程式用Python編寫，於Colab上執行。

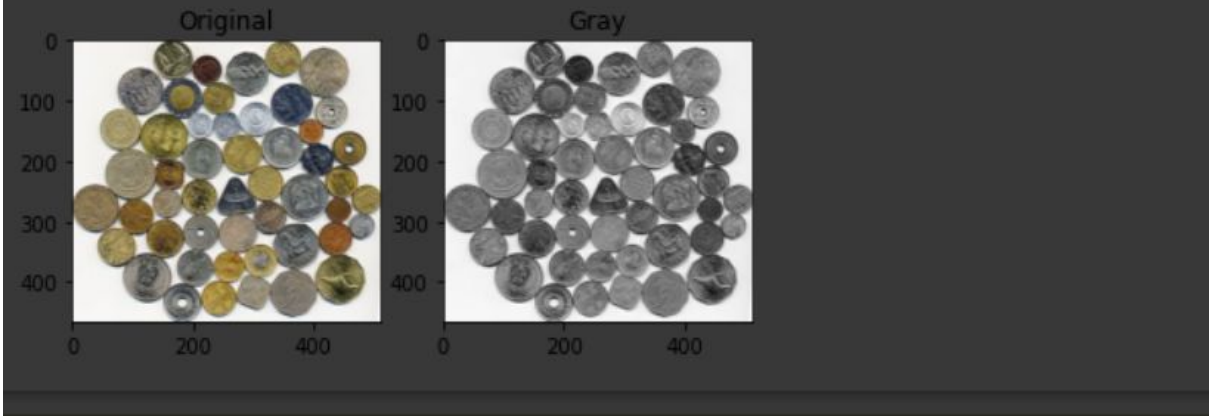
網址：
https://colab.research.google.com/drive/1iFGHfYn9o2h_hV5C40Q8cL9_fo3xeWma?usp=sharing

```
[ ]  1  import numpy as np
     2  import cv2 as cv
     3  from matplotlib import pyplot as plt
```

以上是這次作業用到的函式庫。

**1. 彩色轉灰階結果**

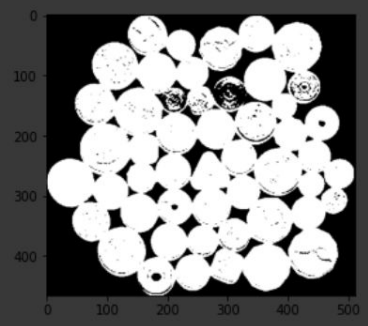
```
1  #  1.彩色轉灰階結果
2  img = cv.imread('hw3_coins2.jpg', cv.COLOR_BGR2RGB)
3  gray = cv.cvtColor(img,cv.COLOR_RGB2GRAY)
4  plt.subplot(1, 2, 1)
5  plt.title("Original")
6  plt.imshow(img[:, :, ::-1])
7  plt.subplot(1, 2, 2)
8  plt.title("Gray")
9  plt.imshow(gray, cmap = 'gray')
10 plt.show()
```




我在做這次作業時，搞錯colormap（使用的是matplotlib的Greys）了，一開始出來的圖片完全不是我要的，換成現在gray才看起來正常。然後opencv讀取彩圖是BGR，所以我先改成RGB（這樣才好顯示出來），然後才換成灰階。

## 2. 二值化(Otsu)結果

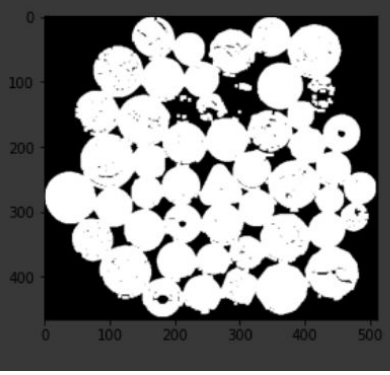
```
# 2.二值化(Otsu)結果
ret, thresh = cv.threshold(gray, 0, 255, cv.THRESH_BINARY_INV + cv.THRESH_OTSU)
plt.imshow(thresh, cmap = 'gray')
plt.show()
```




