MPEG-4 and Rate-Distortion Based Shape Coding Techniques

Aggelos K. Katsaggelos (contact author), Lisimachos P. Kondi

Northwestern University
Department of Electrical and Computer Engineering
McCormick School of Engineering and Applied Science
Evanston, Illinois 60208-3118
Tel: (847) 491-7164
Fax: (847) 491-4455
E-mail: {aggk,lkon}@ece.nwu.edu

Fabian W. Meier

Silicon Graphics
Advanced Entertainment Systems
Mountain View, CA, 94043-1389
Tel: (650) 933-1712
Fax: (650) 965-7651
E-mail: fabian@engr.sgi.com

Jörn Ostermann

AT&T Labs–Research
Speech and Image Processing Services Research Lab
Room NSL 3-231
100 Schultz Drive–Middletown
Red Bank, NJ 07701-7033
Tel: (732) 345-3311
Fax: (732) 345-3033
E-mail: osterman@research.att.com

and

Guido M. Schuster

3Com
Advanced Technologies Research Center
Mount Prospect, Illinois 60016
Tel: (847) 222-2486
Fax: (847) 222-0573
E-mail: gschuste@usr.com

Abstract

In this paper we address the problem of the efficient encoding of object boundaries. This problem is becoming increasingly important in applications such as content-based storage and retrieval, studio and television post-production and mobile multimedia applications. The MPEG-4 Visual Standard will allow the transmission of arbitrarily shaped video objects. The techniques developed for shape coding within the MPEG-4 standardization effort are described and compared first. A framework for the representation of shapes using their contours is presented next. Such representations are achieved using curves of various orders and they are optimal in the rate distortion sense. Finally conclusions are drawn.

1 Introduction

With video being a ubiquitous part of modern multimedia communications, new functionalities in addition to the conventional compression provided by existing video coding standards like H.261, MPEG-1, H.262, MPEG-2, and H.263 are required for new applications. Applications like digital libraries or content-based storage and retrieval have to allow 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. 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 different semantic parts of a scene and to fit the individual needs. All these applications share one common requirement: Video content has to be easily accessible on an object basis.

Given the application requirements, video objects have to be described not only by texture but also by shape. The importance of shape for video objects has been realized early on by the broadcast and movie industry by employing the so-called chroma-keying technique. Coding algorithms like object-based analysis-synthesis coding [1] use shape as a parameter in addition to texture and motion for describing moving video objects. Second-generation image coding segments an image into regions and describes each region by texture and shape [2]. The purpose of using shape was to achieve better subjective picture quality, increased coding efficiency as well as an object-based video representation.

MPEG-4 Visual will be the first international standard allowing the transmission of arbitrarily shaped video objects (VO) [3]. A frame of a VO is called video object plane (VOP). 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 and arrange them into a video scene.

This results in a new dimension of interactivity and flexibility for standardized video and multimedia applications.

After a review of object-based coders and shape coding (section 2), this paper provides an overview of the main algorithms for shape as investigated within MPEG-4 in section 3. Two types of VOs are distinguished: For opaque objects, binary shape information is transmitted. Two bitmap-based (section 3.1), two contour-based (section 3.2), and an implicit shape coder (section 3.3) are presented. The evaluation criteria for measuring coding performance and the test sequences are discussed in section 3.4. The performance of the five shape coders, verified by bitstream exchange, in terms of coding efficiency, error resilience and hardware implementation is compared in section 3.5. Transparent objects are described by gray-scale $\alpha-$ maps (8 bits/pel) defining the outline as well as the transparency of an object (section 3.6).

In section 4 contour-based coding is revisited. A framework is presented for obtaining optimal lossy encoding results in the operational rate-distortion sense (section 4.1). In section 4.2 different distortion measures which can be used in boundary encoding are described. In section 4.3, issues concerning the rate required to encode the boundary are discussed. More specifically, we discuss prediction methods to encode the control points or vertices and how this is related to the order of the dynamic programming used to solve the problem. In sections 4.5 and 4.6 we present several solution approaches to the problem followed by experimental results in section 4.6. Finally, in section 5 we present our conclusions.

2 Review of Object-based Video Coding

2.1 Concepts

Two concepts were developed that introduced shape coding into image and video coding. In 1985, a region-based image coding technique was published [2]. In order to encode an image, it is first segmented into regions of homogenous texture. In a second step, each region is encoded by transmitting its contour as well as one value for defining the luminance of the region. Rate control is achieved by controlling the number of segmented regions. The assumption is that the contours of regions are very important for subjective image quality whereas the texture of the regions is of lower importance. In [4], this concept is extended to code video. Since the available data rate is mostly used for the coding of region contours, contours of image regions are very well preserved. The coder is very well suited for coding of objects with flat textures, since any DCT-based coder has significant problems in representing sharp edges. However texture detail within a region gets lost. In case that a fairly homogenous region of the original image gets coded by more than

one region, contouring artifacts appear. These coding artifacts did not compare favorably with the block and mosquito artifacts of an H.261 coder. In subjective tests, MPEG-4 confirmed that a block-based coder compares favorably to a region-based coder [5] when encoding rectangular video sequences.

Whereas a conventional *frame-based video coder* like MPEG-1 or H.263 encodes a sequence of rectangular frames, an *object-based video coder* encodes arbitrarily shaped video object. This concept was inspired by the development of the object-based analysis-synthesis coder (OBASC) published in 1989 [1]. An OBASC divides an image sequence into arbitrarily shaped moving objects. Objects are encoded independently. An object is defined by its uniform motion and described by motion, shape, and color parameters, where color parameters denote luminance and chrominance reflectance of the object surface. The image analysis of an OBASC estimates the current motion, shape and texture parameters of each object. Furthermore, image analysis determines for which part of the object, the object does not behave according to the underlying source model and hence cannot be predicted using motion compensated prediction alone. These regions are called model failures. Parameter coding encodes the motion parameters. Using these motion parameters, motion compensated prediction is employed to increase the coding efficiency of the shape coder. Finally, the shape and texture of the model failures are coded.

The coding efficiency of OBASC mainly depends on the selection of an appropriate source model and the availability of an automatic image analysis which estimates the model parameters from the video sequence to be coded. Different source models like 2D flexible object with 2D motion, 2D rigid objects with 3D motion, 3D rigid and flexible objects with 3D motion were investigated [6, 7, 8, 9, 10]. The source models using flexible surfaces proved to be particularly successful and outperformed the H.261 coder [8, 9] for video phone test sequences at 64 kbit/s and below. For shape coding, a polygon approximation of the object was used. Lossy shape coding was used in order to save bitrate. The degree of lossiness was determined by subjective experiments. However, OBASC was mainly successful for simple video sequences due to lack of a robust image analysis. Therefore, segementation of moving objects within an object-based coder was investigated [11]. It has to be noted that the success of OBASC is due to the introduction of shape coding and the use of a motion model able to describe flexible deformation thus allowing to limit the areas of model failure to small image regions. MPEG-4 only implements a shape coder but does not allow to model flexible motion due to the use of regular block based motion compensation. This choice has two reasons: At the time of subjective testing, OBASC was not able to outperform the block-based reference coder for scenes with complex motion. Furthermore, the computational complexity of source

models allowing flexible motion [8, 9, 12, 13] is significantly higher than block based motion compensation.

Since MPEG-4 defines a video decoder, the problem of image analysis was avoided by using pre-segmented video sequences named video objects (VOs) as coder input. This decision allowed the development of an object-based video coder. Although automatic segmentation is still an open research topic, segmentation is widely used in controlled environments. TV and studio industries rely to a large extent on the chroma-keying technique that provides a reliable segmentation of objects in front of a uniform background in controlled studio environments.

2.2 2D-Shape Coding

In computer graphics, the shape of an object is defined by means of an α-map M_k of size $X \cdot Y$ pels:

$$M_k = \{m_k(x,y) | 0 \leq x \leq X,\ 0 \leq y \leq Y\},\ 0 \leq m_k \leq 255. \tag{1}$$

The shape M_k defines for each pel (x, y) whether it belongs to the VO ($m_k(x, y) > 0$) or not ($m_k(x, y) = 0$). For an opaque object, the corresponding α-values are 255, for transparent objects they range from 1 to 255 (Figure 1). Coded parameters are indicated by $()'$ like M'_k.

Almost the entire literature on shape coding deals with efficient coding of binary shapes with $m_k(x, y) = 0$ being background and $m_k(x, y) = 255$ being the object. There are two classes of binary shape coders. A *bitmap*-based coder encodes for each pel whether it belongs to the object or not. A *contour*-based coder encodes the outline of the object. In order to retrieve the bitmap of the object shape, the contour is filled with the object label. In the case that there is also texture transmitted with the shape information, an *implicit* shape coder can be used where the shape information can be derived from the texture. The already mentioned chroma-keying method would fall into this category. It is also specified in GIF89a [14]. For each image, one number can be used to define the value of the transparent pels. All pels of this value are not displayed. Today, GIF89a is used in Web applications to allow describing arbitrarily-shaped image and video objects.

Bitmap-based shape coders are used in the FAX standards G4 [15] and JBIG [16]. The Modified Read (MR) code used in the fax G4 standard, scans each line of the document and encodes the location of *changing pels* where the scanline changes its color. In this line-by-line scheme the position of each changing pel on the current line is coded with respect to either the position of a corresponding changing pel in the reference line, which lies immediately above the present line, or with respect to the preceding changing pel

in the current line [17].

Extensive work has been published on contour-based shape representation and coding. Different applications nurtured this research: For lossless and lossy encoding of object boundaries, chain coders [18, 19] and polygon approximations [10, 20, 21, 22, 23] were developed. For recognition purposes, shape representations like Fourier descriptors were developed to allow translation, rotation and scale invariant shape representations [24].

A chain code follows the contour of an object and encodes the direction in which the next boundary pel is located (Figure 2). Algorithms differ by whether they consider a pel having 4 or 8 neighbors for rectangular grids or six neighbors for hexagonal grids. Some algorithms define the object boundary between pels [25]. Freeman [18] originally proposed the use of chain coding for boundary quantization and encoding, which has attracted considerable attention over the last thirty years [26, 27, 28, 29, 30]. The curve is quantized using the grid intersection scheme [18] and the quantized curve is represented using a string of increments. Since the planar curve is assumed to be continuous, the increments between grid points are limited to the 8 grid neighbors, and hence an increment can be represented by 3 bits. For lossless encoding of boundary shapes, an average 1.2 bit/boundary pels and 1.4 bits/boundary pels is required respectively for a 4 and an 8 neighbor grid [19]. There have been many extensions to this basic scheme such as the generalized chain codes [26], where the coding efficiency has been improved by using links of different length and different angular resolution. In [29] a scheme is presented which utilizes patterns in a chain code string to increase the coding efficiency, and in [30] differential chain codes are presented, which employ the statistical dependency between successive links. There has also been interest in the theoretical performance of chain codes. In [27] the performance of different quantization schemes is compared, whereas in [28] the rate distortion characteristics of certain chain codes are studied. In this paper, we are not concerned with the quantization of the continuous curve, since we assume that the object boundaries are given with pixel accuracy. Some chain codes also include simplifications of the contour in order to increase coding efficiency [31, 32]. This is similar to filtering the object shape with morphological filters and then coding with a chain code. The entropy coder may code a combination of several directions with just one code word.

A polygon-based shape representation was developed for OBASC [8, 20]. As a quality measure, the Euclidean distances d_{max} between the original and the approximated contour is used. During subjective evaluations of CIF (352*288 pels) video sequences, it was found that allowing a peak distance of $d^*_{max} = 1.4$ pel is sufficient to allow proper representations of objects in low bitrate applications. Hence the lossy

polygon approximation was developed. The polygon approximation is computed by using those two contour points with the maximum distance between them as the starting point. Then, additional points are added to the polygon where the approximation error between the polygon and the contour are maximum (Figure 3). This is repeated until the shape approximation error is less than d^*_{max}. In a last step, splines are defined using the polygon points. If the spline approximation does not result in a larger approximation error between two neighboring polygon points, the spline approximation is used. This leads to a smoother representation of the shape (Figure 4). Vertex coordinates and the curve type between two vertices are arithmetically encoded.

This polygon/spline representation is also used for coding shapes in inter mode. For temporal prediction, the texture motion vectors are applied to the vertices defining the predicted shape. Then, all vertices within the allowable approximation error d^*_{max} define the new polygon approximation. It is refined as described above such that the entire polygon is within the allowable error d^*_{max}. In a final step, it is again decided whether a polygon or spline approximation is used. The reason for not using a complete spline approximation is due to the fact that temporal prediction using splines is less efficient, because the refinement of a predicted spline representation requires the definition of many more new vertices when compared to a polygon representation.

In [33] B-spline curves are used to approximate a boundary. An optimization procedure is formulated for finding the optimal locations of the control points by minimizing the mean squared error between the boundary and the approximation. This is an appropriate objective when the smoothing of the boundary is the main problem. When, however, the resulting control points need to be encoded, the tradeoff between the encoding cost and the resulting distortion needs to be considered. By selecting the mean squared error as the distortion measure and allowing for the location of the control points to be anywhere on the plane, the resulting optimization problem is continuous and convex and can be solved easily. In order to encode the positions of the resulting control points efficiently, however, one needs to quantize them, and therefore the optimality of the solution is lost. It is well known that the optimal solution to a discrete optimization problem (quantized locations) does not have to be close to the solution of the corresponding continuous problem.

The above methods for polygon/spline representation achieve good results but they do not claim optimality. In section 4, we describe polygon/spline representation approaches which provide optimality in the operational rate distortion sense.

Fourier descriptors were developed for applications in recognition, where shape is an important key. Fourier descriptors allow a translation, rotation and scale invariant representation [34]. In a first step, the coordinates of the contour are sampled clockwise in the xy-plane. This list of 2D coordinates (x_i, y_i) is transformed into an ordered list $(i, (y_{i+1} - y_i)/(x_{i+1} - x_i))$ with $0 \leq i \leq i + 1$ being the contour point number and $(y_{i+1} - y_i)/(x_{i+1} - x_i)$ the change of direction of the contour. Since the samples are periodic over the object boundary perimeter, they can be expanded into a Fourier series. In order to preserve the main characteristics of a shape, only the large Fourier coefficients have to be maintained. Fourier descriptors are not very efficient in reconstructing polygon-like shapes with only a few coefficients. This is one of the reasons, why they never became very competitive in coding efficiency.

3 Shape coding in MPEG-4

The goal of shape coding is to encode the shape information of a moving video object in order to enable applications requiring content based video access (Figure 1). It is assumed that texture and motion information is transmitted for the video object to efficiently code its texture. In this section, bitmap-based, contour-based and implicit shape coders developed within MPEG-4 for coding of binary shapes are presented. The evaluation leading to the selection of a binary shape coder is also reviewed. In a last section, the coding of gray-scale $\alpha-$ maps is described.

3.1 Binary bitmap-based shape coder

In the following sections, two bitmap-based shape coders encoding the shape information on a macroblock basis are described. The first coder uses a non-adaptive context-based arithmetic encoder for encoding the shape information and motion compensation for exploiting temporal redundancies. The second coder is based on an adaptation of the Modified Read coder that is used in place of the arithmetic encoder. Other aspects of the algorithms are identical.

3.1.1 Context-based (CAE) shape coder

Within a macroblock, 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 [35]. Then, this technique is extended to include an inter mode [35, 36].

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 5a). 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, the arithmetic coder stuffs one '1' into the bitstream whenever a long sequence of '0's 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 5a). 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 two 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 and for rate control purposes.

Depending on the upsampling filter, the decoded shape can look somewhat blocky. Several upsampling filters were investigated. The two best performing filters were a simple pel replication filter combined with a 3*3 median filter and an adaptive non-linear upsampling filter. The context of this upsampling filter as standardized by MPEG-4 is shown in Figure 6.

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.

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 previously coded 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 for prediction, 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 5b). 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 frame. Assuming that the motion vector $(d_x, d_y)^T$ points from the current M_k to the previous coded M'_{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.

3.1.2 Modified Modified Read shape coder

The Modified Modified Read (MMR) shape coder is a macroblock-based shape coder [36]. In comparison to the CAE shape coder, the MMR shape coder mainly replaces the context-based arithmetic encoder and the templates by a MMR coder. Therefore, the description of the MMR shape coder is limited to the MMR coder. This MMR coder is derived from the Modified Read Coder in the FAX G4 standard [15].

Figure 7 is used for describing the encoding procedure. For simplicity it is assumed that the macroblock to be coded is subsampled by a factor of 2 resulting in a sub-block of size 8*8 pels to be coded. Each block is coded in raster-scan order. Within each line, the position of pels on the object boundary is encoded. In

Figure 7, it is assumed that the top 5 rows of the block have already been coded, hence the coder knows the position of the pels $a0$ and $b1$ in the current block as well as pels $c0$ and $c1$ in the motion compensated block when coding in intra and inter mode respectively.

Intra Mode

The unknown point $a1$ on the object boundary is encoded with reference to the two pels $a0$, and $b1$. $a0$ is the last changing pel encoded prior to $a1$. $b1$ is the first changing pel on the line above $a0$ to the right of $a0$ and with the opposite color of $a0$, if such point exists (Figure 7). If not, then $b1$ is the leftmost changing pel on the same line as $a0$. In order to encode the distance between $a1$ and $a0$, one of the three modes, Vertical, Horizontal or Vertical Pass, is selected. Assuming that all pels are numbered in raster-scan order starting with 0 in the top-left corner of the block, i.e. in Figure 7 $num(a0) = 35$, and columns c are numbered from left to right, i.e. $c(a0) = 2$, a mode is selected according to

$$mode = \begin{cases} \text{Vertical} & \text{if } |c(a1) - c(b1)| \leq T \\ \text{Horizontal} & \text{if } num(a1) - num(a0) < width \\ \text{Vertical Pass} & \text{otherwise} \end{cases} \quad (2)$$

with the threshold $T = 5$ for no subsampling, $T = 3$ for a subsampling factor of 2, $T = 2$ for a subsampling factor of 4, and $width$ the width of the block to be coded.

In Vertical Mode, the distance $c(a1) - c(b1)$ is encoded using one of eight VLC tables that is selected according to the object boundary direction as defined by a template positioned above pel $b1$ (Figure 8).

In Horizontal Mode, the position of $a1$ is encoded as its distance to $a0$. Just due to the fact that the horizontal and not the vertical mode is selected, the decoder can sometimes deduct the minimum distance between $a1$ and $a0$. In this case, only the difference with respect to this minimum distance is encoded.

In Vertical Pass Mode, one code word is sent for each line without an object boundary. One last code word codes the remaining distance to the next point on the object boundary. $a1$ in Figure 7 (subsampling factor 2) is encoded using vertical pass mode.

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 MMR coding. In both cases the encoder also has the choice of scanning each line from left to right or vice versa.

These adaptations of MMR to macroblock-based shape coding increased the performance by 30-70% compared to the MMR of the FAX G4 standard.

Inter Mode

In Inter Mode, the previously decoded shape is motion compensated as described in section 3.1.1. Figure 7b shows the motion compensated shape information. A pel $c1$ is defined as the first changing pel with a color opposite to the color of $a0$ in the same row as $a0$, if such a pel exists. If not, $c1$ is the first changing pel in the remaining lines of the block. The mode is selected according to Eq. 2 with $b1$ replaced by $c1$ (Figure 7). The codes for transmitting the distance in Vertical mode are not switched by the template but by the mode with which the previous boundary pel was coded.

3.2 Binary contour-based shape Coder

Within MPEG-4, two methods were developed: A vertex-based polynomial shape approximation based on work in [10, 20, 23] and a baseline-based shape coder [37].

3.2.1 Vertex-based shape coding

Vertex-based shape coding codes the outline of the shape. This shape is approximated using a polygon approximation for lossy shape coding. The placement of vertices allows an easy control of local variations of the shape approximation error. For lossless shape coding, the polygon approximation "degenerates" to a chain code [19, 31, 38, 39]. In the following, the encoding of shapes in intra mode is described first. In a second step, the algorithm is extended to exploit temporal redundancy.

Intra Mode

The efficiency of this shape coding method depends to a large extent on the encoder. The art of lossy vertex-based shape coding lies in selecting the appropriate vertices for the polygons. The approach chosen by the experimenters [10, 23, 40] starts with finding the longest axis of the shape and uses the two endpoints as the initial polygon. Let us assume that $\overline{V_k V_{k+1}}$ approximates the original contour between the vertices V_k and V_{k+1}. The original contour segment associated with this approximation $\overline{V_k V_{k+1}}$ segment is called C_k with the contour points $c_{k,i}$ with k identifying the segment, and I the number of contour points of C_k. With $d(\overline{V_k V_{k+1}}, c_{k,i})$ being the Euclidean distance between contour point $c_{k,i}$ and the line $\overline{V_k V_{k+1}}$, the approximation error for a segment C_k of the original contour is given by $d_{max}(k)$:

$$d_{max}(k) = \max_{0 \le i < I} d(\overline{V_k V_{k+1}}, c_{k,i}). \tag{3}$$

For each side of the polygon, it is checked whether the approximation lies within a given tolerance $d_{max}(k) < d^*_{max}$ (Figure 3). If not, a new vertex is inserted at the point of the largest approximation error. Then, for each new polygon side, it is decided whether it lies within the allowable shape approximation and the process is repeated until the peak approximation error is lower than d^*_{max}.

The described vertex selection method selects all vertices on the object boundary. However, for lossy shape coding this might not be optimal. Therefore, the vertices can be shifted by 1 pel within an 8-pel-neighbourhood (Figure 9). The process can be repeated until an optimum approximation given the number of initially selected vertices is reached. This process minimizes the average shape approximation error $\overline{d}_k(a,b)$ for two neighbouring contour segments C_{k-1} and C_k with contour points $c_{k-1,j}$ and $c_{k,i}$, respectively. Given a vertex $V_k(a,b)$ shifted by (a,b) with $-1 \leq a,b \leq 1$, the average shape approximation error is given by

$$\overline{d}_k(a,b) = \frac{\sum_{j=0}^{J-1} d(\overline{V_{k-1}, V_k(a,b)}, c_{k-1,j}) + \sum_{i=0}^{I-1} d(\overline{V_k(a,b), V_{k+1}}, c_{k,i})}{I+J}, \quad -1 \leq a,b \leq 1 \tag{4}$$

with I and J the number of contour points of the segments C_k and C_{k-1}, respectively. Vertex V_k is shifted to the position that resulted in the smallest $\overline{d}_k(a,b)$. In case this shift results in the shifted V_k being equal to V_{k-1} or V_{k+1}, V_k is deleted. After this vertex selection and adjustment, the N vertex positions are encoded.

Vertices are located counter clockwise around the object. In order to save bits, the vertices are renumbered such that the largest difference in x- or y-coordinates appears between vertex V_0 and V_{N-1}. After encoding the position of V_0, each remaining vertex position is encoded differentially to its predecessor. For lossless coding, a differential chain code is used (Figure 2). For lossy coding, the difference vector $V_d = V_k - V_{k-1}$ is computed. Similar to the relative direction used for encoding of lossless shape, relative directions are used in the lossy case with the exception that the eight directions are replaced by 8 octants in the 2D plane. The octant in which the next vertex is located is defined by V_d and transmitted to the receiver (Figure 10). The octant defines whether the x or the y coordinate of V_d is larger. The ranges of the major and the minor component of V_d are transmitted and finally the values of the two components are coded using VLC tables that are selected according to the range of the mayor and minor component. The decoder reconstructs the vertex positions, creates the polygon and then fills the interior of the polygon with the opaque label of the $\alpha-$ map.

In section 4 we present a framework for the development of vertex-based shape coding algorithms

which are optimal in the rate-distortion sense. That is, if polygons are used for approximating the original boundary, their vertices, which may belong on the boundary or lie outside of it, are chosen in such a way the distortion is minimized while the rate for encoding the location of the vertices satisfies a given bit budget. In other words, the selection of the vertices and their encoding is done simultaneously and optimally. Various ways to define the segment distortion and the total distortion are applicable, also including the definitions used above. The framework accepts curves of any order, such as B-splines.

Inter Mode

In order to exploit temporal redundancy, one motion vector is estimated for each contour. The vector is estimated such that the number of overlapping pels between the current shape M_k and the motion compensated previous shape M'_{k-1} is maximum. The motion vector can compensate for object motion of up to 24 pels in horizontal and vertical direction.

Figure 11 shows the previously coded shape M'_{k-1} and the current shape M_k. The gray area shows the overlapping area of the two shapes after motion compensation. The bottom of Figure 11 shows those contour segments that are aligned after motion compensation and those that are unmatched.

In case of lossless shape coding, the encoder transmits the positions of the unmatched segments and refines the shape approximation for these segments as described in the intra mode.

In the case of lossy shape coding, the encoder checks where the unmatched segments leave the band around the current shape that is defined by the allowable shape approximation error d^*_{max} (Figure 12). Now, unmatched segments are those that are located outside of the band. As for lossless shape coding, the encoder transmits the positions of the unmatched segments and then refines the shape approximation for these segments as described in the intra mode.

It appears that this motion compensation can still be improved. Localized motion compensation allowing one vector for each vertex, or segment as defined by a boundary macroblock should result in further improvements of this technique.

3.2.2 Baseline-based shape coding

A baseline shape coder also encodes the contour of an object. It places the shape into a 2D coordinate system such that the projection of the shape onto the x-axis is longest [37]. The x-axis is called baseline, from which the distance (y-coordinate) between the baseline and a point on the shape outline is measured.

The shape contour is sampled clockwise. Neighboring contour points usually have increasing or decreasing x-coordinates. Those contour points where the direction changes, are called turning points. They are signaled to the decoder (Figure 13). The contour is subdivided into segments of length 16 pels.

Intra Mode

The contour points within one segment can be subsampled by factors of 2, 4, or 8. For upsampling, the missing contour points are linearly interpolated from the y-values of the samples. In order to minimize the approximation error, coded contour points can be shifted vertically by ± 1 pel. A contour segment is subsampled, if the approximation error due to the subsampling is within given limits. For each segment, the y-coordinates are differentially encoded.

Inter Mode

The Baseline-based shape coder allows for global and local motion compensation. Global motion compensation aligns the previously coded shape M'_{k-1} with the current shape M_k such that the number of overlapping pels is maximized. In a second step, a 2D motion vector is searched for each segment of the current contour such that the number of misaligned pels is minimum. If the motion compensated prediction error is above a threshold, the prediction error is encoded the same way as contour points are encoded in intra mode.

3.3 Binary chroma-key shape coder

This shape coding technique [41] was inspired from the blue screen technique used in film and TV studios. The color of a pel is used to distinguish between object and background. The object to be coded is placed on a static one-colored background. The color of the background (chroma-key) has to be outside of the color space occupied by the texture of the object. Usually, highly saturated colors fulfill this requirement. The image or sequence of images with the object in front of this one-colored background is then encoded using a conventional coder (here MPEG-4 video in full frame mode). To the decoder, the chroma-key is transmitted. The decoder decodes the images. For each pel, the weighted distance d_K between the chroma-key (K_Y, K_{CB}, K_{CR}) and the corresponding values (X'_Y, X'_{CB}, X'_{CR}) of the decoded pel is computed according to:

$$d_K = W_Y|X'_Y - K_Y| + W_{CB}|X'_{CB} - K_{CB}| + W_{CR}|X'_{XR} - K_{XR}|, \tag{5}$$

where W_Y, W_{CB}, W_{CR} are the weighting factors. For lossless coding, d_K is zero for the background (chroma-key) and non-zero for the pels of the object.

When a sequence is coded using quantization, d_K becomes different than zero also for the background. In order to allow a good separation of the object from the background, the chroma-key should be chosen such that the distance d_K becomes large for all pels of the object. Pels of the decoded images with a color similar to the chroma-key ($dk < threshold$) are considered to be background. In order to extract the object shape, the decoder considers all pels of the background as transparent. The other pels belong to the object [41]. In experiments, it was found that it is not necessary to consider the luminance of the background for the object/background separation. For a chroma-key of (128,220,100), weights of (0,1,1) are appropriate. This segmentation of object and background might become erroneous when very coarse quantization of the images is allowed.

The color of pels at the object boundary inside the object is influenced by the chroma-key because of the quantization of the DCT coefficients. In order to reduce this color bleeding effect, the color can be shifted away from the chroma-key towards the color space occupied by the object.

The implicit shape coding does not allow the extraction of the object shape without decoding the entire frame. Since the shape information is typically carried by the subsampled chroma signal, this technique is not suited for lossless shape coding. Because the shape information is embedded in the texture, the shape coding is lossy as long as there is quantization of the texture. An important advantage of this method is its low computational and algorithmic complexity. The low computational complexity becomes especially apparent if the location of boundary macroblocks is signaled to the decoder such that the shape extraction has to be performed only on this small number of boundary macroblocks.

3.4 Evaluation criteria for coding efficiency

In order to compare the performance of different shape coders, evaluation criteria have to be defined. Within MPEG-4, there are 2 quality measures for objectively assessing the quality of coded shape parameters. One is the maximum of the minimal Euclidean distance d^*_{max} (peak deviation) between each coded contour point and the closest contour point on the original contour as described in section 2.2. This measure allows for an easy interpretation of the shape quality. However, if lossy shape coding results in changing the topology of an object due to opening, closing or connecting holes, the peak deviation d^*_{max} is not a useful measure. Therefore, we used a second measure d_n that is the number of erroneously represented pels of the

coded shape divided by the total number of pels belonging to the original shape. Since different objects can have very different ratios of contour pels to interior pels, a given value for d_n only allows to compare with other d_n of different approximations of the same video object. The measure d_n by itself does not provide sufficient information about the shape quality.

Subjective evaluation of several sequences indicated that a shape representation with an approximation error of $d^*_{max} > 3$ pels is not useful at all for video. All the test sequences were encoded with $d^*_{max} = 3$ pel and the corresponding d^*_n for the sequences was computed. In the following, each sequence was lossily encoded with d_n between 0 and d^*_n.

In order to get reliable results, seven test sequences (Cyclamen, Weather (woman), Kids from sequence Children, Logo from sequence Children, Robot from sequence Destruction, Rain Drops from sequence Destruction, Speakers from test sequence News) with natural and synthetic content ranging from simple scenes like a news speaker to scenes with deforming objects and camera motion were used as the test set (Figure 14). The sequence weather shows a person explaining the current weather situation along the Japanese Sea. The lady turns and partially leaves the picture. The segmentation of the person was obtained using chroma-key studio equipment. The sequence Kids shows two children playing ball. Again, the sequences was recorded in a studio and the segmentation of the persons and the ball was obtained using chroma-key equipment. The sequence Robot was generated using animation software. The robot is moving quickly, the ammunition belt consisting of many little pieces moving fast.

It was found that the objective measures truthfully reflected subjective quality when comparing different bitmap-based shape coders or when comparing different contour-based shape coders. For lossy shape coding, the bitmap-based shape coders create blocky object shapes whereas contour-based shape coders create an object shape showing polygon edges. Since the two classes of shape coders gave different distortions (Figure 15), a comparison between these 2 types had to be done subjectively. Decoded video objects (lossy shape coded at a given average bitrate for shape; texture coded using a quantizer step size of 12) were displayed on TV monitors and informally evaluated by approximately 30 experts.

3.5 Comparison of binary shape coders

The above described shape coding algorithms were thoroughly investigated with respect to their coding efficiency, subjective quality for lossy shape coding, hardware and software complexity and their performance in scalable shape coders. Coding efficiency was compared in rate distortion diagrams, subjective

quality measured using long sessions to compare coded video sequences, and software complexity was measured using tools developed within the MPEG-4 implementation study group, which count the number of operations and measure the memory bandwidth that a shape coder required on a Ultra-SPARC processor [42, 43].

Error Resilience

MPEG-4 wants to enable multimedia communications over different networks like satellite and terrestrial broadcast, telephone networks, Internet, and wireless communication links. Although each communication channel has a network interface that provides a certain protection from errors, it is expected that wireless communication channels will contain a significant amount of residual errors after channel error detection and correction. In order to have video coding algorithms cope with the corrupted bitstream, MPEG-4 video set up a group to develop an error resilient mode of the video coder. The work of that group focused mainly on the frame-based video coder and low bit rates between 24 kbit/s and 48 kbit/s. The bit error patterns used for developing an error resilient decoder include random bit errors at a rate of 10^{-3}, burst errors of 1 ms to 20 ms length with an average bit error rate between 10^{-3} and 10^{-2}, as well as loss of packets with 96 to 400 bits.

In order to ensure that the selected shape coder will also be of use in error prone environments, a group of error resilience experts evaluated the five proposals over a period of 2 months [44, 45]. In order to do error resilient shape coding, the decoder should be able to provide these four capabilities:

1. Error detection: This is the most fundamental requirement. The decoder can encounter a syntactic error, i.e., an illegal code word of variable or fixed length, or a semantic error, i.e., decoding a shape that does not close. The error may not be detected until some point after it actually occurs.

2. Error localization: After an error has been detected, the decoder has to resynchronize with the bitstream without skipping too many bits. One way of achieving this is the introduction of additional resynchronization markers [46].

3. Data recovery: After error localization, data recovery tries to recover some information from the bitstream between the location of the detected error and the determined resynchronization point. Thus, data recovery minimizes information loss. Reversible VLC (VLCs that can be decoded forward and backward) can be of help. In the case of shape coding the usefulness of such a feature has yet

to be demonstrated.

4. Error concealment: Finally, error concealment tries to hide the effects of the erroneous bitstream by replacing the lost shape information by meaningful data, i.e., copying shape data from the previous frame M'_{k-1} into the current frame. The smaller the spatial extent of the error, the more accurate error concealment can be achieved.

Of general concern was that arithmetic decoding itself does not provide any syntactic error detection capabilities. The same is true for VLC tables unless they were designed to be incomplete. The equivalence of incomplete VLC tables would be the insertion of marker bits into the arithmetically coded data. After a given runlength of '0' or '1' symbols, the decoder inserts a marker bit of opposite value. If the decoder cannot detect the marker bits at the right position, it can detect the error. Incomplete VLCs as well as marker bits decrease the efficiency of the coder.

A list of priorities were assigned to different parts of the bitstream in order to help focus on the critical issues of error resilience:

1. Shape mode (most important)
2. Shape motion vectors
3. Texture motion vectors
4. Shape data
5. Texture Data (least important)

Errors in the decoded shape can change the number of macroblocks within a VOP. This would bring the update of the texture and shape information out of alignment with respect to the previously decoded VOP. In order to limit the damage that an erroneous shape can induce to the picture quality, it is important to localize the error in the shape. This is a prerequisite for error concealment. The block-based shape coders propose that macroblock shape is decoded independently of neighboring blocks. A contour-based shape coder adds resilience by protecting the sensitive data. For the vertex-based shape coder, a syntax was proposed that allowed the shape of a VO to be encoded in independent slices of macroblocks. By coding the starting point of the contour approximation twice, i.e. as starting point and as end point of

the contour, contour-based shape coders provide additional semantic error detection capabilities since the coded contours have to be closed.

The experts concluded that all the proposed schemes would be able to provide sufficient error resilience capabilities for transmission of VO over wireless channels. Differences in terms of efficiency would be expected, but without experiments none of the proposed coders seemed to be better suited than another.

Coding efficiency

The proposed shape coders were evaluated on several sequences in intra and inter mode. Figures 16, 17 and 18 compare the four explicit shape coders when coding shapes in intra mode. The bitrate and distortion are averaged over 100 frames. As can be seen, the baseline-based shape coder provides the highest coding efficiency using between 0% and 30% fewer bits than the bitmap-based shape coders. The vertex-based shape coder comes second. The two bitmap-based shape coders perform very similarly. The CAE shape coder is slightly more efficient for lossless shape coding compared to the MMR shape coder.

In inter mode, the comparison of the different shape coders gave a different ranking. Figures 19, 20 and 21 show bitrate and distortion averages for three test sequences. For lossless and subjectively lossless coding, the bitmap-based shape coder outperformed the contour-based shape coder by up to 20%. Again, CAE shape coding is more efficient than MMR shape coding. For larger distortions, the vertex-based shape coder performs similarly to or better than the bitmap-based methods. The motion- compensation employed in the baseline method does not perform as well as the motion compensation of the other methods, causing this coder to perform worst in terms of coding efficiency in inter mode.

When comparing the different shape coding techniques, chroma-keying was not considered as a candidate for MPEG-4 shape coding, because for complex shapes the topology of the shape was not stable enough and color bleeding was visible along some boundaries. Figures 22 and 23 show a comparison between the chroma key shape coder based on Video Verification Model VM 5 and the video Verification Model VM 5 using its explicit shape coder. Since chroma-keying codes the shape in the chroma signal, the following measure SNR_{YUV} for objective picture quality was used:

$$\begin{aligned} mse_{yuv} &= \frac{\sum_0^{Y_{num}}(Y'-Y)^2 + \sum_0^{U_{num}}(U'-U)^2 + \sum_0^{V_{num}}(V'-V)^2}{Y_{num}+U_{num}+V_{num}} \\ SNR_{YUV} &= 10\log_{10}(\frac{255^2}{mse_{yuv}}) \end{aligned} \tag{6}$$

with Y_{num}, U_{num}, and V_{num} the number of pels of the Y,U, and V component that belong to the object.

VM 5 uses a simplified version of the modified MMR shape coder presented in section 3.1.2 [47]. Both coders encode texture, motion and shape. They use the same texture motion estimation algorithm and the same quantizer step sizes of 8, 12, and 20 for the quantization of DCT coefficients. Whereas VM 5 codes the shape losslessly, the chroma- key shape coder was not able to do so, for the higher quantizer, the shape coding error was significant. When comparing the bit rates, VM 5 with its explicit MMR shape coder is always more efficient than the implicit chroma-key shape coder at the price of the higher complexity of the explicit shape coder.

After evaluation of the objective shape coding performance, the main decision to be taken was whether a bitmap-based shape coder or a contour-based shape coder should be selected for the MPEG-4 standard. In order to ease this decision, the group working on shape coding first selected the best contour-based shape coder and the best block-based shape coder.

As far as bitmap-based shape coding is concerned, the CAE shape coder outperformed the MMR shape coder. For the contour-based shape coding, the vertex-based coder was chosen. It outperformed the baseline-based coder in coding efficiency for inter mode as well as in terms of computational complexity. Furthermore, the baseline coder was implemented only once, whereas all the other shape coders had two independent implementations.

Hardware Implementation

The Implementation Studies Group of MPEG-4 evaluated the vertex-based and the context-based shape coders with respect to hardware implementation implications. It is assumed that most of the computations will be done using programmable logic like video signal processors.

When comparing a VLC decoder as used for the vertex-based methods with the arithmetic decoder used in the context-based shape coder, an arithmetic decoder was found more difficult to implement and requires more chip surface.

However, the real bottle neck of a video decoder will be the required bandwidth for off-chip memory access and caching [48]. Here, the block-based context-based shape coder has the advantage that the entire shape block can be loaded into on-chip cache and processed. At the end of decoding, the decoded block can be written to memory. Since blocks are processed in predefined order, address generation for memory access is straight forward. The vertex-based shape coder does not allow for predefined memory access since no prior knowledge is available as to how the polygon will extend from one vertex to the next. Hence cache

prefetching will not work efficiently and loading the entire shape image into on-chip cache is not an option due to the size limitation of the cache. A further disadvantage of the vertex-based shape coder is that the decoding time depends on the number of vertices and the shape of the object. This is in contrast to the fixed processing time required for context-based shape coding allowing for easier task scheduling on the processor.

Summary

As far as shape coding requirements are concerned, all explicit shape coders are able to provide lossless, subjectively lossless, and lossy shape coding. The algorithms can be extended to scalable shape coders, bitstream editing, shape only decoding, low-delay applications as well as applications using noisy transmission channels. Table 1 compares the context-based and the vertex-based shape coders as done at the Bristol MPEG meeting in April 97. None of the algorithms clearly outperforms the other. However, it was felt that the simple hardware implementation of the context-based shape coder was a reason to reject the vertex-based shape coder. After that meeting, the competitive phase of shape coding ended and MPEG-4 focused on optimizing the selected context-based shape coder mainly by developing the adaptive upsampling filter (Figure 6) and an error resilient shape coding mode.

3.6 Gray-scale shape coder

Gray-scale alpha maps allow 8 bits for each luminance pel to define the transparency of that pel. Transparency is an important tool for composing objects into scenes and special effects (Figure 1). Two types of transparencies are distinguished: Binary alpha maps with objects of constant transparency and arbitrary alpha maps for objects with varying transparency.

3.7 Objects with constant transparency

For a transparent object that does not have a varying transparency, the shape is encoded using the binary shape coder and the 8 bit value of the alpha map. In order to avoid aliasing, gray-scale alpha maps usually have lower transparency values at the boundary. Blending the alpha map near the object boundary can be supported by transmitting the coefficients of a 3*3 pel FIR-filter that is applied to the alpha map within a stripe on the inner object boundary. The stripe can be up to 3 pels wide.

3.8 Objects with arbitrary transparency

For arbitrary alpha maps, shape coding is done in two steps [49]. In the first step, the outline of the object is losslessly encoded as a binary shape. In the second step, the actual alpha map is treated like the luminance of an object with binary shape and coded using the MPEG-4 texture coding tools padding, motion compensation and DCT.

4 Optimal contour encoding in the rate distortion sense

4.1 Introduction

We now present a framework for contour encoding that is optimal in the rate distortion sense. We formulate the problem in various ways. The contour approximation can be done using a polygon, B-splines or higher order curves. In all cases, the problem reduces to finding the shortest path in a Directed Acyclic Graph (DAG).

Before continuing, it is worthwhile to present a review of B-splines. These are a family of parametric curves which have proven to be very useful in boundary encoding [50, 51, 52]. In the following discussion, we will concentrate on second order B-splines. However, this theory can be generalized to higher order curves. Also, it should be noted that first order B-splines are equivalent to polygons.

A B-spline is a specific curve type from the family of parametric curves [53]. A parametric curve consists of one or more curve segments. Each curve segment is defined by $(n+1)$ *control points* where n defines the degree of the curve. The control points are located around the curve segment and together with a constant base matrix M the control points solely define the shape of the curve. A two dimensional curve segment Q_u with control points (p_{u-1}, p_u, p_{u+1}) is defined as follows:

$$Q_u(p_{u-1}, p_u, p_{u+1}, t) = \left[\begin{array}{cc} x(t), & y(t) \end{array} \right], \text{ for } 0 \leq t \leq 1 \text{ and 0 otherwise.} \tag{7}$$

The points at the beginning and the end of a curve segment are called knots and can be found by setting $t = 0$ and $t = 1$. The following is the definition for a second degree curve segment, with u as index for the different curve segments, and $p_{u,x}$ and $p_{u,y}$, respectively, as the horizontal and vertical coordinates of control point p_u,

$$\begin{aligned} & Q_u(p_{u-1}, p_u, p_{u+1}, t) = T \cdot M \cdot P \\ & = \left[\begin{array}{ccc} t^2 & t & 1 \end{array} \right] \cdot \left[\begin{array}{ccc} m_{1,1} & m_{1,2} & m_{1,3} \\ m_{2,1} & m_{2,2} & m_{2,3} \\ m_{3,1} & m_{3,2} & m_{3,3} \end{array} \right] \cdot \left[\begin{array}{cc} p_{u-1,x} & p_{u-1,y} \\ p_{u,x} & p_{u,y} \\ p_{u+1,x} & p_{u+1,y} \end{array} \right], \end{aligned} \tag{8}$$

Both, the base matrix M, with specific constant parameters for each specific type of parametric curve, and the control point matrix P, with $(n+1)$ control points, define the shape of Q_u in a two dimensional plane. Every point of the curve segment can be calculated with Eq. (8) by letting t vary from 0 to 1. Every curve segment can be calculated independently in order to calculate the entire curve Q, consisting of N_P curve segments, which is of the following form,

$$Q(t') = \sum_{u=1}^{N_P} Q_u(p_{u-1}, p_u, p_{u+1}, t' - u + 1), \text{ with } 0 \leq t' \leq N_P + 1. \tag{9}$$

Among common parametric curves are the Bezier curve and the B-spline curve. For the proposed shape coding method in this section we chose a second order (quadratic) basis uniform non-rational B-spline curve [53] with the following base matrix,

$$M = \begin{bmatrix} 0.5 & -1.0 & 0.5 \\ -1.0 & 1.0 & 0.0 \\ 0.5 & 0.5 & 0.0 \end{bmatrix}. \tag{10}$$

Figure 24 shows such a second order B-spline curve. The shape coding method presented in this section is independent of the matrix M and degree n, that is, parametric curves of higher order can be used.

Double Control Points: The beginning and the end of the boundary approximation have to be treated as special cases, if the first curve segment should start exactly from the first boundary point and the last curve segment should end exactly at the last boundary point. When we use a double control point (such as $p_{u-1} = p_u$) the curve segment Q_u will begin exactly from the double control point (see Figure 24). We apply this property to the beginning and to the end of the curve, so that $p_0 = p_1$ and $p_{N_P} = p_{N_P+1}$. These two special cases can easily be incorporated into the boundary approximation algorithm.

4.2 Distortion

In order to motivate the distortion measures presented here, we mathematically formulate an example where the boundary approximation is done using polygons and the vertices of the polygons must lie on the original boundary. We change the notation used earlier in the paper for convenience purposes. Let $B = \{b_0, \ldots, b_{N_B-1}\}$ denote the connected boundary which is an ordered set, where b_j is the j-th point of B and N_B is the total number of points in B. Note that in the case of a closed boundary, $b_0 = b_{N_B-1}$. Let $P = \{p_0, \ldots, p_{N_P-1}\}$ denote the polygon used to approximate B, which is also an ordered set, with p_k the k-th vertex of P, N_P the total number of vertices in P and the k-th segment starts at p_{k-1} and ends at p_k. Since P is an ordered set, the ordering rule and the set of vertices uniquely define the polygon. We will elaborate on the fact that the polygon is an ordered set later on.

In general the polygon which is used to approximate the boundary could be permitted to place its vertices anywhere on the plane. In this example, as mentioned earlier, we restrict the vertices to belong to the original boundary. Let $A = \{a_1, a_2, a_3, \ldots\}$ be the set of points that are admissible as vertices. Clearly, in this case, $A = B$.

The k-th polygon segment which connects two consecutive vertices, p_{k-1} and p_k, is an approximation to the partial boundary $\{b_j = p_{k-1}, b_{j+1}, \ldots, b_{j+l} = p_k\}$, which contains $l+1$ boundary points. Therefore, we can measure the quality of this approximation by a segment distortion measure which we denote by $d(p_{k-1}, p_k)$. The polygon distortion measure can then be expressed as the sum or the maximum of all segment distortion measures.

There are several different distortion measures which can be employed. One popular distortion measure for curve approximations is the maximum absolute distance, which has also been employed in [20, 27, 28, 54].

Besides its perceptual relevance, this distortion measure has the advantage that it can be computed efficiently. Let $d'(p_{k-1}, p_k, t)$ be the shortest distance between the line which goes through p_{k-1} and p_k and an arbitrary point t. This distance can be expressed as follows,

$$d'(p_{k-1}, p_k, t) = \frac{|(t_x - p_{k-1,x}) \cdot (p_{k,y} - p_{k-1,y}) - (t_y - p_{k-1,y}) \cdot (p_{k,x} - p_{k-1,x})|}{\sqrt{(p_{k,x} - p_{k-1,x})^2 + (p_{k,y} - p_{k-1,y})^2}}, \tag{11}$$

where the subscripts x and y indicate the x and y coordinates of a particular point.

Then the maximum absolute distance between the partial boundary $\{b_j = p_{k-1}, b_{j+1}, \ldots, b_{j+l} = p_k\}$ and the segment (p_{k-1}, p_k) is given by,

$$d(p_{k-1}, p_k) = \max_{t \in \{b_j = p_{k-1}, b_{j+1}, \ldots, b_{j+l} = p_k\}} d'(p_{k-1}, p_k, t). \tag{12}$$

An example is shown in Fig. 24. Another popular distortion measure is the mean squared distance (error), which has been used in [33, 55] and is of the following form,

$$d(p_{k-1}, p_k) = \frac{1}{N_B} \sum_{t \in \{b_j = p_{k-1}, b_{j+1}, \ldots, b_{j+l} = p_k\}} d'(p_{k-1}, p_k, t)^2. \tag{13}$$

So far we have only discussed the segment distortion measures, i.e., the measures which judge the approximation of a certain partial boundary by a given polygon segment. In general we are interested in a polygon distortion measure which can be used to determine the quality of approximation of an entire polygon. We will treat two different classes of polygon distortion measures. The first class is based on the

maximum operator (or equivalently, on the minimum operator) and is of the following form,

$$D(p_0, \ldots, p_{N_P-1}) = \max_{k \in [0,\ldots,N_P-1]} d(p_{k-1}, p_k), \tag{14}$$

where $d(p_{-1}, p_0)$ is defined to be zero. We will denote all distortion measures based on the above definition as class one distortion measures.

The second class of distortion measures is based on the summation operator and is of the following form,

$$D(p_0, \ldots, p_{N_P-1}) = \sum_{k=0}^{N_P-1} d(p_{k-1}, p_k), \tag{15}$$

where again $d(p_{-1}, p_0)$ is set equal to zero. We will denote all distortion measures based on the above definition as class two distortion measures.

The main motivation for considering these two classes of distortion measures stems from the popularity of the maximum absolute distance distortion measure, which is a class one measure, and the mean squared distance distortion measure, which is a class two measure. If we select the maximum absolute distance as the polygon distortion measure, then we have to use Eq. (12) for the segment distortion and Eq. (14) for the polygon distortion. On the other hand, if we select the mean squared distance as the distortion measure, then we have to use Eq. (13) for the segment distortion and Eq. (15) for the polygon distortion. Note that there are many other polygon distortion measures which fit into this framework, such as the absolute area or the total number of error pels between the boundary and the polygon.

4.2.1 Admissible vertex set different than set of boundary points

We now extend our discussion on distortion measures so that the set of admissible vertex points A is not equal to the set of boundary points B. Also, the boundary approximation is done using curves of any order. Usually, $A \supseteq B$, i.e., A is a superset of B and all original boundary points are eligible to become polygon vertex points. However, this is not necessary, since we can eliminate some boundary points from being candidates for vertex points through a preselection procedure in order to reduce the computational complexity of our algorithm. The preselection procedure indicates which boundary points are unlikely to be selected as optimal vertex points by the boundary encoding algorithm.

Let us now concentrate on the more usual case where $A \supseteq B$. Thus, we now relax the restriction that the admissible vertices for the polygon belong to the original boundary. The main drawback of this restriction is that one can easily construct an example where for a given maximum polygon distortion, the polygon

with the smallest bit rate uses vertices which do not fall onto boundary points. The problem with using vertices which do not belong to the boundary is that for a given polygon segment no direct correspondence exists between the segment and a subset of boundary points. Hence the polygon distortion cannot be formulated as the maximum or sum of the segment distortions unless we define segment distortion in a new way, as we will do in section 4.2.3.

4.2.2 Definition of the admissible vertex set

From a theoretical point of view, the set of admissible vertices should contain all the pels in the image plane. On the other hand, the Directed Acyclic Graph (DAG) shortest path algorithm, which we will be using, has a time complexity which is polynomial in the number of admissible vertices and hence we would like to keep that number as small as possible, without sacrificing coding efficiency. In this approach, the set of all admissible vertices is defined as all the pels which are within a given maximum distance DM from a boundary point (see Fig. 26). Hence the set of admissible vertices forms a "band" of width $2 \cdot DM$ around the original boundary.

4.2.3 Distortion measures

As mentioned earlier, in the case under consideration, there is no obvious correspondence between a segment and a subset of boundary points. Thus, equations (14) and (15) cannot be used directly. A straightforward distortion measure which does not have to be expressed in the form of equations (14) and (15) is the area between the original boundary and the polygon approximation. However, we would prefer to have a distortion measure that can be expressed using these equations because then we can use the DAG shortest path algorithm to find the optimum approximation.

To achieve this in the polygonal approximation case, every admissible vertex (for example p_b in Fig. 26) is associated with the closest boundary point (which is b_b in Fig. 26). Thus, the approximation curve segment (p_a, p_b) is associated with the original boundary segment b_a, b_b. Then, we can define the segment distortion in any one of the previously mentioned ways, such as, maximum absolute distance, mean squared distance, etc.

Let us now consider the second order B-spline approximation case. As mentioned before, a second order B-spline curve segment is defined by three control points, p_{u-1}, p_u, p_{u+1}. The points at the begining and the end of a curve segment are called knots. It can be shown that, for the class of B-splines we are interested

in, the knots are in the midpoints of the straight line segment that connects two consecutive control points. This straight line segment is tangent to the B-spline. Thus, we need to define a correspondence between a curve segment (the portion of the B-spline which is between two knots) and a portion of the original boundary. Again, we associate every knot (for example A in Fig. 27) with the closest boundary point (C in Fig. 27). Then, the approximation curve segment AB is associated with the original boundary segment CD and we can define the segment distortion in any suitable way. Of course, the segment distortion will now be a function of three points, the control points that correspond to the segment: $d(p_{u-1}, p_u, p_{u+1})$.

It should be pointed out that we could have defined the correspondence between an approximation curve segment and an original boundary segment in a different way. The above definition, however, is in our opinion the most natural one.

The same correspondence can be defined in the same way for B-splines of order three or higher. Again, we associate each knot of the B-spline (which can be found for specific values of the parameter t) with the closest boundary point. We then proceed as before.

4.2.4 The distortion band

In this section we present an alternative way of defining the distortion for the maximum approach of the previous section which is easier to implement. We define a distortion band around the original boundary of width D_{max}, the desired maximum allowable distortion. Thus, for a faster implementation of the algorithm, we can calculate the locations of the points from where the boundary approximation is allowed to pass in a preliminary step of our algorithm. Then, it is very easy and fast to see if a candidate segment satisfies the distortion requirement.

4.3 Rate

As mentioned earlier, we assume that the vertices of the polygon are encoded differentially which is an efficient method for natural boundaries since the location of the current vertex is strongly correlated with the location of the previous one. This is the only restriction that we impose on the encoding of the vertices of the polygon or the control points of the parametric curve. We denote the required bit rate for the differential encoding of vertex p_k given vertex p_{k-1} by $r(p_{k-1}, p_k)$. Hence the bit rate $R(p_0, \ldots, p_{N_P-1})$ for the entire polygon is,

$$R(p_0, \ldots, p_{N_P-1}) = \sum_{k=0}^{N_P-1} r(p_{k-1}, p_k), \tag{16}$$

where $r(p_{-1}, p_0)$ is set equal to the number of bits needed to encode the absolute position of the first vertex. For a closed boundary, i.e., the first vertex is identical to the last one, the rate $r(p_{N_P-2}, p_{N_P-1})$ is set to zero since the last vertex does not need to be encoded.

As mentioned earlier, we use a DAG shortest path algorithm in order to determine the optimal control points for our boundary approximation. The DAG shortest path algorithm is a Dynamic Programming (DP) algorithm. A parametric curve of order n requires $n+1$ control points per segment. A straight line segment, which is a first order parametric curve, is defined by two control points, the vertices of the polygon. Three control points are required for a second order B-spline. Thus, in the case of B-splines, the rate required for a segment depends on three control points. Thus,

$$R(p_0, \ldots, p_{N_p+1}) = \sum_{u=0}^{N_P} r(p_{u-1}, p_u, p_{u+1}), \tag{17}$$

where $r(p_{-1}, p_0, p_1)$ is set equal to the number of bits needed to encode the absolute position of the first control point p_0. Note that in the formulation we use for second order B-splines, there are $N_p + 2$ control points in the approximation curve. It is straightforward to generalize the above discussion to higher order parametric curves.

The order of the DP algorithm will be equal to the maximum of the order of the curve and the number of vertices used for the prediction of the current vertex. We are interested in minimizing the order of the DP to reduce computational complexity. Thus, it is advantageous for us to use for the prediction of the current vertex a number of vertices that is equal to the order of the curve.

In the remainder of the paper we introduce fast and efficient algorithms for both classes of polygon distortion measures which solve the following constrained optimization problem,

$$\min_{p_0,\ldots,p_{N_P-1}} D(p_0, \ldots, p_{N_P-1}), \quad \text{subject to:} \quad R(p_0, \ldots, p_{N_P-1}) \leq R_{max}, \tag{18}$$

where R_{max} is the maximum bit rate permitted for the encoding of the boundary. We also present algorithms which solve the dual problem,

$$\min_{p_0,\ldots,p_{N_P-1}} R(p_0, \ldots, p_{N_P-1}), \quad \text{subject to:} \quad D(p_0, \ldots, p_{N_P-1}) \leq D_{max}, \tag{19}$$

where D_{max} is the maximum distortion permitted. Note that there is an inherent tradeoff between the rate and the distortion in the sense that a small distortion requires a high rate, whereas a small rate results in a high distortion. As we will see the solution approaches for problems (18) and (19) are related in the sense

that the algorithms are symmetric with respect to the rate and the distortion or the algorithm developed to solve problem (19) is used iteratively to solve problem (18).

4.4 An algorithm for distortion measures based on the summation operator

In this section we derive a solution to problem (18) for a distortion measure based on Eq. (15), such as the maximum absolute distance. The solution is based on the Lagrange multiplier method [56, 57, 58] and the shortest path algorithm. It should be noted that the algorithm is symmetric in the rate and the distortion and hence the same technique can be employed for the minimum distortion case (Eq. (18)) and the minimum rate case (Eq. (19)). We will therefore only solve the minimum distortion case and the minimum rate case can be solved be applying the following relabeling to the function names: $D(p_0, \ldots, p_{N_P-1}) \leftarrow R(p_0, \ldots, p_{N_P-1})$ and $R(p_0, \ldots, p_{N_P-1}) \leftarrow D(p_0, \ldots, p_{N_P-1})$.

We will present the method for the polygonal approximation case where only boundary points are admissible as polygon vertices. We will, however, discuss how the method is extended to the more general case.

The Lagrange multiplier method is extremely useful for solving constrained resource allocation problems. In this application we will use the Lagrange multiplier method to relax the constraint so that the relaxed problem can be solved using the shortest path algorithm.

We first define the Lagrangian cost function

$$J_\lambda(p_0, \ldots, p_{N_P-1}) = D(p_0, \ldots, p_{N_P-1}) + \lambda \cdot R(p_0, \ldots, p_{N_P-1}), \tag{20}$$

where λ is called the Lagrange multiplier. It has been shown in [56, 57] that if there is a λ^* such that,

$$\{p_0^*, \ldots, p_{N_P-1}^*\} = \arg \min_{p_0, \ldots, p_{N_P-1}} J_{\lambda^*}(p_0, \ldots, p_{N_P-1}), \tag{21}$$

and which leads to $R(p_0, \ldots, p_{N_P-1}) = R_{max}$, then $\{p_0^*, \ldots, p_{N_P-1}^*\}$ is also an optimal solution to (18). It is well known that when λ sweeps from zero to infinity, the solution to problem (21) traces out the convex hull of the operational rate distortion function, which is a non-increasing function. Hence bisection [59] or the fast convex search we presented in [60] can be used to find λ^*. Therefore, if we can find the optimal solution to the unconstrained problem (21), then we can find the optimal λ^* and the convex hull approximation to the constrained problem of Eq. (18).

We can find a very efficient way of minimizing the Lagrangian cost function if we note that

$$J_\lambda(p_0, \ldots, p_{N_p-1}) = \sum_{k=0}^{N_p-1} w(p_{k-1}, p_k) \tag{22}$$

where

$$w(p_{k-1}, p_k) = d(p_{k-1}, p_k) + \lambda \cdot r(p_{k-1}, p_k) \tag{23}$$

are graph weights. It is clear from the above equations that the problem of minimizing the Lagrangian cost function can be formulated in the form of a directed graph (See Figure 28). The vertices of the graph correspond to the admissible vertex points (control points in the higher order case) and the edges correspond to the possible segments of the approximation polygon. The edges have weights $w(p_{k-1}, p_k)$. The a_j's in this figure are the admissible vertex points. For the time being, let us assume that only original boundary points are eligible to become vertex points. Thus, our problem reduces to finding the shortest path between the first and the last vertex of the graph. This will give us the optimal vertices of the approximation polygon, since it will give us the path that minimizes the Lagrangian cost function.

We need to start the search for an optimal polygon at a given vertex. If the boundary is not closed, b_0 has to be selected as the first vertex p_0. For a closed boundary, the selection of the first vertex is less obvious. Ideally the algorithm should find all the optimal vertices, including the first one. Unfortunately, the above DAG requires a starting vertex. Hence, we need to fix the first vertex, even for a closed boundary. Therefore the found solution is optimal, given the constraint of the predetermined first vertex. Clearly we can drop this constraint by finding all optimal approximations using each boundary point as a starting vertex and then selecting the overall best solution. This exhaustive search with respect to the initial vertex is computationally quite expensive. We therefore propose to select the point with the highest curvature as the first vertex, since it is the most likely point to be included in any polygonal approximation. This heuristic almost always results in the best possible selection of the initial vertex and if not, the performance difference is negligible.

We relabel the boundary so that the first vertex of the polygon p_0 coincides with the first point of the boundary b_0. Besides fixing the first vertex of the polygon, we also require that the last vertex p_{N_P-1} is equal to the last point of the boundary b_{N_B-1}. This leads to a closed polygonal approximation for a closed boundary. For a boundary which is not closed, this condition, together with the starting condition, makes sure that the approximation starts and ends at the same points as the boundary.

The classical algorithm for solving such a shortest path problem is Dijkstra's algorithm [61]. We can,

however, use a simpler algorithm by observing that it is very unlikely for the optimal path to select a boundary point b_j as a vertex when the last selected vertex was b_i, where $i > j$. In general we cannot guarantee that the optimal path will not do this since the selection process depends on the vertex encoding scheme, which we have not specified yet. On the other hand, a polygon where successive vertices are not assigned to boundary points in increasing order can exhibit rapid direction changes even when the original boundary is quite smooth (see Fig. 29). Therefore we add the restriction that not every possible combination of (b_i, b_j) represents a valid edge but only the ones for which $i < j$. Hence the edge set E is redefined in the following way, $E = \{(b_i, b_j) \in B^2 : i < j\}$ (see Fig. 30). This restriction results in the fact that a given vertex set uniquely specifies the polygon.

If we impose the above restriction for the admissible solution, our graph becomes a Directed Acyclic Graph (DAG). Thus, we can use the DAG shortest path algorithm [61] which has a lower computational complexity than Dijkstra's algorithm.

4.4.1 Extension to the general case

In this section, we extend the above algorithm to the more general case of higher order approximation curves. Let us now assume that we are using second order B-splines instead of polygons. For simplicity, let us assume that only boundary points are eligible to become control points. We will revisit the problem of section 4.4 for the case of second order B-splines.

Let us rewrite the Lagrangian cost function as

$$J_\lambda(p_0, \cdots p_{N_p-1}) = \sum_{u=1}^{N_P} w(p_{u-1}, p_u, p_{u+1}) \tag{24}$$

where

$$w(p_{u-1}, p_u, p_{u+1}) = d(p_{u-1}, p_u, p_{u+1}) + \lambda \cdot r(p_{u-1}, p_u, p_{u+1}). \tag{25}$$

Note that now $w(.)$, $d(.)$ and $r(.)$ depend on three control points. Also note that, as mentioned earlier, in the begining and the end of the curve, we have double control points.

The above two points lead us to define the vertices of the graph, which we will now call states, in a different way. The states are now two-dimensional, as shown in Fig. 31. Each state now involves two possible control points (a_i, a_j). As mentioned earlier, the weights involve three control points. Except for these differences, the resulting graph is a DAG and the shortest path can be found using the DAG shortest path algorithm.

The algorithm can be extended to higher dimensional B-splines. In this case, the states of the DAG would have a dimension equal to the order of the curves.

Let us now assume that the set of admissible vertices or control points is a superset of the set of boundary points. The above algorithm can be used intact if we order the admissible vertices or control points a_j in a systematic way. This is necessary in order for the problem to be expressed in the form of a DAG. Any reasonable way for ordering the admissible vertices can be used. The interested reader is referred to [62].

4.5 Algorithms for distortion measures based on the maximum operator

4.5.1 Minimum rate case

We now consider the minimum rate case which is stated in Eq. (19). We assume a distortion measure which is based on the maximum operator. This problem can be solved in exactly the same way shown in section 4.4 by redefining $j_\lambda(p_{k-1}, p_k)$ as

$$w(p_{k-1}, p_k) = \begin{cases} \infty & : \quad d(p_{k-1}, p_k) > D_{max} \\ r(p_{k-1}, p_k) & : \quad d(p_{k-1}, p_k) \leq D_{max} \end{cases} . \tag{26}$$

Note that the above definition of the weight function leads to a length of infinity for every path (polygon) which includes a line segment resulting in an approximation error larger than D_{max}. Therefore a shortest path algorithm will not select these paths. Every path which starts at vertex p_0 and ends at vertex p_{N_B-1} and does not result in a path length of infinity, results in a path length equal to the rate of the polygon it represents. Therefore the shortest of all those paths corresponds to the polygon with the smallest bit rate which is the solution to the problem in Eq. (19).

4.5.2 Minimum distortion case

We now consider the minimum distortion case which is stated in Eq. (18). The goal of the proposed algorithm is to find the polygon with the smallest distortion for a given bit budget for encoding its vertices. Sometimes this is also called a rate constrained approach. Recall that for class one distortion measures the polygon distortion is defined as the maximum of the segment distortions (see Eq. (14)). Hence in this section we propose an efficient algorithm which finds the polygonal approximation with the smallest maximum distortion for a given bit rate.

We propose an iterative solution to this problem which is based on the fact that we can solve the dual problem stated in Eq. (19) optimally. Consider D_{max} in Eq. (19) to be a variable. We derived in

section 4.5.1 an algorithm which finds the polygonal approximation which results in the minimum rate for any D_{max}. We denote this optimal rate by $R^*(D_{max})$. It was proven in [62] that the rate $R^*(D_{max})$ is a non-increasing function of D_{max}, which means that $D^1_{max} < D^2_{max}$ implies $R^*(D^1_{max}) \geq R^*(D^2_{max})$.

Thus, we can use bisection [59] to find the optimal D^*_{max} such that $R^*(D^*_{max}) = R_{max}$. Since this is a discrete optimization problem, the function $R^*(D_{max})$ is not continuous and exhibits a staircase characteristic (see Fig. 33). This implies that there might not exist a D^*_{max} such that $R^*(D^*_{max}) = R_{max}$. In that case the proposed algorithm will still find the optimal solution, which is of the form $R^*(D^*_{max}) < R_{max}$, but only after an infinite number of iterations. Therefore if we have not found a D_{max} such that $R^*(D_{max}) = R_{max}$ after a given maximum number of iterations, we terminate the algorithm.

4.5.3 The sliding window

By using the distortion band (sec. 4.2.4) for the maximum distortion approach, the solution of the DAG shortest path algorithm may result in a trivial solution (see Fig. 32). We need a way to force the algorithm along the boundary in order to find a curve. With the introduction of a *sliding window* we not only avoid trivial solutions but also are able to control the speed of the algorithm. The sliding window indicates the admissible selections for the next control point p_u (see Fig. 32). Thus, trivial solutions are eliminated and the computational complexity of the algorithm is decreased.

4.6 Experimental results

4.6.1 Polygonal approximation case

In this section we present experimental results of the proposed algorithms using object boundaries from the "Miss America" sequence. We first present results for class one distortion measures, where the employed distortion measure is the minimum absolute distance. In Fig. 34 we compare the original segmentation, which is displayed in the left figure, versus the optimal segmentation for a maximum distortion D_{max} of one pel, which is displayed in the right figure. The two objects in the original segmentation require 468 bits if encoded by an 8-connect chain code whereas the optimal segmentation can be encoded with only 235 bits. By introducing a permissible maximum error of one pel, we are able to reduce the total bit rate by about 50%. As expected, some of the details have been lost, i.e., the boundary has been "straightened". This smoothing of the boundary might be desired since most segmentation algorithms result in noisy boundaries. In Fig. 35 we show the resulting segmentation for the minimum distortion case for multiple boundaries.

The maximum rate R_{max} has been set to 280 bits and the optimal solution, which uses 274 bits for a $D_{max} = 0.71$ pels, is displayed in the left figure. The right figure is a closeup of the lower boundary in the left figure and the stars indicate the original boundary with the polygonal approximation drawn on top of it.

4.6.2 B-spline approximation case

We encoded the same boundaries as in the previous section using the B-spline algorithm. The distortion band width was also set to 1 pel. The "neck" object required 127 bits whereas the "mouth" object required 84 bits for a total of 211 bits (Figure 36). The polygonal approximation required 235 bits.

We also encoded 100 frames of the "Kids" sequence and averaged the resulting bit rates and d_n. The experiments were ran for $D_{max} = 0.7$, 1, 1.5, 2, 2.5 and 3 pels. The results are shown in Figure 37. It can be seen that the results are comparable with the best results achieved by the baseline algorithm in MPEG-4 (Figure 17). It should be pointed out here that the control point encoding scheme used [62] is very simple and hence further research will very likely result in an overall scheme which is even more efficient.

Figure 38 shows the reconstructed frame 17 of the "Kids" sequence using $D_{max} = 0.7$. The resulting d_n was 0.0171 and 979 bits were required for the encoding.

5 Conclusions

Within MPEG-4, several shape coding algorithms were evaluated. In terms of coding efficiency, a context-based shape coder operating on a macroblock basis and a vertex-based shape coder performed similarly with the context-based shape coder providing better coding efficiency for lossless shape coding and the vertex-based shape coder providing slightly better coding efficiency for lossy shape coding. However, in terms of hardware implementation complexity, the block-based method allows for a regular memory access to the shape information whereas the vertex-based method requires irregular access to an off-chip cache. Therefore the macroblock-based context-based shape coder was selected as the base technology of MPEG-4 shape coding. It integrates nicely with the block-based texture motion compensation and coding. Within a macroblock, the context-based shape coder defines a 9 or 10 pel causal context for encoding the binary shape information of the current pel. An arithmetic encoder is used to encode the current pel given the context. In order to exploit temporal redundancies, shape motion vectors with integer pel precision

are estimated for shape motion compensation. These vectors are encoded predictively with respect to the texture motion vectors. Lossless shape coding of simple head and shoulder scenes at CIF resolution may require 300 bits/frame for the head-and-shoulder object, complicated objects may require an order of magnitude more bits. For transparent objects, the object outline is encoded using the binary shape coding technique. If the object has constant transparency, this transparency value gets transmitted along with optional boundary filter information. For non-constant transparency information, the 8bit α-plane is encoded using a hybrid coder with DCT and motion compensation.

The selection of the shape coding algorithm was based on information available in April 1997. Considering that shape coding is a relatively new area within image processing, more research is required to gain a complete understanding of all the different aspects of shape coding. MPEG-4 will serve as the baseline against which new shape coding techniques will have to compete.

As a first step towards this direction we presented an efficient vertex-based framework for the lossy encoding of object shapes. The object shape is approximated by a polygon or second order B-spline curve which leads to the smallest bit rate for a given distortion. The polygon or B-spline curve is found by a shortest path algorithm since the problem can be described as a single-source directed acyclic graph (DAG). In our discussion, we did not elaborate on the encoding of the vertices for the polygon and the control points for the B-spline. We used a simple extension to 8-connect chain codes in the experimental results. However, a more sophisticated encoding scheme for the control point vectors might result in lower bit rates. Higher order curves as well as distortion bands of variable width can be incorporated into the proposed framework in a straightforward way.

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

Acknowledgment

The authors would like to thank C. S. Boon, Matsushita for providing the data for Figures 21 and 22, and Corinne LeBuhan Jordan, EPFL, for providing Figure 14. J. S. Shin, SAIT, created Figures 15-20 during the MPEG meeting in Bristol to help the evaluation. Ming Rong created Figure 1. Further thanks go to

Yao Wang and Amy Reibman for their review of MPEG-4 related parts of this paper.

References

[1] H. Musmann, M. Hötter, and J. Ostermann, "Object-oriented analysis-synthesis coding of moving images," *Signal Processing: Image Communication*, vol. 1, pp. 117–138, Oct. 1989.

[2] M. Kunt, "Second-generation image coding techniques," *Proceedings of the IEEE*, vol. 73, pp. 549–574, Apr. 1985.

[3] R. Koenen,Ed., "Overview of the MPEG-4 standard," July 1997. ISO/IEC JTC1/SC29/WG11 N1730, Stockholm meeting.

[4] W. Li and M. Kunt, "Morphological segmentation applied to displaced difference coding," *Signal Processing*, vol. 38, pp. 45–56, July 1994.

[5] T. Alpert, V. Baroncini, D. Choi, L. Contin, R. Koenen, and F. P. and. Peterson, "Subjective evaluation of MPEG-4 codec proposals: Methodological approach and test procedures," *Signal Processing: Image Communication*, vol. 9, pp. 303–325, May 1997.

[6] N. Diehl, "Object-oriented motion estimation and segmentation in image sequences," *Signal Processing: Image Communication*, vol. 3, pp. 23–56, 1991.

[7] J. Ostermann, "Object-oriented analysis-synthesis coding based on the source model of moving rigid 3d objects," *Signal Processing: Image Communication*, no. 6, pp. 143–161, 1994.

[8] M. Hötter, "Optimization and efficiency of an object-oriented analysis-synthesis coder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, pp. 181–194, Apr. 1994.

[9] J. Ostermann, "Object-oriented analysis-synthesis coding (OOASC) based on the source model of moving flexible 3D objects," *IEEE Transactions on Image Processing*, vol. 3, Sept. 1995.

[10] P. Gerken, "Object-based analysis-synthesis coding of image sequences at very low bit rates," *IEEE Transactions on Circuits and Systems*, vol. 4, pp. 228–235, June 1994.

[11] G. Martinez, "Shape estimation of articulated objects for object-based analysis-synthesis coding (OBASC)," *Signal Processing: Image Communication*, vol. 9, pp. 175–199, Mar. 1997.

[12] Y. Wang and O. Lee, "Active mesh - a feature seeking and tracking image sequence representation scheme," *IEEE Transactions on Image Processing*, vol. 3, pp. 610–624, Sept. 1994.

[13] Y. Altunbasak and A. M. Tekalp, "Closed-form connectivity-preserving solutions for motion compensation using 2-D meshes," *IEEE Transactions on Image Processing*, vol. 6, pp. 1255–1269, Sept. 1997.

[14] CompuServe Incorporated, Columbus, Ohio, *Graphics Interchange format (sm). Version 89a*, July 1990.

[15] "Facsimile coding schemes and coding functions for group 4 facsimile apparatus." CCITT Recommendation T.6, 1994.

[16] "Coded representation of picture and audio information- progressive bi-level image compression." ISO Draft International Standard 11544, 1992.

[17] A. N. Netravali and B. Haskell, *Digital Pictures, Representation and Compression.* Plenum Press, 1988.

[18] H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 260–268, June 1961.

[19] M. Eden and M. Kocher, "On the performance of a contour coding algorithm in the context of image coding. part i: Contour segment coding," *Signal Processing*, vol. 8, pp. 381–386, July 1985.

[20] M. Hötter, "Object-oriented analysis-synthesis coding based on moving two-dimensional objects," *Signal Processing: Image Communication*, vol. 2, pp. 409–428, Dec. 1990.

[21] G. M. Schuster and A. K. Katsaggelos, "An optimal segmentation encoding scheme in the rate distortion sense," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 1996.

[22] G. M. Schuster and A. K. Katsaggelos, "An optimal boundary encoding scheme in the rate di stortion sense," *IEEE Transactions on Image Processing.* To appear January 1998.

[23] K. J. O'Connell, "Object-adaptive vertex-based shape coding method," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, pp. 251–255, Feb. 1997.

[24] C. T. Zahn and R. Z. Roskies, "Fourier descriptors for plane closed curves," *IEEE Trans. on Computers*, vol. C21, pp. 269–281, Mar. 1972.

[25] P. Salembier, F. Marques, and A. Gasull, *Video Coding*, ch. Coding of partition sequences, pp. 125–170. Kluwer Academic Publishers, Boston, 1996.

[26] J. Saghri and H. Freeman, "Analysis of the precision of generalized chain codes for the representation of planar curves," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-3, pp. 533–539, Sept. 1981.

[27] J. Koplowitz, "On the performance of chain codes for quantization of line drawings," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-3, pp. 180–185, Mar. 1981.

[28] D. Neuhoff and K. Castor, "A rate and distortion analysis of chain codes for line drawings," *IEEE Transactions on Information Theory*, vol. IT-31, pp. 53–68, Jan. 1985.

[29] T. Kaneko and M. Okudaira, "Encoding of arbitary curves based on the chain code representation," *IEEE Transactions on Communications*, vol. COM-33, pp. 697–707, July 1985.

[30] R. Prasad, J. W. Vieveen, J. H. Bons, and J. C. Arnbak, "Relative vector probabilities in differential chain coded line-drawings," in *Proc. IEEE Pacific Rim Conference on Communication, Computers and Signal Processing*, (Victoria, Canada), pp. 138–142, June 1989.

[31] P. Nunes, F. Pereira, and F. Marques, "Multi-grid chain coding of binary shapes," in *Special session on shape coding, ICIP97*, (Santa Barbara), 1997.

[32] T. Ozcelik and A. K. Katsaggelos, *Video Data Compression for Multimedia Computing*, ch. Very Low Bit Rate Video Coding Based on Statistical Spatio-Temporal Prediction of Motion, Segmentation and Intensity Fields, pp. 313–353. Kluwer Academic Publishers, 1997.

[33] A. K. Jain, *Fundamentals of digital image processing.* Prentice-Hall, 1989.

[34] P. van Otterloo, *A contour-oriented approach for shape analysis.* Prentice Hall International (UK), 1991.

[35] N. Brady, F. Bossen, and N. Murphy, "Context-based arithmetic encoding of 2D shape sequences," in *Special session on shape coding, ICIP97*, (Santa Barbara), pp. I–29–32, 1997.

[36] N. Yamaguchi, T. Ida, and T. Watanabe, "A binary shape coding method using modified MMR," in *Special session on shape coding, ICIP97*, (Santa Barbara), pp. I–504–508, 1997.

[37] S. Lee, D. Cho, Y. Cho, S. Son, E. Jang, and J. Shin, "Binary shape coding using 1-D distance values from baseline," in *Special session on shape coding, ICIP97*, (Santa Barbara), pp. I–508–511, 1997.

[38] J. Ostermann, "Methodologies used for evaluation of video tools and algorithms in MPEG-4," *Signal Processing: Image Communications, Special Issue on MPEG-4*, no. 9, pp. 343–365, 1997.

[39] T. Sikora, S. Bauer, and B. Makai, "Efficiency of shape-adaptive transforms for coding of arbitrarily shaped image segments," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 5, June 1995.

[40] J. Kim and B. L. Evans, "Predictive shape coding using generic polygon approximation," in *ISCAS 98*, (Monterrey, California), May 1998. To appear.

[41] T. Chen, C. T. Swain, and B. G. Haskell, "Coding of subregions for content-based scalable video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, pp. 256–260, Feb. 1997.

[42] P. Kuhn, "iprof." http://www.lis.e-technik.tu-muenchen.de/people/second_page/kp/.

[43] P. Kuhn, "A portable instruction level profiler for complexity analysis documentation," tech. rep., Tampere, Finland, 1996. ISO/IEC JTC1/WG11 MPEG96/M0921.

[44] J. Arnold and M. Frater, "Summary of reflector discussions on the error resilience of shape coding proposals," tech. rep., Apr. 1997. Private communication at ISO/IEC JTC1/SC29/WG11 Bristol meeting.

[45] J. Brailean, "Report of ad-hoc group on error resilience," Apr. 1997. ISO/IEC JTC1/SC29/WG11 M2113, Bristol meeting.

[46] "MPEG-4 video verification model version 8.0," T. Ebrahimi, Ed., July 1997. ISO/IEC JTC1/SC29/WG11 MPEG97/N1796.

[47] "MPEG-4 video verification model version 5," T. Ebrahimi, Ed., Nov. 1996. ISO/IEC JTC1/SC29/WG11 MPEG97/N1469.

[48] J. W. Stroming, Y. Kang, T. S. Huang, and S. M. Kang, "New architectures for modified MMR shape coding," in *ISCAS 97*, (Hong Kong), pp. 1205–1208, June 1997.

[49] W. Chen and M. Lee, "Alpha-channel compression in video coding," in *Special session on shape coding, ICIP97*, (Santa Barbara), 1997.

[50] F. W. Meier, "An efficient shape coding scheme using B-splines which is optimal in the rate-distortion sense," Master's thesis, Northwestern University, Evanston, IL, June 1997.

[51] F. W. Meier, G. M. Schuster, and A. K. Katsaggelos, "An efficient boundary encoding scheme which is optimal in the rate-distortion sense," in *Proceedings of the International Conference on Image Processing*, (Santa Barbara, CA), pp. II–9–12, 1997.

[52] F. W. Meier, G. M. Schuster, and A. K. Katsaggelos, "An efficient boundary encoding scheme using B-spline curves which is optimal in the rate-distortion sense," in *2nd Erlangen Symposium, Advances in Digital Image Communication*, (Erlangen, Germany), pp. 75–84, Apr. 1997.

[53] Foley, vanDam, Feiner, and Hughes, *Computer Graphics: Principles and Practice*, pp. 478–516. Addison-Wesley, 1990.

[54] G. M. Schuster and A. K. Katsaggelos, "An optimal lossy segmentation encoding scheme," in *Proceedings of the Conference on Visual Communications and Image Processing*, pp. 1050–1061, SPIE, Mar. 1996.

[55] G. M. Schuster and A. K. Katsaggelos, "An optimal segmentation encoding scheme in the rate-distortion sense," in *Proceedings of the International Symposium on Circuits and Systems*, vol. 2, (Atlanta, GA), pp. 640–643, May 1996.

[56] H. Everett, "Generalized Lagrange multiplier method for solving problems of optimum allocation of resources," *Operations Research*, vol. 11, pp. 399–417, 1963.

[57] Y. Shoham and A. Gersho, "Efficient bit allocation for an arbitrary set of quantizers," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, pp. 1445–1453, Sept. 1988.

[58] K. Ramchandran, A. Ortega, and M. Vetterli, "Bit allocation for dependent quantization with applications to multiresolution and MPEG video coders," *IEEE Transactions on Image Processing*, vol. 3, pp. 533–545, Sept. 1994.

[59] C. F. Gerald and P. O. Wheatley, *Applied numerical analysis.* Addison Wesley, fourth ed., 1990.

[60] G. M. Schuster and A. K. Katsaggelos, "Fast and efficient mode and quantizer selection in the rate distortion sense for H.263," in *Proceedings of the Conference on Visual Communications and Image Processing*, pp. 784–795, SPIE, Mar. 1996.

[61] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to algorithms.* McGraw-Hill Book Company, 1991.

[62] G. M. Schuster and A. K. Katsaggelos, *Rate-Distortion Based Video Compression, Optimal Video frame compression and Object boundary encoding.* Kluwer Academic Press, 1997.

[63] "Core experiments on MPEG-4 video shape coding," J. Ostermann, Ed., Feb. 1997. ISO/IEC/JTC1/SC29/WG11 N1584, Seville meeting.

[64] M. Hötter, *Objectorientiere Analyse-Synthese Codierung basierend auf dem Modell bewegter, zweidimensinalen Objekte.* PhD thesis, Universität Hannover, Fortschritt-Berichte VDI, Reihe 10, No. 217, Hannover, 1992.

[65] C. Le Buhan Jordan, F. Bossen, and T. Ebrahimi, "Scalable shape representation for content based visual data compression," in *Special session on shape coding, ICIP97*, (Santa Barbara), 1997.

[66] C. S. Boon, Matsushita, July 1995. Private communications.

Figure 1: This image shows a scene composed of an object with constant and with arbitrary transparency on a background.

Figure 2: A chain code follows the contour of an object by describing the direction from one boundary pel to the next. Each arrow is represented by 1 out of 4 or 1 out of 8 symbols, respectively. On the right, the symbols for differentially encoding the shape are given.

Figure 3: Successive polygon approximation of a contour (from [7]). The initial polygon AB is extended by points B and C. The iteration from a 4 point to a 5 point approximation is shown here.

Figure 4: Approximation using a polygon/spline approximation (from [64]).

Figure 5: 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 frame of the video object.

Figure 6: The upsampled pels (x) lie between the location of the subsampled pels (o). Neighboring pels (bold o) defining the values of the pels to be upsampled (bold x).

Figure 7: Changing pels are used in modified MMR shape coding to define object boundaries.

Figure 8: For Intra coding, a template positioned relative to $b1$ is used for selecting VLC tables in Vertical Mode.

Figure 9: Example of vertex adjustment (from [63]).

Figure 10: The difference V_d between V_{k-1} and the vertex V_k to be coded determines the octant where V_k is located.

Figure 11: Vertex-based shape coding: Motion compensation aligns the previous and the current shape parameters. Unmatched segments of the contours are identified, here (B_1, A_2), (B_2, A_3), (B_3, A_1) for lossless shape coding.

Figure 12: For lossy shape coding, a band with the allowable shape distortion reduces and shortens the unmatched segments, here (B_1, A_2) and $B_2, A_3)$ (compare with Figure 11).

Figure 13: Baseline-based shape coder: Clockwise contour tracing from a baseline using turning points and the distance from the baseline to the contour points.

Figure 14: Three of the test sequences for evaluation of shape coders are (from left to right) Weather, Children and Robot.

Figure 15: Lossy encoding using a bitmap-based (left) and a contour-based (right) shape coder. The bitmap-based shape coder used pel replication as the upsampling filter (from [65]).

Figure 16: Comparison of rate distortion curves in Intra mode. Bit rate and distortion averages are given for 100 frames.

Figure 17: Comparison of rate distortion curves in Intra mode. Bit rate and distortion averages are given for 100 frames.

Figure 18: Comparison of rate distortion curves in Intra mode. Bit rate and distortion averages are given for 100 frames.

Figure 19: Comparison of rate distortion curves in Inter mode. Bit rate and distortion averages are given for 100 frames.

Figure 20: Comparison of rate distortion curves in Inter mode. Bit rate and distortion averages are given for 100 frames.

Figure 21: Comparison of rate distortion curves in Inter mode. Bit rate and distortion averages are given for 100 frames.

Figure 22: Comparison of chroma-key shape coder and MPEG-4 VM 5 with lossless shape coding ($d_n = 0.0$). Quantizer stepsizes 8, 12 and 20 are used (from [66]).

Figure 23: Comparison of chroma-key shape coder and MPEG-4 VM 5 with lossless shape coding ($d_n = 0.0$). Quantizer stepsizes 8, 12 and 20 are used (from [66]).

Figure 24: A second degree B-spline curve with 8 curve segments Q_u and a double control point at the beginning of the curve.

Figure 25: Definition of the distortion $d(4, 14)$ that corresponds to the edge $(4, 14)$.

Figure 26: The set of admissible vertices (polygon) or admissible control points (B-spline) forms a band of width $2 \cdot DM$ around the boundary.

Figure 27: Definition of the correspondence between B-spline segments and original boundary segments. Round bullets: control points, rectangular bullets: knots.

Figure 28: A specific example of a Directed Acyclic Graph for the polygonal approximation case. The optimal path is shown in bold.

Figure 29: Examples of polygons with rapid changes in direction.

Figure 30: Interpretation of the boundary and the polygon approximation as a weighted directed graph. Note that the set of all segments E equals $\{(b_i, b_j) \in B^2 : i < j\}$. Two representative subsets are displayed: (a) $\{(b_4, b_j) \in B^2 : \forall j > 4\}$ and (b) $\{(b_8, b_j) \in B^2 : \forall j > 8\}$.

Figure 31: An example of a Directed Acyclic Graph for the B-spline approximation case. One of these paths is the optimal path.

Figure 32: The sliding window restricts the selection of control point p_{u+1} to all the admissible control points within the sliding window. The introduction of a sliding window prevents trivial solutions.

Figure 33: The $R^*(D_{max})$ function, which is a non-increasing function exhibiting a staircase characteristic. The selected R_{max} falls onto a discontinuity and therefore the optimal solution is of the form $R^*(D^*_{max}) < R_{max}$, instead of $R^*(D^*_{max}) = R_{max}$.

Figure 34: Left figure: original segmentation which requires 468 bits using the 8-connect chain code. Right figure: optimal segmentation with $D_{max} = 1$ pixel which requires a rate of 235 bits and results in a distortion of 1 pixel.

Figure 35: Left figure: optimal segmentation with $R_{max} = 280$ bits which results in a distortion of 0.71 pixels and a bit rate of 274 bits. Right figure: closeup of the lower boundary; the stars indicate the original boundary and the line represents the polygonal approximation. The upper left corner has been selected as the first vertex.

Figure 36: Results obtained using the B-spline approach. Dotted line: original boundary, continuous line: B-spline approximation, stars: control points.

Figure 37: Rate distortion curve for the "Kids" sequence using the B-spline approach in the Intra-mode. Bit rates and distortion averages are given for 100 frames.

Figure 38: Frame 17 of the "Kids" sequence encoded using $D_{max} = 0.7$. 979 bits were required resulting in $d_n = 0.0171$.

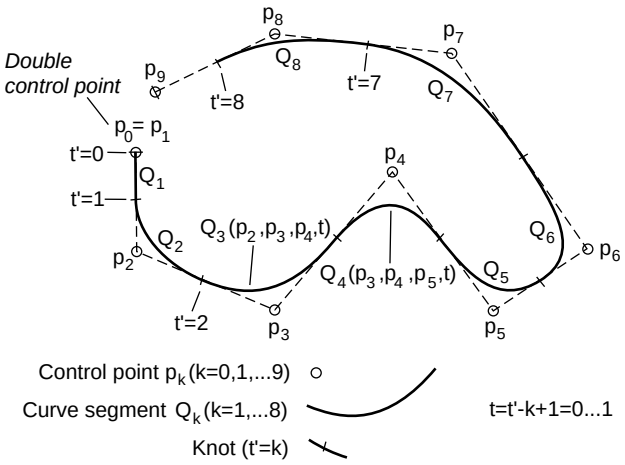


Figure 24: A second degree B-spline curve with 8 curve segments Q_u and a double control point at the beginning of the curve.

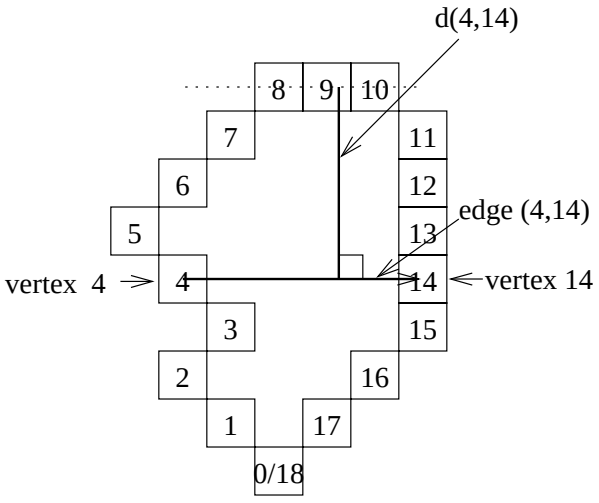


Figure 25: Definition of the distortion $d(4, 14)$ that corresponds to the edge $(4, 14)$

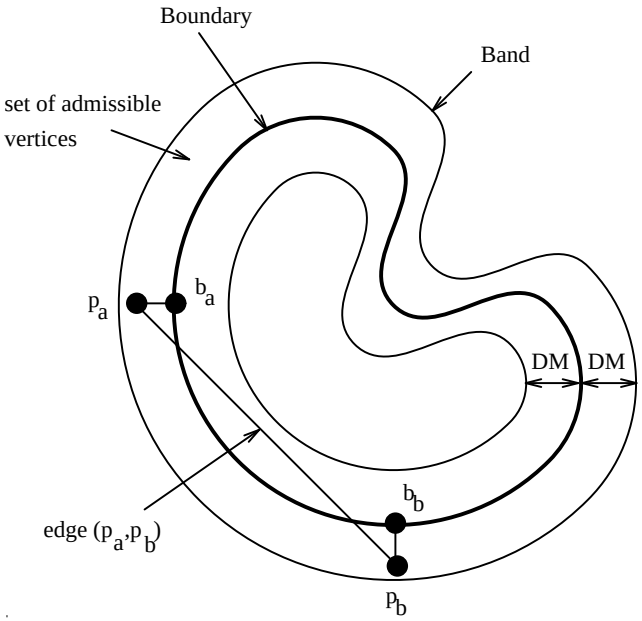


Figure 26: The set of admissible vertices (polygon) or admissible control points (B-spline) forms a band of width $2 \cdot DM$ around the boundary.

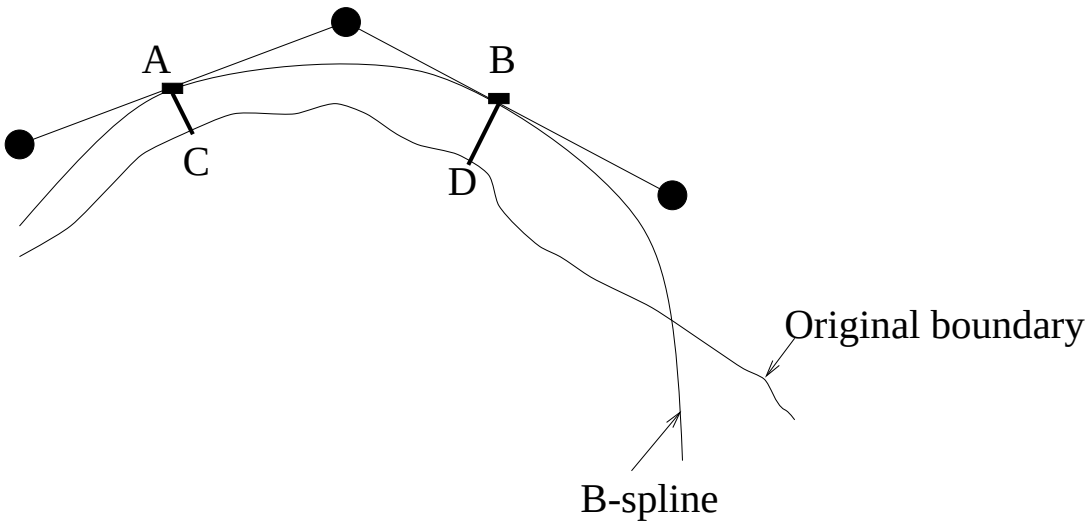


Figure 27: Definition of the correspondence between B-spline segments and original boundary segments. Round bullets: control points, rectangular bullets: knots.

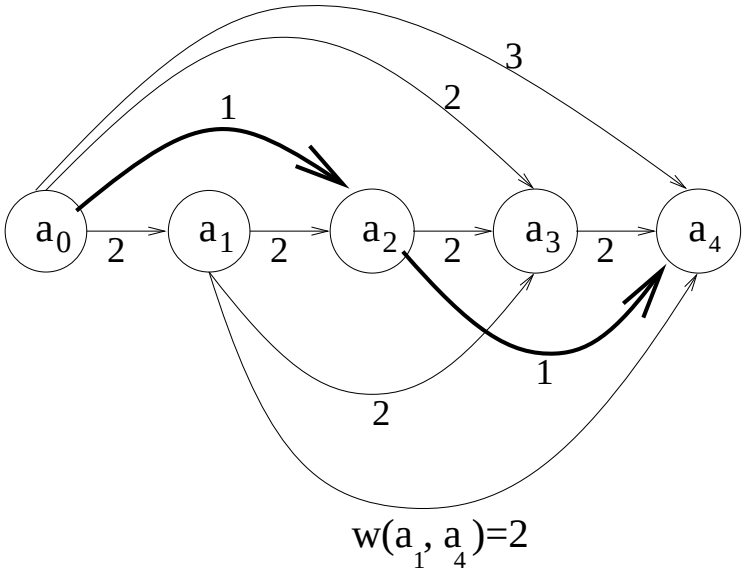


Figure 28: A specific example of a Directed Acyclic Graph for the polygonal approximation case. The optimal path is shown in bold.

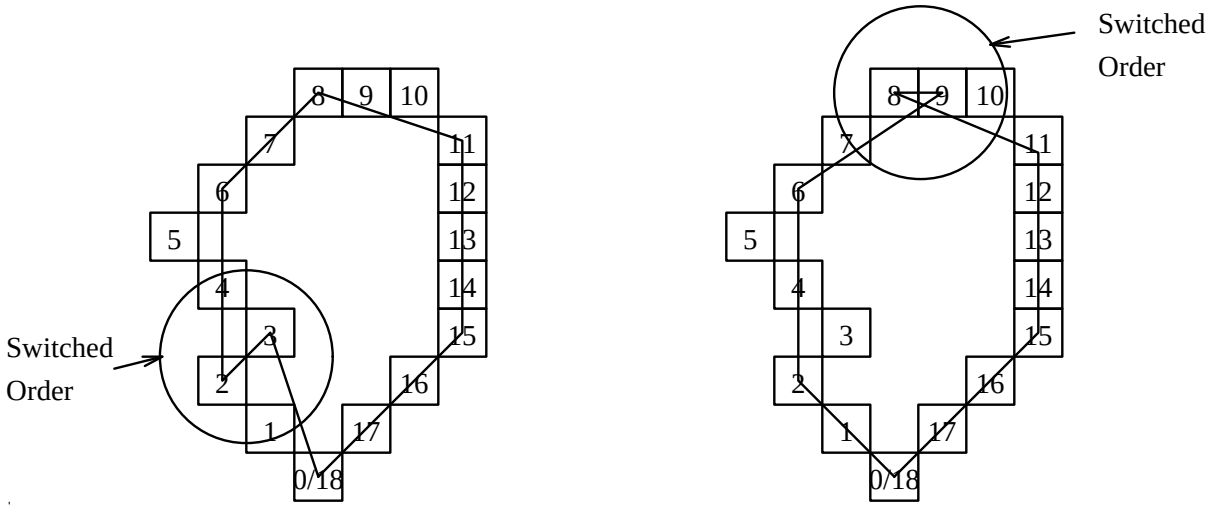


Figure 29: Examples of polygons with rapid changes in direction.

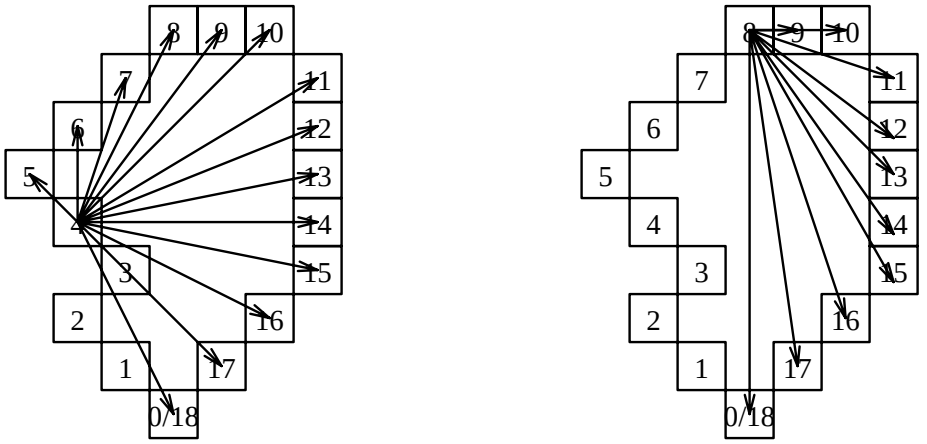


Figure 30: Interpretation of the boundary and the polygon approximation as a weighted directed graph. Note that the set of all segments E equals $\{(b_i, b_j) \in B^2 : i < j\}$. Two representative subsets are displayed: (a) $\{(b_4, b_j) \in B^2 : \forall j > 4\}$ and (b) $\{(b_8, b_j) \in B^2 : \forall j > 8\}$.

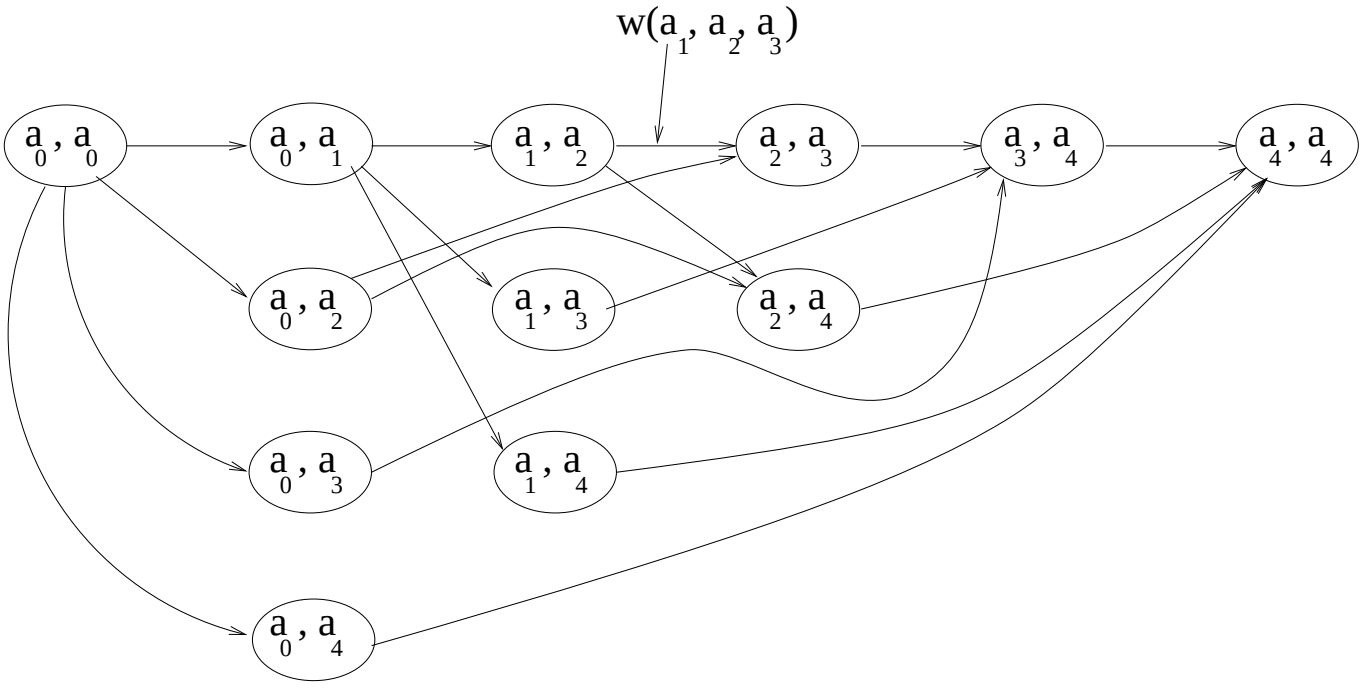


Figure 31: An example of a Directed Acyclic Graph for the B-spline approximation case. One of these paths is the optimal path.

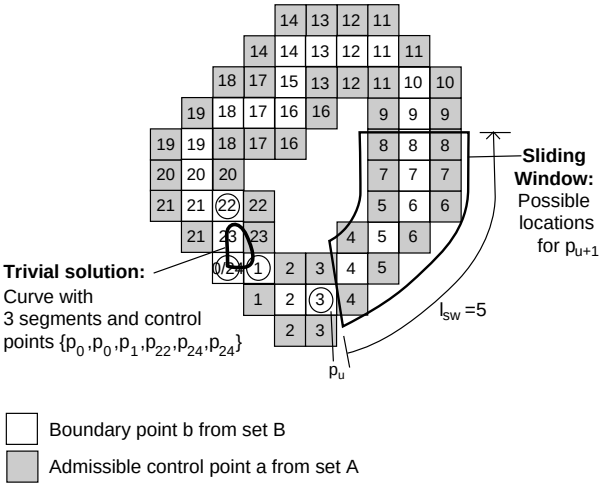


Figure 32: The sliding window restricts the selection of control point p_{u+1} to all the admissible control points within the sliding window. The introduction of a sliding window prevents trivial solutions.

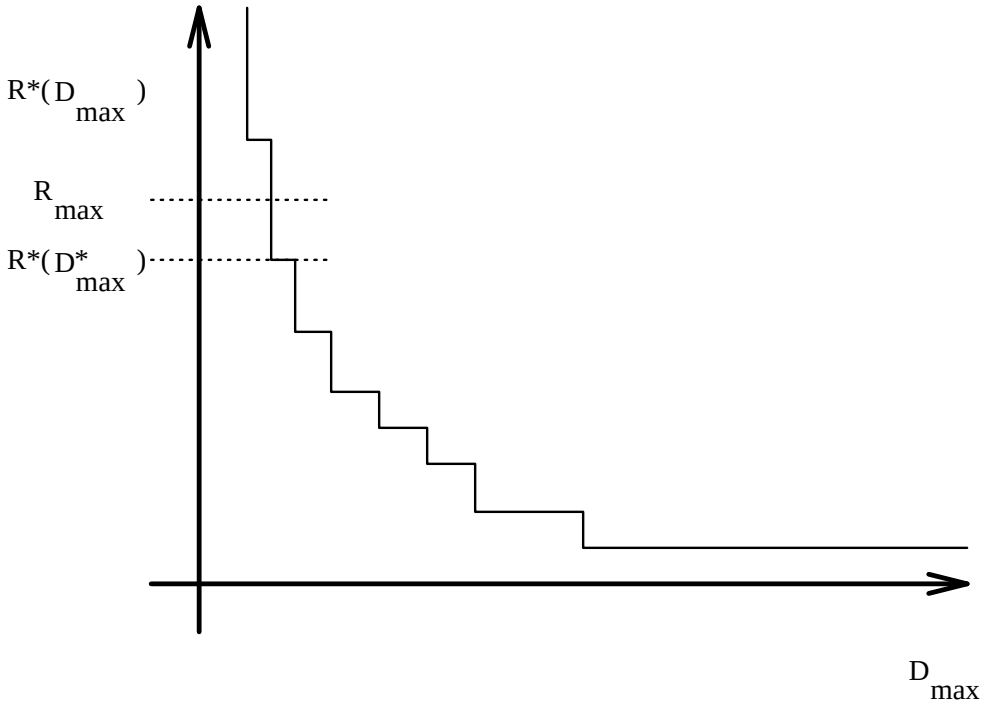


Figure 33: The $R^*(D_{max})$ function, which is a non-increasing function exhibiting a staircase characteristic. The selected R_{max} falls onto a discontinuity and therefore the optimal solution is of the form $R^*(D^*_{max}) < R_{max}$, instead of $R^*(D^*_{max}) = R_{max}$.

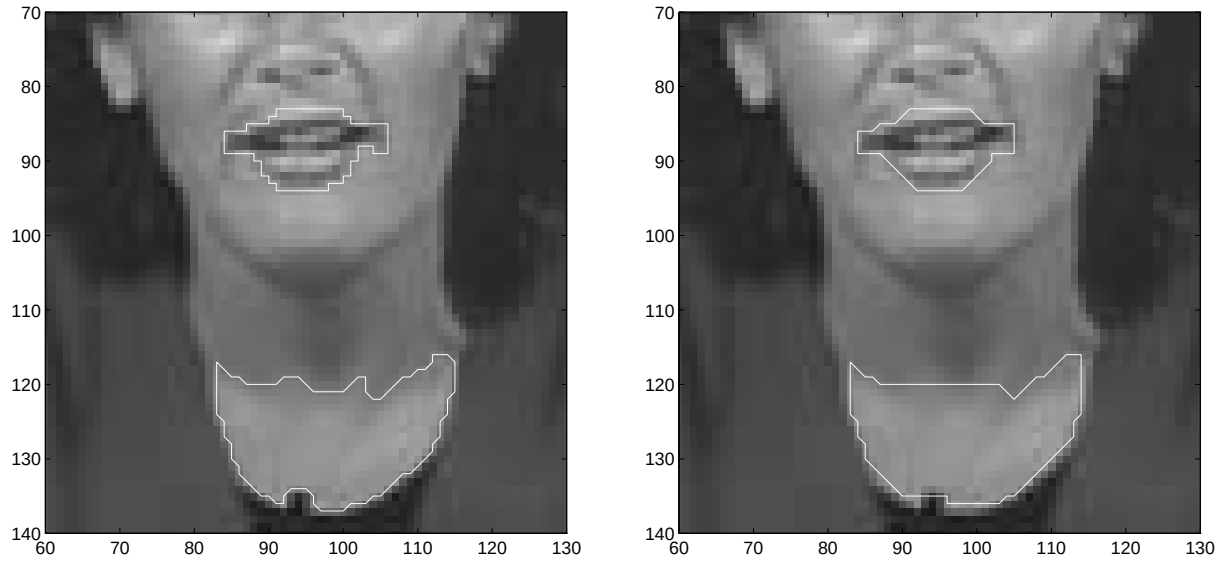


Figure 34: Left figure: original segmentation which requires 468 bits using the 8-connect chain code. Right figure: optimal segmentation with $D_{max} = 1$ pixel which requires a rate of 235 bits and results in a distortion of 1 pixel.

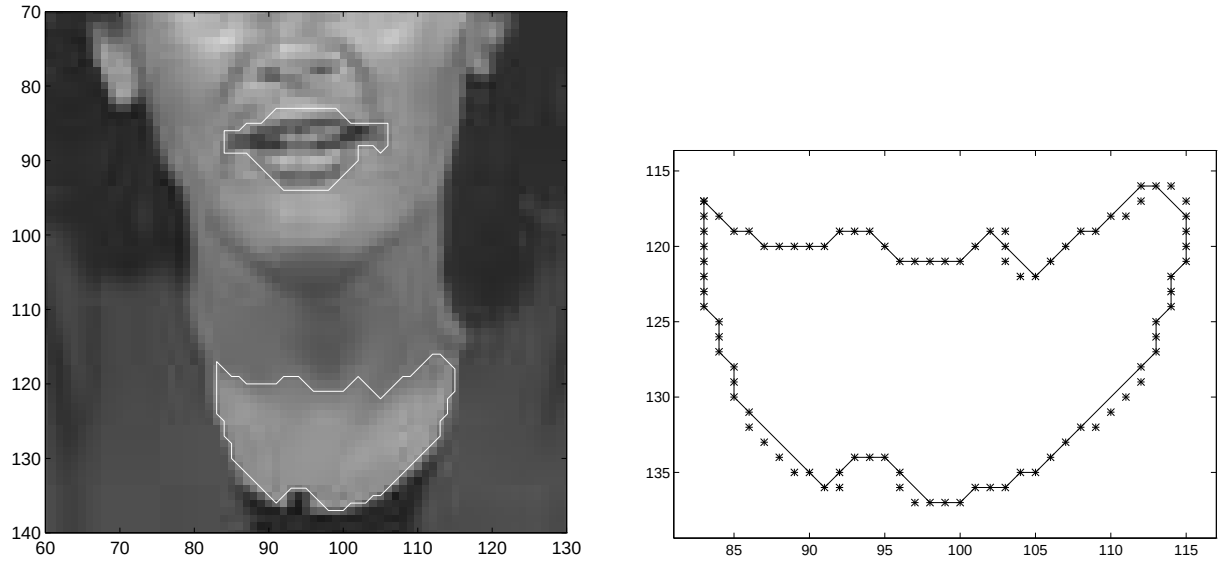


Figure 35: Left figure: optimal segmentation with $R_{max} = 280$ bits which results in a distortion of 0.71 pixels and a bit rate of 274 bits. Right figure: closeup of the lower boundary; the stars indicate the original boundary and the line represents the polygonal approximation. The upper left corner has been selected as the first vertex.

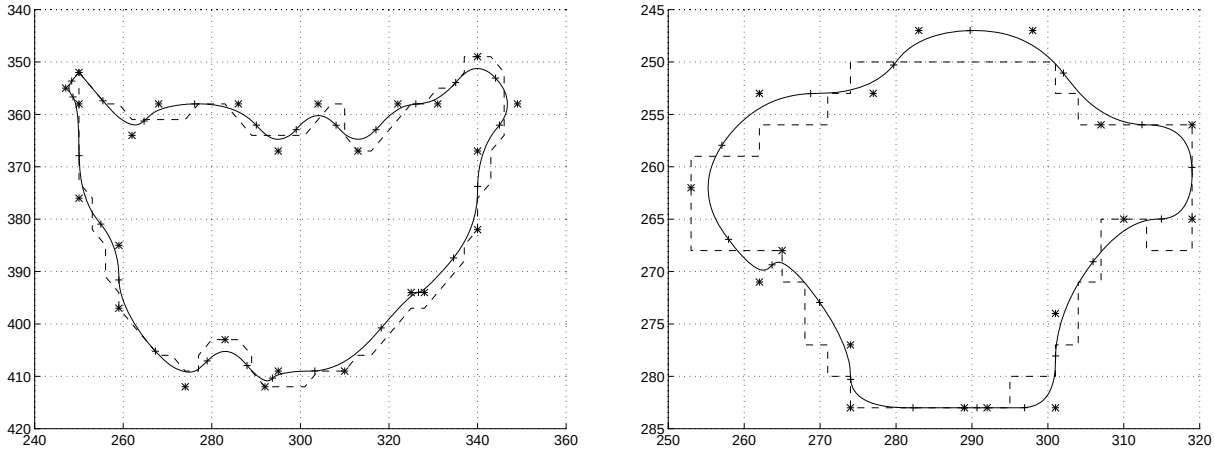


Figure 36: Results obtained using the B-spline approach. Dotted line: original boundary, continuous line: B-spline approximation, stars: control points

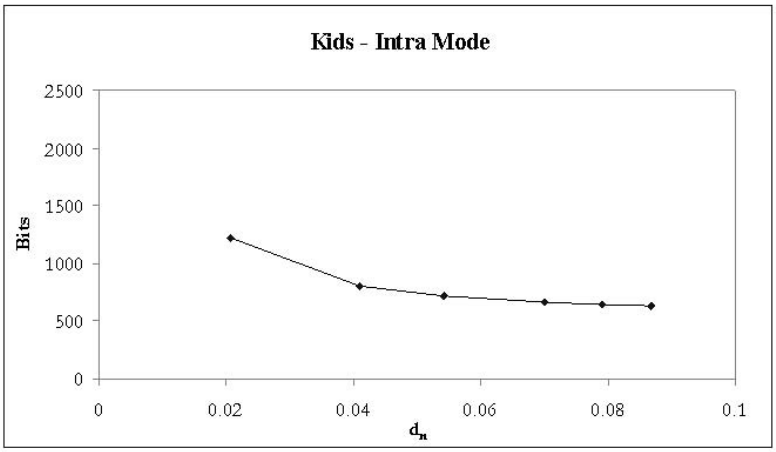


Figure 37: Rate distortion curve for the "Kids" sequence using the B-spline approach in the Intra mode. Bit rates and distortion averages are given for 100 frames.

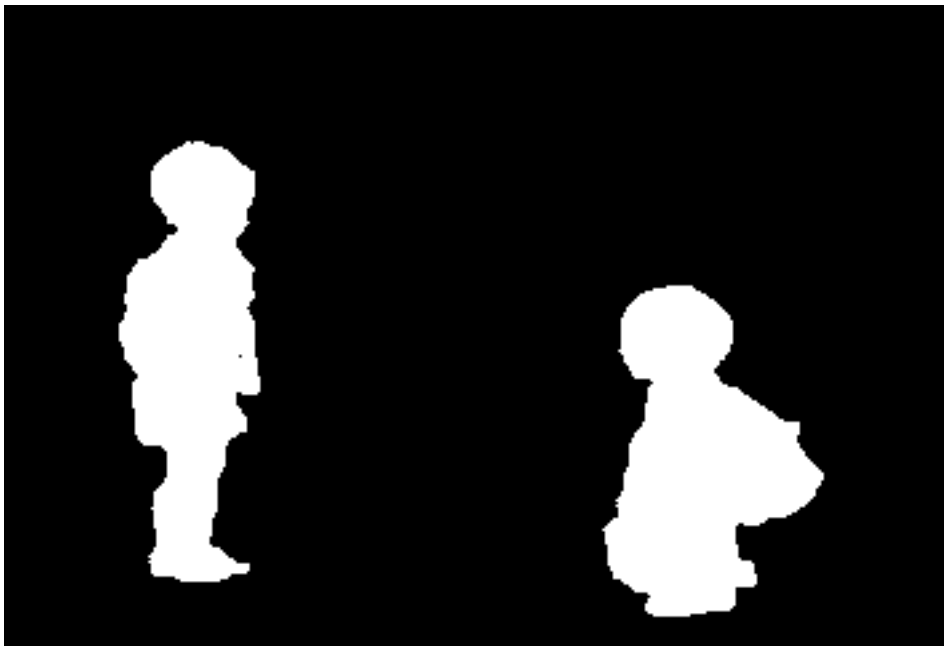


Figure 38: Frame 17 of the "Kids" sequence encoded using $D_{max} = 0.7$. 979 bits were required resulting in $d_n = 0.0171$.

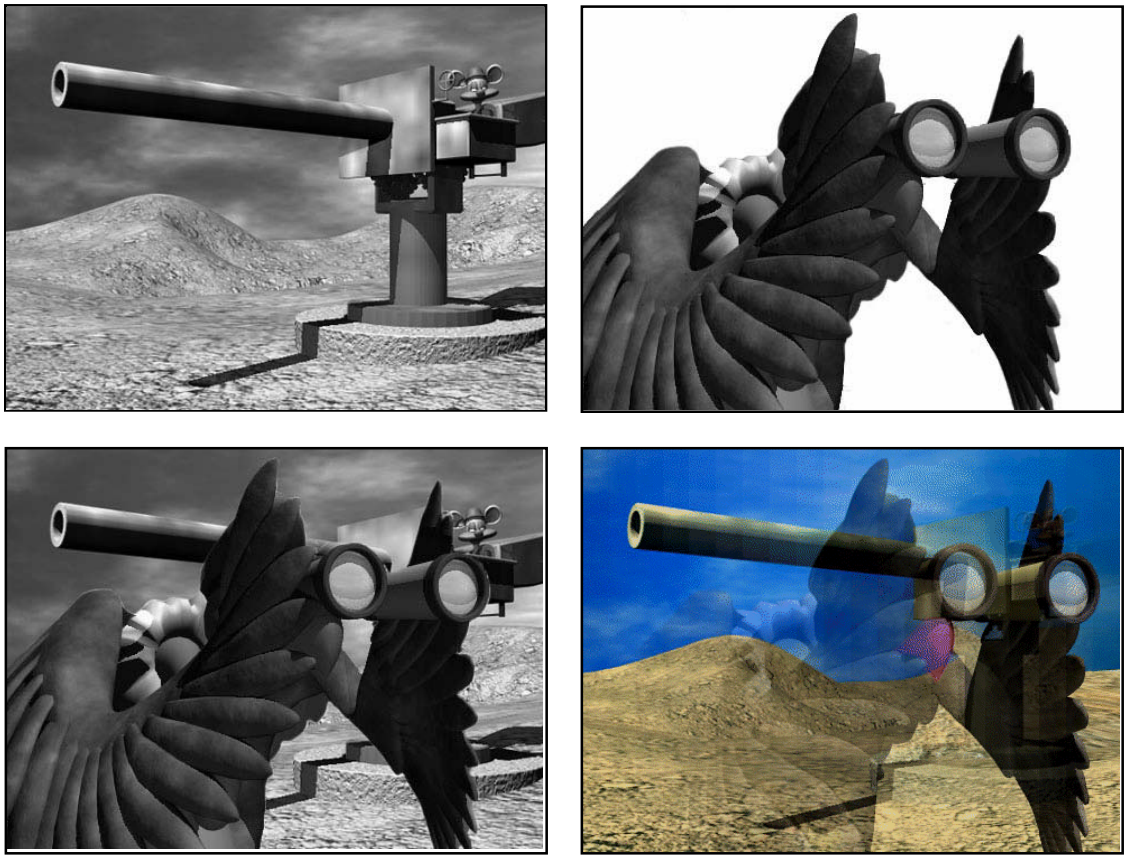


Figure 1: This image shows a scene composed of an object with constant and gray-scale transparency on a background.

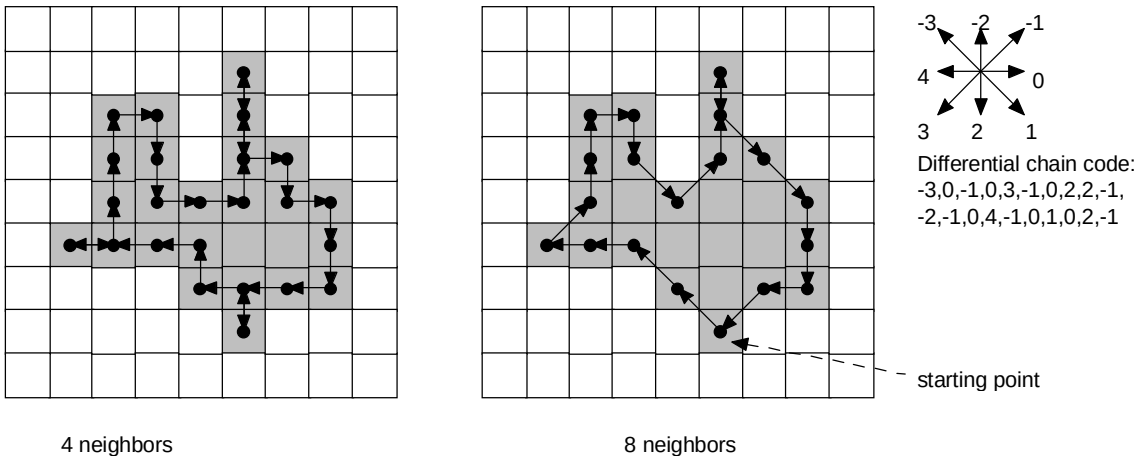


Figure 2: A chain code follows the contour of an object by describing the direction from one boundary pel to the next. Each arrow is represented by 1 out of 4 or 1 out of 8 symbols, respectively. On the right, the symbols for differentially encoding the shape are given.

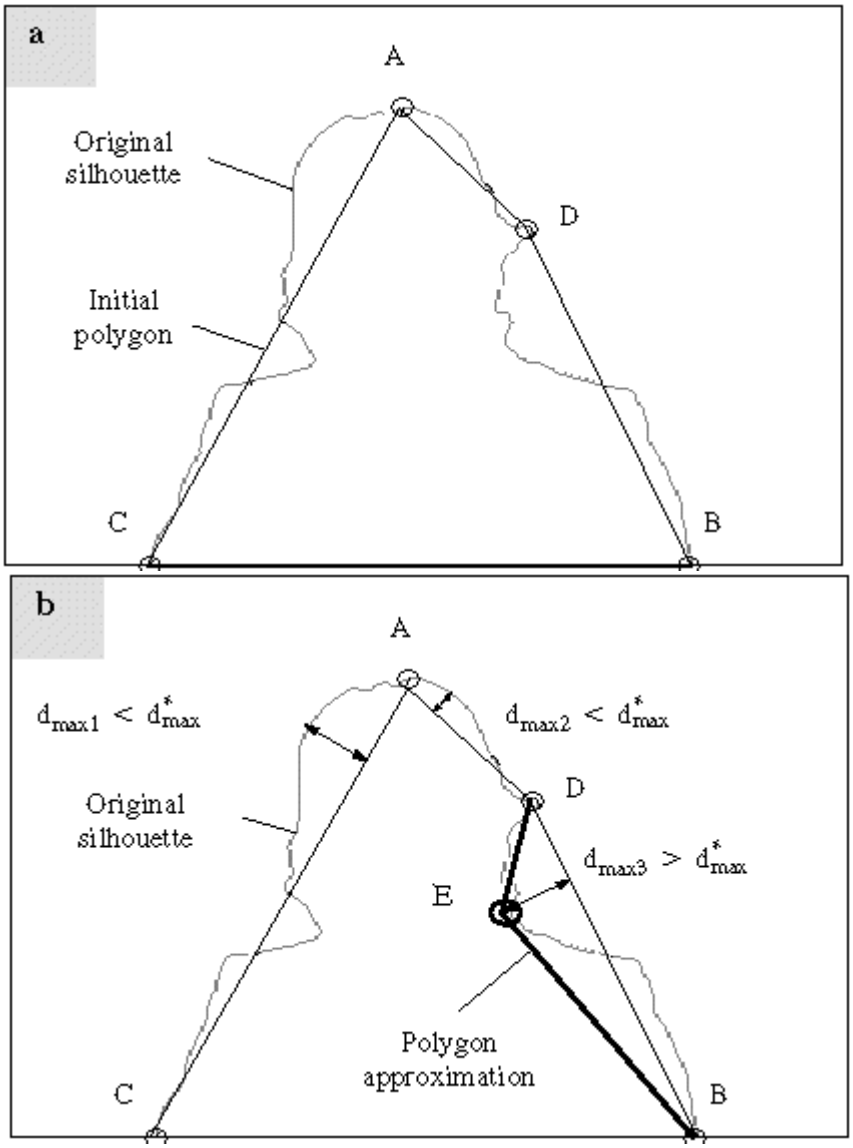


Figure 3: Successive polygon approximation of a contour (from [7]). The initial polygon AB is extended by points B and C. The iteration from a 4 point to a 5 point approximation is shown here.

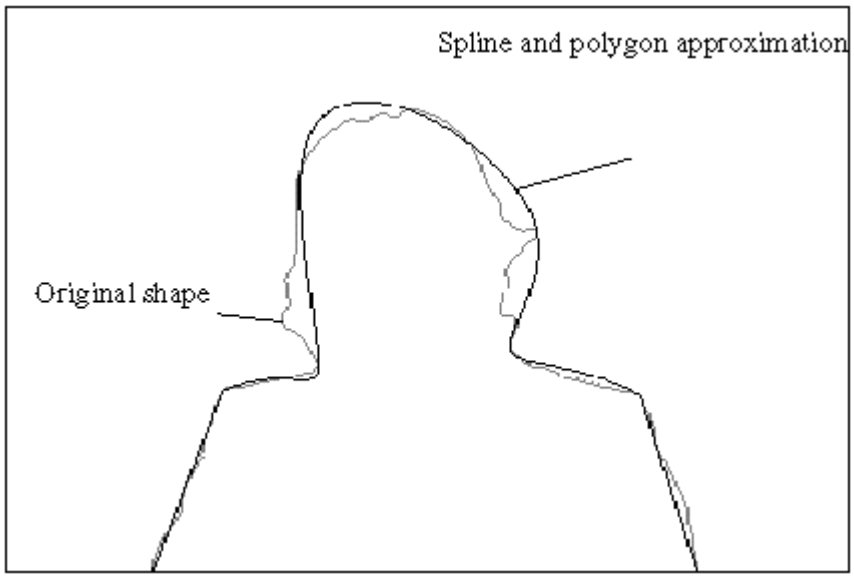


Figure 4: Approximation using a polygon/spline approximation (from [62]).

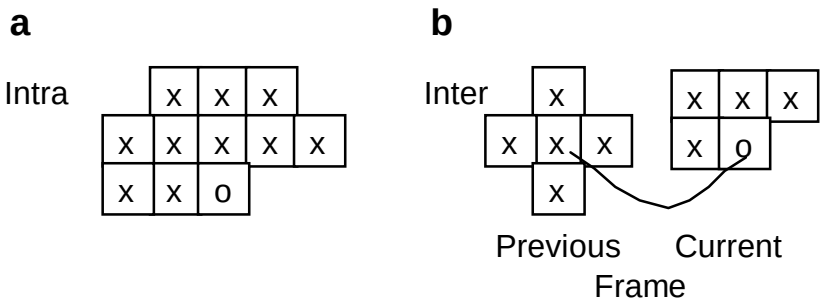


Figure 5: Templates for defining the context of the pel to be coded (o), a) defines the intra mode context, b) the context when coding in inter mode. The alignment is done after motion compensating the previous frame of the video object.

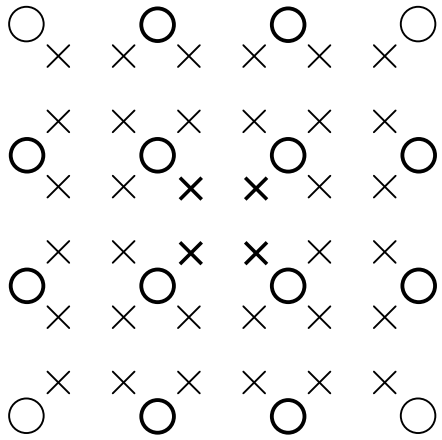


Figure 6: The upsampled pels (x) lie between the location of the subsampled pels (o). Neighboring pels (bold o) defining the values of the pels to be upsampled (bold x).

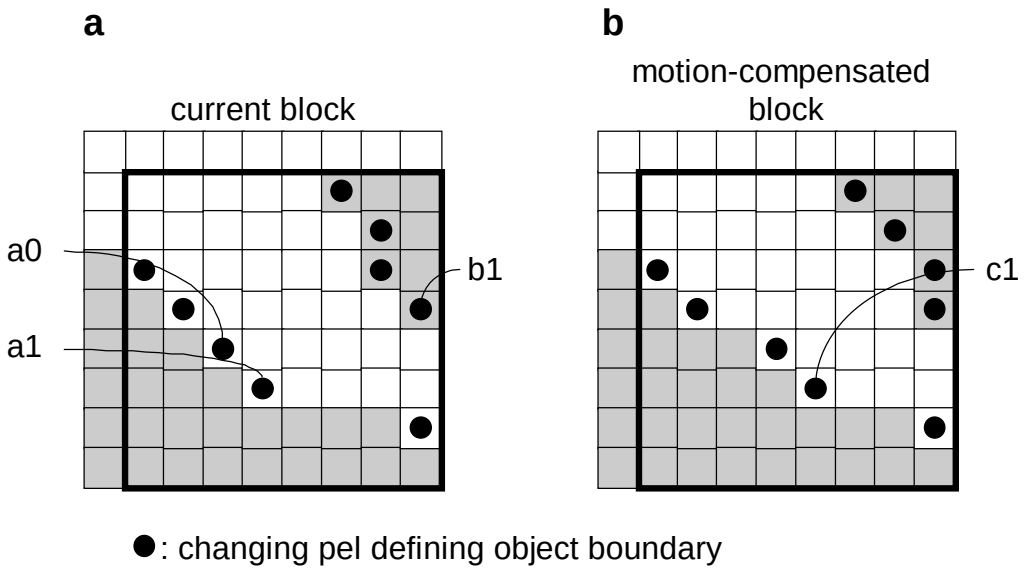


Figure 7: Changing pels are used in modified MMR shape coding to define object boundaries.

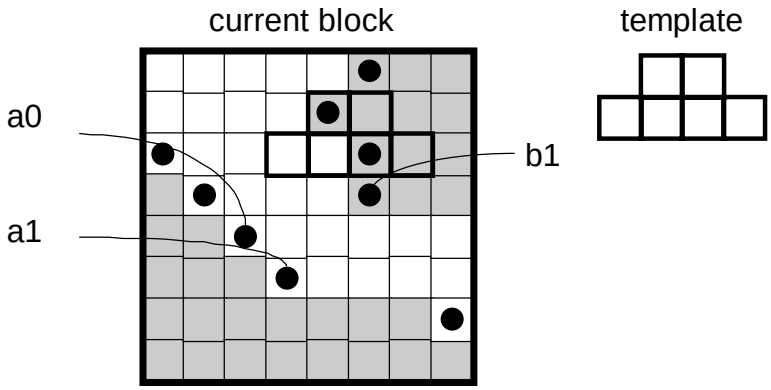


Figure 8: For Intra coding, a template positioned relative to *b1* is used for selecting VLC tables in Vertical Mode.

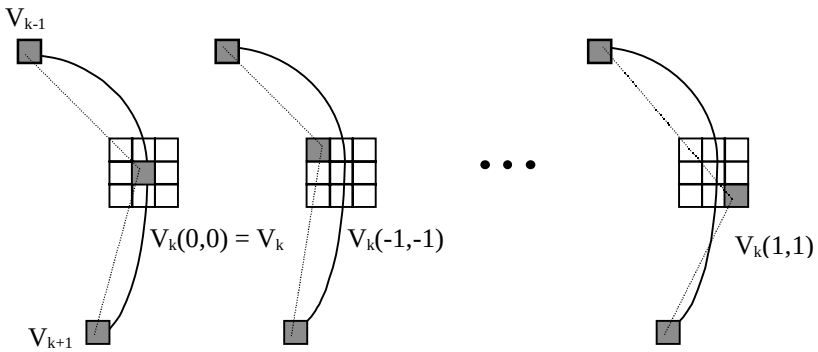


Figure 9: Example of vertex adjustment (from [61]).

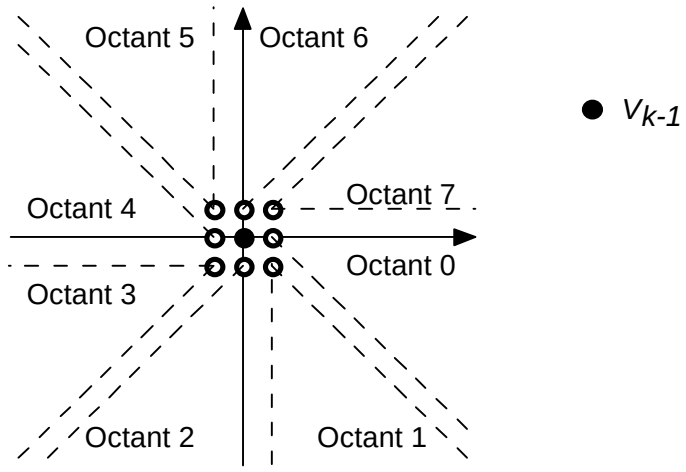


Figure 10: The difference V_d between V_{k-1} and the vertex V_k to be coded determines the octant where V_k is located.

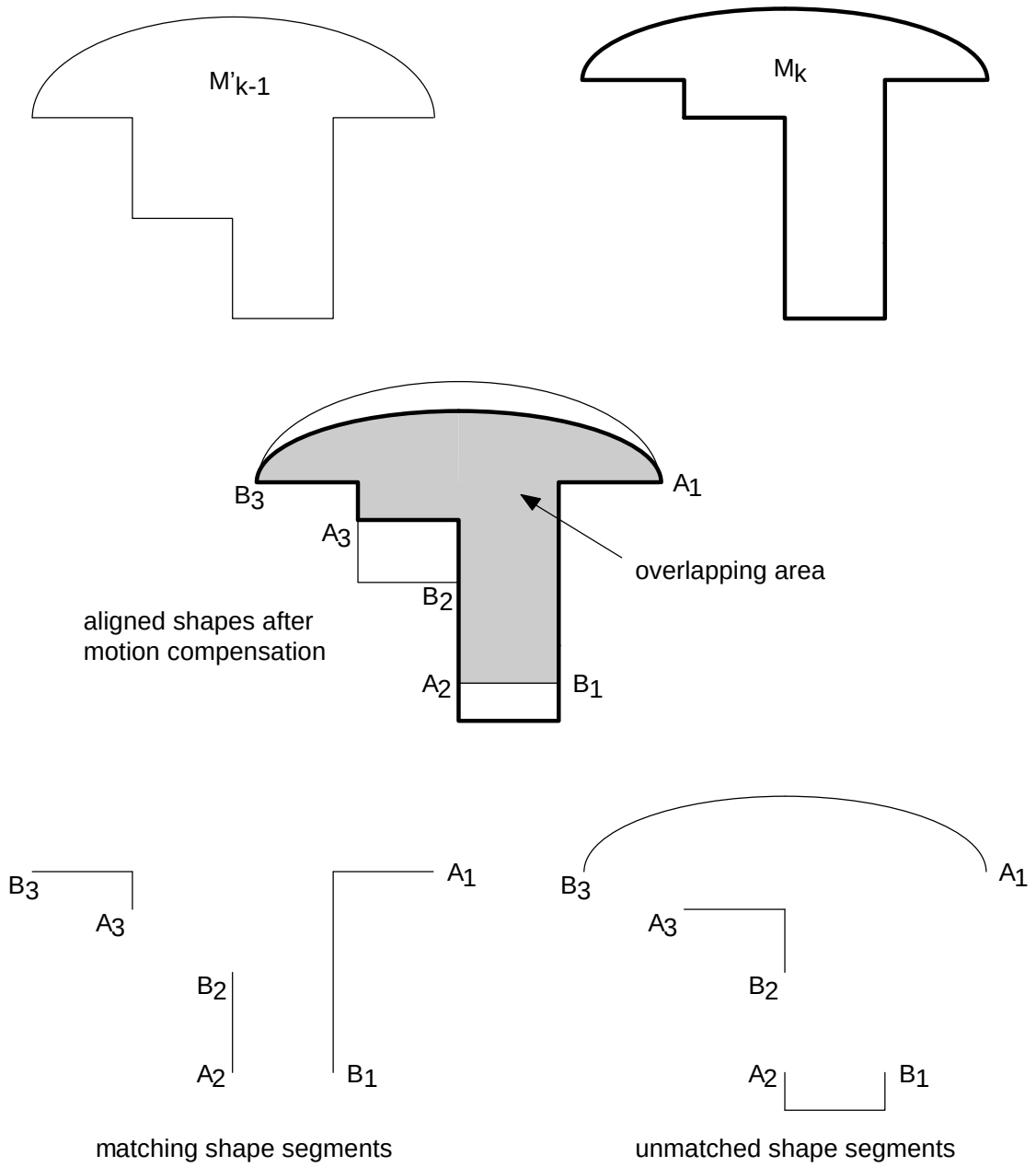
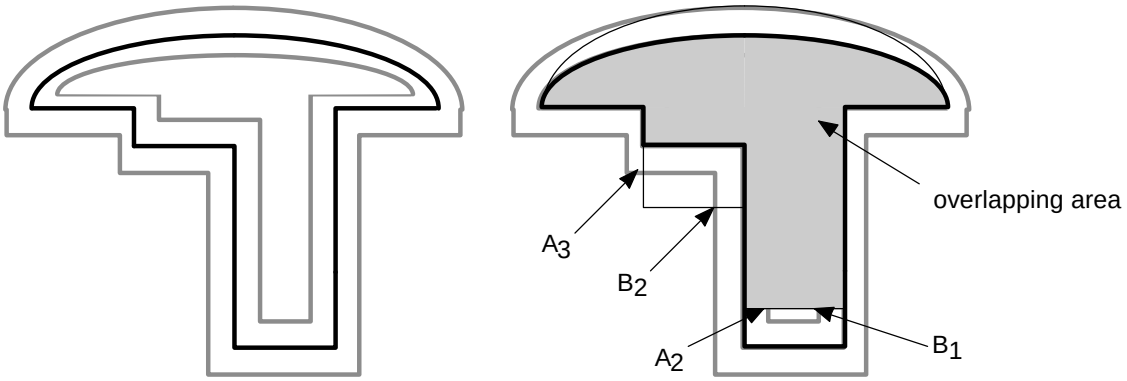


Figure 11: Vertex-based shape coding: Motion compensation aligns the previous and the current shape parameters. Unmatched segments of the contours are identified, here (B_1,A_2), (B_2,A_3), (B_3,A_1) for lossless shape coding.



current shape M_K with band indicating the allowable shape approximation error

aligned shapes after motion compensation

Figure 12: For lossy shape coding, a band with the allowable shape distortion reduces and shortens the unmatched segments, here (B_1,A_2) and (B_2,A_3) (compare with Figure 11).

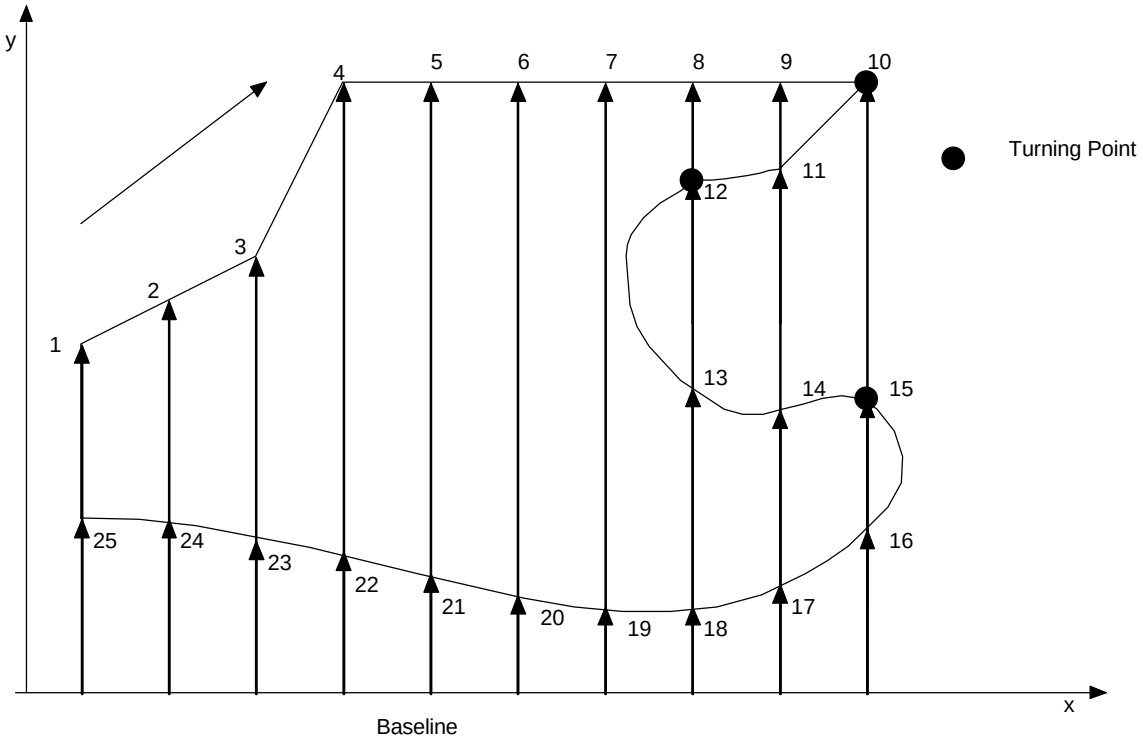


Figure 13: Baseline-based shape coder: Clockwise contour tracing from a baseline using turning points and the distance from the baseline to the contour points.

Figure 14: Three of the test sequences for evaluation of shape coders are (from left to right) Weather, Children, and Robot.

Figure 15: Lossy encoding using a bitmap-based (left) and a contour-based (right) shape coder. The bitmap-based shape coder used pel replication as the upsampling filter (from [64]).

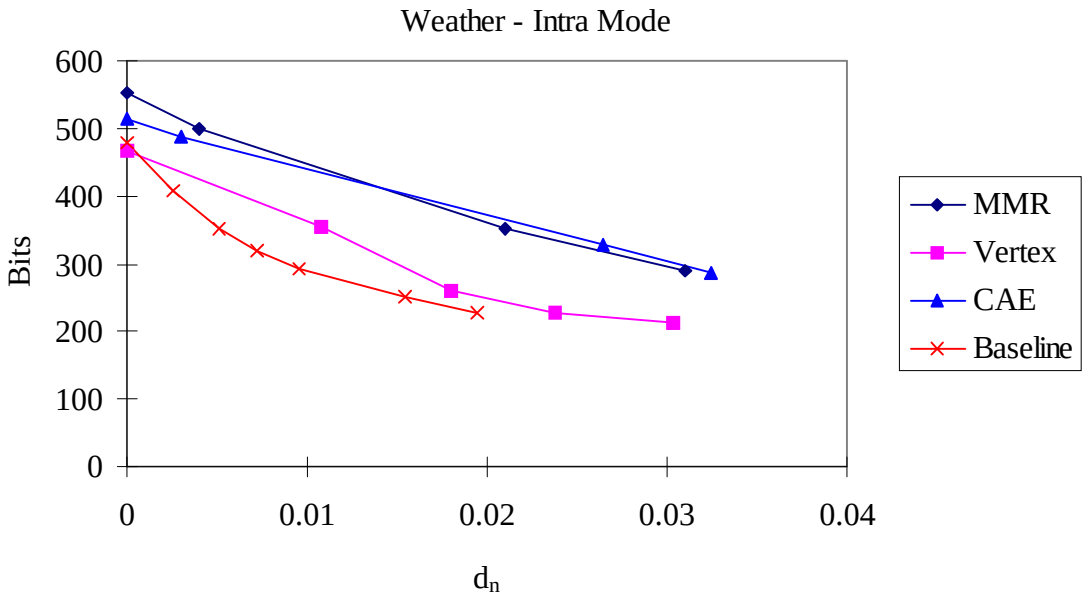


Figure 16: Comparison of rate distortion curves in Intra mode. Bitrate and distortion averages are given for 100 frames.

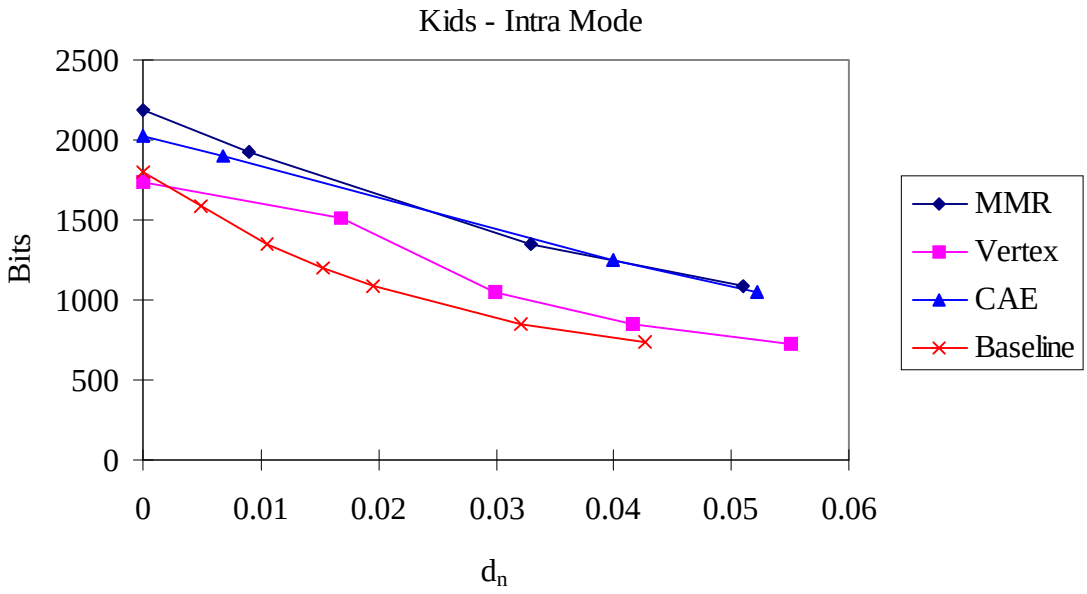


Figure 17: Comparison of rate distortion curves in Intra mode. Bitrate and distortion averages are given for 100 frames.

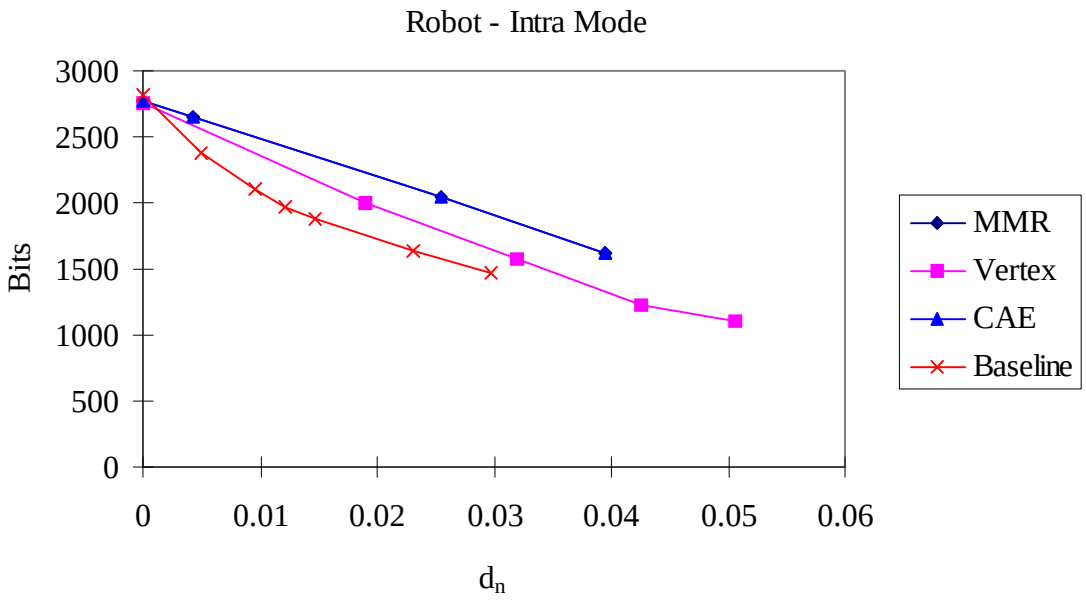


Figure 18: Comparison of rate distortion curves in Intra mode. Bitrate and distortion averages are given for 100 frames.

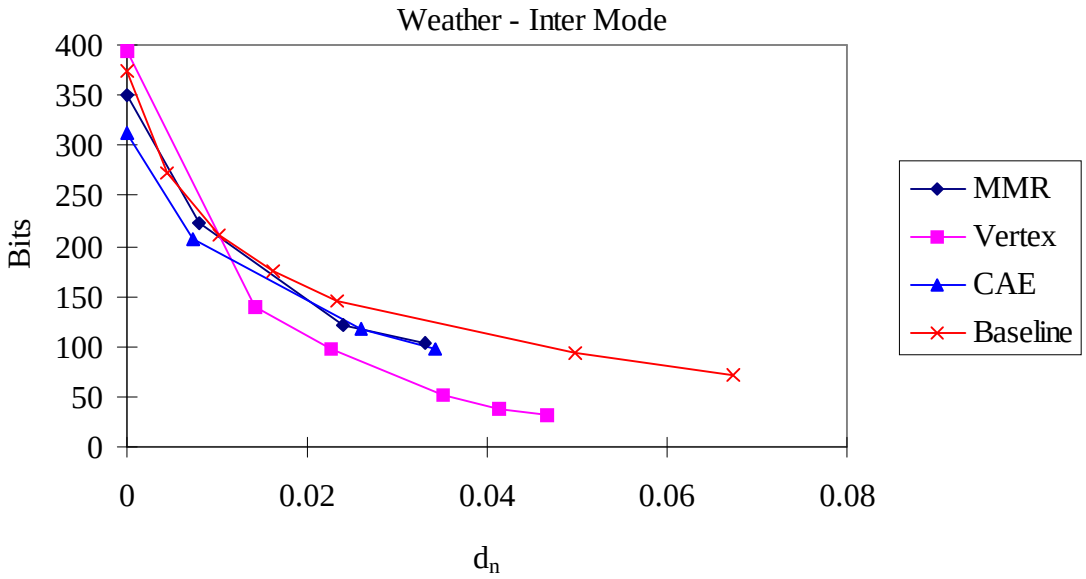


Figure 19: Comparison of rate distortion curves in Inter mode. Bitrate and distortion averages are given for 100 frames.

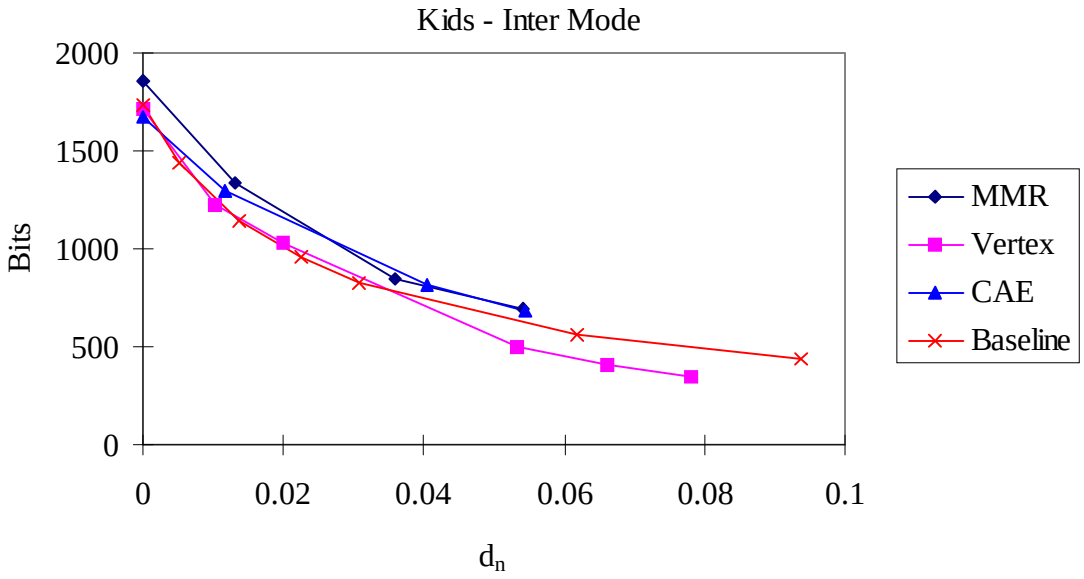


Figure 20: Comparison of rate distortion curves in Inter mode. Bitrate and distortion averages are given for 100 frames.

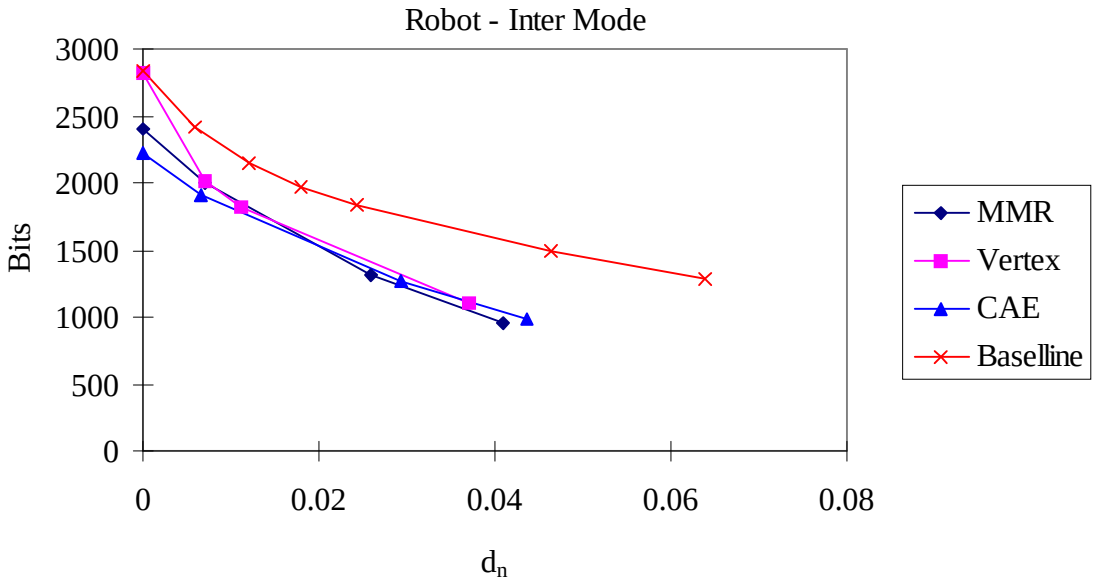


Figure 21: Comparison of rate distortion curves in Inter mode. Bitrate and distortion averages are given for 100 frames.

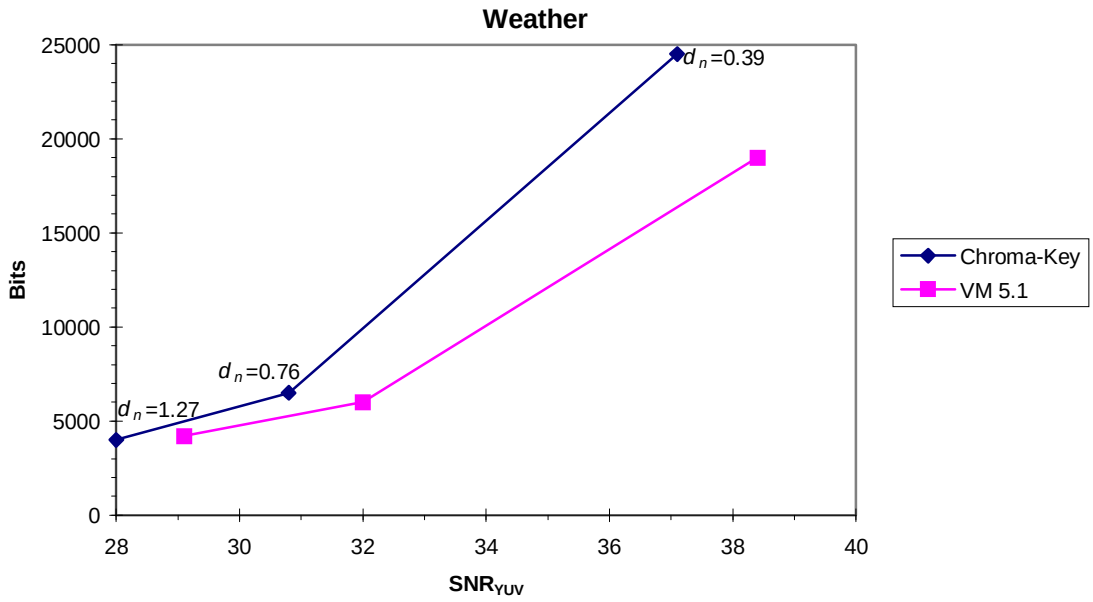


Figure 22: Comparison of chroma-key shape coder and MPEG-4 VM 5 with lossless shape coding (d_n=0.0). Quantizer stepsizes 8, 12 and 20 are used (from [63]).

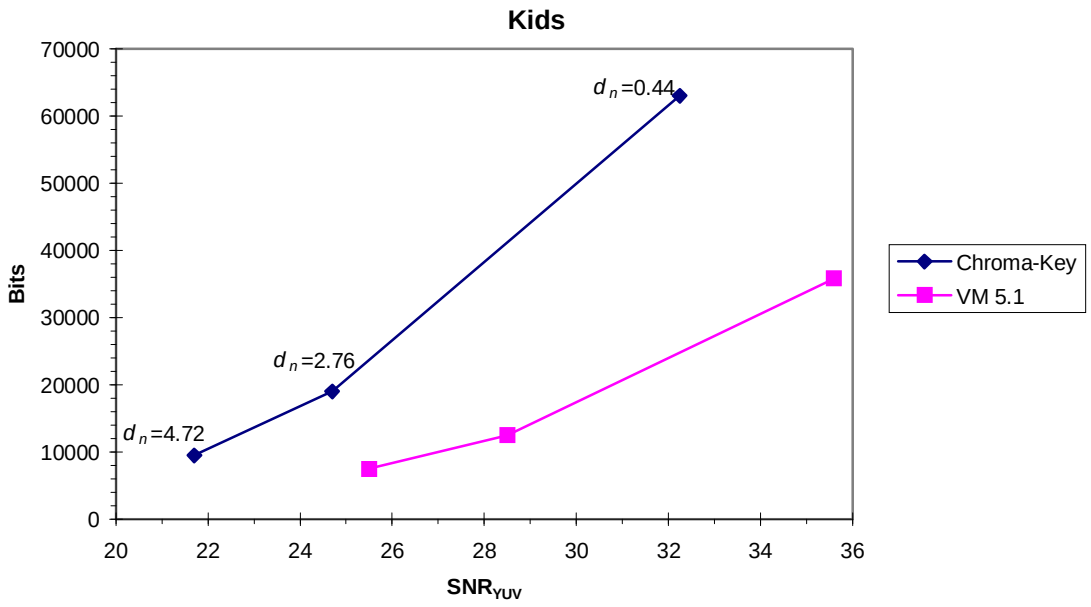


Figure 23: Comparison of chroma-key shape coder and MPEG-4 VM 5 with lossless shape coding (d_n=0.0). Quantizer stepsizes 8, 12 and 20 are used (from [63]).

Table 1: Comparison between the block-based CAE shape coder and the contour-based vertex-based shape coder based on seven test sequences. Relative statements are always with respect to the other shape coder. Please note that this is a snapshot as of April 1997.

	CAE	Vertex-based
Coding efficiency: Intra lossless		7.8% lower data rate
Coding efficiency: Inter lossless	20.5% lower data rate	
Coding efficiency: Inter lossy	better at small distortions	better at large distortions
Scalability overhead for 3 layers (layer 3 lossless)	30-50% higher data rate for shape coding in inter mode	no optimized results for inter coding.
Delay	slightly shorter due to block-based coding	
Hardware Implementation Complexity	decoding on chip without random access to external memory	Huffman decoder smaller than arithmetic decoder, random access to external memory
Software Implementation Complexity	No optimized coder available, but similar performance for non-optimized code.	