Virtual Mouse using Machine Learning and GUI Automation

U Sairam, Assistant Professor
Department of Information Technology
Chaitanya Bharathi Institute of Technology(A)
Hyderabad, India
usairam_it@cbit.ac.in

Dharani Kumar Reddy Gowra
Department of Information Technology
Chaitanya Bharathi Institute of Technology(A)
Hyderabad, India
dharanikumarreddy23@gmail.com

Sai Charan Kopparapu
Department of Information Technology
Chaitanya Bharathi Institute of Technology(A)
Hyderabad, India
saicharan3199@gmail.com

2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS) | 978-1-6654-0816-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICACCS54159.2022.9784972

***Abstract*—With the rapid advancement of computer technology, it is becoming increasingly important for us to develop new ways of interacting with computers. From CLI (Command Line Interface) to GUI (Graphical User Interface) the growth has been astounding. The future is going to be NUI (Natural User Interface) where the user makes use of one's natural environment to provide input to machine. Imagine playing Video games likes Temple Run, Subway Surfers or even play a musical instrument on your computer without touching, this way also helps in improving the digital well-being of humans. NUI/NUX will also be a lot helpful during the time of pandemic where we can reduce touch in many public places.**

An input module, such as a virtual mouse that uses Object Detection, Object Tracking and Gestures to assist us in communicating, could be a viable replacement to the traditional touch screen and hardware mouse. The system design proposed is a TensorFlow-based mouse controlling system, which uses hand gestures that are captured through a webcam using an RGB color scheme. This solution permits users to control the system cursor using their hand, that the computer webcam tracks and perform mouse operations like Left click, Right click, Scroll, Drag, Move using different hand gestures. Python, TensorFlow and OpenCV libraries are used for real time computer vision to implement the system.

Keywords—Virtual Mouse, NUI (Natural User Interface), NUX (Natural User Experience), Object Detection, Object Tracking, TensorFlow, OpenCV, Gesture Detection, GUI Automation,

I. INTRODUCTION

A. Domain Overview

The communication between humans and machines is in continuous development, but the HCI (Human-Computer Interaction) is still limited by usage of certain physical equipment to give the command. Therefore, to fulfil the futuristic requirements in economic and user-friendly process, gesture recognition has become a trending research topic in the field of HCI applications. Human-computer interaction based on computer vision is still in the theoretical and experimental stages, the number of gestures can be identified is relatively small. Most of the gestures proposed are not very user friendly. The key problem in hand gesture interaction is how to make hand gestures understood by computers keeping them User Friendly.

The methods of implementation may be split into three categories: "Data-Glove" based, "Vision based", and "Machine Learning based". Vision based recognition is contactless and most natural. It's growing increasingly popular, and numerous works have been created utilizing it. Due to differences in hand sizes, hand position and orientation, lighting conditions, and other factors, this technique presents some challenges. In Glove-based colored stickers, pointers and gloves or other devices are being used which are not feasible for interacting in real-time environment. Furthermore, these techniques have additional costs, a lack of availability among the general public, and a higher level of upkeep, among other things. In Machine Learning Based approach though accuracy is high, Latency is also relatively low. We have implemented and checked Vision Based approach and found the accuracy is not that efficient and lot of factors have huge impact on accuracy. In Vision based gestures are supposed to be complex and they are not very User friendly. So, we changed our approach to Machine Learning based and implemented using SSD-Mobile Net and trained using transfer learning which has produced high accuracy and Latency is enough to not be noticed.

B. Motivation

Our main motivation came from a LinkedIn Post where the user plays the drums using pen. This was a really cool project then we realized how it was implemented using OpenCV. This kind of implementation can also be used to give instructions while playing games such as Subway Surfers, Temple Run where limited number of actions are required. We wanted to see what it would be like if we were able to control the mouse by just using hand gestures. This is the main motivation behind this project we want to make it as user friendly as possible, with this in mind all gestures are designed.

