计算机科学与技术 学院

计算机视觉 课程实验报告

学号：	姓名：	班级：
实验题目： 图像结构之连通域和距离变换函数		

实验内容：

实验 6.1 连通域

- 实现图像的快速连通域算法，可以提取出图像中的连通域，并将不同连通域用不同颜色显示。
- 对一个二值图像，删除较小的前景区域，只保留最大的一个。

实验 6.2 距离变换

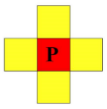
- 了解 OpenCV 的距离变换函数 distanceTransform，使用合适的测试图像进行测试，并将距离场可视化输出。

实验过程中遇到和解决的问题：

（记录实验过程中遇到的问题，以及解决过程和实验结果。可以适当配以关键代码辅助说明，但不要大段贴代码。）

一、 快速连通域算法的实现

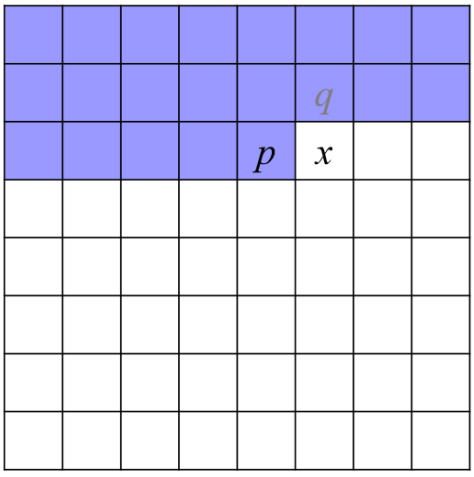
首先对于输入的源图像只进行一遍扫描，采用如图的四领域结构来判断连通域，采用一个 vector 容器来存储对图像连通域的标记信息，其中对于二值图像，黑色为背景像素（0），白色为前景像素（255），对前景像素进行标记 label，后景都标记为-1。



四邻域

扫描时，利用一次扫描+合并等价类实现，采用顺序扫描，从左到右，对于中间的像素有四种情况分别处理如下：

- **一次扫描 + 合并等价类**

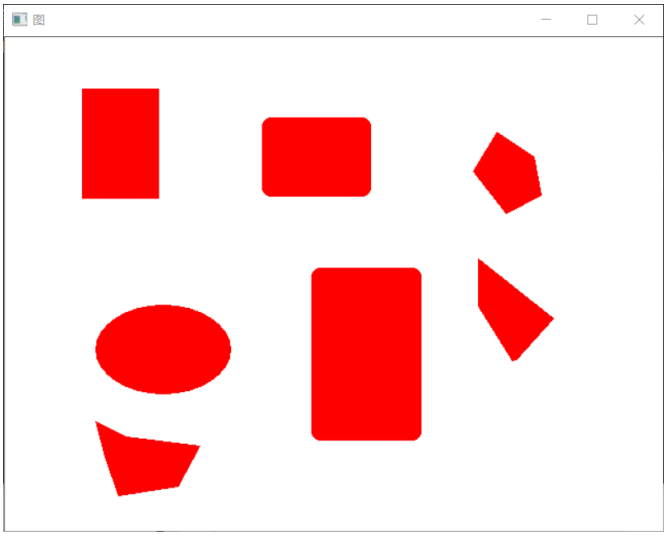


- *x!=p && x!=q*
- *x=p 或 x=q*
- *x=p=q, L(p)=L(q)*
- *x=p=q, L(p)!=L(q)*

具体相关代码如下：

```
//余下rows-1行
for (int i = 1; i < inputImage.rows; i++)
{
    const uchar* lastrowData = inputImage.ptr<uchar>(i-1);//上一行的行指针
    const uchar* rowData = inputImage.ptr<uchar>(i);//本行的行指针
    if(rowData[0]==0)
        labelImg.push_back(-1);
    else {
        if ( 1 == lastrowData[0])
            labelImg.push_back(labelImg[(i-1)*cols]);
        else {
            count++;
            labelImg.push_back(++labelNum);
        }
    }
    for (int j = 1; j < cols; j++)
    {
        int a = labelImg[(i - 1)*cols + j];
        int b = labelImg[i*cols + j - 1];
        if (rowData[j] == 0)
            labelImg.push_back(-1);
        else {
            if (0 == rowData[j - 1] && 0 == lastrowData[j]) {
                count++;
                labelImg.push_back(++labelNum);
            }
            else if (1 == rowData[j - 1] && 0 == lastrowData[j])
                labelImg.push_back(b);

            else if (0== rowData[j - 1] && 1 == lastrowData[j])
                labelImg.push_back(a);

            else {
                if (a != b) {
                    for (size_t k = 0; k < labelImg.size(); k++)
                    {
                        if (labelImg[k] == b)
                            labelImg[k] = a;
                    }
                    labelImg.push_back(a);
                    count--;
                }
                else {

                    labelImg.push_back(a);
                }
            }
        }
    }
}
```

对于得到的标记向量，连通域将会被标记为同样的数字，vector 的大小将等于图片的像素值。提取后对于大于 0 标记的像素赋为红色如图：

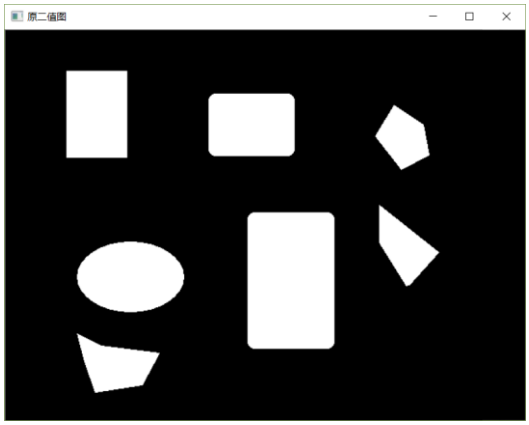


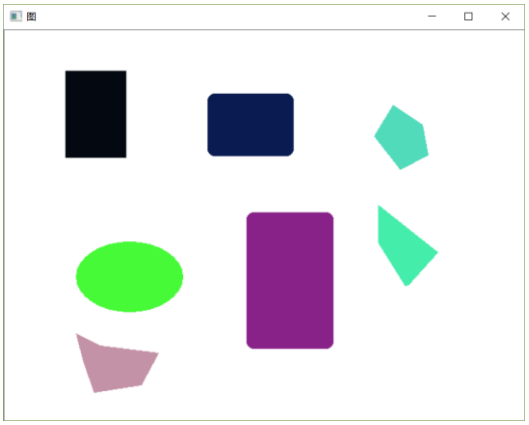
二、 对不同的连通于进行着色

先创建三通道的目标显示图像，而后根据标记值赋予不同的颜色，如下所示：

```
void bepaint(const Mat& simg,Mat& dimg, vector<int>& labelImg) {
    int rows = simg.rows;
    int cols = simg.cols;

    dimg.release();
    dimg.create(rows, cols, CV_8UC3);
    dimg = Scalar::all(0);

    for (int i = 0; i < rows; i++) {

        for (int j = 0; j < cols; j++) {
            int x = labelImg[i*cols + j];

            if (x == -1) {
                dimg.at<Vec3b>(i, j)[0] = 255;
                dimg.at<Vec3b>(i, j)[1] = 255;
                dimg.at<Vec3b>(i, j)[2] = 255;
            }
            else{
                dimg.at<Vec3b>(i, j)[0] = (x*x*x*x) % 255;
                dimg.at<Vec3b>(i, j)[1] = (x*x*x) % 255;
                dimg.at<Vec3b>(i, j)[2] = (x*x) % 255;
            }
        }
    }
}
```

