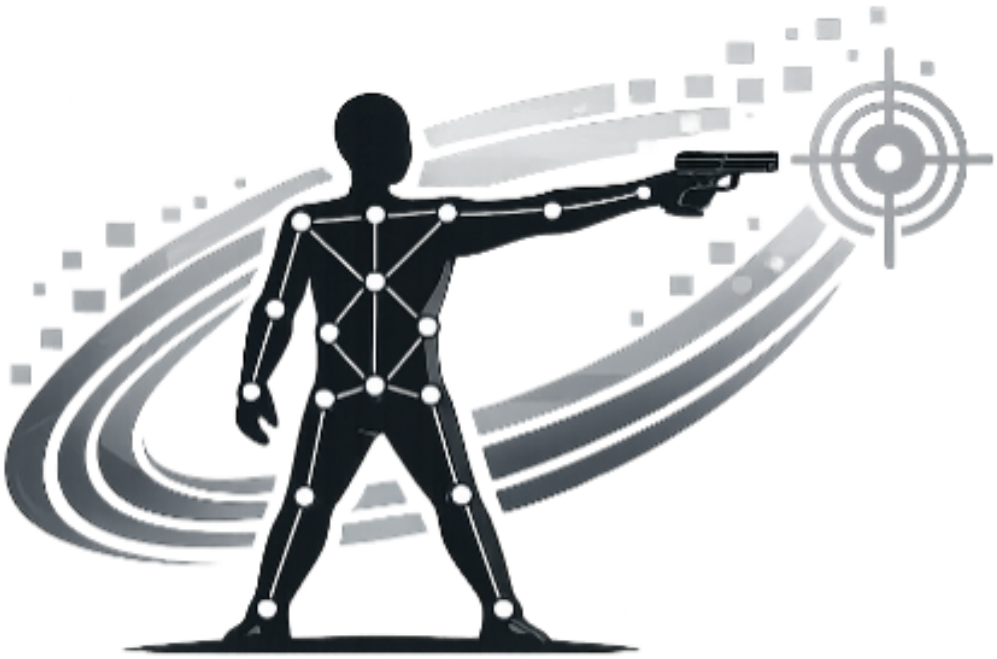
ULPGC Universidad de Las Palmas de Gran Canaria

ESCUELA DE INGENIERÍA INFORMÁTICA

Visión por Computador



Diseño e implementación de una interfaz natural de usuario para videojuegos FPS basada en visión por computador

Autores

Guillermo Raposo Iglesias

Tycho Santana Quintana

Ian Samuel Trujillo Gil

Índice general

Índice de figuras

Índice de tablas

1. Introducción

Este trabajo presenta un prototipo de controlador híbrido para un videojuego tipo FPS, combinando dos modalidades de interacción basadas en visión por computador: (1) seguimiento en tiempo real de un objeto físico que actúa como apuntador para controlar el ratón y el disparo, y (2) detección de gestos corporales mediante estimación de pose humana para controlar el movimiento (W/A/D) y el salto. El sistema se ejecuta en un único bucle de captura de cámara, incorpora un flujo de calibración guiado, aplica filtros para reducir ruido (EMA, antirrebote) y emplea estrategias de recuperación ante pérdidas de tracking (CSRT + filtro de Kalman + búsqueda por textura).

2. Motivación del trabajo

El predominio de los periféricos tradicionales, si bien garantiza precisión, restringe la exploración de nuevas formas de interacción. Se considera de gran interés investigar interfaces naturales que no solo favorezcan la accesibilidad, sino que también extiendan el ciclo de vida de los videojuegos al ofrecer paradigmas de uso inéditos y no explorados. Se plantea que la diversificación de los métodos de entrada tiene el potencial de revitalizar la experiencia de usuario, haciéndola más inmersiva y estimulante.

