INSTITUTO POLITÉCNICO NACION AL

Escuela Superior de Cómputo

ESCOM

Practica 4:

❖ Creando mi mapa de calor

Realización: 20/10/2025

Entrega: 22/10/2025

Presentan:

❖ García Cerón Diego

Grupo: 7CV4

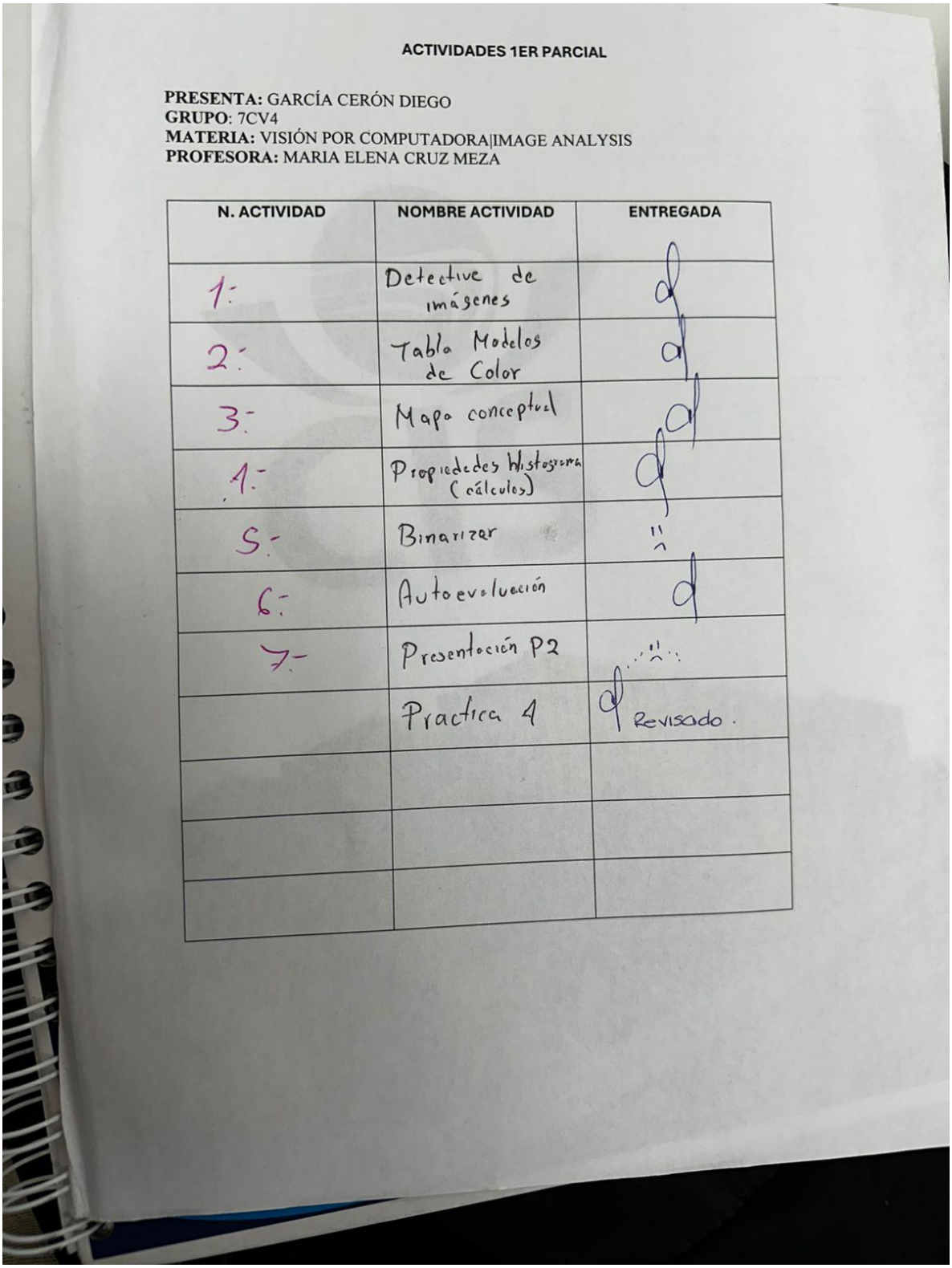
Materia:

VISIÓN POR COMPUTADORA|IMAGE ANALYSIS

Profesora:

MARIA ELENA CRUZ MEZA

FIRMA DE REVISION PRESENCIAL



ACTIVIDADES 1ER PARCIAL

PRESENTA: GARCÍA CERÓN DIEGO
GRUPO: 7CV4
MATERIA: VISIÓN POR COMPUTADORA|IMAGE ANALYSIS
PROFESORA: MARIA ELENA CRUZ MEZA

N. ACTIVIDAD	NOMBRE ACTIVIDAD	ENTREGADA
1:	Detective de imágenes	✓
2:	Tabla Modelos de Color	✓
3:	Mapa conceptual	✓
4:	Propiedades Histograma (cálculos)	✓
5:	Binarizar	:(
6:	Autoevaluación	✓
7:	Presentación P2	:(
	Practica 4	✓ Revisado.

Contenido

INTRODUCCIÓN

La presente práctica tiene como objetivo aplicar el concepto de pseudocolor en imágenes en escala de grises utilizando Python, combinando las bibliotecas OpenCV, Matplotlib y Tkinter. A través de una interfaz gráfica, se permite seleccionar imágenes y aplicar diferentes mapas de color —tanto predefinidos como personalizados— para mejorar su interpretación visual y comprender cómo la asignación artificial de colores puede resaltar detalles, contrastes y patrones relevantes en el procesamiento digital de imágenes.

