



电子科技大学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

学士学位论文

BACHELOR DISSERTATION

论文题目: 基于 DEM 规则格网绘制等高线

学生姓名: 秦自松

学 号: 2607102009

专 业: 测控技术与仪器

学 院: 自动化工程学院

指导教师: 蒋 玲

指导单位: 自动化工程学院

2010 年 06 月 08 日

摘 要

随着地理信息系统（GIS）的不断发展，DEM（Digital Elevation Model, 即数字高程模型）在 GIS 中得到了越来越广泛的应用，成为 GIS 中的一个重要组成部分。

目前，DEM 数据的获得不再仅仅局限于原先保存的纸质地形图的数字化，也许是利用激光测距仪配合地面控制点获得，由这种方法获得的 DEM 格网数据可能需要生成等高线图，以此作为基础地理数据或底图。

本论文研究 DEM 数据模型、数据结构以及由 DEM 数据生产等高线的方法，主要从计算机地图制图的角度，研究基于 DEM 规则格网的等高线自动绘制问题。同时，利用 VC++ 设计编写程序，通过实验证明其正确性和有效性。

关键词：DEM； 等高线； 规则格网； 数字高程模型。

ABSTRACT

Along with the advance of Geographic Information System (GIS), Digital Elevation Model (DEM) are getting more and more widely applied in GIS, and become an important component.

At present, the obtained DEM data no longer confined to preserve the original paper, digital map may be using laser rangefinder coupled with ground control and the grid DEM data may need to generate contour map, as a basic geographic data or for reproduction.

This paper studies the DEM data model, data structure and the method of producing contour DEM data, mainly from the Angle of computer mapping, research the contour of the automatic drawing based on grid DEM data. At the same time, using VC++ design program, through the experiment proves its correctness and effectiveness.

Keywords: DEM, Contours, Grid DEM, Digital elevation model (DEM).

目 录

第 1 章 引言.....	1
1.1 本论文的背景.....	1
1.2 本论文的价值及其意义.....	1
1.3 本论文的国内外研究现状.....	2
第 2 章 相关知识概述.....	3
2.1 DEM 的定义.....	3
2.2 CNSDTF-DEM 数据格式简介.....	4
2.3 等高线的定义及其特点.....	6
2.4 等高线跟踪相关算法.....	6
第 3 章 基于 DEM 规则格网绘制等高线方法研究.....	10
3.1 格网数据的高斯滤波处理.....	10
3.2 等高线的扫描与绘制.....	11
3.2.1 基于 DEM 提取等高线概述.....	11
3.2.2 内插高程点方法.....	12
3.2.3 格网的遍历方法.....	12
3.2.4 等高线的扫描.....	14
3.2.5 等高线的绘制.....	15
3.3 总结.....	17
第 4 章 软件系统的设计.....	18
4.1 程序语言和开发平台简介.....	18
4.1.1 C++的发展历史.....	18
4.1.2 C++的类.....	19
4.1.3 Visual C++简介.....	20
4.2 软件系统流程图.....	22
4.3 数据结构设计.....	23
4.4 DEM 文件的数据读取.....	23

第 5 章 软件的编写与调试.....	25
5.1 软件的编写.....	25
5.2 软件的调试运行.....	28
第 6 章 结束语.....	31
参考文献.....	32
致 谢.....	33
附录 1 CONTOURDOC 类头文件.....	34
附录 2 CCONTOURLINE 类头文件.....	35
附录 3 读取 CNSDTF-DEM 文件头信息.....	36
附录 4 读取 DEM 文件中的高程数据程序示例.....	37
附录 5 函数 ONFILEOPEN() 函数体.....	38
外文资料原文.....	40
外文资料译文.....	47

第 1 章 引言

1.1 本论文的背景

地貌是构成地理环境的基本要素之一，对地貌的三维可视化表示方法的研究是地图设计制作的重要命题。科学、合理的地形起伏表现形式，便于读者从地图上获取大量的地理空间信息，并具有较强的立体视觉效果。

等高线具有能详细刻画地貌特征、便于图上测量等优点，所以 19 世纪以来被公认为是一种科学、准确而可靠的地貌表示法，应用非常广泛。

目前，对地形数据的表达方式有很多种，其中等高线、规则格网和不规则格网是最常见的方法。而规则格网和不规则格网由于自身特点，更适合于计算机的存储和处理。但是，等高线仍然具有无可替代的地位。它通过一组一维曲线来表达地形的高度、走向等信息，科学形象的描述了三维地形的各种信息。通过等高线的分布关系，我们可以直观的了解地形中的山脉、谷地、洼地等的分布。当前数据源已经发生了革命性的变化，很多的地形信息采集途径是用格网的形式直接获取并进行存储的，因此，基于 DEM 格网绘制等高线的算法就显得很重要。格网又通常为规则格网的形式，所以本文就基于 DEM 规则格网绘制等高线的算法展开讨论，并将其制作成软件的形式。

1.2 本论文的价值及其意义

在地理空间信息的表达中地形要素占据着主要地位，是其他要素内容的基础参考。当前，在地形数据的诸多表达手段中，等高线、规则矩形格网 (GRID) 和不规则三角网 (TIN) 经成为三种最重要而有效的工具，其中后两种数据格式致密覆盖于制图区域，更适合于用计算机处理连续三维空间的需要，是很多工程应用的基础。另一方面人们也认识到，虽然计算机可以快捷的将 GRID 和 TIN 转化为诸如彩色晕渲图等视觉效果更为新颖的表达方式，但是在兼顾传递定性和定量信息方面，等高线仍然有着无可比拟的优势，它通过成组的一维曲线来传递地形的高度、走向、陡缓等信息，科学、形象的表达连续的三维地貌微观形态和宏观特征，读者

可以轻易的通过识别相邻等高线的距离、相对方向来获得局部地形参数以及区域内山脉、谷地、洼地等地貌形态的分布和结构关系。常规等高线生产中,制图人员通过选择等高距以及相关的地貌综合策略,体现了面向对象的思想。因此,如果说 GRID 和 TIN 是数字环境中地形信息表达的基础,那么我们可以认为等高线是高层次的表达手段。不过,随着数据获取方式的变革,当前数据源已经发生了革命性的转变,大多新型的地形信息采集途径是以格网 (GRID) 的形式直接获取并存储的,为了适应于高层次的表达,从 GRID 到等高线的算法就显得非常重要。

1.3 本论文的国内外研究现状

近些年来随着科学技术特别是计算技术的迅速发展,在 DEM 数据获取方法、数据存储和数据处理速度方面已经取得了一些突破性进展。DEM 数据采集方式已从最初的摄影测量数据采集法发展到利用 GPS、激光扫描、干涉雷达等新型技术。

国内外对 DEM 数据提取在技术上从不同的角度进行了大量的研究,目前,研究主要集中在等高线图层分割与矢量化、DEM 数据内插方法和 DEM 数据生成质量控制三个方面。Alirza Khotanzad 等提出了一种利用模板匹配的方法从彩色地形图分割等高线的方法[1];王月兰提出了关于地形图图层分割问题的基于信息融合的聚色算法[2];Ohlander R 等提出了一种生成三角网 DEM 的三角网生成算法[3]。

目前,对等高线跟踪提取主要是采用半自动矢量化跟踪方法,国内外一些扫描地形图矢量化软件系统,如美国 Able Software 公司的 R2V 扫描矢量化系统、日本 MARIS 系统、北京吉威特公司的 GeoScan 系统、武汉大学的 GeoStart 系统和 MAPCAD 系统等。这些软件往往对处理对象要求比较高,对于扫描质量不高的扫描地形图,处理起来难度比较大。

第 2 章 相关知识概述

2.1 DEM 的定义

所谓的 DEM (Digital Elevation Models, 即数字高程模型), 是将地面形态用 X, Y, Z 坐标的数字形式进行描述, 在生产中有着很广泛的应用。它与传统的地形图不同, 在各个方面都体现出其自身的优越性。目前, 国内外都将其作为国家基础地理信息的重要产品。

DEM 的建立有多种方法, 按照数据源及采集方式的不同可以分为: ①直接从地面测量得到 DEM 数据, 如利用 GPS、全站仪、野外测量等等; ②从航空或航天影像得到 DEM 数据, 由摄影测量获取, 如解析测图、数字摄影、立体坐标仪观测及空三加密法等等; ③从现有地形图上采集得到 DEM 数据, 如格网读点法、数字化仪手扶跟踪及扫描仪半自动采集然后通过内插生成 DEM 等等。DEM 内插方法主要有分块内插、部分内插和单点移面内插三种。

因为 DEM 描述的是地面高程信息, 它在测绘、水文、气象、地貌、地质、土壤、工程建设、通讯、气象、军事等国民经济和国防建设以及人文和自然科学领域有着广泛的应用。在防洪减灾方面, DEM 是进行水文分析如汇水区分析、水系网络分析、降雨分析、蓄洪计算、淹没分析等的基础; 在无线通讯上, 可用于蜂窝电话的基站分析等等。

DEM 主要有三种表示模型: 规则格网模型、不规则格网模型和等高线模型。

①规则格网模型

规则格网模型有正方形、矩形、三角形等。空间被分成无数的规则格网, 每一个格网对应于一个高程值。正方形的规则格网是最简单的一种, 它的数据排列整齐, 在计算机中可将其存储于一个二维数组, 便于存储和处理, 因此应用最为广泛。图 2.1 为正方形的 DEM 规则格网示意图。

33	12	67	45	22
16	54	73	44	58
40	56	34	66	55
26	37	68	57	44
46	30	61	35	30

图 2.1 正方形 DEM 规则格网示意图

由于规则格网自身的特点,在使用计算机进行处理时很方便,使得它成为 DEM 最广泛的使用格式, 本论文所讨论的也正是这种格式。

②不规则格网模型

虽然规则格网 DEM 模型有着很多的优点,但是也存在一些缺点。如在同一幅地形图中,各个区域地形的起伏程度不一样,为了兼顾地形复杂的区域,在地形平坦的地方就会出现大量的数据冗余。不规则三角网模型(TIN)正是为了弥补这一缺点而设计的。TIN 模型的建立首先要根据区域内有限点的集合将区域分割为相连的三角网,区域内的所有点都位于三角形的顶点、边或三角形内。如果点不在顶点上,该点的高程值一般采用线性内插法得到。TIN 模型和格网 DEM 模型相比,在数据存储方式上要复杂很多,因为它的平面坐标无法从数组下标中得到(格网 DEM 模型可以得到),所以存储方式不能像格网 DEM 模型一样采用二维数组存储,这也使得数据处理相对格网 DEM 模型要复杂。

2.2 CNSDTF-DEM 数据格式简介

格网虽然以栅格形式存贮,但不宜直接采用 TIFF 或 BMP 文件,所以需定义格网的数据交换格式。CNSDTF-DEM 是中国地球空间格网数据交换格式,数据文件包含两部分:文件头和数据体。

