blobDetection_

June 20, 2021

Autor: José Verdú-Díaz

La herramienta inferior forma parte de poly2pix, disponible aquí: https://github.com/Jose-Verdu-Diaz/sa_poly2pixel ____

Cargamos las librerías necesarias

[13]:
```
import cv2
import numpy as np
from matplotlib import pyplot as plt
```

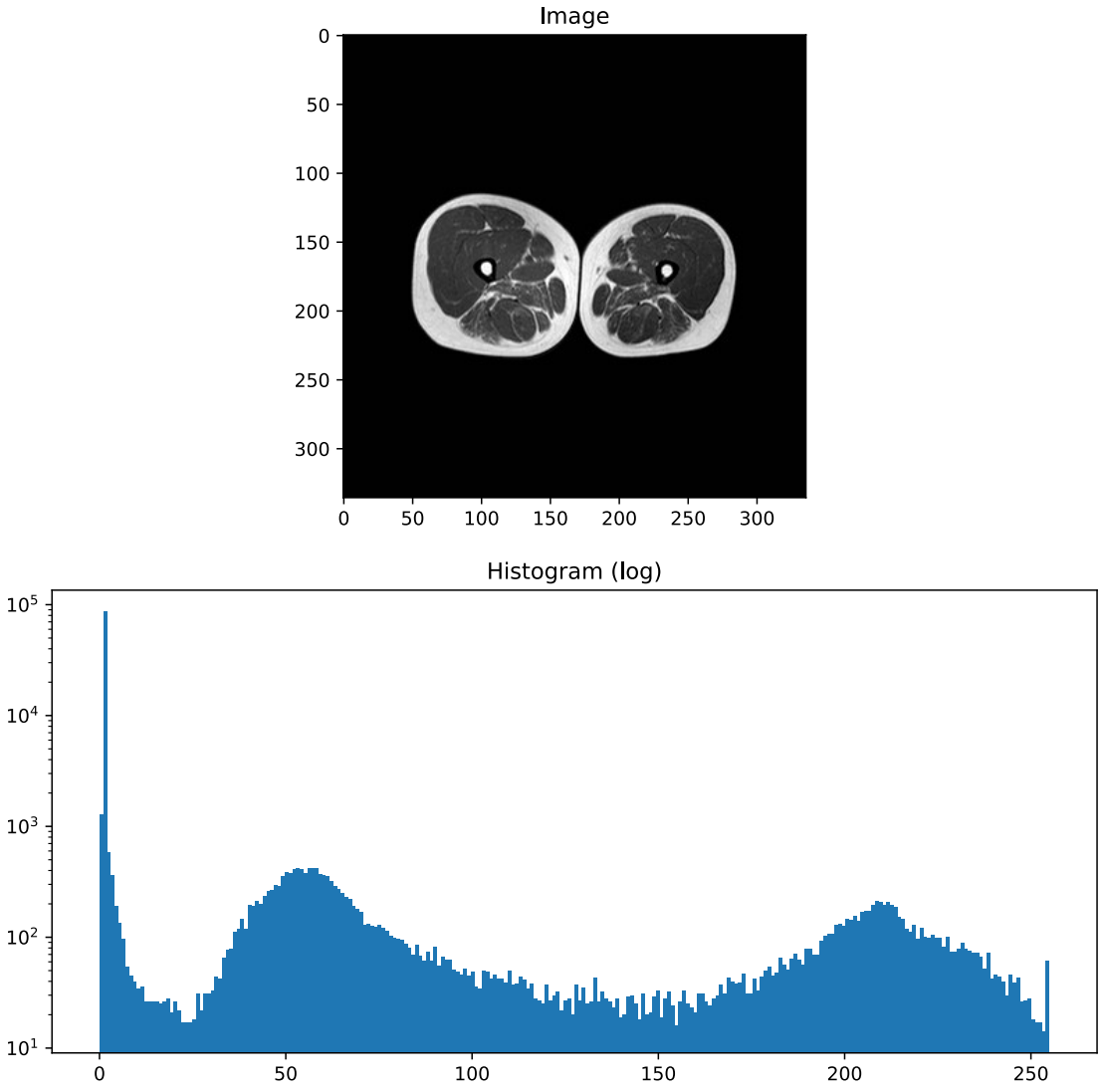
Cargamos la imágen de prueba

[14]:
```
file = '0045'
img_path = f'../../projects/RM1/img/0001-{file}.jpg'
mask_path = f'../../projects/RM1/masks/0001-{file}.bmp'

if os.path.isfile(img_path):
    img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
else:
    print (f'The image {img_path} does not exist.')

if os.path.isfile(mask_path):
    mask = cv2.imread(mask_path, cv2.IMREAD_GRAYSCALE)
else:
    print (f'The image {mask_path} does not exist.')

# Imágenes RGB para la representación de los contornos y centroides
img_rgb = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
mask_rgb = cv2.cvtColor(mask, cv2.COLOR_GRAY2BGR)
```

Mostramos la imagen junto a su histograma (eje y en escala logaritmica). Se pueden ver claramente 3 picos, de izquierda a derecha: el fondo negro, el músculo gris oscuro y el tejido adiposo gris claro. Obsérvese que el fondo no es de un negro perfecto.

[40]:
```
fig, (ax1, ax2) = plt.subplots(2,figsize=(10,10))
ax1.imshow(img, cmap='gray', interpolation='none');
ax1.set_title('Image')
ax2.hist(img.ravel(),256,[0,255]);
ax2.set_title('Histogram (log)')
```

```
ax2.set_yscale('log')
```



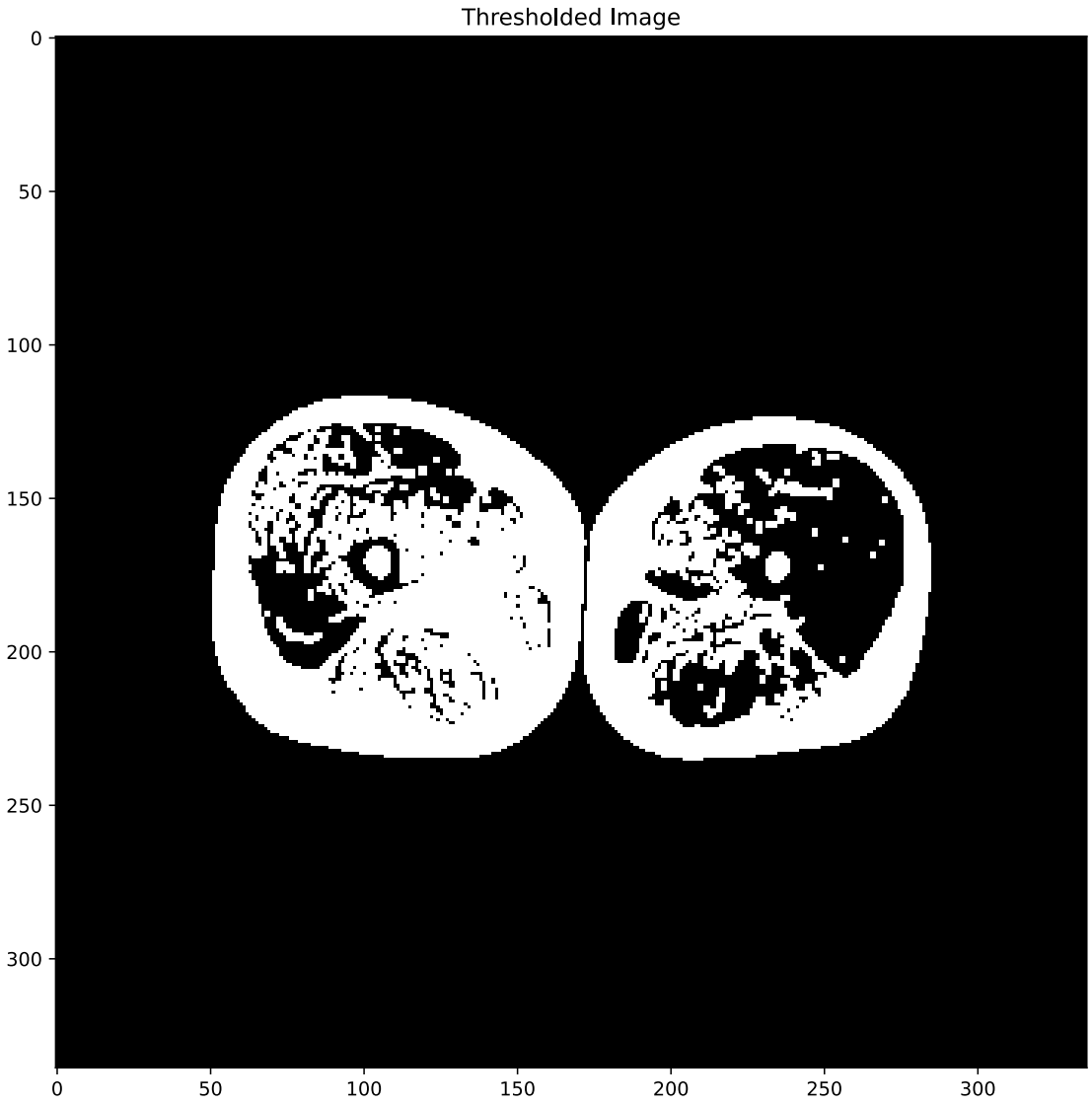
Se aplica un threshold con un valor entre 20 y 100 (con aumentos de 5). Para encontrar el valor idóneo se realiza un barrido desde el límite inferior hasta encontrar únicamente 2 contornos. Además, se aplica una transformación morfológica para eliminar el ruido que pueda confundir al detector de contornos. El objetivo es conseguir una separación clara entre la pierna izquierda y derecha. Éste es el punto crítico donde este sistema puede fallar. Se muestra la imágen obtenida al aplicar el threshold.

[41]:
```
success = False
for th in np.arange(20, 101, 5).tolist():
    ret, thresh = cv2.threshold(img,th,255,cv2.THRESH_BINARY)

    # Remove white noise in mask
```

```
    kernel = np.ones((2,2),np.uint8)
    thresh = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel, iterations = 1)

    contours, hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.
 ↪CHAIN_APPROX_SIMPLE)

    if len(contours)==2:
        success = True
        print(f'Threshold: {th}')
        break
if not success:
    print(f'{len(contours)} found at max threshold')

fig, ax = plt.subplots(figsize=(10, 10));
ax.imshow(thresh, cmap='gray', interpolation='none');
ax.set_title('Thresholded and Morphed Image')
```

```
Threshold: 55
```

[41]: `Text(0.5, 1.0, 'Thresholded Image')`

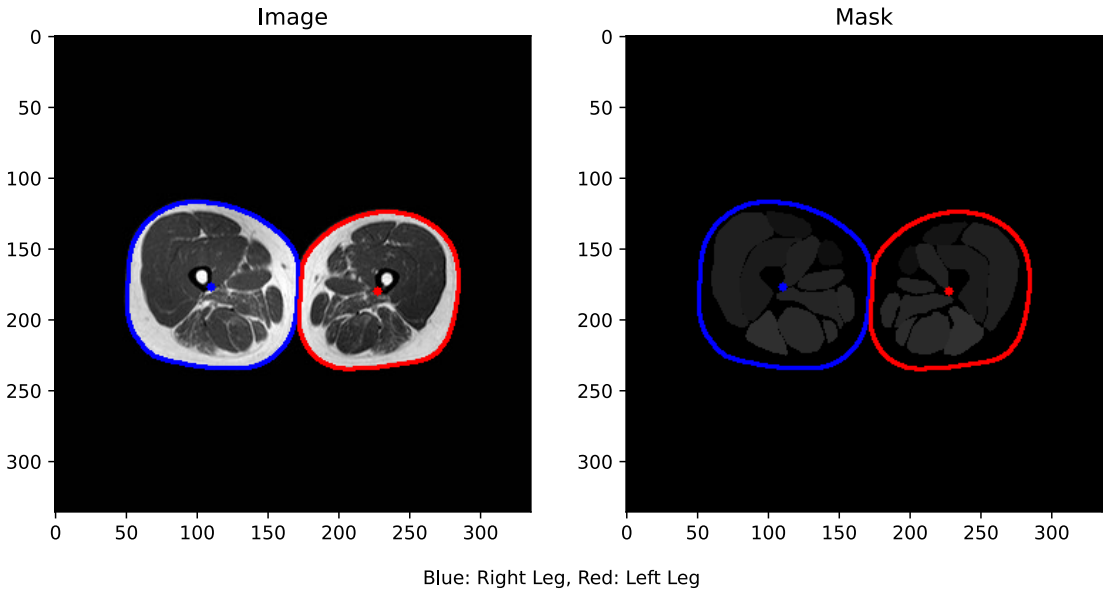


Detectamos los contornos externos (cv2.RETR_EXTERNAL) y calculamos los centroides de cada contorno. Los representamos en azul (pierna derecha) y rojo (pierna izquierda). Podemos distinguir entre izquierda y derecha a partir de la coordenada X de los centroides.

[39]:
```
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.
↪CHAIN_APPROX_SIMPLE)

# Cetro del contorno
centroids = []
for i,c in enumerate(contours):
    M = cv2.moments(c)
    cX = int(M["m10"] / M["m00"])
    cY = int(M["m01"] / M["m00"])
```

```
    centroids.append((cX,cY))

if not len(centroids) == 2:
    print(f'Len of centroids is {len(centroids)}')
else:
    # Derecha: Azul
    # Izquierda: Rojo
        if centroids[0][0] > centroids[1][0]:
            cv2.drawContours(img_rgb, [contours[1]], -1, (0, 0, 255), 2)
            cv2.drawContours(img_rgb, [contours[0]], -1, (255, 0, 0), 2)
            cv2.circle(img_rgb, centroids[1], 3, (0, 0, 255), -1)
            cv2.circle(img_rgb, centroids[0], 3, (255, 0, 0), -1)

            cv2.drawContours(mask_rgb, [contours[1]], -1, (0, 0, 255), 2)
            cv2.drawContours(mask_rgb, [contours[0]], -1, (255, 0, 0), 2)
            cv2.circle(mask_rgb, centroids[1], 3, (0, 0, 255), -1)
            cv2.circle(mask_rgb, centroids[0], 3, (255, 0, 0), -1)

        elif centroids[0][0] < centroids[1][0]:
            cv2.drawContours(img_rgb, [contours[0]], -1, (0, 0, 255), 2)
            cv2.drawContours(img_rgb, [contours[1]], -1, (255, 0, 0), 2)
            cv2.circle(img_rgb, centroids[0], 3, (0, 0, 255), -1)
            cv2.circle(img_rgb, centroids[1], 3, (255, 0, 0), -1)

            cv2.drawContours(mask_rgb, [contours[0]], -1, (0, 0, 255), 2)
            cv2.drawContours(mask_rgb, [contours[1]], -1, (255, 0, 0), 2)
            cv2.circle(mask_rgb, centroids[0], 3, (0, 0, 255), -1)
            cv2.circle(mask_rgb, centroids[1], 3, (255, 0, 0), -1)

        else: print('Error')

fig, axs = plt.subplots(1, 2, figsize=(10, 10));
axs[0].imshow(img_rgb, cmap='gray', interpolation='none');
axs[0].set_title('Image')
axs[1].imshow(mask_rgb, cmap='gray', interpolation='none');
axs[1].set_title('Mask')
plt.figtext(0.5,0.27,'Blue: Right Leg, Red: Left Leg', ha='center', va='bottom')
```

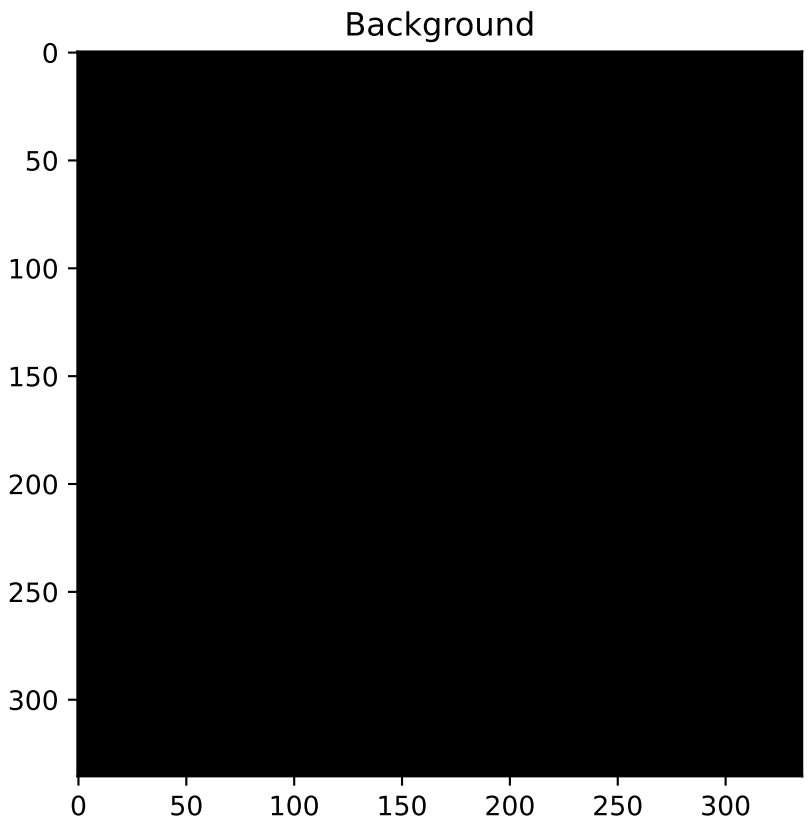
[39]: Text(0.5, 0.27, 'Blue: Right Leg, Red: Left Leg')



Generamos un fondo negro con las mismas dimensiones de la imágen.

[43]:
```
background = np.zeros(img.shape, np.uint8)

fig, ax = plt.subplots(figsize=(5, 5));
ax.imshow(background, cmap='gray', interpolation='none');
ax.set_title('Background')
```

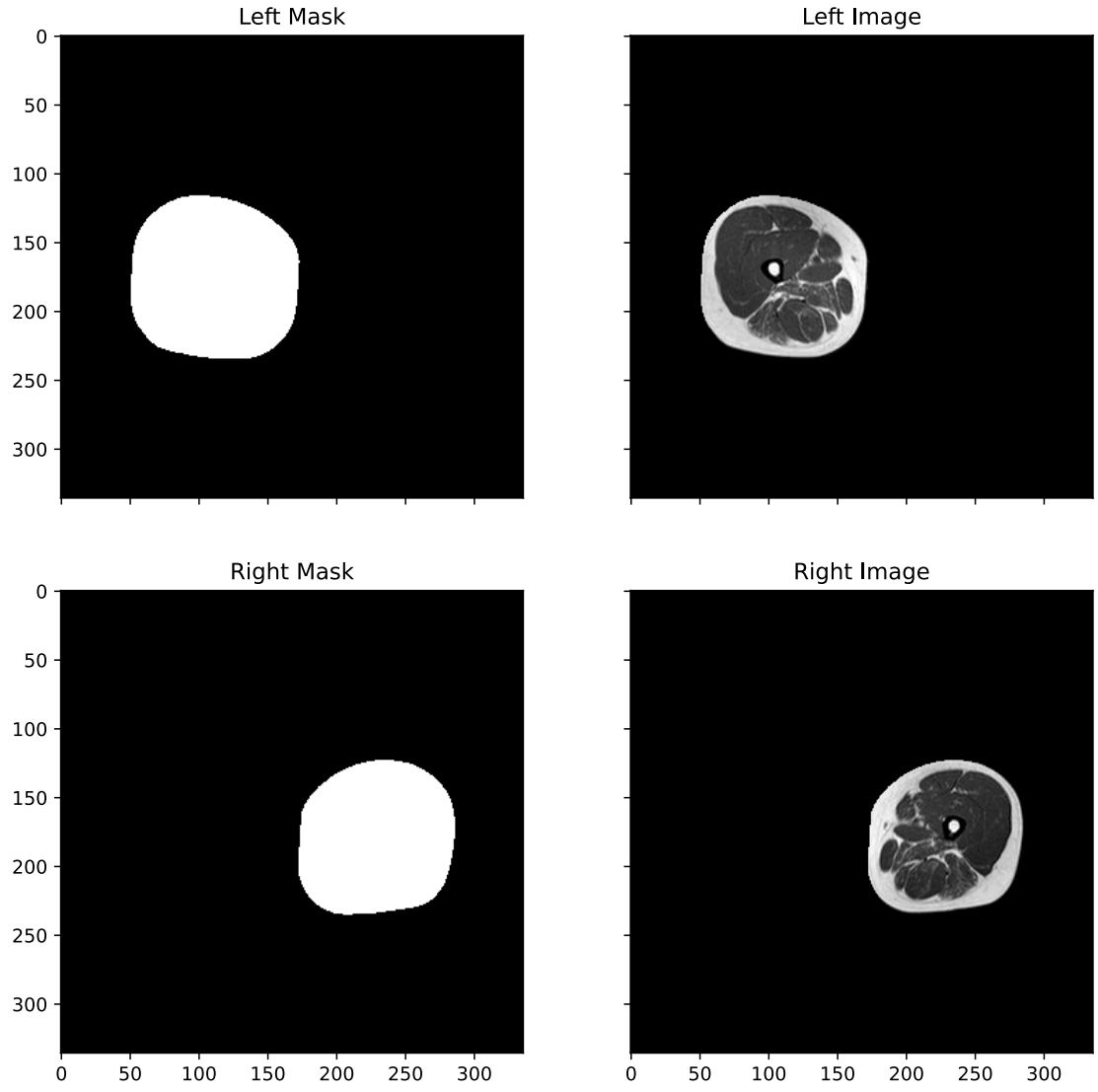
[43]:
```
Text(0.5, 1.0, 'Background')
```



Obtenemos las máscaras a partir de los contornos y las aplicamos sobre la imágen original. Obtenemos imágenes, cada una con una pierna distinta.

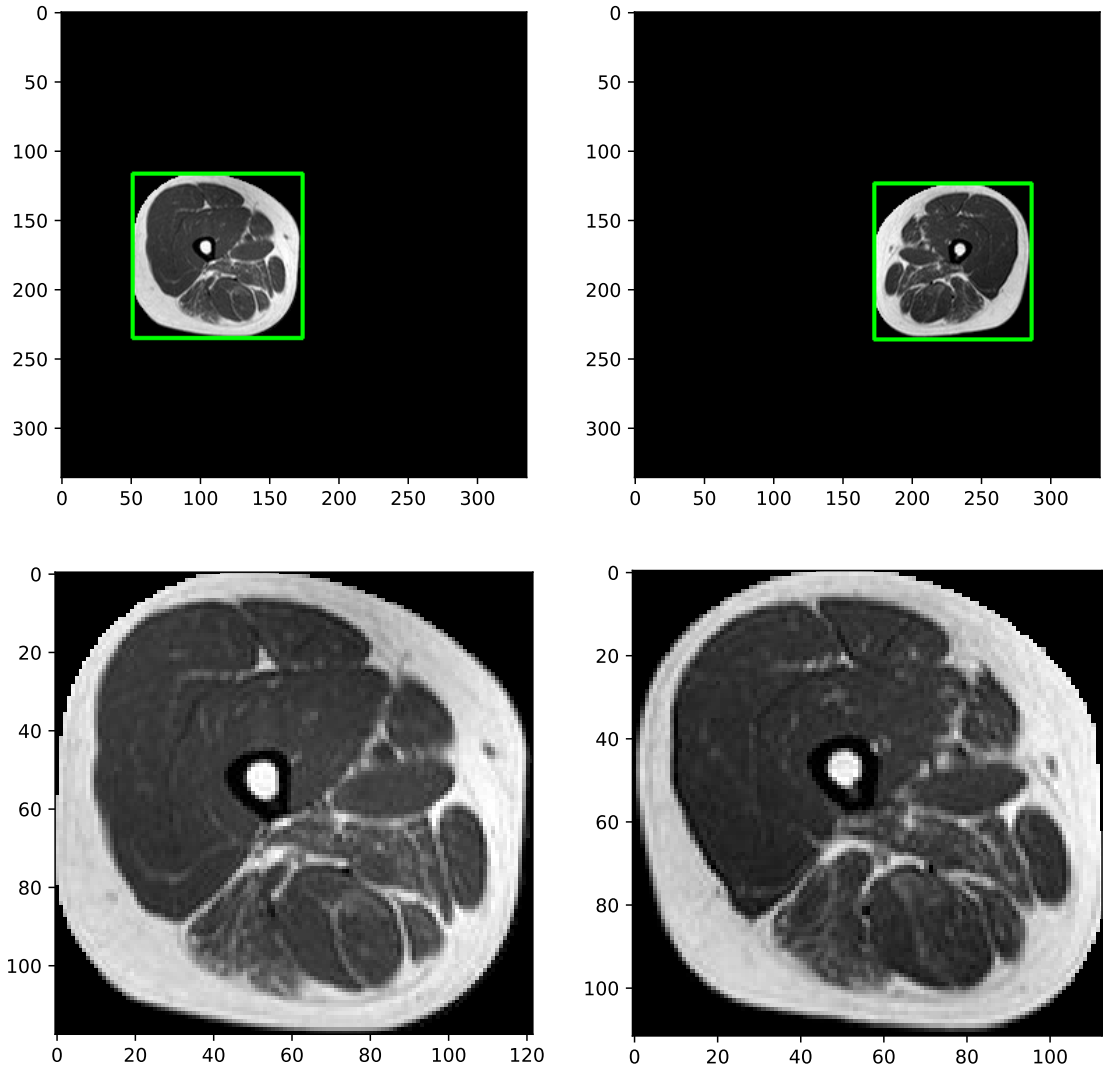
[44]:
```
if centroids[0][0] > centroids[1][0]:
    mask_L = cv2.drawContours(background.copy(), [contours[1]], -1, 255, -1)
    mask_R = cv2.drawContours(background.copy(), [contours[0]], -1, 255, -1)

elif centroids[0][0] < centroids[1][0]:
    mask_L = cv2.drawContours(background.copy(), [contours[0]], -1, 255, -1)
    mask_R = cv2.drawContours(background.copy(), [contours[1]], -1, 255, -1)

else: pass

img_L = cv2.bitwise_and(img,img,mask = mask_L)
img_R = cv2.bitwise_and(img,img,mask = mask_R)

fig, axs = plt.subplots(2,2,figsize=(10,10))
axs[0,0].imshow(mask_L, cmap='gray', interpolation='none');
axs[0,0].set_title('Left Mask')
axs[1,0].imshow(mask_R, cmap='gray', interpolation='none');
axs[1,0].set_title('Right Mask')
```

```
axs[0,1].imshow(img_L, cmap='gray', interpolation='none');
axs[0,1].set_title('Left Image')
axs[1,1].imshow(img_R, cmap='gray', interpolation='none');
axs[1,1].set_title('Right Image')

for ax in fig.get_axes(): ax.label_outer()
```



