Piedra, Papel, Tijeras

Trabajo Final de la asignatura de Visión por Computador Curso 2023/2024

Link

El **codigo** y el **material** utilizado està disponibile en github y alos siguientes enlaces:

Codigo

GitHub - gionniz (https://github.com/gionniz/Computer-Vision/edit/main/Trabajo)

Doc MediaPipe

https://developers.google.com/mediapipe/solutions/vision/hand_landmarker (https://developers.google.com/mediapipe/solutions/vision/hand_landmarker)

https://developers.google.com/mediapipe (https://developers.google.com/mediapipe)

Autores

Trabajo realizado por:

- **Giovanni Sgambato** y **Agata Cavigioli**

Documento de memoria

Motivación/argumentación del trabajo

Objetivo de la propuesta

Descripción técnica del trabajo realizado

Fuentes y tecnologías utilizadas

Conclusiones y propuestas de ampliación

Indicación de herramientas/tecnologías con las que les hubiera gustado contar

Créditos materiales no originales del grupo

Motivación/argumentación del trabajo

La practica propone que se realize una implementacion que haga uso de algunas herramientas presentadas en clase de la asignatura de **Vision por Computador** (dec/enero 2024), las cuales nos permiten trabajar en tareas de **detección de manos y gestos** y utilizarlas pare crear un juego interactivo!

Objetivo de la propuesta

El principal objetivo de este proyecto es de realizar un proyecto que integre técnicas de Visión por Computador vista en clase, familiarizarse con la librería de **Mediapipe** para diseñar un sistema de detección de gestos manuales en tiempo real mediante una cámara web y reconocer posiciones específicas para realizar un simple **juego interacivo** de humano contra ordenador.

Descripción técnica del trabajo realizado

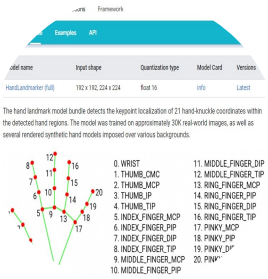
Por primero habrà que preparar un entorno python importando las librerías necesaria y iniciar la webcam (como visto en proyectos anteriores). El siguiente paso consiste en utilizar la librería **Mediapipe** para la "captura" de posiciones de manos y dedos.

```
    drawingModule = mediapipe.solutions.drawing_utils    #modulo dibujo
    hands_module = mediapipe.solutions.hands             #modulo deteccion manos
```

El módulo hands de mediapipe realiza la detección de las manos, mientras que el módulo de dibujo se utiliza para mostrar reultados en frame. (dibuja esqueleto y keypoints de la mano). Luego hay que crear una instancia de la clase y proporcionar algunos parametros (como valores de confianza mínima de detección y confianza mínima de seguimiento).

```
with hands_module.Hands(static_image_mode=False,            #instancia de la clase hands
        min_detection_confidence=0.7, min_tracking_confidence=0.4, max_num_hands=2) as hands:       #parameters de hands
```

A continuación se utiliza el frame capturado por la webcam pasando el frame al proceso hands, esto devolverá un **objeto de resultados** a trave del modulo "multi hand landmarks" que contiene todos los detalles de las manos detectadas en la imagen. Los puntos de referencia de la mano, es decir los **"landmarks"**, corresponden a la localización de los 21 puntos llaves, como se ve en la imagen de documemntacion. Ejecútandolo a través del módulo de dibujo se obtiene una imagen de los landmark y un esqueleto de la pose de la mano detectado.



Ahora Mediapipe nos está dando toda esta información sobre las posiciones xy de cada uno de los puntos, que podemos utilizar para reconocer **tres configuraciones**:

- **piedra** = en este caso, los cinco dedos están cerrados.
- **papel** = los cinco dedos están abiertos.
- **tijeras** = los dedos índice y corazón están abiertos y los demás cerrados.

Básicamente, lo que tenemos que hacer para este ejercicio es un control de estado de cada uno de nuestros dedos.
Por ejemplo mirando el índice podemos ver que Mediapipe nos da tres **key-points** a lo largo del dedo: **6, 7, 8**.
La idea que podemos utilizar es que cuando el dedo está abierto, el key-point en la punta del dedo va a estar por encima del punto en la base del dedo, y cuando el dedo está cerrado tendrà que estar por debajo del punto de la base.

