

# 基于 OBB 包围盒碰撞检测算法的改进

刘超, 蒋夏军, 施慧彬

(南京航空航天大学 计算机科学与技术学院, 江苏 南京 211100)

**摘要:**碰撞检测问题是计算机仿真领域的基本问题之一,随着现在计算机硬件的快速发展,碰撞检测问题已经成为制约整个领域发展的瓶颈之一。在多种碰撞检测算法中,基于方向层次包围盒的算法是一类被广泛应用的算法,这类算法使用一种层次包围盒树的数据结构,树的叶子节点所包含的包围盒实际上是一个三维空间矩形,利用叶子节点中矩形包围盒的特点以及在矩形相交测试阶段计算得到的结果,新的算法对方向包围盒碰撞检测算法中的三角形相交测试算法进行了改进。在原算法中,测试两个三角形相交之前需要将待测的两个三角形转换到同一坐标系中,而新算法中用包围盒的坐标代替三角形的坐标则可以省去这一步。改进算法相比原算法减少了大量冗余的坐标变换操作。

**关键词:**碰撞检测;方向包围盒;三角形相交测试;层次包围盒;坐标变换

**中图分类号:**TP391.41

**文献标识码:**A

**文章编号:**1673-629X(2018)06-0043-06

**doi:**10.3969/j.issn.1673-629X.2018.06.010

## Improved Collision Detection Algorithm Based on Oriented Bounding Box

LIU Chao, JIANG Xia-jun, SHI Hui-bin

(School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211100, China)

**Abstract:** Collision detection is one of the basic problems in the field of computer simulation and has become one of the bottlenecks of restricting the development of the whole field with the rapid development of computer hardware. In a variety of collision detection algorithm, the one based on oriented bounding box has been widely used. It adopts a data structure of bounding volume hierarchy tree and each box in the leaf nodes is actually a rectangle. By using the characteristics of the triangle surrounded by the rectangle in the leaf node and the value calculated in the rectangle-rectangle intersection test phase, the new algorithm contains a better triangle-triangle intersection algorithm in the oriented bounding box collision detection algorithm. In the original algorithm, it is necessary to convert the two triangles into the same coordinate system before the two triangles to be tested, but this step can be omitted by using the coordinates of the bounding boxes to replaces the coordinates of the triangles. The new algorithm reduces a lot of redundant coordinate transformation operations compared to the original one.

**Key words:** collision detection; oriented bounding box; triangle-triangle intersection; hierarchical bounding volumes; coordinate transformation

## 0 引言

近年来,随着虚拟现实技术的快速发展,物体对象之间的碰撞检测已经成为众多学者研究的热点。为了满足虚拟环境的真实性,所有参与者都必须能实时进行交互,而实时进行交互的前提是有效解决碰撞检测问题<sup>[1-2]</sup>。在3D游戏、虚拟装配、机器人路径规划等众多领域,碰撞检测也有着重要的应用<sup>[3]</sup>。

基于层次包围盒的碰撞检测算法是一类应用广泛的碰撞检测算法,根据包围盒的种类不同,基于层次包围盒的碰撞检测算法主要有轴向包围盒(AABB)算法、球包围盒(sphere)算法、方向包围盒(OBB)算法、离散方向多面体(K-DOP)算法等<sup>[4-5]</sup>。此类算法中,每一个待测的模型均对应一个层次包围盒树,树的每个节点代表模型基本组成元素的一个子集,通过检测

收稿日期:2017-08-01

修回日期:2017-12-13

网络出版时间:2018-02-24

基金项目:江苏省自然科学基金(BK20140826)

作者简介:刘超(1992-),男,硕士,研究方向为计算机仿真与虚拟环境中的碰撞检测等;蒋夏军,讲师,CCF会员(E200028291M),研究方向为计算机仿真、数据库技术等;施慧彬,副教授,研究方向为计算机体系结构、可重构计算等。

网络出版地址: <http://cnki.net/kcms/detail/61.1450.TP.20180224.1525.092.html>

两个模型的包围盒树节点是否相交,可以判断模型之间相交或者分离,只有当树的叶子节点相交时,模型之间才相交。

算法的运行时间可以用式 1 表示<sup>[6]</sup>:

$$T = N_v * C_v + N_p * C_p \quad (1)$$

其中,  $T$  表示总时长;  $N_v$  表示参与测试的包围盒数量;  $C_v$  表示测试一对包围盒消耗的时间;  $N_p$  表示参与测试的模型基本组成元素;  $C_p$  表示测试一对基本几何元素消耗的时间。

文中主要讨论刚体模型间的碰撞检测,这些模型的基本组成元素为三角形,根据式 1 可以通过提高算法中三角形的相交测试的效率来提高整个算法的效率。针对方向包围盒碰撞检测算法,文中对其中三角形之间的相交测试算法进行了一些改进,并且设计了相关实验来验证改进算法的效率。

## 1 方向包围盒碰撞检测算法

在 Gottschalk<sup>[6]</sup> 于 1996 年实现的“RAPID”碰撞检测系统中,将方向包围盒应用到碰撞检测算法中。虽然 OBB 包围盒之间的相交测试比较复杂,但因为其良好的紧密性,在一些环境中基于 OBB 包围盒的碰撞检测算法依然有很高的效率。

### 1.1 构建方向包围盒

OBB 包围盒是一个任意方向的长方体,可以用一个中心点、一个三阶方向矩阵和三个  $1/2$  边长表示,其中三阶方向矩阵表示包围盒三条轴的方向。通过计算包围盒内的全部三角形顶点的协方差矩阵  $C$  以及矩阵  $C$  三个特征向量,可以得到 OBB 包围盒的三条轴的方向。

假设模型中包含  $n$  个三角形,第  $i$  个三角形的顶点分别用  $p_i, q_i, r_i$  表示,则可以用下面公式得到这  $n$  个三角形的均值  $u$ :

$$u = \frac{1}{3n} \sum_{i=1}^n (p_i + q_i + r_i) \quad (2)$$

然后由  $u$  得到协方差矩阵  $C$ :

$$C_{jk} = \frac{1}{3n} \sum_{i=1}^n (\vec{p_{ij}} \vec{p_{ik}} + \vec{q_{ij}} \vec{q_{ik}} + \vec{r_{ij}} \vec{r_{ik}}) \quad (3)$$

其中,  $1 \leq j, k \leq 3, \vec{p_i} = p_i - u, \vec{q_i} = q_i - u, \vec{r_i} = r_i - u$ , 下标  $j$  和  $k$  的值分别表示所采用的坐标分量(即  $x, y$  或  $z$ )。

通过式 3 得到的协方差矩阵  $C$  为一个  $3 \times 3$  的对称矩阵,将  $C$  的三个特征向量正规化之后得到的基即为 OBB 的三条轴的方向,最后计算 OBB 内的三角形的顶点在这三条轴上投影的最大值和最小值即可确定 OBB 的三条边长。

### 1.2 OBB 之间的相交测试

方向的任意性使得 OBB 包围盒能更紧密地包围模型,但同时也使得 OBB 之间的相交测试变得更复杂,一种常见的 OBB 包围盒之间的相交测试算法是基于分离轴理论(SAT)的算法。根据分离轴理论,若在三维空间中存在一条直线,任意两个凸多面体在这条直线上的投影分离,则这条直线是这两个凸多面体的分离轴。OBB 包围盒属于凸多面体,因此在任意两个 OBB 包围盒之间若存在一条分离轴,则可以确定它们之间分离。对于三维空间两个 OBB 包围盒,它们之间一共存在 15 条潜在的分离轴需要检测,这 15 条分离轴分别是两个包围盒的 6 条方向轴以及两个包围盒 3 条方向轴两两组合得到的 9 条轴,若它们在这 15 条轴上的投影都不分离,则这两个包围盒之间相交<sup>[7]</sup>。文献[8]提出了一种采用混合层次包围盒的算法,该算法在包围盒测试阶段并不需要检测方向包围盒的 15 条分离轴,而只需要检测其中的 5 条分离轴,大大减少了包围盒的测试时间。

虽然 OBB 包围盒之间的相交测试可以排除模型之间大量不相交的三角形,但在很多情况下仍需要对大量的三角形之间进行相交测试。

### 1.3 三角形相交测试算法

对于空间三角形之间的碰撞检测,存在多种算法,这些算法大致可以分为两类:标量判别型算法和矢量判别型算法<sup>[9-10]</sup>。前者是指通过准确计算来判断三角形之间相交情况的一类算法,这类算法中典型的有 Möller<sup>[11]</sup> 算法和 Tropp<sup>[12]</sup> 算法;后者是指通过一系列计算值的符号来判定两个三角形的位置关系,然后判断其相交情况的一类算法,例如 Guigue&&Deville<sup>[13]</sup> 算法。在文献[14]中,Wei Lingyu 提出了一种基于 Tropp 算法的改进算法,算法的核心思想是:首先判断三角形  $B$  与三角形  $A$  所在的平面是否相交,若相交则求出它们的交线,同时根据三角形  $A$  的两条边所在的直线将三角形  $A$  所在的平面分为四部分,最后根据交线在这四个部分的分布情况判断三角形  $A$  与三角形  $B$  是否相交。

