Nombre: Edgar Javier Fregoso Cuarenta

Registro: 22310285

Materia : visión Artificial

Reporte de Práctica: Reconocimiento de Gestos "Piedra, Papel o Tijera" con Visión Artificial

Objetivo de la práctica:

Implementar un sistema de visión artificial en Python utilizando MediaPipe y OpenCV para detectar gestos de la mano y clasificarlos como piedra, papel o tijera. El objetivo es aplicar técnicas básicas de reconocimiento de patrones visuales y explorar aplicaciones interactivas sin contacto.

Fundamento teórico:

El reconocimiento de gestos con visión artificial es una técnica basada en la detección de formas, contornos o puntos clave del cuerpo humano (en este caso, la mano). Para este proyecto se utiliza:

- **MediaPipe:** Framework desarrollado por Google para la detección y seguimiento en tiempo real de puntos clave del cuerpo. En esta práctica se emplea el modelo Hands para extraer 21 puntos clave de la mano.
- **OpenCV:** Librería de procesamiento de imágenes en tiempo real que permite capturar video, dibujar sobre frames y manipular matrices de imagen.
- **Python:** Lenguaje principal del desarrollo por su integración sencilla con bibliotecas de visión artificial.

Se interpreta el gesto a partir de los dedos levantados:

- **0 dedos levantados:** Piedra (✊)
- **2 dedos levantados:** Tijera (✌️)
- **5 dedos levantados:** Papel (🖐️)

Desarrollo y descripción del funcionamiento:

1. **Captura de video:**
   Se usa la cámara web para obtener video en tiempo real, el cual se analiza frame por frame.

2. **Detección de mano:**
   Con MediaPipe se detectan los puntos clave (landmarks) de la mano. Si hay una mano visible en el cuadro, se procesan las posiciones relativas de sus dedos.
3. **Clasificación del gesto:**
   Se analiza si cada dedo está levantado o no, comparando la posición de las articulaciones del dedo contra su base. A partir del número de dedos extendidos, se clasifica el gesto.
4. **Visualización:**
   El gesto detectado se muestra en pantalla sobre el video en vivo, y se actualiza en tiempo real con cada nuevo frame.

Problemas presentados:

- **Compatibilidad con versiones de Python:**
  MediaPipe aún no es compatible con Python 3.13, por lo que fue necesario instalar Python 3.10 para lograr la instalación correcta.
- **Falsos positivos en los gestos:**
  En algunas ocasiones, la clasificación de gestos falla debido a:
  - Ángulo de la mano con respecto a la cámara.
  - Detección errónea de dedos parcialmente extendidos.
  - Iluminación insuficiente o fondo con ruido.
- **Limitación de detección:**
  MediaPipe está optimizado para manos en posiciones relativamente centradas y orientadas al frente. Si la mano está muy inclinada o mal iluminada, puede no detectarla correctamente.

Conclusiones:

Esta práctica permitió comprender de forma aplicada cómo se pueden usar técnicas modernas de visión artificial para interpretar gestos humanos y desarrollar interfaces sin contacto. El uso de MediaPipe simplifica mucho la implementación, permitiendo un reconocimiento eficiente en tiempo real.

Además, el ejercicio mostró la importancia de seleccionar versiones de software compatibles y de considerar condiciones físicas (iluminación, orientación) al trabajar con visión por computadora.

