

Imágenes a color

...

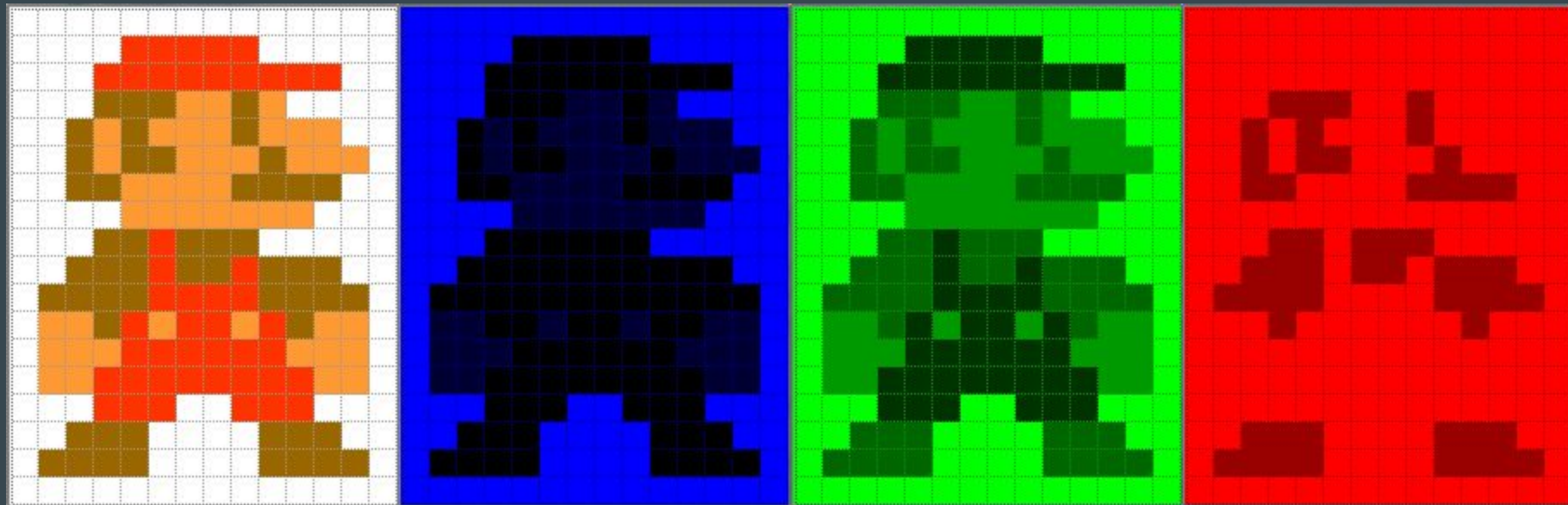
Python CV - Acceso al píxel

Observaciones:

- En python no es necesario definir el tipo de variable.
- Existe el retorno múltiple, por lo que funciones como .shape pueden retornar múltiples variables

```
1 import numpy as np
2 import cv2
3 # Lectura de imagen. El parámetro 0 indica que es en escala de grises
4 entrada = cv2.imread("Lena.png",0)
5 # De la imagen leída se extrae el alto y el ancho
6 alto, ancho = entrada.shape
7 # Se crea una imagen vacía para almacenar el resultado
8 salida = np.zeros((alto,ancho), np.uint8)
9 # Permite crear un ciclo infinito. Similar a LOOP de Arduino
10 while(True):
11
12     for i in range(0, alto): #Se recorre la imagen de 0 al alto
13         for j in range(0, ancho): #Se recorre la imagen de 0 al ancho
14             #Se lee la información del píxel en la posición i,j
15             f = entrada.item(i,j)
16             # Se resta 30 al valor original
17             g = (f-30)
18             #Se asigna el valor calculado, g, a la imagen nueva
19             salida.itemset((i,j),g)
20
21     #Se muestra la imagen de entrada en una ventana llamada original
22     cv2.imshow("Original",entrada)
23     cv2.imshow("Resultado",salida)
24
25     #Se espera a que se presione la tecla Q para finalizar
26     ch = 0xFF & cv2.waitKey()
27     if ch == ord('q'):
28
29         #Se almacena la imagen y se rompe el ciclo
30         cv2.imwrite("Resultado.jpg",salida)
31         break
32
33 #Se destruyen las ventanas creadas
34 cv2.destroyAllWindows()
```

Imágenes a color



255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	100	100	100	100	100	255	255	255	255	255
255	255	255	100	100	100	100	100	100	100	100	255	255	255
255	255	255	50	50	50	200	200	50	200	255	255	255	255
255	255	50	200	50	200	200	200	50	200	200	200	255	255
255	255	50	200	50	50	200	200	200	50	200	200	200	255
255	255	50	50	200	200	200	200	50	50	50	50	255	255
255	255	255	255	200	200	200	200	200	200	200	255	255	255
255	255	255	50	50	100	50	50	100	50	50	255	255	255
255	255	50	50	50	100	50	50	100	50	50	50	255	255
255	50	50	50	50	100	100	100	100	50	50	50	50	255
255	200	200	50	100	75	100	100	75	100	50	200	200	255
255	200	200	200	100	100	100	100	100	100	200	200	200	255
255	200	200	100	100	100	100	100	100	100	200	200	200	255
255	255	255	100	100	100	255	255	100	100	100	255	255	255
255	255	50	50	50	255	255	255	255	50	50	50	255	255
255	50	50	50	50	255	255	255	255	50	50	50	50	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255

R	G	B
255	205	148

R	G	B	X		0,95		=	R	G	B
255	205	148						242	195	141

R	G	B	X		0,5		+	R	G	B	X		0,5		=	R	G	B
255	205	148						213	166	189						234	186	169

R	G	B	+	R	G	B	=	R	G	B
75	200	50		200	75	75		255	225	75

Acceso a Píxel RGB

Observaciones:

- Canal 0 = b, canal 1 = g, canal 2 = r

```
1 | # -*- coding: utf-8 -*-
2 | import numpy as np
3 | import cv2
4 | # Lectura de imagen.
5 | entrada = cv2.imread("Lena.png")
6 | # De la imagen leída se extrae el alto, el ancho y los canales
7 | alto, ancho, canales = entrada.shape
8 | # Se crea una imagen vacía para almacenar el resultado
9 | salida = np.zeros((alto, ancho, 3), np.uint8)
10 | # Permite crear un ciclo infinito. Similar a LOOP de Arduino
11 | while(True):
12 |     for i in range(0, alto): #Se recorre la imagen de 0 al alto
13 |         for j in range(0, ancho): #Se recorre la imagen de 0 al ancho
14 |             #Se lee la información del píxel en la posición i,j de cada canal
15 |             b0 = entrada.item(i,j,0)
16 |             g0 = entrada.item(i,j,1)
17 |             r0 = entrada.item(i,j,2)
18 |             # Se incrementa el canal azul un 10% y se disminuye el rojo 50%
19 |             b1 = b0*1.1
20 |             g1 = g0
21 |             r1 = r0*0.5
22 |             #Se asigna cada valor calculado a la imagen nueva
23 |             salida.itemset((i,j,0),b1)
24 |             salida.itemset((i,j,1),g1)
25 |             salida.itemset((i,j,2),r1)
26 |             #Se muestra la imagen de entrada en una ventana llamada original.
27 |             cv2.imshow("Original", entrada)
28 |             cv2.imshow("Resultado", salida)
29 |             #Se espera a que se presione la tecla Q para finalizar
30 |             ch = 0xFF & cv2.waitKey()
31 |             if ch == ord('q'):
32 |                 #Se almacena la imagen y se rompe el ciclo
33 |                 cv2.imwrite("Resultado.jpg", salida)
34 |                 break
35 | #Se destruyen las ventanas creadas
36 | cv2.destroyAllWindows()
```

Desafíos

1. Acceder a cada píxel de una imagen RGB, sumar el valor de cada píxel de cada canal y dividirlo entre 3 (promedio), el resultado almacenarlo en una imagen vacía de un único canal. Concluir del resultado
2. Implementar el desafío 3 (de la clase anterior) en una imagen en escala de grises de bajo contraste
3. Implementar el desafío 3 (de la clase anterior) en una imagen a color (a cada canal)
4. Implementar el desafío 3 (de la clase anterior) en una imagen a color con base en el promedio

Nota: el máximo y el mínimo deben ser calculados con condicionales y ciclos no con funciones propias de Python, Numpy u OpenCV. Las formas adecuadas serán discutidas en la próxima sesión

Ayuda - Ejemplo para mínimo en un arreglo

```
x = [5,3,9,2]
```

```
minimo = 1000
```

```
for i in range(0,4):
```

```
    if(x[i] < minimo):
```

```
        minimo = x[i]
```


Ejemplo resultado desafío 4

