# 一种基于离散点的透明折射算法

王会芹 庞云阶
(吉林大学计算机科学与技术学院，长春 130012)

**摘 要** 传统真实感图形绘制所采用的造型都是基于多边形的表示方法，由于该方法存在着一些不足之处，为此提出了一种基于离散点的表示方法，该方法首先求出场景中，曲面上的某些离散点，然后将这些离散点排列成若干序列并存储. 同时也提出，并实现了一种基于离散点的透明折射算法，该算法的关键就是如何确定检测线段，由于只有该检测线段上的像素对应序列中的点才有可能在折射线上，这样就抛弃了大量的不需要参与运算的点，因此可减小计算量. 试验结果表明，该表示方法及透明折射算法可以较准确、真实地反映实际场景中的透明现象.

**关键词** 计算机图形学(520·6030) 离散点 深度排序 检测线段 透明 折射光线
**中图法分类号**：TP391.41 **文献标识码**：A **文章编号**：1006-8961(2003)05-0585-05

# A Transparence-refracting Algorithm Based on Discrete Points

WANG Hui-qin, PANG Yun-jie
(*Department of Computer Science and Technology, Jilin University, Changchun* 130012)

**Abstract** The modeling of traditional realistic image generating is on the basis of polygon, which has some deficiency. For example, drawing precision depends on the degree of detail and the method of dissecting, quantity of dissecting and crossing computing should be needed during the generating of realistic image, the procedure of sculpt is complicated, and so on. A representing method based on discrete points is proposed in this paper. First we pick out some points on each curved surface in the scene. Then sort and store them. At the same time, a transparence-refracting algorithm is proposed and realized. The key of this approach is defining the checking line segment. Only the points in the sequences that are corresponding to the pixels on the checking line segment may be on the refracting line and the points in other sequences cannot be on the refracting line. Thus a huge of points that need not participate in computing are abandoned and computing quantity is reduced. The experiment results show that the representing method and the transparence-refracting algorithm can reflect the natural transparent phenomena virtually and accurately. At the same time, this new method can eliminate lines or faces easily and manipulate flexibly, it's data-structure and sculpture are simplified to makes the adding, deleting and modifying curve faces easily. In conclusion the method based on discrete points adapts to the complicated scene and cartoon.
**Keyword** Computer graphics, Discrete points, Depth sorting, Checking line, Transparence, Refracting line

## 0 引 言

实体造型是图形生成的核心和基础，传统的真实感图形绘制都是基于多边形的表示方法，但它存在如下一些缺点：(1)绘制的精度需依赖于多边形表示的细密程度以及多边形剖分时所采用的方法；(2)在真实感图形生成过程中，往往涉及到大量的面片分割和求交运算，因此，计算量和存储量会随着场景复杂度或精度要求的提高而急剧增加；(3)多边形表示方法的造型手续繁杂、数据结构不易操作.

由于基于离散点的表示方法存储的是场景中所有曲面离散点的集合，且将这些点根据它们在屏幕上的投影位置排成若干序列，而每个序列中的点又按照它们在观察坐标系下的值进行深度排序，因此，这种新的绘制方法，不但可以轻而易举地实现消隐，而且造型手续简化、数据结构简单、操作灵活，可以很方便地对场景中的曲面进行增删或修改，故适合

基金项目：高等学校博士学科点专项科研基金项目(20010183041)
收稿日期：2002-06-03；改回日期：2002-12-20

于复杂场景或者动画显示. 同时本文还提出,并实现了基于离散点的透明折射算法.

## 1 基于离散点的表示方法

曲面的表示方法很多,限于篇幅,本文在此不做更多的讨论,只就已经参数化表示的曲面加以分析.

### 1.1 曲面剖分

用均匀参数化方法把空间曲面划分成的若干小多边形[1]如图 1 所示. 这些多边形的顶点将作为离散点,连同其相关属性被存储起来,以形成曲面的初始离散表示.

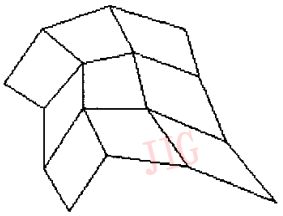

图 1 曲面剖分

### 1.2 从视点出发,对每个曲面求出离散点

一个曲面经透视投影后,在屏幕上可见的点,称为曲面可见点. 就整个曲面而言,其所得到的离散点不但包含了该曲面的所有可见点,而且包含了其他用于阴影、透射等计算所需的离散点信息.

