第 33 卷 第 21 期 计 算 机 工 程 2007 年 11 月
Vol.33 No.21 Computer Engineering November 2007
·多媒体技术及应用· 文章编号：1000—3428(2007)21—0226—03 文献标识码：A 中图分类号：TP391.42

采用空间分割的平滑曲面重构算法

顾耀林，袁江琛
(江南大学信息工程学院，无锡 214122)

摘 要：面绘制是科学计算可视化中一个重要的研究方向，移动立方体是实现面绘制的一个重要算法，八叉树是一种有效的表示三维物体的方法，该文在八叉树生成的基础上，提出一种基于空间分割的表面重构算法，将绘制空间分别按 *X* 轴、*Y* 轴和 *Z* 轴进行分割，生成的树的节点个数小于等于 8 个，与八叉树方法相比，减少了所生成叶结点数量，再通过移动立方体算法生成三角面片。三角面片通过平滑处理，提高了图形显示质量。

关键词：空间分割；八叉树；三次线性

Smooth Surface Reconstruction Algorithm Using Space Segmentation

GU Yao-lin, YUAN Jiang-chen
(School of Information Engineering, Southern Yangtze University, Wuxi 214122)

【Abstract】Isosurfaces are common visualization measures in many fields. Marching cubes algorithm is an effective way to render the object surfaces. Octree is an important way to describe 3D objects. By using the octree, a surface reconstruction algorithm based on space segmentation is presented. The space is divided along X, Y, and Z respectively. The nodes of the tree are less than eight. The number of meshes is less than octree. Then by means of marching cube algorithm, triangles are produced. By trilinear method, the triangles are dealt with smoothly. The surface of the object is reconstructed using C++ and OpenGL. In this way, the quality of the algorithm is good.

【Key words】space segmentation; octree; trilinear

曲面重构是三维可视化技术的一个重要方面，三维数据场绘制方法有两种：面绘制和体绘制。面绘制算法首先从三维数据场提取中间几何元素，并将面上的数据属性值映射为可视元素，再用传统的计算机图形学技术实现画面绘制。为了揭示场中数据属性的特征，可视化中绘制的面一般都是等值面。等值面的生成方法很多，主要有基于等值线的生成方法、基于体素的生成方法、基于能量函数的生成方法。其中，移动立方体算法[1]就是基于体素的生成方法。

1 相关工作

1.1 移动立方体算法

MC 算法是一种进行等值面构造与显示的方法，此算法以三维体数据场中由相邻最近的 8 个体元所构成的立方体为最小等值面搜索单元，并根据每个立方体单元各个顶点的情况来决定该立方体单元内部等值面的构造形式。算法要确定等值面立方体的相交情况，然后再移动到下一个立方体，判断等值面是否与立方体相交的基本原则是：立方体的边的一端的值大小等于阈值，而另一端的值小于等值面的阈值。

原始移动立方体算法存在有生成的三角面片太多、二义性、效率不高及冗余等不足之处。文献[2～4]阐述了减少生成的三角面片的数量，提高三维绘制的速度；文献[5～6]着重阐述了如何消除 MC 算法中的二义性问题，生成较原始 MC 算法更为合理的三角面片结构；文献[7～9]着重阐述了如何提高效率的问题。

1.2 八叉树

八叉树是表达三维实体模型或三维图像的一种重要方式，在计算机图形学、计算机视觉、图像处理等方面获得了广泛的应用。构建八叉树的大致过程是首先构造被测曲面的最小外接六面体，并作为八叉树模型的根节点，然后把该六面体按三维方向平均分割成 8 个子六面体，每个子六面体被视为根节点的子节点，由此将构造空间细分为 2 层的子六面体，此子六面体称为根节点的子节点，节点分为 3 类：第 1 类称为黑节点(black)，表示此节点代表的六面体在实体内；第 2 类称为白节点(white)，表示此节点代表的六面体在实体外；第 3 类称为灰节点，表示此节点代表的直六面体一部分在实体内，一部在实体外。对灰节点进一步分割，重复进行直到子节点的边长小于等于给定的精度为止。文献[10～12]对八叉树算法及改进的八叉树算法进了讨论。

