

Fusión de fuentes de vídeo mediante segmentación cromática y filtros visuales en una aplicación interactiva con Python y OpenCV

Juan Carlos Domínguez Ayala

Marzo – Agosto 2025

Datos Generales

Nombre del Estudiante: Juan Carlos Domínguez Ayala

Módulo: Visión Artificial

Docente: Ing. Vladimir Robles Bykbaev

1 Introducción

Este informe presenta el desarrollo de una aplicación interactiva que permite la integración dinámica de múltiples fuentes de vídeo en tiempo real, mediante técnicas de procesamiento digital de imágenes. La solución implementada utiliza segmentación cromática basada en el modelo de color HSV para eliminar fondos homogéneos (como el verde de un croma) y superponer los objetos detectados sobre otras fuentes visuales. Esta técnica ha demostrado ser eficiente para la eliminación de fondos en aplicaciones en vivo [1].

Para lograrlo, se combinaron herramientas de código abierto como OpenCV y NumPy, junto con módulos personalizados en Python que permiten la captura desde cámara web, la calibración del rango HSV en tiempo real, la selección de regiones poligonales para la fusión de imágenes y la aplicación de filtros visuales avanzados. OpenCV, en particular, ha sido ampliamente utilizada en sistemas de visión artificial por su eficiencia y flexibilidad [2].

Además, se desarrollaron rutinas específicas para el reemplazo de fondo mediante máscaras binarizadas, homografía para ajuste espacial, y fusión con texturas o vídeos externos. Esta arquitectura modular facilita la interacción con el usuario, quien define manualmente las zonas donde se aplicarán los efectos. La aplicación fue construida en un entorno multiplataforma y validada con métricas visuales como la variación de entropía, nitidez, y respuesta a filtros CLAHE, morfológicos y detección de bordes. Estos enfoques son comunes en sistemas de procesamiento en tiempo real [3].

2 Marco Teórico

Segmentación cromática en HSV

La segmentación cromática consiste en separar regiones de interés dentro de una imagen en función del color. En este proyecto, se utilizó el espacio de color **HSV** (Hue, Saturation, Value), el cual permite representar el color de manera más robusta ante variaciones de iluminación en comparación con RGB. Se implementó un módulo de calibración en tiempo real que permite al usuario ajustar los valores mínimo y máximo de los tres componentes para generar una máscara binaria que identifica el fondo verde a eliminar. Esta técnica es ampliamente usada en aplicaciones de croma o fondo verde [1].

Homografía y adaptación geométrica

La homografía es una transformación proyectiva que permite mapear un plano de la imagen original sobre una región poligonal arbitraria en otra imagen. En el sistema desarrollado, esta transformación se aplica para incrustar las fuentes de vídeo dentro de polígonos definidos por el usuario sobre una imagen base. La transformación se calcula a partir de los puntos seleccionados con clics del usuario, lo que permite una personalización completa del espacio de inserción y su correcta alineación visual.

Operaciones morfológicas

Las operaciones morfológicas son herramientas fundamentales en el procesamiento de máscaras binarias. En este proyecto, se utilizaron **apertura** y **cierre** para eliminar ruido y llenar huecos en las máscaras resultantes de la segmentación HSV. Estas técnicas, implementadas en el módulo `chroma_module.py`, son útiles para refinar los bordes de las regiones segmentadas y mejorar la calidad de la fusión final [4].

Filtros visuales avanzados

Se implementaron diversos filtros para evaluar su efecto visual sobre la imagen compuesta:

- **CLAHE** (Contrast Limited Adaptive Histogram Equalization): mejora el contraste local sin sobresaturar las regiones claras u oscuras [5].
- **Ecualización de histograma**: distribuye uniformemente los niveles de intensidad para mejorar el contraste global.
- **Canny**: algoritmo de detección de bordes ampliamente usado en visión por computadora [6].
- **Suavizado (blur)**: reduce el ruido y suaviza los bordes entre regiones fusionadas.

Estos filtros fueron integrados en el sistema como opciones aplicables a cada fuente de vídeo, permitiendo comparar su impacto sobre métricas visuales como la entropía y la varianza.

3 Propuesta de Solución

3.1 Estructura del Sistema

1. **Calibración HSV:** Ajuste en tiempo real del rango HSV con cámara en vivo para segmentar fondos verdes.
2. **Selección de polígonos:** El usuario marca zonas sobre una imagen de fondo para indicar dónde se insertarán las fuentes de video.
3. **Fusión de fuentes:** Cada zona recibe una fuente (cámara o video) con opción de aplicar filtros visuales o chroma key.
4. **Evaluación de rendimiento:** Se midió el tiempo de procesamiento de cada filtro y su impacto visual (entropía, varianza). La Figura 1 muestra el flujo completo del sistema implementado, desde la calibración HSV hasta la aplicación de filtros visuales y evaluación de resultados.



Figure 1: Flujo completo del sistema implementado

4 Resultados

El sistema ejecuta la fusión de múltiples fuentes de vídeo dentro de zonas poligonales definidas por el usuario sobre una imagen de fondo. Estas zonas se seleccionan manualmente con clics y se almacenan como polígonos cerrados mediante la interfaz de OpenCV. La aplicación mantiene una tasa de procesamiento promedio superior a 24 FPS, lo cual fue confirmado al finalizar la ejecución, tal como se imprimió en consola.

Para remover el fondo, se utilizó segmentación en el espacio de color HSV, con posibilidad de calibrar los valores en tiempo real usando deslizadores interactivos. Esta calibración se almacena en el archivo `hsv_config.json` y define los rangos de matiz, saturación y valor necesarios para detectar el color verde en la imagen de entrada.

Luego de aplicar la máscara cromática, se sustituyó el fondo segmentado con una textura generada al combinar las imágenes ‘camuflaje.jpg’ y ‘skull.png’. Esta fusión se adaptó a cada polígono usando transformaciones por homografía, lo que permitió insertar cada fuente en su región correspondiente con ajuste espacial.

Además, se habilitó la aplicación de varios filtros visuales sobre las fuentes seleccionadas. Entre los filtros evaluados se encuentran: `clahe`, `hist_eq`, `morph_open`, `morph_close`, `canny`, `blur` y `clahe_canny`. Cada filtro fue aplicado sobre imágenes capturadas desde la cámara y desde video pregrabado. Para medir su impacto, se calcularon automáticamente los tiempos de procesamiento y se almacenaron los resultados visuales antes y después del filtro.

Se generaron dos gráficos comparativos —de entropía y varianza— a partir de estas imágenes procesadas, para analizar el nivel de detalle (información) y dispersión de intensidad que aporta cada técnica. Las Figuras 2 y 3 muestran estos resultados.

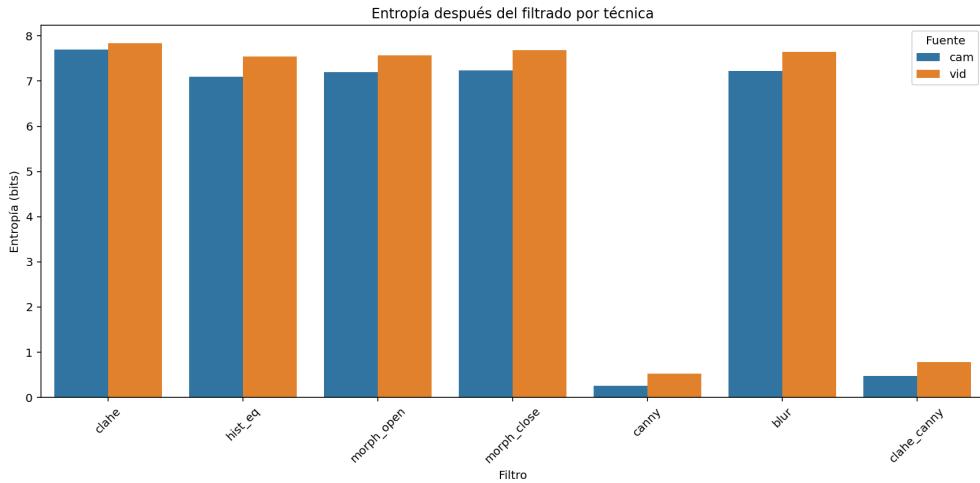


Figure 2: Comparación de entropía por filtro aplicado.

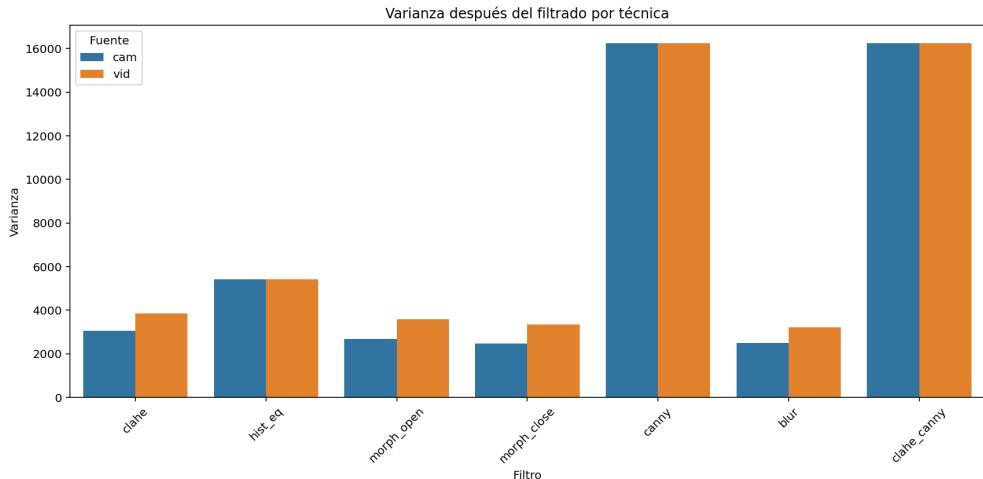


Figure 3: Comparación de varianza en la imagen fusionada según el filtro aplicado.

Los filtros como `clahe` y `hist_eq` generaron imágenes más detalladas, con mayor entropía, mientras que técnicas como `blur` o `morph_close` redujeron la varianza, suavizando bordes e imperfecciones. Esta información permite al usuario decidir qué filtro aplicar según el contexto visual deseado.

5 Análisis de Histogramas por Filtro

Para analizar el efecto de los filtros aplicados sobre las fuentes de vídeo, se generaron histogramas de intensidad de píxeles (0–255), comparando las distribuciones antes y después de cada transformación. Estos histogramas permiten observar visualmente cómo se altera la distribución del brillo, el contraste y los bordes.

Fuente: Cámara

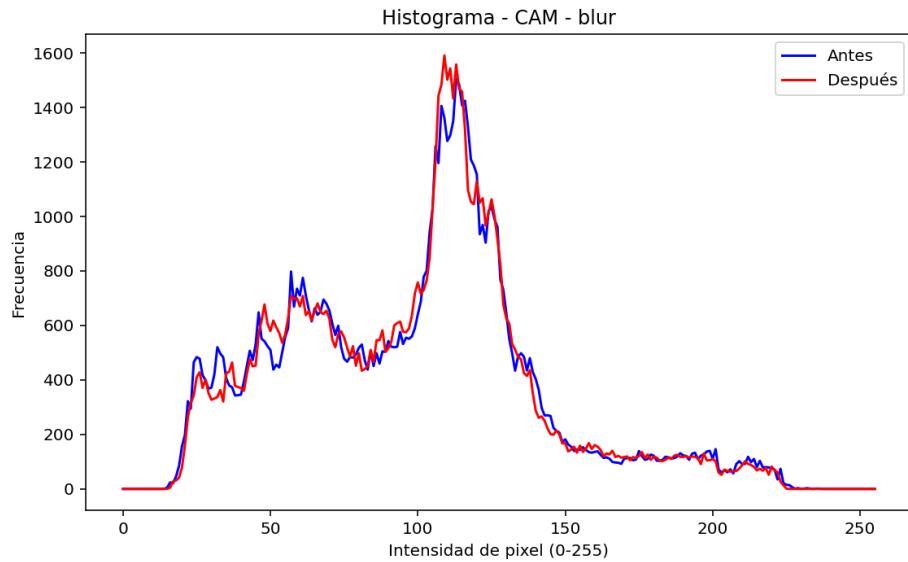


Figure 4: Histograma - Cámara: filtro blur. Suaviza bordes y reduce variaciones rápidas.

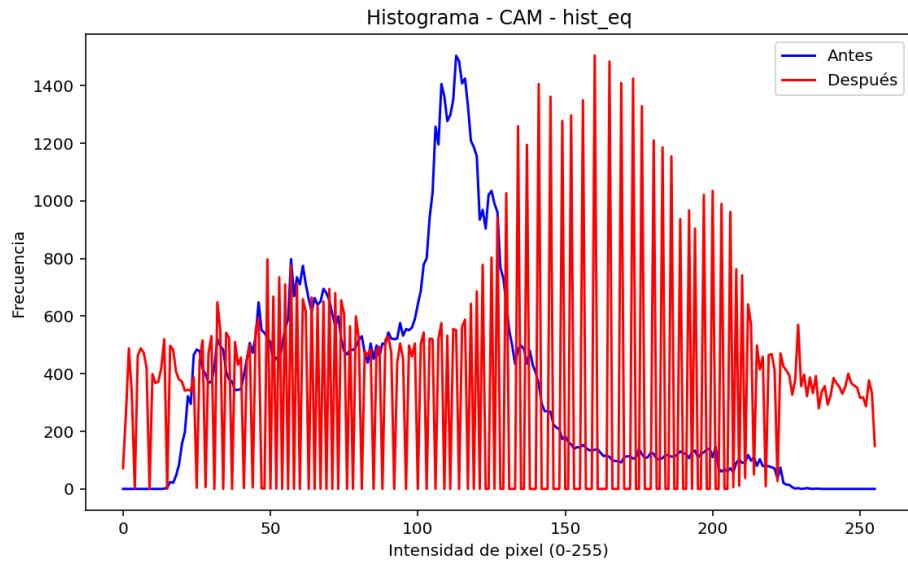


Figure 5: Histograma - Cámara: ecualización de histograma. Aumenta contraste general.

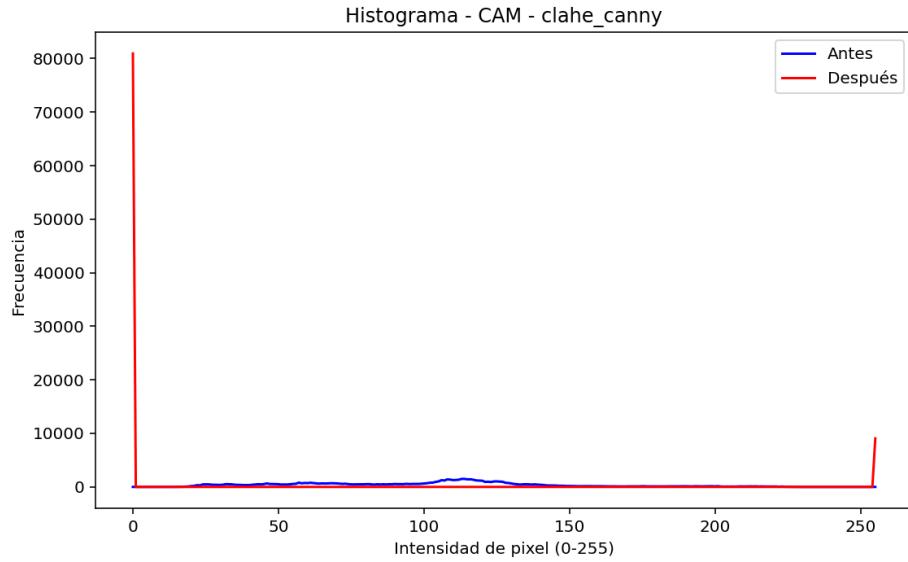


Figure 6: Histograma - Cámara: combinación de CLAHE + Canny. Resalta bordes definidos.

Fuente: Video

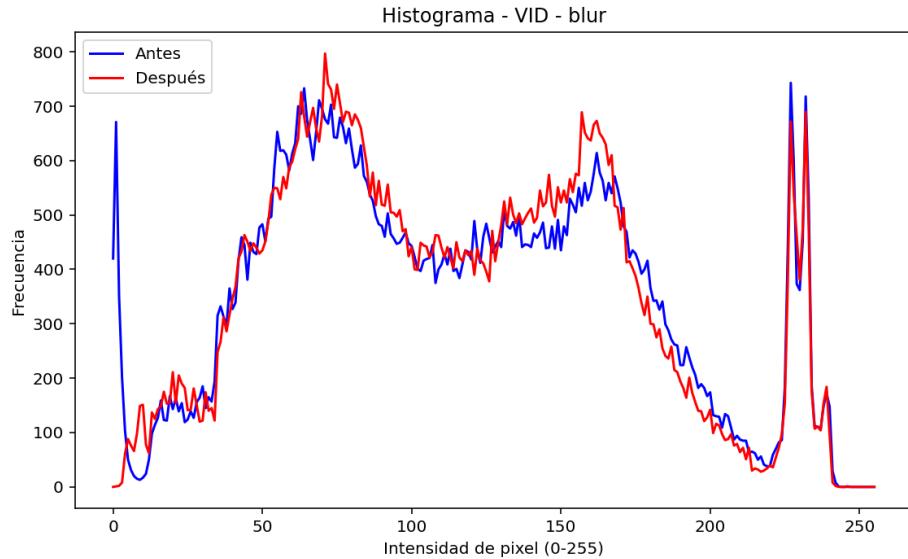


Figure 7: Histograma - Video: filtro blur. Reduce ruido pero conserva forma.

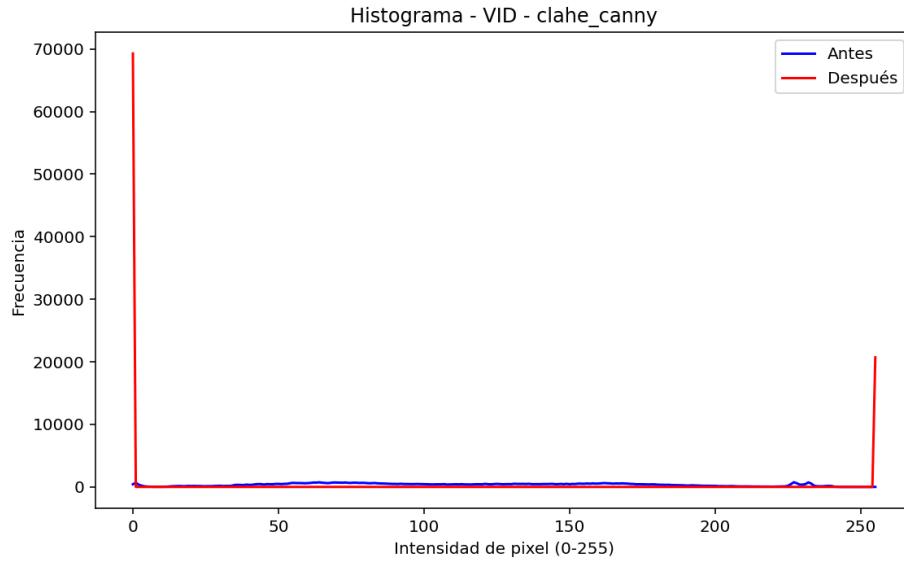


Figure 8: Histograma - Video: CLAHE + Canny. Enfatiza contornos y rangos extremos.

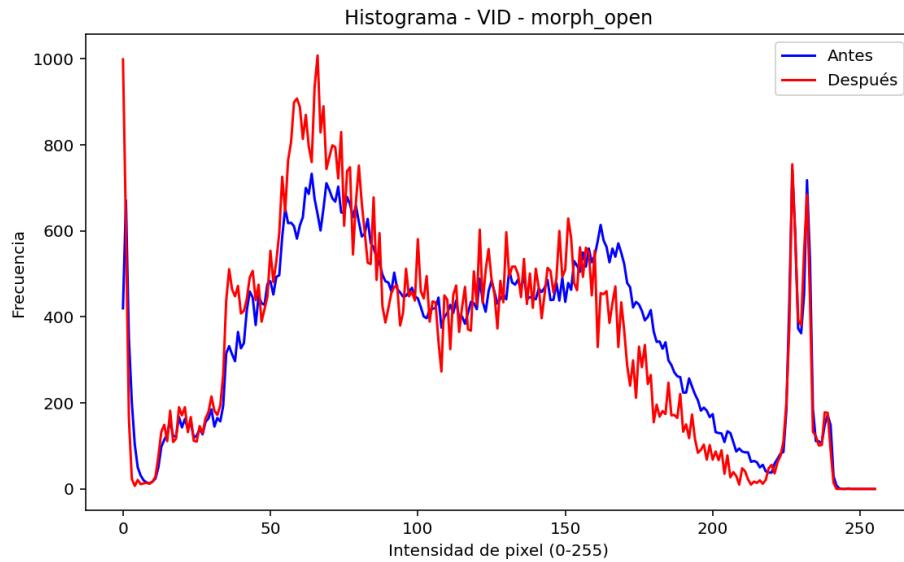


Figure 9: Histograma - Video: morph_open. Elimina ruido aislado preservando estructura.

Los histogramas permiten visualizar con claridad el impacto de cada filtro sobre las imágenes procesadas. Técnicas como CLAHE y ecualización mejoran el contraste, mientras que operaciones morfológicas y suavizados permiten una mayor limpieza y estabilidad visual en la fusión de fuentes.

Video Demostrativo

El funcionamiento del sistema ha sido registrado en el siguiente enlace. También se incluye un código QR para acceso directo desde dispositivos móviles.

- Ver video en línea (OneDrive Stream)



6 Resultados Visuales del Sistema

A continuación, se presentan capturas representativas que ilustran el funcionamiento del sistema en sus diferentes etapas, desde la calibración HSV hasta la visualización final de la fusión con filtros aplicados.

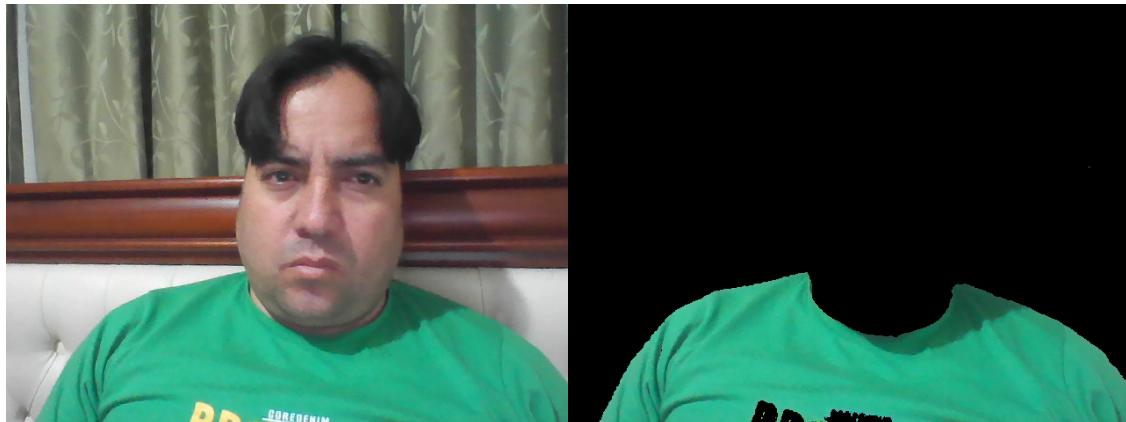


Figure 10: Interfaz de calibración en tiempo real para ajustar los rangos HSV del fondo verde.

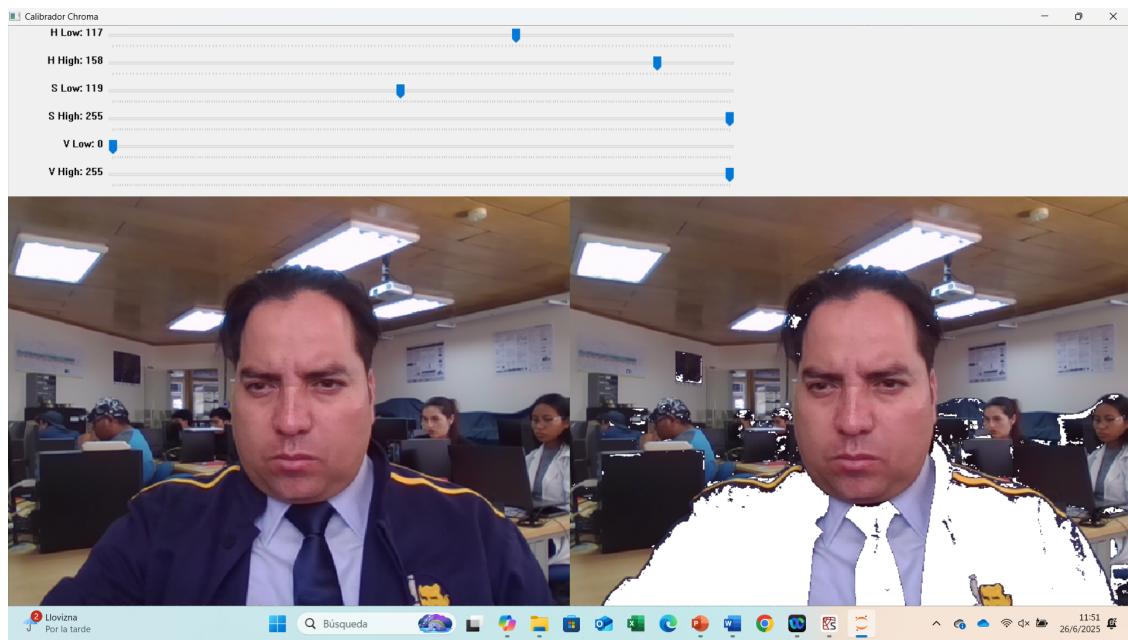


Figure 11: Resultado de la segmentación HSV en tiempo real, con el fondo correctamente identificado.



Figure 12: Visualización de la fusión de fuentes de vídeo aplicando filtros visuales y textura sobre zonas poligonales.



Figure 13: Resultados generados por el sistema en escenarios variados usando filtros personalizados.

7 Conclusiones

El sistema desarrollado logró cumplir con los objetivos planteados, demostrando ser funcional y eficiente en la integración de múltiples fuentes de vídeo dentro de una misma escena. Mediante la segmentación cromática en espacio HSV y el uso de homografía para la adaptación de imágenes a regiones poligonales, fue posible construir una aplicación interactiva capaz de realizar fusiones visuales en tiempo real.

El diseño modular del proyecto permitió incorporar filtros visuales avanzados aplicables de forma selectiva sobre cada fuente, lo cual facilitó el análisis comparativo de su impacto sobre la calidad de la imagen compuesta. Las pruebas realizadas evidenciaron que filtros como `clahe` mejoran el contraste local, mientras que otros como `blur` y `morph_close` ayudan a suavizar bordes o corregir artefactos en regiones segmentadas.

Además, el sistema mantuvo una tasa de cuadros por segundo adecuada para escenarios de ejecución en tiempo real, incluso sin el uso de GPU. Esto refuerza su aplicabilidad en entornos educativos o de bajo costo, donde se requiere una solución liviana y adaptable.

Finalmente, la arquitectura implementada abre la posibilidad de integrar módulos complementarios en el futuro, como detección de objetos mediante redes neuronales o clasificación de escenas, manteniendo la misma base de segmentación, homografía y filtros optimizados.

Recursos Complementarios y Repositorio

Todo el material correspondiente al desarrollo de la Parte 1 del proyecto —incluyendo código fuente, scripts auxiliares, capturas de resultados, histogramas y configuraciones— se encuentra disponible en los siguientes enlaces:

- **Video demostrativo:** Visualizar demostración en línea (OneDrive Stream).
- **Material complementario:** Archivos de soporte en Google Drive.
- **Repositorio del proyecto:** Código y recursos en GitHub.

Estos recursos permiten reproducir y ampliar el sistema presentado, así como revisar los resultados generados en las distintas etapas de desarrollo.

References

- [1] S. Kumar and V. S. Rathore. “Real-Time Green Screen Implementation Using HSV Color Model”. In: *Proceedings of the International Conference on ICT for Sustainable Development*. Ed. by Suresh Chandra Satapathy, Amit Joshi, and Nilanjan Dey. Vol. 1182. Advances in Intelligent Systems and Computing. Springer, 2021, pp. 473–480. DOI: 10.1007/978-981-15-4032-5_50. URL: https://doi.org/10.1007/978-981-15-4032-5_50.
- [2] Gary Bradski. “The OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools* (2000). Disponible en OpenCV.org. URL: <https://opencv.org/about/>.
- [3] M. E. Fathy and M. Y. Siyal. “Real-time image processing approach to measure traffic queue parameters”. In: *IEE Proceedings - Vision, Image and Signal Processing* 142.5 (1995), pp. 297–303.
- [4] Luc Vincent. “Morphological grayscale reconstruction in image analysis: applications and efficient algorithms”. In: *IEEE Transactions on Image Processing* 2.2 (1993), pp. 176–201. DOI: 10.1109/83.217222.
- [5] S. M. Pizer et al. “Adaptive histogram equalization and its variations”. In: *Computer Vision, Graphics, and Image Processing* 39.3 (1987), pp. 355–368. DOI: 10.1016/S0734-189X(87)80186-X.
- [6] John Canny. “A Computational Approach to Edge Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8.6 (1986), pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.