doi:10.3969/j.issn.1671-637X.2015.10.004

多核DSP图像去雾优化方法

白林亭[1,2,3]， 武永卫[1]， 谢建春[2,3]， 文鹏程[2,3]， 刘小剑[2,3]
(1.清华大学高性能所,北京 100084； 2.中航工业西安航空计算技术研究所,西安 710065；
3.机载弹载计算机航空科技重点实验室,西安 710065)

摘 要：数字信号处理器(DSP)专门针对信号处理而设计,以其独特的特点在信号处理、图形图像处理、通信、交通、航空航天等领域得到越来越广泛的应用。在图形图像处理领域中,图像去雾一直是研究热点,开发时多使用Matlab和C语言仿真,将其直接移植到DSP中运行效率通常较低,难以做到实时处理。针对该问题,提出一种快速有效的图像去雾算法,优化算法过程,获得更好的处理结果;利用TI公司的多核DSP和开发工具,对C代码进行实时优化和并行处理,取得很好的优化效果,能够对低分辨率进行图像去雾的实时处理。

关键词：图像处理；多核数字信号处理器；图像去雾；实时优化；并行处理
中图分类号：TN956；O221.6 **文献标志码**：A **文章编号**：1671-637X(2015)10-0014-05

Optimization Methods for Image Haze Removal Based on Multi-core DSP

BAI Lin-ting[1,2,3], WU Yong-wei[1], XIE Jian-chun[2,3], WEN Peng-cheng[2,3], LIU Xiao-jian[2,3]
(1. Dep. of Computer Science & Technology, Tsinghua University, Beijing 100084, China;
2. Computing Technique Research Institute, AVIC, Xi'an 710065, China;
3. Aviation Key Laboratory of Science and Technology on Airborne and Missileborne Computer, Xi'an 710065, China)

Abstract: The Digital Signal Processor (DSP) is specially designed for signal processing. In recent years, the DSP has found wide application in signal processing, image processing, communication, traffic and aerospace etc. As a practical part of image processing, the image haze removal has been researched for decades. During the research and development of haze removal algorithm, the Matlab and C programming language were usually used, and were supported by DSP. Yet, the efficiency was too low by simply running the source code. To solve this problem, a novel algorithm is proposed for optimizing the algorithm and improving the processing effect. The multi-core DSP and development tool of TI Company are used for real-time optimization of C code and parallel processing. A fine optimization effect is obtained, which shows that the method can be used for processing hazy images with low resolution ratio in realtime.

Key words: image processing; multi-core DSP; haze removal; real-time optimization; parallel processing

0 引言

数字信号处理器(Digital Signal Processor,DSP)是一种具有特殊结构的、适用于数字信号处理的微处理器。DSP的结构和指令专门针对信号处理而设计,十分适合进行信号处理运算。随着近年来嵌入式系统的快速发展,DSP在各个领域得到越来越广泛的应用,如道路交通监控、导航、航拍等。由于大气中悬浮颗粒的存在,场景的光线在传输过程中会被散射;同时,接收设备(如摄像头、相机等)在接收场景光线时,也会接收到被悬浮颗粒散射的部分自然光,最终造成接收到的图像质量下降。最常见的降质图像就是在雾霾等天气获得的带雾图像,因此,对带雾图像的去雾处理有很大的需求。在图像处理技术中,图像去雾技术始终为研究热点和重心,在各个应用领域中有着广泛的应用需求。在图像去雾方法的研究中,最为常用的开发调试工具和语言是Matlab和C语言,对于C语言实现的图像去雾算法,可以利用TI公司的C语言开发工具方便地实

收稿日期:2014-10-27 **修回日期**:2015-01-07
基金项目:航空科学基金(2014ZC31004)
作者简介:白林亭(1990—),男,山东泰安人,硕士生,研究方向为图像处理与并行计算。

现算法在 DSP 平台上的移植。但是，将 C 代码直接移植到 DSP 上运行，效率较低，无法满足要求，因此需要针对 DSP 的架构特点和指令特点，对 C 代码进行优化，同时利用多核 DSP 的并行处理能力进行并行处理。

1　DSP 架构特点

与 Intel 等通用处理器相比，DSP 在结构和指令等方面有其独特的特点[1]，包括架构特性、总线、指令、处理器结构等。

与采用冯·诺依曼结构的通用处理器不同，DSP 采用哈佛结构。冯·诺依曼结构也称为普林斯顿结构，其将程序指令存储器和数据存储器合并在一起，指令和数据共用一个存储空间，程序和数据总线共享，通过分时复用共享总线。DSP 采用的哈佛结构是一种将程序指令存储和数据存储分开的存储器结构，数据存储器和指令存储器相互独立，每个存储器独立编址、独立访问。数据总线和存储总线分开，能够同时获取指令字和操作数，提高了指令的执行速度。

DSP 芯片大多采用多总线结构，数据总线和存储总线分开，可以在一个周期内多次访问数据总线和存储总线。此外，DSP 中还包括其他专用总线，如直接存储器访问（Direct Memory Access，DMA）控制器。DMA 控制器可以独立于 CPU 运行，可以在没有 CPU 参与调度的情况下，进行存储器空间之间的数据传输。

DSP 采用流水线技术提高指令的执行效率。采用流水线技术，在当前指令的执行阶段，可以同时完成下面指令的取指和译码阶段。DSP 芯片利用流水线机制，保证多数乘法、加法和乘加运算可以在单周期内完成，能够大大提高基本数学运算的计算速度。

通用处理器中没有专门的硬件乘法器，在进行乘除法运算时多通过移位等操作完成。DSP 主要用于信号处理，其中包括大量乘除法运算，因此设计了专门的硬件乘法器，能够大大加快乘法运算速度。

2　快速图像去雾方法

图像去雾技术一直是计算机视觉和图像处理领域的研究热点，近年来研究重点大多集中在对图像降质因素的考虑和附加信息或假设。基于多幅图像的方法[2-3]通过获取同一场景的不同图像，得到关于场景的更多信息来进行去雾处理；基于用户交互的方法[4]利用用户输入的附加信息来恢复清晰图像。此外，利用假设或先验的单幅图像去雾方法[5-7]也获得了较多成果。

在上述方法中，不需要附加信息的单幅图像去雾技术适应性更强，有更大的应用价值。本文基于文献[7]中暗通道的假设，提出一种新的快速单幅图像去雾方法。方法流程如图 1 所示。

图 1　快速图像去雾算法

Fig. 1　Fast image haze removal algorithm

在暗通道优化过程中，文献[7]中针对粗略计算的暗通道图像，根据大气散射模型和图像抠图公式的相似性，使用图像抠图的方法对暗通道图像进行优化，起到了很好的优化效果。但是该方法的主要计算部分是对大规模矩阵进行求解，从而导致运算时间特别长，难以应用于实际应用场景。针对该问题，文献[8-10]分别使用导向滤波算法、均值滤波算法和双边滤波算法进行替代处理。3 种方法都是利用带雾图像本身的纹理特性，通过滤波将纹理特性叠加到暗通道图像中，使其更贴近实际场景的纹理和深度特性。考虑到暗通道的优化效果，以及几种优化方法的适用性和并行性，本文使用导向滤波的方法对暗通道图像进行优化。根据文献[7]中算法描述，导向滤波算法计算过程如图 2 所示。

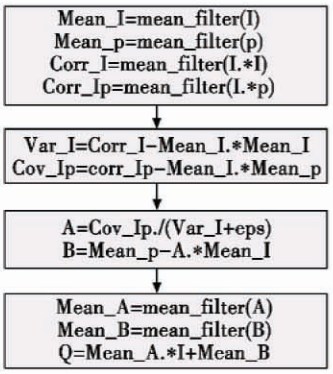

图 2　导向滤波算法

Fig. 2　The guided filter algorithm

图中，mean_filter( ) 函数为快速均值滤波函数，通过累加和方法可以在 $O(M*N)$ 时间内完成，M 和 N 为下采样后图像的高度与宽度。优化前与优化后的暗通道图像如图 3b 与图 3c 所示。

在对暗通道图像进行优化后，要利用上采样恢复源图像的分辨率。最简单的采样方法即插值，常用的二维插值方法有双线性插值、双三次插值、样条插值等。基于多幅图像的上采样方法[11]通过分析多幅图像获取更多的图像信息形成样例辅助进行上采样。利用单幅图像的方法[12]通过卷积和反卷积组成反馈控制迭代进行上采样。考虑到暗通道图像表示雾的浓度，为了保持深度跳变的信息，同时为了加快算法的处理速度，本文算法使用双线性插值方法进行上采样。

最终根据大气散射模型获取输出图像。通过该方法得到的图像亮度普遍偏低,因此有必要利用直方图拉伸等方法进行亮度扩展。输出图像如图3d所示。

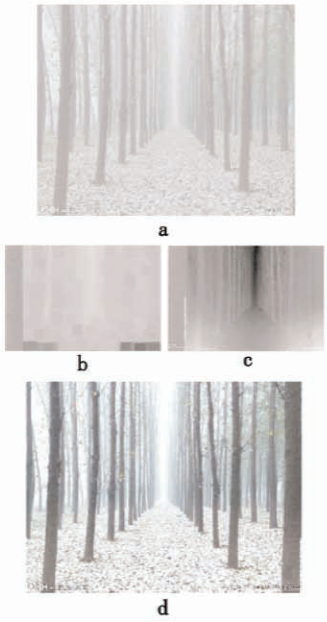

图3 算法过程示例图像

Fig. 3 Images during processing of haze removal

3 多核DSP的算法优化方法

本文使用TI公司的TMS320C6678 DSP开发板进行算法移植和优化。该开发板拥有8颗C66x型号的DSP处理器,单颗处理器能够达到1.25 GHz的频率;挂载2 GB DDR3内存,以及4 MB Shared L2 SRAM。同时,该开发板提供OpenMP支持,可以方便地进行并行优化。通过C语言实现的图像去雾算法可以直接移植到C6678上运行,但直接移植时效率很低,无法满足时间要求,因此要针对DSP的特点进行优化。通过对算法在DSP上移植和运行时的程序剖析,分析造成算法运行性能瓶颈的原因,对算法进行特定优化,具体的优化方法包括算法实现优化[8]、数据结构优化、编译器优化、数学运算和矩阵运算优化、循环优化以及并行优化等[1,13-14]。

3.1 算法优化

在算法实现中,针对多核DSP的并行计算特点,对特定算法过程进行优化处理,具体包括暗通道计算过程的算法优化、暗通道图像优化过程的计算优化。

暗通道图像的计算本质上是对源图像进行最小值滤波计算。在最小值滤波中,理论上对于一个序列的最小值滤波,最坏情况下每个元素都要进行至少1.5次比较[15]。文献[15]提出SMMF(Streaming Maximum-Minimum Filter)算法,对线性数组执行运行时最大最小值滤波时,在最坏情况下平均每个元素需要不超过3次比较。通过对每行和每列使用SMMF算法,实现对矩阵的最小值滤波,如图4所示。

图4 暗通道计算

Fig. 4 The dark channel algorithm

针对输入矩阵中的像素点(x,y),其邻域$w*w$内的最小值滤波保存在输出矩阵的$(y-w,x-w)$中。使用该算法,使得暗通道图像计算速度加快,且和最小值滤波的卷积模板大小无关,这在使用较大的卷积模板时优化效果尤为明显。

在暗通道图像优化过程中,如图2所示,绝大部分的计算为均值滤波。使用box filter[16]的方法可以大大加快均值滤波的过程,但是在导向滤波的计算中,每次均值滤波都要对图像中的每一个像素执行一次除法,而除法运算会消耗大量时间。为此,在导向滤波开始,先对均值滤波算法中的除法核进行倒数运算,可以大大减少除法运算,加快算法执行速度。

3.2 数据结构优化

C66x型号的DSP芯片针对16位数据的乘加运算做了指令集优化,因此,为加快指令的执行速度,在保存输入图像和进行暗通道计算时,使用short或者unsigned short定义变量或数组进行计算,最大化利用芯片的计算能力。对循环计数器,使用int或者unsigned int类型。

3.3 编译器优化

TI公司的开发工具CCS(Code Composer Studio)提供了C/C++编译器,可以将C语言编译、优化、连接并生成可执行文件。通过添加编译选项,可以使用编译器的优化功能,在很大程度上优化源代码。在图像去雾算法中,由于包含大量的矩阵运算,所以在算法实现中会出现大量循环,通过添加$-On$编译选项($n=2,3$)开启软件流水和循环优化,对循环进行流水操作,加快循环过程。使用软件流水后一般会增加代码尺寸,使用$-ms$选项来控制代码尺寸。同时,在调试阶段、程序剖析阶段和最终执行阶段,使用不同的编译选项,提高调试效率和剖析能力。

3.4 循环优化

矩阵运算中,循环多为对两个维度分别遍历形成二重循环或多重循环,循环内部对像素点进行操作,而软件流水只能对最外层循环进行优化。因此,在进行像素点遍历时,将二重循环转变为单重循环,可以更好地进行软件流水。同时,使用$-pm$选项使编译器不产生冗余循环,优化代码尺寸。

3.5 数学运算优化

在图像去雾中,会用到乘法和除法操作,尤其是浮

点除法操作。在直接移植 C 代码时，编译器默认使用移位操作实现乘除法，消耗的周期数较长。TI 公司提供了底层库函数 dsplib，mathlib 以及 imglib，其中包含大量 intrinsics 函数，实现定点及浮点数的加减法、乘除法操作，数据类型转换操作，以及向量和矩阵操作。除了使用底层库函数进行优化外，针对 DSP 指令的特点，对算法中的乘加等运算也进行一定程度的运算。

3.6　并行优化

在图像处理中会用到大量的循环进行矩阵运算。通过软件流水的方式可以大大减少循环的执行时间，而通过该方式优化后的执行时间，依赖于单个处理器核心的处理速度。由于图像处理中的循环多为对像素点的遍历，相邻的两次循环之间的数据依赖较弱，因此可以利用多核处理器将循环拆分，并行执行。

使用多核 DSP 对算法并行优化，有两种优化措施。第一种为对每个循环进行优化，将循环拆分到各个核心中并行执行，降低循环执行时间；第二种为将互相之间没有数据依赖关系的代码段独立出来，分给各个核心，每个核心处理一个代码段，各个核心同时并行处理，降低整个程序的执行时间。TI 公司在 SYS/BIOS 中集成了 OpenMP 并行接口，针对其推出的多核 DSP 芯片，可以方便地进行并行程序开发和优化。

对源图像进行下采样和计算暗通道两个步骤中，主要的计算过程是对图像进行遍历，根据该特点，使用第一种并行方法进行优化。对下采样的循环体，在循环体之前设置并行度，即通过 omp_set_num_threads(n)设置线程数，同时添加编译优化选项#pragma omp parallel for，则编译器会根据设置的线程数量 n，将该循环拆分为 n 个线程并行处理。而针对暗通道部分的计算，在算法优化过程中，使用 SMMF 算法对最小值滤波的过程进行了优化，通过对每行和每列进行最小值滤波，实现邻域的最小值滤波。考虑到该算法在对每行进行 SMMF 算法时，不同行之间没有数据竞争，因此可以将该部分并行处理，每个线程执行一部分行的最小值滤波。同样，在对转置后的每行进行最小值滤波时，继续拆分到不同的线程同时处理。在并行化时，设置每个线程处理连续行，可以最大化利用数据的空间局部性，提升优化效果。for 循环并行优化如以下伪代码所示。

```
omp_set_num_threads(n);
for (i:1 to n)
  do SMMF from row height * (i-1)/n to row height * i/n
```

设置不同的线程数量，时间消耗曲线如图 5 所示。由图中可以看出，随并行度增加，算法消耗时间减少。在核数较少时，线程开销消耗的时间占总运行时间的比例较小；当核数过多时，每个核的运算时间减少，而线程开销占用时间会更明显，因此总时间变化曲线会趋于平缓。

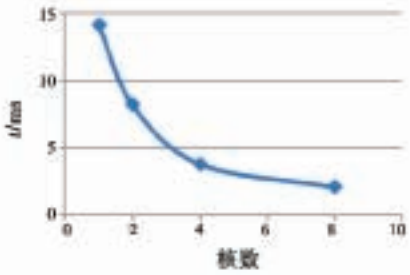

图 5　暗通道计算时间

Fig. 5　Time cost for dark channel algorithm

对暗通道优化部分，主要的计算过程为均值滤波计算。由图 2 可以看出，在对输入图像 p 和导向图像 I 进行均值滤波时，不同的滤波图像数据之间没有相互关系，因此，可以将图 2 中每个步骤内的均值滤波过程分别置于不同线程中同时处理。由于每个均值滤波算法本身是一个相对独立的过程，因此较使用第二种并行优化措施，每个核心处理一段独立的代码段。

具体地，使用#pragma omp parallel sections 告知编译器将之后的代码并行执行，每个线程负责一块独立的代码段。同时，使用#pragma omp section 手动将每个均值滤波过程放置在代码段中。其伪代码如下所示。

```
#pragma omp parallel sections {
    #pragma omp section {
    mean_filter(I)
    }
    #pragma omp section {
    mean_filter(p)
    }
    ···//other sections
}
```

4　优化结果分析

使用本文算法，对 448 × 300 像素点的图像进行去雾处理，并将算法在 TMS320C6678 开发板进行移植。对于算法本身的优化在 PC 机的仿真阶段进行，之后将优化后的程序在 DSP 上进行移植和后续优化。仿真阶段和在 DSP 上进行直接移植的算法执行时间如表 1 所示。

表 1　仿真阶段执行时间

Table 1　Runtime of simulation stages

	Matlab 仿真	PC 机（优化前）	PC 机（优化后）	DSP 移植（优化前）
时间/ms	665	47	31	2179

从表 1 可以看出，对于 448 × 300 像素点的较低分辨率图像，在 PC 机上使用 C 语言实现，不需要刻意进行优化就能够有很好的运行时间效果；如果直接将代码移植到 DSP 上而不加调整和优化，其执行时间无法

忍受。因此,对于绝大多数算法,即使是简单的算法,在 DSP 上进行移植或者实现时,都要充分考虑 DSP 的架构特点,进行适当的优化处理。

在多核 DSP 上,对经过算法优化后的程序依次进行数据结构及循环优化、编译器选项优化、数学运算优化和并行优化。各步骤优化结果如表 2 所示。首先利用编译器优化选项进行编译优化,以及对数据结构和循环进行优化。设置 $-o3$,$-ms$ 和 $-pm$ 选项,同时用 short 存储图像数据,并将部分双重或三重循环改为单重循环。优化后执行时间为 426 ms。之后,对数学运算进行优化,尤其是对除法进行优化。使用内联函数_rcpsp( ) 替代除法算子"/",并且对乘法和加法进行拆分和次序调整。经过该部分数学运算优化后,算法的执行时间缩短为 147 ms。利用一个 DSP 核心进行运算,即使经过了诸多优化步骤,也难以实现实时处理,因此需要利用多核的并行能力进行并行优化。对暗通道计算部分、暗通道优化部分、上采样部分,以及最终输出图像计算部分,使用 OpenMP 在 8 个 DSP 核心上进行并行优化。最终优化后的执行时间为 37 ms。使用多核进行并行优化,可以将算法中最为耗时的暗通道优化过程和无雾图像输出过程进行较大程度的改进,最终使整个算法的执行时间缩短到可以接受的程度。

从以上各个过程的优化结果时间分析可以看出,针对 DSP 的架构特点和指令特点,分析程序中导致执行时间瓶颈的部分,进行特定的优化处理,可以大大缩短程序的执行时间,取得较满意的结果。

表 2 DSP 各步骤优化后执行时间

Table 2 Runtime after optimization of each process on DSP

	DSP 优化前	编译器优化	数学运算优化	并行优化
时间/ms	2179	426	147	37

5 结束语

本文提出一种基于暗通道假设的改进的图像去雾方法,利用导向滤波对透射率图进行优化,取得很好的去雾效果。在多核 DSP 上实现去雾算法,并利用 DSP 本身的架构特点和指令特点,以及多核处理器的并行处理能力,大大缩短了算法的执行时间,使得图像去雾在嵌入式环境中能够有更广泛的应用。针对 DSP 的优化,本文仍有优化不足之处,比如对高速存储空间的充分利用,对存储带宽的利用等。下一步工作会继续探究 DSP 上的优化方法,并将其应用到实际的图像处理中。

参考文献

[1] 郑阿奇. DSP 开发宝典[M]. 北京:电子工业出版社, 2012. (ZHENG A Q. DSP development collection[M]. Beijing: Publishing House of Electronics Industry, 2012.)

[2] 王勇,薛模根,黄勤超. 基于大气背景抑制的偏振去雾算法[J]. 计算机工程, 2009, 35(4): 271-272, 275. (WANG Y, XUE M G, HUANG Q C. Polarization dehazing algorithm based on atmosphere background suppression [J]. Computer Engineering, 2009, 35(4): 271-272, 275.)

[3] NARASIMHAN S G, NAYAR S K. Contrast restoration of weather degraded images[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2003, 25(6): 713-724.

[4] NARASIMHAN S G, NAYAR S K. Interactive deweathering of an image using physical models[C]//IEEE Workshop on Color and Photometric Methods in Computer Vision, France, 2003: 11-18.

[5] TAN R T. Visibility in bad weather from a single image [C]//IEEE Conference on Computer Vision and Pattern Recognition, 2008: 1-8.

[6] FATTAL R. Single image dehazing[C]//ACM Transactions on Graphics (TOG), ACM, 2008. doi: 10.1145/1399504.1360671.

[7] HE K, SUN J, TANG X. Single image haze removal using dark channel prior [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011, 33(12): 2341-2353.

[8] HE K, SUN J, TANG X. Guided image filtering[C]// IEEE Transactions on Pattern Analysis and Machine Intelligence, 2012, 35(6): 1397-1409.

[9] 王燕,伍博,谷金宏. 一种单幅图像去雾方法[J]. 电光与控制, 2011, 18(4): 65-67. (WANG Y, WU B, GU J H. A method for single image defogging[J]. Electronics Optics & Control, 2011, 18(4): 65-67.)

[10] 孙抗,汪澈,周志强,等. 基于双边滤波的实时图像去雾技术研究[J]. 北京理工大学学报, 2011, 31(7): 810-813. (SUN K, WANG B, ZHOU Z Q, et al. Real time image haze removal using bilateral filter[J]. Transactions of Beijing Institute of Technology, 2011, 31(7): 810-813.)

[11] FREEMAN W T, JONES T R, PASZTOR E C. Example-based super-resolution[J]. Computer Graphics and Applications, 2002, 22(2): 56-65.

[12] SHAN Q, LI Z, JIA J, et al. Fast image/video upsampling [J]. ACM Transactions on Graphics(TOG), ACM, 2008. doi: 10.1145/1409060.1409106.

[13] BELL D, WOOD G. Multicore programming guide[R]. Application Report SPRAB27A, Texas Instruments, 2009.

[14] INSTRUMENTS T. TMS320C66x DSP, CPU and instruction set[Z]. SPRUGH7, Nov, 2010.

[15] LEMIRE D. Streaming maximum-minimum filter using no more than three comparisons per element[Z]. ArXiv Preprint cs/0610046, 2006.

[16] CROW F C. Summed-area tables for texture mapping [J]. ACM SIGGRAPH Computer Graphics, 1984, 18(3): 207-212.