segmentation

March 6, 2022

[ ]:
```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
from scipy.ndimage import measurements,morphology
from skimage.morphology import disk
from skimage.segmentation import clear_border, mark_boundaries
from skimage.measure import label,regionprops, perimeter
from skimage.filters import roberts, sobel
from scipy import ndimage as ndi
import os
from tqdm import tqdm
import cv2
```

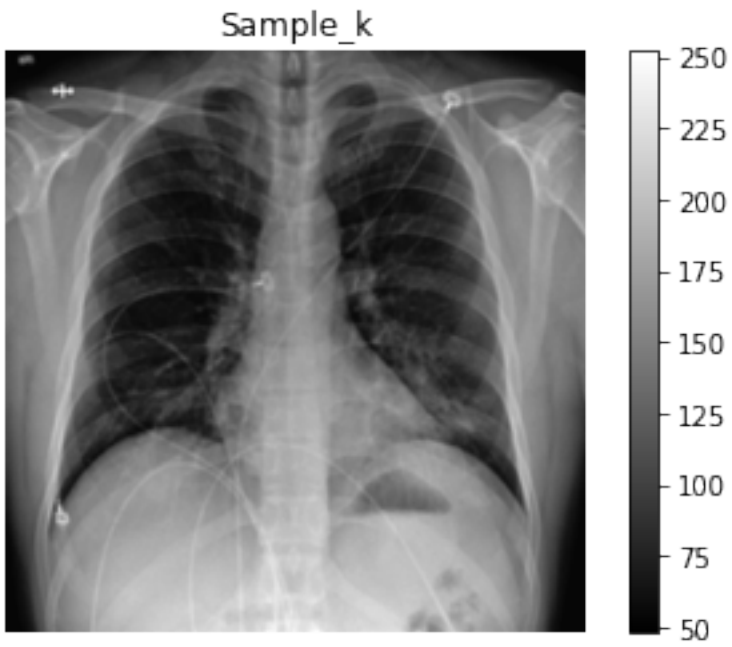
1 Segmentation Steps

[ ]:
```
im_RGB = Image.open(r'COVID-38.png')
im_RGB.size
```

[ ]:
```
(299, 299)
```

[ ]:
```
plt.imshow(im_RGB,cmap='gray')
cbar = plt.colorbar()
plt.title('Sample_k')
plt.axis('off')
```

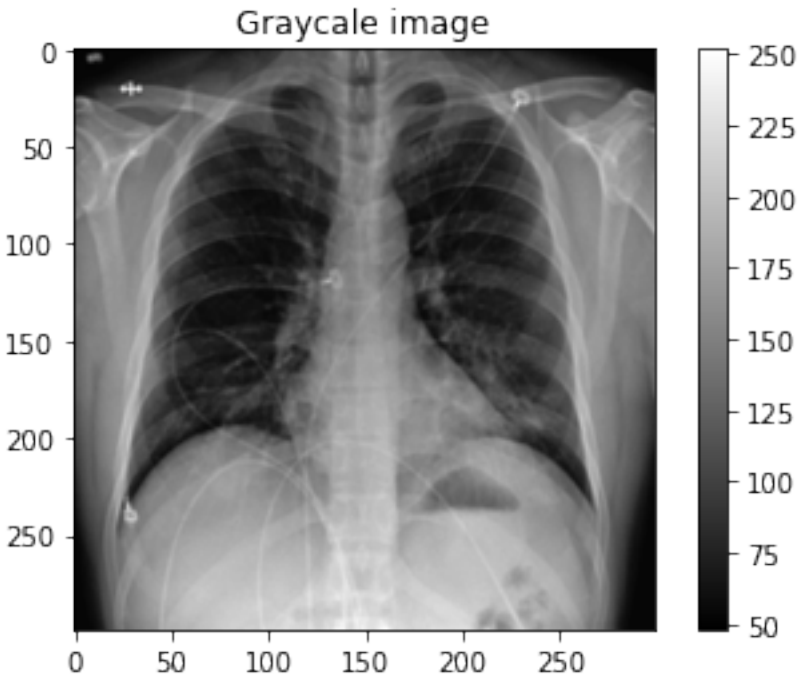
[ ]:
```
(-0.5, 298.5, 298.5, -0.5)
```



[ ]:
```
im_gray = im_RGB.convert("L")
im_gray_array = np.array(im_gray)
im_gray_array1 = im_gray_array
```

[ ]:
```
plt.imshow(im_gray_array,cmap='gray')
cbar = plt.colorbar()
plt.title('Graycale image')
```

