文章编号:1001 - 9081(2008)08 - 2046 - 03

基于八叉树空间分割的 k 近邻搜索算法

黄 森[1,2],张海朝[2],李 超[3]
(1. 平顶山学院 软件学院,河南 平顶山 467000; 2. 河南科技大学 电子信息工程学院,河南 洛阳 471003;
3. 西华师范大学 计算机学院,四川 南充 637002)
(hm821124@126.com)

摘 要:以三维扫描得到的散乱点云为基础,提出了一种基于空间八叉树的快速 k 近邻搜索算法,通过对点集建立包围盒,利用八叉树记录分割过程,从而使近邻点的搜索只局限于采样点所在的包围盒及其周围的包围盒,并通过剪枝策略使搜索范围进一步缩小。大量真实数据的实验结果表明:该算法可以很好地提高近邻点的搜索速度。

关键词:k 近邻;八叉树;包围盒;曲面重建
中图分类号: TP391.41 **文献标志码**:A

Algorithm for finding k-nearest neighbors based on octree segmentation in space

HUANG Miao[1,2], ZHANG Hai-chao[2], LI Chao[3]
(1. *Software Engineering School, Pingdingshan University, Pingdingshan Henan* 467000, *China*;
2. *School of Electronic Information Engineering, Henan University of Science and Technology, Luoyang Henan* 471003, *China*;
3. *Computer College, China West Normal University, Nanchong Sichuan* 637002, *China*)

Abstract: An octree based on the rapid k nearest search was presented for scattered points from 3D scanner. Through the establishment of bounding box on point sets, octree was used to record segmentation process, so that the search of point neighbors was limited to the bounding box of sample points and its neighbor, and through pruning strategies to further narrow the scope of the search. The experiments on a great deal of real data show that the search speed can be well improved.

Key words: k-neighbor; octree; bounding box; surface reconstruction

0 引言

在科学计算可视化、雕塑曲面造型和反求工程等应用领域,经常需要从一个离散点的点集出发,重构点集所属的原有曲面,这就是曲面重建问题。基于散乱点集的曲面重建,无论在理论上还是在实践中都具有普遍性。散乱点集的曲面重建一般需要经过拓扑重建、网格简化与几何重建 3 个步骤,其中拓扑重建就是要重建各采样点之间的拓扑关系,重建的结果为一张(或多张分离的)与原曲面拓扑等价的三角形分片线性曲面。对于大量的密集散乱点群数据,无论采用何种曲面建模方法,其首要问题都是如何高效地处理庞大的三维点群数据(如特征点、邻接点或边界点等的搜寻)。由于原始的测量点数据之间没有显式的几何拓扑关系,任何点的搜寻都必须在点群集合的全局范围内进行,这是造成三维散乱点群几何建模速度很低的主要原因。

通常,计算某点的 k 个最近领域的方法是求出候选点与其余 $n-1$ 个点的欧氏距离,并按从小到大的顺序排列,前面的 k 个点即为候选点的 k 个最近邻域。这种方法很直观,但真实数据的 n 往往很大,用它来计算数据点的 k 个最近邻域必然会很耗时。许多学者针对此问题进行了一些快速算法的研究,其中,文献[1]对此问题进行了开创性的研究。该文献首先给出了一个刻画点集密度的方法,引入了 ρ 密度样本的定义,提出了一个基于等值面抽取的隐式曲面重建算法;文献[2]提出了一个基于 α-shape 壳抽取的算法;文献[3]提出了一个基于 Voronoi 邻域特性的重建算法,但点集 Voronoi 图的计算量仍然非常大;文献[4]提出了一个基于局部构造的快速增量算法,利用空间分块策略进行 k 个最近点搜索。但这些方法既不能保证空间分块具有最佳或接近于最佳的搜索速度,也不能保证每个数据点都能找到 k 个最近邻域。

