HYBRID COMPRESSION OF VIDEO WITH GRAPHICS IN DTV COMMUNICATION SYSTEMS

Mihaela v.d. Schaar-Mitrea[1] and Peter H. N. de With[2]

Abstract— **Advanced broadcast manipulation of TV sequences and enhanced user interfaces for TV systems have resulted in an increased amount of pre- and post-editing of video sequences, where graphical information is inserted. However, in the current broadcasting chain, there are no provisions for enabling an efficient transmission/storage of these mixed video and graphics signals and, at this emerging stage of DTV systems, introducing new standards is not desired. Nevertheless, in the professional video communication chain between content provider and broadcaster and locally, in the DTV receiver, proprietary video-graphics compression schemes can be used to enable more efficient transmission/storage of mixed video and graphics signals. For example, in the DTV receiver case, this will lead to a significant memory-cost reduction. To preserve a high overall image quality, the video and graphics data require independent coding systems, matched with their specific visual and statistical properties. In this paper, we introduce various efficient algorithms that support both the lossless (contour, runlength and arithmetic coding) and the lossy (block predictive coding) compression of graphics data. If the graphics data are *a-priori* mixed with video and the graphics position is unknown at compression time, an accurate detection mechanism is applied to distinguish the two signals, such that independent coding algorithms can be employed for each data-type. In the DTV memory-reduction scenario, an overall bit-rate control completes the system, ensuring a fixed compression factor of 2–3 per frame without sacrificing the quality of the graphics.**

Keywords— **TV communication systems, lossless image compression, graphics, contour coding, arithmetic coding, runlength coding, embedded compression architecture.**

I. Introduction

New technologies and standards for Teletext and On-Screen-Display (OSD) menus in TV systems are emerging [1], resulting in increasingly diverse TV images in which video sequences are mixed with various types of graphical information (see Figure 1). Simultaneously, Digital Video Broadcasting (DVB) based on MPEG compression [2] and digital camcording (e.g. digital Betacam, DV compression [3]) are introduced in professional and consumer TV applications. MPEG-2 has currently become the *de facto* standard for digital video transmission and compression for standard- and high-resolution TV. However, the introduction of graphics is only partially covered in the MPEG-2 standard. Although the standard allows the communication of additional data in a separate channel, there are no facilities for an efficient transmission of graphics data or for a joint graphics-video signal. For the efficient transmission or storage of video and graphics, two alternative systems can be adopted: an architecture where the video and graphics are coded separately, using optimized compression systems, or an architecture where the video and graphics are jointly coded. Effective joint coding would combine video and graphics into one signal representation, while preserving the image quality of both signal-types. However, a more efficient transmission of graphics in coded form between the broadcaster and the DTV-receiver would require the standardization of the adopted compression technique.

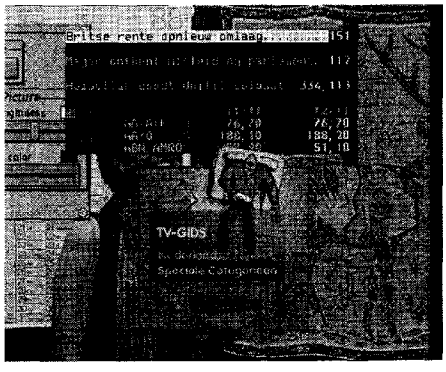


Fig. 1. Example of an image containing video mixed with graphics.

Two other important trends are significant for our study and they do not require a standardization of the employed coding algorithms. Firstly, the professional video communication chain from content provider to broadcaster is also becoming digital, and this chain could benefit from the more efficient transmission of video and graphics. Secondly, the gradual transition to digital equipment is also taking place locally at the broadcaster and the DTV-receiver side. Therefore, the efficient storage of video and graphics signals becomes important also for these systems. It is relevant to notice that for these architectures - professional communication chain and local storage at content provider, broadcaster and receiver - no standardized compression algorithms are required, thereby allowing for differentiation between the various manufacturers. For example, recently, several algorithms have been proposed for reducing the video storage of the DTV-receivers [15] [16] [14] [13]. How-

1 Mihaela van der Schaar-Mitrea is with Philips Research Labs, Briarcliff-Manor, USA. E-mail: Mihaela.vanderSchaar@philips.com

2 Peter H.N. de With is with the University of Mannheim, Faculty of Computer Engineering, Mannheim, Germany. E-mail: dewith@ti.uni-mannheim.de. Per August 2000, he is with CMG Netherlands and the University of Technology Eindhoven.

Original manuscript received June 28, 2000

ever, there are currently no algorithms available in the literature that target the compression of *both* the video and graphics data for these applications.

The MPEG-2 standard (or other standardized video coding standards) cannot be employed for the compression of this combined signal, because it relies heavily on the average moving-video statistics to obtain a high performance. Experiments with the MPEG-2 compression of graphics revealed dissatisfactory performances, because e.g. edges of objects are not well preserved. Furthermore, the compression factors required for the local storage or transmission between content provider and broadcaster are limited (e.g. 2–4), such that using an MPEG-2 compliant codec for this purpose would result in an unnecessary high system costs.

The MPEG-4 standard [17] does provide tools for the compression of both video and graphics. However, the MPEG-4 standard is not used for DTV broadcasting. Furthermore, the graphics/texture compression technique targets especially 3-D graphics, and is therefore not optimized for coding graphics data like Teletext and OSD menus. Also, like in the MPEG-2 case, the costs associated with applying MPEG-4 for the sole purpose of reducing DTV receiver costs or optimizing the professional video communication at the broadcaster side, are not justifiable. Additionally, there are no standardized provisions for detecting the graphics information if this is mixed in advance with the video signal, such that the high-quality compression of the joint video-graphics signal cannot be guaranteed.

In this paper, our aim is to design a cost-effective high-quality compression system for the efficient transmission and local storage of images containing mixed video and graphics. Section II describes the architecture of the professional video/graphics communication chain between content provider and broadcaster and the architecture of the DTV receiver with efficient memory usage. Section III introduces a block predictive coding technique for the compression of video and certain types of graphics. In Section IV, the signal characteristics are presented together with the compression requirements for the various types of graphics. In Section V, different novel algorithms for lossless graphics compression are introduced. Section VI discusses a graphics detection system and a classification technique for enabling quality control of video and graphics coding within mixed images. An efficient system based on arithmetic coding for lossless compression of graphics that matches with the classification is also presented within this section. Section VII outlines the concept of a bit-rate regulation, controlling lossless coding of graphics and the lossy coding of video. Section VIII-A shows the simulation results obtained, followed by the conclusions, in Section IX.

II. System architecture

In this section, two possible architectures are described for the transmission and storage of mixed video and graphics signals. The first achitecture aims at the professional communication chain between the content provider and broadcaster (see Figure 2 a). The second architecture is for reducing the storage costs at the DTV receiver side (see Figure 2 b). Both previously mentioned systems have the following common properties: proprietary systems can be used for the compression of the video and graphics signals, and the image quality after decoding must be near-lossless to avoid interference with the transmission channel standard.

Architecture 1. Professional transmission/storage between content-provider and broadcaster.

At the content provider side, the capturing of the video data and the generation of the corresponding graphics are separate processes. To enable the efficient storage of the video and graphics data prior to their transmission to the broadcaster, compression can be employed. This efficient storage is beneficial even in the professional domain, since the same scene of e.g. a sport event is often captured from various viewing angles. However, the efficient compression of video and graphics by the content provider is especially desired for the transmission of "live" events, when several streams are transmitted simultaneously to the broadcaster. The most important requirement for the coding techniques employed in this architecture, is that their quality should be extremely high, such that no artifacts are visible. Moreover, the compression systems employed in this stage should not influence in any way the transmission chain between broadcaster and consumer, independent whether DVB or analog broadcasting is used. To preserve a high-image quality of both video and graphics, independent compression algorithms should be employed for these two data types. For example, the video data can be efficiently stored employing a near-lossy compression technique (see Section III). For the graphics data, depending on its characteristics (see Section IV), lossless compression techniques should be employed to preserve its quality. In Section V, several novel compression schemes are introduced for the lossless compression of graphics. Since the algorithms used for graphics compression act independently of those for video, a relatively simple (but not necessarily low-cost) system architecture results (see Figure 2 a), because the different coding algorithms can be optimized for the individual types of data and the coded streams can be separately transmitted. Furthermore, there is no fixed compression factor that needs to be guaranteed for the coding of the video or graphics data, such that variable bit-rate streams can be generated.

Architecture 2. Local storage at DTV receiver side.

The second architecture aims at reducing the local memory costs of the DTV receiver, which represents a significant part of the overall receiver costs. This local receiver memory is used for the storage of video images for digital signal processing, local graphics generation, etc. In this paper, the architecture proposed for the DTV receiver is based on the mixed storage of the video-graphics signal. This solution is cost-efficient, since the video segments overlapped by graphics data do not need to be locally stored, thus saving both memory and memory bandwidth at the receiver. Furthermore, only one *shared* memory is required for the storage of the two data types. Thus, if no graphics are generated or transmitted at a certain moment, the entire memory can be used for the video data processing. However, even though the two signals are stored in a combined fashion, different compression techniques should be used for the video and graphics data, to maintain a high-image quality.

If MPEG-2 has been used for transmission of the mixed

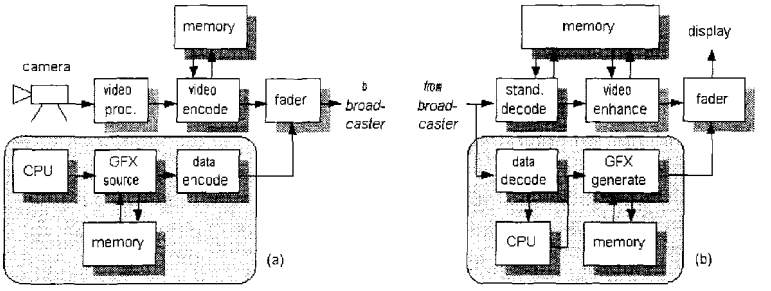


Fig. 2. Block diagram of a TV communication chain, where graphics data are inserted both at a) the content-provider and b) the the TV receiver side.

video-graphics signal, the quality of the graphics data is already diminished (as already mentioned), such that employing different coding algorithms for video and graphics is no longer necessary. Nevertheless, there are cases where an *a-priori* graphics detection algorithm is required to locate the graphics content (see Section VI). For example, such a case occurs in a DTV receiver's image post-processing IC, where video and graphics need to be combined and stored in local memory. After graphics detection, two coding systems are active in parallel (for video and for graphics). To reduce the memory size of the receiver, a fixed memory-reduction factor needs to be guaranteed. Therefore, a bit-rate controller has been designed, which considers the different signal statistics of video and graphics and switches automatically between a lossy and a lossless coding scheme for video and graphics, respectively (see Section VII). If the video and graphics data were only mixed at the receiver side, then the predictive coding techniques of Section III can be succesfully employed, while the lossless compression techniques of Section V can be adopted for the graphics data.

This brief analysis of the two architectures reveals that each system has its own specific problem statement. For the first architecture, we focus on the efficient high-quality compression of graphics (Section V). For the second architecture, several issues are important: the graphics detection prior to receiver processing, and the efficient storage of the mixed signal required for this postprocessing (Sections VI and VII). A common feature is that both systems aim at realizing a compression system that preserves a very high image quality.

III. Block Predictive Coding (BPC)

This section describes briefly the predictive coding technique adopted for video and graphics compression. In [14], a "near-lossless" Block Predictive Coding was proposed for embedded memory compression in DTV receivers. This method could also be successfully used for compressing the video data distributed by the content provider or for the local storage of video data at the content-provider or broadcaster. Due to the high image quality that can be obtained by this coding system at modest compression factors (i.e. 2–3), no interference occurs with the MPEG-2 compression system used by the broadcasters elsewhere in the transmission chain. Moreover, this coding technique allows easy access to the individual pixels in the compressed domain for further editing or post-processing, a feature which is not easily realized with e.g. a transform coder. A third advantage of this system is that it can be implemented with low complexity in hardware.

The block predictive coding is based on Adaptive Dynamic Range Coding (ADRC) [4], which was developed initially for digital video recording. The ADRC data compression technique can be seen as an example of a two-dimensional predictive coding system.

The first step of the predictive coder is the partitioning of the image in small subblocks of 4×4 pixels. Subsequently, the maximum and minimum pixel values of each block are determined. The difference between the minimum and the maximum sample value in a block is called Dynamic Range (DR). The minimum sample value (MIN) is transmitted as a global prediction for all other samples inside the block. After subtracting this minimum value from the actual sample value $s(i,j)$, the difference values $d(i,j)$ of a block are quantized according to the dynamic range, resulting in $dq(i,j)$.

Figure 3(a) shows the general block diagram of the encoder. Three parameters are required for decoding: the block prediction MIN, the dynamic range DR for selecting the correct quantizer table, and the quantized differences $dq(i,j)$.

Feedforward coding [5] was adopted to achieve a fixed compression factor on small groups of blocks (segments) (see Figure 3(b)). The video data are first analysed with a set of various quantization strategies. The strategy (S) chosen after analysis is the finest quantization that yields the desired number of bits reserved for a segment. The coding procedure described is performed on a segment basis, where each segment is compressed individually. In the experiments, horizontal stripes of blocks were used as segments, and coding was performed from the top to the bottom of the image. The coding process is defined formally by:

$$d(i,j) = s(i,j) - MIN, \; i,j \in Block, \tag{1}$$

$$dq(i,j) = Q(d(i,j), DR, S). \tag{2}$$

The aforementioned technique is adequate for the compression of video data with very high quality [14], but it is not suitable for the compression of all various graphics types, due to their different statistics and image perception. In the next section, the graphics characteristics are described in more detail.

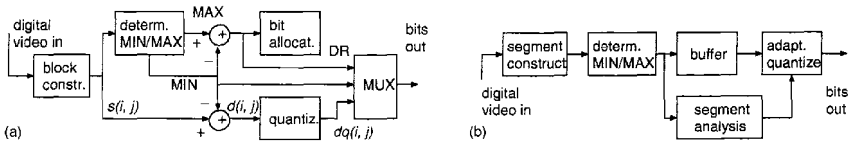


Fig. 3. (a) Block diagram of BPC system encoder and (b) feedforward coding system based on BPC.

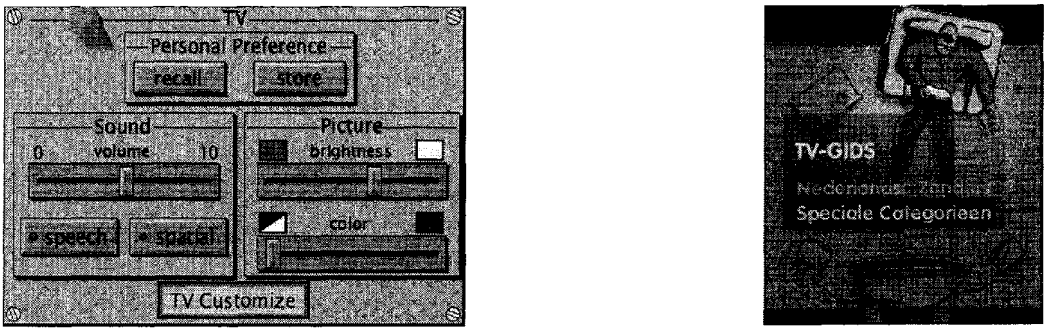


Fig. 4. Examples of different types of graphics which can be mixed with the video data. (a) (left) Typical OS Display user menu. (b) (right) Pixel-based graphics, containing Dutch text.

IV. Properties of Graphics Signals Inserted in TV Images

The graphics elements that are mixed with video in the TV environment can be divided in two classes, depending on their visual and statistical characteristics. The first class includes graphics like Teletext, subtitling and computer menus (see Figure 4(a)) and it is characterized by abrupt changes in the sample values. The second class is formed of graphics resembling video, like pixel-based graphics (see Figure 4(b)). It is obvious that an efficient code for one of the previous two classes of graphics blocks is not acceptable also for the other class, due to the relatively large difference between the signal statistics of the two types of data.

Class 1. Text and menus

The main problem of coding the first class of graphics is that it results in visible colour blurring, which occurs if the same lossy compression technique as for video is adopted. For example, the quality of the subtitling is considerably reduced if the aforementioned BPC technique is employed for compression, as can be noticed from Figure 5. Firstly, at the left side of the displayed character, coding noise is well visible. Secondly, the dark shade surrounding the character is erroneously quantized at the left side. It is clear that such distortions are not acceptable for a high-quality receiver, and thus, coding algorithms should be employed which yield no visible artifacts for the graphics data. Another disadvantage resides in the reduced compression factors obtained by the video coding algorithms for graphics data, because they are not able to exploit the pixel uniformity of the graphical elements. Lossless coding algorithms, matched to the graphics data statistics, gain compression factors five times as large. As a consequence, lossless coding techniques matching the signal statistics should be employed for an efficient high-quality compression of the first graphics class (see Section V).

Class 2. Pixel-based graphics

The second class of graphics (i.e. pixel-based graphics) resembles both the visual and statistical characteristics of video. As a consequence, the BPC system encodes the pixel-based graphics with smaller quantization errors than the graphics in the first class, since the dynamic range is relatively small. Therefore, BPC is a suitable candidate for the compression of this graphics class.

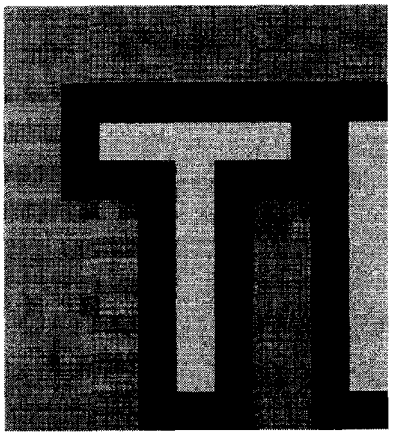


Fig. 5. Artifacts occurring when subtitling is compressed with a lossy coding algorithm.

V. Lossless Coding of Graphics

For the architecture based on separate transmission or storage of video and graphics, lossless coding is investigated (arguments in Section IV). In this section, we evaluate two lossless coding algorithms for the compression of graphics data like Teletext and computer menus: contour and run-length coding. Both algorithms are very efficient in compressing large areas having equal pixel values. For more algorithmic details, the interested reader is referred to [6].

In Section VIII-A a comparison with BPC coding of graphics will also be presented.

A. Contour coding

A digital image may be viewed as a function of two variables, namely (pixel) location (i.e. spatial coordinates), and colour. Using a finite number of colours, an image can be visualized as a number of plateaus with the plateau height equal to the colour level. A large area of pixels with the same colour would produce a large plateau and vice versa. The contour algorithm presented in this section reduces an image to a list of contours of plateaus. Each contour is uniquely determined by specifying:

1. its *colour*;
2. the *location* of upper leftmost pixel on its boundary, called the Initial Point (IP);
3. a *sequence of directions* (N=North, E=East, S=South, W=West) when tracing the boundary of the contour.

Thus, an image I can be represented as the sum of all contours (C) together with their corresponding colours:

$$I = \sum_{C} (colours + \mathrm{IP} + directions). \qquad (3)$$

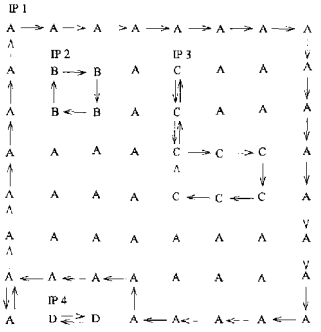


Fig. 6. Contour coding example of a block with 4 different colours.

Example 1: In order to illustrate the coding process, Figure 6 portrays an image composed of four different coloured plateaus - A, B, C and D. Each of these four plateaus can be represented by its colour, start position and boundary directions. For example, the third contour is represented by the colour C, the start position in row 2, column 5 and the set of directions describing its boundary: S, S, E, E, S, W, W, N, N, N. □

In [6] different algorithms have been investigated in order to adapt the contour coding to the characteristics of the graphics data. The compression performances achieved by these algorithms are limited by the image segmentation in blocks of dimension 4 × 4, 8 × 8 and 16 × 16. Consequently, the algorithms are searching for contours with the same colour within the block size specified. The "colour" represents in our case the value of the luminance signal Y, chrominance signal UV, R, G or B signals, each with values between 0 and 255. A series of novel extensions have been added to improve the performance of the fundamental contour coding algorithm:

1. **Implicit encoding of the longest "outer" contour.** In order to reduce the number of bits required for the representation of the contours inside a block, an algorithm was designed which searches the longest "outer" contour and encodes it by simply sending the value of the "colour" in that contour. An "outer" contour is defined as a contour that is not contained within other contours.
2. **Coding of the contours colours.** The compression improves by Huffman coding the "colours" of the contours. The statistics of various "colour" values in different textual graphics regions were measured and consequently, a fixed Huffman code was designed. An example of the bit assignment for the case of the luminance signal is given in Figure 7(a). The compression achievement of this algorithmic extension is considerable (around 20 %), because it exploits the prediction for a typical colour setting of broadcasted TXT pages or a manufacturer setting of On Screen Display menus. To further improve the compression factor, the coherency of the contours which can be split across a certain number of adjacent blocks was exploited. In this case, most of the colours used to describe the contours within the previous block will also be applied inside the current block. This redundancy can be removed by encoding the "colours" of the contours, depending on those of the previous block.
3. **Relative offset encoding for initial points.** For an improved compression, the initial point (IP) of a contour can be represented as an offset of the IP with respect to the previous one. The offsets are Huffman coded with a fixed table (see [6]).
4. **Coding of directions.** The directions were coded using a fixed Markov model, illustrated in Figure 7(b). When using this model, we encode the direction out of a pixel conditioned to the direction of coming to that pixel. By statistical measurements, it was verified that the highest probability occurs in the case that the new direction is equal to the previous one. Based on this probability, a variable-length coding was designed: one bit was assigned for the state transitions with a probability of 0.6, two bits for the state transitions with a probability of 0.2 and three bits for the state transitions with probability of 0.1. The number of bits required for the representation of the contour directions can be further reduced, if impossible directions (due to the adjacency with other contours) and block boundaries are taken into account.

"Colour"	Bit cost
0	2
29	2
76	5
149	6
225	6
255	4
other	1+8

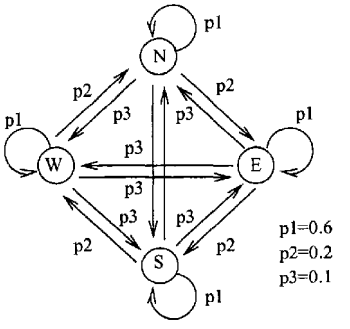


Fig. 7. (a) (left) Huffman code bit costs for the different "colours" of the contours for the luminance. (b) (right) Markov model for the set of directions (N = North, etc.).

The encoding scheme for a block including the aforementioned improvements can be summarized as follows.

- Apply the contour algorithm to determine all the different contours within the block;
- determine the number of contours;
- `if` (number of contours = 1)
  `then` - send number of contours
  - send their colours (Huffman coded);
  `else` - send number of contours;
  - determine the longest "outer" contour;
  - send the longest "outer" contour colour (Huffman coded);
  - send the remaining contours, their colour, starting point (relative offset) and set of directions (Huffman coded skipping the impossible directions).

B. *Runlength coding*

Another interesting candidate for the lossless compression of graphics is the runlength coding algorithm [10]. If the video data are sent in rectangular blocks of 4×4 pixels in size or a multiple of this, runlength coding is employed inside each block. The disadvantage of this coding technique is that the algorithm compresses line by line, and thus, the correlation between adjacent scan lines, i.e. the correlation between adjacent pixels in the vertical direction, is not exploited. For example, the block of 8×8 pixels depicted in Figure 6 is represented as a concatenation of runs having the same colour: (A, 11), (C, 1), (A, 1), (B, 2), (A, 2), (B, 2) etc. (see Figure 8). To improve the efficiency of the

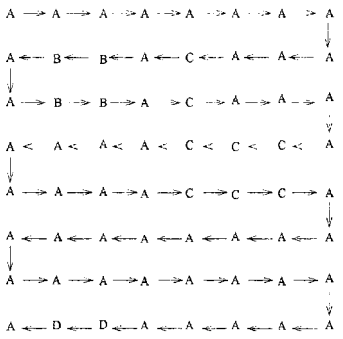


Fig. 8. Contour coding example of a block with four different colours.

coding algorithm, Huffman coding was adopted for both the pixel values and the run-length values. For this reason, the lengths of the runs for the different pixel values were measured, and subsequently, a Huffman code was designed. The Huffman code based upon the statistical distribution of the pixel values is identical with that used for the coding of the signal values for the contour algorithm (see Figure 7(a).

From the experiments conducted, we have found that the application of the Huffman codes for run lengths and pixel values, improves the coding efficiency of YUV signals with 50 % and that of RGB signals even with 65 %, thereby emphasizing the effectivity of variable-length coding in this case. A more detailed presentation of the compression results obtained with contour and runlength coding is given in Section VIII-A.

VI. Compression of mixed images

In this section we concentrate on Architecture 2, which aims at reducing the local memory of a DTV receiver. We assume that the video-graphics signals were mixed earlier in the transmission chain, such that the location of the graphics data is unknown to the receiver (if the location of the graphics data is known, separate compression systems should be adopted: the BPC technique from Section III for video and lossless techniques from Section V for the graphics). The desired receiver structure for this mixed transmission is presented in Figure 9. The compression of such a mixed signal is a complex coding problem, since it is known in advance that the signal contains different signal statistics (see Section IV). In this case, one of the best solutions for compression is to adopt optimal coding techniques for each signal type, fitting to the individual signal characteristics. A key feature of this system is the detection of graphics prior to coding. The partitioning problem of video and graphics is an intrinsically interesting problem (which was already studied in an earlier paper [12]), since it can enable different image enhancement techniques for signals with different characteristics. However, in this section we focus only on the graphics detection for classifying data blocks for optimal compression of the complete signal. The basic rules for distinguishing between the two signals are similar, but the final classification for efficient coding is different.

