基于压缩的 AABB 树的碰撞检测算法

潘振宽　李建波
（青岛大学信息工程学院　青岛266071）

摘　要　用于碰撞检测的 AABB(axis-aligned bounding boxes)方法与其它基于包围盒的方法相比具有相交测试快速和适合变形体碰撞检测的特点。针对工程中大量存在的刚体和变形碰撞情形，本文基于压缩方法对 AABB 方法进行了改进。通过从空间的角度来对传统的 AABB 进行优化，从而节省了大量的存储空间，提高了变性体的碰撞检测效率。

关键词　碰撞检测，AABB，压缩算法，变形体

The Collision Detection Algorithm Based on Compressed AABB Trees

PAN Zhen-Kuan　LI Jian-Bo
(Information Engineering College, Qingdao University, Qingdao 266071)

Abstract　Comparing to other bounding volumes types, AABB used to solve collision detection has the characteristic that it can do quick intersection test and it is suitable for deformable objects. Aiming to solve the collision detection between rigid body and deformable object largely existing in the engineer, this paper proposes a compression algorithm to improve the AABB method. The traditional AABB method has been optimized from the space view and the consequence is that it saves a large amount of storing space and enhances the collision detection efficiency of deformable objects.

Keywords　Collision detection, AABB, Compression algorithm, Deformable object

1　引言

一个快速并且健壮的碰撞检测算法在很多领域比如虚拟现实、计算机游戏、计算机辅助设计都扮演着重要角色。在一个虚拟手术系统中，为了给训练者提供精确的力反馈以及沉浸感(Immersion)，碰撞检测模块必须能够把手术器械和人体组织碰撞的结果准确而又实时地传递给力反馈设备。近几年来，随着虚拟现实技术和分布式仿真技术的兴起，碰撞检测问题成为一个研究的热点[1]。

碰撞检测模块占去大量的存储空间和处理时间，往往导致系统处理的瓶颈，因此，有必要在时间或者是空间上进行优化以满足动态实时性的要求，本文也正是利用了 *AABB* 的一个特殊数学特性从存储空间上进行优化来加快处理碰撞检测。

随着计算机硬件处理能力的不断增强，对处理复杂场景以及细节渲染方面提出了更高的需求。其中的一点就表现在当虚拟场景中包含变形对象时，例如人体的软组织或布料。尽管在碰撞检测领域国内外学者做了大量的工作，但是这些科研工作大部分都集中于解决虚拟场景中刚体之间的碰撞检测。相对于刚体对象，变形对象表现在组成软体对象的顶点之间的相对位置发生变化。

对于刚体对象，大部分碰撞检测算法所依赖的数据结构都可以在预处理阶段处理完毕。但对于软体对象，适用于刚体对象的碰撞检测算法将不能直接应用。因为顶点之间的相对位移发生变化，因此用于碰撞检测算法的数据结构必须实时地重新构建或者更新。而重新构建整个数据结构的时间耗费在一个实时的仿真环境中是不能接受的。因此处理可变形体对象的碰撞检测算法的核心是怎样更新(refit)而不是重新构建(rebuild)数据结构。本文采用自底向上的方式来更新树结构。

碰撞检测算法主要分为两类，空间分解法(space decomposition)和层次包围盒法(hierarchical bounding volumes)。空间分解法比较典型的方法有 octrees[4]，BSP 树[5]，K-d 树[6]。层次包围盒法比较典型的包围盒类型有沿坐标轴的包围盒 AABB(axis-aligned bounding box)[6]，包围球(sphere)[7]，方向包围盒 OBB(oriented bounding box)[8]，k-DOPs (discrete orientation polytopes)[9]。本文采用层次包围盒法方法中最简单的包围盒类型 AABB。

对于可变形体的碰撞检测算法的研究，Van den Bergen[11]提出了的变形体的碰撞检测算法，并由此发布了称为 Solid 碰撞检测库。此算法基于的包围盒类型也是 *AABB*，并在每个时刻由叶子节点开始完成自底向上的更新。本文的更新算法基于 Solid 的更新算法，但是 Solid 方法并未对存储空间进行优化，因此本文可以看作是对 Solid 的改进。下面首先介绍一下 Solid 算法，然后详述本文所作的优化。

2　Solid 算法

Solid 算法采用的是层次包围盒法，其核心思想是利用体积略大而几何特性简单的包围盒将复杂几何对象包裹起来，在进行碰撞检测时，首先进行包围盒之间相交测试，如果包围盒不相交，则物体肯定不相交。只有包围盒相交时，才对其所包裹的对象做进一步相交测试。在构造碰撞物体的包围盒时，若引入树状层次结构，可快速删除不发生碰撞的元素，减少大量不必要的相交测试，从而提高碰撞检测效率。Solid 采用的

