一种改进的扫描线多边形填充算法

张玉芳　刘　君　彭　燕
（重庆大学计算机学院　重庆400044）

摘　要　典型的多边形填充算法主要包括扫描线填充算法和轮廓标志域填充算法，适用于矢量多边形文件的填充算法为扫描线填充算法。论文对原有的多边形扫描线填充算法中的最常用的活性边表和传统扫描线算法进行了分析，结合活性边表和传统的扫描线填充算法的特点，针对复杂的大数据量的多边形填充时间效率较低的问题，提出了一种改进的扫描线多边形填充算法—混合填充算法。该算法采用链表和数组结合的数据结构，形成连续的填充轨迹，有效地提高了时间效率。

关键词　计算机图形学，多边形，填充算法

An Improved Scan Line Polygon Filling Algorithm

ZHANG Yu-Fang　LIU Jun　PENG Yan
(Department of Computer Science, Chongqing University, Chongqing 400044)

Abstract　Typical polygon filling algorithm mainly includes scan line filling algorithm and area marks filling algorithm, and scan line filling algorithm suits vetor polygon filling. This paper makes some research on old and common polygon filling algorithm, and proposes an improved scan line polygon filling algorithm—mixed filling algorithm, which has the both advantages of traditional and AET scan line polygon filling algorithm on complicated and mass data polygon filling. This algorithm adapts both list table structure and array structure, and can form unbroken filling line, and is time efficiently.

Keywords　Computer graphics, Polygon, Filling algorithm

1　引言

在广告设计、信息传输、机械制造等领域，计算机图形学中图形的区域填充是基本图形操作。而多边形的填充问题，是区域填充中最为普遍的问题。例如，在雕刻图章的软件中对空心的汉字的填充，在激光制版中，对复杂图形的填充都是多边形填充问题。在这些领域中，多边形填充算法的优劣直接影响了生产效率和质量。经对现有市场上的图形软件进行分析，发现原有的多边形填充算法在时间效率上可以更进一步提高，填充轨迹可以更为流畅。

多边形填充算法主要有两种：扫描线填充算法和轮廓标志域填充算法。在矢量图形的填充问题上，扫描线填充算法是比较可行的一种算法。本文提出的混合填充算法是针对多边形的扫描线填充算法的改进。比较常用的多边形扫描线填充算法有传统的扫描线算法和活性边表算法。

2　传统的扫描线算法

传统扫描线填充算法的基本步骤如下：

step1：求出每一条水平扫描线与多边形各边的交点。

step2：将每一条水平扫描线上的交点按 X 值不减的次序排序。

Step3：将排序好的交点配成“交点对”。

Step4：将交点对之间进行填充。

在 step4中，如果是位图文件，则是填色，但在矢量图形中，一般是形成新的线段。

其算法可以表述为：(数据结构如下：二维数组 A[Ymax][n]，B[Ymax][n]，Ymax 为扫描线最大值，n 为与扫描线的最大交点数)：

```
Y=Ymax;
While(y>=Ymin){//扫描线最小值
      求扫描线与每条边的交点 X 值，与左侧边的交点存 A[y]，与
      右侧边的交点存 B[y]，A[y]和 B[y]相当于 y 桶，插入桶时按 x
      递增排序；
      y=y+1;
      }
      y=Ymax;
while(y>=Ymin){//Ymin 是扫描线最小值
      for(i=0;i<n;i++)
      填充(A[y][i])与(B[i][y])之间的区段
      y=y+1;
        }
```

3　活性边表算法

活性边表算法也是一种有效的多边形填充算法。与传统的扫描线算法相比，活性边表算法更注意上下扫描线的相关性。设当前扫描线 y 与某边交点坐标为(x,y)，则下一条扫描线 $y-1$与此边的交点 x 坐标为 $x-1/k$，k 为边的斜率。从而避免了求交运算。算法可以分为两大部分：

1）建立边表。对每条边建立如下的节点信息：

```
struct degeInfo{
int x;//保存两端点中，y 坐标较大点的 x 坐标
double deltaX;//-1/k
double deltaY;//与之相交的扫描线条数
struct edgeInfo * next;//指向另一条边的节点
};
```

把边节点放入与之相交的最大 y 扫描线桶中。

2）扫描线填充。填充线段的数据通过建立成为活性表的动态链表来存放。算法的类 C 语言描述如下：Ymax，Ymin 为 y 扫描线的最大最小值；Bucket 是扫描线桶；AEL 是活性表指针。

```
Stuct edgeInfo * Bucket[Ymax], * AEL;
Y=Ymax;
While(y>=Ymin){
Append(AEL,Bucket[y]);//把 Bucket[y]链到 AEL 之后
Sort(&AEL);//对 AEL 按 x 成员递增排序；
Fillscan(&AEL);//按顺序逐对取出节点填充扫描线段；
Update(AEL);//修改 AEL;
```

```
DeleteZeroNode(&AEL)//删除 deltaY 成员为零的节点;
Y=y-1;
  }
```

上述程序段 *append*()把放在扫描线 *y* 桶中的边的信息加入到活性边表中;*sort*()函数排序保证第1和第2节点、第3和第4节点等等奇节点和偶节点间的区段落在区域内。*Update*()修改 *AEL* 每个节点的 *x* 和 *deltaY* 成员:$x=x+deltaX$;$deltaY=deltaY-1$;活性表中 $deltaY=0$的节点对应边的扫描线已填充完毕,调用 *deleteZeroNode*()删除。

为了分析以上两种算法在实际的应用中的时间处理分布情况和效率以及优缺点,我们从以下的思路对其进行剖析。

假设多边形图形由 N 条边组成,其 Y 值的最大值为 $Ymax$,最小值为 $Ymin$,扫描线之间的间隔为1,所以扫描线的条数 $M=Ymax-Ymin$;

算法类型	文件I/O次数	最坏情况下加法次数	最坏情况下乘法次数	最坏情况下排序比较次数	算法总的时间复杂度
传统的扫描线算法	N次	$3*N*M$	$N+N*M$	$N*M$	$O(N*M)$
活性边表算法	1次	$2*N*M$	N	$(N*M)*(M-1)$	$O(N*M^2)$

通过以上的分析,可以看到,传统的扫描线算法在读入文件时 I/O 次数较多,比较浪费时间,计算交点时的运算量太大,这是因为传统的扫描线算法是采用的读入单条线段,对其进行求交处理,完成之后,再读入下一条线段。而活性边表算法将多边形线段一次读入,形成边表,在求交运算上充分利用边表的优势,省去了大量的运算,但其在交点排序中比较运算量较大,这是因为其存放交点的数据结构采用的是动态链表,虽然灵活却牺牲了时间效率。

4 混合填充算法

根据对传统算法和活性边表算法的优缺点进行分析,提出了一种改进的扫描线填充算法---混合填充算法。

1) PLT 文件格式的多边形的特点 PLT 矢量文件格式是专门针对打印等工作的一种矢量文件格式。文件由一系列的点坐标组成,这些点坐标即是边的顶点坐标。

2) 算法采用的数据结构 针对矢量多边形文件的特点,尽量减少调用文件 I/O 的频率,在本算法中采用与活性边表相同的边表结构,将多边形的边信息一次读入,对每条边建立如下的节点信息。

为了更进一步地提高效率,本算法在对每个边处理成边表节点的时候,先要进行一个取整的处理,将每一条边的节点的坐标进行取整,这样可以在不失真的情况下更为高效。同时,扫描线的间隔 Δy 也进行取整,这样可以有效地提高计算节点的交点的效率,也不会降低字型的优美。

与活性边表算法不同的地方是,在本算法中,还采用了传统扫描算法的数组数据结构,二维数组 $A[Ymax][n]$,$B[Ymax][n]$,$Ymax$ 为扫描线最大值,n 为与扫描线的最大交点数。之所以采用数组来存放计算出来的交点坐标,目的是为了将交点在存放时直接左右配对,这样存放避免了采用动态链表结构存放交点数据时,每次新交点插入时的大量而又不必要的重复比较操作,可以提高时间效率。

同时,采用数组存放交点的坐标,可以有效地实现本文提出的新的弓字型填充轨迹,在实际的应用中减少机械起落刀,改善机械填充雕刻出来的字体变形。

3)总的算法 算法的类 C 语言表述如下:Ymax,Ymin 为 y 扫描线的最大最小值;Bucket 是扫描线桶;AEL 是活性表指针。

```
y=Ymax;
while(y>=Ymin){
  append(AEL,Bucket[y]);// 把 Bucket[y]链到 AEL 之后
  placetoarray(&AEL);//对 AEL 按 X 成员递增排序,将奇点放入 A
    [y][i],偶点放入 B[y][i]中;
  update(&AEL);//修改 AEL;
  deleteZeroNode(&AEL);//删除 deltaY 成员为零的节点;
  y=y-1;
    }
y=Ymax;
  dirc=false;//标记量,用来标记填充轨迹线的方向,为 true 轨迹线
             从偶点到奇点,为 false 则反之。
  while(y>=Ymin){//Ymin 是扫描线最小值
    for(I=0;I<n;I++)
      if(dirc=false)
        linepin(A[y][i],B[y][i])//填充线段从 A[y][i]填到 B[y]
                                  [i]
        linepin(B[y][i],B[y-1][i])//填充线段将扫描线连成弓
                                    字形
      else
        linepin(B[y][i],A[y][i])
        linepin(A[y][i],A[y-1][i])
        endif
      y=y-1;
      }
```

填充出来的路径实际效果如图1中左边的图形所示。

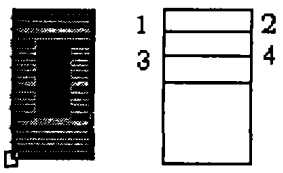

图1

在以上的填充算法中,可以看到本算法增加了一条填充线段,将上下相邻的扫描线之间同端的两个交点连接起来,算法总的时间复杂度并没有增加,但填充的轨迹有了神奇的变化。原有的填充算法在填充时通常采用如图1中右边图所示,先将多边形的边框勾勒出来,再用平行的填充线段从左到右填充。这样的话,如果在雕刻机中就会让雕刀从1点下刀刻到2点,在2点起刀,空走到3点下刀,再刻到4点。空动作量大,且由于机械运动的频繁,填充出来的字型反而不美观。而采用弓字型的填充,因为填充轨迹流畅,完全可以省掉勾勒多边形边框的过程,直接由填充线段组成字型,而且大大减少了机械起落刀的频率。所以,本填充算法在时间效率和实际应用中都有明显的优势。

通过对不同的数据量的文件进行算法测试(在计算机上)得到了如下的实验结果。

数据量(k)	传统扫描算法(ms)	活性边表算法(ms)	混合算法(ms)
文件1(76k)	120	108	111
文件2(115k)	142	120	123
文件3(151k)	161	139	136
文件4(295k)	297	261	259
文件5(573k)	504	438	421
文件6(1039k)	871	653	629

在做该实验时,我们选用了 PLT 矢量文件格式作为实验对象,在 Coreldraw11中导出我们所需要的不同量的复杂多边形文件(这里采用的是空心汉字,文件1是采用的宋体 18号字 10个舞字导出的空心汉字组成的复杂多边形文件)。为了减少 Windows 多任务操作系统中各种系统任务对算法统计

时间的随机影响，每个测试项均重复10，并取其中最小值作为有效结果。通过以上的实验数据，我们可以看到，混合的填充算法在处理较为复杂的数据量大的文件时是有优势的。

123456890
重庆大学

123456890
重庆大学

图2

该算法成功地运用到气动或激光等雕刻制版系统中，因为弓字形的填充轨迹更符合雕刻机雕刀行走的原理，因此在实际的运用中大受欢迎。

将该算法应用到气动雕刻机制版上效果更为明显。

图2的下半部分即为混合算法的填充效果在屏幕上画出来的效果，与雕刻出来的实际图形效果一致。

表1 不同扫描算法该图的完成时间(从计算到完成制版的时间)

算法名称	传统扫描算法	活性边表算法	混合算法
时间(毫秒)	3277	3081	2789

小结 结果表明，本文的混合填充算法利用对多边形数据进行取整，以及合理的数据结构，新的填充算法，可以在处理复杂的多边形填充问题上改进时间效率，改进填充出来的真实度。在激光制版系统、图形刻绘系统中经过实际检测有良好的效果。

参考文献

1 孙家广，杨长贵. 计算机图形学(新版). 清华大学出版社，1995
2 羊四清，李思昆. 改进的扫描线多边形填充算法的研究. 数学理论及应用，1999(6)
3 甘泉. 通用扫描线多边形填充算法. 计算机工程与应用，2000(2)

(上接第144页)

中使用的相同。约束条件为：$C=\mathrm{sum}(S.price)\geqslant 500$。图5(b)描述了上述两种算法基于交易数量的扩展性能，最小支持度阈值为0.5%，使用的数据集与图4(b)中使用的相同。约束条件为：$C=\mathrm{sum}(S.price)\geqslant 500$。

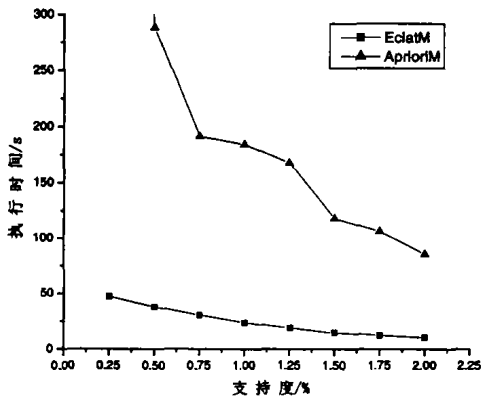

图5 (a)基于最小支持度的扩展性能

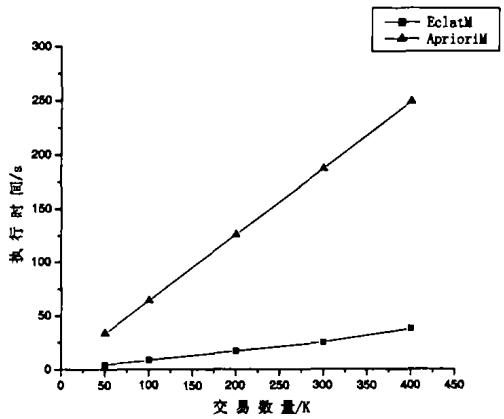

图5 (b)基于交易数量的扩展性能

从上述实验结果可以看出：

(1) EclatA 算法(EclatM 算法)基于最小支持度的扩展性能要优于 CAP 算法(AprioriM 算法)。主要体现在两个方面：一个是 EclatA 算法(EclatM 算法)的执行时间要远远少于 CAP 算法(AprioriM 算法)，另一个是当支持度阈值比较小时，CAP 算法(AprioriM 算法)的执行时间曲线的变化程度要比 EclatA 算法(EclatM 算法)剧烈。

(2) EclatA 算法(EclatM 算法)基于交易数量的扩展性能要优于 CAP 算法(AprioriM 算法)。EclatA 算法(EclatM 算法)和 CAP 算法(AprioriM 算法)均随事务数量的增长而线性增长，但是后两种算法的增长速度要远快于前两种算法。

总之，EclatA 算法(EclatM 算法)具有良好的扩展性，这是由于 Eclat 算法具有 Apriori 算法无法比拟的优越性能。此外，EclatM 算法是在挖掘过程中就进行了约束条件的检测，而 AprioriM 算法是在挖掘出所有频繁项集后再用约束条件来剪裁，这也是反映算法性能的一个重要因素。

结语 本文在 Eclat 算法的基础上，提出了一种新的单调和反单调约束条件下关联规则的挖掘算法。算法采用了垂直数据库结构，利用基于前缀的等价关系将概念格划分为较小的子概念格，在自底向上的频繁集搜索过程中，同时进行约束条件的检验。进一步的工作是如何将这种挖掘算法扩展到多种约束条件下，如简洁性约束和可转变约束。此外，如何在分布式环境下，利用概念格理论和等价类划分方法进行约束关联规则的挖掘也待进一步研究。

参考文献

1 Agrawal R, Imielinski T, Swami A. Mining association rules between sets of items in large databases. In: Proc. of the ACM-SIGMOD Conf. on Management of Data, 1993. 207～216
2 Agrawal R, Srikant R. Fast algorithms for mining association rules[A]. In: Proc. of the 20th Intl. Conf. on Very Large Databases[C], Santiago, Chile, 1994. 487～499
3 Han J, Pei J, Yin Y. Mining frequent patterns without candidate generation[A]. In: Proc. of the ACM-SIGMOD Conf. on Management of Data[C], Dallas, ACM Press, 2000. 1～12
4 Ng R T, Lakshmanan L V S, Han J, Pan A. Exploratory mining and pruning optimizations of constrained association rules[A]. In: Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)[C], Seattle, Washington, 1998. 13～24
5 Pei J, Han J. Can we push more constrained into frequent pattern mining? [A]. In: Proc. 2000 Int. Conf. Knowledge Discovery and Data Mining(KDD'00)[C], Boston, MA, 2000. 350～354
6 Pei J, Han J, Lakshmanan L V S. Mining frequent pattern with convertible constrains [A]. In: Proc. 2001 Int. Conf. Data Engineering(ICDE'01) [C], Heidelberg, Germany, 2001. 332～433
7 Zaki M J. Scalable algorithms for association mining [J]. IEEE Trans. Knowledge and Data Engineering, 2000, 12(3): 372～390
8 高飞，谢维信. 发现含有第一类项目约束的频繁集的快速算法[J]. 计算机研究与发展，2001，38(11)：295～301