Ejercicios Basicos Procesamiento de Imagenes

Ricardo Carrillo Sanchez
Laboratorio de Procesamiento de Imagenes
Facultad de Ingenieria, Universidad Nacional Autonoma de Mexico
Ciudad de México, México
r.carrillosanchez@comunidad.unam.mx

***Abstract*—Los ejercicios realizados en esta breve actividad son una serie de recopilaciones para el manejo básico de imágenes, así como una introducción a la librerías más utilizadas en Python para el manejo de imágenes.**

I. Actividades

1. *Abrir y escribir una imagen a un archivo*

```
1 f = misc.face()
2 io.imwrite('Resources/face.png', f)
3 plt.imshow(f)
4 plt.show()
```

✓ 1.7s

2. *Creación de un arreglo numpy de un archivo de imagen*

```
1 face = misc.face()
2 io.imwrite('Resources/face.png', face) # Se salva la imagen como png
3 face = io.imread('face.png') # Se lee la imagen pgn
4 print(type(face))
5 face.shape, face.dtype
```

[13] ✓ 0.7s

```
<class 'imageio.core.util.Array'>

((768, 1024, 3), dtype('uint8'))
```

¿De qué tipo es la variable face?
La variable face es de tipo Array de la clase image.io

¿Qué resultado arroja face.shape?
face.shape nos arroja la dimensionalidad de la imagen, en concreto hablamos del color RGB además de la resolución de la imagen.

Si fuera una imagen en tonos de gris, ¿cuál sería el resultado esperado de face.shape?

Se mostrará una forma en una sola dimensión y si imprimimos los valores del archivo se verá que van de un rango de 0 255 debido a que el tipo es de entero a 8 bits.

3. *Abrir archivos raw*

```
1 face.tofile('Resources/face.raw') # Se crea el archivo raw binario
2 face_from_raw = np.fromfile('face.raw', dtype=np.uint8)
3 print(face_from_raw.shape)
4 face_from_raw.shape = (768, 1024, 3)
```

[18] ✓ 0.2s

```
(2359296,)
```

¿Qué resultado arroja la primera instrucción face_from_raw.shape?

Muestra la cantidad de bytes totales de la imagen, dado que raw esta en caracteres codificados en cierto formato estos se despliegan como un “archivo de texto”.

4. *Despliegue de imágenes*

```
f = misc.face(gray=True) # Lee la imagen en escala de grises
plt.imshow(f, cmap=plt.cm.gray)
```

[19] ✓ 0.3s

```
<matplotlib.image.AxesImage at 0x27dff78afa0>
```

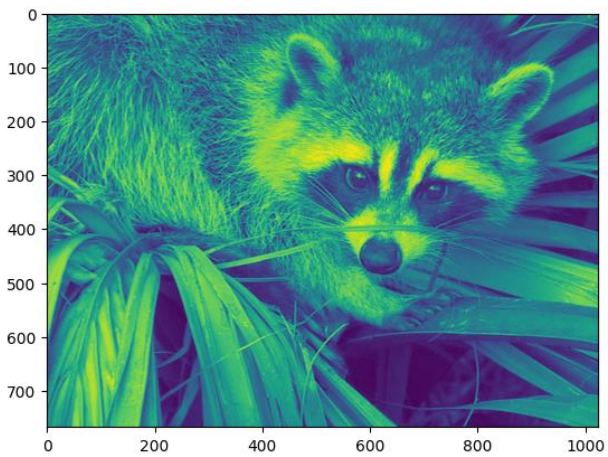
¿Qué pasa si al desplegar la imagen con plt.imshow no se especifica el mapa de color plt.cm.gray?

Acorde a la documentación de matplotlib, esta se ignorará y se interpretara como el colormap viridis

```
f = misc.face(gray=True) # Lee la imagen en escala de gri
plt.imshow(f)
```

[22] ✓ 0.3s

```
<matplotlib.image.AxesImage at 0x27e019be5e0>
```



¿Cuál es el mapa de color default de imshow?

