ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.19, Supplement, December 2008, pp.190−201

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于混合包围体的 OpenMP 并行化碰撞检测算法*

赵 伟[1,2+], 谭睿璞[2], 李文辉[1]

[1](吉林大学 计算机科学与技术学院,吉林 长春 130012)

[2](长春工业大学 计算机科学与工程学院,吉林 长春 130012)

Parallel Collision Detection Algorithm Based on Mixed BVH and OpenMP

ZHAO Wei[1,2+], TAN Rui-Pu[2], LI Wen-Hui[1]

[1](College of Computer Science & Technology, Jilin University, Changchun 130012, China)

[2](School of Computer Science & Engineering, Changchun University of Technology, Changchun 130012, China)

+ Corresponding author: E-mail: prince1205@163.com

Zhao W, Tan RP, Li WH. Parallel collision detection algorithm based on mixed BVH and OpenMP. *Journal of Software*, 2008,19(Suppl.):190−201. http://www.jos.org.cn/1000-9825/19/s190.htm

Abstract: Concerning the requirements of real-time and accurate collision detection in interactive system, a shared memory parallel collision detection algorithm is proposed. Firstly, the algorithm incorporates the merits of both AABB bounding box and bounding spheres to construct a hybrid bounding representation of arbitrary non-convex polyhedra (S-AABB) for attaining speed, and then uses OpenMP parallel programming model to traversal the built hybrid bounding volume hierarchy, so further accelerates the collision detection. At last, The experimental results have shown that the algorithm is advantageous over other current typical collision detection algorithms such as I-COLLIDE regarding efficiency and accuracy, so can meets the real-time and accurate requirements in complex interactive virtual environment. At the same time, comparing with some parallel collision detection algorithms which include MPI using multi-process, Pipelining using multi-thread and multi-process, this paper has shown that our parallel algorithm based OpenMP is superior in terms of time efficiency, resource consumption and stability.

Key words: collision detection; hybrid bounding volume hierarchy; balance tree; parallel technology; OpenMP

摘 要: 针对交互式系统中碰撞检测实时性、精确性的要求,提出了一种共享存储系统的并行碰撞检测算法.利用 AABB 包围盒较好的紧密性和包围球计算简单的优点来构建物体的混合包围体层次(S-AABB),快速排除不相交的物体以加速算法,利用 OpenMP 并行模型来并行遍历混合包围体层次,进一步加速碰撞检测算法.实验结果表明,与现有经典的 I-COLLIDE 等算法相比,该算法在效率、精确性方面具有明显优势,能够满足交互式复杂虚拟环境的实时性和精确性的要求.同时,还与已经提出的 MPI 及 Pipelining 等并行算法进行比较,从时间效率和资源消耗两个方面说明本文基于 OpenMP 算法的优点.

关键词: 碰撞检测;混合包围体层次;平衡树;并行技术;OpenMP

* Supported by the National Natural Science Foundation of China under Grant Nos.60573182, 69883004 (国家自然科学基金)
Received 2008-05-03; Accepted 2008-11-14

碰撞检测在计算机图形学、虚拟现实、计算机游戏、动画、计算机辅助设计、机器人及虚拟制造等领域中均是经典而关键的问题[2],多年来一直受到较多的关注.目前,大多数虚拟对象的几何模型都是由成千上万个基本几何元素组成.由于虚拟环境的几何复杂性的增加,使得碰撞检测的计算复杂性大大提高,复杂场景的交互消耗大量的计算机资源,因此碰撞检测往往成为虚拟环境中的一个瓶颈.

近年来研究人员对碰撞检测进行了广泛而深入地研究,提出了大量高效的新算法.大体上可分为两类:空间分解法和层次包围盒方法[3].尤其是层次包围盒(hierarchical bounding volumes),已经作为公认的比较好的碰撞检测手段被广泛应用.主要包围盒有:沿坐标轴的包围盒(AABB),包围球(Sphere),沿任意方向包围盒 OBB 等,每种包围盒都有其优缺点.本文针对包围盒的特点,提出了一种结合包围球和 AABB 包围盒优点的混合包围体层次(S-AABB)的新算法,加速了碰撞检测的过程.

随着并行机的高速发展,并行技术给研究者带来了前所未有的机遇和挑战.并行碰撞检测在计算几何和机器人控制等领域得到了广泛的应用[4].文献[4]提出了同等大小体素构成模型的并行处理算法,但对于过于简单或是结构复杂的模型较难处理.文献[5,6]提出了基于凸多面体剖分的并行碰撞检测算法,但其实验模型为凸体有一定的局限性.文献[7]提出了一种基于 MPI 的并行碰撞检测算法,虽然算法效率得到了一定的提高,但是物体的八叉树表示相当费时,算法进程间通讯需要花费一定的时间.文献[8]提出了一种基于流水线的并行碰撞检测算法,虽然算法效率有很大提高,但任务进程数 p 对算法性能有很大影响,难以找到最佳值.

与以上方法不同,作者提出了基于混合包围体的 OpenMP 并行化碰撞检测算法.采用分治策略为每个任意形状的物体建立平衡二叉树,然后并行遍历包围盒树,这样极大提高碰撞检测速度,并具有通用性能可以在单处理机和多处理机上运行.

1 理论基础

1.1 基本碰撞检测算法

碰撞检测算法归根到底都要进行基本几何元素的相交检测,而基本几何元素大多由三角形构成,因此高效的三角形和三角形相交检测对于提高碰撞检测算法的整体效率至关重要.文献[9]提出了矢量判别法和标量判别法两种三角形和三角形相交检测算法,并通过实验证明了矢量判别法比标量判别法快 7%,本文以此算法设计了基本碰撞检测算法,主要用于在包围盒粗判以后精确判别包围盒中的物体是否相交.

1.2 混合包围盒层次

在粗略相交检测阶段,本文采用了包围球和包围盒树来加速算法.包围球具有简单,且旋转平移后易于更新的特点,缺点就是紧密性差,需要检测相交的包围盒对数多,而 AABB 包围盒与包围球相比,它对物体包围更加紧密,构造和相交测试较简单.因此,我们利用 AABB 包围盒与包围球各自的优点提出了混合包围体层次(S-AABB),本文采用 S-AABB 混合包围体树进行粗略阶段的相交检测.

1.3 混合模型中不同类型包围盒的相交检测

设 AABB 包围盒内点的坐标为 (x,y,z),包围盒的边长为 H,包围盒的中心坐标为 (X_s,Y_s,Z_s),假设中心点位于坐标原点,则 AABB 包围盒的约束方程为

$$\{(x,y,z)\,|\,0<x,y,z<H\} \tag{1}$$

同样,设包围球内点的坐标为 (x,y,z),包围球的半径为 R_S,球心坐标为 (X_0,Y_0,Z_0),则包围球的约束方程为

$$\left\{(x,y,z)\,\middle|\,(x-X_0)^2+(y-Y_0)^2+(z-Z_0)^2<R_s^2\right\} \tag{2}$$

假设 AABB 包围盒位于坐标原点 $(0,0,0)$,令 $\varphi=tg^{-1}\left(\frac{Y_0}{X_0}\right)$,其中 X_0 不为零,则 AABB 包围盒与包围球之间的距离为

$$dist = \sqrt{X_0^2 + Y_0^2 + Z_0^2} - (R_s + H)\bullet\varphi \tag{3}$$

设 AABB 包围盒与包围球中心点的连线与 Z 坐标轴的夹角为 θ,则有

当 Y_0 >0 且 X_0 >0 时, $\theta = \varphi$;

当 Y_0 >0 且 X_0 <0 时, $\theta = \pi - \varphi$;

当 Y_0 <0 且 X_0 >0 时, $\theta = -\varphi$;

当 Y_0 <0 且 X_0 <0 时, $\theta = \pi + \varphi$;

因此,仅需比较 θ•$dist$ 和 $|Z_0|$ 即可得到两包围盒的相交情况;若 θ•$dist < |Z_0|$,则 AABB 包围盒和包围球相交;否则不相交.

1.4 平衡树技术与分治策略[10]

1.4.1 平衡树

对于二叉树而言,我们称一棵包围盒树为完全的当且仅当包围盒树中的每一个叶结点对应于 S 的一个单元素子集,即只包含一个基本几何元素时,则称一棵包围盒树为完全的.由以上描述可知,包围盒树最多有 $2^n - 1$ 个结点,其中有 n 个叶结点并且要求两子包围盒内的多边形个数大致相等;一棵完全的包围盒树的高度至少为 $\log 5n$,此时称为树是平衡的.

1.4.2 分治策略

并行程序设计中最基本的一种技术即是分治策略.分治策略的特点是将一个问题分为与原来的较大问题有相同形式的子问题,进一步分为较小的问题通常是通过递归方法,直到不需要再分解为止.接着就完成这些简单的任务并把结果合并,然后按更大的任务继续合并,直到获得最后的结果.当每次分割产生两部分时,递归分治模型就会形成一棵二叉树;分治策略也可应用于将一个任务在每一步分为 M(M>2)个部分的情况,则称为 M 路分治,通常有二叉树(M=2)和四叉树(M=4).

2 算法描述及实现

2.1 混合包围体层次的构建

对于一个给定的基本几何元素集,混合包围体层次的构建分为如下 4 个步骤:

(1) 首先为物体的每个基本元素构建小包围球,我们称之为基本球(underlying sphere).主要有两个用途:a.有助于快速划分包围体,b.在进行精确的相交测试前,它可用于预检测.

(2) 建立根节点.就是要建立整个物体的 AABB 包围盒、整个物体的包围球和物体中所有多边形的基本球.

(3) 利用与局部坐标轴垂直的平面,将上述包围体划分成两个子包围体以形成根节点的两个子节点.子节点具有和根节点一样的 3 个特征:AABB 包围盒、包围球和节点中所有多边形的基本球.

