ELSEVIER

Signal Processing: *Image Communication* 15 (2000) 585–599

SIGNAL PROCESSING: IMAGE COMMUNICATION

www.elsevier.nl/locate/image

A contour-based approach to binary shape coding using a multiple grid chain code

Paulo Nunes[a,*], Ferran Marqués[b], Fernando Pereira[a], Antoni Gasull[b]

[a]*Instituto Superior Técnico/Instituto de Telecomunicações, Av. Rovisco Pais – 1049-001 Lisboa, Portugal*
[b]*Universitat Politècnica de Catalunya, UPC, Campus Nord, Modul D5 – 08034 Barcelona, Spain*

Abstract

This paper presents a contour-based approach to efficiently code binary shape information in the context of object-based video coding. This approach meets some of the most important requirements identified for the MPEG-4 standard, notably efficient coding and low delay. The proposed methods support both object-based lossless and quasi-lossless coding modes. For the cases where low delay is a primary requirement, a macroblock-based coding mode is proposed which can take advantage of inter-frame coding to improve the coding efficiency. The approach presented here relies on a grid different from that used for the pixels to represent the shape – the hexagonal grid – which simplifies the task of contour coding. Using this grid, an appraoch based on a differential chain code (DCC) is proposed for the lossless mode while, for the quasi-lossless case, an approach based on the multiple grid chain code (MGCC) principle is proposed. The MGCC combines both contour simplification and contour prediction to reduce the number of bits needed to code the shapes. Results for alpha plane coding of MPEG-4 video test sequences are presented in order to illustrate the performance of the several modes of operation, and a comparison is made with the shape-coding tool chosen by MPEG-4.

Keywords: Shape coding; Contour coding; Hexagonal grid; Multiple grid chain code; Motion estimation/compensation; Object-based video coding

1. Introduction

One of the main features of the MPEG-4 standard is its capability to describe scenes in terms of their compositing entities and their evolution in time [5]. In the MPEG-4 Video framework, each one of these separate entities is called a video object (VO). VOs can be classified into three types with respect to its alpha plane information: opaque, with constant transparency, and with variable transparency [16]. Therefore, the separate transmission of a VO requires the encoding of its alpha plane, notably the support of the alpha plane and the transparency information. While for binary shapes only the alpha plane support has to be coded, grey-level shapes require, in addition, the coding of transparency information. In MPEG-4, transparency information is encoded either with a single transparency value or with techniques very similar to those used for luminance information [14]. The

* Corresponding author. Tel: + 351-1-841-84-60; fax: + 351-1-841-84-72.

E-mail addresses: paulo.nunes@lx.it.pt (P. Nunes), fernando.pereira@lx.it.pt (F. Pereira), ferran@gps.tsc.upc.es (F. Marqués), gasull@gps.tsc.upc.es (A. Gasull)

0923-5965/00/$ - see front matter
PII: S0923-5965(99)00041-7

efficient encoding of the shape information has required the development of very sophisticated techniques, during a process where many technical proposals were carefully evaluated [12].

During the MPEG-4 standardisation process, two main families of binary shape coding approaches have been proposed: bitmap-based and contour-based techniques. Bitmap-based techniques directly encode the shape pixels as belonging to the object or to the background. Two main methods have been presented in this area: the modified modified Reed [19] and context-based arithmetic encoding [2]. Contour-based techniques represent the shape by its boundary. Three different approaches have been proposed relying on the contour representation: vertex-based [4], baseline-based [6], and chain-code-based [15] shape encoders.

Due to the MPEG-4 requirements for shape coding [11], various techniques have developed lossless as well as lossy modes. However, during the comparison process, the need for quasi-lossless techniques arose. Such techniques may introduce losses in the encoding process but these should not be noticeable for the user.

This paper deals with a contour-based shape coding technique that can work in two modes: lossless and quasi-lossless. The paper presents in detail the method that was proposed by the authors in [15,8], extends the discussion of the more complex cases that may be found in the encoder, and describes the decoder. The technique relies on the representation of shape information on a grid different from that used for the luminance information. The shape grid is called the *hexagonal grid* [9,18] and simplifies the contour coding task. Using this grid, an approach based on a differential chain code [3] is proposed for the lossless mode. The quasi-lossless case is handled by an approach based on the multiple grid chain code (MGCC) principle [10]. The MGCC combines both contour simplification and contour prediction to reduce the number of bits necessary for coding the shapes.

The paper is structured as follows. After the introduction, a brief review of the MPEG-4 binary shape coding technique is presented in Section 2. Section 3 discusses the hexagonal grid concept. While Section 4 describes the basis of the lossless shape-coding technique, Section 5 presents the basis of the quasi-lossless shape-coding approach. Section 6 deals with the actual implementation of the proposed shape coding for both the intra-frame and inter-frame modes. Finally, in Section 7 simulation results are presented and conclusions are given.

2. The MPEG-4 binary shape coding approach

Due to its relevance in this paper, this section describes the bitmap-based binary shape coding technique, called context-based arithmetic encoding (CAE), adopted by MPEG for the MPEG-4 standard [14].

To code the binary shape of a video object plane (VOP) with CAE, the alpha plane of the VOP is enclosed in the tightest rectangle containing all the shape pixels – the bounding box (BB) – with dimensions multiple of the macroblock (MB) dimensions, i.e. 16×16 pixels (Fig. 1). This BB is built minimising the number of MBs with shape pixels. This way of creating the BB is especially useful in the case of performing both shape and texture coding together since it decreases the number of MBs that have to be processed and transmitted [14].

Each block of 16×16 pixels within this BB is called a binary alpha block (BAB) and is encoded in a raster scanning order. There are three different categories of BABs as illustrated in Fig. 1:

1. TRANSPARENT (all BAB alpha values are 0),
2. OPAQUE (all BAB alpha values are 255),
3. BOUNDARY (BAB alpha values are either 0 or 255).

While TRANSPARENT and OPAQUE BABs are encoded by a single word describing the BAB type, BOUNDARY blocks may need further data, besides the BAB type, to completely encode all the shape pixels, notably motion information and context data.

2.1. Binary alpha block-type encoding

The methods used to encode the BAB are indicated by the BAB type. Each BAB can be

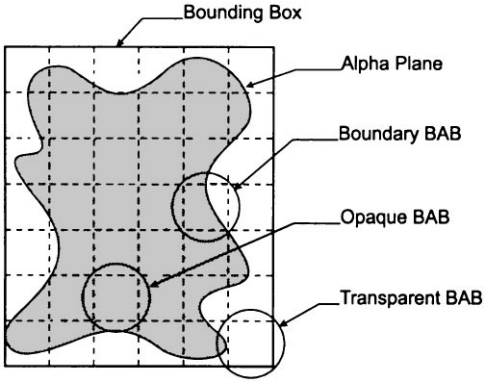

Fig. 1. Alpha plane and corresponding bounding box.

