

REAL-TIME GAMING USING BODY GESTURES WITH MACHINE LEARNING

A MINI PROJECT REPORT

Submitted by

ANITHA R 212220040009

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING



SAVEETHA ENGINEERING COLLEGE, KANCHEEPURAM

ANNA UNIVERSITY: CHENNAI- 600 025

DECEMBER 2022

BONAFIDE CERTIFICATE

Certified that this Mini Project report **“REAL-TIME GAMING USING BODY GESTURES WITH MACHINE LEARNING”** is the bonafide work of **ANITHA R (212220040009)**, who carried out the mini project work under my supervision.

SIGNATURE

Dr. P. V. GOPIRAJAN, M.E., (Ph.D)

Assistant Professor

SUPERVISOR

Dept of Computer Science and Engineering,
Saveetha Engineering College,
Thandalam, Chennai 602105.

SIGNATURE

Dr. G. NAGAPPAN, M.E., PHD

Professor

HEAD OF THE DEPARTMENT

Dept of Computer Science and Engineering,
Saveetha Engineering College,
Thandalam, Chennai 602105.

DATE OF THE VIVA VOCE EXAMINATION:

INTERNAL EXAMINER

EXTERNAL EXAMINER

ACKNOWLEDGEMENT

We express our deep sense of gratitude to our honourable and beloved Founder President **Dr. N. M. VEERAIYAN**, our President **Dr. SAVEETHA RAJESH**, our Director **Dr. S. RAJESH** and other management members for providing the infrastructure needed.

We express our wholehearted gratitude to our principal, **Dr. N. DURAIPANDIAN**, for his wholehearted encouragement in completing this project.

We convey our thanks to **Dr. G. NAGAPPAN**, Professor and Head of the Department of Computer Science and Engineering, Saveetha Engineering College, for his kind support and for providing necessary facilities to carry out the project work.

We would like to express our sincere thanks and deep sense of gratitude to our Supervisor **Dr. P. V. GOPIRAJAN**, Assistant Professor, Department of Computer Science and Engineering, Saveetha Engineering College, Our Project coordinator, **Dr. M. VIJAY ANAND** for his valuable guidance, suggestions and constant encouragement that paved the way for the successful completion of the project work and for providing us necessary support and details at the right time and during the progressive reviews.

We owe our thanks to all the members of our college, faculty, staff and technicians for their kind and valuable cooperation during the course of the project. We are pleased to acknowledge our sincere thanks to our beloved parents, friends and well-wishers who encouraged us to complete this project successfully.

ABSTRACT

Traditional Gaming involve playing games using touch, keyboard or mouse events without incurring any sort of physical movements. What if the Real-time Body Gestures are used for controlling the game's character to make gaming more interesting? Few popular gaming gestures can be controlled by the movements made by the player standing in front of the camera without using any external sensors or any type of hard wares but only with the help of body gestures. This proposed project is planned to be implemented in PC/ Desktop. Using Mediapipe Pose Detection, a high-fidelity body pose tracking Machine Learning solution for developers to detect the pose of a subject's body in real time from a continuous video or static image, Body Pose Recognition could be made possible along with matplotlib library which is used to plot the 33 3D landmarks on the screen. With the help of PyAutoGUI, an API which lets the Python scripts control the mouse and keyboard to automate interactions with other applications, the mouse and keyboard events are controlled using Gestures. This project is mainly aimed to incur physical gestures while playing a game which makes the gaming experience much more amusing and interesting while keeping the player physically active as well.

Keywords: Body Gestures, Pose Detection, Mediapipe, OpenCV, matplotlib, PyAutoGUI.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iv
	LIST OF CONTENTS	v
	LIST OF FIGURES	viii
	LIST OF SYMBOLS	x
	LIST OF ABBREVIATIONS	xi
1	INTRODUCTION	1
	1.1 OVERVIEW OF THE PROJECT	1
	1.2 SCOPE AND OBJECTIVE	1
2	LITERATURE SURVEY	2
	2.1 INTRODUCTION	2
	2.2 LITERATURE SURVEY	2
	2.3 EXISTING SYSTEM	5
3	PROPOSED SYSTEM	6
	3.1 ADVANTAGES	6
	3.2 DISADVANTAGES	6
	3.3 REQUIREMENT ANALYSIS	7
	3.3.1 Hardware Requirement	7
	3.3.2 Software Requirement	7

4	SYSTEM DESIGN	8
	4.1 DESIGN	8
	4.2 ARCHITECTURE	8
	4.2.1 Architecture Description	8
	4.3 FLOW DIAGRAM	9
	4.4 UML DIAGRAMS	10
	4.4.1 Use case Diagram	10
	4.4.2 Class Diagram	10
	4.4.3 Activity Diagram	11
	4.5 FRAMEWORK DESCRIPTION	11
	4.5.1 Mediapipe Framework	11
	4.5.2 Applications of Mediapipe	12
	4.5.3 Working of Mediapipe	13
5	IMPLEMENTATION	14
	5.1 MODULES	14
	5.2 MODULE DESCRIPTION	14
	5.2.1 Testing Pose Detection	14
	5.2.2 Control of Start Mechanism	15
	5.2.3 Control of Horizontal Movements	15
	5.2.4 Control of Vertical Movements	17
	5.2.5 Control of Keyboard and Mouse Events Using PyAutoGUI	18
	5.2 TECHNOLOGY DESCRIPTION	19
	5.2.1 Python Library	19
	5.2.1.1 cv2	19





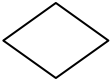



	5.2.1.2 Mediapipe's Pose Detection	20
	5.2.1.3 matplotlib lib and pyplot module	20
	5.2.1.4 math module	21
	5.2.1.5 time module	22
	5.2.1.6 PyAutoGUI	22
	 5.3 FEASIBILITY STUDY	 23
	5.3.1 Economical Feasibility	23
	5.3.2 Technical Feasibility	23
	5.3.3 Social Feasibility	24
	 5.5 SOFTWARE DESCRIPTION	 24
	5.5.1 Python	24
	5.5.2 Pycharm IDE	24
6	 CONCLUSION & FUTURE ENHANCEMENT	 26
	6.1 CONCLUSION	26
	6.2 FUTURE ENHANCEMENT	26
	 APPENDICES	 27
	 REFERENCES	 40

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
4.1	ARCHITECTURE DIAGRAM	7
4.2	FLOW DIAGRAM	9
4.3	USE CASE DIAGRAM	10
4.4	CLASS DIAGRAM	10
4.5	ACTIVITY DIAGRAM	11
4.6	MEDIAPIPE WORKING	23
5.1	POSE DETECTION TESTING SAMPLE	23
5.2	START MECHANISM	24
5.3	CENTRE POSITION	24
5.4	LEFT POSITION	25
5.5	RIGHT POSITION	25

5.6	STANDING POSITION	26
5.7	UP MOVEMENT	26
5.8	DOWN MOVEMENT	27

LIST OF SYMBOLS

S.NO.	SYMBOL NAME	SYMBOL
1.	Usecase	
2.	Actor	
3.	Process	
4.	Start	
5.	Decision	
6.	Unidirectional	
7.	Entity set	
8.	Stop	

LIST OF ABBREVIATIONS

S.NO.	ABBREVIATIONS	EXPANSION
1.	ML	Machine Learning
2.	HD	High Definition
3.	NI	Natural Interaction
4.	NUI	Natural User Interface
5.	RGB	Red Green Blue
6.	HSV	Hue Saturation Value
7.	PC	Personal Computer
8.	UML	Unified Modeling Language
9.	OpenCV	Open Source Computer Vision
10.	ROI	Region Of Interest
11.	GUI	Graphical User Interface
12.	IDE	Integrated Development Environment
13.	CPU	Central Processing Unit
14.	GB	Giga Byte
15.	RAM	Random Access Memory
16.	CSS	Cascading Style Sheets

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF THE PROJECT

This project promotes real time gaming using physical movements instead of traditional method of playing games, without using touch, keyboard, mouse or any other external sensors but only with the help of camera which makes the gaming experience much more interesting and helps player to be physically active as well.

1.2 SCOPE AND OBJECTIVE

SCOPE

The scope of the project is to facilitate body gesture gaming using Pose Detection. This project is limited to be used only in PC/Desktop as of now later it is planned to be implemented for android devices.