Esto funciona perfectamente para los dedos índice, anular medio y meñique pero no funcionará para pulgares por la razon que se doblan hacia los lados.
Aún así, se puede comprobar si el key-point de la punta está más hacia la derecha en comparación con el punto de la base, entonces tenemos un pulgar abierto y en los demás casos estára cerrado.
Ahora es posible **mapear** las poiciones en un array, asignando 1 por cada dedo abierto y 0 por cerrado, y comprobar los estados esperados de piedra, papel y tijeras.

```
for finger in fingers:
    if finger == 'THUMB':
        thumb_status = get_thumb_status(hands_module, hand_landmarks)
        current_state += "1" if thumb_status else "0"
    else:
        finger_status = get_finger_status(hands_module, hand_landmarks, finger)
        current_state += "1" if finger_status else "0"
move = {"00000": "Piedra", "11111": "Papel", "01100": "Tijeras"}.get(current_state, "UNKNOWN")
```

Planear el juego

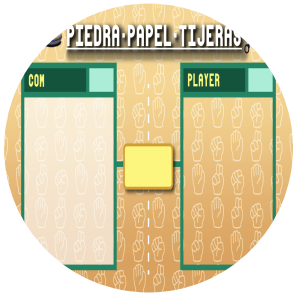
A este punto hay que "empaquetar" todo el sistema como un juego, definendo su **reglas** y **dinamicas**:

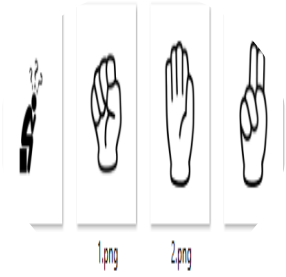
- pulsando un **botón de inicio** (la tecla barra de epacio) empieza un **round** de juego y un **timer** que permita al jugador saber cuándo debe hacer su movimiento.
- una vez que la cuenta atrás llegue a cero, vamos a establecer una **"flag"** booleana **"hold for play"** al valor true, se està asì realizando un **temporizador** que se pone en espera de detectar una jugada del "humano"
- es entonces cuando tenemos que **detectar el movimiento** del usuario, piedra, papel o tijera.
- despues habrà que crear un **movimiento aleatorio** del ordernador
- el último paso es **validar quién ganó** realmente la partida (se ha utilizado una funcion y un diccionario en lugar de ecribir varia condiciones if). Si la jugada del jugador y la del ordenador coinciden, habrà un empate.
- finalmente el programa se pone en **espera** a que empiece la siguiente partida

```
def calculate_game_state(move):
    moves = ["Piedra", "Papel", "Tijeras"]
    wins = {"Piedra": "Tijeras", "Papel": "Piedra", "Tijeras": "Papel"}
    selected = random.randint(0, 2)
    print("Computer played " + moves[selected])
    if moves[selected] == move:
        return 0, moves[selected], selected+1
    if wins[move] == moves[selected]:
        return 1, moves[selected], selected+1
    return -1, moves[selected], selected+1
```

Interfaz

Se ha creado una interfaz de usuario (UI) baica para ofrecer contenidos gráficos y istrucciones tetuales.





Fuentes y tecnologías utilizadas

- Conda (https://docs.conda.io/en/latest/)
- Mediapipe (https://mediapipe.dev/)
- OpenCV (https://opencv.org/)
- Guiones de las prácticas de la asignatura (https://github.com/otsedom/otsedom.github.io/tree/main/VC)

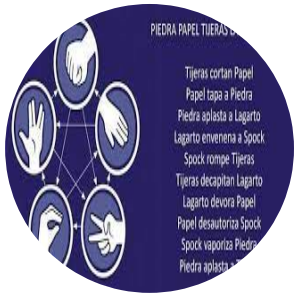
Conclusiones

Mediapipe es una herramienta muy potente para la detteccion de las manos y el reconocimiento de gestos con la cual es posible realizar una variedad de aplicaciones.

Propuesta de ampliación y posibles mejoras

Se han considerado de interés para un futuro desarrollo de la aplicacion las siguientes ampliaciones y mejoras:

- Mejorar la interfaz de usuario
- implementar un juego ma complejo como la version: **"piedra papel tijera lagarto spock"**



Créditos materiales no originales del grupo

Otras referencias y utilidades:

https://pypi.org/project/cvzone/ (https://pypi.org/project/cvzone/)

https://docs.opencv.org/4.x/ (https://docs.opencv.org/4.x/)

https://www.youtube.com/watch?v=ExuE0Pnqzac (https://www.youtube.com/watch?v=ExuE0Pnqzac)

https://www.youtube.com/watch?v=k2EahPgl0ho (https://www.youtube.com/watch?v=k2EahPgl0ho)

https://github.com/ramdas-codes/fncoder/tree/main/Python/Intermediate/RockPaperScissors (https://github.com/ramdas-codes/fncoder/tree/main/Python/Intermediate/RockPaperScissors)