Table 1
List of BAB types for CAE

BAB type	Semantic	VOP Type
0	MVDS = =0 && NO UPDATE	P-, B- VOPs
1	MVDS!= 0 && NO UPDATE	P-, B- VOPs
2	TRANSPARENT	All VOP Types
3	OPAQUE	All VOP Types
4	INTRA CAE	All VOP Types
5	MVDS = =0 && INTER CAE	P-, B- VOPs
6	MVDS != 0 && INTER CAE	P-, B- VOPs

encoded with one of the seven different types listed in Table 1, where MVDs is the motion vector difference for shape, i.e. the difference between the actual motion vector and the motion vector predictor for the current BAB, and NO UPDATE means that the current BAB can be completely reconstructed by motion compensation without the need for any further information.

2.1.1. Intra VOPs

For intra coded VOPs, the number of BAB types allowed is restricted to three out of the seven different values available – (2–4) – encoded using a VLC table indexed by a BAB type context C. For each BAB, the context is computed based on the values of the already encoded neighbouring BAB types, and the BAB type of the BAB being encoded. Fig. 2 shows the template used to compute the context C using Eq. (1), where c_k is the type of the kth BAB in the template.

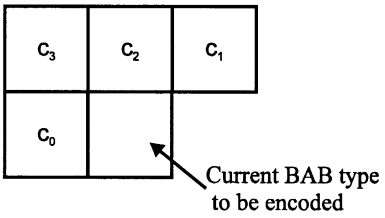

Fig. 2. Template for computing the BAB type context for I-VOPs.

$$C = \sum_{k=0}^{3} (c_k - 2)3^k. \quad (1)$$

If the computation of the context C involves BABs outside the current VOP BB, the corresponding BAB types are assumed to be transparent.

2.1.2. Inter VOPs

In the case of inter coded VOPs (P- and B-VOPs) all BAB types listed in Table 1 are allowed and the BAB-type information is encoded using a different VLC table indexed by the co-located BAB type in the selected reference VOP and the BAB type of the BAB being encoded.

If the sizes of the current and reference VOPs are different, some BABs in the VOP being encoded may not have a co-located equivalent in the reference VOP. In this case the BAB-type matrix of the reference VOP is manipulated in order to match the size of the current VOP [14].

2.2. Binary alpha block motion compensation

Besides the type information, motion information can also be used to encode the BAB data. In this case the BAB is encoded in inter mode and one motion vector (MVs – motion vector for shape) for each BAB is used. Similar to what happens in texture coding, in the MPEG-4 shape coding technique the shape motion vectors are encoded differentially, which means that for each motion vector a predictor is built from neighbouring shape motion vectors already encoded and the difference between this predictor and the actual MVs – the MVDs – is VLC encoded. When no valid shape

Table 2
List of BAB sizes for CAE

CR	BAB size in pixels
1	16×16
2	8×8
4	4×4

motion vectors are available for prediction texture motion vectors can be used.

2.3. Binary alpha block size conversion

The compression ratio of the CAE technique can be increased by allowing the shape to be encoded in a lossy mode. This can be achieved by encoding a down-sampled version of the BAB. In this case the BAB can be down-sampled by factor of 2 or 4 before context encoding and up-sampled by the same factor after context decoding. The relation between the original and the down-sampled BAB sizes is called conversion ratio (CR). Table 2 presents the different BAB sizes allowed, depending on the value of the CR parameter. The error introduced by this down-sampling/up-sampling operation is responsible for the lossy encoding and is called the conversion error.

Encoding the shape using a high down-sampling factor provides a better compression ratio; however the reconstructed shape after up-sampling may exhibit annoying artifacts. In order to minimise these artifacts an adaptive non-linear up-sampling filter has been adopted by MPEG [14].

2.4. Context-based arithmetic encoding

For BOUNDARY blocks which are not exclusively encoded by motion information, context-based arithmetic encoding is applied. For these blocks the BAB data is encoded by a single binary arithmetic codeword (BAC). In this case, each binary shape pixel in the BAB (or in its down-sampled version) is encoded in a raster scanning order (until all pixels are encoded). The process of encoding a given shape pixel using CAE is the following:

1. Compute a shape context number based on a template.
2. Use this context number to index a probability table.
3. Use the indexed probability to drive a binary arithmetic encoder.

Since the encoding performance depends on the scanning order of the BAB, the encoder is allowed to transpose (matrix transposition) the BAB before encoding. This operation is signalled to the decoder through a 1 bit flag called scanning type.

2.4.1. Context computation

The core of the CAE technique is the exploitation of the spatio-temporal redundancy in the motion-compensated BAB data using causal contexts to predict the shape pixels according to predefined templates. Fig. 3 shows the different templates used in CAE encoding.

For intra coded BABs no motion compensation is used; thus only the spatial redundancy is exploited by computing a 10 bit context for each pixel

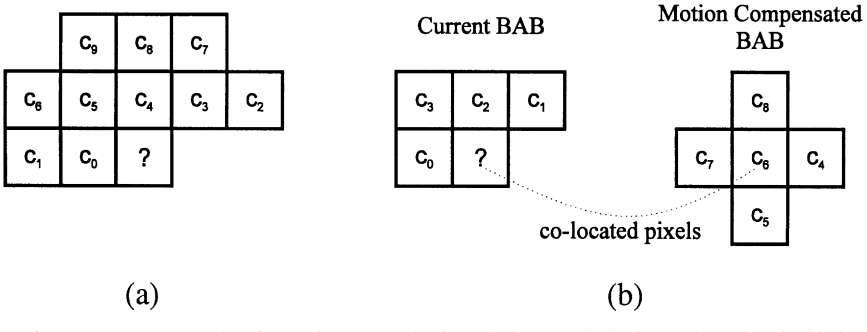

Fig. 3. CAE templates for context computation for: (a) intra coded BABs; (b) inter coded BABs, where the pixel being encoded is marked with '?'.

of the BAB using Eq. (2) with $N = 9$, according to the template of Fig. 3(a), where $c_k = 0$ for transparent pixels and $c_k = 1$ for opaque pixels.

$$C = \sum_{k=0}^{N} c_k 2^k. \tag{2}$$

Temporal redundancy can additionally be exploited for inter coded BABs by computing a 9 bit context using also pixels from the motion compensated BAB (besides the pixels from the BAB being encoded). In this case the context is computed using Eq. (2) with $N = 8$, according to the template of Fig. 3b. As in the intra case, $c_k = 0$ for transparent pixels and $c_k = 1$ for opaque pixels.