Encontramos la "bounding box" de cada pierna y recortamos las imágenes a sus dimensiones. Además volteamos la pierna izquierda: ahora solo existen piernas derechas. Con ello reducimos a la mitad la cantidad de músculos, duplicamos la cantidad de imágenes y (en este caso) reducimos las dimensiones de las imágenes a más de la mitad.

[20]:
```
if centroids[0][0] > centroids[1][0]:
    x,y,w,h = cv2.boundingRect(contours[1])
    box_L = cv2.rectangle(cv2.cvtColor(img_L, cv2.
 ↪COLOR_GRAY2BGR),(x,y),(x+w,y+h),(0,255,0),2)
    cropped_L = img_L[y:y+h, x:x+w]

    x,y,w,h = cv2.boundingRect(contours[0])
    box_R = cv2.rectangle(cv2.cvtColor(img_R, cv2.
 ↪COLOR_GRAY2BGR),(x,y),(x+w,y+h),(0,255,0),2)
    cropped_R = cv2.flip(img_R[y:y+h, x:x+w], 1)

elif centroids[0][0] < centroids[1][0]:
    x,y,w,h = cv2.boundingRect(contours[0])
    box_L = cv2.rectangle(cv2.cvtColor(img_L, cv2.
 ↪COLOR_GRAY2BGR),(x,y),(x+w,y+h),(0,255,0),2)
    cropped_L = img_L[y:y+h, x:x+w]

    x,y,w,h = cv2.boundingRect(contours[1])
    box_R = cv2.rectangle(cv2.cvtColor(img_R, cv2.
 ↪COLOR_GRAY2BGR),(x,y),(x+w,y+h),(0,255,0),2)
    cropped_R = cv2.flip(img_R[y:y+h, x:x+w], 1)


fig, axs = plt.subplots(2,2,figsize=(10,10))
axs[0,0].imshow(box_L, cmap='gray', interpolation='none');
axs[0,0].set_title('Left Contour')
axs[0,1].imshow(box_R, cmap='gray', interpolation='none');
axs[0,0].set_title('Right Contour')
axs[1,0].imshow(cropped_L, cmap='gray', interpolation='none');
axs[1,1].imshow(cropped_R, cmap='gray', interpolation='none');
```



El código se puede automatizar para procesar toda una carpeta con RMs de la siguiente manera:

[ ]: