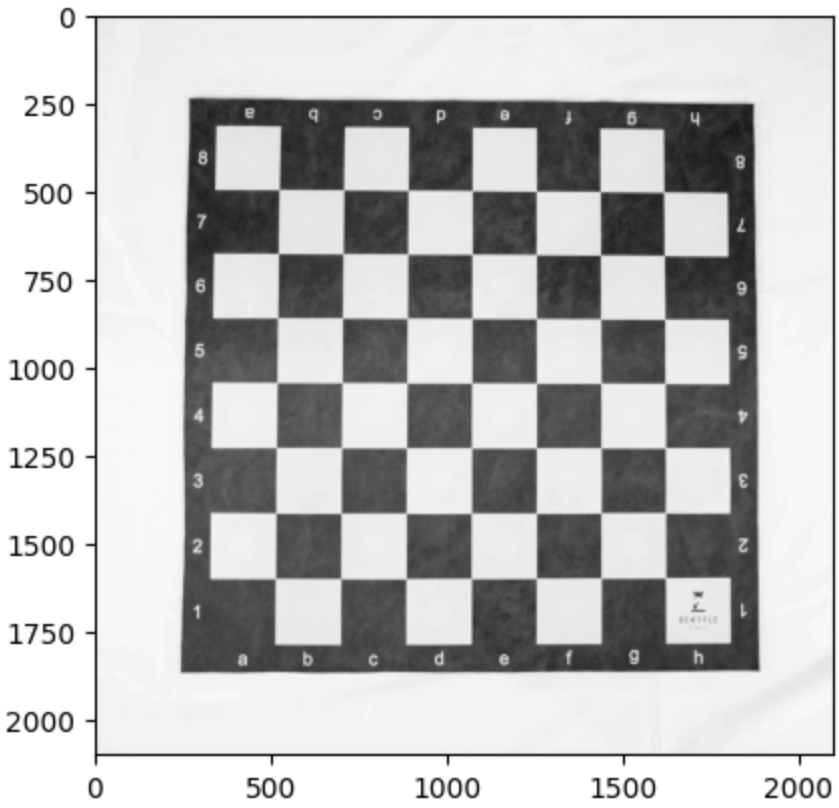
In [37]:
```
import numpy as np
from matplotlib import pyplot as plt
import cv2
```

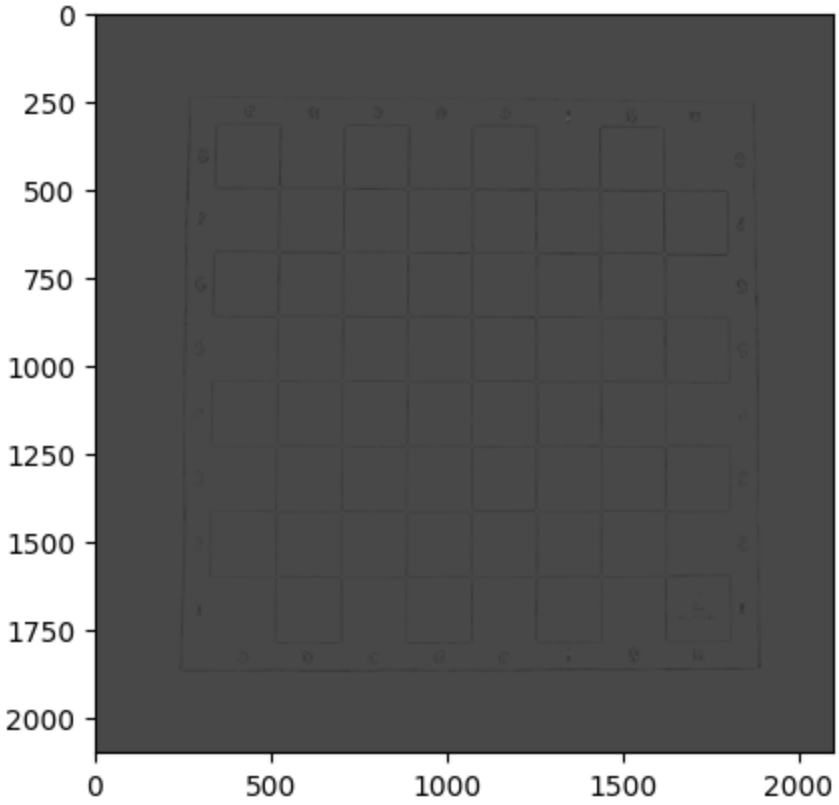
In [38]:
```
img = cv2.imread("bancovua.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(gray, cmap = 'gray')
```

