作业 3：地理卫星云图处理

王琛然 151220104 17721502736@163.com 17721502736

(南京大学 计算机科学与技术系, 南京 210093)

1 实现要求

提取图像中的经纬线，过滤包括大陆线在内的所有其它线条。

2 实现细节

2.1 算法思路

首先，本次实验的任务是提取经纬线，开始的想法是利用霍夫变换提取直线，但尝试后发现霍夫变换对线条的要求过高，并且经纬线中存在很多不规则曲线，所以效果并不好。

由于图像是由像素点构成的，经纬线和大陆线等需要除去的像素点构成的线的区别：经纬线比较平滑，在大范围内方向不发生较大的改变，甚至是直线，如图 1 所示；而大陆线等曲折，存在很多拐点，方向多变，如图 2 所示。因此，提取经纬线即为寻找方向没有大范围改变的线条，可以设为 45° 范围内变化，如朝右上和右行进。朝着所设定的方向深度搜索，直到所走方向没有寻求像素点。返回搜索的结果，若所走的线条长度大于一个阈值，则可以认定其不是大陆线，即可保留输出；否则删去节点。

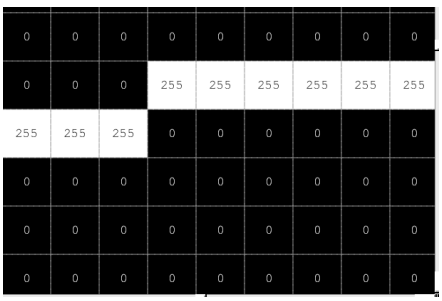


图 1.平滑的经纬线

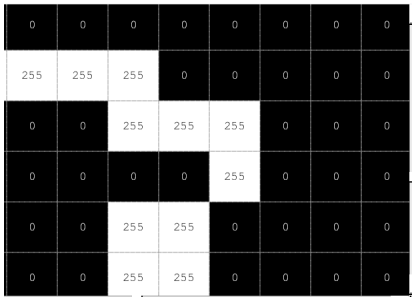


图 2.曲折的大陆线

2.2 算法实现

a. 首先设定 8 邻域方向：东、东南、南、西南、西、西北、北、东北

```
direction = [1, 0; 1, 1; 0, 1; -1, 1; -1, 0; -1, -1; 0, -1; 1, -1];
```

b. 分组：在 45 度范围内变化，分为[东、东南]; [东南、南]; [南、西南]; [西南、西]; [西、西北]; [西北、北]; [北、东北]; [东北、东]共八组，d1 为第一个方向，d2 为第二个方向

```
for d1 = 1: 8
    if d1 < 8
        d2 = d1 + 1;
    else
        d2 = 1;
    end
```

c. 对整张图进行遍历，当起始节点在原图中为白色像素点且结果图中该点是黑色像素点的时候，处理该点：遍历 8 组方向，分别进行深度搜索，调用 search(input_image, x, y, dir1, dir2)函数，返回搜索到的节点和个数。

```
[points, len] = search(image_plus, i, j, direction(d1, :), direction(d2, :));
```

d. search 函数：传入当前搜索坐标和两个方向 dir1、dir2，若方向 dir1 是白色像素点，则向 dir1 方向搜索；若方向 dir2 是白色像素点，则向 dir2 方向搜索；当搜索的两个方向都不存在白色结点，搜索停止，将当前节点加入 points 数组；除此之外，当回溯到上一层节点时，也将刚所搜索的节点加入 points 数组。

```
function [points, len] = search(input_image, x, y, dir1, dir2)
    points = [];
    len = 0;
    next1_x = x + dir1(1);
    next1_y = y + dir1(2);
    next2_x = x + dir2(1);
    next2_y = y + dir2(2);
    if(input_image(next1_x, next1_y) == 255)
        [points, len] = search(input_image, next1_x, next1_y, dir1, dir2);
        points = [points; next1_x, next1_y];
    elseif (input_image(next2_x, next2_y) == 255)
        [points, len] = search(input_image, next2_x, next2_y, dir1, dir2);
        points = [points; next2_x, next2_y];
    else
        points = [points; x, y];
    end
    len = len + 1;
end
```

