Efficient Color Image Compression with Category-Based Entropy Coder

Kenneth K. C. Lee* and Y. K. Chan†
*†Department of Computer Science, City University of Hong Kong
*E-mail: kclee@cs.cityu.edu.hk
†E-mail: csykchan@cs.cityu.edu.hk

Abstract— **Images nowadays are generally multi-channeled in nature. Apart from the luminance component, there are extra channels representing information like chrominance and transparency. Despite the fact that single-channel (grayscale) compressors could be applied directly to multi-channel sources, the performance is usually sub-optimal, owing to the distinctive numeric range and statistical distribution of data. If a scheme codes everything indiscriminately, it may end up allocating too many or too few bits to chrominance, resulting in unnecessarily large file size or artifacts like color-bleeding. This paper describes an efficient image compression scheme which works well for both single-channeled and multi-channeled images. The scheme first decorrelates images with *directional* and *intra-image* decorrealtors and the transformed coefficients are compressed. The proposed scheme makes use of a unified, category-based entropy coder, which adapts well to the statistical distribution of coefficients in different regions and channels. Experiment results show that the scheme outperforms the state-of-the-art JPEG2000 compressor by about 18%, together with a more pleasant perceptual quality.**

I. INTRODUCTION

Image compression reduces file size by removing redundancy and correlation within data stream. The efficiency of an image coder is attributable to three factors: the performance of the two-dimensional signal transform, the performance of the entropy coder, as well as how well the entropy coder inherently fits the characteristics of transformed coefficients. In the past decade, we witnessed the evolution of transform and coding technology went hand-in-hand: In the early 90's, most coding schemes (such as JPEG) [1] are DCT-based. Transformed coefficient blocks are processed by run length coding with entropy schemes such as Huffman. Then years later, "hybrid" coders began to come into the picture. One such example is LLEC [2], which is applicable to both DCT and DWT coefficients. LLEC stores coefficients compactly using zerotree followed by variable-length code such as the Golomb-Rice code. Image compression schemes nowadays are dominated by dyadic wavelet transforms, with coefficients coded in a bit-plane-based manner using binary arithmetic coders. Some examples include SPIHT[3], EZW[4] and EBCOT[5]. JPEG2000[6] is the current state-of-the-art.

To achieve even higher compression efficiency, novel decorrelation schemes are proposed to further remove signal redundancy from images. As demonstrated in [7-10], high compression efficiency could be achieved by decorrelating image signal with *directional* or *intra-image* decorrelators. Instead of coding a pixel directly, the intensity of pixel is first predicted using nearby neighbors. The predicted value is subtracted from the actual value and it is the prediction error (residual) that actually get compressed. Despite the current advance in spatial decorrelator, robust entropy coding scheme is also indispensable. In some sense it is even more important. The reason is that luminance information, chrominance information and prediction residual show different statistical properties. Chrominance usually carries less high-frequencies than luminance and prediction residual is more zero-biased than transformed signal. A good coding scheme should adapt well to the corresponding distributions so that the advantage brought about by spatial decorrelators would not be diluted.

This paper describes a robust image compression scheme which works well on multi-channel images over a wide range of image types and resolutions. Instead of coding the image indiscriminately as a whole, pixel blocks are classified into four types: *Smooth Blocks*, *Edge Blocks*, *Pattern Blocks* and *Irregular Blocks*. Blocks of different types are decorrelated respectively, using *directional* or *intra-image* decorrelators. Transformed pixel and residual data are then coded with a unified entropy coder, which adapts well to the distribution of luminance, chrominance and residual information without any tuning. Experiment results show that the scheme outperforms the start-of-the-art JPEG2000 coder by an average of 18%, with better perceptual quality. The remaining sections are organized as follows: Section II briefly reviews the spatial decorrelation steps, particularly the directional decorrelator and the intra-image decorrelator. Section III describes the unified, category-based entropy scheme in detail. Section IV discloses the experiment results and finally the paper is concluded in Section V.

II. A FEATURE ORIENTED CODING APPROACH

As mentioned in Section I, images are usually decorrelated with block-based DCT or dyadic wavelet transform along vertical and horizontal directions. Despite the fact that the transforms work reasonably well in compacting signal energy, problems appear when the transformed coefficients are quantized and coded with a small bitrate budget. For DCT, smooth area of image and the chrominance component (which is generally smooth) may get *"blocky"* upon high compression ratio. As for DWT, energy compaction along

arbitrarily-aligned edges is sub-optimal. When an image is reconstructed with high frequency coefficients removed, obvious noise will be observed along edges. These artifacts cause various degree of disturbance in grayscale images; while in color images, the artifacts show up in the form of *"color-bleeding"*. Since different areas (*edges, flat surfaces*) of image exhibits different properties, it follows that different area should be decorrelated with different approaches. This is the concept of the *feature oriented coding approach* used in the proposed scheme. Input image is first divided into blocks, classified as one of the four types. A block is classified as *Smooth Block* if it lacks high frequency components. Flat image background and chrominance component of objects having uniform color usually fall into this class. A block is classified as *Edge Block* if it shows high frequency feature which aligns to an arbitrary direction. For instance, silhouettes of objects fall into this class. High frequency blocks which exhibit similarities to nearby blocks are classified as *Pattern Blocks*. For instance, fonts and cloth textures are classified as patterns. Finally *Irregular Blocks* are those high-frequency blocks which are not qualified as any type above. Classified blocks are decorrelated in different manners. Irregular Blocks are simply transformed with DCT; while Smooth Blocks are decorrelated with a hybrid transform combining DCT and DWT [9]. The advantage of hybrid transform is that it enables low bitrate without generating blocking artifacts as in block-based DCT, making it ideal for smooth area and chrominance. Edge Blocks and Pattern Blocks, on the other hand, are transformed with either DCT or hybrid transform, whichever gives better rate-distortion performance. However, before transform, the blocks are first decorrelated with *Directional Decorrelator* and *Intra-Image Decorrelator* respectively, as described in the following subsections.

A. Directional Decorrelator

Since pixels in Edge Blocks are correlated along arbitrary direction, it follows that skewed neighboring pixels (along the arbitrary direction) give a good estimation of the current pixel being encoded. The directional decorrelator is conceptually similar to the DALIC scheme [11] and the intra mode of H.264 [12]: Before a pixel block is coded, boundary pixels outside the block are projected along some predefined direction, forming a *prediction block*. The prediction block, which closely resembles the actual pixel block, is subtracted from the actual pixel block and the residual is transformed and then compressed with entropy coder. Unlike DALIC and H.264, however, we enhance the scheme further by incorporating a total of 16 directions, as shown in Fig. 1. We also introduced "dual-directional" decorrelator [7] which contributes to smaller bitrate with fewer artifacts along edges. Fig. 2 shows the luminance channel of the Lena image (left), together with the predicted counterpart (middle) and the corresponding residual (right). It could be observed that residual has a more biased intensity distribution, enable a more compact representation. It is worth mentioning that chrominance directional blocks reuse the prediction direction of its luminance counterpart in order to save bitrate.

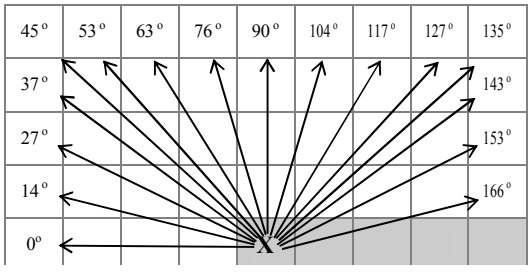

Fig. 1 Sixteen predefined directions used for directional decorrelator. The decorrelator uses only decompressed pixels which are available to the decoder, but not pixels in current block (shaded in grey).

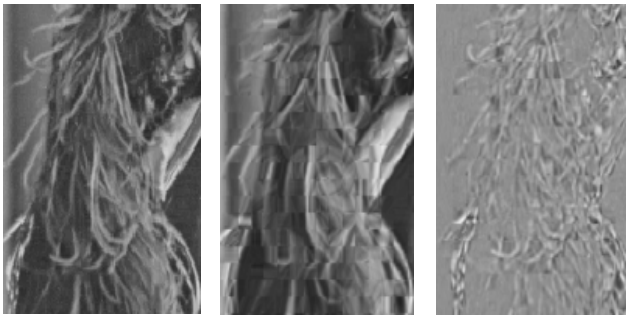

Fig. 2 Original *Lena* image (left), image predicted with dual-directional decorrelator (middle) and the corresponding residual (right). It could be observed that residual has lower dynamic range with biased distribution, hence making it compression-friendly.

B. Intra-Image Decorrelator

Images usually exhibit some degree of localized self-similarity. Although pixel blocks may differs slightly due to unequal orientation and lighting condition, it turns out that coding the differences (residual) between nearby blocks is usually more economical than coding the high-frequency regions directly. As observed in Fig 3, image predicted with intra-image decorrelator (middle) closely resembles the original (left). The residual is dominated by small-magnitude values, marking the slight differences in light intensity as well as pattern disagreement. It is unlikely that we always encounter *perfect match* as in Fig 3, nevertheless, even *partial match* (say, mapping a *'c'* shape to an *'e'* shape) usually, if not always, gives better rate distortion performance than direct DCT transform and entropy coding.

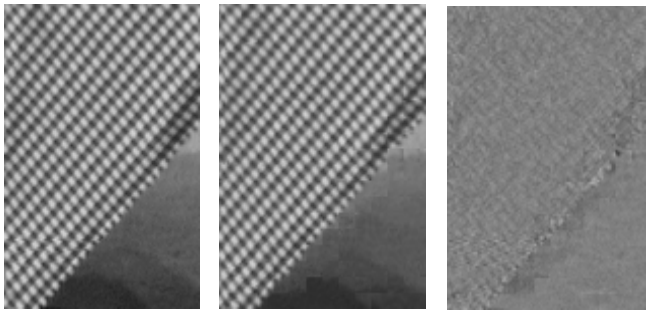

Fig. 3 Original *Barbara* image (left), image predicted with intra-image decorrelator (middle) and the corresponding residual (right). It could be observed that residual is dominated by small magnitudes.

The steps for intra-image decorrelator are similar to the motion estimation step used in video compression: The decorrelator tries to find a pixel block (*predictor*) which closely resembles the to-be-coded block from nearby positions. Once the predictor block is found, the two blocks are subtracted and the residual is then transformed and entropy coded. There are, however, significant differences between intra-image decorrelator and motion estimation. Since the predictor pixels are obtained from the same image, the search range is limited to only the *top* region and the *left* region, where the decoder has already reconstructed. Another difference is the metric used for measuring the *similarity* of blocks. Instead of using Sum of Squared Difference (SSD) as in video compression, a variance-based metric is used:

$$\text{var} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\left(d_{i,j} - \mu\right)^2 \qquad (1)$$

Where N is the dimension of block, $d_{i,j}$ is the intensity difference between actual pixel and predicted pixel in position *(i,j)* and μ is the average magnitude of differences, defined as

$$\mu = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} d_{i,j} \qquad (2)$$

The advantage of using variance-based metric is that it is less likely to reject otherwise similar blocks which differs only in overall intensity. Interested readers may refer to [7] for detail. Similar to directional decorrelator, chrominance pattern blocks reuse the displacement vector of the luminance counterpart.

III. Efficient, Unified Entropy Coder

Despite the fact that image data can be efficiently decorrelated with hybrid transform, directional and intra-image decorrelators; an efficient entropy coder is still needed to translate the advantages into real gain. The entropy coder should adapt well to the property of prediction residual so that the advantage brought about by block classification and decorrelators would not be diluted. This is, however, not an easy task. The challenge comes from the fact that transformed image data and transformed residual exhibit different statistical distributions. Fig. 4 plots the magnitude of transformed coefficients from the *Bike* image, showing magnitudes from 0 to 40. There are two curves in the graph: the solid line shows the distribution of transformed coefficients without using any decorrelator; while the dotted line represents the distribution after directional and intra-image decorrelator are applied. It is obvious that the residual after decorrelation is more zero-biased. Note that the graph is plotted in *log* scale, the frequencies of small-magnitude coefficients are actually *orders* higher. Similar difference is also observed when comparing luminance and chrominance. Chrominance generally carry less high-frequency signal than luminance. Transformed coefficients have a tendency of biasing towards zero. Applying the same entropy coder indiscriminately to different signals (*luminance, chrominance, transformed pixel and transformed coefficients*) could result in suboptimal compression performance since the code lengths fail to reflect the actual occurrence frequencies. It is especially problematic for coders which do not generate dynamic codebook. One may need to predefine several codebooks (*for luminance / chrominance, raw data / residual respectively*) in order to overcome the inefficiency. This section describes a novel, efficient coding scheme which adapts well to different signal sources without tuning. The scheme could be applied to transformed coefficients of almost any dimension. For simplicity, we focus the discussion on 8x8 transformed blocks.

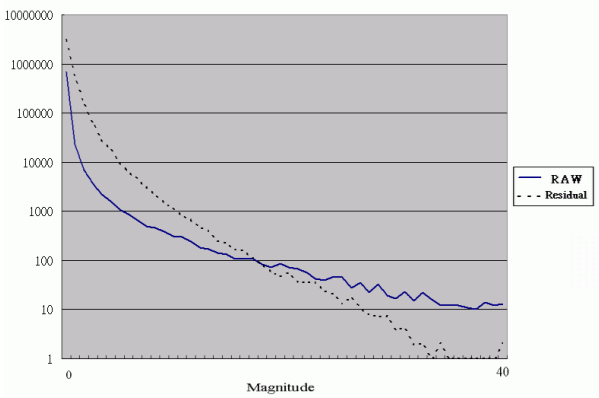

Fig. 4 Distribution of magnitudes of transformed coefficients with and without spatial decorrelation. (*Bike* image).

