Gesture Based Gaming Control

Submitted in partial fulfillment of the requirements
of the Mini-Project 1 for Second Year

Bachelors of Engineering

by

Qureshi Afifa Arif	Roll No.38
Shaikh Zahir Mohammad	Roll No.56
Sheikh Faizan Manawer Ali	Roll No.58
Khan Ainan Ahmed Shabih Ahmed	Roll No.23

Guide:

Prof. Aishwarya Kamat

Department of Artificial Intelligence & Data Science
Rizvi College of Engineering

University of Mumbai

2024-2025

CERTIFICATE

This is to certify that the mini-project entitled **"Gesture Based Gaming Control"** is a bonafide work of "**Qureshi Afifa Arif (38), Shaikh Zahir Mohammad (56), Sheikh Faizan Manawer Ali (58), Khan Ainan Ahmed Shabih Ahmed (23)"** submitted to the University of Mumbai in partial fulfillment of the requirement for the Mini-Project 1 for Second Year of the Bachelor of Engineering in **"Artificial Intelligence & Data Science"**.

_____

Prof. Aishwarya Kamat

Guide

_____

Prof. Junaid Mandviwala

Head of Department

_____

Dr. Varsha Shah

Principal

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signatures)

Qureshi Afifa Arif	Roll No.38
Shaikh Zahir Mohammad	Roll No.56
Sheikh Faizan Manawer Ali	Roll No.58
Khan Ainan Ahmed Shabih Ahmed	Roll No.23

Date:

ABSTRACT

Hand and face gestures will be recognized using computer vision techniques such as Convolutional Neural Networks (CNNs) for image analysis. This will enable thesystem to accurately interpret user gestures and map them to specific in-game actions. Hand gesture recognition is part of Human- Computer Interaction as a place for the user interface to be presented. Various machine learning algorithms, including but not limited to, deep learning models (CNNs and Recurrent Neural Networks), decision trees, and ensemble methods, will be explored to classify and interpret theextracted features, translating them into meaningful game commands. Low latency processing will be a priority to ensure that the system responds quickly to user gestures, maintaining a seamless gaming experience. In summary, this research project aims to develop an innovative gesture-based game control model that leverages handand face gestures, integrating computer vision and machine learning techniques to provide an immersive and responsive gaming experience. This model holds the potential to reshape the future of gaming by offering a more natural and intuitive meansof interaction, enhancing user engagement and enjoyment in the gaming world objects.

Keywords: Gaming, Gesture-Based Gaming, Hand Gestures, Face Gestures, Gesture Recognition, Computer Vision, Machine Learning.

Index

List of Figures

Chapter 1
Introduction

PROBLEM STATEMENT -

To design and develop a system for virtual gesture-based gaming that recognizes hand gestures and face gestures using machine learning algorithm.

SCOPE -

The scope of this project is restricted; to do various operations, we used handgestures and facial movements. However, in the future, we may include moreadvanced features for operations.

Gesture recognition is a technique which is used to understand and analyze the human body language and interact with the user accordingly. This in turn helps in building a bridge between the machine and the user to communicate with each other. Gesture recognition is useful in processing the information which cannot be conveyed through speech or text. Gestures are the simplest means of communicating something that is meaningful. This paper involves implementation of the system that aims to design a vision-based hand gesture recognition system with a high correct detection rate along with a high-performance criterion, which can work in a real time Human Computer Interaction system without having any of the limitations (gloves, uniform background etc.) on the user environment. The system can be defined using a flowchart that contains three main steps, they are: Learning, Detection, Recognition. Using gestures instead of standard input devices, gesture-based gaming has become a novel and engaging method to engage with digital surroundings. With the use of cutting-edge convolutional neural network (CNN) algorithms and OpenCV (Open Source Computer Vision Library), this state-of-the-art technology interprets hand and facial motions to enable users to control and explore a gaming environment.

Developers may establish a smooth interface between the user's physical motions and the virtual environment of a game by utilizing OpenCV'scapabilities. By using CNN algorithms, the system can identify intricate pat- terns and variations in hand and facial motions, therefore elevating gesture- based gaming to a new level.

Chapter 2
Review of Literature

2.1 A Case Study of User Experience on Hand-Gesture Video Games

Abstract: As motion capture systems become reliable and affordable, hand gestures can be identified in real-time and used in video games as alternative game controls. However, there is not enough studies about user experience using gesture input modality for games. In this work, we perform a case study of hand-gesture video games. We try to answer the question "Do players have a delightful experience when they interact with the game using hand gestures?". We map intuitive hand gestures to commands and use them to control the movement of game characters. We perform a user study based on User Experience Questionnaires (UEQ). In comparing to all interactive products registered in UEQ study database, our gesture control interface scores in the top 1% range showing that gesture-based gaming can provide a high level of user satisfaction.

2.2 Visual Hand Gesture Interface for Computer Board Game Control

Abstract: In computer vision interface for interactive computer game, gestures can be used as commands for the game control instead of keyboard or mouse. The utilization of hand gesture in human-computer interaction enables human operators to interact with computer environments in a natural and intuitive manner. This paper proposes visual gesture recognition for manipulating objects in a board game such as chess. The distance between a player's thumb and index finger is used to determine whether the player is in the state of holding a playing object or not. A hand holding an object is tracked until it is stopped and put down the object by widening the distance between the thumb and the finger. The thumb-index distance is measured using the distance between two consecutive peaks of hand contour. The Kalman-filter-based prediction is applied for tracking the finger. The results show high accuracy in objecting picking, hand tracking, and object placing.

2.3 Vision-Based Finger Guessing Game in Human Machine Interaction

Abstract: This study uses motion hand gesture recognition to demonstrate a finger guessing game in human machine interaction. By analyzing body movements at the moment of signing to determine the timing of the signing gesture completion, and through imaging analysis and recognition, this game system can categorize the signs of the users, thus allowing no need of the users for the button, keyboard, mouse, or any special device for the input interface, but simply natural body movement to show the hand gestures required by the finger guessing game. Such human machine interaction provides more sense of reality and better interaction during game play between human and machines. Upon the recognition of user's hand gesture, this finger guessing game system uses the simple tabular cross-linking method to map it against the sign randomly selected by the computer, and further determine the game result. The robot used for interacting with the human is a mobile platform for an autonomic DSP (Digital Signal Processor) closed loop positioning controller, which demonstrates a simple human machine interactive game, by showing the results of finger guessing games for the players and robot. As indicated by test result, this motion hand gesture recognition technology allows the users to interact and play games with robots in ways that are more realistic and natural, thus providing more enjoyment for people in an effective way.