C. Problem Statement

Develop a solution to replace existing input mechanism (USB Mouse), which is efficient, reliable, easy to use with good accuracy, less latency and minimum hardware. The user should be able to control the mouse pointer without any external support like Gloves, Sensing Device and should be able to perform all mouse operation like moving, selecting, scrolling, left click, right click. With the proposed solution user will be able to provide mouse pointer input to the machine or computer with the bare hands without using any external objects like data gloves.

II. RELEVEANT WORK

A. Design of control system based on hand gesture recognition(Shining Song, Dongsong Yan, Yongjun Xie)

Advantages: YCbCr color schema helps in reducing the effect of light condition of room. Otsu's Algorithm is very efficient in processing the foreground from background.
Limitations: Interfacing with OS is not performed, only 4 gestures were implemented and detected.

B. Virtual Mouse Using Object Tracking(Prof. Monali Shetty, Christina A. Daniel, Manthan K.Bhatkar, Ofrin P. Lopes)

Advantages: High accuracy is produced as Glove based solution is implemented.
Limitations: Very low accuracy in Non – plain Background, Gestures used are complex. Glove based solutions are not very user friendly and they are not much different from using regular mouse effort is same.

C. Hand Gesture Real Time Paint Tool – Box: Machine Learning Approach(Vandit Gajjar, Viraj Mavani, Ayesha Gurnani)

Advantages: With the use of Machine Learning methods some amount of flexibility in choosing gestures is obtained.
Limitations: High Latency, Only Developed for Paint Application

D. A Novel Design of an Intangible Hand Gesture Controlled Computer Mouse using Vision Based Image Processing (Rokhsana Titlee, Ashfaq Ur Rahman, Hasan U. Zaman, and Hafiz Abdur Rahman)

Advantages: Easy to process implementation where three operations are implemented.
Limitations: Thresholding is not efficient, OpenCV is easier and easier to use compared to MATLAB.

III. IMPLEMENTATION

To perform all basic mouse operation like Move, Left Click, Right Click, Drag, scroll we need 5 gestures. We have proposed 5 gestures for 5 different actions our dataset should also contain about 20 images for each gesture. These images act as deciding factors on the accuracy. Among these 20 images 15 are used for training and 5 for testing purpose. The gesture detection algorithm should be able to extract and process the frames from live video and return the class, score, boxes of gestures detected. These boxes and their coordinates will be used to track the position. Each frame after processing will return the class and boxes. Given coordinates of each gesture, intersection of diagonals will be considered as reference point and this point will be passed to PyAutoGui part of code where all 5 gestures and their specified actions are to be performed efficiently.

After implementing the image-processing-based approach, we have realized that the accuracy was not as expected so we implemented the project using machine learning methods, on contrary to many other papers the latency is negligible from user's perspective and can be used efficiently as it provides good accuracy. So, after extracting frames from video, each frame is passed onto the trained machine learning model which will detect the position and class label of gesture in frame. Based on the gesture in frame, the action to be performed is decided.

These XML files will be used for training purpose. We will divide training and testing images manually as 15 for training and 5 for testing. We then update the Label Map which contains mapping of class labels and ids. To simplify the training process, we make use of pretrained model called "SSD Mobile Net V2 FPN Lite" and train it using transfer learning and TensorFlow. After completion of training, we connect the video feed from webcam to the custom trained model and try to get the coordinates of gesture in the frame with maximum probability it's score and bounding box coordinate.

After obtaining coordinates of reference point, which is intersection of diagonals of the bounding box. Based on the gesture detected consequent actions are performed by using PyAutoGUI to Automate GUI.

