OperacionesMorfologicas

July 30, 2024

#

Aplicacion de algoritmos de segmentacion y cluster en imagenes agricolas, mediante el uso de herramientas de computo

> MODULO 2: OPERACIONES MORFOLOGICAS AUTOR: LOPEZ ESTEBAN MIGUEL FECHA: 06 DE JULIO DEL 2024

Este módulo abarcara diferentes operaciones morfológicas utilizadas en el procesamiento de imágenes. Estas técnicas son herramientas de gran utilidad para el análisis y mejora de imágenes, especialmente en imágenes binarias y en escala de grises, que es donde más se notan sus aportaciones de manera visual, por esta razón en este módulo estaremos trabajando con imágenes binarias de manera que podamos visualizar los efectos de cada operación.

0.0.1 INDICE

###

Kernel

El kernel, también conocido como elemento estructurante, es una pequeña matriz que se ocupa en las operaciones morfológicas. Este elemento ayuda a definir la vecindad de cada píxel en la imagen. Este elemento es sumamente importante ya que determinara el funcionamiento de las operaciones.

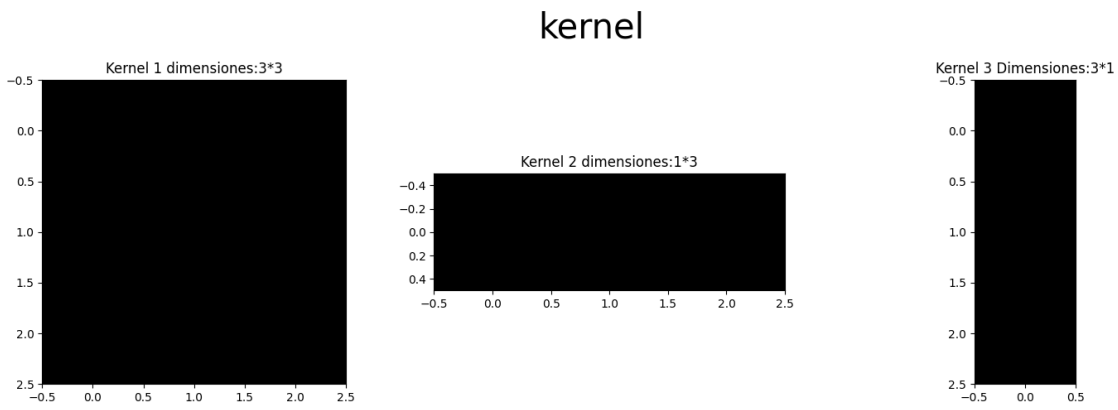
Como ya definimos, el kernel es una matriz la cual cuenta con un tamaño reducido, y suele tener dimensiones impares, como 3x3 o 5x5. Esta matriz puede tener valores '0' o '1' . los valores de '1' serán los que se consideren al momento de hacer las operaciones. En las diferentes operaciones que veremos en este módulo el kernel tiene el mismo funcionamiento, este se sitúa sobre el primer píxel de la imagen y verifica el valor de los pixeles situados debajo de él, dependiendo de la operación este modificara el valor del pixel central, posteriormente este se recorrerá a la derecha o hacia abajo dependiendo su posición actual, es por eso que mientras el kernel tenga un tamaño mayor la

modificación de la imagen será más notable, además de que la forma que tenga este afectara en las operaciones.

Debido a la información vertida anteriormente, crearemos 3 kernel que tengan formas deferentes con la intención de observar cómo es que la forma del kernel afecta en la operación, para esto crearemos matrices de un tamaño dado que llenaremos con '1' mediante la librería numpy.

[ ]:
```
#IMPORTAMOS LAS LIBRERIAS CON LAS QUE TRABAJAREMOS A LO LARGO DEL MODULO.
import cv2
import matplotlib.pyplot as plt
import numpy as np
```

[ ]:
```
#CREAMOS 3 KERNEL DE DIFERENTES TAMAÑOS
kernel1 = np.ones((3,3),dtype='uint8')
kernel2 = np.ones((1,3),dtype='uint8')
kernel3 = np.ones((3,1),dtype='uint8')

#MOSTRAMOS LOS KERNEL
fig,axis = plt.subplots(1,3,figsize=(15,5))

#kernel 1
axis[0].imshow(kernel1,cmap='gray')
axis[0].set_title('Kernel 1 dimensiones:3*3')

#kernel 2
axis[1].imshow(kernel2,cmap='gray')
axis[1].set_title('Kernel 2 dimensiones:1*3')

#kernell 3
axis[2].imshow(kernel3,cmap='gray')
axis[2].set_title('Kernel 3 Dimensiones:3*1')

#agregamos titulo
plt.suptitle('kernel',fontsize=30)

plt.tight_layout()
plt.show()
```



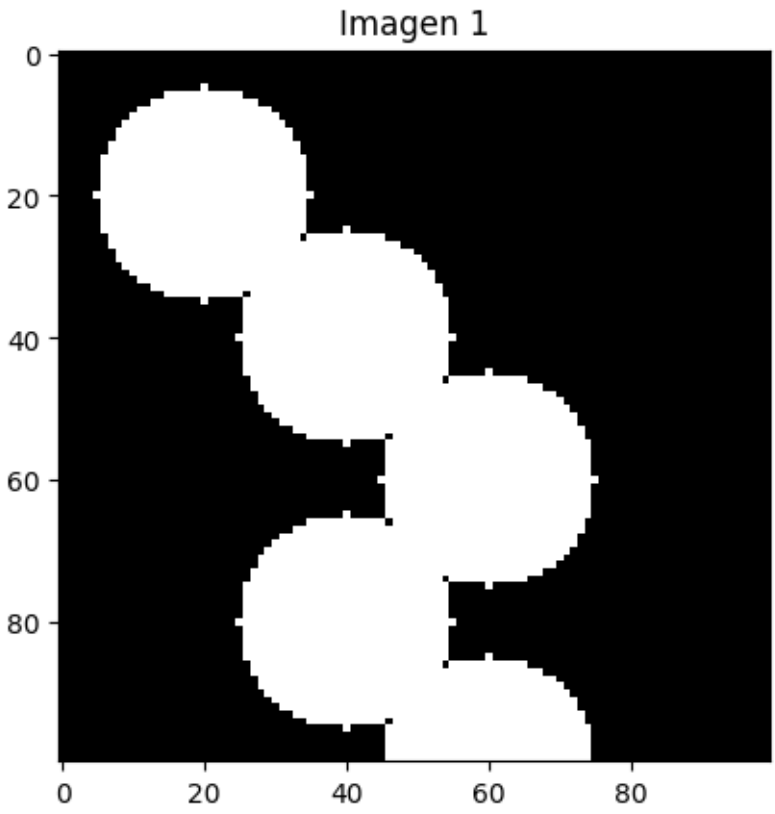
###

Erosión

La erosión es una operación morfológica que reduce el tamaño de las regiones del primer plano (blanco) de una imagen binaria. El kernel se desliza en toda la imagen y reduce los pixeles del primer plano, en palabras más sencillas la erosión elimina pixeles de los bordes de los objetos, esto ayuda a eliminar ruido, separar objetos que estén ligeramente unidos o a reducir el grosor de los objetos.

Para ilustrar esta operación crearemos una imagen binaria, en esta dibujaremos varios círculos, la intensión de aplicar esta operación es separar estos elementos con la intención de hacer más sencilla la cuantificación de estos objetos circulares.

