Available online at www.sciencedirect.com

SCIENCE DIRECT®

J. Vis. Commun. Image R. 16 (2005) 589–620

JOURNAL OF VISUAL Communication & IMAGE Representation

www.elsevier.com/locate/jvci

MPEG video markup language and its applications to robust video transmission ☆

Xiaoming Sun [a], Chang-Su Kim [b], C.-C. Jay Kuo [a,*]

[a] *Integrated Media Systems Center and Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089-2564, USA*

[b] *Department of Information Engineering, The Chinese University of Hong Kong, Shatin, NT, Hong Kong*

Received 16 March 2004; accepted 8 March 2005
Available online 27 April 2005

Abstract

In this research, we develop the MPEG video markup language (MPML) to describe MPEG-4 video contents. As an XML designed specifically for MPEG-4 bitstreams, MPML provides a friendly support for random access, portability, interoperability, flexibility, and extendibility. We propose an efficient compression algorithm to reduce the amount of storage space for MPML documents. Then, as two applications, we show how to use MPML for error-resilient video transmission over wireless channels and video multicast over Internet. It is demonstrated that the MPML-based decoding algorithm can protect image quality effectively against transmission errors over noisy channels by using the random access support of MPML.
© 2005 Elsevier Inc. All rights reserved.

Keywords: MPEG video; Markup language; XML; MPEG-4; Robust video transmission; Video multicast

☆ This research was funded by the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152.

* Corresponding author. Fax: +1 213 740 4651.

E-mail addresses: sunx@pollux.usc.edu (X. Sun), cskim@ieee.org (C.-S. Kim), cckuo@sipi.usc.edu (C.-C. Jay Kuo).

1047-3203/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.jvcir.2005.03.007

1. Introduction

Multimedia contents, especially video, consist of a rich volume of data. Various compression algorithms have been developed to reduce the redundancy of video data in the statistical, spatial, and temporal domains. Typical video compression algorithms are based on the hybrid coding framework of motion-compensated prediction and transform coding [1]. They follow a sequential processing fashion so that the decoding process is restricted by the sequential mode as well. For example, the current macroblocks (MBs) or frames cannot be decoded until the previous MBs or frames are available. The current video coding approach also provides a limited amount of global information for the decoder so that some desirable functionalities cannot be easily accomplished such as random access and error resilience. In this work, we design a new video description scheme to overcome the drawbacks of the sequential processing while maintaining consistency with the current video coding framework.

The drastic increase of Internet applications has brought important impacts on multimedia systems. Internet provides open interfaces to heterogeneous systems to establish a bigger interconnected system. To facilitate Internet publishing and support seamless interoperability with various heterogeneous platforms, multimedia applications require a media descriptive language (or markup language). The extensible markup language (XML) [2] is an emerging generation of markup language for Web-oriented publishing. It provides simple and flexible markup definitions and has a great potential to represent a rich class of document structures. Thus, it has been used in a wide range of applications including Web-oriented databases, document management, and software engineering. Due to its powerful markup syntax, XML can satisfy various requirements arising from heterogeneous systems such as interoperability, scalability, openness, and random access.

These properties of XML can be helpful in various video delivery applications, including error-resilient video transmission and multicast video streaming. As video file gets compressed more, the encoded bitstream becomes more vulnerable to bit errors over wireless channels and packet loss over IP networks. Many techniques have been developed to enhance the error resilience of video bitstreams [1,3], but the lack of random access capability in the decoder makes it difficult to recover from transmission errors. Also, the synchronization of multiple receivers is an important issue in multicast video streaming [4–6]. But traditional video coding schemes do not provide enough global information to facilitate the synchronization. These problems can be effectively overcome by an XML-based markup language due to its inherent support for random access.

Motivated by these considerations, we have developed a markup language for MPEG-4 video, called MPEG video markup language (MPML), in [7–12]. MPML is a portable XML description for MPEG video, which can be customized for various video applications. The extensible MPEG textual format (XMT) [13] is another XML for the description of MPEG video contents. It aims at interacting with existing multimedia XMLs (e.g., X3D [14], SMIL [15], SVG [16]) and equipping users with XML-based abstraction and manipulation of MPEG-4 systems. It has two levels of textual syntax and semantics: XMT-A and XMT-O. XMT-A, which is interoperable with X3D, provides XML description of MPEG-4 binary format for scenes (BIFS) and object descriptors. XMT-O is a higher level abstraction based on SMIL and can be mapped to XMT-A representation. Also, the ISO MPEG-7 framework [17–19] attempts to support interactive scene description, content description, and powerful programmability of various multimedia contents. Currently, XMT is being integrated into the MPEG-7 framework. As compared with XMT and MPEG-7, MPML provides XML description of MPEG components at the bit level. To summarize, MPML, XMT, and MPEG-7 provide low-level, medium-level and high-level XML descriptions of MPEG video contents, respectively. Although designed for different purposes, these XML descriptions are inherently interoperable and can be embedded into one another due to their conformances to the MPEG standard.

In this research, we first propose an efficient compression algorithm for MPML documents to reduce their overheads. Then, as two applications of MPML, we investigate the application of MPML to error-resilient video transmission on the wireless environment and multicast resynchronization over Internet.

The article is organized as follows. We discuss the design of MPML and its compression in Section 2. The MPML-based robust video transmission is introduced in Section 3, and the MPML-based multicast resynchronization is discussed in Section 4. Finally, concluding remarks are given in Section 5.

2. MPML and its compression

In this section, we briefly review the markup theory. Then, we introduce the MPML description for MPEG-4 video contents and its efficient compression algorithm.

2.1. Markup theory and XML

Markup is a set of instructions written on a manuscript to clarify its information. In markup theory, there are six types of markups [20]. Punctuational markup consists of a primary set of markups providing syntactic information about the written utterances. Presentational markup can make the presentation clearer with appropriate horizontal and vertical spacing, page breaks, and so on. Procedural markup provides commands indicating how the document should be formatted and often replaces the presentational markup. Descriptive markup declares the membership of a portion of text stream and thus describes the document structure. Many problems in document development can be tackled by the descriptive markup. Referential markup refers to external entities and is replaced by those entities during the document processing. Meta markup provides us with a facility of controlling the interpretation of markups and extending the vocabulary of descriptive markup languages.

To regulate a standardized markup language, ISO developed the standard generalized markup language (SGML) as a mature language for expressing document

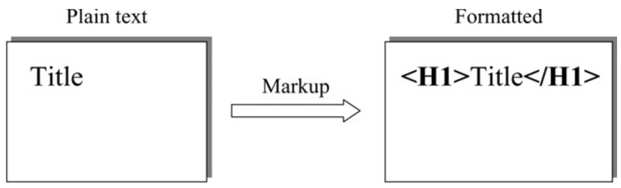


Fig. 1. Separation of plain text and markup.

structure in 1986 [21]. Two separations were first declared in SGML: the separation between plain text and markups and the separation between document structure and its display. Fig. 1 shows an example of the separation between plain text and markups. The markups do not modify the document content. They are only used to describe the document structure, and the original text body can be retrieved by removing the markups. Fig. 2 shows the processing of SGML. The marked-up document is parsed by the SGML parser. Then, the renderer displays the parsed result. It is clear that the parsing process is separated from the rendering process. In this way, the document content (or text) itself does not vary with the style how it is read or displayed. As a result, document portability, interoperability, and maintenance can be improved significantly.

As a simple instantiation of SGML, HTML was invented as a media description language for interconnected computer systems at CERN in 1991, which marked the new era of the World Wide Web. Although HTML is simple and powerful in media representation for Web publishing, it does not fully implement SGML. The markup definitions in HTML are fixed, and they may not satisfy the requirements of various applications. These drawbacks were recognized by researchers, and XML was proposed as a complete incarnation of SGML [2]. XML strictly conforms to the two separation rules of SGML and allows high freedom of markup definitions. Therefore, users can define their own XML as well as the document structure. Due to the powerful syntax, open markup definitions, and friendly support for portability and interoperability, XML is not only used in Web-oriented applications, but also plays an important role in cross-platform and data-exchangeable applications. Microsoft Office software is a good example for the latter case.

XML has been applied to multimedia applications. The synchronized multimedia integration language (SMIL) was recommended by W3C as an XML-based language for authoring interactive multimedia presentations [15]. The scalable vector graphics (SVG) [16] was released by W3C as a standard for describing vector-based graphic format. The vector markup language (VML) is another markup language for vector information coding. The Web3D Consortium, jointly with the W3C Consortium,

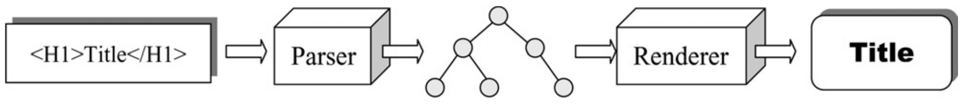


Fig. 2. The processing of SGML.

proposed the extensible 3D (X3D) to describe interactive 3D contents integrated with multimedia. X3D is the successor to virtual reality modeling language (VRML) and improves upon VRML with new features, stricter conformance and additional data encoding formats [13,14].

2.2. MPML description for MPEG-4 video content

2.2.1. Terminologies

The current video coding standards, including MPEG-4, emphasize the sequential coding fashion. However, the sequential coding causes strong dependencies among compressed data, which are not desirable when random access is required in applications. To alleviate this problem, we design an XML-based markup language, called MPML, to describe MPEG-4 video contents [7–12].

Fig. 3 introduces several terminologies for MPML. MPML-coded contents are saved in XML documents, which are referred to as MPML descriptions or documents. XML trees for MPML descriptions are referred to as MPML trees. An MPML parser can read an MPML document and generate the corresponding MPML tree. Conversely, an MPML generator can convert the MPML tree to the MPML document. MPML encoder and decoder are, respectively, the MPEG encoder and decoder integrated with an MPML parser, which can encode and decode both MPEG bitstreams and MPML descriptions. An MPML tree can be directly translated into an MPEG bitstream by an MPML–MPEG parser.

A full MPML description of a given MPEG bitstream is referred to as the complete MPML description. Basically, an MPML description consists of two parts: the

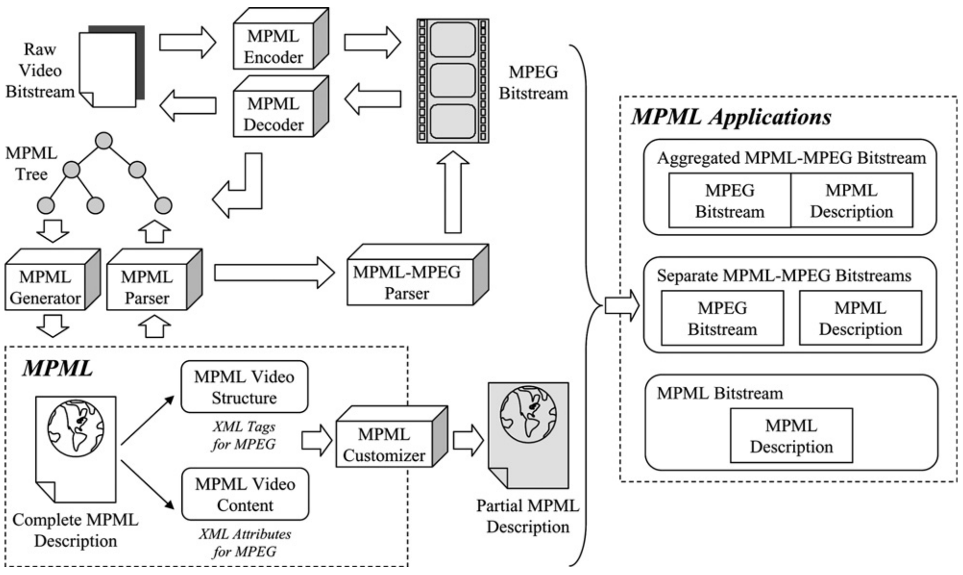


Fig. 3. MPML concepts and applications.

video structure, which contains XML tags for storing the structural information of MPEG video; and the video contents, which contain XML attributes to represent the compressed codewords of MPEG video. Note that the MPML description does not contain XML data but only tags and attributes. The complete MPML description may be too large to be directly used in applications. A partial MPML description can be generated from the complete MPML description by an MPML customizer according to the requirements of applications.

An MPEG bitstream and its MPML description can be stored as an aggregated file or two separate files. The aggregated MPML–MPEG bitstream demands strong error protection of the entire bitstream in hostile error-prone environments. In contrast, the separated MPML–MPEG bitstreams facilitate the use of unequal error protection codes. In Sections 3 and 4, we will apply MPML to error-resilient coding and multicast resynchronization. In these applications, the video content is sent as separate MPML–MPEG bitstreams and the error correction codes are applied only to the MPML description.

2.2.2. Mapping from MPEG syntax to MPML description

In MPEG-4, each visual object (VO) is represented by its shape, texture, and motion [22,23]. VOs can be created in a variety of ways. They can be generated from the spatio-temporal segmentation of natural scenes or the parametric description of graphic objects. Alternatively, one or a set of rectangular image frames can be treated as a VO. The syntax of MPEG-4 bitstream is defined hierarchically as shown in Fig. 4. The frame or picture is called the video object plane (VOP), whose shape and texture

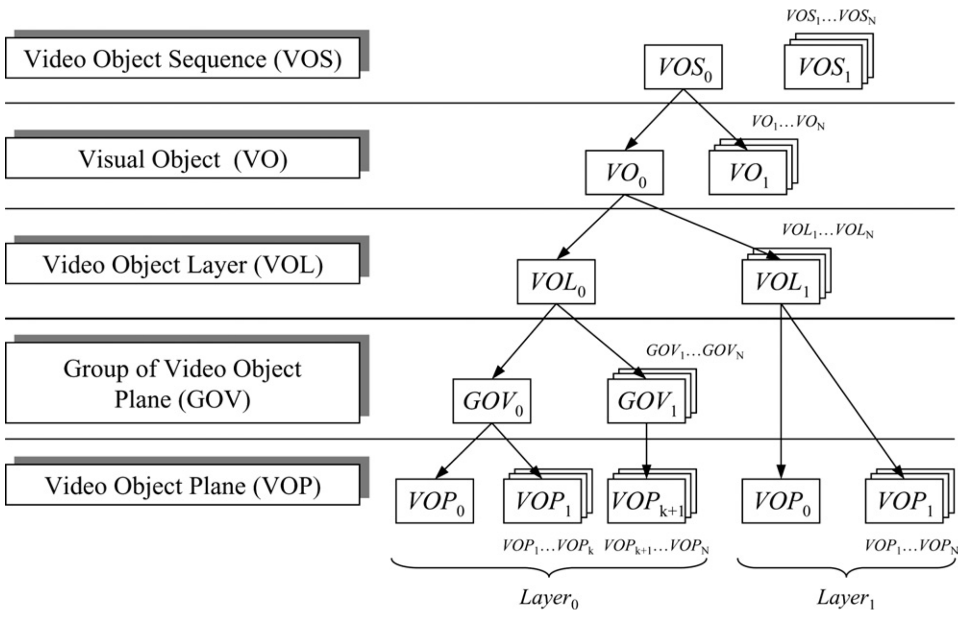


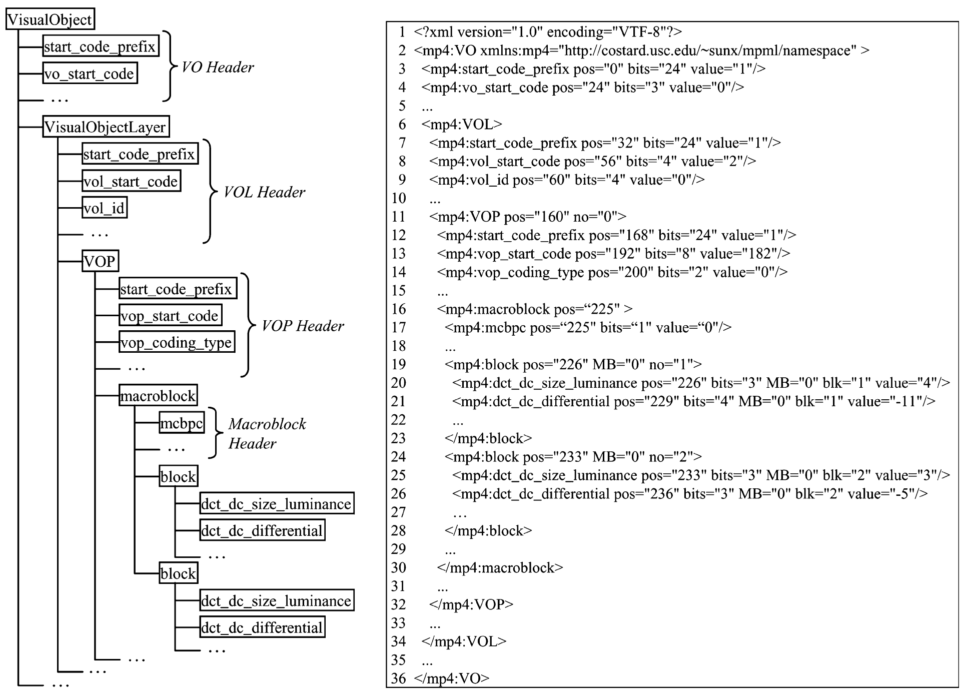
Fig. 4. Hierarchical representation of an MPEG visual bitstream.

data are predictively encoded. A group of VOP (GOV) defines a collection of VOPs, which can be independently decoded. The video object layer (VOL) is reserved for scalable coding options.

Note that the MPEG-4 syntax is hierarchically organized in a well-defined tree. In fact, Fig. 4 illustrates the tree structure of headers in MPEG-4 bitstream, and the actual bitstream can be obtained by the pre-order traversal of the tree. Thus, the root element of a subtree at a certain level (e.g., VOL or VOP) represents a number of bits in the compressed bitstream. This mapping relation can be nicely preserved in most situations, although there are some exceptions due to the requirements of synchronization and error resilience.

Fig. 5 shows the mapping of the MPEG tree structure to the MPML document, which is also tree-structured. We design MPML based on the following two principles. First, MPML adopts the same mnemonics as the C-style pseudo code representation of MPEG-4 standard. Second, MPML preserves the same storage order as MPEG-4 bitstream. Based on these principles, MPML can present a clear view of the MPEG-4 video content.

We have mapped all components in MPEG-4 video into XML tags in our MPML name space. The presence of XML tags outlines the structural information of the video, while the attributes of XML tags carry the descriptive information about the video content. To enable the reconstruction of the original MPEG bitstream from the MPML description, three attributes `pos`, `bits` and `value` are used for most



```
<?xml version="1.0" encoding="VTF-8"?>
<mp4:VO xmlns:mp4="http://costard.usc.edu/~sunx/mpml/namespace" >
  <mp4:start_code_prefix pos="0" bits="24" value="1"/>
  <mp4:vo_start_code pos="24" bits="3" value="0"/>
  ...
  <mp4:VOL>
    <mp4:start_code_prefix pos="32" bits="24" value="1"/>
    <mp4:vol_start_code pos="56" bits="4" value="2"/>
    <mp4:vol_id pos="60" bits="4" value="0"/>
    ...
    <mp4:VOP pos="160" no="0">
      <mp4:start_code_prefix pos="168" bits="24" value="1"/>
      <mp4:vop_start_code pos="192" bits="8" value="182"/>
      <mp4:vop_coding_type pos="200" bits="2" value="0"/>
      ...
      <mp4:macroblock pos=“225” >
        <mp4:mcbpc pos=“225" bits=“1" value=“0"/>
        ...
        <mp4:block pos="226" MB="0" no="1">
          <mp4:dct_dc_size_luminance pos="226" bits="3" MB="0" blk="1" value="4"/>
          <mp4:dct_dc_differential pos="229" bits="4" MB="0" blk="1" value="-11"/>
          ...
        </mp4:block>
        <mp4:block pos="233" MB="0" no="2">
          <mp4:dct_dc_size_luminance pos="233" bits="3" MB="0" blk="2" value="3"/>
          <mp4:dct_dc_differential pos="236" bits="3" MB="0" blk="2" value="-5"/>
          …
        </mp4:block>
        ...
      </mp4:macroblock>
      ...
    </mp4:VOP>
    ...
  </mp4:VOL>
  ...
</mp4:VO>
```

Fig. 5. The tree structure of an MPEG-4 visual bitstream and the corresponding MPML document.

XML tags as shown in Fig. 5. The attribute `pos` represents the position of the corresponding element in the video bitstream, and `bits` informs how many bits the element occupies. From the pair (`pos`, `bits`), we can locate a compressed element within the MPEG bitstream. The attribute `value` is the real value of the element, which is converted from the binary codeword. We can validate the genuineness of the compressed video elements by cross-checking with the `value` attributes.

To enhance error resilience, MPEG-4 provides the optional mode of data partitioning (DP), which separates the header and motion information from the texture information. This mode breaks the basic tree structure of MPEG-4, since the MB and block data are shuffled within a slice. However, to maintain the random access capability of MPML, we still use the MPML tree structure in the DP mode and make necessary modifications. In the VOL header, there is a tag 'data_partitioned' indicating whether the video data are encoded in the DP mode. Even in the DP mode, the texture and motion data are put under the corresponding MB/block nodes, such that they can be randomly accessed from the MPML tree. The `pos` attribute gives the real physical locations within the video bitstream. Using the modified `pos` attributes, MPML parser can achieve the exact correspondence between MPEG-4 bitstream and MPML document.

2.3. Joint processing of MPEG bitstreams with MPML descriptions

2.3.1. Generation of MPML descriptions

MPML is a precise markup description of MPEG-4 video without any essential modifications on the encoding and decoding algorithms. Hence, the MPML generator is a tight combination of the MPEG encoder/decoder and the XML parser, and MPML descriptions can be generated at the encoder or the decoder of MPEG-4 bitstreams.

Fig. 6 illustrates the MPML generator implemented at the decoder end. The XML parser is integrated into the MPEG decoder at the point where the shape, motion, and texture data are fetched from the demultiplexer. When the MPEG decoder gets a codeword from the demultiplexer, the integrated XML parser (i.e., the MPML parser) generates the corresponding MPML node to build up the XML tree. The MPML attributes such as `pos` and `bits` are determined at this moment.

The MPML generator implemented at the encoder end has a similar architecture. Fig. 7 shows the MPML generator at the encoder end, which can generate both the MPML document and the MPEG bitstream. When a codeword is written out to the MPEG bitstream, it is also recognized by the MPML parser.

2.3.2. Indexing and random access of MPML trees

In general, XML parsers provide a set of powerful tools to allow various operations on XML trees, such as random access of nodes. Thus, using the MPML parser that is a specific example of XML parsers, the global indexing and random access of MPML trees can be performed efficiently.

An element of the MPML tree is identified by its name and index. The name represents the level of the element within the tree, and the index represents the order of

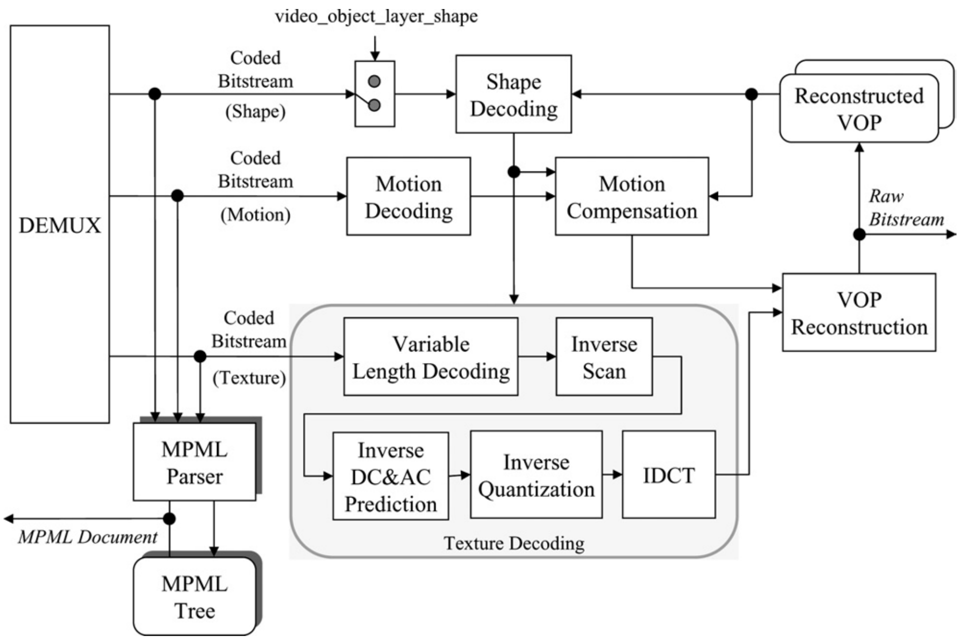


Fig. 6. Generation of MPML at the decoder end.

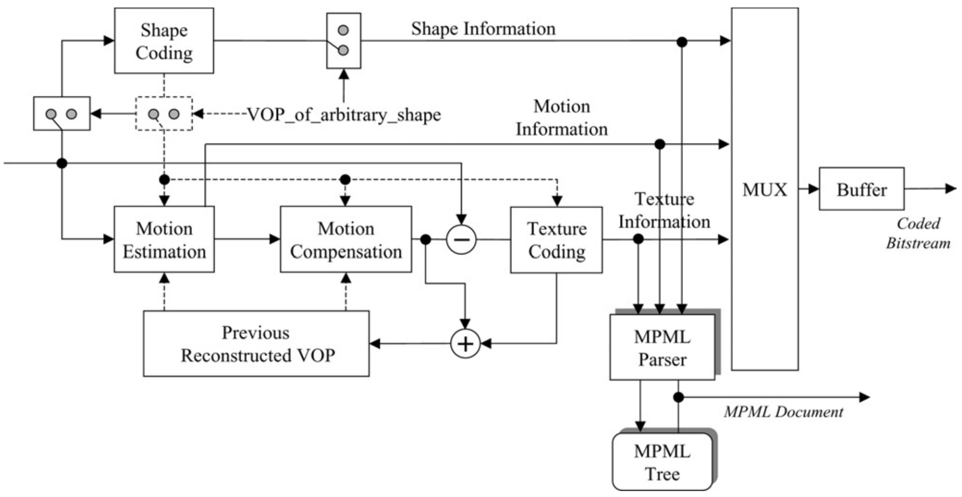


Fig. 7. Generation of MPML at the encoder end.

the element among its siblings. Using the name and the index, we can find the path from the root node to the target node, which contains the desired element.

Fig. 8 shows how to randomly access the attribute `value` of the node 'Visual-Object/VideoObjectLayer/VOP/macroblock/mcbpc.' If we traverse the tree nodes

```
string value = ("VisualObject", 0)
        ("VideoObjectLayer", xmlDocTree->xml_statistics.
          GetCount(XML_VIDEO_OBJECT_LAYER) - 1)
        ("VOP", vol_wrapper->xml_statistics.GetCount(XML_VOP) - 1)
        ("macroblock", vop_wrapper->xml_statistics.GetCount(XML_MACROBLOCK) - 1)
        ("mcbpc", mb_wrapper->xml_statistics.GetCount(XML_MCBPC) - 1).
          GetAttribute("value").GetValue();
```

Fig. 8. Random access of an MPML tree.

sequentially in the same order as the MPEG-4 bitstream, it is not necessary to specify the indices. Instead, we can keep track of already accessed nodes and know the index of the next node to be accessed. In Fig. 8, 'xml_statistics' is the data structure which stores the number of already accessed nodes.

2.3.3. MPML-assisted decoding

To apply the MPML description to the decoding of MPEG-4 video, the description should be parsed by the MPML parser to generate the MPML tree. Fig. 9 shows the structure of the MPML-assisted MPEG decoder. Once the MPML tree is built up, there are two input sources available for the decoding. The first source is fetched from the MPEG bitstream through the demultiplexer, and the second source is obtained by traversing the MPML tree. The compressed video data can be accessed from the MPML tree using the random access capability.

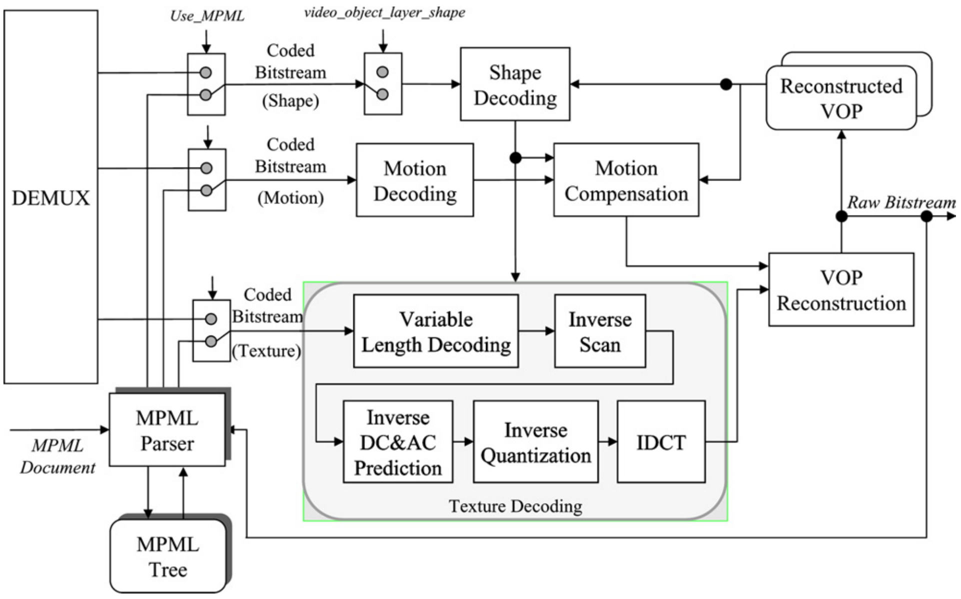


Fig. 9. The MPML-assisted decoding process.

Depending on applications, we can determine which source should be used for the decoding and turn on/off the switch 'Use_MPML' in Fig. 9 accordingly. For example, if the transmitted MPEG bitstream is not reliable due to the network congestion, the information conveyed by the MPML document can be employed to improve the decoding performance. We will discuss the issue of error-resilient decoding of MPEG video in Section 3.

2.4. MPML compression

One of the main advantages with the MPML description is that we can randomly access to any element in the compressed video. However, without careful implementation, MPML documents can be too huge to be used in practical applications. Our experiments confirmed that a text-based MPML document can be up to 200 times bigger than the corresponding MPEG bitstream. MPML documents can be compressed losslessly using Huffman coder or Lempel-Ziv coders. However, these coders achieve a typical compression ratio of 40–60%, which is not enough to compress huge MPML documents. It is essential to develop a more efficient compression algorithm for MPML documents.

General XML documents have well-defined structures and limited vocabulary, and their attribute values are confined within a specific range. XMill [24] and XMLPPM [25] are two examples of XML coders, which exploits these properties to achieve a good compression ratio. XMill first separates the structure from the data. Then, it groups related items and apply the gzip method to compress each group. It can achieve about twice higher compression ratio than general-purpose coders. XMLPPM uses an alternative approach for XML compression based on the prediction by partial match (PPM) modeling. It was shown that XMLPPM provides even better compression performance than XMill.

2.4.1. Structural and semantic properties of MPML documents

MPML documents have a succincter format than general-purpose XML documents. The structural properties of MPML documents can be summarized as follows.

(1) An MPML document is composed of XML tags only.
(2) The MPML document uses a small and fixed set of XML tag names.
(3) The XML tree is fat in general. Its depth is bounded by a small number less than 10. Its widths at the top two levels (VO and VOL) can be large, whereas its widths at the other levels are bounded by a fixed number. For example, a VOP of QCIF format contains only 99 MBs.
(4) The XML tree is roughly balanced.

An MPML document is the description of MPEG-encoded video, which has a well-defined semantics. Thus, the MPML document also has a number of semantic properties that can be used for a higher coding gain. For example, if 'start_code_prefix' and 'vop_start_code' appear contiguously in the MPML document, they are also contiguous in the MPEG bitstream. Also, they are represented by a fixed number of bits.

The MPML document contains the topology information of MPML tree and three kinds of tag attributes (i.e., `pos`, `bits`, and `value`). The topology and tag attributes in MPML documents have the following semantic properties.

(1) The topology can be induced by the positions of MPML tags.
(2) All the attributes are of integer type. Moreover, the `pos` and `bits` attributes are non-negative integers.
(3) The attribute `pos` of the current MPML tag can be obtained by adding `pos` and `bits` of the previous MPML tag.
(4) The `bits` attributes of many MPML tags are fixed numbers.

These structural and semantic properties are used to develop a highly efficient MPML coder.

2.4.2. Compression algorithm for MPML documents

The MPML tree is partitioned into a sequence of subtrees, and each subtree is encoded independently as a partition unit. If the corresponding MPEG-4 bitstream is encoded in the DP mode, a partition unit is a video packet delimited by resynchronization markers. In the default mode, a partition unit is defined as a number of contiguous MBs.

We encode tag names and attributes separately. For the attribute coding, if an MPML tag spends a fixed number of bits, its `bits` attribute is not encoded. Also, from the semantic property (3), the `pos` attribute of an MPML tag can be reconstructed by adding `pos` and `bits` of the previous MPML tag. Hence, all `pos` attributes, except those for macroblock tags, are omitted in the compressed MPML bitstream. The `pos` attributes for macroblock tags are retained to support random access to the macroblock data.

If there is no request, the topology information of the MPML tree is not encoded, since it can be reconstructed by parsing the MPML tags. As an optional mode, we encode the tree topology based on the breadth-first traversal [26].

Fig. 10 shows the architecture of the MPML coder, which is similar to that of the XMill coder [24]. The MPML compression is a two-pass procedure. The SAX (simple API for XML) parser is an XML parser, which facilitates the access of the MPML tree. In the first pass, using the SAX parser and the path processor, we classify the MPML data and put them into different semantic coders. The semantic coders convert the textual MPML data into binary data. Then, those binary data are put into appropriate containers, according to the information they carry. In the second pass, the arithmetic coder (AC) is applied to each container to reduce the bit rate. Note that by grouping related data into the same container, the compression performance of the arithmetic coding can be improved significantly. Finally, to support the DP mode of MPEG-4, the MPML data for MB modes and motion vectors (MVs) are coded into separate segments in the output bitstream.

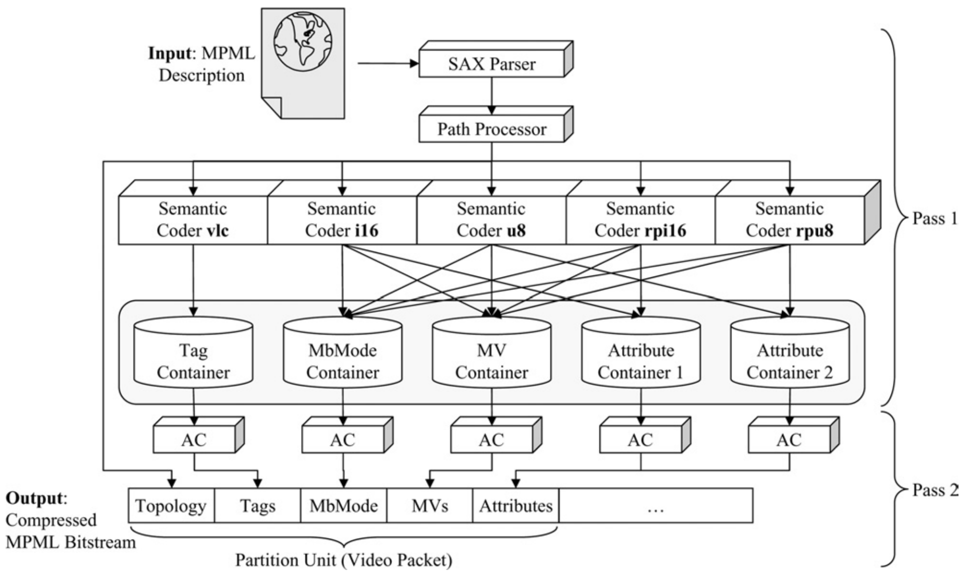


Fig. 10. The architecture of the MPML coder.

The semantic coders convert the textual data of MPML documents into binary data. Table 1 lists the semantic coders employed in the MPML encoder. The first coder **vlc** is used to represent MPML tags with variable length codewords. Several tags are always associated with fixed attributes. For example, 'vop_start_code' has always a fixed value 000001B6 in hexadecimal number. In such cases, the attributes are not encoded. The second coder **i16** is used to encode the attributes that are signed 16-bit integers. For instance, **i16** is used to encode the `value` attribute of 'dct_dc_differential,' which represents the difference between the DC coefficients of adjacent blocks. The third coder **u8** is used to encode the attributes that are unsigned 8-bit integers. In MPML, `value` attributes often represent codewords in VLC tables of MPEG-4. All these tables contain less than 256 codewords. Therefore, those `value` attributes can be represented by the corresponding indices to the VLC tables, which in turn can be encoded by **u8**.

As an optional mode, the coders **i16** and **u8** can be replaced with the range partitioning coders **rpi16** and **rpu8**, respectively. These range partitioning coders

Table 1
The semantic coders in the MPML encoder

Coder	Data items
vlc	MPML tags
i16	Signed attributes, e.g., the value attribute of "dct_dc_differential"
u8	Unsigned attributes, e.g., the indices to MPEG-4 codeword tables
rpi16	The same as those for semantic coder **i16**
rpu8	The same as those for semantic coder **u8**

Table 2
The 16-bit signed range partitioning semantic coder **rpi16**

Flag	Bits allocated	Value range	Total bits
00	2	$[-2^1, 2^1-1]$	4
01	4	$[-2^3, 2^3-1]$	6
00	8	$[-2^7, 2^7-1]$	10
11	16	$[-2^{15}, 2^{15}-1]$	18

use a 2-bit flag to confine the range and then use a different number of bits to represent the actual value. Table 2 shows the coding scheme of **rpi16**. For example, to represent value 6, **rpi16** first encodes the flag 01 and then encodes the actual value with 4 bits 0110. Thus, 6 bits (=010110) are required in total. Since small attributes are more probable than large attributes, the range partitioning coders can save bits. The coding scheme of **rpu8** is defined in a similar way.

The decoding process is performed at the partition unit level. If the topology information is omitted, the MPML tree is built up by parsing MPML tags. After decoding an MPML tag, the corresponding attributes are decoded and put into the proper places.

2.4.3. Experimental results

Fig. 11 and Table 3 show the compression performances of Winzip, XMill, and the proposed algorithm on MPML documents. In this test, "dog and man" QCIF (176×144) sequence is used as a test sequence. Its frame rate is 6 frames/s. It is first encoded into the MPEG-4 bitstream at a bit rate of 48 kbps and then converted into the MPML document. For comparison, the file sizes for the MPEG-4 bitstream are also presented.

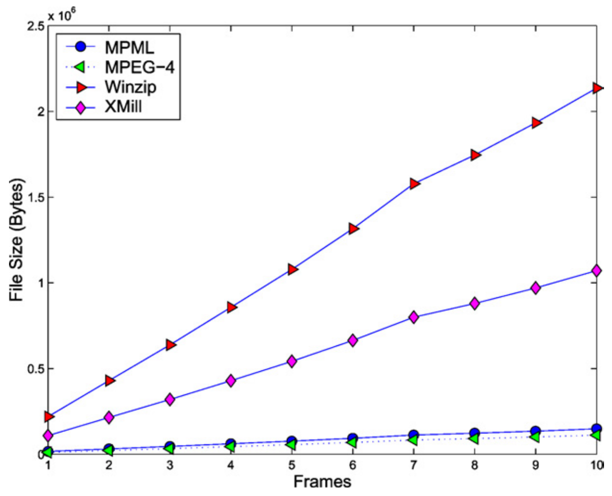


Fig. 11. The compression performances on MPML file.

Table 3
Performance comparison of several compression algorithms

Frames	MPEG-4	Winzip	XMill	MPML
10	11,307(2.97%)	219,633(57.78%)	108,667(28.58%)	17,240(4.53%)
20	22,231(2.92%)	429,733(56.52%)	214,787(28.25%)	31,519(4.15%)
30	33,202(2.91%)	638,143(55.95%)	319,666(28.03%)	46,094(4.04%)
40	44,699(2.94%)	857,303(56.38%)	429,255(28.23%)	61,408(4.04%)
50	56,492(2.97%)	1,078,655(56.75%)	542,314(28.53%)	76,923(4.05%)
60	69,314(3.04%)	1,315,879(57.69%)	664,029(29.11%)	93,660(4.11%)
70	83,532(3.14%)	1,577,616(59.28%)	799,754(30.05%)	112,411(4.22%)
80	91,620(3.01%)	1,745,667(52.45%)	879,688(28.92%)	122,832(4.04%)
90	101,092(2.95%)	1,932,937(56.49%)	969,898(28.35%)	134,886(3.94%)
100	111,587(2.93%)	2,135,576(56.18%)	1,070,988(28.17%)	148,326(3.90%)

For each entry, the first number is the compressed file size required to encode the video up to the frame number, and the number within the parenthesis is the percentage of the compressed file size compared with the size of the raw video sequence.

In Table 3, we have two numbers for each entry. The first number is the number of bytes required to encode the sequence up to the target frame number. The second number is the percentage of the compressed file size in comparison with the raw video sequence. Note that Winzip is a universal coder for arbitrary signals, and XMill is an encoding method for general XML documents. We can see that the proposed MPML compression algorithm provides a substantially better performance than both Winzip and XMill, by exploiting the properties of MPML documents. However, the proposed compression algorithm requires about two times larger file size than the original MPEG-4 bitstream. Therefore, in terms of only compression, the proposed MPML compression algorithm is inferior to the sequential coding of MPEG-4. But, the MPML representation has the advantage of flexibility and random access support. Using these properties, MPML documents can be customized and used to facilitate video communications applications, as will be shown in next two sections.

3. MPML-based robust video transmission

As in the other video coding standards, MPEG-4 also employs variable length coding (VLC) and motion compensated prediction to achieve a high compression performance. However, VLC has a shortcoming that even a single bit error can result in a loss of synchronization and the following bits may not be correctly decoded until the next resynchronization marker. Moreover, transmission errors may propagate to subsequent frames due to the motion compensated prediction. Therefore, various approaches have been proposed to protect image quality against transmission errors [1,3].

The proposed MPML provides a new approach to robust video transmission. Fig. 12 illustrates the proposed MPML-based robust video transmission system. The MPML encoder generates an MPML description as well as an MPEG-4 bitstream.

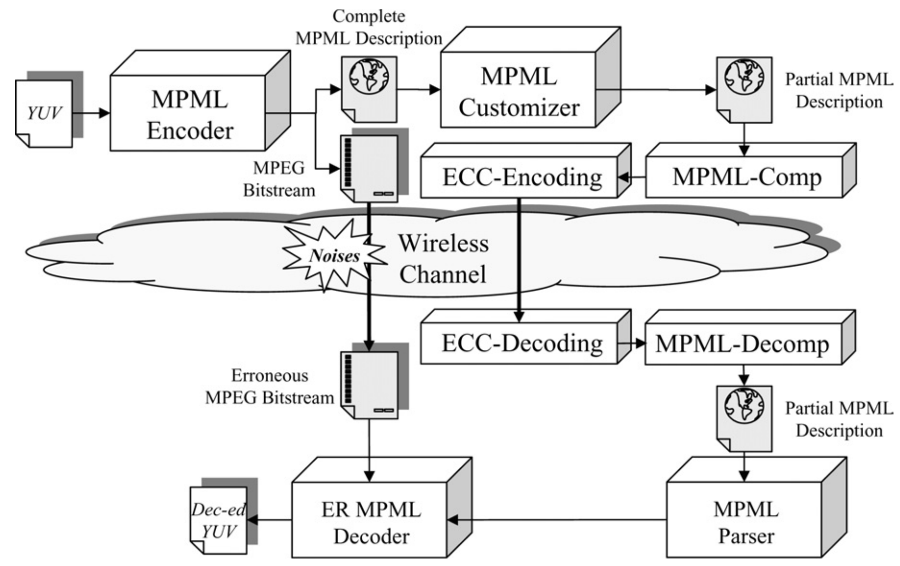


Fig. 12. Robust video transmission system using MPML and adaptive error correction coding (ECC).

Since the MPML description can be randomly accessed, it can be used at the decoder to detect and correct transmission errors in the MPEG-4 bitstream. However, the complete MPML description often requires too much overhead for transmission and can be also corrupted by transmission errors. Therefore, the MPML customizer first tailors it to the partial MPML description according to the importance of video components, and then compress the partial description into binary data. Finally, the compressed MPML description is protected by error correction codes (ECCs) [27] to resist transmission errors.

3.1. MPML customization for robust video transmission

The data in MPEG-4 video can be classified into headers, MVs and DCT coefficients. The header information describes the structural information of MPEG-4 video and plays a more critical role in the decoding process than MVs or DCT coefficients. Thus, the MPML customizer always keeps the header information, while omitting the MVs and DCT coefficients of less important MBs in the partial MPML description to reduce the transmission overhead.

3.1.1. Selection of important MBs

MB losses have different impacts on the overall quality of reconstructed video. For instance, if an MB is not used in the prediction of future frames, its loss causes only temporary degradation. In contrast, if its pixel values are repeatedly used in the prediction of future frames, the overall video quality can be severely degraded by temporal error propagation. In general, the importance of an MB is proportional to the extent to which it is used in the motion compensated prediction.

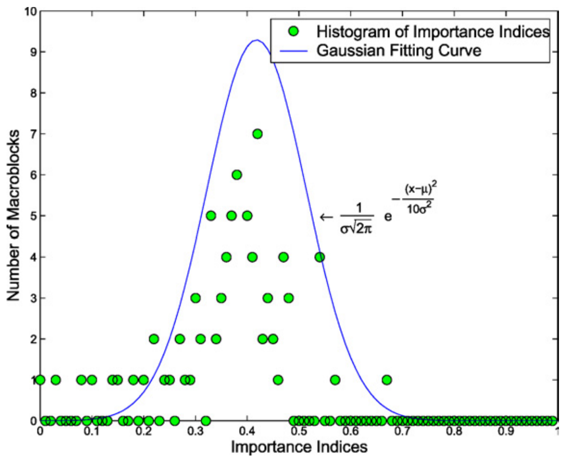


Fig. 13. The distribution of importance indices of MBs in a typical frame.

We adopt the notion of dependency weight, proposed by Kim et al. in [28], to quantify the importance of each MB. Let $M_{n,i}$ denote the ith MB in frame n. Then, the dependency weight $W_{n,i}(j)$ is defined as the normalized number of pixels in $M_{n,i}$ that are used to predict pixels in $M_{n+1,j}$. The importance index $I_{n,i}$ of $M_{n,i}$ is then defined as the sum of $W_{n,i}(j)$'s, given by

$$I_{n,i} = \sum_{j=1}^{K} W_{n,i}(j),$$

where K is the number of MBs in a frame. Note that $I_{n,i}$ indicates how much the loss of $M_{n,i}$ affects the next frame $n+1$.

Fig. 13 shows the distribution of importance indices of MBs in a typical frame. In this work, the distribution is modeled by the Gaussian distribution with mean μ and variance σ^2. The importance indices are computed from MVs as described in [28], and their sample mean and variance are obtained and updated during the encoding. We select a threshold $\theta(k) = \mu + k\sigma^2$, which is parameterized by k. Then, the data for MBs are recorded in the partial MPML description, if their importance indices are higher than the threshold $\theta(k)$.

3.1.2. Error correction coding for customized MPML descriptions

The bit error rate (BER) of a typical wireless link is around 10^{-3}, which is much higher than that of a wired one ($\simeq 10^{-6}$) [29,30]. Therefore, a proper error protection scheme is essential to transmit MPML descriptions reliably over wireless channels. In this work, we develop an unequal error protection method for MPML descriptions to ensure robust transmission while controlling the additional overhead within an acceptable limit.

The nodes in the customized MPML document are protected by error correction codes (ECCs) as follows.

- The header nodes are protected by a (31, 21) BCH code, which can correct up to 2 bit errors for every 21 data bits using 10 parity check bits [27].
- The MB data, including MVs, MB modes and DCT coefficients, are encoded by the (33, 32) parity check code, which can detect a single bit error within 32 data bits using 1 parity check bit.

Note that the header information is more important and thus protected by the strong BCH code. In contrast, the MB data are protected only by the simple parity check code, and their transmission errors are detectable but not correctable.

3.2. Error detection and recovery in the decoder

The conventional MPEG-4 decoder can detect errors in two cases: an illegal bit pattern is encountered or the decoded value does not lie within the legal range. The MPEG-4 standard provides several tools to localize the effect of transmission errors. The start code is a byte-aligned 32-bit code identifying the start of VOP in MPEG-4 bitstream. Start codes can be used to resynchronize the decoding process at the frame level. Data partitioning mode can further localize transmission errors at the video packet (VP) level using resynchronization markers (RMs). However, start codes and RMs can be also corrupted, yielding severe corruption of reconstructed video.

In the proposed algorithm, the customized MPML description is used to reconstruct MPEG-4 video in a robust way. First, the channel decoder detects and corrects possible errors in the MPML description. If data bits contain detectable but not correctable errors, they are discarded. Then, the MPML parser builds up the MPML tree from the corrected description. Since the header nodes are protected by the strong BCH codes, they can be successfully decoded with a high probability and facilitate the random access to the MPEG-4 bitstream.

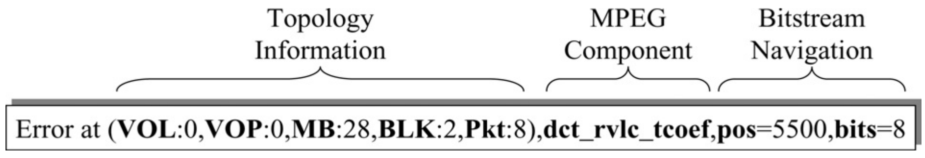
Using the MPML tree, the proposed algorithm can detect transmission errors in the MPEG-4 bitstream as follows. When the decoder decodes a codeword from the bitstream, it searches the MPML tree for the corresponding node. If the node is found and not consistent with the codeword, the decoder declares the codeword as corrupted. If the node is not found, the conventional error detection method is employed.

Table 4 summarizes the types of MPEG-4 decoding errors [22,31]. Note that error types 2, 3, 4, and 5 can be directly detected by comparing the bitstream with the header nodes in the MPML tree. Also, error type 1 can be similarly detected if the block data are present in the MPML tree. Error types 6 and 7 can be detected by checking whether codewords conform to the MPEG-4 syntax.

Fig. 14 shows a typical error record generated by the MPML-assisted error detection algorithm. It informs the decoder of the exact position of the corrupted codeword as well as the number of bits for the codeword. Therefore, the decoder can resume the decoding immediately after the corrupted codeword. This is the main advantage of the MPML-assisted error detection. Notice that in the conventional decoder, an error is often detected after the true position due to the property of variable length codes as shown in Fig. 15. Thus, when an error is detected, the whole video packet is usually discarded to avoid the decoding of false codewords.

Table 4
Categories of MPEG-4 decoding errors

Error type	Description of errors
1	An illegal codeword for DCT coefficient is received or the number of DCT coefficients in a block exceeds 64
2	VO, VOL or GOV headers are corrupted
3	VOP start codes, RMs or motion markers are corrupted
4	VOP parameters are outside valid ranges
5	Parameters in RMs are invalid, i.e., QP is out of range or MB_number is out of range
6	No error is detected while decoding the MBs in the current packet, but the number of the decoded MBs is not consistent with the starting MB number in the next packet header
7	RVLC buffer size is exceeded, or the number of blocks falls outside the valid range in RVLC decoding



```
Error at (VOL:0,VOP:0,MB:28,BLK:2,Pkt:8),dct_rvlc_tcoef,pos=5500,bits=8
```

Fig. 14. An error record generated by the MPML-assisted error detection.

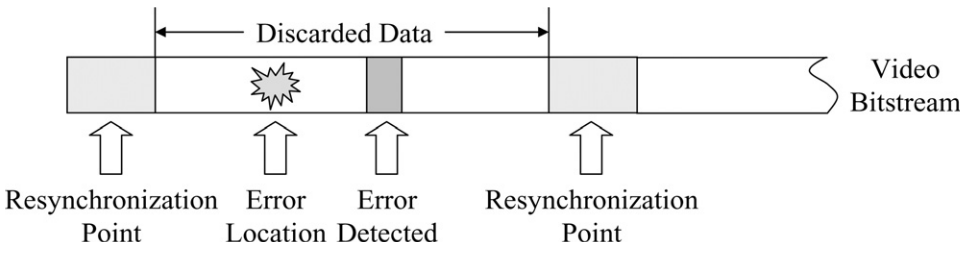


Fig. 15. Error location and its detection point. The whole video packet may be discarded.

Fig. 16 illustrates the proposed reconstruction algorithm for an MB. If the XML tags for the MB are present in the MPML description, their attributes are used to reconstruct the MB. Otherwise, the codewords are decoded from the MPEG bitstream. If errors are detected in the codewords, the corrupted MB is recovered by two error concealment methods. For the damaged inter VOP, the decoded MB in the previous reconstructed frame at the same spatial position is directly copied to the corrupted MB in the current VOP. For the damaged intra-VOP, the pixel values of the erroneous MB are linearly interpolated using the decoded pixel values of its neighboring MBs [32].

3.3. Simulation results

In this section, the performance of the proposed MPML error-resilience algorithm is evaluated on the "Foreman" QCIF sequence, which has a frame rate of

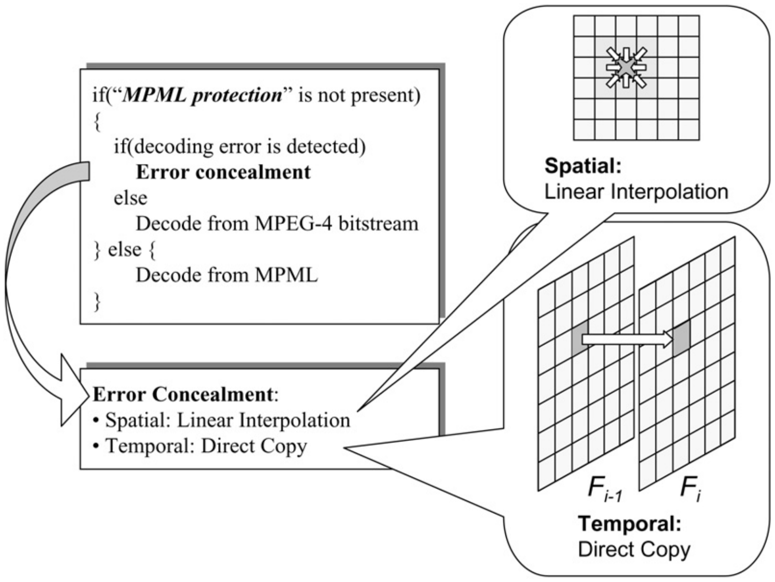


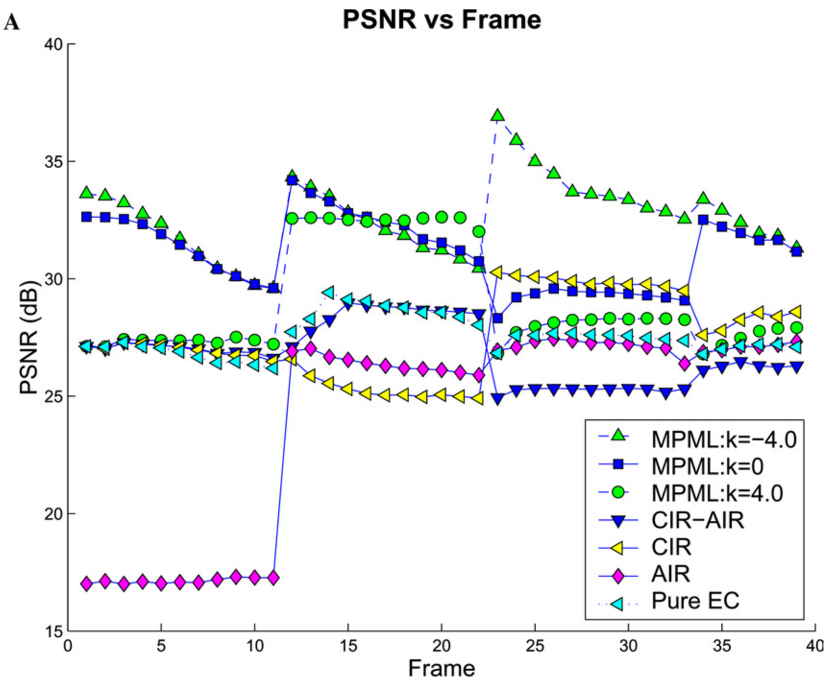
Fig. 16. MPML-based decoding of an MB.

6 frames/s. In the proposed algorithm, an MB is protected by MPML tags, if its importance index is larger than a threshold $\theta(k) = \mu + k\sigma^2$. As k becomes larger, the overhead due to the MPML protection becomes lower. For each k, we choose quantization parameters to make the overall bit rate (for the MPEG-4 bitstream and its MPML protection) 48 kbps. We use the error patterns with random and burst errors by simulating the two-state Gilbert model for wireless fading channels at a BER of 10^{-3} [29,34,35]. We investigate the performance of MPEG-4 in two encoding modes: the default mode and the data partitioning and RVLC (DP-RVLC) mode.

We compare the proposed algorithm with the cyclic intra-refresh (CIR), adaptive intra-refresh (AIR) and CIR-AIR methods [1]. In AIR, the MBs to be encoded in intra-mode is adaptively decided according to the motion activity in each frame. CIR cyclicly decides the timing of intra-refreshment and does not track the motion activities. CIR-AIR is a hybrid method of CIR and AIR to utilize the merits of both methods. The refresh rate is set to 1/3 in all these intra-refresh methods. The pure concealment method, which uses the direct copy and the linear interpolation for the recovery of damaged blocks, is tested also for comparison.

Fig. 17 compares PSNR performances of these methods. The proposed algorithm outperforms the other approaches in the default mode, while it provides worse performances than the intra-refresh approaches (CIR, AIR, and CIR-AIR) in the DP-RVLC mode. This is because transmission errors are more difficult to localize in the default mode. Thus, the random access support of MPML is more beneficiary in the default mode. Also note that the intra-refresh approaches and the MPML protection are complementary to each other and can be used together to suppress temporal error propagation as well as support fast bit error localization.

A



B

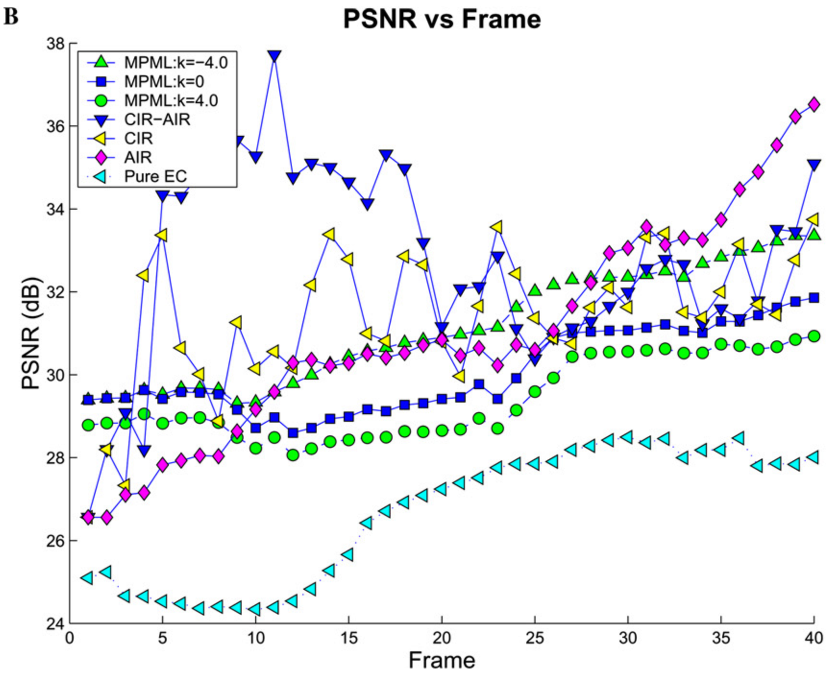


Fig. 17. The PSNR performances of the proposed MPML algorithm and several other approaches: the default (A) and the DP-RVLC (B) modes.

Fig. 18 shows the percentage of the MPML overhead in the total bit rate (=48 kbps) in terms of the parameter k. A small k leads to a large overhead. In the extreme case of $k = -\infty$, all the MB data are protected and the overhead consumes about 60% of the total bit rate. However, as more MB data are protected, the decoder can reconstruct video data more reliably. Fig. 19 shows the PSNR performance of the MPML algorithm as a function of k. Since the channel condition is severe (i.e., BER $= 10^{-3}$), we can see that the proposed algorithm provides a good performance when k has a relatively small value. In general, the parameter k should be selected by considering both the channel condition and the amount of overhead.

In Fig. 20, we compare the 24th frames of the "Foreman" sequence, reconstructed by the proposed algorithm and the other approaches. In this test, the MPEG-4 encoding uses the default mode. We can see that the proposed MPML error resilience algorithm provides a better image quality than the other approaches.

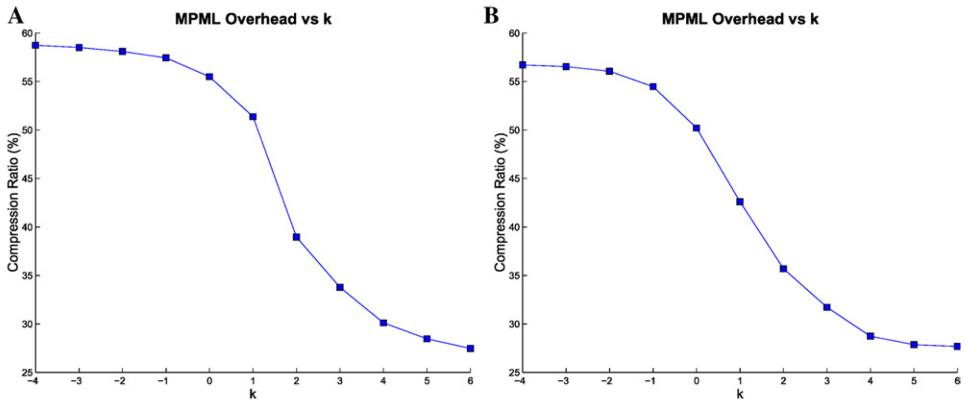


Fig. 18. The percentage of MPML overhead as a function of the parameter k in the default (A) and DP-RVLC (B) modes.

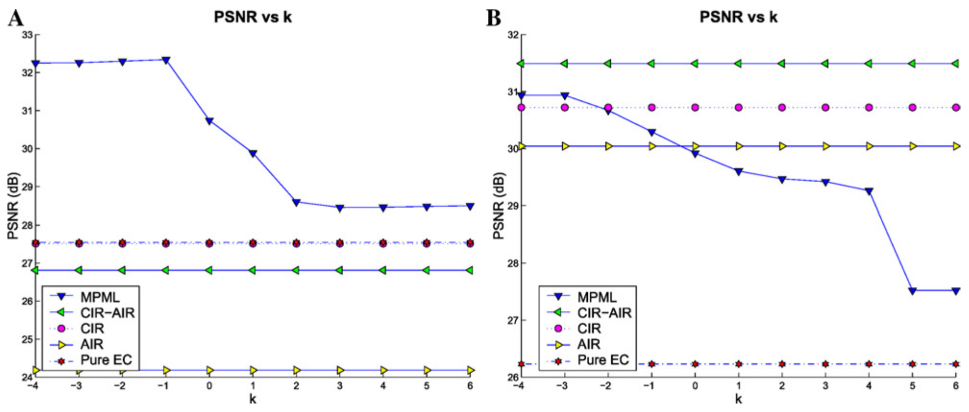


Fig. 19. The PSNR performances of the proposed algorithm in terms of the parameter k in the default (A) and DP-RVLC (B) modes. The performances of other approaches are also shown for comparison.

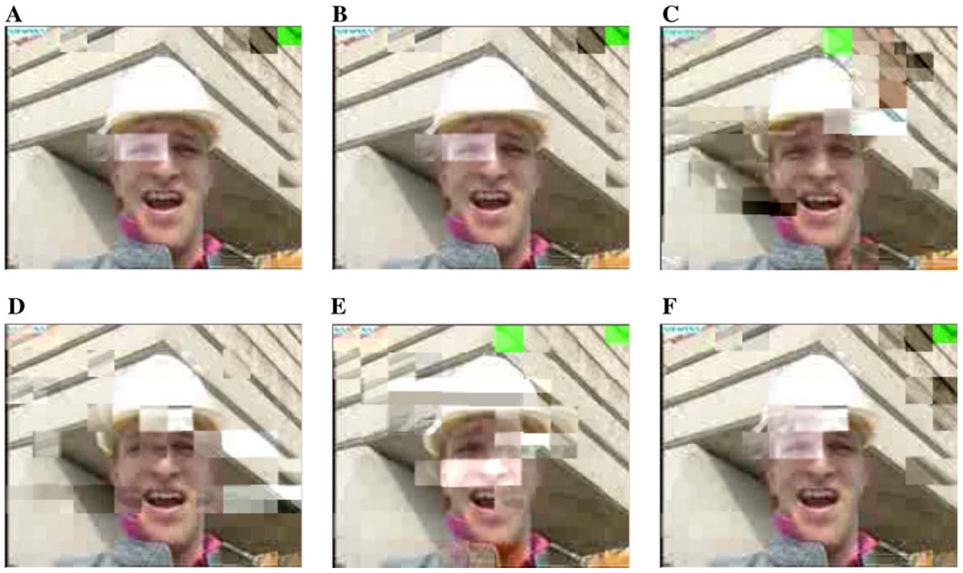


Fig. 20. The "Foreman" 24th frames, obtained by the MPML algorithm at (A) $k = 0$, (B) $k = 2$, (C) AIR, (D) CIR, (E) CIR-AIR, and (F) the pure EC method.

It is also worthwhile to note that the MPML protection has another advantage that it reduces the probability of the decoding failure. Header data can be corrupted at severe error conditions, which often leads to unexpected behaviors of the decoder including the total failure. By protecting the header information and supporting the random access capability, MPML enables the decoder to operate in a highly reliable way.

4. MPML-based multicast resynchronization

Recently, video multicast has received a lot of attention to transmit video information to a group of recipients efficiently. Three approaches have been proposed for video multicast applications [4]. In the first approach, the source transmits a single bitstream to all receivers, and performs the rate control based on the feedback information from the receivers. The main drawback is the feedback implosion, which occurs when several receivers transmit contrary feedback requests. In the second approach, the same video content is encoded into multiple bitstreams with different bit rates and then transmitted to the receivers. The third approach, called layered video multicast [4–6], encodes a video sequence into a base layer and several enhancement layers. The base layer offers a basic level of quality, whereas the enhancement layers provide further refinement information of the video contents. In addition to the base layer, the receiver subscribes to a number of enhancement layers depending on its processing power and bandwidth availability.

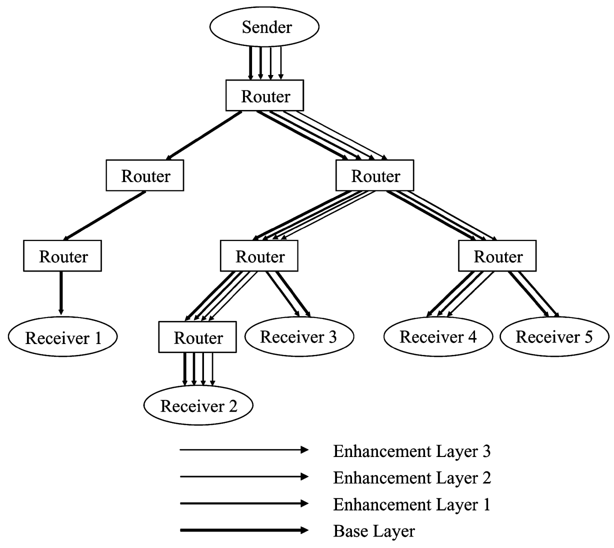


Fig. 21. Illustration of a layered video multicast system.