When building contexts some special cases may appear, notably some pixels may be out of the current VOP BB or the template may cover pixels from BABs which are not known at decoding time. These cases have special pre-defined rules which are known both by the encoder and the decoder [14].

3. The basics of the contour-based approach

3.1. The hexagonal grid

As previously stated, chain-code-based shape coding techniques rely on a contour representation to code the shape information. In order to allow the coding of arbitrary binary shapes, an adequate representation of the shape boundary is necessary. This representation must not only describe the shape unambiguously, but also allow decrease in the redundancy of the encoded representation as much as possible.

A common approach for the binary shape representation is to describe the shape by those pixels of the background (pixels in the BB outside the shape) that have at least one 4-connected neighbour belonging to the shape (the foreground), defining an outer pixel-based contour (pixels marked with a grey level in Fig. 4(a)). In a similar way, the shape can be described by the interior pixels that have at least one 4-connected neighbour belonging to the background, defining an inner pixel-based contour (pixels marked with a grey level in Fig. 4(b)). Although these pixel-based descriptions allow an unambiguous representation of binary shapes, they require some contour elements to be coded more than once and the definition of additional chain code symbols [18].

To simplify the process of representing the shape contour and to avoid the above-mentioned drawbacks, a different grid has been adopted based on the edge sites: the hexagonal grid. Assuming that the pixels of the shape are located in a rectangular grid (squares in Fig. 5(a)), the (hexagonal) edge grid corresponds to all the sites located between two adjacent pixels, extended to the image borders (segments in Fig. 5(a)). It gets its name precisely from the fact that each edge site has six neighbour edge sites. Edge elements are active if they are between

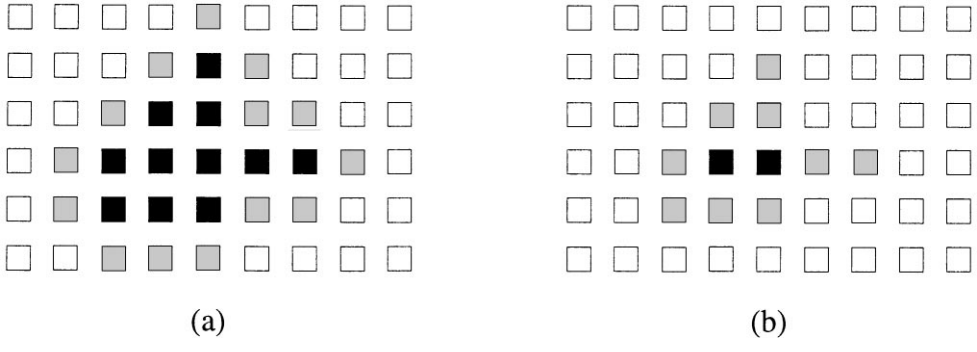

Fig. 4. Shape and corresponding pixel-based contour representations: (a) outer contour; (b) inner contour.

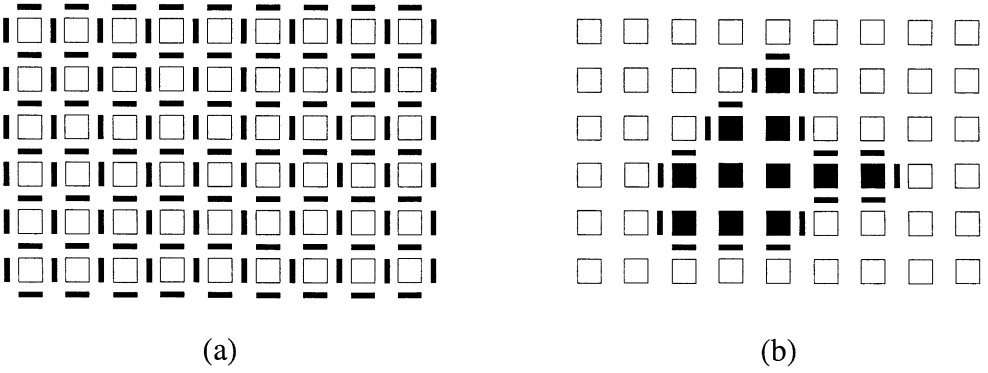

Fig. 5. Shape and corresponding edge-based contour representation: (a) pixel grid and hexagonal edge grid; (b) edge-based contour representation.

two pixels with different labels, one in the background and another in the foreground (Fig. 5(b)). This edge-based representation allows an easy and unambiguous conversion from the label image to the contour image and vice versa, providing a very natural framework to handle all shape-coding techniques based on contour descriptions (e.g. chain-code-based techniques and polygonal approximations).

3.2. Direct and differential chain code

In the context of chain-code-based techniques, the contour-based representation of a shape is built by tracking the contour and representing each movement by a chain code symbol. The set of possible movements depends on the type of contour representation. In the case of the pixel-based contour representations, eight movements are needed, if an 8-connected chain code is considered, or four in the case of the 4-connected chain code (Figs. 6(a) and (b), respectively).

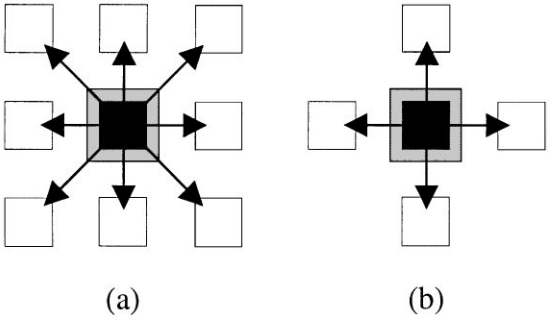

Fig. 6. Pixel-based contour representation using (a) 8-connected chain code; (b) 4-connected chain code.

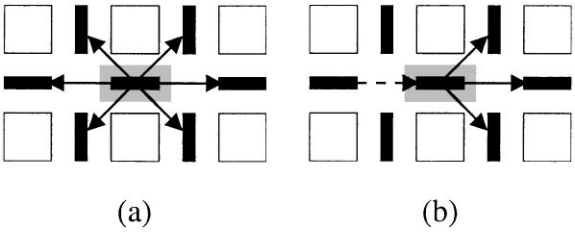

Fig. 7. Edge-based contour representation: example of (a) direct and (b) differential chain code.

In the case of the edge-based contour representation, from a given edge site only six possible sites can be reached when tracking a contour and, therefore, there are only six possible movements (Fig. 7(a)). In this case, however, since the edge-based representation guarantees that no contour element has to be coded twice, it is possible to use a differential chain code (DCC) (Fig. 7(b)). When using a differential description of the contour (as in the DCC case), the number of possible movements reduces to 3: {TURN RIGHT, TURN LEFT, STRAIGHT AHEAD} (Fig. 7(b)). In order to use the differential description, the coding process has to keep in memory the previous movement. This can be done by tracking the direction of the previous movement: {NORTH, SOUTH, EAST, WEST}.

4. Lossless shape coding: differential chain code (DCC)

This section applies the contour-based coding approach described above to the lossless coding of shape. This lossless coding technique uses the hexagonal grid to represent the contour (edge-based representation), as described in the previous section, and a differential chain code representation to code it.

As in the MPEG-4 technique, to code the binary shape, the shape pixels are enclosed in a BB with dimensions multiple of the MB dimensions.

4.1. Selection and coding of the initial contour elements

The contour element where the shape encoding starts is the first active edge site found when scanning the contour image from left to right and top to bottom. Therefore, the first active site is always a horizontal element and thus the initial direction for the encoding process can be set to EAST. Note that the decoder shares this information and, therefore, it does not have to be sent. To independently code each 4-connected region in the alpha plane, a different initial contour element is used. Therefore, after coding the contour of each connected region, the encoder looks for a new initial element until all contour elements have been coded. Since chain coding can independently code each 4-connected region of a possible multi-region VO, it enables an easy redefinition of VOs: a VO containing two separate regions can easily be converted in to two independent VOs.

The position of the first initial element is coded in an absolute way. Given the way the shape BB is constructed [14], the first initial element in the contour image will be always on the first row of MBs. Thus, it is coded using $\log_2 VOP\_width$ bits for the horizontal co-ordinate and 4 bits ($\log_2 16$) for the vertical co-ordinate.

The coding of the subsequent initial elements is done in a way relative to the position of the previous one. To do so, all possible candidates to initial elements (i.e. all the horizontal contour elements, except those in the last row) are indexed following a top-to-bottom and left-to-right scan. Each initial element is then coded using its differential index (i.e. $current\_index - previous\_index$) with $\log_2 (BB\_pixels - previous\_index)$ bits, where *previous_index* and *current_index* are the indices of the previous and current initial elements, respectively, and *BB_pixels* coincide with the number of possible candidates to initial elements (Fig. 5(a)). Therefore, the length of the word used for coding each initial element is adapted with respect to the position of the previous initial element. At the decoder side, the current index of the initial element is computed adding to the decoded value, the value of the previously decoded index for the current VOP.

4.2. Encoding the contour elements: building the DCC

After encoding the initial element of a given 4-connected region, the encoder searches for the following non-coded contour element (active edge site), testing each direction in the following order:

1. RIGHT,
2. STRAIGHT AHEAD,
3. LEFT.

Priority is given to the movement RIGHT so that the contour tracking algorithm first completes every 4-connected region.

For each contour element, the encoder outputs the corresponding DCC symbol from the set of possible symbols, i.e. {RIGHT, STRAIGHT AHEAD, LEFT}. These symbols are then gathered into groups of n symbols and Huffman encoded. When the encoder reaches the end of the current contour with an incomplete group of symbols to code, it chooses from the corresponding Huffman table the shortest word, in terms of number of bits, having this group of symbols as prefix. The contour of a 4-connected region is finished when the contour tracking reaches an already coded contour element, i.e. in the case of the DCC, the initial contour element of the region being coded. After encoding the last element of the contour, the encoder has to signal to the decoder whether there are more contours left to code or not. This is done by putting a one bit flag in the bitstream – the LAST_CONTOUR flag – following the last symbol. When set to '0', this flag indicates that there are more contours

(associated to non-connected regions) to code, and that, when set to '1' all contours for this alpha plane have been coded. The encoding of the alpha plane stops when a chain code has been built for each 4-connected region of the shape and thus all contour elements have been coded.

4.3. Decoding the DCC

The decoding of the DCC is the inverse of the encoding process. After decoding the initial element of a contour, the decoder looks for the next DCC symbol and tracks the contour according to the symbols being decoded, until the initial element is reached. When the initial element is reached, the decoder reads the LAST_CONTOUR flag (1 bit), in order to know whether there are more contours to decode for this alpha plane. If a LAST_CONTOUR flag = '1' is detected, the contour decoding is stopped.

After having completely decoded the contour image for the current alpha plane, the shape decoder has to construct the label image, this means the binary shape. This is done by jointly scanning, just once, the contour and label images from top to down and left to right, propagating the labels based on the contour information.

5. Quasi-lossless shape coding: multiple grid chain code (MGCC)

The technique here proposed for quasi-lossless shape coding is an extension of the differential chain coding, using the edge-based contour representation, which can encode more than one contour element with a single MGCC code. The main difference between DCC and MGCC lies in the fact that in DCC all codes represent a movement between neighbour contour elements while MGCC may link contour elements that are not neighbours. Furthermore, the distance between two contour elements linked by a MGCC code depends on the code. The method tries to increase the probability of large movements and, thus, to reduce the number of codes necessary to represent the contour. Large movements result in approximations of the real contour and lead to losses in the decoding process. Nevertheless, such losses are very small being, at most, of one pixel per MGCC code.

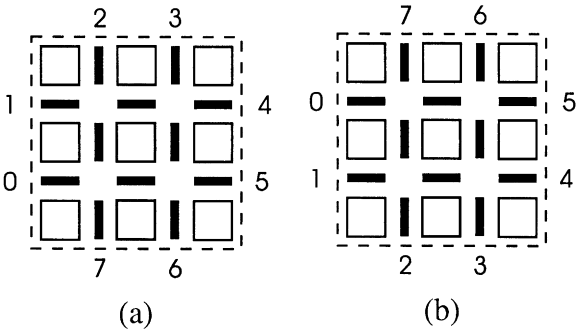

Fig. 8. Definition of MGCC cells: (a) clockwise and (b) counterclockwise.

5.1. MGCC basics

5.1.1. The MGCC cells

MGCC uses as basic cell a 3×3 pixels area of the hexagonal grid (Fig. 8) and a single movement is used to go through the cell. In Fig. 8, starting from the input contour element marked with 0, seven possible output contour elements can be reached $\{1, \ldots, 7\}$, going through the cell. Each one of these output contour elements represents a different movement and thus a different contour configuration. For compression efficiency reasons, there are two types of cells: clockwise and counterclockwise (Figs. 8(a) and (b), respectively). Its choice depends on the evolution of the contour and maximises the number of contour elements coded per cell.

