Image Compression with Multiresolution Singular Value Decomposition and Other Methods*

Ryuichi Ashino† Akira Morimoto ‡
Michihiro Nagase§ Rémi Vaillancourt¶

CRM-2939

January 2004

*This research was partially supported by the Japanese Ministry of Education, Culture, Sports, Science and Technology, Grant-in-Aid for Scientific Research (B), 14340045(2002–2003), (C), 13640171(2001-2002), 15540170(2003) and the Natural Sciences and Engineering Research Council of Canada and the Centre de recherche mathématique of the Université de Montréal.

†Division of Mathematical Sciences, Osaka Kyoiku University, Kashiwara, Osaka 582-8582, Japan; ashino@cc.osaka-kyoiku.ac.jp

‡Division of Information Science, Osaka Kyoiku University, Kashiwara, Osaka 582-8582, Japan; morimoto@cc.osaka-kyoiku.ac.jp

§Department of Mathematics, Graduate School of Science, Osaka University, Toyonaka, Osaka 560-0043, Japan; nagase@math.wani.osaka-u.ac.jp

¶Department of Mathematics and Statistics, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5; remi@uottawa.ca

Abstract

Digital image compression with multiresolution singular value decomposition is compared with discrete cosine transform, discrete 9/7 biorthogonal wavelet transform, Karhunen–Loève transform, and combinations thereof. The coding methods used SPIHT and run-length with Huffmann coding. The performances of these methods differ little from each other. Generally, the 9/7 biorthogonal wavelet transform is superior for most images that were tested for given compression rates. But for certain block transforms and certain images other methods are slightly superior.

Keywords : image compression, singular value decomposition, multiresolution analysis, wavelet, block decomposition

Dedicated to Professor Atsushi Inoue on the occasion of his 60^{th} birthday

To appear in Mathamtical and Computer Modelling

Résumé

On compare la compression d'images numériques obtenue par les méthodes suivantes : multirésolution par décomposition en valeurs singulières, transformation en cosinus discrète, transformation en ondelettes discrètes biorthogonales 9/7, transformation de Karhunen–Loève et méthodes mixtes. Le codage se fait par SPIHT et Huffman. La performance diffère peu entre ces méthodes. En général, la transformation en ondelettes biorthogonales 9/7 est supérieure pur la plupart des images testées à un taux de compression donné. Cependant pour certaines transformations blocs et certaines images, d'autres méthodes sont quelque peu supérieures.

1 Introduction

Image compression is an important aspect of digital image processing [1]. It is used, for instance, for image transmission, like television, and image storage, like fingerprints. Current research in this field is very active. In this paper, image compression with multiresolution singular value decomposition [2] is compared with discrete cosine transform, discrete 9/7 biorthogonal wavelet transform, Karhunen–Loève transform, and combinations thereof [3]. The coding methods use Set Partitioning in Hierarchical Trees (SPIHT) [4] and run-length with Huffmann coding. These methods are briefly reviewed and their performance is tested through numerical experiments on several well-known images. It is found that these methods differ little from each other. Generally, the 9/7 biorthogonal wavelet transform is superior for most images that were tested for given compression rates. But for certain block transforms and certain images other methods are slightly superior.

Comparative studies of compression methods are found in [5] and [6]. This paper presents a comparative study of compression methods including our new hybrid method which combines wavelet and SVD.

Section 2 considers images and linear transformations. Section 3 deals with two-sided orthogonal transforms. Section 4 defines discrete wavelet transforms. Section 5 summarizes multiresolution analysis. Section 6 covers block algorithms. Section 7 describes the coding methods. In Section 8, we propose a hybrid method using the 9/7 wavelets with SVD. In Section 9, tables list the results of numerical experiments with these methods.

2 Images and Linear Transformations

Mathematically, a gray scale image is defined by an $m \times n$ matrix and a color image is defined by three $m \times n$ matrices corresponding to red, green, and blue. In this paper, we only deal with gray scale images.

In image analysis, changing bases plays an important role, as in geometry where changing the origin and the coordinate axes simplify equations of curves. Changing bases is the unified approach that explains all the transforms that appear in this paper.

Let a matrix $A \in \mathbb{R}^{m\times n}$ represent a linear transformation f from an n dimensional vector space V_1 with basis $\mathcal{X} := \{x_1, \ldots, x_n\}$ to an m dimensional vector space V_2 with basis $\mathcal{Y} := \{y_1, \ldots, y_m\}$. Then, A is called the matrix of f with respect to the basis $\mathcal{X}$. Let $\mathcal{X}' := \{x'_1, \ldots, x'_n\}$ and $\mathcal{Y}' := \{y'_1, \ldots, y'_m\}$ be other bases of V_1 and V_2, respectively, and let the matrix $B \in \mathbb{R}^{m\times n}$ represent the linear transformation f from V_1 with $\mathcal{X}'$ to V_2 with $\mathcal{Y}'$. Then, there exist unique nonsingular matrices $P = (p_{ij}) \in \mathbb{R}^{n\times n}$ and $Q = (q_{ij}) \in \mathbb{R}^{m\times m}$ such that

$$x'_j = \sum_{i=1}^{n} p_{ij}x_i, \quad j = 1, \ldots, n, \qquad y'_\ell = \sum_{k=1}^{m} q_{k\ell}y_k, \quad \ell = 1, \ldots, m, \qquad B = Q^{-1}AP.$$

Here, P and Q are called *transition matrices* from $\mathcal{X}$ to $\mathcal{X}'$ and $\mathcal{Y}$ to $\mathcal{Y}'$, respectively.

When f is a linear transformation on V_1, A the matrix of f with respect to the basis $\mathcal{X}$, and P the transition matrix from $\mathcal{X}$ to $\mathcal{X}'$, then the matrix of f with respect to the basis $\mathcal{X}'$ is $P^{-1}AP$.

Square $n \times n$ matrices will be said to be of *order* n.

Definition 1. Two $m \times n$ matrices A and B are said to be *equivalent* if there exist nonsisngular matrices G and H of orders m and n, respectively, such that $B = GAH$. If G and H are orthogonal, then A and B are said to be *orthogonally equivalent*.

Definition 2. Two $n \times n$ matrices A and B are said to be *similar* if there exists a nonsisngular matrix H of order n such that $B = H^{-1}AH$.

3 Two-sided Orthogonal Transforms

For simplicity, we assume that V_1 is $\mathbb{R}^n$ with canonical basis $\mathcal{X} := \{e_1, \ldots, e_n\}$, V_2 is $\mathbb{R}^m$ with canonical basis $\mathcal{Y} := \{e_1, \ldots, e_m\}$, and that a given image $A \in \mathbb{R}^{m\times n}$ is the matrix of a linear transformation f from V_1 to V_2. Our purpose is to find a good pair of bases $\mathcal{X}'$ of V_1 and $\mathcal{Y}'$ of V_2 so that the matrix $B = Q^{-1}AP$ of f enables us to access information of interest about the original image A with the transition matrices P and Q from $\mathcal{X}$ to $\mathcal{X}'$ and $\mathcal{Y}$ to $\mathcal{Y}'$, respectively.

When we choose orthonormal bases for $\mathcal{X}'$ and $\mathcal{Y}'$, their transition matrices P and Q are orthogonal matrices, that is, $P^{-1} = P^T$ and $Q^{-1} = Q^T$. The L^2 norm is invariant under orthogonal transformation, $\|PAQ\|_2 = \|A\|_2$ for all $A \in \mathbb{R}^{m\times n}$, hence, the energy of images is preserved, The same holds for the Frobenius norm which enters the

definition of mean square error,

$$\mathrm{MSE} = \frac{1}{mn}\,\|A\|_F^2 =: \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2.$$

The original image A is transformed to $B = Q^TAP$ by left and right multiplication by orthogonal matrices. For this reason, we simply call this transformation a *two-sided orthogonal transform*. Because of those nice properties, we mainly work with two-sided orthogonal transforms. One aim in using two-sided orthogonal transforms in image processing is to pack in small regions points of the transformed image with large absolute values and in large regions points with small absolute values. Thus, images can be compressed by setting the small pixels equal to 0.