为使得左、右子包围体中的多边形个数大致相等,保证混合包围盒体树的平衡,采用了一个惩罚函数 $f(p)$ 定义如下:

$$f(p) = |n_l - n_r| + \lambda n_c \tag{4}$$

其中,p 为分割面, n_l, n_r, n_c 分别为左包围体内、右包围体内和横跨基本球的个数. λ 是 n_c 的一个影响系数. 理想的分割面就是在所有轴向上惩罚函数值最小的分割面:

$$\min\left\{\begin{array}{l}\min\left\{f(p)\middle|p_x \perp x\_axis, p_x \in [x_{\min}, x_{\max}]\right\}\\ \min\left\{f(p)\middle|p_y \perp y\_axis, p_y \in [y_{\min}, y_{\max}]\right\}\\ \min\left\{f(p)\middle|p_z \perp z\_axis, p_z \in [z_{\min}, z_{\max}]\right\}\end{array}\right\} \tag{5}$$

(4) 对第(3)步中得到的两个子节点分别递归地执行上述包围体的分割过程得到最终的混合包围盒树.

适当的递归终止条件我们用简单的线性判别函数:

$$p(x) = A^T * B \tag{6}$$

其中, $A=(a_1,a_2,a_3,...,a_n)$, $B=(b_1,b_2,b_3,...,b_n)$. $b_1,b_2,b_3,...,b_n$ 为递归终止的各种因素,常取为:包围盒树要求的高度,叶子节点中要求所含基本几何元素的最多数, $a_1,a_2,a_3,...,a_n$ 分别为 $b_1,b_2,b_3,...,b_n$ 对应的加权值.

图 1 是根据上述算法步骤构建的平衡二叉树,每个节点中第 1 行的两个数字表示包含在对应节点中的基本球数目(等于相应多边形个数)和对应节点的包围球的半径;第 2 行的 3 个数字表示的是对应节点的 AABB 包围盒的宽度、高度、深度.

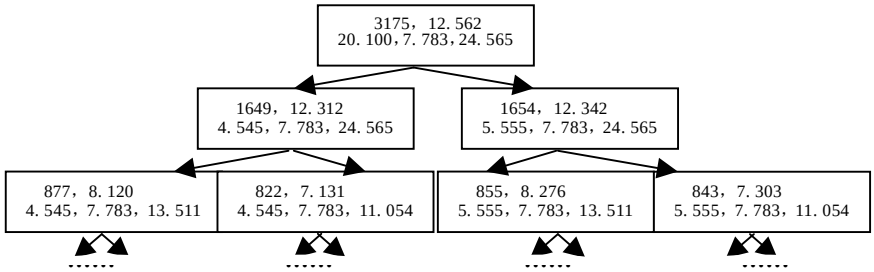

Fig.1 Representation of the hybrid bounding volume hierarchy

图 1 混合包围体层次的表示

2.2 混合包围体层次的遍历

物体 A 和 B 的混合包围体层次构建完成,我们就可以通过遍历它们的平衡树来进行它们之间的碰撞检测,在此,我们引入任务树的概念[8].将遍历两个物体的平衡包围盒树的过程定义成一棵任务树的遍历.任务树的构建如图 2 所示,其中 ai 表示物体 A 任务树中节点,bi 表示物体 B 任务树中的节点.

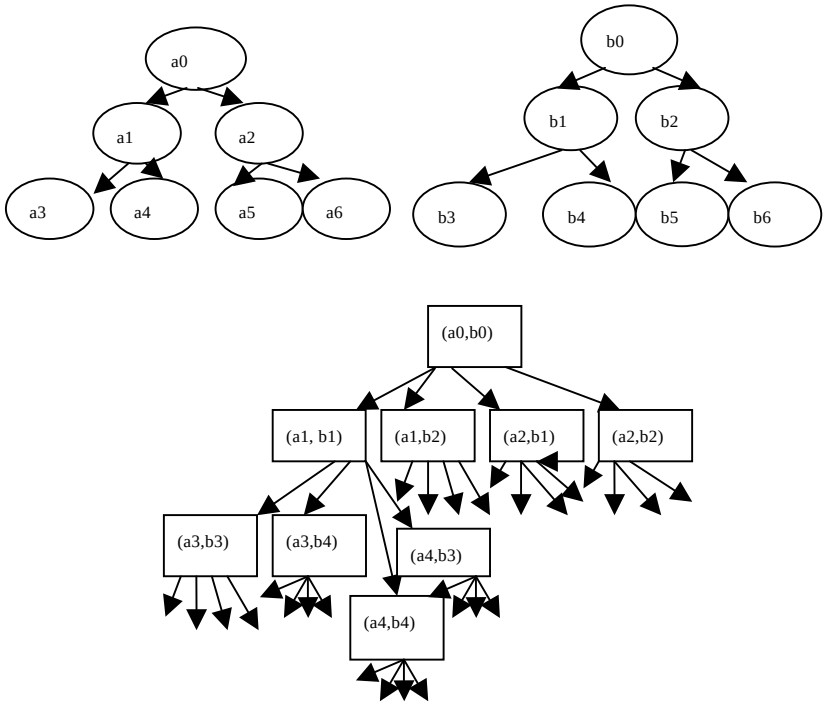

Fig.2 Simple work-tree

图 2 简单任务树

碰撞检测的过程从本质上来说是一个或树的遍历过程.一方面,如果碰撞发生在一些子任务中,那么物体 A

和 B 肯定发生了碰撞.另一方面,如果所有的子任务都没有发生碰撞,那么物体 A 和 B 肯定没有发生碰撞,因此一个任务的结果(TRUE 发生碰撞或者 FALSE 没有发生碰撞)可以通过将所有子任务碰撞结果作逻辑或运算.利用任务树,混合包围体层次的广度遍历过程的算法如下:

```
输入:两物体 A,B
输出:碰撞检测的结果 //TRUE: 相交, FALSE: 不相交
bool WFS-Traversal (RA,RB){
  建立 A,B 两物体的混合包围盒树 RA,RB;   //任务(RA,RB)作为根任务
  LIVESET<-任务节点 (RA,RB)   //将任务节点放入队列
  for i =1 to LIVESET.size
  { //循环执行队列中的所有节点
    while (LIVESET!=NULL){ //队列不为空
    (a1,b1)<- LIVESET;   //从队列中取出一任务节点
    SplitNode=true;   // 分裂变量为真,表示节点继续划分子节点
    if ( (a1,b1)节点的包围球不相交)
     SplitNide=FALSE;   //此任务节点不相交,分裂变量设为假
    else //包围球相交
      Flag1= (Sphere2Box VolumeRatio (a1)>= RatioThreshold(a1) );   // a1 进行包围盒相交的阈值
      Flag2= (Sphere2Box VolumeRatio (b1)>= RatioThreshold(b1) );   // b1 进行包围盒相交的阈值
      if (Flag1)   //   进行 a1 包围盒相交
       if (Flag2)   //   进行 b1 包围盒相交
        if ( (a1,b1) 任务节点的 AABB 包围盒不相交)
         SplitNode=FALSE;   //此任务节点不相交,分裂变量设为假
        endif
        else if ( (a1,b1) 任务节点的 a1 的 AABB 包围盒盒和 b1 包围球不相交)
          SplitNode=FALSE;    //此任务节点不相交,分裂变量设为假
         endif
        endif
        else
         if (Flag2) //进行 b1 节点 AABB 包围盒的相交检测
          if ((b1,a1) 任务节点的 b1 的 AABB 包围盒盒和 a1 包围球不相交)
          SplitNode=FALSE; //此任务节点不相交,分裂变量设为假
         endif
      endif // flag2
    endif // flag1
  endif   // (a1,b1) 任务节点的 b1 包围球和 a1 包围球不相交
  if (SplitNode=FALSE)   //不相交,不再划分子节点
  else //  相交,继续子节点的划分
     if (a1 是叶节点) //a1 是叶节点
      if ( b1 也是叶节点)) //b1 是叶节点
        CollisionDetectionFundamentalAlgorithm(a1,b1);   // 调用基本碰撞检测函数
          if (a1 和 b1 相交)
            return TRUE;   // A,B 发生碰撞
```

```
            endif
            else //b1 不是叶子节点,则产生 b1 的两个子节点
            LIVESET<- (a1,b1-> left); a1,b1-> left 进入队列
            LIVESET<- (a1,b1-> right); a1,b1-> right 进入队列
          endif
        else // a1 不是叶子节点
          if ( b1 是叶子节点))
            // a1 不是叶子节点,则产生 a1 的两个子节点
            LIVESET<- (a1-> left,b1);
            LIVESET<- (a1-> right,b1);
            else  // b1,a1 都不是叶子节点, 则产生 b1,a1 的 4 个子节点
              LIVESET<- (a1-> left,b1-> left);
              LIVESET<- (a1-> left,b1-> right);
              LIVESET<- (a1-> right,b1-> left);
              LIVESET<- (a1-> right,b1-> right);
          endif
        endif
      endif
    }end-while
  }end- for
 return FALSE; // A 和 B 不发生碰撞
}  //WFA-Traversal 遍历结束
```

其中,任务树的节点采用队列(LIVESET)存储.函数 RatioThreshold()返回一个阈值,来决定是否还要进行 AABB 包围盒的相交检测.

3 算法的 OpenMP 并行化

3.1 OpenMP相关知识[11,12]

OpenMP 是基于线程的并行编程模型; 是基于已有线程的共享编程模型,是一个外部编程模型,它能够使程序员完全控制并行化.

OpenMP 它使用 Fork-Join 并行执行模型.所有的 OpenMP 程序开始于一个单独的主线程(master thread),开始执行后,主线程会一直串行的执行,直到遇到第一个并行域才开始并行执行.执行过程如下:1. Fork:主线程创建一队并行的线程,然后,并行域中的代码在不同的线程中并行执行;2. Join:当主线程在并行域中执行完之后,他们会被同步中断,最后只有主线程在执行.

