文章编号:1001-5078(2011)08-0936-05 ·图像与信号处理·

基于DM642的KLT跟踪算法的实现及优化

刘 军,梁久祯,柴志雷
(江南大学物联网工程学院智能系统与网络计算研究所,江苏 无锡 214122)

摘 要:Kanade-Lucas-Tomasi(KLT)算法是基于图像特征点的跟踪算法,由目标对象特征点提取,特征点跟踪两部分组成。本文首先阐述了KLT算法的基本原理,分析了影响算法执行速度的主要原因。分析表明KLT算法的操作主要集中在乘加运算和循环,图像卷积运算和循环占用的执行时间比较长。针对TMS320DM642 DSP的硬件平台特点,提出了算法优化的若干策略。通过配置编译环境,合理安排数据类型,消除存储器相关性,使用内联函数以及分解多层循环等方法,对算法的实现进行了优化。实验结果表明,优化后代码执行速度是优化前的3倍多。

关键词:DM642;KLT;运动跟踪;优化;图像处理
中图分类号:TP391 **文献标识码**:A **DOI**:10.3969/j.issn.1001-5078.2011.08.024

Implementation and optimization of KLT tracking algorithm based on DM642

LIU Jun, LINANG Jiu-zhen, CHAI Zhi-lei
(ISNC, School of Internet of Things Engineering, Jiangnan University, Wuxi 214122, China)

Abstract: KLT is a tracking algorithm based on image feature points, which is composed of two parts, namely the feature point extraction and the feature point tracking. In this paper, the basic principle of the KLT algorithm is proposed, and the main factors which influence the speed of the KLT algorithm are analyzed. It is found that the multiplication-addition and the loop operations cost the most processing time in the KLT algorithm. The image convolution operation and the implementation of loops take much more time. A serise strategies of the algorithm optimization are proposed considering the hardware platform of the TMS320DM642. The algorithm is implemented optimizely, by configing the compile environment, arranging the data types reasonably, eliminating the memory correlation, using the intrinsics and decompositing the number of loops. Experimental results show that the execution speed of the optimized code is three times faster than that without optimization.

Key words: DM642; KLT; motion tracking; optimization; image processing

1 引 言

视频图像运动跟踪是视频运动分析的重要研究领域,也是进行视频运动识别和理解的基础。运动目标跟踪算法可分为基于对比度的跟踪算法,基于图像匹配跟踪算法和基于模板相关匹配跟踪算法。其中基于图像匹配的方法又可以分为基于运动模型和基于图像特征两类方法。KLT算法是基于图像特征的跟踪算法[1-2],在视频图像跟踪、视线跟踪、疲劳驾驶检测、三维重建等研究领域中应用广泛。

由于KLT算法计算量大,且现在多是基于通用CPU的实现,如OPenCV将算法做成库函数提供给用户,所以算法的运行速度比较慢,影响检测效果。

基金项目:江苏省自然科学基金(No. BK2008098)资助。
作者简介:刘 军(1985-),男,硕士,主要从事基于DSP视频图像处理的研究。E-mail:Jerryliu119@gmail.com
收稿日期:2011-01-19;**修订日期**:2011-04-08

为了加快算法运行速度,近年来 Sudipta Sinha 等人将 KLT 算法在 GPU 上实现[3-4],但是并没有找到合适的办法将特征点选取部分也在 GPU 上实现,且对低帧率视频图像的处理效果不好。为了解决 Sudipta Sinha 等人遇到的问题,Julius Fabian Ohmer 和 Nicholas J. Redding 提出用 MonteCarlo 特征点选取方法来代替原来的特征点选取方法[5],但是特征点管理这一部分没有办法在 GPU 上实现;Christopher Zach 等人则改进了 Sudipta Sinha 等人的实现,使其能够处理低帧率视频图像[6]。但 KLT 算法的并行性问题并没有得到很好的解决,且 GPU 价格相对较高,难以广泛应用。

随着数字信号处理技术的发展,数字信号处理器的并行能力越来越高,价格越来越低,使数字信号处理器得到了广泛的应用。TMS320DM642 DSP 就是性价比极高的一款高速数字信号处理器。TMS320DM642 是 TI 公司专门针对视频图像处理设计的高速数字信号处理器,具有功能强大的乘加运算器和并行处理结构,在视频图像处理中应用广泛。本文针对 TMS320DM642 的硬件结构特点,研究 KLT 算法移植于 DM642 平台后如何充分利用硬件资源,将算法并行执行,提高代码运行速度。本文给出了算法移植优化的通用性原则和针对 KLT 算法的特点采取的优化措施。

2 KLT 算法原理及其计算复杂性分析

在视频序列中,图像的变化比较复杂。通常我们只能通过图像灰度函数 $f(x,y,t)$ 来描述图像,其中 (x,y) 是空间变量,t 是时间变量,且通常 t 都是离散的有界的变量。但是如果视频图像帧在很短的时间间隔内采集,那么图像帧之间通常会存在很强的相关性,且差异值比较小。KLT 算法就是建立在这种假设之上。算法以待跟踪窗口 A 在连续图像帧间的灰度差平方和(sum of squared intensity differences,SSD)作为度量。

2.1 算法原理

对灰度图像 I,KLT 算法首先在该图像选取一个包含特征纹理信息的特征窗口 $A \subseteq I$,在 t 时刻灰度图 I 表示为 $f(x,y,t)$,$(t+\delta)$ 时刻对应的灰度图 J 表示为 $f(x,y,t+\delta)$,两幅灰度图满足式(1)所示的关系:

$$f(x,y,t+\delta) = f(x-\Delta x, y-\Delta y, t) \tag{1}$$

令 $d=(\Delta x,\Delta y)$,则上式表示在 $(t+\delta)$ 时刻图像的每个像素点,都可由 t 时刻图像的每个像素点平移向量 d 得到。

设在 $(t+\delta)$ 时刻灰度图像的特征窗口为 $B(X)=B(x,y,t+\delta)$,其中 $X=(x,y)$ 为像素点的坐标,t 时刻灰度图像的特征窗口为 $A(X-d)=A(x-\Delta x, y-\Delta y\ t)$。则窗口 A 和 B 的关系如式(2)所示:

$$B(X) = A(X-d) + n(X) \tag{2}$$

其中,$n(X)$ 是在时间 δ 内由于采集环境变化而产生的噪声。

将 $n(X)$ 平方后在整个窗口上积分,就得到了窗口图像的灰度差平方和(SSD),即式(3):

$$\begin{aligned}\varepsilon &= \iint_{\nu} n(x)^2 \omega(X)\,\mathrm{d}X \\ &= \iint_{\nu} (B(X)-A(X-d))^2 \omega(X)\,\mathrm{d}X\end{aligned} \tag{3}$$

其中,$X=[x,y]^{\mathrm{T}}$,$d=[d_x,d_y]^{\mathrm{T}}$,$\omega(X)$ 是权重函数,通常可以取 1;为了强调中心部分,$\omega(X)$ 也可以使用高斯分布函数[2]。本文中 $\omega(X)$ 采用高斯分布函数。

当采集速度足够快的时候,d 与 X 相比是一个足够小的量的时候,则 d 可忽略,采用泰勒展开来处理 $A(X-d)$,令 $g=\left(\frac{\partial A}{\partial x}(X),\frac{\partial A}{\partial y}(X)\right)^{\mathrm{T}}$,得式(4):

$$A(X-d) = A(X) - g^{\mathrm{T}} d \tag{4}$$

将式(4)带入式(3),然后式(3)两边同时对 d 求导,当导数值为 0 时,ε 取得最小值,也就是求得所需要的 SSD,如式(5)所示:

$$\frac{\partial \varepsilon}{\partial d} = \iint_{\nu} (B(X)-A(X)+g^{\mathrm{T}}d) g w(X)\,\mathrm{d}X = 0 \tag{5}$$

对每连续的两帧灰度图解方程(5),则可以解出特征窗口的位移 $d=(\Delta x,\Delta y)$。由于在式(4)处使用了泰勒展开引入了误差,计算得到的 d 不准确,所以使用 Newtom-Raphson 迭代求 d 来消除误差。

2.2 KLT 算法计算特性分析

KLT 算法主要包括特征点选取和特征点跟踪两个部分。为了增加算法的稳定性,首先将图像平滑,平滑的实质是卷积运算。特征点选取步骤中先计算平滑图像的梯度,再遍历整个图像选取纹理丰富的窗口,并对选取的特征点进行排序。求梯度主要用到的是卷积运算和循环,其他处理主要涉及乘加运算和循环。

在特征点跟踪步骤中,为了加快跟踪的速度,同时保证跟踪到的特征点具有全局性,将平滑图像转换为金字塔图像,再求每一级金字塔图像的梯度,然后选取到的特征点逐点逐级跟踪,由于在式(4)处使用了泰勒级数,产生了误差,所以跟踪的时候使用 Newtom-Raphson 迭代来消除误差;求梯度是卷积操作,跟踪主要是迭代和解矩阵,其他处理主要是乘加操作和循环;对于跟踪成功的特征点需要用仿射运算进行一致性检查,这是一个解 6 阶矩阵的过程,存在多重循环。所以 KLT 算法的操作主要集中在乘

加运算和循环,图像卷积运算和循环占用的执行时间比较长。

3 基于DM642的KLT算法实现和优化

由于影响KLT算法执行速度的主要是乘加运算和循环,所以优化KLT算法,主要就是提高乘加运算和循环的速度。

3.1 算法实现

算法的实现参考了文献[7],具体实现不做累述,主要介绍对DM642存储器的配置,包含两个方面:将频繁使用的数据放到片内缓存中,加快数据读写速度,减少CPU等待时间;使用EDMA让数据处理和存取并行执行[8]。DM642片内是两级Cache结构,第一级是相互独立的程序Cache(L1P)和数据Cache(L1D)。第二级Cache L2是一个程序和数据公用缓存,大小为256 KB,缓存的一部分可用作普通的存储空间来使用,即L2 SRAM。图1是DM642存储结构图及读取速度对比图,双箭头表示各部分可以相互传递数据。由图1可知,存储空间大小和读取速度成反比[9]。

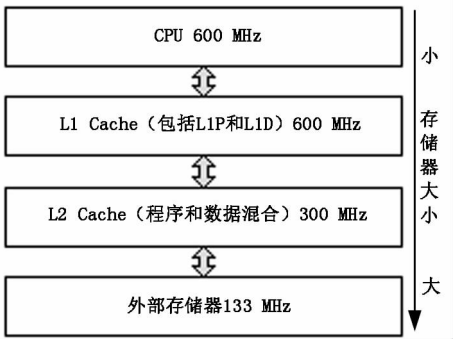

图1 DM642存储结构及读取速度对比图

实验中将L2 Cache全部设置为SRAM,且将其等分为两个缓冲区,构成"乒乓结构"。在任意时刻,总是一个缓存区受处理器控制处理数据,一个缓冲区受EDMA控制读入数据。则数据总是由高速缓存L2SRAM中读取,提高了数据访问速度;EDMA存取数据与处理器处理数据并行。

实验所采集的图像大小是720×576,数据格式是YUV(4:2:2)。UV分量的数据大小为207360 B,小于SRAM的256 KB,则其数据可以全部放在SRAM中,处理器可以直接从SRAM中读取数据进行处理,处理完毕之后再用EDMA搬移到片外存储器中,这样数据的读取速度提高了至少一倍。但是Y分量的数据大小为414720 B,大于SRAM的256 KB,无法放入片内存储器上处理。需要考虑YUV三个分量的处理顺序。先在片内处理U分量,处理完之后处理器从外部存储器读入Y分量数据进行处理,同时启动EDMA将U分量的数据写出到外部存储器,处理器和EDMA就可以并行工作,最后处理V分量,在送数据到显示任务显示和等待下一帧到来的间隙中EDMA搬运V分量数据,使处理器和EDMA再次并行。

3.2 合理设计数据类型

在DM642上实现KLT算法时,在保证数据不溢出的同时,应尽量使用小数据类型。例如图像尺寸用16位的short数据表示已经足够,不必使用int;需要做乘法的整形数据,应该尽量使用16位的短数据[10],因为DM642是32位的处理器,一次能做两对16位数的乘法,可以加快运算速度。

3.3 软件流水

DM642是基于超长指令字(VelociTI.2™)的结构,CPU核有8个独立并行的功能单元,理论上最大可以有8条指令同时运行[9]。在循环中,如果这8个功能单元同时运行,则循环执行速度会大大提高,这种运行方式叫做软件流水。其运行示意图如图2所示。

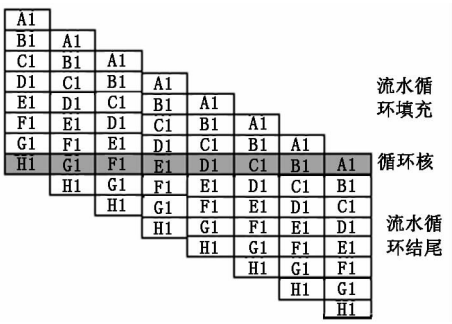

图2 软件流水示意图

假设一个循环里有八条指令A,B,C,D,E,F,G,H,一个周期最多可以执行8条指令,即图2的阴影部分。该部分叫做循环核。循环核之前的部分叫流水循环填充(pipelined loop prolog),之后的叫流水循环结尾(pipelined loop epilog)。要实现高质量的软件流水首先需要选择合适的编译器选项,让编译器执行软件流水。经过比较后,最终本程序选择了如下编译器选项:

-o3 最高程度的文件级优化;

-pm -op3 程序级优化,程序中有模块间相互函数调用,但不会修改全局变量;

-mh 允许投机运行,减少代码的尺寸;

-mt 告知编译器源程序没有使用混叠技术。

除了使用上述的编译器选项,还需要对C代码进行如下修改,使其能够使用软件流水技术:

(1)循环不能包含除内联函数外的任何代码调用;

(2)循环不能有条件中止、跳转等使循环提前

结束的指令；

(3)循环计数器必须递减。

3.4 消除存储器相关性

C编译器会安排没有存储器相关性的指令并行执行，如果C编译器不能确定指令之间是否有存储器相关性，就会认为这些指令相关，安排它们串行执行。对于编译器不能判定，且不存在存储器相关性的指令，需要使用关键字 const 和 restrict 来告知编译器，该数组或者指针所指的存储区不存在相关性。用 const 和 restrict 修饰的数组或者指针就能够读写同时进行，达到并行的目的。在本实验中，计算梯度，图像跟踪等使用的数据都用 const 和 restrict 消除相关性。实验结果表明使用 const 和 restrict 的程序与不使用 const 和 restrict 的程序编译后执行的速度相差1倍以上。

3.5 数据打包

DM642访问存储器比较消耗时间[8]，要提高程序数据的吞吐率，就必须使每条 Load/Store 指令访问多个数据。KLT算法处理的图像数据大部分都是8位数据，所以可以使用字一次访问4个字符型数据，对应的函数为_amem4(void * prt)或者_mem4(void * ptr)，优化前一次传递一个数据，优化后一次性传递4个数据，速度提高了4倍。

3.6 使用内联函数

内联函数执行效率高，代码长度短，且使用方便[8]。上一步中采用数据打包技术存取数据，对于打包读入的数据，需要使用对应的内联函数进行计算，例如使用 dotpu4 或者 dotpsu4 计算4字节乘加运算。使用内联函数的时候要注意尽量使用专为C64系列DSP所设计的内联函数，这样可以充分发挥DM642的硬件功能。

3.7 图像库函数的调用

TI公司为视频、图像常用处理提供了库函数IMGLIB。这些库函数已经优化，运行效率高，消耗时间短。由KLT算法计算复杂性分析可知，算法用到卷积的次数很多，所以本实验中使用了库函数中的卷积函数，卷积核为11。

4 实验结果分析

实验所用测试硬件平台是TMS320DM642，主频为600 MHz，集成开发环境为CCS 2.20.18。

图3是一段视频序列的第6帧和第7帧跟踪效果，物体发生刚性运动，即内部不发生形变，KLT算法对这一类运动的跟踪效果很好[2]。视频图像尺寸为720×576，每秒25帧图像。第1栏为原始图像，第2栏为优化前跟踪效果，第3栏为优化后跟踪效果。在第6帧中选择100个特征点，优化前因为处理速度比采集速度慢，在第7帧中只跟踪到91个特征点；优化后处理速度加快，在第7中跟踪到99个特征点，丢失特征点如图3第3栏所示，用方框标出，位于“嵌”字下部，是由于运动的不稳定，该点在相邻帧间发生漂移造成的。程序优化前后跟踪100个特征点，算法的主要子函数执行时间如表1所示。表1中特征点选择的时间开销主要包括图像平滑和卷积，特征点跟踪的执行时间主要包括图像平滑和卷积，计算金字塔图，计算图像灰度差异值，特征点一致性检查，算法执行总时间由特征点选择和特征点跟踪两部分组成。

图3 跟踪刚性运动效果图

表1 KLT算法主要子函数在DSP上运行的时间

函数名称	函数功能	执行时间/ms	
		优化前	优化后
IMG_conv_11	用于图像平滑和卷积	18.3	4.2
KLT Compute Pyramid	用于计算金字塔图	21.6	5.3
ComputeIntensity Difference	用于计算图像灰度差异值	15.2	3.2
Am_track Feature Affine	特征点一致性检查	60	18.4
KLT Select Good Features	特征点选择	50.5	15.1
KLT Track Features	特征点跟踪	101.4	31.7

由表1可知，KLT算法的主要子函数优化前后在DM642上运行的时间差别很大。从实验结果可知，将KLT算法移植于DM642平台优化后，算法运行速度得到明显提高。

图 4 是斯坦福大学视频库中视频 foreman_qcif 的第 5 帧和第 6 帧的跟踪效果,视频图像尺寸为 176 × 144,每秒 25 帧图像。其中第 1 栏为原始图像,第 2 栏为优化前跟踪效果图,选择 86 个特征点,其中 69 个特征点跟踪到;第 3 栏为优化后跟踪效果图,选择 86 个特征点,其中 79 个特征点跟踪到。由于视频中脸发生的是非刚性运动,所以有特征点有漂移和丢失的现象发生,但是优化后比优化前多跟踪到 10 个特征点。

图 4 跟踪非刚性运动效果图

表 2 是 KLT 算法在通用 CPU,OpenCV 算子,GPU 及其在 DSP 上运行时间的对比。算法在 DSP 上的执行时间是表 1 中算法优化后特征点选择和特征点跟踪两部分执行时间之和;算法在通用 CPU,OpenCV 的运行时间的测试硬件平台为 Intel Core 2 E7400 2.80 GHz,GPU 的运行时间来自文献[3]。GPU 通常和 CPU 组成一个模块,多个模块之间通过网线通信实现并行,很难在一块电路板上集成多个处理器,在嵌入式应用中受限,而 DSP 可以通过串行 RapidIO(SRIO)等方式在一块电路板上集成多个 DSP,GPU 的核不能单独控制,DSP 可以通过汇编语言编程控制单个核,可以更好的安排并行,且GPU 的价格是 DM642 的几倍,不利于控制成本;所以 GPU 处理的速度虽然比 DSP 快,但 DSP 有其特殊的优势。

表 2 KLT 算法在各个平台运行时间(ms)

	通用 CPU	OPenCV	GPU	DSP
硬件平台	E7400 2.80 GHz	E7400 2.80 GHz	Nvidia Geforce 7900 GTX	TMS320DM642
算法执行时间	82.87	78.13	40	46.8

5 结 论

本文首先阐述了 KLT 跟踪算法的基本原理,在分析算法特点的基础上,找到提高算法性能的“瓶颈”,结合 DM642 的硬件结构特点,采用软件流水,合理设置编译器优化选项,数据打包,指定存储器相关性,调用图像处理库函数和存储器优化等策略对 KLT 算法的“瓶颈”进行了重点优化。实验结果表明,通过优化,算法的执行速度大幅度提高。

参考文献:

[1] Bruce D Lucas, Takeo Kanade. An iterative image registration technique with an application to stereo vision [C]//7th International Joint Conference on Artificial Intelligence, Vancouver, 1981, 8(24 – 28): 674 – 679.

[2] Jianbo Shi, Carlo Tomasi. Good features to track [C]// IEEE Conference on Computer Vision and Pattern Recognition, Seattle, 1994, 1(21 – 23): 593 – 600.

[3] Sudipta N Sinha, Janmichael Frahm, Marc Pollefeys, et al. GPU-based video feature tracking and matching: TR 06 – 012[R]. University of North Carolina, USA, 2006.

[4] Sudipta N Sinha. GPU_KLT: A GPU-based Implementation of the Kanade-Lucas-Tomasi Feature Tracker [EB/OL]. University of North Carolina: 10, August, 2008. http://cs.unc.edu/~ssinha/Research/GPU_KLT/.

[5] Julius Fabian Ohmer, Nicholas J Redding. GPU-accelerated KLT tracking with monte-carlo-based feature reselection[C]//Digital Image Computing: Techniques and Applications, Canberra, 2008, 12(1 – 3): 234 – 241.

[6] Christopher Zach, David Gallup, Jan-Michael Frahm. Fast gain-adaptive KLT tracking on the GPU[C]//Computer Vision and Pattern Recognition Workshops, Anchorage, 2008, 6(23 – 28): 1 – 7.

[7] Stan Birchfield. KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker [EB/OL]. Clemson University: 30 August 2007. http://www.ces.clemson.edu/~stb/klt/.

[8] S P Ierodiaconou, N Dahnoun, L Q Xu. Implementation and optimisation of a video object segmentation algorithm on an embedded DSP platform [C]//The Institution of Engineering and Technology Conference on Crime and Security, London, 2006, 6(13 – 14): 432 – 437.

[9] Texas Instruments Incorporated. TMS320DM642 Video/Imaging Fixed Point Digital Signal Processor: SPRS200N [R]. July, 2002.

[10] Texas Instruments Incorporated. TMS320C6000 Optimizing Compiler v 6.1: SPRU187O[R]. May, 2008.