三维空间的最接近点对问题

刘志辉

（中国民航大学 天津 300300）

摘要：最接近点对问题的求解就是在点集空间中求解最接近的一对点的距离。本文利用分治策略并结合预排序和分层映射技术把一维、二维情况下的最接近点对问题推广至三维，使问题在 $O(n*\log n)$ 时间内得以解决，相对时间复杂度为 $O(n^2)$ 的普通方法而言，效率得到很大的提高。

关键词：最接近点对；鸽舍原理；分层映射；

在文献[1]中，对一维和二维情况下的最接近点问题的求解进行了详细描述，于是引起了人们对三维情况下的最接近点对问题的关注。文献[2]中，提出了一种对三维最接近点对问题的求解方法：先对三维空间的点集进行两次预排序，利用与二维情况下相类似的合并算法，在 $O(n*\log n)$ 时间内求出最接近的点对。但是其合并过程所采用的算法不明确，令人质疑。本文只需对空间点集进行一次预排序，然后在合并过程中采用分层映射技术使其在 $O(n)$ 的时间内完成合并，从而整个算法的时间复杂度为 $O(n*\log n)$，并且算法思路清晰。可以证明在渐近意义下，此算法已是最优算法。

一、一维和二维情况下的最接近点对问题

把一维情况下点集 S 中的 n 个点退化为 x 轴上的 n 个实数 $x_1, x_2, \ldots, x_n$。于是最接近点对即为这 n 个实数中相差最小的两个实数。显然可以先将这 n 个实数排序好，映射到 x 轴上，然后用 x 轴上某个点 m 将 S 划分为大小大致相等的两个子集合 S_1 和 S_2，且 $S_1 = \{x \in S \mid x \le m\}$；$S_2 = \{x \in S \mid x > m\}$，如图 1 所示。于是求 S 中的最接近点对转变为分别求 S_1、S_2 中的最接近点对和其合并后的最接近点对。如果已求得 S_1 和 S_2 中的最小点对距离分别为 d_1 和 d_2，合并时只需

$O(1)$ 时间就能求得合并时的最接近点对，即 S_1 和 S_2 中分别离点 m 最近的一个点，如图 1 中的点 p_m 和 q_m。这样 S 中的最接近点对的距离为 $d = \max\{d_1, d_2, |p_m - q_m|\}$。从而一维情况下只需用一次线性扫描就可以找出最接近点对，算法中主要计算时间花费在排序上，总的时间为 $o(n*\log n)$。

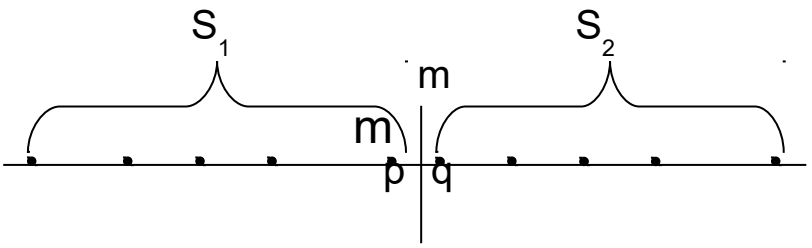

图 1 一维情况的最接近点对分治法

二维情况下的最接近点对可以表述为求 x y 平面中最接近的点对。先对平面中所有点按照 x 坐标进行排序，用 S 记排序后的点集，求出所有点 x 坐标值的中位数，记为 m，然后用直线 $l: x = m$ 把点集 S 划分为大致相等的两个子集 S_1 和 S_2，且 $S_1 = \{p \in S \mid x(p) \le m\}$；$S_2 = \{p \in S \mid x(p) > m\}$。这样把求 S 中的最接近点对转变为分别求 S_1、S_2 中的最接近点对和其合并后的最接近点对。子集中的最接近点对可以递归求得，但合并时最接近点对的求解比一维情况下要复杂些。设 d_1 和 d_2 分别为 S_1 和 S_2 中的最小距离，设 $d = \min\{d_1, d_2\}$。在合并时，对于 S 中距离小于 d 的两点 p 和 q 必定满足：p 和 q 分别属于 S_1 和 S_2，在此设 $p \in S_1$，$q \in S_2$；令 P_1 和 P_2 分别表示距直线 l 的左边和右边的宽为 d 的

两个区间，则 $p \in P_1$， $q \in P_2$，如图 2 左所示。按照正常思维， P_1 中的所有点和 P_2 中的所有点在最坏情况下有 $n^2/4$ 对最接近点对的候选者，但是 P_1 和 P_2 中的点具有稀疏性质，使得不必检查所有这 $n^2/4$ 个候选者，而最多只需要检查 $6\times n/2 = 3n$ 个候选者。 P_1 和 P_2 中的点的稀疏性表现在：对于 P_1 中的任意一点 p， P_2 中与其构成最接近点对的候选者的点必定落在一个 $d\times 2d$ 的矩形 R 中，如图 2 左所示；利用鸽舍原理， R 中这样的候选点最多只有 6 个，如图 2 右所示，6 个虚线矩形中每个矩形中最多只有一个候选点，具体描述可以参考文献[1]。因此，将 P_1 和 P_2 和中所有中点按其坐标 y 排好序，则对 P_1 中所有点最多只要检查 P_2 中排好序的相继 6 个点。但此时合并的计算复杂度需要 $O(n*\log n)$，并不是 $O(n)$，没有达到预想的效果，不过可以通过在求解问题之前，把 S 中的点按其坐标 y 进行预排序来解决。所以二维情况下的最接近点对可以在的 $O(n*\log n)$ 时间内求得。

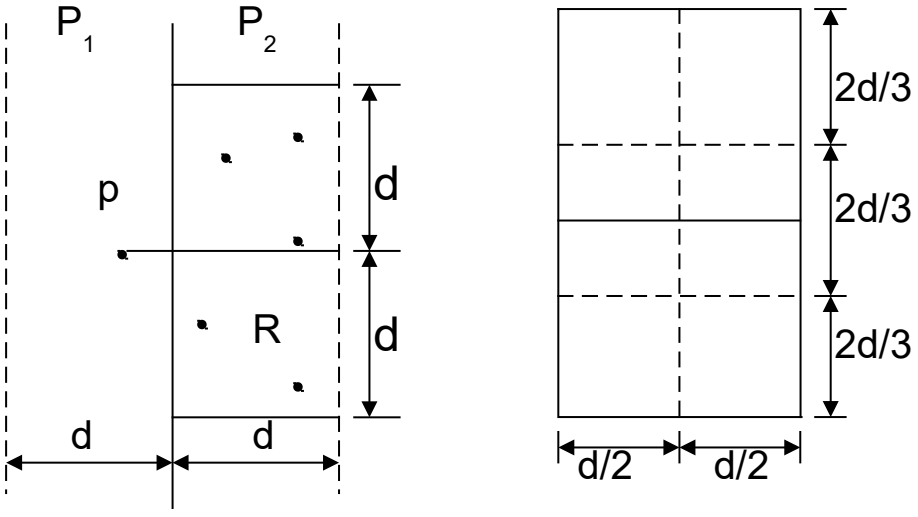

图 2 二维情况的最接近点对分治法

二、最接近点对问题的三维推广

有了一维和二维的最接近点对描述，可以依此推广到三维。对于三维空间，

所有的点有 xyz 三个坐标。于是三维情况下的最接近点对可以表述为求 xyz 空间中最接近的点对。

1、 问题描述

先对三维空间中所有点按照 y 坐标进行排序，用 S 记排序后的点集。求出所有点 y 坐标值的中位数，记为 m ，然后用平面 $l: y = m$ 把点集 S 划分为大致相等的两个子集 S_1 和 S_2 ，且 $S_1 = \{p \in S \mid y(p) \le m\}$； $S_2 = \{p \in S \mid y(p) > m\}$。这样把求 S 中的最接近点对转变为分别求 S_1 、 S_2 中的最接近点对和其合并后的最接近点对。然后依此方法递归地在 S_1 和 S_2 中求解最接近点对，设 d_1 和 d_2 分别为 S_1 和 S_2 中的最小距离，设 $d = \min\{d_1, d_2\}$。但求解合并时最接近点对比一维和二维都要复杂得多，成为问题求解的难点。

合并时，对于 S 中距离小于 d 的两点 p 和 q 必定满足： p 和 q 分别属于 S_1 和 S_2 ，在此设 $p \in S_1$， $q \in S_2$。那么 p 和 q 距平面的距离均小于 d 。因此，若令 P_1 和 P_2 分别表示距平面 l 的左边和右边的宽为 d 的两个长条形区间，则 $p \in P_1$， $q \in P_2$，如图 3 左所示。此时， P_1 中的所有点和 P_2 中的所有点在最坏情况下有 $n^2/4$ 对最接近点对的候选者。但是 P_1 和 P_2 中的点与二维情况下具有类同的稀疏性质，使得不必检查所有这 $n^2/4$ 个候选者。 P_1 和 P_2 中的点的稀疏性表现在：对于 P_1 中的任意一点 p ， P_2 中与其构成最接近点对的候选者的点必定落在一个 $d \times 2d \times 2d$ 的长方体 R 中，如图 3 左所示；利用鸽舍原理， R 中这样的候选点最多只有 24 个[2]，如图 3 右所示，24 个虚线小长方体中每个长

