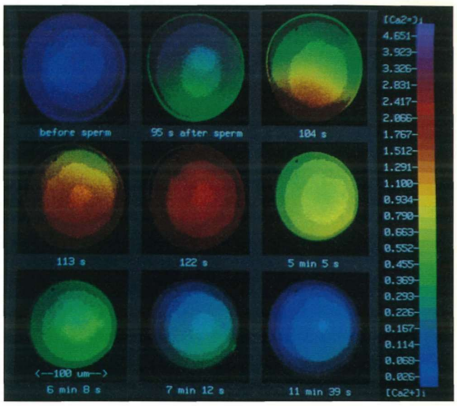

PSEUDOCOLOR

Práctica 2

DESCRIPCIÓN BREVE

Práctica para comprender el concepto de pseudocolor en PDI, identificar sus aplicaciones en distintos contextos, y aplicar mapas de color a imágenes en escala de grises mediante programación en Python, con el fin de mejorar la interpretación visual de datos.

Dra. María Elena Cruz Meza
Escom.IPN

Practica 1-Parte 1: Conversión de Imágenes en Escala de Grises a Pseudocolor con Python

1. Introducción al 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 divideen 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).

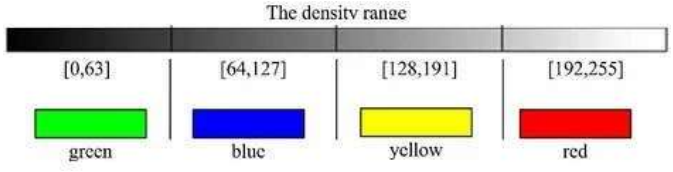

Figura 1. Ejemplo de asignación de bloques de color.

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

2. ¿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.

3. Aplicaciones del Pseudocolor

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

4. Actividades

Objetivo de Aprendizaje

Explicar el concepto de pseudocolor en procesamiento de imágenes, identificar sus aplicaciones en distintos contextos, y aplicar mapas de color a imágenes en escala de grises mediante programación en Python, con el fin de mejorar la interpretación visual de datos.

Objetivo específico:

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.

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

1. **Cargar una imagen en escala de grises**
2. **Aplicar diferentes mapas de colores**
3. **Visualizar los resultados**

Código en Python

A continuación se presenta un ejemplo de cómo aplicar pseudocolor a una imagen en escala de grises utilizando OpenCV y matplotlib:

```
import cv2
import matplotlib.pyplot as plt

# Cargar una imagen en escala de grises
imagen_gris = cv2.imread('imagen_gris.jpg', cv2.IMREAD_GRAYSCALE)
```

```
# 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)

# 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')

# Quitar los ejes para mejor visualización
for ax in axs.flat:
    ax.axis('off')

# Ajustar el diseño y mostrar la figura
plt.tight_layout()
plt.show()
```

Instrucciones para Ejecutar el Código

1. Instala las librerías necesarias: opencv-python, matplotlib.
2. Guarda una imagen en escala de grises como 'imagen_gris.jpg' en el mismo directorio del script.
3. Ejecuta el código en un entorno como *Jupyter Notebook* o un archivo *.py*.
4. Observa cómo se aplican diferentes mapas de color a la imagen original.

4.2 Actividad 2: Ejercicios (trabajo en horas autónomas)

1. Prueba otros mapas de color disponibles en OpenCV (por ejemplo: COLORMAP_BONE, COLORMAP_PINK).
2. Aplica pseudocolor a una imagen médica o térmica.
3. Crea tu propio mapa de color personalizado utilizando NumPy y aplicalo a una forografía tuya, para esto sigue las indicaciones de la Profesora.

Ejemplo: A continuación se te muestra cómo crear un mapa de color personalizado tipo pastel (Figura 2) en Python y aplicarlo a una imagen en escala de grises.

Figura 2. Ejemplos de mapa de color “pastel” y “arcoiris”.

¿Cómo funciona?

1. Se genera una imagen de prueba como un gradiente horizontal de valores entre 0 y 255.
2. Se define una lista de colores pastel en formato RGB normalizado.
3. Se crea el mapa de color con *LinearSegmentedColormap* de *matplotlib*.
4. Se aplica el mapa a la imagen en escala de grises para visualizar el pseudocolor.

Ejemplo (Fragmento clave del código):

```
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.fromlist("PastelMap", colorespastel,
N=256)
```

NOTA: Ver anexo.

4. Compara visualmente cómo diferentes mapas resaltan distintas características de la imagen.

4.3 Actividad 3: Autoevaluación

1) ¿Qué aprendí sobre el concepto de pseudocolor y su utilidad?
2) ¿Pude aplicar correctamente los mapas de color a diversas imágenes?
3) ¿Qué aprendí de esta actividad práctica?
4) ¿Qué decisiones tomé al diseñar mi propio mapa de color?
5) ¿Qué dificultades encontré al modificar el código y cómo las resolví?
6) ¿Cómo mejoró la visualización de la imagen con mi mapa de color?
7) ¿Qué aspectos podría mejorar en futuras actividades similares?
8) ¿Cómo relaciono esta práctica con aplicaciones reales de la IA?

5. Rúbrica de Evaluación

Tabla 1. Rúbrica de Evaluación - Pseudocolor en Imágenes

Criterio	Descripción	Excelente	Bueno	Regular / Deficiente
Comprensión teórica	Demuestra comprensión del concepto de pseudocolor y sus aplicaciones.	Explica claramente el concepto y da ejemplos relevantes.	Explica el concepto con algunos ejemplos.	Explicación incompleta o confusa.
Implementación en Python	Aplica correctamente el código para generar pseudocolor.	Código funcional y bien estructurado.	Código funcional con pequeños errores.	Código incompleto o con errores graves.
Análisis de resultados	Interpreta los resultados obtenidos con diferentes mapas de color.	Analiza y compara los resultados con claridad.	Analiza los resultados con observaciones básicas.	No interpreta los resultados o lo hace incorrectamente.
Presentación de resultados	Organiza y presenta los resultados de forma clara y visual.	Presentación clara, ordenada y visualmente atractiva.	Presentación clara pero con poco detalle visual.	Presentación desorganizada o poco clara.
Reflexión final	Relaciona la actividad con aplicaciones reales de la IA.	Reflexión profunda y bien argumentada.	Reflexión general con algunos ejemplos.	Reflexión superficial o ausente.

6. Fuentes consultadas

[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

ANEXO

A. Código completo en Python para crear y aplicar un mapa de color personalizado tipo pastel a una imagen en escala de grises.

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import LinearSegmentedColormap

# Este ejemplo permite crear una imagen en escala de grises como
gradiente horizontal (prueba usando otra imagen)
# Cada fila tiene valores de 0 a 255 en forma de gradiente
imagen_gris = np.tile(np.linspace(0, 255, 256), (100,
1)).astype(np.uint8)


# Definir colores pastel en formato RGB normalizado (valores entre 0 y
1)
colores_pastel = [
    (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
]

# Crear el mapa de color personalizado
mapa_pastel = LinearSegmentedColormap.from_list("PastelMap",
colores_pastel, N=256)

# Visualizar la imagen original y la imagen con pseudocolor pastel
fig, axs = plt.subplots(1, 2, figsize=(12, 4))

# Imagen en escala de grises
axs[0].imshow(imagen_gris, cmap='gray')
axs[0].set_title('Imagen en escala de grises')
axs[0].axis('off')

# Imagen con mapa de color pastel
axs[1].imshow(imagen_gris, cmap=mapa_pastel)
```

```
axs[1].set_title('Imagen con un mapa de color personalizado tipo
pastel')
axs[1].axis('off')

plt.tight_layout()
plt.show()
```

¿Qué puedes hacer con este código?

- Cambiar los colores pastel por otros tonos suaves.
- Aplicarlo a imágenes reales en escala de grises (por ejemplo, rayos X, mapas térmicos).
- Usarlo como base para que diseñes tu propio mapa de color.

B. Condiciones ideales para la toma de tu fotografía

Si vas a tomar una foto o alguien te va a tomar la foto (que incluya el rostro y el torso, como un retrato medio), sigue estas recomendaciones, que presenta una guía con las condiciones ideales para lograr una imagen adecuada con tu celular:

Iluminación

- **Luz natural suave**: Busca luz indirecta, como la que entra por una ventana o durante la "hora dorada" (al amanecer o al atardecer).
- **Evita sombras duras**: Si hay sol directo, usa una cortina o muévete a la sombra.
- **Luz frontal o lateral suave**: Ilumina el rostro sin crear sombras marcadas. Puedes usar una lámpara difusa si estás en interiores o busca una pared libre de objetos con pintura blanca mate y que la luz te ilumine por todo lados (luz indirecta frontal, laterales, arriba y abajo).

Composición

- **Altura de la cámara**: Colócala a la altura de los ojos para evitar distorsiones.
- **Regla de los tercios**: Activa la cuadrícula en tu cámara y coloca los ojos cerca de la línea superior.
- **Espacio alrededor**: Deja algo de aire por encima de la cabeza y a los lados para que no se vea encajonado.

Expresión y postura

- **Relajado pero atento**: La persona debe estar cómoda, con una expresión natural.
- **Hombros ligeramente girados**: Evita que esté completamente de frente, eso da más profundidad.

Técnica

- **Enfoca el rostro**: Toca la pantalla sobre los ojos para asegurar nitidez.
- **Evita el zoom digital**: Acércate físicamente si necesitas más detalle.
- **Limpia la lente**: Un paso simple que mejora mucho la calidad.

Fondo y ambiente

- **Fondo neutro o desenfocado**: Ayuda a que el rostro destaque.
- **Evita distracciones**: Retira objetos innecesarios detrás de la persona.

C. Guía para la práctica: Mapa de color pastel y mapa personanilzado en imágenes en escala de grises

Parte A: Código base (ya proporcionado)

Realiza las pruebas correspondientes con el código proporcionado y recommendable seguir los siguientes pasos:

- Crea una imagen de prueba en escala de grises.
- Define una paleta pastel en formato RGB normalizado.
- Aplica cada mapa de color usando LinearSegmentedColormap.
- Visualiza la imagen original y la transformada para cada ejemplo.

Parte B: Variaciones sugeridas

1. **Usar una imagen real**
   - Tomar una foto de tu elección.
   - Sustituye la imagen de prueba por una fotografía de tu rostro y torso.
   - Sustituir *imagen_gri*s por una imagen en escala de grises:

```
from PIL import Image
imagen_real = Image.open('ruta/a/tu/imagen.jpg').convert('L')
imagen_gris = np.array(imagen_real)
```

2. **Cambiar la paleta de colores**
   - Diseña una nueva lista de colores en formato RGB normalizado.
   - Prueba con el mapa de ejemplo (pastel) y luego con tonos tierra, fríos, cálidos, o monocromáticos.
   - Ejemplo de tonos tierra:

```
colores_tierra = [
    (0.6, 0.4, 0.2),
    (0.8, 0.7, 0.5),
    (0.9, 0.8, 0.6),
    (0.7, 0.5, 0.3),
    (0.5, 0.3, 0.1)
]
```

3. **Crear un mapa de calor facial**
   - Tomar una foto de tu rostro en color, luego conviertela en escala de grises.
   - Aplicar y luego modifica el mapa pastel para visualizar zonas de luz y sombra.
   - Ideal para conectar con los estilos de iluminación vistos en clase.
   - Visualiza cómo cambia la percepción de luz y sombra.

Parte C: Actividades complementarias

- **Comparar mapas de color**: ¿Cómo cambia la percepción con diferentes paletas?
- **Diseñar su propio colormap**: Cada alumno puede crear una paleta única.
- **Ejemplo: Analizar zonas cálidas/frías.-** Relacionar con iluminación fotográfica.

Herramientas útiles

- *matplotlib.colors.LinearSegmentedColormap*
- *PIL.Ima*ge para cargar imágenes
- *OpenCV* si quieren ir más allá con análisis facial o térmico.

Autoevaluación en la personalización del mapa de calor

Actividad: Análisis visual

Responde las siguientes preguntas:

- ¿Qué zonas del rostro aparecen más cálidas (más iluminadas)?
- ¿Qué relación tiene el concepto de iluminación en la toma de la foto?
- ¿Cómo afecta el mapa de color pastel a la percepción emocional de la imagen?

Parte D: Reflexión creativa

Actividad

Escribe una reflexión (basándote en las preguntas detonadoras).