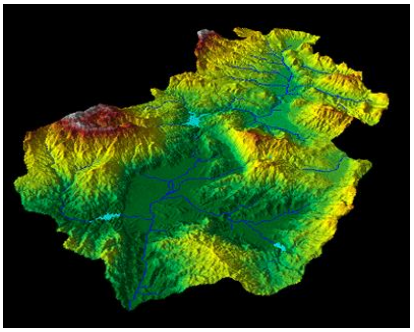
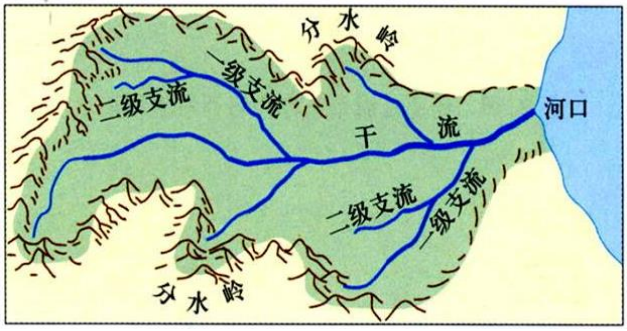
关于分水岭算法的探索

1. 算法思想

这个算法起源于分水岭算法。分水岭算法总体思想是从下往上进行计算。通过判断每次水流上涨之后区域的连通性来计算分水岭。分水岭算法是对梯度图进行分割，如果我们先将梯度图当作一幅 DEM。可以看出传统分水岭算法是从下往上进行计算，通过水流淹没地形





的思想，计算每次洪水上涨之后淹没的区域是否重叠来提取分水岭。如果我们换个思路，从上往下计算，将洪水换位降雨，计算汇流量。那么我们需要提取的位置就是汇流量为 0 的位置，也就是山脊。这个想法是上 GIS 实验的时候学习使用 arcgis 提取山脊线的时候产生的。

Arcgis 提取山脊线的思想是通过计算汇流量来实现的。先假设每一个地形栅格有 1 滴雨滴，然后让雨滴沿着地形流，计算汇流量，山脊部分就是汇流量为 0 的部分。计算汇流量之前可以先进行一次填充的操作，将小的盆地填充，让结果不出现琐碎的现象。

计算汇流量的话需要先计算每个栅格的流向，即这个栅格的雨滴会往哪个反向流。然后再通过流向计算汇流量。在 arcgis 中，结果计算了每个栅格的汇流量，但是在程序中并不需要计算出所有结果，需要计算的部分只是汇流量为 0 的栅格。

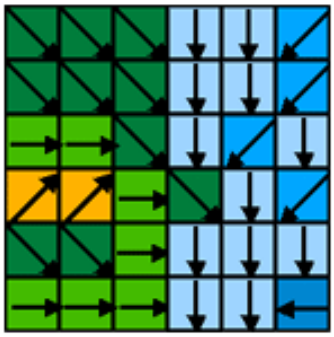
计算流向是通过三邻域进行计算，找到邻域中差距最小的栅格，然后根据栅格的位置赋予不同的数值代表不同的方向。

78	72	69	71	58	49
74	67	56	49	46	50
69	53	44	37	38	48
64	58	55	22	31	24
68	61	47	21	16	19
74	53	34	12	11	12

→

2	2	2	4	4	8
2	2	2	4	4	8
1	1	2	4	8	4
128	128	1	2	4	8
2	2	1	4	4	4
1	1	1	1	4	16

然后再通过流向来计算汇流量，汇流量为 0 的栅格就是山脊。



0	0	0	0	0	0
0	1	1	2	2	0
0	3	7	5	4	0
0	0	0	20	0	1
0	0	0	1	24	0
0	2	4	7	35	2

在程序实现的时候对结果的处理还有细化的部分。如果一个山脊是比较平的那么提出来可能就是一片，所以需要细化。采用的是查表法。

细化是从原来的图中去掉一些点，但仍要保持原来的形状。实际上是保持原图的骨架。

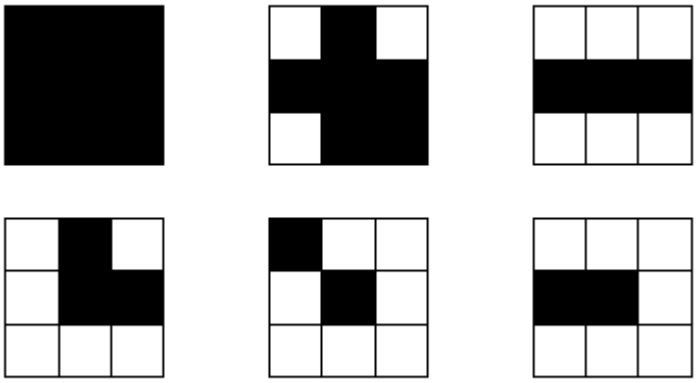
判断一个点是否能去掉是以 8 个相邻点（八连通）的情况来作为判据的，具体判据为：

1，内部点不能删除

2，鼓励点不能删除

3，直线端点不能删除

4，如果 P 是边界点，去掉 P 后，如果连通分量不增加，则 P 可删除



看看上图中间的那些点，判断它们是否可以删除

第一个点不能去除，因为它是内部点

第二个点不能去除，它也是内部点

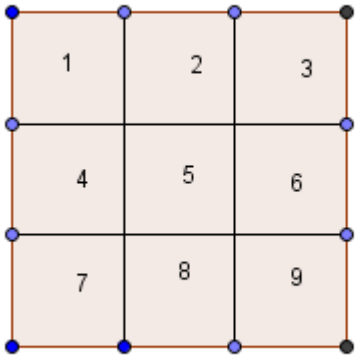
第三个点不能去除，删除后会使原来相连的部分断开

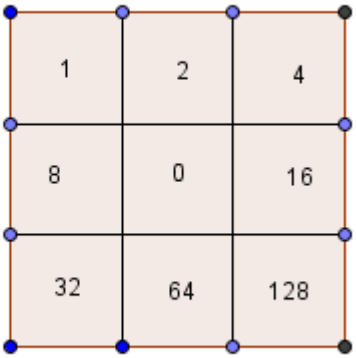
第四个点可以去除，这个点不是骨架

第五个点不可以去除，它是直线的端点

第六个点不可以去除，它是直线的端点

对于所有的这样的点，我们可以做出一张表，来判断这样的点能不能删除



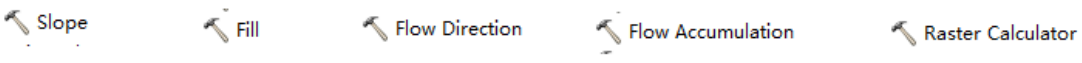


我们对于黑色的像素点，对于它周围的 8 个点，我们赋予不同的价值，若周围某黑色，我们认为其价值为 0，为白色则取九宫格中对应的价值。对于前面那幅图中第一个点，也就是内部点，它周围的点都是黑色，所以他的总价值是 0，对应于索引表的第一项。前面那幅图中第二点，它周围有三个白色点，它的总价值为 1+4+32=37，对应于索引表中第三十八项。我们用这种方法，把所有点的情况映射到 0～255 的索引表中。我们扫描原图，对于黑色的像素点，根据周围八点的情况计算它的价值，然后查看索引表中对应项来决定是否要保留这一点。表格如下，0 表示不可以删除，1 表示可以删除。

```
   array = [
0,0,1,1,0,0,1,1,1,1,0,1,1,1,0,1,\
1,1,0,0,1,1,1,1,1,0,0,0,0,0,0,1,\
0,0,1,1,0,0,1,1,1,1,0,1,1,1,0,1,\
1,1,0,0,1,1,1,1,1,0,0,0,0,0,0,1,\
1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,\
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\
1,1,0,0,1,1,0,0,1,1,0,1,1,1,0,1,\
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,\
0,0,1,1,0,0,1,1,1,1,0,1,1,1,0,1,\
1,1,0,0,1,1,1,1,1,0,0,0,0,0,0,1,\
0,0,1,1,0,0,1,1,1,1,0,1,1,1,0,1,\
1,1,0,0,1,1,1,1,1,0,0,0,0,0,0,0,\
1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,\
1,1,0,0,1,1,1,1,1,0,0,0,0,0,0,0,\
1,1,0,0,1,1,0,0,1,1,0,1,1,1,0,0,\
1,1,0,0,1,1,1,1,0,1,1,0,0,1,0,0,0]
```

2. 流程

Arcgis 中操作的流程

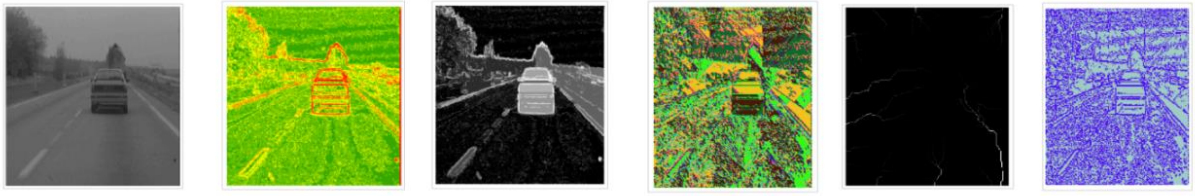


在对应的程序中就是计算坡度->填充->计算流向->就算汇流量为 0 的栅格->细化。

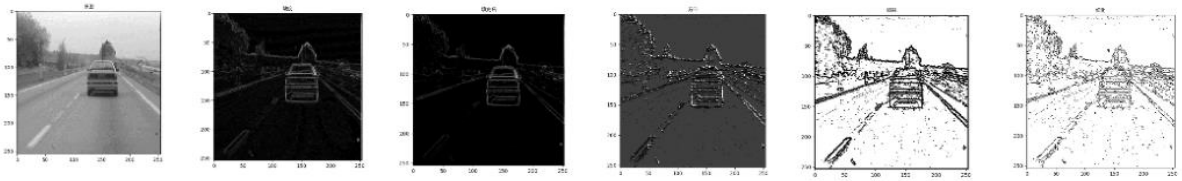
计算坡度以及流向流量的计算都有 arcgis 官方的文档可以参考，但是填充却没有给出具体的算法。我采用的是小于阈值的数值全部填充为阈值。阈值取均值的 1.5 倍。

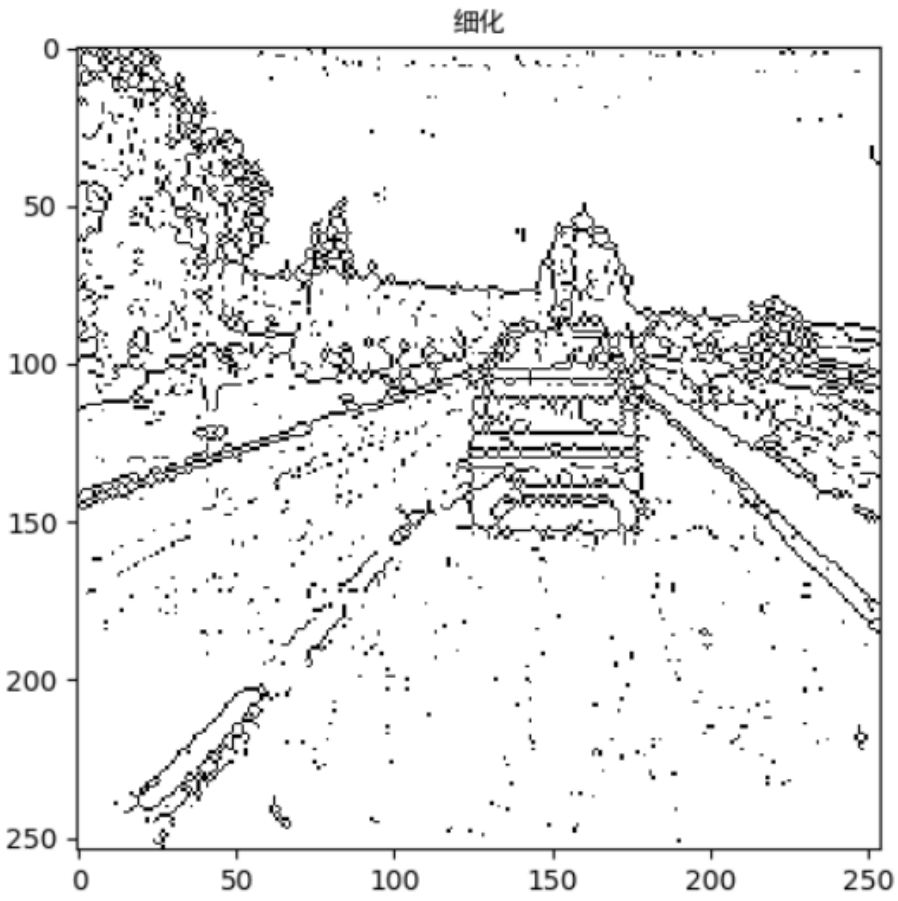
3. 测试结果

Arcgis 中的结果，从左到右依此为原图、坡度、填充后的坡度、流向、汇流量、山脊



Python 计算的结果，从左到右依此为原图、坡度、填充后的坡度、流向、山脊、细化

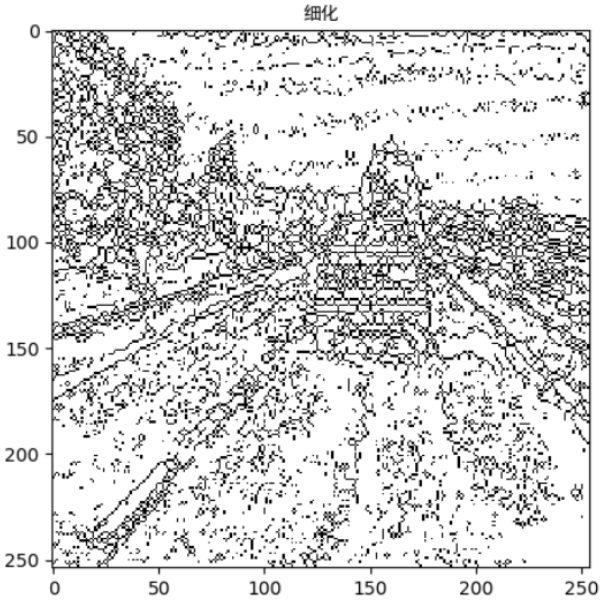




4. 分析讨论

结果分出来的区域比较琐碎，与填充时取的阈值有很大关系。而且无法保证连通，这是最大的缺陷。

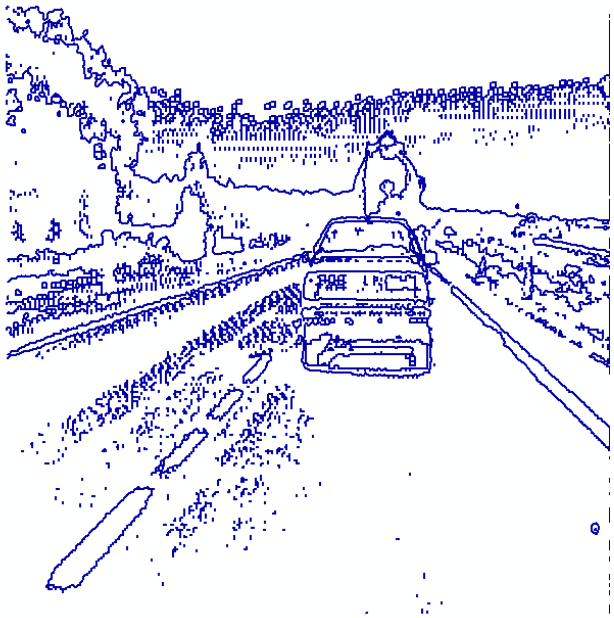
如果阈值取平均值结果就显得更加琐碎。



5. 其它 GIS 算法

因为上述算法无法保证连通性，联想到等高线是可以保证连通性的。如果将原始图像当作一幅 DEM 那么等高线就是一种对图像分割的方式。

Arcgis 算出来的等高线



Python 直接绘制的等高线，并且上色。

用等高线来分割图像效果不错。图像分割没有标准答案，当把一幅图像当作 DEM 来看的时候，GIS 中一些处理地形的算法也可以用来分割图像。