INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Reconocimiento de Formas y Visión por Computadora

Tarea 1

Lozano Rivera Oscar

Viernes, 18 marzo del 2022.

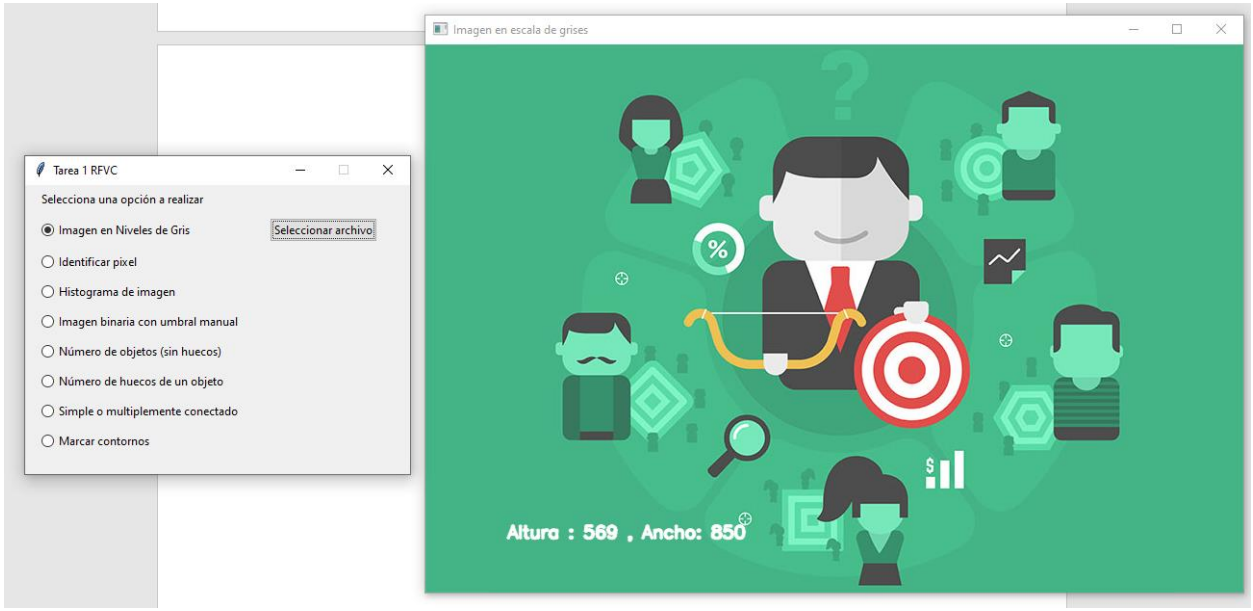
Elaborar un programa que permita leer un programa en Python que:

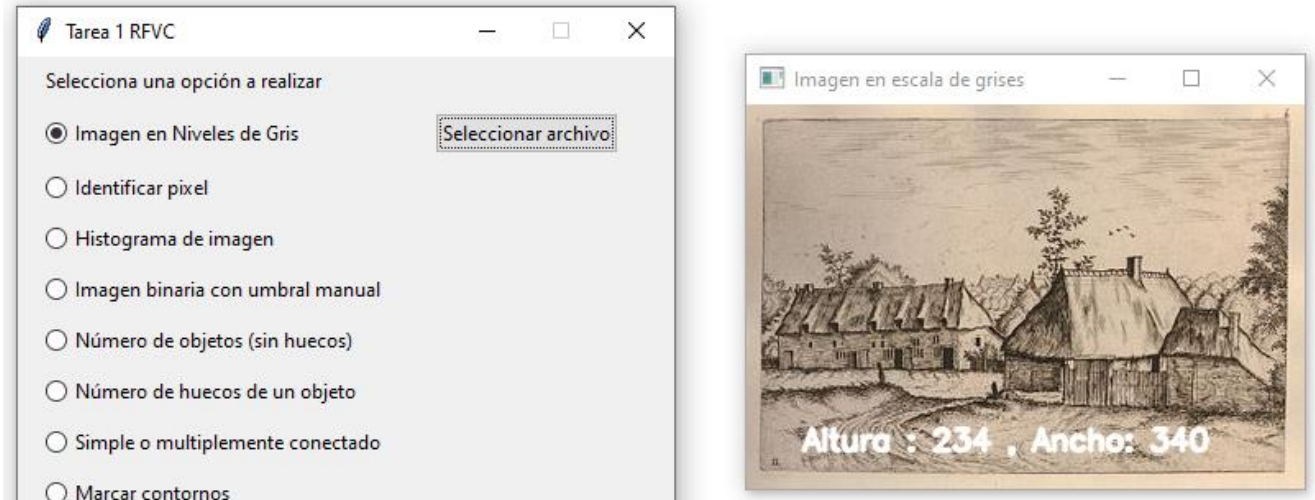
1. Permita leer una imagen en niveles de gris de memoria de la computadora y que permita mostrarla en pantalla junto con sus características de tamaño.

Código:

```
def realizarAccion(opcion,imagen):
    img = imread(imagen)                #Se carga la imagen a memoria
    if opcion==1:                       #Opcion 1 seleccionada en la interfaz gráfica
        imgrey=cvtColor(img, COLOR_BGR2GRAY)    #Convertir imagen de formato RGB a Escala de Grises
        h = img.shape[0]                        #Obtener altura de la imagen en pixeles
        w = img.shape[1]                        #Obtener ancho de la imagen en pixeles
        text="Altura : "+str(h)+" , Ancho: "+str(w)     #
        composite_img = putText(img, text, ( int(w/10) , int(h-(h/10)) ), FONT_HERSHEY_SIMPLEX,
            1.0, (255, 255, 255), 2, LINE_AA, False)    #Agregar características de la imagen como texto
        imshow('Imagen en escala de grises',composite_img)  #Mostrar imagen en una ventana nueva
        waitKey(0)  #comando para detener la imagen
        destroyAllWindows()
```

Ejemplo;


Tarea 1 RFVC
Selecciona una opción a realizar
Imagen en Niveles de Gris
Seleccionar archivo
Identificar pixel
Histograma de imagen
Imagen binaria con umbral manual
Número de objetos (sin huecos)
Número de huecos de un objeto
Simple o multiplemente conectado
Marcar contornos
Imagen en escala de grises
Altura : 569 , Ancho: 850

