A SIMPLE AND EFFICIENT BINARY SHAPE CODING TECHNIQUE BASED ON BITMAP REPRESENTATION

Frank Bossen *Touradj Ebrahimi*

Signal Processing Laboratory
Electrical Engineering Department
EPFL
1015 Lausanne Switzerland

ABSTRACT

This document presents a technique based on the JBIG algorithm for binary shape coding in both lossless and lossy modes. Because it is applied directly to the bitmap representing the shape information, it bypasses the overhead in computation of an intermediate contour representation and its associated conversions. This leads to a simpler algorithm which is more suitable for a larger class of shape data. In addition a mechanism is proposed which allows a rate control for lossy coding mode.

1. INTRODUCTION

Second generation video coding algorithms are also referred to as object oriented, that is, the scene to be coded is segmented into several regions, each of them coded separately. These regions generally identify objects. Each object is represented by four channels: three color channels and an alpha channel which defines the shape of the object. This alpha channel can be either binary or multilevel. The multilevel case allows for semi-transparent object. However the scope of this paper is limited to the coding of binary alpha channels.

This document presents a universal technique for binary alpha channel coding based on the JBIG algorithm. It can operate either in a scalable or a non-scalable mode. The scalability addressed here is spatial. However it can also be viewed as a quality scalability and lead to a simple rate versus distortion control scheme.

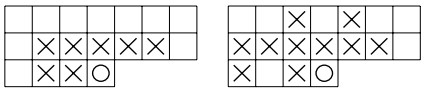
Because it is applied directly to the bitmap representing the shape information, it bypasses the overhead in computation of an intermediate contour representation and its associated conversions. This leads to a simpler algorithm which is more suitable for a larger class of shape data [1].

This document is organized as follows. Section 2 presents some previous work. A non-scalable algorithm is described in section 3 and a scalable one in section 4. Results are shown in section 5 and conclusions drawn in section 6.

2. PREVIOUS WORK

Quadtree, chain coding, and polygonal approximation techniques are quite popular methods for shape coding. The latter two require a contour representation of the object shape. This representation can be defined in several ways, but it is generally difficult to come up with a representation which handles small details well. Therefore these methods are not good candidates for universal shape coders.

Another approach draws its source from text compression techniques. Langdon and Rissanen [2] proposed an efficient method based on finite state machines and arithmetic coding. The idea is quite simple: the image is coded pixel by pixel in a scanline order. For each pixel, the state of the finite state machine is defined by the values of pixels within a template. This template typically includes pixels in the close vicinity of the pixel to be coded. With each state is associated a probability distribution, which is used to drive the arithmetic coder. Figure 1 shows the two templates that were used.



7-pel template 10-pel template

Figure 1. Templates proposed by Langdon and Rissanen. The circle represents the pixel to be coded, and the crosses the pixels belonging to the template.

This coding paradigm has also been adopted for the JBIG [3] standard. The JBIG standard uses one of two 10-pel templates for non-progressive coding. The first template holds on two lines and the second on three, as shown in figure 2. JBIG further allows a progressive transmission of bi-level images. This is achieved by sucessively transmitting layers of a multiresolution decomposition of the image.

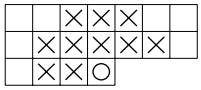


Figure 2. The 3-line 10-pel template of the JBIG algorithm

3. NON-SCALABLE CODING

Let I be the binary alpha channel to be coded, and w and h the width and height of I, respectively. For each pixel (i, j), $I(i, j)$ is defined to be equal to 1 if (i, j) belongs to the object, and 0 if it doesn't, where i and j represent the line and column numbers, respectively. The top left corner pixel of I is defined to be at $(0, 0)$.

To achieve a non-scalable coding algorithm, the same principle of a finite state machine as in [2] is applied. The

chosen template size is 10 bits, because it offers a good tradeoff between perfomance and memory requirements. More formally, for each pixel (i,j) the context (state) is defined by $C(i,j) = \sum_{k=0}^{9} c_k(i,j)2^k$. For the 3-line 10-pel JBIG template the c_k's can be defined as:

$$\begin{array}{llll} c_0(i,j) &= I(i,j-1) & c_1(i,j) &= I(i,j-2) \\ c_2(i,j) &= I(i-1,j+2) & c_3(i,j) &= I(i-1,j+1) \\ c_4(i,j) &= I(i-1,j) & c_5(i,j) &= I(i-1,j-1) \\ c_6(i,j) &= I(i-1,j-2) & c_7(i,j) &= I(i-2,j+1) \\ c_8(i,j) &= I(i-2,j) & c_9(i,j) &= I(i-2,j-1) \end{array} \quad (1)$$

and for the 10-bit template by Langdon and Rissanen:

$$\begin{array}{llll} c_0(i,j) &= I(i,j-1) & c_1(i,j) &= I(i,j-3) \\ c_2(i,j) &= I(i-1,j+2) & c_3(i,j) &= I(i-1,j+1) \\ c_4(i,j) &= I(i-1,j) & c_5(i,j) &= I(i-1,j-1) \\ c_6(i,j) &= I(i-1,j-2) & c_7(i,j) &= I(i-1,j-3) \\ c_8(i,j) &= I(i-2,j+1) & c_9(i,j) &= I(i-2,j-1) \end{array} \quad (2)$$

For all pixels (i,j) outside the bounds of I, $I(i,j)$ is defined to be zero. The particular order of the c_k's doesn't influence the performance of the algorithm, but is here defined for completeness.

3.1. Non-adaptive arithmetic coding

Although the use of an adaptive arithmetic coder is common, the probability distribution $p(I(i,j) \mid C(i,j))$ is here defined to be constant. There are several advantages to this. First, less memory is required since less bookkeeping information needs to be stored. Then, the probability distribution can be represented with fixed point numbers, which allows to use an arithmetic coder with no division operation. Representing all probabilities smaller than 0.5 with a power of two can even yield a multiplication free arithmetic coder. However, the latter solution degrades the compression performance and is thus not retained.

The constant probability distribution $p(I(i,j) \mid C(i,j))$ is defined by the analysis of several typical binary alpha channels. For each context C_k, let $n_{k,0}$ and $n_{k,1}$ be the sum over all the training set of the number of occurences of zeros and ones, respectively. The probability distribution is then derived according to:

$$p(I(i,j)=v \mid C(i,j)=C_k) = \frac{n_{k,v} + b}{n_{k,0} + n_{k,1} + 2b} \quad (3)$$

where the bias b is usually set equal to 1 to avoid null probabilities.

4. SCALABLE CODING

The above presented technique is now further extended to scalable (progressive) coding, as in the JBIG standard [3]. The JBIG multiresolution decomposition algorithm is quite complex since it has to deal with dithered images. For coding object shapes however, a much simpler decompostion algorithm can be used as described in the next paragraph.

4.1. Multiresolution Decomposition

Let $\left\{I^0, \ldots, I^m\right\}$ be the multiresolution decomposition of the binary alpha channel I into $m+1$ layers. The size of each layer I^l is $\lceil 2^{-l}w \rceil \times \lceil 2^{-l}h \rceil$. The largest layer I^0 is defined equal to I and the remaining layers are defined as follows:

$$\begin{aligned} I^{l+1}(i,j) \;=\; & I^l(2i,2j) \;\vee\; I^l(2i+1,2j) \\ & \vee\; I^l(2i,2j+1) \;\vee\; I^l(2i+1,2j+1) \end{aligned} \quad (4)$$

where $\vee$ denotes the boolean *or* operator.

The layer I^m is referred to as the base layer and $I^1, \ldots, I^{m-1}$ to as the enhancement layers.

4.2. Base layer coding

The base layer I^m can be coded using the non-scalable algorithm described in the previous section. However the base layer is generally quite small, and it doesn't hurt very much on the compression performance side to encode it as raw data. This latter solution has thus been adopted.

4.3. Enhancement layers coding

The enhacement layers are coded seperately and in descending order, that is from I^{m-1} down to I^0. The coding of each enhancement layer I^l is very similar to the non-scalable case described in the previous section. The difference lies in the definition of the 10-pel template. In the scalable case, the template includes pixels from previously coded layers. The context $C(i,j) = \sum_{k=0}^{9} c_k(i,j)2^k$ is defined as follows:

$$\begin{array}{llll} c_0(i,j) &= I^l(i-1,j+1) & c_1(i,j) &= I^l(i-1,j-1) \\ c_2(i,j) &= I^l(i-1,j) & c_3(i,j) &= I^l(i,j-1) \\ c_4(i,j) &= I^{l+1}(\tilde{\imath},\tilde{\jmath}-1) & c_5(i,j) &= I^{l+1}(\tilde{\imath},\tilde{\jmath}) \\ c_6(i,j) &= I^{l+1}(\tilde{\imath}-1,\tilde{\jmath}-1) & c_7(i,j) &= I^{l+1}(\tilde{\imath}-1,\tilde{\jmath}) \\ c_8(i,j) &= 2\tilde{\imath}-i & c_9(i,j) &= 2\tilde{\jmath}-j \end{array} \quad (5)$$

where $\tilde{\imath} = \lceil i/2 \rceil$ and $\tilde{\jmath} = \lceil j/2 \rceil$. As for the non-scalable case, for all pixels (i,j) outside the boundaries of a layer l, $I^l(i,j)$ is defined to be zero.

This template is different from the one used in JBIG. Indeed the present template is slightly smaller, and the alignement between the layers is different. The bits c_8 and c_9 correspond to the phases which are defined in the JBIG algorithm.

4.4. Rate control

Rate versus distortion control is achieved by coding only a subset of all events. Lossiness is parametrized by a layer number z and a threshold τ ranging from 0 to $\frac{1}{2}$. Given z and τ, all layers k such that $k > z$ are coded as described above, and all layers k such that $k < z$ are not coded at all. At layer z only the events with a probability between τ and $1-\tau$ are coded. The other events are considered to have the most probable outcome. At the decoding stage, if $z > 0$, I^0 is obtained by upsampling I^z.

The lossless mode is thus defined by $z = 0$ and $\tau = 0$, and the most lossy mode, in which only the base layer is coded, by $z = m$ and $\tau = 0$.

5. RESULTS

The above described techniques have been implemented using the arithmetic coder described in [4]. The probability values used to drive the arithmetic coder are quantized to 16 bit fixed point numbers.

The test data on which results are presented is the *lady* objet of the *weather* sequence (see figure 3). The frame size

is QCIF (176 by 144 pixels) and the number of frames is 100 (sampled at 10 Hz). The training data is the *speakers* object of the *news* sequence and the *kids* object of the *children* sequence. Both are QCIF-sized and 100 frames long. A frame of each sequence is shown is figure 4.

Figure 3. Seventy-first frame of the *lady* object in the *weather* sequence

speakers in *news* sequence

kids in *children* sequence

Figure 4. Seventy-first frame of the training sequences

Technique	Bit count	Compression ratio
non-progressive	866.72	29.24
progressive	1625.84	15.59

Table 1. JBIG compression results: average bit count per frame

The reference for compression performance evaluation of the proposed techniques is the JBIG algorithm. Its performance in both progressive and non-progressive modes is reported in table 1, where the bit counts are averages over the 100 frames. The slice height was set to the height of the frame, and the number of layers in the progressive mode was set to 4. It appears that the non-progressive mode performs much better than the progressive one. The probable reason for this is probably the slower learning rate of the progressive mode, which uses a 12-bit context, whereas the non-progressive uses a 10-bit one.

Template	Bit count	Compression ratio
JBIG	463.73	54.65
JBIG (inbreeding)	447.82	56.59
LR	460.95	54.98
LR (inbreeding)	445.23	56.92

Table 2. Performance comparison between the 3-line JBIG and the 10-pel Langdon & Rissanen (LR) templates for non-scalable coding

To evaluate which template yields the best compression results for the proposed non-scalable algorithm, tests have been run with the 3-line JBIG template and the 10-pel LR (for Langdon and Rissanen) one. Results are reported in table 2. For each template two bit counts are given: the first one based on the probability distribution $p(I(i,j) \mid C(i,j))$ drawn from the training sequences, and the second drawn from the test sequence itself (inbreeding). The differences between inbreeding or not are small, and the assumption that there is a probability distribution which works well for a large class of sequences is thus verified.

The table also shows that the original LR template performs marginaly better than the JBIG one. When comparing with the JBIG results of table 1, it appears that the non-adaptive solution performs much better. Indeed the compression ratio is almost twice as high. The main reason is the small size of the masks to be coded, which doesn't leave time for the algorithm to adapt. Also the initial assumptions of the JBIG algorithm are not adapted to the test data.

Mode	Bit count	Compression ratio
scalable	549.03	46.16
scalable (inbreeding)	525.61	48.22
lossy ($z = 0, \tau = 0.25$)	398.81	63.55

Table 3. Performance of scalable and lossy modes

Further results show that there is a price to pay for additional functionality such as spatial scalability. Indeed the compressed stream is about 20 percent longer when using the scalable scheme (see figure 3). However the probability distribution $p(I(i,j) \mid C(i,j))$ seems to be quite constant over different sequences. The average bit count is reduced by less than 5 percent with an inbreeded distribution. In the scalable case, the performance difference with the JBIG algorithm is even larger. The performance ratio is around 3. This is probably due to the multiresolution decomposition algorithm of JBIG which is tuned for halftoned images and text, and thus not suitable for binary masks.

Lossy coding dramatically reduces the bit count without much degrading the image quality, as shown in figure 5.

6. CONCLUSION

A method based on the JBIG algorithm for coding binary alpha channels has been presented. The JBIG algorithm

Figure 5. Reconstructed seventy-first frame of the *lady* object in the *weather* sequence after lossy shape coding ($z = 0, \tau = 0.25$)

has been adapted to take into account caracteristics of alpha channels, such as the small data size (a movie frame has typically many less pixels than a fax page) and the homogeneity of the data (all parts of any alpha channel look pretty much the same, whereas a fax page can contain data as diverse as roman text, kanji text, and dithered images). The adaptive arithmetic coder has been replaced by a non-adaptive one, since the adaptation time is too short and the nature of the data well known. Also the multiresolution decomposition procedure has been changed in consideration with the nature of shape data. However the proposed solution is not unique, and could easily be changed without noticeably affecting the performance of the algorithm.

Simulation results have shown that the proposed method works well (50 to 1 compression ratios) and would be a good candidate for a univeral binary alpha channel coder. Still improvements could be brought, including a motion estimation and compensation scheme. The template could then be changed to include pixels from the previous, motion compensated, binary mask.

REFERENCES

[1] F.Bossen and T. Ebrahimi. A simple and efficient binary shape coding technique based on bitmap representation. Technical Report MPEG M0964/Tampere, ISO/IEC JTC1/SC29/WG11, 1996.

[2] G.G. Langdon Jr. and J. Rissanen. Compression of black-white images with arithmetic coding. *IEEE Transactions on Communications*, COM-29(6):858–867, June 1981.

[3] ITU-T. JBIG: progressive bi-level image compression. Technical Report T.82, International Telecommunication Union, 3 1993.

[4] A. Moffat, R. Neal, and I. H. Witten. Arithmetic coding revisited. In *Data Compression Conference*, pages 202–211, 1995.

A SIMPLE AND EFFICIENT BINARY SHAPE CODING TECHNIQUE BASED ON BITMAP REPRESENTATION

Frank Bossen and Touradj Ebrahimi

Signal Processing Laboratory
Electrical Engineering Department
EPFL
1015 Lausanne Switzerland

This document presents a technique based on the JBIG algorithm for binary shape coding in both lossless and lossy modes. Because it is applied directly to the bitmap representing the shape information, it bypasses the overhead in computation of an intermediate contour representation and its associated conversions. This leads to a simpler algorithm which is more suitable for a larger class of shape data. In addition a mechanism is proposed which allows a rate control for lossy coding mode.