doi: 10.3969/j.issn.1006-1576.2011.04.021

超声探伤点云数据拓扑关系的建立

张春成
（海军装备部重庆军事代表局，重庆 400042）

摘要：针对超声探伤点云数据的特征，提出一种改进型的点云拓扑关系建立的算法。采用栅格法先对点云进行空间划分，然后在点所处附近栅格内进行邻域搜索。结果表明，与传统的算法相比，该算法能减少计算量，节省时间和内存，提高效率。

关键词：超声探伤；点云数据；拓扑关系；栅格划分；球邻域搜索
中图分类号：TP301.6　　**文献标志码**：A

Establishment Topological Relations with Ultrasonic Flaw Detectable Point Cloud Data

Zhang Chuncheng
(Chongqing Military Representative Bureau of Navy Equipment Department, Chongqing 400042, China)

Abstract: Aiming at the characteristics of point cloud data by ultrasonic flaw detection, there is a modified algorithm provided about creation of topological relations of point cloud. Use a modified grid algorithm for space division; and uses search sphere neighborhood for search neighborhood. Compared with the traditional algorithm, the result shows that it has less computation, so it saves time and memory, and improves efficiency.

Keywords: ultrasonic flaw detection; point cloud data; topological relation; grid division; sphere neighborhood search

0　引言

超声探伤目前主要有 2 大难题。一是准确“三定”问题：准确确定缺陷的尺寸、方位和性质；二是检测结果的可靠性低：由于缺陷不能成像，加之手工操作的主观性，超声检测结果的可靠性相对较差。随着超声探伤技术的发展，特别是相控阵技术和多椭圆定位技术的发展，使用计算机实现基于海量点云数据的三维缺陷表面重建成为可能。

超声探伤设备获得的数据量非常庞大，并且没有任何拓扑关系，要快速建立离散点之间的拓扑关系成为缺陷表面重建的关键。如果直接对点进行全局搜索，那么数据量将非常大，会很耗时。因此，必须提高点的邻域搜索速度。针对超声探伤点云数据的特点，笔者提出适合此种点云数据拓扑关系建立的方法，采用栅格法先对点云进行空间划分，然后在点所处附近栅格内进行邻域搜索。

1　点云数据的空间划分方法简介

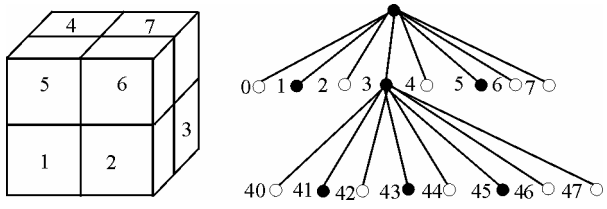
1.1　八叉树法

八叉树空间划分是用一个最小立方体包围点云数据，并将该立方体作为八叉树的根节点，然后把这个立方体划分成 8 个大小相同的小立方体，把每个小立方体作为根节点的子节点。对每个包含点的子节点立方体以递归的方式进行八叉划分，直到子立方体的边长小于或者等于给定的边长。

把 1 棵 $2^n \times 2^n \times 2^n$ 的八叉树空间八等分以后，利用 0～7 的八进制编码序号对八叉树进行编码，在八叉树空间中，任意一个子立方体的位置可以用一个八进制数唯一表示：

$$Q = q_{n-1}8^{n-1} + q_{n-2}8^{n-2} + \cdots + q_k 8^k + \cdots + q_1 8^1 + q_0 8^0 \quad (1)$$

式 (1) 中，q_k 为八进制数编码，$q_k \in [0,7]$，$k \in [0, n-1]$。其中，q_k 表示结点在其兄弟结点中的序号，q_{k+1} 表示 q_k 的父结点在其兄弟结点中的序号。通过这种方式，从 q_0 到 q_{n-1} 就可以完整地表示出八叉树的每个叶子结点到跟结点的路径。点云数据的八叉树划分与编码表示，如图 1。



