

Para acceder al código de las prácticas pulse [aqui](#)

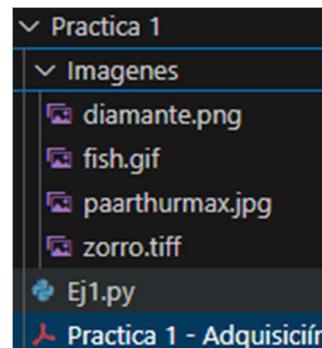
Memoria de la práctica 1 de VCO

Alumno:

-Vicente Burdeus Sánchez

Ej1

1. Descargue algunas imágenes de la carpeta <prácticas\IMAGES> en PoliformaT o busque en internet imágenes de cualquier tipo, como paisajes, flores, galaxias, etc..; con formatos diversos como: jpg, tif, gif o png.



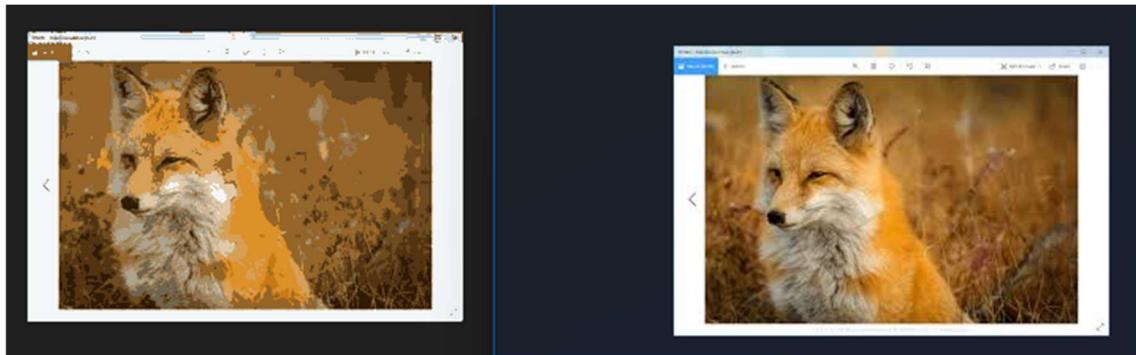
2. Lea alguno de los archivos de color de tipo tif descargado muéstrela en una ventana mediante Pillow. Si no se visualiza bien intente poner el mapa de colores o paleta que se corresponda. Muestre también la barra de colores al lado de la imagen. ¿Cuántos colores tiene la imagen?



```
if __name__ == "__main__":
    img = Image.open(imgname("tif"))
    print("Modo de la imagen:", img.mode)
    colors = img.getcolors(maxcolors=img.size[0] * img.size[1])
    print("Número de colores únicos en la imagen:", len(colors))

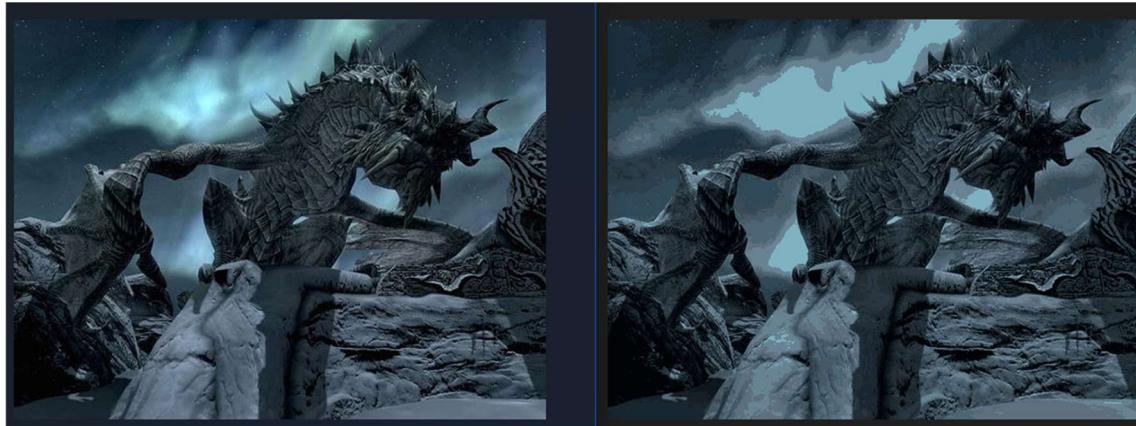
#Modo de la imagen: RGBA
#Número de colores únicos en la imagen: 23821
```

3. Reduzca el número de colores de la imagen anterior a 16 y muéstrela. Compare ambas imágenes visualmente.



```
if __name__ == "__main__":  
  
    img = Image.open(imgname("tif"))  
    img.show()  
    img_reduced = img.convert("P", palette=Image.ADAPTIVE, colors=16)  
    img_reduced.show()
```

4. Repita las operaciones anteriores con una imagen de tipo jpg. Conviértala primero a tipo 'P' con paleta y repita las operaciones anteriores.



Ej2

1. Utilizando cualquiera de las imágenes introducidas en el ejercicio anterior, aplique distintas transformaciones y compruebe el resultado.
2. Convierta cualquier imagen en color en una imagen de niveles de gris y visualice ambas.
3. Tomar una imagen RGB y muestre los tres componentes R (rojo), G (verde) y B (azul) como imágenes de gris por separado. Pruébelo con la imagen ‘aloel.jpg’.

EJ 3 y 4

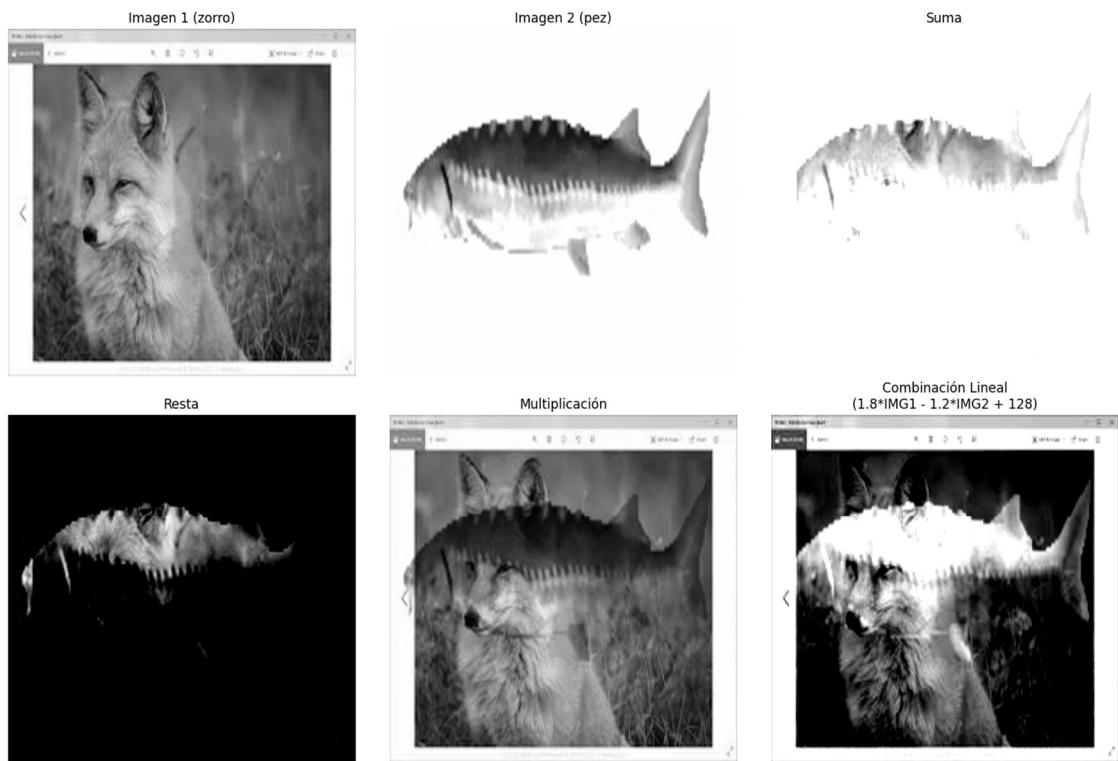
► Ejercicio 3 – Aritmética con imágenes

1. Lea las imágenes ‘cameraman.tif’ y “moon.tif”, que se encuentras en la instalación de MATLAB. 2. Redimensiónelas para que tengan el mismo tamaño, por ejemplo 256 x 256. 3. Realice algunas operaciones aritméticas entre ellas y visualice el resultado. 4. Realice la combinación lineal siguiente y visualice:

