# A Wavelet Coder for Masked Images

*Patrice Y. Simard and Henrique S. Malvar*
Microsoft Research
One Microsoft Way, Redmond, WA 98052
{patrice, malvar}@microsoft.com

**Abstract**

This paper presents an algorithm for the compression of images that have "don't care" regions within them. Such regions are specified by an additional binary image, the mask, which we assume is available at both the coder and the encoder (it is transmitted separately and without loss). Unlike previous approaches, which typically fill in the masked pixels with values geared toward good compression of the image, we use a special wavelet transform which is a function of the mask, but which can be computed efficiently. We compare our method with the previous approaches by measuring the compression for a given PSNR on the valid pixels. Our algorithm is both faster and yields compression improvements of up to 18% over the current state-of-the-art coders.

## 1. Introduction

The problem we address in this paper is improving compression on images that have "don't care" regions, by taking advantage of the fact that not all the pixels need to be reconstructed. Such images appear in "clip-arts" or "cut-outs", where images of arbitrary shapes are used for pasting on other images. Another important application is in general composite document compression [1], where text or lines are superimposed on an image. The high frequencies of the text and lines usually result in poor compression of the composite image. Therefore, we formulate the problem as follows: we want to compress a rectangular image, given an additional binary image (or "mask") indicating for each pixel, whether we care about it (mask = 1) or not (mask = 0). The mask itself can be coded using standard algorithms for bi-level images, such as JBIG [2] or BLC [3]. It is not in the scope of this paper to discuss how to determine or compress the mask, but rather to achieve higher compression rate of the image (or background image), given the mask.

Typical solutions to this problem fall into two categories. The first is to fill the data that is masked with "hallucinated" data, and then use a regular image compression technique. The simplest way to fill the missing data is to fill it with the image average [4], which is not very efficient because it creates sharp discontinuities at the mask boundaries. That not only increase the required bit rate for a given peak signal-to-noise ratio (PSNR), but produces noticeable ringing near the mask boundaries. A slightly better idea is to color each pixel with the color of the closest non-masked pixel. Standard morphology algorithm allows that process to be performed with only two passes over all the pixels, leading to Voronoi-filled regions under the mask. Next, the reconstructed image is low-passed and then the known pixels are restored to their correct values. If the lowpass filter

cutoff frequency is too low, we have the same problem as in the trivial algorithm, while if the cutoff frequency is too high, the sharp edges of the Voronoi diagrams will consume too many bits. This approach, however, is very simple to implement, and computationally efficient.

A better algorithm is to use projection onto convex set (POCS) [5]. This is the approach used in systems such as DjVu [7]. Consider the following two convex sets: the set of images that matches the input on the visible pixels, and the set of images that have certain wavelet coefficient set to zero (e.g. all high-frequency coefficients beyond a certain resolution level). By alternating projection on those two sets, one can find an image that agrees with the visible pixels, and which compresses well because it has many zero wavelet coefficients.

The second category is to use wavelet transforms designed explicitly for irregular grids [8], because we can consider the set of unmasked pixels as an irregular sampling grid. Such wavelet decompositions are needed in problems in computer vision and compression of geometry data in computer graphics [9]. The approach we present here is also based on adapting the wavelet transform to an irregular grid, in our case, the mask pattern. However, whereas the techniques in [8] and [9] lean towards smooth data interpolation, our goal is to maximize compression performance. In Section 2 we present the derivation of our "masked wavelets," based on adaptive lifting. Experimental results are discussed in Section 3, and final remarks are presented in Section 4.

## 2. Masked Wavelets

The approach taken in this paper is to adapt the wavelet locally as a function of the mask. The lifting formalism [10] is most useful here, because the inverse wavelet transform can easily be derived from the straight wavelet transform. Many strategies exist for adapting wavelet transforms while preserving perfect invertibility at all points [11]. However, all such techniques adapt the transform based on signal characteristics, and they do not consider missing data.

