Rensselaer

CNGV

Set Partitioning Waveform Coding

©William A. Pearlman
Department of Electrical, Computer
and Systems Engineering
Rensselaer Polytechnic Institute
Troy, NY
pearlw@rpi.edu

Overview

1 - Motivation

2 - Alphabet-partitioning preprocessing

3 - Sample-set or group partitioning entropy coding

4 - Numerical Results

5 - Conclusions

Rensselaer

CNGV

Motivation

- Practical problems
  - Waveform data sources are not stationary
  - Linear methods (prediction, unitary transforms, etc.) do not exploit all dependencies
  - Large data alphabets

- Solutions
  - Block-based coding
  - Conditioned or extended adaptive entropy coding
  - Alphabet partitioning
  - Combine all above

The Alphabet Partitioning Scheme

Strategy to reduce coding complexity:

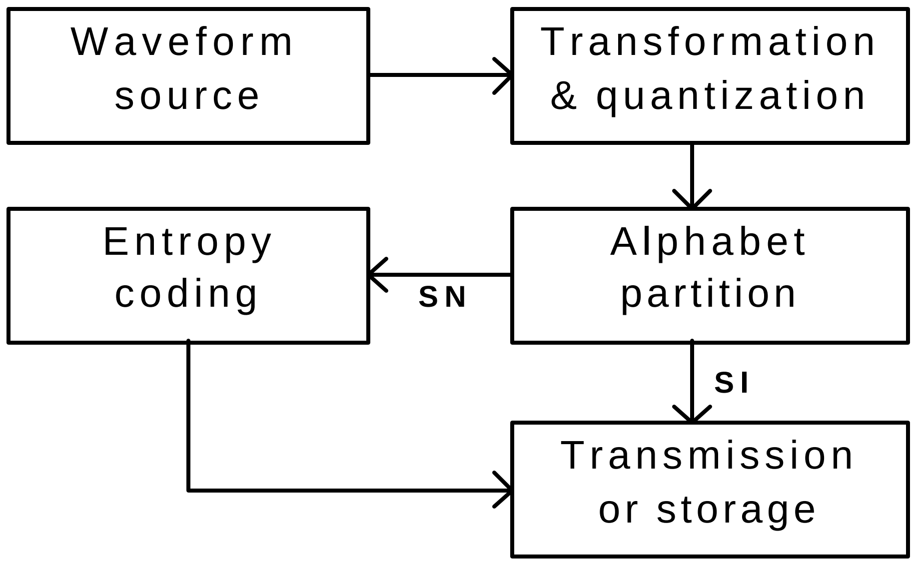
the source alphabet is partitioned in a relatively small number of sets

- each data symbol is coded in two steps:
  1. code the number of the set to which the symbol belongs: the set number (SN)
  2. code the number of that symbol inside the set: the set index (SI)
- when coding the pair (SN, SI) use a powerful method for SN and a simpler and faster method for SI

Rensselaer

CNGV

Processing of Partitioned Data



Amplitude Partitioning Table

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

Entropy Analysis

- Loss due to uncoded set indexes in alphabet partitioning

$$\Delta H = \sum_n \sum_i p_i \log_2 \left(\frac{p_i M_n}{P_n} \right)$$

n = set number, i = set index, M_n = no. of elements,

$$P_n = \sum_{i=1}^{M_n} p_i$$

- Conditions for small loss
  - symbols inside a set have very small probability
  - the distribution inside a set is approximately uniform

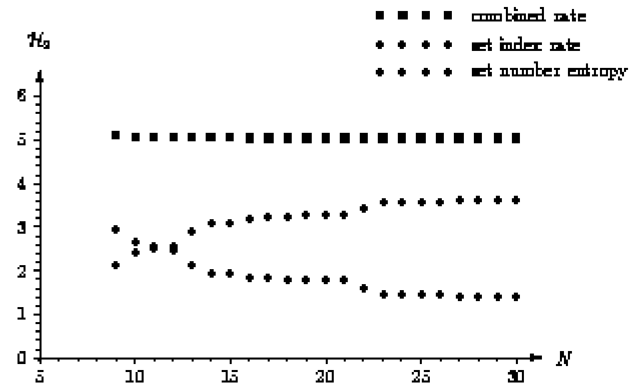


Figure 2: Variation of the contribution of the entropy of the set numbers plus the rate of the set indexes with the number of set N (data source: transformed 12bpp medical image).

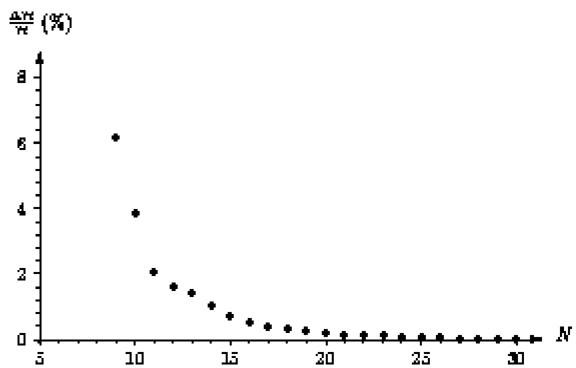


Figure 3: Relative loss in compression due to the reduction of the number of sets N.

	direct method		13 sets			23 sets		
	values	entropy	set number entropy	set index rate	total	set number entropy	set index rate	total
image 1	$\approx 2^{12}$	3.59	2.29	1.35	3.63	2.60	1.00	3.60
image 2	$\approx 2^{15}$	5.94	2.85	3.16	6.01	3.56	2.42	5.98

Table 2: Example of the rates (bpp) obtained with the alphabet partitioning method for lossless compression of medical images.

Group Partitioning Schemes for Encoding Set Numbers

- Example 1: Groups of 4×4 pixels
  - binary mask optional
- Example 2: Groups of 2×2 pixels
  - must use binary masks
- Example 3: Groups of $2^n\times2^n$ pixels
  - same as example 2, using recursive subdivision
- Example 4: Spatial-orientation trees
  - similar to example 3, using alternative subdivision

Groups of 4 x 4

1. Find maximum value v_m (set number) in group.
2. Entropy-code v_m; if $v_m = 0$ stop.
3. If v_m is small use an r-order source extension to entropy-code the $16/r$ groups of r pixels using a $(v_m + 1)^r$ symbol alphabet; otherwise entropy-code the 16 pixel values each with $(v_m + 1)$ symbol alphabet.
4. Stop.

Rensselaer

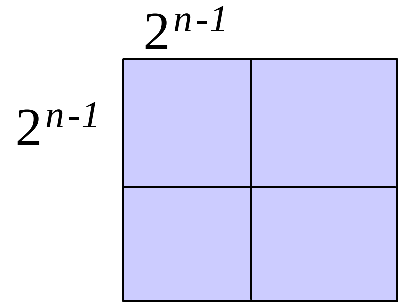
Groups of 2×2 pixels

0	1
1	0

Binary Mask

- Find the maximum set number v_m
- Entropy code v_m , if $v_m = 0$ stop
- Create a binary mask with 4 bits, each bit is 1 if the corresponding pixel has the maximum set number, otherwise 0
- Entropy code the binary mask with 15-symbol alphabet (all-0 never occurs). If mask all 1's or $v_m = 1$ stop.
- Let r be no. of 0's in mask (values < v_m). If $v_m{}^r$ small enough, aggregate symbols and entropy-encode with $v_m{}^r$ size alphabet. Otherwise entropy encode each with v_m size alphabet

Group of $2^n \times 2^n$



1. Entropy code the maximum set number v_m of the $2^n \times 2^n$ pixel
2. Divide each $2^n \times 2^n$ group into four $2^{n-1} \times 2^{n-1}$ pixels
3. Define a 4-bit mask, each bit indicating whether a subgroup has maximum set number v_{mi} equal or less than v_m
4. Entropy code the mask
5. n $\leftarrow$ (n-1) and return to step 1 for each subgroup i, till n is 2.
6. Encode 2 x 2 groups as before

Rensselaer

Group Partitioning Entropy Coding

CNGV

1 Sort data symbols (values) (SN) according with probability (0 = most probable)

2 Divide samples in sets (by image region, time range, etc.)

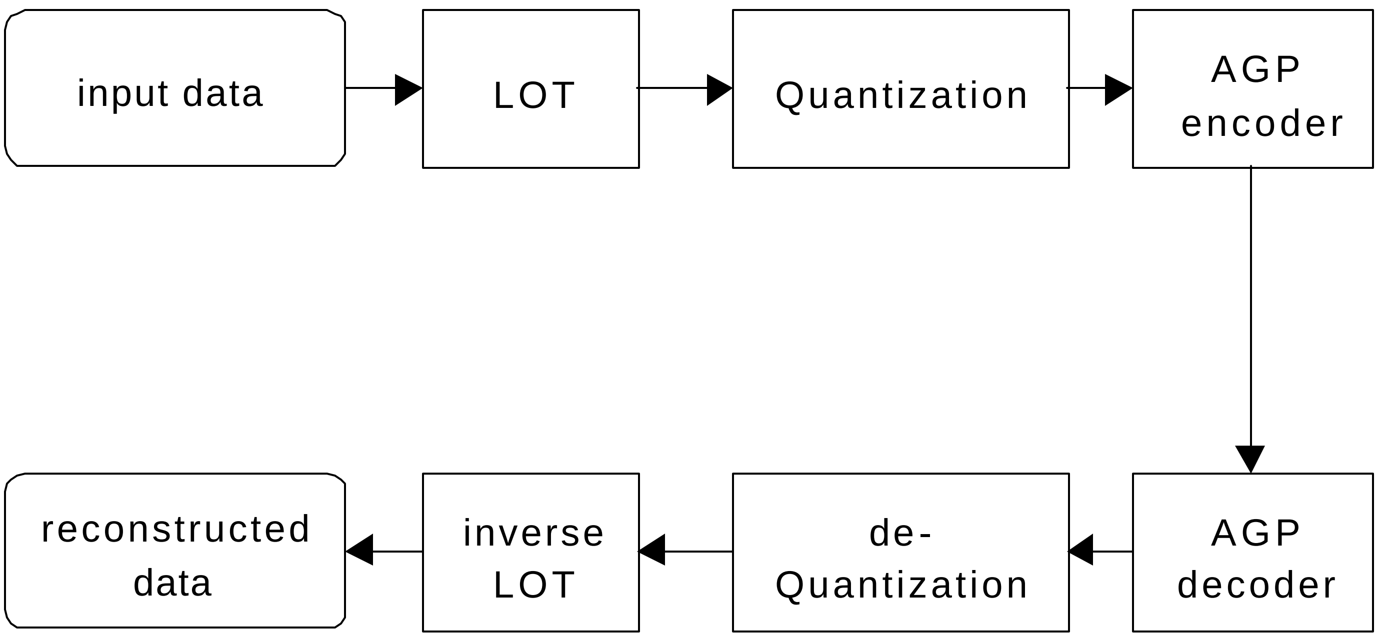
3 Create a list of initial sets, find and entropy-code the maximum value v_m in each set

4 Remove next set from list

- if $v_m = 0$ go to 5
- divide group in n subsets
- compute maximum v_{mi} of each subset
- entropy code binary mask indicating if $v_{mi} = v_m$
- if $v_m > 1$ entropy code each $v_{mi} < v_m$ with v_m-symbol alphabet or extension
- add to list each subset with more than one element and $v_{mi} > 0$

5 If set list is empty then stop; otherwise return to 4

Overall structure



Rensselaer

CNGV

Lossless Image Compression Results

Method	Image transform	Sample-set partitioning	Entropy Coding
QT - S+P	S+P pyramid	Example 3: square blocks	Semi-adaptive Huffman
WV - S+P	S+P pyramid	Example 4: spatial tree	Semi-adaptive Huffman
LSS	S+P pyramid	Similar to example 3	Semi-adaptive Huffman

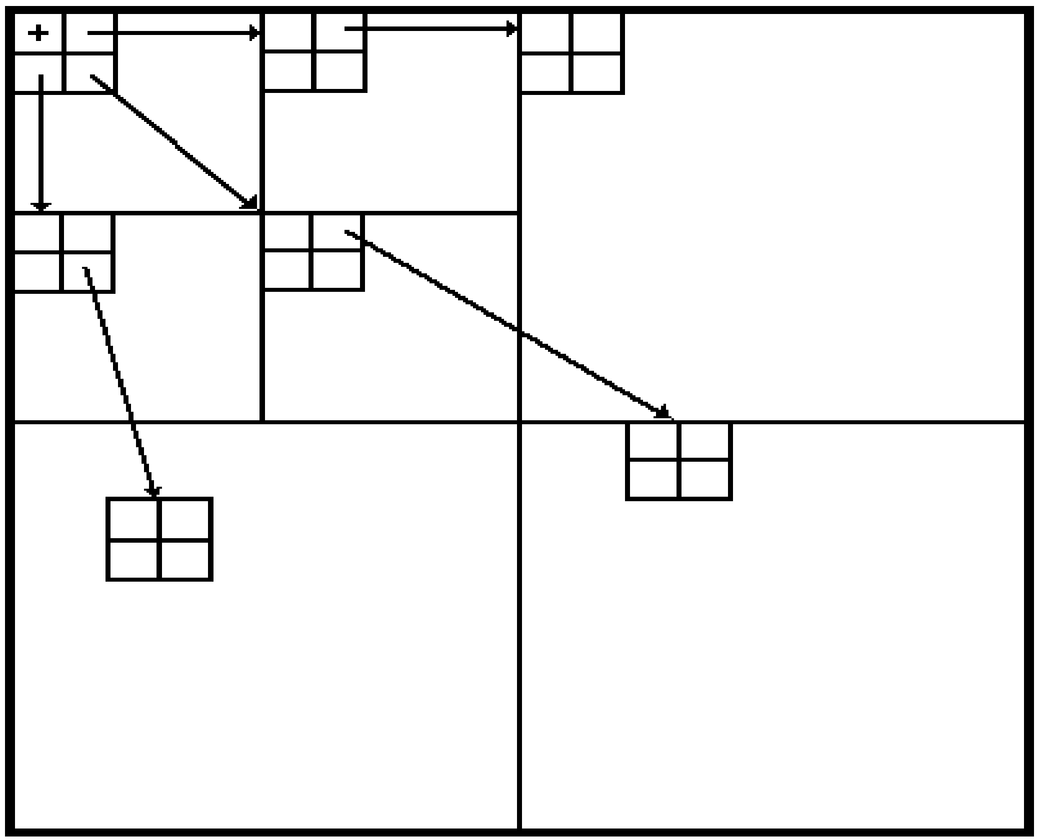
File sizes after lossless compression

Method	Lena	Barbara	Goldhill
QT - S+P	4.27 bpp	4.63 bpp	4.82 bpp
WV - S+P	4.28 bpp	4.65 bpp	4.83 bpp
LSS	4.25 bpp	4.63 bpp	4.81 bpp

Rensselaer

CNGV

Spatial Orientation Trees for Image Pyramids



Rensselaer

CNGV

Methods for Lossy Image Compression

Method	Image Transform	Sample-set Partitioning	Entropy Coding
WV - 9/7	Wavelet 9/7 tap filters	Example 4: spatial tree	Semi-adaptive Huffman
WV - 10/18	Wavelet, 10/18 tap filters	Example 4: spatial tree	Semi-adaptive Huffman
QT - 10/18	Wavelet, 10/18 tap filters	Example 3: square blocks	Semi-adaptive Huffman
DCT - 8	Discrete cosine, 8×8 blocks	Example 4: frequency tree	Semi-adaptive Huffman
DCT - 16	Discrete cosine, 16×16 blocks	Example 4: frequency tree	Semi-adaptive Huffman

Rensselaer

CNGV

Lossy Image Compression Results

Image	Rate (bpp)	DCT 8×8	DCT 16×16	QT 10/18	WV 9/7	WV 10/18	BEST
Lena 512×512	0.25	32.51	33.46	34.16	34.10	34.25	34.35
	0.50	36.18	36.82	37.23	37.21	37.30	37.69
	0.75	38.27	38.70	39.01	39.00	39.04	—
	1.00	39.81	40.13	40.38	40.38	40.45	—
Barbara 512×512	0.25	27.36	28.93	28.73	28.45	28.67	?
	0.50	31.58	32.98	32.87	32.25	32.74	?
	0.75	34.67	35.89	35.86	35.20	35.77	—
	1.00	37.09	38.13	38.16	37.54	38.09	—
Goldhill 512×512	0.25	29.89	30.41	30.50	30.53	30.60	30.71
	0.50	32.69	33.11	33.17	33.13	33.17	33.37
	0.75	34.64	34.98	35.08	34.94	35.13	—
	1.00	36.28	36.56	36.67	36.53	36.67	—

DCT Coding at 0.5 bpp

AGP 8x8 DCT

SPIHT 8x8 DCT

XV JPEG at 0.494 bpp

AGP Coding Reconstructions

Original 8 bpp

Wavelet AGP 0.5 bpp

Coding Results for 512x512 Lena

Method	Rate	PSNR
AGP 8x8 DCT	0.998	38.62
	0.498	33.43
AGP Wavelet	0.993	40.30
	0.497	37.16
PR-SPIHT 8x8	1.005	39.14
DCT	0.497	35.55

Rensselaer

CNGV

CPU Times for Image Compression

Rate (bpp)	WV - 10/18 enc	WV - 10/18 dec	QT - 10/18 enc	QT - 10/18 dec	DCT - 16 enc	DCT - 16 dec
0.25	0.16	0.03	0.20	0.03	0.22	0.03
0.50	0.21	0.07	0.23	0.05	0.25	0.06
0.75	0.26	0.10	0.25	0.07	0.27	0.07
1.00	0.31	0.13	0.28	0.09	0.30	0.10

Lossy compression

	WV - S+P	QT - S+P	LSS
encoding	0.71	0.50	0.48
decoding	0.42	0.36	0.34

Lossless compression

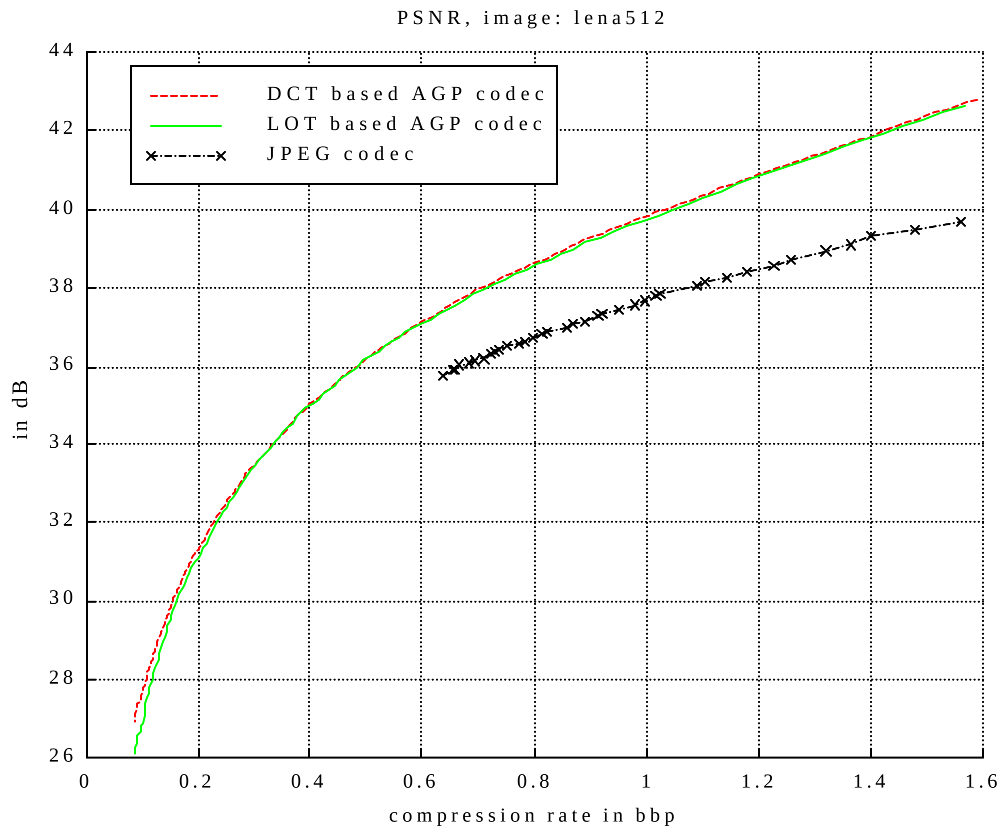
All results in seconds, image Lena 512 ´ 512
133 MHz Pentium, Windows NT

(image transformation times *not* included)

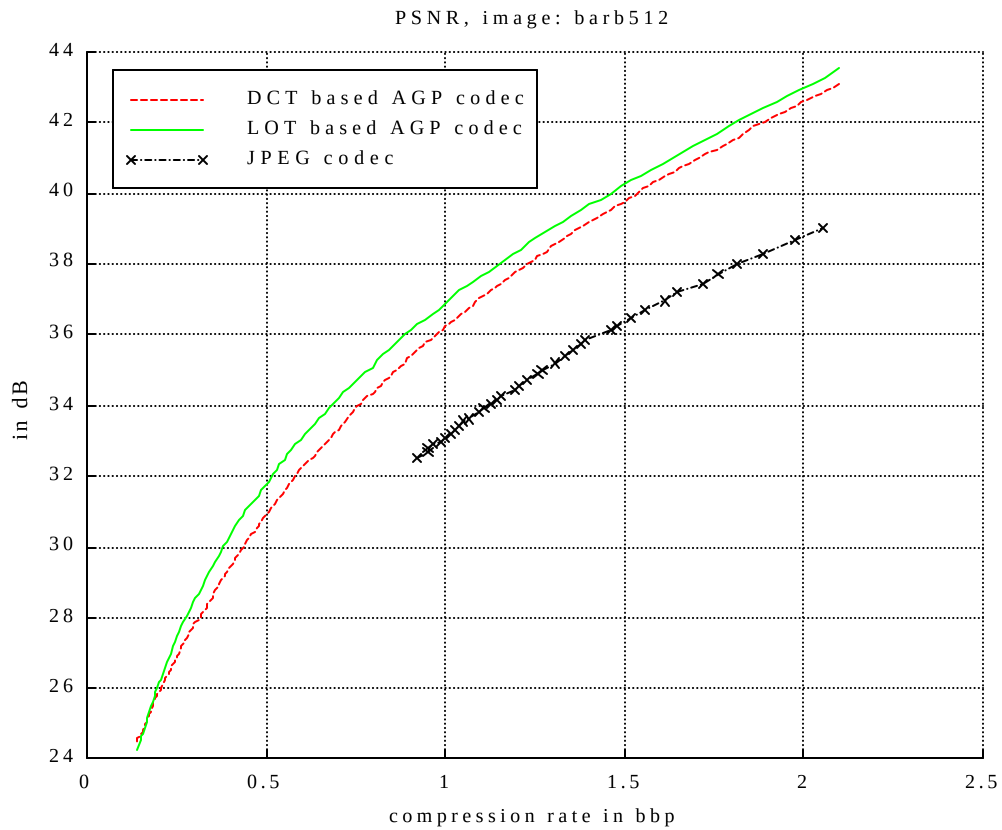
Coding performance

- Objective evaluation
  - Calculating *PSNR* of three different coders on test images
    - The LOT based AGP codec
    - The DCT based AGP codec
    - The JPEG standard codec
- Subjective evaluation
  - Observing visual quality of test images reconstructed at various rates

Numerical Result of Lena



Numerical result of Barbara



Rensselaer

CNGV

Image	Rate (bits per pixel)	*PSNR* in dB			
		LOT-AGP codec	DCT-AGP codec	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	

Lena at 0.25 bpp

LOT-AGP, *PSNR = 32.41*

DCT-AGP, *PSNR = 32.52*

Barbara at 0.25 bpp

LOT-AGP, *PSNR = 27.31*

DCT-AGP, *PSNR = 26.80*

Rensselaer

Conclusions

- Alphabet partitioning allows large reductions in coding complexity, with negligible loss
- Compexity reduction can be exploited by more sophisticated coding schemes
- Group partitioning with adaptive coding yields excellent compression for a wide range of source characteristics
- State-of-the-art results can be obtained with Huffman codes (and DCTs too)

o Ref.: A. Said and W.A. Pearlman, "Low-Complexity Waveform Coding via Alphabet and Sample-Set Partitioning," in Visual Communications and Image Processing '97, Proc. SPIE Vol. 3024, pp.25-37, February 1997.

X. Zou and W. A. Pearlman, "Lapped Orthogonal Transform Coding by Amplitude and Group Partitioning," in Applications of Digital Image Processing XXII, Proc. SPIE Vol. 3808, pp. 293-304, 1999