

Piedra, Papel, Tijeras

Deteccion de manos aplicada a un juego interactivo



Objetivos

- Realizar un proyecto que integre **tecnicas 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 cámara web
- Reconocer **posiciones específicas** para realizar un simple **juego interactivo** de humano contra ordenador.

Descripción técnica del trabajo

El proyecto se compone de 3 partes distintas:

- Implementación de **MediaPipe** para la detección de manos y gestos
- **Planeamiento del juego** y de sus reglas y dinámicas
- **Interfaz gráfica** y componentes visuales

MediaPipe

- Por primero habrá que preparar un **entorno Python** importando las librerías necesaria y iniciar la **webcam**
- El siguiente paso consiste en utilizar la librería **Mediapipe** para la "captura" de posiciones de manos y dedos.
- El módulo **hands** realiza la detección de las manos, mientras que el módulo de **dibujo** se utiliza para mostrar resultados en el frame.

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



```
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
```



MediaPipe

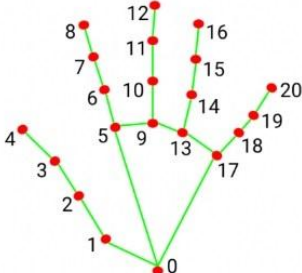
- Pasando el frame al proceso **hands**, esto devolverá un **objeto de resultados** que contiene todos los detalles de las manos detectadas
- Los puntos de referencia de la mano, es decir los "**landmarks**", corresponden a la localización de los **21 puntos llaves**

MediaPipe Solutions Framework

Overview Guide Examples API

Model name	Input shape	Quantization type	Model Card	Versions
HandLandmarker (full)	192 x 192, 224 x 224	float 16	info	Latest

The hand landmark model bundle detects the keypoint localization of 21 hand-knuckle coordinates within the detected hand regions. The model was trained on approximately 30K real-world images, as well as several rendered synthetic hand models imposed over various backgrounds.



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

MediaPipe

Ahora tenemos toda la 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.

```
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")
```



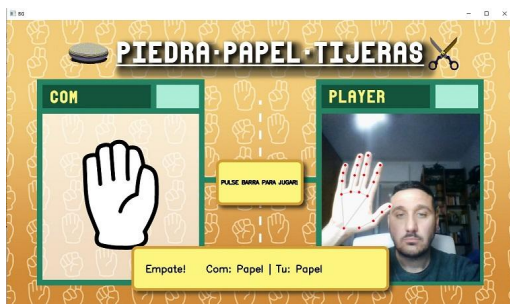
Planeamiento del juego

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

- Pulsando un **botón de inicio** (la tecla barra de espacio) 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 está así 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.
- Después habrá que **crear un movimiento aleatorio del ordenador**

Planeamiento del juego

- El último paso es **validar quién ganó** realmente la partida (se ha utilizado una funcion y un diccionario en lugar de escribir varias condiciones if).
Si la jugada del jugador y la del ordenador coinciden, habrá un **empate**.
- Se actualiza el **resultado** y la **puntuación** de los jugadores
- Finalmente el programa se pone en **espera** a que empiece la **siguiente partida**



Interfaz Grafica

- Se ha creado una **interfaz de usuario (UI)** basica para ofrecer **contenidos gráficos** y **instrucciones textuales**
- Las **imágenes** son **propias** y fueron creadas con **Adobe Illustrator**, y guardadas en formato **.png** con canal alpha para utilizar transparencias



Interfaz Grafica



Conclusiones

- Mediapipe es una herramienta muy potente para la detección de las manos y el reconocimiento de gestos con la cual es posible realizar una variedad de aplicaciones.
- **Propuesta de ampliación** y posibles mejoras:
 - Mejorar la interfaz de usuario
 - implementar un juego para 2 players humanos o mas complejo como la versión: **"piedra papel tijera lagarto spock"**



Link e Recursos

El código y el material utilizado está disponible en github y a los siguientes enlaces:

- **Código**

[GitHub - gionniz](#)

- **Doc MediaPipe**

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

<https://developers.google.com/mediapipe>

- **Otras referencias y utilidades:**

<https://pypi.org/project/cvzone/>

<https://docs.opencv.org/4.x/>

<https://www.youtube.com/watch?v=ExuE0Pngzac>

<https://www.youtube.com/watch?v=k2EahPgl0ho>

<https://github.com/ramdas-codes/fncoder/tree/main/Python/Intermediate/RockPaperScissors>