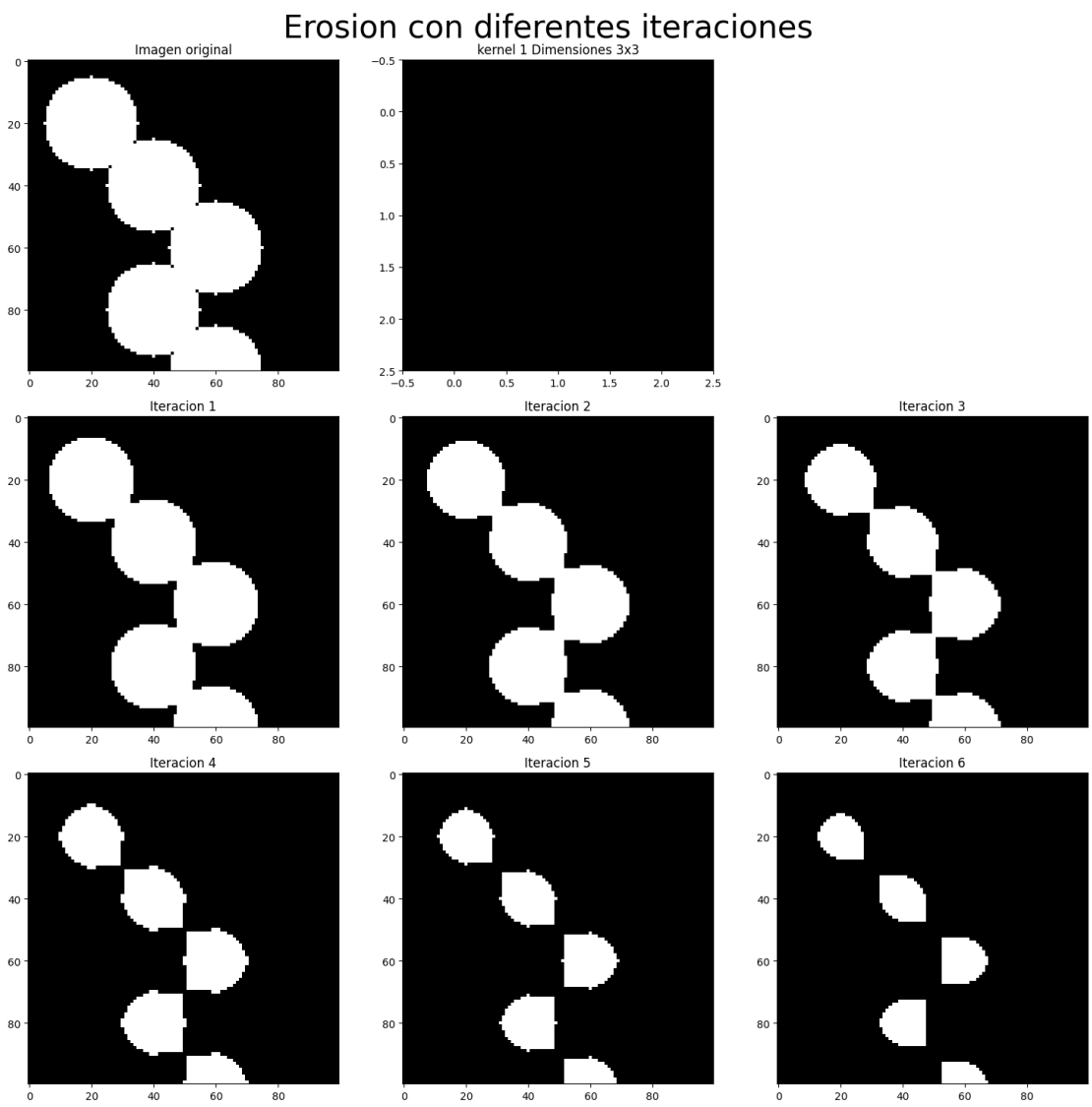
[ ]:
```
#creamos una imagen de tamaño 100*100 en color negro
img1 = np.zeros((100,100),dtype='uint8')

#dibujamos algunos circulos sobre nuestra imagen
cv2.circle(img1,(20,20),15,1,thickness=-1)
cv2.circle(img1,(40,40),15,1,thickness=-1)
cv2.circle(img1,(60,60),15,1,thickness=-1)
cv2.circle(img1,(40,80),15,1,thickness=-1)
cv2.circle(img1,(60,100),15,1,thickness=-1)

#mostramos nuestra imagen
plt.imshow(img1,cmap='gray')
plt.title('Imagen 1')
plt.show()
```



Como observamos dibujamos 5 círculos, lo cual para una persona es fácil identificar, sin embargo, una maquina contaría 1 solo objeto, ya que estos están unidos, por ende procederemos a separarlos mediante la erosión, para esto ocuparemos el kernel 1 creado en el tema anterior.

[ ]:
```
#erosionamos la imagen con el kernel 1 y con diferentes iteraciones
img1_erosionada = cv2.erode(img1,kernel1,iterations=1)
img2_erosionada = cv2.erode(img1,kernel1,iterations=2)
img3_erosionada = cv2.erode(img1,kernel1,iterations=3)
img4_erosionada = cv2.erode(img1,kernel1,iterations=4)
img5_erosionada = cv2.erode(img1,kernel1,iterations=5)
img6_erosionada = cv2.erode(img1,kernel1,iterations=6)

#mostramos las imagenes
#creamos las graficas
fig ,axs = plt.subplots(3,3,figsize=(15,15))

#imagen original
axs[0,0].imshow(img1,cmap='gray')
axs[0,0].set_title('Imagen original')

#kernel 1
```

```
axs[0,1].imshow(kernel1,cmap='gray')
axs[0,1].set_title('kernel 1 Dimensiones 3x3')

#desactivamos los ejes que no ocuparemos
axs[0,2].axis('off')

#iteracion 1
axs[1,0].imshow(img1_erosionada,cmap='gray')
axs[1,0].set_title('Iteracion 1')

#iteracion 2
axs[1,1].imshow(img2_erosionada,cmap='gray')
axs[1,1].set_title('Iteracion 2')

#iteracion 3
axs[1,2].imshow(img3_erosionada,cmap='gray')
axs[1,2].set_title('Iteracion 3')

#iteracion 4
axs[2,0].imshow(img4_erosionada,cmap='gray')
axs[2,0].set_title('Iteracion 4')

#iteracion 5
axs[2,1].imshow(img5_erosionada,cmap='gray')
axs[2,1].set_title('Iteracion 5')

#iteracion 6
axs[2,2].imshow(img6_erosionada,cmap='gray')
axs[2,2].set_title('Iteracion 6')

#titulo y acomodo de las imagenes
plt.suptitle('Erosion con diferentes iteraciones', fontsize=30)
plt.tight_layout()
plt.show()
```



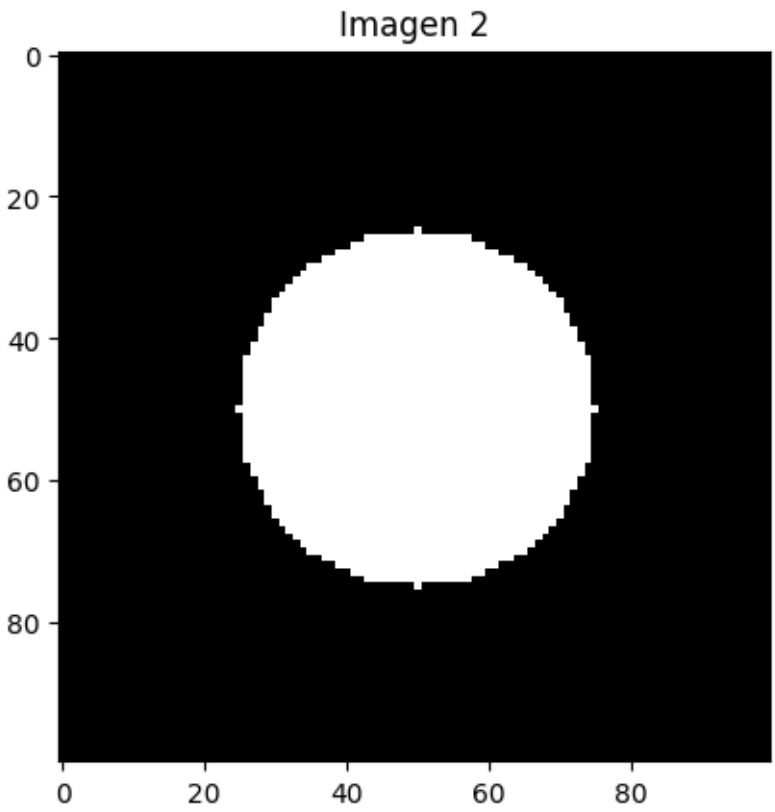
La iteración es la acción de repetir la operación sucesivamente, en este caso podemos observar que al haber hecho la operación una sola vez los objetos no se habían separado de manera adecuada, sin embargo, podemos observar que mientras más iteraciones hagamos en nuestra imagen podemos obtener mejores resultados.

En este caso se trabajó con el kernel1 y podemos ver que además de la separación de los elementos estos cambiaron su forma, podemos jugar con esto si también cambiamos el tamaño de nuestro kernel. Para hacer esto crearemos una imagen con un círculo inscrito, y erosionaremos nuestra imagen con los 3 kernel, con la intención de notar diferencias en los resultados.

[ ]:
```
#creams una imagen donde aplicaremos los kernel
img2 = np.zeros((100,100),dtype='uint8')

#dibujamos un circulo sobre nuestra imagen
```

```
cv2.circle(img2,(50,50),25,1,thickness=-1)

#mostramos la imagen
plt.imshow(img2,cmap='gray')
plt.title('Imagen 2')
```

