第 35 卷 第 1 期 计 算 机 工 程 2009 年 1 月
Vol.35 No.1 Computer Engineering January 2009
·软件技术与数据库· 文章编号：1000—3428(2009)01—0046—03 文献标识码：A 中图分类号：TP391.41

基于 C6000 的滑动窗口图像处理算法存储优化

张 帆，窦 勇，邬贵明
(国防科技大学计算机学院，长沙 410073)

摘 要：片外存储器和片内存储器的数据传输是数字信号处理系统性能提升的瓶颈。针对图像处理中的滑动窗口类问题，该文提出一种有效的存储调度优化方法，分为 3 步：预取数据到快速局部存储器，减少冗余读入及数据传输和处理重叠。在 TMS320DM642 DSP 上应用了该方法，实验结果表明，与优化前相比加速比为 30~70。

关键词：存储优化；图像处理；滑动窗口；C6000 数字信号处理器

Memory Optimization of Sliding Window Image Processing Algorithm Based on C6000

ZHANG Fan, DOU Yong, WU Gui-ming
(School of Computer Science, National University of Defence Technology, Changsha 410073)

【Abstract】 Data transfer between internal memory and external memory are bottlenecks in improving DSP performance. A memory optimization method is presented for sliding window algorithm in image processing. There are three steps in the method: prefetching data to local memory, reducing redundancy read and paralleling data transfer and data processing. It uses the method on TMS320DM642 DSP. Experimental result shows that the performance of program can get 30~70 speedup.

【Key words】 storage optimization; image processing; sliding window; C6000 DSP

图像处理在日常生活和科学研究各个方面有广泛应用，如条码识别、电子警察、人脸检测。图像处理应用中有一类算法具有滑动窗口特点，即对图片里一个窗口内的像素点进行处理，处理结束后将窗口滑动到新的位置，再处理，直到处理完整幅图像。这类算法有广泛的应用，如边缘检测和滤波。本文讨论在 TMS320C6000 平台下对滑动窗口类问题的存储优化技术，提出一种存储优化方法，包括 3 步：预取数据到快速局部存储器，减少冗余读入及数据处理和传输重叠。将本文提出的存储优化方法应用到边缘检测和中值滤波，能够达到 30~70 的加速比。

1 TMS320C6000 存储系统

TMS320C6000系列DSP[1-2]是TI公司开发的高性能32位DSP，DM642是该系列中一款面向媒体应用的定点DSP。

TMS320C6000 系列 DSP 采用多级存储体系结构，见图 1。整个存储系统分为 4 级：通用寄存器，L1 Cache, L2 Cache/SRAM 和片外 SDRAM。L1 Cache 可与通用寄存器直接交换数据，用户不能直接控制。L1 Cache 分为程序 Cache L1P 和数据 Cache L1D。L2 Cache 为程序和数据共用，用户可配置为 Cache 或 SRAM。片外存储器一般为 SDRAM，使用 DMA 方式与 SRAM 交换数据。

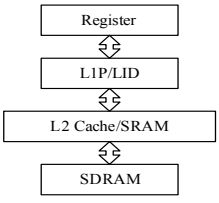


图 1 TMS320C6000 存储系统体系结构

2 滑动窗口

滑动窗口算法的一般描述是：在规模为 $W\times H$ 的图像中按一定规律移动 $w\times h$ 的窗口($W>>w$, $H>>h$)，如图 2 所示，对窗口内像素点的像素值进行一系列运算，运算结束后窗口向右或向下移动一步，直到完成对整幅图像的处理。中值滤波和边缘检测 2 个算法是典型的滑动窗口算法。

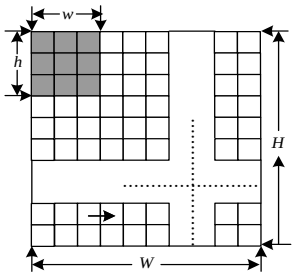


图 2 滑动窗口

3 存储优化

在 DSP 中 CPU 访问片内的 SRAM 比访问片外的 SDRAM 快，如 DM642 访问片内 SRAM 只要 5 个周期左右，而访问片外 SDRAM 约要 105 个周期。因此，尽量将图片数据放到 SRAM 中可以减少程序运行时间，但 SRAM 容量有限，如 DM642 的 SRAM 最大为 256 KB，不能存储较大图片，完整图片只能存储在片外的 SDRAM，如何进行片外和片内存储调度成为限制程序高效运行的一个关键问题。下面具体描述存储调度方法。

作者简介：张 帆(1980－)，男，硕士，主研方向：图像处理，VLIW；窦 勇，教授、博士生导师；邬贵明，博士

收稿日期：2008-06-09 **E-mail：**zhangfanbluesea@163.com

步骤 1 预取数据到快速局部存储器

通常，原始数据放到片外SDRAM。用户在编写C语言程序时，往往使CPU直接从片外SDRAM取数处理，忽略了多级存储体系结构的优点，造成资源浪费。根据多级存储体系结构的特点，可以将一部分马上要用到的数据以DMA方式预取到SRAM，CPU从SRAM里取数进行处理，缩短了CPU的取数时间，从而可以提高程序的执行效率[3]。原始图片以图 2 为例，假设高为H，宽为W，滑动窗口高为h，宽为w，一次迭代预取h行。以后算法都用此假设。

优化后，首先将h行数据预取到SRAM，处理这h行数据，结果暂存在SRAM，处理完h 行数据后，将结果以DMA方式送到SDRAM。若CPU直接从SDRAM取 1 Byte需要Ts拍，取N Byte共需要节拍数$T_{sdram}=N\times Ts$。CPU从SRAM取 1 Byte需要Tr拍，从SDRAM以DMA方式将大量数据预取到SRAM，平均Td拍传送一个 Byte，N Byte共需要节拍数$T_{sram}=N\times(Tr+Td)$。

加速比为

$$T_{sdram}/T_{sram}=(N\times Ts)/[N\times(Tr+Td)]=Ts/(Tr+Td)$$

以 DM642 为例，Ts 约为 105，Tr 约为 5，Td 为 1/4，代入上式，加速比约为 21。可以明显提高传输效率。

步骤 2 减少冗余读入

预取数据大大提高程序效率的同时产生了冗余的数据读入。减少这些重复读入可以提高性能[4]。

若图片高为 H，宽为 W，滑动窗口高为 h，为便于说明取 3，每次迭代预取 3 行。如图 3 所示，优化前，第 1 次迭代预读入第 1 行~第 3 行，第 2 次迭代预读入第 2 行~第 4 行。在这 2 次迭代中，第 2 行和第 3 行读入 2 次，存在冗余读入。以后的预读入类似：第 m 次迭代预读入第 m 行~第 m+2 行。第 m+1 次迭代预读入第 m+1 行~第 m+3 行。在这 2 次迭代时，第 m+1 行和第 m+2 行被读入 2 次，存在冗余读入。

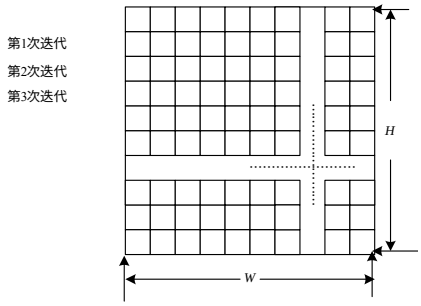


图 3　优化前各次迭代读入

如图 4 所示，冗余可采用如下方式去除：第 1 次迭代时读入第 1 行~第 3 行，以后每次迭代只读入一行，即第 m 次迭代读入第 m+2 行。

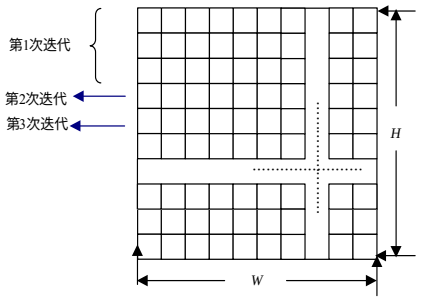


图 4　优化后各次迭代读入

若滑动窗口高为h，优化前累计要迭代H-h＋1 次，每次迭代读入h行，共读入行数：$H_b=h\times(H-h+1)$。优化后，也要迭代H-h＋1 次，但除了第 1 次读入h行，以后每次迭代只需读入 1 行，共读入行数：$H_a=H$。

优化前与优化后读入行数比为

$$H_b/H_a=h\times(H-h+1)/H\approx h(H>>h)$$

优化后的算法如下：

算法 1 减少冗余读入优化后的中值滤波

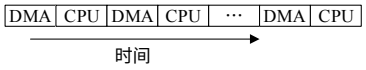
输入：数组 *SS*，存储原始图像，在 SDRAM
输出：数组 *SD*，存储结果图像，在 SDRAM
符号：Mean(*A*,*B*)对数组 *A* 中值滤波，结果放到数组 *B*
数组 *RS*，存储马上要用到的数据，在 SRAM
数组 *RD*，存储 Mean()运算的结果，在 SRAM
步骤：

```
(1)Load SS,RS, (h-1)×W     //预取 h-1 行
(2)For(i=h;i++;i<H-1) {
(3)Load SS＋i×W,RS, W)   //预取 1 行
(4)Mean(RS,RD)
(5)Store RD,SD＋i×W}
```

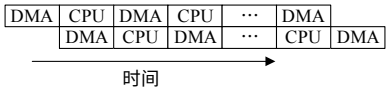
优化后，首先预取 *h*-1 行到 SRAM，然后进入循环，预取一行，对 SRAM 内的 *h* 行处理，输出结果，如果没有处理完所有数据则循环。

步骤 3 数据传输和处理的重叠

每次迭代分为 2 大部分：数据传输和数据处理。用到的功能部件分别为 DMA 和 CPU，2 个部件串行工作。若将数据传输和数据处理重叠，让 2 个功能部件并行工作，可以提高性能。优化前，CPU 和 DMA 工作的时序关系如下：



优化后，CPU 和 DMA 工作的时序关系如下：



因为迭代之间没有相关，所以可以将整个流程从中间分为 2 部分。若共有 2n 次迭代，将前 n 次迭代分为 *PA*，后 n 次迭代分为 *PB*，同时进行。在 *PA* 部分进行数据处理时，*PB* 部分进行数据传输，在 *PB* 部分进行数据处理时，*PA* 部分进行数据传输。优化后的算法如下：

算法 2 数据传输和处理重叠优化后的中值滤波

输入：数组 *SS*，存储原始图像，在 SDRAM
输出：数组 *SD*，存储结果图像，在 SDRAM
符号：Mean(*A*,*B*)对数组 *A* 中值滤波，结果放到数组 *B*
数组 *RS*1，存储马上要用到的数据，在 SRAM
数组 *RS*2，存储马上要用到的数据，在 SRAM
数组 *RD*1，存储 Mean()运算的结果，在 SRAM
数组 RD2，存储 Mean()运算的结果，在 SRAM
步骤：

```
(1)Load SS,RS1, h×W   //预取 h 行
(2)Load SS＋(H/2)×W,RS2, h×W   //预取 h 行
(3)Mean(RS1,RD1)
(4)Wait Load      //等待 Load 完成
(5)For(i=0;i++;i<H/2) {
(6)Store RD1,SD＋i×W     //输出结果
(7)Load SS＋(i＋h)×W,RS1, W     //预取 1 行
(8)Mean(RS2,RD2)
(9)Wait Load      //等待 Load 完成
(10)Store RD2,SD＋(i＋H/2)×W     //输出结果
```

(11)Load SS＋(i＋h＋H/2)×W,RS2, W　//预取 1 行
(12)Mean(RS1,RD1)
(13)Wait Load}　　//等待 Load 完成

优化后，首先运行(1)，预取图片前 *h* 行到 *RS*1。运行(2)，预取从 *H*/2 行开始的 *h* 行到 *RS*2，同时，运行(3)，处理 *RS*1 中的数据，结果存到 *RD*1。(4)等待数据传输完成。从(5)进入循环。运行(6)，启动 DMA，将 *RD*1 中的运算结果送到 SDRAM，运行(7)，再预取一行到 *RS*1，在启动 DMA 的同时运行(8)，处理 *RS*2 中的数据，结果存在 *RD*2。(9)等待(6)和(7)完成。再启动 DMA，运行(10)，将 *RD*2 中的运算结果送到 SDRAM，运行(11)，预取一行到 *RS*2，在启动 DMA 的同时运行(12)，处理 *RS*1 中的数据。(13)等待(8)和(9)完成。如此循环，直到处理完所有数据。

4　试验结果和总结

在 DM642 DSP 上对中值滤波和边缘检测程序进行了优化，以中值滤波为例说明 3 个优化步骤和对最终优化结果所做的贡献程度，滑动窗口为 3×3。预取前后节拍数比较见表 1，优化前指未经过任何优化时程序运行的节拍数；预取指经过预取后程序运行的节拍数；加速比指优化前运行时间与预取优化后运行时间的比值。经过预取优化，加速比约为 15。图片越大，单次 DMA 预取的数据越多，传输效率越高，所以，加速比越大。

表 1　预取优化效果

图片规模	优化前节拍数	预取节拍数	加速比
800×600	47 689 868	2 758 014	17.3
480×360	16 843 519	1 111 801	15.1
320×240	7 396 577	559 590	13.2

在预取优化基础上，进行减少冗余读入优化，性能进一步得到提高，如表 2 所示。加速比为 1.5。

表 2　减少冗余读入优化效果

图片规模	预取节拍数	减少冗余读入节拍数	加速比
800×600	2 758 014	1 892 066	1.5
480×360	1 111 801	755 906	1.5
320×240	559 590	372 729	1.5

在前 2 次优化基础上，经过数据传输和处理重叠优化，性能又进一步提高。优化前后的节拍数比较如表 3 所示。优化后的加速比为 1.5。

表 3　数据传输和处理重叠优化效果

图片规模	减少冗余读入节拍数	数据传输和处理重叠节拍数	加速比
800×600	1 892 066	1 271 648	1.5
480×360	755 906	520 618	1.5
320×240	372 729	265 480	1.4

从表 3 可以看出，预取优化的作用最大。除了中值滤波，还采用该方法优化了边缘检测，优化前及优化后的节拍数如表 4 所示。表中加速比指优化前程序运行节拍数与经过 3 次优化后程序运行节拍数比值，加速比约为 60。

表 4　边缘检测优化效果

图片规模	优化前节拍数	预取节拍数	减少冗余读入节拍数	数据处理和传输重叠节拍数	加速比
800×600	78 463 910	2 583 560	1 678 737	1 074 755	73.0
480×360	27 900 526	1 050 500	673 547	451 039	61.9
320×240	12 298 275	527 645	339 028	234 172	52.5

中值滤波和边缘检测经过所有 3 步优化后，最终的性能加速比如图 5 所示。左边 3 列表示对 3 个不同规模图片进行中值滤波的加速比。右边 3 列表示对 3 个不同规模图片进行边缘检测的加速比。

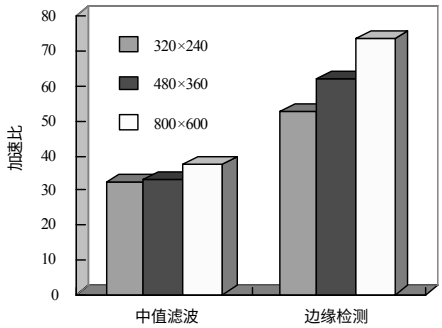


图 5　优化效果

本文介绍了图像处理中的滑动窗口问题的原理，提出了一个存储调度优化方法。试验结果证明该优化方法可以有效提高程序执行效率。

参考文献

[1] TMS320C6000 DSP Cache User's Guide[Z]. (2003-05-06). www.ti.com.

[2] TMS320C6000 CPU and Instruction Set Reference Guide[Z]. (2003-07-06). www.ti.com.

[3] Banerjee S, Sheikh H R, John L K, et al. VLIW DSP vs. Superscalar Implementation of a Baseline H.263 Video Encoder[C]//Proc. of IEEE Asilomar Conf. on Signals, Systems, and Computers. Pacific Grove, CA , USA: IEEE Press, 2000: 1665-1669.

[4] Karadayi K, Golston J, Gove R J, et al. Strategies for Mapping Algorithms to Mediaprocessors for High Performance[J]. IEEE Micro, 2003, 23(4): 58-70.

（上接第 45 页）

参考文献

[1] Beeferman D, Berger A. Agglomerative Clustering of a Search Engine Query Log[C]//Proc. of the 6th International Conference on Knowledge Discovery and Data Mining. Boston, Massachusetts, USA: [s. n.], 2000: 407-416.

[2] Chan Wing-Shun, Leung Wai-Ting, Lee Dik-Lun. Clustering Search Engine Query Log Containing Noisy Clickthroughs[C]//Proc. of the International Symposium on Applications and the Internet. [S. l.]: IEEE Press, 2004: 305-308.

[3] 曾　春, 邢春晓, 周立柱. 个性化服务技术综述[J]. 软件学报, 2002, 13(10): 1952-1961.

[4] 谭　琼, 李晓黎, 史忠植. 一种实现搜索引擎个性化服务的方法[J]. 计算机科学, 2002, 29(1): 23-25.

[5] Claypool M, Le P, Waseda M, et al. Implicit Interest Indicators[C]//Proc. of the ACM Intelligent User Interfaces Conference. New York, USA: ACM Press, 2001: 14-17.

[6] Liang Ting-Peng, Lai Hung-Jen. Discovering User Interests from Web Browsing Behavior: An Application to Internet News Services[C]//Proc. of the 35th Annual Hawaii Int'l Conference on System Sciences. [S. l.]: IEEE Press, 2002: 2718-2727.