Viridis

Imprima la forma de f (propiedad shape de f) y compare contra la forma de la misma imagen a color

```
print(f'Forma BW: {f.shape}')
print(f'Forma RGB: {face.shape}')
```

[24] ✓ 0.4s

```
Forma BW: (768, 1024)
Forma RGB: (768, 1024, 3)
```

En este caso vemos que la forma del RGB contiene un elemento mas que corresponde a los canales de RGB, mientras que BW únicamente presenta las dimensiones de la imagen, por ende solo tiene un único canal.

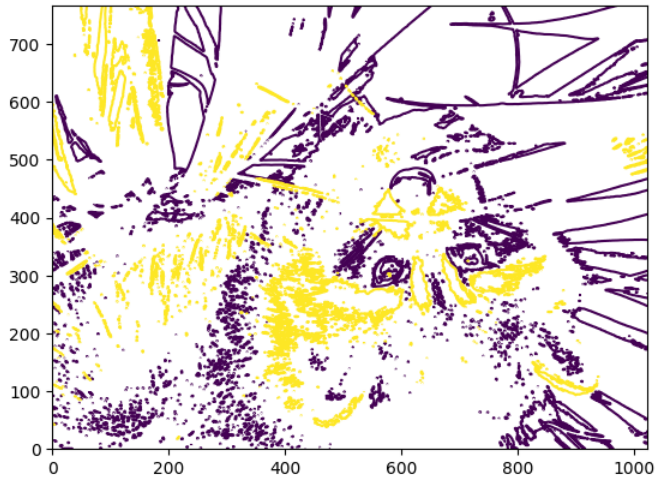
```
plt.imshow(f, cmap=plt.cm.gray, vmin=30, vmax=200)
# Remueve los ejes y las marcas (ticks)
plt.axis('off')
```

✓ 0.1s

```
(-0.5, 1023.5, 767.5, -0.5)
```

```
1 plt.contour(f, [50, 200])
```
0.2s

```
<matplotlib.contour.QuadContourSet at 0x27e01a5bdf0>
```



Investigue y ponga una breve descripción sobre la instrucción contour.

Los gráficos de contorno se utilizan ampliamente para visualizar la densidad, las altitudes o las alturas de la montaña, así como en el departamento meteorológico. Debido a su uso tan amplio, matplotlib.pyplot.contour dibuja líneas de contorno y contornos rellenos, respectivamente.

5. *Manipulaciones básicas*

```
# Obtiene el valor de un pixel de la imagen
face = misc.face(gray=True)
face[0, 40]
```
[53] 0.1s

```
127
```

¿Cuánto vale el pixel face[0, 40]?

Vale 127, correspondiente a gris intermedio.

¿Qué efecto tiene la instrucción face[100:120] = 255 en la imagen de abajo?

Equivale a tomar roda una fila de pixeles y hacer su nuevo valor sea blanco.

```
# Accesando secciones de la imagen
face[10:13, 20:23]
face[100:120] = 255
```
[58] 0.7s

```
plt.imshow(face, cmap = plt.cm.gray)
```
[59] 0.2s

```
<matplotlib.image.AxesImage at 0x27e203e9f40>
```

Pinte una franja vertical gris en la imagen que vaya de la columna 200 a la columna 220. Tip: en los índices tiene que elegir todas las filas con : y luego indicar las columnas deseadas.

```
# Accesando secciones de la imagen
face = misc.face(gray=True)
face[:200 , :220] = 127
plt.imshow(face, cmap = plt.cm.gray)
```
[73] 0.3s

```
<matplotlib.image.AxesImage at 0x27e2028be20>
```

¿Cuánto vale lx, ly?

Equivalen a la longitud de la imagen respectivamente en x y en y.

¿Qué efecto tiene en la imagen la instrucción face[range(400), range(400)] = 255?