According to the lifting scheme, the wavelet transform can be decomposed as a succession of prediction steps (for the highpass filter), and update steps (for the lowpass filter) performed at each resolution. In the traditional wavelet transform, each prediction step computes a linear function of a predetermined set of neighboring pixels to predict the current pixel. The same is true for the update step. However, in the case of masked wavelet, certain pixels have no suitable values for either prediction or update. The main idea presented in this paper consists in, for each pixel, generating an appropriate linear combination of the neighboring pixels that are available, for both the prediction and the update steps. At each pixel location, the mask creates a pattern of availability, which must then be converted into the correct linear combination of the available pixel values.
This is best illustrated with a picture. Figure 1(left) shows one step of the traditional computation of a cubic wavelet using "lifting". That diagram follows the same formalism as in [10]. The diagram shows, for a cubic wavelet, the prediction step at position 3, and the corresponding update step at position 6 (the other positions are omitted in the diagram for clarity). The coefficient next to each arrow indicates how to compute the linear combination in order to perform each step. For instance, the "detail" value (resulting from the high pass filter) at position 3 is computed by computing the following equa-

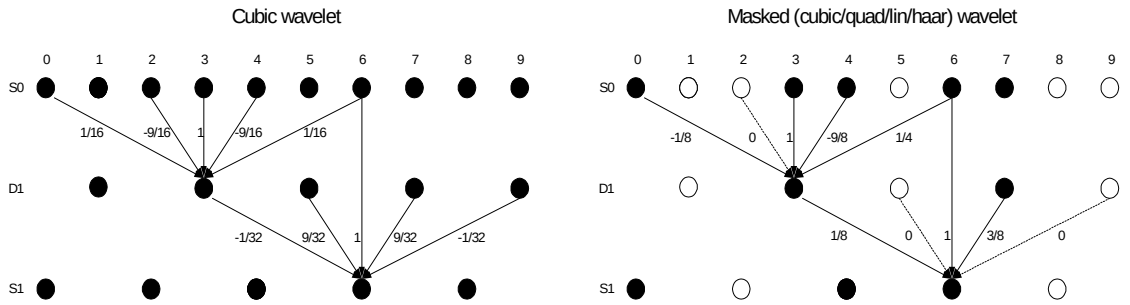


**Figure 1.** Right: lifting steps for a traditional cubic wavelet transform. Left: lifting steps for the masked wavelet transform.

tion: $d_3 = s_3 - (-s_0 + 9s_2 + 9s_4 - s_6)/16$. The update step is computed using equation $s_6 = d_6 + \left(-d_3 + 9d_5 + 9d_7 - d_9\right)/32$. Note that computation can be done in place.

Figure 1(right) shows the problem that arises when some of the pixel values are missing. In the picture, no value is available for positions 1, 2, 5, 8, and 9. Obviously, computing the traditional cubic wavelet would not work because the result would depend on missing values. Setting the missing values to some constant (zero, or some average over the whole image) can introduces sharp discontinuities, which translate into poor compression and/or undesirable artifacts. Our approach, in contrast, changes the wavelet function on a case-by-case basis as a function of the mask. For instance, during the prediction step of lifting, if $k$ values are available for the prediction, a polynomial of degree $k-1$ is used for the interpolation. When only 3 values are available, a quadratic instead of cubic polynomial is used. If only one value was available, a constant polynomial would be used, and the wavelet would be a Haar wavelet. Note that if the signal is a polynomial of degree $k-1$ and $k$ pixels are not masked, the prediction is perfectly accurate. We now explain more formally how the wavelet coefficients are computed.

## 2.1. The prediction step

According to lifting, the predict step computes a prediction for all the coefficients at odd positions from the coefficients at even positions. The difference between the odd coefficient and its prediction is the wavelet coefficient. This can be viewed as a high pass filter with some zeros at the odd positions. For clarity, and without loss of generalization, we will restrict ourselves to a 7-tap filter, as illustrated in Figure 2.

We took some liberty in the indexing. The signal $s$ has been centered on position 3 and indexed accordingly. This notation will simplify the calculations when we introduce moments. The filter $a$ is indexed from 0 to in accordance to the standard matrix notation (which we will use later). The wavelet coefficient $d$ is given by equation:

$$d = s_0 + \sum_{i=0}^{i=\lfloor k/2 \rfloor} s_{2i-k/2} a_i$$

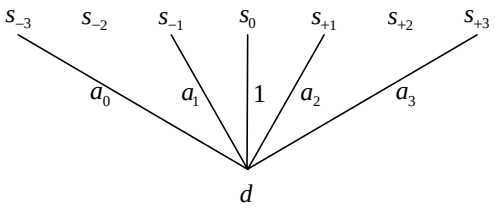


**Figure 2.** Lifting step: $s_0$ is predicted as a function of $s_{-3}, s_{-1}, s_1, s_3$ and the residual d is computed.

