Observaciones generales en base a consultas recibidas en clase

1) Pregunta: ¿Cómo se obtiene una imagen en gris a partir de una imagen en color?

Se trata de un promedio **ponderado** de las intensidades de cada color. Una ponderación estimada es:

$$Y = R*0.3 + G*0.59 + B*0.11$$

A qué se debe la diferencia de los pesos?

El ojo es más sensible al verde y menos sensible al azul.

Cómo se puede lograr la imagen en gris con el Python opencv sin hacer cálculos?

- **La solución más sencilla con la librería opencv es:**

```
Import cv2
Imagengris=cv2.imread('nombre archivo",0)
```

También:

```
Imagengris=cv.cvtColor('nombre archivo",cv.COLOR_BGR2GRAY)
```

Archivo Ejemplo

```
import matplotlib.pyplot as plt
imgris1=cv2.imread('lenna.jpg',0)
imgris2=cv2.imread('lenna.jpg', cv2.IMREAD_GRAYSCALE)
print((imgris1-imgris2).max())
imgcolor=cv2.imread('lenna.jpg')#llevo a memoria la imagen a color
imggris3=cv2.cvtColor(imgcolor,cv2.COLOR_BGR2GRAY)#convierto en gris
imgrgba=cv2.cvtColor(imgcolor,cv2.COLOR_BGR2RGB)#convierto de RGB a BGR
imgrgbb=imgcolor[:,:,[2,1,0]]# opencv trabaja con BGR, debemos pasarlo a RGB para usar matplotlib
#el plano de R recibe el plano en posición 2 #G queda igual en plano 1 #y B recibe el plano 0
#Mostrar imágenes con matplotlib
fig = plt.gcf()
```

Observaciones según preguntas en clase

```
fig.set_size_inches(20,30)


plt.subplot(321),plt.imshow(imgris1,cmap='gray')#para las imágenes grises aclarar cmap='gray'
plt.subplot(322),plt.imshow(imgris2,cmap='gray')
plt.subplot(323),plt.imshow(imgcolor)
plt.subplot(324),plt.imshow(imggris3,cmap='gray')
plt.subplot(325),plt.imshow(imgrgba)
plt.subplot(326),plt.imshow(imgrgbb)
plt.show()
```

2) *Error común: no llevar a memoria la imagen porque la misma no está en la misma carpeta del ipnyb que está corriendo!*

Las imágenes deben estar en la misma carpeta donde está el archivo que estamos corriendo. Lamentablemente, el Jupyter no informa el error. A lo sumo, al intentar operar indica None o NoneType has no attribute…Pero no dice que no encuentra el archivo.

Solución:
Para generalizar, poner siempre el camino. Utilizar una variable con nombre significativo. Es común usar "path".

***Recomendación* para no duplicar innecesariamente las imágenes y por claridad:**
Poner todas las imágenes en una carpeta e utilizar una variable path que remita a dicha carpeta. Esto evita errores.

<u>Archivo Ejemplo</u>

```
import cv2
import matplotlib.pyplot as plt
path='C:\Images\lenna.jpg'
imgris=cv2.imread(path,0)
plt.imshow(imgris,cmap='gray')
#plt.axis('off') #descomentar si no queremos ver las escalas
plt.show()#para evitar mensajes innecesarios
```

3) **Revisar los nombres de variables usadas!!! Lamentablemente, en el entorno que usan no siempre informa claramente que la variable no existe. Cuando lo hace dice, por ejemplo:**

```
a=5
b=6
c=d+a
```

```
    4 print(c)

NameError: name 'd' is not defined
```

4) **Recomendación:**
   Antes de calcular bordes hacer un suavizado!

   Si usan la función de Canny (en el caso de opencv es cv.canny) no es necesario porque es el primer paso que realiza!

5) **Recomendación personal:**
   No es necesario memorizar los comandos. No desesperar! Esto surge con el tiempo. Hay que aprender los pasos que se deben seguir, y a analizar las imágenes. **Piensen que si usan otra librería, los comandos serán diferentes en cuanto a su sintaxis! Importan los CONCEPTOS!**
   Cuando se usa la misma librería durante algún tiempo, se recuerdan los comandos.

Umbralizado:

Es importante diferenciar:

-Global

-Local

Global: (un único umbral para toda la imagen), afortunadamente existen métodos **automáticos**, en los que el usuario no necesita mayor conocimiento. El que tiene su propia función en opencv, es el método de **OTSU.**

Nobuyuki Otsu: *A threshold selection method from grey level histograms*. In: *IEEE Transactions on Systems, Man, and Cybernetics*. New York 9.1979, S.62–66. ISSN 1083-4419

Si el histograma es bimodal, se puede determinar por inspección visual de este el umbral (valor entre los dos modos), esto es artesanal, requiere más conocimiento.

Preguntas recibidas:

Por qué convertir a uint8:

Porque se usaron variables booleanas: True- False. Por lo tanto, no puedo construir el histograma. Una vez convertido a uint8, tengo dos valores: 0 y 1. Si hubiera usado if (como se hizo la otra clase), sin pasar por valores booleanos, hubiera sido directo.

Por qué multiplicar por 255:

Si no lo hacemos debemos modificar los parámetros que pasamos para hacer el histograma. Ya no van a ser 256 bins sino 2. Y las intensidades [0,2] (son posibles solo 0 y 1, pero el Python usa siempre intervalos abiertos a derecha).

Archivo Ejemplo

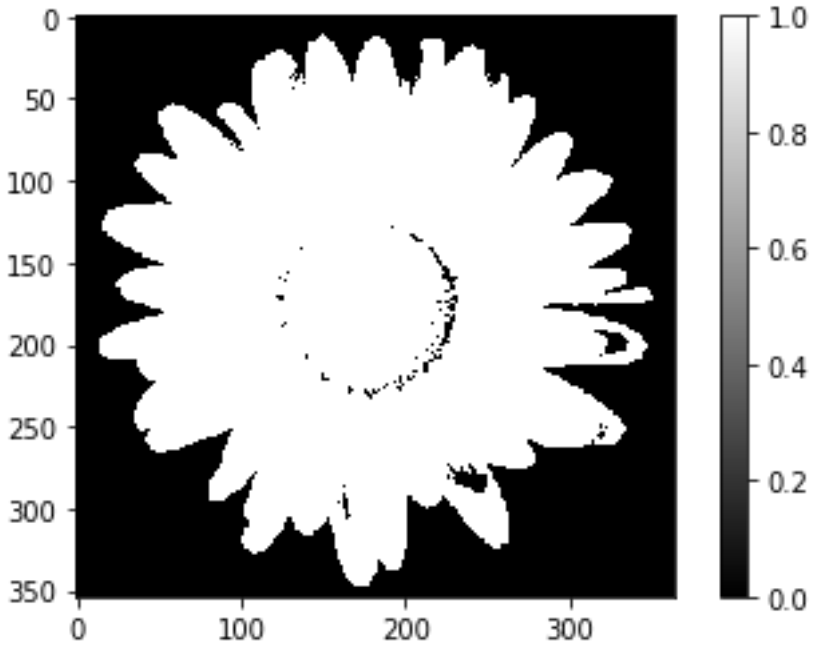
```
import cv2

import matplotlib.pyplot as plt

img = cv2.imread('daisy.jpg',0)#El 0 indica que devuelve imagen monocroma directamente

hist = cv2.calcHist([img],[0],None,[256],[0,256])

plt.plot(hist)

plt.xlabel("Intensidad de Iluminación")

plt.ylabel("Cantidad de Píxeles")

plt.title("Histograma hecho con opencv")

plt.show()

#Ingreso umbral por teclado de acuerdo al histograma

umbral=int(input('ingrese un valor de umbral:' ))

#Por ejemplo 100

binariadaisy=(img>=umbral)

print(binariadaisy.shape)

print(binariadaisy.max())

binariadaisy=binariadaisy.astype('uint8')

print(binariadaisy.max())

histbinaria=
cv2.calcHist([binariadaisy],[0],None,[2],[0,2])

print(histbinaria)

plt.imshow(binariadaisy,cmap='gray')

plt.colorbar()

plt.show()
```

Observaciones según preguntas en clase

```
plt.imshow(img,cmap='gray')
```

Resultado Corrida:

```
(354, 364)
True
1
[[57883.]
 [70973.]]
```



Se hizo Binarización Directa: **Blancos** **1 o 255**

Negros **0**

La inversa es el complemento: Blancos 0

Negros: 1 o 255

Funciones de OpenCV que binarizan globalmente No Automáticas

```
ret1,binaria1 = cv2.threshold(img,umbral,255,cv2.THRESH_BINARY)
```

```
ret2,binaria2 = cv2.threshold(img,umbral,255,cv2.THRESH_BINARY_INV)
```

Se le pasa la imagen, el umbral, el mayor valor deseado (en este caso 255) y si es directo o inverso.

Binarización Global Automática: OpenCV implementó OTSU.

```
ret,binaria= cv2.threshold(img,0,255,cv2.THRESH_OTSU)
```

```
ret,binaria=
cv2.threshold(img,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
```

En el segundo caso, mediante la suma de métodos se indica que utilice Otsu, pero con umbralizado inverso.

No siempre funciona. Esto se puede anticipar observando el histograma.

Binarización Local Adaptable con OpenCV:

```
binaria=
cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY,21,6)
```

```
binaria=
cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,21,6)
```

Si bien son automáticas es necesario que el usuario defina el tamaño del bloque y la constante que se resta (puede ser 0). Serían **casi automáticas.**

Causas por las que se usa umbralizado local

En muchas aplicaciones, no se puede obtener un umbral global para un histograma, es decir, no se puede obtener una buena segmentación con un único umbral para toda la imagen. Esto ocurre cuando el fondo no es constante y el contraste de los objetos varía en la imagen, por lo que la umbralización global daría buenos resultados en una parte de la imagen, pero para el resto de la imagen, no sería la adecuada.

Observaciones según preguntas en clase

Se puede, por ejemplo, por inspección, dividir la imagen convenientemente y en cada bloque, calcular OTSU (o decidir por el histograma). Con este procedimiento pueden aparecer diferencias en las fronteras y se puede hacer muy largo el procesamiento.

Otra solución es calcular umbrales locales en forma **casi automática**. Es lo que hacen estas dos funciones del OpenCV. Para esto se examinan las intensidades de la imagen en los alrededores de cada píxel. La umbralización local es computacionalmente más costosa que la global. Es muy útil a la hora de segmentar objetos en fondos no homogéneos, y para extraer regiones muy pequeñas y dispersas.

binaria = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY,21,6)

binaria = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY,21,6)

Explicación parámetros

#imagen,

#valor máximo asignado a los píxeles que cumplen la condición de ser mayores que el umbral.

#tipo de adaptación (mean o gaussiana),

#tipo de umbralizado: común o inverso

#tamaño del bloque que se estudia (21 en el ejemplo)

#constante que se resta del valor medio ADAPTIVE_THRESH_MEAN_C (o del medio-gaussiano),

El umbral T(x,y) (umbral en el píxel (x,y)) **es la media** de los valores de la vecindad de (x,y) considerando en este caso como vecinos un block de tamaño 21x21, menos una constante, en este caso 6. En el caso de ADAPTIVE_THRESH_GAUSSIAN_C la vecindad está pesada según una ventana gaussiana. Se usa la desviación estándar para determinar el kernel gaussiano. (Similar a lo visto en filtros de suavizado).

Por qué conviene restar una constante? Muchas veces, la diferencia de intensidades en los vecinos es muy pequeño (rango reducido de intensidades en zonas uniformes) y el valor medio es cercano al valor de intensidad del píxel (x,y). Entonces, no sirve como umbral la media, aparece ruido. Esto se mejora si el umbral no es el valor medio sino (Medio-Constante). Entonces, todos los píxeles que están en una vecindad uniforme son enviados a una misma zona. El valor del umbral no tiene que ser muy grande.

Ventana chica o grande? Si la ventana es grande, puede no ser tan bueno el resultado si está afectada por el gradiente de iluminación, pero muy chica tiene más trabajo

computacional y puede no aportar mejor información. También depende del tamaño del objeto que se desea separar del fondo.

Ejercicio: repetir el umbralizado local realizado para la imagen "sudoku.png", poniendo C=0.

Pasos a seguir si desea programar el algoritmo, pero usando la mediana y no la media para determinar el umbral.

1- **Convolucionar la imagen con un kernel adecuado para calcular la media en cada píxel.**
2- **Restar la imagen original de la convolucionada.**
3- **Binarizar considerando la constante C.**

Ejercicio; repetir el ejercicio de umbralizado: mostrar histograma, aplicar umbralizado global y local para la imagen martillo.jpg que se adjunta.