按照網頁上的作法做出Otsu的結果並顯示出來。

## 3. opening去雜訊結果

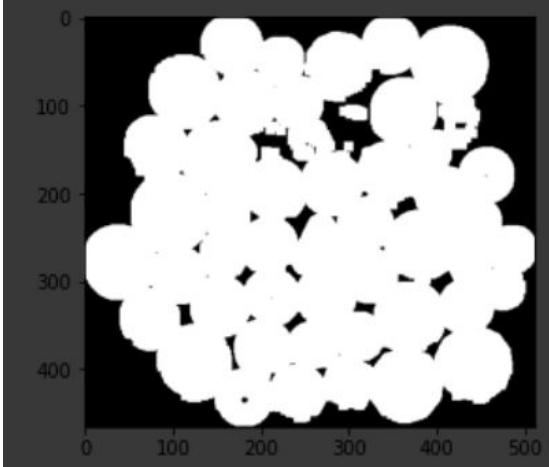
```
# 3.opening去雜訊結果
kernel = np.ones((3, 3), np.uint8)
opening = cv.morphologyEx(thresh, cv.MORPH_OPEN, kernel, iterations = 2)
plt.imshow(opening, cmap = 'gray')
plt.show()
```




也是按照網頁的作法將Otsu的圖片去雜訊，iteration為2，這是我測試出來能得到的最佳參數，過大或過小都會導致最後的結果很差。

**4. sure background**

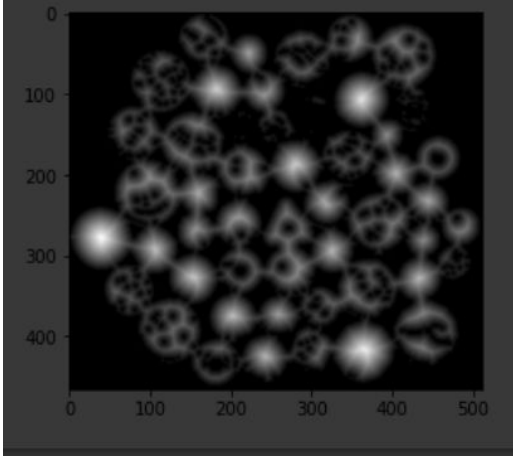
```
# 4.sure background
sure_bg = cv.dilate(opening, kernel, iterations = 3)
plt.imshow(sure_bg, cmap = 'gray')
plt.show()
```




也是按照網頁做出的sure background，iterations也是測試出來最好的參數。

**5. distance transform結果**

```
# 5.distance transform結果
dist_transform = cv.distanceTransform(opening ,cv.DIST_L2, 5)
plt.imshow(dist_transform, cmap = 'gray')
plt.show()
```




也是按照網頁做出的distance transform結果，以便下一步做出sure foreground area。

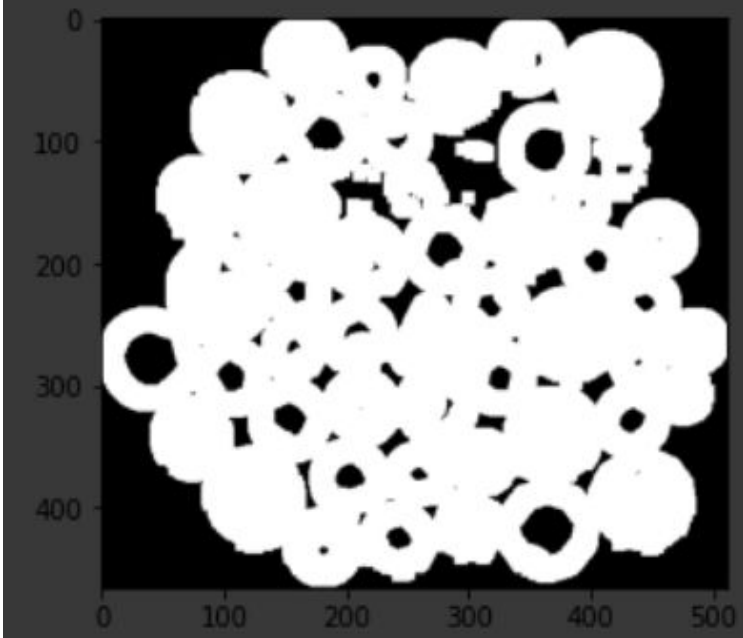
### 6. sure foreground

```
# 6.sure foreground
ret, sure_fg = cv.threshold(dist_transform, 0.5 * dist_transform.max(), 255,0)
sure_fg = np.uint8(sure_fg)
plt.imshow(sure_fg, cmap = 'gray')
plt.show()
```