首先,将空间曲面上经过剖分得到的每个小多边形投影到屏幕上,得到相应的投影区域[2],如图 2 所示,$W$ 对应屏幕上的投影区域 $S$.

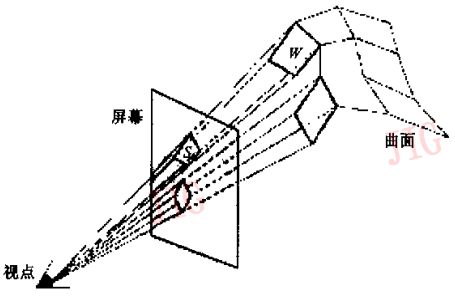


图 2 空间曲面的剖分及投影

然后,对于屏幕上的每个投影区域(如 $S$),逐个像素进行扫描(如图 3 所示),对每个像素点 $s_1(x,y)$,若连接视点与 $s_1$,并将其延长,则其求出的这条直线和空间多边形 $W$ 的交点 $w_1$,即该像素所对应的空间曲面上的位置.

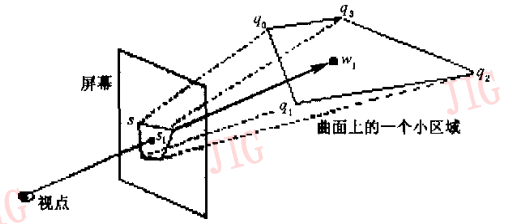


图 3 将屏幕上的像素点进行反投影

点 $w_1$ 处的法向量和颜色均由空间多边形 $W$ 的 4 个顶点 $q_0$、$q_1$、$q_2$、$q_3$ 处的法向量和颜色插值得到. 假设求得点 $w_1$ 的法向量为 $\boldsymbol{G}=(g_x,g_y,g_z)$,视点与点 $w_1$ 的连线的方向矢量为 $\boldsymbol{E}=(e_x,e_y,e_z)$,如果向量 $\boldsymbol{G}$ 与 $\boldsymbol{E}$ 的夹角小于 90°,即 $\boldsymbol{G}$ 与 $\boldsymbol{E}$ 的点乘大于零,则认为眼睛在 $w_1$ 点处看到的是曲面的反面;否则,看到的是正面.

这里之所以采用反投影的方法,而非直接从空间到屏幕的投影,是由于无法确定合适的步长,使得投影到屏幕上的点连续而不重复. 如果步长小,则会导致许多点重复投影到屏幕上同一个像素点的情况;如果步长大,则可能出现屏幕上出现漏洞,不连续的情况. 而这种反投影方法正好解决了这个问题.

### 1.3 存储离散点

本文所采用的存储结构是屏幕上每个像素都对应一个点的序列,序列中存储的是利用前 1.1 和 1.2 节方法得到的空间离散点的坐标、颜色、法向量等属性值,并且每个序列中的点都需要在观察坐标系下进行深度排序,即按照观察坐标系下的 $z$ 值大小进行排序,$z$ 值最小的点排在最前边(如图 4 所示).

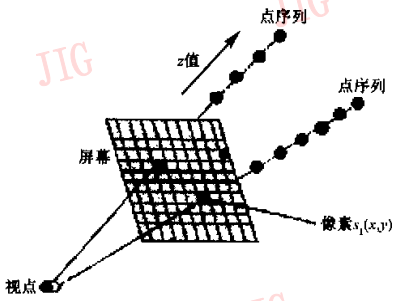


图 4 存储结构

因此,在 1.2 节中每求得空间曲面上的一个点 $w_1$ 时,都要按照观察坐标系下的 $z$ 值大小将其插入到当前像素点 $s_1(x,y)$ 所对应的序列中. 最终目的就是将空间实体上的离散点(如图 5 所示)表示成如

图 4 所示的存储结构. 这样在屏幕上看到的只是每个序列中第 1 个点的颜色值或亮度值，其后面的点虽然是看不到的，但在以后的阴影、透明等计算中却是必须的.

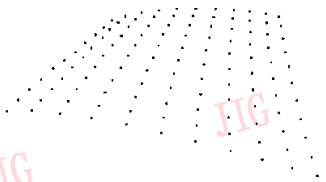

图 5　空间曲面上的离散点

### 1.4　离散点表示方法的特点

(1) 可避免大量的面片求交运算.