When dealing with images represented by $m \times n$ complex matrices $\mathbb{C}^{m\times n}$, we use the Hermitian inner product. As this is a routine change, we shall not mention it in the sequel.

3.1 Singular Value Decomposition

The following theorem is known as the *singular value decomposition* (SVD).

Theorem 1 (Singular Value Decomposition). *Let $A \in \mathbb{R}^{m\times n}$. Then there exist orthogonal matrices U of order m and V of order n such that*

$$U^TAV = \Sigma := \begin{bmatrix}\Sigma_1 & 0\\ 0 & 0\end{bmatrix}, \qquad A = U\Sigma V^T,$$

where Σ_1 is a nonsisngular diagonal matrix of singular values $\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_r > 0$ and r is the rank of A.

Note that A and Σ are orthogonally equivalent.

Remark 1. SVD is nonlinear because the orthogonal matrices U and V depend on A.

If A is a 256 gray scale image, then the representation of each pixel of A requires one byte, that is, eight bits of memory. On the other hand, since U and V are orthogonal matrices, every pixel of U and V needs more than one byte. For image compression, this fullsize SVD method is very costly.

3.2 Discrete Cosine Transform

Define the orthogonal matrix U of order m by

$$U(i,j) = \begin{cases} \sqrt{\frac{1}{m}}, & i = 1,\\ \sqrt{\frac{2}{m}}\cos\left(\frac{\pi(2j-1)(i-1)}{2m}\right), & 2 \le i \le m.\end{cases} \tag{1}$$

The orthogonal matrix V of order n is defined similarly with m replacing n in (1).

Definition 3. The *discrete cosine transform* (DCT) of an image matrix $A \in \mathbb{R}^{m\times n}$ is defined by

$$X = UAV^T$$

and the inverse discrete cosine transform (IDCT) of X is defined by

$$A = U^TXV.$$

Remark 2. The DCT is linear because the orthogonal matrices U and V are independent of A.

4 Discrete Wavelet Transform

We deal with real orthonormal wavelets. The one-dimensional scaling function $\varphi(x)$ is the solution of the following two-scale equation:

$$\varphi(x) = \sum_{n=0}^{L} h_n\sqrt{2}\,\varphi(2x-n), \quad x \in \mathbb{R}, \tag{2}$$

where $\{h_n\}$ is a finite sequence of real numbers. The one-dimensional wavelet function $\psi(x)$ is given by the two-scale expression:

$$\psi(x) = \sum_{n=0}^{L} g_n\sqrt{2}\,\varphi(2x-n), \qquad g_n = (-1)^n h_{L-n}. \tag{3}$$

Let one-dimensional scaling function $\varphi(x)$ and one-dimensional wavelet function $\psi(x)$ be given. A two-dimensional scaling function and three two-dimensional wavelet functions can be constructed from $\varphi(x)$ and $\psi(x)$ by *tensor product*.

Definition 4. The function $\varphi(x)\varphi(y)$ is called a two-dimensional *scaling function*. The functions $\psi(x)\varphi(y)$, $\varphi(x)\psi(y)$, and $\psi(x)\psi(y)$ are called *vertical*, *horizontal*, and *diagonal* wavelet functions, respectively. Scaling and wavelet functions constructed by tensor product are said to be *separable*.

Remark 3. The three wavelet functions, $\psi(x)\varphi(y)$, $\varphi(x)\psi(y)$, and $\psi(x)\psi(y)$ emphasize the vertical, horizontal, and diagonal edges, respectively, because they correspond to a highpass filter in the x direction and lowpass filter in the y direction, a lowpass filter in the x direction and highpass filter in the y direction, and highpass filters in the x and y directions, respectively. The scaling function corresponds to a lowpass filter in the x and y directions.

In this section, we fix the dimensions m and n and consider images represented by $m \times n$ matrices $A \in \mathbb{R}^{m\times n}$.

The orthogonal matrices U of order m and V of order n are defined by means of the coefficients h_n and g_n of (2) and (3), respectively, by the following procedure.

Procedure 1.

1. Each row of the upper $m/2$ by m part of U consists of the sequence $\{h_n\}$. The first row is $h_0, h_1, \ldots, h_L, 0, \ldots$; the second row is the first row shifted to the right by two places, that is, $0, 0, h_0, h_1, \ldots, h_L, 0, \ldots$; the third row is the first row shifted to the right by four places, etc.
2. When h_L reaches the last column, the portion overflowing into the right end moves to the left end periodically.
3. Each row of the lower $m/2$ by m part of U consists of the sequence $\{g_n\}$. As for the upper part, each row is the double right shift of the previous row.
4. The orthogonal matrix V of order n is constructed in the same way.

Definition 5. The *discrete wavelet transform* of an image matrix A is defined by

$$X = UAV^T.$$

In this case, we have periodic boundary conditions. The inverse wavelet transform is defined by

$$A = U^T X V.$$

Remark 4. The DWT is linear. However, U and V depend on the size of A and the length of the scaling and wavelet coefficients.

Note that the matrices U and V are not orthogonal for biorthogonal wavelets. Hence, the inverse wavelet transform $A = \widetilde{U}^T X \widetilde{V}$ is done with other matrix filters $\widetilde{U}$ and $\widetilde{V}$.

5 Multiresolution Processing

Wavelet *multiresolution analysis* is a powerful new approach which represents signal and image at several resolutions. In this section, we explain briefly wavelet-based transforms from a multiresolution point of view.

In discrete wavelet transform, we have

$$X = UAV^T,$$

where the orthogonal matrices U and V consist of two parts. The upper half-parts of U and V correspond to the lowpass filters, and the lower half-parts are the highpass filters. Note that U acts on the column vectors of the image and V^T acts on the row vectors of the image. Therefore, the discrete wavelet transform divides the image into four parts as follows.

Procedure 2.

1. The top left part is produced by the two-dimensional scaling function $\varphi(x)\varphi(y)$.
2. The top right part is is produced by the vertical wavelet function $\psi(x)\varphi(y)$.
3. The bottom left part is produced by the horizontal wavelet function $\varphi(x)\psi(y)$.
4. The bottom right part is is produced by the diagonal wavelet function $\psi(x)\psi(y)$.

The top left part is called an *approximation* because it is smooth and has large values. The other three parts are called *details* because they emphasize horizontal, vertical, and diagonal edges, respectively. These three parts have small absolute values except for the edges.

We have a multi-level decomposition by applying this decomposition to successive approximations. When we apply this decomposition to approximations and details, we have a series of multi-level decompositions called *wavelet packet decompositions*.

6 Block Algorithms

The analysis stage of a two-dimensional separable discrete wavelet transform decomposes an image into four parts, namely, the smooth part, the vertical-edge part, the horizontal-edge part, and the diagonal-edge part. Similar decompositions are achieved by the DCT and the SVD by means of the following *block algorithm*.

Algorithm 1 (Block Algorithm).

1. A given image matrix $X \in \mathbb{R}^{m\times n}$ is divided into $b \times b$ submatrices
$$X^{(k,\ell)}, \qquad 1 \le k \le m/b, \quad 1 \le \ell \le n/b.$$
2. Each submatrix $X^{(k,\ell)}$ is transformed into $X_1^{(k,\ell)}$ by the DCT or the SVD.
3. All (i, j) elements of $X_1^{(k,\ell)}$ are collected to make an $m/b \times n/b$ matrix $X_2^{(i,j)}$.
4. The $X_2^{(i,j)}$ matrices are put in the (i, j) position to produce the $m \times n$ matrix X_3 which contains b^2 parts and is similar to the matrix obtained by the DWT.

6.1 Karhunen–Loève Transform

Let an image matrix $X \in \mathbb{R}^{m\times n}$ be given. Define the $\frac{mn}{b} \times b$ data matrices X_v and X_h by stacking the $b \times b$ submatrices ${X^{(k,\ell)}}^T$ and $X^{(k,\ell)}$ in the form

$$X_v = \begin{bmatrix} {X^{(1,1)}}^T \\ \vdots \\ {X^{(m/b,n/b)}}^T \end{bmatrix}, \qquad X_h = \begin{bmatrix} X^{(1,1)} \\ \vdots \\ X^{(m/b,n/b)} \end{bmatrix}. \tag{4}$$

Calculate R_v and R_h from the data matrices X_v and X_h by

$$R_v = \frac{b}{mn} X_v^T X_v, \qquad R_h = \frac{b}{mn} X_h^T X_h. \tag{5}$$

Definition 6. The *Karhunen–Loève transform* of each submatrix $X^{(k,\ell)}$ is defined by

$$X_1^{(k,\ell)} = K_v^T X^{(k,\ell)} K_h, \tag{6}$$

where the columns of K_v and K_h are eigenvectors of the vertical and horizontal covariance matrices R_v and R_h, respectively.

All the elements of the matrices $X_1^{(k,\ell)}$ are assembled into an $m \times n$ matrix.

The inverse KLT is $X^{(k,\ell)} = K_v X_1^{(k,\ell)} K_h^T$. The two $b \times b$ orthogonal matrices K_v and K_h must be kept for the inverse KLT, because K_v and K_h depend on the image X.

6.2 Kakarala–Ogunbona's Algorithm

Kakarala–Ogunbona's algorithm [2] is a kind of multiresolution algorithm. We explain here the two-dimensional algorithm for level 1.

Algorithm 2.

1. Each $b \times b$ submatrix $X^{(k,\ell)}$ of matrix X is reshaped into a $b^2 \times 1$ column vector.

2. These column vectors are collected into a $b^2 \times (mn/b^2)$ matrix T.

3. T is factored into its singular value decomposition in the form $T = USV^T$.

4. Calculate the $b^2 \times (mn/b^2)$ matrix $A = U^T T = SV^T$.

5. Each column vector of A is reshaped into a $b \times b$ matrix $X_1^{(k,\ell)}$.

6. All the matrices $X_1^{(k,\ell)}$ are rearranged into an $m \times n$ matrix.

Figure 1 illustrates the algorithm for the level 1 singular value decomposition multiresolution analysis on a 32×32 matrix X.

Figure 2 illustrates the difference between svd and 9/7 wavelet multiresolutions at level 1 for the octagon figure. One notices that the four isolated diagonal segments appear in the lower-left and lower-right detail parts of the svd and wavelet multiresolutions, respectively. The singular values and left singular vectors for the level-1 SVD MRA of the octagon image are in the vector S and the columns of U, respectively,

```
S =  4554.4   U = 0.5000   -0.0000   -0.7071   -0.5000
     3524.2       0.5000    0.7071   -0.0000    0.5000
     3524.2       0.5000   -0.7071    0.0000    0.5000
     2024.0       0.5000    0.0000    0.7071   -0.5000
```

One sees that the first column of U is a lowpass filter.

Remark 5. Note that the norm of the first row of A is equal to the largest singular value of T, and the norm of the n-th row of A is equal to the n-th singular value of T, because the matrix V is orthogonal.

Remark 6. In the Kakarala–Ogunbona algorithm, the matrix U^T is used for the transform. When the inverse transform is needed, the $b^2 \times b^2$ orthogonal matrix U and the singular values must be kept.

6.3 DCT

The DCT is applied to each $b \times b$ submatrix $X^{(k,\ell)}$. The matrix U is the same as in (1) with $m = b$. Then

$$X_1^{(k,\ell)} = UX^{(k,\ell)}U^T.$$

Remark 7. In the DCT, U is fixed; hence the orthogonal matrix U does not have to be kept. The matrices $X_1^{(k,\ell)}$ are rearranged into an $m \times n$ matrix.

7 Coding Methods

7.1 SPIHT

The SPIHT [4] algorithm is based on the following two observations.

Observation 1. The pixels of the analyzed image having large absolute values are concentrated in the upper-left corner.

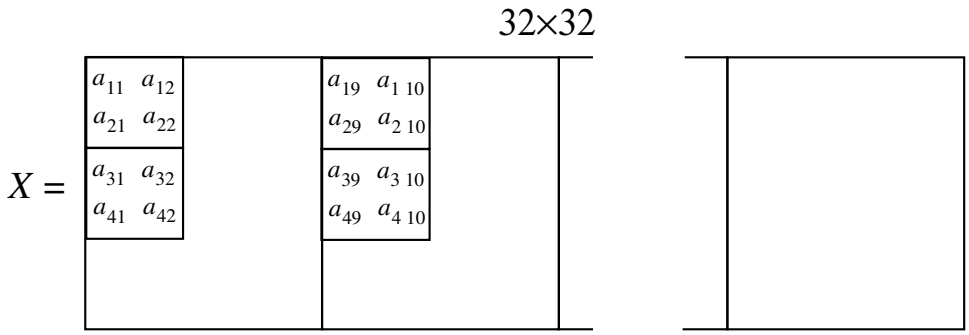
Observation 2. The hierarchical edge structure, that is, when a wavelet coefficient has large absolute value, the points at other levels corresponding to this coefficient also have large absolute values.

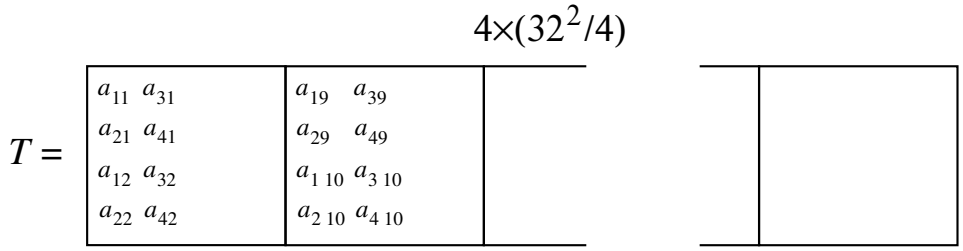
For the wavelet coefficient at level J, we define the children set of coefficients as the 2×2 block at level $J - 1$ corresponding to this coefficient. We also define the descendent set. See left Fig. 3.

SPIHT has three ordered lists:

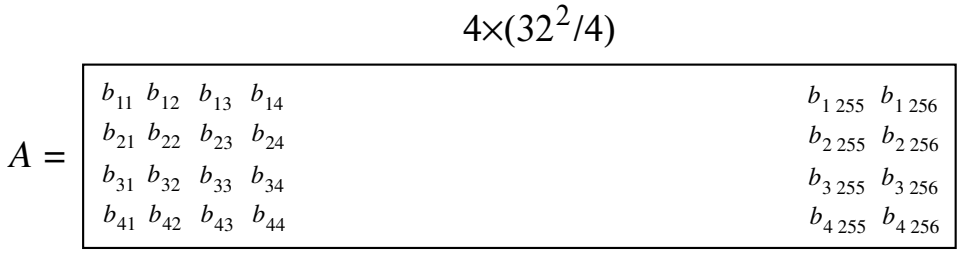
- the list of significant pixels (LSP),
- the list of insignificant pixels (LIP),
- the list of insignificant sets (LIS).

LIP and LIS are searching areas. LSP lists the pixels whose absolute values are greater than 2^N, thus requiring more than N bits. Each pixel of LIP is tested to know whether its absolute value is less than 2^N or not. Each pixel of LIS is tested to know whether the absolute values of all its descendants are less than 2^N. At first step, all the pixels of LIS are type 'A'. Some pixels of LIS will be changed from type 'A' to type 'B' in the following procedure.





$$T = USV^T \qquad A = U^T T$$



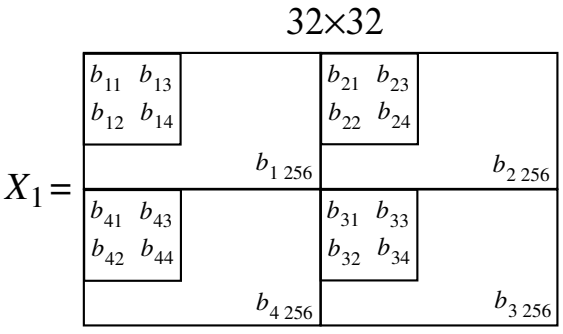


Figure 1: Level 1 SVD MRA for a 32×32 matrix.

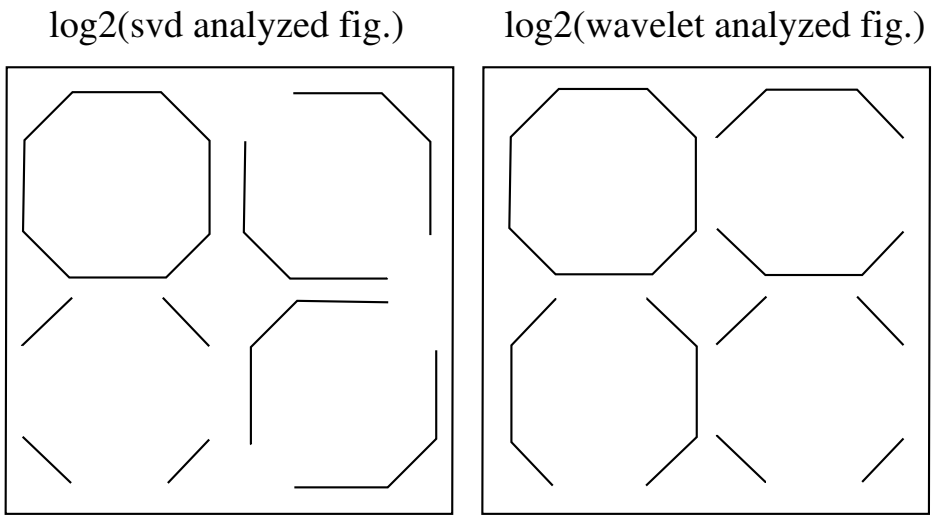


Figure 2: Negative level 1 approximation and detail subimages of octagon figure produced with SVD and 9/7 wavelet MR, respectively. The level 1 approximation is in the top left subimages.

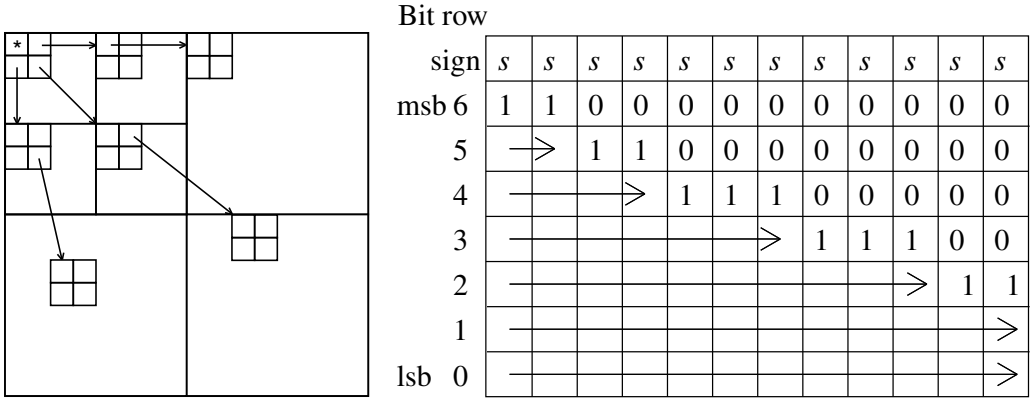


Figure 3: Left: Hierarchical structure. Right: Binary representation of the magnitude-ordered coefficients.

Procedure 3.

1. LSP is taken as an empty list, LIP is the set of top level coefficients. LIS is the set of top level wavelet coefficients and all the pixels of LIS are type 'A'. N is set to the most significant bit of all coefficients.
2. Check all the pixels of LSP and output 0 if the Nth bit of this pixel is 0, and output 1 otherwise.
3. Check all the pixels of LIP and output 0 if the absolute value of this pixel is less than 2^N. Otherwise, output 1 and, moreover, output 0 when the value of this pixel is negative and 1 if positive, and move this pixel to LSP.
4. Check all the pixels of LIS.
   (a) When a pixel is of type 'A', output 0 if the absolute value of all descendants of this pixel is less than 2^N. Otherwise, output 1 and do the following:
      i. Check all four children.
      ii. When the absolute value of a child is greater than or equal to 2^N, output 1 and, moreover, output 0 or 1 according to the sign of this child and add this child to LSP.
      iii. When the absolute value of this child is less than 2^N, add this child to the end of LIP.
      iv. When this pixel has grandchildren, move it to the end of LIS as a pixel of type 'B'.
   (b) When a pixel is of type 'B', output 0 if the absolute values of all descendants, apart from the children, are less than 2^N. Otherwise, output 1 and add each child to the end of LIS as type 'A' and delete this pixel from LIS.
5. Set N to $N-1$ and go to step (2).
6. When the number of output bits exceeds the threshold (which is decided by bpp), then stop this procedure.

7.2 Run-length and Huffmann Coding

The analyzed image can be quantized economically by the following procedure.

Procedure 4.

1. Divide each block of the analyzed image by some integer which depends on the image and the block location. Each pixel of this divided image is rounded to an integer. This quantized analyzed image has many 0 entries.
2. Reshape this image into a long row vector. In this step, use the following two methods:
   (a) Reshape each block into a vector and stack these vectors together.
   (b) Use the hierarchical tree (0 tree) algorithm.
3. Compress the 0 entries of this long row vector by the run-length coding.
4. Compress the run-length coded image by `gzip`.

8 Hybrid Wavelet-SVD Method

We propose a hybrid method which combines wavelet and singular value decompositions. This method consists in the following three steps.

Procedure 5.

1. Transform the $m \times n$ image X into the analyzed image X_1 by the level-two DWT using the 9/7 biorthogonal wavelets.
2. Decompose X_1 into 2×2-block SVD MRA up to level six to get X_2.
3. Compress X_2 by SPIHT and compress the resulting image with `gzip`.

The synthesis procedure consists in the following three steps.

Procedure 6.

1. Uncompress the `gzip` image with `gunzip` and decode the compressed code to $\hat{X}_2$.

2. Take the inverse 2×2-block SVD transform to get the synthesized image $\hat{X}_1$.

3. Obtain the reconstructed image $\hat{x}$ from $\hat{X}_1$ by the inverse DWT.

We have the following observation.

Observation 3.

1. Our hybrid wavelet-SVD method is better than SVD alone.

2. Our hybrid method is better than biorthogonal wavelet for the `fp1` and `barb` images.

The above observation leads us to the following conclusion.

Conclusion 1.

1. *The SVD decomposition depends on the data and cannot deal with data in time-frequency domain. Because our hybrid method contains wavelet analysis, which is a kind of time-frequency analysis, our hybrid method performs better.*

2. *The blocking effect in our hybrid method is weaker than with SVD, because we use long-filter wavelets in the last synthesis step.*

9 Numerical Experiments

The following methods have been used to obtain compression from 8 bits per pixel (bpp) to 1, 0.5 and 0.25 bpp.

- `bior4.4` is the biorthogonal wavelet filter with 9/7 taps of [10].
- `db2` is Daubechies' compactly supported wavelet filter with $N = 2$.
- `2by2SVDMR` and `4by4SVDMR` are the SVD multiresolutions with block size 2 and 4, respectively.
- `JPEG` is MATLAB's `imwrite` function.
- `2by2KLTMR` and `4by4KLTMR` are the KLT multiresolutions with block size 2 and 4, respectively.
- `bior4.4+SVD` consists of the following two steps. In the first step, the image is transformed by `bior4.4` wavelet to level 2. In the second step, the transformed image is decomposed by `2by2SVDMR` to level 6.

Remark 8. The SPIHT algorithm [4] is used for coding the MRA methods.

Definition 7. *Peak Signal to Noise Ratio* (PSNR) and *Signal to Noise Ratio* (SNR) for an original $m \times n$ image, X, and the reconstructed image, $\hat{x}$, are defined as follows:

$$\text{PSNR} = 10 \log_{10} \left(\frac{255^2 mn}{\sum_{i=1}^{m} \sum_{j=1}^{n} [X(i,j) - \hat{x}(i,j)]^2} \right) = 10 \log_{10} \left(\frac{255^2 mn}{\|X - \hat{x}\|_F^2} \right) \tag{7}$$

and

$$\text{SNR} = 10 \log_{10} \left(\frac{\sum_{i=1}^{m} \sum_{j=1}^{n} X(i,j)^2}{\sum_{i=1}^{m} \sum_{j=1}^{n} [X(i,j) - \hat{x}(i,j)]^2} \right) = 10 \log_{10} \left(\frac{\|X\|_F^2}{\|X - \hat{x}\|_F^2} \right). \tag{8}$$

In this work, bpp is the number of bits in the `gzip` file divided by the number of bits in the original image.

The six well-known images listed below and shown in Fig. 4 have been tested.

512×512 Lena	512×512 Boats
512×512 Barb	512×512 Yogi
512×640 Goldhill	768×768 fp1

Here the fp1 image is a sample of the FBI WSQ FINGERPRINT COMPRESSION DEMOS 4.2.5.

The numerical results are listed in Tables 1 to 3.

Figure 4: The six original figures.

Observation 4. Peak Signal to Noise Ratio (PSNR) with the bior4.4 method is generally higher except for the Yogi image at 1 and 0.5 bpp where 2by2SVDMR and 2by2KLTMR are superior.

Conclusion 2.

1. *In case of high compression ratio, that is, low bpp, block effects appeared for SVD, KLT, and JPEG, especially remarkable for* `SVD2by2` *and* `KLT2by2`*. On the other hand, in case of wavelet with long filters, images were out of focus. Our hybrid method using* 9/7 *wavelet with SVD lies between these two opposite cases.*
2. *For the fingerprint, our hybrid method using* 9/7 *wavelet with SVD was superior to the other methods.*
3. *Since Yogi has fewer grey levels,* `SVD2by2` *and* `KLT2by2` *performed better in our experiment because these transforms have short filters.*
4. *For other images, our hybrid method performed a little bit inferior to wavelet* `bior4.4`*, but superior to SVD, KLT, and JPEG.*

Every experiment was run four times successively under the same conditions, and the CPU, as measured with the MATLAB `profile` function, was taken to be the mean value of the last three runs. The computations were done on a portable PC with the following specifications: Pentium III 866 Mhz, 512 MB memory, Microsoft Windows 2000 and MATLAB R13.

10 Visual Inspection of Goldhill Image at 0.25 bpp.

At high compression ratio, that is, low bit per pixel, visual inspection is necessary to ascertain the quality of synthesized images. Here, we comment on several aspects of the goldhill image which has been compressed to 0.25 bpp with the following six algorithms: `bior4.4`, `db2`, `2by2SVDMR`, `4by4SVDMR`, `4by4KLTMR`, and `bior4.4+SVD`.

- **The sky:** good with `bior4.4` and `db2`; blocking effects with `2by2SVDMR`, `4by4SVDMR`, and `4by4KLTMR`; stripes with `bior4.4+SVD`.
- **Boundary between sky and skyline:** although the boundary is clear and smooth in the original image, it is smooth but out of focus with `bior4.4` and `db2`; it is close to the original with `2by2SVDMR`, `4by4SVDMR`, and `4by4KLTMR` but has blocking effects; it is close to the original with `bior4.4+SVD`.
- **Roofs' shingles:** The original shingles have clear rectangular forms. They are smooth but the rectangular forms cannot be seen with `bior4.4` and `db2`. They can be seen with `2by2SVDMR`, `4by4SVDMR`, and `4by4KLTMR` but there are significant blocking effects. An interpolated result between the above two results is obtained with `bior4.4+SVD`.
- **Road's cobble-stones:** As for the roof with `bior4.4` and `db2`; the road looks like a mud road. Small stones, not in the original, have been added with `2by2SVDMR`, `4by4SVDMR`, and `4by4KLTMR`. With `bior4.4+SVD`, they are closed to the original.

References

[1] A. K. Jain, *Fundamentals of digital image processing*, Prentice Hall, Englewood Cliffs NJ, 1989.

[2] R. Kakarala and P. O. Ogunbona, *Signal analysis using a multiresolution form of the singular value decomposition*, IEEE Trans. on Image Processing, **10**, No. 5, (May 2001) 724–735.

[3] P. Waldemar and T. A. Ramstad, *Hybrid KLT-SVD image compression*, 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing, **4** pp. 2713–2716, IEEE Comput. Soc. Press, Los Alamitos, CA.

[4] A. Said and W. A. Pearlman, *A new fast and efficient image codec based on set partitioning in hierarchical trees*, IEEE Trans. on Circuits and Systems for Video Technology, **6** (June 1996), 243–250.

[5] J. J. Gerbrands, *On the relationships between SVD, KLT, and PCA*, Pattern Recognition, **14** (1981) 375–381.

[6] S. O. Aase, J. H. Husøy and P. Waldemar, *A critique of SVD-based image coding systems*, Proceedings of the 1999 IEEE International Symposium on Circuits and Systems VLSL, **4** pp. 13–16, IEEE Press, Piscataway, NJ.

bior4.4 goldhill 512*640 L=6 bpp=0.25 PSNR=30.5292

db2 goldhill 512*640 L=6 bpp=0.25 PSNR=29.77

SVD2by2 gold512*640 L=6 qv=7 bpp=0.25 PSNR=29.3633

SVD4by4 gold512*640 L=3 qv=7 bpp=0.25 PSNR=29.6741

KTL4by4 goldhill L=3 bs=4 qv=7 bpp = 0.25 PSNR=29.8132

bior4.4 wL=2 + SVD gold L=6 bpp=0.25 PSNR=29.9528

Figure 5: Compression of the goldhill image to 0.25 bpp with `bior4.4`, `db2`, `2by2SVDMR`, `4by4SVDMR`, `4by4KLTMR`, and `bior4.4+SVD`, left to right and down, respectively.

[7] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd edition, The Johns Hopkins University Press, Baltimore and London, 1996.

[8] M. Unser, *An extension of the Karhunen–Loève transform for wavelets and perfect reconstruction filterbanks* SPIE, **2034** Mathematical Imaging, (1993) 45–56.

[9] J. Chen, *Image compression with SVD*, ECS 289K Scientific Computation, Dec. 13, 2000. 13 pages. `http://graphics.cs.ucdavis.edu/~jchen007/UCD/ECS289K/Project.html`

[10] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, *Image coding using wavelet transform*, IEEE Trans. Image Processing, **1** (April 1992) 205–220.

Appendix

The appendix contains Tables 1–3 and Figs. 7–11 which present detailed numerical results on the compression methods applied to the six figures considered in this paper.

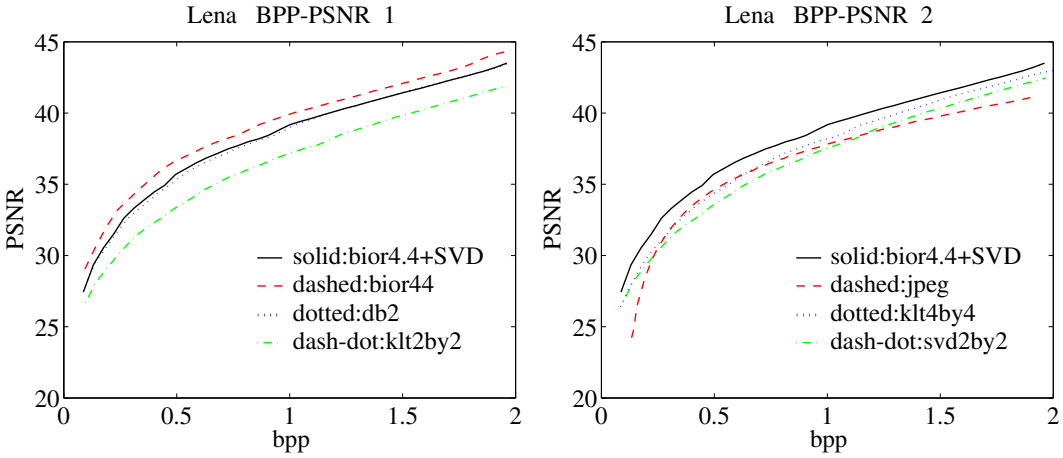


Figure 6: PSNR curve against bpp for `lena` with: (left) `bior4.4+SVD`, `bior4.4`, `db2`, `klt2by2`; (right) `bior4.4+SVD`, `jpeg`, `klt4by4`, and `svd2by2`.

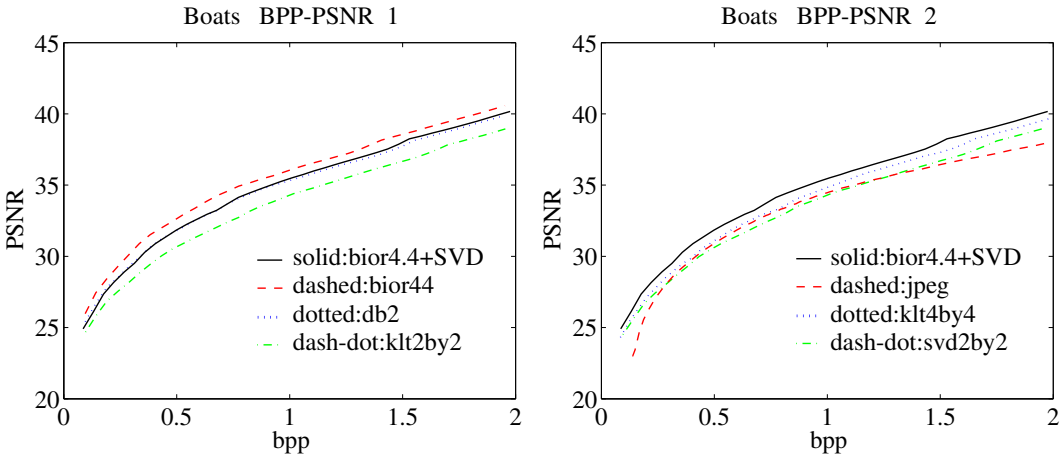


Figure 7: PSNR curve against bpp for `boats` with: (left) `bior4.4+SVD`, `bior4.4`, `db2`, `klt2by2`; (right) `bior4.4+SVD`, `jpeg`, `klt4by4`, and `svd2by2`.

Table 1: Results for 512×512 Lena (top) and Boats (bottom) images at pbb = 1, 0.5 and 0.25 except for JPEG.

bpp	Method	Level	PSNR	MSE	MaxErr	SNR	CPU
1	bior4.4	6	39.9248	6.616	13.766	34.2438	5.74
	db2	6	39.0081	8.1709	13.9631	33.3271	5.11
	by2SVDMR	6	37.5542	11.42	25.1442	31.8731	3.81
	by4SVDMR	4	38.1503	9.9552	22.4506	32.4693	3.96
1.02	JPEG		37.9285	10.4769	32	32.2474	1.00
	2by2KLTMR	6	37.181	12.4445	24.8446	31.5	34.55
	4by4KLTMR	4	38.1987	9.8448	24.6576	32.5177	9.68
	bior4.4+SVD	2+6	39.1869	7.8413	15.7794	33.5059	6.35
0.5	bior4.4	6	36.6857	13.9479	27.1368	31.0047	5.13
	db2	6	35.3878	18.806	25.0413	29.7068	4.76
	2by2SVDMR	6	33.6065	28.3417	49.3352	27.9255	3.43
	4by4SVDMR	4	34.3695	23.7756	49.9308	28.6884	3.47
0.50	JPEG		34.6181	22.4528	56	28.937	0.97
	2by2KLTMR	6	33.4096	29.6567	45.7465	27.7285	34.12
	4by4KLTMR	4	34.3619	23.8175	50.0632	28.6808	9.32
	bior4.4+SVD	2+6	35.7576	17.271	36.5254	30.0766	5.85
0.25	bior4.4	6	33.4193	29.5901	41.9485	27.7383	4.96
	db2	6	32.0355	40.6943	44.8188	26.3544	4.64
	2by2SVDMR	6	30.3235	60.3568	64.4865	24.6425	3.14
	4by4SVDMR	4	30.8061	54.0094	73.974	25.1251	3.24
0.26	JPEG		30.7576	54.6158	82	25.0766	0.97
	2by2KLTMR	6	30.2218	61.7871	61.0016	24.5408	34.74
	4by4KLTMR	4	30.7977	54.1135	63.7171	25.1167	9.17
	bior4.4+SVD	2+6	32.2857	38.416	52.0023	26.6046	5.55
1	bior4.4	6	36.0533	16.1342	22.5899	30.7107	5.41
	db2	6	35.321	19.0977	22.4977	29.9784	5.13
	2by2SVDMR	6	34.2964	24.1792	25.2346	28.9538	3.88
	4by4SVDMR	4	34.7409	21.827	26.9606	29.3983	3.95
1.01	JPEG		34.524	22.9445	42	29.1814	1.01
	2by2KLTMR	6	34.2954	24.1845	24.7067	28.9528	34.56
	4by4KLTMR	4	34.8359	21.3544	26.5875	29.4933	9.64
	bior4.4+SVD	2+6	35.4574	18.5071	24.9773	30.1148	6.23
0.5	bior4.4	6	32.6529	35.3013	40.2116	27.3103	5.12
	db2	6	31.826	42.7051	39.3889	26.4834	4.73
	2by2SVDMR	6	30.6503	55.9823	51.3853	25.3077	3.51
	4by4SVDMR	4	30.9919	51.747	51.7476	25.6493	3.48
0.51	JPEG		30.9622	52.1028	76	25.6196	0.98
	2by2KLTMR	6	30.6895	55.4793	48.4687	25.3469	34.03
	4by4KLTMR	4	31.0816	50.6898	53.2124	25.739	9.27
	bior4.4+SVD	2+6	31.855	42.4207	59.3783	26.5124	5.72
0.25	bior4.4	6	29.4905	73.1191	80.172	24.1479	4.98
	db2	6	28.6923	87.8713	75.3258	23.3497	4.53
	2by2SVDMR	6	27.5786	113.5583	81.147	22.236	3.27
	4by4SVDMR	4	27.8278	107.2269	87.8259	22.4852	3.23
0.25	JPEG		27.3174	120.5969	109	21.9748	0.99
	2by2KLTMR	6	27.658	111.5017	82.5188	22.3154	34.10
	4by4KLTMR	4	27.9562	104.1012	87.8324	22.6136	9.17
	bior4.4+SVD	2+6	28.5882	90.0041	78.3846	23.2456	5.43

Table 2: Results for 512×640 Goldhill at pbb = 1, 0.5 and 0.25 (top), and 512×512 Barb at pbb = 2.5, 1.5 and 1 (bottom), except for JPEG.

bpp	Method	Level	PSNR	MSE	MaxErr	SNR	CPU
1	bior4.4	6	36.3622	15.0265	23.7688	29.3989	6.67
	db2	6	35.7136	17.4472	27.0927	28.7503	5.99
	2by2SVDMR	6	35.0861	20.1591	23.9787	28.1228	4.53
	4by4SVDMR	3	35.7573	17.2721	23.1946	28.7941	4.43
0.99	JPEG		35.5888	17.9557	36	28.6255	1.14
	2by2KLTMR	6	35.3328	19.0461	23.8781	28.3695	43.58
	4by4KLTMR	3	36.0887	16.0033	22.8738	29.1254	11.80
	bior4.4+SVD	2+6	35.9328	16.5882	30.1457	28.9695	7.19
0.5	bior4.4	6	32.9613	32.8817	40.0127	25.998	6.45
	db2	6	32.2567	38.6729	43.5024	25.2935	5.66
	2by2SVDMR	6	31.7024	43.9384	43.1178	24.7391	4.04
	4by4SVDMR	3	32.3218	38.0975	43.6338	25.3586	3.86
0.51	JPEG		32.5053	36.5218	48	25.542	1.16
	2by2KLTMR	6	31.9307	41.6877	42.567	24.9674	43.30
	4by4KLTMR	3	32.525	36.3563	41.976	25.5617	11.35
	bior4.4+SVD	2+6	32.532	36.2975	45.177	25.5688	6.61
0.25	bior4.4	6	30.5292	57.5658	51.2446	23.5659	6.33
	db2	6	29.77	68.5611	64.5146	22.8068	5.34
	2by2SVDMR	6	29.3633	75.2917	73.4238	22.4001	3.74
	4by4SVDMR	3	29.6741	70.0923	66.1833	22.7108	3.65
0.26	JPEG		29.6083	71.1619	70	22.6451	1.15
	2by2KLTMR	6	29.5023	72.9213	74.7445	22.539	43.05
	4by4KLTMR	3	29.8132	67.8827	60.2749	22.8499	11.07
	bior4.4+SVD	2+6	29.9528	65.7348	68.479	22.9896	6.24
2.5	bior4.4	6	35.0219	20.4593	23.9209	28.746	6.26
	db2	6	34.7532	21.7652	24.0993	28.4773	6.15
	2by2SVDMR	6	33.9956	25.9134	24.3521	27.7197	4.64
	4by4SVDMR	4	34.5174	22.9797	22.07	28.2415	4.50
2.51	JPEG		31.3679	47.4558	36	25.092	1.04
	2by2KLTMR	6	33.9714	26.0578	25.3355	27.6955	34.80
	4by4KLTMR	4	34.6531	22.2725	22.7902	28.3772	10.08
	bior4.4+SVD	2+6	35.0748	20.2116	23.3207	28.7989	7.06
1.5	bior4.4	6	30.3506	59.9822	34.2253	24.0747	5.94
	db2	6	30.0205	64.7188	37.389	23.7446	5.56
	2by2SVDMR	6	29.4611	73.6165	39.915	23.1852	4.14
	4by4SVDMR	4	29.8336	67.565	37.9863	23.5577	4.11
1.51	JPEG		28.2041	98.3269	51	21.9282	1.01
	2by2KLTMR	6	29.4167	74.3729	40.5208	23.1408	34.47
	4by4KLTMR	4	29.9307	66.0707	39.6838	23.6548	9.76
	bior4.4+SVD	2+6	30.4038	59.252	38.8405	24.1279	6.51
1	bior4.4	6	28.3242	95.6439	45.1589	22.0483	5.22
	db2	6	27.9396	104.5002	47.2902	21.6637	4.91
	2by2SVDMR	6	27.3752	119.0046	47.9829	21.0993	3.79
	4by4SVDMR	4	27.6971	110.5011	49.2355	21.4212	3.90
1.00	JPEG		26.9836	130.2339	59	20.7077	1.02
	2by2KLTMR	6	27.3426	119.8991	48.3912	21.0668	34.57
	4by4KLTMR	4	27.8701	106.186	48.9431	21.5942	9.43
	bior4.4+SVD	2+6	28.2997	96.1849	55.6651	22.0238	6.14

Table 3: Results for 512×512 Yogi (top) and 768×768 fp1 (bottom) images at pbb = 1, 0.5 and 0.25 except for JPEG.

bpp	Method	Level	PSNR	MSE	MaxErr	SNR	CPU
1	bior4.4	6	40.4154	5.9093	20.4526	33.8765	5.33
	db2	6	40.1399	6.2963	21.1893	33.601	5.04
	2by2SVDMR	6	51.1502	0.49895	7.3822	44.6113	3.96
	4by4SVDMR	4	36.7073	13.8788	36.7056	30.1684	3.94
1.01	JPEG		36.6607	14.0286	42	30.1218	0.93
	2by2KLTMR	6	47.9938	1.0321	5.5872	41.4549	34.78
	4by4KLTMR	4	37.6329	11.2146	36.2385	31.0941	9.54
	bior4.4+SVD	2+6	35.8276	16.9948	36.2836	29.2887	5.99
0.5	bior4.4	6	31.431	46.7712	66.2005	24.8921	5.06
	db2	6	30.3525	59.9555	71.4808	23.8136	4.63
	2by2SVDMR	6	34.1772	24.8517	61.563	27.6384	3.37
	4by4SVDMR	4	28.8366	85.0009	127.6891	22.2977	3.35
0.51	JPEG		28.8926	83.9116	112	22.3537	0.96
	2by2KLTMR	6	34.4421	23.3812	62.9133	27.9033	34.08
	4by4KLTMR	4	29.2121	77.9592	126.2277	22.6732	9.07
	bior4.4+SVD	2+6	28.8919	83.9258	108.8086	22.353	5.65
0.25	bior4.4	6	25.9514	165.1729	115.3645	19.4125	4.76
	db2	6	25.134	199.3799	133.0653	18.5951	4.42
	2by2SVDMR	6	25.649	177.0857	182.0826	19.1101	3.17
	4by4SVDMR	4	24.0952	253.2555	218.0407	17.5563	3.29
0.26	JPEG		24.5029	230.5646	182	17.964	1.00
	2by2KLTMR	6	25.7329	173.696	179.1697	19.194	34.07
	4by4KLTMR	4	24.3188	240.5475	190.2592	17.7799	8.93
	bior4.4+SVD	2+6	24.6841	221.1439	162.3407	18.1452	5.48
1	bior4.4	6	39.6836	6.9939	14.7271	37.2212	10.30
	db2	6	38.1298	10.0024	22.6324	35.6674	9.72
	2by2SVDMR	6	36.5685	14.3293	21.8489	34.1062	6.29
	4by4SVDMR	4	38.0429	10.2044	18.848	35.5806	6.33
1.00	JPEG		37.8249	10.7297	22	35.3626	1.70
	2by2KLTMR	6	35.7928	17.1316	22.3083	33.3305	76.86
	4by4KLTMR	4	38.2215	9.7934	17.411	35.7591	20.23
	bior4.4+SVD	2+6	40.2211	6.1798	14.963	37.7587	11.93
0.5	bior4.4	6	35.7242	17.4044	32.5514	33.2619	9.76
	db2	6	33.8344	26.893	35.0699	31.3721	9.04
	2by2SVDMR	6	31.4826	46.2189	44.0702	29.0202	5.46
	4by4SVDMR	4 33.6328	28.1708	37.6675	31.1704	5.59	
0.50	JPEG		33.9999	25.8873	37	31.5376	1.68
	2by2KLTMR	6	31.1654	49.7211	43.4191	28.703	76.19
	4by4KLTMR	4	33.7659	27.3208	38.3423	31.3035	19.51
	bior4.4+SVD	2+6	35.9536	16.5091	28.774	33.4912	11.04
0.25	bior4.4	6	32.5333	36.2869	44.5788	30.0709	9.30
	db2	6	30.5249	57.6221	53.0734	28.0626	8.73
	2by2SVDMR	6	28.1966	98.4971	67.8917	25.7342	5.14
	4by4SVDMR	4	29.5129	72.7427	65.4109	27.0505	5.16
0.25	JPEG		28.9897	82.0562	84	26.5273	1.68
	2by2KLTMR	6	28.0517	101.8381	70.4279	25.5893	75.78
	4by4KLTMR	4	29.6661	70.2217	73.4828	27.2037	19.16
	bior4.4+SVD	2+6	32.4364	37.1059	42.7483	29.974	10.56

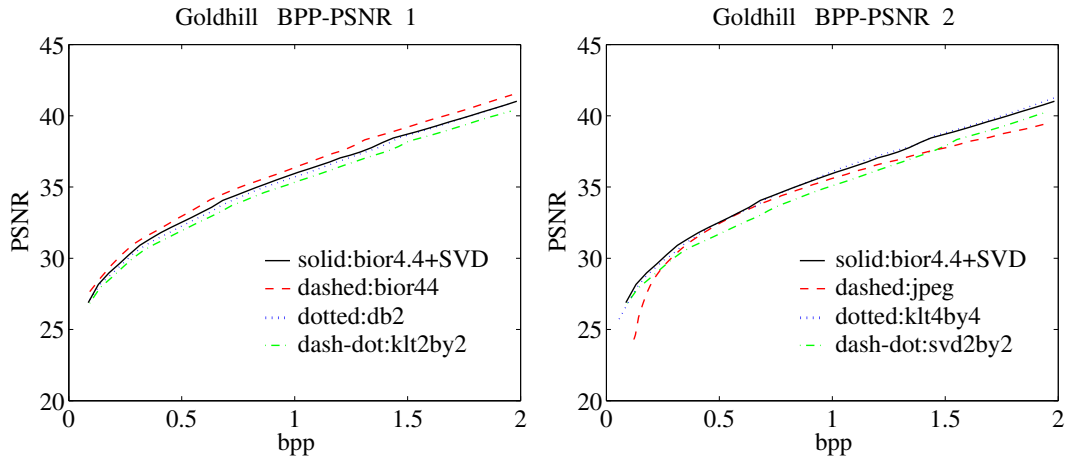


Figure 8: PSNR curve against bpp for `goldhill` with: (left) `bior4.4+SVD`, `bior4.4`, `db2`, `klt2by2`; (right) `bior4.4+SVD`, `jpeg`, `klt4by4`, and `svd2by2`.

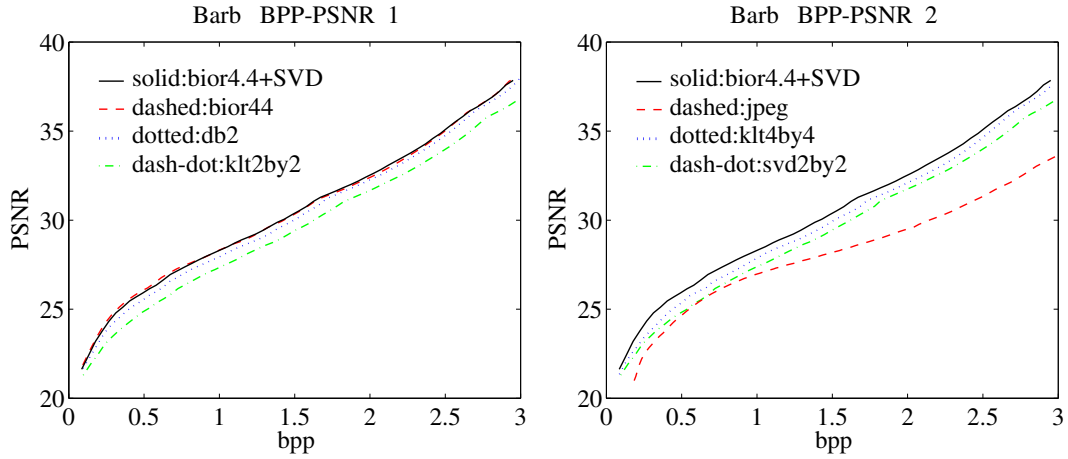


Figure 9: PSNR curve against bpp for `barb` with: (left) `bior4.4+SVD`, `bior4.4`, `db2`, `klt2by2`; (right) `bior4.4+SVD`, `jpeg`, `klt4by4`, and `svd2by2`.

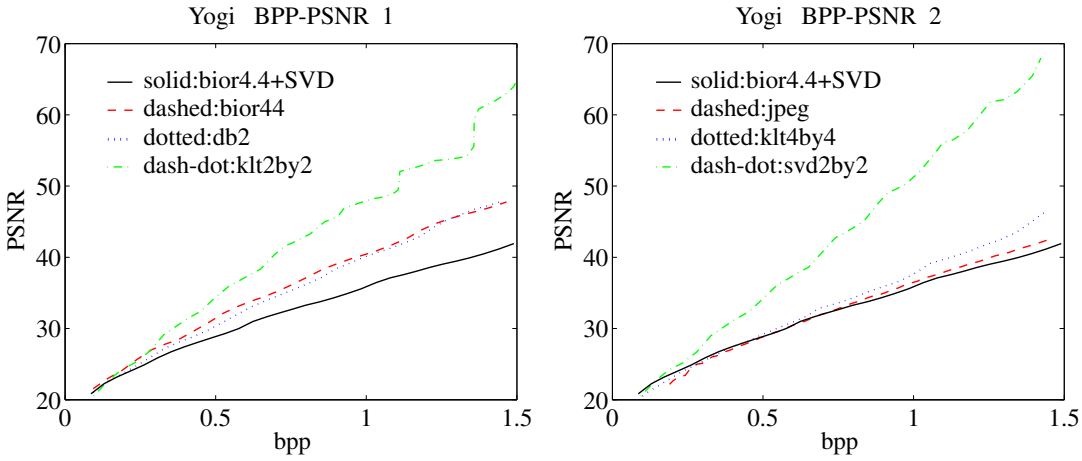


Figure 10: PSNR curve against bpp for `yogi` with: (left) `bior4.4+SVD`, `bior4.4`, `db2`, `klt2by2`; (right) `bior4.4+SVD`, `jpeg`, `klt4by4`, and `svd2by2`.

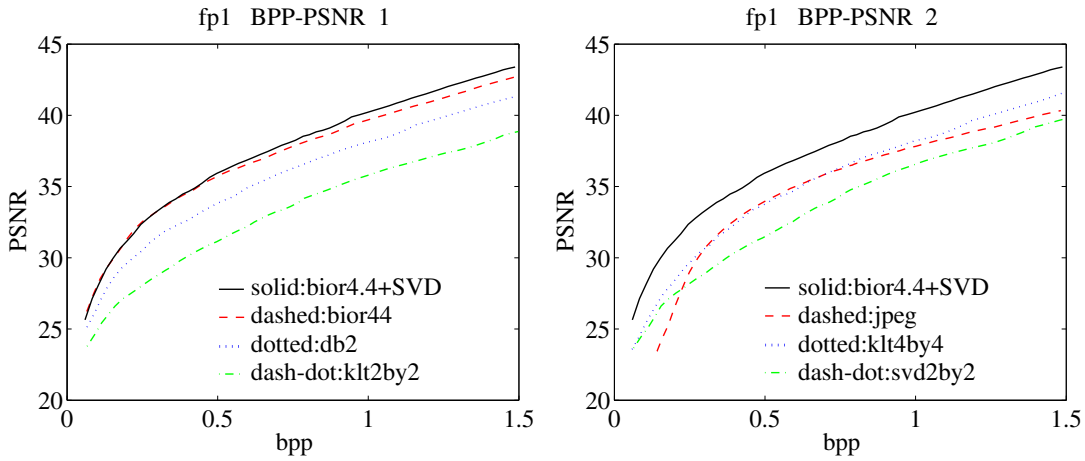


Figure 11: PSNR curve against bpp for `fp1` with: (left) `bior4.4+SVD`, `bior4.4`, `db2`, `klt2by2`; (right) `bior4.4+SVD`, `jpeg`, `klt4by4`, and `svd2by2`.