山东大学 计算机科学与技术 学院

计算机视觉 课程实验报告

学号：201900130151	姓名： 莫甫龙	
实验题目：图像结构		

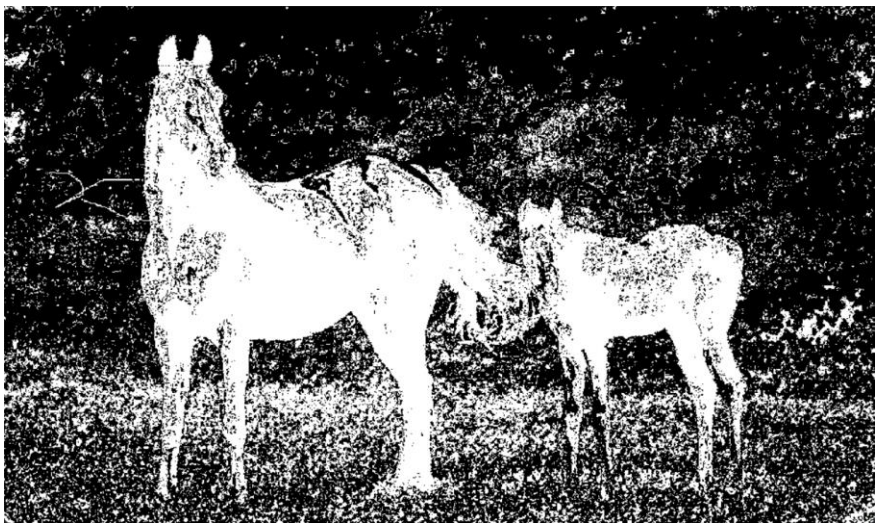
实验过程中遇到和解决的问题：

（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。）

1. 连通域

首先先将图像转化为二值图像，方便后面的处理，这边阈值设为 127。

```
Mat image = imread( filename: "C:\\Users\\15339\\Desktop\\6.png");
Mat out;
Mat gray_image;
cvtColor( src: image,  dst: gray_image,  code: CV_BGR2GRAY);
threshold( src: gray_image,  dst: out,  thresh: 127,  maxval: 255,  type: THRESH_BINARY);
modify( &: out);
```



接着就是实现 8 连通的快速连通域算法，和 4 连通的快速连通域算法差不多，都是要进行两遍遍历，第一次是将每个白色的像素点都给上一个标记，第二次就是处理等价对。而处理等价对，我使用的是并查集。八连通的标记算法用的是如下算法，在其上稍微做了一点改变：

1) 判断此点八邻域中的最左，左上，最上，上右点的情况。如果都没有点，则表示一个新的区域的开始。
2) 如果此点八邻域中的最左有点，上右都有点，则标记此点为这两个中的最小的标记点，并修改大标记为小标记。
3) 如果此点八邻域中的左上有点，上右都有点，则标记此点为这两个中的最小的标记点，并修改大标记为小标记。
4) 否则按照最左，左上，最上，上右的顺序，标记此点为四个中的一个。

对于第二和第三种情况，可以看出是会存在等价对的情况的，在出现这种情况的时候就将当前点标记为最小的标记点，如果这两个点的标记不一样，那么就将这两个点的标记加入并查集。

```
if (y == 0)//最左边    最左边的只需要和上面、右上的比较
{
    int above = input.at<uchar>( i0: x - 1, y);
    int u_r = input.at<uchar>( i0: x - 1,  i1: y + 1);
    if (above != 0)
        label[x][y] = label[x - 1][y];
    else if (u_r != 0)
        label[x][y] = label[x - 1][y + 1];
    else {
        num++;
        label[x][y] = num;
    }
    if (above != 0 && u_r != 0 && (label[x - 1][y] != label[x - 1][y + 1]))//加入并查集
        merge( i: label[x - 1][y],  j: label[x - 1][y + 1]);
```