(a) 空间划分　　(b) 结点编码

图 1　八叉树划分与编码表示

如图 1(b)，黑点结点的编码为：{41,43,45}。每一个叶子结点的编码可以根据八叉树空间左下角坐标值（其中，左下角坐标为八叉树空间最小坐标

收稿日期：2010-12-09；修回日期：2011-01-25
作者简介：张春成（1982—），男，重庆人，学士，助理工程师，从事通信与信号处理、机电工程研究。

值的顶点坐标）x、y、z的二进制计算出来：

$$\begin{cases} x = a_{n-1}8^{n-1} + a_{n-2}8^{n-2} + \cdots + a_k 8^k + \cdots + a_1 8^1 + a_0 8^0 \\ y = b_{n-1}8^{n-1} + b_{n-2}8^{n-2} + \cdots + b_k 8^k + \cdots + b_1 8^1 + b_0 8^0 \\ z = c_{n-1}8^{n-1} + c_{n-2}8^{n-2} + \cdots + c_k 8^k + \cdots + c_1 8^1 + c_0 8^0 \end{cases} \tag{2}$$

由八叉树空间中各小立方体单元之间的相互关系可以得 $q_k = a_k 2^0 + b_k 2^1 + c_k 2^2$。在已知八叉树空间中立方体单元的编号时，也可以计算出该立方体左下角的坐标值：

$$\begin{cases} x = \sum_{i=1}^{n-1} |q_i|_2 \times 2^i \\ y = \sum_{i=1}^{n-1} |q_i / 2|_2 \times 2^i \\ z = \sum_{i=1}^{n-1} [q_i / 4]_2 \times 2^i \end{cases} \tag{3}$$

式 (3) 中，$|q_i|_2$ 表示 q_i 除以 2 的余数，$[q_i/4]_2$ 表示 q_i 除以 4 取整，n 是八叉树空间的分层数。

对于一个给定的点云数据，当进行八叉树划分的层数不同时，划分的立方体大小及其包含的数据点个数也不同，从而导致搜索空间中的非近邻点个数不同。如果划分的层次不够合理，就会影响邻域点的搜索速度。

1.2　k-d tree 法

对于 k 维空间数据的查询来说，k 维二叉索树（k-dimensional binary search trees，k-d tree）是一种有效的数据结构。k-d tree 法是一种基于二叉树的坐标轴分割法，它对二维点集分割的过程如图 2。首先，按 X 坐标轴寻找分割线，遍历所有数据点计算 X 坐标的平均值，通过最接近该平均值的点做 1 条平行于 Y 坐标直线，将数据点分割成 2 部分；然后对分割成的 2 个子空间分别按 Y 坐标轴寻找分割线，遍历子空间中的数据点计算 Y 坐标的平均值，通过最接近 Y 坐标平均值的点做 1 条平行于 X 轴的直线，将子空间分割成 2 部分。以此反复对数据点进行分割，直到分割的区域剩 1 个点为止。

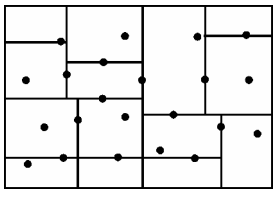


图 2　二维 k-d tree 构造示意图

k-d tree的分割结果对应一个二叉树，树的非叶子结点对应一个数据点和通过该点的分割线，每个叶子结点对应一个数据点。通过二叉树的形式建立点的拓扑关系，k-d tree法也可扩展到三维空间数据点的分割。

推广到三维空间的k-d tree树与八叉树划分结果有一定的相似之处，其结果使数据点的空间划分更为均匀，对于分布不均匀的点，云数据能够取得一个较好的搜索效率。但由于是通过二叉划分的，更深一层的划分会使划分区域的点数成倍缩小，因此很难控制一个合理的划分使搜索速度最优。

1.3　栅格法

栅格划分是一种常用且有效的划分策略。对于空间散乱点云数据，首先求出所有数据点坐标的最大值与最小值，然后根据这些最大最小坐标值建立一个与坐标轴平行的最小长方体包围盒。根据给定的划分边长将长方体空间划分成一系列小立方体（称为栅格），把所有数据点归入到其对应的栅格中。

对于一个给定点云数据$\{P_i, i=1,2,\cdots,n\}$，其数据点在坐标轴方向上的极值分别为 $x_{\max}$、$x_{\min}$、$y_{\max}$、$y_{\min}$、$z_{\max}$、$z_{\min}$。按照给定边长 L 划分为 m 个栅格，其中：

$$m = \left\lceil \frac{(x_{\max} - x_{\min})}{L} \right\rceil \bullet \left\lceil \frac{(y_{\max} - y_{\min})}{L} \right\rceil \bullet \left\lceil \frac{(z_{\max} - z_{\min})}{L} \right\rceil \tag{4}$$

