中图法分类号: TP391.4 **文献标志码**: A **文章编号**: 1006-8961(2011)09-1676-05
论文索引信息:马娟,朵云峰,赵文亮. 两种空间分块策略 K 近邻搜索算法的比较研究[J]. 中国图象图形学报,2011,16(9):1676-1680

两种空间分块策略 *K* 近邻搜索算法的比较研究

马娟[1),2)],朵云峰[3)],赵文亮[2)]
[1)](西南交通大学土木工程学院测量工程系,成都 610031) [2)](昆明冶金高等专科学校测绘学院,昆明 650033)
[3)](昆明冶金高等专科学校计算机信息学院,昆明 650033)

摘 要:空间分块策略是 *K* 近邻搜索算法研究中的有效方法,然而现有算法进行空间划分时给出的子立方体大小主要取决于 *K* 值的大小,*K* 值变化时需重新进行空间划分,影响了时间效率和稳定性。利用空间分块策略的优点,提出一种以建立离散数据空间索引为空间划分目标的 *K* 近邻搜索新算法。该算法预先对空间包围盒进行微分块,形成的子立方体结构仅与离散数据和预设参数相关,同一点云数据只需进行一次空间分配。搜索过程中,以计算点为球心建立空间动态球,判定符合条件的子立方体,进行 *K* 近邻搜索。测试结果表明,新算法较现有算法点云分配和遍历时间效率、随机点搜索时间稳定性及对不同 *K* 值的适应性等方面更具有优势。

关键词: 空间分块;*K* 近邻;动态球;算法比较

Comparison of two algorithms for finding *K*-nearest neighbors based on spatial sub-cubes

Ma Juan[1),2)], Duo Yunfeng[3)], Zhao Wenliang[2)]
[1)](*Department of Surveying Engineering, Southwest Jiaotong University, Chengdu* 610031 *China*)
[2)](*Department of Surveying, Kunming Metallurgy College, Kunming* 650033 *China*)
[3)](*Department of Computer, Kunming Metallurgy College, Kunming* 650033 *China*)

Abstract: Space block strategy is the effective method in finding *K*-nearest neighbors. However, during dividing the min-max box of the dataset, the size of sub-cubes mainly is decided by the *K* value, and the min-max box is needed to divide again along with the change of *K* value in existing algorithms, it has affected time efficiency and stability of algorithm. Using the advantages of space block strategy, a new algorithm for finding *K*-nearest neighbors is presented with establishing the space index of scattered points as the spatial division goal. The min-max box of the dataset is divided into a set of uniform sub-cubes in advanced, the structure of sub-cubes is only related to the scattered points and the default parameter, the same points cloud data just is distributed one time. In the course of searching, the dynamic sphere is builded using the test point as the center of sphere, and judging eligible sub-cubes in order to find *K*-nearest neighbors. The experimental results show that the new algorithm is more superiority than existing algorithms in points cloud distribution and searching time efficiency, the stability of searching time of random points, and the adaptability of different *K* value.

Keywords: spatial sub-cubes; *K*-nearest neighbors; dynamic sphere; comparison of algorithms

收稿日期:2010-06-07;**修回日期**:2010-11-10
基金项目:云南省应用基础研究面上项目(2009CD102)。

第一作者简介:马娟(1978—),女,讲师。西南交通大学地图制图学与地理信息工程专业博士研究生,主要从事 3 维 GIS 理论及应用研究。E-mail:maxining@126.com。

0　引　言

K 近邻[1]搜索在计算机几何学[2]、模式识别[1]、信息检索[3]、分类[4-6]、离散数据的曲面重建[7-9]等领域运用广泛,实现海量数据的快速搜索是其重要特征。近年来许多学者进行了此方面的研究[2,9-13],主要有 Voronoi 图策略和空间分块策略,其中又以空间分块策略[2,9-11]较优。文献[10-11]提出以 K 值为依据的空间划分算法,给出接近于最佳搜索速度的分块大小,且在搜索终止准则上进行改进,利用方向控制搜索范围的扩展,提高了搜索速度,但其在 K 值变化时均需对空间离散数据在子空间中重新进行分配。在空间分块策略的基础上,提出一种与 K 值无关联的空间微分块与动态球判定相结合的 K 近邻搜索算法,并与现有空间分块策略算法[10-11]进行比较。通过比较两者在空间分块策略上的异同和关键技术,经实验证明,本文算法简单,运行稳定可靠,搜索效率高。