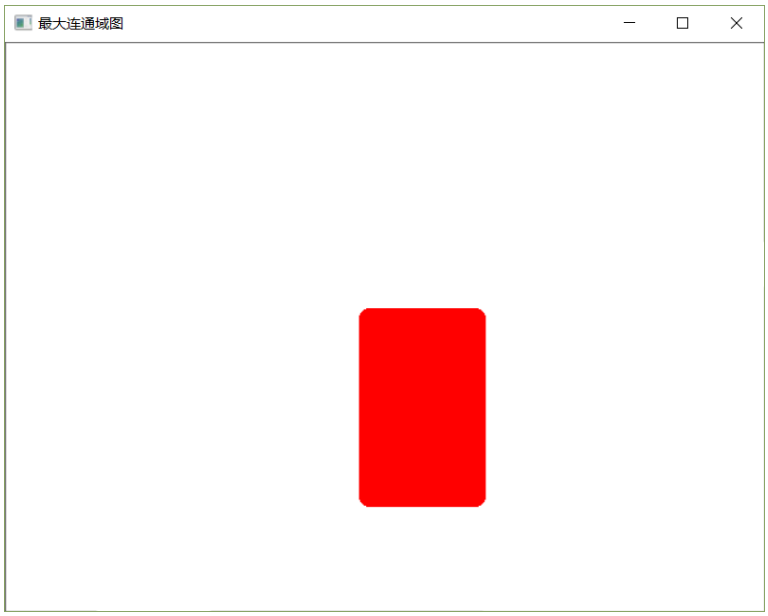
效果如下：





三、 删除较小的前景区域，保留最大连通域

对于一遍扫描后可以得到的标签数目（连通域的个数），可以动态的建立 label 结构体数组，对每个连通域的像素点个数进行遍历统计，最后得到最大的连通域，进行显示如下：



四、 距离变换函数

功能：用来计算原图像中距离变换图像；

```
void distanceTransform( InputArray src,
OutputArray dst,
OutputArray labels,
int distanceType,
int maskSize,
int labelType=DIST_LABEL_CCOMP );
```

函数说明：

用于计算图像中每一个非零点像素与其最近的零点像素之间的距离，输出的是保存每一个非零点与最近零点的距离信息；图像上越亮的点，代表了离零点的距离越远。

参数：

src 是单通道的 8bit 的二值图像（只有 0 或 1）

dst 表示的是计算距离的输出图像，可以使单通道 32bit 浮点数据

distanceType 表示的是选取距离的类型，可以设置为 CV_DIST_L1,CV_DIST_L2,CV_DIST_C 等，具体如下：

```
  DIST_L1              = 1,      //!< distance = |x1-x2| + |y1-y2|
  DIST_L2              = 2,      //!< the simple euclidean distance
  DIST_C               = 3,      //!< distance = max(|x1-x2|,|y1-y2|)
  DIST_L12             = 4,      //!< L1-L2 metric: distance =2(sqrt(1+x*x/2) -
1))
   DIST_FAIR           = 5,      //!< distance = c^2(|x|/c-log(1+|x|/c)),c =
1.3998
  DIST_WELSCH = 6,    //!< distance = c^2/2(1-exp(-(x/c)^2)), c= 2.9846
  DIST_HUBER   = 7     //!< distance = |x|<c ? x^2/2 :c(|x|-c/2), c=1.345
```

maskSize 表示的是距离变换的掩膜模板，可以设置为 3，5 或 CV_DIST_MASK_PRECISE，对 CV_DIST_L1 或 CV_DIST_C 的情况，参数值被强制设定为 3，因为 3×3 mask 给出 5×5 mask 一样的结果，而且速度还更快。

labels 表示可选输出 2 维数组；

labelType 表示的是输出二维数组的类型；

先利用高斯滤波和 threshold 函数对原图进行优化和阈值化（二值化）处理得到输入图像，而后调用 distanceTransform 函数得到距离场图像显示如下：

利用距离场图像可以用来细化轮廓和查找物体质心，如下：