這裡也是按照網頁來做出sure foreground，只是有調整一些參數值來讓結果更好。

**7. unknown regions**

```
# 7. unknown regions
unknown = cv.subtract(sure_bg, sure_fg)
plt.imshow(unknown, cmap = 'gray')
plt.show()
```




這裡也是按照網頁來做出unknown region，沒有調整參數。

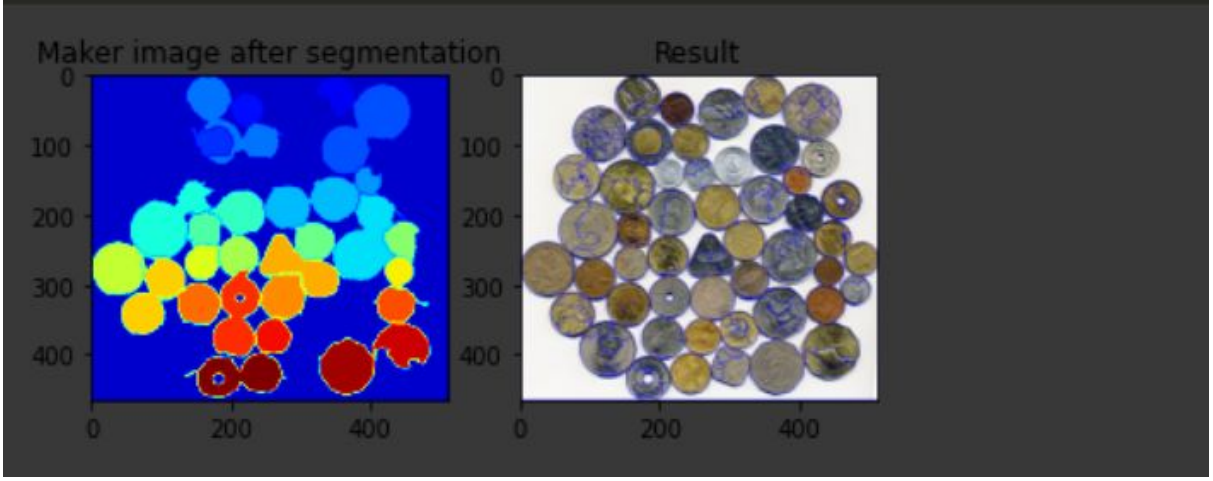
**8. markers**

```
# 8.markers
ret, markers = cv.connectedComponents(sure_fg)
# Add one to all labels so that sure background is not 0, but 1
markers = markers + 1
# Now, mark the region of unknown with zero
markers[unknown == 255] = 0
plt.imshow(markers, cmap = 'jet')
plt.show()
```


這裡也是按照網頁來做，顏色就淺的藍色部份是sure background，深藍的則是unknown region，而sure coins則是用各種不同顏色來表示。

