Evidencias del procedimiento para leer archivos BMP

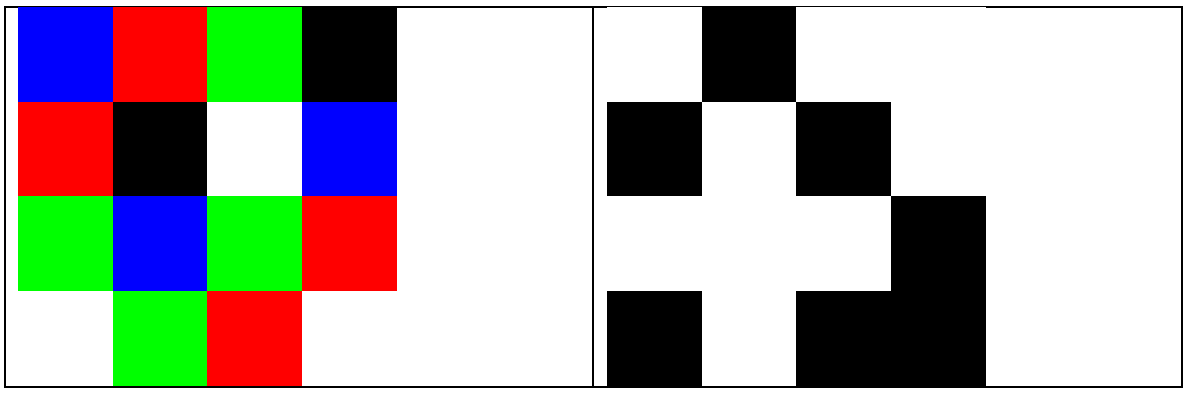
Se tomo la imagen ejemplo001 bmp, para realizarle un filtrafo binario, el usado el siguiente código

```
#Binary filter: Black and White

file = open('./images/example001.bmp','rb')
fileo = open('./images/example001bin.bmp','wb')
metadata = file.read(54)
fileo.write(metadata)
blanco = [0xff,0xff,0xff]
negro = [0x00,0x00,0x00]

file.seek(54,0)
no_pix = 0
limite = (pow(2, 24)-1)/2
while(True):
   pixel_data = file.read(3)
   if(len(pixel_data) > 0):
      valor_int = int.from_bytes(bytes(pixel_data),byteorder='little')
      if(valor_int<limite):
         fileo.write(bytes(blanco))
      else:
         fileo.write(bytes(negro))
      no_pix += 1
   else:
      break
print('No Pixels: '+str(no_pix))
file.close()
fileo.close()
```

Posterior mente se confirmo el cambio realizado, teniendo dos archivos en el mismo directorio



Seguidamente se realizado el rpcedmiento para la siguiente imagen “volcan.bmp” con el filtro blanco y negro

Procedimiento;

1.- abrir la herramienta web colab, con cuenta de Google

2.- crear un note book

3.- crear el directorio necesario en su árbol de dirección

4.- subir los archivos necesarios

5.-probar los códigos proporcionados, haciendo la documentación pertinente con herramienta de “texto” que el editor de código tiene

6.- revisar el resultado obtenido

Código propuesto *modificado

```
#Binary filter: Black and White version 2 archivo "volcan"

file = open('./images/volcan.bmp','rb')
fileo = open('./images/volcanbin.bmp','wb')
metadata = file.read(54)
fileo.write(metadata)
blanco = [0xff,0xff,0xff]
negro = [0x00,0x00,0x00]

file.seek(54,0)
no_pix = 0
limite = (pow(2, 24)-1)/2
while(True):
  pixel_data = file.read(3)
  if(len(pixel_data) > 0):
    valor_int =
int.from_bytes(bytes(pixel_data),byteorder='little')
    if(valor_int<limite):
      fileo.write(bytes(blanco))
    else:
      fileo.write(bytes(negro))
    no_pix += 1
  else:
    break
print('No Pixels: '+str(no_pix))
file.close()
fileo.close()
```

Después se realizó la operación inversa, de negro a blanco, que es el filtro inverso

Procedimiento

1.- se tomo la información anterior

2.- se modifico el código, cambiando de lugar las variables negro y blanco

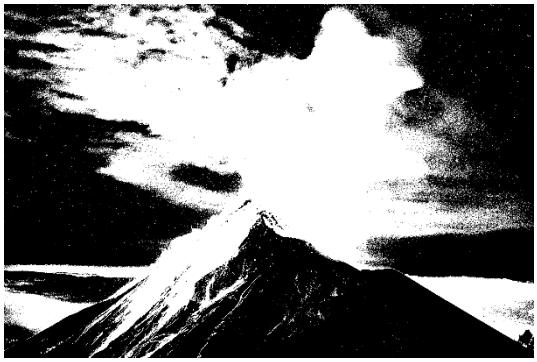
3.- se reviso el resultado

Codigo modificado* para filtro inverso

```
#Binary filter: White and Black version 2.1 archivo
"volcan"

file = open('./images/volcan.bmp','rb')
fileo = open('./images/volcanbin2.bmp','wb')
metadata = file.read(54)
fileo.write(metadata)
negro = [0xff,0xff,0xff]
blanco = [0x00,0x00,0x00]

file.seek(54,0)
no_pix = 0
limite = (pow(2, 24)-1)/2
while(True):
  pixel_data = file.read(3)
  if(len(pixel_data) > 0):
    valor_int =
int.from_bytes(bytes(pixel_data),byteorder='little')
    if(valor_int<limite):
      fileo.write(bytes(blanco))
    else:
      fileo.write(bytes(negro))
    no_pix += 1
  else:
    break
print('No Pixels: '+str(no_pix))
file.close()
fileo.close()
```

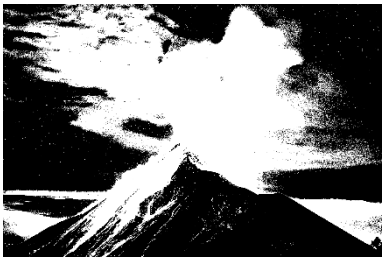


Comparación de datos obtenidos y conclusión

Se confirma el funcionamiento de los filtros, considerandolos entonces como editores de imágenes, con solo el uso del lenguaje python, se resalta su uso por la simplicidad, y se agradece el entorno de colab, ya que fue verdaderamente intuitivo las operaciones realizadas, tanto como invertir las variables de lugar, y obtener el proceso inverso.

Finalmente se presentan las cuestiones de si este algoritmo de edición, con sus respectivas adecuaciones funcionan con otros formatos de imagen, en especial las codificadas

¿se podrá hacer algo similar con audio y video?



DIRECTORIO FINAL