对点云数据进行划分后，栅格信息是由二维数组存储的，每个数据点存入其对应的栅格所对应的索引数组中。进行点的邻域搜索时，可先在点所在的栅格里搜索，如果失败，可在与本栅格相邻的栅格中搜索。这样，避免了对整个点云数据的搜索，大大减少了对非近邻点的计算，提高了搜索的效率。

栅格法是通过给定的栅格边长来对空间数据点进行剖分，可以灵活地控制栅格的大小，从而尽可能减少近邻点搜索过程中对非近邻数据点的处理。

2　超声探伤缺陷点的邻域搜索

2.1　超声探伤点云数据的特征和空间划分法的选取

利用相控阵超声检测技术、衍射时差和多椭圆定位（time of flight diffraction and locating with ellipses，TOFD-LWE）超声检测技术能得到缺陷表面的三维点集（点云）数据。点云数据是杂乱无序的，但同一缺陷的点云数据的获取时间相对集中。在探伤得到的 n 个点存储二维数组 point[$\cdots, x_i\ y_i\ z_i, \ldots]_{n\times 3}$ 中，i 为点$\{x_i, y_i, z_i\}$的索引，$1 \le i \le n, i \in Z$。数组中每个点顺序是按它们产生的时间先后顺序来排列的，同一缺陷的点相对集中。判断两点是否在同

一缺陷的距离阈值为 L：如果两点之间的距离大于 L，则判定它们分别属于不同的 2 个缺陷；如果距离小于等于 L，则判定它们属于同一个缺陷。

假设以 L 为空间划分的最小立方体单元，有一点 p 存在这样单元中，要寻找 p 点的邻域，则只需要在这个单元及其外围一圈的单元中进行邻域搜索。因为超出这个邻域外的点与 p 点距离大于 L，也就意味着它们不属于同一缺陷，也就没有邻域搜索的意义。这样种假设的方法与栅格算法一致；另外，栅格算法的思想较八叉树法和 k-d tree 法清晰和简单，计算复杂度相对小些，更容易编程实现，所以笔者采用栅格算法进行超声点云数据的空间划分。

2.2 超声探伤点云数据的栅格化

根据超声探伤缺陷点云数据的特征以及缺陷三维表面重建的需要，采用栅格法来对点云数据进行空间划分，并且对栅格算法做了一些修改。

2.2.1 点所处栅格号的计算

判断两点是否在同一缺陷的距离阈值为 L，将 L 作为栅格边长。将 point 数组中的第 1 个点 $p_1=(x_1, y_1, z_1)$作为栅格算法的参考点。对数组中序号为 i 的点 p_i，其所属栅格号为：

$$g=(g_x,\ g_y,\ g_z)=\begin{cases}g_x=\lfloor(x_i-x_1)/L\rfloor\\ g_y=\lfloor(y_i-y_1)/L\rfloor\\ g_z=\lfloor(z_i-z_1)/L\rfloor\end{cases}\tag{5}$$

$\lfloor x\rfloor$的含义为：小于等于 x 的最大整数，例如 $\lfloor 0\rfloor=0,\lfloor 0.1\rfloor=0$。$g_x$ 表示栅格号在 x 轴上的分号，g_y 表示栅格号在 y 轴上的分号，g_z 表示栅格号在 z 轴上的分号。

2.2.2 点云数据栅格化数据结构的建立

建立存储栅格号的数组 grid[]$_{\times 3}$={…,g_x g_y g_z,…}。栅格号的排序规则为：把 $g_xg_yg_z$ 看作一个数值，按从小到大的先后顺序排列。栅格号为{0 0 0}的栅格是初始栅格。

建立栅格化后，点云数据的坐标索引数组 coordIndex[]={…, $I_{g_xg_yg_z}$,…}。其中，$I_{g_xg_yg_z}$ 表示 $g_xg_yg_z$ 号栅格内点索引的集合，每个索引之间用空格隔开。每个栅格点云索引集合之间用逗号隔开。coordIndex[]中的 $I_{g_xg_yg_z}$ 与 grid[]$_{\times 3}$ 中的{g_x g_y g_z}为一一对应关系，它们在各自数组中的索引（序号）相同，各自前面的逗号索引也是相同的。