在 OBB 包围盒碰撞检测算法中若使用上述几种算法来检测三角形之间是否相交,则这些算法的输入值均为两个三角形的六个顶点坐标,而在模型对象中三角形的顶点坐标是基于模型坐标系的,即用以上算法检测两个三角形之前,需根据两个模型在世界坐标系中的位移信息将两个三角形转换到同一坐标系中,大量的三角形进行坐标变换操作将会影响整个算法的效率。在 OBB 包围盒之间的相交测试中,OBB 包围盒之间也有类似的坐标转换操作,根据 OBB 层次包围盒树中叶子节点的 OBB 包围盒的特点,可以

使用位于同一坐标系中的 OBB 包围盒的信息来表示待测的三角形坐标,并将所得到的坐标带入上述三角形相交算法中进行测试。对比多种算法后,发现 Wei Lingyu 的算法更适合文中改进后的算法。

## 2 改进的三角形相交测试算法

在 Wei Lingyu 的算法中,三角形之间的测试大致可以分为三个阶段。第一阶段,检测三角形  $B$  和三角形  $A$  所在的平面是否相交,若相交则计算出相交的线段,下面简称三角形  $A$  所在的平面为平面  $A$ ;第二阶段,根据三角形  $A$  两条边所在的直线将平面  $A$  分为 4 部分,并根据交线在平面  $A$  的分布情况判断两个三角形是否分离;第三阶段,进一步分析第二阶段无法判断三角形之间分离的情况,检测交线与三角形  $A$  是否相交,若交线与三角形  $A$  相交则三角形  $A$  和  $B$  之间相交,反之三角形  $A$  和  $B$  之间分离。

### 2.1 计算相交线段

在 OBB 碰撞检测算法中,层次包围盒树的叶子节点中的每个三角形的包围盒实际上是一个三维空间矩形,如图 1 所示。这个矩形可以用中心点  $c$ ,矩形长  $l$ ,宽  $d$ ,以及三阶方向矩阵  $[x, y, z]$  来表示,向量  $x$  和向量  $y$  表示矩形两条边的方向,向量  $z$  垂直于矩形所在的平面。

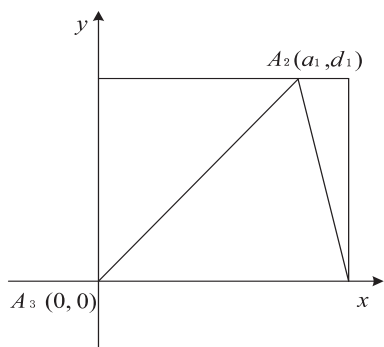


图1 叶子节点中的包围盒

在两个矩形(OBB)之间的相交测试中,已经计算出两个矩形变换到同一坐标系的旋转矩阵  $r[r_x, r_y, r_z]$  以及平移向量  $t[t_1, t_2, t_3]$ ,其中  $r_x = (r_{x_1}, r_{x_2}, r_{x_3})$ 。

结合图 1,三角形  $A$  的坐标和三角形  $B$  的坐标可以表示为:

$$\begin{cases} A_1 = (l_1, 0, 0), B_1 = r_x * l_2 + t \\ A_2 = (a_1, d_1, 0), B_2 = r_x * a_2 + r_y * d_2 + t \\ A_3 = (0, 0, 0), B_3 = t \end{cases} \quad (1)$$

其中,  $a_i$  是改进后需要额外计算的值,在计算三角形的包围矩形(OBB)过程中已经得到,即可以在预处理阶段计算出  $a_i$  的值。在接下来的步骤中,实际并不需要计算出两个三角形的 6 个坐标值。

假设三角形  $B$  的边和平面  $A$  之间存在交点,如图

2 所示。

其中,  $r_i = B_i - A_3, i = 1, 2, 3, e_j = A_j - A_3, j = 1, 2$ , 因此可以得到等式:

$$\alpha_1 e_1 + \alpha_2 e_2 = \beta_{ij} r_i + \beta_{ji} r_j \quad (2)$$

其中,  $\beta_{ij} + \beta_{ji} = 1, i < j$ 。等式右边表示交点位于三角形  $B$  的三条边所在的直线上,只有当  $0 \leq \beta_{ij}, \beta_{ji} \leq 1$  时交点位于三角形  $B$  的三条边上,等式的左边表示交点位于平面  $A$  中。令此交点为  $P$ , 向量  $m = P - A_3 = \beta_{ij} r_i + \beta_{ji} r_j$ , 则  $m \cdot (e_1 \times e_2) = 0$ , 结合条件  $\beta_{ij} + \beta_{ji} = 1$  可以解得:

$$\beta_{ij} = \frac{D_j}{D_j - D_i}, i \neq j, D_i \neq D_j \quad (3)$$

其中,  $D_i = r_i \cdot (e_1 \times e_2)$ , 根据上面三角形的坐标信息,容易得到:

$$\begin{cases} D_1 = (r_{x_3} \cdot l_2 + t_3) \cdot l_1 \cdot d_1 \\ D_2 = (r_{x_3} \cdot a_2 + r_{y_3} \cdot d_2 + t_3) \cdot l_1 \cdot d_1 \\ D_3 = t_3 \cdot l_1 \cdot d_1 \\ e_1 \times e_2 = (0, 0, l_1 \cdot d_1) \end{cases} \quad (4)$$

