基于连续分区与串联编码的线段裁剪新算法

朱亚臣 谭建荣 陆国栋 冯毅雄
(浙江大学 CAD&CG 国家重点实验室, 杭州 310027)

摘 要 提出连续分区裁剪的新思想,按不同的方式多次将平面分成不同的区域,从而简单而快速地舍弃窗外线段,避免没有必要的求交运算。2 次分区后提出串联编码技术,将 2 次编码技术中的两次编码串联起来,继承了第 1 次编码所做的工作,节省了很多时间;在此基础上,3 次分区中提出点对称技术,并将其应用于多次分区,取得了较好的效果。点对称技术所划分平面区域边界的斜率为 0°、±45°或无穷大,适合于程序的实现,且与 2 次编码技术相比,可以舍弃更多的窗外线段,避免更多的求交运算。算法实现清楚地表明该算法能明显提高线段裁剪效率。

关键词 线段裁剪 连续分区 串联编码 点对称技术
中图法分类号: TP391 **文献标识码**: A **文章编号**: 1006-8961(2007)04-0732-08

Novel Algorithm for Line Clipping Based on Continuous Zoning and Series Coding Technique

ZHU Ya-chen, TAN Jian-rong, LU Guo-dong, FENG Yi-xiong
(*State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou* 310027)

Abstract Continuous zoning, a novel clipping thought, is put forward in this paper. The plane is divided into many different areas in different way continuously, which can abandon the lines outside the window easily and quickly, and avoid unnecessary computing the point of intersection. After secondary zoning, we implemented the series coding technique to save time and to establish connection between both codes. Then the point symmetry technique which is applied to multiple zoning is carried out in cubic zoning. The slope of the area boundary in cubic zoning is 0°、±45° or infinity, which is suitable for the procedure realization. Comparing to secondary coding technique, point symmetry technique can avoid redundant operation of computing the point of intersection. The procedure realization clearly expresses that the algorithm in this paper can improve the line clipping efficiency.

Keywords line clipping, continuous zoning, series coding, point symmetry technique

1 引 言

线段裁剪作为计算机图形学中的基本问题,已经有了许多有效的算法[1~5]。在实际应用中,由于线段裁剪的基础性及操作的频繁性,裁剪效率直接影响整个图形系统的效率。经典的线段裁剪算法主要有编码裁剪算法、中点分割算法和梁-Barskey 算法[6]。其中编码裁剪算法最为简单、方便,应用也最为广泛。

实际上,线段裁剪算法在本质上可以归纳为两个方面:一是尽量避免求交,二是加快求交速度。在编码裁剪算法基础上进一步拓展的 2 次编码技术,以其特有的优点——尽快和尽可能多地舍弃窗外线段,从而尽量避免求交,成为近年来诸多线段裁剪算法的首选技术[7,8]。

基金项目:国家自然科学基金项目(50505044,60573175);浙江省教育厅科研项目(20050944);国家"973"计划基金项目(2002CB312106,2004CB719400)
收稿日期:2005-10-13;**改回日期**:2006-02-23

第一作者简介:朱亚臣(1981 ~),男。浙江大学机械设计及理论专业博士研究生。主要从事 CAD、CG 方面的研究。E-mail:zyc_zju@163.com

2 次编码通过对线段端点的预编码,在编码裁剪算法判断的基础上,又可简单舍弃一部分窗外线段,对于剩下的窗外线段和所有与窗口相交的线段,则需要求交流程来求出该线段与窗口的交点,并判断交点是否有效。可见,2 次编码与编码裁剪算法相比,能够舍弃更多的窗外线段,提高了线段裁剪效率。但是它没有继承编码裁剪算法所做的工作,因为它使先前的编码不再有效;其次,它没有摆脱编码裁剪算法思想的束缚,还只是能舍去完全在新窗口外同一侧的线段,对于那些不符合这一条件而又在窗口外的线段,2 次编码却"束手无策"。

本文提出一种新的基于连续分区与串联编码的线段裁剪算法,在求交流程之前,通过对平面连续分区,简单而迅速地舍弃比 2 次编码更多的窗外线段,避免更多不必要的求交运算。该算法的关键在于平面分区的设计,平面分区采用 3 次或多次分区的设计,对于不同的应用场合,选择合理的分区次数,可以达到较好的效果。

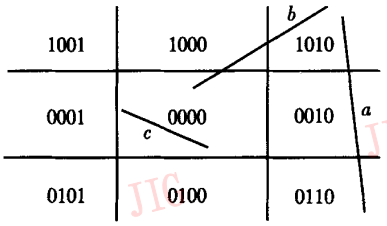

图 1　1 次分区(编码裁剪算法)

Fig. 1　Once zoning(Coding clipping algorithm)

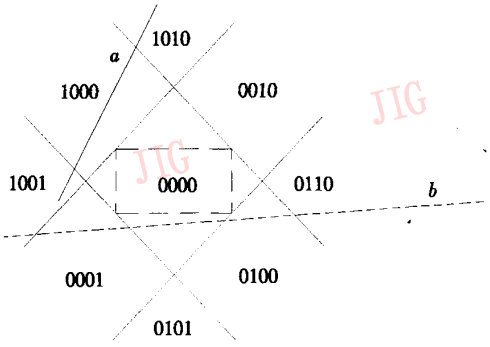

图 2　2 次分区(2 次编码)

Fig. 2　Secondary zoning(Secondary coding)

2　2 次分区与串联编码

2.1　连续分区思想与 2 次分区

连续分区就是按不同的方式多次将平面分成不同的区域,每次分成的区域都可以用简单的编码或关系式表示,用以简单而迅速地舍弃各类窗外线段,从而避免许多求交运算。将 2 次编码中两次编码过程分别视为第 1、第 2 次分区。

首先,2 次编码应用编码裁剪算法把平面分成 9 个区域,将线段两端点按区域预编码(如图 1 所示)。通过对端点编码的位运算就可以取舍窗内线段和完全在窗口外同一侧的线段。其判断条件如下:

$$code(A) \mid code(B) == 0 \qquad (1)$$

$$code(A) \& code(B) != 0 \qquad (2)$$

其中,A,B 为线段的两个端点;$\mid$ 是各位的逻辑或运算;& 是各位的逻辑与运算。当满足式(1)时,线段完全在窗口内,为完全可见线段,如图 1 中的线段 c,取该线段;当满足式(2)时,线段完全在窗口外同一侧,为显然完全不可见线段,如图 1 中的线段 a,舍弃该线段。然后,如图 2 所示,将原窗口(虚线所示)扩大并旋转 45°,形成新窗口(如实线所示),使新窗口刚好覆盖原窗口。根据新窗口的边界,重新把平面分成 9 个区域,并对区域编码,则完全在新窗口外同一侧的线段,如图 2 中的线段 a,就可以通过与式(2)相同的判断条件而舍弃。然而,对于某些窗外线段,例如图 2 中的线段 b,2 次编码就"无能为力"了,只能通过求交流程来解决。

2 次编码对平面进行了两次分区,本文对这两次分区方法深入研究之后得出:将两次编码有效地衔接起来——串联形成 1 次编码,这样,虽然能舍弃的窗外线段与 2 次编码同样多,但可以使先前的编码仍然有效,而不是重新对平面分区、编码,从而可以节省很多时间,大大提高裁剪效率。其中利用的技术就是本文提出的串联编码技术。

2.2　串联编码

串联编码就是将 2 次编码中的第 2 次编码追加在第 1 次编码之后,形成新的串联的 8 位编码,这样,平面被分成 29 个区域,每个区域都用一个不同的 8 位编码表示,如图 3 所示,各区域的编码如表 1 所示,边界的编码包含在各区域编码中。然后根据线段端点编码的位运算即可迅速实现线段的取舍,其判断条件与式(1)、(2)相同。例如,图 3 中的线段 c 可根据式(1)而取,线段 a、b 可根据式(2)而舍弃。若 XL,XR,YB,YT 分别表示裁剪窗口的左边界、右边界、下边界和上边界,串联编码算法如下:

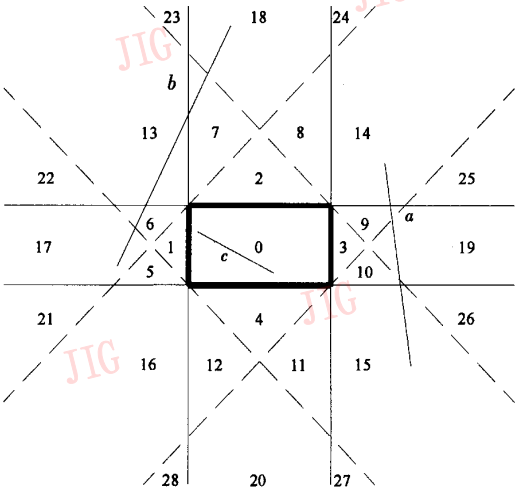

图 3 串联编码

Fig. 3 Series coding

```
#define LEFT_BOTTOM 1
#define RIGHT_TOP 2
#define RIGHT_BOTTOM 4
#define LEFT_TOP 8
#define LEFT 16
#define RIGHT 32
#define BOTTOM 64
#define TOP 128
code = 0;
if(x < = XL) code = code | LEFT;
else if(x > = XR) code = code | RIGHT;
if(y < = YB) code = code | BOTTOM;
else if(y > = YT) code = code | TOP;
if(x + y - YB < = XL) code = code | LEFT_BOTTOM;
else if(x + y - YT > = XR) code = code | RIGHT_TOP;
if(y - x + XR < = YB) code = code | RIGHT_BOTTOM;
else if(y - x + XL > = YT) code = code | LEFT_TOP;
reture code;
```

表 1 3 次分区信息表

Tab. 1 Information of cubic zoning

区域编号	1 次编码	2 次编码	串联编码	条件判定 2(对应 B 点所在区域)	
0	0000	0000	00000000	—	
1	0001	0000	00010000	$code(B)$ = 64 | 32 | 16 | 68 | 36 | 34 | 18 | 100 | 50 | 38 | 102 | 54	(2,3,4,8,9,10,11,14,15,19,25,26)
2	1000	0000	10000000	$code(B)$ = 128 | 32 | 16 | 129 | 34 | 18 | 17 | 50 | 145 | 19 | 51 | 147	(1,3,4,5,10,11,12,15,16,20,27,28)
3	0010	0000	00100000	$code(B)$ = 128 | 64 | 16 | 129 | 136 | 72 | 17 | 200 | 145 | 137 | 153 | 201	(1,2,4,5,6,7,12,13,16,17,21,22)
4	0100	0000	01000000	$code(B)$ = 128 | 64 | 32 | 136 | 72 | 68 | 36 | 200 | 100 | 76 | 204 | 108	(1,2,3,6,7,8,9,13,14,18,23,24)
5	0001	0001	00010001	$code(B)$ = 64 | 32 | 68 | 36 | 34 | 100 | 38 | 102	(2,3,8,9,10,14,19,25)
6	0001	1000	00011000	$code(B)$ = 32 | 16 | 36 | 34 | 18 | 50 | 38 | 54	(3,4,9,10,11,15,19,26)
7	1000	1000	10001000	$code(B)$ = 32 | 16 | 34 | 18 | 17 | 50 | 19 | 51	(3,4,10,11,12,15,20,27)
8	1000	0010	10000010	$code(B)$ = 128 | 16 | 129 | 18 | 17 | 145 | 19 | 147	(1,4,5,11,12,16,20,28)
9	0010	0010	00100010	$code(B)$ = 128 | 16 | 129 | 136 | 17 | 145 | 137 | 153	(1,4,5,6,12,16,17,21)
10	0010	0100	00100100	$code(B)$ = 128 | 64 | 129 | 136 | 72 | 200 | 137 | 201	(1,2,5,6,7,13,17,22)
11	0100	0100	01000100	$code(B)$ = 128 | 64 | 136 | 72 | 68 | 200 | 76 | 204	(1,2,6,7,8,13,18,23)
12	0100	0001	01000001	$code(B)$ = 64 | 32 | 72 | 68 | 36 | 100 | 76 | 108	(2,3,7,8,9,14,18,24)
13	1001	1000	10011000	$code(B)$ = 32 | 16 | 34 | 18 | 50	(3,4,10,11,15)
14	1010	0010	10100010	$code(B)$ = 128 | 16 | 129 | 17 | 145	(1,4,5,12,16)
15	0110	0100	01100100	$code(B)$ = 128 | 64 | 136 | 72 | 200	(1,2,6,7,13)
16	0101	0001	01010001	$code(B)$ = 64 | 32 | 68 | 36 | 100	(2,3,8,9,14)
17	0001	1001	00011001	$code(B)$ = 32 | 36 | 34 | 38	(3,9,10,19)
18	1000	1010	10001010	$code(B)$ = 16 | 18 | 17 | 19	(4,11,12,20)
19	0010	0110	00100110	$code(B)$ = 128 | 129 | 136 | 137	(1,5,6,17)
20	0100	0101	01000101	$code(B)$ = 64 | 72 | 68 | 76	(2,7,8,18)
21	0101	1001	01011001	$code(B)$ = 32 | 36	(3,9)
22	1001	1001	10011001	$code(B)$ = 32 | 34	(3,10)
23	1001	1010	10011010	$code(B)$ = 16 | 18	(4,11)
24	1010	1010	10101010	$code(B)$ = 16 | 17	(4,12)
25	1010	0110	10100110	$code(B)$ = 128 | 129	(1,5)
26	0110	0110	01100110	$code(B)$ = 128 | 136	(1,6)
27	0110	0101	01100101	$code(B)$ = 64 | 72	(2,7)
28	0101	0101	01010101	$code(B)$ = 64 | 68	(2,8)

注:式 $code(B)$ = 34 \| 32 代表程序表达式 $code(B)$ = = 34 || $code(B)$ = = 32,依此类推。

与 2 次编码相比,串联编码解决了边界编码问题,使编码技术更加完善,且串联编码方便、统一、可继承性好,同时又为后续分区提供了前提条件。

3　3 次分区与点对称

现在只考虑所要裁剪线段 AB 的任一端点(例如起点 A,按终点 B 讨论效果相同)位于除裁剪窗口区域外 28 个区域时的情况,因为点 A 在裁剪窗口内时,线段 AB 或完全可见或与裁剪窗口相交,前种情况已在上节中讨论过了,后种情况将经过后述的条件判定 1 直接进入求交流程。

当点 A 位于某一区域时(以图 4 所示情况为例),经过前面两次分区以后,许多情况的窗外线段已被舍弃,但当端点 B 位于两剖面线区域时的窗外线段 AB 仍不能舍弃。3 次分区及后述的多次分区就是解决这两个区域中的线段舍弃问题,且能无需乘除,继续简单舍弃许多线段。其中利用的技术就是本文提出的点对称技术。

如图 5 所示,做点 A 关于与其所在区域相对应的两个窗口顶点(O_1、O_3)的对称点 A'_1、A'_2,再分别过点 A'_1、A'_2 做两条特殊的射线(斜率为 0°和 45°),形成两个夹角为 135°的剖面线区域(如图 5 所示),这两个区域称为端点 B 的有效区,此过程即为 3 次分区。对应窗口顶点与特殊射线的方向可由以下法则确定。

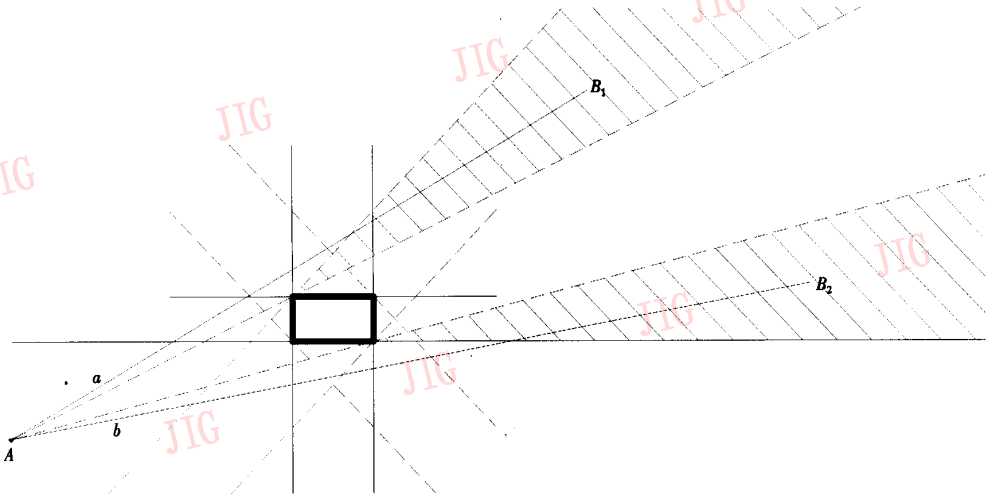

图 4　舍弃剩余区域分析

Fig. 4　Analysis of areas left after being abandoned

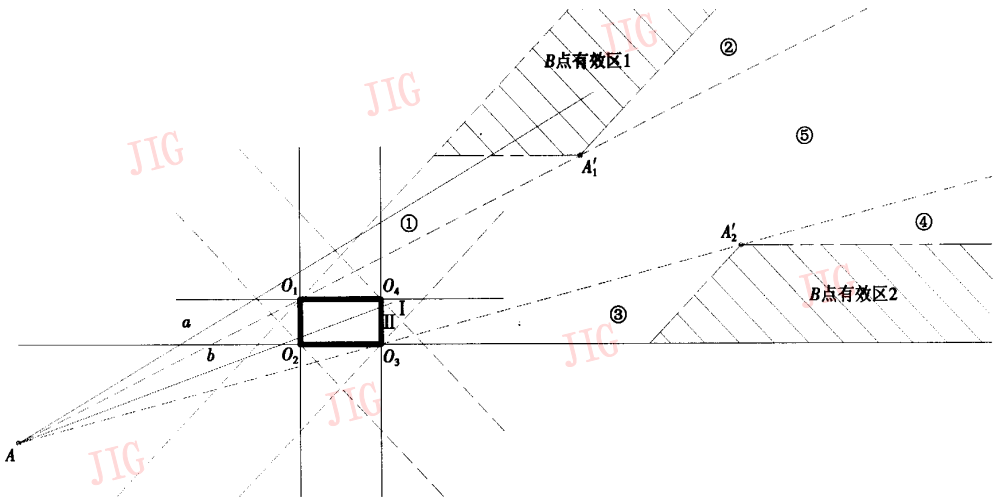

图 5　3 次分区(点对称技术)

Fig. 5　Cubic zoning(Point symmetry technique)

法则 1 过点 A 做任一与裁剪窗口不相交的射线，再将其绕 A 分别顺时针、逆时针旋转，将射线首次碰到的两个窗口顶点作为点 A 所在区域的对应窗口顶点。

法则 2 过点 A'所作的特殊射线不应与两对称线相交。

当点 B 位于有效区时，线段 AB 就可以通过简单的逻辑与运算而迅速舍弃。由于有效区的两个边界均为特殊斜率的射线，故可根据法则 1、2 编程简单实现。图 5 所示情况的有效区的实现条件如下：

$$(y_B > y_{A_1'}) \&\& (y_B - x_B + x_{A_1'} > y_{A_1'}) == 1 \quad (3)$$

$$(y_B < y_{A_2'}) \&\& (y_B - x_B + x_{A_2'} < y_{A_2'}) == 1 \quad (4)$$

其中，$x_{A_1'} = 2XL - x_A$，$y_{A_1'} = 2YT - y_A$，$x_{A_2'} = 2XR - x_A$，$y_{A_2'} = 2YB - y_A$，(x_A, y_A)为点 A 的坐标，(x_B, y_B)为点 B 的坐标，&& 为逻辑与运算。

式(3)表示点 B 位于有效区 1 中，式(4)表示点 B 位于有效区 2 中。

这样，原来需要采用求交运算处理的部分窗外线段，如图 5 中的线段 a 和图 2 中的线段 b，就可以采用 3 次分区而迅速舍弃，其判断条件分别为式(3)、(4)。当然，对于 3 次分区的不同区域，端点 B 的有效区及其实现条件也不同，故应对 28 个区域分别进行讨论。虽然所分情况之多，但是每种情况都用一个 *case* 模块来描述，再用 *switch*(*code*(*A*))函数调用与之相对应的 *case* 模块即可迅速实现其功能。

可见，点对称技术摆脱了 2 次编码思想的束缚，除了能舍弃完全在窗口外同一侧的线段外，还能舍弃许多其他情况的窗外线段。

易知，对于那些经过点对称运算能够被舍弃的窗外线段来说，点对称技术确能使裁剪算法提前结束；但对于那些明显与裁剪窗口相交的线段，如图 1 中的线段 b 和图 5 中的线段 b，点对称技术是多余的，它反而降低了裁剪效率。因此在线段进入点对称流程之前必须进行一些条件判定，尽量避免没有必要的运算。无需点对称运算的条件判定如下：

条件判定 1 若线段两端点 A、B 有一点在裁剪窗口内，另一点在裁剪窗口外，则该线段显然需要求交，可直接进入求交流程，其判定条件为

$$code(A) \mid code(B) == code(A) \| (A) \mid code(B) == code(B) \quad (5)$$

条件判定 2 对于图 5 所示情况，若点 B 位于区域 Ⅰ（编码为 00100010）或区域 Ⅱ（编码为 00100000）时，则线段 AB 显然也需要求交，也可直接进入求交流程，其判定条件为

$$code(B) == 34 \| code(B) == 32 \quad (6)$$

显然，对于不同的区域，条件判定 2 也有所不同，所以此判定应放在对应 *case* 模块的开始。（各区域对应的条件判定 2 见表 1）。

求交流程要经过求交点和判断交点是否有效两个步骤，条件判定 1、2 保证线段与裁剪窗口必有交点，故可省去判断交点是否有效这一求交步骤，从而可以加快求交的速度。

下面讨论经过 3 次分区处理后剩余线段 AB 的处理方法。如图 5 所示。此时，端点 B 可位于①～⑤5 个区域中。其中，位于区域①～④时 AB 为窗外线段，位于区域⑤时 AB 与裁剪窗口相交。

首先，求出线段 AB 所在直线的方程 l，然后通过判断对应窗口顶点 $O_1(XL, YB)$、$O_3(XR, YT)$与 l 的相对位置，决定线段 AB 的取舍。舍弃线段 AB 的条件判定为条件判定 3。

条件判定 3 若点 O_1 位于 l 下方或点 O_3 位于 l 上方，则线段 AB 为窗外线段，舍弃该线段，其判定条件为

$$YB > (y_B - y_A) \cdot (XL - x_A)/(x_B - x_A) + y_A \quad (7)$$

$$YT < (y_B - y_A) \cdot (XR - x_A)/(x_B - x_A) + y_A \quad (8)$$

式(7)表示点 B 位于区域①或②中，式(8)表示点 B 位于区域③或④中。

若条件判定 3 不成立，则线段 AB 与裁剪窗口必相交，并且至少求出了一个交点，减少了求交流程的工作量。

同条件判定 2 一样，对于不同的区域，条件判定 3 也有所不同，故也应针对 28 个区域分别进行讨论。可见串联编码的重要性。

3 个条件判定保证了本文算法的求交流程只需处理与裁剪窗口相交的线段，这样使算法更加清晰明了。

4 多次分区

3 次分区是将点 A 关于裁剪窗口对应顶点进行 1 次对称得对称点 A'，以 A'所确定的有效区舍弃部分窗外线段，称其为 1 次对称技术。为了扩大有效区范围，舍弃更多的窗外线段，可以采用 4 次分区，

甚至更多次分区,也即 2 次对称或多次对称技术。易知,多次分区仍采用对称点求取与逻辑与运算,实现简单,可以取得较好的效果。图 6 给出了 4 次分区的线段裁剪实例:在 1 次对称的基础上,作点 A 关于 A'的对称点 A'',采用与第 3 节类似的有效区实现条件,则 A''又确定了一个不同的范围更大的有效区,于是又可以舍弃许多窗外线段。

图 6 中的线段 a、b 均属此类情况。可见,分区次数越多,所能舍弃的窗外线段也就越多,从而可以避免更多的求交运算,并且,多次分区是整个裁剪流程的附加流程,它的存在与否不影响裁剪流程的完整性,它只起到加速器的作用,能高效地舍弃窗外线段。

综上所述,基于连续分区与串联编码的线段裁剪新算法的完整流程图如图 7 所示。

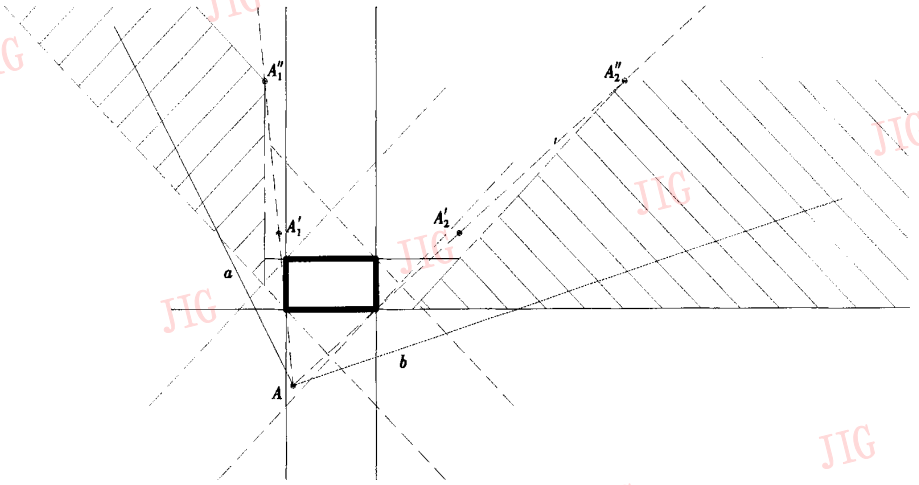

图 6　4 次分区(2 次对称技术)

Fig. 6　Quartic zoning(Secondary symmetry technique)

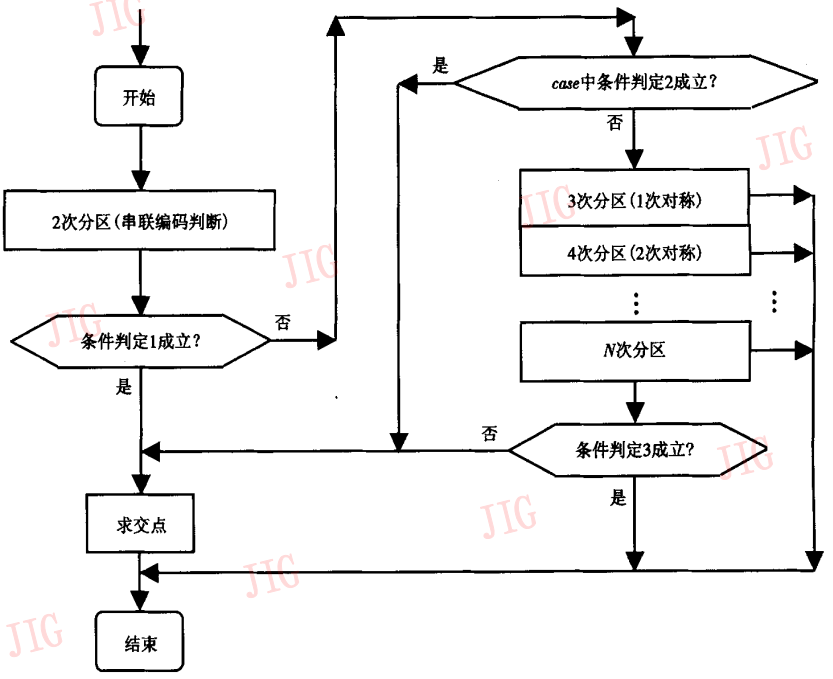

图 7　线段裁剪新算法流程图

Fig. 7　Flow chart of novel algorithm for line clipping

5 裁剪算法实现与讨论

为方便起见,本实验只采用了 3 次分区。如图 8 所示,线段 $a \sim h$ 分别代表不同位置的线段族,故具有一般性。其中,线段 a 可以由 1 次编码舍弃,线段 b 符合条件判定 1,线段 c 可以由 1 次编码而取,线段 d 必须经过求交流程,线段 e 与裁剪窗口相交而不符合条件判定 2,线段 f 符合条件判定 2,线段 g 可以由 2 次编码舍弃,线段 h 可以由 3 次分区舍弃。表 2 给出了分别对这 8 条线段重复裁剪 500 万次时,新算法与 2 次编码所用时间的比较。

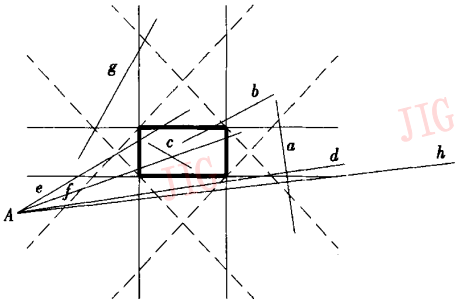

图 8 裁剪线段举例

Fig. 8 Examples for line clipping

表 2 裁剪时间比较一

Tab. 2 Comparision one of clipping time

单位:s

算法	裁剪线段							
	a	b	c	d	e	f	g	h
2 次编码	1.071	2.534	1.122	3.435	2.874	4.156	2.033	2.854
新算法	1.291	2.704	1.272	2.984	2.283	3.475	1.322	1.593
2 次编码-新算法	-0.220	-0.170	-0.150	0.451	0.591	0.681	0.711	1.261

由表 2 可以看出:

(1) 新算法在裁剪线段 a、b 和 c 时,速度比 2 次编码稍慢。这是因为 2 次编码中两次编码是分开的,而串联编码将两次编码集成为一体了,在裁剪这 3 条线段时只用到了 1 次编码,没有用到第 2 次编码。

(2) 新算法在裁剪线段 g 时,速度明显比 2 次编码快很多。这说明串联编码确实比独立的两次编码节省了很多时间,体现了串联编码的优势,这一点与上述讨论情况相符。新算法在裁剪线段 d、e、f、h 时都会有此优势。

(3) 裁剪线段 d、e 时,新算法虽然比 2 次编码多用了无效的点对称运算,但是,裁剪速度仍比 2 次编码快很多,这说明点对称运算消耗时间较少,更说明了串联编码的优势所在。

(4) 与 2 次编码相比,新算法裁剪线段 f 的情况明显优于裁剪线段 d、e 的情况,这就体现了判定条件 2 所起的作用。

(5) 点对称技术的优势完全可以从裁剪线段 h 看出:新算法裁剪该线段的效率比 2 次编码提高了 44.2%。

所以,新算法只有在裁剪前 3 种线段时效率比 2 次编码稍低,但在裁剪后 5 种线段时新算法有明显优势,故整体看来新算法优于 2 次编码。

对线段 $a \sim h$ 重复裁剪不同次数时,新算法与 2 次编码所用时间的比较如表 3 所示。随着裁剪次数的增多,2 次编码与新算法所用的时间之差也增大。可见,与 2 次编码相比,新算法提高了线段裁剪效率。

表 3 裁剪时间比较二

Tab. 3 Comparision two of clipping time

单位:s

算法	裁剪次数(万)						
	20	50	100	150	200	250	300
2 次编码	0.851	2.133	4.256	6.379	8.502	10.625	12.789
新算法	0.741	1.853	3.686	5.528	7.370	9.203	11.036
2 次编码-新算法	0.110	0.280	0.570	0.851	1.132	1.422	1.753

6 结　论

实际上,2 次编码可以看作是并联编码,因为其中的两次编码互不影响、各不相干。本文从另一个角度出发,以连续分区不断舍弃窗外线段为主要思想,首先提出了新的串联编码技术,并在此基础上又提出了新的点对称技术,形成了一个有新意的"一思想,两技术"的线段裁剪算法,取得了很好的效果。该算法在保持编码技术优点的同时,还继承了第一次编码所做的工作,并且能舍弃许多 2 次编码无法舍弃的窗外线段。同时,条件判定又可以加快求交的速度,故本算法既从尽量避免求交,又从加快求交速度两方面对线段裁剪算法进行了改进。实验结果表明,新算法确能提高线段的裁剪效率。

本算法的可扩展性为用户提供了更为灵活的方式:可以根据裁剪对象的不同,相应地更改控制程序,就可以获得所需要的裁剪效率。本文提出的连续分区思想、串联编码与点对称技术可以广泛地应用于其他的图形裁剪,如线段的多边形窗口裁剪等。

参考文献(References)

1 Sproull R F, Sutherland I E. A Clipping Divide[M]. Washington, DC, USA:Thompson Books, 1998: 765 ~ 775.

2 Liang Y D, Barsky B A. A new concept and method for the line clipping[J]. ACM Transactions on Grsphics, 1984, 3(1): 1 ~ 22.

3 Lu Guo-dong, Wu Xuan-hui, Peng Qun-sheng. An efficient line clipping algorithm based on adaptive line rejection[J]. Computers and Graphics, 2002, **26**(3): 409 ~ 415. [陆国栋,吴煊辉,彭群生. An efficient line clipping algorithm based on adaptive line rejection[J]. Computers and Graphics, 2002, **26**(3):409 ~ 415.]

4 Wang Jun, Liang You-dong, Peng Qun-sheng. A 2-D line clipping algorithm with the least arithmetic operations[J]. Chinese Journal Computer, 1991, **7**(7): 495 ~ 504. [王骏,梁友栋,彭群生. 具有最少算术运算量的二维线裁剪算法[J]. 计算机学报, 1991, **7**(7): 495 ~ 504.]

5 Wang Hao-hong, Wu Rui-xun, Cai Shi-jie. A new efficient line clipping algorithm based on geometric transformation[J]. Journal of Software, 1998, **9**(10): 728 ~ 733. [汪灏泓,吴锐迅,蔡士杰. 一种基于几何变换的高效的线裁剪新算法[J]. 软件学报,1998, **9**(10): 728 ~ 733.]

6 Lu Guo-dong, Zhang Shu-you. Engineering computer graphics[M]. Beijing: Science Press, 2004:141. [陆国栋,张树有. 工程计算机图形学[M]. 北京:科学出版社, 2004:141.]

7 He Dong-qi, Lu Guo-dong, Tan Jian-rong. A high efficient polygon clipping algorithm against rectangular window based on encoding and classification technique[J]. Computer Engineering & Application, 2003, **39**(21): 56 ~ 58,89. [何陈祺,陆国栋,谭建荣. 基于编码与分类技术的任意多边形裁剪新算法[J]. 计算机工程与应用, 2003, **39**(21): 56 ~ 58,89.]

8 Shang Ming-qing, Lu Guo-dong, Tan Jian-rong. A new line clipping algorithm based on window and line geometry transformation[J]. Computer Engineering & Application, 2003, **39**(20): 71 ~ 73,121. [商明清,陆国栋,谭建荣. 基于窗口与线段双重几何变换的线段裁剪新算法[J]. 计算机工程与应用, 2003, **39**(20): 71 ~ 73,121.]