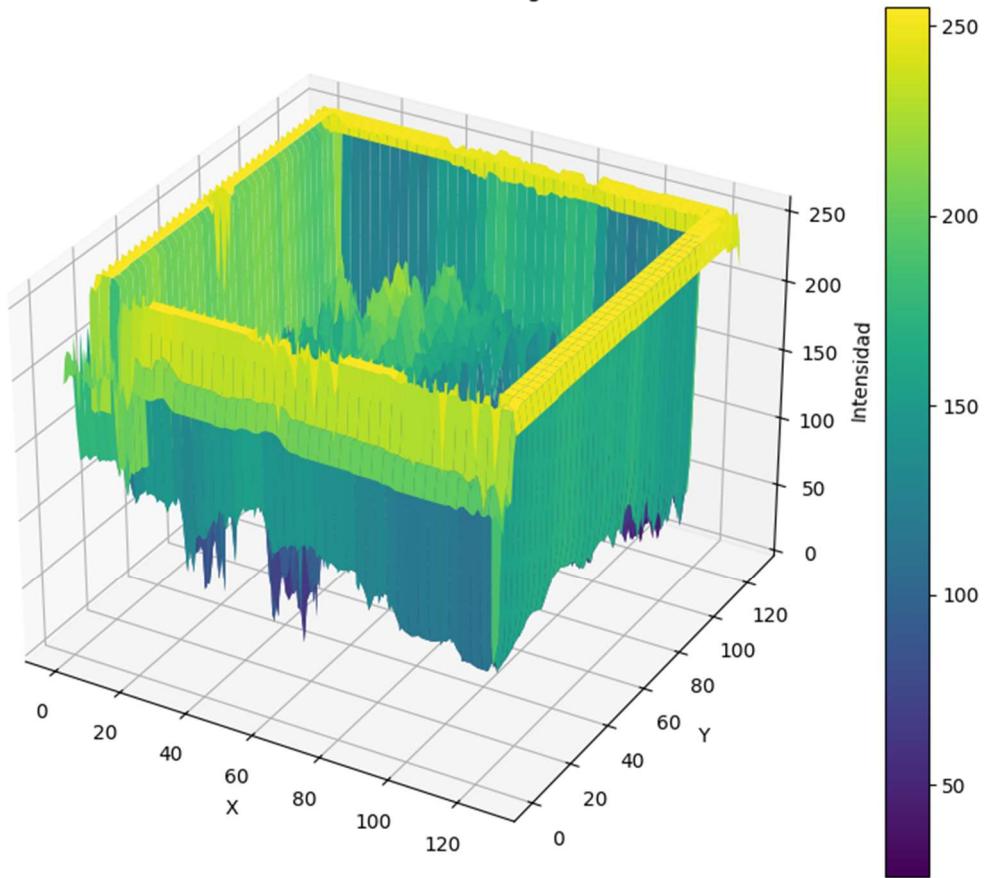
$$S = CAM * 1.8 - MOON * 1.2 + 128$$

► Ejercicio 4 – Visualización 3D

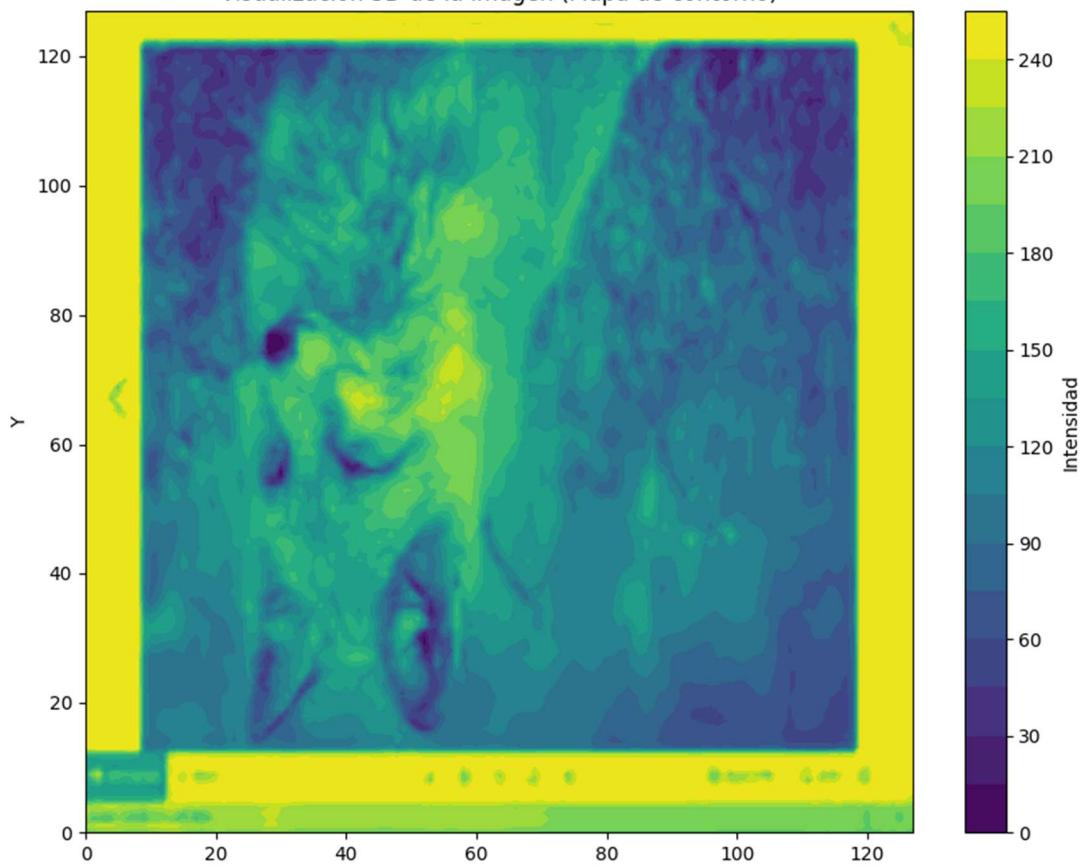
1. Tome alguna de las imágenes anteriores, de tipo grayscale, y represéntela como una superficie empleando las funciones plt.contourf y numpy.meshgrid.



Visualización 3D de la imagen



Visualización 3D de la imagen (Mapa de contorno)



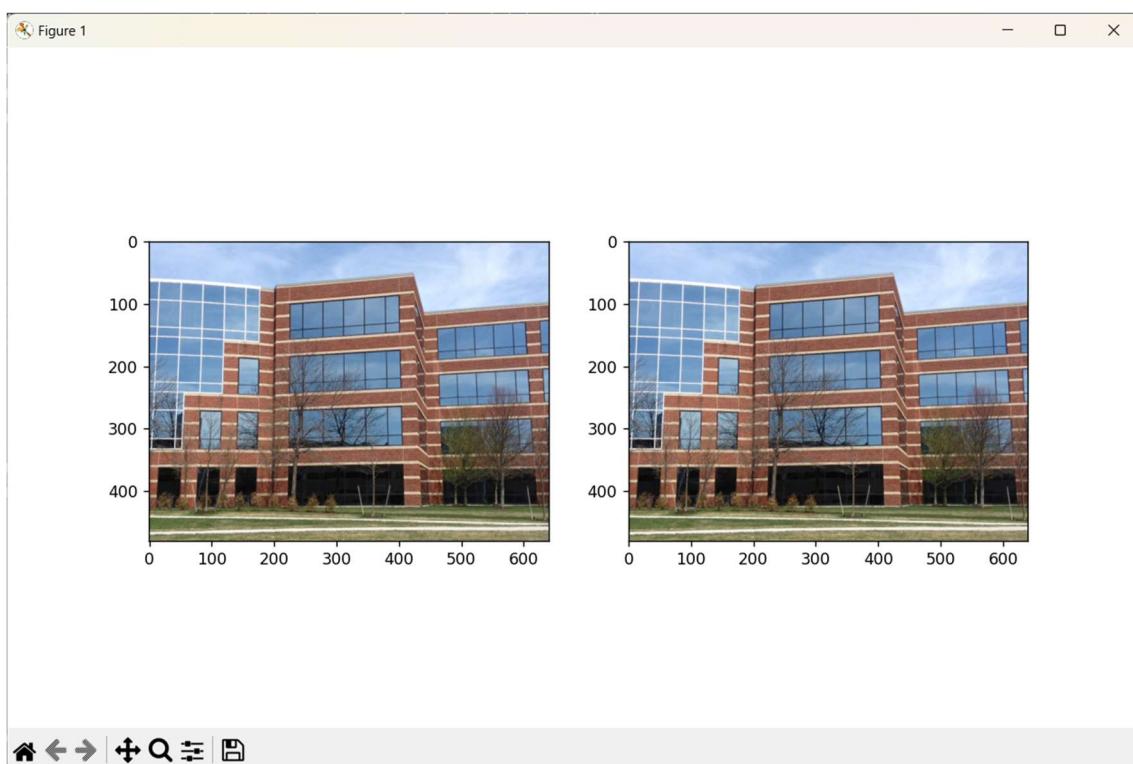
Memoria de la practica 2 de VCO

Alumno:

-Vicente Burdeus Sánchez

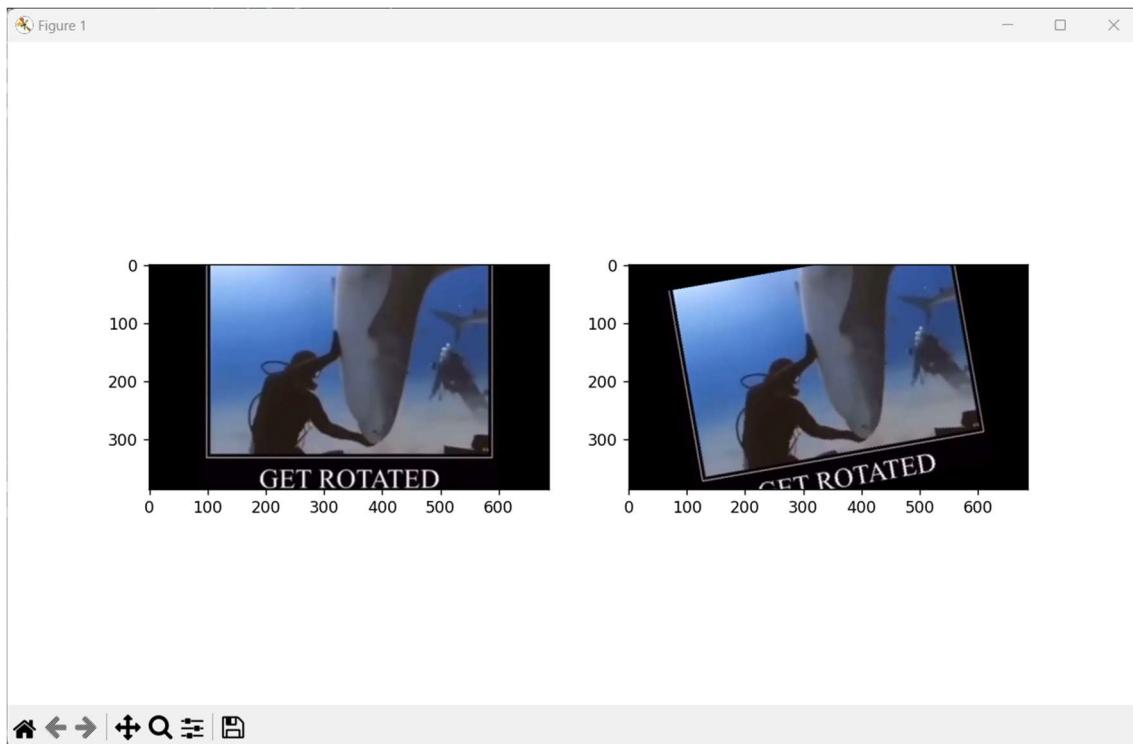
Ejercicio 1

1. Tome una imagen cualquiera de las utilizadas en la práctica anterior, o de las disponibles en PoliformaT. Muestre la imagen en una ventana empleando el script “Afin_image.py”.



