三维散乱数据点集 k 近邻的快速搜索算法

伍爱华

（长沙民政职业技术学院，湖南 长沙 410004）

摘 要：从数据点的空间排列特点出发提出了 k 近邻搜索算法，利用多向链表对数据集进行排序，综合考虑了数据集的范围、点的总数、搜索步长及最近点数目 k，并采用了空间包围策略，可以给出接近于最佳搜索速度的步长 e 和 k 值，并且在搜索终止准则上进行改进，使近邻点的搜索范围大大缩小，搜索速度加快。

关键词：k 近邻；多向链表；快速搜索

中图分类号：TP301.6 **文献标识码**：A **文章编号**：1673-9833(2007)02-0084-04

Algorithm for Finding *k*-Nearest Neighbors of Scattered Points Set in Three Dimensions

Wu Aihua

(Changsha Social Work College，Changsha 410004，China)

Abstract：An algorithm to finding the *k*-nearest neighbors of points is provided quickly which is based on the space character of data-points. Data set is queued by multilinked list. By using envelopment-space, the range of data set , the total numbers of points, the searching step and the numbers of nearest neighbors, the method is easy to obtain the value of *e* and *k* for a nearly optimal searching. By improving the terminate rule of searching, this method has some excellent characters such as shorten searching range and quicken speed.

Key words: *k*-nearest neighbors; multilinked list; quickly search

1 问题的提出

曲面重建技术在虚拟现实、科学计算可视化、雕塑曲面造型、逆向工程等领域有着广泛的应用前景。作为最具普遍性的曲面重建问题，散乱点集曲面重建无论在理论上还是在实际应用上都有重要意义。现代的测量数据方法，已经使得数据量达到了几兆、几十兆甚至上百兆。由于数据量大，所以无论用什么方法来处理，首先都必须解决如何高效处理庞大的三维点群数据(特征点、临界点或边界点的搜索)。原始的测点数据之间没有相应的、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，在几兆、几十兆无序测点数据集合中遍历搜寻是造成三维散乱点群几何建模速度很慢的主要原因。因此，建立测量点群之间的几何拓扑(空间位置)关系，是提高密集散乱点群几何建模速度的关键。而三维散乱点群空间划分的拓扑关系是建立在预处理点集间的邻域结构基础上的。

对点集预处理就是搜索某点的 k 个最近邻域（简称 k 近邻）。k 近邻的计算有很多方法，通常计算某点 k 近邻的方法是求出候选点到其余 $n-1$（总点数为 n）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，但是真实的数据集的规模往往很大，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，这些方法可分为3类：1)利用点集 Voronoi 图来进行 k 个最近点的搜索，但点集的 Voronoi 图的计算量仍然非常大。2)利用空间分块策略进行 k 个最近点的搜索[1-3]。3)利用八叉树来编码建立包围盒进行 k 个最近点的搜索[4,5]。但这些文献的方法既不能保证它的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

收稿日期：2007-02-16

作者简介：伍爱华（1972-），女，湖南邵阳人，长沙民政职业技术学院讲师，主要研究方向为计算智能.

近邻域，而且有的学者只研究了平面点集的 k 个最近邻域搜索问题。

本文提出一种基于多向链表（Multilinked list）的 k 近邻快速搜索算法。该算法从数据点的空间排列特点出发，综合考虑了数据集的范围、点的总数、搜索步长及最近点数目 k，可以给出接近于最佳搜索速度的步长和 k 值，并且在搜索终止准则上进行改进，以便进一步提高 k 近邻的搜索速度。

2　相关概念

为了便于散乱点搜索算法的描述和讨论，先给出一些相关的基本概念。

2.1　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。

2.2　点的邻域及 k 近邻

点的邻域在散乱点群三维重建中是一个重要的概念，其定义为：设 $d(X，P_i)$为点 X 到点 P_i 的距离，则点 P_i 的邻域 V_i 可定义为：

$V_i=\{X\in R^n \mid d(X,P_i)<d(X,P_j), i\neq j\}$，$1\leq i\leq N$。这表明，邻域 V_i 中的点到 P_i 的距离都小于到点集中其它点的距离。

给定一个曲面散乱点集 $P=\{P_i(x_i，y_i，z_i)，i=1，2，\cdots，n\}$称邻域 V_i 中距点 P_i 最近的 k 个点为点 P_i 的 k 近邻。它反应了该点 P_i 的局部信息，k 近邻中的每个点称为点 P_i 的邻近点。

2.3　多向链表（Multilinked list）

链表是动态数据结构的一种基本形式。链表中的数据元素可以用任意的存储单元来存储，逻辑相邻的两元素的存储空间可以是不连续的。为表示逻辑上的顺序关系，对表的每个数据元素除存储本身的信息之外，还需存储一个指示其直接后继和前驱（单链表除外）的信息。这两部分信息组成数据元素的存储映象，称为结点。链表就是把若干个结点链成了一串。链表中的结点可增可减，可多可少，可在中间任意一个位置插入和删除。

用多向链表表示点集的基本思想是：对每个点存储为一个结点，结点由数据域和指针域组成，其中：数据域 x、y、z 分别存储该结点的 x、y、z 轴坐标值，数据域 flag 存储访问标记，指针域 x_{prio}，x_{next}，y_{prio}，y_{next}，z_{prio}，z_{next} 分别存储当前结点在 3 个坐标轴方向的前驱和后继，指针 eighbor_set 存储数据集的 k 个最近邻域集的头指针。每增加或者删除一个点，6 个指针域的值都要作相应的改变。

3　算法

在三维空间中，某候选点的 a 邻域是以该点为球心半径为 a 的球体。但是，真实的数据集的规模往往很大，而且对每个点都要求出其与候选点的欧氏距离，由于欧氏距离计算中有大量的加法和乘法运算，大大增加了计算的时间和复杂度，因此用它来判断数据点是否在 a 邻域内的计算量很大。考虑到计算机作比较运算是非常快的，并且在正方体中非常容易作比较运算。并且若该球体内数据点的个数不多于 k 个，则该球体内接正方体所包含的数据点集必定是该点 k 近邻的子集。同样的原理，把球放大成其外切正方体，若其外切正方体内包含的数据点个数也不多于 k 个，则该球的内接正方体所包含的数据点集必定是该点 k 近邻的子集。

这样，只通过简单的比较运算，我们筛选出所有满足条件的 k 近邻子集。接下来，只要计算少量两个正方体间数据点与候选点间的欧氏距离，就能找到候选点的所有 k 近邻。

基于这个思想，首先根据数据集的特点构造有 x、y、z 3 个方向按升序排列的多向链表，在逻辑上反映了数据集的三维拓扑结构；然后使数据集的每个点归入到相应的结点，最后利用多向链表内结点的信息对每个候选点进行 k 个最近邻域的搜索。在搜索的过程中，根据点集的稠密程度及 k 值对搜索步长进行修正。

3.1　基本数据结构

3.1.1　数据点表

数据点表保存数据点集的坐标信息，所有点按原有的顺序排列，表中每个元素结构如下：

```
typedef struct points{
  float x，y，z；// 数据点的三维坐标值；
    struct points *next；// 数据点的下一个点；
}POINTS。
```

3.1.2　多向数据链表

该表保存每个数据点的坐标信息及其在坐标轴方向的前驱和后继，其结构如下：

```
typedef struct mul_list_node {
   POINTS *point ；// 数据点的指针，指向数据点；
int flag；// 数据点的访问标记；
   bool visited；// 标记此结点是否为候选点 k 近邻元素， true 表示是，false 表示否；
   POINTS *x_prio,*y_prio,*z_prio；//结点在x，y，z 3 个方向上的前驱指针；
   POINTS *x_next,*y_next,*z_next；//结点在x，y，z 3 个方向上的后继指针；
   POINTS *neighbor_set ；// 数据集的 k 个最近邻域集的头指针；
```

}MULLISTNODE,*MULLISTNODE。

3.1.3 数据点的 k 个最近邻域链表

该表保存数据点 k 近邻结点信息，表中每个元素的结构如下：

typedef struct neig_list{
POINTS *point；// 数据点的指针，指向数据点；
POINTS *prior；// prior 指向此近邻的前一个结点；
POINTS *next；// next 指向此近邻的后一个结点；
}NEIGLIST。