where $k$ is the number of tap in the filter (in this case $k = 7$). The moments of the high pass filter can be written as (setting $s_i = i^n$):

$$M_n = 0^n + \sum_{i=0}^{i=\lfloor k/2 \rfloor} (2i - \lfloor k/2 \rfloor)^n a_i$$

If we assume that a regular signal can be approximated by a low order polynomial (using Taylor expansion) of order $j$, then if we choose $a$ so as to set the first $j+1$ moment to zero, then the wavelet transform will have many zeros and compress very well. Since in our example, $a$ has 4 degrees of freedom, we can set the first 4 moments to zero. It is easy to verify this results in the following system:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ -3 & -1 & 1 & 3 \\ 9 & 1 & 1 & 9 \\ -27 & -1 & 1 & 27 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{1}$$

which we can rewrite in matrix notation as: $wa = c$. The solution to this system are the coefficients used in the well know cubic wavelet: $a = [1/16, -9/16, -9/16, 1/16]$.

We now generalize this to the case when a mask is present. In other words, some of the coefficients $s_{2i-k/2}$ are missing. This can be modeled by introducing a matrix $m$,

$$m = \begin{bmatrix} m_0 & 0 & 0 & 0 \\ 0 & m_1 & 0 & 0 \\ 0 & 0 & m_2 & 0 \\ 0 & 0 & 0 & m_3 \end{bmatrix}$$

where $m_i \in \{0,1\}$, such that: $wma = c$. The effect of $m$ is to zero columns of *w*. Since we have fewer coefficients to negate the moments, we must also reduce the number of vanishing moments. This is achieved by using

$$pwma = pc \tag{2}$$

where *p* is given by:

$$p = \begin{bmatrix} p_0 & 0 & 0 & 0 \\ 0 & p_1 & 0 & 0 \\ 0 & 0 & p_2 & 0 \\ 0 & 0 & 0 & p_3 \end{bmatrix}$$

| $(m_0, m_1, m_2, m_3)$ | $a_0$ | $a_1$ | $a_2$ | $a_3$ |
|---|---|---|---|---|
| 0 0 0 0 | 0 | 0 | 0 | 0 |
| 0 0 0 1 | 0 | 0 | 0 | -1 |
| 0 0 1 0 | 0 | 0 | -1 | 0 |
| 0 0 1 1 | 0 | 0 | -3/2 | 1/2 |
| 0 1 0 0 | 0 | -1 | 0 | 0 |
| 0 1 0 1 | 0 | -3/4 | 0 | -1/4 |
| 0 1 1 0 | 0 | -1/2 | -1/2 | 0 |
| 0 1 1 1 | 0 | -3/8 | -3/4 | 1/8 |
| 1 0 0 0 | -1 | 0 | 0 | 0 |
| 1 0 0 1 | -1/2 | 0 | 0 | -1/2 |
| 1 0 1 0 | -1/4 | 0 | -3/4 | 0 |
| 1 0 1 1 | -1/8 | 0 | -9/8 | 1/4 |
| 1 1 0 0 | 1/2 | -3/2 | 0 | 0 |
| 1 1 0 1 | 1/4 | -9/8 | 0 | -1/8 |
| 1 1 1 0 | 1/8 | -3/4 | -3/8 | 0 |
| 1 1 1 1 | 1/16 | -9/16 | -9/16 | 1/16 |

**Table 1.** Solutions of equation (2) for different values of $m$ when the dimension of the system is 4.

with

$$p_i = (tr(m) > i)?1:0;$$

The constraints on $p_i$ ensure the lines in system (1) are zeroed from the bottom, for each zero coefficient in $m$. In other words, if there are $j$ coefficients $m_i$ which are not equal to $0$, then $p_i = 1$ for $i = [0..j-1]$, and $p_i = 0$ otherwise. It can easily be verified that the system always has a unique least-squares solution for every value of $m_i \in \{0,1\}$. The solutions for $a$ for every possible values of $m$ are given by system (2) and summarized for the cubic wavelet in Table 1.

## 2.2. The update step

The update step is a little less intuitive. In the prediction step, we wanted the first moments of the high pass filter to vanish. In the update step, we want the first moments of the low pass filter to vanish, after the signal has been multiplied by $(-1)^i$. In other words, if a regular[1] signal is multiplied by the highest frequency signal, $(-1)^i$, the low pass filter should output zero. This condition can easily be cast as 0-moment constraint, as in the previous section, except that the input will be of the form $s_i = (-1)^i i^n$ instead of $s_i = i^n$. Using similar notation as for the predict step, the update step is shown in Figure 3, which corresponds to the equation:

$$s = s_0 + \sum_{i=0}^{i=\lfloor k/2 \rfloor} d_{2i-k/2} b_i$$

Where *k* is the number of tap in the filter (in this case $k = 7$). The moments of the low pass filter can be written as

[1] By regular, we mean that it can be written as a low order polynomial

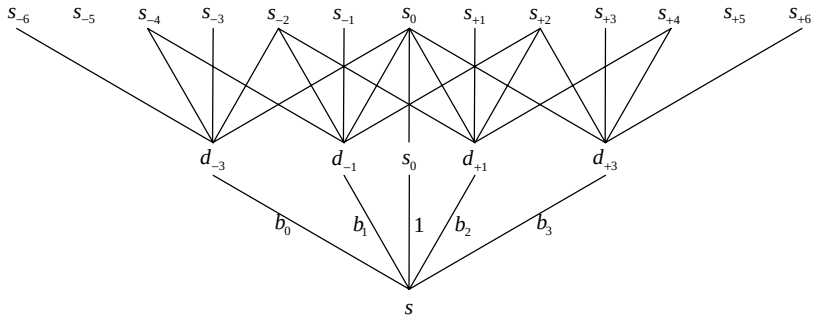


**Figure 3.** Update step: $s$ is updated as a function of $d_{-3}, d_{-1}, d_1, d_3$.

$$M_n = 0^n + \sum_{i=0}^{i=\lfloor k/2 \rfloor} d_{2i-k/2} b_i$$

but, for each $d_i$, we can rewrite the equation locally as (assuming $s_i = (-1)^i i^n$):

$$d_j = s_j + \sum_{i=0}^{i=\lfloor k/2 \rfloor} s_{j+2i-k/2} a_i = -j^n + \sum_{i=0}^{i=\lfloor k/2 \rfloor} (j+2i-k/2)^n a_i$$

since $j$ and $k/2$ are odd. Because individual $a_i$ are set to generate zero moments, we also have:

$$0 = j^n + \sum_{i=0}^{i=\lfloor k/2 \rfloor} (j+2i-k/2)^n a_i$$

which implies $d_j = -2j^n$. We can therefore write:

$$M_n = 0^n + \sum_{i=0}^{i=\lfloor k/2 \rfloor} -2(2i-k/2)^n b_i$$

For the wavelet to compress well regular signal, we want as many moment equal to zero as possible. Since we have 4 degrees of freedom, we can set the first 4 moments to zero. It is easy to verify this results in the following system:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ -3 & -1 & 1 & 3 \\ 9 & 1 & 1 & 9 \\ -27 & -1 & 1 & 27 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3}$$

Which we can rewrite in matrix notation as $wb = c'$. The solution to this system are the coefficients used in the well know cubic wavelet: $b = [-1/32, 9/32, 9/32, -1/32]$.

Now, lets assume that some of the coefficients $s_i$ are missing. Let's first assume that all missing values are only at even locations. We can solve this system in a similar fashion as before. The solution verifies:

$$pwmb = pc' \tag{4}$$

Note that $m$ and $p$ matrix depend on the location centered in $s_0$. Each location views a different part of the mask and has therefore it own $m$ and $p$. Solutions to (4) are given

Table 2. To derive (4), we have assumed that all the odd locations were not masked. If some odd locations are masked, but the number of masked value is less than $n$, then $d_j = -2\,j^n$, and equation (4) holds. Otherwise, there are two many masked pixels in the predict step to nullify the n-th moment in the update step (the wavelet coefficient will still be as small, but not zero). The inverse wavelet transform is easily computed by undoing each step locally, thanks to the lifting formalism.

## 2.3. Implementation

Wavelet transforms are usually normalized so that quantization error on each of the wavelet coefficients introduces comparable reconstruction error. With masked wavelet normalization for each coefficient is a complex function of the mask. We have computed the optimal normalization and found out that it did not improve compression by more than 0.5%. The normalization we use is the one that would be optimal with no mask.

The filter coefficients can be stored into 1 or 2 tables (they differ only by a factor –1/2). The tables can be indexed by a 4 bits number corresponding to the mask. The only difficulty is in the coding of the inverse wavelet transform. One has to realize that the coefficient values to be used are the one that invert the predict and the update steps that were made with a mask viewed from different locations. A simple solution is to keep a 7 bit index representing the mask. The four subgroups of 4 bits in the 7 bits correspond to the correct lines in the table for the 4 positions during the inverse transform.

In the absence of a mask, the 2D wavelet transform is computed by a performing a succession of 1D wavelet transforms. This is possible because the 2D filter is separable and can be done using two 1D filters. This however is not true in the presence of a mask since masks are in general not separable. To compute the 2D masked wavelet, we currently do a succession of 1D masked wavelet transforms. This is not ideal, and the order in which we perform the 1D sweeps does change the result. It would be better to com-

| $(m_0, m_1, m_2, m_3)$ | $b_0$ | $b_1$ | $b_2$ | $b_3$ |
|---|---|---|---|---|
| 0 0 0 0 | 0 | 0 | 0 | 0 |
| 0 0 0 1 | 0 | 0 | 0 | 1/2 |
| 0 0 1 0 | 0 | 0 | 1/2 | 0 |
| 0 0 1 1 | 0 | 0 | 3/4 | -1/4 |
| 0 1 0 0 | 0 | 1/2 | 0 | 0 |
| 0 1 0 1 | 0 | 3/8 | 0 | 1/8 |
| 0 1 1 0 | 0 | 1/4 | 1/4 | 0 |
| 0 1 1 1 | 0 | 3/16 | 3/8 | -1/16 |
| 1 0 0 0 | 1/2 | 0 | 0 | 0 |
| 1 0 0 1 | 1/4 | 0 | 0 | 1/4 |
| 1 0 1 0 | 1/8 | 0 | 3/8 | 0 |
| 1 0 1 1 | 1/16 | 0 | 9/16 | -1/8 |
| 1 1 0 0 | -1/4 | 3/4 | 0 | 0 |
| 1 1 0 1 | -1/8 | 9/16 | 0 | 1/16 |
| 1 1 1 0 | -1/16 | 3/8 | 3/16 | 0 |
| 1 1 1 1 | -1/32 | 9/32 | 9/32 | -1/32 |

**Table 2.** Solutions of equation (4) for different values of $m$ when the dimension of the system is 4.

pute directly a 2D masked wavelet, but this would generate gigantic tables. It is not clear that the benefits would justify the expense, but the idea is under investigation.

## 3. Experimental Results

To evaluate the performance of the proposed masked wavelet coder, we compared it with the PWC encoding [12] of the background image, using POCS to fill in under the mask. We set the POCS module to stop after 10 iterations, which makes encoding computationally intense (because 10 direct and inverse wavelet transform steps are required), whereas the masked wavelet coder has a speed comparable to that of PWC. We used both codecs to encode a set of artificially generated composite images, by applying a predefined mask to a 256×256 cut-out of the popular "Lena" image. Figure 4 shows an example of reconstructed images using both PWC/POCS and the masked wavelet coder. For the same PSNR under the mask, both encoders produce images of similar subjective quality, and essentially free from ringing artefacts near the mask boundaries.

Table 3 shows the distortion-rate performance of both codecs. The performance improvement with the masked wavelet coder is naturally more significant for masks with a higher percentage of don't care pixels. We note that at higher bit rates the performance gap is higher, probably because the PWC/POCS coder has to spend more bits to encode the hallucinated areas generated by POCS (which are not quantized to zero at high rates).

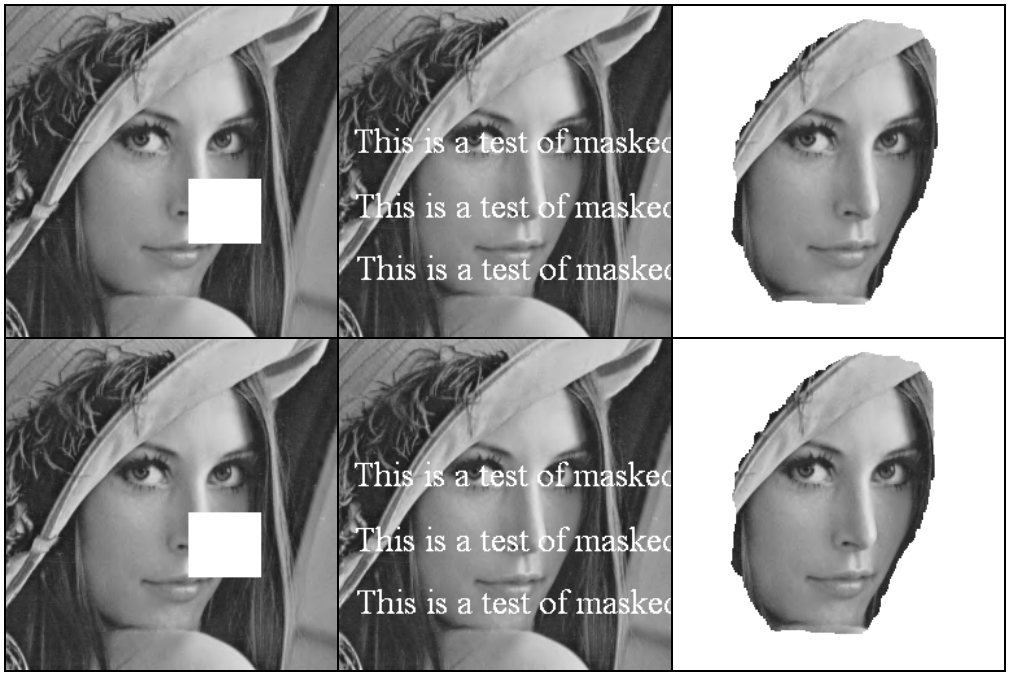


**Figure 4.** 256×256 masked image coding result, for PSNR = 40 dB for unmasked pixels. Top: PWC/POCS; bottom: proposed masked wavelet codec. Note the absence of artifacts near the mask boundaries. The proposed wavelet coder is an order of magnitude faster than PWC/POCS.

| PSNR | Mask # 1 | | | Mask # 2 | | | Mask # 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| | File length, bytes | | Gain % | File length, bytes | | Gain % | File length, bytes | | Gain % |
| | PWC POCS | Masked Wavelet | | PWC POCS | Masked Wavelet | | PWC POCS | Masked Wavelet | |
| 38 | 8,576 | 8,772 | -2.3% | 8,847 | 9,087 | -2.7% | 3,513 | 3,009 | 14.3% |
| 40 | 11,617 | 11,835 | -1.9% | 11,928 | 12,152 | -1.9% | 4,915 | 4,193 | 14.7% |
| 42 | 14,950 | 15,126 | -1.2% | 15,350 | 15,426 | -0.5% | 6,560 | 5,634 | 14.1% |
| 44 | 18,250 | 18,329 | -0.4% | 18,706 | 18,651 | 0.3% | 8,231 | 6,970 | 15.3% |
| 46 | 21,579 | 21,456 | 0.6% | 22,103 | 21,791 | 1.4% | 9,902 | 8,409 | 15.1% |
| 48 | 24,784 | 24,423 | 1.5% | 25,421 | 24,683 | 2.9% | 11,441 | 9,688 | 15.3% |
| 50 | 27,987 | 27,285 | 2.5% | 28,666 | 27,543 | 3.9% | 12,989 | 10,860 | 16.4% |
| 52 | 31,217 | 29,967 | 4.0% | 32,039 | 30,174 | 5.8% | 14,579 | 12,071 | 17.2% |
| 54 | 34,290 | 32,694 | 4.7% | 35,154 | 32,853 | 6.5% | 16,111 | 13,220 | 17.9% |

**Table 3.** Performance of coders for the masked images shown in Fig. 4.

At lower bit rates, however, we observe a decrease in the gain for masks # 1 and # 2. In view of our experiments, we believe that the choice of the wavelet filters and normalization factors for coding are not the cause. We suspect that aliasing in mask # 1 and irregularity in mask # 2 are at least partially responsible. This is a topic that we continue to investigate.

For mask # 3 in Fig. 4, the reduction in file size with the masked wavelet coder is as high as about 18%. Note that PWC/POCS is computationally an order of magnitude more expensive during encoding. Also, PWC/POCS has a performance that should be comparable to state-of-the-art composite image coders such as DjVu [7]. Therefore, our proposed masked wavelet coder should improve on the state-of-the-art by similar margins.

## 4. Conclusion

There are two approaches to compression of images with “don’t care” pixels. These pixels can be filled with interpolated values or a special wavelet can be designed to work around them. If compression is the objective, the first approach solves a more difficult problem than necessary. Finding an appropriate value for a missing pixel can hurt compression in at two ways. First, the ideal value may not be the same for the different bands HL, LH and HH at different levels. Since only one value must be chosen (and used by all bands), it can result in a sub-optimal solution. Second, all wavelet coefficients must be encoded, including the newly filled pixels, wasting bits.

The second approach is solving a more specific problem. Find a wavelet transform which, given a mask, will yield small wavelet coefficients wherever the signal is regular. This is the vanishing moments constraint used in standard wavelet design. Using this principle, we have derived an algorithm that is both efficient and which improve compression performance by as much as 18%.

Future work include 2D masked wavelets to address the nonseparability issue, masked wavelets with gray-level masks (alpha-blending), and algorithms for automatically generating mask for optimal compression, using the masked wavelet as a measure.

## References


[1] R. L. de Queiroz, "Compression of compound documents," Proc. IEEE International Conf. on Image Processsing, Kobe, Japan, Oct. 1999.

[2] B. G. Haskell, P. G. Howard, Y. A. LeCun, A. Puri, J. Ostermann, M. R. Civanlar, L. R. Rabiner, L. Bottou, and P. Haffner, "Image and video coding-emerging standards and beyond," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, Nov 1998, pp. 814-837.

[3] H. S. Malvar, "Fast Adaptive Encoder for Bi-Level Images," *IEEE Data Compression Conf.*, Snowbird, UT, Mar. 2001 (submitted).

[4] R. L. de Queiroz, "On data filling algorithms for MRC layers," *Proc. IEEE International Conf. on Image Processsing*, Vancouver, Canada, Sept. 2000.

[5] D. C. Youla and H. Webb, "Image restoration by the method of convex projections," *IEEE Trans. on Medical Imaging*, vol. 1, pp. 81–94, Oct. 1982.

[6] H. Chen, M. R. Civanlar, and B. G. Haskell, "A block transform coder for arbitrary shaped image segments," *Proc. IEEE International Conf. on Image Processsing*, Austin, TX, pp. 85–89, Nov. 1994.

[7] P. Haffner, L. Bottou, P. G. Howard, P. Simard, Y. Bengio, and Y. Le Cun, "Browsing through high quality document images with DjVu," *Proc. IEEE International Forum on Research and Tech. Advances in Digital Libraries*, Santa Barbara, CA, pp. 309–318, Apr. 1998. See also the DjVu software at http://www.lizardtech.com/products/djvu.html.

[8] I. Daubechies, I. Guskov, P. Schröeder, and W. Sweldens, "Wavelets on Irregular Point Sets," *Phil. Trans. R. Soc. Lond. A*, to appear.

[9] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, T. Lounsbery, and W. Stuetzle, "Multiresolution analysis of arbitrary meshes," *Proc. ACM Computer Graphics (SIGGRAPH)*, Los Angeles, CA, pp. 173–182, Aug. 1995.

[10] W. Sweldens, "The Lifting Scheme: A new philosophy in biorthogonal wavelet constructions" in A. F. Laine and M. Unser, eds., *Wavelet Applications in Signal and Image Processing III*, pp. 68–79, Proc. SPIE 2569, 1995.

[11] G. Davis, S. G. Mallat, and Z. Zhang, "Adaptive time-frequency decompositions," *Optical Engineering*, vol. 33, pp. 2183–2191, July 1994.

[12] H. S. Malvar, "Fast progressive wavelet coding," *Proc. IEEE Data Compression Conf.*, Snowbird, UT, pp. 336–343, Mar. 1999.