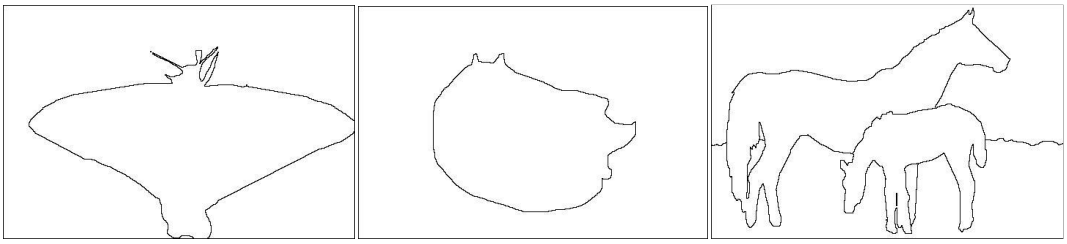
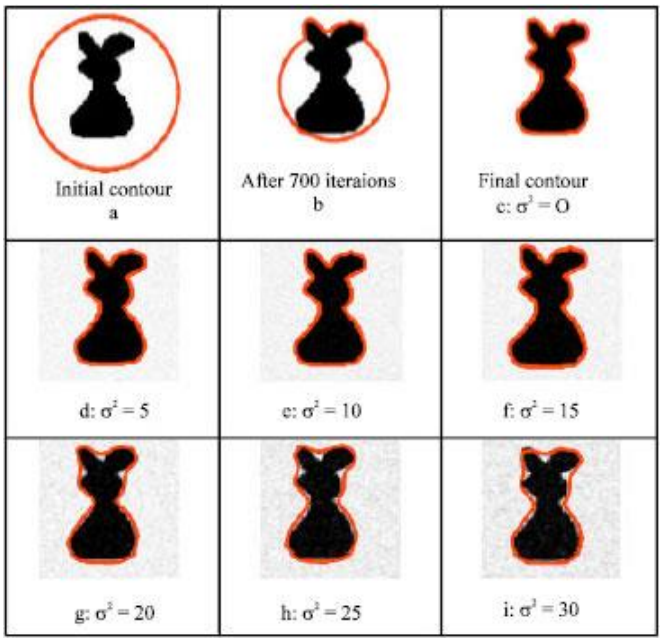
Asignatura	Datos del alumno	Fecha
Percepción Computacional	Apellidos: González Isorna	11/06/2019
	Nombre: Alberto	

Laboratorio: Contornos activos





L1_Alberto_Isorna

June 11, 2019

1 Laboratorio: Contornos activos

El objetivo de este laboratorio es familarizarnos con el uso de contornos activos aplicados a imagenes reales.

Los pasos que seguiremos serán los siguientes:

1. Importación de librerías y lectura de ficheros
2. Segmentación de imágenes con Contornos Activos
    1. Mariposa
    2. Buho
    3. Caballo
3. Conclusiones

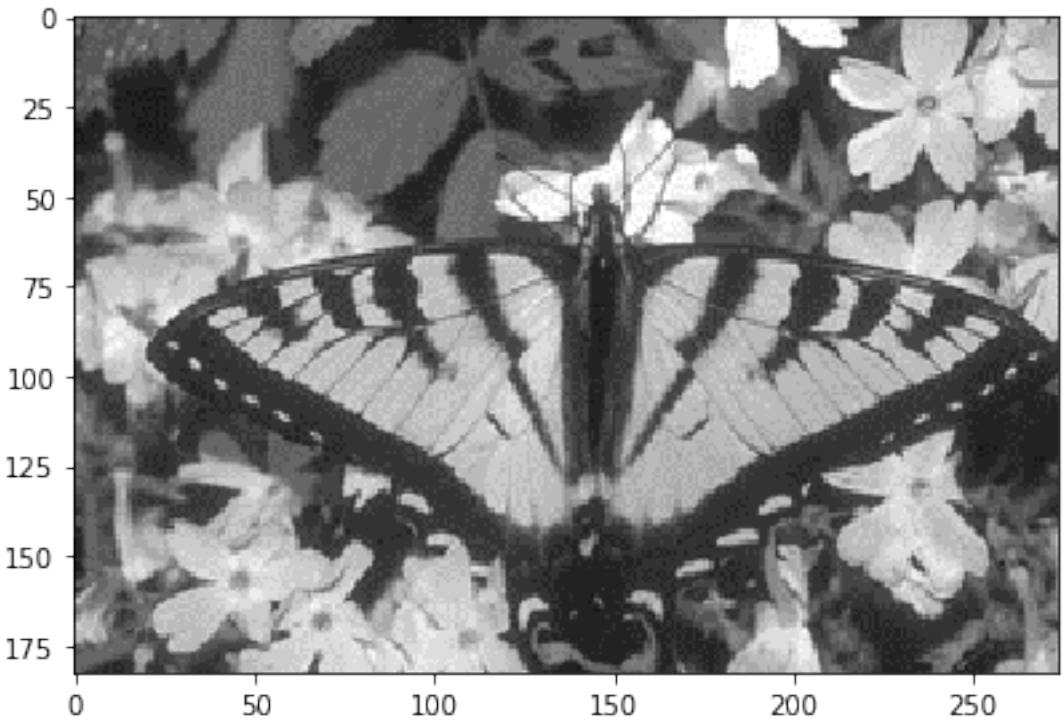
2 Importación de librerías y lectura de ficheros

In [509]:
```
# --- Importamos librerías
import matplotlib.pyplot as plt
import numpy as np

from skimage import color
from skimage import data
from skimage import io

from skimage.filters import gaussian,median
from skimage.segmentation import active_contour

from skimage.transform import resize
from math import log
from scipy import stats
```

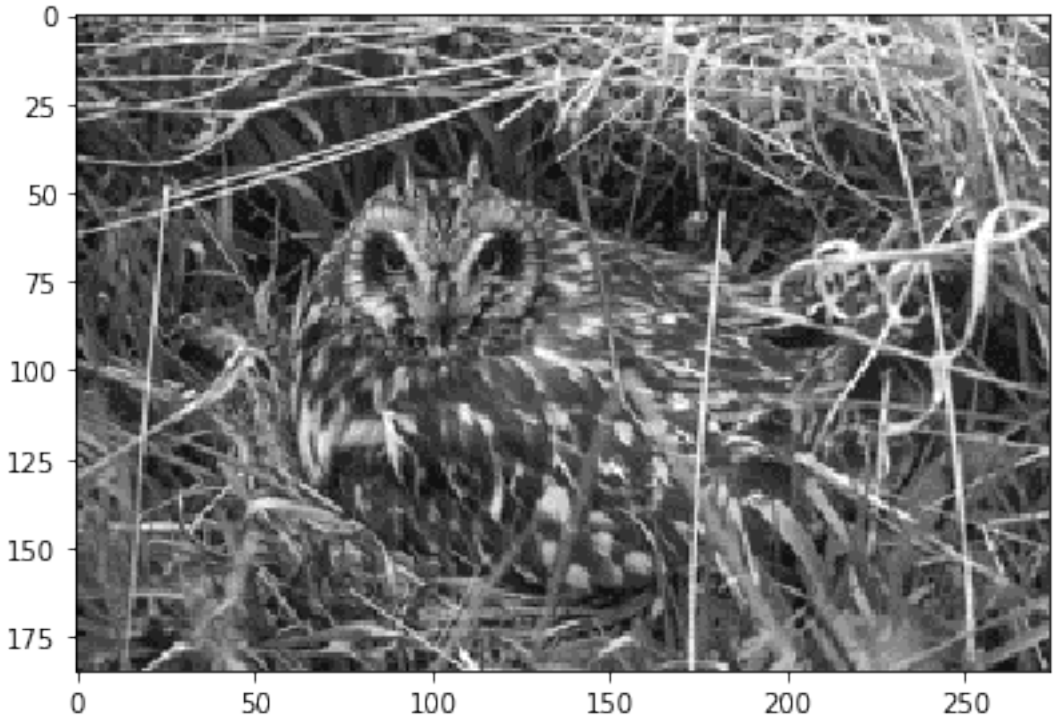
In [511]:
```
# --- Lectura de ficheros

# vector con nombres
vs =['tema10_act1a.png','tema10_act1b.png','tema10_act1c.png']
# vector con imagenes
vim=[]
```

```
        for s in vs:
            print(s)
            # Cargamos la imagen
            myim = io.imread(s)
            # La imagen tiene 3 canales asi que hacemos un cambio a gris
            im = color.rgb2gray(myim)
            print('shape = {}'.format(im.shape))
            # la mostramos
            io.imshow(im)
            plt.show()
            # guardamos en el vector
            vim.append(im)
```

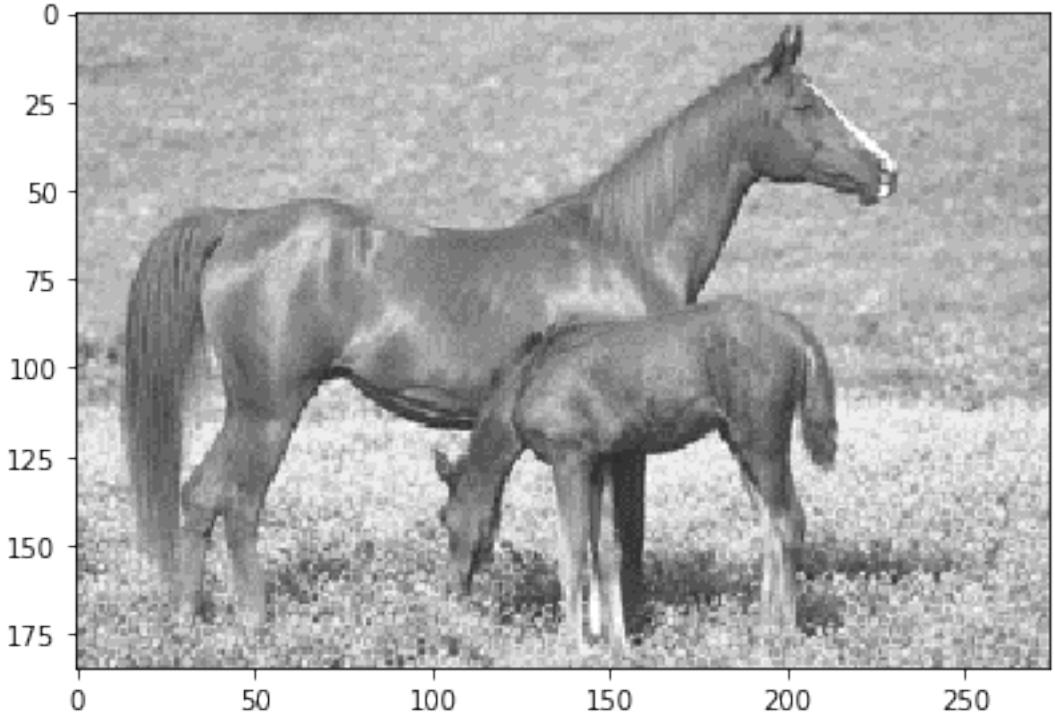
```
tema10_act1a.png
shape = (183, 274)
```




```
tema10_act1b.png
shape = (185, 274)
```




```
tema10_act1c.png
shape = (185, 274)
```



Vemos en las imagenes como tienen colores similares y regiones no muy distingibles.
A simple vista parece que la imagen del buho va a ser la mas dificil de tratar.

3 Segmentación de imágenes con Contornos Activos

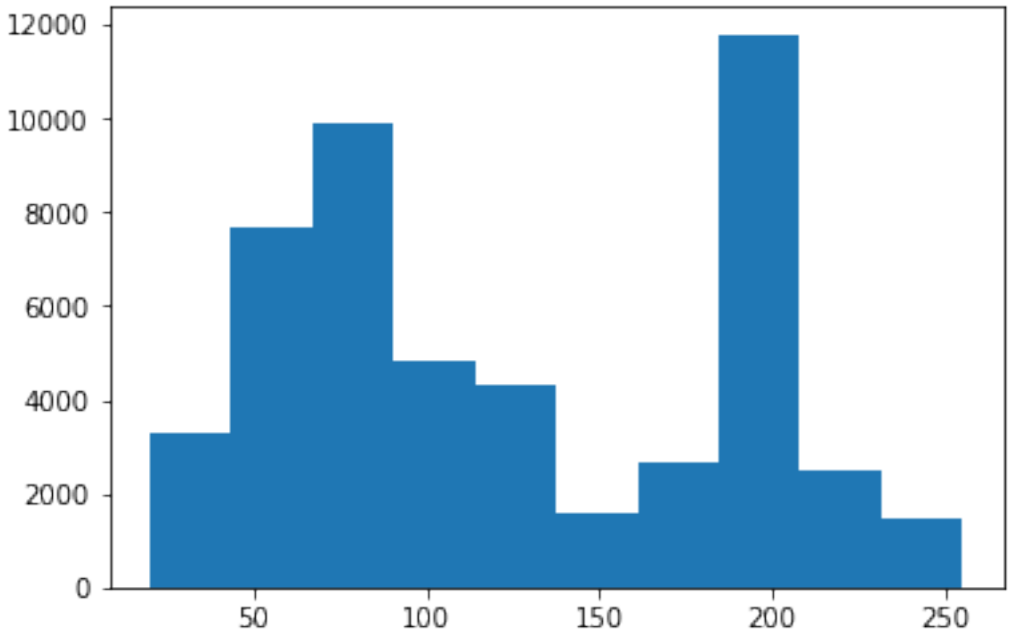
La estrategia que intentaremos será primero visualizar el histograma, posteriormente haremos separación binaria de la imagen y intentaremos aplicar metodos morfológicos que ayuden a realizar la segmentación de imagenes.

3.1 Mariposa

La mariposa tiene la peculiaridad que tiene una forma conocida, de un triángulo invertido aproximadamente, así que nos serviremos de esta propiedad. En cuanto a los colores, desafortunadamente no nos van a servir mucho para hacer la diferenciación.

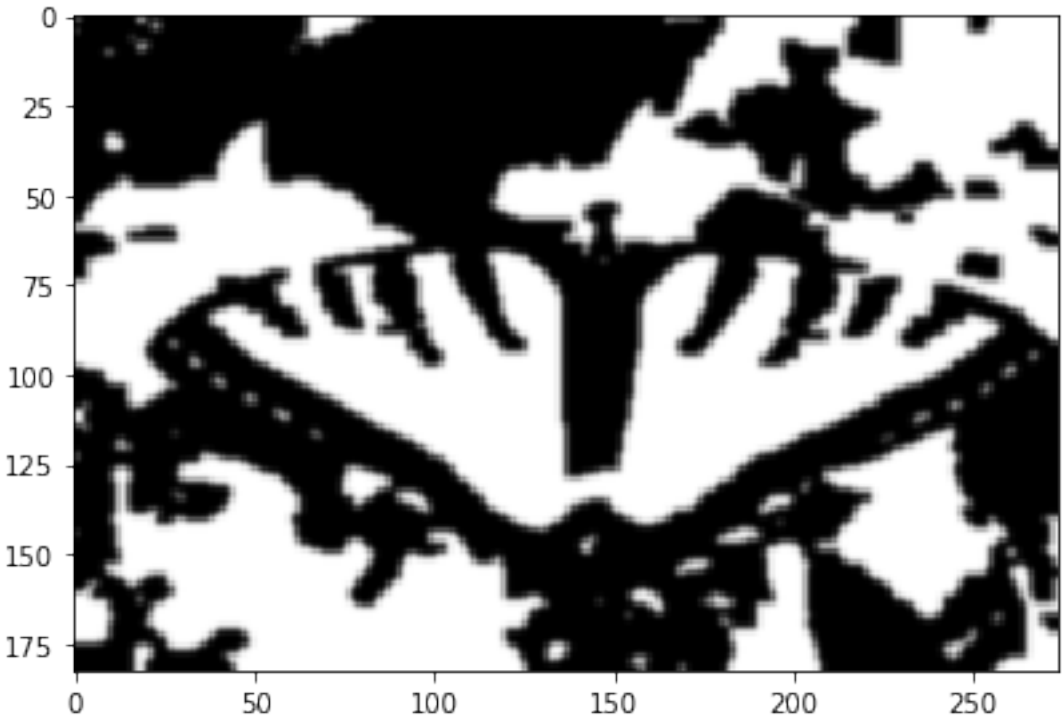
```
In [386]: im = vim[0].copy()
```

```
In [387]: # Display the histogram.
          plt.hist(im.ravel())
          plt.show()
```



Vemos dos zonas claramentes diferenciadas, las flores oscuras y las claras que se confunden con la mariposa.

```
In [389]: from skimage import morphology as mf
          from skimage import feature

          #r = 7
          #se = mf.disk(r)
          se = mf.rectangle(3,5)

          # binarizamos
          imf = im > 120

          # cerramos
          imc = mf.binary_closing(imf,se)

          #filtro gausiano
          img = gaussian(imc,1)

          # mostramos la imagen
          io.imshow(img)

Out[389]: <matplotlib.image.AxesImage at 0x1fc87d42cf8>
```

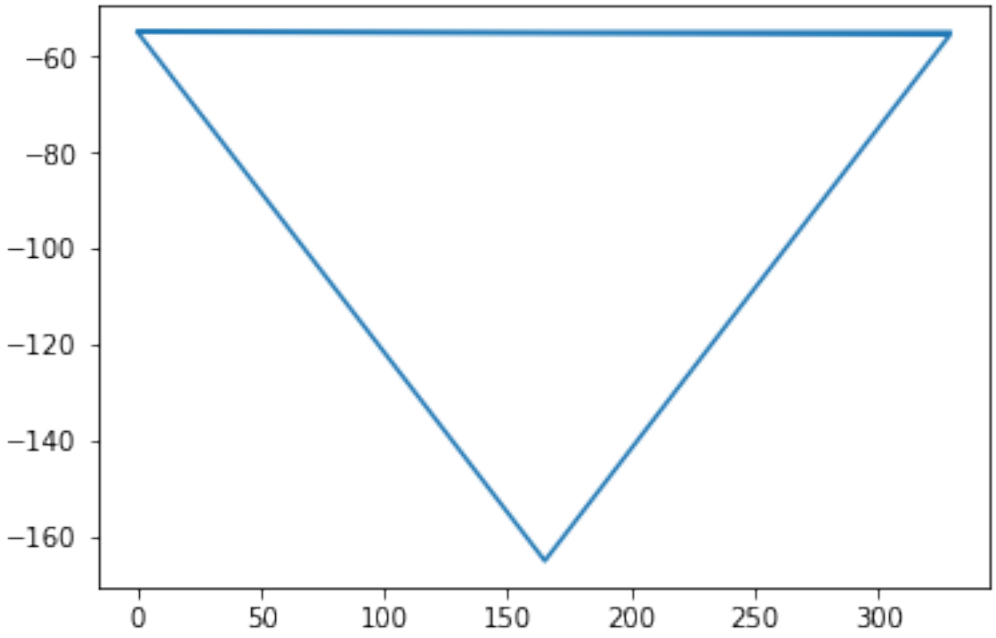


Tras aplicar la binarización y apertura para intentar separar la mariposa del fondo, obtenemos la imagen anterior. Vemos como hemos conseguido separar casi por completo el borde de la mariposa.

3.1.1 Iniciación Snake - Mariposa

Vamos a utilizar un triángulo invertido, para ello vamos a hacer un bucle en el que se va dibujando una recta y en el que hay un cambio de pendiente cuando i = mend/2 , siendo mend el punto final

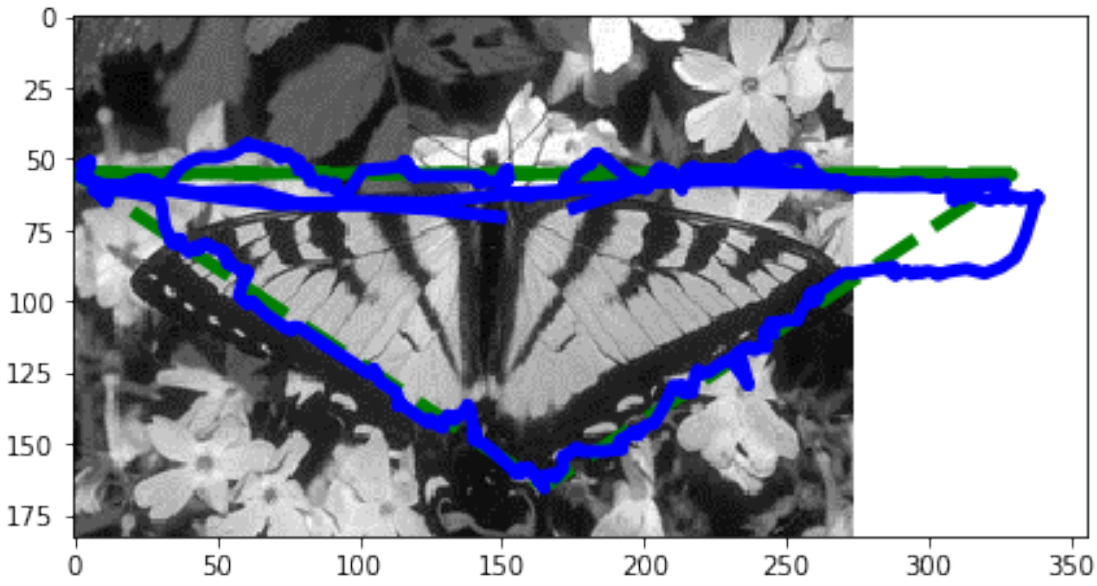
In [390]:
```
# Iniciación del snake

x0 = 0
y0 = 55

mend = 330
nup = 400
h = 110

m1 = 2*h/mend
m2 = -m1
a = -1

x = np.empty(nup)
y = x.copy()
b = np.zeros(nup)

# Creamos el triangulo
for i in np.arange(0, mend, mend/nup):
    a = a + 1
    x[a] = i + x0
    if (i < mend/2):
        y[a] = m1*i + y0
    else :
        y[a] = m2*i + 2*h + y0

    b[a] = y0

xf = np.append(x,x)
yf = np.append(y,b)

# Creamos el trinagulo
init = np.array([xf,yf]).T

#lo dibujamos
plt.plot(xf,-yf)
plt.show()
```



3.1.2 Active contour Model - Mariposa

Tras varios intentos, los parámetros que mas se ajustan son los siguientes.

```
In [392]: # Realizamos el active contouring model ACM
          malpha=0.003
          mbeta = 0.1
          mgamma = 0.003

          snake = active_contour(img, init, alpha = malpha, beta = mbeta, gamma= mgamma)

          #Lo dibujamos
          fig,ax = plt.subplots(figsize=(7,7))
          ax.imshow(im, cmap = plt.cm.gray)
          ax.plot(init[:,0],init[:,1], '--g', lw =5)
          ax.plot(snake[:,0],snake[:,1], '-b', lw =5)

          plt.show()
```



Vemos como la parte inferior se ajusta bien a la mariposa pero la parte superior no tanto.

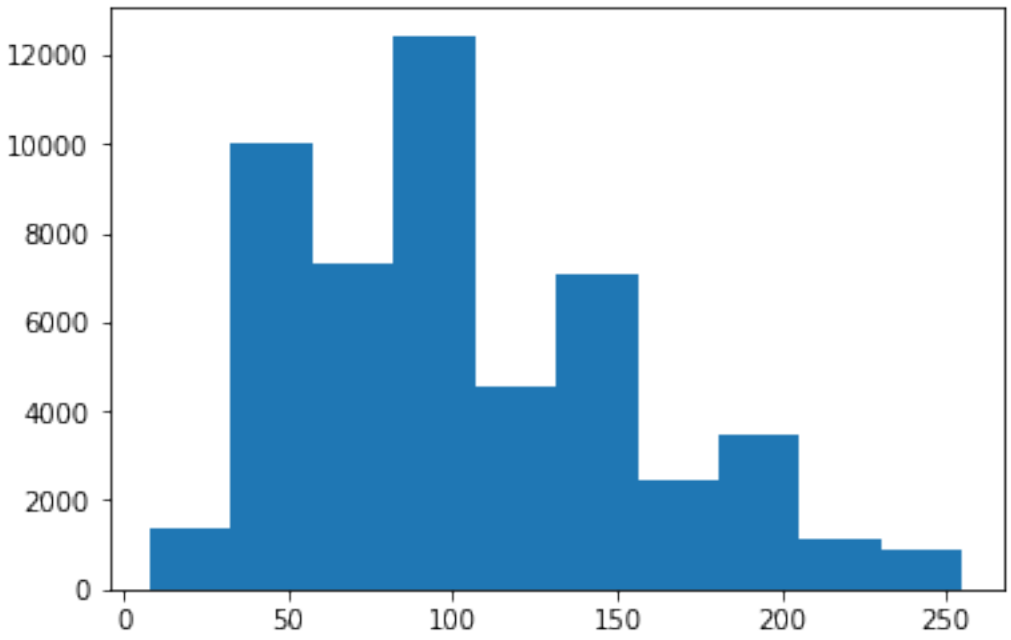
Al elegir una forma triangular ajustamos bien la parte inferior pero no la superior de la mariposa al tener ser mas concava.

Cabe aclarar que aunque hayamos superpuesto la imagen inicial al contorno, la imagen a la que hemos calculado el algortimo es la imagen binaria que hemos mostrado anteriormente.

3.2 Buho

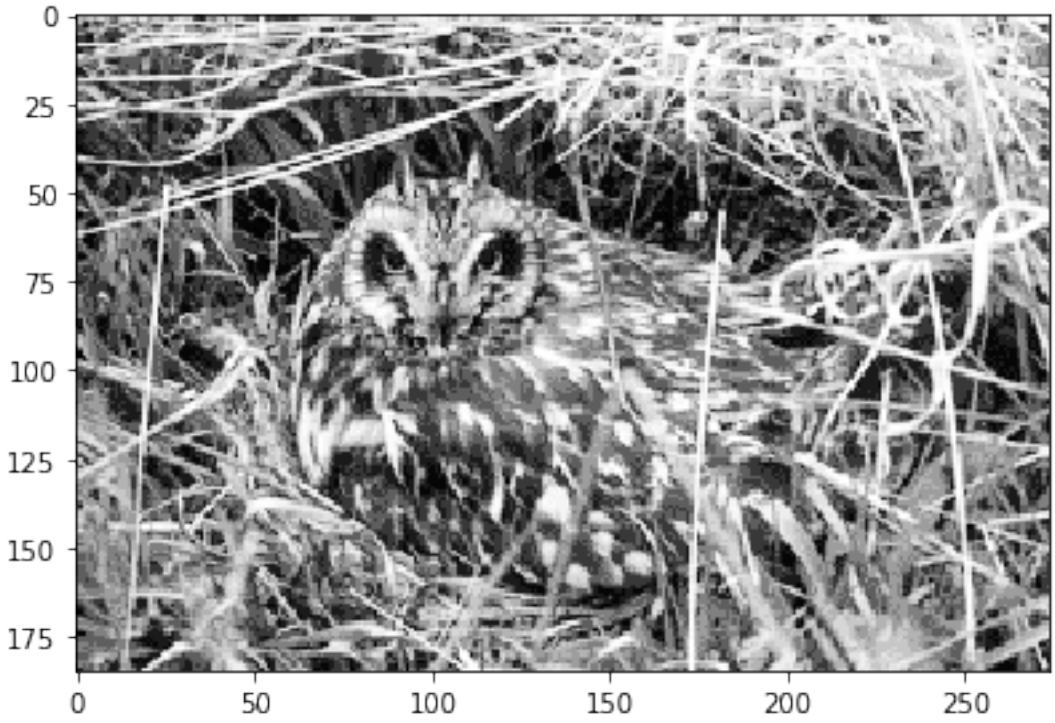
El buho representa una gran dificultad al ser el fondo bastante uniforme y estar camuflado entre las ramas. Como unica peculiaridad podemos aprovechar las direcciones de las ramas verticales y horizontales en contraposicion a las formas ovaladas y circulares de las plumas del buho.

```
In [393]: im = vim[1]

In [396]: # Display the histogram.
          plt.hist(im.ravel())
          plt.show()
```



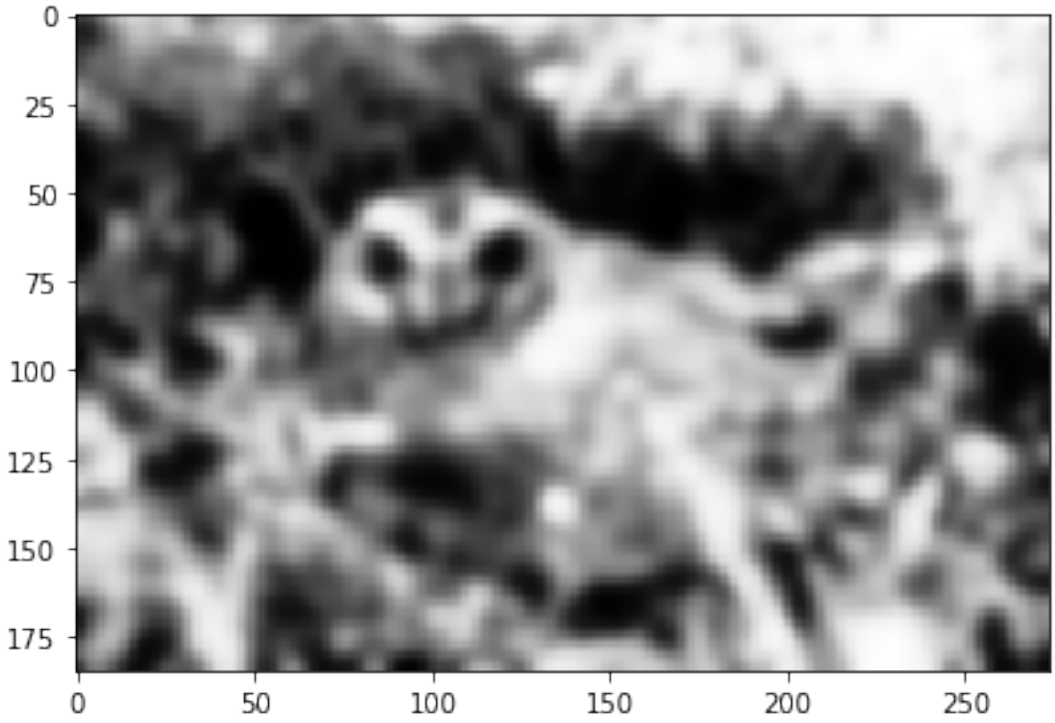
Vemos un histograma bastante uniforme, vamos a intentar ampliarlo.

```
In [397]: from skimage import exposure
          # el histograma esta muy concentrado, lo ecualizamos
          imeq = exposure.equalize_hist(im, nbins=256)
          io.imshow(imeq)

Out[397]: <matplotlib.image.AxesImage at 0x1fc87fa4748>
```



Podemos distinguir mejor el buho del fondo, aunque todavíaes bastante dificil.

```
In [398]: # vamos a intentar una erosion
          mse = np.array([[1, 0, 1, 0, 1],
                          [0, 1, 1, 1, 0],
                          [1, 1, 1, 1, 1],
                          [0, 1, 1, 1, 0],
                          [1, 0, 1, 0, 1]], dtype=np.uint8)

          # erosion
          ime = mf.erosion(im,mse)

          #filtro gausiano
          img = gaussian(ime,3)

          # el histograma esta muy concentrado, lo ecualizamos
          imeq = exposure.equalize_hist(img, nbins=256)
          io.imshow(imeq)

Out[398]: <matplotlib.image.AxesImage at 0x1fc88fd98d0>
```

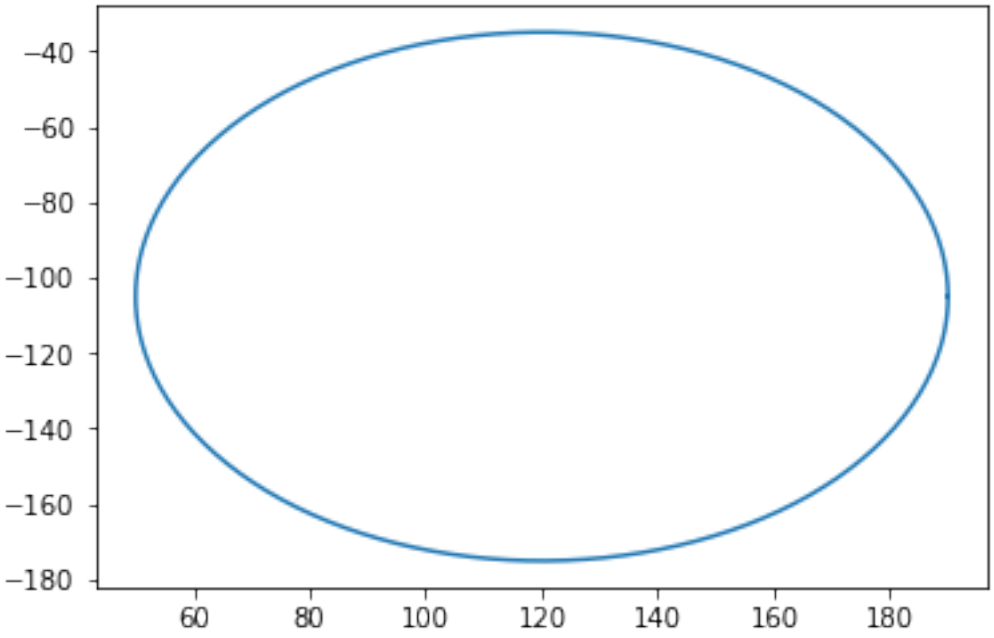


Tras distintos tratamientos e intentos, la imagen candidata ha sido esta, aunque podría mejorarse.

3.2.1 Iniciación Snake - Búho

Vamos a aprochar la forma casi circular del buho

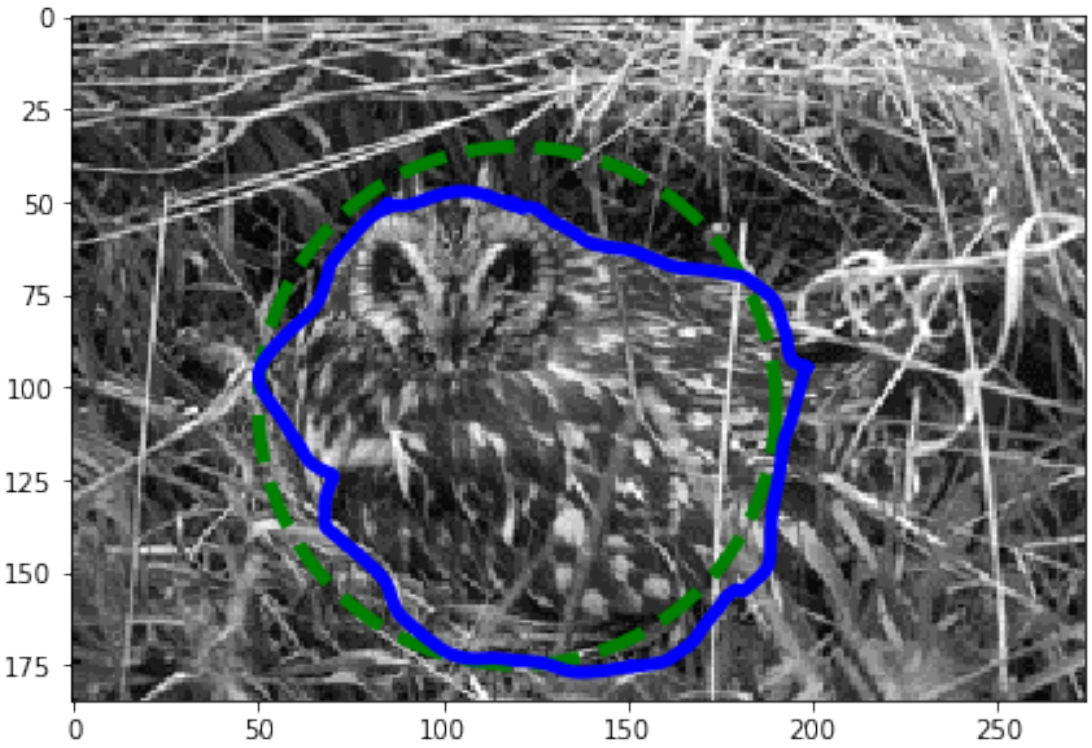
```
In [411]: x0 = 120
          y0 = 105
          r = 70

          t = np.linspace(0,2*np.pi,400)
          x = x0 + r*np.cos(t)
          y = y0 + r*np.sin(t)

          init = np.array([x,y]).T

          plt.plot(x,-y)
          plt.show()
```



3.2.2 Active contour Model - Búho

Tras varios intentos, los parámetros que mas se ajustan son los siguientes.

```
In [412]: # Realizamos el active contouring model ACM
          malpha=10*0.003
          mbeta = 0.01*0.1
          mgamma = 0.003

          snake = active_contour(imeq, init, alpha = malpha, beta = mbeta, gamma= mgamma)

          # Dibujamos el resultado
          fig,ax = plt.subplots(figsize=(7,7))
          ax.imshow(im, cmap = plt.cm.gray)
          ax.plot(init[:,0],init[:,1], '--g', lw =5)
          ax.plot(snake[:,0],snake[:,1], '-b', lw =5)

          plt.show()
```



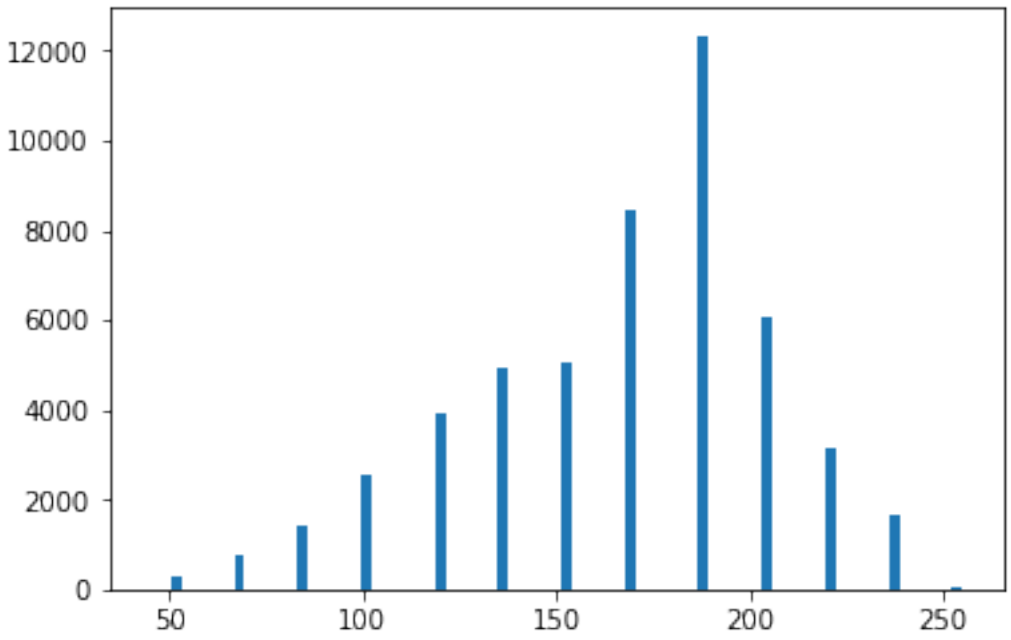
Conseguimos adaptarnos satisfactoriamente al buho, aunque con algunos errores.

4 Caballo

El caballo a simple vista parece el mas facil de los tres, ya que tiene bastante contraste y zonas diferenciadas, pero tiene la dificultad de que el caballo en sí es una forma bastante no uniforme.

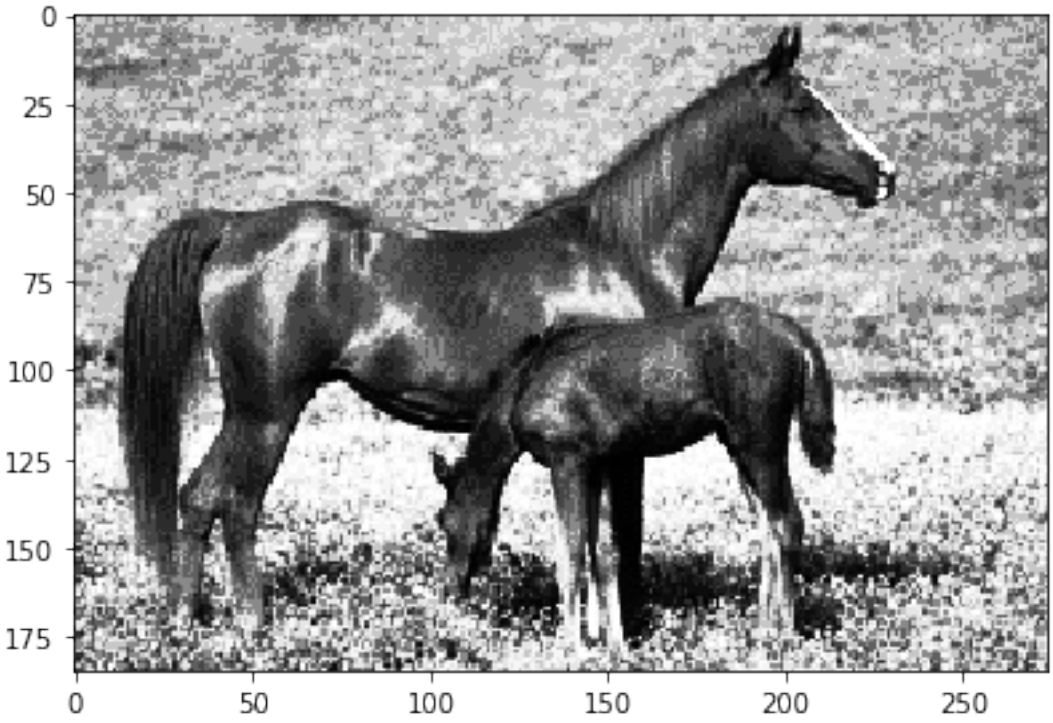
```
In [453]: im = vim[2]
```

```
In [454]: # Dibujamos el histograma.
          plt.hist(im.ravel(),bins='auto')
          plt.show()
```

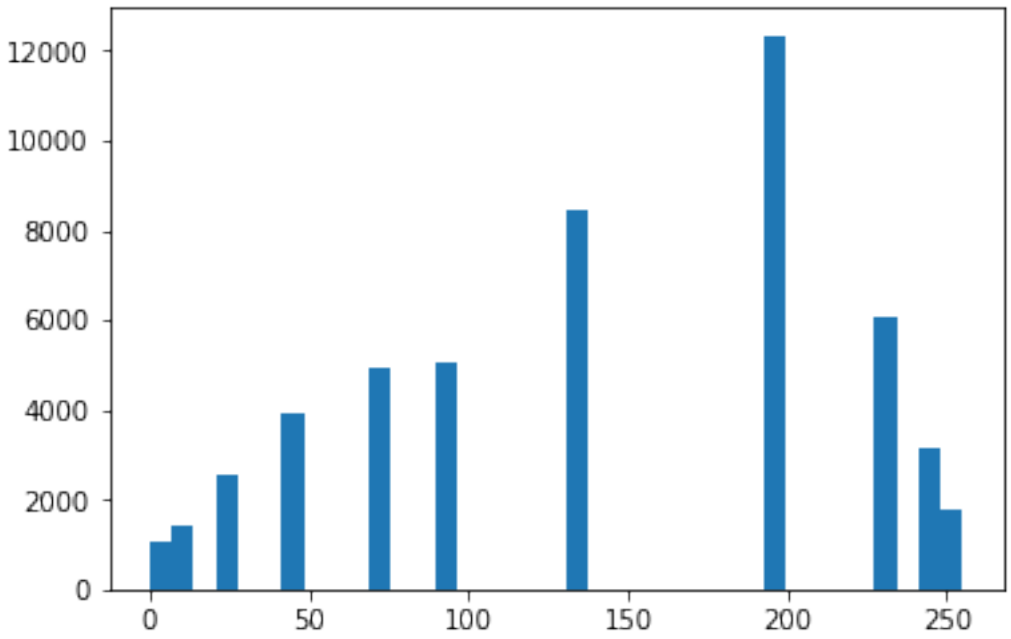


Efectivamente vemos un histograma bastante diferenciado, vamos primero a separarlo y suivar la imagen. La funcón a continuación escala la imagen de 0 a 255 para conservarlo tras el filtrado.

```
In [416]: # -------- Funcion escalado de imagen [0,255] -----------
          def mescale(arr):
              new_arr = ((arr - arr.min()) * (1/(arr.max() - arr.min()) * 255)).astype('uint8')
              return new_arr

In [417]: # ecualizamos
          imeq = mescale(exposure.equalize_hist(im, nbins=256))
          io.imshow(imeq)

Out[417]: <matplotlib.image.AxesImage at 0x1fc89746828>
```



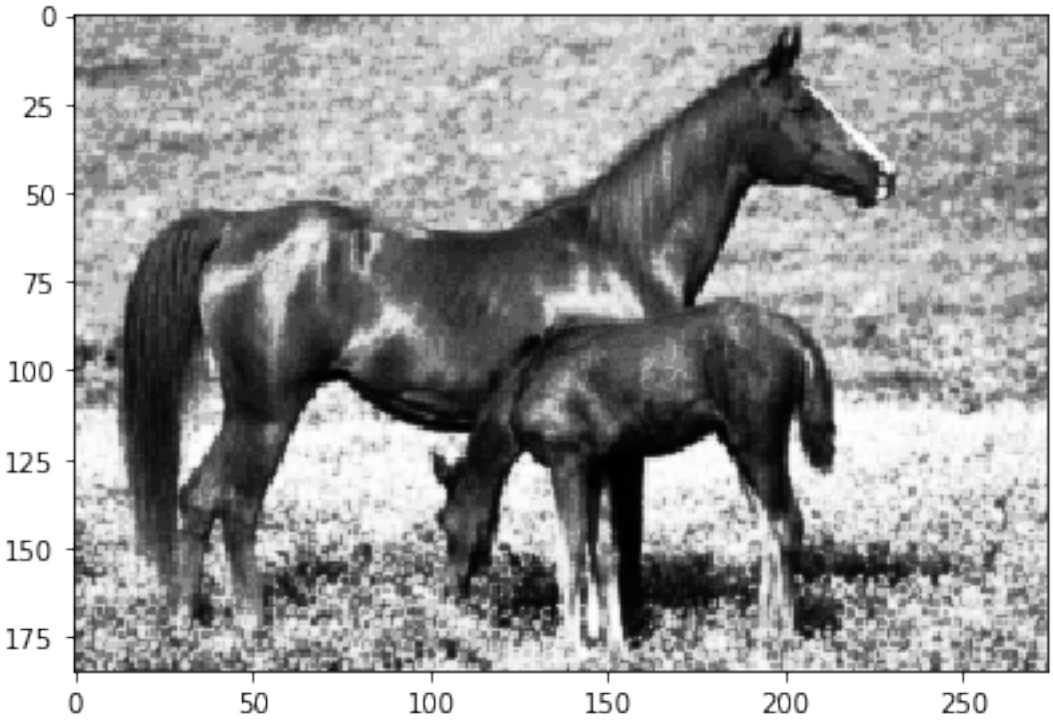
In [418]:
```
# Dibujamos el histograma.
plt.hist(imeq.ravel(),bins='auto')
plt.show()
```



Podemos separar bastante el fondo del objetivo

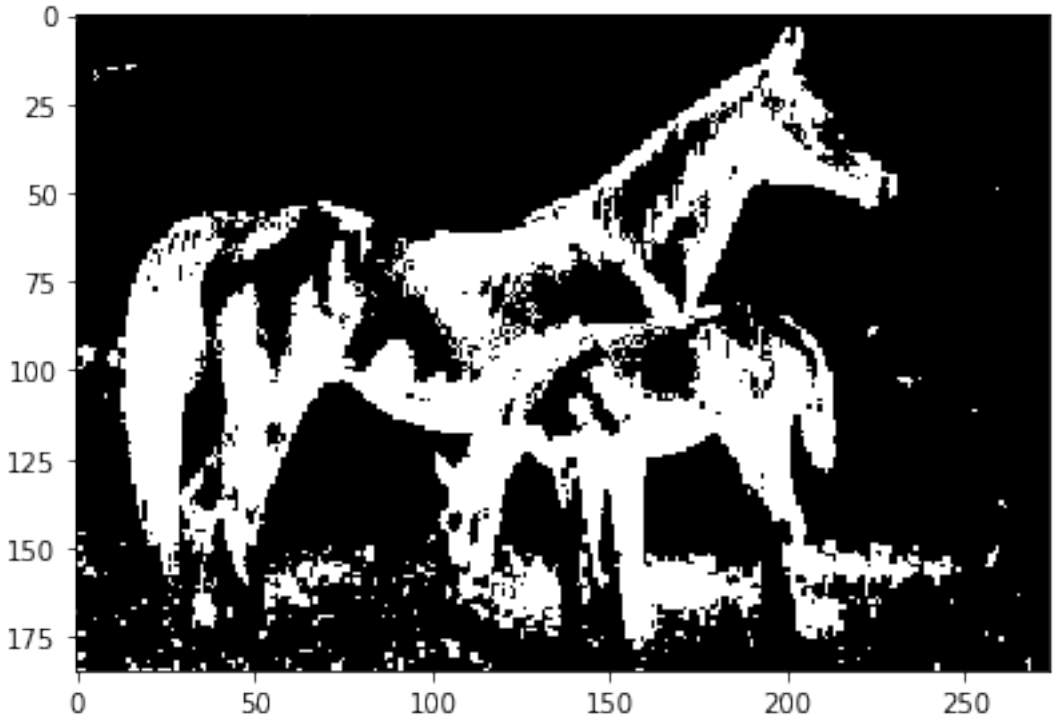
```
In [419]: #filtro gaussiano
          imm = mescale(gaussian(imeq,0.5))
          io.imshow(imm)
```

```
Out[419]: <matplotlib.image.AxesImage at 0x1fc89855400>
```




```
In [420]: # binarizamos
          imb = imm < 80
          io.imshow(imb)
```

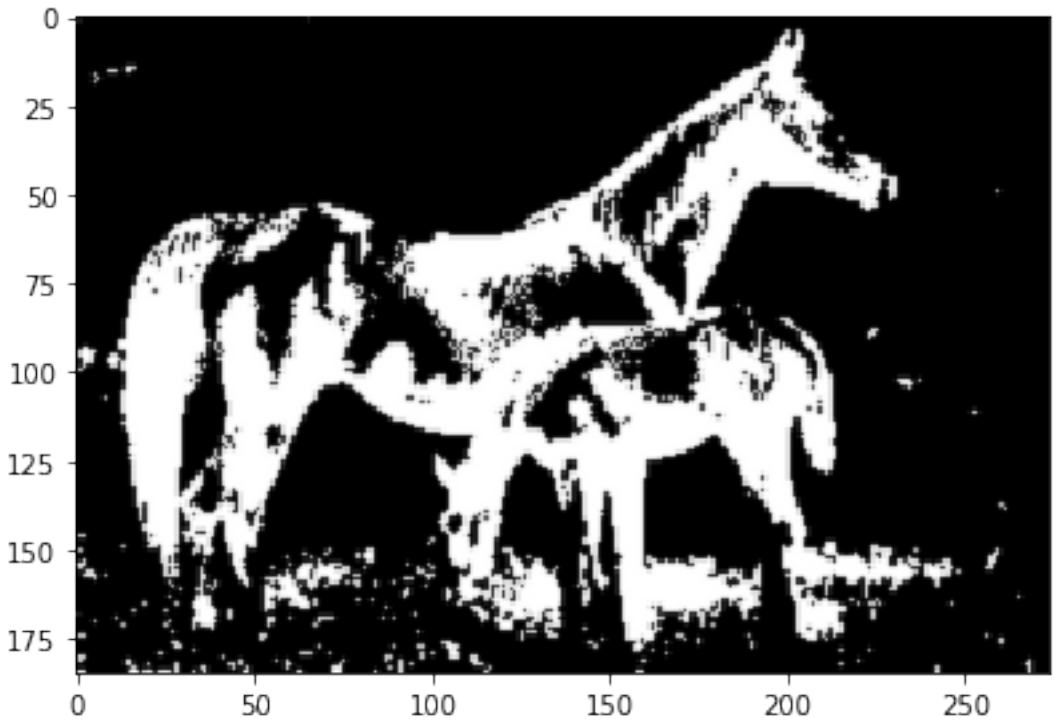
```
Out[420]: <matplotlib.image.AxesImage at 0x1fc898b4940>
```



Vemos una pequeña dificultad añadida, el suelo es bastante similar al tono del caballo, por lo que dificultará la busqueda.

A continuación, aplicamos una apertura para mejorar la separación y suavizamos con un filtro Gaussiano.

```
In [512]: # apertura
          se = mf.rectangle(1,1)
          ima = mf.binary_opening(imb,se)
          io.imshow(ima)

          #filtro gaussiano
          img = mescale(gaussian(ima,0.5))
          io.imshow(img)

Out[512]: <matplotlib.image.AxesImage at 0x1fcd3bb2d68>
```



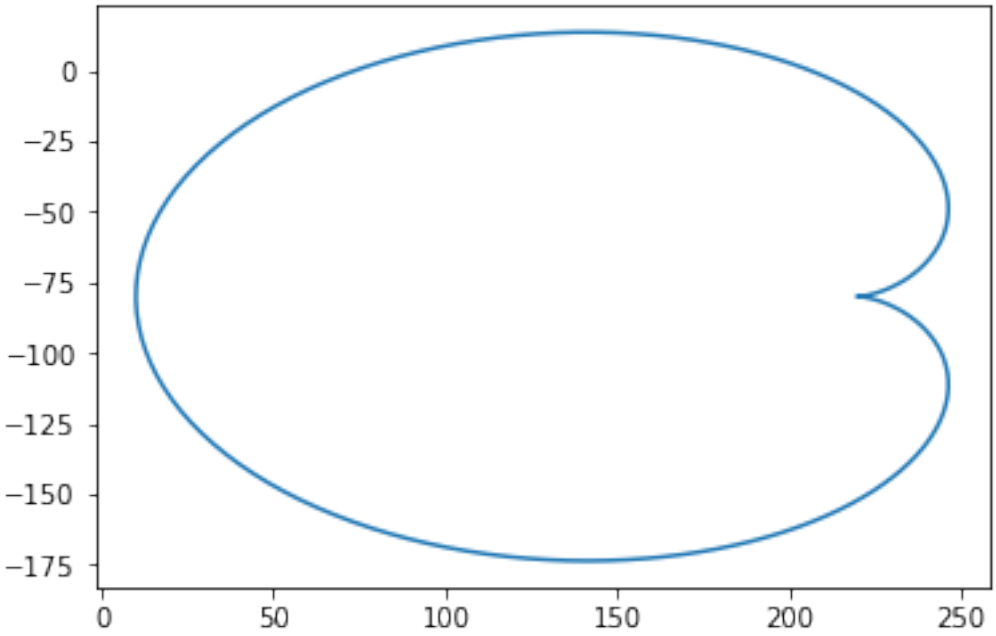
Hemos conseguido distinguir bastante bien el objetivo del fondo y reducir el ruido que teníamos previamente.

4.0.1 Iniciación Snake - Caballo

Vamos a optar por una cardiode para intentar coger bien la forma de la cara del caballo

In [505]:
```
# Calculamos el paramtro inicial

x0 = 110
y0 = 80
a = 150
b = 90


t = np.linspace(0,2*np.pi,400)

# corrdenadas parametricas de una cardiode
x = 0.7*a*(1-np.cos(t))*np.cos(t) + 2*x0
y = 0.8*b*(1-np.cos(t))*np.sin(t) + y0

#  Initialisation coordinates of snake
init = np.array([x,y]).T
```

In [506]:
```
plt.plot(x,-y)
plt.show()
```



alpha : float, optional Snake length shape parameter. Higher values makes snake contract faster.

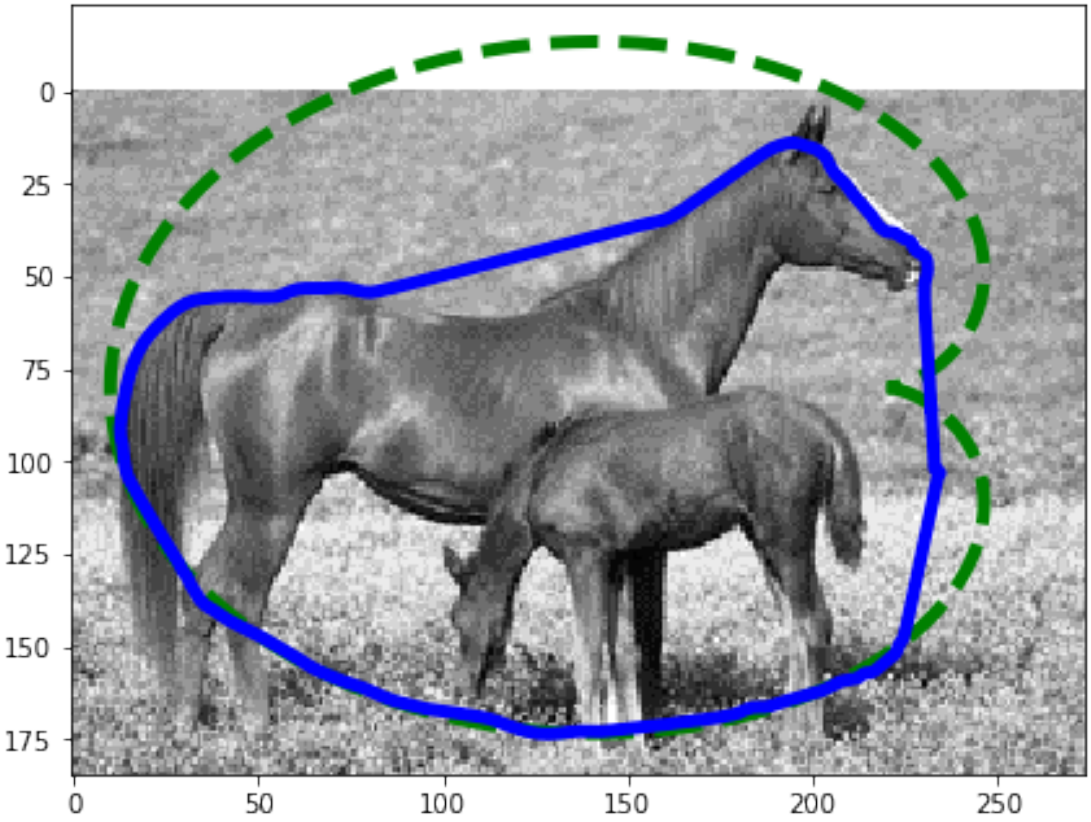
beta : float, optional Snake smoothness shape parameter. Higher values makes snake smoother.

gamma : float, optional Explicit time stepping parameter.

4.0.2 Active Contour Model - Caballo

Tras varios intentos, los parámetros que mas se ajustan son los siguientes.

```
In [507]: # Realizamos el active contouring model ACM
          malpha= 200*0.003
          mbeta = 1*0.1
          mgamma = 50*0.003

          snake = active_contour(img, init, alpha = malpha, beta = mbeta, gamma= mgamma)

          # Dibujamos el resultado
          fig,ax = plt.subplots(figsize=(7,7))
          ax.imshow(im, cmap = plt.cm.gray)
          ax.plot(init[:,0],init[:,1], '--g', lw =5)
          ax.plot(snake[:,0],snake[:,1], '-b', lw =5)
          plt.show()
```



En este caso, pese a que parecía simle la diferenciación, nos encontramos con las limitaciones del algortimo.

El ACM no converge en las concavidades y esta forma tiene numerosas concavidades, y a parte la forma inicial dista bastante de la forma de un caballo.

5 Conclusiones

A modo de conclusión, cabe destacar:

- El ACM es un algoritmo eficaz pero es poco robusto. Tenemos que cumplir varios requisitos, como que tengamos una alta diferenciación, contornos suaves, sin concavidades y una inicialización similar al objeto en cuestión.
- El hecho de trabajar con imágenes reales que impidan cumplir con los requisitos citados previamente hace que el algoritmo sea ineficaz. Cabe destacar como por ejemplo en el caso del búho que para el ojo humano era mas dificil que el caballo, el algoritmo tiene buenos resultados debido a la forma ovalada de este.
- Hay que realizar buen preprocesado para que el algoritmo no diverja o confunda regiones, de momento empleando solo suavizados o métodos morfológicos no hemos podido obtener resultados muy satisfactorios.

- En el caso de formas complicadas, como el caballo, podríamos emplear formas genéricas de caballo para elegir como inicialización. Hemos intentado elegir la inicialización de esa forma pero el paso de la forma genérica a curva paramétrica no era evidente y al final no ha sido posible.
- En resumen, hemos visto que es un algoritmo sencillo y eficaz si se respetan sus condiciones. Podría tener muchas aplicaciones como la detección de rostros, alimentos como fruta o verduras en una cadena de producción, detección de monedas, pero no sería eficaz para formas complejas y con concavidades, como es el caso del caballo.