Trees, Windows and Tiles for Wavelet Image Compression

Wee Sun Lee *

Abstract

We investigate the task of compressing an image by using different probability models for compressing different regions of the image. In an earlier paper, we introduced a class of probability models for images, the k-rectangular tiling of an image, which is formed by partitioning the image into k rectangular regions and generating the coefficients within each region by using a probability model selected from a finite class C of probability models. We also described a computationally efficient sequential probability assignment algorithm that is able to code an image with a code length that is close to the code length produced by the best model in the class. In this paper, we investigate the performance of the algorithm experimentally on the task of compressing wavelet subbands. We compare the method with compression methods that aim to compress as well as the best pruning of a quad-tree and compression methods that exploit the local statistics in a window around the coefficient being compressed. For a class C consisting of a small number of Laplacian distributions and the uniform distribution, we find that the best tiling method works best, but the difference in performance is significant for only a few of the images tested.

1 Introduction

Wavelet coefficients are known to be large around edges and small in smooth regions of an image. For effective compression of these coefficients, it is desirable to use different probability models for compressing different regions of a wavelet subband. In this paper, we compare the performance of compression methods that are able to adapt the probability assignment to the region that is being coded, by either choosing one of a finite number of probability models or weighting the probability models to form a weighted distribution.

In an earlier paper [7], we introduced a class of probability models formed by partitioning an image into k rectangular regions and generating the coefficients within each region by using a probability model from the finite class of N probability models. We call the class of probability models that is generated in this way, the class of *k-rectangular tiling* of the image. We also described a computationally efficient sequential probability assignment

*Department of Computer Science, School of Computing, National University of Singapore, Singapore 117543. Email: leews@comp.nus.edu.sg. Web: http://www.comp.nus.edu.sg/∼leews.

algorithm, *best tiling*, which is able to code an image with a code length that is close to the code length produced by the best model in the class.

In this paper, we compare experimentally the best tiling method with *quad-tree* methods which aim to compress as well as the best pruning of a quad-tree, and *windowing* methods which use the local statistics to estimate the best probability assignment for the current coefficient. For the quadtree methods, we use both optimal pruning and a Bayesian tree weighting method. For windowing, we use two methods: using the neighbouring pixels to select a probability model to use and using the neighbouring pixels to find a weighting for the probability models.

We work with wavelet coefficients quantized with a single scalar quantizer. With quadtrees, optimal pruning can be performed in a rate-distortion sense, where different quantizers are selected with different probability models. However, in this work, we use only a single scalar quantizer and attempt to minimize the code length for the fixed quantizer. For the tree weighting method, we use the Bayesian tree weighting method which has been used for compressing finite memory sources in [11] and for prediction in [5]. The tree weighting method is guaranteed to outperform an optimal pruning method (with a single scalar quantizer) when coding is done using the same prior distribution.

Variants of windowing have previously been used in successful wavelet image coders [3, 8]. The methods essentially assume that the coefficients outside a small window around the coefficient that is being coded are irrelevant. We use two simple variants of windowing, one that weighs a finite number N of probability models using Bayesian weighting assuming that only the pixels inside the window exist and another that selects the most probable model given the pixels inside the window.

Algorithms that compress close to the *best segmentation* of a sequence, where a different model is used in each segment have been considered in [12, 10]. In [10], methods that switch between two text compression algorithms that have different properties are studied. The class of k-rectangular tiling of an image can be considered as a two dimensional extension of the class of segmentation of a sequence. As in [10], the tiling method can also be used with N different compression algorithms, each of which performs well in a different region of the image. However, in this paper, we use a finite number of Laplacian distributions and a uniform distribution as our probability models.

We find that the best tiling method uniformly outperforms all the other methods described here, although the difference in performances of the methods are not large. All the methods used here do not exploit inter-band information in compressing the wavelet coefficients. We also compared the performance of the coders against a zerotree coder (SPIHT [9]) which exploits a parent-child relationship between wavelet subbands. We find that the zerotree coder performs better on some synthetic images but provides approximately the same compression performance on natural images.

2 Compression Methods

All the methods, except optimal pruning, can be considered as sequential probability assignment methods. In sequential probability assignment, the predictor provides probability assignment $a(x_i|x^{i-1})$ for x_i given x^{i-1}, where x^{i-1} denote the sequence of coefficients

$(x_0, \ldots, x_{i-1})$. The probability assigned to the image $(x_0, \ldots, x_t)$ is then

$$p(x_0, \ldots, x_{t-1}) = a(x_0)a(x_1|x_1) \cdots a(x_i|x^{i-1}) \cdots a(x_{t-1}|x^{t-2}).$$

Using an arithmetic coder, the code length $-\log_2 p(x_0, \ldots, x_{t-1})$ can be approached with a very small redundancy. Each probability assignment method attempts to provide an appropriate assignment function $a(\cdot|x^{i-1})$ such that a high probability is assigned to the image in order to minimize the code length.

2.1 Quadtree

We describe a simple dynamic programming algorithm for finding the pruned quadtree model that minimizes the total description length of transmitting the image. Start the routine *Prune* by using the root as a parameter. If the current node is a leaf, the routine returns the minimum total description length of coding the node as the *cost* of the node. If the current node is an internal node, *Prune* calls itself four times with the four children as the parameters. It then compares the following two sums: the minimum total description length of coding the node plus one and the sum of the costs of all its children. If the former is smaller, the children of the nodes are pruned away and the former becomes the cost of the node. Otherwise, the children are retained and the latter becomes the cost of the node. One bit is kept at the node to indicate whether or not the children are pruned away. The routine then returns the cost of the node.

For an image $x_0, \ldots, x_{t-1}$, the Bayesian tree weighting produces

$$p(x_0, \ldots, x_{t-1}) = \sum_{i=0}^{M-1} p(f_i)p(x_0, \ldots, x_{t-1}|f_i)$$

where M is the number of possible tree probability models induced by possible tree prunings and assignments of models at the leaves. The value of $-\log p(f_i)$ is the resulting ideal code length of the pruned tree model f_i coded using the tree coding procedure used in the optimization method and the prior probabilities for the models at the leaves. Note that the sum includes $p(f_{opt})p(x_0, \ldots, x_{t-1}|f_{opt})$, where f_{opt} is the the optimal model according to the optimization process described above. Hence, for uniform scalar quantization, the weighting method will always be at least as good as the optimization method. If a very large number of the terms in the sum are of similar magnitude as the optimal term, then the weighting method will result in significantly better performance than the optimization method.

The tree weighting algorithm used here is a variant of the weighting algorithm presented in [5, 11] and is fully described in [7]. The models in each node is initialized to the prior distribution on C. At time t, the coefficient traces a path from the root to the leaf, hence we can partition the nodes in the quadtree T into those that are on the path and those that are not on the path. Let $\mathbf{a}_t$ be the probability vector produced by the algorithm for coding the coefficient at time t and let $\mathbf{a}^j = (a_0, \ldots, a_{b-1})$ be the probability vector associated with model $c_j \in C$ with an alphabet of size b. We code coefficient x_t using

$$\mathbf{a}_t = \sum_{j=1}^{N} W^t_{\text{total}}(r, j)\mathbf{a}^j$$

where r is the root of T. The function $W^t_{\text{total}}(u,j)$ is the sum of the weights of all the models generated by all possible prunings of the subtree of T rooted at u that have c_j as the path leaves. This function is calculated by

$$W^t_{\text{total}}(u,j) = \begin{cases} W^t_{\text{model}}(u,j) & \text{path leaf} \\ \frac{1}{2}W^t_{\text{model}}(u,j) & u \text{ path internal node,} \\ +\frac{1}{2}W^t_{\text{total}}(s,j)\prod_{v\in\text{children of } u, v\neq s} W^t_{\text{node}}(v) & s \text{ path node child of } u \end{cases}$$

where the the functions $W^t_{\text{model}}(u,j)$ and $W^t_{\text{node}}(u)$ are updated by

$$W^{t+1}_{\text{model}}(u,j) = \begin{cases} W^t_{\text{model}}(u,j) & \text{off path} \\ W^t_{\text{model}}(u,j)a^j_{x_t}/a_{t,x_t} & \text{on path} \end{cases}$$

and

$$W^{t+1}_{\text{node}}(u) = \begin{cases} W^t_{\text{node}}(u) & \text{off path} \\ \sum_{j=1}^N W^{t+1}_{\text{model}}(u,j) & \text{path leaf} \\ \frac{1}{2}\sum_{j=1}^N W^{t+1}_{\text{model}}(u,j) + \frac{1}{2}\prod_{v\in\text{children of } u} W^{t+1}_{\text{node}}(v) & \text{path internal node.} \end{cases}$$

All the calculations happen only along the path of the coefficient and hence the computational complexity for each coefficient is proportional to the length of the path. If each node is partitioned into four (approximately) equal sized children, the height of the tree is $O(\log n)$. Hence the total computational complexity is $O(n^2 \log n)$.

2.2 Windowing

When assigning a probability mass $a(x_i|x^{i-1})$, windowing considers only members of x^{i-1} that fall within a window of neighbouring pixels. We denote these members as $x^{i-1,w}$. Given N possible models, the weighting method sets

$$a(x_i|x^{i-1}) = p(x_i|x^{i-1,w}) = \frac{\sum_{j=1}^N p(c_j)p(x^{i-1,w}|c_j)p(x_i|c_j)}{\sum_{j1}^N p(c_j)p(x^{i-1,w}|c_j)}$$

for probability models $\{c_1,\ldots,c_N\}$, while the selection method uses the probability model that performs best in the window.

In this paper, we use a rectangular window of width $2w+1$, centered around the coefficient being coded with uniform probability for the a priori probability $p(c_j)$. Windowing should work well when the statistics of the image change slowly. However, when a large region can be coded optimally with the same probability model, windowing may be suboptimal since it only considers the statistics in a small region around the coefficient that is being coded.

2.3 Best Tiling

The best tiling method is derived using the specialist model, first proposed by Blum [2] and studied in [4]. Using the specialist model, a sequential probability assignment algorithm that has redundancy $O(k \log \frac{Nn}{k})$ with respect to the class of any tiling of an image with k arbitrary sized rectangular tiles was given in [7]. By redundancy, we mean the additional code length produced by the algorithm compared to the optimal tiling using k tiles of probability models. The bound holds regardless of the input sequence encountered and the value of k does not have to be known in advance by the algorithm. Hence, the algorithm should perform well whenever a tiling of the image which works well with a small number of tiles exists. The computational complexity of the algorithm is $O(Nn^3)$. If we restrict the comparison class to rectangles with W discrete widths, which is what we do in this paper, the computational complexity can be improved to $O(WNn^2)$.

We first consider the case of tiling an $n \times n$ image with rectangles of arbitrary height but one fixed width w. We assume that the coefficients are processed in a raster scan order. The origin is at the top left hand corner of the image, the vertical coordinate is y, the horizontal coordinate is z and the coordinates increase in the direction of the scan. Let $\mathbf{a}_{(y,z)}$ be the probability vector produced by the algorithm for coding the pixel (y, z) and let $\mathbf{a}^j = (a_0, \ldots, a_{b-1})$ be the probability vector associated with model c_j with an alphabet of size b. From [7], we have

$$\mathbf{a}_{(y,z)} = \frac{\sum_{j=1}^{N} Q_j^{(y,z)} \mathbf{a}^j}{\sum_{j=1}^{N} Q_j^{(y,z)}},$$

where $Q_j^{(y,z)}$ is updated by the following equations:

$$\begin{aligned}
Q_j^{(y,z+1)} &= R_j^{(y,z)} Q_j^{(y,z)} - R_j^{(y,z)} Z_j^{(y,z)}(y,z) + Z_j^{(y,z)}(y, z+w) \\
Z_j^{(y,z)}(y, z+w)) &= (n-y)\left(\frac{R_j^{(y-1,z+w)} Z_j^{(y-1,z+w)}(y-1, z+w)}{(n-y+1)} + 1\right) \\
R_j^{(y,z)} Z_j^{(y,z)}(y,z)) &= (n-y) R_j^{(y,z),w} \left(\frac{R_j^{(y-1,z)} Z_j^{(y-1,z)}(y-1,z)}{(n-y+1)} + 1\right) \\
R_j^{(y,z),w} &= \begin{cases} R_j^{(y,z)} & \text{if } z = 0 \\ R_j^{(y,z-1),w} R_j^{(y,z)} & \text{if } z < w \\ R_j^{(y,z-1),w} \frac{R_j^{(y,z)}}{R_j^{(y,z-w)}} & \text{if } w \le z \le n-1 \\ \frac{R_j^{(y,z-1),w}}{R_j^{(y,z-w)}} & \text{if } z > n-1 \end{cases} \\
R_j^{(y,z)} &= a^j_{x_{(y,z)}} / a_{(y,z),x_{(y,z)}}
\end{aligned}$$

The initial conditions needed are $Q_j^{(0,0)} = nw$ and

$$Q_j^{(y,0)} = \sum_{k=0}^{w-1} (n-y) \left(\frac{R_j^{(y-1,k)} Z_j^{(y-1,k)}(y-1,k)}{(n-y+1)} + 1\right).$$

With D distinct widths, we only have to run D distinct copies of the algorithm and sum the values of $Q_j^{(y,z)}$ for each j. A minor complication arises at the boundaries of the z coordinate since multiple rectangles of different widths which goes beyond the boundaries are in fact equivalent. One simple method of getting around this is to modify the algorithm for all values of widths w other than the largest width in such a way that $R_j^{(y,z)} Z_j^{(y,z)}(y,z) = 0$ for $z < w$, $Z_j^{(y,z)}(y, z+w) = 0$ for $z \geq n - w - 1$ and $Q_j^{(y,0)} = 0$.

3 Simulation Results

For all the simulations, we use 8 probability models in C: a uniform distribution and seven Laplacian distributions $p(x) = \frac{1}{2\beta_j} e^{-|x|/\beta_j}$ with $\beta_j \in \{1, 2, 4, 8, 16, 32, 64\}$ that has been discretized according to the quantization intervals. Uniform quantization with a dead band (bin containing zero is twice as large as the other bins) is used. Uniform prior distribution for the models in C is used in all cases. The tiling method is used with tiles of five different widths: 2,4,8,16 and 32. For windowing, we find that a value $w = 2$ performs best for most of the images, hence results are presented only for that value of w. For quadtrees (before pruning), nodes which contain no more than 2 rows or columns are considered as leaves. A five level decomposition with the 9-7 tap biorthogonal spline filters [1] is used.

We performed simulations on 24 images, 12 images of size 256×256 from *GreySet1* and 12 larger images from *GraySet2* of the Waterloo Bragzone (*http://links.uwaterloo.ca/bragzone.base.html*). The first six smaller images are synthetic and are shown in the first row of Figure 1 while the other six shown in the second row are natural images. The images from *GraySet2* are shown in Figure 2.

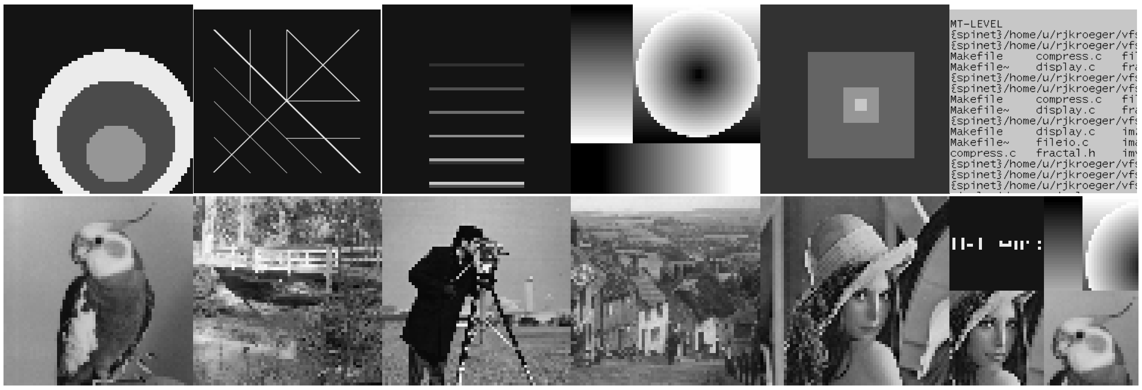

Figure 1: Synthetic images. From left to right, the images are *circles, crosses, horiz, slope, squares, text*. Natural images. From left to right, the images are *bird, bridge, camera, goldhill, lena, montage*.

The PSNR results for the synthetic images at 0.25 and 0.5 bits per pixel are shown the first halves of Table 1 and Table 2.

As expected, quadtree weighting performed uniformly better than optimal quadtree pruning. For windowing, weighting performed better for the higher entropy images while selecting performed better for the lower entropy images. Windowing and quadtree methods perform better for different images. The tiling method gave uniformly better result

Figure 2: Larger images. In raster scan order, the images are *barbara, boat, france, frog, goldhill, lena, library, mandrill, mountain, peppers, washsat, zelda*.

Image	Quadtree Prune	Quadtree Weight	Window Select	Window Weight	Tile	SPIHT
circles	31.47	32.10	32.07	32.48	32.63	**32.95**
crosses	26.08	26.43	27.55	27.55	27.55	**28.45**
horiz	43.22	44.38	44.72	43.49	50.23	**50.62**
slope	41.19	41.92	41.14	40.98	43.75	**44.07**
squares	52.24	53.32	52.41	50.90	57.93	**61.16**
text	14.93	14.97	14.39	14.89	**15.21**	14.40
bird	37.2	37.29	37.33	37.29	37.72	**37.75**
bridge	24.20	24.24	23.92	24.19	**24.34**	24.33
camera	27.43	27.67	27.55	27.79	**27.97**	**27.97**
goldhill	26.96	27.03	26.76	27.02	**27.19**	27.10
lena	28.57	28.74	28.52	28.79	**29.01**	28.96
montage	29.78	30.24	29.86	30.16	**31.02**	30.76

Table 1: PSNR values for GraySet1 at 0.25 bits per pixel.

than quadtree and windowing for this experiment. The tiling method performed significantly better for the images *horiz* and *squares*. Visual inspection of the images reveal that these images (and hence their corresponding wavelet coefficients) can be tiled using a small number of tiles. These results are in agreement with the theory which assures us that the algorithm will perform well whenever the subbands can be tiled using a small number of tiles. The tiling method performed similarly to windowing on the images *circles* and *crosses*. The image *circles* contains circular objects which can only be tiled using many rectangular tiles. Similarly, crosses contains many diagonal lines which again can only be satisfactorily covered using many rectangular tiles of probability models.

For the synthetic images, SPIHT performed significantly better than the other methods on the low entropy images. One reason for this is that the lowest entropy model we used is the Laplacian model $p(x) = \frac{1}{2\beta_j} e^{-|x|/\beta_j}$ with $\beta_j = 1$, hence we are unable to take advantage of lower entropy areas. Another possible reason is that interband dependencies may provide a significant amount of information which may not be available locally for these images.

Image	Quadtree Prune	Quadtree Weight	Window Select	Window Weight	Tile	SPIHT
circles	38.38	39.42	39.83	40.12	40.15	**41.65**
crosses	31.90	32.65	32.99	33.24	33.13	**34.93**
horiz	54.23	55.15	54.22	54.03	59.15	**72.89**
slope	49.51	49.66	49.51	49.26	50.44	**51.08**
squares	60.31	60.31	59.70	59.70	61.36	**inf**
text	17.09	17.16	16.20	16.97	**17.70**	17.04
bird	40.89	40.99	41.17	41.17	**41.36**	41.34
bridge	26.29	26.36	25.90	26.31	**26.50**	26.39
camera	30.93	31.16	30.97	31.39	**31.49**	31.47
goldhill	29.33	29.39	29.04	29.44	**29.59**	29.52
lena	32.30	32.49	32.38	32.71	**32.79**	32.74
montage	35.02	35.41	35.10	35.47	**36.29**	36.20

Table 2: PSNR values for GraySet1 at 0.5 bits per pixel.

The rest of the images tested are natural images. The PSNR results at 0.25 and 0.5 bits per pixel for *Greyset 1* are shown in the second halves of Table 1 and Table 2. The results for the images from *Greyset 2* are shown in Table 3 and Table 4.

Image	Size	Quadtree Prune	Quadtree Weight	Window Select	Window Weight	Tile	SPIHT
barbara	512×512	28.17	28.35	28.32	28.40	**28.57**	28.13
boat	512×512	30.57	30.74	30.57	30.69	**30.97**	**30.97**
france	496×672	23.00	23.28	22.71	23.22	**23.87**	22.91
frog	498×621	25.35	25.38	25.22	25.36	**25.43**	25.33
goldhill	512×512	30.34	30.41	30.22	30.42	**30.58**	30.56
lena	512×512	33.77	33.92	33.84	33.92	**34.12**	34.11
library	352×464	19.56	19.73	19.40	19.73	**20.17**	19.82
mandrill	512×512	23.18	23.24	22.94	23.20	**23.39**	23.27
mountain	480×640	19.25	19.32	18.99	19.31	**19.46**	19.37
peppers	512×512	32.87	33.05	33.07	33.16	33.45	**33.47**
washsat	512×512	34.14	34.14	33.84	34.07	**34.21**	34.18
zelda	512×512	37.34	37.39	37.34	37.31	**37.51**	37.50

Table 3: PSNR values for GraySet2 at 0.25 bits per pixel.

Again quadtree weighting is uniformly better than quadtree pruning. However, for almost all the images, the advantage of quadtree weighting is rather small. For windowing, weighting performed better on almost all the images but again the margin is rather small. The performance of windowing and quadtrees are rather similar for the natural images, with different methods being slightly better on different images. Tiling uniformly outperformed both quadtrees and windowing on these experiments but the margin is rather small for almost all the images. The exceptions include *montage*, *france* and *library* which contains either strong vertical and horizontal components or large areas of low entropy.

Image	Size	Quadtree Prune	Quadtree Weight	Window Select	Window Weight	Tile	SPIHT
barbara	512×512	31.98	32.10	32.19	32.27	**32.38**	32.11
boat	512×512	33.97	34.13	34.03	34.25	34.41	**34.45**
france	496×672	27.55	27.85	27.37	27.85	**29.10**	28.04
frog	498×621	26.60	26.61	26.25	26.56	**26.66**	26.64
goldhill	512×512	32.89	32.98	32.76	33.01	**33.16**	33.13
lena	512×512	36.86	36.97	36.90	37.09	37.17	**37.21**
library	352×464	22.31	22.51	22.09	22.53	**22.99**	22.75
mandrill	512×512	25.52	25.58	25.29	25.60	**25.76**	25.65
mountain	480×640	21.23	21.31	20.83	21.27	**21.48**	21.44
peppers	512×512	35.47	35.58	35.50	35.70	35.82	**35.92**
washsat	512×512	36.08	36.13	35.76	36.13	**36.22**	36.19
zelda	512×512	39.48	39.54	39.38	39.54	39.59	**39.66**

Table 4: PSNR values for GraySet2 at 0.5 bits per pixel.

The performance of tiling is similar to the performance of SPIHT, with significant advantage only in the image *france*.

4 Discussion

In this paper, we used only simple Laplacian distributions as our models. More complex distributions such as the Generalized Gaussian Distributions used in [8] may be able to perform better. An equally interesting possibility is to switch between compression algorithms instead of static probability distributions, as is done in [10]. More sophisticated quantization methods such as trellis-coded quantization [6] may also improve performance. It is straight forward to use such quantizers with forward adaptation methods which send the models before the quantized coefficients (such as optimal quadtree pruning). Unfortunately, we do not know how to perform forward adaptation efficiently with the class of k-rectangular tiling of the image.

5 Conclusions

We have compared the performance of quadtree, windowing and best tiling for for compression of wavelet coefficients. As expected from the theoretical results, weighting outperformed pruning for quadtree compression of wavelet coefficients quantized with a single quantizer but the margin of improvement is small. Weighting is also slightly better than selecting for windowing methods on most natural images. The performance of the best tiling method in our simulations agrees with the theoretical result which suggests that the method will work well when the image can be tiled using a small number of rectangular tiles. Best tiling worked better than the other methods on the experiments we performed but the improvement is significant for only a few images.

6 Acknowledgements

The image coders used in this paper were modified from the *Wavelet Image Compression Construction Kit*, written by Geoff Davis, John Danskin and Ray Heasman (http://www.cs.dartmouth.edu/ gdavis/wavelet/wavelet.html).

References

[1] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Transactions on Image Processing*, pages 205–220, April 1992.

[2] Avrim Blum. Empirical support for winnow and weighted-majority based algorithms: results on a calendar scheduling domain. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 64–72, 1995.

[3] C. Chrysafis and A. Ortega. Efficient context-based lossy wavelet image coding. In *Proc. of Data Compression Conference*, 1997.

[4] Yoav Freund, Robert E. Schapire, Yoram Singer, and Manfred K. Warmuth. Using and combining predictors that specialize. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, 1997.

[5] David P. Helmbold and Robert E. Schapire. Predicting nearly as well as the best pruning of a decision tree. *Machine Learning*, 27(1):51–68, 1997.

[6] R. L. Joshi, H. Jafarkhani, J. H. Kasner, T. R. Fischer, N. Farvardin, M. W. Marcellin, and R. H. Bamberger. Comparison of different methods of classification in subband coding of images. *IEEE Transactions on Image Processing*, 6:1473–1486, November 1997.

[7] Wee Sun Lee. Tiling and adaptive image compression. Submitted. `http://www.comp.nus.edu.sg/~leews/publications/tiling_rev.pdf`, 1999.

[8] S. M. LoPresto, K. Ramchandran, and M. T. Orchard. Image coding based on mixture modelling of wavelet coefficients and a fast estimation-quantization framework. In *Proceedings of the Data Compression Conference*, pages 221–230, 1997.

[9] A. Said and W.A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Trans. on Circuit and Systems for Video Technology*, 6(3):243–249, 1996.

[10] Paul A. J. Volf and Frans M. J. Willems. Switching between two universal source coding algorithms. In *Data Compression Conference*, pages 491–500, 1998.

[11] F. M. J. Willems, Y. M. Shtarkov, and T. J. Tjalken. The context-tree weighting method: Basic properties. *IEEE Trans. on Information Theory*, 41(3):653–664, May 1995.

[12] Frans M. J. Willems. Coding for a binary independent piecewise-identically-distributed source. *IEEE Transactions on Information Theory*, 42(6):2210–2217, November 1996.