Practica 4 “filtro de 16 tonos” ODJL

Instrucción;

Elija una palera de oclores asociada a un olor base, por ejemplo: rojo, violeta, naranja, etc, y en base a ese color busque o defina una seria de 16 tonos. toamdno como referencia el ejemplo de filtro binario, contruya un codigo en python que aplique un filtro usando la pleta de 16 colores seleccionada. subir como evidencia, la imagen del volcan con el filtro aplicado, y el codigo generado.

a esta tecnica se le conoce como ESTENAGROFIA y se usa en espionaje.

Desarrollo.

Se tomo como color base el guinda 800040, para ello se escalo primero en los 16 tonos, tomando como punto medio la guinda seleccionada, de manera intuitiva, se define matemáticamente la separación de cada tono en 16 bites, de 128 colores / 8 canales = 16 segementos.

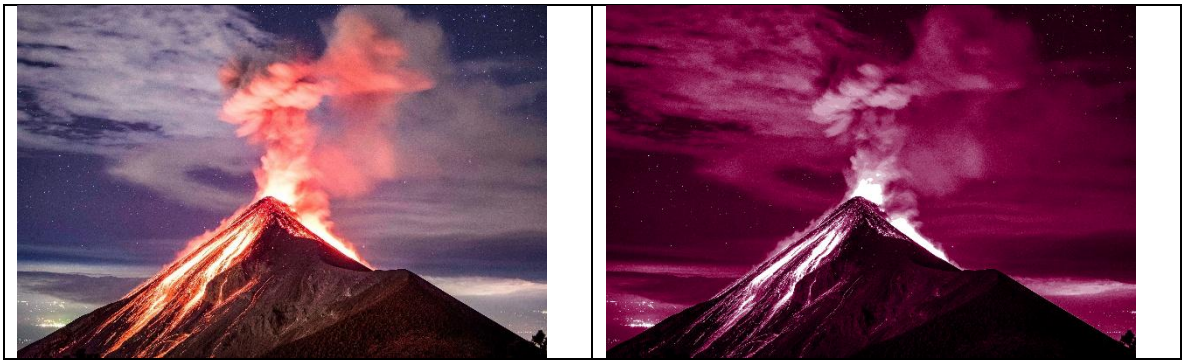
Se tomo el código de referencia de la practica anterior, pero se realizaron operaciones matemáticas para asignar las secciones de la gama original

```
import sys

# --- CONFIGURACIÓN DE LA PALETA GUINDA ---
# Base Hex: #800040 -> RGB(128, 0, 64) -> BGR(64, 0, 128)
# Hemos interpolado: Negro -> Guinda -> Blanco

# Paleta Guinda en Hexadecimal (Formato BGR: 0xAZUL, 0xVERDE, 0xROJO)
paleta_guinda_hex = [
    # --- Sombras (Negro a Guinda) ---
    [0x00, 0x00, 0x00],      # 0: Negro (0,0,0)
    [0x08, 0x00, 0x10],      # 1: (8,0,16)
    [0x10, 0x00, 0x20],      # 2: (16,0,32)
    [0x18, 0x00, 0x30],      # 3: (24,0,48)
    [0x20, 0x00, 0x40],      # 4: (32,0,64)
    [0x28, 0x00, 0x50],      # 5: (40,0,80)
    [0x30, 0x00, 0x60],      # 6: (48,0,96)
    [0x38, 0x00, 0x70],      # 7: (56,0,112)

    # --- Luces (Guinda a Blanco) ---
    [0x40, 0x00, 0x80],      # 8: BASE GUINDA (64,0,128) - Hex #400080 (en BGR)
    [0x58, 0x20, 0x90],      # 9: (88,32,144)
    [0x70, 0x40, 0xA0],      # 10: (112,64,160)
    [0x87, 0x60, 0xAF],      # 11: (135,96,175) -> Aprox 0x87
    [0x9F, 0x80, 0xBF],      # 12: (159,128,191)
    [0xB7, 0xA0, 0xCF],      # 13: (183,160,207)
    [0xCF, 0xC0, 0xDF],      # 14: (207,192,223)
    [0xFF, 0xFF, 0xFF]       # 15: Blanco (255,255,255)
]
try:
    print("Iniciando proceso de filtro Guinda...")

    # Abrir archivo original y crear destino
    file  = open('./images/volcan.bmp','rb')
    fileo = open('./images/volcan_guinda.bmp', 'wb')
```

```
    # 1. Copiar Metadata (Header 54 bytes)
    metadata = file.read(54)
    fileo.write(metadata)

    # 2. Procesar imagen píxel por píxel
    no_pix = 0

    while True:
        # Leer 3 bytes por píxel (Blue, Green, Red)
        pixel_data = file.read(3)

        if len(pixel_data) == 3:
            b = pixel_data[0]
            g = pixel_data[1]
            r = pixel_data[2]

            # Calcular Luminosidad (Grises)
            # Fórmula standard: 0.11B + 0.59G + 0.3R
            intensidad = int(0.114*b + 0.587*g + 0.299*r)

            # Escalar de 0-255 a 0-15
            indice = int((intensidad / 256) * 16)

            # Protección de límites (clamping)
            if indice > 15: indice = 15
            if indice < 0: indice = 0

            # Escribir el tono guinda correspondiente
            # Convertimos la lista [B,G,R] a bytes
            nuevo_pixel = bytes(paleta_guinda[indice])
            fileo.write(nuevo_pixel)

            no_pix += 1
        else:
            break

    print(f"¡Éxito! Imagen generada: 'volcan_guinda.bmp'")
    print(f"Total de píxeles modificados: {no_pix}")

    file.close()
    fileo.close()

except FileNotFoundError:
    print("Error: No se encuentra el archivo 'volcan.bmp'")
except Exception as e:
    print(f"Error inesperado: {e}")
```

*se hace la aclaración de la asistencia de l a IA Gemini para su desarrollo, se observa la prudente “printf” de cada proceso exitoso, para rastrear errores, se espera hacer una versión diferente, sin la asistencia de IA, para analizar el flujo de conversión de la imagen, buscando un algoritmo mas eficiente o de una lógica diferente, para diferentes imagenes

Evidencias; se obtuvo el siguiente resultado



Conclusión;

Es claro que el manejo de pixeles como fuente extra de información es interesante, pero podría carecer de información relevante, además de comprometer o ser detectable al momento de “ver” la imagen. Lo que me suguiere que técnicas mas sofisticadas para almacenar gran cantidad de información de manera oculta en un archivo cualquiera, por ello, la metadata no siempre es de fiar, ya que se podría corromper el contenido de la data mientras la metadata este correcta, el sistema podría ejecutar algo en la data que no debiera, por ello, un análisis previo que busque información de mas en el archivo es necesaria.

Me parece que los navegadores tienen esa seguridad para la descarga de contenido multimedia

Aun que esta imagen tiene información de texto, claramente no tiene sentido como contenido multimedia, lo que podría hacer que se deseche haciendo que su poroposito de transmitir la información (las instrucciones de la practica), sea difícil de obtener si no se tiene las herramientas o el contexto de la actividad