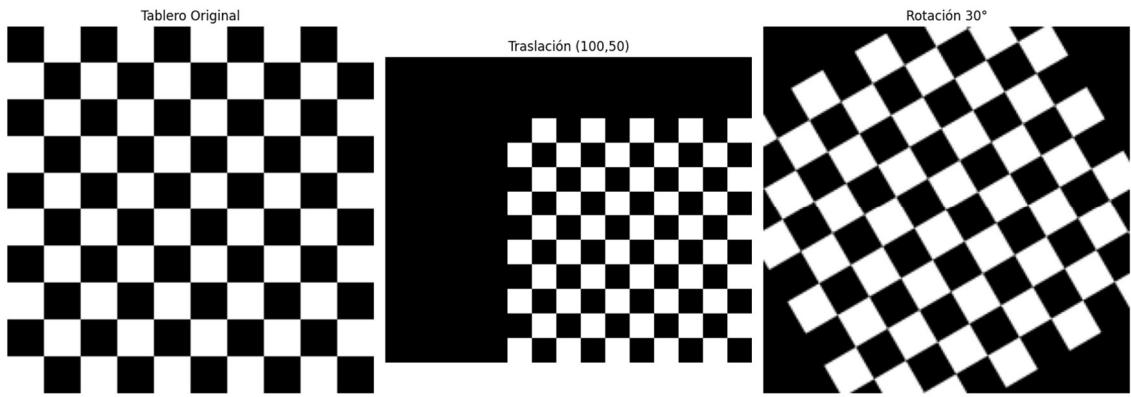
2. Modifique el script anterior para aplicar una rotación de +10º a la imagen, siguiendo el ejemplo anterior, y muestre el resultado en la otra ventana, o utilice la misma ventana empleando la función matplotlib.pyplot. -*

```
● ● ●  
img_t = img_t.rotate(10)
```

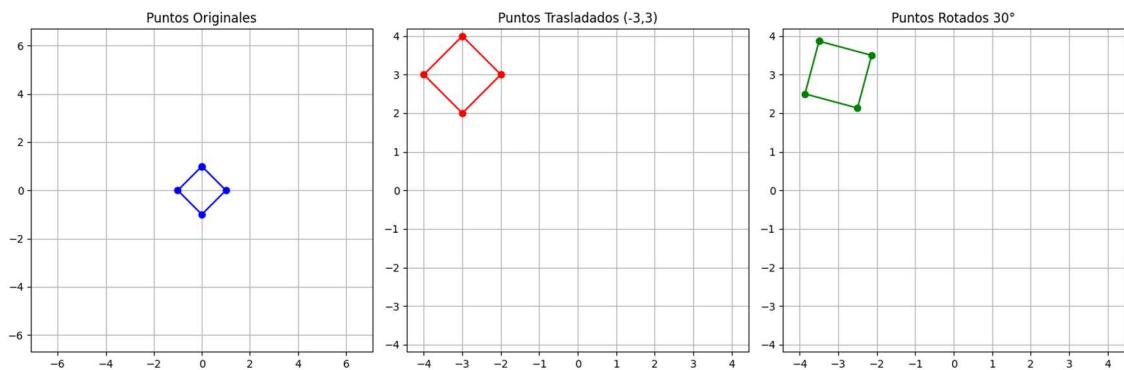


Ejercicio 2

1. Utilice el script “checkerboard.py” para crear una imagen de un tablero de ajedrez (Checkerboard) de 10x10 cuadrados de 20 pixeles de lado. Muéstrela en una imagen.
2. Cree una matriz de transformación afín T para una traslación simple $t=(tx,ty)=(100,50)$. Cree una segunda matriz de transformación R para una rotación de ángulo $\Theta=30$ grados. Aplique las transformaciones indicadas a la imagen y muestre los resultados.



1. Escriba el script anterior en un archivo “RotacionPuntos.py”
2. Cambie la matriz de transformación afín T para que produzca una traslación simple $t=(tx, ty)=(-3, 3)$.
Cree una segunda matriz de transformación R para una rotación de ángulo Theta=30 grados. Aplique las transformaciones indicadas a los puntos y muestre los resultados, que deben ser como la siguiente figura:



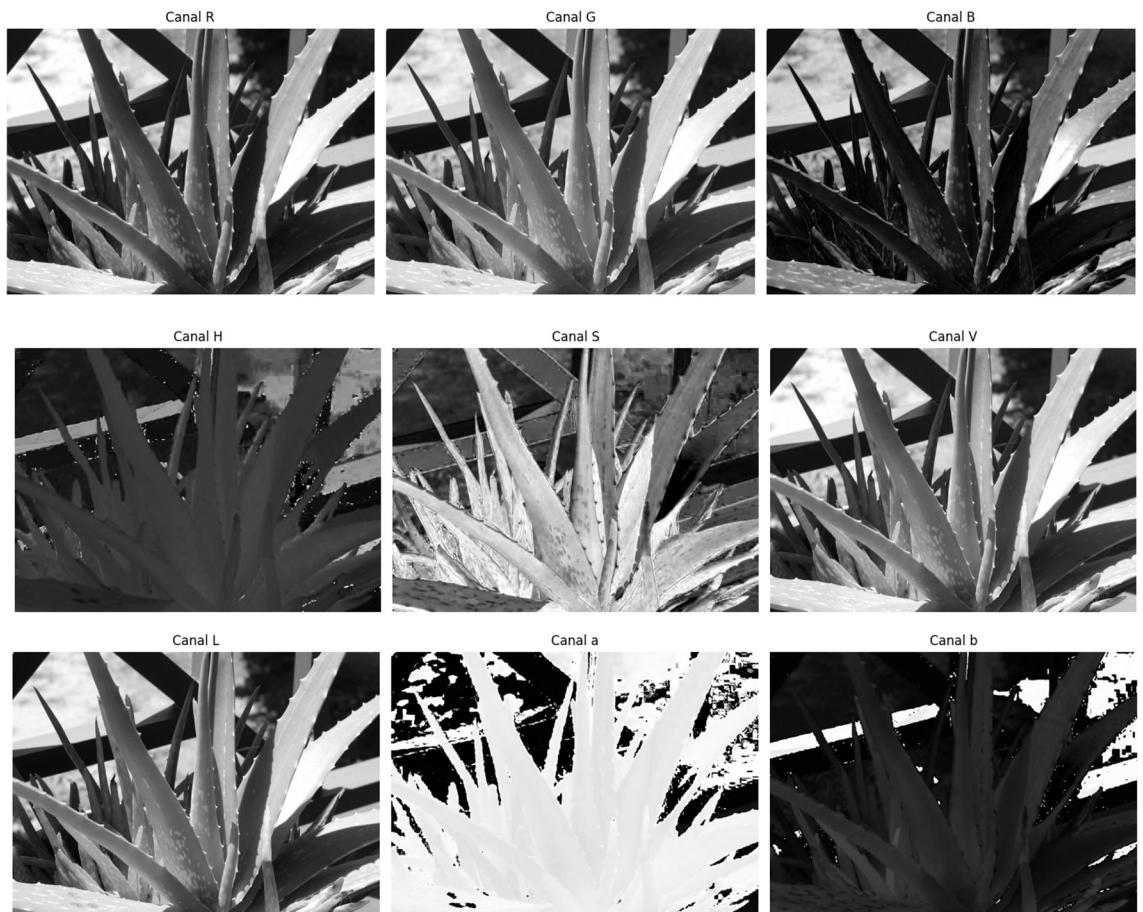
1. Cree un archivo “VisualizaOpenCV.py” y escriba el script anterior, añadiendo las instrucciones indicadas.
Compruebe si se visualizan bien las imágenes con Matplotlib. ¿Qué problema se plantea?

Pista: Las imágenes en OpenCV tienen el formato ‘BGR’, mientras que Matplotlib utiliza el formato ‘RGB’.

Busque en OpenCV una función para hacer esa conversión antes de visualizar.



1. Tomar una imagen RGB y muestre los tres componentes R (rojo), G (verde) y B (azul) como imágenes de gris por separado. Pruebelo con la imagen 'AloeVera.jpg'.
2. Transforme la misma imagen a HSV y muestre los tres componentes H (hue), S (Saturation) y V (Value) como imágenes en gris por separado.
3. Ídem que lo anterior, pero a espacio Lab.



