Optimal Pruning Quad-Tree Block-Based Binary Shape Coding

Zhenliang Shen, Michael Frater, John Arnold

School of ITEE, University College,
The University of New South Wales, Australia

ABSTRACT

In object-based video coding, shape masks are used to specify the shape of a video object. The binary shape means each pixel is either background or object. In this paper, we present a new lossless block-based coding algorithm for binary shapes that combines a quad-tree structure with block-based arithmetic coding; an optimal pruning quad-tree structure searching algorithm is proposed to further reduce the spatial redundancy of quad-tree blocks. Experimental results demonstrate that this new approach provides a saving in bits generated between approximately 40-75% compared to the MPEG-4 binary shape coding algorithm.

***Index Terms*—**Quad-Tree, CAE, adaptive arithmetic coding

1. INTRODUCTION

In object-based video coding, foreground video objects are separated from the background and coded independently, such as MPEG-4 [1]. The video objects have three types of information: are texture, shape, and motion. The texture has the YUV components. The shape information associated with a video object takes the form of a mask specifying the transparency of each pixel. The shape mask could be either greyscale or binary. We are concerned only with binary shape mask in this paper. The term of "binary shape" is used to describe shape masks where each pixel is either completely inside the object or completely outside it, i.e. there is no blending of pixels at object boundaries. There are two major classes of technology for the compression of binary shape masks: contour-based coding and bitmap-based coding. In contour-based methods, the boundary information of the closed contour enclosing the object is used to represent the shape information. These methods include chain coding [2] and vertex-based coding [3]. Bitmap-based methods encode the source binary image directly. These include modified-read (MR) coding [4], context-based arithmetic encoding (CAE) [5], and Quad-Tree coding [6].

Some recent works on binary shape coding are as follows. The skeleton-based method decouples the shape information into two independent signal data sets: the skeleton and the boundary distance from the skeleton [7]. A new contour-based algorithm provides improved coding efficiency to reduce the redundancy of Differential Chain Coding (DCC) [8]. In [6], quad-tree based coding with block compensation is used, with the bitstream size reduced by allowing some small distortion of the shape mask. A new technique based on a local analysis of the digital straightness of the causal part of the object boundary (DSLSC) has been suggested [9]. MEPG-4 uses CAE as its binary shape coding because of its coding efficiency and the feasibility of hardware implementation [5]. In this paper, we propose a lossless block-based coding scheme for binary shape masks, based on optimal pruning of a quad-tree coding in conjunction with context-based arithmetic coding. The rest of the paper is organized as follows. The quad-tree block-based coding method is presented in Section 2. The optimal pruning quad-tree structure searching is introduced in Section 3. Finally, experimental results and conclusion are presented in Sections 4 and 5.

2. Quad-Tree Block-Based Binary Shape Coding

In MPEG-4 binary shape coding, a binary shape mask is broken up into blocks of 16x16 pixel, known as *binary alpha blocks* (BAB). These BABs are encoded in the same raster scan order as motion and texture data. Each BAB is identified as either transparent, opaque or boundary. For boundary blocks, each pixel of blocks is coded using a context-based arithmetic coder [10]. The context-based arithmetic coder encodes the probability table based on a context number, where the context number is generated from a template of the surrounding pixels. In the new technique proposed in this paper, we use a quad-tree decomposition in which each boundary block is split into four equal-size sub-blocks. Each of these 8x8 pixel sub-blocks, is then identified as being transparent, opaque, or boundary. This process is repeated as many times as is required. Figure 1 illustrates the quad-tree structure in a 16x16 block. In the quad-tree structure, a three-symbol system ('0', '1', and 'B') is instead of the original binary pixels. The quad-tree structure of the 16x16 block in Figure 1 can be expressed in the following sequence:

B 000B B 000B 0001 B 00B 0111 1 B B 0111 1B 0111 1 1

The order of the quad-tree sequence is from the top to the bottom layer. The larger the font size, the larger the block is, i.e. the higher the layer containing the larger block. For each 16x16 boundary block, MPEG-4 codes each pixel; hence there are 256 coding units, each of which takes one of two values ('0', '1'). In the quad-tree structure, the 256-pixel block is reconstructed by using 37 quad-tree units, each of which takes of one of three values ('0', '1', 'B'). In our quad-tree coding scheme, in fact, the coding order is not same as the above quad-tree sequence. Each frame has to be coded five times, which is coded from larger size block to small size block separately. (See Figure 2) For each block of different layer, it is coded by using context-based arithmetic encoder [10].

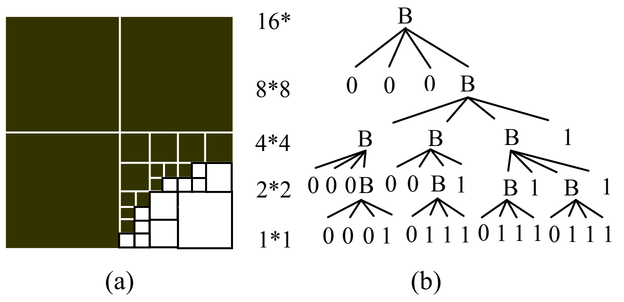

Figure-1 (a) Transparent, opaque, and boundary blocks in 16-by-by block. (b) Quad-Tree values in the hierarchical structure

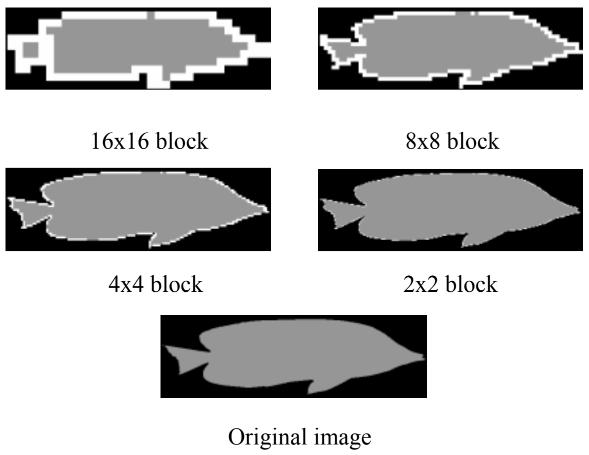

Figure-2 Different layers in the quad-tree structure (black indicates background, grey indicates the object, and the white is the boundary blocks.

The efficiency of CAE is highly dependent on the choice of the template. In general, the more pixels in the template, the more accurate the probability distribution of the coded pixel is, leading to improved coding performance. Moreover, in templates of the same size, the higher the correlation between the pixels in the template and the pixel being coded, the more accurate the context probability table is and the less spatial redundancy the coded pixels have. A limitation on the choice of the pixels to form the template is that they must be decoded before the current pixel. In MPEG-4, only the pixels in top and left of the current coding pixel can be chosen in its template because of the raster scan coding order. The coding efficiency of CAE should be better if the information of the bottom and right of the current coding pixel can be used in the template. In quad-tree block-based coding, since the difference size blocks are coded separately, the block information of the larger size is known for the coding block in current layer. Therefore, blocks from the upper layer can be used in the template, even where they lie to the right or below the current block, while blocks from the current layer can only be used if they precede it in the blocks scan order. In our scheme, considering the distribution of the blocks, four different templates are designed for each of the four sub-blocks of the boundary block. Figure 3 shows these intra templates in any layers except the 16x16 layer (we use the same template as MPEG-4 in this layer). B1, B2, B3, and B4 are used to denote the four blocks in the current layer, which are the sub-blocks of a boundary block of upper layer. C1, C2, C3, and C4 are blocks whose value is known to the decoder in upper layer but not in the current layer. C5-C9 are blocks that have already been coded in the current layer, and whose values are therefore known to the decoder. For inter coding, four similar templates are created by adding information from the previous frame. The information of upper layer is of lower resolution than the current block, but it still includes important information that has a significant impact on coding efficiency. Because the blocks in Quad-Tree structure have three values, the size *(c)* of the context number can be expressed as $C = \sum_k c_k * 3^k$, where *k* is the number of blocks in the template.

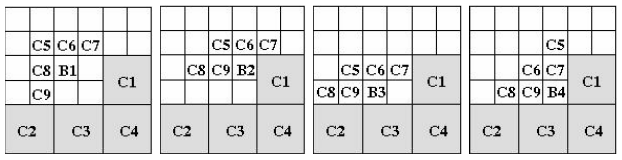

Figure-3 Intra templates of Quad-Tree block-based coding with different sub-blocks (B1, B2, B3, and B4 are sub-blocks to be coded, C1-C4 are blocks from upper layer, C5-C9 are blocks having been coded in current layer.)

3. Optimal Pruning Quad-Tree Block-Based Adaptive Arithmetic Coding

In Figure 1, the expression of quad-tree structure can significant reduce the spatial redundancy in a 16x16-pixel block because there are many homogenous sub-blocks. However, the quad-tree structure only works well in the simple boundary block. For some complicated boundary blocks, the quad-tree structure does not have the same efficiency as in the Figure 1, and can be even worse than coding each pixel directly. For example, Figure-4 (a) is a 16x16 block with complicated boundary information. This quad-tree structure has 1 block in 16x16 layer, 4 blocks in 8x8 layer, 16 blocks in 4x4 layer, 64 blocks in 2x2 layer, and 140 pixels in 1x1 layer, which means it needs 268 bits to store this quad-tree structure (Each quad-tree block has 3 symbols which value is *1.5 bits/block* and pixel is binary). Thus, the total numbers of bits are larger than required for a simple bit map with one bit per pixel. For some blocks, the quad-tree structure expression could be efficient in the 16x16 layer; but not for all the sub-blocks. This could be demonstrated in the Figure 3 (b). In this block, there is 1 block in 16x16 layer, 4 blocks in 8x8 layer, 8 blocks in 4x4 layers, 20 blocks in 2x2 layer, and 60 pixels in 1x1 layer, which needs 110 bits and is more efficient comparing to code each pixel directly. However, these four 8x8 sub-blocks have totally different quad-tree expressions. The top two 8x8 sub-blocks are not boundary blocks so they do not have to be split further. Analyzing the quad-tree structure of bottom left 8x8 sub-block, there are 4 blocks in 4x4 layer, 4 blocks in 2x2 layer, and 4 pixels in 1x1 layer, which need 16 bits. It is very efficient comparing to use the pixel expression directly. For the bottom right 8x8 sub-block, however, there are 4 blocks in 4x4 layer, 16 blocks in 2x2 layer, and 56 pixels in 1x1 layer, which needs 86 bits. In comparison of 64 binary pixels in this 8x8 sub-block, the quad-tree structure is inefficient. If the quad-tree structure is applied in the 16x16 layer and two top and bottom left sub-blocks, and the bottom right sub-block is coded pixel directly, the new mixed structure have 16 blocks and 68 pixels. In here, each block has four states, which they are all '0', all '1', blocks with quad-tree ('B*q*'), and blocks without quad-tree ('B*p*') (The value is 2 bits/block in here). Hence, it needs 100 bits for the new mixed structure and saves 10 bits in comparison of previous quad-tree. Depending on the complexity of the boundary information, the sub-blocks within different layers have an optimal expression by using either quad-tree structure or pixels.

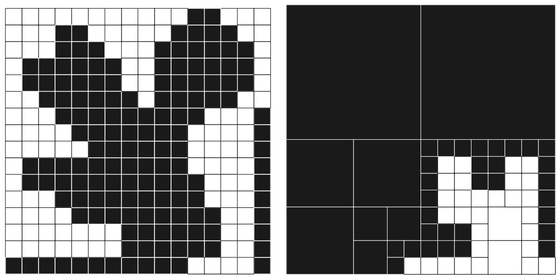
(a) (b)

Figure-4 16x16 blocks with complicated boundary

We are inspired by the work of leaf merging by [11, 12], which analyses the sub-optimal rate-distortion properties of quad-tree in the case of geometric image modeling. In this paper, since the binary shape coding is lossless, we use the minimum block and pixel entropy searching in each quad-tree layer based on context-based adaptive arithmetic coding to rectify the initial quad-tree structure. The entropy of each block and pixel is counted based on their context-based probability:

$$Entropy = -p_n(context) * \log_2 p_n(context) \qquad (1)$$

where $p_n(context)$ is the estimated probability distribution of block or pixel in value '*n*' with the particular context number, it can be taken as

$$p_n(context) = \frac{\sum N(context) * p_n(context) + k}{\sum N(context) + c} \qquad (2)$$

N(context) denotes the count of coded pixel in that context number. '*k*' and '*c*' are constant ('*k*' and '*c*' are set 1 in this paper) [10].

The optimal pruning quad-tree structure searching is to count the entropy of block and pixel from 4x4 layer to the larger layer. In 4x4 layer, the total entropy of quad-tree structure in particular layer is compared to the total entropy of 16 pixels in the same layer. If the block entropy is smaller than the pixel entropy, the quad-tree structure is used in this layer; otherwise, the pixel is coded directly in this layer. The minimum entropy of each 4x4 block is stored to measure in the 8x8 layer. This searching algorithm will be stopped at the 16x16 layer. The new quad-tree structure in the Figure-4 (b) can be express as the follow. (After the B*p* block, all pixels in this block have to be coded.)

$$B_q\ 00B_q\ 000B_q\ 000B_q\ 0001\ B_p\,(0,0,0,\ldots\ldots 1,1)$$

4. Experimental Results

Results comparing the performance of the new binary shape coding method against MPEG-4 are presented in this section. Tests were carried out using shape masks associated with five standard video test sequences. For each shape mask, the alpha values between 0 and 255 were converted to binary by thresholding. Background noise, located away from the object of interest, was also removed. The even field of each sequence was used. All coding results are expressed in bits per filed. Table-1 shows the result of intra coding. The MPEG-4 results were obtained using the standard MPEG-4 Intra probability tables for the CAE. The number of bits includes those generated by the CAE coding and BAB type coding. Table-2 shows the results of inter coding. The MPEG-4 results include the CAE coding, BAB type coding, and the motion vector differences. In the quad-tree block-based coding, both the intra and inter probability tables were generated using all five test video sequences. The dynamic probability tables were updated after each block coded in the optimal pruning quad-tree block coding. The results for DSLSC are taken from [9], and rescaled from bytes per frame to bits per field. From the Tables 1 and 2, the quad-tree block-based coding is about 25-50% more efficient than MPEG-4 in Intra coding and about 35-65% in Inter coding. The optimal pruning quad-tree block-based coding is about 35-55% more efficient than MPEG-4 in Intra coding and about 40-70% in Inter coding. Thus, the proposed method is superior to the MPEG-4 binary shape coding in terms of the compression ratio

Table 1 Intra coding results of MPEG-4 CAE, DSLSC, Quad-Tree Block-Based coding, and Optimal Pruning QT Block-based Coding (bits/field)

Sequence	MPEG4	DSLSC	Quad Tree	Optimal pruning QT
Akiyo	1831	844	1083	906
Bream	2338	1524	1694	1348
Children	3195	2696	2413	2097
News	2341	2104	1393	1059
Weather	1624	1044	1001	801

Table 2 Inter coding results of MPEG-4 CAE, DSLSC, Quad-Tree Block-Based, and Optimal Pruning QT Block-based Coding (bits/field)

Sequence	MPEG4	DSLSC	Quad Tree	Optimal pruning QT
Akiyo	1059	480	417	379
Bream	2154	1268	1187	1089
Children	2954	2112	1897	1714
News	958	680	344	301
Weather	1235	636	423	371

5. CONCLUSION

In this paper, a new block-based lossless coding method for binary shape masks has been presented. This technique is based on quad-tree coding, in which context-based arithmetic coding is used to minimize the number of bits required. Adaptive arithmetic coding is used to search the optimal pruning quad-tree structure. For a number of standard test video sequences, it has been demonstrated that this new technique requires between 30 and 70% less bits than are required for MPEG-4 CAE. More efficient searching algorithm of optimal pruning quad-tree structure is left as our future research.

6. REFERENCES

[1] MPEG Video Group, "MPEG-4 Video VM 18.0," January 2001.

[2] M. Ghanbari, *Video coding: an introduction to standard codecs* London: Institution of Electrical Engineers, c1999.

[3] A. K. Katsaggelos, L. P. Kondi, F. W. Meier, J. Ostermann, and G. M. Schuster, "MPEG-4 and rate-distortion-based shape-coding techniques," *Proceedings of the IEEE*, vol. 86, pp. 1126 - 1154, June 1998.

[4] N. Yamaguchi, T. Ida, and T. Watanabe, "A binary shape coding method using modified MMR," *International Conference on Image Processing, 1997. Proceedings.*, vol. 1, pp. 504 - 507, Oct. 1997.

[5] N. Brady, "MPEG-4 standardized methods for the compression of arbitrarily shaped video objects," *IEEE Transactions Circuits and Systems for Video Technology*, vol. 9, pp. 1170 - 1189, 1999.

[6] L. Y. Wang, C. H. Lai, and K. R. Pan, "Quad-tree-based image shape coding with block compensation," *Third International Conference on Information Technology and Applications. ICITA 2005.*, vol. 1, pp. 716 - 719, July 2005.

[7] H. Wang, G. Schuster, A. Katsaggelos, and T. appas, "An efficient rate-distortion optimal shape coding approach utilizing a skeleton-based decomposition," *IEEE Transactions on Image Processing* vol. 12, pp. 1181-1193, Oct. 2003.

[8] L. Y. Wang and C. H. Lai, "An efficient contour-based algorithm for binary video shape coding," *IEEE International Midwest Symposium on Circuits and Systems. MWSCAS 2004.*, vol. 1, pp. I - 261-4, July 2004.

[9] S. Aghito and S. Forchhammer, "Context-Based Coding of Bilevel Images Enhanced by Digital Straight Line Analysis," *IEEE Transactions on Image Processing*, vol. 15, pp. 2120 - 2130, Aug. 2006.

[10] I. Witten, R. Neal, and J. Cleary, "Arithmetic coding for data compression," in *Communications of the ACM*, vol. 30, June 1987, pp. 520-540.

[11] R. Shukla, P. Dragotti, M. Do, and M. Vetterli, "Rate-distrotion optimized tree structured compression algorithms for piecewise polynomial images," *IEEE Trans. Image Proc.*, vol. 14, March 2005.

[12] R. D. Forni and D. S. Taubman, "On the benefits of leaf merging in quad-tree motion models," *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, vol. Volume 2, 11-14 Sept. 2005 Page(s):II - 858-61, Sep 2005.