计算图片中的硬币数量

1. 图像分析

给定的图片为一张散布着硬币的照片，硬币之间无重叠，若二值化以后硬币内部无孔洞就可以直接使用形态学的方法计算图片内部的连通域个数。若二值化以后图片仍然存在细碎的敌方 2 则要使用形态学操作里的腐蚀操作将小区域去除，使得新图像中的每一个连通域都对应原图中的一枚硬币。然后再使用计算连通域的方法进行计数。

2. 实验过程

主要使用 python 以及 opencv 进行图像的处理，代码如下所示：

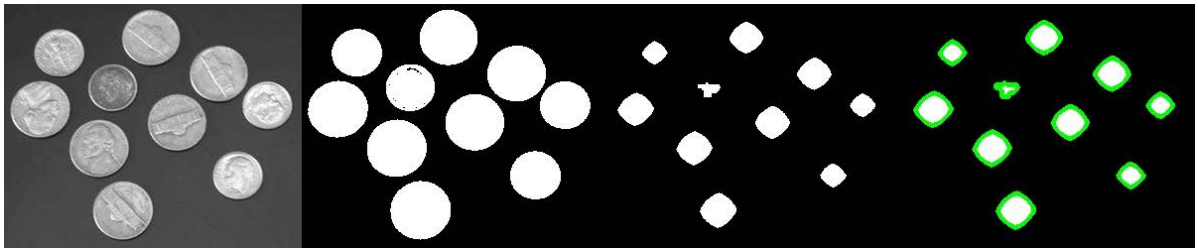
```
import cv2
import numpy as np


coins = cv2.imread("coins.jpg")
coinsGray = cv2.cvtColor(coins, cv2.COLOR_BGR2GRAY)
ret, coinsBin = cv2.threshold(coins, 95, 255, cv2.THRESH_BINARY)

kern = np.ones((5, 5), np.uint8)
coinsEroded = cv2.erode(coinsBin, kernel=kern, iterations=5)
coinsEroded = cv2.cvtColor(coinsEroded, cv2.COLOR_BGR2GRAY)
contours, hierarchy = cv2.findContours(coinsEroded, cv2.RETR_TREE, cv
2.CHAIN_APPROX_SIMPLE)

coinsEroded = cv2.cvtColor(coinsEroded, cv2.COLOR_GRAY2BGR)
coinsDrawn = coinsEroded.copy()
coinsDrawn = cv2.drawContours(coinsDrawn, contours, -1, (0,255,0), 3)

res = np.hstack((coins,  coinsEroded, coinsDrawn))
cv2.imshow('1', res)
cv2.waitKey(0)
cv2.imwrite('res.jpg', res)
print("The image contains %d coins." % (len(contours)))
```

3. 结果分析

程序运行结果如下图所示，可以看到程序正确数出了图中的 10 个硬币。

```
The image contains 10 coins.

Process finished with exit code 0
```

程序对图片处理的过程如下图所示：



从左到右分别是原图、二值化处理、腐蚀操作、寻找轮廓以后得到的图。对 contours 这个 list 求长度就可以得到硬币的个数。

寻找图片中线圈的内外轮廓

1. 图像分析

待处理的图片是一个线圈，外部轮廓较为清晰，但内部轮廓有很多杂乱的线，不易识别，可以考虑先对其进行二值化然后进行腐蚀操作去除掉内圈中较细的线，然后再进行膨胀操作以保持原图中线圈的大小。得到较为清楚的图像后使用 opencv 的寻找最小外接圆确定线圈的外轮廓，由于线圈是同心圆结构，在外轮廓确定后内轮廓的圆心也就能确定了，此时再对内轮廓的半径进行拟合便可得到内轮廓。