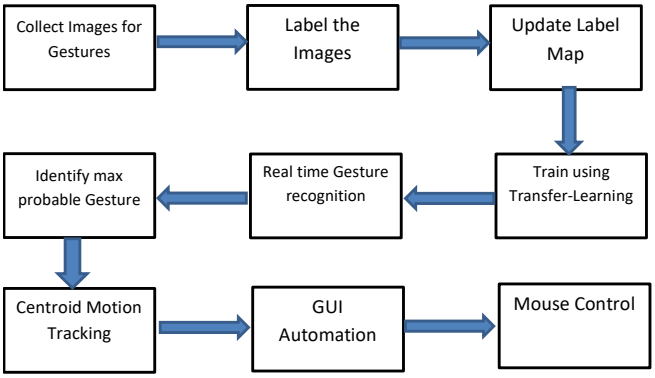

Fig 3.1 System Design

Collecting Images of Gestures**:** As we are doing the training using transfer learning, we require only a smaller number of images for training purpose. So, we collect only 20 images for each gesture where 15 images will be used for training and 5 images for testing. These images play a key role in loss metric of trained model. The five gestures collected are Index, Left Thumb, Right Thumb, Fist, Palm.

These images are stored and renamed accordingly. More diverse Images in dataset provide more accuracy in different and complex backgrounds. These images collected with use of time package.

Label the Images: With the help of "Labelimg" package we can label each image with class name and draw a bounding box around the area of interest. After the image is labelled a XML file is generated which contains the data about image name, class name, bounding box coordinates. These XML files would be used later on to generate the TF Records.

Update the Label Map**:** Label Map contains the mapping of class name and their ID's which are integer values. This label map has file extension of ".pbtxt".This file will be used later on to generate the tf record files for training and testing data.TF Record is a simple format for storing the binary records which will be used for custom training the model.

Training Using Transfer Learning**:** Transfer learning is a technique used in machine learning to reuse a pre- trained model on a new problem. In transfer learning, a machine

exploits the knowledge gained from a previous task to improve generalization about another.

Fig 3.2 Index Finger used to move Mouse

Fig 3.3 Fist used to Drag and Select

Fig 3.4 Palm used to perform scroll

Real- Time Gesture Detection: Our custom trained model using the pre-trained model SSD Mobile Net will process video as frames and return the classes, boxes and probability scores in each frame.

Identifying Max Probable: After obtaining results of each frame, we need to find the index position of gesture which has maximum score and is above our threshold limit of 50%. Only this data would be represented in the final data to user .

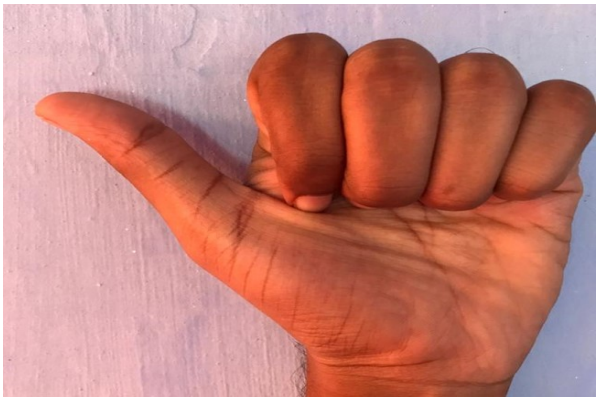

Fig 3.5 Left Thumb used to perform Left Click

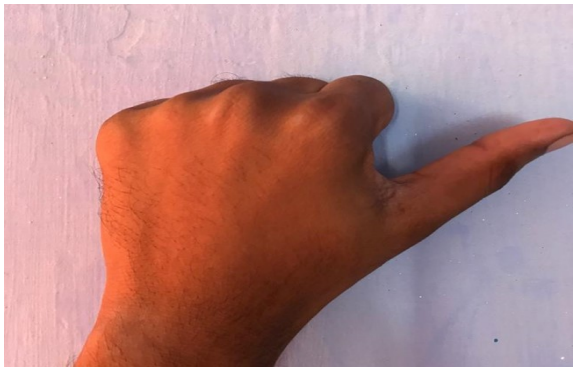

Fig 3.6 Right Thumb Used to perform Right Click

Centroid Motion Tracking**:** The intersection point of diagonals in the bounding box of maximum probable gesture will be considered as reference point. In every frame this point would be used to perform actions.

GUI Automation & Control**:** With the help of PyAutoGUI Package we implement the logic for all 5 gestures perform different mouse actions in such a way that all of them are User friendly.

We make use of different packages such as cv2(OpenCV), OS, time, uuid (Universal Unique Identifier), OpenCV is useful for capturing images through webcam, OS package is used to process the file paths, IMAGES_PATH will represent the storage location of images which is collected images directory. Labels and no of images variables are used to store data and reuse when required. Uuid will be used to generate unique names to each and every image. For each label a sub directory inside collected images is created and all images of that label are stored in it.

Fig 3.7 Created folders for each Label

Once the images are collected we need to place all of them in same folder and pass it to the Labeling package where a User interface is displayed to the user to draw the bounding box and give a class name to the area of interest. This will generate a XML file containing the class name and bounding box data.

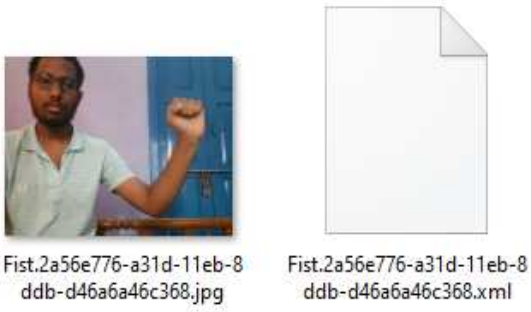

Fig 3.8 Generated XML file using Labeling

This Label Map contains the mapping of class names and their ID's which will be used by computer to process the class data. We make the changes and then create a file with ". pbtxt" extension which will be used later on. TFrecords are used for storing the binary data in a format which is understood by TensorFlow. The commands are run in console and they generate two files names "train. Record" and "test.record". We extract the model's repo from TensorFlow and setup the custom trained model by moving the config file pipeline to training folder where we need to make changes to the custom model and train it.

We need to setup the config path and import all the required packages the "pipeline.config" file consists of data regarding the features of custom trained model like no of classes how are the training images to be resized and how they are to be processed.

This config file takes all the setup from our Label Map to generated TFRecord files which will be used to start our training of pre trained model using transfer learning and tensor flow. After completion of training checkpoints will be generated these check points act as a restore options which can be used when required now, we have to load these check points to complete the training process.

Detect is function which takes frame and return the dictionary containing classes, scores, boxes of all the gestures detected.

We import all the required packages for detection and automation such as "pyautogui" which is used for GUI Automation and to perform all mouse operations, "time" package is used to create delays and perform all time related operations, "cv2" packages is nothing but OpenCV package used to capture video and perform all Image processing operations, "datetime" package is used to get the current time and perform arithmetic operation on time object, "sys" package for all path related stuff. Along with these packages we also require "NumPy" to perform large matrix operations and "TensorFlow" to perform classification.

```
import os
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
```

```
# Load pipeline config and build a detection model
configs = config_util.get_configs_from_pipeline_file(CONFIG_PATH)
detection_model = model_builder.build(model_config=configs['model'], is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(CHECKPOINT_PATH, 'ckpt-6')).expect_partial()

@tf.function
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections
```

Fig 3.9 Loading the checkpoint

The coordinates used by TensorFlow are different from the ones used in OpenCV so they are converted if and only if the maximum score is greater than 50%. Start is tuple which indicates the left top point and end indicates the right bottom point. To track the gesture we need a reference point. In this implementation we consider intersection of diagonals as reference point. To get the relative position we get a difference tuple between previous and current location.

Drag gesture is a bit complex and before performing any operation we need to check if is drag variable is false. If it is false we press down the mouse left button and set is drag as True. If is drag is True we simply move the mouse to reference point as already the mouse left click is pressed down.

```
if classes[max_score_index]==0:
    #Fist Drag
    if not is_Drag:
        pyautogui.mouseDown()
        is_Drag=True
    else:
        pyautogui.moveTo(ref_pt)
    #pyautogui.dragTo(ref_pt)
```

Fig 3.10 Implementation of Drag

Even in the implementation of Index we need to handle the drag closing part which is releasing the mouse left click and resetting is drag to false. To perform move we simply use moveTo() function and pass reference point as argument.

