- 1. Project Summary

Abstract

The project introduces an AI-powered, gesture-controlled virtual combat system that allows players to use natural body movements—like punches, kicks, and jumps—to control in-game actions. Using MediaPipe BlazePose for real-time pose estimation, it tracks 33 human body landmarks and maps gestures to keyboard inputs via pynput, making it compatible with any PC game. The system runs in real-time with low latency and can be extended for gaming, rehabilitation, or immersive learning environments.

Introduction

This section discusses the motivation behind creating immersive, controller-free gaming using computer vision and deep learning. It highlights the need for natural user interaction in modern games and presents the concept of a full-body interactive combat system that supports real-time pose tracking, adaptive gesture learning, multiplayer gameplay, and haptic feedback via Arduino devices.

Problem Statement

Develop a vision-based virtual combat system that translates real-world body movements into real-time game controls, removing the need for hardware controllers and enhancing engagement using pose estimation, AI, and haptic feedback.

Objectives

Create a controller-free interface using body gestures.

Implement real-time pose tracking with MediaPipe BlazePose.

Design a robust gesture detection mechanism with debouncing.

Introduce calibration for personalized gesture thresholds.

Map gestures to keypresses using pynput.

Maintain <100 ms latency and modular scalability.

Literature Review

Existing Systems: Traditional controllers, Kinect, VR trackers, and pose-estimation prototypes.

Limitations: High cost, limited portability, or poor real-time accuracy.

Proposed System: Uses a webcam and AI-based pose estimation for a low-cost, accurate, and scalable alternative.

Methodology

Uses MediaPipe BlazePose for landmark detection, OpenCV for frame processing, and pynput for control mapping. A calibration mechanism ensures individual accuracy, and a debouncing system minimizes gesture misfires.

System Architecture

The architecture involves five main layers:

Input Layer – Webcam captures movement.

Processing Layer – Pose detection and gesture classification using CNN and threshold logic.

Control Layer – Maps recognized gestures to game actions.

Networking Layer – Enables LAN multiplayer.

Feedback Layer – Provides haptic responses via Arduino.

Algorithms

BlazePose (MediaPipe) – For 2D keypoint detection.

Bounding-box gesture detection – For hit detection.

CNN (optional future enhancement) – For gesture classification.

Debouncing logic – To reduce false positives.

pynput – For simulating keyboard controls.

Implementation

Implemented in Python using OpenCV, MediaPipe, and pynput. The code processes webcam frames, detects poses, checks for gestures (like punches or crouches), and maps them to keyboard keys for in-game actions. Optional Arduino-based haptic feedback adds realism.

Results

Successfully recognized fighting gestures in real-time.

Works with standard PC games (e.g., Tekken 3 via emulator).

Achieved smooth gameplay, low latency, and accurate recognition.

Conclusion

The project effectively transforms human gestures into game actions, proving that AI-driven pose recognition can replace physical controllers. It has applications in gaming, fitness, and rehabilitation.

Future Scope

Use transformer-based models for better accuracy.

Expand gesture library (combo and defensive moves).

Integrate with VR/AR platforms and Unity/Unreal.

Add cloud multiplayer and adaptive difficulty.

Extend to healthcare and accessibility systems.

- 2. Project Explanation

What It Is

A vision-based AI system that recognizes human gestures via a webcam and maps them into virtual combat game controls. It uses MediaPipe for pose detection, OpenCV for image handling, and pynput for simulating keypress events.

Why It’s Done

To replace traditional controllers with natural, intuitive interaction.

To make games accessible to users with mobility limitations.

To demonstrate practical AI use in HCI (Human–Computer Interaction) and gaming.

How It Works

Webcam captures live frames.

MediaPipe BlazePose detects 33 key landmarks.

Gesture Engine checks positions (like wrist-to-head proximity) using bounding boxes.

Debouncing logic validates gestures to prevent misfires.

Recognized gestures are mapped to keyboard events via pynput.

Game receives key events → performs corresponding action.

How to Use It

Install Python + dependencies (opencv-python, mediapipe, pynput).

Run the script while a game/emulator is active.

Stand in front of the webcam and perform gestures.

The system triggers game actions in real time.

Why These Tools

Python: Simplifies AI & CV integration.

OpenCV: Efficient frame capture and processing.

MediaPipe: Lightweight real-time pose estimation.