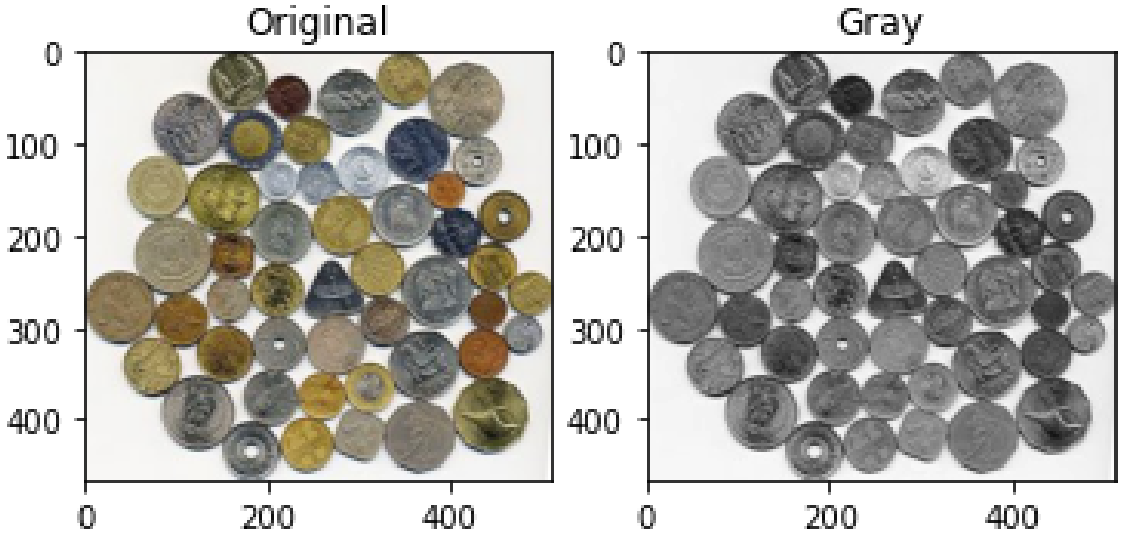
**9. watershed結果**

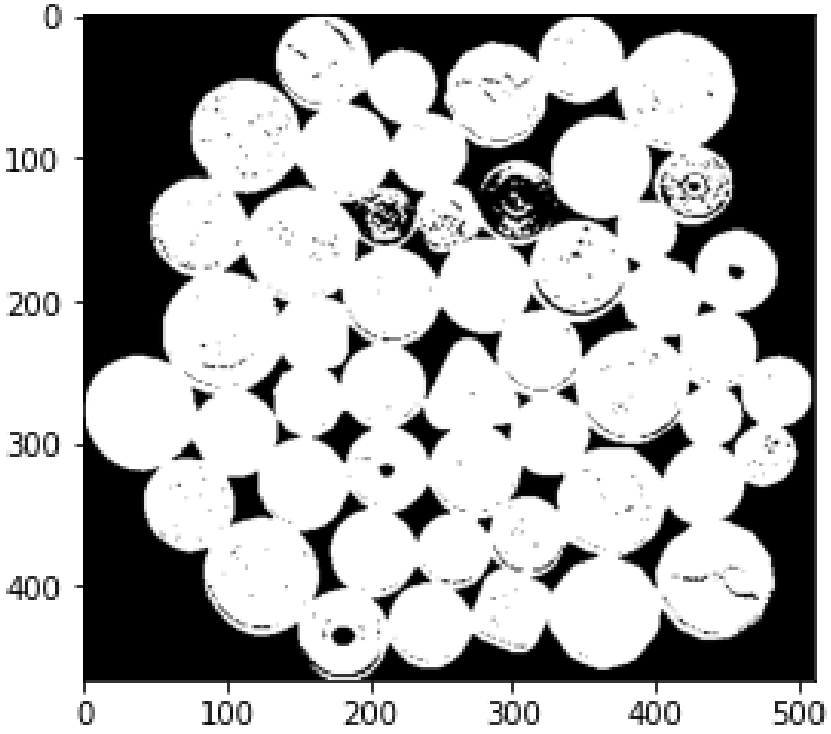
```
# 9.watershed結果
markers = cv.watershed(img,markers)
img[markers == -1] = [255,0,0]
plt.subplot(1, 2, 1)
plt.title('Maker image after segmentation')
plt.imshow(markers, cmap = 'jet')
plt.subplot(1, 2, 2)
plt.title('Result')
plt.imshow(img[:, :, ::-1])
plt.show()
```

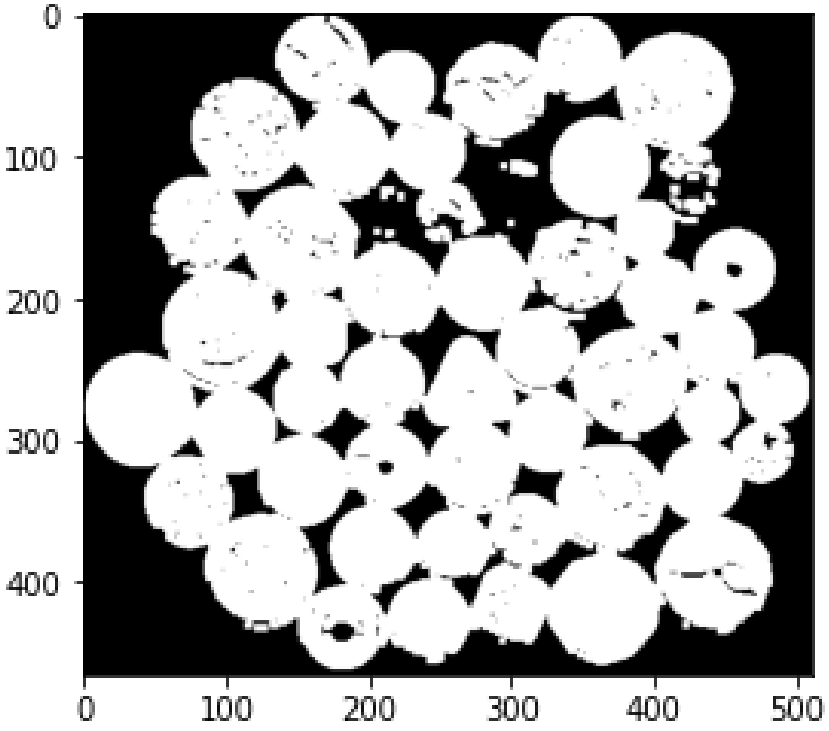


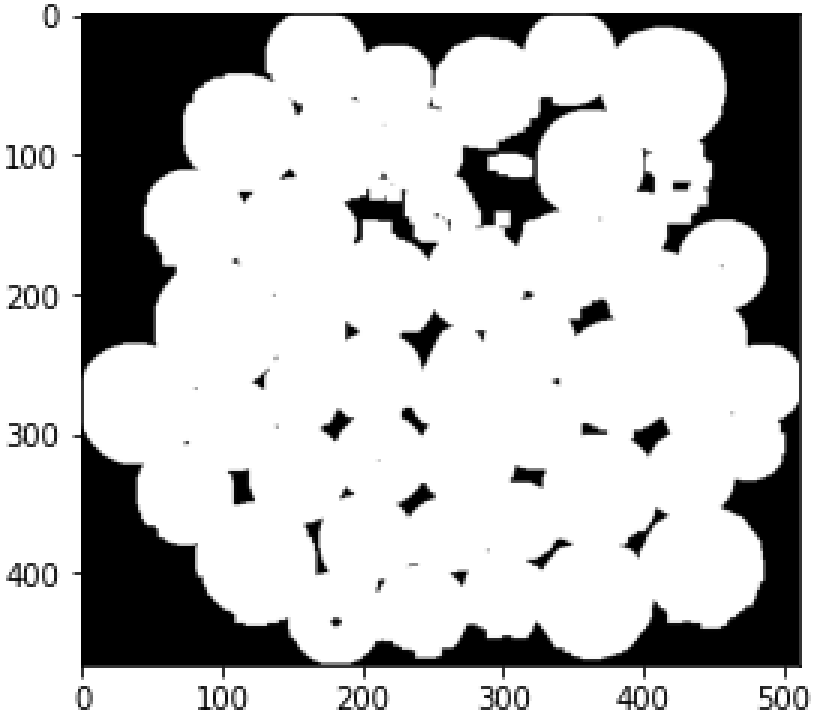

最後將原圖和上面的markers做watershed，依照得出的結果對原圖做segmentation，用深藍色將硬幣框起來，就是Result。

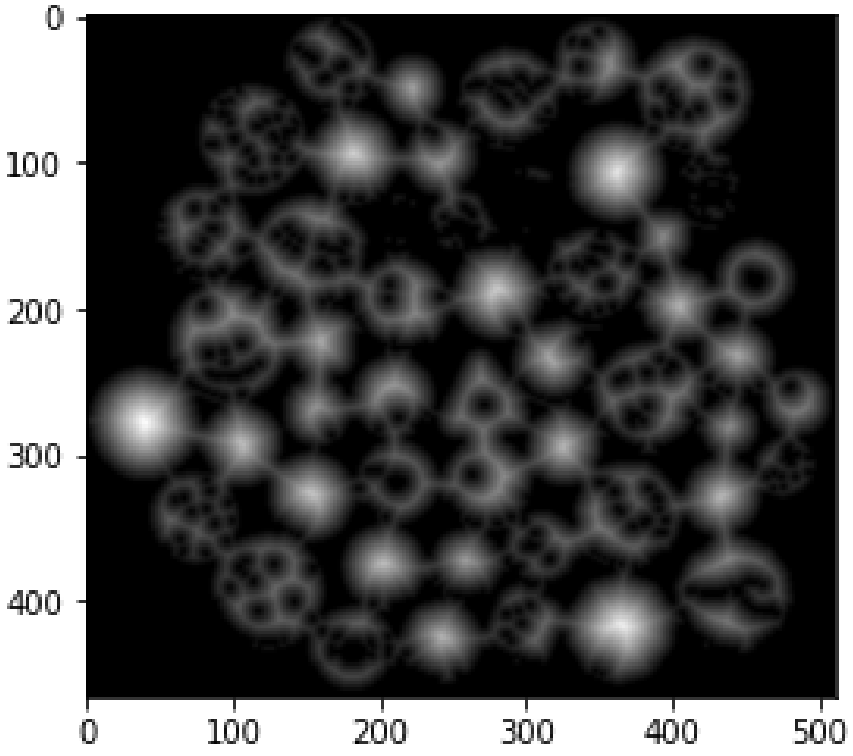
