[Appears in *Mathematical Morphology and its applications to image and signal processing: Proceedings of the Fifth International Symposium on Mathematical Morphology*, pages 361–370, Palo Alto, California, 2000. Kluwer.]

TWO-STAGE LOSSY/LOSSLESS COMPRESSION OF GRAYSCALE DOCUMENT IMAGES

KRIS POPAT and DAN S. BLOOMBERG
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
{popat,bloomberg}@parc.xerox.com

Abstract. This paper describes a two-stage method of document image compression wherein a grayscale document image is first processed to improve its compressibility, then losslessly compressed. The initial processing involves hierarchical, coarse-to-fine morphological operations designed to combat the noiselike variability of the low-order bits while attempting to preserve or even improve intelligibility. The result of this stage is losslessly compressed by an arithmetic coder that uses a mixture model to derive context-conditional graylevel probabilities. The lossless stage is compared experimentally with several reference methods, and is found to be competitive at all rates. The overall system is found to be comparable with JPEG in terms of mean-square error performance, but appears to outperform JPEG in terms of subjectively judged document image intelligibility.

Key words: document image compression, image morphology, arithmetic coding, multiresolution, Gaussian mixtures

1. Introduction

Grayscale scanning offers several advantages over binary scanning in terms of image quality and downstream flexibility, even when the documents being scanned are binary. The downside is the need to capture, process, and store much more information. Compression of the scanned grayscale document images is therefore of great importance.

JPEG [11] is the most widely used method for compressing natural scenes, but introduces undesirable artifacts around the sharp edges found in document images, particularly within text and line-art regions. Recently, methods have been proposed in which different segments of a document image are encoded using different techniques; a good example is the DjVu format developed by AT&T [6].

While it is advantageous to use segmentation information to adapt compression locally, it is not necessary to encode the segments separately. In particular, the approach proposed here is to use the segmentation information to switch among probability models used with arithmetic coding, thus allowing the encoding to be carried out on a single raster layer. The segmentation information must be transmitted, but this requires few bits relative to those required for the image itself. We propose a two-stage approach: (a) morphologically-based segmentation and a one-time lossy transformation to improve compressibility and maintain intelligibility, and (b) lossless compression by arithmetic coding. Section 2 describes the lossy transformation and segmentation method, Section 3 describes the lossless compression, and Section 4 presents some experimental results.

2. Lossy Stage

Unlike conventional lossy compression techniques, such as JPEG, which performs lossy quantization and lossless encoding of the coefficients atomically, here we separate the lossy step from the lossless one. Because the lossy step is performed in the image domain, rather than the transform domain, the change in visual appearance can be controlled by minimizing the maximum pixel value change.

Most of the entropy in a scanned grayscale image is in the low-order bits (LSBs) of each pixel. Because these bits are the least visible, they can be set to zero by rounding. For example, for 8 bit pixels, three LSBs can be set to zero by adding 4 (binary 100) and truncating to the 5 MSBs, taking care to avoid overflow. To take account of neighboring correlations, we use a multiscale approach were four pixels at one scale are compared and are either left unchanged or averaged with post-rounding. The thresholds used for the comparison at each scale can be chosen either to preserve low-contrast features (such as bleed-through) or to remove them.

2.1. 'Pyramid Scheme'

For document images, it is important to represent the pixels in transition regions between light and dark (i.e., at edges) with fidelity. This is accomplished in the following manner. The lossy stage first rounds the n_r LSBs of all pixels at full resolution to 0. Then it generates a pyramid of n_d reduced images of dimension 2^{-n_d} relative to the original. At each $2 \times 2 \rightarrow 1 \times 1$ stage, the image is tiled into 2×2 pixels and the maximum deviation from the average within the tile is compared to a level-dependent threshold. If the deviation is smaller than the threshold, a single pixel is saved with the average value (again with n_r LSBs rounded to 0); otherwise, the pixel is marked with the value 1, which is distinguished from all possible rounded average values. Beyond the first stage in the pyramid, pixels with value 1 can be encountered and are ignored in the averaging process.

After the reduced images are generated, the average values are propagated back up the chain. Consider the propagation from level m to $m-1$. If a 1 is encountered at level m, then the four corresponding pixels at $m-1$ are left unchanged. Otherwise, any of the four pixels at $m-1$ that are not 1 are set to the pixel value from level m.

The parameters in the encoder are thus: n_r, the number of LSBs rounded at each stage to 0; n_d, the number of reduced images generated in the pyramid; and $\{t_m, m = 1, ...n_d\}$, the thresholds set for each level. The thresholds and n_r cannot be chosen independently, and a workable choice is $t_m = 2^{n_r}$ for $m \leq 2$ and $t_m = 2^{n_r - 1}$ for $2 < m \leq n_d$.

It may be desirable to choose different encoder parameters for text and halftone regions. For example, with large values for n_r, n_d and the thresholds, the text regions are highly compressible, with smoothed background and removal of bleed-through. However, such parameters can cause visible contouring and smearing in halftone regions. Instead, we may want a larger number of gray levels, albeit at low resolution, and might choose smaller values of n_r, n_d and thresholds. Unlike the text regions, where significant added compressibility is achieved by the multiresolution operations, in halftone regions most of the compressibility is due to the initial rounding.

2.2. Segmentation

To apply different parameters to halftone and text regions, and to allow the lossless stage to switch to a probability model appropriate for the category and choice of parameters, it is necessary for the encoder to generate a segmentation mask. There are many methods for generating a mask covering the halftone regions, and we describe a particularly efficient morphologically-based one that uses a binarized version of the image, generated from a global threshold.

The threshold can be chosen from a (subsampled) histogram of image pixels. If there is a significant amount of text or line-art, the set of background pixels will be evident in the histogram. A global threshold for projecting the foreground pixels can then be chosen at the dark edge of this set, by placing the pixels in overlapping histogram bins, finding the darkest bin containing a sufficient fraction of all pixels, and choosing a value near the minimum (dark) boundary of this bin.

If no threshold value is found, no segmentation is performed, and the image is compressed as halftone. Otherwise, the image is pixelwise lowpass filtered using the threshold, giving a *foreground mask* binary image. From this binary image, a *halftone seed* is derived morphologically, and a *halftone mask* is generated by binary reconstruction into the foreground mask from the seed. The seed is generated by a series of closings and openings, which is efficiently carried out on an image pyramid using a sequence of threshold reductions[3], along with closings and openings. Threshold reductions with threshold values of 1 and 4 are equivalent to dilations and erosions with a 2x2 structuring element, respectively, followed by subsampling.

3. Lossless Compression

The processing described in the previous section modifies a document image to make it more compressible, without changing it so much as to detract from its aesthetic appeal or its intelligibility. The remaining problem is to represent the resulting array of cleaned-up pixels in a compact manner. Although in some circumstances we may wish to consider encoding methods which incur further loss, for simplicity we assume that the lossy stage has resulted in precisely the image we wish to represent, and accordingly restrict consideration to lossless compression methods. Throughout this section, the unqualified terms “image” and “pixel” will refer to the result of the lossy stage.

3.1. Why Not Generic Lossless Compression?

One possibility would be to use a generic compression routine such as the *gzip* program [5]. This approach has the appeal of simplicity and robustness, but does not take full advantage of what is known in advance about the statistics of the specific class of images being encoded. Still, this approach can be effective, provided that the two-dimensional pixel array is represented as a one-dimensional sequence in an appropriate manner. For instance, by encoding differences between pixels values in an appropriate way, the Portable Networks Graphics (PNG) format can achieve good compression on graphics images; see the discussion about "filters" in [1].

Generic compressors like *gzip* function by discovering patterns in the data presented to them, but they do not interpret the data in order to draw reasonable generalizations about similar patterns. Specifically, they do not exploit the following *smoothness* property of the joint probability law that can be thought to govern pixel neighborhoods: *small changes in pixel amplitudes in a pattern correspond to small changes in the probability assigned to that pattern.* This property suggests that an advantage is to be had by sharing statistics among patterns that are deemed similar.

3.2. Conditioning Contexts and Mixture Models

We thus consider techniques wherein approximate pattern matches are used when learning the statistical regularity upon which the compression will be based. Arithmetic coding [14, 13] offers a convenient means of separating the statistical modeling task from the actual compression task without giving up performance, thereby allowing the use of specialized statistical models capable of exploiting the above-mentioned smoothness property. For practical reasons, the pixels are processed sequentially rather than in the aggregate, but within that constraint, the use of arithmetic coding allows us to freely specify any statistical model. The remaining constraint is that the statistical model may be conditioned only on preceding pixels in the chosen ordering.

The conditioning structure of the statistical model we consider is patterned after the grayscale extension [10] of the causal-neighborhood context model originally proposed in [7] for binary images. Specifically, for every pixel location in the sequence, a set of nearby but strictly preceding pixel locations is specified as a conditioning context. We consider the simplest case, wherein the pixels are encoded in raster order and the set of conditioning pixels is specified relative to each encoded pixel by a constant causal context neighborhood template (see Figure 1). [1] An estimate of the conditional density is obtained by appropriately normalizing a Gaussian mixture estimate of the joint density of the conditioning pixels and the pixel being encoded. Although normalizing a joint mixture estimated in this way generally does not result in the best conditional density estimate of comparable complexity [9], this approach is

[1] Hierarchical, coarse-to-fine sequencing is also possible, but multiple statistical models must then be employed, and the conditioning neighborhoods required become more complex. Furthermore, preliminary results have not demonstrated a clear performance advantage in the present application that might offset this added complexity.

Fig. 1. Two examples of causal context neighborhoods. Solid dots indicate conditioning pixels, while the unfilled dots indicate the pixel currently being encoded or decoded.

conceptually simple, and in practice has often been found to perform adequately relative to more involved estimation methods.

After the lossy stage, the image pixels assume values in a relatively small set. For example, for eight-bit original images and when $n_r = 3$, there are only thirty-one possible values for each pixel: 0, 8, 16, ..., 248. Such coarse discretization is slightly at odds with the smoothness property mentioned earlier, but we wish to exploit smoothness for the generalization benefit it offers, and resort to the following artifice. We imagine that the value of each pixel represents an independent quantization of a hypothetical continuous valued pixel. The mixture is used to model the conditional density of this continuous valued pixel, conditioned on specific previous quantized pixel values. The probability mass function provided to the arithmetic coder is obtained by integrating this conditional density over each quantization region. Since the hypothetical continuous-valued pixel is unavailable for training the mixture, the quantized values are used instead, after adding to each a small amount of uniformly distributed noise. Independent quantization of individual pixels is an imperfect model of the lossy stage, as it does not account for the important spatial interaction that occurs there. Nevertheless, we have found that it is a useful model for deriving a probability mass function for arithmetic coding, as is borne out in the results presented in Section 4.

3.3. Details of the Lossless Compression Method

The pixels are always processed in raster order. Let x denote the current pixel being encoded, and let $(y_1, \ldots, y_N)$ denote a vector of preceding conditioning pixels specified by a fixed context neighborhood of the type shown in Figure 1. A set of training images, each deemed similar in nature to the image segment to be encoded and each processed by the lossy stage using the same parameter values, is determined. For instance, if the cleaned-up image segment to be encoded is a line drawing, the training images selected should also be line drawings, processed using the same parameters in the lossy stage.

These training images are scanned by sliding the context neighborhood along the image and, at every pixel location for which the entire neighborhood lies within the image boundaries, assembling the values indicated by the neighborhood into a vector. In this way, a collection of training vectors $\{(x, y_1, \ldots, y_N)_i,\ i = 1, \ldots, T\}$ is obtained. The number of training vectors T is targeted to be $T \approx 100K$, where K is the number of components to be used in the mixture model. To control T, the training images are chosen to be of sufficient size and number to yield a number of vectors somewhat larger than the desired T, then subsampling is used to reduce the number to the desired value. Pseudorandom noise distributed uniformly on $(-\alpha\delta/2, \alpha\delta/2)$ is added to each coordinate of every vector thus obtained, where δ is the minimum spacing between any two graylevels in the training images, and $\alpha \in [0, 1]$ is a parameter that controls the amount of noise added. Note that this noise is added only to the training vectors for the purpose of more robustly fitting the mixture model; it is not used after the model has been fit, i.e., when images are actually encoded and decoded.

The mixture model consists of K separable Gaussian components, where K is a parameter that controls model complexity. Because the training and test image sets are disjoint, choosing too large a K (i.e., one that causes overfitting) would be signalled by poor performance on the test images. The estimation of the mixing proportions and component density parameters is accomplished via the expectation-maximization algorithm [12]. After training, the mixture model $\hat{p}(x_c, y_1, \ldots, y_N)$ is used to provide a conditional density estimate

$$\hat{p}(x_c | y_1, \ldots, y_N) \propto \hat{p}(x_c, y_1, \ldots, y_N)$$

for each (hypothetical) continuous-valued pixel in a new image, where the $y_1, \ldots, y_N$ are now regarded as fixed. To obtain a probability mass estimate for each actual (i.e., quantized) pixel value x', $\hat{p}(x_c | y_1, \ldots, y_N)$ is integrated over the quantization region that supports that value. Specifically, we use the estimate

$$\hat{\Pr}(x = x' | y_1, \ldots, y_N) = \int_{x' - \delta/2}^{x' + \delta/2} \hat{p}(x_c | y_1, \ldots, y_N) dx_c$$

for each quantized pixel value x', with the exceptions that the lower integration limit for the smallest x' is set to $-\infty$ and the upper integration limit for the largest x' is set to ∞.

For a given conditioning vector $(y_1, \ldots, y_N)$, let $p(x)$ denote the probability mass function $\hat{\Pr}(x = x' | y_1, \ldots, y_N)$. For use in arithmetic coding, $p(x)$ is approximated by a fixed-precision probability mass function $q(x)$ such that $q(x) > 0\,\forall x$. For a given $p(x)$, we choose $q(x)$ from the feasible set to minimize the expected ideal bit rate $-\sum_{x'} p(x') \log_2 q(x')$, which is equivalent to minimizing the relative entropy [4] between $p(x)$ and $q(x)$.

Near the borders of the image, some of the conditioning pixels will lie outside the image. In order to still have a conditional density estimate in such cases, the estimate $\hat{p}(x_c, y_1, \ldots, y_N)$ is integrated over the coordinates corresponding to the unavailable conditioning pixels, and the result is used to obtain the desired conditional density estimate as described above.

Although we explicitly account for the quantized nature of x when obtaining $\hat{\Pr}(x = x' | y_1, \ldots, y_N)$, we do not make any adjustment for the fact that the conditioning pixels are also quantized. However, little is at stake here: the consequence of quantizing the conditioning pixels is to perturb the location of the line parallel to the x-axis in $(x, y_1, \ldots, y_N)$-space along which the conditional density is evaluated. If the perturbation is small, then the smoothness property mentioned in Section 3.1 suggests that the resulting change in the estimated pixel probabilities will likewise be small. In contrast, accounting for the quantized nature of x is essential, as the probability mass assigned to the various values that x assumes directly determines the number of bits required for encoding those values.

Arithmetic coding is carried out as described in [8], using a code-register precision of 16 bits and a 15-bit-wide carry-control register.[2] This allows the two registers to coexist as fields in a single 32-bit hardware register without involving the sign bit. When an image is encoded, a header is first created indicating: the dimensions of the image, the minimum spacing δ between graylevels, and the minimum and maximum graylevels that occur in the image. Since each of these quantities is a relatively small integer, the number of bits needed for this header is negligible compared with the number needed for the rest of the image. After the last pixel has been encoded, the code- and carry-control registers are flushed into the code bit stream and the procedure terminates. The bits thus produced are packed into bytes and written into a binary file; this file constitutes the compressed representation of the image. It has been confirmed experimentally that decoding this compressed representation results in an exact bit-for-bit replica of the image.

4. Results and Conclusions

Experiments were performed on a set of nine 512×512 subimages of 300 dpi, 8 bpp grayscale scans of selected pages of the March 1998 issue of the IEEE Transactions on Image Processing. For clarity, we compress segments of various types separately, rather than switching models on the basis of the segmentation mask as would be done in a practical system. Accordingly, three subimages were selected in each of the following categories: text, line drawings, and halftone. For each image in each category, the compression technique described in this paper was applied, using the other two images in the category to train the mixture model.

To find suitable combinations of parameter values for the lossy stage, all combinations of the parameter values $n_r \in \{2, \dots, 6\}$, $n_d \in \{3, 4, 5\}$, and threshold sequences among those listed in Table I were applied to each of the nine images. The compressibility of each result was estimated by the minimum of the bit rates obtained over all techniques described in Appendix A. The minimum rate and mean-square error (MSE) together define a point in the rate-MSE plane for each of the sixty candidate parameter combinations for each image. The convex hull of these points was determined using the *qhull* program [2] for each original image, and the parameter value combinations that appeared most frequently among vertices of the facets facing the origin were adopted for use in testing the lossless stage. The $(n_r, n_d, \{t_m\})$ triples found in this way were: (5,3,3), (4,3,1), (3,3,2), (3,3,3), and (2,3,3). Several context

[2] Other versions of arithmetic coding could be used as well, but care must be taken to properly interface the statistical model described above to the coder, particularly if conversion to an intermediate binary source representation is required for the coder to operate.

TABLE I
Threshold sequences used in lossy stage.

Label	t_m, $m = 1, \ldots, 6$
1	$\{16, 8, 8, 8, 8, 8\}$
2	$\{8, 8, 4, 4, 4, 4\}$
3	$\{4, 4, 4, 4, 4, 4\}$

neighborhoods and values of K and α were tried for the lossless stage; no one set worked best across all images. Figure 2 shows the MSE performance of the overall approach, averaged across images in each category using leave-one-out crossvalidation, for $K = 256$, $\alpha = 0.5$, and the context neighborhood shown in Figure 1a. Significantly larger values of K were found to result in overfitting. Also presented in Figure 2 are results for the lossless methods described in Appendix A, and for JPEG applied to the original images using several different quality factors. It can be seen that the proposed lossless technique is competitive at all rates, and outperforms most of the reference techniques at all but the highest bit rates.

JPEG can be seen to generally outperform the proposed technique in Figure 2. However, MSE is not the whole story, as larger MSE can correspond to improved intelligibility. An example of this is in Figure 3, where JPEG is more faithful to the original scan, but the proposed technique results in greater intelligibility by removing bleed-through and preserving edges.

Based on these results, we tentatively conclude that the proposed technique can be an attractive alternative to established methods when compressing grayscale document images, as it offers good MSE performance while maintaining or even improving intelligibility. The lossless stage is interesting in its own right, as it appears to outperform several standard approaches, particularly when the image supplied to it has low complexity.

Appendix

A. Lossless Image Compression Techniques used for Reference

The following lossless compression techniques for grayscale images were used for reference purposes in this study.

– BTPC: Binary Tree Predictive Coding (Version 4.1) by John A. Robinson. The program used (in lossless mode) was `cbtpc`, available from http://www.engr.mun.ca/~john/btpc.html.
– CALIC: Context-based, Adaptive, Lossless Image Coder (arithmetic coding version) by Xiaolin Wu and Nasir Memon. The program used was `enCALICa.sun`, available from ftp://ftp.csd.uwo.ca/pub/from_wu/v.arith.
– HIST: A rough measure of compressibility obtained by computing the entropy of the normalized image histogram, treating each pixel indepen-

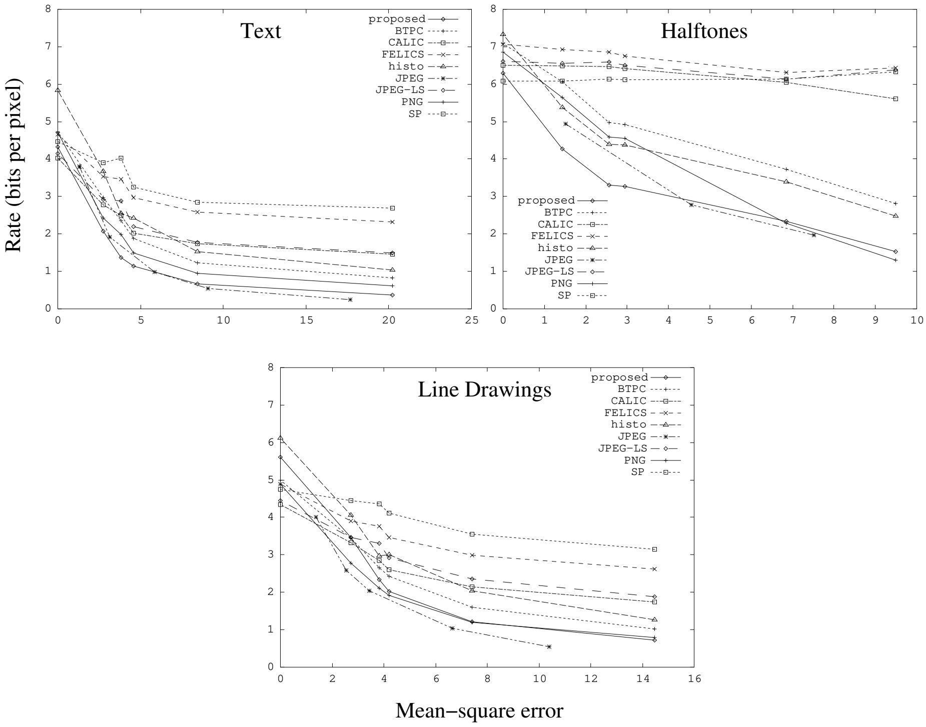

Fig. 2. Average crossvalidated MSE-Rate performance for each of the three image categories.

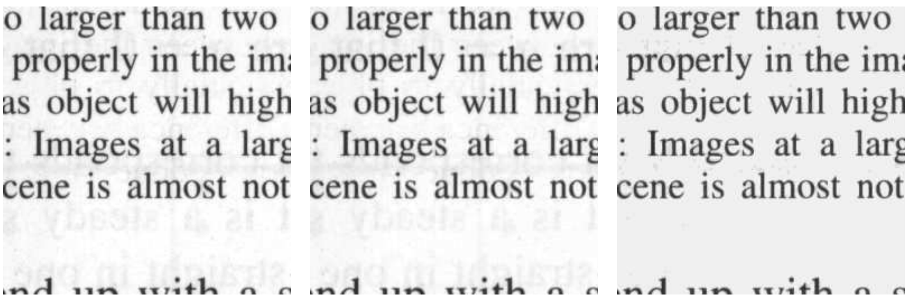

Fig. 3. Left: A 256×256 patch from one of the original text images. Middle: result of JPEG compression at 0.97 bpp (MSE = 5.84). Right: result of proposed scheme at 0.78 bpp (MSE = 7.58).

dently.

- FELICS: Fast, Efficient, Lossless Image Compression System by Paul G. Howard and Jeffrey Scott Vitter. The program used was `mgfelics`, available from http://www.cs.mu.oz.au/mg/mg-1.2.1.tar.gz.
- JPEG-LS: JPEG-LS Reference Encoder, Hewlett-Packard LOCO-I imple-

mentation. The program used was `locoe`, available from
http://www.hpl.hp.com/loco/software.htm

- PNG: The `pnmtopng` open-source program (version 2.37.1) by Alexander Lehmann, Willem van Schaik, and Greg Roelofs. Available in ftp://swrinde.nde.swri.edu/pub/png/applications. This program is based on the Portable Networks Graphics (PNG) library [1].
- SP: Arithmetic coding version of the lossless image compression program by Amir Said and William A. Pearlman. The program used was `sp_compress`, available from
http://www.cipr.rpi.edu/research/SPIHT/EW_Code/lossless.tar.gz.

References

1. M. Adler, T. Boutell, C. Brunschen, A. M. Costello, L. D. Crocker, A. Dilger, O. Fromme, J. Gailly, C. Herborth, A. Jakulin, N. Kettler, T. Lane, A. Lehmann, C. Lilley, D. Martindale, O. Mortensen, K. S. Pickens, R. P. Poole, G. Randers-Pehrson, G. Roelofs, W. van Schaik, G. Schalnat, P. Schmidt, T. Wegner, and J. Wohl. Png (portable network graphics) specification, version 1.0. Technical report, World Wide Web Consortium (W3C), October 1996. http://www.w3.org/TR/REC-png.
2. C. Barber, D. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Trans. on Mathematical Software*, December 1996. http://www.geom.umn.edu/software/qhull.
3. D. S. Bloomberg. Multiresolution morphological analysis of document images. In *SPIE Conf. 1818, Visual Communications and Image Processing*, pages 648–662, Boston, 1992.
4. T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley and Sons, 1991.
5. J. Gailly and M. Adler. The gzip homepage, December 1999. http://www.gzip.org/algorthm.txt.
6. P. Haffner, L. Bottou, P. Howard, P. Simard, Y. Bengio, and Y. LeCun. Browsing through high quality document images with djvu. In *Proc. of Advances in Digital Libraries 98.* IEEE, 1998.
7. G. G. Langdon and J. J. Rissanen. Compression of black-white images with arithmetic coding. *IEEE Trans. Comm.*, COM-29:858–867, June 1981.
8. A. C. Popat. Scalar quantization with arithmetic coding. Master's thesis, Dept. of Elec. Eng. and Comp. Science, M.I.T., Cambridge, Mass., 1990. ftp://ftp.media.mit.edu/pub/k-arith-code.
9. A. C. Popat. *Conjoint Probabilistic Subband Modeling.* PhD thesis, Massachusetts Institute of Technology, 1997.
10. K. Popat and R. W. Picard. Cluster-based probability model applied to image restoration and compression. In *ICASSP-94: 1994 International Conference on Acoustics, Speech, and Signal Processing*, pages 381–384, Adelaide, Australia, April 1994. IEEE.
11. M. Rabbani and P. W. Jones. *Digital Image Compression Techniques.* SPIE Optical Engineering Press, Bellingham, Washington, 1991.
12. R. A. Redner and H. F. Walker. Mixture densities, maximum likelihood, and the EM algorithm. *SIAM Review*, 26(2):195–239, April 1984.
13. J. J. Rissanen and G. G. Langdon. Arithmetic coding. *IBM J. Res. Develop.*, 23(2):149–162, March 1979.
14. I. Witten, R. Neal, and J. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, June 1987.