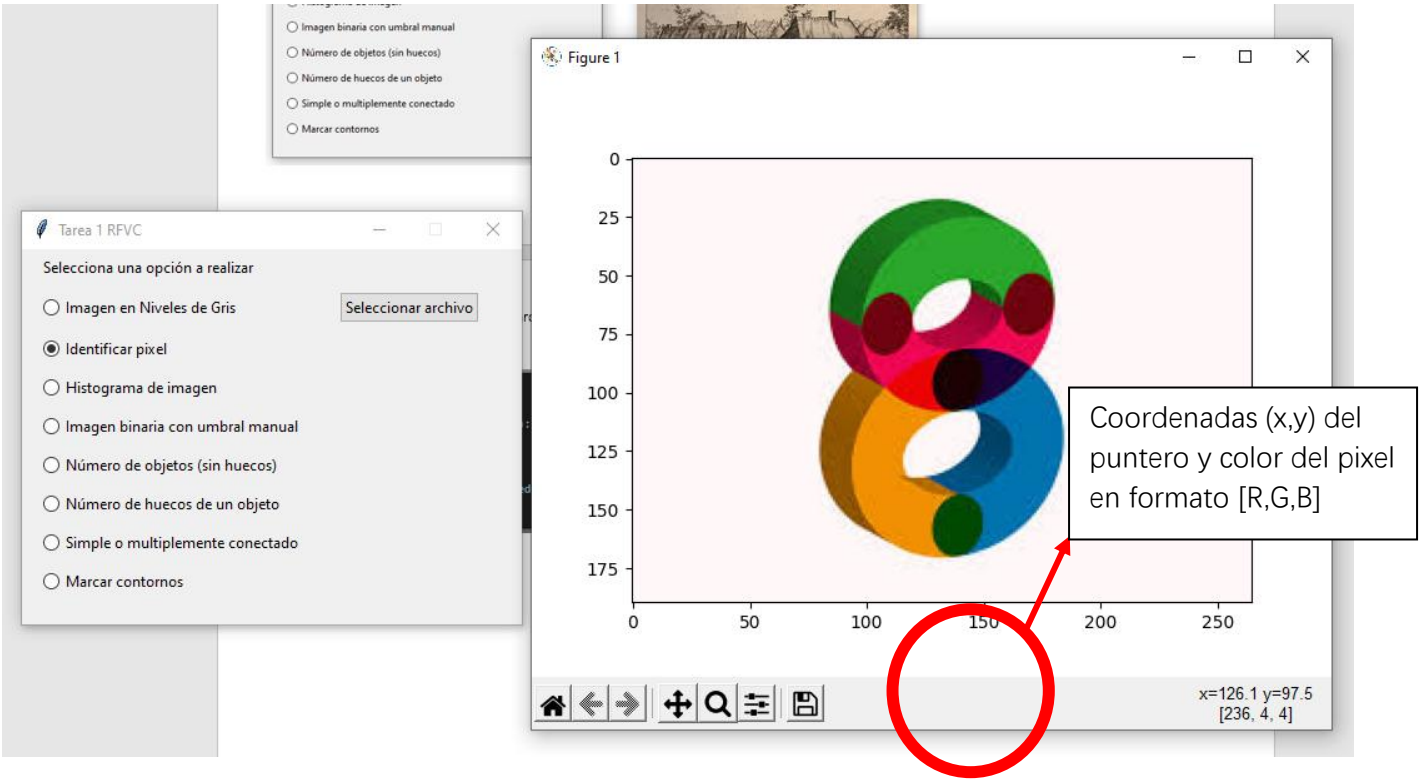

Tarea 1 RFVC
Selecciona una opción a realizar
Imagen en Niveles de Gris
Seleccionar archivo
Identificar pixel
Histograma de imagen
Imagen binaria con umbral manual
Número de objetos (sin huecos)
Número de huecos de un objeto
Simple o multiplemente conectado
Marcar contornos
Imagen en escala de grises
Altura : 234 , Ancho: 340

2. Permita con el puntero del ratón marcar un píxel dentro de la imagen y permita mostrar este valor en la pantalla.

Código:

```python
def realizarAccion(opcion,imagen):
    img = imread(imagen)                    #Se carga la imagen a memoria
    if opcion==2:                           #Opcion 2 seleccionada en la interfaz gráfica
        img_2 = img[:,:,[2,1,0]]            #Tomar todos los canales (R,G,B) de la imagen
        plt.imshow(img_2, animated= True)   #Dibujar en una ventana mediante la biblioteca matplotlib
        plt.show()                          #Mostrar la nueva ventana con matplotlib
```

Ejemplo:

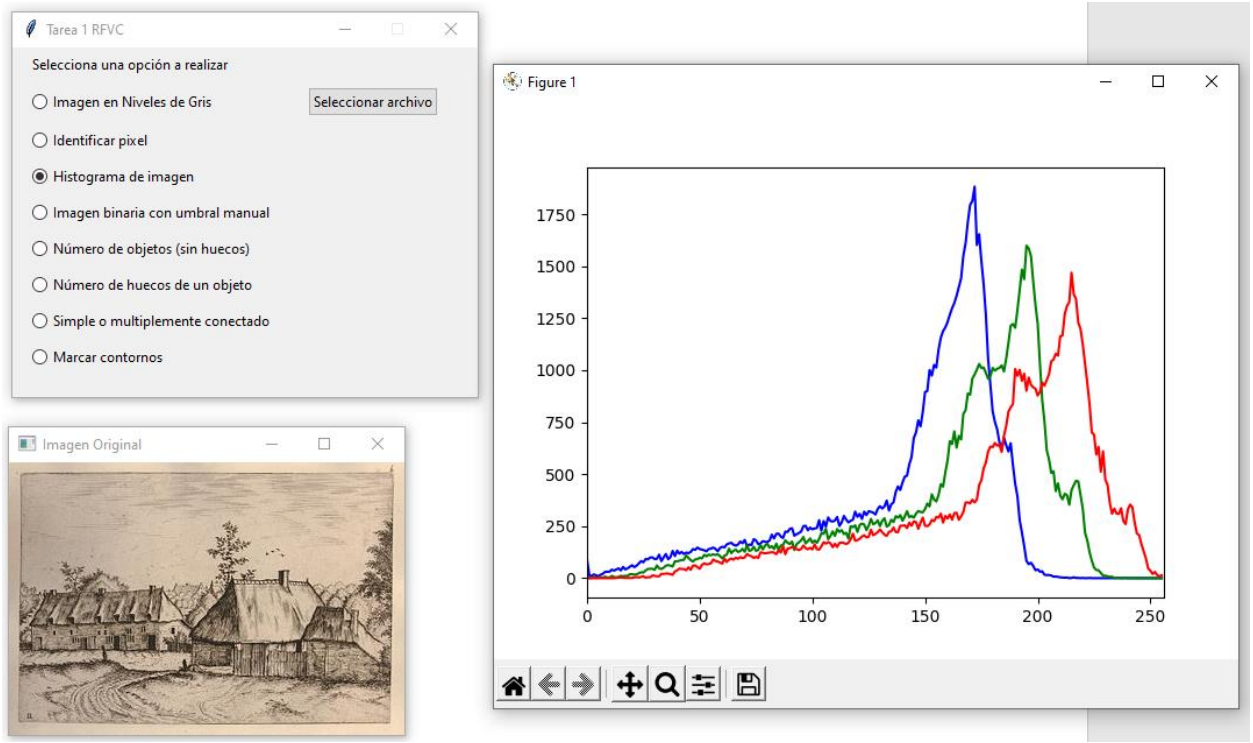


3. Permita calcular y mostrar en pantalla el histograma de dicha imagen.

Código:

```python
def realizarAccion(opcion,imagen):
        img = imread(imagen)                                #Se carga la imagen a memoria
        if opcion==3:                                       #Opcion 3 seleccionada en la interfaz gráfica
            fig,axes=plt.subplots(nrows=1, ncols=1)         #Definir el plano en el que se graficará con mathplotlib
            color = ('b','g','r')
            for i,col in enumerate(color):                  #Crear un histograma por cada color (R,G,B)
                histr = calcHist([imagen],[i],None,[256],[0,256])   #Calcular el histograma de la imagen según el color
                plt.plot(histr,color = col)                 #Dibujar el histograma en el plano asignado
                plt.xlim([0,256])                           #Definir el límite en el eje x del plano
            imshow("Imagen Original",img)                   #Mostra la imagen en una nueva ventana
            plt.show()
```

Ejemplo:

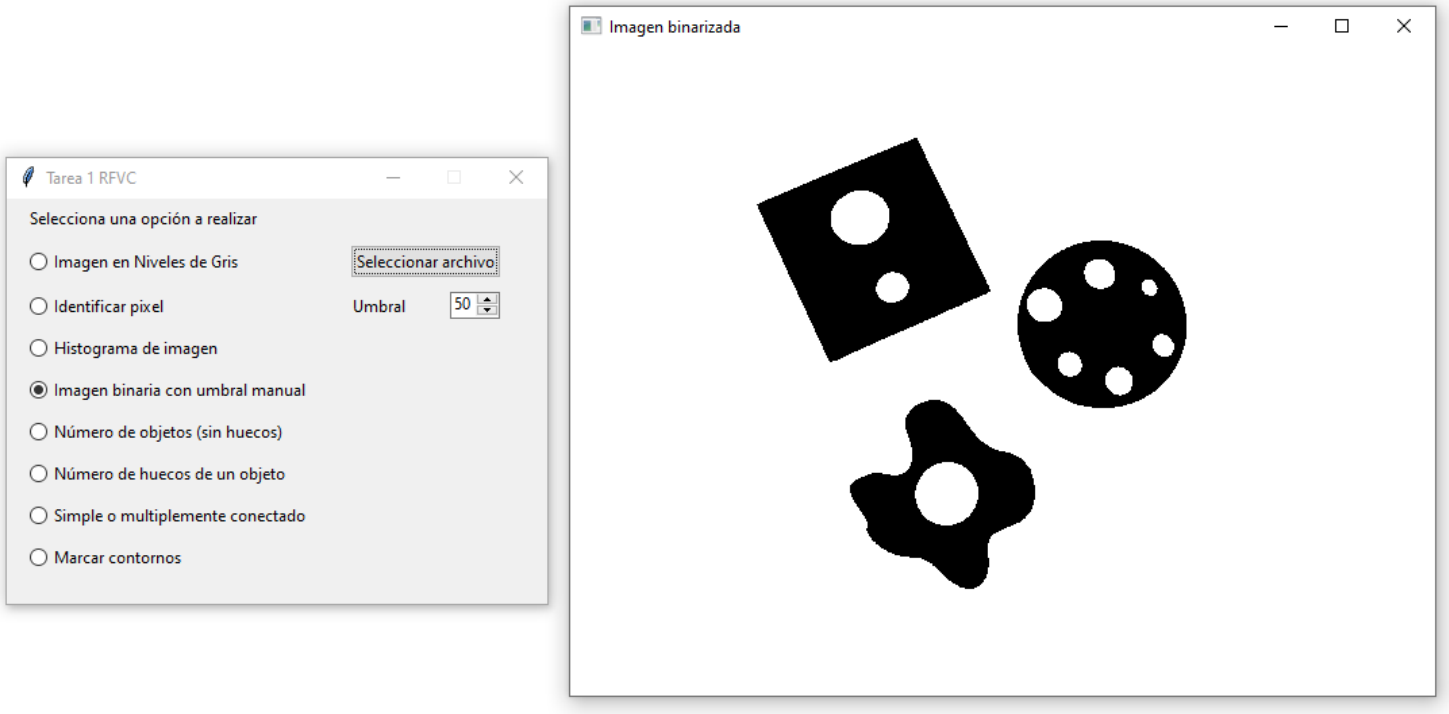


4. Permita seleccionar un valor de umbral manual entre 0 y 255 y con este umbralar dicha imagen y mostrar en pantalla la correspondiente imagen en blanco y negro (imagen binaria).

Código:

```python
def realizarAccion(opcion,imagen):
    img = imread(imagen)                                    #Se carga la imagen a memoria
    if opcion==4:                                          #Opcion 4 seleccionada en la interfaz gráfica
        umbral= int(umbralSel.get())                       #Obtener umbral que seleccionó el usuario
        t2, imgbin = threshold(img, umbral, 256, THRESH_BINARY)   #Convertir imagen a binaria con el umbral seleccionado
        imshow('Imagen binarizada',imgbin)                #Mostrar imagen en una ventana nueva
        waitKey(0)                                         #Comando para detener la imagen
        destroyAllWindows()
```

Ejemplo:

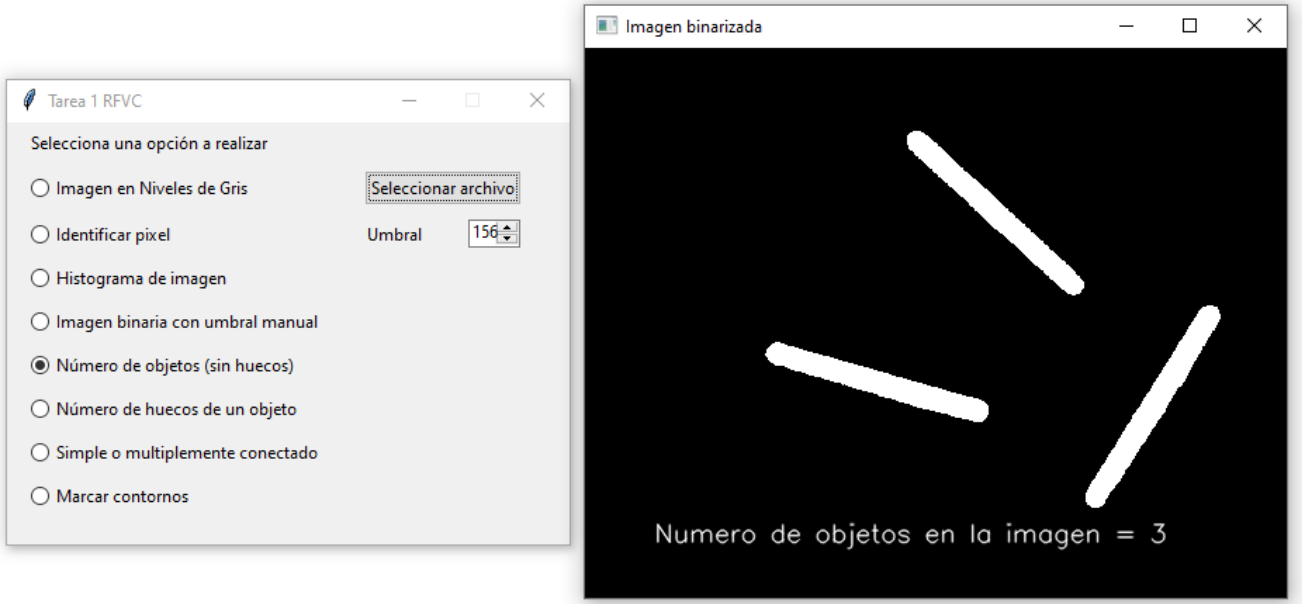


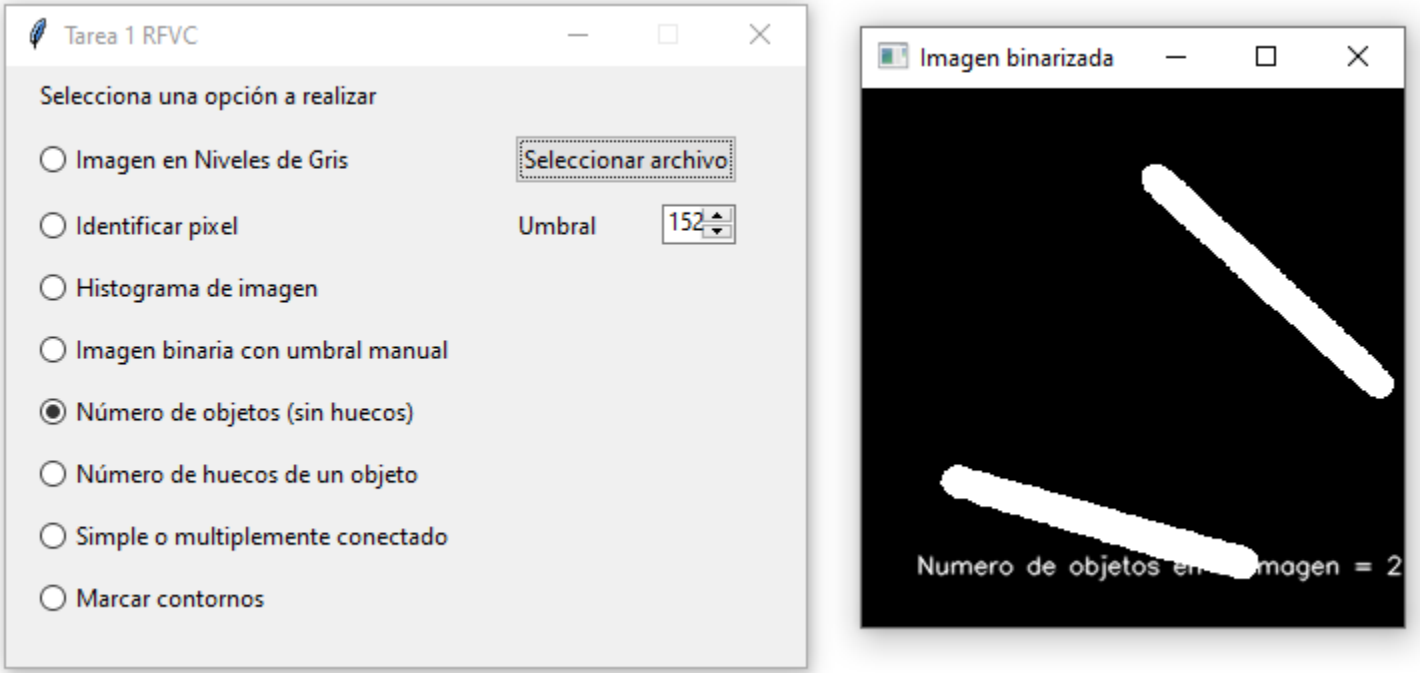
5. Permita a partir de la imagen binaria determinar el número de objetos en la imagen sin huecos. Usar la formulación basada en bit-quads.

Decidí que el usuario pueda cambiar el umbral al convertir la imagen de escala de grises a binaria, para que pueda observar la diferencia entre escoger un umbral y otro.
Código:

```
def realizarAccion(opcion,imagen):
        img = imread(imagen)                                        #Se carga la imagen a memoria
        if opcion==5:                                               #Opcion 5 seleccionada en la interfaz gráfica
            umbral= int(umbralSel.get())                            #Obtener umbral que seleccionó el usuario
            imgrey=cvtColor(img, COLOR_BGR2GRAY)                    #Se convierte la imagen a escala de grises
            t2, imgbin = threshold(imgrey, umbral, 255, THRESH_BINARY)  #Se binariza la imagen
            arreglo= np.asarray(imgbin)                             #Convertir la información de la imagen a un arreglo numpy
            n1=0
            n2=0
            n3=0
            h = img.shape[0]                        #Obtener altura de la imagen en pixeles
            w = img.shape[1]                        #Obtener ancho de la imagen en pixeles
            """Se recorre la imagen para buscar las máscaras
                |1 0|       |1 1|       |1 0|
                |0 0|       |1 0|       |0 1|

            """
            for fil,array in enumerate(arreglo):
                for col,a in enumerate(array):
                    if a==255:
                        if col < len(array)-1 and fil < len(arreglo)-1 and col>1 and fil>1: #Evitar buscar máscaras en los bordes
                            if arreglo[fil,col+1] == 0   and arreglo[fil+1,col] == 0   and arreglo[fil+1,col+1] == 0:
                                n1+=1                                   #Contar mascara 1
                            if arreglo[fil,col+1] == 255 and arreglo[fil+1,col] == 255 and arreglo[fil+1,col+1] == 0:
                                n2+=1                                   #Contar mascara 2
                            if arreglo[fil,col-1] == 0   and arreglo[fil+1,col-1] == 255   and arreglo[fil+1,col] == 0:
                                n3+=1                                   #Contar mascara 3

            text="Numero de objetos en la imagen = " + abs(n1-n2+n3)            #Texto sobre la información de la imagen
            composite_img = putText(imgbin, text, ( int(w/10) , int(h-(h/10)) ), FONT_HERSHEY_SIMPLEX,
                0.6, (255, 255, 255), 2, LINE_AA, False)                        #Agregar el número de objetos de la imagen como texto
            imshow('Imagen binarizada ',imgbin)                         #Mostrar imagen en una ventana nueva
            waitKey(0)                                                  #Comando para detener la imagen
            destroyAllWindows()
```

Ejemplo:


Tarea 1 RFVC
Selecciona una opción a realizar
Imagen en Niveles de Gris
Seleccionar archivo
Identificar pixel
Umbral
156
Histograma de imagen
Imagen binaria con umbral manual
Número de objetos (sin huecos)
Número de huecos de un objeto
Simple o multiplemente conectado
Marcar contornos
Imagen binarizada
Numero de objetos en la imagen = 3


Tarea 1 RFVC
Selecciona una opción a realizar
Imagen en Niveles de Gris
Seleccionar archivo
Identificar pixel
Umbral
152
Histograma de imagen
Imagen binaria con umbral manual
Número de objetos (sin huecos)
Número de huecos de un objeto
Simple o multiplemente conectado
Marcar contornos
Imagen binarizada
Numero de objetos en la imagen = 2

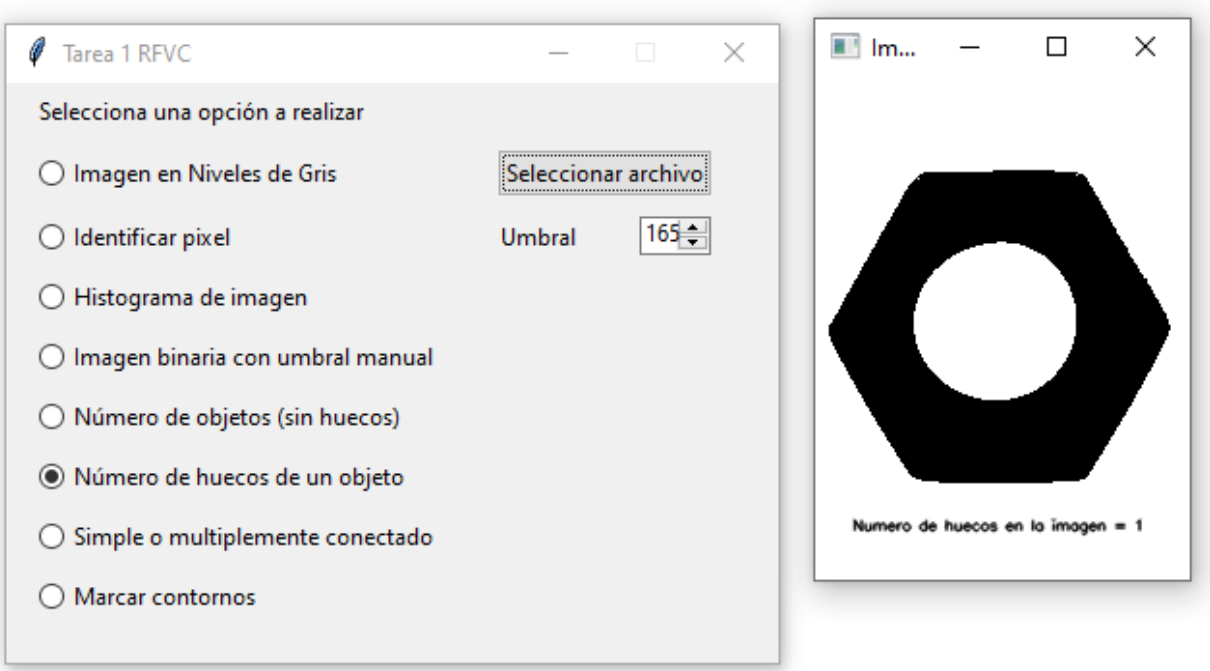
6. Permita a partir de una imagen binaria con un solo objeto, su número de huecos. Usar la formulación basada en bit-quads.

