In [1]:
```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
import time
import sys
```

In [2]:
```
accumulator_kernel = 20
gaussian_kernel = 3
max_detection_radius = 100
intensity_threshold=60
```

In [3]:

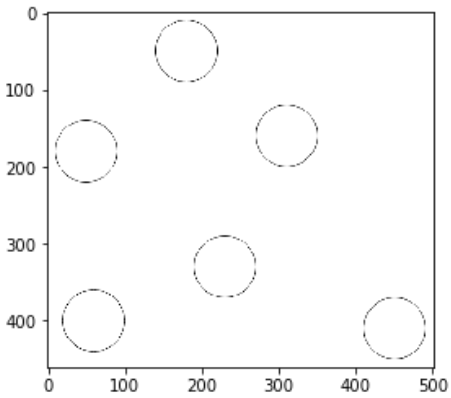
```
def drawPixel(x, y, pixelData, width, height):
    if x < height and y < width:
        pixelData[x,y] += 1
    return pixelData


def paint_pixels(x, y, image):
    image[x,y,1] = 255
    image[x,y,0] = 0
    image[x,y,2] = 0
    return image

def draw_circle(x0, y0, radius, image):
    x = radius
    y = 0
    decisionOver2 = 1 - x   # Decision criterion divided by 2 evaluated at
x=r, y=0

    while x >= y:
        image = paint_pixels(x + x0, y + y0, image)
        image = paint_pixels(y + x0, x + y0, image)
        image = paint_pixels(-x + x0, y + y0, image)
        image = paint_pixels(-y + x0, x + y0, image)
        image = paint_pixels(-x + x0, -y + y0, image)
        image = paint_pixels(-y + x0, -x + y0, image)
        image = paint_pixels(x + x0, -y + y0, image)
        image = paint_pixels(y + x0, -x + y0, image)
        y+=1
        if decisionOver2 <= 0:
            decisionOver2 += 2 * y + 1 # Change in decision criterion for y
-> y+1
        else:
            x-=1
            decisionOver2 += 2 * (y - x) + 1 # Change for y -> y+1, x -> x-1
    return image

def accumulator_data(x0, y0, radius, pixelData, width, height):
    ''' This is the implementation of Midpoint Circle Algorithm. Refer the r
eport for more details. '''
    x = radius
    y = 0
    decisionOver2 = 1 - x   # Decision criterion divided by 2 evaluated at
x=r, y=0

    while x >= y:
        pixelData = drawPixel(x + x0, y + y0, pixelData, width, height)
        pixelData = drawPixel(y + x0, x + y0, pixelData, width, height)
        pixelData = drawPixel(-x + x0, y + y0, pixelData, width, height)
        pixelData = drawPixel(-y + x0, x + y0, pixelData, width, height)
        pixelData = drawPixel(-x + x0, -y + y0, pixelData, width, height)
        pixelData = drawPixel(-y + x0, -x + y0, pixelData, width, height)
        pixelData = drawPixel(x + x0, -y + y0, pixelData, width, height)
        pixelData = drawPixel(y + x0, -x + y0, pixelData, width, height)
        y+=1
        if decisionOver2 <= 0:
            decisionOver2 += 2 * y + 1 # Change in decision criterion for y
-> y+1
        else:
            x-=1
            decisionOver2 += 2 * (y - x) + 1 # Change for y -> y+1, x -> x-1
    return pixelData

def get_edge_locations(edged_image):
    edges = np.where(edged_image==255)
    return edges

def get_max_possible_radius(edges, max_detection_radius):
    xmin = min(edges[0])
```

# Normal scenario

In [4]:
```
src = cv.imread(cv.samples.findFile('a.png'))
```

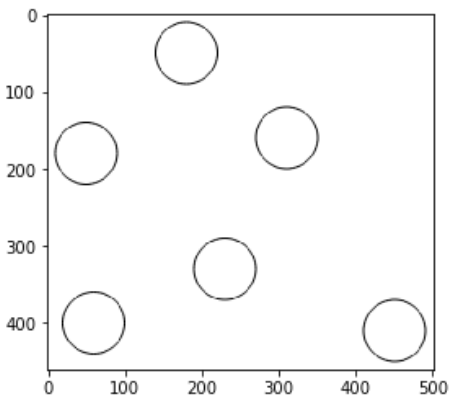
In [5]:
```
gray_image = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
plt.imshow(gray_image,cmap='gray')
gray_image.shape
```

Out[5]: (461, 501)

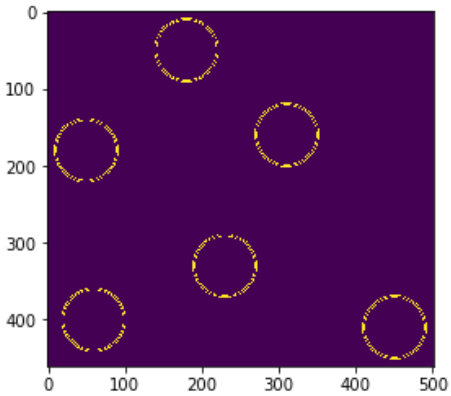



In [6]:
```
output = src.copy()
blur_image = cv.GaussianBlur(gray_image,(gaussian_kernel,gaussian_kernel),0)
plt.imshow(blur_image,cmap='gray')
```

Out[6]: <matplotlib.image.AxesImage at 0x7f570c35f5f8>



In [7]:
```
edged_image = cv.Canny(blur_image,3,3)
plt.imshow(edged_image)
height,width = edged_image.shape
edges = get_edge_locations(edged_image)
```



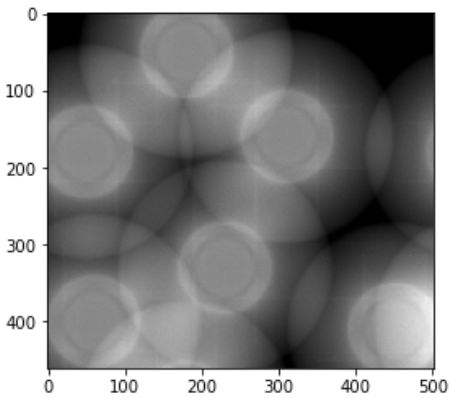

In [8]:
```
max_radius = get_max_possible_radius(edges, max_detection_radius)
max_radius
```

Out[8]: 100

In [9]:
```
acc_array = construct_accumulator_array(edges, width, height, max_radius)
```

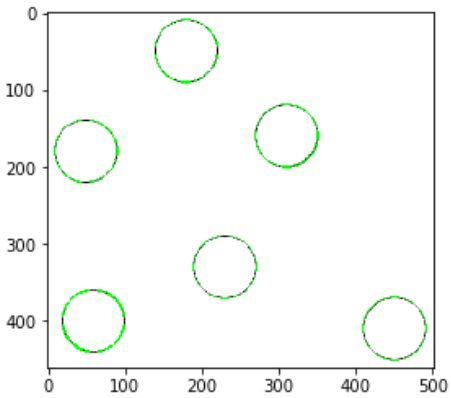
In [10]:
```
plt.imshow(np.sum(np.copy(acc_array),axis=2),cmap='gray')
```

Out[10]: <matplotlib.image.AxesImage at 0x7f570c2a4470>



In [11]:
```
output = threshold_accumulator_plot_circles(output, acc_array, max_radius, w
idth, height, intensity_threshold, accumulator_kernel)
plt.imshow(output,cmap='gray')
```

Out[11]:
```
<matplotlib.image.AxesImage at 0x7f570c27d3c8>
```



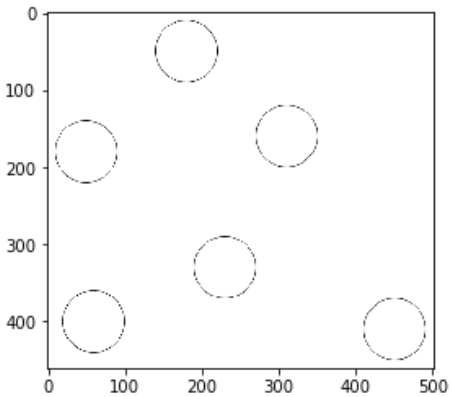

W tym scenariuszu transformata została skalibrowana, również zweryfikowałem czy kod działa poprawnie. W akumulatorze transformaty widzimy że najwięcej głosów ozyskały miejsca pokrywające się z położeniem okręgów. Widać również jak transformata `szuka` okręgów o większym jak i mniejszym promieniu. Zostało zastosowane delikatne rozmycie aby okręgi były trochę grubsze.

# Strong Blur scenario

In [12]:
```
src = cv.imread(cv.samples.findFile('a.png'))
```

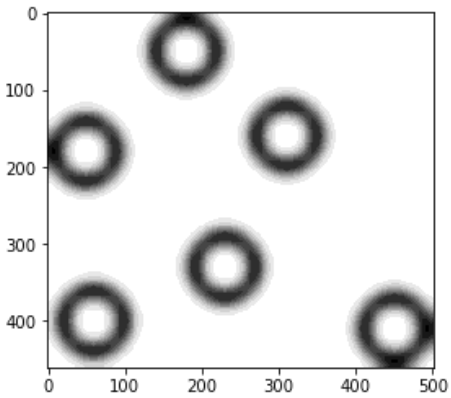
In [13]:
```
gray_image = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
plt.imshow(gray_image,cmap='gray')
gray_image.shape
```

Out[13]:
```
(461, 501)
```



In [14]:
```
output = src.copy()
blur_image = cv.GaussianBlur(gray_image,(63,63),0)
plt.imshow(blur_image,cmap='gray')
```

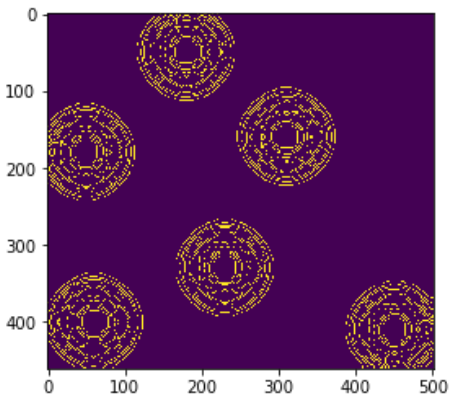
Out[14]:
```
<matplotlib.image.AxesImage at 0x7f570c1b1518>
```




Z tak mocno rozmytym obrazem trudno jest nawet wykryć krawędzie

In [15]:
```
edged_image = cv.Canny(blur_image,3,3)
plt.imshow(edged_image)
height,width = edged_image.shape
edges = get_edge_locations(edged_image)
plt.imshow(edged_image)
```

Out[15]:
```
<matplotlib.image.AxesImage at 0x7f570c15d5f8>
```



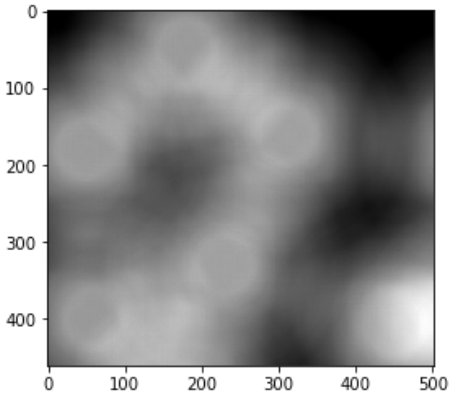

In [16]:
```
max_radius = get_max_possible_radius(edges, max_detection_radius)
max_radius
```

Out[16]:
```
100
```

In [17]:
```
acc_array = construct_accumulator_array(edges, width, height, max_radius)
```

In [18]:
```
plt.imshow(np.sum(np.copy(acc_array),axis=2),cmap='gray')
```

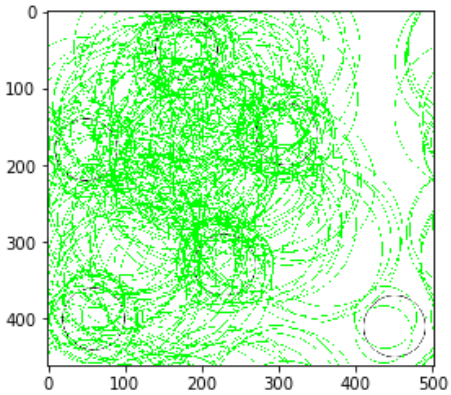
Out[18]:
```
<matplotlib.image.AxesImage at 0x7f570c0eaa58>
```



In [19]:
```
output = threshold_accumulator_plot_circles(output, acc_array, max_radius, w
idth, height, intensity_threshold, accumulator_kernel)
plt.imshow(output,cmap='gray')
```

```
index 519 is out of bounds for axis 1 with size 501
index 539 is out of bounds for axis 1 with size 501
index 515 is out of bounds for axis 1 with size 501
index 515 is out of bounds for axis 1 with size 501
index 515 is out of bounds for axis 1 with size 501
index 515 is out of bounds for axis 1 with size 501
index 555 is out of bounds for axis 1 with size 501
index 562 is out of bounds for axis 1 with size 501
index 592 is out of bounds for axis 1 with size 501
index 548 is out of bounds for axis 1 with size 501
index 558 is out of bounds for axis 1 with size 501
index 584 is out of bounds for axis 1 with size 501
index 546 is out of bounds for axis 1 with size 501
index 575 is out of bounds for axis 1 with size 501
index 586 is out of bounds for axis 1 with size 501
index 537 is out of bounds for axis 1 with size 501
index 594 is out of bounds for axis 1 with size 501
index 550 is out of bounds for axis 1 with size 501
index 550 is out of bounds for axis 1 with size 501
index 594 is out of bounds for axis 1 with size 501
index 473 is out of bounds for axis 0 with size 461
index 473 is out of bounds for axis 0 with size 461
index 476 is out of bounds for axis 0 with size 461
index 475 is out of bounds for axis 0 with size 461
index 476 is out of bounds for axis 0 with size 461
index 511 is out of bounds for axis 1 with size 501
index 540 is out of bounds for axis 1 with size 501
index 548 is out of bounds for axis 1 with size 501
index 588 is out of bounds for axis 1 with size 501
index 497 is out of bounds for axis 0 with size 461
index 496 is out of bounds for axis 0 with size 461
index 479 is out of bounds for axis 0 with size 461
index 480 is out of bounds for axis 0 with size 461
index 475 is out of bounds for axis 0 with size 461
index 479 is out of bounds for axis 0 with size 461
index 496 is out of bounds for axis 0 with size 461
index 502 is out of bounds for axis 1 with size 501
index 477 is out of bounds for axis 0 with size 461
index 546 is out of bounds for axis 1 with size 501
index 494 is out of bounds for axis 0 with size 461
index 493 is out of bounds for axis 0 with size 461
index 497 is out of bounds for axis 0 with size 461
index 499 is out of bounds for axis 0 with size 461
index 496 is out of bounds for axis 0 with size 461
index 496 is out of bounds for axis 0 with size 461
index 491 is out of bounds for axis 0 with size 461
index 498 is out of bounds for axis 0 with size 461
index 508 is out of bounds for axis 0 with size 461
index 506 is out of bounds for axis 0 with size 461
index 462 is out of bounds for axis 0 with size 461
index 463 is out of bounds for axis 0 with size 461
index 509 is out of bounds for axis 1 with size 501
index 507 is out of bounds for axis 0 with size 461
index 524 is out of bounds for axis 0 with size 461
index 461 is out of bounds for axis 0 with size 461
index 512 is out of bounds for axis 0 with size 461
index 518 is out of bounds for axis 0 with size 461
index 523 is out of bounds for axis 0 with size 461
index 529 is out of bounds for axis 0 with size 461
index 530 is out of bounds for axis 0 with size 461
index 525 is out of bounds for axis 0 with size 461
index 531 is out of bounds for axis 0 with size 461
index 534 is out of bounds for axis 0 with size 461
index 519 is out of bounds for axis 0 with size 461
index 490 is out of bounds for axis 0 with size 461
index 498 is out of bounds for axis 0 with size 461
index 527 is out of bounds for axis 0 with size 461
index 542 is out of bounds for axis 0 with size 461
```

Out[19]: <matplotlib.image.AxesImage at 0x7f570c1ea550>



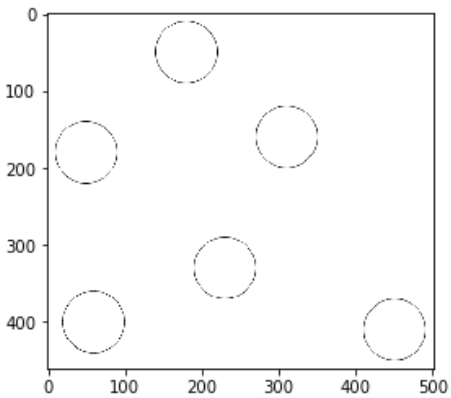

Widzimy że silne rozmycie zupełnie zmyliło transformatę. Na akumulatorze widzimy ze że co prawda obszary gdzie znajdują się okręgi otrzymują sporo głosów, natomiast są bardzo zagłuszone przez otoczenie. W tym scenariuszu musiałem również obniżyć progi gdż w przeciwnym wypadku okręgni nie były zupełnie wykrywane.

# 'Salt and Pepper' filter scenario

In [20]:
```
src = cv.imread(cv.samples.findFile('a.png'))
```

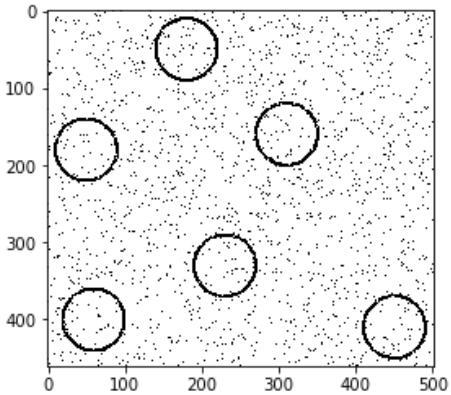
In [21]:
```
gray_image = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
plt.imshow(gray_image,cmap='gray')
gray_image.shape
```

Out[21]: (461, 501)



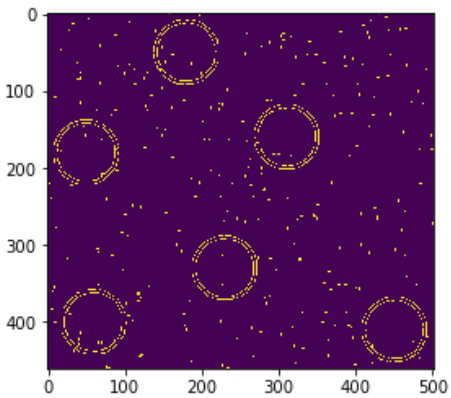
In [22]:
```
from skimage.util import random_noise
output = src.copy()
blur_image = np.uint8(random_noise(cv.GaussianBlur(gray_image,(gaussian_kern
el,gaussian_kernel),0), mode='s&p',amount=0.07))
plt.imshow(blur_image,cmap='gray')
```

Out[22]:
```
<matplotlib.image.AxesImage at 0x7f570c341ba8>
```




In [23]:
```
edged_image = cv.Canny(blur_image,3,3,3)
height,width = edged_image.shape
plt.imshow(edged_image)
edges = get_edge_locations(edged_image)
edges
```

Out[23]:
```
(array([  0,   0,   0, ..., 460, 460, 460]),
 array([ 41,  43,  46, ..., 475, 476, 500]))
```



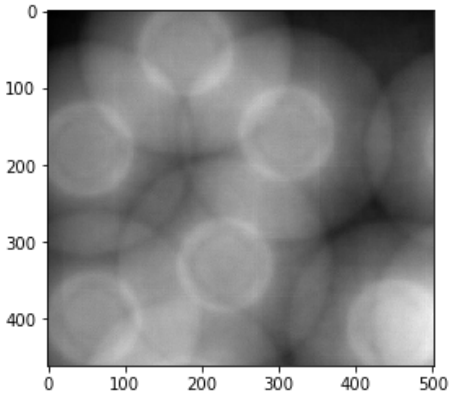

In [24]:
```
max_radius = get_max_possible_radius(edges, max_detection_radius)
max_radius
```

Out[24]:
```
100
```

In [25]:
```
acc_array = construct_accumulator_array(edges, width, height, max_radius)
```

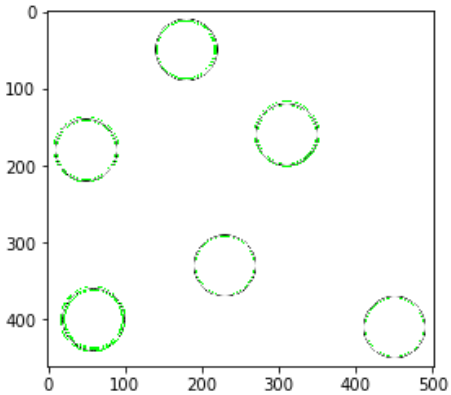
In [26]:
```
plt.imshow(np.sum(np.copy(acc_array),axis=2),cmap='gray')
```

Out[26]:
```
<matplotlib.image.AxesImage at 0x7f56f4dcea58>
```




In [27]:
```
output = threshold_accumulator_plot_circles(output, acc_array, max_radius, w
idth, height, intensity_threshold, accumulator_kernel)
plt.imshow(output,cmap='gray')
```

Out[27]:
```
<matplotlib.image.AxesImage at 0x7f56f4da5ba8>
```



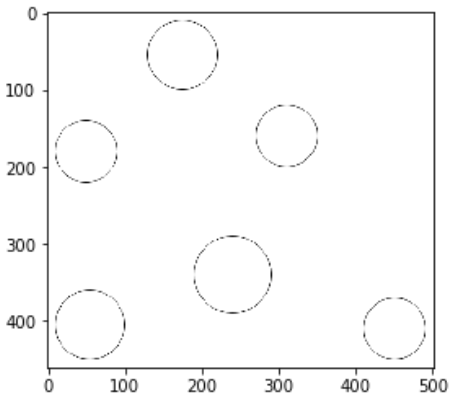

Widzimy że filtr 'salt and pepper' nie zmylił transformaty zupełnie, co prawda filtr nie był bardzo intensywny, ale udało mu się i tak zmniejszyć precyzję wykrywanych okręgów. Widzimy że zielone i czarne okręgi nie pokrywają się całkowicie.

# Different sizes scenario

In [28]:
```
src = cv.imread(cv.samples.findFile('b.png'))
```

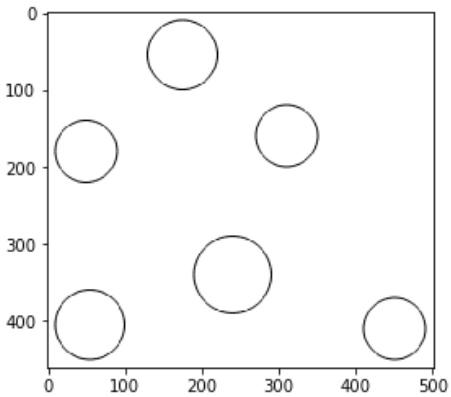
In [29]:
```
gray_image = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
plt.imshow(gray_image,cmap='gray')
gray_image.shape
```

Out[29]:
```
(461, 501)
```

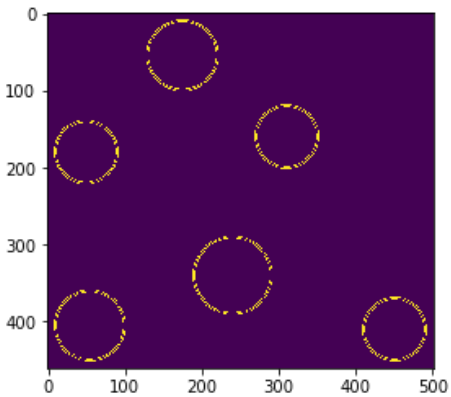



In [30]:
```
output = src.copy()
blur_image = cv.GaussianBlur(gray_image,(gaussian_kernel,gaussian_kernel),0)
plt.imshow(blur_image,cmap='gray')
```

Out[30]:
```
<matplotlib.image.AxesImage at 0x7f56f4cdf710>
```




In [31]:
```
edged_image = cv.Canny(blur_image,3,3)
plt.imshow(edged_image)
height,width = edged_image.shape
edges = get_edge_locations(edged_image)
```



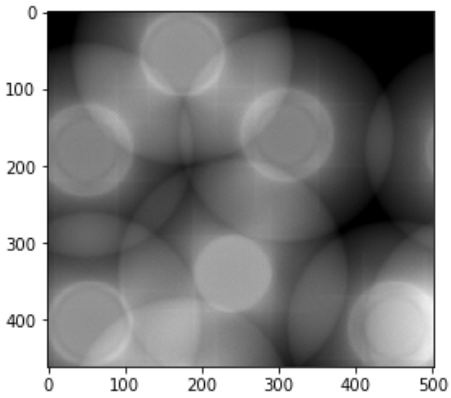
In [32]:
```
max_radius = get_max_possible_radius(edges, max_detection_radius)
max_radius
```

Out[32]: 100

In [33]:
```
acc_array = construct_accumulator_array(edges, width, height, max_radius)
```

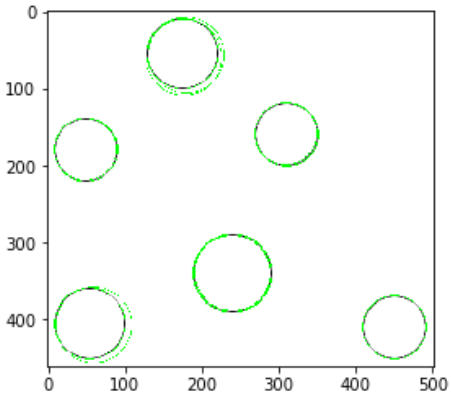
In [34]:
```
plt.imshow(np.sum(np.copy(acc_array),axis=2),cmap='gray')
```

Out[34]: <matplotlib.image.AxesImage at 0x7f56fffd1748>




In [35]:
```
output = threshold_accumulator_plot_circles(output, acc_array, max_radius, w
idth, height, intensity_threshold, accumulator_kernel)
plt.imshow(output,cmap='gray')
```

Out[35]: <matplotlib.image.AxesImage at 0x7f56fffaf630>



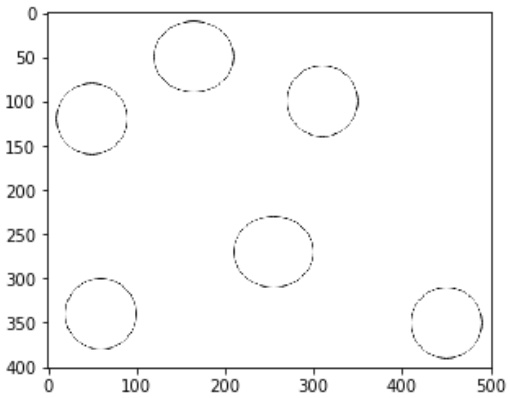

Kółka rożnej wielkości zostały wykryte natomiast zniekształciły one wyniki na sąsiadujących okręgach.

# Circles with Elipses scenario

In [36]:
```
src = cv.imread(cv.samples.findFile('c.png'))
```

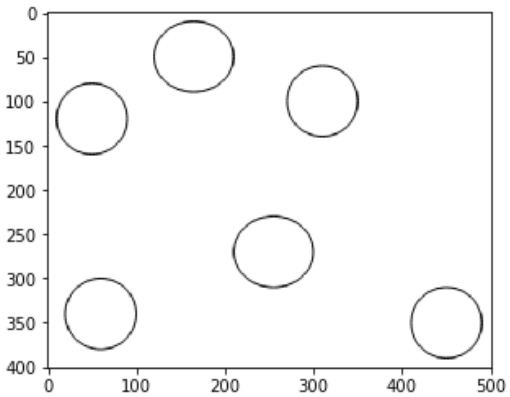
In [37]:
```
gray_image = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
plt.imshow(gray_image,cmap='gray')
gray_image.shape
```

Out[37]:
```
(401, 501)
```

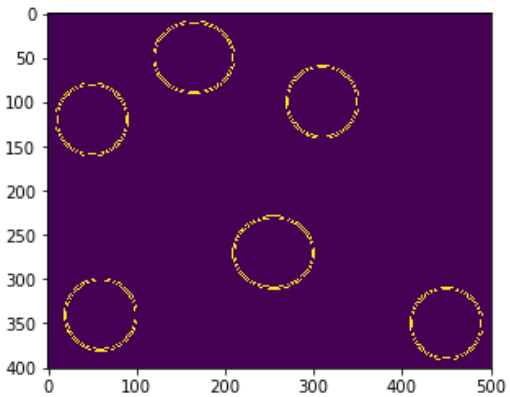



In [38]:
```
output = src.copy()
blur_image = cv.GaussianBlur(gray_image,(gaussian_kernel,gaussian_kernel),0)
plt.imshow(blur_image,cmap='gray')
```

Out[38]:
```
<matplotlib.image.AxesImage at 0x7f570c0195c0>
```




In [39]:
```
edged_image = cv.Canny(blur_image,3,3)
plt.imshow(edged_image)
height,width = edged_image.shape
edges = get_edge_locations(edged_image)
```



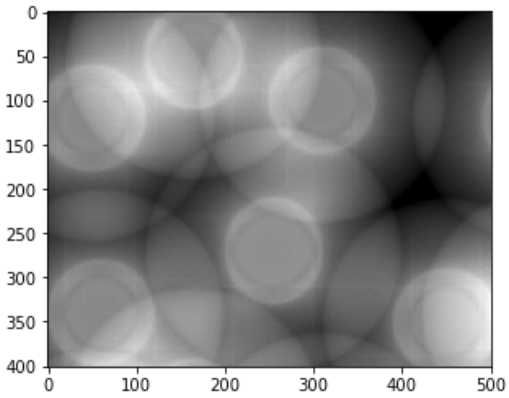
In [40]:
```
max_radius = get_max_possible_radius(edges, max_detection_radius)
max_radius
```

Out[40]: 100

In [41]:
```
acc_array = construct_accumulator_array(edges, width, height, max_radius)
```

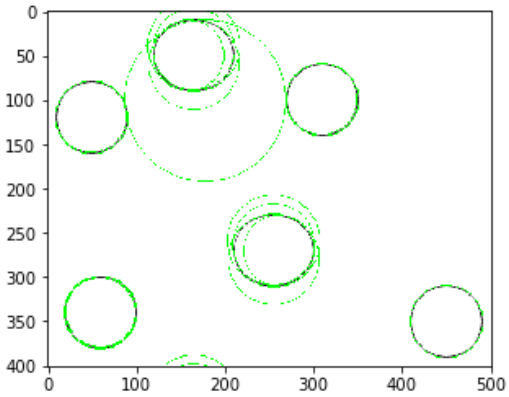
In [42]:
```
plt.imshow(np.sum(np.copy(acc_array),axis=2),cmap='gray')
```

Out[42]: <matplotlib.image.AxesImage at 0x7f56f4cab940>




In [43]:
```
output = threshold_accumulator_plot_circles(output, acc_array, max_radius, w
idth, height, intensity_threshold, accumulator_kernel)
plt.imshow(output,cmap='gray')
```

Out[43]: <matplotlib.image.AxesImage at 0x7f56f55796a0>



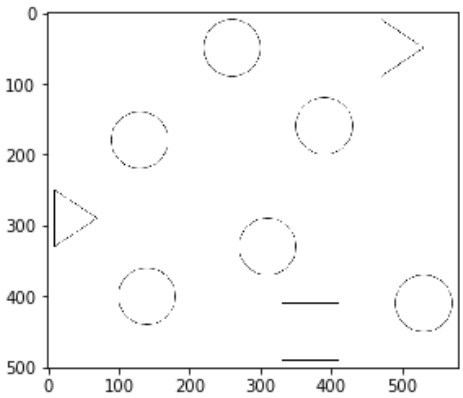

Widzimy że elipsy są dużo trudniejsze niż okręgi o lekko różnych kształtac, tym razem elipsy dostały po kilka dopasowań, na każdą elipsę zostało wykryte kilka kółek, co nie dziwi że wykryte okręgi są styczne do elips.

# With non circular shapes scenario

In [44]:
```
src = cv.imread(cv.samples.findFile('d.png'))
```

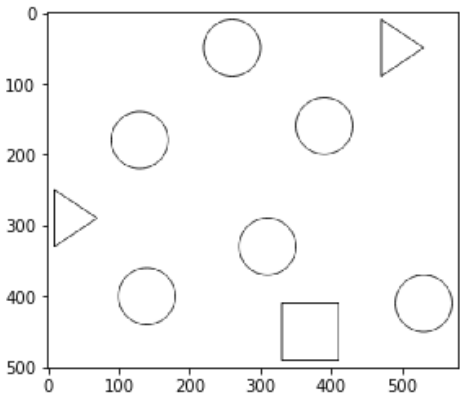
In [45]:
```
gray_image = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
plt.imshow(gray_image,cmap='gray')
gray_image.shape
```

Out[45]:
```
(501, 581)
```

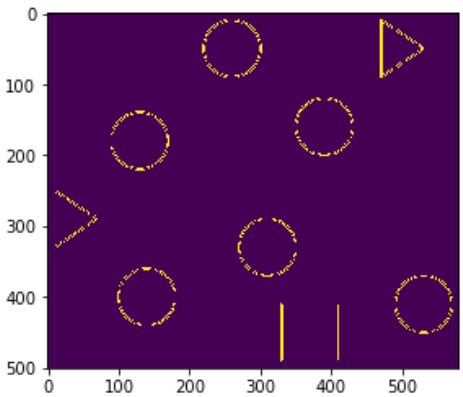



In [46]:
```
output = src.copy()
blur_image = cv.GaussianBlur(gray_image,(gaussian_kernel,gaussian_kernel),0)
plt.imshow(blur_image,cmap='gray')
```

Out[46]:
```
<matplotlib.image.AxesImage at 0x7f56f54c5198>
```




In [47]:
```
edged_image = cv.Canny(blur_image,3,3)
plt.imshow(edged_image)
height,width = edged_image.shape
edges = get_edge_locations(edged_image)
```



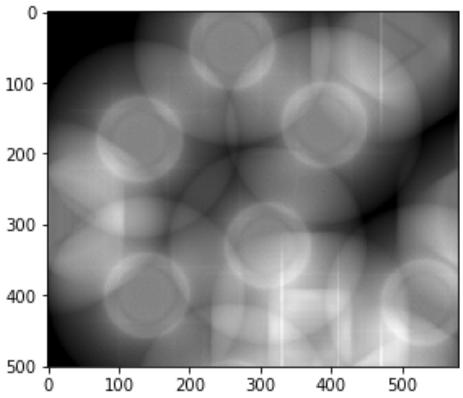
In [48]:
```
max_radius = get_max_possible_radius(edges, max_detection_radius)
max_radius
```

Out[48]: 100

In [49]:
```
acc_array = construct_accumulator_array(edges, width, height, max_radius)
```

In [50]:
```
plt.imshow(np.sum(np.copy(acc_array),axis=2),cmap='gray')
```

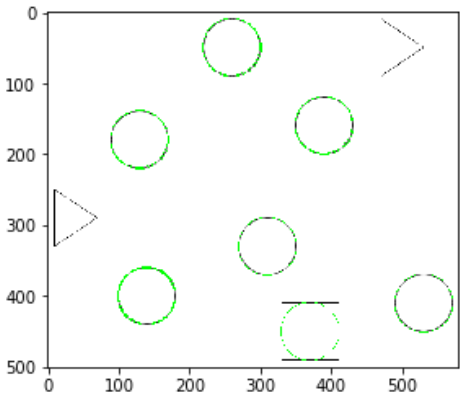
Out[50]: <matplotlib.image.AxesImage at 0x7f56ffcc8c50>




In [51]:
```
output = threshold_accumulator_plot_circles(output, acc_array, max_radius, w
idth, height, intensity_threshold, accumulator_kernel)
plt.imshow(output,cmap='gray')
```

```
index 511 is out of bounds for axis 0 with size 501
index 531 is out of bounds for axis 0 with size 501
index 525 is out of bounds for axis 0 with size 501
index 549 is out of bounds for axis 0 with size 501
index 563 is out of bounds for axis 0 with size 501
```

Out[51]: <matplotlib.image.AxesImage at 0x7f56fff290f0>



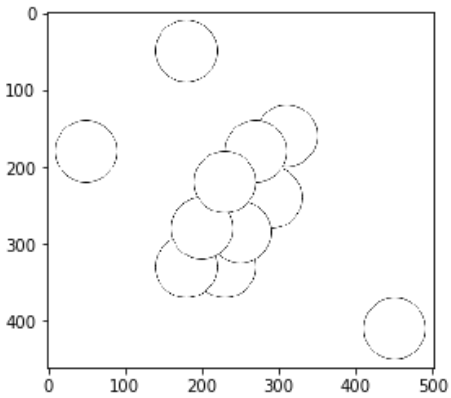

Kształty nie koliste nie spodododowały zakłuceń, wyjątkiem może być kwadrat gdzie transformata próbuje wpisać w niego okrąg, natomiast dało by się go usunąć dobierając lepiej progi, obecne progi są takie same jak w przypadku bazowym, aby móc zaobserować zachodzące zmiany.

# Overlaping circles scenario

In [52]:
```
src = cv.imread(cv.samples.findFile('e.png'))
```

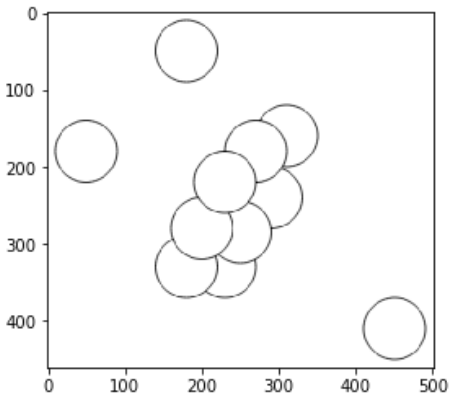
In [53]:
```
gray_image = cv.cvtColor(src, cv.COLOR_BGR2GRAY)
plt.imshow(gray_image,cmap='gray')
gray_image.shape
```

Out[53]:
```
(461, 501)
```

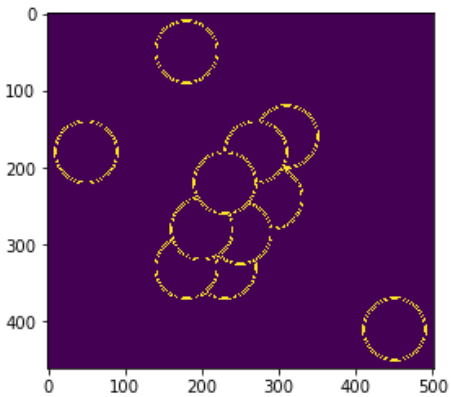



In [54]:
```
output = src.copy()
blur_image = cv.GaussianBlur(gray_image,(gaussian_kernel,gaussian_kernel),0)
plt.imshow(blur_image,cmap='gray')
```

Out[54]:
```
<matplotlib.image.AxesImage at 0x7f56ffe99630>
```




In [55]:
```
edged_image = cv.Canny(blur_image,3,3)
plt.imshow(edged_image)
height,width = edged_image.shape
edges = get_edge_locations(edged_image)
```



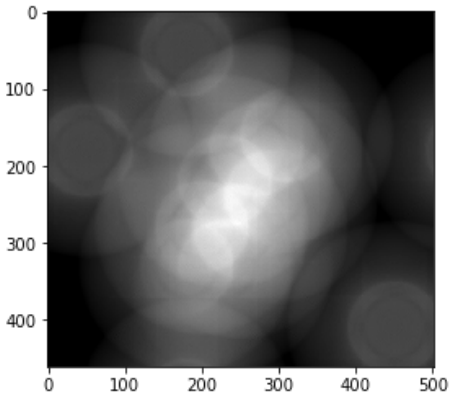
In [56]:
```
max_radius = get_max_possible_radius(edges, max_detection_radius)
max_radius
```

Out[56]: 100

In [57]:
```
acc_array = construct_accumulator_array(edges, width, height, max_radius)
```

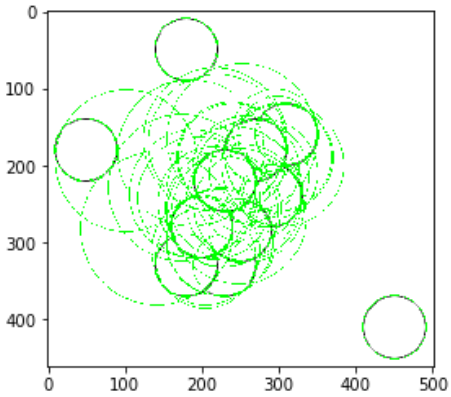
In [58]:
```
plt.imshow(np.sum(np.copy(acc_array),axis=2),cmap='gray')
```

Out[58]: <matplotlib.image.AxesImage at 0x7f570c1f3978>




In [59]:
```
output = threshold_accumulator_plot_circles(output, acc_array, max_radius, w
idth, height, intensity_threshold, accumulator_kernel)
plt.imshow(output,cmap='gray')
```

Out[59]: <matplotlib.image.AxesImage at 0x7f56fff592b0>




Transformata radzi sobie z zachodzącymi na siebie kołami, natomiast takie ich ustawienie powoduje istotny wzrost szumu na akumulatorze, który można odfiltrować za pomocą wyższych progów, Na akumulatorze widać że zbiorowisko okręgów jest wyraźnie jasną palmą.