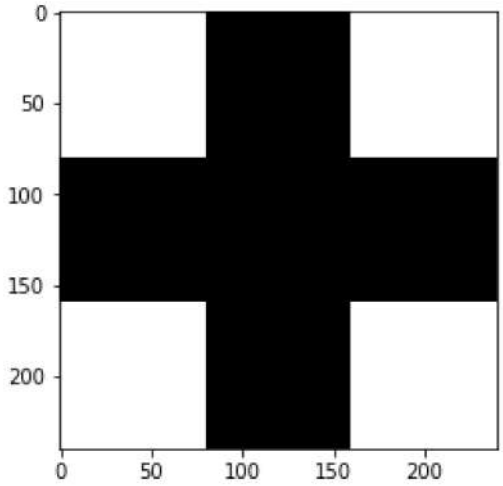
## 1) Realice y grafique una imagen con una cruz negra y fondo blanco.

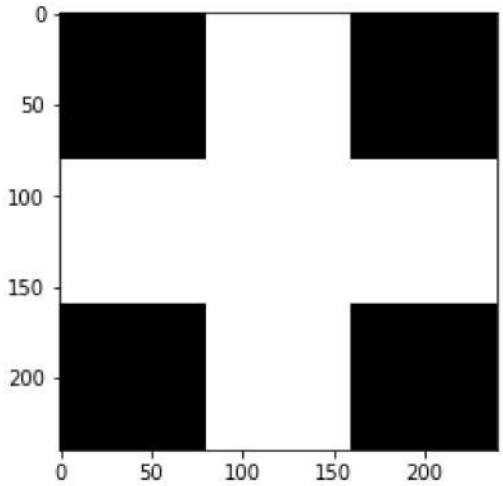
In [628]:
```
import numpy as np
import matplotlib.pyplot as plt

negro=np.zeros((80,80))
blanco=negro+1

fila=np.concatenate((blanco,negro,blanco),axis=1)

cruz=np.concatenate((fila, np.zeros((80,240)),fila),axis=0)

plt.imshow(cruz, cmap='gray')
plt.show()
print(cruz.shape)
```




```
(240, 240)
```

## 2) Realice y grafique una imagen con una cruz blanca y fondo negro (negativo de la primera).

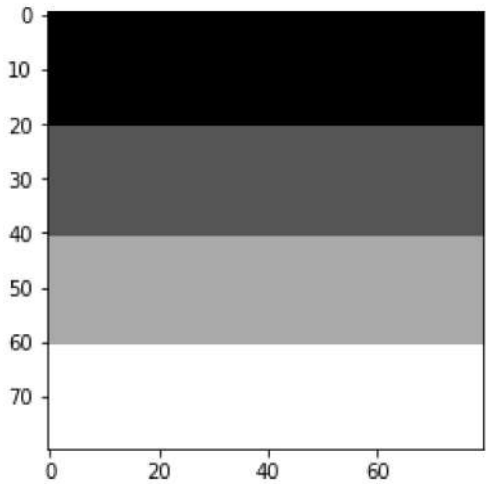
In [49]:
```
import numpy as np
import matplotlib.pyplot as plt
cruzinvertida=np.zeros((240,240))
cruzinvertida[:,80:160]=0.5
cruzinvertida[80:160,:]=0.5
plt.imshow(cruzinvertida, cmap='gray')
plt.show()
print(cruzinvertida.shape)
```




```
(240, 240)
```

### 3) Realice y grafique, una imagen con cuatro bandas h orizontales de intensidad homogénea decreciente de 1 a 0de arriba abajo.

In [101]:
```
blanco=np.ones((80,80))
banda=np.linspace(0,1,4)

blanco[0:21,:]=banda[0]
blanco[21:41,:]=banda[1]
blanco[41:61,:]=banda[2]
blanco[61:81,:]=banda[3]


plt.imshow(blanco, cmap='gray',vmin=0 ,vmax=1)
plt.show()
print(banda)
```

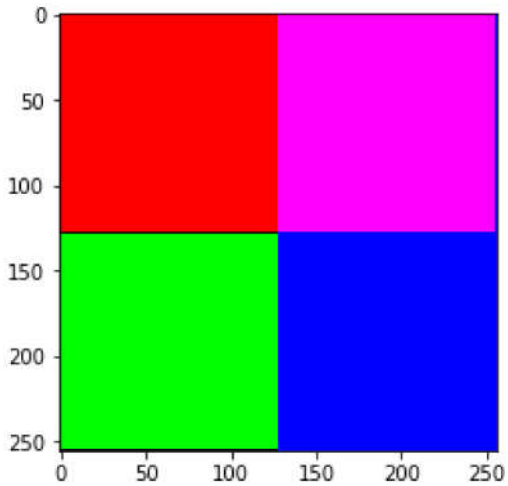



```
[0.         0.33333333 0.66666667 1.        ]
```

## 4) Construir y graficar una imagen 128x128 dividida en cuatro cuadrados de colores: rojo, azul, amarillo y uno a elección.

In [45]:
```
import matplotlib.pyplot as plt
import numpy as np
arr_image=np.zeros((256,256,3),'uint8')
print(type(arr_image[0,0,0]))
c=range(256)
#print(c)

for i in c:
    arr_image[0:128,0:255,0]=255 #rojo creciente en las primeras 128 columnas
    arr_image[129:255,0:128,1]=(255) #verde decreciente en las primeras 128 co
lumnas

arr_image[:,128:,2]=255 #(solo) azul en las últimas columnas

#Muestro filas 1, 127 y 128, 256, primera columna
#print(arr_image[0,0,:])
#print(arr_image[127,0,:])
#print(arr_image[128,0,:])
#print(arr_image[255,0,:])

print("R-rojo:")
print( arr_image[: ,:, 0])
print("G-verde:")
print( arr_image[:,:,1])
print("B-azul:")
print( arr_image[:,:,2])
plt.imshow(arr_image)

plt.show()
```

```
<class 'numpy.uint8'>
R-rojo:
[[255 255 255 ... 255 255   0]
 [255 255 255 ... 255 255   0]
 [255 255 255 ... 255 255   0]
 ...
 [  0   0   0 ...   0   0   0]
 [  0   0   0 ...   0   0   0]
 [  0   0   0 ...   0   0   0]]
G-verde:
[[  0   0   0 ...   0   0   0]
 [  0   0   0 ...   0   0   0]
 [  0   0   0 ...   0   0   0]
 ...
 [255 255 255 ...   0   0   0]
 [255 255 255 ...   0   0   0]
 [  0   0   0 ...   0   0   0]]
B-azul:
[[  0   0   0 ... 255 255 255]
 [  0   0   0 ... 255 255 255]
 [  0   0   0 ... 255 255 255]
 ...
 [  0   0   0 ... 255 255 255]
 [  0   0   0 ... 255 255 255]
 [  0   0   0 ... 255 255 255]]
```



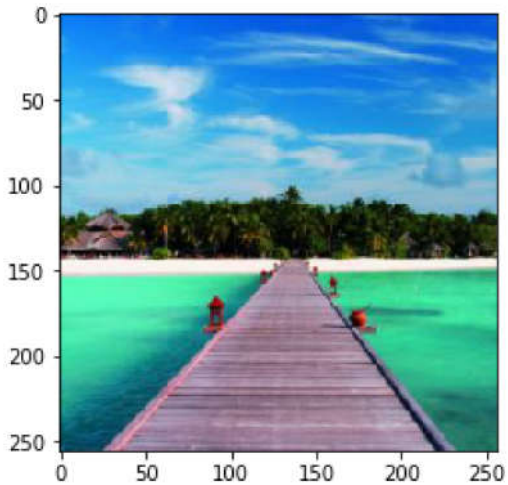

In [112]:
```
print(c)
print(c[255])
```

```
range(0, 256)
255
```

## 5) Abrir una imagen guardada en un archivo, convertirla en Imagen monocroma, y graficar ambas imágenes a la par, colocarle título a ambas.

In [120]:
```
import SimpleITK as sitk
path="paisajepuente.jpg" #nombre de la imagen (y ubicación, si está ubicada en otra carpeta)
ima=sitk.ReadImage(path) #lee el archivo y genera una variable TIPO IMAGEN
ima_arr = sitk.GetArrayFromImage(ima) #variable TIPO ARREGLO (3D)
print("La imagen", path)
print("es una imagen de tamaño",ima_arr.shape)
print("tipo de dato",type(ima_arr))
#sitk.Show(ima) #help(sitk.Show) #This function requires that Fiji ( https://fiji.sc ) or ImageJ ( http://rsb.info.nih.gov/ij/)
#be properly installed for Mac and Windows
import matplotlib.pyplot as plt
#se grafica usando el TIPO ARREGLO (3D)que en este caso se consiguió con la función de SimpleITk
#ima_arr = sitk.GetArrayFromImage(ima)
plt.imshow(ima_arr)
plt.show()
#Responde en segunda orden
```

```
La imagen paisajepuente.jpg
es una imagen de tamaño (256, 256, 3)
tipo de dato <class 'numpy.ndarray'>
```




In [121]:
```
monocroma=np.transpose(sum(np.transpose(ima_arr)/3)) #intensidad monocroma=(rojo+verde+azul)/3
print("dimensiones de imagen monocroma  (R+G+B)/3=" , monocroma.shape)
```

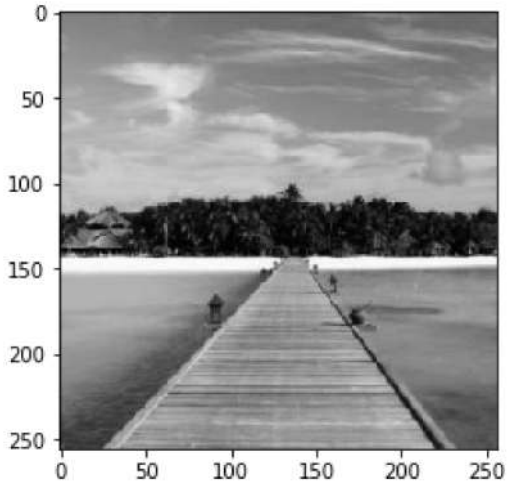
```
dimensiones de imagen monocroma  (R+G+B)/3= (256, 256)
```