2. 实验过程

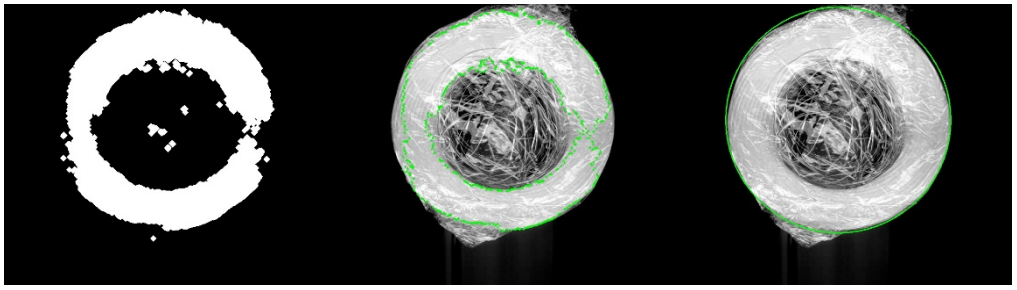
主要使用 python 以及 opencv 进行图像的处理，代码如下所示：

```python
import cv2
import numpy as np
import math


def distance(x, y): # return squared euclid distance
    return (x[0]-y[0])**2 + (x[1]-y[1])**2


tape = cv2.imread('tape.jpeg')
```

```
tapeGray = cv2.cvtColor(tape, cv2.COLOR_BGR2GRAY)
ret, tapeBin = cv2.threshold(tapeGray, 180, 255, cv2.THRESH_BINARY)

kern = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(5,5))
kern2 = cv2.getStructuringElement(cv2.MORPH_ELLIPSE,(3,3))

tapeEroded = cv2.erode(tapeBin, kernel=kern, iterations=2)
tapeDilated = cv2.dilate(tapeEroded, kernel=kern2, iterations=7)
cv2.imwrite('tapeDilated.jpg', tapeDilated)

contours, hierarchy = cv2.findContours(tapeDilated, cv2.RETR_TREE, cv
2.CHAIN_APPROX_SIMPLE)

longestContour = contours[0]
for contour in contours:
    if len(longestContour) < len(contour):
        longestContour = contour

tapeDrawn = tape.copy()

(x, y), radius = cv2.minEnclosingCircle(longestContour)
center = (int(x), int(y))
radius = int(radius)
cv2.circle(tapeDrawn, center, radius, (0,255,0), 2)
longestContour = np.squeeze(longestContour)
longestContour = longestContour.tolist()

innerCircle = []
for x in longestContour:
    if distance(x, center) < (radius-50)**2:
        innerCircle.append(x)

distanceSum = 0
for x in innerCircle:
    distanceSum += distance(x, center)

estimatedR = math.sqrt(distanceSum/len(innerCircle))
estimatedR = int(estimatedR)
cv2.circle(tapeDrawn, center, estimatedR, (0,255,0), 2)
cv2.imwrite('tapeDrawn.jpg', tapeDrawn)
```

3. 结果分析

在二值化后进行腐蚀、膨胀操作后得到的图以及找到的轮廓如下图所示:

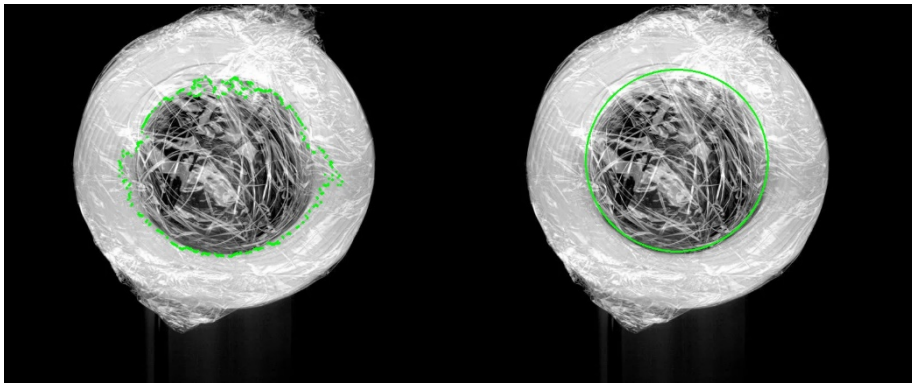


由于处理后的图片仍然存在很多杂点，所以寻找轮廓会返回很多结果，只保留最长的轮廓然后寻找最小外接圆就可以得到上图的结果。

寻找到外轮廓的同时我们还得知了外轮廓的圆心以及半径，这样就可以把外轮廓附近的的点从总轮廓中剔除出去。然后使用最小二乘法对内轮廓进行拟合，半径应为：

$$r = \sqrt{\frac{\sum_{i=0}^{n} dist(c_i,\ O)^2}{n}}$$

其中c_i指内轮廓点集中的第i个点，O指外轮廓与内轮廓的圆心，$dist(x,y)$代表x,y之间的欧氏距离。结果如下图所示：



综合上面的结果，内外轮廓如下图：