2.4 A Hand Gesture Set for Navigating and Interacting with 3D Virtual Environments

Abstract: Recent advances in consumer-level VR technologies have increased interest in using hand gestures to interact more intuitively with computers. Unfortunately, there are still few accepted norms for gestures that map to functions, and identifying suitable hand gestures for a given use case is difficult. In this paper we review research on how to define a good gesture set. Based on this we develop and evaluate a hand gesture set for controlling and navigating interactive virtual environments, such as games or simulations. A user study shows that the resulting gesture set is intuitive, easy to perform and remember, and overall likeable. The lowest scores were received for gestures for changing the users's location.

Chapter 3

3.1 Theory

3.1.1 Project Implementation

The proposed framework begins with Information assortment of different pictures through certain means and afterward at long last predicts the inscription from picture. The means included are as per the following: Information Assortment, Picture Pre-handling, Component Extraction, Arrangement.

Preprocessing step: This step is a vital stage in AI. Pre-processing comprises of embedding the missing qualities, the fitting information range, and removing the usefulness.

Feature Selection: Feature extraction should simplify the amount of datainvolved to represent a large data input.

Programming Language: **Python**

Python is a deciphered, significant level, broadly useful programming language. Made by Guido van Rossum and first delivered in 1991, Python has a plan reasoning that underscores code clarity, outstandingly utilizing significant whitespace. It gives builds that empower clear programming on both little and enormous scopes. Van Rossum drove the language local area until July 2018. Python is powerfully composed and trash gathered.

3.1.2 System Architectural Design

The architecture illustrates a system architecture for gesture recognition using hand tracking. It begins with hand movements in front of a webcam, where the camera captures real-time images of the hand's gestures. These captured images form the input for the gesture recognition system. The input image is then passed through a pre-processing step, which involves techniques such as noise removal, normalization, and possibly resizing or color adjustments to ensure the image is in a suitable format for further processing. Pre-processing enhances the quality of the image and makes the subsequent feature extraction phase more accurate.

After pre-processing, the system extracts relevant features from the image. Feature extraction refers to identifying key aspects of the hand gesture, such as shape, orientation, or specific hand landmarks (like fingertips or palm edges). These extracted features are crucial as they represent the data required to recognize gestures accurately.

The processed data is then fed into a gesture recognition and hand-tracking system, which uses algorithms or machine learning models to match the input with known hand gestures.

Finally, the recognized hand gesture is outputted, completing the recognition process. This architecture is ideal for human-computer interaction, enabling control of devices using hand gestures.

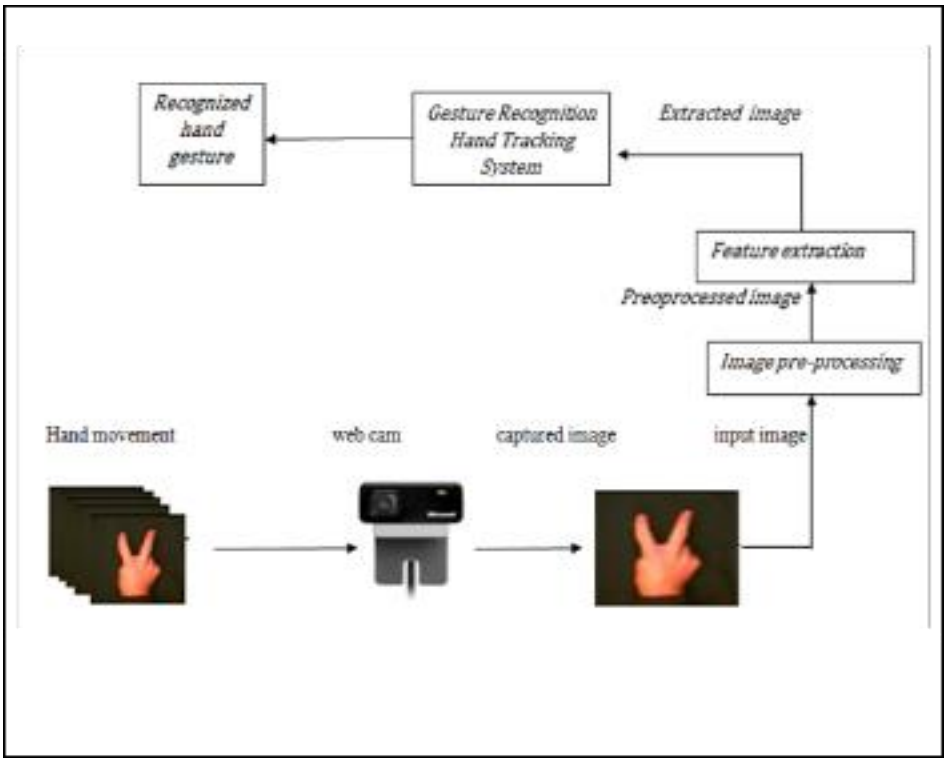


Figure 1: System Architecture

The architecture shown in the image depicts a gesture recognition system that tracks hand movements to interpret gestures. It begins by capturing images of hand movements through a webcam. These images, referred to as "input images," are taken in real-time and passed through the next stage of processing. The captured image is a visual representation of the user's hand making specific gestures.

The next stage involves image pre-processing and feature extraction. During image pre-processing, the captured images undergo certain modifications, like resizing, smoothing, or enhancing, to make them suitable for further analysis. Afterward, feature extraction isolates critical information from the pre-processed image, such as shapes, contours, or specific hand features that can help identify the gesture. This step generates an "extracted image" that is passed to the gesture recognition system.

Finally, the "Gesture Recognition Hand Tracking System" uses the extracted features to recognize the specific gesture being performed. It analyzes the patterns and compares them with pre-defined gesture models to determine the corresponding gesture.

The proposed system undergoes some modules such as:

- Webcam: A standard webcam is utilized to catch video input. Higher goaland approach rates can give more exact and responsive motion acknowledgement.

- Computer: The webcam is associated with a PC where the picture handling and signal acknowledgment calculations are executed.

- Frame Capture: The webcam catches a progression of video outlines at aspecific edge rate (outlines each second or FPS). Each edge is basically a still picture. The edges might go through preprocessing moves toward upgrade highlights, lessen commotion, or change lighting conditions.

- Feature Extraction: Applicable highlights like tone, shape, or movement are extricated from the preprocessed outlines.

- API Integration: The motion acknowledgment framework necessities to speak with the gaming application, frequently through an application programming connection point or some type of middleware.

- Gesture Classification: Whenever highlights are extricated and followingis played out, a characterization calculation decides the sort of signal being performed. This could include AI procedures like brain organizationsor customary PC vision calculations.

3.1.3 Requirements

H/W (Hardware) Requirements:

- Processor- Pentium IV/Intel I3 core An Intel Corei3 is an Intel proprietary processor that is built on the frame-work of multiprocessor architecture. It is a type of dual-core processor with an integrated graphic processing unit (GPU). It is a successor of theCore 2 series of processors produced by Intel.

