# 图像分割之分水岭算法

## 理论

任何灰度图像都可以看作是一个地形表面，其中高强度表示山峰和丘陵，而低强度表示山谷。用不同颜色的水(标签)填充每个孤立的山谷(局部极小值)。当水上升时，根据附近的峰(梯度)，不同山谷不同的颜色的水，显然会开始融合。为了避免这种情况，你在水就要融合的地方及时增加屏障（增高水坝）。你继续填满水，建造屏障，直到所有的山峰都被淹没。然后，您创建的屏障会给出分割结果。这就是分水岭背后的“哲学”。你可以访问分水岭的CMM网页，里面有动画帮助理解。

但是这种方法会由于图像中的噪声或其他不规则性因素而导致过度分割的结果。OpenCV实现了一种基于标记的分水岭算法，你可以指定哪些是要合并的谷点，哪些不是。我们所做的是给我们所知道的对象赋予不同的标签（marker）。用一种颜色(或强度)标记我们确定的为前景或对象的区域，用另一种颜色标记我们确定为背景或非对象的区域，最后用0标记我们不确定的区域。然后应用分水岭算法，其将使用我们给出的标签进行更新（填水），对象的边界值将为-1。

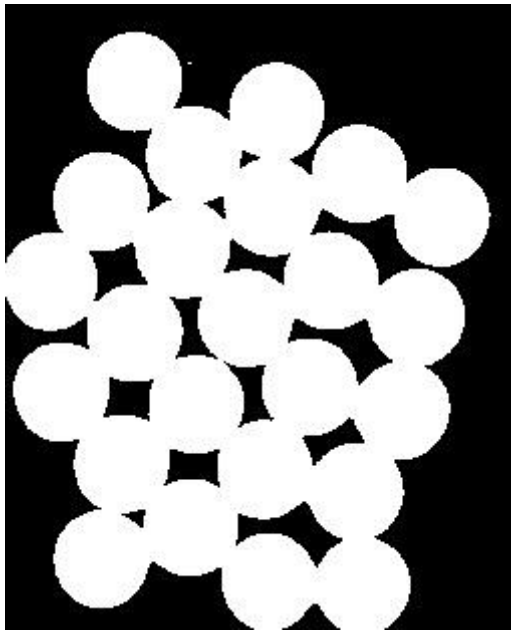
## 代码

下面我们将看到一个关于如何使用距离变换（Distance Transform）和分水岭（watershed）分割相互接触的对象的例子。下面的图像，硬币互相接触。

首先使用Otsu的二值化方法把图片变成二值图像。

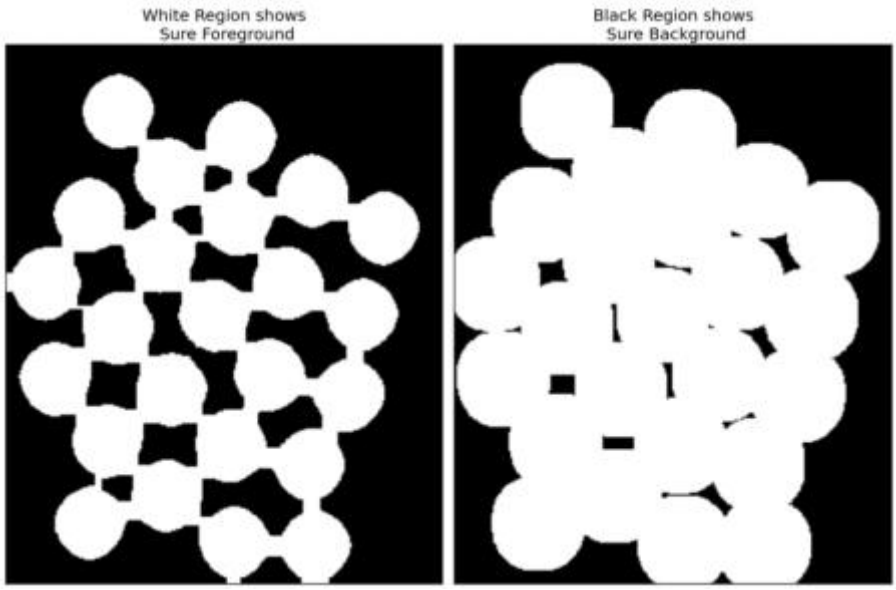
```
import numpy as np
import cv2
img = cv2.imread('E:/pictures/coins.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)
```

结果为：



现在我们需要去除图像中的一些小的白色噪声，我们可以使用形态学开运算。为了去除物体上的小洞（黑色噪声），我们可以使用形态闭运算。现在我们可以确定，靠近物体（硬币）中心的区域是前景，远离物体的区域是背景。还有我们不确定的区域就是是硬币的边界区域。

