计算机科学
COMPUTER SCIENCE

基于轮廓线的网格体物体空间快速消隐算法

宋海川, 邱荪泓, 汪鑫星, 李一锦, 陈振华, 陈小雕

引用本文

宋海川, 邱荪泓, 汪鑫星, 李一锦, 陈振华, 陈小雕. 基于轮廓线的网格体物体空间快速消隐算法[J]. 计算机科学, 2025, 52(4): 222-230.

SONG Haichuan, QIU Sunhong, WANG Xinxing, LI Yijin, CHEN Zhenhua, CHEN Xiaodiao. Fast Contour-based Object-space Hidden Line Removal Algorithm for Mesh [J]. Computer Science, 2025, 52(4): 222-230.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240700042

基于轮廓线的网格体物体空间快速消隐算法

宋海川[1] 邱荪泓[1,2] 汪鑫星[1] 李一锦[3] 陈振华[1] 陈小雕[4]
1 华东师范大学计算机科学与技术学院 上海 200062
2 先进计算与关键软件(信创)海河实验室 天津 300450
3 南京大学计算机学院 南京 210023
4 杭州电子科技大学计算机学院 杭州 310018

摘 要 隐藏线消除,即消除在一定视角下被遮挡的线段,是解决三维场景中视觉混淆问题的关键技术。其中,物体空间的隐藏线消除技术可以计算可见性变化点的精确坐标,因此在实际工程中被广泛用于三维可视化建模、高精度图纸的绘制等。然而,先前的物体空间消隐算法在处理实际工程中常用的网格体模型时,往往会因为模型面内部含有大量三角面片而计算效率低下。因此,提出了基于轮廓线的网格体物体空间快速消隐算法。该算法通过网格体轮廓线投影的交集进行三角面片的筛选及其求交计算,从而避免了大部分的无效求交计算。同时,在求交后根据待定可见性变化点所在线段与轮廓线和模型的射入、射出情况进行快速可见性判断,进一步提高了算法效率。实验结果显示,在两种常见的消隐模式下处理普通和复杂网格体模型的消隐,所提算法相较于对比算法效率分别提高了20倍和80倍以上,与主流几何内核ACIS的消隐处理效率差距在2.5倍以内。

关键词:消隐算法;物体空间消隐;网格体;轮廓线;CAD
中图分类号 TP391.72

Fast Contour-based Object-space Hidden Line Removal Algorithm for Mesh

SONG Haichuan[1],QIU Sunhong[1,2],WANG Xinxing[1],LI Yijin[3],CHEN Zhenhua[1] and CHEN Xiaodiao[4]
1 School of Computer Science and Technology,East China Normal University,Shanghai 200062,China
2 Haihe Laboratory of Information Technology Application Innovation,Tianjin 300450,China
3 School of Computer Science,Nanjing University,Nanjing 210023,China
4 School of Computer Science,Hangzhou Dianzi University,Hangzhou 310018,China

Abstract Hidden line removal,which eliminates lines occluded under certain viewing angles,is a key technique for addressing visual clutter issues in 3D scenes. Object-space hidden line removal techniques can calculate the precise locations of visibility transformation points,making them widely used in practical engineering for 3D visualization modeling,high-precision drawing,and other purposes. While there are many mature object-space hidden line removal algorithms available for planar polyhedra,these algorithms often suffer from low computational efficiency when handling commonly used mesh models in practical engineering due to the large number of triangles within model surfaces. To address this issue,this paper proposes a fast contour-based object-space hidden line removal algorithm for mesh. This algorithm filters triangular facets based on the intersection of mesh object contour line projections and performs intersection calculations,thereby avoiding most redundant intersection computations. Additionally,after intersection calculations,the algorithm rapidly determines visibility based on the line segments where potential visibility transformation points lie in relation to the contour lines and model,further enhancing efficiency. Experimental results show that when processing the hidden line removal of ordinary and complex mesh models in two common hidden line removal modes,the efficiency of the algorithm presented in this paper is over 20 times and 80 times higher,respectively,than compared algorithm,and the efficiency difference between our algorithm and the mainstream geometric kernel ACIS is within 2.5 times.

Keywords Hidden-line removal algorithm,Object-space hidden-line removal,Mesh,Contour lines,CAD

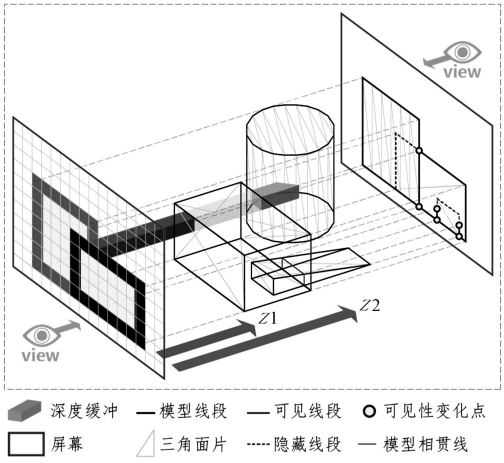
1 引言

在计算机辅助设计(Computer-Aided Design,CAD)中,绘制模型线框图时消除被遮挡线的步骤,被称为隐藏线消除(Hidden-line Removal),或简称为消隐[1]。消隐在实际工程中可以为几何造型、仿真模拟、图纸绘制等环节提供可靠的真实感图形,为产品的后续设计与制造提供保障。

如图1所示,在CAD系统中,通常将模型离散化得到的

到稿日期:2024-07-08 返修日期:2024-09-13
通信作者:宋海川(hcsong@cs.ecnu.edu.cn)

网格体(Mesh)和由用户指定的观察视角(view)共同作为消隐计算的输入。消隐计算分为物体空间的消隐和图像空间的消隐[2-3]。对于图像空间的消隐，首先根据 view 对网格体进行投影变换，然后对变换后的图形进行光栅化，最后通过对屏幕进行逐像素分析，确定当前模型的哪些像素被其他模型的像素遮挡，从而在图像上实现不可见部分的隐藏。Z-buffer[1]是该类消隐的代表性算法，该算法通过深度缓冲来对比像素深度值。如图 1 所示，深度缓冲存放每个像素的深度值，其中 $Z1$ 和 $Z2$ 是对应像素点处两对象的深度值，且 $Z2$ 绝对值大于 $Z1$ 绝对值，故取 $Z1$ 对应的像素在屏幕显示。然而，图像空间的消隐计算精度受到像素分辨率的限制，通常无法满足工程制图的精度要求，因此该类方法主要用于动态场景[4]和复杂场景[5]的消隐等近似消隐的计算。



注：左侧屏幕为图像空间的消隐结果，右侧屏幕为物体空间的消隐结果。

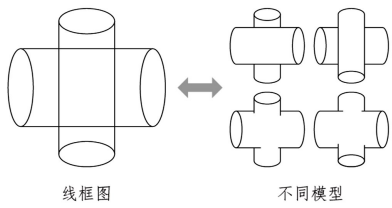
图 1 图像空间和物体空间消隐

Fig. 1 Image-space and object-space hidden line removal

对于物体空间的消隐，根据 view 对网格体进行投影变换后，不需要进行光栅化处理，而是对其中的几何对象进行分析和处理，从而得到对应视角下的可见性变化点(如图 1 圆圈所示)的坐标，并进一步得到消隐结果。其精度为用户指定的几何容差。因此，在实际工程制图中主要使用物体空间的消隐计算。

2 相关工作

模型在给定 view 下的二维线框图不具有深度信息，往往会导致图像的多义性，如图 2 所示。为了消除模型线框图的多义性，需要进行物体空间的消隐。物体空间的消隐输入如图 1 所示，view 由视点位置和指向观察方向的单位向量定义，原始模型的边离散结果集合 E 使用黑色线段表示，离散模型的三角面片边集合 D 使用灰色线段表示。因此，$L=E\cup D$ 为离散模型的全部边集合，L 总元素个数为 n。L 根据观察视角 view 进行投影变换后的集合为 P_L，P_L 中所有元素之间的二维交点个数为 k。离散模型上所有三角面的集合为 T，元素个数为 p，易得 $3p=2n$。T 根据 view 进行投影变换的集合为 P_T，P_T 元素和 P_L 元素之间的二维交点个数为 r，在三维空间中的交点个数为 i。网格体模型的个数为 m，网格顶点总数为 v。



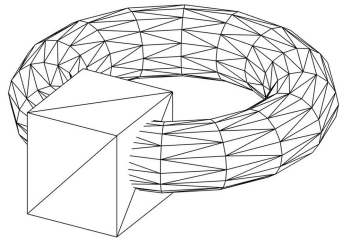
注：不同模型可能具有相同的投影线框图，因此对于同一线框图可能有多种理解，造成歧义。

图 2 线框图的二义性

Fig. 2 Ambiguity of wireframe

在前人的工作中，Roberts[6]提出了基于视线的边和多边形裁剪消隐算法，但该方法严格要求所有模型都是凸的，即模型不包含凹陷或孔洞。Weiss[7]突破了这一限制，并且提出了处理平面和二次曲面组合体的模型消隐的方法。Weiler[8]进一步提出了基于投影求交的多边形和多边形裁剪消隐算法。Appel 等[9-11]提出了将多边形的边互相求交，从而划分出细分线段，再用细分线段逐一对多边形面计算可见性的方法。Devai[12]分析了该类方法在处理 n 条边的消隐数据时，在最坏的情况下时间复杂度为 $O(n^3)$，并提出了复杂度为 $O(n^2)$ 的算法。但是，该算法首先将每个输入多边形的边延伸为直线，再在投影平面上处理这些直线的交点，因此即使在交点较少的情况，时间开销也为 $O(n^2)$[13]。Ottmann 等[14-15]提出了时间复杂度为 $O((n+k)\log^2 n)$ 的扫描线算法。Nurmi[16]提出了改进的扫描线算法，将时间复杂度降为 $O((n+k)\log n)$，但是该算法的运行效率在交点数量较大时整体运行效率较低。Goodrich[17]提出了基于多边形排列的消隐方法，构造多边形排列图并进行求交计算和可见性判断，在最坏情况下的时间复杂度为 $O(n\log n+k+r)$，但该方法只适用于模型在三维空间中不相交的情况。Hsu 等[18]提出了可以处理模型在三维中存在相交情况的方法，该方法先将所有的多边形进行循环求交，再将交线也加入需要判断可见性的线段列表中，进而得到所有线段的可见性判断。Song 等[19]对凸多面体线消隐算法进行改进，加入了包围盒检测和深度优先排序，提高了凸多面体消隐的效率。Xu 等[20]提出了容差可控的矢量裁剪算法，实现了容差可控的线消隐算法。

上述方法在处理多个由大量三角面片组成的模型消隐时，往往会产生大量的时间开销，且部分算法无法处理工业中常出现的如图 3 所示的模型相交情况下的消隐。



注：其中圆环穿过了正方体，其处于正方体内部的离散边不可见。

图 3 相交的网格体模型

Fig. 3 Intersecting mesh model

3 基于轮廓线的网格体物体空间快速消隐算法

3.1 算法概述

针对上述问题，本文提出基于轮廓线的网格体物体空间快速消隐算法，其时间复杂度为：

$$O(s^2+ef+c+n) \tag{1}$$

如图 4 所示，各轮廓线的边集合 S 用黑色粗实线表示，S 中元素个数为 s，$s \ll n$；轮廓线交集 I_S 用灰色粗实线表示，I_S 中元素个数为 j；轮廓线交集涉及的边集合 I_E 为灰色粗虚线范围内的黑色细线，I_E 中元素个数为 e，$j<e<n$；轮廓线交集涉及的三角面片集合 I_T 为灰色粗虚线范围内的三角形，I_T 中元素个数为 f，$f<p$；待定可见性变化点集合 C 用黑色点和灰色点表示，其中黑色点是真实的可见性变化点，灰色点处的可见性并没有发生变化。C 中元素总数为 c，$c<k+i<k+r$。本文算法的突出贡献在于将时间复杂度中的主要项 $O(n^2)$ 降为 $O(s^2)$。

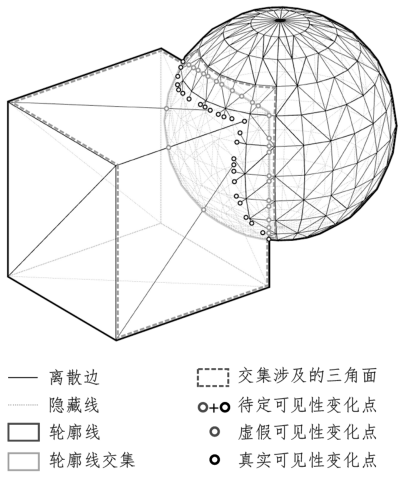


图 4 本文算法数据结构示意图

Fig. 4 Data structure of the propose algorithm

本文算法的具体步骤如算法 1 所示。

算法 1 基于轮廓线的网格物体空间快速消隐算法

输入：由 B-rep 体离散而来的网格体组成数组 $Meshes_{in}$

输出：绑定了消隐的网格体数组 $Meshes_{out}$

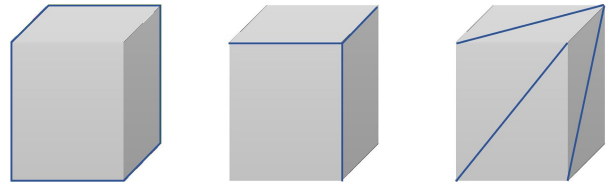
1. 构造阶段
   1.1. 遍历三角面片，进行背面剔除；
   1.2. 轮廓线段搜索。
2. 求交与可见性判断阶段
   2.1. 构建轮廓线交集；
   2.2. 筛选可能相交的三角面片；
   2.3. 获取待定可见性变化点；
   2.4. 求解三角边可见性
3. 消隐数据绑定阶段

为了简化表述，后文用线段和三角边来代指 L 的元素。

3.2 构造阶段

3.2.1 阶段概述

如图 5 所示，该阶段需要构造的 3 个线段集为：1）用于求消隐数据的轮廓线段集 S；2）需要求消隐数据的边缘（Edge）上的离散线段集 E；3）需要求消隐数据的面（Face）内部的离散线段集 D。



(a)轮廓多边形　(b)需要求消隐数据的边缘上的离散线段　(c)需要求消隐数据的面内部的离散线段

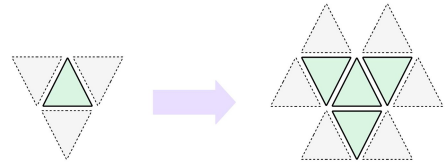
图 5 线段集示意图

Fig. 5 Line segment sets diagram

该阶段流程的具体说明如下：首先，广度搜索遍历三角面片，遍历过程中，求解并记录三角面片的面向性，同时进行背面剔除；然后，通过邻接三角面片面向性是否相同来判断邻接边是否为轮廓线段并记录，若为轮廓线段，则搜索与当前轮廓线段相接的下一条轮廓线段，直到组成一个完整的轮廓线多边形。

3.2.2 广度搜索遍历三角面片

通过图 6 所示的方式来广度搜索遍历网格的三角面片。首先，将未被遍历过的最小编号三角面片入队。然后，队首三角面片出队，取与之邻接的未被遍历过的三角面片入队，若邻接三角面片都被遍历过，则重复第一步操作。



注：实线三角形为出队三角面片，虚线三角形为入队三角面片。

图 6 广度搜索遍历三角面片示意图

Fig. 6 Diagram of breadth-first search traversal of triangles

在三角面片入队过程中，求其面向性，并记录到其属性中。若该三角面片与刚出队的三角面片的面向性不同，说明两者的邻接边为轮廓线段。如图 7 所示，将该邻接边的终点加入轮廓线段搜索队列，开始轮廓线段搜索。图中黑色实线包含的三角面片面向屏幕，黑色虚线包含的三角面片背向屏幕，则连接两者的三角边为轮廓线段，即灰色实线。

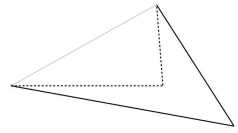


图 7 轮廓线段定义示意图

Fig. 7 Diagram of contour line segment definition

3.2.3 轮廓线段搜索

通过图 8 所示的方式进行轮廓线段搜索。从第一条找到的轮廓线段的结束端点开始广度搜索与其相连的所有三角边，并判断是否为轮廓线段：对该三角边连接的两个三角面片的面向性进行判断，并记录到各自的属性中，然后更新未被遍历过三角面片的最小编号。若各自属性中已记录了面向性，则直接取两者进行判断。若为轮廓线段，则将其记录

到当前轮廓线的轮廓线段列表，并从该轮廓线段的结束端点开始，重复图 8 的操作，直到找到的轮廓线段的结束端点等于第一条轮廓线段的起始端点，即轮廓线形成一个封闭的多边形，如图 9 所示。Elber 等[21]论证了曲面的轮廓线必为封闭的曲线，可以用相似的证明方式证明三维网格的轮廓线必为封闭多边形。

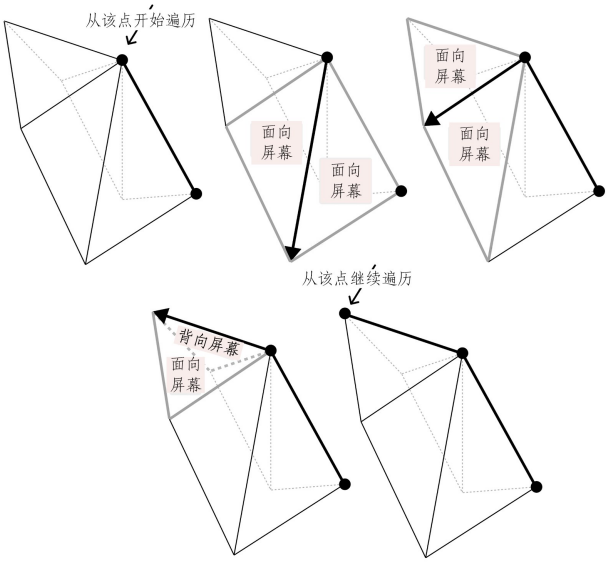
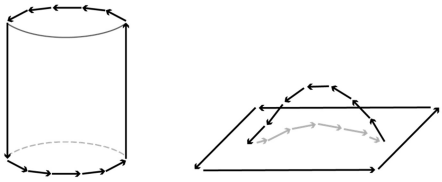


图 8 轮廓线段搜索过程示意图

Fig. 8 Diagram of contour line segments search process



注：箭头表示轮廓线段，轮廓线段组成的轮廓线必为封闭多边形。

图 9 轮廓线段多边形示意图

Fig. 9 Diagram of contour line segments polygon

考虑网格体 M 上的点 p，将在模型面上由三角边组成的不穿过任何轮廓段的连续路径到达的 p 点组成的集合记为 R_p。如果存在点 $q\in M$，使得 $q\notin R_p$，那么点 q 组成的集合记为 R_q。显然，$R_p\cap R_q=\emptyset$，因为如果存在点 $r\in R_p\cap R_q$，那么存在一条从 p 到 r 的连续路径，以及从 r 到 q 的路径，进而存在从 p 到 q 的路径，而这与选择 q 的方式矛盾。将这些路径连接的区域集合称为 G。G 的边缘即为轮廓线段组成的多边形，因此三维网格的轮廓线必为封闭多边形。

3.3 求交与可见性判断阶段

3.3.1 阶段概述

该阶段需要使用轮廓线之间的二维交集筛选可能导致线段可见性变化的三角面片，然后求交并判断交点是否为可见性变化点，最后通过可见性变化点来求解可见性，并将可见性信息记录到对应三角边的属性中。

该阶段流程的具体说明如下：首先，将两个轮廓线 A 和 B 的投影进行二维求交，获得两个轮廓线的投影交集；然后，取交集的一个交点，从交点所属的三角面片开始广度搜索遍历，分别从 A 和 B 中筛选出与轮廓线投影交集有重合部分的三角面片；再将 A 中筛选出的三角面片与 B 中筛选出的三角面片循环求交，得出待定可见性变化点；最后，利用待定可见性变换点求解三角边的可见性。

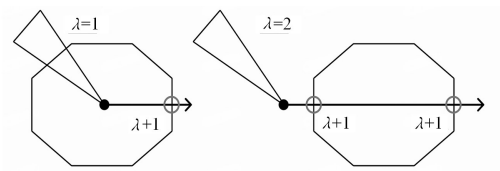
3.3.2 构建轮廓线交集

取其中一个网格的轮廓线 A（下述的三角面片都是在背面剔除后留下来的三角面片），将轮廓线 A 与另一个轮廓线 B 求交，获得交点，将交点分别记录到所属的三角边上，并同时记录到临时轮廓交点列表。

3.3.3 筛选可能相交的三角面片

取临时轮廓交点列表中 y 值最大的交点，取该交点所属的 B 中的三角面片，对该三角面片进行下述操作。

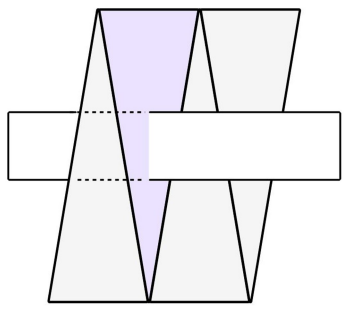
首先，将该三角面片加入需要求交的三角面片列表。取该三角面片的 3 条三角边，并取其所属的另一个三角面片的面向性，若与当前三角面片相同，则将其加入判断列表。然后，依次取判断列表中的三角面片，在二维上判断其是否有点在轮廓线 A 的内部（使用射线法判断，如图 10 所示）或有三角边跨立了轮廓线 A（为了处理图 11 所示情况）。若有，则将该三角面片加入“轮廓线可能穿过的三角面片列表 L_A”，并将该三角面片重新输入本模块。最后，重复上述步骤，但将轮廓 A 和轮廓 B 交换，用于构造“边的可见性可能被轮廓 B 中的三角面片改变的三角面片列表 L_B”。筛选出来的组成两个列表的三角面片即为集合 I_T。



注：令点向右发射一条射线，求该射线与轮廓线的交点数，若为奇数，说明点在轮廓线内部，若为偶数，说明点在轮廓线外部。

图 10 射线法示意图

Fig. 10 Ray casting method diagram



注：白色矩形表示的多边形的所有点都在平行四边形表示的网格体的轮廓线的外部，但事实上多边形的可见性在灰色三角面片处发生了变化。

图 11 三角面片特殊相交情况示意图

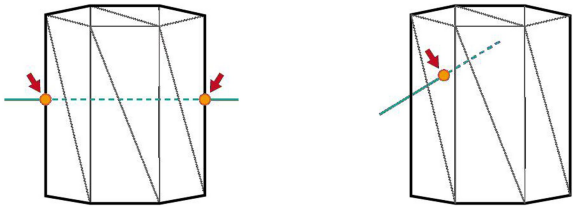
Fig. 11 Diagram of special intersection case of triangles

3.3.4 获取待定可见性变化点

可见性变化的点只会发生在如图 12 所示的两处。

第一种可见性变化点如图 12(a)所示，可见性变化点在与轮廓线二维相交处（交点在三维上可能分离），可见性

变化点之间的线段或线段的部分被整个物体遮挡；第二种可见性变化点如图 12(b)所示，线段部分穿过物体，可见性发生变化的点在与三角面片三维相交处。综上，线段的可见性变化只可能发生在交点处，两交点之间的可见性是相同的，因此，待定可见性变化点即 L_A 的元素与 L_B 的元素的三角边间的交点和已经求出的轮廓线交点，在该步骤通过循环求交即可获得。



(a)第一种可见性变化点 (b) 第二种可见性变化点

图 12 箭头所指点即为可见性变化点

Fig. 12 Visibility change points by the arrow

3.3.5 求解三角边可见性

求出两轮廓线的待定可见性变化点后，可以根据下文所述方法按顺时针方向依次求轮廓线各段可见性。以一平行四边形轮廓线对圆柱体的可见性求解为例，进行可见性求解方法的描述。

如图 13 所示，黑色点表示第一种可见性变化点的待定可见性变化点，灰色点表示第二种可见性变化点，圆圈表示可见性判断辅助点。求解浅灰色实线段代表的第一段的可见性后，其他段的可见性求解方式相同。从待定可见性变化点中 y 值最大的点 1 开始，取平行四边形轮廓线中交点 1 所在的浅灰色实线段，与圆柱轮廓线中点 1 所在的深灰色粗线表示的三角边所属的三角面片在二维上求交，获得可见性判断辅助点 a。取 a 与 1 的中点，向屏幕发射一条射线，判断射线是否穿过深灰色边三角面片。若是，则表明交点 1 和 A 之间的轮廓线段被遮挡，即线段 $1-A$ 不可见；否则表明未被遮挡，线段 $1-A$ 可见。用同样的方法即可判断待定可见性变化点 A 和 2 之间、3 和 B 之间、B 和 4 之间的线段的可见性。

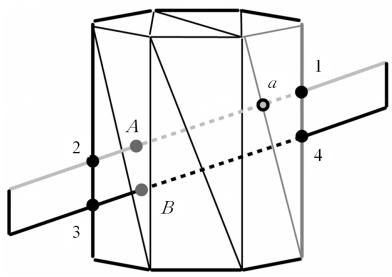


图 13 轮廓线段的可见性信息求解示意图

Fig. 13 Diagram of solving the visibility information of contour line segments

当求在边缘上和在面内部的离散线段的可见性时，如图 14 所示，将一条边视为一个完整的轮廓线，黑色点表示可能是第一种可见性变化点的待定可见性变化点，灰色点表示第二种可见性变化点，圆圈表示可见性判断辅助点，即可参考轮廓线各段可见性求解的方法来利用待定可见性变化点求解边的可见性。

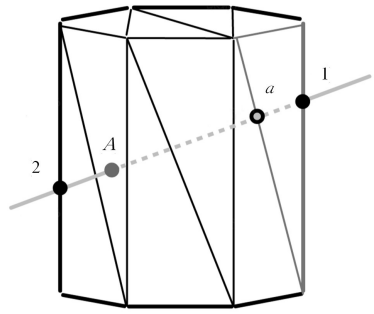


图 14 边缘上和面内部的离散线段的可见性求解示意图

Fig. 14 Diagram of solving the visibility of line segments on edges and inside faces

3.4 绑定消隐数据阶段

遍历所有网格的用于求消隐数据的轮廓多边形、需要求消隐数据的边缘上的离散线段、需要求消隐数据的面内部的离散线段，将对应的消隐数据绑定到对应的网格体上。

4 实验验证

4.1 消隐效率分析

本文实验对比对象之一为根据 Janssen 等[22-23]提出的应用于工程实际的消隐算法而实现的基础算法。本节将对基础算法与本文算法在各阶段的效率分别进行对比。在构造阶段，基础算法遍历集合 T 和 L 中的元素并求其二维投影来构造集合 P_T 和 P_L，因此其时间复杂度为 $O(p+n)=O(n)$。本文方法取消了单独遍历 L 的元素并求其二维投影的环节，时间复杂度为 $O(p)=O(n)$，因此该阶段两种算法的时间复杂度总体上相同。

在求交与可见性判断阶段，基础算法首先通过线线二维求交、线面三维求交获得可见性一致的细分线段：将集合 P_L 的元素之间进行二维求交计算，再将 P_L 与 P_T 的元素进行三维求交计算，交点之间即为细分线段，细分线段内任意两点的可见性一致。这一步的时间复杂度为 $O((n+p)n)=O(n^2)$。然后，每条细分线段对每个三角面判断可见性，时间复杂度为 $O((k+i)p)=O((k+i)n)$。最后，遍历细分线段，将相邻可见性相同的细分线段合并(merge)，时间复杂度为 $O(k+i)$。因此该阶段基础算法的时间复杂度为：

$$O(n^2+(k+i)n) \tag{2}$$

在该阶段，本文方法构建的轮廓线交集的时间复杂度为 $O(s^2)$；筛选可能相交的三角面片的时间复杂度为 $O(jf)$，获取待定可见性变化点的时间复杂度为 $O(ef)$，又因为 $I_S \in I_E$，$j<e$，所以两项可以合并为 $O(ef)$；求解三角边可见性的时间复杂度为 $O(c)$。因此，该阶段本文算法的时间复杂度为：

$$O(s^2+ef+c) \tag{3}$$

在消隐数据绑定阶段，本文算法与基础算法的时间复杂度相同，为 $O(p+n)=O(n)$。

综上，本文方法的时间复杂度为：

$$O(s^2+ef+c+n) \tag{4}$$

4.2 消隐实验结果

实验的硬件环境为 12th Gen Intel(R)CPU i7-10870H 2.20GHz，RAM 为 16 GB，显卡为 NVIDIA GeForceRTX 3060 Laptop GPU，显存为 14GB。软件系统环境为 Win11，编程语言使用 C++，C++语言标准为 ISOC++ 17。使用的 B-rep 测试模型通过调用行业领先的几何内核 ACIS[24] 接口构建，测试对比网格数据来自 ACIS 离散模块的离散网格结果。耗时测试实验中采用循环测试 100 次的平均值作为结果，网格使用 OpenGL 渲染得到可视化结果。

实验对比对象为基础算法[22-23]和 ACIS 的消隐。实验对比了两种模式的消隐，模式 1 不计算面(Face)内离散三角边的消隐数据，模式 2 计算面(Face)内离散三角边的消隐数据。为了简化表述，下文成对出现的可视化结果中，默认第一个为模式 1 的消隐结果，第二个为模式 2 的消隐结果。按三角面片数将测例分为 3 类模型：测例 1 为三角面片数不超过 100 的简单模型；测例 2 到测例 5 为三角面片数不超过 10 000 的普通模型；测例 6 到测例 8 为三角面片数超过 50 000 的复杂模型。

测例 1 为 C 形方槽模型，能验证本文算法是否能正确处理模型轮廓线在二维投影存在自交的情况的消隐。实验输入以及结果如表 1 和表 2 所列，可视化结果如图 15 所示。

表 1 测例 1 实验输入数据

Table 1 Test case 1 experimental input data

数据名称	数目
三角面片顶点数 v	16
三角面片边数 n	42
三角面片数 p	28
模式 1 输出消隐线段数	24
模式 2 输出消隐线段数	43

表 2 测例 1 实验结果数据

Table 2 Test case 1 experimental result data

(ms)

	模式 1 结果	模式 2 结果
基础算法[22-23]	0.69	1.88
ACIS	0.08	1.85
本文算法	0.47	1.49

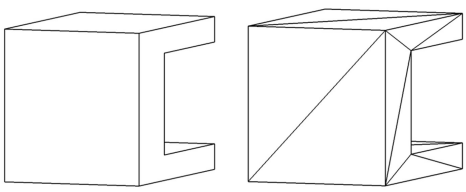


图 15 测例 1 可视化结果

Fig. 15 Test case 1 visualization result

实验结果表明，本文算法能正确快速地处理存在此类情况的模型的消隐。

测例 2 为圆环模型，能验证本文算法是否能处理模型存在多条轮廓线且轮廓线二维投影间存在包含关系的情况的消隐。实验输入以及结果如表 3 和表 4 所列，可视化结果如图 16 所示。

表 3 测例 2 实验输入数据

Table 3 Test case 2 experimental input data

数据名称	数目
三角面片顶点数 v	1008
三角面片边数 n	3024
三角面片数 p	2016
模式 1 输出消隐线段数	122
模式 2 输出消隐线段数	3079

表 4 测例 2 实验结果数据

Table 4 Test case 2 experimental result data

(ms)

	模式 1 结果	模式 2 结果
基础算法[22-23]	46.02	4307.92
ACIS	0.81	37.57
本文算法	2.42	39.85

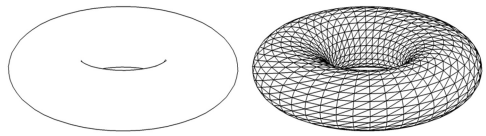


图 16 测例 2 可视化结果

Fig. 16 Test case 2 visualization result

实验结果表明，本文算法能正确快速地处理存在此类情况的模型的消隐。

测例 3 由一大一小两个球通过布尔并获得，能验证本文算法是否能正确地处理模型存在布尔运算的情况的消隐。实验输入以及结果如表 5 和表 6 所列，可视化结果如图 17 所示。

表 5 测例 3 实验输入数据

Table 5 Test case 3 experimental input data

数据名称	数目
三角面片顶点数 v	897
三角面片边数 n	2685
三角面片数 p	1790
模式 1 输出消隐线段数	128
模式 2 输出消隐线段数	2723

表 6 测例 3 实验结果数据

Table 6 Test case 3 experimental result data

(ms)

	模式 1 结果	模式 2 结果
基础算法[22-23]	38.56	3560.16
ACIS	0.79	31.42
本文算法	2.84	46.82

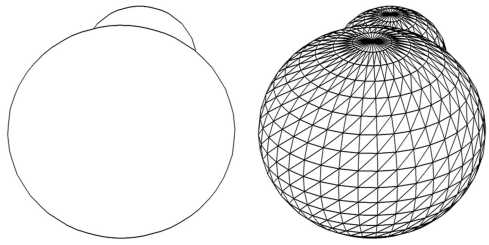


图 17 测例 3 可视化结果

Fig. 17 Test case 3 visualization result

实验结果表明，本文算法能正确快速地处理存在此类情况的模型的消隐。

测例 4 为由多个物体组成的场景，能验证本文算法能否正确地处理多个模型间存在相交的情况的消隐。实验输入以及结果如表 7 和表 8 所列，可视化结果如图 18 所示。

表 7 测例 4 实验输入数据

Table 7 Test case 4 experimental input data

数据名称	数目
三角面片顶点数 v	2744
三角面片边数 n	8208
三角面片数 p	5472
模式 1 输出消隐线段数	245
模式 2 输出消隐线段数	5239

表 8 测例 4 实验结果数据

Table 8 Test case 4 experimental result data

(ms)

	模式 1 结果	模式 2 结果
基础算法[22-23]	135.07	1204.02
ACIS	1.89	89.42
本文算法	3.19	179.82

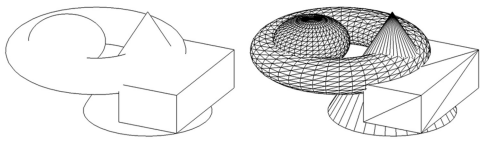


图 18 测例 4 可视化结果

Fig. 18 Test case 4 visualization result

实验结果表明，本文算法能正确快速地处理存在此类情况的模型的消隐。

测例 5 为复杂零件模型，包含一个带有孔洞的零件主体模型、位于主体中心的转轴模型和一些附属的螺钉模型，模型之间进行了布尔并。实验输入以及结果如表 9 和表 10 所列，可视化结果如图 19 所示。

表 9 测例 5 实验输入数据

Table 9 Test ase 5 experimental input data

数据名称	数目
三角面片顶点数 v	3765
三角面片边数 n	11265
三角面片数 p	7510
模式 1 输出消隐线段数	1425
模式 2 输出消隐线段数	12532

表 10 测例 5 实验结果数据

Table 10 Test case 5 experimental result data

(ms)

	模式 1 结果	模式 2 结果
基础算法[22-23]	1963.56	69588.52
ACIS	13.95	50.86
本文算法	24.26	111.16

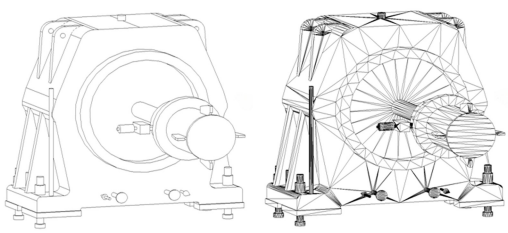


图 19 测例 5 可视化结果

Fig. 19 Test case 5 visualization result

测例 6 为船舶工作平台模型，包含护栏和梯子等结构，底部包含一些管道和阀门模型。实验输入以及结果如表 11 和表 12 所列，可视化结果如图 20 所示。

表 11 测例 6 实验输入数据

Table 11 Test case 6 experimental input data

数据名称	数目
三角面片顶点数 v	38941
三角面片边数 n	116955
三角面片数 p	77970
模式 1 输出消隐线段数	9939
模式 2 输出消隐线段数	185223

表 12 测例 6 实验结果数据

Table 12 Test case 6 experimental result data

(ms)

	模式 1 结果	模式 2 结果
基础算法[22-23]	364865.82	7930416.33
ACIS	1876.44	7825.44
本文算法	4166.47	19107.12

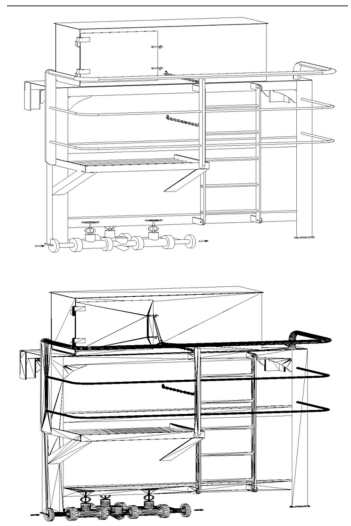


图 20 测例 6 可视化结果

Fig. 20 Test case 6 visualization result

测例 7 为分油机等船舶设备模型，模型包含多个分油机模型及其配套的管路和控制设备模型，以及在主体之上的两个挂钩模型。实验输入以及结果如表 13 和表 14 所列，可视化结果如图 21 所示。

表 13 测例 7 实验输入数据

Table 13 Test case 7 experimental input data

数据名称	数目
三角面片顶点数 v	36086
三角面片边数 n	107934
三角面片数 p	71956
模式 1 输出消隐线段数	11751
模式 2 输出消隐线段数	117384

表 14 测例 7 实验结果数据

Table 14 Test case 7 experimental result data

(ms)

	模式 1 结果	模式 2 结果
基础算法[22-23]	280360.45	6661341.76
ACIS	590.96	3616.04
本文算法	1368.04	7533.70

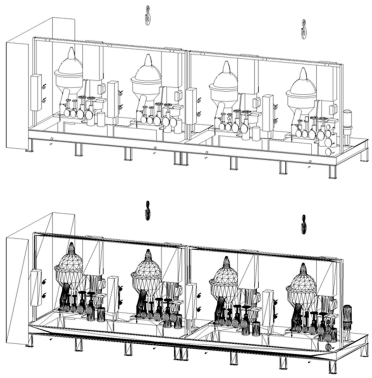


图 21 测例 7 可视化结果

Fig. 21 Test case 7 visualization result

测例 8 为船舶机舱中的燃油分油机系统模型，模型包含多个分油机模型、相关管道和控制设备模型，以及支撑平台、通道和梯子等维护结构模型。实验输入以及结果如表 15 和表 16 所列，可视化结果如图 22 所示。

表 15 测例 8 实验输入数据

Table 15 Test case 8 experimental input data

数据名称	数目
三角面片顶点数 v	82671
三角面片边数 n	248913
三角面片数 p	165942
模式 1 输出消隐线段数	75999
模式 2 输出消隐线段数	213540

表 16 测例 8 实验结果数据

Table16 Test case 8 experimental result data

(ms)

	模式 1 结果	模式 2 结果
基础算法[22-23]	1602490.68	42059711.87
ACIS	4976.89	13477.54
本文算法	10437.64	29325.52

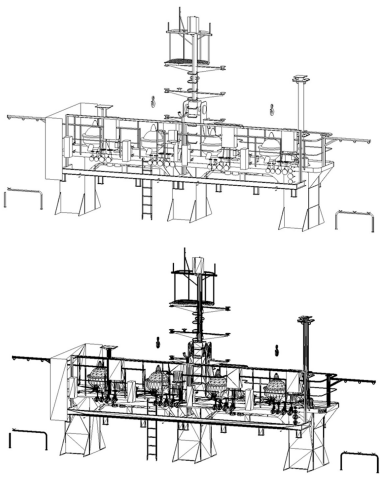


图 22 测例 8 可视化结果

Fig. 22 Test case 8 visualization result

综上，在两种常见的消隐模式下处理普通和复杂网格体模型的消隐，本文算法相较于对比算法效率分别提高 20 倍和 80 倍以上，与主流几何内核 ACIS 的消隐处理效率差距在 2.5 倍以内。

结束语 在计算机图形学领域，隐藏线消除技术是解决三维场景中视觉混淆问题的关键技术。本文提出了一种基于轮廓线多边形的物体空间快速消隐算法。

本文算法的关键在于构造阶段和求交与可见性判断阶段的合理设计。通过广度搜索遍历三角面片和轮廓线段搜索等高效的算法设计，能够快速识别出轮廓线多边形和需要消隐数据的线段，提高了算法的效率。同时，通过构建轮廓线交集，筛选可能相交的三角面片，以及利用待定可见性变化点求解可见性等步骤，能够快速准确地求解可见性变化点，并进一步求解三角边的可见性信息，实现了对网格体快速精确的消隐处理。后续可考虑用 BVH 树等空间层次结构来加速筛选可能相交的三角面片，进一步加速消隐计算。

参 考 文 献

[1] SUN J G，HU S M. Fundamental Course in Computer Graphics [M]. Beijing：Tsinghua University Press，2009.

[2] SUTHERLAND I E，SPROULL R F，SCHUMACKER R A. A characterization of ten hidden-surface algorithms [J]. ACM Computing Surveys(CSUR)，1974，6(1)：1-55.

[3] JIN H L，GAO J X. A Summary of Algorithms for Removing the Hidden Lines and Surfaces [J]. Computer and Digital Engineering，2006，34(9)：27-31.

[4] GUO H，FU H G，LUO D H. Realization of hidden line removal in 3D dynamic geometry [J]. Journal of Computer Applications，2007，27(3)：663-665.

[5] LUO G L，WANG R，WU H，et al. Fast hidden line removal method for large-scale 3D substation scene model based on Z-buffer algorithm optimization [J]. Journal of Graphics，2021，42(5)：775.

[6] RUOBERTS L G. Machine perception of three-dimensional solids[D]. Massachusetts：Massachusetts Institute of Technology，1963.

[7] WEISS R A. BE VISION，a package of IBM 7090 FORTRAN programs to draw orthographic views of combinations of plane and quadric surfaces[M]// Seminal graphics：pioneering efforts that shaped the field. 1998：7-17.

[8] WEILER K，ATHERTON P. Hidden surface removal using polygon area sorting [J]. ACM SIGGRAPH computer graphics，1977，11(2)：214-222.

[9] APPEL A. The notion of quantitative invisibility and the machine rendering of solids[C]// Proceedings of the 1967 22nd National Conference. 1967：387-393.

[10] GALIMBERTI R. An algorithm for hidden line elimination[J]. Communications of the ACM，1969，12(4)：206-211.

[11] LOUTREL P P. A solution to the hidden-line problem for computer-drawn polyhedra[J]. IEEE Transactions on Computers，1970，100(3)：205-213.

[12] DEVAI F. Quadratic bounds for hidden line elimination[C]// Proceedings of the Second Annual Symposium on Computational Geometry. 1986：269-275.

[13] GHALI S. A survey of practical object space visibility algorithms[J/OL]. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=3a735dce573effddf635dd03310ef50fafc2c3e9.

[14] OTTMANN T, WIDMAYER P. Solving visibility problems by using skeleton structures[C]// International Symposium on Mathematical Foundations of Computer Science. Berlin, Heidelberg: Springer, 1984: 459-470.

[15] OTTMANN T, WIDMAYER P, WOOD D. A worst-case efficient algorithm for hidden-line elimination[J]. International Journal of Computer Mathematics, 1985, 18(2): 93-119.

[16] NURMI O. A fast line-sweep algorithm for hidden line elimination[J]. BIT Numerical Mathematics, 1985, 25(3): 466-472.

[17] GOODRICH M T. A polygonal approach to hidden-line and hidden-surface elimination[J]. CVGIP: Graphical Models and Image Processing, 1992, 54(1): 1-12.

[18] HSU W I, HOCK J L. An algorithm for the general solution of hidden line removal for intersecting solids[J]. Computers & graphics, 1991, 15(1): 67-86.

[19] SONG R J, ZHANG J L, LI X D. Study and improvementation of hidden-line removal algorithm for convex polyhedrons[J]. Computer Engineering and Design, 2012, 33(6): 2358-2362.

[20] XU X, SHI K L, YONG J H. Hidden-Line Elimination with Tolerance[J]. Journal of System Simulation, 2013, 25(9): 2079-2084.

[21] ELBER G, COHEN E. Hidden curve removal for free form surfaces[J]. ACM SIGGRAPH Computer Graphics, 1990, 24(4): 95-104.

[22] JANSSEN T L. A simple efficient hidden line algorithm[J]. Computers & Structures, 1983, 17(4): 563-571.

[23] SPILLERS W R, LAW K H. On the hidden line removal problem[J]. Computers & structures, 1987, 26(4): 709-717.

[24] 3D ACIS Modeler[EB/OL]. (2024-06-15)[2024-06-16]. https://www.spatial.com/products/3d-acis-modeling.

SONG Haichuan, born in 1986, Ph.D, associate professor. His main research interests include computer vision, 3D printing, and point projection.

(责任编辑:何杨)