En este contexto, la presente propuesta desarrolla un sistema de control basado exclusivamente en visión por computador, prescindiendo de sensores dedicados. Para la validación experimental del prototipo se ha utilizado el videojuego Counter-Strike 2 de Valve. La solución técnica aborda la interacción mediante un enfoque híbrido: el seguimiento de un objeto físico para la precisión del apuntado y el reconocimiento de pose corporal para el desplazamiento.

Dado el carácter académico y experimental de este desarrollo, se recomienda restringir su empleo a entornos controlados, como modos de entrenamiento o servidores privados. Asimismo, se insta encarecidamente a revisar y respetar los Términos y Condiciones de Servicio de cualquier software o plataforma de terceros donde se pretenda utilizar esta tecnología, con el fin de evitar infracciones de las normativas de uso.

3. Objetivo de la propuesta

3.1. Objetivo general

Con este proyecto se pretende diseñar e implementar un controlador híbrido basado en visión por computador que permita apuntar y disparar mediante el seguimiento de un objeto físico, así como moverse y saltar a través de gestos corporales. Dicho sistema se integrará en una aplicación FPS comercial simulando las entradas de ratón y teclado convencionales.

3.2. Objetivos específicos

Para alcanzar la meta principal, se han definido los siguientes objetivos específicos:

- **Diseñar un flujo de calibración** sencillo que permita extraer los perfiles visuales del objeto apuntador en diferentes condiciones de luz.
- **Implementar un algoritmo de seguimiento** robusto y rápido, empleando predicción de trayectoria para manejar oclusiones y movimientos bruscos, así como mecanismos de recuperación que garanticen un funcionamiento totalmente autónomo tras la configuración.
- **Integrar la detección de pose humana** estableciendo una lógica de decisión basada en zonas neutrales y activas para la traducción de gestos a comandos de movimiento.
- **Desarrollar un módulo de inyección de eventos** (teclado y ratón) que permita el control de teclas mantenidas y pulsaciones instantáneas en el sistema operativo.
- **Validar la funcionalidad del prototipo** mediante pruebas de jugabilidad en un entorno real (Counter-Strike 2), evaluando la precisión y la respuesta del sistema.

4. Alcance, dificultad y originalidad

4.1. Alcance y desafíos del proyecto

El alcance del presente trabajo comprende la implementación integral de un sistema interactivo en tiempo real, abarcando desde la captura de vídeo y la calibración inicial, hasta la estimación de pose, el seguimiento visual, el filtrado de señal y la lógica de decisión para el control de periféricos.

La complejidad técnica reside fundamentalmente en garantizar la robustez del sistema frente a perturbaciones comunes en visión por computador —como ruido, oclusiones parciales, cambios de escala y variaciones de iluminación—, manteniendo simultáneamente una latencia mínima y una estabilidad suficiente para asegurar una experiencia de usuario fluida y funcional.

La innovación de la propuesta radica en una arquitectura híbrida que combina dos modalidades de entrada (seguimiento de objeto y pose corporal) y en una estrategia de seguimiento escalonada. Esta integra un tracker CSRT como motor principal y un filtro de Kalman para la predicción de trayectorias. Adicionalmente, se implementa un mecanismo de recuperación basado en búsqueda por textura (*template matching*) reforzado con un veto estricto en el espacio de color HSV, diseñado para reanudar el seguimiento tras una pérdida sin incurrir en falsos positivos.

5. Descripción técnica del trabajo realizado

El desarrollo está organizado en cuatro módulos, tal y como se refleja en el notebook entregado: módulo de inputs, módulo de detección de gestos (pose), módulo de seguimiento del objeto apuntador y módulo de ejecución principal.

5.1. Arquitectura del sistema

En la siguiente Figura se resume la arquitectura: una única cámara alimenta dos pipelines (pose y tracking del objeto). Cada pipeline genera acciones que se traducen a entradas del sistema (ratón/teclado) y se envían al juego. El envío se puede limitar a cuando la ventana del juego está en primer plano.

