JSCAS 98

CODING OF ARBITRARILY SHAPED OBJECTS WITH BINARY AND GREYSCALE ALPHA-MAPS: WHAT CAN MPEG-4 DO FOR YOU?

Jörn Ostermann

AT&T Labs – Research, Room 3-231, 100 Schultz Dr., Red Bank, NJ 07701, USA
ostermann@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 shape coding algorithms and related texture coding algorithms are described. Whereas binary shapes are coded using a context-based arithmetic encoder with motion compensation, the coding of greyscale alpha maps requires to code the values of the alpha maps inside of the object in addition to the binary shape information. In order to allow for an efficient coding of the texture at the boundary of arbitrarily shaped objects, padding techniques are used in the encoder and decoder.

1. Introduction

MPEG-4 Visual will be the first international standard allowing the transmission of arbitrarily shaped video objects (VO) [1][2]. Following an object-based approach, MPEG-4 Visual 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. The compositor of the terminal takes the decoded VOs and arranges them into a video scene (Figure **1**). This enables applications like content-based storage and retrieval that have to provide access to video data based on object descriptions, where objects are described by texture, shape and motion. Studio and television post-production applications require editing of video content with objects represented by texture and shape and will benefit in coding efficiency that MPEG-4 provides. For collaborative scene visualization like augmented reality, we want to place video objects into the scene. Mobile multimedia applications require content-based interactivity and content-based scalability in order to allocate limited bitrate to fit the individual needs.

As with texture coding, efficiency of shape coding depends to a large extend on the encoder. Two types of VOs are distinguished: For opaque objects, binary shape information is transmitted (Section 2). Transparent objects are described by greyscale alpha maps (8 bits) defining the outline as well as the transparency of an object (Section 5). Texture coding for boundary blocks of arbitrarily shaped VOs is discussed in Section.3. Section 4 explains the structure of the MPEG-4 encoder when coding arbitrarily shaped video objects. An overview of the decoder complexity and coding efficiency is given in Section 6.

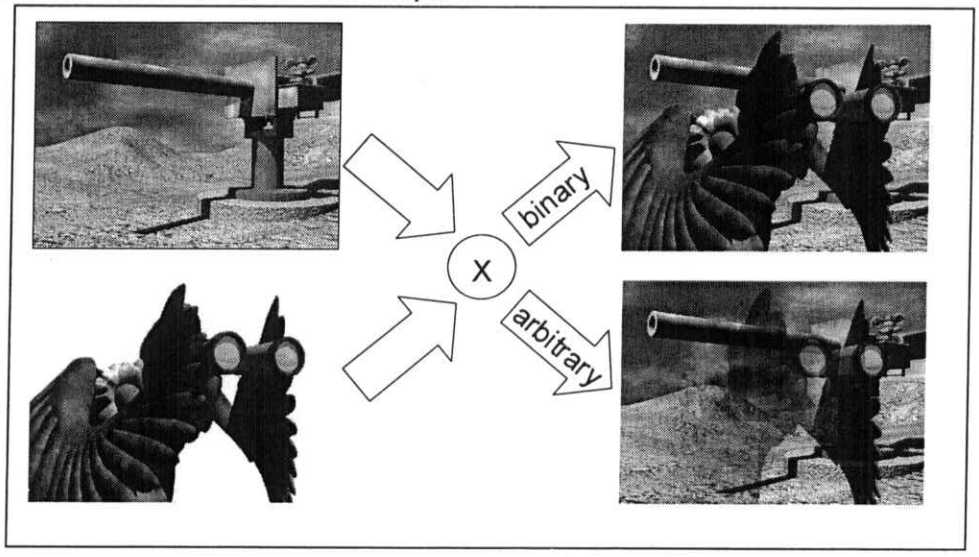

Figure 1: An arbitrarily shaped video object is composed onto a rectangular background image. Binary shape coding allows describing objects with constant transparency whereas grayscale alpha maps allow describing objects with arbitrary transparency providing for more flexibility for image composition.

2. Context-Based Arithmetic Shape Coding

In order to enable content based access to video objects, MPEG-4 codes the shape of video objects [3]. A frame with an arbitrarily shaped VO is called video object plane (VOP). 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. 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.

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.

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.

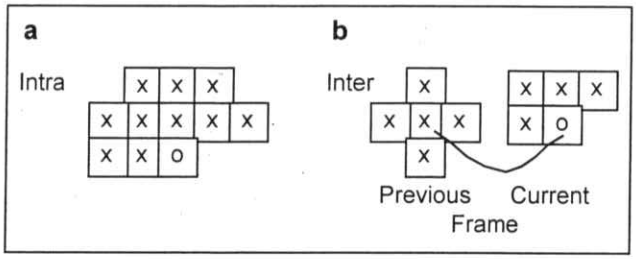

Figure 2: Templates for defining the context of the pel to be coded (o), a) defines the intra mode context, b) the context when coding in inter mode. The alignment is done after motion compensating the previous VOP.

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.

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.

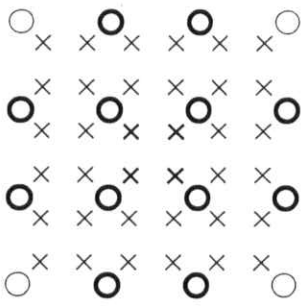

Figure 3: For shape upsampling, the upsampled pels (x) lie between the location of the subsampled pels (o). Neighboring pels (bold o) defining the values (transparent or opaque) of the pels to be upsampled (bold x).

3. Texture Coding of Boundary 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 called padding. For intra mode, a lowpass extrapolation filter was developed, for inter mode setting these pels to 0 results in good coding efficiency.

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. I.e., pels outside of the motion compensated arbitrarily shaped VOP get defined by this padding process. Boundary blocks are padded using repetitive padding: Boundary pels are the pels on the object boundary belonging to the VOP. 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.

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.

The parameter coder encodes first the shape of the boundary blocks using shape and texture motion vectors for prediction. Then shape motion vectors are coded. The shape motion coder knows which motion vectors to code by analyzing the possibly lossily encoded shape parameters. For texture prediction, the reference VOP is padded as described above. The prediction error is then padded using the original shape parameters to determine the area to be padded. Using the original shape as a reference for padding is again an encoder choice not implemented in the MPEG-4 CD software [1]. Finally, the texture of each macroblock is encoded using DCT.

5. Grey-scale alpha map coding

For a transparent object that has a constant transparency, the shape is encoded using the binary shape coder and the 8 bit value of the alpha map [4]. Blending the alpha map near the object boundary is not supported by the video decoder since this is a composition issue.

For object with varying alpha maps, shape coding is done in two steps. In the first step, the outline of the object is losslessly encoded as a binary shape (Section 2). In the second step, the actual alpha map is treated like the luminance of an object with binary shape and coded using padding, motion compensation and DCT as outlined in Section 3.

6. Experimental Results

In this section, 2 question are answered: How many bits do we have to spend in order to code arbitrarily shaped video objects and how much computation does the decoder require.

For coding of arbitrarily video objects like a weather announcer in a newscast, shape coding requires 500 bits/frame in intra mode and 300 bits in inter mode. For scenes with a moderate amount of motion like 2 kids playing ball, the required rate increases to 1700 bits/frame for intra and 1600 bits/frame for inter mode. These numbers are based on CIF, 10 Hz. For scenes with more complex objects and higher motion, the required bitrate increases further. If the goal of the encoder is to provide subjectively lossless coding, then the required bitrate can be reduced by 30% to 50%. Subjectively lossless coding can be achieved by prefiltering the shape information using morphological filters and allowing temporal prediction errors of the shape not to be compensated. When coding shapes lossily, the topology of the VO should not be changed such that no small independent areas at object boundaries are created. Again, this is an encoder issue not described in the literature or the standard.

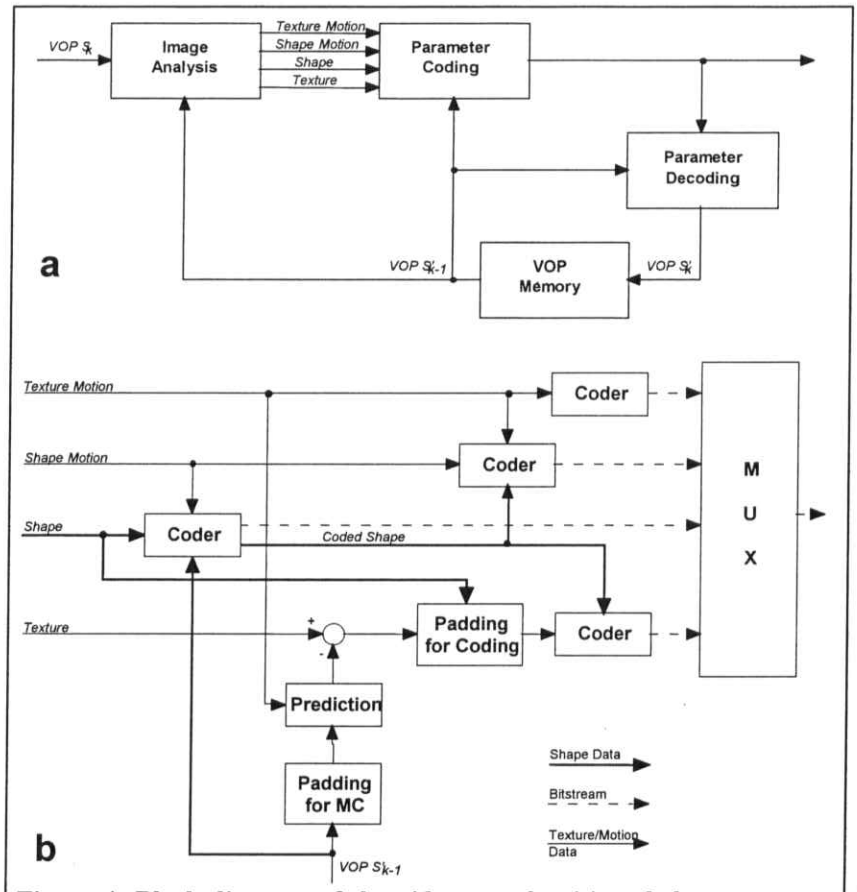

Figure 4: Block diagram of the video encoder (a) and the parameter coder (b) for coding of arbitrarily shaped video objects.

For gray-scale alpha map coding, the binary shape information has to be transmitted. The actual alpha map is coded like luminance information. There are two main differences between the alpha map and a luminance signal: 1.)Usually, the human observer is not very sensitive to quantization errors in the alpha mask. Therefore, the alpha map is much more coarsely quantized. 2.) The alpha map contains few high frequencies. Hence, alpha maps can be coded with relatively few bits.

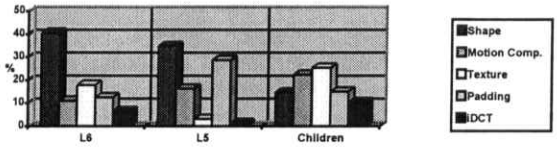

Figure 5: Allocation of relative time to different tasks of a decoder in % shown for 3 different test sequences (from [5]).

Coding of arbitrarily shaped objects instead of rectangular frames requires shape coding as well as padding. In Fig. 5, the time an optimized coder spends decoding shape and padding is given for several bitstreams with complicated shapes. Decoding of shapes with a high portion of boundary blocks can require up to 50% of the decoding time. Therefore, concern was raised that the complexity of the shape decoder is too high for low complexity terminals. There are several methods by which the complexity can be reduced by more than 50%:

- Coding of shape only in subsampled mode.
- Replace padding according to Section 3 with constant value padding with the value for padding transmitted in the bitstream.
- Switch off the adaptive scanning of the shape blocks.
- Replace the upsampling filter with a simple pel repetition filter.

This reduction comes at a price. The bitrate for this low complexity coder increases up to 10%.

7. Conclusions

MPEG-4 is the first international standard covering shape coding. It will be finalized in November 1998. The purpose of the shape coder within MPEG-4 is to enable many new functionalities and applications like object-based database access, content-based image and video representation, video editing, efficient storage of movies prior to composition and more. The primary purpose is not to increase coding efficiency for conventional video although that has also been shown for some video sequences.

The MPEG-4 enables coding of binary shape signals where the object has a constant transparency value allowing for translucent objects and grayscale alpha maps that allow objects to have varying transparency.

In order to allow efficient encoding of the arbitrarily–shaped video objects, special attention has to be given to optimal texture prediction at object boundaries. The presented encoder architecture allows this. For terminals with low computational power significant savings in computational complexity can be achieved by simple changes to the coder. They are currently considered by MPEG-4.

8. References

[1] "Text for CD 14496-2 Visual", ISO/IEC JTC1/SC29/WG11 MPEG97/N1902, November 1997.

[2] J. Ostermann, E. S. Jang, J.-S.Shin, T. Chen, "Coding of Arbitrarily Shaped Video Objects in MPEG-4", Special session on shape coding, ICIP 97, Santa Barbara, 1997.

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

[4] W. Chen, M. Lee, "Alpha-Channel Compression in Video Coding", Special session on shape coding, ICIP 97, Santa Barbara, 1997.

[5] "Report of the adhoc group on binary shape complexity at simple profile", ed. Andy Hotchkiss, ISO/IEC JTC1/SC29/WG11 MPEG98/M3183, San Jose Meeting, February 98.