3.1.4 数据点的邻域距离表

该表保存数据点近邻结点的距离信息，表中每个元素的结构如下：

typedef struct dist_list{
POINTS *point；// 数据点的指针，指向数据点；
float dist；// 数据点与候选点间的距离；
POINTS *prior；// prior 指向此近邻的前一个点；
POINTS *next；// next 指向此近邻的后一个点；
}DISTLIST。

3.2 k 个最近邻域的搜索

利用上面得到的多向数据链表，对每个候选点 P_i 搜索 k 个最近邻域的步骤如下：

Step1 记下链表中指向候选点 P_i 的一个指针值（有多种方法取，不妨为 P_i->x_{prio}->x_{next}），然后根据三维坐标值和初始步长 $h = e$，算出其外层正方体坐标；

Step2 先后以 x、y、z 轴为索引（x 轴的优先级别最高，y 轴次之，z 轴最末）向前向后搜索，把在正方体内的点进行标记为 $flag$ =1，并记下正方体内数据点个数 num_1，标记搜索方向 dir =1；

Step3 $temp$=h，若 $num_1 \le k$ 且 dir =1，取步长 $h = \sqrt{3}*h$；否则 $h = h/\sqrt{3}$；算出其外层正方体坐标，先后以 x、y、z 轴为索引（x 轴的优先级别最高，y 轴次之，z 轴最末）向前向后搜索，把在正方体内且 $visited$=$false$ 的点标记为 $flag$=$flag$+2，并记下正方体内数据点个数 num_2；

Step4 若 $num_1 \ge k$，令正方体内所有结点 $flag$=1，num_1=num_2，dir=−1；若 $num_1 \le k$ 且 $num_2 \ge k$，令 dir=−1，h=$temp$；若 $num_1 \le k$ 且 $num_2 \le k$，则所有标记 $flag$=3 的点所构成的集合为候选点 k 近邻的子集，标记为 $visited$=$true$，并把结点记入候选点的最近邻域链表，同时令正方体内所有结点 $flag$ 值为 1，k=k−num_1，num_1=num_2，若 dir=−1，跳到 Step6；

Step5 返回 step3;

Step6 在当前正方体内搜索与候选点最近的 k 个点，按距离升序的方式，记录 k 个邻近点与候选点间的距离及指针，若某邻近点与候选点距离小于 h，则对这个点进行标记，同时对访问过的正方体进行标记；

Step7 在当前正方体内，如果候选点的 k 个最近邻域已找到，并且候选点到第 k 个最近点的距离小于 h，则候选点的 k 个最近邻域搜索结束，记入候选点的最近邻域链表，并对已标记的点以及正方体进行复位，以便于下一个候选点的 k 个最近邻域搜索；否则，正方体的搜索步长 h=h+e，返回 Step6 继续搜索。

4 实验结果

为了验证本文算法的正确性和有效性，我们进行了两组对比实验，实验数据为不规则的单峰曲面点云(见图 1)$z = \sin\left(\sqrt{x^2+y^2}\right)\Big/\sqrt{x^2+y^2}$和 peaks 曲面点云(见图 2)，所有实验都是在赛扬 1.7G 的 PC 机上完成的，测试时间为搜索一个点的 k 个最近邻域的平均 CPU 运行时间。

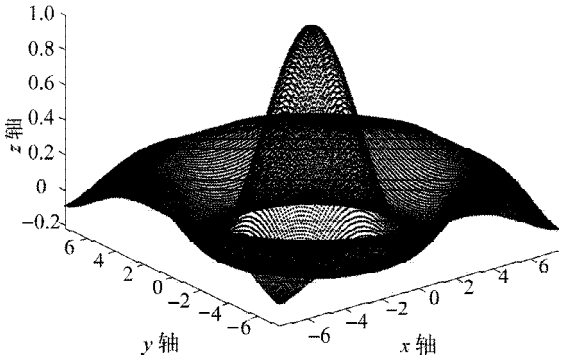

图 1 单峰曲面点云

Fig. 1 Scattered Points of single-peak surface

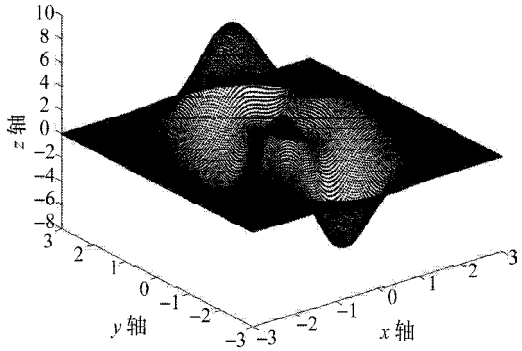

图 2 peaks 曲面点云

Fig. 2 Scattered Points of multi-peak surface

实验 1 e 值的最佳初值实验。

实验数据为不规则曲面点云数据。分别取不同疏密程度的点集进行测试，在测试过程中取不同 k 值和 e 值进行测试。从实验结果可以看出，本文的算法对于不同曲面、不同疏密程度、不同 k 值来说，最佳搜索速度的 e 值范围比较集中，为

$$e = \min\left(\frac{(x_{\max}-x_{\min})*k}{a*n}, \frac{(y_{\max}-y_{\min})*k}{a*n}, \frac{(z_{\max}-z_{\min})*k}{a*n}\right)。$$

其中 $x_{\max}$，$x_{\min}$，$y_{\max}$，$y_{\min}$，$z_{\max}$，$z_{\min}$ 为点云在 x、y、z 方向上的最大值和最小值，n 为点的总个数，a 为步长调节因子，a 的最佳值为 6 ~ 8(如表 1 所示)。

表1　不同a和k值实验结果

Table 1　Experiment data under different a and k　　s

曲面	点数	k值	a						
			1	4	6	8	10	12	14
单峰曲面	2 500	8	1.255	0.520	0.321	0.322	0.348	0.405	0.485
		10	1.977	1.576	1.041	0.526	0.500	0.551	0.594
		20	1.884	0.923	1.002	1.191	1.277	1.640	1.644
	3 600	8	1.055	0.500	0.336	0.332	0.322	0.412	0.446
		10	1.691	0.921	0.516	0.500	0.504	0.676	0.690
		20	2.047	1.029	0.804	0.784	0.781	0.806	0.852
	6 400	8	0.779	0.391	0.397	0.403	0.476	0.519	0.552
		10	1.285	0.638	0.513	0.517	0.538	0.594	0.602
		20	1.983	1.015	0.808	0.809	0.814	0.919	0.979
	4 000	8	0.979	0.521	0.525	0.548	0.613	0.695	0.717
		10	1.709	1.016	0.715	0.699	0.717	0.777	0.845
		20	2.984	1.501	1.285	1.286	1.302	1.593	1.727
Peaks曲面	9 000	8	0.981	0.505	0.502	0.504	0.555	0.609	0.632
		10	1.422	0.990	0.631	0.629	0.625	0.700	0.791
		20	2.788	1.887	1.181	1.180	1.345	1.608	2.015
	16 000	8	1.015	0.593	0.577	0.535	0.509	0.583	0.630
		10	1.373	0.775	0.618	0.620	0.690	0.749	0.826
		20	2.176	1.230	1.198	1.195	1.327	1.469	1.974

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a值为8，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。

表2　不同k值实验结果

Table 2　Experiment data under different k

k值	6	8	10	30	50	70	90
搜索时间t	0.376	0.504	0.629	1.915	3.204	4.431	5.699

5　结束语

本文提出的k近邻搜索算法从数据点的空间排列特点出发，利用多向链表对数据集进行排序，综合考虑了数据集的范围、点的总数、搜索步长及最近点数目k，并采用了空间包围策略，可以给出接近于最佳搜索速度的步长和k值，并且在搜索终止准则上进行改进，使近邻点的搜索范围大大缩小，搜索速度快，并使后续的处理方便，如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] 单东日，柯映林. 基于二维Delaunay近邻的空间散乱数据曲面重建算法[J].中国机械工程，2003，14(9)：756-759.

[2] 周儒荣，张丽艳. 海量散乱点的曲面重建算法研究[J]. 软件学报，2001，12(2)：249-255.

[3] 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，2004，7(7)：909-912.

[4] 余　迁. 基于散乱数据的曲面重构研究[D]. 西安：西北工业大学，2006.

[5] 朱心雄. 自由曲线曲面造型技术[M]. 北京：科学出版社，2000：215-233.

[6] 刘晓东，刘国荣，王　颖，等. 散乱数据点的k近邻搜索算法[J]. 微电子学与计算机，2006，23(4)：23-30.