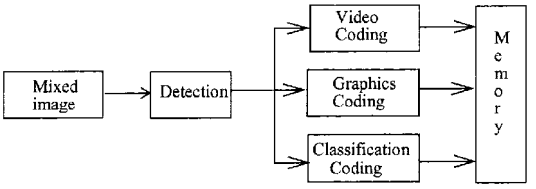


Fig. 9. Detection and classification mechanism for high-quality coding of mixed images.

A. *Detection and classification for coding*

In [12] a cost-effective method for distinguishing between video and graphics data were identified. The algorithm results in an adequate detection of "pure" graphics data, such as Teletext, buttons and computer menus, in more than 90% of all the (original) graphics regions. This detection technique considers the luminance values and dynamic ranges of the neighbouring pixels within a block for classifying a local area of the image. The detection for coding was performed on a block-basis, because the employed compression algorithms for both video and graphics are block-based.

The properties of video and graphics data used for the graphics detection algorithm are as follows.

- Within the graphics blocks, a relatively large number of adjacent pixels (mostly more than 6) have the same luminance values. This property remains valid even for special graphics data, like subtitling or pixel-based graphics. Video data mostly do not possess this property.
- The graphics data are mostly composed of areas having either the same luminance values (e.g. DR=0), or having large amplitude differences between adjacent lumi-

nance samples (e.g. DR≥30). Parts of a picture with small variations in the luminance values are typical for the background in the video scenes and therefore they are excluded as graphics data.
- Graphical data are mostly distributed along more blocks: a graphical object located in a block is most likely to be continued in an adjacent block.

The detection mechanism divides the blocks in two classes: video or graphics, depending on the characteristics of the actual block pixels. The video blocks are quantized using BPC, while for the graphics blocks, a lossless compression mechanism was employed. Runlength coding was initially chosen for compression of the graphics signal due to its low complexity and its relatively high performance in compressing the graphics components (see e.g. [6]). However, the runlength coding technique can result in a data expansion in the case of pure video or graphics data resembling video. To enable an optimal compression of the mixed signal, a different classification technique for the graphics blocks is proposed.

B. Introduction of neighbouring-area blocks

An attractive approach is to divide the various types of graphics in two classes as already indicated in Section IV:
- real graphics data, defined by sharp boundaries and large areas containing identical signal values;
- graphics data with video resemblance.

Subsequently, the first class of graphics can be compressed using lossless coding techniques in order to avoid blurring, while for the second class, similar algorithms as for video (e.g. BPC) are efficient. Therefore, the second class of graphics should be detected as if it were "video".

An improved compression result for graphics areas can be obtained by first detecting the regions (initially blocks) where the graphics predominates — called "pure" graphics blocks — and, subsequently, growing in each direction with one block, forming the "neighbouring-area" region. This partitioning is illustrated in Figure 10, where subtitling is superimposed on a video image. The "pure" graphics and "neighbouring-area" blocks form the graphics region that must be losslessly compressed. The "neighbouring-area" blocks are necessary for capturing the entire graphics elements, e.g. subtitling and graphics borders. In this way, graphics data are coded entirely without loss. However, due to the different characteristics of pure graphics and neighbouring data, different error-free coding algorithms are used. The choice of the adopted algorithm (video/ "pure" graphics/ "neighbouring-area" graphics) is made explicit to the decoder. The improved classification algorithm using neighbouring-area blocks is given below.

Algorithm

GraphicsNo = 0;
{ determine the number of adjacent pixels with the same luminance values }
for $i = 1$ **to** PixelsPerBlock
  if (i mod PixPerBline $\neq 0$) $\bigwedge(y(i) = y(i+1))$
  then GraphicsNo = GraphicsNo+1; **endif**
  if ($i >$ PixPerBline) $\bigwedge(y(i) = y(i-$ PixPerBline$))$
  then GraphicsNo = GraphicsNo+1; **endif**
endfor

{ establish the dynamic range of the luminance values }
if (GraphicsNo ≥ 8)
then determine the dynamic range of the luminance values DRy;
  if ((DRy ≥ 30) $\bigvee$ (DRy $= 0$))
  then block = graphics;
  else block = video; **endif**
else
  block = video;
endif

{ this part gives a preliminary video-graphics block classification }
if ((block = video) $\bigwedge$ ($\exists$ neighbour(block) = graphics))
then block = neighbouring-area-block;
endif

Note that, since the system is now based on three different classes of blocks, more than one bit is required for the identification.

C. Coding of neighbouring-area and graphics blocks

The "neighbouring-area" statistics are comparable to those of video, therefore error-free coding techniques for video have been considered. The goal is to eliminate the spatial correlations between the adjacent pixels within a block. For lossless coding of the blocks within the "neighbouring-area", the BPC technique cannot be directly applied, because quantization errors would be introduced. A modified version of this algorithm without the quantizer block can be employed for decorrelation. This means that in Figure 3(a), the quantizer and bit-cost allocation blocks are replaced by a lossless variable-length coder.

The choice for a variable-length coder is particularly advantageous, because it can be used for alternative purposes in our system, namely:
- pure graphics blocks (only variable-length coded);
- neighbouring-area blocks (modified BPC + variable-length coding);
- compression of the classification sequence (only variable-length coding);
- enhanced compression for pure video blocks (see [11]).

Data type	Coding techniques
Graphics	RLC, Huffman, Arithmetic
Neigb. blocks	Mod. BPC + Huffman, Mod. BPC + Arithmetic
Video	BPC, BPC + Arithmetic
Classification	RLC, Arithmetic

TABLE I
OVERVIEW OF DATA TYPES AND APPLIED CODING TECHNIQUES.

For variable-length coding, both fixed Huffman and adaptive arithmetic coding have been investigated (see [12]). The best results were obtained with an adaptive universal arithmetic coder [8] [9], employing different tables for each data type, i.e. "pure" graphics, "neighbouring-area" graphics and the classification sequence. Evidently, the tables depend on the data statistics, and both the encoder

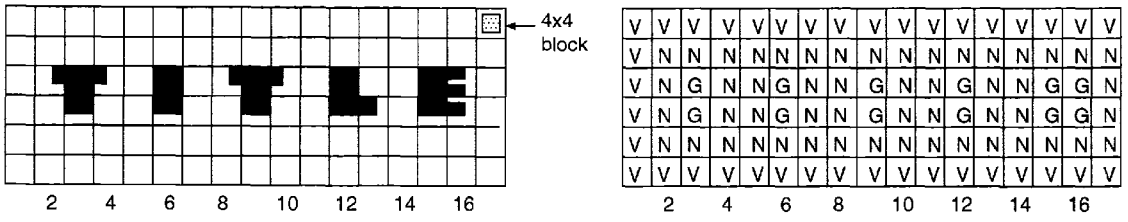


Fig. 10. Division of the frame into video (V), pure graphics (G) and neighbouring-area blocks (N).

and decoder adapt the table statistics after each block.

In the case of enhanced video compression, we have exploited the difference between luminance (Y) and chrominance (UV) signals. To further improve the compression performance, the arithmetic code can be made dependent on the dynamic range (DR) of the block [6] [11]. Depending on the DR, four different classes of data have been considered for each of the luminance (Y) and chrominance (UV) signals. This leads to eight tables for the coding of the data. Table I shows an overview of the data types and their corresponding coding techniques.

VII. Bit-rate control

For the visual material consisting of video mixed with graphics, a fixed compression factor cannot be guaranteed, because of the lossless coding of the graphics in a variable number of bits. Other problems which occur are:

- the statistics of the graphics data can vary between consecutive frames;
- video and graphics signal statistics are highly non-stationary within a frame;
- the relation between input data and output bit rate is only globally known;
- temporal prediction does not guarantee the bit cost of the actual frame.

The target of the rate controller is a constant bit rate per video frame to enable a wide range of low-cost applications. A regulation system has been developed that can switch between a lossless and a lossy mode in the coding of the graphics components for ensuring a fixed compression factor although initially a fixed compression factor cannot be guaranteed. The switch between the two compression modes is controlled by the difference between the bit costs paid for graphics in the previous frames. Since the video coding is based on segments (e.g. horizontal stripe of an image), the regulation controls also on a segment basis. The system operates as follows.

Step 1. Initialization. The bit rate controller assumes video in a frame.

Step 2. Determination of the total bit cost spent in the present frame. The total bit cost B is composed from the three classes and equals

$$B = B_V + B_G + B_N + B_{class} \ , \qquad (4)$$

where for example

$$B_V = \sum_{j=1}^{No.Segm.} \sum_{i=1}^{No.Block/Segm.} c_V(i,j) V_b(i,j) \ , \qquad (5)$$

in which the classifier $c_V(i,j)$ equals unity for video blocks and zero elsewhere, and $V_b(i,j)$ denotes the bit cost of the video block (i,j). The remaining terms B_G and B_N in (4) are defined similarly as in (5), but with different labels. B_{class} stands for thecost of classification bits. The bit costs in the equations are the real costs used for transmission.

Step 3. Determination of the total bit cost that can be spent for video in the next frame. This cost is determined assuming that the cost for graphics in the present frame resembles that of the next frame. The calculation is represented as

$$EB_V = TB - EB_G - EB_N - EB_{class} \ , \qquad (6)$$

where TB is the target bit cost for one frame, EB_G represents the cost that would occur if all pure graphics blocks would have been losslessly compressed. EB_N and EB_{class} have a similar meaning. These costs are computed independently from the target bit rate.

Step 4. Calculation of the compression factor of the video blocks in the next frame, which is simply the quotient between the uncoded cost of the video blocks and the estimated cost EB_V. If this quotient is less than two, then the controller ensures that the video blocks are compressed with a factor of two, because the lossless encoding of graphics will be realized. The value of EB_V is also checked for having at least a minimum bit cost amount for the video left over.

Step 5. Switch between BPC and arithmetic coding for the compression of graphics blocks. For each segment, we verify that the bit cost paid for coding until the current segment i, together with the bit cost required for BPC coding of the remaining segments, is below the target bit rate TB. Hence, we have

$$\sum_{i=1}^{k} SB(i) + \sum_{i=k+1}^{No.Segm.} SB_V(i) \le TB \ , \qquad (7)$$

where SB represents the cost of segment i. If (7) is not sufficiently satisfied, then the next segment will be encoded as video. The decision for lossy coding of the graphics in segment i does not influence the coding of the following segments.

In the above-mentioned algorithm, a system is implemented that always achieves the desired compression, while simultaneously trying to perform the lossless coding of all graphical objects.

VIII. Simulation results

In this chapter, we describe our results obtained by computer simulation on the basis of the two architectural mod-

els that were introduced at the beginning of the paper.

A. Comparison of contour, runlength coding and BPC for graphics (Architecture 1)

Architecture 1 (A1) focuses on efficient compression of graphics data, since graphics are transmitted independently from video data. In this subsection we consider the performance of the two techniques of Section V for efficient compression, contour coding (Section V-A) and runlength coding (Section V-B). For this reason, we have implemented a computer simulation of both contour and runlength coding for Teletext ("Teletext.yuv"), presented in the RGB format. Each picture is divided in horizontal stripes defined by an area of e.g. 4 lines × 640 samples of 8-bit resolution and containing blocks with either R, G or B components. For the conversion of picture data to stripes, field-based processing is used.

Compression factors (A1)

Table II portrays the compression factors obtained when applying the aforementioned coding techniques. The first observation is that an impressive factor of 10–15 is achieved (lossless). This is certainly explained by the statistics of the input data which is rather structured. Secondly, it can be noticed that both described coding algorithms perform equally well for the individual signal components R, G and B. Thirdly, the compression result improves when larger blocks are processed. The increased compression performance for larger blocks occurs because larger contours can be described as an entity and are not split according to the block grid. Hence, our conclusion is that for the described coding techniques, larger blocks (e.g. 32×32) are preferable for an improved compression performance. From Table II it can be seen that contour and runlength coding give similar compression results. Contour coding performs slightly better for small blocks, whereas runlength coding is better for larger blocks. It is useful to compare contour and runlength coding with a lossy coding technique, i.e. the BPC algorithm of Section III. The criterion for comparison here is that the quality of the lossy coding should approach that of runlength and contour coding. Table III shows the performance of BPC under these conditions. The quality is indicated by the SNR derived from the Mean-Squared-Error related to peak white (i.e. 255). The same sequence coded with the BPC algorithm operating in a near-lossless mode leads to a compression factor of only 2–3.

	contour algorithm		
signal	4×4	8×8	16×16
R	11.11	11.90	12.82
G	10.63	11.11	11.36
B	11.36	11.36	14.71
	runlength algorithm		
R	10.63	12.82	15.15
G	9.80	11.90	13.51
B	11.11	14.28	16.66

TABLE II
COMPRESSION FACTORS OF CONTOUR AND RUNLENGTH CODING FOR VARIOUS BLOCK SIZES.

Graphics type	SNR (R,G,B)	CF
Teletext	58.3, 57.8, 61.8	3.66
Computer menu	45.0, 44.3, 45.8	2.34
Pixel-based GFX	42.7, 42.8, 42.9	2.04

TABLE III
COMPRESSION FACTORS (CF) AND SNR OBTAINED BY BPC FOR DIFFERENT GRAPHICS TYPES.

System aspects (A1)

In the remainder of this subsection we evaluate briefly some system aspects involved for a possible implementation of the lossless coding techniques presented for graphics compression. Table IV portrays an overview of the relevant issues for a real-time application.

Despite its better compression for small blocks (i.e. 4×4 pixels), the contour algorithm has the drawback that the coding process requires an undefined number of steps which depend on the complexity of the contours (defined by the image contents). Both the number of contours and their size vary from image to image. In the case of a real-time constraint, or when the complexity of the codec needs to be kept limited (i.e. in Architecture 2), runlength coding provides a more favourable choice. For the pixel-based graphics, BPC performance is acceptable, leading to a compression factor of more than 2, while preserving a very high image quality.

B. Compression results of graphics in mixed images (Architecture A2)

In Architecture 2 (A2), the mixed video-graphics images are received and the system architecture is based on graphics detection prior to any coding decision. The key elements of the architecture and the corresponding coding techniques were presented in Sections VI and VII. In this subsection we describe the experiments which have been carried out employing the earlier described compression system.

We have simulated both a system using the runlength coding technique and a system based on the novel arithmetic coder for graphics from Section VI-C. The compression factor was chosen relatively small, i.e. 2–3, in order to obtain a high picture quality (SNR around 40–50 dB). For simulation, an extensive set of sequences containing both pure video or video mixed with graphics was chosen.

Graphics detection (A2)

Two groups of sequences were used for the evaluation of the detection algorithms: a sequence of frames where different types of graphics were mixed with the video signal ("Graphics.yuv") and a group of sequences without graphics ("Renata1.yuv", "Baltimore.yuv", etc.), but with textures locally resembling graphics data. Pure video frames have been included in the set in order to test the accuracy of the graphics detection. An inaccurate detection would result in a decrease of the compression factor, since the encoder would employ lossless compression techniques for video signals. The correct detection percentages for video and graphics described in Section VI-C are shown in the first columns of Table V (Algorithm 1). The table shows that Algorithm 1 results in an accurate detection of video

Coding technique	Contour coding	Run-length coding	BPC
Teletext application	Yes	Yes	No
Graphics application	Yes, but inefficient for video-like GFX	Yes, but inefficient for video-like GFX	Yes, efficient for video-like GFX
Throughput	Variable proc. time variable output length	Fixed proc. time variable output length	Fixed proc. time fixed output length
Complexity	High	Low	Low
Real-time suitability	Costly	Yes	Yes

TABLE IV
COMPARISON OF SYSTEM ASPECTS OF THE VARIOUS CODING TECHNIQUES PRESENTED.

Scene (*.yuv)	Alg. 1 video	Alg. 1 gfx	Alg. 2 video	Alg. 2 gfx
Graphics	89.9%	98.0%	95.9%	99.2%
Renata	99.5%	–	99.7%	–
Baltimore	96.5%	–	98.9%	–

TABLE V
DETECTION PERFORMANCE OF THE TWO GRAPHICS (GFX) DETECTION ALGORITHMS FOR MIXED AND PURE VIDEO SCENES.

Algorithm	CF video	CF gfx	CF n-bour	CF class.	CF total
RLC	n.a.	n.a.	n.a.	n.a.	1.92
RLC+N-blx	2.10	3.81	1.29	1.65	2.22
HFM+N-blx	2.27	4.02	1.47	1.81	2.41
ARC+N-blx	2.45	4.87	1.48	1.81	2.70

TABLE VI
COMPRESSION FACTORS (CF) OF CODING ALGORITHMS FOR GRAPHICS (N-BLX = NEIGHBOURING-AREA BLOCKS, RLC = RUNLENGTH, HFM = HUFFMAN, ARC = ARITHMETIC).

in pure video frames. For the sequence containing graphics, the procedure results in a very accurate detection of the graphics area, thereby guaranteeing that the image improvement algorithms are applied only upon the video areas. The absolute score is high, especially when compared to the simplicity of the algorithm presented.

Experiments have been performed employing the block classification of the previous frame to minimize decision failures. As a result of the temporal prediction, the decision became more accurate, resulting in a reduced number of wrongly detected graphics and neighbouring-area blocks, as illustrated in Table V (Algorithm 2). However, the increase in accuracy is relatively modest compared to the expense of increased buffer memory. Thus, detection of the graphics blocks using the classification of the previous frame is not very cost-effective.

Coding results (A2)

The efficiency of the coding algorithms for video frames containing moving graphics will now be evaluated by looking at the compression factors obtained. Table VI portrays the total compression factor for the various block types. It can be seen that the arithmetic coder clearly outperforms the runlength coder, particularly for the graphics blocks (gfx), while it is only marginally better for the neighbour blocks (N-blx). The total coding results improve globally with 10%.

The total compression factor obtained employing algorithm RLC+N-blx (Runlength+N-blocks) is better (i.e. 2.22) compared with a detection system based on the properties listed in Section VI-A, because the pixel-based graphics are now coded as video. For the pure video data, an average compression factor larger than two obtained with BPC is a satisfactory result. As expected, the pure graphics blocks are very well compressed by runlength coding (CF = 3.81). However, the moderate total compression factor of the system is caused by the poor coding performance of the "neighbouring-area" blocks (CF = 1.29). The more "noisy" statistics of this data leads to a much lower coding efficiency. In order to give the system a wider range of applicability, an arithmetic coder, adaptive to four classes of dynamic ranges was evaluated for the compression of the luminance (Y) and chrominance (UV) signals of pure graphics and "neighbouring-area" blocks. As already indicated, the arithmetic coder has a better performance for those blocks. Thus, the adaptive coder can be employed for pure graphics blocks, "neighbouring-area" blocks, classification sequence and enhanced video compression. The SNR obtained using the best graphics detection and arithmetic coding (see last line in Table VI) is 50.2 dB for Y and 51.3 dB for U and V.

IX. CONCLUSIONS

In this paper, various techniques have been investigated for the compression of the video and graphics data with very high-quality. Two system architectures have been considered for designing these algorithms: (1) the professional communication chain between content-provider and broadcaster and (2) the local memory architecture of a DTV receiver. In the first architecture, dedicated to the professional communication chain, the video and graphics data should be separately transmitted and stored. For this case, we have adopted coding techniques that exploit the statistical and visual properties of each individual signal. For the compression of the video and pixel-based graphics, compression factors of 2–3 were obtained with "near-lossless" quality. For the graphics data, several novel algorithms have been presented, which achieve compression factors up to 16, depending on the allowed complexity and the signal statistics. If desired, these techniques can also be succesfully applied for efficient local storage at the content-provider and broadcaster. The second architecture is for reducing the memory costs of a DTV receiver, where mixed video-graphics signals are processed. The key features of the coding system employed for this architecture are *a-priori* detection, block-predictive coding and arithmetic coding. All these components have been implemented with minimal complexity, such that the overall costs are kept below the

costs of the saved memory. These cost trade-offs are often difficult to make if the silicon and memories are fabricated in different IC processes, but easy to evaluate whenever memories are embedded on-chip.

REFERENCES

[1] European Association of Consumer Electronics Manufacturers, EACEM Technical Report No. 8, "Enhanced Teletext Specification", Draft 3, Nov. '94.

[2] "Coding of Moving Pictures and Associated Audio", International Standard, ISO/IEC 13818, Nov. '94.

[3] P.H.N. de With and A.M.A. Rijckaert, "Design considerations of the video compression system of the new digital DV recording standard", IEEE Trans. Consum. Electron., Vol. 43, Nov. '97.

[4] T. Kondo *et.al.*, "A new ADRC for consumer digital VCR", IEE Proc. 8th Video Audio & Data Recording Conf., Birmingham (UK), 1990, IEE Conf. Publ. No. 319, pp. 144–150, April '90.

[5] P.H.N. de With, M. Breeuwer, P.A.M. van Grinsven, "Data Compression Systems for Home-Use Digital Video Recording", IEEE JSAC, Vol. 10, No. 1, pp. 97–121, Jan. '92.

[6] P.H.N. de With and M. v.d. Schaar, "Evaluation of lossless compression of Teletext and graphics images for TV Systems", Proc. SPIE-EOS Europto Conf. Digital Compr. Techn. & Systems Video Com., Germany, Vol. 2952, pp. 506–517, Oct. '96.

[7] R.W.J.J. Saeijs, P.H.N. de With, A.M.A. Rijckaert, C. Wong, "An Experimental Digital Consumer Recorder for MPEG-Coded Video Signals", IEEE Trans. on Consum. Electron., Vol. 41, No. 3, pp. 651–661, Aug. '95.

[8] G.G. Langdon and J. Rissanen, "Compression of Black-white Images *with Arithmetic Coding*", *IEEE Trans. on Commun.*, Vol. COM-29, No. 6, pp. 858–867, 1981.

[9] I.H. Witten, R.M. Neal and J.G. Cleary, "Arithmetic Coding for Data Compression", Commun. of the ACM, Vol. 30, No. 6, pp. 520–540, 1987.

[10] S.W. Golomb, "Run-Length Coding", IEEE Trans. Inform. Theory, Vol. 12, No. 1, pp. 399–401, 1966.

[11] M. v.d. Schaar and P.H.N. de With, "A Comparison between Huffman and Arithmetic Coding for Video Compression", Proc. 17th Int. Sympos. Inform. Theory in the Benelux, Enschede (NL), 1996, ISBN 90-365-0812-6, pp. 25–30, May '96.

[12] M. v.d. Schaar and P.H.N. de With, "Compression of mixed video and graphics images for TV systems", Conf. on VCIP '98, San Jose CA, SPIE Proc. Vol. 3309, Jan. '98.

[13] M. v.d. Schaar and P.H.N. de With, "Novel embedded compression algorithm for memory reduction in MPEG codecs", VCIP '99, San Jose CA, SPIE Proc. Vol. 3653 part 2, pp. 864–873, Jan. '99.

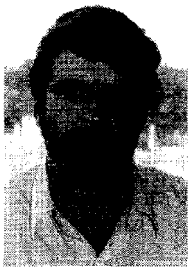
[14] P.H.N. de With. P.H. Frencken and M. v.d. Schaar, "An MPEG decoder with embedded compression for memory reduction", IEEE Trans. Consum. Electron., Vol. 44, No. 3, pp. 545–555, Aug. '98.

[15] A. Vetro *et al.*, "A minimum drift 3-Layer scalable DTV decoder", ICCE '98 Digest Techn. Papers, Los Angeles CA, pp. 54–55, June '98.

[16] J.G.N. Henderson, "All-Format Decoders and Set-Top Boxes", Proc. ICIP '98, Vol. 1, Chicago IL, p. 4, Oct. '98.

[17] "Information Technology - Coding of Audio-Visual Objects: Visual", Committee Draft, ISO/IEC 14496-2, N2202, March '98.

Mihaela van der Schaar graduated in electrical engineering from Eindhoven University of Technology, the Netherlands, in April 1996. In the same year she joined Philips Research Laboratories Eindhoven, where from 1996 to 1998 was involved in investigating low-cost very high quality video compression techniques and their implementation for TV and computer systems. Since 1998, she is an expatriate at Philips Research Briarcliff, USA, where she is a member of the Video Communications Department and is involved in the research of video coding techniques for Internet video streaming. Since 1999, she is an active participant to the MPEG-4 video standard, contributing to the Fine Granularity Scalability tool. Her research interests include video and graphics coding and streaming over IP and she co-authored more than 15 papers in this field and holds several patents.



Peter H.N. de With graduated in electrical engineering from the University of Technology in Eindhoven. In 1992, he received his Ph.D. degree from the University of Technology Delft, The Netherlands, for his work on video bit-rate reduction for recording applications. He joined Philips Research Laboratories Eindhoven in 1984, where he became a member of the Magnetic Recording Systems Department. From 1985 to 1993 he was involved in several European projects on SDTV and HDTV recording. In the early nineties he contributed as a video coding expert to the DV standardization committee. In 1994 he became a member of the TV Systems group, where he was working on advanced programmable video processing architectures. In 1996 he became senior TV systems architect and in October 1997, he was appointed as full professor at the University of Mannheim, Germany. At the faculty Computer Engineering, he was heading a group working on video coding, processing and its realization. Since August 2000, he is with CMG Eindhoven, the Netherlands, as a senior consultant and associate professor in Information and Communication Systems at the University of Technology Eindhoven, The Netherlands. Regularly, he is a teacher of the Philips Technical Training Centre and for other post-academic courses. He has written and contributed to numerous international publications and conference proceedings and he holds a list of US patents. In 1995 and 1999, he co-authored papers that received the IEEE CES Transactions Paper Award. In 1996, he obtained a company Invention Award. In 1997, Philips received the ITVA Award for its contributions to the DV standard. Mr. de With is a senior member of the IEEE, program committee member of the IEEE CES and board member of the Benelux working group for Information and Communication Theory.