方体中最多只有一个候选点。因为对于如图 3 右中，每一个小长方体 $(d/2)\times(2d/3)\times(d/2)$ 中如果有 2 个以上 S 中的点，设 u 和 v 是这样的 2 个点，则

$$(x(u)-x(v))^2+(y(u)-y(v))^2+(z(u)-z(v))^2=\frac{17}{18}d^2$$

因此，$dist(u,v)<d$，这与前面 d 的意义相矛盾。也就是说长方体 R 中最多只有 24 个 S 中的点，其实数字 24 并不重要，重要的是长方体 R 中最多只有常数个这样的点。因此，在分治法的合并过程中，最多只需要检查 $24\times n/2=12n$ 个候选者，而不 $n^2/4$ 个候选者。在此并不能意味着可以在 $O(n)$ 时间内完成分治法的合并过程，因为对于 P_1 中的任意一点，我们并不确切知道要检查 P_2 中哪 24 个点。

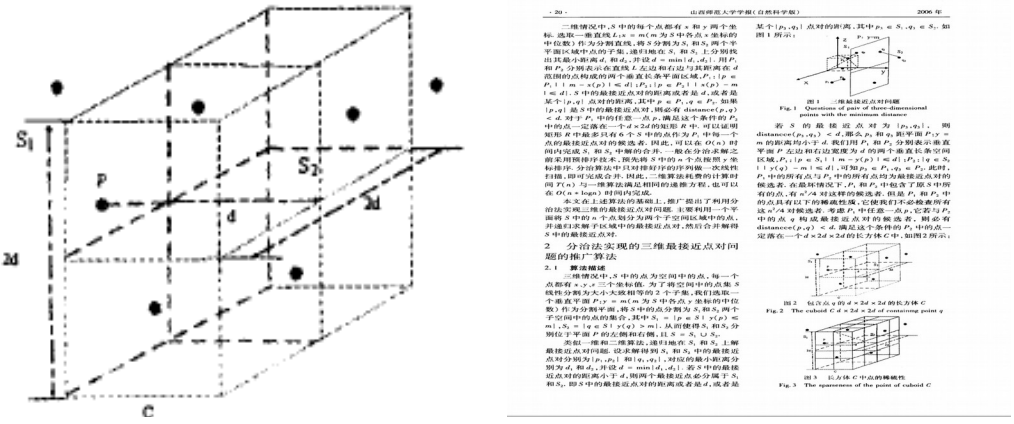

图 6 二维情况的最接近点对分治法

2、 分层映射和预排序

为了解决以上合并过程中遇到的问题，在[2]中先将 P_1 和 P_2 中的所有点按其 x 坐标排好序，然后对排好序的 P_1 和 P_2 再进行 z 坐标排序，也就是在此进行两次排序。令经过这样的两次排序后，P_1 和 P_2 变为 Z_1 和 Z_2。这时对于 Z_1 中的每一点最多只要检查 Z_2 中的相继 24 个点，因此对于 Z_1 中所有点，作一次

扫描，就可以在 Z_2 中找出所有最接近点对的候选者。但这种方法令人质疑，且要经过两次排序，在空间点数很大时，效率提高得不明显。

相对上面的不足，分层映射显出了它的优越性。在合并时，先对于 P_1 和 P_2 中的所有点顺 z 轴方向分层映射到长为 $2d$ 的长条形区域内，然后分别对各层中 P_1 和 P_2 中的点按其 x 坐标排序。接着对每一层中排好序的 P_1 中的点只需检查同层中排好序的 P_2 中的相继 24 个点，因此对于排好序的 P_1 中所有点，逐层扫描，通过一次扫描，就可以在排好序的 P_2 中找出所有最接近点对的候选者。