文件头的信息分两类：基本的且必须的信息和扩充的附加信息。扩充部分可以省略。

文件头的基本组成单元是项目，格式为“项目名：项目值”，每个项目单独占一行。文件头中各项目的内容逐条说明如下：

- (1) DataMark-----中国地球空间数据交换格式-格网数据交换格式(CNSDTF-RAS 或 CNSDTF-DEM)的标志。基本部分，不可缺省。
- (2) Version-该空间数据交换格式的版本号,如 1.0。基本部分,不可缺省。
- (3) Unit-----坐标单位,K 表示公里,M 表示米,D 表示以度为单位的经纬度,S 表示以度分秒表示的经纬度(此时坐标格式为 DDDMMSS. SSSS, DDD 为度, MM 为分, SS. SSSS 为秒)。基本部分，不可缺省。
- (4) Alpha-----方向角。基本部分，不可缺省。
- (5) Compress-----压缩方法。0 表示不压缩,1 表示游程编码。基本部分，不可缺省。
- (6) X0-----左上角原点 X 坐标。基本部分，不可缺省。
- (7) Y0-----左上角原点 Y 坐标。基本部分，不可缺省。
- (8) DX-----X 方向的间距。基本部分，不可缺省。
- (9) DY-----Y 方向的间距。基本部分，不可缺省。
- (10) Row-----行数。基本部分，不可缺省。
- (11) Col-----列数。基本部分，不可缺省。
- (12) Hzoom-----高程放大倍率。基本部分，不可缺省。设置高程的放大倍率,使高程数据可以整数存贮,如高程精度精确到厘米,高程的放大倍率为 100。如果不是 DEM 则 HZoom 为 1。

数据体部分为格网的值，它是格网的要素类型编码或高程，格网数据的存贮采取从北到南，从西到东的顺序，并以纯文本存储。在存储时，第一个数据为左上角原点 (X_0, Y_0) 的高程，然后依次是第一行其它点的高程，每一个高程值之间用空格分开。通常为了美观和方便阅读，每行为 10 个数据，每列数据右对齐。当格网的一行结束时换行，并开始下一行数据，直到数据全部存储结束。

2.3 等高线的定义及其特点

所谓等高线是指在地形图中高程相等的点所连成的闭合曲线在标准平面上的投影。也可将等高线看成是不同海拔高度的水平面与实际地面的交线，所以等高线通常是闭合曲线，当只考虑一定区域时，等高线也可为开曲线。根据等高线的定义，其有以下特性：

- ①同一等高线上的点，高程相等；
- ②同一幅地形图内，除悬崖以外，不同高程的等高线不相交；
- ③相邻等高线的高程差一般是相同的，所以等高线越密集的地方表示地面坡度越大，反之，地面坡度越小。

因此等高线能够直观地反映出地面起伏趋势和地面形态的特征。

2.4 等高线跟踪相关算法

目前，关于地形图等高线的跟踪识别问题，已经有了一些研究成果，同时也在实际工作中得到了一些应用。以下是常见的三种等高线跟踪算法：

1、直接寻找线状地物的中心法

这种方法在地形图早期的自动化输入中起到了一定的作用，它具有快速的特点。具体算法为：

- 1) 以当前点和前 k 个点的连线作为线条的走向 P 。由此确定后续点的搜索方向和区域；

- 2) 以当前点 x 为轴心, 以走向 P 为中心线, 向左右呈扇形搜索, 次序为中左、右、左、右……, 每次偏移一个像素, 寻找下一个点, 完成一次扇形搜索后仍未找到后继点, 则存在断点;
- 3) 步长加 1, 进行下一次搜索;
- 4) 重复步骤 3, 直到找到下一个点为止, 如果步长加到最大仍未找到后继点则线条终止;
- 5) 重复 1-4 步骤, 直到等高线跟踪结束。该算法对于单像素的线状地物, 有着较好的跟踪效果。但是, 在实际情况下, 由于经过扫描输入的地形图中的等高线般是具有一定宽度的线状地物。采用这种方法, 无法得到等高线的中心线, 效果并不理想。尤其对于较宽的等高线, 在跟踪的过程中, 所得到的等高线与等高线的中心存在较大的偏差。另外, 当等高线的曲率变化较大时, 跟踪的效果更差。

为了改善该方法的不足, 人们也提出了一些改进的方法, 如增加对搜索所得点的中心判定及修正方法等。在一定程度上提高了跟踪识别的效果。

2、不规则带状地物的跟踪识别

该方法基于以下的思想:

等高线在一定程度上类似与带状地物, 因此可采用类似的方法。以等高线一侧为基础的中心线跟踪识别法, 可直接获得等高线中心线的精确位置。如图 2.2 所示:

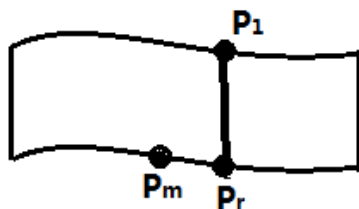


图 2.2 不规则带状地物的跟踪识别

令等高线与线段的交点为 P_1 、 P_r ，则中点位置 P_m 为：

$$X_m = (X_1 - X_r) / 2, Y_m = (Y_1 - Y_r) / 2$$

具体算法如下：

- 1) 等高线的右侧为当前点出发搜索下一个点 P_r ；
- 2) 作当前等高线中心线的延长线为等高线中心线的走向；
- 3) 从点 P_r 向等高线左边缘作与走向垂直的直线交左侧边缘于 P_1 ；
- 4) 取 P_r 、 P_1 线段的中点为等高线的新中点，并将等高线的中心线延长至这一点；
- 5) 重复步骤 1--4 直到结束。

该算法在对一般均匀带宽的地物的跟踪识别时有较好的效果。但是没考虑到扫描地形图中等高线出现的一些特殊情况。例如：虽然等高线在理论上是宽度一定的棕色曲线，但对等高线线宽由于某些原因有一定变化的区域，其跟踪效果较差。

3、利用数学形态学的等高线跟踪识别算法

该算法的基本思想是：

经过分色和色域滤波二值化后形成的等高线，线条宽度是不均匀的(一般为 2--5 个像素)，为了提高识别精度，简化跟踪识别的策略，首先必须进行细化。

在图像处理中，细化定义为：确定一幅图像骨架的过程。而一幅图像的骨架是指能够描述原图像几何与拓扑特性性质的最小子集。按上述定义，对等高线而言，细化过程应该满足三个要素：

- 1) 细化后的等高线线宽为一个像素；

- 2) 细化后的等高线应该保持原有的 8 邻域连通性;
- 3) 细化过程应该保证各向同性, 以保证最终的骨架位置误差小于一个像素。

根据以上要求, 由数学形态学求骨架算子, 可以设计出来一套方法, 求得等高线的骨架。

该算法对于宽度均匀的等高线跟踪识别效果比较理想。但是, 该算法速度比较慢, 而且同样对于等高线因为某些原因, 本身有一定损失时, 无法自动补足损失对识别的影响。因此, 也不是最理想的跟踪识别方法。

第 3 章 基于 DEM 规则格网绘制等高线方法研究

3.1 格网数据的高斯滤波处理

为了使从 DEM 数据中提取出的等高线更加平滑，可以先对其进行高斯滤波。以下为高斯滤波的简介。

高斯滤波实质上是一种信号的滤波器，其用途是信号的平滑处理，我们知道数字图像用于后期应用，其噪声是最大的问题，由于误差会累计传递等原因，很多图像处理教材会在很早的时候介绍 Gauss 滤波器，用于得到信噪比 SNR 较高的图像（反应真实信号）。与此相关的有 Gauss-Laplace 变换，其实就是为了得到较好的图像边缘，先对图像做 Gauss 平滑滤波，剔除噪声。

滤波器是一个数学模型，通过这个模型来将图像数据进行能量转化，能量低的就排除掉，噪声就是属于低能量部分。

其实，用编程来计算的话就是一个模板运算，拿图像的八连通区域来说，中间点的像素值就等于八连通区的像素值的均值，这样达到平滑的效果。

若使用理想滤波器，会在图像中产生振铃现象。采用高斯滤波器的话，系统函数是平滑的，避免了振铃现象。

本论文中所用到的高斯滤波算法程序示例如下所示：

```
int i, j;

for (int i=1; i<=row-2; i++)

{

    for (int j=1; j<=col-2; j++)

    {
```



```
matrix[i][j]=(matrix[i-1][j-1]+2*matrix[i-1][j]+matrix[i-1][j+1]+2*matrix[i][j-1]+2*matrix[i][j+1]+matrix[i+1][j-1]+2*matrix[i+1][j]+matrix[i+1][j+1]+4*matrix[i][j])/16;}}
```

该算法可用图 3.1 简要说明：

1	2	1
2	4	2
1	2	1

图 3.1 高斯滤波图示

最中间(红色)的表示当前高程值，经过高斯滤波后它的值为周围 8 个点和本身如图倍数的和再除与 16。

3.2 等高线的扫描与绘制

3.2.1 基于 DEM 提取等高线概述

基于 DEM 规则格网绘制等高线的方法大致可分为：

- a) 高次曲面内插法；
- b) 格网线性内插法。

高次曲面内插法是首先根据一定范围内的格网点拟合曲面，接着在得到的拟合曲面上用横截平面得到等高线。

格网线性内插法则是通过内插高程点的方法得到绘制等高线所需点，然后将这些点依次连接构成等高线。其过程大致为：首先在格网中选择一个起始点，然后按照一定的规则遍历格网，当遍历到边界或回到起始点，则一条等高线被找到，结束本次遍历转而进行下一次遍历操作。在此方法中，重点在于内插点和遍历的

过程。显然，在一次遍历过程中，每一条格网边都只可能存在一个内插点（因为总是将格网边看成线性变化的）。因此，为了避免重复遍历也被遍历过的边而浪费大量的时间，定义一个 bool 型变量来标记某条边是否已被遍历过，在每次遍历时首先判断该变量，如果已被遍历过，则跳过此边。

3.2.2 内插高程点方法

在当前扫描等高线的高程 z 与当前遍历单元边的两端点 a 和 b 的高程 z_1 和 z_2 满足如下条件：

$$z_1 \geq z \geq z_2 \text{ 或 } z_1 \leq z \leq z_2$$

即： z 在左端点和右端点的高程值之间。

时，单元边上存在内插高程点。将格网单元边的高程看作线性变化，那么，该内插高程点的坐标为 (x, y) 。

当该单元边为横向边时：

$$x = x_1 + \frac{z - z_1}{z_2 - z_1}(x_2 - x_1), \quad y = y_1 = y_2$$

当该单元边为纵向边时：

$$x = x_1 = x_2, \quad y = y_1 + \frac{z - z_1}{z_2 - z_1}(y_2 - y_1)$$

有时，待内插点的高程等于单元边的端点，此时，我们采用将该端点高程增加一个微小量的方法。这样就避免了内插点在端点的情况，使得所有内插点都在端点之间，减小了问题的难度，同时，这样做也不会引起很大的误差。

3.2.3 格网的遍历方法

格网的遍历分为三种：横向边界、纵向边界和内部网络。而在每一次中又按照从左到右，从上到下的顺序。之所以采用这样的顺序，是因为高程数据的存储采用的是二维数组，采用这样的顺序能够用简单的循环语句实现。程序示例如下：

1. 扫描横向边界

```
for(i=0;i<=row-1;i+=row-1)

{

    for (j=0;j<=col-2;j++)

        { //在此处判断内插点是否存在、计算坐标值及将该点和当前高程值记录下。}

}
```

2. 扫描纵向边界

```
for(j=0;j<=col-1;j+=col-1)

{

    for (i=0;i<=row-2;i+=1)

        { //在此处判断内插点是否存在、计算坐标值及将该点和当前高程值记录下。}

}
```

3. 扫描内部格网

```
for(j=1;j<=row-2;j+=1)

{

    for (i=0;i<=col-2;i++)

        { //在此处编写判断内插点是否存在,并计算坐标值及将该点和当前高程值记录下。}

}
```

如果采样点数目(网格分辨率)有限,那么会出现一个格网单元跨过鞍部的情形。该网格单元的四条单元边上就可能同时都有内插高程点,(徐庆荣等,1997)认为只要不连接对边产生相交,其他两种连接都可以。但是人们往往希望得到更为合适的结果,然而在网格内部的地形走向无法判断的情况下,就需要设计对应的策略来决定等高线下一步内插走向,现有的方法包括最短连接、固定方向搜索、添加辅助线或点(对角线或中心点)、曲面法以及外围邻居判断法(刘勇奎等,1997)等。最短连接法通常是从当前进入点出发计算到两个邻边(不考虑对边)上内插点的距离,取其小者连接,如果遍历的次序不同都会产生不同的结果,如图 3.2:

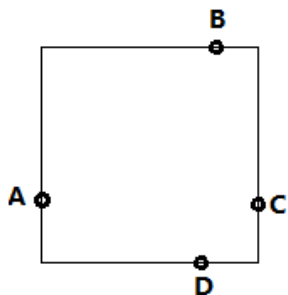


图 3.2 交叉网格的处理

四条单元边上各有一个等高点,如果搜索是从 A 进入的话,则连接的方式 A-D 和 B-C,而如果是从 C 进入,则连接就变为 C-D 和 A-B。本文对此稍加改进,考虑两种连接方式下各自得到的总边长,取其小者。如图 1 中,如果长度 $L_1 = IABI + ICDI < L_2 = IADI + IBCI$ 则取连接方式为 AD 和 BC,若 $L_1 > L_2$,则取 AB 和 CD 连接。这样无论从哪个方向进入此网格,其最终连接的方式都是唯一的。

3.2.4 等高线的扫描

等高线的扫描基于上一节(3.2.3)所提到的格网遍历,整个扫描过程如下:

1. 遍历过程中找到一个内插高程点;
2. 以该内插高程点所在格网边为起始边,依次向周围最近的 6 条格网边扫描,以最短距离为原则扫描下一内插高程点。

3. 重复步骤2，直到内插点与起始点重合或到格网边界。

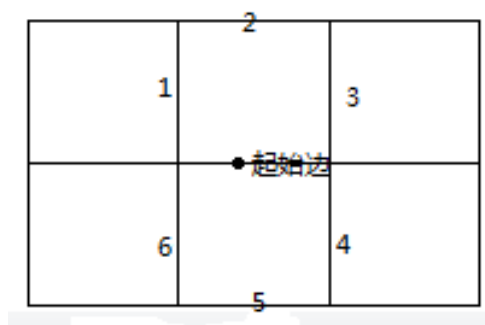


图 3.3 等高线的扫描

人们在阅读使用传统介质等高线地图时，可以从标注高程的一组等高线中识别出地势的走向，而在仅有一根等高线的情况下，即便加注了高程，也是无法判断出有意义的高低变化信息，但是如果为这根等高线标注了方向(例如用箭头表示方向并缺省认为左高右低)，则可以获得局部的基本信息。而对于没有视觉理解功能的计算机而言情况更甚，为了辅助基于等高线的数字地形分析(如等高线树的构建、分层设色填充等)，等高线的方向必须是一致的(王涛等，2004)，对于计算机而言等高线的方向可以蕴涵在等高线节点的先后顺序之中，仍旧假设缺省的等高线方向为左高右低。如果原始数据已经是等高线，那么可以通过一定的处理将等高线方向调整为一致的。而如果是从格网高程模型中提取等高线，那么对上面的算法稍加修改即可很方便的得到方向一致化的等高线。直接使用上面的算法所得到的闭曲线都是逆时针的，但是在高程上的方向并不一定是一致的，即并不一定都是左高右低，这种问题发生在洼地部分。

3.2.5 等高线的绘制

在扫描得到所有的等高线后，只要将每条等高线的点依次连接起来就可以完成等高线的绘制。为了使读者更容易从等高线中看出地形的特征，我们可以将高程值与颜色一一对应，不同高程的等高线对应不同的颜色。程序示例如下：

```
{  
  
    predraw=pDoc->LineSet.GetAt(i);  
  
    CPen m_pen;  
  
    int k=int(255*sqrt((predraw->height-pDoc->zmin)/pDoc->zmax));  
  
    m_pen.CreatePen(PS_SOLID,10,RGB(255,k,255-k));  
  
    pDC->SelectObject(&m_pen);  
  
    predraw->DrawLine(pDC);  
  
}
```

其中 DrawLine () 函数如下所示:

```
DrawLine(CDC *pDC)  
  
{  
  
    int pointnum = m_Points.GetSize();  
  
    bool closed=false;  
  
    for(int i=0;i<pointnum-1;i++)  
  
    {  
  
        pDC->MoveTo(m_Points.GetAt(i).x,m_Points.GetAt(i).y);  
  
        pDC->LineTo(m_Points.GetAt(i+1).x,m_Points.GetAt(i+1).y);  
  
    } return;  
  
}
```

3.3 总结

在基于 DEM 规则格网绘制等高线时，首先对高程数据进行高斯滤波处理，可以将 DEM 高程数据中的一些突变情况进行平滑处理。在下图 3.4 中，上面的图为未进行高斯滤波的等高线图，下面的图为经过高斯滤波后的等高线图。

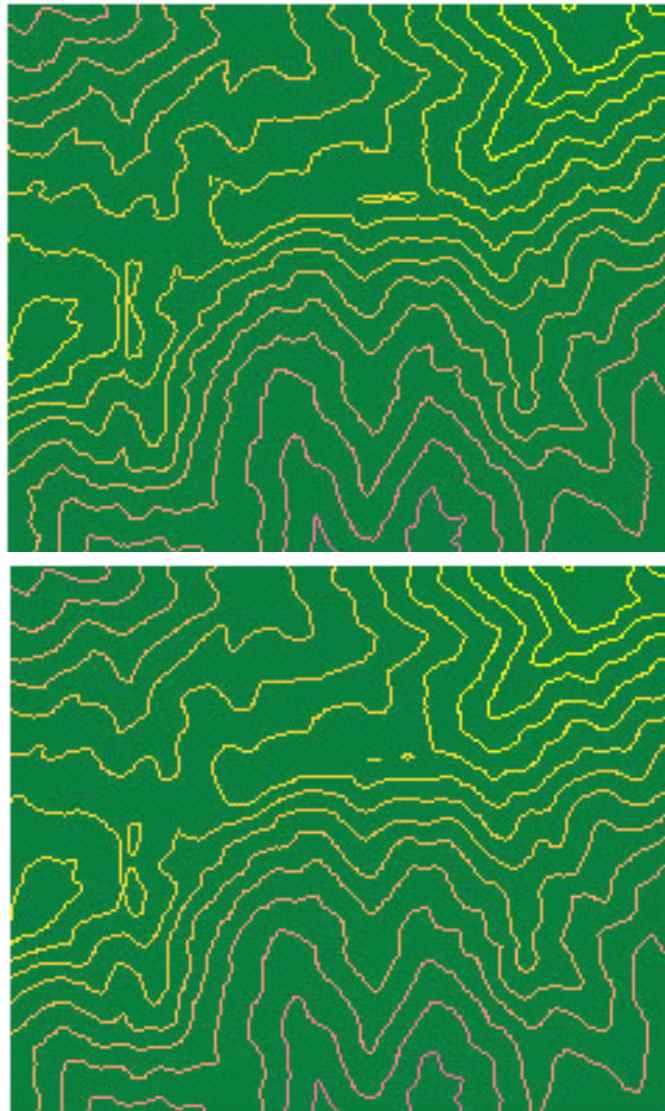


图 3.4 高斯滤波效果对比

第 4 章 软件系统的设计

4.1 程序语言和开发平台简介

为了完成本论文的任务要求，选择 C++ 进行程序的编写，并使用微软公司的 Visual C++6.0 作为软件的编写平台。在开始编写之前，有必要对 C++ 及 Visual C++6.0 做简要的介绍。

4.1.1 C++的发展历史

C 语言之所以要起名为“C”，是因为它是主要参考了那个时候的一门叫“B”的语言，它的设计者认为 C 语言是 B 语言的进步，所以就起名为 C 语言；但是 B 语言并不是因为之前还有个 A 语言，而是 B 语言的作者为了纪念他的妻子，他的妻子名字的第一个字母是 B；当 C 语言发展到顶峰的时刻，出现了一个版本叫 C with Class，那就是 C++ 最早的版本，在 C 语言中增加 class 关键字和类，那个时候有很多版本的 C 都希望在 C 语言中增加类的概念；后来 C 标准委员会决定为这个版本的 C 起个新的名字，那个时候征集了很多种名字，最后采纳了其中一个人的意见，以 C 语言中的 ++ 运算符来体现它是 C 语言的进步，所以就叫 C++，也成立了 C++ 标准委员会；美国 AT&T 贝尔实验室的本贾尼·斯特劳斯特卢普 (Bjarne Stroustrup) 博士在 20 世纪 80 年代初期发明并实现了 C++（最初这种语言被称作“C with Classes”）。一开始 C++ 是作为 C 语言的增强版出现的，从给 C 语言增加类开始，不断的增加新特性。虚函数 (virtual function)、运算符重载 (operator overloading)、多重继承 (multiple inheritance)、模板 (template)、异常 (exception)、RTTI、命名空间 (name space) 逐渐被加入标准。1998 年国际标准组织 (ISO) 颁布了 C++ 程序设计语言的国际标准 ISO/IEC 1488-1998。C++ 是具有国际标准的编程语言，通常称作 ANSI/ISO C++。1998 年是 C++ 标准委员会成立的第一年，以后每 5 年视实际需要更新一次标准，下一次标准更新将是在 2003 年，目前我们一般称该标准 C++03。遗憾的是，由于 C++ 语言过于复杂，以及他经历了长年的演变，直到现在 (2003 年) 只有 Visual C++ 2003 CTP 开发环境的编

译器完全符合这个标准。另外，就目前学习 C++ 而言，可以认为他是一门独立的语言；他并不依赖 C 语言，我们可以完全不学 C 语言，而直接学习 C++。根据《C++ 编程思想》(Thinking in C++) 一书所评述的，C++ 与 C 的效率往往相差在正负 5% 之间。所以有人认为在大多数场合 C++ 完全可以取代 C 语言 (然而我们在单片机等需要谨慎利用空间、直接操作硬件的地方还是要使用 C 语言)。

4.1.2 C++ 的类

C++ 中，我们通过定义类来定义自己的数据结构。类机制是 C++ 中最重要的特征之一。事实上，C++ 设计的主要焦点就是使所定义的类型的行为可以像内置类型一样自然。

1. 类的定义：

类是同一类所有对象的变量和方法的蓝图或原型。例如，可以定义一个包含当前档位等实例变量的自行车类。这个类也定义和提供了实例方法（变档、刹车）的实现。实例变量的值由类的每个实例提供。因此，当你创建自行车类以后，必须在使用之前对它进行实例化。当创建类的实例时，就建立了这种类型的一个对象，然后系统为类定义的实例变量分配内存。然后可以调用对象的实例方法实现一些功能。相同类的实例共享相同的实例方法。除了实例变量和方法，类也可以定义类变量和类方法。可以从类的实例中或者直接从类中访问类变量和方法。类方法只能操作类变量 - 不必访问实例变量或实例方法。

系统在第一次在程序中遇到一个类时为这个类建立它所有类变量的拷贝，这个类的所有实例共享它的类变量。

2. 类和对象：

对象和类的说明很相似。实际上，类和对象之间的差别经常是一些困惑的起源。在现实世界中很明显，类不是它描述的对象 - 自行车的蓝图不是自行车。但是在软件中就有难区分类和对象。这部分是由于软件对象只是现实世界的电子模型或抽象概念。但是由于很多人用“对象”指类和它们的实例这两者。

3. 类的好处:

对象提供了模型化和信息隐藏的好处。类提供了可重用性的好处。自行车制造商一遍又一遍地重用相同的蓝图来制造大量的自行车。软件程序员用相同的类,即相同的代码一遍又一遍地建立对象。

4.1.3 Visual C++简介

Visual C++是一个功能强大的可视化软件开发工具。自 1993 年 Microsoft 公司推出 Visual C++1.0 后,随着其新版本的不断问世,Visual C++已成为专业程序员进行软件开发的首选工具。

虽然微软公司推出了 Visual C++.NET(Visual C++7.0),但它的应用的很大的局限性,只适用于 Windows 2000, Windows XP 和 Windows NT4.0。所以实际中,更多的是以 Visual C++6.0 为平台。

Visual C++6.0 不仅是一个 C++编译器,而且是一个基于 Windows 操作系统的可视化集成开发环境(integrated development environment, IDE)。Visual C++6.0 由许多组件组成,包括编辑器、调试器以及程序向导 AppWizard、类向导 Class Wizard 等开发工具。这些组件通过一个名为 Developer Studio 的组件集成为和谐的开发环境。

Visual C++大概可以分成三个主要的部分:

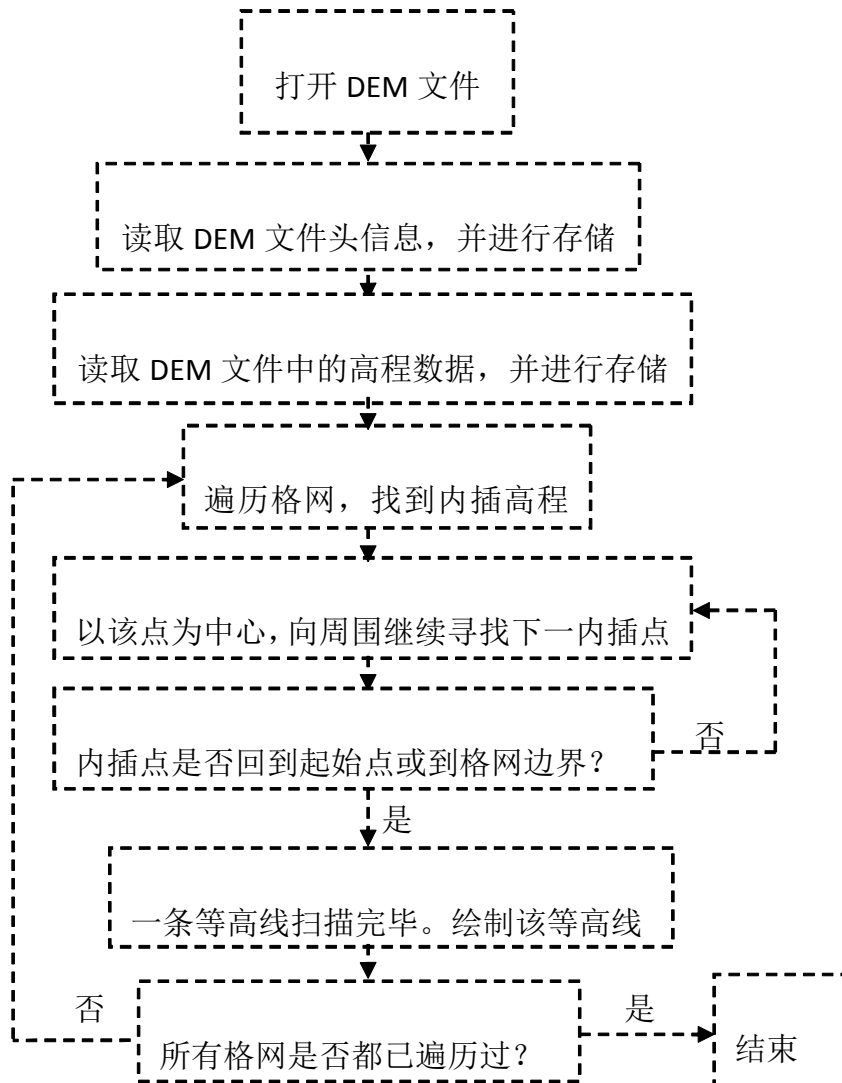
1. Developer Studio, 这是一个集成开发环境,我们日常工作的 99%都是在它上面完成的,再加上它的标题赫然写着“Microsoft Visual C++”,所以很多人理所当然的认为,那就是 Visual C++了。其实不然,虽然 Developer Studio 提供了一个很好的编辑器和很多 Wizard,但实际上它没有任何编译和链接程序的功能,真正完成这些工作的幕后英雄后面会介绍。我们也知道,Developer Studio 并不是专门用于 VC 的,它也同样用于 VB, VJ, VID 等 Visual Studio 家族的其他同胞兄弟。所以不要把 Developer Studio 当成 Visual C++,它充其量只是 Visual

C++的一个壳子而已。

2. MFC。从理论上讲，MFC 也不是专用于 Visual C++，Borland C++，C++Builder 和 Symantec C++同样可以处理 MFC。同样使用 Visual C++编写代码也并不意味着一定要用 MFC，只要愿意，用 Visual C++来编写 SDK 程序，或者使用 STL，ATL，一样没有限制。不过，Visual C++本来就是为 MFC 打造的，Visual C++中的许多特征和语言扩展也是为 MFC 而设计的，所以用 Visual C++而不用 MFC 就等于抛弃了 Visual C++中很大的一部分功能。但是，Visual C++也不等于 MFC。

3. Platform SDK。这才是 Visual C++和整个 Visual Studio 的精华和灵魂，虽然我们很少能直接接触到它。大致说来，Platform SDK 是以 Microsoft C/C++编译器为核心（不是 Visual C++），配合 MASM，辅以其他一些工具和文档资料。上面说到 Developer Studio 没有编译程序的功能，那么这项工作是由谁来完成的呢？是 CL，是 NMAKE，和其他许许多多命令程序，这些我们看不到的程序才是构成 Visual Studio 的基石。

4.2 软件系统流程图



4.3 数据结构设计

我们来回顾一下本论文的任务——基于 DEM 规则格网的等高线绘制，这是本论文的题目。

那么，DEM 文件中的数据如何有效的进行组织是我们遇到的第一个问题。只有将这些数据有效的组织起来，完成论文提出的任务才可能实现。这些数据将在程序运行的开始用前面提到的读取 DEM 文件部分程序将其从 DEM 文件中读入，并进行存储，方便后面程序的调用。根据以上分析，定义一个 ContourDoc 类。见附录 1。

格网的高程数据已经有效的进行的组织存储了，那么，接下来就要开始本论文的重点——扫描等高线了，在找到一个内插点时，我们就要将它进行存储。由于这是组成一条等高线的一个点，那么，我们就将其存放在一个数组中，一个数组就对应了一条等高线。根据这样的想法，定义一个 ContourLine 类。见附录 2。

4.4 DEM 文件的数据读取

在 DEM 文件中存储着本论文任务所要的数据。为了完成本任务，首先要把我们所需要的数据读入到程序中。DEM 文件内容示例如下所示：

DataMark:CNSDTF-DEM

Version:1.0

Unit:M

Alpha:0.000000

Compress:0.000000

X0:258000.000

Y0:324000.000

DX:22.500

DY:22.500

Row:321

Col:481

ValueType:Integer

Hzoom:1000

192743 191068 187814 181388 173357 165286 157185 152266

..... 省略

4783 4748 4713 4679 4643 4608 4573 4539

根据以上的 DEM 文件内容示例,可以编写程序来读取我们所需要的数据内容,读取到的数据将被存储在变量中,以便后续使用。

1. 读取 CNSDTF-DEM 文件头信息

CNSDTF-DEM 数据格式的文件头包含着有关该数据文件的相关信息,它是我们对数据体中的数据进行处理时所必须用到的。程序示例见附录 3。

2. 读取高程数据

高程数据是 CNSDTF-DEM 数据格式中的第二部分,它是基于 DEM 规则格网绘制等高线的主要数据。它在 CNSDTF-DEM 数据格式中的存储是有规律的,可以使用一个二维数组来对其进行存储,这样也同时方便后续的等高线提取。程序示例见附录 4。

第 5 章 软件的编写与调试

5.1 软件的编写

到此，所有的准备工作已结束。接下来的工作将是软件的实现，首先打开 Visual C++ 6.0 然后按如下步骤：

1. 点击“新建工程”，选择“MFC AppWizard(exe)”，工程名填入“contour”，选好工作目录，然后点击“OK”。

2. 出现如下图 5.1 所示界面，选择“D 基本对话框”，点击“完成”。

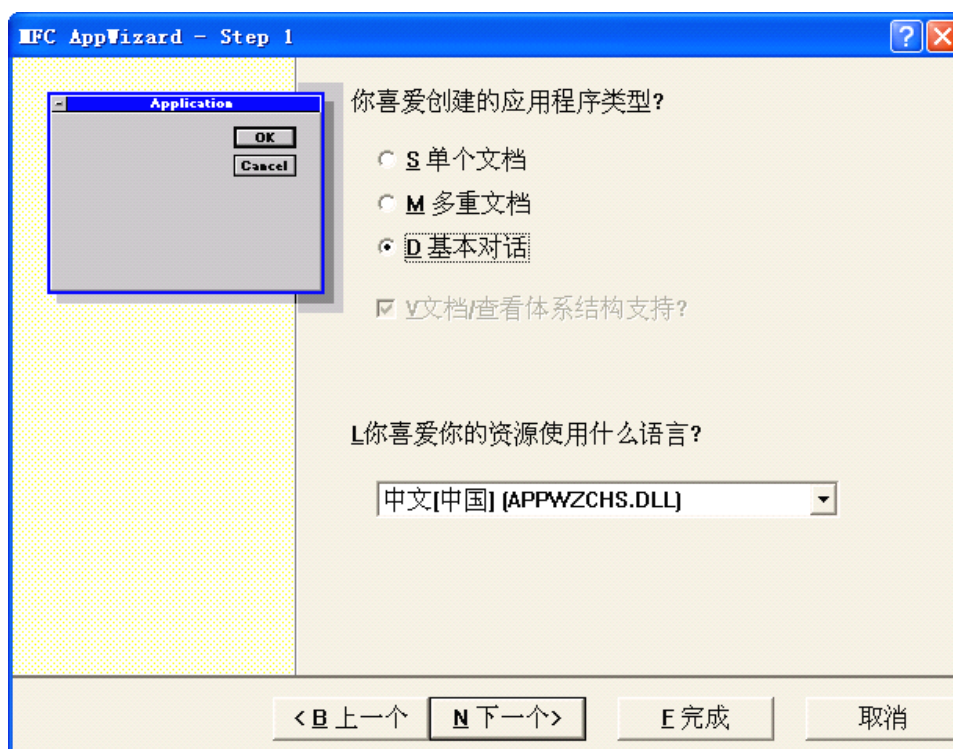


图 5.1 新建工程

3. 此时建立了一个基于对话框的工程。将 IDD_CONTOUR_DIALOG 设计为图 5.2 所示的样式，该对话框将作为本软件的主窗口。其中按钮 ID 为“IDC_FileOpen”，标题为“打开 DEM 文件”，两个 Edit box 的 ID 分别为 IDC_EDIT1 和 IDC_EDIT2，

右键点击，选择属性，在 Styles 选项卡设置为如图 5.3 所示，下方的 picture 控件 ID 为 “IDC_PICTURE”。



图 5.2 主对话框的界面设计

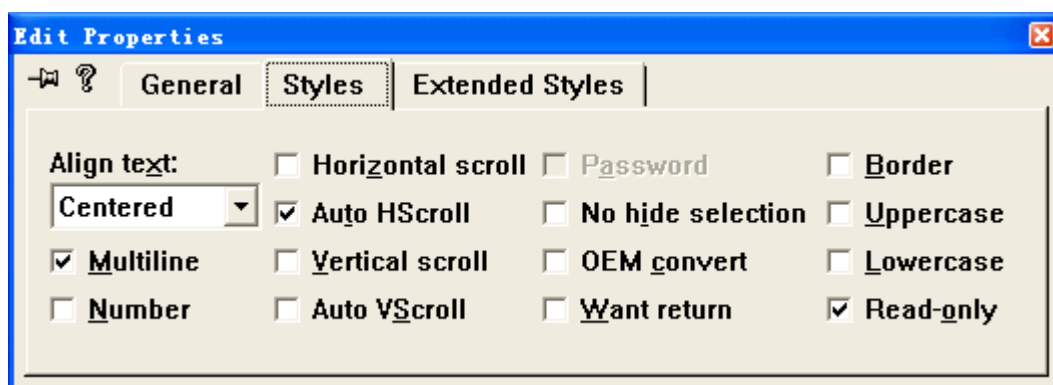


图 5.3 控件的属性设置

4. 添加成员变量。依次点击查看、建立类向导、Member Variables, Class name 处选择 CcourtourDlg, 选择 IDC_EDIT1, 点击 Add Variables。在 Member variable name 处填入 “m_str”, Category 处选择 “Value”, Variable type 处选择 “CString”, 点击 “OK”。同样方法添加 IDC_EDIT2 的成员变量 “m_str1”。

5. 再插入一个对话框, ID 为 “IDD_DIALOG1”, 设计为如图 5.4 所示。这个对话框用于在打开 DEM 文件后显示从 DEM 文件中读取到的一些简要信息, 使我们对该 DEM 文件有初步的了解, 这有助于后面将要进行的等高距的输入。两个 Edit box 的 ID 分别为 IDC_EDIT1 和 IDC_EDIT2, 其中 IDC_EDIT1 的属性设置与上一步中的两个 Edit box 相同。按钮 “OK” 和 “Cancel” 为原有的, 只需调整位置即可。

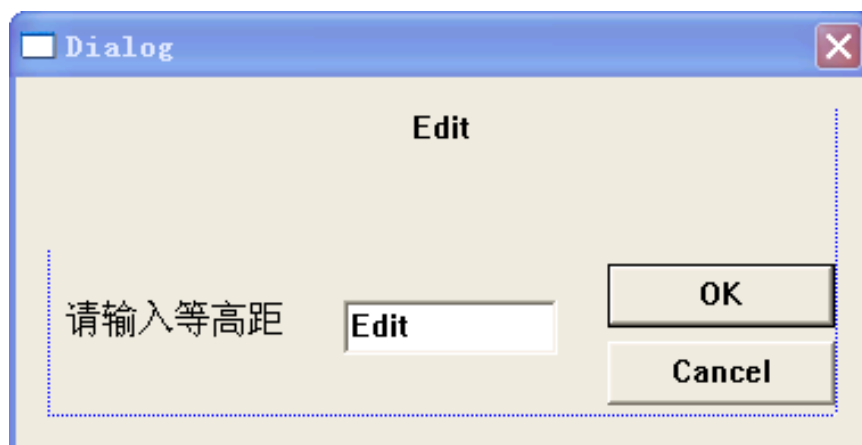


图 5.4 输入等高距提示对话框

6. 添加成员变量。分别为 IDC_EDIT1 和 IDC_EDIT2 添加成员变量 “m_text” (Variable type 为 CString)、“m_height” (Variable type 为 long)。

7. 添加类定义。如前面提到的, 将附录 1 和附录 2 所示的两个类添加到工程。

8. 添加 “打开 DEM 文件” 函数 “OnFileOpen()”。双击按钮 “打开 DEM 文件”, 弹出添加成员函数对话框, 在 “Member function name” 处输入该成员函数的函数名 “OnFileOpen”, 点击 “OK” 跳转至代码编辑处。添加函数见附录 5 所示。

9. 添加函数 Gaussfilter()、cacculation()、ScanLine()、RecruScan、

`intersectornot()`、`dist()`、`OnDestroy()`、`Integration()` 于函数“`OnFileOpen()`”之后。由于程序过长，不在本论文中展现。

5.2 软件的调试运行

完成了软件的编写，接下来就可以开始对其进行调试运行了。在此过程可以发现并改正一些错误。

依次点击 `Compile`、`Build`、`buildExecute`，在下方的调试窗口中查看是否有错误。若出现错误，则查看错误提示并改正错误直到没有错误。出现软件的主窗口，如图 5.5 所示：

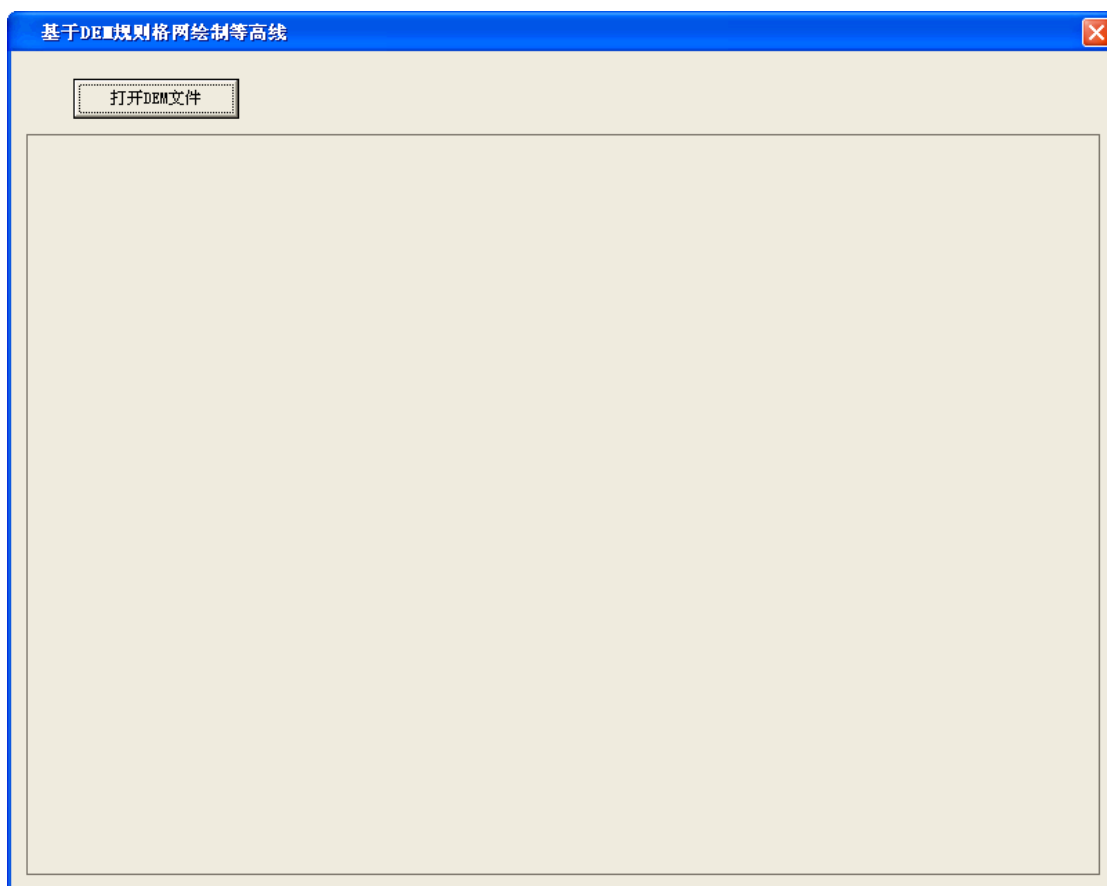


图 5.5 软件的主界面

此时，软件已正常运行，点击窗口中的“打开 DEM 文件”按钮，弹出“打开提示对话框”（如图 5.6 所示），选择 DEM 文件所在目录并选择 DEM 文件，点击打开。

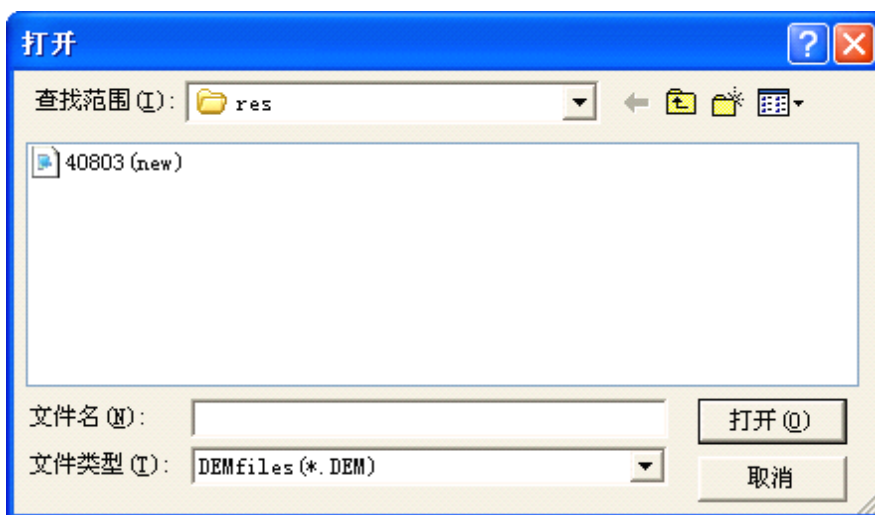


图 5.6 打开 DEM 文件对话框

弹出对话框（如图 5.7 所示），提示输入等高距，在输入框中输入等高距（这里的等高距应大于 25），点击“OK”按钮。

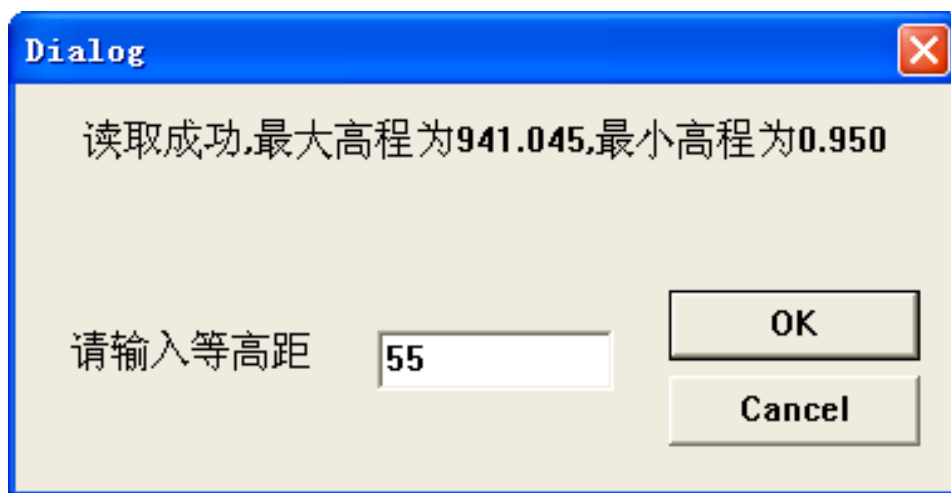


图 5.7 等高距输入对话框

等高线将自动绘制，如图 5.8 所示。

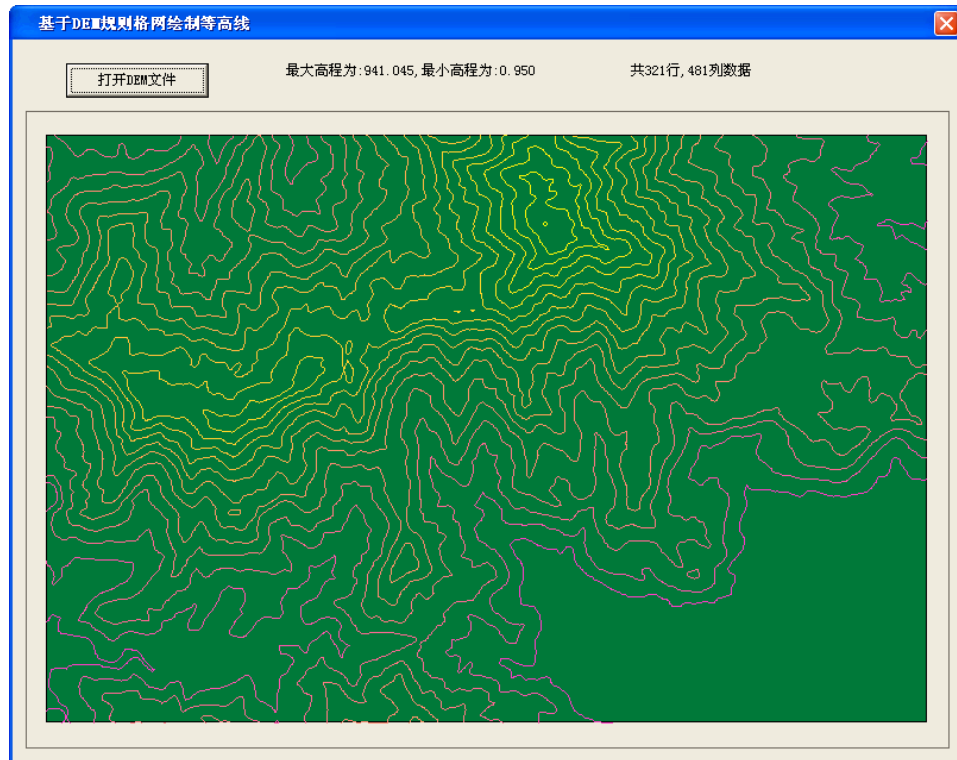


图 5.8 软件运行结果显示

到这里，从理论的介绍到编写程序并最终实现了基于 DEM 规则格网绘制等高线的软件编写和调试运行工作。这也同样完成了本论文最初提出的任务。

第 6 章 结束语

1. 规则格网的高程矩阵，可以很容易地用计算机进行处理，特别是栅格数据结构的地理信息系统。它还可以很容易地计算等高线、坡度坡向、山坡阴影和自动提取流域地形，使得它成为 DEM 最广泛使用的格式，目前许多国家提供的 DEM 数据都是以规则格网的数据矩阵形式提供的。

2. 等高线通常被存成一个有序的坐标点对序列，可以认为是一条带有高程值属性的简单多边形或多边形弧段。由于等高线模型只表达了区域的部分高程值，往往需要一种插值方法来计算落在等高线外的其它点的高程，又因为这些点是落在两条等高线包围的区域内，所以，通常只使用外包的两条等高线的高程进行插值。等高线通常可以用二维的链表来存储，第一维为一条等高线上的所有点的有序序列，第二维度为所有等高线的集合，绘制等高线可以通过两个 for 循环实现，外层循环用于循环遍历所有等高线，内层循环用于循环遍历某条等高线的所有点。

3. 在提取等高线之前应用高斯滤波方法对格网进行了滤波，降低了格网点存在突变情况是对等高线绘制的影响。

4. 研究了等高线提取时的内插问题和格网的遍历问题。内插时可以将格网边的两端点之间点高程看作线性分布，这样简化了等高线的提取。

5. 等高线的绘制采用不同颜色对应不同高程，使得等高线更加直观。

参考文献

- [1] Alireza Khotanzad, Edmund Zink. Contour Line and Geographic Feature Extraction from USGS Color Topographical Pager Maps. IEEE: TRANSACTIONS ANALYSIS AND MACHINE INTELLIGENCJS JANUARY, 2003
- [2] 王月兰. 彩色地形图聚色及等高线数字矢量化算法的研究: [硕士学位论文]. 国防科技大学, 1999-12
- [3] Ohlander R, Price K, Reddy D R. Picture segmentation using a recursive region splitting method .CGIP, 1988-8
- [4] 吴飞. 基于等高线数据建立高质量 DEM: [硕士学位论文]. 武汉: 武汉大学, 2005
- [5] 胡友元, 黄杏元. 计算机地图制图. 北京: 测绘出版社, 1987
- [6] 乔利军. DEM 数据提取关键技术研究: [硕士学位论文]. 长沙: 国防科技大学, 2005
- [7] 王涛, 毋河海. 一种从高程格网中提取等高线的算法. 测绘科学, 第 31 卷 (第二期): 108-110
- [8] 黄力, 邱涤珊, 王峰. GIS 中 DEM 的表示方法[J], 计算机与信息技术, 1999
- [9] Michael B Gousie. Converting Elevation Contours to a Grid, Dept of Computer Science
- [10] 赵纪凯. 军用地形图中等高线的跟踪识别研究. [硕士学位论文]. 西安电子科技大学, 2000
- [11] 王涛, 雷蓉. 从规则高程格网中提取等高线的优化算法研究. 青年优秀论文选登, 2006

致 谢

在此论文即将完成之际，我首先要感谢我的指导老师蒋玲老师在论文选题、撰写以及修改过程中付出的心血和劳动。在一学期的毕业设计工作中，老师给予了我无私的关怀和帮助，把我引入到这丰富多彩的地理信息系统世界中，并指导我完成了本论文的所有工作，他严谨的治学态度以及求实的科研作风永远是我学习的榜样，正是老师的严格要求使我顺利的完成了软件的编写及论文的撰写。

我还要感谢我的朋友们在论文撰写过程给予我的所有支持和帮助，感谢他们和我一起渡过了四年美好的大学时光。

最后我还要衷心感谢学校的各位老师、领导以及全班所有同学对我的关心与帮助。

附录 1 ContourDoc 类头文件

```
#include <afxtempl.h>

#include "ContourLine.h"

class CContourDoc : public CDocument

{public:

    CContourDoc();

    DECLARE_DYNCREATE(CContourDoc)

public:

    float ltx;    float lty;    float gridx;    float gridy;    int row;

    int col;    float zoom;    float zmax;    float zmin;    float ** matrix;

    CArray <CContourLine *,CContourLine *> LineSet;

public:

    virtual BOOL OnNewDocument();

    virtual void Serialize(CArchive& ar);

public:

    float distant;

    virtual ~CContourDoc();

}
```


附录 2 CcontourLine 类头文件

```
#include <afxtempl.h>

struct POINTNEW

{

    float x;

    float y;

};

class CContourLine

{

public:

    void DrawLine(CDC * pDC);

    CContourLine();

    virtual ~CContourLine();

    float height;

    CArray<POINTNEW,POINTNEW> m_Points;

};
```

附录 3 读取 CNSDTF-DEM 文件头信息

```
char m[300];    //定义一个字符型数组，用来暂时存放读取到的数据

is.seekg(0, ios::beg);    //定位到文件起始位置

is.getline(m, sizeof(m)-1, '\n');    //读取文件中第一行到 m 中

//判断文件是否为 CNSDTF-DEM 数据格式，若不是则弹出提示

if(lstrcmpi(m, "DataMark:CNSDTF-DEM")!=0)

{AfxMessageBox("请使用 CNSDTF-DEM 版本的文件。"); return;}

is.getline(m, sizeof(m)-1, '\n');is.getline(m, sizeof(m)-1, '\n');

is.getline(m, sizeof(m)-1, '\n');is.getline(m, sizeof(m)-1, '\n');

is.getline(m, sizeof(m)-1, '\n');//读取第六行到 m 中，该行为 X0

sscanf(m, "X0:%f", &pDoc->ltx);//将 X0 的值存入 ltx 变量中

is.getline(m, sizeof(m)-1, '\n');//读取第七行到 m 中，该行为 Y0

sscanf(m, "Y0:%f", &pDoc->lty);//将 Y0 的值存入 lty 变量中

is.getline(m, sizeof(m)-1, '\n');

sscanf(m, "DX:%f", &pDoc->gridx);//将 DX 的值存入 gridx 变量中

is.getline(m, sizeof(m)-1, '\n');

sscanf(m, "DY:%f", &pDoc->gridy);//将 DY 的值存入 gridy 变量中

is.getline(m, sizeof(m)-1, '\n');sscanf(m, "Row:%d", &pDoc->row);

is.getline(m, sizeof(m)-1, '\n');sscanf(m, "Col:%d", &pDoc->col);

is.getline(m, sizeof(m)-1, '\n');is.getline(m, sizeof(m)-1, '\n');

sscanf(m, "Hzoom:%f", &pDoc->zoom);//将 Hzoom 的值存入 zoom 变量中
```

附录 4 读取 DEM 文件中的高程数据程序示例

```
matrix=new float *[row]; int k;

for (k=0;k<pDoc->row;k++)

{pDoc->matrix[k]=new float [pDoc->col];}

int i=0, j=0, temple[10];

do

{is.getline(m, sizeof(m)-1, '\n');

    int n=sscanf(m, "%d%d%d%d%d%d%d%d%d", &temple[0], &temple[1],

        &temple[2], &temple[3], &temple[4], &temple[5], &temple[6], &temple[7],

        &temple[8], &temple[9]);

    if((i==0)&&(j==0))

    {pDoc->zmax=(float) temple[0];pDoc->zmin=(float) temple[0];}

    for (k=0;k<n;k++)

    {pDoc->matrix[i][j]=(float) temple[k];

        pDoc->zmax=(pDoc->zmax>temple[k])?pDoc->zmax:(float) temple[k];

        pDoc->zmin=(pDoc->zmin<temple[k])?pDoc->zmin:(float) temple[k];

        j++;}

    if (j>=pDoc->col) {i++;j=0;}

}while(i<pDoc->row);
```

附录 5 函数 OnFileOpen() 函数体

//DEM 数据读取部分程序添加于此处，见附录 2、3

```
CDistant cd; CString text;
```

```
text.Format(" 读 取 成 功 ， 最 大 高 程 为 %.3f, 最 小 高 程 为  
%.3f", pDoc->zmax/pDoc->zoom, pDoc->zmin/pDoc->zoom);
```

```
cd.m_text=text; CString str, str1;
```

```
str.Format(" 最 大 高 程 为 :%.3f, 最 小 高 程  
为:%.3f", pDoc->zmax/pDoc->zoom, pDoc->zmin/pDoc->zoom);
```

```
m_str=str;
```

```
str1.Format("共%d 行,%d 列数据", pDoc->row, pDoc->col);
```

```
m_str1=str1; UpdateData(FALSE);
```

```
int flag=cd.DoModal();if (flag==IDCANCEL) return;
```

```
if (flag==IDOK) {pDoc->distant=cd.m_height*pDoc->zoom;}
```

```
if (pDoc->distant<=25*pDoc->zoom)
```

```
{MessageBox("等高距输入有误。");return;}
```

```
else
```

```
{
```

```
int i, j;
```

```
flagx=new bool *[pDoc->row];
```

```
flagy=new bool *[pDoc->row-1];
```

```
for (i=0;i<pDoc->row;i++)
```

```
{flagx[i]=new bool [pDoc->col-1];}

for (i=0;i<pDoc->row-1;i++)

{

    flagy[i]=new bool [pDoc->col];

}

for (i=0;i<pDoc->row;i++)

{

    for(j=0;j<pDoc->col-1;j++)

    {

        flagx[i][j]=false;

    }

}

for (i=0;i<pDoc->row-1;i++)

{

    for(j=0;j<pDoc->col;j++)

    {

        flagy[i][j]=false;

    }

}

Gaussfilter();          cacculation();

Invalidate()
```

The algorithm of creating contour lines based on DEM

TANG Lihua, XU Ai-jun, Fang Lu-ming

1. INTRODUCTION

Contour lines refer to isolated curves which connect adjacent points with the equal elevation on the ground. The elevation of a place, where a contour line crosses, is the same. Mapping contour lines is a process which adopts interpolation, a kind of mathematic method, to interpolate a great number of geometrical or physical values which are discrete and regular and transforms points of same values into a map. It represents three dimensional information on the two dimensional surface. It is a powerful tool in the application of analysis of special features of geographic elements. It could generally grasp the feature of special changes in the object you studied and is widely used in the fields of hydrology, environment, climate, planning, etc. There are various DEM-based methods used to map contour lines. At present, there are many algorithm of how contour automatically generated[1-11]. There are two methods commonly used, include method of irregular triangle and method of lattice. The two methods differ each other both in their algorithm and process, each has its own feature. The method of irregular triangle could be better applied in the calculating of irregular discrete sample data. Mapping contour lines with the method of lattice include the following steps: Firstly, adopting linear interpolation directly on the side of lattices to acquire the contour points[5-7]. Then you will find out all the discrete points of a contour line according to a certain law.

Finally, complete the cartography by applying the method of smooth curves. The method of lattice is applicable either in collecting samples on the nodes of lattice or in data calculated from discrete points interpolating into the lattices [5-7,9].

Though there lie some disadvantages in the square lattice model while representing the terrain, but it has already widely used relatively. This is owing to the simple structure of regular square lattice model which could do topographical analysis, description, classification and create a responsive algorithm[1-11]. This essay, on the base of those commonly used, DEM-based algorithm of contour lines, put forth a new GIRD-based algorithm of contour.

2. TRACE OF A SINGLE CONTOUR

If you want to map the contour with a certain elevation, you need to trace all the contour points this particular contour line goes through. It has something to do with the problem of how to browse through points and units of each lattice on the whole and looking for these contour points. The tracing model of a single contour line is illustrated in the following figure 1.

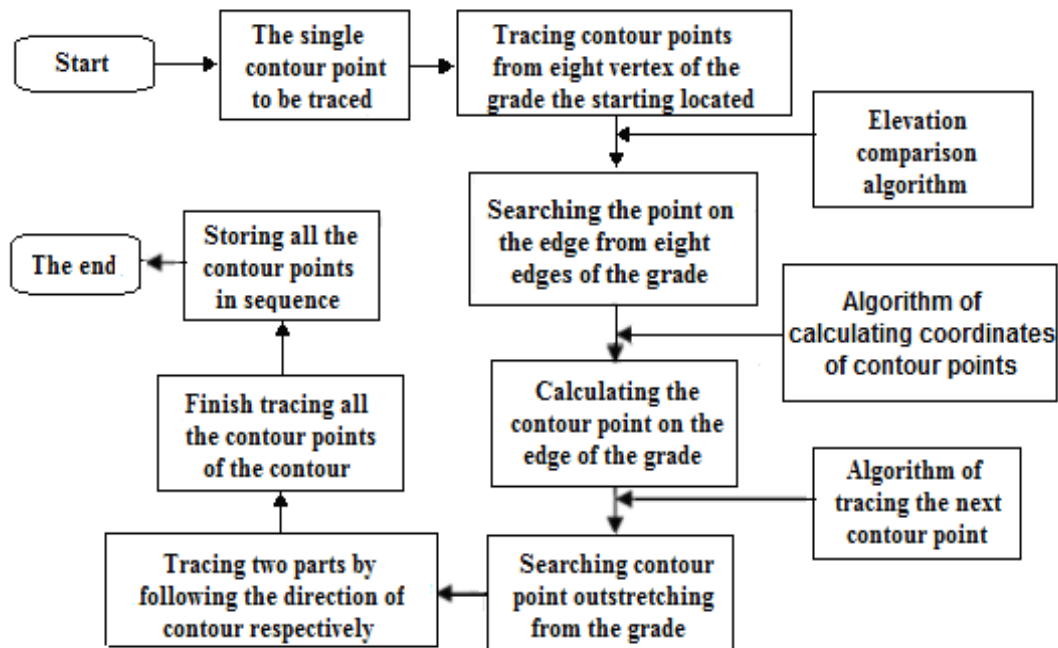


Fig.1 The tracing model of a single contour

2.1 ALGORITHM OF ELEVATION COMPARISON

Let's judge whether the contour lines have crossed the side of a lattice. Now hypothesizing if contour lines have gone through a certain side called AB, the condition is given: $A(x_1, y_1)$, $B(x_2, y_2)$. We choose Z_A , Z_B , elevations of each point A and B, to compare with the elevation of contour point to be traced. If the value of contour point is between Z_A , Z_B , we can conclude that the contour line passes through side AB, otherwise the conclusion is the opposite. It has been shown in the figure 2.

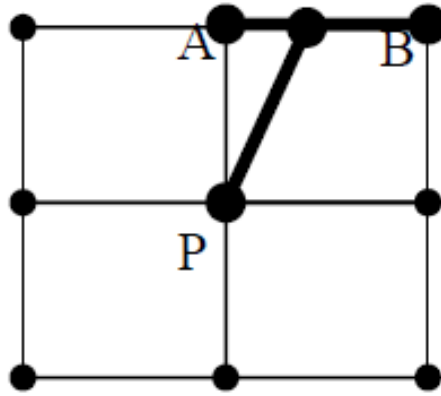


Fig. 2. The elevation comparison model of a single contour

We hypothesize $P(m_tempx, m_tempy)$ as the tracing contour point, elevation of P is $Z_P = m_tempz$. If contour point M , which is to be traced by point P , passes through AB , the elevation of point M must be between elevations of point A and B .

2.2 THE CALCULATION ALGORITHM OF COORDINATES OF CONTOUR POINT

The calculation of contour point coordinates crossed by contour, in other words, calculating the two dimensional coordinate of point M whose elevation is between that of A and B . The algorithm is as followed.

(1) If $Z_A < Z_B$, the elevation of A is lower than that of B : we can use the similar triangles, their proportional sides to get the value. Transformation of flat model is showed in figure 4.

$$h_1 = Z_B - Z_A, \quad h_2 = Z_P - Z_A = m_tempz - Z_A$$

① If $x_1 = x_2$, $\triangle AMN \cong \triangle ABC$, then:

$$\frac{MN}{BC} = \frac{AN}{AC}, \quad \text{namely: } \frac{h_2}{h_1} = \frac{|tempy - y_1 * d|}{d},$$

a. If $y_1 > y_2$, shown in figure 5, then $tempy = y_1 * d + \frac{h_2}{h_1} * d$;

- b. If $y_1 < y_2$, shown in figure 6, then $tempy = y_1 * d - \frac{h_2}{h_1} * d$;
 $tempx = x_1 * d$.

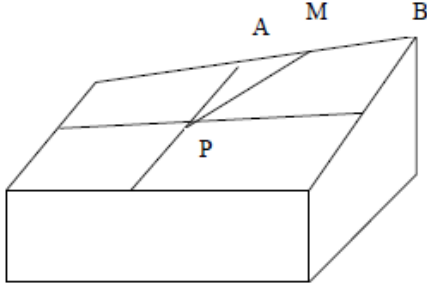


Fig. 3 Line AB of a lattice crossed by contour

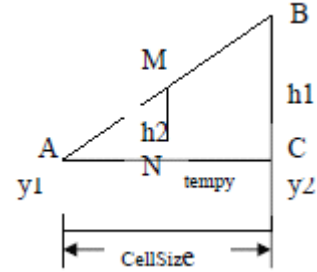


Fig.4 Using the similarity of triangles to calculate the coordinate

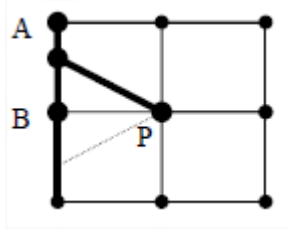


Fig. 5 the case of $y_1 > y_2$

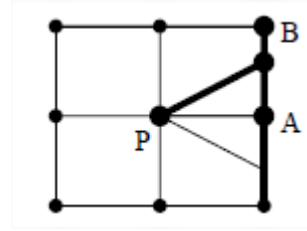


Fig.6 the case of $y_1 < y_2$

- ② If $y_1 = y_2$, $\triangle AMN \cong \triangle ABC$, then

$$\frac{MN}{BC} = \frac{AN}{AC}, \text{ namely: } \frac{h_2}{h_1} = \frac{|tempx - x_1 * d|}{d},$$

- a. If $x_1 > x_2$, shown in figure 5, then $tempx = x_1 * d + \frac{h_2}{h_1} * d$;

- b. If $x_1 < x_2$, shown in figure 6, then $tempx = x_1 * d - \frac{h_2}{h_1} * d$;

$$tempy = y_1 * d$$

(2) If $Z_A > Z_B$, the elevation of A is higher than that of B, in a similar way, we can use the similar triangles, their proportional sides to get the value.

$$h_1 = Z_A - Z_B; \quad h_2 = Z_P - Z_B = m_tempz - Z_B$$

- ① If $x_1 = x_2$, $\triangle AMN \cong \triangle ABC$, then:

$$\frac{MN}{BC} = \frac{AN}{AC}, \text{ namely: } \frac{h2}{h1} = \frac{|tempy - y1 * d|}{d},$$

- a. If $y1 > y2$, then $tempy = y2 * d + \frac{h2}{h1} * d$;
 - b. If $y1 < y2$, then $tempy = y2 * d - \frac{h2}{h1} * d$;
- $tempx = x2 * d$.

② If $y1 = y2$, $\Delta AMN \cong \Delta ABC$, then:

$$\frac{MN}{BC} = \frac{AN}{AC}, \text{ namely: } \frac{h2}{h1} = \frac{|tempx - x2 * d|}{d},$$

- a. If $x1 > x2$, then $tempx = x2 * d + \frac{h2}{h1} * d$;
 - b. If $x1 < x2$, then $tempx = x2 * d - \frac{h2}{h1} * d$;
- $tempy = y2 * d$.

2.3 TRACING ALGORITHM OF NEXT CONTOUR POINT

Two contour points $M1(ddx, ddy)$ and $M2(tempx, tempy)$ have already been traced, they can determine the direction of contour which crosses lattices. $P(x, y)$ is the contour point next to be traced. It is presented in figure 7.

The function of this algorithm is to follow the direction of contour in search of next contour point. Following the direction of $M2M1$ to trace the next contour point $P(x,y)$ and store it in the structure of $Ppoint$.

(1) $i = 0$, and choose the integers of ddx, ddy , this is to say, $dx = (int)ddx, dy = (int)ddy$. The purpose is to left side of this particular point It is illustrated by figure 8. There are only two circumstance while placing the points of the same lattice to the left : The point is either in the transverse lines on the bottom or in the vertical line on the left.

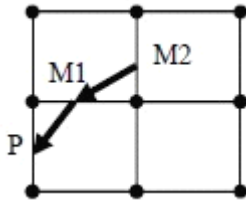


Fig. 7. The direction of contour traced

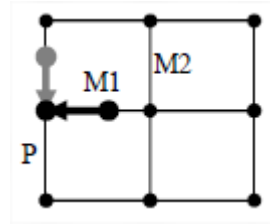


Fig. 8. The point on the left

(2) If $ddx = dx$ and $ddy = dy$, point M1 is squarely on the point of lattice, we can judge whether it passes through the eight corner points of a bigger lattice made up of four smaller ones according the algorithm mentioned above. Writing down the point if it does, otherwise, judging whether it crosses eight sides of the bigger lattice. With the calculation of Computer algorithm, we can get that point and write it down. It is showed by the frame of imaginary lines in figure 9.

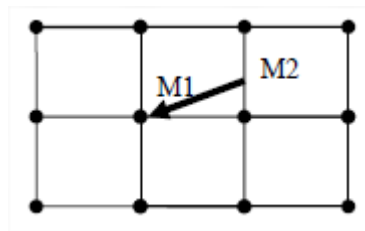


Fig. 9. Tracing the next contour point

(3) If point M1 doesn't locate in the vertex of lattice, but instead on the side of lattice, then the whole lattice would be divided into two parts with the M2 as a boundary. Then we will follow those two directions to trace contour lines.

(4) If $ddy < tempy$, since $M2(tempx, tempy)$ is the starting point, the one to be traced is below the area of boundary marked by tempy while following the direction of M2M1 as shown in figure 10.

① If $ddx \neq dx$ and $ddy = dy$, then point M2 is right on the horizontal line of the lattice where M2M1 located, this is illustrated in figure 11. If that's the case, the contour line following the direction of M2M1 can either goes though other two points A and B of the lower lattice in which M1 situated, or it can passes through the rest of three edge except the one point M1 on it.

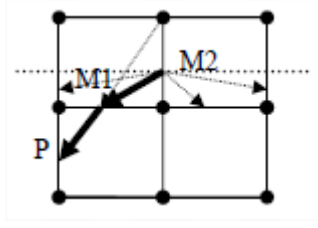


Fig. 10. The M2(tempx,tempy) is the beginning point

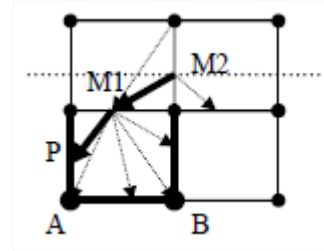


Fig. 11. The track direct while $ddx \neq dx$ and $ddy = dy$

② If $ddx = dx$ and $ddy \neq dy$, the outcome is that point M1 is on the vertical edge of lattice in which M2M1 lies. Under this circumstance, this contour following the direction of M2M1 has two options. It can either cross point A or B if the left lattice where M1 situated or passes through other three edges except the one M1 on it.

(5) If $ddy > tempy$, as M2(tempx,tempy) is the beginning point, we need to follow the direction of M2M1 to trace contour point which is above the boundary area where it is marked by tempy. We could trace contour point from this zone.

① If $ddx \neq dx$ and $ddy = dy$, which means point M1 is on the horizontal edge of lattice in which M2M1 lies (as shown in figure 12). It is showed by exhibition 12. Under this circumstance, this contour following the direction of M2M1 has two options. It can either cross point A or B of the upper lattice where M1 situated or passes through other three edges except the one M1 on it. We can also acquire contour coordinate of the lattice in the same way.

② If $ddx = dx$ and $ddy \neq dy$, which refers point M1 is on the vertical edge of lattice in which M2M1 lies(as shown in figure 13). It is showed by exhibition 13. Under this circumstance, this contour following the direction of M2M1 can either pass through point A or B of the left lattice where M1 situated or passes through other three edges except the one M1 on it. We can also acquire contour coordinate of the lattice in the same way.

(6)Acquired contour coordinate on return.

基于DEM格网提取等高线的算法实现

作者：唐丽华、徐爱军、方陆明

第一章 概述

所谓等高线是指在地面上高程相等的点所连成的闭合曲线。位于同一等高线上的点，高程相等。等高线提取是一个采用插值的数学方法的过程，同一等高线上大量的插值在几何或物理上是离散的分布在地图中。它代表三个二维表面的三维信息。这是一个对特殊的地理信息进行分析的功能强大的工具。它一般可以表现出地貌特殊的变化，广泛的应用于水文、环境、气候、规划等领域。基于DEM模型的等高线绘制方法有很多。目前，自动生成等高线的算法有很多。常用的有两种方法：不规则三角形格网法和规则格网法。这两种方法在算法和程序上都有各自的特色。不规则三角形格网法能够很好地适应不规则的离散采样数据计算。提取等高线的方法一般包括以下步骤：首先，通过直接对格网线性插值获得构成等高线的点；然后，按照一定的规则将这些点依次连接起来；最后，通过在格网上内插点将曲线进行平滑处理。

虽然通过在格网区域上内插点的方法有一些缺点，但是它已经得到了广泛的应用。这是由于它是一个结构简单的常规模型，该模型能够做平面波地形分析、描述、分类并且可以通过构造一个算法来实现。本文在那些常用算法的基础上提出了新的基于DEM格网提取等高线的算法。

第二章 单一等高线的提取算法

如果你想提取某一高程的等高线，你需要找到这条等高线上的所有的点。这里有一个问题，如何遍历所有的格网并且找到这些点？图1表示的是提取单一等高线的模型。

2.1 高程比较算法

我们先来判断等高线是否穿过某一个格网边。假设要判断等高线是否穿过一个特定格网边AB，其中点A、B分别为 $A(x_1, y_1, Z_A)$ 、 $B(x_2, y_2, Z_B)$ 。如果等高线的高程在 Z_A, Z_B 之间，那么，等高线穿过格网边AB，否则不穿过。如图2所示。

假设P (m_tempx, m_tempy, m_tempz) 是等高线上的一点。如果点M是点P在同一等高线上的相邻点，那么M点的高程必须在点A和点B的高程之间。

2.2 等高线上点的坐标计算

计算等高线上点的坐标，即：计算高程值在A、B之间的点M的二维坐标。计算方法如下：

(1) 如果 $Z_A < Z_B$ ，利用相似三角形的原理，如图4 所示：

$$h1 = Z_B - Z_A, \quad h2 = Z_P - Z_A = m_tempz - Z_A$$

①如果 $x1 = x2$ ， $\triangle AMN \cong \triangle ABC$ ，那么：

$$\frac{MN}{BC} = \frac{AN}{AC}, \quad \text{即: } \frac{h2}{h1} = \frac{|tempy - y1 * d|}{d},$$

a. 如果 $y1 > y2$ ，如图 5所示：

$$\text{那么 } tempy = y1 * d + \frac{h2}{h1} * d, \quad tempx = x1 * d .;$$

b. 如果 $y1 < y2$ ，如图 6 所示：

$$\text{那么 } tempy = y1 * d - \frac{h2}{h1} * d, \quad tempx = x1 * d .;$$

② If $y1 = y2$ ， $\triangle AMN \cong \triangle ABC$ ，那么

$$\frac{MN}{BC} = \frac{AN}{AC}, \quad \text{即: } \frac{h2}{h1} = \frac{|tempx - x1 * d|}{d},$$

a. 如果 $x1 > x2$ ，如图 5 所示：

$$\text{那么 } tempx = x1 * d + \frac{h2}{h1} * d, \quad tempy = y1 * d .;$$

b. 如果 $x1 < x2$ ，如图 6 所示：

$$\text{那么 } tempx = x1 * d - \frac{h2}{h1} * d, \quad tempy = y1 * d .;$$

(2)如果 $Z_A < Z_B$, 用类似的方法可以计算

$$h1 = Z_A - Z_B; \quad h2 = Z_P - Z_B = m_tempz - Z_B$$

① 如果 $x1 = x2$, $\Delta AMN \cong \Delta ABC$, 那么:

$$\frac{MN}{BC} = \frac{AN}{AC}, \text{ 即: } \frac{h2}{h1} = \frac{|tempy - y1 * d|}{d},$$

$$\text{a. 如果 } y1 > y2, \text{ 那么 } tempy = y2 * d + \frac{h2}{h1} * d;$$

$$\text{b. 如果 } y1 < y2, \text{ 那么 } tempy = y2 * d - \frac{h2}{h1} * d;$$

$$tempx = x2 * d$$

② 如果 $y1 = y2$, $\Delta AMN \cong \Delta ABC$, 那么:

$$\frac{MN}{BC} = \frac{AN}{AC}, \text{ 即: } \frac{h2}{h1} = \frac{|tempx - x2 * d|}{d},$$

$$\text{a. 如果 } x1 > x2, \text{ 那么 } tempx = x2 * d + \frac{h2}{h1} * d;$$

$$\text{b. 如果 } x1 < x2, \text{ 那么 } tempx = x2 * d - \frac{h2}{h1} * d;$$

$$tempy = y2 * d$$

2.3 查找下一内插点

假设已经找到两个内插点 M1 (ddx, ddy) 和 M2 (tempx, tempy) , 那么等高线的方向就确定了。P(x, y) 是下一个内插点, 如图 7 所示:

该算法的功能是按照等高线的方向进行下一内插点的查找。以下的方向, M2M1 追踪下轮廓点 P(x, y), 并将其存储在如此的结构。按照 M2M1 的方向进行下一内插点的查找 P(x, y), 并将其存储到数据结构 Ppoint 中。

(1) $i=0$, 选择 ddx, ddy 为整数, 即: $dx=(int)ddx$, $dy=(int)ddy$ 。这样做的目的是要避免如图 8 所示的错误。这时有两种处理方法: 在水平线的左边或在垂直线的下面。

(2) 如果 $ddx=dx$, $ddy=dy$, 那么点 M1 不需要改变。按照上面的方法, 我们能够判断等高线是否穿过有 4 个小方格组成的 8 个点的大方格。如果穿过, 记录下来这一点。否则, 判断是否穿过更大的方格。使用计算机算法来计算, 我们可以得到内插点并且记录下来。如图 9 所示:

(3) 如果点 M1 不在格网的顶点上, 而是在格网的边上, 那么, 整个格网被一 M2M1 为边界分成两部分。然后我们将按照这两个方向进行等高线的查找。

(4) 如果 $ddy < tempy$, 由于 M2($temp_x, temp_y$) 是起始点, 按照 M2M1 的方向, 下一个内插点应该在 $temp_y$ 的下方, 如图 10 所示。

① 如果 $ddx \neq dx$ 并且 $ddy = dy$, 那么点 M2 正好在 M2M1 所在格网的水平边上, 如图 11 所示。如果是这样的话, 按照 M2M1 的方向, 等高线将穿过 A、B 或其它的三边。

② 如果 $ddx = dx$ 并且 $ddy \neq dy$, 那么 M1 在格网的左边界上。在这种情况下, 等高线的方向 M2M1 有两种选择, 穿过 A、B 或者是穿过其它三边。

(5) 如果 $ddy > tempy$, 由于 M2($temp_x, temp_y$) 是起始点, 我们将按照 M2M1 的方向向上查找内插点。我们可以在这一区域内查找到内插点。

① 如果 $ddx \neq dx$ 并且 $ddy = dy$, 这就是说, 点 M1 在格网水平边上 (如图 12 所示)。在此情况下, 等高线的方向 M2M1 有两种选择, 它可以穿过点 A、B 或通过其它三条格网边。通过同样的方法可以获得内插点的坐标。

② 如果 $ddx = dx$ 并且 $ddy \neq dy$, 这时 M1 在垂直边上 (如图 13 所示)。在此情况下, 等高线的方向 M2M1 可以穿过点 A、B 或穿过其它三条边。通过同样的方法可以获得内插点的坐标。

(6) 返回。