中图法分类号: TP3 文献标志码: A 文章编号: 1006-8961(2011)01-0124-05
论文索引信息: 陈超, 张兆印. 采用区域编码的椭圆对直线裁剪算法[J]. 中国图象图形学报,2011,16(1):124-128

采用区域编码的椭圆对直线裁剪算法

陈超, 张兆印
(黑龙江大学计算机科学技术学院, 哈尔滨 150080)

摘 要: 裁剪算法的核心问题是速度问题,而求裁剪窗口和裁剪对象的交点是影响裁剪速度的主要因素。特别是椭圆对线段的裁剪,由于椭圆的方程是二次的,求椭圆与线段的交点需要求解一元二次方程,涉及开方运算,非常浪费机器时间。为提高裁剪速度,设计出 5 位的区域编码,利用此技术能够迅速而准确地判断出椭圆和线段的位置关系。对于完全可见或显然完全不可见的线段立即做出保留或弃掉的决定,避免求交运算;对于能够明确断定与椭圆相交的线段,采用中点分割算法求椭圆和线段的近似交点,避免求解一元二次方程和开方运算;对于其他情形的线段通过求解一元二次方程来完成裁剪。基于前述思想设计出的椭圆对线段裁剪算法与现有的同类算法相比,算法实现简单,裁剪速度具有较大提高。

关键词: 线段裁剪; 椭圆形窗口; 区域编码; 中点分割算法

Line clipping algorithm with respect to elliptical window based on region encoding

Chen Chao, Zhang Zhaoyin
(*School of Computer Science and Technology, Heilongjiang University, Harbin* 150080 *China*)

Abstract: The key in clipping algorithm is the efficiency which is mainly influenced by computing the intersection points between the clipping window and the clipped object. Particularly, for line clipping with respect to elliptical window, to compute the intersection points between the ellipse and the line the quadratic equation has to be solved which involves the extraction of square root that is considered inefficient. To address this, we develop a technique of 5-bit region encoding by which the relationship between an ellipse and a line segment can be determined quickly and accurately. The line segments that are completely visible or completely invisible are discarded; the line segments that surely intersect with the ellipse are dealt with by middle-point segmentation algorithm to obtain the approximate intersection points; and the remaining line segments are clipped by solving the quadratic equations. The proposed algorithm is much more efficient than traditional algorithms, and is straightforward.

Keywords: line clipping; ellipse window; region encoding; middle point

0 引 言

裁剪窗口常见的是矩形,但是椭圆作为窗口的裁剪在实际工作中也很常见。用矩形裁剪线段已有很多经典的算法,如 Sutherland-Cohen 裁剪算法[1]、Sproull 和 Sutherland 的中点分割裁剪算法[2]等。而由于椭圆的方程是二次的,求椭圆与裁剪对象的交点一般都要求解二次方程[3-4],涉及效率较低的开方运算。裁剪处理是许多操作的基础,裁剪效率的提高是非常有意义的。主要影响裁剪效率的是裁剪窗口与裁剪对象的求交运算。为了避免不必要的求

收稿日期:2009-06-19;修回日期:2009-09-24

第一作者简介:陈超(1963—),男,副教授。2008 年于哈尔滨工程大学获计算机应用技术专业硕士学位,主要研究方向为计算机图形学。E-mail: chenchao@hlju.edu.cn。

交,一般做法是,首先根据窗口与裁剪对象的位置关系,直接丢弃显然落在窗口外面的裁剪对象,保留完全落在窗口内的裁剪对象,如果不能明确断定二者的相交性,最后才联立二者的方程,通过讨论方程组是否有解来判断二者是否相交,相交时求得交点。设计裁剪算法,人们都要在“判断位置关系”和“求交”这两方面下功夫而进行深入研究。

对于椭圆对线段的裁剪,可以用椭圆最小包围盒(即椭圆的外切矩形)的四条边直线与椭圆将整个平面分成 10 个区域,引进区域编码技术,利用线段端点编码的位运算,设计表示椭圆与线段各种位置关系的判别条件,且尽量将二者不相交的情形判断出来,避免很多不必要的求交运算。在明确二者相交时,利用中点分割算法,用中点作为交点的近似值,避免求解一元二次方程和开方运算。在最坏情况即不能明确断定二者的相交性时,采用通常方法——将直线的参数方程代入椭圆方程得到关于参数 t 的一元二次方程,讨论方程判别式 Δ 的符号:若 $\Delta \leqslant 0$,则椭圆与线段不相交(将相切亦当作不相交);否则,首先求方程两个不等的根 t_1 和 t_2,然后测试 t_1 和 t_2 是否落在闭区间[0,1]上来断定椭圆与线段的相交性。这种裁剪算法能够减少运行时间,提高裁剪效率,并且实现简单。

1　区域编码

1.1　区域编码

以如下椭圆为例,它的最小包围盒是一个以

$$\frac{(x-x_c)^2}{a^2}+\frac{(y-y_c)^2}{b^2}=1 \tag{1}$$

$O'(x_c,y_c)$ 为中心,边平行于坐标轴,边长分别是 $2a$ 和 $2b$ 的矩形。包围盒的左、右、下和上 4 条边直线的方程分别为:$x_l=x_c-a$,$x_r=x_c+a$,$y_b=y_c-b$ 和 $y_t=y_c+b$。这 4 条直线与椭圆将平面分成 10 个区域(将椭圆与其包围盒之间的 4 个不连续部分视为一个区域),如图 1。对这 10 个区域进行 5 位编码,称为区域编码。编码结构如下:

I	T	B	R	L

编码规则是,从右往左(从低位到高位),编码中各位取值如下:

$$L=\begin{cases}1 & \text{若区域位于包围盒左边直线的左面时}\\ 0 & \text{其他}\end{cases}$$

$$R=\begin{cases}1 & \text{若区域位于包围盒右边直线的右面时}\\ 0 & \text{其他}\end{cases}$$

$$B=\begin{cases}1 & \text{若区域位于包围盒下边直线的下面时}\\ 0 & \text{其他}\end{cases}$$

$$T=\begin{cases}1 & \text{若区域位于包围盒上边直线的上面时}\\ 0 & \text{其他}\end{cases}$$

$$I=\begin{cases}1 & \text{若区域位于椭圆内时}\\ 0 & \text{其他}\end{cases}$$

如图 1,各区域的编码已标明在各自区域之内。图中未标明编码的区域(即椭圆与其包围盒之间的区域)的编码是 00000。

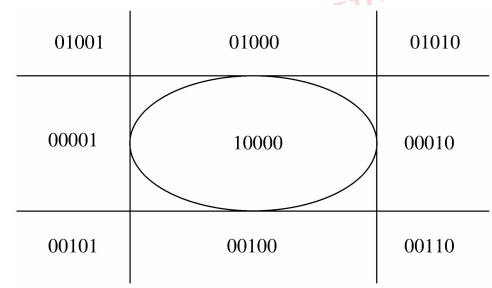

图 1　椭圆窗口的区域编码

Fig. 1　Ellipse window region encoding

1.2　点的编码

对于任意一点 P,它一定落在这 10 个区域中的某一个区域之内,所以可以定义点 P 所处区域的编码就是点 P 的编码。这样落在同一个区域的点,它们的编码相同;落在不同区域的点,它们的编码不同。

对于给定的任意一点(x,y),求其编码的算法为

```
#define  LEFT      1
#define  RIGHT     2
#define  BOTTOM    4
#define  TOP       8
#define  INTERIOR  16
void MakeCode(int x, int y, int *pcode)
{int c=0;
 if(x<=xl)   c|=LEFT;
 else if(x>=xr)   c|=RIGHT;
 if(y<=yb)   c|=BOTTOM;
 else if(y>=yt)   c|=TOP;
 if(!c && (x-xc)*(x-xc)/(a*a)+
          (y-yc)*(y-yc)/(b*b)<1)
     c|=INTERIOR;
 *pcode=c; // pcode 传回点(x,y)的编码
}
```

2 位置关系的判别方法及裁剪策略

可将椭圆与线段的位置关系分 5 种：

1）线段的两个端点均位于椭圆内，如图 2 中的 l_0。判别方法是线段两个端点编码的逻辑与运算结果为 16。这样的线段是完全可见的，裁剪时应该将其完全保留。

2）线段的两个端点均位于椭圆包围盒某条边直线的外侧（没有窗口的一侧），如图 2 中的 l_1。判别方法是线段两个端点编码的逻辑与运算结果为 1 或 2、4、8。这样的线段是显然完全不可见的，裁剪时应该将其完全丢弃。

3）线段的一个端点在椭圆内另一个端点在椭圆外，如图 2 中的 l_2。判别方法是线段两个端点编码的逻辑或运算结果大于等于 16。这样的线段与椭圆相交且仅相交一次，有一个唯一的交点，裁剪时应先求交点，交点分线段为两段，弃掉位于椭圆外的一段，保留位于椭圆内的一段。

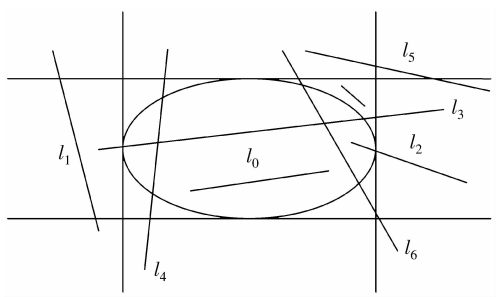

图 2 椭圆与线段的位置关系

Fig. 2 The location relationship between ellipse and line

4）线段的两个端点分别落在上下中间最左和最右两个相对的区域内（如图 2 中的 l_3），或落在左右中间最下和最上两个相对的区域内（如图 2 中的 l_4）。判别方法是线段两个端点编码的逻辑或运算结果等于 3 或 12。这样的线段与椭圆有两个交点，裁剪时应先求交点，保留两个交点之间的一段，弃掉位于椭圆外的两段。

5）包括除 1）—4）以外所有情形。它们的共同特征是线段的两个端点都在椭圆的外面，或者与椭圆无交点，如图 2 中的 l_5（切点不算交点，与椭圆相切的线段归于这种情形），这时线段是完全不可见的，应该完全丢弃；或者与椭圆有两个交点，如图 2 中的 l_6，裁剪时应先求交点，保留两个交点之间的一段，弃掉位于椭圆外的两段。这种位置关系无法直接采用编码判别。

3 交点的计算

3.1 中点分割算法

如果线段的一个端点在椭圆内另一个端点在椭圆外，那么线段与椭圆有唯一一个交点。在求这个交点时，为了避免求解一元二次方程和开方运算，可采用中点分割算法。中点分割算法的基本思想是用“中点”逼近交点。具体做法是，不断地在线段的中点处将线段一分为二，弃掉完全在椭圆内或椭圆外的一段，对折过程中产生的中点序列收敛于交点，选择适当的中点近似交点。

设线段 P_0P_1 的端点 $P_0(x_0,y_0)$ 在椭圆内，端点 $P_1(x_1,y_1)$ 在椭圆外。求椭圆（式（1））与线段 P_0P_1 交点的中点分割算法可以描述如下：

```
a2 = a * a; b2 = b * b;
flag = 1; //状态变量：值 1 表示未求得交点；值 0 表示已求得准确交点
while( ( abs(x1 - x0) > 1 || abs(y1 - y0) > 1 ) && flag )
{xm = (x0 + x1) >> 1; //用位运算求中点
 ym = (y0 + y1) >> 1;
 dm = (xm - xc) * (xm - xc) / a2 + (ym - yc) *
      (ym - yc) / b2 - 1; //中点代入椭圆的方程
 if( dm == 0 ) //中点在椭圆上，求得交点
          flag = 0;
 else if( dm > 0 ) //中点在椭圆外面
          {x1 = xm;
           y1 = ym;
          }
 else //中点在椭圆内
          {x0 = xm;
           y0 = ym;
          }
}
if( flag ) //满足分辨率精度 1 或 √2，这个中点作为交点的近似
   {xm = (x0 + x1) >> 1;
    ym = (y0 + y1) >> 1;
   }
//算法结束后，xm 和 ym 分别存放交点的横坐标和纵坐标。
```

当椭圆与线段的位置关系属于第 3）种时，直接调用中点分割算法就可求得椭圆与线段的唯一一个交点；当椭圆与线段的位置关系属于第 4）种时，

首先将 $x=x_c$ 或 $y=y_c$ 代入直线方程得到线段上位于椭圆内的一点,该点将线段分成两段,每段都是一个端点在椭圆内另一个端点在椭圆外,然后对这两段分别调用中点分割算法,就得到线段与椭圆的两个交点。

3.2　椭圆与线段交点的一般求法

当椭圆与线段的位置关系属于第 5) 种时,由于椭圆与线段的相交性还没有确定,所以中点分割算法就不适合了。此时,可采用一般方法。

经过两点 $P_0(x_0,y_0)$ 和 $P_1(x_1,y_1)$ 的直线的参数方程为

$$\begin{cases} x = x_0 + c_1 t \\ y = y_0 + c_2 t \end{cases} \quad t \in (-\infty,\infty) \tag{2}$$

