CAD矢量曲线抽稀的算法和实现

吴铭杰

(福建省测绘院,福建 福州 350003)

摘 要:主要介绍对于AutoCAD的图形曲线进行抽稀处理的一种方法,并详细阐述了该使用程序快捷有效的实现算法。

关键词:Douglas-Peucker算法;曲线抽稀;矢量压缩;堆栈;AutoLisp
中图分类号:TP311 **文献标识码**:B **文章编号**:1672-5867(2010)03-0207-02

Simplification Algorithms and Realization of CAD Vector Curve

WU Ming-jie
(Fujian Surveying and Mapping Institute, Fuzhou 350003, China)

Abstract: This paper mainly describes a way to simplify AutoCAD vector curve, and elaborates an easy and effective algorithm using the program.

Key words: Douglas-Peucker algorithm; curve simplification; vector compression; stack; AutoLisp

0 引 言

曲线抽稀处理,归根结底就是曲线矢量数据压缩,即曲线特征点的提取,要求通过计算,将组成曲线的点数据集合按照规定的精度要求提取子集,使子集的曲线在表现上近似于原始曲线,并使子集相对于原集合尽量地小,亦即在数据量上尽可能地少。

随着地理信息系统(GIS)的不断发展,其在各个领域应用的不断深入,各种形式的地图数据需要发布与传输,此时,作为图形的主要组成——曲线的矢量压缩也日益显现其重要的作用。因为较少的数据量就意味着较快的传输效率,也就意味着更加流畅的应用效果。

在测绘工作中,无论是使用矢量采集、数字测图,还是缩编的方式生产图形产品,都会遇到曲线节点过于密集而导致数据量偏大的问题。这时,就要求生产单位对现有的曲线进行必要的抽稀处理。AutoCAD图形文件(包含DWG,DXF)是当前国内测绘产品所采用的一种通用的文件格式,它或作为测绘的最终产品提交用户,或作为中间交换数据提供后续处理,研究并提出针对CAD数据的矢量曲线抽稀的算法及其实现对于测绘生产有着现实意义。

本文主要介绍基于经典的Douglas-Peucker矢量压缩算法的CAD矢量曲线抽稀的处理流程,提出一种简单快捷的实现方法。最后,本文将给出使用AutoCAD内置的AutoLisp代码实现DWG矢量曲线的抽稀操作的示例代码。

1 Douglas-Peucker算法的基本思路

矢量数据压缩的目的是删除冗余数据,减少数据的存贮量,节省存贮空间,加快后继处理的速度。矢量压缩的算法主要有Douglas-Peucker算法、垂距法、光栏法等,笔者在工作中之所以选择Douglas-Peucker算法,是由于当前的实际工作中,大多数情况下Douglas-Peucker算法的压缩效果较好,并且该算法具有旋转、平移的不变性,在给定限差之后,压缩结果不会改变,可以很好地满足作业要求。

Douglas-Peucker算法的基本思路如下:

1)对曲线的首末点虚连一条直线(对于闭曲线应该选择曲线的最左侧与最右侧端点作为首末点)。

2)找出曲线上与连线距离最大的顶点,判断其到连线的距离是否大于给定阈值D,若是,则保留该点,并以该点为界,将曲线分为前后两个部分重复上述操作;否则,舍去两端点间所有的中间点。

3)最后将所有的保留点组成新的点集,作为压缩结果。

图1中显示了Douglas-Peucker算法数据压缩的简

收稿日期:2009-09-20

作者简介:吴铭杰(1976-),男,福建福州人,工程师,学士,1999年毕业于武汉测绘科技大学摄影测量与遥感专业,主要从事4D产品以及相关应用的开发与生产工作。

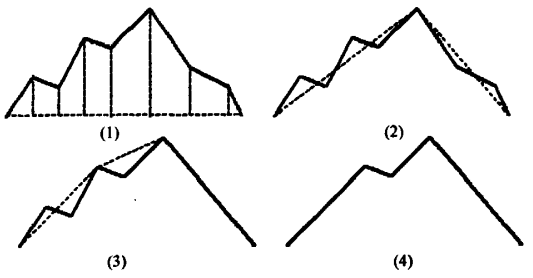

图 1 Douglas – Peucker 算法示意图

Fig. 1 The principle of Douglas – Peucker algorithm

单过程,该算法很好地保留了曲线的特征点,通过对限差值的合理设定,可以很好地应用于测绘地形图中诸如等高线、地类界等线状地物地貌信息的压缩。

2 Douglas – Peucker 算法的非递归实现

从 Douglas – Peucker 算法的基本思路可以看出该算法实际上是一个递归实现的过程,而笔者在工作上使用 C ++ 进行编程的时候也是使用递归的方式予以实现的。

在这里,本文通过合理地使用堆栈结构,给出另外一种简单可靠的、非递归的实现方式,可以方便地使用编程语言予以实现。

程序实现的基本步骤:

1)设定抽稀阈值 D;

2)提取曲线的顶点组成点集,并按曲线方向顺序排列 $\{Pt\}_{0\sim n-1}$(总计 n 个点,首点序号为 0,末点为 $n-1$);

3)初始化两个堆栈 A,B,用于存放点序编号,其中堆栈 A 初始化压入 0,堆栈 B 初始化压入 $n-1$;

4)由堆栈 A 读取栈顶元素 i 并搜索对应点 Pt_i 赋予 $p1$,由堆栈 B 读取栈顶元素 j 并搜索对应点 Pt_j 赋予 $p2$,将 $p1$,$p2$ 作为虚连直线 $P1P2$ 的顶点;

5)计算 $\{Pt\}_{i+1\sim j-1}$ 中各点到直线 $P1P2$ 的距离,并比较得到最大值 $D\max$ 以及对应的点序号 m;

6)比较 $D\max$ 与抽稀阈值 D 大小,如果 $D\max > D$,将 m 压入堆栈 B 中,否则将 j 压入堆栈 A 并由堆栈 B 中删除栈顶有元素 j;

7)返回第 4)步继续执行,直至堆栈 B 为空;

8)按堆栈 A 中存储序号由点集取出点坐标反向组成新的点集就是所要提取的抽稀后的特征点。

用上述实现方法进行编程时可以很方便地使用表处理函数进行编程,从而大大降低了程序编写的难度,同时效率也明显优于递归实现的方法。

3 AutoLISP 代码

下面给出使用 AutoLisp 实现的主要代码:

```
;;;堆栈初始化
(setq Stack1 (append (list 0) Stack1) Stack2 (append (list (1 - num)) Stack2 ))
(while (not (null Stack2))
    (setq i (car Stack1) j (car Stack2))
    (setq pt1 (nth i pts) pt2 (nth j pts))
    (setq k (1 + i) distmax 0 )
    (while ( < k j)
        (setq pt3 (nth k pts) dist (wmj:Object:Dist pt1 pt2 pt3))
        (if ( > dist distmax)
            (setq n k distmax dist)
        );_if
        (setq k (1 + k))
    );_while
    (if ( > distmax distlimit)
        (setq Stack2 (append (list n) Stack2))
        (setq Stack1 (append (list (car Stack2)) Stack1) Stack2 (cdr Stack2))
    );_if
);_while
```

4 结束语

本文介绍的矢量曲线压缩算法和实现代码,可以很好地在 AutoCAD 软件中完成对指定曲线的抽稀运算,能够满足测绘作业中的 CAD 图形编辑的需要,在应用于生产工作时,取得了很好的实际效果(如图 2 所示)。不足之处在于 AutoLISP 语言的代码执行效率不高,在处理数据量较大的文件(如等高线等线状信息比较丰富的中小比例尺地形图等)时,耗时较多。由于算法的实现设计较为简洁、合理,可以很方便地使用运行效率更高的基于 Visual C ++ 或者. NET 的 ObjectARX 编程来改写文中的代码,从而提高代码执行的效率。

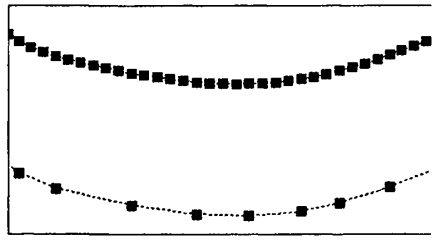

图 2 抽稀前后曲线节点比较

Fig. 2 The comparison of the nodes before and after simplification

参考文献:

[1] 吴立新. 地理信息系统的原理和算法[M]. 北京:科学出版社,2003.

[2] 郭仁忠. 空间分析[M]. 北京:高等教育出版社,2001.

[3] 赵景亮,李志刚. AutoCAD 2004 与 AutoLISP 二次开发技术[M]. 北京:清华大学出版社,2004.

[编辑:宋丽茹]