文章编号:1001 - 9081(2006)08 - 1916 - 03

基于 MPI 的匹配方体并行计算研究

罗秋明[1],王 梅[2],雷海军[1]
(1. 深圳大学 超级计算中心,广东 深圳 518060; 2. 深圳职业技术学院 电信学院,广东 深圳 518055)
(qmluo@tom.com)

摘 要:双目立体视觉的匹配方体计算过程可以进行 SIMD 类型的并行计算,基于 MPI 通信环境将视差值的计算任务分配到不同的计算节点上,然后将各节点计算所获得的 DSI 图像汇集在根节点上,最终通过数据规整快速获得所需的匹配方体。同时建立了该并行算法基于处理器时钟周期的相对精确的计算时间复杂度模型,用于分析不同计算平台上的性能。由于计算过程中数据相关性较低,因此在基于 MPI 与 Myrinet 网络的 Linux 集群计算平台上获得了较好的加速比。

关键词:匹配方体;视差空间图像;立体视觉;消息传递接口;并行计算
中图分类号: TP311.52 **文献标识码**:A

MPI based parallel computation of matching cube

LUO Qiu-ming[1], WANG Mei[2], LEI Hai-jun[1]
(1. *Supercomputing Center, Shenzhen University, Shenzhen Guangdong* 518060, *China*;
2. *EI, Shenzhen Polytecnic, Shenzhen Guangdong* 518055, *China*)

Abstract: According to PCAM, a parallel algorithm was figured out to accelerate the computation of matching-cube for stereoscopic vision. The computation of matching-cube was divided by vertical coordinate into many sub-computations, which minimized the communication between computing nodes. By assigning these computation jobs of different disparity values to multiple computing nodes and gathering all these DSI to root node, a matching cube was obtained. A relatively accurate computational time complexity modal was built on CPU cycles to analyze the performance on different platforms, which was very important to real-time applications. As the data-dependence during computation was very low, a nearly linear speedup could be obtained on MPI cluster parallel platform.

Key words: matching-cube; Disparity Space Image(DSI); stereoscopic vision; Message Passing Interface(MPI); parallel computing

0 引言

双目立体视觉的视差/距离计算之前需要先获得匹配方体(Matching Cube)[1,2],即由对应于不同扫描行的所有待选视差值的视差空间图像(Disparity Space Image,DSI)构成的三维数据,然后在这个匹配方体内完成最佳匹配值的搜索过程,从而得到各个像点的视差值,最后得到距离信息。由于此类计算复杂度很高,特别是对于双目立体视频的实时处理提出了更高的要求,因此往往需要并行计算以达到要求,包括专用器件[3,4]、大规模 MPP[5]、SMP 或其他并行计算形式。集群架构的并行计算方式由于具有高的性价比而逐渐普及,因此有必要研究利用集群计算机上 MPI(Message Passing Interface)[6]环境下的并行算法,相对于以往使用数据并行程序语言 parallaxis[5]的"细粒度"算法实现,MPI 具有更高的可读性和易编程性。

在计算不同扫描行对应的 DSI 之间或者不同视差值之间不存在数据依赖性,以此作为任务分割的依据。下面先介绍 DSI 的匹配值计算和匹配方体,然后给出在 MPI 环境下的并行算法及验证结果,最后分析试验结果。

1 DSI 计算过程

为了便于表达和计算,我们采用平行配置的立体成像系统,如图 1,此时世界坐标 $OXYZ$ 和左摄像机坐标 $OX'Y'Z'$ 重合,右摄像机坐标 $O''X''Y''Z''$ 和它们之间有一个平移量 $b = (x_b,0,0)$,没有旋转量,此时的内极线就是左右摄像机的成像扫描线。物体 p 的距离决定它的两个象点的水平坐标之差(视差值),基于区域的方法将计算左右视图各个水平扫描线上各点邻域图的相似性,然后根据一定的准则选出这些象点的对应关系,反推出目标的距离。一个水平扫描线上对应不同视差值的相似值构成了 DSI(x,d 坐标,二维数据),所有 DSI 构成了匹配方体 Matching Cube(x,y,d 坐标,三维数据),如果相似值取相关函数,则其计算过程如下(见图 2):

$$c_{x,y,d} = \frac{cov_{xy,d}(I^l,I^r)}{\mathrm{var}_{xy}(I^l) \times \mathrm{var}_{x+d,y}(I^r)} \tag{1}$$

其中:

$$\mathrm{cov}_{ij,d}(I^l,I^r) = \sum_{m=iK}^{i+K}\sum_{n=j-L}^{j+L}(I^l_{m,n} - \overline{I^l_{mn}})(I^r_{m+d,n} - \overline{I^l_{m+d,n}}) \tag{2}$$

$$\mathrm{var}^2_{ij}(I^l) = \sum_{m=i-K}^{i+K}\sum_{n=j-L}^{j+L}(I^l_{m,n} - \overline{I^l_{m,n}})^2 \tag{3}$$

收稿日期:2006 - 02 - 13;**修订日期**:2006 - 05 - 08

作者简介:罗秋明(1974 -),男,广东人,助理研究员,博士,主要研究方向:计算机视觉、集群技术; 王梅(1971 -),女,江苏人,讲师,硕士,主要研究方向:计算机应用软件; 雷海军(1968 -),男,湖南人,副教授,博士,主要研讨方向:图像处理、视频压缩技术.

$$\mathrm{var}_{ij}^2(I^r)=\sum_{m=i-K}^{i+K}\sum_{n=j-L}^{j+L}(I^r_{m,n}-\overline{I^r_{m,n}}) \tag{4}$$

式中,K、L是用于计算相似值的邻域半径,所以$(2K+1)$、$(2L+1)$是匹配子图的长宽值。

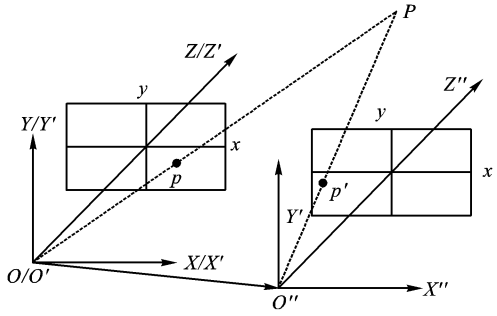

图1　立体成像系统

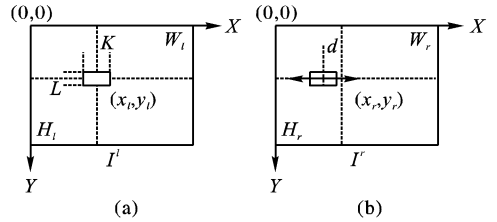

图2　相似值计算涉及的图块

2　并行算法

对上述匹配方体的计算过程按 Ian Foster 的 PCAM (Partitioning, Communication, Agglomeration and Mapping)[7] 方法进行并行算法设计,即分解、通信、汇聚和影射四个步骤。

首先进行数据域分解而不是进行功能分解,对于匹配方体 $Cube=\{c_{x,y,d}\}$,可以沿 X 轴、Y 轴或 D 轴进行分割。如果沿 D 轴分割则最大可分割数目较小(十几到几十),可扩展性受限,而且通信量比其他的分割方法大。沿 X 轴和 Y 轴大体相似,当计算节点数相等时通信量也相等,但沿 Y 轴分解可以在水平方向上进行一种快速计算,按 Y 轴分解如图3。

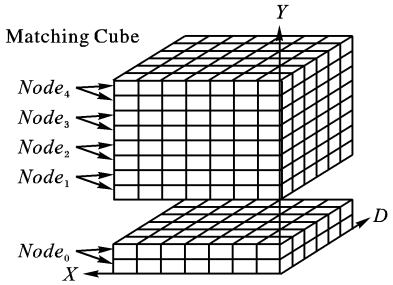

图3　匹配值任务分解

在分解方式确定后,可以安排通信过程,包括初始时的数据分配和最后的数据规整。初始处理需要将图像子块分到各个计算节点上,假设左右视图的图像大小为 $M*N$,视差深度为 D,计算节点总数为 $Node_N$,忽略子图边界上的处理,则通信量为$(2*M*N)$。对于节点 $Node_i$,需要向它传送 $I^l(x,y)$ 和 $I^r(x,y)$,其中 $\{0<x<M, N*(i-1)/Node_N<y<N*(i)/Node_N\}$,根节点将左右视图用 MPI_Scatter( ) 函数操作,将各个子图分发到对应的计算节点上。在计算结束后需要规整的数据量为$(M*N*D)$,利用 MPI_Gather( ) 函数操作从各个计算节点上收集所有的 $\{c_{x,y,d}\mid 0<x<M, N*(i-1)/Node_N<y<N*(i)/Node_N, 0<d<D\}$,总通信量为$(2*M*N+M*N*D)$。

后两步在这里是非常直接的,因为我们的分解过程是单层扁平的,可以直接映射到计算节点上。为了自动适应不同的计算节点数目,在任务启动后通过 MPI_Comm_size( ) 获得可用的计算节点数,并据此进行数据块的划分映射。$Rank=0$ 的根节点先将 I^l、I^r 的数据划分子块分发到各个计算节点,然后各个节点按照公式(1)计算局部的对应不同扫描行的 DSI 子图,形成局部的 Matching Cube,最后利用汇聚操作将所有子 Matching Cube 在根节点上形成总的 Matching Cube。

因此,匹配方体的计算过程将变成并行的计算方式。由于采用 SPMD(Single Program Multiple Data)[7] 编程模式,利用 MPICH 的 mpirun 命令启动的所有进程(由 -np 参数指定需要启动的进程总数)都运行相同的代码,但是在各自的代码中由于由系统传入的环境变量不同,使得它们启动后所获取的 *rank* 号不同。在代码中利用这个编号的不同,可以让它们分别执行不同的数据计算任务。该算法的框架如下:

1)启动所有 MPI 进程,进入 MPI 环境初始化,随即判断本进程 *rank* 是否为 0,若是则从硬盘中读入左右视图 I^l、I^r 到内部数据缓冲区,其他进程不需做操作;

2) 所有 *Rank* 为 0 ~ $Nodes_N$ - 1 的进程执行 MPI_Scatter ( )操作,$Node_N$ 是进程的总数,其中指定源节点为 *rank* 0,可将 I^l、I^r 划分成子图分发到对应的计算节点上,因此每个进程获得的子图为 $\{I(x,y)\mid 0<x<M, N*myrank/Node_N<y<N*myrank/Node_N\}$,其中 *myrank* 是个进程自己的 *rank* 编号,经过这个步骤后各个进程获得了不同的数据,接下来的处理方式是相同的,对应于 SPMD 编程模式;

3)所有 *Rank* 为0 ~ $Nodes_N$ - 1 的 MPI 进程各自根据公式(1) ~ (4) 独立计算所分配的子块,得出局部的所获得的子图对应的匹配方体 $SubCube(x,y,d)\mid\{0<x<M, N*myrank/Node_N<y<N*myrank/Node_N, 0<d<D\}$。对某一个固定的 d 值计算 DSI 的具体的计算过程为:将 I^r 图像右移 d 个像素,$I^r_{shift-d}(x,y)=I^r(x-d,y)$,逐个遍历 $I^l(x,y)$ 的所有元素,计算出 $I^l(x,y)$ 和 $I^r_{shift-d}(x,y)$ 所有像素的尺寸为 $(2K+1)\times(2L+1)$ 临域的相似值,不断改变 d 值计算出所有需要的 DSI 以构成匹配方体的子块;

4) 所有 *Rank* 为 0 ~ $Node_N$ - 1 的 MPI 进程执行 MPI_Gather( ) 操作,指出目的进程为 *rank* 0,将各 MPI 进程计算出的匹配方体的子块数据汇聚到 *rank* 0 的 MPI 进程中组装成完整的匹配方体数据,完成一次计算;

5)重复 1)计算下一对左右视图。

在集群环境下,基于 MPI 的并行计算性能受任务通信的影响很大,通信次数越多、数据量越大,则计算耗时越长,因此需要尽可能减少通信次数与通信数据量。对于匹配值的计算,可以有多种数据划分方法,比如沿 x 轴、y 轴或者 d 轴。首先排除沿 d 轴的划分,因为它的数据传输量将是另外两种方式的 $Nodes_N$ 倍(假设可获得的 CPU 数目为 $Nodes_N$,即可启动的总进程数目),沿 x 轴、y 轴方向的数据划分通信量相近,但沿 y 轴分解时,采用类似滑动窗口的技术计算某一 d 值的 DSI,可以加快计算。一般来说图像处理计算的可并行程度较高,可以有多种数据划分方式,需要权衡多方面因素才能确定。

现在绝大多数处理器都有浮点器件,因此可以认为加法和乘法所需的执行时间是相同的,一次求平方根需要 100 个时钟周期,则串行方式的(见公式(1) ~ (4))计算复杂度为:$D*(M*N*((K'*L'*3)+(K'*L'*2)+(K'*L'*2)+200))=7*D*M*N*K'*L'+200DMN$,其中 $K'=2K+1$,

$L' = 2 * L + 1$,即 $Osieral(7DMNK'L' + 200DMN)$。用 $Nodes$ 个进程并行实现后,计算复杂度下降为 $Oparallel((7DMNK'L' + 200DMN)/Nodes)$,同时增加了通信量 $(2 * M * N + M * N * D)$ 字节。如采用 Myrinet 网络,则启动时间 a,带宽为 B,等效于处理器时间为 $Ocomm(2 * Nodes * a + (2 * M * N + M * N * D)/B)$,由于 a 值为 10μs 量级,等效于 3.04Ghz CPU 的 3 × 104 个时钟周期,B = 1.8Gbps(理论值为 2Gbps),$1/B$ 等效于 3.04Ghz CPU 的 13.5 个时钟周期,并行计算总时间复杂度为:$Oparallel((7DMNK'L' + 200DMN)/Nodes + 2 * Nodes * 3 * 10^4 + (2 * M * N + M * N * D) * 13.5)$,即 $Oparallel((7DMNK'L' + 200DMN)/Nodes + Nodes * 6 * 104 + 27 * M * N + 13.5 * M * N * D))$。提速比 $SpeedUp$ 为:

$$SpeedUp = \frac{O_{sieal}}{O_{parallel}} = \frac{7DMNK'L' + 200DMN}{\frac{7DMNK'L' + 200DMN}{Nodes} + Nodes * 6 * 10^4 + 27MN + 13.5MND} = \frac{1}{\frac{1}{Nodes} + \frac{Nodes * 6 * 10^4 + 27MN + 13.5MND}{7DMNK'L' + 200DMN}}$$

也就是说当分母后面一项的值相对较小时才能接近线性提速比。

3 实验结果

实验在“深超-21c”(DeepSuper-21c)集群计算平台上进行,该集群具有 128 个计算节点,每个计算节点是双 CPU(Xeon 3.04GHz)的 SMP 机器,互联采用的带宽为 2Gpbs 的 Myrinet 2000 网络,采用 MPICH2.0 作为计算软件环境。测试用的图像参数为 M = 1024,N = 1024,D = 32,K = 3(即 K' = 7),L = 3(即 L' = 7)。经过测试 2 ~ 32 个并行进程的处理速度获得了如图 4(a)加速比曲线。

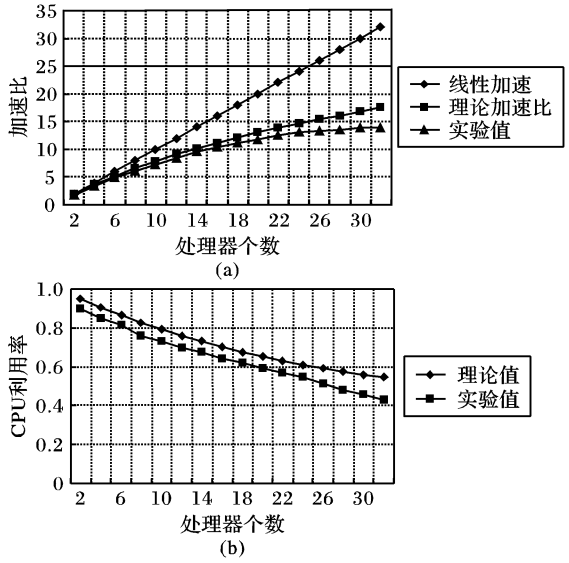

图 4 加速比与效率曲线

从图中可以看出,由于理论提速比模型比较精确,因此理论值与实验值基本吻合,并且可以看出当处理器个数增大时,处理器的利用率下降(图 4(b)),提速比的增速下降,这是由于通信所消耗的时间占总计算时间的比例增大而引起的,即 SpeedUp 表达式分母中第二项的比重增大。当处理器个数大于 16 时,特别是大于 24 时,提速比曲线已经相当平缓,增加处理器个数的收益明显下降,因此对于同一对左右视图不要超过 16 个处理器,如果有多对视图,可以按 16 个处理器为一组进行计算,即 256 个处理器可以分成 16 组来并行处理 16 个不同的立体视图对。

4 结语

立体视觉的各种计算复杂度很高,特别是各种性能较好的复杂算法耗时非常长。为了实现实时处理,一是牺牲部分性能简化成快速算法,例如文献[8]将双向匹配 BM(Bidirectional Matching)过程简化成单向过程;二是需要特殊硬件支持,如专用器件[3,4]。2006 年 2 月 SRI 研制的 STOC(Stereo on a Chip)是基于 FPGA(Field Programmable Gate Array)[9]的,并且将算法退化为基于特征的匹配才得以获得实时性能,但所用算法是固定的,不具有灵活性;三是采用并行计算,如大规模 MPP 使用的“细粒度”数据并行程序语言 parallaxis[5],这种平台和编程语言用户群数目少而且程序的可读性较差。目前,采用 SMP 或 PC 集群计算形式正在不断普及,由于 SMP 机器的可扩展性较差,因此后者更具有前景。随着 PC 集群架构的逐渐普及,在这种廉价平台上的立体视觉并行计算还有大量的工作需要开展,特别是新研究的算法不存在有效的简化算法时,这种并行方式可以大大加快计算速度,缩短获得试验结果的时间,因此选择当前主流的 MPI 环境,用 SPMD 模型来完成立体视觉中最核心的匹配方体并行计算,具有可扩展性好、代码可读性高的优点,对其他类似的处理可用相似的方法,通过构建廉价的 PC 集群和 MPI 并行编程实现并行计算。

进行了提速比分析,由于将通信启动时间和带宽因素转换成了 CPU 周期数,因此获得了比较准确的理论曲线,与实验数据比较吻合,可以作为以后相关处理的参考,将通信启动时间和带宽重新设置可以分析在 SMP 机器上的性能表现。

参考文献:

[1] ZITNICK CL, KANADE T. A cooperative algorithm for stereo matching and occlusion detection[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(7): 675 - 684.

[2] INTILLE SS, BOBICK AF. Disparity-space images and large occlusion stereo[R]. M. I. T Media Lab Perceptual Computing Group, Technical Report, No. 220, 1995.

[3] NEDEV NN. A real time 3D stereotelevision intelligent sensor system with improved efficiency[A]. Proceedings of ISIE '98[C]. IEEE International Symposium on Volume 1, 1998. 259 - 262.

[4] Kayaalp AE, Eckman JL. Near real-time stereo range detection using a pipeline architecture, Systems, Man and Cybernetics[J]. IEEE Transactions on Systems, Man and Cybernetics, 1990, 20(6): 1461 - 1469.

[5] BRÄUNL T, FEYRER S, RAPF W, *et al*. Parallel Image Processing [M]. Berlin Heidelberg: Springer-Verlag, 2003.

[6] GROPP W, LUSK E. Installation and Users Guide for Mpich, a Portable Implementation of MPI[R]. Technical Report ANL-01/x, Argonne National Laboratory, 2001.

[7] FOSTER I. Designing and Building Parallel Programs[M]. Addison Wesley, Pearson Education, Inc, 2002.

[8] STEFANOA LD, MARCHIONNI M, MATTOCCIA S, *et al*. A fast area-based stereo matching algorithm[EB/OL]. http://www.cipprs.org/vi2002/pdf/s3-2.pdf, 2004.

[9] Videre Design Home Page[EB/OL]. www.videredesign.com, 2006.