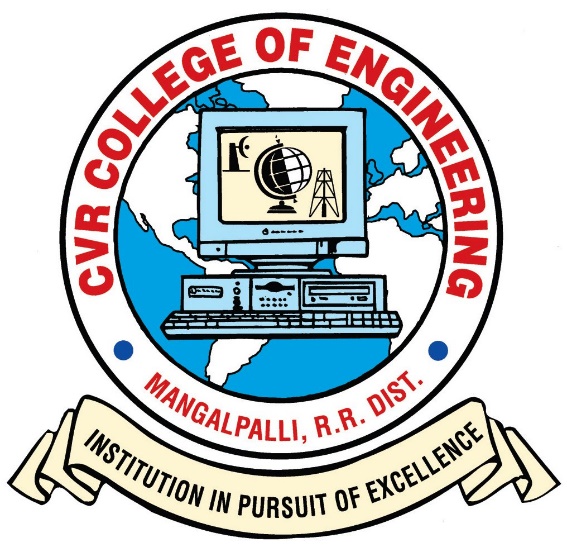
****

# CVR COLLEGE OF ENGINEERING

**(*An Autonomous Institution, NAAC Accredited and Affiliated to JNTUH, Hyderabad*)**

Vastunagar, Mangalpalli(V),Ibrahimpatnam(M),

Rangareddy (D), Telangana- 501 510

**DEPARTMENT OF CSE (ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)**

# Gesture-Based Game Control Using Computer Vision

Submitted By:  
  
D. Poojitha — Roll No: 23B81A6698  
K. Poojitha — Roll No: 23B81A6699

## Introduction

Traditional game controls depend on keyboards or controllers, which limit intuitive user interaction. This project presents a gesture-based control system using computer vision techniques to recognize hand gestures such as thumbs up, open palm, or fist. These gestures are then mapped to game actions like jumping, ducking, or idling. The Chrome Dino game was chosen as a case study to demonstrate real-time control using MediaPipe, OpenCV, and PyAutoGUI.

## Objectives

• Implement real-time hand gesture recognition.

• Map gestures to in-game actions such as Jump, Duck, and Idle.

• Provide a contactless, intuitive game control mechanism.

• Demonstrate integration of MediaPipe, OpenCV, and PyAutoGUI.

## System Requirements

Hardware Requirements:

Processor: Intel i3 or higher  
RAM: 4 GB or more  
Storage: Minimum 100 MB free space

Software Requirements:

Operating System: Windows / Linux  
Python Version: 3.x  
Libraries: MediaPipe, OpenCV, PyAutoGUI  
IDE: VS Code / PyCharm / Jupyter Notebook

## Methodology

The system uses computer vision-based hand gesture recognition to control game movements. MediaPipe is employed for real-time hand landmark detection. Each recognized gesture is classified and mapped to specific in-game actions. The implementation pipeline involves the following stages:

1. Capture live video feed from webcam using OpenCV.

2. Detect hand landmarks using MediaPipe.

3. Classify hand gestures based on landmark patterns (thumbs up, fist, open palm).

4. Map each gesture to a specific keyboard event using PyAutoGUI.

5. Send keyboard actions (e.g., SPACE for jump, DOWN for duck) to control the game.

## Implementation

Python is used to implement the complete system. MediaPipe provides real-time hand landmark detection, OpenCV handles image capture and frame processing, and PyAutoGUI automates keyboard input.

## Project Structure

gesture-based-game-control/

├── README.md

├── requirements.txt

└── src/

├── \_init\_.py

├── main.py

└── detector.py

## Dectector.py

import cv2

import mediapipe as mp

class HandGestureDetector:

"""

Detects hand gestures using MediaPipe Hands:

👍 Thumbs up -> Jump

✊ Fist -> Duck

🖐 Open palm -> Idle

"""

def \_init\_(self, max\_hands=1, detection\_confidence=0.7, tracking\_confidence=0.7):

self.hands = mp.solutions.hands.Hands(

max\_num\_hands=max\_hands,

min\_detection\_confidence=detection\_confidence,

min\_tracking\_confidence=tracking\_confidence

)

self.mp\_draw = mp.solutions.drawing\_utils

self.gesture = None

def detect\_gesture(self, frame):

h, w, \_ = frame.shape

frame\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

result = self.hands.process(frame\_rgb)

gesture = "Idle"

if result.multi\_hand\_landmarks:

for hand\_landmarks in result.multi\_hand\_landmarks:

self.mp\_draw.draw\_landmarks(frame, hand\_landmarks, mp.solutions.hands.HAND\_CONNECTIONS)

