MEMORIA
TRABAJO FINAL

VISIÓN POR COMPUTADOR

Andrés Felipe Vargas Cortés
Miguel Ángel Peñate Alemán

Índice

Contenido Página

Introducción

Motivación y Descripción de la Propuesta

El objetivo principal de este proyecto es desarrollar un videojuego doméstico orientado a la simulación de coreografías de baile. La dinámica del juego consistirá en mostrar una coreografía en pantalla, mientras el jugador, utilizando una cámara, ejecuta los movimientos correspondientes. Los movimientos realizados por el jugador serán capturados en tiempo real, analizados y comparados con los pasos de baile presentados en el video.

Durante este proceso de comparación, el sistema asignará una puntuación basada en la precisión y fidelidad con las que el jugador reproduzca la coreografía. Al finalizar la interpretación, se presentará una puntuación acumulada que reflejará el desempeño global del jugador durante la sesión de baile.

Entre los objetivos específicos del proyecto se encuentra el desarrollo de un sistema de puntuación capaz de evaluar los movimientos del jugador y asignar una calificación proporcional al nivel de precisión alcanzado. Este sistema considerará diversos factores, tales como la correspondencia de los movimientos con la coreografía mostrada, el tiempo de reacción, la velocidad de ejecución y la coordinación motriz.

Asimismo, se busca diseñar un producto eficiente y completamente funcional que permita la interacción en tiempo real, garantizando un rendimiento fluido y sin interrupciones, incluso bajo la carga computacional asociada a la captura y procesamiento de los movimientos del jugador.

La propuesta surge de la motivación de aplicar los conocimientos adquiridos durante el curso en el desarrollo de una idea que ofrezca una experiencia interactiva y evaluativa. Es común que las personas intenten imitar las coreografías de las canciones más populares utilizando videos disponibles en plataformas como YouTube o redes sociales, debido a la facilidad y comodidad de acceso a estos recursos.

Sin embargo, la mayoría de los usuarios no tienen acceso a un sistema de puntuación avanzado que permita capturar y evaluar sus movimientos de manera precisa. Por esta razón, nuestro equipo ha decidido trabajar en esta idea con el objetivo de ofrecer una opción más completa y funcional. Aunque el proyecto presenta cierta complejidad, estamos convencidos de que su desarrollo valdrá la pena al proporcionar una experiencia novedosa y enriquecedora para los usuarios.

La idea ha sido inspirada por el famoso videojuego “Just Dance!”, al cual se hace referencia en el proyecto y del que se han extraído algunos videos para el desarrollo, de forma que se hace referencia a una versión gratuita de este, para aquellos usuarios que deseen optar por este tipo de opciones con intenciones de probar la experiencia de un software de este estilo.

Objetivos del Proyecto

En el apartado anterior se ha descrito el objetivo e ideas principales acerca de la propuesta del proyecto, no obstante, aún quedan muchos detalles para definir una idea que sigue siendo algo abstracta, a continuación, se detalla la modalidad de trabajo aplicada, así como los objetivos que se han propuesto para la implementación del producto.

Se ha seguido una metodología donde a partir de pequeñas iteraciones se han integrado diferentes funcionalidades y detalles al producto de software, para ello se ha partido de diferentes casos de uso que se han planteado.

El enfoque de los casos de uso se centra en la experiencia que podría tener el usuario a través de las necesidades expresadas que han sido planteadas por el equipo, en las cuales se han definido aquellos aspectos con los que debería contar el producto.

Los casos de uso definidos a continuación cuentan con varios campos que son:

- ID: Identificador del caso de usuario.
- Nombre: Nombre del caso de uso de forma que pueda ser referenciado fácilmente.
- Descripción: Breve definición del caso de uso.
- Criterios de Validación: Flujo que debe cumplirse para dar por implementado el caso de uso.
- Estimación: Referencia al tiempo estimado para que el caso de uso sea implementado.
- Prioridad: Referencia a la relevancia estimada por el equipo, cuanto más alta sea más prioritaria será su implementación para el producto.

