ICIP 98

EFFICIENT ENCODING OF BINARY SHAPES USING MPEG-4

Jörn Ostermann

AT&T Labs Research
100 Schultz Dr.
Red Bank, NJ 07701, USA
Email: osterman@research.att.com

ABSTRACT

MPEG-4 Visual, that part of the upcoming MPEG-4 standard describing the coding of natural and synthetic video signals, allows the encoding of video objects using motion, texture and shape information. In this paper, the MPEG-4 context-based arithmetic encoder for encoding binary shape information is presented in the context of a new MPEG-4 video encoder architecture. The encoder architecture enables us to efficiently encode lossless and lossy shape, motion and texture of a moving video object. Several non-normative choices for efficient computation and bit efficient encoding of arbitrarily shaped video objects are investigated in order to enable real-time encoding.

1. INTRODUCTION

MPEG-4 Visual will be the first international standard allowing the transmission of arbitrarily shaped video objects (VO). A time instant of a VO is called Video Object Plane (VOP). A VOP is a rectangular video frame or a part thereof. Following an object-based approach, the MPEG-4 video encoder applies the motion, texture and shape coding tools to the VOP using I, P, and B modes similar to the modes of MPEG-2. The encoder transmits texture, motion, and shape information of one VO within one bit stream. The bit streams of several VOs and accompanying composition information can be multiplexed such that the decoder receives all the information to decode the VOs and arrange them into a video scene (Figure 1).

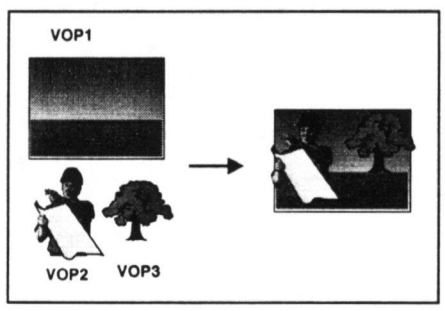

Figure 1: Object-based coding requires the decoder to compose different Video Object Planes (VOP) into a scene.

Two types of VOs are distinguished: For opaque objects, binary shape information is transmitted as a bitmap using a context-based arithmetic coder (Section 2). Transparent objects are described by binary shape and a transparency value. Alternatively, the transparency is described by a gray-scale alpha map (8 bits) defining the outline as well as the varying transparency of an object. The outline of these objects is also encoded as a bitmap using the binary shape coder.

Coding of texture for arbitrarily shaped VO is described in Section 3. MPEG-4 leaves it up mainly to the encoder to achieve efficient coding of texture at object boundaries by employing an appropriate texture extrapolation algorithm. The efficient coding of a VO with lossily encoded shapes requires an encoder architecture as described in Section 4.

In Section 5, experimental results will be presented. Different shape encoding modes are investigated in order to achieve a bit efficient representation of a VO or in order to allow for fast encoding that is important for real-time applications.

2. Context-Based Arithmetic Shape Coding

In order to enable content based access to video objects, MPEG-4 codes the shape of video objects. The shape is encoded as a bitmap. For binary shape coding, a rectangular bounding box enclosing the arbitrarily shaped VOP is formed such that its horizontal and vertical dimensions are multiples of 16 pels (macroblock size).

Each block of size 16x16 pels within this bounding box is called binary alpha block (BAB). Each BAB is associated with the co-located macroblock. Three types of BABs are distinguished and signaled to the decoder: Transparent blocks do not contain information about the object, opaque blocks are located entirely inside the object and boundary blocks cover part of the object as well as part of the background. For boundary blocks a context-based shape coder was developed. This coder exploits the spatial redundancy of the binary shape information to be coded. Pels are coded in scan-line order and row by row. In the following paragraphs, shape encoding in intra mode is described in Section 2.1. Then, this technique is extended to include an inter mode (Section 2.2).

2.1 Intra Mode

In intra mode, three different types of macroblocks are distinguished: Transparent and opaque blocks are signaled as macroblock type. The macroblocks on the object boundary containing transparent as well as opaque pels belong to the third type. For these boundary macroblocks, a template of 10 pels is used to define the causal context for predicting the shape value of the current pel (Figure 2a). For encoding the state transition, a

context-based arithmetic encoder is used. The probability table of the arithmetic encoder for the 1024 contexts was derived from sequences that are outside of the test set used for comparing different shape coders. With two bytes allocated to describe the symbol probability for each context, the table size is 2048 bytes. In order to avoid emulation of start codes like VOP start code, the arithmetic coder stuffs one '1' into the bitstream whenever a long sequence of '0' is sent.

The template extends up to 2 pels to the left, to the right and to the top of the pel to be coded (Figure 2a). Hence, for encoding the pels in the 2 top and left rows of a macroblock, parts of the template are defined by the shape information of the already transmitted macroblocks on the top and on the left side of the current macroblock. For the 2 right-most columns, each undefined pel of the context is set to the value of its closest neighbor inside the macroblock.

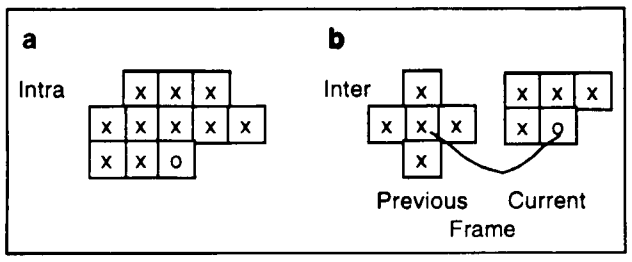

Figure 2: Templates for defining the context of the pel to be coded (o), a) defines the intra mode context, b) the context when coding in inter mode. The alignment is done after motion compensating the previous VOP [4].

In order to increase coding efficiency as well as to allow lossy shape coding, a macroblock can be subsampled by a factor of 2 or 4 resulting in a sub-block of size 8*8 pels or 4*4 pels, respectively. The sub-block is encoded using the encoder as described above. The encoder transmits to the decoder the subsampling factor such that the decoder decodes the shape data and then upsamples the decoded sub-block to macroblock size. Obviously, encoding the shape using a high subsampling factor is more efficient, but the decoded shape after upsampling may or may not be the same as the original shape. Hence, this subsampling is mostly used for lossy shape coding.

Depending on the upsampling filter, the decoded shape can look somewhat blocky. Several upsampling filters were investigated. The best performing filter in terms of subjective picture quality is an adaptive non-linear upsampling filter. The context of this upsampling filter is shown in Figure 3.

The efficiency of the shape coder differs depending on the orientation of the shape data. Therefore the encoder can choose to code the block as described above or transpose the macroblock prior to arithmetic coding.

2.2 Inter Mode

In order to exploit temporal redundancy in the shape information, the coder described above is extended by an inter mode requiring motion compensation and a different template for defining the context.

For motion compensation, a 2D integer pel motion vector is estimated using full search for each macroblock in order to minimize the prediction error between the previous coded VOP shape M'_{k-1} and the current shape M_k. The shape motion vectors are predictively encoded with respect to the shape motion vectors of neighboring macroblocks. If no shape motion vector is available, texture motion vectors are used as predictors. The shape motion vector of the current block is used to align a new template designed for coding shape in inter mode (Figure 2b). The template defines a context of 9 pels resulting in 512 contexts. The probability for one symbol is described by 2 bytes giving a probability table size of 1024 bytes. Four pels of the context are neighbors of the pel to be coded, 5 pels are located at the motion compensated location in the previous VOP. Assuming that the motion vector $(d_x d_y)^T$ points from the current VOP_k to the previous coded VOP'_{k-1}, the part of the template located in the previously coded shape is centered at $m'(x-d_x y-d_y)$ with $(x,y)^T$ being the location of the current pel to be coded.

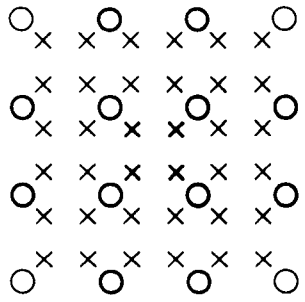

Figure 3: For shape upsampling, the upsampled pels (x) lie between the location of the subsampled pels (o). Neighboring pels (bold o) defining the values (transparent or opaque) of the pels to be upsampled (bold x).

In inter mode, the same options as in intra mode like subsampling and transposing are available. For lossy shape coding, the encoder may also decide that the shape representation achieved by just carrying out motion compensation is sufficient thus saving bits by avoiding the coding of the prediction error. The encoder can select one of 7 modes for the shape information of each macroblock: transparent, opaque, intra, inter with and without shape motion vectors, and inter with/without shape motion vectors and prediction error coding. These different options with optional subsampling and transposition allow for encoder implementations of different coding efficiency and implementation complexity.

3. Texture Coding of Boundary Blocks

For motion compensated prediction of the texture of the current VOP, the reference VOP is motion compensated using overlapped block motion compensation. In order to guarantee that every pel of the current VOP has a value to be predicted from, some or all of the boundary and transparent blocks of the reference VOP have to be padded. Boundary blocks are padded using repetitive padding: First boundary pels are replicated in horizontal direction, then in vertical direction making sure that if a value can be assigned to a pel by both padding directions an average value is assigned to the pel. Since this repetitive padding puts a significant computational burden on the decoder, a simpler

mean padding is used in a second step. Transparent macroblocks bordering boundary blocks are assigned to an average value determined by the pels of its neighboring padded blocks.

In order to encode the texture of a boundary block, MPEG-4 treats the macroblock as a regular macroblock and encodes each block using an 8*8 DCT. The decoder decodes the texture and discards all information that falls outside of the decoded shape. In order to increase coding efficiency, the encoder can choose the texture of pels outside of the object such that the bitrate is minimized. This non-normative process is also called padding. For intra mode, a lowpass extrapolation filter was developed, for inter mode setting these pels to 0 gives good efficiency.

4. Encoder Architecture

Figure 4a shows the block diagram of this object-based video coder. In contrast to the block diagram shown in the MPEG-4 standard, this diagram focuses on the object-based mode in order to allow a better understanding of how shape coding influences the encoder and decoder. Image analysis creates the bounding box for the current VOP S_k and estimates texture and shape motion of the current VOP S_k with respect to the reference VOP S'_{k-1}. Shape motion vectors of transparent macroblocks are set to 0. Parameter coding encodes the parameters predictively. The parameters get transmitted, decoded and the new reference VOP is stored in the VOP memory and also handed to the compositor of the decoder for display. The increased complexity due to the coding of arbitrarily shaped video objects becomes evident in Figure 4b that shows a detailed view of the parameter coding.

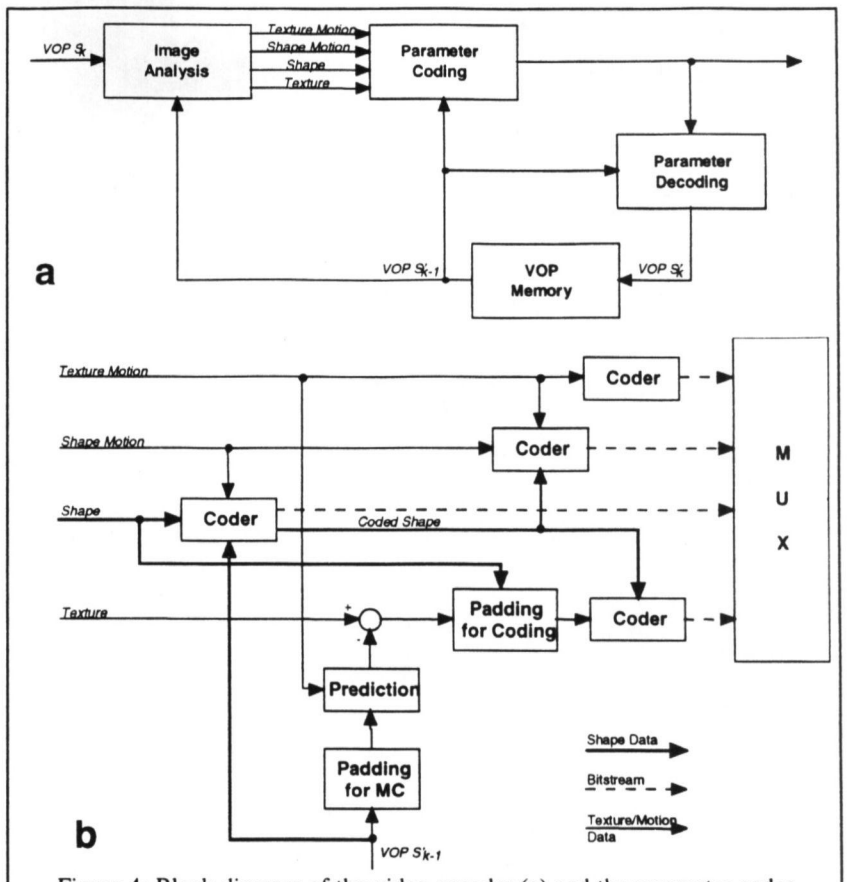

Figure 4: Block diagram of the video encoder (a) and the parameter coder (b) for coding of arbitrarily shaped video objects.

5. Experimental Results

The encoder is evaluated showing the influence of the non-normative texture padding prior to DCT coding and the influence of different shape coding options. These are the encoder options evaluated:

1. Padding for coding using original or coded shape.
2. Padding for coding using 0 padding or lowpass extrapolation filter.
3. Shape motion vectors: no motion compensation versus hierarchical shape motion estimation versus full search shape motion estimation versus texture motion vectors.
4. Lossy shape coding using fixed subsampling versus adaptive subsampling of shape.
5. Horizontal scanning of macroblocks versus vertical scanning versus adaptive scanning.

The following paragraphs will provide detailed experimental results in the full paper.

6. CONCLUSIONS

MPEG-4 Visual standardizes an object-based decoder. At the encoder, several choices regarding texture extrapolation at object boundaries and shape encoding are available to the encoder. It was found that for texture extrapolation a simple filter setting the texture outside of the *original* object shape to gray is sufficient. In the case of lossy shape coding, not using the *original* object shape for defining the support region of the texture extrapolation may increase the overall bitrate by more than 100%. Therefore, lossy shape coding makes only sense if the encoder architecture presented in this paper is used.

When using texture motion vectors for motion compensation of the shape, the overall bitrate increases usually by 1-2% with the benefit that shape motion vectors have not to be computed. Using a hierarchical shape motion vector estimator instead of a full search estimator decreases the computational load of the encoder without effecting the bitrate.

Fixing the macroblock scanning to horizontal instead of adaptively switching between horizontal and vertical scanning increases the bitrate by just 1% but saves significant compute time since the compute intensive arithmetic coding with context determination has to be used only half as often.

7. REFERENCES

[1] ISO/IEC JTC1/WG11 N1902, "Text for CD 14496-2 Visual", November 1997.

[2] ISO/IEC JTC1/WG11 N1901, "Text of CD 14496-1", November 1997.

[3] ISO/IEC JTC1/WG11 N1905, "Text of CD 14496-5", November 1997.

[4] N. Brady, F. Bossen, N. Murphy, "Context-based arithmetic encoding of 2D shape sequences", Special session on shape coding, ICIP 97, Santa Barbara, 1997