1　空间分块策略比较

现有算法[10-11](以 K 值为依据的空间划分算法,下同)与本文算法(空间微分块与动态球判定算法)均以建立空间包围盒为基础,然后进行空间划分,形成若干空间子立方体,但两者在空间分块的目标和方法上都有所不同。

1)现有算法　空间划分的目标是给出接近最佳搜索速度的分块大小,空间子立方体的大小主要取决于 K 值的大小。K 值变动时,空间子立方体的结构会随之发生变化,这即意味着 K 值变化时必须对空间离散数据进行重新分配。

2)本文算法　空间划分的目标是建立离散数据空间索引,以便快速搜索到 K 近邻候选点,空间子立方体的大小与 K 值无关联。空间子立方体的结构仅与离散数据和预设参数相关,数据源和参数确定后,其结构不随 K 值变化,即 K 值变动时不需重新进行离散数据分配。

2　算法比较

2.1　算法描述

1)现有算法(算法 1)　首先把数据空间分成许多大小相同的立方体子空间;然后将每个离散点归入到相应的子空间内;最后利用子空间内点的信息对每个候选点进行 K 个最近邻域的搜索。

2)本文算法(算法 2)　首先对空间离散数据以原有顺序编号,根据坐标值大小建立空间包围盒;然后采取与 K 值无关联的分块策略,进行离散点分配;再以计算点为球心建立动态球;最后确定候选点并进行 K 个最近邻域点的搜索。

2.2　空间包围盒和子立方体划分

两种算法建立空间包围盒的方法相同:比较离散点集合 x、y、z 坐标大小,建立以 $x_{max}-x_{min}$、$y_{max}-y_{min}$、$z_{max}-z_{min}$ 为边长,以 $(x_{min},y_{min},z_{min})$ 为原点的长方体包围盒,如图 1 所示,即为包含所有数据的最小长方体空间。长方体包围盒体积为

$$V=(x_{max}-x_{min})(y_{max}-y_{min})(z_{max}-z_{min}) \quad (1)$$

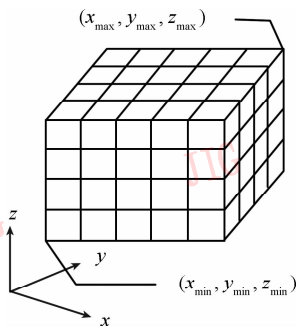

图 1　空间包围盒和空间分块

Fig. 1　The spatial bounding box and spatial sub-cubes

2.2.1　确定子立方体边长

设离散点个数为 N,最小长方体包围盒的体积为 V,将包围盒划分为若干个子立方体。现有算法以式(2)确定边长,K 为邻域数,β 为调节系数;本文算法以式(3)确定边长,n 为子立方体平均点数(n 取2～5)。

$$L=\beta\cdot\sqrt[3]{\frac{K}{N}V}=\beta\cdot\sqrt[3]{\frac{K(x_{max}-x_{min})(y_{max}-y_{min})(z_{max}-z_{min})}{N}} \quad (2)$$

$$L=\sqrt[3]{\frac{n}{N}V}=\sqrt[3]{\frac{n(x_{max}-x_{min})(y_{max}-y_{min})(z_{max}-z_{min})}{N}} \quad (3)$$

2.2.2　确定离散点所在的子立方体

两种算法方法相同。按式(4)计算空间包围盒 x、y、z 方向上的子立方体包围盒的数量,$\lceil\ \rceil$ 表示向上

取整，$cube[i][j][k]$为子立方体的空间位置（i、j、k为子立方体编号），其中 $i=1,\cdots,m_x$；$j=1,\cdots,m_y$；$k=1,\cdots,m_z$；以式(5)计算 i、j、k，判断 P_i 点所在的子立方体，如图 2 所示，分配离散点，并保存子立方体编号和离散点，未分配离散点的子立方体不记录。

$$\begin{cases} m_x = \lceil (x_{\max} - x_{\min})/L \rceil \\ m_y = \lceil (y_{\max} - y_{\min})/L \rceil \\ m_z = \lceil (z_{\max} - z_{\min})/L \rceil \end{cases} \tag{4}$$

$$\begin{cases} i = \lceil (x_i - x_{\min})/L \rceil \\ j = \lceil (y_i - y_{\min})/L \rceil \\ k = \lceil (z_i - z_{\min})/L \rceil \end{cases} \tag{5}$$

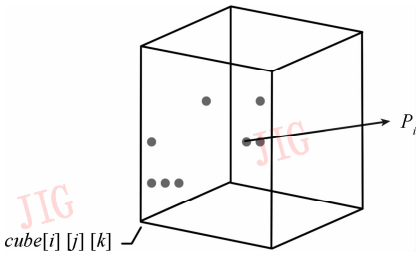

图 2 离散点所在的子立方体

Fig. 2 The sub-cubes in which scattered points locate

2.3 K 近邻搜索

判断 P_i 点所在的子立方体 $cube[i][j][k]$。

1）现有算法 计算 P_i 点到所在立方体包围盒的最小距离 d_s。按式(6)计算 a,b,c,d,e,f 的值，n 为扩展圈数，则 $d_s = min\{a,b,c,d,e,f\}$。该算法的候选点条件为：候选点到 P_i 点的距离 $d \leqslant d_s$。

$$\begin{cases} a = \{[i+(n+1)]L + x_{\min}\} - P_{ix} \\ b = P_{ix} - \{[i-n]L + x_{\min}\} \\ c = \{[j+(n+1)]L + y_{\min}\} - P_{iy} \\ d = P_{iy} - \{[j-n]L + y_{\min}\} \\ e = \{[k+(n+1)]L + z_{\min}\} - P_{iz} \\ f = P_{iz} - \{(k-n)L + z_{\min}\} \end{cases} \tag{6}$$

若候选点的个数大于或等于 K 值，则选择最近的 K 个点为 K 邻域；若候选点的个数小于 K 值，则按图 3 扩展搜索范围。图中填充颜色的子立方体为扩展后的搜索范围。

2）本文算法 以 P_i 点为球心，r 为半径，建立空间动态球。式(7)为 r 的计算公式，其中，β 用来调节球半径 r 的大小。该算法的候选点条件为：候选点到 P_i 点的距离 $d \leqslant r$。

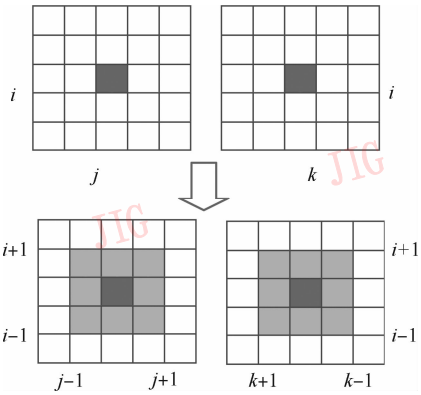

图 3 搜索范围扩展剖面图

Fig. 3 The section plane of searching scope expanding

$$r = \beta \cdot \sqrt[3]{\frac{k}{N}V} = \beta \cdot \sqrt[3]{\frac{k(x_{\max}-x_{\min})(y_{\max}-y_{\min})(z_{\max}-z_{\min})}{N}} \tag{7}$$

设动态球与空间坐标轴分别相交于 $A(x_A,\mathbf{0},\mathbf{0})$、$B(x_B,\mathbf{0},\mathbf{0})$、$C(\mathbf{0},y_C,\mathbf{0})$、$D(\mathbf{0},y_D,\mathbf{0})$、$E(\mathbf{0},\mathbf{0},z_E)$、$F(\mathbf{0},\mathbf{0},z_F)$，按式(8)计算搜索范围：$cube[i_1,\cdots,i_2][j_1,\cdots,j_2][k_1,\cdots,k_2]$，此子立方体集合中的点即为 k 近邻候选点，如图 4 所示。

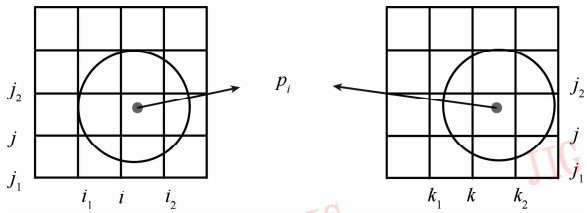

图 4 搜索范围剖面图

Fig. 4 The section plane of searching scope

$$\begin{cases} i_1 = \lceil (x_i - r - x_{\min})/L \rceil \\ i_2 = \lceil (x_i + r - x_{\min})/L \rceil \\ j_1 = \lceil (y_i - r - y_{\min})/L \rceil \\ j_2 = \lceil (y_i + r - y_{\min})/L \rceil \\ k_1 = \lceil (z_i - r - z_{\min})/L \rceil \\ k_2 = \lceil (z_i + r - z_{\min})/L \rceil \end{cases} \tag{8}$$

若候选点的个数大于或等于 K 值，则选择最近的 K 个点为 K 近邻；否则，则扩大 r 值，重新判断搜索范围。

3 实验分析

两种算法均使用 Java 编程实现，在 Intel Core (TM)2 1.5G CPU 的 PC 机上进行算法实验分析。

实验采用真实物体的 3 维激光扫描点云数据,实验数据兼顾了点云数量、物体形状、均匀度,具有一定的普遍性,图 5 为头像点云及某计算点的 K 邻域。

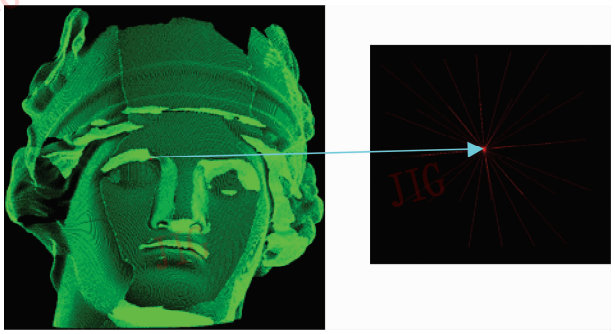

图 5　头像点云及某计算点的 K 邻域

Fig. 5　Points cloud for the head statue and K-NN of some point

实验 1　比较两种算法的点云分配和遍历搜索时间。实验采用头像、猫、丝带点云数据,搜索时间为遍历所有点的 K 近邻平均 CPU 运行时间,结果如表 1 所示(—表示不需重新进行分配)。算法 1 对相同点云在 K 值变化时,需要重新进行点云分配,算法 2 对相同点云仅需进行 1 次分配,减少了分配时间。由实验结果可以看出,由于空间分块策略的改进和采用动态球判定搜索范围,算法 2 搜索速度得到较大的提高。

实验 2　比较两种算法的随机点 K 近邻搜索时间稳定性。在上述 3 个点云中随机选取一组点进行单点测试,分别以点云中的 60、123、1 580、2 500 号点为例,对比两种算法搜索不同点 K 近邻的时间变化。K 值取 30,结果如表 2 所示。测试结果表明,算法 2 的随机点 K 近邻搜索时间变化较小,具有较好的时间稳定性。

实验 3　比较两种算法对不同 K 值的适应性。选取头像点云(294 644 点)作为实验数据,分别取 K = 10、30、50、80、100、150,测试两种算法遍历所有点的搜索时间和两点间距离计算次数,结果如表 3 所示,