OBJECTIVE

- To promote real time gaming using physical movements instead of traditional method of playing games, without using touch, keyboard, mouse or any other input devices.
- To make gaming experience much more amusing and interesting by making the player to get involved in the game much effectively.
- This way of playing games need slight physical movements which would help the player to be mentally and physically active as well.

CHAPTER 2

LITERATURE SURVEY

2.1. INTRODUCTION

A literature survey or a literature review in a project report is that section which shows various analysis and research made in the field of your interest and the results already published, taking into account the various parameters of the project and the extent of project. Once the programmers start building the tool programmers need a lot of external support. It is the most important part of your report as it gives you a direction in the area of your research. Literature survey is the most important sector in the software development process. Before developing the tools and the associated designing it is necessary to determine the survey the time factor, resource requirement, manpower, economy, and company strength. It can be used to identify what are the different datasets that are available to train the dataset.

2.2. LITERATURE SURVEY

1. Graziano Fronteddu, Simone Porcu, Alessandro Floris, Luigi Atzori, 2022. “A dynamic hand gesture recognition dataset for human-computer interfaces”, DIEEE, University of Cagliari, 09123 Cagliari, Italy and CNIT, University of Cagliari, 09123 Cagliari, Italy.

Graziano Fronteddu, Simone Porcu, Alessandro Floris, Luigi Atzori proposed a dataset of 27 dynamic hand gesture types acquired at full HD resolution from 21 different subjects, which were carefully instructed before performing the gestures and monitored when performing the gesture, the subjects had to repeat the movement in case the performed hand gesture was not correct.

2. Tuan LinhDang , Sy DatTran , Thuy Hang Nguyen , Suntae Kim , NicolasMonet, 2022. “An improved hand gesture recognition system using keypoints and hand bounding boxes”, Hanoi University of Science and Technology and Naver Corporation.

Tuan LinhDang , Sy DatTran , Thuy Hang Nguyen , Suntae Kim , NicolasMonet developed a static hand gesture recognition system, consisting of three modules: Feature extraction Module, Processing Module, and Classification Module. The feature extraction module uses human pose estimation with a top-down method to extract not only the keypoints but also body and hand bounding boxes and uses two-pipeline architecture.

3. Katherina A. Jurewicz , David M. Neyens, 2022. “Redefining the human factors approach to 3D gestural HCI by exploring the usability-accuracy tradeoff in gestural computer systems”, Oklahoma State University,USA and Clemson University, USA.

Katherina A. Jurewicz , David M. Neyens performed a user elicitation study, and gestures were classified according to a novel feature extraction gesture taxonomy and a traditional taxonomy of classifying gestures as a unit. The feature-extraction approach revealed several advantages because it fosters a bottom-up approach to identifying gesture features.

**4. Dr. Parameshachari B D¹ , Rubeena Muheeb¹ ,Nagashree R N¹
 ,Deekshith B N¹, Keerthikumar M¹ , Rashmi P² , Rachana C R, 2020.
 “Design of an Gesture Recognition Based Car Gaming”, International
 Journal of Advanced Networking & Applications (IJANA).**

Dr. Parameshachari B D , Rubeena Muheeb and other DoS in ComputerScience from Mysuru proposed a design of Gesture recognition based Car gaming in which gestures where used along with Kinect Sensors to detect the gesture. The Kinect here can be defined as a Natural Interaction (NI) device, which operates in the context of Natural User interface (NUI).

5.Dimple Talasila, Gopi Manoj Vuyyuru, 2020. “Hand Gesture Gaming using Ultrasonic Sensors & Arduino”, ISSN paper published by IJERT licensed under Creative Commons Attribution 4.0 International License.

Gopi Manoj Vuyyuru from Tata Consultancy Services Bangalore provided a prototype on Hand Gesture Gaming using Ultrasonic Sensors & Arduino. Two ultrasonic sensors fixed on the top of laptop separated by a distance, circuit connections are made between ultrasonic sensors and arduino uno board. And then the ultrasonic sensor is used to measure the distance between sensor and our hand placed in front of it. Based on the distance specific operation is performed.

6. Niyati Gosalia, Priya Jain, Ishita Shah, Abhijit R.Joshi Dr., Neha Katre, Sameer Sahasrabudhe, 2015. “3D Gesture-recognition Based Animation Game”, Procedia Computer Science.

Niyati Gosalia, Priya Jain, Ishita Shah, Abhijit R.Joshi Dr., Neha Katre, Sameer Sahasrabudhe conducted a research on the existing version of a 3-D gesture-recognition based animation game, MathMazing with the purpose of overcoming the drawbacks of this system. In addition to this, we have summarized our approach to the proposed system.

2.3. EXISTING SYSTEM

- Gestures played a great role in playing games with much more interaction between the player and the game. Previously hand gestures were most widely used for controlling the game’s character or also using external sensors and hardwares the character of the game is controlled.
- Image processing was used to control the keyboard or mouse events which requires using a physical object to control. In this way, first the image is captured and then the RGB values are converted to HSV values to which the Gaussian blur is applied to blur out the unwanted spaces or objects and the contours are applied to highlight the most needed part of the image through which the keypress events are controlled.
- This method is time consuming and a physical object with color variations is needed to extract the specific part needed to control the keypress events.

CHAPTER 3

PROPOSED SYSTEM

Interactive method of playing games can be enabled by using Pose Detection that detect body gestures which doesn't need any physical object whereas for Image Processing it is necessary, for this the Mediapipe's Pose Detection Machine learning solution is used. Pyautogui library, used to control keyboard and mouse events, automatically triggers the required keypress events depending upon the movement of the person that we've captured using Mediapipe pose Detection Model. Here the four basic movements like move left, move right, jump and crouch are controlled by triggering the keys Up arrow, Down arrow, left Arrow and right arrow or W,A,S,D in the keyboard

3.1. ADVANTAGES

- The proposed system is much interesting as playing games using body gestures would take the gaming experience to a next level with greater entertainment.
- No external hardwares or sensors are needed for pose detection just a camera infused pc/laptop is far more enough.
- When the games are played using this proposed system, games would become more interesting and hence the download rate of compatible games would increase to a greater extent.

3.2. DISADVANTAGES

- The proposed system is compatible for only PC/Laptop as of now
- Multi-Player games are not supported since only single player is detected not more than a player.

3.3 REQUIREMENT ANALYSIS

3.3.1. HARDWARE REQUIREMENT

- RAM: 6 GB
- Camera infused laptop or PC
- Storage: 500 GB.
- CPU: 2 GHz or faster.
- Architecture: 32-bit or 64-bit.
- Nvidia Graphics Card (Optional for better performance)

3.3.2. SOFTWARE REQUIREMENT

- Operating System - Windows 8 or higher (or equivalent).
- Internet Connection – for online games

CHAPTER 4

SYSTEM DESIGN

4.1. DESIGN

Design is the first step in development phase for any techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization. Once the software requirement has been analysed and specified the software design involves three technical activities-Design, Coding, Implementation, Testing that are required to build and verify the software.

4.2 ARCHITECTURE

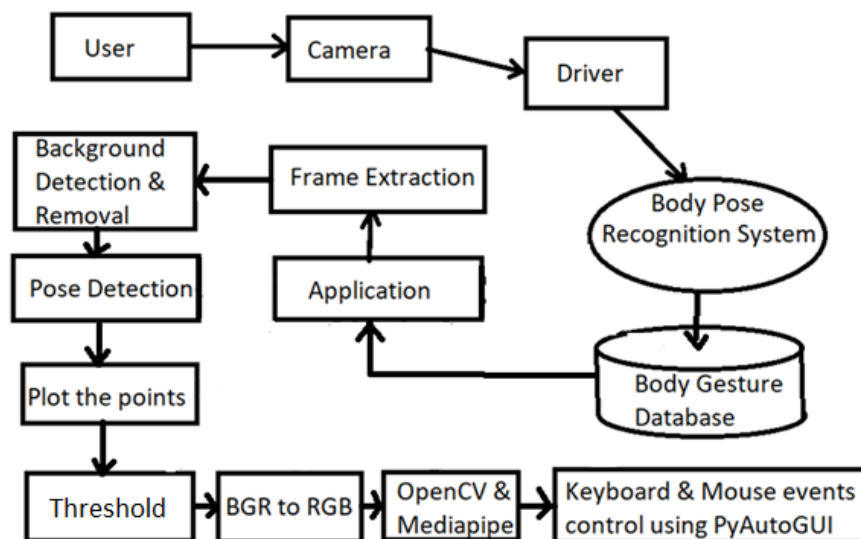


Fig. 4.1 Architecture Diagram

4.2.1 ARCHITECTURE DESCRIPTION

User stands in front of the camera and accesses the driver and the pose is recognized by the Pose Recognition System which in turn gets stored in Gesture Database then the application extracts frames from the video captured real time and the background is ignored as only the player is to be focussed mainly then the frames are converted from RGB to BGR format and given as input to opencv

then the pose is detected using mediapipe and the points are plotted using matplotlib and the threshold values are checked against for exact recognition of the pose. Finally the corresponding key press events of PyAutoGUI is triggered to control the keyboard while the pose is being used to control the game's Character