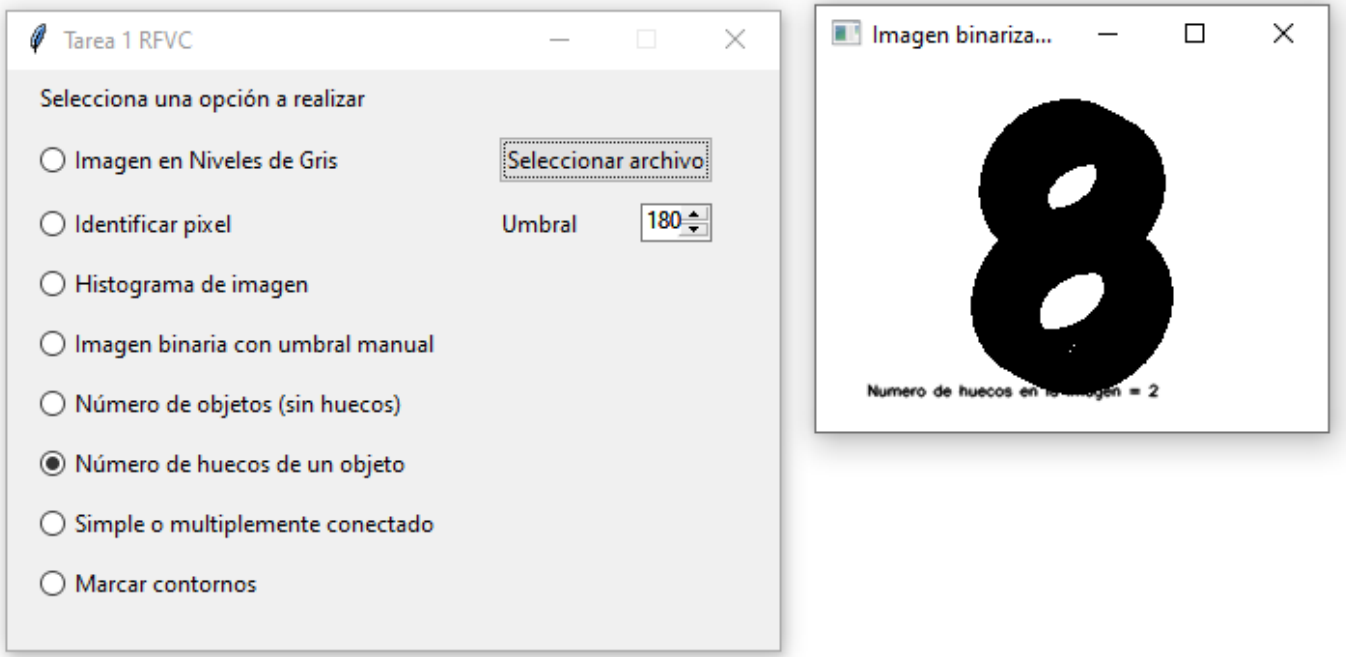
Decidí que el usuario pueda cambiar el umbral al convertir la imagen de escala de grises a binaria, para que pueda observar la diferencia entre escoger un umbral y otro.

Código:

```
def realizarAccion(opcion,imagen):
        img = imread(imagen)                            #Se carga la imagen a memoria
        if opcion==6:
            umbral=int(umbralSel.get())
            imgrey=cvtColor(img, COLOR_BGR2GRAY)        #Se convierte a escala de grises
            t2, imgbin = threshold(imgrey, umbral, 255, THRESH_BINARY)  #Se binariza la imagen
            h = img.shape[0]                           #Obtener altura de la imagen en pixeles
            w = img.shape[1]                           #Obtener ancho de la imagen en pixeles
            arreglo= np.asarray(imgbin)                #Convertir la información de la imagen a un arreglo numpy

            n1=0
            n2=0
            n3=0
            """Se recorre la imagen para buscar las máscaras
                |1 0|       |1 1|       |1 0|
                |0 0|       |1 0|       |0 1|
            """
            for fil,array in enumerate(arreglo):
                for col,a in enumerate(array):
                    if a==255:
                        if col < len(array)-1 and fil < len(arreglo)-1 and col>1 and fil>1: #Evitar buscar máscaras en los bordes
                            if arreglo[fil,col+1] == 0   and arreglo[fil+1,col] == 0   and arreglo[fil+1,col+1] == 0:
                                n1+=1                                   #Contar mascara 1
                            if arreglo[fil,col+1] == 255 and arreglo[fil+1,col] == 255 and arreglo[fil+1,col+1] == 0:
                                n2+=1                                   #Contar mascara 2
                            if arreglo[fil,col-1] == 0   and arreglo[fil+1,col-1] == 255   and arreglo[fil+1,col] == 0:
                                n3+=1                                   #Contar mascara 3
            text="Numero de huecos en la imagen = " + str(abs(1-(n1-n2+n3)))          #Texto sobre la información de la imagen
            composite_img = putText(imgbin, text, ( int(w/10) , int(h-(h/10)) ), FONT_HERSHEY_SIMPLEX,
                0.6, (255, 255, 255), 1, LINE_AA, False)                       #Agregar el número de objetos de la imagen como texto

            imshow('Imagen binarizada ',composite_img)
            waitKey(0)                                                  #Comando para detener la imagen
            destroyAllWindows()
```

Ejemplos:


Tarea 1 RFVC
Selecciona una opción a realizar
Imagen en Niveles de Gris
Seleccionar archivo
Identificar pixel
Umbral
165
Histograma de imagen
Imagen binaria con umbral manual
Número de objetos (sin huecos)
Número de huecos de un objeto
Simple o multiplemente conectado
Marcar contornos
Im...
Numero de huecos en la imagen = 1

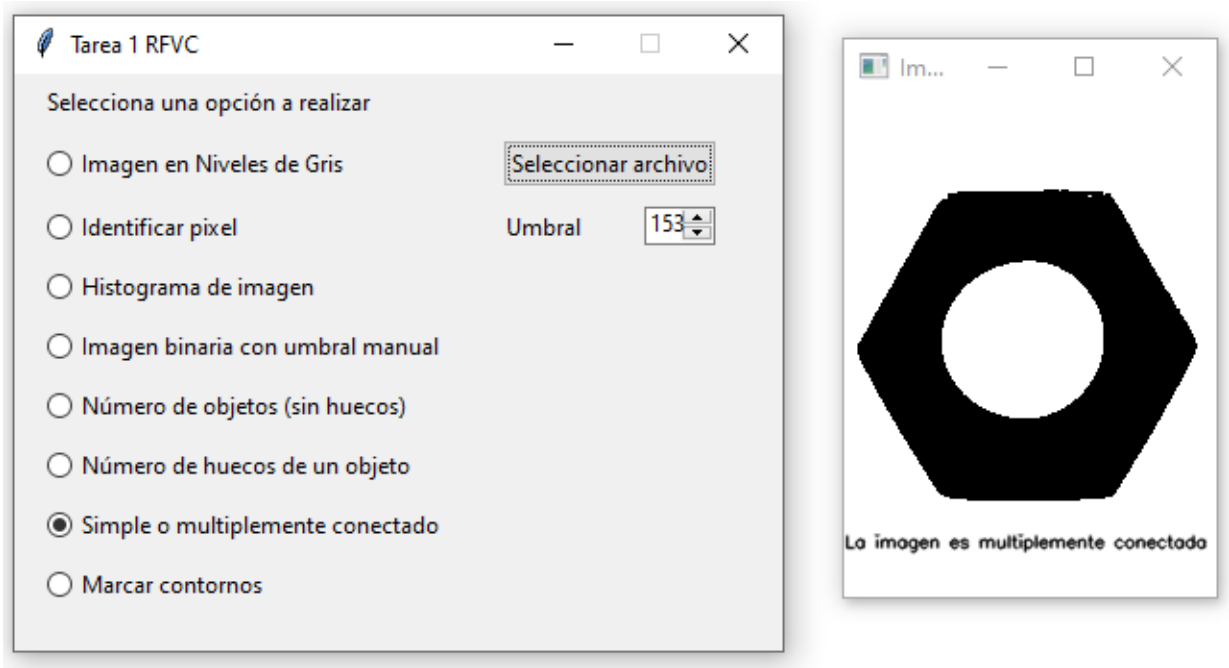

Tarea 1 RFVC
Selecciona una opción a realizar
Imagen en Niveles de Gris
Seleccionar archivo
Identificar pixel
Umbral
180
Histograma de imagen
Imagen binaria con umbral manual
Número de objetos (sin huecos)
Número de huecos de un objeto
Simple o multiplemente conectado
Marcar contornos
Imagen binariza...
Numero de huecos en

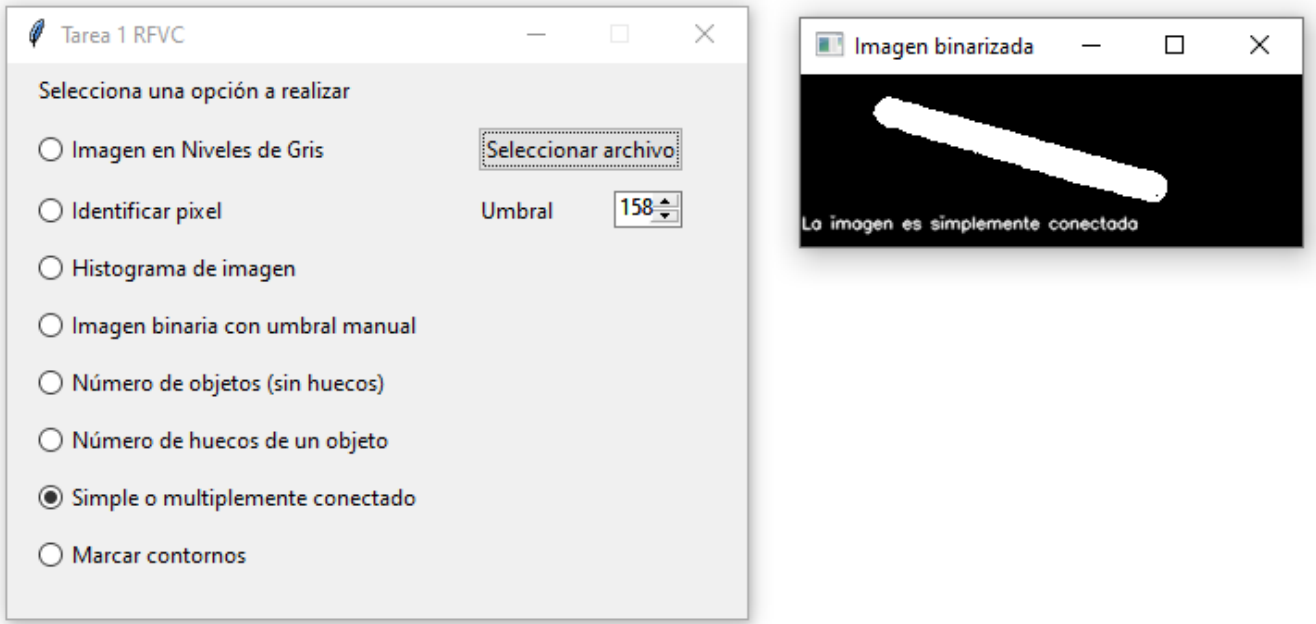
7. Permita a partir de una imagen binaria con un solo objeto, si éste es simple o múltiplemente conectado. Usar la formulación basada en bit-quads. Decidí que el usuario pueda cambiar el umbral al convertir la imagen de escala de grises a binaria, para que pueda observar la diferencia entre escoger un umbral y otro.

Código:

```
def realizarAccion(opcion,imagen):
        img = imread(imagen)                        #Se carga la imagen a memoria
        if opcion==7:                               #Opcion 7 seleccionada en la interfaz gráfica
            umbral=int(umbralSel.get())             #Obtener umbral que seleccionó el usuario
            imgrey=cvtColor(img, COLOR_BGR2GRAY)    #Se convierte a escala de grises
            t2, imgbin = threshold(imgrey, umbral, 255, THRESH_BINARY)  #Se binariza
            h = img.shape[0]                        #Obtener altura de la imagen en pixeles
            w = img.shape[1]                        #Obtener ancho de la imagen en pixeles
            arreglo= np.asarray(imgbin)             #Convertir la información de la imagen a un arreglo numpy
            n1=0
            n2=0
            n3=0
            """Se recorre la imagen para buscar las máscaras
                |1 0|       |1 1|       |1 0|
                |0 0|       |1 0|       |0 1|
            """
            for fil,array in enumerate(arreglo):
                for col,a in enumerate(array):
                    if a==255:
                        if col < len(array)-1 and fil < len(arreglo)-1 and col>1 and fil>1:#Evitar buscar máscaras en los bordes
                            if arreglo[fil,col+1] == 0   and arreglo[fil+1,col] == 0   and arreglo[fil+1,col+1] == 0:
                                n1+=1                       #Contar mascara 1
                            if arreglo[fil,col+1] == 255 and arreglo[fil+1,col] == 255 and arreglo[fil+1,col+1] == 0:
                                n2+=1                       #Contar mascara 2
                            if arreglo[fil,col-1] == 0   and arreglo[fil+1,col-1] == 255   and arreglo[fil+1,col] == 0:
                                n3+=1                       #Contar mascara 3
            if 1-(n1-n2+n3)!=0:
                text="La imagen es simplemente conectada" + str(abs(1-(n1-n2+n3)))          #Texto sobre la información de la imagen
            else:
                text="La imagen es multiplemente conectada" + str(abs(1-(n1-n2+n3)))        #Texto sobre la información de la imagen
            composite_img = putText(imgbin, text, ( int(w/10) , int(h-(h/10)) ), FONT_HERSHEY_SIMPLEX,
                0.6, (255, 255, 255), 1, LINE_AA, False)                        #Agregar el número de objetos de la imagen como texto

            imshow('Imagen binarizada ',composite_img)                  #Mostrar imagen en una nueva ventana
            waitKey(0)                                                  #Comando para detener la imagen
            destroyAllWindows()
```

Ejemplo:


Tarea 1 RFVC
Selecciona una opción a realizar
Imagen en Niveles de Gris
Seleccionar archivo
Identificar pixel
Umbral
153
Histograma de imagen
Imagen binaria con umbral manual
Número de objetos (sin huecos)
Número de huecos de un objeto
Simple o multiplemente conectado
Marcar contornos
Im...
La imagen es multiplemente conectada


Tarea 1 RFVC
Selecciona una opción a realizar
Imagen en Niveles de Gris
Seleccionar archivo
Identificar pixel
Umbral
158
Histograma de imagen
Imagen binaria con umbral manual
Número de objetos (sin huecos)
Número de huecos de un objeto
Simple o multiplemente conectado
Marcar contornos
Imagen binarizada
La imagen es simplemente conectada

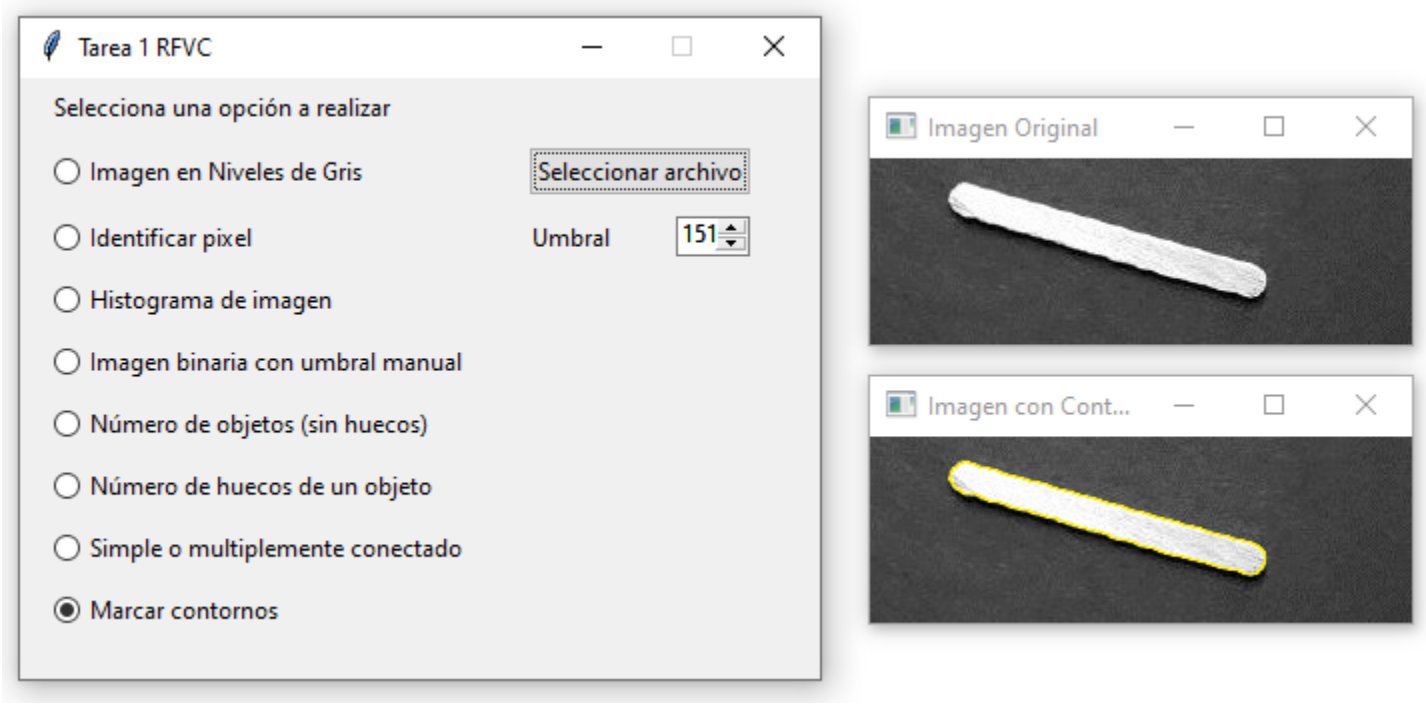
8. Permita a partir de una imagen binaria con uno o más objetos, con y sin huecos marcar en color, por ejemplo, amarillo, los píxeles de todos los contornos. Use el método basado en vecindades, muestre el resultado para ambos casos 4 y 8 conectado.

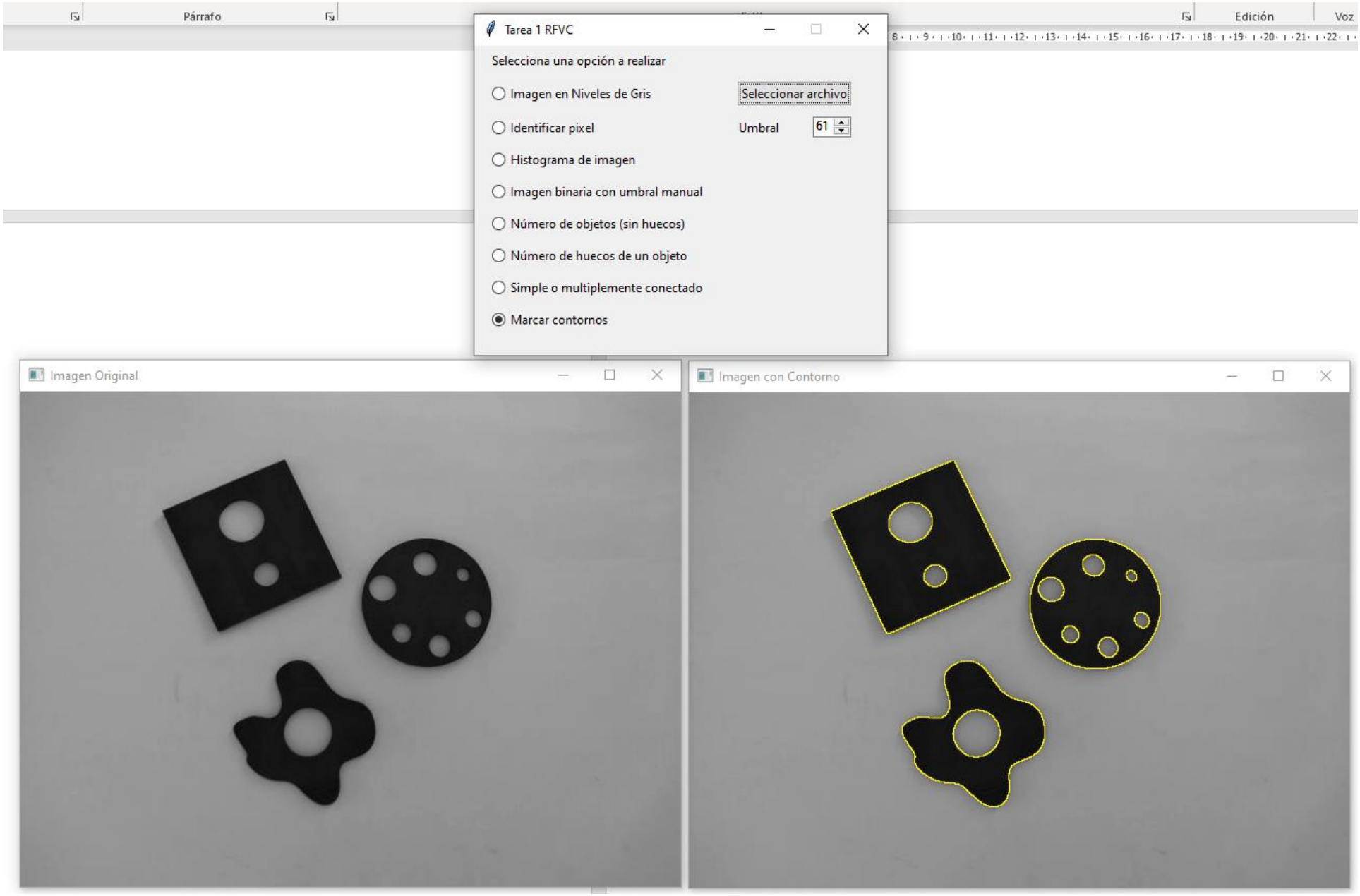
Decidí que el usuario pueda cambiar el umbral al convertir la imagen de escala de grises a binaria, para que pueda observar la diferencia entre escoger un umbral y otro.

Código:

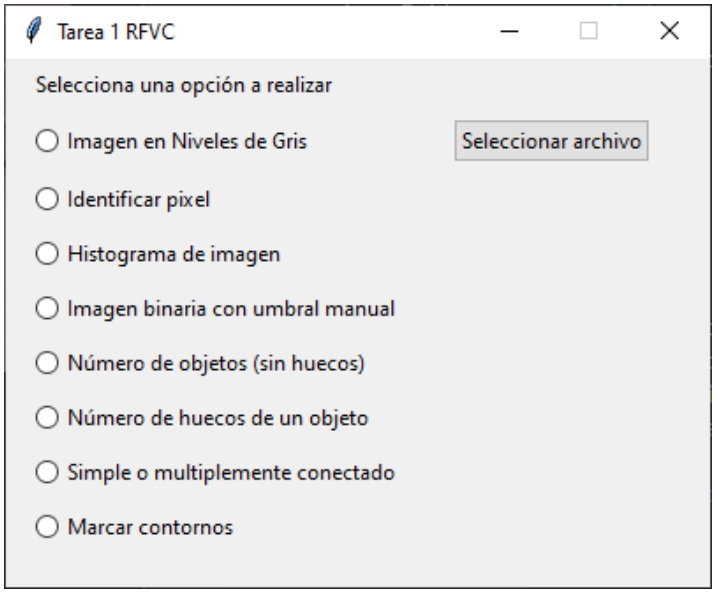
```
def realizarAccion(opcion,imagen):
        img = imread(imagen)                        #Se carga la imagen a memoria
        if opcion==8:
            umbral=int(umbralSel.get())             #Obtener umbral que seleccionó el usuario
            imagenO= imread(imagen)                 #Se obtiene la imagen
            img= imread(imagen)                     #Se obtiene la imagen
            imgrey=cvtColor(img, COLOR_BGR2GRAY)    #Se convierte a escala de grises
            t2, imgbin = threshold(imgrey, umbral, 255, THRESH_BINARY)  #Se binariza la imagen
            arreglo = np.asarray(img)               #Convertir la información de la imagen Original a un arreglo numpy
            arregloBin = np.asarray(imgbin)         #Convertir la información de la imagen Binarizada a un arreglo numpy
            n1=0
            n2=0
            n3=0
            #Buscar los pixeles que forman parte del objeto
            for fil,array in enumerate(arregloBin):
                for col,a in enumerate(array):
                    if a==255:
                        if col < len(array)-1 and col > 0 and fil < len(arregloBin)-1 and fil > 0: #Evitar buscar fuera de los bordes
                            #Buscar si en la 4-vecindad hay algún bit que forme parte del fondo de la imagen
                            if arregloBin[fil,col+1]==0 or arregloBin[fil,col-1]==0 or arregloBin[fil+1,col]==0 or arregloBin[fil-1,col]==0:
                                #Cambiar el color del pixel en la imagen original
                                arreglo[fil][col][0]=0
                                arreglo[fil][col][1]=240
                                arreglo[fil][col][2]=255
                            #Buscar si en la 8-vecindad hay algún bit que forme parte del fondo de la imagen
                            elif arregloBin[fil+1,col+1]==0 or arregloBin[fil+1,col-1]==0 or arregloBin[fil-1,col+1]==0 or arregloBin[fil-1,col-1]==0:
                                #Cambiar el color del pixel en la imagen original
                                arreglo[fil][col][0]=0
                                arreglo[fil][col][1]=240
                                arreglo[fil][col][2]=255

            imshow('Imagen Original ',imagenO)      #Mostrar imagen original en una nueva ventana
            imshow('Imagen con Contorno ',arreglo)  #Mostrar imagen con contorno en una nueva ventana

        waitKey(0)  #comando para detener la imagen
        destroyAllWindows()
```

Ejemplo:

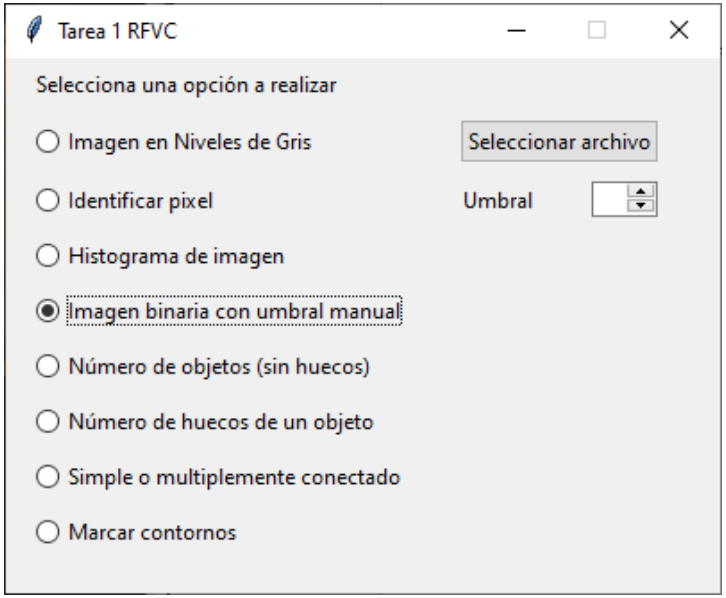

Tarea 1 RFVC
Selecciona una opción a realizar
Imagen en Niveles de Gris
Seleccionar archivo
Identificar pixel
Umbral
151
Histograma de imagen
Imagen binaria con umbral manual
Número de objetos (sin huecos)
Número de huecos de un objeto
Simple o multiplemente conectado
Marcar contornos
Imagen Original
Imagen con Cont...


Párrafo
Edición
Voz
Tarea 1 RFVC
Selecciona una opción a realizar
Imagen en Niveles de Gris
Identificar pixel
Histograma de imagen
Imagen binaria con umbral manual
Número de objetos (sin huecos)
Número de huecos de un objeto
Simple o multiplemente conectado
Marcar contornos
Seleccionar archivo
Umbral
61
Imagen Original
Imagen con Contorno

Interfaz Gráfica realizada con la biblioteca de Python *tkinter* (1):



Interfaz Gráfica realizada con la biblioteca de Python *tkinter* (2):



Selección de archivo de tipo imagen (JPG, JPEG, PNG)