2.3 点的邻域搜索

给定一个点 p，它所在的栅格称为 0 级栅格（只有 1 个），向外扩展一周的栅格称为该点的 1 级栅格（26 个），再向外扩展一周的栅格称为该点的 2 级栅格（98 个），以此类推，p 向外扩展 n 次，得到的是它的 n 级栅格，n 级栅格共包含 $(2n+1)^3-(2n-1)^3$，其中 $n\geq 1$。

点 p 的 0 级搜索是指，在点 p 所在的栅格中进行邻域点的搜素。点 p 的 1 级搜索是指，在点 p 的 0 级栅格以及 1 级栅格（共 27 个栅格）中进行搜索。以此类推，点 p 的 n 级搜索是指，在以 0 级栅格为中心的包含 $(2n+1)^3$ 个栅格的立方体内进行搜索。

设点 p 的邻域是以 p 为圆心、L 为半径的球体。点 p 的邻域搜索是指在点云数组 point[]$_{n\times 3}$ 中，寻找与 p 点距离小于等于 L 的点。点 p 的 1 级以外的栅格内的点云与它距离大于 L，因此，对该点的邻域搜索只进行 1 级栅格搜索。点的邻域搜索大致分为 4 步：1) 运用式 (5) 找到点 p 所在的栅格，其栅格号为：$G_0=(g_x, g_y, g_z)$；2) 计算出点 p 的 1 级搜索栅格号为：$G_1=\begin{pmatrix}g_x-1 & g_y-1 & g_z-1\\ g_x & g_y & g_z\\ g_x+1 & g_y+1 & g_z+1\end{pmatrix}$，共计 3×3×3 个栅格；3) 求点 p 的邻域栅格：$G_L=G_1\cap$grid[]$_{\times 3}$；4) 在 G_L 对应的点云中，逐点与点 p 进行比较筛选，找出距离不大于 L 的点，这些点就是点 p 的邻域。

3 点云数据拓扑关系建立的流程

点云数据拓扑关系的建立分为点云数据的栅格化和点的球邻域搜索 2 部分。对于超声探伤缺陷散乱点云数据 point[]$_{n\times 3}$，其中任意一点 p_k 的半径为 L 的球邻域记为 $N_{p_k}^L$，计算 p_k 的邻域算法如下：

1) 初始化。创建栅格号数组 grid[]$_{\times 3}$ 和栅格化后坐标索引数组 coordIndex []。读取点云数组 point[]$_{n\times 3}$ 中的第 1 点 p_1 作为点云栅格化的参考点，并计算出该点所在的栅格号为(0 0 0)，作为初始栅格存储在数组 grid[]$_{\times 3}$ 中；把 p_1 的索引存储在数组 coordIndex []中。

2) 在数组 point[]$_{n\times 3}$ 中，读取上一步骤处理点的下一点 p_i。

3) 计算出 p_i 所在的栅格号 g^{p_i}。在数组 grid[]$_{\times 3}$ 中查找 g^{p_i} 是否存在。

（下转第 80 页）

```
/**************接收离散量线程**************/
DWORD WINAPI ReceiveDIThread(LPVOID lpParam)
{
  while(1)
  {
    getword=GetPortWord(HwCtrl,DI_PortAddr);//读取 DI_PortAddr 地址的离散量数据
    if((getword&receive_DI_right[send429_num])==receive_DI_right[send429_num])
    {
      receive_DI_A[send429_num]=getword;
      break;
    }//将接收数据与标准数据比对
    ……
  }
}
```

4 结束语

实际应用表明：该测试仪设计合理，工作可靠稳定，操作维护方便，功能上完全满足对某型飞机交联组件的测试需要，有效地提高了检测和诊断效率。

参考文献：

[1] 杨军锋, 朱家海, 谢红星. 航空测试仪器中的 PC/104 总线技术[J]. 空军工程大学学报: 自然科学版, 2000, 1(4): 13-16.

[2] 杨勇智, 黄胜伦, 冯和军, 等. 基于 PC/104 总线的某型飞机武器控制系统的 ATS[J]. 空军工程大学学报: 自然科学版, 2003, 4(3): 36-39.

[3] 党广利, 冯金富, 闫威. 某型机载武器控制系统自动测试仪的研制[J]. 计算机测量与控制, 2003, 11(5): 362-364.

[4] 鲁兴举, 郑志强, 彭学锋. 基于 ARINC429 总线的导弹综合测试系统[J]. 兵工自动化, 2005, 24(2): 14-15.

[5] 周明光, 马海潮. 计算机测试系统原理与应用[M]. 北京: 电子工业出版社, 2005: 151-160.

[6] 田鹏飞, 胡昌华, 何华锋, 等. 基于 VXI 总线的开关量 I/O 模件设计[J]. 弹箭与制导学报, 2010, 30(2): 21-24.

[7] 杨士元. 数字系统的故障诊断与可靠性设计[M]. 北京: 清华大学出版社, 2000: 100-114.

[8] 郑慧, 范忠诚. 零基础学 Visual C++[M]. 北京: 机械工业出版社, 2008: 315-330.

****

（下转第 66 页）

如果存在，则获取 g^{p_i} 在数组 grid[]$_{\times 3}$ 中的索引赋值给 gindex，在数组 coordIndex []中找到索引为 gindex 的 $I_{g_xg_yg_z}$，把 i 存储在 $I_{g_xg_yg_z}$ 中。如果不存在，在数组 grid[]$_{\times 3}$ 中，找到小于 $g_xg_yg_z$ 的最大栅格号赋值给 mmg，获取它的索引赋值给 mmgindex，把 g^{p_i} 存储在这个栅格号的后面，在数组 coordIndex[]中到索引为 mmgindex 的 $I_{g_xg_yg_z}$，在其后新建一个 $I_{g_xg_yg_z}$，把 i 存储在新建的 $I_{g_xg_yg_z}$ 中。

4) 流程跳到第 2 步。这样流程就进入了循环，直到程序在进行第 2 步时不成功，这就表示点云数据的栅格化已经完成。此时流程进入第 5 步。

5) 对于寻找球领域的点 p_k，建立其领域索引数组 LcoordIndex[]$_{\times 1}$。计算出该点理论上的 1 级搜索栅格号 $G_1^{p_k}$，共计 27 个。建立 1 级搜索栅格号数组 grid1pk[]$_{\times 3}$，把这 27 个栅格号按由小到大的先后顺序存储在这个数组里。读取 grid1pk[]$_{\times 3}$ 中的第一个栅格号。

6) 将当前读取的栅格号与 grid[]$_{\times 3}$ 求交集。如果交集为空，则流程进入第 7 步。如果交集不为空集，则根据该栅格号在 grid[]$_{\times 3}$ 中的索引，在数组 coordIndex[]中找到该栅格号的 $I_{g_xg_yg_z}$，求出每个点与 p_k 的距离，将距离小于 L 的点的索引存储在 LcoordIndex[]$_{\times 1}$ 中。

7) 读取 grid1pk[]$_{\times 3}$ 中的下一个栅格号。流程跳到第 6 步，程序进入循环。直到程序在“读取 grid1pk[]$_{\times 3}$ 中的下一个栅格号”时失败，这表示 p_k 的领域搜索完成，$N_{p_k}^{L}$ = LcoordIndex[]$_{\times 1}$。

4 结束语

该改进型栅格算法与传统的栅格算法相比，不用寻找点云数据在各坐标轴上的最大、最小值，不用寻找点云数据的包围合，不用存储不包含点的栅格号；提出的球领域搜索算法与 K 领域搜索算法相比，不用进行多级栅格搜索。因此，减少了计算量，节省了时间和内存，提高了效率。下一步，还将对该算法进行验证。

参考文献：

[1] 赵伟玲. 三维点云的数据预处理和圆提取算法研究[D/OL].万方数据, [20100-08-14]. http://d.g.wanfangdata.com.cn/Thesis_Y1436701.aspx.

[2] 刘立强. 散乱点云数据处理相关算法的研究[D/OL]. 万方数据, [20100-10-8]. http://d.g.wanfangdata.com.cn/Thesis_Y1678646.aspx.