基于 DSP 的 KLT 特征点跟踪算法优化实现①

稂龙亚[1], 梁久祯[2], 汪玉成[1], 杨 阳[1]

[1](安徽南瑞继远软件有限公司, 合肥 220088)
[2](江南大学 物联网工程学院, 无锡 214122)

摘 要: 针对 C6000 系列定点 DSP 下 KLT 特征点跟踪算法的快速实现问题, 通过结合 DSP 体系结构特点以及软件流水技术, 提出了一种该类型 DSP 下的算法优化实现方案. 基于算法流程的模块化分析, 完成了该算法的定点化设计与实现工作; 针对算法中计算密集型模块, 提出了具体的改善软件流水, 减少存储器访问, 任务级并行设计等优化实现方法. 在 DSP 软仿真平台 CCS 下进行实验测试, 该优化实现方案下的运行效率和资源利用率均有较大的提升, 且所使用的算法优化方法适用于所有 C6000 系列 DSP.

关键词: 定点 DSP; 算法优化实现; 特征点跟踪; KLT 跟踪算法

Optimized Implementation of KLT Feature Points Tracking Algorithm Based on DSP

LANG Long-Ya[1], LIANG Jiu-Zhen[2], WANG Yu-Cheng[1], YANG Yang[1]

[1](Anhui Nari Jiyuan Software Co.,Ltd. Hefei 220088, China)
[2](School of Internet of Things Engineering, Jiangnan University, Wuxi 214122, China)

Abstract: The paper proposes an optimized implementation of the algorithm based on the platform of C6000 DSPs, in consideration of the DSP hardware resource characteristics and software pipelining technique. Based on the analysis of algorithm process modular. We completed the designation and implementation of Floating-to-Fixed-Point conversion; for Compute-Intensive modules, we present the concrete optimized method, including improving the performance of Software pipelining, reducing memory access, achieving task level parallelism. The experiment made in Code Composer Studio results show that, the efficiency and resource utilization of the optimized implementation scheme are increasing, and the optimization method in this paper also apply to all C6000 DSPs.

Key words: fixed-point DSP; optimized implementation of the algorithm; the feature point tracking; the KLT-based tracking algorithm

1 引言

一般而言算法的研究、优化、实现以及到最终产品, 整个过程往往漫长而复杂, 尤其在嵌入式平台的开发中, 算法的仿真与验证平台(Host)往往与实际算法程序的运行平台(Target)硬件资源各不相同, 处理器体系结构、指令集相差甚大, 需要针对硬件结构特点进行算法改进与优化, 而基于某一类型处理器的算法优化与实现, 不但需要对处理器体系结构有足够的了解, 而且需要熟悉待优化算法的原理, 并最终将两者结合完成代码实现.

随着数字多媒体业务的不断发展, 大量的廉价而高性能的 32 位定点 DSP 芯被广泛应用于数字图像处理、信号探测、医疗器械等工业领域. 数字信号处理器(DSP)通过专门的体系结构和指令集设计, 拥有较高的处理性能和较低的资源消耗, 特别适合于通信、视频编码、视频跟踪等复杂计算密集型算法的执行. 其中美国 TI 公司的 C6000 系列 DSP 一直是数字信号处理领域中的主流, 尤其是定点 DSP 因性能高、功耗低、成本小, 是 DSP 处理器市场中的首选芯片, 拥有非常高的 DSP 市场占有率. 针对 C6000 系列 DSP 的算法移植与优化实现工作得到了不少工程人员的关注, Hunter I[1]介绍了嵌入式 C6000 系列 DSP 下的系统设

① 收稿时间:2015-03-15;收到修改稿时间:2015-05-23

计与系统优化问题, 杨光宇[2]重点讨论 C6000 系列 DSP 的 C/C++程序优化技术, Chang C Y [3]研究了在 C6000 系列定点 DSP 下实现一个高效的有源噪声控制系统, 任雁鹏[4]在定点 DSP 上对 MP3 算法中复杂度较高多个模块进行了算法优化研究, 张珍[5]研究了在定点 DM642 处理器下连续隐马尔科夫模型 CHMM 识别算法的优化实现问题.

KLT 特征点跟踪算法[6]已被广泛的研究与使用, 例如在智能交通[7]、视频监测[8], 图像拼接[9]等场景中. 在该算法工程应用中, Birchfield S[10]完成了 KLT 特征点跟踪器的设计以及 PC 平台下的 C 代码实现, 达到了特征点的亚像素级跟踪. Sinha SN[11]探讨了 GPU 上该特征点跟踪算法的加速实现问题, 并研究了该实现在视频分析系统中的应用. Chai ZL[12]针对 KLT 算法在嵌入式平台上的实现进行了研究, 在 Xilinx Virtex-5 FPGA 上完成了该算法的硬件架构设计与实现. 在该算法的 DSP 平台实现研究中, 刘军[13]在 C6000 系列的定点 DM642 平台上完成了 KLT 算法的移植和项目级优化工作, 但其并没有利用该 DSP 的硬件特性和 KLT 跟踪算法的特点进行算法级和汇编级优化, 尤其在移植实现后, 算法中仍然存在着大量浮点数据和浮点运算, 无法充分发挥定点 DSP 的效能. 本文在其工作基础上, 结合定点 DSP 体系结构与 KLT 特征点跟踪算法的特点, 进一步完成了算法定点化、汇编级优化、任务级并行设计等优化实现工作. 该优化实现方案以文献[10]算法实现为参考, 首先对算法进行了定点化实现; 其次根据 DSP 的硬件资源特性以及软件流水技术[14], 提出了定点 DSP 下该算法的优化实现方法.

2 KLT特征点跟踪算法

KLT 特征点跟踪算法[15]是一种以 SSD(Sum of Squared intensit Differences)为度量的最优估计的匹配方法.

设 t 时刻对应的图像帧表示 $I(x,y)$, $t+\varepsilon$ 时刻对应的图像帧表示 $J(x,y)$. 假设在一定条件下满足灰度守恒定律, 设两幅跟踪图像 $I(x,y)$和 $J(x,y)$得式(1), 其中特征点 $X(x,y)$的位移为 d=(dx,dy), 噪声 $n(X)$, 在灰度守恒的假设下, 所求位移 d 使得 $n(X)$最小, 即残差值最小, 由式(2)使得 $\frac{\partial \varepsilon}{\partial d}=0$, 从而求取位移 d.

$$J(X+d)=I(X)+n(X) \tag{1}$$

$$\varepsilon=\int_W [J(X+d)-I(X)]^2\omega(X)\mathrm{d}X \tag{2}$$

这里 W 是给定的特征窗口, $\omega(X)$ 是窗口加权函数, 一般为 1, 也可设为高斯函数以强调特征窗口的中心区域. 为了使 ε 最小化, 将 $J(X+d)$泰勒展开, 保留前两项, 令 g 是泰勒展开的一阶泰勒系数, 并对变换后的 ε 两边对 d 求导, 当导数为 0 时, ε 取得最小值, 即 SSD 最小, 如式(3):

$$\frac{\partial \varepsilon}{\partial d}=\int_W [J(X)-I(X)+g^{\mathrm{T}}d]g\omega(X)\mathrm{d}X=0 \tag{3}$$

对于连续两帧灰度图, 使用式(3)求解得到特征点的位移 d, 由于该式在泰勒展开阶段中引入了误差, 通过 Newton-Raphson 迭代求 d 以消除误差, 计算式(4).

$$d_{k+1}=d_k+\begin{bmatrix} g_x^2 & g_xg_y \\ g_xg_y & g_y^2 \end{bmatrix}^{-1} \times\left[\int_W [J(X)-I(X)]\begin{bmatrix} g_x \\ g_y \end{bmatrix}\omega(X)\mathrm{d}X\right] \tag{4}$$

一般而言, 为了提高匹配的精度和速度, 迭代计算的初始值 d_0 设为 0, d_k 表示第 k 次牛顿迭代计算得到的位移, 令 *iteration* 为迭代次数, 则迭代匹配的终止条件为 $|dx_{k+1}-dx_k|$ 或 $|dy_{k+1}-dy_k|$ 小于阈值 0.1, 当迭代次数超过经验值 10 时, 亦结束跟踪. 该匹配跟踪算法是一种典型的基于最优估计的特征点跟踪算法, 它无需进行全局搜索, 耗时量可控, 因此该方法自提出以来获得了广泛的关注与应用.

3 KLT特征点跟踪算法定点化实现

在定点类型 DSP 中, 如果算法使用了浮点运算, 那么这些浮点运算最终还是要通过定点指令来模拟, 用定点指令模拟浮点运算往往较为费时, 因此需要将算法中的浮点运算尽可能地转换为定点, 在代码中显式地使用定点运算. 在定点化的前提下可以很方便地探讨该算法中哪些部分适合于 DSP 平台, 尤其是对于计算密集型部分进行汇编化指令调整, 可以直接优化软件流水代码, 这也是 DSP 最低层的优化, 同时也使得硬件资源的配置和使用更为简单.

由于特征点跟踪精度的需要, 整个 KLT 算法的数据流均使用浮点数据进行操作和保存, 其实现版本参考文献[10], 该算法实现中存在大量的浮点数据和处理, 其中包括平滑时的高斯核、平滑后的图像数据、

金字塔数据、*gradx*、*grady*、*ex*、*ey* 以及所求跟踪结果位移矢量 *d*. 针对定点 DSP, 来完成如此大量的浮点运算, 显然这是个很有难度的问题.

基于 C6000 系列定点 DSP 上的 KLT 特征点跟踪算法优化实现工作, 首先需要进行定点化实现, 该算法的定点化设计与实现借鉴了文献[12], 算法数据流图及相关标定信息见图 1. 该图中虚线以上为算法中包含一维卷积运算的图像平滑与梯度计算处理, 虚线以下为算法中位移矢量公式求解部分. 在该图中, 带下滑线标注的为数据信息, 包含变量以及二维数据, 以"_"开头为处理函数, 以 Q 开头的信息表示对应的数据标定信息, 该标定信息均位于数据标识右侧.

4 基于DSP的算法优化实现

为了提高 KLT 特征点跟踪算法在定点 DSP 上的执行效率和资源利用率, 在定点化的基础上本文针对算法中计算密集型模块做进一步软件流水优化和算法优化调整, 结合 C6000 系列定点 DSP 的硬件特点, 主要围绕算法中的图像平滑和梯度计算、位移公式求解三部分的优化实现. 参考 C64X+ DSP 指令集[16], 将整个 KLT 跟踪算法的计算密集型模块首先进行汇编化, 对应的优化工作则可根据定点 DSP 资源特点, 直接增加、修改、调整汇编代码, 充分利用各种 DSP 的硬件资源, 这些是 C 代码所不具备的.

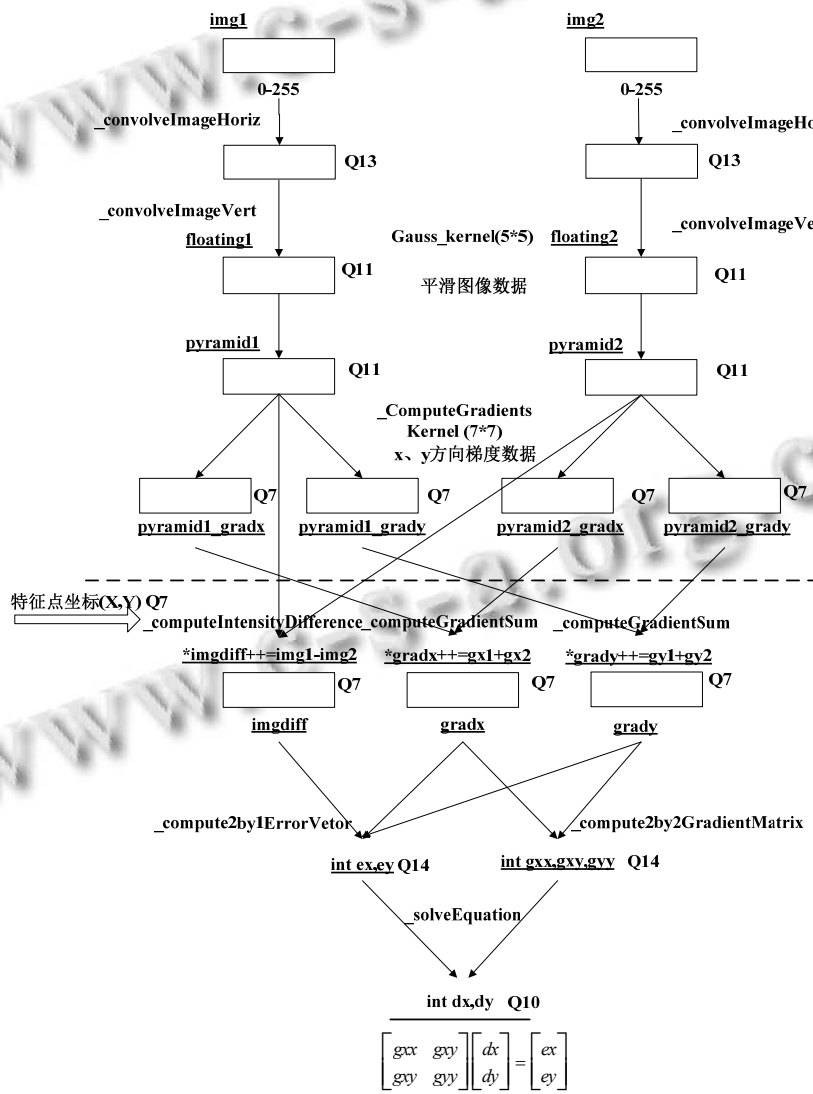

图 1 特征点跟踪算法数据流图及标定信息

优化工作主要考虑 DSP 的特性: 其一 C6000 系列 DSP 片内有 2 个数据通道, 8 个功能单元和 2 个通用寄存器组, 每组对应一个数据通路, 具有较强的乘法和移位能力. 其二 C6000 系列 DSP 采用超长指令字(VLIW), 即在每个时钟周期最高可以提供 8 条 32 位指令, 总字长为 256 位的指令包可以同时分配到 8 个并行处理单元.

4.1 平滑处理模块与梯度计算模块的优化

4.1.1 构建行处理软件流水

程序中的循环是影响算法实时性的一个主要因素, 尤其是当循环迭代的次数较多时, 会使程序的执行效率降低, 所以对循环代码的优化也是程序优化必不可少的部分. 软件流水就专门用于优化循环代码, 其有效的增加了单位周期内的并行的指令数. 在 Birchfield S 浮点实现的垂直和水平卷积代码实现中, 若将该部分直接移植到定点 DSP 平台下, 只能打开单点处理的软件流水, 因为该实现最小循环是单点卷积处理, 即软件流水核开启到结束, 仅完成一点的单点一维卷积处理, 甚至不包括数据读取的指针偏移和结果存储, 该部分代码并不能形成行处理软件流水. 结合 KLT 特征点跟踪算法中平滑处理和梯度计算中的高斯核长度有限, 可以将一维卷积处理的循环展开, 形成行处理软件流水, 即软件流水开启到结束, 完成一整行像素点的一维卷积处理.

4.1.2 存储器访问优化

频繁的存储器访问, 会阻塞 DSP 的执行, 对比于 Birchfield S 浮点实现, 本文算法优化实现: 首先改变了图像数据读取方式, 优化理由: 算法中一维卷积处理具有连续性, DSP 中寄存器资源丰富, 且 MV 指令相比 LDW 指令, 该指令的执行单元更多, 执行期更短. 即一维卷积的数据读取时, 处理完前一个点后, 后一点的处理数据有很大一部分是与前一点的数据是相同, 结合 DSP 寄存器资源丰富的特点, 可以保存该数据于寄存器中, 并搬移前面的已加载的数据而无需重新加载. 在该 DSP 平台中, 对该部分公共数据直接使用 MV 指令代替 LDW 指令, 一维卷积每一个点处理时数据读取次数降为 1 次, 而在原 Birchfield.S 实现中每个像素点的一维卷积, 会读取 L(高斯核长度)次, 该优化使得每个点时减少 L-1 次数据读取. 以高斯核长度 L=5 为例, 行数据存储起始地址为 n, 优化前后每行前 3 个点一维卷积处理的数据访问如图 2 所示, 虚线矩为优化掉的数据读取. 其次考虑到 DSP 寄存器丰富的特点, 在 KLT 匹配跟踪算法中卷积处理期间高斯核亦可以永久存储于寄存器中, 每次一维卷积卷积只需加载一次高斯核于寄存器中, 直至处理结束, 也提高了 DSP 寄存器资源利用率.

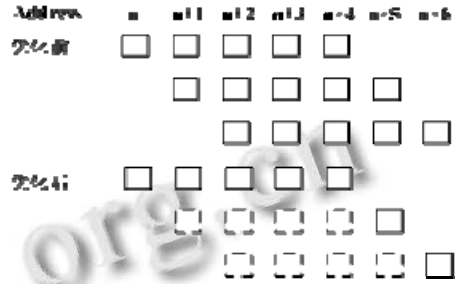

图 2　优化前后像素点数据访问

4.1.3 任务级并行设计

通过以上优化, 该算法模块在 C6000 系列 DSP 平台下仍然是可以继续优化, 硬件乘法器是 DSP 中一种重要的硬件资源, 优化后的软件流水均是完成单点一维卷积处理, 因为高斯核长度为奇数, 故单点处理中的乘法操作次数是奇数, 而我们的 DSP 的乘法器个数是偶数(双数据通路中各含一个), 在单点处理模式下, 即使达到最高的优化效率, 乘法器还是存在一定浪费, 而且对于单点处理而言, 交叉通道的资源限制也是需要考虑的, 因为一旦图像数据与核数据分开存于两个数据通路的寄存器中时, 就会受到交叉通路资源的限制, 但是两数据如果均存放在同一个数据通路中, 又会对另一个数据通路中的资源造成浪费.

结合 C6000 系列 DSP 存在双数据通路的特点, 可否同时处理两点, 其优势在于: 乘法器利用率会得到提高, 在软件流水的编排时会降低交叉通路的资源限制所造成的影响. 同时处理两点应该如何设计, 是进行单数据行下前后两点同时处理, 还是双数据行并行模式上下行同时单点处理这也是值得考虑的问题. 经实验测试发现双行下单点处理更好, 因为单行下前后两点处理时, 很多数据是共同需要的, 该方式下容易出现存储器访问冲突.

结合对 KLT 跟踪算法中卷积模块的分析, 由高斯函数的可分离性原理知, 其前后两行处理是相互独立、互不影响的, 即在优化实现方案中构建相互独立的任务级双数据行并行一维卷积处理具有可行性. 在该实现下一旦开启软件流水, 上下两行数据的一维卷

积将同时进行处理, DSP 的双数据通路将得到充分利用, 双数据行任务级并行处理的示意图如图 3 所示, 在该图中上下两行数据分别独立地使用DSP核中的两个数据通路进行任务级并行一维卷积处理.

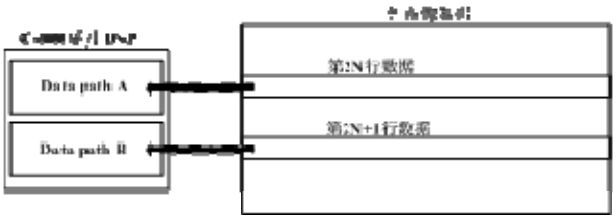

图 3 任务级双数据行一维卷积并行处理

经过以上优化方法, 在 DSP 开发平台 CCS 下, KLT 特征点跟踪算法的平滑处理模块与梯度计算模块部分所得的一维卷积软件流水核编译信息如图 4 所示.

```
                           A-side   B-side
.L units                     1        0
.S units                     0        0
.D units                     2        2
.M units                     7*       7*
.X cross paths               0        0
.T address paths             3        3
Long read paths              0        0
Long write paths             0        0
Logical  ops (.LS)           0        0     (.L or .S unit)
Addition ops (.LSD)         14       12     (.L or .S or .D unit)
Bound(.L .S .LS)             1        0
Bound(.L .S .D .LS .LSD)     6        5

Searching for software pipeline schedule at ...
   ii = 7  Schedule found with 3 iterations in parallel
```

图 4 优化后一维卷积软件流水核编译信息

4.2 特征点位移矢量公式求解模块的优化

该模块主要由 5 部分组成, 首先由于在特征窗口差分计算以及特征窗口梯度和计算的两个函数实现中, 其循环内部需要反复调用插值函数, 在 DSP 平台下, 对于循环内部存在函数调用的情况是无法构建软件流水的, 不断的完成函数跳转等控制指令并不是 DSP 的优势所在. 但是在 gxx, gxy, gyy, ex, ey 变量求解函数 _compute2by2GradientMatrix 和 _compute2by1 Error Vector 中, 这两个处理函数是可以充分利用 DSP 平台优势的, 即构建高性能的软件流水核.

```
                           A-side   B-side
.L units                     0        1
.S units                     0        0
.D units                     2*       1
.M units                     1        1
.X cross paths               0        1
.T address paths             2*       1
Long read paths              0        0
Long write paths             0        0
Logical  ops (.LS)           0        0     (.L or .S unit)
Addition ops (.LSD)          4        3     (.L or .S or .D unit)
Bound(.L .S .LS)             0        1
Bound(.L .S .D .LS .LSD)     2*       2*

Searching for software pipeline schedule at ...
   ii = 2  Schedule found with 8 iterations in parallel
```

图 5 _compute 2by2GradientMatrix 函数软件流水信息

求解 gxx, gxy, gyy 变量的_compute2by2Gradient Matrix 函数内部是可以通过合理指令编排构建 ii 为 2 的软件流水, 优化后该函数软件流水编译信息见图 5. 通过对软件流水循环核代码分析, 发现其循环内部的数据操作存在一定关联性, 无法同时并行更多的指令, 同样的情况发生在_compute2by1ErrorVector 函数中. 优化实现时试图将_compute 2by2GradientMatrix 和 _compute2by1ErrorVector 归并在一起计算, 发现仍然存在较大关联性, 优化效果较差. 求解位移量模块中的 solveEquation 函数是一个顺序执行的过程, 即由已知变量逐步完成求解的过程, 其实现借助 DSP 内联函数 divlli 完成.

5 优化结果及分析

为了对该KLT跟踪算法优化实现方案的效果进行测试, 实验在 CCS3.2 开发环境下配置 C64XX DSP Simulator完成, 该DSP软仿真平台由TI公司提供, 完全模拟了真实硬件平台, 通过使用 CCS3.2 下的 Profile 工具对代码运行效率进行评估, 也可以直观地获得代码实现的相关数据, 大大节省了频繁下载与调试的时间. 一般而言, DSP 算法工程师都是基于该软仿真平台下完成 DSP 算法优化与实现工作. 本文实验主要从两个角度对优化实现方案进行了测试工作:

(1) 该优化实现的执行效率问题:

主要对不同优化策略前后的优化效果情况进行评估, 并对软件流水性能进行分析. 即首先单纯统计优化前后完成所需的时钟数、寄存器、乘法器资源, 其次以迭代间隙为软件流水核指标, 对于各模块优化后形成的软件流水核的性能进行分析.

(2) 该优化实现所求位移矢量的精度问题:

特征点跟踪算法的目标就是获得该特征点在下一帧中的新位置坐标, 这个新坐标是由上一帧已知的跟踪坐标和求解所得的位移矢量之和获得, KLT 跟踪算法本身就是一个迭代求取相对位移失量的算法, 本文主要围绕特征点相对位移失量来进行研究.

5.1 算法优化实现结果分析

在CCS 3.3开发环境C64XX DSP软仿真测试平台上, 探讨在 KLT 特征点跟踪算法中耗时的一维垂直卷积运算部分的优化策略及对应的时钟消耗和资源利用. 实验以 256×192 图片, 特征点大小 7x7, 高斯核长度为 7 为例, 在合理配置 CACHE 下, 首先统计优化前后图

像垂直卷积中单行完成和全部完成所需的时钟数, 不考虑存储器阻塞所引起的 CPU 挂起. 其次对于优化后形成的行处理软件流水, 统计软件流水核的性能: 迭代间隙, 单点理论时钟数, 核内资源利用率, 数据见表 1.

Address Range	Symbol Name	Symbol Type	Access Count	cycle.CPU:...
0x1b9e8-0x1ba20	_compute2by1ErrorVector	loop	49	1127
0x1ba68-0x1baa4	_compute2by2GradientMatrix	loop	49	1225

(a) 优化前 C 代码实现所消耗的时钟资源

Address Range	Symbol Name	SLR	Symbol Type	cycle.CPU:...
0x1ba6a-0x1bad4	process_2by1ErrorVector	2-21:2by1Erro...	loop	112
0x1bb0c-0x1bb74	process_2by2Matrix	2-25:2by2Grad...	loop	110

(b) 软件流水优化处理后所消耗的时钟数

图 6　位移量求解中两函数的优化前后的时钟消耗

表 1　优化前后运行效率及软件流水核数据

策略	迭代间隙	单点理论 Clycle	利用率		时钟资源	
			乘法器	寄存器	单行垂直卷积	垂直卷积总计
定点化	-	-	-	-	4480	1151741
行处理流水	9	9	38.9%	38.5%	1690	435715
减少数据访问	4	4	87.5%	36.3%	760	196617
任务级优化	7	3.5	100%	59.15%	658	169480

通过上表数据得:首先经过内循环展开, 构建行处理软件流水, 大大加快单行一维卷积处理, 两个点的循环迭代相差 9 个时钟, 其次通过算法存储器访问优化, 减少数据读取时间, 也降低了软件流水核内的寄存器需求量, 迭代间隙 ii 为 4, 因核长度为 7, 且 DSP 核中存在 2 个并行的乘法器, 流水迭代间隙 ii 理论值可达 3.5, 最后通过任务级优化, 软件流水核内乘法器利用率达到了 100%, 图 4 的编译信息也验证了该点, 即在 ii=7 个时钟内两个通路的乘法器(.M)均使用了 7 次, 一维水平卷积亦达相同性能的流水.

通过以上优化策略, 大大减少了特征点跟踪中的平滑处理和梯度计算所耗时钟数, 如表 2 所示. 通过表 2 发现运算效率均提高了 7 倍以上.

表 2　优化前后函数执行时间

	平滑函数(cycle)	梯度函数(cycle)
优化前	2358451	4815123
优化后	271023	681994

本文还对特征点位移矢量求解模块进行软件流水优化处理, 在相同算法参数下, 以 7×7 窗口大小为例统计了两个变量求解函数优化前后所消耗的时钟数, 如图 6 所示, 结合图 5 中的软件流水编译信息知, 每点处理的循环迭代需要 ii=2 个时钟, 我们的特征点窗口大小为 7×7, 需要处理 49 个像素点, 再加入软件流水填充期和软件流水排空期, 基本符合图 6 所测数据.

需要说明的是, 本文实验虽然选用 256×192 大小的小尺度图片, 但是: (1)以上优化方法均不受图片大小限制; (2)图片尺度越大, DSP 软件流水核循环核期越长, 得到每点的实际平均时间就越小, 这一点是由 DSP 软件流水技术的特点决定的.

5.2 算法定点化结果分析

KLT 跟踪算法通过求取相对位移失量来完成特征点匹配跟踪, 该变量的计算结果直接决定了特征点匹配跟踪结果, 故本文主要对该变量进行测试试验.

实验平台 A: 硬件平台: PC 机 (intel Pentium E5200), 软件平台: 在 Viusal Studio 2008 平台下测试 Birchfield S 浮点实现的 KLT 匹配跟踪算法所获得的首位移量(*dx*, *dy*), 所获数据为下表中工程组 A.

实验平台 B: 软仿真平台: 在 CCS3.3 中 C64XX DSP Simulator 下, 测试本章优化实现后获取的首位移失量(*dx*,*dy*), 因存在数据标定, 为了对比的直观性, 该数据进行反标定化, 对应于工程组 B.

实验分别测试了两组不同的跟踪实验数据, 并对相同的 10 个特征点进行实验测试, 数据见表 3.

本章对 KLT 匹配跟踪算法进行了定点化, 并重点针对平滑处理、梯度计算以及位移公式求解三部分进行了汇编级算法优化, 取得了较好的优化效果, 同时也存在部分精度损失. 表 3 中位移失量精度数据基本能达到小数点后两位, 精度丢失主要是由定点化过程中造成的, 汇编优化各阶段本身不存在精度丢失.

在定点化过程中丢失的主要原因有: (1) 定点化

中的数据标定设计存在问题, 但因 DSP 具有较强的移位指令和寄存器资源, 可以根据不同的精度需要, 重新进行数据标定, (2) 32 位定点 DSP 硬件资源以及指令操作数的限制, 但是通过 32 位定点 DSP 指令是可以模拟 64 位或更高位运算的, 例如在定点化实现及后续优化过程中, 部分优化实现涉及到有符号 32 位乘法, 有符号 64 位减法, 有符号的 64 位偏移, 有符号 64 位除法, 而 TI 公司的 32 位 DSP 处理器指令集并没有完全提供这些指令, 需要自己根据 C6000 系列 DSP 指令系统中已有的 16 位指令和 32 位指令来模拟完成这些操作, 本文使用的方法是借助 CCS 下的混合模式, 对 C 代码实现进行模式转化, 并对转化后的汇编代码实现进行研究, 再结合数学原理分析或者咨询 TI E2E 社区工程师, 这一工作是可以完成的.

表 3 在 A、B 两平台下两组不同实验数据所得的位移失量数据

特征点坐标	跟踪起始坐标	工程组	数据组 1(*dx*,*dy*)	数据组 2(*dx*,*dy*)
(60,60)	(60,60)	A	(-0.506594,0.002364)	(-0.134948,-0.006113)
		B	(-0.505859,0.001953)	(-0.134789,-0.006483)
(70,70)	(70,70)	A	(-0.359902,-0.063475)	(-0.456603,-0.314490)
		B	(-0.359375,-0.063477)	(-0.45610,-0.314453)
(80,80)	(80,80)	A	(-0.385974,0.015881)	(-0.135700,-0.357920)
		B	(-0.385742,0.015625)	(0.135742,-0.357420)
(90,90)	(90,90)	A	(-0.031250,0.002930)	(-0.085675,-1.084550)
		B	(-0.031992,0.003342)	(-0.084901,-1.085938)
(90,115.2)	(91.9,114.62)	A	(3.293830,-1.286714)	(3.628024,-0.823815)
		B	(3.291992,-1.282227)	(3.626953,-0.823242)
(42.0,57.3)	(40.9,57.6)	A	(-1.180384,0.526264)	(-0.853103,0.413958)
		B	(-1.181641,0.520508)	(-0.853516,0.411133)
(9.1,80.3)	(10.1,80.3)	A	(0.617058,-0.075470)	(-0.156250,0.189453)
		B	(0.617188,-0.075195)	(-0.157227,0.190430)
(90.1,22)	(90.3,21)	A	(0.122070,-2.035203)	(-0.286218,1.140273)
		B	(0.128980,-2.035035)	(-0.287109,1.137695)
(64.57,23.56)	(63.57,24.67)	A	(-1.837278,2.356242)	(0.876121,-0.367636)
		B	(-1.837891,2.356445)	(0.874023,-0.367188)
(75,84)	(74.85)	A	(-1.689290,1.474618)	(0.011050,-0.616239)
		B	(-1.689453,1.474609)	(0.011719,-0.616211)

6 结语

本文通过对 KLT 特征点跟踪算法进行模块化分析, 并结合 C6000 系列定点 DSP 的硬件特点, 完成了算法定点化、汇编级优化、任务级并行设计等优化实现工作. 在 CCS 仿真平台下, 首先验证了优化实现方案运行效率和资源利用率的提升: 算法的图像平滑与梯度计算模块的一维卷积软件流水核中乘法器利用率达到 100%, 寄存器利用率也有较大提升; 在相对位移失量迭代计算模块中, Z 矩阵和残差 e 矩阵变量求解的软件流水迭代间隙为 2, 即该特征窗口中每点处理仅需两个时钟资源, 其次对所求得的相对位移失量数据进行了分析, 解释了少部分精度丢失的原因以及对应的解决方案. 本文算法优化实现方法对 C6000 系列定点 DSP 的算法优化以及特征点跟踪算法的实际应用, 具有较好的指导意义.

参考文献

1 Hunter I. Overview of embedded DSP design. Robert S, Stephan W, eds. Proc. of the 17th European Signal Processing Conference. 2009: 475–479.

2 杨光宇,高晓蓉,王黎,等.基于 TI C6000 系列 DSP 的 C/C++ 程序优化技术.现代电子技术,2009, 32(8):56–59.

3 Chang CY. Efficient active noise controller using a fixed-point DSP. Signal Processing, 2009,89(5):843–850.

4 任雁鹏,梁利平.MP3 音频解码速度优化.微电子学与计算机,2010(4):43–45.

5 张珍.智能机器人语音识别技术.现代电子技术,2011, 34 (12):57–60.

6 Tomasi C, Kanade T. Detection and tracking of point features . School of Computer Science, Carnegie Mellon Univ., 1991.

7 Patel CI, Patel R. Counting Cars in Traffic Using Cascade Haar with KLP. International Journal of Computer & Electrical Engineering, 2013, 5(4).

8 Fradi H, Eiselein V, Keller I, et al. Crowd context-dependent privacy protection filters. 2013 18th International Conference on Digital Signal Processing (DSP). IEEE, 2013: 1–6.

9 Xu T, Ming D, Xiao L, et al. Stitching algorithm of sequence image based on modified KLT tracker. 2012 Fifth International Symposium on Computational Intelligence and Design (ISCID). IEEE, 2012, 2: 46–49.

10 Birchfield S.Klt: an implementation of the kanade-lucas-tomasi feature tracker. [2007-10-17]. http://www.ces. clemson. edu/~stb/klt/

11 Sinha SN, Frahm JM, Pollefeys M, et al. Feature tracking and matching in video using programmable graphics hardware. Machine Vision and Applications, 2011, 22(1): 207–217.

12 Chai ZL, Shi JB. Improving klt in embedded systems by processing oversampling video sequence in real-time. In: Athanas PM, Becker J, Cumplido R, eds. 2011 International Conference on Reconfigurable Computing and FPGAs. Piscataway: IEEE Computer Society, 2011. 297–302.

13 刘军,梁久祯,柴志雷.基于DM642的KLT跟踪算法的实现及优化.激光与红外,2011,41(8): 936–940.

14 Instruments T. TMS320C6000 Optimizing Compiler User's Guide, March 2010. Literature Number: SPRU198J.

15 Lucas BD, Kanade T. An iterative image registration technique with an application to stereo vision. In: Hayes P J, eds. Proc. of the 7th International Joint Conference on Artificial Intelligence. New York: William Kaufmann, 1981. 674–679.

16 Instruments T. TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide, July 2010. Literature Number: SPRU732J.