在真实感图形生成过程中，由于所有操作都是针对离散点进行的，从而避免了大量的面片求交运算.

(2) 数据结构简单，便于曲面的增删.

当场景中需要增加或删除一个曲面时，原有的存储结构不必改变，只需将属于该曲面的离散点插入到当前存储序列中，或者从当前存储结构中删除即可.

(3) 造型手续简化，适合动画显示.

在场景造型时，由于该方法对各个曲面是单独考虑的，即一个曲面的增删修改对其他曲面没有影响，从而可以在很大程度上保留原来的计算结果，该特点适合于动画显示.

(4) 在场景复杂度或精度要求越高的情况下，该算法有着明显的优越性.

由于离散点的数量只是与场景中所表现实体的数量及大小有关，因此在对计算的精度要求提高时，该算法的存储量和计算量均不变；而在场景的复杂度增加时，该算法不但计算量远小于传统方法，且存储量的增加速度也远小于传统方法的存储量增加速度.

## 2　基于离散点的透明折射算法

在这个算法中，不仅需要逐个扫描屏幕上每个像素点，而且需要从每个像素对应序列中的第 1 个点开始，计算出经过多次折射后，最终反映到屏幕上的颜色值. 如图 6 所示，假设点 $p_1$ 是像素 $s_1$ 对应序列中的第 1 个点，则最终在屏幕上显示的是从点 $p_3$ 发出的光，经过点 $p_2$ 和点 $p_1$ 两次折射而到达眼睛的折射光线. 其中，点 $p_2$ 和点 $p_3$ 分别是像素 $s_2$ 和像素 $s_3$ 对应序列中的点.

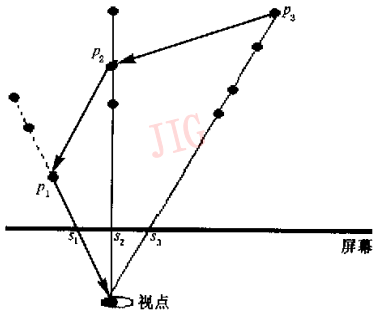


图 6　光线发生折射的过程

### 2.1　定义动态数组

由于求解像素颜色值的过程是光线实际折射的逆过程，因此在计算每个像素之前，需要定义一个动态数组，用于存放求得的一个或多个点，即折射光线逆向过程中所遇到的所有点，每个点包括其法向量、颜色、坐标等属性.

### 2.2　计算折射方向

首先需要按照如下公式计算出折射光线的方向

$$\mathbf{T}=-\eta\mathbf{V}-(\cos\theta_2-\eta\cos\theta_1)\mathbf{N}$$

其中，$\cos\theta_2=\sqrt{1-\eta(1-\cos^2\theta_1)}$，$\eta=\dfrac{\eta_1}{\eta_2}$，其中 $\eta_1$ 为入射光线一边的折射率，$\eta_2$ 为折射光线一边的折射率，$\theta_1$ 为入射角，$\theta_2$ 为折射角，$\cos\theta_1$ 为单位矢量 **N** 与 **V** 的夹角的余弦，**N** 为当前点所在曲面的法向量，**V** 为视线方向，并且 **N** 和 **V** 都是单位向量，计算出的折射线方向 **T** 也是单位向量[3].

### 2.3　确定检测线段

如图 7 所示，当前点 $p_1$ 在屏幕上的投影为 $s_1$，可按照上述方法求出折射线 $\boldsymbol{L}_1$ 及其方向，现在要做的就是找出 $\boldsymbol{L}_1$ 前进方向上遇到的第 1 个点 $p_2$.

如果依次判断所有序列中的每个点，那么计算量将会非常大. 如果只将计算过程中需要检测的线段称为检测线段，那么该步骤的作用就是要确定出一条检测线段. 由于只有该线段上的像素对应序列中的点才有可能在 $\boldsymbol{L}_1$ 上，而该线段之外的其他像素对应序列中的点都不需要参与运算，从而可大大地减小计算量.

这样，首先需要求出一条射线 $\boldsymbol{L}_2$，它经过视点，并且平行于 $\boldsymbol{L}_1$，由于视点坐标和 $\boldsymbol{L}_1$ 的方向矢量都已知，所以这条射线很容易可以求出，然后根据 $\boldsymbol{L}_2$ 与屏幕是否有交点，将具体情况分为如下两种：

(1) $\boldsymbol{L}_2$ 与屏幕相交.