4.3 FLOW DIAGRAM

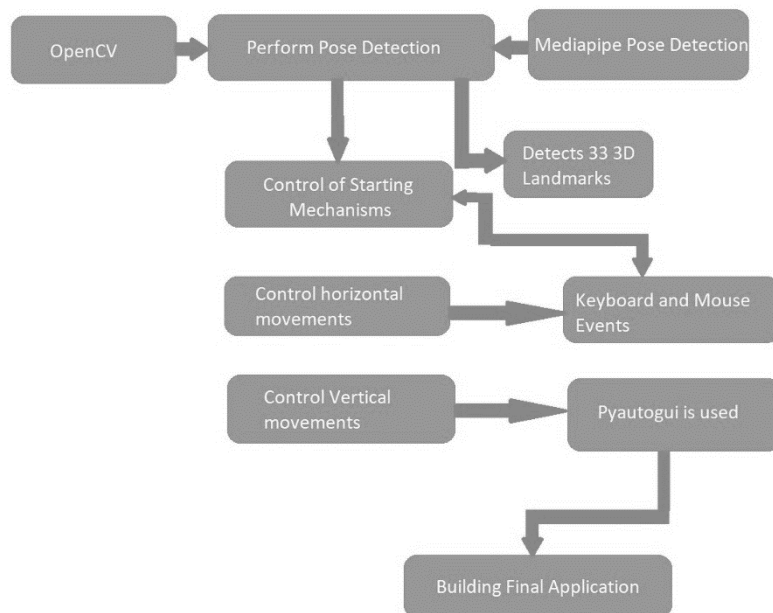


Fig. 4.2 Flow Diagram

4.4 UML DIAGRAMS

4.4.1 USE CASE DIAGRAM

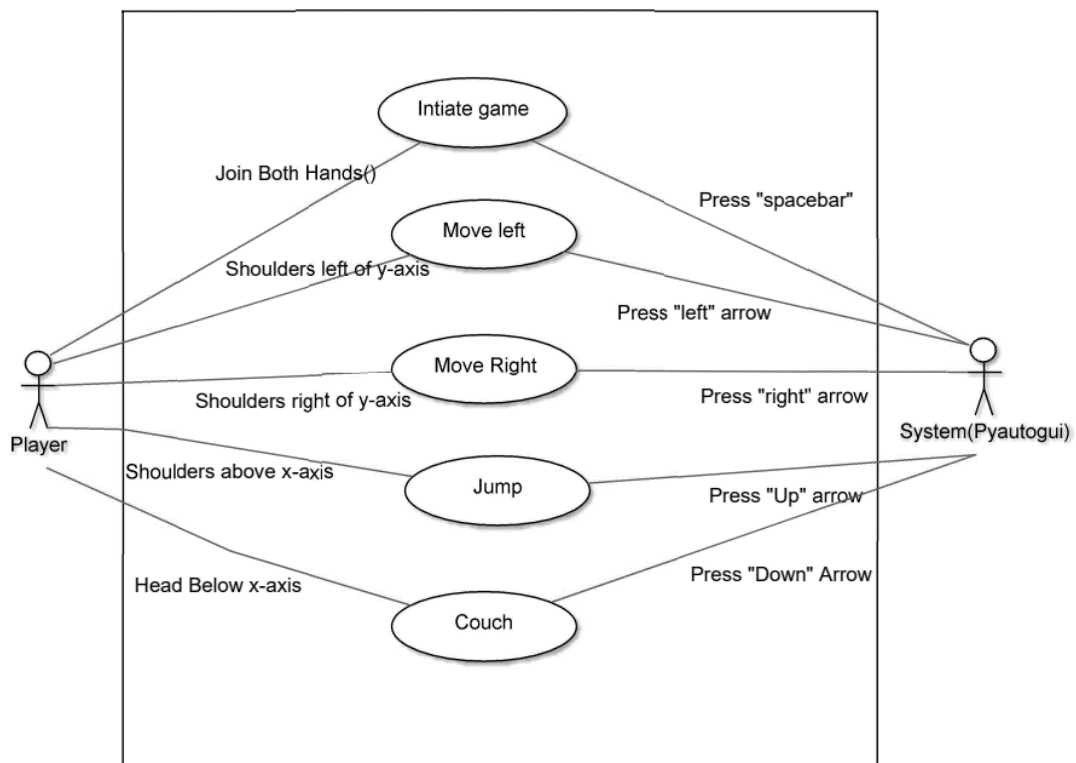


Fig 4.3 USE CASE DIAGRAM

4.4.2 CLASS DIAGRAM

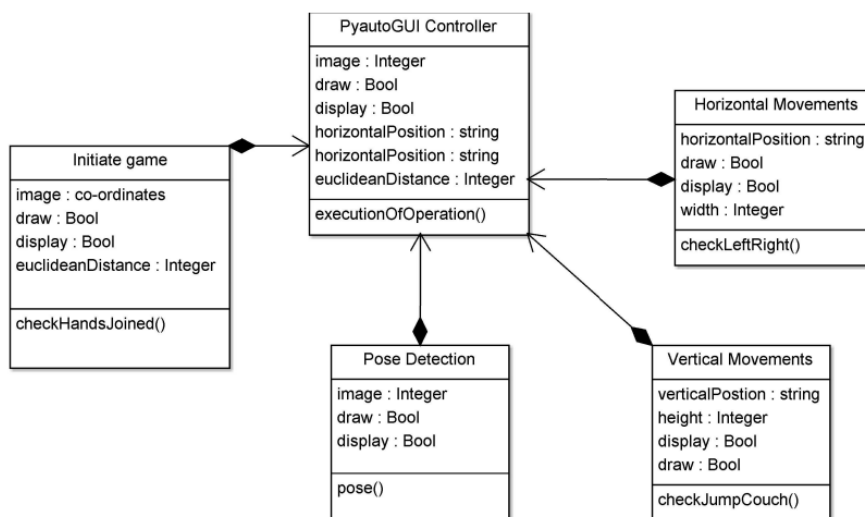


Fig 4.4 CLASS DIAGRAM

4.4.3 ACTIVITY DIAGRAM

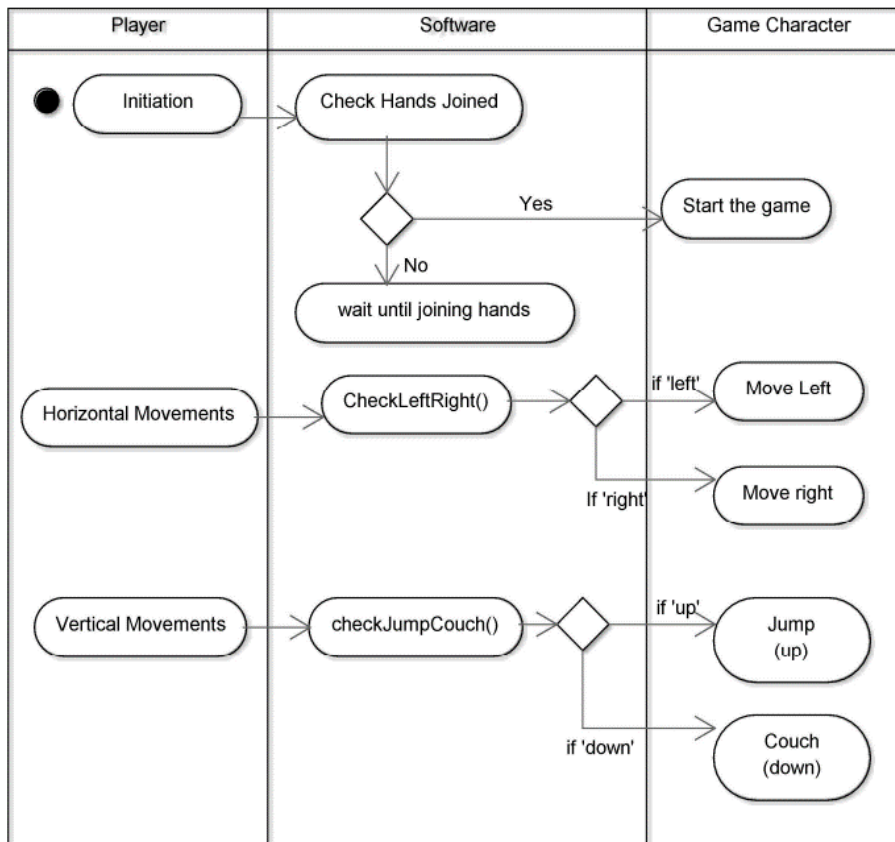


Fig 4.5 ACTIVITY DIAGRAM

4.5 FRAMEWORK DESCRIPTION

4.5.1 MEDIAPIPE FRAMEWORK

Mediapipe is a Framework for building machine learning pipelines for processing time-series data like video, audio, etc. Mediapipe supports multimodal graphs. To speed up the processing, different calculators run in separate threads. For performance optimization, many built-in calculators come with options for GPU acceleration. Working with time series data must be in proper synchronization; otherwise, the system will break. The graph ensures this so that flow is handled correctly according to the timestamps of packets. The Framework handles synchronization, context sharing, and inter-operations with CPU calculators. It is an open-source, cross-platform Machine Learning framework used for building

complex and multimodal applied machine learning pipelines. It can be used to make cutting-edge Machine Learning Models like face detection, multi-hand tracking, object detection, and tracking, and many more. Mediapipe basically acts as a mediator for handling the implementation of models for systems running on any platform which helps the developer focus more on experimenting with models, than on the system.

4.5.2 APPLICATIONS OF MEDIAPIPE

- i) Human Pose Detection and Tracking
- ii) Object Detection and Tracking
- iii) 3D Object Detection
- iv) Face Mesh
- v) Instant Motion Tracking

4.5.3 WORKING OF MEDIAPIPE

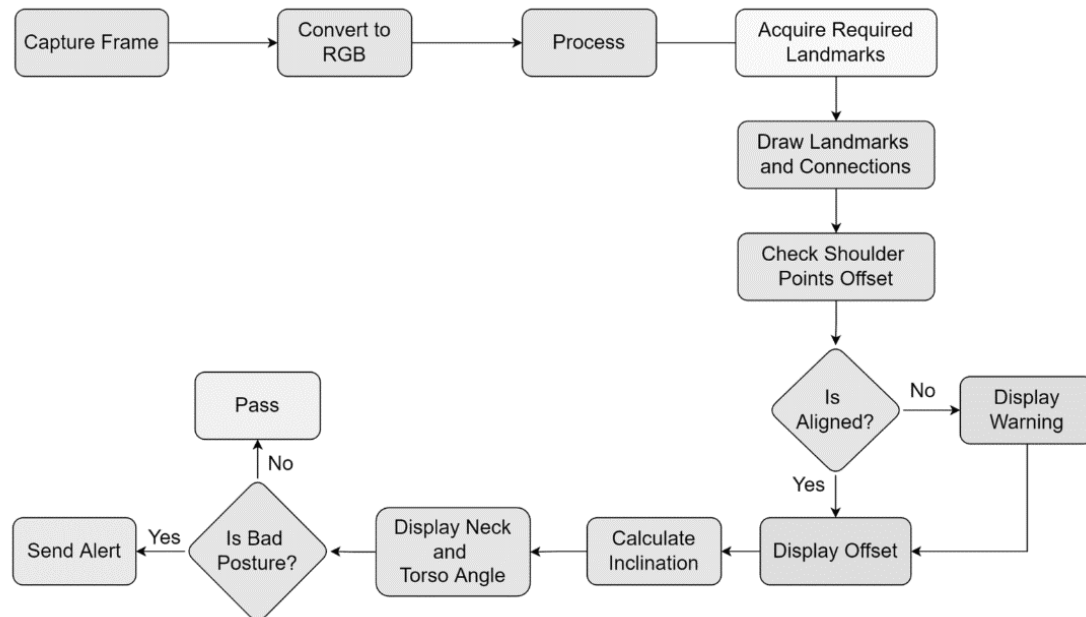


Fig. 4.6 MEDIAPIPE WORKING