5.1.2. The MGCC losses

A MGCC code does not always represent a unique contour configuration. As illustrated in Fig. 9, a movement (e.g. from 0 to 3) may correspond to two different contour configurations. Therefore, the decoding algorithm has to decide between two possible contour configurations. The ambiguity in this decision introduces the coding losses. Note, however, that the only possible error is the labelling of the central pixel in the cell. That is, the maximum error is the wrong labelling of isolated pixels in the borders of the regions and never in their interior.

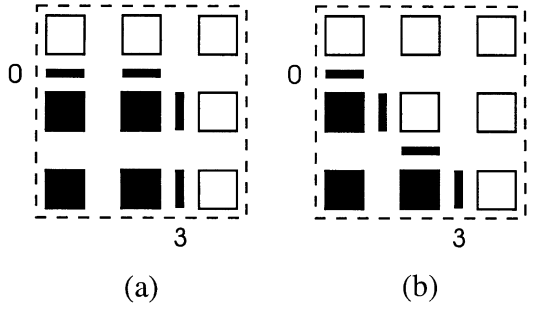

Fig. 9. Ambiguities introduced by MGCC.

5.1.3. Dynamic selection of the MGCC cells

As can be seen in Fig. 8, some of the MGCC symbols represent larger movements than others. This is the case for symbols 3 and 4 with respect to symbols 1 and 7. As a consequence, the more often these symbols are used, the higher the performance of the coding technique.

In order to use as many large movements as possible, the MGCC technique dynamically selects the type of cell between clockwise and counterclockwise. The selection of the next MGCC cell type relies on the prediction of the contour trajectory, assuming boundary smoothness and using a rule known by both the encoder and the decoder. The following selection rules try to increase the frequency of the symbols corresponding to larger movements, assuming that the contour trajectory does not change from cell to cell:

Output symbol	Previous cell type	Next cell type
1,2,3	Clockwise	Counterclockwise
	Counterclockwise	Clockwise
4,5,6,7	Clockwise	Clockwise
	Counterclockwise	Counterclockwise

Fig. 10 shows the next cell types for all possible contour configurations of a counterclockwise cell. Note that the selection of the type of cell automatically defines the position of the cell in the image grid. Therefore, two consecutive cells do not have to be aligned as shown in some cases of Fig. 10.

Fig. 11 illustrates the improvement achieved when selecting dynamically the next cell showing two possible solutions for encoding the same contour segment. The first solution uses cell adaptation, and as a consequence the position of the cell used for the second movement is modified, allowing the second part of the contour to be coded using only movement 4. The second solution does not perform cell adaptation and an extra symbol is needed to code the same contour.

5.2. Selection and encoding of the initial contour elements

For the selection and encoding of the initial contour elements, the same algorithm as that for the lossless shape coding mode is applied, i.e. the shape is enclosed in a BB and the contour element where the contour encoding starts is the first active edge site found when scanning the image from left to right and top to bottom.

5.3. Encoding the contour elements: building the MGCC

The initial contour element selected is the input element in an MGCC cell (site marked 0 in Fig. 8). As for the lossless case, after encoding the initial element, the encoder searches for the next non-coded contour elements, testing each direction in the following order:

1. RIGHT,
2. STRAIGHT AHEAD,
3. LEFT.

Similar to the lossless case, priority is given to the movement RIGHT so that the contour tracking algorithm first completes every 4-connected region. After the encoder finds an output element of the MGCC cell, it generates the corresponding symbol from the set of possible MGCC symbols, i.e. $\{1, \ldots, 7\}$. Symbols will be grouped and Huffman encoded. The output site becomes the input site of the following cell. The contour of a 4-connected region is completed when the contour tracking reaches an already coded contour element.

In the MGCC case, since the type and position of the cells is changed dynamically, it is possible that the last movement in a contour does not coincide with the initial contour element, but with an

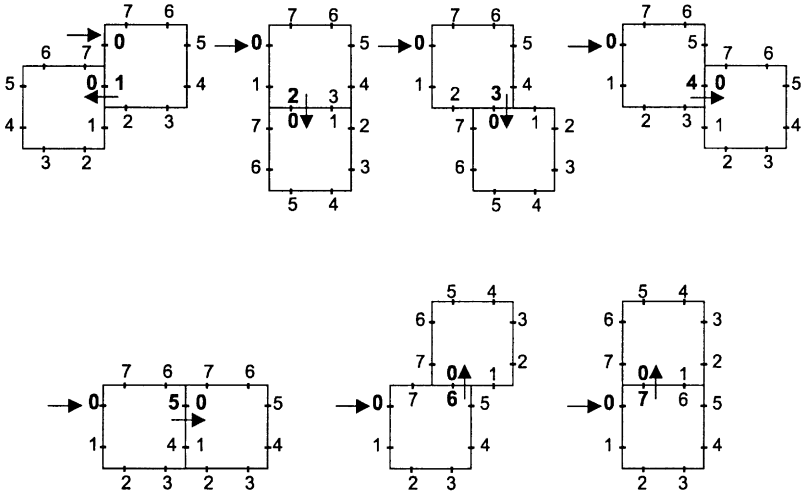

Fig. 10. Selection of the next MGCC cell for a counterclockwise cell.

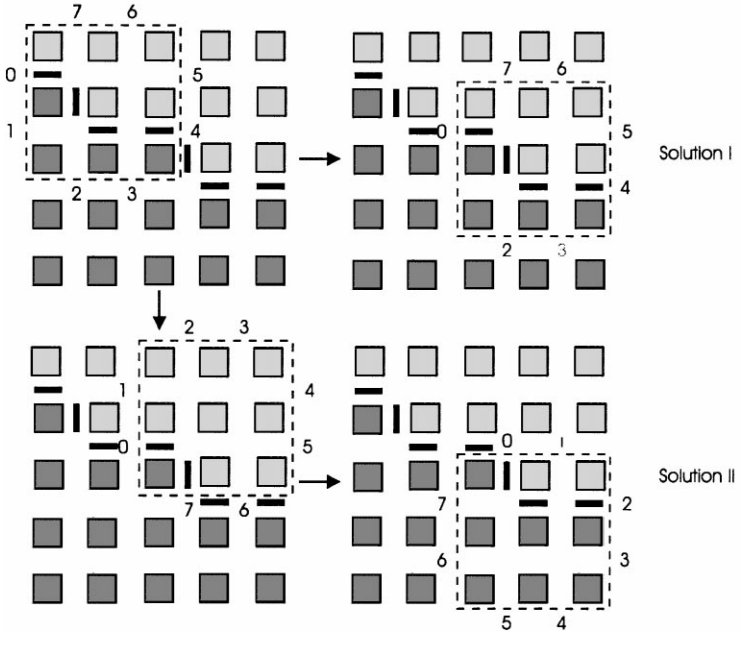

Fig. 11. Example of MGCC cell adaptation.

element inside the first cell of the contour being coded. In order that the decoder can detect the end of the contour, it needs to know without ambiguities all the contour elements of the first cell. This is done by sending one bit which allows to decode the first MGCC cell without ambiguities after the initial MGCC symbol. This bit tells the decoder whether the second contour element is in the STRAIGHT AHEAD direction or not, which is sufficient to inform the decoder which precise contour shall draw for the first cell.

Similar to the lossless case, after encoding each contour, the encoder tells the decoder whether there are more contours to code by means of the

LAST_CONTOUR flag bit. The encoding of the VOP alpha plane stops after a multiple grid chain code has been built for each 4-connected region in the shape and all contours have been coded.

5.4. Decoding the MGCC

Although MGCC decoding is again the inverse of the encoding process, for each decoded MGCC symbol, only the input and output contour elements of the corresponding cell are known for sure. The interior elements are not precisely known and thus need to be defined by the decoder. For these cases, the decoder makes a choice, labelling the corresponding sites with a temporary "uncertainty label". The input and output cell elements are labelled with a definitive label. While temporary labels may be changed if topological inconsistencies are found, definitive labels are never changed. This way, the input and output contour elements, whose locations are fixed by the encoding process, are used as anchor sites to define the interior ones. When the information provided by the input and output contour elements is not sufficient to allow a decision to be taken, the configuration producing the smoothest contour is chosen.

The decoding of each contour is similar to the DCC case. After decoding the initial contour element, the decoder decodes the first MGCC cell and the flag indicating which interior contour configuration has to be chosen for the first cell. For the subsequent contour element, it decodes the next MGCC symbol and tracks the contour according to the symbols being decoded and the decoding rules stated above. This process continues until reaching the initial cell. Then the decoder reads the LAST_CONTOUR flag (1 bit) in order to know whether there are more contours to decode for this alpha plane or not. If a LAST_CONTOUR flag = '1' is detected, contour decoding is stopped.

As in the DCC case, after having completely decoded the contour image for the current alpha plane, the shape decoder has to construct the label image. This is done by jointly scanning the contour and label images, from top to down and left to right. However, since in the MGCC case the shape is coded with losses, a special processing in the conversion from contours to labels is necessary in order to detect and correct possible invalid contour configurations caused by "less fortunate" decisions in the assignment of the temporary labels.

6. Object versus macroblock-based coding modes

In the previous sections, the chain-coding techniques for lossless and quasi-lossless intra binary shape coding were presented. Depending on whether the shape is coded simultaneously or not, the object-based or MB-based coding modes are used.

For non-real-time applications, where low delay is not a primary requirement, an object-based coding scheme is adopted where the shape is coded (and multiplexed) simultaneously – *object-based mode*. Thus, a video decoder using this mode can start decoding motion and texture information only after the whole shape has been decoded. An MB-based coding scheme is also proposed where the shape is divided into MBs of 16×16 pixels that are then independently encoded in a raster scanning way, thus allowing the video decoder to decode the motion and texture of each MB immediately after decoding the corresponding shape – *MB-based mode*. This coding mode is typically less efficient than the object-based coding mode since each shape MB is independently coded without using any information from the neighbouring source (if no motion is used). While for the object-based coding mode only intra coding has been considered, for the MB-based coding mode both intra and inter coding modes can be used. For the inter coding mode, and in order to improve coding efficiency, motion compensation is used, without coding the residues, and with a varying threshold depending on the acceptable shape degradation. Another approach to object-based coding mode, in the context of partition coding, using temporal prediction, is presented in [7].

While for the object-based coding mode, the shape is transmitted whole at once, for the MB-based coding mode, the shape information is sent MB by MB. For coding purposes, each shape MB is classified into three categories:

1. TRANSPARENT (all MB alpha values are 0),
2. OPAQUE (all MB alpha values are 255),
3. BOUNDARY (MB alpha values are either 0 or 255).

For the first two types, only an MB-type word is sent, while for the BOUNDARY MBs motion estimation is performed. Based on the sum of absolute differences (SAD) between the current MB and its best prediction, an inter/intra coding mode decision is made. Note that, for binary shapes, the SAD is just the number of alpha values that differ between the two MBs – current and prediction. In conclusion, a shape MB can be coded in one of the following four modes:

1. TRANSPARENT: no need to send any information besides the MB type – MB_TRANSPARENT.
2. OPAQUE: no need to send any information besides the MB type – MB_OPAQUE
3. INTRA: the DCC or MGCC schemes are independently applied to each shape MB.
4. INTER: a motion vector with two components is sent (one for the vertical co-ordinate and another for the horizontal co-ordinate); no residues are coded.

6.1. Intra/inter MB coding mode decision

An MB is coded in intra or inter mode depending on the SAD value for its best prediction and on the value of a control parameter, called alpha threshold (α_{thr}). This parameter indicates the shape degradation accepted when motion compensation is performed since no residues are coded. The rule for choosing between inter and intra modes is the following:

if ($SAD_{\text{best prediction}} \leqslant \alpha_{thr}$)

  mode = inter

else

  mode = intra

$\alpha_{thr} = 0$ corresponds to lossless coding while increasing α_{thr} allows to achieve higher compression rates at the expense of higher levels of controlled degradation. This also means that inter coding can be used both for the lossless (DCC and $\alpha_{thr} = 0$) and lossy (MGCC) modes. Note that in the case of selecting the lossy mode, the types of losses that are introduced in the inter mode differ from that of the intra mode. In the inter mode, losses can be larger than one pixel and may appear in the interior of the shape. This is also related to the control of the intra/inter mode by means of a global measure, i.e. the SAD value.

6.2. Shape MB motion estimation/compensation

6.2.1. MB motion estimation

Motion estimation is performed at the MB level. The comparisons are made between the current alpha MB and the displaced alpha MB in the previous reconstructed alpha plane. A full search around the original MB position is used with a maximum displacement of $[-16, +15]$ integer pixel positions. The (x, y) displacement given the smallest SAD is chosen to give the alpha MB motion vector.

6.2.2. Differential coding of alpha motion vectors

When using the inter mode coding, motion vectors have to be transmitted. The motion vector components (horizontal and vertical) are differentially coded by using a spatial neighbourhood of three motion vectors already transmitted (Fig. 12). In the case where the MB is in the border of the current alpha plane, the following rules apply depending on the number of candidate predictors outside the VOP alpha plane:

1. If the MB of one and only one candidate predictor is outside the VOP, the corresponding motion vector candidate predictor is set to zero.
2. If the MBs of two and only two candidate predictors are outside the VOP, the corresponding motion vectors candidate predictors are set to the third candidate predictor.
3. If the MBs of all candidate predictors are outside the VOP, the corresponding motion vectors candidate predictors are set to zero.

The alpha motion vector coding is performed separately on the horizontal and vertical components. For each component, the median value of the three

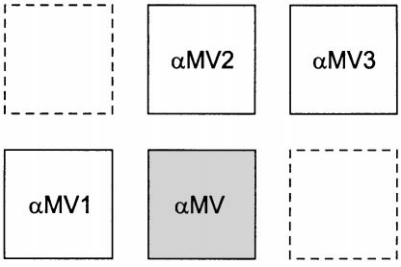

Fig. 12. MB neighbourhood for motion vector prediction.

candidates for the same component is computed:

$$P_x = \text{Median}(\alpha\text{MV1}_x, \alpha\text{MV2}_x, \alpha\text{MV3}_x),$$

$$P_y = \text{Median}(\alpha\text{MV1}_y, \alpha\text{MV2}_y, \alpha\text{MV3}_y),$$

where αMVN_x is the 'x' co-ordinate for the alpha motion vector of candidate predictor N.

The motion vector differences, given by the following expressions for each component, are then VLC encoded:

$$\alpha\text{MVD}_x = \alpha\text{MV}_x - P_x,$$

$$\alpha\text{MVD}_y = \alpha\text{MV}_y - P_y,$$

where αMVD_x is the 'x' co-ordinate for the differential alpha motion vector.

6.2.3. MB motion compensation

The decoder computes the current motion vector in the inverse way using the motion vectors from the previous decoded MBs and adding the current decoded motion vector difference as follows:

$$\alpha\text{MV}_x = \alpha\text{MVD}_x + P_x,$$

$$\alpha\text{MV}_y = \alpha\text{MVD}_y + P_y,$$

where P_x and P_y are computed in the same way as in the encoder. After obtaining the current motion vector, the decoder gets the corresponding alpha MB in the previous reconstructed alpha plane.

7. Results and conclusions

To complete the study of the shape coding methods proposed here, coding results for well-known video test sequences are presented in this section. Tables 3–5 present the average number of bits per alpha plane for the following cases:

- Two of the objects of the MPEG-4 sequence "Children", notably the "Kids" and the "Logo", in SIF format (100 frames),
- One object of the sequence "Cyclamen" in SIF format (100 frames),
- One object of the MPEG-4 sequence "Total Destruction", called "Robot", in SIF format (44 frames),
- One object of the MPEG-4 sequence "Weather", called "Girl", in QCIF format (100 frames),

Table 3
Bits/alpha plane – MB-based coding mode (intra and inter modes)

	DCC			MGCC	
	$\alpha_{thr} = 0$	$\alpha_{thr} = 4$	$\alpha_{thr} = 8$	$\alpha_{thr} = 4$	$\alpha_{thr} = 8$
Kids	4113	2791	1954	2831	1948
Logo	2401	1870	1550	2528	1611
Cyclamen	7396	4309	2733	5016	3035
Robot	5325	4364	3599	4210	3482
Girl	945	480	305	520	309
Stefan	766	682	570	656	559

Table 4
Bits/alpha plane – MB-based coding mode (only intra mode)

	DCC	MGCC
Kids	4387	4329
Logo	5595	5154
Cyclamen	9437	9508
Robot	5591	5351
Girl	1215	1243
Stefan	776	747

Table 5
Bits/alpha plane – object-based coding mode (only intra mode)

	DCC	MGCC
Kids	1859	1615
Logo	3419	3026
Cyclamen	4102	3813
Robot	3028	2669
Girl	478	446
Stefan	400	356

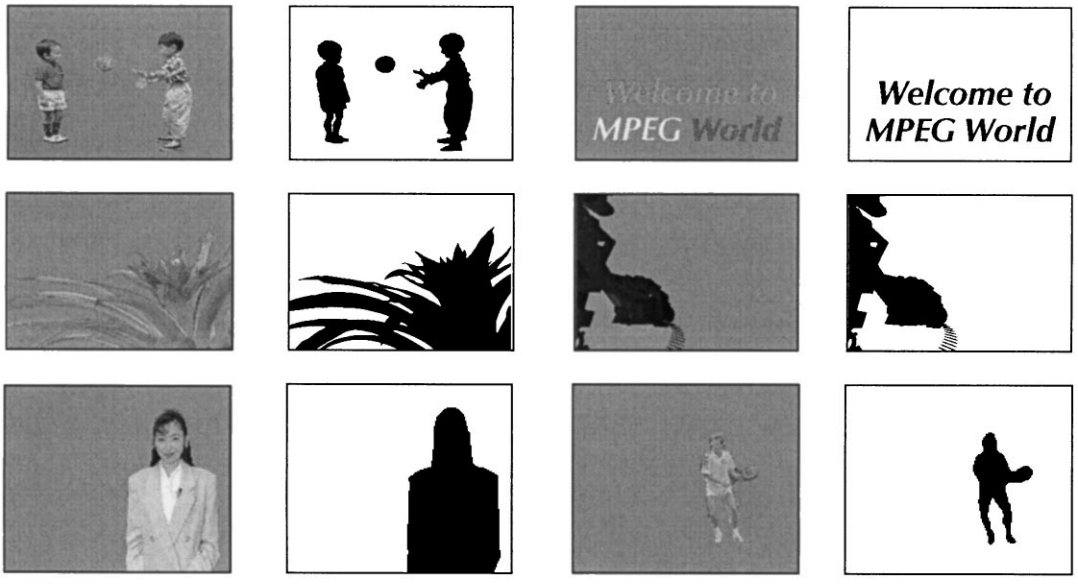

Fig. 13. Example VOPs for "Kids", "Logo", "Cyclamen", "Robot", "Girl" and "Stefan".

- One object of the MPEG-4 sequence "Stefan", in QCIF format (100 frames).

Fig. 13 shows one frame for each of these objects and the corresponding alpha planes.

Tables 3 and 4 show that the use of the inter coding mode for lossless coding (i.e. DCC with $\alpha_{thr} = 0$ versus only DCC) with the MB-based mode, allows a significant reduction in the number of bits used. This is more visible for "Logo" due to the pure translational motion of this object but all tested objects use less bits depending on the amount and type of motion that they have. Moreover, Tables 4 and 5 show the significant reduction in the number of bits achieved by the object-based mode relative to the MB-based mode, due to a much more efficient use of spatial redundancy.

In order to allow some comparison with the MPEG-4 standard shape coding solution (CAE), Table 6 presents the results for the coding of the same object shapes using the technique in the MPEG-4 Video VM 8.0 [13] as implemented in the MoMuSys VM8-971119 software implementation [1]. From the results, it may be concluded that Inter CAE and the best directly comparable technique proposed here (DCC object-based intra coding – Table 5) have similar performance in the case of the "Kids", "Girl" and "Stefan" objects; for the other objects, Inter CAE outperforms, notably for very structured objects such as the "Logo". However, for the intra coding mode MGCC outperforms CAE (see Tables 5 and 6).

Table 6
Bits/alpha plane – CAE (VM 8.0) (lossless coding)

	Inter CAE	Intra CAE
Kids	1683	2069
Logo	872	3038
Cyclamen	2355	4278
Robot	2225	2833
Girl	313	533
Stefan	388	400

Having seen the results, it is important to highlight that MGCC introduces new important features with respect to CAE. First, the types of coding losses introduced by the MGCC do not change the homotopy of the shape being coded. As reported in [17], this is a problem that has to be carefully handled in the case of CAE since CAE coding losses can lead to the creation of holes inside the decoded objects or even to the division of a single object into several objects. This issue has to be taken into account when encoding the VOP information.

Moreover, MGCC has the possibility to be extended to code complete image partitions. This extension is possible by introducing the concept of triple point in the encoding process [18].

Since the CAE technique uses tools and modes, notably arithmetic encoding and the prediction of the MB coding modes, that may improve the chain code based techniques proposed here, in the future the work will proceed by the integration of these additional tools to study the corresponding performance improvement. Moreover, an object-based coding mode using temporal prediction will also be developed.

Although shape coding has been for many years a topic of research, it was the push given by the MPEG-4 standardisation effort, and its competitive environment, that finally allowed a significant jump in terms of shape coding performance. This strong push was motivated by the key role that shape coding plays in the MPEG-4 object-based representation of video data. With shape, video objects will follow, and with video objects people will interact, giving birth to a new generation of multimedia applications.

Acknowledgements

The authors acknowledge the support of the European Commission under the ACTS project MoMuSys. This work was also supported by the "Acção Integrada Luso-Espanhola, Codificação de Vídeo para Terminais Audiovisuais Móveis". P. Nunes acknowledges PRAXIS XXI for his Ph.D. scholarship.

References

[1] ACTS MoMuSys Project, MPEG-4 video verification Model 8.0 software implementation: MoMuSys VM8-971119, November 1997.

[2] N. Brady, F. Bossen, N. Murphy, Context-based arithmetic encoding of 2D shape sequences, in: Proceedings of the 1997 IEEE International Conference on Image Processing, Santa Barbara, CA, USA, October 1997.

[3] H. Freeman, On the encoding of arbitrary geometric configurations, IRE Trans. Electron. Comput. EC(10) (June 1961) 260–268.

[4] P. Gerken, Object-based analysis-synthesis coding of image sequences at very low bit rates, IEEE Trans. Circuits Systems Video Technol., Special Issue on Very Low Bit Rate Video Coding 4 (3) (June 1994) 228–235.

[5] R. Koenen, F. Pereira, L. Chiariglione, MPEG-4: Context and objectives, Signal Processing: Image Communication 9 (4) (May 1997) 295–304.

[6] S. Lee, D.-S. Cho, S. Cho, S. Son, E. Jang, J.-S. Shin, Binary shape coding using 1-D distance values from baseline, in: Proceedings of the 1997 IEEE International Conference on Image Processing, Santa Barbara, CA, USA, October 1997.

[7] F. Marqués, A. Gasull, Partition coding using multi-grid chain code and motion compensation, in: Proceedings of the 1996 IEEE International Conference on Image Processing, Lausanne, Switzerland, September 1996, Vol. II, pp. 935–938.

[8] F. Marqués, P. Nunes, Description of the core experiment S4b: Multi-grid chain coding method, Doc. ISO/IEC JTC1/ SC29/WG11 M1326, Chicago MPEG meeting, September 1996.

[9] F. Marqués, J. Sauleda, A. Gasull, Shape and location coding for contour images, in: Proceedings of the 1993 Picture Coding Symposium, Lausanne, Switzerland, March 1993, pp. 18.6.1–18.6.2.

[10] T. Minami, K. Shinohara, Encoding of line drawings with a multiple grid chain code, IEEE Trans. Pattern Anal. and Mach. Intell. PAMI-8 (2) (March 1986) 269–276.

[11] MPEG Requirements Group, MPEG-4 requirements, Version 11, Doc. ISO/IEC JTC1/SC29/WG11 N2723, Seoul MPEG meeting, March 1999.

[12] MPEG Video Group, Core experiments on MPEG-4 video shape coding, Doc. ISO/IEC JTC1/SC29/WG11 N1471, Maceió MPEG meeting, November 1996.

[13] MPEG Video Group, MPEG-4 video verification model 8.0, Doc. ISO/IEC JTC1/SC29/WG11 N1796, Stockholm MPEG meeting, July 1997.

[14] MPEG Video and SNHC Groups, FDIS of ISO/IEC 14496-2 Part 2: Visual, Doc. ISO/IEC JTC1/SC29/WG11 N2502, Atlantic City MPEG meeting, October 1998.

[15] P. Nunes, F. Pereira, F. Marqués, Multi-grid chain coding of binary shapes, in: Proceedings of the 1997 IEEE International Conference on Image Processing, Santa Barbara, CA, USA, October 1997.

[16] J. Ostermann, Coding of binary shape in MPEG-4, in: Proceedings of the 1997 Picture Coding Symposium, Berlin, Germany, September 1997, pp. 659–662.

[17] J. Ostermann, Efficient encoding of binary shapes using MPEG-4, in: Proceedings of the 1998 IEEE International Conference on Image Processing, Chicago, IL, USA, October 1998.

[18] P. Salembier, F. Marqués, A. Gasull, Coding of partition sequences, in. L. Torres, M. Kunt (Eds.), Video Coding: The second Generation Approach, Kluwer Academic Publishers, Dordrecht, 1996, pp. 125–170.

[19] N. Yamaguchi, T. Ida, T. Watanabe, A binary shape coding method using modified MMR, in: Proceedings of the 1997 IEEE International Conference on Image Processing, Santa Barbara, CA, USA, October 1997.