# Landmark indices (see MediaPipe Hands documentation)

thumb\_tip = hand\_landmarks.landmark[4]

thumb\_ip = hand\_landmarks.landmark[3]

index\_tip = hand\_landmarks.landmark[8]

middle\_tip = hand\_landmarks.landmark[12]

ring\_tip = hand\_landmarks.landmark[16]

pinky\_tip = hand\_landmarks.landmark[20]

# Convert to pixel coordinates

t\_y = int(thumb\_tip.y \* h)

ti\_y = int(thumb\_ip.y \* h)

i\_y = int(index\_tip.y \* h)

m\_y = int(middle\_tip.y \* h)

r\_y = int(ring\_tip.y \* h)

p\_y = int(pinky\_tip.y \* h)

# --- Gesture Logic ---

# Thumbs Up: thumb tip above thumb IP, other fingers down

if (t\_y < ti\_y) and (i\_y > ti\_y) and (m\_y > ti\_y) and (r\_y > ti\_y) and (p\_y > ti\_y):

gesture = "Jump"

# Fist: all fingertips close together (down)

elif abs(i\_y - m\_y) < 20 and abs(m\_y - r\_y) < 20 and abs(r\_y - p\_y) < 20:

gesture = "Duck"

else:

gesture = "Idle"

self.gesture = gesture

return gesture, frame

## Main.py

import cv2

import pyautogui

import time

from src.detector import HandGestureDetector

def main(camera\_index=0):

detector = HandGestureDetector(max\_hands=1)

cap = cv2.VideoCapture(camera\_index)

print("🎮 Gesture-Based Dino Game Controller Started!")

print("➡ 👍 Thumbs Up = Jump")

print("➡ ✊ Fist = Duck")

print("➡ 🖐 Open Palm = Idle/Run")

print("Press 'Q' to quit.\n")

time.sleep(2)

while True:

ret, frame = cap.read()

if not ret:

break

frame = cv2.flip(frame, 1)

gesture, annotated = detector.detect\_gesture(frame)

# Game control mapping

if gesture == "Jump":

pyautogui.press("space")

elif gesture == "Duck":

pyautogui.keyDown("down")

else:

pyautogui.keyUp("down")

cv2.putText(

annotated, f"Gesture: {gesture}", (20, 50),

cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2

)

cv2.imshow("Dino game gesture control (press Q to exit)", annotated)

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

break

cap.release()

cv2.destroyAllWindows()

if \_name\_ == "\_main\_":

main()

## requirements.txt

opencv-python>=4.7.0

mediapipe>=0.10.0

pyautogui>=0.9.53

numpy>=1.24.0

pillow>=9.0.0

## Execution

1. python -m venv venv
2. Windows: venv\Scripts\activate
3. pip install --upgrade pip
4. pip install –r requirements.txt
5. python -m src.main

## About MediaPipe

**MediaPipe** is an open-source framework developed by Google for building real-time computer vision and machine learning pipelines. It provides pre-trained models for **hand, face, and pose detection**, making it ideal for gesture recognition tasks.

In this project**, MediaPipe Hands** is used to detect **21 key landmarks** of the human hand from a live webcam feed. These landmarks are analyzed to identify gestures like **open palm, fist, and thumbs up**, which are then mapped to game actions (Jump, Duck, Idle).

MediaPipe ensures **fast and accurate hand tracking**, enabling smooth, contactless game control when integrated with **OpenCV** for video input and **PyAutoGUI** for keyboard simulation.

## Results and Discussion

The system successfully recognized hand gestures in real-time and controlled the Chrome Dino game accordingly. The model achieved smooth gesture tracking with minimal delay under good lighting conditions. Users could jump or duck without touching any physical device, achieving a contactless experience.

## Advantages

• Enables contactless interaction with games.

• Provides an intuitive user experience.

• Operates in real time with minimal delay.

• Can be extended for gesture-controlled robotics or AR/VR applications.

## Limitations

• Performance decreases under poor lighting or occlusions.

• Limited gesture set (Jump, Duck, Idle) in current version.

• Dependent on camera quality and frame rate.

## Future Work

• Add more gestures (e.g., pointing to shoot, wave to restart).

• Integrate AI-based gesture classification for higher accuracy.

• Optimize processing for low-end hardware.

• Extend to mobile or AR/VR gesture-based gaming.

## Conclusion

The Gesture-Based Game Control project demonstrates the effective use of computer vision for real-time gesture recognition. By integrating MediaPipe, OpenCV, and PyAutoGUI, the project achieves intuitive, contactless control of the Chrome Dino game. This system enhances user engagement and provides a foundation for future developments in gesture-based interfaces.

## References

1. MediaPipe Hands Documentation

2. OpenCV Python Documentation

3. PyAutoGUI Documentation