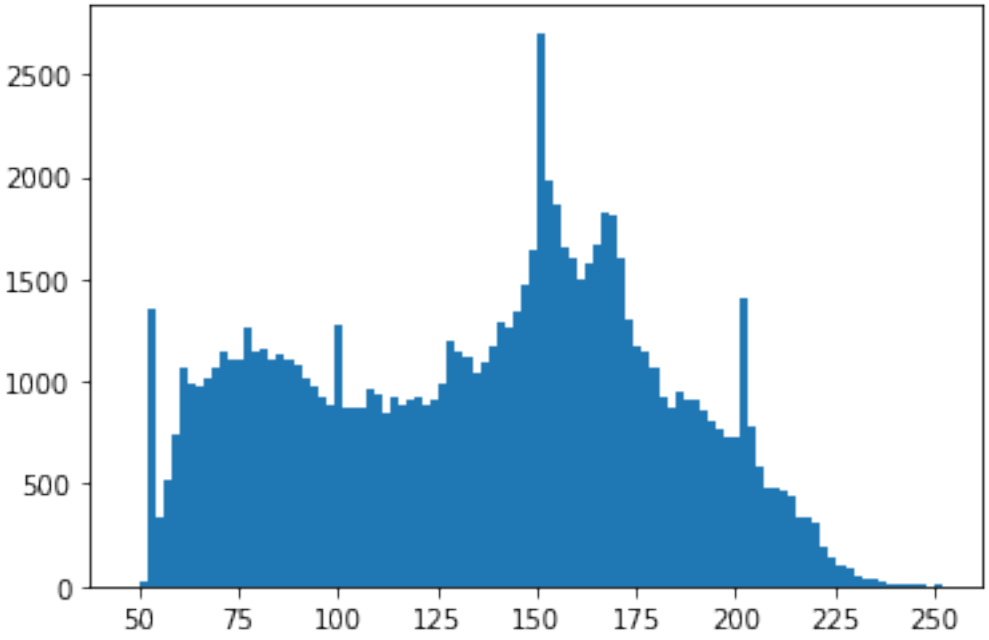
[ ]:
```
Text(0.5, 1.0, 'Graycale image')
```



[ ]:
```
plt.hist(im_gray_array.flatten(),bins=100)
```

[ ]:
```
(array([3.000e+00, 2.400e+01, 1.354e+03, 3.400e+02, 5.220e+02, 7.430e+02,
        1.063e+03, 9.910e+02, 9.790e+02, 1.021e+03, 1.072e+03, 1.144e+03,
        1.110e+03, 1.105e+03, 1.257e+03, 1.142e+03, 1.158e+03, 1.109e+03,
        1.139e+03, 1.110e+03, 1.084e+03, 1.010e+03, 9.810e+02, 9.190e+02,
        8.850e+02, 1.283e+03, 8.660e+02, 8.700e+02, 8.770e+02, 9.670e+02,
        9.330e+02, 8.410e+02, 9.260e+02, 8.810e+02, 9.120e+02, 9.200e+02,
        8.880e+02, 9.090e+02, 9.840e+02, 1.203e+03, 1.146e+03, 1.116e+03,
        1.036e+03, 1.098e+03, 1.168e+03, 1.286e+03, 1.264e+03, 1.336e+03,
        1.475e+03, 1.638e+03, 2.703e+03, 1.984e+03, 1.860e+03, 1.659e+03,
        1.601e+03, 1.497e+03, 1.571e+03, 1.672e+03, 1.823e+03, 1.811e+03,
        1.609e+03, 1.305e+03, 1.177e+03, 1.152e+03, 1.064e+03, 9.290e+02,
        8.710e+02, 9.490e+02, 9.150e+02, 9.110e+02, 8.600e+02, 8.120e+02,
        7.620e+02, 7.340e+02, 7.320e+02, 1.408e+03, 7.770e+02, 5.900e+02,
        4.780e+02, 4.830e+02, 4.620e+02, 4.410e+02, 3.380e+02, 3.370e+02,
        3.170e+02, 1.980e+02, 1.390e+02, 1.020e+02, 8.600e+01, 4.500e+01,
        3.900e+01, 3.500e+01, 1.900e+01, 1.600e+01, 1.300e+01, 1.100e+01,
        7.000e+00, 4.000e+00, 1.000e+00, 4.000e+00]),
 array([  48.  ,   50.04,   52.08,   54.12,   56.16,   58.2 ,   60.24,
          62.28,   64.32,   66.36,   68.4 ,   70.44,   72.48,   74.52,
          76.56,   78.6 ,   80.64,   82.68,   84.72,   86.76,   88.8 ,
          90.84,   92.88,   94.92,   96.96,   99.  ,  101.04,  103.08,
         105.12,  107.16,  109.2 ,  111.24,  113.28,  115.32,  117.36,
         119.4 ,  121.44,  123.48,  125.52,  127.56,
```

```
        129.6 , 131.64, 133.68, 135.72, 137.76, 139.8 , 141.84, 143.88,
        145.92, 147.96, 150.  , 152.04, 154.08, 156.12, 158.16, 160.2 ,
        162.24, 164.28, 166.32, 168.36, 170.4 , 172.44, 174.48, 176.52,
        178.56, 180.6 , 182.64, 184.68, 186.72, 188.76, 190.8 , 192.84,
        194.88, 196.92, 198.96, 201.  , 203.04, 205.08, 207.12, 209.16,
        211.2 , 213.24, 215.28, 217.32, 219.36, 221.4 , 223.44, 225.48,
        227.52, 229.56, 231.6 , 233.64, 235.68, 237.72, 239.76, 241.8 ,
        243.84, 245.88, 247.92, 249.96, 252.  ]),
<BarContainer object of 100 artists>)
```