To perform left click we create a delay of 1.6 seconds i.e. whenever a left click is pressed only after 1.6 seconds left click will be pressed again .This delay helps in processing frames with continuous sequence of left clicks and making it user friendly. Similar code will be used for right click also.

```
elif classes[max_score_index]==1:
    #Index
    #pyautogui.move(rel_loc)
    if is_Drag:
        pyautogui.mouseUp()
        is_Drag=False
    pyautogui.moveTo(ref_pt)
```

Fig 3.11 Implementation of Move

```
elif classes[max_score_index]==2:
    #Left_thumb
    if is_Drag:
        pyautogui.mouseUp()
        is_Drag=False
    curr_time=datetime.datetime.now()
    if (curr_time-prev_action).total_seconds()>1.6:
        pyautogui.doubleClick()
        prev_action=curr_time
    else:
        pass
```

Fig 3.12 Implementation of Left Click

If the reference point of scroll gesture is in left half of screen Page up operation will be performed else Page Down operation will be performed.

```
elif classes[max_score_index]==3:
    #Palm
    if is_Drag:
        pyautogui.mouseUp()
        is_Drag=False
    if ref_pt[0]<=320:
        pyautogui.press('pgup')
    else:
        pyautogui.press('pgdn')
```

Fig 3.13 Implementation of Scroll

IV. Results

After using Labeling package and labelling the image an XML file is generated which will contains the class name and coordinates of bounding box containing the area of interest in which the gesture is present. These generated XML files act as manifest to the image for which it is generated. The gestures used are Index is used to move the mouse pointer, Fist is used to perform select or Drag operation, Thumb pointed left side is used for left click and thumb pointed right side is used for right click. Palm if present on left half of screen perform Page Up and on the right half of screen perform Page Down.

The user interface provided and scores of the detected class will be displayed as shown in the below results.

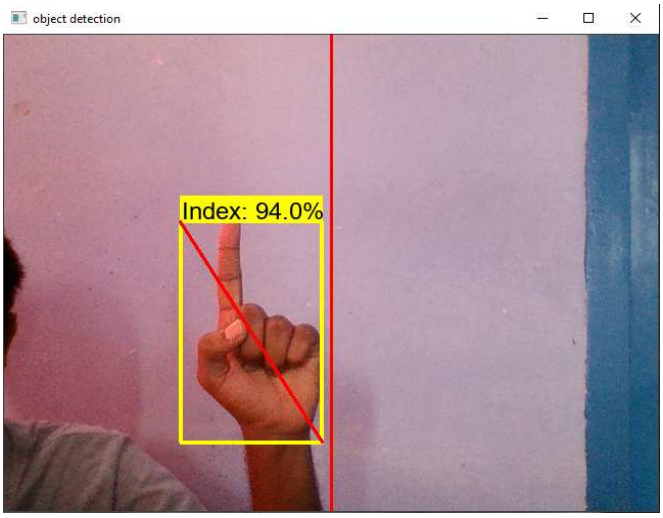

Fig 4.1 Index detected with 94% chance

Index finger pointed upwards will be used to move the mouse pointer as seen above the red diagonal line indicates the reference point and the intersection point of both diagonals will be treated as reference point and any change in the position of reference point will be reflected on the mouse pointer, with this gesture user can move the mouse pointer. Thumb pointed left will perform left click and only after a period of 1.6 seconds we can again perform left click. Thumb pointed right will make the right click same 1.6 seconds delay will be there for right click also.

