第 20 卷第 3 期
2006 年 9 月

山西师范大学学报(自然科学版)
Journal of Shanxi Normal University
Natural Science Edition

Vol.20　No.3
Sept.　2006

文章编号:1009-4490(2006)03-0019-04

分治法实现最接近点对问题的三维推广算法

张晓红[1,2], 胡金初[2]

(1. 山西师范大学 数学与计算机科学学院,山西 临汾 041004;
2. 上海师范大学 数理信息学院,上海 200234)

摘　要:最接近点对问题是空中交通控制系统应用中的一个重点问题,也是计算机几何学研究的基本问题之一.利用分治法已经解决该问题的一维和二维情况,且算法都可以在 $O(n*\log n)$ 时间内完成.本文在原有一维和二维算法基础上,提出了利用分治法实现该问题的三维情况的算法,并对算法的效率进行了分析.

关键词:最接近点对;分治法;三维;效率

中图分类号:TP301.6　　文献标识码:A

0　引言

在计算机应用中,常用点、圆等简单的几何对象表达现实世界中的实体.在涉及这些几何对象的问题中,常需要了解其领域中其他几何对象的信息.最接近点对问题常用于空中交通的计算机自动控制系统中.也是计算机几何学研究的基本问题之一.

该问题可以描述为:任意给定 n 个点,找出其中的一对点,使得在 n 个点组成的所有点对中,该点对间的距离最小.严格地讲,n 个点中的最接近点对可能多于 1 对,为了简化问题,我们只找出其中的 1 对作为问题的解.

利用分治法解决此问题的基本思想是:将 n 个点构成的集合 S 分成 2 个子集 S_1 和 S_2($S_1 \cup S_2 = S$),每个集合中约有 $n/2$ 个点.在每个子集中递归的求其最接近的点对,最后合并得出 S 中的最接近点对.

1　问题的一维和二维情形的算法[1-3]

1.1　一维情形的算法

一维情况下,S 中的 n 个点退化为数轴上的 n 个实数点 $x_1, x_2, \ldots, x_n$.最接近的点对即为这 n 个实数点中距离相差最小的 2 个实数点.取 n 个点坐标的中位数 m 将 S 划分为 2 个集合 S_1 和 S_2.递归地在 S_1 和 S_2 上分别找出其最接近的点对 $\{p_1, p_2\}$ 和 $\{q_1, q_2\}$,并设 $d = \min\{|p_1 - p_2|, |q_1 - q_2|\}$.$S$ 中的最接近点对或者是 $\{p_1, p_2\}$,或者是 $\{q_1, q_2\}$,或者是某个 $\{p_3, q_3\}$,其中一定有 $p_3 \in (m-d, m]$,$q_3 \in (m, m+d]$.如果 $(m-d, m]$ 中有 S 中的点,则此点就是 S_1 中的最大点.同理,如果 $(m, m+d]$ 中有 S 中的点,则此点就是 S_2 中的最小点.从而,我们只需要找出 S_1 中的最大点和 S_2 中的最小点,并计算其距离后与 d 作比较,即可解得 S 中的最接近点对.亦即我们可以用线性时间将 S_1 的解和 S_2 的解合并成为 S 的解.由此,该算法的分割步骤(中位数的求解耗时 $O(n)$)和合并步骤总共耗时 $O(n)$.求解算法耗费的计算时间 $T(n)$ 满足的递推方程:

$$\begin{cases} O(1) & n < 4 \\ T(n) = 2T(n/2) + O(n) & n \geqslant 4 \end{cases}$$

可解得 $T(n) = O(n*\log n)$.

1.2　二维情形的算法

收稿日期:2006-01-04
作者简介:张晓红(1980—),女,山西柳林人,山西师范大学数学与计算机科学学院教师,上海师范大学在读硕士研究生,主要从事网络与多媒体方面的研究.

二维情况中,S 中的每个点都有 x 和 y 两个坐标. 选取一垂直线 L :$x = m$(m 为 S 中各点 x 坐标的中位数) 作为分割直线,将 S 分割为 S_1 和 S_2 两个半平面区域中点的子集,递归地在 S_1 和 S_2 上分别找出其最小距离 d_1 和 d_2,并设 $d = \min\{d_1, d_2\}$. 用 P_1 和 P_2 分别表示在直线 L 左边和右边与其距离在 d 范围的点构成的两个垂直长条平面区域,P_1 :$\{p \in P_1 \mid |m - x(p)| \leqslant d\}$;P_2 :$\{p \in P_2 \mid |x(p) - m| \leqslant d\}$. S 中的最接近点对的距离或者是 d,或者是某个 $\{p, q\}$ 点对的距离,其中 $p \in P_1$,$q \in P_2$. 如果 $\{p, q\}$ 是 S 中的最接近点对,则必有 distance$(p, q) < d$. 对于 P_1 中的任意一点 p,满足这个条件的 P_2 中的点一定落在一个 $d \times 2d$ 的矩形 R 中. 可以证明矩形 R 中最多只有 6 个 S 中的点作为 P_1 中每一个点的最接近点对的候选者. 因此,可以在 $O(n)$ 时间内完成 S_1 和 S_2 中解的合并. 一般在分治求解之前采用预排序技术,预先将 S 中的 n 个点按照 y 坐标排序. 分治算法中只对排好序的序列做一次线性扫描,即可完成合并. 因此,二维算法耗费的计算时间 $T(n)$ 与一维算法满足相同的递推方程,也可以在 $O(n * \log n)$ 时间内完成.

本文在上述算法的基础上,推广提出了利用分治法实现三维的最接近点对问题. 主要利用一个平面将 S 中的 n 个点划分为两个子空间区域中的点,并递归求解子区域中的最接近点对,然后合并解得 S 中的最接近点对.

2 分治法实现的三维最接近点对问题的推广算法

2.1 算法描述

三维情况中,S 中的点为空间中的点,每一个点都有 x、y、z 三个坐标值. 为了将空间中的点集 S 线性分割为大小大致相等的 2 个子集,我们选取一个垂直平面 P :$y = m$(m 为 S 中各点 y 坐标的中位数) 作为分割平面,将 S 中的点分割为 S_1 和 S_2 两个子空间中的点的集合,其中 $S_1 = \{p \in S \mid y(p) \leqslant m\}$,$S_2 = \{q \in S \mid y(q) > m\}$. 从而使得 S_1 和 S_2 分别位于平面 P 的左侧和右侧,且 $S = S_1 \cup S_2$.

类似一维和二维算法,递归地在 S_1 和 S_2 上解最接近点对问题. 设求解得到 S_1 和 S_2 中的最接近点对分别为 $\{p_1, p_2\}$ 和 $\{q_1, q_2\}$,对应的最小距离分别为 d_1 和 d_2,并设 $d = \min\{d_1, d_2\}$. 若 S 中的最接近点对的距离小于 d,则两个最接近点必分属于 S_1 和 S_2. 即 S 中的最接近点对的距离或者是 d,或者是某个 $\{p_3, q_3\}$ 点对的距离,其中 $p_3 \in S_1$,$q_3 \in S_2$. 如图 1 所示:

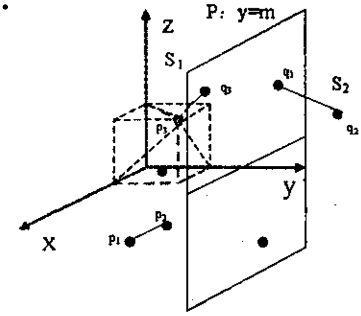

图 1 三维最接近点对问题

Fig. 1 Questions of paiv of three-dimensional points with the minimum distance

若 S 的最接近点对为 $\{p_3, q_3\}$,则 distancce$(p_3, q_3) < d$,那么 p_3 和 q_3 距平面 P :$y = m$ 的距离均小于 d. 我们用 P_1 和 P_2 分别表示垂直平面 P 左边和右边宽度为 d 的两个垂直长条空间区域,P_1 :$\{p \in S_1 \mid |m - y(p)| \leqslant d\}$;P_2 :$\{q \in S_2 \mid |y(q) - m| \leqslant d\}$,可知 $p_3 \in P_1$,$q_3 \in P_2$. 此时,P_1 中的所有点与 P_2 中的所有点均为最接近点对的候选者. 在最坏情况下,P_1 和 P_2 中包含了原 S 中所有的点,有 $n^2/4$ 对这样的候选者. 但是 P_1 和 P_2 中的点具有以下的稀疏性质,它使我们不必检查所有这 $n^2/4$ 对候选者. 考虑 P_1 中任意一点 p,它若与 P_2 中的点 q 构成最接近点对的候选者,则必有 distancce$(p, q) < d$. 满足这个条件的 P_2 中的点一定落在一个 $d \times 2d \times 2d$ 的长方体 C 中,如图 2 所示:

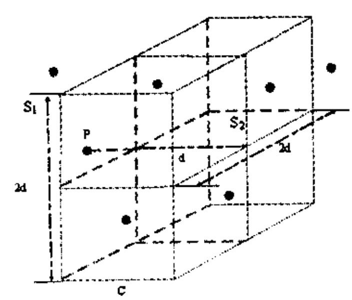

图 2 包含点 q 的 $d \times 2d \times 2d$ 的长方体 C

Fig. 2 The cuboid C $d \times 2d \times 2d$ of containmg point q

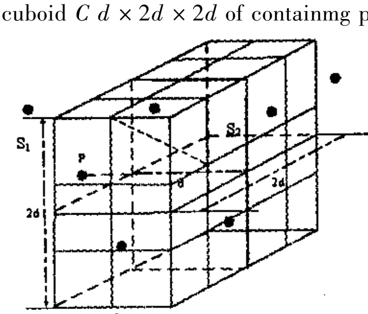

图 3 长方体 C 中点的稀疏性

Fig. 3 The sparseness of the point of cuboid C

由 d 的意义可知,P_2 中任何 2 个 S 中的点的距离都不小于 d. 由此可以推出长方体 C 中最多只有 24 个 S 中的点. 事实上,我们可以将长方体 C 的长为 $2d$ 的两条边分别 3 等分和 4 等分,将它的长为 d 的边 2 等分,由此导出 24 个大小相等的$(d/2) \times (d/2) \times (2d/3)$ 的小长方体. 如图 3 所示:
若长方体 C 中有多于24 个 S 中的点,则由鸽舍原理易知至少有一个$(d/2) \times (d/2) \times (2d/3)$ 的小长方体中有 2 个以上 S 中的点. 设 u,v 是这样 2 个点,它们位于同一小长方体中,则 $[distance(u,v)]^2 \leqslant (d/2)^2 + (d/2)^2 + (2d/3)^2 = 17d^2/18 < d^2$. 即 $\mathrm{distance}(u,v) < d$. 这与 d 的意义相矛盾. 也就是说长方体 C 中最多只有 24 个 S 中的点.

由于这种稀疏性质,对于 P_1 中任一点 p,P_2 中最多只有 24 个点与它构成最接近点对的候选者. 因此,在分治法的合并步骤中,我们最多只需要检查 $24 \times n/2 = 12n$ 对候选者,而不是 $n^2/4$ 对候选者.

为了确切地知道对于 P_1 中每个点 p 最多检查 P_2 中的哪 24 个点,我们可以将点 p 和 P_2 中所有 S_2 的点投影到平面 P :$y = m$ 上. 由于能与 p 点一起构成最接近点对候选者的 S_2 中点一定在长方体 C 中,所以它们在平面 P 上的投影点与点 p 在 P 上投影点的距离小于 d. 由上面的分析可知,这种投影点最多只有 24 个. 因此,若将 P_1 和 P_2 中所有 S 的点依次按其 x 坐标和 z 坐标排好序,则对 P_1 中任意点 p 而言,对已经按照 x 坐标和 z 坐标排好序的点列作一次线性扫描,就可以找出所有最接近点对的候选者. 设点 $q(x,y,z)$ 为 P_2 中可以与 P_1 中的一点 $p(x_0,y_0,z_0)$ 构成候选点对的排好序的24 个点中的一点,则满足 $x \in (x_0 - d, x_0 + d)$, $z \in (z_0 - d, z_0 + d)$,即投影点在以 p 的投影点为中心的 $2d \times 2d$ 的正方形区域中的点是我们要考察的候选点. 这就意味着我们可以在 $O(n)$ 时间内完成分治法的合并步骤.

2.2　算法的伪代码表示

至此,我们可以给出用分治法求解三维最接近点对的算法 spair 如下:

double spair (S)
{n =| S| ;/ * | S| 表示 S 中点的个数 * /
if (n < 2) return ∞ ;
1. m = S 中各点 y 坐标的中位数;
利用平面 P :$y = m$ 划分构造子集 S_1 和 S_2 ;
/ * $S_1 = \{p \in S | y(p) <= m\}$,$S_2 = \{p \in S | y(p) > m\}$ * /
2. d_1 = spair (S_1) ;
d_2 = spair (S_2) ;
3. dm = min(d_1,d_2) ;
4. 设 P_1 是 S_1 中距垂直分割面 P 的距离在 d_m 之内的所有点组成的集合;
P_2 是 S_2 中距垂直分割面的距离在 dm 之内的所有点组成的集合;
将 P_1 和 P_2 中点依其依次按照其 x 坐标值和 z 坐标值排序;
并设 X_1、X_2 是 P_1、P_2 依据 x 坐标值排好序的点列 ;Z_1、Z_2 是 X_1、X_2 再依据 z 坐标值排好序的点列;
5. 通过扫描 Z_1 以及对于 Z_1 中每个点检查 Z_2 中相继的最多 24 个点完成合并;
当 Z_1 中的扫描指针沿着某一个方向移动时,Z_2 中的扫描指针可在 $2d_m \times 2d_m$ 的方形区间内移动;
设 dl 是按这种扫描方式找到的点对间的最小距离;
6. d = min(d_m,dl) ;
return d ;
}

说明 :算法中的分割平面也可以是由各点 x 坐标值的中位数决定的,与 YOZ 面平行的平面 ;或者取各点 z 坐标值的中位数决定的,与 XOY 面平行的平面作为分割面. 其余坐标做相应改动.

2.3　算法的效率分析

下面我们来分析一下算法 spair 的计算复杂性. 设对于空间 n 个点的点集 S,算法耗时 $T(n)$. 算法的第1 步完成查找各点 y 坐标值的中位数并进行划分,我们已经知道中位数的查找算法[2] 可以在线性时间 $O(n)$ 内完成 ;第 2 步实现划分后子问题的求解,耗时 $2T(n/2)$;第 3 步和第 6 步用了常数时间 ;经过 2.1 中的分析可知第 5 步合并过程也用了 $O(n)$ 时间. 若在每次执行第 4 步时进行排序,则在最坏情况下第 4 步要用 $O(n * \log n)$ 时间. 这不符合我们的要求. 因此,在这里我们采取与二维算法类似的预排序技术,即在使用分治法之前,预先将 S 中 n 个点依次按其 x 坐标值和 z 坐标值排好序. 在执行分治法的第 4 步时,只要对已经排好序的点列作一次线性扫描,即可抽取出我们所需要的排好序的 $P1$ 和 $P2$ 中的点列 Z_1 和 Z_2. 然后,在第 5 步中再对 Z_1 作一次线性扫描,即可求得 dl. 因此,第4 步和第 5 步的两遍扫描合在一起只要用 $O(n)$ 时间. 这

样一来,经过预排序处理后的算法 $spair$ 所需的计算时间 $T(n)$ 也满足递推方程:

$$\begin{cases} O(1) & n < 4 \\ T(n) = 2T(n/2) + O(n) & n \geqslant 4 \end{cases}$$

求解此方程可得 $T(n) = O(n * \log n)$,预排序所需的计算时间也为 $O(n * 1\text{og}n)$. 因此,整个算法所需的计算时间为 $O(n * \log n)$.

3 结论

本文提出了用分治法实现的三维空间中最接近点对算法. 是对最接近点对问题从二维到三维空间形式推广的一种方法. 可以应用于复杂的空间交通控制系统中. 并论证了算法可以在 $O(n * \log n)$ 时间内完成. 在渐近的意义下,此算法已是最优的了.

参考文献:

[1] 王晓东. 算法设计与分析[M]. 北京 清华大学出版社,2002.

[2] Donald E. Knith. The Art of Computer Programming[M]. Addison – Wesley,1998.

[3] Sara Baase. Computer Algorithms Introduction to Design and Analysis, 3rd edition[M]. Pearson Education, 2000.

The Algorithm of Finding Pair of Three-Dimensional Points with the Minimum Distance by Means of Divide and Conquer

ZHANG Xiao-hong, HU Jin-chu

(1. *School of Mathematic and Comjruter Science*, *Shanxi Normal University*, *Linfen* 041004, *Shanxi*, *China*;
2. *Mathematics & Science College*, *Shanghai Normal Unviersity*, *Shanghai* 200234, *China*)

Abstract: Finding pair of dimensional points with the minimum distance is the key problem in the application of the air traffic control system and it is also a basic one of the computerized geometry study. By means of divide and conquer, the problem has been solved from the point of linearity and space, furthermore, it can be accomplished within the O(n * logn). Under the base of unidimensional and two – dimensional algorithm, this paper put forward an three-dimensional algorithm by means of divide and conquer and analyzed the efficiency of the algorithms.

Key words: Pair of Points with the Minimum Distance; Divide and Conquer; Three-dimension; Efficiency