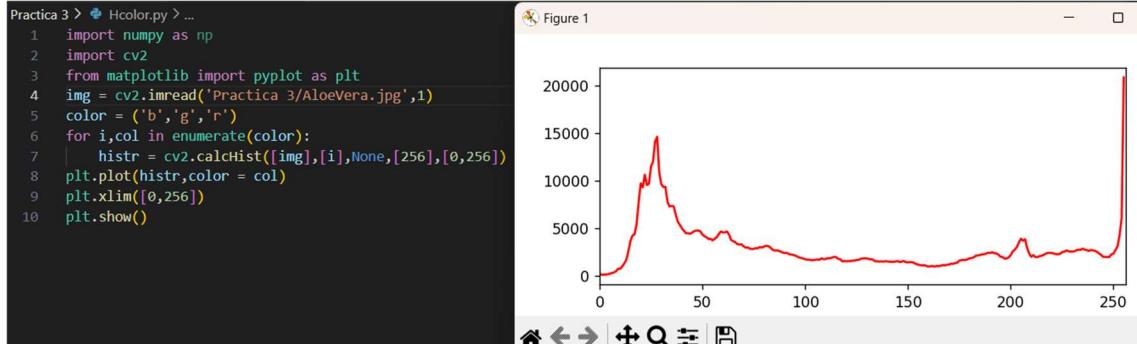
Memoria de la practica 3 de VCO

Alumno:

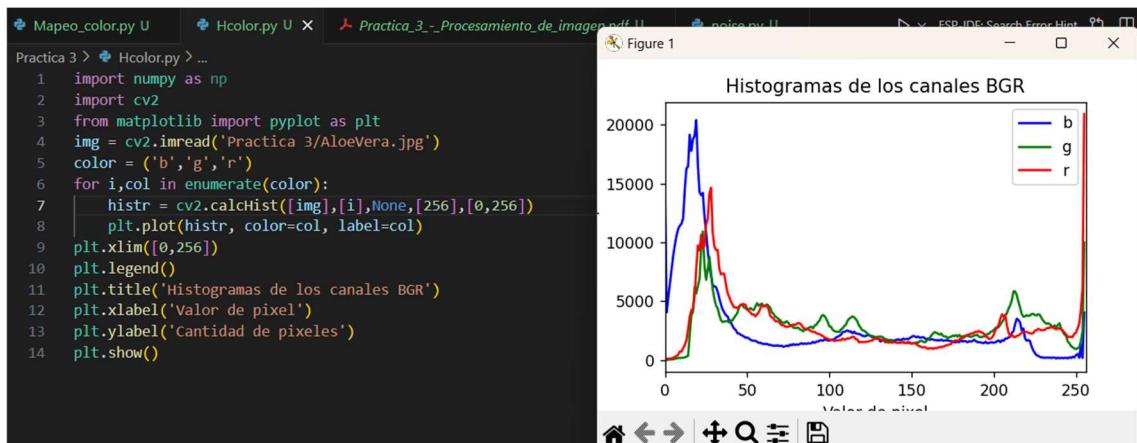
-Vicente Burdeus Sánchez

1.

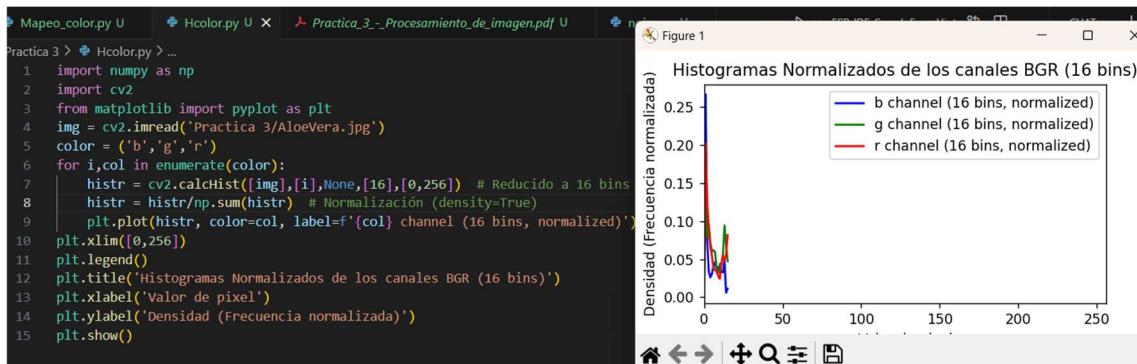
Ejecutar el código



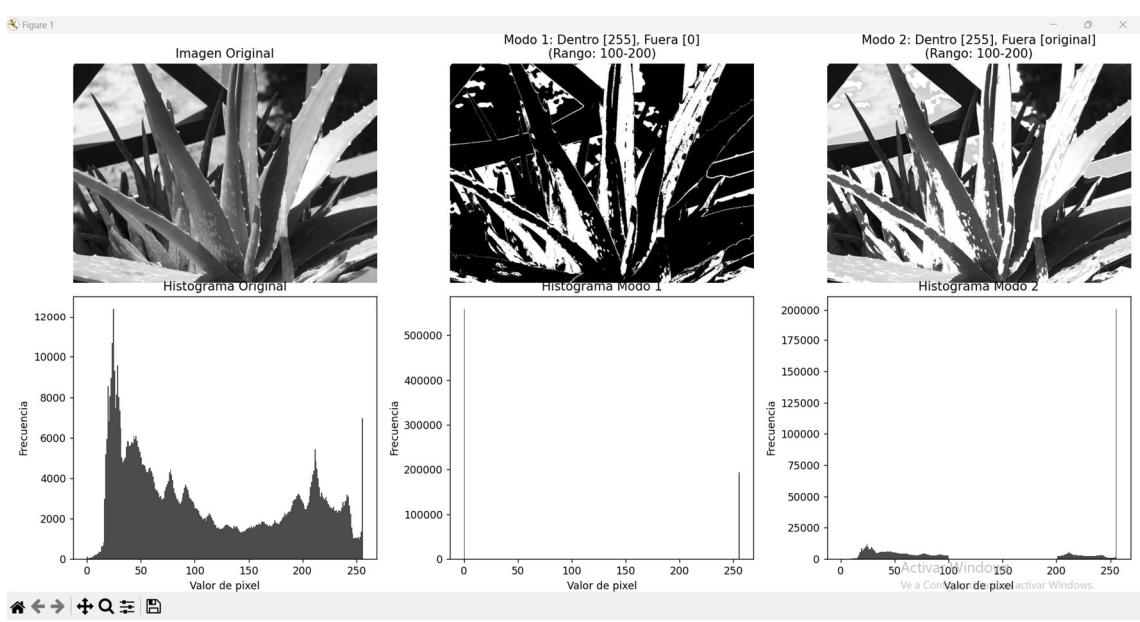
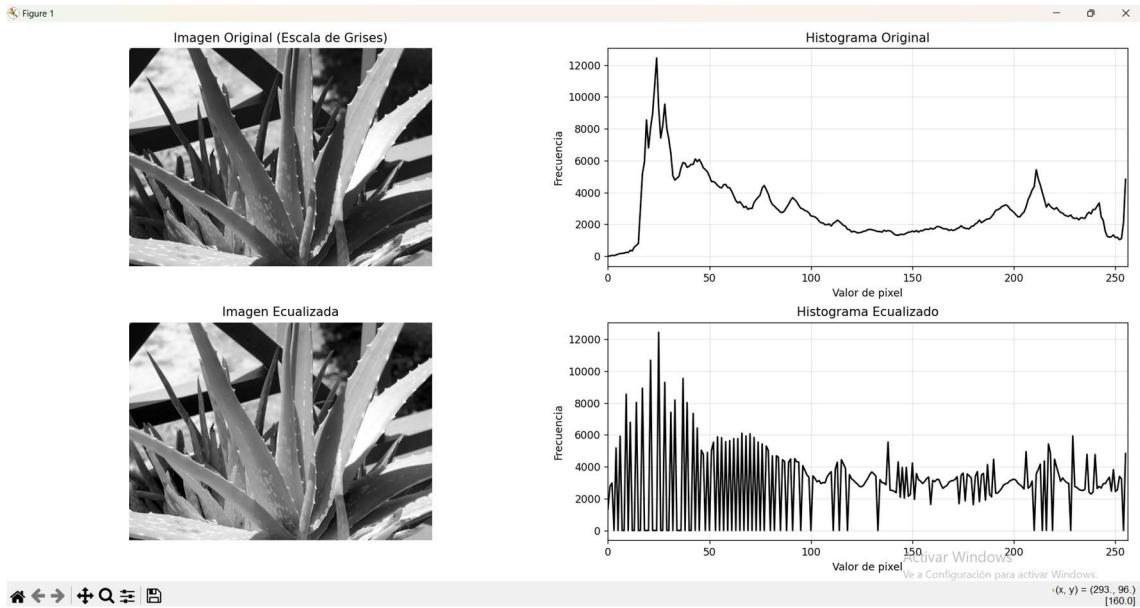
Dibujar los tres canales



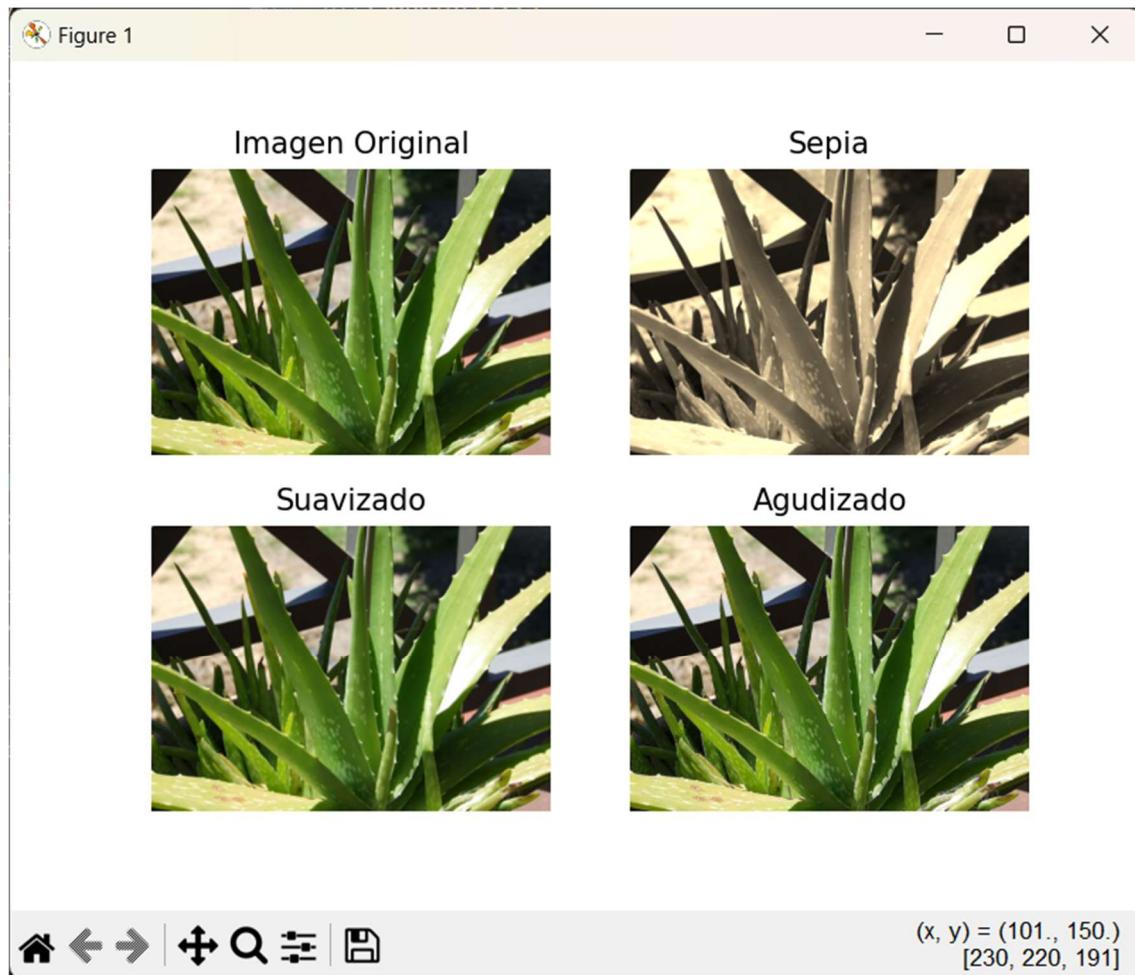
Modifique alguna de las propiedades de los histogramas, como por ejemplo el número de bins a valor 16 o la 'density = True', y observe el resultado



2.