2 空间分割

2.1 原理

八叉树方法在对六面体进行分割时按三维方向平均分割成 8 个子六面体，对于大规模数据来说，所生成的节点的数量是比较多的，本文提出的基于空间分割原理生成的树，是在八叉树的分割基础上，将空间分别沿 *X* 轴、*Y* 轴、*Z* 轴 3 个方向进行分割。如果一个节点是灰节点，则先沿 X 轴方向分割，生成两个子节点，判断两个子节点的状态是黑节点、白节点还是灰节点，如果是黑节点或白节点则停止分割，如果是灰节点，则继续沿 *Y* 轴方向进行分割，再判断生成的子节点的状态，如果仍是灰节点，再沿 *Z* 轴方向进行分割。由于分割分别是沿 3 个方向进行的，在分割过程中，只要不是灰节点，该节点的分割过程就结束。与八叉树相比，八叉树被分割的节点有 8 个孩子，而本文提出的基于空间分割生成

作者简介：顾耀林(1948－)，男，教授，主研方向：计算机图形学，虚拟现实；袁江琛，硕士研究生

收稿日期：2006-11-10 **E-mail：**yjc_33@yahoo.com.cn

的树分别沿 *X*, *Y*, *X* 3 个方向依次分割，如果某个方向分割时检测到是黑节点或白节点则不再分割，这样一个节点的孩子小于等于 8 个，减少了节点数，从而减少存储节点的空间，加速算法的执行。空间分割顺序及所生成的树如图 1 所示。

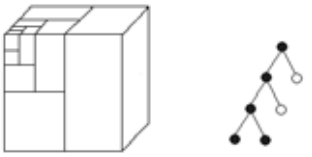


图 1　空间分割顺序及所生成的树

2.2　算法实现

2.2.1　数据结构

本算法在对空间进行分割的过程中，是分别通过*X*轴、*Y*轴、*Z*轴3个方向进行的，每次分割所生成的节点数为2个，并且所生成的节点也是树的结构，因此，数据结构定义如下：

```
typedef struct Tree
{
  float x,y,z;      //(x,y, z)是节点表示的直六面体的顶点坐标
  foat a,b,c;       // a,b,c 分别为沿空间坐标轴 x,y,z 的长度
  int level;        // level 为分割的层数
  int flag;         // flag 为节点的类型：0 表示白节点，1 表示黑
//结点，2 表示灰结点
  struct Tree *child[2];        //指向 2 个子结点的指针，若是叶子
//结点则为空
}
```

2.2.2　树的生成

空间分割过程由根节点分别沿 *X* 轴、*Y* 轴、*Z* 轴方向剖分生成，因此树的建立过程采用递归方法生成一棵树。

```
CreatTree(Tree & T)
{
If(节点为灰节点 and 不满足精度要求)
        沿 X 轴方向将节点分成 2 个子节点；
If(节点为灰节点 and 不满足精度要求)
        沿 Y 轴方向将节点分成 2 个子节点；
If(节点为灰节点 and 不满足精度要求)
        沿 Z 轴方向将节点分成 2 个子节点；
}
```

2.2.3　树的遍历

由于树的生成是按 3 个不方向分割实体得到，因此在对树进行遍历时，通过先序遍历得到的节点顺序即是在分割时的顺序，通过递归方法实现对树的遍历。

```
TraverseTree(Tree T, int   Visit ) )
{
If(节点为灰节点 and 不满足精度要求)
{ 将访问标志 Visit 设置为已访问；
        TraverseTree(child[1]);
        TraverseTree(child[2]);
 }
 else if(节点为灰节点 and 满足精度要求)
        该节点通过 MC 算法产生三角面片；
}
```