如图 7 所示,当计算出 $L_2$ 与屏幕的交点 $c_1$ 后,则线段 $s_1c_1$ 即为检测线段.

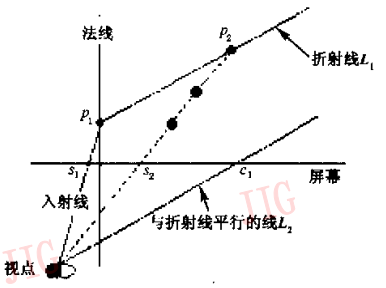


图 7 $L_2$ 与屏幕相交的情况

(2) $L_2$ 与屏幕不相交.

如图 8 所示,在 $L_1$ 上取一点 $Q$,若使其能够投影到屏幕上某点 $b_1$,则需要检测的就是以点 $s_1$ 为端点,且以$\overrightarrow{s_1b_1}$为方向矢量的射线,并需重复检测,一直到屏幕的边界或者遇到某点 $p_2$ 在折射线上为止.

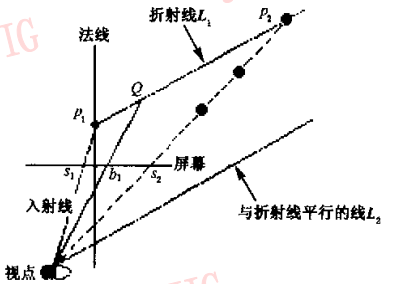


图 8 $L_2$ 与屏幕不相交的情况

在确定检测线段后,对该线段上每个像素位置对应序列中的点,首先求出它与折射线 $L_1$ 的距离 $d$,若 $d$ 小于某阈值(阈值由 2.4 节方法动态确定),则认为该点在折射线上;一旦找到某点(如 $p_2$)在折射线上,则需判断光线是否在该点再次发生折射.如果在点 $p_2$ 发生折射,那么,将 $p_2$ 作为当前点,重复上述过程,直至找到某点在该射线上,且该点是不透明的或者是它所在序列中的最后一个点为止(如图 6 所示,在与像素 $s_1$ 对应的动态数组中,最终得到的点依次为 $p_1$、$p_2$、$p_3$).

### 2.4 动态确定阈值

由于所存储的都是离散点,所以只有先确定一个阈值,才能判断某点(如 $p_2$)是否在折射光线上.如果点 $p_2$ 到光线的距离小于阈值,则认为点在光线上;又由于点与点之间的距离不一定相同,有的地方密集,而有的地方稀疏,因此需要根据点的疏密来动态确定阈值,以使得计算更精确.

如图 9 所示,假设现在要判断点 $p_2$ 是否在折射光线上,需要动态确定点 $p_2$ 处的阈值.具体方法是:首先记点 $p_2$ 与视点的连线为 $l$,视点与其周围的 8 个像素点的连线分别为 $l_1$、$l_2$、…、$l_8$;然后分别计算点到直线 $l_1$、$l_2$、…、$l_8$ 的距离,记为 $d_1$、$d_2$、…、$d_8$;最后判断光线处于哪些射线与之间,假设光线从 $l_1$ 和 $l$ 两条线之间穿过,那么将 $d_1/2$ 作为当前的阈值,如果光线既从 $l_1$ 和 $l$ 之间穿过,又从 $l_3$ 和 $l$ 之间穿过,那么取 $\min(d_1,d_3)/2$ 作为当前的阈值,这样就可以根据阈值和距离来确定点 $p_2$ 是否在射线上.

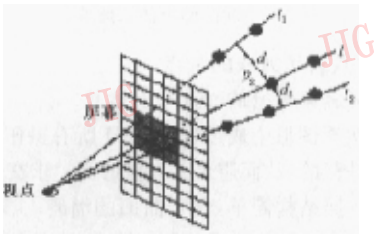


图 9 动态确定阈值

### 2.5 计算当前像素点的颜色值

若动态数组中只有一个点,那么当前像素的颜色即为该点的颜色;否则,从动态数组中最后一个点开始,依次取出两个点(如 $p_3$ 和 $p_2$),然后对其运用公式 $I=tI_2+(1-t)I_3$ 进行明暗度计算.其中 $I_2$ 和 $I_3$ 分别为点 $p_2$ 和点 $p_3$ 处的明暗度,$t$ 为点 $p_2$ 所在物体表面的透明度($0\leq t\leq 1$),$t=0$ 表明点 $p_2$ 所在的物体表面完全透明,$t=1$ 表面表面点 $p_2$ 所在的物体表面完全不透明,这样,其后边的物体对当前点的明暗度不造成任何影响.在该动态数组中,除了最后一个点之外,其他点不会出现 $t=1$ 的情况[4].

总之,这种方法之所以可行是因为所存储的每个点序列中的所有点都在同一条视线上,如图 7 所示,由于折射线是一条以 $p_1$ 为端点的射线,所以经过视点,且与折射线相交的视线只能在入射线与 $L_2$ 所夹区域之内,这个区域之外的所有视线与折射线都不可能相交,其序列中的点就更不可能在折射线上了.由此可见,只需判断检测线段上的像素所对应序列中的点是否在折射线上即可.

## 3 实 验

图 10 至图 13 是采用本文所述方法绘制的图象.其中,图 10 和图 11 是透明折射现象的绘制效果,图 12 和图 13 则反映了复杂场景的绘制效果.

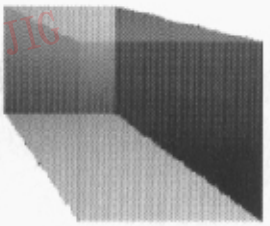

图 10　透明立方体

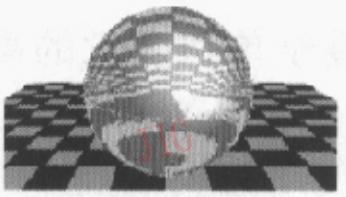

图 11　透明球

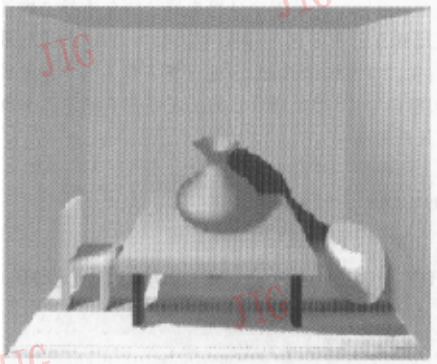

图 12　多实体

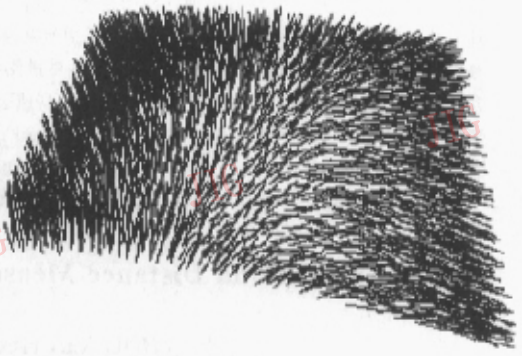

图 13　毛发

## 4　结　语

实验结果表明，使用离散点表示方法不仅可以使复杂的场景造型简单化，而且可以在加密或更改部分数据时，不影响别的数据，而在此基础上提出的透明折射算法思想简单，绘制速度较快，已在不同的场景中取得了很好的效果（图 10 至图 13）. 同时，对基于离散点的真实感图形生成技术的其他方面（如阴影、纹理等），算法也比传统算法有所简化. .

当然，算法还有一些需要改进的地方，比如对简单实体的存储量相对较大、如何与传统方法相结合等，这些问题需要在今后的研究中不断地进行改进.

## 参 考 文 献


1　孙家广. 计算机图形学[M]. 北京：清华大学出版社，1998：309～330.

2　王征旋，钟慧湘，庞云阶. 计算机图形学教程[M]. 长春：吉林大学出版社，1999：31～40.

3　彭群生，鲍虎军，金小刚. 计算机真实感图形的算法基础[M]. 北京：科学技术出版社，1999：125～127.

4　唐荣锡，汪嘉业，彭群生等. 计算机图形学教程[M]. 北京：科学出版社，1999：224～225.


**王会芹**　1977 年生，2002 年获吉林大学计算机科学与技术学院硕士学位. 主要研究方向为计算机图形学、图象处理.

**庞云阶**　1939 年生，吉林大学计算机科学与技术学院教授，博士生导师. 主要研究方向为计算机图形学、图象处理、计算机辅助设计等.