Ejercicio 4 aplicar filtros



Ejercicico 5



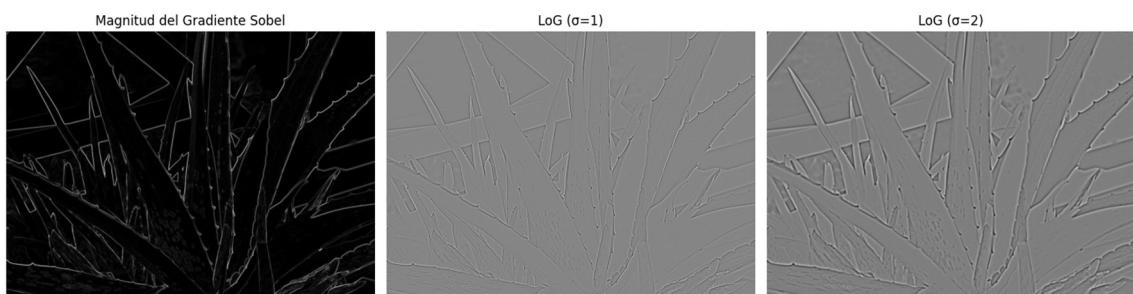
Añada un ruido gaussiano y aplique a la imagen con ruido un filtrado Gaussiano con la función anterior. Muestre en una ventana la imagen original y la resultante.



1. Tome una imagen cualquiera y pruebe las funciones de Sobel para detección de bordes horizontales y verticales.

2. Combine el Laplaciano y el Gaussiano para crear un LoG.

3. Muestre los distintos resultados en ventanas diferentes.

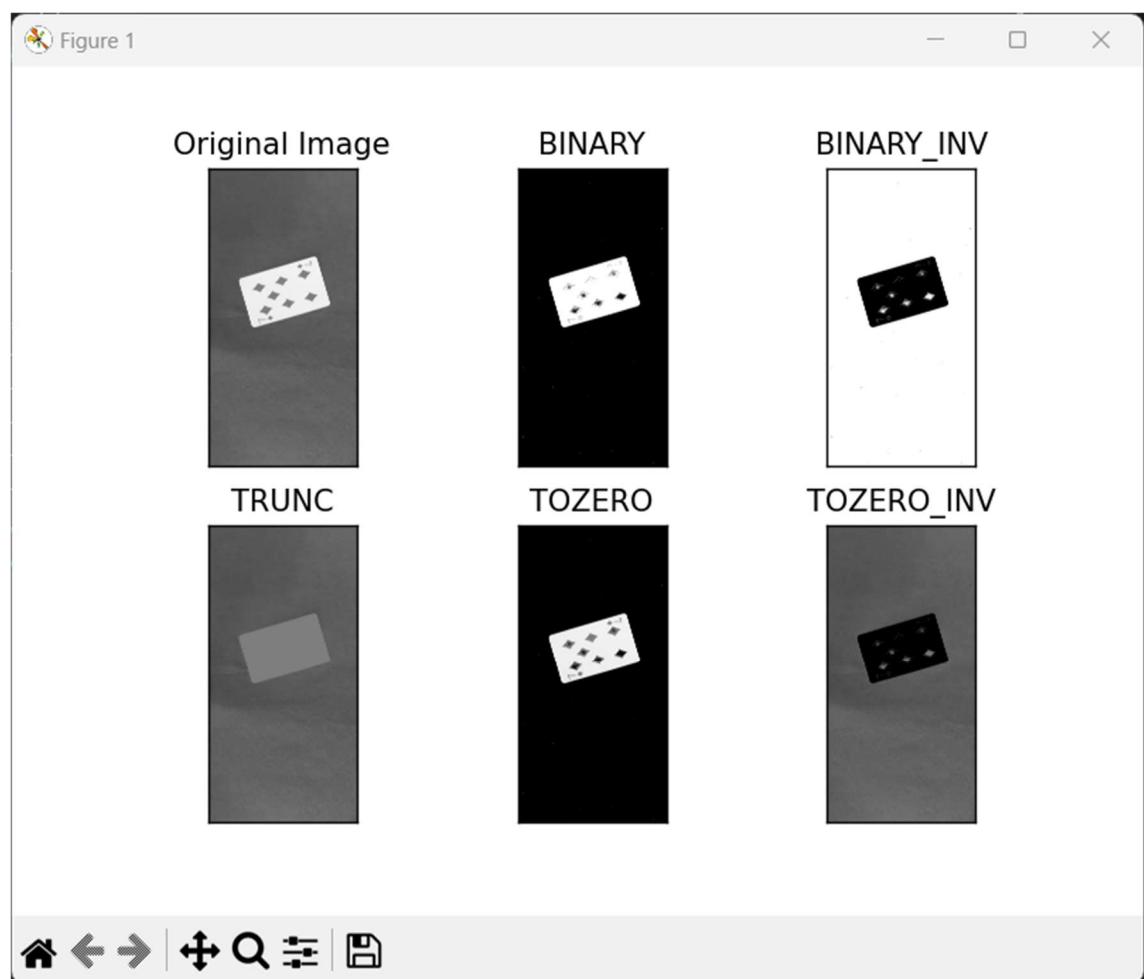


Memoria de la práctica 4 de VCO

Alumno:

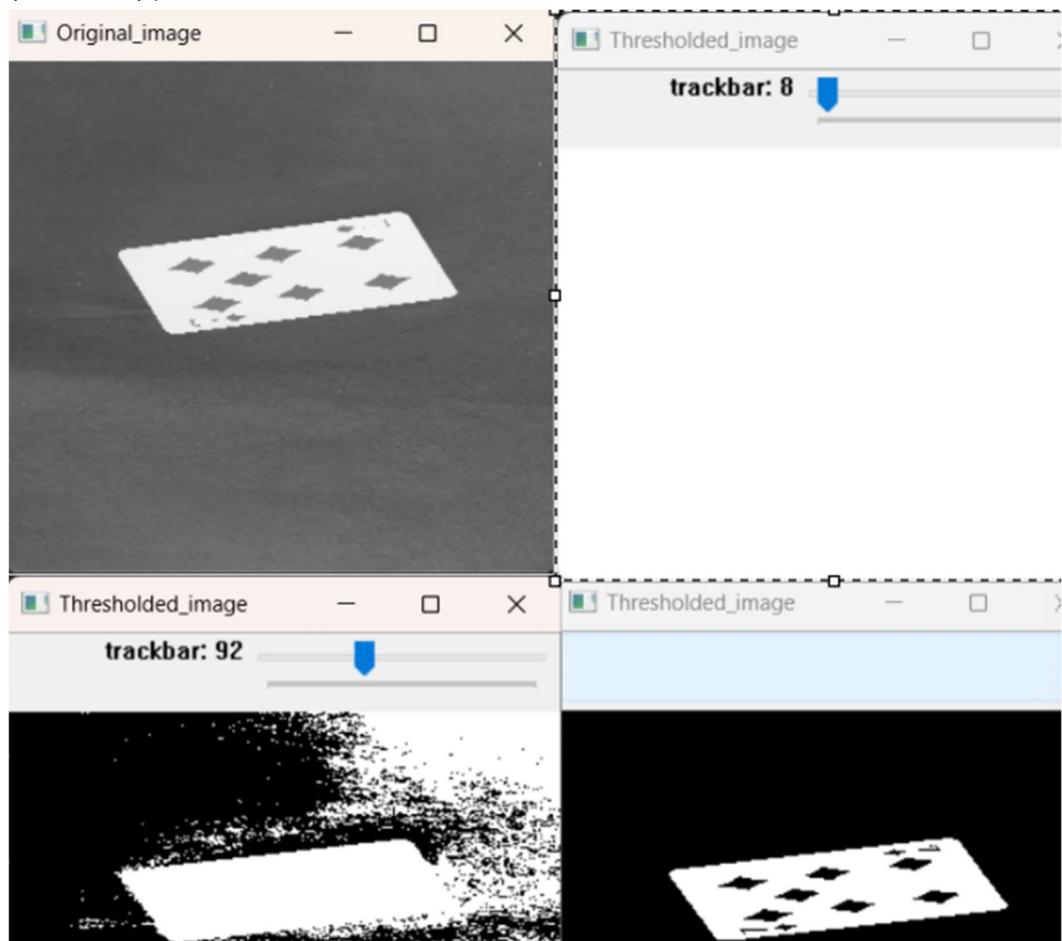
-Vicente Burdeus Sánchez

1. En la carpeta <Recursos/Imágenes VxC/Baraja póke 1/> de PoliformaT hay un conjunto de imágenes de cartas de póker que vamos a utilizar a partir de ahora. Busque esas imágenes y póngalas en una carpeta propia.
2. Introduzca el código anterior con el nombre ‘umbralización.py’ y pruebe el resultado con distintas imágenes de la carpeta de cartas.
3. Añada los otros métodos de umbralización indicados en la lista ‘type’.



