UNIVERSITY OF JOENSUU
DEPARTMENT OF COMPUTER SCIENCE
Report Series A

Forward-adaptive variant of the baseline JBIG for applications requiring spatial access

Eugene I. Ageenko and Pasi Fränti

Report A-1998-2

ACM I.4.2
UDK 681.3.06
ISSN 0789-7316
ISBN 951-708-617-2

Forward-adaptive variant of the baseline JBIG for applications requiring spatial access

Eugene I. Ageenko
Pasi Fränti

Department of Computer Science, University of Joensuu
P.O. Box 111, FIN-80101 Joensuu, FINLAND
Phone: +358-13-251 3103
Fax: +358-13-251 3290
Email: franti@cs.joensuu.fi

Abstract: A method for compressing large binary images is proposed for applications where spatial access to the image is needed. The proposed method is a two-pass combination of forward-adaptive modeling and backward-adaptive context based compression with re-initialization. The method supports spatial access and it improves the compression performance significantly if the baseline JBIG is used in the same way. The method can be implemented with only small modifications to the baseline JBIG; current software implementation can therefore be easily utilized.

Indexing terms: image compression, document imaging, context modeling, JBIG.

1. Introduction

Large size images are nowadays widely used in applications such as *Geographic Information System* (GIS) [1]. When an image is accessed, the entire file is typically read and decompressed into memory. This is not possible if the uncompressed raster image size exceeds the available memory resources. Besides, high-speed channels are not always available and the transmission of the entire document might cause inconvenient delays. Thus, the decompression of the entire image can be a major source of inefficiency. At the same time, only a small part of the image is often needed, or the image is processed and/or viewed fragment by fragment [2].

Spatial access to the image allows user to operate directly on the compressed data without retrieving the entire image. It is considered as one of the main requirements for document imaging applications [3]. Spatial access can be supported by *tiling* – dividing the image into fixed size rectangular blocks, which are denoted here as *clusters*. A *cluster index table* is constructed from the pointers indicating where the data of each cluster is located in the compressed file. The index table is stored at the beginning of the compressed file. To access any part of the image, only clusters consisting of the desired pixels need to be requested and decompressed.

Binary images can be stored compactly using the JBIG compression algorithm, which consists of adaptive context-based modeling and arithmetic coding [4]. Unfortunately JBIG does not support tiling, as the entire image prior to the accessed part must be decompressed. JBIG includes optional "*striping*" mode, which divides the image into horizontal stripes with resetting the statistics after coding each stripe. This solution, however, divides the image only in one dimension. An alternative solution is to divide the image into clusters of size $C \times C$ and apply JBIG to them separately. We shall further denote such combination of tiling and JBIG as *T-JBIG*.

A problem of the straightforward combination of tiling and JBIG is the high learning cost. JBIG starts from scratch and its model adapts to the statistics of the image during the compression. In principle, the adaptation is fast and the learning cost in the early stage of compression is relatively low. However, the effect of the learning cost increases essentially when coding smaller sizes of data – e.g. small clusters. Better initial model should therefore be applied to overcome the learning cost problem.

Here we propose a forward-adaptive variant of JBIG for tiling purpose. The method is a two-stage combination of forward-adaptive and backward-adaptive modeling. Statistics are first collected globally over the image for constructing better initial model. The model is stored in the compressed file. In the second stage, backward-adaptive context based compression (baseline JBIG) is applied with re-initializing the statistics in the beginning of each cluster. This technique allows for more efficient clustering because of faster adaptation and smaller learning cost. We shall further denote the forward-adaptive variant as *FA-JBIG*.

The *FA-JBIG* method supports tiling and spatial access without sacrificing the compression efficiency. Cluster sizes as low as 128×128 pixels can be applied with similar compression performance. The method even improves the compression performance of the baseline JBIG by about 4 % when cluster size is chosen optimally (384×384 for a set of test images). The method can be implemented with only small modifications to the baseline JBIG; current software implementation can therefore be easily utilized.

