Aston University
School of Engineering and Applied Science

Technical Report

Grayscale and Colour Image Codec based on Matching Pursuit in the Spatio-Frequency Domain.

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

Grayscale and Colour Image Codec based on Matching Pursuit in the Spatio-Frequency Domain. [1]

Ryszard Maciol [2], 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 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 transformed-based image coding. Moreover they provide new options to exploit inter-channel redundancies 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 $\mathcal{H}$. 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 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 $\mathcal{H}$ 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 $g_{\gamma_n} \in \mathcal{D}$ 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

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: $Rf_1^r = f^r, Rf_1^g = f^g, Rf_1^b = f^b$.
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

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.

$$\gamma_n = \max_{\gamma \in \Gamma} \max_{r,g,b} \left(|\langle Rf_n^r, g_\gamma \rangle|, |\langle Rf_n^g, g_\gamma \rangle|, |\langle Rf_n^b, g_\gamma \rangle| \right). \tag{2}$$

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

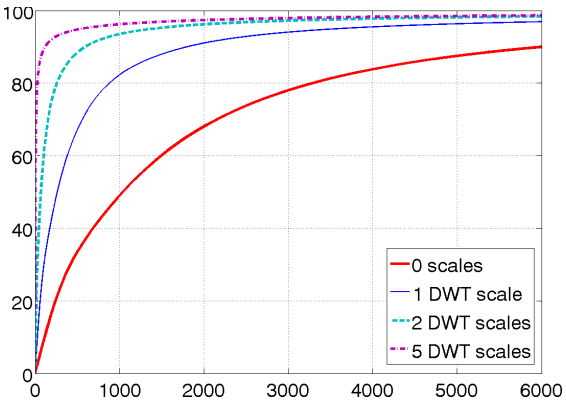

Figure 1: Percentage of the image energy (y-axis) represented by a given number of atoms (x-axis) using different numbers of wavelet scales (grayscale Goldhill).

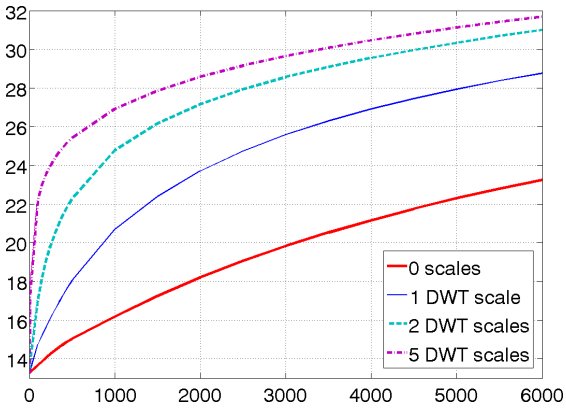

Figure 2: PSNR performance in dB (y-axis) for a given number of atoms (x-axis) using different numbers of wavelet scales (grayscale Goldhill).

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,

$$\langle f, g\rangle = \langle T\{f\}, T\{g\}\rangle \text{ for all } f, g \in \mathcal{H}, \tag{3}$$

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

$$T\{f\} \approx \sum_{n=1}^{N} \langle T\{Rf_n\}, T\{g_{\gamma_n}\}\rangle\, T\{g_{\gamma_n}\}. \tag{4}$$

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:

$$\mathcal{D}^{(non-sep)} = \{G_1, G_2, \ldots, G_B\}\,. \tag{5}$$

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:

$$\mathcal{D}^{(sep)} = \{g_i \otimes g_j\}_{i,j=1,\ldots,b}\,. \tag{6}$$

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 $\mathcal{D}_c$ and grayscale $\mathcal{D}_g$ coding separately. We initialised the both dictionaries with 2 bases: $\mathcal{D}_*^2 = \{g_1, g_2\}$, 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:

$$g_{(\sigma,f,w,\phi)}(t) = \exp\left(-\frac{\pi}{4\sigma}t^2\right)\cos\left(\frac{\pi f t}{w} + \phi\right), \tag{7}$$

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 σ, f, ϕ are taken, following [11], to be: $\sigma \in \{1, 2, 4, 8, 12, 16, 20, 24\}$, $f \in \{0, 1, \ldots, w\}$, $\phi \in \{0, \pi/8, \pi/4, 3\pi/8, \pi/2\}$. 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 720×576. 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 and for each wavelet subband. 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. $w_i \leqslant w_{i+1}$ 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 $T^{(non-sep)}$ multiply-accumulate operations:

$$T^{(non-sep)} = NM\left(\sum_{i=1}^{b} w_i\right)^2. \tag{8}$$

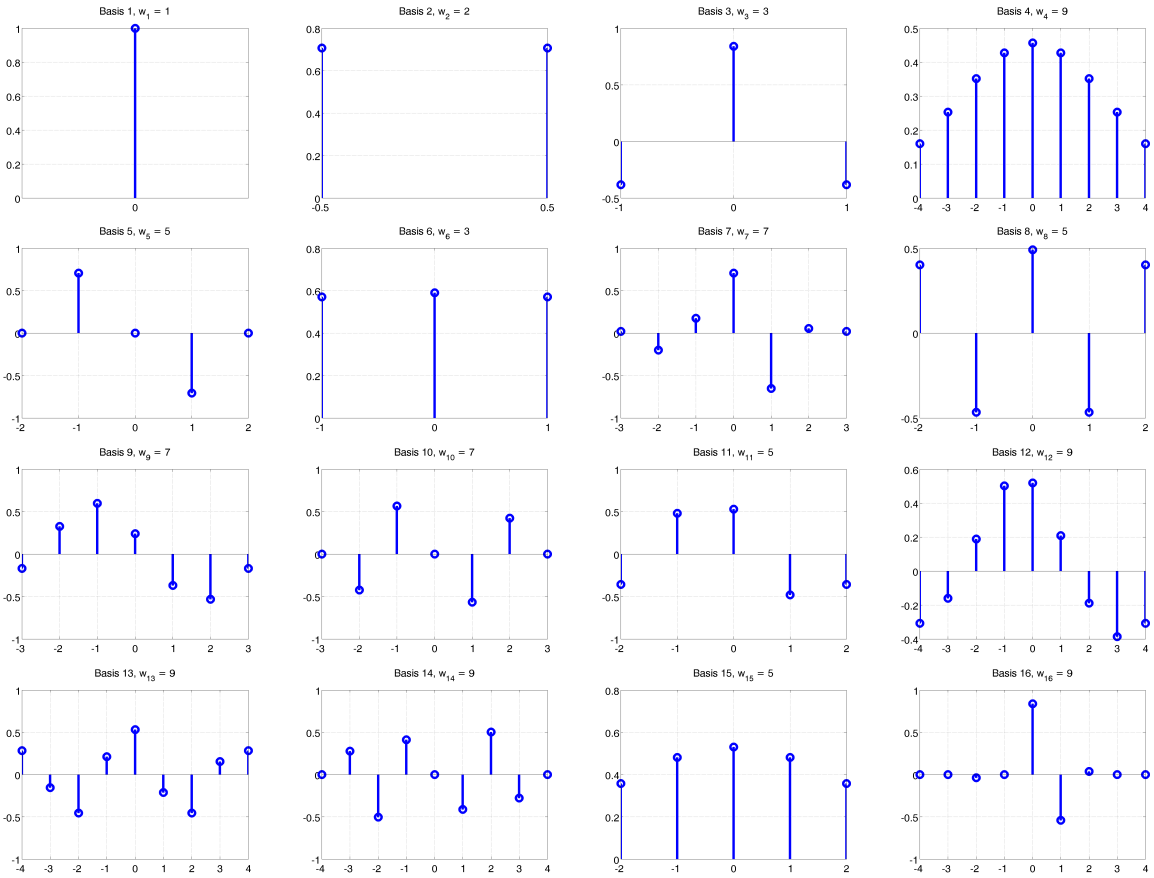

Figure 3: Colour dictionary trained on RGB Goldhill 720×576 image with bases in order they are picked during training.

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

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

Algorithm 3 Inner products calculation from [14].

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 3). Usefullness 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.

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

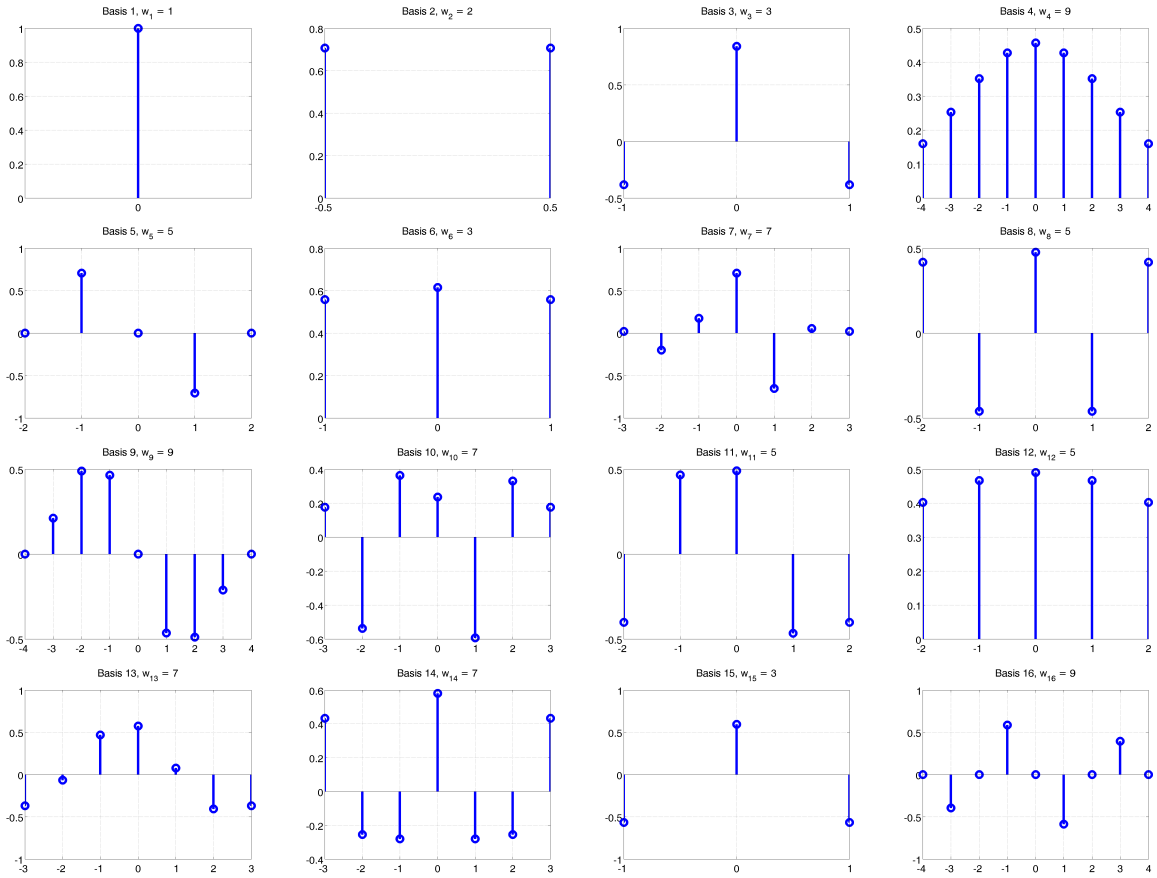

Figure 4: Grayscale dictionary trained on Y-channel of Goldhill 720×576 image with bases ordered in order they are picked during training.

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:

$$T^{(sep)*} = NM\left(3S + 2F\right). \tag{10}$$

For the colour dictionary $\mathcal{D}_c$ 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 $\mathcal{D}_g$ (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:

$$T^{(sep)} = T_{in}^{(sep)} + \sum_{n=1}^{K} \left(T_{update_n}^{(sep)} + T_{search_n}^{(sep)} \right). \tag{11}$$

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.

$$T_{search_n}^{(sep)} = O(NM). \tag{12}$$

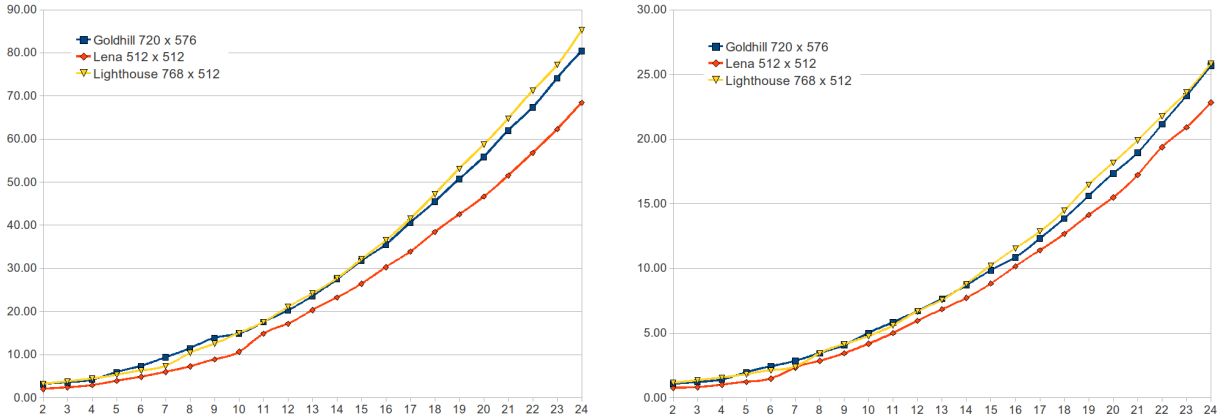

Figure 5: Increase in complexity during the process of adding functions picked by Basis Picking [11] to a dictionary. Colour decomposition (left), grayscale (right), time in seconds on Linux PC with Intel Core 2 Duo (y-axis) and number of 1D basis in a dictionary (x-axis).

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

$$\begin{aligned} W_n &= w_n + 2 \max_i w_i - 2 \leqslant 3W, \\ H_n &= h_n + 2 \max_i w_i - 2 \leqslant 3W, \end{aligned} \tag{13}$$

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 $\mathcal{D}$. By combining Equations 10-12:

$$\begin{aligned} T^{(sep)} &= NM(3S + 2F) + \\ &+ \sum_{n=1}^{K} \big(W^2(3S + 2F) + O(NM)\big). \end{aligned} \tag{14}$$

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

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

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

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

Equations 14-15 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 in 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 720×576 image at least as fast as 768×512 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 720×576 (see Figure 5).

3 Quantisation and Atom Coding

For data compression applications MP decomposition has to be encoded into a bit-stream. The values $a_n = \langle Rf_n, g_{\gamma_n} \rangle$ 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]:

$$||f||^2 = \sum_{n=1}^{N} a_n^2 + ||Rf_{N+1}||^2. \tag{17}$$

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:

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

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]:

$$0 < A_n < 2a_n. \tag{19}$$

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:

$$\begin{aligned} \text{bit-planes:} \quad & k = M, M-1, \ldots, 1, 0, -1, \ldots \\ \text{refinements:} \quad & r = 1, 1 + 2^{1-PL}, \ldots, 2 - 2^{1-PL} \\ \text{quantisation bins:} \quad & B_{kr} = \left[r2^k, (r + 2^{1-PL})2^k \right). \end{aligned} \tag{20}$$

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 8-9), 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 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^{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 uniform scalar quantisation [7]. The values sent to the encoder are d_n^i (given for $i = 1, 2$ by Equation 21).

$$\begin{aligned} Q(a_n^i) &= \min\left(\text{round}\left(L \frac{|a_n^i|}{|A_n^{max}|} \right), L \right) \\ d_n^i &= L + 1 - \text{sgn}(a_n^{max}) \, \text{sgn}(a_n^i) Q(a_n^i). \end{aligned} \tag{21}$$

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 9, 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.

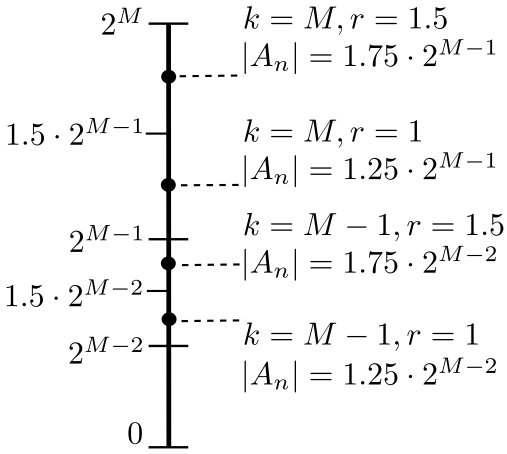

Figure 6: Precision Limit Quantisation with parameter $PL = 2$ (bitplanes M and $M-1$ shown).

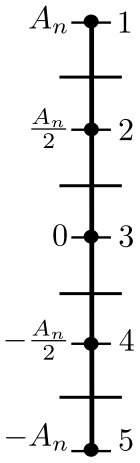

Figure 7: Uniform Quantisation with parameter $L = 2$ (5 quantisation bins indexed from 1 to 5).

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	$s_n \in \{-1, 1\}$
$2-3$	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	$\lambda_n \in \{1, \ldots, B\}$
$7-8$	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.

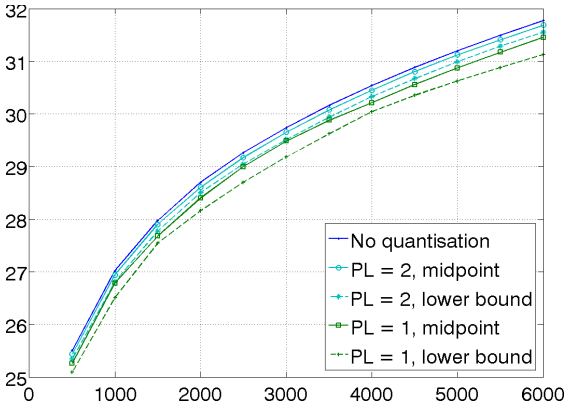

Figure 8: Effect of in-loop quantisation when decomposing grayscale Goldhill (5 scales, dictionary $\mathcal{D}_g$).

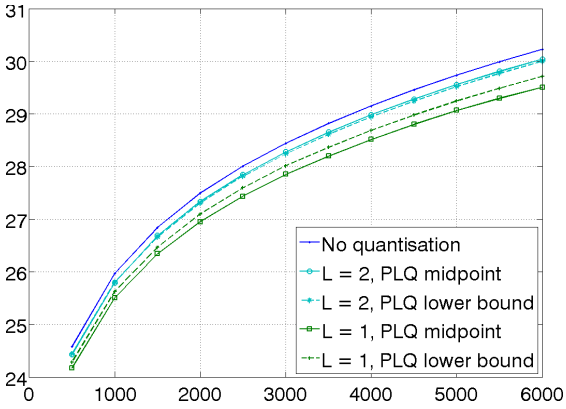

Figure 9: Effect of in-loop quantisation when decomposing colour Goldhill (5 scales, $PL = 2$, dictionary $\mathcal{D}_c$).

Algorithm 5 Encoding MP decomposition.

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

Atom locations (columns 7-8) are encoded as two values x_n and y_n that come from the ranges $1 \dots W_{x_n}$ and $1 \dots W_{y_n}$, where W_{x_n}, W_{y_n} are sizes of the sub-band w_n respectively. The encoding procedure of the sorted sequence $\{v_s\}_{s=1,2,\dots,K}$ 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,λ_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.

input: $\{v_s\}_{s=1,2,\dots,K}$ with $s < s' \Rightarrow v_s \leq v_{s'}$
$s_l = K$ *symbols remaining*
$a_l = N$ *alphabet entries remaining*
while $s_l > 0$ and $a_l > 1$ **do**
if $s_l > 2a_l$ **then**
encode z_l (if any) zero lengths assuming range $0 \dots s_l$
encode run of length R in range $0 \dots s_l$
$s_l = s_l - R$
$a_l = a_l - 1 - z_l$
else
encode symbol v_s in range $1 \dots a_l$
$s_l = s_l - 1$
$a_l = N - a_s$
end if
end while

