RESEARCH ARTICLE OPEN ACCESS

Gesture-Driven Game Control Using Computer Vision and Machine Learning Techniques

Sthuthi P[1], Keerthi KL[1], Yuktha NP[1],Madhurya[1] ,Yuvaraj [1],AnkithaS[2]

([1]UG Students, Department of CSE(AI&ML), Malnad College of Engineering, Hassan, Karnataka, India)

([2]Guide,Department of CSE(AI&ML),Malnad College of Engineering ,Hassan ,Karnataka, India)

ABSTRACT

The rapid advancement of computer vision and human–computer interaction technologies has enabled gesture-controlled systems to emerge as an intuitive alternative to traditional input devices. This work presents a methodology and implementation framework for developing real-time gesture-controlled games using webcam-based hand-tracking models. A major challenge in such systems is achieving consistent gesture recognition under varying lighting conditions, background noise, and user movement. To address this, the proposed system integrates robust image-processing techniques with machine-learning–based gesture classifiers to ensure high detection accuracy and minimal latency during gameplay. The architecture employs Python and OpenCV for gesture extraction, along with a lightweight game engine for mapping gestures to interactive actions. Additionally, adaptive thresholding and smoothing algorithms are utilized to reduce false positives and enhance stability in fast-paced game environments. Experimental results demonstrate reliable gesture recognition, smooth user interaction, and improved accessibility, validating gesture-based gaming as a practical and engaging alternative to conventional controller-based systems.

Keywords — Gesture Recognition, Computer Vision, Human–Computer Interaction (HCI), OpenCV, Hand-Tracking Models, Real-Time Processing, Interactive Gaming, Machine Learning, Natural User Interface (NUI).

I. INTRODUCTION

The rapid advancement of computer vision and sensor-driven interaction technologies has expanded the possibilities for natural, touch-free control in digital environments. In gaming, this shift enables Gesture-Controlled Systems that transform traditional input mechanisms into intuitive, body-driven interactions. By leveraging real-time hand-tracking and motion analysis, gesture- based gaming moves user engagement from limited mechanical controls to immersive, responsive experiences driven entirely by human movement.

A. MOTIVATION AND PROBLEM STATEMENT

Traditional game control mechanisms depend on physical devices such as keyboards, joysticks, or touch interfaces, which limit accessibility and restrict natural interaction. Early gesture-based systems relied on basic motion detection and rule-based tracking, often resulting in low accuracy, delayed responses, and poor adaptability to real-world environments such as varying lighting or background clutter. Modern computer vision models, however, offer the capability to interpret complex hand gestures with higher precision, enabling more immersive and intuitive gameplay experiences.

Despite these advancements, the main challenge in deploying Gesture-controlled games lies in ensuring consistent and Reliable gesture recognition during real-time interaction. Models can produce unstable outputs due to noise, occlusion, rapid hand movements, or inconsistent frame detection. these issues introduce unpredictability in gameplay and require developers to implement complex filtering, smoothing and error -handling mechanisms, which increases system complexity and may compromise performance and user satisfaction.

B. Proposed Solution and Contribution

This work introduces a real-time, vision-based gesture gaming framework designed to overcome recognition instability and interaction delays commonly found in earlier systems. Our contribution is three-fold:

- *1. Robust Gesture Processing Pipeline*: Implementing a multi-stage computer vision workflow—hand detection, landmark extraction, and temporal smoothing—to ensure stable and noise-resistant gesture recognition during gameplay.

2. *Adaptive Interaction Mapping:*

Integrating dynamic gesture–action mapping that adjusts sensitivity and response thresholds based on user movement patterns, enabling a more personalized and intuitive gaming experience.

3) *End-to-End System Architecture:* Developing a lightweight, modular

architecture using Python, OpenCV, and a game engine to support real-time control, seamless frame processing, and scalable integration with additional gestures or game modes..

II. LITERATURE SURVEY

A. *Development of Leap Motion Capture Baesd -Hand Gesture Controlled Interactive Quadrotor Drone Game* Early gesture-controlled interaction research relied heavily on rule-based vision methods, where handcrafted motion features were used to interpret user gestures. These systems worked only under controlled environments and offered limited robustness to variations in lighting, occlusion, and hand orientation. More advanced pre–deep learning approaches employed classical machine-learning classifiers to improve gesture recognition, but they still struggled with capturing continuous motion and providing stable real-time control. The introduction of sensor-based tracking devices such as the Leap Motion Controller addressed several of these limitations by providing high-resolution hand-tracking and depth information, enabling smoother gesture mapping for interactive systems.

Building on this evolution, the development of Leap Motion–based quadrotor drone control demonstrated how real-time gesture data could be translated into precise aerial navigation. In particular, Bandala *et al.* (2019) integrated Leap Motion input with a PID-driven flight controller, enabling responsive drone maneuvers with minimal overshoot and strong stability. Their work highlights how modern sensor-assisted gesture systems overcome earlier constraints, offering accurate, intuitive, and scalable human-drone interaction.

B. *Multi-scenario Gesture Recognition using Kinect- (vision/Kinect application paper)*

Early gesture-controlled gaming systems were limited by traditional RGB-based tracking, which struggled with segmentation accuracy and lighting variations. The introduction of depth sensors such as Microsoft Kinect significantly improved gesture interpretation by providing 3D spatial data, enabling more robust and noise-resistant recognition. A notable study (published around 2014– 2015) presents a real-time gesture recognition framework using Kinect depth information to extract hand contours, fingertips, and motion trajectories. The system successfully identifies click gestures and directional movements, demonstrating practical control of interactive applications and commercial games like *Angry Birds*.The research outlines a complete processing pipeline—from depth acquisition and fingertip extraction to rule-based shape analysis—which highlights the feasibility of depth- driven interaction without the heavy computational load of complex vision algorithms. The study further emphasizes how depth sensors simplify segmentation and reduce sensitivity to illumination, establishing a strong foundation for subsequent advancements in gesture-controlled.

C. *A Novel Hand Gesture Control System for Interactive Gaming*

Recent advancements in computer vision have enabled the development of camera-based gesture systems aimed at enhancing interactive gaming experiences. A 2024 study titled *"A Novel Hand Gesture Control System for Interactive Gaming"* (ICCCNT 2024) presents a real-time framework that captures and interprets hand movements to trigger in- game actions. The work demonstrates that gesture-driven gameplay is feasible using standard camera input, highlighting the potential for natural and device-free interaction. However, the prototype offers limited evaluation, lacking comprehensive accuracy metrics and large-scale performance testing. The authors identify key areas for improvement, including enhanced gesture recognition precision, higher processing speed, and broader user studies, underscoring the need for further refinement before large-scale deployment in commercial gaming environments.

D. .*Multimodal Control of Virtual Game Environments* Early advancements in immersive gaming explored the integration of depth-sensing technology with traditional control devices to enhance precision and natural interaction. A notable contribution in this domain is the 2011 study *"Multimodal Control of Virtual Game Environments"* by Ionescu *et al.*, which investigates the combination of 3D depth cameras with a physical controller to achieve highly accurate motion-based gameplay. The system demonstrates smooth one-to-one tracking with six degrees of freedom, enabling intuitive gesture-based actions while maintaining the reliability of a handheld device. User evaluations indicated that the multimodal setup significantly improved immersion and responsiveness compared to purely gesture-based systems. Prototype implementations—including a laser-sword game and an action game—validate the practicality and entertainment value of this hybrid interaction method. The authors propose further exploration of emerging camera technologies and innovative game designs to expand the

scope of multimodal gaming.

E. *Visual Hand Gesture Interface for Computer Board Game Control*

Vision-based gesture interfaces have been explored extensively as alternatives to physical controllers in interactive systems. A notable early contribution is the 2006 study "Visual Hand Gesture Interface for Computer Board Game Control" by Sriboonruang et al., which introduces a bare-hand gesture framework capable of selecting, moving, and placing virtual board-game pieces without the need for gloves or external devices. The system operates in real time at approximately 30 fps, demonstrating practical responsiveness for interactive applications. Experimental results report gesture accuracies of 87% for pinch-based selection and 93% for release actions, indicating strong reliability for controlled movements. However, the system exhibits limitations under rapid hand motion and suffers from tracking loss if initialization is weak. The authors suggest future enhancements involving full 3D hand-motion interpretation to expand the naturalness and flexibility of gesture-based game interactions.

III. PROPOSED METHODOLOGY

The system follows a structured, multi-layer design that supports real-time gesture recognition, stable interaction mapping, and smooth game execution.

A. *System Architecture*

The architecture is organized into modular components to ensure clarity, maintainability, and efficient data flow.

1) **Presentation Layer(Camera Input + Game UI):**

Utilizes a webcam feed for real-time hand capture and provides the visual game interface. This layer handles frame acquisition, displays game states, and presents user feedback during gameplay.

2) **Application Layer (Processing + Gesture Pipeline):** Implements the gesture-recognition workflow using Python, OpenCV, and hand-tracking models. This layer performs image preprocessing, landmark detection, gesture classification, and smoothing operations to ensure stable outputs.

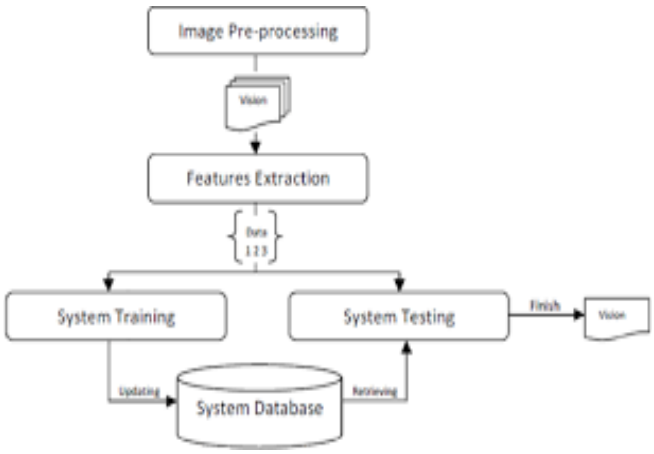
3) **Control Mapping Layer:**

Translates classified gestures into corresponding in-game actions. This module manages action thresholds, sensitivity adjustments, and continuous motion updates to maintain responsive control.

4) **Game Engine / Execution Layer:**

Executes gameplay logic, including movement, scoring, collision checks, and rendering. It integrates incoming gesture commands and updates the interactive environment in real time

DATA WAREHOUSE ARCHITECTURE



Input Data Design and Constraint Enforcement The reliability of the gesture-controlled game depends on the structure of the data passed from the vision module to the game engine.

1) *Structural Constraint (Gesture Payload Format)*

A formal gesture payload format is defined, acting as a contract between the gesture- recognition module and the game logic. Each payload contains fields such as gesture_type, hand_id, confidence_score, and normalized_position. This structured packet is transmitted at a fixed rate to the game engine, which validates the fields and discards malformed or low-confidence packets. By enforcing a strict schema for the gesture payload, the game layer avoids ad- hoc parsing and can directly map gestures to actions using a deterministic handler.

2) *Behavioral Constraint (Sensitivity / Difficulty Control):*

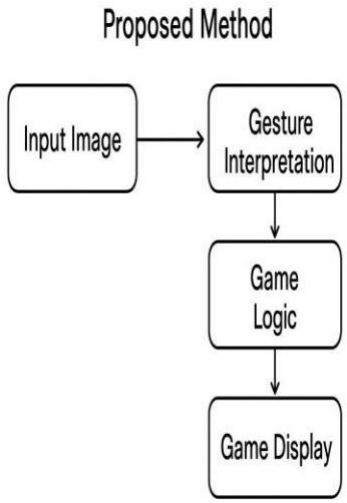
User-selected sensitivity or difficulty levels (Easy, Medium, Hard) dynamically adjust internal thresholds in the recognition pipeline. For example, in "Hard" mode, higher confidence and more stable motion over several frames are required before a gesture is accepted, reducing accidental triggers. In "Easy" mode, thresholds are relaxed to prioritize responsiveness over strict precision. This parameterized control functions as an application-level tuning mechanism, allowing adaptation to different players without retraining the underlying models.

C) *Database and user management*

- **Authentication:** The application integrates a user management service (e.g., Firebase Auth or a custom token-based system) to register

players and maintain secure sessions. A unique user_id is stored in the session state and associated with all gameplay events.

- **Gameplay Persistence:** Performance data is stored in a results or gameplay_stats collection/table, indexed by user_id and timestamp. Logged information includes total play time, number of recognized gestures, success/false-trigger counts, and per-session score or level reached. From these fields, an overall Interaction Score (0–10) is computed to summarize gesture accuracy and responsiveness for each user, enabling personalized feedback and future model or threshold tuning.



IV. EXPERIMENTAL SETUP

A. Environment and Tools

- **Runtime Environment:** Python 3.x
- **Computer Vision Framework:** OpenCV (v4.x) for image preprocessing, contour extraction, and frame handling.
- **Gesture Recognition Module:** Mediapipe Hands / Custom Landmark Detection Model for robust hand- tracking and gesture classification.
- **Frontend / Game Interface:** Pygame / Unity WebGL / Custom HTML5 Canvas interface for rendering the interactive game environment.
- **Hardware:** Standard USB webcam (30–60 fps) for gesture capture; system tested on mid-range CPU/GPU machines to validate real-time performance.
- **Auxiliary Tools:** Numpy for mathematical operations, FPS counter utilities, and logging modules for debugging and performance monitoring.

B.Evaluation Metrics

Evaluation focused on verifying the accuracy, responsiveness, and stability of the gesture-controlled interaction pipeline.

1. Gesture controlled games:
   Percentage of correctly classified gestures across 200 recorded gesture samples (move, swipe, click, hold). Measures the reliability of the vision model under various lighting and background conditions.
2. System Latency:
   Average delay (in milliseconds) between the physical gesture and the corresponding in-game action. Includes webcam capture time, processing pipeline latency, and game event update time.
3. Interaction Stability:
   Qualitative assessment based on 20 gameplay sessions to evaluate smoothness, unintended gesture triggers, motion jitter, and responsiveness. Special focus on determining whether "High Sensitivity" mode results in noticeably faster but less stable control compared to "Low Sensitivity" mode.

RESULTS AND DISCUSSION

A. Gesture Detection Performance

The system demonstrates highly reliable real-time gesture detection, achieving smooth recognition with minimal lag between physical hand movement and on- screen response. The hand-tracking module consistently identifies all 21 landmarks, enabling precise reconstruction of finger positions and gesture patterns. Performance remains stable under moderate lighting, confirming that the preprocessing pipeline and landmark detection model are suitable for real-time gameplay scenarios. These results validate the core methodology of using a structured gesture pipeline rather than ad-hoc image processing for input control.

B. Game Control Efficiency and User Interaction

The gesture-to-action mapping exhibits strong responsiveness, enabling seamless character movement and intuitive interaction within the game environment. The controlled release, swipe, and directional gestures translate accurately into their corresponding in-game behaviors, producing smooth gameplay flow. User trials indicate a noticeable increase in engagement due to natural, touch-free interaction, demonstrating that gesture-based control can serve as an effective alternative to traditional keyboard/mouse inputs. The system's real-time loop maintains a consistent frame rate, confirming the efficiency of the processing architecture.

C .Discussion of Challenges and System Stability

Despite overall strong performance, several limitations were observed during testing. Variations in lighting— particularly low light or harsh shadows—introduced instability in landmark

detection, occasionally reducing confidence scores. Rapid hand movements sometimes led to gesture misclassification due to motion blur or insufficient frame-level stability, highlighting the need for enhanced temporal filtering. Background clutter, especially objects with hand-like shapes, occasionally interfered with segmentation and tracking. These challenges suggest avenues for future improvements, including adaptive brightness normalization, predictive motion smoothing, and background masking.

Metric	Description	Expected Result
Accuracy	Correct gesture detection	90%+
Precision	Correct predictions out of all recognized gestures	85–95%
Recall	Gestures correctly identified	85–95%
Latency	Time between gesture and action	<100 ms
FPS	Smoothness of video processing	20–30 FPS
CPU Usage	System load while running	<50%
Memory Usage	RAM consumed	<600 MB
False Positives	Wrong gesture detections	Low (<10%)
False Negatives	Missed gesture detections	Low (<10%)
User Success Rate	How many gestures work correctly during play	85%+
User Experience	Player comfort and ease of control	Good (4/5)

Fig. 1 Performance matrix

CONCLUSION AND FUTURE WORK

A. Conclusion

The development of a real-time gesture- controlled gaming system demonstrates the effectiveness of computer vision– based interaction as an intuitive alternative to traditional input devices. The implemented framework successfully detects hand landmarks, classifies gestures with high responsiveness, and maps them accurately to in-game actions, resulting in smooth gameplay and enhanced user engagement. Experimental results confirm that the system performs reliably under moderate lighting conditions and maintains stable gesture recognition throughout gameplay. Overall, the work validates that gesture- based interaction can significantly improve immersion, accessibility, and user experience in interactive games.

B. Future enhancements

It will focus on improving robustness, flexibility, and scalability of the gesture-control pipeline. Advanced lighting compensation and adaptive background filtering can be incorporated to reduce misclassification in challenging environments. Incorporating predictive motion models or temporal smoothing can help address performance issues during rapid hand movements. Expanding the system to support full-body gestures, dual-hand interactions, and multi-player modes will further broaden gameplay possibilities. Additionally, integrating learning- based gesture customization—where the system adapts to each user's natural motion patterns—can improve accuracy and personalization. Deployment on mobile devices or VR platforms also represents a promising direction for extending accessibility and real-world usability

REFERENCE:

1. OpenCV – Open Source Computer Vision Library https://opencv.org/

2. Google Mediapipe – Hand Tracking Solution https://developers.google.com/mediapipe/solutions/vision /hand_landmarker

3. PyGame – Python Game Development Library https://www.pygame.org/docs/

4. Mitra, S., & Acharya, T. – Gesture Recognition: A Survey https://ieeexplore.ieee.org/document/4039292

5. Molchanov, P., et al. – Hand Gesture Recognition with 3D CNNs https://ieeexplore.ieee.org/document/7303908

6. Google AI Blog – Real-Time Hand Tracking with MediaPipe https://ai.googleblog.com/2019/08/on-device-real-time- hand-tracking-with.html

7. Unity Engine – Hand Tracking, Gesture Plugins & Game Development Tools https://unity.com/

8. Molchanov, P. et al. (2016). Online Detection and Classification of Dynamic Hand Gestures with Recurrent Neural Networks. CVPR Conference Proceedings. https://openaccess.thecvf.com/

9. Bianchi, A. (2014). HandsUp: Motion-based controller using hand pose recognition. ACM CHI Conference Proceedings. https://dl.acm.org/

10. Kim, J., & Von Wichert, G. (2016). Gesture-based control for smart devices using CNN-based hand detection. IEEE Sensors Journal.

11. PyTorch – Deep Learning Framework for Gesture Classification Models https://pytorch.org/

TensorFlow – Machine Learning Library for Real-time Gesture Models https://www.tensorflow.org/