**Taller de “Introducción al procesamiento digital de imágenes”**

**##### Práctica 01**

**# 1. Lectura de archivos**

import cv2 #carga la biblioteca OpenCV en Python

img = cv2.imread('imagen.jpg') #Lee imágenes.

cv2.imshow('Imagen', img) #Muestra la imagen img en una ventana.

cv2.waitKey(0) #Espera a que el usuario presione una tecla.

| Salida: |
| --- |

**# 2. Conversión de color**

# Esta instrucción sirve para convertir una imagen a escala de grises usando OpenCV.

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

cv2\_imshow(gray)

| Salida: |
| --- |

**# 3. Transformaciones geométricas**

img\_resized = cv2.resize(img, (200, 200)) #cambia el tamaño de la imagen ((ancho, alto) en píxeles)

img\_flipped = cv2.flip(img, 1) #Horizontal

| Salida img\_resized :  Salida img\_flipped: |
| --- |

### Explicación:

## cv2.resize() es usado comúnmente para:

## --> Ajustar imágenes para entrenar modelos de visión por computadora.

## --> Crear miniaturas o versiones más pequeñas.

## --> Normalizar tamaños antes de un procesamiento.

# cv2.flip() crea una copia de la imagen invertida y comúnmente se usa para:

## Aumentar el conjunto de datos en machine learning.

## Crear efectos de espejo.

## Corregir imágenes invertidas por la cámara.

#### El segundo argumento (1) indica volteo horizontal:

#### 0 → volteo vertical (arriba ↔ abajo).

#### 1 → volteo horizontal (izquierda ↔ derecha).

#### -1 → volteo en ambas direcciones.

**# 4. Operaciones aritméticas y lógicas**

### ajuste de brillo y el contraste

brighter = cv2.convertScaleAbs(img, alpha=1.2, beta=30)

| Salida: |
| --- |

### Explicación:

### **alpha=1.2**

### Factor de escala (controla el **contraste**).  
 ### 1 aumenta el contraste;  
 ### < 1 lo reduce.

### **beta=30**

### Valor que se suma a cada píxel (controla el **brillo**).  
 ### Positivo = más brillo;  
 ### Negativo = más oscuro.

### Aplica una transformación lineal a cada píxel de la imagen.

### Fórmula que usa internamente:

### nuevo\_valor = 𝛼 × valor\_original + 𝛽

### Luego toma el valor absoluto y lo convierte a uint8 (0 a 255).

**##### Práctica 02**

**## Parte 01**

# Librerias

from skimage import data

import cv2

import matplotlib.pyplot as plt

import requests

import numpy as np

### Explicación:

## cv2 (OpenCV) para procesamiento de imágenes.  
 ## matplotlib para mostrar imágenes.  
 ## requests para descargar imagen de internet.  
 ## numpy para manipulación de matrices.  
 ## skimage.data para cargar imágenes de prueba.

**## Parte 02**

# Leer imagen desde una URL

url = "https://raw.githubusercontent.com/opencv/opencv/master/samples/data/lena.jpg"

response = requests.get(url)

# Check if the request was successful

if response.status\_code == 200:

# Read the image from the downloaded content

image\_array = np.asarray(bytearray(response.content), dtype=np.uint8)

image = cv2.imdecode(image\_array, cv2.IMREAD\_COLOR)

# Convertir de BGR (OpenCV) a RGB (Matplotlib)

image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

plt.imshow(image\_rgb)

plt.title('Imagen de prueba: Lena')

plt.axis('off')

plt.show()

else:

print(f"Failed to download image from {url}. Status code: {response.status\_code}")

### Explicación:

### Descarga la imagen "Lena" desde internet, la decodifica con

### OpenCV, convierte de BGR a RGB para mostrarla correctamente y la ### muestra con matplotlib.

**## Parte 03**

# Separar los canales R, G, B

R, G, B = cv2.split(image\_rgb)

# Mostrar la imagen original y los 3 canales

fig, axs = plt.subplots(1, 4, figsize=(15,5))

axs[0].imshow(image\_rgb)

axs[0].set\_title("Imagen original")

axs[0].axis('off')

axs[1].imshow(R, cmap='Reds')

axs[1].set\_title("Canal Rojo (R)")

axs[1].axis('off')

axs[2].imshow(G, cmap='Greens')

axs[2].set\_title("Canal Verde (G)")

axs[2].axis('off')

axs[3].imshow(B, cmap='Blues')

axs[3].set\_title("Canal Azul (B)")

axs[3].axis('off')

plt.show()

### Explicación:

### Separa la imagen en sus tres canales de color (Rojo, Verde, Azul) y ### los muestra individualmente con un mapa de color acorde.

**## Parte 04**

grayscale = (0.299 \* R + 0.587 \* G + 0.114 \* B).astype(np.uint8)

# Mostrar la imagen original y la imagen en grises

fig, axs = plt.subplots(1, 2, figsize=(10, 5))

axs[0].imshow(image\_rgb)

axs[0].set\_title("Imagen Original (RGB)")

axs[0].axis('off')

axs[1].imshow(grayscale, cmap='gray')

axs[1].set\_title("Escala de Grises (Manual)")

axs[1].axis('off')

plt.show()

### Explicación:

### Convierte la imagen RGB a escala de grises con fórmula ponderada

### (más fiel a la percepción humana).

**## Parte 05**

print("Dimensiones de la imagen original:", image\_rgb.shape)

print("Dimensiones de la imagen en el canal R:", R.shape)

print("Dimensiones de la imagen en el canal G:", G.shape)

print("Dimensiones de la imagen en el canal B:", B.shape)

print("Dimensiones de la imagen en escala de grises:", grayscale.shape)

### Explicación:

### Muestra las dimensiones de la imagen original, canales y escala

### de grises.

**## Parte 06**

# Cargar imagen de prueba

image = data.camera() # Imagen en escala de grises (512x512)

# Mostrar

plt.imshow(image, cmap='gray')

plt.title('Imagen de prueba: Camera')

plt.axis('off')

plt.show()

### Explicación:

### Carga y muestra la imagen "camera" (escala de grises)

### incorporada en skimage.

**## Parte 07**

image

### Explicación:

### Mostrar el contenido de la imagen y sus dimensiones.

**## Parte 08**

image.shape

### Explicación:

### Mostrar el contenido de la imagen y sus dimensiones.

**## Parte 09**

# Crear cuadrado negro en la esquina superior derecha de 0 a 12 y mayor a 500

for i in range(image.shape[0]):

if i <= 12:

for j in range(image.shape[1]):

if j >= 500:

image[i,j] = 0 #se pinta de negro por que se pasa como valor 0, si quisieramos pintar de blanco se tendria que pasar 255

# Mostrar

plt.imshow(image, cmap='gray')

plt.title('Imagen de prueba: Camera')

plt.axis('off')

plt.show()

### Explicación:

### Pinta un rectángulo negro (valor 0) en la esquina superior derecha

### de la imagen.

**## Parte 10**

# Crear un array vacío para el histograma

histogram = np.zeros(256) # 256 niveles de gris (0-255)

# Contar la frecuencia de cada valor de gris

for i in range(image.shape[0]):

for j in range(image.shape[1]):

pixel\_value = image[i, j]

histogram[pixel\_value] += 1

# Mostrar la imagen en grises y su histograma

fig, axs = plt.subplots(1, 2, figsize=(12, 5))

# Imagen en grises

axs[0].imshow(image, cmap='gray')

axs[0].set\_title("Imagen en Escala de Grises")

axs[0].axis('off')

# Histograma

axs[1].bar(range(256), histogram, color='black')

axs[1].set\_title("Histograma")

axs[1].set\_xlabel("Valor de Gris")

axs[1].set\_ylabel("Frecuencia")

plt.show()

### Explicación:

### Calcula la frecuencia de cada nivel de gris en la imagen para

### hacer un histograma.

**## Parte 11**

# Inversión

inverted = np.zeros\_like(image)

for i in range(image.shape[0]):

for j in range(image.shape[1]):

inverted[i, j] = 255 - image[i, j]

# Mostrar la imagen en grises y su histograma

fig, axs = plt.subplots(1, 2, figsize=(12, 5))

# Imagen en grises

axs[0].imshow(image, cmap='gray')

axs[0].set\_title("Imagen en Escala de Grises")

axs[0].axis('off')

# Imagen invertida

axs[1].imshow(inverted, cmap='gray')

axs[1].set\_title("Imagen Invertida")

axs[1].axis('off')

plt.show()

### Explicación:

### Genera una imagen negativa (blanco ↔ negro invertidos).

**## Parte 12**

# Umbralización y binarización

threshold = 125 # Umbral

threshol2 = 100 # Umbral

bin = np.zeros\_like(image)

for i in range(image.shape[0]):

for j in range(image.shape[1]):

if image[i, j] > threshold:

bin[i, j] = 255

# elif image [i,j] > threshol2:

# bin[i, j] = 100

else:

bin[i, j] = 0

# Mostrar

fig, axs = plt.subplots(1, 2, figsize=(12, 5))

# Imagen en grises

axs[0].imshow(image, cmap='gray')

axs[0].set\_title("Imagen en Escala de Grises")

axs[0].axis('off')

# Imagen binarizada

axs[1].imshow(bin, cmap='gray')

axs[1].set\_title("Imagen binarizada")

axs[1].axis('off')

plt.show()

### Explicación:

### Convierte la imagen a binaria usando un umbral

### fijo: píxeles mayores a 125 se ponen blancos (255), el resto negros (0).