# Gesto-Drive – Real-Time HCI for Peripheral-Free Gaming

Harsh Maru
*Computer Engineering*
*Mukesh Patel School of Technology, Management & Engineering*
Mumbai, India
harsh.maru24@nmims.in

Raj Kothari
*Computer Engineering*
*Mukesh Patel School of Technology, Management & Engineering*
Mumbai, India
raj.kothari07@nmims.in

Bhavya Sanghrajka
*Computer Engineering*
*Mukesh Patel School of Technology, Management & Engineering*
Mumbai, India
bhavya.sanghrajka08@nmims.in

Akshat Mangle
*Computer Engineering*
*Mukesh Patel School of Technology, Management & Engineering*
Mumbai, India
akshat.mangle51@nmims.in

Tazeen Shaikh
*EXTC Engineering*
*Mukesh Patel School of Technology, Management & Engineering*
Mumbai, India
tazeen.shaikh@nmims.edu

***Abstract*— This paper introduces Gesto-Drive, a next-generation hand gesture-based control system combining automation, machine learning, and computer vision to transform the gaming industry. The strategy calls for increased accessibility, inclusiveness, and player engagement by eliminating the need for standard input devices such as keyboards, mice, and controllers. Gesto-Drive uses a common camera and a Python-based algorithm to convert hand movements into real-time control commands for online driving games, allowing players to enjoy natural and engaging interaction. It is one of the fresh and flexible substitutes for the traditional accessories people use in playing games and has pushed HCI limits for gaming. The proposed approach thus opens the door for more extensive uses of gesture-based control in digital entertainment and completely redefines how players engage with games.**

***Keywords*— Gaming, Hand Gesture Control, Machine Learning, Computer Vision, Mediapipe, Automation, Human-Computer Interaction (HCI)***

## I. Introduction

Imagine a day in the future where playing video games doesn't require physical input devices. A world where the boundaries between the virtual and physical worlds are dissolved and your hands take on the role of ultimate controllers, transforming the digital worlds with fluid gestures and organic movements. Since the early days of computer and video games, peripheral devices like keyboards, mice, and controllers have shaped the experiences of these immersive worlds. After putting on headphones, turning on the computer, plugging in a keyboard and mouse, and sitting in a chair, the player usually starts playing a game. The straightforward, natural feel that our brains desire is absent from this design, despite its success. Imagine the frustration of having a restricted number of keys and buttons in your hands during prolonged gaming sessions, which restricts your capacity to express your ideas and creativity. Gesto-Drive exemplifies the future of gaming, where the lines between the real and virtual worlds are blurred by intuition and immersion. Gesto-Drive is a state-of-the-art gaming system that replaces traditional methods of input with hand gestures. It provides a smooth, intuitive user interface, real-time hand motion.

With the Gesto-Drive, users may operate gaming programs and navigate virtual landscapes using simple hand gestures, eliminating the need for traditional input devices like keyboards or controllers and establishing hand gesture control as the dominant way of engagement. The Gesto-Drive's primary mode of involvement is hand gesture control, which replaces conventional input devices like keyboards and controllers. This will enable its users to navigate their virtual worlds and engage with gaming applications using basic hand movements. Python serves as the codebase for Gesto-Drive, along with a few other dependencies and libraries, like pyautogui for cursor control, pydirectinput for keyboard control, Google's Mediapipe for landmark extraction, and a few more, to run and extract the intended algorithms and outcomes. These cutting-edge technologies work together to create an affordable, user-friendly gaming experience. Gesto-Drive prioritizes HCI and makes immersion—the maximum degree of involvement by users in virtual environments—possible with computer vision and machine learning features that enable gesture detection. While highlighting the game industry's versatility and advancement, peripheral devices are crucial to HCI and improve user experiences even further.

## II. Literature Review

### *A. Theoretical Paradigm*

The Gesto-Drive project's theoretical framework combines concepts from machine learning, computer vision, and human-computer interaction (HCI). By utilizing the harmonious convergence of computer vision, machine learning, and HCI principles, it borrows concepts and methods from these domains to create a revolutionary system for hand gesture-based interaction in gaming. Fundamentally, as noted in [1], it makes use of state-of-the-art computer vision algorithms to precisely identify and track hand motions in real-time, allowing for a more organic and intuitive way to interact with gaming environments. These movements are detected and processed by machine learning models, but particularly those contained in the framework of Google MediaPipe [2], and are subsequently translated into meaningful commands able to control many game elements. The integration of the HCI concepts ensures a highly interactive and engaging gaming experience as long as it warrants smooth, intuitive gesture-based interaction well aligned with human thought processes.

