文章编号:1671-8860(2009)05-0615-04　　文献标志码:A

# 一种基于立方体小栅格的 K 邻域快速搜索算法

赵俭辉[1]　龙成江[1]　丁乙华[1]　袁志勇[1]

(1　武汉大学计算机学院,武汉市珞喻路 129 号,430079)

**摘　要**:提出了一种新的基于立方体小栅格的 $K$ 邻域搜索算法。首先,采用二次划分的方法将点云划分到相应的立方体小栅格中;然后,为采样点所在的立方体小栅格确定最终子空间、内子空间和外子空间,结合采样点的球空间,就能很快确定该采样点的 $K$ 邻域的搜索范围。与已有方法相比,该算法具有更高的搜索效率。

**关键词**: $K$ 邻域;三维点云;立方体小栅格;搜索算法

**中图法分类号**:P208

通过计算采样点的 $K$ 邻域来确定散乱点云的几何拓扑关系,是数据处理过程中常见的预处理方法,它广泛应用于估算采样点的法向大小[1]、噪点的滤波处理[2]、曲面的光顺处理[3]等过程之中。对于海量数据而言,计算所有采样点的 $K$ 邻域相当耗时,在时间上难以满足实际需要。因此,很多学者从提高速度的角度进行了大量的研究,提出了许多解决方法[4-10]。文献[7]提出的 KNN 算法基于八叉树的索引结构,并利用了一个重要的思想:凡在同一区域的采样点,它们邻域的搜索范围有着一定程度的重叠。本文提出的算法则是基于立方体小栅格,同时借鉴了文献[7]的思想。与已有的方法相比,本文算法可以更快地找到相邻采样点,从而提高了搜索速度。

## 1　算法与实现

### 1.1　相关定义

设点云数据中 $x$、$y$、$z$ 坐标值的最小值分别为 $x_{\min}$、$y_{\min}$、$z_{\min}$,最大值分别为 $x_{\max}$、$y_{\max}$、$z_{\max}$,且所有立方体小栅格的棱长均为 $L$,则索引号为 $(i,j,k)$ 的立方体小栅格可以表达为:$x = x_{\min} + iL$, $y = y_{\min} + jL$, $z = z_{\min} + kL$,其中,$(x,y,z)$ 为该立方体小栅格的左下角顶点。基于上述表达,本文算法给出了以下 5 个三维空间的定义。

子空间 $S(r)$:将 $\{(i,j,k) \mid \max(|i-i_0|, |j-j_0|, |k-k_0|) \leq r\}$ 称为由指定的立方体小栅格 $(i_0,j_0,k_0)$ 扩展 $r$ 倍棱长 $L$ 后得到的子空间,如果子空间的左下角和右上角顶点分别属于立方体小栅格 $(i_1,j_1,k_1)$ 与 $(i_2,j_2,k_2)$,则该子空间又可表示为 $(i_1,j_1,k_1,i_2,j_2,k_2)$。

球空间 $S_{\text{sphere}}$:设点 $O$ 为指定的立方体小栅格 $(i_0,j_0,k_0)$ 内任意位置点,则称以点 $O$ 为中心、$R$ 为半径的球形空间为点 $O$ 的球空间。

最终子空间 $S_{\text{ultimate}}$:由指定的立方体小栅格 $(i_0,j_0,k_0)$ 扩展 $s$ 倍棱长 $L$ 后,如果所得的子空间 $S(s)$ 包含小栅格 $(i_0,j_0,k_0)$ 内任意位置点的 $K$ 邻域,则称该子空间为最终子空间。

内子空间 $S_{\text{in}}$:由指定的立方体小栅格 $(i_0,j_0,k_0)$ 扩展 $t$ 倍棱长 $L$ 后,如果所得的子空间 $S(t)$ 含有的所有采样点都属于小栅格 $(i_0,j_0,k_0)$ 内任意位置点的 $K$ 邻域,则称该子空间为内子空间。

