基于图象空间分布的Shading算法*

劳志强 石教英 潘云鹤

(浙江大学CAD&CG国家重点实验室 杭州 310027)

摘要 本文首先介绍了基于局部光照模型的串行Shading算法,然后对设计分布式Shading算法所用到的动态任务分配、数据适应性任务分配、任务粒度等因素作了较为详细的论述,并给出了所设计的新的数据适应性任务划分算法,最后给出了所设计的分布式Shading算法的描述和实验结果.

关键词 分布式图形,Shading,图象空间,对象空间,任务划分,负载平衡.

中图法分类号 TP391

Shading是一种用局部光照模型来产生真实感图形的绘制技术.同其它真实感图形绘制技术(如:光线跟踪算法、辐射度算法)相比,在同样场景的情况下,产生一幅Shading图的计算量相对于光线跟踪/辐射度等算法来说小了很多,但由于其仍需计算每个可见点处的光强值,且实际应用中图形复杂程度在不断地提高,所以用Shading绘制复杂场景(如10 000个多边形组成)仍然需要较长的时间.

真实感图形生成的加速研究一直是一个非常活跃的研究领域.并行计算是其中一种行之有效的方法.并行算法[1,2]和相应的并行体系结构[3,4]在过去的时间里有了很大的发展,使得有些图形应用达到了实时要求,然而并行机的价格往往使一般用户望而止步.计算机网络的不断普及给图形算法的并行处理提供了一种新的可能,它使得人们有可能在充分利用网络资源的基础上,设计出在网络上并行计算的图形算法,即分布式图形算法.本文就是我们在这方面所做工作的一部分——Shading算法的分布式实现.

1 Shading的图形学基础

1.1 局部光照模型

我们知道,当光照到一个物体表面时会产生3种光,即反射光、透射光以及被物体表面吸收的光,其中只有透射光和反射光能产生视觉效果.在局部光照模型中,环境假设由白光照明,且反射光和透射光的颜色由用户来选定,这种简化使光照明模型的建立和应用变得十

---

* 本文研究得到国家自然科学基金资助.作者劳志强,1967年生,博士生,主要研究领域为计算机图形学,分布式系统,智能CAD系统等.石教英,1937年生,教授,博士导师,主要研究领域为图形加速硬件,分布式图形,科学计算可视化,声象一体化仿真等.潘云鹤,1946年生,教授,博士导师,主要研究领域为计算机美术,智能CAD系统等.

本文通讯联系人:劳志强,杭州310027,浙江大学CAD&CG国家重点实验室

本文1996-10-03收到修改稿

分方便. 在 Phong Shading 所用的局部光照模型中不包括透射光,所以在这种模型中物体表面的颜色仅由其反射光决定. 一般来说,反射光由 3 个分量组成,即环境反射、漫反射和镜面反射. 环境反射分量假定入射光分量则表示特定光照射在景物表面上产生的反射光.

漫反射分量表示特定光源在景物表面的反射光中那些向空间各方向均匀反射出去的光,这种反射光的计算可以使用朗伯余弦定律,即对于一个漫反射体,表面的反射光亮度和光源入射角(入射光线和表面法向量的夹角)的余弦成正比,如图 1 所示.

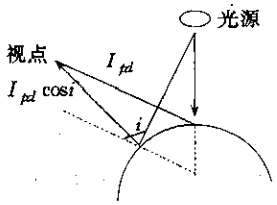

图1 简单漫反射模型

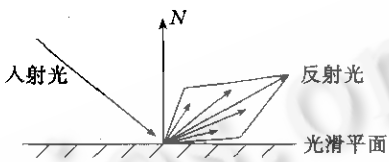

图2 光滑平面的镜面反射模型

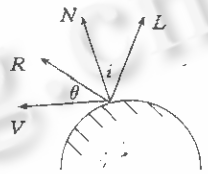

图3 局部光照模型计算中涉及的各方向向量

所以有

$$I=I_{pd}\cos i \tag{1}$$

其中 I 为表面反射光的光亮度,I_{pd} 为光源垂直入射时反射光的光亮度,i 为光源入射角.

式(1)所得到的物体表面的反射光亮度与实际情况不符. 在实际应用中,常常要加上一个称为环境反射分量的常量,即得到下式

$$I=I_{ps}+I_{pd}\cos i \tag{2}$$

其中 I_{ps} 为环境反射分量. 一般来说,$I_{ps}=(0.02-0.2)I_{pd}$.

镜面反射光为朝一定方向的反射光,它遵循反射定律. 对一般光滑表面,由于表面实际上是由许多朝向不同的微小平面组成,其镜面反射光分布于表面镜面反射方向的周围,如图 2 所示.

在实用时,常采用余弦的幂次来模拟一般表面的镜面反射光的空间分布,如下式所示

$$I=I_{ps}\cos^n\theta \tag{3}$$

其中 I 为观察者接受到的镜面反射光亮度,I_{ps} 为镜面反射方向上的镜面反射光亮度,θ 为镜面反射方向和视线方向的夹角,n 为镜面反射光的会聚指数.

前面说过,表面反射光可认为是环境反射、漫反射、镜面反射 3 个分量的组合. 对于一特定的物体表面,这 3 种分量所占得比例具有一定的值,令 K_a,K_d,K_s 分别表示环境反射、漫反射和镜面反射分量的比例系数,则一个实用的局部光照模型可表示如下

$$I=K_a\times I_{pa}+\Sigma(K_d\times I_{pd}\times\cos(i)+K_s\times I_{ps}\times\cos^n\theta) \tag{4}$$

其中 I_{pa} 为环境反射分量,I_{pd} 为光源垂直入射时反射光的亮度,I_{ps} 为镜面反射方向上的镜面反射光亮度,K_a 为环境反射系数,K_d 为漫反射系数,K_s 为镜面反射系数,i 为光源入射角,θ 为镜面反射方向和视线方向的夹角,n 为镜面反射光的会聚指数.

在实际计算时,对表面的每一点,仅需求出它的法向量 N、光线向量 L、视线向量 V 和镜面反射向量 R. 如图 3 所示.

有

$$(L_0\cdot N_0)=\cos i,\ (R_0\cdot V_0)=\cos\theta$$

其中 L_0,N_0,R_0,V_0 是 L,N,R,V 相应的单位向量.

1.2 串行 Phong Shading 算法

Phong Shading的思想是对离散的法向量采用双线性插值,构造一个连续的法向量函数,将这一连续的法向量插值函数代入式(4)中,得到一个非线性的光亮度插值公式,这一方法大大减少了Ground Shading所无法解决的马赫带效应,产生出真正的高光效果.通常产生一幅三维物体的Shading图需要包括以下4个步骤:①对感兴趣的物体做几何的和透视的变换.②根据视域将不要的物体去掉,即所谓裁剪.③对视域中的部分实行消隐.④用式(4)局部光照模型对视域中的物体进行Shading且扫描转换,然后将位图输出到Video Buffer中加以显示.

这说明Shading是在消隐的基础上所进行的,所以不同的消隐算法将直接影响到所设计的Shading算法的效率.基于扫描线消隐[4]和基于Z-Buffer消隐等的Shading算法由于本身的计算粒度较细,所以不适于体积分布式计算.在这里我们选用的是基于Warnock消隐的Shading方法.由于Warnock算法建立在对图象空间四分的基础上,这与我们所设计的分布式Shading算法中的任务划分方法有相似之处,可以说任务划分算法所做的工作是在图象空间中对场景数据进行一级划分,而在单机上则用Warnock算法结合Phong Shading模型再对场景数据进行进一步的划分直到达到显示分辨率或无需划分为止.我们在设计分布式Shading算法中的任务划分算法时,首先是受到了Warnock算法的启发,再进一步参考了Whelan任务划分算法,从而最终设计出下面要介绍的任务划分算法.

2 分布式Shading算法的设计与实现

2.1 分布式图形处理中的定义

定义2.1. 设I是一幅待生成的图形,对I进行剖分,满足

①$I=I_1\cup I_2\cup\ldots\cup I_n$ ②$I_i\cap I_j=\varnothing(i\neq j)$

处理器记为$p_i(0\leqslant i\leqslant n)$.$I_i=\text{output}(p_i)$,称这种方式为图象空间分解,并把基于图象空间分解所获得的并行定义为图象空间并行,简称为ISP(image space parallel).

定义2.2. 设$G=\{g_1,g_2,\ldots,g_m\}$,其中$g_i(0\leqslant i\leqslant m)$是图形对象,对$G$作剖分,有

①$I=G_1\cup G_2\cup\ldots\cup G_n$ ②$G_i\cap G_j=\varnothing(i\neq j)$

令$G_i=\text{input}(p_i)$,称这种方式为对象空间分解,并把基于对象空间分解所获得的并行定义为对象空间并行,简称OSP(object space parallel).

定义2.3. 设有一组子任务,有$T=\{t_i|i>0\}$,网上可用的处理机组$P=\{P_i|i>0\}$,如果由用户或程序员显式指定执行t_i的处理机t_j,那么称这种任务分布方式为任务静态分布.同样在进行任务分布时,由系统根据当前系统中各个处理器性能和当前负载情况把任务动态地分配到多个处理机上执行,称这种分布为任务动态分布.

定义2.4. 对于一个数据的输入集合$D=\{d_i|i>0\}$,如果任务的划分是独立于D所进行的,这种分布方法称之为数据非适应性分布方法.如果任务的划分是依赖于D的,这种分布方法称之为数据适应性分布方法.

定义2.5. 分布式程序必须包含可以同时执行的独立部分,这些独立部分的大小被称之为并行粒度.并行粒度的大小有时是以这些程序的执行时间来衡量的.在图形程序的不同的并行层次有不同的并行粒度,常用的并行层次如下(按粒度递增的顺序排列):指令、语句、子程序、同一程序的多个实例同时执行、任务.

2.2 任务划分算法的设计与实现

2.2.1 设计任务划分算法时要考虑的因素

① 图象空间划分

图形的分布式方法有对象空间和图象空间两大类方法. 对象空间方法涉及到对场景中各对象数据的分析,这对于要计算场景中各对象间关系的图形算法(如光线跟踪、辐射度算法等)来说是一种十分有效的任务划分方法,但在 Shading 中,对象间除了相互间的遮挡关系需要处理外(而这部分工作由消隐程序处理),没有其它关系需要处理,所以没有必要用到对象空间方法. 加之对象空间方法的场景分析很困难,尤其对于自交物体,其场景分析相当费时,而这部分工作是额外的,不能用并行计算所代替,这将大大减少由于并行计算所带来的速度的提高. 因此我们在设计任务划分方法时用的是图象空间方法.

② 动态任务划分和数据适应性任务划分

在设计图象空间任务划分算法时,有许多因素要考虑,如子任务与各处理机间的对应关系、数据在图象空间中的分布状况、任务粒度大小的选择、处理机间数据传送的通讯时间开销等等. 为了寻找一种有效的分布策略以达到各处理机间任务的负载平衡,我们在设计任务划分算法时采用了两个策略,即任务的动态分布策略和数据适应性分布策略. 动态分布策略使得处理机与子任务间的对应关系是不固定的,完全根据计算过程中子任务序列和处理机当时的处理情况而定,这种策略在子任务的大小基本相等的情况下有利于达到负载的平衡;数据适应性策略则根据数据在图象空间中的分布情况来进行任务划分,这样做的结果使得子任务的大小基本相等,进一步维护了分布式计算中的负载平衡. 在选定了这两个策略之后,在很大程度上对负载的平衡起了决定性的作用.

③ 任务划分粒度和局部数据调用

动态分布任务的方法减少了负载的不平衡,但因为在计算快要结束时,子任务很小,只有很少的计算需要进行,这时子任务的数目对于负载平衡的维护有很大的影响;另一方面,任务数目越多,负载不平衡的情况越来越容易消除. 但是由于分配导致的连续性的破坏,通讯时间和网络冲突的增加所带来的额外开销越来越多,因此选择合适的任务粒度比是一个直接影响到系统性能的关键因素. 我们在实现时发现粒度比 R(R=子任务数目/处理机数目)的取值在 12:1~24:1 范围内较为合适.

为进一步减少分布式计算中的通讯开销,我们选用局部数据调用策略. 局部数据调用策略是预先将算法的执行代码和场景数据复制到各 Worker 机器上,在执行分布式算法的时候,通过网络提供的远程执行功能 rexec 调用相应的 Worker 机器上的程序执行,这样做虽然占用了较多的存储空间,但减少了网络上的时间开销,且任务粒度大小的选择广泛.

2.2.2 任务划分算法

数据适应性划分策略中有代表的是 Whelan[5]的中间等分划分算法. 该算法的基本思想是将场景中多边形数目作为任务划分的标准,每次划分都将场景一分为二,保证划分成的两部分中多边形数目相等,然后根据任务粒度大小要求,对子区域再进行划分,循环往复,直至达到任务粒度为止.

该算法可以较好地做到任务大小大致相等,但是它却增加了由于要确定中间划分位置而额外需要的排序操作. 该划分算法具有很好的负载平衡能力,但是由于产生子区域所需要

的额外时间开销超过了负载平衡所节省的时间,从算法的总体效果看不是很理想.不过其主导思想是可取的.

我们在参考了Whelan中间等分任务划分算法的基础上提出了我们自己的任务划分方法,但是这种方法在决定任务划分时用的时间相对于Whelan方法要少得多,该算法只基于区域中数据元素的数目,而不管这些多边形的中心位于何处.该算法基于一个假设,即某个区域的处理时间与该区域内的多边形数目成线性关系,基于此原则,这种算法可以得到较好的负载平衡且额外开销较小,该算法的基本思想可如下描述:首先用一张二维的网格覆盖整个显示区域,网格数=粒度比×数理机数目×4.每一格网格有一个权值,即该网格中的多边形数目(基于多边形的包围盒进行统计),然后将相邻的网格进行合并,计算合并后区域中多边形数目权值,该过程一直进行下去直至到达某一点,这时区域中所有的网格都被覆盖,如图4所示.这样得到一颗划分二叉树,如图5所示.

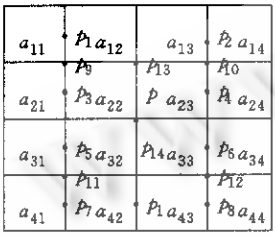

图4　划分算法的划分过程

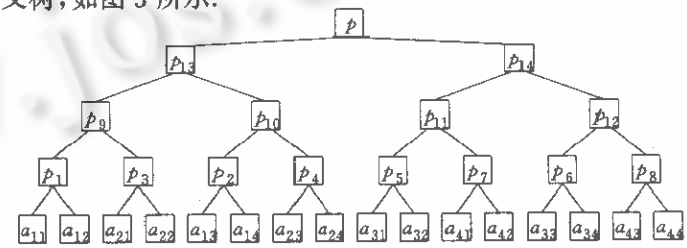

图5　划分算法得到的二叉树

其中父亲结点的权值等于两个儿子结点的权值之和,这里需要给出的是每个结点的数据结构,每个结点要保存的信息包括区域的起始坐标(xl,yb)、区域的长宽(x_length,Y_length)、结点的权值、遍历标志、区域的两个儿子的指针,用C语言可以表示如下:

```
typedef stract {
    short       xl ;          //区域最左坐标
    short       yb ;          //区域最底坐标
    short       x_lehgth ;    //区域长度
    short       y_lehgth ;    //区域宽度
    short       count ;       //节点的权值
    int         flag ;        //遍历标志
    NODE        * leftson ;   //左儿子指针
    NODE        * rightson ;  //右儿子指针
} NODE
```

二叉树产生好以后,接下来要这颗二叉树来划分任务,我们采用自顶向下的遍历方式来遍历该二叉树.首先找到一个权值最大的结点(即该结点所包括的区域中含有最多数目的多边形),将该结点所指的区域一分为二,接着找到权值次大的结点,将其所覆盖的区域一分为二,循环往复,直至达到要求的任务数目为止,由于每格网中多边形数目在产生网格时已经产生且保存好了,因此,这时不再需要遍历整个多边形表来决定哪些多边形与该区域有关,分割过程除达到所要求的任务粒度以外,叶结点也是一个限制条件.该算法可以描述如下:

step1:用一网络覆盖图象空间.

step2:计算各格网格中多边形数目,包括包含在内的多边形、包含了网格的多边形以及相交多边形.

step3:将相邻的网格两两合并,合并的次序先横并后纵并,同时将合并过程用二叉树的形式计录下来,父亲结点的权值等于两个子结点的权值之和.

step4:若合并后的网格不等于图象空间,Goto step3.

step5:从二叉树的根开始自顶向下遍历二叉树,首先将根所表示的区域从中间一分为二.

step6：比较根的左右儿子结点权值. 假设左儿子权值大(右儿子节点大的情况处理方法相似)，将其所代表区域一分为二，将右儿子节点保存起来.

step7：从左儿子节点开始，比较其左、右儿子节点的权值，同样假设左儿子节点的权值为大(右儿子节点大的情况处理方法相似). 先将右儿子节点保存起来，然后比较左儿子节点与保存起来的节点的权值，若以前被保存的某节点的权值比该节点大，那么将该保存节点从保存区域中取出，将其代表区域一分为二，而将该左儿子节点保存起来；若左儿子权值大，则将其代表区域一分为二.

step8：若没有达到要求的任务数目，Goto step7.

step9：结束.

当这颗二叉树被遍历完之后，所产生的这些区域都可以并行地执行. 这里值得一提的是，构造这颗二叉树所需要的时间在整个算法实现中所占的比例是非常小的.

2.2.3 分布式Shading算法

我们的分布式Shading算法是基于Phong模型的，采用Supervisor-Worker模型，算法可描述如下：

```
Supervisor 部分：
  BEGIN
    create_scene_close_box();
    calculate_grid_weight();
    create_task_queue();
    REPEAT
      sys_receive(ANY_PROCESSOR, 30, &result, sizeof(result));
      if result.image = null then
          assign(I, result);
          subscene=allocate_scene();
          sys_send(result, processor, 30, &subscene, sizeof(subscene));
      until done
      draw_image(I);
  END

WORK 部分：
   BEGIN
     result.image=null;
     LOOP
       sys_send(supervisor, 30, &result, sizeof(result));
       sys_receive(supervisor, 30, & subscene, sizeof(subscene));
       result=Shading(subscene);
     END LOOP
   END
```

2.2.4 分布式Shading算法的实现及加速效果分析

2.2.4.1 分布式图形处理支撑环境DGPSE(distributed graphics processing support environment)

DGPSE是浙江大学CAD&CG国家重点实验室在工作站网上实现的一个分布式图形处理支撑环境.[6]该环境支持Client-Server，Client-Server-Broker和Supervisor-Worker-Collector等多种模型，其主要功能包括：分布式处理功能(处理机调度、进程通讯、终止条件检测、出错报告及处理等)以及提供一定的调试设施供用户调试程序. 我们的分布式Shading算法就是在该环境下实现的.

2.2.4.2 实验结果及加速比分析

定义2.6. Amdahl定律[7]给定了有几个处理器的多处理器系统上并行算法的加速比计算公式. 该定律最初是为向量处理提出的，设$T(1)$是在一个处理器上执行某一算法的时间，

$T(n)$则为在 n 个处理器上的执行时间,有加速比 sp_n:

$$sp_n=T(1)/T(n) \tag{5}$$

表1、2是我们在实现该算法时得到的实验数据.由于场景比较简单,当处理机增加时,任务粒度变小,通信量增大,从而加速比受到一定的影响,但是加速效果还是非常明显的.

表1　算法在不同数目工作站环境下运行的时间图(单位:s)

	400	800	1 500
1	139.9	216.9	237.7
2	117.8	132.9	154.4
3	101.1	102.3	110.1
4	83.8	89.6	94.8

表2　算法在不同数目工作站环境下运行的加速比

	400	800	1 500
2	1.18	1.63	1.54
3	1.38	2.12	2.15
4	1.67	2.42	2.50

表1对应的曲线图如图6所示,表2对应的曲线图如图7所示.

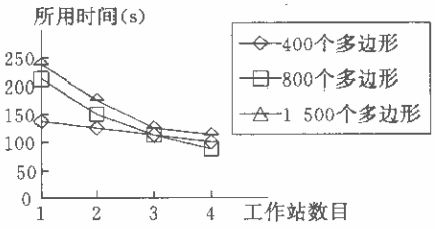

图6 分布式Shading算法对应的时间图

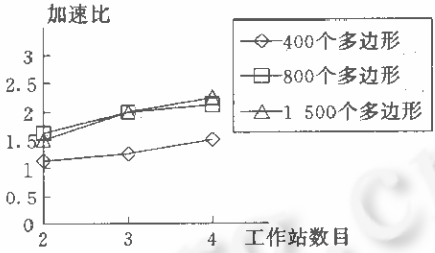

图7 分布式Shading算法对应的加速比图

3 小　结

本文对分布式Shading算法的设计与实现作了较为详尽的描述,尤其是对任务划分时所用到的动态划分、数据适应性划分以及局部数据调用等方法进行了较为详细的描述,并给出了我们在基于这些技术的基础上设计出的任务划分算法.最后给出了算法的实现数据.从实验结果中可以看出该算法的加速是较为效果的,同时可以得出如下结论:①算法用到的工作站数目越多,加速效果越明显;②场景越复杂,加速效果越明显(从图6、7的曲线可以看出场景中多边形的数目越多,该算法的加速比就越高).进一步的的研究可采用任务适应性划分方法,使该分布式图形系统具有进程调度、内存共享等分布式操作系统的功能.

参考文献

1 Theohairs Thehairs, Iar Page. Two parallel methods for polygon clipping. Computer Graphics Forum, 1989,8:107~114.

2 Scott Whitman. Parallel algorithm and architecture for 3D image generation. SIGGRAPH'90, 1990.

3 Kilgour A C, Earnshaw R A. Parallel architectures for high performance graphics system. Fundamental Algorithms for Computer Graphics, Sprubg-Verlag, Berlin, 1985. 695～703.

4 梁友栋,石教英,彭群生等. 计算机图形学的算法基础. 北京:科学出版社,1987.

5 Whelan D S, Animac. A multiprocessor architecture for realtime computer animation [Ph. D dissertation]. California Institute of Technology, 1985.

6 Pan Zhigeng, Shi Jiaoying. DGPSE: a distributed graphics processing support environment. In: Tang Zesheng ed. New Advances in Computer Aided Design & Computer Graphics, the Third International Conference on CAD & Computer Graphics, Beijing: International Academic Publishers, 1993. 54～57.

7 Martin Fruhauf. Interactive scientific visualization algorithms and systems. FHG-AGD Technical Report, 1990.

DISTRIBUTED APPROACH TO SHADING IN IMAGE SPACE

LAO Zhiqiang SHI Jiaoying PAN Yunhe

(*State Key Laboratory of CAD&CG Zhejiang University Hangzhou* 310027)

Abstract This paper firstly gives a description of the serial Shading algorithm based on local illumination model, then gives a detailed description of the methods that are used in the distributed Shading algorithm, which are dynamic task allocation method, data adaptive task allocation method, task granularity etc. Later on, the authors give a full description of a new data adaptive algorithm task allocation algorithm, finally give the description of the distributed Shading algorithm and the result.

Key words Distributed graphics, shading, image space, object space, task allocation, load balance.

Class number TP391