Chap 4 空间数据转换

4.4 矢量结构与栅格结构的相互转换

1. 矢量数据结构向栅格数据结构的转换（栅格化）
2. 栅格数据结构向矢量数据结构的转换（矢量化）

4.4.1矢量到栅格

栅格化过程包括以下操作：

1）确定栅格矩阵(行列数\分辨率)；

2）点的变换

3）线的变换

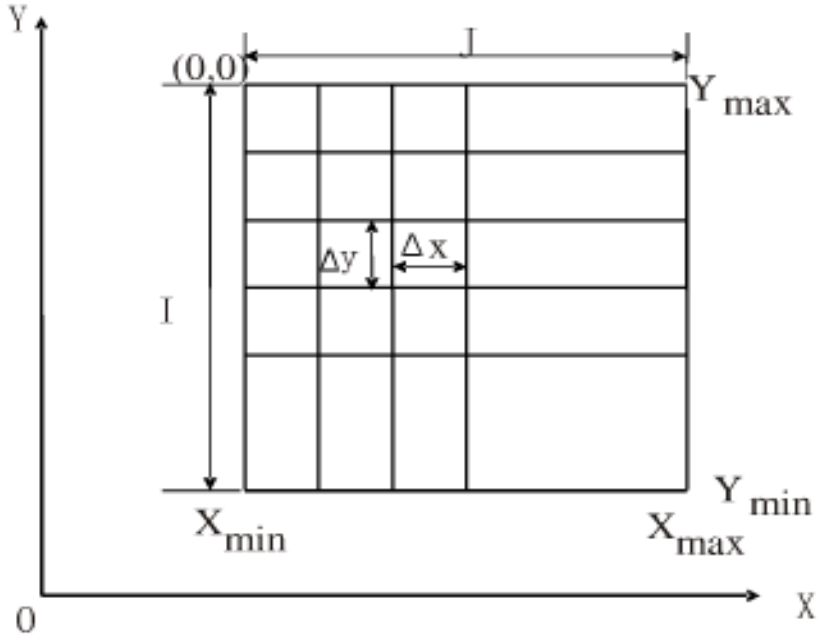
4）多边形的变换(面的变换)

1) 确定栅格矩阵

- 矢量数据转换成栅格数据后，图形的几何精度必然要降低，所以选择栅格尺寸的大小要尽量满足精度要求，使之不过多地损失地理信息。
- 为了提高精度，栅格需要细化，但栅格细化，数据量将以平方指数递增，因此，精度和数据量是确定栅格大小的最重要的影响因素。

1) 确定栅格矩阵

- 在转换之前需要确定栅格单元的大小，栅格单元的大小又称为栅格图像的分辨率，直接决定了栅格数据的精度。



I=(Ymax-Ymin)/dy

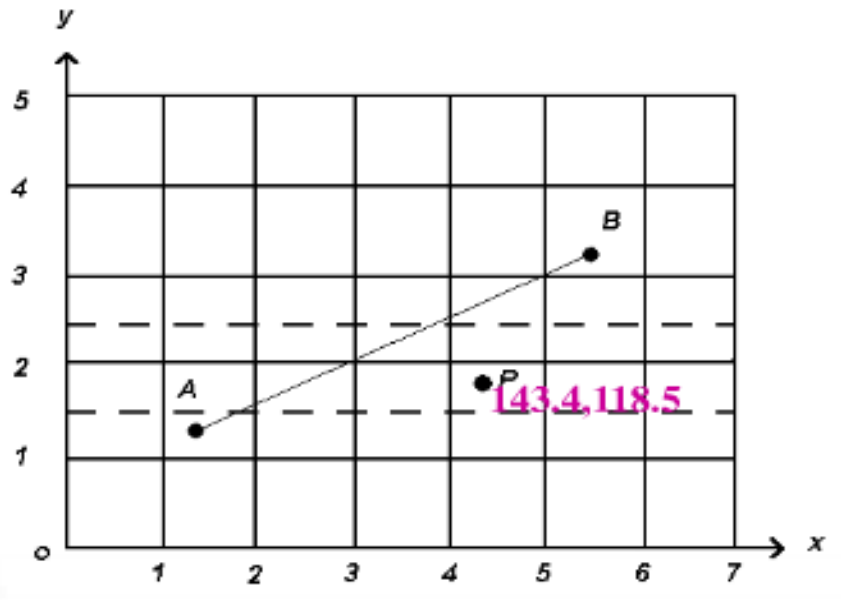
J=(Xmax-Xmin)/dx

I，J为整数，尾数入上去

dx=(Xmax-Xmin)/J
dy=(Ymax-Ymin)/I

2）点的栅格化

$$\begin{cases} I_p = 1 + \mathrm{INT}[(y_{\max} - y)/dy] \\ J_p = 1 + INT[(x - x_{\min})/dx] \end{cases}$$



P(143.4,118.5)

西南(100,100)

东北(170,150)

dx，dy=10

$$\begin{cases} I_p = 1 + \mathrm{INT}[(y_{\max} - y)/dy] \\ = 1 + INT[(150 - 118.5)/10] \\ = 1 + 3 = 4 \\ J_p = 1 + INT[(x - x_{\min})/dx] \\ = 1 + INT[(143.4 - 100)/10] \\ = 1 + 4 = 5 \end{cases}$$

3）线的栅格化方法

- 线是由多个直线段组成的，因此线的栅格化的核心就是直线段如何由矢量数据转换为栅格数据。
- 栅格化的两种常用方法为:
  - DDA法(Digital Differential Analyzer数字微分分析法)
  - Bresenham法

DDA法(数字微分分析法)

设线段AB与栅格的交点坐标为(x_i,y_i)，则:

$$\begin{cases} x_{i+1} = x_i + \dfrac{x_B - x_A}{n} = x_i + \Delta x \\ y_{i+1} = y_i + \dfrac{y_B - y_A}{n} = y_i + \Delta y \end{cases}$$

其中：

$$n = \mathrm{int}\left[\max\left(|x_B - x_A|, |y_B - y_A|\right)/d\right]$$

$$\Delta x = \frac{x_B - x_A}{n}, \Delta y = \frac{y_B - y_A}{n}$$

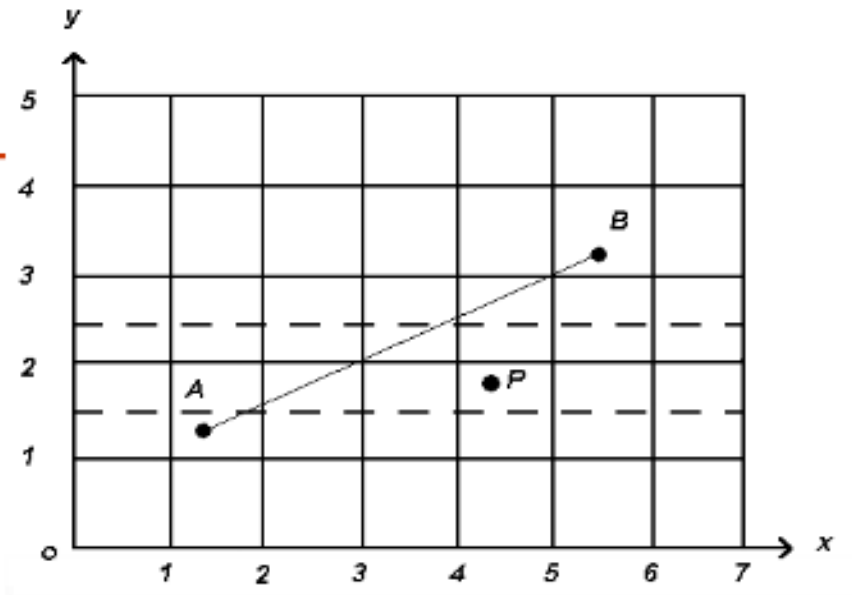
$$x_0 = x_A, y_0 = y_A, x_n = x_B, y_n = y_B$$

这样从i＝0计算到i＝n－1，即可得直线与格网的n个交点坐标，对其取整就是该点的栅格数据了。

- 该方法的基本依据是直线的微分方程，即dy/dx＝常数。其本质是用数值方法解微分方程，通过同时对x和y各增加一个小增量来计算下一步的x,y值，即这是一种增量算法。
- 在该算法中，必须以浮点数表示坐标，且每次都要舍入取整，因此，尽管算法正确，但速度不够快。

A(113,112)，B(155,132)

西南 (100,100)东北(170,150)



$$n = \text{int}\left[\max\left(|x_B - x_A|, |y_B - y_A|\right)/d\right]$$
$$= \text{int}\left[\max\left(|155-113|, |132-112|\right)/10\right]$$
$$= \text{int}\left[\max(42,20)/10\right] = 4$$

$$\Delta x = \frac{x_B - x_A}{n} = 10.5, \Delta y = \frac{y_B - y_A}{n} = 5$$

$$x_0 = 113, y_0 = 112, x_n = 155, y_n = 132$$

$$\begin{cases} x_1 = x_0 + \Delta x = 113 + 10.5 = 123.5 \\ y_1 = y_0 + \Delta y = 112 + 5 = 117 \end{cases}$$

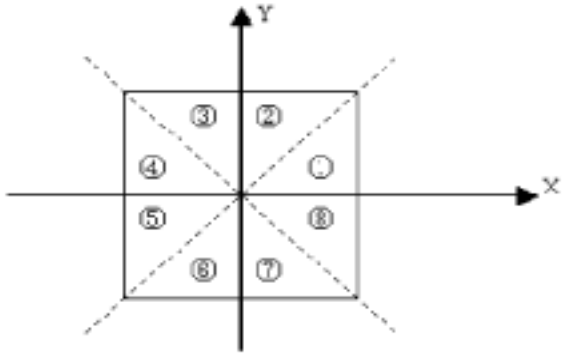
$$\begin{cases} x_2 = x_1 + \Delta x = 123.5 + 10.5 = 134 \\ y_2 = y_1 + \Delta y = 117 + 5 = 122 \end{cases}$$

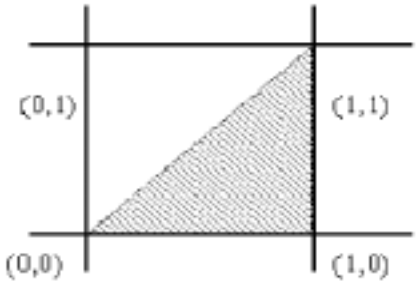
$$\begin{cases} x_3 = x_2 + \Delta x = 134 + 10.5 = 144.5 \\ y_3 = y_2 + \Delta y = 122 + 5 = 127 \end{cases}$$

$$\begin{cases} x_4 = x_3 + \Delta x = 144.5 + 10.5 = 155 \\ y_4 = y_3 + \Delta y = 127 + 5 = 132 \end{cases}$$

Bresenham算法

- 该算法原来是为绘图机设计的，但同样适合于栅格化。该算法构思巧妙，只需根据由直线斜率构成的误差项的符号，就可确定下一列坐标的递增值。
- 根据直线的斜率，把直线分为8个卦限。下面举斜率在第一卦限的情况为例，其余卦限的情况类似。





- 该算法的基本思路可描述为：若直线的斜率为$1/2 \le \Delta y/\Delta x \le 1$，则下一点取(1，1)点，若$0 \le \Delta y/\Delta x < 1/2$，则下一点取(1，0)点。

9

在算法实现时，令起始的误差项为e＝-1/2，
然后在推断出下一点后，令e＝e＋△y/△x,若e≥0时，e＝e-1
这样只要根据e的符号就可确定下一点的增量，即：

若e≥0，取(1，1)点

若e＜0，取(1，0)点

例如，一直线的斜率为1/3，起始点：e0＝-1/2,取点0

第1点：e1＝-1/2 +1/3＝-1/6取点①

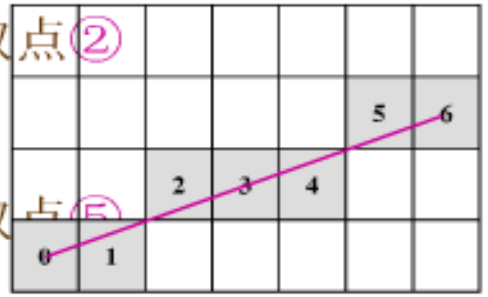
第2点：e2＝-1/6 +1/3 = 1/6且e2’=-5/6；取点②

第3点：e3＝-5/6+1/3=-1/2取点③

第4点：e4＝-1/2+1/3=-1/6取点④

第5点：e5＝-1/6 +1/3 = 1/6且e2’=-5/6；取点⑤

第6点：e6＝-5/6+1/3=-1/2取点⑥

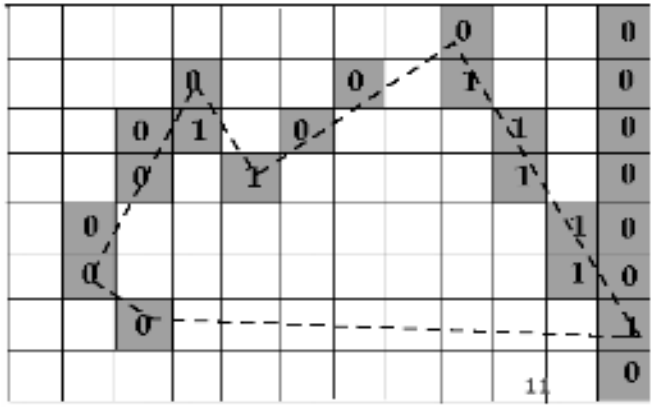


依次进行，直到到达直线的另一端点。

这种算法不仅速度快、效果好，而且可以理论上证明它是目前同类各种算法中最优的。

4）面域的栅格化

- 边界线的转化与线的栅格化方法相同，接下来就是属性的填充。
- 填充的方法很多，关键问题是正确判断哪些栅格单元位于多边形之内，哪些位于多边形之外。为此，多边性必须严格封闭，没有缝隙。
- 属性的填充方法有：
  - 内部点扩散法
  - 射线算法
  - 平行线扫描法与铅垂线跌落法
  - 边界代数充填算法
  - 边界点跟踪算法

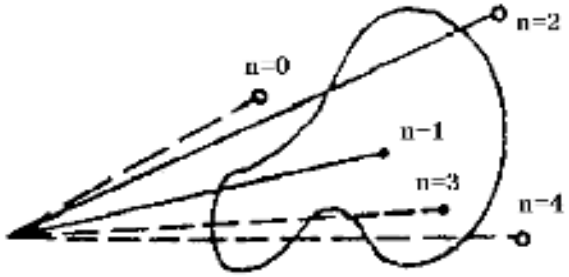


（1）内部点扩散算法

- 该算法由每个多边形一个内部点（种子点）开始，向其八个方向的邻点扩散，判断各个新加入点是否在多边形边界上，如果是边界上，则该新加入点不作为种子点，否则把非边界点的邻点作为新的种子点与原有种子点一起进行新的扩散运算，并将该种子点赋以该多边形的编号。重复上述过程直到所有种子点填满该多边形并遇到边界停止为止。
- 扩散算法程序设计比较复杂，并且在一定的栅格精度上，如果复杂图形的同一多边形的两条边界落在同一个或相邻的两个栅格内，会造成多边形不连通，这样一个种子点不能完成整个多边形的填充。

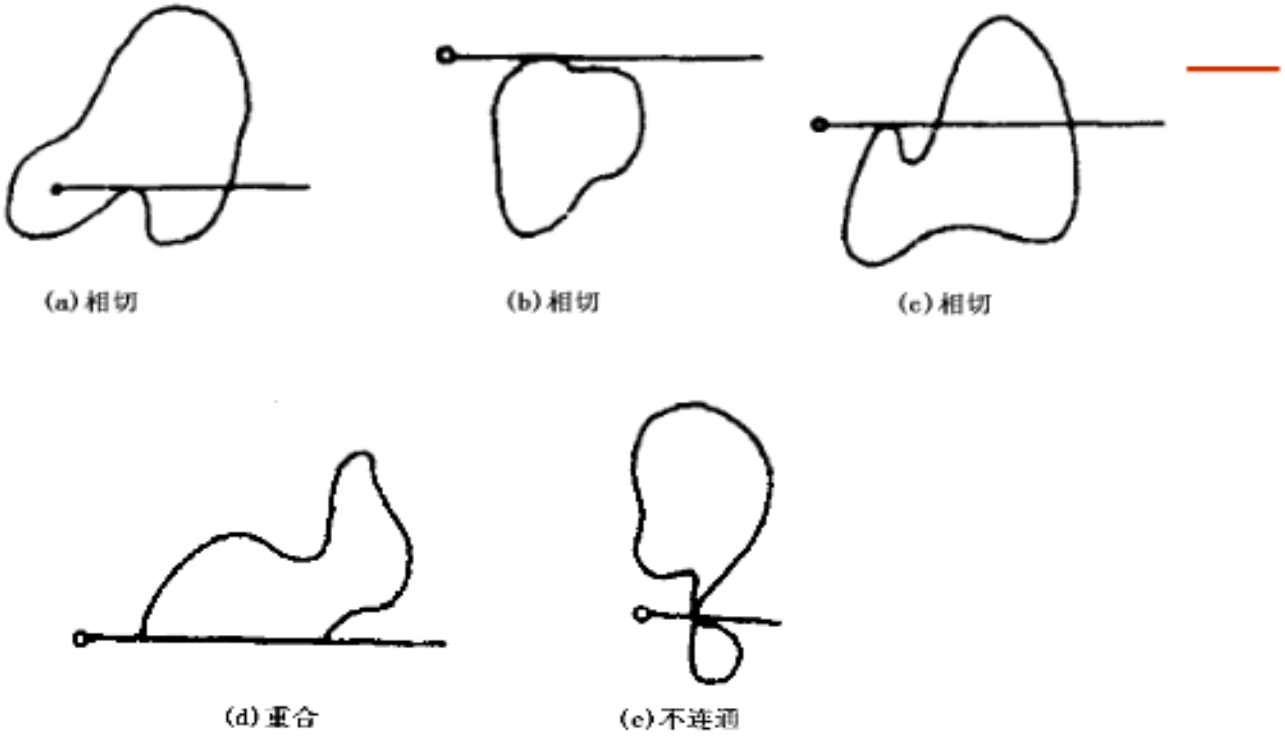
（2）射线算法

- 射线算法可逐点判断数据栅格点在某多边形之外或在多边形内，由待判点向图外某点引射线，判断该射线与某多边形所有边界相交的总次数，如相交偶数次，则待判点在该多边形外部，如为奇数次，则待判点在该多边形内部。
- 采用射线算法，要注意的是：射线与多边形边界相交时，有一些特殊情况会影响交点的个数，必须予以排除。



○ 外部点　　· 内部点　　*n* 交点个数

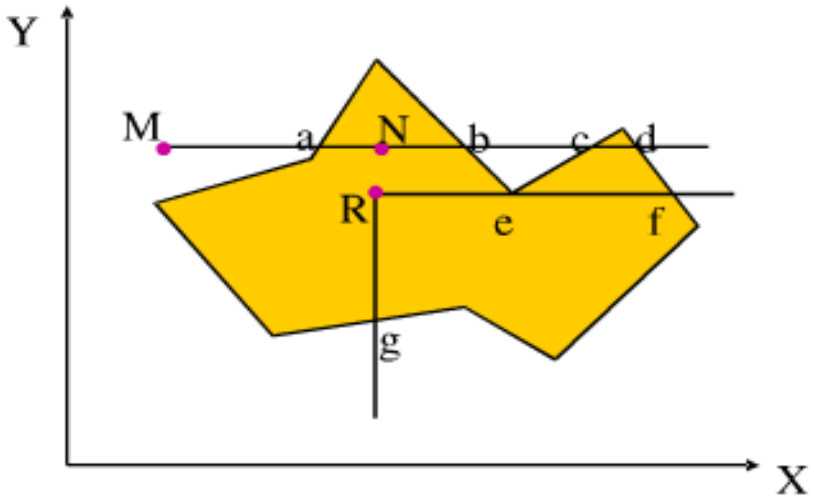
射线算法



射线算法的特殊情况

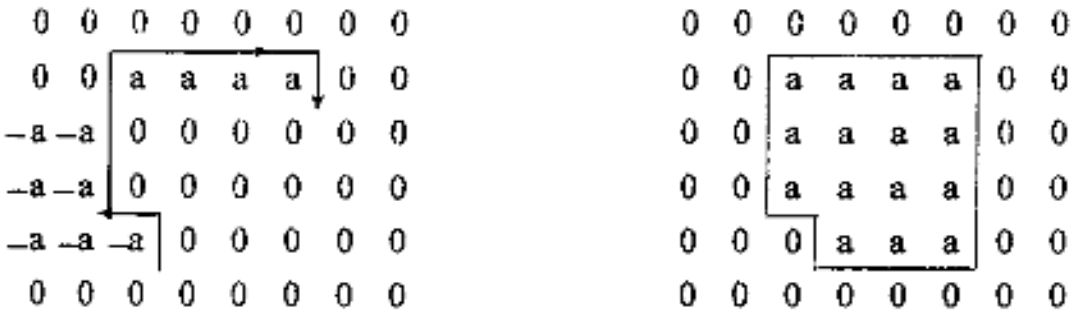
（3）平行线扫描法与铅垂线跌落法

- 射线算法的改进，将射线改为沿栅格阵列列方向或行方向扫描线，判断与射线算法相似。省去了计算射线与多边形边界交点的大量运算，大大提高了效率。



（4）边界代数算法（BAF-Boundary Algebra Filling）

- 边界代数多边形填充算法是一种基于积分思想的矢量格式向栅格格式转换算法，它适合于记录拓扑关系的多边形矢量数据转换为栅格结构。
- 转换单个多边形：设多边形编号为a，初始化的栅格阵列各栅格值为零，以栅格行列为参考坐标轴，由多边形边界上某点开始顺时针搜索边界线，当边界上行时，位于该边界左侧的具有相同行坐标的所有栅格被减去a；当边界下行时，该边界左侧（前进方向看为右侧）所有栅格点加一个值a，边界搜索完毕则完成了多边形的转换。



单个多边形的转换

16

转换多个多边形

- 当边界弧段上行时，该弧段与左图框之间栅格增加一个值（左多边形编号减去右多边形编号）；当边界弧段下行时，该弧段与左图框之间栅格增加一个值（右多边形编号减去左多边形编号）。

0	0	0	0	0	0	0	0	0	0
0	0	0	0	5	5	5	0	0	0
0	0	0	2	5	5	5	5	0	0
0	0	2	2	5	5	5	5	2	0
0	0	2	2	5	5	5	2	2	0
0	2	2	2	2	2	2	2	2	0
0	2	2	2	2	0	0	0	0	0
0	2	2	2	2	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	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-3	-3	-3	-3	0	0	0	0	0	0
-3	-3	-3	-3	0	0	0	0	0	0
-3	-3	-3	-3	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
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	0	0	0	0

0	0	0	0	0	0	0	0	0	0
0	0	0	0	5	5	5	0	0	0
0	0	0	2	5	5	5	5	0	0
0	0	2	2	5	5	5	5	2	0
0	0	2	2	5	5	5	2	2	0
0	2	2	2	2	2	2	2	2	0
0	2	2	2	2	0	0	0	0	0
0	2	2	2	2	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	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-3	-3	-3	-3	0	0	0	0	0	0
-3	-3	-3	-3	0	0	0	0	0	0
-3	-3	-3	-3	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
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	0	0	0	0

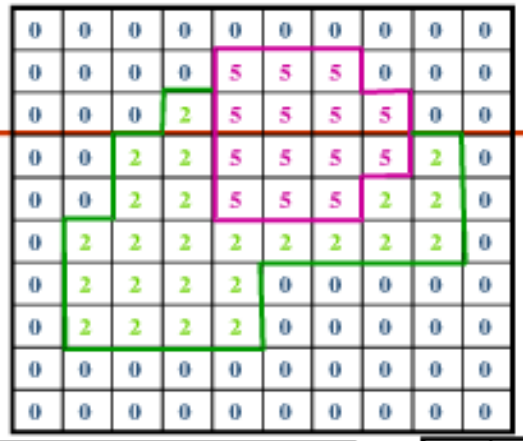
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-3	-3	-3	-3	0	0	0	0	0	0
0	0	0	0	3	3	3	3	0	0
0	0	0	0	3	3	3	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0
0	0	0	0	5	5	5	0	0	0
0	0	0	2	5	5	5	5	0	0
0	0	2	2	5	5	5	5	2	0
0	0	2	2	5	5	5	2	2	0
0	2	2	2	2	2	2	2	2	0
0	2	2	2	2	0	0	0	0	0
0	2	2	2	2	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

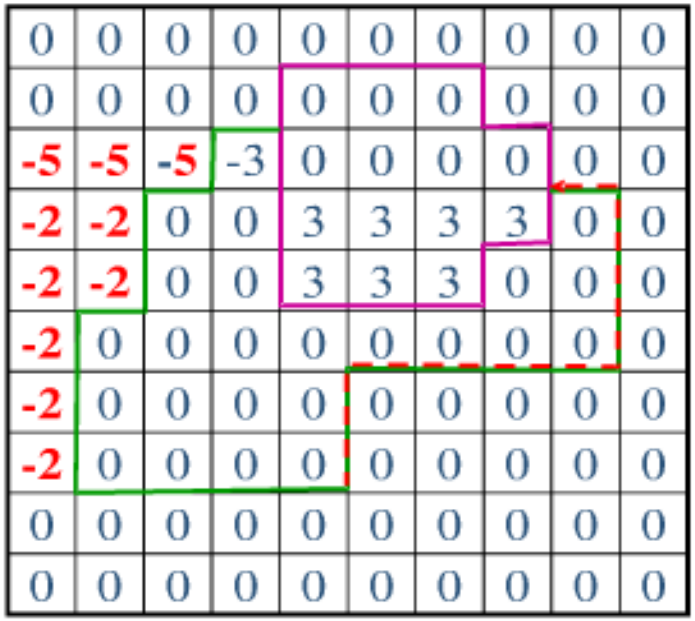
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-3	-3	-3	-3	0	0	0	0	0	0
0	0	0	0	3	3	3	3	0	0
0	0	0	0	3	3	3	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-5	-5	-5	-3	0	0	0	0	0	0
-2	-2	0	0	3	3	3	3	0	0
-2	-2	0	0	3	3	3	0	0	0
-2	0	0	0	0	0	0	0	0	0
-2	0	0	0	0	0	0	0	0	0
-2	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	0	0	0

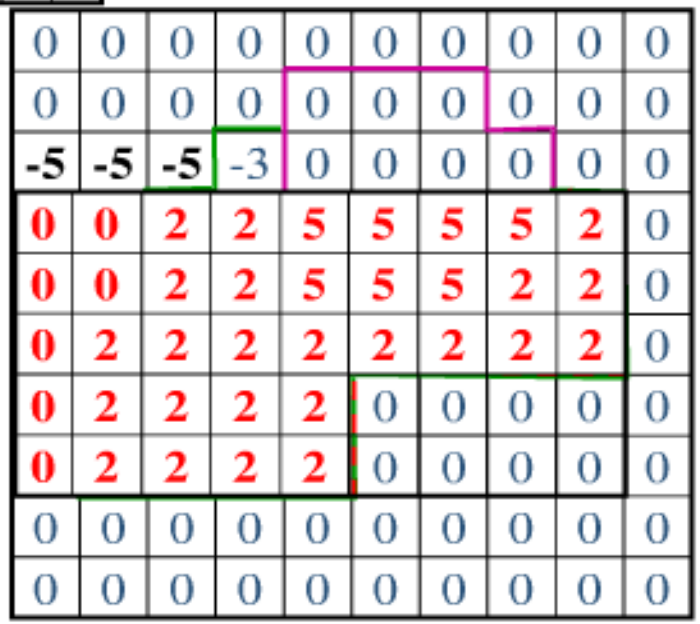
20



0	0	0	0	0	0	0	0	0	0
0	0	0	0	5	5	5	0	0	0
0	0	0	2	5	5	5	5	0	0
0	0	2	2	5	5	5	5	2	0
0	0	2	2	5	5	5	2	2	0
0	2	2	2	2	2	2	2	2	0
0	2	2	2	2	0	0	0	0	0
0	2	2	2	2	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

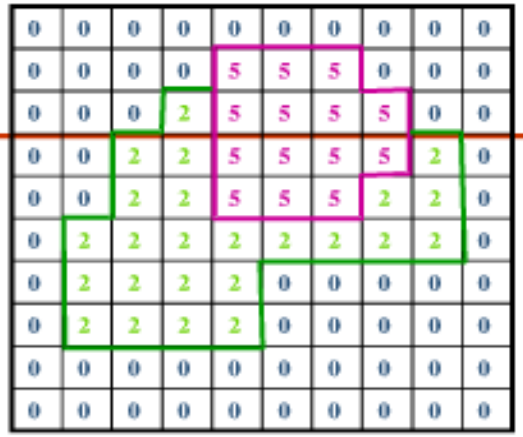


0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-5	-5	-5	-3	0	0	0	0	0	0
-2	-2	0	0	3	3	3	3	0	0
-2	-2	0	0	3	3	3	0	0	0
-2	0	0	0	0	0	0	0	0	0
-2	0	0	0	0	0	0	0	0	0
-2	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-5	-5	-5	-3	0	0	0	0	0	0
0	0	2	2	5	5	5	5	2	0
0	0	2	2	5	5	5	2	2	0
0	2	2	2	2	2	2	2	2	0
0	2	2	2	2	0	0	0	0	0
0	2	2	2	2	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

21



0	0	0	0	0	0	0	0	0	0
0	0	0	0	5	5	5	0	0	0
0	0	0	2	5	5	5	5	0	0
0	0	2	2	5	5	5	5	2	0
0	0	2	2	5	5	5	2	2	0
0	2	2	2	2	2	2	2	2	0
0	2	2	2	2	0	0	0	0	0
0	2	2	2	2	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	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
-5	-5	-5	-3	0	0	0	0	0	0
0	0	2	2	5	5	5	5	2	0
0	0	2	2	5	5	5	2	2	0
0	2	2	2	2	2	2	2	2	0
0	2	2	2	2	0	0	0	0	0
0	2	2	2	2	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

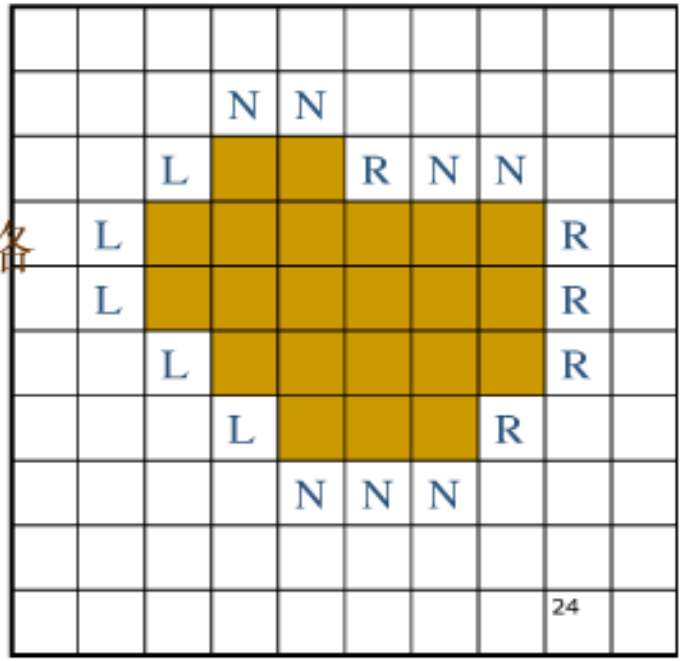
0	0	0	0	0	0	0	0	0	0
0	0	0	0	5	5	5	0	0	0
0	0	0	2	5	5	5	5	0	0
0	0	2	2	5	5	5	5	2	0
0	0	2	2	5	5	5	2	2	0
0	2	2	2	2	2	2	2	2	0
0	2	2	2	2	0	0	0	0	0
0	2	2	2	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

- 边界代数法不是逐点判断与边界的关系完成转换，而是根据边界的拓扑信息，通过简单的加减代数运算将边界位置信息动态地赋给各栅格点，实现了矢量格式到栅格格式的高速转换，而不需要考虑边界与搜索轨迹之间的关系，因此算法简单、可靠性好，各边界弧段只被搜索一次，避免了重复计算。
- 但是这并不意味着边界代数法可以完全替代其它算法，在某些场合下，还是要采用种子填充算法和射线算法，前者应用于在栅格图像上提取特定的区域；后者则可以进行点和多边形关系的判断

（5）边界点跟踪算法

- 以多边形为单位
- 按顺时针方向跟踪单元格
  - 上行L
  - 横向N
  - 下行R

  （岛则相反）
- 逐行扫描，充填LR间的单元格



4.4.2 栅格到矢量

- 从栅格单元转换到几何图形的过程称为矢量化，矢量化过程要保证以下两点：
  1）拓扑转换，即保持栅格表示出的连通性与邻接性；
  2）转换物体正确的外形。

4.4.2栅格数据结构向矢量数据结构的转换

- 主要步骤：
  1. 边界提取
     a) 边缘锐化（遥感影像或分类栅格，面状地物）
     b) 线的细化（扫描矢量图，线状地物）
     c) 二值化
  2. 边界追踪
  3. 线的简化及曲线圆滑
  4. 拓扑关系生成

1. 边界提取

1) 边缘检测（图像处理）

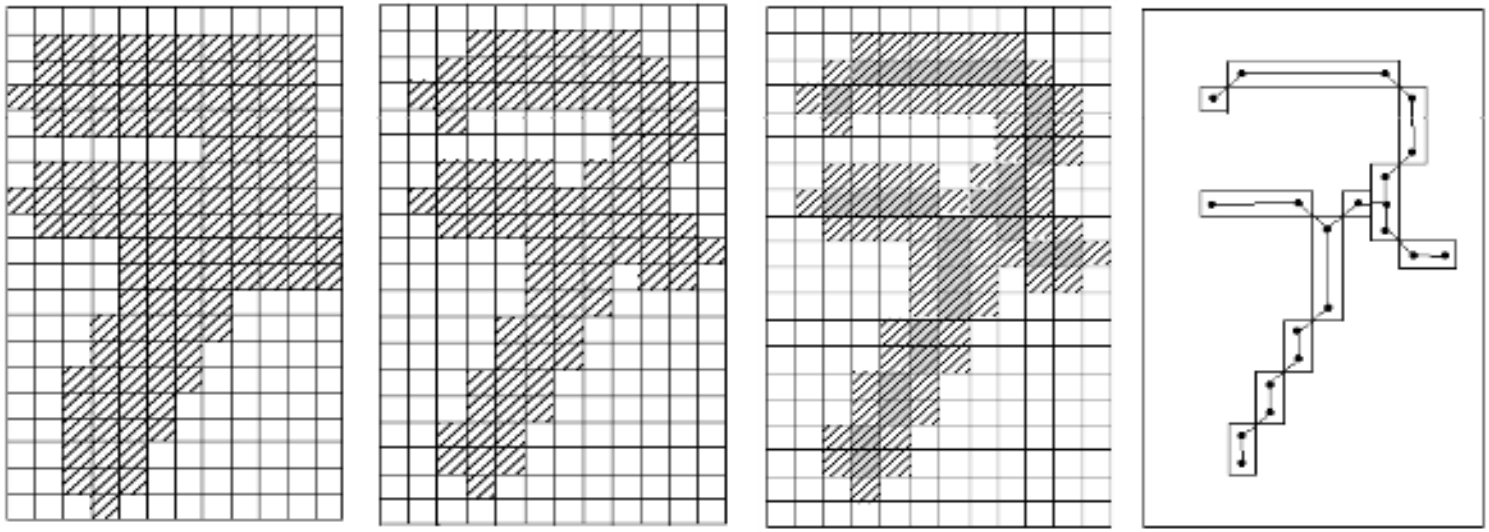
对于影像栅格数据，检测出不同特征区域的边界。

Lenna的Prewitt边界

Lenna的Sobel边界

2) 线的细化：将占有多个栅格宽的图形要素缩减为只有1个像素。

1. 剥皮法：每次剥掉等于一个栅格宽的一层，最后只留下彼此连通的由单个栅格组成的图形。



2.骨架法：确定图形的骨架，而将非骨架上的多余栅格删除。

具体做法是扫描全图，凡是像元值为1的栅格都用V值取代。V值是该栅格与北、东和北东三个相邻栅格像元值之和，即

$$v = f(i,j) + f(i-1,j) + f(i,j+1) + f(i-1,j+1)$$

在V值图上保留最大V值的栅格，删去其他栅格，但必须保证连通。因为最大V值的栅格只能分布在图形的中心线上（骨架上），因此选取最大值栅格的过程就是细化的过程 。

3.数学形态法

31

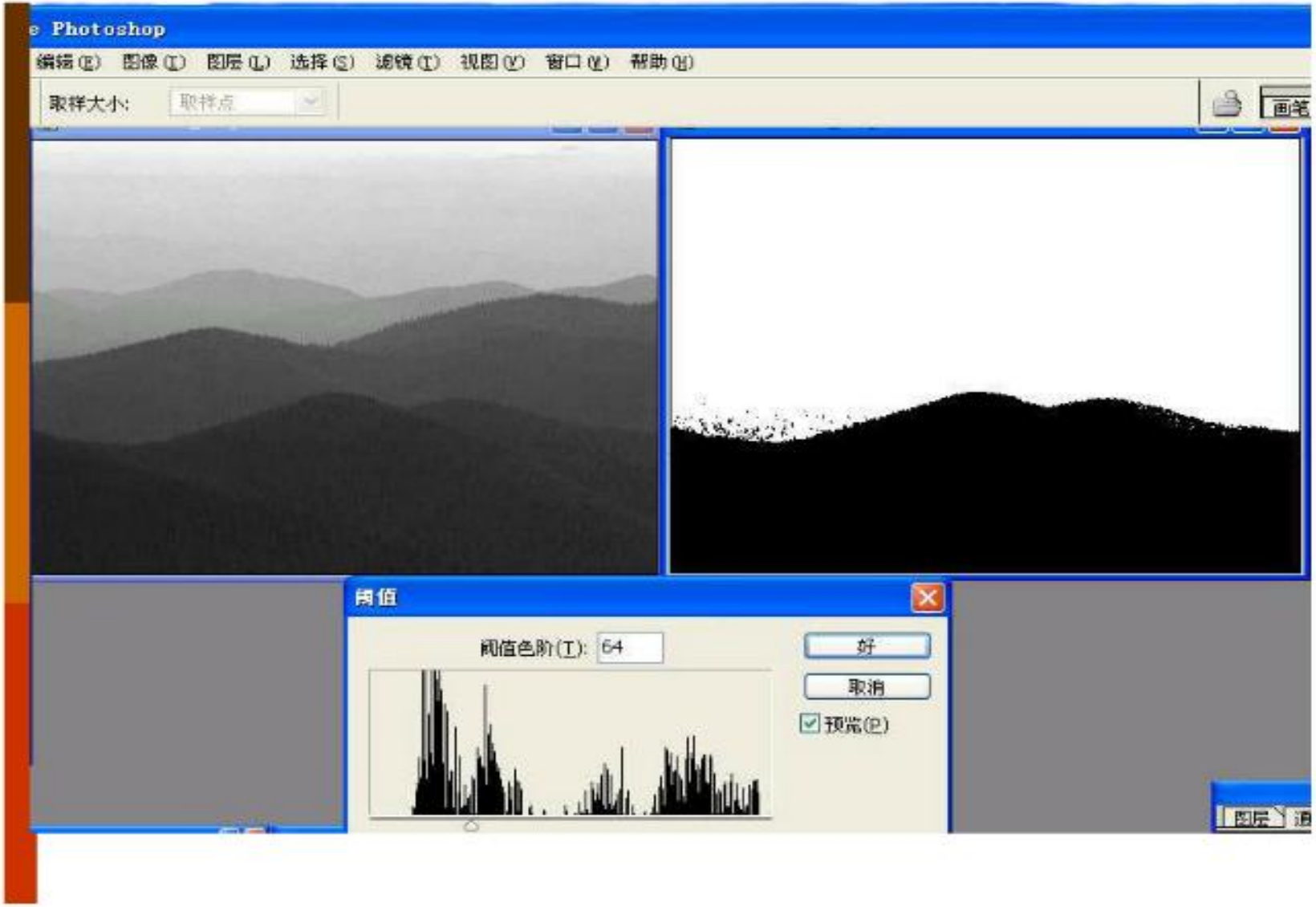
1. 边界提取

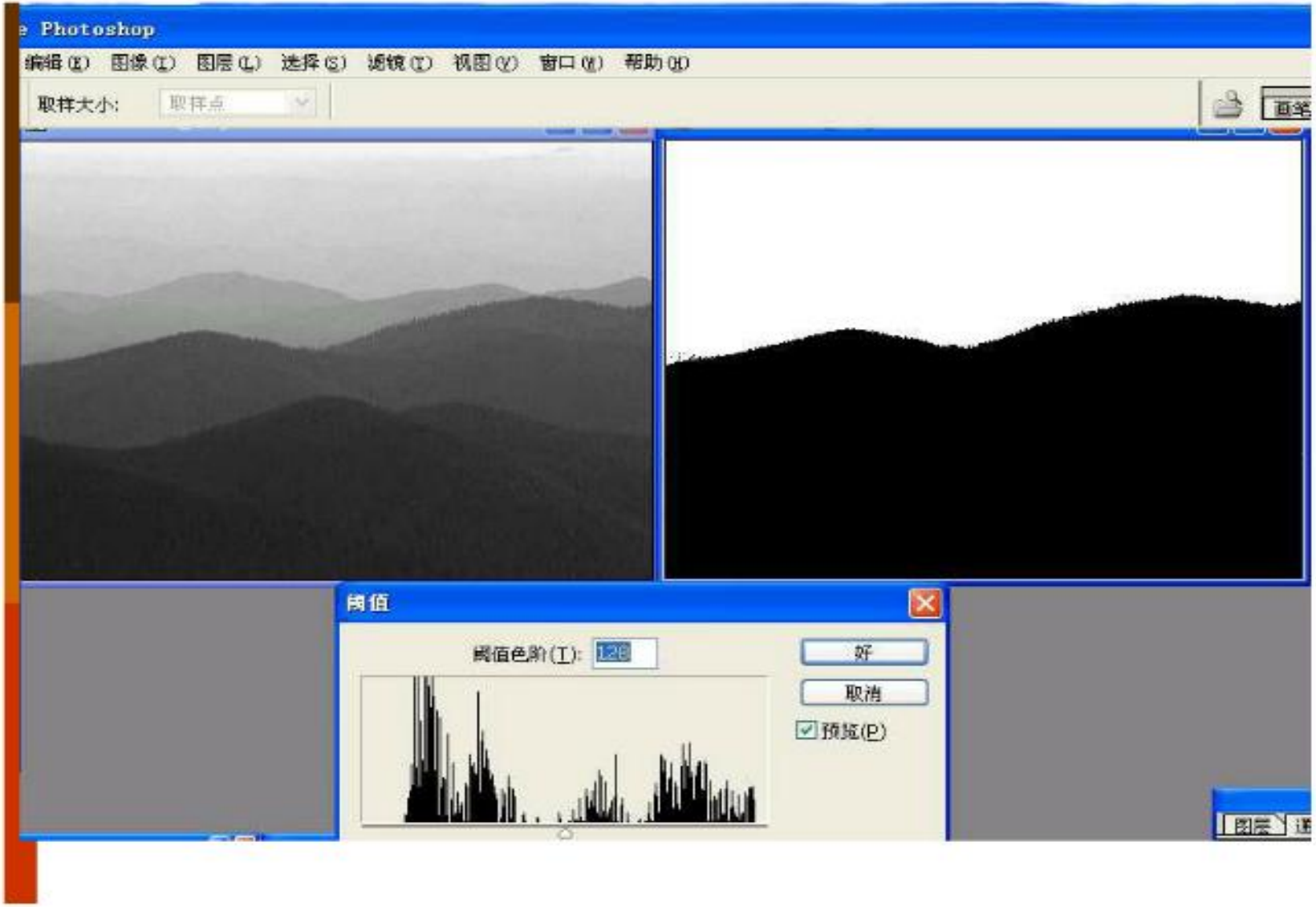
3) 二值化

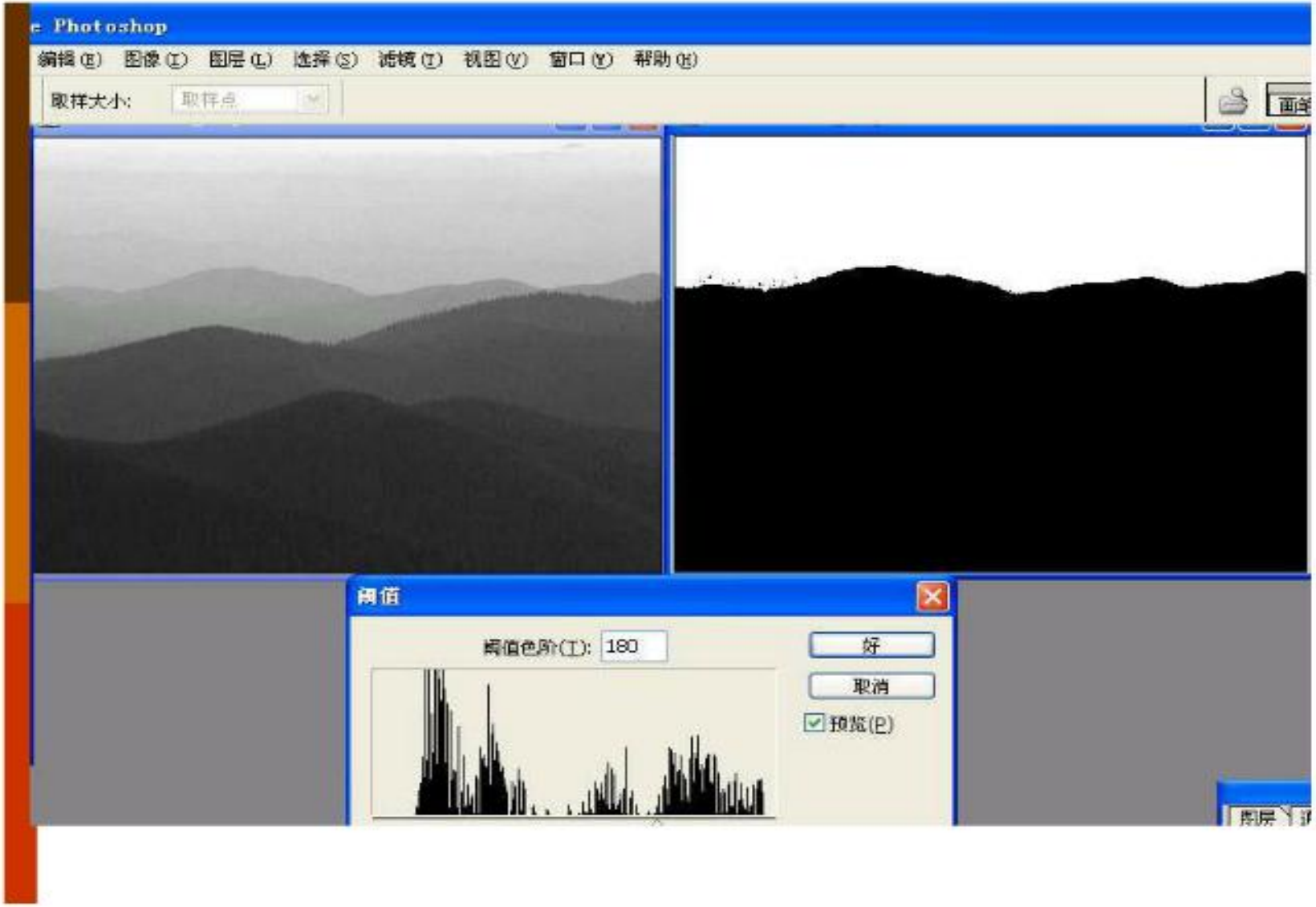
一般情况下，栅格数据是按0～255的不同灰度值表达的。

为了简化追踪算法，需把256个灰阶压缩为2个灰阶，即0和1两级。为此，假设任一格网的灰度值为G(i,j)，阀值为T，那么，根据下式就可以得到二值图。

$$G(i,j)=\begin{cases} 1 & f(i,j)>=T \\ 0 & f(i,j)<=T \end{cases}$$


e Photoshop
编辑(E) 图像(I) 图层(L) 选择(S) 滤镜(T) 视图(V) 窗口(W) 帮助(H)
取样大小:
取样点
画笔
阈值
阈值色阶(T): 64
好
取消
预览(P)
图层


Photoshop
编辑(E) 图像(I) 图层(L) 选择(S) 滤镜(T) 视图(V) 窗口(W) 帮助(H)
取样大小:
取样点
画笔
阈值
阈值色阶(T):
128
好
取消
预览(P)
图层


e Photoshop
编辑(E) 图像(I) 图层(L) 选择(S) 滤镜(T) 视图(V) 窗口(W) 帮助(H)
取样大小:
取样点
阈值
阈值色阶(T): 180
好
取消
预览(P)

2. 边界线追踪：边界线跟踪的目的就是将细化处理后的栅格数据，整理为从结点出发的线段或闭合的线条，并以矢量形式存储（坐标）。

$$X = X_{\min} + (\Delta X.j - \frac{\Delta X}{2})$$

$$Y = Y_{\max} - (\Delta Y.i - \frac{\Delta Y}{2})$$

3. 线的简化及曲线圆滑：由于搜索是逐个栅格进行的，所以弧段或多边形的数据列十分密集。为了减少存储量，在保证线段精度的情况下可以删除部分数据点。

4. 拓扑关系生成：判断弧段与多边形间的空间关系，以形成完整的拓扑结构并建立与属性数据的关系。

4.4.3 矢量格式和栅格格式的相互转换

- 矢量格式向栅格格式的转换
  1. 建立空白栅格
  2. 点、线的栅格化
  3. 多边形填充
     - ①内部点扩散法
     - ②复数积分算法
     - ③射线算法
     - ④扫描算法
     - ⑤边界代数算法
- 栅格格式向矢量格式的转换
  1. 边界提取
  2. 边界追踪
  3. 去除多余点及曲线圆滑
  4. 拓扑关系生成

专业术语

- 实体数据结构 spaghetti data structure
- 双重独立地图编码 DIME
- 游程长度编码 Run-Length Encoding
- 四叉树数据结构 Quad tree data structure
- 链码结构 Chain Encoding

思考题

一、基础部分

1、总结矢量数据和栅格数据在结构表达方面的特色。

2、简述栅格数据压缩编码的几种方式和各自优缺点。

3、简述矢量数据编码的几种方式和各自优缺点。

4、栅格与矢量数据结构相比较各有什么特征？