多面体消隐算法的文献查阅报告

21200612 16 班 郭冠男

作业要求

根据文献查阅报告的基本要求，将本文相关的信息列举如下：

1. 本文主要查阅了以下三篇文献:

[1] *A Characterization of Ten Hidden-Surface Algorithms*，Ivan E. Sutherland et al, *ACM Computing Surveys (CSUR)*, Volume 6 Issue 1, pp 1-55, March 1974

[2] Z 缓冲区消隐算法的改进，生滨等，计算机工程与应用，第 37 卷第 23 期，114-116，2001 年

[3] Z 缓冲消隐算法的改进，简学东等，计算机应用与软件，第 24 卷第 9 期，149-150，2007 年 9 月

2. 本文选题为：(4) 《计算机图形学》凸多面体消隐算法。本文以针对“*A Characterization of Ten Hidden-Surface Algorithms*”的阅读为主，以另两篇论文中的观点为辅，总结了一些文中提到的消隐算法。在介绍原文中的主要理念的同时，还阐述了对于消隐算法的一些个人见解，其中如有谬误敬请指出。

概述

A Characterization of Ten Hidden-Surface Algorithms 是一篇综述性质的论文，总结了 1963-1974 十年间各机构对“面消隐问题（Hidden-Surface)”或“线消隐问题（Hidden-Lines)”的研究，并创造性地提出了一套“分类方法（Taxonomy)”对十种已知消隐算法进行了系统的总结，对其后很多研究的开展以及教材的编写都有着深远的影响。由于发表时间很早，其中有一些观念与现在通行的硬件设施已经并不完全相符。但文中对于各种“一致性(Coherence)”的抽象在今天看来仍然具有广泛的适用性。例如，当文中提到 Warnock 四分算法时提到，需要将算法得到的“区域一致（Area Coherence)”的矩形单元借助其他算法以使得输出兼容光栅扫描从而提高效率，而这一点对于今天的输出设备而言往往已经不再需要给予特殊考虑。

文章的介绍部分概述了“含有不透明物体的真实感图形绘制问题”，并指出了这一问题的复杂性——一方面我们需要确定哪些对象需要绘制，哪些对象不需要绘制；另一方面，对于那些需要绘制的对象，我们需要进一步确定，这些对象内部哪部分需要被显示，哪部分被其他对象遮挡，而这就涉及了复杂的三维空间中的面遮挡关系。

文章的背景部分定义了一些后文中需要使用到的术语并且介绍了“三维透视变换（Perspective Transformation)”的概念。所谓“三维透视变换”是这样的一个变换：它将一个三维空间中的点集 A，变换到三维空间中的另一个点集 B，使得在 A 中进行透视投影的结果等价于在 B 中进行平行投影的结果。这就使得，对于一个消隐算法而言，如果能在平行

投影成像结果中正确地进行消隐，那么它就也可以被用于透视投影的成像中。后文中的全部内容均是基于这样的一个理念，因此文章只考虑平行投影下的线面消隐问题。更近一步，我们不妨假定投影平面永远是xOy平面，而投影方向永远与z轴方向平行。

十种算法

这些算法中使用到了一些通用的定义或约束，在此一并给出：

1. 平面方程的方向：由于我们所考虑的每一个面都是多面体的一个外表面，因此当我们定义这些面的平面方程时要保证，多面体内的点带入方程得到结果一定小于零。换言之，我们需要保证平面$Ax + By + Cz + D = 0$中法向量(A, B, C)方向垂直于该表面且指向多面体外部。
2. 正向面和背向面（back face）：假定观察者位于 z 轴上负无穷远处进行观察。如果一个面的法向量的 z 分量小于等于零（即指向观察者），我们称这个面为正向面，否则我们称这个面为背向面。在各种算法运行前，我们都可以先删去所有背向面，算法的正确性一般都不会受到影响。
3. 外围边（contour edge）：本文中给出的所有算法均不考虑悬边和悬面，因此可以将一条边理解为两个面的公共线。外围边是指那些恰好连接了一个正向面和一个背向面的边。
4. 深度（depth）：三维空间中某点处的深度即为该点到投影面xOy的距离。

L.G.Roberts (1963) 体积遮挡方法

Roberts 的方法使用了“体积（volume）”这一概念用于检测某一条边是否被其他对象所遮挡。当我们要检查一条边E_1E_2是否被某个对象遮挡时，我们就从 E 上的每一点引出一条平行于投影方向的射线（假设采用平行投影）。这里“对象”指：一个封闭的凸多面体。对于一条射线上的一点 P，如果 P 位于某一个凸多面体内，那么边E_1E_2上的对应点就会被遮挡。具体而言，我们要检验点P是否在凸多面体内，只需要检验P带入凸多面体的每个面的平面方程结果是否均小于零。

本文中并没有进一步给出这一算法的具体实现，但我推测可以使用二维空间中的半平面交解决这一问题。对于$P(\alpha, \beta) = (1 - \alpha)E_1 + \alpha E_2 + \beta \cdot (0, 0, -1), \alpha \in [0, 1], \beta \in [0, +\infty)$，我们把$\alpha, \beta$视为自变量，将$P(\alpha, \beta)$带入凸多面体的每一个平面方程都会得到一个与$\alpha, \beta$有关的二元一次线性不等式。我们可以在$\{\alpha, \beta\}$空间中通过半平面交计算得到最后符合条件的$\alpha, \beta$取值范围。其中，$\alpha$的取值范围即表明了线段上被其他对象遮盖了的部分。Roberts 使用了包络盒的方式优化了这个算法实现，如果一个凸多面体的包络盒与这些射线构成的射线束都没有公共点，则多面体本身也和射线束没有交点。注：$(0, 0, -1)$为一个指向 z 轴负方向的单位向量。

三种边交算法（Edge-Intersection Algorithms）

这三种算法分别由 A.Appel (1967)，P.P.Loutrel (1967)，R.Galimberti (1969) 提出。三种

算法都以考虑边与边（所在的面）之间的遮挡关系，进而给出一个消隐线算法。

在 Appel 的论文中定义了“不可见度（quantitative invisibility）”的概念，对于某条边上的某一点P，如果这一点被N个正表面遮挡，则称这一点的不可见度为N。不难发现，对于一条边而言，只有当这条边在xOy面上的投影与某条外围边在xOy面上的投影相交时，这条边上的不可见度才可能发生改变。当不可见度发生改变时，要么增一，要么减一，取决于当前边的前进方向是穿入这条外围边所在的多边形还是穿出这条外围边所在的多边形。

我们可以先使用穷举所有面的方式暴力计算出某个顶点处的不可见度，再从这个结点开始向外拓展。具体而言，我们要枚举所有连接该顶点的边，考虑这些边上的哪些位置发生了不可见度的改变。当整个程序运行结束后，每条边上不可见度为零的部分即为需要被显示的部分。我们可以以顶点为媒介向外拓展不可见度的值，两条边E_1E_2与E_2E_3有公共顶点E_2，如果对边E_1E_2的计算已经完成，那么我们可以利用E_2点的不可见度（即E_1E_2的终点不可见度）作为E_2E_3的起点不可见度。这一点体现了“边一致性（edge coherence）”。Loutrel 的做法与 Appel 大同小异，基本上只是名词定义不同。Galimberti 的做法本质上与 Appel 的做法相同，只不过他的做法要求我们记录具体哪些面遮挡了某条边上的某一段，这些面构成的集合称为“本性（nature）”。而 Appel 的方法中的不可见度即为“本性”中的元素个数，由于 Appel 的方法并不关注具体哪个面遮挡了某点而只关心符合条件的面数，因此效率更高。

这个算法需要特殊处理某个面恰好经过某个顶点，或者某条边恰好经过某个顶点的情况，对于这种情况我们就不能简单地把这个点的不可见度带入到与它相连的其他边中，否则会导致出乎意料的错误。

四种像空间算法（Object-Space Algorithms）

像空间就是最终显示输出图形的二维空间，由于我们的输出设备总是以像素（picture element）为最小输出单位，从输出的角度讲，如果我们计算的精度远远高于输出的精度，再高的精度也意义不大。注：*A Characterization of Ten Hidden-Surface Algorithms* 一文中使用“picture element”一词表示像素，而不是现行的惯用词“pixel”。

以像素为单位的输出会导致两种常见的输出错误：一是锯齿（sawtooth）的出现，二是细长物体由于欠采样而被忽略。我们可以使用抗锯齿的方法消除锯齿，具体而言，我们可以让一个像素点不再只输出该点处的亮度值，而输出其邻域内的亮度均值。由于细长物体上的任意一个点可能都不在光栅图的采样点上，如果需要保证这个物体仍能在最终的图像上有所体现，那么就要使用类似抗锯齿的方法进行邻域均值处理。

Warnock 四分算法

四种像空间算法中 Warnock 四分算法（1968）最为独特，它是十种算法中唯一主要利用“区域一致（area coherence）”的算法。区域一致是指：当输出的图像在某点处具有某一颜色或亮度，其附近的点往往也具有相同或相近的颜色或亮度。当我们要生成原三维空间在某一矩形范围内的像时，可以先用这一矩形区域对所有可能显示的面进行裁剪，如果裁剪后我们发现这个矩形范围内没有任何面或者存在一个很靠前的面完全挡住了其后的所有面，则直

接显示这一纯色矩形区域。否则，就对这个矩形范围进行四分，划分为四个大小相等的子矩形递归这一算法。其他三种像空间算法都可以针对光栅扫描进行存储空间的优化，具体实现类似论文[2]，而这种算法难以保证搜索顺序与光栅扫描的输出顺序一致，因此在光栅扫描设备上需要另加处理才能正确绘制。

三种点采样扫描线算法

三种算法分别由 W.Romney (1967), W.J.Bouknight (1969), G.S.Watkins (1970) 提出。在逻辑上他们都先对每条边按照最小 y 坐标排序。再用一条平行于 xOz 平面的扫描面去和三维空间中的每条边相截，此时，多边形的一些正向面上会被截得一个线段。最后我们再考虑所有截得的线段间的遮挡关系。这样我们就避开了对于三维空间中的面的遮挡关系的讨论，只需要讨论一个二维空间中线的遮挡关系。这种算法利用了扫描线一致性（scan-line coherence）进行优化，换言之，我们并不需要在每次移动扫描面时都对所有平面两两求交，我们只需要记录在扫面线移动的过程中，三维空间中的每一条边何时进入扫描面以及何时离开扫描面即可。

Romney 对所有当前还没离开扫描面的边置于一个按当前交点 x 分量排序的边数组中，每当扫描面移动导致交点 x 分量变化时，就采用冒泡排序重新进行排序。由于每次移动扫描面时，所有边与扫描面的 x 交点变化不会很大，而且新进入扫描线的边数不会很多，因此冒泡排序的效率在这种基本有序的数组中较为理想。而接下来的一步，Romney 的做法类似于对这一行像素点建立一个 z 缓冲，称为 x 向占据表（X occupied table），从而解决这一行的显示问题。

Bouknight 的做法也需要进行这种排序，排序后在 x 方向处理每一个多边形的进入退出事件。由于一个多边形在某条扫描面上交得一个线段，称这个线段在投影面上的投影为“映像区间”（span）。如果每个映像区间两两没有交集，那么可以直接显示；如果映像区间有交集，则必须考虑两个多边形表面是否存在互相穿透的情况，找到对应交点后再进行分段显示。

Watkins 在对两个多边形表面是否存在互相穿透的判断更为激进（aggresive）。当两个多边形在扫描面中对应的线段难以判断谁更靠近投影面，直接对其中深度跨度较大的线段按照长度均分为两个线段，直到某一时刻任意两个线段 AB 和 CD 都满足：要么 $\min(Dep(A), Dep(B)) \geq \max(Dep(C), Dep(D))$，要么 $\min(Dep(C), Dep(D)) \geq \max(Dep(A), Dep(B))$为止。如果一条线段上的最小深度都大于另一条线段上的最大深度，那么这条线段一定可以被另一条线段遮挡且不存在交叉的情况。

两种序列优先算法（List-Priority Algorithms）

所谓序列优先，就是将三维空间中的每个面赋予一个优先级，保证无论视角如何变化，优先级高的面总不会被优先级低的面遮挡。

R.A.Schumaker (1969) 给出了一种序列优先算法。我们对于一个簇（cluster）内的所有平面进行两两比较观察遮挡关系，得到每个面的簇内优先级，对于每一个簇作为整体两两比

较观察遮挡关系得到簇的优先级。无论视角怎么变化，簇内优先级只需要计算一次。而簇的优先级需要根据视角的变化动态重算。在一个簇内，凸壳上的面优先级最高，我们可以构造一个有向图来表述面与面之间的序关系，通过拓扑排序确定每个面的优先级。如果这个有向图中存在回路，则回路上的某些面不得不被划分到其他簇中，以保证赋予优先级的过程可以进行。在最终显示时，我们将以簇优先级为第一关键字，簇内优先级为第二关键字确定每一个面的优先级。类似地，显示还是按照扫描线的方式处理，但是有了优先级的帮助扫描线中可以省去很多不必要的深度判断。

设置簇的一个基本原则为：任意两个簇之间必须是线性可分的，否则我们将难以计算簇优先级。假如簇 A 与簇 B 可由平面α分隔开，那么当观察点 P 与 A 位于α同侧时，我们说 A 的簇优先级高于 B，反之 B 的簇优先级高于 A。

M.E.Newell (1972) 的方法使用了一种称为覆写 (overwrite) 的做法实现了对透明效果的支持，高优先级的表面会覆写低优先级的表面。算法其他部分与 Schumaker 的做法比较类似。

总结

从数据来源的角度讲，上文中提到的十种算法大致可以分为以下三类：第一类是完全在对象空间空间进行的算法，这类算法在计算过程中几乎没有精度损失，可以得到一种准确的遮挡关系，而其他两类算法均或多或少与输出设备的分辨率有关。体积遮挡法以及三种边交算法均属于此类。第二类算法是所谓的像空间算法，这类算法的编写依赖于对显示设备分辨率的了解，并且可以借助对显示的了解，简化算法的实现。三种扫描线算法以及四分算法均属于此类。序优先算法介于两类之间，一方面它们先利用对象空间的性质确定出了簇内优先级，这个优先级将不随视角的变化而变化，另一方面，它们在输出时仍考虑了输出设备的特性，设置扫描线缓冲逐行进行显示内容的计算。从目标的角度讲，上述的对象空间算法更关注消隐线问题，而扫描线类算法更关注消隐面问题。对象空间的算法只需要稍加改动就可以实现消隐面的确定，但是扫描线类算法往往不关注线框的消隐。