Object detection using Mediapipe. The transparent boxes represent computation nodes (calculators) in a Mediapipe graph, solid boxes represent external input/output to the graph, and the lines entering the top and exiting the bottom of the nodes represent the input and output streams respectively. The ports on the left of some nodes denote the input side packets.

CHAPTER 5

IMPLEMENTATION

5.1 MODULES

- Testing Pose Detection
- Initiation of Game using 'Joined Hands' Gesture
- Horizontal Movements Detection
- Vertical Movements Detection
- Control of Keyboard events using PyautoGUI

5.2 MODULE DESCRIPTION

5.2.1 TESTING POSE DETECTION

A sample image is taken as an input and the pose detection is performed on the image using matplotlib.



Fig 5.1 Pose Detection Testing Sample

5.2.2 Control of Start Mechanism

Joined Hands Gesture is used to Initiate the game. Here if the distance between the two wrists are less than 130 then it is confirmed that the hands are in joined condition.

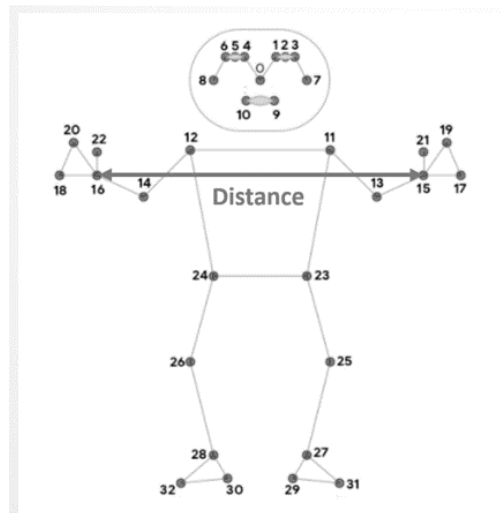


Fig. 5.2 Start Mechanism

5.2.3 CONTROL OF HORIZONTAL MOVEMENTS

i) Centre Position

When the x-coordinate of the left shoulder of the person is less than half of the width and the x coordinate of the right shoulder is greater than half of the width then the person is confirmed to be in the centre position

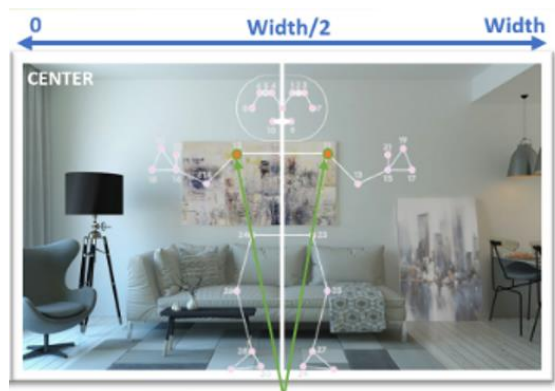


Fig. 5.3 Centre Position

ii) Left Movement

When the x-coordinate of the left and right shoulders are less than half of the width then the person is considered to be on the left position

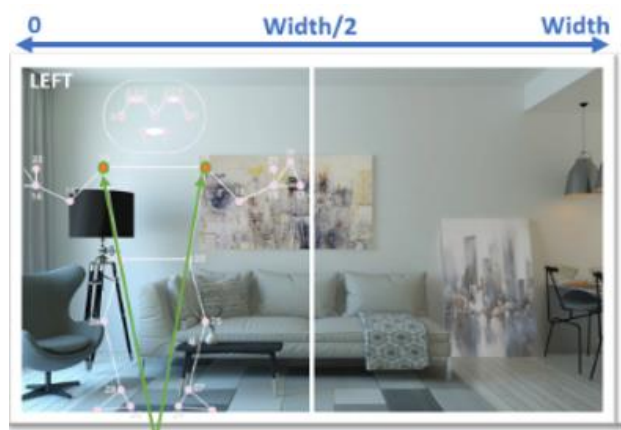


Fig. 5.4 Left Position

iii) Right Movement

When the x-coordinate of the left and right shoulders are greater than half of the width then the person is confirmed to be on the right position.