具体的分层映射过程为：找出 P_1 和 P_2 的合集中 z 坐标值最小的点 o；从 o 点开始，顺着 z 轴，每间距 $2d$ 作一平行于 xoy 面的分割面，如此分割到 P_1 和 P_2 的合集中 z 坐标值最大的点或包括 z 坐标值最大的点终止。这时，P_1 和 P_2 中的所有点就分别被映射到分割的长条形区域中，每一层可以用一个数组来保存所映射的点集。在此还要注意一个问题，在前面讨论 P_1 和 P_2 中的点的稀疏性质时，要求对于 P_1 中的任意一点 p，P_2 中与其构成最接近点对的候选者的点必定落在以从 p 点引出的一条垂直于分割平面 l 的直线为中心轴的一个 $d \times 2d \times 2d$ 的长方体 R 中，而 P_1 中的点并不能保证都位于每一层的中平面上，所以对于每层中非中心平面上的 P_1 中的点，还必须要投影到相邻的分割层中。投影的原则为：中心平面以上的点投影到与该层上相邻的分割层中；中心平面以下的点投影到与该层下相邻的分割层中。

很容易发现，采用以上分层映射技术和排序的方法来完成合并，在最坏情况下其计算复杂度需要 $O(n*\log n)$，并不是 $O(n)$，因此这种效果是不理想的。产生原因就在于排序时消耗了 $O(n*\log n)$ 的时间。这时我们可以采用二维情况下的最接近点对的预排序技术来降低合并时排序所耗费的不必要的时间，使合并在的 $O(n)$ 时间内完成，即在问题求解之前对中的所有点进行一次 x 坐标预排序。

3、 分层轴的选取

对于三维情况下的最接近点对问题的求解还牵涉到一个分层轴的选取过程。令 V 为包含三维空间中所有点的最小边界长方体，d_x 为 V 在 x 轴方向的长度，d_y 为 V 在 y 轴方向的长度，d_z 为 V 在 z 轴方向的长度。那么选取 $\min\{d_x, d_y, d_z\}$ 的方向轴为分层轴，这样有利于减少分层层次，降低分层的时空开销。对于 d_x、d_y 和 d_z 值差不多的情况，任意选取一个坐标轴作为分层轴。

4、 算法伪代码描述

结合以上描述，用分治法求三维点集最接近点对的算法 Cpair3 如下：

```
bool Cpair(S,d)
{
    n=|S|;
    if(n<2){
        d= ∞ ;
        return false;
    }
    SelestAxis(S,x,y,z); //选取分层轴,在此设选取的分层轴为 Z 轴
  X=SortX(S); //对 S 中所有点进行 X 坐标排序
  Y=SortY(S); //对 S 中所有点进行 Y 坐标排序
  Cpair(X,Y,d,n);
  return true;
```

}

void Cpair(X,Y,d,n)

{

if(n<2){

d= ∞ ;

return ;

}

m=Y 中各点 y 坐标的中位数;

构造 Y_1 和 Y_2; // $Y_1 = \{p \in Y \mid y(p) \le m\}, Y_2 = \{p \in Y \mid y(p) > m\}$

Cpair(X,Y_1,d_1,n/2);

Cpair(X,Y_2,d_2,n-n/2);

d_m=min(d_1, d_2);

结合预排序 X，构造 P_1 和 P_2，并对 P_1 和 P_2 中所有点顺 Z 轴进行分层映射; //

$P_1 = \{p \in Y_1 \mid\mid y(p) - m \mid \le d_m\}, P_2 = \{p \in Y_2 \mid\mid y(p) - m \mid > d_m\}$

逐层扫描每一层，对每层中 P_1 中的每个点检查同层的 P_2 中与其距离在 d_m 之内的所有点;

按照上述扫描方式找到的最近点对的距离 d_n;

d=min{d_m,d_n};

}

5、 算法效率

由前面的问题描述可知，整个问题的求解过程需要两次排序，一次用于分治划分，一次用于合并时的点对扫描，所以用于排序的时间为$O(n*\log n)$。由于在合并过程中消耗时间为$O(n)$，因此用分治法求解问题所耗费的时间$T(n)$可以用下式表达：

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

计算得出$T(n) = O(n\log n)$，结合排序时所消耗的时间，整个算法可以在$O(n*\log n)$时间内完成。

三、总结

三维情况下的最接近点对问题在日常生活中经常遇到，特别是在民航的空中管制工作中。本文采用预排序和分层映射技术，使问题可以在$O(n*\log n)$的时间内得到解决，相对时间复杂度为$O(n^2)$的普通方法而言，效率得到很大的提高。但同时也会发现时间效率的提高是以分层映射时的空间消耗为代价的。在侧重时间效率而空间复杂度要求不高时，该算法显现出了巨大的优势。

参考文献：

[1] 王晓东．计算机算法设计与分析(第三版)．电子工业出版社．2007

[2] 张晓红，胡金初．分治法实现最接近点对问题的三维推广算法．山西师范大学学报．2006