一种改进的基于无链表 SPIHT 的图像压缩算法

王建军[1,2],刘 波[2]

1. 中国科学院西安光学精密机械研究所,西安 710119
2. 中国科学院空间科学与应用研究中心,北京 100190

摘要 SPIHT 算法以其简单高效而著称,但由于 LSP、LIP 和 LIS 3 个链表的使用,内存需求量大,且需要动态分配或删除链表节点;另外,排序阶段存在的重复扫描也严重影响了算法的效率和性能,因此算法不易在硬件平台上实习,也不适用于低内存和实时应用场合。本文针对 SPIHT 算法的不足,提出了一种改进的无链表 SPIHT 算法。首先,在排序阶段加入对 A 类集合的分类判断,优化了码流输出,提高了压缩性能;其次,在存储重要信息时,算法以状态标识矩阵代替链表,既节约了内存开销也避免了内存的动态管理,最大输出位数和集合极值矩阵的使用则减少了扫描次数,提高了运行效率。

关键词 图像压缩;小波变化;SPIHT 算法;无链表 SPIHT 算法

中图分类号 TN919.81 **文献标识码** A **文章编号** 1000-7857(2010)06-0042-04

Improved Listless SPIHT Based Image Compression Algorithm

WANG Jianjun[1,2], LIU Bo[2]

1. *Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an 710119, China*
2. *Center for Space Science and Applied Research, Chinese Academy of Sciences, Beijing 100190, China*

Abstract The algorithm of Set Partitioning in Hierarchical Trees (SPIHT) is well known for its simplicity and efficiency. However, the use of three lists (LSP, LIP and LIS list) requires a high, variable and data dependant memory. Besides this, a repeated scanning also reduces the algorithm's efficiency to make it difficult for the SPIHT algorithm to be implemented in a hardware platform, especially for low memory and real time applications. In order to solve this problem, a modified listless SPIHT based algorithm is proposed in this paper. Firstly, the state mark matrixes are used to replace three lists (LSP, LIP and LSP list) for the significant information, and the modified SPIHT algorithm saves memory and avoids the dynamic memory management. The use of the maximum value array and the number of the maximum output bits also reduces the degree of scanning in the sorting pass and enhances the algorithm's efficiency. Secondly, in the modified SPIHT algorithm a new test is added for the type A set to optimize the output bit stream, which improves the algorithm's performance.

Keywords image compression; wavelet transform; SPIHT algorithm; listless SPIHT algorithm

0 引言

近年来,随着数字图像的广泛应用,图像的数据量激增,为了实现图像的有效存储与传输,必须对图像数据进行压缩。在图像压缩领域,基于小波变换的编码方法研究成果颇丰。在过去 20 多年中,人们提出了许多极富价值的基于小波变换的压缩算法,如 Shapiro 的嵌入式零树编码算法(Embedded Zerotree Wavelet,EZW)[1]、Said 和 Pearlman 的分级树集合分裂算法(Set Partitioning in Hierarchical Trees,SPIHT)[2]、Taubman

收稿日期:2009-12-25
基金项目:中国科学院空间科学与应用研究中心青年创新基金项目(O8211DA29S)
作者简介:王建军,博士研究生,研究方向为图像处理与编码技术,电子信箱:dayu504@126.com;刘波(通信作者),研究员,研究方向为空间信息与仿真技术,电子信箱:boliu@cssar.ac.cn

的嵌入式优化截断块编码算法（Embedded Block Coding with Optimized Truncation，EBCOT）[3]。

在这些算法中，SPIHT 算法以其简单和高效而著称，它是 EZW 算法的改进，由于采用了更为有效的空间方向树结构和比特平面编码方法，不仅能够获得很高的压缩编码效果，而且能够产生嵌入式码流，从而支持解码器的多码率解码与渐近传输[4]；而与 EBCOT 算法相比，SPIHT 算法虽然在压缩性能上稍低，但由于其内部无率失真优化单元，计算复杂度远低于 EBCOT 算法。

虽然 SPIHT 算法取得了很大的成功，但理论分析和实验结果表明，该算法也存在一些不足。在编码过程中，当一个集合重要而直接子节点全部非重要时，需要对 4 个系数重复编码 4 次并输出 4 个 0，影响算法性能；在实现方面，量化编码过程中存在着大量影响算法效率的重复扫描，且使用 3 个动态链表存储编解码过程中的重要信息，内存需求量极大，动态分配节点、删除节点等链表操作也不利于硬件实现。这些都已成为了 SPIHT 算法在实时、低内存硬件平台上实现的瓶颈。为此，本文提出一种改进的无链表 SPIHT 算法。首先，算法通过对 A 类集合加入分类判断，优化了码流输出，提高了算法的性能；再次，算法通过状态标志矩阵代替动态链表和其他提高迭代效率的方法，提高了执行效率，降低了内存消耗，有效地避免了内存的动态管理。

1 SPIHT 算法

SPIHT 算法利用小波变换能量的集中性和小波系数各级子带的相似性，在特殊的扫描方式下，能有效地从高能量层到低能量层渐进编码，优化系数输出[5]。SPIHT 编码器使用如下集合表示小波系数的空间方向树结构：$D(i,j)$表示位于(i,j)位置所有子孙的坐标集合（A 类集合）；$O(i,j)$表示位于(i,j)位置小波系数的直接子节点的坐标集合；$L(i,j)=D(i,j)-O(i,j)$（B 类集合）；$H(i,j)$表示所有根结点的集合；$C(i,j)$表示(i,j)位置的小波系数值。用 $S_n(U)$表示一个集合 U 的重要性，用 T 表示当前扫描阈值。对于一个系数，如果 $C(i,j)\geqslant T$，则 $S_n(C(i,j))=1$，称 $C(i,j)$为重要系数，否则 $S_n(C(i,j))=0$，称 $C(i,j)$为非重要系数；对于一个集合 U，如果 $\max\{C(i,j)\}\geqslant T,(i,j)\in U$，则 $S_n(U)=1$，称 U 为重要集合，否则 $S_n(U)=0$，称 U 为非重要集合。在算法执行过程中，使用 3 个链表记录相关信息：不重要集合链表 LIS，不重要像素链表 LIP，重要像素链表 LSP。具体算法如下。

1）初始化。输出 $n_{max}=\|\log_2\max(C(i,j))\|$，置 LSP 为空表，将坐标$(i,j)\in H$ 加入 LIP 表，将 H 中有后代的像素加入 LIS 表，并置为 A 类集合。

2）排序。对 LIP 的每项(i,j)做如下操作：输出 $S_n(C(i,j))$，如果 $S_n(C(i,j))=1$，将(i,j)移入 LSP 表，并输出 $C(i,j)$的符号。

对 LIS 表中的每项(i,j)进行如下操作。

① 如果(i,j)是 A 类集合，则输出 $S_n(D(i,j))$；如果 $S_n(D(i,j))=1$，则将 $D(i,j)$分裂为 $L(i,j)$和 4 个 $C(k,l)\in O(i,j)$，输出每一个 $S_n(C(k,l))$的值；如果 $S_n(C(k,l))=1$，将(k,l)移入 LSP，并输出 $C(k,l)$的符号；如果 $L(i,j)$不为空集，将(i,j)作为 B 类集合移至 LIS 表末，转步骤②，否则，将(i,j)从 LIS 表中删除。

② 如果(i,j)是 B 类集合，则输出 $S_n(L(i,j))$；如果 $S_n(L(i,j))=1$，则将每一个 $C(k,l)\in O(i,j)$作为 A 类集合加入到 LIS 末尾，并将(i,j)从 LIS 表中删除。

3）细化。输出 LSP 链表中每一项(i,j)的第 n 位。

4）如果 $n>0$，且没有达到要求码率，返回 2），否则算法终止。

2 改进无链表 SPIHT 算法

2.1 A 类集合直接子节点的重要性判断与性能改进

如上所述，在 SPIHT 算法中，如果在排序阶段判断一个 A 类集合为重要集合时，如果直接子节点的集合 $O(i,j)$对应的小波系数全部为非重要系数时，需要连续重复输出 4bit 的 0。即：在已知 $S_n(O(i,j))=0$ 的情况下，仍需对 4 个已知的非重要系数逐个编码，因此在对 A 类集合的非重要直接节点系数编码时，SPIHT 算法消耗了大量的比特数。

为此，如果 A 类集合重要时，本文加入了对直接子节点集合的判断，并根据直接子节点集合的重要性，将其分为如下两类：

S_0：A 类集合重要，且直接子节点集合为非重要集合，即 $S_n(O(i,j))=0$；

S_1：A 类集合重要，且直接子节点集合为重要集合，即 $S_n(O(i,j))=1$。

国际标准测试图像 Lena 在排序的不同阶段，S_0 与 S_1 所占比例如表 1 所示。

表 1 Lena 图像在排序不同阶段 S_0 和 S_1 所占比例

Table 1 Ratio of S_0 and S_1 for different phase of sorting pass for Lena

	排序前期/%	排序中期/%	排序后期/%
S_0	41.37	45.66	43.72
S_1	58.63	54.34	56.28

本次改进中，加入了对 A 类集合直接子节点的判断。如果集合重要，且集合属于 S_0 型，输出一个 1 用以表明集合重要，然后直接输出一个 0 用以表明 4 个直接子节点非重要，并将它们加入到 LIP 表中；如果集合重要，且集合属于 S_1 型，输出一个 1 用以表明集合重要，然后输出一个 1 表明集合属于 S_1 型，接着，与 SPIHT 算法类似，输出 4 个直接子节点的重要性判断结果，如为非重要系数，则加入到 LIP 表中，如为重要系数，则加入 LSP 表中，并输出系数符号标识。

在改进算法中，虽然加入了一位数据用以判断 A 类重要集合的类型，但当集合属于 S_0 型，只需 1 位 0 即可标识 4 个

子节点的重要性,而在 SPIHT 中,需要输出 4 位 0,因此,算法节约了 3bit 的码流输出;当集合属于 S_1 型,算法较原始 SPIHT 额外输出一位 1,其他码流开销与 SPIHT 的相等。由表 1 知,对于 Lena 图像,在排序的不同阶段,均满足 $3S_0>S_1$,因此,改进算法节约的码流远大于类型判断增加的码流开销,从而使压缩性能得到提升。而实验结果进一步证明 Barbara、Baboon 等其他国际标准测试图像在排序的不同阶段也均满足 $3S_0>S_1$ 条件。

2.2 算法的无链表改进

SPIHT 算法中,由于 3 个动态链表的使用,算法内存消耗大,且插入、删除等链表操作频繁,内存资源需要动态管理,因而不易于硬件实现,也不适用于低内存和实时应用场合。为此,在本次改进中,改进 SPIHT 算法不再使用 LIP、LSP 和 LIS 3 个链表,取而代之以状态标识矩阵来存储关键系数和集合信息,从而降低内存消耗并避免内存的动态管理;其次,为提升算法效率,合并排序与细化过程;最后,使用集合极值矩阵和最大输出位数进一步提高算法的迭代效率。算法的改进思想具体如下。

1)状态标识矩阵:在本次改进中,编码过程的关键信息存储在 2 个状态标识矩阵:非重要系数状态标识矩阵 SMBIP 和非重要集合状态标识矩阵 SMBIS。SMBIP 用来标识非重要系数的位置,其大小与原始图像大小相等,且矩阵单元元素仅为 1bit;SMBIS 用来标识没有被分割过的零树集合,其大小为图像大小的 1/4。

2)最大输出位数:对于渐近图像编码,编码过程是一个随阈值递减而逐渐扫描的过程。每次扫描,重要系数的有效比特位被输出,当量化扫描达到要求的比特率时,扫描停止,重要系数的低比特位被忽略。实验证明,在相同的压缩率,被忽略的低比特位数大致相等[6]。在本次改进中,根据编码的比特率要求,预先设定最大输位数 NMOB,用以表明重要像素的最大输出位数。当一个系数重要时,直接输出当前位到第 NMOB 位的二进制表示。使用这种方式,改进算法将排序和细化两个阶段合并,不需要额外的链表或状态标识矩阵存储重要系数的位置信息,这样既节约了内存消耗也提升了算法的效率。

3)集合极值矩阵:在 SPIHT 算法中,在零树集合不断分割的过程中,编码器需要不断重复扫描零树集合的系数,以判断集合的重要性。因此为了改进算法的效率,在改进 SPIHT 算法中,引入集合极值矩阵 MVA 存储各零树集合的最大值。零树集合采用由底至上的方法,在算法初始化确定极大值时自动生成,因而可有效避免扫描阶段的重复判断。MVA 的大小为图像大小的 1/4。

综上所述,在改进算法中,编码器的重要信息被存储在了 SMBIP、SMBIS 和 MVA 中,因此内存需求下降,而且避免了动态管理。对于一幅 512×512 大小的 8 位灰度图像,SPIHT 算法需用 18bit 存储系数的坐标信息(行 9 位,列 9 位),如果在编码过程中,LSP、LIP 和 LIS 3 个链表所包含的实体数最多为像素数的 2 倍,则所需内存量为

$$512\times512\times2\times18/(1024\times1024\times8)=1.125\text{MB}$$

而改进算法所需的内存量则为(假设 MVA 单位元素用 16bit 存储)

$$(512\times512\times1+256\times256\times1+256\times256\times16)/(1024\times8)=168\text{kb}$$

另外,NMOB 和 MVA 的使用也提高了算法的效率,所以算法具有快速、简单、易于硬件实现的特点。

2.3 算法完整描述

1)初始化

① 内存分配:为 SMBIP 分配与原始图像大小相等单位元素为 1bit 的二维矩阵;为 SMBIS 分配原始图像 1/4 大小单位元素为 1bit 的二维矩阵;为 MVA 分配原始图像 1/4 大小单位元素为 16bit 的二维矩阵。

② 由最高频开始,从下至上遍历小波系数矩阵,以每一集合的最大值初始化 MVA 矩阵,并输出 $n_{max}=\|\log_2\max(C(i,j))\|$,设置 $n=n_{max}$。

③ 根据图像重建的比特率要求,设置 $NMOB=n_e$ 并输出。

④ 如果 $(i,j)\in H$(H 为低频子带集合),设置 $SMBIP(i,j)=1$,$SMBIS(i,j)=1$;否则,设置 $SMBIP(i,j)=1$,$SMBIS(i,j)=1$。

2)排序

① 遍历 SMBIP

• 如果 $SMBIP(i,j)=1$ 且 $S_n(C(i,j))=1$,设置 $SMBIP(i,j)=0$,输出一位 1 以及 $C(i,j)$的符号标识,输出系数的第 $n-1$ 位到 n_e 位的二进制表示。

• 如果 $SMBIP(i,j)=1$ 且 $S_n(C(i,j))=0$,输出一位 0。

② 遍历 SMBIS

• 若 $SMBIS(i,j)=1$ 且 $MVA(i,j)<2^n$,输出一位 0。

• 若 $SMBIS(i,j)=1$ 且 $MVA(i,j)>2^n$,输出一位 1,设置 $SMBIS(i,j)=0$。如果 $D(i,j)\in S_0$,输出一位 0,对于每一个系数 $C(k,l)\in O(i,j)$,设置 $SMBIS(k,l)=1$,设置 $SMBIP(k,l)=1$;如果 $D(i,j)\in S_1$,输出一位 1,对于每一个系数 $C(k,l)\in O(i,j)$,设置 $SMBIS(k,l)=1$,如果 $S_n(C(k,l))=1$,设置 $SMBIP(k,l)=0$,输出一位 1 以及 $C(i,j)$的符号标识,输出系数的第 $n-1$ 位到 n_e 位的二进制表示,如果 $S_n(C(k,l))=0$,设置 $SMBIP(k,l)=1$,输出一位 0。

3)更新

设置 $n=n-1$,如果 $n<NMOB$ 或者已近达到要求的比特率,算法终止;否则,转 2)。

3 实验结果

为了验证算法的高效性和实用性,本文选用 5 级 9/7 提升小波变换[7]和如图 1 所示空间方向树结构[8],在 Windows XP,Matlab 7.0 环境下对 512bit×512bit×8bit 国际标准测试图像 Lena 进行了仿真实验。

表 2 列出了提出的改进算法与原始 SPIHT 算法在不同

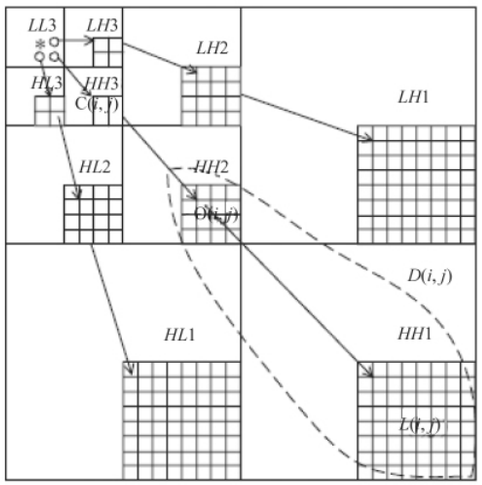

图 1　小波系数空间方向树结构

Fig. 1　Space orientation tree structure for wavelet coefficient

表 2　改进算法与 SPIHT 算法的性能比较

Table 2　Performance comparison between the modified algorithm and the original SPIHT

比特率/bpp	编码方法	PSNR/dB	耗时/ms
0.25	本文算法,*NMOB*=3	34.37	107
	SPIHT 算法	33.61	284
0.50	本文算法,*NMOB*=6	37.91	146
	SPIHT 算法	36.88	378
0.75	本文算法,*NMOB*=8	39.76	198
	SPIHT 算法	38.53	458

比特率下 PSNR 和 CPU 运算时间的比较。由表 2 可知,改进算法在 PSNR 和编码速度方面较 SPIHT 算法有显著的提高。图 2 显示了改进算法在不同比特率下的重建图像,由图 2 可知,重建图像在不同的比特率下均具有良好的视觉效果。

(a) 0.25bpp,*NMOB*=3
(a) 0.25bpp with *NMOB*=3

(b) 0.50bpp,*NMOB*=6
(b) 0.50bpp with *NMOB*=6

(c) 0.75bpp,*NMOB*=8
(c) 0.75bpp with *NMOB*=8

图 2　图像重建结果

Fig. 2　Reconstructed image

4　结论

针对 SPIHT 算法存在的不足,本文提出了一种改进的无链表 SPIHT 图像压缩算法。首先,算法通过对 A 类集合加入判断,优化了码流输出,提升了压缩性能;其次利用状态标识矩阵代替动态链表,节约了内存消耗,避免了内存的动态管理,集合极大值矩阵和最大输出位数的使用改变了扫描方式,减少了重要性判断时的比较次数。实验结果表明,算法在 PSNR 和编码速度方面较原始 SPIHT 算法都有显著的提高。通过本文对改进 SPIHT 算法的讨论可以看出,改进算法易于在硬件平台上实现,且适用于低内存和实时应用场合。

参考文献(References)

[1] Shapiro J M. Embedded image coding using zerotrees of wavelet coefficients [J]. *IEEE Transactions on Signal Processing*, 1993, 41(12): 3445-3462.

[2] Said A, Pearlman W. A new, fast, and efficient image codec based on set partitioning in hierarchical trees [J]. *IEEE Transactions on Circuit and Systems for video technology*, 1996, 6(3): 243-250.

[3] Taubman D. High performance scalable image compression with EBCOT [J]. *IEEE Transactions on Image Processing*, 2000, 9(7): 1158-1170.

[4] 张春田, 苏育挺, 张静. 数字图像压缩编码[M]. 北京: 清华大学出版社, 2004: 234-247.
Zhang Chuntian, Su Yuting, Zhang Jin. Digital image compression and coding[M]. Beijing: Tsinghua University Press, 2004: 234-247.

[5] Muzaffar T, Choi T S. Linked significant tree wavelet based image compression [J]. *IEEE Transactions on Signal Process*, 1993, 88(10): 2554-2563.

[6] Sun Y, Zhang H, Hu G. Real-time implementation of a new low-memory SPIHT image coding algorithm using DSP chip [J]. *IEEE Transactions on Signal Processing*, 2002, 11(9): 1112-1116.

[7] Daubechies I, Sweldens W. Factoring wavelet transforms into lifting steps [J]. *Journal of Fourier Analysis and Its Applications*, 1998, 4(3): 247-269.

[8] Pan H, Siu W C, Law N F. A fast and low memory image coding algorithm based on lifting wavelet transform and modified SPIHT [J]. *Signal Processing: Image Communication*, 2008, 23(3): 146-161.

(责任编辑　朱宇)