包围盒类型是 $AABB$，一个物体的 $AABB$ 是包围该物体并平行于坐标轴的最小的六面体。图1给出了一个简化了的二维空间的例子。

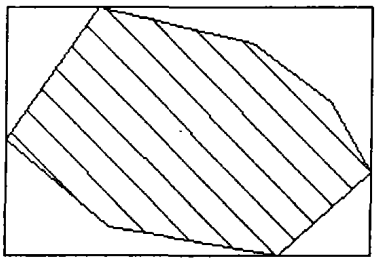

图1　物体的 $AABB$ 包围盒

计算一个物体的 $AABB$ 非常简单，只需求出构成物体的基本几何元素集合中每个元素的顶点坐标在 x,y,z 轴上的最大值和最小值即可。基于此特征，$AABB$ 具有一个非常好的数学特性，父节点的包围盒完全可以由其两个孩子节点的包围盒求出(Solid 采用的是二叉树结构)。Solid 算法也正是利用了这项特性来完成从叶子节点开始的自底向上的动态更新，本文所提出的压缩算法也是基于此特性。下面给出其数学证明。

设父节点的基本几何元素集合 $F_E=\{E_1,E_2,\cdots,E_n\}$，其左右孩子节点的基本几何元素集合分别为 $L_E=\{E_1,E_2,\cdots,E_m\}(m<n)$，$R_E=\{E_{m+1},\cdots,E_n\}$。其中满足 $L_E\cup R_E=\Phi$，$L_E\cap R_E=F_E$。父节点的 $AABB$ 为 A_F，其左右孩子节点的 $AABB$ 分别为 A_L，A_R。设求基本几何元素集合 E 的 $AABB$ 包围盒的操作为 $A(E)$，即需要证明 $A(A(L_E)\cap A(R_E))=A(F_E)$。由于求 $AABB$ 包围盒在三个坐标轴上的操作相同，不失一般性，以下只证明 X 轴。设 L_E 集合在 X 轴上投影的最小值为 $X_{min(L)}$，最大值为 $X_{max(L)}$。R_E 集合在 X 轴上投影的最小值为 $X_{min(R)}$，最大值为 $X_{max(R)}$。设 F_E 集合在 X 轴上投影的最小值为 $X_{min(F)}$，最大值为 $X_{max(F)}$。由于 $L_E\cup R_E=\Phi$，$L_E\cap R_E=F_E$，因此 $X_{min(F)}=\min\{X_{min(L)},X_{min(R)}\}$。同理 $X_{max(F)}=\max\{X_{max(L)},X_{max(R)}\}$。由于一个物体的 $AABB$ 是用其基本几何元素在 x,y,z 轴上的最大值最小值来存储的，因此可以认为 $A(L_E)=\{X_{min(L)},X_{max(L)}\}$。$A(R_E)=\{X_{min(R)},X_{max(R)}\}$，$A(A(L_E)\cap A(R_E))$ 操作就是在集合 $\{X_{min(L)},X_{max(L)},X_{min(R)},X_{max(R)}\}$，中分别求出最小值和最大值，根据先前的叙述很明显就 $X_{min(F)}$ 和 $X_{max(F)}$，即 $A(F_E)$。图2给出了其图形表示。

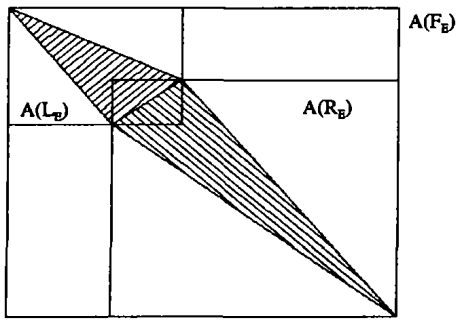

图2　包含左右子节点的 $AABB$ 包围盒

3　压缩算法

在采用压缩算法之前，我们必须首先构建物体 $AABB$ 树(采用的是二叉树)，构建 $AABB$ 树的基本算法为：

(1)求出物体所包含的整个基本几何元素集合的 $AABB$。

(2)沿着当前的 $AABB$ 的最长轴，按三角形的质心划分为两个集合。如果一个子集为空集，则创建任意两个大小近似相等的子集。

(3)如果子集只包含一个三角形，则创建一个叶节点。

(4)否则，对每个子集重复的执行以上过程。

在一般的应用中，为了指定包围盒的范围，四个字节的 float 型浮点数就能满足要求。因此每个节点需要6个 sizeof(float)=24个字节。一个二叉树排序 n 个基本的三角形面片恰好需要 $2n-1$ 个节点。因此当三角形面片的数量超过 2^{15} 时，所需的节点数就会大于 $2^{16}-1$。此时为了增加检索的效率，需要给每个节点添加两个四字节的分别指向左右子树的索引。当节点是一个叶子节点时，则可以利用根节点不会被任何一个内部节点索引的性质，可以令叶子节点的一个索引为1(根节点的编号为1)，另一个索引为三角形的编号。据此性质则可以迅速地定位叶子节点。此时每个节点需要存储24+8=32个字节。

前面已经证明父节点的 $AABB$ 完全可以由其左右两个子节点的 $AABB$ 求出。这就意味着父节点和其两个子节点之间存在大量的数据冗余，两个子节点最多有6个范围值不同于父节点。因此可以把冗余的值只保存一份，这可以通过在父节点中增加一个字节的标志位来完成。

在此结构中，用一个位来表示哪一个范围属于哪一个子节点。用"1"表示这个范围值属于左子节点，"0"表示这个范围值属于右子节点。这一共需六位，剩余的两位表示节点是否为叶子节点，"1"代表是而"0"则恰好相反。约定在存储结构中，$AABB$ 盒是按照 x,y,z 并按最小值最大值依次排列。即表示为 $(x_{min},x_{max},y_{min},y_{max},z_{min},z_{max})$。以图2为例说明(因为是二维的，因此在 z 轴上设定随机值，下面的数据只反映了左右子节点的相对大小)：

$A(L_E)=(-11.2,2.3,3.2,7.8,5.3,9.8)$，$A(R_E)=(1.7,10.9,-9.6,4.9,1.9,8.0)$

很明显，此时 $A(F_E)=(-11.2,10.9,-9.6,7.8,1.9,9.8)$。此时，父节点的存储结构见表1。

表1　父节点的存储结构

1	0	0	1	1	0	1	1
标志位(一个字节)							
范围值(4×6=24个字节)							
左子节点或三角形索引(4个字节)				右子节点或三角形索引(4个字节)			

从父节点的标志位可以很容易识别它的范围值是取自左右子节点中的哪一个。比如表1中的标志位的第四位为1，第四位代表 y_{max}，这就代表 y_{max} 取自左子节点，因此左子节点中的 $y_{max}=7.8$ 就可以只在父节点中存储一次。可以看出父节点中的标志位的前六位中0和1的个数是相同的。这并不是偶然现象，以图2中的二维空间为例，父节点的 $x_{min},x_{max},y_{min},y_{max}$ 分别来自左、右、右、左子节点。但有两种特殊情况见图3，图4。

在图3中左子节点 ABC 的 $AABB$ 盒为 $FGBE$，右子节点 ACD 的 $AABB$ 包围盒为 $JAHI$，父节点的 $AABB$ 包围盒为 $JGBK$。从图3中可以明显地看出左子节点的 $AABB$ 包围盒 $FGBE$ 的 y_{max} 的值等于右子节点 $AABB$ 包围盒 $JAHI$ 的 y_{max}，因此父节点的 y_{max} 既可以是取自左子节点又可以取自右子节点。此时父节点的标志位的前三位为1，0，1，第四位既可

以是0又可以是1。但此时为了保持1和0的个数平衡，第四位取0。这样做是为了保持左右子节点保存形式的统一。

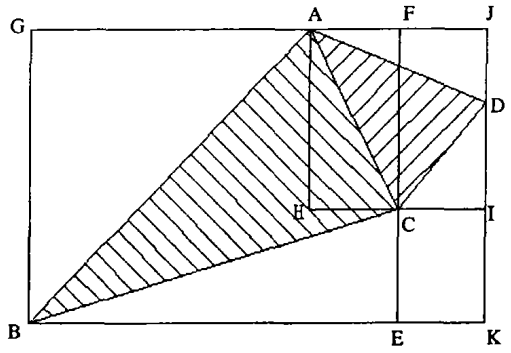

图3　左右子节点有一个相同值的 *AABB*

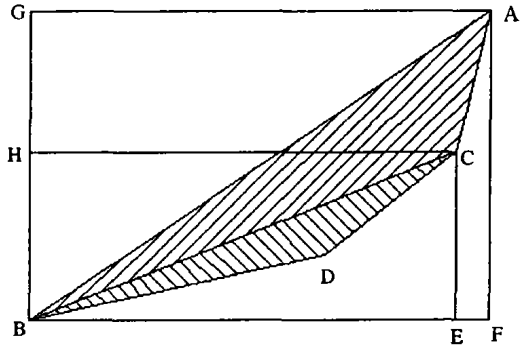

图4　冗余度最大的 *AABB*

图4是最特殊的情况，此时左子节点 *ABC* 的 *AABB* 包围盒 *AGBF* 完全包含右子节点 *CBD* 的 *AABB* 包围盒 *CHBE*，因此父节点的 *AABB* 包围盒就是其左孩子节点的 *AABB* 包围盒 *AGBF*。此时也是节点值冗余度最大的情况，但同图3一样，为了保持左右子孩子节点存储结构的统一，父节点的标志位的前四位为0，0，1，1。

实际上表1所示的节点结构为树的根节点的存储结构，此时每个内部节点的范围值就可以只存储3个浮点数(依次按 x,y,z 轴存储)共需12个字节，再加上1个字节的标志位和8个字节的索引，因此一共需21个字节，远远小于未经优化时的32个字节。在一个17层的完全二叉树数结构中，共有$2^{17}-1$个节点，根节点增添的一个字节可以忽略不计，大约能节省$2^{17}\times(32-21)$个字节，这将是很大的内存节省。

总结　本文通过对 solid 碰撞检测算法进行空间上的优化以通过节省内存空间来加快碰撞检测的执行结果。本文是借助 *AABB* 一个特殊的数学属性来完成此目的。实际上，表1所示的根节点的结构是一个指向物体整个三角面片集合浮点范围的单独数据类型。因此其他内部节点的具体范围值都需要参考根节点的范围值来取得自身的范围值，这就增加了间接性，需额外消耗一定的时间。但是因为每个节点都有索引，并且节省的大量内存空间可以大大缩短数据装入内存的运行时间，因此可以弥补因查找而额外消耗的时间。在用 T. Möller[12]的方法进行三角形之间的碰撞检测时，当随机三角形数目小于等于10k时，由于查找的间接性，因此和传统的方法所消耗的时间相差无几。但当三角形数目大于10k时，压缩算法的优越性就体现了出来。当用30k的随机三角形时，压缩算法比传统算法快10%左右。因此本算法较适合于几何特性比较复杂的物体之间的碰撞检测。

参考文献

1　阎丽霞. 虚拟手术关键技术研究:[浙江大学博士论文]. 2001

2　Hubbard P M. Collision Detection for Interactive Graphics:[PHD Thesis]. Department of Computer Science, Brown University, March 1995

3　Cohen J D, Lin M C, Manocha D, Ponamgi M. I-COLLIDE an interactive and exact collision detection system for large-scale environments. Symposium on Interactive 3D graphics, Monterey, CA USA, 1995. 189～196

4　Moore M, Wilhelms J. Collision Detection and Response for Computer Animation. ACM Computer Graphics (Proc. of SIGGRAPH '88), 1988, 22(4): 289～298

5　Naylor B, Amatodes J A, Thibault W. Merging BSP Trees Yields Polyhedral Set Operation. ACM Computer Graphics (Proc. of Siggraph'90), 1990, 24(4): 115～124

6　Held M, Klosowski J T, Mitchell J S B. Evaluation of Collision Detection Methods for Virtual Reality Fly-Throughs. In: Proc. Seventh Canadian Conf. on Computational Geometry, 1995, 2: 205～210

7　Hubbard P M. Approximating Polyhedra with Spheres for Time-Critical Collision Detection. ACM Transaction on Graphics, 1996, 15(3)

8　Gottschalk S, Lin M C, Manocha D. OBBTrees: A Hierarchical Structure for Rapid Interference Detection. ACM Computer Graphics (Proc. of SIGGRAPH' 96), 1996. 171～180

9　Klosowski J T, et al. Efficient Collision Detection Using Bounding Volumes Hierarchical of k-DOPs. IEEE Transaction on Visualization and Computer Graphics, 1998, 4(1): 21～36

10　Smith A, Kitamura Y, Takemura H, Kishino F. A Simple and Efficient Method for Accurate Collision Detection Among Deformable Polyhedral Objects in Arbitrary Motion. In: Proc. of the IEEE Virtual Reality Annual Intl. Symposium, 1995. 136～145

11　van den Bergen G. Efficient Collision Detection of Complex Deformable Models using AABB Trees. Journal of Graphics Tools, 1997, 2(4): 1～14

12　Möller T. A fast triangle-triangle intersection test. Journal of Graphics Tools, 1997, 2(2): 25～30