Lapped Orthogonal Transform Coding by Amplitude and Group Partitioning

Xiangyu Zou[1] and William A. Pearlman[2]

Center for Digital Video and Media Research
Electrical, Computer & Systems Engineering Dept.
Rensselaer Polytechnic Institute
Troy, NY 12180-3590

ABSTRACT

Transform coding has been the focus of the research for image compression. In previous research, the Amplitude and Group Partitioning (AGP) coding scheme is proved to be a low complexity algorithm with high performance[1,2] , clearly one of the state-of-art transform coding techniques. However, the previous AGP is used along with the Discrete Cosine Transform (DCT) and the discrete wavelet transform. In this paper, a different transform, the Lapped Orthogonal Transform (LOT), replaces the DCT in conjunction with the AGP. This is the first time LOT and AGP have been combined in a coding method. The definition and design of the LOT are discussed. An objective metric to measure the performance of transform, coding gain, is calculated for both the DCT and the LOT. The LOT has slightly higher coding gain than the DCT. The principles of the LOT based AGP image codec (LOT-AGP) are presented and a complete codec, encoder and decoder, is implemented in software. The performance of the LOT-AGP is compared with other block transform coding schemes: the baseline JPEG codec[4] and the DCT based AGP image codec[1] (DCT-AGP) by both objective evaluation and subjective evaluation. The Peak Signal to Noise Ratio (PSNR) is calculated for these three coding schemes. The two AGP codecs are much better than the JPEG codec on PSNR, from about 1.7dB to 3 dB depending on bit rate. The two AGP schemes have PSNR differences only to a small degree. Visually, the LOT-AGP has the best-reconstructed images among these three at all bit rates. In addition, the coding results of two other state-of-art progressive image codecs are cited for further comparison. One is the Set Partitioning in Hierarchical Trees (SPIHT) algorithm[5] with a dyadic wavelet transform, and the other is Tran and Nguyen's method with the generalized LOT transform[6]. The AGP coding and the adaptive Huffman entropy coding of LOT-AGP are less complex and the memory usage is smaller than in these two progressive codecs. Comparing these three codecs, i.e. the LOT-AGP and the two progressive codecs in PSNR small only small differences in PSNR. SPIHT has about 1 dB higher PSNR than the LOT-AGP and Tran and Nguyen's method for the test image Lena. For the test image Barbara, the PSNR of the LOT-AGP is about 0.5 dB higher than that of the SPIHT and 0.5 dB lower than that of Tran and Nguyen's method. This low-complexity and high performance codec may provide a new direction for the implementation of image compression.

Keywords: image compression, lapped orthogonal transform (LOT), low complexity, entropy coding

1. INTRODUCTION

Very crucial to many applications, such as sharing image and video on the INTERNET, are techniques for image compression with acceptable visual quality for decoded images. There has always been intense interest in development of efficient image compression algorithms. A lot of work in image compression has focused on transform coding.

The transform coders are designed to remove the redundancy in images for purposes of bit rate reduction, based upon signal processing and information theory. The JPEG standard is one of the most widely know standards for lossy image compression. The approach recommended by the JPEG [*] is a transform coding approach using the Discrete Cosine Transform (DCT) of 8×8 sub-blocks[4]. Nevertheless, for the DCT, the blocks do not fit the boundaries of the real objects in the image scene, leading to visually annoying blocking artifacts, especially for compression at low bit rates.

[1] Currently with Motorola, Inc., Schaumburg, IL 60196, ; e-mail: xzou@eurpd.csg.mot.com.
[2]E-mail: pearlw@rpi.edu.

A new class of transformation called the Lapped Orthogonal Transform (LOT) mitigates blocking artifacts and shows greater coding efficiency than the DCT[3,7,8]. A fast computable approximation to the LOT has been designed to save the computation cost[7,8]. Therefore, the LOT is a good candidate when picking a transformation for image compression.

A major concern for the implementation of image compression is the algorithmic complexity, which has prevented realizing truly high performance codec in image compression. However, there has been promising progress in achieving high compression performance with low complexity. The image codec, called amplitude and group partitioning (AGP), sits at the opposite end of trade off between complexity and compression[1,2].

AGP was used by Said and Pearlman [1] to encode wavelet and block DCT of images. The DCT coding by AGP (DCT-AGP), despite its high coding efficiency, produces reconstructed images that show blocking artifacts at low bit rates. The purpose of this paper is to alleviate blocking effects with a new LOT coding by AGP (LOT-AGP). The LOT-AGP has high compression quality and reduced blocking artifacts with a small increase in computational cost due to the LOT.

2. LOT

2.1. Introduction

In transform coding systems, the input signal is typically divided into blocks, which are then subjected to an energy-preserving unitary transformation. The aim of the transformation is to convert statistically dependent pixels into a set of essentially independent transform coefficients, preferably packing most of the signal energy into a minimum number of coefficients, preferably packing most of the signal energy into a minimum number of coefficients. The resulting transform coefficients are quantized, coded, and transmitted. At the receiver, the signal is recovered by computing the inverse transformation after decoding and dequantizing the transmitted data.

The DCT transform coding is widely used such as in the JPEG standard. However, the basis functions of the DCT have abrupt changes at the endpoints of their supports, which cause one of the main problems of the DCT, blocking effects especially at low bit rates. These effects are perceived in reconstructed images as visible discontinuities, or artifacts, at the cross-block boundaries. In order to avoid this, we should choose basis functions without abrupt changes.