**以下是這次做好的所有圖**

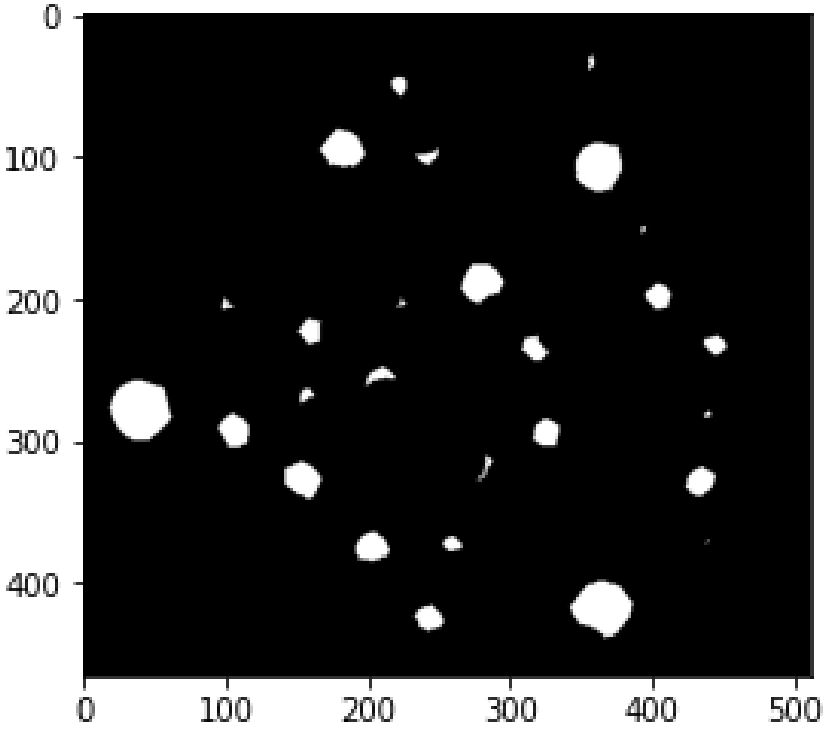

Original
Gray
0
100
200
300
400
0
200
400

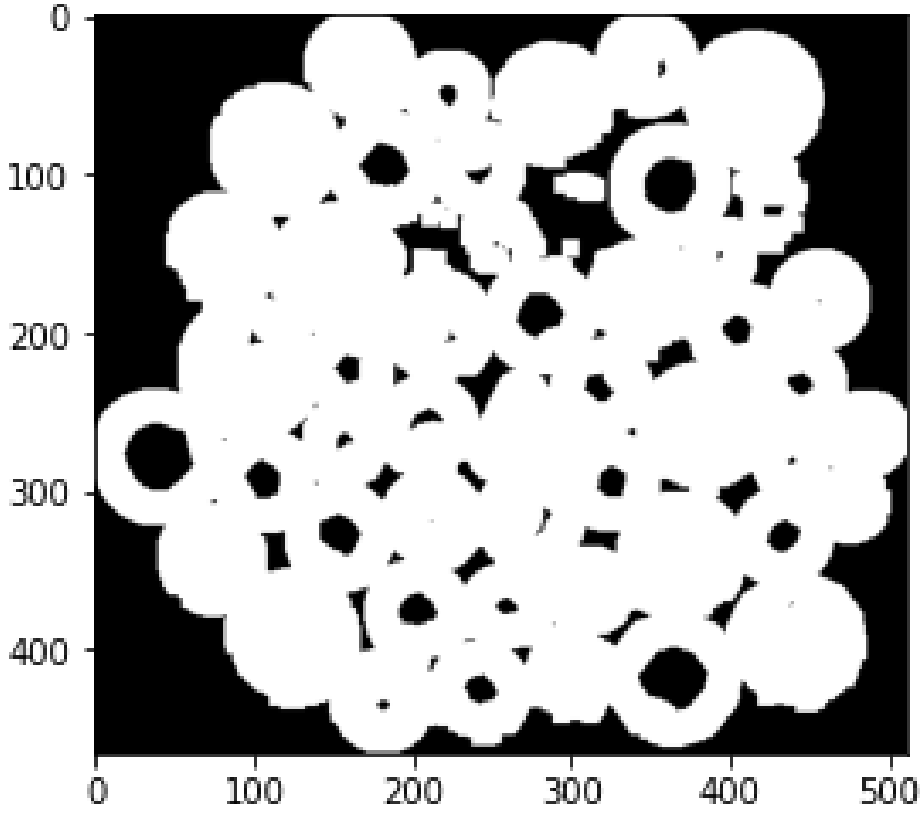

0
100
200
300
400
0
100
200
300
400
500

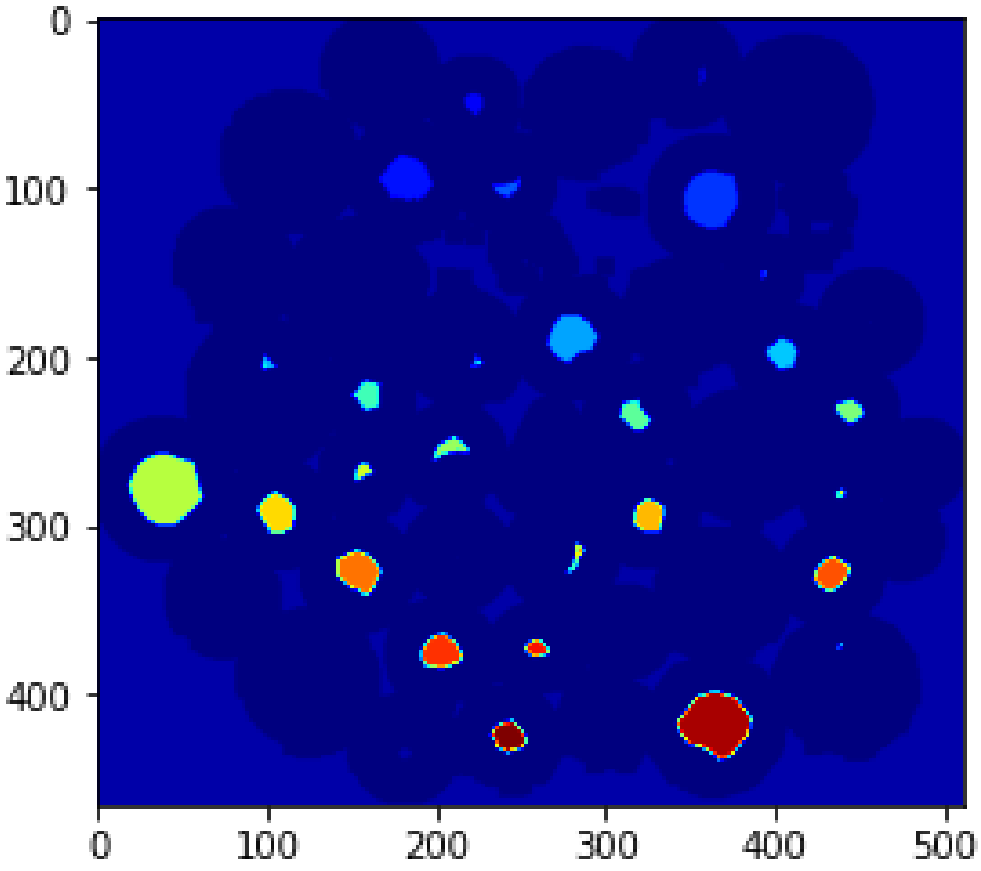

