
Ryszard Maciol, Yuan Yuan and Ian T. Nabney

First revised version: March 15, 2011

Last updated: May 23, 2011

A shortened version is going to be published in
The Proceedings of the 16th International Conference on Image Analysis and Processing
ICIAP 2011

Ryszard Maciol, Yuan Yuan and Ian T. Nabney

Aston University
School of Engineering and Applied Science
B4 7ET, Birmingham, United Kingdom

Abstract

This report presents and evaluates a novel idea for scalable lossy colour image coding with Matching Pursuit (MP) performed in a transform domain. The benefits of the idea of MP performed in transform domain proposed in [19] are analysed in details. The main contribution of this work is extending MP with wavelets to colour coding and proposing a coding method. We exploit correlations between image subbands after wavelet transformation in RGB colour space. Then, a new and simple quantisation and coding scheme of colour MP decomposition based on Run Length Encoding (RLE), inspired by the idea of coding indexes in relational databases, is applied. As a final coding step arithmetic coding is used assuming uniform distributions of MP atom parameters. The target application is compression at low and medium bit rates. Coding performance is compared to JPEG 2000 showing the potential to outperform the latter with more sophisticated than uniform data models for arithmetic coder. The results are presented for grayscale and colour coding of 12 standard test images.

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

1 Introduction

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 redundencies by applying decorrelating transforms. The current coding standard JPEG 2000 utilises the YCbCr 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 transformed-based image coding. Moreover they provide new options to exploit inter-channel redundencies in colour images [5, 9].

The next section starts with an overview of the simplest sparse approximation technique called Matching Pursuit. Then its extension to multichannel signals is outlined. The potential for colour coding directly in RGB space is recognised. The idea of performing MP in the transform domain is described later in Section 2.3. Then the details of the implementation of the transform part of our codec are presented (Sections 2.4-2.5). Section 3 explains quantisation and coding of transformed data. Our new idea of coding colour coefficients is presented in Section 3.3. In Section 4 a comparison with JPEG 2000 is done. Finally, conclusions and directions for future work are given in Section 5.

1 The latest version of this report can be found at: http://mp-rgb.info

2 Ryszard Maciol would like to thank Aston University for funding the studentship and support that made this work possible.
2 Implementation and Complexity of Matching Pursuit

2.1 Single-channel Matching Pursuit

Mallat and Zhang proposed in 1993 [10] a simple greedy technique to obtain a sparse approximation of a given signal \( f \) from a Hilbert space \( H \). The algorithm, called Matching Pursuit (MP), finds the approximation of \( f \) by a sum of \( N \) atoms \( g_{\gamma_n} \) selected from a dictionary \( D \):

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

(1)

The dictionary is a set of functions from \( H \) normalised to have unit norm. For any dictionary that spans \( H \) a decomposition given by Equation 1 converges to \( f \) as \( N \to \infty \) [10]. Full Search MP, used in image and video compression applications [3, 4] for single channel signals, is summarised by Algorithm 1. At each iteration the atom most correlated with the actual signal residual \( Rf_n \) is selected and removed from \( Rf_n \). In image processing the space \( H \) is a space of all possible images represented as matrices and has finite dimension.

Algorithm 1 Single channel Matching Pursuit [10].

Initialisation: \( Rf_1 = f \).

for \( n = 1 \) to \( N \) do

Find atom \( g_{\gamma_n} \in D \) such that:

\[
|\langle Rf_n, g_{\gamma_n} \rangle| = \max_{g_{\gamma_n} \in D} (|\langle Rf_n, g_{\gamma_n} \rangle|).
\]

Update residual:

\[
Rf_{n+1} = Rf_n - \langle Rf_n, g_{\gamma_n} \rangle g_{\gamma_n}.
\]

end for

2.2 Colour Atom Search

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 Algorithm 2.

Algorithm 2 Multi-channel Matching Pursuit for RGB images.

Initialisation: \( Rf^r_1 = f^r, Rf^g_1 = f^g, Rf^b_1 = f^b \).

for \( n = 1 \) to \( N \) do

Find atom \( g_{\gamma_n} \in D \) that maximises the \( L_2 \)-norm:

\[
\gamma_n = \max_{\gamma_n \in \Gamma} \sqrt{\langle Rf^r_n, g_{\gamma_n} \rangle^2 + \langle Rf^g_n, g_{\gamma_n} \rangle^2 + \langle Rf^b_n, g_{\gamma_n} \rangle^2}.
\]

Update residuals:

\[
Rf^r_{n+1} = Rf^r_n - \langle Rf^r_n, g_{\gamma_n} \rangle g_{\gamma_n}.
\]

\[
Rf^g_{n+1} = Rf^g_n - \langle Rf^g_n, g_{\gamma_n} \rangle g_{\gamma_n}.
\]

\[
Rf^b_{n+1} = Rf^b_n - \langle Rf^b_n, g_{\gamma_n} \rangle g_{\gamma_n}.
\]

end for

It has to be noted that theoretical results in [9] were given for finding an atom that maximises the absolute value of an inner product over all the channels is maximised. The selection rule from [9] can be expressed by Equation 2.

\[
\gamma_n = \max_{\gamma_n \in \Gamma} (|\langle Rf^r_n, g_{\gamma_n} \rangle|, |\langle Rf^g_n, g_{\gamma_n} \rangle|, |\langle Rf^b_n, g_{\gamma_n} \rangle|).
\]  

(2)

In [5], where Algorithm 2 was applied in image space (i.e. to raw RGB values), atoms with maximal \( L_2 \) norm were selected. This corresponds to Mean Squared Error (MSE) minimisation, does not affect convergence and is also used here. The flexibility in choosing selection criteria gives the potential to use criteria related to quality metrics more correlated with human visual perception. Nevertheless, finding such criteria is beyond scope of this report and is left as a potential further research.

The general idea behind matching the same atom to all three channels is to explore the inter-channel correlations and dependencies of a typical image directly in RGB colour space. In the spatio-frequency domain the dependencies between corresponding subbands for RGB channels can be even stronger [6]. In this paper we explain the idea of MP performed in transform domain and apply it for the first time to colour image coding. MP was first performed in wavelet transform domain for grayscale image coding.
in [21] and for grayscale video coding in [20]. It was shown in [21] that MP with wavelets can achieve a coding performance comparable to JPEG 2000 for grayscale images. We extend the ideas from [21] to colour coding and propose a new method of coding coefficients.

2.3 Matching Pursuit in Transformed Domain

MP has been found to be useful for residual video coding [14]. 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 you encode the stream once and decode many times. The structure of a dictionary is the most critical for the complexity as well as for the coding efficiency. For still image coding the use of non-separable filters and the Fast Fourier Transform (FFT) has been proposed in [4]. The filters had the footprints up to a quarter of the image size to represent image features at different scales. The coding performance was comparable to JPEG 2000 at low bit rates. However, matching of the long and non-separable filters makes the method from [4] computationally extremely demanding. For practical image coding one should prefer a dictionary with short support 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 [21]. MP decomposition was performed for wavelet subbands. Like the codec in [4] the method proposed in [21] 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. It is shown that performing MP after transformation using 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,

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

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

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

In practice it can be computationally easier to match filters in the spatio-frequency domain. A similar idea was used indirectly in [4] where convolutions with filters in a dictionary were performed in the Fourier domain. To give an example for the discrete case, consider a dictionary entry with support $W$: $g(t) = 1/\sqrt{W}$ for $t = 1, 2, ..., W$. 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 long support functions will have short support after transformation and hence, MP is computationally more efficient in the transform domain.

In [21] filters designed for video coding in the image domain were applied to wavelet subbands after performing 2D-DWT with CDF 9/7 filters from lossy mode of JPEG 2000. Filters applied locally in the
wavelet domain correspond to the global structures in the image domain. As 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 the 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 Figure 1. In Figure 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 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 next section describes the process of choosing a dictionary for our work and Section 3 deals with coding.

2.4 Building the Dictionaries

The dictionary, in the most general form, can be defined as a set of $B$ matrices:

$$
\mathcal{D}^{(\text{non-sep})} = \{G_1, G_2, \ldots, G_B\}.
$$

For image representation we are interested in translation invariant dictionaries what means that matrices $G_i$ can be located at any point in the image. Due to the complexity issues we are focused on the dictionaries that are separable, i.e. each matrix $G_i$ can be represented as a tensor product of vectors:

$$
\mathcal{D}^{(\text{sep})} = \{g_i \otimes g_j\}_{i,j=1,\ldots,b}.
$$

Hence a set of $b$ 1D filters $\{g_i\}_{i=1,2,\ldots,b}$ defines a dictionary. In this study we are interested in short-support discrete filters that are matched with wavelet coefficient as described in the previous section. Using Basis Picking method from [11] we trained a dictionary for colour $D_c$ and grayscale $D_g$ coding separately. We initialised the both dictionaries with 2 bases: $D_c^2 = \{g_1, g_2\}$, where $g_1 = \{1\}$ (Dirac delta) and $g_2 = \{\sqrt{2}/2, \sqrt{2}/2\}$. Our search was limited to the filters of maximal footprint of 9 as advised in [13]. Picking was performed from candidate set of 496 Gabor functions of the form:

$$
g(\sigma, f, w, \phi)(t) = \exp\left(-\frac{\pi t^2}{w}\right) \cos\left(\frac{\pi ft}{w} + \phi\right),
$$

sampled at $2w + 1 = 3, 5, 7$ or 9 points ($t \in \{-w, \ldots, w\}$) and normalised to have unit norm. The values of parameters $\sigma, f, \phi$ are taken, following [11], to be: $\sigma \in \{1, 2, 4, 8, 12, 16, 20, 24\}$, $f \in \{0, 1, \ldots, w\}$, $\phi \in \{0, \pi/4, \pi/2, 3\pi/4, \pi/2\}$. At each iteration of picking we select a function, that decreases the most the mean squared error of 6000 atoms decomposition. Training images were colour and grayscale (i.e. Y channel after YCC colour transform) version of Goldhill image of size 720 $\times$ 576. 1D filters that define the used dictionaries, as their appeared during the training process, are collected in Figures 3 and 4. In order to be efficient for a wide range of images, the dictionary have to exhibit a particular structure. Short, uncorrelated filters should be included since the longer filters found for one image can fail to decompose others efficiently. Those trained on Goldhill have the desired properties, therefore we use them for evaluation of our coding method. We found $b = 16$ bases as a good trade-off between complexity and coding efficiency. Dictionaries design is one of the topics under our investigation and the details are beyond the scope of this report. The next section only deals with the question of how the structure of a dictionary affects the complexity of our implementation of the MP.

2.5 Complexity Analysis

The MP algorithm is implemented in this work similarly to the full 2D separable inner product search from [19]. The maximal inner products and the corresponding atom indexes are stored for each location. At each iteration, inner products have to be recomputed only on a sub-area of one subband image. For colour coding it has to be done for all channels and requires approximately 3 times more multiply-accumulate operations than for grayscale. In this work we analyse the complexity of described approach showing how it depends on the structure of the dictionary $\mathcal{D}$. We argue that using short support separable filters and performing search in a transform domain keeps the computational complexity of the encoder tractable.

Let us denote the lengths of the filters $g_i$ by $w_i$ and assume that the atoms are sorted by their length, i.e. $w_i \leq w_{i+1}$ for $i = 1, \ldots, b - 1$. Following [14], if separability is ignored then calculation of all inner products for an image of the size $N$ by $M$ would require $T^{(\text{non-sep})}$ multiply-accumulate operations:

$$
T^{(\text{non-sep})} = NM \left(\sum_{i=1}^{b} w_i\right)^2.
$$

4
By exploiting separability in the same way as in [14] (see Algorithm 3), the number of operations can be reduced to:

$$T^{(sep)} = NM (b + 1) \sum_{i=1}^{b} w_i.$$  \hfill (9)

**Algorithm 3** Inner products calculation from [14].

```plaintext
for i = 1 to b do
    Calculate vertical inners $V_i = \langle Rf_n, g_i \rangle$
    for j = 1 to b do
        Calculate horizontal inners $\langle Rf_n, g_i \otimes g_j \rangle = \langle V_i, g_j \rangle$
    end for
end for
```

We propose a simple optimisation which is based on calculating inner products with shorter filters in inner loop (Algorithm 4). Usefulness of such a modification depends on the structure of a dictionary, i.e. on the number of filters and their lengths.

**Algorithm 4** Optimisation of inner products calculation.

```plaintext
for i = 1 to b do
    Calculate vertical inners $V_i = \langle Rf_n, g_i \rangle$
    Calculate horizontal inners $H_i = \langle Rf_n, g_i \rangle$
    for j = 1 to i - 1 do
        Calculate horizontal inners $\langle Rf_n, g_i \otimes g_j \rangle = \langle V_i, g_j \rangle$
        Calculate vertical inners $\langle Rf_n, g_j \otimes g_i \rangle = \langle H_i, g_j \rangle$
    end for
    Calculate vertical inners $\langle Rf_n, g_i \otimes g_i \rangle = \langle H_i, g_i \rangle$
end for
```
If we denote $S = \sum_{i=1}^{b} w_i$, the sum of filter sizes representing number of operations in outer loop of Algorithm 3 and $F = \sum_{i=1}^{b} \sum_{j=1}^{i-1} w_j$, which represents number of operations in inner loop then the number of multiply-accumulate operations for Algorithm 3 is equal to:

$$T^{(sep)*} = NM (3S + 2F).$$

(10)

For the colour dictionary $D_c$ trained on Goldhill image (with $b = 16$) used in our case we have $S = 95$, $F = 524$ and:

$$T^{(sep)} = (16 + 1) \cdot 95NM = 1615NM$$

$$T^{(sep)*} = (3 \cdot 95 + 2 \cdot 524)NM = 1333NM,$$

which means a 17% speed-up for each iteration. For the grayscale dictionary $D_g$ (with $b = 16$) $S = 85$, $F = 462$ and:

$$T^{(sep)} = (16 + 1) \cdot 85NM = 1445NM$$

$$T^{(sep)*} = (3 \cdot 85 + 2 \cdot 462)NM = 1179NM,$$

which means around 18% less multiply-accumulate operations.

The overall complexity of the proposed implementation of MP can be estimated as follows, regardless the improvement:

$$T^{(sep)} = T^{(sep)}_{in} + \sum_{n=1}^{K} \left(T^{(sep)}_{update_n} + T^{(sep)}_{search_n}\right).$$

(11)

Initialisation and update steps involve mainly multiply-accumulate operations performed according to Algorithm 4. The search for the maximum value is performed over the whole image at initialisation and only over the modified subbands at each iteration.

$$T^{(sep)}_{search} = O(NM).$$

(12)
Update of the maximum inner product is performed only on an area of size $W_n \times H_n$:

$$W_n = w_n + 2 \max_i w_i - 2 \leq 3W,$$

$$H_n = h_n + 2 \max_i w_i - 2 \leq 3W, \quad (13)$$

where $w_n \times h_n$ is the footprint of the atom found at iteration $n$, $W = \max_i (w_i) = w_b$ denotes the longest basis length in a dictionary $D$. By combining Equations 10-12

$$T^{(sep)} = NM(3S + 2F) +$$

$$+ \sum_{n=1}^{K} (W^2(3S + 2F) + O(NM)). \quad (14)$$

By applying inequalities from Equation 13 to Equation 14 we can estimate complexity as:

$$T^{(sep)} \leq (3S + 2F)(NM + 9KW^2) + O(KNM). \quad (15)$$

Taking the upper bounds for $S$ and $F$: $S \leq bW$ and $F \leq b^2W$ it follows that:

$$T^{(sep)} = O(b^2W^3K) + O(KNM). \quad (16)$$

Equations 14 and the approximation given by Equation 16 shows that size of the dictionary and lengths of bases are critical for a complexity of the MP algorithm. Moreover, as shown by Equations 13 and 14 the maximal basis size affects significantly the number of operations. In the case of performing MP in transformed domain, described in Section 2.3, there is typically: $b^2W^3 > NM$ which implies that complexity of the main part of calculation strongly depends on the maximum filter size as well as the number of filters in a dictionary.

The details of complexity are even more complicated since they depend on which subband the atoms are found as well as on the sizes of the atoms themselves. Initial iterations, when more lower frequency atoms are picked, are computed faster. Experimental results for complexity analysis of MP performed during the process of training dictionaries on different images are shown in Figure 5. It is shown that the size of an input image is less critical for complexity than the structure of a dictionary. MP with the dictionaries trained on Goldhill decomposes $720 \times 576$ image at least as fast as $768 \times 512$ image. On the other side, it has to be remembered that there is usually more atoms needed for bigger images to achieve the same distortion. Our current C++ implementation finds 6000 colour atoms in less than 40 seconds for the dictionary used in this work and colour images of dimension $720 \times 576$ (see Figure 5).
3 Quantisation and Atom Coding

For data compression applications MP decomposition has to be encoded into a bit-stream. The values \( a_n = (Rf_n, g_n) \) have to be quantised (e.g. rounded) to the values \( A_n \) that come from a finite alphabet. 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 Equation \( 1 \) the Parseval-like equality is satisfied \( [10] \):

\[
||f||^2 = \sum_{n=1}^{N} a_n^2 + ||Rf_{N+1}||^2. \quad (17)
\]

Equation \( 17 \) is a direct consequence of the update step from Algorithm \( 1 \). If we replace \( a_n \) by \( A_n \) in the update step to reflect in-loop quantisation then Equation \( 17 \) will change to:

\[
||f||^2 = \sum_{n=1}^{N} A_n(2a_n - A_n) + ||Rf_{N+1}||^2. \quad (18)
\]

To preserve convergence of the algorithm the energy of residuum \( Rf_n \) has to keep decreasing \( [10] \). Therefore we may use any quantisation method for which \( A_n(2a_n - A_n) > 0 \) which is equivalent to \( [15] \):

\[
0 < A_n < 2a_n. \quad (19)
\]

Our grayscale implementation utilises Precision Limit Quantisation (PLQ) \( [12] \) while the colour codec uses PLQ and Uniform Quantisation (see Section \( 3.2 \)).

3.1 PLQ Quantisation

The original idea of PLQ comes from bit-plane coding where only the most significant bits are encoded for each coefficient (here each atom amplitude) \( [12] \). Quantisation bins in PLQ with parameter \( PL(PL > 0) \) are of the form:

- bit-planes: \( k = M, M - 1, \ldots, 1, 0, -1, \ldots \)
- refinements: \( r = 1, 1 + 2^{1-PL}, \ldots, 2 - 2^{1-PL} \)
- quantisation bins: \( B_{kr} = (r2^k, (r + 2^{1-PL})2^k] \). \( (20) \)

The original approach was to keep only the most significant bit and some refinement bits governed by \( PL \) which means that the \( a_n \) was quantised to \( |A_n| = r2^k \) \( [12] \). However, it is known from quantisation theory \( [7] \), and in our case confirmed by experiments (Figures \( 5, 6 \)), that quantisation to the middle point of the bin is a better choice (optimal when data are uniformly distributed over a bin). Therefore the value \( |a_n| \in B_{kr} \) is quantised to: \( |A_n| = (r + 2^{-PL})2^k \). Figure \( 6 \) presents the PLQ quantisation with \( PL = 2 \). For colour coding there is only a minor difference between the two approaches (see Figure \( 8 \) for \( L = 2 \)). This is because PLQ quantisation affects only one out of three amplitudes. The idea of PLQ allows grouping the atoms with the same value \( A_n \) and only signal group counts. The quantisation parameter is taken to be \( PL = 2 \) as advised in \( [21] \) and confirmed by our results (see Figure \( 6, 8 \)) that there is only a small difference in PSNR between MP with PLQ and without any quantisation.

3.2 Colour Amplitude Quantisation

The amplitude with maximal value over the three colour channels \( (a_{n,\text{max}}) \) is quantised using PLQ and serves as a base for grouping atoms. The atoms with the same quantised absolute value of maximal amplitude \( |A_{n,\text{max}}| \) 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 uniform scalar quantisation \( [7] \). The values sent to the encoder are \( d_{n,}^i \) (given for \( i = 1, 2 \) by Equation \( 21 \))

\[
Q(a_{n}^i) = \min \left( \text{round} \left( L \frac{|a_{n}^i|}{A_{n,\text{max}}} \right), L \right),
\]

\[
d_{n,}^i = L + 1 - \text{sgn}(a_{n,\text{max}}) \text{sgn}(a_{n}^i)Q(a_{n}^i). \quad (21)
\]

The value of \( L \) has been experimentally chosen to be as low as 2 in order to maximally reduce the number of bits required. The lowest value that does not violate condition from Equation \( 19 \) is \( L = 1 \). However then the error introduced at the quantisation step affects significantly the rate of convergence of the Multi-channel MP (see Figure \( 3 \) when \( L = 1 \)). It has to be noted that the values of signs are included into indexes sent to encoder after quantisation which is shown in Figure \( 7 \).
3.3 Atom Encoding

After MP decomposition and quantisation, the data to be encoded form a matrix in which rows represent the 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. $d^1_n, d^2_n$, quantised amplitude differences  \( d^i_n \in \{1, 2, \ldots, 2L + 1\} \)
3. $c_n$, maximum amplitude colour channel  \( c_n \in \{1, 2, 3\} \)
4. $w_n$, sub-band index  \( w_n \in \{1, 2, \ldots, 3S + 1\} \)
5. $\lambda_n$, 2D dictionary entry  \( \lambda_n \in \{1, \ldots, B\} \)
6. $x_n, y_n$, atom location inside the sub-band \( w_n \) depending on the size of subband \( (22) \)

Atoms from columns 1-6 are encoded group by group using Run Length Encoding (RLE) with the columns ordered in lexicographical order recommended for databases indexes \[8\]. Encoding inside one group is done recursively which is summarised in Algorithm 5.
Atom locations (columns 7-8) are encoded as two values \(x_n\) and \(y_n\) that come from the ranges \(1 \ldots W_{x_n}\) and \(1 \ldots W_{y_n}\), where \(W_{x_n}, W_{y_n}\) are sizes of the sub-band \(w_n\) respectively. The encoding procedure of the sorted sequence \(\{v_{s}\}_{s=1,2,\ldots,K}\) from an alphabet of size \(N\) can be summarised by Algorithm 6. The data sent to the encoder are always mapped into values from 1 to \(N\). For grayscale decompositions there are only 5 columns: \(s_n, w_n, \lambda_n, x_n\) and \(y_n\) from which only 3 are reordered. Figure 10 presents the order in which the data consisting of 10 grayscale atoms are encoded without the permutation of columns.

Algorithm 6 Encoding one column.

\[
\text{function} \quad \text{encode}(m, d, i_{\text{start}}, i_{\text{end}}) \\
\text{inputs:} \\
\quad \text{Matrix } M \text{ of data rows.} \\
\quad \text{depth parameter } d. \\
\quad \text{the first } i_{\text{start}} \text{ and the last row } i_{\text{end}}. \\
\text{body:} \\
\quad \text{encode } d\text{-th column using Algorithm 6} \\
\text{if } d < \text{MAX then} \\
\quad \text{group the same symbols} \\
\quad \text{for all groups } g \text{ do} \\
\quad \quad \text{encode}(M, d + 1, i_{\text{start}}, i_{\text{end}}) \\
\quad \text{end for} \\
\text{else} \\
\quad \text{encode atom positions} \\
\text{end if}
\]

At each iteration we make a decision whether to encode the symbol \(v_s\) directly or to signal its run length. At \(s\)-th iteration the number of symbols remaining to encode is \(s_1 = K - s + 1\) and they can come from the alphabet of size \(a_1\) \((v_s \in \{v_{s-1}, v_{s-1} + 1, \ldots, N\}, \text{ i.e. } v_s \text{ can take any value greater or equal than the previous symbol})\). The values \(s_1\) and \(a_1\) are available for both encoder and decoder at each iteration. Run length coding can be only efficient when the run length is of length at least 2. We estimate the average expected run length as \(s_1/a_1\), i.e. as a ratio of the number of remaining symbols to the size of alphabet. When \(s_1/a_1 > 2\) we encode a count of the next expected symbols which can be 0, otherwise a raw symbol \(v_s\) is encoded.

A final step of encoding is done using arithmetic coding [18] with models that assume uniform distributions of data. Arithmetic coding allows, knowing the probability distribution of data, to achieve compression ratio close to a theoretical bound given by the Shannon’s entropy. Uniform distribution has the highest entropy among discrete distributions. Therefore the results shown here serve as the upper bound for the size of encoded stream.

The atom positions are modelled as uniformly distributed in a range depending on the size of wavelet sub-band in which the atom has been found. The remaining data run lengths \((R\text{ and } z_l\text{ in Algorithm 6})\) are coded as uniformly distributed in a range \(0 \ldots s_l\). The symbols \(v_s\) are modelled as the minimums of the remaining symbols. In our case, there are \(s_l\) symbols \(v_s, v_{s+1}, \ldots, v_K\) to be encoded, then, since they are sorted:

\[v_s = \min\{v_s, v_{s+1}, \ldots, v_K\}. \quad (23)\]
It is well known in theory of probability \cite{16} that if \(v_s, v_{s+1}, \ldots, v_K\) comes from the same distribution with cumulative distribution function (CDF) \(F\) then \(v_s\) given by Equation \ref{eq:23} has CDF: \(1 - (1 - F)^{s_i}\).

For the discrete case of \(s_i\) symbols from the alphabet of size \(a_i\) the probabilities are given as:

\[
p(v_s = k) = (1 - F(k - 1))^{s_i} - (1 - F(k))^{s_i}, \quad \text{for } k = 1 \ldots a_i.
\]

If the symbols are independently drawn from a uniform distribution then \(F(k) = k/a_I\) and the probability distribution for \(v_s\) is given by Equation \ref{eq:25}:

\[
p(v_s = k) = \frac{a_i - k + 1)^{s_i} - (a_i - k)^{s_i}}{a_i^{s_i}}, \quad \text{for } k = 1 \ldots a_i.
\]

Those values are calculated every time a symbol \(v_s\) is to be sent directly to the arithmetic coder and are used to update the model during encoding and decoding.

\[
M = \begin{pmatrix}
  s_n & u_n & \lambda_n & x_n & y_n \\
  1 & 1 & 10 & 18 & 19 \\
  1 & 1 & 25 & 9 & 5 \\
  1 & 1 & 26 & 7 & 19 \\
  1 & 1 & 114 & 8 & 4 \\
  1 & 1 & 165 & 10 & 15 \\
  1 & 1 & 209 & 2 & 3 \\
  2 & 1 & 9 & 9 & 11 \\
  2 & 1 & 139 & 1 & 2 \\
  2 & 1 & 148 & 1 & 15 \\
  2 & 1 & 16 & 1 & 6
\end{pmatrix}
\]

Figure 10: Example of encoding of one sorted group of coefficients.

4 Coding Results

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

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

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 does measure the mathematical properties of the algorithms used. Comparisons with JPEG 2000 are done using the same wavelet filters (CDF 9/7) and the same number of 5 scales. For fair comparison of colour codecs an option of JPEG 2000 which minimises mean squared error (i.e., no weights switch for the Kakadu implementation \cite{17}) was used. In Figures \ref{fig:12}-\ref{fig:34} which show comparison of R-D characteristics measured by PSNR, performance of a default mode of Kakadu is also considered.

4.1 Column Order

The set of experiments for standard test images of different sizes has been done to find an optimal column order to apply Algorithm\cite{6} Each permutation was tried for 12 grayscale (see Table \ref{tab:1}) and colour images (see Table \ref{tab:2}). The differences in the size of a bit-stream for different column orders affect the coding performance significantly. For grayscale, where there are only 6 possible column permutations, the differences between maximum and minimum bit-stream sizes are around 10%. For colour, where we have 720 orders, the differences can exceed 20%. 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 amplitudes are the last sorting criteria for both grayscale and colour. Rather surprisingly, there are column permutations that perform close to optimal for all tested images: \(\pi_g = (2, 3, 1)\) for grayscale and \(\pi_c = (5, 2, 3, 4, 6, 1)\) for colours. The optimal order depends on the method of encoding. Nevertheless experimental results suggest that wavelet scales should be the first criterion while atom indexes and signs the last two.
<table>
<thead>
<tr>
<th>image</th>
<th>the best ($\pi$)</th>
<th>the worst ($\pi$)</th>
<th>$\pi_g = (2, 3, 1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>airplane512x512</td>
<td>101776(2,3,1)</td>
<td>108487(3,1,2)</td>
<td>10176</td>
</tr>
<tr>
<td>baboon512x512</td>
<td>103017(2,1,3)</td>
<td>113788(3,1,2)</td>
<td>104073</td>
</tr>
<tr>
<td>barbara720x576</td>
<td>105699(2,1,3)</td>
<td>115314(3,1,2)</td>
<td>106577</td>
</tr>
<tr>
<td>goldhill720x576</td>
<td>102978(2,3,1)</td>
<td>111453(3,1,2)</td>
<td>102978</td>
</tr>
<tr>
<td>house768x512</td>
<td>105457(2,3,1)</td>
<td>114743(3,1,2)</td>
<td>105457</td>
</tr>
<tr>
<td>lena512x512</td>
<td>102321(2,3,1)</td>
<td>110012(3,1,2)</td>
<td>102321</td>
</tr>
<tr>
<td>lighthouse768x512</td>
<td>104441(2,1,3)</td>
<td>112076(3,1,2)</td>
<td>104905</td>
</tr>
<tr>
<td>motorcross768x512</td>
<td>101065(2,3,1)</td>
<td>108161(3,1,2)</td>
<td>101065</td>
</tr>
<tr>
<td>parrots768x512</td>
<td>107218(2,3,1)</td>
<td>113520(3,1,2)</td>
<td>107218</td>
</tr>
<tr>
<td>peppers512x512</td>
<td>102222(2,3,1)</td>
<td>108971(3,1,2)</td>
<td>102222</td>
</tr>
<tr>
<td>sailboat512x512</td>
<td>99303(2,3,1)</td>
<td>107284(3,1,2)</td>
<td>99303</td>
</tr>
<tr>
<td>sailboats512x768</td>
<td>107438(2,3,1)</td>
<td>116146(3,1,2)</td>
<td>107438</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>image</th>
<th>the best ($\pi$)</th>
<th>the worst ($\pi$)</th>
<th>$\pi_g = (5, 2, 3, 4, 6, 1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>airplane512x512</td>
<td>123815(3,4,2,5,6,1)</td>
<td>142816(1,6,5,2,4,3)</td>
<td>124130</td>
</tr>
<tr>
<td>baboon512x512</td>
<td>124316(5,3,4,2,6,1)</td>
<td>144858(1,6,5,3,2)</td>
<td>124455</td>
</tr>
<tr>
<td>barbara720x576</td>
<td>126937(5,2,3,4,6,1)</td>
<td>148111(6,1,4,5,3,2)</td>
<td>126937</td>
</tr>
<tr>
<td>goldhill720x576</td>
<td>124938(5,3,2,4,6,1)</td>
<td>142009(1,6,4,5,3,2)</td>
<td>124971</td>
</tr>
<tr>
<td>house768x512</td>
<td>125048(2,3,5,4,6,1)</td>
<td>147576(1,6,4,5,3,2)</td>
<td>125081</td>
</tr>
<tr>
<td>lena512x512</td>
<td>127077(5,3,2,4,6,1)</td>
<td>142753(6,1,4,5,2,3)</td>
<td>127113</td>
</tr>
<tr>
<td>lighthouse768x512</td>
<td>121129(5,2,3,4,6,1)</td>
<td>147995(1,6,4,5,3,2)</td>
<td>121129</td>
</tr>
<tr>
<td>motorcross768x512</td>
<td>119399(5,2,3,4,6,1)</td>
<td>141864(6,1,4,5,3,2)</td>
<td>119399</td>
</tr>
<tr>
<td>parrots768x512</td>
<td>130512(5,3,2,6,1,4)</td>
<td>145337(6,1,4,5,3,2)</td>
<td>130739</td>
</tr>
<tr>
<td>peppers512x512</td>
<td>128686(5,3,2,4,6,1)</td>
<td>138515(6,1,4,5,3,2)</td>
<td>128803</td>
</tr>
<tr>
<td>sailboat512x512</td>
<td>123035(5,2,3,4,6,1)</td>
<td>140519(1,6,4,5,3,2)</td>
<td>123035</td>
</tr>
<tr>
<td>sailboats512x768</td>
<td>125867(5,2,3,4,6,1)</td>
<td>147651(6,1,4,5,3,2)</td>
<td>125867</td>
</tr>
</tbody>
</table>

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

### 4.2 Comparison with JPEG 2000

R-D performance graphs are shown in Figures 11-34. In general both grayscale and colour codecs are comparable to JPEG 2000, often outperforming the latter at low bit-rates (see Figures 13-14). However for many standard test images like Parrots (Figures 15-16), House (Figures 21-22) or Sailboats (Figures 33-34) performance is still significantly worse. Tables 3-4 compare performance at given bit-rates. Competitive performance at low bit-rates for grayscale confirms usefulness of MP for low bit-rate coding. For colour images MP can beat the standard for more images and up to higher rate. On average (see Tables 3-4) MP beats the standard at 0.1 bpp. However the performance is worse for higher rates.

Modelling distributions of wavelet scale indexes and run lengths for arithmetic coding should improve coding performance. For example, as mentioned in Section 2.3 in initial iterations the atoms are more likely to be found in low frequencies. The presented codec assigns equal probabilities to each symbol in each column. Since the uniform distribution has the highest entropy among discrete distributions, the results shown in this report serve as an upper bound for the size of a bit-stream. Both encoding and decoding 8000 colour atoms into bit-stream which corresponds to a bit-rate of around 0.40 bpp for the Goldhill image take less than 0.2 s. This is negligible when compared to finding these atoms by the MP algorithm which takes more than 40 seconds.

### 5 Conclusions and Future Work

We have presented a novel approach for decomposing and encoding images that has shown comparable R-D performance to the current coding standard JPEG 2000. Our idea of encoding atoms seems to be especially promising for colour images. MP is performed after a discrete wavelet transform to reduce complexity and improve sparsity [21]. MP decomposition of an image is represented as a matrix of rows. These rows are sorted in lexicographical order after permutation of column and encoded using run length and then arithmetic coding with a 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 investigation include: more sophisticated data modelling for
<table>
<thead>
<tr>
<th>bit-rate:</th>
<th>0.1 bpp</th>
<th>0.3 bpp</th>
<th>0.5 bpp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>J2K</td>
<td>MP-Gray</td>
<td>J2K</td>
</tr>
<tr>
<td>airplane512x512</td>
<td>28.13</td>
<td>28.06</td>
<td>33.82</td>
</tr>
<tr>
<td>baboon512x512</td>
<td>21.32</td>
<td>21.57</td>
<td>23.57</td>
</tr>
<tr>
<td>barbara720x576</td>
<td>25.21</td>
<td>26.02</td>
<td>30.21</td>
</tr>
<tr>
<td>goldhill720x576</td>
<td>28.90</td>
<td>29.12</td>
<td>32.30</td>
</tr>
<tr>
<td>house768x512</td>
<td>23.93</td>
<td>24.13</td>
<td>27.37</td>
</tr>
<tr>
<td>lena512x512</td>
<td>29.90</td>
<td>29.83</td>
<td>34.94</td>
</tr>
<tr>
<td>lighthouse768x512</td>
<td>25.77</td>
<td>25.81</td>
<td>29.57</td>
</tr>
<tr>
<td>motorcross768x512</td>
<td>21.74</td>
<td>21.80</td>
<td>25.06</td>
</tr>
<tr>
<td>peppers512x512</td>
<td>33.62</td>
<td>33.43</td>
<td>39.04</td>
</tr>
<tr>
<td>sailboat512x512</td>
<td>29.66</td>
<td>29.44</td>
<td>34.16</td>
</tr>
<tr>
<td>sailboats512x768</td>
<td>30.08</td>
<td>30.13</td>
<td>35.36</td>
</tr>
<tr>
<td>Average</td>
<td>26.95</td>
<td>27.05</td>
<td>31.24</td>
</tr>
</tbody>
</table>

Table 3: Coding performance at given bit-rates for grayscale images.

<table>
<thead>
<tr>
<th>bit-rate:</th>
<th>0.1 bpp</th>
<th>0.3 bpp</th>
<th>0.5 bpp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>J2K</td>
<td>MP-RGB</td>
<td>J2K</td>
</tr>
<tr>
<td>airplane512x512</td>
<td>26.37</td>
<td>26.87</td>
<td>31.05</td>
</tr>
<tr>
<td>baboon512x512</td>
<td>20.01</td>
<td>20.24</td>
<td>21.87</td>
</tr>
<tr>
<td>barbara720x576</td>
<td>23.89</td>
<td>24.23</td>
<td>28.03</td>
</tr>
<tr>
<td>goldhill720x576</td>
<td>27.24</td>
<td>27.22</td>
<td>29.93</td>
</tr>
<tr>
<td>house768x512</td>
<td>23.32</td>
<td>23.19</td>
<td>26.42</td>
</tr>
<tr>
<td>lena512x512</td>
<td>27.68</td>
<td>27.64</td>
<td>31.31</td>
</tr>
<tr>
<td>lighthouse768x512</td>
<td>25.18</td>
<td>25.00</td>
<td>28.68</td>
</tr>
<tr>
<td>motorcross768x512</td>
<td>21.15</td>
<td>20.91</td>
<td>24.23</td>
</tr>
<tr>
<td>peppers512x512</td>
<td>30.72</td>
<td>30.40</td>
<td>35.92</td>
</tr>
<tr>
<td>sailboat512x512</td>
<td>25.57</td>
<td>25.80</td>
<td>29.61</td>
</tr>
<tr>
<td>sailboats512x768</td>
<td>22.67</td>
<td>23.14</td>
<td>25.89</td>
</tr>
<tr>
<td>Average</td>
<td>25.24</td>
<td>25.28</td>
<td>28.91</td>
</tr>
</tbody>
</table>

Table 4: Coding performance comparisons against JPEG 2000 with no-weights option at given bit-rates for colour images.

coding and finding the optimal dictionary. We believe that, when restricted to short separable filters applied in transform domain, the choice of the encoding method is more critical for the final size of a bit-stream than selection of a particular dictionary basis. On the other hand the structure of a dictionary is critical for reducing computational complexity.

References


(y-axis: PNSR [dB], x-axis: bit-rate [bpp]).
images: Lena, Barbara, Parrots

Figure 11: Grayscale Lena, 512 × 512
Figure 12: Colour Lena, 512 × 512
Figure 13: Grayscale Barbara, 720 × 576
Figure 14: Colour Barbara, 720 × 576
Figure 15: Grayscale Parrots, 768 × 512
Figure 16: Colour Parrots, 768 × 512
(y-axis: PNSR [dB], x-axis: bit-rate [bpp]).
images: Baboon, Goldhill, House

Figure 17: Grayscale Baboon, 512 × 512
Figure 18: Colour Baboon, 512 × 512
Figure 19: Grayscale Goldhill, 720 × 576
Figure 20: Colour Goldhill, 720 × 576
Figure 21: Grayscale House, 768 × 512
Figure 22: Colour House, 768 × 512
RD-performance comparisons between proposed MP coding and JPEG 2000. (y-axis: PNSR [dB], x-axis: bit-rate [bpp]).

images: Airplane, Lighthouse, Motorcross

Figure 23: Grayscale Airplane, 512 × 512

Figure 24: Colour Airplane, 512 × 512

Figure 25: Grayscale Lighthouse, 768 × 512

Figure 26: Colour Lighthouse, 768 × 512

Figure 27: Grayscale Motorcross, 768 × 512

Figure 28: Colour Motorcross, 768 × 512
(y-axis: PNSR [dB], x-axis: bit-rate [bpp]).
images: Peppers, Sailboat, Sailboats