3.2 基本算法的并行化

本文基本串行算法队列中的节点可以并行执行,大大节省了时间,从整体上提高了算法的效率.OpenMP 并行化基本串行算法如下:

```
输入:两物体 A,B (RA,RB)
输出:碰撞检测的结果 //TRUE: 相交, FALSE: 不相交
bool Parallel-Traversal (RA,RB){
  LIVESET<-任务根节点(RA,RB) //将任务根节点放入队列
  while (LIVESET!=NULL)
```

```
{ //队列不为空
  (a1,b1)<- LIVESET;   //从任务队列中
  #pragma omp parallel for
  if (当前任务节点的包围体不相交)
    取消当前任务;
  else if (当前任务包含有一个叶子节点)
    CollisionDetectionFundamentalAlgorithm(); //在当前任务处调用基本碰撞检测函数
      if (结果为 TRUE)
        取消当前并行执行的所有其它任务;
        return TRUE;   // 发现碰撞
      endif
      else //  当前任务包含中间结点
        以同样的方式将当前任务的所有子节点放入队列 LIVESET;
    endif
   endif //
    主线程等待其他并行的任务完成
   # pragma omp critical
     StoreIntersectingTaskNodes();   //存储相交的任务节点信息
 } end-while
return FALSE;
}
```

4 实验结果及性能分析

本算法采用 C++语言在 SGIONYX2 工作站(有 4 个 R10000 处理器)上实现,并用 OpenMP 编程模式实现并行化,部分并行采用多线程共享同一存储地址来实现,此处理器同一进程中能同时执行的线程数最多为 64,取为 32.

实验环境设定了一个简单的动态场景,环境中有 31 个物体做随机运动,基本几何元素数(三角面片数)多于 80 000 个.为了说明本算法的优越性,我们做了如下 4 个实验.

实验 1. 时间复杂性分析

实验中将本文算法与其它经典算法进行了时间复杂性(n 表示物体的三角形个数)、帧频及运行 1000 步的时间测试,实验结果如表 1 及图 3(a)所示.

其中 BCDA(base collision detection algorithm)为基于三角形与三角形相交的基本碰撞检测算法;I-COLLIDE 算法[1]为采用了 AABB 包围盒的算法;SPHERE 为采用了包围球的算法;SCDA(serial collision detection algorithm)为采用混合包围盒的串行碰撞检测算法;PCDA(parallel collision detection algorithm)为本文的基于混合包围体的 OpenMP 并行化碰撞检测算法.

Table 1 Analysis of time complexity

表 1 时间复杂性分析

算法类型	算法时间复杂度	帧频 (frames/s)	运行 1 000 步的时间 (s)
BCDA	$O(n^2)$	3.19	5 109
I-COLLIDE	$O(n\log n)$	4.05	4 231
SPHERE	$O(n\log n)$	4.05	3 265
SCDA	$O(n\log n)$	7.05	185
PCDA	$O(n\log n)$	17.70	74

从结果可以看出,混合包围体算法与单独采用一种包围体的算法相比,碰撞检测的时间效率有了很大的提高.采用 PCDA 算法,帧频可达 17.70/s,是其他算法帧频的 3~6 倍,运行 1 000 步所用的时间仅为其他经典算法的 1/100～1/3.

实验 2. 算法并行化分析.

为说明并行计算的优点,我们将 BCDA、I-COLLIDE、包围球算法(SPHERE)、SCDA 用 OpenMP 并行化, 在上述实验环境下,实验结果如表 2 及图 3(b)所示.

Table 2 The analysis of parallelism

表 2 算法并行化分析

算法类型	串行、并行算法	运行 1 000 步的时间 (sec.)
BCDA	NP(Non-parallel)	5 109
	P(parallel)	2 335
I-COLLIDE	NP(Non-parallel)	4 231
	P(parallel)	1 666
SPHERE	NP(Non-parallel)	3 265
	P(parallel)	1 336
SCDA	NP(Non-parallel)	185
	P(parallel)	74

从结果可以看出,采用 OpenMP 编程模式并行化算法,在给定的上述运行环境下,每一种算法的运行时间都减少了 2~3 倍,其中本文提出的基于混合包围盒并行算法 PCDA 比其串行算法快 2.5 倍,进一步加快了碰撞检测过程.

实验 3. 性能与效率测试.

对并行算法的测试主要是测试加速比和并行效率,可以通过式(7)和式(8)完成:

$$S_{(p)}=\frac{t_s}{t_p} \tag{7}$$

其中,$S_{(p)}$表示加速比 t_s 表示使用单处理器系统执行算法的时间,t_p 表示使用具有 p 个处理器的多处理机执行所需的时间.

$$E=\frac{t_s}{t_p\times p}=S_{(p)}\times\frac{1}{p} \tag{8}$$

其中 E 表示并行效率, p 表示处理器的个数.并行效率可定义为加速比与CPU个数之间的比值.

在上述实验条件及环境下,我们测试了在不同数量的处理器下并行算法的加速比和并行效率.实验得出本文提出的并行算法的加速比 $S_{(p)}$=2.697,并行效率 E =0.674.实验结果如图 4(a)、图 4(b)所示.

从理论上讲,在多处理器下,并行效率可达约 60%,加速比可达 5.5.但由实验结果可知,并行效率和加速比很难得到这种理想值,主要原因在于实验环境,如果实验环境简单,碰撞检测的计算任务小,几个处理器就足够了,若处理器太多,则难以实现最佳负载平衡同时处理器也得不到充分利用.反之如果实验环境复杂,则需要处理器数增加.因此在一定的实验环境下,随着处理器个数的增加,算法的加速比在峰值后呈现下降趋势,并行效率也在下降.对于不是特别复杂的虚拟环境,采用多处理器 2~4 个就足够了.

并行化处理显著地提高了碰撞检测算法的速度,能够满足交互式虚拟环境实时性、精确性的要求.

实验 4. OpenMP 与其他并行算法比较.

实验中我们将本算法与已经提出的并行算法从时间效率和资源消耗两个方面进行了比较.实验结果如表 3 及图 5(a)、图 5(b)所示.

其中,MPI 为采用文献[7]中 MPI 的并行碰撞检测算法;OpenMP 为本文基于 OpenMP 的并行碰撞检测算法;Pipelining 为采用文献[8]中流水线和分治技术的并行碰撞检测算法(其中任务 p 取 6).

Table 3 the comparison of parallel algorithms

表 3 并行算法的比较

碰撞检测算法类型	时间复杂度	帧频 (s)	运行 1 000 步的时间 (s)
SCDA	$O(n\log n)$	7.05	185
MPI	$O(n\log n)$	12.6	106
OpenMP	$O(n\log n)$	17.70	74
Pipelining	$O(n/p\log n)$	51.6	49

从实验结果可以看出,在给定的上述运行环境下,从时间效率方面来说,本文基于 OpenMP 的并行碰撞检测算法的时间效率高于 MPI,但低于 Pipelining,因为 MPI 采用多进程,进程间需要花费通信时间,而 Pipelining 不仅在单进程中采用了多线程,同时整个任务还采用了多进程,但是 Pipelining 的任务进程数 p 对算法性能有很大影响,难以找到最佳值,因此算法稳定性低.

从资源消耗方面来说,本文基于 OpenMP 的并行碰撞检测算法采用多线程,比其他采用多进程的算法要节省内存空间.同时 OpenMP 具有可移植性,简单,可扩展性;灵活支持多线程,具有负载平衡的潜在能力;支持 Orphan Scope,使程序更具有模块化等优点.

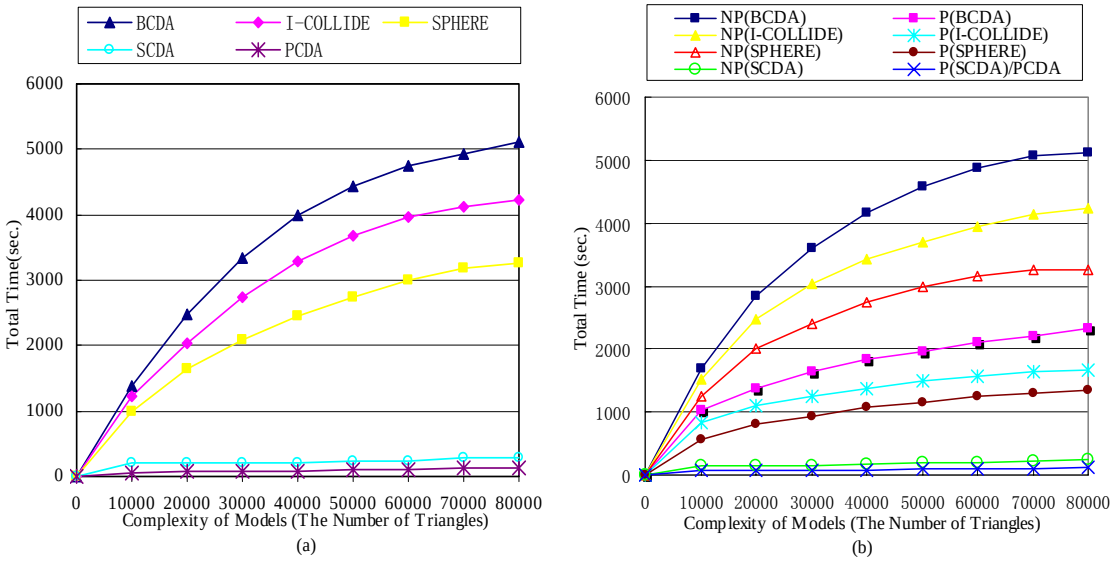

Fig.3 (a) Compare about experiment data of five algorithms such as BCDA, I-COLLIDE, SPHERE, SCDA and PCDA run 1000 steps; (b) Compare about experiment data of the four algorithms such as BCDA, I-COLLIDE, SPHERE, SCDA and their parallel algorithms run 1 000 steps.

图 3 (a) BCDA, I-COLLIDE, SPHERE, SCDA 与 PCDA 运行 1 000 步的实验数据的比较, (b) BCDA, I-COLLIDE, SPHERE, SCDA 及其并行化算法运行 1 000 步的实验数据的比较

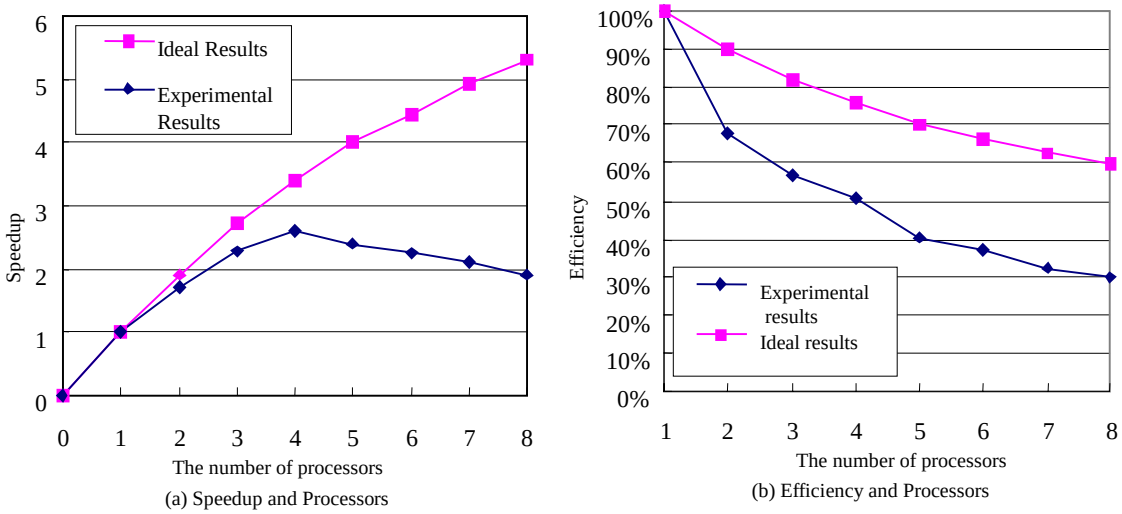

Fig.4 (a) The relationship between speedup and the number of processors, (b) The relationship between parallel efficiency and the number of processors

图4 (a) 加速比与处理器个数的关系,(b) 并行效率与处理器个数的关系

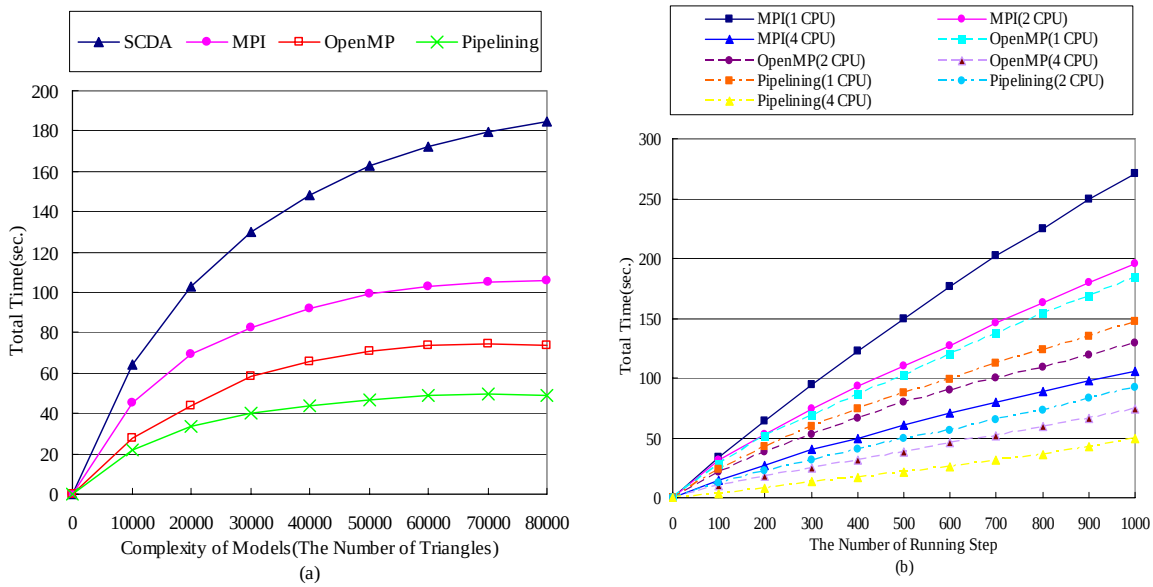

Fig.5 (a) Compare about experiment data of the parallel algorithms such as MPI, OpenMP and Pipelining run 1 000 steps. (b) Compare about experiment data of the parallel algorithms such as MPI, OpenMP and Pipelining with 1,2 and 4 CPU

图5 (a) MPI,OpenMP与Pipelining 三种并行算法运行1000步的实验数据的比较.(b) MPI,OpenMP与Pipelining三种并行算法分别在具有1个,2个,4个处理器的工作站上运行1 000步的实验数据的比较.

5 总结与展望

本文提出了一种基于混合包围体的 OpenMP 并行化碰撞检测算法.该算法的特点主要表现在 4 个方面:

(1) 利用 AABB 包围盒和包围球的优点来构建混合包围体层次;在精确碰撞检测之前,通过球与球、球与包围盒、包围盒与包围盒的相交检测排除大量不相交的多边形.

(2) 算法用分治策略建构平衡包围体树,保证了并行的各子问题规模大致相等,从而能够充分利用多处理机的性能.

(3) 采用 OpenMP 并行化算法,用了多线程技术,缩短了碰撞检测时间.

(4) 算法适用于对动态、复杂、交互式的场景进行实时碰撞检测.

尽管如此,算法仍有一些有待改进的地方.今后的工作将主要集中在:

(1) 算法充分并行化.本文仅对碰撞检测算法中的主要部分作了并行处理,还有不少部分有着固有的并行性,如矩阵运算,包围盒之间的求交算法本身均可以实现并行;

(2) 构建平衡包围盒树时,可以采用 M 路分治算法充分并行化;

(3) 包围盒划分时线性判别函数中因素和加权值的适当选择;

(4) 采用并行虚拟机(PVM)技术,使算法在互联网络的 PC 机上实现并行;

(5) 还可采用新的并行技术来扩展算法,如 MPI、破对称技术等.

致谢 感谢国家自然科学基金项目的支持,感谢吉林大学各位教授的指导和帮助,同时也感谢对本文提出批评和建议的专家.

References:

[1] Cohen J, Lin M, Manocha D, Ponamgi M. I-COLLIDE: An interactive and exact collision detection system for large-scale environments. In: Proc. of the 1995 ACM Symp. on Interactive 3D Graphics Conf. Monterey: ACM, 1995. 189−196.

[2] Fan ZW, Wan HG, Gao SM. Streaming real time collision detection using programmable graphics hardware. Journal of Software, 2004,15(10):1505−1514 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1505.htm

[3] Zheng Y, Ning RX, Liu JH, Yao J. Research on fast collision detection method in virtual assembly environment. Journal of System Simulation, 2005,17(9):2167−2170 (in Chinese with English abstract).

[4] Lawbr OS, Kalé LV. A voxel-based paralel collision detection algorithm. In: Proc. of the 2002 Int'l Conf. on SUPERCOMPUTING Conf. New York: ACM, 2002. 285−293.

[5] Xue GT, Li C, You JY. Algorithm of parallel collision detection based on dividing a convex polyhedron to tetrahedrons. Journal of Shanghai Jiaotong University, 2004,38(8):1385−1388 (in Chinese with English abstract).

[6] Jin WH, Rao SR, Tang WQ, Liu SQ. A fast polygon convex decomposition algorithm based on point visibility. Journal of Computer Research and Development, 1999,36(12):1455−1460 (in Chinese with English abstract).

[7] Liu XP, Cao L. Parallel octree collision detection based on MPI. Journal of Computer-Aided Design & Computer Graphics, 2007, 19(2):184−187 (in Chinese with English abstract).

[8] Zhao W, He YS. Rapid algorithm for parallel collision detection. Journal of Jilin University (Engineering and Technology Edition) 2008,38(1):152−157 (in Chinese with English abstract).

[9] Xu Q, Lü XF, Ma DW. A survey of triangle and triangle intersection test. Computer Simulation, 2006,23(8):76−78 (in Chinese with English abstract).

[10] Fan ZW, Wan HG, Gao SM. A parallel algorithm for rapid collision detection. Journal of Systerm Simulation, 2000,12(5):548−552 (in Chinese with English abstract).

[11] Figueiredo M, Fernando T. An efficient parallel collision detection algorithm for virtual prototype environments. In: Proc. of the 10th Int'l Conf. on Parallel and Distributed Systems (ICPADS 2004). Washington: IEEE Computer Society, 2004. 249–256.

[12] Wilkinson B, Allen M, Wrote; Lu XD *et al*. Trans. Parallel Programming. 2nd ed., Beijing: China Machine Press, 2005. 189−193 (in Chinese).

附中文参考文献:

[2] 范昭炜,万华根,高曙明.基于流的实时碰撞检测算法.软件学报,2004,15(10):1505−1514. http://www.jos.org.cn/1000-9825/15/1505.htm

[3] 郑轶,宁汝新,刘检华,姚珺.虚拟装配环境下快速碰撞检测方法的研究.系统仿真学报,2005,17(9):2167−2170.

[5] 薛广涛,李超,尤晋元.基于凸多面体剖分的并行碰撞检测算法.上海交通大学学报,2004,38(8):1385−1388.

[6] 金文华,饶上荣,唐卫清,刘慎权.基于顶点可见性的凹多边形快速凸分解算法.计算机研究与发展,1999,36(12):1455−1460.

[7] 刘晓平,曹力.基于 MPI 的并行八叉树碰撞检测.计算机辅助设计与图形学学报,2007,19(2):184−187.

[8] 赵伟,何艳爽.一种基于并行的碰撞检测算法.吉林大学学报(工学版),2008,38(1):152−157.

[9] 许强,吕晓峰,马登武.三角形和三角形相交测试技术研究.计算机仿真,2006,23(8):76−78.

[10] 范昭炜,万根华,高曙明.基于并行的快速碰撞检测算法.系统仿真学报,2000,12(5):548−552.

[12] Wilkinson B, Allen M,著;陆鑫达,等,译.并行程序设计.第 2 版,北京:机械工业出版社,2005.189−193.

赵伟(1967－),男,吉林长春人,副教授,主要研究领域为数据库系统,虚拟现实技术.

谭睿璞(1982－),女,硕士生,主要研究领域为数据库系统,虚拟现实技术.

李文辉(1963－),博士,教授,博士生导师,主要研究领域为计算机图形学,虚拟现实技术.