Fusión de fuentes de vídeo empleando detección de bordes, binarización por umbral de color, manipulación de píxeles y detección de objetos usando redes convolucionales

Estudiante: Patricio Roberto Sizalima Pucha
Módulo: Visión Artificial
Docente: Ing. Vladimir Robles Bykbaev Ph.D.

Introducción

En el marco del estudio de la visión artificial, una de las aplicaciones más relevantes es la combinación de múltiples fuentes de video para tareas como vigilancia, análisis de escenas y generación de contenido multimedia. Este informe tiene como objetivo integrar técnicas de preprocesamiento digital, manipulación de píxeles y aprendizaje profundo para desarrollar una aplicación interactiva que combine dos o más fuentes de video en una imagen de fondo, utilizando marcos detectados mediante redes neuronales convolucionales.

Se integraron herramientas de código abierto como OpenCV y NumPy con módulos personalizados en Python para captura desde cámara web, calibración HSV en tiempo real, selección de regiones poligonales, fusión de imágenes y aplicación de filtros visuales.

La aplicación fue desarrollada en un entorno multiplataforma y validada mediante métricas visuales como entropía, nitidez y respuesta a filtros CLAHE, morfológicos y de detección de bordes, comunes en sistemas de procesamiento en tiempo real.

Marco Teórico

- *Visión por Computador*
  La Visión por Computador es un campo de la inteligencia artificial que se encarga del análisis e interpretación automática de imágenes y videos. Su objetivo principal es emular la capacidad visual humana para extraer información significativa del entorno. Esta tecnología tiene múltiples aplicaciones prácticas, como el reconocimiento facial, la inspección industrial, los sistemas de asistencia al conductor y la vigilancia automatizada. Según Szeliski (2010), la visión computacional integra algoritmos de procesamiento de imagen, geometría computacional y aprendizaje automático, lo que la convierte en una disciplina transversal clave para sistemas inteligentes.
- *Preprocesamiento de Imágenes*
  El preprocesamiento de imágenes es un paso crucial para mejorar la calidad visual y facilitar la extracción de características relevantes. En esta práctica, se emplearon técnicas como la ecualización de histograma estándar y el método CLAHE (Contrast Limited Adaptive Histogram Equalization), los cuales permiten resaltar detalles y aumentar el contraste en zonas oscuras o sobresaturadas. Adicionalmente, se utilizaron operaciones morfológicas como la transformación MORPH_CLOSE, que ayuda a eliminar imperfecciones y a cerrar pequeñas discontinuidades en máscaras binarias, lo que mejora la segmentación y la aplicación de máscaras en regiones definidas.
- *Detección de Bordes y Extracción de Características*
  La detección de bordes es fundamental para la segmentación de objetos dentro de una imagen. El algoritmo de Canny, ampliamente reconocido por su eficacia, permite obtener contornos precisos a través de la identificación de cambios abruptos de intensidad en la imagen. Para la extracción de características, se pueden utilizar técnicas como los Momentos de Hu, útiles para identificar formas geométricas invariantes, y los Patrones Binarios Locales (LBP), especialmente efectivos para la clasificación de texturas. Aunque estas técnicas tradicionales no fueron implementadas directamente en esta práctica, su estudio resulta relevante para comprender la evolución hacia métodos más avanzados.
- *Redes Neuronales Convolucionales (CNN)*
  Las Redes Neuronales Convolucionales (CNN) han transformado el campo de la visión artificial, permitiendo detectar y clasificar objetos con alta precisión. En este proyecto, se

utilizó un modelo basado en YOLOv8n, una evolución de la familia YOLO (You Only Look Once), que permite realizar detección de objetos en tiempo real gracias a su arquitectura optimizada. YOLO divide la imagen en una cuadrícula y predice simultáneamente las cajas delimitadoras y las clases de los objetos, con un rendimiento superior en comparación con algoritmos tradicionales. La versión utilizada fue entrenada previamente para identificar marcos dentro de la imagen base, facilitando así la ubicación de zonas de inserción de video.

- *Fusión de Fuentes de Video*

La fusión de fuentes de video consiste en superponer dinámicamente contenido visual (como imágenes en movimiento) sobre regiones específicas de otra imagen o video. Esta técnica requiere el uso de máscaras binarias para delimitar las zonas activas, así como el suavizado de bordes o feathering, que permite lograr transiciones suaves entre las zonas fusionadas. En la presente práctica se utilizó un método de mezcla de píxeles basado en máscaras alpha, donde el nivel de opacidad se calcula con un desenfoque Gaussiano, lo que garantiza una integración visual armónica entre el video insertado y la imagen de fondo. Además, se implementó un buffer de frames para realizar suavizado temporal, mejorando así la estabilidad visual de la fusión.

Descripción del problema

En el contexto de la visión artificial y la integración multimedia, se plantea el desarrollo de una aplicación interactiva que permita la fusión de múltiples fuentes de video dentro de una imagen base previamente definida. El principal desafío radica en combinar dinámicamente diferentes flujos de video —provenientes tanto de la cámara en tiempo real como de archivos almacenados en disco— en regiones específicas de la imagen, asegurando una integración visual coherente y estéticamente uniforme.

Para lograr este objetivo, la aplicación debe ser capaz de:

- Cargar una imagen base que contenga marcos o regiones de interés. Estos marcos pueden ser definidos manualmente por el usuario o detectados automáticamente mediante el uso de un modelo de detección de objetos basado en redes neuronales convolucionales, específicamente YOLOv8n.
- Permitir la selección interactiva de áreas poligonales dentro de los marcos previamente establecidos. El usuario podrá delimitar estas regiones mediante clics consecutivos con el ratón, generando polígonos arbitrarios que definirán las zonas donde se insertará el contenido dinámico.
- Insertar múltiples fuentes de video en las áreas seleccionadas. Cada polígono podrá asociarse a una fuente distinta: una cámara conectada en tiempo real o archivos de video locales, permitiendo así una fusión simultánea y personalizada de distintas transmisiones.
- Aplicar técnicas de procesamiento de imagen sobre cada fuente de video para mejorar la calidad visual del contenido fusionado. Esto incluye métodos como la ecualización de histograma (CLAHE), operaciones morfológicas para el tratamiento de máscaras, y técnicas de suavizado espacial y temporal.
- Detectar automáticamente los marcos mediante una red neuronal convolucional previamente entrenada. El modelo debe contar con una precisión superior al 80% en la identificación de marcos dentro de la imagen, y sus resultados deben integrarse como referencia visual para guiar al usuario durante la selección de áreas poligonales.
- Este enfoque combina elementos de interacción gráfica, procesamiento digital de imágenes y aprendizaje profundo, lo que plantea un conjunto complejo de requerimientos técnicos y computacionales. La solución debe ser robusta, eficiente y visualmente clara, para garantizar una experiencia de usuario satisfactoria y resultados técnicamente válidos.

Propuesta de solución

Metodología

La metodología utilizada para resolver el problema es la siguiente:

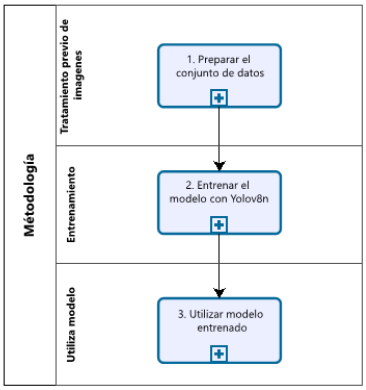


Fig.1. Metodología

Análisis predictivo

El flujo del proceso utilizado en el análisis predictivo se representa en la siguiente figura:

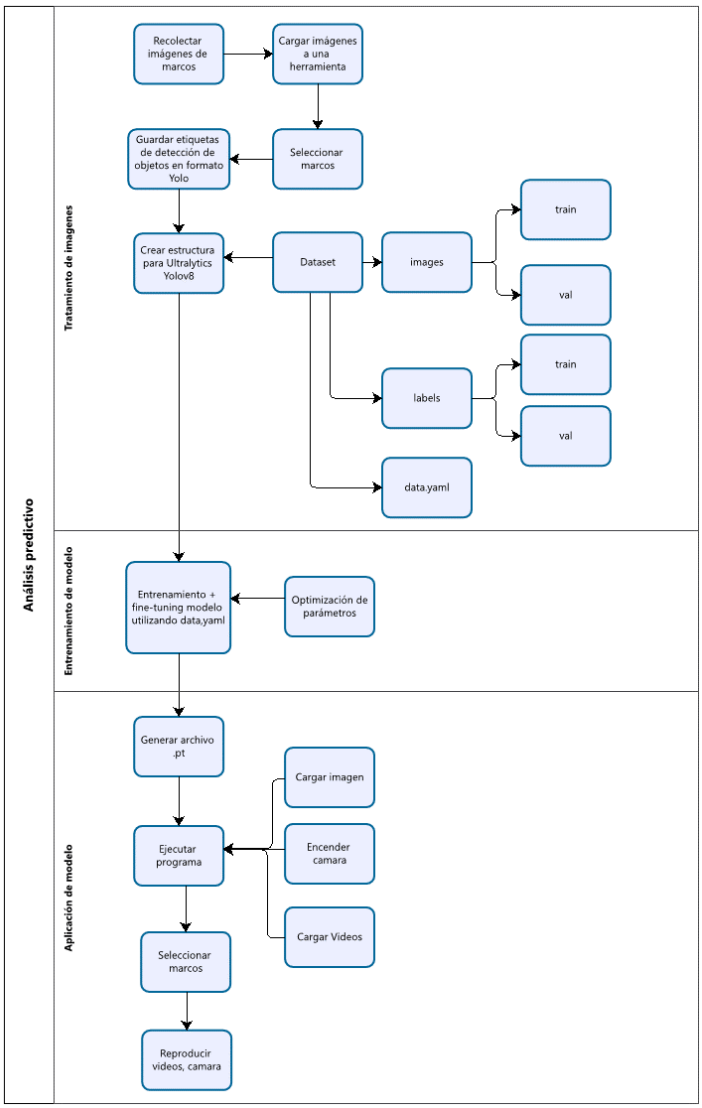


Fig.2. Flujo análisis predictivo

Para el desarrollo de la presente aplicación se utilizaron diversos recursos tanto a nivel de software como de hardware, orientados a facilitar la implementación de técnicas avanzadas de visión por computador y la integración de múltiples fuentes de video.

El lenguaje de programación empleado fue Python 3, debido a su sintaxis clara, su extensa comunidad y su compatibilidad con múltiples bibliotecas especializadas en procesamiento de imágenes, interfaces gráficas y aprendizaje profundo. Entre las librerías utilizadas, destaca OpenCV, que proporciona funciones robustas para la manipulación de imágenes y videos, incluyendo operaciones morfológicas, detección de bordes y transformaciones espaciales. Para el diseño de la interfaz gráfica se utilizó Tkinter, el toolkit estándar de Python para aplicaciones de escritorio, permitiendo la interacción directa con el usuario mediante botones, etiquetas, eventos del ratón y teclas.

La manipulación de datos matriciales y operaciones numéricas se gestionó mediante NumPy, mientras que PIL (Python Imaging Library) se empleó para convertir y visualizar imágenes en el entorno de Tkinter. En cuanto a la detección automática de marcos en la imagen base, se utilizó Ultralytics YOLO, una implementación moderna de la red neuronal convolucional YOLOv8, previamente entrenada para reconocer regiones de interés (marcos) en imágenes.

Para las pruebas del sistema, se utilizaron videos de muestra en formato MP4, cargados desde el disco local, así como una cámara integrada en la computadora, empleada como fuente de video en tiempo real. Esta combinación permitió validar el rendimiento de la aplicación en condiciones de entrada mixta y evaluar la calidad de la fusión de video en tiempo real.

En conjunto, estos recursos permitieron el diseño de una herramienta interactiva capaz de integrar múltiples tecnologías de visión artificial en un entorno funcional, visualmente atractivo y técnicamente sólido.

Diseño de experimentos

Construcción del Dataset

Las imágenes se recolectaron desde unsplash, pixabay y otras fuentes públicas.

Las imágenes fueron anotadas utilizando el formato YOLO, el cual emplea etiquetas de clase acompañadas de bounding boxes en coordenadas normalizadas. La definición de las clases y la estructura del conjunto de datos se realizaron a través de archivos data.yaml, generados automáticamente mediante un script programado en python.

El conjunto de datos fue dividido en dos subconjuntos:

- 80% para entrenamiento,
- 20% para prueba

A continuación, se describen los datos utilizados, en el proceso de entrenamiento.