根据式 3, 只有当  $D_i$  和  $D_j$  有不同的符号(或者其中的一个值为 0)时能满足  $0 \leq \beta_{ij}, \beta_{ji} \leq 1$ , 即三角形  $B$  的边和平面  $A$  存在交点。如果  $D_1, D_2, D_3$  的符号相同则表示三角形  $B$  和平面  $A$  分离, 即三角形  $A$  和三角形  $B$  不相交; 如果  $D_1, D_2, D_3$  都为 0, 则表示三角形  $A$  和三角形  $B$  共面, 此时使用共面三角形相交测试算法进行检测, 但这种情况通常极少出现。当  $D_1, D_2, D_3$  的符号不全相同时, 三角形  $B$  与平面  $A$  存在两个交点, 设这两个交点为  $P_1, P_2, m_1 = P_1 - A_3, m_2 = P_2 - A_3$ 。

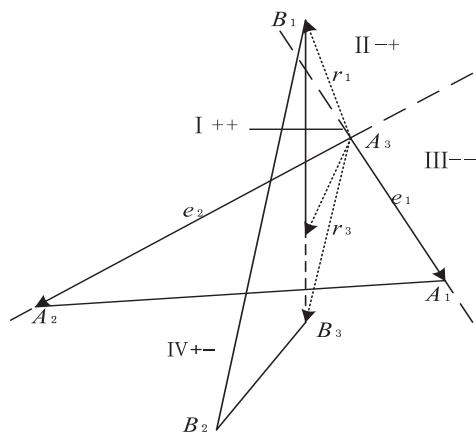


图2 三角形  $B$  与平面  $A$  相交

### 2.2 计算交点在平面中的位置

在图 2 中, 平面  $A$  被向量  $e_1$  向量  $e_2$  所在的直线分为 4 部分, 通过比较  $e_i \times m_i$  和  $e_1 \times e_2$  的方向, 可以确定  $P_i$  在平面  $A$  中的位置。例如当  $e_1 \times m_1$  与  $e_1 \times e_2$  方向相同且  $e_2 \times m_1$  与  $e_1 \times e_2$  方向相反时,  $P_1$  点位于平面  $A$  的第 IV 部分, 通过同样方法可以确定  $P_2$  的位置。比较  $e_i \times m_i$  与  $e_1 \times e_2$  的方向后, 一共可以得到 16 种结

果,如表 1 所示。

表 1 交点在平面  $A$  的分布情况

	(+,+)	(+,-)	(-,+)	(-,-)
(+,+)	N	Case <sub>1</sub>	N	Case <sub>2</sub>
(+,-)	Case <sub>1</sub>	Case <sub>3</sub>	Case <sub>4</sub>	Case <sub>1</sub>
(-,+)	N	Case <sub>4</sub>	N	N
(-,-)	Case <sub>2</sub>	Case <sub>1</sub>	N	N

符号“+”表示  $\mathbf{e}_i \times \mathbf{m}_i$  和  $\mathbf{e}_1 \times \mathbf{e}_2$  的方向相同,“-”表示方向相反。其中(+,+)表示  $P_i$  位于平面  $A$  的第 I 部分,(-,+)表示  $P_i$  位于平面  $A$  的第 II 部分,(-,-)表示  $P_i$  位于平面  $A$  的第 III 部分,(+,-)表示  $P_i$  位于平面  $A$  的第 IV 部分,“N”表示两个三角形分离,在 Case<sub>1</sub> ~ Case<sub>4</sub> 四种情况中,无法直接根据  $P_i$  的位置判断待测的三角形是否分离,需要进一步检测交线是否与三角形  $A$  相交,如图 3 所示。

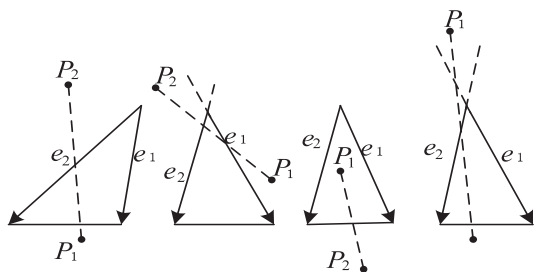


图 3 无法直接判断的 4 种情况

## 2.3 测试交线与三角形是否相交

若交线  $P_1P_2$  与三角形  $A$  相交,则至少满足下面其中一个条件:

- (1)  $P_1P_2$  与  $\mathbf{e}_1$  相交;
- (2)  $P_1P_2$  与  $\mathbf{e}_2$  相交;
- (3)  $P_1P_2$  与  $\mathbf{e}_1 - \mathbf{e}_2$  相交;
- (4)  $P_1P_2$  位于三角形  $A$  内部。

根据这四个条件,分别讨论 Case<sub>1</sub> ~ Case<sub>4</sub> 四种情况下  $P_1P_2$  与三角形  $A$  相交的情况。

Case<sub>1</sub>:根据图 3 中  $P_1$  与  $P_2$  的位置关系,条件 1、4 无法满足,因此只需检测  $P_1P_2$  是否满足条件 2 或 3。检测  $P_1P_2$  与  $\mathbf{e}_2$  是否相交等价于判断  $\mathbf{e}_1 \times \mathbf{e}_2$  与  $(\mathbf{m}_2 - \mathbf{e}_2) \times (\mathbf{m}_1 - \mathbf{e}_2)$  的方向是否相同,若两个向量的方向相同,则  $P_1P_2$  与  $\mathbf{e}_2$  相交,否则分离。而当  $P_1P_2$  与  $\mathbf{e}_2$  不相交且  $P_1$  在三角形  $A$  内时,  $P_1P_2$  一定与  $\mathbf{e}_1 - \mathbf{e}_2$  相交。因此在  $P_1P_2$  与  $\mathbf{e}_2$  不相交的情况下,  $P_1$  在三角形  $A$  内部与条件 3 等价,而判断  $P_1$  在三角形  $A$  内部只需确定向量  $\mathbf{e}_1 \times \mathbf{e}_2$  与向量  $(\mathbf{e}_2 - \mathbf{e}_1) \times (\mathbf{m}_1 - \mathbf{e}_1)$  同向。

Case<sub>2</sub>:若交线  $P_1P_2$  与三角形  $A$  相交,则  $P_1P_2$  一定至少与  $\mathbf{e}_1$  和  $\mathbf{e}_2$  中一个向量相交。因此只需检测  $P_1P_2$  是否满足条件 1 或条件 2 即可判断交线与三角形  $A$  是否相交。类似数据中的判断方法,  $P_1P_2$  与  $\mathbf{e}_i$  相交等

价  $(\mathbf{m}_2 - \mathbf{e}_i) \times (\mathbf{m}_1 - \mathbf{e}_i)$ 、 $\mathbf{e}_1 \times \mathbf{e}_2$ 、 $\mathbf{m}_1 \times \mathbf{m}_2$  三者的方向相同。

Case<sub>3</sub>:若  $P_1P_2$  与三角形  $A$  相交,则  $P_1P_2$  只可能满足条件 3 或者条件 4,即  $P_1P_2$  与  $\mathbf{e}_1 - \mathbf{e}_2$  相交或者  $P_1P_2$  位于三角形  $A$  内部。而在条件 3 或条件 4 中,  $P_1$  和  $P_2$  都至少有一点位于三角形  $A$  内部,因此在 Case<sub>3</sub> 中,  $P_1P_2$  与三角形  $A$  相交等价  $P_1$  和  $P_2$  中至少有一点存在三角形  $A$  内。当  $(\mathbf{e}_2 - \mathbf{e}_1) \times (\mathbf{m}_i - \mathbf{e}_1)$  与  $\mathbf{e}_1 \times \mathbf{e}_2$  方向相同时,  $P_i$  在三角形  $A$  内部。

Case<sub>4</sub>:与 Case<sub>2</sub> 类似,  $P_1P_2$  只可能与  $\mathbf{e}_1$  和  $\mathbf{e}_2$  相交,但在 Case<sub>4</sub> 中,首先比较  $\mathbf{m}_1 \times \mathbf{m}_2$  与  $\mathbf{e}_1 \times \mathbf{e}_2$  的方向,若两个向量的方向相同,则  $P_1P_2$  只可能与  $\mathbf{e}_2$  相交,方向相反则  $P_1P_2$  只可能与  $\mathbf{e}_1$  相交。

## 2.4 消除算法中的除法运算

在计算机中,一次除操作所消耗的时间远远高于加法、乘法、减法等运算的时间,因此合理地减少算法中的除法运算能提高整个算法的效率。下面讨论如何避免算法中的除法运算。

整个三角形相交测试的算法中,只有在求三角形  $B$  与平面  $A$  的交点  $P_1$  和  $P_2$  时需要进行除法运算:

$$\mathbf{m}_i = \beta_{ij} \mathbf{r}_i + \beta_{ji} \mathbf{r}_j, \beta_{ij} = \frac{D_j}{D_j - D_i}, i \neq j, D_i \neq D_j$$