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_l ($s_l = K - s + 1$) and they can come from the alphabet of size a_l ($v_s \in \{v_{s-1}, v_{s-1}+1, \dots, N\}$, i. e. v_s can take any value greater or equal than the previous symbol). The values s_l and a_l 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_l/a_l, i. e. as a ratio of the number of remaining symbols to the size of alphabet. When $s_l/a_l > 2$ 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 and z_l in Algorithm 6) are coded as uniformly distributed in a range $0 \dots s_l$. The symbols v_s are modelled as the minimums of the remaining symbols. In our case, there are s_l symbols $v_s, v_{s+1}, \dots, v_K$ to be encoded, then, since they are sorted:

$$v_s = \min\{v_s, v_{s+1}, \dots, v_K\}. \tag{23}$$

It is well known in theory of probability [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 23 has CDF: $1-(1-F)^{s_l}$. For the discrete case of s_l symbols from the alphabet of size a_l the probabilities are given as:

$$p(v_s = k) = (1 - F(k-1))^{s_l} - (1 - F(k))^{s_l}, \text{ for } k = 1 \ldots a_l. \tag{24}$$

If the symbols are independently drawn from a uniform distribution then $F(k) = k/a_l$ and the probability distribution for v_s is given by Equation 25:

$$p(v_s = k) = \frac{(a_l - k + 1)^{s_l} - (a_l - k)^{s_l}}{{a_l}^{s_l}}, \text{ for } k = 1 \ldots a_l. \tag{25}$$

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.

$$M = \begin{array}{c} \begin{array}{ccccc} s_n & w_n & \lambda_n & x_n & y_n \end{array} \\ \left(\begin{array}{ccccc} 1 & 1 & 10 & 18 & 19 \\ 1 & 1 & 25 & 9 & 5 \\ 1 & 1 & 26 & 7 & 19 \\ 1 & 1 & 114 & 8 & 4 \\ 1 & 1 & 165 & 10 & 15 \\ 1 & 1 & 209 & 2 & 3 \\ 2 & 1 & 9 & 9 & 11 \\ 2 & 1 & 139 & 1 & 2 \\ 2 & 1 & 148 & 1 & 15 \\ 2 & 5 & 16 & 1 & 6 \end{array} \right) \end{array}$$

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:

$$PSNR = 10 \log_{10} \left(\frac{3 \cdot 255^2}{MSE_r + MSE_g + MSE_b} \right), \tag{26}$$

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 [17]) was used. In Figures 12-34, 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 6. Each permutation was tried for 12 grayscale (see Table 1) and colour images (see Table 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.

image	the best (π)	the worst (π)	$\pi_g = (2,3,1)$
airplane512x512	101776(2,3,1)	108487(3,1,2)	101776
baboon512x512	103017(2,1,3)	113788(1,3,2)	104073
barbara720x576	105699(2,1,3)	115314(1,3,2)	106577
goldhill720x576	102978(2,3,1)	111453(3,1,2)	102978
house768x512	105457(2,3,1)	114743(1,3,2)	105457
lena512x512	102321(2,3,1)	110012(3,1,2)	102321
lighthouse768x512	104441(2,1,3)	112076(1,3,2)	104905
motorcross768x512	101065(2,3,1)	108161(1,3,2)	101065
parrots768x512	107218(2,3,1)	113520(3,1,2)	107218
peppers512x512	102222(2,3,1)	108971(3,1,2)	102222
sailboat512x512	99303(2,3,1)	107284(3,1,2)	99303
sailboats512x768	107438(2,3,1)	116146(3,1,2)	107438

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

image	the best (π)	the worst (π)	$\pi_c = (5,2,3,4,6,1)$
airplane512x512	123815(3,4,2,5,6,1)	142816(1,6,5,2,4,3)	124130
baboon512x512	124316(5,3,4,2,6,1)	144858(1,6,4,5,3,2)	124455
barbara720x576	126937(5,2,3,4,6,1)	148111(6,1,4,5,3,2)	126937
goldhill720x576	124938(5,3,2,4,6,1)	142009(1,6,4,5,3,2)	124971
house768x512	125048(2,3,5,4,6,1)	147576(1,6,4,5,2,3)	125081
lena512x512	127077(5,3,2,4,6,1)	142753(6,1,4,5,2,3)	127113
lighthouse768x512	121129(5,2,3,4,6,1)	147995(1,6,4,5,3,2)	121129
motorcross768x512	119399(5,2,3,4,6,1)	141864(6,1,5,4,3,2)	119399
parrots768x512	130512(5,3,2,6,1,4)	145187(6,4,1,5,3,2)	130739
peppers512x512	128686(5,3,2,4,6,1)	138515(6,1,4,5,2,3)	128803
sailboat512x512	123035(5,2,3,4,6,1)	140519(1,6,4,5,3,2)	123035
sailboats512x768	125867(5,2,3,4,6,1)	147651(6,1,4,5,3,2)	125867

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

bit-rate:	0.1 bpp		0.3 bpp		0.5 bpp	
image	J2K	MP-Gray	J2K	MP-Gray	J2K	MP-Gray
airplane512x512	**28.13**	28.06	**33.82**	33.23	**36.78**	36.04
baboon512x512	21.32	**21.57**	23.57	**23.83**	**25.56**	25.48
barbara720x576	25.21	**26.02**	30.21	**30.44**	**33.35**	33.09
goldhill720x576	28.90	**29.12**	32.30	**32.38**	**34.25**	34.11
house768x512	23.93	**24.13**	**27.37**	27.30	**29.71**	29.33
lena512x512	**29.90**	29.83	**34.94**	34.58	**37.32**	36.85
lighthouse768x512	25.77	**25.81**	**29.57**	29.28	**32.11**	31.49
motorcross768x512	21.74	**21.80**	25.06	**25.11**	**27.47**	27.16
parrots768x512	**33.62**	33.43	**39.04**	38.43	**41.61**	40.95
peppers512x512	**29.66**	29.44	**34.16**	33.74	**35.84**	35.46
sailboat512x512	25.12	**25.23**	**29.49**	29.20	**31.59**	31.24
sailboats512x768	30.08	**30.13**	**35.36**	34.92	**38.50**	37.57
Average	26.95	**27.05**	**31.24**	31.04	**33.67**	33.23

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

bit-rate:	0.1 bpp		0.3 bpp		0.5 bpp	
image	J2K	MP-RGB	J2K	MP-RGB	J2K	MP-RGB
airplane512x512	26.37	**26.87**	31.05	**31.24**	33.40	**33.44**
baboon512x512	20.01	**20.24**	21.87	**22.02**	23.07	**23.16**
barbara720x576	23.89	**24.23**	28.03	**28.04**	**30.33**	30.20
goldhill720x576	**27.24**	27.22	29.93	**29.95**	**31.46**	31.38
house768x512	**23.32**	23.19	**26.42**	26.02	**28.49**	27.83
lena512x512	**27.68**	27.64	**31.31**	31.25	**32.97**	32.95
lighthouse768x512	**25.18**	25.00	**28.68**	28.18	**30.95**	30.19
motorcross768x512	**21.15**	20.91	**24.23**	23.87	**26.27**	25.77
parrots768x512	**30.72**	30.40	**35.92**	35.23	**38.54**	37.73
peppers512x512	25.57	**25.80**	29.61	**29.68**	31.17	**31.24**
sailboat512x512	22.67	**23.14**	25.89	**26.21**	27.38	**27.74**
sailboats512x768	**29.12**	28.76	**33.95**	33.09	**36.65**	35.55
Average	25.24	**25.28**	**28.91**	28.73	**30.89**	30.60

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

[1] F. Bergeaud and S. Mallat. Matching pursuit of images. In *Proc. International Conference on Image Processing*, volume 1, pages 53–56, 1995.

[2] C. Christopoulos, A. Skodras, and T. Ebrahimi. The JPEG 2000 still image compression standard. *IEEE Signal Processing Magazine*, 18(5):36–58, 2001.

[3] P. Czerepinski, C. Davies, N. Canagarajah, and D. Bull. Matching pursuits video coding: Dictionaries and fast implementation. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(7):1103–1115, 2000.

[4] R. M Figueras i Ventura, P. Vandergheynst, and P. Frossard. Low-rate and flexible image coding with redundant representations. *IEEE Transactions on Image Processing*, 15(3):726–739, 2006.

[5] R. M. Figueras i Ventura, P. Vandergheynst, P. Frossard, and A. Cavallaro. Color image scalable coding with matching pursuit. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages 53–56, 2004.

[6] E. Gershikov, E. Lavi-Burlak, and M. Porat. Correlation-based approach to color image compression. *Image Communications*, 22(9):719–733, 2007.

[7] A. M. Gray and R. Gersho. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, fifth edition, 1992.

[8] D. Lemire and O. Kaser. Reordering columns for smaller indexes. *Information Sciences*, 181(12):2550–2570, 2011.

[9] A. Lutoborski and V. M. Temlyakov. Vector greedy algorithms. *Journal of Complexity*, 19(4):458–473, 2003.

[10] S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

[11] D. M. Monro. Basis picking for matching pursuits image coding. In *Proc. International Conference on Image Processing*, volume 4, pages 2495–2498, 2004.

[12] D. M. Monro and W. Poh. Improved coding of atoms in matching pursuits. In *Proc. International Conference on Image Processing*, volume 3, pages 759–62, 2003.

[13] D. M. Monro and Y. Yuan. Bases for low complexity matching pursuits image coding. In *Proc. IEEE International Conference on Image Processing*, volume 2, pages 249–252, 2005.

[14] R. Neff and A. Zakhor. 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] R. Neff and A. Zakhor. Modulus quantization for matching-pursuit video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(6):895–912, 2000.

[16] M. Siotani. Order statistics for discrete case with a numerical application to the binomial distribution. *Annals of the Institute of Statistical Mathematics*, 8:95–104, 1956. 10.1007/BF02863574.

[17] D. Taubman. Kakadu JPEG 2000 implementation: http://www.kakadusoftware.com/.

[18] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

[19] Y. Yuan, A. N. Evans, and D. M. Monro. Low complexity separable matching pursuits [video coding applications]. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 3, pages 725–728, 2004.

[20] Y. Yuan and D. M. Monro. 3D wavelet video coding with replicated matching pursuits. In *Proc. IEEE International Conference on Image Processing*, volume 1, pages 69–72, 2005.

[21] Y. Yuan and D. M. Monro. Improved matching pursuits image coding. In *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 201–204, 2005.

RD-performance comparisons between proposed MP coding and JPEG 2000.
(y-axis: PNSR [dB], x-axis: bit-rate [bpp]).
images: *Lena*, *Barbara*, *Parrots*

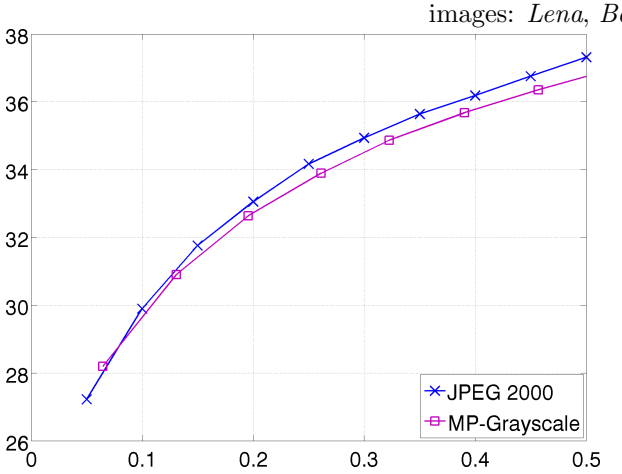

Figure 11: Grayscale Lena, 512 × 512

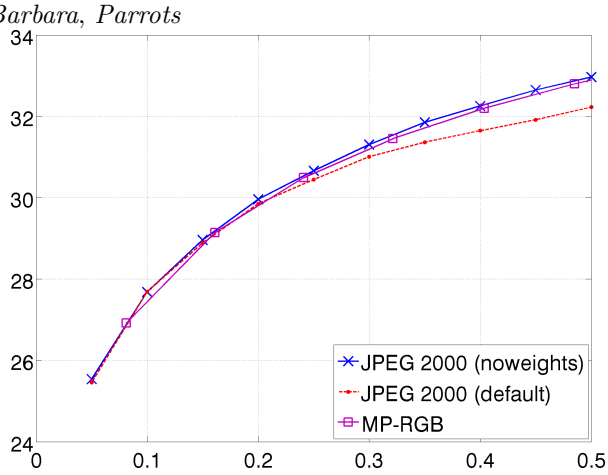

Figure 12: Colour Lena, 512 × 512

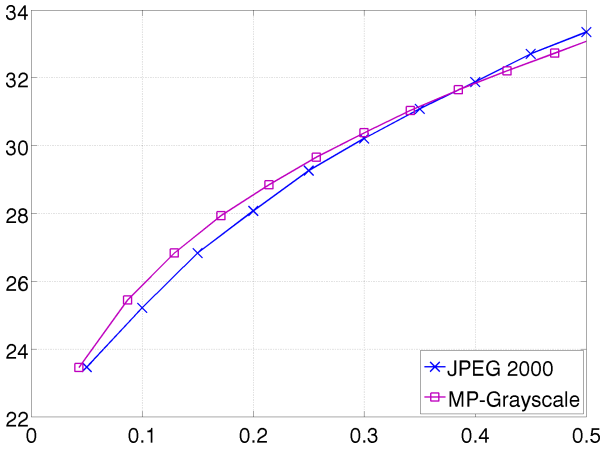

Figure 13: Grayscale Barbara, 720 × 576

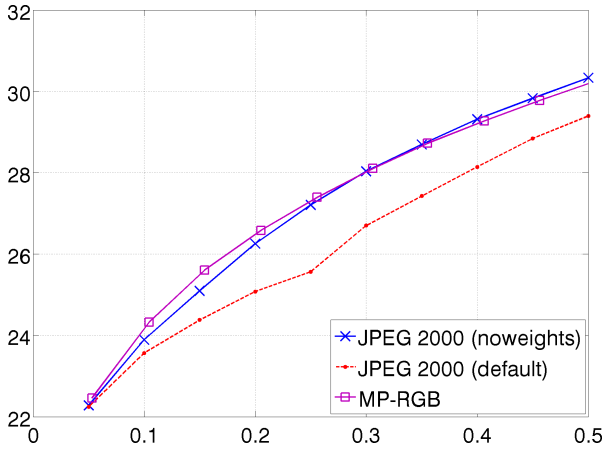

Figure 14: Colour Barbara, 720 × 576

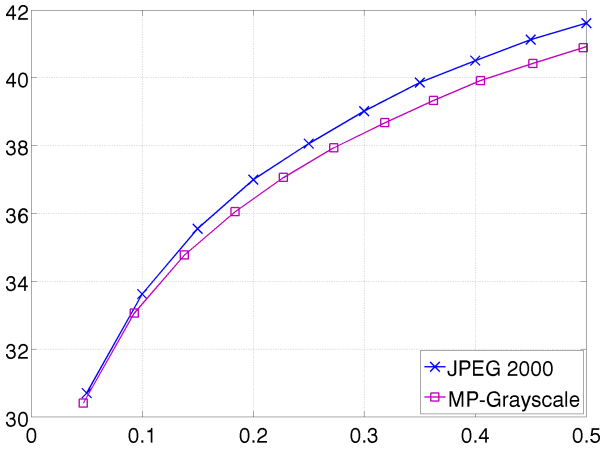

Figure 15: Grayscale Parrots, 768 × 512

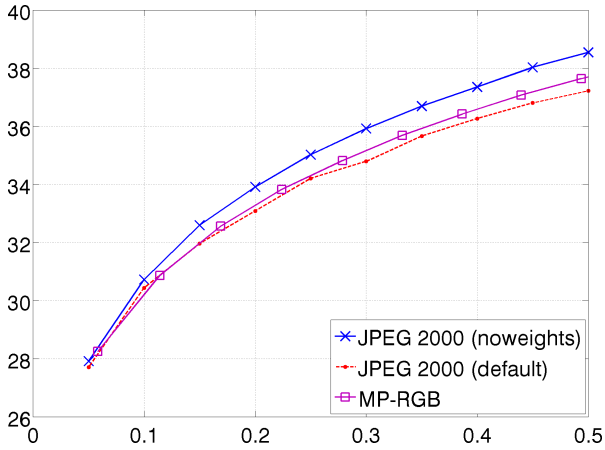

Figure 16: Colour Parrots, 768 × 512

RD-performance comparisons between proposed MP coding and JPEG 2000.
(y-axis: PNSR [dB], x-axis: bit-rate [bpp]).
images: *Baboon*, *Goldhill*, *House*

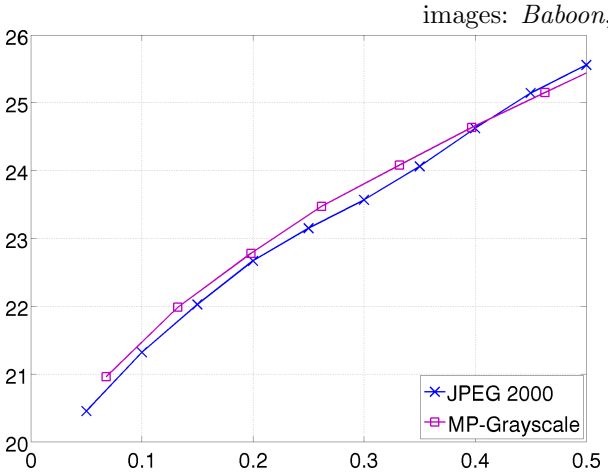

Figure 17: Grayscale Baboon, 512 × 512

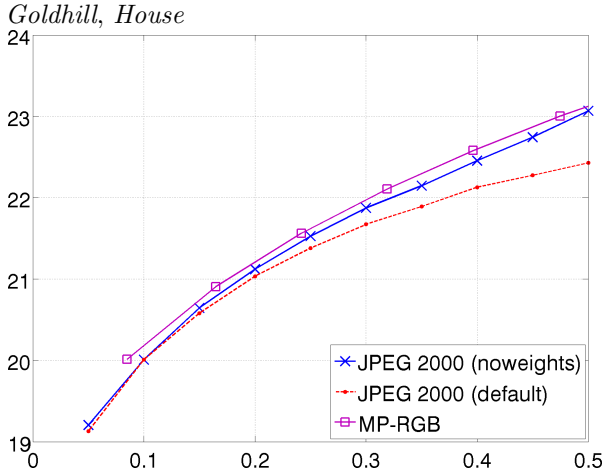

Figure 18: Colour Baboon, 512 × 512

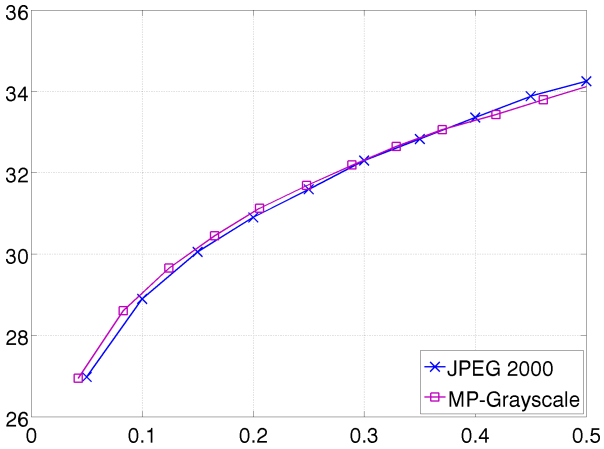

Figure 19: Grayscale Goldhill, 720 × 576

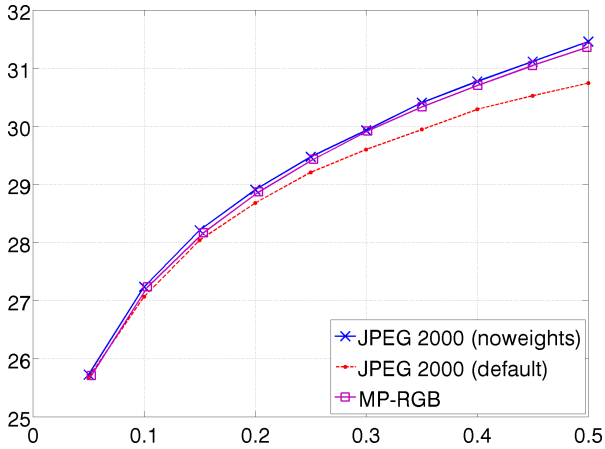

Figure 20: Colour Goldhill, 720 × 576

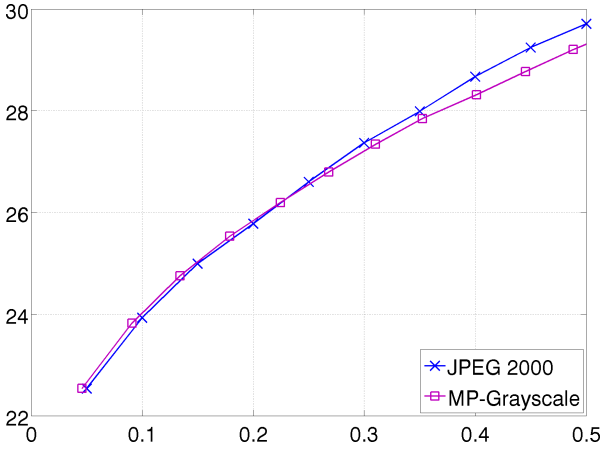

Figure 21: Grayscale House, 768 × 512

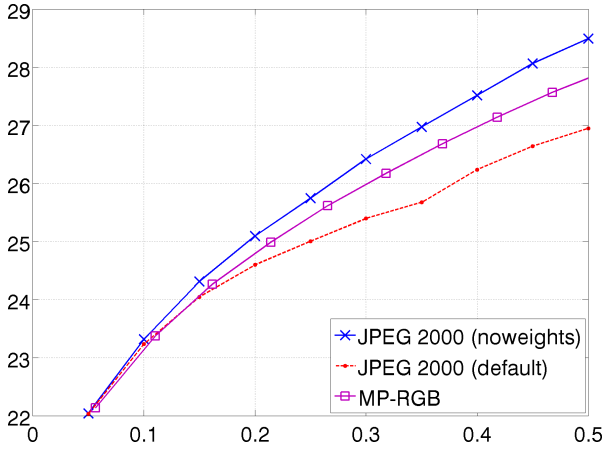

Figure 22: Colour House, 768 × 512

RD-performance comparisons between proposed MP coding and JPEG 2000.
(y-axis: PNSR [dB], x-axis: bit-rate [bpp]).
images: *Airplane*, *Lighthouse*, *Motorcross*

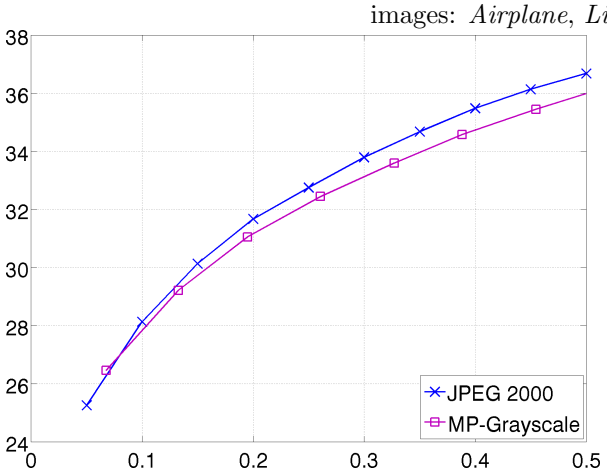

Figure 23: Grayscale Airplane, 512 × 512

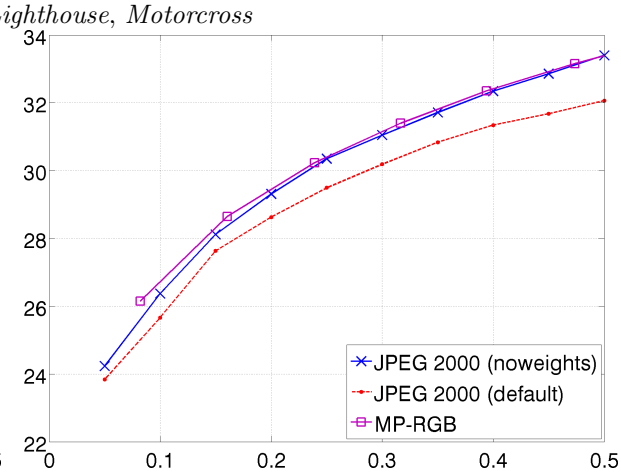

Figure 24: Colour Airplane, 512 × 512

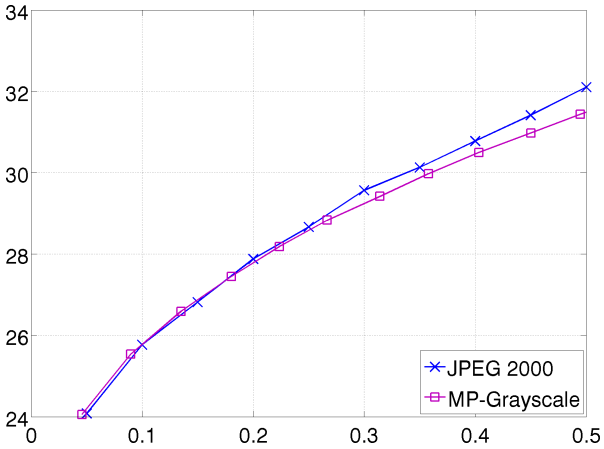

Figure 25: Grayscale Lighthouse, 768 × 512

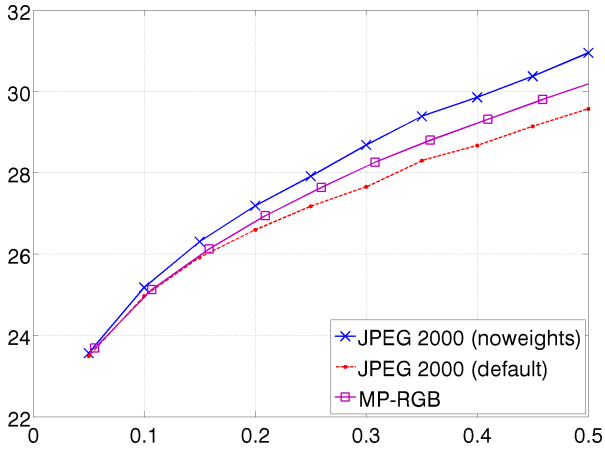

Figure 26: Colour Lighthouse, 768 × 512

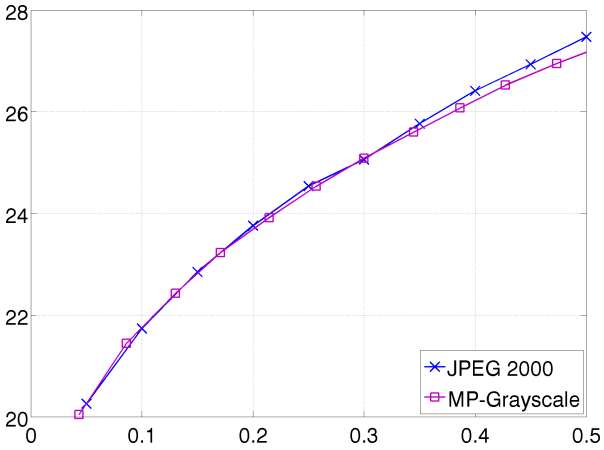

Figure 27: Grayscale Motorcross, 768 × 512

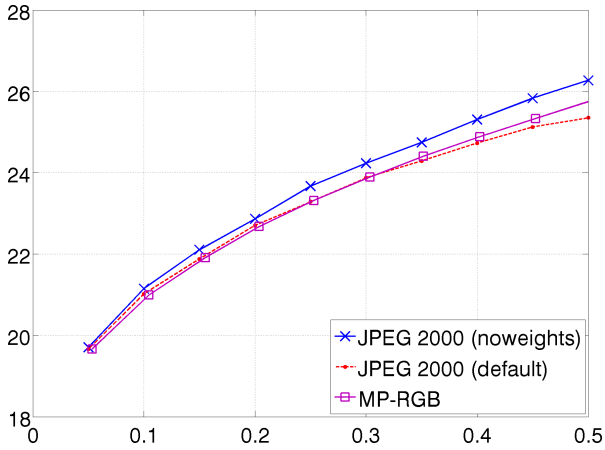

Figure 28: Colour Motorcross, 768 × 512

RD-performance comparisons between proposed MP coding and JPEG 2000.
(y-axis: PNSR [dB], x-axis: bit-rate [bpp]).
images: *Peppers*, *Sailboat*, *Sailboats*

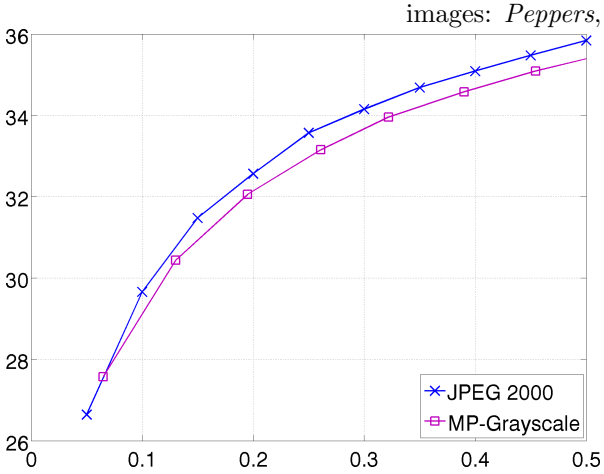

Figure 29: Grayscale Peppers, 512×512

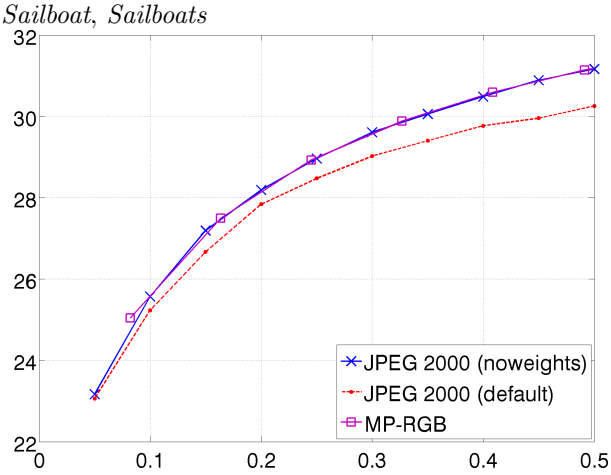

Figure 30: Colour Peppers, 512×512

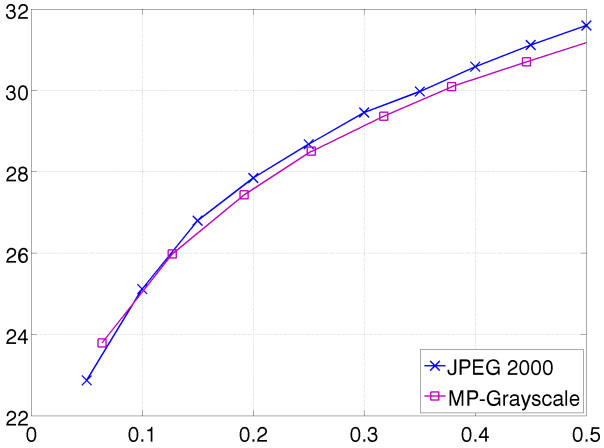

Figure 31: Grayscale Sailboat, 512×512

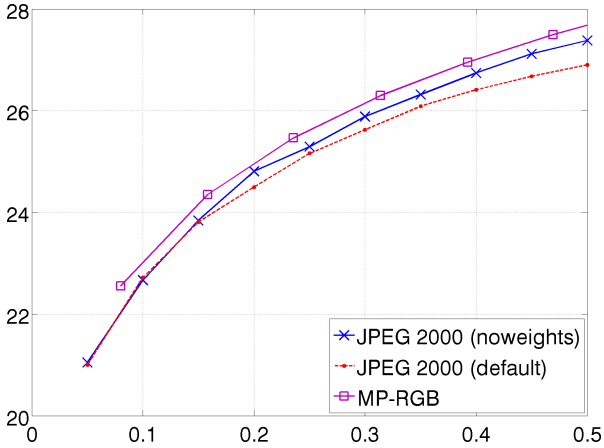

Figure 32: Colour Sailboat, 512×512

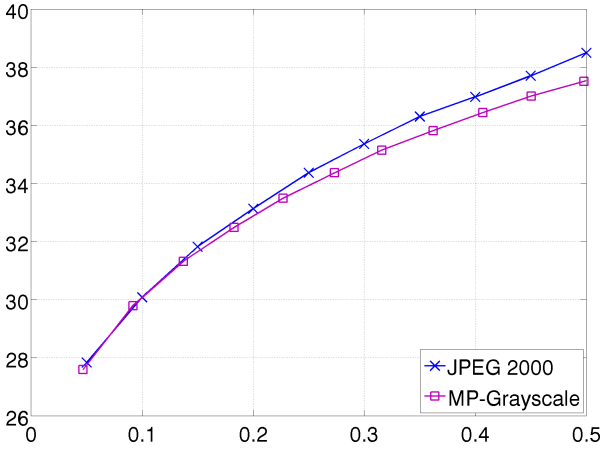

Figure 33: Grayscale Sailboats, 512×768

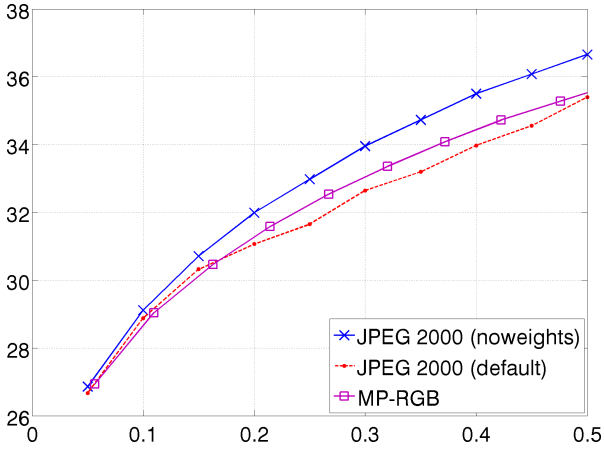

Figure 34: Colour Sailboats, 512×768