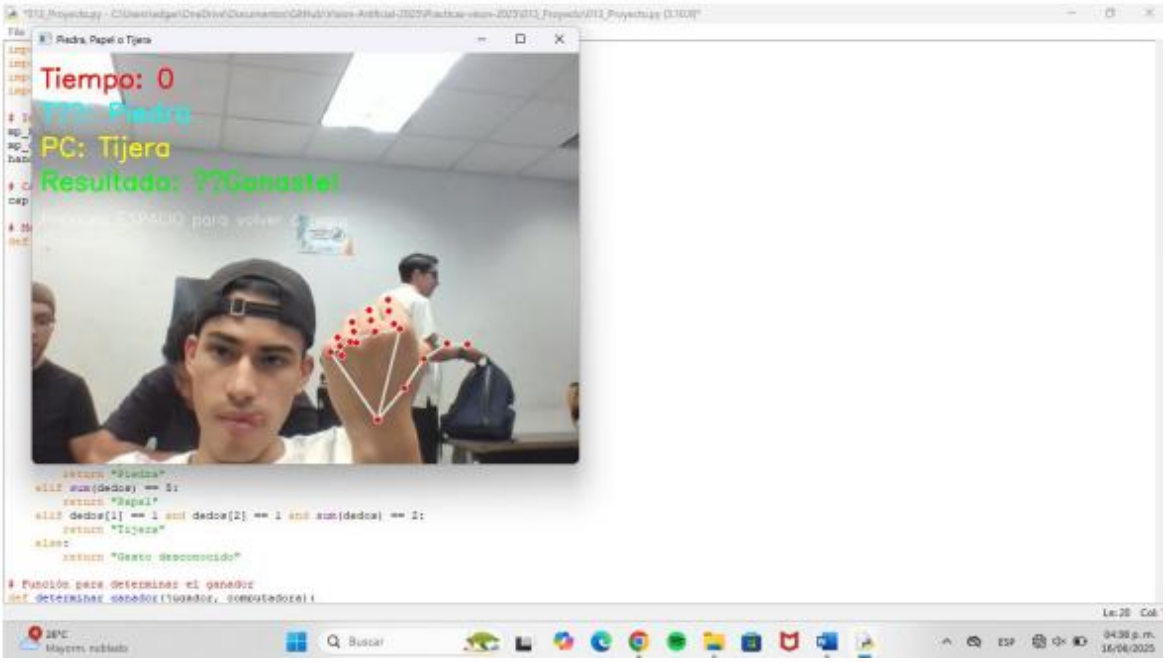
```
import cv2
import mediapipe as mp
import random
import time

# Inicializa MediaPipe
mp_hands = mp.solutions.hands
mp_drawing = mp.solutions.drawing_utils
hands = mp_hands.Hands(min_detection_confidence=0.7,
min_tracking_confidence=0.7)

# Cámara
cap = cv2.VideoCapture(0)

# Mapeo de gestos
def detectar_jugada(landmarks):
  dedos = []

  # Dedos índice a meñique
  for i in [8, 12, 16, 20]:
    if landmarks.landmark[i].y < landmarks.landmark[i - 2].y:
      dedos.append(1)
    else:
```

```
            dedos.append(0)

    # Pulgar
    if landmarks.landmark[4].x > landmarks.landmark[3].x:
        dedos.insert(0, 1)
    else:
        dedos.insert(0, 0)

    if sum(dedos) == 0:
        return "Piedra"
    elif sum(dedos) == 5:
        return "Papel"
    elif dedos[1] == 1 and dedos[2] == 1 and sum(dedos) == 2:
        return "Tijera"
    else:
        return "Gesto desconocido"

# Función para determinar el ganador
def determinar_ganador(jugador, computadora):
    if jugador == computadora:
        return "Empate"
    elif (jugador == "Piedra" and computadora == "Tijera") or \
         (jugador == "Papel" and computadora == "Piedra") or \
         (jugador == "Tijera" and computadora == "Papel"):
        return "¡Ganaste!"
    else:
```

```
        return "Perdiste"

# Lista de opciones
opciones = ["Piedra", "Papel", "Tijera"]

# Temporizador
contador = 5
jugando = False
jugada_jugador = None
resultado = ""
tiempo_inicio = None

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    frame = cv2.flip(frame, 1)
    imagen_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    resultado_manos = hands.process(imagen_rgb)

    if resultado_manos.multi_hand_landmarks:
        for hand_landmarks in resultado_manos.multi_hand_landmarks:
            mp_drawing.draw_landmarks(frame, hand_landmarks,
mp_hands.HAND_CONNECTIONS)
```

```
        if not jugando:
            tiempo_inicio = time.time()
            jugando = True
            jugada_jugador = None
            resultado = ""

        # Después de 5 segundos, captura la jugada
        if jugando and time.time() - tiempo_inicio > contador and jugada_jugador is
None:
            jugada_jugador = detectar_jugada(hand_landmarks)
            jugada_computadora = random.choice(opciones)

            if jugada_jugador in opciones:
                resultado = determinar_ganador(jugada_jugador, jugada_computadora)
            else:
                resultado = "Gesto no válido"

    if jugando:
        tiempo_transcurrido = int(time.time() - tiempo_inicio)
        tiempo_restante = max(0, contador - tiempo_transcurrido)
        cv2.putText(frame, f"Tiempo: {tiempo_restante}", (10, 40),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

        if jugada_jugador:
            cv2.putText(frame, f"Tú: {jugada_jugador}", (10, 80),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 0), 2)
```

```
            cv2.putText(frame, f"PC: {jugada_computadora}", (10, 120),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2)
            cv2.putText(frame, f"Resultado: {resultado}", (10, 160),
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
            cv2.putText(frame, "Presiona ESPACIO para volver a jugar", (10, 200),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 1)

    cv2.imshow("Piedra, Papel o Tijera", frame)

    key = cv2.waitKey(1) & 0xFF
    if key == 27:  # ESC
        break
    elif key == 32:  # Espacio
        jugando = False
        jugada_jugador = None
        resultado = ""

cap.release()
cv2.destroyAllWindows()
```


Piedra, Papel o Tijera
Tiempo: 0
TÚ: Piedra
PC: Tijera
Resultado: ??Conoste!
Presiona ESPACIO para volver a jugar
return "Piedra"
elif sum(dedos) == 5:
return "Papel"
elif dedos[1] == 1 and dedos[2] == 1 and sum(dedos) == 2:
return "Tijera"
else:
return "Gesto desconocido"
# Función para determinar el ganador
def determinar_ganador(jugador, computadora):
Ln: 20 Col
26°C
Mayorm. nublado
Buscar
ESP
04:38 p. m.
16/06/2025