A continuación, se muestran los casos de uso que han sido implementados por el equipo, el resto de los casos de uso que se han propuesto se consideran en el apartado de Conclusiones y propuestas de amplificación.

ID: 1	Nombre: Captura de movimientos
Descripción: Se debe recopilar a través de la cámara los movimientos realizados por el usuario.	
Criterios de validación: Una vez se inicie el programa se debe activar correctamente la cámara, con la cual se deben capturar los movimientos del usuario sin retrasos.	
Estimación: 1h	Prioridad: Alta

ID: 2	Nombre: Video de Coreografía
Descripción: Se mostrará al usuario un video con la coreografía a replicar.	
Criterios de validación: Cuando el usuario inicie una coreografía se debe mostrar por pantalla el video correspondiente, con la música y los pasos de baile.	
Estimación: 3h	Prioridad: Alta

ID: 3	Nombre: Analizar movimientos capturados
Descripción: Los movimientos capturados deben ser analizados para posteriormente identificar similitudes con los pasos de baile del video.	
Criterios de validación: El sistema detecta correctamente las posiciones y movimientos clave, los datos son recogidos para su análisis.	
Estimación: 4 h	Prioridad: Alta

ID: 4	Nombre: Comparar movimientos con coreografía
Descripción: El sistema debe comparar los movimientos capturados con los pasos mostrados en pantalla.	
Criterios de validación: Las comparaciones se realizan en tiempo real, identificando las diferencias significativas entre los movimientos.	
Estimación: 5 h	Prioridad: Alta

ID: 6	Nombre: Asignar Puntuación
Descripción: El sistema debe asignar una puntuación basada en la precisión de los movimientos del jugador.	
Criterios de validación: La puntuación refleja la similitud entre los movimientos del jugador y la coreografía, los resultados serán visibles al usuario en tiempo real.	
Estimación: 3h	Prioridad: Alta

ID: 7	Nombre: Subida de Coreografías Externas
Descripción: Una opción que permita al usuario incluir algún video externo a la aplicación.	
Criterios de validación: El usuario tendrá la capacidad de subir un video propio al programa, de forma que pueda ser utilizado para imitar la coreografía que en él se incluya.	
Estimación: 3h	Prioridad: Media

ID: 9	Nombre: Procesamiento de Coreografías Externas
Descripción: Se procesarán los videos subidos por el usuario, obteniendo los puntos y visibilidad para convertirlo en una nueva coreografía.	
Criterios de validación: A través de los videos subidos por el usuario se implementa una funcionalidad para extraer los movimientos del baile, mediante los distintos puntos y posición.	
Estimación: 4 h	Prioridad: Media

ID: 10	Nombre: Substracción de Audio de Videos
Descripción: Se debe obtener el audio de los videos subidos por el usuario para reproducirlos junto con el video.	
Criterios de validación: El programa debe incluir una forma de acceder al audio de los videos incluidos por el usuario para posteriormente reproducirlo junto a la coreografía.	
Estimación: 2h	Prioridad: Media

Implementación

Descripción Técnica del Trabajo

En este apartado se realizará un desglosamiento del trabajo realizado, este se ha basado en el desarrollo de los distintos objetivos mostrados anteriormente, de modo que se revisará el proceso de implementación haciendo hincapié en aquellos puntos relacionados con la visión por computador, aunque primeramente se introducirán los elementos que tienen lugar dentro del proyecto.

Según se pudo ver en el apartado de la Introducción la captura del movimiento del usuario resulta muy importante, es por ello por lo que se optó por el uso de Mediapipe, una librería que permite el seguimiento en tiempo real de manos, rostro y cuerpo, que resulta idóneo para nuestro planteamiento.

- Módulos y Bibliotecas Importados.

Se adjuntan los módulos y bibliotecas utilizados, así como una breve descripción de su uso:

- **OpenCV (cv2)**: Manejo de video, imágenes y superposición de gráficos.
- **pygame**: Manejo de audio, interfaz gráfica y eventos del juego.
- **MediaPipe (mp)**: Detección de poses corporales y manos.
- **NumPy (np)**: Operaciones matemáticas y manejo de arreglos.
- **json**: Lectura y escritura de datos estructurados.
- **PIL (Pillow)**: Manejo de imágenes y GIFs.
- **MoviePy**: Extracción de audio y manejo de videos.
- **Queue y Threading**: Procesamiento concurrente para manejar video en tiempo real.
- **time**: Control de tiempos en procesos interactivos.

- Variables Globales y Configuración.

Entre las variables globalizadas, se pueden encontrar los módulos de Mediapipe, variables que referencian al destino del video utilizado para la coreografía, también se utilizan algunos elementos gráficos como botonos, logotipos o pantallas que son accedidos de forma local, algunos son Gifs que superponen en tiempo real.

En cuanto a los parámetros de juego, se controla mediante una variable de modo, que especifica el funcionamiento según su valor, además de otros parámetros configurables como la elección de bailar con la detección de esqueleto activa o desactivada o para el control de tiempo de espera, así como control de los nodos entre otras.

El proyecto se ha unificado en una única celda, de forma que el usuario solo tenga que ejecutar el proyecto con un click de ratón, su funcionamiento se divide en dos opciones.

- Extraer información del video.

En este proceso se recupera el video que el usuario ha proporcionado, a partir del cual se obtiene el audio, así como los distintos puntos de referencia del esqueleto que ha captado Mediapipe, mediante su módulo Pose diseñado para la detección y captura de poses humanas, además de funciones de OpenCV.

- Empezar con el baile.

En este proceso se suceden varios eventos, entre ellos se abre la cámara del usuario mediante la cual se buscan personas usando Mediapipe para capturar las poses detectando las posiciones del cuerpo a través de varios puntos de referencias, finalmente estos puntos son comparados frame por frame con los puntos extraídos del vídeo, de esta comparación surge la puntuación de usuario, en función de esta se muestran animaciones al usuario.

De forma simplificada las anteriores descripciones representan el funcionamiento normal del proyecto, a continuación, se detallará con un poco más de detalle el flujo que siguen las funciones.

- Descripción de las Funciones desarrolladas.

Se nombran y comentan brevemente aquellas funciones que han sido desarrolladas, con motivo de documentar su funcionamiento y desempeño de forma simple:

❖ **Overlay de imágenes (overlay_image()):**
  - Superpone imágenes en un frame respetando transparencia y tamaño.
  - Soporta redimensionamiento y uso de canal alfa.

❖ **Gestión de GIFs (get_next_gif_frame(), change_gif()):**
  - Extraen y procesan frames de un GIF para mostrar animaciones en tiempo real según las acciones del jugador.

❖ **Cálculo de similitud entre poses:**
  - **calcular_distancia_escala():** Calcula la distancia entre los hombros para escalar los datos.
  - **calcular_similitud():** Compara puntos clave escalados del video de referencia con los del jugador para determinar similitudes.

❖ **Evaluación del desempeño (puntuación ()):**
  - Obtiene puntos de referencia del JSON, calcula la similitud con los movimientos del jugador y asigna un puntaje (Perfecto, Bueno, Fallido).

- ❖ **Procesamiento del video de referencia (process_video())**:
  - Renderiza el video de referencia con posibles overlays (esqueleto, GIFs) y sincroniza con la música.

- ❖ **Procesamiento de la cámara (process_camera())**:
  - Detecta poses o manos del jugador en tiempo real.
  - Detecta interacción con botones de juego (como "Play" o "Toggle Skeleton").
  - Calcula puntajes basados en la similitud con el video de referencia.

- ❖ **Gestión del flujo del juego (jugar_con_puntos())**:
  - Coordina la visualización del video y la cámara.
  - Maneja eventos del juego, como cambio de modos (inicio, juego, puntuación).

- ❖ **Extracción de puntos de referencia (extraer_y_guardar_puntos())**:
  - Procesa el video de referencia para detectar poses en cada frame.
  - Guarda los puntos clave (landmarks) en un archivo JSON para usarlos en la comparación con el jugador.

- Uso de hilos concurrentes.

El programa se encarga de ejecutar hilos que desempeñan funciones diferentes de forma concurrente, se trata de:

- ❖ process_video: Renderiza el video de referencia, procesa sus puntos y calcula similitudes.

- ❖ process_camera: Procesa la cámara en tiempo real, detecta poses o manos, y maneja interacciones con botones.

La variable Stop_threads, se usa como bandera que permite detener de forma segura los hilos que se ejecutan en paralelo dentro del programa.

Todos estos objetos derivan de Threating, que ofrece el objeto Thread, uno de los recursos importados.

- Desglose de las funciones principales.

En primera instancia se va a analizar la función de extracción de puntos de referencia y audio del programa, llamada *extraer_y_guardar_puntos().*

Parámetros de entrada:

- *Video_path:* Entrada del destino del video subido por el usuario, este video será procesado para extraer tanto el audio, como las poses detectadas en el video.

- *Output_file:* Nombre del fichero JSON que guardará todos aquellos puntos que se hayan detectado, incluyendo la posición en 3 dimensiones (X, Y, Z).

Ilustración de la función:

```python
def extraer_y_guardar_puntos(video_path, output_file):
    puntos_video_referencia = []
    # Cargar el video y extraer el audio
    video_clip = VideoFileClip(video_path)
    video_clip.audio.write_audiofile(audio_output_path)

    print("Audio extraído y guardado en:", audio_output_path)

    with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:
        video_cap = cv2.VideoCapture(video_path)
        while True:
            ret_vid, frame_vid = video_cap.read()
            if not ret_vid:
                break

            # Obtén el índice del frame actual
            frame_index = int(video_cap.get(cv2.CAP_PROP_POS_FRAMES))

            rgb_frame_vid = cv2.cvtColor(frame_vid, cv2.COLOR_BGR2RGB)
            results_vid = pose.process(rgb_frame_vid)

            if results_vid.pose_landmarks:
                puntos = []
                for landmark in results_vid.pose_landmarks.landmark:
                    puntos.append({
                        "x": landmark.x,
                        "y": landmark.y,
                        "z": landmark.z,
                        "visibility": landmark.visibility
                    })

                # Agrega los puntos junto con el índice del frame al resultado
                puntos_video_referencia.append({
                    "frame": frame_index,
                    "puntos": puntos
                })

        video_cap.release()

    # Guarda los puntos en un archivo JSON
    with open(output_file, 'w') as f:
        json.dump(puntos_video_referencia, f)

    print(f"Puntos de referencia guardados en {output_file}")
```

Ilustración 1. Función de Extracción de Información

Descripción del proceso de la función:

La función comienza extrayendo el audio y exportándolo a un fichero cuyo destino se encuentra en la variable *audio_output_path.*

Cuando termina de escribirse en este archivo, comienza el proceso de extracción de puntos de referencia, para ello se utiliza el módulo Pose de Mediapipe, ideal para detectar y obtener poses humanas por parte del usuario, a través de funciones de OpenCV se consiguen los puntos de referencia de las poses detectadas usando los píxeles de estas posiciones, finalmente se guardan los datos frame a frame,
(incluyendo la posición en los ejes xyz) en el archivo de salida indicado en los parámetros de la función.

Resultados de la función:

Para cuando finaliza la función se obtienen los datos de audio, así como los puntos de referencia los cuales tienen la siguiente forma:

```
[{"frame": 86, "puntos": [{"x": 0.5352080464363098, "y":
0.35021165013313293, "z": -0.274441659450531, "visibility":
0.9981062412261963}...]
```

Por cada frame, se recupera la posición de todos los puntos detectados, donde para cada punto se estima su posición y su visibilidad, estos deben haber superado un cierto rango de confianza por parte de Mediapipe (Del 50%).

A continuación, se prosigue con la función de *jugar_con_puntos()*, la cual lleva a cabo el proceso de inicializar el juego, así como preparar las colas para los hilos concurrentes que iniciarán el video y la cámara del usuario.

Parámetros de entrada:

- *Json_file:* Se trata del nombre del fichero del que se van a extraer los datos de los puntos de referencias capturados en la función de extracción comentada anteriormente.

- *dance_video*: Dirección en la que se encuentra el video subido por el usuario, el cual será reproducido por los hilos de la función.

- *Ilustraciones de la función*:

```python
def jugar_con_puntos(json_file, video_path):
    global stop_threads

    with open(json_file, 'r') as f:
        puntos_video_referencia = json.load(f)

    pygame.init()
    pygame.mixer.music.load(audio_output_path)

    cap_video = cv2.VideoCapture(video_path)
    cap_camera = cv2.VideoCapture(0)

    if not cap_video.isOpened() or not cap_camera.isOpened():
        print("Error: No se pudo abrir el video o la cámara.")
        return

    frame_width = 640  # Reducir a la mitad
    frame_height = 360  # Reducir a la mitad
    video_frame_rate = cap_video.get(cv2.CAP_PROP_FPS)

    screen_width = frame_width * 2  # Doblar para mostrar video + cámara
    screen_height = frame_height
    screen = pygame.display.set_mode((screen_width, screen_height))  # Usar dimensiones reducidas

    video_queue = Queue()
    camera_queue = Queue()

    video_thread = Thread(target=process_video,
    args=(cap_video, video_queue, video_frame_rate, frame_width, frame_height, puntos_video_referencia))
    camera_thread = Thread(target=process_camera,
    args=(cap_camera, camera_queue, puntos_video_referencia, video_frame_rate, frame_width, frame_height, cap_video))

    video_thread.start()
    camera_thread.start()
```

```python
while video_thread.is_alive() or camera_thread.is_alive():
    if not video_queue.empty():
        frame_rgb = video_queue.get()
        current_image = pygame.image.frombuffer
        (frame_rgb.tobytes(), frame_rgb.shape[1::-1], "RGB")
        screen.blit(current_image, (0, 0))

    if not camera_queue.empty():
        frame_camera_rgb = camera_queue.get()
        frame_camera_surface = pygame.image.frombuffer
        (frame_camera_rgb.tobytes(), frame_camera_rgb.shape[1::-1], "RGB")
        screen.blit(frame_camera_surface, (frame_width, 0))

    pygame.display.flip()

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            stop_threads = True
            video_thread.join()
            camera_thread.join()
            cap_video.release()
            cap_camera.release()
            pygame.quit()
            return

stop_threads = True
video_thread.join()
camera_thread.join()
cap_video.release()
cap_camera.release()
pygame.quit()
```

Ilustración 2. Función de Procesamiento de Hilos concurrentes

- *Descripción del proceso de la función.*

En esta función se realizan un conjunto diferente de acciones, empezando por abrir el fichero de puntos de referencia para su lectura, también se inicializa el programa mediante la librería de Pygame y se preparán las variables para el control del video y de la cámara, que se abren a continuación.

Después de configurar las dimensiones de la ventana, se inician los hilos y colas del video y de la cámara, a continuación, se entra en un bucle que dependerá de ambos hilos, cada uno tendrá una función propia y se ejecutarán de forma simultánea.

Video_thread: Muestra en un formato manejable por Pygame el video subido por el usuario frame a frame.

Camera_thread: Captura y muestra por pantalla la información recopilada por la cámara del usuario frame por frame.

- *Resultados de la función.*

Con esta función se obtiene un proceso continuo que muestra al usuario la coreografía captada en el vídeo subido, así como su propia interpretación captada por cámara.

También se va a describir la función de asignación de puntuaciones, llamada *puntuacion()* que se encarga de determinar mediante la comparación de las poses detectadas en el video y la cámara la similitud existente y devolver una puntuación al usuario.

Parámetros de entrada:

- *Json_file:* Nombre del fichero con puntos de referencia detectados en el video subido por el usuario, al cual se accede.

- *Frame_buscado*: Hace referencia al índice del frame del cual se está tratando de obtener la comparación.

- *Results_cam*: Se trata de la lista de los frames capturados por la cámara del usuario.

- *Ilustración de la función*:

```python
def puntuacion(json_file, frame_buscado, results_cam):
    puntos_referencia = obtener_puntos_por_frame(json_file, frame_buscado)
    if puntos_referencia:
        puntos_usuario = [
            {"x": lm.x, "y": lm.y, "z": lm.z, "visibility": lm.visibility}
            for lm in results_cam.pose_landmarks.landmark
        ]

        escala_referencia = calcular_distancia_escala(puntos_referencia)
        escala_usuario = calcular_distancia_escala(puntos_usuario)

        similitud = calcular_similitud(puntos_referencia, puntos_usuario, escala_referencia, escala_usuario)

        if similitud < 5:
            change_gif(100)
            return 100
        elif similitud < 10:
            change_gif(50)
            return 50
        else:
            change_gif(0)
            return 0
    return 0
```

Ilustración 3. Función de Asignación de Puntuación

- *Descripción del proceso de la función.*

A partir del fichero Json en el que se han guardado los puntos de referencia del video y el índice del frame con el que se está trabajando, se obtienen los puntos de referencia asociados al frame en concreto.

A continuación, se obtiene para el mismo frame los puntos de referencia de la cámara, finalmente se realiza un escalado entre los puntos de referencia de video y cámara del usuario.

Finalmente, se calcula una similitud a partir de la media de las divisiones entre los puntos de la cámara y los puntos del fichero Json escalados, cuanto más cercana sea la similitud mejor puntuación recibirá el usuario, significará que los puntos de referencia y los capturados por la cámara están a una distancia cercana a lo esperado.

- *Resultados de la función.*

Esta función ofrece un simple pero eficaz método de comparación de puntos, que devuelve una suma de puntos según la cercanía entre los puntos escalados.

Fuentes y Tecnologías Utilizadas

Para el desarrollo del proyecto se ha seguido el diseño de las prácticas realizadas durante el curso, el diseño está basado en notebooks de Python, en concreto Jupyter Notebooks.

El uso de notebooks de Python proporciona algunas ventajas como la interactividad, la facilidad de visualización y ejecución, con el uso de una estructura modular, uso de celdas, etc.

A continuación, se muestran las versiones de las tecnologías que han sido utilizadas para el desarrollo del proyecto.

Tecnologías Utilizadas	Versiones Utilizadas
Python	3.11.5
Anaconda	2024.6 o 2024.10
Librerías de Python	La Última Disponible

Luego de tener instaladas las versiones correctas de las utilidades necesarias para el funcionamiento del proyecto se recomienda el uso de un editor de código compatible con Jupyter Notebooks como puede ser Visual Code Studio (Anteriormente mencionado).

Desde Anaconda prompt será necesaria la creación de un nuevo Kernel (Núcleo) que permite la ejecución de los comandos de Python, para ello se pueden ejecutar los siguientes comandos desde la terminal.

Comando para la creación del entorno:

```
conda create --name VC_Proyecto python=3.11.5
```

Comando para la activación del entorno:

```
conda activate VC_Proyecto
```

Después de ejecutar correctamente las instrucciones anteriores es necesario instalar todas las dependencias necesarias, que se muestran a continuación:

```
pip install opencv-python
pip install pygame
pip install mediapipe
pip install numpy
pip install pillow
pip install moviepy
```

Cuando todas las librerías hayan sido instaladas será posible ejecutar el proyecto sin errores.

Entre las fuentes utilizadas, se encuentran los siguientes enlaces:

Github otsedom : https://github.com/otsedom/otsedom.github.io/tree/main/VC

Referencias a Prácticas anteriores: https://github.com/miguetech16/VC-Practicas

Módulo de Inteligencia artificial: https://chatgpt.com/

Las fuentes de cada uno de los módulos y librerías importados se anotan a continuación:

- **OpenCV**: https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html
- **Pygame**: https://www.pygame.org/docs/
- **Mediapipe**: https://ai.google.dev/edge/mediapipe/solutions/guide?hl=es-419
- **Numpy**: https://numpy.org/doc/
- **Json**: https://docs.python.org/3/library/json.html
- **Pillow**: https://pillow.readthedocs.io/en/stable/
- **MoviePy**: https://zulko.github.io/moviepy/
- **Queue**: https://docs.python.org/3/library/queue.html
- **Threading**: https://docs.python.org/3/library/threading.html
- **Time**: https://docs.python.org/3/library/time.html

Conclusiones

Resolución del Trabajo Realizado

En cuanto al trabajo realizado el equipo considera que el producto desarrollado cumple con los objetivos más prioritarios que se propusieron junto con la propuesta del trabajo, se trata de un producto que promete un uso funcional para el fin por el que fue implementado.

Sin embargo, el proyecto no representa el producto final y se podría mejorar en muchos aspectos, tal como la incorporación de varias funcionalidades necesarias, la mejora de las funciones descritas, procesos más depurados y una mejor interfaz de usuario.

La propuesta de trabajo abarca un alcance que podría extenderse sin aumentar considerablemente la dificultad y con un amplio rango de mejora, ciertamente se han usado recursos y se han implementado diversas técnicas que se han estudiado durante el transcurso de la asignatura, las cuales han ofrecido un alto rendimiento, por encima de lo esperado.

Algunas de las bases de la propuesta que consistían en desarrollar un producto funcional que resultará fluido para el usuario, las cuales han sido las principales iniciativas, para ello se hizo uso de hilos para implementar los procesos de forma concurrente y sin cortes, no obstante, es cierto que se trata de un proceso pesado y el uso del módulo Pose de Mediapipe hace uso de múltiples recursos del sistema.

Por lo que se hace necesario el uso de un sistema informático que cumpla con ciertos componentes y características mínimas para correr el programa sin sufrir interrupciones, cortes o ralentización en la ejecución.

En un principio el equipo consideró el uso de módulos entrenados para la detección de personas, de forma que a partir de estos se pudiera entrenar otro modelo para la detección de la pose de los sujetos, por ejemplo, alguna versión disponible de YOLO, utilizada en otras prácticas de la asignatura, sin embargo, esto quedó descartado después de estudiar las posibilidades ofrecidas por Mediapipe y su sencillez.

Para la mejora de la interfaz se podría hacer uso de la librería ofrecida por Tkinter, que ofrece un repertorio de posibilidades de integración para la interfaz de usuario, que mejoraría la experiencia, por lo que se usaría para la implementación de algunas de las propuestas de amplaición y mejora.

Por otro lado, el equipo ha sabido trabajar de forma responsable, independiente y organizada, ha superado las dificultades de las tareas y se ha podido llegar a un buen resultado en cuanto a tiempo y esfuerzo que superan las estimaciones no tan optimistas que se propusieron e implementaron, disponibles al principio del documento.

Propuestas de Ampliación

En este apartado se muestran algunas de las propuestas que no pudieron ser completadas, así como otras ideas que el equipo ha propuesto, pero se han descartado por falta de tiempo o recursos.

Es importante mencionar que la implementación de las siguientes propuestas de ampliación no implica la versión final del producto, ya que esté excede las capacidades del equipo.

ID: 5	Nombre: Multidetección de usuarios
Descripción: El programa esta provisto para permitir el uso correcto de más de un jugador.	
Criterios de validación: Podrá seleccionarse el número de usuarios concurrentes para la ejecución, de forma que más de un jugador pueda incorporarse a la detección de movimientos y obtención de puntuación.	
Estimación: 8 h	Prioridad: Alta

ID: 8	Nombre: Capacidad de varias Coreografías
Descripción: El usuario puede subir más de un video, para su extracción de audio y puntos.	
Criterios de validación: El programa admitirá una opción de subida de videos, de forma que el usuario puede subir más de un video, para la extracción de audio y puntos, que le permita ejecutar más de una única coreografía.	
Estimación: 5 h	Prioridad: Alta

ID: 11	Nombre: Pausa del Programa
Descripción: El usuario podrá pausar manualmente el proceso de ejecución del programa, de forma que esté quede paralizado.	
Criterios de validación: El programa contará con un icono de pausa, al hacer click en este icono se detendrá el proceso correspondiente, los recursos utilizados también deben detenerse y quedar a espera del usuario.	
Estimación: 3 h	Prioridad: Media