所以我们需要提取的是硬币的区域。用侵蚀法磨损边界像素。剩下部分，我们都能确定是硬币。如果物体不互相接触，这个方法就很明显确定彼此之间的硬币。但是由于它们彼此接触，另一个好的方法是距离变换法并使用一个合适的阈值。接下来我们需要找到确定不是硬币的区域。用膨胀法增加了物体到背景的边界。这样，因为硬币边界区域被扩展，我们就可以确保此结果中的任何背景区域包含于源图像背景。请看下图。

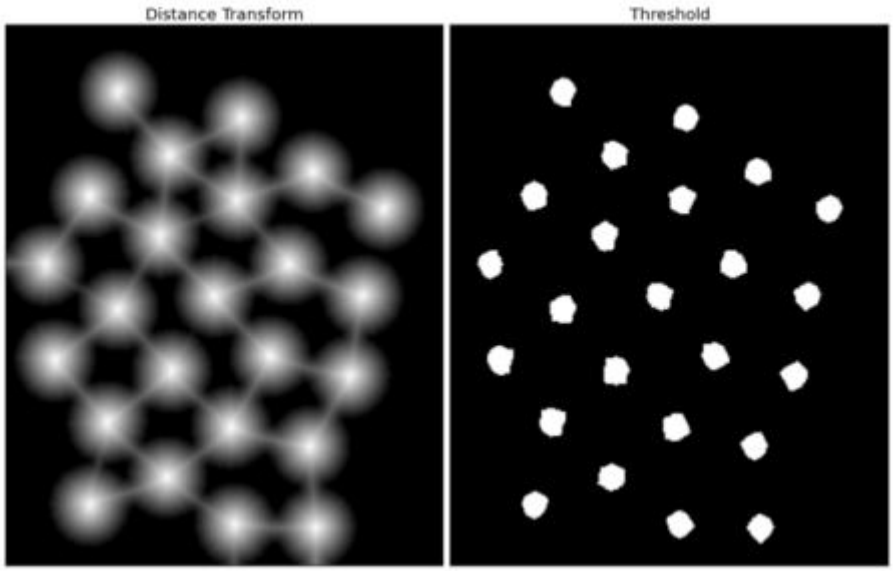



剩下的区域是我们不知道的，可能是硬币也可能是背景。用分水岭算法应该能找到它。这些区域通常在硬币的边界附近，也就是前景和背景相遇的地方(或者是两种不同硬币相遇的地方)。用sure_bg减去sure_fg可以得到（看最终结果second分窗图）。

```
# noise removal
kernel = np.ones((3,3),np.uint8)
opening = cv2.morphologyEx(thresh,cv2.MORPH_OPEN,kernel, iterations = 2)
# sure background area
sure_bg = cv2.dilate(opening,kernel,iterations=3)
# Finding sure foreground area
```

```
dist_transform = cv2.distanceTransform(opening,cv2.DIST_L2,5)
ret, sure_fg = cv2.threshold(dist_transform,0.7*dist_transform.max(),255,0)
# Finding unknown region
sure_fg = np.uint8(sure_fg)
unknown = cv2.subtract(sure_bg,sure_fg)
```

看结果。我们得到了硬币的一些区域，我们确定这些区域是硬币，现在它们被分离了。(在某些情况下，您可能只对物体大概分割成几个感兴趣，而不是物体的相对更精确的分割情况。在这种情况下，你不需要使用距离变换，只要侵蚀就足够了。)