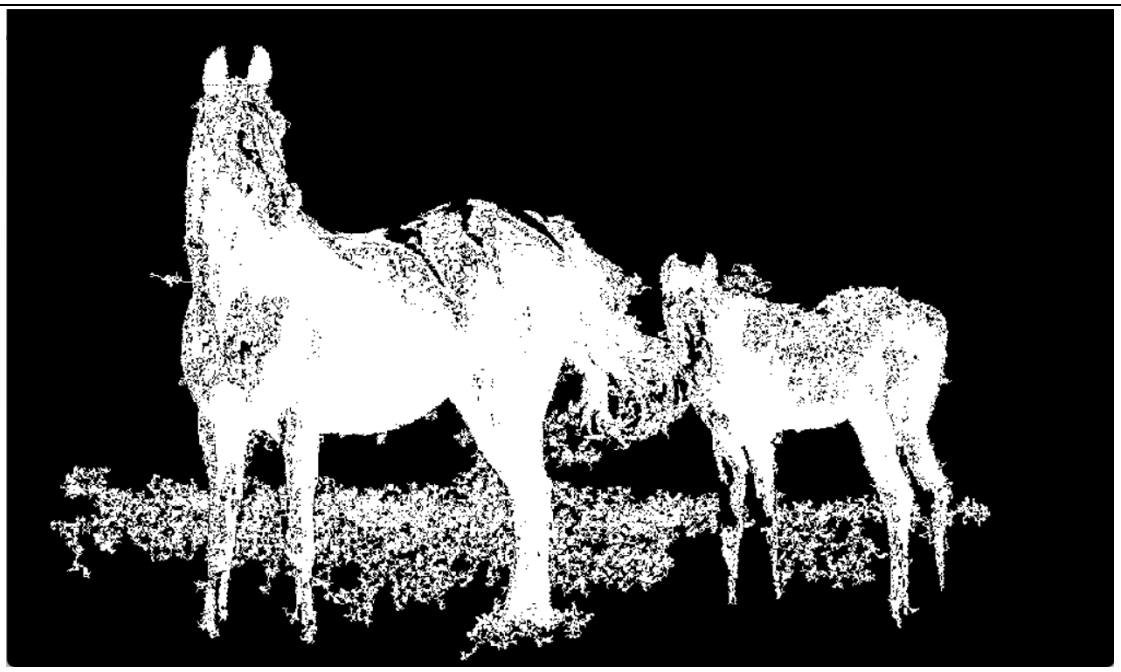
那么在进行完这一系列操作以后，我们就能得到很多棵树，这时候，只需要将每个像素点的标为置为根结点的标记即可，也就是合并连通域。

```
for (int x = 0; x < input.rows; x++) {
    for (int y = 0; y < input.cols; y++) {
        label[x][y] = find(label[x][y]);  //合并连通域
    }
}
```

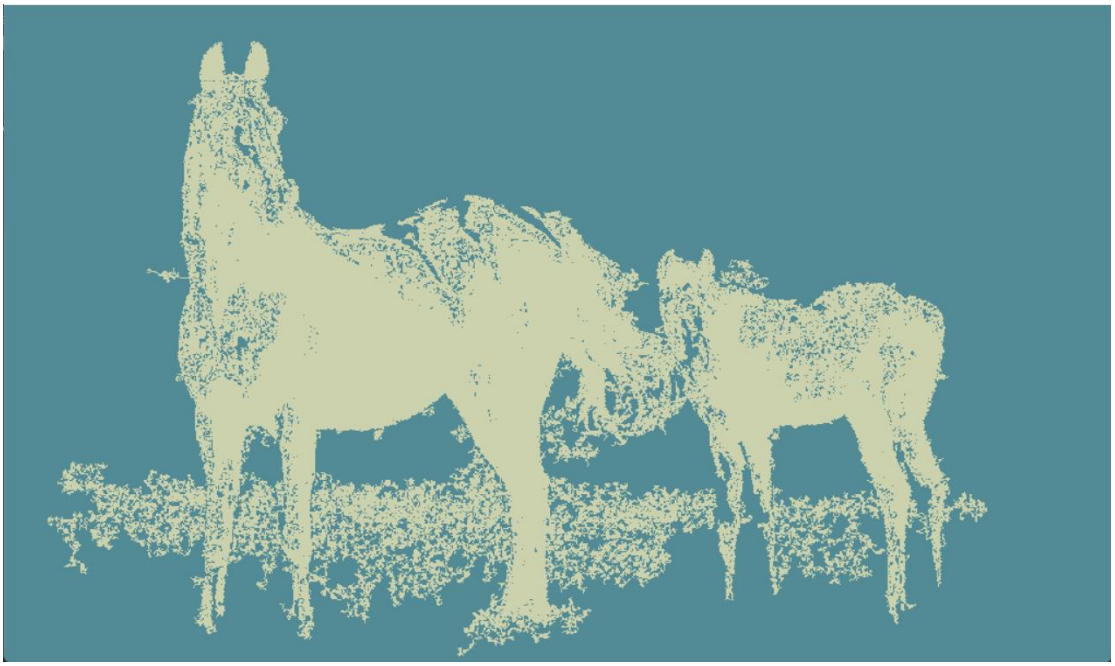
接着就是统计每个连通域的个数和连通域的面积，然后将最大面积的连通域输出即可。

```
for (int x = 0; x < input.rows; x++) {
    for (int y = 0; y < input.cols; y++) {
        if (label[x][y] != 0)  //记录每个连通域的面积
            sum1[label[x][y]]++;
        if (label[x][y] == find(label[x][y]) && label_sum[label[x][y]] == 0) {//统计总共有多少个连通域
            sum++;
            label_sum[label[x][y]]++;
        }
    }
}
for (int i = 0; i < 15000; i++) {//找到面积最大的连通域的标记
    if (max <= sum1[i]) {
        max = sum1[i];
        max_i = i;
    }
}
```

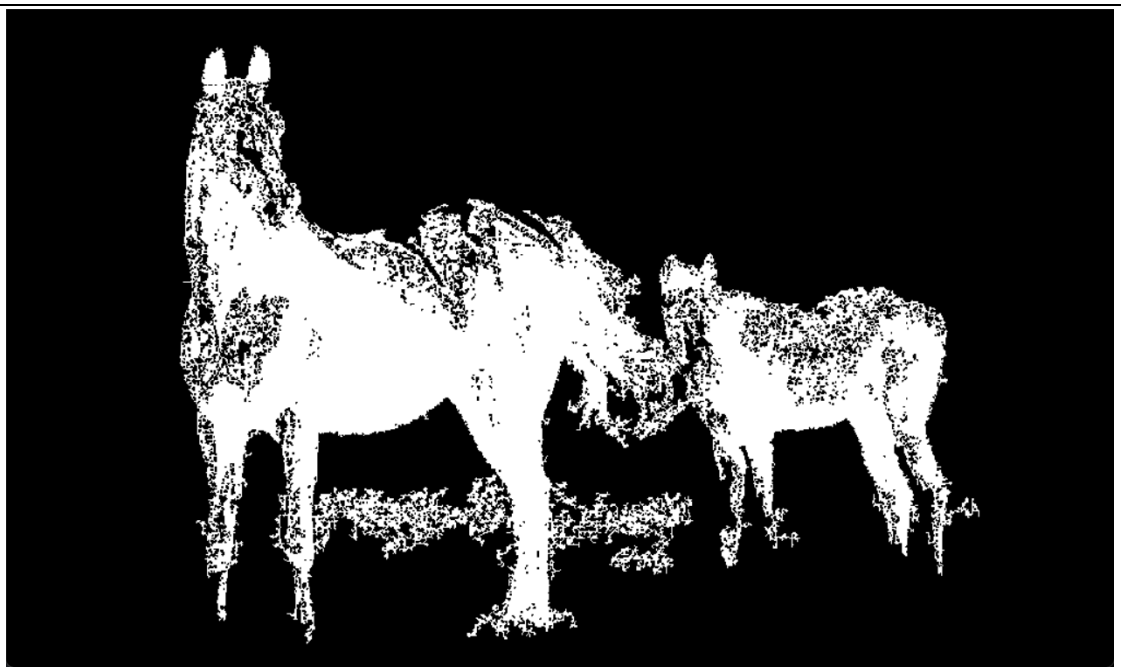
```
未合并时的连通域个数:10953
最大面积的连通域标记为:3336
总连通域:7324
```



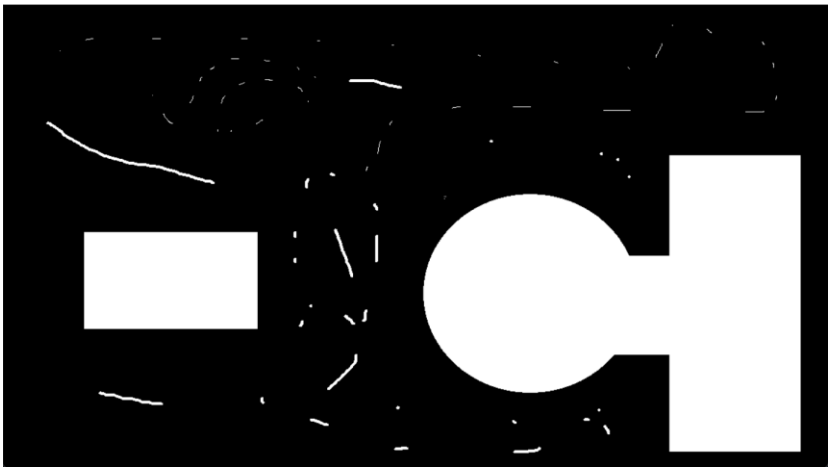
下面这张图是通过 opencv 自带的 connectedComponentsWithStats 函数计算的连通域来实现的

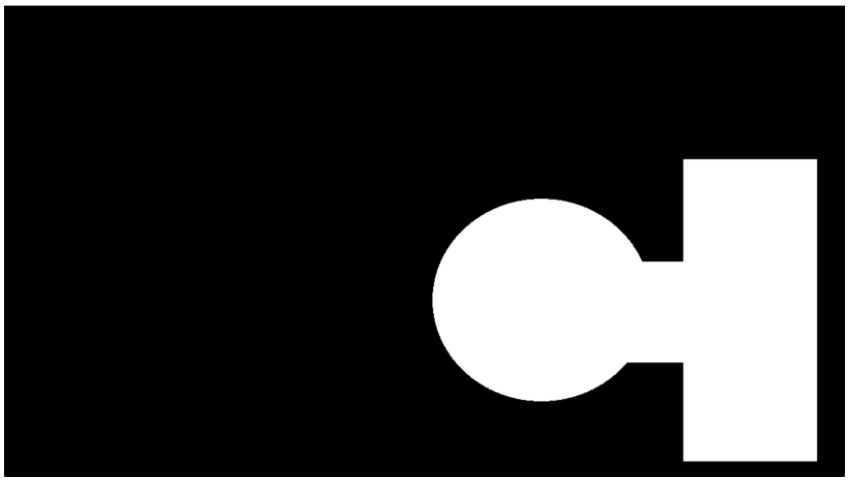


通过不断地调开始转换为二值图像的阈值发现，当阈值变大的时候，马身下的草地就越少。这是阈值为 200 时的效果图。



又换了张图来运行。





2. 距离变换：

直接调用 distanceTransformhanshu 就行。

```
    Mat image = imread( filename: "C:\\Users\\15339\\Desktop\\hand.png");
    Mat out;
    Mat gray_image;
    cvtColor( src: image,  dst: gray_image,  code: CV_BGR2GRAY);
    threshold( src: gray_image,  dst: out,  thresh: 145,  maxval: 255,  type: THRESH_BINARY);
    Mat output;
    distanceTransform( src: out,  dst: output,  distanceType: DIST_L2,  maskSize: 3,CV_32F);
    normalize( src: output,  dst: output,  alpha: 0,  beta: 1,  norm_type: NORM_MINMAX);
    imshow( winname: "hand",  mat: output);
    waitKey();
}
```

结果分析与体会：

八连通的快速连通算法，和四连通的快速连通算法比起来，就是要多考虑左上角和右上角两个点，虽然只是多了两个点，但是其中涉及的情况却多了 12 种，而上面的八邻域标记算法中，它按照左边，左上，上面，右上的顺序来标记当前点，只需要考虑左上和右上、左边和右上两种特殊情况即可，这就大大减少了考虑的情况。

而 distanceTransform 这个函数主要用于计算非零像素到最近零像素点的最短距离。一般用于求解图像的骨骼。