凸多面体碰撞检测的棱线投影分离算法

张 智, 邹盛涛, 李佳桐, 张乐乐, 李 超

(哈尔滨工程大学自动化学院 哈尔滨 150001)
(foevergoodzst@163.com)

摘 要: 针对凸多面体碰撞检测问题, 以直线投影法为基础对分离面投影法进行改进, 提出一种采用棱线投影分离的凸多面体实时精确碰撞检测算法. 首先分析了凸多面体各种相对位置关系并提出了投影分离线的概念, 针对凸多面体的各种分离情况证明投影分离线的存在; 其次选取凸多面体相向面上的棱集构造准投影分离线, 通过沿着准投影分离线方向投影可将 3D 凸多面体碰撞检测降维为 2D 凸多边形的碰撞检测问题; 最后将分离投影的思想延用至为 2D 凸多边形的碰撞检测, 再次将 2D 问题降维为 1D 问题. 算法分析和实验结果表明, 该算法对于凸多面体碰撞检测具有较高的响应速度和检测精度.

关键词: 凸多面体; 碰撞检测; 投影分离线
中图法分类号: TP391.9; TP242.2

A Collision Detection Algorithm between Convex Polyhedrons Based on Projection of Edges

Zhang Zhi, Zou Shengtao, Li Jiatong, Zhang Lele, and Li Chao

(*College of Automation, Harbin Engineering University, Harbin* 150001)

Abstract: This paper presents a fast and accurate collision detection algorithm for convex polyhedrons, which employs the method of projection-separating edges based on linear projection. This algorithm improves the algorithm of projection-separating planes. First, this paper proposes the concept of projection-separating lines, and proves the existence of the separating lines in any separated condition. Second, the quasi projection-separating lines set is created from the opposite surfaces on convex polyhedrons, then the problem of the collision detection between 3D convex polyhedron is converted to the collision detection between 2D convex polygons by the projection of the lines; Third, the projection method has been applied to convex polygons and 2D problem is converted to 1D problem. The experimental results show that the proposed algorithm has high detection accuracy and response speed.

Key words: convex polyhedron; collision detection; projection-separating lines

碰撞检测在虚拟装配、系统仿真、机器人路径规划等领域都有重要的应用[1]. 例如在机器人的无碰撞路径规划中, 需要将机器人的关节和环境障碍以及目标物体等不规则物体等效为凸多面体集合, 然后进行多个凸多面体集合间碰撞检测, 因此碰撞检测的实时性和准确性将直接影响到机器人

收稿日期：2014-06-18；修回日期：2015-05-11. 基金项目：国家自然科学基金(61104037, 61304060); 中央高校基本科研业务费专项资金(HEUCF041307, HEUCFX41304); 国家科技合作专项项目(2013DFR10030). 张 智(1981—), 男, 博士, 副教授, 主要研究方向为智能控制技术、虚拟现实; 邹盛涛(1991—), 男, 硕士研究生, 主要研究方向为机器人控制技术; 李佳桐(1990—), 女, 硕士研究生, 主要研究方向为智能控制技术; 张乐乐(1992—), 男, 硕士研究生,主要研究方向为机器人控制技术; 李 超(1991—), 男, 硕士研究生, 主要研究方向为机器人控制技术.

最优轨迹规划的速度与精度. 碰撞检测采用离散时间检测算法, 主要有 4 类: 1) Gilbert 等[2]提出的距离算法(Gilbert-Johnson-Keerthi, GJK)在明氏距离多面体参数中进行搜索和跟踪最小距离点对, 该算法具有线性时间复杂度, Cameron[3]利用爬山法对其进行改进并提出了 EGJK 算法(enhanced-Gilbert-Johnson-Keerthi, EGJK); 2) Mirtich 等[4]提出了基于特征的算法(Voronoi-clip, V-Clip), 该算法能够快速收敛到模型的邻近特征, 碰撞检测效率较高, 但需要考虑 5 种特征组合, 实现难度较大; 3) Chung 等[5]提出一种分离向量算法(Chung-Wang, CW), 其检测效率是 GJK 算法的 2 倍, 但 CW 算法的收敛性证明不完善; 李学庆等[6]对 CW 算法进行改进并提出 HS-jump 算法, 给出了严格的终止条件, 该算法对于近似球状凸多面体效率较高; 4) 黎自强等[7]提出一种投影分离面法, 通过构造准投影分离面来加速和实现碰撞检测.

本文在借鉴黎自强算法[7]思想的基础上, 针对凸多面体的各种相对位置进行了系统分析, 首先分析了黎自强算法[7]的优缺点, 指出其分离检测中不完善之处; 并提出一种新的基于棱线投影分离的凸多面体碰撞检测方法以实现更完备的分离检测; 在各种相对位置情况下均证明了算法的有效性. 此外, 本文将投影线分离思想扩展到凸多边形碰撞检测, 计算出凸多面体和凸多边形的最近点(或最近距离). 实验和分析证明, 本文算法对于凸多面体各种相对位置情况均能实现有效碰撞检测, 并且具有与黎自强算法[7]相近的计算速度.

1 凸多面体的碰撞检测

1.1 凸多面体的相对位置分析

为了描述凸多面体的碰撞检测关系, 根据 2 个凸多面体从不相交到相交瞬间的相对位置关系, 分为 6 类, 如图 1 所示.

1) 穿刺. 当凸多面体相交时, 交集是一个空间域且小于 2 个凸多面体时, 称为穿刺, 如图 1a 所示.

2) 包含. 当凸多面体相交时, 交集是一个空间域且等于其中一个凸多面体时, 称为包含, 如图 1b 所示.

3) 面相交. 当临界碰撞位置包含凸多面体面上的内点(除边和顶点外的点)时, 称为面相交, 如图 1c~图 1e 所示.

4) 棱相交. 当临界碰撞位置为 2 个凸多面体各自棱上的点时, 且两棱不平行(否则为棱点相交), 称为棱相交, 如图 1f 所示.

5) 棱点相交. 当临界碰撞位置为某一凸多面体的顶点和另一凸多面体棱上的点时, 称为棱点相交, 如图 1g 所示.

6) 点相交. 当临界碰撞位置为 2 个凸多面体各自顶点时, 称为点相交, 如图 1h 所示.

需要说明的是, 3) ~6) 中指的是 2 个凸多面体按不同的方式靠近尚未接触前的“临界相交” 情况, 为方便描述, 后文均将“临界相交” 简称为“相交”.

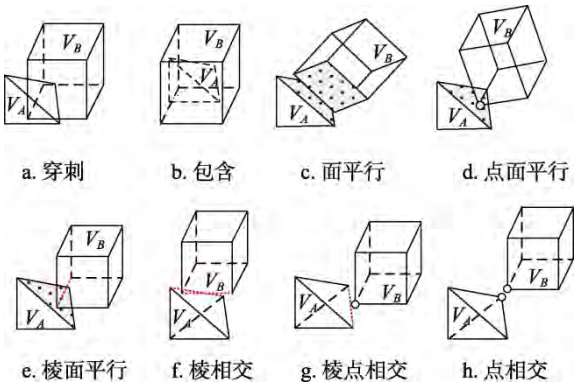


图 1 凸多面体的相对位置关系

1.2 与传统分离检测方法对比

凸多面体具有良好的几何特性, 针对凸多面体的各种相对位置关系, 一种十分有效的方法即是分离测试. 一般来说, 对于包含相同数量面(F)和棱边(E)的凸多面体, 确定一次干涉检测最多需要考察的潜在分离轴数量为 $2F+E^2$, 若全部轴测试完毕, 则对象一定相交. 文献[5]提出一种快速搜索分离轴的算法, 该算法始于任意非零向量, 根据某一特定公式求取反射向量, 通过迭代最终得到分离向量, 平均搜索速度高于 EGJK 算法. 文献[6]等对文献[5]的分离轴算法进行改进, 给出搜索分离轴的终止条件, 对于近似球状的物体该算法检测效率较高. 文献[8]基于分离面法构造分离包围盒实现凸多面体快速碰撞检测, 但检测精确性有待完善. 文献[7]选用凸多面体的部分表面作为分离面, 并沿着分离面与坐标平面的交线方向投影, 将 3D 凸多面体碰撞转换为 2D 凸多边形的相交问题, 该算法平均检测效率较高, 对于多数情况均能准确给出碰撞结果; 但是对于第 1.1 节中分析的棱相交、棱点相交等情况, 该算法碰撞检测失效. 因此, 本文提出一种新的基于棱线投影分离的

碰撞检测算法，不仅解决了文献[7]中的问题，而且对投影分离的原理进行了严格证明，并给出了最近点计算方法.

2　凸多面体在坐标平面上的投影

2.1　凸多面体沿任意空间直线在坐标面的投影

设凸多面体的顶点坐标为 $A_i(x_i,y_i,z_i)$，对于凸多面体 V 的面 F_j 的棱边 E_k，求出方向矢量 $\boldsymbol{D}_k(a_k,b_k,c_k)$.当 $\boldsymbol{D}_k$ 与坐标平面 xoz 不平行，即 $b_k\neq 0$ 时，点 A_i 在平面 xoz 上的投影为 $A_i'(x_i',y_i',z_i')$. 计算公式为

$$[x_i' \quad y_i' \quad z_i' \quad 1]=[x_i \quad y_i \quad z_i \quad 1]\times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\dfrac{a_k}{b_k} & -\dfrac{c_k}{b_k} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

若 $a_k\neq 0$，A_i 沿 $\boldsymbol{D}_k$ 向坐标平面 yoz 作投影；若 $c_k\neq 0$，A_i 沿 $\boldsymbol{D}_k$ 向坐标平面 xoy 作投影.

2.2　计算凸多面体在坐标平面的投影边界

由顶点集求取其凸多边形外轮廓有诸多方法，文献[7]采用了 Quickhull 算法，当用数组表示顶点时计算难度较大，而且需要考虑无法构造四边形等特殊情况，本文采用 Andrew 算法[5].

值得注意的是，当构造凸多面体在坐标平面的投影边界时，如果只发生平移运动，2 个凸多面体的所有棱线方向将保持不变，则沿着同一棱线方向在坐标平面的投影边界也将不变(如图 2 所示，本文规定向右方向为坐标轴 x，竖直向上为坐标轴 y，z 轴指向满足右手法则)，因此仅需将上次投影多边形沿相同矢量(仅选取一点作前后 2 次投影，2 次坐标值作差即得到该矢量)平移即可，无需重复计算所有顶点的投影坐标值及重新构造投影边界，这样可提高投影运算的速度，提高凸多面体碰撞检测效率. 具体过程如下：