1 Con el script anterior hay que modificar el valor del archivo y/o los valores de umbral para cada imagen.

Para tener una versión más interactiva tome el archivo umbralización_global.py de PoliformaT. Lea el código y estudie cómo se lee una carpeta y se listan las imágenes. También cómo se utiliza un deslizador (Trackbar) para definir el valor del umbral.



2. Aplique la umbralización global de tipos cv2.THRESH_BINARY y cv2.THRESH_BINARY_INV, a diversas imágenes de las cartas.

¿Qué objeto es el más indicado para segmentar en este caso? La idea es que, a veces, es conveniente extraer un objeto que nos delimita las partes de la imagen que queremos observar. En este caso el objeto a discriminar es la carta, aunque ese objeto tenga agujeros en su interior.

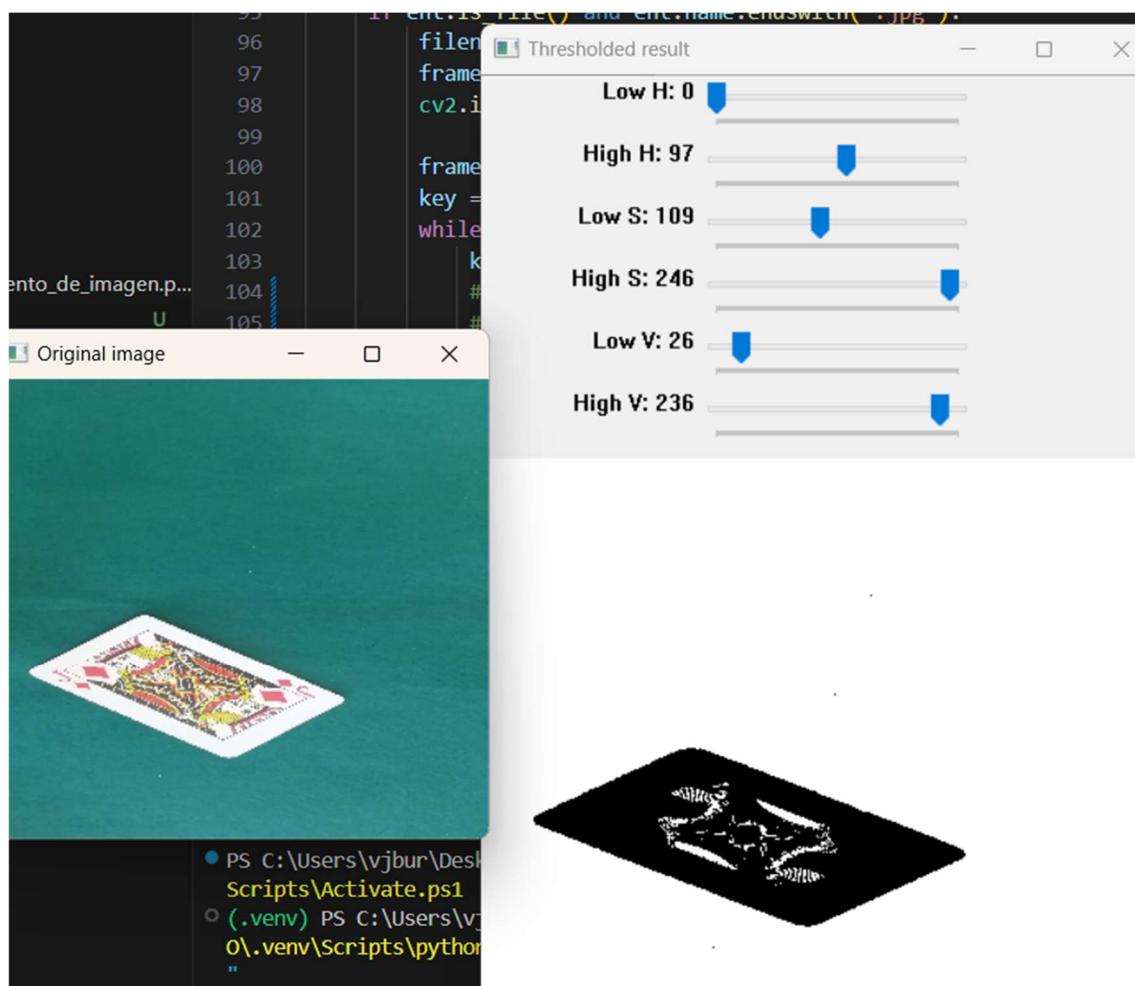
3. ¿Qué umbral sería apropiado para la umbralización global funcione bien con todas las imágenes?

En el caso en el que queramos únicamente diferencial la carta sin ver que carta es podemos usar un umbral muy alto alrededor de 220 el cual hace correcto todos los casos.

EJ3

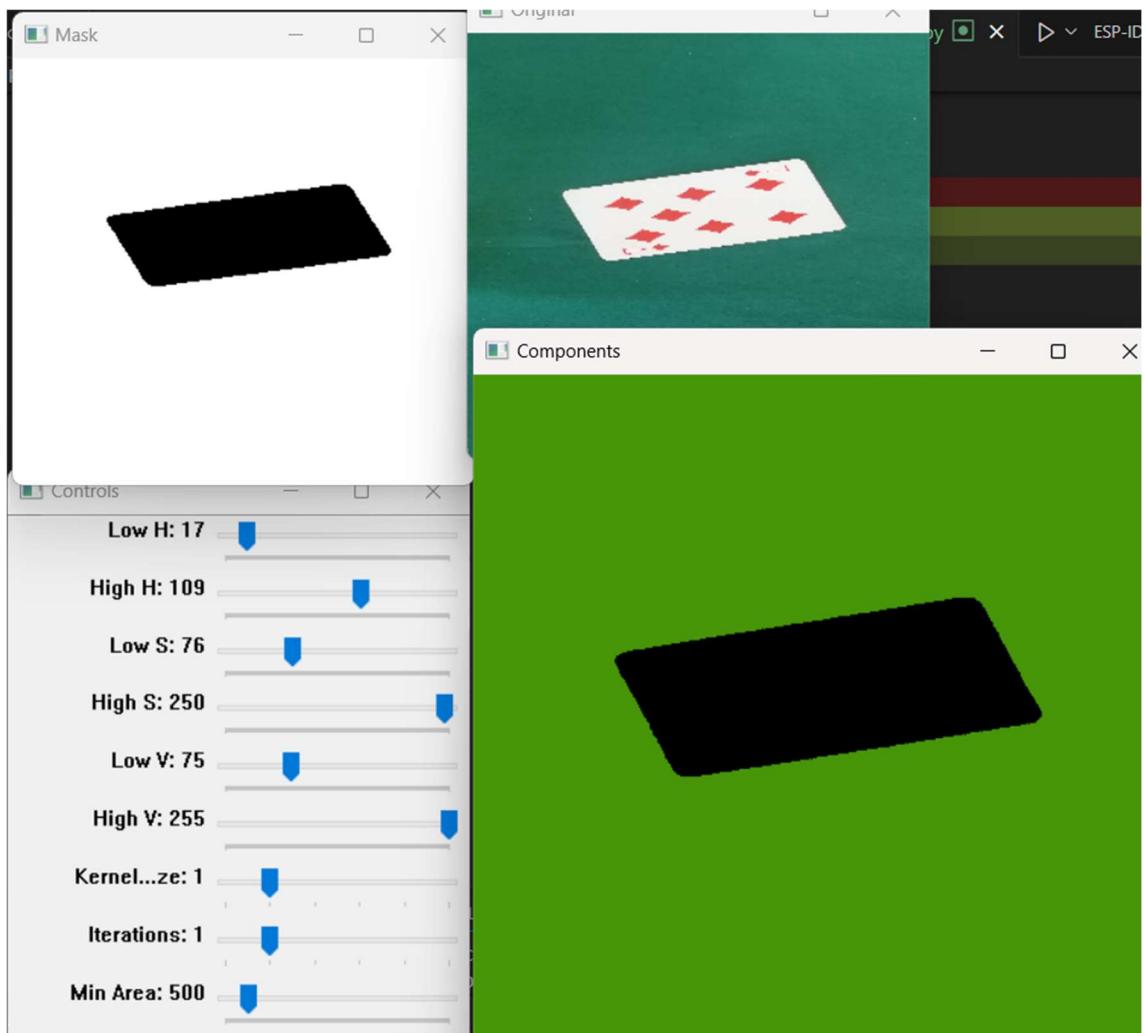
Ej 4

1. Tome de PoliformaT el script ‘umbralizacion_color.py’. Revise el programa para entender lo que está haciendo y complete las funciones que faltan para convertir de BGR a HSV y busque la información de OpenCV de la función cv2.inRange(.....) e inserte ésta en el código.
2. Pruebe con distintas imágenes de la carpeta de cartas de póker.
3. Establezca los umbrales de H y S para segmentar el fondo verde y verifique que funciona bien para todas las cartas.



EJ 5

1. Tome de script anterior y cree un nuevo archivo denominado 'componentes_conectadas.py'. Revise el programa para entender lo que está haciendo y pruebe con distintas imágenes de la carpeta de cartas de póker.
2. Cambie el código para que funcione con 8 conexión. Prueve cambiando el área mínima.



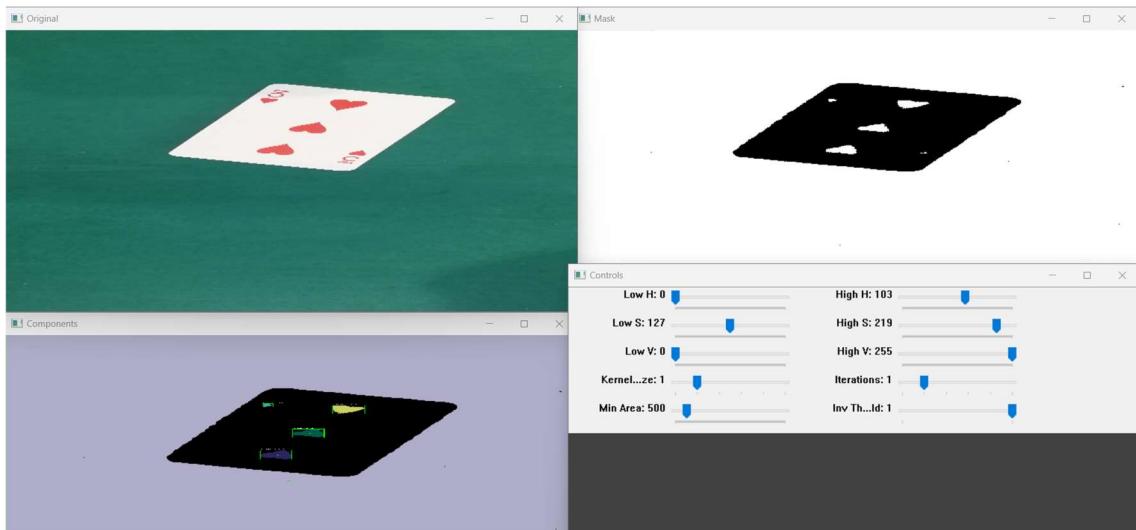
EJ 6

1. Añada al código anterior la obtención del recuadro (Bounding Box) de cada objeto que supera el filtro de tamaño y muéstrela en la imagen original mediante `cv2.rectangle(image, start_point, end_point, color, thickness)`. Utilice al color azul claro (255,255,0) y grosor de 10 px.
2. Ídem que lo anterior, pero mostrando el centroide mediante `cv2.circle(image, center_coord, radius, color, thickness)`. Utilice el color amarillo (255,255,0), radio 4 y grosor 5.
3. Pruebe con distintas imágenes de la carpeta de cartas de póke

EJ 7

1. Añada la función anterior al archivo ‘componentes_conectadas.py’ y muestre la imagen de etiquetas coloreadas en una nueva ventana, además de las anteriores.

2. Pruebe con distintas imágenes de la carpeta de cartas de póker.



Memoria de la practica 5 de VCO

Alumno:

-Vicente Burdeus Sánchez

EJ1

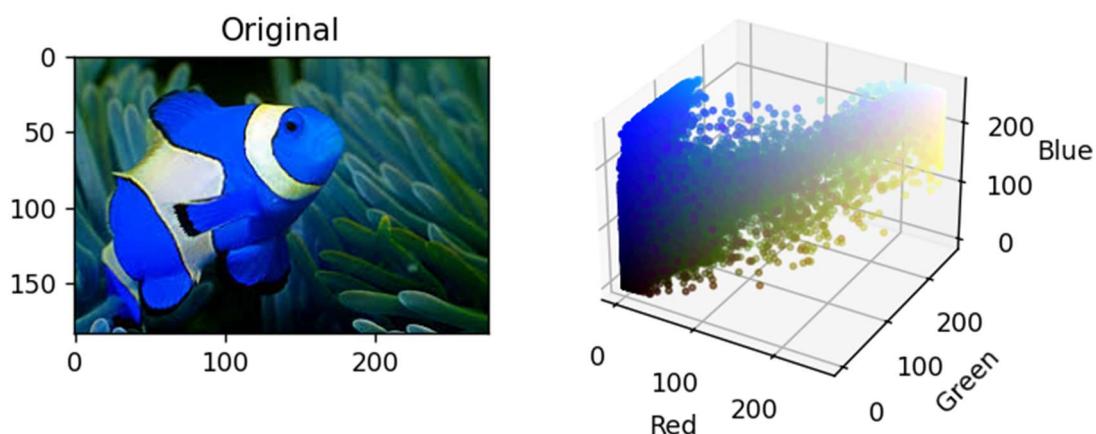
1. Escriba el código anterior y pruebe el resultado.

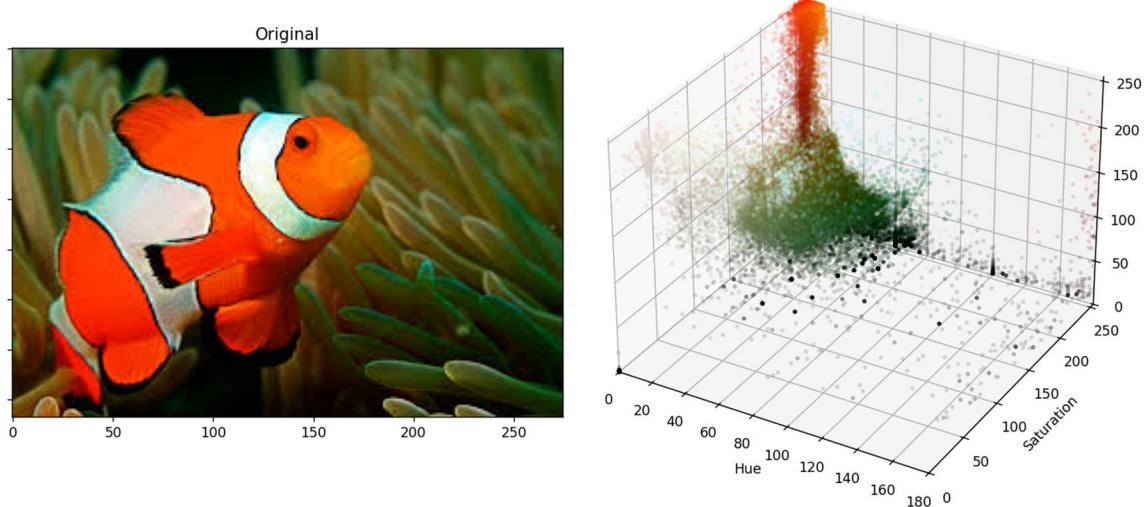
Supuestamente deberá obtener una imagen de nemo con tonos rojos, sin embargo, el pez se muestra en tonos azules.

¿Podría indicar a qué es debido esto e introducir la solución en el script?

2. Repita lo anterior, pero trabajando con la imagen en HSV. Deberá convertir a HSV y utilizar la función:

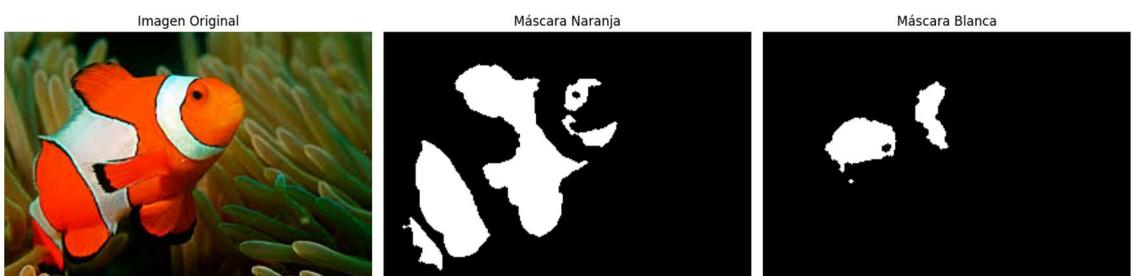
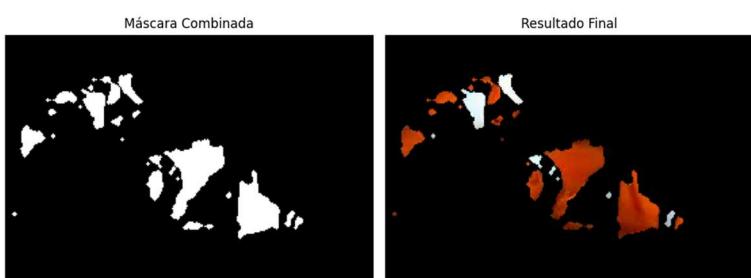
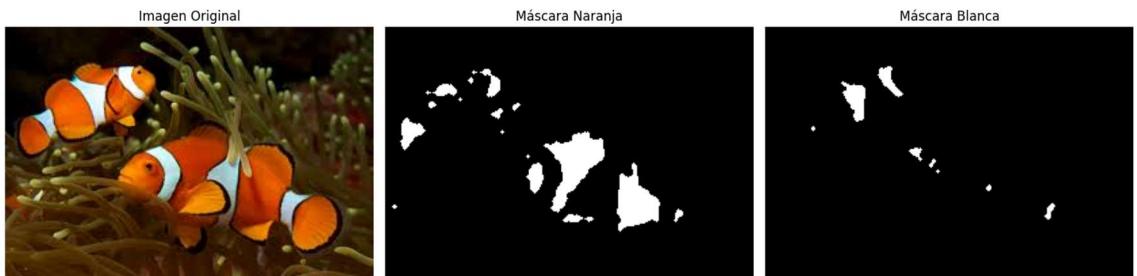
```
h, s, v = cv2.split(nemo_hsv)
```



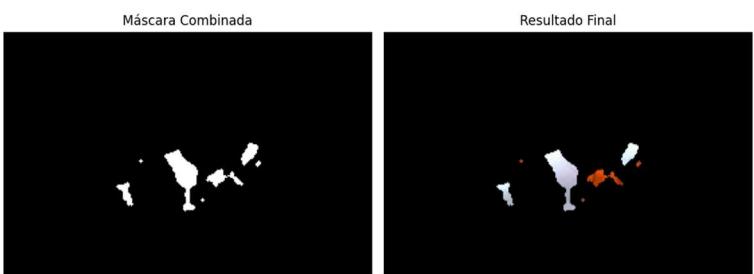
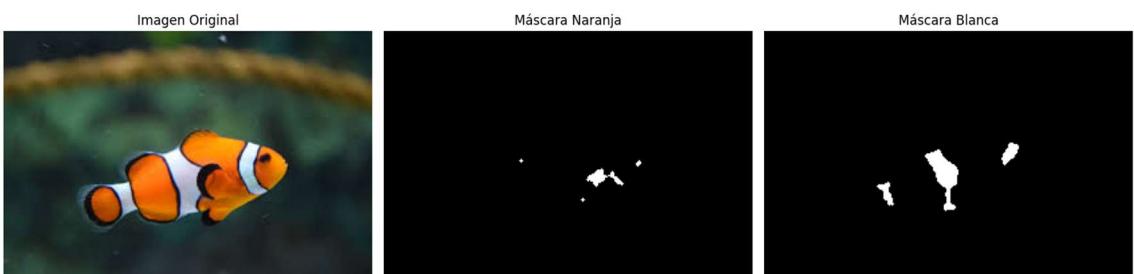
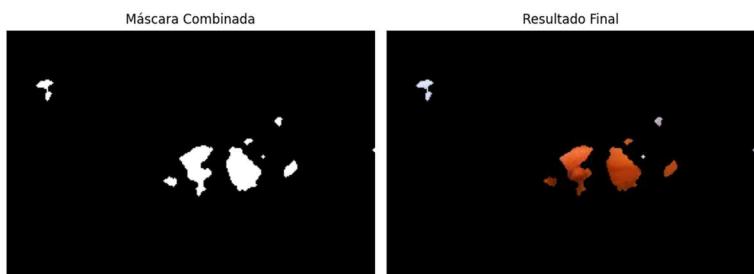


EJ2

1. Modifique el programa del ejercicio 1 para incluir la segmentación indicada en el código anterior.
2. Compruebe que funciona bien con las 6 imágenes de nemo que hay en la carpeta “./images/”.



