姓名

- 倪诗宇

学号

- 201900180065

实验日期

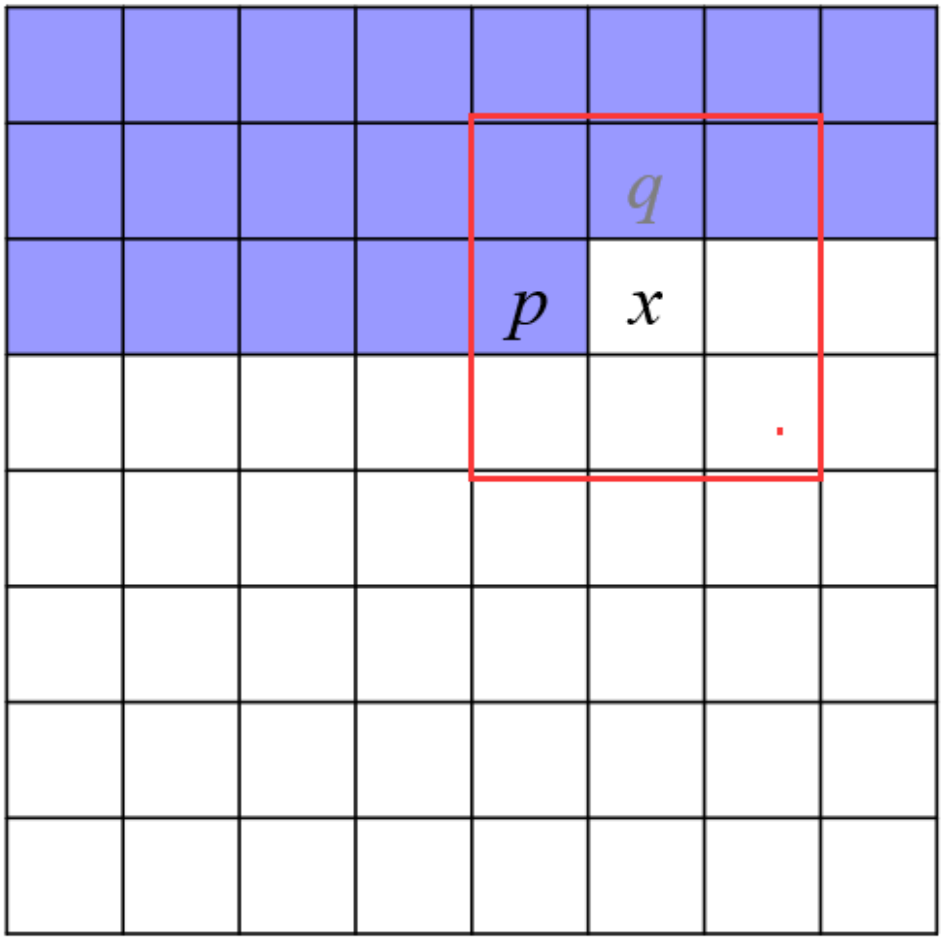
- 2021.10.23

实验题目

- 连通域与距离变换

实验过程中遇到的问题和解决办法

- 问题一：
  - 问题：不知道如何对周围的值进行并查集操作



后来发现，图像是单通道的，一个像素要么是黑，要么是白。而我们要找的连通域肯定是白色的，因此如果 x 位置的像素不是白色的，那么就不用处理。如果 x 位置的像素是白色的，这时候我们去看 x 左边一个和上面三个像素，如果是白色，直接与 x 进行合并，因为白色都是一样的，可以直接归为同一类

- 本次实验没有遇到很多问题

结论分析与体会

- 将三通道图像转为灰度图的函数

```
cvtColor(source_image,binary_image , COLOR_RGB2GRAY);//转为灰度图
```

  - 第一个参数是原图像
  - 第二个参数是目标图像
  - 第三个参数是转换的代码或标识，即在此确定将什么制式的图片转换成什么制式的图片
- 计算距离场的函数

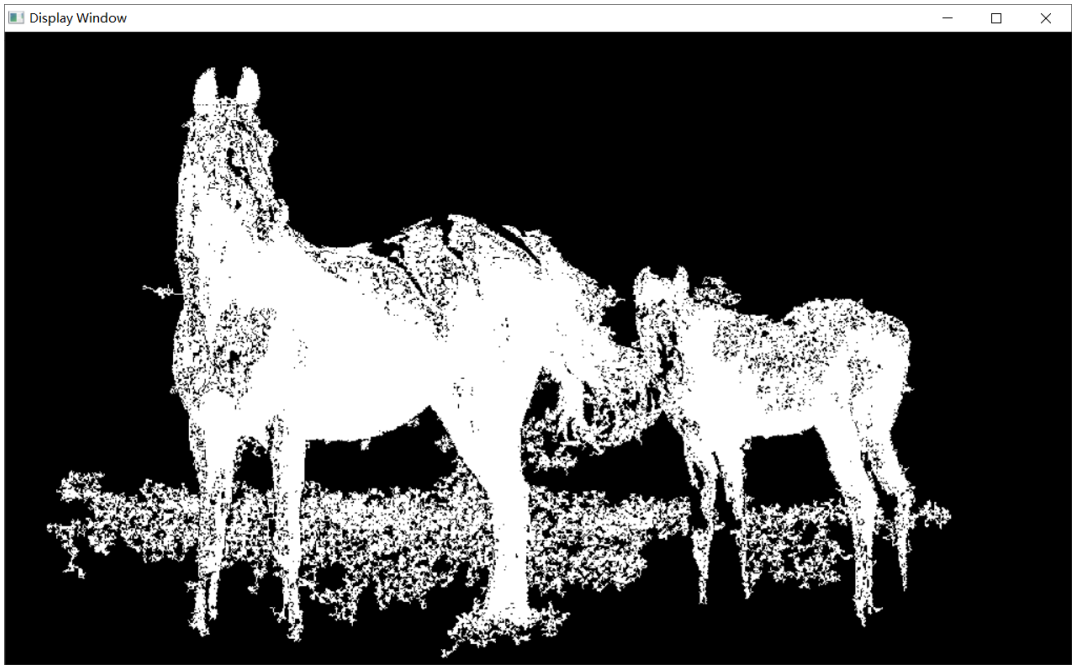
```
distanceTransform(triangle_image, target_image , DIST_L2, 3); //欧式距离 ,  3 * 3的mask
```