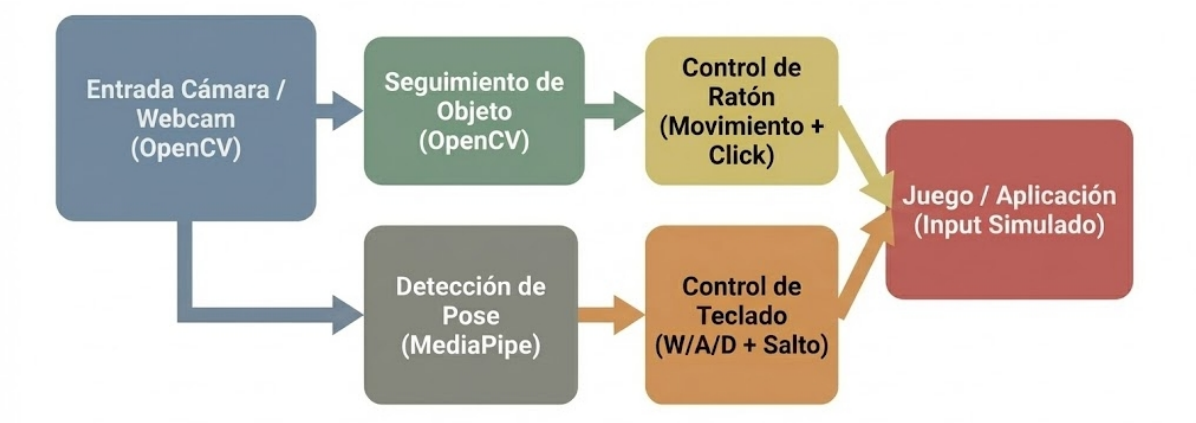


Figura 5.1: Diagrama de la arquitectura del prototipo.

5.2. Módulo de inputs (teclado y ratón)

El módulo de inputs encapsula la interacción con teclado y ratón. Para el teclado se mantienen pulsaciones de teclas de movimiento (W/A/D) y se emite la tecla de salto (Espacio). Para el ratón se implementa una factoría que selecciona la estrategia adecuada según el sistema operativo: en Linux se apoya en xdotool y en Windows en pydirectinput. Se incluye una rutina de liberación segura de teclas al finalizar la ejecución. Asignación por defecto:

Tabla 5.1: Asignación de controles por defecto

Entrada/Gesto	Acción en el juego
W	Avanzar
S	Retroceder
A / D	Desplazamiento lateral (Izquierda/Derecha)
Espacio	Saltar
Click Izq.	Disparar

5.3. Módulo de detección de gestos por pose

La detección de gestos se basa en MediaPipe Pose Landmarker (modelo 'lite') ejecutado en modo VIDEO. A partir de los landmarks del cuerpo, se extraen features geométricas (inclinación lateral, altura relativa de cadera, diferencias verticales de hombros, etc.) que alimentan una lógica de decisión. Para mejorar estabilidad se aplica un suavizado exponencial (EMA) y un antirrebote (debouncer) que exige varios frames consecutivos antes de cambiar de estado.

Las especificaciónes de la calibración del módulo de poses son las siguientes:

- **Espera previa:** 5.0 s (tiempo para colocarse).
- **Captura de postura neutral:** 2.0 s.
- **Captura de postura 'forward':** 1.2 s (referencia para caminar).

Tabla 5.2: Parámetros de configuración del módulo de pose

Componente	Parámetro	Valor
Lateral	Umbral inclinación (`LEAN_X_THRESH`)	0.25
Suavizado	EMA altura cadera (`EMA_Y`)	0.30
	EMA dif. vertical (`EMA_DY`)	0.35
Salto	Umbral elevación (`JUMP_UP_HIP`)	0.10
	Umbral velocidad (`JUMP_VEL`)	0.06
	Cooldown (`JUMP_COOLDOWN_SEC`)	0.55
Forward	Mínimo entrada dy (`DY_ENTER_MIN`)	0.030
	Histéresis salida (`DY_EXIT_RATIO`)	0.65

Respecto a la lógica de decisión, se prioriza el movimiento lateral cuando la inclinación supera el umbral; el salto se detecta con una combinación de elevación de cadera

y velocidad vertical, con cooldown; el avance se activa cuando la feature dinámica (dy) supera un umbral adaptativo (en función de la variabilidad en neutral y de la diferencia observada en forward), aplicando histéresis para evitar oscilaciones.