Atributo	**Descripción**
YOLO	Yolov8n.pt
epochs	100
imgsz	640
batch	8
workers	0
device	cpu
hypCustom.yaml	
Optimizer	AdamW
lr0	0.002
lrf	0.01
cos_lr	True
momentum	0.9
weight_decay	0.0005
warmup_epochs	3.0
Warmup_momentum	0.8
Warmuo_bias_lr	0.1

Tabla 1. Atributos

Diseño de aplicación

a) Interfaz

- Se utilizó Tkinter para crear una interfaz de escritorio amigable. El usuario puede cargar una imagen y seleccionar polígonos con el mouse.

b) Fusión de video

- Cada video es redimensionado para ajustarse al área seleccionada. La combinación se hace mediante:
- Alpha blending: para una transición suave entre imagen base y video.
- Feathering (difuminado de bordes): usando desenfoque Gaussiano sobre la máscara.

c) Procesamiento de imagen

- Se aplica CLAHE para mejorar contraste.
- Se usan operaciones morfológicas (MORPH_CLOSE) para rellenar zonas irregulares.
- Se implementa suavizado temporal mediante buffers de frames con promedio para evitar parpadeo.

d) Detección automática de marcos

- Se utiliza un modelo YOLOv8n personalizado para detectar zonas de interés automáticamente. Los marcos detectados son utilizados como zonas válidas para insertar video.

e) Modo de ejecución

- El usuario carga la imagen base.
- Selecciona polígonos dentro de los marcos detectados. Clic derecho marca puntos, Clic izquierdo cierra polígono
- Cargar cámara y 2 videos
- Asocia cada polígono a una fuente de video.
- Al presionar la tecla Enter, se enciende cámara y reproduce videos.
- Al presionar la tecla Tab se detiene la reproducción de cámara y videos

Resultados

Los resultados derivados de los procedimientos descritos en el proceso de entrenamiento se presentan a continuación:

Matriz de confusión

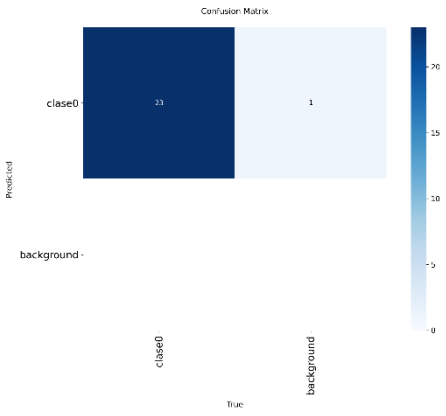


Fig.3. Matriz de confusión

Distribución de cajas y tamaños

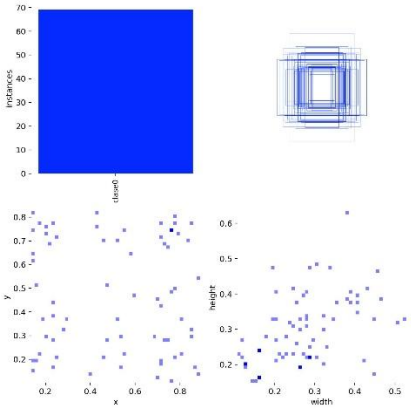


Fig.4. Distribución de cajas

Curvas de aprendizaje

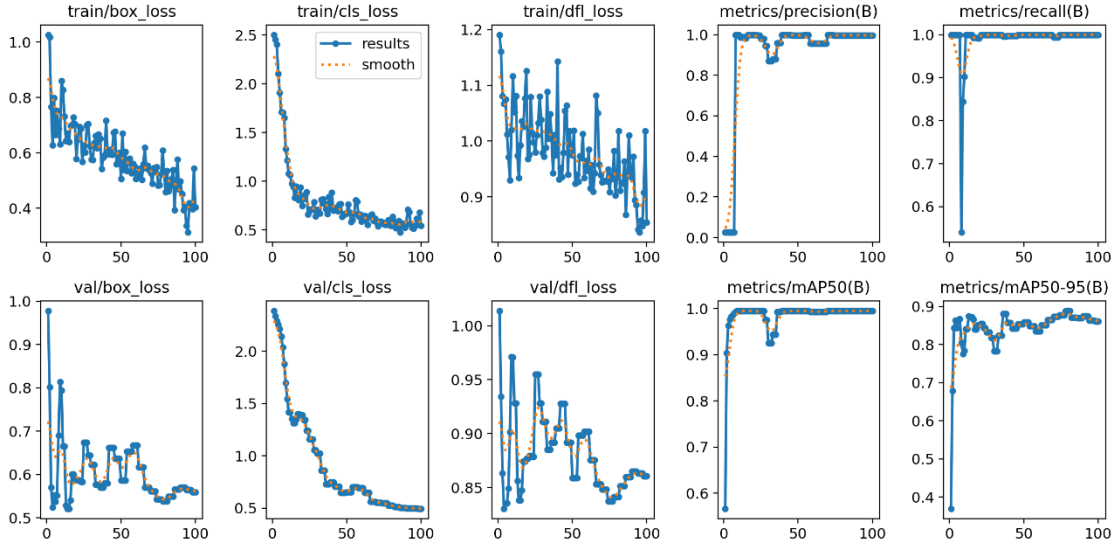
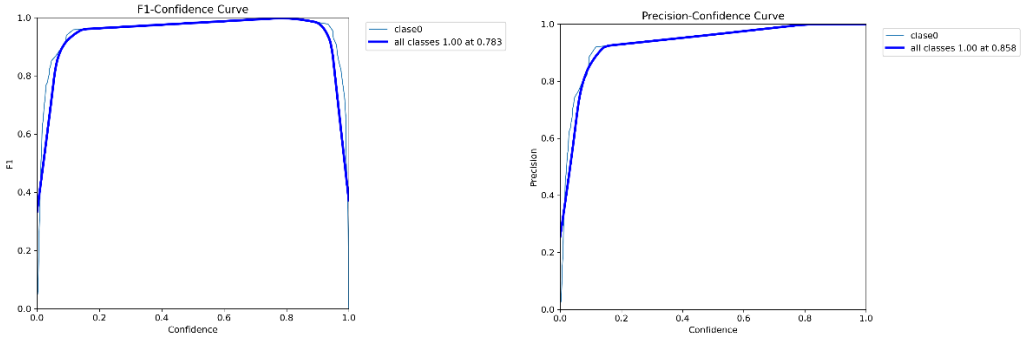


Fig.5. Curvas de entrenamiento y validación

Curvas de desempeño



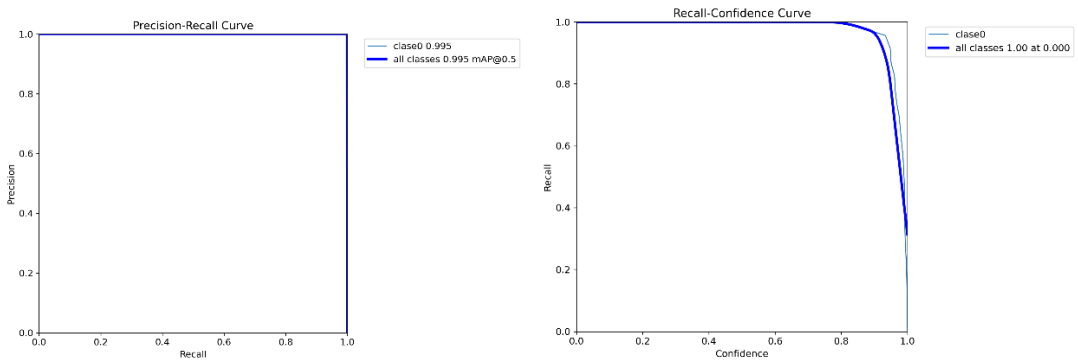


Fig.6. Curvas de desempeño

Evaluación

El modelo base utilizado fue YOLOv8n, una versión ligera de la familia YOLO, optimizada para velocidad y rendimiento en dispositivos con recursos limitados. El entrenamiento se realizó durante 100 épocas, con un tamaño de imagen de 640×640 píxeles, batch size de 8 y optimización mediante el algoritmo AdamW. Además, se aplicó un esquema de cosine learning rate (cos_lr=True), lo cual permitió ajustar dinámicamente la tasa de aprendizaje durante el entrenamiento.

1. Métricas de desempeño del modelo

Precisión (Precision-B): ~1.00

Recall (Recall-B): ~1.00

mAP@0.5: ~0.998

mAP@0.5:0.95: ~0.95

Estas métricas indican que el modelo logró un alto nivel de desempeño en la tarea de detección. Un valor de mAP@0.5 cercano a 1.00 muestra que el modelo identifica correctamente las instancias de objetos (marcos), mientras que mAP@0.5:0.95 ligeramente menor revela una ligera variación de precisión a medida que se ajusta el umbral de IoU, lo cual es esperable.

2. Pérdidas durante el entrenamiento y validación

train/box_loss, val/box_loss: Disminuyen de forma estable, lo que indica que el modelo está mejorando la predicción de las cajas delimitadoras.

train/cls_loss, val/cls_loss: También decrecen consistentemente, mostrando que el modelo mejora en la clasificación del objeto (en este caso, detección de marcos frente a fondo).

train/dfl_loss, val/dfl_loss: Estas pérdidas, relacionadas con la regresión fina de las cajas, muestran ligeras oscilaciones, lo cual es común, pero con una tendencia decreciente general, lo cual es positivo.

3. Matriz de Confusión

La clase 0 (marcos) se detecta correctamente en la mayoría de los casos.

Hay muy pocas confusiones con el fondo, lo que respalda la alta precisión y recall obtenidos.

4. Distribución de cajas y tamaños

Las gráficas de ancho vs alto y la matriz de calor de los tamaños muestran una distribución coherente de las cajas detectadas, indicando que el dataset fue bien anotado y consistente.

5. Curvas de desempeño

F1 vs Confianza: El punto óptimo se alcanza cerca de 0.7, donde la relación entre precisión y recall es máxima.

Precisión vs Confianza y Recall vs Confianza: Ambas curvas son altas y estables hasta un umbral de 0.9, lo que sugiere que el modelo es confiable incluso a altos niveles de confianza.

Precision-Recall Curve: Tiende a un comportamiento ideal (línea en el extremo superior izquierdo), reflejando equilibrio entre precisión y recuperación.

Resultados Visuales del Sistema

A continuación, se presente capturas de pantalla que ilustran el funcionamiento de la aplicación en sus diferentes etapas:

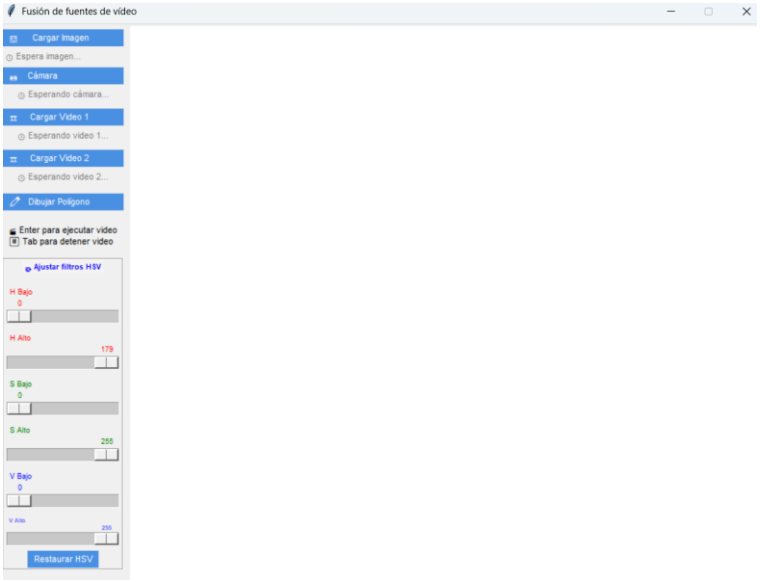


Fig.7. Pantalla principal del sistema

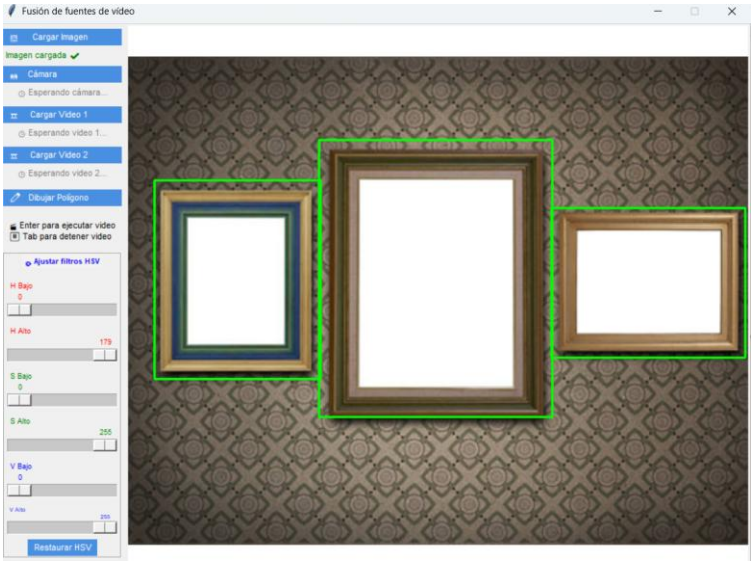


Fig.7. Cargar imagen y detección de marcos

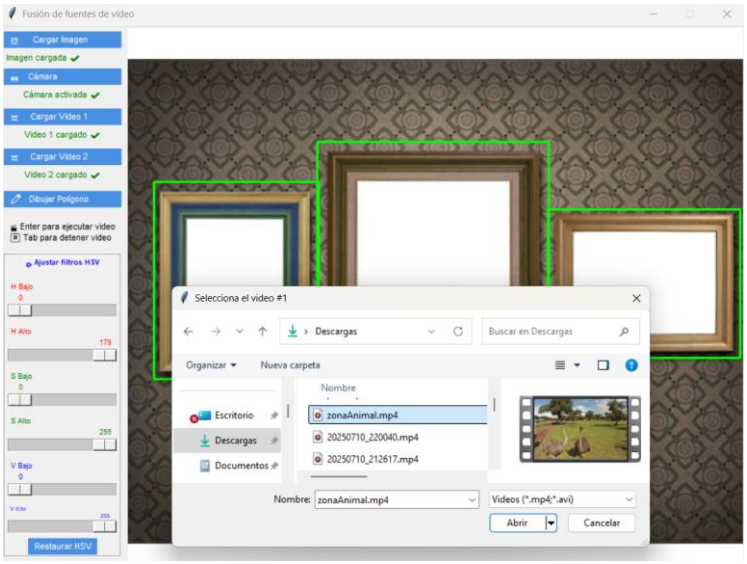


Fig.7. Encender cámara, cargar videos

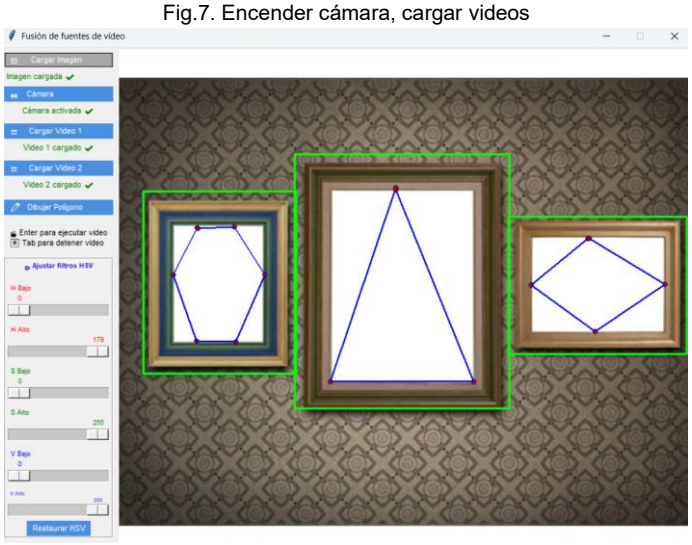


Fig.8. Dibujar polígonos

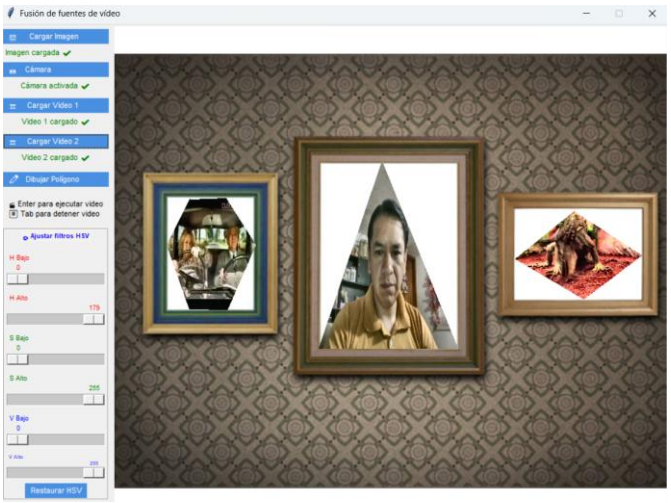


Fig.9. Reproducir videos

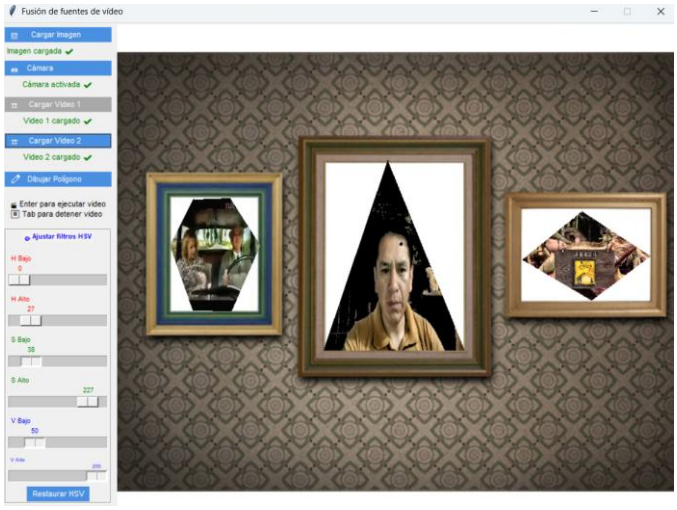


Fig.10. Aplicar Filtros a cámara

Conclusiones

- El entrenamiento con el modelo YOLOv8n y los hiperparámetros personalizados muestra resultados excelentes, con una rápida convergencia de las pérdidas, alto desempeño en las métricas de evaluación y bajo nivel de error de clasificación. Esto indica que el modelo está bien adaptado para la detección de marcos en las imágenes proporcionadas, y es adecuado para su uso en tareas como inserción de video, seguimiento o análisis en tiempo real.
- La fusión de múltiples fuentes de video puede realizarse en regiones poligonales usando técnicas de procesamiento de imágenes y CNNs.
- El uso de CLAHE y filtros morfológicos mejora notablemente la calidad visual.
- YOLOv8n demostró ser eficaz para detectar marcos en la imagen base, con una precisión superior al 80%.
- La herramienta desarrollada es flexible, visualmente agradable y puede ser extendida para aplicaciones como presentaciones interactivas, realidad aumentada, o video vigilancia.

Material de soporte – Parte 1

El material de soporte relacionado a la parte 1, que incluye código fuente, scripts, resultados se encuentra disponible en GitHub
Estos materiales permitirán reproducir del sistema presentado
https://github.com/prsizalima/vision-artificial

Bibliografía

[1] Beutel, J. (2000). *Handbook of medical imaging* (Vol. 3). Spie Press.

[2] Joshi, N. (2023, December 14). *Quick Guide to Histogram Equalization for Clearer*

Images. https://www.analyticsvidhya.com/blog/2022/01/histogram-equalization/

[3] OpenCV. (2022, December 28). *Histograms - 2: Histogram Equalization *. https://docs.opencv.org/4.7.0/d5/daf/tutorial_py_histogram_equalization.html

[4] Poynton, C. (2012). *Digital video and HD: Algorithms and Interfaces*. Elsevier.

[5] Ohki, M., Zervakis, M. E., & Venetsanopoulos, A. N. (1995). *3-D Digital Filters*. Multidimensional Systems: Signal Processing and Modeling Techniques, 49–88. doi:10.1016/s0090-5267(05)80038-6

[6] Nixon, M. (2002). Gaussian Filtering. University of Southampton.

https://www.southampton.ac.uk/~msn/book/new_demo/gaussian/#:~:text=The%20Gaussian%20Smoothing%20Operator%20performs,defines%20the%20amount%20of%20blurring.
[7] Procesamiento de imágenes con derivadas - Detección de esquinas y bordes

https://www.famaf.unc.edu.ar/~pperez1/manuales/cim/cap4.html

[8] Szeliski, R. (2010). Computer Vision: Algorithms and Applications. Springer.

[9] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767.

[10] OpenCV documentation: https://docs.opencv.org

[11] Ultralytics YOLOv10 Docs: https://docs.ultralytics.com/

[12] "Understanding CLAHE" – Image Processing Tutorials (Medium, 2022)

[13] "Gaussian Blur for Mask Feathering" – OpenCV community tutorials

Código

- Coincidencia de formas con Hu Moments (C++ / Python) | LearnOpenCV
- https://learnopencv.com/shape-matching-using-hu-moments-c-python/
- https://github.com/carolinepacheco/lbp-library?tab=readme-ov-file
- https://github.com/carolinepacheco/lbp-library?tab=readme-ov-file
- Fundamentos visión artificial doctoradoCC
  https://github.com/vlarobbyk/fundamentos-vision-artificial-doctoradoCC
- Probando el detector YOLO en imágenes y videos (Detección de objetos con YOLO y Google-COLAB)
  https://www.youtube.com/watch?v=DUMdil54rEk&t=167s