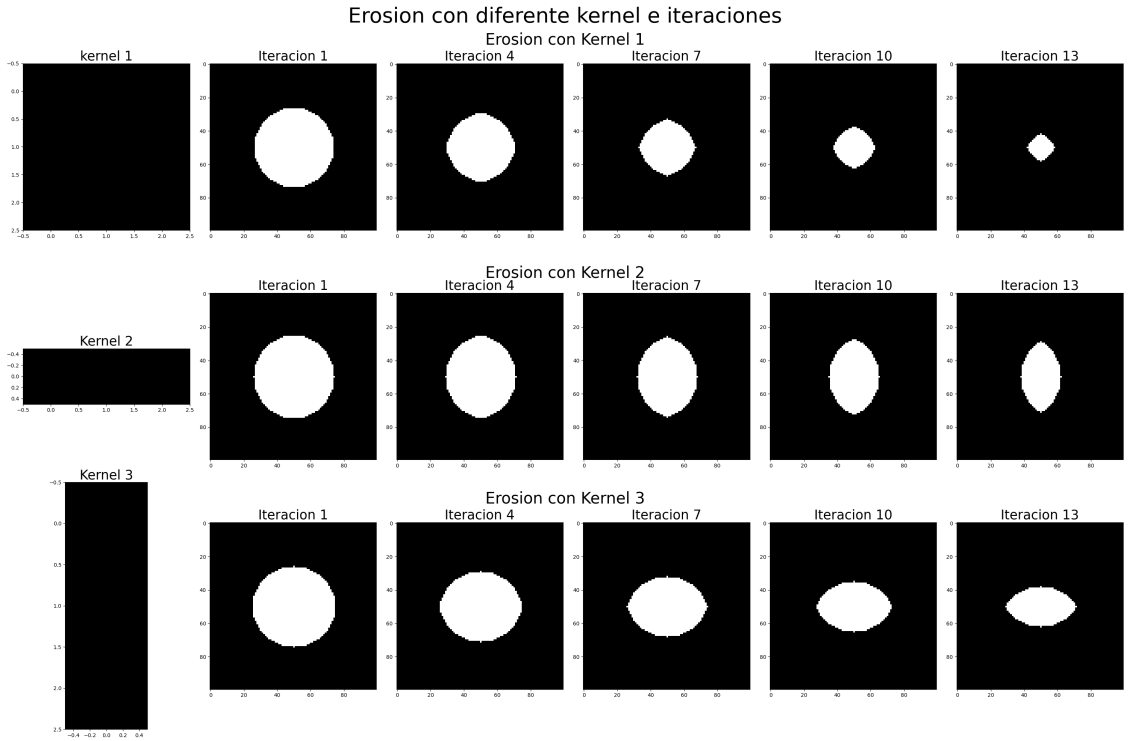
[ ]:
```
Text(0.5, 1.0, 'Imagen 2')
```



[ ]:
```
#erosionamos la imagen con los diferente kernel y diferentes iteraciones

#kernel 1
img_erosionada_k1_1 = cv2.erode(img2,kernel1,iterations=1)
img_erosionada_k1_2 = cv2.erode(img2,kernel1,iterations=4)
img_erosionada_k1_3 = cv2.erode(img2,kernel1,iterations=7)
img_erosionada_k1_4 = cv2.erode(img2,kernel1,iterations=10)
img_erosionada_k1_5 = cv2.erode(img2,kernel1,iterations=13)

#kernel 2
img_erosionada_k2_1 = cv2.erode(img2,kernel2,iterations=1)
img_erosionada_k2_2 = cv2.erode(img2,kernel2,iterations=4)
img_erosionada_k2_3 = cv2.erode(img2,kernel2,iterations=7)
```

```
img_erosionada_k2_4 = cv2.erode(img2,kernel2,iterations=10)
img_erosionada_k2_5 = cv2.erode(img2,kernel2,iterations=13)

#kernel 3
img_erosionada_k3_1 = cv2.erode(img2,kernel3,iterations=1)
img_erosionada_k3_2 = cv2.erode(img2,kernel3,iterations=4)
img_erosionada_k3_3 = cv2.erode(img2,kernel3,iterations=7)
img_erosionada_k3_4 = cv2.erode(img2,kernel3,iterations=10)
img_erosionada_k3_5 = cv2.erode(img2,kernel3,iterations=13)


#mostramos los resultados
fig,axis = plt.subplots(3,6,figsize=(30,20))

#kernel 1
axis[0,0].imshow(kernel1,cmap='gray')
axis[0,0].set_title('kernel 1',fontsize=25)
axis[0,1].imshow(img_erosionada_k1_1,cmap='gray')
axis[0,1].set_title('Iteracion 1',fontsize=25)
axis[0,2].imshow(img_erosionada_k1_2,cmap='gray')
axis[0,2].set_title('Iteracion 4',fontsize=25)
axis[0,3].imshow(img_erosionada_k1_3,cmap='gray')
axis[0,3].set_title('Iteracion 7',fontsize=25)
axis[0,4].imshow(img_erosionada_k1_4,cmap='gray')
axis[0,4].set_title('Iteracion 10',fontsize=25)
axis[0,5].imshow(img_erosionada_k1_5,cmap='gray')
axis[0,5].set_title('Iteracion 13',fontsize=25)

#kernel 2
axis[1,0].imshow(kernel2,cmap='gray')
axis[1,0].set_title('Kernel 2',fontsize=25)
axis[1,1].imshow(img_erosionada_k2_1,cmap='gray')
axis[1,1].set_title('Iteracion 1',fontsize=25)
axis[1,2].imshow(img_erosionada_k2_2,cmap='gray')
axis[1,2].set_title('Iteracion 4',fontsize=25)
axis[1,3].imshow(img_erosionada_k2_3,cmap='gray')
axis[1,3].set_title('Iteracion 7',fontsize=25)
axis[1,4].imshow(img_erosionada_k2_4,cmap='gray')
axis[1,4].set_title('Iteracion 10',fontsize=25)
axis[1,5].imshow(img_erosionada_k2_5,cmap='gray')
axis[1,5].set_title('Iteracion 13',fontsize=25)

#kernel 3
axis[2,0].imshow(kernel3,cmap='gray')
axis[2,0].set_title('Kernel 3',fontsize=25)
axis[2,1].imshow(img_erosionada_k3_1,cmap='gray')
axis[2,1].set_title('Iteracion 1',fontsize=25)
```

```
axis[2,2].imshow(img_erosionada_k3_2,cmap='gray')
axis[2,2].set_title('Iteracion 4',fontsize=25)
axis[2,3].imshow(img_erosionada_k3_3,cmap='gray')
axis[2,3].set_title('Iteracion 7',fontsize=25)
axis[2,4].imshow(img_erosionada_k3_4,cmap='gray')
axis[2,4].set_title('Iteracion 10',fontsize=25)
axis[2,5].imshow(img_erosionada_k3_5,cmap='gray')
axis[2,5].set_title('Iteracion 13',fontsize=25)

#agregamos titulo y subtitulos
plt.suptitle('Erosion con diferente kernel e iteraciones',fontsize=40)
fig.text(.5,.93,'Erosion con Kernel 1',ha='center',fontsize=30)
fig.text(.5,.62,'Erosion con Kernel 2',ha='center',fontsize=30)
fig.text(.5,.32,'Erosion con Kernel 3',ha='center',fontsize=30)

plt.tight_layout()
plt.show()
```



Podemos observar que cambiando el tamaño del kernel el circulo modificó su forma, sin embargo, siempre redujo su tamaño dado que siempre se erosiono la imagen, dado lo anterior dependiendo del tamaño y forma del kernel podemos modificar nuestras imágenes.

###

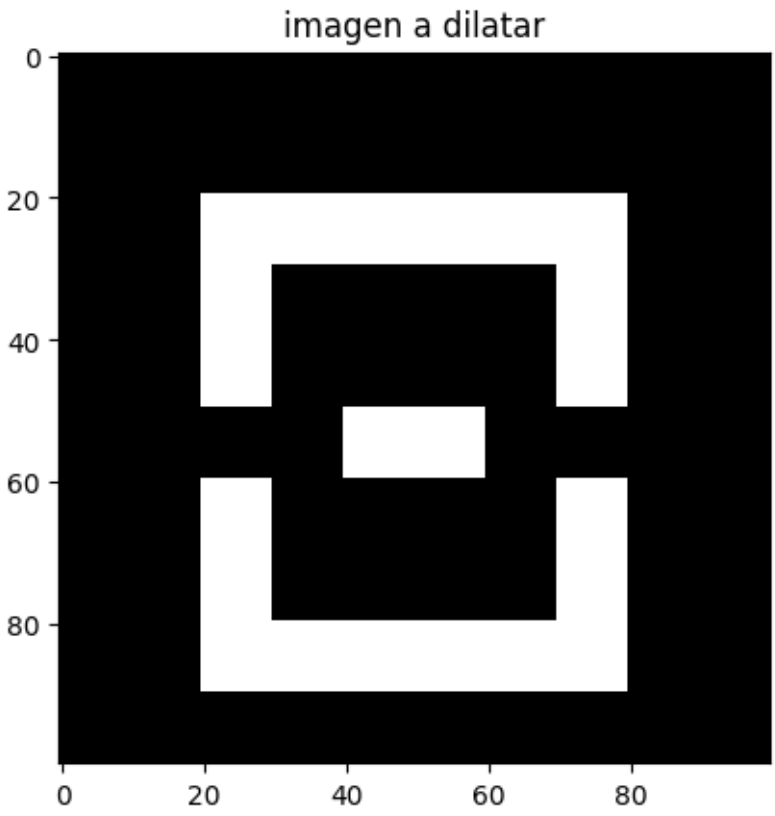
Dilatación

La dilatación es la operación morfológica que aumenta el tamaño de las regiones del primer plano (blanco) en imágenes binarias. De igual manera que en la erosión el kernel recorre la imagen y mediante este los pixeles de primer plano se pueden expandir, en términos sencillos la dilatación se encarga de agregar pixeles en los bordes de los objetos.

Esta operación ayuda a rellenar pequeños huecos en objetos, ayuda a conectar componentes que tienen una ligera separación entre sí y aumenta el grosor de los objetos.