In [128]:
```
print(np.max(monocroma))
```

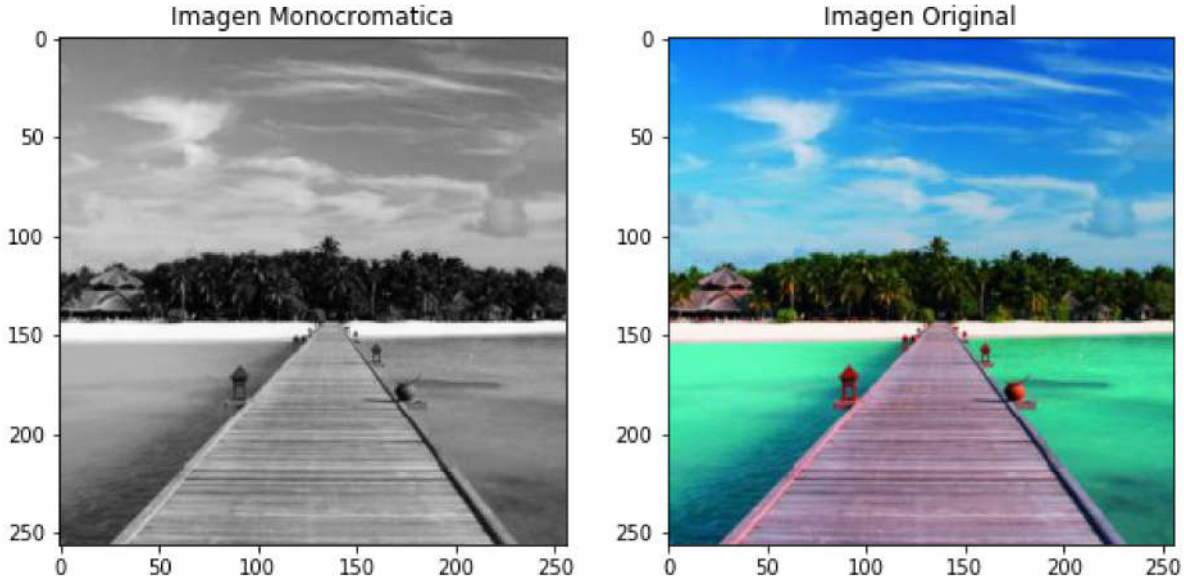
```
250.0
```

In [125]:
```
plt.imshow(monocroma,cmap='gray')
```

Out[125]: <matplotlib.image.AxesImage at 0x2d45c7f2a20>




In [131]:
```
fig, axs = plt.subplots(nrows=1, ncols=2, sharex=True)
fig.set_figheight(8)
fig.set_figwidth(10)
axs[0].imshow(monocroma, cmap='gray')
axs[0].set_title('Imagen Monocromatica')
axs[1].imshow(ima_arr)
axs[1].set_title('Imagen Original')
plt.show()
```




## 6) Construir y graficar una imagen a color que mezcle una imagen con paisaje a la que se le superponga una persona de otra imagen.

In [32]:
```
import cv2
img = cv2.imread('huron.jpg')# abrimos la imagen
#img = img.astype('uint8')
#print(img.shape)#dimensiones de la image
#visualización:
cv2.imshow('image',img)
cv2.waitKey()
cv2.destroyAllWindows()
```

In [33]:
```
#Parámetros para el demarcado de la ROI
showCrosshair = False
fromCenter = True#False
#demarcado manual de la ROI
roi = cv2.selectROI('delimite la ROI+ent+tecla', img, fromCenter, showCrosshai
r)
cv2.waitKey()
cv2.destroyAllWindows()
```

In [34]:
```
print('ROI:',roi)
print('(columna, fila, anchocol, anchofila)')
```

```
ROI: (117, 165, 524, 318)
(columna, fila, anchocol, anchofila)
```

In [35]:
```
f1=roi[1]
fu=roi[1]+roi[3]
c1=roi[0]
cu=roi[0]+roi[2]
print(f1,fu,c1,cu)

recorte=img[f1:fu,c1:cu,:]
print(recorte.shape)
cv2.imshow('imagen recorte',recorte)
cv2.waitKey()
cv2.destroyAllWindows()
```

```
165 483 117 641
(318, 524, 3)
```

In [36]:
```
import cv2
img2 = cv2.imread('arboles.jpg')# abrimos la imagen
print(img2.shape)#dimensiones de la image
#visualización:
cv2.imshow('image',img2)
cv2.waitKey()
cv2.destroyAllWindows()
print('es muy grande la imagen para visualizarla completa ')
```

```
(1704, 2272, 3)
es muy grande la imagen para visualizarla completa
```

In [37]:
```
print('alto de imagen original dividido 750 es =',img2.shape[0]/750, 'usaremos
un redondeo entero como escala')
```

```
alto de imagen original dividido 750 es = 2.272 usaremos un redondeo entero c
omo escala
```

In [38]:
```
#visualización de la imagen 'paisaje.jpg' escalada
escala=3
tam1=int(img2.shape[0]/escala)
tam2=int(img2.shape[1]/escala)
print('nuevas dimensiones para visualizar: ',tam1,tam2, ';  filas y columnas o
riginales reducidas por ', escala)
cv2.namedWindow('image',cv2.WINDOW_NORMAL)
cv2.resizeWindow('image', tam2,tam1) #tam2=# x's = #columnas y #tam1=# y's = #
filas
cv2.imshow('image',img2)
cv2.waitKey()
cv2.destroyAllWindows()
```

```
nuevas dimensiones para visualizar:  568 757 ;  filas y columnas originales r
educidas por  3
```

In [39]:
```
imagen_nueva=img2.copy()
print(img.shape)
print(recorte.shape)
print('des-comentar la línea que sigue dará error pues el recorte se sale de r
ango en la imagen del paisaje')
#imagen_nueva[f1:fu,c1:cu,:]=recorte
```

```
(607, 910, 3)
(318, 524, 3)
des-comentar la línea que sigue dará error pues el recorte se sale de rango e
n la imagen del paisaje
```

In [41]:
```
F=700
C=40
imagen_nueva=img2.copy() #paisaje nuevo
imagen_nueva[(f1-F):(fu-F), (c1-C):(cu-C),:]=recorte #sobre-escribimos el reco
rte sobre el paisaje
#graficamos con la misma escala para paisaje original
cv2.namedWindow('image',cv2.WINDOW_NORMAL)
cv2.resizeWindow('image', tam2,tam1) #tam2=# x's = #columnas y #tam1=# y's = #
filas
cv2.imshow('image',imagen_nueva)
cv2.waitKey()
cv2.destroyAllWindows()
```

## 7) Cargar una imagen monocroma y realizar por separado las siguientes modificaciones : Aclararla, Oscurecerla y Obtener el negativo.

In [47]:
```
import PIL
from PIL import Image

path="pajaromonocromo.jpg"
I=Image.open(path) # El tipo de dato es imagen
Igris=I.convert('L')# Se crea una imagen monocroma

# get an array
import numpy as np
f = np.asarray(Igris)#Transforma el dato en array 'uint8' monocroma (2D)
print("La matriz correspondiente a la imagen es de dimensión ",f.shape," \nSu
 tipo es ", type(f),
      "\nCada elemento es de tipo", type(f[0,0]))

import matplotlib.pyplot as plt
%matplotlib inline

plt.imshow(f,cmap="gray")
plt.axis('off')#para quitar la numeración de filas y columnas
plt.colorbar() #para la barra con la referencia de color
plt.show()
```

```
La matriz correspondiente a la imagen es de dimensión  (229, 300)
Su tipo es  <class 'numpy.ndarray'>
Cada elemento es de tipo <class 'numpy.uint8'>
```

In [48]:
```
import SimpleITK as sitk
pajaro="pajaromonocromo.jpg" #nombre de la imagen (y ubicación, si está ubicad
a en otra carpeta)
ima=sitk.ReadImage(pajaro) #lee el archivo y genera una variable TIPO IMAGEN
pajaro_arr = sitk.GetArrayFromImage(ima) #variable TIPO ARREGLO (3D)
#print("La imagen", pajaro)
print("es una imagen de tamaño",pajaro_arr.shape)
print("tipo de dato",type(pajaro_arr))
#sitk.Show(ima) #help(sitk.Show) #This function requires that Fiji ( https://f
iji.sc ) or ImageJ ( http://rsb.info.nih.gov/ij/)
#be properly installed for Mac and Windows
import matplotlib.pyplot as plt
#se grafica usando el TIPO ARREGLO (3D)que en este caso se consiguió con la fu
nción de SimpleITk
#ima_arr = sitk.GetArrayFromImage(ima)
plt.imshow(pajaro_arr,cmap='gray')
plt.show()
```

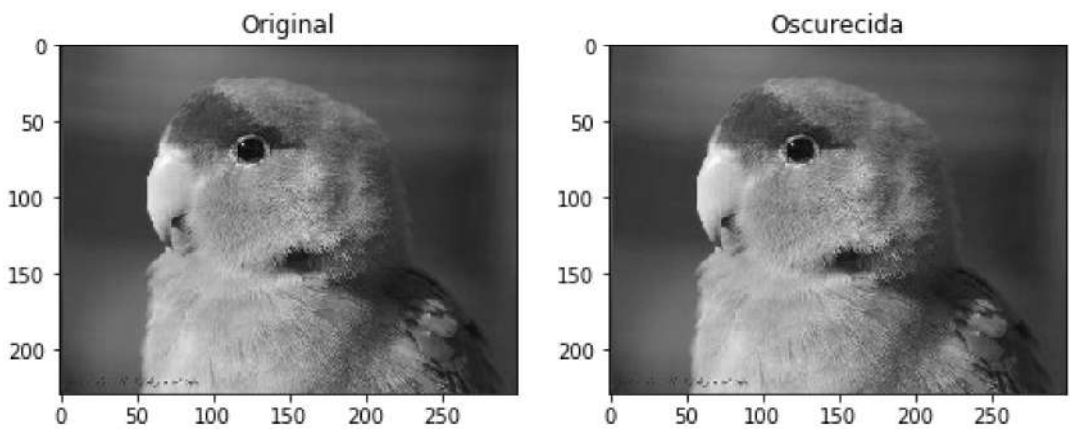
```
es una imagen de tamaño (229, 300)
tipo de dato <class 'numpy.ndarray'>
```