e. 判断 search 函数的搜索的路径长度，设置阈值为 40：当搜索路径长度大于 40 时，可以认为其是经纬线，遍历搜索的节点，填充 output 矩阵中相应的像素。

```
if len >= 40
    points = points - 1;
    for k = 1 : len
        output(points(k, 1), points(k, 2)) = 255;
    end
end
```

f. 返回 output 矩阵为处理过的图片

2.3 search函数的合理性

该算法的重点是 search 函数，本质上是深度优先搜索算法，但有所不同的是：由于算法是对于每一个有效点遍历，所以不一定对两个方向的最长路径进行搜索，仅仅到满足搜索的停止条件时即可，简化算法实现；

同时，由于图片中的线条宽度均为 1，所以经纬线一般不会出现两个方向都存在白色像素的情况（如图 3 所示），或者说，两个方向都存在白色像素的长度不会很长，相当于一个单枝的二叉树（如图 4

所示）。所以算法可以继续进行简化：当 dir1 方向有白色像素点时即沿着 dir1 方向搜索；当 dir1 方向为黑色像素而 dir2 方向为白色像素点时沿着 dir2 方向搜索。同时，这种单枝二叉树的搜索可以有效的除去曲折的大陆线。

0	0	0	0	0	0	0	0	0
0	0	0	0	0	255	255	255	255
0	255	255	255	255	0	0	0	0
255	0	0	0	0	0	0	0	0

图 3. 经纬线像素分布

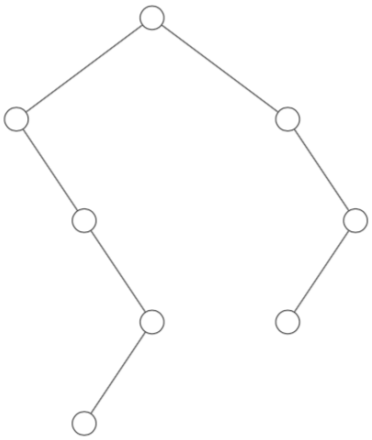


图 4. 单枝二叉树

3 实验结果

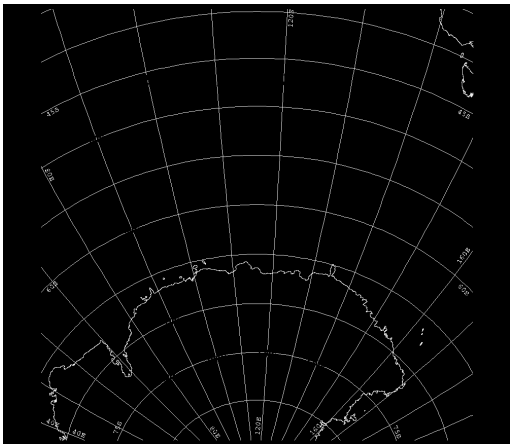
3.1 实验设置

使用 matlab2017 版， 代码在 code 文件夹中，包括：
imageprocessing_test..m：程序入口
my_imageprocessing.m：图像处理
search.m：搜索函数
image 文件夹存放五张原图和五张原图的处理结果
运行时只需运行 imageprocessing_test..m 文件
image 文件夹：存放 5 张原图和 5 张处理过的图片
doc 文件夹：实验报告

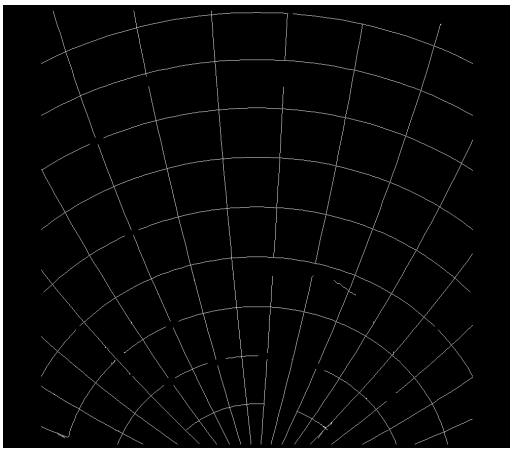
3.2 实验结果

3.2.1 视觉效果

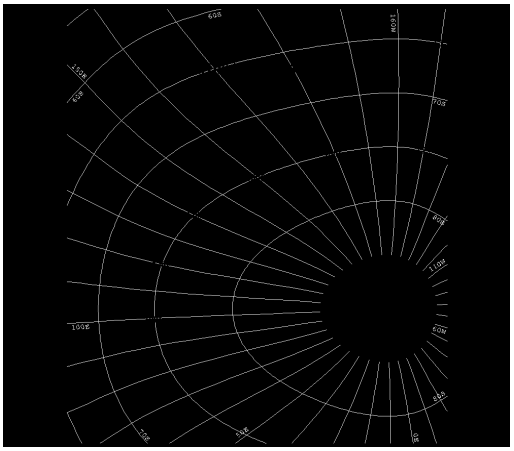
1.png:



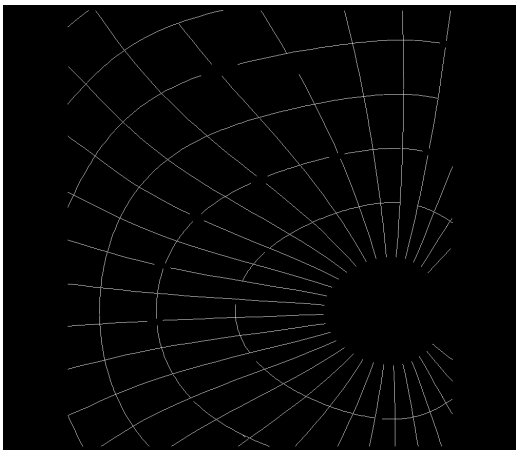
1-p.png:



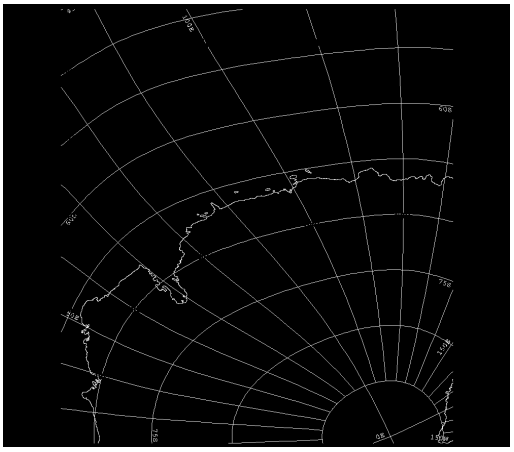
2.png:



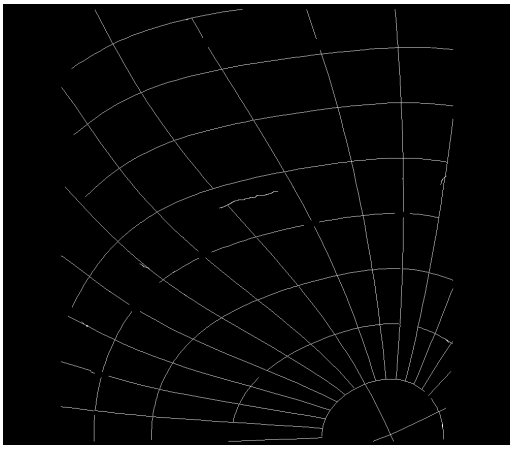
2-p.png:



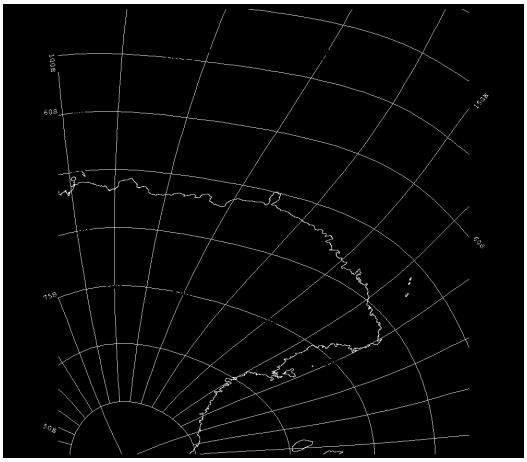
3.png:



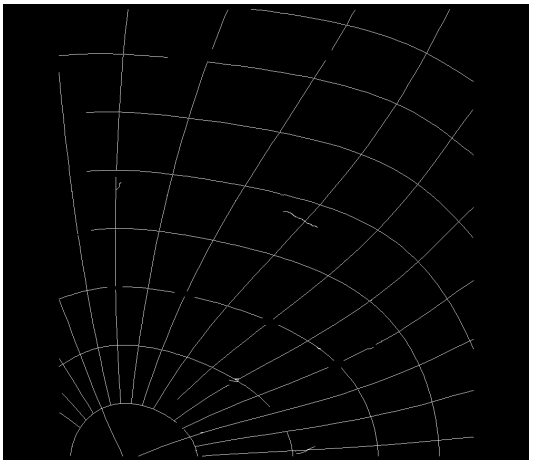
3-p.png:



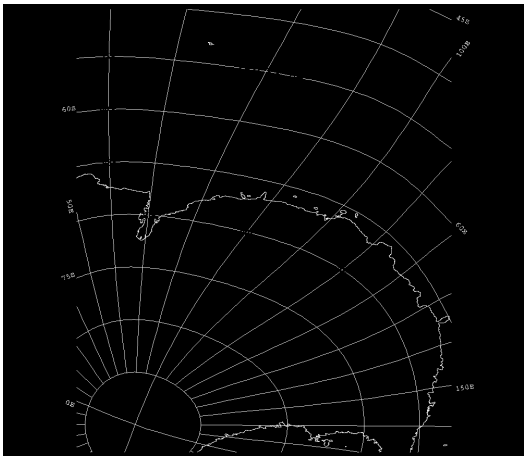
4.png:



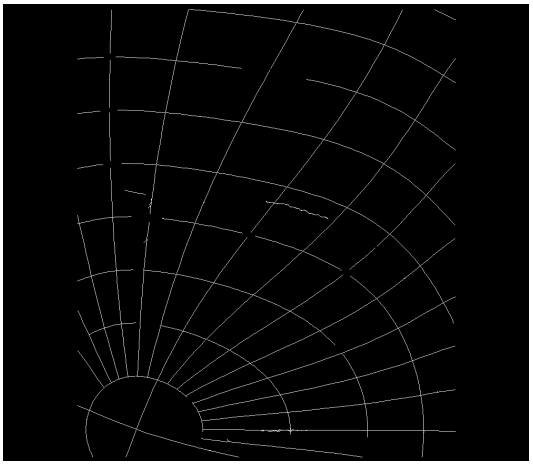
4-p.png:



5.png:



5-p.png:



3.2.2 score 评价

	Score	Score1	Score2
1.png	1.3731e+04	0.9556	0.9669
2.png	1.3113e+04	0.9525	0.9934
3.png	1.0961e+04	0.9502	0.9184
4.png	1.1376e+04	0.9607	0.9356
5.png	1.1285e+04	0.9554	0.9202

在 score 评价上，图片的冗余点剔除率基本持平，在目标像素点的保留率上，第 3、4、5 张效果略差。

4 实验总结

本次实验采用深度优先搜索的思想大体上提取了经纬线，但在第 3、4、5 张图片的结果可以看到，对于略微平滑的大陆线并不能很好的检测；而在 5 张图中或多或少都存在着当经纬线变形有拐点时，在在经纬线相交的地方或经纬线断裂处会丢失部分像素信息的问题。除此之外，由于 search 函数的定义，没有完全深度搜索，而是一种特殊形式的简化搜索，所以对于一些特殊情况，如对于存在小拐点并且位置靠近边界的像素点（线条平滑但所得的线条长度由于图像大小的限制而小于阈值）无法输出。总之，本次实验效果仍存在些许不足，希望日后可以将其完善。

5 参考文献

[1] https://www.cnblogs.com/tiandsp/archive/2013/07/05/3174262.html matlab 练习程序(广度优先搜索 BFS、深度优先搜索 DFS)