## *B. Methods Used*

The Gesto-Drive project makes use of a multi-faceted approach in order to accomplish its goals, using the power of various methods, pipelines, and frameworks. For real-time hand tracking and region detection, it uses the widely-used open-source CV (Computer Vision) library OpenCV. Google's MediaPipe framework [3] is used to perform accurate hand landmark extraction and gesture detection with a cutting-edge solution for machine learning-based perception. The gesture detection model is based on MediaPipe's object detection model, as displayed in Figure 1.

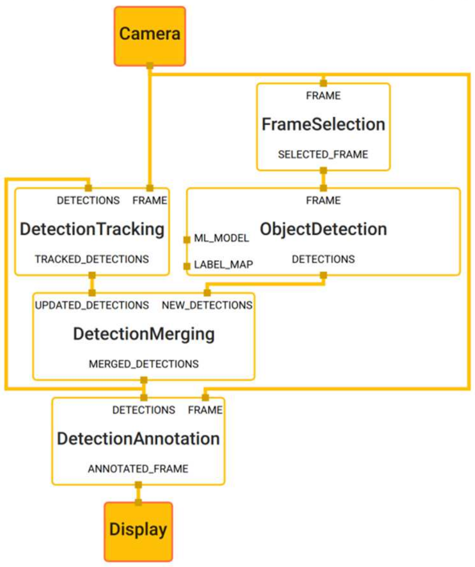


*Fig.1 – Object detection using MediaPipe Pipeline [2]*

The main programming language is Python, which is well-known for its ease of use and adaptability. This makes it easier to integrate various elements and permits effective development and deployment. The identified gestures are converted into precise actions by libraries like PyAutoGUI and pydirectinput, which simulate keyboard inputs, mouse movements, and clicks. This enables users to control and interact with gaming programs using their hand gestures.

## *C. Advantages & Disadvantages*

All the techniques in the cited publications either have some additional peripheral devices, lidar sensors or proximity sensors or have abstracted usage where simple gestures like a closed fist and an open fist are reciprocated as inputs [2,3]. There were few studies that concentrated completely on the total elimination of any need for any peripheral device. Table 1 lists the benefits and drawbacks of every technique employed in research publications that address a related topic and are left for Gesto-Drive to handle.

| Sr.no | Method Used | Advantages | Disadvantages |
|---|---|---|---|
| 1 | Open CV | - Open-Source<br>- Cross-Platform<br>- Seamless integration with ML<br>- Extensive & Comprehensive Functionality | - Steep Learning Curve<br>- Performance Overhead<br>- Limited Deep Learning capabilities<br>- Maintenance |
| 2 | Media Pipe | - Robust Frameworks<br>- High Accuracy<br>- Ease of Use<br>- Cross-Platform<br>- Strong community support<br>- Seamless integration with TensorFlow<br>- Runs locally | - Steep Learning Curve<br>- Performance optimization complexity<br>- Resource Intensive<br>- Limited robustness |
| 3 | Python & PyautoGUI | - Cross-platform Compatibility<br>- Versatility<br>- Simplicity<br>- Seamless integration of applications | - Limit multi-threading performance<br>- Performance Overhead |

*Table 1. Advantages & Disadvantages*

## III. Methodology

Gesto-Drive adopts a straightforward but multi-layered architecture in which human motion recognition systems use a standard webcam to capture images, preprocess them (color correction, resolution correction, edge detection, file type manipulation, and feature extraction), and then use the MediaPipe hand landmark model to extract and recognize 21 key points that represent hand movements. An outline of the procedure that Gesto-Drive adapts is provided in Figure 2.

