第42卷 第3期 西 安 交 通 大 学 学 报 Vol.42 №3
2008年3月 JOURNAL OF XI′AN JIAOTONG UNIVERSITY Mar. 2008

多分散系统不同粒径颗粒碰撞的多重八叉树搜索算法

鲁录义[1]，周逢森[1]，冯诗愚[1]，顾兆林[2]
(1.西安交通大学能源与动力工程学院，710049，西安；2.西安交通大学人居环境与建筑工程学院，710049，西安)

摘要：利用线性八叉树的拓扑结构对八叉树大小邻居搜索算法进行改进，在V氏八叉树颗粒搜索算法的基础上提出了一种快速预判大小颗粒碰撞的多重八叉树搜索算法.新算法对各种粒径分布的颗粒系统均有较好的适应性，且受颗粒形状和堆积密度的影响较小.对一个包含大中小3种粒径的颗粒系统进行计算，并与V氏八叉树颗粒搜索算法结果进行比较，发现多重八叉树搜索算法在运行时间上有非常强的优势.

关键词：颗粒碰撞；多分散系统；八叉树
中图分类号：TQ026.7 **文献标志码**：A **文章编号**：0253-987X(2008)03-0304-05

Mutil-Octree Searching Algorithm for Different Size Particle Collision in Polydisperse Systems

LU Luyi[1], ZHOU Fengsen[1], FENG Shiyu[1], GU Zhaolin[2]
(1. School of Energy and Power Engineering, Xi′an Jiaotong University, Xi′an 710049, China; 2. School of Human Settlements and Civil Engineering, Xi′an Jiaotong University, Xi′an 710049, China)

Abstract: Preliminary contact detection algorithm for particle collision plays a very important role in the discrete element model for polydisperse flow systems. In this study, the neighbor finding algorithms of octree were improved using the topology of linear octree. A mutil-octree contact detection algorithm for different size particle collision was developed based on Vemuri-octree contact detection algorithm. The present algorithm is applicable in the polydisperse systems with different particle size distribution, irrespective of solid particle volumetric fraction and particle shape. The simulation of three-size particle collision indicates that the present algorithm is faster than Vemuri-octree contact detection algorithm.

Keywords: particle collision; polydisperse; octree

近年来，离散单元法已发展成为散体力学分析中的一种有效工具，特别是在离散度大、流动显著的颗粒流系统问题分析中[1].由于离散单元模型(DEM)[2-3]能够对单个颗粒进行跟踪，提供颗粒运动过程中的详细信息(如速度、位置等)，模拟结果要优于连续模型(CM)[4]，但它的发展却受计算机硬件条件的限制，目前所能模拟的颗粒数限于10万量级上.一个包含庞大颗粒数目的系统，检测颗粒是否发生碰撞就是其中一个相当复杂的问题，需要花费大量的CPU时间，甚至要占整个计算时间的60%[5].因此，开发一个快速有效的搜索碰撞算法早已成为颗粒系统模拟中重要的研究方向之一.

最原始的碰撞搜索法是N^2搜索算法(N为系统中的颗粒总数)，这种算法逐个比较一个颗粒与剩下$N-1$个颗粒的相对位置，确定整个系统内碰撞的计算量为$O(N^2)$.该算法的计算量随着颗粒数的增加将呈几何级的增长，极大地制约了计算速度，因此出现了不同的碰撞搜索算法，如边界球法[6]、区域搜索法[7]、边界盒子法[8]等.但是，这些搜索算法大都是以等直径圆形颗粒假设为基础的，当颗粒数目

收稿日期：2007-05-29. 作者简介：鲁录义(1980—)，男，博士生；顾兆林(联系人)，男，教授，博士生导师. 基金项目：国家自然科学基金资助项目(40675011)；陕西省科技攻关计划资助项目(2004K08-G15).

