**DESAFÍOS DE PROCESAMIENTO DIGITAL E IMÁGENES**Franco, A, Pelaez, C.

**Politécnico Colombiano Jaime Isaza Cadavid, Medellín - Colombia**

Facultad de Ingenierías

Agosto de 2025

**Resumen**

Desarrollaremos una práctica integral de visión por computador para consolidar fundamentos y habilidades aplicadas. Crearemos y procesaremos datos sintéticos y reales, implementaremos operaciones básicas con y sin librerías, mediremos tiempos, compararemos enfoques y evaluaremos la calidad de los resultados. Usaremos Python, OpenCV, NumPy y herramientas ofimáticas para obtener estadísticas, visualizaciones y evidencias reproducibles. El informe final integrará código, imágenes, tablas y conclusiones, demostrando dominio de generación de datos, análisis de color, filtrado y seguimiento, así como criterios experimentales y de rendimiento.

**1. Introducción**

* Este laboratorio aborda los fundamentos prácticos de la visión por computador mediante una serie de ejercicios integrados que combinan generación de datos, procesamiento de imágenes y evaluación cuantitativa. El propósito es fortalecer criterios de diseño experimental y rendimiento: implementar soluciones con y sin librerías especializadas, medir tiempos de ejecución, comparar resultados y sustentar decisiones técnicas. Se trabajará principalmente en Python con apoyo de OpenCV y NumPy, además de herramientas ofimáticas para documentar y visualizar evidencias. La metodología prioriza la reproducibilidad y trazabilidad: código claro, registros de parámetros, tablas de métricas e imágenes de soporte. El producto final es un informe único que consolida procedimientos, resultados y conclusiones, demostrando dominio operativo sobre tareas básicas de visión por computador y la capacidad de analizar su impacto en calidad y eficiencia computacional.

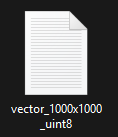
**2. Materiales y procedimiento**

* Lista de materiales
* Computador con Python 3.x
* Cámara web
* Jupyter o Google Colab
* OpenCV, NumPy, Matplotlib
* Google Sheets
* Git y repositorio remoto
* Google Drive u otro almacenamiento
* Imágenes y videos de prueba  
  + Lista de procedimientos
* Preparar entorno y organizar carpetas
* Adquirir y preparar datos
* Implementar algoritmos base
* Instrumentar mediciones de tiempo
* Visualizar resultados
* Comparar enfoques
* Analizar y concluir
* Compilar evidencias
* Sustentar resultados