令  $\mathbf{n}_i = (D_i - D_j)$ ,  $\mathbf{m}_i = D_i \mathbf{r}_j - D_j \mathbf{r}_i$ , 因为  $D_i$  与  $D_j$  异号,可以交换二者的值使  $D_i > D_j$ , 因此  $\mathbf{m}_i$  与  $\mathbf{n}_i$  的方向相同。故在 2.2 中用  $\mathbf{e}_i \times \mathbf{n}_i$  代替  $\mathbf{e}_i \times \mathbf{m}_i$  与  $\mathbf{e}_1 \times \mathbf{e}_2$  的方向进行比较将不会对结果产生影响。同样在 2.3 中,使用  $\mathbf{n}_i$  代替  $\mathbf{m}_i$ , 同时将参与运算但不包含  $\mathbf{m}_i$  的项乘以  $D_i - D_j$ , 例如  $(\mathbf{e}_2 - \mathbf{e}_1) \times (\mathbf{m}_1 - \mathbf{e}_1)$  与  $(\mathbf{e}_2 - \mathbf{e}_1) \times \mathbf{n}_1 - \mathbf{e}_2 \times \mathbf{e}_1 \cdot (D_i - D_j)$  的方向相同。

## 3 算法的基本运算次数

由于在仿真环境中不同模型的三角形坐标值不在同一个坐标系中,因此使用传统的三角形相交测试算法时必须根据模型的位移信息将待测的三角形转换到同一坐标系中。设  $A$  和  $B$  分别是模型 1 和模型 2 中的两个三角形,模型 1 和模型 2 在坐标系中的位移信息分别用四阶矩阵  $\mathbf{M}_1$  和  $\mathbf{M}_2$  表示,则将三角形  $B$  的三个点转移到三角形  $A$  所在的坐标系的公式为:

$$\mathbf{B}'_i = \mathbf{M}_{12} \cdot \mathbf{B}_i (i = 1, 2, 3) \quad (5)$$

其中,  $\mathbf{M}_{12} = \mathbf{M}_1^{-1} \mathbf{M}_2$ 。式 5 一共需要进行 27 次乘法运算和 27 次加法运算。

在 OBB 碰撞检测算法中用 OBB 包围盒的信息表示待测的三角形,并结合 Wei Lingyu 的三角形相交测试算法进行测试,因此在改进算法中可以避免上述 27 次乘法操作和 27 次加法操作。与 Wei Lingyu 的算法相比,文中对算法的改进主要集中在第一阶段,即计算



$D_1, D_2, D_3$  以及  $e_1 \times e_2$  的值。在 Wei Lingyu 的算法中, 计算以上结果一共需要进行 18 次减法操作、33 次加法操作和 42 次乘法操作, 其中包括将两个三角形转换到同一个坐标系所需要的 27 次乘法操作和 27 次加法操作<sup>[14]</sup>。而改进算法计算  $D_1, D_2, D_3$  以及  $e_1 \times e_2$  如下:

$$\begin{cases} D_1 = (r_{x_3} \cdot l_2 + t_3) \cdot l_1 \cdot d_1 \\ D_2 = (r_{x_3} \cdot a_2 + r_{y_3} \cdot d_2 + t_3) \cdot l_1 \cdot d_1 \\ D_3 = t_3 \cdot l_1 \cdot d_1 \\ e_1 \times e_2 = (0, 0, l_1 \cdot d_1) \end{cases}$$

其中,  $l_1 \cdot d_1$  的值只需要计算一次, 而  $a_2$  的值可以在 OBB 碰撞检测算法的预处理阶段计算得到, 所以改进后的算法计算  $D_1, D_2, D_3$  以及  $e_1 \times e_2$  的值只需要 6 次乘法操作和 3 次加法操作。与 Wei Lingyu 的算法相比少了 36 次乘法操作、30 次加法操作以及 18 次减法操作。改进算法最终一共需要 51 ~ 57 次基本运算操作, 这些运算包括加减法、乘法以及比较两个值的大小。而 Wei Lingyu 的算法一共需进行 135 ~ 141 次基本运算操作<sup>[14]</sup>, Guigue&&Deville 的算法需要进行 168 ~ 198 次基本运算操作<sup>[13]</sup>, Möller 的算法需要进行 178 ~ 200 次基本运算操作<sup>[11]</sup>。

