文章编号:1001 - 9081(2006)11 - 2730 - 03

虚拟现实中动画和碰撞的层次细节技术

黄俊莲[1],吕博学[2]
(1. 大庆石油学院 计算机与信息技术学院,黑龙江 大庆 163318;
2. 北京工业大学 计算机学院,北京 100022)
(netjunlian@sohu.com)

摘 要:研究了动画和碰撞处理中的层次细节技术,提出了动画层次细节二叉树、碰撞层次细节控制矩阵等多种有效的应用解决方法。实验结果表明这些方法应用在存在大量动画和撞碰处理的虚拟三维场景中有效地提高了渲染实时性。

关键词:虚拟现实;层次细节;动画;碰撞
中图分类号: TP391.9 **文献标识码**:A

Level-of-detail technology of animation and collision in virtual reality

HUANG Jun-lian[1], Lü Bo-xue[2]
(1. *Computer & Information Technology College, Daqing Petroleum Institute, Daqing Heilongjiang* 163318, *China*;
2. *College of Computer Science, Beijing University of Technology, Beijing* 100022, *China*)

Abstract: Level-of-detail (LOD) technology of animation and collision were studied. Some effective methods such as animation LOD binary tree, and collision LOD control matrix were put forward. Experiment results show that the improvement of rendering on time can be achieved by applying these techniques to three-dimensional scene with numerous animation and collision operations.

Key words: virtual reality; level of detail; animation; collision

0 引言

虚拟现实中对复杂三维场景的实时绘制提出了越来越高的要求,其中,层次细节(Level-of-detail,LOD)技术是实时绘制技术中最重要的技术之一,同时也被广泛地应用在三维场景的实时渲染领域中。目前,普遍将层次细节技术用于模型简化[1~4]:首先由 Clark[1] 于 1976 年提出,之后由 Hoppe[2] 提出了采用递进网格表示的模型简化算法,1997 年 Garland[3] 提出了基于二次错误率判别的边收缩方式的实时网格模型简化算法,等等。其实,影响渲染实时性的因素不仅仅只有模型绘制方面,大量的、复杂的动画和撞碰处理已然成为影响渲染速度的重要因素。针对存在大量动画和撞碰处理的虚拟场景,相应的优化处理已经成为了提高渲染速度的有效手段。本文将层次细节技术引入到动画和碰撞处理两个方面,提出对动画和碰撞的层次细节技术[5],将静态数据的层次细节处理扩展到动态数据上。

1 动画层次细节技术

从动画的实现角度来说,动画是模型在时间域上不同属性值(例如:位置值)的体现。动画层次细节的思想是在视觉效果允许的情况下根据视点位置不同来控制动画的层次细节。例如场景中模拟人行走时,视点距离人的模型越近动画细节越细致、平滑;视点距离人的模型越远动画细节越粗糙。

首先本文有如下定义:

定义 1 模型 x 在时域 $[0, l]$ 关于属性 $\boldsymbol{P}$ 的动画,可以定义为:$\boldsymbol{A}_{x,p} = \begin{pmatrix} \boldsymbol{T} \\ \boldsymbol{P} \end{pmatrix}$ 其中:$\boldsymbol{T} = \{0, t_1, \cdots, t_{n-2}, 1\}$ 为 n 长的时间序列,n 取值表明了动画的复杂性。$\boldsymbol{P} = \{p_0, p_1, \cdots, p_{n-2}, p_{n-1}\}$ 为 n 长的属性值序列。

$F_{x,p,t}$ 定义了动画 $A_{x,p}$ 在 t 时的更新频率(影响渲染实时性的因素之一)。

不难得出,在时刻 t 时:

当 $t/l \in [t_i, t_{i+1}]$ $(t_i, t_{i+1} \in [0, 1])$,则可求得模型 x 在 t 时刻的属性 $\boldsymbol{P}$ 的值:

$$\boldsymbol{P}_t = F\left(\begin{pmatrix} \{t_i, t_{i+1}\} \\ \{p_i, p_{i+1}\} \end{pmatrix}, \frac{t}{l}\right)$$

其中 $F(A,t)$ 为动画 $A_{x,p}$ 的插值函数(影响渲染实时性的因素之一)。

可以得出,影响动画实时性因素有:动画的复杂度 n、在 t 时的更新频率 $F_{x,p,t}$ 和动画的插值函数 $F(A,t)$。

本文基于定义 1 提出对动画 $A_{x,p}$ 的层次细节算法。

算法 1 静态动画层次细节算法

1) 预先建立 $A_{x,p}$ 的不同层次细节 $\{A^0_{x,p}, \cdots, A^{n-1}_{x,p}\}$。其中 $A^0_{x,p}, \cdots, A^{n-1}_{x,p}$ 的 n,$F_{x,p,t}$, $F(A, t)$ 复杂度各不相同。

2) 实时绘制时,根据模型 x 到视点的距离来选择 $A_{x,p}$ 的层次细节,即距离越远越粗糙原则。

算法 2 动态动画层次细节算法

1) 初始 $A_{x,p}$ 的 n, $F_{x,p,t}$ 和 $F(A,t)$ 值。

2) 在实时绘制过程中,在 t 时刻更新 $A_{x,p}$ 时,根据模型 x 到

收稿日期:2006 - 05 - 12;**修订日期**:2006 - 06 - 22

作者简介:黄俊莲(1978 -),女,黑龙江大庆人,助教,硕士研究生,主要研究方向:数据库; 吕博学(1981 -),男,山西大同人,硕士,主要研究方向:图形、多媒体技术.

视点的距离 d 和后续动画 $\left\{\begin{pmatrix}\{t_i,\ t_{i+1}\}\\ \{p_i,\ p_{i+1}\}\end{pmatrix},\ \begin{pmatrix}\{t_{i+1},\ t_{i+2}\}\\ \{p_{i+1},\ p_{i+2}\}\end{pmatrix},\ \cdots\right\}$ 幅度动态确定 n, $F_{x,p,t}$ 和 $F(A,t)$,实现 $A_{x,p}$ 动态层次细节处理。不同的层次细节下 $A_{x,p}$ 的更新频率 $F_{x,p,t}$ 不同。$A_{x,p}$ 层次细节高(模型离视点近或动画幅度小)时动画刷新快,在每相邻的关键属性值之间会生成大量的中间属性值,从而使产生的动画平滑无停顿,但由于插值计算频繁,渲染的负荷也就越大。同理,层次细节低(模型离视点远或动画幅度小)时动画刷新慢,只在每相邻的关键属性值之间生成少量的中间属性变化值,从而动画不是很平滑(离视点远时或动画幅度小时动画不平滑的效果可以被人所接受),这时插值的计算量小,渲染的负荷也就小,从而实现动画的动态层次细节。

在此给出一个确定 $F_{x,p,t}$ 的示例:

假设:1) 动画采用均匀插值函数; 2) 不可以跳跃关键帧。

设在时刻 t 时: $t/l \in [t_i,\ t_{i+1}](t_i,\ t_{i+1} \in [0,\ 1])$

$$F'_{x,p,t} = \begin{cases} f_0', & d_N < d_t \\ f_1', & d_{N-1} < d_t \leqslant d_N \\ \cdots & \\ f_{N-1}', & 0 < d_t \leqslant d_0 \end{cases}$$

或 $F'_{x,p,t} = F'_0 + \mathrm{e}^{-\lambda' \rho_t}$

其中 d_t 为视点到动画模型间的距离,F'_0, λ' 分别表示频率初值和常数因子。

$$F''_{x,p,t} = \begin{cases} f_0'', & \rho_M < \rho_t \\ f_1'', & \rho_{M-1} < \rho_t \leqslant \rho_M \\ \cdots & \\ f_{M-1}'', & 0 < \rho_t \leqslant \rho_0 \end{cases}$$

或 $F_{x,p,t}'' = F_0'' + \mathrm{e}^{-\lambda'' \rho_t}$

其中 ρ_t 表示动画 $\begin{pmatrix}\{t_i,\ t_{i+1}\}\\ \{p_i,\ p_{i+1}\}\end{pmatrix}$ 的幅度,F''_0,λ'' 分别表示频率初值和常数因子。

由假设 1) 有: $\rho_t = \left|(p_{i+1} - p_i)/(t_{i+1} - t_i)\right|$,综合视点和动画幅度两种因素,$F_{x,p,t} = w_1 F'_{x,p,t} + w_2 F''_{x,p,t}$,其中 w_1, w_2 为影响因子。

另外,本文借鉴了网格模型简化算法[1~5],预先简化动画序列为动画 LOD 二叉树,将相邻动画片段的幅度误差作为动画简化度量标准。在绘制时基于动画 LOD 二叉树根据视点实时对动画序列进行折叠和分裂操作,实现多层次细节处理。本文给出如下示例。

设动画 $A_{x,p} = \begin{pmatrix} 0 & 0.1 & 0.3 & 0.7 & 0.9 & 1 \\ 3 & 7 & 8 & 9 & 11 & 6 \end{pmatrix}$,属性为 $(\rho,\ d,\ F,\ Fn())$,其中: ρ 为动画的幅度, d 为调用动画的距离阈值,F 是动画更新频率,定义 $F = 1/\rho$, $Fn()$ 是动画插值函数。可以生成图 1 所示的动画 LOD 二叉树。

如果在时刻 $t = 0.05$(t 为相对时刻,范围在 0 到 1 之间)时,当在视点离动画模型在 50 以外则使用动画(1),在 40 至 50 范围则使用动画(2),否则使用动画(3)。

本文的视点规则采用了距离规则和动作误差规则两种。距离规则把动画所涉及的模型到视点的距离作为 LOD 的视点规则;动作误差规则把动画幅度作为 LOD 的视点规则。

值得注意的还有动画的整体性,在很多情况下多个动画的统一执行才可以准确地体现一个动画系统(如人体动画),所以在对动画层次细节处理时要特别注意到动画的整体性。对有联系的动画组合要进行统一的层次细节的处理,这样才能不影响动画的准确性。

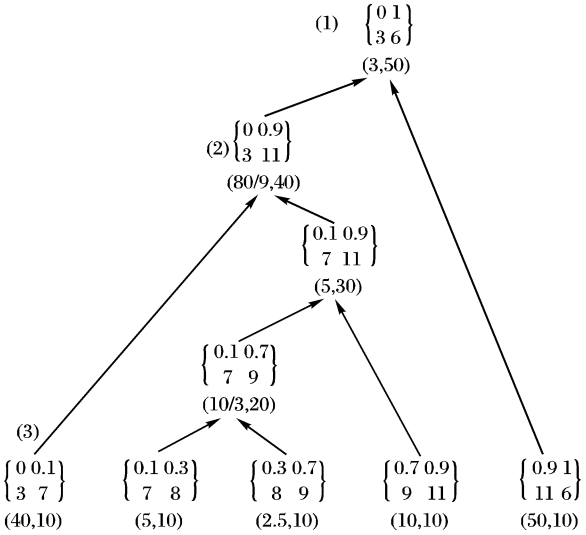


图 1　动画 LOD 二叉树

2　碰撞层次细节技术

首先,给出一个加入了碰撞检测层次细节技术的简单碰撞算法[6]:

```
// 碰撞对数据数组
CArray3d < CCollision > m_ArrayCollision;
// m_ArrayCollision 计录了所有的碰撞对
// 碰撞检测与处理函数
int CollisionUpdate( float fTime)                    // fTime 是当前时间
{
  CCollision * pCollisionCurrent = NULL;
  for (j = 0; j < m_ArrayCollision. GetSize(); j ++)
  // 循环每一个碰撞对
  {
    pCollisionCurrent = m_ArrayCollision[ j];
    // pCollisionCurrent 是一个碰撞对
    if( pCollisionCurrent -> Run( fTime) );
    // 用碰撞对的检测频率进行 LOD 控制
    {
      CollisionLODProcess( pCollisionCurrent );
      // 对碰撞对进行 LOD 碰撞检测和处理
    }
  }
}
// 在碰撞对中对检测频率进行层次细节控制
int CCollision: : Run( float fTime)
{
  if ( fTime - m_fLastUpdateTime > m_fUpdateFre) {
    // m_fUpdateFre 为撞碰检测的更新频率
    m_fLastUpdateTime = fTime;
    LODUpdate();                                     // LOD 控制更新频率
    //根据视点和碰撞对象属性更新 m_fUpdataFre
    return 1;
  }
  else{
    return 0;
  }
}
```

2.1 用碰撞对位置和视点位置来确定碰撞检测频率

从上面的算法可以看到:m_fUpdataFre 取值较大时,有可能在检测时遗漏掉碰撞情况以至产生穿透现象。m_fUpdataFre 取值较小时,虽然碰撞计算结果准确,但计算量增大。

