

数字图像处理

第四周课堂练习

李竹

杭州电子科技大学

电子信息学院



练习1 图像形态学

基本形态学函数

0.形状



1.尺寸



2.锚点, 默认值 (-1, -1) 表示锚点位于元素中心。



```
Mat getStructuringElement(int shape, Size ksize, Point anchor = Point(-1,-1));
```

0.输入



1.输出



2.结构元素



```
void erode( InputArray src, OutputArray dst, InputArray kernel,  
            Point anchor = Point(-1,-1), int iterations = 1,  
            int borderType = BORDER_CONSTANT,  
            const Scalar& borderValue = morphologyDefaultBorderValue() );
```

```
void dilate( InputArray src, OutputArray dst, InputArray kernel,  
            Point anchor = Point(-1,-1), int iterations = 1,  
            int borderType = BORDER_CONSTANT,  
            const Scalar& borderValue = morphologyDefaultBorderValue() );
```

练习1 图像形态学

形态学综合函数

2.操作类型

```
enum MorphTypes{  
    MORPH_ERODE    = 0,  
    MORPH_DILATE    = 1,  
    MORPH_OPEN      = 2,  
    MORPH_CLOSE     = 3,  
    MORPH_GRADIENT  = 4,  
    MORPH_TOPHAT    = 5,  
    MORPH_BLACKHAT  = 6,  
    MORPH_HITMISS   = 7,  
};
```

0.输入



1.输出



3.结构元素

```
void morphologyEx( InputArray src, OutputArray dst,  
    int op, InputArray kernel,  
    Point anchor = Point(-1,-1), int iterations = 1,  
    int borderType = BORDER_CONSTANT,  
    const Scalar& borderValue = morphologyDefaultBorderValue() );
```


练习1 图像形态学

对下图先进行二值化，然后分别进行腐蚀、膨胀、开运算和闭运算。



练习2 连通域标记

对下图先进行二值化，然后进行连通域标记，并绘制出每个连通域的外接四边形（bounding box），并使用 `cout <<`，将硬币的个数输出至状态栏。



Opencv中的连通域标记函数

connectedComponentsWithStats函数，可以对黑白二值图进行连通域标记，同时返回连通域的状态和中心坐标。

函数原型：

```
CV_EXPORTS_W int connectedComponentsWithStats(InputArray image, OutputArray labels,  
OutputArray stats, OutputArray centroids,  
int connectivity = 8, int ltype = CV_32S);
```

0.函数返回值，连通域数量，包括背景

1.输入，二值化图像

2.输出，size和输入图像一致，记录每个像素的连通域标签号

3.输出 状态矩阵，
size=连通域数量×5

5.使用8-领域或4-领域

4.记录连通域中心的矩阵，
size=连通域数量×2

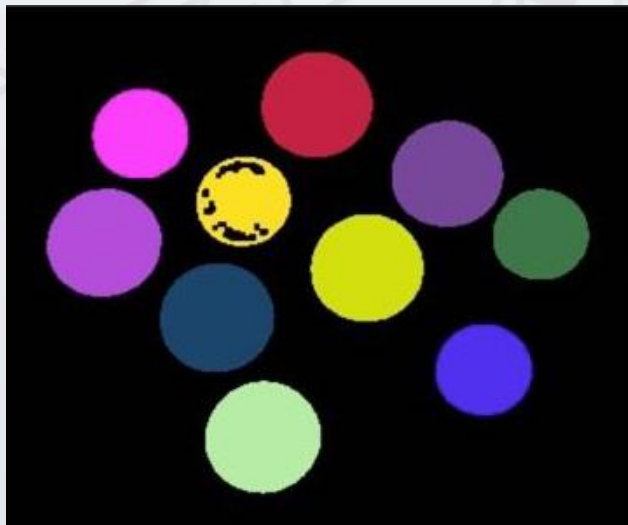
6.输出标签的数据类型，
可选CV_32S 或者 CV_16U 。

Opencv中的连通域标记函数

3.连通域中心的矩阵

```
CV_EXPORTS_W int connectedComponentsWithStats(InputArray image, OutputArray labels,  
OutputArray stats, OutputArray centroids,  
int connectivity = 8, int dtype = CV_32S);
```

包括背景共有11连通域



中心坐标矩阵size=11×2

[185.9378797560267, 157.9491606157421;
181.318391562294, 56.63941990771259;
80.24148745519713, 73.30913978494624;
256.1298701298701, 96.31331168831169;
140.192078133478, 111.6370048833424;
59.49531542785759, 135.6611492816989;
309.5810870525851, 131.0565620857269;
210.0705394190871, 150.1184168528567;
124.0959455277004, 178.6146703806871;
276.9467016164264, 208.4713848842289;
150.4366366366366, 247.0654654654655]

第一行为背景

statsMat.at<int>(连通域序号, 状态);

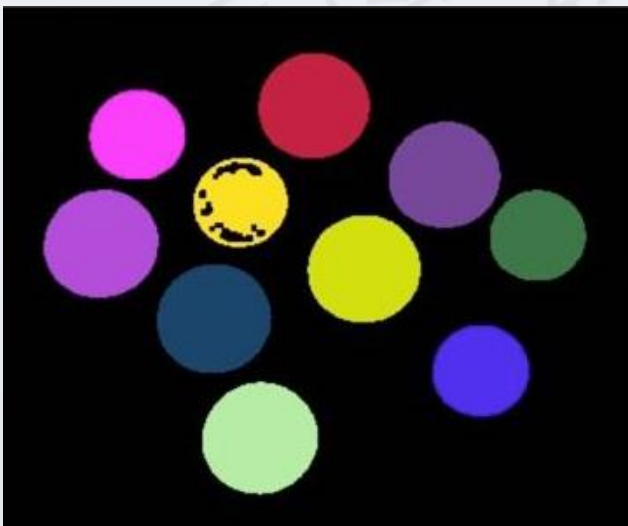
每一行为一个连通域的中心坐标 [x, y]。

Opencv中的连通域标记函数

3.状态矩阵

```
CV_EXPORTS_W int connectedComponentsWithStats(InputArray image, OutputArray labels,  
OutputArray stats, OutputArray centroids,  
int connectivity = 8, int ltype = CV_32S);
```

包括背景共有11连通域



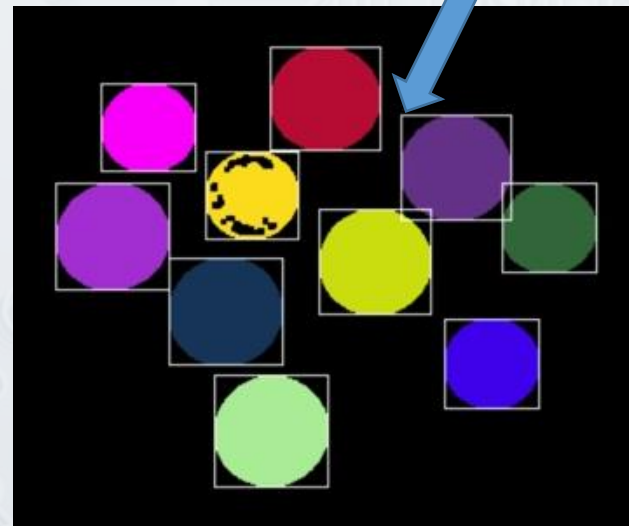
状态矩阵size=11×5

[0, 0, 368, 309, 86075;
150, 27, 64, 60, 3034;
53, 48, 55, 51, 2232;
225, 66, 64, 61, 3080;
113, 87, 54, 51, 1843;
27, 105, 66, 62, 3202;
283, 105, 55, 52, 2263;
178, 120, 65, 61, 3133;
92, 148, 66, 62, 3231;
250, 183, 55, 52, 2289;
118, 215, 66, 65, 3330]

第一行为背景

[x, y]为左上角

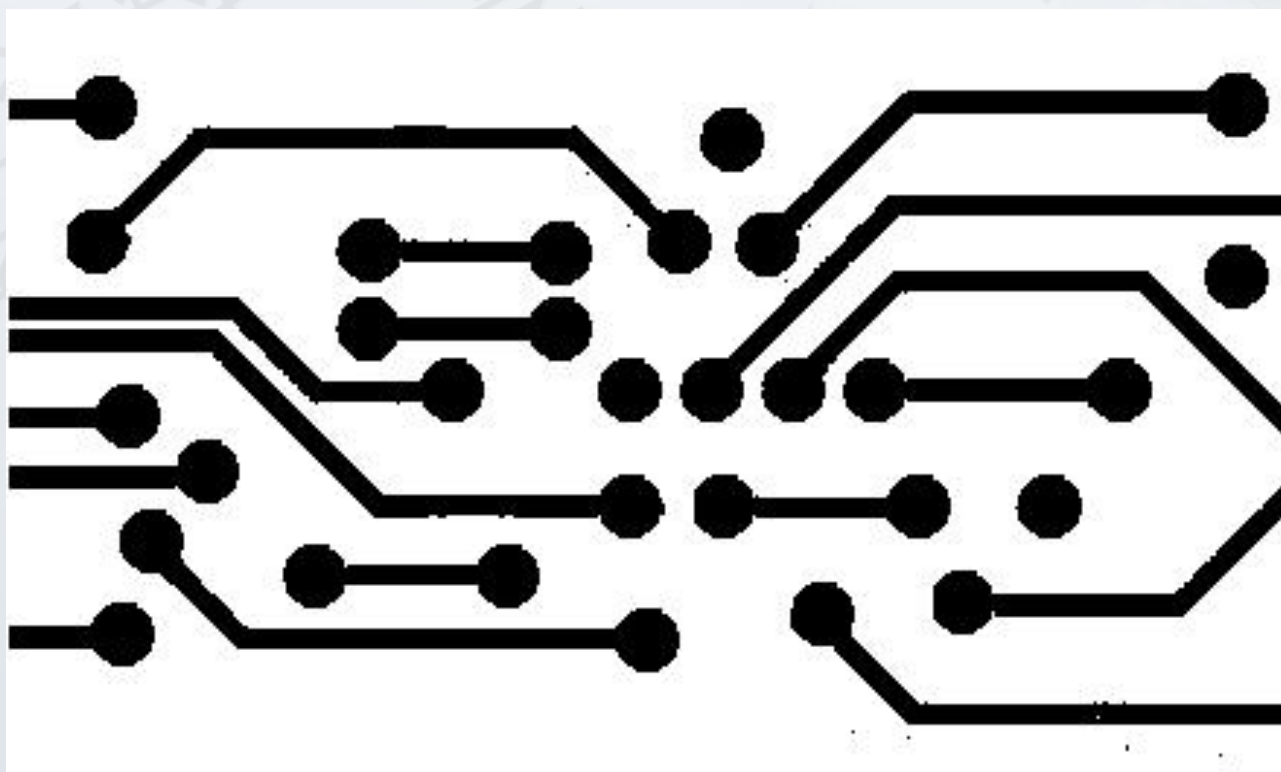
bounding box



每行5个参数分别为该连通域最小外接四边形
(bounding box) 的 x, y, width,height和面
积 (像素数量)。

练习3

使用大津法分割下图，并对其进行连通域标记，利用图像形态学中所学的知识实现自动计算原点个数。



练习4

对以下图片进行自动计数。