OBJETIVOS

- Aplicar un mapa de color personalizado tipo pastel a una imagen en escala de grises usando Python, y explorar cómo los colores afectan la percepción visual de la información.

COMPETENCIAS

- Comprender el sistema de visión humano y modelos de color.
- Manipular imágenes digitales en Python.
- Colaborar en equipos para resolver problemas técnicos.

MARCO TEÓRICO

PSEUDOCOLOR

El pseudocolor es una técnica que consiste en asignar un color en imágenes monocromas, o colores artificiales a imágenes en escala de grises basándose en varias propiedades del contenido de nivel de gris de la imagen original.

Mediante la siguiente transformación se pueden asignar diferentes colores a una imagen en gris

$$a = a_{i-1} + \frac{a_i - a_{i-1}}{128}(p - g_{i-1})$$

$$donde: a = R, G, B, g_0 = 0 \; y \; g_i = 128$$

Esta técnica no busca recuperar los colores originales, sino facilitar la interpretación visual de los datos mediante la aplicación de mapas de color, en la Figura 1 podemos observar un ejemplo con el que se diseña una table de colores. Se divide en subintervalos de color, para el rango de niveles de color de [0-63] asigna rojo, de [64- 127] amarillo, de [128-191] azul y de [192-255] verde, respectivamente. La Figura 2 muestra el uso de la tabla para convertir imágenes en escala de grises en imágenes seudocolores. diferentes mapas de color (Figura b) aplicados a la imagen original de Lena (Figura a).

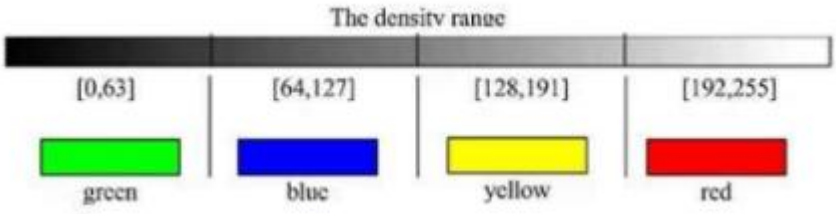


Figura 1. Ejemplo de asignación de bloques de color.

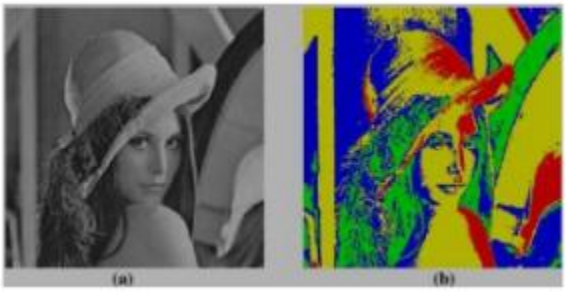


Figura 2. Imagen de Lena en escala de gris y sus correspondientes mapas de color [1].

¿QUÉ ES UN MAPA DE COLOR?

Un mapa de color (colormap) es una función que asigna colores RGB a cada valor de intensidad de una imagen en escala de grises. Por ejemplo, el mapa 'JET' asigna azul a los valores bajos, verde a los medios y rojo a los altos.

Aplicaciones del Pseudocolor

- Imágenes médicas (rayos X, resonancias)
- Imágenes satelitales
- Análisis térmico
- Visión por computadora
- Realce de características en imágenes científicas

DESARROLLO

PARTE A: OPERACIONES RELACIONALES Y LÓGICAS

Actividad 1. Práctica: "Aplicar pseudocolor a una imagen en escala de grises"

Se copio el código que se compartió en classroom solo se modificó para poder elegir la imagen que queremos cargar, para mayor comodidad.

El algoritmo tiene como propósito mejorar la interpretación visual de imágenes en escala de grises mediante la aplicación de diferentes mapas de color (pseudocolores). Esto permite resaltar variaciones

de intensidad que pueden ser difíciles de distinguir en tonos grises, facilitando el análisis visual o científico de la imagen.

1. **Selección de la imagen:**
    a. El programa utiliza un cuadro de diálogo gráfico (Tkinter) que permite al usuario seleccionar una imagen desde su sistema de archivos.
    b. Se aceptan formatos comunes como .jpg, .png, .bmp y .tiff.

```
# Abrir el cuadro de diálogo para seleccionar una imagen
ruta_imagen = filedialog.askopenfilename(
    title="Selecciona una imagen",
    filetypes=[("Archivos de imagen", "*.jpg *.jpeg *.png *.bmp *.tif *.tiff")]
)
```

Figura 3. Código selección de imagen

2. **Carga en escala de grises:**
    a. La imagen seleccionada se carga con OpenCV utilizando cv2.imread() con el parámetro cv2.IMREAD_GRAYSCALE.
    b. Este paso convierte la imagen original (RGB o BGR) en una matriz bidimensional donde cada valor representa un nivel de intensidad (0 = negro, 255 = blanco).

```
# Verificar si se seleccionó una imagen
if not ruta_imagen:
    print("No se seleccionó ninguna imagen.")
else:
    # Cargar la imagen en escala de grises
    imagen_gris = cv2.imread(ruta_imagen, cv2.IMREAD_GRAYSCALE)
```