增加且非均一直径分布时，其计算量也可达$O(N^2)$，或因需要消耗大量系统资源而导致时间的剧增[9]，难以满足实际工程中颗粒流系统模拟的需要. 针对工程中的颗粒系统颗粒密度不均、形状各异的现象，Vemuri 等[9]提出了一种基于八叉树的碰撞预判算法(简称 V 氏八叉树算法，V-octree)，该算法可以处理任意形状的颗粒且碰撞计算量不随颗粒密度的增加而明显增大，因为网格划分得足够大，可以在任意方向上包容系统中最大的颗粒，同时对于系统中的 N 个颗粒采用八叉树邻居搜索算法，平均搜索时间为 $O(N)$，最差时搜索时间为 $O(Nh)$，h 为八叉树的高度.

纵观碰撞搜索法的发展，从 N^2 搜索算法到区域搜索法再到经典的 NBS(No Binary Search)搜索方法[5]，它们的计算量分别为 $O(N^2)$、$N/C \cdot O(C^2)$ (C 为小区域中的颗粒数)和 $O(N)$. 分析其中机理可知，降低计算量的有效方法是减少搜索范围和碰撞预判的次数，换句话说就是要做到邻居网格尺寸尽量小且邻居网格中的颗粒数目尽量少. Vemuri 等为了实现处理任意形状颗粒的碰撞，将网格划分得足够大，但采用大网格划分来解决形状复杂的问题却增加了碰撞预判的次数，在处理颗粒直径非均一分布时，特别是系统中小颗粒占有较大的比重时，此算法的计算时间就会有明显增加. 因此，为了有效地减少颗粒碰撞的计算量，需要对大小颗粒进行分类，采用合适的网格去包容颗粒，大小颗粒各自搜索大小不等的邻居. 基于这样的思想，开发了一种快速有效解决大小颗粒碰撞的多重八叉树算法.

1　显式八叉树的拓扑结构与邻居搜索

八叉树[10]是一种用于描述三维空间体的树状数据结构，一般采用线性八叉树方式进行存储. 为了更直观地说明问题，本文将八叉树退化为四叉树，即将三维问题变为二维问题，典型的线性四叉树如图 1 所示.

线性四叉树在编码上有方向性、层次性、可压缩性[11]，但更重要的是其具有分块连续性、拓扑结构的一致性和编码与坐标行列值的一一对应关系.

分块连续性：如图 1 所示，在〈00〉区(为了与十进制区别，本文用“〈〉”表示四进制)编码范围从 0 到 3，在〈03〉区从 12 到 15，在〈3〉区从 48 到 63. 当确定某个区的第一个编码后(本文定义为该区的首地址)，根据区域的大小立即可以推算出该区域内所有的编码，如〈03〉区的首地址即为 12.

编码与坐标一一对应：如图 1 所示，无论是四进制编码还是十进制编码，它们在空间位置上都是惟一的.

拓扑结构的一致性：八叉树在任何一层上始终有相同的拓扑结构，如图 1 中计算区域(粗黑线表示的大正方形)可分为〈0〉、〈1〉、〈2〉、〈3〉4 个区，而 2 区可以分为〈20〉、〈21〉、〈22〉、〈23〉4 个小区. 同时，〈1〉、〈2〉、〈3〉区的编码可以直接由〈0〉区的编码推算出. 在图 1 中，〈02〉表示第 2 个层次上的〈0〉区，而〈002〉表示第 3 个层次上的〈0〉区. 非常明显，〈002〉编码前移一位即可以得到编码〈02〉. 同理，〈001〉编码前移两位即可以得到编码〈1〉.

基于线性四叉树的拓扑结构，本文提出了一种四叉树快速搜索邻居的新方法. 在邻居搜索中利用结点所在行与列的信息和分块连续性的特点，实现了简单有效快速寻找结点的邻居，包括等邻居及大小邻居.

1.1　等邻居搜索

等邻居是指结点编码相同并与之相邻的结点，如图 1 中〈300〉是〈301〉的等邻居，〈21〉是〈23〉的等邻居. 对于等邻居采用的搜索方法是：首先通过结点的编号找到结点所在的行和列，结点的邻居必然在其相邻的行列中. 由于行列值与结点编码的一一对应关系，在确定相邻的行与列之后，就可以得到结点的邻居编码.

1.2　小邻居搜索

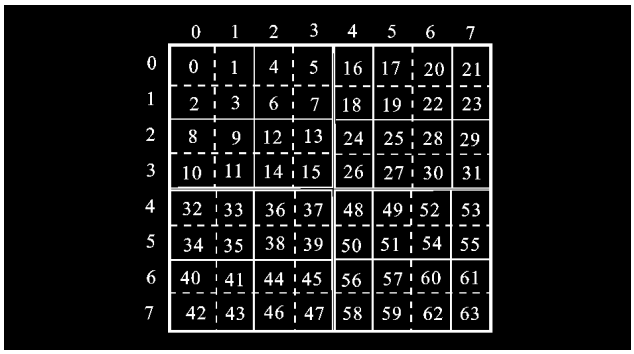
小邻居是指结点编码比之长且与之相邻的结点，如图 1 中〈300〉是〈21〉的小邻居，〈21〉是〈1〉的小邻居. 小邻居的搜索方法与等邻居的搜索过程大致相同，所不同的是并不直接得到小邻居的结点编码，而是找到与其同等邻居的编码，即小邻居的首地址. 由于四叉树是分块连续性的，通过结点所在的方位，可以非常快速地找到小邻居编码.

1.3　大邻居搜索

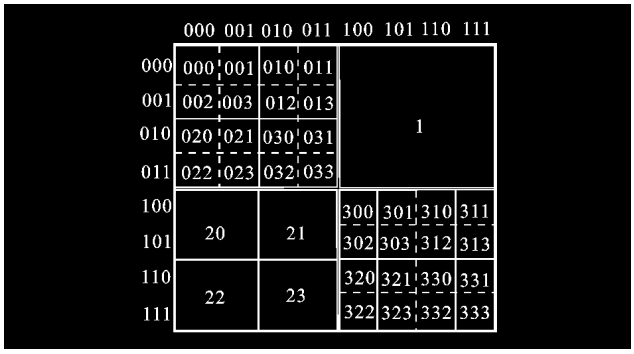
大邻居是指结点编码比之短并与之相邻的结点，如图 1 中〈21〉是〈033〉的大邻居，〈1〉是〈21〉的大邻居. 在大邻居搜索中，首先要进行的就是进位操作，得到与大邻居同等长度的结点编码，然后再采用与等邻居相同的搜索方法.

2　颗粒碰撞中的多重八叉树搜索算法

多重八叉树搜索基于空间分解，将计算区域(假定为正方形)用逐次二分的方法划分成为若干尺寸相同的正方形小网格(即叶子结点[10])，并将不同尺



(a)十进制



(b)四进制

图1 四叉树的十进制与四进制表示

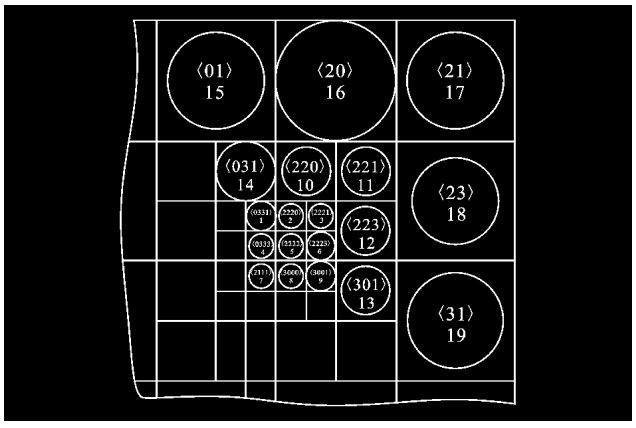
寸的颗粒映射到对应大小的网格中.映射采用的原则是视颗粒的最小外接球直径为颗粒的等效粒径,将粒径小于第 i 层网格边长且大于第 $i+1$ 层网格边长的颗粒映射到第 i 层网格中,所有比最细一层(j 层)网格边长小的颗粒都映射到第 j 层网格中.如图2所示,颗粒15～19的粒径比第2层网格(编码长度为2)的边长小但比第3层网格的边长大,按照映射原则将它们映射到第2层网格中,而颗粒1～9的粒径均比最细一层(第4层)网格的边长小,故将它们映射到第4层网格中.

为了方便讨论,将第2层网格中的颗粒称为大颗粒,第3层网格中的颗粒称为中颗粒,第4层网格中的颗粒称为小颗粒.在颗粒碰撞搜索过程中将碰撞对进行分类,分为小颗粒-小颗粒、小颗粒-中颗粒、小颗粒-大颗粒、中颗粒-中颗粒、中颗粒-大颗粒、大颗粒-大颗粒6种(碰撞搜索算法采用单向搜索,因为根据牛顿第三定律,一个碰撞对只需要计算一次碰撞).这6种碰撞对的搜索同时存在于一棵八叉树中,故称之为多重八叉树搜索算法.小-小、中-中、大-大3种同等直径的颗粒间的碰撞采用同等邻居搜索算法,而小-中、小-大、中-大均采用大邻居搜索算法.以图2的颗粒系统为例,具体搜索过程如下.

同等级颗粒碰撞搜索:颗粒5被映射到〈2222〉,那么颗粒5可能与映射到〈2222〉的9个等邻居中的颗粒1～9发生碰撞.

大等级颗粒碰撞搜索:颗粒5被映射到〈2222〉,〈2222〉的上一层网格编号为〈222〉,那么颗粒5可能与映射到〈222〉的9个大邻居中的颗粒10～14发生碰撞.同理,颗粒5可能与映射到〈22〉的9个大邻居中的颗粒15～17发生碰撞.

V氏八叉树算法以颗粒系统中最大颗粒的直径为最小网格的边长,对于图2所示的颗粒系统,V氏八叉树仅进行第2层的网格划分.与多重八叉树网格划分相比,一个网格中将映射更多的颗粒.以网格〈22〉为例,映射有颗粒2、3、5、6、10、11、12.参照表1(A、B、C 分别表示小颗粒、中颗粒和大颗粒的数目,M 表示基础网格的数量,即C映射所对应的网格数,也就是采用V氏八叉树算法大网格划分的网格数,那么 B 所对应的网格数量则为 $4M$,C 所对应的网格数量则为 $16M$)的公式分析可知:V氏八叉树算法的邻居搜索次数比多重八叉树的少,但预判碰撞次数却要多.若要比较V氏八叉树算法和多重八叉树算法的运行时间,还必须比较邻居搜索和碰撞预判的复杂程度.



〈·〉表示四进制的网格编号;网格编号下方的数字为颗粒的编号

图2 颗粒系统示意图

邻居搜索采用下式将颗粒映射到对应网格,再寻找网格的邻居

$$n=\frac{i}{2r} \tag{1}$$

式中:n 表示颗粒映射到的网格编号;$i=(x、y、z)$;$2r$ 表示映射网格的边长.采取下式进行碰撞预判

$$(xyz_i-xyz_j)^2<(D_1+D_2)^2/4 \tag{2}$$

式中:D、xyz 分别表示颗粒直径、颗粒坐标张量;下标 i、j 表示颗粒 i 和颗粒 j.两种计算所涉及的具体操作如表2所示,由于碰撞算法中可以将开方运算转化为平方运算,可以近似认为乘除、平方、开方运

算时间相同，且为加减操作的 5 倍[12]. 碰撞判断所需的时间是邻居搜索的 1.5 倍以上.

结合表 1 和表 2 对颗粒系统进行分析，可以判断 V 氏八叉树算法和多重八叉树算法的优劣. 同时，表 1 也是判断多重八叉树算法网格细化等级的重要指标. 表 1 列出了采用 3 种等级的网格划分时，两种算法的对比情况. 如果仅采用两种等级的网格划分，那么表 1 中小颗粒的数量为 0，中颗粒的数量为($B+A$)，大颗粒数量仍为 C，通过计算可以判断多重八叉树是采用 2 级还是 3 级网格划分.

对于大多数颗粒系统采用 3 级网格划分可以达到满意的效果，但当小颗粒的颗粒密度比较大时，才考虑是否进行更多等级的网格划分. 当采用 4 级划分时，对小颗粒再进行一次分级，分为小颗粒(数量为 A')和更小颗粒(数量为 $A-A'$)，它们所对应的网格的平均颗粒数量分别为 $A'/(16M)$和$(A-A')/(64M)$，通过计算可以判断采用 4 级划分是否合适.

3　数值实验

本文采用法向力-位移模型对多重八叉树搜索算法进行检验，并与相同条件下的 V 氏八叉树算法进行比较，数值模拟封闭方腔内颗粒的自由落体运动，大中小颗粒数的比例采用 1∶4∶16，大中小颗粒相互碰撞并与壁面碰撞.

表 1　邻居搜索次数及预判碰撞对数

	碰撞类型	中心颗粒数	邻居搜索次数	平均颗粒数量	碰撞对数
多重八叉树	小(小)	A	9	$A/(16M)$	$9A\times A/(16M)$
	小(中)*	A	9	$B/(4M)$	$9A\times B/(4M)$
	小(大)	A	9	C/M	$9A\times C/M$
	中(中)	B	9	$B/(4M)$	$9B\times B/(4M)$
	中(大)	B	9	C/M	$9B\times C/M$
	大(大)	C	9	C/M	$9B\times C/M$
合计		$A+B+C$	$9(3A+2B+C)$		$\frac{9}{16M}(A^2+4B^2+16C^2+4AB+16BC+16AC)$
V 氏八叉树		$A+B+C$	$9(A+B+C)$	$(A+B+C)/M$	$\frac{9}{16M}(16A^2+16B^2+16C^2+32AB+32BC+32AC)$
差值			$-9(2A+B)$		$\frac{9}{16M}(15A^2+12B^2+28AB+16BC+16AC)$

*：小(中)表示以小颗粒为中心，中等颗粒为预碰目标颗粒.

表 2　操作算法对比

	运算次数			
	加减	乘除	平方	开方
碰撞判断	6		3	1
邻居搜索	2	3		

本文计算中采用了 64、256、1 024、4 096、16 384共 5 种网格数划分，48、192、768、3 072 个等效颗粒，模拟了 3 种颗粒密度下运行时间随颗粒数的变化. 本文定义等效颗粒数为大中小颗粒面积折算成的小颗粒个数，定义颗粒数密度为等效颗粒数与总的基本网格数的比值. 在 InterP4-1.6 GHz、1 024 MB、window XP system 平台上，分析两种算法运行 20 000 个时间步长(步长为 10^{-6} s)所需的时间，如图 3 所示. 两种搜索算法的运行时间均随着颗粒数的增加而线性增加，不随颗粒密度的增加而急剧增加，但多重八叉树搜索算法的运行时间仅为 V 氏八叉树算法运行时间的 1/4 左右.

同时，由表 1 的分析可知邻居搜索的次数和预判碰撞对数决定于大中小颗粒数的比例. 当颗粒系统仅有大颗粒时，多重八叉树的运行时间与 V 氏八叉树的运行时间相同，而当小颗粒比重较大，颗粒系

统中仅存在少量大颗粒时，两种算法的运行时间相差将会很大，多重八叉树搜索算法更能显示出其时间上的优势. 例如，当大小颗粒数之比为 1∶1 000 时，运行 20 000 个时间步长，V 氏八叉树算法的运行时间为 2 241 s，多重八叉树算法仅为 24 s. 由此可以看出，多重八叉树算法可以适用于各种粒径分布的颗粒系统，特别是小颗粒比重大的颗粒系统.

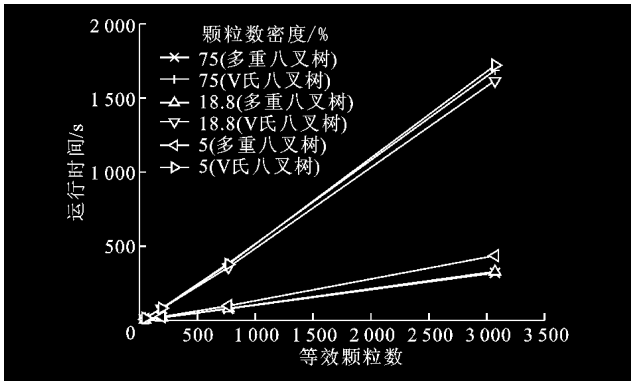


图 3 多重八叉树与 V 氏八叉树算法的运行时间对比

4 结 论

(1)利用线性八叉树的分块连续性、拓扑一致性以及编码与坐标的一一对应关系，大大简化了八叉树邻居的搜索算法，并实现了八叉树的多重搜索.

(2)采用多重八叉树搜索算法可以有效地解决大小颗粒碰撞问题. 算法的搜索计算量为 $O(N)$，具有良好的稳定性，不受颗粒形状、颗粒密度的影响，适用于各种常见分布的颗粒系统，特别是粒径概率分布相差较大的颗粒系统.

(3)多重八叉树搜索算法的计算速度受网格划分层数的影响，而网格划分层数主要受小颗粒数密度的影响，可以参考表 1 中的方法进行合适的网格划分.

(4)多重八叉树搜索算法有着非常强的工程背景，可以广泛地应用到颗粒流系统，如流化床、颗粒分离等工程领域.

参考文献：

[1] 徐泳，孙其诚，张凌，等. 颗粒离散元法研究进展[J]. 力学进展，2003，33(2)：251-260.
XU Yong, SUN Qicheng, ZAHGN Ling, et al. Advances in discrete element methods for particulate materials[J]. Advances in Mechanics, 2003, 33(2): 251-260.

[2] DEEN N G, van SINT ANNALAND M, van der HOEF M A, et al. Review of discrete particle modeling of fluidized beds [J]. Chemical Engineering Science, 2007, 62(18/20): 28-44.

[3] CUNDALL P A, STRACK O D L. A discrete numerical model for granular assembles[J]. Geotechnique, 1979, 29(1): 47-65.

[4] 王维，李佑楚. 颗粒流体两相流模型研究进展[J]. 化学进展，2000，12(2)：208-217.
WANG Wei, LI Youchu. Progress of the simulation of particle fluid two phase flow[J]. Progress in Chemistry, 2000, 12(2): 208-217.

[5] MUNJIZA A, ANDREWS K R F. NBS contact detection algorithm for bodies of similar size [J]. International Journal for Numerical Methods in Engineering, 1998, 43(4): 131-149.

[6] LUBACHEVSKY B D. How to simulate billiards and similar systems[J]. Journal of Computation Physics, 1991, 94(2): 255-283.

[7] BENJAMIN C S, STEVEN F Q, ANDREW H C C. Acceleration of the discrete element method (DEM) on a reconfigurable co-processor [J]. Computers and Structures, 2004, 82 (20/21): 1707-1718.

[8] BARAFF D. Interactive simulation of solid rigid bodies [J]. IEEE Computer Graphics and Applications, 1995, 15(33): 63-75.

[9] VEMURI B C, CHEN L, VU-QUOC L, et al. Efficient and accurate collision detection for granular flow simulation[J]. Graphical Models and Image Processing, 1998, 60(6): 403-422.

[10] CHEN H H, HUANG T S. A survey of construction and manipulation of octrees [J]. Computer Vision, Graphics, and Image Processing, 1988, 43(3): 409-431.

[11] VOROS J. A strategy for repetitive neighbor finding in octree representations[J]. Image and Vision Computing, 2000, 18(14): 1085-1091.

[12] WILLEN D C, KRANTZ J I. 8088 Assemble language programming: the IBM PC[M]. Indiana, USA: Howard W. Sanms&Co., Inc., 1983.

(编辑 荆树蓉 赵大良)