Out[38]:
```
<matplotlib.image.AxesImage at 0x799c6643a320>
```



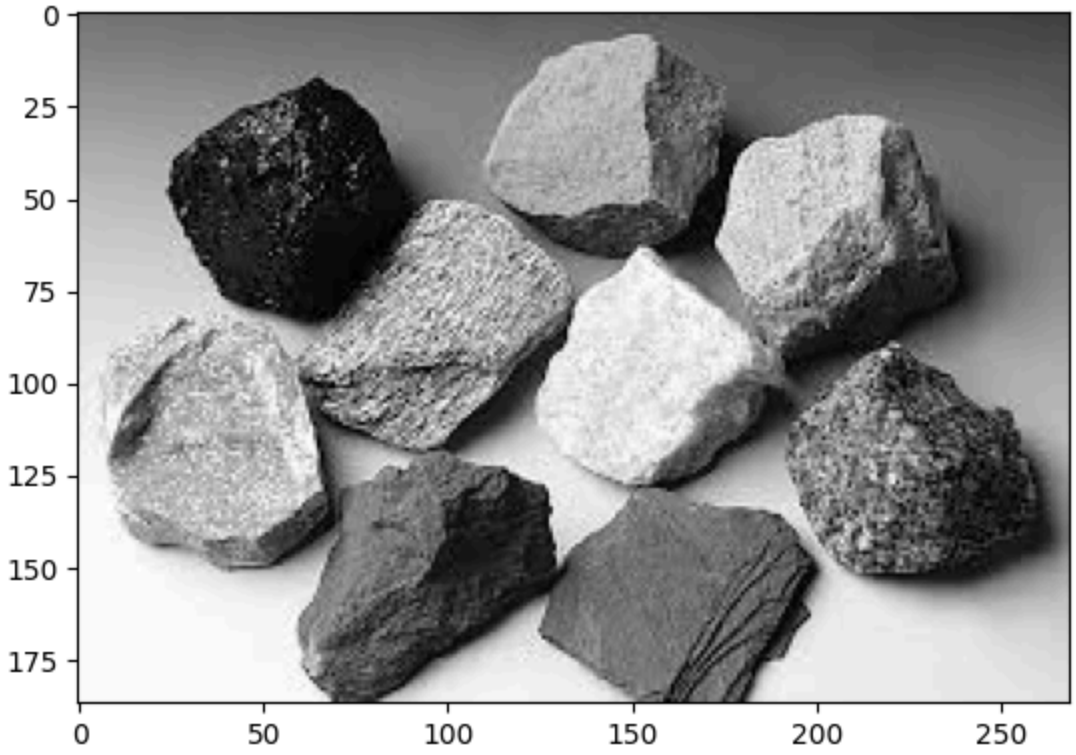

In [39]:
```
corner = cv2.cornerHarris(src = gray, blockSize = 3,
                          ksize = 3, k = 0.06)
plt.imshow(corner, cmap = 'gray')
```

Out[39]:
```
<matplotlib.image.AxesImage at 0x799c6641c820>
```



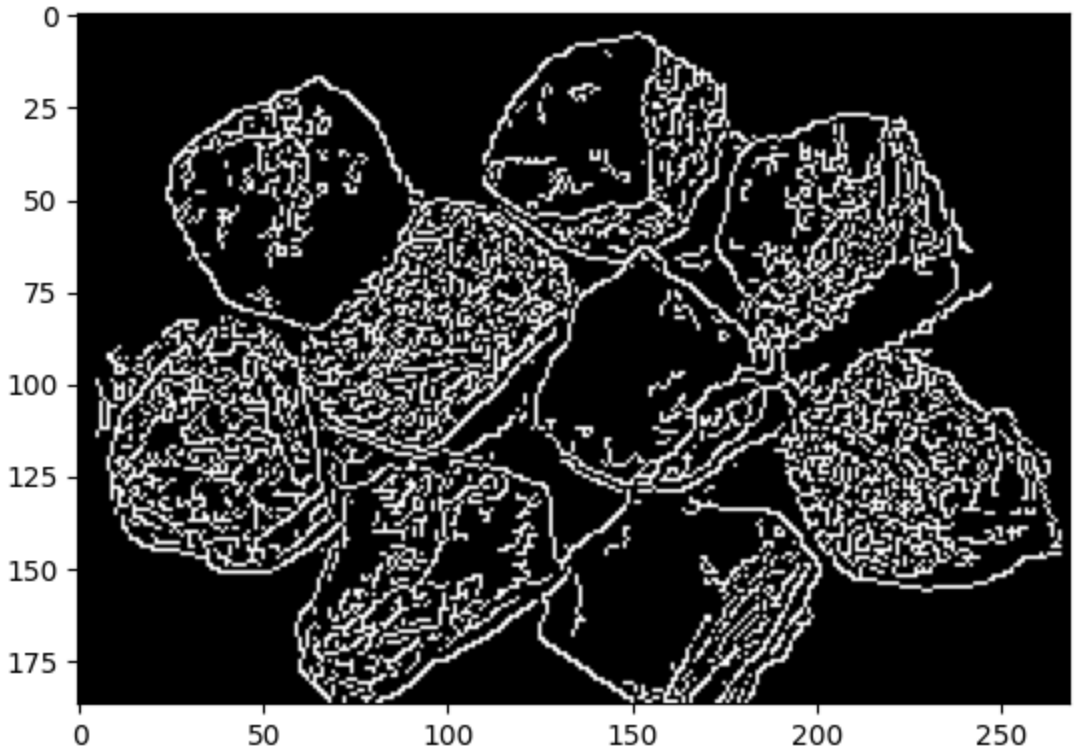

In [40]:
```
img = cv2.imread("da.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(gray, cmap = 'gray')
```

Out[40]:
```
<matplotlib.image.AxesImage at 0x799c6633e590>
```



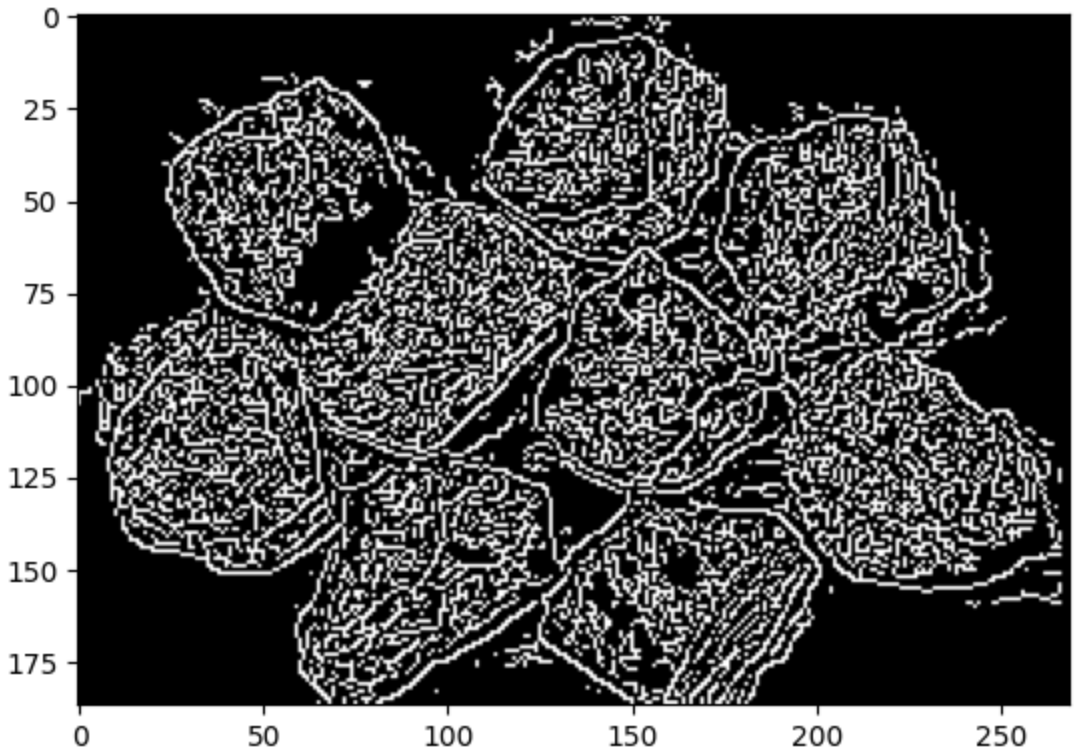

In [41]:
```
edges = cv2.Canny(img, 100,200)
plt.imshow(edges, cmap = 'gray')
```

Out[41]: <matplotlib.image.AxesImage at 0x799c661d8250>



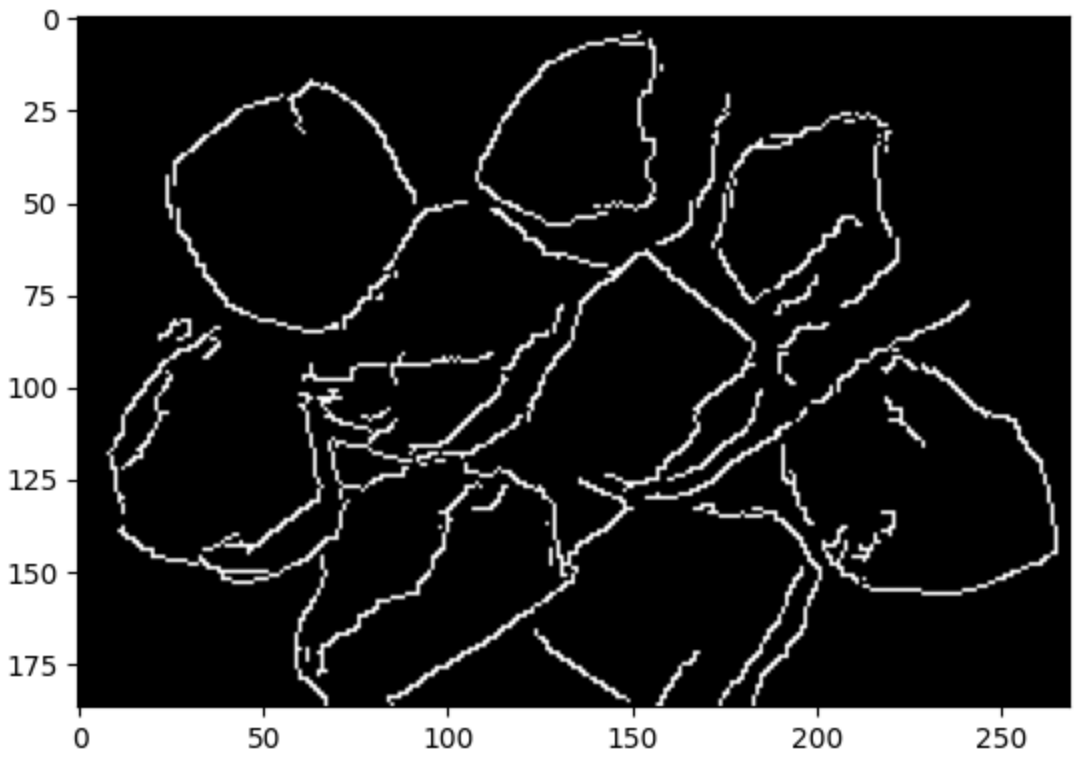

In [42]:
```
m = np.median(img)
lower = 0.7 *m
upper = 0.3 *m
edges = cv2.Canny(img, lower,upper)
plt.imshow(edges, cmap = 'gray')
```

Out[42]: <matplotlib.image.AxesImage at 0x799c6623ea40>



In [43]:
```
blur = cv2.blur(img, (5,5))

edges = cv2.Canny(blur, 100,200)
plt.imshow(edges, cmap = 'gray')
```

Out[43]:
```
<matplotlib.image.AxesImage at 0x799c660d0af0>
```




In [44]:
```
img = cv2.imread("bienxe.webp")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(gray, cmap = 'gray')
```

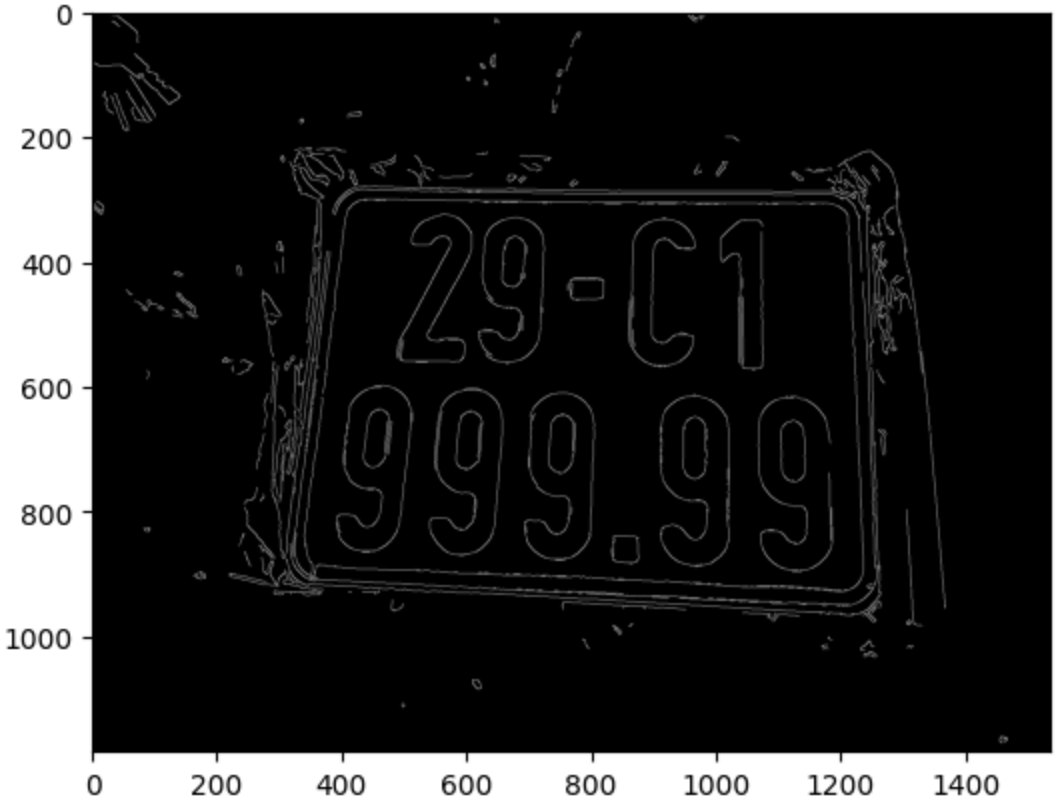
Out[44]:
```
<matplotlib.image.AxesImage at 0x799c66137250>
```


In [45]:
```
ret, thresh = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY)
plt.imshow(thresh, cmap = 'gray')
```

Out[45]:
```
<matplotlib.image.AxesImage at 0x799c65fc90f0>
```

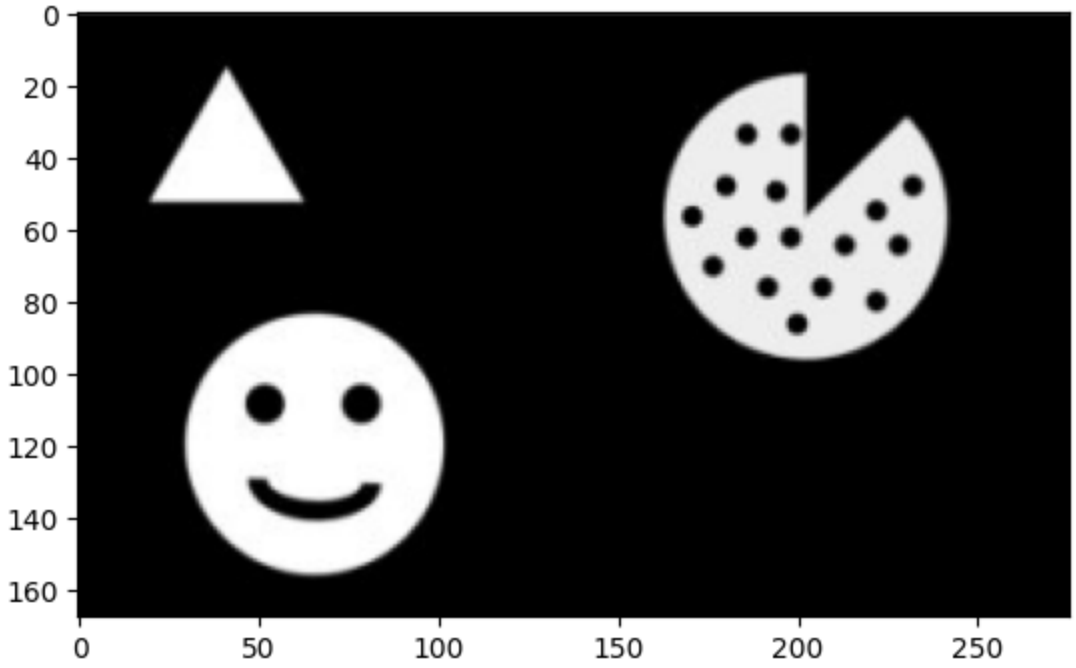
In [46]:
```
blur = cv2.blur(gray, (5,5))

edges = cv2.Canny(blur, 100,200)
plt.imshow(edges, cmap = 'gray')
```

Out[46]:
```
<matplotlib.image.AxesImage at 0x799c6603cfd0>
```



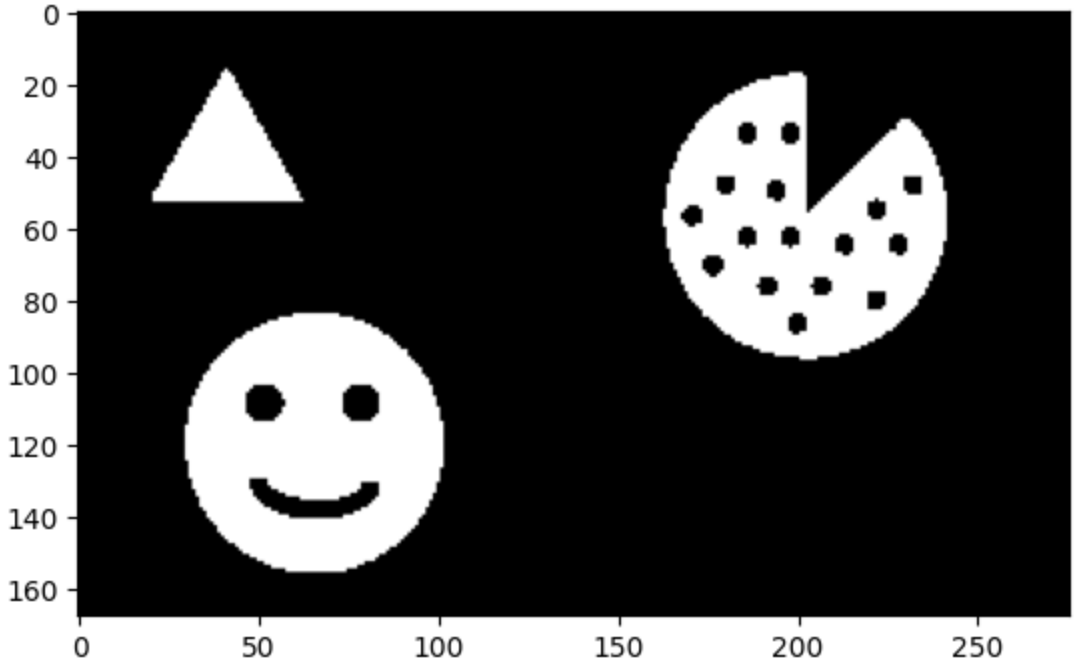

In [47]:
```
img = cv2.imread("Smile.png")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
plt.imshow(gray, cmap = 'gray')
```

Out[47]:
```
<matplotlib.image.AxesImage at 0x799c65eb4910>
```




In [48]:
```
ret, thresh = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY)
plt.imshow(thresh, cmap = 'gray')
```

Out[48]:
```
<matplotlib.image.AxesImage at 0x799c65f36e90>
```

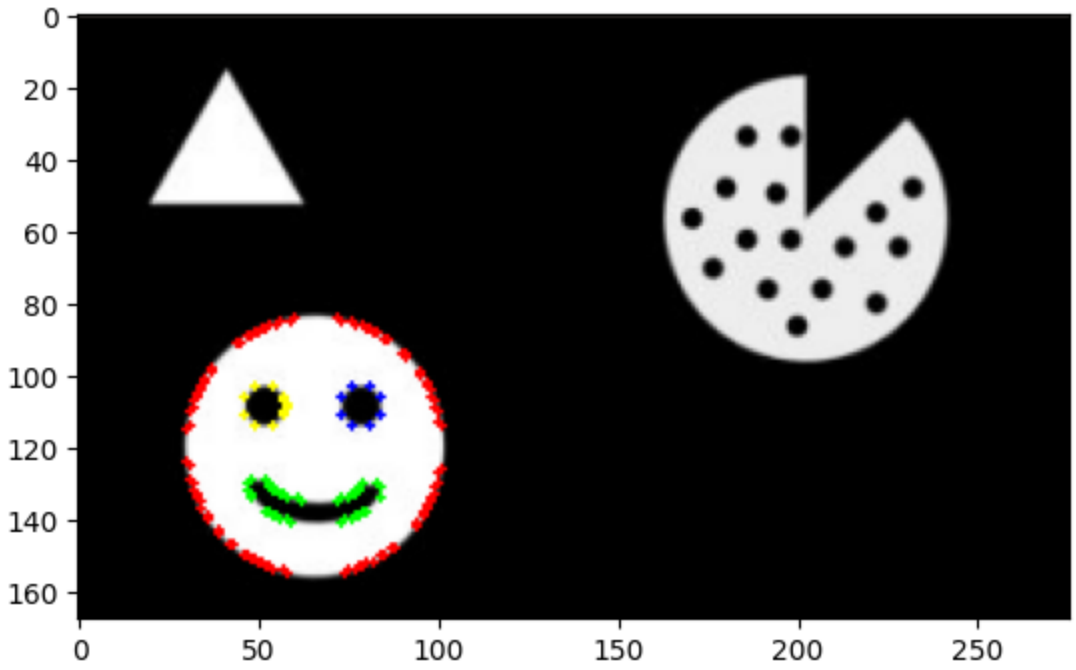



In [49]:
```
contour, hierachy = cv2.findContours(thresh.copy(),
                                     cv2.RETR_CCOMP,
                                     cv2.CHAIN_APPROX_SIMPLE)
```

In [50]:
```
test = img.copy()
cv2.polylines(test, contour[0], isClosed = True, color = [255,0,0], thickness = 2)
cv2.polylines(test, contour[1], isClosed = True, color = [0,255,0], thickness = 2)
cv2.polylines(test, contour[2], isClosed = True, color = [0,0,255], thickness = 2)
```

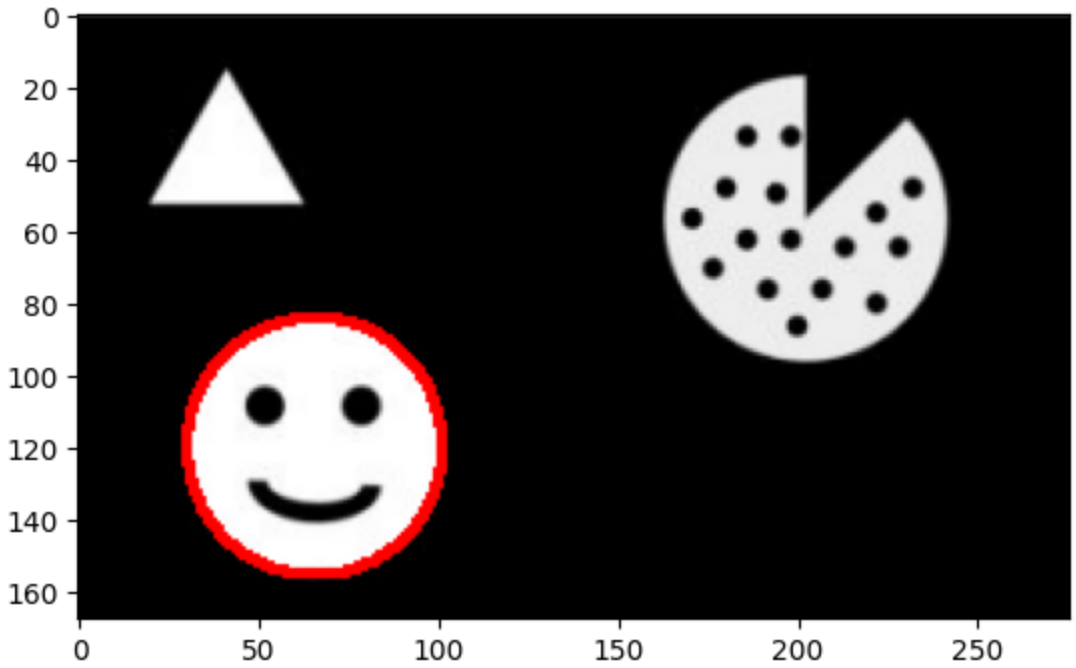
```
cv2.polylines(test, contour[3], isClosed = True, color = [255,255,0], thickness = 2)
plt.imshow(test)
```

Out[50]:
```
<matplotlib.image.AxesImage at 0x799c663febc0>
```




In [51]:
```
test = img.copy()
a = cv2.drawContours(test, contour, 0, color = [255,0,0], thickness = 2)

plt.imshow(a)
```

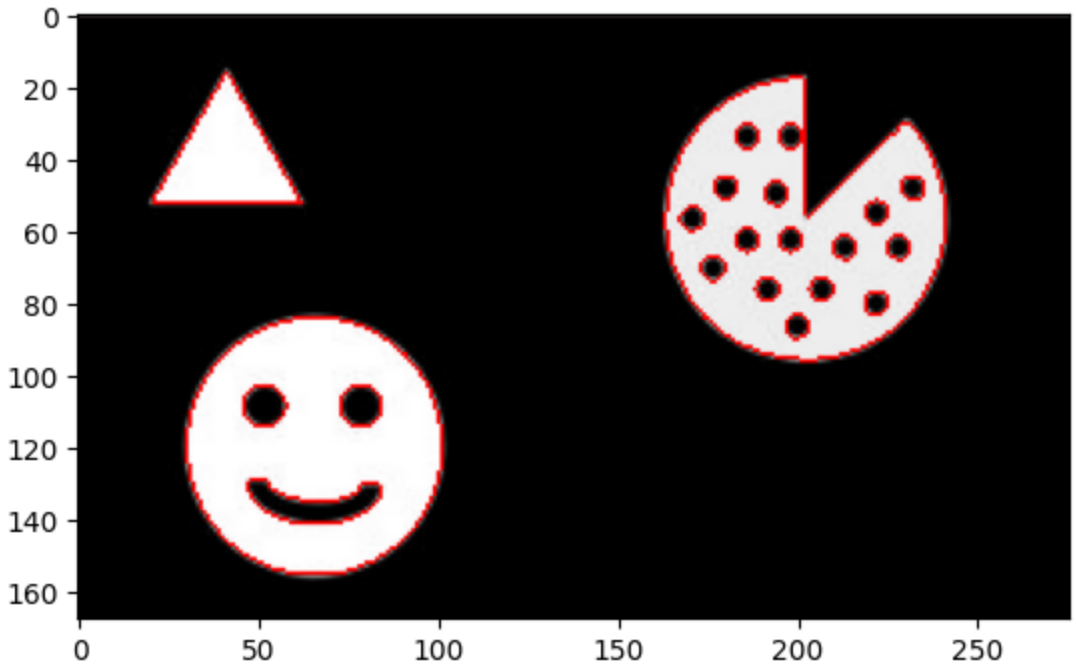
Out[51]:
```
<matplotlib.image.AxesImage at 0x799c66202290>
```




In [52]:
```
test = img.copy()
for i in range(len(contour)):
```

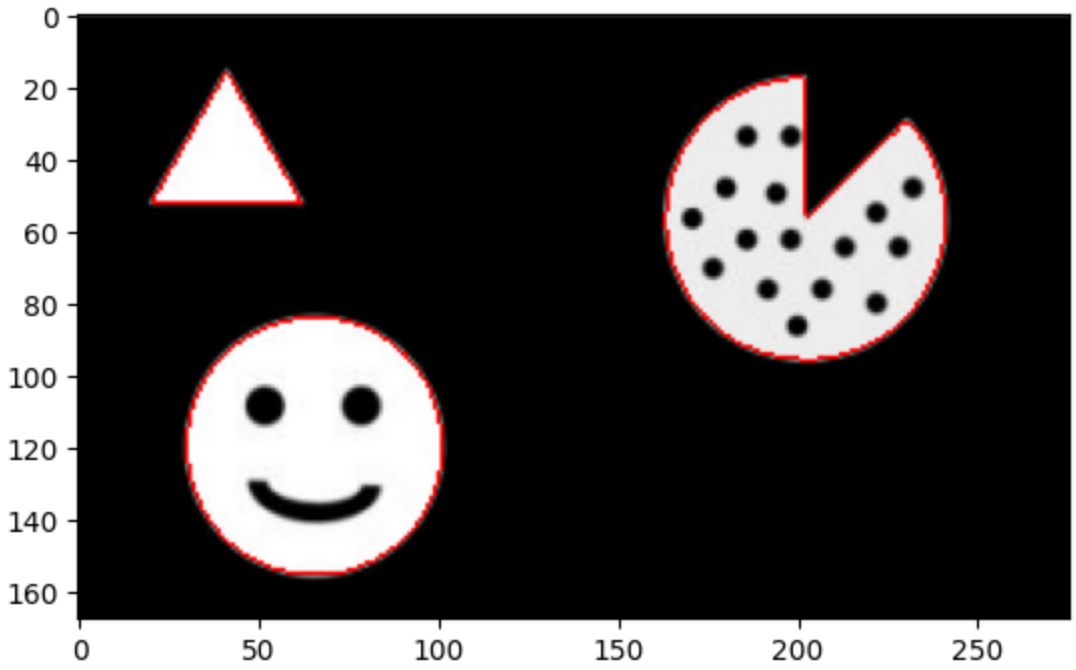
```
  a = cv2.drawContours(test, contour, i, color = [255,0,0], thickness = 1)
plt.imshow(a)
```

Out[52]:
```
<matplotlib.image.AxesImage at 0x799c65e05ea0>
```




In [53]:
```
test = img.copy()
for i in range(len(contour)):
  if hierachy[0][i][3] == -1:
    a = cv2.drawContours(test, contour, i, color = [255,0,0], thickness = 1)
plt.imshow(a)
```

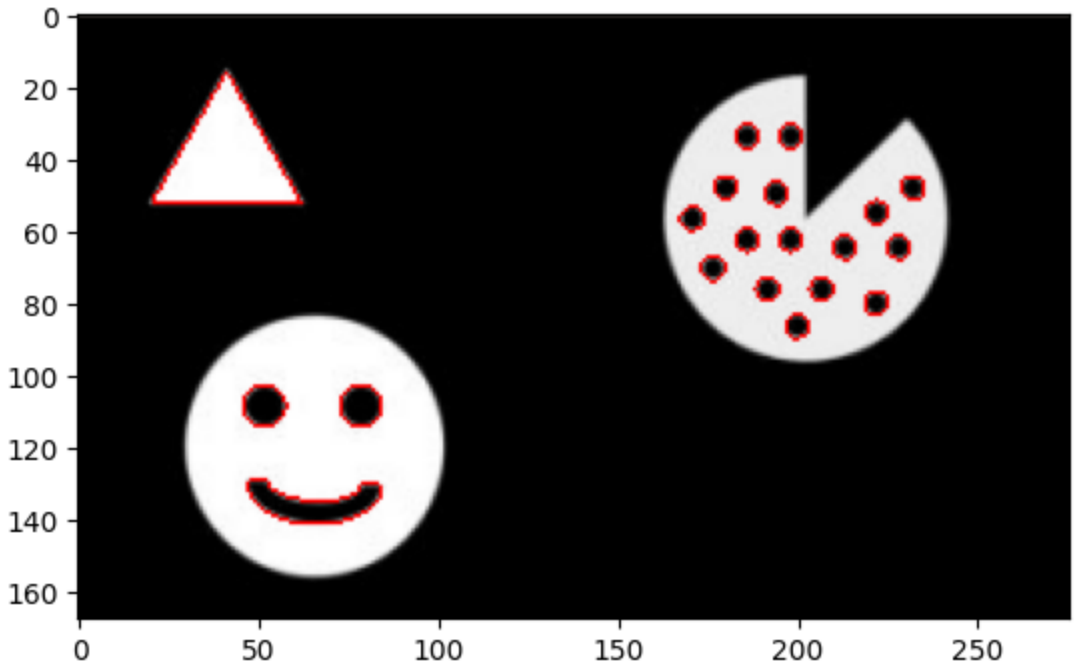
Out[53]:
```
<matplotlib.image.AxesImage at 0x799c65d44c40>
```




In [54]:
```
test = img.copy()
for i in range(len(contour)):
```

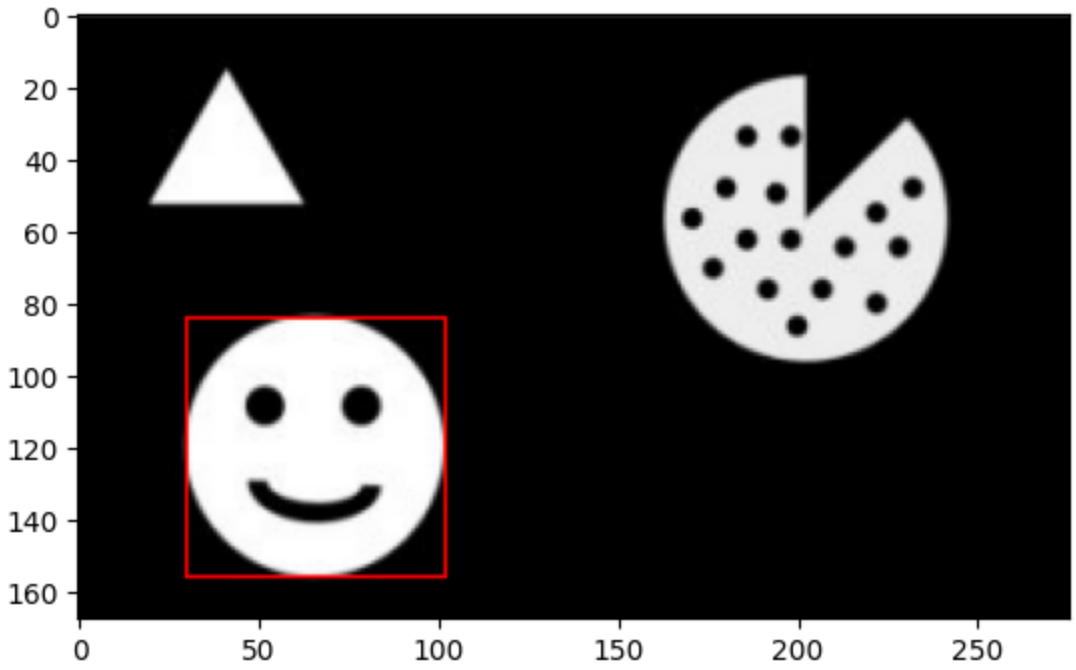
```
  if hierachy[0][i][2] == -1:
    a = cv2.drawContours(test, contour, i, color = [255,0,0], thickness = 1)
plt.imshow(a)
```

Out[54]: <matplotlib.image.AxesImage at 0x799c65d9ff70>




In [55]:
```
test = img.copy()
x1,y1,w,h = cv2.boundingRect(contour[0])
a = cv2.rectangle(test, (x1,y1), (x1+w, y1+h), color = [255,0,0], thickness = 1)
plt.imshow(a)
```

