Revista Innovación y Software
Vol. 7, No. 1, Mes Marzo - Agosto, 2026
ISSN: 2708-0935
Pág. 30-48
https://revistas.ulasalle.edu.pe/innosoft

Tipo de artículo: Artículos originales
Temática: Procesamiento de imágenes
Recibido: 7/7/2025 | Aceptado: 29/7/2025 | Publicado: 30/3/2026

Identificadores persistentes:
DOI: 10.48168/innosoft.s29.a343
ARK: ark:/42411/s29.a343

FiltroVis: Una interfaz de usuario para OpenCV en Flet

FiltroVis: An OpenCV user interface in Flet

Valentín Ramos Manuel[1][0009-0003-2893-0015], **Aldo Manuel Aguilar González**[2][0009-0001-7898-7592], **Oscar Chávez Bosquez**[3][0000-0002-0324-9886]*

[1]División Académica de Ciencias y Tecnologías de la Información, Universidad Juárez Autónoma de Tabasco, Cunduacán, Tabasco, México, 86690.. 212h17107@alumno.ujat.mx
[2]División Académica de Ciencias y Tecnologías de la Información, Universidad Juárez Autónoma de Tabasco, Cunduacán, Tabasco, México, 86690.. 212H17035@alumno.ujat.mx
[3]División Académica de Ciencias y Tecnologías de la Información, Universidad Juárez Autónoma de Tabasco, Cunduacán, Tabasco, México, 86690.. oscar.chavez@ujat.mx

*Autor para correspondencia: oscar.chavez@ujat.mx

Resumen
En este artículo se describe el desarrollo de FiltroVis, una interfaz de usuario (UI) interactiva y multiplataforma para la aplicación y visualización en tiempo real de filtros clásicos de visión computacional. La metodología utilizada fue de tipo incremental, estructurada en tres prototipos sucesivos, empleando el framework Flet para la construcción de la interfaz gráfica y la biblioteca OpenCV para el procesamiento del video, resolviendo el problema de rendimiento mediante la implementación de procesos concurrentes. Se obtuvieron resultados satisfactorios en términos de funcionalidad y rendimiento en tiempo real, logrando la correcta aplicación de cada filtro y demostrando la portabilidad del sistema en entornos operativos como Windows, Linux y MacOS, además de vía web. El objetivo de FiltroVis es ofrecer una herramienta educativa y funcional de licencia libre que integre los fundamentos teóricos y las herramientas accesibles como Flet y OpenCV.

Palabras claves: Procesamiento de imágenes, Multiplataforma, Visión computacional

Abstract
This article describes the development of FiltroVis, an interactive, cross-platform user interface (UI) for the real-time application and visualization of classic computer vision filters. We employed an incremental methodology, structured into three successive prototypes, using the Flet framework to build the graphical interface and the OpenCV library for video processing. Performance issues were addressed by implementing concurrent processes. Satisfactory results were obtained in terms of functionality and real-time performance, achieving the correct application of each filter and demonstrating the system's portability across operating environments such as Windows, Linux, and macOS, as well as via the web. The goal of FiltroVis is to offer a free, functional educational tool that integrates theoretical foundations with accessible tools such as Flet and OpenCV.

Keywords: *Image processing, Cross-platform, Computer vision*

Introducción

La Visión computacional es el área de la Computación que se encarga de automatizar y emular la capacidad de la visión humana, permitiendo que las máquinas puedan interpretar, analizar y extraer datos significativos de imágenes y vídeos [1]. La visión computacional tiene diversas áreas de aplicación incluyendo reconocimiento de objetos, monitoreo de seguridad, realidad aumentada, reconocimiento facial, y imágenes médicas. En ese sentido, OpenCV (Open Source Computer Vision Library) es una de las bibliotecas más completas para el desarrollo de aplicaciones de visión por computadora, ya que ofrece más de 2000 algoritmos optimizados que incluyen desde operaciones básicas de procesamiento de imágenes hasta métodos complejos de aprendizaje automático [2]. Al ser de código abierto, es accesible para cualquier desarrollador ya que cuenta con bindings (enlaces) oficiales para ser utilizado en lenguajes de programación como Python (en más popular), C++, Java, entre otros.

Por otro lado, Flet es un framework de Python para construir interfaces gráficas de usuario (UI) interactivas y en tiempo real, que funcionan tanto en aplicaciones web, de escritorio y móviles. Su ventaja es que permite a los desarrolladores construir interfaces visualmente atractivas y responsivas usando únicamente Python, sin necesidad de escribir código en lenguajes de front-end como HTML, CSS o JavaScript [3].

En este artículo se propone una UI (User Interface) desarrollada en Flet para la aplicación de filtros de visión computacional, integrando la biblioteca OpenCV con una UI moderna, intuitiva y multiplataforma. El objetivo es ofrecer una herramienta interactiva para la aplicación y visualización de filtros de procesamiento de imágenes a través de una cámara de video conectada a una PC, la cual permite al usuario seleccionar y aplicar algoritmos clásicos de visión computacional en tiempo real.

Trabajos relacionados

En esta sección se presentan herramientas que, al igual que el proyecto propuesto, buscan facilitar la interacción con algoritmos de visión computacional mediante interfaces orientadas al usuario. Estos trabajos ofrecen el uso de filtros de OpenCV, utilizando diversos lenguajes y bibliotecas. A continuación se describen brevemente las propuestas más representativas y su relación con este estudio.

En primer lugar, la imagen presentada en la Figura 1 muestra la ventana implementa un procedimiento básico para la captura y visualización de video en tiempo real mediante la biblioteca OpenCV usando la instrucción cv2.imshow(), lo que permite al usuario visualizar en tiempo real el flujo de video proveniente de la cámara [2]. Este tipo de implementación corresponde a uno de los ejemplos fundamentales de OpenCV, utilizado en múltiples aplicaciones de visión por computadora. Sin embargo, presenta ciertas limitaciones ya

que la ventana generada carece de elementos interactivos avanzados, no permite controles nativos sobre la interfaz y depende directamente del sistema operativo para su renderizado, lo que dificulta la integración con interfaces de usuario modernas.

Figura 1. Ventana por defecto De OpenCV.

En [4] se describe un proyecto desarrollado en una nueva biblioteca de Python llamada PyVisual que utiliza OpenCV para mostrar el funcionamiento de filtros clásicos mediante demostraciones interactivas diseñadas con fines educativos. Su interfaz se basa en ventanas nativas y controles simples, pero desarrollada sobre una herramienta aún en fase de prueba. En [5] se presenta una aplicación creada con Python, OpenCV y PyQt5, orientada a la visualización de la cámara web desde una GUI de escritorio y al ajuste básico de parámetros mediante componentes gráficos tradicionales. Finalmente, en [6] se propone una herramienta elaborada en C++, que combina OpenCV con una capa gráfica basada en OpenGL, en una aplicación de escritorio orientada al procesamiento de imágenes.

Si bien estos trabajos permiten manipular filtros y parámetros de OpenCV, todos se encuentran limitados a entornos locales y a interfaces tradicionales. Por otro lado, FiltroVis integra OpenCV dentro de una interfaz moderna desarrollada en Flet, ofreciendo ejecución multiplataforma en Windows, Linux, macOS y Web, y proporcionando una aplicación multiplataforma para la enseñanza del procesamiento de imágenes en tiempo real.

Estado del arte

En esta sección se proporciona un panorama general sobre cómo la visión artificial y el procesamiento digital de imágenes se constituyen en diferentes disciplinas. Diversos estudios coinciden en sen˜alar que la biblioteca OpenCV conforma una de las herramientas más empleadas en el desarrollo de sistemas de visión computacional, debido a su carácter de código abierto, su eficiencia y la amplia gama de algoritmos optimizados que ofrece. Por otro lado, en la literatura revisada se describe que OpenCV facilita manejar desde las operaciones básicas de procesamiento de imágenes hasta procedimientos avanzados como detección de características, segmentación, análisis de estructuras y manipulación de video en tiempo real [2].

Existe un consenso entre los investigadores en el campo de la Visión computacional sobre la necesidad crítica de obtener imágenes estandarizadas. Se considera como estandarización a la eliminación de la interferencia y el ruido generado por el medio ambiente en el momento de la captura de la imagen. Este es esencial para facilitar las fases de pre-procesamiento y el proceso completo [7].

En la información revisada dentro del ámbito agrícola, múltiples investigaciones han demostrado la utilidad de OpenCV en tareas de selección visual de frutas, donde se requiere identificar propiedades como color, forma o textura. Diversos autores [8,9] reportan que la biblioteca permite ejecutar análisis de manera rapida y confiables, reduciendo la dependencia del juicio humano y mejorando la precisión en la clasificación automatizada de frutos.

De manera similar, OpenCV también ha mostrado un rol importante en el procesamiento de imágenes médicas, especialmente en la identificación y análisis de lesiones dermatológicas, ya que diversas investigaciones indican que OpenCV facilita la mejora de la calidad visual, la detección de contornos relevantes y la segmentación de regiones de interés [10].

Asimismo, OpenCV ha sido la base sistemas de reconocimiento facial que utilizan técnicas modernas de Inteligencia Artificial, como las Redes Neuronales Convolucionales. Trabajos puntuales son el reconocimiento facial de personas para el control de acceso [11] y sistemas que permiten identificar las emociones de una persona mediante el reconocimiento de rostros [12].

Por último, OpenCV ha sido utilizado en el desarrollo de herramientas educativas y aplicaciones interactivas [13]. En nuestro proyecto, implementamos la biblioteca para aplicar filtros clásicos como conversión a escala de grises, desenfoque Gaussiano o detección de bordes Canny, manteniendo un desempen˜o estable y en tiempo real, lo cual resulta fundamental en aplicaciones que requieren interacción directa con video capturado mediante cámaras locales.

Metodología computacional

El desarrollo de la aplicación siguió una metodología incremental, en la cual el sistema fue construido mediante la adición progresiva de componentes y mejoras funcionales en cada etapa [14]. En lugar de desarrollar el producto completo desde el inicio, cada incremento representó una versión funcional del sistema, que incorporaba nuevas características, refinaba las existentes y mejoraba el desempeño general [15].

Además, se utilizaron herramientas de desarrollo de software que garantizaron aspectos como compatibilidad, eficiencia o estabilidad [16]. Se utilizó la ultima version de ambas bibliotecas para el desarrollo de FiltroVis, siendo éstas las versiones 0.28 y 4.12 de Flet y OpenCV respectivamente. Dichas versiones son compatibles con la última versión de Python (3.13.5), por lo cual gracias a la integración y soporte multiplataforma que proporcionan estas versiones de herramientas permitió desarrollar interfaces interactivas y responsivas. Siendo que Flet es una biblioteca moderna basada en Flutter, permite manejar componentes visuales que integran un estilo visual consistente a tendencias de diseño.

Por otro lado OpenCV, si bien es muy reconocida en el campo de la vision por computadora, esta versión propocionó herramientas para la captura de video desde las cámaras locales, y gracias a OpenCV fue posible realizar las transformaciones en tiempo real de los frames capturados manteniendo un equilibrio de precisión, velocidad y consumo de recursos del sistema.

Se puede decir que el procesamiento de las imágenes se basa en la manipulación de los valores de los píxeles que conforman una imagen con el objetivo de poder resaltar o transformar la información relevante [17]. Es por ello que en esta aplicación se implementaron diversos filtros clásicos que permiten los distintos efectos visuales y se comprenden los principios matemáticos en dichas operaciones [18]. A continuación se presentan los filtros utilizados para esta aplicación [19, 20].

Filtro Escala de grises

Este filtro transforma la imagen capturada reduciendo toda la información a un solo tono de intensidad, lo que permite ver mejor los contrastes y detalles de luz. El resultado fue una imagen más ligera y rápida de procesar, lo cual es útil antes de aplicar otros filtros más complejos. La conversión se obtiene mezclando los colores principales según la forma en que solemos percibirlos.

Para mostrar cómo se hace esta mezcla, en la Ecuación (1), se puede ver la operación que toma un poco del rojo, del verde y del azul para así obtener un solo valor en gris:

$$Escala = 0,299(R) + 0,587(G) + 0,114(B) \tag{1}$$

Con está operación se consigue que cada píxel quede representado solo con un tono, conservando la iluminación de la imagen aunque ya no tenga color.

Filtro Canny

Este filtro permite detectar bordes de manera precisa, marcando los contornos de objetos o zonas donde cambia la intensidad de la imagen. Durante las pruebas se observó que es sensible a los cambios bruscos, lo cual ayuda a que los límites entre figuras se puedan ver definidos. El filtro Canny, introducido por John Canny en el año 1986, toma en cuenta qué tanto cambia una zona con respecto a otra para decidir si es un borde.

Primero calcula el cambio general, como se muestra en la Ecuación (2):

$$G = G2 + G2$$

(2)

Y después calcula hacia qué dirección ocurre ese cambio, como se indica en la Ecuación (3):

$$\theta = \tan^{-1} \frac{G_y}{G_x}$$

(3)

Estas dos ecuaciones permiten que el filtro identifique únicamente los bordes importantes y elimine detalles que no aportan.

Filtro Gaussiano

Este filtro suaviza la imagen para reducir el ruido de los frames, mientras que la inversión de colores y el enfoque da como resultado el realce en el contraste y nitidez de los objetos capturados. En la práctica, se observó una imagen más uniforme, sin bordes abruptos, lo que valida la efectividad del suavizado como herramienta para eliminar ruido. Este proceso se basa en la convolución de la imagen original con una función de distribución normal bidimensional, conocida como núcleo o kernel Gaussiano, esto se representa en la Ecuación (4), donde se muestra cómo se calcula ese suavizado:

$$G(x, y) = \frac{1}{2\pi\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2}$$

Donde σ es un valor que controla el grado de suavizado.

Filtro Inversión de colores

Este filtro genera una imagen negativa, es decir, cambia lo claro por oscuro y lo oscuro por claro. El efecto es particularmente útil para evidenciar la manipulación bit a bit de los valores de los píxeles, confirmando la correcta implementación de la operación lógica NOT. Este cambio se realiza con una operación matemática, como se ve en la Ecuación (5):

$$I_{out}(x, y, c) = 255 - I_{in}(x, y, c) \tag{5}$$

Donde 255 representa el valor máximo de brillo. Al restar el valor original se consigue el color contrario, creando el efecto negativo.

Filtro Enfo

Este filtro mejora la nitidez de la imagen, resaltando los bordes y los detalles. Durante las pruebas se notó que la imagen obtenía mayor claridad, especialmente en zonas donde antes se veía más suave. Este filtro se basa en la convolución de la imagen original con un núcleo diseñado para aumentar la diferencia entre un píxel y sus vecinos.

La matriz que se usa es la que se muestra en la Figura 2, donde el número central (5) representa el valor principal y los números alrededor (-1 y 0) ayudan a resaltar los bordes.

Figura 2. Matriz de realce.

Esta matriz sirve para resaltar los detalles. Finalmente, el funcionamiento del filtro se resume en la operación mostrada en la Ecuación (6):

$$I'(x, y) = I(x, y) * K \tag{3}$$

Donde el símbolo $*$ denota la operación de convolución bidimensional.

Adicionalmente a estas herramientas se integraron otras bibliotecas complementarias como Numpy, la cual es utilizada para el manero de matrices y operaciones numéricas asociadas al procesamiento de imágenes, Threading la cual es una biblioteca dedicada a la gestión de procesos concurrentes que evitan bloqueos durante la ejecución de video, Base64 es una biblioteca empleada para la codificación de imágenes antes de su representación en la interfaz de usuario.

Diseño

Los diagramas UML que se utilizan en la ingeniería de software son fundamentales para permitir a cualquier equipo el desarrollo de ideas complejas de manera clara y sin ambigüedades, lo que facilita la comprensión

de los registros del sistema, al igual que la identificación de los posibles problemas de diseño. Gracias a estos diagramas se asegura que todos los participantes tengan una visión compartida del sistema que se contruye o mantiene. Por ello son herramientas clave para la modelación orientada a objetos a lo largo de todo ciclo de vida del desarrollo de un sistema [21].

La Figura 3 presenta el diagrama de clases que describe la estructura interna de la aplicación. El sistema se organiza alrededor de una clase principal llamada CameraControl, esta clase contiene los atributos que gestionan el estado de la aplicación: gestiona si la cámara está encendida o apagada, controla qué filtro se aplica, maneja los botones de la interfaz y procesa cada frame de video para mostrar el efecto seleccionado. Básicamente, concentra en un solo lugar toda la funcionalidad de la aplicación [22].

La Figura 4 muestra el diagrama de despliegue, el cual representa cómo se distribuyen físicamente los componentes del sistema. Del lado del usuario, una persona accede mediante su navegador web. Este se conecta vía red al servidor, donde se encuentra el programa principal. En el host, Python ejecuta la aplicación, usando Flet para crear la interfaz que ve el usuario y OpenCV para procesar el video de la cámara y aplicar los filtros en tiempo real.

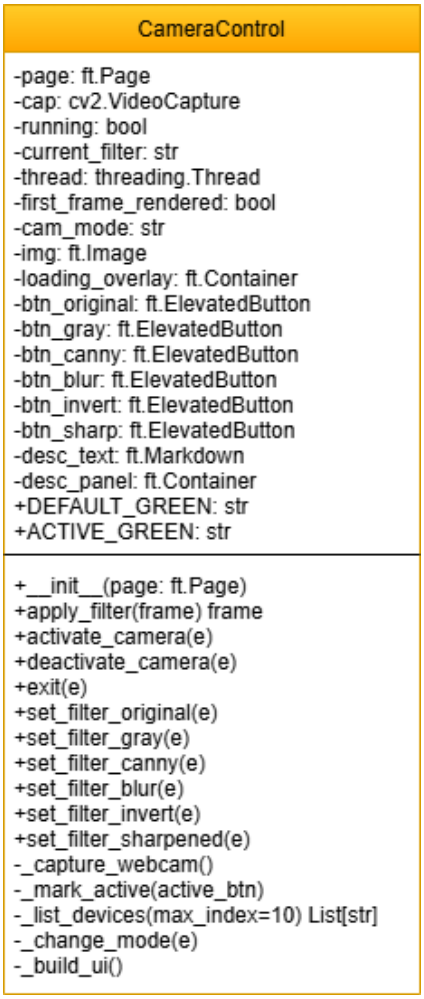

Figura 1. Diagrama de clases.

Desarrollo por prototipos

En total se desarrollaron 3 prototipos funcionales: (i) reproducción de video en Flet, (ii) UI con los componentes principales, (iii) aplicación con filtros. A continuación se describe cada uno de los prototipos.

Prototipo 1

En este caso la imagen obtenida de la cámara web del usuario se muestra sin procesamiento adicional, sirviendo como punto de referencia visual para comparar los efectos de los demás filtros. Esta vista permite evaluar de

Revista Innovación y Software
Vol. 7, No. 1, Mes Marzo - Agosto, 2026
ISSN: 2708-0935
Pág. 30-48
https://revistas.ulasalle.edu.pe/innosoft

forma directa los cambios introducidos por cada transformación y comprobar la fidelidad de la captura de video en tiempo real, la estabilidad del flujo de datos entre la cámara, el procesamiento y la interfaz de usuario. Esto

