Tareas Tema 3

ALEJANDRO ALBERT GRAMAJE

INDICE

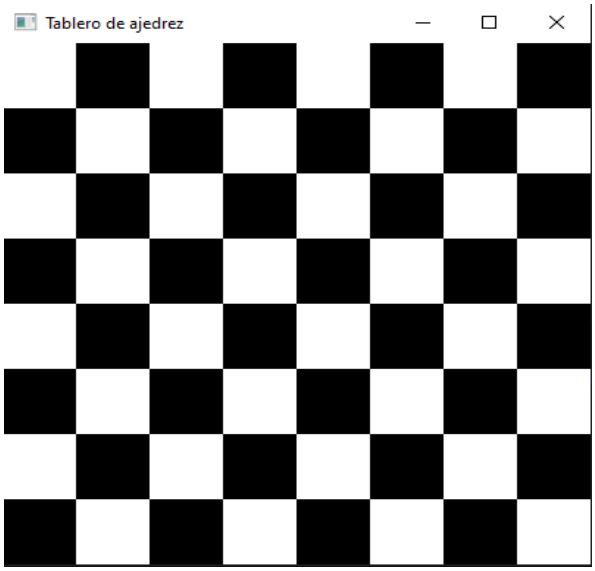
1. Tablero de ajedrez en blanco y negro

Este programa genera una imagen tipo tablero de ajedrez en escala de grises utilizando OpenCV y NumPy. Primero se crea una imagen negra del tamaño especificado por el usuario. Luego, se divide en un número de celdas igual al número de divisiones indicado. Se alternan celdas blancas y negras usando una fórmula sencilla (fila + col) % 2 para conseguir hacer el tablero de ajedrez.

Código:

```python
import cv2
import numpy as np

# Crea una imagen con patrón de ajedrez en blanco y negro
def crear_tablero(ancho, alto, divisiones):
    # Imagen inicial en negro (uint8 -> escala de grises)
    tablero = np.zeros((alto, ancho), dtype=np.uint8)
    # Tamaño de cada celda
    h_celda = alto // divisiones
    w_celda = ancho // divisiones

    for fila in range(divisiones):
        for col in range(divisiones):
            # Alternar blanco y negro
            if (fila + col) % 2 == 0:
                y1 = fila * h_celda
                x1 = col * w_celda
                tablero[y1:y1+h_celda, x1:x1+w_celda] = 255  # píxeles blancos

    return tablero

if __name__ == "__main__":
    # Pedimos al usuario dimensiones y número de casillas
    ancho = int(input("Ancho de la imagen: "))
    alto = int(input("Alto de la imagen: "))
    divisiones = int(input("Número de divisiones: "))

    img = crear_tablero(ancho, alto, divisiones)
    cv2.imshow("Tablero de ajedrez", img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Imagen Resultante:



Acciones por terminal:

```
PS C:\Clase\tercero\Vision> & C:/Clase/tercero/Vision/.venv/Scripts/
Ancho de la imagen: 400
Alto de la imagen: 400
Número de divisiones: 8
```

2. División en subimágenes

Este programa permite cargar una imagen cualquiera y dividirla en varias subimágenes. Se pide al usuario el número de filas y columnas en las que desea dividir la imagen, y se calcula el tamaño de cada recorte.

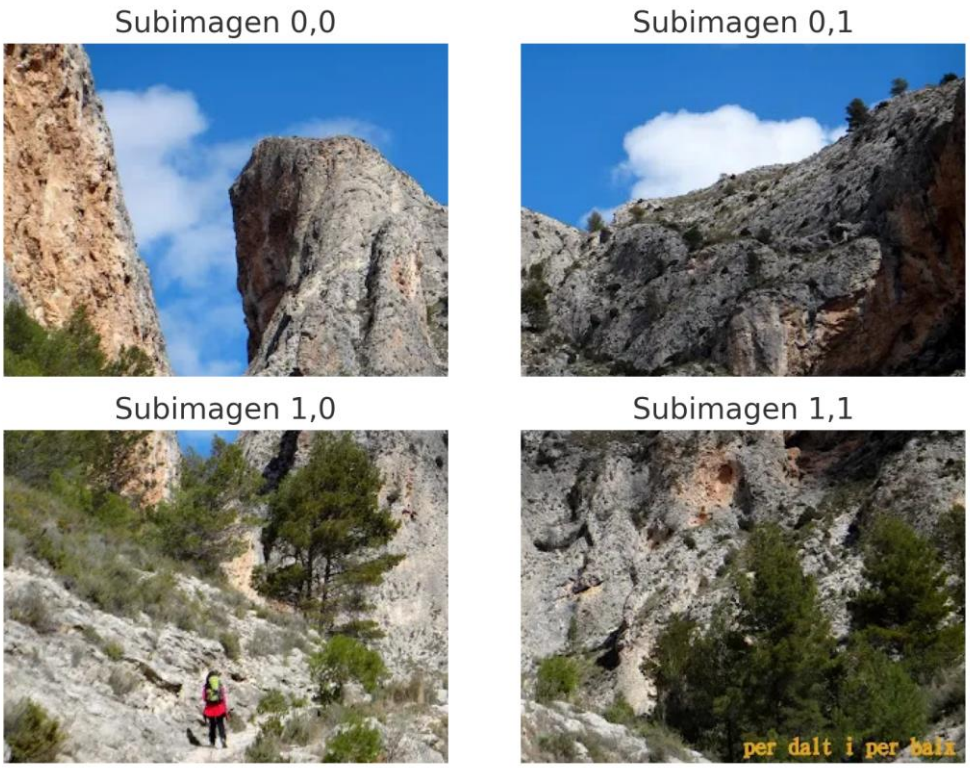
Cada subimagen se extrae usando índices con slicing de NumPy, se guarda automáticamente en una carpeta y se muestra por pantalla.

Fragmento de código:

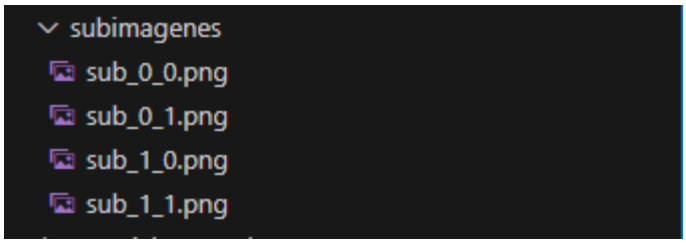
```python
import cv2
import os

# Divide una imagen en filas x columnas subimágenes y las guarda
def generar_subimagenes(ruta, filas, columnas):
    img = cv2.imread(ruta)
    if img is None:
        print("No se pudo cargar la imagen.")
        return

    h, w = img.shape[:2]
    h_corte = h // filas
    w_corte = w // columnas

    # Creamos carpeta si no existe
    os.makedirs("subimagenes", exist_ok=True)

    for i in range(filas):
        for j in range(columnas):
            # Cortamos la región correspondiente
            subimg = img[i*h_corte:(i+1)*h_corte, j*w_corte:(j+1)*w_corte]
            # Guardamos cada subimagen
            nombre = f"subimagenes/sub_{i}_{j}.png"
            cv2.imshow(f"Subimagen {i},{j}", subimg)
            cv2.imwrite(nombre, subimg)

    cv2.waitKey(0)
    cv2.destroyAllWindows()

if __name__ == "__main__":
    ruta = "Practicas/Practica_Tema_3/img/Foto_Barranc.png"
    filas = int(input("Número de filas: "))
    columnas = int(input("Número de columnas: "))
    generar_subimagenes(ruta, filas, columnas)
```

Imagen original:

Vista de ejemplo con 2x2 subimágenes:



Carpeta donde se guardan las imágenes:



3. Imagen con degradado

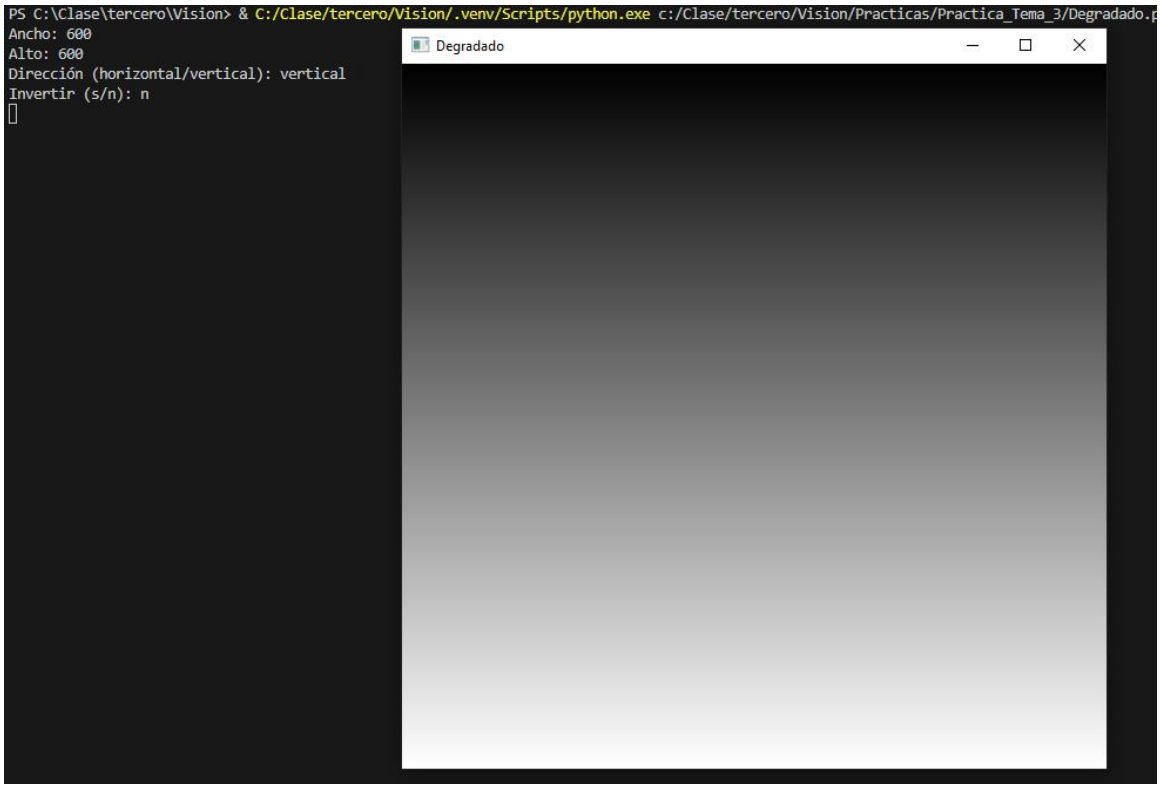
Este programa genera una imagen en escala de grises con un degradado lineal. El usuario puede elegir si quiere el degradado en dirección horizontal o vertical, y también si desea invertirlo (de blanco a negro o al revés).

Para construir el degradado se utiliza np.linspace para crear la transición de valores y np.tile para extenderla en la dirección adecuada

Fragmento de código

```python
import cv2
import numpy as np

# Genera un degradado horizontal o vertical, directo o invertido
def crear_degradado(ancho, alto, direccion="horizontal", inverso=False):
    if direccion == "horizontal":
        # Genera un vector de 0 a 255 a lo ancho
        grad = np.linspace(0, 255, ancho, dtype=np.uint8)
        if inverso:
            grad = grad[::-1]  # lo invertimos
        imagen = np.tile(grad, (alto, 1))  # copiamos la fila en todas las filas
    else:
        # Igual pero en vertical
        grad = np.linspace(0, 255, alto, dtype=np.uint8)
        if inverso:
            grad = grad[::-1]
        imagen = np.tile(grad[:, np.newaxis], (1, ancho))  # copiamos la columna en todas las columnas

    return imagen

if __name__ == "__main__":
    ancho = int(input("Ancho: "))
    alto = int(input("Alto: "))
    direccion = input("Dirección (horizontal/vertical): ")
    inverso = input("Invertir (s/n): ").lower() == "s"

    img = crear_degradado(ancho, alto, direccion, inverso)
    cv2.imshow("Degradado", img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Imagen resultante y uso por terminal:



4. Tablero de colores aleatorios

Este programa crea una imagen parecida al tablero de ajedrez anterior, pero en vez de blanco y negro, cada celda tiene un color aleatorio diferente.

La imagen se divide en bloques iguales y a cada uno se le asigna un color generado al azar. Esto se consigue generando tres valores RGB aleatorios para cada celda y pintándola con cv2 sobre la imagen inicial.

Fragmento de código:

```python
import cv2
import numpy as np

# Crea una imagen tipo tablero, pero con colores aleatorios en cada celda
def crear_tablero_color(ancho, alto, divisiones):
    tablero = np.zeros((alto, ancho, 3), dtype=np.uint8)  # 3 canales: BGR
    h_celda = alto // divisiones
    w_celda = ancho // divisiones

    for fila in range(divisiones):
        for col in range(divisiones):
            # Generamos un color aleatorio con 3 valores
            color = np.random.randint(0, 256, 3).tolist()
            y1 = fila * h_celda
            x1 = col * w_celda
            tablero[y1:y1+h_celda, x1:x1+w_celda] = color

    return tablero

if __name__ == "__main__":
    ancho = int(input("Ancho: "))
    alto = int(input("Alto: "))
    divisiones = int(input("Divisiones: "))

    img = crear_tablero_color(ancho, alto, divisiones)
    cv2.imshow("Tablero a color", img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Imagen resultante y uso por terminal