为了同时获得满意的速度和准确度,本文采用可适应性变化的步长来取代固定步长,同时由步长来确定碰撞处理的层次细节,当碰撞发生的频率较高且视点离碰撞的物体很近时,检测步长相应变小(即要增加碰撞检测的层次细节);当碰撞发生的频率降低或视点离碰撞的物体很远(因为即使发生穿透现象也不会影响视点观察效果,在穿透后还需进行调整处理)时,检测步长相应变长(即要减小碰撞检测的层次细节)。这一问题也被称为时间步长问题。

时间步长问题的实质是计算碰撞频率问题,原因是在有些时间片中物体根本不可能发生碰撞。如果已知了物体的运动参数,可以通过预知物体未来碰撞时间和地点,由此避免作多余的时间片检测[7,8]。但很多时候很难假定物体的预先运动轨迹。

本文在该问题中加入了视点因素,在不影响视觉效果的情况下允许穿透现象。由视点位置和碰撞对的位置关系共同来决定了的碰撞对的碰撞检测频率。

在此给出一个确定 $F_{x,p,t}$ 的示例,同时引入碰撞检测频率 LOD 矩阵。

假设:1)碰撞属于无规则碰撞;2)可以检测和处理穿透现象。

定义碰撞对 (A, B),O 为 (A, B) 的中点,$F_{A,B}$ 为 (A, B) 的碰撞检测频率,$d_{A,B}$ 为 A,B 间的距离,$d_{V,O}$ 为视点到 O 点的距离。

同动画层次细节类似可有以下公式:

$$F'_{A,B} = \begin{cases} f_{N+1}', & d_N < d_{A,B} \\ f_N', & d_{N-1} < d_{A,B} \leqslant d_N \\ \cdots \\ f_0', & 0 < d_{A,B} \leqslant d_0 \end{cases}$$

$$F''_{A,B} = \begin{cases} f_{M+1}'', & d_M < d_{V,O} \\ f_M'', & d_{M-1} < d_{V,O} \leqslant d_M \\ \cdots \\ f_0'', & 0 < d_{V,O} \leqslant d_0 \end{cases}$$

综合视点位置和碰撞对的位置关系两种因素,$F_{A,B} = w_1 F'_{A,B} + w_2 F''_{A,B}$,其中 w_1, w_2 为影响因子。

本文引入碰撞检测频率的 LOD 控制矩阵 $\boldsymbol{C}_{FrequencyLOD}$ 来控制 $F_{A,B}$:

$$\boldsymbol{C}_{FrequencyLOD} = \begin{pmatrix} 0 & d'_0 & d'_1 & \cdots & d'_j & \cdots & d'_{N+1} \\ d''_0 & f_{0,0} & f_{0,1} & \cdots & f_{0,j} & \cdots & f_{0,N+1} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ d''_i & f_{i,0} & f_{i,1} & \cdots & f_{i,j} & \cdots & f_{i,N+1} \\ \vdots & \vdots & \vdots & & \vdots & & \vdots \\ d''_{M+1} & f_{M+1,0} & f_{M+1,1} & \cdots & f_{M+1,j} & \cdots & f_{M+1,N+1} \end{pmatrix}$$

若:$d'_{i-1} < d_{A,B} < d'_i$,且 $d''_{j-1} < d_{V,O} < d''_j$,则:$F_{A,B} = f_{i,j} \circ \boldsymbol{C}_{FrequencyLOD}$ 设置了碰撞对碰撞频率的 LOD 值。同时该矩阵也用来确定碰撞的层次。

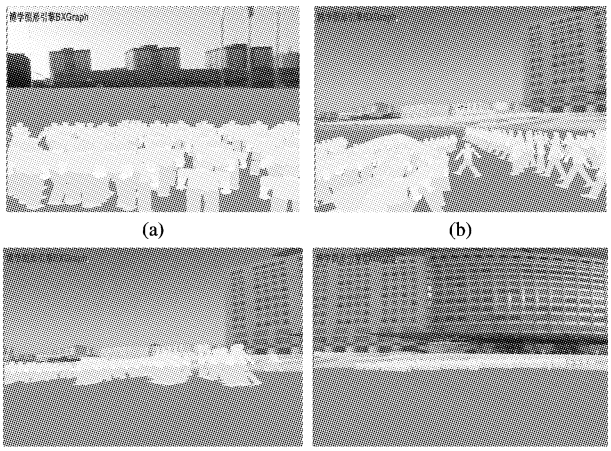
2.2 碰撞检测的层次细节处理

本文预先建立碰撞包围盒和碰撞检测方法的多种层次细节,然后在实时绘制中根据视点和碰撞对的属性来确定碰撞检测频率及层次细节。

碰撞检测实际上是对碰撞物体的包围盒进行碰撞检测。首先我们要给每一个参与碰撞物体建立不同层次细节的包围盒,如点包围盒(用点来代替物体参与碰撞检测)、球包围盒、轴向包围盒 AABB、OBB 包围盒等等。在不同的层次细节下采用不同的包围盒。包围盒检测方法是碰撞检测用到的最广的方法,同时包围盒还被运用于对光线跟踪和求交运算的加速。对包围盒的层次细节处理可以很大程度地减少碰撞检测的计算量。另外,碰撞使用的检测方法也因碰撞层次细节的不同而不同,层次细节越高检测的方法越复杂,越低则越简单。

3 结语

本文对 150 人的人群进行动画和碰撞模拟,在 4 种不同视点位置下(如图 2)采用动画 LOD 二叉树和碰撞检测频率的 LOD 控制矩阵的方法进行层次细节控制,并以平均帧速率为指标比较了应用前后的实验效果,测试所用硬件平台为 P4 1.8G CPU,NVIDIA Quadro FX 500 显卡,512M DDR 内存,17″纯平显示器;屏幕区域大小为 1 024 × 768 像素,颜色为 32 真彩色。



(a) (b) (c) (d)

图 2 人群动画中不同视点下的效果图

在如上 4 种视点位置下,针对存在大量的、复杂的动画和碰撞数据的人群场景采用层次细节的方法进行渲染,可大幅提高系统的渲染效率,同时也能有效地保持场景渲染的真实感。表 1 为平均帧速率(fps)指标下的性能比较。

表 1 人群动画层次细节的测试数据表

视点位置	平均帧速率/fps	
	未使用 LOD 处理	使用 LOD 处理
(a) 图视点位置	3.7	7.2
(b) 图视点位置	4.2	8.2
(c) 图视点位置	3.5	11.1
(d) 图视点位置	3.9	18.0

目前,虚拟现实中动画、碰撞处理已成为影响绘制实时性的重要因素,例如:人群动画模拟、粒子系统模拟等。本文借鉴模型层次细节技术,研究动画和碰撞的层次细节算法,提出了多种有效的应用手段。在实际的场景渲染应用中这些方法都得到了有效的验证:在视觉效果保证的前提下达到了对动态场景的实时渲染。

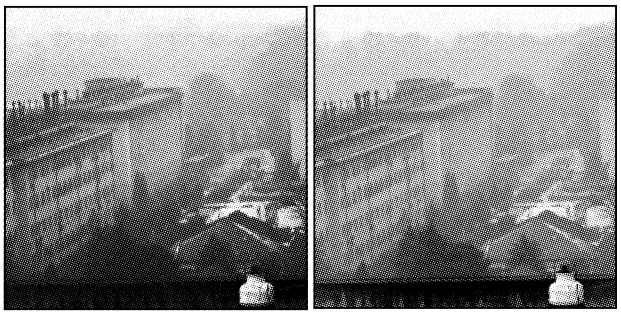
(下转第 2735 页)

的最大和最小值。显然,I_{hi} 和 I_{low} 的选取是处理效果好坏的关键。通过仿真分析我们发现,对浓雾退化图像,其交互式去雾处理结果的直方图类似呈正态分布。因此可以通过计算各谱分量亮度值的均值和方差来确定其 I_{hi} 和 I_{low}。根据正态分布的特性,我们取:

$$\begin{cases} I_{hi} = \mu + 3\sigma \\ I_{low} = \mu - 3\sigma \end{cases} \tag{12}$$

其中,μ 和 σ 分别是该谱分量亮度值的均值和标准差。

3　仿真结果



(a) 直方图均衡处理结果　　(b) 交互式去雾处理结果

图 3　薄雾退化图像处理

(a) 原始浓雾退化图像　　(b) 直方图均衡处理结果

(c) 交互式去雾处理结果　(d) 经亮度调整的交互式去雾处理结果

图 4　浓雾退化图像处理

我们对图 2 所示的薄雾退化图像进行了去雾处理,结果如图3所示。图3(a)为采用直方图均衡增强处理的结果。从该图中可以看到,近处的景物被过增强,而远处的景物变得更模糊了,且存在块效应和色彩失真。同时我们采用交互式去雾算法进行处理,以图 2 中所示的消失点为参考点,取 $\beta = 0.3$,$n = 0.6$,$d_{\min}$ 与 $d_{\max}$ 分别为 0.05 和 0.9,处理结果如图 3(b)所示。从该图中可以看到,近处与远处的景物得到了不同程度的增强,细节比较分明,色彩较自然,且远处树木的轮廓比较清晰。

图 4(a)所示为浓雾退化图像,由于雾很浓,整个图像呈昏暗色。图 4(b)为直方图均衡增强结果,同样存在近区景物过增强及块效应。我们取 $\beta = 0.8$,$n = 0.3$,$d_{\min}$ 与 $d_{\max}$ 分别为 0.2 和 0.9,进行交互式去雾处理,其结果如图 4(c)所示。从该图中可以看到,图像景物比较清晰,但整体亮度较低。图 4(d)为图 4(c)经对比度正态截取拉伸调整后的结果,无论是近处景物的细节还是远处景物的轮廓都得到了有效增强,图像整体亮度得到了改善,获得了较好的视觉效果。

参考文献:

[1] 祝培,朱虹,钱学明,等. 一种有雾天气图像景物影像的清晰化方法[J]. 中国图象图形学报,2004, 9(1):124 – 128.

[2] SCHECHNER YY, NARASIMHAN SG, NAYAR SK. Instant dehazing of images using polarization[A]. Proceedings of the IEEE Computer Society Conference on CVPR[C]. Kauai, Hawaii: IEEE Computer Society, 2001. 325 – 332.

[3] SCHECHNER YY, NARASIMHAN SG, NAYAR SK. Polarization-based vision through haze[J]. Applied Optics, 2003, 42(3): 511 – 525.

[4] OAKLEY JP, SATHERLEY BL. Improving image quality in poor visibility conditions using a physical model for contrast degradation [J]. IEEE Transactions on Image Processing, 1998, 7(2): 167 – 179.

[5] NARASIMHAN SG , NAYAR SK . Removing weather effects from monochrome images[A]. Proceedings of the IEEE Computer Society Conference on CVPR[C]. Kauai, Hawaii: IEEE Computer Society, 2001. 186 – 193.

[6] NARASIMHAN SG, NAYAR SK. Interactive (De) Weathering of an Image using Physical Models[A]. Proceedings of the ICCV Workshop on Color and Photometric Methods in Computer Vision[C]. Nice, France: IEEE Computer Society, 2003. 1387 – 1394.

[7] NARASIMHAN SG, NAYAR SK. Contrast restoration of weather degraded images[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003, 25(6): 713 – 723.

[8] NAYAR SK, NARASIMHAN SG. Vision in bad weather[A]. Proceedings of the IEEE International Conference on Computer Vision [C]. Corfu, Greece: IEEE Computer Society, 1999. 820 – 827.

(上接第 2732 页)

参考文献:

[1] JAMES C. Hierarchical geometric models for visible surface algorithms[M]. Communications of ACM, 1976, 19(8): 454 – 460.

[2] HOPPE H. Progressive Meshes[A]. Proceedings of ACM SIGGRAPH 1996[C], 1996. 99 – 108.

[3] GARLAND M, HECKBERT PS. Surface simplification using quadric error metrics[A]. Proceedings of ACM SIGGRAPH 1997[C], 1997. 209 – 216.

[4] LUEBKE D. A developer's survey of polygonal simplification algorithms[J]. IEEE Computer Graphics and Applications, 2001, 21(3): 24 – 35.

[5] 吕博学. 虚拟现实中层次细节技术研究及应用[D]. 北京工业大学, 2005.

[6] 石教英. 虚拟现实基础及实用算法[M]. 北京:科学出版社, 2002.

[7] CANNY J . Collision detection for moving polyhedra [J] . IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986, 8(2): 200 – 209.

[8] HUBBARD PM. Collision detection for interactive graphics applications[J]. IEEE Transactions on Visualization and Computer Graphics, 1995, 1(3).