In [49]:
```
PajaroOscurecido= pajaro_arr/3
print("Intensidad Píxel (0,0) original: ",pajaro_arr[0,0],"Su tipo de dato es"
, type(pajaro_arr[0,0]))
print("Intensidad Píxel (0,0) luego del oscurecimiento: ",PajaroOscurecido[0,0
],"\nTipo de dato ", type(PajaroOscurecido[0,0]),
       "\nLa nueva matriz imagen es de tipo:", type(PajaroOscurecido))
print()

PajaroOscurecido=PajaroOscurecido.astype('uint8') ##g=g.astype('uint8') #conve
rtir el tipo de dato
print("Después de la conversión de tipo, la imagen oscurecida tiene un tipo de
dato ",type(PajaroOscurecido),
      "\nY cada elemento es de tipo ",type(PajaroOscurecido[0,0]))
#print()
print('Máximo de la imagen original= ',pajaro_arr.max())
print()
print('Máximo de la imagen oscurecida= ',PajaroOscurecido.max())
```

```
Intensidad Píxel (0,0) original:  54 Su tipo de dato es <class 'numpy.uint8'>
Intensidad Píxel (0,0) luego del oscurecimiento:  18.0
Tipo de dato  <class 'numpy.float64'>
La nueva matriz imagen es de tipo: <class 'numpy.ndarray'>

Después de la conversión de tipo, la imagen oscurecida tiene un tipo de dato
<class 'numpy.ndarray'>
Y cada elemento es de tipo  <class 'numpy.uint8'>
Máximo de la imagen original=  255

Máximo de la imagen oscurecida=  85
```

In [50]:
```
fig = plt.figure()
fig.set_figheight(9)
fig.set_figwidth(9)
a = fig.add_subplot(1, 2, 1)
imgplot = plt.imshow(pajaro_arr,cmap='gray')
a.set_title('Original')
#plt.colorbar( orientation='horizontal')
a = fig.add_subplot(1, 2, 2)
imgplot = plt.imshow(PajaroOscurecido,cmap='gray')
#imgplot.set_clim(0, 255)
a.set_title('Oscurecida')
#plt.colorbar( orientation='horizontal')
print('QUE PASÓ?????')
```

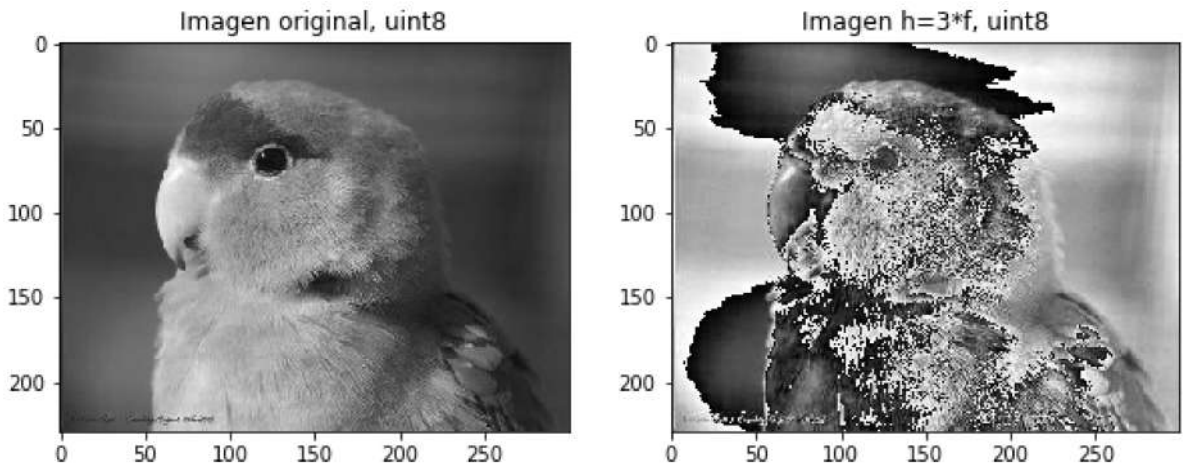
```
QUE PASÓ?????
```




In [51]:
```
#continuación- Muestra imágenes
fig, axs = plt.subplots(nrows=1, ncols=3, sharex=True)
fig.set_figheight(10)
fig.set_figwidth(15)
im=axs[0].imshow(pajaro_arr,cmap="gray")#,vmin=0, vmax=255)
axs[0].set_title("Imagen original, nivel de gris r")
axs[0].axis('off')
axs[1].imshow(PajaroOscurecido,cmap="gray")
axs[1].set_title("Imagen transf: T(r)=r/2, OJO visualización")
axs[1].axis('off')
axs[2].imshow(PajaroOscurecido,cmap="gray",vmin=0, vmax=255)
axs[2].set_title("Imagen T(r)=r/2, visualización igual rango")
axs[2].axis('off')
plt.show()
```

In [52]:
```
h=3*pajaro_arr[:,:] #A cada r le asigna T(r)=3r
#hf=pajaro_arr.astype('float')*3 #T(r)=3r
# Al ser uint 8 cuando se supera el maximo comienza de cero
fig,axs = plt.subplots(nrows=1, ncols=2, sharex=True)

fig.set_figheight(7)
fig.set_figwidth(10)

print(h.max())

axs[0].imshow(pajaro_arr,cmap="gray",vmin=0, vmax=255)
axs[0].set_title("Imagen original, uint8")
axs[1].imshow(h,cmap="gray",vmin=0, vmax=255)
axs[1].set_title("Imagen h=3*f, uint8")
plt.show()
```

```
255
```

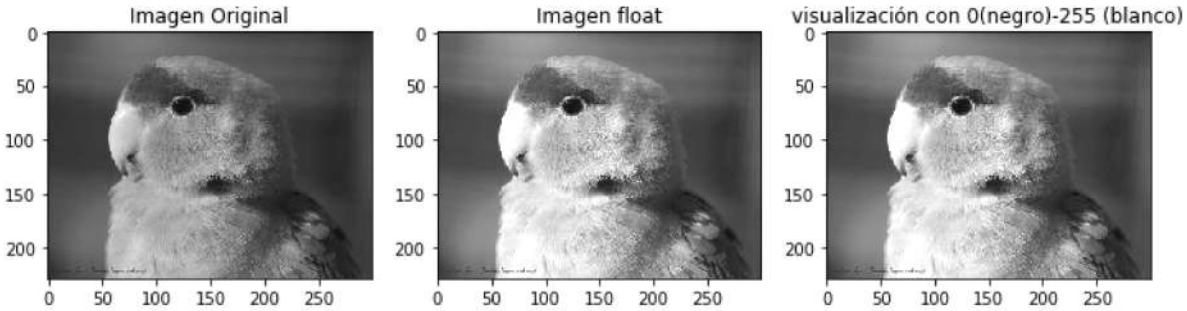



In [53]:
```
print("Con tipo flotante")
print("Convertimos la original en flotante antes de multiplicar por 3")
print("Tomamos una intensidad alta que multiplicada por 3 va a exceder el inte
rvalo [0,255]")
hf=pajaro_arr.astype('float')*3 #T(r)=3r
print("Original r=f(50,170)=",pajaro_arr[50,170]) #
print("Transformada T(r)=3*r, h(50,170)=",hf[50,170], type(hf[50,170]))
print("Este valor excede el rango 0 -255")
```

```
Con tipo flotante
Convertimos la original en flotante antes de multiplicar por 3
Tomamos una intensidad alta que multiplicada por 3 va a exceder el intervalo
[0,255]
Original r=f(50,170)= 111
Transformada T(r)=3*r, h(50,170)= 333.0 <class 'numpy.float64'>
Este valor excede el rango 0 -255
```

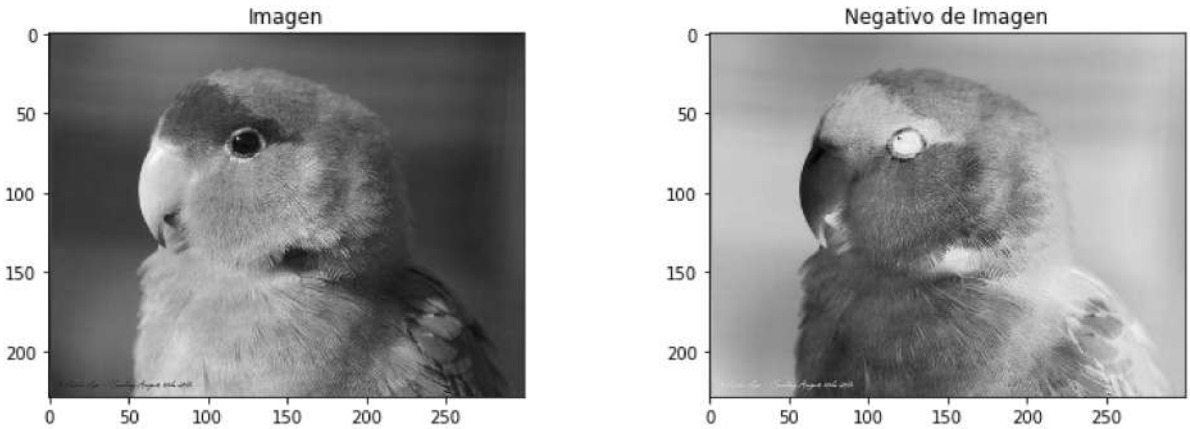
In [54]:
```
#hiluminado=f+100
hiluminado=pajaro_arr*3
hiluminadomax=hiluminado.max()
hiluminadomin=hiluminado.min()
print("Cálculo automático tipo de dato: ", type(hiluminadomax))
print("Maximo transformada",hiluminadomax)
print("Minimo transformada",hiluminadomin)
print()

hil=pajaro_arr.astype('float')
#hiluminado2=hil*2
hiluminado2=hil+60



hiluminadomax2=hiluminado2.max()
hiluminadomin2=hiluminado2.min()
print("Cálculo pasando a float-Tipo de dato", type(hiluminadomax2))
print("Maximo transformada",hiluminadomax2)
print("Minimo transformada",hiluminadomin2)

print()
print("Truncando el resultado obtenido ACA")
#
result = np.where(hiluminado2>255)
hiluminado2[result]=255
print(type(hiluminado2))


Aclarado=hiluminado2.astype('uint8')
#
hiluminadomax2=hiluminado2.max()
hiluminadomin2=hiluminado2.min()
print("Máximo transformada",hiluminadomax2)
print("Mínimo transformada truncando",hiluminadomin2)

print("Tipo de dato", type(hiluminadomax2))
print("Cambio tipo de dato a uint8 ")
#hiluminado2=hiluminado2.astype('uint8')
print(hiluminado.min())
import matplotlib.pyplot as plt
%matplotlib inline

fig,axs = plt.subplots(nrows=1, ncols=3, sharex=True)

fig.set_figheight(12)
fig.set_figwidth(12)

axs[0].imshow(pajaro_arr,cmap="gray",vmin=0, vmax=255)
axs[0].set_title("Imagen Original")
axs[1].imshow(hiluminado2,cmap="gray")#,vmin=0, vmax=255)
axs[1].set_title("Imagen float")
axs[2].imshow(Aclarado,cmap="gray")#,vmin=0, vmax=255)
axs[2].set_title("visualización con 0(negro)-255 (blanco)")
plt.show()
```

```
Cálculo automático tipo de dato:  <class 'numpy.uint8'>
Maximo transformada 255
Minimo transformada 1

Cálculo pasando a float-Tipo de dato <class 'numpy.float64'>
Maximo transformada 315.0
Minimo transformada 62.0

Truncando el resultado obtenido ACA
<class 'numpy.ndarray'>
Máximo transformada 255.0
Mínimo transformada truncando 62.0
Tipo de dato <class 'numpy.float64'>
Cambio tipo de dato a uint8
1
```