Crea un círculo Den la imagen, habiendo puesto el “fondo” en negro por medio de la aplicación de una máscara.

```
lx, ly = face.shape
X, Y = np.ogrid[0:lx, 0:ly]
mask = (X - lx / 2) ** 2 + (Y - ly / 2) ** 2 > lx * ly / 
# Masks
face[mask] = 0
# Indexado con rangos
face[range(400), range(400)] = 255
plt.imshow(face, cmap = plt.cm.gray)
```

0.2s

```
<matplotlib.image.AxesImage at 0x27e209e6c70>
```

Modifique el código para que la máscara sea un círculo más pequeño y despliegue el resultado.

Dado que la mascara representa la ecuación de la circunferencia en su forma canónica únicamente habrá que reducir el componente final de la desigualdad (radio²) dividiéndolo por un factor mas grande, en este caso 8.

```
lx, ly = face.shape
X, Y = np.ogrid[0:lx, 0:ly]
mask = (X - lx / 2) ** 2 + (Y - ly / 2) ** 2 > lx * ly / 8
# Masks
face[mask] = 0
# Indexado con rangos
face[range(400), range(400)] = 255
plt.imshow(face, cmap = plt.cm.gray)
```

0.1s

```
<matplotlib.image.AxesImage at 0x27e20e50ac0>
```

II. INFORMACION ESTADISTICA

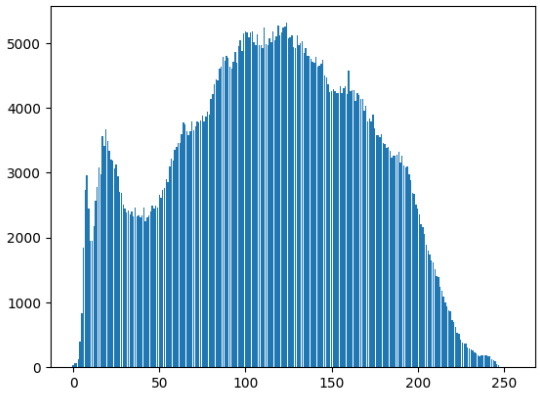
Despliegue el histograma de la imagen en grises del mapache original (es decir, antes de poner las franjas y de aplicar la máscara) usando:

```
hist, bins = np.histogram(face, bins=256, range=(0,256))
plt.bar(bins[0:-1], hist)
```

```
hist, bins = np.histogram(face, bins=256, range=(0,256))
plt.bar(bins[0:-1], hist)
```

[82] 0.3s

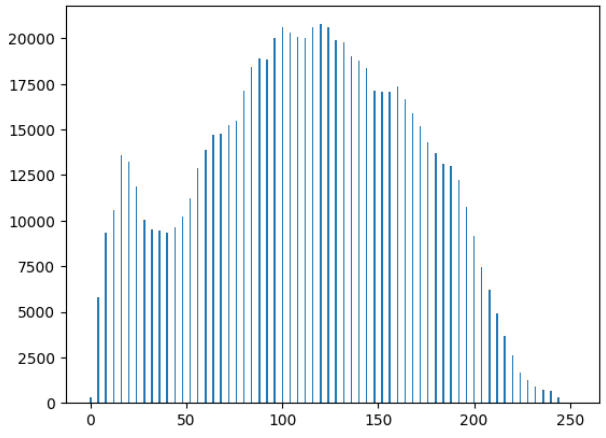
```
<BarContainer object of 256 artists>
```



Modifique el código para que sólo se tengan 64 bins en el histograma y muestre el histograma resultante.

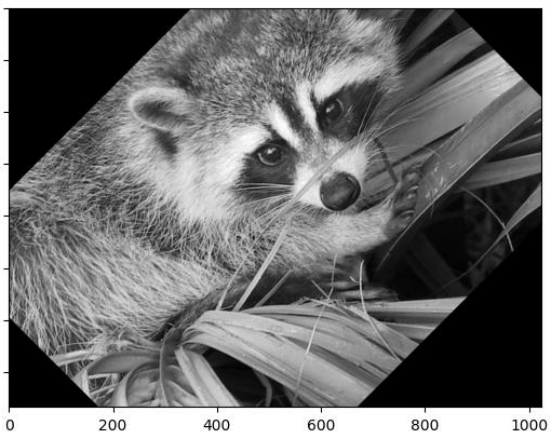
```
1 hist, bins = np.histogram(face, bins=64, range=(0,256))
2 plt.bar(bins[0:-1], hist)
```
[84] ✓ 0.1s

```
<BarContainer object of 64 artists>
```




```
1 plt.imshow(rotate_face_noreshape, cmap = plt.cm.gray)
```
[100] ✓ 0.2s

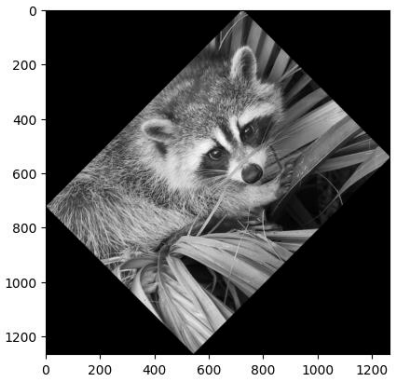
```
<matplotlib.image.AxesImage at 0x27e264839a0>
```



III. TRANSFORMACIONES GEOMETRICAS

```
1 face = misc.face(gray=True)
2 lx, ly = face.shape
3 # Recorte
4 crop_face = face[lx // 4: - lx // 4, ly // 4: - ly // 4]
5 # up <-> down Voltear verticalmente
6 flip_ud_face = np.flipud(face)
7 # Rotación
8 rotate_face = rotate(face, 45)
9 rotate_face_noreshape = rotate(face, 45, reshape=False)
10 plt.imshow(rotate_face, cmap = plt.cm.gray)
11
```
[99] ✓ 0.4s

```
<matplotlib.image.AxesImage at 0x27e26433340>
```



Investigue el operador // y ponga una breve descripción.

Realiza una división entera, truncando el resultado.

EJ. 7//2 = 3

¿Qué efecto tiene el signo negativo en

crop_face = face[lx // 4: - lx // 4, ly // 4: - ly // 4]?

Tiene el objetivo de que se hagan los bordes de la imagen “iguales” de modo que hace un acercamiento debido a la división entera.

Voltee la imagen original del mapache, pero esta vez horizontalmente, de izquierda a derecha.

```
1 face = misc.face(gray=True)
2 lx, ly = face.shape
3 # Recorte
4 crop_face = face[lx // 4: - lx // 4, ly //4: - ly // 4]
5 flip_lr_face = np.fliplr(face)
6 plt.imshow(flip_lr_face, cmap = plt.cm.gray)
```
[148] ✓ 0.4s

```
<matplotlib.image.AxesImage at 0x27e2c7e3520>
```

IV. Conclusiones

Esta actividad ha resultado de mucha utilidad ya que permitió operar y ver la representación numérica en lenguajes como Python de imágenes, por lo que esto da partida a muchas aplicaciones en patrones e inteligencia artificial.

References

[1] "NumPy documentation — NumPy v1.23 Manual", Numpy.org, 2022. [Online]. Available: https://numpy.org/doc/1.23/. [Accessed: 28- Aug- 2022].

[2] J. H, "The Basics of Indexing and Slicing Python Lists", Medium, 2022. [Online]. Available: https://towardsdatascience.com/the-basics-of-indexing-and-slicing-python-lists-2d12c90a94cf. [Accessed: 28- Aug- 2022].

[3] scikit-image 0.19.2 docs — skimage v0.19.2 docs", Scikit-image.org, 2022. [Online]. Available: https://scikit-image.org/docs/stable/. [Accessed: 28- Aug- 2022].

[4] 2.6. Image manipulation and processing using Numpy and Scipy — Scipy lecture notes", Scipy-lectures.org, 2022. [Online]. Available: https://scipy-lectures.org/advanced/image_processing/. [Accessed: 29- Aug- 2022].