第3卷 第2期
1998年2月

中国图象图形学报
Journal of Image and Graphics

Vol. 3, No. 2
Feb. 1998

在流水线结构上的基于 Shear-Wrap 的并行体数据绘制算法*

黄小虎　李　维　郑南宁

(西安交通大学人工智能与机器人研究所,西安　710049)

摘　要　在以前的基于目标空间划分的并行体数据绘制算法中,局部绘制和图象融合是两个串行的过程,在节点机的局部绘制阶段几乎没有数据通讯,但在数据融合阶段数据通讯量非常大,出现总线争用甚至通讯阻塞,而且在这个阶段有非常大的同步开销。本文利用流水线结构,让局部体数据绘制和图象融合并行执行,很好地解决了上述缺点。并在一个基于微机的流水线结构上实现了一个新的基于目标空间划分的并行体数据绘制算法。

关键词　并行计算,并行体数据绘制,分布式系统

1　引　言

直接体数据绘制技术被认为是操作三维标量场和矢量场数据最有力的工具,但是由于它的运算量极其庞大,同时它还需要有非常大的内存空间,通常几百兆,甚至更大。因此这种算法被广泛使用受到了严重的制约。在过去的几年中,国外的多家公司和研究机构发展了多种并行结构和相应的并行算法。如SGI公司的 Challenge 和 Standford 大学的 DASH 多处理机系统。另外一些专用系统如:英国的 Super Vision 系统、美国的 Pixel Plane5 等。但无论是通用的并行机还是专用的并行机系统,不但它们的费用对于绝大多数的研究机构和大学难以承受,而且这些机器的维护和运行也是非常昂贵的。为此研究成本低,高效的并行结构和与之相适应的并行体数据绘制算法对推动交互式直接体数据技术的理论理论研究和应用研究都有着极为重要的意义。

1.1　并行体数据绘制技术

目前并行体数据绘制算法主要存在两种策略:一种是基于数据空间的并行(DSS),另一种是基于图象空间的并行(ISS)。DSS 算法是根据一定的划分方法将体数据划分成小的体数据,首先每个处理器对局部体数据进行重采样和体素的融合处理,然后将每一个处理机对部分体数据绘制后得到的二维图象进行融合得到最后的图象。由于这种算法在绘制过程中没有太大的数据量需要在处理机间进行传输,因此它适合于基于信包通讯的多处理机系统和分布式处理系统。

相应地 ISS 算法是将 2D 的图象空间划分成多个子空间,每一个处理机只负责计算图象空间的一部分象素。这种算法在绘制过程中随着观察方向的变化,需要体数据在各处理机间进行移动,除非每一个处理机的内存中都有一个全部体数据的拷贝,但这在实际中常常是不可行的。由于此算法的数据通讯量非常大,因此它适应于基于共享存储器的并行处理系统。

近几年随着微机技术和计算机网络技术的发展,引发了人们利用局域网将多台工作站连接起来,使用基于消息传递的通讯策略来研究并行分布式体数据绘制算法。到目前为止已有多个基于这类分布式计算系统的体数据绘制算法被提出[1,2]。虽然这些系统和算法加速了体数据绘制的速度,同时可以利

* 本文得到国家 863 高技术项目基金资助:863-512-27-04
收稿日期:1996-12-21;收到修改日期:1997-04-24

用已有多台工作站，而不需要其它的硬件设备，便可以操作庞大的三维数据，但是这些算法和系统都存在以下缺点：

(1) 在图象融合阶段有非常大的同步开销。另外因各处理机上的图象数据必需在处理机间进行移动，通讯处于极度饱和状态。同时随着融合处理的进行，将有越来越多的处理机处于空闲，最后只有一个处理机处于工作状态。因此在这个阶段并行机的效率变得很低。

(2) 随着节点机数目的增多，早期光线退出变得越来越无效。早期光线退出能够导致串行体数据绘制算法性能提高50%甚至更多。

(3) 没有利用计算和通讯在时间上的重叠来消除通讯瓶颈，因此这些算法由于受通讯带宽瓶颈的限制而不能进行实时绘制。

(4)当节点机数目增加到某个值后，算法的加速比曲线下降得非常快。

(5)由于工作站的价格并不低，因此随着节点机数目的增多，系统的性能价格比不好。

本文提出了一种基于微机网络的并行体数据绘制算法。我们利用现有设备，将多台 Pentium 微机用 100M/s 的以太网卡构成一个流水线结构，使用虚拟并行软件工具 PVM 进行节点机之间的信包通讯。采用 DSS 并行策略，以切片数据为单位，给流水线上的每个节点机分配一叠连续的切片数据。节点机上的绘制算法是以 Standford 大学 Lacroute 等人提出的一种快速体数据绘制算法[3]为基础。为了提高并行算法的效率，节点机上的局部体绘制算法采用按中间图象行的顺序进行绘制。而 Lacroute 的算法是以切片数据为顺序，只有当处理到最后一个切片数据时，中间图象才开始产生。我们采用了静态和动态两种方法来取得系统的负载平衡，在进行任务的动态分配中，节点机并不因此而有过多的通讯开销(这是因为节点机的计算和通讯在时间上是重叠的)。

在本算法中通讯和计算同时进行，而且通讯贯穿整个绘制过程，这样充分地利用了通讯和计算在时间上的重叠从而减少了节点机为等待消息而进行的开销。节点机上的局部绘制和图象的融合并行进行，但在以前所有的体数据绘制算法中[1,2,4,5]，局部绘制和图象的融合分为两个串行阶段，在局部绘制阶段数据的通讯量很少，但在图象的融合阶段数据通讯量急剧增加，出现严重的总线争用，甚至发生通讯阻塞。在本算法中避免了这些缺点。各节点机与它们的后继节点的通讯可同时进行，相应地增加了系统的通讯带宽，而且不会出现通讯线路的争用。由于采用流水线结构，数据的通讯和局部绘制在时间上重叠，所以克服了利用基于局域网的分布式并行系统来完成实时体数据绘制算法所存在的通讯带宽瓶颈。

2 节点机上的基于 Shear-Wrap 的快速体数据绘制

基于 Shear-Wrap 的体数据绘制算法[3]的基本思想是：通过一个 Shear 操作先把体数据从目标空间变到 Shear 空间，在 Shear 空间中切片数据面与观察方向互相垂直。然后进行投影和绘制得到一幅变形的中间图象，最后对中间图象进行 Wrap 操作得到了最后的正确图象。Shear-Wrap 算法的主要优点是中间图象与切片数据面互相平行，因此在绘制过程中，简化了寻找体素的地址计算。

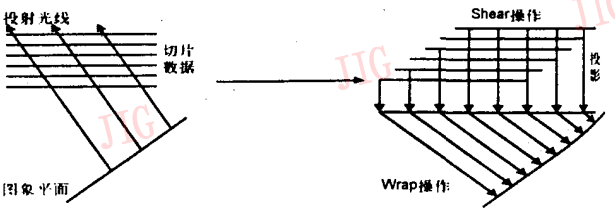

图 1 基于 Shear-Wrap 的平行投影体数据绘制算法示意图

Fig. 1 Diagram of parallel volume rendering algorithms based on Shear-Wrap

Shear-Wrap 算法的基本流程如下：

(1) 根据观察方向，将体数据转换到 Shear 空间。在 Shear 空间中切片数据面互相平行，并且与观察方向垂直。

(2) 以从前到后的顺序，使用“over”算子对重新采样的切片数据(slices)进行融合操作，在 Shear 空间得到变形的二维图象。

(3) 对在 Shear 空间里的变形二维图象进行 Wrap 操作得到正确的二维图象。

在网络分布式环境下进行并行计算，影响算法效率的一个重要因素是通讯带宽的瓶颈，为此尽可能多地使计算和通讯在时间上重叠可以在很大程度上克服通讯带宽瓶颈的限制。如果能够使通讯和计算在时间上完全地重叠，那么并行算法的效率可以达到在真正多处理机上的效率。鉴于这个原因，应尽可能使通讯与计算同时进行。在节点机上的体数据

绘制算法中我们采用了以中间图象行的顺序对体数据进行绘制的方法，即首先绘制中间图象的一行，接着绘制它的下一行、再下一行，以次类推。在 Lacroute 的算法中采用的是以切片数据为顺序对体数据进行绘制，即首先对第 1 个切片数据进行重采样、插值运算等，接着对第 2 个、第 3 个切片……数据进行重采样、插值运算，直到开始处理最后一个切片数据时，中间图象才开始产生。在本算法中每当一个节点机从它的前驱节点接收到一行中间图象时，它立即检查这一行中间图象是否已在此节点机中产生，如果产生就进行这一行的图象融合，然后将融合后的这一行图象发送到它的下一个节点继续进行数据融合。此外它进行本节点机上的体数据绘制任务。通过这种方法使得通讯和计算在时间上的重叠得以实现。

3　流水线结构

本系统采用流水线结构，如图 2 所示。

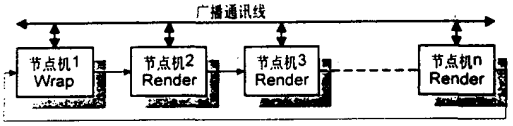

图 2　系统流水线结构图

Fig. 2　System pipeline framework

第 1 个节点用于 Wrap 操作、控制台和图象显示，其它节点用于局部体数据绘制和各节点机上所产生的中间图象的融合。

在每一个节点机上有两条通讯线路，一条用于与它相邻的节点机之间的点对点的通讯，每当一个节点产生一行中间图象时，为进行象素融合它通过这条点对点的通讯线路将这一行象素发送到它的下一个节点。另一条用于广播通讯，每当一个节点在对本节点上的局部体数据进行绘制时，如果中间图象的某一象素产生了早期光线退出，此节点通过广播通讯将产生早期光线退出的信包广播发送给它的所有后续节点，同时广播通讯还用于广播来自控制台的控制命令。例如绘制开始、绘制结束以及观察方向和观察位置发生变化时的参数等。

利用广播通讯将象素早期光线退出的信息通知它的后续节点可以避免一些节点做一些不必要的绘制算法和通讯，它的所有后续节点收到这个广播信包后，不管这些后续节点是否已经对这些象素进行了绘制或图象融合操作，它们都将停止对这些象素进行一切操作：绘制、传输和图象融合。同时将那些已经产生早期光线退出的象素通过广播通讯线直接发送到节点 1 进行 Wrap 操作然后显示。

4　体数据分配和任务的静态平衡

基于体数据空间的并行算法需要我们将体数据空间分解成更小的子空间。目前主要存在 3 种方法来分解体数据空间：块状分配、条状分配和轴状分配，如图 3 所示。理想情况是每一个节点机有相同的任务量，而且各节点机之间的通讯量最少。

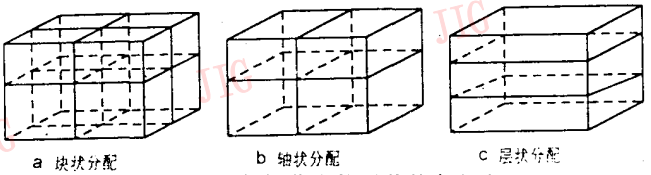

图 3　体数据分配的三种基本方法

Fig. 3　Main methods for volume data assignment

根据 Shear-wrap 的特性，中间图象与体数据的 Slices 面是平行的，同时观察方向垂直于中间图象，为此我们按从前到后的顺序将体数据以 Slice 为单位分给流水线上的各台节点机。在分配前，进行一次任务量的静态估计，根据每一个 Slices 中的非透明 Voxel 的个数来决定任务量的多少。如果观察方向确定，我们可以将 Slices 沿着观察主轴的方向，从前到后，或者从后到前的顺序将体数据以 Slice 为单位分配到流水线上的各个节点机上。每个节点机只负责计算一叠连续的切片数据。节点机上产生的中间图象，沿着流水线的方向进行图象融合操作：$c = \sum_{i=1}^{n} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha)$[2]。但是如果观察方向不确定，就不能这样分配了。实际上，操作者可以在三维空间上以任意方向来观察体数据。这分 3 种情况：观察方向更接近平行 X 轴、Y 轴、Z 轴，如果观察方向主要平行于 X 轴，那么我们以沿着 X 轴方向，来分配 Slices，相应的沿着 Y 轴，沿着 Z 轴。因为在交互式 Render 中，这三种情况随时都有可能发生。为了适应这 3 种情况，就必须在每一个节点机中保存沿 X、Y、Z 分别分配 Slice 数据，否则将在绘制过程中，有可能产生 Slices 数据在节点机中的通讯开销，这将影响实时绘制的速度。为此我们在每一个节点机的内存中保存着分别沿 X、Y、Z 分配的 Slices 数据，如图 4 所示。虽然这种数据分配方法存在一定的数据

冗余度，但是这对节点机的内存是能够容忍的。假定体数据的大小为 5 123，每个体素 2 个字节，那么体数的总容量是 128M 字节。如果流水线上有 10 个节点机，按没有冗余度的数据分配方法，每个节点机大约分配 12.8M 字节左右的数据。在本算法中，分别沿着 X、Y、Z 轴分配数据，那么每个节点应得到 12.8M * 3-512/10 * 512/10 * 512 * 2 * 2＝33.28M 字节的数据量。因为分别沿着 X、Y、Z 轴分配数据，必然会出现数据的重叠分配，沿 X 轴分配给某个节点机的数据，它已经包含了一部分沿 Y 轴分配给此节点的数据和一部分沿 Z 轴分配给它的数据。因此，为了节省内存，在一个节点机中可以只存 33.28M的数据，而不需要存 12.8M * 3＝38.4M 字节的数据。

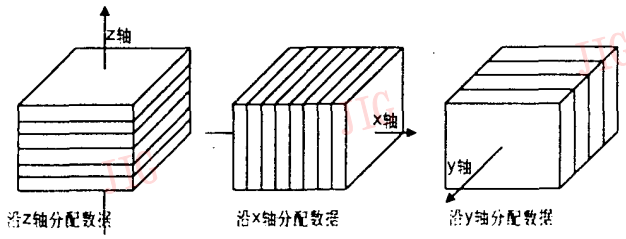

图 4 体数据静态分配

Fig. 4 Static assignment of volume data

5 中间图象融合

在基于体数据空间划分的并行绘制算法中，各个节点机上产生的图象必须进行融合才能形成最后的图象。本文应用了 Under 算子[2]来进行数据融合操作。Under 算子的基本思想是：如果在单个投射光线上有 n 个采样点：

$$c_1\alpha_1, c_2\alpha_2, c_3\alpha_3 \cdots c_{n-1}\alpha_{n-1}, c_n\alpha_n;$$

c_n, α_n 分别表示某一个采样点的颜色和透明度那么这 n 个采样点对象素的累积贡献 C 是：

$$c_1\alpha_1 + (1-\alpha_1)[c_2\alpha_2 + (1-\alpha_2)[c_3\alpha_3 + \cdots c_n\alpha_n]$$

即，$C = \sum_{i=1}^{n} c_i\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j)$

数据融合就是按照从前到后的顺序对各个节点机上的中间图象进行上述运算操作。到目前为至，已发展了多个图象融合算法，最具代表性的是基于 K-D 树结构的并行算法[2]和二次交换数据融合算法[1]。在基于 K-D 树结构的算法中，数据融合阶段有非常大的同步开销同时随着融合处理的进行，有越来越多的计算机处于空闲状态。二次交换数据融合技术显然也存在相当大的同步开销。本文利用 Shear-Wrap 算法的一些独特的性质，采用流水线并行结构，消除了在数据融合阶段存在的非常大的同步开销以及通讯阻塞问题。通讯与计算同时进行，避免了 CPU 在等待接收信包时处于空闲状态。另外由于使用了流水线，相邻节点机之间的通讯可以同时进行，实际上等于增加通讯带宽。通讯带宽在其它并行算法中一直是影响并行算法的瓶颈。

5.1 象素融合的基本思想

节点机之间的通讯时间可以被分为两部分：为进行通讯 CPU 所进行的必要操作时间和信包在通讯线路上的延迟。在本系统中前者大约占总通讯时间的 1%～2%，也就是说如果能够将通讯和计算在时间上重叠，系统就能够克服通讯带宽的瓶颈。在上述算法中节点机上的任务（局部绘制和图象融合）所需 CPU 的时间总是远远大于信包在通讯线路的总延迟，因此我们利用流水线结构消除了通讯对并行算法性能的制约。

本系统中节点机上的局部体数据绘制算法是以中间图象的顺序来绘制体数据，当一节点机产生中间图象的一行后，立即将这一行发送到它的后续节点，它的后续点接到这一行的中间图象后，再看中间图象的这一行是否也在此节点机上产生，如果此行在本节点机中已产生，那么就进行这一行的融合操作，融合操作完毕后，将融合后的这一行继续发送到它的下一个节点。

为进行上述图象的融合算法，我们在每一个节点机上开辟了 3 个缓冲区：接收缓冲区、发送缓冲区和本节点的中间图象缓冲区。接收和发送缓冲区用于节点机的通讯，中间图象缓冲区用于存放局部绘制的中间图象。为了节省内存的开销，我们不将节点机产生的中间图象直接放在中间图象缓冲区中，而是每当一行新的图象生成时，将这一行图象放在一个动态申请的内存块中，而只把指向这个内存块的指针存放在中间图象缓冲区中。为了更清楚地描述本文的并行算法，用程序伪代码描述如下：

```
While( task-Id ! = End ) ;
    如果任务标志位不是结束，程序继续
  While( receive-buffer ! = Null );
    检查接收缓冲区
  if( this image row in this node ! = Null );
    从接收缓冲区中取出一行，并检查与之相应的行是
    否本节点已绘制，如果绘制，将进行融合
    compositing(the image row);开始融合操作
    send(the image row composited to next node);
```

```
        将融合后的图象行发送到下一个节点；
      endif
    end
  if(the tasks of local fast volume rendering ! = End)；
      是否局部绘制任务已结束，如果没有
    Fast-Volume-Rendering( ) ；
      继续进行局部体数据绘制
  endif
End
```

我们在每一个节点机上都设了一个任务标志位，当一个节点机收到一个来自控制台的结束任务的广播信包时，它就将这个标志位置为结束。

在任务分配时，就考虑让流水线的第 1 个节点作为 Wrap 操作。每当流水线末端节点生成中间图象的一行，就将它发送到第 1 个节点进行 Wrap 操作。因为第 1 个节点不与其它节点进行融合操作，因此它的数据融合任务比其它的节点机要小。Wrap 操作的花销与体数据的大小无关，只与图象大小有关。也就是说图象的大小确定了，Wrap 的花销也就确定了。在本算法中 Wrap 将花费总绘制时间的 8～10%左右。为此我们不给第 1 个节点分配绘制任务。

6　动态任务平衡

尽管通过计算每个节点机上的非透明的体素个数来进行任务的静态分配，但是随着观察角度的变化以及受早期光线退出的影响，静态分配不能够完全保证任务的平衡，为此我们发展了一个动态平衡策略。基本思想如下：每当一个节点机上的总的任务量(剩下的局部绘制任务，正在等待需要数据融合的任务)小于某一个值(这个值可以动态地调整)，就向它的前驱节点申请任务。这个节点可能会从它的前驱节点中得到的任务类型是它前驱节点上的局部绘制任务的一部分，也就是说如果它的前驱节点同意将一部分绘制任务分给它，它的前驱节点就要将一部分体数据传送给它。数据分配方式：它的前驱节点机从未被绘制的中间图象行中选出两行或几行，将其所对应的体数据从自己的内存中找出来，并将这些体数据发送给它。在 Shear 空间中，对于每一个中间行，它所对应的所有切片数据行是确定的，因此找出这些切片数据行的计算花销是非常小的。在本文的算法中，每一个节点机首先要将局部体数据从目标空间转换到 Shear 空间，所以节点机为任务动态分配所进行的计算花销相对于节点机上的局部绘制时间可以不计。任务分配策略：当它的前驱节点收到一个任务请求包后，并不是每一次都给它的后序节点分配任务，这要取决于节点机上所剩下的局部绘制任务量、为任务分配所需要的额外开销以及通讯开销。如果它所剩下的任务量非常的小，以至于如果进行任务分配，因额外开销和通讯开销会导致算法的性能下降，那么就不进行任务分配。如果它所剩下的任务量特别多时，任务分配的大小不但要考虑分配所需的通讯开销和通讯线路上的延迟，而且任务分配的大小还要根据当前请求任务的节点机上的可用内存容量，不能因任务分配的太大，而使分配的体数据无法存放。在这里我们还需要提到的一点是，在进行任务分配时，如果任务的分配大小是 n 行的中间图象绘制量，那么必须将在局部绘制这 n 行图象时需要的所有切片数据行传送到它的后续节点，即在这个节点机上，对于每一个切片，我们取 $n+2$ 行采样点数据，而不是 n 行，这是因为在局部体数据绘制中要进行二维采样点的插值运算。同时正是这个原因这 n 行的图象最好是连续的，以节省数据的通讯量。

7　早期光线退出

在基于体数据空间划分的并行算法中，早期光线退出技术会随着处理机数目的增加，变得越来越无效。早期光线退出在串行算法中可以将绘制速度提高近 50%或者更多，也就是说，在基于体数据空间划分的并行算法中，如果不利用早期光线退出技术，将有大约 50%或者更多的并行计算能力被浪费掉。本文为了尽可能利用早期光线退出，采取了以下策略：在节点机中，每当产生一次光线早期退出，此节点就将产生光线退出所在中间图象中的位置，通过广播通讯发送到它的所有后续节点机中。当每个节点机在进行象素融合操作时，首先检查它的接收缓冲中该中间图象的象素是否在它的某一个前驱节点中已经早期光线退出，如果这个象素在它的某一个前驱节点中已早期光线退出，那么跳过这个象素，不对它进行象素融合操作。因为流水线上的各节点机的局部体数据绘制算法是异步的，如果要充分地发挥早期光线退出的效率，在绘制过程中，每个节点的绘制行就必须略滞后于它的前驱节点的绘制行，即如果我们从第 1 行到第 n 行的顺序来绘制图象，而且某个节点正在绘制第 3 行图象，那么它的前驱节点就必须在绘制第 4 行或第 4 行以后的图象。为此在进行任务的静态分配和任务的动态分配时，分

给每一个节点机的任务量略大于它的前驱节点上的任务量,使得一个节点的绘制尽可能地略滞后它的前驱节点。系统不会因为分给每一个节点机上的任务量略大于它的前驱节点而引起新的任务不平衡。这是因为在本系统中,一个节点的所有后续节点都比这个节点本身有更多的机会获得象素的早期光线退出,而且离它越远的节点机获得的越多。这样它的后续节点通过获得早期光线退出平衡了多分的任务量。

8 实验结果

我们用 100Mbit 的以太网将 16 台 Pentium 微机连接成流水线结构。使用 PVM(并行虚拟开发工具)来处理节点机之间的同步和数据通讯。每个节点机在 Windows NT 下运行。使用的测试数据是由 Standford 大学提供的"brainsmaill"核磁共振体数据。它包含 128×128×109 个体素(voxel),占用内存 1.3M,18%的体素是非透明的。生成的图象是 256×256 的灰度图象。

表 1 给出了系统各部分的开销以及性能参数。在计算表 1 中的性能指标时没有包括图象的显示时间和系统的预处理时间(系统的初始化以及为任务平衡分配所进行的非透明的体素数量的统计)。在进行绘制前,要使用了一个分类函数对体数据进行分类处理,我们没有将这个分类处理花销计算在绘制的总花销里,这是因为,用户对一个体数据进行分类后,一般不会马上改变分类函数,而是会从不同的角度和位置来观察分类后的体数据。光照计算是基于 Phone 模型,在着色的过程中为了减少光照的计算量,我们使用了查找表技术。在绘制开始计算着色查找表,在对某一个体素进行着色时只需要计算出该体素的面的法线向量,然后通过着色查找表直接得到该体素的色彩值。为了使这个着色表尽量地小并同时提高绘制精度,每当观察方向改变,这个查找表就必须重新进行计算。

表 1 系统性能参数统计表

Table 1 System performance

	流水线上处理的处理机数					
测试数据 brinsmall(128×128×109)	1	2	4	8	12	16
绘制时间(秒/帧)	5	2.64	1.28	0.9	0.63	0.512
节点机局部绘制时间(最大值 s)	5	2.62	1.248	0.71	0.43	0.32
数据通讯时间(通讯线路的延迟 ms)	0	0.11	0.10	0.9	0.12	0.122
接收和发送数据开销(ms)	0	0.02	0.018	0.018	0.0181	0.019
动态任务分配开销 overhead(ms)	0	0.72	0.68	0.4	0.33	0.29
节点机为等待消息的开销(ms)	0	0.010	0.05	0.031	0.047	0.010

表 1 中的所有数据是对同一个数据集根据不同的观察角度绘制 58 帧图象后所得的平均值。绘制的图象如图 5、图 6、图 7 所示。我们都知到在基于流水线的并行算法中,算法总的执行时间是运行时间最长的那个节点所花的时间。表 1 中各项数据都是对运行时间最长的那个节点机的各项花销所作的统计。绘制时间就是并行算法绘制一帧图象的总时间。数据通讯时间,即通讯线路的延迟,从表 1 中我们看出随着节点机的增加它的值变化不大。这是因为在本算法中是按层状分配体数据,所以尽管处理机增加,一个节点传送给它的后续节点的图象数据量有可能并不减少,而且数据的通讯带宽是一定的,所以这个值有可能变化不大,但这个值会因观察角度的变化而发生变化。但是,因为数据通讯和节点机的局部绘制在时间上是重叠的,而且节点机的局部体绘制时间相对于通讯延迟的时间要大得多,所以通讯延迟时间并不影响图象的绘制时间。

9 结 论

据我们所知,到目前为止在国内外还未见到将 PC 局域网用于并行体数据绘制算法的报道。本文的研究取得了较理想的实验结果。从它的性能价格比看,上述算法优于现在已存在的绝大多数在基于分布式环境的并行体数据绘制算法。从它的绘制速度看,它的绘制速率几乎和某些利用 MIMD 并行计算机所取得的绘制速率相当。随着 PC 技术的发展,它的性能将越来越强大,而它的价格将越来越低,因

此可以预见利用多台微机来实现交互式的体数据绘制算法很快将会成为一个研究热点。在今年的 Siggraph96 年会上,专门举办了一个专题研讨会"利用微机技术实现三维图形的绘制",与会者反应热烈。目前,如果我们使用更多的 PC 机,同时提高通讯速率(例如,使用 FDDI,或者通过总线使用存储器映象),实时的绘制速率(10～20Hz)是可以取得的。

图 5　绘制的人脑图象

Fig. 5　Brain image rendered

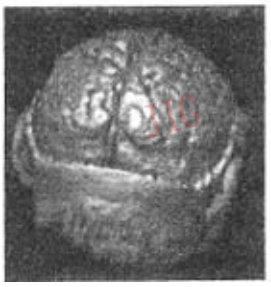

图 6　观察角度转 90 度后

Fig. 6　Brain image viewed on 90°

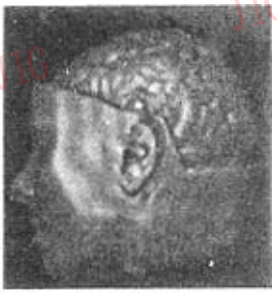

图 7　观察角度转 180 度后

Fig. 7　Brain image viewed on 180°

参考文献

1 Marc Levoy. Display of Surfaces from Volume Data, IEEE Computer Graphics & Applications, 1988, (5): 29～37.

2 Kwan Liu Ma, James Painter. Parallel Volume Visulization on Workstaions, Computer & Graphics 1993, 17(1): 31～37.

3 Philippe Lacroute. Fast Volume Rendering Using a Shear～Warp Factorization of the Viewing Transformation, The Proceedings of Computer Graphics, 1994.

4 Ulrich Neumann, Commmunication Costs for Parallel Volume～Rendering Algorithms. IEEE Computer Graphics & Applications, 1994, 6: 19～58.

5 Kokinis John S, Edward Lent G, Gokhale. Nihar. A Distributed, Parallel, Interactive Volume Rendering Package. The Proceeding of the Visualization 94 Conference, Washington, DC, 1994, (10) 21～30.

6 Jaswinder Pal Singh, Anoop Gupta, Marc Levoy. Parallel Visualization Algorithms: Performance and Architectural Implications. IEEE COMPUTER, 1994, (6): 45～55.

7 Huang Xiaohu, Li Wei, Zheng nanning. A Fast Volume Render On Pc, IS&T/SPIE'S Symposium 97(will pepublished in Feb. of next year)

黄小虎,博士研究生,主要研究领域为实时三维图形生成,交互式科学可视化。

李维,讲师,研究工作为图形图象处理,虚拟现实。

郑南宁,博士,博士生导师,第三界中国青年科学家称号获得者,主要研究领域为图形图象处理,人工智能和模式识别,虚拟现实。

A New Parallel Volume Rendering based on Shear-Wrap Using a Pipeline Framework

Huang Xiaohu, Li Wei, Zheng Nanning

(*Institute of Artificial Intelligence and Robotics Xian Jiaotong University, Xian*, 710049)

Abstract In distributed computing environment, an important factor which will affect the performance of parallel algorithm is communication bandwidth. This paper presents a parallel volume rendering algorithm using a shear-warp factorization of the viewing Transformation on the local area network of PCs. We fully take use of the overlap of communication and computing to overcome the bottleneck of communication. In many parallel volume rendering algorithm based on object partition, local volume rendering and image composition are divided into two serial processes, in the period of local volume , the communication hardly happen, contrarily, in the period of image composition, the communication is very busy, even communication congest happen, furthermore there are a very big synchronism overhead in this period. The paper take use of Pipeline based on PC, local volume rendering and image composition are executed concurrently, the drawbacks above are solved well. Our experiment which is completing on the Pipeline including 16 Pentium shows that communication do not affect the performance of algorithm, and the overheads are very little to rendering time. The paper provide a method for studying low-price, high-efficiency real-time volume rendering system.

Keywords Parallel computing, Parallel volume rendering, Distributed system

书信往来

我校图象图形工作有了很大进展

安徽科技大学汪炳权教授 97 年 12 月 30 日来信摘登：

近十年来，在学会的组织推动下，我校图象图形方面的教学及研究工作有了很大进展，本校电子工程与信息科学系开设多门图象图形课程。计算机视觉、图象识别、图象传输等方面培养了数十名硕士研究生，完成了近十项省部级项目。去年我校顺利通过国家“211 工程”预审，图象工程实验室被列为重点建设项目。人力、物力均得到较大充实。同时，我校计算机系、自动化系、人工智能研究所也有一定的图象图形方面的教师从事相应的教学及研究工作。这还要致谢学会和学报的支持和促进！

为了推动学会理事会人选年轻化工作，我们特推荐我校副校长韦穗同志，51 岁，女，原来在中科院智能机械研究所从事模式识别、虚拟现实、机器人视觉方面的研究工作，造诣颇深，是国家 863 计划专家组智能机器人课题专家成员。