In [55]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt

img = cv2.imread('pajaromonocromo.jpg', cv2.IMREAD_GRAYSCALE)#guarda la imagen
en grises (monocroma 2D)
print(img.shape,type(img))
img_negativo=255-img

fig, axs = plt.subplots(nrows=1, ncols=2, sharex=True)
fig.set_figheight(4)
fig.set_figwidth(13)
im=axs[0].imshow(img, cmap='gray',vmin=0 ,vmax=255)
axs[0].set_title('Imagen')
axs[1].imshow(img_negativo, cmap='gray',vmin=0 ,vmax=255)
axs[1].set_title('Negativo de Imagen')

plt.show()
```

```
(229, 300) <class 'numpy.ndarray'>
```

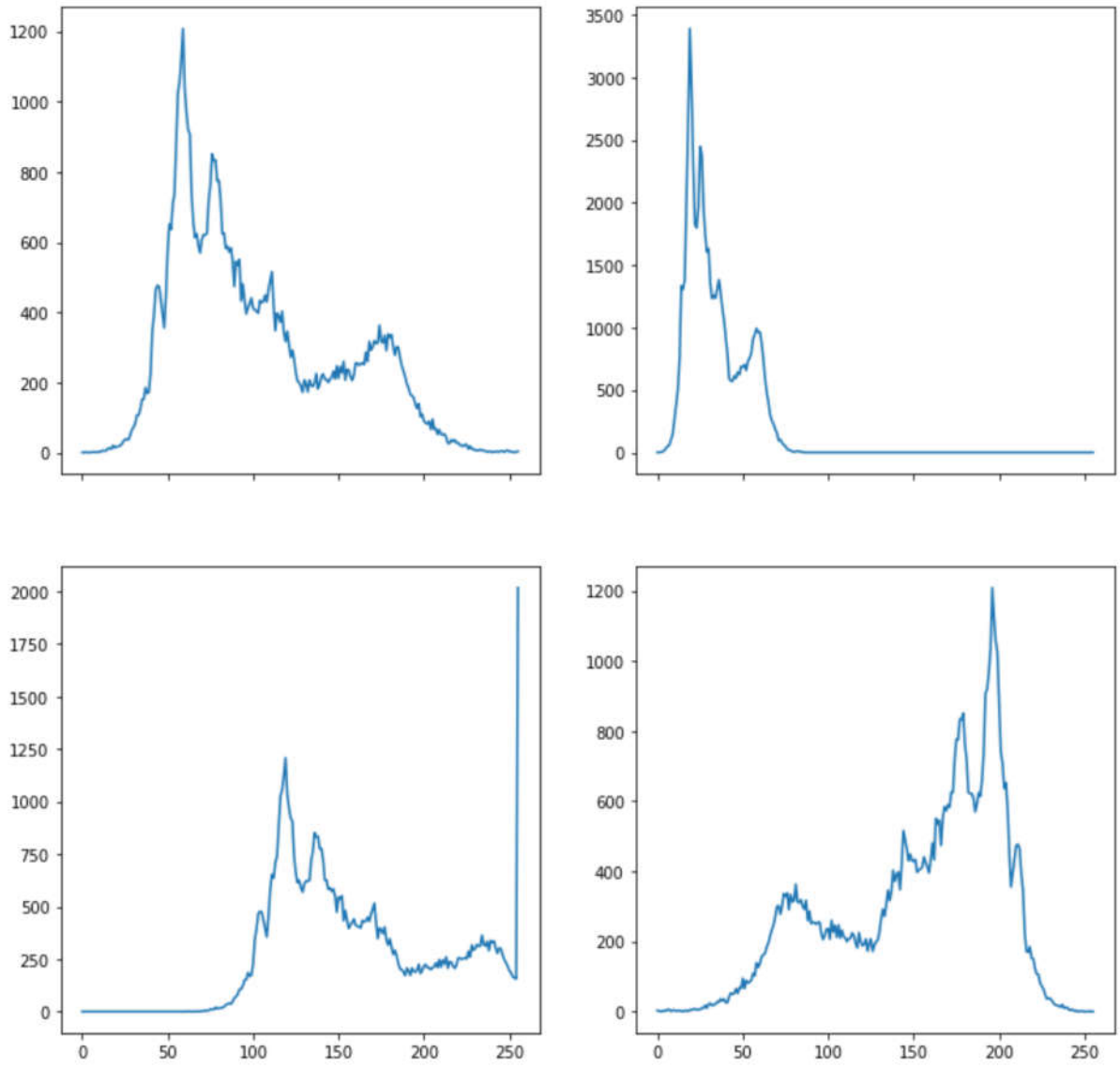



## 8) Obtener el valor de gris más repetido en cada imagen.

## listo

In [61]:
```
import cv2
import statistics as st
from statistics import mode

hist1 = cv2.calcHist([pajaro_arr],[0],None,[256],[0,256])
hist2 = cv2.calcHist([PajaroOscurecido],[0],None,[256],[0,256])
hist3 = cv2.calcHist([Aclarado],[0],None,[256],[0,256])
hist4 = cv2.calcHist([img_negativo],[0],None,[256],[0,256])
#A=np.array([2,5,6,7])

#print(hist1.shape,type(hist1[0,0]))

print(max(hist1))
print(max(hist2))
print(max(hist3))
print(max(hist4))
A = np.where(hist1 == hist1.max()) # Encuentra la posicion donde se encuentra
 el valor maximo en hist1
B = np.where(hist2 == hist2.max())
C = np.where(hist3 == hist3.max())
D = np.where(hist4 == hist4.max())

print('59,19,255 y 196 son los valores de grises mas repetidos para las imagen
monocroma original la oscurecida, la aclarada y la negativa respectivamente')
#A = hist1.argmin()
print(A)
print(B)
print(C)
print(D)#la imprime
#print(hist1.astype(np.int))
#print(st.mode(hist1))
#print(hist1[59])
fig, axs = plt.subplots(nrows=2, ncols=2, sharex=True)
fig.set_figheight(12)
fig.set_figwidth(12)
axs[0,0].plot(hist1)
#axs[0].set_title('Histograma de original')
#axs[0].set_xlabel('intensidad de iluminacion')
#axs[0].set_ylabel('cantidad de pixeles')
axs[0,1].plot(hist2)
#axs[1].set_title('Histograma de aclarada')
#axs[1].set_xlabel('intensidad de iluminacion')
#axs[1].set_ylabel('cantidad de pixeles')
axs[1,0].plot(hist3)

axs[1,1].plot(hist4)
#axs[2].set_title('Histograma de oscurecida')
#axs[2].set_xlabel('intensidad de iluminacion')
#axs[2].set_ylabel('cantidad de pixeles')
plt.show()
```

```
[1208.]
[3389.]
[2017.]
[1208.]
59,19,255 y 196 son los valores de grises mas repetidos para las imagen monoc
roma original la oscurecida, la aclarada y la negativa respectivamente
(array([59], dtype=int64), array([0], dtype=int64))
(array([19], dtype=int64), array([0], dtype=int64))
(array([255], dtype=int64), array([0], dtype=int64))
(array([196], dtype=int64), array([0], dtype=int64))
```



In [62]:
```
#otra forma de calcular la intensidad de gris mas rapetida
#print(pajaro_arr,pajaro_arr.shape)

import statistics as st
from statistics import mode

pajaro_arrV = pajaro_arr.flatten()
PajaroOscurecidoV = PajaroOscurecido.flatten()
AclaradoV = Aclarado.flatten()
img_negativoV = img_negativo.flatten()
print(st.mode(pajaro_arrV))
print(st.mode(PajaroOscurecidoV))
print(st.mode(AclaradoV))
print(st.mode(img_negativoV))
```

```
59
19
255
196
```

## 9) Aplicar una máscara de suavizado de 5x5 a la componente roja de una imagen a elección, un filtro de bordes a la componente verde componer una imagen a color donde se reemplace las componentes rojas y verdes modificadas.

In [63]:
```
#Filtro gaussiano
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('paisajepuente.jpg')#carga la imagen como numpy de uint8
gris = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)#Convierte en escala de grises