A. Limitation of Existing Schemes

There are numerous coding schemes which work with 8x8 transformed coefficients. One example is JPEG [1], which combines run-length coding with Huffman. Another example is LLEC [2], which captures zero region using zerotree and codes non-zero coefficients with Golomb-Rice code, in 2x2 units. It is reported that LLEC gives better performance than JPEG. Both schemes, however, doesn't fit well to residual coding. One problem is that too many bits are spent on coefficients of small magnitudes. For instance, LLEC consumes 3 bits for +/-1 and 5 bits for +/-2, which appear far more frequently in residual than in transformed raw image. Another limitation is that the schemes uses *zero-order* entropy and is unable to take advantage of the correlations *between coefficients* and the correlation *between 2x2 blocks* (LLEC). For instance, it is rather common to have a 2x2 block storing coefficients with magnitudes { +/-1 , 0 , 0 , 0 } respectively. LLEC does not aware of this high-order statistics and consumes 3+1+1+1=6 bits for this common pattern. All of these imply that a more efficient coder is needed so that it adapts well to different distributions and acts conservatively for both residual and transformed raw image.

B. Category-Based Approach for Small Coefficients

To take advantage of higher order statistics, we begin by defining *"typical coefficient patterns"* of manageable size. Through the observation of coefficients, it was observed that correlation exists between coefficients within the same 2x2 block. For instance, if the upper-left corner of the block is +/-1, most likely the remaining coefficients bear magnitude

around 0 to 2. By classifying blocks into typical patterns, one could bypass the complexity of a generic high-order entropy coder. This contributes to low memory and low complexity design of entropy coder.

Table I
Classification of 2x2 blocks in proposed coder

Category	Type	Example
0	All Zero	*{0,0,0,0}*
1	Singleton 1 at top row	*{0,1,0,0}*
2	Singleton 1 at bottom row	*{0,0,0,1}*
3	Combination of 0 and 1	*{1,0,1,0}*
4	Combinations of 0,1,2, dominated by small coefficients	*{0,0,1,2}*
5	Combinations of 0,1,2, dominated by large coefficients	*{1,2,2,2}*
6	Combinations of 0,1,2,3, dominated by small coefficients	*{0,0,1,3}*
7	Combinations of 0,1,2,3, dominated by large coefficients	*{1,2,2,3}*
8	Combinations of 0,1,2,3, with multiple 3s	*{2,3,3,3}*
9	Single 4/5 with small coefficients	*{4,1,0,0}*
10	Single 4/5 with big coefficients	*{5,3,2,2}*
11	Double 4/5 with coefficients bearing magnitudes of 0,1,2,3	*{5,4,2,1}*
12	Variable Length Code specialized for large numbers	*{9,7,5,4}*

Classification of 2x2 blocks into pattern categories serves a dual advantage. Not only can it take full advantage of the correlation *within* the 2x2 block, it also enable quick adaptation to different coefficient distributions, making it ideal for both transformed pixel and residual. For example, consider the { +/-1 , 0 , 0 , 0 } pattern mentioned in the last subsection. According to Table I, the block is classified as category 1 *(Singleton 1 at top row)*. Since the category sets limitations on the **count** of "1" (*single "1"*) and its **position** (*first row*), only one bit is needed to identify whether the "1" is on the left or on the right. Also, one more bit is needed to represent the sign of the "1" coefficient. It results in an overall consumption of 2 bits for this frequently occurring block and it is much more conservative than the 6 bits used in the classical LLEC scheme. The real importance of the category-based scheme is that, instead of coding coefficient magnitudes according to their stochastic properties, now we only need to code the position information and the signs, which are random and uncorrelated. *(i.e. we have removed all redundancy from the signal)*

Apart from magnitude correlations *within* a single 2x2 block, correlation also exists *across* blocks. If a 2x2 block is having high magnitudes (e.g. category 12), most likely the blocks next to it also bear high to middle magnitudes and falls into category 8 to 12. It implies that, instead of coding the category ID directly, it is more efficient to store the difference / delta of category ID from that of its neighbor. The upper half of Fig. 5 shows the category IDs of two typical 8x8 blocks, for transformed raw image (left) and transformed residual (right) respectively. *(There are sixteen 2x2 sub-blocks within a 8x8 block: four per horizontal and vertical directions)*

12	9	3	0
10	7	1	0
5	3	0	0
0	0	0	0

5	3	0	0
3	2	0	0
1	0	0	0
0	0	0	0

+2	+2	+2	0
+3	+4	+1	0
+2	+3	0	0
0	0	0	0

+2	+1	0	0
+1	+2	0	0
+1	0	0	0
0	0	0	0

Fig. 5 Category IDs of typical 8x8 blocks: transformed raw image (upper left), transformed residual (upper right). The lower half of figure shows the same blocks as the upper counterpart, with IDs represented as delta

Comparing the upper half and the lower half of Fig. 5, it could be observed that when the IDs are translated to delta *(difference against neighboring blocks on the right or bottom)*, the magnitudes are reduced. Instead of storing large category IDs up to 12, deltas usually bear small values of about 1 to 3. The same applies to both transformed image and transformed residual. It implies that there's no need to setup separate coding methods for raw image and residual respectively. Also, as observed in the upper half of Fig 5, category IDs usually increase monotonically towards the upper left corner (where large magnitude DC and ACs are located). It makes the values of deltas more predictable and allows efficient coding. For instance, one could assign a shorter code length to +1, +2 and assign a longer code length to -4, which is infrequent.

C. Coding of Large Magnitude Coefficients

Large magnitude coefficients are relatively infrequent and it's even scarce in transformed residual. When large magnitude coefficients are encountered, the 2x2 block is classified as category 12 and the coefficients are coded one by one using classical variable-length codes. As suggested in Fig. 4, magnitude of coefficients coarsely follows the Laplacian distribution. Therefore one natural way of coding (as in LLEC) would be storing the coefficients with Golomb-Rice code, which excels in encode / decode speed, without using any explicit code table. The straight-forward (yet working) coding approach is as follows: Numbers are first classified into groups according to the magnitudes. Shorter code prefixes (01, 001, 0001) are assigned to frequent groups and longer code prefixes (00001, 000001…) are assigned to infrequent groups. Each variable-length symbol is generated on-the-fly by appending a single "1" bit to a string of variable "0" bits. The codes generated are equivalent to that of a *skewed* Huffman tree, which is formed whenever Laplacian (exponentially) distributed symbols are encountered.

For the sake of better compression, however, we further massage the scheme mentioned above. Fig 6. plots the magnitude of coefficients of the *Bike* image. Unlike Fig. 4, only the coefficients of category-12 blocks are plotted.

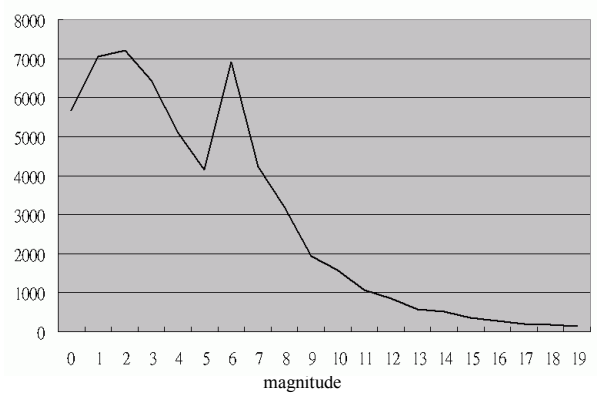

Fig. 6 Distribution of transformed coefficients classified as category 12. Only magnitudes from 0 to 19 are shown (*Bike* image).

As observed in Fig. 6, the counts of coefficients are highest in the small-magnitude region. Coefficient count decreases steadily as magnitude increases from 2 to 5. Then, there is a sudden surge at magnitude 6. Distribution of magnitudes beyond 6 is coarsely Laplacian, the count drops rapidly as the coefficient magnitude increases. The reason for this remarkably interesting distribution is that the majority of coefficients from 0 to 5 are efficiently covered by categories 0 to 11 shown in Table I. The small coefficients here are actually those *coincident cases*, in which small coefficients and large coefficients are packed into the same 2x2 block by chance (e.g. pattern {9, 7, 2, 0}). To tackle this "disturbed" Laplacian distribution, we employ a modified low-complexity, tableless entropy scheme which assigns a shorter code to magnitude 6. Table II shows the first few code prefixes used. It can be seen that magnitude 6 consumes only 3 bits, which is shorter than magnitude class 4..5 and magnitude class 7..8. Code prefix beyond class 7..8 could be derived trivially by prepending '0' bits to the prefix of the last class.

Table II
Variable length code prefixes used for transformed coefficients

Magnitude	Code Prefix
0..3	1xx
4..5	011x
6	010
7..8	001x
9..10	0001x
11..12	00001x
……	……

D. Consideration on Run-length of Zeroes

Apart form small coefficients, magnitude zero itself actually take up considerable space of the transformed signal. The strategy to efficiently represent the huge patch of zeroes, therefore, is the key to high compression performance. JPEG employs the run-length approach, counting zeroes *coefficients* along a zig-zag order. Meanwhile LLEC employs the zerotree approach. In the proposed scheme, we choose to store the run length of 2x2 zero *blocks* along predefined scan order. The remaining question is how the run-length could be represented efficiently. Obviously the run-lengths show statistical distribution, however the exact distribution is affected by different factors like channel type (luminance / chrominance), quantizer and the spatial position within image. For instance, smooth surface tends to have longer run-length while the run length for edge area and pattern area are likely to be shorter. To enable good adaptation to local image features without using two-pass coding, we employed a variant of the simple *move-to-front* coding as follows:

Initialize once: list $\boldsymbol{p} = \{\boldsymbol{i} \mid 0 \le \boldsymbol{i} \le 15\}$

Code (length = $\boldsymbol{s}$*) {*
EntropyCode($\boldsymbol{p_s}$*)*
$\boldsymbol{n} \leftarrow \boldsymbol{r} \cdot \boldsymbol{p_s}$ *, where* $\boldsymbol{r}$ *is constant in range [0,1)*
$\boldsymbol{p_j} \leftarrow \boldsymbol{p_j}+1 \quad , \forall \boldsymbol{p_j} \ge \boldsymbol{n}$
$\boldsymbol{p_s} \leftarrow \boldsymbol{n}$
}

The basic concept is that we maintain a list (array) representing how likely different run-lengths are encountered. Probable values are stored in the front and represented with a shorter variable-length bitcode. Whenever a run-length value $\boldsymbol{s}$ is encountered, it is moved forward by multiplying the current position with a scaling constant $\boldsymbol{r}$. This allows quick adaptation of local image feature as well as channel nature. For instance, when the encoder is encoding a smooth area with long run-length, the encoder state adapts and starting from the next block, long run-lengths could be represented with smaller number of bits. Similar adaptation also occurs when the encoder finished coding luminance and starts with chrominance. When coding chrominance, long run-lengths will be favored automatically and smaller number of bits is required. This simple yet efficient scheme achieves better compression than those with hard-coded tables and it also spare us from using two-pass encoding. From experience, setting $\boldsymbol{r}$ to a value of 0.5 to 0.75 is considered appropriate. Smaller value leads to faster convergence but also makes false prediction / adaptation more probable.

E. Put It All Together

The encoding pipeline could be summarized as follows: First of all, pixel blocks are classified into Smooth, Edge, Pattern and Irregular Regions. Edges and Patterns are preprocessed with directional decorrelator and intra-image decorrelator respectively. For chrominance Edge Blocks and Pattern Blocks, direction and vector information of its luminance counterpart could be reused. After decorrelation, image signal and decorrelated residual are transformed with DCT or hybrid transform. Transformed coefficients are then entropy-coded: For each coefficient block, the run-length of 2x2 zero blocks is first counted and stored. The run-length is also fed back to the system so as to adjust the code length of future blocks. Non-zero coefficients are coded in 2x2 units: Blocks are classified into categories (Table I) according to magnitudes, and the category IDs are translated into deltas for efficient compression. If a 2x2 block is having small magnitude, the block is represented with only position and

sign information. While for large magnitude blocks, each coefficient is stored one by one using variable length code as mentioned in the last subsection. The process repeats until all blocks in all channels are processed.

IV. Experiment Result

To demonstrate the performance of the new coding scheme, it is put to comparison against the latest versions of two of the best state-of-the-art JPEG2000 implementations: Kakadu v6.0 [13] by D. Taubman (who was the principle author of JPEG2000 Verification Model), and the Jasper System v1.900.1 [14] by Michael D. Adams at University of Victoria. Kakadu was utilized by over 100 applications including Apple's Quicktime 6 and Yahoo Messanger; meanwhile Jasper was utilized by products such as Ghostscript and KDE (K Desktop Environment). In order to be comprehensive, test images of various resolutions (from 176x144 to 2048x2560) and different types (from real-life photo to computer graphics and text) are used for testing. Part of the experimental results is listed in Table III.

Experiment results show that the proposed scheme consistently outperforms JPEG2000 (Kakadu & Jasper) by a considerable margin over a wide range of different resolutions and image types. Our scheme performs particularly well for small images and images with high-frequency structures (e.g. edges, patterns) where the performance of dyadic wavelet transform is limited. In terms of PSNR, the proposed system is better than Kakadu JPEG2000 by up to a maximum of about 6 dB. If the overall gain is translated to file size, an average file size reduction of about 18% (and a maximum of 46%) is observed. The subjective perceptual quality of our scheme is also better in the sense that artifacts along edges are less apparent. Fig. 7 shows the compressed CIF "Foreman" image at 0.125bpp, using JPEG2000 and the proposed scheme respectively. It is rather obvious that JPEG2000 blurred the diagonal strips as well as high frequency structures. There are also slight *"color-bleeding"* near the neck and the collar. Fig. 8 shows parts of the "Mobile and Calendar" image (720x576) compressed at 0.25bpp. It could be observed that the proposed scheme works reasonably well for paintings, fonts, as well as real-life objects under natural lighting.

V. Conclusions

This paper describes a low-complexity entropy coding scheme used for image compression. It is shown that sub-block classification could help taking advantage of correlations between coefficients and between neighboring blocks, hence making it ideal for both transformed image data and prediction residual. Instead of coding magnitudes directly, the scheme only needs to code the highly-constrained sign and position information, leading to efficient compression performance without significant increase in memory and computational requirement. The compression efficiency is satisfactory and it outperforms the state-of-the-art JPEG2000 implementations (Kakadu & Jasper) consistently across different image types and resolutions. In terms of visual perception, the proposed scheme is also more preferable since edges and image details are preserved without over-blurring and color-bleeding. Extension on the described scheme is still possible to achieve even higher compression performance. For instance, if computational complexity is not a major concern, entropy coding of coefficient positions within 2x2 blocks may introduce an extra gain of 1-2%.

References

[1] ISO/IEC 10918-1, "Information Technology – Digital Compression and Coding of Continuous-Tone Still Images".

[2] Debin Zhao, Y. K. Chan and Wen Gao, "Low Complexity and Low-Memory Entropy Coder for Image Compression", IEEE Trans. Circuits Syst. Video Technology, vol. 11, no. 10, pp. 1140-1145, 2001.

[3] A. Said and W.A. Pearlman, "New, fast, and efficient image codec based on set partitioning in hierarchical trees", IEEE transactions on circuits and systems for video technology, vol. 6, no. 3, pp. 243-249, 1996.

[4] J. Shapiro, "Embedded image coding using zerotree of wavelet coefficients," IEEE Trans. Signal Processing, vol. 41, no. 12, pp. 3445-3463, 1993.

[5] D. Taubman, "High performance scalable image compression with EBCOT", IEEE Trans. on Image Processing, vol. 9, no. 7, pp. 1158-1170, 2000.

[6] ISO/IEC, 15444-1, "Information Technology - JPEG 2000 image coding system-Part 1: Core coding system".

[7] Kenneth K. C. Lee and Y. K. Chan, "Efficient Still Image Compression with Spatial Decorrelation". The First International Symposium on Information and Computer Elements (ISICE) 2007, pp 205-210.

[8] Kenneth K. C. Lee, Marian Choy and Y. K. Chan, "Benchmarking JPEG2000 with a Novel Hybrid Compression Scheme". International Workshop on Advanced Image Technology 2005, pp. 13-18.

[9] Kenneth K. C. Lee and Y. K. Chan, "Image Compression Techniques and Devices Employing a Hybrid Compression Scheme", HK Short Term Patent Application Number: 05100095.9, 6th January, 2005.

[10] Kenneth K. C. Lee, Marian Choy and Y. K. Chan, "Spatial Domain Contribution to a High Compression Efficiency System", International Workshop on Advanced Image Technology 2006, pp. 89-94.

[11] Debin Zhao, Y. K. Chan and Wen Gao, "Extending DACLIC for near Lossless Compression with Postprocessing of Greyscale Images", In IEEE Proceedings of Data Compression Conference, Snowbird, Utah, USA, April, 1999, pp.563.

[12] "Information technology -- Coding of audio-visual objects -- Part 10: Advanced Video Coding", ISO/IEC 14496-10:2003

[13] Kakadu A Comprehensive Framework for JPEG2000 http://www.kakadusoftware.com/

[14] The JasPer Project Home Page http://www.ece.uvic.ca/~mdadams/jasper/

.

Table III
Comparison of compression performance against Kakadu JPEG2000 and Jasper JPEG2000 over an image set of different types and resolutions. It is observed that the proposed system consistently outperforms JPEG2000 by up to 6 dB

		Kakadu JPEG2000 ver 6.0 [13] (in dB)	**Jasper JPEG2000 ver 1.900.1 [14]** (in dB)	**Proposed scheme** (in dB)	**Bitrate of the best JPEG2000 in order to achieve same PSNR as ours**	**Percentage gain by proposed scheme over the best JPEG2000**
Foreman (176x144)	1.00 bpp	36.1725	35.3740	**38.6086**	1.28 bpp	21.9%
	0.50 bpp	30.6693	30.1041	**34.3397**	0.80 bpp	37.5%
	0.25 bpp	26.9290	26.0512	**30.3268**	0.47 bpp	46.2%
News (176x144)	1.00 bpp	35.6174	35.3430	**37.8145**	1.17 bpp	14.5%
	0.50 bpp	29.2761	28.8176	**32.0267**	0.68 bpp	26.5%
	0.25 bpp	24.7791	24.0075	**27.4015**	0.37 bpp	32.4%
Football (352x240)	1.00 bpp	33.0657	32.6800	**33.7720**	1.12 bpp	10.7%
	0.50 bpp	30.1190	29.7051	**30.5644**	0.57 bpp	12.3%
	0.25 bpp	28.0786	27.5679	**28.4421**	0.29 bpp	13.8%
Salesman (360x288)	1.00 bpp	37.7898	37.0125	**38.4132**	1.08 bpp	7.4%
	0.50 bpp	33.1344	32.5174	**34.1634**	0.59 bpp	15.3%
	0.25 bpp	29.4709	29.0532	**30.4762**	0.30 bpp	16.7%
Barbara (512x512)	1.00 bpp	37.1545	35.7720	**37.9816**	1.12 bpp	10.7%
	0.50 bpp	32.1692	30.8623	**33.4529**	0.61 bpp	18.0%
	0.25 bpp	28.3645	27.2782	**29.3605**	0.31 bpp	19.4%
Lena (512x512)	1.00 bpp	40.4057	39.3065	**40.5429**	1.03 bpp	2.9%
	0.50 bpp	37.3006	36.3159	**37.6775**	0.55 bpp	9.1%
	0.25 bpp	34.1269	33.1328	**34.6636**	0.29 bpp	12.3%
Goldhill (512x512)	1.00 bpp	36.5790	35.8682	**36.8987**	1.05 bpp	4.8%
	0.50 bpp	33.2315	32.7004	**33.4846**	0.53 bpp	5.7%
	0.25 bpp	30.5444	30.0713	**30.6860**	0.27 bpp	5.7%
Mobile & Calendar (720x576)	1.00 bpp	32.9794	32.5202	**33.7272**	1.09 bpp	8.3%
	0.50 bpp	28.0769	27.6265	**28.9441**	0.58 bpp	13.8%
	0.25 bpp	24.8941	24.4828	**25.5796**	0.31 bpp	19.4%
Bike (2048x2560)	1.00 bpp	38.1223	37.3175	**39.2706**	1.17 bpp	14.5%
	0.50 bpp	33.5427	32.9318	**35.3206**	0.68 bpp	26.5%
	0.25 bpp	29.6435	29.0480	**31.7446**	0.38 bpp	33.3%
Café (2048x2560)	1.00 bpp	32.0674	31.5277	**33.0535**	1.12 bpp	10.7%
	0.50 bpp	26.8348	26.3066	**27.8682**	0.59 bpp	15.3%
	0.25 bpp	23.1588	22.6999	**24.0284**	0.30 bpp	16.7%
Woman (2048x2560)	1.00 bpp	38.4510	37.5151	**38.7472**	1.04 bpp	3.8%
	0.50 bpp	33.6518	32.8515	**34.2732**	0.55 bpp	9.1%
	0.25 bpp	30.0102	29.2674	**30.4572**	0.28 bpp	9.1%
CG Wallpaper from Bandai (1024x768)	1.00 bpp	37.2580	37.5643	**41.0557**	1.30 bpp	23.1%
	0.50 bpp	29.9531	29.7963	**33.3657**	0.72 bpp	30.6%
	0.25 bpp	25.2677	24.9357	**28.1746**	0.39 bpp	35.4%
Windows screen capture (800x600)	1.00 bpp	27.0843	27.0390	**33.4456**	1.49 bpp	32.9%
	0.50 bpp	20.5794	20.2714	**23.9335**	0.74 bpp	32.4%
	0.25 bpp	17.2600	17.5840	**18.8183**	0.36 bpp	30.6%
Scanned Document (960x1440)	1.00 bpp	48.0495	48.0748	**48.8263**	1.05 bpp	4.8%
	0.50 bpp	36.7538	36.8932	**40.6655**	0.66 bpp	24.2%
	0.25 bpp	29.5993	29.2333	**32.7425**	0.35 bpp	28.6%

Fig. 7 Compressed CIF *Foreman* image at 0.125bpp, by JPEG2000 (top) and the proposed scheme (bottom) respectively. Slight color bleeding is observed in JPEG2000 near the neck and collar.

Fig. 8 Compressed *Mobile and Calendar* image (720x576) at 0.25bpp, by JPEG2000 (top) and the proposed scheme (bottom) respectively. It is observed that the proposed scheme works well on painting, fonts and real-life objects.