式中,$c_1=x_1-x_0$,$c_2=y_1-y_0$。当 $0\leqslant t\leqslant 1$ 时,式(2)表示的是关联两个端点 $P_0(x_0,y_0)$ 和 $P_1(x_1,y_1)$ 的线段 P_0P_1。

将式(2)代入式(1)中得到一个关于参数 t 的二次方程

$$At^2+2Bt+C=0 \tag{3}$$

式中

$$A = a^2c_2^2 + b^2c_1^2$$

$$B = a^2c_2(y_0-y_c) + b^2c_1(x_0-x_c)$$

$$C = a^2(y_0-y_c)^2 + b^2(x_0-x_c)^2 - a^2b^2$$

当式(3)的判别式 $\Delta=B^2-AC\leqslant 0$ 时,二次方程无实根或有两个相等的实根(此时椭圆与直线相切,切点不算交点),椭圆与直线无交点,所以椭圆与线段 P_0P_1 无交点。

当判别式 $\Delta>0$ 时,二次方程有两个不相等的实根

$$t_i = \frac{-B\pm\sqrt{\Delta}}{A} \quad i=1,2 \tag{4}$$

椭圆与直线有两个交点。对于椭圆与线段的第 5) 种位置关系,这里所遇到的情况是:

1) 两个条件 $0\leqslant t_1\leqslant 1$ 和 $0\leqslant t_2\leqslant 1$ 同时成立,此时线段与椭圆有两个交点,$t_i(i=1,2)$ 为交点处的参数,将其代入式(2)就得椭圆与线段的交点,这两个交点之间的线段就是裁剪要求的线段;

2) 两个条件 $t_1<0$ 和 $t_2<0$ 同时成立,或者两个条件 $t_1>1$ 和 $t_2>1$ 同时成立,这两种情形都表明线段是完全不可见的。

实际操作时,如果令

$$t_l = \max\{0,\min\{t_1,t_2\}\}$$
$$t_u = \min\{1,\max\{t_1,t_2\}\} \tag{5}$$

那么若 $t_l<t_u$,则两个交点都在线段上,保留两个交点所确定的可见段,如图 3 中线段②(图中实线为要裁剪的线段,虚线为线段的延长线);若 $t_l>t_u$,则两个交点均不在线段上,线段完全不可见,弃之,如图 3 中的线段①和③。

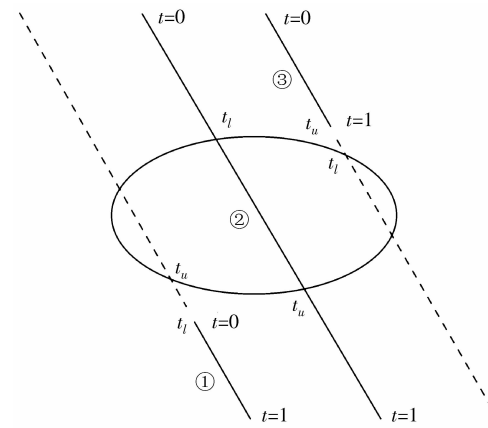

图 3　对第 5)种位置关系的裁剪

Fig. 3　Regarding 5) kind of position relations clipping

于是,对第 5)种位置关系,裁剪算法如下:

```
c1 = x1 - x0;
c2 = y1 - y0;
A = a^2*c2^2 + b^2*c1^2;
B = a^2*c2*(y0 - yc) + b^2*c1*(x0 - xc);
C = a^2*(y0 - yc)^2 + b^2*(x0 - xc)^2 - a^2*b^2;
D = B^2 - AC;
if(D ≤0 )
    线段完全不可见,弃之;
else
    {t1 = (-B - √D)/A;
     t2 = (-B + √D)/A; //显然 t1 < t2
     tl = max{0,t1};
     tu = min{1,t2};
     if( tl < tu)
     线段上从(x(tl),y(tl))到(x(tu),y(tu))之间的一段是可见段;
     else
     线段完全不可见,弃之;
     }
```

4 椭圆对线段的裁剪算法

4.1 算法描述

椭圆对任意线段 P_0P_1 的裁剪算法为

MakeCode(x_0, y_0, &$code_0$);
MakeCode(x_1, y_1, &$code_1$);
if($code_0$&$code_1$ =16)
　　保存线段 P_0P_1;
else if($code_0$&$code_1$ >0)
　　弃掉线段 P_0P_1;
else if($code_0$|$code_1$ ≥16)
调用中点分割算法,求椭圆与线段的唯一一个交点,保存交点与位于椭圆内的端点之间的一段;
else if($code_0$|$code_1$ =3 ‖
　　$code_0$|$code_1$ =12)
　{求线段上位于椭圆内的一点;
调用中点分割算法两次,求椭圆与线段的两个交点,保存两个交点之间的一段;
　}
else
调用对第 5)种位置关系裁剪的算法。

4.2 算法时间复杂度分析

最好情况是椭圆与线段具有第 2)种位置关系,即线段是显然完全不可见的。这种情况算法执行乘除法和开方运算均为 0 次。

最坏情况是椭圆与线段具有第 5)种位置关系,并且线段与椭圆有两个交点,它的两个端点都位于椭圆与其包围盒之间的区域内。这种情况算法先后执行乘除法和开方的次数有:

1) 在求两个端点编码的时候执行了 6 次乘法。

2) 裁剪时执行了 13 次乘除法和 1 次开方运算。

即在最坏情况下,共执行了 19 次乘除法和 1 次开方运算。但是,在各种情况以等概率发生的假设下,这种最坏情况发生的概率小于$\frac{1}{5}\times\frac{1}{2}=\frac{1}{10}$。

4.3 与其他同类算法的比较

本文算法明显比文献[3-4]中的算法要快很多,因为文献[3-4]中的算法是完全通过一元二次方程判别式的符号和求解方程来完成裁剪的。

文献[5]给出的算法虽然完全避免了一元二次方程的求解,但是需要建立 10 余张多行多列的规范化交点表,这些表的建立、查表等操作需要许多额外的时间开销,它们的存储还会浪费很大的空间开销,算法不是很简单。本文给出的算法,因为求端点的编码以及位置关系的判别所涉及的大多数都是位运算,所以花费的额外时间比较少,并且额外只需要两个整型单元存储线段两个端点的编码。

另外,即使椭圆的长、短轴不与坐标轴平行,涉及以椭圆中心为轴的旋转变换也只需对待裁剪线段的两个端点作旋转变换。此时,本算法的效率也要比文献[3-5]中的算法高。

5 结 论

裁剪问题不但是计算机图形学的基本问题,而且也是图像处理、编辑软件等的基础操作。裁剪算法的核心问题是速度问题。提出的椭圆对线段裁剪算法具有以下特点:

采用编码技术可以快速准确判断出不需要裁剪的线段——完全可见或显然完全不可见的线段,避免了不必要的求交费时运算;

采用编码技术还可以快速准确判断出两种椭圆与线段相交的情形,对此采用中点分割算法求交点,不但避免了求解一元二次方程和开方运算,而且由于中点分割算法只用到加法和移位运算,所以适合硬件实现和并行计算;

虽然在万不得已的情况下还是采用了求解一元二次方程的通常做法,但是,在等概率时这种最坏情况发生的几率只为 1/5,所以该算法的裁剪速度提高了很多。实验表明,本算法是快速可行的。

志谢

本文的工作得到黑龙江大学高层次(团队)计划资助。

参考文献(References)

[1] Newman W M, Sproull R F. Principles of Interactive Computer Graphics[M]. New York: McGraw-Hill, 1979: 63-76.

[2] Sproull R F, Sutherland I E. A clipping divider [C]//Fall Joint Computer Conference, Washington: Thompson Books, 1968: 765-775.

[3] Liu Yongkui. A research on graphics clipping algorithms[J]. Computer Engineering and Applications, 2005, (21): 18-23. [刘勇奎. 图形裁剪算法研究[J]. 计算机工程与应用, 2005, (21): 18-23.]

[4] Liu Yongkui. Circular and elliptic clipping windows [J]. Computer Engineering and Design, 1994, (4): 33-37. [刘勇奎. 圆形及椭圆形裁剪窗口[J]. 计算机工程与设计, 1994, (4): 33-37.]

[5] Huang Xinxian, Wu Qingbiao. A fast line clipping algorithm for ellipse windows[J]. Computer Applications and Software, 2005, 22(2): 23-24. [黄新贤, 吴庆标. 一种快速的椭圆形窗口的裁剪算法[J]. 计算机应用与软件, 2005, 22(2): 23-24.]