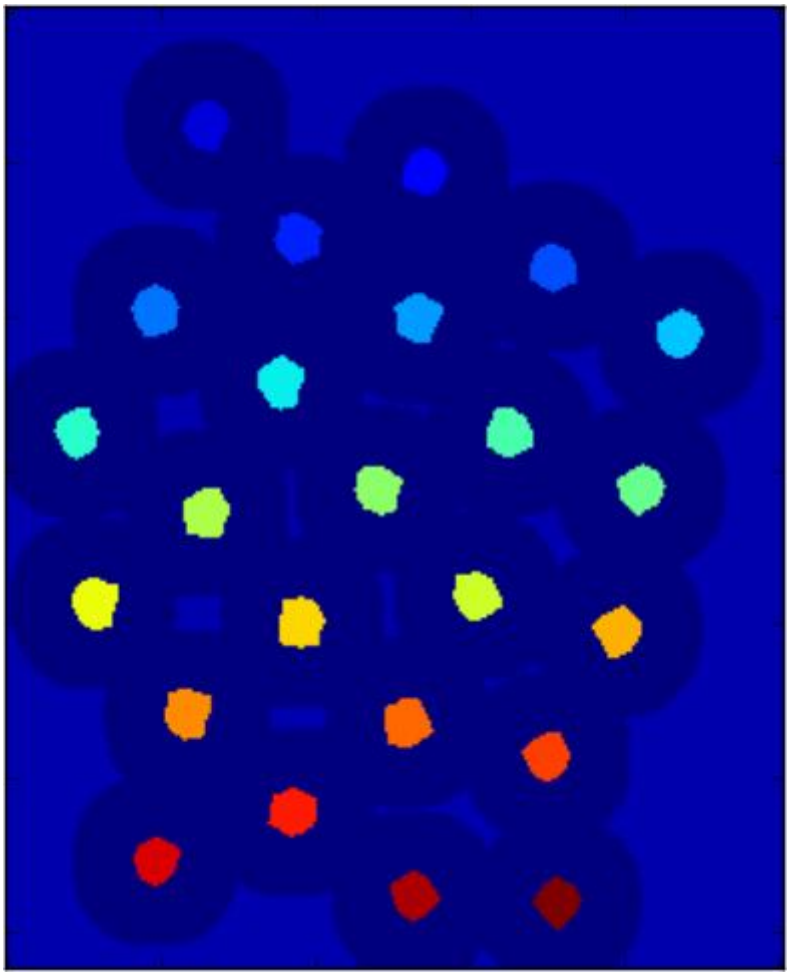



现在我们确定了哪些是硬币的区域，哪些是背景等等。因此，我们创建了marker(它是一个与原始图像大小相同的矩阵，但是使用int32数据类型)，并在其中标记区域。我们确定的区域(无论是前景还是背景)被标记为任意正整数，但是不同的整数，而我们不确定的区域则被保留为零。为此，我们使用了cv2.connectedComponents()。它用0来标记图像的背景，然后用从1开始的整数来标记其他对象。

但是我们知道，如果将background标记为0,watershed()方法将认为它是未知区域。所以我们要用不同的整数来标记它。相反，我们将标记未知区域为0。

```
# Marker labelling
ret, markers = cv2.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
```

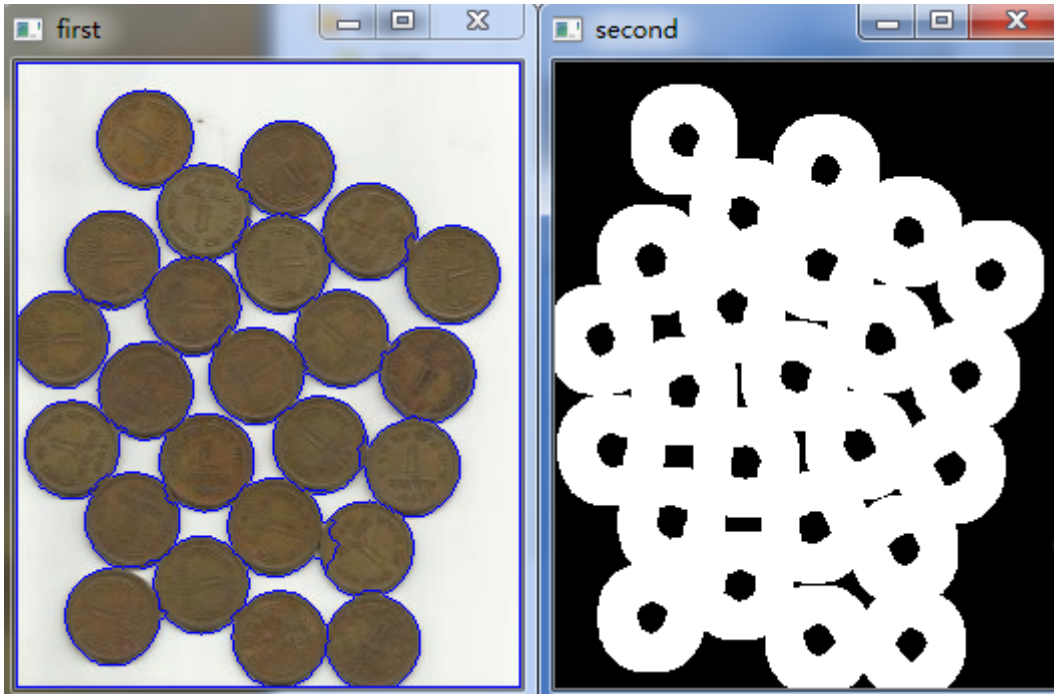
查看colormap中显示的结果。深蓝色区域表示未知区域。硬币的颜色是不同的。与未知区域相比，背景确定的剩余区域以浅蓝色显示。



现在是最后一步，应用分水岭，边界区域将标记为-1。

```
markers = cv2.watershed(img,markers)
img[markers == -1] = [255,0,0]
cv2.namedWindow('first',cv2.WINDOW_AUTOSIZE)
cv2.imshow('second',unknown)
cv2.imshow('first',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

最终结果：


first
second