5.4. Módulo de seguimiento del objeto apuntador

Aunque en una fase preliminar se evaluó la viabilidad de emplear redes neuronales, las pruebas empíricas mostraron resultados insatisfactorios, fallando sistemáticamente en la detección del arma. Esta limitación se atribuyó a la inexistencia de un *dataset* público que contuviera muestras frontales del objeto bajo las variaciones angulares necesarias.

En consecuencia, el diseño final del módulo se fundamenta en técnicas de visión clásica, prescindiendo de detectores preentrenados. En su lugar, se adopta un esquema de calibración manual por fases, donde el usuario selecciona los puntos característicos del objeto directamente sobre la ventana de cámara.

El flujo de calibración del módulo de seguimiento del objeto apuntador (3 fases, 9 clics) es el siguiente:

- En cada fase se registran 3 clics sobre el objeto que se quiere usar como apuntador intentando rodear el mismo.
- Se calcula la circunferencia mínima que engloba los puntos y, a partir de ella, el cuadrado que la circunscribe, que se usa como región de interés (ROI).
- La ROI se almacena como plantilla de referencia y se calcula el color medio en HSV para filtrado posterior.
- Tras completar las 3 fases se inicializa el tracker CSRT y el filtro de Kalman.

La estrategia escalonada de tracking y recuperación que tiene el prototipo es la siguiente:

- **Tracking principal:** CSRT.
- **Predicción:** filtro de Kalman actualizándose constantemente para usar cuando el tracker falla y entra en modo de recuperación.
- **Recuperación local:** template matching multiescala (para tolerar cambios de distancia/zoom) alrededor de la predicción (ventana configurable).
- **Recuperación 'pánico':** búsqueda en toda la imagen si el objeto se mantiene perdido durante cierto tiempo.
- **Filtrado del color:** comprobación HSV con márgenes estrictos para veto y penalización lineal en la puntuación de coincidencia para priorizar candidatos del color exacto.

Los parámetros fundamentales que rigen este comportamiento híbrido se detallan a continuación:

Tabla 5.3: Parámetros de configuración del motor de seguimiento

Parámetro	**Valor**	**Descripción**
Ventana de búsqueda	±100 px	Área local alrededor de la predicción de Kalman.
Umbral de coincidencia	0.70	Similitud mínima requerida en *Template Matching*.
Umbral de pérdida	30 frames	Límite para activar la búsqueda global (modo pánico).
Sensibilidad base	0.5	Factor de conversión inicial (píxeles a movimiento).
Veto de color (Matiz)	±15°	Desviación máxima en el canal H (HSV).

Una vez estimado el centro del objeto seguido en la imagen, se transforma la distancia con respecto al centro de la cámara en desplazamiento del ratón en el juego. La sensibilidad se puede ajustar en tiempo real mediante el uso de las teclas y puede aplicarse una sensibilidad distinta en el eje Y, aunque el prototipo lleva ambas sensibilidades sincronizadas y no contempla la posibilidad de establecer un valor diferente.

5.5. Detección de disparo por cambios de brillo

De forma adicional, el sistema detecta un 'fogonazo' o incremento brusco de brillo dentro de la ROI del objeto seguido para interpretar los disparos. La técnica es deliberadamente ligera: se convierte la ROI a escala de grises, se calcula la media de intensidad y se compara con un brillo ambiente adaptativo. Si la diferencia supera un umbral y se respeta un tiempo de espera, se envía un evento de clic izquierdo.

Tabla 5.4: Configuración de la detección de disparo

Parámetro	**Valor**
Umbral de brillo (`brightness_threshold`)	25
Cooldown de disparo (`shoot_cooldown`)	0.1 s
Adaptación a luz ambiente	EMA: 0.8 ambiente / 0.2 actual

5.6. Módulo principal de ejecución

El bucle principal captura frames de cámara, actualiza el estado del tracker y, si la calibración del objeto está completa, habilita la lógica de pose. El usuario controla el

estado del sistema con atajos de teclado y dispone de feedback visual (texto, puntos de clic y ROI). Los controles principales vienen indicados en la siguiente tabla:

Tabla 5.5: Controles de teclado del prototipo

Tecla	**Acción**
`Q` / `Esc`	Salir del programa
`C`	Activar/desactivar control de ratón por cámara
`M`	Activar/desactivar control de movimiento por pose
`R`	Reiniciar calibración del objeto apuntador (Tracker)
`P`	Reiniciar calibración del módulo de pose (Cuerpo)
`+` / `-`	Ajustar sensibilidad del ratón en tiempo real

6. Fuentes y tecnologías utilizadas

Las tecnologías utilizadas para el desarrollo del prototipo presentado se han clasificado según su función en el sistema:

6.1. Tecnologías

6.1.1. Entorno de Desarrollo y Herramientas

- **Python 3.10.19:** Como lenguaje de programación para el desarrollo del prototipo.
- **Anaconda:** Como gestor de entornos.
- **Visual Studio Code:** Entorno de desarrollo integrado (IDE) utilizado para la escritura y depuración del código.
- **Git:** Sistema de control de versiones para la gestión del código fuente.
- **GitHub:** Plataforma de alojamiento de código fuente y gestión de proyectos basada en el sistema de control de versiones Git.

6.1.2. Librerías de Visión Artificial y Cálculo

- **OpenCV (opencv-contrib-python):** Para captura, tracking CSRT, Kalman, template matching y utilidades de imagen.
- **MediaPipe (Pose Landmarker):** Para la estimación de pose humana en tiempo real.
- **NumPy:** Para operaciones numéricas rápidos y eficientes (Tiempo real).

6.1.3. Librerías de Automatización y Control del Sistema

- **pynput:** Para control de teclado.
- **xdotool (Linux):** Para inyección de eventos del ratón y detección de la ventana activa en Linux.
- **pydirectinput** y **pygetwindow:** Para detección de ventana activa y enviar eventos de ratón en Windows.
- **platform y sys:** Librerías estándar utilizadas para la detección del sistema operativo y adaptación de la ejecución multiplataforma.

6.1.4. Recursos Externos, Hardware y Otros

- **Modelo preentrenado (*pose_landmarker_lite*):** Descargado desde repositorio de modelos de MediaPipe (URL incluida en el código).
- **El videojuego Counter-Strike 2 (Valve):** Se utiliza como caso de uso; no se incluye software propietario en el repositorio.

- **Dispositivo de captura de imagen:** Webcam estándar con resolución de 720p.
- **Discord:** Plataforma para conversar mediante video, voz y texto.
- **Pistola de juguete :** Pistola de juguete que se utiliza como elemento a trackear.

7. Reproducibilidad

A continuación se detallan los pasos necesarios para reproducir el funcionamiento del prototipo:

1. **Obtención del código:** Clonar el repositorio o descargar el código fuente en el equipo local.
2. **Configuración del entorno:** Instalar las dependencias utilizando uno de los siguientes métodos:
   - *Opción A (Conda - Recomendado):* Crear el entorno desde el archivo de configuración:
     `conda env create -f environment.yml`
     `conda activate [nombre_entorno]`
   - *Opción B (Pip):* Crear un entorno virtual estándar e instalar las librerías:
     `pip install -r requirements.txt`
3. **Dependencias del sistema (Solo Linux):** Si se ejecuta en Linux, es necesario instalar la herramienta de automatización:
   `sudo apt-get install xdotool`
4. **Ejecución:** Ejecutar el notebook del proyecto.
5. **Calibración del tracker:** Realizar la calibración del objeto de seguimiento (selección mediante 9 clics en 3 fases).
6. **Inicialización del control:** Realizar la calibración automatica de pose neutra del usuario y activar los interruptores (*toggles*) de control de movimiento y ratón en la interfaz.
7. **Despliegue:** Manteniendo el script en ejecución, cambiar el foco activo (*Alt+Tab*) a la ventana del videojuego para comenzar la interacción.

