Trabajo de Curso

Grupo 19

Visión por computador
Universidad de Las Palmas de Gran Canaria

Neon Caster

Jesus Santacruz Martín-Delgado
Roberto Tejero Martín

26/01/2026

Índice

1. Introducción

En esta memoria se documenta el desarrollo de "Neon Caster", el trabajo práctico final de la asignatura de Visión por Computador de la Universidad de Las Palmas de Gran Canaria. El proyecto consiste en el diseño e implementación de un sistema de entretenimiento interactivo en tiempo real, donde el cuerpo del usuario actúa como único controlador, eliminando la necesidad de periféricos físicos convencionales como el teclado y el ratón.

El objetivo principal de la propuesta consiste en lograr una interacción real y significativa. No se busca que el programa simplemente "detecte" la presencia de una persona, sino que sea capaz de interpretar sus movimientos para controlar la experiencia. La finalidad es convertir el cuerpo humano en el dispositivo de mando, conectando directamente la acción física del usuario frente a la cámara con la respuesta en la pantalla.

La solución final se ha optimizado utilizando un único modelo de estimación de pose (MediaPipe Pose). Al concentrar toda la lógica de interacción en los 33 puntos corporales principales detectados, ha permitido optimizar considerablemente el rendimiento del proyecto y aumentando a la vez la fluidez esencial para la jugabilidad.

2. Tecnologías empleadas

Para la implementación técnica, se ha seleccionado un ecosistema de desarrollo basado en Python, priorizando su versatilidad y su amplio soporte en el ámbito de la visión por computador. La arquitectura del software se sustenta en tres librerías fundamentales que operan en capas diferenciadas: adquisición de imagen (OpenCV), inferencia semántica (MediaPipe) y motor de visualización (Pygame).

2.1. OpenCV (Open Source Computer Vision Library)

Esta biblioteca constituye la base del módulo de percepción. Su función principal en el sistema es gestionar la interfaz de entrada/salida de vídeo y realizar el preprocesamiento de bajo nivel necesario antes de la etapa de inferencia.

Se ha implementado específicamente para:

- **Captura de flujo de vídeo:** Gestión del buffer de la webcam a través de **cv2.VideoCapture**, estableciendo una resolución de 1280x720 píxeles para garantizar un equilibrio entre calidad y carga computacional.

- **Transformaciones matriciales:** Aplicación de operaciones geométricas como el "efecto espejo" (cv2.flip) para facilitar la coordinación ojo mano del usuario, y la conversión del espacio de color BGR a RGB, formato requerido por el modelo de detección.

- **Superposición gráfica auxiliar:** Utilización de primitivas gráficas para dibujar hitboxes y zonas de interés directamente sobre los fotogramas procesados.

2.2. MediaPipe Pose

El núcleo inteligente de la aplicación es MediaPipe Pose, una solución de aprendizaje automático de Google para la estimación de pose de alta fidelidad. Frente a técnicas tradicionales de segmentación por color o sustracción de fondo, esta herramienta permite una comprensión semántica de la estructura corporal.

La elección de esta tecnología se justifica por:

- **Topología detallada:** El modelo extrae 33 marcas de referencia (landmarks) en 3D. El proyecto utiliza intensivamente los puntos del tren superior: hombros (11-12), codos (13-14), muñecas (15-16) y caderas (23-24) para calcular la lógica de interacción.

- **Eficiencia en CPU:** Su arquitectura está optimizada para ejecutarse en tiempo real en dispositivos estándar sin necesidad de aceleración GPU, factor determinante para mantener una tasa estable de FPS.

- **Facilidad de uso**: La librería de python se encarga automáticamente de la gestión, permitiendo centrarse en la lógica de interacción a partir de los *landmarks* obtenidos.

2.3. Pygame

Mientras que las herramientas anteriores gestionan la visión, Pygame actúa como el motor de presentación y lógica de juego. Las capacidades nativas de visualización de OpenCV resultan limitadas para una aplicación interactiva compleja, por lo que se integró esta librería multimedia.

Su implementación ha permitido:

- **Renderizado avanzado:** Gestión de sprites con transparencias para los elementos dinámicos (enemigos, ítems, medallas, explosiones).

- **Control del ciclo de ejecución:** Regulación precisa del tiempo mediante pygame.time.Clock, desacoplando la velocidad de la simulación de la velocidad de procesamiento de la cámara.

- **Sistema de Audio e Interfaz:** Reproducción de efectos de sonido y renderizado de texto/imagenes para la interfaz de usuario (HUD) superpuesta al vídeo.

3. Diseño del Juego y Mecánicas

El diseño de la aplicación se estructura en torno a un modelo de supervivencia arcade. El objetivo fundamental es resistir el mayor tiempo posible ante oleadas de amenazas crecientes, gestionando un número limitado de recursos (vidas, escudos y munición) mediante la ejecución física de gestos corporales.

3.1. Concepto General y Recursos

El núcleo del juego se basa en la gestión de tres estadísticas vitales que determinan la condición de derrota o la capacidad ofensiva del usuario:

- **Integridad (Vidas):** El usuario comienza con un máximo de 3 vidas. Cada impacto directo recibido por un enemigo en una zona vulnerable del cuerpo reduce este contador. Si el contador llega a cero, la partida finaliza.

- **Defensa (Escudos):** Se dispone de una cantidad inicial de 3 cargas de escudo. Estas cargas son consumibles que permiten activar la invulnerabilidad temporalmente.

- **Ofensiva (Balas):** El jugador cuenta con munición limitada (3 disparos inicialmente) para ejecutar ataques especiales de limpieza de pantalla.

3.2. Sistema de Acciones del Jugador

La interacción no se basa en botones, sino en la detección semántica de posturas corporales específicas. El sistema interpreta la geometría del esqueleto para activar dos estados principales:

3.2.1. Mecánica de Defensa (Escudo)

Para mitigar el daño, el usuario debe adoptar una postura defensiva cruzando los brazos sobre el pecho.

- **Activación:** El sistema valida la acción cuando la distancia entre las muñecas es inferior a un umbral predefinido (SHIELD_ACTIVATION_DIST) y se detecta el cruce de las extremidades.

- **Efecto:** Al activarse, se consume una carga de escudo y se genera una barrera visual que protege al usuario de cualquier daño.

- **Restricciones:** Existe un tiempo de enfriamiento (cooldown) tras cada uso para evitar el abuso de la mecánica.

3.2.2. Mecánica de Ataque (Carga y Disparo)

El usuario dispone de una capacidad ofensiva para eliminar todas las amenazas presentes en pantalla simultáneamente. Esta acción se divide en dos fases:

- **Fase de Carga:** El jugador debe juntar las manos frontalmente. El sistema mide el ratio entre la distancia de las muñecas y el ancho de los hombros; si este ratio se mantiene en un rango específico (0.5 - 1.2), se acumula energía progresivamente.

- **Fase de Disparo:** Una vez la carga está completa (100%), el usuario debe separar las manos. Al superar un ratio de apertura de 1.5, se libera el proyectil, destruyendo a todos los enemigos activos y consumiendo una bala.

3.3. Elementos del Entorno

El entorno virtual reacciona dinámicamente a la presencia del jugador mediante dos tipos de entidades:

3.3.1. Enemigos

Se generan en los bordes de la pantalla con trayectorias que atraviesan el área de juego. A diferencia de los juegos tradicionales, no colisionan con un rectángulo genérico, sino que buscan la intersección con los puntos clave del esqueleto del usuario (hombros, codos, caderas, etc.).

- **Comportamiento:** Su velocidad es variable y aumenta progresivamente con la dificultad de la partida.

- **Interacción:** Si contactan con el cuerpo, restan una vida. Si son bloqueados por el escudo, otorgan puntos extra.

3.3.2. PowerUps (Bonificadores)

Para fomentar el movimiento, aparecen periódicamente ítems que caen desde la parte superior de la pantalla. Estos objetos permiten recuperar los recursos gastados:

- **Tipos:** Corazón (recupera vida), Escudo (recupera carga defensiva) y Bala (recupera munición).

- **Recolección:** El usuario debe interceptar físicamente el objeto con la posición de sus manos (left_wrist o right_wrist) antes de que salga por el margen inferior.

3.4. Progresión y Puntuación

El sistema implementa una curva de dificultad ascendente basada en rondas:

- **Sistema de Rondas:** Al alcanzar una puntuación determinada el juego avanza de nivel. Cada nueva ronda incrementa ligeramente la velocidad de los enemigos y reduce el tiempo entre apariciones, exigiendo reflejos más rápidos.

- **Feedback Visual:** El estado del juego se comunica mediante una interfaz (HUD) superpuesta que muestra las barras de salud y munición mediante iconos, así como medallas animadas al completar cada ronda.

- **Puntuación:** Se recompensa al jugador por distintas acciones:

  +2 puntos: Esquivar un enemigo (sale de pantalla sin tocar al usuario).

  +3 puntos: Recoger un power-up.

  +5 puntos: Bloquear un impacto con el escudo.

  +10 puntos: Destruir enemigos con el ataque especial.

4. Implementación y Algoritmos de Visión

La implementación técnica de "Neon Caster" se aleja de la detección de colisiones convencional basada en cajas (bounding boxes). En su lugar, se ha desarrollado un motor de inferencia geométrica que analiza la posición relativa de 33 puntos clave del cuerpo humano en cada fotograma. A continuación, se detalla la arquitectura del código y los algoritmos diseñados.

4.1. Arquitectura del Código

El núcleo del programa es un bucle infinito que orquesta la sincronización entre la captura de vídeo y el renderizado gráfico. El flujo de ejecución por fotograma sigue una estructura secuencial:

1. **Preprocesamiento:** La imagen capturada por OpenCV se invierte horizontalmente (efecto espejo) y se convierte de BGR a RGB para adecuarse a la entrada esperada por el modelo tensorial.
2. **Inferencia:** Se invoca el método pose.process(), que devuelve una estructura de datos normalizada con las coordenadas (x, y, z) de cada articulación.
3. **Lógica Geométrica:** Se extraen las coordenadas absolutas en píxeles de los puntos de interés (muñecas, codos, hombros y caderas) para calcular distancias y estados.
4. **Renderizado:** Se actualiza la superficie de Pygame, dibujando los elementos del juego sobre el fotograma de vídeo procesado.

4.2. Detección de Gestos (Heurísticas)

Para interpretar la intención del usuario sin botones, se han implementado algoritmos basados en la geometría euclidiana y proporciones antropométricas.

4.2.1. Algoritmo de Defensa (Cruce de Brazos)

La detección del escudo no se basa en una pose pre entrenada, sino en una verificación algorítmica de la posición de las extremidades. El sistema considera que el usuario está defendiendo si se cumplen simultáneamente tres condiciones:

1. **Cruce Horizontal:** La distancia entre la muñeca derecha y el hombro izquierdo es menor que la distancia entre la muñeca izquierda y el hombro izquierdo (y viceversa), indicando una intersección de los antebrazos.
2. **Proximidad de Muñecas:** La distancia euclidiana entre ambas muñecas (wrist_dist) es inferior al umbral SHIELD_ACTIVATION_DIST.
3. **Posición Vertical:** Ambas muñecas deben estar situadas por encima de la altura media de las caderas, evitando falsos positivos cuando el usuario descansa los brazos.

4.2.2. Algoritmo de Ataque (Ratio de Apertura)

Para garantizar que la mecánica de ataque funcione independientemente de la distancia a la que se encuentre el usuario de la cámara, no se utilizan distancias absolutas en píxeles, sino un ratio relativo.

Se calcula el ancho de los hombros (shoulder_width) como medida de referencia. El estado de carga y disparo se determina comparando la separación de las manos con esta referencia:

Ratio = Distancia Muñecas \ Ancho Hombros

- **Carga:** Se detecta cuando el ratio se encuentra entre 0.5 y 1.2 (ATTACK_MIN_WRIST_RATIO y ATTACK_MAX_WRIST_RATIO), lo que corresponde a tener las manos juntas frente al torso.

- **Disparo:** Se activa cuando el ratio supera el umbral de 1.5 (ATTACK_TRIGGER_RATIO), indicando una apertura brusca de los brazos superior al ancho de los hombros.

4.3. Sistema de Colisiones Avanzado

En lugar de simplificar al jugador como un rectángulo, se ha implementado un sistema de colisión esquelética. Se ha definido un subconjunto de puntos críticos vulnerables (BODY_KEYPOINTS), que incluye la nariz, hombros, codos, muñecas y caderas .

La función check_skeleton_collision itera sobre cada enemigo activo y calcula su distancia euclidiana respecto a cada uno de estos puntos clave del usuario. Esto permite una jugabilidad de alta precisión donde el usuario puede esquivar proyectiles simplemente inclinando el torso o levantando un codo, aumentando la inmersión.

4.4. Gestión de Estados y Optimización

Dada la carga computacional de procesar visión artificial en tiempo real, la gestión de estados es crítica para mantener la fluidez.

- **Cooldowns (Enfriamiento):** Para evitar la ejecución repetitiva de acciones costosas o ventajas injustas, se implementan contadores decrecientes (shield_cooldown, attack_cooldown) que bloquean las acciones durante un número fijo de fotogramas tras su uso.<
- **Optimización de Recursos:** Las imágenes de los enemigos y efectos visuales se precargan y redimensionan al inicio (load_image), evitando operaciones de disco o escalado durante el bucle de juego. Asimismo, los elementos que salen de la pantalla

se eliminan de las listas de objetos (state['enemies']) para liberar memoria y reducir el número de comprobaciones de colisión por ciclo.

5. Problemas Encontrados y Soluciones Adoptadas

Durante el desarrollo de la práctica se identificaron desafíos técnicos que obligaron a replantear la arquitectura inicial del sistema. A continuación se detallan las principales dificultades halladas y las estrategias implementadas para mitigarlas.

5.1. Pérdida de Seguimiento en Poses Complejas

Se observó que MediaPipe Pose mantiene una alta precisión cuando las extremidades están separadas, pero su rendimiento decae notablemente en poses donde hay superposición de miembros.

- **El Problema:** Específicamente en la mecánica del "Escudo", que requiere cruzar los brazos sobre el pecho, el modelo a menudo perdía la referencia de las muñecas o confundía la izquierda con la derecha al perderse la profundidad visual.

- **La Solución:** Para mitigar esto, no se exige una pose perfecta. Se implementó un algoritmo tolerante que valida la acción si las muñecas están dentro de un radio de proximidad (SHIELD_ACTIVATION_DIST) y por encima de las caderas, ignorando pequeños "temblores" o inestabilidades momentáneas en la detección. Esta solución puede aumentar los falsos positivos pero mejora la jugabilidad considerablemente.

5.2. Carga Computacional por Modelos Simultáneos

La propuesta original contemplaba el uso de MediaPipe Pose (para el cuerpo) y MediaPipe Hands (para gestos de las manos).

- **El Problema:** Al implementar ambos modelos simultáneamente en el bucle principal, la carga de CPU se duplicó, provocando una caída drástica de la tasa de fotogramas (FPS). Esto aumentaba la latencia entre el movimiento del usuario y la respuesta en pantalla, haciendo el juego injugable.

- **La Solución:** Se optó por descartar el modelo específico de manos. En su lugar, se utilizan exclusivamente los puntos de referencia de las muñecas (índices 15 y 16) proporcionados por el modelo de pose corporal. Esto redujo el tiempo de inferencia a la mitad, permitiendo mantener los 30 FPS estables sin sacrificar la funcionalidad principal.

5.3. Limitaciones Multimedia de OpenCV

Inicialmente, se intentó desarrollar toda la aplicación utilizando únicamente OpenCV para el renderizado.

- **El Problema:** OpenCV es una biblioteca de procesamiento de imágenes, no un motor gráfico. Presentaba carencias críticas para un videojuego: no soporta reproducción de audio, el manejo de transparencias (canales alfa) para los sprites es manual y costoso, y el renderizado de texto es limitado.

- **La Solución:** Se integró la biblioteca Pygame. Esto permitió delegar la lógica de juego, la gestión de la música/efectos de sonido (pygame.mixer) y el renderizado de interfaces gráficas avanzadas a un motor especializado, usando OpenCV únicamente para la captura y procesamiento de la visión.

5.4. Variabilidad de la Distancia del Usuario (Eje Z)

Al carecer de sensores de profundidad, el sistema no sabe a qué distancia real está el usuario.

- **El Problema:** Un gesto de "abrir los brazos" ocupa una cantidad de píxeles muy diferente si el usuario está a 1 metro o a 3 metros de la cámara, lo que invalidaba el uso de distancias fijas en píxeles para detectar ataques.

- **La Solución:** En lugar de medir distancias absolutas, el código calcula la relación entre la separación de las muñecas y el ancho de los hombros del usuario. Esto hace que las mecánicas funcionen correctamente independientemente de la distancia del jugador a la cámara.

6. Conclusión

El proyecto "Neon Caster" ha cumplido su objetivo principal de desarrollar un sistema de entretenimiento interactivo en tiempo real que utiliza el cuerpo humano como controlador exclusivo. La implementación de los algoritmos de visión por computador ha logrado una conexión directa e intuitiva entre el movimiento físico del usuario y la respuesta en pantalla.

La elección estratégica de MediaPipe Pose como único modelo de inferencia, complementada con OpenCV para la gestión de vídeo y Pygame para el motor de juego, resultó ser un equilibrio óptimo entre rendimiento y fidelidad. Este enfoque permitió mantener una tasa de fotogramas estable, esencial para la jugabilidad de un arcade.

Los algoritmos heurísticos diseñados para la detección de gestos, como el Cruce de Brazos (defensa) y el Ratio de Apertura (ataque), resultaron ser suficientemente robustos al basarse en proporciones relativas al ancho de los hombros, garantizando la independencia de la distancia del usuario a la cámara. De igual modo, el Sistema de Colisiones Esquelético elevó la precisión del juego al mapear las amenazas directamente a puntos críticos del cuerpo, fomentando una interacción más activa y precisa por parte del jugador.

En resumen, el trabajo final se ha completado siguiendo las ideas originales y se han obtenido unos resultados aceptables, a pesar de que se han detectado fallos leves en la detección del esqueleto.