- RAM- 4GB (min) It means a computer has four (4) gigabytes of Random Access Memory (RAM). This is the memory that is used by the computer to store and runall of its tasks including the operating system and all of its applications. It is simple a measure of total memory capacity of the machine.

- Hard Disk- 500GB Currently, hard drives are divided into 4 major types i.e Parallel Advanced Technology Attachment (PATA) Serial Advanced Technology Attachment (SATA) Small Computer System Interface (SCSI) Solid State Drive (SSD)

S/W (Software) Requirements:

- Tools - Python IDE (Integrated Development Environment)

- Programming Language – Python

- Version - Python 3.6.8

3.2 Methodology

3.2.1 Existing Systems for Gesture-Based Recognition

Before the advent of sophisticated deep learning techniques like Convolutional Neural Networks (CNNs), a variety of machine learning and image processing algorithms were used for gesture-based recognition. These traditional systems relied on more straightforward algorithms, feature engineering, and predefined rules. They were limited in their ability to handle complex gestures or variations in lighting, background, and hand orientation.

3.2.2 Support Vector Machines (SVM)

SVM which was a widely used algorithm for gesture recognition before deep learning models like Convolutional Neural Networks (CNNs) gained prominence. SVMs have several strengths, but they also have critical drawbacks, which led to the adoption of CNNs in gesture recognition tasks, especially in areas like gesture-based gaming control.

3.2.3 Overview of SVM

Support Vector Machines (SVM) is a powerful supervised learning algorithm that is primarily used for classification tasks. It works by finding the optimal hyperplane that best separates different classes in a feature space. The idea behind SVM is to maximize the margin between data points of different classes, so that the data is separated as clearly as possible. In gesture recognition systems, the algorithm relies on extracting features from images or sequences, which are then fed into the SVM model for classification

Feature Engineering Requirements

One major limitation of SVM is that it requires manual feature engineering. In traditional SVM-based gesture recognition systems, engineers had to manually design features such as the edges of the hand, contours, or geometric features like distances between fingers. These features were then used as inputs to the SVM model. This process was not only time-consuming but also error-prone. If the features were not properly designed, the system could fail to recognize gestures accurately.

Difficulty in Scaling to Complex Gestures

Another drawback of SVM is its limited ability to scale well to complex, multi-class classification problems. In gesture-based gaming, users may perform a variety of gestures, and the system needs to distinguish between subtle differences in hand movements. While SVM works well for binary classification, its performance degrades when applied to multi-class tasks, which are common in gesture recognition. SVM typically handles multi-class problems by breaking them down into multiple binary classification tasks, which increases computational complexity and reduces performance in terms of both speed and accuracy.

Limitations with Non-linear Data

Although SVMs can handle non-linear data using kernel functions (like the radial basis function, or RBF), they still face challenges when dealing with highly complex, non-linear gesture recognition tasks. Gesture movements are inherently dynamic and non-linear, making it difficult for SVM to separate classes with simple kernel functions. While kernel methods can improve performance, they further increase computational complexity and require manual tuning to optimize.

Real-Time Recognition Challenges

In gesture-based gaming, real-time recognition is essential to provide a smooth and responsive user experience. SVMs, however, struggle with real-time recognition due to their high computational requirements. As mentioned earlier, SVM models trained on large datasets are slow to compute, especially in situations where multiple binary classifiers are needed for multi-class recognition. The need for fast processing is especially critical in gaming environments, where even a slight delay in recognizing gestures can negatively impact the player's experience.

Why CNNs are Better for Gesture-Based Gaming Control

CNNs are now the preferred choice for gesture recognition in applications like gesture-based gaming control due to their ability to automatically learn features, handle complex multi-class problems, and deliver real-time performance. They are highly flexible and can be trained on large datasets to recognize a wide variety of gestures. Additionally, CNNs can be deployed efficiently on modern hardware, such as GPUs, enabling them to process large amounts of data quickly and provide real-time gesture recognition.

3.3 Algorithm

3.3.1 Convolutional Neural Networks

CNN or the convolutional brain organization (CNN) is a class of profound learning brain organizations. In short consider CNN an AI calculation thatcan take in an information picture, dole out significance (learnable loadsand predispositions) to different perspectives/objects in the picture, and have the option to separate one from the other.

The job of CNN is to decrease the pictures into a structure that is simpler to process, without losing highlights basic towards a decent forecast.This is significant when we really want to make the calculation adaptable. It is vital to figure out that ANN or Fake Brain Organizations, comprised of various neurons isn't equipped for removing highlights from the picture. This is where a mix of convolution and pooling layers comes into the image.

CNN works by extracting features from the images. Any CNN consists of the following:

- The input layer which is a grayscale image
- The Output layer which is a binary or multi-class labels
- Hidden layers consisting of convolution layers, ReLU (Rectified Linear Unit) layers, the pooling layers, and a fully connected Neural Network

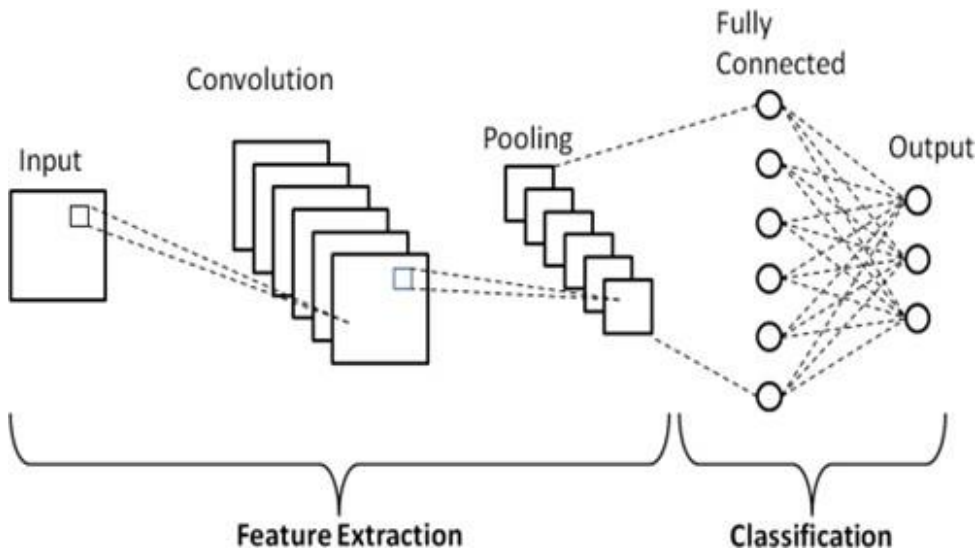


Figure 2: CNN Architecture

A CNN is an exceptional instance of the brain network depicted previously. A CNN comprises of at least one convolutional layers frequently with a subsampling layer, which are trailed by at least one completely associated layers as in a standard brain organization.

In the brain network computational model, the signs that movement alongthe axons (e.g., x0) communicate multiplicatively (e.g., w0x0) with the dendrites of the other neuron in light of the synaptic strength at that neuro-transmitter (e.g., w0). Synaptic loads are learnable and control the impactof some neuron. The dendrites convey the sign to the cell body, where they all are added. In the event that the last total is over a predefined edge,the neuron fires, sending a spike along its axon.

In the computational model, it is expected that the exact timings of the terminating don't make any difference and just the recurrence of the terminating imparts data. Inlight of the rate code translation, the terminating pace of the neuron is demonstrated with an actuation capability f that addresses the recurrenceof the spikes along the axon.

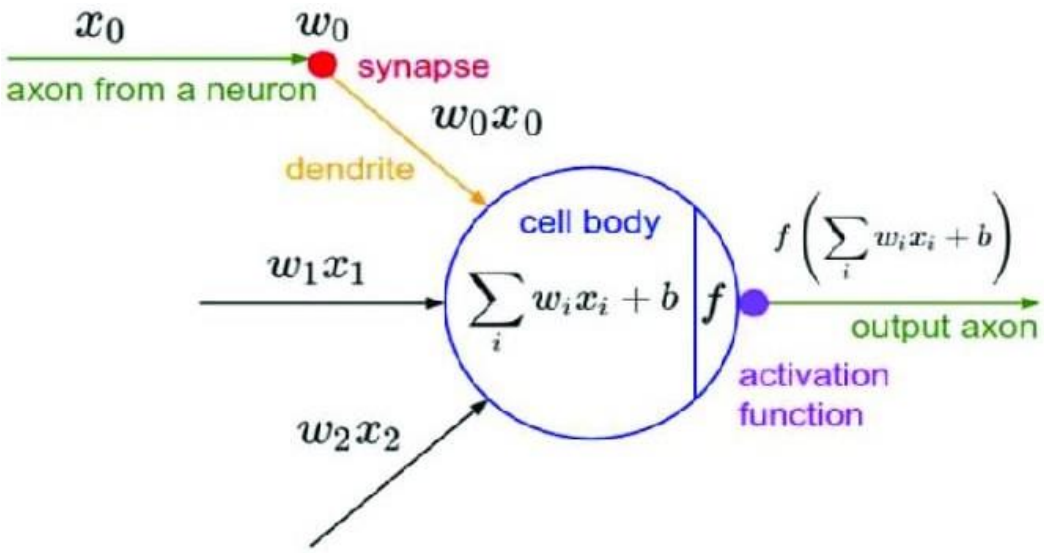


Figure 3: llustration of a Mathematical Model

In a genuine creature brain framework, a neuron is seen to get input signals from its dendrites and delivering yield signals along its axon. The axon branches out and associates through neurotransmitters to dendrites of different neurons.

At the point when the mix of info signals arrives at some limit condition among its feedback dendrites, the neuron is set off and its actuation is imparted to replacement neurons.

3.3.2 Layers of CNN

By stacking various and various layers in a CNN, complex structures areworked for grouping issues.

- Convolutional Layer- Assuming we notice Figure cautiously we will see that the bit shifts multiple times across picture. This cycle is called Step. At the point when weutilize a step worth of 1 we want 9 cycles to coverthe whole picture. The CNN learns the loads of these Bits all alone.

- Fully connected layers- Completely associated layers are many times utilized as the last layers ofa CNN. These layers numerically total a weighting of the past layer of elements, demonstrating the exact blend of 'fixings' to decide a particular objective result. In the event of a completely associated layer, every one of the components of the relative multitude of highlights of the past layer get utilized in the computation of every component of each result highlight.

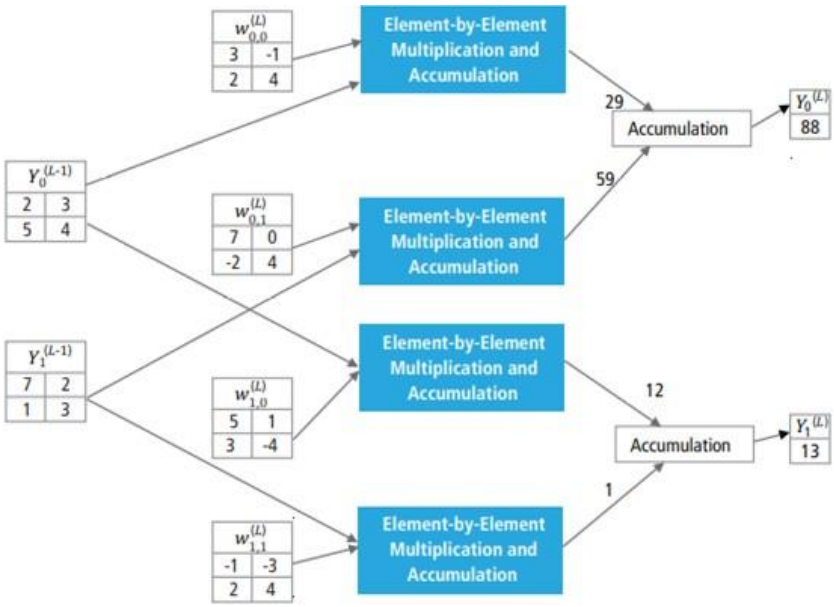


Figure 4: Process of Fully Connected Layer

Chapter 4

Results and Discussions

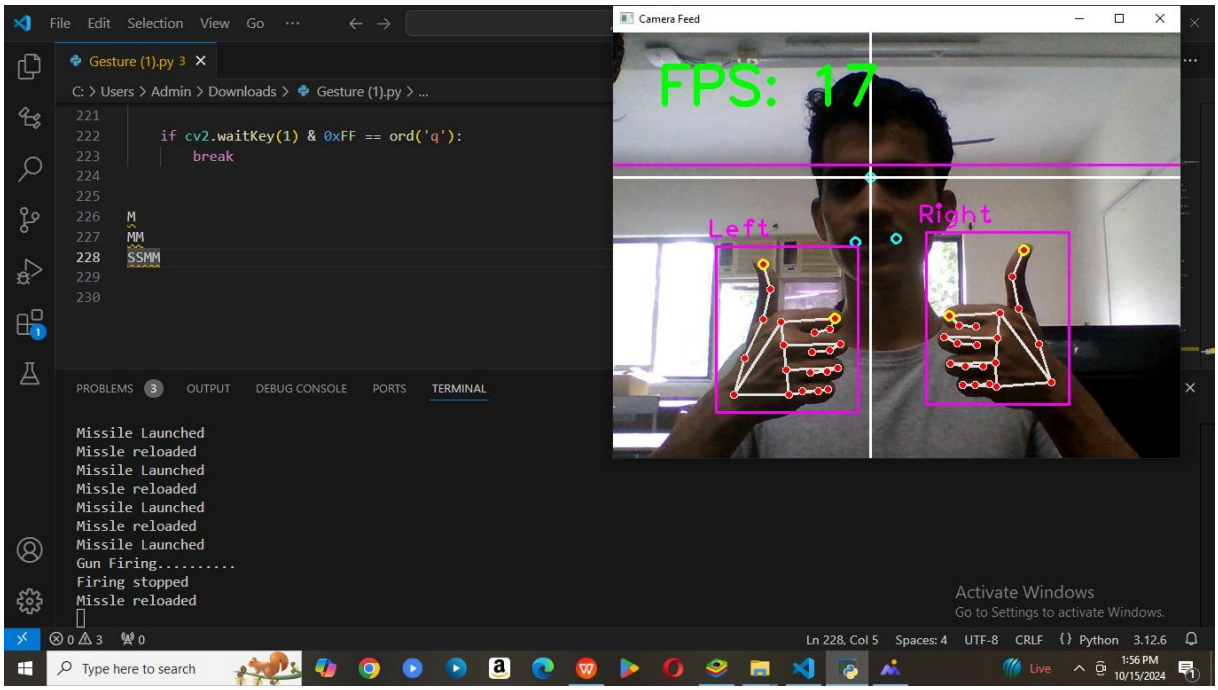


Figure 5: Missile Reloading Gesture

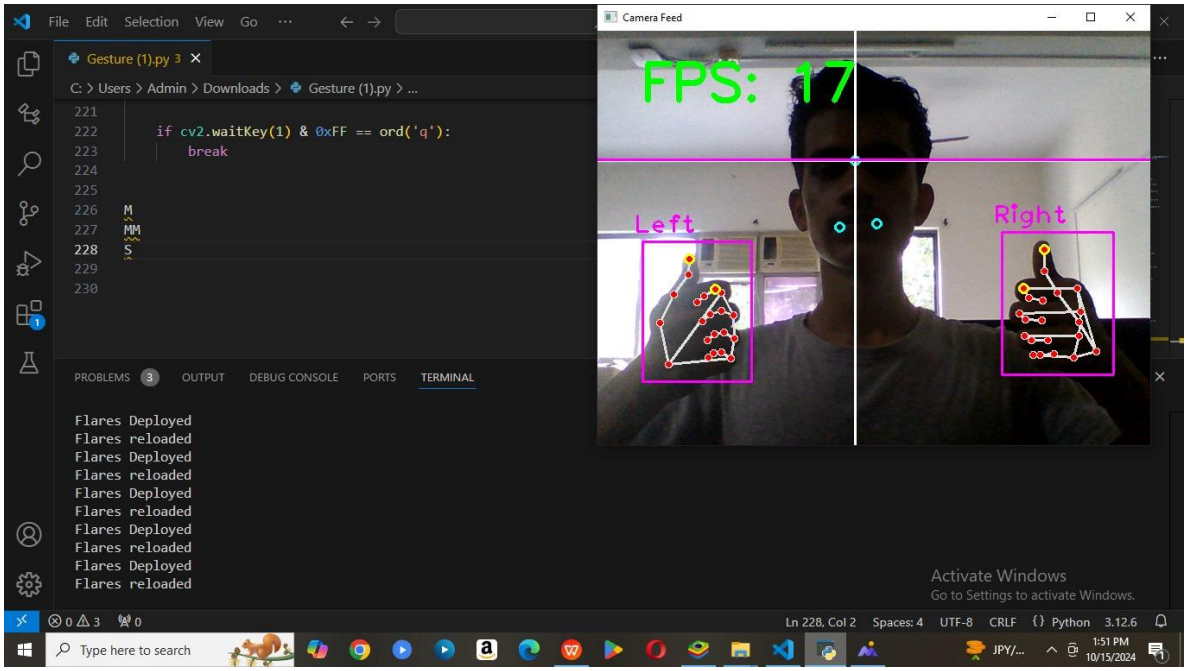


Figure 6: Missile Launch Gesture

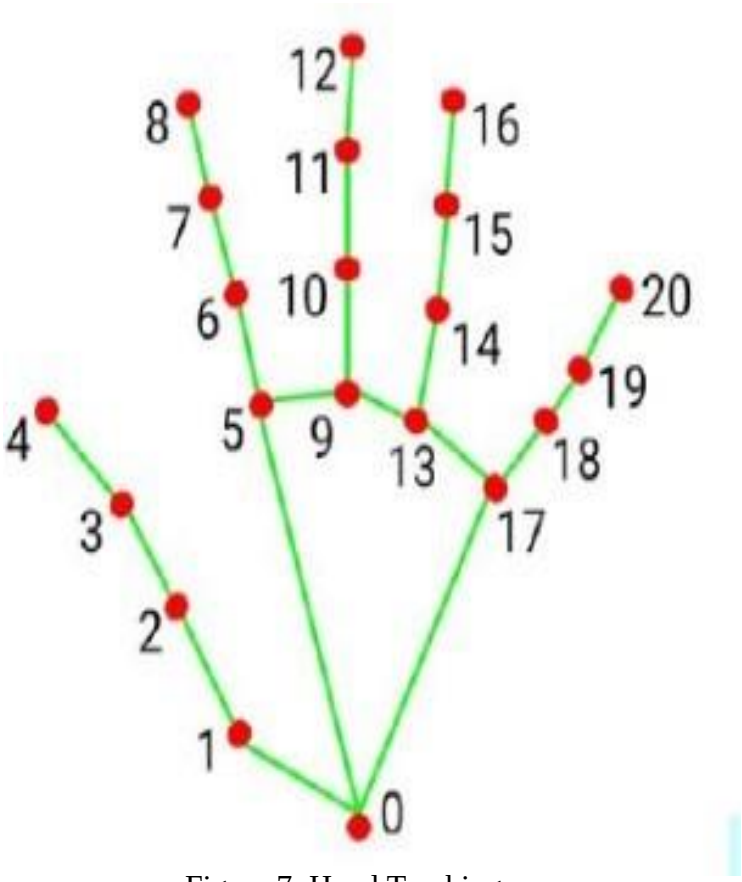


Figure 7: Hand Tracking

Hand Tracking: Cameras or sensors detect the overall position of the hand in 3D space. The system creates a digital representation of the hand skeleton, identifying keypoints (e.g., joints and fingertips) and their relative positions.

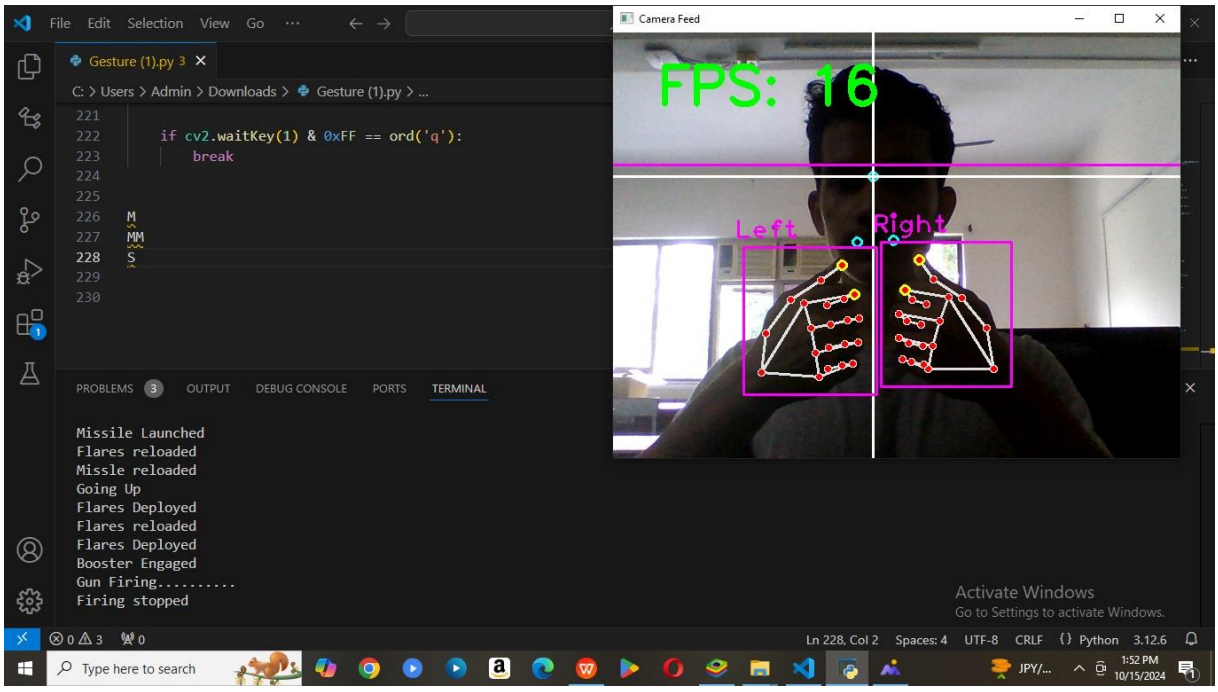


Figure 8: Booster Gesture

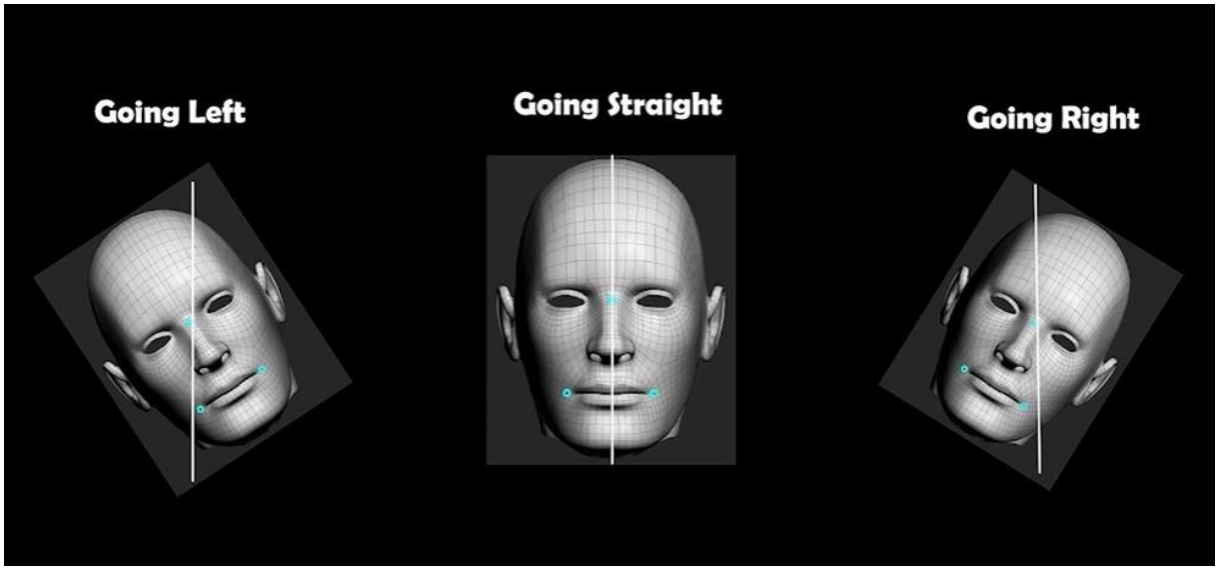
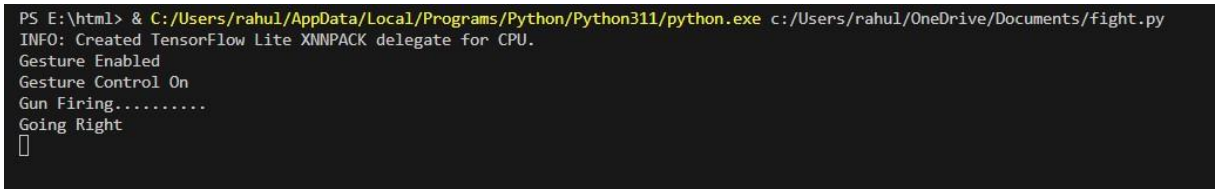


Figure 9: Facial Movement Gesture

If the player tilts their face to the left, the aircraft in the game will respond by moving to the left. If the player tilts their face to the right, the aircraft will move to the right.