4 实验结果

实验使用的硬件环境: i3-4150 CPU, 频率 3.5 GHz, 内存大小 4 G。软件环境: Window 7 操作系统, Microsoft visual studio 2010 开发平台。

一共设计了三组实验来验证改进算法的效率, 模型分别选择佛像模型、网格模型以及海绵模型, 其中海绵和佛像模型如图 4 所示。其中佛像模型由 125 000

个三角形组成, 网格和海绵模型则分别由 15 360 和 91 328 个三角形组成。分别验证了模型之间距离为 -0.02, -0.01, 0.00, 0.01 四种情况下改进的三角形相交算法的效率, 其中当距离为 -0.02, -0.01, 0.00 时, 模型之间相交, 距离为 0.01 时, 模型之间分离。

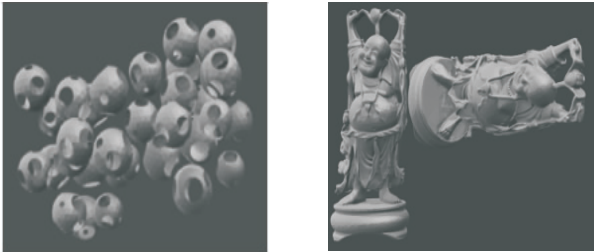


图 4 海绵、佛像模型

在仿真环境中, 决定模型之间的最短距离的值为两个模型中的三角形顶点坐标、模型坐标信息以及模型的旋转矩阵。对于以上四个给定距离, 在保持模型之间距离不变的情况下, 即两个模型之间的最短距离保持不变, 坐标值和旋转矩阵变化, 对每个距离下的三组模型都进行了 38 304 次碰撞测试。实验中每次碰撞测试所使用的模型坐标以及旋转矩阵均由文献 [15] 中的算法产生。

通过修改 RAPID 碰撞检测包实现了文中算法, 同时使用 Möller、Guigue&&Deville 以及 Wei Lingyu 的三角形相交测试算法来代替 RAPID 包中原有的基于 SAT 的三角形相交测试算法, 并用这些修改后的 RAPID 碰撞包检测所得到的结果与改进算法进行比较。

表 2 显示了模型之间的距离分别为 -0.02, -0.01, 0.00, 0.01 时, 38 304 次碰撞测试中三角形相交测试总的消耗时间。

表 2 三角形相交测试消耗的时间 s

模型	距离	SAT	Guigue & Deviller	Möller	Wei Lingyu	改进算法
佛像	-0.02	11.143 786	2.303 844	2.351 189	2.027 382	1.382 897
	-0.01	6.162 997	1.249 108	1.261 563	1.086 723	0.719 747
	0.00	3.521 038	0.764 895	0.772 085	0.663 928	0.424 650
	0.01	0.274 933	0.057 553	0.059 147	0.051 740	0.035 094
网格	-0.02	38.591 383	7.203 118	7.324 967	6.194 681	4.286 781
	-0.01	32.642 794	6.295 596	6.353 060	5.351 267	3.433 154
	0.00	11.773 122	2.192 790	2.282 244	1.962 606	1.325 326
	0.01	1.845 929 7	0.372 713	0.381 266	0.331 341	0.222 730
海绵	-0.02	8.240 766	1.442 644	1.467 393	1.253 657	0.858 669
	-0.01	7.012 493	1.262 974	1.283 515	1.094 068	0.735 854
	0.00	6.259 849	1.192 558	1.217 337	1.037 917	0.713 694
	0.01	0.597 230	0.111 545	0.112 909	0.096 073	0.067 222

根据表 2 可知, SAT 算法判断三角形相交的效率要显著低于其他算法, 因此判断凸多面体相交的分离轴算法实际上不适于三角形之间的相交测试。上述几种三角形相交测试算法中, Wei Lingyu 的算法的效率要比 Guigue&&Deviller 算法以及 Möller 算法的效率高 10% ~ 15% 左右。对 Wei Lingyu 的算法进行改进后, 改进算法的效率要比 Guigue&&Deviller 算法以及 Möller 算法高出 40% ~ 45% 左右, 比 Wei Lingyu 的算法高出 30% ~ 36% 左右。

## 5 结束语

基于传统的方向包围盒碰撞检测算法, 重复利用方向包围盒相交测试所得到的中间值, 对其中的三角形相交测试算法进行了一些改进, 同时在实验结果中发现基于分离轴理论的三角形相交测试算法效率很低, 据此推断基于分离轴理论的空间矩形的相交测试算法的效率可能类似于分离轴理论用于测试三角形。而在方向包围盒碰撞检测算法中, 叶子节点中的方向包围盒 OBB 之间的相交测试实际上是空间矩形之间的相交测试, 因此基于现存的高效的三角形相交测试算法, 可以在以后的工作中尝试找出一种新的空间矩形之间的相交测试算法来代替传统的基于分离轴理论的算法。

## 参考文献:

- [1] 宋城虎, 闵林, 朱琳, 等. 基于包围盒和空间分解的碰撞检测算法[J]. 计算机技术与发展, 2014, 24(1): 57-60.
- [2] 李苗. 实时碰撞检测算法分析与比较[J]. 计算机与现代化, 2011(6): 88-90.
- [3] 潘仁宇, 孙长乐, 熊伟, 等. 虚拟装配环境中碰撞检测算法的研究综述与展望[J]. 计算机科学, 2016, 43(11A): 136-139.

(上接第 42 页)

- [5] 薛丽霞, 李涛, 王佐成. 一种自适应的 Canny 边缘检测算法[J]. 计算机应用研究, 2010, 27(9): 3588-3590.
- [6] 贺强, 晏立. 基于 LOG 和 Canny 算子的边缘检测算法[J]. 计算机工程, 2011, 37(3): 210-212.
- [7] 谭艳, 王宇俊, 李飞龙, 等. 几种典型的图像边缘检测算法的分析比较[J]. 电脑知识与技术, 2012, 8(3): 1604-1608.
- [8] 贺赛先, 唐艳. 一种基于人类感知的边缘连接方法[J]. 红外技术, 2005, 27(4): 338-342.
- [9] 文贡坚, 王润生. 一种稳健的直线提取算法[J]. 软件学报, 2001, 12(11): 1660-1666.
- [10] 杨智明. 图的广度优先搜索遍历算法的分析与实现[J]. 农业网络信息, 2009(12): 136-137.

- [4] 赵伟, 曲慧雁. 基于云计算 Map-Reduce 模型的快速碰撞检测算法[J]. 吉林大学学报: 工学版, 2016, 46(2): 578-584.
- [5] 马登武, 叶文, 李瑛. 基于包围盒的碰撞检测算法综述[J]. 系统仿真学报, 2006, 18(4): 1058-1061.
- [6] GOTTSCHALK S, LIN M C, MANOCHA D. OBB-tree: a hierarchical structure for rapid interference detection[C]//Proceedings of the 23rd annual conference on computer graphics and interactive techniques. New York: ACM, 1996: 170-181.
- [7] ERICSON C. 实时碰撞检测算法技术[M]. 刘天慧, 译. 北京: 清华大学出版社, 2010.
- [8] CHANG J W, WANG Wenping, KIM M S. Efficient collision detection using a dual OBB-sphere bounding volume hierarchy[J]. Computer-Aided Design, 2010, 42(1): 50-57.
- [9] 许强, 吕晓峰, 马登武. 三角形和三角形相交测试技术研究[J]. 计算机仿真, 2006, 23(8): 76-78.
- [10] 邹益胜, 丁国富, 何邕, 等. 空间三角形快速相交检测算法[J]. 计算机应用研究, 2008, 25(10): 2906-2910.
- [11] MÖLLER T. A fast triangle-triangle intersection test[J]. Journal of Graphic Tools, 1997, 2(2): 25-30.
- [12] TROPP O, TAL A, SHIMSHONI I. A fast triangle to triangle intersection test for collision detection[J]. Computer Animation and Virtual Worlds, 2006, 17(5): 527-535.
- [13] GUIGUE P, DEVILLERS O. Fast and robust triangle-triangle overlap test using orientation predicates[J]. Journal of Graphics Tools, 2003, 8(1): 25-32.
- [14] WEI Lingyu. A faster triangle-to-triangle intersection test algorithm[J]. Computer Animation and Virtual Worlds, 2014, 25(5-6): 553-559.
- [15] TRENKEL S, WELLER R, ZACHMANN G. A benchmarking suite for static collision detection algorithms[C]//International conference in central European computer graphics, visualization & computer vision. [s. l.]: [s. n.], 2007: 265-270.

- [11] 曾俊. 图像边缘检测技术及其应用研究[D]. 武汉: 华中科技大学, 2012.
- [12] GLIANNAROU S, TANIA S. A novel framework for object recognition under severe occlusion[J]. Computational Intelligence, 2013, 410: 235-258.
- [13] GEORGIEVA P, MIHAYLOVA L, JAIN L C. Advances in intelligent signal processing and data mining: theory and applications[M]. Berlin: Springer, 2013.
- [14] 张研, 韩露. 用广度优先搜索算法实现路径搜索[J]. 电脑编程技巧与维护, 2012(19): 78-81.
- [15] SMEULDERS A W M, SANTINI S, WORRING M, et al. Content based image retrieval at the end of the early years[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(12): 1349-1380.