论文编码：TP39

首都师范大学
Capital Normal University

首都师范大学本科毕业设计论文

三维模型重建理论与方法的研究

——三维模型光顺算法的研究

Research on Surface Reconstruction Theory and Method

——Research on Triangular Mesh Smoothing

论文作者_____

院　　系　信息工程学院

专　　业　计算机科学与技术

学　　号_____

指导教师_____

完成日期　2013 年 4 月 10 日

首都师范大学信息工程学院

摘　　要

随着计算机辅助几何设计理论和技术的发展和应用，计算机图形学的应用越来越广泛。其中，对点云数据进行网格重建是现在计算机图形学研究的热点问题之一。通常点云数据由3D扫描仪扫描实物模型所得，采集到的点云数据往往不包含法向信息，但采样点的法向量信息对于重建过程非常重要，所以对没有法向量输入的点云进行重建是本文的研究重点。三维重建可应用于逆向工程、虚拟现实、模具重建、文物修复、医学等众多领域。

由于测量工具或人为因素等原因，扫描得到的点云数据会伴有噪声，因此在对三维数据进行相关处理之前必须先对其进行光顺、去噪处理。目前三角网格模型是表示复杂三维模型的主流方式之一，光顺就是消除网格噪声的同时，还要保持它的拓扑信息和几何特征，避免出现过光顺和体积收缩现象。

本文分别对缺失法向信息的点云重建问题和三角网格光顺问题进行深入研究，提出了一些创新的算法。主要内容包括：

- 提出一种基于八叉树的区域增长三维重建算法。主要利用八叉树结构进行点云搜索速度优化，利用PCA计算平坦区域选择种子三角形，制定加入点准则使其逼近最优解，通过判断加入点的类型进行拓扑重建。可以针对缺失法向量信息的点云数据，并且不需要点云数据进行预处理操作，重建效果良好。
- 提出一种保持特征的自适应各向异性三角网格光顺算法。对顶点在法向和切平面法向同时进行移动，利用1环邻域计算期望法向量，使用期望法向量通过函数计算偏移量，由雨伞操作算子计算自适应偏移系数加速光顺效果。可以有效去除网格噪声并且保持模型的原有特征。
- 介绍了作者所开发的三维模型重建与光顺的数字几何处理系统。该系统提供了一个包括点云重建、三角网格光顺和三维模型显示等技术在内的数字几何处理框架，并着重介绍和实现点模型的表现、八叉树建立及搜索、平坦度计算、拓扑重建以及计算期望法向量等本文提出的新算法及其实现。

关键词：曲面重建；光顺；八叉树；区域增长；各向异性

首都师范大学信息工程学院

Abstract

With the development and application of the theory and technology of CAD, computer graphics has been more and more widely used. Among them, the surface reconstruction of point cloud is now one of the hot issues of the researches. The point could data is obtained from 3D scanner to scan physical models without the normal information, while the normal information plays a very import role in reconstruction progress. Surface reconstruction of point cloud without the normal information is the main idea of this paper. And surface reconstruction is used in many fields of reverse engineering、virtual reality、CAD\CAM、restauro、medical and so on.

There exist some noises in the scanned point cloud for the scanner tools or human factors, and other reasons. The denoising should be done before digital geometry processing. Triangular mesh is one of the most popular ways to illustrate 3D objects, triangular mesh smoothing can remove noises and keep the topology information and geometric characteristics without over smoothing or volume shrinkage.

The innovative algorithms have been proposed after the in-depth study of triangular mesh smoothing and surface reconstruction of points without the normal information. Included:

- A new mesh growing surface reconstruction algorithm is proposed based on octree. The main use of the octree structure is point cloud search speed optimization. We use PCA to calculate the most flat area and select seed triangle. And make some principles to ensure the growing of points be close to the optimal. Finally, we judge the type of points to determine topology reconstruction. The algorithm can deal with point cloud data without the normal information and pre-progressing and the reconstruction results are good.
- An adaptive and anisotropic smoothing algorithm is proposed for removing noise and keeping feature of triangular mesh model. We move the vertices in both the normal and tangential. First calculate the expected normal vector by 1 ring triangles and obtain the offset. Then umbrella operator is used to calculate the adaptive coefficient to accelerate the mesh smoothing. The algorithm can effectively remove the noises and keep the original features of the model.
- A digital geometry processing system of surface reconstruction and mesh smoothing is implemented and introduced. The system actually is a framework of surface reconstruction, mesh smoothing and 3D geometry representation. The main components of the system

首都师范大学信息工程学院

include various novel algorithms on representation, establishent and search of octree, flatness calculation, topology reconstruction and the expected normal calculation.

Keywords: surface reconstruction; smoothing; octree; mesh growing; anisotropic

首都师范大学信息工程学院

目　录

第 1 章 绪论 1
1.1 三维数据的获取和表示 1
1.2 三维模型处理研究现状 2
1.2.1 三维点云重建 2
1.2.2 三角网格光顺 3
1.3 三维模型处理的应用与意义 3
1.4 本文工作 4
第 2 章 基于八叉树的区域增长三维重建算法 6
2.1 预备知识 6
2.1.1 K 近邻 6
2.1.2 主成分分析（PCA） 6
2.2 算法原理 6
2.2.1 建立八叉树结构 6
2.2.2 选择种子三角形 7
2.2.3 制定加入点准则 8
2.2.4 拓扑重建 9
2.3 算法流程 10
2.4 实验结果与分析 11
2.5 本章小结 15
第 3 章 保持特征的自适应各向异性三角网格光顺算法 17
3.1 算法原理 17
3.1.1 优化期望法向量 17
3.1.2 优化顶点偏移量 18
3.1.3 优化偏移系数 19
3.1.4 更新顶点位置 20
3.1.5 光顺误差分析 20
3.2 算法流程 21
3.3 实验结果与分析 21
3.4 本章小结 25
第 4 章 三维重建与光顺系统软件实现 26
4.1 系统软件主要功能 26
4.2 系统软件的显示实现 27
4.3 系统软件的算法实现 28
4.3.1 重要的数据结构 28
4.3.2 重要的函数 29
4.4 算法伪代码 30
4.4.1 三维重建算法 30
4.4.2 三角网格光顺算法 31
4.5 系统软件演示 32
4.6 本章小结 37
第 5 章 总结与展望 38
5.1 本文总结 38
5.2 未来研究展望 38

首都师范大学信息工程学院

参考文献..... 39
致　谢..... 41

首都师范大学信息工程学院

第 1 章 绪论

1.1 三维数据的获取和表示

随着逆向工程、文物修复、虚拟现实、工业设计、医学辅助诊断等应用需求的推动，更多的科学家和学者对数字几何处理越来越青睐。获取一个三维模型的方式主要分为正向工程和逆向工程。正向工程主要是直接对模型进行设计，开发流程为“构思，设计，产品”。然而生活中很多实物如文物、人脸等不能够直接用简单的曲面进行表示，于是，逆向工程在人们生活中越来越重要。逆向工程的基本思想是通过已有的实物模型，精密地去测量其外形坐标数据，再通过测量的数据构建出实物的几何模型。

目前有基于单张或者多张图片的重建工作研究，依赖采集设备的相关参数，对实物进行数学建模，得到相关的点云数据。这样的方法可以对简单的模型进行重建，对于复杂的模型很难精确其细部特征，多应用于医学领域。也有基于图片流之间的特征匹配，进行大规模的重建建筑场景等研究。基于图片的特征匹配无法精确的表示一个实物模型的三维几何信息。

随着电子技术和经济的发展，目前最常用的获取点云数据的方式是利用三维激光扫描仪扫描实物模型，三维扫描仪会自动测量实物表面的很多点，点集中通常包括点的坐标信息，有些较为先进的扫描仪可以得到点的颜色信息。点云数据就是三维点坐标的集合，这些点用 X，Y，Z 坐标定义。点云数据之间没有内在联系，通常情况下，一个模型的点云集合表示该模型的一个流形表面。点云数据由于测量工具或者环境因素等原因会含有噪声，并且扫描得来的信息不包括法向量信息。

三角网格是表示实物模型的主流方式之一。其中，三角网格是由一系列的点、边和面构成。三角网格是将点云数据重建所得，如图 1.1 所示，(a)图是 egea 的实物模型，为一个人的头部的立体模型，(b)图所示为该模型扫描后得到的散乱点云数据，而(c)模型是对点云数据进行重建后得到的三角网格模型。



(a) 实物模型　　(b) 点模型　　(c) 网格模型

图 1.1 egea 模型

首都师范大学信息工程学院

1.2 三维模型处理研究现状

三维点云重建是计算机图形学研究的热点问题之一，并且能够应用于图形处理、虚拟现实、模具重建、逆向工程、医学、游戏等众多领域。三维点云重建的关键在于寻找点云之间的拓扑关系，构造三角网格模型逼近原始曲面，重建算法的好坏依赖于重建结果的准确性、速度和鲁棒性。

由于测量数据精度或者重建算法缺陷等原因，三维模型对应的三角网格模型可能存在噪声，在对几何模型进行处理之前必须进行去噪光顺操作。去噪光顺的数据对象可以是点数据，也可以是经过三角化的网格。本文研究的是对三角化网格进行光顺。

1.2.1 三维点云重建

重建算法主要分为两类：隐式重建和显式重建。

隐式重建的基本思想是寻找隐式曲线方程 $f(p)$，使得 $f(p)=0$，其中 p 是原有点云或者部分原有点云。常见的隐式重建算法有：由 Ohtake 等[1]提出了多层紧支撑径向函数（*RBF*）插值算法。假设点云数据分布在一个曲面上，对点云进行由粗到精分层，然后由粗到精进行插值。该算法快速，但需要点的法向量作为输入数据，计算量较大。Kazhdan 等[2]将点云重建转换为空间泊松问题，该算法抗噪声能力较强，但也需要输入点云法向量信息。林裕旭等[3]提出一种“场拟合”的隐式曲面重建算法，将三维曲面表达为极性场的零值面，使用贪婪算法求解粒子的分布。该算法能保持细节，鲁棒性较好，但计算复杂，消耗较多内存。

显式重建主要分为基于 *Voronoi*/*Delaunay* 重建算法和区域增长算法。*Voronoi* 图的主要思想是为点创建区域，使其区域只受核心点影响。由于按照 *Voronoi* 多边形定义求解 *Voronoi* 多边形在计算机中实现较为困难，借助 *Delaunay* 三角化实现是一个研究热点。由 Amenta 等[4]提出 *Crust* 算法，这种算法是利用密集采样点云和极点对中轴线进行良好逼近，它的时间复杂度是 $O(n^2)$，其中 n 是点云和极点的数量。为加快运行速度和节省内存，Amenta 等[5]随后提出 *Cocone* 算法，最坏时间复杂度是 $O(m^2)$,其中 m 是点云的数量，该算法重建效果较好，但是难以处理带洞和非均匀采样的点云数据。由 Dey 等[5]提出 *Tight Cocone* 算法，该算法可以处理带洞点云数据，并且不需要引入额外的点。由 Dey 等提出 *Robust Cocone* 算法[6]和 *Super Cocone* 算法[7]，分别适用于处理带噪声的数据和处理大量点云数据重建。由 Dey 等[8]提出基于八叉树结构的 *Cocone* 算法，可处理大型点云数据。

区域增长法是先选定种子三角形，然后对前沿边使用一些准则进行增长。常见的算法有 Bernardini 等[9]提出滚雪球（*Ball Pivoting Algorithm*）算法，该算法在对前沿边增长的时候，利用半径为 r 的球按照 *BPA* 原则进行滚动建立网格。这种算法对含噪声的点云数据处

肯恩师范大学信息工程学院

理鲁棒性较差。Li 等[10]提出一种权值驱动方法（*PDA*），在选择加入点时将符合条件的点按照权值排列，选择权值最大的点加入。由 Angelo 等[11]提出一种新型快速重建算法，该算法提出一种 *G2S* 标准，对均匀分布的点云数据可良好逼近二维 *Delaunay* 算法，但该算法对非均匀点云数据处理鲁棒性不强。上述两者都使用哈希表存储点云及 K 邻近数据，对于大量点云数据如何解决冲突是一个重要问题。

1.2.2 三角网格光顺

三角网格光顺算法主要有两大类：局部迭代法和全局优化法。局部迭代法的思想是更新顶点位置时利用其原始网格邻域内顶点信息。Field[12]提出了经典的拉普拉斯算法，网格顶点朝着它的邻域顶点重心方向移动。该算法简单、快速，但是随着迭代次数的增加，容易出现模型收缩、过光顺现象。Taubin[13]将信号处理知识引入到去除噪声中，提出了一种带两个相反符号的加权拉普拉斯算法，该算法有效抑制高频的同时，轻度增强了低频，有效地改善了三维模型体积收缩的问题。Desbrun 等[14]提出平均曲率流算法，每个网格顶点以平均曲率的速度朝着其法向量方向移动。但由于该算法的非凸组合性，在一些特殊情况下，网格容易出现畸变。Ohtake[15]提出了一种顶点沿法向和切向移动的各向异性光顺算法，不仅能够有效去除噪声，也保持了网格的特征，但需要人工设置一些交互参数。赵晔等[16]提出一种基于均值漂移的特征保持的网格光顺算法，该算法通过双边滤波获得真实坐标。章雅娟[17]等提出一种改进的三角网格模型光顺算法，通过设定不同的滤波因子，对顶点的位置进行调整。

Yagou[18]将图像处理中均值滤波和中值滤波方法应用到图形学领域，均值滤波不能够保持原图形的尖锐特征，中值滤波可以保持尖锐特征，但是增加了网格的不规则化。唐[19]等改进双边滤波，提出了一种基于 CUDA 的三维点云快速光顺算法，但计算量较大，使用 GPU 技术辅助。全局优化法中最常见的是能量函数法，这种方法首先提出一个能量函数，通过求解该函数来更新网格中顶点位置。但因计算量大，没有得到广泛应用。2010 年，张冬梅等提出线性求解离散二次能量函数的光顺算法[20]，复杂度低，还能避免收缩现象，但仍需大量计算。

1.3 三维模型处理的应用与意义

三维模型的重建和光顺可应用于逆向工程、虚拟现实、模具重建、文物修复、医学、游戏制作等众多领域，是计算机图形学研究的热点问题。

逆向工程：传统的几何造型技术对三维模型进行直接设计，缺点在于很难经过直接的

首都师范大学信息工程学院

设计表现现实生活中已经存在的实物，比如古董、雕像等。随着应用需求的推动，逆向工程应运而生。逆向工程就是扫描已有的实物模型，通过扫描得到点云集合，对点云进行曲面重建逼近原始曲面。重建后的三角网格模型可能存在噪声，为了美化和精确模型的几何特征，在对模型进行数字几何处理之前需要进行三角网格光顺。

虚拟现实：虚拟现实是指利用电脑模拟产生一个三维空间的虚拟世界。常见的一些应用包括虚拟展馆、虚拟旅游、电影制作、建筑视景和地形地貌模拟等。通过对大量图片或者大型建筑进行扫描，利用图像处理的配准技术，对场景模型进行重建，达到真实的视觉效果。

模具重建：对工业生产的较为复杂的模具进行扫描获取点集，通过三维重建技术可获得到模具的几何特征，应用于模具的批量生产和设计。

文物修复：由于一般不能够对文物使用接触的方式获得它的三维点云，通常定期使用三维扫描仪进行扫描得到点云数据，然后进行模型重建。当文物出现损坏时，可以通过以往的模型信息进行文物的修复和保护工作。

医学：随着技术的发展，医疗设备的应用越来越广泛。在使用仪器设备进行诊查时，医务人员只能通过二维图像进行疾病的判断，如何将二维图像转换成三维形状也是研究的热点问题。医护人员通过三维形状对疾病进行诊断，可提高诊断的正确率。

游戏制作：游戏是人们生活中必不可少的部分。在游戏制作中，使用了大量的计算机图形学、计算机视觉技术。尤其是制作游戏场景，利用三维重建技术和图像处理技术，可以获取和实现大型场景的漫游。

1.4 本文工作

在前人的基础上，为了降低三维模型重建的时间复杂度，不使用三维模型的法向量作为输入数据，本文提出一种基于八叉树的区域增长三维重建算法。利用八叉树结构，对空间数据点的搜索进行加速。在平坦区域选取初始三角形，制定加入点的准则，使其逼近最优解。将三角形的三条边作为前沿边进行拓扑重建。

为了快速有效去除噪声，保持原模型的尖锐特征，并且减少人工交互的步骤和计算量，本文提出一种自适应各向异性光顺算法。首先根据 1 环域内的三角面法向量来计算其期望法向量，由期望法向量得到各顶点偏移量。最后计算雨伞操作算子[21]，得出偏移系数。本文算法引入了偏移系数，更能有效快速地光顺去噪。

本文各章节的内容安排如下：

第一章 绪论：主要介绍了三维点云数据重建和三角网格光顺的国内外研究现状，近

首都师范大学信息工程学院

些年的三维重建算法和三角网格光顺算法，总结归纳了经典算法的基本原理及优缺点，阐述了三维模型处理的应用与意义。

第二章 基于八叉树的区域增长三维重建算法：主要介绍本文创新的基于八叉树的区域增长三维重建算法，针对无法向量信息的点云数据进行重建，采用八叉树结构进行空间搜索加速，使用区域增长方法，利用主成分分析计算平坦区域并选择初始三角形，对选择加入点准则进行细化，使得重建结果逼近最优。该算法重建效果良好，具有鲁棒性。

第三章 保持特征的自适应各向异性三角网格光顺算法：主要介绍本文创新的保持特征的自适应各向异性三角网格光顺算法，对顶点在法向和切平面法向同时进行移动，利用 1 环邻域计算期望法向量，使用期望法向量通过函数计算偏移量，由雨伞操作算子计算自适应偏移系数加速光顺效果。可以有效去除网格噪声并且保持模型的原有特征。

第四章 三维重建与光顺系统软件实现：主要介绍了三维重建与光顺系统软件的技术难点和实现过程，介绍和实现三维模型的显示、八叉树建立及搜索、平坦度计算、拓扑重建以及计算期望法向量等本文提出的新算法及其实现。

第五章 总结与展望：对本文研究内容做了总结，并对今后的研究工作和研究趋势进行展望。

首都师范大学信息工程学院

第 2 章 基于八叉树的区域增长三维重建算法

2.1 预备知识

2.1.1 K 近邻

K 近邻是一个样本在特征空间中的 K 个最相似的样本。在点云重建中，K 近邻指的是在三维空间中距离该点最近的 K 个点。一个点和其 K 近邻可以看作是一个小类别，粗略的代表了该类的性质。搜索一个点 K 近邻比较耗时，如果使用暴力求解法的话时间复杂度为 $O(n^2)$。搜索空间数据常用的数据结构为哈希表、八叉树、*K-D* 树等。

对于一个点云模型，如果采样足够充分，意味着一个采样点的 K 近邻可以粗略的估计为一个二维平面，而这个二维平面等同于原曲面在该采样点处的切平面，同时也是该采样点处的重构曲面。

2.1.2 主成分分析（PCA）

主成分分析是一种统计方法，它用来对多变量表示数据点集合寻找尽可能少的正交矢量表征数据信息的特征。它的主要方法是通过对协方差矩阵进行特征分解，来得到数据的主成分（特征向量）和它们的权值（特征值）。常用于高维数据降维，并且保留主要的信息。

在三维模型重建过程中，利用 PCA 可以求得采样点的切平面和法向量信息。首先通过采样点及其 K 近邻求得协方差矩阵，对协方差矩阵进行特征分解后，得到特征向量和对应的最小特征值。其中，该最小特征值对应的特征向量可以粗略估计为采样点的法向量。

2.2 算法原理

本文重建算法的核心思想是利用八叉树结构，对空间数据点的搜索进行加速。对三维点云数据选取初始三角形，制定加入点的准则，使其逼近最优解。将三角形的三条边作为前沿边进行拓扑重建。

2.2.1 建立八叉树结构

八叉树是一种描述三维空间的树状数据结构。八叉树的每个节点表示为一个长方体的体积元素，每个节点有八个子节点，一般以中心点作为节点的分叉中心。首先读取点云数据，点云总数为 N，标记点云在 XYZ 轴上的最大、小值，然后计算出包围盒的中心点坐标、宽度、高度和长度。八叉树可以按照层数为 H 或叶子节点中最多包含点云数量为 K 进行分

首都师范大学信息工程学院

割，若按照前者进行分割，得到的叶子节点的大小、体积都是相等的，计算简单，但是点云分布可能不均匀。若按照后者分割，得到的叶子节点中点云的数量相对平均，节省空间，但是各个叶子节点的体积不同，会给计算带来一定困难。

本文中选择第二种方法，即按照叶子节点中最多包含点云数量为 K 进行分割，这样得到的点云分布相对均匀，对于区域增长算法中平坦度计算有一定的好处。在实验中，我们选择 K=48 。

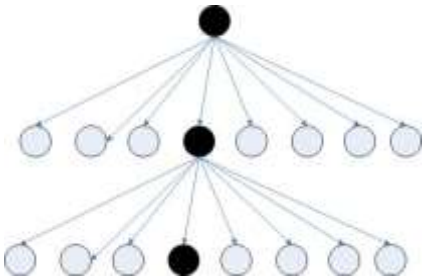


图 2.1 八叉树结构

区域增长算法的时间受空间搜索点的速度影响，常见的空间搜索数据结构包括：哈希表、四叉树、*R* 树、*K-D* 树、八叉树等。其中哈希表是线性存储方法，但是对于密集度较高的点云数据，很难找到合适的哈希函数和处理冲突处理方法。对于八叉树结构来说，如图 2.1 所示，包围盒的总数量 M 和八叉树层数 l 满足：$8^l = M$,其中，当 $l >= 8$ 时，$M >= 16777216$，盒子的数量 M 已经很大。如图 2.2 所示，在查询时采用张永玉[22]等人的编码方法，对每个立方体按照 *Morton* 码进行编码。即对一个编码为 00 的节点继续分割为 8 个子节点，它们的编码依次为 000，001，002，003，004，005，006，007。查找一个点的最坏时间复杂度是 $O(1+K)$,其中 K 是一个盒子里面点的总数。要远远小于普通数组存储平均时间复杂度的 $O(N/2)$，可存储的点的数量是 $K*M$ 。

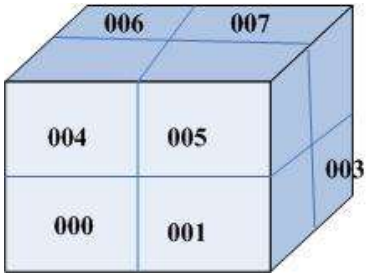


图 2.2 *Morton* 编码示意图

2.2.2 选择种子三角形

选择种子三角形的方法有多种，由 Lin 等[23]提出的方法是：首先在点云中寻找点 p ，点 p 满足在 Z 轴上的值最大,然后选择离 p 点最近的点 q 作为第二个点，以 pq 为轴， pq 的

首都师范大学信息工程学院

长度 l 为直径和高作为一个圆柱体，不断扩大 l,直到点云模型中的点包含到圆柱体中，最后寻找一个点 r,使得 $|pr|+|pq|+|qr|$ 最小，即 pqr 为种子三角形，该算法符合 *Delaunay* 三角剖分原则，但是实际应用比较复杂，并且选入的 p 点有可能是噪声点。陈金锐[24]提出在平坦区域内寻找种子三角形，在平坦区域寻找一个靠近中间的点 p，选择离 p 点最近的两个点 q 和 r 作为第二个点和第三个点组成种子三角形 pqr。该算法需要对每个点进行搜索 K 近邻，时间消耗较多。本文对文献[24]提出的寻找平坦区域方法进行改进，使得搜索点的时间减少。本文算法在选择种子三角形时考虑两个因素：粒子密度和平坦度。在搜索候选点时，首先对该候选点所在最小包围盒中的点的数量进行判断，大于等于 K 个点的才进行计算平坦度，不但能够减少计算量，对干扰噪声也有过滤作用。最后对符合要求的候选点进行计算，按照公式（2.1）[24]进行计算平坦度。

$$k(p_i)=\frac{1}{K}\sum\frac{fabs(dot((p_{ij}-p_i),v))}{\|p_{ij}-p_i\|} \tag{2.1}$$

其中，v 是点 p_i 邻域协方差矩阵 $M(p_i)$ 最小特征值对应的特征向量。点 p_i 邻域协方差矩阵 $M(p_i)$ 由公式（2.2）求得：

$$M(p_i)=\sum_{j=1}^{K}(p_{ij}-p_{ic})\otimes(p_{ij}-p_{ic}) \tag{2.2}$$

其中 p_{ic} 是点 p_i 的 K 邻域的算数平均值，见公式（2.3）：

$$p_{ic}=\frac{1}{K}\sum_{j=1}^{K}p_{ij} \tag{2.3}$$

如果采样点充分，季振方[25]提出可以把最小特征值对应的特征向量估计为这个采样点的法向方向，因为这个点的 K 近邻的伸展度在它的法向方向上最小。对于越平坦的区域，$dot((p_{ij}-p_i),v)\to 0$，平坦度越小，反之，平坦度越大。在实验中，我们取 K=20，平坦度最大的点满足 $\{p_i \mid k(p_i)=\min(k(p_j)),j\in S\}$，其中 S 是所有符合要求的点云数据。得到平坦度最大的点后，选择与其相邻最近的两个点作为第二个点 q 和第三个点 r，组成种子三角形 pqr。

2.2.3 制定加入点准则

在选定种子三角形后，确定选择加入点的准则非常重要，它直接影响着重建的效果。文献[10]提出的 *PDA* 加入点准则，首先去掉不符合条件的点，然后对剩下的候选点计算权值并按权值由大到小排列，最后选择权值最大的点作为加入点，这种方法对于不同的模型可能对应不同的权值系数。文献[11]提出了 *G2S* 方法，这种方法保证在一个三角面片的外接圆内没有其他点存在，该方法更好的保证了重建后的质量。

首都师范大学信息工程学院

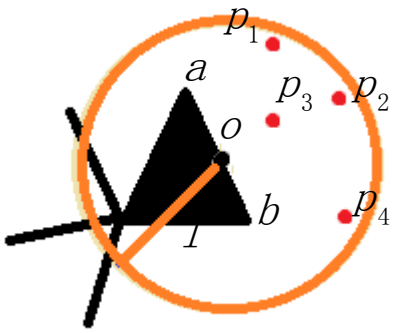


图 2.3 加入点示意图

本文采用算法和文献[11]主要思想是类似的，即保证在一个三角面片的外接圆内没有其他点存在。算法的具体步骤如下：

1) 确定搜索范围：如图 2.3 所示，寻找前沿边 ab 中点 o 所在最小包围盒，以中点 o 作为中心点，半径 $r = l * k$ 的球为搜索范围，其中，l 是前沿边 ab 的长度。其中 k 是一个自定义的正整数，k 的值越小，重建后的结果更精细，但是 k 的值过小，会导致重建不完整。

2) 对落入搜索区域内的点进行筛选：a）加入点必须在前沿边的生长方向，否则有可能导致一边多用的情况。生成的新的三角形的法向量与原三角形的法向量的夹角要在$[-\alpha, \alpha]$之间，在实验中，α 选择在$[45°, 90°]$，不满足条件的点淘汰。b）新形成的三角形应该避免狭长的情况，否则不利用下一步的拓扑重建。分别计算新的点形成的两条边与前沿边的夹角，其中夹角应该 β 范围内，实验中 β 选择为$[25°, 135°]$。 c）计算剩余的点到中点 o 的距离 d_{tmp}，若 d_{tmp} 小于当前最短距离 $d_{\min}$，更新 $d_{\min}$ 的值和对应的点 p。

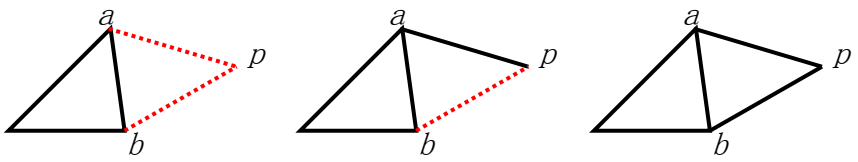
3) 判断是否继续搜索加入点：若只在最小包围盒内搜索加入点，结果会造成结果偏差和错误。若对八叉树中所有叶子节点进行搜索，结果正确但是速度会降低。本文借用文献[26]中搜索最近 K 邻近点算法，将其进行修改应用到搜索加入点中。首先计算中心点到长方体 6 面的最短距离 $dist$，如果 $dist < d_{\min}$，跳到当前包围盒的上一层，继续从步骤 2 开始执行。如果 $dist >= d_{\min}$，结束搜索，选择 $d_{\min}$ 对应的点 p 作为加入点。

4) 如果没有找到合适的点，说明这条前沿边是一个边界。

对于上述图 4 而言，最终选择的点应该是 p_3。

2.2.4 拓扑重建

首都师范大学信息工程学院



(a) Join (b) Link (c) Fill

图 2.4 网格拓扑

首先介绍数据结构删除队列 *Map*，是用来存储已经被两个三角形共享的边。查找一条边是否为删除边的时间复杂度是 $O(1)$。在拓扑重建时，首先将种子三角形的三条边按顺时针顺序放入到前沿边队列中，弹出一条前沿边 *ab* 后，首先判断 *ab* 是否在删除边 *Map* 中，如果不在，按照步骤 3 进行选择加入点 *p*，若不存在符合要求的点 *p*,将 *ab* 放入到边界队列中，若存在点 *p* 则对加入点 *p* 的类型进行判断。点 *p* 有两种类型，一种是未被使用过，即点 *p* 不属于任何前沿边。第二种是被使用过，即该点已经属于某条前沿边。