外子空间 $S_{\text{out}}$:最终子空间 $S_{\text{ultimate}}$ 与内子空间 $S_{\text{in}}$ 的差集为外子空间。

### 1.2　棱长计算与点云划分

设点云中采样点的总数为 $N$,包围点云的长方体的长、宽、高分别为 $a = x_{\max} - x_{\min}$、$b = y_{\max} - y_{\min}$、$c = z_{\max} - z_{\min}$,长方体的体积为 $V = abc$。

假定点云均匀分布,即每个采样点占用一个立方体小栅格,则立方体小栅格的棱长为 $L_0 = (V/N)^{1/3}$。那么就可以将包围点云的长方体划分成 $l_0 \times m_0 \times n_0$ 个立方体小栅格,其中 $l_0 = [a/L_0]$, $m_0 = [b/L_0]$, $n_0 = [c/L_0]$。从所有的立方体小栅格中找到含有采样点的小栅格,其总数为

**收稿日期**:2009-03-19。

**项目来源**:国家自然科学基金资助项目(60603079)。

$N_{cube}$。然后,可以计算出含有采样点的小栅格中单位体积内包含采样点的个数,即点云的密度:

$$\rho = \frac{N}{N_{cube} \times L_0^3} \quad (1)$$

点云划分的基本原则为:点云密度越大则小栅格的体积越小,点云密度越小则小栅格的体积越大,以保证立方体小栅格中包含数目适中的采样点。因此,$L$、$K$、$\rho$之间的关系可以表达为:

$$L = \alpha \times (K/\rho)^{1/3} \quad (2)$$

式中,$\alpha$为调控因子,不同类型的点云数据可能使用不同的$\alpha$值。

基于重新计算的棱长$L$,将点云进行第二次划分并重新分配到$l \times m \times n$个立方体小栅格中,其中,$l = [a/L]$,$m = [b/L]$,$m = [c/L]$。

### 1.3 $S_{sphere}$、$S_{ultimate}$、$S_{in}$、$S_{out}$的确定

1) $S_{sphere}$的确定。设点$O$为指定的立方体小栅格$(i_0, j_0, k_0)$中任意位置的点,对立方体小栅格进行逐步扩展,直到扩展$p$倍棱长$L$后得到的子空间$S(p)$中所含的采样点数目第一次大于或等于$K$。设$dx$、$dy$、$dz$($0 < dx \leq L$,$0 < dy \leq L$,$0 < dz \leq L$)分别是点$O$与其所在的立方体小栅格$x$轴方向两个面、$y$轴方向两个面、$z$轴方向两个面的距离的较大值,则可以确定以点$O$为中心的球空间$S_{sphere}$的半径$R$为:

$$R = [(dx + p \times L)^2 + (dy + p \times L)^2 + (dz + p \times L)^2]^{1/2} \quad (3)$$

显然,$S(p)$中任意点与点$O$的距离$d$均满足:

$$d \leq R \leq (p+1)\sqrt{3}L \quad (4)$$

因此,球空间$S_{sphere}$中含有的采样点的数目必然大于或等于$K$,确保$K$个近邻点均在球空间中。

2) $S_{ultimate}$的确定。将指定的立方体小栅格$(i_0, j_0, k_0)$扩展$([(p+1)\sqrt{3}]+1)$倍棱长$L$后得到子空间$S([(p+1)\sqrt{3}]+1)$,显然,$S(p) \subset S([(p+1)\sqrt{3}]+1)$。而$S(p)$中任意采样点与点$O$的距离$d$都小于$([(p+1)\sqrt{3}]+1)L$,这样,立方体小栅格$(i_0, j_0, k_0)$中任一采样点的$K$邻域必在子空间$S([(p+1)\sqrt{3}]+1)$中,即为$S_{ultimate}$。

3) $S_{in}$的确定。当指定的立方体小栅格$(i_0, j_0, k_0)$扩展$(p-1)$倍棱长$L$后得到子空间$S(p-1)$,其中所含采样点的数目小于$K$。因为$S(p-1)$之外的任意点与点$O$的距离都大于$(p-1)L$,所以在$S(p-1)$中与点$O$的距离小于或等于$(r-1)L$的任意采样点必然属于立方体小栅格$(i_0, j_0, k_0)$中任意位置点的近邻点。

设当立方体小栅格$(i_0, j_0, k_0)$扩展$q(q < p-1)$倍棱长$L$后得到的子空间$S(q)$中任意一点与点$O$的距离(由式(4)可知此距离不超过$(q+1)\sqrt{3}L$),都小于$(p-1)L$,则$S(q)$中所有的采样点都必然属于小栅格$(i_0, j_0, k_0)$中任一采样点的$K$邻域,即$S(q)$是所求的内子空间$S_{in}$,则:

$$(q+1)\sqrt{3}L \leq (p-1)L \quad (p、q为非负整数) \quad (5)$$

解得$q < \frac{p-1}{\sqrt{3}} - 1$。当$p < 3$时,$q$无解,即内子空间$S_{in}$为空;当$p \geq 3$时,最大的内子空间$S_{in}$为$S([(p-1)/\sqrt{3}]-1)$。

4) $S_{out}$的确定。由$S_{out}$的定义可得$S_{out} = S_{ultimate} - S_{in}$。当$p < 3$时,$S_{in}$为空,因而在这种情况下对应的外子空间$S_{out}$即为$S_{ultimate}$。

### 1.4 算法的实现

1) 计算立方体小栅格的棱长$L$并进行点云划分。

2) 遍历一次点云数据,对于任一采样点$P$,如果其所在的立方体小栅格已访问过,则直接转入步骤3);否则,计算立方体小栅格的$S_{ultimate}$、$S_{in}$、$S_{out}$,记录立方体小栅格扩展的棱长倍数$p$,并设置此立方体小栅格为已访问过。

3) 计算采样点$P$对应的$S_{sphere}$的半径$R$。

4) 若$S_{in}$非空,则$S_{in}$中的采样点全部为采样点$P$的近邻点,采样点$P$的$K$邻域中剩下的近邻点则从$S_{sphere}$与$S_{out}$的交集中选取;若$S_{in}$为空,则采样点$P$的$K$个近邻点从$S_{sphere}$中选取。

5) 返回步骤2),直到所有的采样点遍历完毕。

## 2 实验结果与算法比较

实验环境为:AMD Sempron(th) 2800+处理器,1.61 GHz主频,448 MB内存,Windows XP系统,Microsoft Visual Stdio.Net 2005集成开发环境,而实验数据是常用的三维点云模型。

### 2.1 $K$邻域结果

如图1所示,从兔子模型上选取了几种不同类型的采样点(标记为"十"字):图1(b)为兔子身体表面的一般采样点;图1(c)为兔耳曲率变化剧烈的脊部点;图1(d)是人为添加的与背部距离明显的悬空点;图1(e)是人为添加的位于两只耳朵之间的悬空点。针对不同类型的采样点,本文算法都准确求出了其$K$邻域,即图中用曲线围起来的区域中的点集。