表 1　两种算法的 K 近邻搜索时间比较

Tab. 1　Comparison of two algorithms for K-NN searching time

/ms

算法	头像(294 644 点)						猫(10 000 点)						丝带(5 000 点)					
	10		30		50		10		30		50		10		30		50	
	分配	搜索	分配	搜索	分配	搜索	分配	搜索	分配	搜索	分配	搜索	分配	搜索	分配	搜索	分配	搜索
1	778.5	0.382	734.5	0.585	755.4	1.219	32.90	0.376	33.94	0.557	34.70	0.829	12.24	0.349	12.97	0.643	12.34	1.022
2	624.3	0.216	—	0.3866	—	0.569	24.85	0.142	—	0.280	—	0.462	9.3550	0.175	—	0.357	—	0.567

表 2　随机点的 K 近邻搜索时间

Tab. 2　K-NN searching time of random points

/ms

算法	头像(294 644 点)				猫(10 000 点)				丝带(5 000 点)			
	60	123	1 580	2 500	60	123	1 580	2 500	60	123	1 580	2 500
1	0.702 6	0.618 4	0.878 7	1.297 0	0.817 1	1.242 6	1.536 2	1.562 9	1.360 1	1.679 9	2.480 1	4.488 5
2	0.447 5	0.438 3	0.412 6	0.394 5	0.438 3	0.499 4	0.416 2	0.456 8	0.949 0	0.817 2	1.294 9	1.022 6

表 3　同一物体不同 K 值的搜索时间

Tab. 3　K-NN searching time in different K Values for the same object

算法		10		30		50		80		100		150	
		时间/ms	次数	时间/ms	次数	时间/ms	次数	时间/ms	次数	时间/ms	次数	时间/ms	次数
1	合计	114 086	24 367 059	172 248	48 704 653	359 200	82 087 818	640 173	144 375 560	771 908	229 056 245	931 841	396 590 824
	平均	0.387	82.7	0.584	165.3	1.219	278.6	2.172	490.0	2.619	777.4	3.1626	1346.0
2	合计	639 08	57 160 936	113 909	81 763 710	167 799	100 002 173	282 799	125 871 916	377 439	142 519 302	590 466	260 612 618
	平均	0.216 9	194.0	0.386	277.5	0.569 5	339.4	0.959	427.2	1.281	483.7	2.004	884.5

并在此基础上建立两种算法相同点云不同 K 值对两点间距离计算次数及搜索时间的影响示意图，如图 6所示。从表 3 和图 6 可以看出，两种算法 K 值适应性都较好，但算法 2 较算法 1 在搜索时间上更具效率，且两点间计算次数、搜索时间随 K 值变化更呈线性趋势，K 值适应性较强。

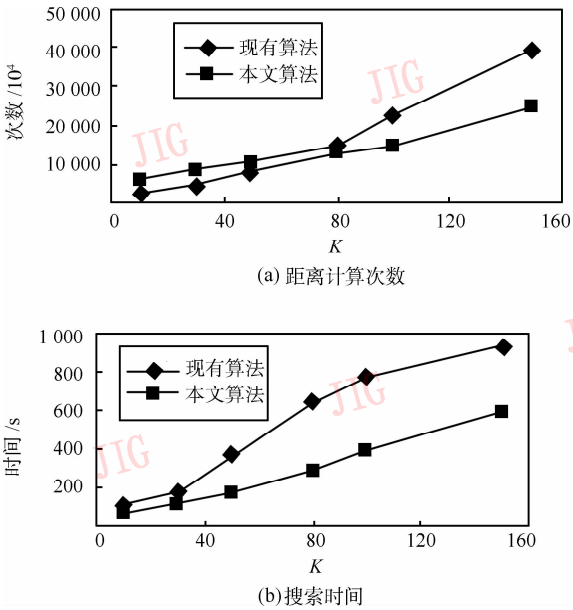

图 6 两种算法的 k 值变化影响分析

Fig. 6 Analysis on distance calculations and search time in different K values of two algorithms

4 结 论

以空间分块策略为基础，提出一种 K 值无关联的空间微分块与动态球判定相结合的 K 近邻搜索算法，并与现有以 K 值为依据的空间划分算法进行了比较。实验结果证明，提出的算法更具优势，主要有：1）采用与 K 值无关联的空间微分块方法，较好地解决了现有算法中 K 值发生变化时引起的离散点重新分配问题；2）K 近邻搜索时间随 K 值变化更具线性，适应性较强，且具有更高的搜索效率；3）相同点云数据不同离散点的 K 近邻搜索时间变化较小，具有较好的稳定性。

参考文献(References)

[1] Kudo M, Masuyama N, Toyama J, et al. Simple termination conditions for k-nearest neighbor method[J]. Pattern Recognition Letters, 2003, 24: 1203-1213.

[2] Piegl L A, Tiller W. Algorithm for finding all k nearest neighbors [J]. Computer-aided Design, 2002, 34(2): 167-172.

[3] Xiao Hui, Yang Bisheng. An improved KNN search algorithm based on road networ k distance[J]. Geomatics and Information Science of Wuhan University, 2008, 33(4): 437-439. [肖晖, 杨必胜. 一种改进的基于道路网络距离的 K 近邻查询算法[J]. 武汉大学学报: 信息科学版, 2008, 33(4): 437-439.]

[4] Hastie T, Tibshirani R. Discriminant adaptive nearest neighbor classification[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1996, 18(6): 607-616.

[5] Bian Fuling, Wan You. A novel spatial co-location pattern mining algorithm based on K-nearest feature relationship[J]. Geomatics and Information Science of Wuhan University, 2009, 34(3): 331. [边馥苓, 万幼. K-邻近空间关系下的空间同位模式挖掘算法[J]. 武汉大学学报: 信息科学版, 2009, 34(3): 331.]

[6] Tomasi C, Manduchi R. Stereo matching as a nearest-neighbor problem[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1998, 20(3): 333-340.

[7] Kobbelt L P, Botsch M, Schwanecke U, et al. Feature sensitive surface extraction from volume data[C]//Computer Graphics Proceedings, Annual Conference Series, ACM, SIGGRAPH. Los Angeles, California: ACM, 2001: 57-66.

[8] Wang Qing, Wang Rongqing, Bao Hujun, et al. A fast progressive surface reconstruction algorithm for unorganized points[J]. Journal of Software, 2000, 11(9): 1221-1227. [王青, 王融清, 鲍虎军, 等. 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.]

[9] Zhou Rurong, Zhang Liyan, Su Xu, et al. Algorithmic research on surface reconstruction from dense scattered points[J]. Journal of Software, 2001, 12(2): 249-255. [周儒荣, 张丽艳, 苏旭, 等. 海量散乱点的曲面重建算法研究[J]. 软件学报, 2001, 12(2): 249-255.]

[10] Xiong Bangshu, He Mingyi, Yu Huajing. Algorithm for finding k-nearest neighbors of scattered points in three dimensions[J]. Journal of Computer-aided Design&Computer Graphics, 2004, 16(7): 909-912. [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.]

[11] Zhang Tao, Zhang Dinghua, Wang Kai, et al. A novel strategy for fast searclling of k-nearest neighbors[J]. Mechanical Science and Technology for Aerospace Engineering, 2008, 27(10): 1233-1235. [张涛, 张定华, 王凯, 等. 空间散乱点 k 近邻搜索的新策略[J]. 机械科学与技术, 2008, 27(10): 1233-1235.]

[12] 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.

[13] Dickerson M T, Drysdale R L S, Sack J R. Simple algorithms for enumerating interpoint distances and finding k nearest neighbors [J]. International Journal of Computational Geometry and Applications, 1992, 2(3): 221-239.