#n=int(input('Ingrese tamaño matriz para filtro gaussiano (número impar y posi
tivo) '))
n = 5

print(img.shape)
print(gris.shape)
print(img.max())
blur = cv2.GaussianBlur(gris,(n,n),0) #aca es donde se aplica el filtro Gaussi
ano

fig = plt.gcf()
fig.set_size_inches(18.5, 10.5)

plt.subplot(121),plt.imshow(gris,cmap='gray',vmin=0,vmax=255),plt.title('Origi
nal')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(blur,cmap='gray',vmin=0,vmax=255),plt.title('Filtr
o Gaussiano')
plt.xticks([]), plt.yticks([])
#print(type(blur[0] [0]))#verifica que devuelve uint8
plt.show()
```

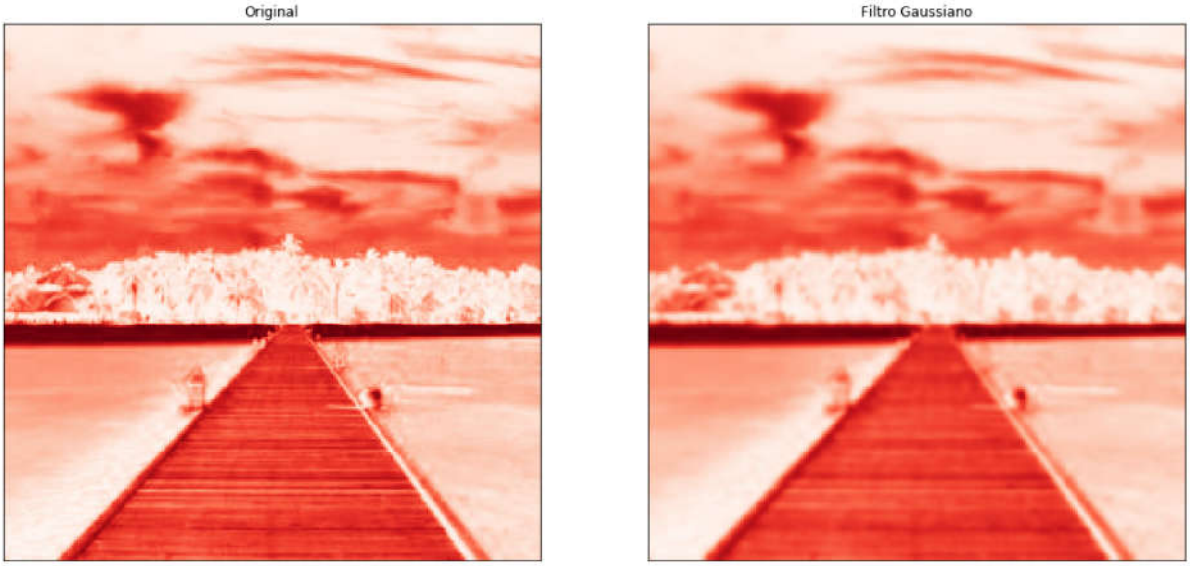
```
(256, 256, 3)
(256, 256)
255
```

In [64]:
```
# VER LO QUE ESTA EN NTB2
#Continúa el ejemplo anterior
#Mostrar las componentes verde y azul

import numpy as np
rojo = img[:,:,2]
verde = img[:,:,1]
azul = img[:,:,0]


monocroma=np.transpose(sum(np.transpose(img)/3)) #intensidad monocroma=(rojo+v
erde+azul)/3
print("dimensiones de imagen monocroma  (R+G+B)/3=" , monocroma.shape)

#para mostrar gráfico de paleta
l=np.linspace(0,ima_arr.max(),256)#vector de intensidades
l=l.astype('uint8')

tr=np.zeros((256,40),'uint8')
for i in range(40):
    tr[:,i]=l

paleta='inferno'

fig, axs = plt.subplots(nrows=2, ncols=3, sharex=True)
fig.set_figheight(10)
fig.set_figwidth(15)
axs[0,0].imshow(ima_arr)# ,cmap=paleta)
axs[0,0].set_title("imagen color RGB")
axs[0,1].imshow(monocroma ,cmap='gray')
axs[0,1].set_title("imagen monocroma traspuesta (R+G+B)/3")
axs[0,2].imshow(tr,cmap='gray')
axs[0,2].set_title(" referencia (paleta)")
axs[1,0].imshow(rojo ,cmap='Reds')
axs[1,0].set_title("componente roja:R=Reds")
axs[1,1].imshow(verde ,cmap='Greens')
axs[1,1].set_title("componente verde:G=Greens")
axs[1,2].imshow(azul ,cmap='Blues')
axs[1,2].set_title("componente azul:B=Blues")

plt.show()
```

```
dimensiones de imagen monocroma  (R+G+B)/3= (256, 256)

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-64-62800ea9773c> in <module>
     13
     14 #para mostrar gráfico de paleta
---> 15 l=np.linspace(0,ima_arr.max(),256)#vector de intensidades
     16 l=l.astype('uint8')
     17

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

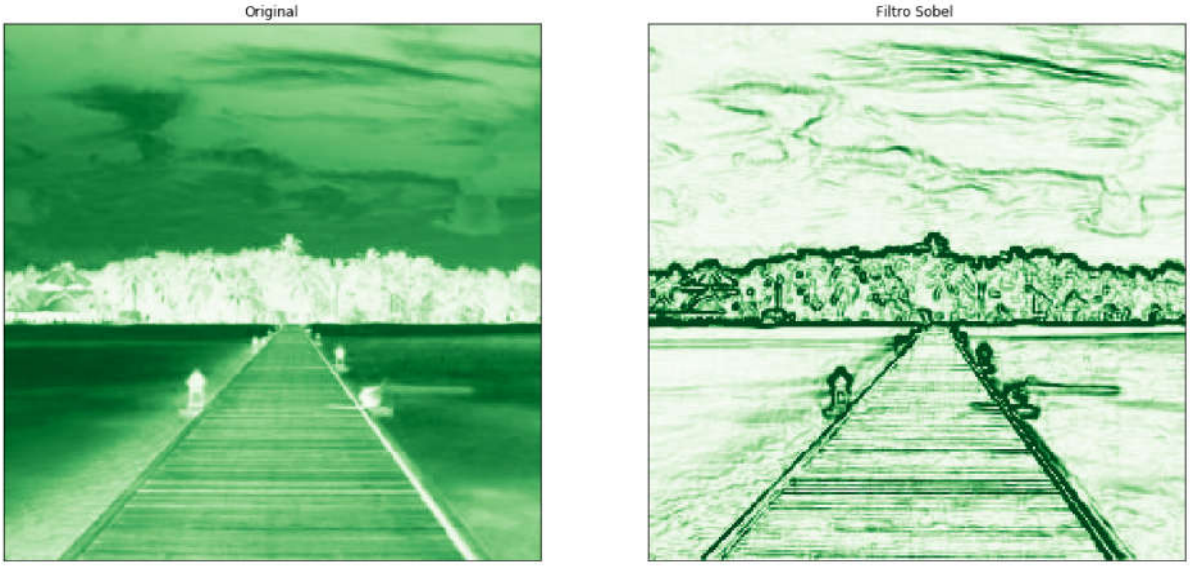
In [65]:
```
n = 5

print(img.shape)
print(rojo.shape)
print(img.max())

blur = cv2.GaussianBlur(rojo,(n,n),0) #aca es donde se aplica el filtro Gaussi
ano

fig = plt.gcf()
fig.set_size_inches(18.5, 10.5)

plt.subplot(121),plt.imshow(rojo,cmap='Reds',vmin=0,vmax=255),plt.title('Origi
nal')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(blur,cmap='Reds',vmin=0,vmax=255),plt.title('Filtr
o Gaussiano')
plt.xticks([]), plt.yticks([])
#print(type(blur[0] [0]))#verifica que devuelve uint8
plt.show()
```

```
(256, 256, 3)
(256, 256)
255
```



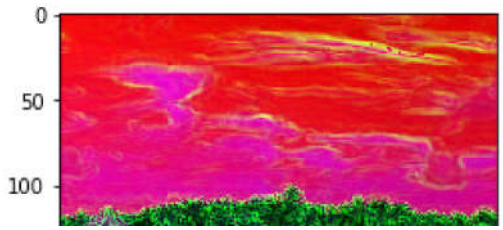
In [66]:
```
sobelx = cv2.Sobel(verde,cv2.CV_64F,1,0,ksize=3)#1 0 indica sentido x, ksize:
 tamaño kernel
sobely = cv2.Sobel(verde,cv2.CV_64F,0,1,ksize=3)#0 1 indica sentido y, ksize:
 tamaño kernel

verdeB=abs(sobelx)+abs(sobely)


fig = plt.gcf()
fig.set_size_inches(18.5, 10.5)

plt.subplot(121),plt.imshow(verde,cmap='Greens',vmin=0,vmax=255),plt.title('Or
iginal')
plt.xticks([]), plt.yticks([])
plt.subplot(122),plt.imshow(abs(sobelx)+abs(sobely),cmap='Greens',vmin=0,vmax=
255),plt.title('Filtro Sobel ')
plt.xticks([]), plt.yticks([])
```

Out[66]:
```
(([], <a list of 0 Text xticklabel objects>),
 ([], <a list of 0 Text yticklabel objects>))
```



In [67]:
```
compuesta = np.ones((256,256,3),'uint8')

compuesta[:,:,0] = azul
compuesta[:,:,1] = verdeB
compuesta[:,:,2] = blur

plt.imshow(compuesta)
```

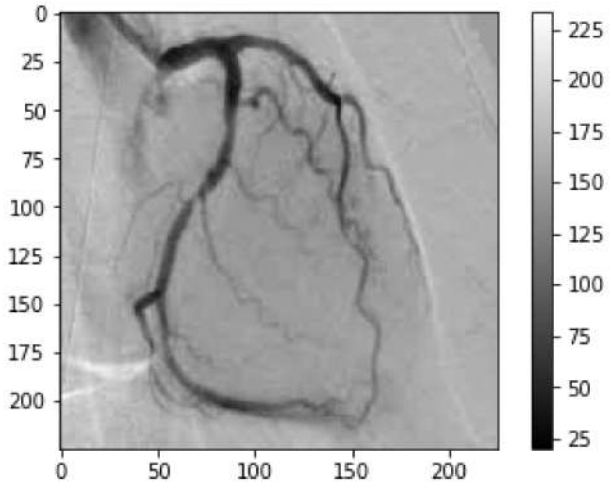
Out[67]: <matplotlib.image.AxesImage at 0x2b8052626a0>

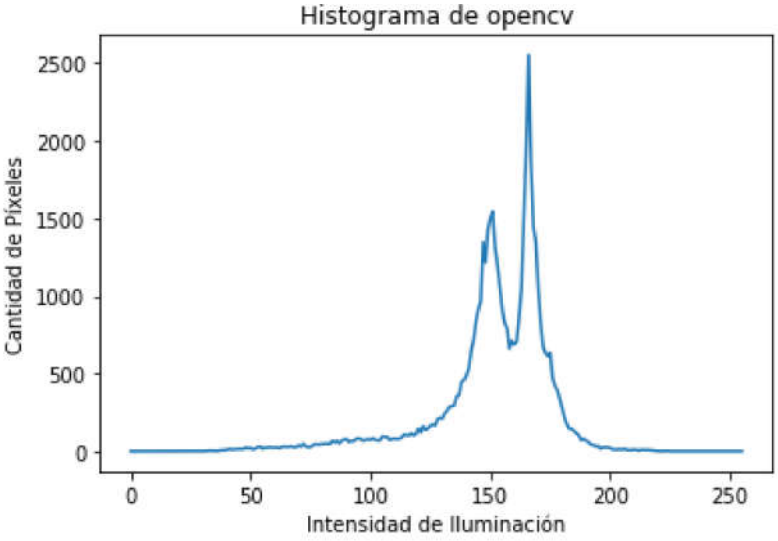



**10) Binarice una imagen a elección buscando demarcar uno o varios objetos en esta (puede ser eligiendo un umbral a partir del histograma de una imagen monocroma). Al resultado aplicarle 2 o 3 filtros morfológicos en el orden que crea adecuado para mejorar la segmentación con elemento estructurante creado por usted, según crea adecuado para el problema (justifique).**

In [68]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('corazon.jpg',0)#El 0 indica que devuelve imagen monocroma di
rectamente
#imgg = cv2.imread('lenna.jpg', cv2.IMREAD_GRAYSCALE)#equivalente al anterior
#print(abs(img-imgg).max())
print('tamaño de la imagen monocroma:', img.shape)

print(img.max())

#plt.imshow(img)
plt.imshow(img, cmap='gray')
plt.colorbar()
plt.show()

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 de opencv")
plt.show()
```

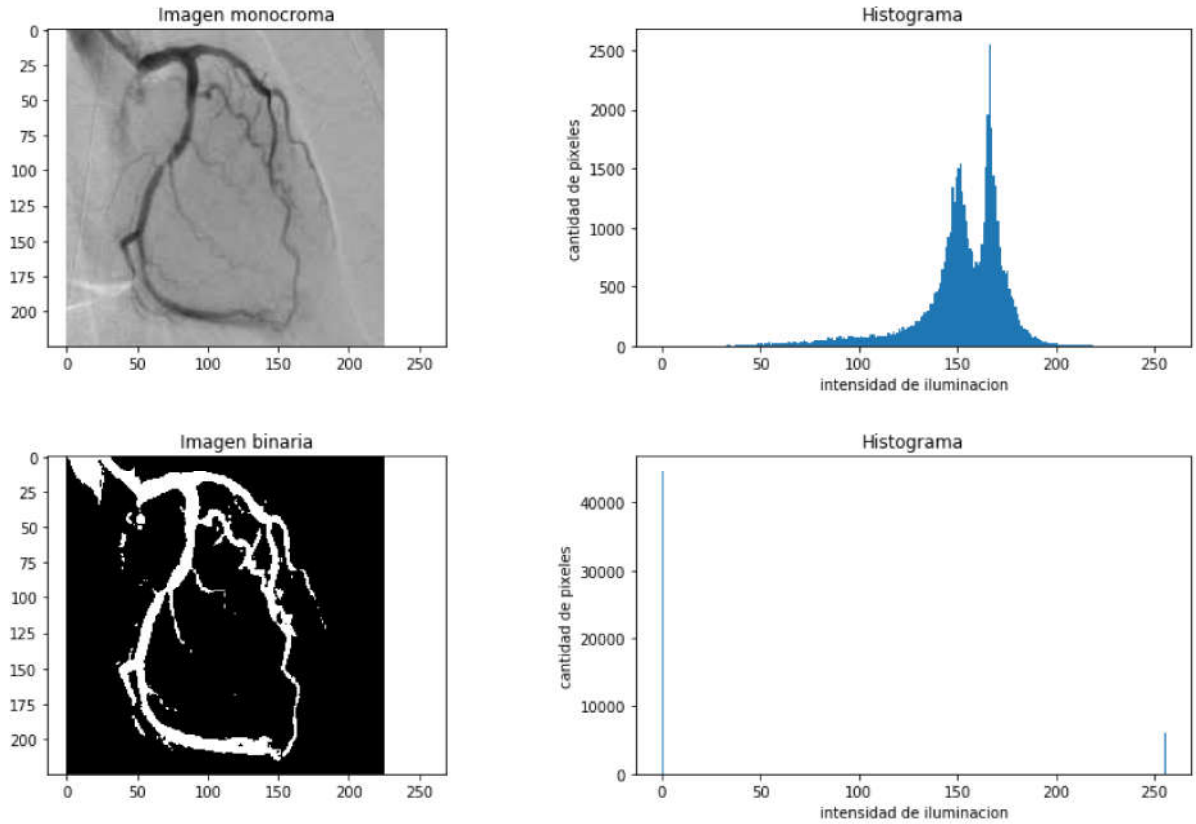
```
tamaño de la imagen monocroma: (225, 225)
233
```



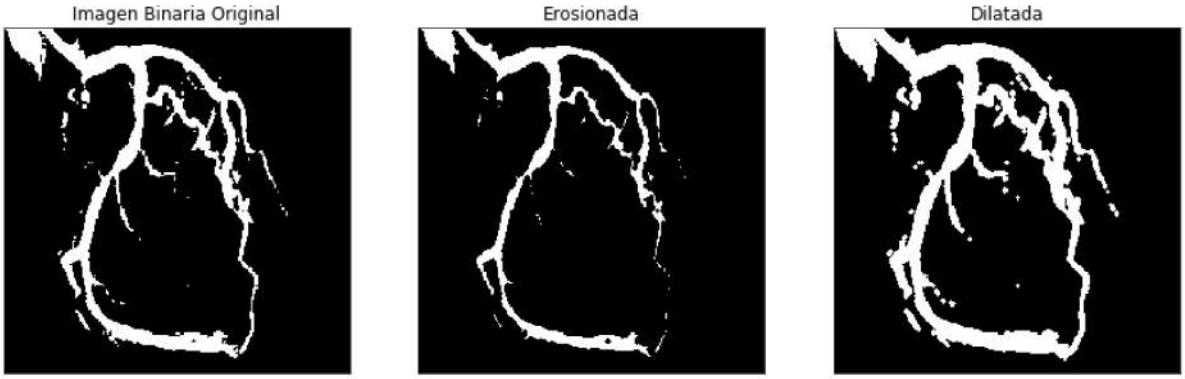
In [69]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread('corazon.jpg',0) #imagen monocroma
#img = img.astype('uint8')
print(img.max())

#sobelx = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=3)#1 0 indica sentido x, ksize: t
amaño kernel
#sobely = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=3)#0 1 indica sentido y, ksize: t
amaño kernel

#imgb = abs(sobelx)+abs(sobely)
#print(img[24,200])

umbral=140 #arbitrario

ret3,binaria3 = cv2.threshold(img,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)
# OTSU elige el umbral

binaria=(img>=umbral)
#print(binaria)

binaria=255*binaria.astype('uint8')
print(type(binaria),binaria.shape)

##Primer par de gráficos:
fig, axs = plt.subplots(nrows=1, ncols=2, sharex=True)
fig.set_figheight(4)
fig.set_figwidth(15)
axs[0].imshow(img, cmap='gray',vmin=0 ,vmax=255)
axs[0].set_title('Imagen monocroma')
axs[1].hist(img[:,:].ravel(), 256,[0,256] )
axs[1].set_title('Histograma')
axs[1].set_xlabel('intensidad de iluminacion')
axs[1].set_ylabel('cantidad de pixeles')

##Segundo par de gráficos:
fig, axs = plt.subplots(nrows=1, ncols=2, sharex=True)
fig.set_figheight(4)
fig.set_figwidth(15)
axs[0].imshow(binaria3, cmap='gray',vmin=0 ,vmax=255)
axs[0].set_title('Imagen binaria')
axs[1].hist(binaria3[:,:].ravel(), 256,[0,256] )
axs[1].set_title('Histograma')
axs[1].set_xlabel('intensidad de iluminacion')
axs[1].set_ylabel('cantidad de pixeles')
print('umbral calculado por Otsu:', ret3)
plt.show()
```

```
233
<class 'numpy.ndarray'> (225, 225)
umbral calculado por Otsu: 130.0
```




## 11) Mostrar en una misma ventana, la imagen original, la binaria y la resultante de aplicarle los filtros. Comente si tuvo buenos resultados.

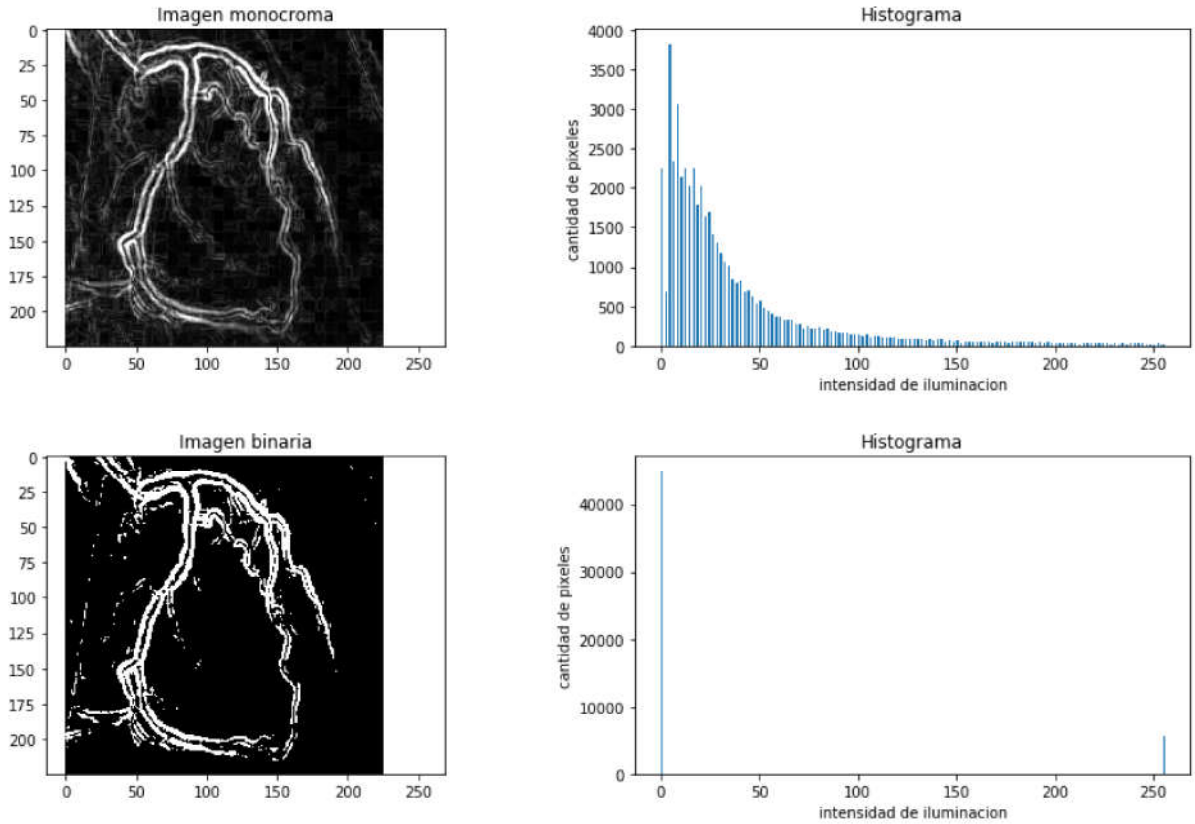
In [70]:
```
#kernel = np.ones((3,3),np.uint8)#creamos el elemento estructurante
kernel = np.array([[0,1,0],[1,1,1],[0,1,0]])
kernel = kernel.astype('uint8')
bin_erosionada = cv2.erode(binaria3,kernel,iterations = 1)# erosionamos la ima
gen binaria
bin_dilatada = cv2.dilate(binaria3,kernel,iterations = 1) #dilatamos la imagen
binaria

#graficamos: imagen binaria, erosionada y dilatada
fig = plt.gcf()
fig.set_size_inches(14.5, 10.5)
plt.subplot(1,3,1), plt.imshow(binaria3,cmap='gray'),plt.title('Imagen Binaria
Original'),plt.xticks([]), plt.yticks([])
plt.subplot(1,3,2),plt.imshow(bin_erosionada,cmap='gray'),plt.title('Erosionad
a'),plt.xticks([]), plt.yticks([])#dilatada
plt.subplot(1,3,3),plt.imshow(bin_dilatada,cmap='gray'),plt.title('Dilatada'),
plt.xticks([]), plt.yticks([])#erosionada
plt.show()
```



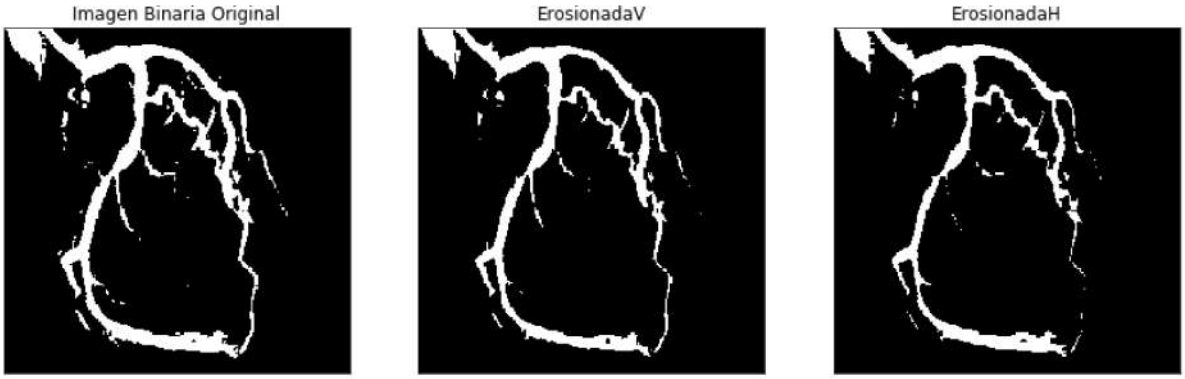

## 12)Encuentre bordes o el contorno del objeto u objetosen la imagen binaria utilizando operaciones morfológicas.

In [71]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
img = cv2.imread('corazon.jpg',0) #imagen monocroma
#img = img.astype('uint8')
print(img.max())

sobelx = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=3)#1 0 indica sentido x, ksize: ta
maño kernel
sobely = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=3)#0 1 indica sentido y, ksize: ta
maño kernel

imgb = abs(sobelx)+abs(sobely)
#print(img[24,200])

umbral=90 #arbitrario

#ret3,binaria3 = cv2.threshold(img,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTS
U)# OTSU elige el umbral

binaria=(imgb>=umbral) # Devuelve una matriz de unos y ceros o tipos de datos
 booleanos (T and F)
#print(binaria)

binaria=255*binaria.astype('uint8')
print(type(binaria),binaria.shape)

##Primer par de gráficos:
fig, axs = plt.subplots(nrows=1, ncols=2, sharex=True)
fig.set_figheight(4)
fig.set_figwidth(15)
axs[0].imshow(imgb, cmap='gray',vmin=0 ,vmax=255)
axs[0].set_title('Imagen monocroma')
axs[1].hist(imgb[:,:].ravel(), 256,[0,256] )
axs[1].set_title('Histograma')
axs[1].set_xlabel('intensidad de iluminacion')
axs[1].set_ylabel('cantidad de pixeles')

##Segundo par de gráficos:
fig, axs = plt.subplots(nrows=1, ncols=2, sharex=True)
fig.set_figheight(4)
fig.set_figwidth(15)
axs[0].imshow(binaria, cmap='gray',vmin=0 ,vmax=255)
axs[0].set_title('Imagen binaria')
axs[1].hist(binaria[:,:].ravel(), 256,[0,256] )
axs[1].set_title('Histograma')
axs[1].set_xlabel('intensidad de iluminacion')
axs[1].set_ylabel('cantidad de pixeles')
print('umbral calculado por Otsu:', ret3)
plt.show()
```

```
233
<class 'numpy.ndarray'> (225, 225)
umbral calculado por Otsu: 130.0
```

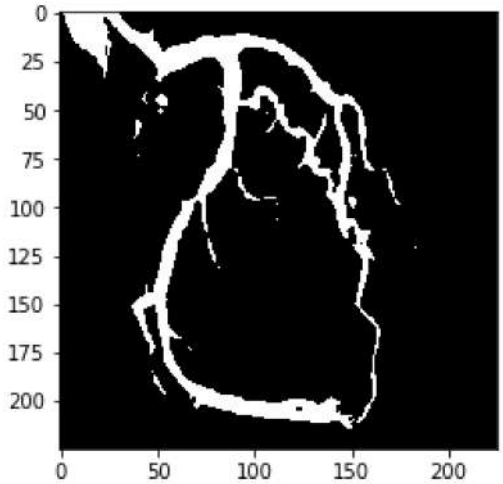


In [72]:
```
kernel2 = np.array([[0,0,0],[1,1,1],[0,0,0]])
kernel1 = np.array([[0,1,0],[0,1,0],[0,1,0]])
kernel1 = kernel1.astype('uint8')
kernel2 = kernel2.astype('uint8')
bin_erosionadaV = cv2.erode(binaria3,kernel1,iterations = 1)# erosionamos la i
magen binaria
bin_erosionadaH = cv2.erode(binaria3,kernel2,iterations = 1) #dilatamos la ima
gen binaria

#graficamos: imagen binaria, erosionada y dilatada
fig = plt.gcf()
fig.set_size_inches(14.5, 10.5)
plt.subplot(1,3,1), plt.imshow(binaria3,cmap='gray'),plt.title('Imagen Binaria
Original'),plt.xticks([]), plt.yticks([])
plt.subplot(1,3,2),plt.imshow(bin_erosionadaV,cmap='gray'),plt.title('Erosiona
daV'),plt.xticks([]), plt.yticks([])#dilatada
plt.subplot(1,3,3),plt.imshow(bin_erosionadaH,cmap='gray'),plt.title('Erosiona
daH'),plt.xticks([]), plt.yticks([])#erosionada
plt.show()
```




In [73]:
```
erisionadaT = bin_erosionadaV + bin_erosionadaH
plt.imshow(erisionadaT,cmap='gray')
```

Out[73]: <matplotlib.image.AxesImage at 0x2b80835dc18>



In [74]:
```
print(erisionadaT.max())
```

```
255
```

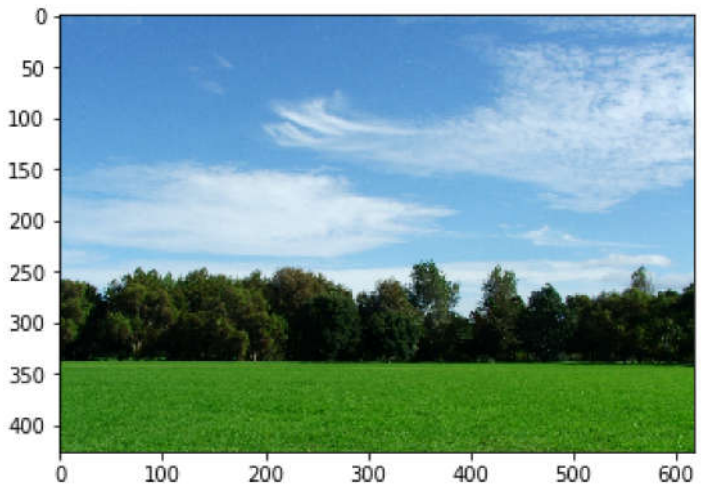
## 13)Cargar una imagen a color (a elección), que a su criterio tenga entre 2 y 7 clases en ella.

- a) Explique coloquialmente la cantidad y el criterio visual que utilizó para identificarlas
- b) Agregue una o dos características (capas, bandas, etc, pueden ser bordes de alguna componente) más a la imagen que considere puedan ser relevantes para la clasificación.
- c) Construya una muestra de entrenamiento de la imagen utilizando todas las características
- d) Realice un análisis exploratorio con gráficos de la muestra de entrenamiento discriminando con diferente color cada clase.
- e) Clasifíquela utilizando la muestra de entrenamiento y K-NN (k vecinos más cercanos). O con el método que guste.
- f) Generar una máscara o elemento estructural, y realice con éste una mejora de la clasificación obtenida en el ejercicio anterior utilizando un filtro de moda. Mirando el resultado final decida si considera que la clasificación fue buena y si la cantidad de clases elegida fue la adecuada.

In [83]:
```
import cv2
satelital=cv2.imread('arboles.jpg')
satelital= cv2.resize(satelital,(618,426))
cv2.imshow('pulse una tecla aquí',satelital)
cv2.waitKey() #espera pulsado de tecla.
cv2.destroyAllWindows()
```

In [84]:
```
from matplotlib import pyplot as plt

plt.imshow(satelital[:,:,[2,1,0]])
print(satelital.shape)
plt.show()
```

```
(426, 618, 3)
```



In [85]:
```
import numpy as np
dims=satelital.shape
cant_filas=dims[0]
cant_columnas=dims[1]
cant_pixeles=cant_filas*cant_columnas
data0=np.zeros([cant_pixeles,4],dtype='int')
for f in range(cant_filas):
    for c in range(cant_columnas):
        nro=f*cant_columnas+c
        data0[nro,0]=nro
        data0[nro,1:4]=satelital[f,c,:]

import pandas as pd
df = pd.DataFrame(data0, columns=["pixel_id","azul","verde","rojo"])

print(df.shape)
print('Base de pixeles:')
df.head(9)
#print(satelital[0,0,1])
```

```
(263268, 4)
Base de pixeles:
```

Out[85]:

| | pixel_id | azul | verde | rojo |
|---|---|---|---|---|
| **0** | 0 | 197 | 135 | 71 |
| **1** | 1 | 195 | 132 | 72 |
| **2** | 2 | 198 | 134 | 76 |
| **3** | 3 | 196 | 132 | 74 |
| **4** | 4 | 201 | 137 | 77 |
| **5** | 5 | 200 | 134 | 74 |
| **6** | 6 | 197 | 130 | 71 |
| **7** | 7 | 202 | 134 | 75 |
| **8** | 8 | 202 | 133 | 75 |

In [86]:
```
showCrosshair = False
fromCenter = False


roi1 = cv2.selectROI("delimite la ROI+ent+tecla", satelital, fromCenter, showC
rosshair)#ROI:region of interest

cv2.waitKey() #espera pulsado de tecla.
print('ROI1:',roi1)
print('(columna, fila, anchocol, anchofila)')
cv2.destroyAllWindows()
# columna inicial,fila inicial,ancho columna,ancho fila del roi
```

```
ROI1: (49, 16, 411, 158)
(columna, fila, anchocol, anchofila)
```

In [87]:
```
showCrosshair = False
fromCenter = False
roi2 = cv2.selectROI("delimite la ROI+ent+tecla", satelital, fromCenter, showC
rosshair)
cv2.waitKey() #espera pulsado de tecla.
print('ROI2:',roi2)
print('(columna, fila, anchocol, anchofila)')
cv2.destroyAllWindows()
```

```
ROI2: (76, 257, 193, 81)
(columna, fila, anchocol, anchofila)
```

In [88]:
```
showCrosshair = False
fromCenter = False
roi3 = cv2.selectROI("delimite la ROI+ent+tecla", satelital, fromCenter, showC
rosshair)
cv2.waitKey() #espera pulsado de tecla.
print('ROI3:',roi3)
print('(columna, fila, anchocol, anchofila)')
cv2.destroyAllWindows()
```

```
ROI3: (28, 346, 524, 72)
(columna, fila, anchocol, anchofila)
```

In [89]:
```
rois=[roi1,roi2,roi3]
```

In [90]:
```
import numpy as np

bordes = cv2.Sobel(satelital,cv2.CV_64F,1,1,ksize=5)#capa de bordes ()

satelital2=np.zeros((bordes.shape[0],bordes.shape[1],6)) # defino una imagen c
on 6 canales

satelital2[:,:,0:3]=satelital #3 primeros canales imagen original
satelital2[:,:,3:6]=bordes # 3 últimos canales con bordes
print(satelital2.shape)
```

```
(426, 618, 6)
```

In [91]:
```
import numpy as np
cantidad=0
for i in range(3):
    cantidad=cantidad+(rois[i][2]*rois[i][3])
print (cantidad)
data=np.zeros([cantidad,6])
target=np.zeros(cantidad)
cuenta=0
for i in range(3):
    roi=rois[i]

    for fil in range (roi[1],roi[1]+roi[3]):
        for col in range (roi[0],roi[0]+roi[2]):
            data[cuenta,:]=satelital2[fil,col,:]
            target[cuenta]=i
            cuenta=cuenta+1

print(cuenta)
import pandas as pd
df = pd.DataFrame(data, columns=["azul","verde","rojo","bordes_azul","bordes_v
erde","bordes_rojo"])
print(df.shape) # la cantida de pixeles es la suma de los pixeles seleccionado
s cuando se recorta suelo,fondo,cultivo
print('target=',target, '; dimensión target=',target.shape)
df.head(10)
```

```
118299
118299
(118299, 6)
target= [0. 0. 0. ... 2. 2. 2.] ; dimensión target= (118299,)
```

Out[91]:

| | azul | verde | rojo | bordes_azul | bordes_verde | bordes_rojo |
|---|---|---|---|---|---|---|
| **0** | 200.0 | 135.0 | 74.0 | 0.0 | -15.0 | 19.0 |
| **1** | 200.0 | 135.0 | 74.0 | -13.0 | 23.0 | -41.0 |
| **2** | 202.0 | 134.0 | 75.0 | -14.0 | 19.0 | -79.0 |
| **3** | 197.0 | 135.0 | 74.0 | -23.0 | -22.0 | -53.0 |
| **4** | 200.0 | 137.0 | 68.0 | -8.0 | -14.0 | 44.0 |
| **5** | 204.0 | 137.0 | 72.0 | 7.0 | 6.0 | 85.0 |
| **6** | 200.0 | 133.0 | 75.0 | -12.0 | -18.0 | 4.0 |
| **7** | 202.0 | 137.0 | 79.0 | 20.0 | 4.0 | -43.0 |
| **8** | 199.0 | 133.0 | 74.0 | 62.0 | 44.0 | -21.0 |
| **9** | 201.0 | 135.0 | 74.0 | 22.0 | 20.0 | -17.0 |

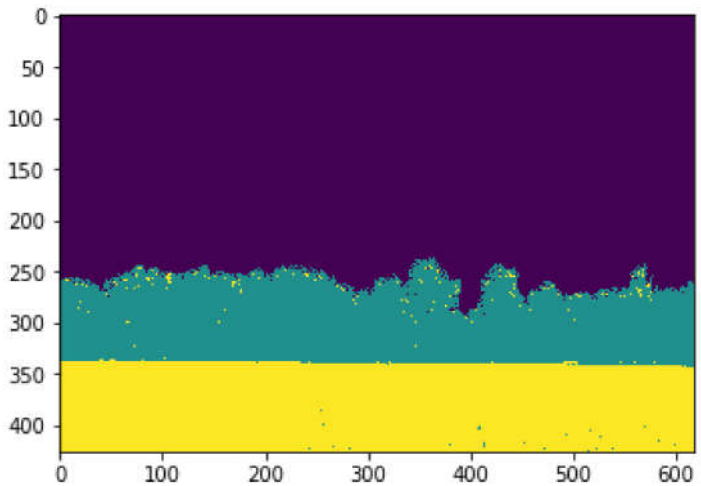
In [92]:
```
from sklearn.neighbors import KNeighborsClassifier
n_neighbors = 5

knn = KNeighborsClassifier(n_neighbors) #se define el clasificador
knn.fit(data, target)#se lo entrena/ajusta con los datos de entrenamiento
print('Presición (Accuracy) del clasificador K-NN en el conjunto de entrenamie
nto: {:.2f}'
     .format(knn.score(data, target)))
print('Específicamente el {:.2f}'
     .format(100*knn.score(data, target)),'% de los datos de entrenamiento fue
ron clasificados correctamente')
```

```
Presición (Accuracy) del clasificador K-NN en el conjunto de entrenamiento:
1.00
Específicamente el 99.89 % de los datos de entrenamiento fueron clasificados
correctamente
```

In [93]:
```
a=satelital.shape
print(a)
clasif=np.zeros(a[0:2])
print(knn.predict([[40, 80, 40, 0,0, 255]]))
for i in range(a[0]): #filas
    for j in range(a[1]): #columnas
        clasif[i,j]=knn.predict([satelital2[i,j,:]])

print('No se puede evaluar el porcentaje de acierto o Presición en el resto de
la imagen')
print('Realizaremos una valoración visual')
```

```
(426, 618, 3)
[2.]
No se puede evaluar el porcentaje de acierto o Presición en el resto de la im
agen
Realizaremos una valoración visual
```

In [94]:
```
plt.imshow(clasif)
plt.show()
```




In [ ]: