第 19 卷第 4 期　　广西科学院学报　　Vol. 19, No. 4
2003 年 11 月　　Journal of Guangxi Academy of Sciences　　November 2003

# 免解二次方程的圆形窗口裁剪算法

# A Line-clipping Algorithm for Circle Clip Window that Needn't to Solve the Quadratic Equation

黄文钧　　谢宁新

Huang Wenjun　Xie Ningxin

（广西民族学院数学与计算机科学系　南宁　530006）

(Dept. of Math. & Comp. Sci., Guangxi Univ. for Nationalities, Nanning, 530006)

**摘要**　在相关文献提出的基于矩形窗口裁剪的圆形窗口裁剪算法的基础上，通过判断圆形窗口与待裁线段的位置关系，提出一个免解二次方程的圆形窗口裁剪算法. 该算法省去矩形裁剪步骤，同时也避免了解二次方程，大大减少算法的计算量.

**关键词**　裁剪　圆形窗口　算法

**中图法分类号**　TP301.6；O182

**Abstract**　Through judging the relation of the position between the circle clip window and the line to be clipped, a new line-clipping algorithm is given based on the correlative literatures. This algorithm can saves the cutting procedure, avoid to solve the quadratic equation and reduces the calculating amounts greatly.

**Key words**　clipping, circle clip window, algorithm

在计算机图形学中，二维裁剪窗口包括矩形、任意多边形和圆形等. 对于矩形窗口和任意多边形的情形，一些文献上已有深入地讨论[1,2]. 但对于圆形窗口，教科书上很少提及[1]. 由于在一些图形处理场合中需要圆形窗口裁剪，许多文献对圆形窗口进行了比较深入的讨论，其中的文献 [3] 提出了一种基于矩形窗口裁剪的圆形窗口裁剪算法. 其基本步骤是：在圆形窗口之外“罩上”一个外切矩形，然后先用矩形窗口裁剪[3]，再用圆形窗口裁剪，如图 1 所示. 该算法尽量减少圆形窗口裁剪，少解二次方程. 然而对于图 1 中的线段 $L1$，该算法的第一步是把它排除在圆形窗口之外，但对于 $L2$、$L3$、$L4$、$L5$、$L6$ 等 5 种情形，矩形裁剪做了无效果的裁剪，且还是避免不了圆形裁剪和解二次方程. 本文在文献 [3] 的基础上，通过判断圆形窗口与待裁线段的位置关系，提出一种高效、快速圆形窗口裁剪算法.

## 1　圆形窗口与待裁剪线段的位置关系

在平面上，圆与线段的位置关系共有 6 种，如图 1 所示.

(1)线段与线段所在直线在圆外，如 $L1$ 的情形.

2003-07-03 收稿，2003-07-15 修回。

(2)线段或线段所在直线与圆相切,如 $L2$ 的情形.

(3)整条线段在圆内,如 $L3$ 的情形.

(4)线段的一端点在圆内,另一端点在圆外,如 $L4$ 的情形.

(5) 线段的两端点在圆外,但其延长线和圆相交,如 $L5$ 的情形.

(6) 线段的两端点在圆外,但有部分在圆内,如 $L6$ 的情形.

对于以上 6 种位置关系的判断, 我们将提出相应的简洁数学方法.

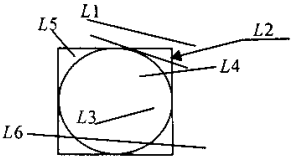


图 1 圆形窗口与待裁剪线段的 6 种位置关系

## 2 判断方法及定理

### 2.1 判断方法

在图 2 中,设圆形窗口的圆心为 $(x_C, y_C)$,半径为 $R$;待裁剪线段为 $L$,两端点到圆心的距离分别为 $d_0, d_1$;圆心到 $L$ 的距离记为 $d$;过圆心且与 $L$ 垂直的直线为 $L_C$. 于是有如下的事实成立:

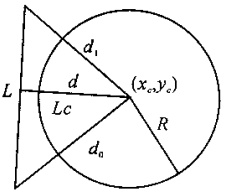


图 2 圆形窗口与任意一条待裁剪线段 $L$ 的位置关系

(1) 若 $d \geqslant R$,则 $L$ 在圆外(或相切);

(2) 若 $d < R, d_0 \leqslant R$ 且 $d_1 \leqslant R$,则线段 $L$ 在圆内;

(3) 若 $d < R, (d_0 - R)(d_1 - R) < 0$,则 $L$ 的一端在圆内,另一端在圆外(此时 $L$ 与圆有一个交点);

(4) 若 $d < R, d_0 \geqslant R$ 且 $d_1 \geqslant R$,同时 $L$ 的两端点在直线 $L_C$ 的同侧,则 $L$ 在圆外;

(5) 若 $d < R, d_0 \geqslant R$ 且 $d_1 \geqslant R$,同时线段 $L$ 的两端在 $L_C$ 的两侧,则 $L$ 有部分在圆内(此时 $L$ 和圆有两个交点).

以上的判断方法是显然的,故不证明.

### 2.2 定理

**定理 1** 设平面上两点 $p_0(x_0, y_0), p_1(x_1, y_1)$,直线 $L: Ax + By + C = 0$.

(1) 若 $(Ax_0 + By_0 + C)(Ax_1 + By_1 + C) > 0$,则两点 $p_0(x_0, y_0)$ 和 $p_1(x_1, y_1)$ 在直线 $L$ 的同侧;

(2) 若 $(Ax_0 + By_0 + C)(Ax_1 + By_1 + C) < 0$,则两点 $p_0(x_0, y_0)$ 和 $p_1(x_1, y_1)$ 在直线 $L$ 的异侧.

**证明** 其证明方法简单,此处从略.

**定理 2** 设直线 $p_1p_2$ 与圆 $O$ 的交点为 $j$,圆心 $O$ 点在线段 $p_0p_1$ 的垂足记为 $t$,点 $O$ 与点 $t$ 的距离记为 $d$,圆半径记为 $R$,如图 3 所示.

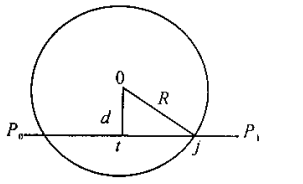


图 3 交点 j 与 d、半径 R 的关系

于是,$\lambda = tj/jp_1$,则点 $j$ 的坐标 $(x_j, y_j)$,可用下面公式求得:

$$x_j = (x_t + \lambda x_1)/(1 + \lambda),$$

$$y_j = (y_t + \lambda y_1)/(1 + \lambda).$$

**证明** 实际上是平面解析几何学中的定比分点公式.

这里我们看到，在求直线 $p_1p_2$ 和圆 $O$ 的交点时，利用定理 2 就可避免解二次方程.

## 3　圆形窗口裁剪算法

把以上数学方法和定理相结合，得到圆形窗口裁剪算法.

**步骤 1**　开始：float　$L[2][2]$；/* 存储一条待裁剪线段 */

float　$x_C, y_C, R$；/* 圆心与半径 */

float　$d$；/* 圆心$(x_C, y_C)$ 到直线 $L[2][2]$ 的距离 */

float　$L_C[2][2]$；/* 存储与 $L[2][2]$ 垂直且过圆心$(x_C, y_C)$ 的直线 */

float　$d_0, d_1$；/* 记录 $L[2][2]$ 两端点到圆心$(x_C, y_C)$ 的距离 */

**步骤 2**　若 $d \geqslant R$，跳到步骤 4，结束，否则进行步骤 3；

**步骤 3**　若 $d < R$

(1) 若 $d_0 < R$ 且 $d_1 < R$，显示线段 $L[2][2]$，结束；

(2) 若 $d_0 > R$ 且 $d_1 < R$，并且线段 $L[2][2]$ 的两端点在直线 $L_C$ 的同侧，则跳到步骤 4；

(3) 若 $d_0 > R$ 且 $d_1 < R$，并且线段 $L[2][2]$ 的两端点在直线 $L_C$ 异侧，则用定理 2 求出直线 $L[2][2]$ 与圆形窗口的 2 个交点，并且显示这 2 个交点确定的线段，结束；

(4) 若 $(d_0 - R)(d_1 - R) < 0$，则求直线 $L[2][2]$ 与圆形窗口的交点(只有一个)，并且显示该交点和直线 $L[2][2]$ 在圆形窗口内的那个端点确定的线段，结束；

**步骤 4**　不显示 $L[2][2]$；

**步骤 5**　结束.

表 1　2 种算法计算量对比

| | 乘法 | 除法 | 开平方 |
|---|---|---|---|
| 本文算法 | $5 \times 2$ | $3 \times 2$ | $1 \times 2$ |
| 文献[3] | $13 \times 3$ | $2 \times 3$ | $1 \times 3$ |

## 4　算法实现和计算量分析

### 4.1　算法实现

作者用 C 语言编程，在 TC 的环境中用 P4 微机实现了该算法，程序运行的结果见图 4 和图 5.

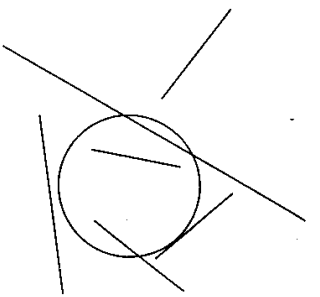

图 4　裁剪前的图形

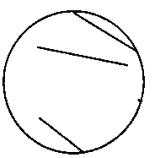

图 5　裁剪后的图形

### 4.2　计算量分析

设圆的方程为

$$(x - x_C)^2 + (y - y_C)^2 = R^2.$$

直线方程为

$$y = kx + b.$$

(1) 对于上述的 6 种直线和圆形窗口的关系，本文仅对 2 种情形(图 1 中的 $L4$ 及 $L6$) 作求交计算，而文献[3]要对 3 种情形(图 1 中的 $L4, L5, L6$) 作求交计算.

(2) 对于求交，本文算法用 5 次乘法，3 次除法，1 次开平方；而文献[3]用 13 次乘法，2 次除法，1 次开平方.

(3) 综合(1)和(2)，求交计算量见表 1 所示. 表 1 不包括文献[3]的矩形窗口裁剪计算量. 通过比较表 1 和文献[3]的矩形裁剪计算量(或运行时间)，可知文中提出的免解二次方程的圆形窗口裁剪算法

(下转第 164 页)

| 序号 | 最高湿度 | 重叠湿度 | 事务 1 湿度 | 事务 2 湿度 | 事务 3 湿度 |
|---|---|---|---|---|---|
| 1 | 300 | 500 | 0 | 300 | 200 |
| 2 | 100 | 150 | 0 | 50 | 100 |
| 3* | 0 | 310 | 200 | 100 | 10 |
| 4 | 150 | 250 | 0 | 150 | 100 |

* 为已安排过的位置

图 2 已安排事务的湿度

## 2 算法的时间复杂度分析

设事务个数为 $n$,可填充的位置数为 $k$,约束条件数为 $x$,则首次浸润时的循环次数为 $n \times k \times x$,设已安排好位置的事务数为 $y$,则每次搜索最高有效权值所需循环次数为 $k - y$,每次过滤所需的循环次数为 $k \times x \times (n - y)$,最多需要搜索和过滤 $n - 1$ 次,故整个算法的时间复杂度为:

$$\sum_{y=0}^{n-1}(n - y) \times k \times x \approx O(n^2 \times k \times x).$$

以目前的计算机运行速度来看,即使安排一个有上千个事务、上万个条件、数千个可安排位置的安排表,所需时间也不过几个小时,一般常见的应用可在几分钟内完成,所得效果距理想值十分接近。

(责任编辑:黎贞崇)

(上接第 161 页)

高效、快速.

## 5 结束语

文中提出的免解二次方程的圆形窗口裁剪算法主要特点是效率高. 例如在求出圆心到待裁剪线段的距离之后,不仅用它来判断待裁剪线段与圆的位置关系,还继续用它来求待裁剪线段和圆的交点. 另外,求待裁剪线段和圆的交点时,当求出第一个交点之后,求第二个交点只须用中点坐标公式即可,避免了第二次除法,而常规求交方法或者文献[3]就无法做到这一点.

### 参考文献

1 郝恩(美),贝克(美). 计算机图形学 C 语言版(英文). 北京:清华大学出版社,1998.2.

2 黄文钧. 一种快速的参数化裁剪算法. 西北大学学报,1999,6:29.

3 邹北骥. 基于矩形窗口裁剪的圆形窗口裁剪算法. 计算机工程与科学,1998,8:20.

(责任编辑:黎贞崇)