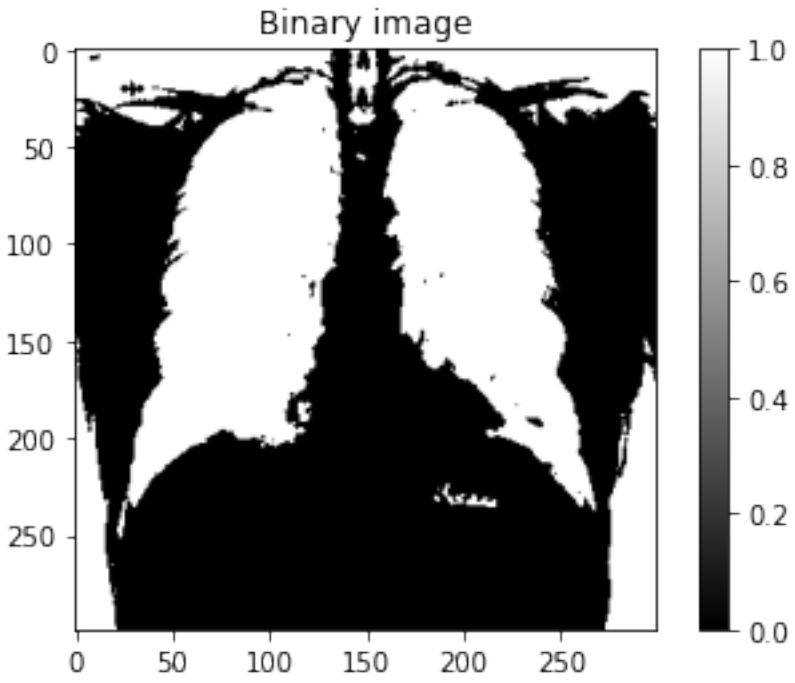


[ ]:
```
c=1.4 #contras
b = 0 #brightness
im_gray_array = c*np.int16(im_gray_array)+b
im_gray_array = np.uint8(np.maximum(np.minimum(im_gray_array,255),0))
```

[ ]:
```
th = 180
im_bin = (im_gray_array > th) | (im_gray_array < 25)
im_bin = 1-im_bin
```

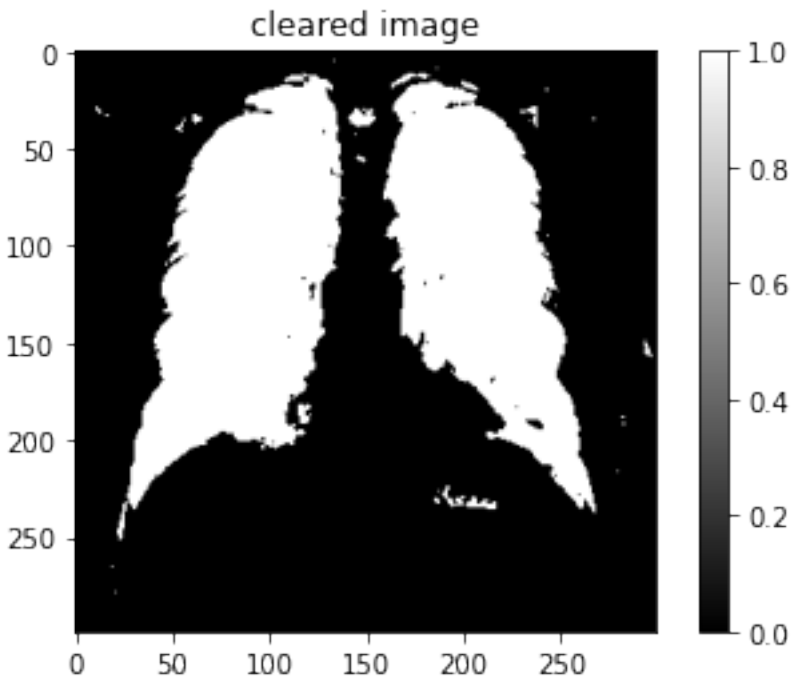
[ ]:
```
plt.imshow(im_bin,cmap='gray')
cbar = plt.colorbar()
plt.title('Binary image')
```

[ ]:
```
Text(0.5, 1.0, 'Binary image')
```



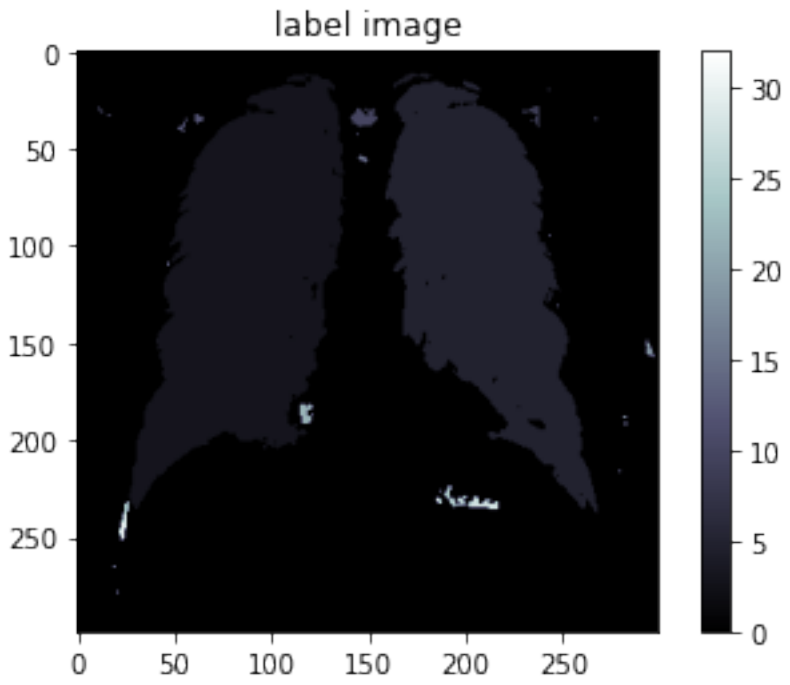
[ ]:
```
#Remove the blobs connected to the border of the image.

cleared = clear_border(im_bin)
plt.imshow(cleared,cmap='gray')
cbar = plt.colorbar()
plt.title('cleared image')
```

[ ]:
```
Text(0.5, 1.0, 'cleared image')
```



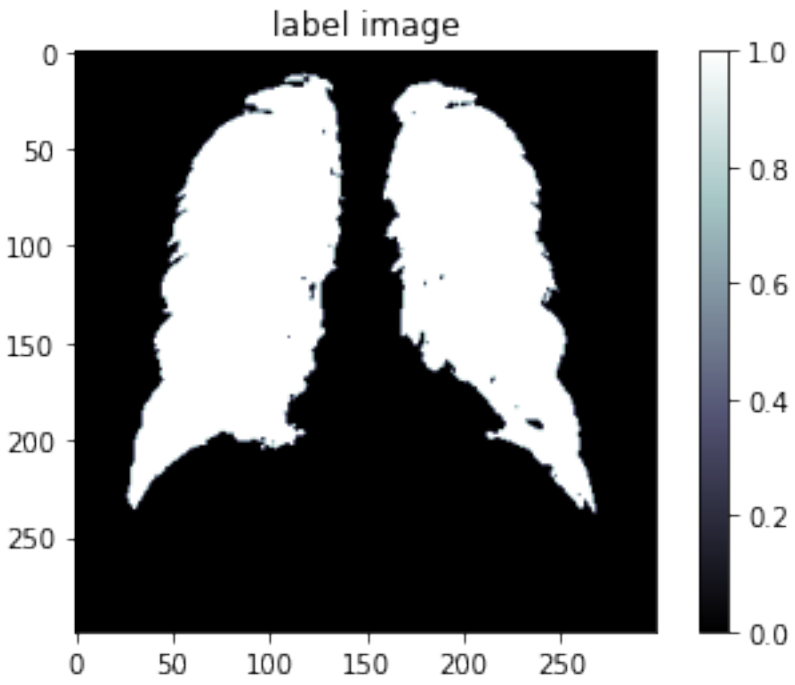
[ ]:
```
#label the image

label_image = label(cleared)
plt.imshow(label_image,cmap=plt.cm.bone)
cbar = plt.colorbar()
plt.title('label image')
```

[ ]:
```
Text(0.5, 1.0, 'label image')
```



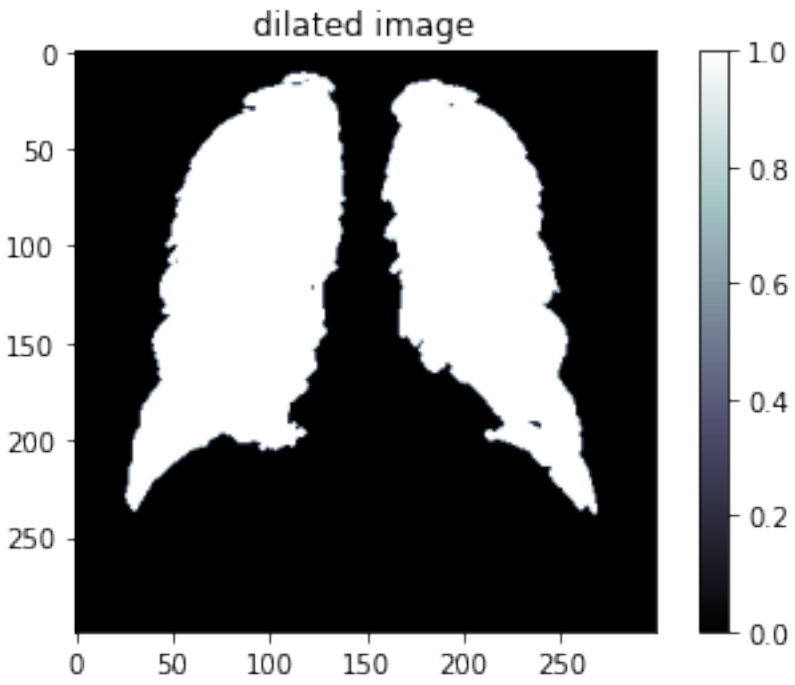
[ ]:
```
#Keep the labels with 2 largest areas.
areas = [r.area for r in regionprops(label_image)]
areas.sort()
if len(areas) > 2:
    for region in regionprops(label_image):
        if region.area < areas[-2]:
            for coordinates in region.coords:
                   label_image[coordinates[0], coordinates[1]] = 0
binary = label_image > 0

plt.imshow(binary,cmap=plt.cm.bone)
cbar = plt.colorbar()
plt.title('label image')
```

[ ]:
```
Text(0.5, 1.0, 'label image')
```



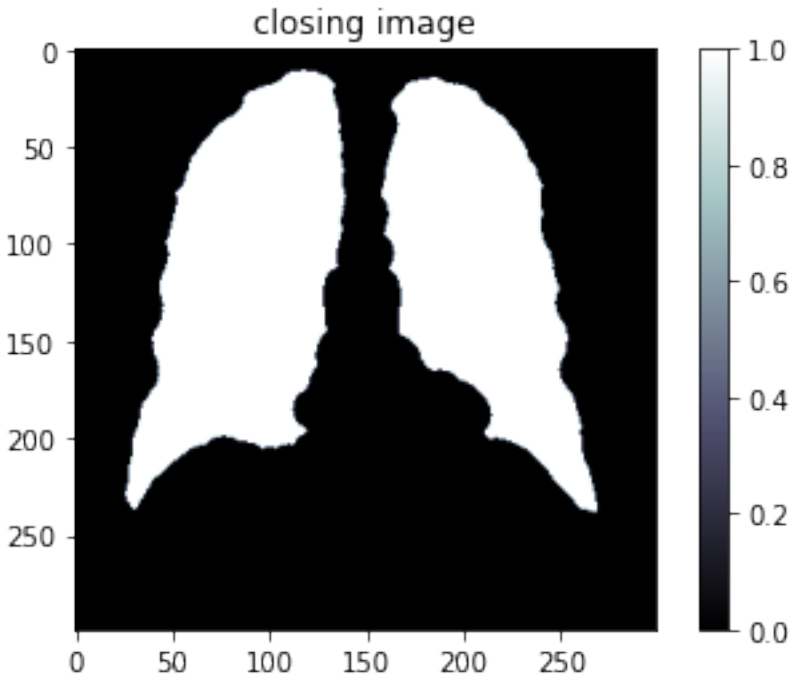
[ ]:
```
#dilation operation with a disk of radius 2. This operation is seperate the␣
↪lung nodules attached to the blood vessels.

binary = morphology.binary_dilation(binary)
plt.imshow(binary,cmap=plt.cm.bone)
cbar = plt.colorbar()
plt.title('dilated image')
```

[ ]:
```
Text(0.5, 1.0, 'dilated image')
```



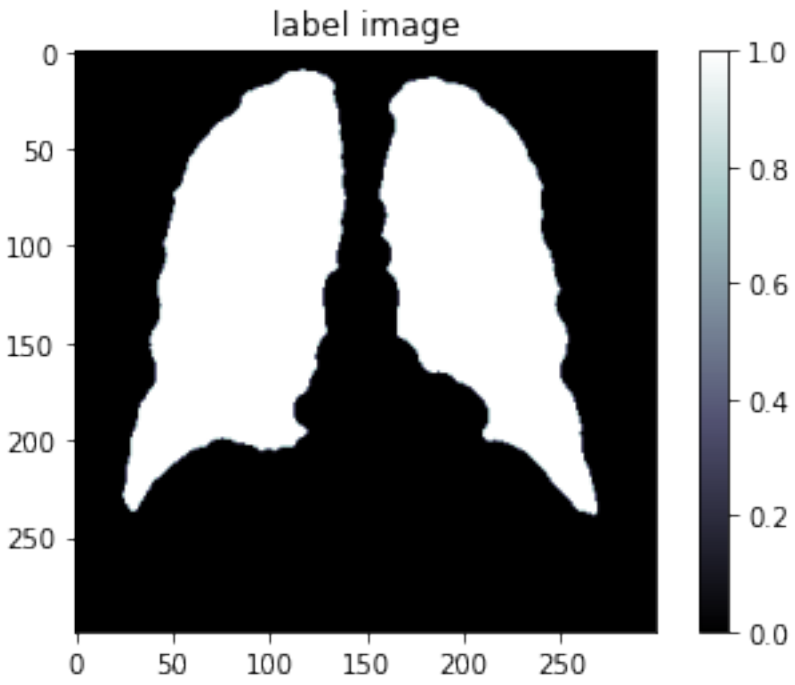
[ ]:
```
#Closure operation with a disk of radius 10. This operation is to keep nodules␣
↪attached to the lung wall.

selem = disk(10)
binary = morphology.binary_closing(binary, selem)
plt.imshow(binary,cmap=plt.cm.bone)
cbar = plt.colorbar()
plt.title('closing image')
```

[ ]:
```
Text(0.5, 1.0, 'closing image')
```



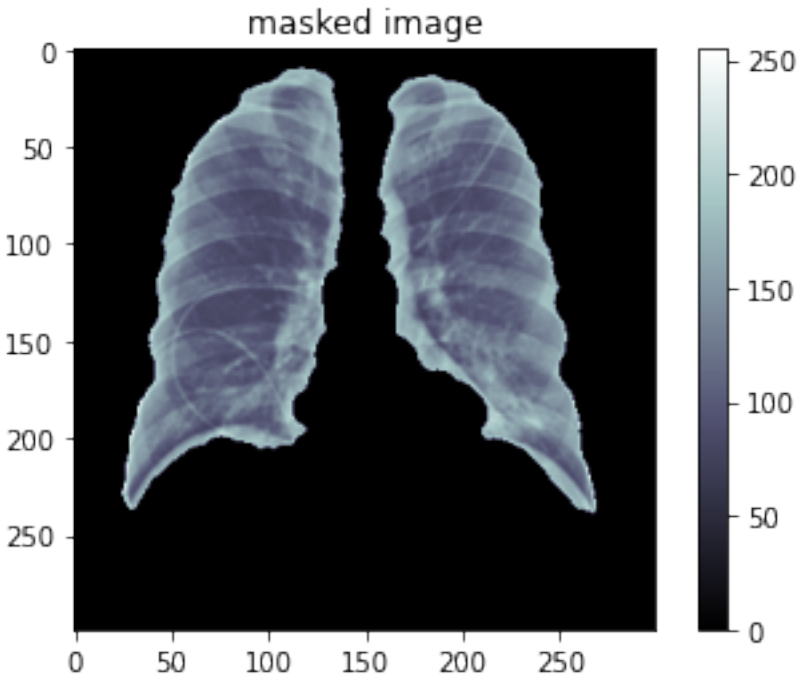
[ ]:
```
#Fill in the small holes inside the binary mask of lungs.

edges = roberts(binary)
binary = ndi.binary_fill_holes(edges)
plt.imshow(binary,cmap=plt.cm.bone)
cbar = plt.colorbar()
plt.title('label image')
```

[ ]:
```
Text(0.5, 1.0, 'label image')
```



[ ]:
```
#Superimpose the binary mask on the input image.

get_high_vals = binary == 0
im_gray_array[get_high_vals] = 0
plt.imshow(im_gray_array,cmap=plt.cm.bone)
cbar = plt.colorbar()
plt.title('masked image')
```

[ ]:
```
Text(0.5, 1.0, 'masked image')
```



[ ]:
```
'''
def make_mb_image(i_img, i_gt, i_pred,ds_op=lambda x: x[::4,::4]):
    n_img = (i_img-i_img.mean())/(2*i_img.std())+0.5
    c_img = plt.cm.bone(n_img)[:,:,:3]
    c_img=mark_boundaries(ds_op(c_img),label_img=ds_op(i_pred), color =␣
↪(1,0,0),mode='thick')
  ␣
↪c_img=mark_boundaries(c_img,label_img=ds_op(i_gt),color=(0,1,0),mode='thick|')
    return c_img

plt.imshow(make_mb_image(im_gray_array1, im_bin, binary))

'''
```

[ ]:
```
"\ndef make_mb_image(i_img, i_gt, i_pred,ds_op=lambda x: x[::4,::4]):\n    n_img
= (i_img-i_img.mean())/(2*i_img.std())+0.5\n    c_img =
plt.cm.bone(n_img)[:,:,:3]\n
c_img=mark_boundaries(ds_op(c_img),label_img=ds_op(i_pred), color =
(1,0,0),mode='thick')\n
c_img=mark_boundaries(c_img,label_img=ds_op(i_gt),color=(0,1,0),mode='thick|')\n
return c_img\n\nplt.imshow(make_mb_image(im_gray_array1, im_bin, binary))\n\n"
```

[ ]:
```
def image_print(im):
```

```
    plt.imshow(im,cmap=plt.cm.bone)
    cbar = plt.colorbar()
    plt.title('masked image')
```

[ ]:
```
imagePaths = []
for dirname, _, filenames in os.walk(r'C:
 ↪\Users\josep\Desktop\covid\COVID-19_Radiography_Dataset\Covid'):
    for filename in filenames:
        if (filename[-3:] == 'png'):
            imagePaths.append(os.path.join(dirname, filename))
imagePaths.sort(key=lambda x:str (x[:-5]))
```

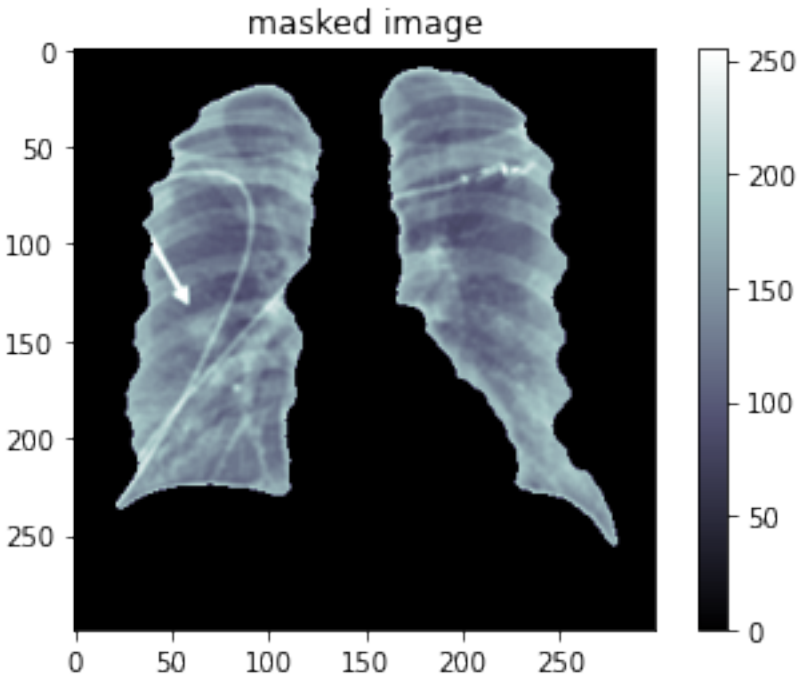
2 Covid Segmentation

[ ]:
```
def covid_segementation(path, plot=False):

    count = 0
    for i in tqdm(path):
        count = count+1
        im_RGB = cv2.imread(i)
        im_gray = cv2.cvtColor(im_RGB,cv2.COLOR_BGR2GRAY)
        im_gray_array = np.array(im_gray)
        c=1.4 #contras
        b = 0 #brightness
        im_gray_array = c*np.int16(im_gray_array)+b
        im_gray_array = np.uint8(np.maximum(np.minimum(im_gray_array,255),0))
        th = 180
        im_bin = (im_gray_array > th) | (im_gray_array < 25)
        im_bin = 1-im_bin
        cleared = clear_border(im_bin)
        label_image = label(cleared)
        areas = [r.area for r in regionprops(label_image)]
        areas.sort()
        if len(areas) > 2:
            for region in regionprops(label_image):
                if region.area < areas[-2]:
                    for coordinates in region.coords:
                           label_image[coordinates[0], coordinates[1]] = 0
        binary = label_image > 0
        binary = morphology.binary_dilation(binary)
        selem = disk(10)
        binary = morphology.binary_closing(binary, selem)
        edges = roberts(binary)
        binary = ndi.binary_fill_holes(edges)
        get_high_vals = binary == 0
        im_gray_array[get_high_vals] = 0
```

```
        cv2.imwrite(f'segmentation/Covid - Segemented/Covid_seg_{count}.
↪png',im_gray_array)

    if plot == True:
        image_print(im_gray_array)
```

[ ]:
```
test = covid_segementation(imagePaths, True)
```

```
100%|      | 3616/3616 [03:05<00:00, 19.44it/s]
```



[ ]:
```
imagePaths = []
for dirname, _, filenames in os.walk(r'C:
↪\Users\josep\Desktop\covid\COVID-19_Radiography_Dataset\Normal'):
    for filename in filenames:
        if (filename[-3:] == 'png'):
            imagePaths.append(os.path.join(dirname, filename))
imagePaths.sort(key=lambda x:str (x[:-5]))
len(imagePaths)
```

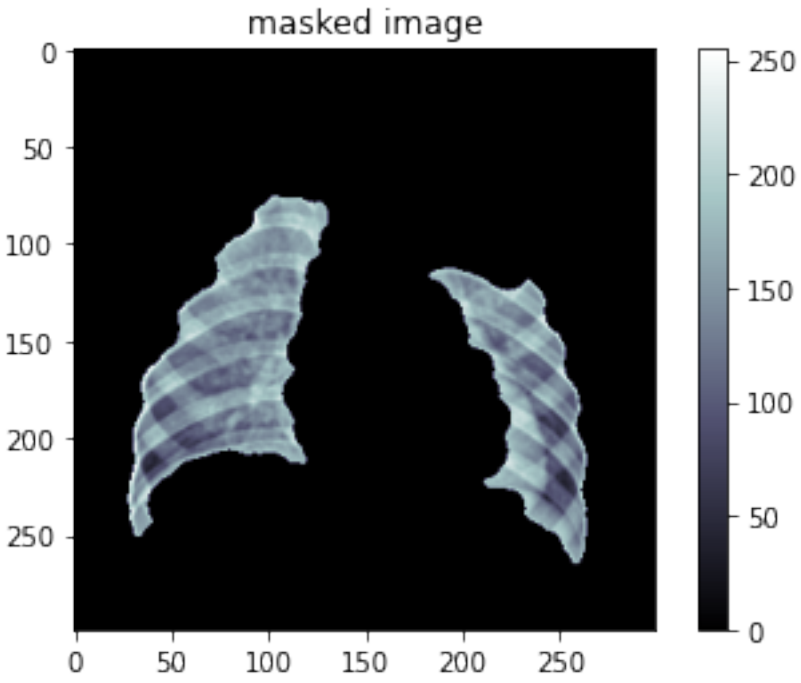
[ ]: 3616

3 Normal Lungs Segmentation

[ ]:
```
def Normal_segementation(path, plot=False):

    count = 0
    for i in tqdm(path):
        count = count+1
        im_RGB = cv2.imread(i)
        im_gray = cv2.cvtColor(im_RGB,cv2.COLOR_BGR2GRAY)
        im_gray_array = np.array(im_gray)
        c=1.4 #contras
        b = 0 #brightness
        im_gray_array = c*np.int16(im_gray_array)+b
        im_gray_array = np.uint8(np.maximum(np.minimum(im_gray_array,255),0))
        th = 180
        im_bin = (im_gray_array > th) | (im_gray_array < 25)
        im_bin = 1-im_bin
        cleared = clear_border(im_bin)
        label_image = label(cleared)
        areas = [r.area for r in regionprops(label_image)]
        areas.sort()
        if len(areas) > 2:
            for region in regionprops(label_image):
                if region.area < areas[-2]:
                    for coordinates in region.coords:
                           label_image[coordinates[0], coordinates[1]] = 0
        binary = label_image > 0
        binary = morphology.binary_dilation(binary)
        selem = disk(10)
        binary = morphology.binary_closing(binary, selem)
        edges = roberts(binary)
        binary = ndi.binary_fill_holes(edges)
        get_high_vals = binary == 0
        im_gray_array[get_high_vals] = 0


        cv2.imwrite(f'segmentation/Normal - Segemented/Normal_seg_{count}.
↪png',im_gray_array)

    if plot == True:
        image_print(im_gray_array)
```

[ ]:
```
test = Normal_segementation(imagePaths, True)
```

```
100%|      | 3616/3616 [03:09<00:00, 19.06it/s]
```


masked image
0
50
100
150
200
250
0
50
100
150
200
250
250
200
150
100
50
0

[ ]: