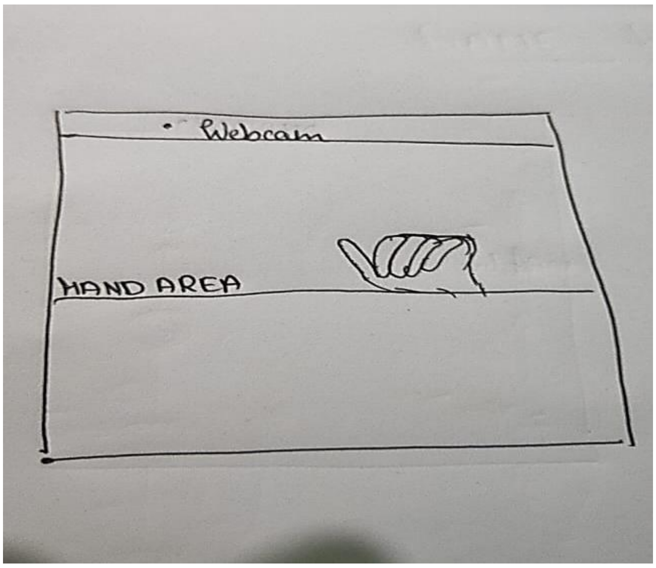
<u>Context Diagram:</u>

Hand Gesture Recognition

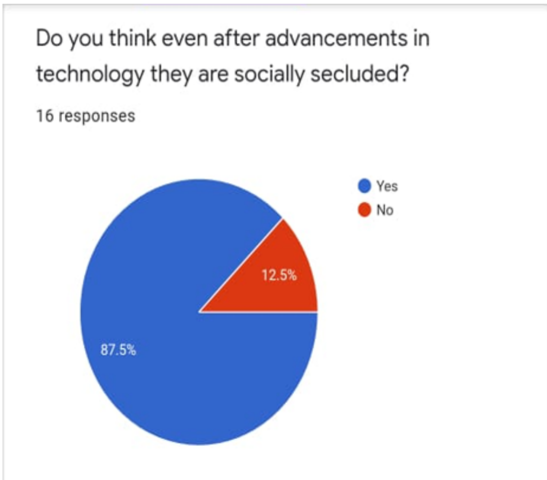


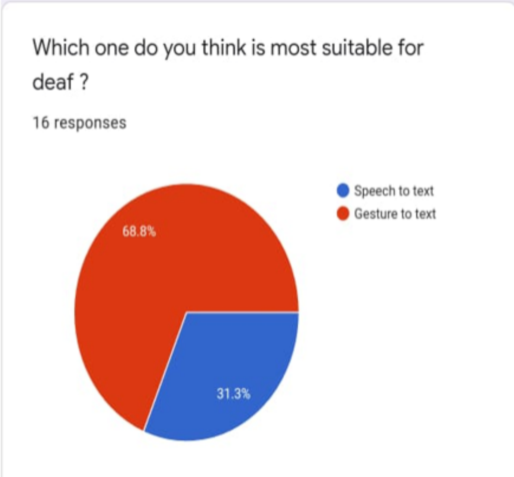
Story Boarding

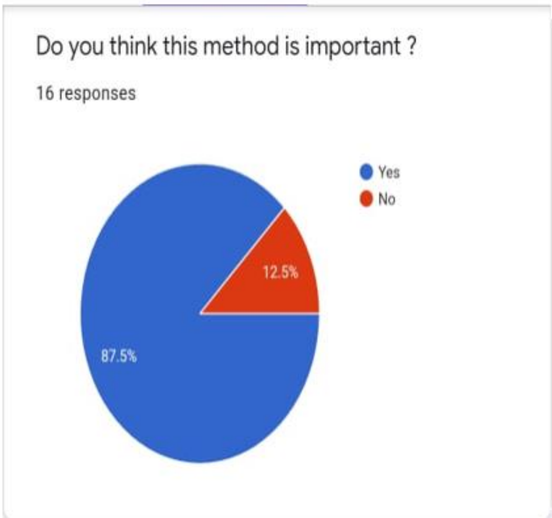


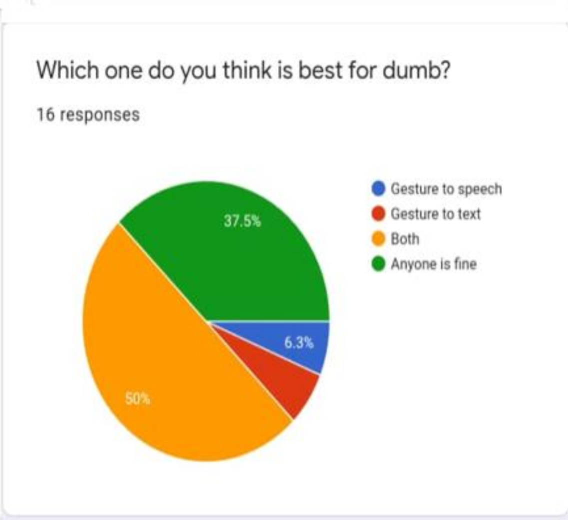


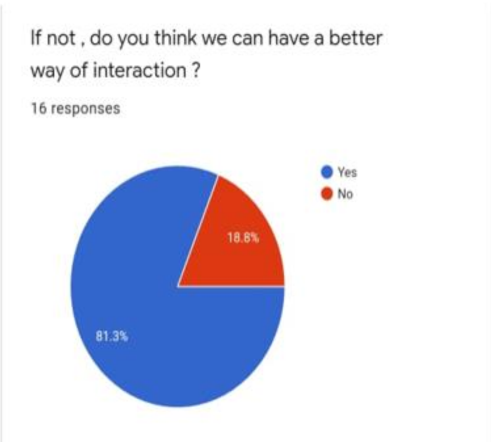
-Data Analysis & Feedback:

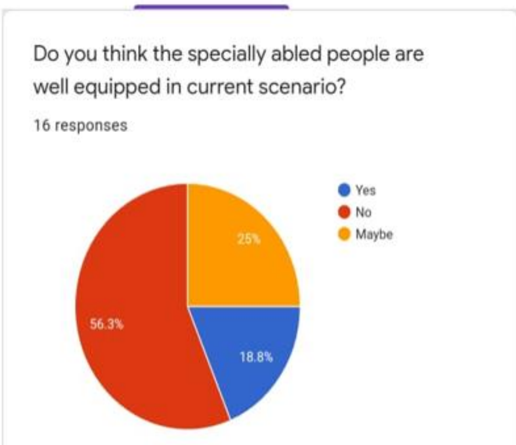


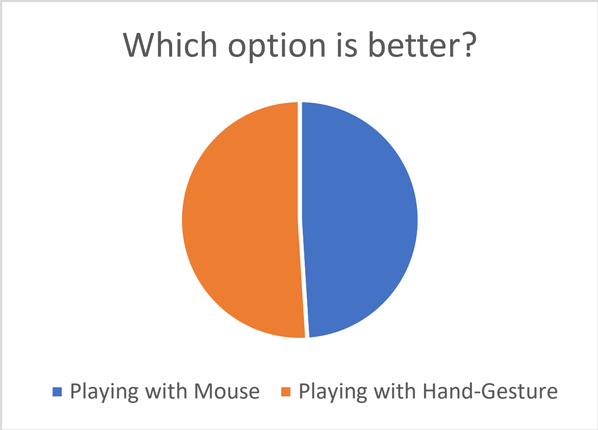




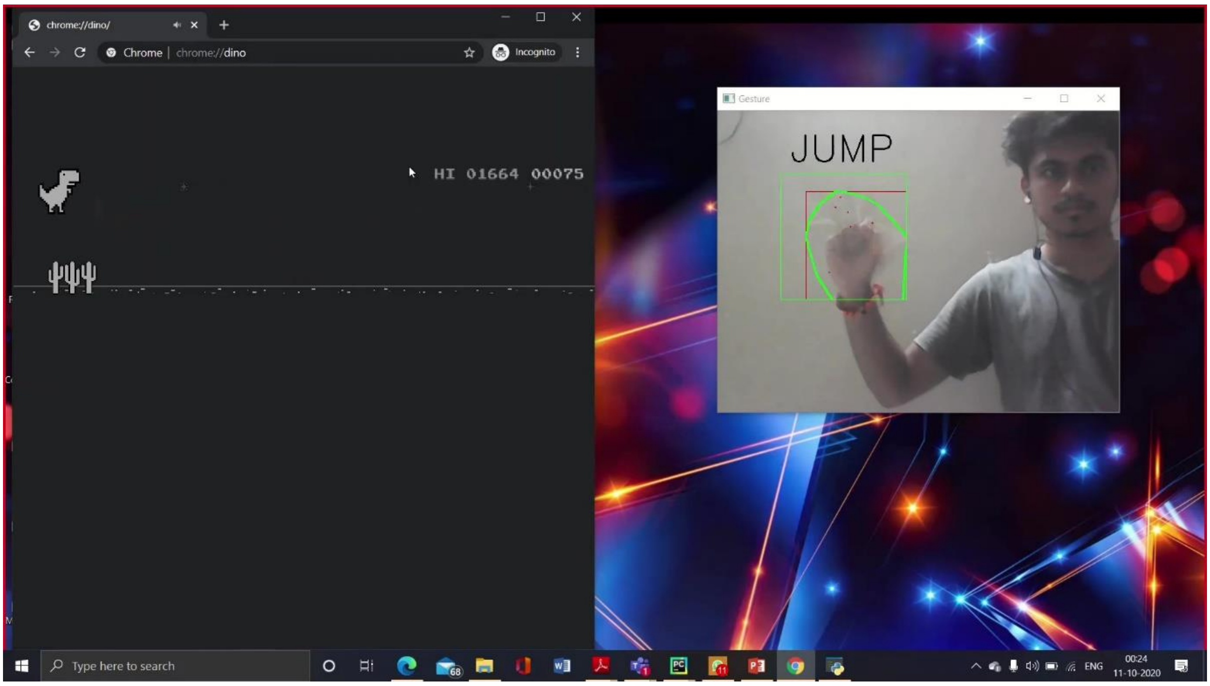








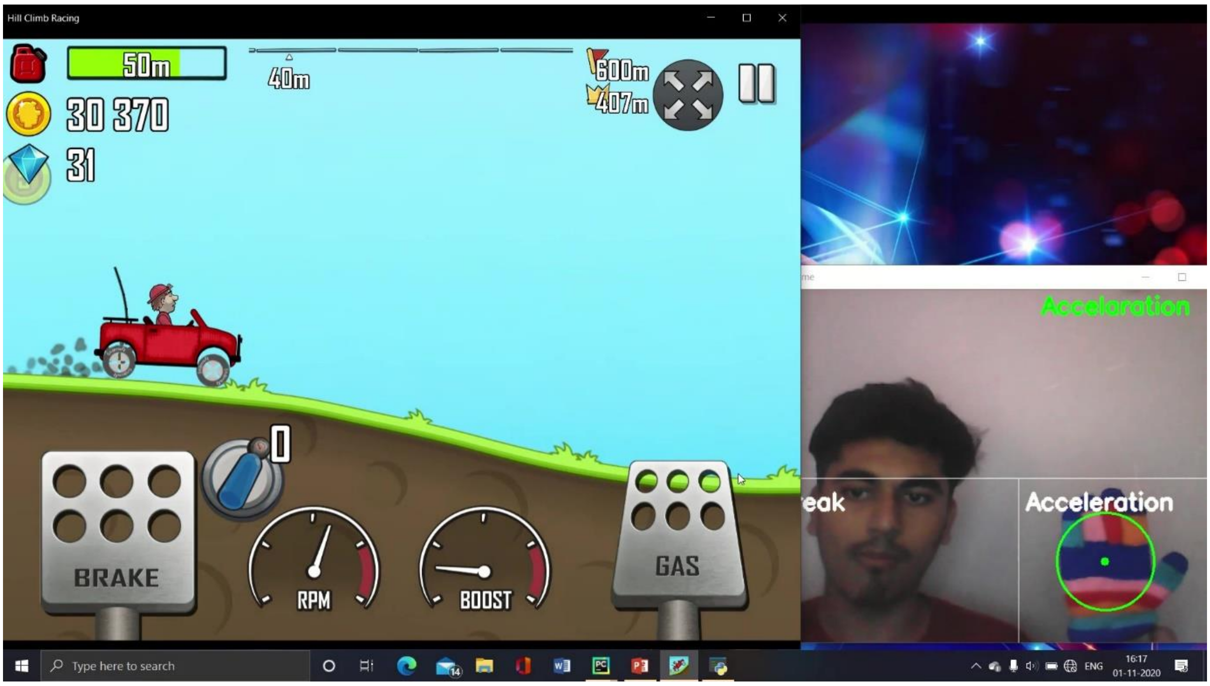
Sample code & output screenshots:

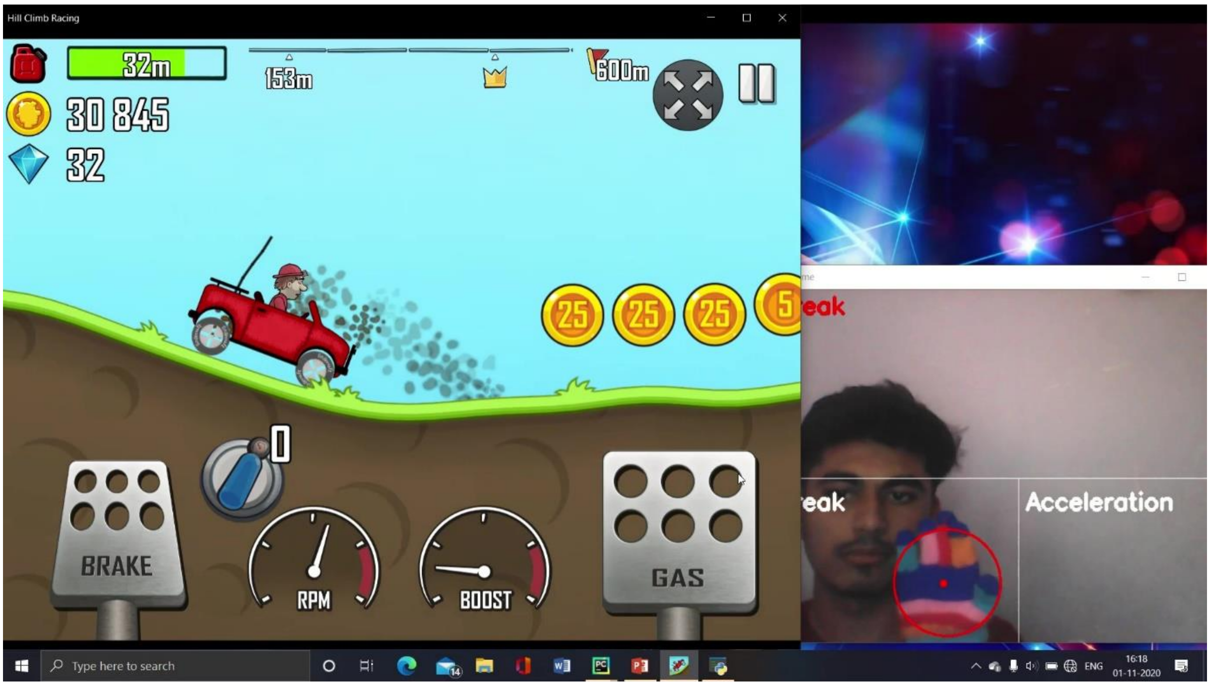



```
import numpy as np
import cv2
import math
import pyautogui

# Open Camera
capture = cv2.VideoCapture(0)

while capture.isOpened():

    # Capture frames from the camera
    ret, frame = capture.read()
```

```
    # Get hand data from the rectangle sub window
    cv2.rectangle(frame, (100, 100), (300, 300), (0, 255, 0), 0)
    crop_image = frame[100:300, 100:300]

    # Apply Gaussian blur
    blur = cv2.GaussianBlur(crop_image, (3, 3), 0)

    # Change colour-space from BGR -> HSV
    hsv = cv2.cvtColor(blur, cv2.COLOR_BGR2HSV)

    # Create a binary image with where white will be skin colors and rest
is black
    mask2 = cv2.inRange(hsv, np.array([2, 0, 0]), np.array([20, 255, 255]))

    # Kernel for morphological transformation
    kernel = np.ones((5, 5))

    # Apply morphological transformations to filter out the background
noise
    dilation = cv2.dilate(mask2, kernel, iterations=1)
    erosion = cv2.erode(dilation, kernel, iterations=1)

    # Apply Gaussian Blur and Threshold
    filtered = cv2.GaussianBlur(erosion, (3, 3), 0)
    ret, thresh = cv2.threshold(filtered, 127, 255, 0)

    # Find contours
    contours = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    hierarchy = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    image = cv2.findContours(thresh, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
    try:
        # Find contour with maximum area
        contour = max(contours, key=lambda x: cv2.contourArea(x))

        # Create bounding rectangle around the contour
        x, y, w, h = cv2.boundingRect(contour)
        cv2.rectangle(crop_image, (x, y), (x + w, y + h), (0, 0, 255), 0)

        # Find convex hull
        hull = cv2.convexHull(contour)

        # Draw contour
        drawing = np.zeros(crop_image.shape, np.uint8)
        cv2.drawContours(drawing, [contour], -1, (0, 255, 0), 0)
        cv2.drawContours(drawing, [hull], -1, (0, 0, 255), 0)

        # Fi convexity defects
        hull = cv2.convexHull(contour, returnPoints=False)
        defects = cv2.convexityDefects(contour, hull)

        # Use cosine rule to find angle of the far point from the start and
end point i.e. the convex points (the finger
        # tips) for all defects
        count_defects = 0

        for i in range(defects.shape[0]):
            s, e, f, d = defects[i, 0]
```

```
            start = tuple(contour[s][0])
            end = tuple(contour[e][0])
            far = tuple(contour[f][0])

            a = math.sqrt((end[0] - start[0]) ** 2 + (end[1] - start[1]) **
2)
            b = math.sqrt((far[0] - start[0]) ** 2 + (far[1] - start[1]) **
2)
            c = math.sqrt((end[0] - far[0]) ** 2 + (end[1] - far[1]) ** 2)
            angle = (math.acos((b ** 2 + c ** 2 - a ** 2) / (2 * b * c)) *
180) / math.pi

            # if angle >= 90 draw a circle at the far point
            if angle <= 90:
                count_defects += 1
                cv2.circle(crop_image, far, 1, [0, 0, 255], -1)

            cv2.line(crop_image, start, end, [0, 255, 0], 2)

        # Press SPACE if condition is match
        if count_defects >= 4:
            pyautogui.press('space')
            cv2.putText(frame, "JUMP", (115, 80), cv2.FONT_HERSHEY_SIMPLEX,
2, 2, 2)

    except:
        pass

    # Show required images
    cv2.imshow("Gesture", frame)

    # Close the camera if 'q' is pressed
    if cv2.waitKey(1) == ord('q'):
        break

capture.release()
cv2.destroyAllWindows()
```





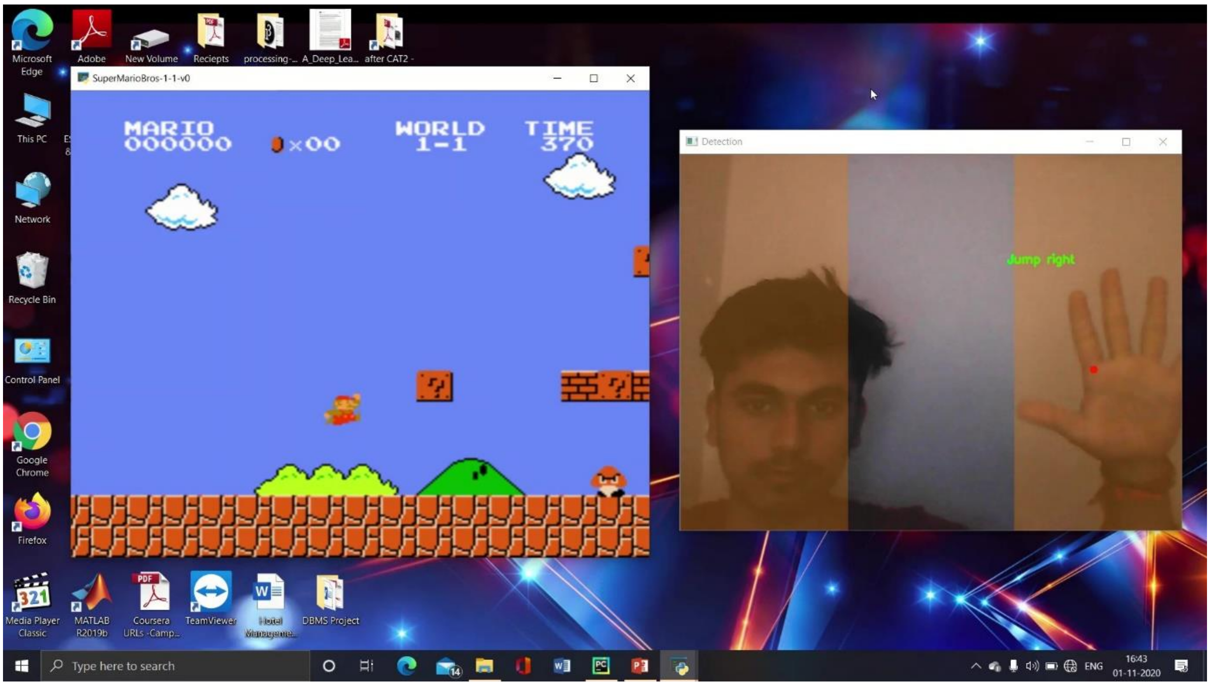

```
"""
@author: Heet Thakrar <heetthakrar@gmail.com>
@description: Passing input to the Keyboard(from camera).
"""
import ctypes
import time

SendInput = ctypes.windll.user32.SendInput

#List of Scan codes: https://wiki.osdev.org/PS/2_Keyboard
#cursor right pressed
right_pressed=0x4D
```

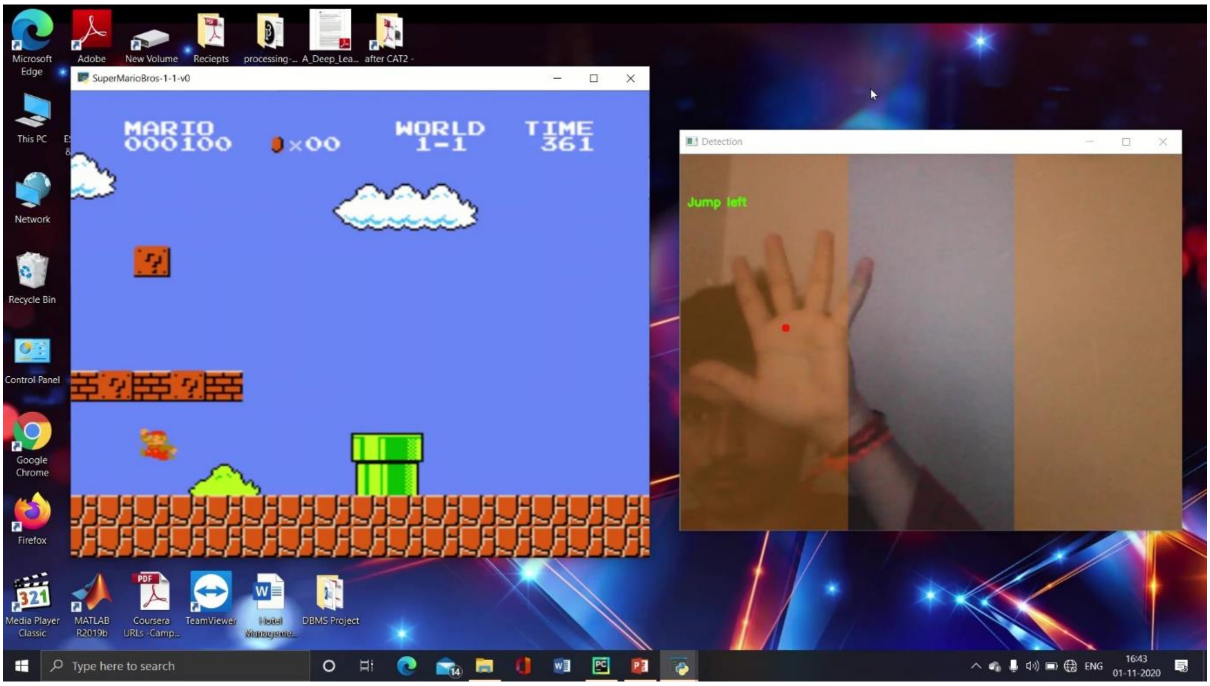
```
#cursor leftt pressed
left_pressed=0x4B

# C struct redefinitions
PUL = ctypes.POINTER(ctypes.c_ulong)
class KeyBdInput(ctypes.Structure):
    _fields_ = [("wVk", ctypes.c_ushort),
                ("wScan", ctypes.c_ushort),
                ("dwFlags", ctypes.c_ulong),
                ("time", ctypes.c_ulong),
                ("dwExtraInfo", PUL)]

class HardwareInput(ctypes.Structure):
    _fields_ = [("uMsg", ctypes.c_ulong),
                ("wParamL", ctypes.c_short),
                ("wParamH", ctypes.c_ushort)]

class MouseInput(ctypes.Structure):
    _fields_ = [("dx", ctypes.c_long),
                ("dy", ctypes.c_long),
                ("mouseData", ctypes.c_ulong),
                ("dwFlags", ctypes.c_ulong),
                ("time",ctypes.c_ulong),
                ("dwExtraInfo", PUL)]

class Input_I(ctypes.Union):
    _fields_ = [("ki", KeyBdInput),
                 ("mi", MouseInput),
                 ("hi", HardwareInput)]

class Input(ctypes.Structure):
    _fields_ = [("type", ctypes.c_ulong),
                ("ii", Input_I)]

def PressKey(hexKeyCode):
   extra = ctypes.c_ulong(0)
   ii_ = Input_I()
   ii_.ki = KeyBdInput( 0, hexKeyCode, 0x0008, 0, ctypes.pointer(extra) )
   x = Input( ctypes.c_ulong(1), ii_ )
   ctypes.windll.user32.SendInput(1, ctypes.pointer(x), ctypes.sizeof(x))

def ReleaseKey(hexKeyCode):
    extra = ctypes.c_ulong(0)
    ii_ = Input_I()
    ii_.ki = KeyBdInput( 0, hexKeyCode, 0x0008 | 0x0002, 0,
ctypes.pointer(extra) )
    x = Input( ctypes.c_ulong(1), ii_ )
    ctypes.windll.user32.SendInput(1, ctypes.pointer(x), ctypes.sizeof(x))

if __name__=='__main__':
   while (True):
      PressKey(0x11)
      time.sleep(1)
      ReleaseKey(0x11)
      time.sleep(1)

"""
@author: Heet Thakrar <heetthakrar@gmail.com>
@description: Game controlling with Fists in Navy blue gloves using openCV.
Left Fist- Break Right Fist- Acceleration
```

```
"""

from imutils.video import VideoStream
import numpy as np
import cv2
import imutils
import time
from directkeys import right_pressed,left_pressed
from directkeys import PressKey, ReleaseKey


break_key_pressed=left_pressed
accelerato_key_pressed=right_pressed

# define the lower and upper boundaries of the "navy blue" object in the
HSV colour space
blueLower = np.array([110, 40, 40])
blueUpper = np.array([130,255,255])

vs = VideoStream(src=0).start()

# allow the camera or video file to warm up
time.sleep(2.0)
initial = True
flag = False
current_key_pressed = set()
circle_radius = 30
windowSize = 160
lr_counter = 0

# keep looping
break_pressed=False
accelerator_pressed=False
while True:
    keyPressed = False
    break_pressed=False
    accelerator_pressed=False
    # grab the current frame
    frame = vs.read()
    height,width = frame.shape[:2]

    #Flipped the frame so that left hand appears on the left side and right
hand appears on the right side
    frame = cv2.flip(frame,1);

    # resize the frame, blur it, and convert it to the HSV colour space
    frame = imutils.resize(frame, height=300)
    frame = imutils.resize(frame, width=600)
    blurred = cv2.GaussianBlur(frame, (11, 11), 0)
    hsv = cv2.cvtColor(blurred, cv2.COLOR_BGR2HSV)

    # crteate a mask for the orange colour and perform dilation and erosion
to remove any small
    # blobs left in the mask
    mask = cv2.inRange(hsv, blueLower, blueUpper)
    mask = cv2.erode(mask, None, iterations=2)
    mask = cv2.dilate(mask, None, iterations=2)

    # find contours in the mask and initialize the current
    # (x, y) center of the orange object
```

```
    # divide the frame into two halves so that we can have one half control
the acceleration/brake
    # and other half control the left/right steering.
    left_mask = mask[:,0:width//2,]
    right_mask = mask[:,width//2:,]

    #find the contours in the left and right frame to find the center of
the object
    cnts_left = cv2.findContours(left_mask.copy(), cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)
    cnts_left = imutils.grab_contours(cnts_left)
    center_left = None

    cnts_right = cv2.findContours(right_mask.copy(), cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)
    cnts_right = imutils.grab_contours(cnts_right)
    center_right = None
    # only proceed if at least one contour was found
    key_count=0
    key_pressed=0
    if len(cnts_left) > 0:
        # find the largest contour in the mask, then use
        # it to compute the minimum enclosing circle and centroid
        c = max(cnts_left, key=cv2.contourArea)
        ((x, y), radius) = cv2.minEnclosingCircle(c)
        M = cv2.moments(c)
        # find the center from the moments 0.000001 is added to the
denominator so that divide by
        # zero exception doesn't occur
        center_left = (int(M["m10"] / (M["m00"]+0.000001)), int(M["m01"] /
(M["m00"]+0.000001)))
        #print("center_left",center_left)
        # only proceed if the radius meets a minimum size
        if radius > circle_radius:
            # draw the circle and centroid on the frame,
            cv2.circle(frame, (int(x), int(y)), int(radius),
                (0, 0, 255), 2)
            cv2.circle(frame, center_left, 5, (0, 0, 255), -1)
            #Bottom Left region
            if center_left[1] > 250:

cv2.putText(frame,'Break',(10,30),cv2.FONT_HERSHEY_SIMPLEX,1,(0,0,255),3)
                PressKey(break_key_pressed)
                break_pressed=True
                current_key_pressed.add(break_key_pressed)
                #Break key- 75 #Acc key-77
                key_pressed=break_key_pressed
                keyPressed = True
                key_count=key_count+1
    # only proceed if at least one contour was found
    if len(cnts_right) > 0:
        c2 = max(cnts_right, key=cv2.contourArea)
        ((x2, y2), radius2) = cv2.minEnclosingCircle(c2)
        M2 = cv2.moments(c2)
        center_right = (int(M2["m10"] / (M2["m00"]+0.000001)),
int(M2["m01"] / (M2["m00"]+0.000001)))
        center_right = (center_right[0]+width//2,center_right[1])
        # only proceed if the radius meets a minimum size
        if radius2 > circle_radius:
            cv2.circle(frame, (int(x2)+width//2, int(y2)), int(radius2),
                (0, 255, 0), 2)
```

```
            cv2.circle(frame, center_right, 5, (0, 255, 0), -1)
            #Bottom Right region
            if center_right[1] >250 :

cv2.putText(frame,'Accelaration',(350,30),cv2.FONT_HERSHEY_SIMPLEX,1,(0,255
,0),3)
                PressKey(accelerato_key_pressed)
                key_pressed=accelerato_key_pressed
                accelerator_pressed=True
                keyPressed = True
                current_key_pressed.add(accelerato_key_pressed)
                key_count=key_count+1

    frame_copy=frame.copy()
    #Bottom left region rectangle
    frame_copy = cv2.rectangle(frame_copy,(0,height//2
),(width//2,width),(255,255,255),1)

cv2.putText(frame_copy,'Break',(10,280),cv2.FONT_HERSHEY_SIMPLEX,1,(255,255
,255),3)
    #Bottom right region rectangle
    frame_copy =
cv2.rectangle(frame_copy,(width//2,height//2),(width,height),(255,255,255),
1)

cv2.putText(frame_copy,'Acceleration',(330,280),cv2.FONT_HERSHEY_SIMPLEX,1,
(255,255,255),3)

    # show the frame to our screen
    cv2.imshow("Frame", frame_copy)

    #If part: We need to release the pressed key if none of the key is
pressed else the program will keep on sending
    #Else part:If different keys(Only one key in each frame) are pressed in
previous and current frames, then we must
    #release previous frame key, Also release the key in current frame key
for smoother control
    if not keyPressed and len(current_key_pressed) != 0:
        for key in current_key_pressed:
            ReleaseKey(key)
        current_key_pressed = set()
    elif key_count==1 and len(current_key_pressed)==2:
            for key in current_key_pressed:
                if key_pressed!=key:
                    ReleaseKey(key)
            current_key_pressed = set()
            for key in current_key_pressed:
                ReleaseKey(key)
            current_key_pressed = set()

    key = cv2.waitKey(1) & 0xFF
    # if the 'q' key is pressed, stop the loop
    if key == ord("q"):
        break


vs.stop()
# close all windows
cv2.destroyAllWindows()
```






```
"""
@author: Heet Thakrar <heetthakrar@gmail.com>
"""
import tensorflow as tf
import cv2
import multiprocessing as _mp
from src.utils import load_graph, mario, detect_hands, predict
from src.config import ORANGE, RED, GREEN

tf.compat.v1.flags.DEFINE_integer("width", 640, "Screen width")
tf.compat.v1.flags.DEFINE_integer("height", 480, "Screen height")
tf.compat.v1.flags.DEFINE_float("threshold", 0.6, "Threshold for score")
```

```
tf.compat.v1.flags.DEFINE_float("alpha", 0.3, "Transparent level")
tf.compat.v1.flags.DEFINE_string("pre_trained_model_path",
"src/pretrained_model.pb", "Path to pre-trained model")

FLAGS = tf.compat.v1.flags.FLAGS


def main():
    graph, sess = load_graph(FLAGS.pre_trained_model_path)
    cap = cv2.VideoCapture(0)
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, FLAGS.width)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, FLAGS.height)
    mp = _mp.get_context("spawn")
    v = mp.Value('i', 0)
    lock = mp.Lock()
    process = mp.Process(target=mario, args=(v, lock))
    process.start()
    while True:
        key = cv2.waitKey(10)
        if key == ord("q"):
            break
        _, frame = cap.read()
        frame = cv2.flip(frame, 1)
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        boxes, scores, classes = detect_hands(frame, graph, sess)
        frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
        results = predict(boxes, scores, classes, FLAGS.threshold,
FLAGS.width, FLAGS.height)

        if len(results) == 1:
            x_min, x_max, y_min, y_max, category = results[0]
            x = int((x_min + x_max) / 2)
            y = int((y_min + y_max) / 2)
            cv2.circle(frame, (x, y), 5, RED, -1)

            if category == "Open" and x <= FLAGS.width / 3:
                action = 7  # Left jump
                text = "Jump left"
            elif category == "Closed" and x <= FLAGS.width / 3:
                action = 6  # Left
                text = "Run left"
            elif category == "Open" and FLAGS.width / 3 < x <= 2 *
FLAGS.width / 3:
                action = 5  # Jump
                text = "Jump"
            elif category == "Closed" and FLAGS.width / 3 < x <= 2 *
FLAGS.width / 3:
                action = 0  # Do nothing
                text = "Stay"
            elif category == "Open" and x > 2 * FLAGS.width / 3:
                action = 2  # Right jump
                text = "Jump right"
            elif category == "Closed" and x > 2 * FLAGS.width / 3:
                action = 1  # Right
                text = "Run right"
            else:
                action = 0
                text = "Stay"
            with lock:
                v.value = action
            cv2.putText(frame, "{}".format(text), (x_min, y_min - 5),
```

```
                        cv2.FONT_HERSHEY_SIMPLEX, 0.5, GREEN, 2)
        overlay = frame.copy()
        cv2.rectangle(overlay, (0, 0), (int(FLAGS.width / 3),
FLAGS.height), ORANGE, -1)
        cv2.rectangle(overlay, (int(2 * FLAGS.width / 3), 0), (FLAGS.width,
FLAGS.height), ORANGE, -1)
        cv2.addWeighted(overlay, FLAGS.alpha, frame, 1 - FLAGS.alpha, 0,
frame)
        cv2.imshow('Detection', frame)

    cap.release()
    cv2.destroyAllWindows()


if __name__ == '__main__':
    main()
```

Conclusion:

Gesture based interfaces allow human computer interaction to be in a natural and intuitive manner.

The most important advantage of the usage of hand gesture based input modes is that using this method the user can interact with the application from a distance without any physical interaction with the keyboard/mouse. This paper develops a hand gesture recognition system for interacting with different app like image browser; games etc. and provides a fruitful solution towards a user-friendly interface between human and computer. The gesture vocabulary designed can be further extended for controlling different applications like game control etc. We build 3 game systems- Dino, Mario and Hill Climb racing on top of this hand gesture recognition method to show its applicability to be a key enabler for many other hand gestures based HCI systems. The system provides the flexibility to the users and specifically physically challenged users to define the gesture according to their feasibility and ease of use.

Future Prospects-

Our games/gesture recognition system can even be developed to include virtual reality to change the character, can be improvised on a large server to incorporate multi player levels to have a better real life social experience with an allowance for voice or text based conversation. The graphics at present are proposed to be simple, however they can be highly influenced to indulge the user with frontal, parietal, occipital and temporal involvement of brain for an impactful experience.

References:

- S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. IEEE Trans. on PAMI, 2002.
- M. Bray, E. Koller-Meier, and L. V. Gool. Smart particle filtering for 3d hand tracking, Sixth IEEE International Conf. 2004
- Choi, Yoo-Joo., Lee, Je-Sung, Cho. We-Duke. A Robust Hand Recognition in Varying Illumination. Advances in HCI, Shane Pinder (Ed.), 2006.
- Le T N, Cong D T, Ba T N, et al. Contour Based Hand Gesture Recognition Using Depth Data. International Conference on Signal Processing, Image Processing and Pattern Recognition, 2013.
- Kang S K, Mi Y N, Rhee P K. Colour Based Hand and Finger Detection Technology for User Interaction. International

Conference on Convergence and Hybrid Information Technology. IEEE, 2008.

- Hill Climb Racing for Windows devices, Fingersoft, Oulu, Finland, 2012.