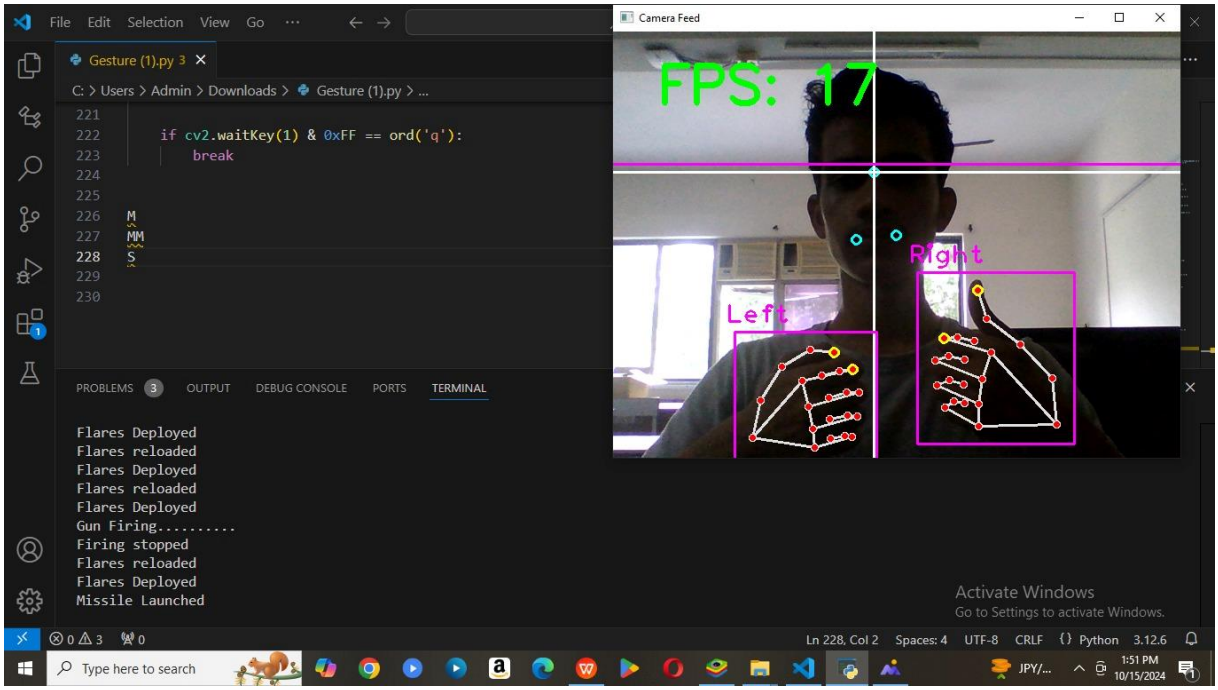


Figure 10: Flares Deploy Gesture

Chapter 5

Conclusions

The gesture-based gaming control system utilizes a webcam to capture real-time hand movements, which are then transformed into images. These captured images are processed and prepped for further analysis through image preprocessing steps, ensuring the data is clean and usable.

Next, the system extracts relevant features from the preprocessed images, focusing on characteristics like hand contours and finger positions. These features are essential for accurately recognizing specific gestures, which are then mapped to corresponding commands for gaming control.

Finally, the recognized gestures allow for smooth interaction with the game, where hand movements are translated into in-game actions. The system is designed for real-time performance, ensuring minimal delay between gesture recognition and the execution of commands in the gaming environment. This allows for an intuitive and immersive gaming experience driven by hand gestures.

Chapter 6

References

Journal Papers,

1. Rafi Aziizi Muchtar Department of Informatics Universitas Jenderal Achmad Yani Cimahi, Indonesia rafiaziizim18@if.unjani.ac.id
2. D. RIANA, “Deep Neural Network for Click-Through Rate Prediction,” Interna- tional Journal of Software Engineering and Computer Systems, vol. 8, no. 2, pp. 33–42, 2022
3. W.Ye, J. Cheng, F. Yang, and Y. Xu, “Two-Stream Convolutional Network for Improving Activity Recognition Using Convolutional Long Short-Term Memory Networks,” 2022D. Avola et al., “MS-Faster R-CNN: Multi-Stream Backbone for Improved FasterR CNN Object Detection and Aerial Tracking from UAV Images,” pp. 1–18, 2021.
4. J. Zou, W. Yuan, and M. Yu, “Maritime Target Detection of Intelligent Ship Basedon Faster R-CNN,” Proceedings - 2019 Chinese Automation Congress, CAC 2019
5. A.Gupta and M. S. Balan, ”Action recognition from optical flow visualizations”, Proceedings of 2nd International Conference on Computer Vision Image Processing, pp. 397-408, 2018
6. S. M. S. Shajideen and V. H. Preetha, “Human-computer interaction system using2D and 3D hand gestures,” 2018 International Conference on Emerging Trends and Innovations In Engineering And Technological Research, ICETIETR 2018
7. Ayushi Bhawal, Debaparna Dasgupta, Arka Ghosh, Koyena Mitra, Surajit Basak, “Arduino Based Hand Gesture Control of Computer”, IJESC, Volume 10, Issue No.6, June 2020.
8. Udit Kumar, Sanjana Kintali, Kolla Sai Latha, Asraf Ali, N. Suresh Kumar, “Hand Gesture Controlled Laptop Using Arduino”, April 2020.
9. Sarita K., Gavale Yogesh, S. Jadhav, “HAND GESTURE DETECTION USING ARDUINO AND PYTHON FOR SCREEN CONTROL”, International Journal of Engineering Applied Sciences and Technology, 2020, Vol. 5, Issue 3, ISSN No. 2455-2143, July 2020.
10. Rohit Mukherjee, Pradeep Swethen, Ruman Pasha, Sachin Singh Rawat, “Hand Gestured Controlled laptop using Arduino”, International Journal of Management, Technology And Engineering, October 2018.

Appendix

```
import cv2
import time
from cvzone.FaceMeshModule import FaceMeshDetector
from cvzone.HandTrackingModule import HandDetector
import threading
import time
import pyautogui




# Initialize the video capture
cap = cv2.VideoCapture(0)  # Use '0' if '1' is not working
cap.set(3, 640)  # Set frame width
cap.set(4, 480)  # Set frame height
# Variables for FPS calculation
prev_time = 0


detector1 = FaceMeshDetector(maxFaces=1)
detector2 = HandDetector(maxHands=2, detectionCon=0.75)



go_right = 0
go_left = 0
go_up = 0
go_down = 0
ges_ctrl_track = 0
gesture_on = None
misle_coldwn = 0
flare_coldwn = 0
fire_on = 0
booster_coldwn =0



def ges_enable():
    global gesture_on
    global gesture_on_thread
    time.sleep(3)
    gesture_on = 1
    print("Gesture Control On")
    gesture_on_thread = threading.Thread(target=ges_enable)

def misle_cooldown_func():
```

```
    global misle_coldwn
    global misle_coldwn_thread
    time.sleep(2)
    misle_coldwn = 0
    print("Missle reloaded")
    misle_coldwn_thread = threading.Thread(target=misle_cooldown_func)

def flare_coldwn_func():
    global flare_coldwn
    global flare_coldwn_thread
    time.sleep(2)
    flare_coldwn = 0
    print("Flares reloaded")
    flare_coldwn_thread = threading.Thread(target=flare_coldwn_func)

def booster_coldwn_func():
    global booster_coldwn
    global booster_coldwn_thread
    time.sleep(2)
    booster_coldwn = 0
    print("Boosters cooling down")
    booster_coldwn_thread = threading.Thread(target=booster_coldwn_func)




gesture_on_thread = threading.Thread(target=ges_enable)
misle_coldwn_thread = threading.Thread(target=misle_cooldown_func)
flare_coldwn_thread = threading.Thread(target=flare_coldwn_func)
booster_coldwn_thread = threading.Thread(target=booster_coldwn_func)



while True:
    success, img = cap.read()
    img = cv2.flip(img, 1)  # Flip the frame horizontally
    img, faces = detector1.findFaceMesh(img, draw=False)


    if faces:
            ctrl_x, ctrl_y = faces[0][168][0], faces[0][168][1]
            left_x, left_y = faces[0][57][0] + 10, faces[0][57][1]
            right_x, right_y = faces[0][287][0] - 10, faces[0][287][1]

            cv2.circle(img, (ctrl_x, ctrl_y), 5, (255, 255, 0), 2)
            cv2.circle(img, (left_x, left_y), 5, (255, 255, 0), 2)
            cv2.circle(img, (right_x, right_y), 5, (255, 255, 0), 2)
            cv2.line(img, (ctrl_x, 0), (ctrl_x, 1000), (255, 255, 255), 2)
            cv2.line(img, (0, ctrl_y), (1000, ctrl_y), (255, 255, 255), 2)

            hands, img = detector2.findHands(img, draw=True, flipType=False)
            if len(hands) == 2:
```

```
                if hands[0]['type'] == "Left":
                    hands_l, hands_r = hands[0], hands[1]
                else:
                    hands_l, hands_r = hands[1], hands[0]

                lmlist_l, lmlist_r = hands_l['lmList'], hands_r['lmList']

                r_idx_x, r_idx_y = lmlist_r[6][0], lmlist_r[6][1]
                l_idx_x, l_idx_y = lmlist_l[6][0], lmlist_l[6][1]
                r_thmb_x, r_thmb_y = lmlist_r[4][0], lmlist_r[4][1]
                l_thmb_x, l_thmb_y = lmlist_l[4][0], lmlist_l[4][1]

                cv2.circle(img, (r_idx_x, r_idx_y), 5, (0, 255, 255), 2)
                cv2.circle(img, (l_idx_x, l_idx_y), 5, (0, 255, 255), 2)
                cv2.circle(img, (r_thmb_x, r_thmb_y), 5, (0, 255, 255), 2)
                cv2.circle(img, (l_thmb_x, l_thmb_y), 5, (0, 255, 255), 2)


                l_fing_upno = detector2.fingersUp(hands_l)
                r_fing_upno = detector2.fingersUp(hands_r)


                if l_fing_upno[0] == 1 and l_fing_upno[4] == 1 and
r_fing_upno[0] == 1 and r_fing_upno[4] == 1 and \
                        sum(l_fing_upno) == 2 and sum(r_fing_upno) == 2 and
ges_ctrl_track == 0:

                    updwn_line_control = ctrl_y
                    ges_ctrl_track = 1
                    print("Gesture Enabled")
                    gesture_on_thread.start()




    if gesture_on == 1:
        cv2.line(img, (0, updwn_line_control), (1000, updwn_line_control),
(255, 0, 255), 2)

         # Aircraft Moving Up
        if abs(updwn_line_control - ctrl_y) > 20 and updwn_line_control >
ctrl_y and go_up == 0:
            pyautogui.keyDown('W')
            go_up = 1
            print("Going Up")
        elif abs(updwn_line_control - ctrl_y) < 20 and updwn_line_control >
ctrl_y and go_up == 1:
            pyautogui.keyUp('W')
            go_up = 0
```

```
        # Aircraft Moving down
        if abs(updwn_line_control - ctrl_y) > 35 and updwn_line_control <
ctrl_y and go_down == 0:
            pyautogui.keyDown('S')
            go_down = 1
            print("Going Down")
        elif abs(updwn_line_control - ctrl_y) < 35 and updwn_line_control <
ctrl_y and go_down == 1:
            pyautogui.keyUp('S')
            go_down = 0


        # Moving the Aircraft to the Left
        if right_x < ctrl_x and go_right == 0:
            pyautogui.keyDown('D')
            go_right = 1
            print('Going Right')

        # Moving the Aircraft to the Left
        elif left_x > ctrl_x and go_left == 0:
            pyautogui.keyDown('A')
            go_left = 1
            print('Going Left')

            # Keeping Aircraft StraightDDDDDD
        elif right_x > ctrl_x and left_x < ctrl_x:
            if go_right == 1:
                pyautogui.keyUp('D')
                go_right = 0
            if go_left == 1:
                pyautogui.keyUp('A')
                go_left = 0

        # Missile Launch
        if abs(r_thmb_y - r_idx_y) < 30 and abs(l_thmb_y - l_idx_y) < 30 and
misle_coldwn ==0:
                misle_coldwn = 1
                print("Missile Launched")
                pyautogui.press("up")
                misle_coldwn_thread.start()


        # Flares Deploying
        elif abs(r_thmb_y - r_idx_y) > 30 and abs(l_thmb_y - l_idx_y) < 30 and
flare_coldwn == 0:
            flare_coldwn = 1
            print("Flares Deployed")
            pyautogui.press("down")
            flare_coldwn_thread.start()

         # Aircraft's Gun Fire
```

```
        elif abs(r_thmb_y - r_idx_y) < 30 and abs(l_thmb_y - l_idx_y) > 30 and
fire_on == 0:
            pyautogui.keyDown("left")
            fire_on = 1
            print("Gun Firing.....")
        if abs(r_thmb_y - r_idx_y) > 30 and fire_on == 1:
            pyautogui.keyUp("left")
            fire_on = 0
            print("Firing stopped")

        # Engaging Speed Boosters
        if abs(r_idx_x - l_idx_x) < 50 and booster_coldwn == 0:
            pyautogui.press("right")
            booster_coldwn = 1
            print("Booster Engaged")
            booster_coldwn_thread.start()




    current_time = time.time()
    fps = 1 / (current_time - prev_time) if prev_time != 0 else 0
    prev_time = current_time


    cv2.putText(img, f'FPS: {int(fps)}', (50, 80), cv2.FONT_HERSHEY_SIMPLEX,
2, (0, 255, 0), 5)


    cv2.imshow("Camera Feed", img)


    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

Acknowledgements

I am profoundly grateful to PROF. AISHWARYA KAMAT for his expert guidance and continuous encouragement throughout to see that this project rights its target.

I would like to express deepest appreciation towards Dr. Varsha Shah, Principal RCOE, Mumbai and Prof. Junaid Mandviwala HOD Artificial Intelligence & Data Science Department whose invaluable guidance supported me in this project.

At last I must express my sincere heartfelt gratitude to all the staff members of Artificial Intelligence & Data Science Department who helped us directly or indirectly during this course of work.

Qureshi Afifa Arif	Roll No.38
Shaikh Zahir Mohammad	Roll No.56
Sheikh Faizan Manawer Ali	Roll No.58
Khan Ainan Ahmed Shabih Ahmed	Roll No.23

Publications

[Add you published research paper on this topic in any Conference / Journal.]