判断 *p* 点的类型：

1） 如图 2.4(a)所示，若 *p* 是未被使用点，直接加入点 *p*，并且执行 *Join* 操作[10]，即产生一个新的三角形 *apb* 并加入到三角形索引表中，并且将新产生的两条边 *ap*, *pb* 加入到前沿边队列中。

2） 若 *p* 是被使用过的点，判断的 *p* 连接情况：

a) 如果 *p* 与 *a*, *b* 都相连，如图 5(c)所示，执行 *Fill* 操作[10]，即产生新的三角形 *apb*，加入到三角形索引表中。然后将 *ap*, *pb* 加入到删除边 *Map* 中去。

b) 如果 *p* 与 *a*, *b* 中的一个点相连，如图 5(b)所示，执行 *Link* 操作[10]，即产生新的三角形 *pba*,加入到三角形索引表中。然后将已经存在的那条边 *ap* 加入到删除边 *Map* 中，再将新的一条边 *bp* 加入到前沿边队列中去。

c) 如果 *p* 与 *a*, *b* 都不相连，如图 5(a)所示，进行 *Join* 操作[10]，产生新的三角形 *apb*，加入到三角形索引表中。将新产生的两条边 *ap*, *pb* 加入到前沿边队列中去。

最后将 *ab* 放入到删除边 Map 中，弹出 *ab* 继续下一条的前沿边的拓扑，直到所有的前沿边都弹出队列，拓扑重建结束。

拓扑重建后，可以通过边界队列中的边界，进行洞的检测和补洞。

2.3 算法流程

本文算法输入的数据类型为点云数据，即点的三维坐标的集合。主要流程为：

1) 读取点云模型数据，然后建立八叉树结构，在建立八叉树的同时，将点云数据

首都师范大学信息工程学院

放入到相对应的最小包围盒中，在实验中，每个包围盒的最大点数为 48。

2) 对每个包围盒中满足一定密度的点云数据进行计算平坦度，用于确定种子三角形的第一个点 p。然后选择与 p 最为接近的两个点 q 和 r 作为第二个点和第三个点组成种子三角形 pqr。

3) 确定加入点准则，即选择加入点的规则。基本思想是尽量靠近 *Delaunay* 三角剖分原则和结果。在选择点时，对点有一个过滤的过程，主要考虑的是组成的新三角形 pqr 的形状还有与原三角形的前沿边的延伸的方向是否一致。

4) 最后进行拓扑工作，对前沿边按照加入点准则进行选择点。对新组成的三角形按照边的属性进行分类，对不同类型的三角形进行不同的操作，直到所有的前沿边都拓扑结束。

2.4 实验结果与分析

运用本文算法，在主频为 2.10GHz，内存 4.00GB（2.96GB 可用）的 Pentium Dual-Core CPU 的计算机上用 Microsoft Visual Studio 2010 和 OpenGL 图形库进行仿真实验。

选取经典模型 loop、egea、foot、rocker-arm 、fandisk_mc、face-YO、horse、standford_bunny 等进行测试。实验数据及结果如表 2.1 所示，实验重建结果如图所示。

表 2.1 实验数据及结果

实验模型名称	顶点数	生成面数	实验模型名称	顶点数	生成面数
loop	1440	3000	egea	8268	18486
foot	10010	22516	rocker-arm	10044	22068
fandisk_mc	11984	26518	face-YO	13746	28045
horse	19851	41911	standford_bunny	34834	72413

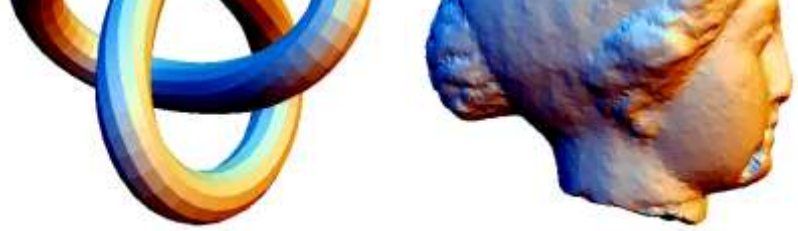
（模型资料来源：http://shapes.aimatshape.net/和 http://www.lodbook.com/models/）

由表 2.1 可以看到，重建的模型点的数量由 1000 到 40000 之间不等，包含了不同级别顶点数目的模型，实验覆盖的模型较。

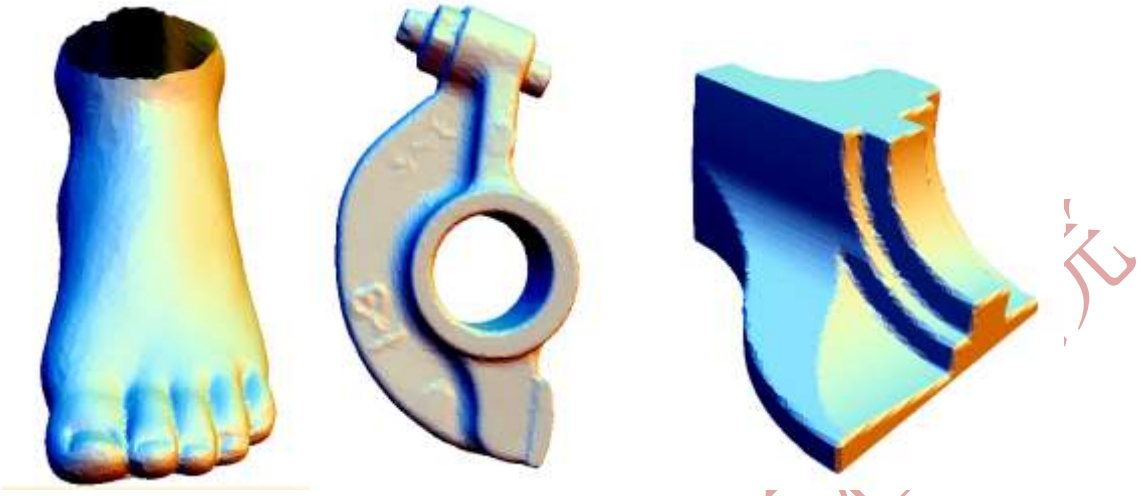
1) 不同几何特征模型

如图 2.5(a)-(h)所示，本文算法在处理不同的模型有较好的鲁棒性，对于模型不同的几何特征，重建后的结果表面较为光滑，说明重建的细度和精度较好。重建后的模型产生碎片很少，也表明该算法对重建的质量保证较好。而由图 2.6 和图 2.7 所示，对于一个模型中存在不均匀的采样率时，本文算法产生的结果拓扑结构规则整齐，模型的各个部分平滑连接，证明本文算法具有较好的鲁棒性。

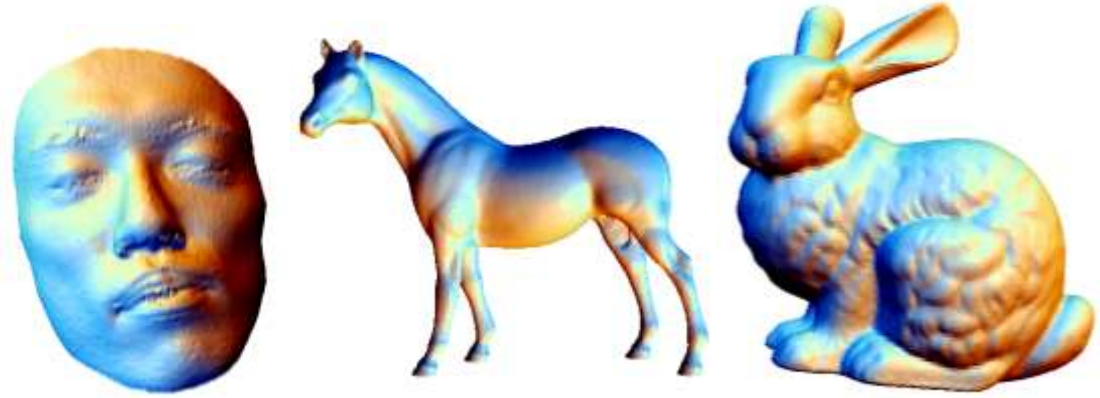
首都师范大学信息工程学院



(a) loop (b) egea

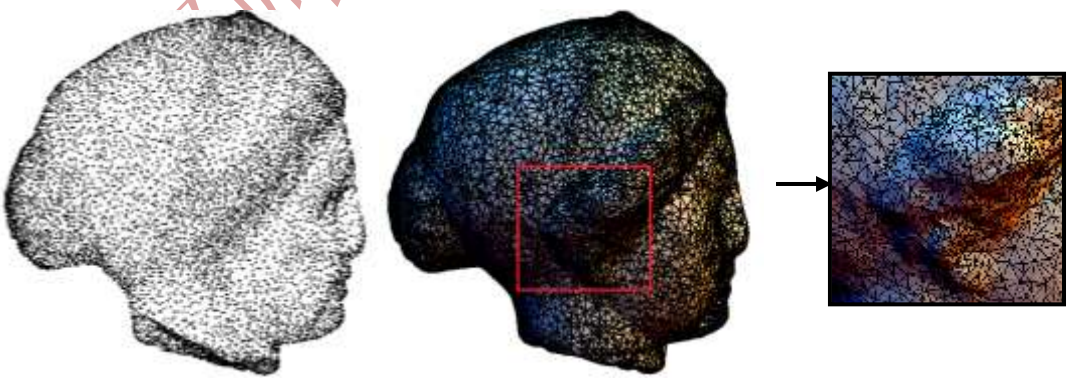


(c) foot (d) rocker-arm (e) fandisk_mc



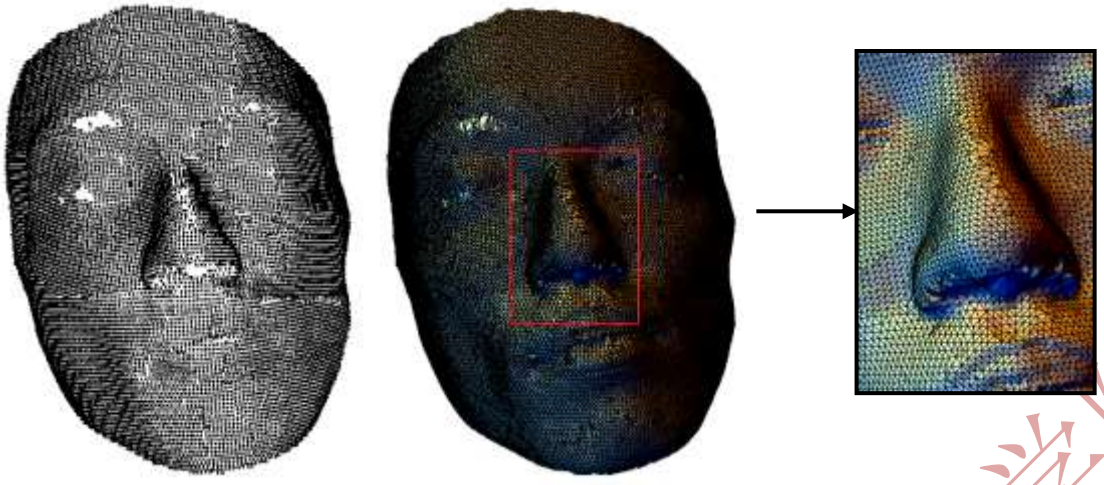
(f) face-YO (g) horse (h) standford-bunny

图 2.5 经典模型重建结果



(a) egea 点云 (b) egea 网格 (c) egea 局部

图 2.6 egea 重建结果

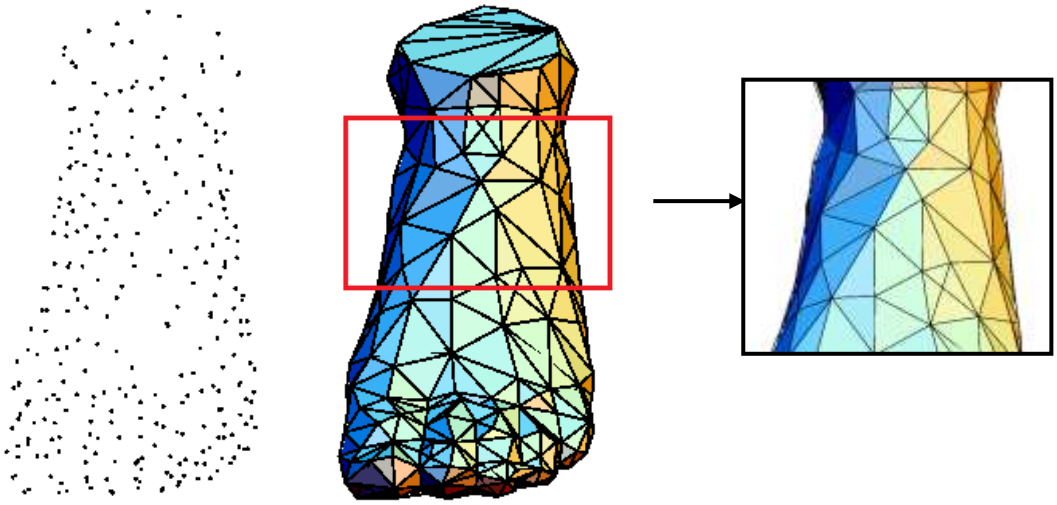


(a) face-YO 点云　　(b) face-YO 网格　　(c) face-YO 局部

图 2.7　face-YO 重建结果

2)　不同采样密度模型

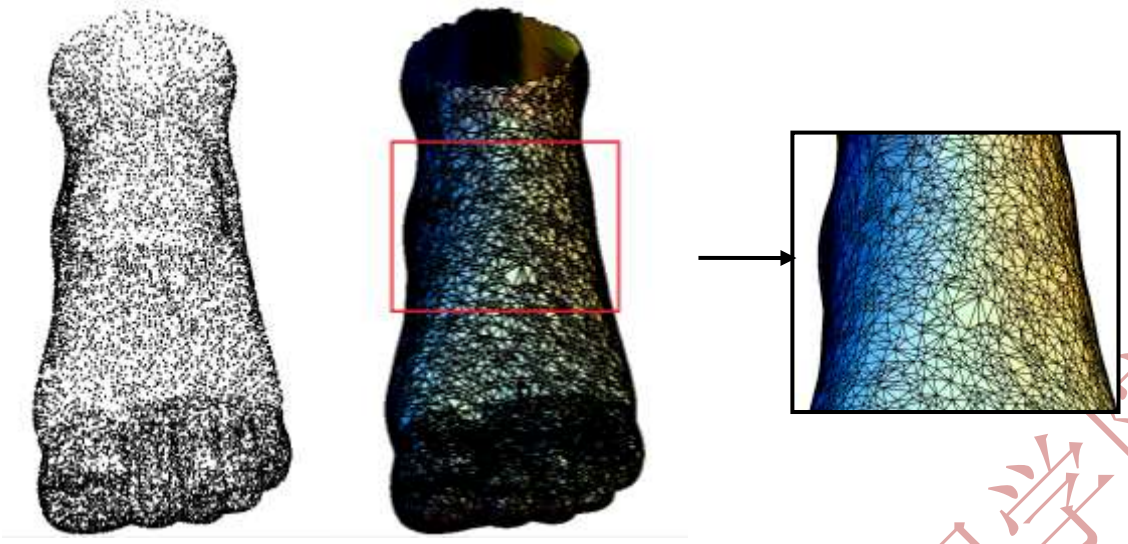
由图 9 和图 10 所示，图 9(a)和 10(a)显示的是 foot 不同采样密度的点模型，9(b)和 10(b)是重建后的网格模型。9(c)和 10(c)是红色部分的放大截图。对于 foot 模型不同的采样密度，本文算法重建后的结果都能够表现出模型的形状和特征，并没有因为采样密度的稀疏导致重建出来的模型不完整或者变形等情况。



(a) foot 点云　　(b) foot 网格　　(c) foot 局部

图 2.8　foot_1 重建结果

长信息工程学院



(a) foot 点云　　(b) foot 网格　　(c) foot 局部

图 2.9　foot_2 重建结果

3) 与 Tight Cocone 算法比较

为了显示本文算法的重建质量效果，将本文算法和由 Dey[5]等设计编码 Tight Cocone 曲面重建算法进行比较。非常感谢由 Tamal Dey 提供的 Tight Cocone 重建程序。实验中采用的是 Windows 版本，使用的选项参数为 tcocone –r 1.0。图 2.9 为比较模型的原图，图 2.11 是图 2.10 中用红色的框标记的地方。图 2.11 的(a),(c),(e)为使用 Tight Cocone 算法重建后的结果，(b),(d),(f)为使用本文算法重建后的结果。

由图 2.11(a),(c),(e)与 12(b),(d),(f)分别进行比较，不难看出使用 Tight Cocone 算法重建后的结果不如本文算法重建后的网格密集，并且本文算法重建得到的都是三角网格，排列规整，密度均匀。而使用 Tight Cocone 算法重建后的结果网格出现了分布不均匀的情况。从对比中可以看出，本文算法的重建效果要明显好于 Tight Cocone 算法。

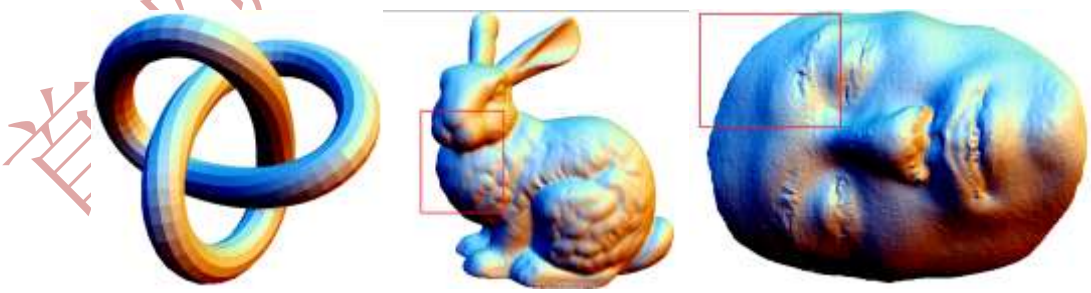
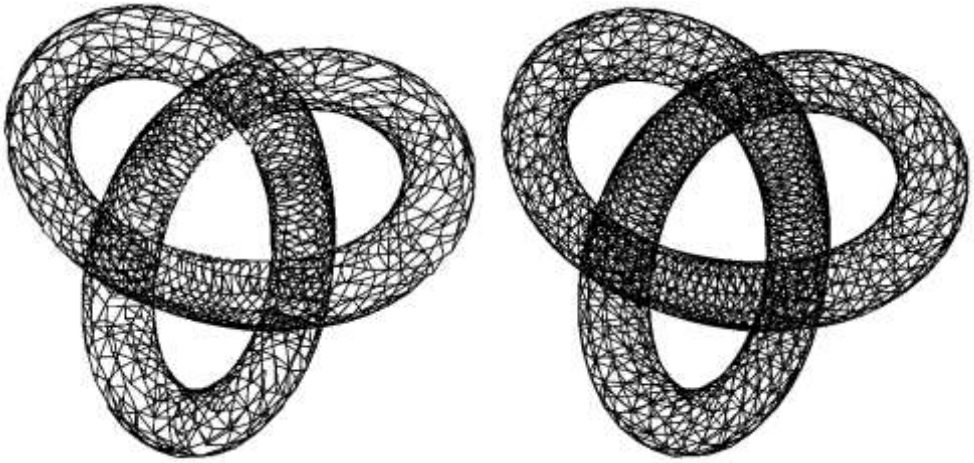
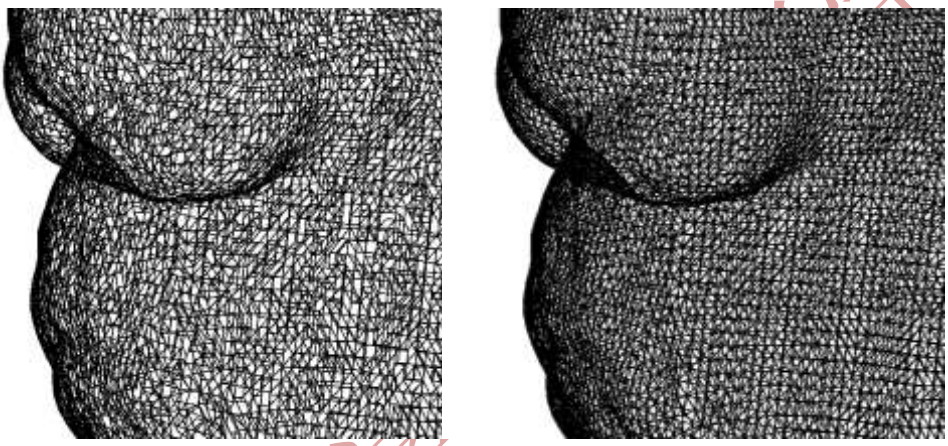


图 2.10　比较 loop、standford_bunny 和 face-YO 模型

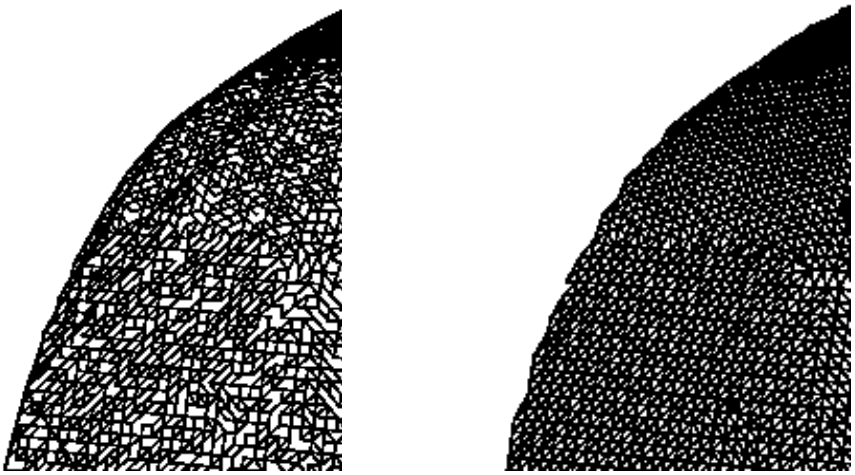
首都师范大学信息工程学院



(a) loop_1 网格模型　　(b) loop_2 网格模型



(c) standford_bunny_1 网格模型　　(d) standford_bunny_2 网格模型



(e) face-YO_1 网格模型　　(f) face-YO_2 网格模型

图 2.11　模型部分放大网格模型

2.5　本章小结

本文借助区域增长重建算法的易实现性和八叉树处理搜索空间数据的高效性，提出了

一种基于八叉树的区域增长三维重建算法。本文算法既不需要点集的法向量信息，也不需要大量繁杂的科学计算。实验证明，本文算法重建效果良好。但对过于稀疏的点云数据处理存在有一定的局限性，对一些采样不均匀的模型可能会存在碎片。由于本文算法不需要对三维点云数据进行前期处理，故三维点云重建消耗时间较长，今后会考虑对点云数据计算进行并行处理，进一步提高算法的速度。

首都师范大学信息工程学院

第 3 章 保持特征的自适应各向异性三角网格光顺算法

3.1 算法原理

本文光顺算法的核心思想是计算期望法向量使其接近原法向量，利用期望法向量计算偏移量和偏移系数，对顶点进行自适应调整，从而达到光顺的目的。

3.1.1 优化期望法向量

设图 3.1 所示图形为**错误!未找到引用源。**Φ，将给定目标三角形**错误!未找到引用源。**的相邻三角形分为两大类：第一种是与三角形 R 共边的相邻面集合，即图 3.1 所示 $S_i(i = 1,2,3)$**错误!未找到引用源。**集合，称 1 环邻域。另一种是与三角形**错误!未找到引用源。**共点的相邻面集合，$Q_j(j = 1,2,\ldots\ldots, n)$**错误!未找到引用源。**集合，称 2 环邻域。

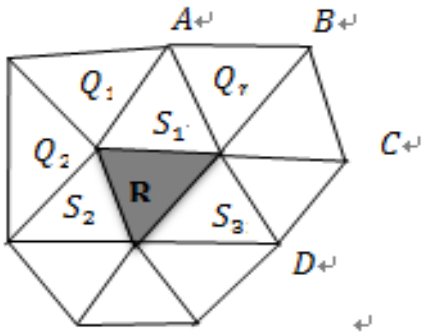


图 3.1 目标面**错误!未找到引用源。**的相邻面

考虑到为了准确表示期望法向量，权值由三角形**错误!未找到引用源。**的相邻三角形面积决定。当一个邻接面面积越大时，其权值越大，对目标面 R 的影响也越大。

期望法向量的计算公式为：

$$m(R) = \frac{1}{\sum A(S)} \sum_{S \in N(R)} A(S)n(S) \qquad (3.1)$$

其中，**错误!未找到引用源。**表示平面 S **错误!未找到引用源。**的面积，$N(R)$**错误!未找到引用源。**表示三角形 R 的相邻三角形集合。

为了能使期望法向量尽量接近真实的移动方向，分析了 1 环邻域内面法向量与 2 环邻域内面法向量的对期望法向量的影响。如图 3.2 所示，两个相交平面 Π_1、Π_2。假设 Φ-{$ABCD$ 围成的图形}**错误!未找到引用源。**在平面**错误!未找到引用源。**上，该平面包含了目标面**错误!未找到引用源。**，而由 $ABCD$ 所围成的图形在平面 Π_2 上。平面**错误!未找到引用源。**上的图形与目标面**错误!未找到引用源。**是点相连，如果同时考虑边相连和点相连的面，即 1 环邻域和 2 环邻域内的面的法向量，得到的期望法向量会偏离**错误!未找到引用源。**平面，

根据空间局部性原理，可以看到三角形**错误!未找到引用源。**处于相对平坦区域。采用 2 环邻域方法会导致模型特征不被保持，网格结构发生不确定变化。实验证明，仅考虑 1 环邻域面的法向量相对于考虑 1 环和 2 环邻域面的法向量得到的期望法向量更能够保持原模型的尖锐特征。

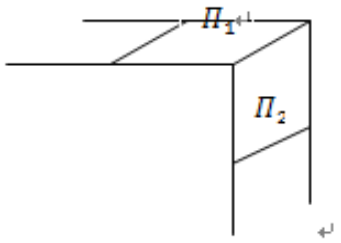


图 3.2 两个相交平面

3.1.2 优化顶点偏移量

如何根据期望法向量移动顶点，是一个重要问题。采用文献[4]的偏移方式，假设**错误!未找到引用源。**是三角网格中顶点 P 的偏移量。其计算公式为：

$$V(R) = \left[\overrightarrow{PC} \cdot m(R)\right] m(R) \tag{3.2}$$

错误!未找到引用源。其中，**错误!未找到引用源。**是包含**错误!未找到引用源。**为顶点的三角面片，**错误!未找到引用源。**代表的是三角面片**错误!未找到引用源。**的重心。$m(R)$ **错误!未找到引用源。**是三角面片**错误!未找到引用源。**的期望法向量。如图 15 所示，当区域比较平坦的时候，**错误!未找到引用源。**会接近平面**错误!未找到引用源。**原来的法向量**错误!未找到引用源。**，由此 $\overrightarrow{PC} \cdot m(R) \to 0$ **错误!未找到引用源。**，**错误!未找到引用源。**的值会较小，即顶点移动的偏移量较小，达到了保持特征的目的。相反，如果**错误!未找到引用源。**的值比较大，说明该顶点处于不平坦的区域，需要进行调整，以达到光顺的目的。

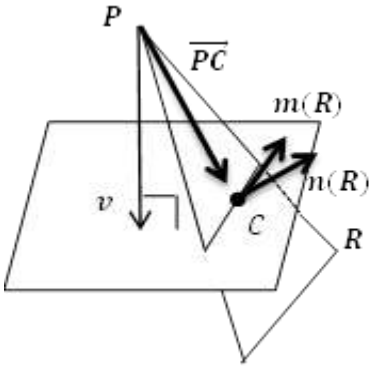


图 3.3 顶点偏移方向

首都师范大学信息工程学院

3.1.3 优化偏移系数

为了减少人工交互步骤，快速去除噪声，提高算法的自适应，本文提出偏移系数的概念。偏移系数就是根据顶点**错误!未找到引用源。**所处区域的特征，自动调节顶点偏移的比例。假设顶点**错误!未找到引用源。**的偏移系数为d_i。**错误!未找到引用源。**应具备如下性质：

1） 当顶点**错误!未找到引用源。**处于平坦区域，**错误!未找到引用源。**，这样可以保持原有特征。

2） 当顶点**错误!未找到引用源。**处于不平坦区域，d_i的值应随着噪声的频率增高而变大。

因此，本文设计一个自适应偏移系数函数**错误!未找到引用源。**。该函数的自变量**错误!未找到引用源。**为著名的雨伞操作算子**错误!未找到引用源。**的值，即**错误!未找到引用源。**。**错误!未找到引用源。**计算公式为：

$$U(i) = \frac{1}{\sum_{j \in NV(i)} w_{ij}} \sum_{j \in NV(i)} w_{ij} v_j - V_i \tag{3.3}$$

$$U(i)^2 = (x_i^2 + y_i^2 + z_i^2) \tag{3.4}$$

错误!未找到引用源。

其中，$NV(i)$**错误!未找到引用源。**是指顶点**错误!未找到引用源。**的邻接顶点集合，即与**错误!未找到引用源。**直接相连的顶点。**错误!未找到引用源。**是顶点**错误!未找到引用源。**的权值，主要有两种取法：1）取**错误!未找到引用源。**，即所有顶点的权值是相同的。2）利用顶点**错误!未找到引用源。**和**错误!未找到引用源。**的距离e_{ij}**错误!未找到引用源。**的倒数作为权值，表示了距离顶点v_i越近的顶点对其影响越大。在实验中采用第一种取法。

如图 3.4 所示，当**错误!未找到引用源。**时，**错误!未找到引用源。**会向下方移动，抑制了高频信息；如图 3.5 所示，当**错误!未找到引用源。**时，v_i会向上方移动，抑制了高频信息。**错误!未找到引用源。**可以很好的表示该噪声程度的大小。

对函数**错误!未找到引用源。**，本文尝试了如下函数：

$$g(x) = x \tag{3.5}$$

$$g(x) = \exp(x) \tag{3.6}$$

$$g(x) = 1 - e^{-(x-\mu)^2/\sigma^2} \tag{3.7}$$

错误!未找到引用源。其中$\mu = 0, \sigma$为一个小正数。

上述函数在定义域范围内都是为单调递增函数（**错误!未找到引用源。**），满足上述偏移系数**错误!未找到引用源。**的性质。实验证明：公式（3-5）能够很好保持特征，但需要迭代次数较多。公式（3-7）需要多次设置实验**错误!未找到引用源。**的值，某种程度上违背

首都师范大学信息工程学院

了自适应的目标。公式（3-6）能够有效快速去除噪声，并且保持特征。

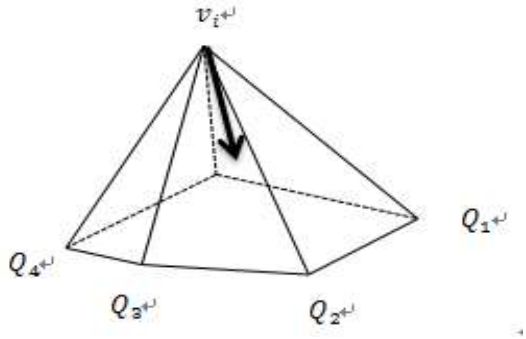


图 3.4 雨伞操作算子**错误!未找到引用源。**

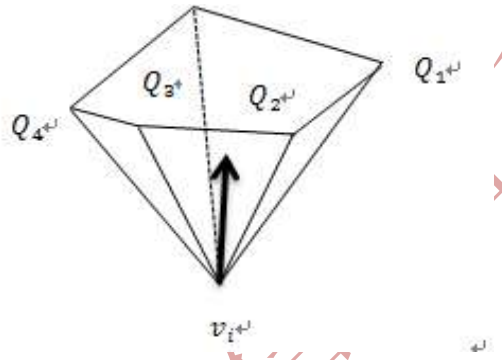


图 3.5 雨伞操作算子**错误!未找到引用源。**

3.1.4 更新顶点位置

求得偏移量和偏移系数后，得到顶点**错误!未找到引用源。**的更新公式为：

$$v_i^{'} = v_i + d_i V(R) \tag{3.8}$$

其中，v_i 和 $v_i^{'}$ 分别表示迭代前后顶点 v_i 的坐标。

3.1.5 光顺误差分析

为了将传统的经典光顺算法与本文算法进行比较，采用文献[15]提出的顶点误差法。假设**错误!未找到引用源。**是原网格模型，**错误!未找到引用源。**是光顺后的网格模型。采用如下计算公式来计算误差：

$$E_v = \sqrt{\frac{1}{3A(M^{'})} \sum_{P \in M^{'}} A(P^{'}) dist(P^{'}, M)^2} \tag{3.9}$$

其中，$P^{'}$ 是网格 $M^{'}$ 中的一个顶点，$dist(P^{'}, M)$ **错误!未找到引用源。**为顶点**错误!未找到引用源。**到模型**错误!未找到引用源。**中离**错误!未找到引用源。**最近的三角形的距离。**错误!未找到引用源。错误!未找到引用源。**是模型 $M^{'}$ **错误!未找到引用源。**中以**错误!未找**

首都师范大学 工程学院

到引用源。为顶点的所有三角面片的面积总和。**错误!未找到引用源。**是模型**错误!未找到引用源。**的面积。该误差大致体现了模型的体积变化。

3.2 算法流程

三角网格光顺可用于模型的去噪，通过调整每个点的位置信息，达到模型光顺的效果。保持特征的自适应各向异性三角网格光顺算法步骤为：

1) 首先根据目标面的 1 环域内的三角面法向量来计算其期望法向量。在计算三角面的期望法向量时，利用各邻接三角形面积作为权值，保证了准确表示期望法向量的同时，尽量与原法向量相似。

2) 然后根据期望法向量，计算各顶点偏移量[15]，当期望法向量与原法向量相差较多时，计算出来的偏移量值也会较大，使得顶点调整的幅度变大。

3) 最后计算雨伞操作算子[19]，当雨伞操作算子较小时，说明该点处于平坦区域，利用函数对雨伞算子进行计算，得出偏移系数，根据偏移系数和偏移量，得出新的顶点位置。

3.3 实验结果与分析

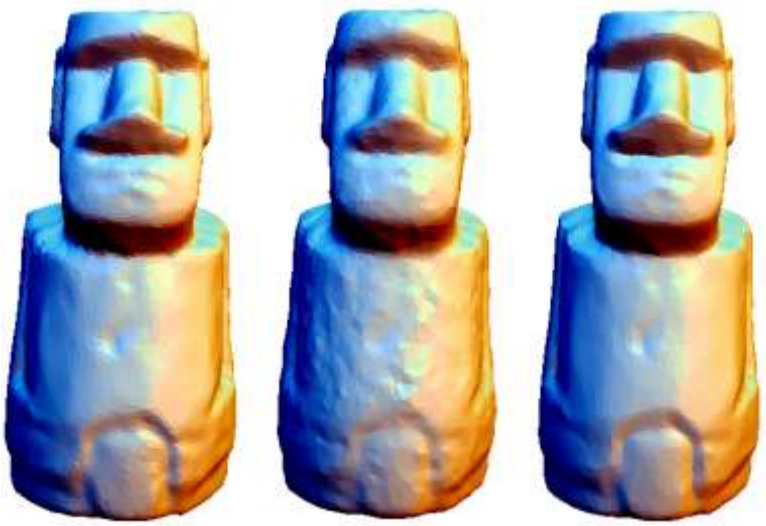
运用本文算法，在主频为 2.10GHz，内存 4.00GB（2.96GB 可用）的 Pentium Dual-Core CPU 的计算机上用 Microsoft Visual Studio 2010 和 OpenGL 图形库进行仿真实验，将本文算法和前人算法进行比较。

选取原始 moai 模型，对其添加 20%的随机噪声，将文本算法和拉普拉斯算法[12]、Taubin 的**错误!未找到引用源。**的算法[13]和平均曲率流算法[14]进行了光顺比较，如图 3.6 所示，数据对比结果见表 3.1。



（a）原图　　(b) 加入 20%噪声　　(c) Laplace 算法

首都师范大学信息工程学院



(d) Taubin 算法　　(e) 平均曲率流　　(f)本文算法

图 3.6 moai 模型本文算法与经典算法比较

表 3.1 moai 光顺算法的数据比较

算法	顶点数	三角面数	步长	迭代次数	误差	执行时间
拉普拉斯光顺	10002	20000	0.5	20	0.022616	78 毫秒
Taubin **错误!未找到引用源。**算法	10002	20000	0.5	20	0.006112	172 毫秒
平均曲率流	10002	20000	— —	20	0.010658	1060 毫秒
本文算法	10002	20000	— —	20	0.005699	5601 毫秒

选取原始 Octa-flower 模型，对其添加 20%的随机噪声，将文本算法和拉普拉斯算法[12]、Taubin 的**错误!未找到引用源。**的算法[13]和平均曲率流算法[14]进行了光顺比较，如图 3.7 所示，数据对比结果见表 3.2。



（a）原图　　(b)加入 20%噪声　　(c) Laplace 算法

首都师范大学信息工程学院



(d) Taubin 算法　　(e)平均曲率流　　(f)本文算法

图 3.7　Octa-flower 模型本文算法与经典算法比较

表 3.2 Octa-flower 光顺算法的数据比较

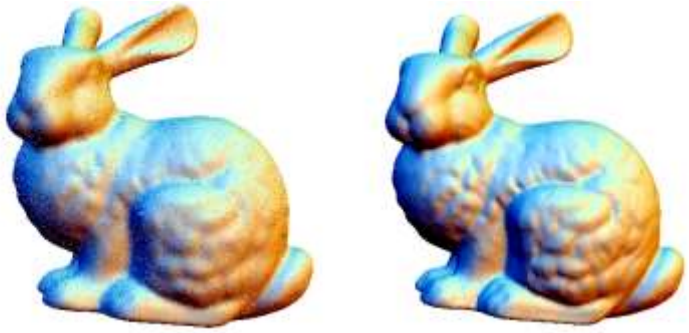
算法	顶点数	三角面数	步长	迭代次数	误差	执行时间
拉普拉斯光顺	7919	15834	0.5	20	0.135568	62 毫秒
Taubin **错误! 未找到引用源。**算法	7919	15834	0.5	20	0.035798	125 毫秒
平均曲率流	7919	15834	—	20	0.041597	1045 毫秒
本文算法	7919	15834	—	20	0.030784	3604 毫秒

图 3.8 和图 3.9 是对原始模型加噪后，本文算法与其他算法的光顺效果比较。（a）为原始模型，（b）为添加 20%的随机噪声后的模型，（c）-（f）为各种算法迭代 20 次后的光顺效果。（c）为拉普拉斯光顺，从图中可看到出现了明显的过光顺现象。（e）为平均曲率流算法，明显导致了网格的不规则。（d）和（f）的效果相近，但由表 3.1 和 3.2 看出，（f）本文算法的误差明显小于（d）Taubin 算法。但本文算法计算量明显大于经典算法，执行时间也相对较长。



(a) bunny 噪声网格　　（b）拉普拉斯光顺效果　　(c) Taubin 光顺效果

首都师范大学信息工程学院



(d) 平均曲率流光顺效果 （e）本文算法光顺效果

图 3.8 bunny 模型光顺效果比较



(a) monk 噪声网格 （b）拉普拉斯光顺 (c) Taubin 光顺



(d)平均曲率流光顺（e）本文算法光顺

图 3.9 monk 模型光顺效果比较

图 3.8 和图 3.9 所示分别为 bunny 模型和 monk 模型迭代 10 次光顺效果，图 3.8（a）是 bunny 噪声网格，3.9（a）为 monk 噪声网格，3.8（b）和 3.9（b）是经典拉普拉斯算法

步长选取 0.5 的光顺效果,3.8（c）和 3.9（c）是 Taubin 算法步长选取 0.5 的光顺效果，3.8（d）和 3.9（d）是平均曲率流算法的光顺效果，3.8（e）和 3.9（e）是本文算法光顺效果。可以明显看出，拉普拉斯算法会出现过光顺的现象，平均曲率流算法在图 3.8（d）中出现了过光顺的现象，并且残留较多噪声。本文的光顺效果要明显好于（b）,（c）,（d）经典算法，在去噪的同时，保持了原有的细节特征。

3.4 本章小结

本文针对三角网格光顺问题，提出了一种自适应各向异性的光顺算法。通过讨论环邻域面法向量的影响程度，计算模型中三角面的期望法向量，利用了图形的几何特征自适应求得偏移系数，然后调整顶点位置。实验表明，本文光顺算法达到了较为理想的光顺效果，能够自适应地保持网格尖锐特征，减少人工交互步骤，并且通过实验数据分析，本文算法相对其他算法误差较小，可有效保持模型的形状。

首都师范大学信息工程学院

第 4 章 三维重建与光顺系统软件实现

4.1 系统软件主要功能

三维重建与光顺系统软件的主要功能如图 4.1 所示，主要包括的功能包括数据的输入、三维点云重建、三角网格光顺、数据的保存和基本信息查询等功能。

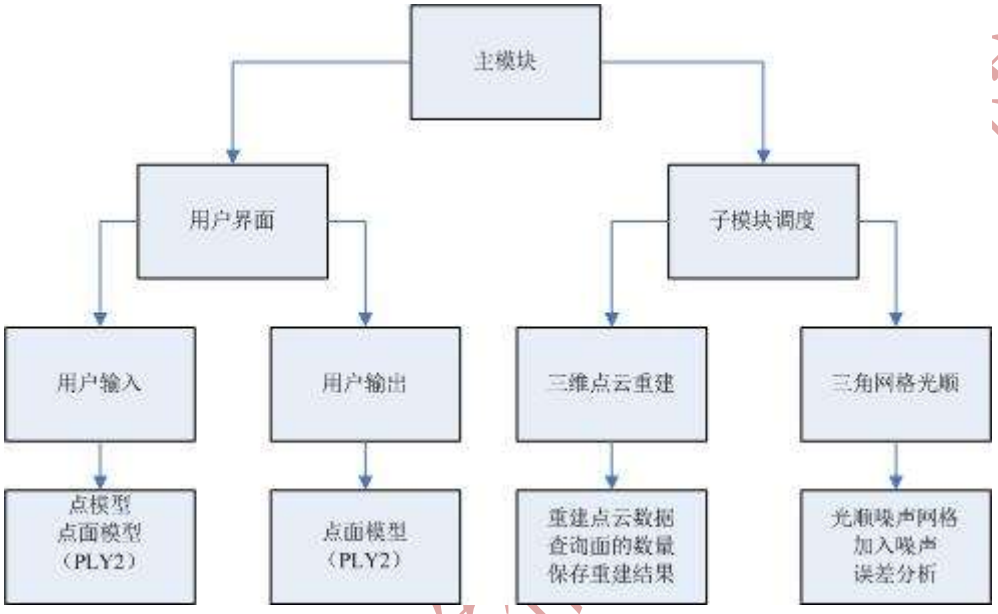


图 4.1 概要设计

功能详细描述：

- 用户输入：主要输入的是三维点云重建和三角网格光顺的源数据，重建源数据的内容包括点的数量和各个点的 XYZ 坐标，文件的后缀名可为.xyz。光顺的源数据的内容包括点的数量、点的坐标、面的数量和组成面的三个点的索引，文件的后缀名可为 ply2。对于用户的数据输入，只要点和面的信息全面，都可以进行操作。
- 用户输出：主要输出的是三维点云重建和三角网格光顺后的结果，重建后的结果可以保存为 PLY2 格式的文件，里面主要包括的信息是点的数量、重建后面的数量、点的具体坐标和重建后组成面的索引。可用于三角网格的输入。光顺后的结果依然为 PLY2 格式的文件，面的信息没有改变，改变的是三维点云的 XYZ 信息。
- 重建点云数据：按照本文的重建算法，对输入的源数据进行计算和操作，最后得面的拓扑关系。（将重建的算法步骤进行整合，方便用户使用）
- 查询重建的结果：对重建的结果可以进行查询，主要查询的是重建之后产生的面的数量。
- 光顺噪声网格：按照文本的光顺算法，对输入的源数据进行计算和操作，最后

得到点的更新坐标。（将光顺的算法步骤进行整合，方便用户使用）

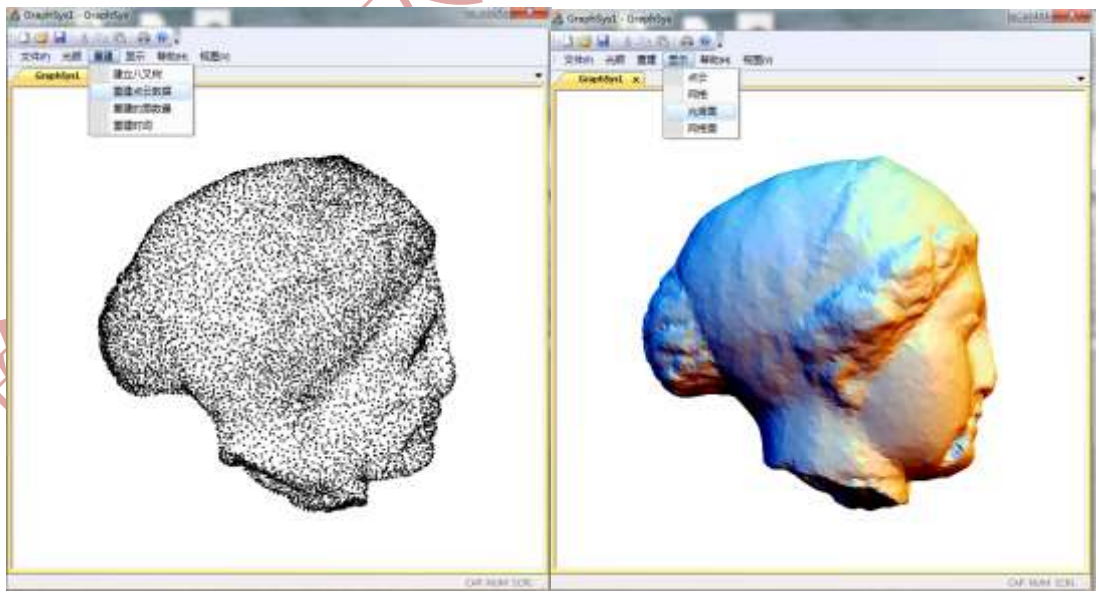
- 加入噪声：可以选择对模型进行加入不同程度的噪声，主要是为了研究不同程度的噪声的光顺效果。
- 误差分析：可以对光顺后的模型与原模型进行对比，得到一个误差值作为光顺效果的参考值。

4.2 系统软件的显示实现

三维重建与光顺系统软件的显示实现主要用到的是 OpenGL 编程。OpenGL（Open Graphics Library）是一个专业的图形程序接口，是一个功能强大，调用方便的底层图形库并且使用范围非常广泛。调用基本的 OpenGL 函数，可是实现基本的点绘制、线绘制和面绘制。为了将三维模型进行更好的显示，增加用户感受，可对三维模型进行光照处理。

OpenGL 库中有对光照的函数支持，这里只对其做简单叙述，不作为重点内容解释介绍。OpenGL 中采用的为简单光照模型，主要考虑的光分为：辐射光、环境光、漫射光和镜面光。对一个三维模型进行光照，首先利用 OpenGL 函数进行创建光源，设置光源中 RGB 参数。利用函数设置人的视觉坐标系，启动光源。可以对设置的参数进行修改达到想要的效果。

如图 4.2 所示，图 4.2（a）是 egea 模型的点云显示效果，4.2（b）显示的是 egea 开启光照后的显示效果。可以通过调节光照的参数，达到不同的光照效果。



(a)egea 的点云显示　　（b）egea 的光照显示

图 4.2 egea 模型的光照实例

4.3 系统软件的算法实现

三维重建与光顺系统软件的算法实现的语言为 C++，使用的平台为 Microsoft Visual Studio 2010 下的 MFC 框架。使用 C++编写主要考虑的因素为 C++语言编写的程序效率较高，并且有 STL 模板的支持，可以给编程工作带来一些便利。

4.3.1 重要的数据结构

点：点的数据结构重要性在于重建点云数据最后输出的结果为一个三角网格模型，三角网格主要是由面组成，面的信息需要点云的索引位置。所以点除了坐标信息以外，还要包括点的索引信息，即点在点集中的顺序位置，用变量 tag 标记。在拓扑重建操作中，需要对点云的类型进行判断，用一个变量 flag 来标记点的类型。以下是一个点云的数据结构 NodePoint 实例示意。

```
struct NodePoint
{
    float x,y,z;                //点的三维坐标
    int tag;                    //点的位置索引
    int flag;                   //点的类型标记
};
```

八叉树：八叉树用于三维点云重建过程，它需要对子节点进行递归建立，由下到上，由左到右，由前到后。以下是一个八叉树的数据结构 Octree 设计实例示意及解释说明。

```
class Octree
{
public:
    bool        m_SubDivided;           // 该节点是否分解过
    float       m_Center[3];            // 节点包围盒中心坐标
    float       m_Width[3];             // 节点包围盒三个轴向的宽度
    Octree    * m_pOctreeNodes[8];      // 八个子节点
    NodePoint *     m_pVertices;            // 保存全局的点坐标
    int             m_PointsCount;          // 节点中的点的个数
    int m_level;      //该点的是第几层
    bool visited;     //这个盒子是否被访问过
};
```

在八叉树的建立过程中，主要的难点在于递归建立每一个节点的子节点并且对数据点

首都师范大学信息工程学院

p 的坐标进行分类。假设中心点为坐标原点(0,0,0)，那么点 p 在该中心点所在包围盒的位置就为空间坐标系的几个位置，将 $p(x, y, z)$ 与（0,0,0）进行比较，将 $x \geq 0, y \geq 0, z \geq 0$ 设置为第一象限，$x \geq 0, y \geq 0, z \leq 0$ 为第二象限，同理，一共有八个象限。m_SubDivided 变量是用于标记该包围盒是否为叶子节点，如果为 true 的话，说明这个包围盒被分解过，那么它就是中间节点，反之为叶子节点。m_Center 是该包围盒的中点坐标，m_Width 是该包围盒的长宽高，m_pOctreeNodes 该节点的 8 个子节点。m_PointsCount 是该包围盒所包含的点的数量。Visited 是该包围盒是否被访问过，主要用于搜索邻近点。

3DObject：一个三维模型的主要信息包括点云数据、点云个数、面的个数和面的信息。面的信息是组成面的索引。以下是一个 3DObject 的实例示意。

```
Class 3DObject
{
    int vertexNum;          //点的个数
    int faceNum;            //面的个数
    float* m_ pVertices;    //点的坐标
    int * m_faces;          //面的索引
};
```

边 Map：在三维重建过程中，需要保存一些重建过程中产生的一些中间状态信息。其中，删除边 Map 中存储的信息就是已经被两个三角面片共享的边，关联边 Map 放入的是所有重建过程中已经产生过的前沿边。在重建过程中，每次都需要对前沿边进行判断，看当前前沿边是否在删除边 Map 中。边主要包括的信息是边的两个顶点，为了减少查询时间，采用点的索引进行保存，因为索引是唯一的。将两个顶点的索引同时作为 Map 中的关键值进行判断，搜索的时间为 $O(1)$。

STL 模板：STL(Standard Template Library)是标准模板库，它是一些容器的结合，可以简化代码和提高代码质量。主要用到的 STL 是 Map，Vector 标准模板，用于搜索和存储数据。Map 模板可以对数据进行快速和高效的检索，定义形式为 Map<Key，Value>。其中，Key 是键值，用来标记对象唯一性的属性。Value 是该对象的其他属性。Vector 模板可以进行存储对象，它的访问形式类似于一维数组，使用 Vector 模板的好处是不用担心数组越界等问题。

4.3.2 重要的函数

在三维重建和光顺算法的实现上，主要的函数包括：

- 建立八叉树：

首都师范大学信息工程学院

八叉树的建立过程是对初始节点创建其属性和八个子节点，然后递归的对八个子节点分别进行创建。在创建过程中，需要把点云放入到八叉树结构中，当一个节点的点云数量满足条件时，停止继续分解，将该节点作为叶子节点。在递归求解过程中，局部变量和全局变量的关系容易混淆。

- 平坦度计算：

平坦度的计算需要求解其协方差矩阵对应的最小特征值。协方差矩阵是一个矩阵，其每个元素是各个向量元素之间的协方差。协方差用于描述两个随机变量的相关程度。

- 八叉树搜索：

首先找到目标点所在的包围盒，然后对包围盒内的点进行搜索。如果该包围盒内的点不满足一定条件，将搜索范围扩大到上一级包围盒，并且将刚刚搜索过的包围盒进行标记，以便下次不再访问，对上一级的八个包围盒进行递归搜索。

4.4 算法伪代码

4.4.1 三维重建算法

```
ReadCloudPoints();//读入点云数据
BuildOctree()&&ComputeFlatFactor(); //建立八叉树结构并计算平坦度
FindFirstTriangle();//寻找第一个三角形
While(IsEmptyQueue(queue))//前沿边队列不为空
{
L(a,b)=Pop(queue);//弹出前沿边
P=SearchPoint(L(a,b));//根据前沿边选择加入点 p
Switch(P)://判断 p 的类型
Case unused: Join; break;//如果没有用过，执行 Join 操作
Case used://p 被使用过
If(L(p,a)&&L(p,b)) Fill;break;//与前沿边都相连，执行 Fill 操作
Else if If(L(p,a)):Link;break;//与前沿边其中一个相连，执行 Link 操作
Else If(L(p,b)): Link;break;
Else Join;break;//没有与前沿边相连，执行 Join 操作
}
```

首都师范大学信息工程学院

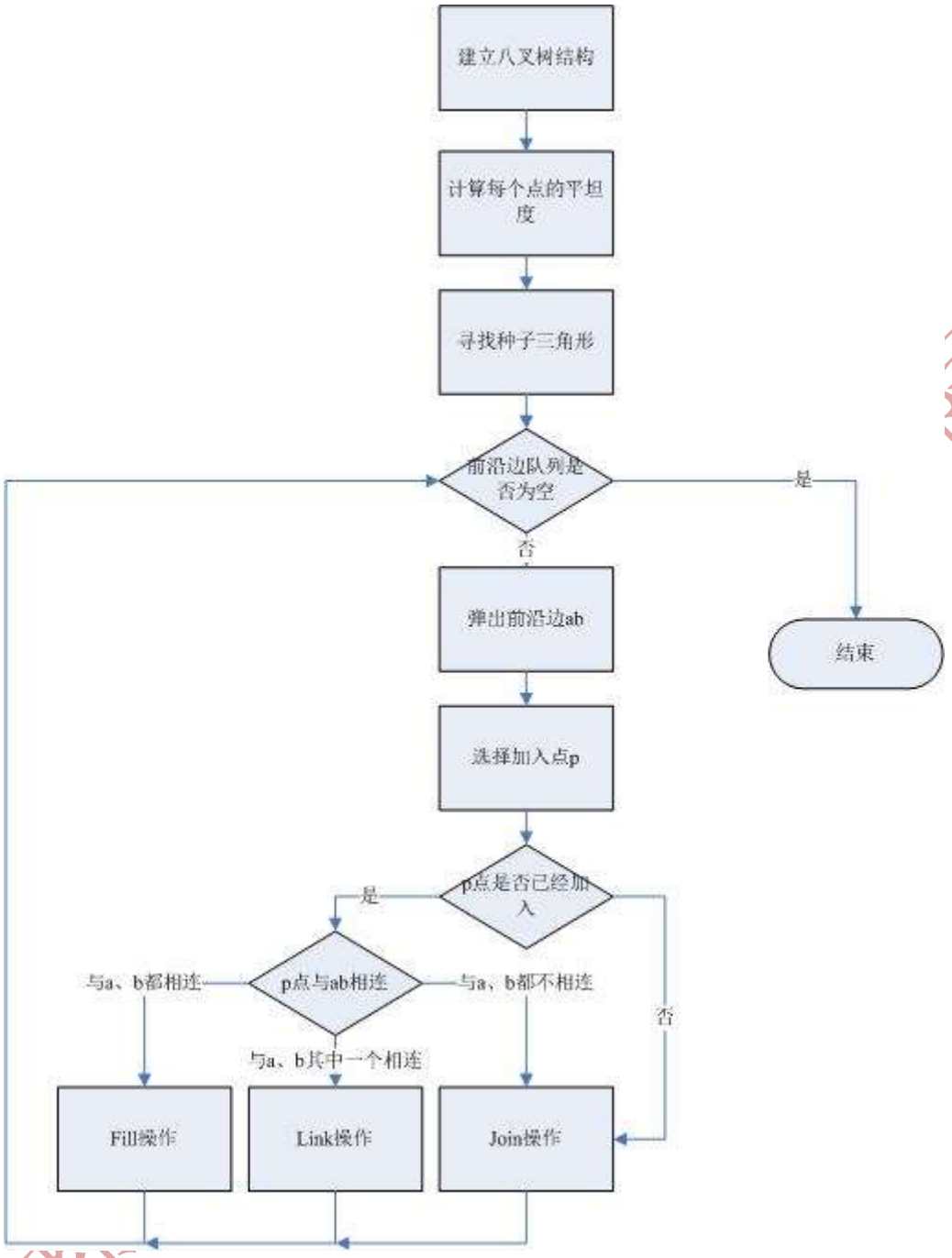


图 4.3 重建流程图

4.4.2 三角网格光顺算法

```
For(every point)
{
    ComputeOneRingNormal();//计算 1 环邻域期望法向量
    ComputeOffset();//计算每个点的偏移量
    ComputeOffsetCoef();//计算每个点的偏移系数
    MovePoint();//移动每个顶点的位置
```

}

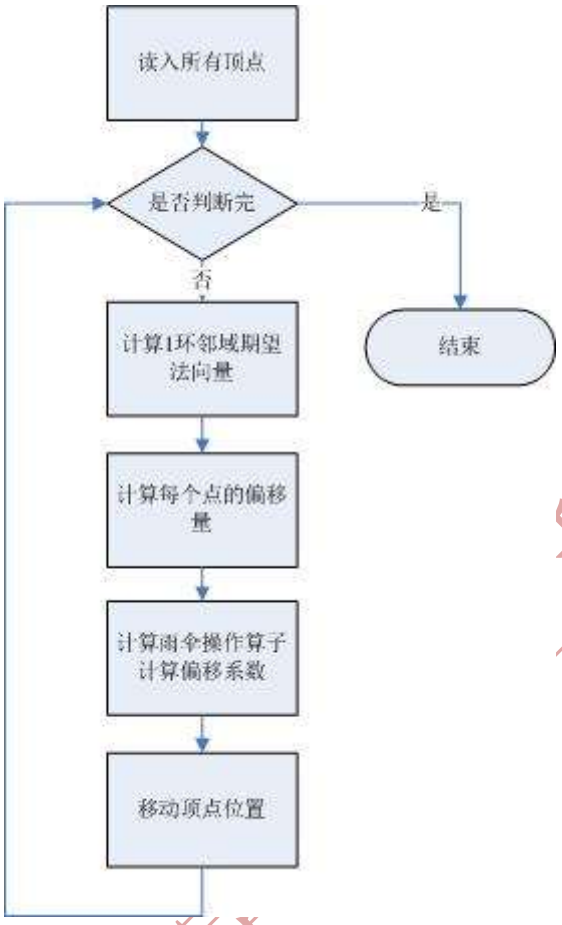
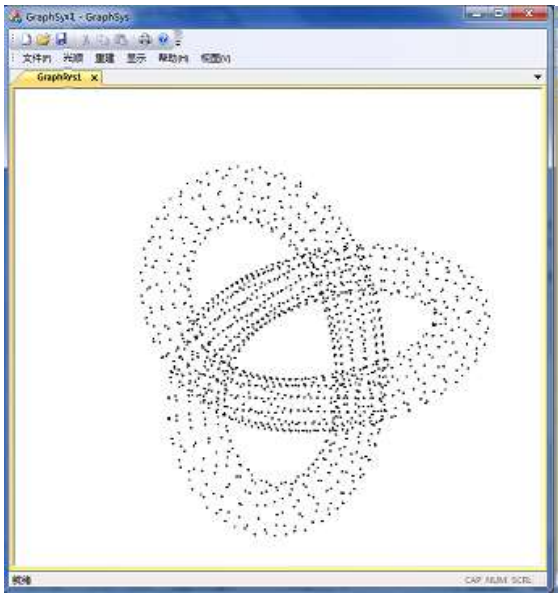


图 4.4 光顺流程图

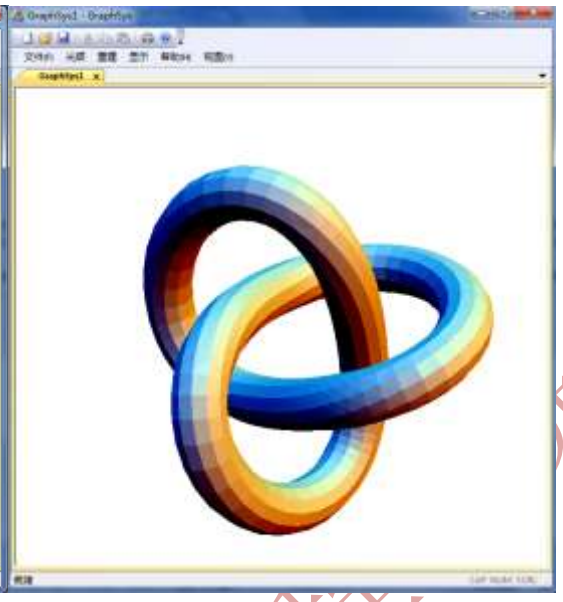
4.5 系统软件演示

由图 4.5 所示，采用了 loop, dilo, face-MH 和 dino 模型进行重建，其中（a），（c），（e），（g）显示的是三维模型的点模型。其中，（a）图为一个环模型，（b）图为重建后的效果，可以看到重建后的结果良好，没有出现碎片，在环内的交叉部分也没有出现重建错误。（e），（f），（h）为重建后的三维模型，可以看到，对于不同几何特征和不同点云数量级别的点云模型，重建的效果都比较好。既保持了原有模型的特征，并且重建出现的碎片很少。

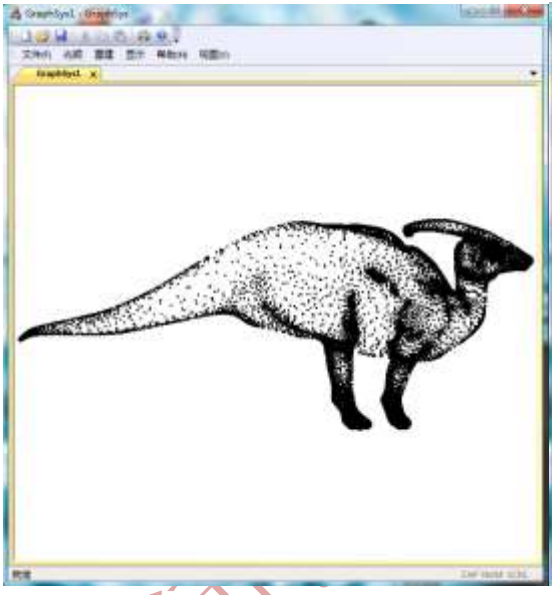
首都师范大学 程学院



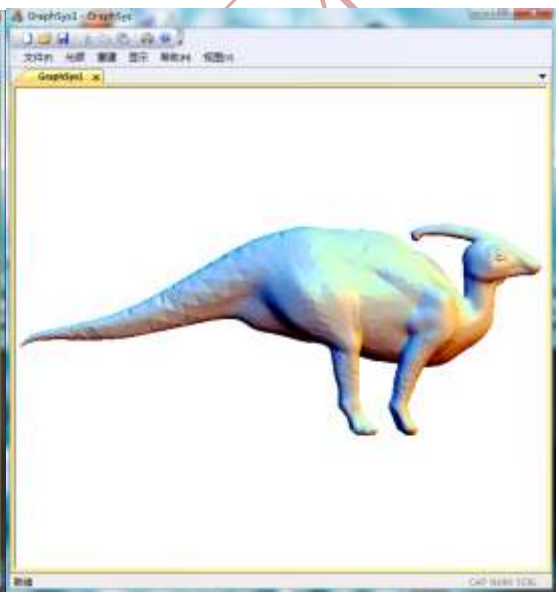
(a) loop 点云



(b) loop 重建模型

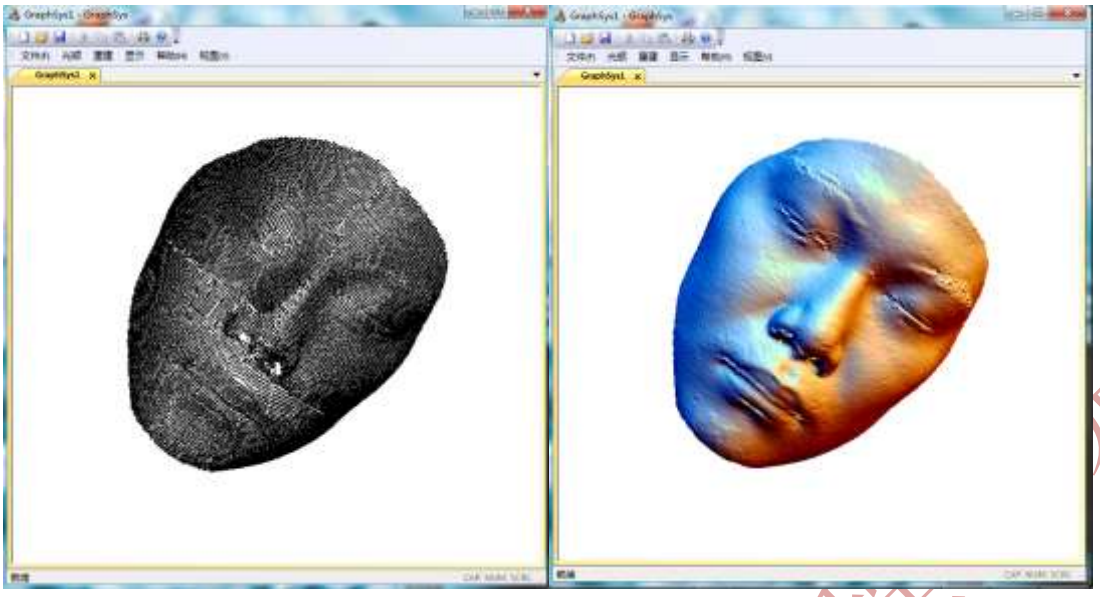


(c) dilo 点云

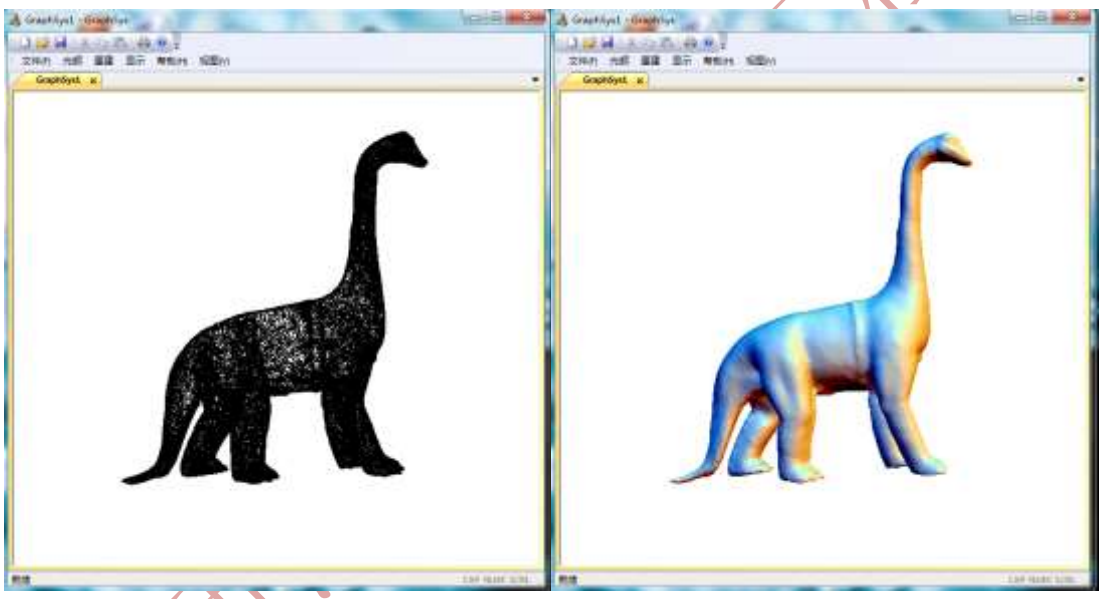


(d) dilo 重建模型

首都师范大学



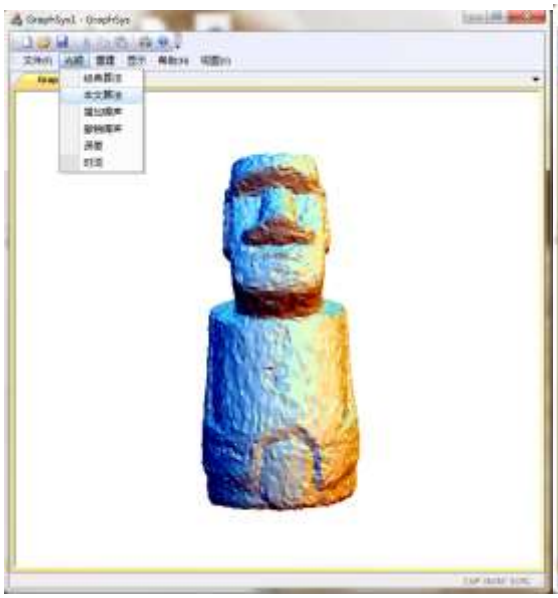
(e) face-MH 点云　　(f) face-MH 重建模型



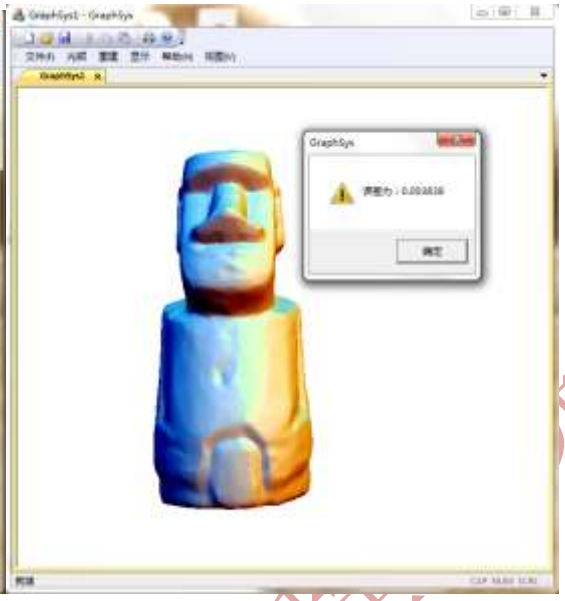
(g) dino 点云　　(h) dino 重建模型

图 4.5 重建实例

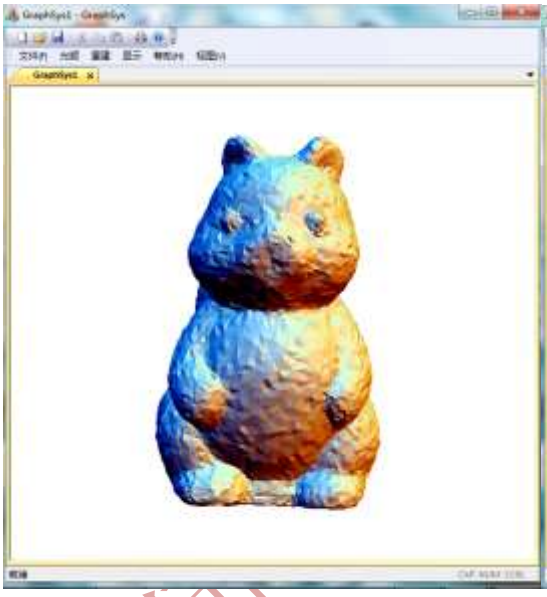
如图 4.6 所示，采用了 moai,risu, armadillo,bimba 和 Max-Planck_bust 进行光顺。光顺迭代 10 次。其中（a），（c），（e），（g），（i）显示的是有噪声的模型，（b），（d），（f），（h），（j）显示的是光顺后的结果。从光顺后的结果来看，原有的噪声被去除，并且模型的体积几乎没有改变。在光顺后也没有出现过光顺现象，原有的特征也得到了保持。



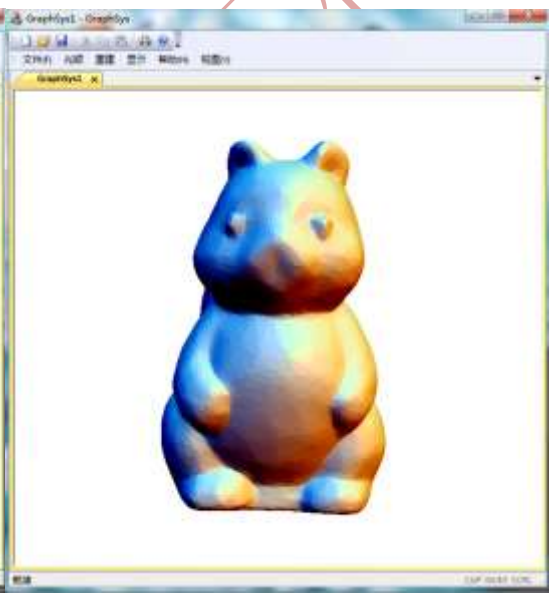
(a) moai 加噪



(b) moai 去噪模型

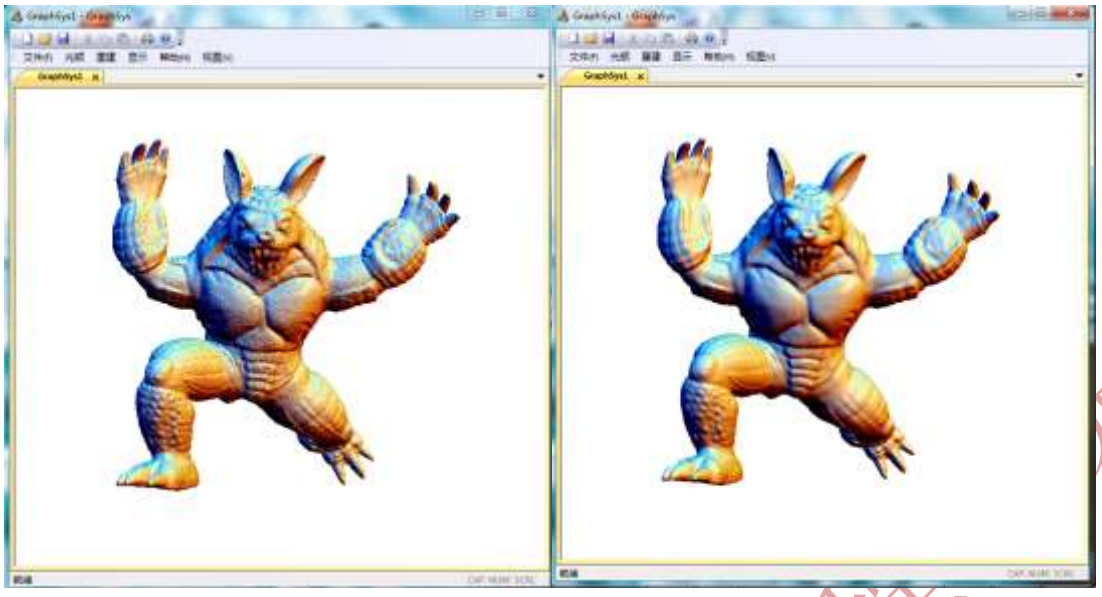


(c) risu 加噪

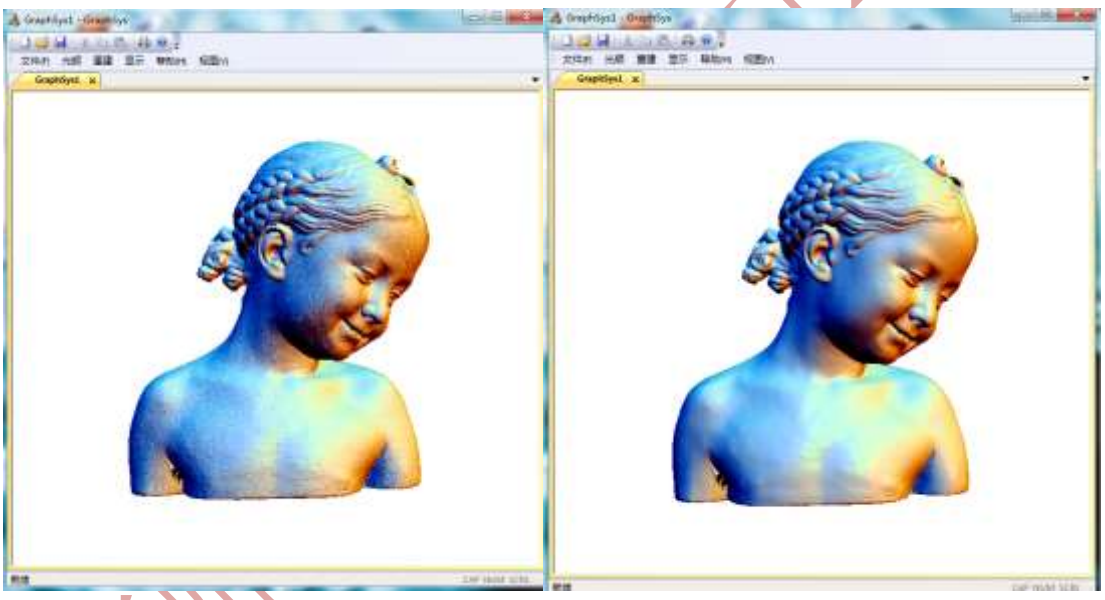


(d) risu 去噪模型

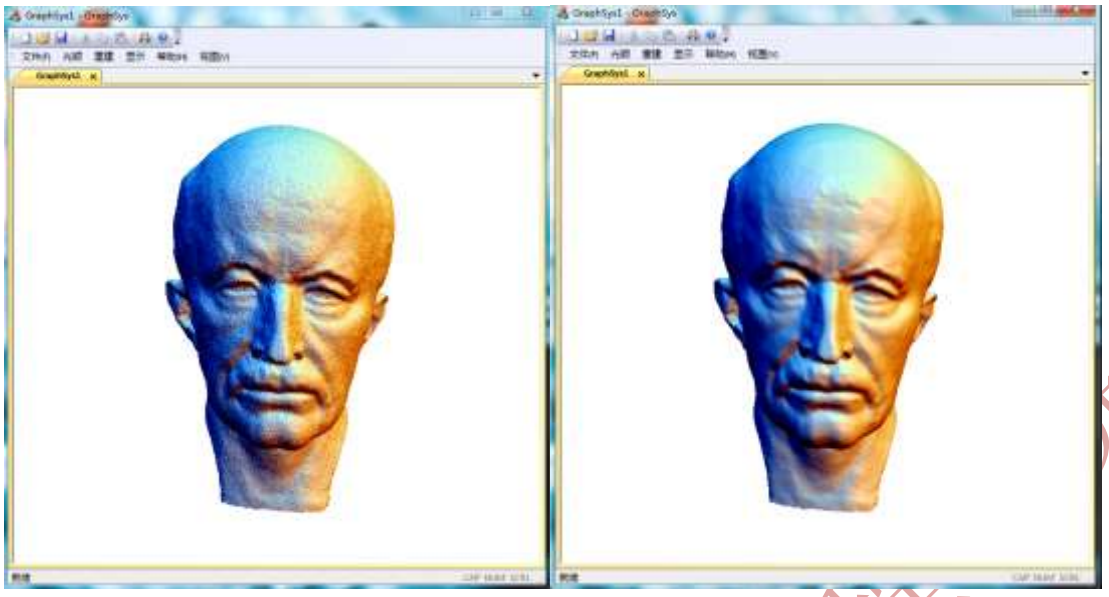
首都师



(e) armadillo 加噪　　(f) armadillo 去噪模型



(g) bimba 加噪　　(h) bimba 去噪模型



(i) Max-Planck_bust 加噪　　(j) Max-Planck_bust 去噪模型

图 4.6 光顺实例

4.6 本章小结

本章主要介绍了重建与光顺算法系统软件的设计，包括主要功能设计，编码实现的一些要点以及系统软件的演示。从系统软件的演示来看，良好地实现了三维模型的显示功能、光顺功能和重建功能。

不足之处在于当点云的数据很大时，程序执行的时间较长，不能达到良好的交互水平。对系统软件还需要进一步的优化和并行处理。

首都师范大学信息工程

第 5 章 总结与展望

5.1 本文总结

本文在前人的基础上，为了降低三维模型重建的时间复杂度，不使用三维模型的法向量作为输入数据，提出一种基于八叉树的区域增长三维重建算法。基于八叉树的区域增长三维重建算法具有较好的鲁棒性，对于不同几何特征的模型和不同的采样率、采样密度有较好的处理效果。该算法不需要点的法向量信息与提前预处理，所以耗费的重建时间较多。

为了快速有效去除噪声，保持原模型的尖锐特征，并且减少人工交互的步骤和计算量，提出一种自适应各向异性光顺算法。自适应各向异性光顺算法可以有效的去除噪声，在存在较多噪声的情况下，也具有较好的重建效果。但由于计算过程较为复杂，所以消耗的时间也比较多。

5.2 未来研究展望

在今后的研究工作中，可以考虑对重建和光顺过程进行并行化处理，提高速度。可以将算法移植到云平台进行分布式计算，分布式计算是一种需要进行大量计算的工程数据分区成小块，由多台计算机分别计算，在上传运算结果后，将结果统一合并得出数据结论的科学。

也可以使用 GPU 代替 CPU 进行加速处理，GPU 是一个专门的图形的核心处理器，使得运行速度加快，提高软件交互水平。

首都师范大学信息工程学院

参考文献

[1]Y.Ohtake, A.Belyaev, H-P.Seidel. 3D scattered data interpolation and approximation with multilevel compactly supported RBFs [J]. Graphical Models,2005,67(3):150-165.

[2]Kazhdan M,Bolitho M,Hoppe H. Poisson surface reconstruction[A]. In: Eurographics symposium on Geometry processing[C]. Switzerland:2006,pp.61-70.

[3]林裕旭，陈纯，宋明黎，佳俊.隐曲面重建的极性场模型方法[J].计算机辅助设计与图形学学报，2009,21(2):1035-1041.

[4]Amenta N,Bern M,Kamvysselis M. A new Voronoi-based surface reconstruction algorithm[A]. In: Computer Graphics SIGGRAPH 98 Proceedings [C]. New York:1998,pp.415-421.

[5]Dey TK, Goswami S. Tight Cocone : A Watertight Surface Reconstructor[J]. Journal of Computing and Information Science in Engineering, 2003,3(4):302-307.

[6]Dey TK, Goswami S. Provable surface reconstruction from noisy samples[J]. Computational Geometry,2006,35(1-2):124-141.

[7]Dey TK, Giesen J,Hudson J.Delaunay based shape reconstruction from large data[A]. In: The proceeding of the IEEE symposium on parallel and large-data visualization and graphics[C]. 2001,pp.19-27.

[8]Dey TK,Dyer R,Wang L. Localized Cocone surface reconstruction[J]. Computers & Graphics, 2011,35(3):483-491.

[9]Bernardini F,Mittleman J,Rushmeier H,Silva C,Taubin G. The Ball Pivoting algorithm for surface reconstruction[J]. IEEE Transactions on Visualization and Computer Graphics, 1999,5(4):349-359.

[10]Li XK,Han CY, Wee WG.On surface reconstruction:A priority driven approach[J]. Computer-Aided Design, 2009,41(9):626-640.

[11]Angelo LD,Stefano PD,Giaccari L. A new mesh-growing algorithm for fast surface reconstruction[J]. Computer-Aided Design,2011,43(6):639-650.

[12] Field D A. Laplacian smoothing and Delaunay triangulations[J]. Communications in Applied Numerical Methods,1988,4(6):709-712.

[13] Taubin G. A signal processing approach to fair surface design[A].In:Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH[C]. Los Angeles:1995,pp.351-358.

[14] Desbrun M, Meyer M, Schroder P, Barr A H. Implicit fairing of irregular meshes using diffusion and curvature flow[A]. In: Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH[C]. Los Angeles:1999,pp.317-324.

[15] Ohtake Y, Belyaev A G, Bogaevski I A. Mesh regularization and adaptive smoothing[J]. Computer-Aided Design,2001,33(4):789-800.

[16]赵晔，翟世梅.一种基于带噪声的三角网格模型的光顺算法[J].计算机应用研究，2012,29（9）：3309-3311.

[17]章雅娟，李泽安，程晨.一种改进的三角网格模型光顺算法[J].计算机工程，2012,38（9）：220-222,239.

[18] Yagou H, Ohtake Y, Belyaey A. Mesh smoothing via mean and median filtering applied to face normals[A]. In: Geometric Modeling and Processing Proceedings[C]. 2002,pp.124-131

[19]唐杰,徐波等.一种基于 CUDA 的三维点云快速光顺算法[J].系统仿真学报,2012,24(8):1633-1642.

首都师范大学信息工程学院

[20]张冬梅，刘利刚等. 保特征的加权最小二乘三角网格光顺算法[J].计算机辅助设计与图形学学报,2010,22(9):1497-1501.

[21] Kobbelt L, Campagna S, Vorsatz J, Seidel H-P. Interactive multi-resolution modeling on arbitrary meshes[A]. In: Computer Graphics SIGGRAPH 98 Proceedings[C]. 1998,pp.105-114.

[22]张永玉，马劲松. 3D GIS 中线性八叉树空间索引的建立与查询算法研究[J].计算机工程与科学，2009,31(2):61-63.

[23]LIN Hong-wei,Tai Chiew-lan,Wang Guo-jin. A mesh reconstruction algorithm driven by an intrinsic property of point cloud[J].Computer-Aided Design,2003,36(1):1-9.

[24]陈金锐. 点云数据三维重建研究[D].武汉：武汉理工大学，2011.

[25]季振方. 基于空间状态八叉树的点云模型表面重建方法[D].山东：山东大学，2011.

[26] 熊邦书，何明一，俞华璟.三维散乱数据的 k 个最近邻域快速搜索算法[J].计算机辅助设计与图形学学报,2004,16(7):909-912,917.

首都师范大学信息工程学院

致　谢

首先感谢我的导师刘旭敏教授在我本科生阶段的学习、工作中给我的悉心指导和谆谆教诲。刘老师严谨的治学态度对我影响颇深，特别是对我解决问题能力的提高上有很大的帮助。正是刘老师对我无私的指导，让我对科研有了初步的了解和信心。刘老师一心为学生着想和实事求是的工作作风、踏踏实实的为人之道也让我受益匪浅。相信这些都会在我以后的生活工作有所帮助。在此向我的导师致以崇高的敬意、深深的感谢和良好的祝愿。

另外要感谢我的同伴，康炎丽同学，是她和我一起相互支持，相互学习，相互讨论才能够顺利的完成项目。她严谨的态度和清晰的思路给我留下了深刻的印象。还要感谢我的学姐李彩玲硕士，从项目开题开始，是她细心的指导和无私的帮助，让我打开了计算机图形学这扇大门。在项目进展中，她对我的研究工作给予了很大关心和鼓励。还有纳森同学，他对项目的实现细节上提供了自己的见解。

此外，真诚感谢我在实习工作中遇到的老师和同学，他们在思想上、生活上都给我了很大启迪，让我对自己的学习和工作有了一个新的认识。

特别感谢我的父母及所有亲人，感谢他们对我无私的关怀和呵护，并且感谢所有关心支持并鼓励我的朋友们！

谢谢！

首都师范大学信息工程学院