第37卷 第21期
Vol.37 No.21

计 算 机 工 程
Computer Engineering

2011年11月
November 2011

·工程应用技术与实现· 文章编号：1000—3428(2011)21—0214—03 文献标识码：A 中图分类号：TP391.9

三角形二叉树在数控加工仿真中的应用

裴洪虎[1,2]，郭锐锋[1,2]，邵志香[1,2]

(1. 中国科学院研究生院，北京 100039；2. 中国科学院沈阳计算技术研究所高档数控国家工程研究中心，沈阳 110171)

摘 要：三角片离散法在动态显示时需要渲染大量的三角片，难以保证数控加工仿真的实时性与真实性。为此，利用三角形二叉树 LOD 模型实现三角片的分裂与合并，通过误差二叉树、制分裂及强制合并算法控制三角片的数量。实验结果表明，与三角片离散法相比，该方法的仿真效果较好。

关键词：数控加工仿真；三角片离散法；细节层次；三角形二叉树；误差树

Application of Binary Triangle Tree in Numerical Control Machining Simulation

PEI Hong-hu[1,2], GUO Rui-feng[1,2], SHAO Zhi-xiang[1,2]

(1. Graduate University of Chinese Academy of Sciences, Beijing 100039, China;
2. National Engineering Research Center for High-End CNC, Shenyang Institute of Computer Technology, Chinese Academy of Sciences, Shenyang 110171, China)

【Abstract】Triangular facet dispersion method needs to display enormous amount of triangles. Aiming at this probolem, this paper adopts the binary triangle tree Level of Detail(LOD) model, with the LOD method to control the Split-up of triangle, with the error tree、split and merge forcibly to control the number of triangles. These guarantee the real-timing and authenticity of the simulation. This paper implements the algorithm by programming. Compared with Triangular facet dispersion method by experimental data, the superiority of this algorithm is verified.

【Key words】Numerical Control(NC) machining simulation; triangular facet dispersion method; Level of Detail(LOD); binary triangle tree; variance tree

DOI: 10.3969/j.issn.1000-3428.2011.21.073

1 概述

数控加工仿真是利用计算机对数控加工过程的模拟，其核心问题是仿真的实时性与真实性。数控加工仿真的算法总体上分为实体仿真算法和离散仿真算法 2 类。实体仿真算法是最为精确的仿真算法，能提供最完整的几何信息，缺点是计算量非常大，不能保证实时性。离散仿真算法包括视点离散法、三角片离散法[1]等。离散法实现简单，目前多数对仿真算法的研究都集中在离散法上。

三角片离散法先将工件上表面离散为均匀点阵，再将这些点阵连接成三角网格。程序运行时，按照刀具路径不断修改上表面点阵的高度，再对三角网格进行渲染，以此仿真材料的去除过程。三角片离散法要选择适当的网格离散密度，密度越高，三角片越多，模拟的精度越高，但所消耗的时间就越长。例如：将工件上表面分成 500×500 个小矩形，每个小矩形分成 2 个三角片，则上表面有 50 万个三角片，要达到 15 f/s 的速率(自然的动态特性要求每秒生成和显示 30 帧图形画面，至少不能少于 10 帧)，则显卡需要每秒渲染 750 万个三角片，这远远超出了一般显卡的处理能力。

三角片离散法生成的网格没有进行优化或者近似优化，工件上表面平坦区域跟凹凸区域的三角形密度相同，这必然要生成大量的三角形，影响渲染的实时性。采用细节层次(Level of Detail, LOD)[2]技术能较好解决图像质量和渲染的实时性之间的矛盾。本文采用三角形二叉树[3-4]LOD 模型算法，利用 Z-MAP[5]模型表达工件，通过三角形二叉树 LOD 模型控制三角片的分裂与合并。工件表面起伏大、特征明显的地方，三角片密度大；平坦的地方三角片密度较少，从而减少加工表面三角片的数量，保证显示质量和实时性。

2 三角形二叉树 LOD 模型

2.1 LOD 技术

1976 年，Clark 提出了 LOD 模型的概念，认为当物体覆盖屏幕较小区域时，可以使用该物体描述较粗的模型，并给出了一个用于可见面判定算法的几何层次模型，以便对复杂场景进行快速绘制。

LOD 技术是一种有效的图形生成加速方法，它是解决硬件性能不够发达的技术产物，可有效解决渲染速度瓶颈问题。LOD 技术通常根据物体模型的节点在显示环境中所处的位置和重要度，决定物体渲染的资源分配，降低非重要物体的面数和细节度，从而获得高效率的渲染运算。

2.2 三角形二叉树

2.2.1 三角形二叉树的概念

直角顶点和斜边中点的连线可以把一个等腰直角三角形分割成 2 个全等的等腰直角三角形，这种分割过程可以无限地进行下去。每一个三角形都是 2 个较小三角形的父三角形，所有的三角形构成了一个二叉树的层次结构。

基金项目：国家科技重大专项基金资助项目“开放式数控系统支撑技术创新平台建设”(2011ZX04016-071)

作者简介：裴洪虎(1986－)，男，硕士研究生，主研方向：数控加工仿真；郭锐锋，研究员、博士、博士生导师；邵志香，博士研究生

收稿日期：2011-05-10 **E-mail：**peihh@sict.ac.cn

图 1 显示了三角形二叉树的层次划分情况，每个较高层次，都将三角形一分为二。

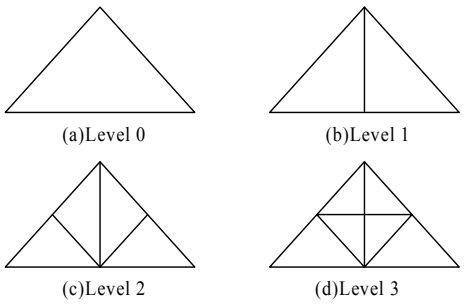


图 1　三角形二叉树的层次

2.2.2　三角形二叉树的数据结构

本文将一个三角形二叉树节点保存在一个 TriTreeNode 结构中，其中包含 5 种基本关系，定义如下：

```
struct TriTreeNode{
TriTreeNode *LeftChild;
//当前三角片的左孩子
TriTreeNode *RightChild;
//当前三角片的右孩子
TriTreeNode *BaseNeighbor;
//当前三角片的基邻居
TriTreeNode *LeftNeighbor;
//当前三角片的左邻居
TriTreeNode *RightNeighbor;
//当前三角片的右邻居
};
```

TriTreeNode 节点邻接关系如图 2 所示。

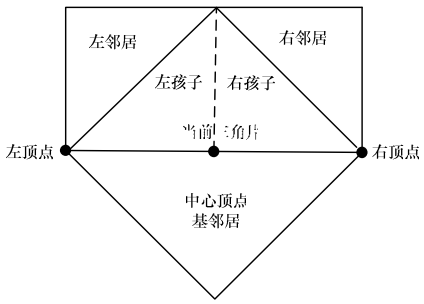


图 2　三角形二叉树的数据结构

2.3　三角形二叉树与 LOD 的结合

三角形二叉树 LOD 模型，即用不同层次结构的三角形二叉树定义不同层次的 LOD 模型，通过控制三角形二叉树的层次结构来实现对 LOD 模型的层次细节控制。对于较高的 LOD 层次，采用层次划分较多的三角形二叉树，对应较高的图形显示精度；对于较低层次的 LOD，采用层次划分较少的三角形二叉树，减少实际渲染的三角片数量。

3　算法步骤

本文提出的算法如下：

(1)毛坯表面初始化。以正方形毛坯为例，将毛坯上表面分解为 $M\times M$ 大小的 Z-MAP 区域，每个区域构造一个三角网格。将每个区域块中所有面片互连，也就是设置三角形二叉树左右孩子节点、左、右邻居节点和基邻居节点。

(2)确定切削区域并计算误差 *Variance* 值。在每次读入一行 NC 代码后，求得刀具扫描体在 *XY* 面的投影，即切削区域。然后计算该区域内所有三角片的 *Variance* 值。

(3)三角片的合并与分裂。根据 *Variance* 值，修改落在刀具扫描体内所有初始顶点的高度信息，对误差值小预定界限的三角形进行合并，并对与刀具扫描体相交的三角片进行分裂，直到满足一定精度为止。这一步是整个算法的核心。

(4)三角片的渲染。使用三维图形库渲染所有分裂出的三角片。

3.1　毛坯表面的初始化

由于在进行加工仿真之前，不知道零件模型的详细情况，无法针对某个加工细节进行比较细致的离散，因此本文采用均匀离散的方法：将工件表面按一定尺寸分成若干等大的区域，一个区域由左右 2 个等腰直角三角形构成，每个三角形形成一个独立的三角形二叉树，每个区域由 2 个二叉树组成，当需要更高的细节时，再做进一步划分。

3.2　切削区域的确定

如图 3 所示，刀具的刀心从 *O*1 点移动到 *O*2 点，矩形 *ABCD* 是扫描体在 *XY* 平面上的包围盒。显然，只有在包围盒内的点才有可能是切削点。

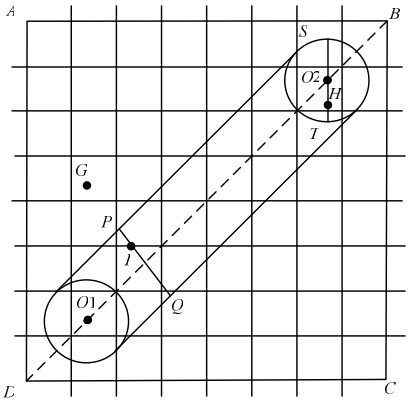


图 3　切削区域的确定

具体步骤如下：

(1)首先判断包围盒内的点到直线 *O*1*O*2 的距离是否大于刀具半径 *R*。显然与直线 *O*1*O*2 的距离大于刀具半径 *R* 的点肯定不在扫描体中，如 *G* 点。

(2)判断点在直线 *O*1*O*2 上的投影点对 *O*1*O*2 的分比 λ；若 $0\leqslant\lambda\leqslant 1$，则该点就应该在扫描路径上，如 *I* 点。

(3)若 $\lambda<0$，则判断该点在 XY 平面上到 *O*1 点的距离；如果 $\lambda>1$，则判断该点到 *O*2 点的距离。如果该距离小于或等于刀具半径，则是扫描路径起始端上的点，如 *H* 点；否则，该点不在扫描体内。

3.3　几何误差的定义与计算

本文定义一个 *Variance* 值来决定节点是否需要分割以及需要分割的程度。图 4(a)中 *D* 点处的 *Variance* 值的计算公式如下：

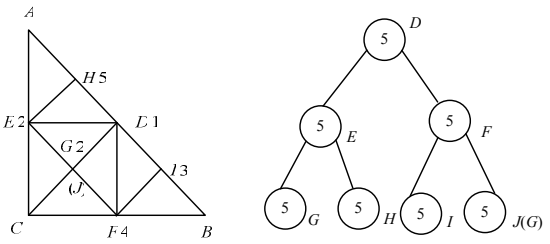
$$Variance(D)=\left|Z_{\text{center}}-\frac{(Z_{\text{left}}+Z_{\text{right}})}{2}\right|$$

其中，Z_{left}、Z_{right} 是 *A*、*B* 两点的高度值；Z_{center} 是斜边中点 *D* 处的高度值。

但如果仅以 *D* 点处误差值作为判定是否分割三角片，会出现一些问题。例如，在图 4(a)中，*D*、*E*、*F* 点处的误差值分别为 1、2、4。假设误差值 1 满足精度要求，而 2、4 都不满足精度要求。如果仅以 *D* 的误差值作为判断，那么 *D* 点满足精度要求，所以，将不再对三角片 *ABC* 分割。这样用一张三角片来表示原来模型显然是不够精确的。

因此，考虑应以该三角片及其以下各层三角片斜边中点

的误差值的最大者来判断该三角片是否分割，需要计算出一棵误差树，这颗二叉树上的节点存储的是该点对应三角片及其以下各层三角片斜边中点的误差值的最大者，如图 4(b)所示。



(a)各斜边中点误差 (b)误差树

图 4 各层三角片斜边中点误差及误差树

3.4 三角片的分裂

如图 3 所示，当刀具从 $O1$ 加工到 $O2$ 时，如果区域的三角片与刀具扫描体相交并且超过预定的几何误差，则执行分裂操作，在分裂过程中，同时修改三角片的邻接关系。

当分裂一个三角片时，由于邻近三角片所处的层次不同，会导致在邻接处出现裂缝。如图 5 所示。在红色区域左边有 2 个小三角形△AOD 和△COD，右边为一个大三角形△ABC，AC 边是它们的公共边。在左边，AC 边被分成了 AD 和 DC 2 段，分别包含在△AOD 和△COD 中，渲染时将作为两段分别进行渲染；在右边，AC 边只包含在△ABC 中，直接被渲染。如果 D 点的高度值不是 A、C 两点高度的平均值，那么两边的三角形在AC边处必然不会重合，在AC边将产生裂缝。

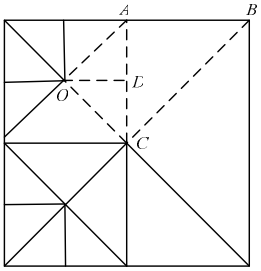


图 5 裂缝的产生

一个节点和它的邻节点只存在 2 种关系：共直角边关系(左右邻节点)和共斜边关系(下邻节点)。当它与它的下邻节点呈相互下邻关系时，看作一个钻石结构。在钻石结构上分割一个节点可以很容易的镜象到其他节点，因此在网格上不会出现裂缝。遵循以下规则，可避免裂缝的产生：(1)若节点是钻石结构的一部分，那么分割该节点和它的下邻节点。(2)若节点是网格的边，那么只分割这个节点。(3)若节点不是钻石结构的一部分，那么强制分割它的下邻节点。

强制分裂是一个递归调用的过程，直至遇到一个结点是钻石结构的一部分或在网格边缘为止，其过程如下：首先检查当前结点是否为钻石结构的一部分，如果不是，则对当前结点的基邻居执行分裂操作；然后继续原先的分裂操作，对基邻居执行分裂操作做同样的检查，递归执行上面的过程，直到满足递归返回的条件为止，如图 6 所示。

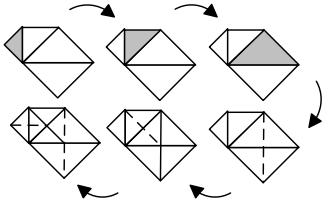


图 6 强制分裂过程

3.5 三角片的合并

合并是与分割相对应的三角形操作，它只对有且仅有一层子三角形的非叶子结点三角形进行，对于那些误差值小于一定界限的三角形，应当删除它的子三角型，用高度的估计值代替真实值，并且重新调整三角形之间的关联，使原来与子三角形的关联与本身相关联。关于分裂、合并的一个重要依据是任意一个三角划分可以从任意其他的三角划分经过一系列分裂、合并操作得到。

为保持网格的连续性并避免裂缝，所有邻居三角形必须进行合并。图 7 描述了一个强制合并过程。需要对左图中的三角形 T 进行合并，经过递归操作得到右图中的结果。可以看到，三角形 T 的所有子孙节点都被合并，且三角形 T 的所有邻居节点也被强制合并。

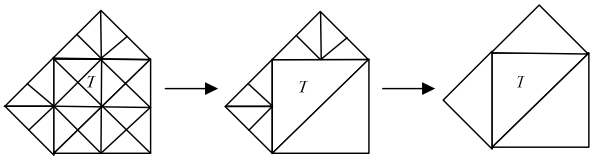


图 7 强制合并过程

4 实验结果与分析

实验以加工奥迪汽车标志为例，测试本文三角形二叉树 LOD 算法的性能，并与三角片离散法做比较，图 8 为网格模式下的渲染图。

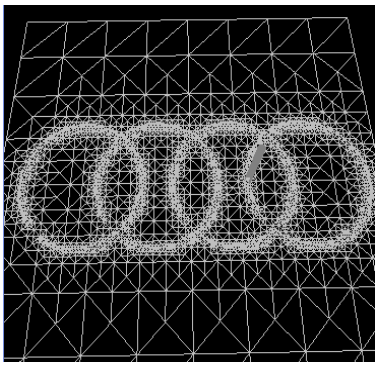


图 8 网格模式渲染图

本文的实现平台是 Windows XP，VC++6.0，采用 OpenGL 图形库[6]。实验所用的机器配置为：Inter Core Duo CPU 2.6 GHz，1 GB Physical Memory 集成显卡。毛坯上表面大小为 512 mm× 512 mm，球头刀的半径为 4 mm，误差取 0.01，NC 文件共 2 400 行。

表 1 为采用离散三角片法(分将毛坯上表面划分为 256×256 和 128×128 两种情况)和三角形二叉树 LOD 模型法的数据对比。可以看出，三角形二叉树 LOD 法需渲染的三角片数为 4 939，所用时间为 41 s，平均帧率为 57 f/s；而均匀三角片离散法采用 256×256 大小划分时，需渲染的三角片数多达 131 072，所用时间为 464 s，平均帧率仅为 4 f/s。采用三角片离散法需渲染的三角片数量是采用三角形二叉树 LOD 模型法的 26.5 倍，帧率则约为它的 1/16。

表 1 均匀三角片离散法和 LOD 方法的比较

方法	开始时间/ms	结束时间/ms	消耗时间/s	平均帧率/(f·s^{-1})	三角片总数
三角片(256×256)	30 427 210	30 891 428	464	4	131 072
三角片(128×128)	30 264 350	30 398 986	134	17	32 768
LOD	29 838 458	29 880 067	41	57	4 939

(下转第 219 页)