Colour Image Coding with Matching Pursuit in the Spatio-frequency Domain

Ryszard Maciol, Yuan Yuan, and Ian T. Nabney

School of Engineering and Aplied Science,
Aston University, Aston Triangle, Birmingham, B4 7ET, United Kingdom
{maciolrp,y.yuan1,i.t.nabney}@aston.ac.uk

Abstract. We present and evaluate a novel idea for scalable lossy colour image coding with Matching Pursuit (MP) performed in a transform domain. The idea is to exploit correlations in RGB colour space between image subbands after wavelet transformation rather than in the spatial domain. We propose a simple quantisation and coding scheme of colour MP decomposition based on Run Length Encoding (RLE) which can achieve comparable performance to JPEG 2000 even though the latter utilises careful data modelling at the coding stage. Thus, the obtained image representation has the potential to outperform JPEG 2000 with a more sophisticated coding algorithm.

Keywords: Colour image coding, Matching Pursuit, Wavelets, Run Length Encoding.

1 Introduction

1.1 Colour Image Coding

Due to the large size of raw image and video data files there is great demand for lossy compression methods. Most still image and video data are in colour and are represented for display in RGB colour space thus tripling the raw file size comparing to grayscale. Nevertheless, most of the research effort in algorithms for lossy colour image compression is focused on single-channel methods which are then extended to exploit inter-colour redundancies by applying decorrelating transforms. The current coding standard JPEG 2000 utilises the YC_bC_r transform which attempts to separate luminance (Y) from chrominance (C). Coarser quantisation of C channels improves coding performance without significant visual degradation [2]. JPEG 2000 also utilises the concept of transform coding using discrete wavelets and supports the generation of scalable bit-streams. Sparse approximation techniques that raised interest in the field of image and video compression in the late 90s [1,14] could potentially be the next step in scalable image coding. Moreover they provide new options to exploit inter-channel redundancies in colour images [5,9].

G. Maino and G.L. Foresti (Eds.): ICIAP 2011, Part I, LNCS 6978, pp. 306–317, 2011.

Algorithm 1. Single channel Matching Pursuit [11].

Initialisation: $Rf_1 = f$.
for $n = 1$ to N **do**
  Find atom $g_{\gamma_n} \in \mathcal{D}$ such that:
  $|\langle Rf_n, g_{\gamma_n}\rangle| = \max_{g_\gamma \in \mathcal{D}} (|\langle Rf_n, g_\gamma\rangle|)$.
  Update residual:
  $Rf_{n+1} = Rf_n - \langle Rf_n, g_{\gamma_n}\rangle g_{\gamma_n}$.
end for

1.2 Matching Pursuit

Mallat and Zhang proposed in 1993 [11] a simple greedy technique to obtain a sparse approximation of a given signal f from a Hilbert space $\mathcal{H}$. The algorithm, called Matching Pursuit (MP), finds the approximation of f by a sum of N atoms g_{γ_n} selected from a dictionary $\mathcal{D}$:

$$f \approx \sum_{n=1}^{N} \langle Rf_n, g_{\gamma_n}\rangle g_{\gamma_n}. \tag{1}$$

The dictionary is a set of functions from $\mathcal{H}$ normalised to have unit norm. For any dictionary that spans $\mathcal{H}$ a decomposition given by Eq. 1 converges to f as $N \to \infty$ [11]. Full Search MP, used in image and video compression applications [3,4] for single channel signals, is summarised by Alg. 1. At each iteration the atom most correlated with the actual signal residual Rf_n is selected and removed from Rf_n.

1.3 Multi-channel Matching Pursuit

MP can be extended to decompose vector signals without losing the convergence property [9]. The atom that, according to some criterion, best matches all the components of the input signal is selected. Multi-channel MP for RGB images is summarised by Alg. 2.

Algorithm 2. Multi-channel Matching Pursuit for RGB images.

Initialisation: $Rf_1^r = f^r, Rf_1^g = f^g, Rf_1^b = f^b$.
for $n = 1$ to N **do**
  Find atom $g_{\gamma_n} \in \mathcal{D}$ that maximises the L_2-norm:
  $\gamma_n = \max_{\gamma \in \Gamma} \sqrt{\langle Rf_n^r, g_\gamma\rangle^2 + \langle Rf_n^g, g_\gamma\rangle^2 + \langle Rf_n^b, g_\gamma\rangle^2}$.
  Update residuals:
  $Rf_{n+1}^r = Rf_n^r - \langle Rf_n^r, g_{\gamma_n}\rangle g_{\gamma_n}$.
  $Rf_{n+1}^g = Rf_n^g - \langle Rf_n^g, g_{\gamma_n}\rangle g_{\gamma_n}$.
  $Rf_{n+1}^b = Rf_n^b - \langle Rf_n^b, g_{\gamma_n}\rangle g_{\gamma_n}$.
end for

This algorithm was applied in image space (i. e. to raw RGB values) to colour image coding in [5]. The idea was to explore inter-channel correlations and dependencies of a typical image directly in RGB colour space. In the spatio-frequency domain the dependencies between corresponding subbands of R, G and B channels can be even stronger [6]. In this paper we explain the idea of MP performed in the transform domain and apply it for the first time to colour image coding. The first time MP was performed in the wavelet transform domain for grayscale image coding in [19] and for grayscale video coding in [18]. It was shown in [19] that MP with wavelets can achieve a coding performance comparable to JPEG 2000 for grayscale images. We extend the ideas from [19] to colour coding proposing a new method of coding coefficients.

The next section discusses details of our implementation of MP. Section 3 analyses MP performed in the transform domain. Section 4 describes quantisation and coding of the MP data into bit-stream. Section 5 presents coding results and compares performance with JPEG 2000. Finally, Section 6 concludes the paper and gives the ideas for the performance improvement.

2 Implementation of Matching Pursuit

The main shortcoming of MP is high computational complexity of encoder (atom finding process). On the other side decoding (composing an image back) requires just summing up the atoms which makes MP suitable for asymmetric application in which one encodes the stream once and decodes many times. We argue here that using short support separable filters and performing search in a transform domain keeps also the computational complexity of encoder tractable. Separability refers to the property that each 2D dictionary entry is a tensor product of two 1D vectors. The dictionary is fully specified by a typically small set of 1D filters (mother functions). This reduces number of multiply-accumulate operations when calculating convolutions [14].

The MP algorithm is implemented similarly to the *full 2D separable inner product search* from [20]. The maximal inner products and the corresponding atom indexes are stored for each location in the image. At each iteration, inner products have to be recomputed only on a sub-area of the image. For colour coding it has to be done for all channels and requires approximately three times more multiply-accumulate operations than for grayscale. The overall complexity of our MP implementation can be summarised as:

$$\tau^{(sep)} = \tau^{(sep)}_{init} + \sum_{n=1}^{N} \left(\tau^{(sep)}_{update_n} + \tau^{(sep)}_{search_n} \right). \tag{2}$$

If W denotes the maximum length of bases in the dictionary, S_x the width and S_y the height of the image then, following [20], the overall complexity can be estimated as:

$$\begin{aligned} \tau^{(sep)} = {} & O^{init}(S_x S_y K^2 W) + \\ & O^{update}(N K^2 W^3) + O^{search}(N S_x S_y). \end{aligned} \tag{3}$$

Eq. 3 shows that the size of the dictionary and lengths of bases are critical for complexity of the general MP algorithm with maximum length of basis more important than the number of bases. Moreover, when MP is performed in a transform domain we typically have: $K^2W^3 > S_xS_y$. This implies that recalculation of the inner products is the most demanding part of the algorithm.

3 Matching Pursuit in Transform Domain

MP has been found to be useful for residual video coding [14]. For still image coding the use of non-separable filters of footprint up to quarter of the image size to represent image features at different scales and Fast Fourier Transform (FFT) has been proposed in [4]. The coding performance was comparable to JPEG 2000 at low bit rates. However, matching of long and non-separable filters makes the method from [4] computationally extremely demanding. For practical image coding, as concluded in Sec. 2, one should prefer a dictionary with short filters. When the filters are shorter than 64 samples the FFT slows down the calculation of convolutions. To preserve low complexity and a dictionary capable of capturing image features at different scales the use of the 2D Discrete Wavelet Transform (2D-DWT) has been proposed in [19]. MP decomposition was performed for wavelet subbands. Like the codec in [4] the method proposed in [19] is comparable in coding performance to the JPEG 2000 standard but additionally has a tractable computational complexity.

In this work we present and analyse in more detail the idea of performing MP in transform domain. Performing MP after transformation like DCT or DWT reduces complexity and improves coding performance. Let us start with the simple observation that applying an orthonormal linear transform T does not change the output of MP. If the transform T is linear and preserves inner product,

$$\langle f, g\rangle = \langle T\{f\}, T\{g\}\rangle \text{ for all } f, g \in \mathcal{H}, \tag{4}$$

then the MP decomposition of signal f (see Eq. 1) obtained in the transform domain is:

$$T\{f\} \approx \sum_{n=1}^{N} \langle T\{Rf_n\}, T\{g_{\gamma_n}\}\rangle\, T\{g_{\gamma_n}\}. \tag{5}$$

In practice it can be computationally easier to match filters in the spatio-frequency domain. To give an example for the discrete case, consider a dictionary entry with support W: $g(t) = 1/\sqrt{W}$ for $t = 1, 2, \ldots, W$. Its DCT or DFT is the Dirac delta $g(\omega) = 1$ with support 1. Performing an inner product with such a short signal requires only one multiplication. It is known that for transforms like DCT, DFT or DWT the filters applied locally in the transform domain correspond to some global structures in the image domain. Therefore MP can be more efficient when performed in the transform domain. In principle, a very similar idea was used indirectly in [4] where convolutions with filters in a dictionary were performed in the Fourier domain. In [19] filters designed for video coding in the image domain were applied to wavelet subbands after performing

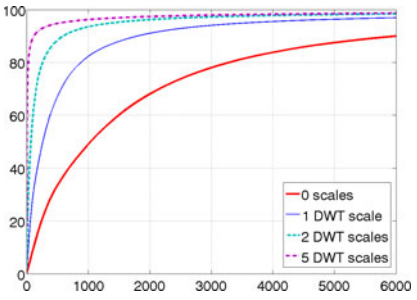


Fig. 1. Percentage of the image energy (y-axis) represented by a given number of atoms (x-axis) using different numbers of wavelet scales (grayscale Goldhill)

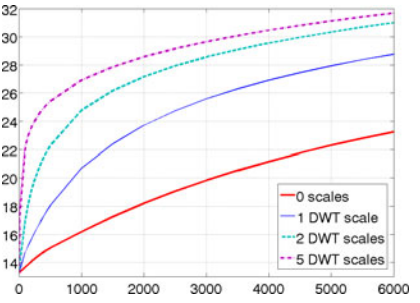


Fig. 2. PSNR performance in dB (y-axis) for a given number of atoms (x-axis) using different numbers of wavelet scales (grayscale Goldhill)

2D-DWT with CDF 9/7 filters from lossy mode of JPEG 2000. As the wavelet transform does not change a signal dimension, the overall size of a dictionary in the image domain remains the same. Thanks to the energy compaction property of DWT, the atoms found in the wavelet domain in initial iterations have high amplitudes. Hence, they contribute more to the whole image energy as shown in Fig. 1. In Fig. 2 we see corresponding values of PSNR. The dictionary applied for wavelet subbands is capable of giving a few orders of magnitude sparser representation than the same dictionary applied in the image domain. Moreover at initial steps of MP there are usually more atoms found in lower frequencies what gives a potential for more efficient coding. The dictionaries we use in this study for colour and grayscale coding were trained using Basis Picking method from [12] on colour and grayscale (i. e. luminance only) Goldhill image respectively. Both dictionaries contain 16 1D bases of maximal footprint 9.

4 Quantisation and Coding

4.1 Quantisation

For data compression applications MP decomposition has to be encoded into a bit-stream. The values $a_n = \langle Rf_n, g_{\gamma_n} \rangle$ have to be quantised (e. g. rounded) to the values A_n. Quantisation is performed inside the MP loop [14] with the aim of correcting the introduced quantisation error during later iterations. For the MP decomposition given by Eq. 1 the Parseval-like equality is satisfied [11]:

$$||f||^2 = \sum_{n=1}^{N} a_n^2 + ||Rf_{N+1}||^2. \tag{6}$$

Eq. 6 is a direct consequence of the update step from Alg. 1. If we replace a_n by A_n in the update step to reflect in-loop quantisation then, for real values, Eq. 6 will change to:

$$||f||^2 = \sum_{n=1}^{N} A_n(2a_n - A_n) + ||Rf_{N+1}||^2. \quad (7)$$

To preserve convergence of the algorithm the energy of residual Rf_n has to keep decreasing [11]. Therefore we may use any quantisation method for which $A_n(2a_n - A_n) > 0$ which is equivalent to a_n, A_n having the same sign and their absolute values to follow Eq. 8 [15].

$$0 < |A_n| < 2|a_n|. \quad (8)$$

Our grayscale implementation utilises Precision Limit Quantisation (PLQ) [13]. The original idea of PLQ is to keep only the most significant bit of a_n and some refinement bits governed by the parameter PL. Then the value $|a_n|$ is quantised to: $|A_n| = r2^k$, where k indicates bitplane and r refinement. The value of the parameter PL is taken to be $PL = 2$, as advised in [19], which in our case means that $r \in \{1.25, 1.75\}$.

The colour codec uses PLQ and Uniform Quantisation. The amplitude with maximal value over the three colour channels (a_n^{max}) is quantised using PLQ and serves as a base for grouping atoms. The atoms with the same quantised absolute value of maximal amplitude ($|A_n^{max}|$) compose one group. We record the channel c_n for which the maximal value occurred. The remaining two amplitudes for the other two colours are quantised using dead-zone uniform scalar quantisation with L bins [7]. The value of L has been experimentally chosen to be as low as $L = 2$ in order to maximally reduce the number of bits required. The two numbers d_n^1 and d_n^2, sent to the encoder, represent either dead-zone or the quantised amplitude with its sign.

4.2 Atom Encoding

After MP decomposition and quantisation, the data to be encoded form a matrix in which rows represent atoms. There are 8 columns containing the following variables for colour coding:

$1:$	s_n, sign of the maximal amplitude,	$s_n \in \{-1, 1\}$,
$2-3:$	d_n^1, d_n^2, quantised amplitude differences,	$d_n^* \in \{1, 2, \ldots, 2L+1\}$,
$4:$	c_n, maximum amplitude colour channel,	$c_n \in \{1, 2, 3\}$,
$5:$	w_n, sub-band index,	$w_n \in \{1, 2, \ldots, 3S+1\}$,
$6:$	λ_n, 2D dictionary entry,	$\lambda_n \in \{1, 2, \ldots, 256\}$,
$7-8:$	x_n, y_n, atom location inside the sub-band w_n,	$x_n \in \{1, \ldots, W_{x_n}\}, y_n \in \{1, \ldots, W_{y_n}\}$.

For grayscale coding there are only 5 columns: s_n,w_n,λ_n,x_n and y_n from which only 3 are being reordered. Data from columns 1-6 (or 1-3 for grayscale) are encoded group by group using Alg. 3 based on Run Length Encoding (RLE). The rows are ordered in a lexicographical order recommended for databases indexes [8]. Encoding inside each group is done calling recursively column by column the Alg. 3. The coding performance depends on the column order (see Sec. 5).

Algorithm 3. One stage of encoding.

input: $\{v_s\}_{s=1,2,\ldots,n}$ with $s < s' \Rightarrow v_s \leq v_{s'}$
$s = 1$
while $s < n$ **do**
  if there are 2 times more symbols than alphabet entries remaining **then**
    encode all zero lengths (if any) and one non-zero run length l
    $s = s + l$
  else
    encode symbol v_s directly
    $s = s + 1$
  end if
end while

Therefore a fixed permutation of columns π is applied prior to the sorting. For each stage of encoding the input of Alg. 3 is the sorted sequence $\{v_s\}$ from an alphabet of the size determined by the column number. At each iteration a decision is made whether to encode the symbol v_s directly or to signal its run length. RLE is used when the run length of 2 or more symbols is expected. An expected run length is indicated by the ratio of the remaining symbols count to the size of the alphabet they can come from. The atom locations (the last two columns) are always encoded as the two raw values x_n and y_n from the ranges $1 \ldots W_{x_n}$ and $1 \ldots W_{y_n}$ respectively, where $W_{x_n} \times W_{y_n}$ is a dimension of the sub-band w_n. All the symbols are sent to the arithmetic coder [17] that uses models which assume uniform distributions for each column. Arithmetic coding allows, knowing the probability distribution of data, to achieve compression ratio close to a theoretical bound given by the Shannon's entropy [17]. Uniform distribution has the highest entropy among the discrete distributions. Therefore the results shown here can serve as the upper bound for the sizes of encoded streams. [1]

5 Coding Performance

As evaluation metric we use PSNR. For colour images it is averaged over RGB channels:

$$PSNR = 10 \log_{10} \left(\frac{3 \cdot 255^2}{MSE_r + MSE_g + MSE_b} \right), \tag{9}$$

where MSE_r, MSE_g, MSE_b are mean squared errors calculated for R, G and B channels respectively. Although PSNR is known to correlate poorly with human visual perception, especially in the case of colour images, it measures the mathematical properties of the algorithms used. Comparisons with JPEG 2000 are done using the same wavelet filters and the same number of scales $S = 5$. For fair comparison of colour codecs an option of JPEG 2000 which minimises mean squared error (i. e *no_ weights* switch for Kakadu implementation [16]) was used.

[1] More details about implementation of the whole coding system and its evaluation can be found in [10].

Table 1. Number of bits required for 6000 grayscale atoms for different column orders

image	the best order	the worst order	sub-optimal (π_g) (2,3,1)	$2 \cdot \frac{worst-best}{worst+best}$
Barbara, 720 × 576	105699 (2,1,3)	115314 (1,3,2)	106577	8.70%
Goldhill, 720 × 576	102978 (2,3,1)	111453 (3,1,2)	102978	7.90%
Lena, 512 × 512	102321 (2,3,1)	110012 (3,1,2)	102321	7.24%
Lighthouse, 768 × 512	104441 (2,1,3)	112076 (1,3,2)	104905	7.05%
Parrots, 768 × 512	107218 (2,3,1)	113520 (3,1,2)	107218	5.71%
Peppers, 512 × 512	102222 (2,3,1)	108971 (3,1,2)	102222	6.39%

Table 2. Number of bits required for 6000 colour atoms for different column orders

image	the best order	the worst order	sub-optimal (π_c) (5,2,3,4,6,1)	$2 \cdot \frac{worst-best}{worst+best}$
Barbara, 720 × 576	126937 (5,2,3,4,6,1)	148111 (6,1,4,5,3,2)	126937	15.40%
Goldhill, 720 × 576	124938 (5,3,2,4,6,1)	142009 (1,6,4,5,3,2)	124971	12.79%
Lena, 512 × 512	127077 (5,3,2,4,6,1)	142753 (6,1,4,5,2,3)	127113	11.62%
Lighthouse, 768 × 512	121129 (5,2,3,4,6,1)	147995 (1,6,4,5,3,2)	121129	19.97%
Parrots, 768 × 512	130512 (5,3,2,6,1,4)	145187 (6,4,1,5,3,2)	130739	10.65%
Peppers, 512 × 512	128686 (5,3,2,4,6,1)	138515 (6,1,4,5,2,3)	128803	7.36%

At first, a set of experiments for standard test images of different sizes has been done to find an optimal column order to apply Alg. 3. Each permutation was tried for 6 grayscale (see Tab. 1) and colour images (see Tab. 2). The differences in the size of a bit-stream for the different column orders are significant. For grayscale, where there are only 6 possible column permutations, the differences between maximum and minimum bit-stream sizes are less than 10% (Tab. 1). However, for colour, where we have 720 orders, the differences can be up to 20% (Tab. 2). In the proposed coding scheme the best or close to the best performance is achieved when atoms are sorted by wavelet scale first. Atom indexes and signs of the amplitudes are the last sorting criteria for both grayscale and colour. The column permutations that perform close to optimal for all tested images are: $\pi_g = (2, 3, 1)$ for grayscale and $\pi_c = (5, 2, 3, 4, 6, 1)$ for colours.

R-D performance plots are shown in Fig. 4 including PSNR results for a default mode of Kakadu. Fig. 3 presents a visual comparison example. In general both grayscale and colour codecs are comparable to JPEG 2000, often outperforming the latter at low bit rates. However for many standard test images like Parrots (Fig. 4c and 4f) or Lighthouse a coding performance is still significantly worse. On average (see Tab. 3) MP performs better than the standard at 0.1 bpp but for the higher rates the average performance is worse. We believe that modelling the distributions of wavelet scale indexes and run lengths for arithmetic coding could give a significant improvement. For example, as mentioned in Sec. 3, in initial iterations the atoms are more likely to be found in low frequencies.

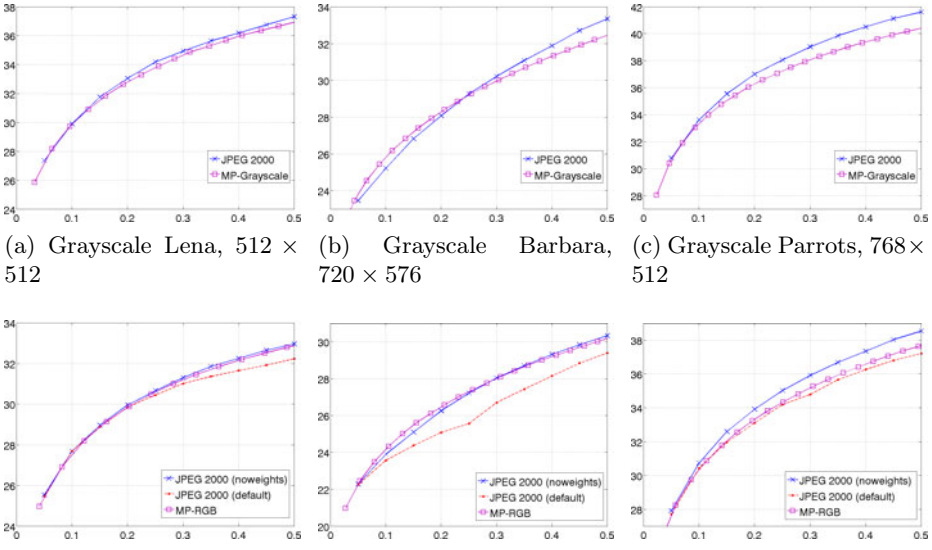
(a) Original image (b) MP, 6086 atoms, 29.95 dB

(c) J2K, default, 29.60 dB (d) J2K, no-weights, 29.93 dB

Fig. 3. Visual comparison at 0.30 bpp against JPEG 2000 for Goldhill, 720 × 576

Table 3. Coding performance comparisons against JPEG 2000 at fixed bit-rates

	0.1 bpp		0.3 bpp		0.5 bpp	
grayscale image	J2K	MP-Gray	J2K	MP-Gray	J2K	MP-Gray
Barbara, 720 × 576	25.21	**26.02**	30.21	**30.44**	**33.35**	33.09
Goldhill, 720 × 576	28.90	**29.12**	32.30	**32.38**	**34.25**	34.11
Lena, 512 × 512	**29.90**	29.83	**34.94**	34.58	**37.32**	36.85
Lighthouse, 768 × 512	25.77	**25.81**	**29.57**	29.28	**32.11**	31.49
Parrots, 768 × 512	**33.62**	33.43	**39.04**	38.43	**41.61**	40.95
Peppers, 512 × 512	**29.66**	29.44	**34.16**	33.74	**35.84**	35.46
Average	28.84	**28.94**	**33.37**	33.14	**35.75**	35.33
colour image	J2K	MP-RGB	J2K	MP-RGB	J2K	MP-RGB
Barbara, 720 × 576	23.89	**24.23**	28.03	**28.04**	**30.33**	30.20
Goldhill, 720 × 576	**27.24**	27.22	29.93	**29.95**	**31.46**	31.38
Lena, 512 × 512	**27.68**	27.64	**31.31**	31.25	**32.97**	32.95
Lighthouse, 768 × 512	**25.18**	25.00	**28.68**	28.18	**30.95**	30.19
Parrots, 768 × 512	**30.72**	30.40	**35.92**	35.23	**38.54**	37.73
Peppers, 512 × 512	25.57	**25.80**	29.61	**29.68**	31.17	**31.24**
Average	26.71	**26.72**	**30.58**	30.39	**32.57**	32.28



(a) Grayscale Lena, 512 × 512 (b) Grayscale Barbara, 720 × 576 (c) Grayscale Parrots, 768 × 512

(d) Colour Lena, 512 × 512 (e) Colour Barbara, 720 × 576 (f) Colour Parrots, 768 × 512

Fig. 4. R-D performance comparisons between the proposed MP coding and Kakadu implementation of the JPEG 2000 standard (y-axis: PSNR [dB], x-axis: bit-rate [bpp])

Our current C++ implementation encodes 8000 colour atoms (this corresponds to a bit-rate of 0.40 bpp for Goldhill image) under Linux on PC with Intel Core 2 Duo in less than 0.2 s which is negligible comparing to finding these atoms by MP algorithm that takes around 80 seconds for the dictionary used in this work and images of dimension 720 × 576.

6 Conclusions

We have presented a novel approach of decomposing and encoding images that has shown comparable R-D performance to the current coding standard JPEG 2000 at low bit rates. Our idea of encoding atoms is especially promising for colour images. MP is performed after the discrete wavelet transform to reduce complexity and improve sparsity [19]. MP decomposition of an image is represented as a matrix of rows. These rows are sorted in lexicographical order after permutation of columns and encoded using run length and then arithmetic coding with simple data model that assumes equal probabilities of each type of symbol (each column). The optimal column orders were found for both grayscale and colour data. The open questions that are currently under our investigation include: more sophisticated data modelling for coding and finding the optimal dictionary.

Acknowledgement. Ryszard Maciol would like to thank Aston University for funding the studentship and support that made this work possible.

References

1. Bergeaud, F., Mallat, S.: Matching pursuit of images. In: Proc. International Conference on Image Processing, vol. 1, pp. 53–56 (1995)
2. Christopoulos, C., Skodras, A., Ebrahimi, T.: The JPEG 2000 still image compression standard. IEEE Signal Processing Magazine 18(5), 36–58 (2001)
3. Czerepinski, P., Davies, C., Canagarajah, N., Bull, D.: Matching pursuits video coding: Dictionaries and fast implementation. IEEE Transactions on Circuits and Systems for Video Technology 10(7), 1103–1115 (2000)
4. Figueras i Ventura, R.M., Vandergheynst, P., Frossard, P.: Low-rate and flexible image coding with redundant representations. IEEE Transactions on Image Processing 15(3), 726–739 (2006)
5. Figueras i Ventura, R.M., Vandergheynst, P., Frossard, P., Cavallaro, A.: Color image scalable coding with matching pursuit. In: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 3, pp. 53–56 (2004)
6. Gershikov, E., Lavi-Burlak, E., Porat, M.: Correlation-based approach to color image compression. Image Communications 22(9), 719–733 (2007)
7. Gray, A.M., Gersho, R.: Vector Quantization and Signal Compression, 5th edn. Kluwer Academic Publishers, Dordrecht (1992)
8. Lemire, D., Kaser, O.: Reordering columns for smaller indexes. Information Sciences 181(12), 2550–2570 (2011)
9. Lutoborski, A., Temlyakov, V.M.: Vector greedy algorithms. Journal of Complexity 19(4), 458–473 (2003)
10. Maciol, R., Yuan, Y., Nabney, I.T.: Grayscale and colour image codec based on matching pursuit in the spatio-frequency domain. Tech. rep., Aston University, `http://eprints.aston.ac.uk/15194/` (2011)
11. Mallat, S.G., Zhang, Z.: Matching pursuits with time-frequency dictionaries. IEEE Transactions on Signal Processing 41(12), 3397–3415 (1993)
12. Monro, D.M.: Basis picking for matching pursuits image coding. In: Proc. International Conference on Image Processing, vol. 4, pp. 2495–2498 (2004)
13. Monro, D.M., Poh, W.: Improved coding of atoms in matching pursuits. In: Proc. International Conference on Image Processing, vol. 3, pp. 759–762 (2003)
14. Neff, R., Zakhor, A.: Very low bit-rate video coding based on matching pursuits. IEEE Transactions on Circuits and Systems for Video Technology 7(1), 158–171 (1997)
15. Neff, R., Zakhor, A.: Modulus quantization for matching-pursuit video coding. IEEE Transactions on Circuits and Systems for Video Technology 10(6), 895–912 (2000)
16. Taubman, D.: Kakadu JPEG 2000 implementation, `http://www.kakadusoftware.com/`

17. Witten, I.H., Neal, R.M., Cleary, J.G.: Arithmetic coding for data compression. Communications of the ACM 30(6), 520–540 (1987)
18. Yuan, Y., Monro, D.M.: 3D wavelet video coding with replicated matching pursuits. In: Proc. IEEE International Conference on Image Processing, vol. 1, pp. 69–72 (2005)
19. Yuan, Y., Monro, D.M.: Improved matching pursuits image coding. In: Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 2, pp. 201–204 (2005)
20. Yuan, Y., Evans, A.N., Monro, D.M.: Low complexity separable matching pursuits [video coding applications]. In: Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 3, pp. 725–728 (2004)