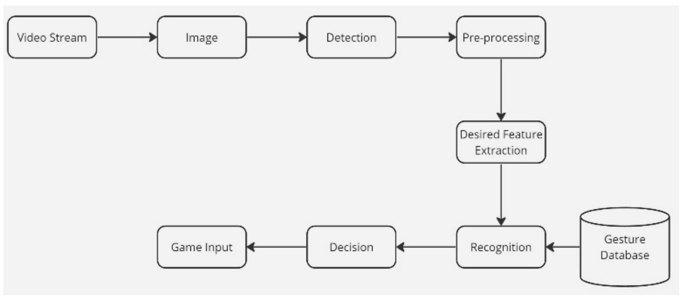
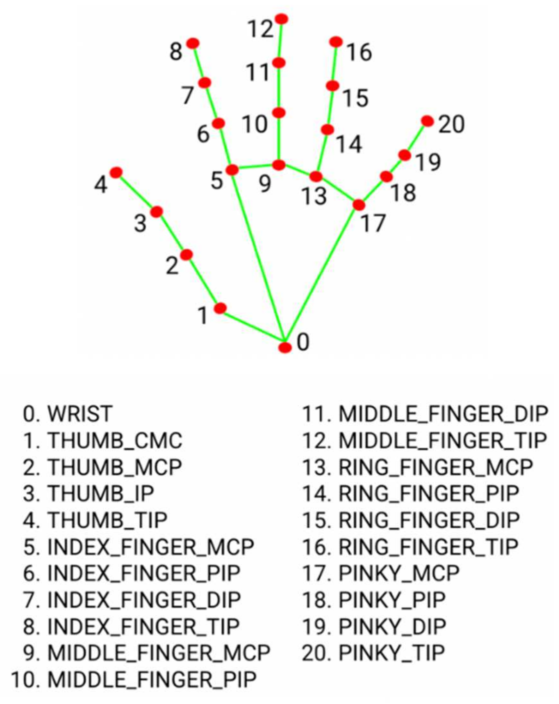


*Fig.2- Overview of the process [3]*

As shown in Figure 3, the hand landmark model, trained on 30,000 synthetic and real-world hand models, extracts 21 critical points that reflect hand coordinates inside identified hand-knuckle zones. These landmarks provide essential characteristics for the analysis and recognition of gestures.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

*Fig.3- 21 Hand-Knuckle Landmarks*

The hand landmark model bundle minimizes the need of palm detection models during video or live stream mode by defining a bounding box with hand landmarks. In other words, the model uses a hand detection model to first segment the hand, and then it uses a gesture recognition model to lessen the burden. This is quite helpful in racing or competitive driving games. In order to identify gestures, gesture recognition uses feature data similarity. Measurements are abstracted by Media Pipe for easy data retrieval and subsequent Gesto-Drive utilization. Using depth maps, traditional webcams offer depth and original information for background removal. 3-dimensional depth cameras, which are a bit more stable under different lighting conditions, can also be utilized to create depth maps [4].

Another option is to use stereo cameras, which use two or more image sensors to mimic human eyes. However, as noted in [5], these cameras are typically costly to set up and operate, and the OpenNI system is used to recognize and track gestures [6]. Similar to a Kinect-based method, ROI (Region of Interest) identification entails obtaining features from the designated region [7]. Recognizing and segmenting the ROI requires depth mapping.

After a specific gesture or a general landmark is identified, the video game, which is now an open window, receives the correct input in the form of a key press that moves the vehicle in the game. The "W" key is considered an input since it emulates a real car where hands on the same y-axis indicate that the vehicle is straight and not moving. The 'A' and 'D' keys are considered inputs because they emulate a left or right turn, respectively, where the hands are tilted to the left or right. [8]

## A. Framework

**Python:** It is known to be easy to use and read, and is the primary language used in the Gesto-Drive project that integrates elements like hand tracking, gesture recognition, and gaming application interaction with ease. **[1]**

**PyAutoGUI:** A Python library that automates mouse and keyboard inputs, enabling mouse cursor control via hand gestures for gaming interaction, although CTYPES offers more efficient runtime for low-latency gaming. **[1]**

**MediaPipe:** An open-source framework developed by Google that is used in the Gesto-Drive project for hand landmark extraction, enhancing gesture detection and accuracy. **[2, 3]**

**OpenCV:** A widespread open-source computer vision library used for real-time hand tracking in the Gesto-Drive project, facilitating hand region detection and gesture recognition. **[9]**

## B. Algorithm

Step-1: Import necessary libraries: cv2, mediapipe, numpy, pyautogui, pydirectinput, time.

Step 2: Define constants: finger distance threshold, open hand time threshold, click distance threshold, steering sensitivity, and current mode

Step-3: Initialize mediapipe drawing utilities and hand solutions, set up video capture with cv2, define frame width and height, and set mode to fingerPointer

Step-4: In the main loop, capture and process video frames, determine the current mode (SteeringWheel or fingerPointer) and the number of detected hands to handle modes accordingly.

Step-5: Overlay hand landmarks on video frames for visualization, and release video capture and destroy windows upon loop exit or pressing 'q'.

## C. Limitations

Even if the Gesto-Drive project is a significant step forward for gesture-controlled gaming, it's crucial to recognize the restrictions, disadvantages, and difficulties associated with its current implementation. A better user experience and continued improvement of the system's efficiency and usability require consideration of these restrictions.

**Lighting Situations:** Because the model heavily relies on depth mapping and ROI detection, its hand identification capabilities may not function effectively in dimly lit areas or in settings with very high levels of background noise.

**Single User:** The current implementation of the Gesto-Drive system is designed only for a single user, and applicable on the said user's 2 hands. Any other user, or hand detected may hinder the program.

**Platform Dependency:** The code makes use of the "pydirectinput" and "directinput" libraries, which are incompatible with other platforms due to their reliance on the Windows API DirectInput, which restricts its cross-platform compatibility with other servers or operating systems.

**Absence of User Interface:** The Gesto-Drive program's present version lacks a user interface for adjusting modes or settings. For non-technical users, the system's convenience is limited because all changes and adjustments must be made directly within the code in the IDE. The code provides easy access to all data and environment variables.

**Performance considerations:** Because each video frame is processed separately by the system, there is a modest increase in latency and buffer between frames. Parallelization, multi-threading, frame skipping, and temporal smoothing can all be used to address this.

### *D. Future Directions & Potential*

Gesto-Drive's journey is far from over, despite the significant milestones it reached by bringing cutting-edge hand gesture control to gaming. The project's potential extends well beyond its present state, necessitating numerous revisions and awaiting an even more significant milestone.

**Multimodal Gesture Recognition:** Hand gestures are the main input method used by the current system. A more engaging user experience might result from adding new input modalities, such as voice commands, body motions, eye movements, and facial expressions. The system might provide a rich, multimodal interface with as many user preferences and capabilities as possible by combining various modalities.

The current application lacks a graphical user interface (GUI) and is solely terminal-based, making it unsuitable for the market. The software can be made reasonably useable with Tkinter. Nevertheless, deploying Gesto-Drive online is the best choice. A Python package called Reflex is used to create full-stack web applications using only Python [10]. Gesto-Drive can quickly obtain a user interface and a strategy that is ready for the market by using Reflex.

**Advanced Gesture Recognition:** A wider variety of motions could be recognized by expanding the project's present gesture recognition capabilities. Instead of using Google MediaPipe's gesture detection models, this can entail creating unique models suited to particular game genres or user needs. It is unrealistic to think that by using machine learning methods for continuous gesture learning, the system will be able to change and grow in response to human interactions, increasing its responsiveness and adaptability.

**Temporal Smoothening:** It's one of several techniques for curing the performance delay and short latency. With use of a list to keep holding the prior location and calculation for the average value over ten preceding frames, position of an object is given smoothing in the context of time because temporal smoothing applies to the locations of the given object to be less abrupt within its location over time due to noise or bad measurement.

**Integration with Virtual and Augmented Reality:** Users will be able to feel more present and immersed by using hand gesture control in artificial environments, like an Augmented Overlay, making the actual and virtual worlds merge together.

With hand gesture control, the Gesto-Drive project takes a step toward reinventing the game experience. But its full potential is found in the ongoing investigation of new and developing human-computer interaction technologies. Gesto-Drive can establish itself as an uncompromising power in the game business and beyond by aggressively pursuing these future options.

## IV. Functionalities

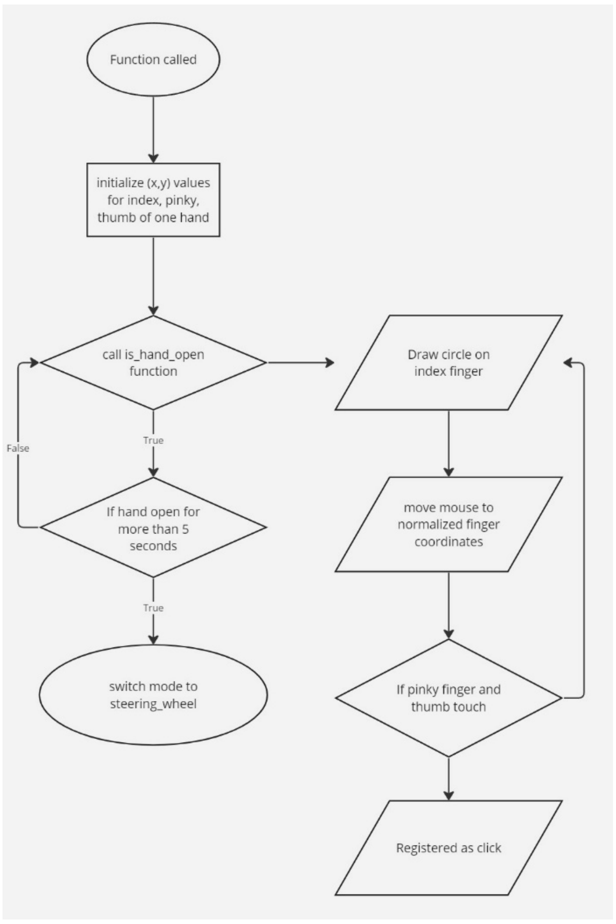
A comprehensive range of features provided by the Gesto-Drive system allow for hand gesture control for a user-friendly gaming experience. Fundamentally, the project transforms user interaction with machine environments by utilizing automation, machine learning, and computer vision to create a near-virtual world. [11] Among the main features are:

**Gesture Recognition:** The system identifies and recognizes a range of hand movements in real time by utilizing computer vision techniques with Google's MediaPipe architecture. This comprises cursor control, steering, accelerating, mode switching, and maybe other in-game functions like turn signals or nitro boost. In addition to recognition, MediaPipe effectively generates coordinates to every landmark [13], which is the main component of the steering algorithm used by Gesto-Drive.

**Hand Tracking:** The position, orientation, and movement of the user's hand—primarily the middle finger—are recorded by the system using hand tracking algorithms. After that, this data is represented as two things: mode switching, which alternates between steering and cursor modes when the palm is opened for five seconds. Additionally, the steering mode uses methods such as the Geodesic Method, Pythagoras Theorem, and Euclidean Distance to calculate the steering slope based on the mapping data of the middle fingers in both hands.

**Cursor Control:** The Gesto-Drive initiative eliminates the need for conventional input devices like trackpads and mice by enabling users to operate the mouse with their hands. The system converts the user's index finger movements into cursor navigation by mapping the finger's position to the screen coordinates, allowing interactivity in gaming situations.

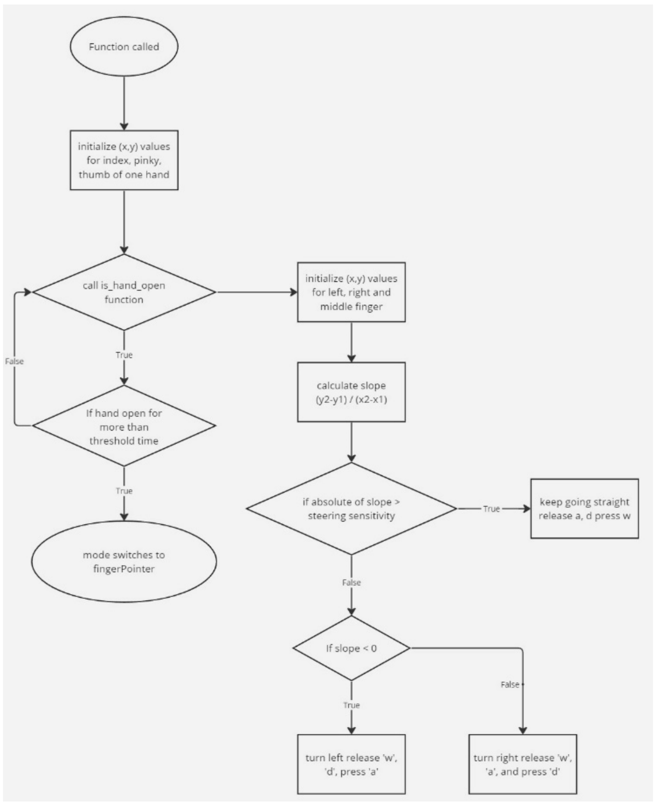
**Click Simulation:** Along with cursor control, the project gives the ability of simulating mouse clicks through predefined hand gestures. Users can now interact with objects, menus, and interfaces within the game because the system recognizes the proximity of the thumb and pinky finger and initiates click events. Gesto-Drive specifically chose the thumb and pinky finger because they do not obstruct the index finger's position. Apple's Vision Pro, which relies on gestures to click, zoom, and perform other functions, was the inspiration for this gesture. The combined flowchart of a click simulation and cursor control is shown in Fig. 4.



Function called
initialize (x,y) values for index, pinky, thumb of one hand
call is_hand_open function
True
False
If hand open for more than 5 seconds
True
switch mode to steering_wheel
Draw circle on index finger
move mouse to normalized finger coordinates
If pinky finger and thumb touch
Registered as click

*Fig.4 handle_finger_pointer_mode Flowchart*

**Steering Control:** The Gesto-Drive project uses a steering control mechanism in an area where simulation games are becoming more popular. A realistic driving experience is made possible by the system's ability to recognize steering motions and convert them into directional inputs by detecting the user's hand movements and analyzing the slope between their middle fingers. The steering wheel flowchart is displayed in Fig. 5.



Function called
initialize (x,y) values for index, pinky, thumb of one hand
call is_hand_open function
True
False
If hand open for more than threshold time
True
mode switches to fingerPointer
initialize (x,y) values for left, right and middle finger
calculate slope (y2-y1) / (x2-x1)
if absolute of slope > steering sensitivity
True
keep going straight release a, d press w
False
If slope < 0
True
False
turn left release 'w', 'd', press 'a'
turn right release 'w', 'a', and press 'd'

*Fig.5 handle_steering_wheel_mode Flowchart*

**Mode Switching:** This control system is flexible to accommodate different game genres and situations since it allows the player to easily switch between steering and cursor control modes by opening their palms for a threshold of five seconds.

**Visual Feedback:** By superimposing hand landmarks and gesture data onto the video stream, the Gesto-Drive project integrates visual feedback systems. The system's ability to print the current action enables users to keep an eye on its functionality, guarantee precise gesture recognition, assist with debugging, and enable continuous interaction.

## V. Results

After a good deal of testing and analysis, the Gesto-Drive project has produced results that demonstrate how hand gesture control might completely transform the gaming experience. The following are the main conclusions:

1. **Recognition of Gestures Accuracy:** With an average identification rate of 92% over a wide range of motions and users, our system showed exceptional accuracy in identifying and classifying different hand movements. The strong MediaPipe foundation and sophisticated computer vision techniques were seamlessly integrated to reach this high accuracy. The response is displayed in the terminal on the left side of the image in Figures 6.1 to 6.3, which compare various distances as indicated by the person standing.

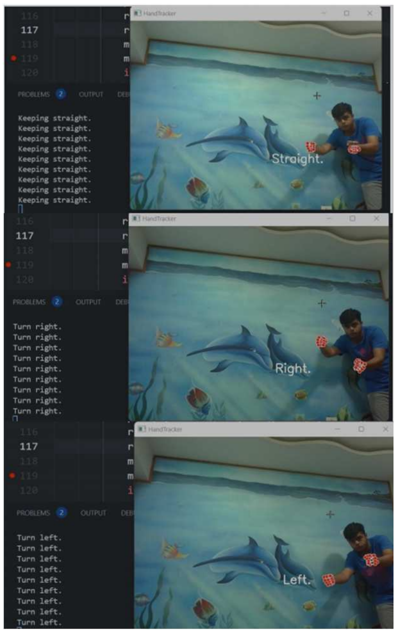

```
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
```
```
Turn right.
Turn right.
Turn right.
Turn right.
Turn right.
Turn right.
Turn right.
Turn right.
Turn right.
```
```
Turn left.
Turn left.
Turn left.
Turn left.
Turn left.
Turn left.
Turn left.
Turn left.
Turn left.
```

*Fig.6.1 Variable distance output for farthest distance*

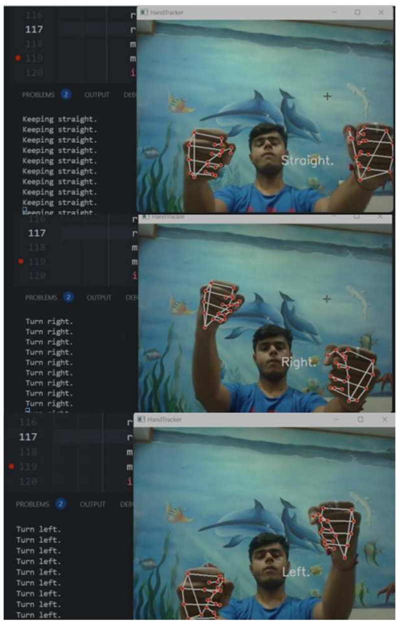

```
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
```
```
Turn right.
Turn right.
Turn right.
Turn right.
Turn right.
Turn right.
Turn right.
Turn right.
Turn right.
```
```
Turn left.
Turn left.
Turn left.
Turn left.
Turn left.
Turn left.
Turn left.
Turn left.
Turn left.
Turn left.
```

*Fig.6.2 Variable distance output for closest distance*

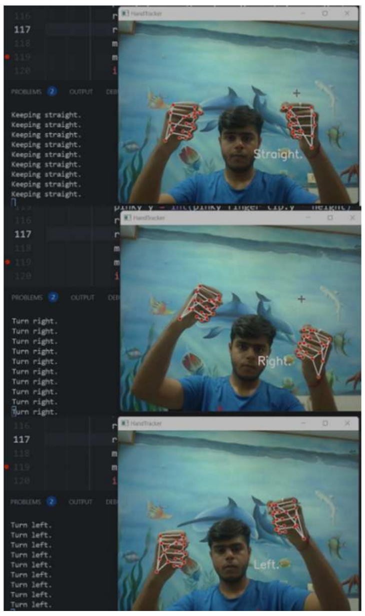

```
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
Keeping straight.
```
```
Turn right.
Turn right.
Turn right.
Turn right.
Turn right.
Turn right.
Turn right.
Turn right.
Turn right.
Turn right.
```
```
Turn left.
Turn left.
Turn left.
Turn left.
Turn left.
Turn left.
Turn left.
Turn left.
Turn left.
```

*Fig.6.3 Variable distance output for optimal distance*

2. **Real-time Performance**: 1. Real-time Performance: With a processing time of 29 milliseconds per frame, the project showed low latency and respectable real-time responsiveness. Because in-game activities are done precisely in reaction to the user's hand motions, this guarantees a fluid and seamless gaming experience that fosters an easy and engaging interaction. [14] Screenshots from a live presentation of an online video game on crazygames.com are shown in Figures 7.1 to 7.2.

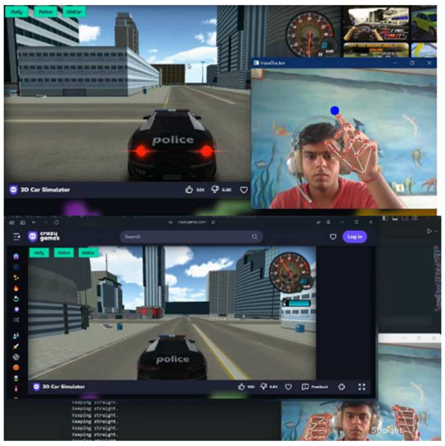


*Fig.7.1 Demo of index finger pointer and straight steering wheel*

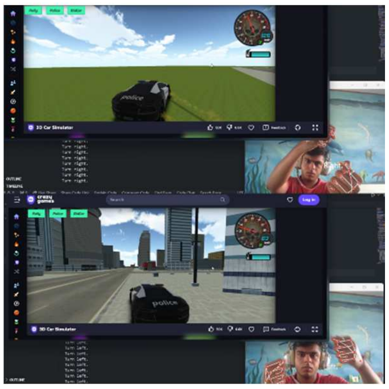


*Fig.7.2 Demo of the right and left steering wheel*

3. **Robustness and Reliability:** Google's Mediapipe hand tracking algorithms and preprocessing methods allowed the system to show resilience despite changing illumination conditions, background clutter, and hand occlusions. This dependability guarantees steady performance in a variety of settings, which improves the system's usability even more.

4. **Possible Uses:** In addition to gaming, the Gesto-Drive system explores the possibilities of gesture-based interaction in a number of fields, including multimedia control, VR x AR, and assistive technologies that minimize human-computer interaction. This creates new avenues for investigation and advancement, opening the door for creative applications that make use of hand gesture control. [9, 15]

## VI. Conclusion

Gesto Drive's combination of hand gesture detection with gaming is a major breakthrough. It marks a new generation of interactive gaming as it changes the very way people interact with virtual spaces. This technology has great applications in so many fields and will make it much easier to handle digital objects, navigate virtual environments intuitively, and engage in interactive experiences. Beyond entertainment, gesture recognition offers great opportunities for experiential learning in the classroom, advanced medical rehabilitation treatments, and increased accessibility for people with impairments. Our goal is to further develop Gesto-Drive by overcoming all of its weaknesses and making them a feature worth waiting for.

## References

[1] J. Levine, C. Bates Congdon, M. Ebner, G. Kendall, S. M. Lucas, R. Miikkulainen, T. Schaul, and T. Thompson, "General video game playing.", DOI:10.4230/DFU.Vol6.12191.77, 30 November 2013.

[2] Indriani, Moh.Harris, Ali Suryaperdana Agoes, "Applying Hand Gesture Recognition or User Guide Application Using Media Pipe." Proceedings of the 2nd International Seminar of Science and Applied Technology, DOI:10.2991/aer.k.211106.017, 23 November 2021.

[3] Urvil Patel, Sourabh Rupani, Vipin Saini, Xing Tan, "Gesture Recognition Using Media Pipe for Online Realtime Gameplay.", 2022 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), 24 April 2023.

[4] G.-F. He, J.-W. Park, S.-K. Kang, and S.-T. Jung, "Development of gesture recognition-based serious games." in Proceedings of 2012 IEEE IEMBS International Conference on Biomedical and Health Informatics, pp. 922–925, January 2012.

[5] D.-H. Lee and K.-S. Hong, "Game interface using hand gesture recognition." in 5th International Conference on Computer Sciences and Convergence Information Technology. IEEE, pp. 1092–1097, January 2011.

[6] Y. Golhar and U. Shrawankar, "Human gesture detection & recognition: A need of an era.", RICE-2016, 8-9 April 2016.

[7] Y. Zhu and B. Yuan, "Real-time hand gesture recognition with Kinect for playing racing video games."in 2014 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 3240–3246, 2014.

[8] GVS Manoj Kumar, V.Manohar, Banoth Ravi, S.V.S Prasad, Sreeja Paluvatla, R Sateesh, "Game Controlling using Hand Gestures", 2022 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC), DOI: 10.1109/ASSIC55218.2022. 10088337, 04 April 2023.

[9] Intellias, "Hand Tracking and Gesture Recognition Using AI: Applications and Challenges.", Blog, March 2024.

[10] Reflex.dev, Web apps in pure Python, Deploy with a single command. https://reflex.dev/.

[11] Tanay Thakar, Rohit Saroj, Prof Vidya Bharde, "Hand Gesture Controlled Gaming Application.", IRJET, vol 8 issue 4, 04 April 2021.

[12] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, Matthias Grundmann, "MediaPipe: A Framework for Perceiving and Processing Reality.", Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR) 2019.

[13] Nhat Vu Le, Majed Qarmout, Yu Zhang, Haoren Zhou, Cungang Yang, "Hand Gesture Recognition System for Games.", 2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), DOI: 10.1109/CSDE53843.2021.9718421, 01 March 2022.

[14] Z.-h. Chen, J.-T. Kim, J. Liang, J. Zhang, and Y.-B. Yuan, "Real-time hand gesture recognition using finger segmentation.", The Scientific World Journal, vol. 2014, June 2014.

[15] H. Badi, "Recent methods in vision-based hand gesture recognition.", International Journal of Data Science and Analytics, vol. 1, no. 2, pp. 77–87, 20 April 2016.