Figura 4. Código Conversión a grises

3. **Aplicación de mapas de color (pseudocolor):**
    a. A la imagen en escala de gri ses se le aplican distintos colormaps mediante la función cv2.applyColorMap().
    b. Cada mapa traduce los valores de intensidad en un rango de colores perceptibles:
    c. COLORMAP_JET: Transición del azul al rojo pasando por verde y amarillo.
    d. COLORMAP_HOT: Transición de negro → rojo → amarillo → blanco.
    e. COLORMAP_OCEAN: Paleta basada en tonos azul-verde, simulando un entorno marino.

```
# Aplicar diferentes mapas de color (pseudocolor)
imagen_jet = cv2.applyColorMap(imagen_gris, cv2.COLORMAP_JET)
imagen_hot = cv2.applyColorMap(imagen_gris, cv2.COLORMAP_HOT)
imagen_ocean = cv2.applyColorMap(imagen_gris, cv2.COLORMAP_OCEAN)
```

Figura 5. Código mapas de colores basicos

4. **Visualización comparativa:**
    a. Se utiliza matplotlib.pyplot para mostrar en una cuadrícula las imágenes resultantes:
    b. La imagen original en escala de grises.
    c. Las tres versiones coloreadas con pseudocolor.
    d. Cada imagen se muestra sin ejes para una mejor apreciación visual.

```
# Mostrar las imágenes en una cuadrícula para comparación visual
fig, axs = plt.subplots(2, 2, figsize=(10, 8))
axs[0, 0].imshow(imagen_gris, cmap='gray')
axs[0, 0].set_title('Imagen en escala de grises')
axs[0, 1].imshow(cv2.cvtColor(imagen_jet, cv2.COLOR_BGR2RGB))
axs[0, 1].set_title('Pseudocolor: JET')
axs[1, 0].imshow(cv2.cvtColor(imagen_hot, cv2.COLOR_BGR2RGB))
axs[1, 0].set_title('Pseudocolor: HOT')
axs[1, 1].imshow(cv2.cvtColor(imagen_ocean, cv2.COLOR_BGR2RGB))
axs[1, 1].set_title('Pseudocolor: OCEAN')
```

Figura 6. Código vista grafica

5. **Presentación final:**
    a. Las imágenes se organizan en una figura de 2x2 usando plt.subplots().
    b. plt.tight_layout() ajusta automáticamente los márgenes para que no se superpongan los títulos ni los gráficos.
    c. Finalmente, plt.show() presenta el resultado al usuario.

Al cargar una imagen en grises de ejemplo en el código podemos ver el resultado siguiente:

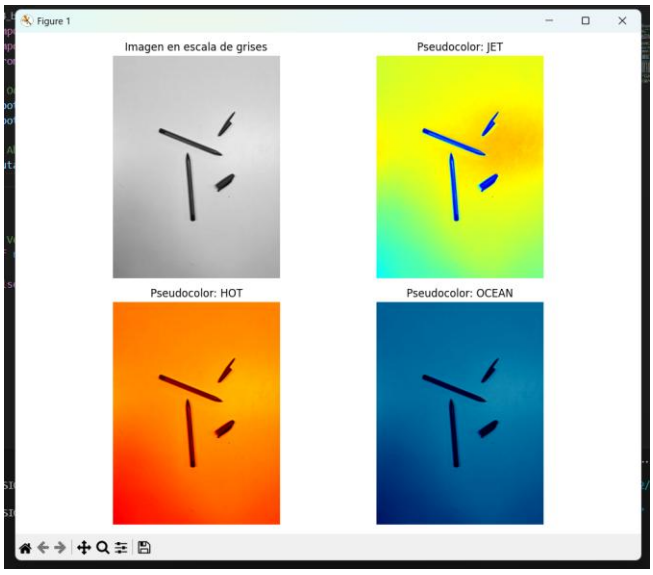


Figura 7. Aplicación de primeros mapas de color

Actividad 2. Práctica: "Aplicar más pseudocolor que ofrece OpenCv a una imagen en escala de grises"

Según OpenCv ofrece muchos mapas de colores que se pueden aplicar, por lo que en la tabla siguiente se aglomeran los mapas de color que se pueden aplicar gracias a OpenCv:

Nombre del mapa	Constante en OpenCV	Descripción visual
Jet	cv2.COLORMAP_JET	Azul → Verde → Amarillo → Rojo. Muy usado para visualización científica.
Hot	cv2.COLORMAP_HOT	Negro → Rojo → Amarillo → Blanco. Simula calor o energía térmica.
Ocean	cv2.COLORMAP_OCEAN	Azul → Verde oscuro → Azul claro. Aspecto marino.
Parula	cv2.COLORMAP_PARULA	Azul → Verde → Amarillo. Suave y agradable para datos científicos.
Rainbow	cv2.COLORMAP_RAINBOW	Arcoíris completo. Muy colorido.
HSV	cv2.COLORMAP_HSV	Tonos según matiz HSV (cíclico).
Autumn	cv2.COLORMAP_AUTUMN	Rojo → Amarillo (colores cálidos).
Bone	cv2.COLORMAP_BONE	Escala gris-azulada, imita rayos X.
Cool	cv2.COLORMAP_COOL	Cian → Magenta (colores fríos).
Pink	cv2.COLORMAP_PINK	Similar a "bone" pero con tonos rosados.
Spring	cv2.COLORMAP_SPRING	Magenta → Amarillo (alegre y cálido).
Summer	cv2.COLORMAP_SUMMER	Verde claro → Amarillo (fresco y suave).
Winter	cv2.COLORMAP_WINTER	Azul → Verde (colores fríos).
Copper	cv2.COLORMAP_COPPER	Marrón → Naranja → Blanco (metálico).
Inferno	cv2.COLORMAP_INFERNO	Paleta perceptualmente uniforme (de negro a rojo intenso).
Plasma	cv2.COLORMAP_PLASMA	Negro → Púrpura → Amarillo (muy contrastante).
Magma	cv2.COLORMAP_MAGMA	Negro → Rojo → Blanco (lava).
Cividis	cv2.COLORMAP_CIVIDIS	Similar a viridis, optimizado para daltonismo.
Viridis	cv2.COLORMAP_VIRIDIS	Azul → Verde → Amarillo, equilibrado y legible.
Twilight	cv2.COLORMAP_TWILIGHT	Tonos azulados y púrpuras.
Turbo	cv2.COLORMAP_TURBO	Alternativa moderna a "jet" con mejor uniformidad.

Tabla 1. Mapas de color disponibles en OpenCv

Modificando el programa anterior base se agregó la librería de tkinter para poder crear una interfaz gráfica y de esta forma poder seleccionar todos los mapas de color anteriormente descritos en la tabla, todas las opciones disponibles se muestran en una lista y se aplican a la imagen, los cuales son JET, HOT, OCEAN, PARULA, RAINBOW, HSV, AUTUMN, BONE, COOL, PINK, SPRING, SUMMER, WINTER, COPPER, INFERNO, PLASMA, MAGMA, CIVIDIS, VIRIDIS, TWILIGHT, TURBO.

Estos se cargan dinámicamente mediante:

```python
def build_available_colormaps():
    available = {}
    for nice_name, const_name in COLORMAP_NAMES.items():
        if hasattr(cv2, const_name):
            available[nice_name] = getattr(cv2, const_name)
    return available
```

Figura 8. Código carga dinámica de mapa de colores

Para esta segunda actividad es necesario buscar una imagen médica, por ejemplo, en mi caso aye de rayos x aplicados en un hombro y en un torso para aplicar los modelos de color,

Figura 9. Imagen medica original en grises.

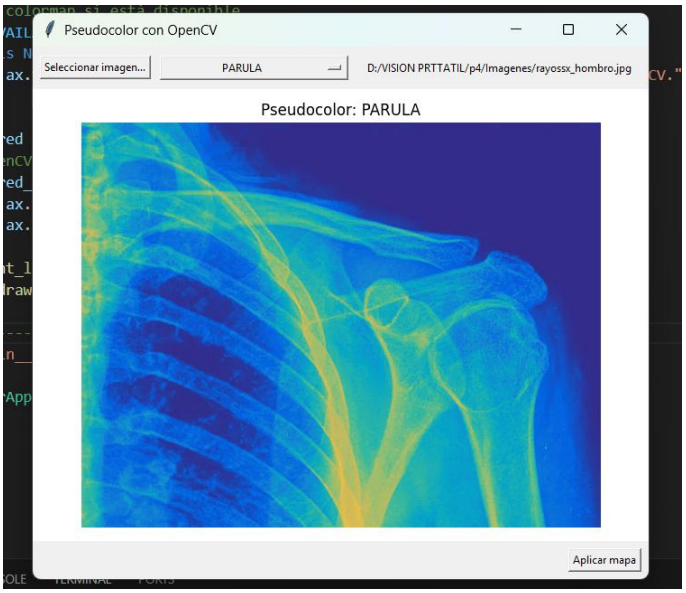


Figura 10. Imagen médica con modelo de color PARULA.

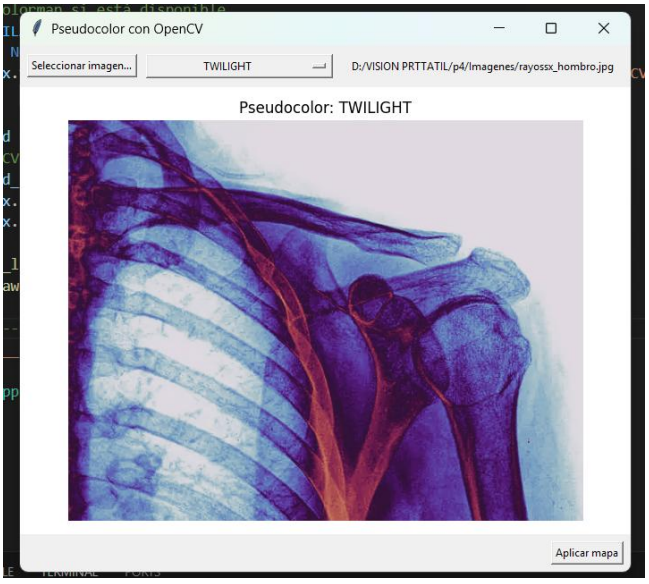


Figura 11. Imagen médica con modelo de color TWILIGHT.

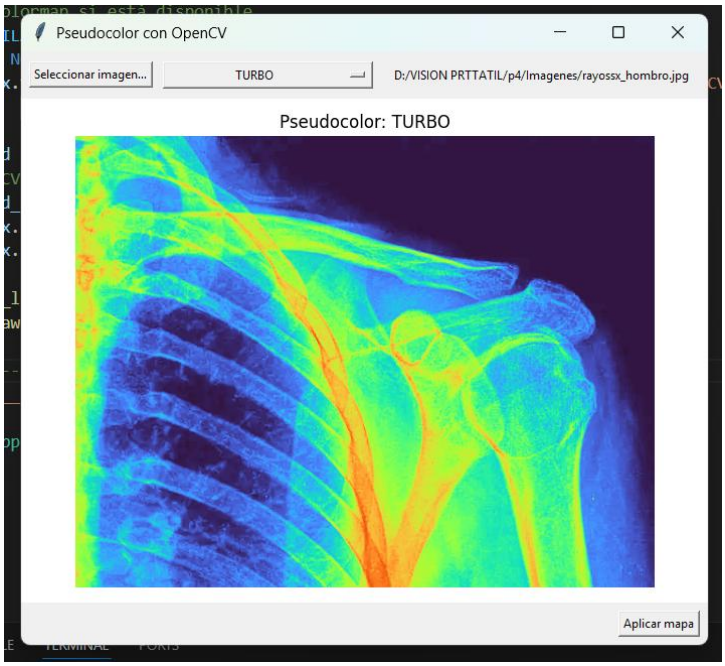


Figura 12. Imagen médica con modelo de color TURBO.

Para el siguiente paso se copió el código que anexo la profesora como ejemplo de como hacer un modelo custom, por lo que se aplicó dos modificaciones, se le aplico soporte para colormaps definidos manualmente, utilizando LinearSegmentedColormap de Matplotlib.

a) Mapa personalizado fijo ("Pastel")

```
colorespastel = [
    (1.0, 0.8, 0.9),  # rosa claro
    (0.8, 1.0, 0.8),  # verde menta
    (0.8, 0.9, 1.0),  # azul lavanda
    (1.0, 1.0, 0.8),  # amarillo suave
    (0.9, 0.8, 1.0)   # violeta claro
]
mapapastel = LinearSegmentedColormap.from_list("PastelMap", colorespastel, N=256)
```

Figura 13. Código mapa de color custom

b) Mapa personalizado aleatorio ("Random")

```python
def make_random_colormap(name="RandomMap", n_anchors=5, seed=None) -> LinearSegmentedColormap:
    """
    Genera un colormap aleatorio suave:
    - n_anchors: cantidad de puntos clave de color.
    - seed: semilla para reproducibilidad (None => diferente cada vez).
    """
    rng = np.random.default_rng(seed)
    # Colores aleatorios en [0,1]; forzamos primero y último para cubrir extremos
    anchors = np.linspace(0.0, 1.0, n_anchors)
    colors = rng.random((n_anchors, 3))
    # Opcional: asegurar contraste suficiente en extremos
    colors[0] = colors[0] * 0.2        # más oscuro al inicio
    colors[-1] = 0.8 + colors[-1]*0.2  # más claro al final
    # Empaquetar como lista (pos, color)
    seg = list(zip(anchors, colors))
    return LinearSegmentedColormap.from_list(name, seg, N=256)
```

Figura 14. Código mapa de color custom aleatorio

c) Conversión a tablas de búsqueda

Para aplicar los colormaps personalizados a las imágenes en grises, se implementó una conversión de cada colormap en una tabla de 256 valores RGB, usando:

```python
def cmap_to_lut(cmap: LinearSegmentedColormap) -> np.ndarray:
    """
    Convierte un Matplotlib colormap en una LUT (256x3) uint8.
    Cada fila i es el color RGB para la intensidad i en [0..255].
    """
    samples = np.linspace(0.0, 1.0, 256)
    rgba = cmap(samples)  # (256, 4)
    rgb = (rgba[:, :3] * 255.0).astype(np.uint8)  # (256, 3) en 0..255
    return rgb

CUSTOM_LUTS = {}  # se llena más abajo
```

Figura 15. Código conversión a tablas de búsqueda

Para hacer las comparativas sobre los mapas de colores aleatorios se usará otra imagen, la cual es la figura de rayos x de un torso:

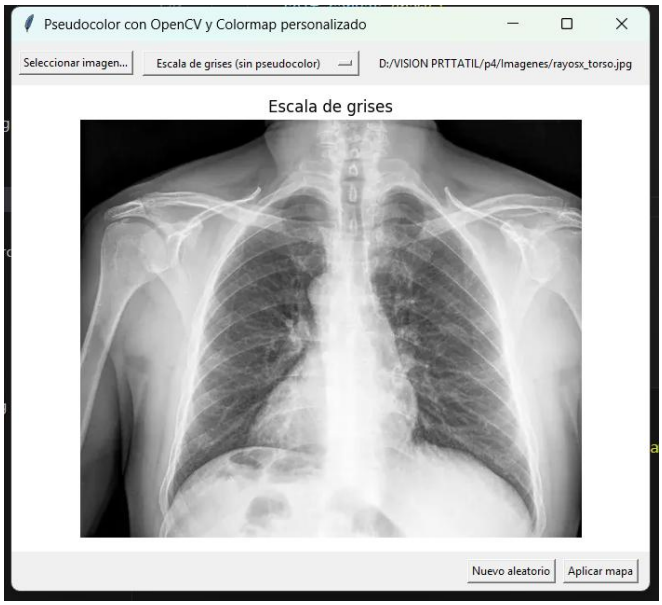


Figura 16. Imagen médica torso en grises.

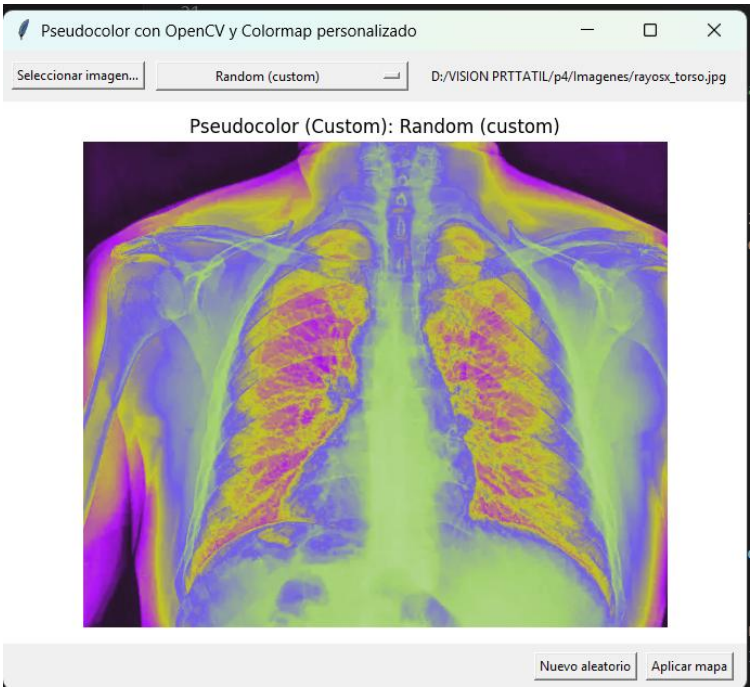


Figura 17. Imagen médica torso en mapa aleatorio

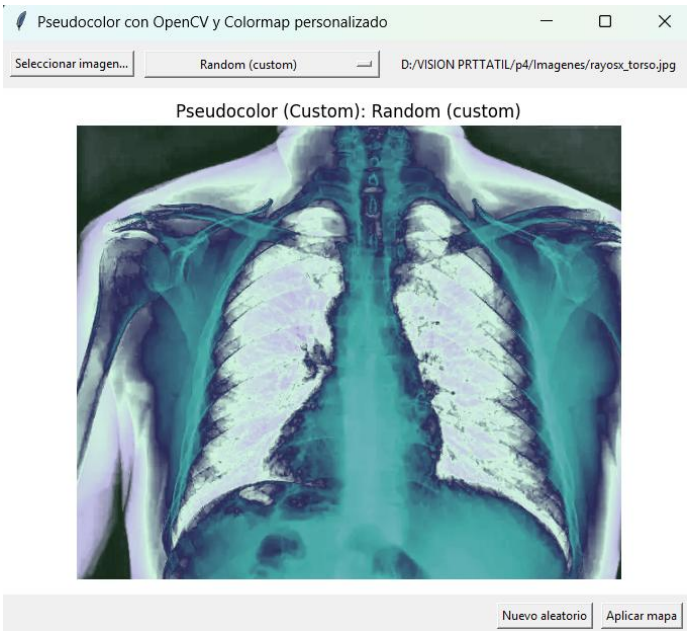


Figura 18. Imagen médica torso en mapa aleatorio

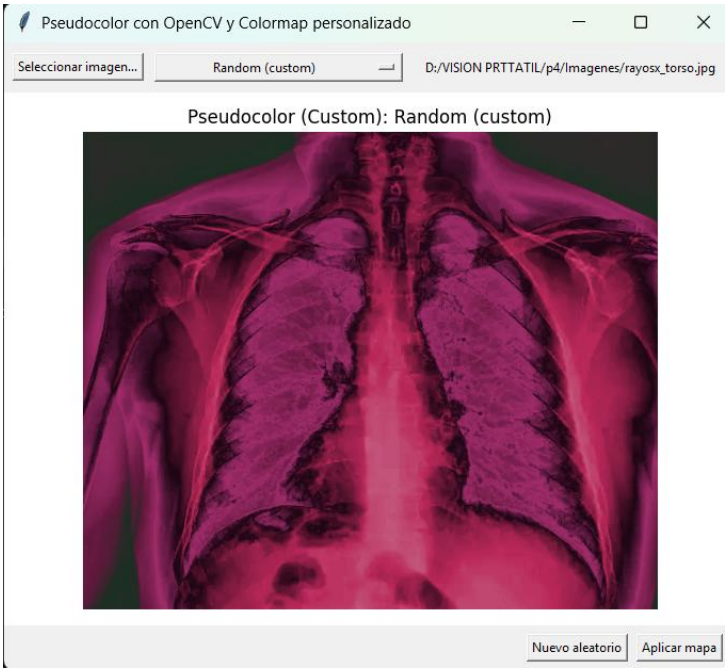


Figura 19. Imagen médica torso en mapa aleatorio

Comparativa entre los mapas de colores aplicados:

Versión	Descripción visual	Efecto principal	Ventajas	Desventajas / Riesgos	Uso recomendado

Escala de grises (original)	Imagen radiográfica natural, luminancia proporcional a la densidad del tejido.	Representa fielmente las intensidades originales.	Permite análisis clínico preciso; sin distorsión de valores.	Difícil diferenciar estructuras de densidad similar.	Diagnóstico o referencia base.
Pseudocolor 1 (Random – espectro cálido-frío)	Tonos violetas, amarillos y verdes; áreas densas en tonos fríos.	Incrementa contraste entre huesos y tejidos.	Realce de bordes y estructuras internas.	Colores pueden inducir interpretaciones erróneas.	Visualización exploratoria o educativa.
Pseudocolor 2 (Random – azul verdoso)	Gama azul-turquesa con zonas claras resaltadas en blanco.	Aspecto "radiográfico invertido" suave.	Mejora la diferenciación de zonas blandas.	Menor contraste en zonas de alta densidad.	Presentaciones visuales o demostraciones anatómicas.
Pseudocolor 3 (Random – magenta predominante)	Colores cálidos intensos; zonas óseas y densas en tonos oscuros.	Resalta estructuras óseas y sombras profundas.	Permite identificar claramente las zonas de mayor absorción.	Excesiva saturación; puede ocultar gradientes sutiles.	Enfatizar patrones óseos o contornos.

Tabla 2. Comparación de x-Ray con diferentes Mapas de color

Actividad 3. Autoevaluación

1) **¿Qué aprendí sobre el concepto de pseudocolor y su utilidad?**

Que el pseudocolor es una técnica de procesamiento digital que consiste en asignar colores artificiales a los niveles de intensidad de una imagen en escala de grises. Esta técnica no recupera los colores originales, sino que transforma las diferencias de brillo en diferencias de color, facilitando la interpretación visual de patrones, detalles o variaciones que no serían evidentes en tonos de gris. Es muy útil en análisis médico, imágenes satelitales y visión artificial, donde el contraste cromático mejora la comprensión de la información.

2) **¿Pude aplicar correctamente los mapas de color a diversas imágenes?**

Sí, pude aplicar correctamente los mapas de color predefinidos de OpenCV y también modelos personalizados. Verifiqué que cada mapa generaba un resultado distinto sobre la misma imagen,

modificando la percepción del contraste y destacando zonas de interés. Además, comprobé que el programa era capaz de procesar diferentes formatos de imagen (JPG, PNG, BMP, TIFF) sin errores.

3) ¿Qué aprendí de esta actividad práctica?

Aprendí a integrar bibliotecas de Python como **OpenCV**, **Matplotlib** y **Tkinter** para crear una herramienta interactiva capaz de visualizar imágenes con distintos esquemas de color. Comprendí cómo funcionan internamente los colormaps y cómo convertirlos en tablas de búsqueda (LUTs) para aplicarlos de manera eficiente. También reforcé mis conocimientos sobre la relación entre escala de grises e intensidad de píxeles.

4) ¿Qué decisiones tomé al diseñar mi propio mapa de color?

Decidí crear un colormap con tonos pastel suaves, buscando mantener una transición armónica de color sin saturación excesiva, ideal para observar detalles sutiles en imágenes. Posteriormente, añadí una versión aleatoria generada dinámicamente, con el objetivo de comparar resultados visuales y analizar cómo la elección de colores influye en la interpretación de una imagen.

5) ¿Qué dificultades encontré al modificar el código y cómo las resolví?

Tuve algunos errores al usar la función fromlist en lugar de from_list de LinearSegmentedColormap. Al revisar la documentación de Matplotlib, identifiqué la sintaxis correcta y lo solucioné sin necesidad de instalar librerías adicionales. También ajusté la compatibilidad entre los espacios de color BGR (OpenCV) y RGB (Matplotlib), aplicando conversiones antes de visualizar las imágenes para evitar que los colores se vieran alterados.

6) ¿Cómo mejoró la visualización de la imagen con mi mapa de color?

Mi mapa de color personalizado permitió una visualización más estética y suave, facilitando la identificación de zonas con diferentes niveles de intensidad sin generar distracciones visuales. En comparación con colormaps de alto contraste como JET o HOT, el colormap pastel ofreció una interpretación más equilibrada y agradable, ideal para análisis visuales donde el detalle es más importante que el contraste extremo.

7) ¿Qué aspectos podría mejorar en futuras actividades similares?

Podría agregar más controles en la interfaz para ajustar parámetros como el número de anclas de color, el brillo o el contraste del mapa. También sería interesante implementar la opción de guardar los resultados o comparar dos mapas simultáneamente en pantalla. Otra mejora posible sería permitir la edición manual de colores en tiempo real mediante una paleta interactiva.

8) ¿Cómo relaciono esta práctica con aplicaciones reales de la IA?

Esta práctica se relaciona directamente con la previsualización y análisis de datos visuales en modelos de IA. En visión por computadora, los pseudocolores se usan para representar mapas de calor, gradientes de activación o resultados de segmentación. Comprender cómo los valores de intensidad

se transforman en colores permite interpretar visualmente los resultados de redes neuronales y mejorar la explicación de decisiones en modelos de aprendizaje profundo.

Actividad Anexo con fotografía propia

Para hacer la prueba de los mapas de colores le tome una foto a mi credencial de la escuela ya que es una imagen limpia, que muestra mi rostro sin accesorios, en un fondo blanco y liso tal como lo piden en las especificaciones, lo único es la falta de brillo a la imagen. La imagen es la siguiente:

Figura 20. Imagen propia de rostro limpio.

Para las pruebas le coloque el colormap que ofrece opencv HCV, en donde se resaltan los contornos del rostro.

Figura 21. Imagen propia de rostro con HSV.

Después hice una prueba con un color custom el cual se genera aleatoriamente:

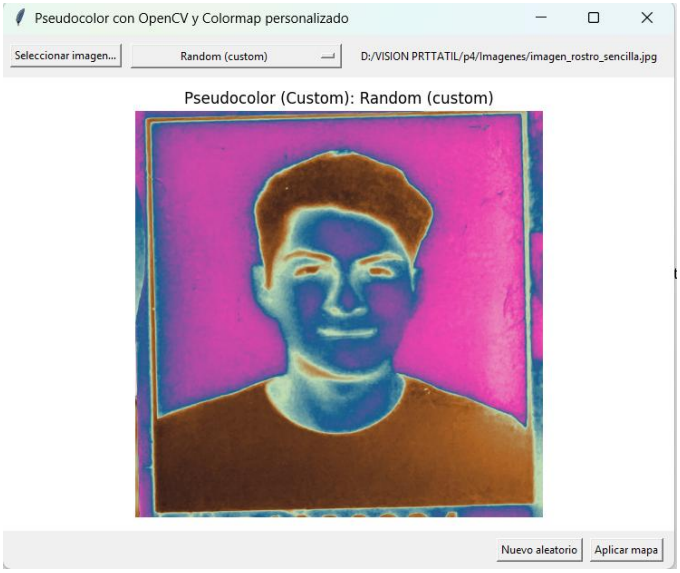


Figura 21. Imagen propia de rostro custom.