3　三角面片的平滑显示

采用空间分割产生的六面体是进行 MC 算法的基础，这些六面体的大小不一，因此，利用 MC 算法所产生的三角面片的大小差异也较大，对于一些较大的三角面片，会显得较为粗糙，因此，在利用 MC 算法产生的三角面片的基础上，对三角形利用三次线性插值进行精细化，使产生的重构曲面更加光滑、逼真。

给定一个三次线性标量场

$$S(x,y,z)=(1-x)(1-y)(1-z)C_{000}+(1-x)(1-y)zC_{001}+(1-x)y(1-z)C_{010}+(1-x)yzC_{011}+x(1-y)(1-z)C_{100}+x(1-y)zC_{101}+xy(1-z)C_{110}+xyzC_{111} \tag{1}$$

其中，C_{ijk} $(i,j,k\in\{0,1\})$是六面体顶点的标量值。等值面上的点(x,y,z)都符合如下公式：

$$S(x,y,z)=r \tag{2}$$

其中，r 是一给定的阈值。

3.1　点在等值面上的投影

给定一个点 $a=(x_a,y_a,z_a)^{\mathrm{T}}$，计算由 a 出发的一条射线和等值面的交点。在特定的情况下，这条射线与某一条坐标轴平行，问题就简化为一个线性等式。$Px(a)$为直线 $a+\lambda(1,0,0)^{\mathrm{T}}$ 与等值面的交点，$Py(a)$为直线 $a+\lambda(0,1,0)^{\mathrm{T}}$ 与等值面的交点，$Pz(a)$为直线 $a+\lambda(0,0,1)^{\mathrm{T}}$ 与等值面的交点，其中 $Pz(a)$的情况如式(3)所示：

$$Pz(a)=\begin{bmatrix} r-c_{000}(1-x_a)(1-y_a)-c_{010}(1-x_a)y_a \\ -\dfrac{c_{100}x_a(1-y_a)-c_{110}x_ay_a}{(c_{001}-c_{000})(1-x_a)(1-y_a)} \\ xa.ya,+(c_{011}-c_{010})(1-x_a)y_a \\ +(c_{101}-c_{100})x_a(1-y_a) \\ +(c_{111}-c_{110})x_ay_a \end{bmatrix}^{\mathrm{T}} \tag{3}$$

称 $Px(a)$，$Py(a)$和 $Pz(a)$分别是点 a 沿 x，y，z 轴方向在等值面上的投影，这表明对于一个给定的点，可以在 3 个简单的方向上构造出等值面上的 3 个点，如图 2 所示。

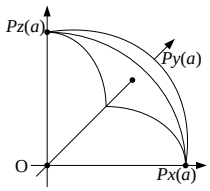


图 2　等值面上的 3 个点

对于一个给定的点，有且仅有一个等值面经过，可以计算该点的法向量 $n(a)$：

$$n(a)=(S_x(x_a,y_a,z_a),S_y(x_a,y_a,z_a),S_z(x_a,y_a,z_a))^{\mathrm{T}} \tag{4}$$

其中，S_x, S_y, S_z是由式(1)定义的部分偏移量。

3.2　线段在等值面上的投影

给定一个直线段 $x(t)=(1-t)a+t(b)$，那么可以得到该直线段沿x, y, z方向投影到等值面上所得到的曲线$P_x(t)$, $P_y(t)$, $P_z(t)$。如图 3 所示，给出了$P_z(t)$的情况。

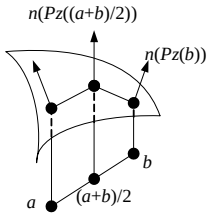


图 3　曲线*Pz(t)*

曲线$P_z(t)$可以表示为

$$Pz(t)=\frac{\sum_{i=0}^{3}\omega_i b_i B_i^3(t)}{\sum_{i=0}^{3}\omega_i B_i^3(t)} \tag{5}$$

其中，$B_i^3(t)$ 是 Bernstein 多项式，

$\omega_0 = z_n(P_z(a))$， $\omega_1 = \frac{4}{3}z_n(P_z(\frac{a+b}{2})) - \frac{1}{3}z_n(P_z(b))$，

$\omega_3 = z_n(P_z(b))$， $\omega_2 = \frac{4}{3}z_n(P_z(\frac{a+b}{2})) - \frac{1}{3}z_n(P_z(a))$，

$b_0 = P_z(a)$， $b_3 = P_z(b)$，

$$b1 = \begin{bmatrix} (1-\frac{\omega_0}{3\omega_1})xb_0 + \frac{\omega_0}{3\omega_1}xb_3 \\ (1-\frac{\omega_0}{3\omega_1})yb_0 + \frac{\omega_0}{3\omega_1}yb_3 \\ (1+\frac{\omega_0}{3\omega_1})z_{Pz}(\frac{a+b}{2}) - \frac{\omega_0}{3\omega_1}zb_3 \end{bmatrix}, b_2 = \begin{bmatrix} \frac{\omega_3}{3\omega_2}xb_0 + (1-\frac{\omega_3}{3\omega_2})xb_3 \\ \frac{\omega_3}{3\omega_2}yb_0 + (1-\frac{\omega_3}{3\omega_2})yb_3 \\ (1+\frac{\omega_0}{3\omega_2})z_{Pz}(\frac{a+b}{2}) - \frac{\omega_0}{3\omega_2}zb_0 \end{bmatrix} \quad (6)$$

同样也可计算得到$P_x(t)$和$P_y(t)$的三次多项式。

3.3 三角形在等值面上的投影

给定一个三角形 $x(u,v,w) = ua + vb + wc$，a, b, c 是三角形的顶点且 $u+v+w=1$。$P_x(x(u,v,w))$, $P_y(x(u,v,w))$，$P_z(x(u,v,w))$ 是三角形沿 x, y, z 轴方向在等值面上的投影，三者都可以三次多项式来表示，如图 4 所示。

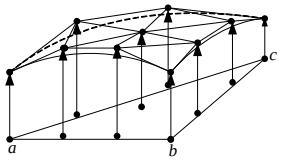


图 4　三角形在等值面上的投影

$P_z(x(u,v,w))$ 可表示为 Bézier 三角形，用式(7)来表示：

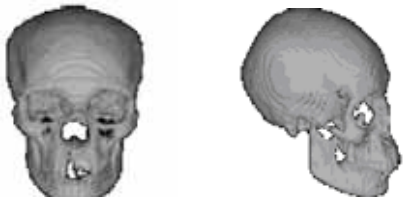
$$P_z(x(u,v,w)) = \frac{\sum_{i+j+k=3}\omega_{ijk}b_{ijk}B_{ijk}^{\ 3}(u,v,w)}{\sum_{i+j+k=3}\omega_{ijk}B_{ijk}^{\ 3}(u,v,w)} \quad (7)$$

其中，Bézier 点和其边界曲线上权值可以由式(6)计算得到。

4　实验结果与分析

实验使用的数据源是 128 层 128×128 的三维数据场，灰度值大于 140 为骨骼，在 Windows 环境下，使用 OpenGL 绘图软件包绘制。

利用该算法产生的图形，由于生成的节点数量随着分解层数的增加而较少，通过 MC 算法生成的三角面片数量相应地也减少，所以执行较快，同时具有一定的交互性，可以进行视角的设置。实验结果如图 5 所示，图 5(a)是头骨正面图，图 5(b)是旋转了一定角度后头骨的侧面图。从实验结果看，该算法能较好地完成面绘制。



(a)　(b)

图 5　实验结果

八叉树与该算法方法所生成的节点对比如表 1 所示，可以看出本算法能有效地减少所生成的节点数量，从而有效地提高了 MC 算法进行面绘制的执行速度。

下面来分析该算法的空间复杂度。用八叉树存储三维数据场信息所占用的存储空间主要与实体的尺寸、复杂程度以及所采用的空间分辨率有关，分层越少，存储所占用的空间就越少。从表 1 可看出，使用八叉树算法和使用本算法在节点深度为 2 层时，节点的数量是相同的，由于本算法所采用的数据结构中增加了一个标志，占有一定空间，八叉树算法中的一个节点所需要的存储空间为 26B，本算法中一个节点所需要的存储空间为 28B，因此在这种情况下，本算法所占用的存储空间比八叉树所占用的存储空间再大一些。但在实际应用中，节点深度越低，空间分割越粗糙，所要显示的图形越不精确。当节点深度等于 3 时，本算法比八叉树产生较少的节点数，但由于标志位占用一定的空间，因此比八叉树算法多占用存储空间。当节点深度大于 3 时，本算法所占用的存储空间少于八叉树算法，并且随着深度的增加，这种优势越明显。

表 1　本算法与八叉树方法比较

节点深度	区域大小	八叉树节点数量	八叉树存储量/B	本算法节点数量	本算法存储量/B
0	128*128*128	1	26	1	28
1	64*64*64	8	208	8	224
2	32*32*32	64	1 664	64	1 792
3	16*16*16	512	13 312	496	13 888
4	8*8*8	4 096	106 496	3 464	96 992
5	4*4*4	32 768	851 968	28 826	807 128
6	2*2*2	262 144	6 815 744	183 026	5 124 728
7	1*1*1	2 097 152	54 525 952	805 208	22 545 824
总计		2 396 745	62 315 370	1 021 093	28 590 604

5　结束语

八叉树作为三维造型的一种方法，在实际应用中有着重要的意义，本文提出的基于空间分割原理是在八叉树的基础上演化而来的，减少了八叉树所生成的节点数量，节省了存储空间，同时由于生成的节点数少，通过 MC 算法生成的三角面片数量减少，加速了算法的执行。但在所生成的树的结构上有一定的缺点，即所生成的树的深度较大，这是本算法需要改进的地方。

参考文献

1 Lorensen W E, Cline H E. Marching Cubes: A High Resolution 3D Surface Construction Algorithm[J]. ACM Computer Graphics, 1987, 21(4): 163-168.

2 徐　毅, 李晓梅, 许润涛. 对体可视化 Marching Cube 算法的改进[J]. 计算机工程, 1999, 25(11): 52-54.

3 谢小棉, 李树祥, 江贵平, 等. 基于 MC 的医学三维等值面的平滑与归并[J]. 中国图像图形学报, 2001, 6(A)(8): 806-809.

4 薛　强, 蔡文立, 石教英. Marching Boxes: 一个多精度等值面提取算法[J]. 计算机辅助设计与图形学学报, 1998, 10(1): 7-15.

5 Nielson G N, Hamann B. The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes[C]//Proc. of the Visualization'91. [S.l.]: IEEE Computer Society Press, 1991: 83-90.

6 王正山, 吕理伟, 顾耀林, 等. 基于改进 MC 算法的三维表面重建[J]. 微电子学与计算机, 2005, 22(9): 3-6.

7 马仁安, 张二华, 杨静宇, 等. 步进立方体算法的 SOB 数据结构的改进[J]. 中国图象图形学报, 2003, 8(A)(11): 1309-1313.

8 徐晓玲, 李现民, 李桂清, 等. 体素重建中的快速移动立方体方法[J]. 系统仿真学报, 2001, 14(4): 509-513.

9 熊邦书, 何明一, 俞华景. 基于空间连通性的快速曲面重建算法[J]. 系统仿真学报, 2005, 17(1): 75-78.

10 Wilhelms J, Van Gelder A. Octrees for Faster Isosurface Generation [J]. ACM Computer Graphics, 1992, 11(3): 201-227.

11 Boada I, Navazo I. An Octree Isosurface Condification Based on Discrete Planes[C]// Proceedings of the 17th Spring Conference on Computer Graphics'01. 2001.

12 马文华. 三维物体的空间二值矩阵表示到线性八叉树表示的转换[J]. 宁夏工学院学报(自然科学版), 1996, 8(2): 44-49.