文章编号：2096-1618(2023)02-0148-06

k 近邻空间插值算法优化研究

陈超英， 陈宫燕， 李彦军， 穷 达， 索朗卓嘎
（西藏林芝气象局，西藏 林芝 860000）

摘要：在 k 近邻空间插值中，如果能减少近邻点的搜索次数，可进一步提高空间插值的性能。引入 k 近邻距离阈值的概念和计算方法，并以该阈值为基础，发展了 k+M 优化算法。其算法核心是在空间插值过程中，获取初始栅格的 k+M 近邻点集，计算 k+M 近邻距离阈值。若从初始栅格向右移动至其他栅格的距离小于该阈值，则直接利用初始栅格的近邻点集进行空间插值。实验证明，该算法相对于每个栅格均搜索近邻点的算法，性能有明显的提升。

关 键 词：k 近邻点集；空间插值；优化算法
中图分类号：TP301.6 **文献标志码**：A
doi：10.16836/j.cnki.jcuit.2023.02.004

0 引言

空间插值用于将离散点的测量数据转换为连续的数据曲面，广泛应用于分析区域降水量、环境污染态势、资源利用程度、公共基础设施影响效应等领域。k 近邻搜索作为 IDW、克里金插值等的基础算法，其性能直接影响插值速度。构建采样点的空间索引是 k 近邻搜索算法优化的基础，目前主要的空间索引方法有网格索引[1]、四叉树[2]、R 树[3-4]、R$^+$ 树[5]、R* 树[6]、Voronoi 图[7-9]等。在空间索引的基础上，为进一步加快近邻点搜索速度，前人采用了各种优化算法，如在局部搜索过程中，引入方向控制[10]，或者采用近邻搜索剪枝策略[11-14]减少邻近点搜索范围，甚至引入 GPU 提升 k 近邻插值的性能[15-16]。

现有的 k 近邻优化算法主要针对单个位置的快速搜索。在空间插值中，每一个栅格中心均需要搜索 k 个近邻点。如果能采用某种算法，快速判断空间上相邻的栅格是否存在相同的 k 近邻点集，对于提升空间插值的性能具有重要意义。本文提出了 k 近邻距离阈值的概念，并以此为基础发展了 k+M 近邻空间插值的优化算法，并通过实验证明了算法的有效性[1]。

1 相关定义

首先给出空间插值中 k 近邻点集、k 近邻空间插值、k 近邻圆、相同 k 近邻点集、k 近邻距离阈值的定义。

定义 1 设在二维平面上存在一个采样点集合 $\{S_1, S_2, \cdots, S_N\}$，其采样点的数量为 N。每一个采样点为 $S_i(1<i<N)$。待插值栅格像元为 C。在二维平面上找出直线距离距 C 的中心最近的 k 个点，并按距离由近及远排序，构建一个有序的采样点子集 $\{S_1^C, S_2^C, \cdots, S_k^C\}$，称为栅格像元 C 的 k 近邻点集。

k 近邻点集的距离按顺序表示为一个集合，即 $\{d_1^C, d_2^C, \cdots, d_k^C\}$。

定义 2 依据每一个栅格像元的 k 近邻点集进行插值估算，称为 k 近邻空间插值。

依据定义，IDW、克里金插值时，若以固定点数进行插值估算，则此时为 k 近邻空间插值。

定义 3 以栅格像元 C 的中心为圆心，以 d_k^C 为半径的圆，称为栅格像元 C 的 k 近邻圆。

根据定义，k 近邻点集中的所有样点均在 k 近邻圆内或圆周上。第 k 个样点必然在近邻圆的圆周上，如图 1 所示。

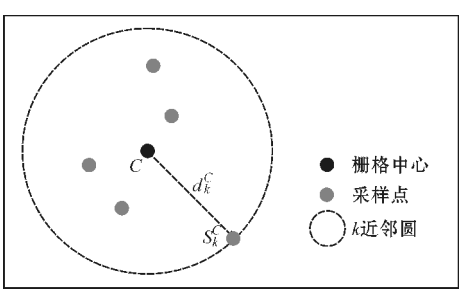


图 1 待插值栅格中心的 k 近邻点及 k 近邻圆

定义 4 两个 k 近邻点集中，任一个点集的每一个采样点，均能在另一个点集中找到，称这两个 k 近邻点集为相同 k 近邻点集。

定义 5 存在一个阈值 ε，以栅格像元 C 的中心为圆心，半径小于 ε 的圆内所有位置上，其 k 近邻点集与 C 的 k 近邻点集属于相同 k 近邻点集，该阈值称为 k 近邻距离阈值。

收稿日期：2022-04-15
基金项目：西藏自治区自然科学基金资助项目（XZ202101ZR0042G）
通信作者：陈宫燕. E-mail：chengongyan_35@163.com

从定义可见,在空间插值时,若上一个栅格像元已找到 k 近邻点集,在下一个栅格像元的空间估算时,若栅格中心的偏移距离小于 ε,则直接利用上一个栅格像元的 k 近邻点集进行插值计算,从而减少栅格像元的 k 近邻点搜索次数,提升空间插值的性能。

2　算法描述

2.1　k 近邻距离阈值的确定

为确定插值栅格的 k 近邻距离阈值,需要研究同一个采样点,随栅格中心的移动,其距离的变化规律。在空间插值中,一般按照从左到右、从上到下的顺序遍历每一栅格单元。在遍历过程中,频率最高的是从左向右单个栅格单元地移动。

设空间插值时栅格中心从 C 向右移至 C_1,其移动距离为 c。已知 C 的第 k 个近邻点 S_k^C 与 C 的距离为 d_k^C,且 $d_k^C>c$,则 S_k^C 的可能位置在 C 的 k 近邻圆圆周上。当向右移动至 C_1 时,S_k^C 与 C_1 的距离为 $d_k^{C_1}$,如图 2所示。

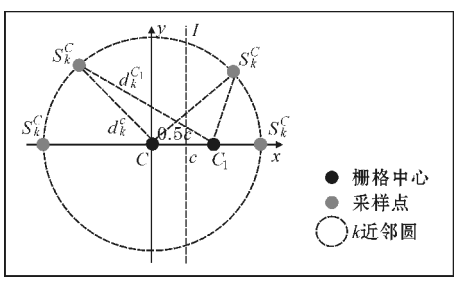


图 2　栅格中心移动时近邻点的距离变化示意

从图 2 可见,在 C 和 C_1 的中点处引一条平行于 y 轴的直线 l。当 S_k^C 位于 l 与近邻圆圆周的交点时,有 $d_k^C=d_k^{C_1}$;当 S_k^C 位于 l 的左侧时,$d_k^C<d_k^{C_1}$;当 S_k^C 位于 l 的右侧时,$d_k^C>d_k^{C_1}$。依据三角形两边之差小于第三边,可以得到如下结论:

(1)当 S_k^C 位于 l 的左侧时,栅格中心向右移动,近邻点 S_k^C 与栅格中心的距离增加,且有 $d_k^{C_1}-d_k^C\leqslant c$。仅当 S_k^C 与 C 和 C_1 在同一条直线上时,有 $d_k^{C_1}-d_k^C=c$。即当栅格中心向右移动,近邻点 S_k^C 与栅格中心的距离增加最大的位置在$\overrightarrow{C_1C}$方向上。

(2)当 S_k^C 位于 l 的右侧时,栅格中心向右移动,近邻点 S_k^C 与栅格中心的距离减小,且有 $d_k^C-d_k^{C_1}\leqslant c$。仅当 S_k^C 与 C 和 C_1 在同一条直线上时,有 $d_k^C-d_k^{C_1}=c$。即当栅格中心向右移动,近邻点 S_k^C 与栅格中心的距离减小最大的位置在$\overrightarrow{CC_1}$方向上。

根据以上两个结论,可以很容易求取栅格 C 的 k 近邻距离阈值 ε。现引入 $k+1$ 近邻点,表示为 S_{k+1}^C。该点为除 C 的 k 近邻点集外,距离 C 最近的采样点。依据 ε 的定义,在 k 近邻距离阈值范围内的任意位置 C_t,必然满足 S_{k+1}^C 与 C_t 的距离大于等于 S_k^C 与 C_t 的距离。由于栅格中心向右移动,距离增加和减小的最大位置分别在直线 CC_1 的两侧。设 S_k^C 和 S_{k+1}^C 分别位于直线 CC_1 上的左侧(点 A)和右侧(点 B),距离 C 的距离分别为 d_k^C 和 d_{k+1}^C。直线 CC_1 上,与点 A 距 C 相等的右侧点为 D,D 与 B 的中点为 E,则当 $c<DE$ 时,有 $C_1A<C_1B$;$c=DE$ 时,$C_1A=C_1B$;$c>DE$ 时,$C_1A>C_1B$。因此可得:

$$\varepsilon=\frac{1}{2}(d_{k+1}^C-d_k^C)\tag{1}$$

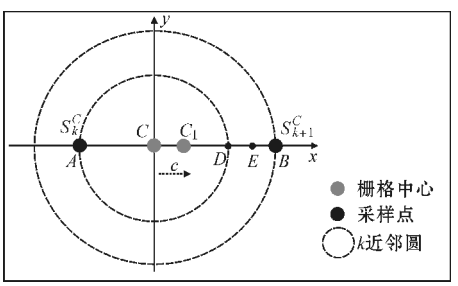


图 3　k 近邻距离阈值的确定

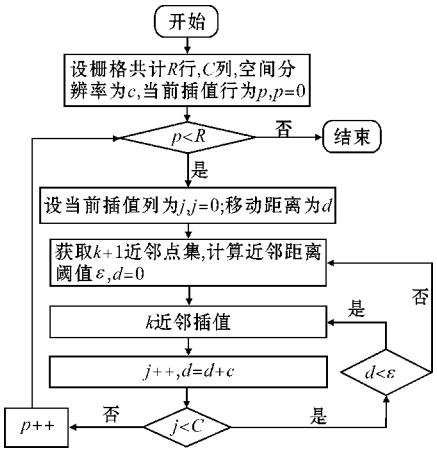
依据式(1),可得到如下定理:

定理 1　在二维平面上,以点 C 为中心,点 C 的第 $k+1$ 个近邻点与第 k 个近邻点的距离之差的一半为半径的圆内所有位置上,其 k 近邻点集与点 C 的 k 近邻点集属于相同 k 近邻点集。

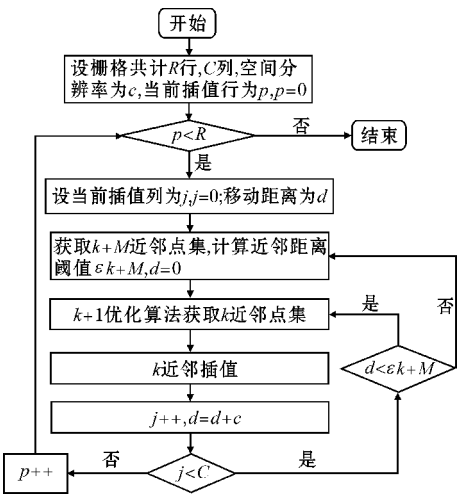
2.2　k 近邻空间插值算法的初步优化

依据定理 1,k 近邻空间插值算法的初步优化思路为:在 k 近邻空间插值时,对于某栅格 C,首先获取$k+1$ 近邻点集,计算 k 近邻距离阈值。以 C 为起点,向右遍历后续栅格时,若移动距离小于 ε,则直接利用 C 的近邻点集进行插值估算,否则重新搜索新栅格位置上的 $k+1$ 近邻点集,优化算法流程如图 4(a)所示。为方便后续阐述,该优化算法称为 $k+1$ 优化算法。

从算法复杂度上分析,搜索近邻点集时,比算法优化前多增加一个点的搜索,其算法复杂度增加,其增加量与采样点分布、近邻点搜索方法相关;k 近邻距离阈值省略了部分栅格的近邻点搜索,算法复杂度降低,其降低量与采样点分布、待插值栅格空间分辨率密切相关。假定采样点在插值区域中均匀分布,其点密度为 ρ,待插值栅格空间分辨率为 c。空间插值以行为单元,每一行均需重新搜索近邻点集,则近邻点搜索次数与未优化前搜索次数的比例为



(a) k+1 优化算法



(b) k+M 优化算法

图4 k+1 和 k+M 优化算法流程图

$$r=\frac{c\sqrt{\rho\pi}}{0.5(\sqrt{k+1}-\sqrt{k})} \tag{2}$$

虽然式(2)仅给出了理想状态下搜索次数的优化情况,但仍可从中得到以下规律:同一个采样点集的 k 近邻空间插值,待插值栅格空间分辨率越高,近邻点搜索次数减少的比例越大;在相同的栅格空间分辨率条件下,采样点分布密度越大,近邻点搜索次数减少的比例越小。

2.3 基于 k+M 的近邻空间插值优化算法

通过 k 近邻距离阈值,减少重新搜索近邻点的次数为算法优化的核心。如果将上述优化算法的 k+1 推广至 k+M,可进一步对算法进行优化。依据定理 1,可推论出定理 2。

定理 2 在二维平面上,以点 C 为中心,点 C 的第 k+M($M\geqslant 1$)个近邻点与第 k 个近邻点的距离之差的一半为半径的圆内的所有位置上,其 k 近邻点集为点 C 的 k+M 近邻点集的子集。

依据定理 2,k+M 优化算法为:对于某栅格 C,首先获取 k+M 近邻点集,计算 k+M 近邻距离阈值 ε_{k+M}:

$$\varepsilon_{k+M}=\frac{1}{2}(d_{k+M}^{C}-d_{k}^{C}) \tag{3}$$

向右遍历栅格时,若相对于 C 的移动距离小于 ε_{k+M},则直接利用 C 的 k+M 近邻点集计算 k 近邻点集。否则,重新搜索新栅格位置上的 k+M 近邻点集。在由 k+M近邻点集计算 k 近邻点集时,采用前述 k+1 优化算法进行局部搜索优化。优化算法流程如图 4(b)所示。

当优化算法从 k+1 扩展至 k+M 时,采样点在插值区域中均匀分布,近邻点搜索次数与未优化前搜索次数的比例为

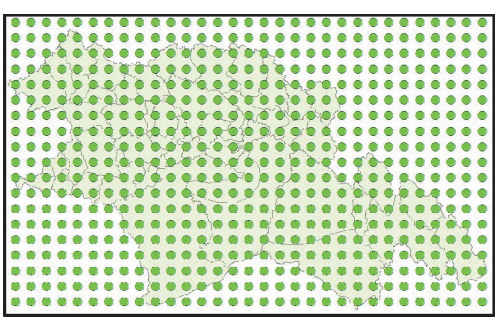
$$r=\frac{c\sqrt{\rho\pi}}{0.5(\sqrt{k+M}-\sqrt{k})} \tag{4}$$

根据式(4),在其他条件相同的情况下,M 越大,近邻点搜索次数越少。但这并不意味着 M 越大越好。随着 M 的增加,单个栅格搜索近邻点集的算法复杂度不断增加,需要通过实验确定适当的 M 值。

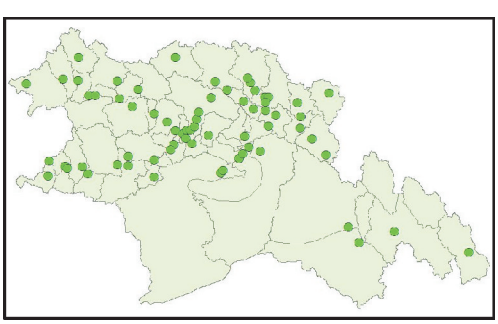
3 实验与结果

3.1 M 和 k 值分析

采用两个采样点集进行空间插值参数分析。第一个采样点集共计 589 个点,代表均匀分布模式,命名为 A,如图 5(a)所示。第二个为林芝地区气象站样点



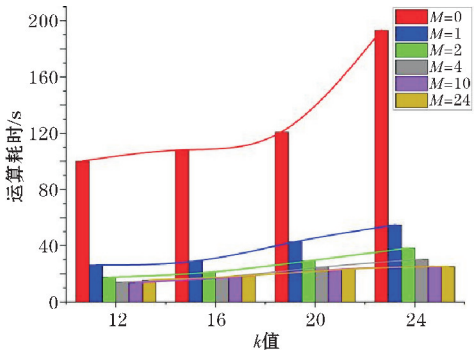
(a)采样点集 A



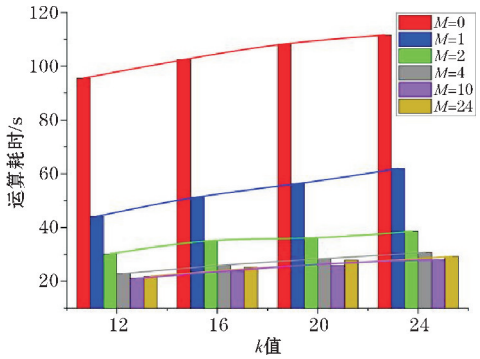
(b)采样点集 B

图5 两种不同分布模式的采样点集

集，共计 68 个点，代表非均匀分布模式，命名为 B，如图 5（b）所示。实验计算机的 CPU 为 Inter（R） Core（TM） i7-7700HQ。实验时，均采用网格索引法建立采样点集的空间索引，空间插值算法为 IDW，插值分辨率为50 m，生成的栅格大小为 12302×7211。在不同 k 值条件下，统计两个采样点集 $M=0$、$M=1$ 和 $M>1$ 时插值所需时间。其中，$M=0$ 代表未优化算法，每一个栅格均需搜索 k 近邻点集；$M=1$ 和 $M>1$ 分别为 $k+1$ 优化算法和 $k+M$ 优化算法。实验结果如图 6 所示。



（a）采样点集 A 插值时间



（b）采样点集 B 插值时间

图 6　两种不同分布模式的空间插值耗时统计

从图 6 可见，无论哪一种分布模式，$k+1$ 和 $k+M$ 优化算法均有明显的性能提升，$k+M$ 优化算法优于$k+1$优化算法。具体而言，A 和 B 的 $k+1$ 优化算法平均所需时间分别为未优化算法的29.2% 和45.7%；随着 M 的增加，空间插值所需时间先减少后增加，在$M=0.5$ k附近时，平均耗时最少。特别地，当 $M=6$ 时，A 和 B 的空间插值平均时间分别为未优化算法的15.8% 和17.8%。

3.2　$k+M$ 算法与 ArcGIS 空间插值性能对比分析

为进一步验证本文算法对 k 近邻空间插值的优化程度，利用同样的数据集，在同样分辨率和插值范围内将 $k+6$ 优化算法与 ArcGIS 空间插值进行对比，并用$k+6$优化算法的耗时除以 ArcGIS 耗时得到耗时比。统计结果如表 1 所示。

表 1　$k+6$ 优化算法与 ArcGIS 空间插值所需时间的对比

k	A			B		
	$k+6$	ArcGIS	耗时比	$k+6$	ArcGIS	耗时比
12	13.71	36.07	0.38	12.27	26.37	0.47
16	17.29	45.02	0.38	15.57	30.01	0.52
20	23.06	51.87	0.44	18.07	33.99	0.53
24	27.64	59.98	0.46	20.46	37.71	0.54

从表 1 可见，在数据集 B 上，本文的优化算法耗时仅为 ArcGIS 的 50% 左右；随着样点分布密度的增加，算法优化越明显。特别地，在数据集 A 上，本文的优化算法耗时在 ArcGIS 的 50% 以下。

由于已有算法主要从单个像元的近邻搜索进行优化，而本文则是从临近像元的近邻搜索剪枝策略上进行优化。从某种意义上讲，本文算法与 ArcGIS 空间插值的优化思路存在差异，其性能对比统计仅能说明算法的有效性。

3.3　不同空间分辨下的空间插值优化性能评价

以采样点集 B 为数据源，设 $k=12$，采用不同空间分辨率进行固定点数的 IDW 空间插值。统计各算法的空间插值耗时，并以算法实际耗时/未优化算法耗时计算耗时百分比，结果如表 2 所示。

表 2　不同空间分辨率下空间插值所需时间

空间分辨率/m	耗时/s			百分比/%		
	$M=0$	$M=1$	$M=6$	$M=0$	$M=1$	$M=6$
50	93.29	37.21	12.27	100.00	39.89	13.15
75	43.41	12.07	6.06	100.00	27.80	13.96
100	24.56	7.84	3.66	100.00	31.92	14.90
150	11.14	4.12	1.87	100.00	36.98	16.79

从表 2 可见,对于同一个采样点集的空间插值,随着空间分辨率的提高,无论是 k+1 优化算法还是 k+M 优化算法,其性能均不断提升。这说明,空间分辨率的提高,近邻点搜索次数减少的比例越小,算法优化程度更高。

4 讨论

为进一步提高 k 近邻空间插值的性能,利用 k 近邻距离阈值减少栅格点搜索近邻点的次数,提升空间插值的性能。虽然本文算法在二维平面上以水平向右移动为例给出了 k 近邻距离阈值的计算公式,并推导了定理 1。一般情况下,空间上的点向任意方向移动,当近邻点沿着运动方向与该点在同一条直线上时,其距离增加或减小的量达到最大值。因此,定理 1 和定理 2 可直接推广到三维空间中,以解决三维空间中连续位置的 k 近邻点搜索的优化问题。

从与 ArcGIS 插值性能对比可以发现,本文算法有效地对 k 近邻插值进行了优化。可以将本文算法作为网格索引、四叉树、R 树等传统单像元近邻搜索算法的后端,对空间插值进一步优化。

另外,本文仅以均分分布和非均匀分布两种分布模式对 k 近邻空间插值的参数进行分析,发现两种模式下随着 M 的增加,空间插值所需时间均呈现先减少后增加的趋势,其最优参数在0.5 k附近。但实际的采样点集的分布模式更为复杂,如聚集分布、分散分布,甚至不同的局部区域存在不同的空间分布模式。因此,参数分析的结果在不同的分布模式下可能存在一定的偏差。

5 结束语

首先给出 k 近邻距离阈值的计算公式,并得到 k+1优化算法。以此为基础,阐述了 k+M 优化算法的流程。通过对照实验,发现 k+1 和 k+M 优化算法均有明显的性能提升。当 M 取值为0.5 k附近时,空间插值耗时为未优化算法的40% 以下,达到了算法优化的目标,其研究成果具有一定的理论意义和实践价值。

参考文献:

[1] 王映辉. 一种 GIS 自适应层次网格空间索引算法[J]. 计算机工程与应用,2003(9):58-60.

[2] 彭召军,王青山,熊伟,等. 基于改进四叉树的地理实体快速查询算法[J]. 地理空间信息,2017,(1):32-35.

[3] Yang Y, Bai P, Ge N, et al. LAZY R-tree: The R-tree with lazy splitting algorithm[J]. Journal of Information Science, 2019, 46(1).

[4] Jin P, Xie X, Wang N, et al. Optimizing R-tree for flash memory [J]. Expert Systems with Applications, 2015, 42(10): 4676-4686.

[5] MŠumák, P Gurskí. R^{++}-Tree: An Efficient Spatial Access Method for Highly Redundant Point Data [J]. Advances in Intelligent Systems & Computing, 2014, 241: 37-44.

[6] Qin-Yang W U. Improved R^*-tree spatial index: Improved R^*-tree spatial index [J]. Journal of Computer Applications, 2010, 30(2): 419-422.

[7] Stanoi I, Riedewald M, Agrawal D, et al. Discovery of Influence Sets in Frequently Updated Databases [J]. Vldb, 2001: 99-108.

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

[9] Dickerson M T, Drysdale R S, Sack J. SIMPLE ALGORITHMS FOR ENUMERATING INTERPOINT DISTANCES AND FINDING k NEAREST NEIGHBORS[J]. International Journal of Computational Geometry & Applications, 1992, 2(3): 9200014.

[10] 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(10):1233-1235.

[11] 刘宇,朱仲英,施颂椒. 空间 k 近邻查询的新策略[J]. 上海交通大学学报,2001(9):1298-1302.

[12] Abu-Aisheh Z, Raveaux R, Ramel J Y. Fast Nearest Neighbors Search in Graph Space Based on a Branch-and-Bound Strategy[J]. Springer, Cham, 2017.

[13] 廖巍,熊伟,王钧,等. 可伸缩的增量连续 k 近邻查询处理[J]. 软件学报. 2007(2):268-278.

[14] Cheema M A, Lin X, Zhang W, et al. Influence Zone: Efficiently Processing Reverse k Nearest Neighbors Queries [C]. Data Engineering (ICDE), 2011 IEEE 27th International Conference on. IEEE, 2011.

[15] Wang H,Zhao W L,Zeng X. Large-Scale Approximate k-NN Graph Construction on GPU[J]. 2021.

[16] 田盼,华蓓,陆李. 基于 GPU 的 K-近邻算法实现[J]. 计算机工程,2015(2):189-192.

An Algorithmic Optimization Study of k-nearest Neighbor Space Interpolation

CHEN Chaoyin, CHEN Gongyan, LI Yanjun, QIONG Da, SUO LANG Zhuoga

(Linzhi City Meteorological Bureau, Linzhi 860000)

Abstract: For the k-nearest-neighbor spatial interpolation algorithm, the performance of spatial interpolation can be further improved if the number of searches for nearest-neighbor points can be reduced. In this paper, the concept and calculation method of the k-nearest neighbor distance threshold is proposed first, and based on it the $k+M$ optimization algorithm is developed. The key part of the $k+M$ optimization algorithm is obtaining the $k+M$ nearest neighbor points of the initial grid in the process of spatial interpolation, and calculating the $k+M$ nearest neighbor distance threshold. If the moving distance of the initial grid is less than the set threshold, the spatial interpolation can be performed directly using the nearest neighbor points of the initial grid. Experimentally, compared with the algorithm of searching the nearest neighbor points for each raster, the performance of the algorithm used in this paper is significantly improved and its research results have certain theoretical significance and practical value.

Keywords: k-nearest neighbor points; spatial interpolation; optimization algorithm