Segmentación de nemo2.jpg



EJ3

1. Modifique el programa del ejercicio 2 para añadir una segunda segmentación de los tonos blancos y obtenga una máscara mask_white

2. Combine ambas máscaras y muestra el resultado incluir la segmentación indicada en el código anterior.

```
final_mask = mask_orange + mask_white
```

```
final_result = cv2.bitwise_and(nemo, nemo, mask=final_mask)
```

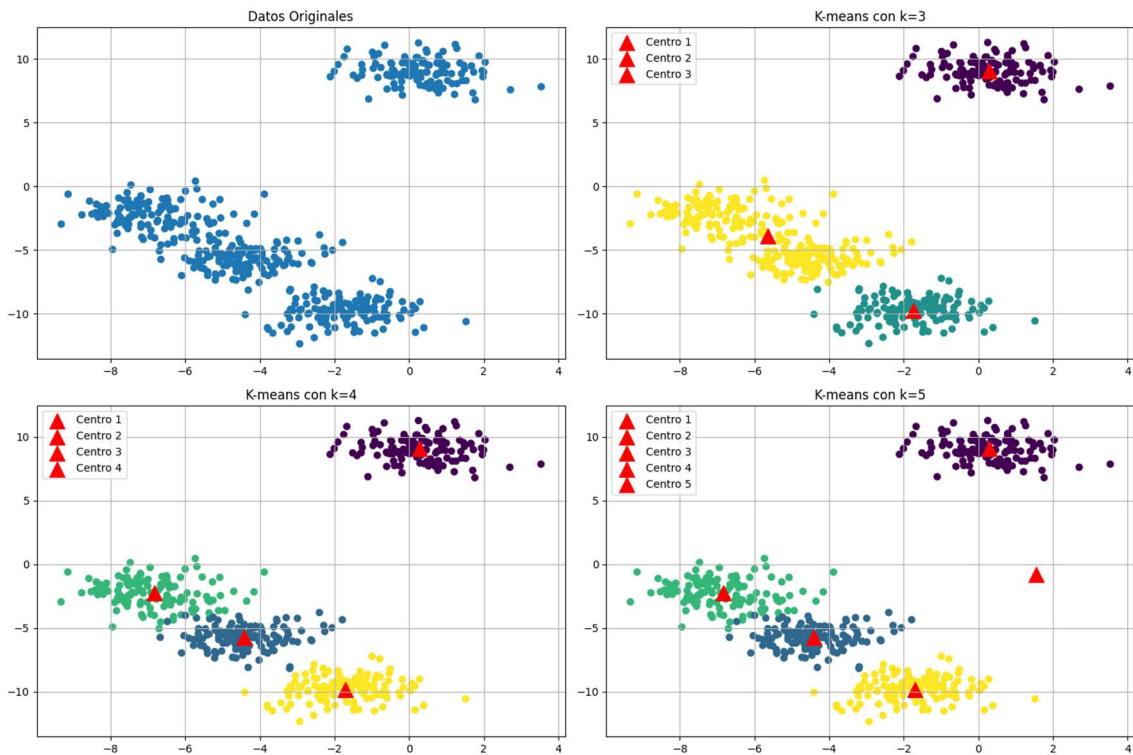
3. Compruebe que funciona bien con las 6 imágenes de nemo que hay en la carpeta “./images/”.



2.

Ej 4

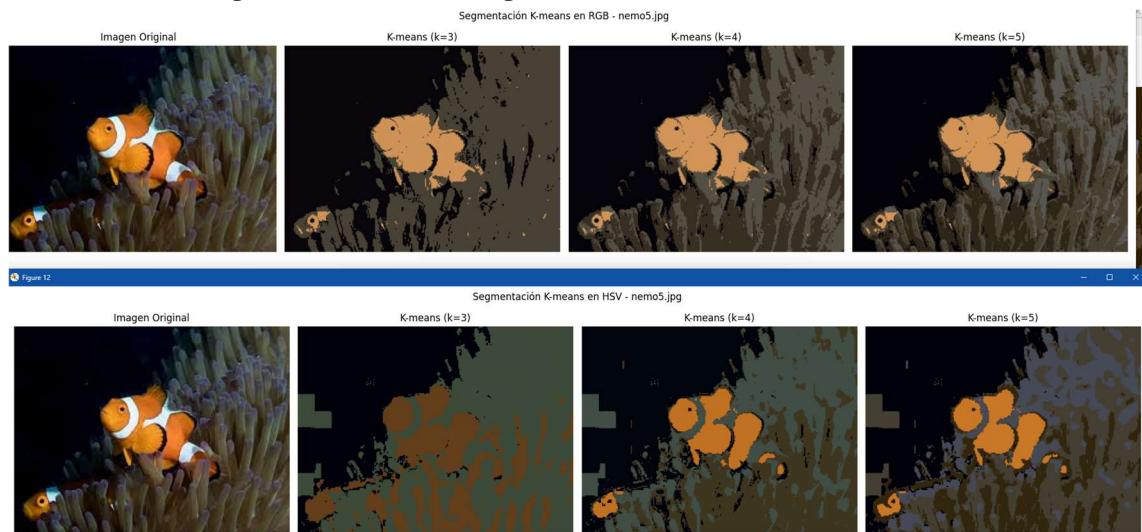
1. Tome el script ‘k-means_alt.py’ disponible en PoliformaT .
2. Ejecute el código con $k=3, 4$ y 5 clases y compruebe el resultado



EJ 5

1. Modifique el código anterior y cree el script “k-means-image.py” que permita hacer el algoritmo k_Means con datos de tipo imagen. Ahora los datos serán de tamaño ($n_pixeles, 3$) correspondientes a las 3 coordenadas de color (RGB, HSV, etc..).
2. En caso necesario, modifique la visualización en la función “plt.scatter()”.

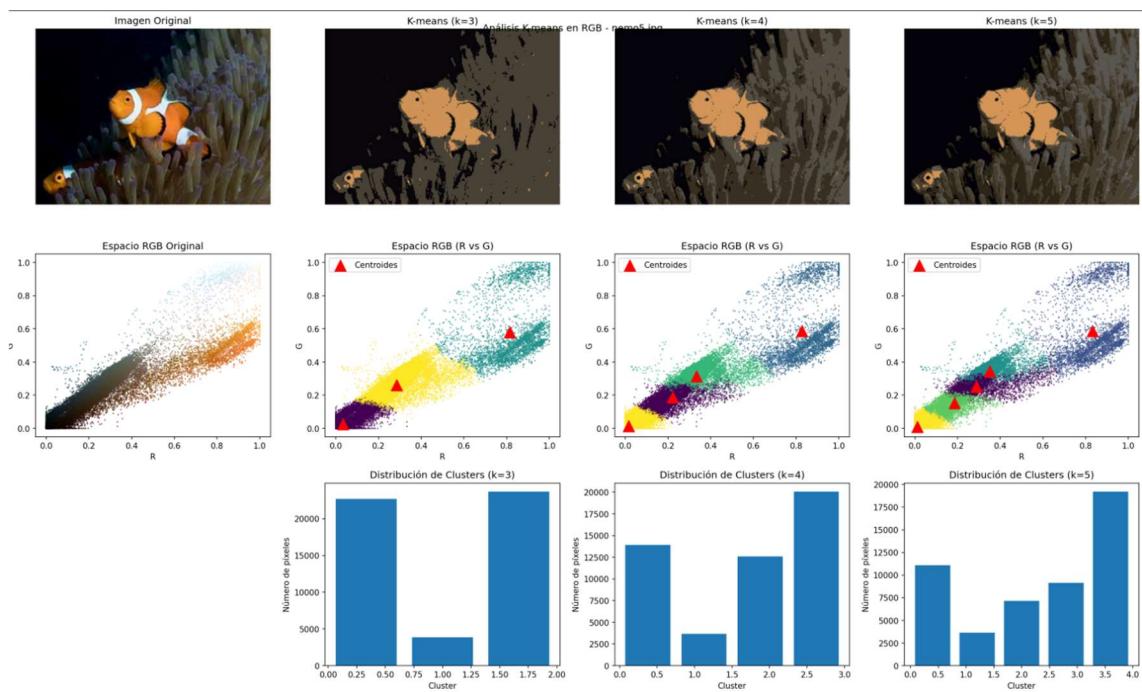
3. Utilice este algoritmo con las imágenes de nemo anteriormente indicadas.



El resto de imágenes estas en la carpeta Practica 5 - > imágenes

EJ 6

1. Busque en internet el algoritmo k-Means++ y documente como sería
2. Busque un script Python de ese algoritmo y pruébelo.



EJ 7

1. Copie el script anterior en un archivo ‘MeanShift.py’ y ejecútelo.
2. Compruebe el resultado.
3. Pruebe distintos valores del parámetro ‘bandwidth’ (20, 30, 40, ..).
4. Compruebe el resultado con otras imágenes de nemo.