Some approaches have been introduced to reduce blocking effects, such as overlapping and filtering[10,11]. However, the overlapping method increases bit rates for coding, and the filtering method blurs images at the cross block regions. A more successful method is a lapped transform for block signal coding[3,7,8].

2.2. LOT

2.2.1. Introduction

Historically, the Lapped Orthogonal Transform (LOT) denotes a lapped transform whose input length is equal to twice its output length (L = 2M). Previous research showed the LOT is more appropriate for image coding, comparing with other lapped transforms, because the LOT achieves higher compression efficiency for images and has less computational cost[8].

2.2.2. Design of the LOT

In order to make the design of LOT robust, we can begin with a family of valid LOTs. Then, the optimal LOT within this family or subspace can be found. Our first valid LOT matrix[3] is P_0:

$$P_0 = \frac{1}{2}\begin{bmatrix} D_e - D_0 & D_e - D_0 \\ J(D_e - D_0) & -J(D_e - D_0) \end{bmatrix} \tag{1}$$

where D_e and D_o are the $M \times M/2$ matrices containing the even and odd DCT basis functions of length M, respectively. This particular choice leads to the fast computation of the LOT.

If Z is an orthogonal matrix of order M, we can get a family of LOTs like the following:

$$P = P_0 Z \tag{2}$$

The covariance matrix of the LOT coefficients for a given signal covariance matrix R_{xx} is $R_0 = P_0^T R_{xx} P_0$. If Z is the matrix whose columns are the eigenvectors of R_0, P should minimize the required bit rate for a given distortion level. The signal model chosen in this work is a first order Markov model with $\rho = 0.95$. The quasi-optimal LOT is calculated for M =8 and L = 2M. The basis functions of the LOT decay toward zero at the boundaries. Thus the discontinuity from zero to the boundary value is much lower than that of the standard DCT functions, so that the blocking effects will be reduced.

We are more interested in a fast algorithm of the LOT, which is independent of the signal model. The matrix P_0 can be implemented by a manipulation of the DCT transform and the orthogonal factor Z can be approximated by[3]:

$$Z \approx \begin{pmatrix} I & 0 \\ 0 & \tilde{Z} \end{pmatrix} \tag{3}$$

where $\tilde{Z}$ is an orthogonal matrix of order $M/2$. For $M \leq 16$ case, $\tilde{Z}$ can be further approximated as a cascade of $M/2-1$ plane rotations, in the form of $\tilde{Z} = T_1 T_2 \cdots T_{M/2-1}$. Each plane rotation is defined as:

$$T_i = \begin{bmatrix} I & 0 & 0 \\ 0 & Y(\boldsymbol{q}_i) & 0 \\ 0 & 0 & I \end{bmatrix} \tag{4}$$

The first identity factor in the above equation is of order $i-1$, i.e., the top left element of $Y(\boldsymbol{q}_i)$ is in the position $(i-1, i-1)$ of the $M/2 \times M/2$ matrix T_i. The matrix $Y(\boldsymbol{q}_i)$ is a 2×2 butterfly:

$$Y(\boldsymbol{q}_i) = \begin{pmatrix} \cos\boldsymbol{q}_i & \sin\boldsymbol{q}_i \\ -\sin\boldsymbol{q}_i & \cos\boldsymbol{q}_i \end{pmatrix} \tag{5}$$

where $\boldsymbol{q}_i$ is the rotation angle.

Using the fast LOT implementation, we can approximate the basis functions of the quasi-optimal LOT in Figure 3. The angles that optimize the LOT for the maximum coding gain are[8]:

$$[\boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3] = [0.13\boldsymbol{p}, 0.16\boldsymbol{p}, 0.13\boldsymbol{p}] \tag{6}$$

There are choices of other combinations of angles for different optimization criteria. The angles in equation (6) are used in this work.

2.2.3. The LOT Filter Bank

The transform coding can be regarded as a special case of multirate filter bank coding, so the basis functions of some transform can be viewed as the impulse responses of a bank of filters. The basis functions of the LOT are good bandpass filters, which are centered at $(k+1/2)\boldsymbol{p}/M$, with a bandwidth of $\boldsymbol{p}/M$, $k = 0,1,\ldots,M-1$. In the stop band, the DCT has an alias level, i.e. the maximum stop band gain for all the filters, of -10 dB. The fast LOT has an alias level of around -17 dB, an improvement of 7 dB. The alias will not be canceled perfectly, unless the subband signal is transmitted without distortion, which can hold only for lossless coding. In low-bit-rate compression, the distortion is most likely high, so aliasing exists. The lower alias level of the LOT reduces the edge effects at the block boundaries, i.e. the blocking effects in an image coding application.

2.2.4. The LOT of Finite Length Signals

In our previous discussion, we assume the input and output are infinite. In image compression, a two-dimensional LOT would typically be separable, and so we would need to compute the LOT of the rows and the columns of the image. Since each row or column is finite-length sequence, some modification is needed to the original equation.

A simple way to calculate the LOT at boundaries, also used in this work, is obtained by reflecting the data at the segment boundaries. This is equivalent to using half the even DCT basis functions.

2.3. Transform Gain of the LOT and the DCT

An optimal transform should minimize the required bit rate under some given distortion level. This is equivalent to maximizing the "energy compaction" measure[12], or so-called "transform gain",

$$G_{TC} = \frac{\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{s}_i^2}{\left(\prod_{i=1}^{N} \boldsymbol{s}_i^2\right)^{1/N}} \tag{7}$$

where $\boldsymbol{s}_i^2$ is the variance of the *i*th transform coefficient and also the *i*th diagonal entry of the matrix $R_0 = P_0^T R_{XX} P_0$.

The transform gain G_{TC} was compared among the DCT and the LOTs for the block size of eight for a one dimensional signal or 8×8 for a two dimensional signal. Two LOTs were involved in the calculation. One was the quasi-optimal LOT as in the Figure 3; the other was the fast LOT as in Figure 4. First, a signal covariance model is considered, which is the first-order

Markov model for correlation coefficient ρ = 0.95 in the the covaiance matrix in Equation (8) below. Secondly, some test images are used to calculate the transform gain.

$$R_{XX} = \begin{bmatrix} 1 & r & r^2 & \cdots & r^{L-1} \\ r & 1 & r & \cdots & r^{L-2} \\ \vdots & & \ddots & & \vdots \\ r^{L-2} & \cdots & r & 1 & r \\ r^{L-1} & \cdots & r^2 & r & 1 \end{bmatrix} \tag{8}$$

The first order Markov Model, with $r = 0.95$.			
	The quasi-optimal LOT (16×8)	The fast LOT (16×8)	The DCT (8×8)
Transform Gain G_{TC}	8.3886	8.3125	7.6312

Table 1. The transform gain G_{TC} of one-dimensional block transform for first order Markov Model, with ρ = 0.95.

	The quasi-optimal LOT (16×8)	The fast LOT (16×8)	The DCT (8×8)
Lena (256×256)	25.1789	24.4513	21.9906
Lena (512×512)	57.7503	55.6129	49.6647
Barbara (512×512)	24.4118	23.6405	19.2908

Table 2. The transform gain G_{TC} for three testing images.

From these two tables, we would conclude that the LOT has higher transform gain than the DCT. These tables also show that the fast LOT is close to the quasi-optimal LOT based upon first order Markov model. This finding suggests that the fast LOT should be picked for practical application, because of the considerations of computational cost and transform performance.

3. AMPLITUDE AND GROUP PARTITIONING CODING

3.1. Introduction

The whole process of transform coding is divided into several stages, mainly transformation, quantization, and encoding.

3.2. Quantization

Uniform quantization is defined as division of the input amplitude by a quantizer step size Q followed by rounding to the nearest integer, according to $F^Q = \left\lfloor \frac{F}{Q} \right\rfloor$, where F^Q is the index of the quantization interval, or quantization bin number. The reverse process of quantization is dequantization, which is mapping the bin number to the midpoint of the associated quantization interval by $F' = F^Q \times Q + Q/2$. Quantization affords a compressed representation of the original signal, but introduces error or distortion.

The JPEG standard recommends some quantization tables, composed of quantization step sizes for different transform coefficients based upon visual tests. The JPEG default quantization tables are not used here. Instead, the same quantization step is used for each coefficient of the LOT. This approach followed by entropy coding achieves better compression results, especially at high bit rates[13].

3.3. Amplitude Partitioning

After transformation and quantization, the data is concentrated around zero but also distributed widely. If we encode these samples directly with an entropy code, such as Huffman Code, the size of the alphabet is huge. Under some scenarios, the length of codes could exceed the length of a machine word. This "overflow" could bar the design of entropy codebooks.

When more sophisticated techniques, such as encoding several samples together or encoding samples conditionally, are introduced, the large size of alphabet may make the implementation impossible. Efficient approaches to address large alphabets have appeared in several places. For example, such a method is used in the JPEG standard[4].

A short description of this method, alphabet partitioning, also called amplitude partitioning[1], is:

- The source alphabet is divided into a relatively small number of sets.
- Each sample is located by two symbols:
    1. A symbol identifying its set.
    2. A symbol identifying its location within that particular set.

The symbols in items 1 and 2 are called the set number and the set index, respectively. This pair of numbers represents each sample. When encoding this pair, the set number is entropy coded and the set index is coded simply, like the binary representation of that index itself.

This reduction of complexity is associated with some sacrifice of compression ratio. If the probability distribution of symbols inside sets is close to uniform, the loss in compression efficiency, or the increase of rate, is relatively small. This observation is reasonable, because a uniformly distributed source can be optimally coded with fixed length codes. There is an algorithm, whose optimization criterion is reducing complexity and minimizing the increase of rate, to guide the design of partitioning tables[1]. However, the optimal solution depends upon the statistical model of the source. The authors here use a good amplitude partition as in Table 3 for all test images. Each symbol from the source is assigned to a magnitude set. A symbol is represented by a set number, a set index, and a sign bit, if nonzero.

3.4. Group Partitioning

After amplitude partitioning, only the set number of each symbol is encoded with a powerful and more complex entropy code, such as adaptive Huffman code. The alphabet for set numbers has shrunk to a small and manageable size. For instance, using the method suggested in Table 3, this alphabet comprises twenty symbols, i.e. {0, 1, …, 19}. On the other hand, we do pay the penalty in inefficiency for encoding set indexes with fixed length codes. However, with such a small size alphabet, we can further exploit the dependence between neighboring samples, and achieve compression ratios larger than that of the zero order entropy. One technique is to use conditional coding, which is nicely integrated into the technique called the group partitioning.

There are various versions of group partitioning in previous research[1]. They emanate from one general scheme. We illustrate the general scheme with one of those versions, where the original block has $2^n \times 2^n$ pixels. The group partitioning method is to code set numbers of a block in a hierarchical manner.

1. Encode the maximum set number in the $2^n \times 2^n$ block by an entropy code.
2. Divide the original block into four $2^{n-1} \times 2^{n-1}$ sub-blocks. Create a binary mask, where a “1” or “0” signifies whether or not the maximum set number is reached in its respective sub-block. This four-bit binary mask is also entropy-coded. Set n to n-1 and return to step 1 for each sub-block whose maximum set number is non-zero. Continue until 2×2 pixel blocks have been reached.
3. Encoding 2×2 pixels uses the similar approach as $2^n \times 2^n$ pixels, entropy-coding the whole block with a maximum set number and a binary mask indicating the location of the maximum set number.
4. When treating each 2×2 block at pixel-level, entropy-code a pixel’s set number if that set number is smaller than the maximum set number of this 2×2 block, otherwise skip its set number.
5. Encode a pixel’s set index and sign bit with their binary representation.

We have some observations about the process above:

- Whenever the maximum set number of any $2^n \times 2^n$ block is 0, for any n in the process down to n=1, the encoding process is stopped and that block is represented with only one bit of information. This is a big advantage of recursive partitioning from a large block to a smaller block. It can represent a large block of zeros very efficiently, which is a very common scenario for an image source after transformation and quantization.
- If the maximum set number of any block is 1, the encoding process is also very simple. Only set indexes and sign bits of pixels are coded. This kind of distribution of data also happens frequently.
- When the maximum set number of one block is small enough, such as less than three using the alphabet partitioning table in Table 3, the conditional codes are used to encoding the maximum set number of sub-blocks or pixels. Those binary masks are also coded using codes conditioned upon their corresponding maximum set number.

Magnitude Set Number	Magnitude Interval	Sign Bit	Index Length
0	[0]	No	0
1	[1]	Yes	0
2	[2]	Yes	0
3	[3]	Yes	0
4	[4, 5]	Yes	1
5	[6, 7]	Yes	1
6	[8, 11]	Yes	2
7	[12, 15]	Yes	2
8	[16, 23]	Yes	3
9	[24, 31]	Yes	3
10	[32, 47]	Yes	4
11	[48, 63]	Yes	4
12	[64, 127]	Yes	6
13	[128, 255]	Yes	7
14	[256, 511]	Yes	8
15	[512, 1023]	Yes	9
16	[1024, 2047]	Yes	10
17	[2048, 4095]	Yes	11
18	[4096, 8191]	Yes	12
19	[8192, 16383]	Yes	13

Table 3. The amplitude partitioning table used in this work

With these features, the group partitioning is powerful enough to achieve very high compression ratio. The group partitioning also has a lot of flexibility to adjust some important parameters such as the extent of conditional coding. The recursive partitioning of $2^n \times 2^n$ block into four $2^{n-1} \times 2^{n-1}$ sub-blocks is only one example of group partitioning. Other hierarchical recursive partitioning schemes are indeed possible. 2×2 blocks can be encoded as two aggregated symbols of 2, 3 or 4 together if the maximum set number is small.

General Scheme of Combination of Amplitude Partitioning and Group Partitioning

The encoding scheme is [1]:

1. Order the source symbols according to their probabilities, with "0" representing the most common symbol, "1" the second most common, etc.
2. Partition the source samples into a certain number of sets.
3. Create a list with the initial sets, find the maximum sample-value v_m, i.e. the maximum set index, inside those sets and entropy-code those numbers.
4. For each set with $v_m > 0$ do
   a) Remove the set from the list and divide it into subsets;
   b) Entropy-code a binary "mask" with n bits (one for each subset) such that a bit is set to 1 if the maximum in that subset, v_{mi} is equal to v_m and otherwise set to 0;
   c) If $v_m > 1$ then entropy-code every $v_{mi} < v_m$, possibly aggregating them to create the proper source extensions;
   d) Add to the list of sets each subset with more than one element and with $v_{mi} > 0$.
5. If the set list is empty, then stop; otherwise, go to step 4.

The binary masks are encoded conditionally upon their corresponding maximum set numbers. When a set's maximum set number is less than 3, conditional codes are also used to encode this set's maximum subset numbers.

The encoding process can be stopped at a very early stage if the maximum set number of some set equals zero. When a maximum set number of a set is one, the corresponding binary mask will represent the maximum set number of its subsets, ones or zeros. The encoding process is thus very efficient to represent large blocks of zero or values close to zero.

A variation of this implementation can introduce more use of conditional codes or code a block of data together. However, this variation will increase the alphabet size and finally put too many burdens on the implementation.

3.5. Overview of Encoder and Decoder

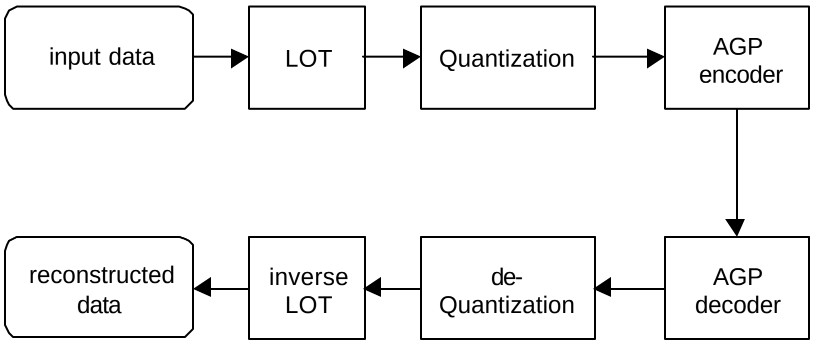


Figure 1. The structure of the encoder and decoder.

Figure 6 is the summary of the overall structure of the encoder and decoder used in this work. All the coefficients of the LOT of the input data are quantized with a uniform quantizer of the same step size *Q*. The bin numbers are then inputted to the AGP encoder. The mean-squared reconstruction error increases and the rate of the coder decreases with increasing *Q*. The entropy code is an adaptive Huffman code, which is relatively simple but very efficient. One advantage of the adaptive Huffman code is that the encoding process does not require learning time. The Huffman encoder encodes data, collects statistical information of input data and adjusts the codebook at the same time. The Huffman decoder reconstructs the codebook in the same way as the encoder. The approach saves the overhead information of transmitting the codebook along with the encoded image data.

4. CODING PERFORMANCE AND CONCLUSION

4.1. Introduction

It is interesting to compare the coding performance of different schemes for image compression. The JPEG standard is one popular example for transform coding[4]. The JPEG coder used below is the PVRG-JPEG CODEC 1.1, which can be downloaded from ftp://havefun.stanford.edu/pub/jpeg/JPEGv1.2.tar.Z. In previous research, the DCT transform coding by AGP (DCT-AGP) is a state of art case for transform coding[7]. The authors implement the LOT transform coding by AGP (LOT-AGP), which is closely related to the DCT-AGP. The block size for both LOT and DCT is 8×8 for comparison.

4.2. Objective Evaluation

In image coding, it is customary to use a normalized error measure, called Peak *SNR*, which is defined as $PSNR = 10\log_{10}\left(\frac{255^2}{s_r^2}\right)$, where s_r^2 is the mean-squared reconstruction error, to evaluate the performance[14]. Three test images were encoded, and decoded by three image codecs, which are the PVRG-JPEG CODEC 1.1, the DCT-AGP, and the LOT-AGP. The *PSNR* of the reconstructed images were recorded as a function of bit rate in bits per pixel (bpp), which is determined from the actual size in bytes from the compressed files. The subject test images are Lena (512×512), Barbara (512×512), and Gold-Hill (512×512).

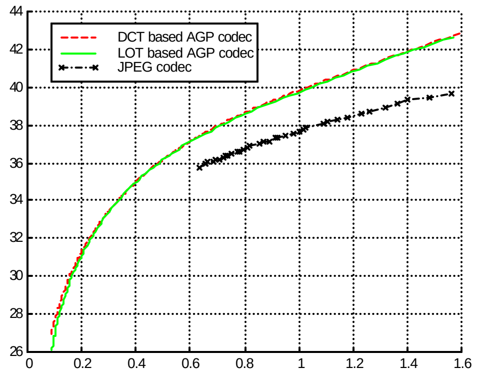


Figure 2. The plot of *PSNR* in dB vs. bits per pixel using the image Lena(512×512).

Figure 2 suggests that the performance of the LOT-AGP is close to that of the DCT-AGP. When these two compress the test image, Lena, into the same file size, the DCT-AGP has slightly larger *PSNR* than the LOT-AGP. However, these two codecs are much better than JPEG codec on compression efficiency, from about 1.7dB to 3 dB depending on the bit rate. The test image, Barbara, gives different results than the first test image, Lena, because Barbara has more high frequency components than Lena. The analogous result for Goldhill conveys the same information as Figure 7 because of the similarity between this test image Goldhill and Lena.

From these three plots, we can conclude: The LOT-AGP and DCT-AGP are much more efficient in compression than the JPEG standard. For some images, the DCT-AGP has slightly higher *PSNR* than the LOT-AGP. On the other hand, for images which have a lot of high frequency information, the LOT-AGP is better than the DCT-AGP.

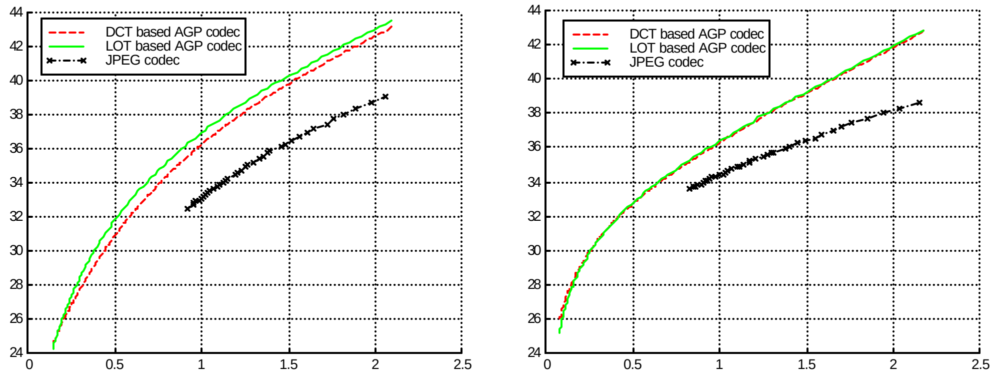


Figure 3. Left: the plot of *PSNR* in dB vs. bits per pixel using the image Barbara (512×512); Right: The Plot of *PSNR* in dB vs. bits per pixel using the image Goldhill (512×512).

Exact numerical data about the *PSNR* difference are in Table 4. The author omitted the JPEG codec in this table because its compression results are inferior to the two AGP codecs. For further comparison, the coding results of two other progressive image coders are cited below, which are the best progressive wavelet coder SPIHT[14], and a LOT, having length 8, based progressive image coder[6].

Image	Rate (bits per pixel)	*PSNR* in dB			
		LOT-AGP	DCT-AGP	SPIHT arith. codec[14]	LOT-8 prog. codec[6]
Lena (512×512)	0.10	26.91	27.76	-	-
	0.25	32.41	32.52	34.11	33.57
	0.50	36.13	36.21	37.21	36.75
	0.75	38.20	38.28	39.04	-
	1.00	39.69	39.84	40.41	40.09
Barbara(512×512)	0.15	24.62	24.67	-	-
	0.25	27.31	26.80	27.58	28.80
	0.50	31.76	30.86	31.40	32.70
	0.75	34.66	33.88	34.26	-
	1.00	36.91	36.24	36.41	37.43
Goldhill(512×512)	0.10	26.33	26.82	-	-
	0.25	29.80	29.89	30.56	-
	0.50	32.78	32.71	33.13	-
	0.75	34.70	34.65	34.95	-
	1.00	36.37	36.29	36.55	

Table 4. *PSNR* in dB for various image coding algorithms

For the test image Lena (512×512), the PSNR of the DCT-AGP is less than 0.2 dB higher than that of the LOT-AGP at most bit rates. The LOT-AGP has about 0.8 dB higher PSNR than the DCT-AGP using the test image Barbara (512×512). There is roughly 0.1dB gain in PSNR for LOT-AGP over DCT-AGP for the test image Goldhill (512×512).

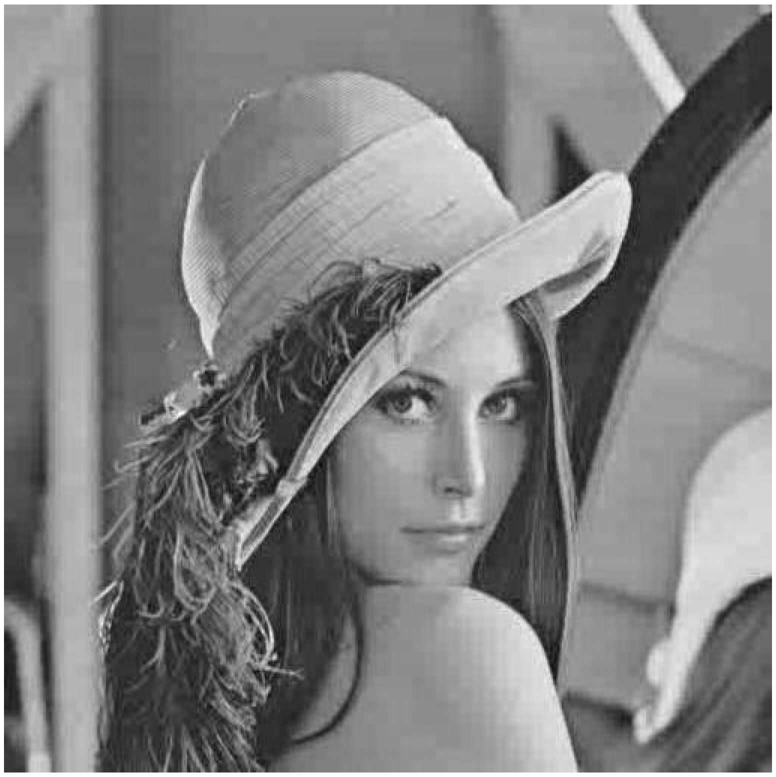
The two progressive codecs are more complex that the LOT-AGP. However, the LOT-AGP delivers close *PSNR* results to the two progressive codecs. For the test image Lena, the SPIHT codec offers the highest *PSNR* at the same bit rate. For the test image Barbara, the LOT-AGP codec outperforms SPIHT at several bit rates. Generally, the LOT-AGP codec has comparable compression performance to the two progressive codecs.

4.3. Subjective Evaluation

PSNR is one of the objective metrics for evaluating a reconstructed image. However, it does not measure the extent of the blocking artifacts in images. The DCT-AGP and LOT-AGP reconstruct test images at different bit rates for viewing on a high-resolution monitor. Some are printed in the following to illustrate the visual results.

Blocking effects are more obvious at low bit rates, such as 0.10 bpp, 0.15 bpp and 0.25 bpp. Images reconstructed by the DCT-AGP at those low bit rates are totally messed up by blocks. However, the LOT-AGP keeps much more detail of original images at these low bit rates, like the hair of Lena, the pattern on the cloth and table cloth in Barbara, and background buildings and the street stones in Goldhill. At 0.50 bpp, blocks are still noticeable in some regions of the images from the DCT-AGP. The LOT-AGP gives almost no blocking artifacts at this rate, 0.50 bpp. The images, compressed at 0.5 bpp, were zoomed in to the ratio of 1:2 or 1:3 for further observation. Blocking artifacts become much more prominent for the DCT-AGP than the LOT-AGP.

For the bit rate 0.75 bpp, it is surprising to spot blocks in some regions of the images from the DCT-AGP. Most of these blocks are recognized in the background of the images. The LOT-AGP produces high visual quality images at the same rate. After images zoomed in to the ratio of 1:2 or 1:3, we have the similar observation as the bit rate 0.50 bpp. When the bit rate goes up to 1.00 bpp, the images have almost no noticeable artifacts. Moreover, the same observation holds when images are zoomed in to the ratio of 1:2 or 1:3. Overall, images compressed by the LOT-AGP have better visual quality, i.e. fewer blocking artifacts, than the DCT-AGP. When the bit rate increases, the LOT-AGP finally outperforms the DCT-AGP in *PSNR* comparisons. The LOT-AGP at the same bit rate has more ringing effects than the DCT-AGP.



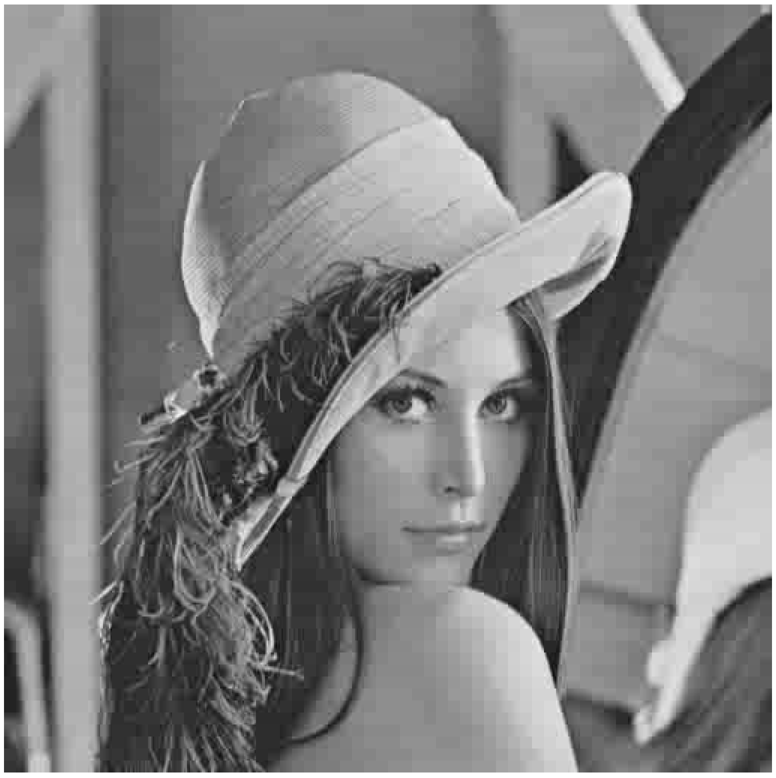


Figure 4. Lena (512×512) compressed at 0.50 bpp. Top: by the LOT-AGP, *PSNR* = 36.13 dB; bottom: by the DCT-AGP, at 0.50 bpp, *PSNR* = 36.21 dB.

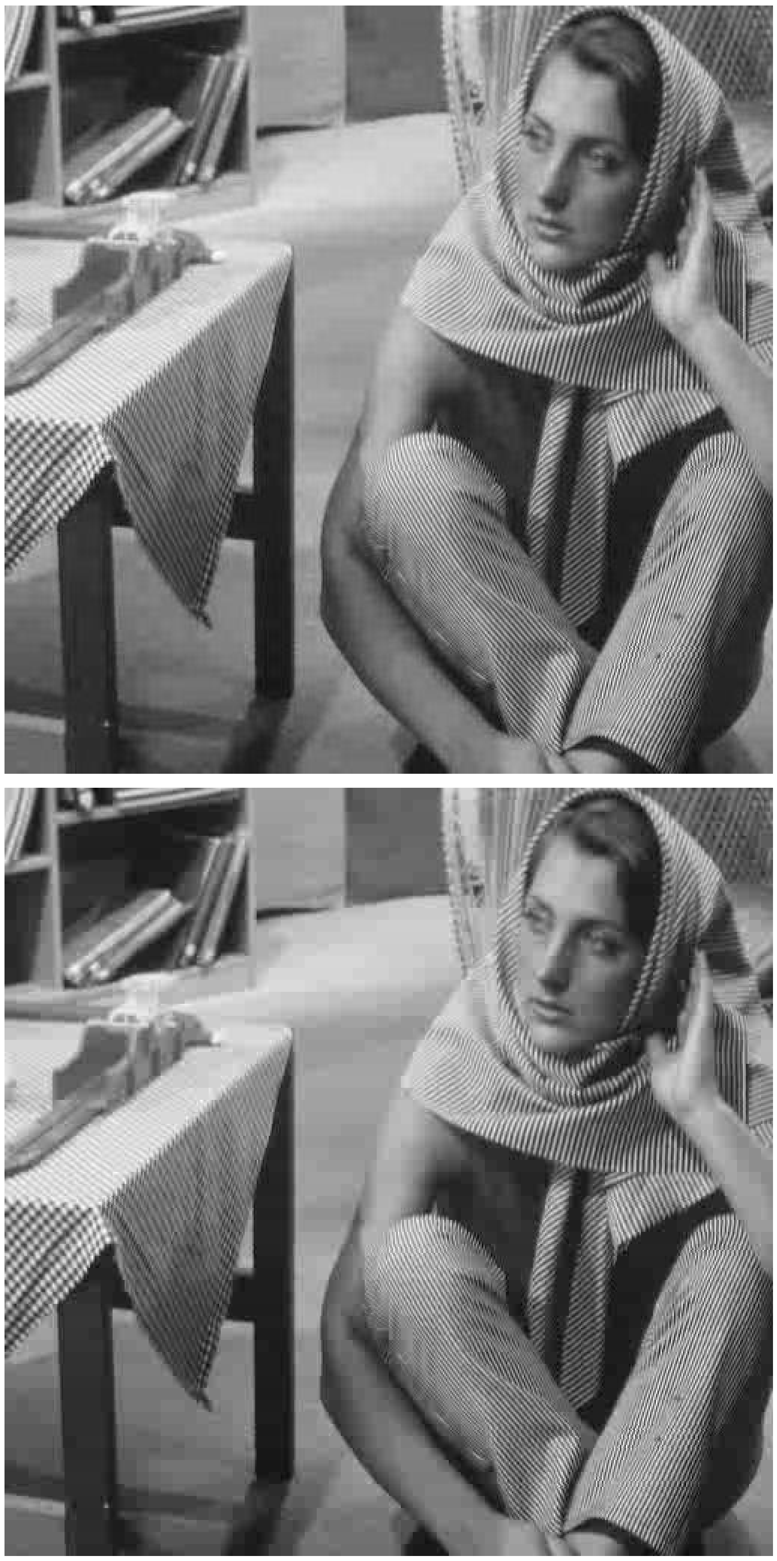


Figure 5. Barbara (512×512) compressed at 0.50 bpp. Top: by the LOT-AGP, *PSNR* = 31.76 dB; bottom: by the DCT-AGP, *PSNR* = 30.86 dB.

4.4. Conclusions

The principles and performance of the LOT-AGP are discussed here. LOT-AGP delivers higher quality images at various bit rates than the traditional DCT transform coding schemes. Extremely low bit rate as 0.10 bpp or 0.15 bpp has little importance for implementations. Compressing images in the range of rates from 0.25 to 0.75 bpp seems to be the ideal place to enjoy the advantage of the LOT-AGP. On the other hand, this scheme only increases computational cost by a small amount over that of the DCT. The LOT-AGP meets our expectation for a low-complexity codec.

ACKNOWLEDGEMENTS

The authors wish to thank Dr. Ricardo L. de Queiroz for providing his LOT software. The authors are thankful to Dr. Amir Said for his suggestions and comments on the amplitude and group partitioning coding scheme.

5. REFERENCES

1. A. Said and W. A. Pearlman. “Low-Complexity Waveform Coding via Alphabet and Sample-Set Partitioning,” in *Visual Communications and Image Processing '97,* J. Biemond and E. J. Delp, eds., *Proc. SPIE* 3024, pp. 25 – 37.
2. W. A. Pearlman. “High Performance, Low Complexity Image Compression,” *Applications of Digital Image Processing X, Proc. SPIE 3164,* July 1997, pp. 234 – 246.
3. H. S. Malvar and D. H. Staelin. “The LOT: Transform coding without blocking effects,” *IEEE Trans. Acoustics, Speech, Signal Processing,* vol. 37, April 1989, pp. 553 – 559.
4. G. K. Wallace. “The JPEG Still Picture Compression Standard,” *Communications of the ACM,* vol. 34, April 1991, pp. 31 – 44.
5. A. Said and W. A. Pearlman. “A New Fast and Efficient Image codec Based on Set Partitioning in Hierarchical Trees,” *IEEE Transactions on circuits and Systems for Video Tech.*, June 1996, pp. 243 – 250.
6. T. D. Tran and T. Q. Nguyen, “A progressive transmission image coder using linear phase paraunitary filter banks,” *Proc. 31st IEEE Asilomar Conference on Signals, Systems, and Computers,* Pacific Grove, CA, Nov. 1997.
7. H. S. Malvar and D. H. Staelin. “Reduction of blocking effects in image coding with a lapped orthogonal transform,” in *Proc. ICASSP 88,* NY, April 1988, pp. 781 – 784.
8. H. S. Malvar. *Signal Processing With Lapped Transforms.* Norwood, MA: Artech House, 1992.
9. A. B. Watson. “Image Compression Using the Discrete Cosine Transform,” *Mathematica Journal,* 4(1), 1994, pp. 81 – 88.
10. H. C. Reeve, III, and J. S. Lim. “Reduction of blocking effect in image coding,” in *Proc. ICASSP 83,* Boston, MA, pp. 1212 – 1215.
11. D. E. Pearson and M. W. Whybray. “Transform coding of images using interleaved blocks,” *IEE Proc.,* Part F, vol. 131, August 1984, pp. 466 – 472.
12. N. S. Jayant and Peter Noll. *Digital Coding of Waveforms,* PRENTICE-HALL, INC. Englewood Cliffs, NJ, 1984.
13. Stéphane Mallat. *A Wavelet Tour of Signal Processing.* ACADEMIC PRESS, San Diego, CA, 1998.
14. A. Said and W. A. Pearlman. “A New Fast and Efficient Image codec Based on Set Partitioning in Hierarchical Trees,” *IEEE Transactions on circuits and Systems for Video Tech.*, June 1996, pp. 243 – 250.