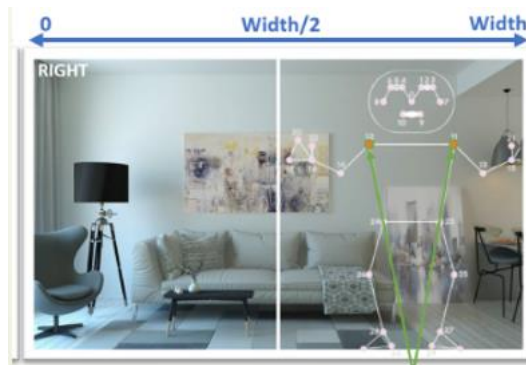


Fig. 5.5 Right Position

5.2.4 CONTROL OF VERTICAL MOVEMENTS

i) Standing Position

When the y-coordinate of the midpoint of shoulders lie between the threshold range the person is confirmed to be in standing position

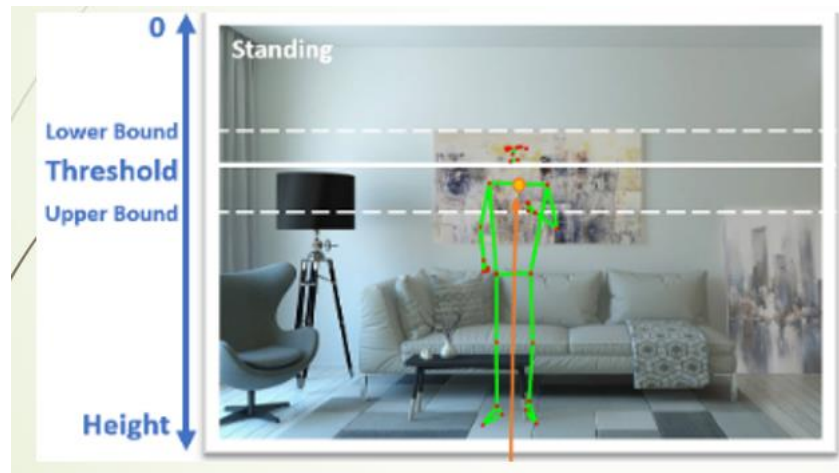


Fig. 5.6 Standing Position

ii) Up Movement(JUMP)

When the y-coordinate of the midpoint of the shoulders is greater than the upper bound of threshold range then the person is Jumping.

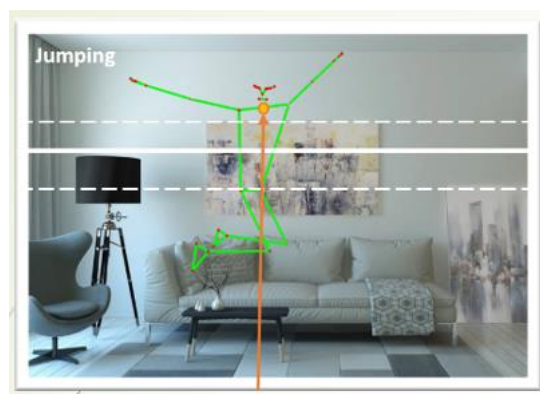


Fig. 5.7 Up Movement

iii) Down Movement(CROUCH)

When the y-coordinate of the midpoint of the shoulders is less than the lower bound of threshold range then the person is Crouching.



Fig. 5.8 Down Movement

5.2.5 CONTROL OF KEYBOARD AND MOUSE EVENTS USING

PyAutoGUI

► For Keyboard events:

- `pyautogui.press()` function is used with two parameters 'name of key' and 'number of presses' where 'number of presses' is an optional parameter.

► For Mouse events:

- `pyautogui.click()` function is used with three parameters 'x coordinate', 'y coordinate' and 'button (right/left)' where x coordinate and y coordinate are optional parameters.

5.2 TECHNOLOGY DESCRIPTION

5.2.1. PYTHON LIBRARY

A Python library is a reusable chunk of code that you may want to include in your programs/ projects. The Python Standard Library is a collection of exact syntax, token, and semantics of Python. It comes bundled with core Python distribution. We mentioned this when we began with an introduction. For visualization of the dataset and prediction we use various python libraries.

Python libraries that are used in the project are:

- Cv2
- Mediapipe's Pose Detection
- matplotlib
- PyAutoGUI

5.2.1.1. cv2:

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. It can process images and videos to identify objects, faces, or even the handwriting of a human. Opencv is an open source library which is very useful for computer vision applications such as video analysis, CCTV footage analysis and image analysis. OpenCV-Python is a library of Python bindings designed to solve computer vision problems. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database,

remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

5.2.1.2. MEDIAPIPE'S POSE DETECTION:

Human pose estimation from video plays a critical role in various applications such as quantifying physical exercises, sign language recognition, and full-body gesture control. For example, it can form the basis for yoga, dance, and fitness applications. It can also enable the overlay of digital content and information on top of the physical world in augmented reality. Mediapipe Pose is a high-fidelity body pose tracking solution that renders 33 3D landmarks and a background segmentation mask on the whole body from RGB frames. The Mediapipe solution utilizes a two-step detector-tracker ML pipeline, proven to be effective in our Mediapipe Hands and Mediapipe Face Mesh solutions. Using a detector, the pipeline first locates the person/pose region-of-interest (ROI) within the frame. The tracker subsequently predicts the pose landmarks and segmentation mask within the ROI using the ROI-cropped frame as input. Note that for video use cases the detector is invoked only as needed, i.e., for the very first frame and when the tracker could no longer identify body pose presence in the previous frame. For other frames the pipeline simply derives the ROI from the previous frame's pose landmarks. The pipeline is implemented as a Mediapipe graph that uses a pose landmark subgraph from the pose landmark module and renders using a dedicated pose renderer subgraph. The pose landmark subgraph internally uses a pose detection subgraph from the pose detection module.

5.2.1.3. Matplot Library & Pyplot Module:

Matplotlib is a plotting library for creating static, animated, and interactive visualizations in Python. Matplotlib can be used in Python scripts, the Python

and IPython shell, web application servers, and various graphical user interface toolkits like Tkinter, awxPython, etc.

Pyplot is a Matplotlib module which provides a MATLAB-like interface. Matplotlib is designed to be as usable as MATLAB, with the ability to use Python and the advantage of being free and open-source. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. The various plots we can utilize using Pyplot are Line Plot, Histogram, Scatter, 3D Plot, Image, Contour, and Polar.

5.2.1.4. math module:

The Python math module provides functions that are useful in number theory as well as in representation theory, a related field. These functions allow you to calculate a range of important values, including the factorials of a number, the greatest common divisor of two numbers the sum of iterables etc. Both simple mathematical calculations like addition (+), and subtraction (-), and advanced mathematical calculations like trigonometric operations, and logarithmic operations can be performed by the functions in the math module. Python has a set of built-in math functions, including an extensive math module, that allows you to perform mathematical tasks on numbers. This Python module does not accept complex data types. The more complicated equivalent is the cmath module. The value of numerous constants, including pi and tau, is provided in the math module so that we do not have to remember them. Using these constants eliminates the need to precisely and repeatedly write down the value of each constant. The math module is used to access mathematical functions in the Python. All methods of this functions are used for integer or real type objects, not for complex numbers. The math built-in module includes a number of constants and methods that support mathematical operations from basic to advanced.

5.2.1.5. time module:

As the name suggests Python time module allows to work with time in Python. It allows functionality like getting the current time, pausing the Program from executing, etc. The Python time module provides many ways of representing time in code, such as objects, numbers, and strings. It also provides functionality other than representing time, like waiting during code execution and measuring the efficiency of your code. The time module in python is an inbuilt module that needs to be imported when dealing with dates and times in python. Python provides library to read, represent and reset the time information in many ways by using “time” module. Date, time and date time are an object in Python, so whenever we do any operation on them, we actually manipulate objects not strings or timestamps. In this section we’re going to discuss the “time” module which allows us to handle various operations on time. The time module follows the “EPOCH” convention which refers to the point where the time starts.

5.2.1.6. PyAutoGUI:

PyAutoGUI is a cross-platform GUI automation Python module for human beings. Used to programmatically control the mouse & keyboard.

PyAutoGUI has the following features:

- Moving the mouse and clicking in the windows of other applications.
- Sending keystrokes to applications (for example, to fill out forms).
- Take screenshots, and given an image (for example, of a button or checkbox), and find it on the screen.
- Locate an application’s window, and move, resize, maximize, minimize, or close it (Windows-only, currently).

5.3 FEASIBILITY STUDY

- The feasibility of the project is analysed in this phase and the business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.
- The feasibility study investigates the problem and the information needs of the stakeholders. It seeks to determine the resources required to provide an information systems solution, the cost and benefits of such a solution, and the feasibility of such a solution.
- The goal of the feasibility study is to consider alternative information systems solutions, evaluate their feasibility, and propose the alternative most suitable to the organization. The feasibility of a proposed solution is evaluated in terms of its components.

5.3.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. Thus, the developed system requires an IDE and person for data collection. The software is mostly open source and dataset can be collected from the software.

5.3.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. Thus, the developed system requires an IDE such as Pycharm or google

Colab to evaluate datasets and collection of dataset for different environment conditions requires a person to follow the instructions.

5.3.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

5.5 SOFTWARE DESCRIPTION

5.5.1. PYTHON (Programming language)

Python has huge set libraries which can be easily used for machine learning. Python is one of the languages which can be used to write codes in the Map-Reduce model while working in the Hadoop Ecosystem. Also, Spark, which is one of the new technologies for scalable big-data analysis, has a machine learning library in python. So, simplicity and wider applicability goes hand-in-hand to make it the so-called machine learning language. Python is an interpreter, high-level, general-purpose programming language. Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, it also has a comprehensive standard library. Python interpreters are available for many operating systems.

5.5.2. Pycharm IDE

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development. PyCharm provides smart code completion, code inspections, on-

the-fly error highlighting and quick-fixes, along with automated code refactorings and rich navigation capabilities.

PyCharm is a Python IDE with complete set of tools for Python development. In addition, the IDE provides capabilities for professional Web development using the Django framework. Code faster and with more easily in a smart and configurable editor with code completion, snippets, code folding and split windows support.

PyCharm Features

- **Intelligent Coding Assistance** - PyCharm provides smart code completion, code inspections, on-the-fly error highlighting and quick-fixes, along with automated code refactorings and rich navigation capabilities.
- **Intelligent Code Editor** - PyCharm's smart code editor provides first-class support for Python, JavaScript, CoffeeScript, TypeScript, CSS, popular template languages and more. Take advantage of language-aware code completion, error detection, and on-the-fly code fixes!
- **Smart Code Navigation** - Use smart search to jump to any class, file or symbol, or even any IDE action or tool window. It only takes one click to switch to the declaration, super method, test, usages, implementation, and more.

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 CONCLUSION

There are many ways of playing games using various technologies available. Amongst the latest technologies the most fascinating method of playing games is using body gestures. Pose Detection which is used in the system would enable real-time gaming using physical movements instead of traditional method of playing games, without using touch, keyboard, mouse or any other external hardwares or sensors. This helps player to be physically active and makes gaming more interesting. A negligible amount of latency could be faced which can be corrected using correct system requirements.

6.2 FUTURE ENHANCEMENT

The system is very flexible in terms of expansion. Modules can be added to make a better system. Features such as multi-player support can also be included.

- As of now the system is compatible for use only in PC/Desktop and in view of future enhancements the system could be extended to support for android and ios devices as well.
- Also, the system supports only single player games as only one player's pose is detected and the future scope can be extended for supporting multi-player games using multiple players' pose detection.

APPENDICES

SAMPLE CODE

```
import cv2
import pyautogui
from time import time
from math import hypot
import mediapipe as mp
import matplotlib.pyplot as plt

mp_pose = mp.solutions.pose

pose_image = mp_pose.Pose(static_image_mode=True,
min_detection_confidence=0.5, model_complexity=1)

pose_video = mp_pose.Pose(static_image_mode=False, model_complexity=1,
min_detection_confidence=0.7,
min_tracking_confidence=0.7)

mp_drawing = mp.solutions.drawing_utils

def detectPose(image, pose, draw=False, display=False):

    output_image = image.copy()

    imageRGB = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    results = pose.process(imageRGB)

    if results.pose_landmarks and draw:
        mp_drawing.draw_landmarks(image=output_image,
landmark_list=results.pose_landmarks,
connections=mp_pose.POSE_CONNECTIONS,

landmark_drawing_spec=mp_drawing.DrawingSpec(color=(255, 255, 255),
thickness=3, circle_radius=3),

connection_drawing_spec=mp_drawing.DrawingSpec(color=(49, 125, 237),
thickness=2,
circle_radius=2))
```

if display:

```
plt.figure(figsize=[22, 22])
plt.subplot(121);
plt.imshow(image[:, :, ::-1]);
plt.title("Original Image");
plt.axis('off');
plt.subplot(122);
plt.imshow(output_image[:, :, ::-1]);
plt.title("Output Image");
plt.axis('off');
```

else:

```
return output_image, results
```

def checkHandsJoined(image, results, draw=False, display=False):

```
height, width, _ = image.shape
```

```
output_image = image.copy()
```

```
left_wrist_landmark =
(results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_WRIST].x *
width,
```

```
results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_WRIST].y *
height)
```

```
right_wrist_landmark =
(results.pose_landmarks.landmark[mp_pose.PoseLandmark.RIGHT_WRIST].x *
width,
```

```
results.pose_landmarks.landmark[mp_pose.PoseLandmark.RIGHT_WRIST].y
* height)
```

```
euclidean_distance = int(hypot(left_wrist_landmark[0] -
right_wrist_landmark[0],
                             left_wrist_landmark[1] - right_wrist_landmark[1]))
```

```
if euclidean_distance < 130:
```

```

    hand_status = 'Hands Joined'

    color = (0, 255, 0)

else:

    hand_status = 'Hands Not Joined'

    color = (0, 0, 255)

if draw:
    cv2.putText(output_image, hand_status, (10, 30),
cv2.FONT_HERSHEY_PLAIN, 2, color, 3)

    cv2.putText(output_image, f'Distance: {euclidean_distance}', (10, 70),
        cv2.FONT_HERSHEY_PLAIN, 2, color, 3)

if display:

    plt.figure(figsize=[10, 10])
    plt.imshow(output_image[:, :, :-1]);
    plt.title("Output Image");
    plt.axis('off');

else:

    return output_image, hand_status

def checkLeftRight(image, results, draw=False, display=False):
    horizontal_position = None

    height, width, _ = image.shape

    output_image = image.copy()

    left_x =
int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.RIGHT_SHOULDER].x * width)

    right_x =
int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_SHOULDER].x * width)

```



```
ER].x * width)
```

```
if (right_x <= width // 2 and left_x <= width // 2):
```

```
    horizontal_position = 'Left'
```

```
elif (right_x >= width // 2 and left_x >= width // 2):
```

```
    horizontal_position = 'Right'
```

```
elif (right_x >= width // 2 and left_x <= width // 2):
```

```
    horizontal_position = 'Center'
```

```
if draw:
```

```
    cv2.putText(output_image, horizontal_position, (5, height - 10),  
cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 3)
```

```
    cv2.line(output_image, (width // 2, 0), (width // 2, height), (255, 255, 255),  
2)
```

```
if display:
```

```
    plt.figure(figsize=[10, 10])  
    plt.imshow(output_image[:, :, :-1]);  
    plt.title("Output Image");  
    plt.axis('off');
```

```
else:
```

```
    return output_image, horizontal_position
```

```
def checkJumpCrouch(image, results, MID_Y=250, draw=False,  
display=False):
```

```
    height, width, _ = image.shape
```

```
    output_image = image.copy()
```

```
    left_y =  
int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.RIGHT_SHOUL
```

```

DER].y * height)

    right_y =
int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_SHOULDER].y * height)

    actual_mid_y = abs(right_y + left_y) // 2

    lower_bound = MID_Y - 15
    upper_bound = MID_Y + 100

    if (actual_mid_y < lower_bound):

        posture = 'Jumping'

    elif (actual_mid_y > upper_bound):

        posture = 'Crouching'

    else:

        posture = 'Standing'

    if draw:
        cv2.putText(output_image, posture, (5, height - 50),
cv2.FONT_HERSHEY_PLAIN, 2, (255, 255, 255), 3)

        cv2.line(output_image, (0, MID_Y), (width, MID_Y), (255, 255, 255), 2)

    if display:

        plt.figure(figsize=[10, 10])
        plt.imshow(output_image[:, :, ::-1]);
        plt.title("Output Image");
        plt.axis('off');

    else:

        return output_image, posture

camera_video = cv2.VideoCapture(0)
camera_video.set(3, 1280)