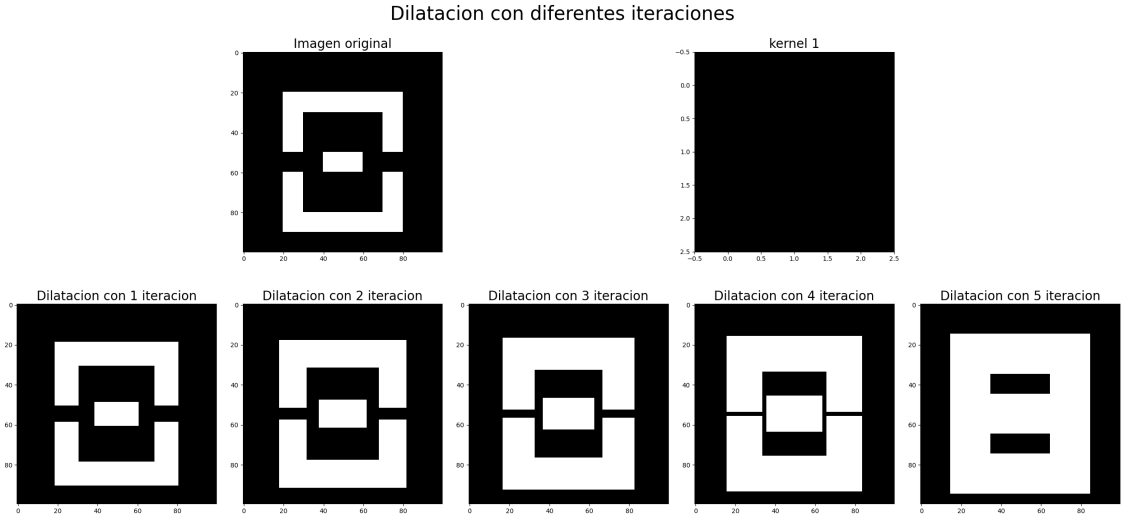
Para ilustrar esta operación crearemos una imagen en la que dibujaremos un "8" sin embargo este estará cortado, y observaremos el funcionamiento de esta operación.

[ ]:
```
#creamos una imagen
ocho = np.zeros((100,100),dtype='uint8')

#dibujamos un 8 sobre la imagen

#parte de arriba
ocho[20:30,20:80]=1
ocho[20:50,20:30]=1
ocho[20:50,70:80]=1

#parte de abajo
ocho[80:90,20:80]=1
ocho[60:90,20:30]=1
ocho[60:90,70:80]=1

#parte central
ocho[50:60,40:60]=1

#mostramos la imagen
plt.imshow(ocho,cmap='gray')
plt.title('imagen a dilatar')
plt.show()
```



[ ]:
```
#dilatamos la imagen con el kernel 1
dilatacion1 = cv2.dilate(ocho,kernel1,iterations=1)
dilatacion2 = cv2.dilate(ocho,kernel1,iterations=2)
dilatacion3 = cv2.dilate(ocho,kernel1,iterations=3)
dilatacion4 = cv2.dilate(ocho,kernel1,iterations=4)
dilatacion5 = cv2.dilate(ocho,kernel1,iterations=5)

#mostramos la imagen dilatada
fi,axis = plt.subplots(2,5,figsize=(25,12))

#apagamos los ejes que no se ocuparan
axis[0,0].axis('off')
axis[0,2].axis('off')
axis[0,4].axis('off')

#imagen oriiginal
axis[0,1].imshow(ocho,cmap='gray')
axis[0,1].set_title('Imagen original',fontsize=20)

#kernel 1
axis[0,3].imshow(kernel1,cmap='gray')
```

```
axis[0,3].set_title('kernel 1',fontsize=20)

#iteracion 1
axis[1,0].imshow(dilatacion1,cmap='gray')
axis[1,0].set_title('Dilatacion con 1 iteracion',fontsize=20)

#iteracion 2
axis[1,1].imshow(dilatacion2,cmap='gray')
axis[1,1].set_title('Dilatacion con 2 iteracion',fontsize=20)

#iteracion 3
axis[1,2].imshow(dilatacion3,cmap='gray')
axis[1,2].set_title('Dilatacion con 3 iteracion',fontsize=20)

#iteracion 4
axis[1,3].imshow(dilatacion4,cmap='gray')
axis[1,3].set_title('Dilatacion con 4 iteracion',fontsize=20)

#iteracion 5
axis[1,4].imshow(dilatacion5,cmap='gray')
axis[1,4].set_title('Dilatacion con 5 iteracion',fontsize=20)

#titulo y acomodo de las imagenes
plt.suptitle('Dilatacion con diferentes iteraciones',fontsize=30)
plt.tight_layout()
plt.show()
```

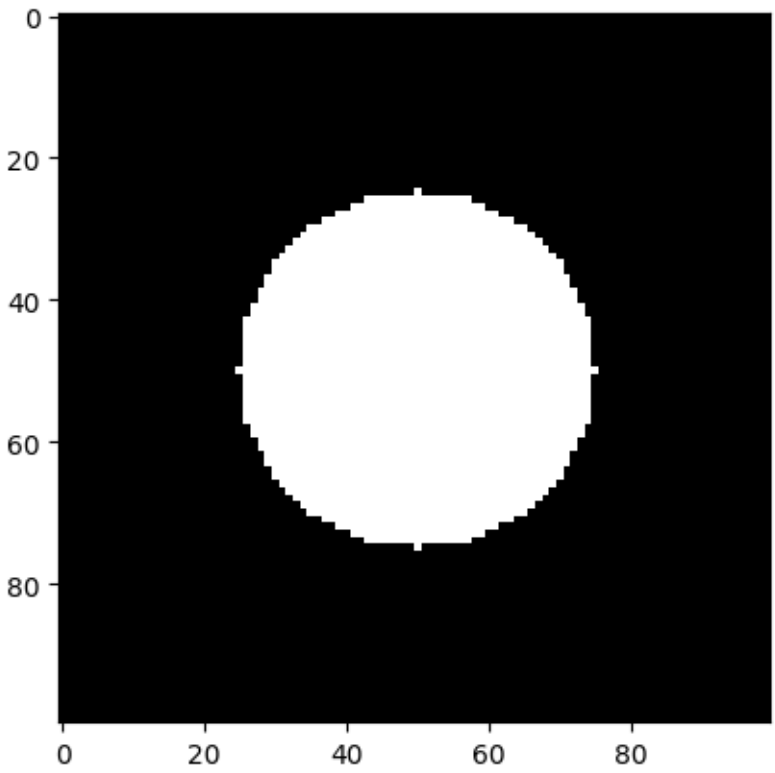


Podemos observar que hemos cerrado correctamente el número 8 en la quinta iteración, sin embargo, si modificamos el kernel podemos hacer modificaciones en la imagen.

A continuación, usaremos img2 creada en el tema erosión para observar como esta se deforma cuando cambiamos el kernel.

[ ]:
```
#retomamos img2 creada en el tema de erosion
plt.imshow(img2,cmap='gray')
```

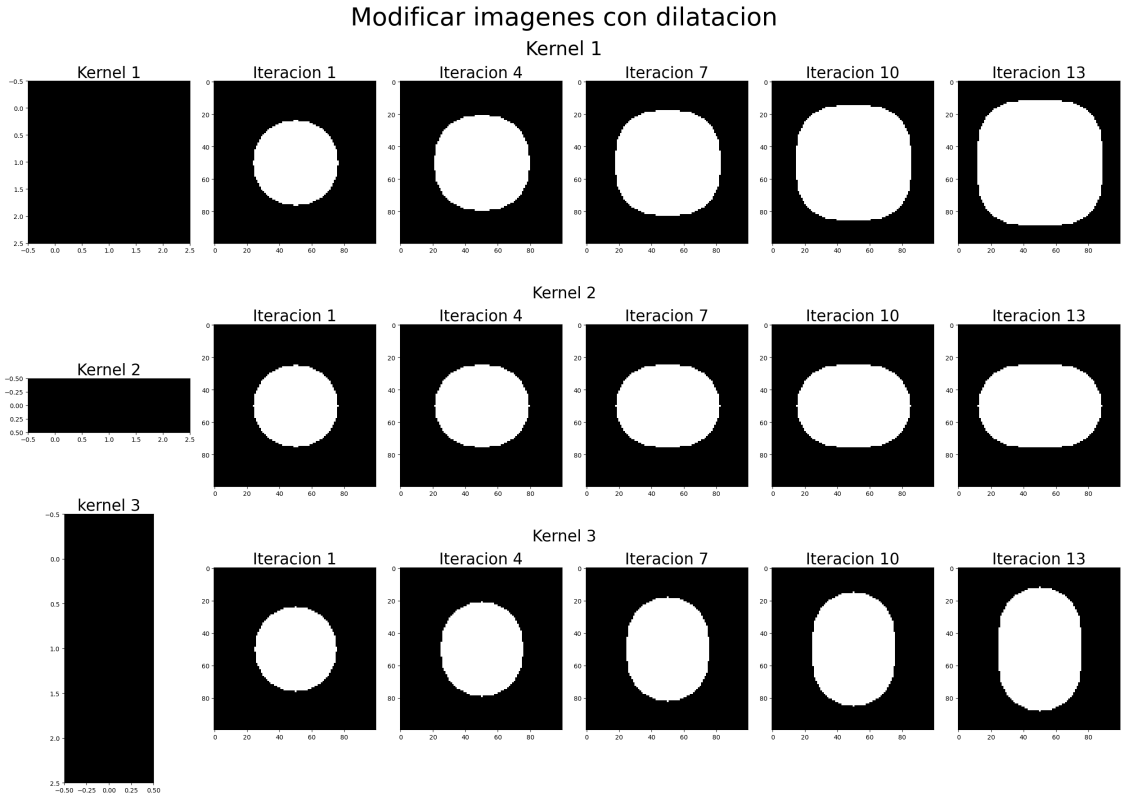
[ ]:
```
<matplotlib.image.AxesImage at 0x2ae1db9d2e0>
```