The rest of the paper is organized as follows. The baseline JBIG is recalled in Section 2. The forward-adaptive variant is then introduced in Section 3. Its effect on the compression performance is studied analytically in Section 4 and empirically in Section 5. Conclusions are drawn in Section 6.

2. Baseline JBIG

In the baseline JBIG pixels are coded by arithmetic coding in scan raster order using *context-based probability model* and *arithmetic coding* [4]. The combination of already coded neighboring pixels defines the context. In each context the probability distribution of the black and white pixels are adaptively determined. The current pixel is then coded by *QM-coder* [5, 6], the binary arithmetic coder adopted in JBIG. Statistics start from scratch and they are updated after the coding of each pixel. The model thus dynamically adapts to the image statistics during the coding process. This backward-adaptive modeling of JBIG has the advantage that only one pass over the data is needed and no overhead (such as models or code tables) needs to be stored in the compressed file.

The probability estimation of JBIG is renormalization driven and based on the Bayesian estimation concept. It is directly embedded in the arithmetic-coding procedures of *QM-coder* and implemented as a state automaton consisting of 113 states, each assigned by a different probability for the *least probable symbol* (LPS). Each context has its own pointer to the automaton, and an extra bit to indicate which color is the LPS. When the compression starts, the pointers of all contexts are initialized to the *zero-state* (state with the index 0) of automaton having approximately equal probabilities of MPS and LPS. When the coding proceeds, the probabilities are updated using a stochastic renormalization technique.

The states of the automaton are organized in chains as shown in Fig. 1. The automaton has mirror symmetry about the change in the sense of LPS, so it consists of 226 states in total but we consider only the one half. The chains are ordered according to the level of adaptation. The states in the upper chains are more sparsely distributed throughout the probability range and therefore they allow faster adaptation. The adaptation starts from the zero-state. After each MPS renormalization, a transition is made to the next state in right in the chain (with smaller LPS probability). After each LPS renormalization, a transition is made to the appropriate state in the next level chain (with larger LPS probability) or to the predecessor state in the same chain (in case of non-transient states). Transient states are visited only during the learning stage and eventually the pointers stabilize to the non-transient states. If the statistics change later the model re-adapts within the non-transient states.

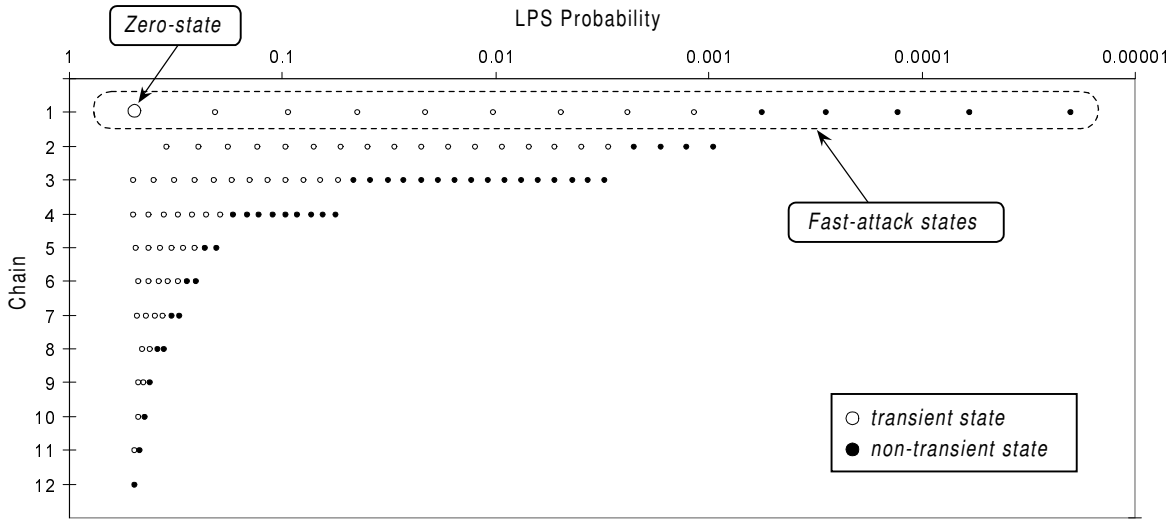

Fig. 1. Spatial organization of the state automaton of QM-coder.

3. Forward-adaptive variant

In tiling, the image is divided into clusters of $C \times C$ pixels and each cluster is compressed independently from each other. This allows independent decompression of the clusters and therefore it provides spatial access to the compressed image file. In the forward-adaptive variant we compress the clusters by a two-stage procedure consisting of (1) the

construction and storage of the model, and (2) pixelwise compression of the clusters. The model is constructed from statistics gathered over the whole image and stored in the compressed file. The model is used for re-initializing the probability estimation pointers when the compression of a cluster starts. The compression requires two passes over the image but decompression can be performed with one pass only.

Model construction

In the first stage, the input image is analyzed and a probability model is calculated globally by counting the number of white and black pixels for each context separately. The calculated probabilities are mapped to the nearest *fast-attack* states in the state automaton using a look-up table. The indices of these states are stored in the compressed file. For each context, four bits are required for the state index and one bit for indicating the LPS colour. Only the fast-attack states are used. More accurate probability estimation would be obtained if we used all 113 states of the automaton. The fast-attack states, however, allow faster adaptation in the case when the initial model does not match the statistics of a particular cluster. Furthermore, the 14 states can represent all probabilities with sufficient accuracy and they can be stored with fewer bits. The probabilities of the fast-attack states and their corresponding threshold values (upper limits) are shown in Table 1.

The necessary extensions to the QM-coder are shown in Appendix. The probability mapping is implemented using the *GetFastAttackStateIndex* function. On the basis of the input probability, the function decides which color is the LPS and then finds the closest matching state to the LPS probability. The state index is found by a sequential search implemented in the *FindFastAttackState* function. The result is a five-bit index where four bits are required for the state index within the range [0, 13] and one bit for indicating the LPS. Otherwise the coding is performed using the standard routines of the QM-coder. Existing implementations can therefore be easily modified for the forward-adaptive variant.

Table 1. Fast attack states and their probabilities.

State:	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
p_{LPS}:	0.49690	0.20691	0.09417	0.04435	0.02120	0.01021	0.00493	0.00239	0.00116	0.00056	0.00028	0.00013	0.00006	0.00002
Limit:	0.30891	0.14590	0.06891	0.03255	0.01537	0.00726	0.00343	0.00162	0.00076	0.00036	0.00017	0.00008	0.00004	0.00000

Compression

In the second stage, the clusters are compressed by the backward-adaptive JBIG. The compression is essentially the same as the baseline JBIG. The only differences are that the QM-coder is reinitialized and the model is restored each time when the compression of a new cluster starts. The states are restored by the *RestoreState* function. It takes the context number (*context*) and the state index (*index*) as input and restores the fields (*mps* and *cstate*) for the appropriate context in the QM-coder accordingly. It is also noted that the pixels of the neighboring cluster can not be used in the context template. The pixels outside the cluster are thus assumed to be of the dominant image color (assumed to be white). After the cluster has been coded, the data buffer is filled by dummy bits and flushed to the code stream. Decompression is similar to the compression, except that the model construction phase is not needed. Instead, the model is read from the compressed file.

One-pass variant of the method

Dynamic modeling of the baseline JBIG has the advantages that only one pass over the image is needed. This is not the case for the forward-adaptive variant. The two-stage compression therefore needs the entire image to be stored in memory, which may restrict its application for on-line transmission (such as fax machines). The proposed idea can also be modified for one-pass compression by using precalculated model for the re-initialization. Instead of calculating the statistics from the compressed image, the initial model is optimized for a set of representative training images and is available for both encoder and decoder. This variant is of the same complexity as the baseline JBIG but less efficient than the two-pass variant. We shall further denote this one-pass variant as *S-JBIG* due to the static re-initialization.

4. Effect on the compression performance

The forward-adaptive method improves the compression performance because the adaptation does not start from scratch but a precalculated model is used for the initialization. This decreases the learning cost. The re-initialization increases also local adaptability by pushing the models from slow adaptive non-transient states back to the fast-attack states when the coding of a new cluster starts. On the other hand, there are also several sources of deficiencies, which originate from the following:

- Overhead required by the model table.
- Overhead required by the cluster indices.
- Compression deficiency due to cluster boundaries.

The model table is needed because of the forward-adaptive modeling. Cluster indices are recorded and stored to indicate the starting point of a cluster in the compressed bit stream. They are needed if the spatial access is implemented. Tiling the image into independent clusters has also the drawback that pixels of the neighboring cluster cannot be used in the context template. This results in less accurate prediction of pixels along the cluster borders. In the following we study these compression deficiencies as a function of the cluster size. The overhead is measured as the number of extra bits relative to the compressed file size.

The model table is stored in the compressed file using five bits per context: four bits for the state index and one bit for indicating the LPS color. The total overhead of the model for a *k*-pixel context model is thus:

$$Model\ overhead = 5 \cdot 2^k \cdot \frac{1}{S_C} = O\left(\frac{R}{X \cdot Y}\right) \qquad (1)$$

Here S_C is the compressed file size, $X \cdot Y$ the image size, and R the compression ratio of the baseline JBIG. The overhead is constant in respect to the cluster size. In case of the one-pass variant the model overhead is avoided but this results in a less accurate initial model and thus higher learning cost.

Cluster indices can be stored compactly as the offset (in bytes) from the previous cluster location; the actual cluster index table can be easily reconstructed in memory. Two bytes per cluster are enough to point clusters up to 724×724 pixels. In the worst case, when no compression is achieved (theoretically possible), the cluster is stored as such without compression. This situation is indicated by a special cluster offset code #FFFF. Additional overhead originates from the dummy bits that must be added to the last code byte to even the code stream at the end of each cluster. Four bits per cluster are needed on average to fill the byte. These overheads (*cluster overhead*) amount to 20 bits per cluster in total. The cluster overhead is proportional to the number of clusters (N_C) and inversely proportional to the compressed file size (S_C), and it is calculated as:

$$\textit{Cluster overhead} = 20 \cdot N_C \cdot \frac{1}{S_C} = O\left(\frac{X \cdot Y}{C^2} \cdot \frac{R}{X \cdot Y} \right) = O\left(\frac{R}{C^2} \right), \tag{2}$$

where $C \cdot C$ is the cluster size, $X \cdot Y$ the image size, and R the compression ratio of the baseline JBIG. Tiling the image into independent clusters has also the drawback that pixels outside the cluster cannot be used in the context template. The compression deficiency (*boundaries overhead*) caused by less efficient compression of pixels along cluster boundaries is proportional to constant width area along the perimeter of each cluster and is estimated as:

$$\textit{Boundaries overhead} = O\left(C \cdot N_C \cdot \frac{1}{S_C} \right) = O\left(C \cdot \frac{X \cdot Y}{C^2} \cdot \frac{R}{X \cdot Y} \right) = O\left(\frac{R}{C} \right) \tag{3}$$

The total overhead of the above three is thus:

$$\textit{Total overhead} = O\left(\frac{R}{X \cdot Y} \right) + O\left(\frac{R}{C^2} \right) + O\left(\frac{R}{C} \right) \tag{4}$$

To sum up, both the cluster overhead and the boundaries overhead are inversely proportional to the cluster size. The boundary overhead is the dominant of these two. The model overhead, however, depends on the image size only but it is relatively small for larger images.

Note that the cluster overhead can be fully avoided if spatial access is not required. Moreover, if the clusters are compressed sequentially in scan raster order, the boundaries overhead can also be reduced approximately to 1/3 because the pixels in the neighboring clusters in left and above can be utilized. Thus, the forward-adaptive variant could also be applied without spatial access at the cost of the model overhead, and a small deficiency due to cluster boundaries. In this case, the motivation would be improved compression performance, in which case the overall effect depends on whether the improvement overweighs the overhead.

5. Empirical results

The performance of the proposed method is tested by compressing the test images shown in Fig. 2. The image set consists of four GIS images and a newspaper image with a complex structure of text, line-drawings, and halftone pictures. We evaluate the four compression methods shown in Table 2. The baseline JBIG is a point of comparison. The rest are different variants supporting spatial access. The combination of tiling and JBIG (*T-JBIG*) is the trivial solution whereas the one-pass variant (*S-JBIG*) has a static initial model optimized for the eight CCITT test images. The forward-adaptive variant (*FA-JBIG*) is a two-pass method and it optimizes the initial model for each input image separately.

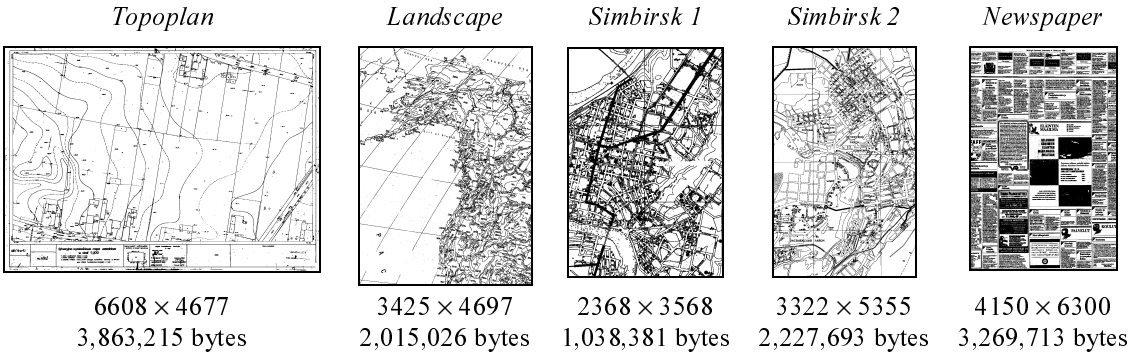

Fig. 2. Set of test images

Table 2. Evaluated compression methods and their properties.

Method	Tiling	Cluster indices	Initial model	Passes	Spatial access
JBIG	no	no	-	1	no
T-JBIG	yes	yes	-	1	yes
S-JBIG	yes	yes	static	1	yes
FA-JBIG	yes	yes	semi-adaptive	2	yes

We study first the amount of overhead as a function of the cluster size. The observed values for a set of test images are shown in Fig. 3. The cluster and model overheads are calculated using the equations (1) and (2). The boundary overhead is calculated as the difference in the bit rate when the images are compressed by the *FA-JBIG* with and without the restriction of the cluster boundaries. The total overhead for a realistic cluster size 128×128 is 8.4 % for the set of GIS images and 4.0 % for the newspaper image.

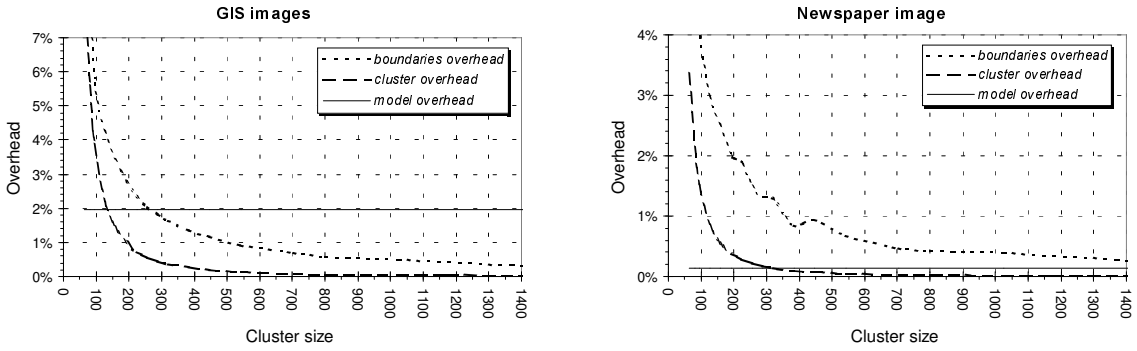

Fig. 3. Overhead of the *FA-JBIG* as a function of the cluster size.
The values are relative to the compressed file size of the baseline JBIG.

The improvement of using initial model and re-initialization are shown in Fig. 4. The benefit of a better initial model is significant (14.4 % for GIS and 9.3 % for Newspaper with the cluster size 128×128) and it increases greatly as the cluster size becomes smaller. The semi-adaptive initial model (*FA-JBIG*) is slightly better than the static initial model (*S-JBIG*). The re-initialization itself has also positive effect on the compression results (5.2 % for GIS and 4.0 % for Newspaper at maximum for *FA-JBIG*). A periodic re-initialization of the model would therefore be useful even if spatial access was not implemented.

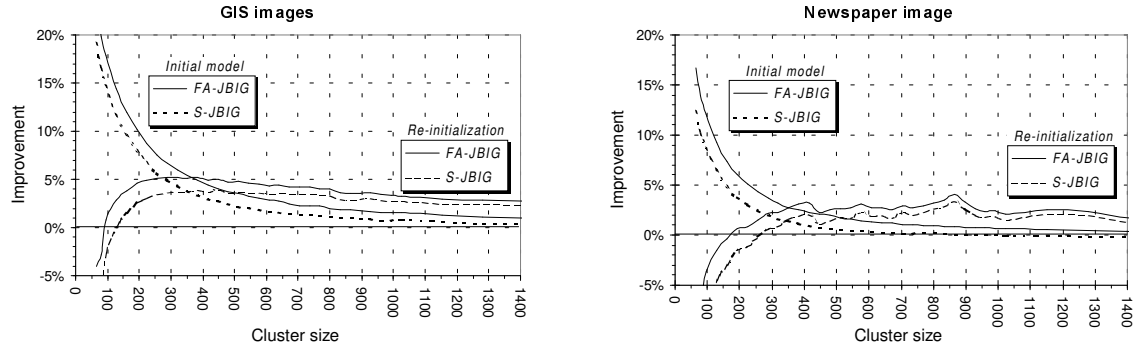

Fig. 4. Compression improvement as a function of cluster size.
The values are relative to the compressed file size of *T-JBIG*.

The overall effect of the clustering and the forward-adaptive modeling is summarized in Fig. 5. The benefits of a better initial model overweigh the overheads in all cases, except if very small cluster size is selected. In this case, the cluster and boundaries overheads become too large to be compensated. The re-initialization improves the local adaptability enough that even the *T-JBIG* gives somewhat better results than the baseline *JBIG*. The adaptation from the zero-state, however, takes longer time and therefore *S-JBIG* and *FA-JBIG* give better overall performance. The *FA-JBIG* surpasses the baseline JBIG for cluster sizes greater than 128×128, and it reaches maximal improvement (4.6 %) at 384×384 in the case of the GIS images. The compression ratios of the four methods are summarized in Table 3 for 128×128, 256×256, and for the optimal cluster size (for maximal compression performance).

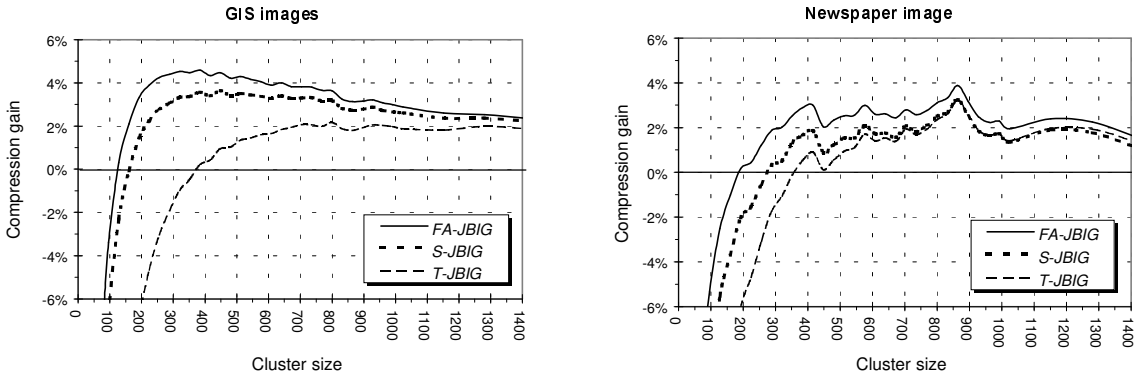

Fig. 5. Compression performance as a function of cluster size compared to the baseline JBIG.

Table 3. Summary of the compression performance.

	GIS Images			Newspaper image		
Cluster size:	128×128	256×256	Optimal	128×128	256×256	Optimal
JBIG	-	-	17.63	-	-	6.82
T-JBIG	15.26	17.11	18.02	6.05	6.62	7.05
S-JBIG	17.23	18.13	18.29	6.46	6.78	7.05
FA-JBIG	17.68	18.41	18.47	6.65	6.90	7.10

6. Conclusions

A forward-adaptive variant of the baseline JBIG was proposed. The method supports spatial access and at the same time it improves the compression performance of the baseline JBIG. Compression improvement is obtained because of a better initial model and because of periodic re-initialization of the model. These benefits overweigh the overheads of tiling and the model when cluster size is about 128×128 or larger. The method improves the compression performance of the baseline JBIG by about 4 % when cluster size is chosen optimally (384×384 for a set of test images).

A minor drawback is that the compression requires two passes over the data; decompression can still be performed with a single pass only. A one-pass variant can be obtained by using a precalculated (static) model for the initialization. This has the same benefits than the forward-adaptive without buffering the entire image at the cost of slightly higher learning cost. Both variants can be implemented with only small modifications to the baseline JBIG; current software implementation can therefore be easily utilized.

Acknowledgements

The work of Pasi Fränti was supported by a grant of the Academy of Finland, and the work of Eugene I. Ageenko by a grant of Center for International Mobility (CIMO).

References

1. H. Samet, *Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS*, Addison-Wesley , Reading, MA, 1989.
2. R. Pajarola and P. Widmayer, "Spatial indexing into compressed raster images: how to answer range queries without decompression". *Proc. Int. Workshop on Multimedia DBMS*, Blue Mountain Lake, NY, 94-100, 1996.
3. E.I. Ageenko and P. Fränti, "Enhanced JBIG-based compression for satisfying objectives of engineering document management system", *Optical Engineering*, **37** (5), May 1998. (to appear)
4. JBIG: 'Progressive Bi-level Image Compression', ISO/IEC International Standard 11544, ITU Recommendation T.82, 1993.
5. W.B. Pennebaker, J.L. Mitchell, G.G. Langdon, R.B. Arps, "An overview of the basic principles of the Q-coder adaptive binary arithmetic coder". *IBM Journal of Research and Development*, **32** (6), 717-726, 1988.
6. W.B. Pennebaker, J.L. Mitchell, *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, 1993.

Appendix: Extensions for QM-coder

```
float FastAttackStateBounds [13] = {
      .30891,   .14590,   .06891,   .03255,   .01537,
      .00726,   .00343,   .00162,   .00076,   .00036,
      .00017,   .00008,   .00004    };

int FindFastAttackState (float Prob)
{
  int i;

  for (int i=0; i<13; i++)
  {
     if ( Prob > FastAttackStateBounds [i] )  return (i);
  }
  return (13);
}
```

```
int GetFastAttackStateIndex (float WhiteProb)
{
   float LpsProb;
   int index;

   if (WhiteProb < 0.5) LpsProb = WhiteProb;
   else LpsProb = 1 - WhiteProb;
   index = WhiteProb < 0.5 ? 0x00 : 0x10;
   index = index | FindFastAttackState (LpsProb);
   return (index);
}

void RestoreState (int context, int index)
{
   mps [context] = (index & 0x10) ? 0 : 1;
   cstate [context] = (index & 0x0f);
}
```