```

```

camera_video.set(4, 960)

cv2.namedWindow('Real Time Gaming Using Gestures With Pose Detection',
cv2.WINDOW_NORMAL)

time1 = 0

game_started = False

x_pos_index = 1

y_pos_index = 1

MID_Y = None

counter = 0

num_of_frames = 10

while camera_video.isOpened():

    ok, frame = camera_video.read()

    if not ok:
        continue

    frame = cv2.flip(frame, 1)

    frame_height, frame_width, _ = frame.shape

    frame, results = detectPose(frame, pose_video, draw=game_started)

    if results.pose_landmarks:

        if game_started:
            frame, horizontal_position = checkLeftRight(frame, results, draw=True)

            if (horizontal_position == 'Left' and x_pos_index != 0) or (
                horizontal_position == 'Center' and x_pos_index == 2):

                pyautogui.press('left')

                x_pos_index -= 1

```

```

elif (horizontal_position == 'Right' and x_pos_index != 2) or (
    horizontal_position == 'Center' and x_pos_index == 0):

    pyautogui.press('right')

    x_pos_index += 1

# -----
-----

else:

    cv2.putText(frame, 'JOIN BOTH HANDS TO START THE GAME.',
(5, frame_height - 10), cv2.FONT_HERSHEY_PLAIN,
        2, (0, 255, 0), 3)

if checkHandsJoined(frame, results)[1] == 'Hands Joined':

    counter += 1

    if counter == num_of_frames:

        if not (game_started):

            game_started = True

            left_y =
int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.RIGHT_SHOUL
DER].y * frame_height)

            right_y =
int(results.pose_landmarks.landmark[mp_pose.PoseLandmark.LEFT_SHOULD
ER].y * frame_height)

            MID_Y = abs(right_y + left_y) // 2

            pyautogui.click(x=1300, y=800, button='left')

        else:

            pyautogui.press('space')

```

```

        counter = 0

    else:

        counter = 0

    if MID_Y:

        frame, posture = checkJumpCrouch(frame, results, MID_Y, draw=True)

        if posture == 'Jumping' and y_pos_index == 1:

            pyautogui.press('up')

            y_pos_index += 1

        elif posture == 'Crouching' and y_pos_index == 1:

            pyautogui.press('down')

            y_pos_index -= 1

        elif posture == 'Standing' and y_pos_index != 1:

            y_pos_index = 1

    else:

        counter = 0

    time2 = time()

    if (time2 - time1) > 0:
        frames_per_second = 1.0 / (time2 - time1)

        cv2.putText(frame, 'FPS: {}'.format(int(frames_per_second)), (10, 30),
cv2.FONT_HERSHEY_PLAIN, 2, (0, 255, 0),
                    3)

    time1 = time2

```

```
cv2.imshow('Real Time Gaming Using Gestures With Pose Detection',  
frame)
```

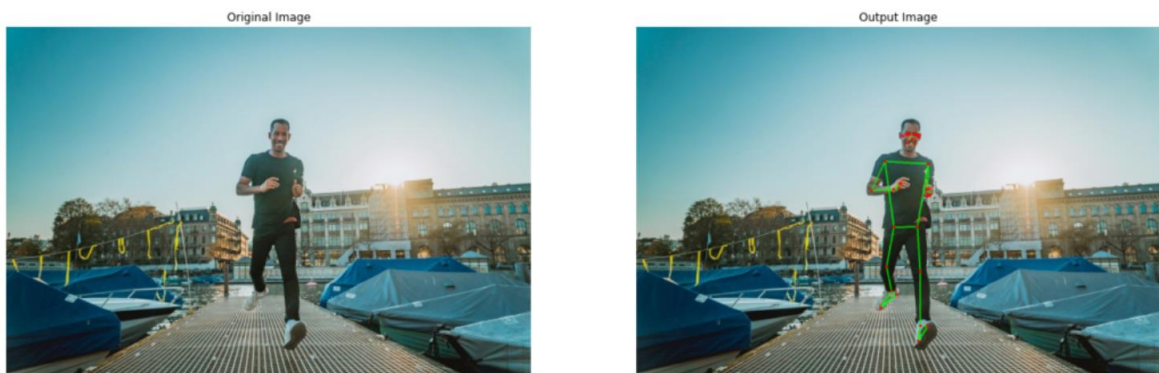
```
    k = cv2.waitKey(1) & 0xFF  
    if (k == 27):  
        break  
camera_video.release()  
cv2.destroyAllWindows()
```