[ ]:
```
#dilatamos las imagens con los diferentes kernel e iteraciones
#kernel 1
dilatacion_k1_1 = cv2.dilate(img2,kernel1,iterations=1)
dilatacion_k1_2 = cv2.dilate(img2,kernel1,iterations=4)
dilatacion_k1_3 = cv2.dilate(img2,kernel1,iterations=7)
dilatacion_k1_4 = cv2.dilate(img2,kernel1,iterations=10)
dilatacion_k1_5 = cv2.dilate(img2,kernel1,iterations=13)

#kernel 2
dilatacion_k2_1 = cv2.dilate(img2,kernel2,iterations=1)
dilatacion_k2_2 = cv2.dilate(img2,kernel2,iterations=4)
dilatacion_k2_3 = cv2.dilate(img2,kernel2,iterations=7)
dilatacion_k2_4 = cv2.dilate(img2,kernel2,iterations=10)
dilatacion_k2_5 = cv2.dilate(img2,kernel2,iterations=13)
```

```
#kernel 3
dilatacion_k3_1 = cv2.dilate(img2,kernel3,iterations=1)
dilatacion_k3_2 = cv2.dilate(img2,kernel3,iterations=4)
dilatacion_k3_3 = cv2.dilate(img2,kernel3,iterations=7)
dilatacion_k3_4 = cv2.dilate(img2,kernel3,iterations=10)
dilatacion_k3_5 = cv2.dilate(img2,kernel3,iterations=13)

#mostramos los resultados
fig,axis = plt.subplots(3,6,figsize=(25,18))

#kernel 1
axis[0,0].imshow(kernel1,cmap='gray')
axis[0,0].set_title('Kernel 1',fontsize = 25)
axis[0,1].imshow(dilatacion_k1_1,cmap='gray')
axis[0,1].set_title('Iteracion 1',fontsize = 25)
axis[0,2].imshow(dilatacion_k1_2,cmap='gray')
axis[0,2].set_title('Iteracion 4',fontsize = 25)
axis[0,3].imshow(dilatacion_k1_3,cmap='gray')
axis[0,3].set_title('Iteracion 7',fontsize = 25)
axis[0,4].imshow(dilatacion_k1_4,cmap='gray')
axis[0,4].set_title('Iteracion 10',fontsize = 25)
axis[0,5].imshow(dilatacion_k1_5,cmap='gray')
axis[0,5].set_title('Iteracion 13',fontsize = 25)

#kernel 2
axis[1,0].imshow(kernel2,cmap='gray')
axis[1,0].set_title('Kernel 2',fontsize = 25)
axis[1,1].imshow(dilatacion_k2_1,cmap='gray')
axis[1,1].set_title('Iteracion 1',fontsize = 25)
axis[1,2].imshow(dilatacion_k2_2,cmap='gray')
axis[1,2].set_title('Iteracion 4',fontsize = 25)
axis[1,3].imshow(dilatacion_k2_3,cmap='gray')
axis[1,3].set_title('Iteracion 7',fontsize = 25)
axis[1,4].imshow(dilatacion_k2_4,cmap='gray')
axis[1,4].set_title('Iteracion 10',fontsize = 25)
axis[1,5].imshow(dilatacion_k2_5,cmap='gray')
axis[1,5].set_title('Iteracion 13',fontsize = 25)

#kernel 3
axis[2,0].imshow(kernel3,cmap='gray')
axis[2,0].set_title('kernel 3',fontsize = 25)
axis[2,1].imshow(dilatacion_k3_1,cmap='gray')
axis[2,1].set_title('Iteracion 1',fontsize = 25)
axis[2,2].imshow(dilatacion_k3_2,cmap='gray')
axis[2,2].set_title('Iteracion 4',fontsize = 25)
axis[2,3].imshow(dilatacion_k3_3,cmap='gray')
```

```
axis[2,3].set_title('Iteracion 7',fontsize = 25)
axis[2,4].imshow(dilatacion_k3_4,cmap='gray')
axis[2,4].set_title('Iteracion 10',fontsize = 25)
axis[2,5].imshow(dilatacion_k3_5,cmap='gray')
axis[2,5].set_title('Iteracion 13',fontsize = 25)

#agregamos subtitulos
fig.suptitle('Modificar imagenes con dilatacion',fontsize=40)
fig.text(.5,.92,'Kernel 1',ha='center',fontsize=30)
fig.text(.5,.62,'Kernel 2',ha='center',fontsize=25)
fig.text(0.5,.32,'Kernel 3',ha='center',fontsize=25)

plt.tight_layout()
plt.show()
```



Como podemos observar dependiendo el kernel nuestra imagen cambia su forma mientras aumenta su tamaño, lo cual se hace mas notable mientras mas iteraciones hacemos.

###

Apertura

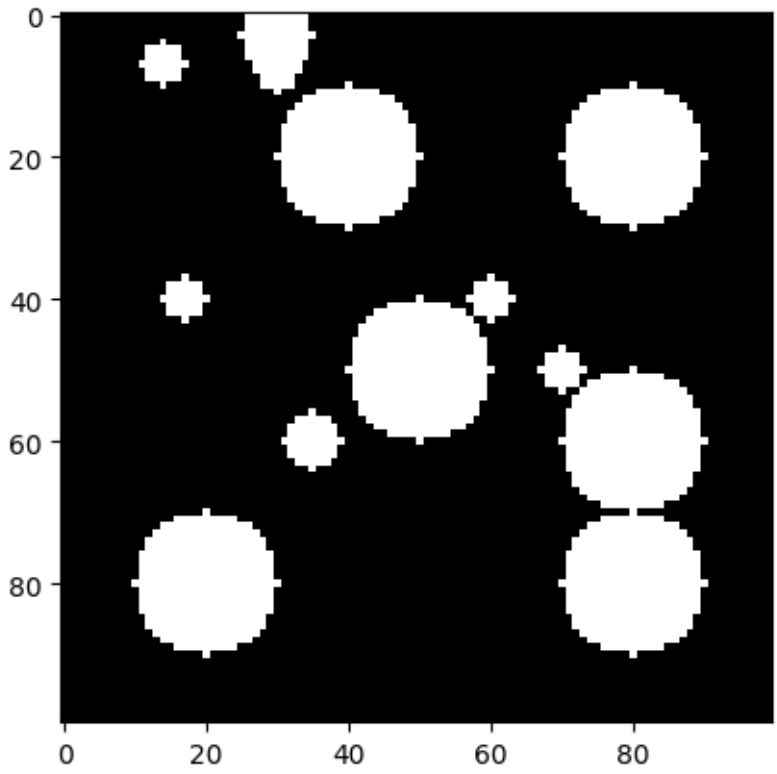
La apertura es la operación morfológica que combina las 2 operaciones vistas anteriormente (erosión y dilatación) las realiza en el mismo orden, es decir primero erosiona la imagen y después la dilata,

al igual que en estas operaciones se usa un kernel para trabajar. Esta operación es usada para eliminar pequeños objetos o el ruido de una imagen , esto mientras se mantiene la forma de los objetos más grandes.

Para observar el funcionamiento de esta operación crearemos una imagen, dentro de la cual dibujaremos varios círculos de diferentes tamaños y observaremos el funcionamiento de la operación, para esto usremos el kernel1 que ya hemos trabajado en las operaciones anteriores.

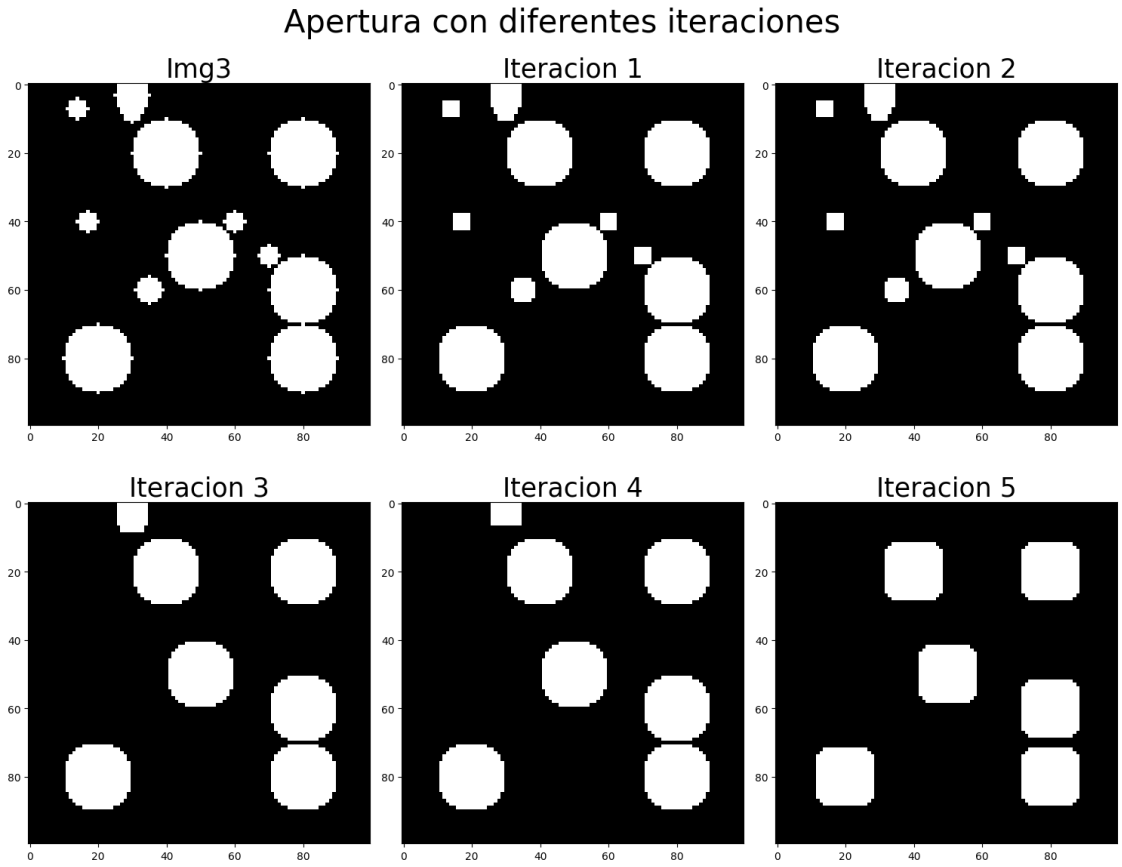
[ ]:
```
#creamos nuestra imagen
img3 = np.zeros((100,100),dtype='uint8')

#dibujamos sobre nuestra imagen
cv2.circle(img3,(30,3),5,1,thickness=-1)
cv2.circle(img3,(35,60),4,1,thickness=-1)
cv2.circle(img3,(17,40),3,1,thickness=-1)
cv2.circle(img3,(30,8),3,1,thickness=-1)
cv2.circle(img3,(14,7),3,1,thickness=-1)
cv2.circle(img3,(60,40),3,1,thickness=-1)
cv2.circle(img3,(70,50),3,1,thickness=-1)
cv2.circle(img3,(50,50),10,1,thickness=-1)
cv2.circle(img3,(40,20),10,1,thickness=-1)
cv2.circle(img3,(80,80),10,1,thickness=-1)
cv2.circle(img3,(80,60),10,1,thickness=-1)
cv2.circle(img3,(80,20),10,1,thickness=-1)
cv2.circle(img3,(20,80),10,1,thickness=-1)

#mostramos la imagen
plt.imshow(img3,cmap='gray')
```

[ ]:
```
<matplotlib.image.AxesImage at 0x2ae701b0a40>
```



[ ]:
```
#aperturamos la imagen con diferentes iteraciones y usamos el kernel 1
apertura_1 = cv2.morphologyEx(img3,cv2.MORPH_OPEN,kernel1,iterations=1)
apertura_2 = cv2.morphologyEx(img3,cv2.MORPH_OPEN,kernel1,iterations=2)
apertura_3 = cv2.morphologyEx(img3,cv2.MORPH_OPEN,kernel1,iterations=3)
apertura_4 = cv2.morphologyEx(img3,cv2.MORPH_OPEN,kernel1,iterations=4)
apertura_5 = cv2.morphologyEx(img3,cv2.MORPH_OPEN,kernel1,iterations=5)

#mostramos los resultados
fig,axis = plt.subplots(2,3,figsize=(15,12))

axis[0,0].imshow(img3,cmap='gray')
axis[0,0].set_title('Img3',fontsize=25)
axis[0,1].imshow(apertura_1,cmap='gray')
axis[0,1].set_title('Iteracion 1',fontsize=25)
axis[0,2].imshow(apertura_2,cmap='gray')
axis[0,2].set_title('Iteracion 2',fontsize=25)
axis[1,0].imshow(apertura_3,cmap='gray')
axis[1,0].set_title('Iteracion 3',fontsize=25)
axis[1,1].imshow(apertura_4,cmap='gray')
axis[1,1].set_title('Iteracion 4',fontsize=25)
axis[1,2].imshow(apertura_5,cmap='gray')
axis[1,2].set_title('Iteracion 5',fontsize=25)
```

```
fig.suptitle('Apertura con diferentes iteraciones',fontsize=30)

plt.tight_layout()
plt.show()
```



Podemos observar como los círculos pequeños desparecen conforme mas iteraciones, de igual manera podemos notar como los círculos se definen de manera exagerada hasta el punto de llegar a una forma mas cuadrada.

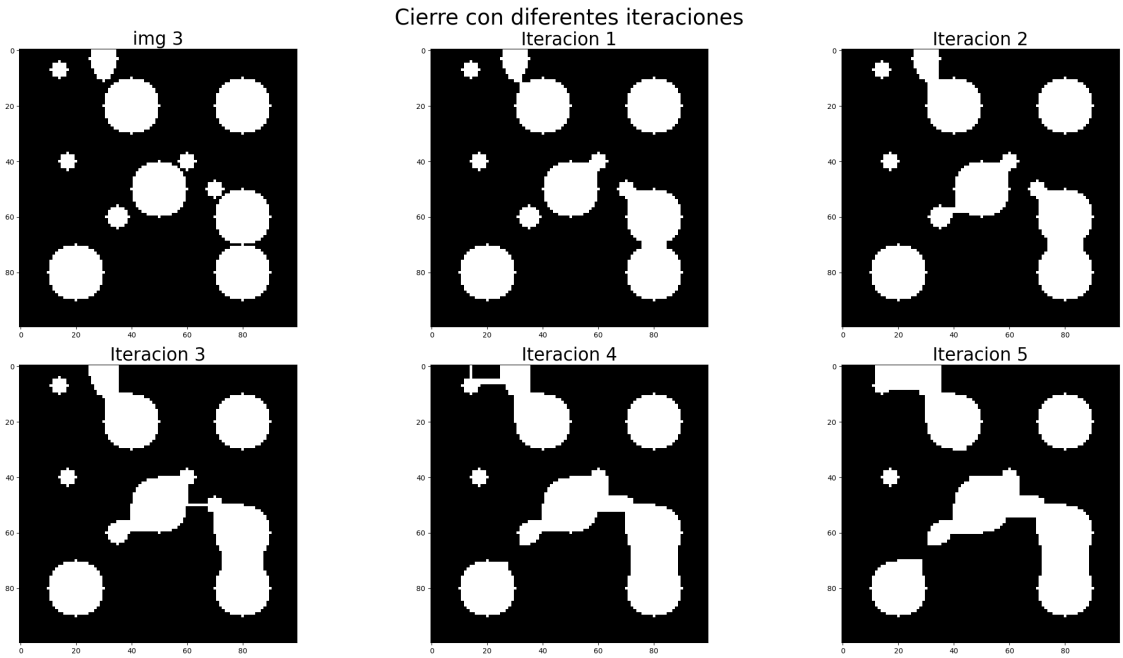
###

Cierre

Esta operación al igual que la apertura combina las operaciones de erosión y dilatación, sin embargo, en este caso se realizan de manera inversa, primero dilatamos la imagen con la intención de cerrar huecos en la imagen y tratar de cerrar algunos espacios entre objetos, después erosionamos la imagen para eliminar el ruido de la imagen y definir de manera correcta los bordes de los elementos.

En este caso para ver el funcionamiento de esta operación reusaremos img3 creada en el tema de "apertura" sobre esta imagen usaremos la operación de cierre con el kernel1 que creamos en el inicio de este módulo.

[ ]:
```
#usamos la operacion de cierre con el kernel 1 en la img3 con diferentes␣
 ↪iteraciones
cierre_1 = cv2.morphologyEx(img3,cv2.MORPH_CLOSE,kernel1,iterations=1)
cierre_2 = cv2.morphologyEx(img3,cv2.MORPH_CLOSE,kernel1,iterations=2)
cierre_3 = cv2.morphologyEx(img3,cv2.MORPH_CLOSE,kernel1,iterations=3)
cierre_4 = cv2.morphologyEx(img3,cv2.MORPH_CLOSE,kernel1,iterations=4)
cierre_5 = cv2.morphologyEx(img3,cv2.MORPH_CLOSE,kernel1,iterations=5)

#mostramos los resultados
fig,axis = plt.subplots(2,3,figsize=(25,13))

axis[0,0].imshow(img3,cmap='gray')
axis[0,0].set_title('img 3',fontsize=25)
axis[0,1].imshow(cierre_1,cmap='gray')
axis[0,1].set_title('Iteracion 1',fontsize=25)
axis[0,2].imshow(cierre_2,cmap='gray')
axis[0,2].set_title('Iteracion 2',fontsize=25)
axis[1,0].imshow(cierre_3,cmap='gray')
axis[1,0].set_title('Iteracion 3',fontsize=25)
axis[1,1].imshow(cierre_4,cmap='gray')
axis[1,1].set_title('Iteracion 4',fontsize=25)
axis[1,2].imshow(cierre_5,cmap='gray')
axis[1,2].set_title('Iteracion 5',fontsize=25)

plt.suptitle('Cierre con diferentes iteraciones',fontsize=30)
plt.tight_layout()
plt.show()
```



Se observa como con esta operación los elementos se van juntando cada vez mas entre ellos, hasta el punto de volverse uno solo.

###

Gradiente morfológico

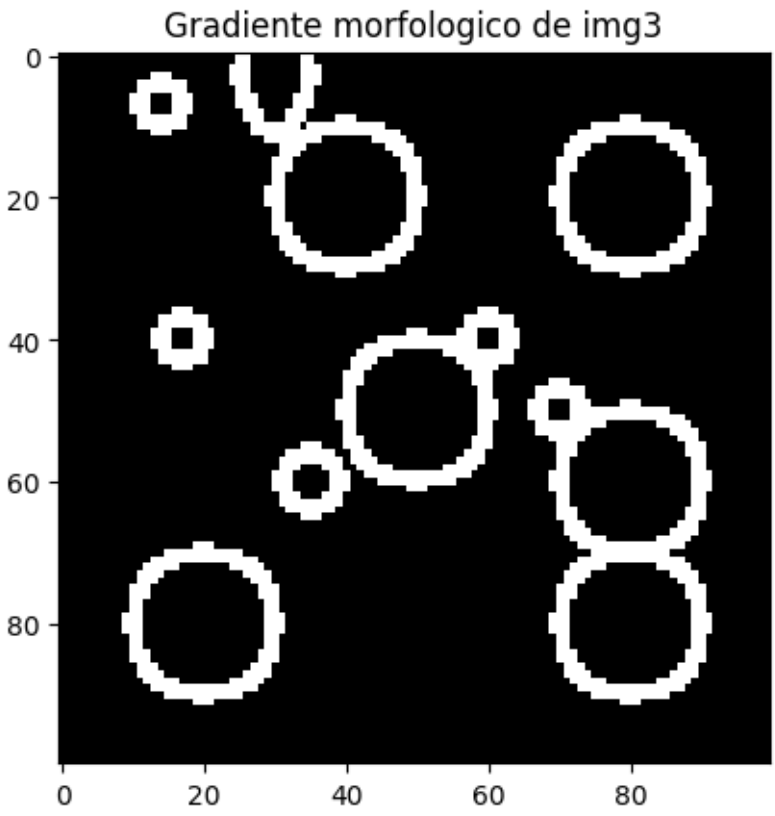
El gradiente morfológico es una operación que resalta los bordes de los objetos en una imagen, podemos definir a esta operación como el resultado de la siguiente resta:

$$gradienteMorfologico = imagenDilatada - imagenErosionada$$

Esto se debe a que la imagen dilatada proporciona un aumento en los bordes de los objetos, mientras que la erosión elimina parte de estos, es decir que solo aumentamos o disminuimos el tamaño de los bordes, por lo que al restar estas imágenes obtenemos un borde definido.

para ilustrar esta operación, usaremos img3 y el kernel1 de manera que podamos visualizar el funcionamiento del gradiente morfológico.

[ ]:
```
#calculamos el gradiente de nuestra imagen
gradiente = cv2.morphologyEx(img3,cv2.MORPH_GRADIENT,kernel1,iterations=1)

#mostramos nuestra imagen
plt.imshow(gradiente,cmap='gray')
plt.title('Gradiente morfologico de img3')
plt.show()
```



Claramente observamos como obtuvimos únicamente los bordes de los elementos, por lo cual nuestra función se ejecutó correctamente.

###

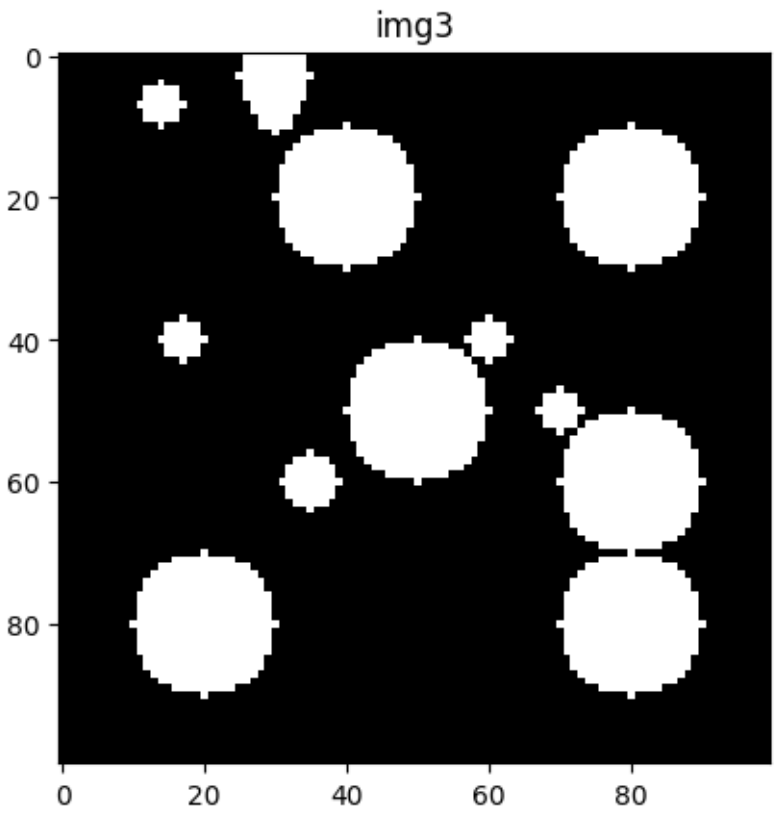
TOP HAT

Esta operación se encarga de resaltar las pequeñas estructuras brillantes en la imagen,en esencia hablamos de realizar la siguiente operacion:

$$TopHAT = imgOriginal - imgApertura$$

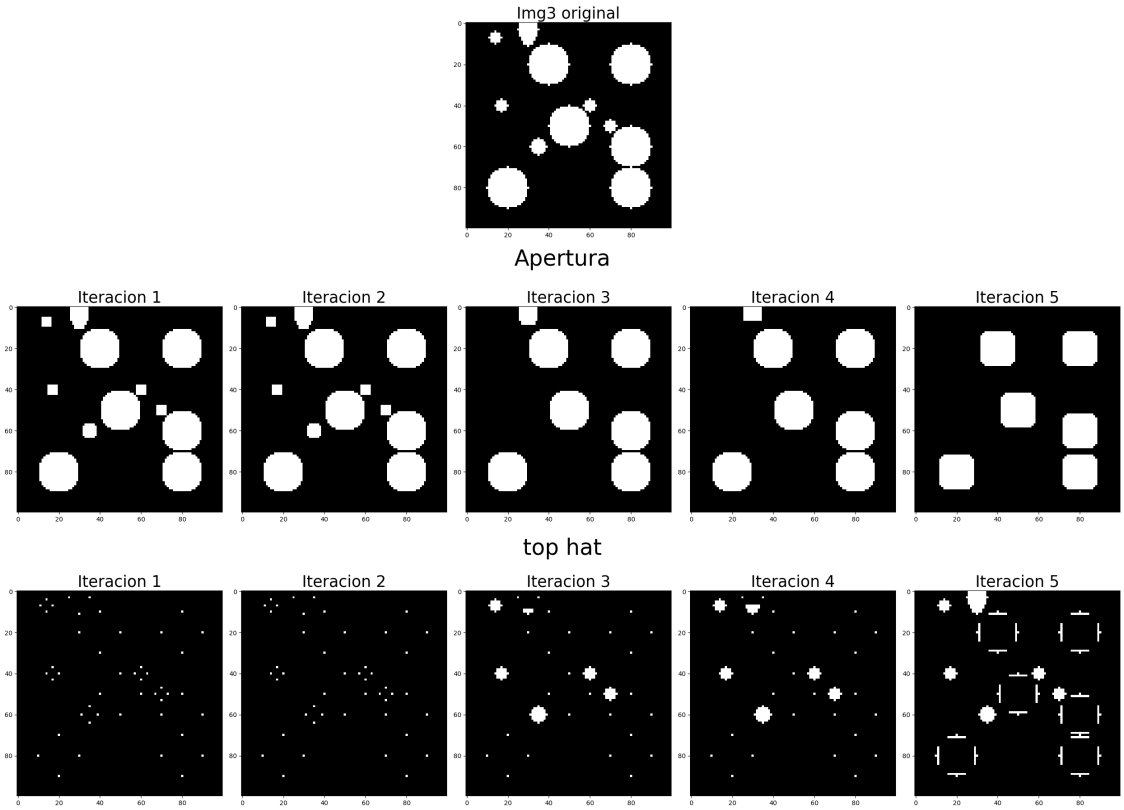
Es decir que a la imagen original le restemos la imagen con la operacion de apertura, podemos observar este resultado de manera visual con img3 y kernel1 como se muestra a continuacion.

[ ]:
```
#reutilizamo la imagen 3
plt.imshow(img3,cmap='gray')
plt.title('img3')
plt.show()
```



[ ]:
```
#ya que es una diferencia entre la imagen original y la imagen de apertura␣
 ↪mostraremos los cambios

#aplicamos la operacion de top hat
top_hat_1 = cv2.morphologyEx(img3,cv2.MORPH_TOPHAT,kernel1,iterations=1)
top_hat_2 = cv2.morphologyEx(img3,cv2.MORPH_TOPHAT,kernel1,iterations=2)
top_hat_3 = cv2.morphologyEx(img3,cv2.MORPH_TOPHAT,kernel1,iterations=3)
top_hat_4 = cv2.morphologyEx(img3,cv2.MORPH_TOPHAT,kernel1,iterations=4)
top_hat_5 = cv2.morphologyEx(img3,cv2.MORPH_TOPHAT,kernel1,iterations=5)

#mostramos la compararcion
fig,axis = plt.subplots(3,5,figsize=(25,20))

#mostramos la imagen original
axis[0,2].imshow(img3,cmap='gray')
axis[0,2].set_title('Img3 original',fontsize=25)

#apagamos los ees que no ocupemos
axis[0,0].axis('off')
axis[0,1].axis('off')
```

```
axis[0,3].axis('off')
axis[0,4].axis('off')

#apertura
axis[1,0].imshow(apertura_1,cmap='gray')
axis[1,0].set_title('Iteracion 1',fontsize=25)
axis[1,1].imshow(apertura_2,cmap='gray')
axis[1,1].set_title('Iteracion 2',fontsize=25)
axis[1,2].imshow(apertura_3,cmap='gray')
axis[1,2].set_title('Iteracion 3',fontsize=25)
axis[1,3].imshow(apertura_4,cmap='gray')
axis[1,3].set_title('Iteracion 4',fontsize=25)
axis[1,4].imshow(apertura_5,cmap='gray')
axis[1,4].set_title('Iteracion 5',fontsize=25)

#top hat
axis[2,0].imshow(top_hat_1,cmap='gray')
axis[2,0].set_title('Iteracion 1',fontsize=25)
axis[2,1].imshow(top_hat_2,cmap='gray')
axis[2,1].set_title('Iteracion 2',fontsize=25)
axis[2,2].imshow(top_hat_3,cmap='gray')
axis[2,2].set_title('Iteracion 3',fontsize=25)
axis[2,3].imshow(top_hat_4,cmap='gray')
axis[2,3].set_title('Iteracion 4',fontsize=25)
axis[2,4].imshow(top_hat_5,cmap='gray')
axis[2,4].set_title('Iteracion 5',fontsize=25)

plt.suptitle('Diferencia entre apertura y top hat',fontsize=45)
fig.text(0.5,0.64,'Apertura',ha='center',fontsize=35)
fig.text(0.5,0.32,'top hat',ha='center',fontsize=35)
plt.tight_layout()
plt.show()
```



Podemos observar como los elementos que se eliminaron en la operacion de apertura son los elementos que se muestran en top hat.

###

Black Hat

Esta operación resalta las pequeñas estructuras obscuras en la imagen, al igual que en las operaciones anteriores, esta es el resultado de una diferencia entre dos imágenes, podemos visualizar a esta como el resultado de la siguiente operación:

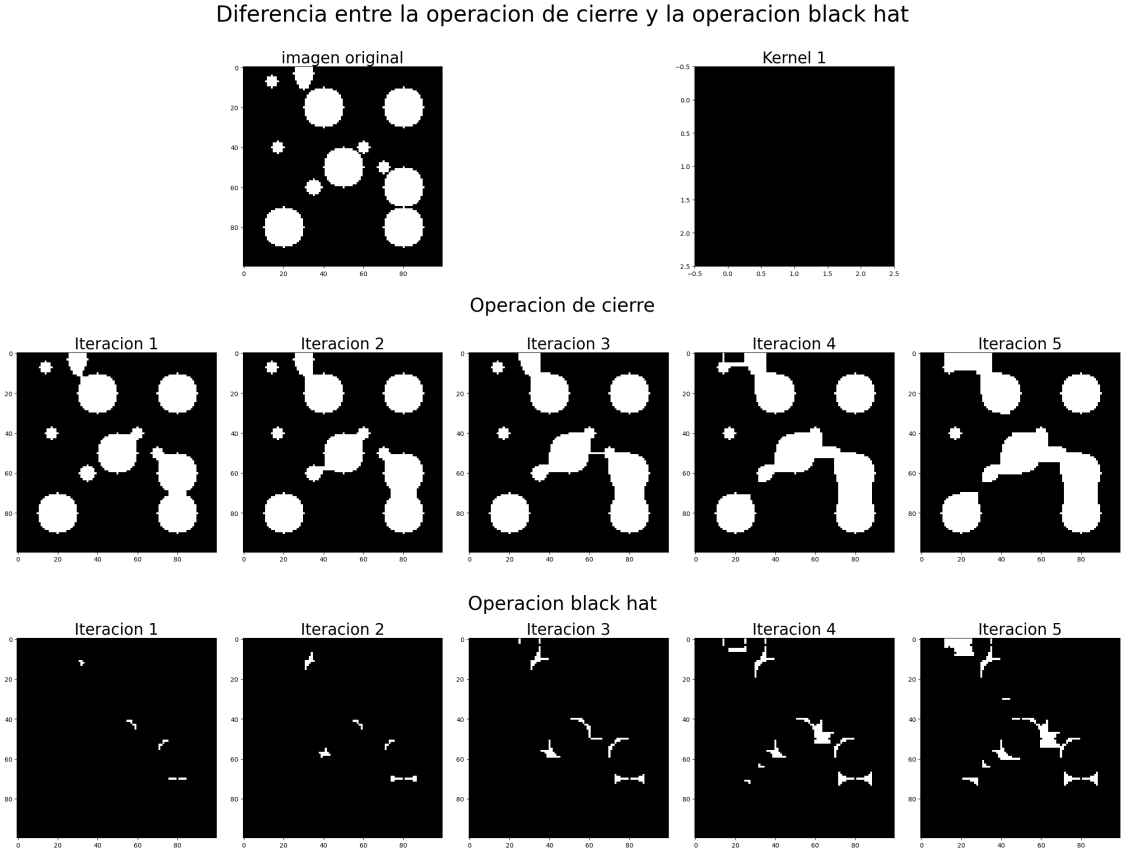
$$BlackHat = imgOriginal - imgCierre$$

Es decir que para obtener esta imagen le restamos a la imagen original la misma imagen, pero con operación de cierre. Ilustraremos esta operación con img3 y kernel1.

[ ]:
```
#aplicamos la opercion Black hat en la img3 y usamos el kernel 1 y diferentes␣
↪iteraciones
blackhat_1 = cv2.morphologyEx(img3,cv2.MORPH_BLACKHAT,kernel1,iterations=1)
blackhat_2 = cv2.morphologyEx(img3,cv2.MORPH_BLACKHAT,kernel1,iterations=2)
blackhat_3 = cv2.morphologyEx(img3,cv2.MORPH_BLACKHAT,kernel1,iterations=3)
blackhat_4 = cv2.morphologyEx(img3,cv2.MORPH_BLACKHAT,kernel1,iterations=4)
```

```
blackhat_5 = cv2.morphologyEx(img3,cv2.MORPH_BLACKHAT,kernel1,iterations=5)

#mostramos la comparacion entre la operacion de cierre y la de black hat
#creamos nuestras graficas y definimos sus tamaños
fig, axis = plt.subplots(3,5,figsize=(25,20))

#desactivamos los ejes que no vamos a ocupar
axis[0,0].axis('off')
axis[0,2].axis('off')
axis[0,4].axis('off')

#mostramos la imagen original
axis[0,1].imshow(img3,cmap='gray')
axis[0,1].set_title('imagen original',fontsize=25)

#mostramos el kernel con el que vamos a trabajar
axis[0,3].imshow(kernel1,cmap='gray')
axis[0,3].set_title('Kernel 1',fontsize=25)

#mostramos la operacion de cierre
axis[1,0].imshow(cierre_1,cmap='gray')
axis[1,0].set_title('Iteracion 1',fontsize=25)
axis[1,1].imshow(cierre_2,cmap='gray')
axis[1,1].set_title('Iteracion 2',fontsize=25)
axis[1,2].imshow(cierre_3,cmap='gray')
axis[1,2].set_title('Iteracion 3',fontsize=25)
axis[1,3].imshow(cierre_4,cmap='gray')
axis[1,3].set_title('Iteracion 4',fontsize=25)
axis[1,4].imshow(cierre_5,cmap='gray')
axis[1,4].set_title('Iteracion 5',fontsize=25)

#mostramos la operacion black hat
axis[2,0].imshow(blackhat_1,cmap='gray')
axis[2,1].imshow(blackhat_2,cmap='gray')
axis[2,2].imshow(blackhat_3,cmap='gray')
axis[2,3].imshow(blackhat_4,cmap='gray')
axis[2,4].imshow(blackhat_5,cmap='gray')
#colocamos los titulos a cada imagen
axis[2,0].set_title('Iteracion 1',fontsize=25)
axis[2,1].set_title('Iteracion 2',fontsize=25)
axis[2,2].set_title('Iteracion 3',fontsize=25)
axis[2,3].set_title('Iteracion 4',fontsize=25)
axis[2,4].set_title('Iteracion 5',fontsize=25)

#colocamos titulos y subtitulos
plt.suptitle('Diferencia entre la operacion de cierre y la operacion black␣
 ↪hat', fontsize=35)
```

```
fig.text(0.5,.64,'Operacion de cierre',ha='center',fontsize=30)
fig.text(0.5,.31,'Operacion black hat',ha='center',fontsize=30)

plt.tight_layout()
plt.show()
```



###

Referencias