本文提出一种基于空间八叉树分割的 k 近邻搜索算法,该算法综合考虑了数据集的范围、点的总数及最近点数目 k,可以给出接近于最佳搜索速度的分块大小,并且在搜索终止准则上进行改进,以进一步提高 k 个最近邻域的搜索速度。实验结果表明,本算法跟以上算法相比具有适应面广、复杂度低、速度快等优点。

1 相关概念

为了便于散乱点搜索算法的描述和讨论,先给出一些相关的概念:

1)散乱数据点集。本文讨论的三维散乱点集是指仅有采样点的三维坐标,而无其他任何相关信息的点集。

2)k 近邻。给定一个曲面散乱点集 $P=\{P_i(x_i,y_i,z_i), i=1,2,\cdots n\}$,设某个点为 $V(x_v,y_v,z_v)$,则称 P 中距离点 V 最近的 k 个点为点 V 的 m 邻域点集,记为:$MNB|V|=\{P_1,P_2,\cdots,P_m\}$,称为点 V 的 k 近邻,它反应了该点 V 的局部信息。k 近邻中的每个点称为点 V 的邻近点。树节点的数据结构描述如下:

```
struct TreeNode{
  float x,y,z;                    //包围盒最小的顶点
  float L;                        // 边长
```

收稿日期:2008 - 03 - 04;**修回日期**:2008 - 05 - 04。
基金项目:国家自然科学基金资助项目(60475021);河南省杰出青年基金资助项目(0412000400);洛阳市科技攻关计划项目(0701041A)。
作者简介:黄森(1982 -),女,河南社旗人,硕士,主要研究方向:计算机图形图像处理; 张海朝(1963 -),男,河南宜阳人,副教授,主要研究方向:计算机图形图像处理、系统工程、智能控制系统; 李超(1980 -),男,河南鹿邑人,硕士研究生,主要研究方向:嵌入式系统。

```
struct TreeNode * father;                    // 父节点指针
struct TreeNode * child[8];                  //子节点指针
struct PointNode * point;                    // 包含的数据点的指针
String code;                                 // 本节点编码,8 进制
int index;                                   // 本节点在兄弟节点中的序号
bool leaf ;                                  //是否叶子节点
}
```

3)包围盒[5-6]。以 p 点为内点,并且各条棱边均平行于坐标轴的六面体称为 p 点的包围盒,用 (x,y,z,l) 表示,其中 x、y、z 是正六面体的最小顶点的坐标,l 是正六面体的边长。

4) 八叉树。八叉树是依据栅格数据三维空间分布的特点,将整个图像区域($2^n \times 2^n \times 2^n$) 按照八个象限进行递归分割,逐步分解成一系列被单一类型区域内含的立方体区域。八叉树有 3 种不同的存储结构,分别是规则方式、线性方式以及一对八方式。相应的八叉树也分别称为规则八叉树、线性八叉树以及一对八式八叉树。不同存储结构的空间利用率及运算操作的方便性不同。

2　基于八叉树的空间分割算法

2.1　分割算法描述

为了从一个点集 P 出发建立点集之间的邻域结构,本文散乱点集邻域结构构造算法的基本思想为:先确定 k 近邻的 k 值;之后对样点进行一定的预处理,建立样点的包围盒,对包围盒进行基于八叉树结构的分割;广度遍历生成的八叉树,利用数据点的空间分布与包围盒的对应关系,快速搜索出任意点 v 的邻域点集,完成点集邻域结构的构造,以各邻域点集为基础建立各采样点的三角剖分,完成拓扑结构的寻找工作,为下一步的拓扑重建做准备。

采用八叉树来记录散乱点的结构有多方面的好处:1) 八叉树在邻域搜索方面速度很快;2) 八叉树在后期网格优化时能增进优化的方便性;3) 八叉树在图形消隐等方面都有很好的效果。已有的 k 近邻计算方法都不太适用于点集分布不均匀的情况。而八叉树由于进行了递归的空间分割,所以在点集不均匀的情况也适用。

2.1.1　包围盒的计算

立方体包围盒用 (x,y,z,l) 表示,x、y、z 表示最小点的坐标,l 表示立方体的边长。最小点的坐标可根据点集中最小点的坐标获得。根据点集的最大最小坐标可以计算出包围盒的 l, $l = z_{max} - z_{min}$。由于浮点数处理的误差,在实践中将 l 取得稍微大一点,一般都根据包围盒的大小按一定的比例取值,在本文中取比例因子为 1/1 000,即将最小点坐标取为($x_{min} - m/1\,000$, $y_{min} - n/1\,000$, $z_{min} - l/1\,000$)。同时最后取立方体的边长为 $l' = (1 + 1/500)l$。

2.1.2　分割结束条件

本文采用八叉树来记录散乱数据点集的结构,而进行分割的目的是将 k 近邻的计算限制在一个较小的范围以及对数据点结构进行构造。所采用的分割结束条件是判断立方体中点集的数目,如果数目小于一个特定值 m,则结束分割。m 的值与 k 正相关,即要求得每一个点周围的 k 近邻,必须在一个足够大的范围内求得,所以 m 与 k 是正相关的。假设每个立方体子空间中平均数据点的个数 m 是 k 的函数,则有:

$$m = \alpha k \tag{1}$$

假设散乱点集中数据点的数目为 n,分割结束时子立方体数目为 8^i, 则有:

$$i = \lceil \log_8 n/m \rceil \tag{2}$$

$$l_{min} = l/2^i \tag{3}$$

其中,i 的取值为向上取整,i 表示八叉树中最下层叶子节点的深度。l_{min} 为最小子立方体的边长,l 为立方体包围盒的边长。

通过实验分析,对于不同疏密程度和不同拓扑结构的三维散乱数据点集,都可以给出统一的接近于最佳 k 近邻搜索速度的 α 值。

2.1.3　包围盒分割

以立方体包围盒的最小点为坐标原点,对点集进行坐标平移。接下来就对包围盒进行分割。我们采用线性八叉树对包围盒进行空间分割。首先,将该立方体包围盒作为八叉树的根节点, 然后把该立方体包围盒分割成大小相同的八个子立方体,且每个子立方体均被视为根节点的子节点,由此划分下去,将包围盒分割成很多个小立方体。

分割规则:如果子立方体内包含散乱数据,则该长方体记为“黑体”;如果子立方体内不包含散乱数据,则该子立方体记为“白体”,对于黑体, 如果包含的点集数目大于 m,则需要进一步分割, 重复此步骤,直到所有的子立方体中散乱点集数目都小于等于 m。

一个立方体按分割规则分割为 8 个子立方体,设立方体为 (x,y,z,l), 将立方体分割开得 8 个子长方体,分别为:

$$\begin{cases} A0:(x,y,z,l/l) \\ A1:(x+l/2,y,z,l/2) \\ A2:(x,y+l/2,z,l/2) \\ A3:(x+l/2,y+l/2,z,l/2) \\ A4:(x,y,z+l/2,l/2) \\ A5:(x+l/2,y,z+l/2,l/2) \\ A6:(x,y+l/2,z+l/2,l/2) \\ A7:(x+l/2,y+l/2,z+l/2,l/2) \end{cases} \tag{4}$$

在这里采用线性八叉树来记录点集及其结构,本文采用 FD 线性八叉树表示法,因为这种表示方法的算法效率是最高的。在 FD 线性八叉树中,每个叶节点用其 FD 位置码来表示。因为非叶节点的 FD 位置码可以计算出来,为了节省存储空间, 不存储非叶节点。叶节点的深度即为位置码的长度。

对于一个长方体包围盒进行上述规则的 8 等份划分, 则可利用 0 ~ 7 八进制编码序号的特点,将任一个节点的位置与一个 8 进制数唯一对应,即:

$$Q = q_{n-1}8^{n-1} + q_{n-2}8^{n-2} + \cdots q_2 8^2 + q_1 8^1 + q_0 8^0 \tag{5}$$

其中,n 为节点所处的层数(即节点深度);q_j 为八进制, $q_j \in [0,7]$, $j \in \{0,1,\cdots,i-1\}$, q_j 表示该节点在其兄弟节点之间的序号,即该叶节点在父节点中所处的象限。这样,从 q_0 到 q_i 完整地表示出了从根节点到该节点的路径。对于如图 1 所示的八叉树,5 个“黑体” 叶子节点的编码分别为:第 1 层“黑体” 叶子节点为:4;第 2 层“黑体” 叶子节点为:12,62; 第 3 层“黑体” 叶子节点为:632,637。

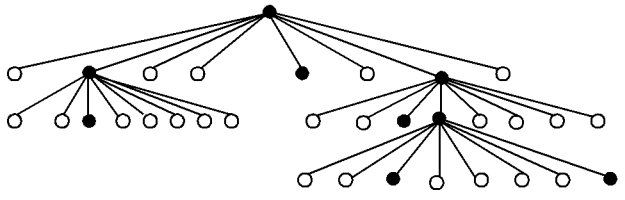


图 1　八叉树结构

对散乱点集中的任意一个数据点都可根据其空间中的坐标 (x,y,z) 计算出它的八叉树编码。

首先由数据点的坐标 (x,y,z) (x',y',z') 得到点所在包围盒的空间位置序号坐标 (x',y',z'), 即:

$$\begin{cases} x' = \lfloor x/(l/2^i) \rfloor \\ y' = \lfloor y/(l/2^i) \rfloor \\ z' = \lfloor z/(l/2^i) \rfloor \end{cases} \tag{6}$$

其中,i 可由式(2)求得,x',y',z' 取值都是向下取整。(x',y',z')都可以表示成二进制的数,即为二进制的多项式:

$$\begin{cases} x' = a_{i-1}2^{i-1} + a_{i-2}2^{i-2} + \cdots + a_1 2^1 + a_0 2^0 \\ y' = b_{i-1}2^{i-1} + b_{i-2}2^{i-2} + \cdots + b_1 2^1 + b_0 2^0 \\ z' = z_{i-1}2^{i-1} + z_{i-2}2^{i-2} + \cdots + z_1 2^1 + z_0 2^0 \end{cases} \tag{7}$$

又由式(4)可以推得:

$$\begin{cases} x' = \sum_{j=0}^{i-1} (q_j \bmod 2) \times 2^j \\ y' = \sum_{j=0}^{i-1} (q_j/2 \bmod 2) \times 2^j \\ z' = \sum_{j=0}^{i-1} (q_j/4 \bmod 2) \times 2^j \end{cases} \tag{8}$$

由式(8)可以得出编码为:

$$q_j = c_j 2^2 + b_j 2^1 + \alpha_j 2^0 \tag{9}$$

2.2 近邻结构的建立

要建立散乱点的邻域结构即是要求出每个节点的 k 近邻,在本文的算法中也就是要求出包围盒中每一个点的 k 近邻,由于本文采用八叉树来进行空间分割,即点集 k 近邻的搜索将转化为八叉树近邻的搜索。本文的搜索算法思想为根据散乱点数据的空间位置得到点所处的空间子立方体,也就是对八叉树中的一个叶节点,根据八叉树近邻节点搜索得到该点的 k 近邻。

八叉树中的每一个节点都对应包围盒空间中的一个子立方体。如果两个立方体区域相邻,那么它们对应的八叉树中的两个节点互为临近点。求点的 k 近邻不仅要在该点所在的立方体中搜索,而且要在该立方体相邻的立方体中搜索,则一共要在27个立方体中寻找。八叉树近邻点分为三类:点近邻、边近邻和面近邻。寻找点的 k 近邻即要在该点所在立方体节点的所有八叉树邻接点中搜索。

每个八叉树叶子节点的临接点可根据该节点的编码直接获得,即周围26个临接立方体的空间位置坐标(X,Y,Z)为:

$$\begin{cases} X = x' \pm 1 \\ Y = y' \pm 1 \\ Z = z' \pm 1 \end{cases} \tag{10}$$

式中 X, Y, Z 不能同时与 x',y',z' 相等。

由邻接子立方体的空间位置坐标(X,Y,Z),根据式(9)可求得周围26个子立方体的位置编码,由位置编码即可直接得到邻接立方体及立方体中的点集。

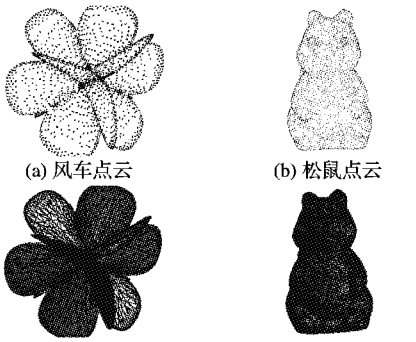
近邻结构的建立过程即是对八叉树的一次深度遍历及处理过程,对八叉树中的每一个为黑体的叶节点进行处理,对叶节点中每个点进行处理。以点 v 为例:1)在该点所在的叶节点中计算最近邻点(以 k 个为结束);2)在周围26个相邻子立方体中搜索最近邻点。按以上步骤深度遍历处理八叉树中的叶节点,最后得到所有散乱点数据的 k 近邻 $MNV|v|$。在判断点之间的距离时,采用欧氏距离的平方来进行比较,使计算简化。

同时,在上述的 k 近邻搜索过程中,本文算法采用了"剪枝策略"来缩小搜索范围。即根据第 k 个最近邻点与点 v 的距离平方 t 来判断是否该在各个相邻的子立方体中搜索。搜索原则:1)如果相邻节点不是"黑体"节点,则不对该节点进行搜索;2)如果为点相邻节点,则计算 v 点和该相邻点的距离平方 s,如果 $s \geqslant t$,则说明该子立方体中不存在 k 近邻点,不对该立方体进行搜索;3)如果为边相邻点,则计算 v 点和该相邻边的距离平方 s,如果 $s \geqslant t$,则说明该子立方体中不存在 k 近邻点,不对该立方体进行搜索;4)如果为面相邻节点,则计算 v 点和该相邻面的距离平方 s,如果 $s < t$,则说明该子立方体中不存在 k 近邻点,不对该立方体进行搜索;5)在搜索完所有相邻点对应的子立方体后得到的近邻的数据有可能小于我们设定的一个数 d,这时认为该点为噪声点,对其予以删除。

3 实验分析

3.1 算法结果分析

为了验证该算法的有效性和正确性,我们进行了两组实验。机器配置为1GB RAM,AMD 3400 + GHz,实验数据为真实物体的点云数据如图2(a)和(b)所示,它们的重建效果分别如图2(c)和(d)所示。



(a) 风车点云 (b) 松鼠点云 (c) 风车重建效果 (d) 松鼠重建效果

图2 实验

实验1 α 值的最佳取值实验(如表1所示)。

实验数据为风车点云数据和松鼠点云数据。分别取不同疏密程度的点云数据进行测试,在测试过程中取不同 k 值和 α 值进行测试。从实验结果可以看出,该算法对于不同物体、不同疏密程度、不同 k 值来说,最佳搜索速度的 α 值范围比较集中为1.5 ~ 3.0。

实验2 不同 k 值实验。

为检验本文算法对不同 k 值的适应性,取不同的 k 值来进行实验。实验数据为长方体点云数据。根据实验1取 α 值为1.6,从实验结果看出,搜索时间随着 k 的增加近似于线性增长,如表2所示。

3.2 算法时间复杂度分析

本文算法的各个主要步骤的标准时间复杂度。

1)计算最小包围盒的边长的时间为:

$$T = n/m \times C_m^2 \times t \tag{11}$$

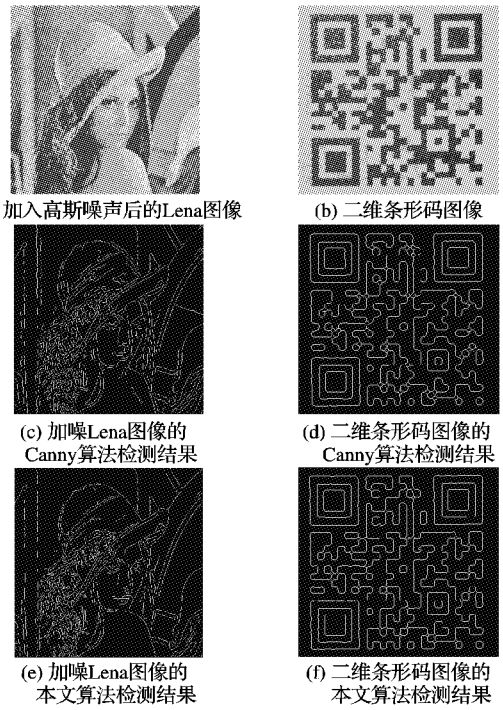
其中,n 为数据点的总点数,m 为子域大小,t 为求两点之间距离所用的时间。

2)求所有点的 m 邻域为 $O(m \times n)$。

由于本文采用的包围盒分割方法中仅仅对包含有数据点的"黑体"进行再分割,同时用八叉树记录分割过程,在建立数据点的拓扑关系时,仅搜索空间中实际最近邻的包围盒,实验证明只要所给的散乱点集足够密,本文方法可以快速准确地完成曲面重建中的数据分割工作,建立数据点的拓扑关系,并能显著降低密集散乱点云的几何建模所花费的时间。以图2所示的点云数据为例,仅生成10 080个点的 $m(m=6)$ 邻域这一步,如果采用文献[3]方法,需要16min,而采用

(下转第2051页)

示)。在对二维条码图像的边缘检测中可以看到,用 Canny 算法检测出的图像使得角点变圆(如图(d)的最底部中间部分、右下角等),而本文算法检测结果有着明显的改善,提取的边缘基本上能保持原来的形状(如图(f)所示)。



(a) 加入高斯噪声后的Lena图像　(b) 二维条形码图像
(c) 加噪Lena图像的Canny算法检测结果　(d) 二维条形码图像的Canny算法检测结果
(e) 加噪Lena图像的本文算法检测结果　(f) 二维条形码图像的本文算法检测结果

图 1　检测结果及比较

4　结语

本文提出了一种基于各向异性扩散方程的 Canny 边缘检测算法。将 Canny 算法中的高斯滤波替换成各向异性扩散方程,并对去噪后的图像使用线性灰度变换以增强图像。改进后的 Canny 算子边缘检测算法提高了信噪比和定位精度,检测到的伪边缘显著减少,角点保持比较好,取得了更好的效果。

参考文献:

[1] 张小琳. 图像边缘检测技术综述[J]. 高能量密度物理, 2007(1): 37 - 40.

[2] 季虎, 孙即详, 邵晓芳, 等. 图像边缘提取方法及展望[J]. 计算机工程与应用, 2004, 40(14): 70 - 73.

[3] CANNY J. A computational approach to edge detection[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence. 1986, 8(6): 678 - 698.

[4] 余洪山, 王耀南. 一种改进型 Canny 边缘检测算法[J]. 计算机工程与应用, 2004, 40(20): 27 - 29.

[5] 麦特尔. 现代数字图像处理[M]. 孙洪, 译. 北京: 电子工业出版社, 2006: 156.

[6] PERONA P, MALIK J. Scale-space and edge detection using anisotropic diffusion[J]. IEEE Transactions on Patter Analysis and Machine Intelligence, 1990, 12(7): 629 - 639.

[7] 梅跃松, 杨树兴, 莫波. 基于 Canny 算子的改进的图像边缘检测方法[J]. 激光与红外, 2006, 36(6): 501 - 503.

[8] 范彦革, 刘旭敏. 各向异性扩散的研究[J]. 计算机工程与应用, 2006, 42(29): 55 - 59.

[9] 张小洪, 杨丹, 刘亚威. 基于 Canny 算子的改进型边缘检测算法[J]. 计算机工程与应用, 2003, 39(29): 113 - 115.

(上接第 2048 页)

$200 \times 200 \times 200$ 的栅格对数据空间进行划分,运行这一步只需要 27 s。从式(11)可以看出,m 邻域大小的选取关系到拓扑关系的品质以及算法的效率。m 取值过大,在曲率较大而采样点又不十分密集的情况下,会使切平面受到非当前局部区域测量点的影响,m 取值过小,测点噪声的影响变大,不利于后续的曲面重建工作。因此需要在二者之间进行平衡。在实践中发现,将 m 取为 6 ~ 12,一般都能取得较好的结果。

表 1　α 的最佳取值实验结果

点数	k 值	α 值								
		0.2	0.5	1.0	2.0	3.0	5.0	8.0	12.0	20.0
2 500	10	1.178	0.415	0209	0.210	0.237	0.298	0.380	0.511	0.674
	20	1.925	1.513	0.956	0.423	0.395	0.447	0.495	0.607	1.538
	50	1.831	0.836	0.918	1.112	1.203	1.578	1.584	2.102	4.112
5 000	10	0.970	0.398	0.224	0.220	0.211	0.305	0.338	0.397	0.698
	30	1.635	0.832	0.413	0.394	0.400	0.578	0.592	0.612	0.932
	50	2.006	0.944	0.711	0.691	0.686	0.712	0.761	0.982	1.044
7 500	10	0.689	0.281	0.287	0.294	0.370	0.414	0.448	0.532	0.620
	30	1.213	0.537	0.408	0.412	0.434	0.492	0.500	0.676	0.829
	50	1.934	0.931	0.716	0.717	0.720	0.831	0.892	0.935	1.354
15 000	10	0.895	0.402	0.397	0.400	0.452	0.508	0.532	0.640	0.751
	30	1.354	0.903	0.530	0.528	0.524	0.603	0.697	0.714	1.313
	50	2.771	1.834	1.101	1.102	1.273	1.544	1.967	2.299	2.953
25 000	10	0.932	0.491	0.474	0.432	0.405	0.481	0.530	0.751	1.112
	30	1.305	0.682	0.516	0.520	0.592	0.653	0.733	0.910	1.526
	50	2.130	1.534	1.121	1.118	1.253	1.400	1.924	2.110	3.327

表 2　不同 k 值实验结果

k	4	6	10	30	50	80	120
搜索时间 /ms	0.196	0.202	0.221	0.394	0.685	1.126	1.975

4　结语

本文采用 k 近邻搜索算法利用八叉树进行包围盒空间分割,近邻点的搜索只局限于采样点所在的包围盒及其周围的包围盒,并采用了剪枝策略,使近邻点的搜索范围大大缩小,对噪声点也可以自动剔除,搜索速度快,并且可以方便后续的数据处理工作,比如说网格简化时可简单地进行点集合并、模型显示时也可方便地进行消隐处理。

参考文献:

[1] HOPPE H, DEROSE T, DUCHAMP T. Surface reconstruction from unorganized poins[J]. Computer Graphics, 1992, 26(2): 71 - 78.

[2] GUO B, MENON J, WILLETTE B. Surface reconstruction using alpha shapes[J]. Computer Graphics Forum, 1997, 16(4): 177 - 190.

[3] AMENTA N, BERN M, KAMVYSSELIS M. A new voronoi-based surface reconstruction algorithm[C]// Proceeding of International Conference on Computer Graphics and Interactive Techniques 1998. NewYork: ACM Press, 1998: 415 - 421.

[4] 王青. 散乱数据点的增量快速曲面重建算法[J]. 软件学报, 2000, 11(9): 1221 - 1227.

[5] PIEGL L A, TILLER W. Algorithm for finding all k nearest neighbors[J]. Computer Aided Design, 2002, 34(2): 167 - 172.

[6] GOODSELL G. On finding p-th nearest neighbors of scattered points in two dimensions for small p [J]. Computer Aided Geometric Design, 2000, 17(4): 387 - 392.