ID: 12	Nombre: Salida del Programa
Descripción: Se implementa una funcionalidad para detener la ejecución del programa manualmente y volver a la pantalla inicial.	
Criterios de validación: El programa contará con un icono de salida, que permitirá al usuario volver a la pantalla inicial y cerrar todos los recursos no necesarios que se estuvieran ejecutando.	
Estimación: 2 h	Prioridad: Media

ID: 13	Nombre: Resolución de Pantalla
Descripción: El usuario podrá configurar la resolución de pantalla a la que se va a adaptar el programa.	
Criterios de validación: Se podrá seleccionar la resolución de la ventana del programa, de forma que se ajuste a las dimensiones que se establezcan por el usuario.	
Estimación: 3 h	Prioridad: Media

ID: 14	Nombre: Guardar Repetición
Descripción: El usuario podrá guardar en su equipo un video de su interpretación.	
Criterios de validación: El usuario tendrá la opción después de terminar la coreografía de guardar una repetición del video capturado por su cámara, el cual debe incluir el audio del video.	
Estimación: 6 h	Prioridad: Baja

ID: 15	Nombre: Idiomas
Descripción: El usuario podrá seleccionar el idioma del programa.	
Criterios de validación: El usuario tendrá la opción de seleccionar el idioma con el que se ajusten los textos del programa, incluyendo al menos dos opciones, español e inglés.	
Estimación: 4 h	Prioridad: Baja

Organización del Equipo

El equipo ha organizado reuniones para la mejor organización y reparto de tareas del proyecto, estas reuniones se han realizado con una frecuencia de entre una a dos veces por semana, con horario cambiante, dependiendo de las necesidades de cada individuo.

Estas reuniones han sido útiles para poder compartir ideas sobre los objetivos que se debían alcanzar, proposición de ideas o puntos de vistas, así como la básica intención de conocer las tareas a desarrollar por cada individuo.

Aunque no se ha llevado a cabo ningún tipo de historial de reuniones, por lo general estás no han superado los 10 minutos consecutivos, sin embargo, en alguna de ellas se aprovechó para continuar con las tareas correspondientes a cada uno para el adelantamiento de las tareas, en las que si fue necesaria una coordinación por ambas partes.

En líneas generales no se ha considerado una mayor frecuencia de reuniones o del tiempo que requieren, ya que podría resultar ineficiente en algunos casos.

Detalles de Interés

Créditos materiales no originales

- Imágenes utilizadas.

Se incluyen imágenes extraídas de fuentes externas, en este caso imágenes accesibles desde Google Imágenes. Estas imágenes han sido utilizadas para ilustrar conceptos, ejemplos visuales o contenido gráfico específico.

Algunas imágenes han sido editadas para adaptarse al diseño o formato del proyecto, respetando los derechos de autor y las licencias de uso correspondientes.

- Módulos y librerías utilizadas.

El proyecto incorpora librerías y módulos de terceros para funciones específicas, que han sido mencionados anteriormente, estas herramientas han sido descargadas desde repositorios oficiales

- Videos de prueba utilizados.

Los videos utilizados para pruebas provienen de plataformas públicas como YouTube con permisos adecuados o archivos proporcionados por terceros con fines legítimos.

Recursos de Interés

Este apartado está destinado al acceso de información relevante mediante enlaces e imágenes, que están relacionadas con el proyecto.

- Enlace al código fuente:

Se puede acceder al código fuente del proyecto a través del siguiente enlace:

VC-Practicas/Trabajo_Final/TF_VC.ipynb at main · miguetech16/VC-Practicas

- Imagen de Portada del Trabajo:

Ilustración 4. Carátula del Proyecto

- Videos resumen del Trabajo:

Debido a que no es posible adjuntar videos a este tipo de documentos, todo el material audiovisual creado para representar el Proyecto se ha subido al repositorio de Github y queda disponible dentro del fichero de README.md, en el cual se muestran algunas imágenes o gifs, así como enlaces para acceder a los videos, por ello se proporciona a continuación el enlace a dicho repositorio:

VC-Practicas/Trabajo_Final at main · miguetech16/VC-Practicas