Out[55]: <matplotlib.image.AxesImage at 0x799c65c3a050>



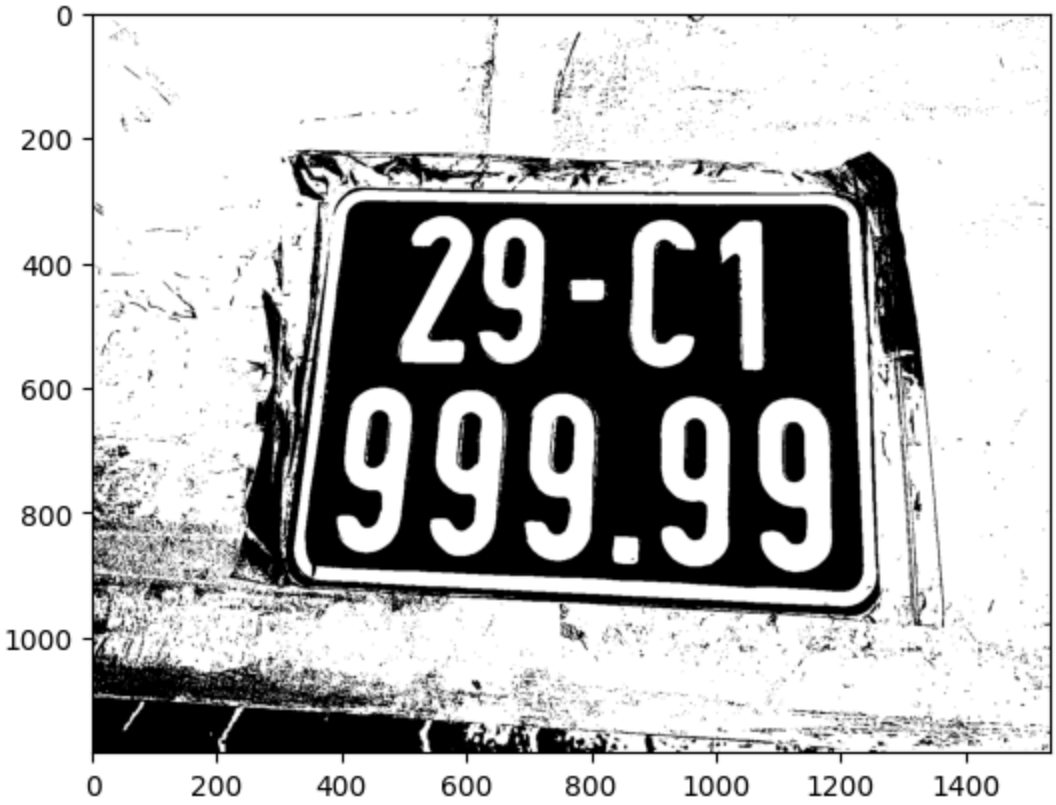

In [56]:
```
img_bien = cv2.imread("bienxe.webp")
gray = cv2.cvtColor(img_bien, cv2.COLOR_BGR2GRAY)
```

```
plt.imshow(gray, cmap = 'gray')
```

Out[56]:
```
<matplotlib.image.AxesImage at 0x799c65ac5a80>
```

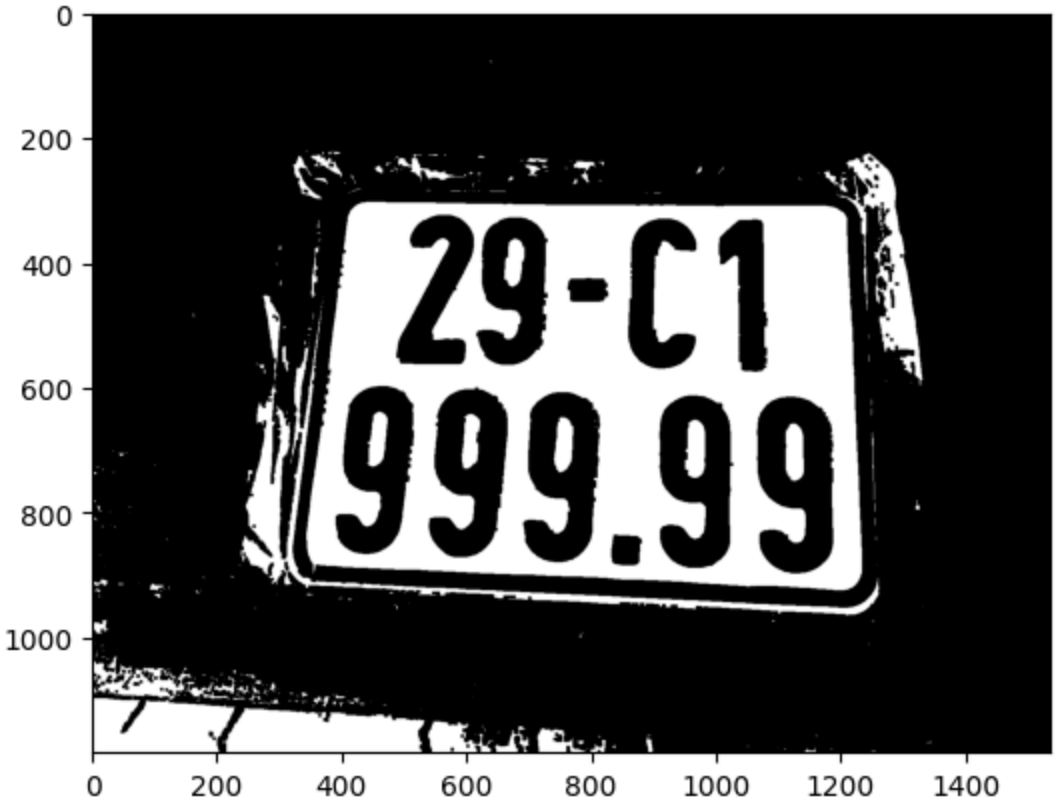

In [57]:
```
ret, binr = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY_INV)
plt.imshow(binr, cmap = 'gray')
```

Out[57]:
```
<matplotlib.image.AxesImage at 0x799c65b38af0>
```




In [58]:
```
kernel = np.ones((5, 5), np.uint8)
invert = cv2.bitwise_not(binr)
erosion = cv2.erode(invert, kernel,
                    iterations=1)
plt.imshow(erosion, cmap = 'gray')
```

Out[58]:
```
<matplotlib.image.AxesImage at 0x799c659c2110>
```




In [61]:
```
ret, thresh = cv2.threshold(gray, 120, 255, cv2.THRESH_BINARY)
plt.imshow(thresh, cmap = 'gray')
```

Out[61]:
```
<matplotlib.image.AxesImage at 0x799c65a37220>
```

In [62]:
```
contour, hierachy = cv2.findContours(thresh.copy(),
                                     cv2.RETR_CCOMP,
                                     cv2.CHAIN_APPROX_SIMPLE)
```

In [72]:
```
MAX_w = 900
MIN_w = 200
bien = img_bien.copy()
if len(contour) > 0:
  for c in contour:
    x,y,w,h = cv2.boundingRect(c)
    ar = w/h

    if (max(w,h) < MAX_w) and (min(w,h) > MIN_w) and (np.abs(1.0 - ar) < 0.5):
      cv2.rectangle(bien, (x,y), (x+w, y+h), (255,0,0),5)
      break
plt.imshow(bien)
```

Out[72]: <matplotlib.image.AxesImage at 0x799c65d0a260>


In [ ]: