一种基于图像空间的碰撞检测算法

邹益胜[1]，丁国富[1]，周晓莉[2]，何　邕[1]，贾美薇[1]
（1.西南交通大学机械工程学院先进设计与制造技术研究所，成都 610031；2.西南交通大学智能控制与仿真中心，成都 610031）

摘　要：*根据光线与封闭物体间的相交特性，设计并开发了一种基于图像空间的碰撞检测算法。采用 VBO 技术提高图形渲染速度以提高算法的性能。*利用待测物体对的轴向包围盒(AABB)设置合理的视锥和视口，减少图形的绘制量，进一步提高算法性能。测试结果表明，该算法可以直接处理非凸体，处理复杂模型的碰撞检测问题实时性好、平稳性高，但是其性能受到分辨率的影响。最后讨论了将该算法扩展到多个物体间进行碰撞检测的实现策略。

关键词：图像空间；碰撞检测; VBO；非凸体
中图分类号：TP391.9　　**文献标识码：**A　　**文章编号：**1004-731X (2011) 05-0944-06

Collision Detection Algorithm Based on Image Space

ZOU Yi-sheng[1], *DING Guo-fu*[1], *ZHOU Xiao-li*[2], *HE Yong*[1], *JIA Mei-wei*[1]
(1.Advanced Design and Manufacturing Technology Institute, College of Mechanical Engineering, Southwest Jiaotong University, Chengdu 610031, China;
2. Intelligent Control and Simulation Research Center, Southwest Jiaotong University, Chengdu 610031, China)

Abstract: *According to the intersection characteristic between ray and closed object, an image based collision detection algorithm was designed and implemented. VBO technology was used to improve the performance of the algorithm by increasing rendering speed of the graphics.* Aligned axis bounding boxes (AABB) of the detected objects were used to set up reasonable frustum and viewport to reduce rendering data of the graphics, which improved the performance of the algorithm further. Test results show that the algorithm can deal with non-convex object directly and has high real-time and good stationarity in the collision detection between complex objects, but its performance is affected by resolution ratio. At last, the realization strategy of the collision detection among many objects by the algorithm was discussed.

Key words: image space; collision detection; VBO; non-convex object

引　言

碰撞检测是虚拟现实、计算机仿真和机器人运动规划等领域不可回避的问题之一。在计算机仿真和虚拟现实等领域，为了更真实地模拟现实场景，虚拟场景中的物体间如果发生干涉现象，要求快速准确地响应。而为了获得更逼真的仿真效果，对虚拟场景的描述越来越精细，虚拟场景的复杂度越来越大，对仿真的实时性要求也越来越高，因此研究快速的碰撞检测算法对增强计算机仿真和虚拟现实环境中的真实感有着重要的意义。由于基于图像空间的碰撞检测技术具有检测速度快、平稳性高、对场景复杂度的敏感性低及减轻 CPU 的计算负荷等方面的优越性，并且随着图形硬件卡加速技术的快速发展，其现已成为碰撞检测领域的研究热点之一。

基于图像空间的碰撞检测方法是先将几何物体投影到二维图像平面上，再对图像平面中在不同阶段投影所得到的颜色缓存、深度缓存、模板缓存和累积缓存等信息进行分析和比较，以此对物体间的干涉情况做出判断。

国内外很多学者对基于图像空间的碰撞检测算法进行了研究。Shinya[2]和 Rossignac[3]等人进行了开创性的研究工作。由于 Baciu[4]等人的突破性工作，使基于图像空间的碰撞检测算法可在常规图形工作站甚至普通 PC 机上运行，他们提出的 RECODE 算法充分利用深度缓存和模板缓存的组合功能，大大提高碰撞检测的效率，但其未能有效处理非凸物体。Hoff[6]、Kim[7]和 Govindaraju[8]等人的研究工作有效地结合了基于图形空间和基于图像空间的碰撞检测技术，在

收稿日期：2009-01-17　　**修回日期：**2009-04-24
基金项目：国家自然科学基金(50975240)，四川省青年基金(09ZQ026-003)
作者简介：邹益胜(1980-)，男，浙江文成人，博士，研究方向为 VP/VM，可视化；**丁国富**(1972-)，男，四川乐至人，博士，教授，博导，研究方向为 VP/VM/VR，先进制造技术，可视化；**周晓莉**(1981-)，女，四川成都人，硕士，助教，研究方向为系统仿真；**何邕**(1983-)，男，四川达州人，博士生，研究方向为 VP/VM，可视化；**贾美薇**(1985-)，女，河北唐山人，博士生，研究方向为 VP/VM，可视化。

算法的初期阶段和精确检测阶段分别利用这两种检测技术，利用不同检测技术的优点，获得更理想的算法效率和精度，同时平衡 CPU 与 GPU 的负载。

随着计算机图形硬件技术的发展，离屏渲染（渲染到纹理）技术的出现为基于图像空间的碰撞检测方法提供了新的舞台。通过该技术，可以保存帧与帧之间的缓存信息，基于图像空间的碰撞检测方法则可充分利用这些过程信息之间的差异性进行物体间干涉情况的判别。离屏渲染技术最新的解决方案是 FBO（Framebuffer Object）。FBO 相当于一个输出指向开关，通过与系统之间建立或解除绑定关系来控制所渲染的图像缓存信息的输出位置。当系统与 FBO 绑定时，渲染信息将输出到与 FBO 绑定的纹理上；而当系统与 FBO 解除绑定时，渲染信息则输出到系统默认的帧缓冲区。Han-Young Jang[9]等采用 FBO 进行碰撞检测算法的研究，他们先按顺序绘制所有待检测物体的正面，并渲染到不同纹理中，再逐个渲染物体的背面，通过分析每个物体背面像素点与不同纹理中的正面像素点之间的深度关系来判别物体间的干涉情况。

国内在基于图像空间的碰撞检测方面的主要研究成果包括：范昭炜[1]等采用 RECODE 算法原理，通过将物体表面凸分解及基于层次二叉树重组，并利用三角形带加速渲染，提高算法效率，可以处理非凸体。霍滨焱[10]、宋永军[11]等也进行了类似的研究工作。朱连章[12]等对 Govindaraju 等人提出的 CULLIDE 算法进行了性能提升研究。王季[13]等为待测物体的 OBB 表面生成深度纹理，以带深度纹理的包围盒替代物体的几何模型，并以此在 GPU 上进行深度比较来判断两个物体之间的干涉情况。

1 算法的设计与实现

1.1 算法基本原理

Han-Young Jang[9]等提出的算法在判别待检测物体间是否发生碰撞时利用了封闭物体的一种特性，即物体的正背面沿着视线方向依次出现。如果这种顺序被打乱，则表明该物体和其他物体发生碰撞。

受该判别思想的启发，对封闭物体这一特性进一步分析可以发现，从光源方向看，在封闭物体外的光源所发出的光线与该物体正面相交（定义为进入物体）、背面相交（定义为离开物体）的次数相同，而在封闭物体内的光源所发出的光线进、出该物体的次数相差 1，且离开的次数大于进入的次数，如图 1(a)所示。根据这一特性，假设从封闭物体外的光源向该物体投射光线且光线止于物体内一点，则光线进入该物体的次数比离开该物体的次数多 1 次，如图 1(b)所示。

1.2 算法的关键设计和实现

根据干涉的物理原理易知，若物体 A 和 B 发生干涉，则其中一物体上至少有一点落在另一物体内或其表面上。为

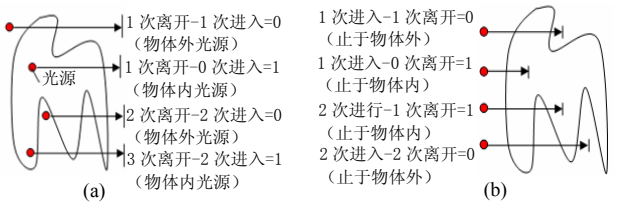

图 1　光线与封闭物体的相交特性

便于算法描述且不失一般性，假设物体 A 和 B 发生干涉，且物体 A 上的一个点 P 落在物体 B 中。

分别将点 P 和物体 B 投影到屏幕上，光源位于屏幕外观察者位置，光线沿垂直于屏幕的方向射入，止于点 P，利用深度测试描述光线与封闭物体的相交特性，利用模板缓存存储光线进入和离开物体的次数。

先将点 P 进行投影，图 2 所示象素点 1~3 为该点投影后相对物体 B 可能的深度关系。

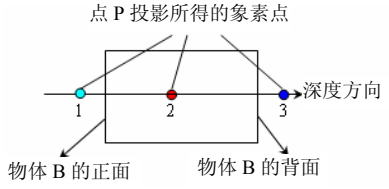

图 2　点 P 与物体 B 的投影关系

绘制物体 B 的正面，在点 P 投影所得的象素点上进行深度测试，如果新绘制点比点 P 更靠近光源，说明光线进入了物体。以图 2 为例，当点 P 投影后的象素深度位于 2、3 位置时，说明光线进入物体 B，而当象素深度位于 1 位置时，说明光线未进入物体。

绘制物体 B 的背面，同在点 P 投影所得的象素点上进行深度测试，如果新绘制点比点 P 更靠近光源，说明光线离开了物体。图 2 中点 P 投影后的象素深度位于 3 位置时，说明光线离开物体，位于 1、2 位置说明光线未离开物体。

通过上述分析可知：当点 P 投影后的象素深度位于 1 时，光线进入物体 0 次，离开 0 次，差值为 0 次；当其位于 2 时，光线进入物体 1 次，离开 0 次，差值为+1 次；当其位于 3 时，光线进入物体 1 次，离开 1 次，差值为 0 次。从图 2 中易知，当且仅当点 P 投影后的象素深度位于 2 时，两物体相交，这说明利用深度测试来描述光线与封闭物体的相交特性是可行的。

光线进、出物体的次数可用模板缓存记录。初始时所有象素点的模板缓存值都置为 0，当深度测试判别为光线进入物体时，该象素的模板缓存值加 1，当深度测试判别为光线离开物体时，该象素的模板缓存值减 1。若该象素点的最终模板缓存值等于 1，则可判定两物体发生干涉。

从上述分析可知，深度测试只发生在点 P 投影所得到的

象素点上。利用这一特性可以减少参与测试的象素量，提高算法性能，模板测试可以实现这种优化，具体方法如下：

绘制点 P 时，将其投影所得的象素的模板值置为 1。然后绘制物体 B 正面，打开模板测试，只对模板值为 1 的象素进行深度测试，光线进入物体时模板值加 1。另外，由算法基本原理可知，只有当光线先进入物体，两个物体才有可能相交，在光线进入物体的象素上的模板值大于 1。再绘制物体 B 背面，打开模板测试，只对模板值大于 1 的象素点进行深度测试，光线离开物体，模板值减 1。如果该象素点的模板值等于 2，则判定两物体发生干涉。

进一步对物体 A 上的一点 P 进行分析，该点可能是物体 A 的一个顶点，也有可能是位于物体 A 的一条边上，或是物体 A 的某个面上。对于前两种情况，可以统一认为点 P 位于物体 A 的一条边上。而对于第三种情况，可以交换物体 A、B 的角色，即该点位于物体 B 的一条边上。综上，可以通过判别一个物体的边投影后的像素点与另一个物体正、背面投影的深度关系来判定两物体的干涉情况。为便于算法描述且不失一般性，假设物体 A 对其边进行投影，物体 B 对其正、背面分别进行投影，算法实现如下。

(1) 打开模板缓存和深度缓存，模板测试和深度测试的条件都设置为 GL_ALWAYS，将物体 A 的边进行投影，如通过模板测试，则模板缓存值加 1，然后关闭深度缓存。图 3 中淡蓝色的象素区域为物体 A 投影后的结果，区域中的数字为该象素的模板缓存值。

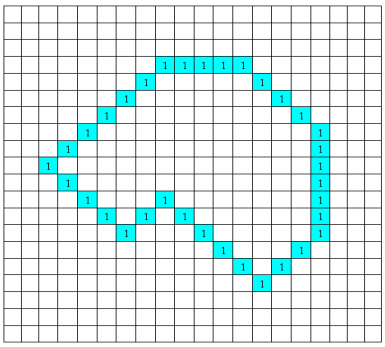

图 3 物体 A 投影

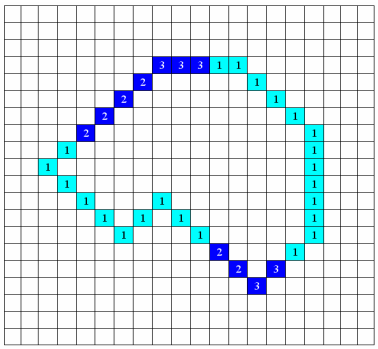

图 4 记录光线进入物体的次数

(2) 只渲染物体 B 的正面，设置模板测试条件，只对模板值为 1 的象素（图 3 中淡蓝色的象素区域）进行深度测试，统计在这些象素上光线进入物体 B 的次数。当深度测试判定光线进入物体时，则该象素点上的模板缓存值加 1。

(3) 只渲染物体 B 的背面，设置模板测试条件，只对模板值大于 1 的象素（图 4 中深蓝色的象素区域）进行深度测试，统计在这些象素上光线离开物体 B 的次数。当深度测试判定光线离开物体时，则该象素点上的模板缓存值减 1。

在图 5 中，如果视口内存在模板缓存值等于 2 的象素（红色区域），则判定两物体发生干涉。

需要注意的是，本文算法仅适用于封闭物体间的干涉检测，对于不封闭的物体，可能会出现误判现象。如在图 6 中，绘制物体 B 的正面时，该处象素的模板缓存值加 1，但绘制物体 B 的背面时，由于该处没有实体，因此模板缓存值不需要减 1，从而引起算法的误判。

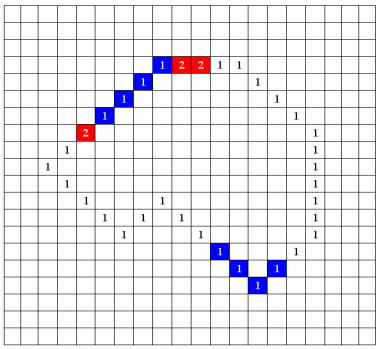

图 5 记录光线离开物体的次数

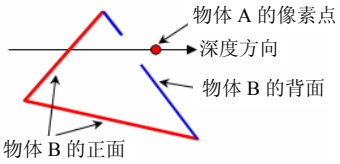

图 6 算法对于不封闭物体的误判

上述算法的伪代码实现如图 7 所示。

1.3 算法的优化

(1) 视口和视锥的优化设置

在两物体的干涉检测中，干涉往往发生在两物体局部较小的区域中。对干涉检测来说，非干涉区域属于不感兴趣区域。因此，通过图形投影视口和视锥体的优化设置，剔除非干涉区域的图形以减少图形的绘制量，提高算法的效率，同时放大干涉区域的视口，提高算法的精度。视口和视锥体的设置可以通过两个物体的 AABB 相交部分确定。

(2) 利用 VBO 技术提高渲染速度

基于图像空间的碰撞检测算法需要进行物体的绘制，因此提高渲染速度是一种有效的优化方法。方法之一是采用顶

点数组(Vertex Arrays)，通过减少函数的调用(如不需调用 glVertex 等）来提高渲染速度，但是顶点数组的数据存放在客户端，每次调用时都需要从客户端向服务器端发送数据，因此存在数据传输的瓶颈问题。方法之二是采用显示列表（Display List）。显示列表是一组存储在一起的 OpenGL 函数，当调用一个显示列表时，它所存储的函数就会按照顺序执行。显示列表在服务器端执行，但显示列表一旦编译完成，列表中的数据就不能改变。基于此，OpenGL 提供了一个名为 Vertex Buffer Object (VBO)的扩展，结合了顶点数组和显示列表的优点，克服了它们的不足，提高了渲染的速度。VBO 的工作原理和顶点数组类似，但 VBO 使用的是高速的显卡内存，而不是普通的系统 RAM 内存。它不仅仅降低了每帧的内存操作，而且减少了数据在显卡和 CPU 之间的传输，所以降低了渲染时间。同时，VBO 所引用的顶点数据是允许改变的，这在生成 VBO 的时候可以指定。VBO 扩展需要较新的图形硬件支持。

经过上述优化，本文算法的流程如图 8 所示。

1.4 算法的测试

将本文的算法命名为 Image_CD 算法。测试硬件条件：Intel(R) Core (TM)2 CPU E8400，3.0GHz，4GB 内存，NVIDIA Quadro FX 570 显卡，XP 系统。测试内容包括：不同渲染加速技术对算法性能的影响、不同场景规模对算法性能的影响、不同相交率下的算法平稳性测试和不同分辨率对算法的性能的影响。测试结果如图 9-12 所示。

```
bool ImagebasedCD(A, B) {
  //1 初始化
  SetupView(); //设置视口及投影模式
  glColorMask(GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE); //禁止写入颜色缓存
  glDepthMask(GL_TRUE); //允许写入深度缓存
  glClearDepth(1.0);   //设置深度缓存清空值为1.0
  glClearStencil(0.0);   //设置模板缓存清空值为0.0
  glClear(GL_DEPTH_BUFFER_BIT | GL_STENCIL_BUFFER_BIT); //清空缓存
  //2 绘制物体A的边
  glDepthFunc(GL_ALWAYS);  //总是通过深度测试
  glStencilFunc(GL_ALWAYS, 1, 1); //总是通过模板测试
  glStencilOp(GL_KEEP, GL_REPLACE, GL_REPLACE); //通过模板测试，模板值置1
  glEnable(GL_DEPTH_TEST);  //打开深度测试
  glEnable(GL_STENCIL_TEST); //打开模板测试
  glDisable(GL_CULL_FACE); //关闭面剔除模式
  SetDrawMode(GL_LINE); //设置以线框模式绘制图形
  Draw(A);  //绘制物体A
  //3 绘制物体B的正面
  glDepthMask(GL_FALSE); //禁止写深度缓存
  glDepthFunc(GL_LEQUAL); //小于等于当前值时通过测试，考虑了表面重合的情况
  glStencilFunc(GL_EQUAL, 1, 1); //只对模板值为1的像素进行操作
  glStencilOp(GL_KEEP, GL_KEEP, GL_INCR); //模板和深度测试都通过，模板值加1
  glEnable(GL_CULL_FACE); //开启面剔除模式
  glCullFace(GL_BACK); //背面剔除
  SetDrawMode(GL_FILL);//设置以填充模式绘制图形
  Draw(B);  //绘制物体B
  //4 绘制物体B的背面
  glDepthFunc(GL_LESS); //小于当前值时通过测试
  glStencilFunc(GL_GREATER, 1, 1); //只对模板值大于1的像素进行操作
  glStencilOp(GL_KEEP, GL_KEEP, GL_DECR); //模板和深度测试都通过，模板值减1
  glCullFace(GL_FRONT); //正面剔除
  Draw(B);  //绘制物体B
  //5 判断碰撞情况
  for(视口内的每个象素)
    if(模板缓存值==2)
      return true;   //发生碰撞
  return false;      //未发生碰撞
}
```

图 7　算法实现的伪代码

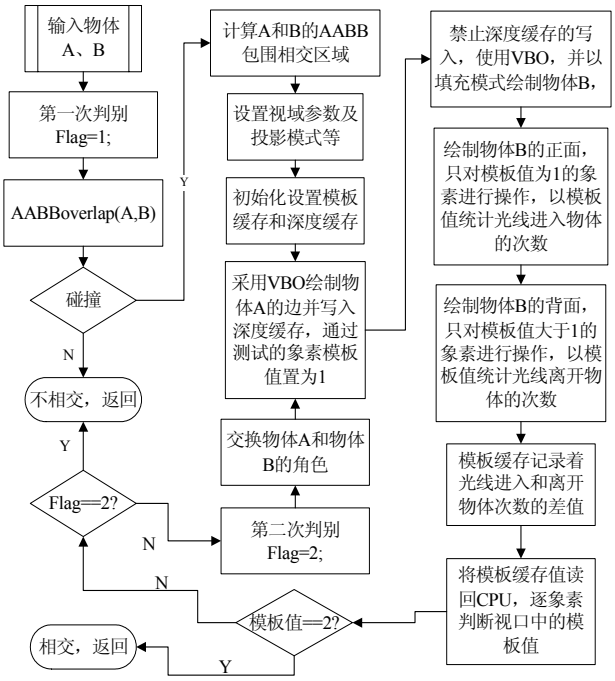

图 8　优化后的算法流程图

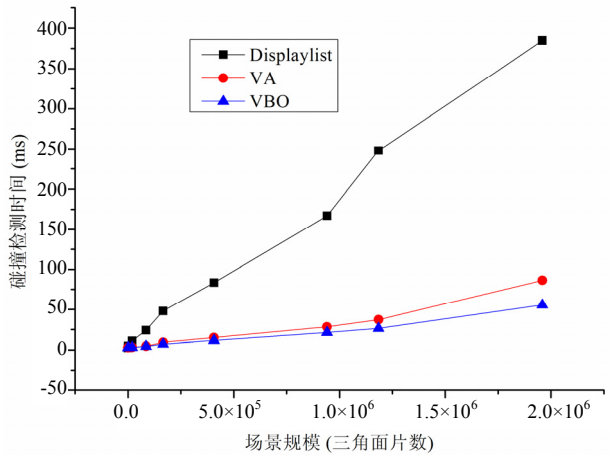

图 9　渲染加速方法对算法性能的影响测试结果

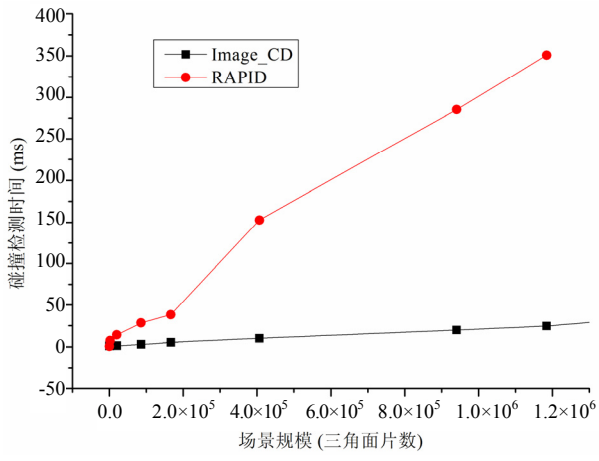

图 10　场景规模对算法性能的影响测试结果

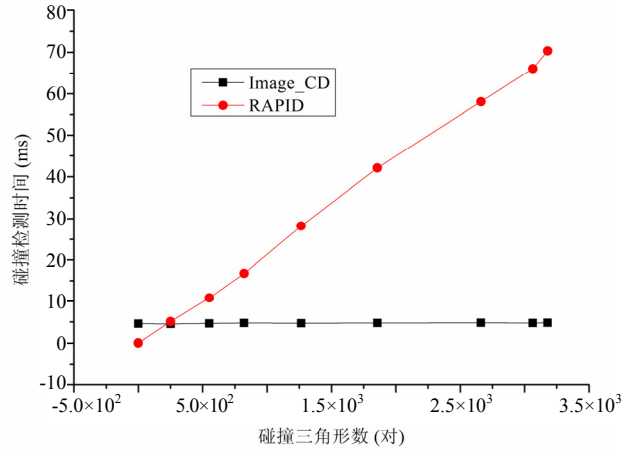

图 11　相交率对算法性能的影响测试结果

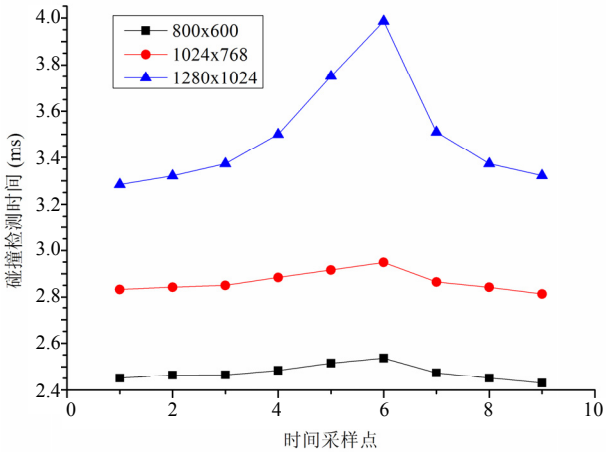

图 12　分辨率对算法性能的影响测试结果

2　讨论

该算法可以扩展到多个物体间的碰撞检测。分析该算法可以发现，算法第(3)步所操作的像素是在第(2)步的基础上完成的，即在第(2)步中通过模板和深度测试的像素点个数要大于或等于在第(3)步中通过测试的像素点个数。当第(2)步和第(3)步中通过测试的像素点个数相等时，说明光线进入物体的次数和离开物体的次数相同，则可判定两个物体不干涉。而如果第(2)步中通过测试的像素点个数大于第(3)步中通过测试的像素个数，说明光线进入物体的次数大于离开物体的次数，则可判定两物体干涉。这种判别方式可以利用硬件遮挡查询功能实现。NVIDIA 显卡支持的遮挡查询扩展 NV_occlusion_query 可以返回通过深度测试的像素个数。

对于一个由 n 个物体组成的场景，分别记为：O_1，O_2，…,O_n，借鉴 CULLIDE 算法[8]的两步实现思路，将该算法推广到多个物体间进行碰撞检测时可通过以下两个步骤实现：

第一步：清空模板缓存和深度缓存，关闭颜色缓存，按顺序渲染 O_1，O_2，…,O_n。

for(i = 1; i<=n;)

{

1) 开启深度缓存和模板缓存，设置模板和深度测试条件（GL_ALWAYS），以 GL_LINE 模式绘制 O_i，将新绘入的像素模板值置为 1；

2) i++；

3) 关闭深度缓存，设置模板和深度测试条件，以 GL_FILL 模式绘制 O_i，利用遮挡查询判断 O_i 是否与绘入深度缓存中的物体相交。如果相交，将 O_i 加入碰撞集；

}

第二步：清空模板缓存和深度缓存，关闭颜色缓存，根据第一步中的流程，按顺序渲染 O_n，O_{n-1}，…,O_1，同时利用遮挡查询判断新绘制的物体是否与已绘制的物体相交。如果相交，则将该物体加入碰撞集。

经过以上两步，便形成了整个场景的碰撞集，碰撞集中的每个物体都和其他至少一个物体发生过干涉，但具体在哪些物体对之间发生碰撞，还需要进一步分析。

3　结论

本文通过研究基于图像空间的碰撞检测算法的原理，根据光线与封闭物体间的相交特性，设计了一种基于图像空间的快速碰撞检测算法。采用 VBO 技术提高图形渲染速度以提高算法的性能。利用待测物体的轴向包围盒（AABB）设置合理的视锥和视口，减少图形的绘制量，达到进一步提高算法性能的目的。测试结果表明，基于图像空间的碰撞检测算法在处理复杂模型的碰撞检测问题时显示了良好的性能，受场景规模变化的影响较小。在同一场景下，该算法也表现出良好的平稳性，受模型相交程度的影响远小于基于 CPU 的碰撞检测算法。该算法受分辨率的影响，分辨率越高，检测精度越高，但检测速度越慢。

对于变形体，由于基于包围盒的碰撞检测算法需要更新包围盒，实时处理的效果较差，而基于图像空间的碰撞检测算法不需要预处理，因此更适用于变形体间的碰撞检测。但对于碰撞点的确定，有待进一步展开研究。另外，FBO 技术的发展，有助于基于图像空间的算法摆脱图形硬件的限制，也可以保存中间帧的图像信息，与 FBO 相关的算法应用研究也可以进一步展开。

参考文献：

[1] 范昭炜, 万华根, 高曙明. 基于图像的快速碰撞检测算法[J]. 计算机辅助设计与图形学学报, 2002, 14(9): 805-810.

[2] Shinya M, Forgue M. Interference detection through rasterization [J]. Journal of Visualization and Computer Animation (S1049-8907), 1991, 2(4): 131-134.

[3] Rossignac J, Megahed A, Schneider BO. Interactive inspection of solids: cross-section and interferences [J]. Computer Graphics (S0097-8930), 1992, 26(2): 353-360.

[4] Baciu G, Wong SKW, Sun H. RECODE: An image-based collision detection algorithm [J]. Journal of Visualization and Computer Animation (S1049-8907), 1999, 10(4): 181-192.
[5] 邹益胜, 丁国富, 许明恒, 何邕. 实时碰撞检测算法综述[J]. 计算机应用研究, 2008, 25(1): 8-12.
[6] Hoff I K E, Zaferakis A, Lin M C, Manocha D. Fast 3D geometric proximity queries between rigid & deformable models using graphics hardware acceleration [R]// Technical Report TR02-004. USA: Dept. of Computer Science, University of North Carolina at Chapel Hill, 2002.
[7] Kim Y J, Lin M C, Manocha D. Fast penetration depth estimation using rasterization hardware and hierarchical refinement [C]// Proc of Symposium on Computational Geometry, 2003. USA: ACM, 2003: 386-387.
[8] Govindaraju N K, Redon S, Lin M C, Manocha D. CULLIDE: Interactive collision detection between complex models in large environments using graphics hardware [C]// Proc of ACM SIGGRAPH/ Eurographics Graphics Hardware, 2003. USA: ACM, 2003: 25-32.
[9] Han-Young Jang, Taek Sang Jeong, Jung Hyun Han. Image-Space Collision Detection Through Alternate Surface Peeling [C]// 2007, ISVC. USA: Springer, 2007(1): 66-75.
[10] 霍滨焱. 基于图像空间的碰撞检测算法[D]. 哈尔滨工程大学硕士学位论文. 2005.
[11] 宋永军, 苏鸿根. 一种基于图像的刚体碰撞检测[J]. 计算机应用与软件. 2004, 21(5): 82-84.
[12] 朱连章, 庄华. 基于图像空间的复杂模型碰撞检测算法[J]. 计算机工程与设计. 2007, 28(15): 3675-3677, 3681.
[13] 王季, 翟正军, 蔡小斌. 基于深度纹理的实时碰撞检测算法[J]. 计算机辅助设计与图形学学报. 2007, 19(1): 59-63, 68.

(上接第 920 页)

步扩展，例如远程数据的自动收集、多个中心节点运行控制程序的同时运行，这些扩展需要对方案、仿真任务等模型增加描述能力。

参考文献:

[1] 张柯, 邱晓刚, 彭春光, 等. 分布仿真实验管理系统的设计与实现[J]. 系统仿真学报, 2008, 20(24): 6627-6630. (Design and Implementation of Distributed Simulation Experiment Management System [J]. Journal of System Simulation (S1004-731X), 2008, 24(12): 6627-6630.)
[2] 段伟, 彭春光, 张柯, 等. 分布仿真实验管理系统中实验规划方法研究[J]. 系统仿真学报, 2008, 20(24): 6631-6635. (Research of Experiment Planning Technique in Distributed Simulation Experiment Management System [J]. Journal of System Simulation (S1004-731X), 2008, 20(24): 6631-6635.)
[3] 刘晓铖, 张柯, 陈彬, 等. 分布仿真实验管理系统中实验运行控制工具的设计[J]. 系统仿真学报, 2008, 20(24): 6646-6649. (Design of the Simulation Procedure Controller in Distributed Simulation Experiment Management System [J]. Journal of System Simulation (S1004-731X), 2008, 20(24): 6646-6649.)
[4] LTC John Surdu, Robert L Wittman Jr, Jeff Abbott. Military Scenario Definition Language Study Group Final Report [C]// Proceeding of the 2005 Fall SIW. USA: SIW, 2005.
[5] 王振, 缪旭东. 基于 XML 的作战方案形式化描述[J]. 系统仿真学报, 2006, 18(8): 41-44. (The formalized description about battle schema based on XML [J]. Journal of System Simulation (S1004-731X), 2008, 20(24): 41-44.)
[6] 彭春光, 段伟, 张柯, 等. 分布仿真实验管理系统的兼容性研究[J]. 系统仿真学报, 2008, 20(24): 6643-6645. (Research of Compatibility of Distributed Simulation Experiment Management System [J]. Journal of System Simulation (S1004-731X), 2008, 20(24): 6643-6645.)
[7] 刘健, 张柯, 彭春光, 等. 分布仿真实验管理系统中守护端的设计与实现[J]. 系统仿真学报, 2008, 20(24): 6636-6638. (Design and implementation of Daemon in Distributed Simulation Experiment Management System [J]. Journal of System Simulation (S1004-731X), 2008, 20(24): 6636-6638.)

(上接第 943 页)

因此，当结点总数增加时，传统算法用时增加较快，改进算法用时增加较慢，而且所寻找到的直径是最优的直径，因此，能够适合普遍的要求。撒网法用时最少，所找到的直径比较接近最优直径，但有一定差距，因此，适合结点总数大、算法时间要求高的场合。

4 结论

紧优双环网络的寻找和构造一直是双环网络研究领域一项最重要的课题，本文用仿真的方法对三种寻找紧优双环网络方法进行了比较，结果表明：传统的 L-型瓦算法的时间复杂度较高，随着 N 的增加，其所需要的时间较长；L-型瓦的改进算法在不降低计算精度的情况下大大地提高了效率；当网络中的节点数很大时，采取撒网法来得到一个接近下界的直径，其效率最高，但损失了精度。

参考文献：

[1] 李乔, 徐俊明, 张忠良. 最优双环网络的无限族[J]. 中国科学. A 辑, 1993, 23(9): 979-992.
[2] 徐俊明. 不含紧优和几乎紧优双环网络无限族[J]. 科学通报, 1999, 44(5): 486-490.
[3] 周建钦. k 紧优双环网及其无限族[J]. 数学学报, 2005, 48(6): 1213-1220.
[4] 李颖, 陈业斌, 李中奎. 有向双环网络 G(N;r,s)双紧优分布特性研究[J]. 华中科技大学(自然科学版). 2010, 38(5): 16-18.
[5] Chiuyuan Chen, James K Lan, Wen-Shiang Tang. An efficient algorithm to find a double-loop network that realizes agiven L-shape [J]. Theoretical Computer Science (S0304-3975), 2006, 359: 69-76.
[6] C Y Chen, F K Hwang. Equivalent nondegenerate L-shapes of double-loop networks [J]. Networks (S1097-0037), 2000, 36: 118-125.
[7] C Y Chen, F K Hwang. Equivalent L-shapes of double-loop networks for the degenerate case [J]. Journal of Interconnection Networks (S0219-2659), 2000, 1: 47-60.
[8] 方木云, 赵保华, 屈玉贵. 双环网络 G(N;1,s)的 L-型瓦仿真算法[J]. 系统仿真学报, 2005, 17(4): 914-916.
[9] 陈业斌. 基于二叉树的有向双环网络最优路由算法[J]. 华中科技大学(自然科学版). 2008, 36(6): 43-46.
[10] 林宣治, 陈宝兴. 关于有向双环网络 L-型瓦的四个参数[J]. 漳州师范学报(自然科学版). 2006, 33(2): 12-16.