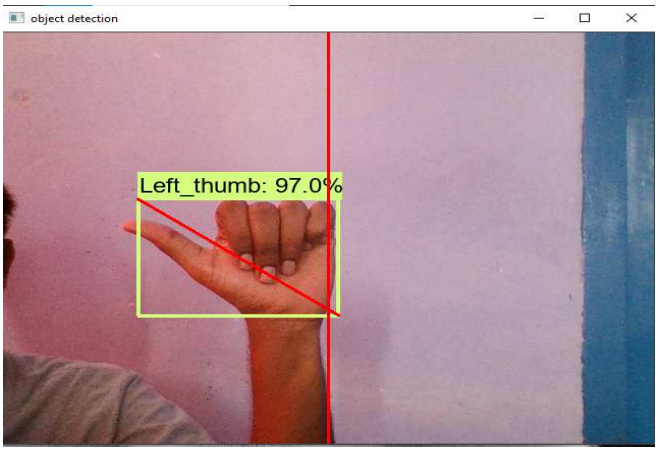

Fig 4.2 Left Thumb detected with 97% chance

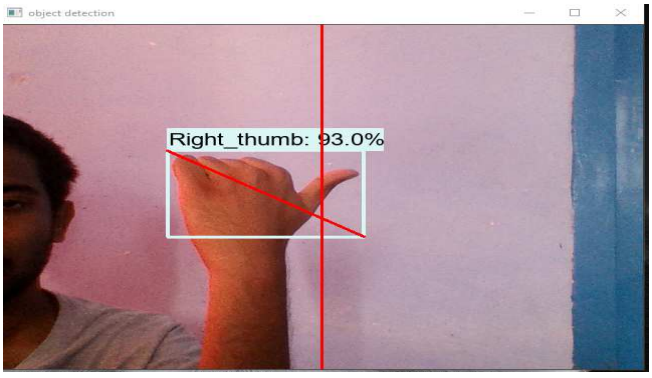

Fig 4.3 Right thumb detected with 93% chance

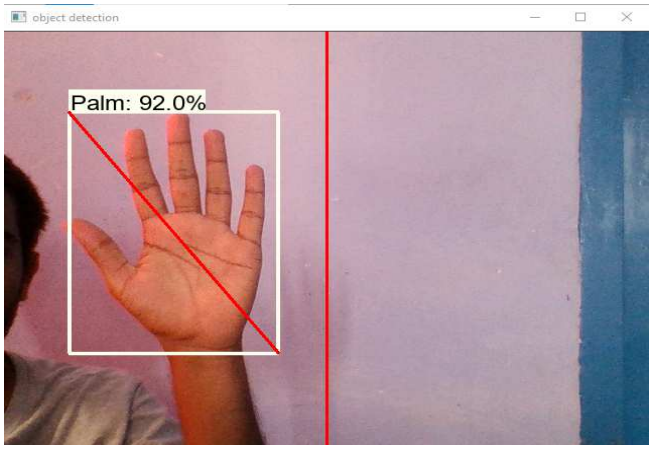

Fig 4.4 Palm on left half detected with 92% chance

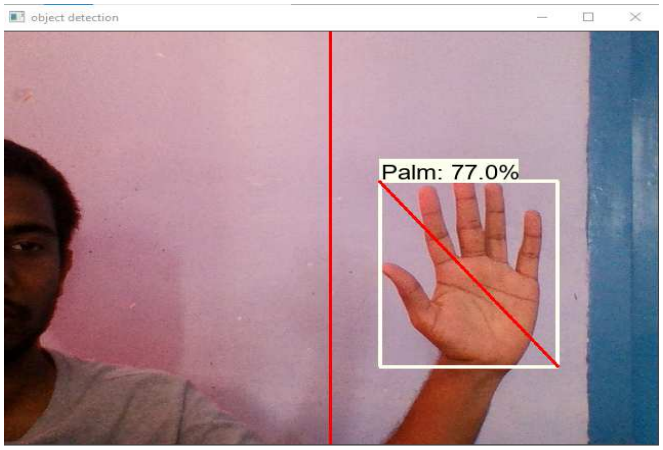

Fig 4.5 Palm on right side detected with 77% chance

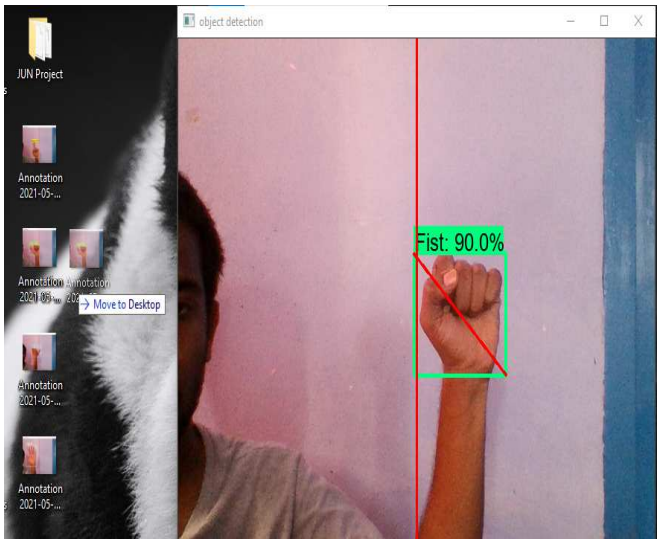

Fig 4.6 Fist detected with 90% chance

V. CONCLUSION & FUTURE SCOPE

With the Proposed gesture recognition and GUI Automation we expect to provide an easy-to-use tool with high accuracy in both plain and non-plain background and less latency where the user does not require any other external medium like glove, sensor, Color caps. The Images are processed in RGB format only each frame we detect the gesture with maximum probability with score above 50% and perform the respective GUI Automation task of mouse control. All the gestures used are chosen such that they are user friendly. As stated in Research papers there wasn't much latency with this approach. Moreover, using TensorFlow makes it to be user friendly.

This solution implemented using gestures is of low cost compared to the data glove and motion detecting camera implementation which is in favor of gesture recognition application and promotion. The Human computer Interaction system developed using gesture recognition and GUI Automation is more convenient and intelligent. In future the no of training images can be reduced or even avoided with efficient image preprocessing. Only one application can access camera at a time this problem can be resolved.

References

[1] Jeon, C., Kwon, O.-J., Shin, D., & Shin, D. (2017). *Hand-Mouse Interface Using Virtual Monitor Concept for Natural Interaction. IEEE Access*.