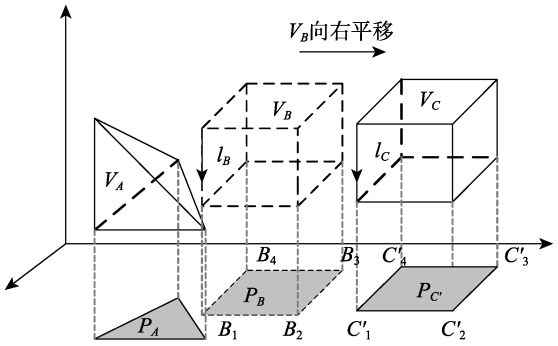


图 2　凸多面体平移示意图

已知 V_A,V_B 投影点 $\{A_1',A_2',A_3',A_4'\},\{B_1',B_2',B_3',B_4'\}$，一段时间后 V_B 平移到 V_C，V_A 保持不变，由于 $l_B\parallel l_C$，应用式(1)可知，V_A,V_C 沿着 l_C 投影后 V_A 的投影坐标值仍为 $\{A_1',A_2',A_3',A_4'\}$；若 C_1' 点经计算已知，则 V_C 其余投影点计算公式为 $C_i'=B_i'+\boldsymbol{B}_1'\boldsymbol{C}_1'$ $(i=2,3,4)$.

由于向量 $\boldsymbol{B}_1'\boldsymbol{C}_1'$ 的模是一个常量，据凸多边形边界的计算方法可知，V_B 与 V_C 具有相同的投影边界.

3　2 个凸多面体的碰撞检测

3.1　投影分离线及其性质

基于第 2.1 节给出的投影计算方法，继续引入投影分离线的概念，并给出其相关性质证明，作为后文分离检测方法的基础.

定理 **1.** 空间不相交的凸多面体 V_A,V_B 沿着分离面 F 上任意一条直线 l (称为投影分离线)的方向投影到某坐标平面，则它们的投影区域 P_A,P_B 必不相交.

证明. 如图 3 所示，首先令凸多面体 V_A,V_B 沿直线 l 方向延伸，形成的扫掠体[5]分别记为 W_A,W_B，P_A,P_B 分别是 W_A,W_B 与 xoz 面相交形成的截面，故 $P_A\in W_A,P_B\in W_B$，由于 W_A,W_B 是凸多面体 V_A,V_B 沿着相同的直线 l 扫掠形成，故 $W_A\cap W_B=\varnothing$,所以 $P_A\cap P_B=\varnothing$，定理 1 得证. 证毕.

根据定理 1 可知，当凸多面体沿着它们分离面内的某条投影分离线方向向某坐标平面投影时，通过检测平面投影区域的分离即可判定空间凸多面体的分离. 需要说明的是，多数情况下投影的坐标平面可任意选择，但当投影分离线恰平行于某

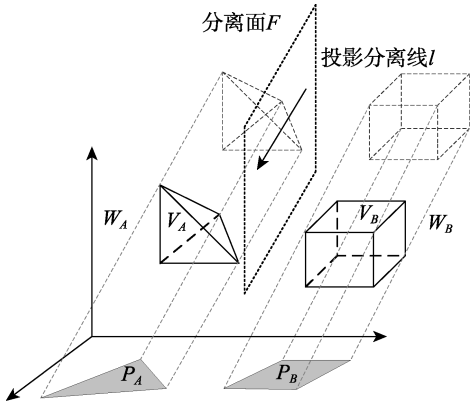


图 3　凸多面体的投影分离图

坐标平面时，则必须另外选择与其不平行的坐标平面作投影.

推论 **1.** 若存在直线 l 位于凸多面体 V_A,V_B 的分离面上，则直线 l 是 V_A,V_B 的投影分离线.

根据定理 1，推论 1 结论显然成立.

3.2 基于棱线投影分离的凸多面体碰撞检测

第 1.1 节给出了凸多面体可能发生碰撞的情况，本节针对后面 4 种情况分别讨论基于投影分离线的分离检测方法，而对于前 2 种情况(穿刺或包含)，只要下面 4 种情况均不能有效分离，即可判为碰撞.

定理 **2.** 凸多面体 V_A,V_B 不碰撞，则 V_A,V_B 所组成的棱集至少存在一个投影分离线.

为证明定理 2，首先针对 V_A,V_B 可能发生碰撞的 4 种情况(面相交，棱相交，棱点相交，点相交)证明均存在投影分离线.

证明. 1) 面相交情况下凸多面体将在某个凸多面体相向面发生碰撞，文献[7]研究了面相交的情况，指出图 4 中 V_A 的 S_A 面构成分离面 F，再由推论 1 可知，S_A 面上的任意边均可作为投影分离线(如棱边 l_A)，第 1 种情况得证.

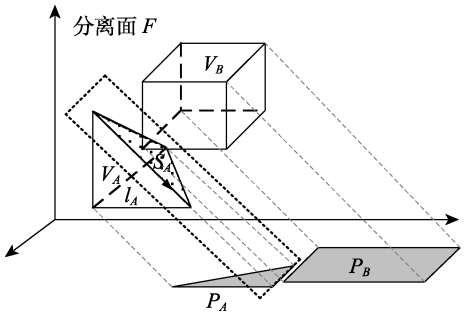


图 4 面相交时的投影线分离图

2) 棱相交时凸多面体 V_A,V_B 间的最短距离为直线 l_A 到 l_B 的距离，即将发生棱对棱的碰撞(如图 5 所示). 在未发生碰撞前分离面必存在[7]，且分离面必然也能将 l_A 与 l_B 分开，当凸多面体不断接近并趋于临界碰撞时，两直线 l_A 与 l_B 的距离也将趋于 0，此时的分离面必然包含 l_A 和 l_B，故棱线 l_A 和 l_B 均为投影分离线，第 2 种情况得证.

3) 棱点相交时凸多面体 V_A,V_B 间的最短距离为点 Q_B 到直线 l_A 的距离，即将发生点对棱的碰撞(如图 6 所示)，仍采用第 2) 步的证明思想. 未碰撞时分离面必存在，且能将点 Q_B 和棱线 l_A 分开，当凸多面体趋于碰撞时，点 Q_B 和棱线 l_A 距离趋于 0，则分离面必然包含点 Q_B 和棱线 l_A，故棱线 l_A 为投影分离线，第 3 种情况得证.

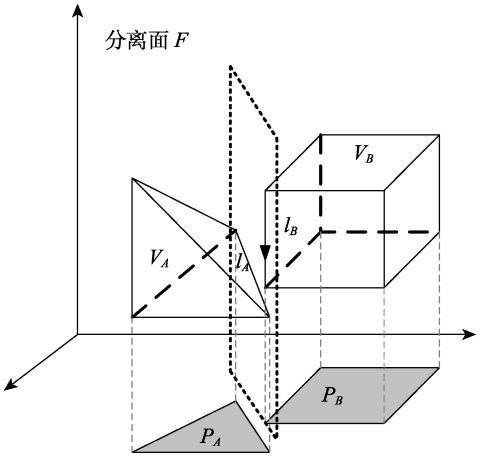


图 5 棱相交时的投影线分离图

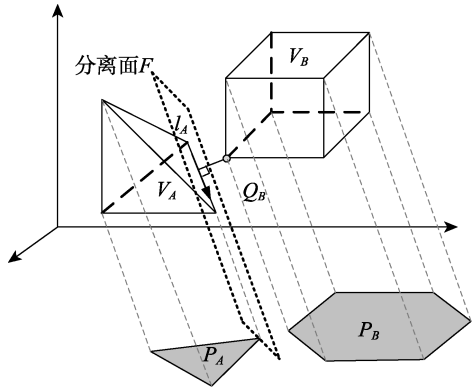


图 6 棱点相交时的投影线分离图

4) 2 个凸多面体间的最短距离为点对点的距离，在第 5.1 节 2) 中向直线投影作分离预检测时自然涵盖了该情况，此处略.

综上所述，针对凸多面体 V_A,V_B 的 4 种不相交位置，均能从相向面的棱集中找到投影分离线实现分离检测，定理 2 得证. 证毕.

为了实现完备的分离检测，首先需构造凸多面体 V_A,V_B 对应相向面集合 $\{{}^{AB}F_i\,|_{I_i>0}\}$. 根据文献[7]，2 个凸多面体碰撞时首先发生在相向面，再由定理 2 可知，将该集合中所有棱边(去掉重复边)取出即可构成准投影分离线集 $\{{}^{AB}E_i\}$. 在实际选择准投影分离线时，应按照其所在平面的摩擦值排序，并优先选取摩擦值较大的平面所对应的棱边作分离检测(根据文献[7]结论，多数情况下碰撞发生在摩擦系数较大的平面中)，一旦分离成功则终止算法，这样可提高检测速度.

至此，本文证明了通过构造投影分离线集实现不相交凸多面体分离检测的方法；反之，若遍历投影分离线集均不能实现分离，则 2 个凸多面体

碰撞. 还需注意，由于相邻的面存在公共棱，在实际执行算法时必须标记已经作过投影分离检测的棱(记录其法向量值)，如果再次遍历到相同的棱或相同的法向量值，则跳出此次投影分离检测，防止重复向坐标平面作投影，减少计算量.

4　2 个凸多边形的碰撞检测

对于 2 个凸多边形的碰撞检测，常见的方法有 3 类: 1) 线性规划法[5]; 2) 分离直线法[9]; 3) 向量叉积法[10]. 本文将空间凸多面体的投影分离法沿用至平面凸多边形干涉检测，并且能准确计算 2 个不相交凸多边形的最近点(或最近距离).

4.1　2 个凸多边形的准投影分离线集

凸多边形与凸多面体具有类似的性质，因此由定理 2 可知，两不相交的凸多边形一定存在投影分离线. 与凸多面体类似，按棱摩擦系数降序原则，分别在准分离面的棱集中依次取 j_2 和 h_2 个相向棱边，即构成 P_A,P_B 的投影分离线集 $\{{}^{A}_{z}e_i\},\{{}^{B}_{z}e_i\}$. 如果凸多边形 P_A,P_B 分离，则投影分离线一定在 P_A,P_B 构成的准投影分离线 $\{{}^{A}e_i\},\{{}^{B}e_i\}$ 集中.

本文依次选取凸多边形 P_A,P_B 的棱边对坐标轴作投影，通过检测坐标轴投影区域的分离即可判定平面凸多边形的分离，凸多边形与凸多面体投影分离检测算法类似，本文不再详述.

4.2　基于棱线投影分离的凸多边形分离检测

当凸多边形分离时，针对图 7 所示的情况，凸多边形 P_A,P_B 间的最短距离 h' 与投影区域的最短距离 h'' 满足

$$h'=h''\left|\sin\beta\right| \tag{2}$$

其中，β 为投影分离线 l 与坐标轴的夹角(如果 $\beta=0$，则改向另一轴作投影).

若 $h''>0$，则 $h'>0$，则凸多边形 P_A,P_B 不相交; 若 $h''\leqslant 0$，则 P_A,P_B 相交.

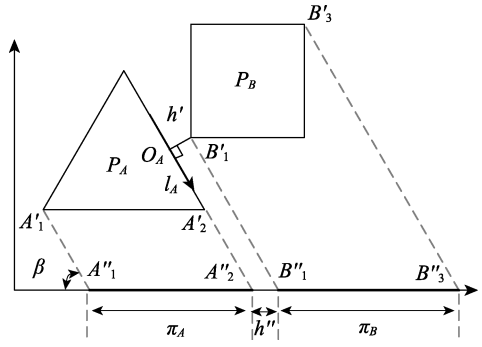


图 7　凸多边形投影分离图

对于任意相对位置关系，2 个凸多边形上对应的最近点可按如下方式求取: 检测某凸多边形顶点的投影点中最靠近另一凸多边形投影的点，如该点唯一，则对应凸多边形上的一个定点; 如为 2 个点，则对应凸多边形上的一个边. 本文可以根据点与点或点与边等相对位置关系计算最近点(例如图 7 中的 B_1' 点和 O_A 点)，进而计算最短距离. 同理，依据此方法，可最终反推出 3D 凸多面体对应的最近点和最近距离.

值得注意的是，对于凸多边形碰撞检测，向量叉积法[10]搜索分离边速度较快，碰撞检测执行效率高于本文算法，但本文算法为了兼顾最短距离求取，仍然保留了对原文中凸多边形棱线投影分离检测方法的论述，实际使用中可根据是否需要计算最短距离而选择不同方法.

5　凸多面体的碰撞检测算法

下面给出基于棱线投影分离的凸多面体碰撞检测的完整算法.

5.1　预碰撞检测

预碰撞检测是为了加快算法执行效率，包括分离预检测和碰撞预检测.

1) 分离预检测. 利用中心线上的正投影判断 2 个凸多面体分离，可以加快分离检测. 若凸多面体 $V_A(A_1,A_2,\cdots,A_n),V_B(B_1,B_2,\cdots,B_m)$ 的中心分别为 C_A,C_B，直线 C_AC_B 的参数方程为 $y=(1-t)C_A+tC_B$，分别过 2 个凸多面体的顶点 P_i 作线段 C_AC_B 的垂线，则垂足的参数为

$$t_i=\boldsymbol{C_AC_B}\cdot\boldsymbol{C_AP_i}/\left|\boldsymbol{C_AC_B}\right|^2,\ \ i=1,2,\cdots,n+m \tag{3}$$

其中，当 $1\leqslant i\leqslant n$ 时，$P_i=A_i$; 当 $n+1\leqslant n+m$ 时，$P_i=B_i$. 设 V_A,V_B 在 C_AC_B 上的投影区间分别为 $(t_{A\min},t_{A\max})$ 和 $(t_{B\min},t_{B\max})$，若 $t_{A\max}<t_{B\min}$ 或 $t_{B\max}<t_{A\min}$ 时，可判定 2 个凸多面体分离.

2) 碰撞预检测. 具体过程如下: 计算凸多面体 V_A,V_B 的中心 C_A,C_B，计算 C_A 到 V_A 各面和 C_B 到 V_B 各面的最小距离 d_1,d_2，若 $\left|\boldsymbol{C_AC_B}\right|\leqslant d_1+d_2$，则输出凸多面体碰撞.

5.2　基于棱线投影分离的碰撞检测算法

通过有限次的投影运算，即可快速、准确给出凸多面体碰撞检测结果. 本文基于棱线投影分离思想设计了凸多面体碰撞检测流程图，如图 8 所示.

参数的选取. 图 8 中 j_1 和 h_1 代表在 2 个凸多面体中选择的检测面个数，理论上，只要 j_1 和 h_1

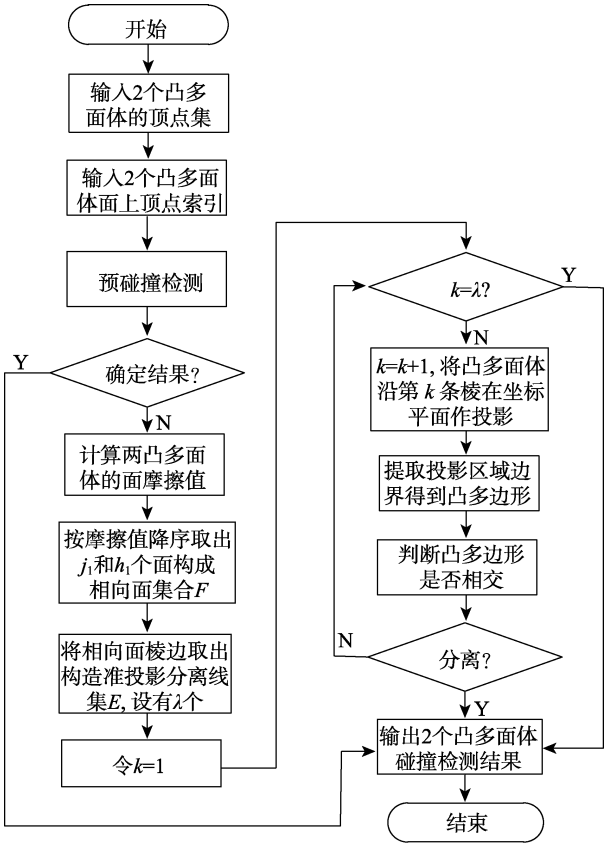


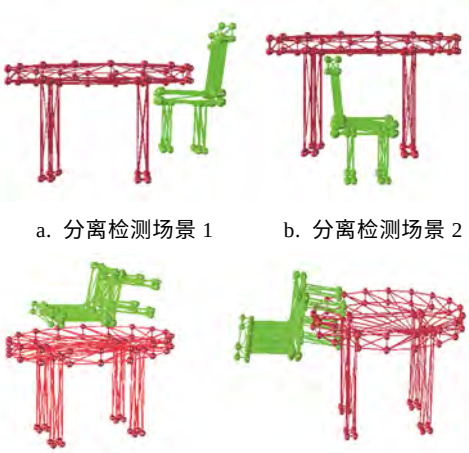
图 8 碰撞检测算法流程图

分别选取到 2 个凸多面体的所有相向面，即可实现分离. 但当凸多面体面数较多时，碰撞多在摩擦系数最大的若干面内发生，根据文献[7]中的结论，并根据本文的实验结果，当 j_1,h_1 取 3~5 时即可保证有效分离. 因此，为进一步加快算法速度，最终 j_1 和 h_1 选取策略如下：当相向面个数小于 5 时，则遍历改凸多面体所有相向面；否则，取 5 个摩擦系数最大的面检测；此外，为了继续加快算法执行速度，根据时间连贯性理论，在实际执行投影分离时可以将上一次有效投影分离线加入此次的准投影分离线中并作优先测试. 另外，无碰撞情况下当需要计算凸多面体最近距离时，可在分离面分离成功时，按照第 4.2 节的方法计算对应的最近距离和最近点对.

6 实 验

本文实验环境为 Windows XP 操作系统，Intel Xeon CPU 2.4GHz，2GB 内存的计算机，利用 3dsMax-2012 建立几何模型并导出数据文件，利用 VC++6.0 开发一套仿真软件，虚拟模型的显示基于 OpenGL 实现[11].

实验 **1.** 任意选择不相交两物体桌子、椅子相对位置如图 9 所示，其中，桌子有 5 个凸多面体，308 个点，624 个面；椅子有 10 个凸多面体，68 个点，132 个面. 依次选取桌子中凸多面体分别与椅子中所有凸多面体作碰撞检测，实验结果表明，本文算法均能快速、准确地给出分离结果.



a. 分离检测场景 1 b. 分离检测场景 2

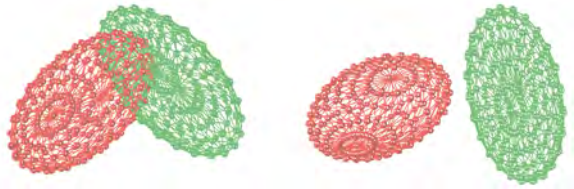
c. 分离检测场景 3 d. 分离检测场景 4

图 9 各种邻近位置检测示意图

本文同时对文献[7]投影分离面法进行各种邻近位置碰撞检测. 实验结果显示，对于绝大多数情况，黎自强算法[7]均能快速而准确地给出分离(或干涉)结果，但对于某些特殊情况(棱相交、棱点相交、点相交等情况)，该算法在准确性方面还有待完善(具体原因见第 7 节).

实验 **2.** 仿照文献[5, 7]，对大小和形状相同的椭球进行剖分(a=500, b=c=400)，得到 2 个不同的凸多面体，如图 10 所示. 按相交，相邻，相距较远各取 20 组位置(方向随机各异)共 6 组凸多面体，每个位置测试 20 次，取平均时间. 文献[7]算法，EGJK 算法，HS-jump 算法和本文算法检测结果如图 11 所示. 其中，本文算法 A，本文算法 B 分别为不考虑、考虑当仅作平移运动时的检测效率.

从图 11 可以看出，本文算法平均时间高于 HS-jump 算法和 EGJK 算法；此外，本文算法和文献[7]算法平均时间同级且均能快速的给出碰撞检测



a. 椭球相交示意图 b. 椭球分离示意图

图 10 椭球碰撞检测示意图

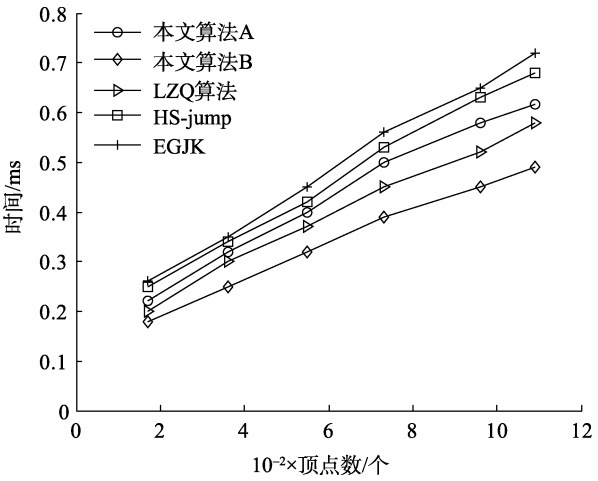
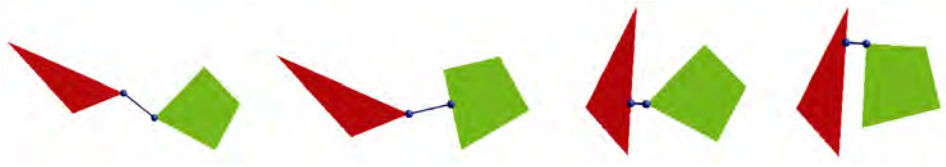


图 11　凸多面体碰撞检测效率比较

结果，但本文算法考虑了凸多面体各种可能碰撞情况，因此平均时间要略低于文献[7]算法时间，当仅考虑凸多面体作平移运动时，本文算法 B 将具有更高的检测效率(分析见第 2.2 节).

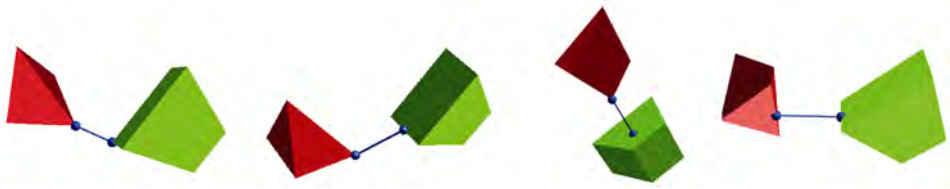
实验 **3.** 选取形状任意的三角形和四边形，五面体和六面体，随机选取 4 种相对位置情况下分别计算最近点，凸多边形、凸多面体顶点坐标、最近点结果如图 12、图 13 所示. 表 1~2 中仅列举了图 12~ 13 中 2 种情况的检测结果.

图 12, 13 中，最近点对用蓝色圆球表示，最近距离用蓝色线段表示. 从表 1~2 以及图 12~13 可以看出，在不同的分离情况下，本文算法均能给出最近点和最近距离.



a. 测试情况 1　　b. 测试情况 2　　c. 测试情况 3　　d. 测试情况 4

图 12　凸多边形最近点计算示意图



a. 测试情况 1　　b. 测试情况 2　　c. 测试情况 3　　d. 测试情况 4

图 13　凸多面体最近点计算示意图

表 1　凸多边形最近点计算结果统计表　　m

测试情况	凸多边形顶点坐标	最近点坐标
1	$A_1(-2.400,1.200)$, $A_2(-1.100,-0.200)$, $A_3(-0.100,0.200)$ $B_1(0.510,-0.283)$, $B_2(1.500,0.707)$, $B_3(1.712,-0.919)$, $B_4(2.207,0.000)$	$P(-0.100,0.200)$ $Q(0.510,-0.283)$
2	$A_1(-2.000,1.200)$, $A_2(-0.700,-0.200)$, $A_3(0.300,0.200)$ $B_1(1.210,-0.417)$, $B_2(0.895,0.946)$, $B_3(2.386,0.264)$, $B_4(1.869,1.172)$	$P(0.300,0.200)$ $Q(1.029,0.368)$

表 2　凸多面体最近点计算结果统计表　　m

测试情况	凸多面体顶点坐标	最近点坐标
1	$A_1(-0.439,1.031,0.000)$, $A_2(-0.587,0.019,0.487)$, $A_3(0.387,0.019,0.487)$, $A_4(-0.587,0.019,-0.487)$, $A_5(0.046,0.019,-0.487)$ $B_1(1.752,1.106,0.342)$, $B_2(1.690,0.565,-0.176)$, $B_3(2.227,-0.027,0.378)$, $B_4(2.289,0.514,0.897)$, $B_5(0.907,-0.125,0.639)$, $B_6(1.837,-0.371,0.784)$, $B_7(0.969,0.416,1.157)$, $B_8(1.899,0.171,1.302)$	$P(0.387,0.019,0.487)$ $Q(0.907,-0.125,0.639)$
2	$A_1(1.248,1.123,1.400)$, $A_2(1.992,0.421,1.887)$, $A_3(2.551,1.219,1.887)$, $A_4(1.992,0.421,0.913)$, $A_5(2.356,0.939,0.913)$ $B_1(2.080,0.882,-0.200)$, $B_2(2.041,0.506,-0.850)$, $B_3(2.452,-0.268,-0.428)$, $B_4(2.491,0.107,0.222)$, $B_5(1.106,-0.280,-0.340)$, $B_6(1.987,-0.659,-0.174)$, $B_7(1.146,0.095,0.310)$, $B_8(2.026,-0.283,0.476)$	$P(1.992,0.421,0.913)$ $Q(1.954,0.058,0.283)$

7 算法分析

本文算法借鉴了黎自强算法[7]的相关思路，因此需要进行两种算法的对比分析.

7.1 投影思路对比分析

1) 黎自强算法[7]将 2 个不相交凸多面体的相向面取出并分别构造成它们的准投影分离面集合，且从这 2 个集合中找到投影分离面(见原文定理 2)，来判断凸多面体分离.

2) 本文提出了投影分离线法，算法构造 2 个不相交凸多面体的准投影分离线集，并从这 2 个集合从找到投影分离线来判断凸多面体分离.

通过对比分析：可以看出，黎自强算法[7]适用于面相交的相对位置情况，对于其他特殊情况未作考虑. 本文指出并严格证明了 2 个不相交的凸多面体必定存在投影分离线，且投影分离线存在于凸多面体相向面的棱边，通过 2 个凸多面体沿着投影分离线投影即可判断分离.

7.2 投影方式对比分析

1) 黎自强算法[7]选取凸多面体部分表面构成准投影分离面，并沿着准投影分离面与坐标平面的交线方向其他坐标平面作投影.

2) 本文选取凸多面体部分棱边构造投影分离线集，并沿着投影分离线向坐标平面作投影.

虽然黎自强算法[7]和本文均采用凸多面体向 2D 坐标平面投影的方法，将 3D 空间凸多面体的相交问题转换为 2D 平面凸多边形的相交问题，但黎自强算法[7]对于向该交线方向投影没有做明确说明.

本文严格证明了投影分离线存在于不相交凸多面体的分离面上(见定理 1)，并继续证明投影分离线存在于凸多面体棱线中，将 2 个凸多面体沿着投影分离线投影即可实现分离.

7.3 投影效果对比分析

1) 黎自强算法[7]执行平均时间与本文算法同级，且略高于本文算法.

2) 本文算法适用于不相交凸多面体的各种相对位置情况，准确性和完整性好于黎自强算法[7].

7.4 时间复杂度分析

分别记 2 个凸多面体的顶点数为 n 和 m，棱边数分别为 λ 和 ξ. EGJK 算法和 HS-jump 算法在凸多面体位置和方向非增量变化的情况下，计算复杂度为 $O(n+m)$，黎自强算法[7]最复杂运算为 $O(n+m)$ 次乘除法，但相交较深和物体比较细长时 HS-jump 算法的效率较低，计算复杂度为 $O(nm)$ 次乘除法，黎自强算法[7]时间复杂度为 $4(n+m+1)$ 次乘除法. 本文算法在最坏情况下执行一次检测需要 $O(\lambda+\xi)$ 次乘除法运算；对于移动对象，由于投影轮廓无需构造，仅需 $O(n+m)$ 次加减运算；对于完全包含或者相距较远的情况，利用中心线投影方式仅需 $O(1)$ 次乘除运算，实验结果可知，本文算法平均检测效率高于 EGJK 算法和 HS-jump 算法，与黎自强算法[7]相当.

8 结 语

本文对凸多面体碰撞检测的投影分离算法进行改进，提出了一种采用棱线投影分离的碰撞检测算法. 该算法是一种具有完备性的碰撞检测新算法，检测效率与 EGJK 和 HS-jump 等主流算法同级，但并非最快(一些不完备算法可能具有更高的效率，可以快速处理部分情况). 该算法采用棱线投影代替平面方向投影，针对凸多面体各种可能碰撞情况均进行了系统分析和严格证明，分析和证明了不相交凸多面体的投影分离线必存在于 2 个凸多面体构成的棱线集中；然后利用相向面及摩擦系数的概念提取分离棱线子集，从而减小搜索范围、提高检测速度；此外，将直线投影分离的思想延用至 2D 凸多边形的碰撞检测，实现将 3D 空间凸多面体碰撞检测问题映射为 2D 直至 1D 问题，极大地简化了算法的实现难度和计算过程，除此之外算法还能计算不相交凸多面体的最近点. 本文算法的最好效果体现在凸多面体不相交或者邻近时的情况，特别当凸多面体仅作平移运动时，由于无需重复计算所有投影坐标值和重新构造多边形投影边界，算法将具有更高的检测效率.

参考文献(References):

[1] Zhang Yingzhong, Fan Chao, Luo Xiaofang. Separating-axis calculation of motion path for continuous collision detection of convex polyhedrons[J]. Journal of Computer-Aided Design & Computer Graphics, 2013, 25(1): 7-14(in Chinese)
(张应中, 范 超, 罗晓芳. 凸多面体连续碰撞检测的运动轨迹分离轴算法[J]. 计算机辅助设计与图形学学报, 2013, 25(1): 7-14)

[2] Gilbert E G, Johns D W, Keerthi S S. A fast procedure for computing the distance between complex objects in three-dimensional space[J]. IEEE Journal of Robotics and Automation, 1988, 4(2): 193-203

[3] Cameron S. A comparison of two fast algorithms for computing the distance between convex polyhedral[J]. IEEE Transactions on Robotics and Automation, 1997, 13(6): 915- 920

[4] Mirtich B. V-Clip: fast and robust polyhedral collision detec-

tion[J]. ACM Transactions on Graphics, 1998, 17(3): 177-208

[5] Ericson C. Real-time collision detection[M]. Beijing: Tsinghua University Press, 2010: 43-293(in Chinese)
(Ericson C. 实时碰撞检测算法技术[M]. 刘天惠，译. 北京: 清华大学出版社, 2010: 43-293)

[6] Li Xueqing, Meng Xiangxu, Wang Jiaye, *et al*. Detecting collision of polytopes using a heuristic search for separating vectors[J]. Chinese Journal of Computers, 2003, 26(7): 837-847 (in Chinese)
(李学庆，孟祥旭，汪嘉业，等. 基于启发式搜索分离向量的凸多面体碰撞检测[J]. 计算机学报, 2003, 26(7): 837- 847)

[7] Li Ziqiang. A fast projection-separation approach for collision detection between polytopes[J]. Journal of Computer-Aided Design & Computer Graphics, 2010, 22(4): 639-646(in Chinese)
(黎自强. 凸多面体快速碰撞检测的投影分离算法[J]. 计算机辅助设计与图形学学报, 2010, 22(4): 639-646)

[8] Wang Yi. Research on key technology of collision detection in virtual reality[D]. Changchun: Jilin University. College of Computer Science & Technology, 2009(in Chinese)
(王　祎. 虚拟现实中碰撞检测关键技术研究[D]. 长春: 吉林大学计算机科学与技术学院, 2009)

[9] Chazelle B, Dobkin D P. Detection is easier than computation[C] //Proceedings of the 12th Annual ACM Symposium on Theory of Computing. New York: ACM Press, 1980: 146-153

[10] Zhou Xin, Zhang Shuyou, Pan Zhigeng. Point in-out polygon test based on coding chain and eigenpolygon[J]. Journal of Computer-Aided Design & Computer Graphics, 2006, 18(9): 1317-1321(in Chinese)
(周　欣，张树有，潘志庚. 基于链码和特征形的多边形内外点判断算法[J]. 计算机辅助设计与图形学学报，2006, 18(9): 1317-1321)

[11] Wright R S, Lipchak B, Jr, Haemel N. OpenGL Super- Bible [M]. 4th ed. Beijing: Posts & Telecom Press, 2010: 50-56(in Chinese)
(Wright R S, Lipchak B, Jr, Haemel N. OpenGL 超级宝典[M]. 张　琪，译. 4 版. 北京: 人民邮电出版社, 2010: 50-56)