文章编号:1001 - 9081(2006)06 - 1362 - 03

基于提升小波的 SPIHT 改进算法

陈红新,刘正光,张宏伟,杨正瓴
(天津大学 电气与自动化工程学院,天津 300072)
(calmdown123@ msn. com)

摘 要:针对多级树集合分裂算法(Set Partitioning In Hierarchical Tree,SPIHT)复杂的特点,采用整数实现的提升格式代替了原来的小波变换,简化了计算过程。对小波系数采用基于块的结构划分,在频域重新建立空间方向树。根据块中相邻系数相关性建立上下文模型,对输出信息进行自适应算术编码,提高了编码效率。实验表明,在相同的比特率条件下,重构图像的 PSNR 值高于原算法

关键词:提升格式; 零树;多级树集合分裂算法;上下文;算术编码
中图分类号: TP391.41 **文献标识码**:A

Improvement of SPIHT algorithm based on lifting scheme

CHEN Hong-xin, LIU Zheng-guang, ZHANG Hong-wei, YANG Zheng-ling
(*School of Electrical and Automation Engineering, Tianjin University, Tianjin,* 300072, *China*)

Abstract: Due to the computational complexity of SPIHT, the integer-to-integer lifting scheme, which could simplify the computational process, was adopted instead of the normal wavelet transform. Wavelet coefficients were divided into blocks, based on which the spatial orientation trees were re-organized. The context-based model was constructed according to the correlativity of neighboring coefficients in blocks, and the outputting signals were encoded by adaptive arithmetic code to improve coding efficiency. Experiments show that the PSNR value is more than that of the original SPIHT at the same bit rate.

Key words: lifting scheme; zero-tree; SPIHT; context; arithmetic algorithm

在基于小波变换的图像压缩算法中,零树小波编码(EZW)是一种非常有效而且实现简单的方法。其输出的码流具有嵌入式特征,在解码端解码时,可以在任何位置停止码流的接收,与原码相比,这部分码流解码出的图像具有更低的质量或分辨率,但是解码出的图像是完整的。近年来,在嵌入式零树编码(EZW)算法的基础上出现了许多新的改进算法,SPIHT(多级树集合分裂算法)就是其中最为最典型的一个。

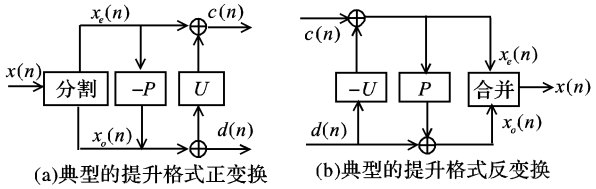
一般的小波变换采用浮点运算,计算量大,本文采用整数实现的提升格式实现小波变换,减少了计算量。将小波系数划分为块结构处理,减少了 LIS 和 LIP 链表的结点的个数,减小了内存分配。对扫描输出符号进行自适应算术编码时,可以充分利用相邻系数的相关性信息,建立更加有效的上下文进行模型,有效地提高了编码效率。

1 整数实现的小波变换

对图像进行第一代小波变换实际上是用二维离散小波采用 Mallat 的塔式分解方法,通过对图像水平和垂直方向交替采用低通和高通滤波得到各个子带的小波系数。

由于时域卷积在频域转化为乘积,所以第一代小波往往通过傅立叶变换在频域实现,计算量相当大。1994 年 Sweldens 提出了一种不依赖于傅立叶变换新的小波构造方法——提升格式。这种方法保持了原有的小波特性,又摆脱了上述局限性,称为第二代小波[1]。

提升格式包括分割、预测和更新三部分。其典型的正变换和反变换如图 1 所示。



(a)典型的提升格式正变换 (b)典型的提升格式反变换

图 1 正变换与反变换

系统采用 CDF9/7 双正交滤波器,低通滤波器有 9 个系数,高通滤波器有 7 个系数,分析和合成高通滤波器都有 4 阶消失矩,其提升格式的整数变换为:

$$\begin{aligned}
&d_i^0 = s_{2l+1}, s_i^0 = s_{2l}\\
&d_i^1 = d_i^0 + \lfloor \alpha \times (s_i^0 + s_{i+1}^0) + 1/2 \rfloor\\
&s_i^1 = s_i^0 + \lfloor \beta \times (d_{i-1}^1 + d_i^1) + 1/2 \rfloor\\
&d_i^2 = d_i^1 + \lfloor \gamma \times (s_i^1 + s_{i+1}^1) + 1/2 \rfloor\\
&s_i^2 = s_i^1 + \lfloor \delta \times (d_{i-1}^2 + d_i^2) + 1/2 \rfloor\\
&d_i = k_1 \times d_i^2, s_i = k_0 \times s_i^2\\
&k_0 = \rho, k_1 = k_0^{-1}
\end{aligned} \tag{1}$$

$\lfloor x \rfloor$ 表示不大于 x 的最大整数,+1/2 是为了进行四舍五入。在实际编程实现中,将比例因子 ρ 舍去,可真正实现无失真的整数到整数的小波分解和重建。

2 SPIHT 算法简介

SPIHT 算法称为多级树的零树分裂编码算法,是一种静态图像小波域系数的压缩方法,基本思想源自于零树编码,是

收稿日期:2005 - 12 - 29;**修订日期**:2006 - 03 - 06

作者简介:陈红新(1978 -),男,江苏盐城人,博士研究生,主要研究方向:图像压缩与编码、检测技术; 刘正光(1945 -),男,福建闽清人,教授,博士生导师,主要研究方向:模式识别、图像处理、电力电子技术、智能控制; 张宏伟(1972 -),男,河北承德人,博士研究生,主要研究方向:图像压缩与编码、图像分割与识别; 杨正瓴(1964 -),男,河北灵寿人,副教授,博士,主要研究方向:智能数据处理、混沌理论与应用.

零树编码方法的一种扩展和增强的方法[2]。该方法根据图像小波域系数与人眼视觉特性(HVS)的特点,在零树数据结构的基础上对系数进行分类扫描和精细扫描,在低比特率下仍可获得较满意的重构图像质量。SPIHT 算法是基于零树结构的一种编码方法,它充分利用了零树结构的基本原则,是典型的基于位平面的编码算法[4,5]。

编码中使用了三个表:不重要系数表 LIP(the list of insignificant pixels)、重要系数表 LSP(the list of significant pixels)和不重要集合表 LIS(the list of insignificant sets)。LSP 初始化为空表,LIP 用最低频子带系数(如三级分解中的 LL3、LH3、HL3、HH3 中的系数)初始化,LIS 用每一个空间方向树的根结点(如三级分解中的 LH3、HL3、HH3 中的系数位置)来初始化[3]。

编码的过程可以概括为:在不同的比特层面上,小波系数的在三个集合的转换以及重要性标志 *bit* 的输出,对重要图的确定主要是通过空间方向树的多次分裂来实现的,集合分裂的过程如图 2 所示。

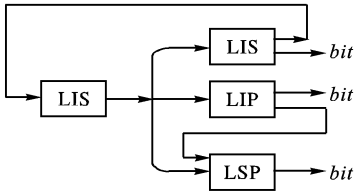


图 2　集合分裂示意图

3　基于块划分的 SPIHT 算法

3.1　上下文建模的原理基础

对图像直接进行熵编码并不能获得理想的压缩效果,原因是图像相邻像素之间存在着很强的相关性。通常情况下,信源先后发出的消息符号是相互依存的,具有这种特征的信源称为有记忆信源。有记忆信源的极限熵为:

$$\lim_{x\to\infty} H_n(X) = \lim_{x\to\infty} H(x_n \mid x_1, x_2, \cdots, x_{n-1}) \tag{2}$$

为了能进一步提高压缩性能,通常采用自适应算术编码对输出的信源符号进行编码。一般自适应算术编码器的效率可以通过用已知的信息为条件,对每个已知的条件采用不同的符号概率来估计来提高。在已经知道 Y 的条件下 X 的条件熵是 $H(X \mid Y)$,并且:

$$H(X \mid Y) \leqslant H(X) \tag{3}$$

以 Y 为上下文的条件自适应算术编码通常产生较小的比特率的比特流。如果按序列地标记编码符号序列 $x_1, x_2, \cdots, x_n$,为了最小化编码码率,编码序列使用的平均比特数应该等于:

$$-\log_2 \prod_{i=1}^{n} P(x_1 \mid x_{i-1}, \cdots, x_1) \tag{4}$$

但是 $P(x_1 \mid x_{i-1}, \cdots, x_1)$,$1 \leqslant i \leqslant n$ 实际上通常是未知的,而且在编码过程中必须基于过去观察在线地估计出。估计 $P(x_1 \mid x_{i-1}, \cdots, x_1)$ 的过程就称为信源的概率建模。当前符号的概率所基于的过去观察值集合别称为建模的上下文。上下文的选取对概率模型的产生有很大的影响,决定着编码序列的比特率[6,7]。

3.2　基于块划分的算法描述

在 SPIHT 算法中首先将所有小波系数分为 2×2 的数据块,如图 3 所示。图中灰色的部分为低频子带,每个块包含四个数据。这样分的目的是为了在后续的算术编码中可以利用本块内相邻系数据的相关性建立上下文。

在图 3 中,每个小正方形代表一个系数,灰色部分为低频子带。图中 (i,j) 为 *LL* 子带数据块的左上角数据坐标,符号“×”代表其直接子女块,符号 ●、★ 和 ☆ 分别代表 (i,j) 在三个方向的子孙块。从图中可以看出,与标准 SPIHT 算法相比,所有后代的数目减少为以前的 1/4。

将小波系数采用分块的方法划分后,大多数的操作都是基于块的。在初始化阶段,将 *LL* 子带中经过 2×2 分块后,每块的左上角的系数输入到 LIP 链表,将这些系数的坐标输入到 LIS,将重要性位图 LSP 设置为全零。

由图可以看出在块中左上角系数有 3 个子女,而其他系数在相邻的高频子带中对应位置有 4 个子女,最高频的系数没有子女。

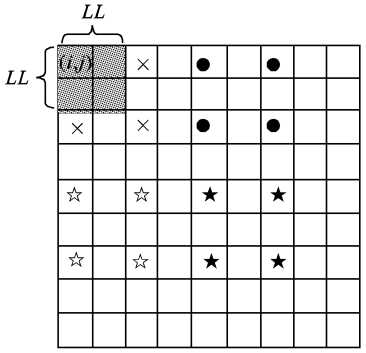


图 3　基于块的空间方向树

对于无效系数集合 LIP 的重要性扫描过程可作如下描述:

1)根据坐标取数据块,根据块的状态得到此系数的上下文状态,用于后面的概率估计。

2)对该系数进行重要性扫描,测试结果放入 *bit*。对于比特层 n 的系数重要性定义如下:

$$bit = S_n(i,j) = \begin{cases} 1, & \text{如果} \mid S(i,j) \mid \geqslant \text{当前阈值} \\ 0, & \text{其他情况} \end{cases}$$

3)根据上下文值 m_1,对 *bit* 进行自适应算术编码。

4)更新块结构的上下文状态,用于下一个数据的扫描。

5)将该数据移至重要系数集合 LSP,并且将 (i,j) 系数值减去阈值,等待精细扫描。

6)将该系数的符号位(表示其正负)进行自适应算术编码,上下文模型为 m_2。

7)此数据块的 LIP 扫描结束。

在步骤(3)自适应算术编码中采用了基于块的上下文模型 m_1,结构如图 3。

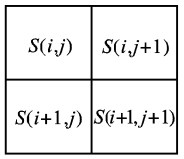


图 4　上下文系数分布

这种上下文模型的好处在于,充分利用子带内部的系数相关性,而且实现简单。图中每个矩形方框代表一个系数,每个系数有两种重要性标志 0 和 1,组合起来共有 $2^4 = 16$ 种上下文。如果用如下的方法表示图中 4 个系数的重要性:

$$C_1 = S(i,j) \tag{5}$$

$$C_2 = S(i,j+1) \tag{6}$$

$$C_3 = S(i+1,j) \tag{7}$$

$$C_4 = S(i+1,j+1) \tag{8}$$

则在本算法中的上下文序号为:

$$m_1 = 2^{C_1}2^3 + 2^{C_2}2^2 + 2^{C_3}2 + 2^{C_4} \tag{9}$$

范围是 0 ~ 15,共 16 个序号。因为这 4 个系数中已经包含所需编码的系数,而且在此系数编码前的比特层里,它必定是不重要的,即 C_1、C_2、C_3 和 C_4 中至少有一个为 0,所以实际中上下文序号范围为 0 ~ 14,共 15 个序号。最大值的序号出现在右下角的系数扫描时,其他 3 个系数的重要性标志为 1。

在系数由不重要向重要转变时,除了输出 *bit* 外,还需要输出该系数的符号位。虽然相邻之间系数的符号位有一定的联系,但是实验表明,对其建立上下文概率模型 m_2 对整个编码的意义不大,认为其上下文数目为 1,序号为:

$$m_2 = 0$$

对 LIS 的扫描分为两种,对 $D(i,j)$ 和 $L(i,j)$ 扫描。$D(i,j)$ 代表位于 (i,j) 位置的小波系数的所有子孙的坐标集合,$L(i,j)$ 为 (i,j) 系数的所有子孙的坐标中除去直接子孙 $O(i,j)$ 坐

标的部分。

对于 $D(i,j)$ 的描述如下：

1）根据当前阈值，判断 $D(i,j)$ 是否有效，有效时 $S_n(D(i,j))$ 为 1，无效时为 0。判断时，先判断其直接子女 $O(i,j)$；

2）根据块状态设置上下文序号，对 $S_n(D(i,j))$ 位进行基于上下文算术编码，所用上下文为 m_3；

3）如果 $S_n(D(i,j))$ 为 1，则：

a）对每个属于直接子女块 $O(i,j)$ 进行重要性扫描，并进行算术编码，这里所用的上下文暂命名为 m_4；

b）对块中的每个系数进行重要性扫描，如果重要则将其加入 LSP，并且对其符号位编码，所用上下文为 m_2；

c）如果上述块中还存在不重要的系数，则代表此块不重要，将其加入到 LIP。

4）如果此系数的 $L(i,j)$ 集合不为空，则将此集合加入到 LIS 尾部，并且将其集合类型改为 $L(i,j)$。如果为空，则将此集合删除，表明此为空间方向树的最末端，无后续叶节点。

对集合 $L(i,j)$ 的扫描与 $D(i,j)$ 类似，在算术编码过程中用到的上下文模型为 m_5。在上述描述中分别用到了上下文模型 m_3、m_4 和 m_5，这 3 个上下文的建立采用了和 m_1 类似的方法，都是利用了根数据块中 4 个系数的 $S_n(i,j)$ 确定上下文序列。但是和 m_1 也有区别，因为需要扫描的部分为独立于该数据块的后代，所以在公式(9)中，C_1、C_2、C_3 和 C_4 可以同时取值 0 或 1。因为上下文序号的范围为 0 ~ 15 共 16 种上下文。

经过了分类扫描过程，下一步对有效值表 LSP 进行细化处理。在精细扫描过程中，用到的上下文模型 m_6 与符号输出的上下文 m_2 一样，上下文个数为 1。

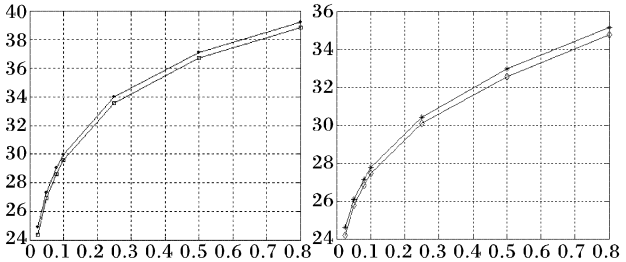
4 实验结果

本文采用 512 × 512 的标准测试图 Lena 和 Goldhill 对本文算法在不同比特率条件下进行测试。采用的小波为 CDF/97 提升小波，进行 5 级小波变换。在利用上下文估计概率以后，采用自适应的方法动态调整上下文，使之更利于后续符号的编码。压缩结果的对比如表 1、图 5 所示。

由实验数据可以看出，采用了整数提升格式和基于上下文的算术编码后压缩性能明显提高。在低比特率条件下，产生大量连续的无效位输出，这种情况下使用算术编码将提高系统的压缩性能。

表 1 压缩性能比较

Bit-rate	Lena		Goldhill	
	SPIHT	本文算法	SPIHT	本文算法
0.025	24.358	24.908	24.199	24.601
0.05	26.896	27.297	25.77	26.091
0.08	28.625	29.057	26.838	27.127
0.1	29.593	29.967	27.471	27.757
0.25	33.562	33.982	30.076	30.42
0.5	36.718	37.113	32.539	32.96
0.8	38.838	39.185	34.741	35.147



(a)Lena的两种算法对比图 (b)Goldhill的两种算法的对比图

图 5 实验结果

参考文献：

[1] DAUBECHIES I, SWELDENS W. Factoring Wavelet Transforms into lifting steps[J]. Journal of Fourier Analysis applications. 1998, 4(3): 245 - 267.

[2] SHAPIRO JM. Embeded image coding using zerotrees of waveletcoefficients[J]. IEEE Transaction on Signal Processing, 1993, 41: 3445 - 3462.

[3] SAID A, PEARLMAN W. Image compression using the spatial-orientation tree[A]. Proc. IEEE Intl, Symp[C]. Circuits and Systems, 1993. 279 - 282.

[4] SAID A, PEARLMAN WA. A new, fast and efficient image codec based on set partitioning in hierarchical trees[J]. IEEE Transaction on Circuit and Systems for video Technology, 1996, 6: 243 - 250.

[5] KHAN E, GHANBARI M. Efficient SPIHT based embedded color image coding[J]. IEEE Electronics Letters, 2001, 37: 951 - 952.

[6] WRIITEN LAN H, NEAL RADFORD M, CLEARY JOHN G. Arithmetic coding for Data Compression[J]. Comm of ACM, 1987, 30(6): 520 - 540.

[7] WU X, MEMO ND. Context-based adaptive lossless image coding [J]. IEEE Trans Commun, 1997, 45: 437 - 444.

(上接第 1356 页)

“Mather&Daughter”几乎保持不变，而运动大的序列“Stefan”只有微小的减少。这说明 UDCS 算法在保持同样性能的前提下，较大程度上提高了搜索速度。

根据表 1 可计算出 UDCS 相对 CDS 和 DS 的平均速度改善率(SIR)和平均 PSNR 的变化。对于视频会议序列，如“Mather&Daughter”，由于在中心点附近有高的中心偏置特性，因此 UDCS 相对 CD 和 CDS 算法分别可达到 84% 和 64% 的速度改善。对于相对大的运动序列，“Stefan”所得到的速度改善分别为 48%、38%。图 6(a)和 7(a)分别显示了序列“M&D”和“Stefan”的平均的搜索点数曲线。这些曲线清楚地表明了 UDCS 算法的搜索点数比其他 MBA 减少很多。图 6(b)和图 7(b)显示了两个序列相应 PSNR 性能与不同 BMA 的比较。这些图说明了 UDCS 算法的失真错误稍微大于 3SS、4SS、DS、和 CDS。

参考文献：

[1] THAM JY, RANGANATH S, RANGANATH M, *et al.* A novel unrestricted center-biased diamond search algorithm for block motion estimation[J]. IEEE Trans. Circuits Syst. Video Technol., 1998, 8(4): 369 - 377.

[2] ZHU S, MA KK. A new diamond search algorithm for fast block-matching motion estimation[J]. IEEE Trans. Image Process., 2000, 9(2): 287 - 290.

[3] ZHU C, LIN X, CHAU LP, *et al.* A novel hexagon-based search algorithm for fast block motion estimation[J]. in Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, 2001, 3: 1593 - 1596.

[4] FUKUNAGA S, NAKAYA Y, SON SH, *et al.* MPEG-4 video verification model version 14.0[Z]. ISO/IECJ TCI/SC29/WG11 MPEG99/N2932, Victoria, Australia, Oct. 1999.

[5] CHEUNG CH, PO LM. A novel cross-diamond search algorithm for fast block motion estimation[J]. IEEE Trans. Circuits Syst. Video Technol., 2002, 12(12): 1168 - 1177.

[6] CHEUNG CH, PO LM. Novel cross-diamond-hexagonal search algorithms for fast block motion estimation[J]. IEEE Trans. Multimedia, 2005, 7(1): 16 - 22.