0
100
200
300
400
0
100
200
300
400
500

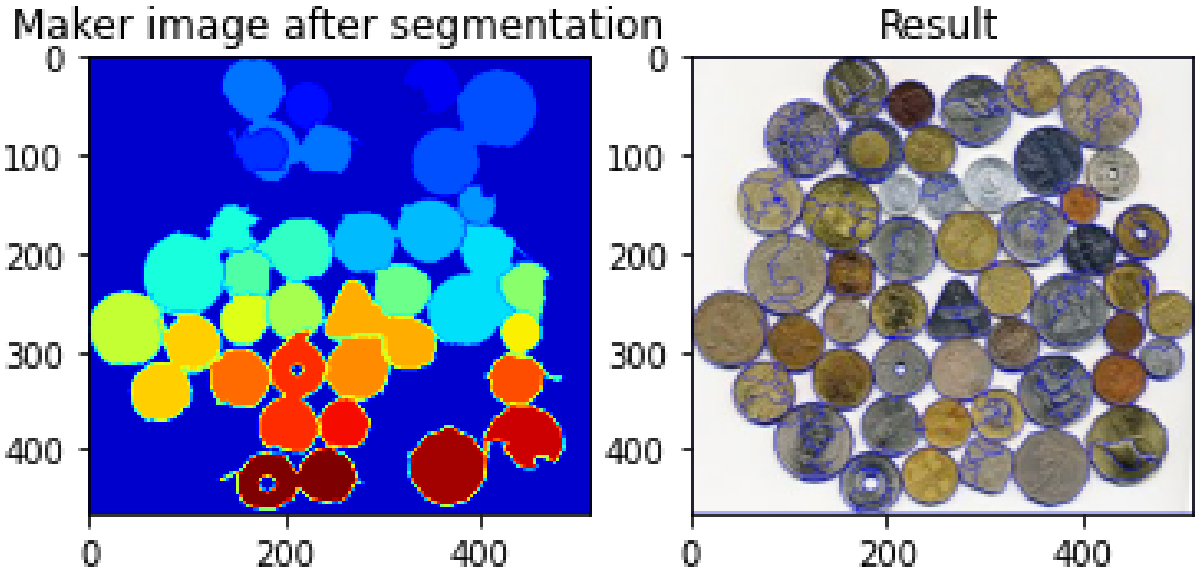

0
100
200
300
400
0
100
200
300
400
500


0
100
200
300
400
0
100
200
300
400
500


0
100
200
300
400
0
100
200
300
400
500


0
100
200
300
400
0
100
200
300
400
500


0
100
200
300
400
0
100
200
300
400
500


Maker image after segmentation
Result
0
100
200
300
400
0
200
400
0
100
200
300
400
0
200
400