



**Universidad Autónoma Chapingo**

**Departamento de Mecánica Agrícola**  
**Ingeniería Mecatrónica Agrícola**

**Informe de prácticas B**

**Asignatura:**

**Visión por computadora**

**Nombre del profesor:**

**Luis Arturo Soriano Avendaño**

**Alumno:**

**Cocotle Lara Jym Emmanuel**  
**[1710451-3]**

**GRADO:**

**6°**

**GRUPO:**

**7**

Chapingo, Texcoco Edo. México

**Fecha de entrega: 30/05/2021**

## Índice

Introducción .....	3
Desarrollo.....	3
Practica 5.- Manejo de interacciones.....	3
Objetivos: .....	3
Código: .....	4
Resultados .....	7
Practica 6.- OpenCV propiedades de imágenes .....	7
Objetivos: .....	7
Código: .....	8
Resultados: .....	8
Practica 7.- OpenCV descomposición en canales de imágenes .....	9
Objetivos: .....	9
Código: .....	9
Resultados: .....	11
Practica 8.- OpenCV escala de grises.....	14
Objetivos: .....	14
Código: .....	14
Resultados: .....	15
Practica 9.- OpenCV generación de imágenes .....	15
Objetivos: .....	15
Código: .....	15
Resultados: .....	17
Practica 10.- OpenCV modificación de pixeles.....	17
Objetivos: .....	17
Código: .....	18
Resultados: .....	18
Conclusión .....	19
Bibliografía:.....	20

## Introducción

OpenCV es una librería Open Source de Visión por computadora que nos permite manipular imágenes y videos para realizar una variedad de tareas que van desde la detección automática de caras, a la visualización de las imágenes de una cámara Web; o hasta enseñarle a un robot a reconocer objetos en la vida real. Fue creada en 1999, por Gary Bradski, quien trabajaba en Intel, y que lanzó OpenCV con la intención de acelerar la Visión por computadora y la Inteligencia Artificial proporcionando una infraestructura sólida para todos los que trabajan en el campo. La librería está escrita en C y C++ y se puede ejecutar bajo Linux, Windows y Mac OS X. Posee un desarrollo activo de interfaces para Python, Java, MATLAB y otros lenguajes, incluyendo el soporte para plataformas como Android e iOS para aplicaciones móviles.

OpenCV contiene más de 500 funciones que abarcan muchas áreas de Visión por computadora, incluyendo tales como: inspección de productos de fábrica, imágenes médicas, seguridad, interfaz de usuario, calibración de cámara, visión estéreo y robótica. Debido a que la Visión por computadora y el Machine Learning a menudo van de la mano, OpenCV también contiene una librería completa de uso general de Machine Learning (ML); la cual se centra en el reconocimiento de patrones estadísticos y el agrupamiento. OpenCV es sin dudas el lugar por dónde comenzar para empezar a incursionar en el mundo de la Visión por computadora [1].

Con ayuda de la librería de OpenCV se puede hacer una gran variedad de operaciones para poder manipular una imagen de acuerdo con lo que se busque, y en esta práctica se busca hacer uso de esta librería para conocer algunas de sus funciones.

## Desarrollo

En este informe de prácticas se mostrarán los objetivos de cada práctica, así como el código que se realizó y los resultados que arrojó dicho código.

Para la realización de estas prácticas se ocupó:

- Computadora
- Software Sublime Text

### Practica 5.- Manejo de interacciones

Objetivos:

- Conocer el funcionamiento de los ciclos for y while.

- Aplicar los ciclos for y while.
- Conocer las diferentes formas de aplicación de los ciclos for.

Código:

```

1 # Jym Emmanuel Cocotle Lara
2 # Manejo de iteraciones en Python
3 n = 0 # Declaramos una variable y la inicializamos en 0
4
5
6 # Manejamos el incremento de una variable y los mostramos
7 for i in range(0, 5): # Declarando el ciclo for (índice (i),
rango(0.5), incremento)
8     n = n+1 # se suma 1 a la variable n durante el ciclo for
9     print(n) # Visualizamos los valores de la variable n
10
11
12 # Mostrar el valor del índice
13 for i in range(0, 5): # El ciclo for se ejecuta hasta que la
variable i sea igual a 5
14     print(i) # Visualizamos los valores de la variable i
15
16
17 # Manejo de listas con for
18 lista = [2, 3, 1, 4, 5, 7, 6] # Declaramos una lista con diferentes
valores
19
20
21 for j in lista: # Con ayuda de la variable j y con un ciclo for
recorremos los valores de la lista
22     print(j) # Visualizamos los valores de la lista
23
24
25 # Manejo de los valores de la lista como índices
26 lista_nueva = [2, 3, 1, 4, 5, 7, 6, 8] # Declaramos una lista con
diferentes valores
27 k = 0 # Declaramos la variable k y lo igualamos con 0
28
29
30 for i in lista: # Con ayuda de la variable i y con un ciclo for
recorremos los valores de la lista
31     k = k+i # Se suma 1 a la variable k durante el ciclo for
32     print(k) # Visualizamos los valores de la variable k
33
34
35 # Manejo de dos listas para crear una lista

```

```

36 lista_a = ["one", "two", "three"] # Declaramos una primera lista con
valores
37 lista_b = [1, 2, 3] # Declaramos una segunda lista con valores
38
39
40 for i, j in zip(lista_a, lista_b): # Iteramos dos listas al mismo
tiempo
41     print(i, j) # Visualizamos los valores de las dos listas
42
43
44 # Ejemplo de una función que detecta dígitos
45 def tiene_digitos(s): # Creamos una nueva función
46     out = 0 # Declaramos la variable out y lo igualamos con 0
47     for c in s: # Con ayuda de la variable c se recorre la variable s
48         if c.isdigit(): # Si el valor en la variable s es un
digito entra al ciclo
49             out = out+1 # Cada vez que se cumple la condición
se suma 1 a la variable
50     return out # Devuelve el valor de la variable out
51
52
53 print(tiene_digitos("La chica del apartamento 512")) # Detectamos el
numero de digitos que tiene la frase
54 print(tiene_digitos("La chica del apartamento 2e")) # Detectamos el
numero de digitos que tiene la frase
55 print(tiene_digitos("La chica del apartamento 12")) # Detectamos el
numero de digitos que tiene la frase
56
57
58 for i in range(5): # El iterador i recorre hasta el valor 5
59     if i == 2: # Si el valor de i es igual a 2, se entra en la
condición
60         continue # El ciclo for se sigue ejecutando
61     elif i == 3: # Si el valor de i es igual a 3 se rompe el ciclo
62         break # El ciclo for se termina
63     print(i) # Visualizamos el valor de i
64
65
66 # Sentencia de programación IF-ELSE (Si-SINO)
67 def termostato(temperatura_real, temperatura_deseada): # Creamos una
función con dos parámetros de entrada
68     if temperatura_real < temperatura_deseada: # Si la temperatura
real es menor que la temperatura deseada la condición funciona
69         status = "Heat" # A la variable status se le asigna la
palabra Heat

```

```

70     elif temperatura_real > temperatura_deseada: # Si la temperatura
deseada es menor que la temperatura real la condición funciona
71         status = "Air Cooling" # A la variable status se le
asigna la palabra Air cooling
72     else: # Si ninguna de las condiciones se cumple entonces:
73         status = "Off" # A la variable status se le asigna la
palabra Off
74     return status # Devolvemos el valor de la variable status
75
76
77 print(termostato(25, 25)) # Se evalúan dos temperaturas y se muestra
el valor de status
78 print(termostato(18, 25)) # Se evalúan dos temperaturas y se muestra
el valor de status
79 print(termostato(38, 25)) # Se evalúan dos temperaturas y se muestra
el valor de status
80
81
82 # Estructura anidada del IF
83 def comparacion_anidada(x, y): # Creamos una función con dos
parámetros de entrada
84     if x > 2: # Si la variable x es mayor que 2 entonces:
85         if y < 2: # Si la variable y es menor que 2 entonces:
86             out = x+y # La variable out es igual a x más y
87         else: # Si y no es menor que 2 entonces:
88             out = x-y # La variable out es igual a x menos y
89     else: # Si x no es mayor que x entonces:
90         if y > 2: # Si la variable y es mayor que 2 entonces:
91             out = x*y # La variable out es igual a x por y
92         else: # Si y no es mayor que 2 entonces:
93             out = 0 # La variable out es igual a 0
94     return out # Devolvemos el valor de la variable out
95
96
97 print(comparacion_anidada(8, 1)) # Evaluamos dos números y se
muestra el valor de out
98 print(comparacion_anidada(8, 9)) # Evaluamos dos números y se
muestra el valor de out
99 print(comparacion_anidada(1, 9)) # Evaluamos dos números y se
muestra el valor de out
100 print(comparacion_anidada(-1, 1)) # Evaluamos dos números y se
muestra el valor de out
101
102
103 # Estructura while
104 i = 0 # Declaramos la variable i y la igualamos a 0

```

```

105 n = 8 # Declaramos la variable n y la igualamos a 8
106
107
108 while n >= 1: # La condición while permanece activa mientras la
condición es verdadera
109     n /= 2 # El valor de n se divide entre 2 y se reestablece el
valor de n
110     i += 1 # A la variable i se le suma 1 y se reestablece el valor
de i
111 print(f"n = {n}, i = {i}") # Visualizamos el valor de n e i

```

## Resultados

```

1
2
3
4
5
0
1
2
3
4
2
3
1
4
5
7
6
2
5
6

```

### 1.- Resultados de la práctica 5.

```

10
15
22
28
one 1
two 2
three 3
3
1
2
0
1
Off
Heat
Air Cooling
9
-1
9
0
0 = 0.5, i = 4

```

### 2.- Resultados de la práctica 5.

Con esta practica se pudo hacer uso del ciclo for, y se aplico en diferentes estructuras, y se observo el comportamiento que tenían en comparación con un ciclo while.

## Practica 6.- OpenCV propiedades de imágenes

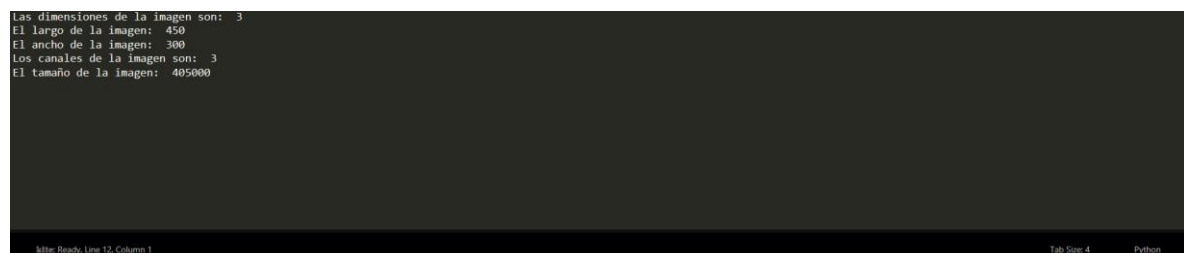
### Objetivos:

- Manejo de imágenes y sus propiedades a través de la librería de OpenCV.
- Uso de la librería OpenCV para la visualización de imágenes.
- Uso de la librería OpenCV para guardar imágenes.

### Código:

```
1 # Jym Emmanuel Cocotle Lara
2 # OpenCV propiedades de imágenes
3 import cv2 # Agregamos la librería de OpenCV
4
5
6 image_path = "Imagenes/paisaje.jpg" # Agregamos la ruta de la imagen
7
8
9 # Leemos y cargamos la imagen de la ruta indicada
10 image = cv2.imread(image_path) # Leemos la imagen
11
12
13 # Imprimimos las propiedades de la imagen
14 print("Las dimensiones de la imagen son: ", image.ndim) #
Visualizamos el numero de dimensiones de la imagen
15 print("El largo de la imagen: ", image.shape[0]) # Visualizamos el
valor de pixeles del largo de la imagen
16 print("El ancho de la imagen: ", image.shape[1]) # Visualizamos el
valor de pixeles del ancho de la imagen
17 print("Los canales de la imagen son: ", image.shape[2]) #
Visualizamos el numero de canales de la imagen
18 print("El tamaño de la imagen: ", image.size) # Visualizamos el
tamaño de la imagen
19
20 # Mostramos la imagen
21 cv2.imshow("Paisaje", image) # Le damos nombre a la ventana de la
imagen
22 cv2.waitKey(0) # Esperamos hasta que una tecla sea oprimida
```

### Resultados:



```
Las dimensiones de la imagen son: 3
El largo de la imagen: 450
El ancho de la imagen: 300
Los canales de la imagen son: 3
El tamaño de la imagen: 405000
```

Editor: Ready, Line 12, Column 1 Tab: Size: 4 Python

### 3.- Resultados de la práctica 6.





*4.- Imagen mostrada en la práctica 6.*

Es esta práctica se hizo uso de la librería OpenCV para conocer las diferentes propiedades de una imagen como lo pueden ser el largo, el ancho y la cantidad de canales que posee una imagen, y así mismo poder visualizar una imagen.

### Practica 7.- OpenCV descomposición en canales de imágenes

Objetivos:

- Uso del comando Split para la descomposición de canales de una imagen.
- Visualizar los diferentes canales de una imagen.
- Comparar los diferentes canales que posee una imagen.

Código:

```
1 # Jym Emmanuel Cocotle Lara
2 # OpenCV descomposición en canales de imágenes
3 import cv2 # Agregamos la librería de OpenCV
4 from matplotlib import pyplot as plt # De la librería matplotlib
importamos el complemento de pyplot y lo igualamos con plt
5
6
7 img = cv2.imread("Imagenes/paisaje.jpg") # Cargamos la imagen
8
9
10 if img is None: # Si no se encuentra ninguna imagen con ese nombre
    entonces:
11     print("La imagen no se encuentra en la ruta especificada") # se
muestra un mensaje de alerta
```

```
12
13
14 # Mostramos la imagen
15 cv2.imshow("Paisaje", img)  # Le damos nombre a la ventana de la
    imagen
16 cv2.waitKey(0)  # Esperamos hasta que una tecla sea oprimida
17 cv2.destroyAllWindows()  # Se destruyen todas las ventanas
18
19
20 # Separando los canales
21 blue, green, red = cv2.split(img)  # Se divide los canales de colores
    de la imagen y se igualan a valores
22 cv2.imshow("Blue", blue)  # En una ventana nueva se muestra solo el
    canal azul
23 cv2.waitKey(0)  # Esperamos hasta que una tecla sea oprimida
24
25
26 cv2.imshow("Green", green)  # En una ventana nueva se muestra solo el
    canal verde
27 cv2.waitKey(0)  # Esperamos hasta que una tecla sea oprimida
28
29
30 cv2.imshow("Red", red)  # En una ventana nueva se muestra solo el
    canal rojo
31 cv2.waitKey(0)  # Esperamos hasta que una tecla sea oprimida
32
33
34 # Guardamos los canales de la imagen
35 cv2.imwrite("Imagenes/Blue.png", blue)  # Se guarda la imagen con que
    solo posee el canal azul
36 cv2.imwrite("Imagenes/Green.png", green)  # Se guarda la imagen con
    que solo posee el canal verde
37 cv2.imwrite("Imagenes/Red.png", red)  # Se guarda la imagen con que
    solo posee el canal rojo
38
39
40 image1 = cv2.imread("Imagenes/Blue.png")  # Se lee la imagen guardada
    anteriormente
41 image2 = cv2.imread("Imagenes/Green.png")  # Se lee la imagen guardada
    anteriormente
42 image3 = cv2.imread("Imagenes/Red.png")  # Se lee la imagen guardada
    anteriormente
43
44
45 final_frame = cv2.hconcat((img, image1, image2, image3))  # En una
    imagen nueva ponemos la imagen original y las imagenes creadas
```

```
46 cv2.imwrite("Imagenes/final_image.png", final_frame) # Guardamos la
imagen final
47 cv2.imshow("Image and chanel", final_frame) # # En una ventana nueva
se muestra la imagen final
48 cv2.waitKey(0) # Esperamos hasta que una tecla sea oprimida
```

Resultados:



5.- Imagen Original.



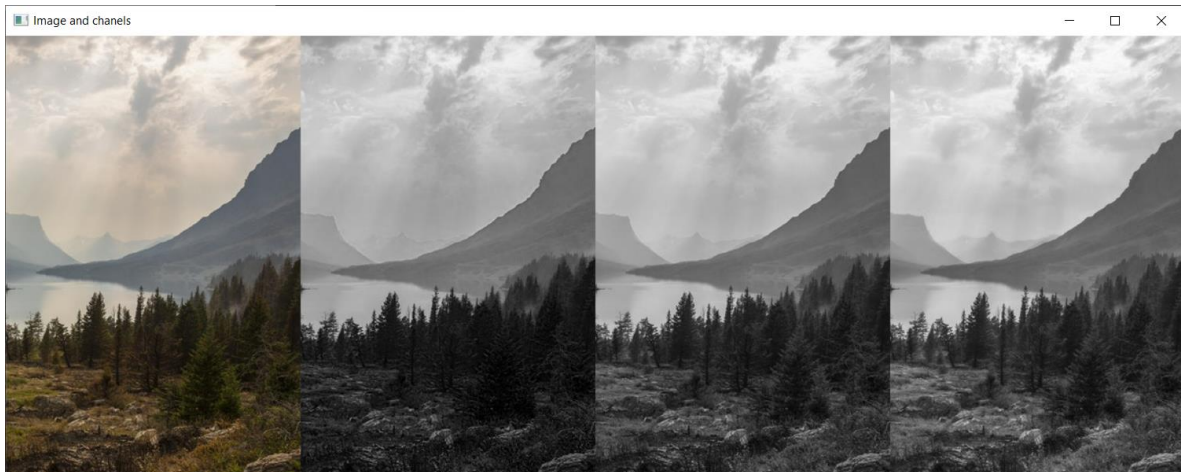
*6.- Imagen con solo el canal azul.*



*7.- Imagen con solo el canal verde.*



*8.- Imagen con solo el canal rojo.*



*9.- Imagen comparativa.*

A través de la práctica pude conocer los diferentes canales de una imagen y como descomponerlos y posteriormente visualizarlos, al igual que conocí como unir diferentes imágenes en una sola imagen.

## Practica 8.- OpenCV escala de grises

### Objetivos:

- Uso de la librería OpenCV para convertir una imagen en escala de grises.
- Visualización de la imagen en escala de grises.

### Código:

```
1 # Jym Emmanuel Cocotle Lara
2 # OpenCV escala de grises
3 import cv2 # Agregamos la librería de OpenCV
4
5
6 # Leemos la imagen desde el directorio
7 imagen_escala_grises = cv2.imread("Imagenes/paisaje.jpg",
cv2.IMREAD_GRAYSCALE)
8
9
10 # Creamos la imagen en escala de grises en el directorio
11 cv2.imwrite("Imagenes/paisaje_gray.jpg", imagen_escala_grises)
12
13
14 # Mostramos la imagen en escala de grises
15 cv2.imshow("imagen en escala de grises", imagen_escala_grises)
16 cv2.waitKey(0) # Esperamos hasta que una tecla sea oprimida
```

Resultados:



*10.- Imagen en escala de grises.*

En esta práctica se pudo transformar una imagen en escala rgb a escala de grises con ayuda de la librería de OpenCV, para posteriormente visualizarla y guardarla.

#### Practica 9.- OpenCV generación de imágenes

Objetivos:

- Uso de la librería OS
- Creación de imágenes a partir de bytes randoms.
- Conversión de imágenes a escala de grises y rgb.

Código:

```
1 # Jym Emmanuel Cocotle Lara
2 # OpenCV generación de imágenes
3 import cv2 # Agregamos la librería de OpenCV
4 import numpy # Agregamos la librería de numpy
5 import os # Agregamos la librería de os
6
7
8 # Generamos un array con 120,000 bytes random
9 random_byte_array = bytearray(os.urandom(120000))
10
11
```

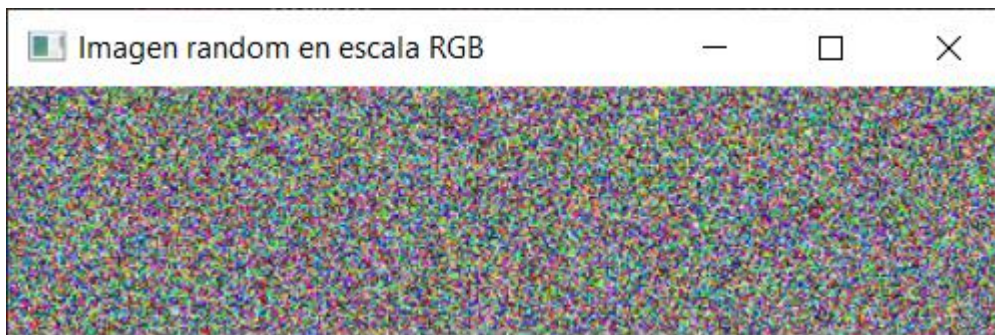
```
12 # Asignamos los valores aleatorios a un array
13 flat_numpy_array = numpy.array(random_byte_array)
14
15
16 # Convertir los bytes a imagen 400X300, 300X400 etc...
17 imagen_escala_grises = flat_numpy_array.reshape(300, 400) #
Reasignamos los bytes para crear una imagen en escala de grises
18 cv2.imwrite("Imagen_random_escala_grises.jpg", imagen_escala_grises)
# Guardamos la imagen en escala de grises
19
20
21 #Escribimos la imagen con color
22 imagen_rgb = flat_numpy_array.reshape(100, 400, 3) # Reasignamos los
bytes para crear una imagen con escala rgb
23 cv2.imwrite("Imagen_random_rgb.jpg", imagen_rgb) # Guardamos la
imagen con escala rgb
24
25
26 cv2.imshow("Imagen random en escala de grises", imagen_escala_grises)
# Mostramos la imagen en escala de grises
27 cv2.waitKey(0) # Esperamos hasta que una tecla sea oprimida
28
29
30 cv2.imshow("Imagen random en escala RGB", imagen_rgb) # Mostramos la
imagen en escala rgb
31 cv2.waitKey(0) # Esperamos hasta que una tecla sea oprimida
```



Resultados:



*11.- Imagen en escala de grises a partir de bytes randoms.*



*12.- Imagen en rgb a partir de bytes randoms.*

Con ayuda de la librería os se pudo crear una imagen a partir de bytes aleatorios y posteriormente se visualizo en escala de grises y rgb.

#### Practica 10.- OpenCV modificación de pixeles

Objetivos:

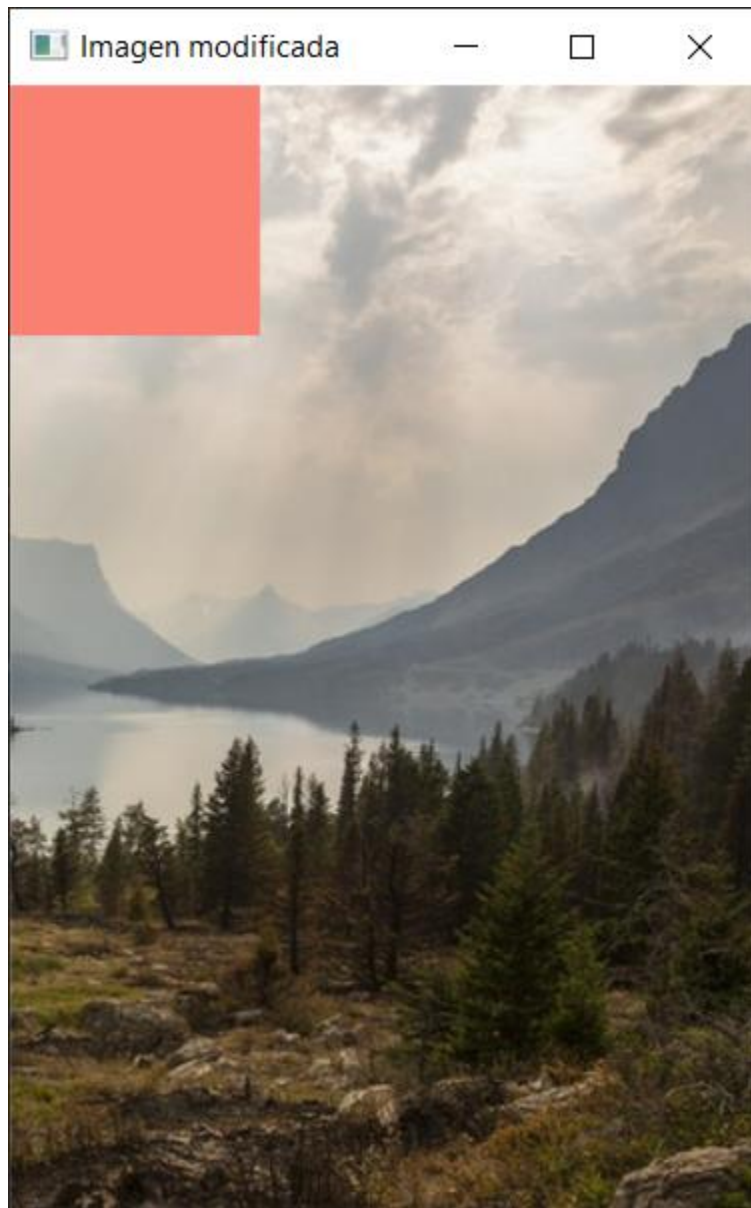
- Modificación de pixeles.
- Cambio de color de pixeles.

### Código:

```
1 # Jym Emmanuel Cocotle Lara
2 # OpenCV modificación de pixeles
3 import cv2 # Agregamos la librería de OpenCV
4
5
6 # Escribimos la ruta de la imagen
7 ruta_imagen = "Imagenes/paisaje.jpg"
8
9
10 # Cargamos la imagen
11 image = cv2.imread(ruta_imagen)
12
13
14 # Accedemos al pixel localizado en la ubicación (0,0)
15 (b, g, r) = image[0, 0]
16 print("Valores azul, verde, rojo del pixel en (0,0)", format((b, g,
r))) # Visualizamos los valores del pixel 0,0
17
18
19 # Manipulamos el pixel de color salmon
20 image[0:100, 0:100] = (114, 128, 250) # Se asigna un nuevo valor a
los pixeles en x de 0 hasta 100 al igual que en y
21
22
23 # Mostramos la imagen modificada
24 cv2.imshow("Imagen modificada", image)
25 cv2.waitKey(0) # Esperamos hasta que una tecla sea oprimida
```

### Resultados:

```
Valores azul, verde, rojo del pixel en (0,0) (146, 154, 161)
[Finished in 8.4s]
```



*14.- Imagen con pixeles modificados.*

A través de esta practica se pudo identificar los valores de un pixel ubicado en cierta posición y a partir de esto modificar el color de este.

## Conclusión

Con ayuda de la librería de OpenCV se pueden hacer una gran variedad de operaciones en lo que a tratamiento de imágenes se refiere, por lo que resulta muy útil para la visión por computadora. Aunque no se mostraron todas las funciones de OpenCV, es importante conocer estas funciones ya que es a partir de estas que se pueden tratar las imágenes.

El uso de cada función depende de la aplicación a realizar, por lo que resulta muy conveniente el gran repertorio que posee OpenCV.

### Bibliografía:

1. Briega, R. (2021). Visión por computadora - Libro online de IAAR. Retrieved 29 May 2021, from <https://iaarbook.github.io/vision-por-computadora/#:~:text=OpenCV%20es%20una%20librería%20Open,objetos%20en%20la%20vida%20real>.