**REPORTE DE PRÁCTICA**

**IDENTIFICACIÓN DE LA PRÁCTICA**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Práctica** | **3** | **Nombre de la práctica** | | **Operaciones de pixel RGB** |
| **Fecha** |  | **Nombre del profesor** | | **Alma Nayeli Rodríguez Vázquez** |
| **Nombre del estudiante** | | | **Ricardo De León** | |

**OBJETIVO**

|  |
| --- |
| El objetivo de esta práctica consiste en procesar una imagen RGB mediante la modificación de los valores de intensidad de sus pixeles. |

**PROCEDIMIENTO**

|  |
| --- |
| Realiza la implementación siguiendo estas instrucciones. |
| Realiza un programa en Python utilizando OpenCV en el que leas una imagen desde archivo en formato RGB. Realiza las siguientes operaciones de pixel a la imagen:   1. Mayor y menor contraste 2. Mayor y menor brillo 3. Complemento 4. Combinación   Reporta los resultados obtenidos. |

**IMPLEMENTACIÓN**

|  |
| --- |
| Agrega el código de tu implementación aquí. |
| import cv2 as cv  import numpy as np  import matplotlib.pyplot as plt  img = cv.imread("viena.jpeg")  img2 = cv.imread("eiffel.jpeg")  cv.imshow("img", img)  # # contraste más, más cercano al 255 para eso se multiplica la imagen 'I' por '1' más un factor 'FC'.  # imgHC = img \* 1.5  # # Evitar desborde osea que se vean los negros como borrosos o brillosos  # np.putmask(imgHC, imgHC > 255, 255)  # # Se necesita el unit8 para que imshow lo despliegue bien  # imgHC = np.uint8(imgHC)  # cv.imshow("imgHC", imgHC)  # # contraste más, más cercano al 255 para eso se multiplica la imagen 'I' por un factor 'FC' menor que 1.  # imgLC = img \* .5  # # Se necesita el unit8 para que imshow lo despliegue bien  # imgLC = np.uint8(imgLC)  # cv.imshow("imgLC", imgLC)  # # Brillo  # # Más brillo  # # contraste más, más cercano al 255 para eso se sumas la imagen 'I' por un factor 'FC' mayor que 1.  # imgHB = np.double(img) + 100  # np.putmask(imgHB, imgHB > 255, 255)  # # Se necesita el unit8 para que imshow lo despliegue bien  # imgHB = np.uint8(imgHB)  # cv.imshow("imgHB", imgHB)  # # Brillo  # # Menos brillo  # # contraste más, más cercano al 255 para eso se menos la imagen 'I' por un factor 'FC' menor que 1.  # imgLB = np.double(img) - 100  # np.putmask(imgLB, imgLB < 0, 0)  # # Se necesita el unit8 para que imshow lo despliegue bien  # imgLB = np.uint8(imgLB)  # cv.imshow("imgHB", imgLB)  # # complemento  # img\_comp = 255 - img  # cv.imshow('img\_comp', img\_comp)  # Combinación  print(img.shape)  print(img2.shape)  img2 = cv.imread("eiffel.jpeg")  img = cv.resize(img,[750,500])  img\_comb = 0.3 \* img + 0.3 \* img2  img\_comb = np.uint8(img\_comb)  cv.imshow('img\_comb', img\_comb)  cv.waitKey() |

**RESULTADOS**

Agrega las imágenes indicadas en los espacios correspondientes.

|  |  |
| --- | --- |
| Imagen 1 RGB | Imagen 2 RGB |
|  |  |
| Imagen 1  Más contraste con fc = 1.5 | Imagen 1  Menos contraste con fc = 0.5 |
|  |  |
| Imagen 1  Más brillo con fb = 100 | Imagen 1  Menos brillo con fb = - 100 |
|  |  |
| Imagen 1  Complemento | Combinación de imagen 1 y 2 con w=0.5 |
|  |  |
| Combinación de imagen 1 y 2 con w=0.7 | Combinación de imagen 1 y 2 con w=0.3 |
|  |  |

**CONCLUSIONES**

|  |
| --- |
| Escribe tus observaciones y conclusiones. |
| Podemos observar la manipulación de imágenes como ls usamos normalmente en un celular ahora con el código en python sabiendo que parametros se tienen que mover para usar el contraste, brillo y combinación de imágenes. |