Procesamiento de Imágenes Digitales - Entrega 4 **Gálvez Camacho,Dionisio**

1)

La erosión de una imagen binaria dado “f” dado por un elemento estructural produce una nueva imagen binaria donde (x,y) = 1 si el origen del elemento estructural cubre la imagen de entrada , “0” en caso contrario. Por tanto esto va a tener un efecto de que todos los píxeles cerca del límite serán descartados dependiendo del tamaño del núcleo. Por lo tanto, el grosor o el tamaño del objeto en primer plano disminuye o simplemente la región blanca disminuye en la imagen. Es útil para eliminar pequeños ruidos blancos. Esto no nos sirve para lo que pide el ejercicio , ya que elimina un poco el ruido pero deforma las figuras.

La dilatación es el contrario de la erosión. Aquí,un píxel tiene el valor de ‘1’ si al menos uno de ellos debajo del kernel es ‘1’. Por tanto incrementa la región blanca en la imagen o aumenta el tamaño del objeto en primer plano.
Porque, la erosión elimina los ruidos blancos, pero también reduce nuestro objeto. Entonces si dilatamos. Como el ruido se ha ido, no volverá, pero nuestro objeto aumenta de tamaño. También es útil para unir partes rotas de un objeto. Es útil para nuestro ejercicio pero no nos soluciona el problema , aunque queda un poco de ruido en la imagen.

La apertura es la erosión seguida por la dilatación. La apertura se llama así porque abre un hueco entre dos objetos conectados por un conjunto de píxeles.La dilatación restaura cualquier región que haya sobrevivido a la erosión. Sirve para eliminar el ruido.

La clausura es el contrario de la apertura , es la dilatación seguida de la erosión. Esto tendrá el efecto de que los objetos conectados aumentarán de tamaño y luego al realizar la erosión suavizará los bordes y elimina huecos estrechos , disminuye un poco de tamaño los objetos.

Por tanto podemos realizar primero clausura para aumentar de tamaño los objetos conectados y luego suavizar estos mismos objetos y eliminar los huecos que queden entre ellos. Seguidamente podemos aplicar la apertura a la imagen que le hemos aplicado la clausura , para volver a aplicar erosión la cual apenas variará ya que la hemos aplicado antes en la clausura , y seguidamente la dilatación para restaurar los objetos que hayan sobrevivido a la erosión para así poder eliminar el ruido de la imagen. Así podremos aplicar un elemento estructural del 3x3 o 5x5 por ejemplo y así eliminar el ruido. El cual pienso que quedará mejor si aplicamos máscara 3x3 ya que son ruidos de tamaño pequeño y nos costará menos eliminarlo con una matriz de menos tamaño,y no deformará tanto los círculos pequeños.

Imagen original y aplicando clausura+apertura 3x3 kernel:

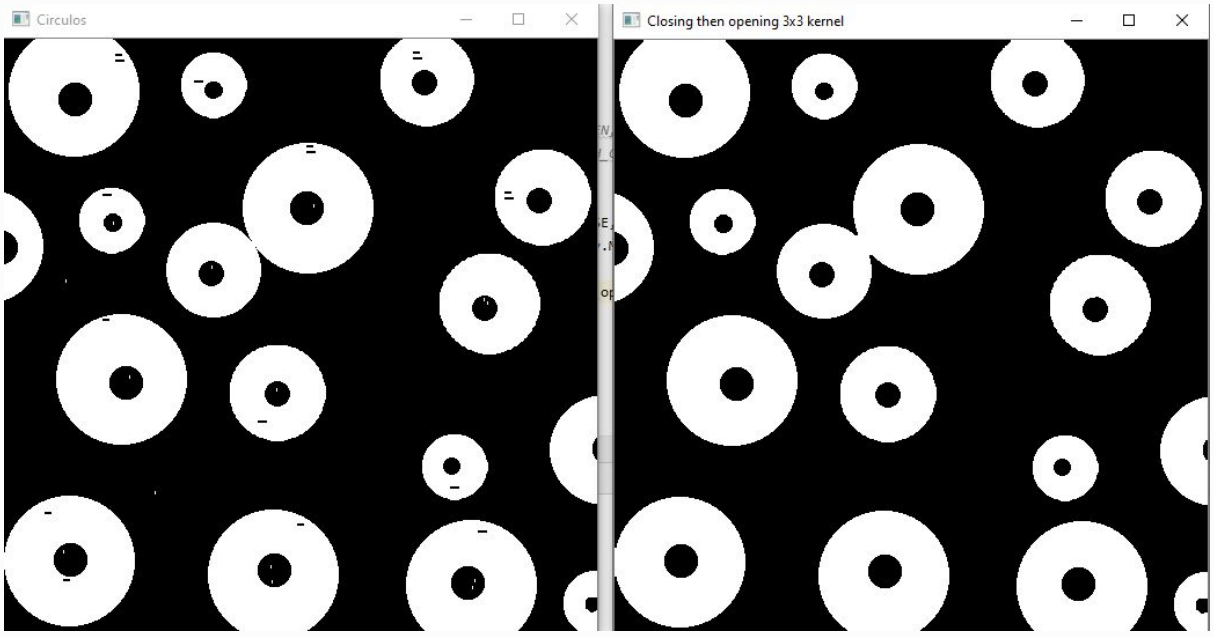
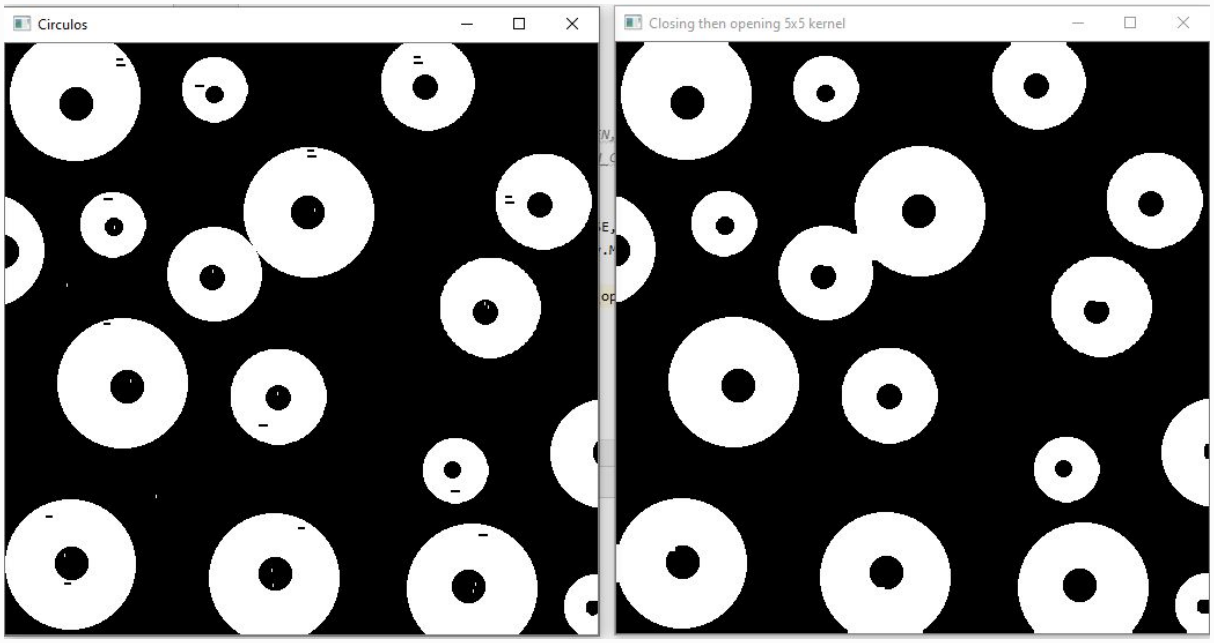


Imagen original y aplicando clausura+apertura 5x5 kernel:



Código aplicado:

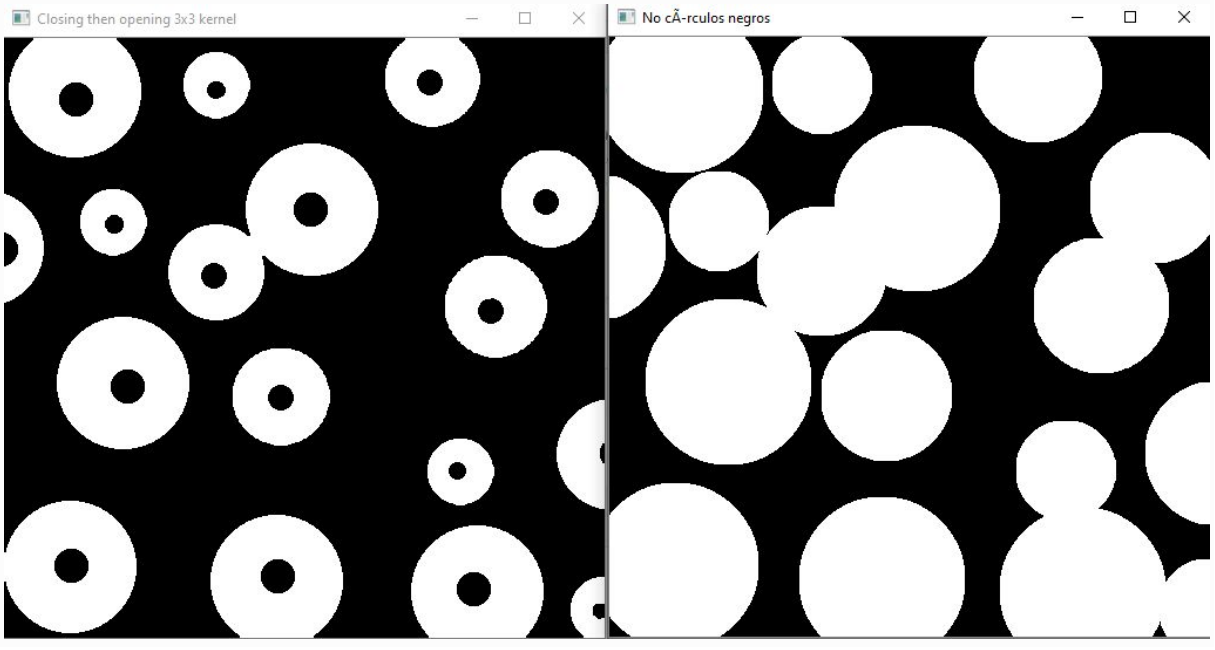
```
import cv2 as cv
import numpy as np
import math

#### 1 ####
img = cv.imread("circulos_ruidos.png", 0)
kernel = np.ones((3,3),np.uint8)
closing = cv.morphologyEx(img, cv.MORPH_CLOSE, kernel)
openingClosing = cv.morphologyEx(closing, cv.MORPH_OPEN, kernel)
cv.imshow("Closing then opening 3x3 kernel",openingClosing)
```

2)

Para quitar los pequeños puntos negros dentro de los círculos blancos , yo aplicaría una máscara de forma que los unos vayan quedando de forma circular dentro de la misma. Esta máscara se la aplicaría a la imagen de antes a la cual aplicamos clausura y luego apertura con un filtro de dilatación para eliminar estos círculos negros pequeños. Aquí,un píxel tiene el valor de ‘1’ si al menos uno de ellos debajo del kernel es ‘1’. Por tanto incrementa la región blanca en la imagen y desaparecen los círculos negros , por ello aumenta también de tamaño los círculos blancos grandes.

Ejemplo con máscara circular de tamaño 15 con una iteración:



Código aplicado:

```
#### 2 ####
def maskCircle(r):
    mask = np.ones((2*r,2*r),np.uint8)
    for x in range(0,r):
        for y in range(0,r):
            if not(math.sqrt((x - r) ** 2 + (y - r) ** 2) < r):
                mask[x][y] = 0
            if not(math.sqrt((x) ** 2 + (y - r) ** 2) < r):
                mask[x+r][y] = 0
            if not(math.sqrt((x) ** 2 + (y) ** 2) < r):
                mask[x+r][y+r] = 0
            if not(math.sqrt((x - r) ** 2 + y ** 2) < r):
                mask[x][y+r] = 0
    return mask

print(maskCircle(10))

dilation = cv.dilate(openingClosing,maskCircle(15),iterations = 1)
cv.imshow("No círculos negros",dilation)
cv.waitKey(0)
```