8. Conclusiones y propuestas de ampliación

8.1. Resultados

Evaluación de resultados: El prototipo desarrollado demuestra la viabilidad técnica de controlar un videojuego de disparos en primera persona (FPS) utilizando una única cámara convencional, combinando el seguimiento de un objeto físico para el apuntado y la estimación de pose para las acciones. La estabilidad del cursor ha mejorado significativamente tras la aplicación de técnicas de filtrado y suavizado.

Limitaciones observables:

- Durante las pruebas de estrés se han identificado desafíos en condiciones de alta dinámica. Específicamente, los movimientos a alta velocidad del objeto apuntador generan desenfoque de movimiento, lo que en ocasiones provoca la pérdida del objetivo por parte del algoritmo CSRT o, en el peor de los casos, la reasignación errónea del seguimiento hacia elementos del fondo con características similares.
- Adicionalmente, la interacción asociada al disparo se ha mostrado muy sensible a las condiciones de iluminación de distintos entornos y a la distancia respecto a la cámara. Variaciones en la luz (sombras, contraluces o baja iluminación) y cambios en la escala del gesto por estar más cerca o más lejos pueden degradar la detección, provocando que en ciertos escenarios el disparo no se active de forma fiable o lo haga de manera inconsistente, constituyendo otra limitación del sistema.

Para una validación futura más exhaustiva, se propone medir métricas cuantitativas como: FPS efectivos, latencia extremo-a-extremo (*end-to-end*), tasa de recuperación tras oclusión y la exactitud en la clasificación de gestos.

8.2. Conclusiones

Se ha implementado con éxito un controlador híbrido funcional que integra visión por computador y emulación de periféricos. El sistema aporta una solución de bajo coste frente a hardware especializado, validando una arquitectura modular que combina tracking de objetos, predicción de movimiento (Kalman) y lógica basada en reglas geométricas para la detección de gestos. La inclusión de una calibración guiada, aunque manual, permite adaptar el sistema a diferentes entornos, demostrando que tecnologías accesibles como MediaPipe y OpenCV son suficientes para prototipar interfaces humano-computadora complejas.

8.3. Propuestas de ampliación

Para evolucionar el prototipo hacia un producto final o una herramienta más robusta, se sugieren las siguientes líneas de trabajo:

- **Automatización de la calibración:** Sustituir la selección manual de color por algoritmos de segmentación y detección automática, minimizando la intervención del usuario en el arranque.
- **Robustez en seguimiento rápido:** Implementar técnicas de seguimiento más avanzadas que permitan mantener la estabilidad ante movimientos de alta velocidad y desenfoque, evitando la pérdida del objetivo o la deriva (*drift*) hacia otros elementos del entorno.
- **Expansión del vocabulario de gestos:** Integrar nuevas acciones semánticas mapeadas a la pose corporal, como agacharse (cambio de altura de caderas), recargar (manos unidas) o cambio de arma.
- **Mejora de la Interfaz (UI/UX):** Desarrollar una interfaz gráfica amigable (GUI) que permita visualizar el estado del tracking y ajustar sensibilidades en tiempo real sin reiniciar el núcleo.
- **Optimización y Portabilidad:** Migrar el código de un entorno interpretado (Notebook) a una aplicación compilada o un servicio en segundo plano, optimizando el uso de hilos (*multithreading*) para reducir la latencia de entrada.

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

A continuación se detalla la relación de recursos identificados como idóneos para la optimización de este proyecto. Este listado integra tanto componentes físicos (materiales y hardware) como activos de software (datasets y herramientas digitales) que, de haber estado disponibles, habrían contribuido significativamente a elevar la calidad técnica y funcional del producto final.

- Un dataset de pistolas vistas de frente cuando apuntan a la cámara desde diferentes ángulos, que podría haber facilitado el encontrar la pistola en la escena para evitar falsos positivos, además de facilitar la recuperación ante perdida de trackeo.
- Un dataset con las diferentes poses(adelante, izquierda, derecha, salto), para sustituir/mejorar el trabajo realizado mediante mediapipe.
- Actuadores que permitan enviar señales al equipo para evitar tratar mediante visión por computador algunos problemas que se resuelven de manera más sencilla con otras técnicas, como la acción de disparo, que se implementaría mejor con un actuador en el gatillo de la pistola.

10. Diario de reuniones del grupo

Con el objetivo de optimizar los tiempos de desarrollo y evitar desplazamientos innecesarios, se estableció una dinámica de trabajo híbrida, priorizando las comunicaciones telemáticas para el control rutinario. El detalle de estas sesiones, así como la modalidad y los hitos abordados en cada una, se recogen en la siguiente tabla.

Tabla 10.1: Cronograma de desarrollo del proyecto

Fecha	**Modalidad**	**Hito**
15/12/2025	Presencial	Planificación inicial para trabajar durante el periodo vacacional y organizar la adquisición de material para el proyecto.
26/12/2025	Telemática	Verificación semanal de cómo iba el avance del trabajo del proyecto.
02/01/2026	Telemática	Verificación semanal de cómo iba el avance del trabajo del proyecto.
08/01/2026	Presencial y Telemática	Reunión para fijar fecha para montar el sistema final y preparar la entrega.
09/01/2026	Presencial y Telemática	Montaje del vídeo demo y finalización de la entrega.

11. Créditos materiales no originales del grupo

Todo el código propio se entrega con licencia [por definir] y autoría del grupo. Los materiales no originales utilizados se citan a continuación.

- **MediaPipe (Google):** modelo de pose preentrenado utilizado por Pose Landmarker.
- **OpenCV:** librería de visión por computador.
- **pynput / xdotool / pydirectinput:** herramientas para inyección de entradas.
- **Counter-Strike 2:** marca y software propiedad de Valve (solo como entorno de prueba).

12. Enlaces y material adicional

12.1. Enlaces de interés

- **Repositorio:** GitHub
- **Notebook principal:** Jupyter Notebook
- **Vídeo comercial:** YouTube
- **Licencia MIT:** Licence

12.2. Uso de IA

El presente trabajo ha contado con la asistencia de herramientas de Inteligencia Artificial para las siguientes tareas específicas:

- **Generación de activos visuales:** Logo de la memoria, diagrama de la arquitectura del sistema.
- **Desarrollo de software:** Asistencia en la generación, depuración y optimización de scripts en Python.
- **Maquetación y formato:** Apoyo en la sintaxis y estructuración del documento en LaTeX.

Bibliografía

[1] K. Thushara, "Seguimiento de objetos con reterminal y cámara pi con opencv." https://wiki.seeedstudio.com/es/reTerminal_DM_Face-tracking/, 2023. Sitio web.

[2] M. Castrillón Santana, "Tema 7: Movimiento." Diapositivas de la asignatura Visión por Computador, 2025. Material docente en Campus Virtual.

[3] otsedom, "Repositorio de la entrega del trabajo práctico de la asignatura de visión por computador." https://github.com/otsedom/otsedom.github.io/tree/main/VC/Trabajo, 2025. Repositorio de GitHub.

[4] otsedom, "Repositorio de la práctica p6 de la asignatura de visión por computador." https://github.com/otsedom/otsedom.github.io/tree/main/VC/P6, 2025. Repositorio de GitHub.

[5] Two Steps From Hell y Thomas Bergersen, "Two steps from hell - protectors of the earth (invincible)." https://www.youtube.com/watch?v=TU_ENLACE_AQUI, 2011. Música de fondo del vídeo comercial. Video de YouTube.