Tomando en cuenta ambas imágenes podemos hacer una comparativa la cual se muestra en la siguiente tabla:

Aspecto	Mapa HSV (OpenCV)	Mapa Random (Custom)
Tipo de gradiente	**Cíclico de tono (recorre todo el espectro visible)**	**Aleatorio, sin orden predecible**
Relación color–intensidad	**Continua, pero cíclica (vuelve a colores cálidos en valores altos y bajos)**	**No monótona; puede romper la relación entre intensidad y color**
Contraste visual	**Alto; resalta bien los bordes y las transiciones**	**Variable; depende del azar del mapa**
Interpretación de intensidades	**Intuitiva, aunque puede generar confusión por el ciclo de tono**	**Ambigua; requiere una leyenda para entender los valores**
Realce de bordes y detalles	**Muy bueno; produce una apariencia "térmica"**	**Inconsistente; puede perder microdetalles**
Uniformidad perceptual	**Media; puede introducir bandas o saltos abruptos**	**Baja; diferencias visuales no corresponden necesariamente a diferencias reales**
Aplicación ideal	**Análisis visual, detección de gradientes, mapas térmicos**	**Visualizaciones artísticas o experimentales**
Ventajas principales	**Colores llamativos y contrastes claros**	**Libertad estética, posibilidad de resaltar rangos personalizados**
Desventajas principales	**No perceptualmente uniforme; puede confundir valores altos/bajos**	**No confiable para interpretación cuantitativa sin leyenda**

Recomendación	**Útil para exploración rápida y resalte de estructuras**	**Solo para efectos visuales o énfasis personalizado**

Tabla 3. Comparación de rostro con diferentes Mapas de color

Actividad Anexo Análisis visual

- **¿Qué zonas del rostro aparecen más cálidas (más iluminadas)?**

En las imágenes pseudocoloreadas, las zonas más cálidas (tonos amarillos, rojos o magentas) corresponden a las regiones con mayor intensidad luminosa en la imagen original. En el rostro, estas suelen ser:

- Frente y pómulos, por ser superficies planas que reflejan más luz.
- Nariz y mentón, que suelen recibir directamente la iluminación frontal.
- En algunos casos, las mejillas si la luz incide lateralmente.

Estas áreas más claras representan valores altos en la escala de grises original, y el colormap los traduce en colores "cálidos", asociados visualmente con calor, energía o cercanía.

- **¿Qué relación tiene el concepto de iluminación en la toma de la foto?**

La iluminación determina directamente el rango tonal y de contraste de la imagen, lo que influye en cómo se distribuyen los valores de intensidad que el colormap transformará en colores.

- Una iluminación frontal y uniforme genera zonas amplias de alta intensidad → más áreas cálidas en el pseudocolor.
- Una iluminación lateral o con sombras marcadas produce contrastes fuertes → alternancia entre zonas frías y cálidas, resaltando volúmenes y relieves.

En síntesis: la calidad y dirección de la luz en la toma original condicionan la lectura cromática en el mapa pseudocoloreado. Una buena iluminación da un pseudocolor más expresivo y equilibrado.

- **¿Cómo afecta el mapa de color pastel a la percepción emocional de la imagen?**

Los colores pastel tienden a suavizar el impacto visual y transmitir una sensación de tranquilidad, calidez y suavidad emocional.

- Reducen el dramatismo y hacen que la imagen se perciba más amable o estética.
- Disminuyen el contraste percibido, lo que atenúa la intensidad emocional (en comparación con mapas de color más saturados o de alto contraste).
- En un contexto artístico o de retrato, el mapa pastel puede humanizar o "dulcificar" la percepción del rostro.

Desde la psicología del color, los tonos pastel favorecen interpretaciones más positivas y relajadas, mientras que los tonos fríos o saturados tienden a expresar tensión, fuerza o dramatismo.

CONCLUSIÓN GARCIA CERON DIEGO

En esta práctica logré comprender y aplicar el concepto de pseudocolor como una herramienta fundamental en el procesamiento digital de imágenes. A través de la implementación de distintos mapas de color, tanto predefinidos de OpenCV como personalizados creados con Matplotlib, pude observar cómo la asignación artificial de colores a una imagen en escala de grises mejora significativamente su interpretación visual.

El desarrollo de una interfaz gráfica con Tkinter permitió una interacción más intuitiva con los resultados, haciendo posible comparar en tiempo real diversos esquemas cromáticos y analizar su efecto sobre la percepción de los detalles. Además, la creación de colormaps personalizados —incluyendo una versión aleatoria— demostró la flexibilidad del método y la importancia del diseño cromático en la comunicación visual de la información.

En conclusión, esta actividad fortaleció mis conocimientos sobre la relación entre procesamiento de imágenes, percepción visual y herramientas computacionales, sentando una base sólida para futuras aplicaciones en áreas como visión por computadora, análisis médico e interpretación de resultados en modelos de inteligencia artificial.

REFERENCIAS

[1] Jiang, N., Wu, W., Wang, L. et al. Quantum image pseudocolor coding based on the density-stratified method. Quantum Inf Process 14, 1735–1755 (2015). https://doi.org/10.1007/s11128-015-0986-0

[2] “OpenCV: ColorMaps in OpenCV”. OpenCV documentation index. Accedido el 24 de octubre de 2025. [En línea]. Disponible: https://docs.opencv.org/4.x/d3/d50/group__imgproc__colormap.html