OUTPUT SCREENSHOTS

POSE DETECTION USING IMAGES

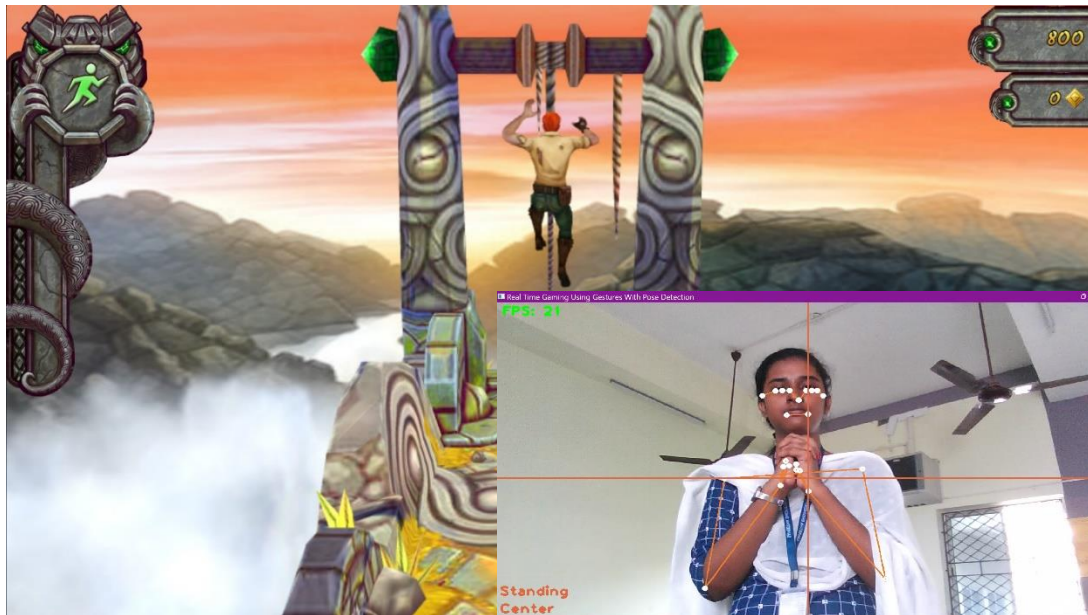


Pose Detection Using Image - Example 1

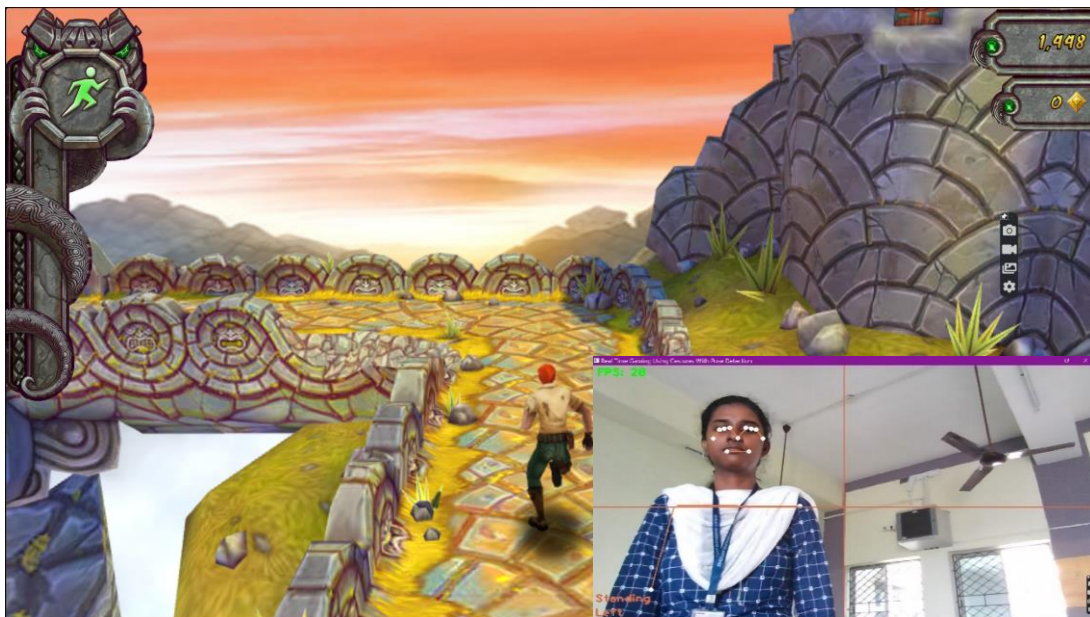


Pose Detection Using Image - Example 2

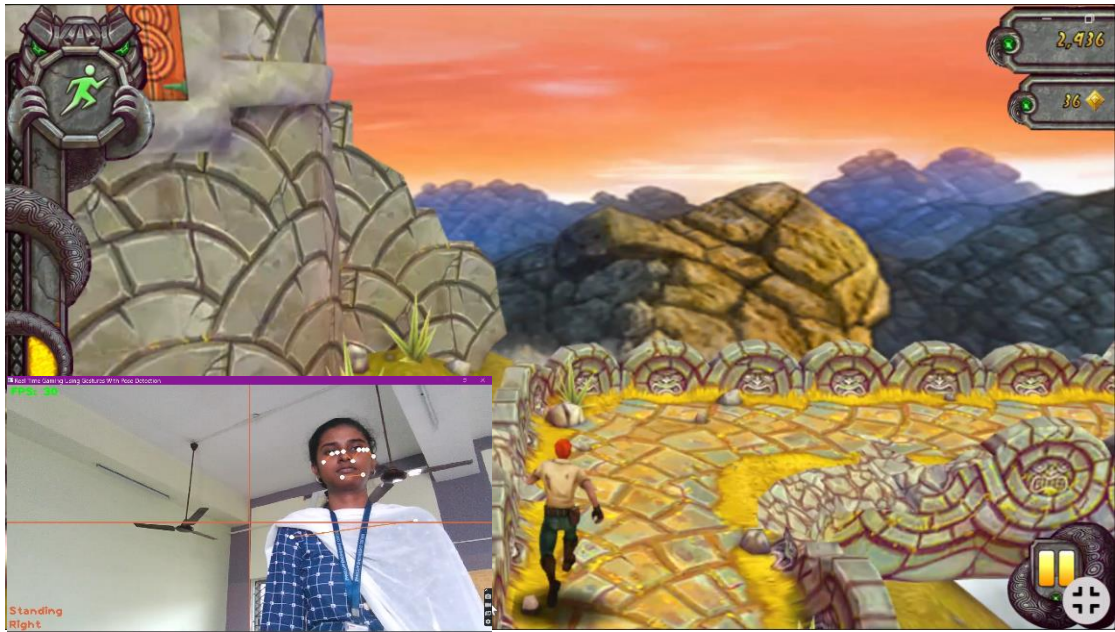
GAME IMPLEMENTATION SCREENSHOTS



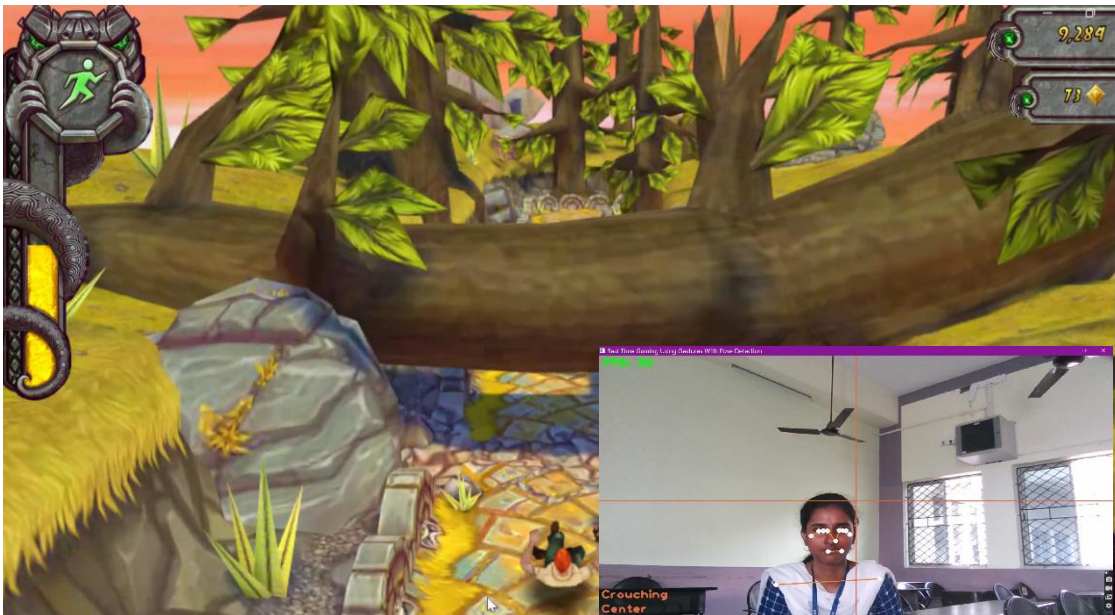
Starting Game Using Join Hands Gesture



Left Movement



Right Movement



Crouching Movement



Jumping Movement

REFERENCES

1. Dimple Talasila, Gopi Manoj Vuyyuru, 2020. “Hand Gesture Gaming using Ultrasonic Sensors & Arduino”, ISSN paper published by IJERT licensed under Creative Commons Attribution 4.0 International License.
2. Fang, Y., Wang, K., Cheng, J. and Lu, H., 2007, July. A real-time hand gesture recognition method. In 2007 IEEE International Conference on Multimedia and Expo (pp. 995-998). IEEE.
3. Graziano Fronteddu, Simone Porcu, Alessandro Floris, Luigi Atzori DIEE, 2022. “A dynamic hand gesture recognition dataset for human-computer interfaces”, University of Cagliari, 09123 Cagliari, Italy and CNIT, University of Cagliari, 09123 Cagliari, Italy.
4. Katherina A. Jurewicz , David M. Neyens , 2022. “Redefining the human factors approach to 3D gestural HCI by exploring the usability-accuracy tradeoff in gestural computer systems ”, Oklahoma State University, USA and Clemson University, USA.
5. Khan, R.Z. and Ibraheem, N.A., 2012. “Hand gesture recognition: a literature review”, International journal of artificial Intelligence & Applications, 3(4), p.161.
6. Niyati Gosalia, Priya Jain, Ishita Shah, Abhijit R.Joshi Dr., Neha Katre, Sameer Sahasrabudhe, 2015. “3D Gesture-recognition Based Animation Game”, Procedia Computer Science.
7. Rautaray, S.S. and Agrawal, A., 2011, December. Interaction with

virtual game through hand gesture recognition. In 2011 International Conference on Multimedia, Signal Processing and Communication Technologies (pp. 244-247). IEEE.

8. Rubeena Muheeb¹ ,Nagashree R N¹ ,Deekshith B N¹, Keerthikumar M¹, Rashmi P² , Rachana C R, Dr. Parameshachari B D¹ , 2020. “Design of an Gesture Recognition Based Car Gaming”, International Journal of Advanced Networking & Applications (IJANA).
9. Tuan LinhDang , Sy DatTran , Thuy Hang Nguyen , Suntae Kim , NicolasMonet, 2022. “An improved hand gesture recognition system using keypoints and hand bounding boxes”,Hanoi University of Science and Technology and Naver Corporation.
10. Zhang, X., Chen, X., Li, Y., Lantz, V., Wang, K. and Yang, J., 2011. A framework for hand gesture recognition based on accelerometer and EMG sensors. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 41(6), pp.1064-1076.