多线程技术在 JEPG2000 图像解码中的应用①

范少卓，邓家先，王成成

(海南大学 信息科学与技术学院，海口 570228)

摘 要：多线程技术已经得到广泛地应用，基于多线程的视频解码技术具有很高的研究价值和实用价值，它可以大大提高图像的解码速度。提出了一种在 Windows 环境下多线程编程的实现方法，并将其应用在了 JEPG2000 图像解码中，实现了图像解码并行处理，提高了图像的解码速度。结果显示，应用多线程技术之后，JEPG2000 解码程序在双核，四核处理器上的运行速度相对单核处理器分别提高了 2 倍和 4 倍。多线程技术充分利用多核计算机的资源提高了程序运行的效率。

关键词：多线程；并行；JEPG2000 解码

Application of Multi-Threading in JEPG2000 Video Decoding

FAN Shao-Zhuo, DENG Jia-Xian, WANG Cheng-Cheng

(Information Science & Technology School, Hainan University, Haikou 570228, China)

Abstract: Multithread technology is becoming widely popular. Video decoding based on multi-threading has important research value and use value, it will achieve high improvement in decoding speed. This paper introduces the impletmentation of multithread programming in Mcrosoft Windows. Applying the multithreading technique in JEPG2000 video decoding, the image datas could be processed parallelly, and the decoding speed will be increased. The experiment results show that, system with binuclear, tetranuclear and eight kernels improves the decoding speed by two times, four times and eight times, respectively, compared to system with mononuclear. The Multi-threading technique improves the efficiency of program running by making full use of the resources of multi-core computer.

Keywords: multi-thread; parallel; JEPG2000 decoding

随着科学技术的发展，计算机的处理器速度在急剧增长。然而由于单核 CPU 处理能力受限于制造工艺，不能单纯依靠增加 CPU 处理速度来增加处理器的处理能力。所以人们逐渐把目光转到了多核处理器上来。这样与之相关的多线程技术很快发展了起来。

线程是比进程更小的执行单元，是 CPU 调度与时间分配的对象。多线程借着 Windows 庞大的装机量广泛的进入个人电脑世界，带给个人电脑巨大的冲击。现在，多线程最主要的应用还是在多人，多任务程序设计中，在提高程序效率方面应用的还不是很多[1]。本文利用线程的并行开销小，将多线程技术应用在图像解码当中，可以充分利用了电脑的资源提高程序运行效率。

1 多线程技术

线程(thread),有时被称为轻量级进程(Lightweight Process，LWP)，是程序执行的最小单元。一个标准的线程由线程 ID，当前指令指针(PC)，寄存器集合和堆栈组成。通常在一个进程中可以包含若干个线程，它们可以利用进程所拥有的资源[2]。在引入线程的操作系统中，通常都是把进程作为分配资源的基本单位。由于线程比进程更小，基本上不拥有系统资源，故对它的调度所付出的开销就会小得多，能更高效的提高系统内多个程序间并发执行的程度，从而显著提高系统资源的利用率和吞吐量。因而近年来推出的通用操作系统都引入了线程，以便进一步提高系统的并发性，并把它视为现代操作系统的一个重要指标[3]。

① 收稿时间:2010-07-14;收到修改稿时间:2010-08-22

本设计中使用 VC6.0 提供的 MFC 来实现多线程程序的编写。VC 全称 Microsoft Visual C++是微软提供的一个集成开发环境，MFC 全称 Microsoft Foundation Classes 是微软对基础类库的封装，里边提供了线程操作的基本函数[4]。

2　JEPG2000解码

2.1 JPEG2000 标准

JPEG2000 是由 ISO/IEC JTC1/SC29/WG1 制定的一种最新的图像压缩标准。该标准采用了先进的压缩技术并在可伸缩压缩图像及灵活性方面有许多先进的特征，其系统功能比 JPEG 标准优越，尤其 JPEG2000 采用的是离散小波变换(DWT)替代了 JPEG 中采用的离散余弦变换(DCT)，并采用了最新的编码算法来支持灵活性[5]。JPEG2000 可广泛应用于通信、图像处理、信号处理、信息理论和多媒体等领域中。JPEG2000 与 JPEG 相比优势明显，且向下兼容，取代传统的 JPEG 标准指日可待[6]。

2.2 JPEG2000 编解码过程

在编码器中，首先对源图像进行预处理，对处理的结果进行离散小波变换，得到小波系数。然后对小波系数进行量化和熵编码，最后组成标准的输出码流(位流)[7]。

解码器是编码器的反过程，首先对码流进行解包和熵解码，然后是反向量化和离散小波反变换，对反变换的结果进行后期处理合成，就得到解码后的数据[7]。

3　WindowsMFC多线程技术在JEPG2000解码中的应用

3.1 完成面向对象的 JEPG2000 解码程序

3.1.1 标准 c 语言 JEPG2000 解码程序

本设计中应用的原始 JEPG2000 解码程序是由标准 c 来编写的，整体结构框图如图 1 所示[8]：

首先进行预处理，这个过程需要送入各种参数包括图像的宽高、解码精度、量化门限值和小波级数等。图像宽和高来源于原始图像的信息均取 512，解码精度来源于解码精度要求本实验中取 8，量化门限值来源于 JEPG2000 解码标准，小波系数取 3，表示使用的是 3 级小波变换。并且给输入图像输出图像以及 YUV 图像和各种系数申请内存空间。完成预处理后，码流会送入 T2 解码模块，T2 解码主要完成的是数据包的解码(packet_decode)任务。数据包解码完成后数据就会送到 T1 解码模块完成熵解码。然后送入 Iwt 反向小波变换模块来进行处理，得到 RGB 三基色图像。三基色图像数据由 RGB 转 YUV 模块转化为 YUV 数据，最后经由反向 DC 电平转换达到最终数据，并且释放内存空间。

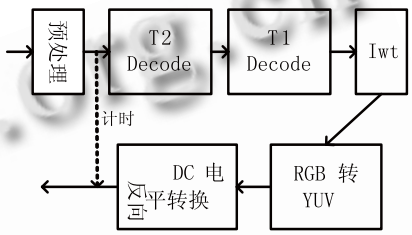

图 1　标准 c 语言 JEPG2000 解码框图

这种面向过程的结构虽然结构清晰但是不利于多线程程序的编写，需要转化成面向对象的形式。这样可以轻松的生成图像解码的对象，而且在一定程度上避免了资源共享，减少了资源冲突的状况，本设计使用 c++的类与继承来实现面向对象的程序设计。

3.1.2 将 JPEG2000 解码程序改写成易于多线程编写的 c++

本设计中 c++程序的总体结构图，包括类的继承结构和各类中包含的主要函数如图 2 所示。

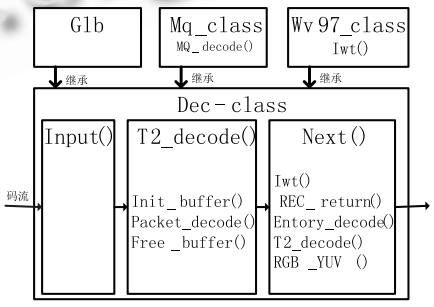

图 2　c++程序总体结构图

为了简化类的继承结构本设计中采用了多继承的结构。

Dec_class 类是最终用来生成解码对象的子类，它继承了 Glb(全局变函数和变量)、Mq_class(Mq 解码)、

Wv97_class(97 小波变换)三个类。Glb 类是全局变函数和变量类，它定义了整个程序里用到的全局变量和解码过程中调用的系数表等数据结构，Mq_class 类是 Mq 解码类，里边的 Mq_decode()函数可以实现 Mq 解码的功能，Wv97_class 类是小波变换类，可以实现反向小波变换的功能。Dec_class 子类作为这 3 个类的子类继承它们的功能。

Dec_class 子类里主要包括 Input()、T2_decode()、Next()三个部分，分别对应标准 c 程序里的预处理，T2Decode 和 T1Decode 及其后续部分。组成了一个完整的解码过程。这样生成一个 Dec_clas 子类的对象后，只要顺序调用这些函数就能完成 JEPG2000 图像解码的工作。

在后面大家可以看到每一路线程就是通过生成一个属于自己的 Dec_class 类对象来完成解码任务的。

3.2 完成 JEPG2000 解码程序的多线程编写

多线程程序结构设计是整个设计的核心部分，也是本文的重点所在。为了实现多路并行解码的结构，本文设计了如下的线程结构。现以 4 线程程序编写为例来进行介绍：

整个设计框图如图 3 所示：

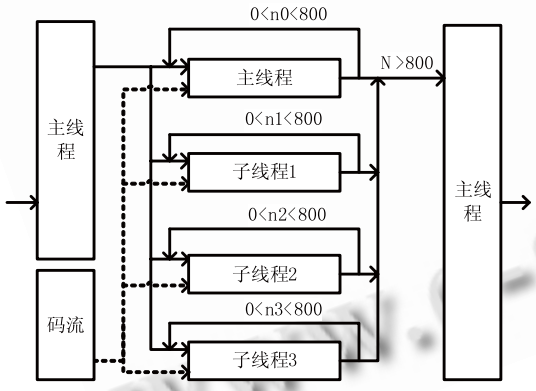

图 3 JEPG2000 解码多线程程序设计框图

首先在程序开始的时候生成主线程。然后由主线程生成 3 路子线程，并和主线程一起形成 4 线程并行结构。3 路子线程和主线程分别生成一个 Dec_clas 子类对象负责一路输入码流的解码工作。当每路线程都完成自己的任务以后，释放 3 路子线程。主线程负责整个程序的后续工作。

具体实现过程包括：

3.2.1 主线程任务的具体实现

主线程负责整个程序初期的初始化工作，包括线程句柄的设定，线程句柄空间的开辟，码流的链接定位，时间函数的设定等。并负责子线程的生成。

设计中需要开辟 3 路子线程和主线程并行处理，所以需要建立 3 个线程句柄，使用语句：

```
HANDLE    *hThread;
hThread=(HANDLE*)realloc(hThread,sizeof(HANDLE)*3);
```

来建立线程句柄并给线程句柄开辟内存空间。线程句柄建立以后需要将码流和各个线程的输入链接在一起，并且将解码以后的数据和输出文件链接在一起。使用：

```
strcpy(Lena.cmpfile,argv[1]);
strcpy(Lena.outputfile,argv[2]);
```

来完成这个操作。其中 Lena.cmpfile 和 Lena.outputfile 分别是 lena 图像的输入和输出文件。argv[1]和 argv[2]是子线程 1 的输入和输出。至此，主线程的准备工作已经做好，建立时间函数来统计程序运行时间。

3.2.2 子线程的建立和具体任务分配

子线程由主线程调用线程创建函数 CreateThread(psa,cbStack,pfnStartAddr,

pvParam,fdwCreate,pdwThreadID)

参数取 NULL,0,function0,NULL,NULL,NULL。

其中需要说明的是 function0 函数，它是线程函数(WINAPI 函数)，每个线程的任务都放在线程函数里边。在线程运行的时候执行里边的任务。在本设计中每个线程需要生成一个 Dec_class 类对象后，顺序调用里边函数完成一路 JEPG2000 图像的解码任务。

3.2.3 线程同步

由于本设计中 4 路线程采取并行结构，并且每一路分别解码一路输入，相互之间没有使用共同的资源，所以不涉及共享资源的问题，线程同步实现起来比较简单。

本设计中只需在所有子线程结束的时候建立一个事件同步类对象来做一下所有子线程任务完成判断，通知主线程可以做下一步的工作：销毁子线程，释放内存空间。使用函数 WaitForMultipleObjects (1, hThread,TRUE,INFINITE)来完成同步类对象的创建工

作。

3.2.4 计时模块的设计

根据对实际解码速度的测试，单幅图像的解码完成在毫秒级，很难对解码的速度进行准确的判断。为了增加设计的准确性，减小设计的误差。设计中使用循环解码的设计来增加线程任务量。如图 5 所示，每一个线程都循环解码图像 800 次。分别使用变量 n0,n1,n2,n3 来判断主线程和 3 个子线程的解码次数，如果解码次数小于 800 则跳到线程初始位置开始重新开始图像的解码，同时线程解码次数加 1。WaitForMultipleObjects(1,hThread,TRUE,INFINITE)函数会判断每一路线程的解码状态，只有所有线程都完成预定的任务以后，才会进行后续的工作。

计时函数需要在主线程建立子线程以前开始计时，在 WaitForMultipleObjects(1,hThread,TRUE,INFINITE)函数之后结束计时，这样可以计时整个图像解码的时间，为后续的效率分析工作提供数据。

3.3 利用标准测试图像完成测试工作

测试图像 Lena、Water、Baboon、pentagon、Bridge1 的解码速度，分别对每个图像连续解码 800 次，重复实验 50 次，得出平均时间作为解码的平均速度计入记录。

4 试验结果

本设计为了比较多线程程序对解码速度的影响使用了 4 台计算机平台进行对比测试。

为了得到更有对比性的数据，本设计分别在 3 个硬件平台上运行了单线程程序，双线程程序和四线程程序。得到的实验数据分别列于表 4，表 5，表 6 和表 7 中。其中 N 为线程个数，解码事件单位为秒(s)。

表 1 为单核计算机解码 800 副图像所需时间随线程个数(N)变化情况表：

表 1　单核 cpu 执行时间表

	Lena	Barbara	Elaine	Mountain
N=1	92.78	97.09	100.40	85.14
N=2	92.40	97.59	100.97	85.84
N=4	92.50	97.04	100.97	85.52
N=8	92.30	97.09	100.54	85.87

可以清楚的看出，在单核计算机上多线程程序解码速度和单线程几乎一样，并没有明显提高解码速度。

表 2 为双核计算机解码 800 副图像所需时间随线程个数(N)变化情况表：

表 2　双核 cpu 执行时间表

	Lena	Barbara	Elaine	Mountain
N=1	60.37	64.14	63.70	55.79
N=2	31.56	32.93	33.21	28.06
N=4	31.67	33.03	33.12	28.06
N=8	30.67	32.12	32.09	27.93

表 2 显示，N=1（单线程）解码速度明显比 N=2,4,8 时慢一半左右，但是 N=2,3,4 时解码速度基本相同。

表 3 为四核计算机解码 800 副图像所需时间随线程个数(N)变化情况表：

表 3　四核 cpu 执行时间表

	Lena	Barbara	Elaine	Mountain
N=1	56.76	59.87	60.37	52.85
N=2	29.29	30.57	30.64	26.15
N=4	14.75	15.65	15.46	13.23
N=8	14.75	15.53	15.37	13.40

表 3 显示，N=1（单线程）解码速度约为 N=2 时解码速度的一半，N=2 时解码速度约为 N=4,8 时的一半。当 N>=4 时解码速度约为单核的四倍左右。

表 4 为四核处理器解码 800 副图像所需时间随线程个数（N）变化情况表：

表 4　八核 cpu 执行时间表

	Lena	Barbara	Elaine	Mountain
N=1	56.96	60.07	60.55	53.07
N=2	29.49	30.66	30.96	26.25
N=4	14.95	15.85	15.66	13.32
N=8	7.35	7.80	7.74	6.70

表 4 显示，随着计算机核心个数增加，在 N=1,2,4,8 时解码速度基本也是等比增长的，8 线程程序解码速度约为单线程的 8 倍，4 线程约为单线程的 4 倍，双线程约为单线程解码速度的两倍。

5 结论

综合表 3，4，5，6，7。可以得出以下几条结论：

1) 单线程程序无论电脑 cpu 核心个数是多少，图

（下转第 160 页）

从 HResults 的结果看出有一个识别错误，检查 recout.mlf 文件，发现 tc00.rec 和 tc01.rec 都被识别成了 YI。再检查 hmm7 中 hmmdefs 里 yi 和 ling 的模型数据，对比发现两者数据一致。到存放训练数据的 data 文件夹中通过 HSLab 调出之前录制的 sig 文件，发现 ling 和 yi 的语音数据出现了重复，需要对 ling 和 yi 的训练语音进行重新录制。用新的数据建模后得到新的识别结果，如图 3 所示，执行 HResults 后得到识别率达到了 100%。

4 结论

中英文特定词语音识别系统，一方面对中英文语音同时建模，使其能够同时识别中英文的语音。但由于中英文发音本身的差异，只在单词级进行识别。另一方面，为了增加系统的扩展性以及减少模型建立的时间，通过代码控制整个系统的建模过程。控制代码能够在给出训练语音和词典的情况下建立特定词语音识别系统，并且特定词不限于中英文的单词，可以是选择的任何语种的单词，语音的识别单元都是单词级，建立的语音识别系统属于孤立词语音识别系统。

参考文献

1 易克初,田斌,付强.语音信号处理.北京:国防工业出版社,2000.191－259.
2 石现峰,张学智,张峰.基于 HTK 的语音识别系统设计.计算机技术与发展,2006,16(10):37－38.
3 Young S, Evermann G, Gales M.The HTK Book. Cambridge UniversityEngineering Department. Version 3.1, 2001.40－120.
4 孙宁,孙劲光,孙宇.基于神经网络的语音识别技术研究.计算机与数字工程,2006,34(3):58－61.
5 何湘智.语音识别的研究和发展.计算机与现代化,2002,79(3):3－6.

(上接第 148 页)

像处理速度取决于单个 cpu 核心最高速度。其余核心在程序运行期间不能得到充分的利用。

2) 多线程程序的图像处理速度取决于线程个数和核心个数的最小值。

3) 当核心个数大于线程个数的时候，图像解码的核心个数等于线程个数，图像解码的速度取决于线程个数；当线程个数大于核心个数的时候，系统将各个线程优化到每个核心里边，图像解码的速度取决于核心个数。

4) 将多线程技术应用在 JEPG2000 图像解码中，可以充分利用多核心计算机的资源，提高图像解码速度。在线程个数大于电脑核心个数的时候，双核、四核、八核的计算机，相对单核处理速度分别可以提高两倍，四倍和八倍。

参考文献

1 张困申,晓黄.Api for Windows98/2000.北京:清华大学出版社,2001.
2 Jeffery R. Programming Applications for Microsoft Windows, 1996.61－62.
3 Jim B, Robert W.Multithreading Applications in Win32, 2002.
4 Jim B, et al. 侯捷译. WIN32 多线程程序设计.武汉:华中科技大学出版社,2002.
5 ISO/IEC 1.29.15444 JPEG-2000: JPEG 2000 Part I Final Draft International Standard. Cupertino: SO/IEC, 2000.
6 Wang L.The Core Algorithm and New Developments of JPEG2000 Standard. Tracks for Standard & Technology, 2005: 67－69.
7 刘凯,李云松,吴成柯.一种比特平面并行处理的零数编码结构.电路与系统学报,2005,(5):23－24.
8 邓家先.遥感图像编码技术研究[博士学位论文].西安:西安电子科技大学,2004.