一种任意多边形裁剪快速算法

杜月云[1] 周子平[1] 张云龙[2]
[1](商丘职业技术学院计算机系 河南 商丘 476000)
[2](空军第一航空学院基础部 河南 信阳 464000)

摘 要 提出一种用 VC ++ 语言实现的多边形裁剪快速算法。与以往的算法相比,算法中不仅多边形可以是任意的,而且在求交、并和差的过程中用符号判断代替耗时的乘法运算,采用预处理方法等技术来减少程序的遍历次数,从而加快了计算速度。算法中用 MFC 的 CObList 类和 CArray 类的对象来动态存储数据,大大节约了内存开销,是一种高效的算法。另外,算法编制的软件,已得到了有效的应用。

关键词 多边形裁剪 线裁剪 VC ++ 语言 MFC 类库

A QUICK ALGORITHM FOR DISCRETIONARY POLYGON CLIPPING

Du Yueyun[1] Zhou Ziping[1] Zhang Yunlong[2]
[1](*Department of Computer, Shangqiu Vocational and Technical College, Shangqiu 476000, Henan, China*)
[2](*Fundamental Department, The First Aaronautical Institute of Air Force, Xinyang 464000, Henan, China*)

Abstract A quick algorithm for polygon clipping is proposed in this paper, and realized with the Visual C ++ language. Compared with previous methods, in this algorithm the polygon is discretional; and judgment on sign is used to replace the time-consuming multiplying calculations for intersecting, union, and difference of the line segments. The techniques such as pretreatment are adopted for reducing the times of traversal of the program to accelerate calculation speed. The data are stored dynamically with the objects of classes of CObList and CArray of MFC, which saves a great deal of memory. So the algorithm is really of high efficiency. Moreover, the application of software programmed with this algorithm has also gotten effective use.

Keywords Polygon clipping Line clipping Visual C ++ language MFC class libraries

0 引 言

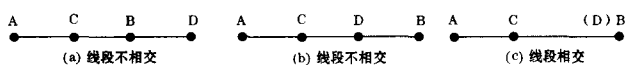
在图形系统中,二维裁剪是最基础、最常用的操作之一,其典型的应用包括对图形进行消隐处理、对各种三维图形进行排料、整形处理等。在图形消陷、缩放、模式识别、导线和元件布局、线性规划[1]以及土木工程 CAD 中的梁与柱、墙与板、墙与柱,特别是各类基础工程量的计算等领域中都要广泛应用到多边形的裁剪。目前,对裁剪算法的研究主要集中在裁剪直线和多边形两方面。在实用中,多边形裁剪与线剪裁则具有更高的使用频率,因此它是裁剪算法研究的主要课题[2]。多边形愈复杂其裁剪算法就愈难以实现,文献[2]中综述了现有的解决方案,但这些方案或是局限于某一类多边形,或者时间复杂性和空间复杂性都较高。

本文对两个任意多边形相交时可能出现的位置关系做了一种新的、更严格的定义,用 VC ++ 语言实现了两个任意多边形的交、并和差算法。将该软件应用于实际教学与工作中收到了很好的效果。其中的裁剪多边形和被裁剪多边形都可以是凸多边形、凹多边形,也可以是带内环的多边形。本算法具有数据结构简单、占用空间少、计算速度快的特点,不需要对两多边形的边重合或两多边形在顶点处相交的情况作特殊处理,优于文献[2]中所述的算法。

1 基本概念

多边形裁剪是指用一个窗口(又称为裁剪多边形)裁剪掉实体多边形(被裁剪多边形)位于这个窗口之外的部分。

我们定义多边形线段相交,是指来自两个多边形的线段重合时,如果至少有一个相同的顶点,则称这两个线段相交,而如果没有重合的顶点,则认为其不相交。如图 1(a)、1(b),视为线段 AB 与线段 CD 不相交;如图 1(c),视为线段 AB 与线段 CD 相交,且交线为 CD。



(a) 线段不相交 (b) 线段不相交 (c) 线段相交

图 1

2 算法的数据结构

利用 MFC 类中的 CobList 类以及 CArray 类产生我们需要的动态数组对象,可以方便地实现多边形顶点及交线的数据存储,

收稿日期:2007 - 11 - 12。杜月云,讲师,主研领域:计算机工程。

这一动态数组有别于其他高级语言中的数组，它不需要预先在内存中开辟一块足够大的内存，而是根据输入多边形的顶点数以及所求得的交线数，自动分配内存，避免了不必要的内存开销。同时 MFC 类封装了对上述动态数组中数据元素的遍历、获取、插入、删除等函数，调用简单、快捷。这一数据结构与 Greiner-Hormann 算法[2] 中的双向链表结构相比，大大节省了存储空间，同时也进一步降低了数据结构及其算法的复杂性。

每个多边形由一个动态数组来表示，数组按多边形顶点的输入顺序存储其顶点数据，数组的每一个元素对应于多边形的一个顶点。数组的最后一个元素与第 1 个元素相同，从而构成一个封闭的多边形，而每相邻的两个元素的一个组合对应于多边形的一条边，而有一个共同元素的一对组合对应于多边形的两条相邻边，公共元素是两条相邻边的相交顶点。

举例：图 2 所示的多边形 S 和多边形 C 的顶点，分别用动态数组 S 和 C 来定义：

```
CArray < Vertex, Vertex > S, C;
S = { S1, S2, S3, S4, S1 };
C = { C1, C2, C3, C4, C5, C6, C1 };
```

其中，Vertex 是自定义的一种结构。

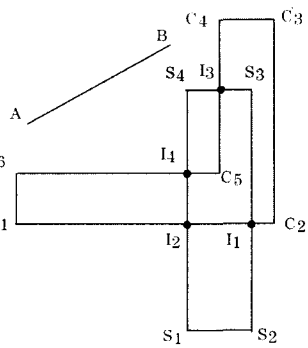


图 2 包围盒示例

同时定义交点数组、临时数组和交线的指针数组为：

```
CArray < MyJD, MyJD > m_JD;
CArray < MyPoint, MyPoint > m_tempP;
CTypedPtrList < CObList, CDrawBase * > CoLineList;
```

其中，MyJD、MyPoint 是自定义的另一种结构。

3 算法描述

限于篇幅，下面仅介绍求交算法，可分为 4 步，稍做修改即可用于多边形求并以及求差等。

第一步 求包围盒。即求多边形 S 的包围盒，并剔除多边形 C 中不在该包围盒内部的边。剔除后记为 C'；再用多边形 C 的包围盒（记为包围盒 2）剔除多边形 S 不在包围盒 2 中的边，剔除后记为 S'，这样可以避免不必要的求交运算。

第二步 对 S' 的各边循环，将各边与 C' 的各边求交，对 S' 的任意一边 S_iS_{i+1} 其具体操作包括：(1) 清空临时数组 *m_tempP*；(2) 调用函数 *PointRgn*( ) 判断边 S_iS_{i+1} 的起点 S_i 是否位于多边形 C 内：若位于 C 内，则将点 S_i 加入临时数组 *m_tempP* 中；(3) 依次取出 C' 的两个相邻顶点，即 C' 的边 C_iC_{i+1}，进行求交前的预处理（见后）；(4) 若判定两边相交（线段与线段交点不是虚交点），则调用子函数求交点，并分别将求得的交点加入到临时数组 *m_tempP* 和交点数组 *m_JD* 中。直至遍历了 C' 中的所有边；(5) 调用子函数对临时数组 *m_tempP* 中的各点按边 S_iS_{i+1} 的方向排序，并依次判断相邻两个交点的中点是否位于多边形 C 内：若是，则将以这两个交点为起点和终点的边加入交线指针数组 CoLineList 中。至此，两多边形的所有交点和多边形 S 在多边形 C 内的部分都已经求出。

第三步 组织交点，构造多边形 C 在多边形 S 内的边。具体操作是对 C' 内的边循环，对每一边 C_iC_{i+1}：(1) 清空临时数组 *m_tempP*；(2) 若 C_i 位于多边形 S 内，则将其加入临时数组 *m_tempP*；(3) 判断 *m_JD* 中的所有各点的 Start 和 End 是否与 C_iC_{i+1} 的起点和终点相同，若相同，则将其加入临时数组 *m_tempP* 中；(4) 若 C_{i+1} 位于多边形 S 内，则将其加入临时数组 *m_tempP*；(5) 调用子函数对临时数组 *m_tempP* 中的各点按边 C_iC_{i+1} 的方向排序，并依次判断相邻两个交点的中点是否位于多边形 C 内：若是，则将以这两个交点为起点和终点的边加入交线指针数组 CoLineList 中；

第四步 输出结果多边形。对交线指针数组循环，直至所有的线段均被输出。当多边形是有内环的多边形时，仅需将带内环的多边形的内环与外环的顶点分别存储在两个 CArray 动态数组中，并使得内外环的顶点的输入顺序相反，即可采用上述算法将内环、外环多边形分别与被另一多边形求交。

其中第二步求交前的预处理算法则可以大大提高运算的效率。主要包括以下两点：

(1) 运用包围盒进行预处理

求出多边形 S 的包围盒，再用包围盒过滤多边形 C 位于包围盒外的边以减少不必要的求交和判断。以图 2 所示为例，先用多边形 S 的包围盒去剔除多边形 C 的边 C_6C_1、C_2C_3 和 C_3C_4，接着用多边形 C 剩余边的包围盒剔除多边形 S 不在该包围盒内的边。从而在计算过程中仅需要判断，或者求解多边形 S 与多边形 C 剩余边的交点，因而可以大大减少计算量。又因为做一次乘法运算远比加减法（比较运算）耗时，所以采用这一预处理大大减少了算法的复杂度。

(2) 判定预处理子过程

设被裁剪线段 Q_jQ_{j+1} 的直线标准方程为：

$$F(x,y) = Ax + By + C = 0 \tag{1}$$

其中 $A = y_{j+1} - y_j, B = x_j - x_{j+1}, C = -(Ax_j + By_j)$。

显然，对于 Q_jQ_{j+1} 上的点，有 $F(x,y) = 0$；对于其上方的点有 $F(x,y) > 0$；而对于其下方的点有 $F(x,y) < 0$。

当多边形 P 某条边 $P_iP_{i+1}(i = 0,1,\cdots,n)$ 的两个端点在线段 Q_jQ_{j+1} 的同一侧时，线段 Q_jQ_{j+1} 与边 P_iP_{i+1} 不相交。把符合这种情况的边 P_iP_{i+1} 的两个端点坐标分别代入 $F(x,y) = Ax + By + c$，则有

$$F(P_{i,x},P_{i,y}) \cdot F(P_{(i+1),x},P_{(i+1),y}) > 0 \tag{2}$$

为避免进行乘法运算，上述不等式的判断可转化为：

$$\text{sign}(F_i) - \text{sign}(F_{i+1}) = 0 \quad (\text{sign}(F_i) \neq 0 \text{ 且 } \text{sign}(F_{i+1}) \neq 0) \tag{3}$$

其中函数 sign(x) 为符号函数。

而当 sign(F_i) = 0 或 sign(F_{i+1}) = 0 时，则表示点 P_i 或 P_{i+1} 在 Q_jQ_{j+1} 上或其延长线上。这时需要根据具体情况分别处理，如图 3 所示，程序中都给予了充分的考虑。

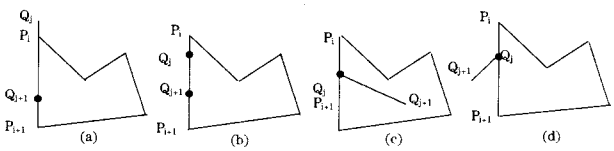


图 3 P_iP_{i+1} 与 Q_jQ_{j+1} 的位置关系

程序中，用线段 Q_jQ_{j+1} 按顺时针逐边裁剪窗口多边形 P（顶点分别是 P_0、P_1，…，P_n）。对于多边 P_iP_{i+1}，若满足式(3) 时，则可判定线段 Q_jQ_{j+1} 与边 P_iP_{i+1} 无有效交点，不用再求交，但在转去判断下一条边界 $P_{i+1}P_{i+2}$ 前，还要做如下的判断：

设边 P_iP_{i+1} 的直线标准方程为：

$$G_i(x,y) = A_i x + B_i y + C = 0$$

其中 $A_i = P_{i,y} - P_{(i+1),y}, B_i = P_{(i+1),x} - P_{i,x}, C = -(A_i P_{i,x} + B_i P_{i,y})$。

对边 P_iP_{i+1} 上的点,有 $G_i(x,y) = 0$;对 P_iP_{i+1} 上方的点有 $G_i(x,y) > 0$;而对 P_iP_{i+1} 下方的点则有 $G_i(x,y) < 0$。当 $G_i(x_j, y_j) \cdot G_{(i+1)}(x_{j+1}, y_{j+1}) > 0$ 时,即:

$$\text{sign}(G_i) - \text{sign}(G_{i+1}) = 0 \quad (\text{sign}(G_i) \neq 0 \text{ 且 } \text{sign}(G_{i+1}) \neq 0) \tag{4}$$

时表示线段 Q_jQ_{j+1} 的两端点在多边形边 P_iP_{i+1} 的同一侧,断定线段 Q_jQ_{j+1} 与边 P_iP_{i+1} 没有有效交点。而当 $\text{sign}(G_i) = 0$ 或 $\text{sign}(G_{i+1}) = 0$ 时,则表示点 Q_j 或 Q_{j+1} 在 P_iP_{i+1} 上或其延长线上,程序中都有充分的考虑。

判断时,首先按式(3)进行判断,若不满足,则继续用式(4)来判断。若两式均判定线段 Q_jQ_{j+1} 与边 P_iP_{i+1} 没有有效交点,则进行求交运算,之后进入下一条边的判断。

图 4 给出了本算法的用户界面及部分多边形裁剪举例。

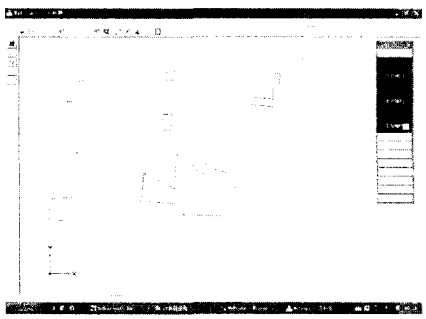


图 4　用户界面及多边形裁剪举例

4　结束语

本文较系统地描述了一种新的多边形裁剪算法,根据 MFC 类的特性,用动态数组实现了对多边形顶点、交点以及交线的数据存储,大大节约了系统开销。相比以往的算法,本算法减少了对多边形顶点数据的遍历次数,通过求交前的预处理,用简单的判断减少了复杂求交的次数,从而降低了算法复杂性。采用本算法成功实现了三维 CAD 及图形算量软件中的各类扣减,在实际工作得到了有效的应用。

参　考　文　献

[1] 周培德. 计算几何[M]. 北京:清华大学出版社,2000.

[2] 刘勇奎,高云,黄有群. 一个有效的多边形裁剪算法[J]. 软件学报,2003,14(4):845 - 856.

(上接第 237 页)

强边缘,而图 5(b) 中的频谱由中心向垂直方向扇形扩展,说明其区域图像对应有连续排列的水平强边缘和少许垂直边缘,无周期排列的边缘;从 $S(r)$ 曲线中可见,图 5(a) 中在 $r = 0$ 和 1 处有两个峰值,表明其能量集中在原点附近,说明其强边缘成周期排列,而图 5(b) 的 $S(r)$ 曲线的峰值连续出现在 $r = 3 \sim 90$ 范围,表明其能量在原点最弱,从原点向外逐渐增强,说明其强边缘无很强的周期性。

因为仪表的刻度区域是由顺序圆弧排列的短直线组成,具有一定的周期性。并可以准确推断出,图 4(a) 所表示的区域一定是表盘的刻度线所在位置。

2　实验结果

由上述的纹理分析方法,可以得到图 4(a) 连通区域 1 和图 4(b) 连通区域 2 的统计数据,如表 2 所示。

表 2　连通区域 1 和连通区域 2 的纹理统计分析结果

连通区域	平均亮度	平均对比度	平滑度	第 3 阶矩	一致性	熵
1	10.59	50.87	0.03	9.30	0.92	0.24
2	4.98	35.29	0.01	4.69	0.96	0.13

从表 2 的数据可以判断出这两个区域是表盘刻度区域。再根据上面的频谱分析方法,由图 5(a) 和(b) 的 $S(r)$、$S(\theta)$ 曲线和频谱,可以进一步确定图 4(a) 连通区域 1 和图 4(b) 连通区域 2 一定是表盘刻度区域,而其它区域一定不是刻度区域。然后用文献[9] 的仪表指针检测方法定位出指针位置,二者综合起来就可以得到只包括指针和刻度的读数区域。实验结果如图 6 所示。由图 6 可见,定位效果非常好。因此以后就可以只对指针和刻度进行处理,排除其它干扰,准确识别指针读数。

图 6　指针和刻度的读数区域定位

3　结　论

本文提出的基于形态学和纹理分析的仪表读数区域定位方法充分利用了方形指针式仪表指针与边框相连接和刻度线排列有一定的周期性的特点,定位准确,而且可以排除仪表盘上刻度区域以外的大量仪表参数字符、指针、镜面弧形反射区等的影响,具有较强的鲁棒性。为以后读数自动识别打下了坚实的基础。不过该算法是针对方形指针式仪表来设计的,对圆形仪表需要重新设计算法。因此我们将继续完善仪表刻度识别的方法。

参　考　文　献

[1] 何智杰,陈彬,金连文. 高精度指针仪表自动读数识别方法[J]. 计算机辅助工程,2006,9:9 - 13.

[2] 陈彬,金连文. 一种仪表指针位置检测的中心投影法[J]. 计算机应用研究,2005(1):246 - 248.

[3] 戴亚文,王三武,李金龙. 基于图像处理技术复杂仪表的自动识别[J]. 计量技术,2003(3):33 - 35.

[4] 李盛阳,叶梧,冯穗力,等. 基于 DSP 的仪表监控系统[J]. 电视技术,2004(10):89 - 91.

[5] 林金龙,石青云. 用点 HOUGH 变换实现圆检测的方法[J]. 计算机工程,2003,29(11):17 - 18.

[6] Rafael C Gonzalez. 数字图像处理[M]. 北京:电子工业出版社,2003:273 - 280.

[7] William K Pratt. 数字图像处理[M]. 北京:机械工业出版社,2005:300 - 343.

[8] 吾健辉. 基于数字图像处理技术的针式仪表读数识别[J]. 大众科技,2005(6):68 - 69.

[9] 张艳玲,汪仁煌. 基于仪表先验知识和梯度方向滤波的指针位置检测方法[C]. 第五届全国信息获取与处理学术会议,2007,8.