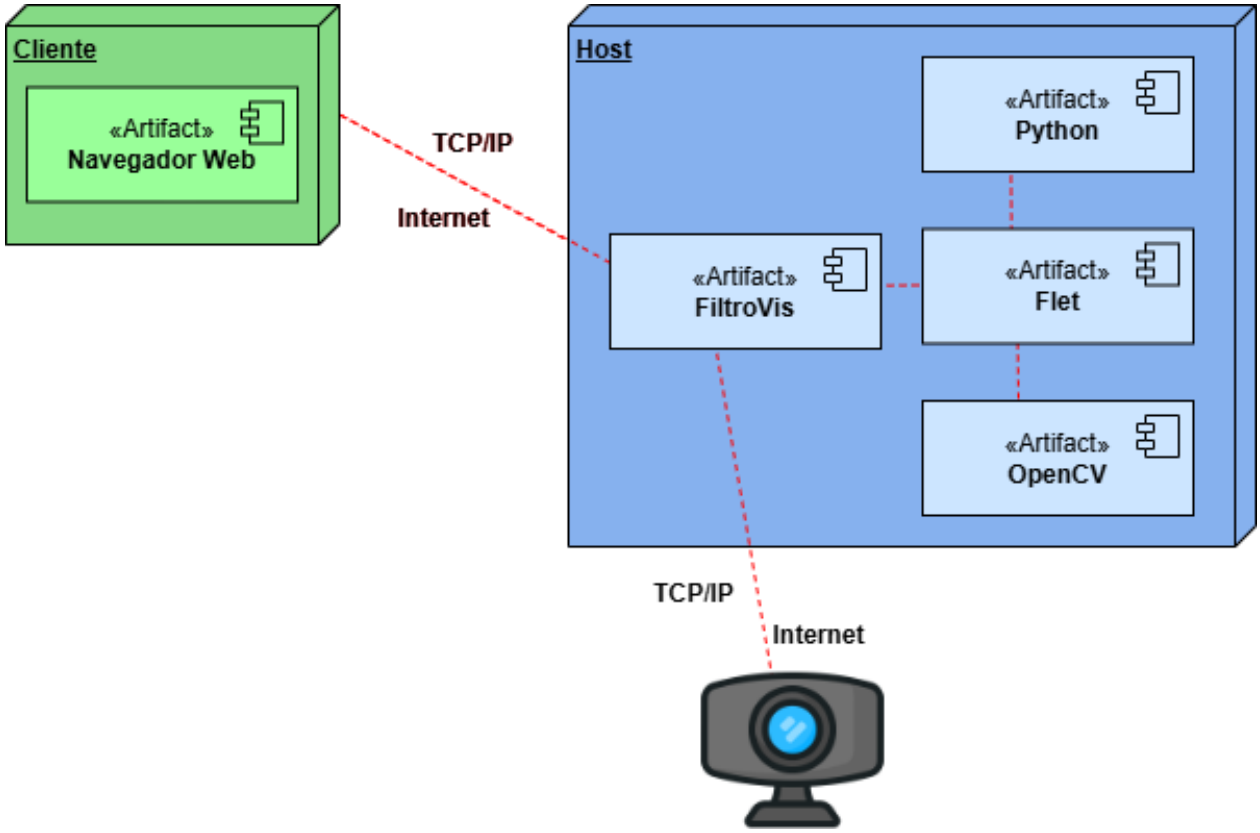

Figura 2. Figura 4. Diagrama de despliegue.

que denominamos como filtro original, no aplica ninguna operación sobre la imagen capturada, mostrando la sen˜al tal cual es adquirida por el sensor de la cámara (Figura 5).

Prototipo 2

La Figura 6 representa el segundo incremento del desarrollo. En esta versión se implementó la estructura básica de la aplicación, centrada en el control de la cámara mediante OpenCV y la visualización en tiempo real en una interfaz desarrollada con Flet. La interfaz gráfica fue sencilla pero funcional, enfocada en demostrar la captura continua de video. Esta etapa sentó las bases técnicas del sistema y permitió establecer la comunicación efectiva de la cámara.

Revista Innovación y Software
Vol. 7, No. 1, Mes Marzo - Agosto, 2026
ISSN: 2708-0935
Pág. 30-48
https://revistas.ulasalle.edu.pe/innosoft

Prototipo 3

De igual manera, en la Figura 7 representa al tercer incremento dentro del ciclo de desarrollo. En esta etapa se incorporó la selección dinámica de dispositivos, permitiendo al usuario elegir entre diferentes cámaras conectadas, incluyendo Webcams locales y cámaras inteligentes. Además, se an˜adieron todos los filtros considerados en estre proyecto. Esta versión también introdujo un panel explicativo en formato Markdown, donde se describe la teoría, fórmulas matemáticas y ejemplos de cada filtro, combinando así el enfoque práctico con el educativo.

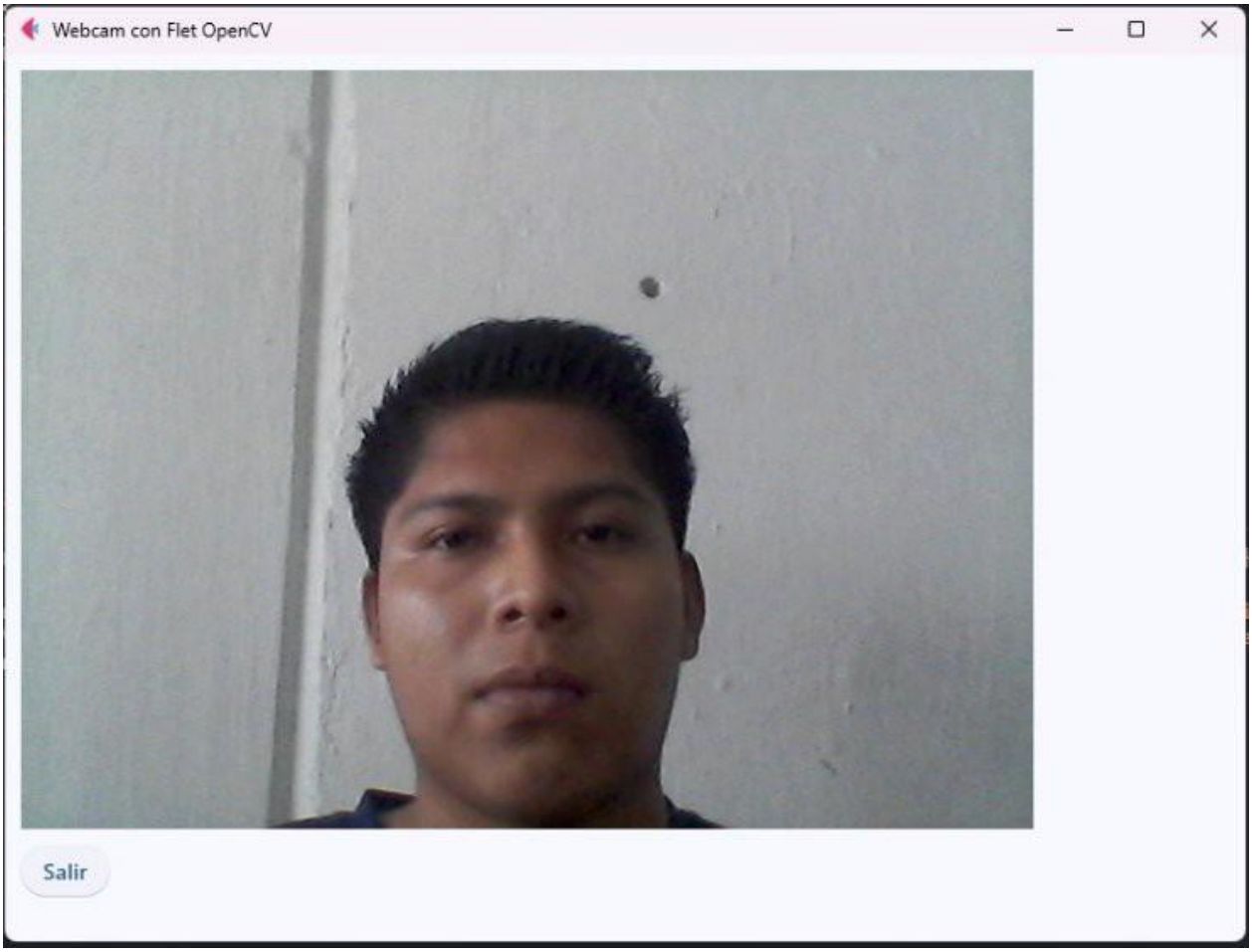

Figura 3. Prototipo 1.

Figura 5. Prototipo 1.

Figura 6. Prototipo 2.

Revista Innovación y Software
Vol. 7, No. 1, Mes Marzo - Agosto, 2026
ISSN: 2708-0935
Pág. 30-48
https://revistas.ulasalle.edu.pe/innosoft

Se mejoró significativamente la interfaz, haciéndola más moderna, organizada y adaptable a distintos taman˜os de pantalla, lo que incrementó la usabilidad y el atractivo visual del sistema.

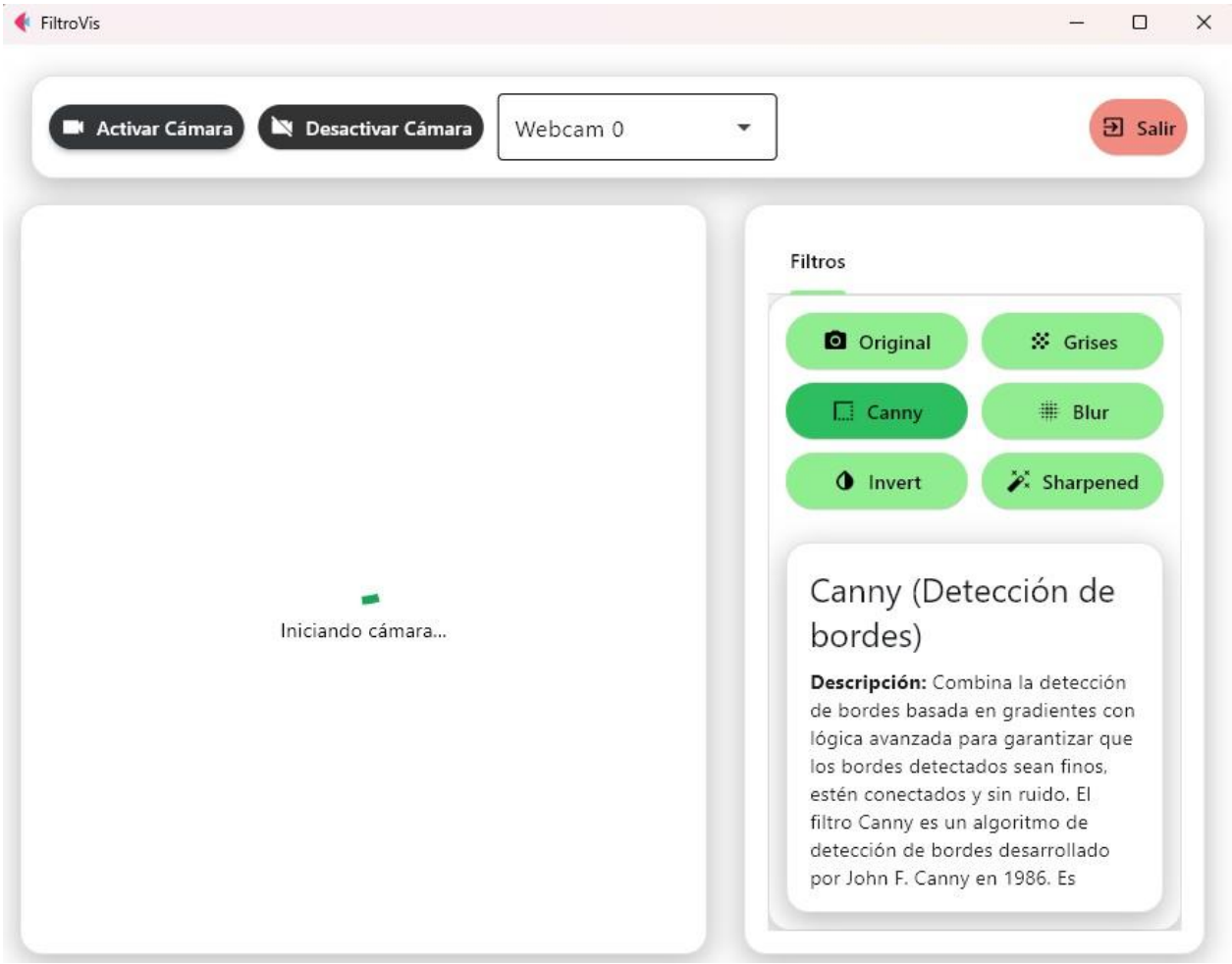

Figura 4. Prototipo 3.

Figura 7. Prototipo 3.

Pruebas

La aplicación FiltroVis fue desarrollada con un enfoque multiplataforma, permitiendo la ejecución en diferentes sistemas operativos, tal sea el caso de MS Windows, MacOS y diferentes distribuciones de Linux, manteniendo

Revista Innovación y Software
Vol. 7, No. 1, Mes Marzo - Agosto, 2026
ISSN: 2708-0935
Pág. 30-48
https://revistas.ulasalle.edu.pe/innosoft

en todos ellos una interfaz coherente y funcional. Su propósito principal es ofrecer un entorno visual que permita aplicar en tiempo real distintos filtros de procesamiento de imagen sobre la sen˜al de una cámara web.

Ejecución en MS Windows

Este fue presentado utilizando el filtro Canny que permite visualizar los bordes de las imágenes detectadas por la cámara Figura 8. El hardware utilizado fue una laptop Lenovo IdeaPad S145-14IIL con sistema operativo Microsoft Windows 11 Home. El equipo cuenta con un procesador Intel®Core™ i5-1035G1 de cuatro núcleos de hasta 3.6 GHz, 8 GB de memoria RAM DDR4 y una tarjeta gráfica integrada Intel®UHD Graphics.

Ejecución en MacOS

Este fue presentado con la captura de video original que demostró fielmente la captura de video desde los diferentes dispositivos conectados (Figura 9). El hardware utilizado fue una computadora Apple iMac 27 Retina 5K con procesador Intel®Core™ i5 de 4 núcleos a 3.2 GHz y 8 GB de memoria RAM DDR3, equipada con tarjeta gráfica AMD Radeon R9 M390 con 2 GB de memoria dedicada.

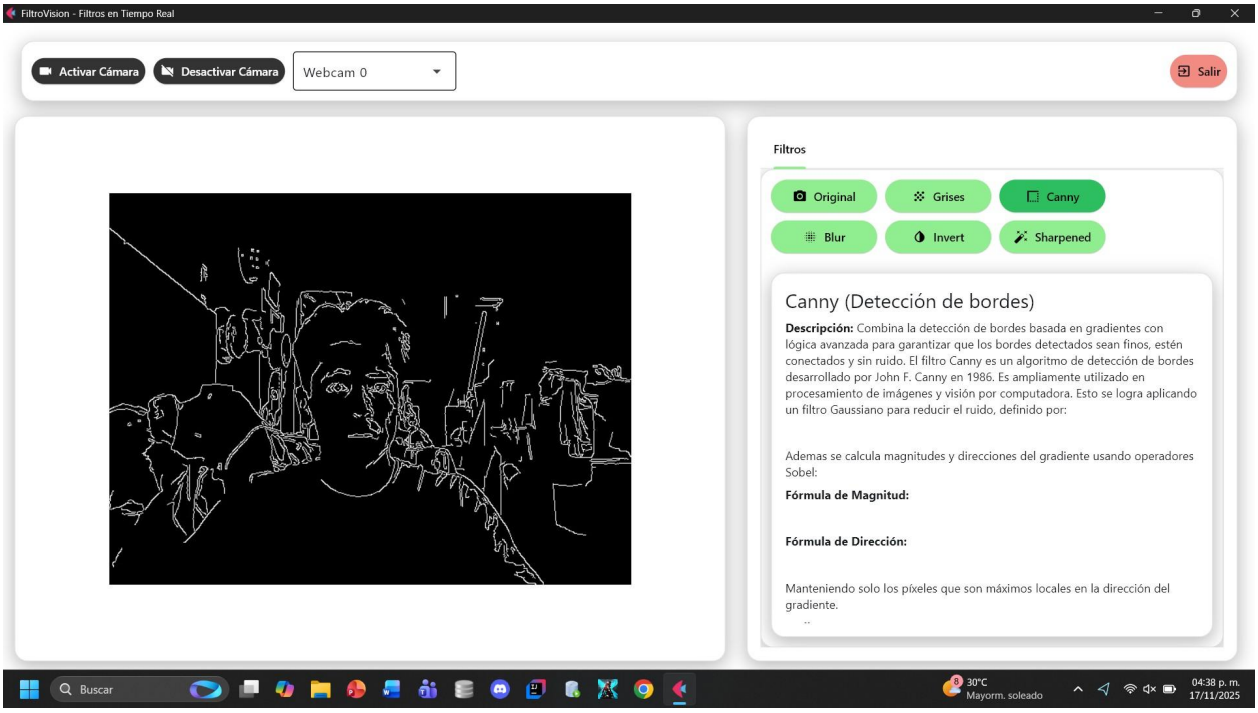

Figura 5. Figura 8. Prototipo en MS Windows.

Revista Innovación y Software
Vol. 7, No. 1, Mes Marzo - Agosto, 2026
ISSN: 2708-0935
Pág. 30-48
https://revistas.ulasalle.edu.pe/innosoft

Ejecución en Linux

En este caso se utilizó Ubuntu Linux 24.04.3 LTS. se presenta la captura de video original (ningún filtro) pero en esta ocasión se muestra el video obtenido desde una cámara web conectada de manera externa, la cual aparece como Webcam 2 en la Figura **??**. El hardware utilizado fue una laptop Honor NBR-WAX9 con procesador Intel®Core™ i5-10210U×8 y 8GB de memoria RAM, Tarjeta gráfica Intel®UHD Graphics (CML GT2), versión de núcleo Linux 6.14.0-36-generic y gestor de ventanas Wayland.

Ejecución Web

El framework de desarrollo Flet proporcionó las bases para que FiltroVis se pudiera enlazar vía Web utilizando dos dispositivos conectados en una misma red, tal como se muestra en la Figura 11. En este caso, se utilizó el sistema operativo MS Windows tanto en el host como en el cliente, y el navegador Google Chrome Versión 142.0.7444.176. Cabe destacar que el video se mostró de manera fluída y sin interrupciones en la intranet de prueba, lo que permite utilizar FiltroVis como una cámara de video vigilancia.

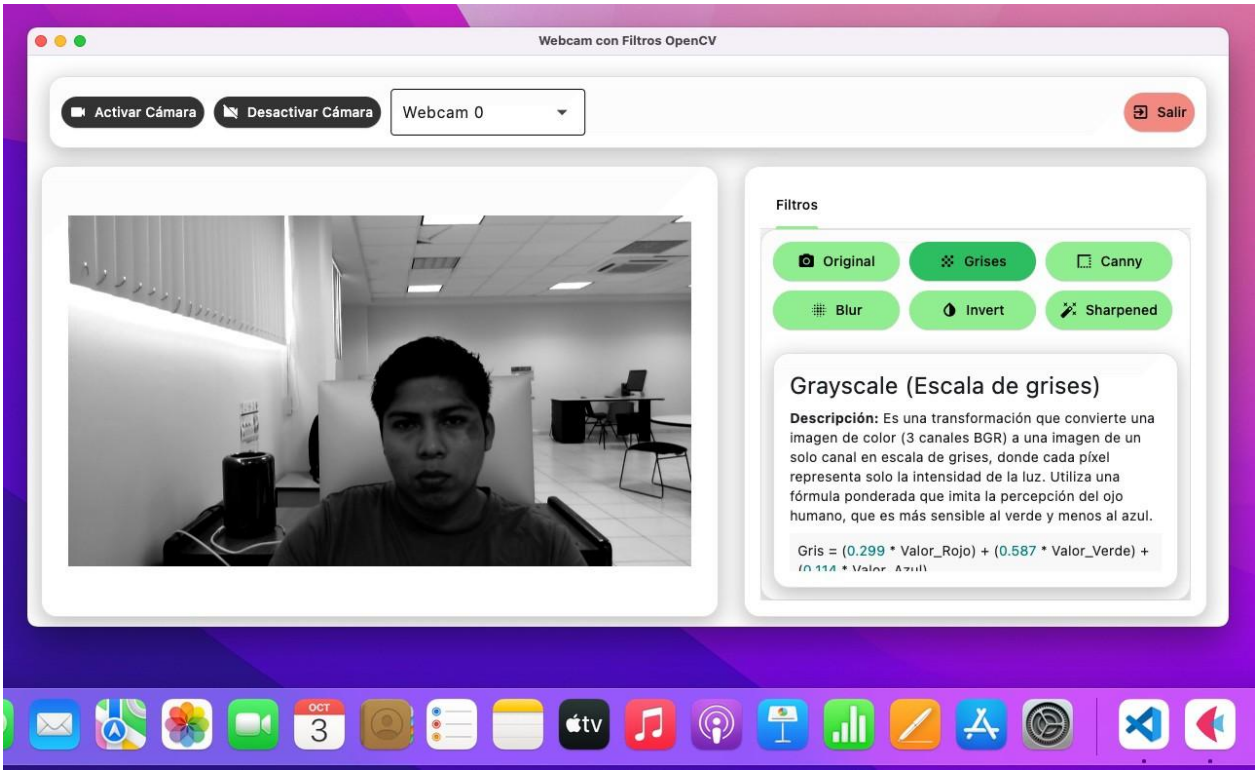

Figura 6. Prototipo en MacOS

Conclusiones

Durante el desarrollo de la aplicación se obtuvieron resultados satisfactorios en cuanto a la funcionalidad, la integración de filtros y el rendimiento del sistema en tiempo real. Sin embargo, el proceso presentó diversas dificultades que permitieron analizar el comportamiento del software y fortalecer la comprensión técnica del entorno de trabajo. Uno de los principales desafíos fue la configuración de las dependencias y versiones de las bibliotecas, particularmente al integrar Flet 0.28 con OpenCV 4.12. En algunos casos, se presentaron problemas relacionados con la detección de la cámara y el bloqueo de hilos al ejecutar la visualización en tiempo real, los cuales fueron resueltos mediante la implementación de procesos concurrentes utilizando el módulo Threading.

Otro punto de análisis fue la eficiencia del procesamiento de video. Se observó que el rendimiento variaba según el dispositivo de captura y la resolución de la cámara. Esto permitió comprobar la necesidad de optimizar la frecuencia de actualización y el tamaño de las imágenes procesadas para mantener una velocidad de respuesta adecuada sin comprometer la calidad visual. En cuanto a la interfaz, se realizaron ajustes iterativos para mejorar la disposición de los elementos y la experiencia del usuario. El uso de contenedores, columnas y pestañas en Flet permitió obtener una interfaz más limpia y moderna, capaz de adaptarse a diferentes resoluciones. Durante estas pruebas se documentaron los resultados mediante capturas de pantalla que muestran el comportamiento del sistema bajo distintos entornos operativos, evidenciando la portabilidad del prototipo.

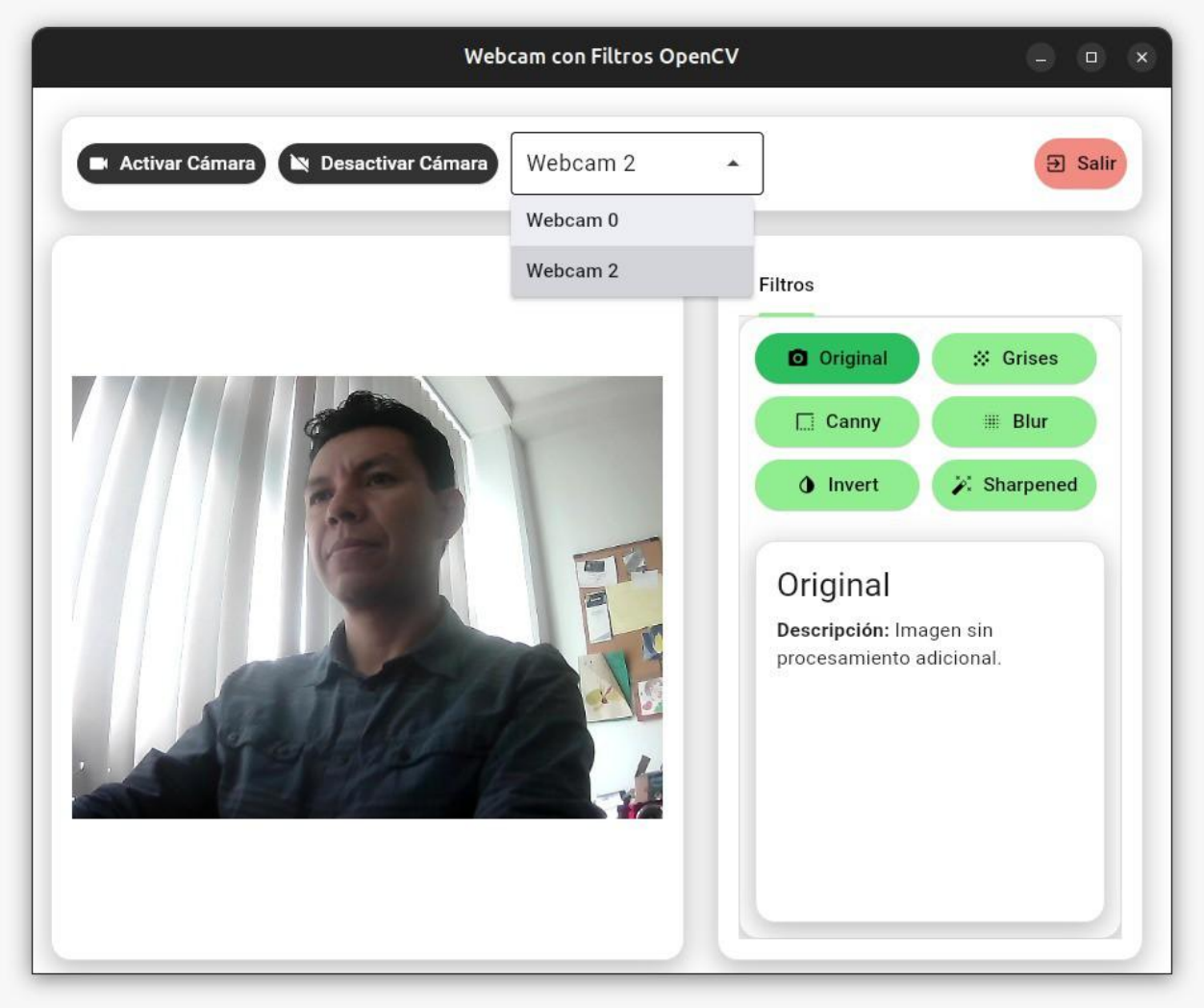

Figura 7. Prototipo en Ubuntu Linux.

Finalmente, el desarrollo de esta aplicación de procesamiento de imágenes con filtros logró el objetivo de integrar de manera efectiva los fundamentos teóricos de la aplicación de filtros a imágenes utilizando Flet y OpenCV, herramientas de código abierto accesibles que permitieron aplicar una metodología de entrega incremental, alcanzando un sistema robusto y eficiente que puede descargarse desde https://github.com/valentin-manuel/FiltroVis.

Contribución de Autoría

Valentín Ramos Manuel: Investigación, Análisis formal, Software, Redacción - borrador original. Aldo Manuel Aguilar González: Visualización, Software, Redacción - borrador original. Oscar Chávez-Bosquez: Conceptualización, Metodología, Análisis formal, Recursos, Supervisión, Administración de proyectos, Escritura, revisión y edición.

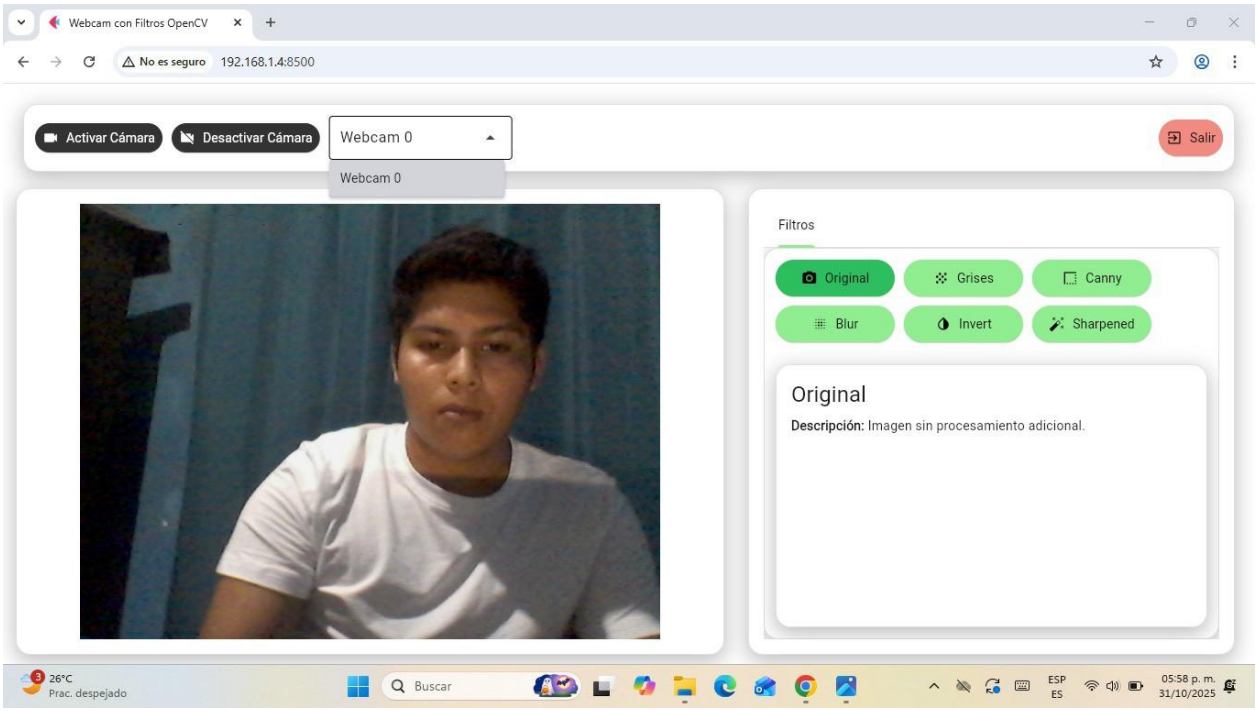

Figura 8. Prototipo Web.

Referencias

[1] L. Sucar and G. Gómez, *Visión Computacional*. México: Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), 2011.

[2] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, 2000.

[3] Flet, "Build multi-platform apps in Python powered by Flutter — Flet," https://flet.dev, 2025.

[4] M. Hassan, "Opencv images: Building modern guis using python," https://www.youtube.com/watch?v=2lnHv0zyP8U, 2025.

[5] F. Wolff, "Opencv gui step-by-step tutorial for beginners," https://fedmsg.com/opencv-gui/, 2023.

[6] D. M. Escriva, "Opencvgui: An opencv graphical user interface," https://damiles.github.io/OpenCVGUI/, 2017.

[7] M. Palacios-Ortega, L. Cruz-Florez, L. Ortiz-Aguilar, J. Mosinõ, and A. Merino-Torres, "Diseño de un prototipo ergonómico para la estandarización de imágenes con aplicaciones en visión artificial," *Ideas en Ciencias de la Ingeniería*, vol. 3, no. 1, pp. 4–21, 2024. [Online]. Available: https://doi.org/10.36677/rici.v3i1.24445

[8] S. Parraga-Badillo and M. Coral-Ygnacio, "Implementaciones de selección visual en frutas: una revisión sistemática de literatura," *Revista Científica de Sistemas e Informática*, vol. 4, no. 1, pp. 1–24, 2023. [Online]. Available: https://doi.org/10.51252/rcsi.v4i1.591

[9] J. Álvarez-Bermejo, D. Morales-Santos, E. Castillo-Morales, L. Parrilla, and J. López-Ramos, "Efficient image-based analysis of fruit surfaces using ccd cameras and smartphones," *Journal of Supercomputing*, vol. 75, no. 3, pp. 10 226–10 337, 2019. [Online]. Available: https://doi.org/10.1007/s11227-018-2284-y

[10] A. Gálvez, A. Iglesias, I. Fister, C. Otero, and J. Díaz, "Nurbs functional network approach for automatic image segmentation of macroscopic medical images in melanoma detection," *Journal of Computational Science*, vol. 56, 2021. [Online]. Available: https://doi.org/10.1016/j.jocs.2021.101481

[11] J. E. M. R. Campos, C. S. C. Rodríguez, L. D. A. Luján, and A. C. M. de los Santos, "Sistema de reconocimiento facial para el control de accesos mediante inteligencia artificial," *Innovación y Software*, vol. 4, no. 1, pp. 24–36, 2023. [Online]. Available: https://doi.org/10.48168/innosoft.s11.a78

[12] A. P. Canazas, J. J. R. Blaz, P. D. T. Martínez, and X. J. Mamani, "Sistema de identificación de emociones a través de reconocimiento facial utilizando inteligencia artificial," *Innovación y Software*, vol. 3, no. 2, pp. 140–150, 2022. [Online]. Available: https://doi.org/10.48168/innosoft.s9.a74

[13] S. Dissanayaka, O. Mudanayaka, T. Halloluwa, and C. D. Silva, "Imagelab: Simplifying image processing exploration for novices and experts alike," https://arxiv.org/abs/2401.03157, 2024.

[14] I. Sommerville, *Ingeniería de Software*, 9th ed. México: Pearson, 2011.

[15] R. Pressman and B. Maxim, *Ingeniería del software: un enfoque práctico*, 9th ed. México: McGraw-Hill/Interamericana, 2020.

[16] D. Mery, "Visión por computador," https://domingomery.ing.uc.cl/teaching/vision/, 2004.

[17] M. M. Ortíz, "Procesamiento digital de imágenes," https://www.cs.buap.mx/~mmartin/notas/PDI-MM-Rev.2013.pdf, 2013.

[18] GIMP, "Matriz de convolución," https://docs.gimp.org/2.6/es/plug-in-convmatrix.html, 2025.

[19] T. D. Mínguez, *Visión Artificial. Aplicaciones prácticas con OpenCV - Python*, 2nd ed. Barcelona: Marcombo, 2025.

[20] C. Lazo and P. Huijse, "Introducción al procesamiento de imágenes digitales," https://phuijse.github.io/UACH-INFO185/clases/unidad1/04_im%C3%A1genes.html, 2022.

[21] I. Jacobson, G. Booch, and J. Rumbaugh, *El Proceso Unificado de Desarrollo de Software*. México: Pearson Educación, 2000.

[22] C. Larman, *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado*. México: Prentice Hall, 2002.