  图像的距离变换实现了像素与图像区域的距离变换，使得最后生成的图像在该自己元素位置处的像素为0，临近的背景的像素具有较小的值，且随着距离的增大它的的数值也就越大。对于距离图像来说，图像中的每个像素的灰度值为该像素与距离其最近的背景像素间的距离，也就是说，给每个像素赋值为离它最近的背景像素点与其距离，一幅二值图像的距离变换可以提供每个像素到最近的非零像素的距离。

  - 第一个参数是原图像
  - 第二个参数是计算后的图像
  - 第三个是使用什么距离
  - 第四个是mask的大小
- 连通域实验过程
  - 把图像转为灰度图，并以127位界限，化成 0 和 255
  - 并查集函数将二维图以一维的方式存储

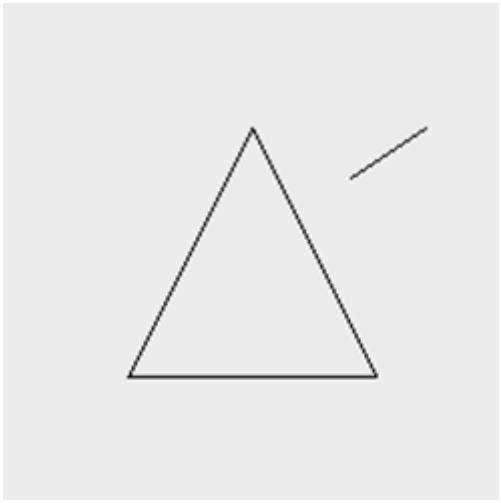
```
int calcu(int x , int y){
    return x * binary_image.cols + y;
}
```

  - 之后遍历整个灰度图，进行并查集操作
  - 寻找最大的连通域
  - 不是这个连通域内的点的像素都设为0
  - 显示输出



- 距离变换实验过程
  - 先转换为灰度图
  - 然后计算距离场，归一化

    原图：



```
distanceTransform(triangle_image, target_image , DIST_L2, 3); //欧式距离 ,  3 * 3的mask
cout << target_image.type(); // 5,代表CV_32F。保存的数据类型是32位浮点类数据
normalize(target_image, target_image, 0, 1, NORM_MINMAX);//imshow遇到小数会乘255，会溢出
```

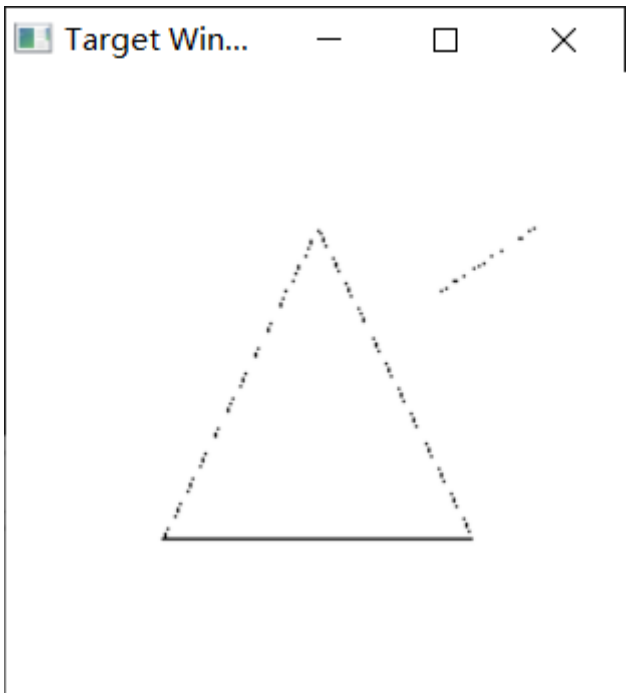
  - **imshow的说明：**

    1，如果原始图片是8位无符号整数，就按照原来的数字进行显示。也就是数字范围是[0,255]

    2，如果原始图片是16位无符号整数或者32位整数，就除以256进行显示。也就是说0到256*256的范围被压缩到0到255。

    3，如果图片是32位或者64位的浮点类型数据，那么像素值就会乘以255。也就是说，0到1的范围被映射到0到255.

  - 不归一化输出的结果



- 最后显示