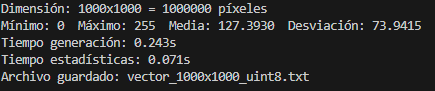
**3. Resultados**

****

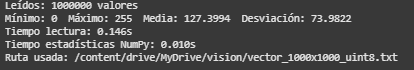
**Figura 1:** Matriz generada

****

**Figura 2:** Archivo generado

****

**Figura 3:** Tiempos generados local

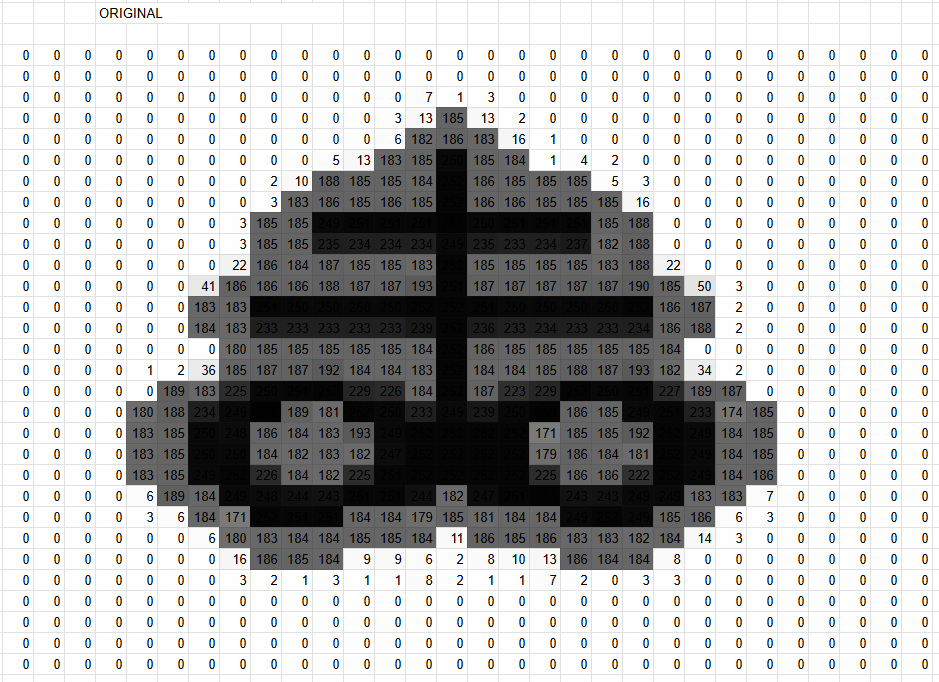
****

**Figura 4:** Tiempos generados colab

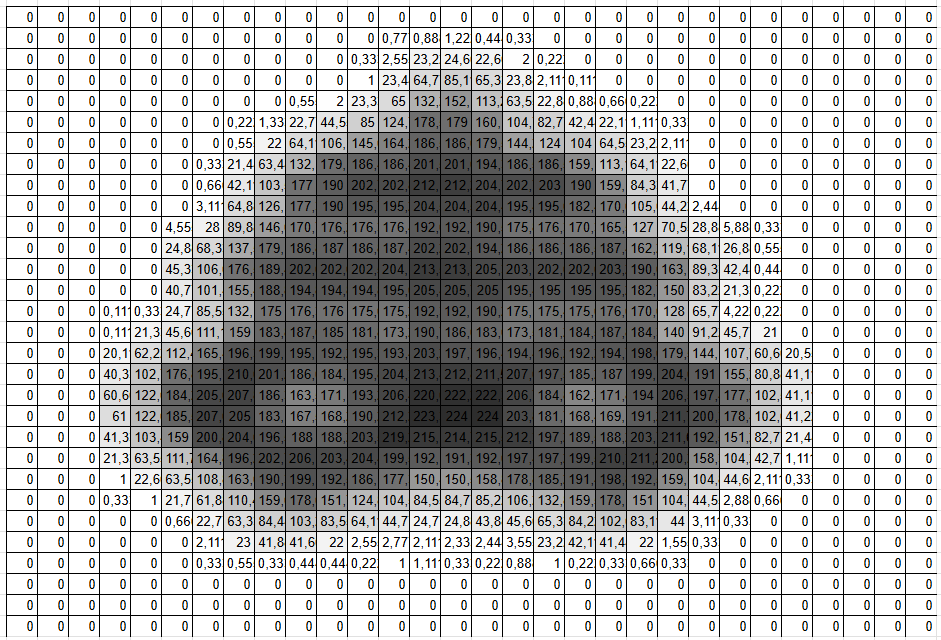
**Tabla 1.** Comparación de tiempos

| *Tiempo* | *Generación* | *Estadísticas* | *Lectura* |
| --- | --- | --- | --- |
| Local | 0.199 s | 0.071 s | - |
| Colab | - | 0.010 s | 0.146 s |

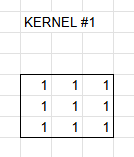
* En esta parte se presentan los resultados obtenidos en forma de tabla



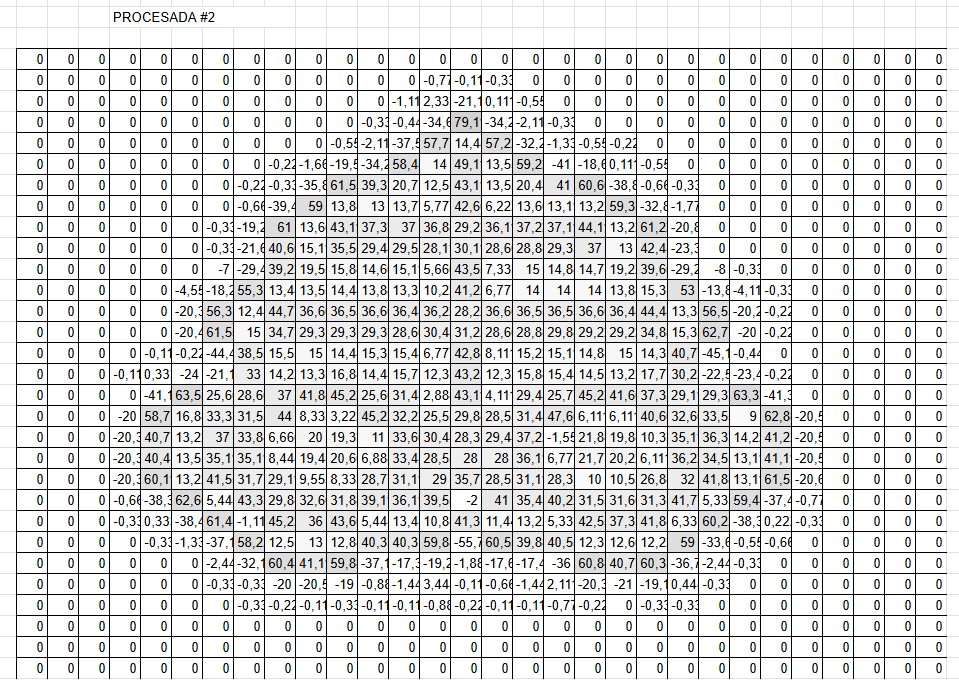
**Figura 5:** Original



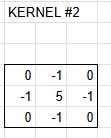
**Figura 5:** Procesada 1



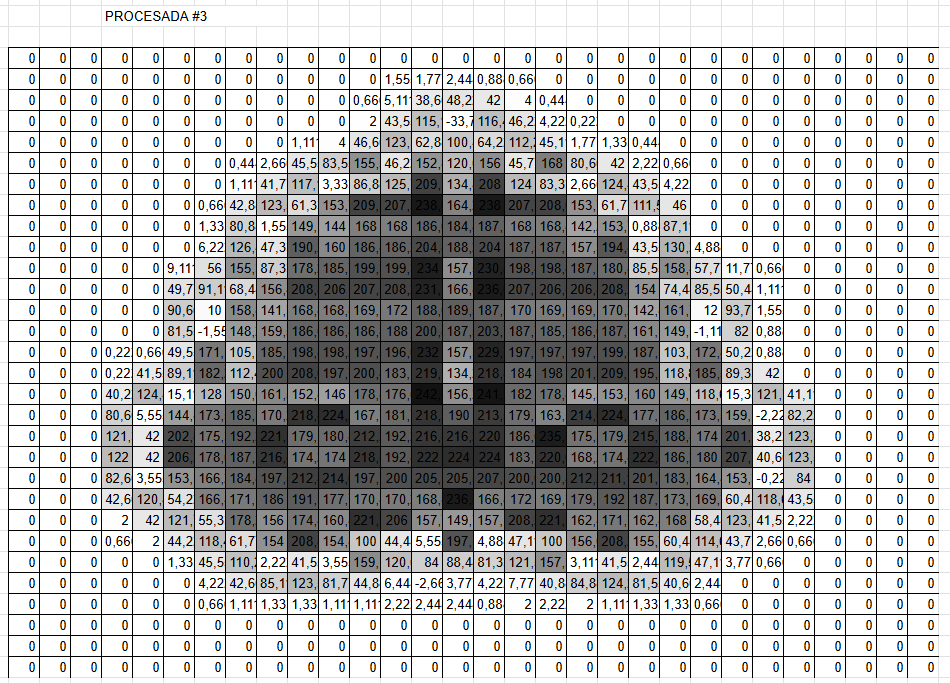
**Figura 6:** Kernel procesada 1



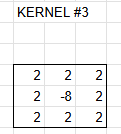
**Figura 7:** Procesada 2



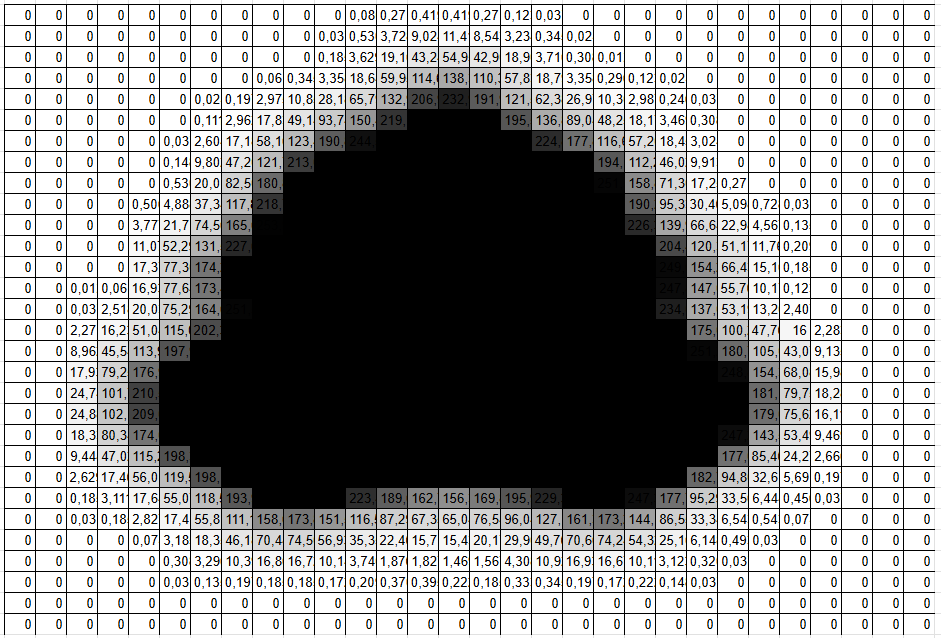
**Figura 8:** Kernel procesada 2



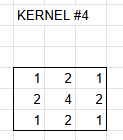
**Figura 8:** Procesada 3



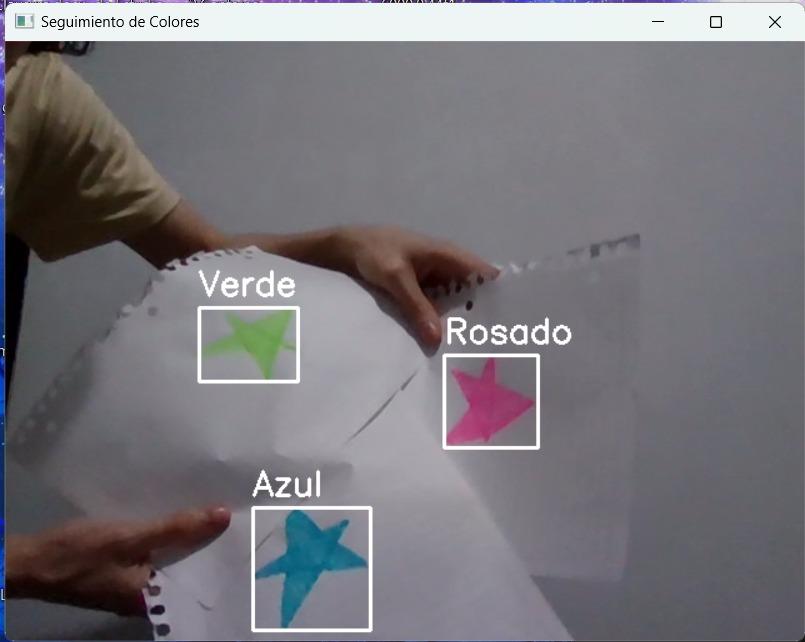
**Figura 10:** Kernel procesada 3

****

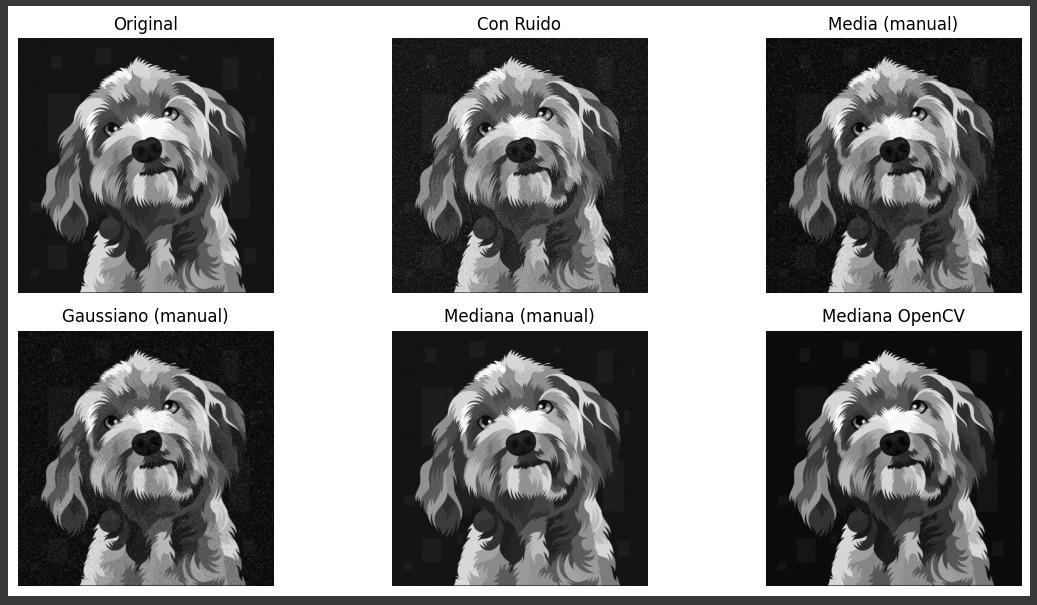
**Figura 11:** Procesada 4



**Figura 12:** Kernel procesada 4



**Figura 12:** Seguimiento de colores



**Figura 13:** Convoluciones para filtrado

El filtro de media suaviza la imagen pero deja restos del ruido.

El filtro gaussiano también suaviza, pero puede difuminar los bordes de la imagen.

El filtro mediana manual elimina eficazmente el ruido sal y pimienta sin afectar demasiado los bordes.

El filtro mediana de OpenCV muestra resultados casi idénticos al manual, validando la implementación.

Observaciones adicionales

Con valores de prob menores, el ruido es menos perceptible y los filtros parecen producir imágenes similares.

La mediana es la técnica más adecuada para ruido sal y pimienta, mientras que media y gaussiano funcionan mejor para ruido gaussiano.

**4. Discusión y conclusiones**

* Reto 1:
  + NumPy en Colab calcula estadísticas más rápido que la implementación manual local (0.010 s vs 0.071 s), como se puede observar en las figura 3,4.
  + En Colab, el costo dominante es la E/S: la lectura ocupa aproximadamente el 94% del tiempo total (0.146 s de 0.156 s), como se puede observar en la figura 4.
  + Localmente, el tiempo de generación (0.199 s) supera al de calcular estadísticas como se puede observar en la figura 3; las listas y bucles en Python añaden sobrecarga.
* Reto 2:
* Se realiza la procesada de imágenes con la ayuda de un kernel para esto se utiliza la fórmula (imagen 1\*kernel 1,1+ imagen2\*kernel 1,2+imagen 3\* kernel 1,3+imagen 4 \* kernel 2,1+ imagen 5\* kernel 2,2+ imagen 6\*kernel 2,3+imagen 7\*kernel 3,1+ imagen 8 \*kernel 3,2+imagen 9\*kernel 3,3)/9, Esta fórmula fue utilizada en los siguientes pares de figuras: (5 - 6), (7 - 8), (9-10), (11 - 12).
* Se considera que a número negativo más alto el resultado será menor
* Reto 4:
* Durante la implementación del algoritmo de seguimiento de colores en video se realizaron varias pruebas con figuras dibujadas en una hoja blanca, usando marcadores de diferentes colores (azul, verde y rosado).
* Detección del azul y verde: El sistema detectó de manera estable estas figuras, mostrando el recuadro y el texto sobre la imagen sin mayor dificultad. Esto se debe a que los rangos HSV para azul y verde están bien definidos y no suelen solaparse con el fondo blanco.
* Problemas con el rojo/rosado: Inicialmente, el marcador considerado como “rojo” era en realidad de tono rosado, lo que hizo que el rango estándar de rojo en HSV no fuera suficiente. Fue necesario ajustar los rangos a valores más cercanos al magenta para lograr una detección adecuada. Esto demuestra la importancia de calibrar los rangos HSV en función de las condiciones reales (tipo de marcador, iluminación, cámara).
* Velocidad y rendimiento: El procesamiento en tiempo real fue satisfactorio. La cámara pudo capturar y procesar los cuadros sin retardos evidentes, lo que permite un seguimiento fluido.
* El algoritmo es capaz de diferenciar y clasificar colores en tiempo real, mostrando tanto el recuadro de detección como el nombre del color sobre la figura.
* La elección de los rangos HSV es crítica: pequeños cambios en iluminación o en el tono del marcador requieren ajustes para mejorar la detección.
* La técnica es efectiva para colores bien diferenciados (azul, verde), pero puede ser menos precisa en colores cercanos en el espectro, como rojo y rosado.
* Este ejercicio muestra cómo la visión por computador puede aplicarse a sistemas de reconocimiento de objetos simples, y sienta las bases para aplicaciones más complejas como reconocimiento de múltiples objetos, clasificación avanzada o seguimiento de movimiento.
* Reto 5:
* El ruido sal y pimienta afecta significativamente la calidad visual de la imagen.
* Los filtros manuales permiten comprender el efecto de cada operación sobre los píxeles.
* La mediana es más eficiente para eliminar este tipo de ruido sin perder detalles.
* La función cv2.medianBlur de OpenCV es confiable y reproduce los mismos resultados que la implementación manual.
* Este ejercicio demuestra la importancia de seleccionar el filtro adecuado según el tipo de ruido presente en la imagen.