[2] Salunke, T. P., & Bharkad, S. D. (2017). *Power point control using hand gesture recognition based on hog feature extraction and k-nn classification. 2017 International Conference on Computing Methodologies and Communication (ICCMC).*

[3] Ghute, M. S., Anjum, M., & Kamble, K. P. (2018). Gesture Based Mouse Control. 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA).

[4] V. Gajjar, V. Mavani and A. Gurnani, "Hand gesture real time paint tool-box: Machine learning approach," 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), 2017, pp. 856-860.

[5] M. Shetty, C. A. Daniel, M. K. Bhatkar and O. P. Lopes, "Virtual Mouse Using Object Tracking," 2020 5th International Conference on Communication and Electronics Systems (ICCES), 2020, pp. 548-553.

[6] T. P. Salunke and S. D. Bharkad, "Power point control using hand gesture recognition based on hog feature extraction and k-nn classification," 2017 International Conference on Computing Methodologies and Communication (ICCMC), 2017, pp. 1151-1155.

[7] S. Song, D. Yan and Y. Xie, "Design of control system based on hand gesture recognition," 2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC), 2018, pp. 1-4.

[8] R. Titlee, A. U. Rahman, H. U. Zaman and H. A. Rahman, "A novel design of an intangible hand gesture controlled computer mouse using vision based image processing," 2017 3rd International Conference on Electrical Information and Communication Technology (EICT), 2017, pp. 1-4.

[9] Y. Adepu, V. R. Boga and S. U, "Interviewee Performance Analyzer Using Facial Emotion Recognition and Speech Fluency Recognition," 2020 IEEE International Conference for Innovation in Technology (INOCON), 2020, pp. 1-5.

[10] Santhosh Voruganti, U. Sairam ,"Digital Image Watermarking using ChaoticEncryption and Arnold Transform" 2021 International Journal for Research in Applied Science & Engineering Technology,2021,pp.1825-1835.