TensorFlow/PyTorch: Deep learning flexibility for training models.

pynput: Seamless keyboard event simulation.

These were chosen for their speed, accuracy, open-source support, and ease of integration.

🔹 3. Algorithms & Implementation

Component	Technique Used	Purpose / Reason
Pose Detection	MediaPipe BlazePose	Tracks 33 body landmarks in real-time
Gesture Classification	Threshold & CNN (optional)	Recognizes actions (punch, kick, crouch)
Debouncing Logic	Temporal Filtering	Prevents false detections
Key Mapping	pynput Controller	Sends keyboard inputs
Visualization	OpenCV GUI	Displays detection overlay
Haptic Feedback	Arduino + PySerial	Adds physical feedback

Data Flow

Capture frame → RGB conversion

Detect landmarks

Compute geometric relationships

Recognize gesture

Map to keypress

Send input to game

Performance

Real-time processing at 30 FPS

Recognition accuracy: ~90%

Low latency (<100 ms)

🔹 4. Viva Questions & Answers

Basic

Q: What is your project title?

A: AI-Powered Gesture Recognition for Virtual Combat.

Q: What is the aim?

A: To create a controller-free gaming interface using real-time gesture recognition.

Q: Tools and libraries used?

A: Python, OpenCV, MediaPipe, TensorFlow, PySerial, and pynput.

Technical

Q: How does the system recognize gestures?

A: By tracking 33 body landmarks using MediaPipe and comparing joint positions to predefined thresholds.

Q: What algorithm is used for pose detection?

A: MediaPipe BlazePose.

Q: What is the role of CNN here?

A: CNN can be used for advanced gesture classification based on landmark images.

Q: How does OpenCV help?

A: It handles real-time video capture and frame visualization.

Q: How accurate is your model?

A: Around 90% accuracy with real-time response under 100 ms latency.

Conceptual

Q: What is gesture recognition?

A: The process of interpreting human body movements into machine-understandable commands.

Q: What is feature extraction here?

A: Extracting landmark coordinates like wrists, shoulders, hips for pattern analysis.

Q: How does AI contribute?

A: AI models process spatial-temporal relationships between landmarks to classify gestures.

Reasoning

Q: Why use MediaPipe instead of OpenPose?

A: MediaPipe is lightweight, fast, and works efficiently on CPUs without GPUs.

Q: Why Python?

A: Because of its strong AI and CV library ecosystem.

Q: How does your system ensure real-time response?

A: Optimized frame processing, minimal model complexity, and efficient debouncing.

Result & Usage

Q: What results did you get?

A: The system successfully recognized combat gestures with 90% accuracy and worked seamlessly with PC games.

Q: How does a user interact with it?

A: Stand in front of a webcam, perform gestures like punches or crouches, and the game responds instantly.

- 5. Key Concepts to Revise

Gesture Recognition

Feature Extraction

Pose Estimation

Convolutional Neural Networks (CNN)

MediaPipe Landmark Detection

Image Preprocessing

Training vs Testing

Accuracy, Precision, Recall

Real-Time Computer Vision

Human–Computer Interaction (HCI)

Debouncing Logic

Haptic Feedback Integration

Bounding-Box Detection

- 6. One-Minute Oral Summary

“My project is titled AI-Powered Gesture Recognition for Virtual Combat. It uses MediaPipe BlazePose and OpenCV to recognize real-time human gestures through a webcam and maps them into in-game controls using pynput. For example, a punch or kick performed by the player triggers corresponding actions in the game. The system provides a controller-free, immersive gaming

experience and can even include Arduino-based haptic feedback for realism. It runs with low latency, achieves around 90% accuracy, and can be extended to fitness, rehabilitation, and VR applications."

- 7. Additional Insights

System Flow (Simplified)

Player → Webcam → MediaPipe (Pose Detection)

→ Gesture Engine → Keypress Mapping (pynput)

→ Game Execution → Optional Haptic Feedback (Arduino)

Real-World Applications

Gaming (VR/AR and Esports)

Fitness and rehabilitation tracking

Interactive learning environments

Robotics control via body gestures

Accessibility for differently-abled users

Future Improvements

Add AI-driven adaptive gesture learning.

Integrate with AR/VR headsets.

Use transformer-based vision models.

Enable multiplayer over cloud servers.