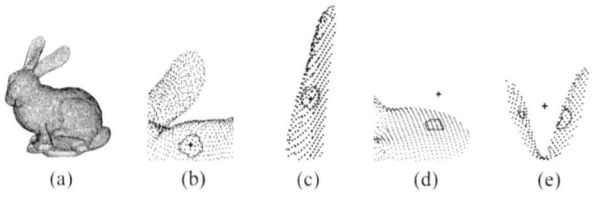


图 1　几种典型采样点及其 $K$ 邻域

Fig. 1　Several Typical Sampling Points and Their $K$-nearest Neighbors

为了进一步测试算法,随机选取了不同形状及不同采样点数目的扫描点云,包括脚、小孩、牙齿、球套等模型,如图 2 所示。

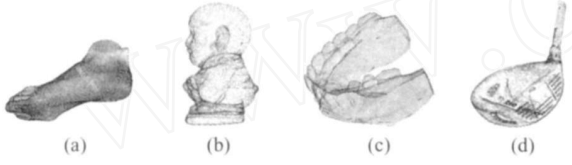


图 2　随机选取的测试数据

Fig. 2　Experimental Data Selected Randomly

### 2.2　调控因子 $\alpha$ 对算法时间的影响

利用图 2 中的点云数据,测试了调控因子 $\alpha$ 的大小对算法执行时间的影响。分别取值 $\alpha$ 为 0.20、0.22、0.24、0.26、0.28、0.30,在这 6 种情况下进行实验,依次计算出在 $K$ 取值为 20、30、50 时不同点云数据中平均每个采样点的 $K$ 邻域搜索所需要的时间。实验结果如表 1所示。

从表 1 的实验数据可以看出,调控因子 $\alpha$ 可以通过立方体小栅格棱长的大小间接地影响 $K$ 邻域的搜索速度。若 $\alpha$ 过大,则扩展的步长(即小栅格的棱长)也会很大,虽然能很快为采样点所在的立方体小栅格确定出最终子空间 $S_{\text{ultimate}}$、内子空间 $S_{\text{in}}$ 和外子空间 $S_{\text{out}}$,但却容易使外子空间 $S_{\text{out}}$ 和采样点的球空间 $S_{\text{sphere}}$ 含有数量过大的采样点,从而降低了算法的效率。若 $\alpha$ 过小,虽然外子空间 $S_{\text{out}}$ 和采样点的球空间 $S_{\text{sphere}}$ 含有采样点的数量少一些,但小栅格扩展的步长过小,扩展的速度太慢,同样也会降低算法的整体效率。

表 1　不同调控因子 $\alpha$ 值对本算法时间的影响/ ns

Tab. 1　Effect of Control Factor $\alpha$ on Computational Cost/ ns

| 点云模型 | 脚 | | | 小孩 | | | 牙齿 | | | 球套 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 点云数量 | 96 713 | | | 32 570 | | | 117 871 | | | 39 479 | | |
| 近邻点数 $K$ | 20 | 30 | 50 | 20 | 30 | 50 | 20 | 30 | 50 | 20 | 30 | 50 |
| 0.20 | 0.148 5 | 0.185 6 | 0.328 9 | 0.153 0 | 0.197 1 | 0.332 8 | 0.177 1 | 0.215 2 | 0.383 6 | 0.164 2 | 0.203 8 | 0.314 6 |
| 0.22 | 0.130 5 | 0.177 5 | 0.310 8 | 0.149 2 | 0.195 2 | 0.323 3 | 0.147 0 | 0.209 8 | 0.380 8 | 0.158 3 | 0.204 6 | 0.332 8 |
| 0.24 | 0.125 2 | 0.172 1 | 0.330 4 | 0.146 3 | 0.190 0 | 0.318 5 | 0.138 0 | 0.192 6 | 0.377 7 | 0.156 3 | 0.208 2 | 0.361 3 |
| 0.26 | 0.122 5 | 0.174 5 | 0.344 6 | 0.143 0 | 0.188 5 | 0.324 3 | 0.134 8 | 0.188 8 | 0.365 7 | 0.155 5 | 0.213 3 | 0.387 1 |
| 0.28 | 0.126 8 | 0.187 4 | 0.332 3 | 0.139 1 | 0.192 8 | 0.347 8 | 0.129 4 | 0.185 7 | 0.361 8 | 0.157 5 | 0.222 8 | 0.417 1 |
| 0.30 | 0.131 2 | 0.196 8 | 0.380 4 | 0.145 3 | 0.209 6 | 0.373 7 | 0.133 6 | 0.187 3 | 0.342 5 | 0.159 5 | 0.230 7 | 0.453 9 |

### 2.3　算法的比较

将本文算法与基于八叉树分割的 KNN 算法[7]和基于 Kd 树的算法[8]进行了比较。实验结果包括 3 组数据:①从模型文件中读取点云并划分到空间存储结构(本文算法使用立方体小栅格存储结构,KNN 算法使用八叉树存储结构,Kd 树算法则使用 Kd 树存储结构)所用的时间;②在已经划分好的空间存储结构中搜索出 $K$ 个近邻点所用的时间;③整个过程所用的时间。如表 2 所示,实验数据为不同 $K$ 值时各个点云数据中平均每个采样点的 $K$ 邻域搜索所需要的时间。

从表 2 的实验结果可以看出,3 种 $K$ 邻域搜索

表 2　三种不同 $K$ 邻域算法的比较/ ns

Tab. 1　Comparison of 3 Different $k$-nearest Neighbors Search Algorithms/ ns

| 点云模型 | | 脚 | | | 小孩 | | | 牙齿 | | | 球套 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 点云数量 | | 96 713 | | | 32 570 | | | 117 871 | | | 39 479 | | |
| 近邻点数 $K$ | | 20 | 30 | 50 | 20 | 30 | 50 | 20 | 30 | 50 | 20 | 30 | 50 |
| 本文算法 | ① | 0.023 4 | 0.023 0 | 0.022 6 | 0.023 5 | 0.023 0 | 0.022 8 | 0.022 9 | 0.022 3 | 0.021 5 | 0.024 1 | 0.023 7 | 0.023 3 |
| | ② | 0.099 1 | 0.149 1 | 0.288 2 | 0.115 6 | 0.165 5 | 0.295 7 | 0.106 5 | 0.163 4 | 0.321 0 | 0.131 4 | 0.180 1 | 0.291 3 |
| | ③ | 0.122 5 | 0.172 1 | 0.310 8 | 0.139 1 | 0.188 5 | 0.318 5 | 0.129 4 | 0.185 7 | 0.342 5 | 0.155 5 | 0.203 8 | 0.314 6 |
| KNN 算法 | ① | 0.024 4 | 0.024 4 | 0.024 4 | 0.024 9 | 0.024 9 | 0.024 9 | 0.023 9 | 0.023 9 | 0.023 9 | 0.024 9 | 0.024 9 | 0.024 9 |
| | ② | 0.241 5 | 0.279 5 | 0.400 2 | 0.193 8 | 0.231 2 | 0.318 5 | 0.318 2 | 0.366 0 | 0.477 9 | 0.212 1 | 0.251 3 | 0.336 4 |
| | ③ | 0.265 9 | 0.303 9 | 0.424 6 | 0.218 7 | 0.256 1 | 0.343 4 | 0.342 1 | 0.389 9 | 0.501 8 | 0.237 0 | 0.276 2 | 0.361 3 |
| Kd 树算法 | ① | 0.025 0 | 0.025 0 | 0.025 0 | 0.024 4 | 0.024 4 | 0.024 4 | 0.023 5 | 0.023 5 | 0.023 5 | 0.025 3 | 0.025 3 | 0.025 3 |
| | ② | 0.220 7 | 0.372 1 | 0.743 8 | 0.275 4 | 0.355 1 | 0.728 8 | 0.230 1 | 0.395 9 | 0.816 3 | 0.186 0 | 0.318 5 | 0.657 4 |
| | ③ | 0.245 6 | 0.397 1 | 0.768 8 | 0.229 8 | 0.379 5 | 0.753 2 | 0.233 6 | 0.419 4 | 0.839 8 | 0.211 3 | 0.343 8 | 0.682 7 |

算法的时间差别主要体现在空间存储结构划分之后的搜索过程中,而本文算法耗时最少,在搜索效率方面的优势明显。

## 3 结 语

本文提出了基于立方体小栅格的 $K$ 邻域搜索新算法,采用二次划分的策略将海量点云数据划分到相应的立方体小栅格中,利用任意立方体小栅格与周围立方体小栅格的相邻关系,通过含有采样点的立方体小栅格的扩展,能很快地确定与其相应的最终子空间 $S_{\text{ultimate}}$、内子空间 $S_{\text{in}}$ 和外子空间 $S_{\text{out}}$,再结合采样点的球空间 $S_{\text{sphere}}$,可以快速完成采样点 $K$ 邻域的搜索。实验结果表明,本文算法在时间上具有一定的优势。

## 参 考 文 献


[1] Mitra N J, Nguyen A. Estimation Surface Normals in Noisy Point Cloud Data[C]. The 19th ACM Symposium on Computational Geometry, San Diego, CA, 2003

[2] Jones T R, Durand F, Desbrum M. Non-iterative, Feature-Preserving Mesh Smoothing[C]. The SIGGRAPH '03 Conference, San Diego, CA, 2003

[3] Weyrich T, Pauly M, Heinzle S, et al. Post Processing of Scanned 3d Surface Data[C]. Eurographics Symposium on Point-based Graphics, Switzerland, 2004

[4] Amenta N, Bern M, Kamvysselis M, et al. A New Voronoi-Based Surface Reconstruction Algorithm[C]. The 25th Annual ACM Conference on Computer Graphics, Orlando, 1998

[5] Kolahdouzan M, Shahabi C. Voronoi-Based K Nearest Neighbor Search for Spatial Network Databases[C]. The 30th VLDB Conference, Toronto, Canada, 2004

[6] Die Kerson M T, Drysdale R L S, Sac K J R. Simple Algorithms for Enumerating Inter-point Distances and Finding $K$ Nearest Neighbors[J]. International Journal of Computational Geometry and Applications, 1992, 2(3): 221-239

[7] Sankaranarayanan J, Samet H, Varshney A. A Fast All Nearest Neighbor Algorithm for Applications Involving Large Point-clouds[J]. Computers & Graphics, 2007, 31: 157-174

[8] Bentley J L. Kd Trees for Semidynamic Point Sets[C]. The 6th Annual ACM Symposium on Computational Geometry, San Francisco, 1990

[9] Arya S, Mount D M, Netanyahu N S, et al. An Optimal Algorithm for Approximate Nearest Neighbor Searching[C]. The 5th Annual ACM-SIAM Symposium on Discrete Algorithms, Arlington, VA, 1994

[10] 梁荣华,陈纯,潘志庚,等. 一种面向三维点集的快速表面重构算法[J]. 中国图像图形学报,2003,8A(1): 63-67



**第一作者简介**:赵俭辉,博士,副教授。研究方向为计算机图形图像与人机工程。

E-mail: jianhuizhao@whu.edu.cn


# A New K-Nearest Neighbors Search Algorithm Based on 3D Cell Grids


*ZHAO Jianhui*[1] *LONG Chengjiang*[1] *DING Yihua*[1] *YUAN Zhiyong*[1]

(1 School of Computer, Wuhan University, 129 Luoyu Road, Wuhan 430079, China)



**Abstract:** Based on 3D cell grids, we proposed a new $K$-nearest neighbors search algorithm. The point cloud was divided twice and distributed to 3D cell grids, then ultimate space, internal space, external space were decided for each grid. With the help of each point's sphere space, $K$-nearest neighbors of the point can be found quickly. Compared with the existed methods, the proposed algorithm has more efficient performance.



**Key words:** $K$-nearest neighbors; point cloud; 3D cell grids; search algorithm



**About the first author:** ZHAO Jianhui, Ph.D, associate professor, majors in computer graphics, image processing, and human factors. E-mail: jianhuizhao@whu.edu.cn