In this work, we focus on the layered video multicast, which uses limited network resources more effectively. Fig. 21 illustrates the concept of layered video multicast. The sender transmits several layers of compressed bitstream simultaneously. Receiver 1 only subscribes to the base layer to reconstruct a low quality video, while receiver 2 subscribes to the base layer and all the enhancement layers to obtain the highest quality.

Synchronization is a challenging problem in video multicast applications. Since each receiver has a limited buffer to cache bitstreams, the video content should be synchronized to a certain degree among all receivers. In the extreme case when all receivers have zero buffer size, the strict synchronization should be maintained. In this work, we propose the MPML fine grain scalability (FGS) representation to synchronize video stream in a flexible manner.

Another problem in layered video multicast is the loss of video packets. If a bitstream at a certain layer is corrupted by packet losses, then the decoded data at higher layers are of little use. Classical retransmission protocols are not effective in this situation, since the retransmission can degrade the streaming performance to the unaffected receivers. To overcome this problem, the proposed algorithm uses the MPML description to provide extra resynchronization points, where the receivers can resume the normal decoding process.

4.1. MPML customization for multicast resynchronization

Besides the spatial, temporal, and SNR scalabilities, the MPEG-4 standard includes the fine grain scalability (FGS) as a more efficient scalable coding tool [23]. In the spatial, temporal, and SNR scalable coding, the number of layers is pre-deter-

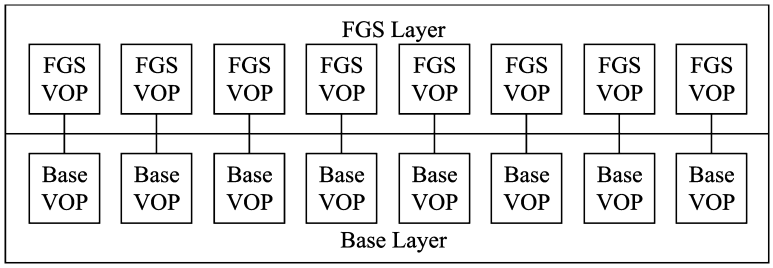


Fig. 22. The FGS coding format.

mined at the encoding time. Thus, a different quality requirement at the decoder may cause the encoder to re-encode the whole video content into a new multi-layer format. FGS provides a more efficient and flexible solution to this problem [33].

The FGS coding format consists of two layers, i.e., a base layer and an enhancement layer, as shown in Fig. 22. The base layer can be independently decoded and provides a low video quality. The enhancement layer contains the DCT coefficients, which represent the difference between the original frame and the reconstructed frame at the base layer decoder. The differential DCT coefficients are encoded using the bit-plane coding scheme. The multicast video server can treat each bit-plane as an enhancement layer.

MPML also supports the FGS mode in the MPEG-4 standard. Fig. 23 shows the complete MPML description of FGS base and enhancement layers. The base layer follows the regular VOP syntax [22], but new components (e.g., 'fgs_layer_type' and 'fgs_ref_layer_id') are introduced into the VOL header. The enhancement layer also has a tree structure, which can be naturally mapped to the XML tree representation. A VOP in the enhancement layer consists of a sequence of FGS bit-planes. Each FGS bit-plane is an array of 64-bit block bit-planes. The 64 bits of each DCT block are zigzag-scanned and encoded into (RUN, EOP) symbols using the run-length coding. Each XML tag is named after the corresponding mnemonics in the MPEG-4 standard documentation and contains three attributes `pos`, `bits`, and `value` in general. The original bitstream can be losslessly reconstructed from the complete MPML description.

We tailor the complete MPML description to a customized version for the multicast synchronization problem. In the customized MPML description, all header nodes are included to provide a random access point at the start of each VOP for the base layer and each bit-plane for the enhancement layer. To minimize the size of the MPML description, we omit all the MPML tags for MB data, i.e., MVs, MB modes, and DCT coefficients.

4.2. Resynchronization in the decoder

We propose a video multicast system based on the MPML FGS description, which is shown in Fig. 24. Concurrently with MPEG-4 bitstreams, the MPML encoder generates partial MPML documents for the base and enhancement layers.

A

```
<?xml version="1.0" encoding="UTF-8"?>
<VisualObject>
  <start_code_prefix pos="0" bits="24" value="1"/>
  <vo_start_code pos="24" bits="3" value="0"/>
  <vo_id pos="27" bits="5" value="0"/>
  <VideoObjectLayer>
    <video_object_layer_start_code pos="32" bits="32"/>
    <start_code_prefix pos="32" bits="24" value="1"/>
    <vol_start_code pos="56" bits="4" value="2"/>
    <vol_id pos="60" bits="4" value="0"/>
    <random_accessible_vol pos="64" bits="1" value="0"/>
    <video_object_type_indication pos="65" bits="8" value="4"/>
    <is_object_layer_identifier pos="73" bits="1" value="1"/>
    …
    <reduced_resolution_vop_enable pos="149" bits="1" value="0"/>
    <scalability pos="150" bits="1" value="0"/>
    <VOP no="0" pos="152">
      <start_code_prefix pos="152" bits="24" value="1"/>
      <vop_start_code pos="176" bits="8" value="182"/>
      <vop_coding_type pos="184" bits="2" value="0"/>
      <modulo_time_base pos="186" bits="1" value="-2"/>
      …
      <vop_quant pos="198" bits="5" value="16"/>
      <macroblock mb="0" pos="203" vop="0">
        <block no="1" mb="0" pos="203">
          <dct_dc_size_luminance mb="0" pos="209" bits="3" blk="1" value="1"/>
          <dct_dc_differential mb="0" pos="212" bits="4" blk="1" value="9"/>
          <dct_tcoef pos="216" bits="8" value="30"/>
          …
          <dct_tcoef_sign pos="343" bits="1" value="1"/>
        </block>
        <block no="2" mb="0" pos="203">
          <dct_dc_size_luminance mb="0" pos="344" bits="3" blk="2" value="1"/>
          …
        </block>
        …
      </macroblock>
      <macroblock mb="1" pos="661" vop="0">
        <block no="1" mb="1" pos="661">
          <dct_dc_size_luminance mb="1" pos="667" bits="3" blk="1" value="3"/>
          …
        </block>
        …
      </macroblock>
      …
    </VOP>
  </VideoObjectLayer>
</VisualObject>
```

B

```
<?xml version="1.0" encoding="UTF-8"?>
<VisualObject>
  <start_code_prefix pos="0" bits="24" value="1"/>
  <vo_start_code pos="24" bits="3" value="0"/>
  <vo_id pos="27" bits="5" value="0"/>
  <VideoObjectLayer>
    <VideoObjectLayerHeader no="0" pos="32"/>
    <video_object_layer_start_code pos="32" bits="32"/>
    <start_code_prefix pos="32" bits="24" value="1"/>
    <vol_start_code pos="56" bits="4" value="2"/>
    <vol_id pos="60" bits="4" value="1"/>
    <random_accessible_vol pos="64" bits="1" value="1"/>
    <video_object_type_indication pos="65" bits="8" value="18"/>
    <fgs_layer_type pos="73" bits="2" value="1"/>
    <video_object_layer_priority pos="75" bits="3" value="2"/>
    …
    <fgs_frequency_weighting_enable pos="137" bits="1" value="0"/>
    <quarter_sample pos="138" bits="1" value="0"/>
    <resync_marker_disable pos="139" bits="1" value="1"/>
    <VOP no="0" pos="144">
      <start_code_prefix pos="144" bits="24" value="1"/>
      <fgs_vop_start_code pos="168" bits="8" value="185"/>
      <fgs_vop_coding_type pos="176" bits="2" value="0"/>
      <modulo_time_base pos="178" bits="1" value="-2"/>
      <marker_bit pos="179" bits="1" value="1"/>
      <vop_time_increment pos="180" bits="5" value="0"/>
      <marker_bit pos="185" bits="1" value="1"/>
      <fgs_vop_max_level_y pos="186" bits="5" value="6"/>
      <fgs_vop_max_level_u pos="191" bits="5" value="5"/>
      <fgs_vop_max_level_v pos="196" bits="5" value="5"/>
      <marker_bit pos="201" bits="1" value="1"/>
      <fgs_vop_number_of_vop_bp_coded pos="202" bits="5" value="10"/>
      <fgs_vop_mc_bit_plane_not_used pos="207" bits="5" value="6"/>
      <fgs_vop_selective_enhancement_enable pos="212" bits="1" value="0"/>
      <fgs_vop_bitplane pos="216" vop="0" bp="0">
        <fgs_bp_start_code_prefix pos="219" bits="24" value="10"/>
        <fgs_vop_bp_id pos="243" bits="5" value="0"/>
        <fgs_vop_bp_block no="0" pos="248" bp="0">
          <fgs_cbp pos="248" bits="6" value="111"/>
          <fgs_vop_bp_escape pos="254" bits="10" value="946"/>
          <fgs_run_eop_code pos="264" bits="7" value="101"/>
          <fgs_sign_bit pos="271" bits="1" value="0"/>
        </fgs_vop_bp_block>
        <fgs_vop_bp_block no="1" pos="272" bp="0">
          <fgs_vop_bp_escape pos="272" bits="10" value="946"/>
          …
        </fgs_vop_bp_block>
        …
      </fgs_vop_bitplane>
      …
    </VOP>
    …
  </VideoObjectLayer>
  <visual_object_sequence_end_code pos="2410968" bits="32" value="433"/>
</VisualObject>
```

Fig. 23. The complete MPML description for MPEG-4 FGS base layer (A) and enhancement layer (B).

The FGS server is in charge of the delivery of the MPEG-4 bitstreams, while the MPML parser sends out the customized MPML documents. The customized MPML documents provide the navigation information to help the MPML decoder randomly access the MPEG-4 bitstreams, but introduces additional transmission overhead. The MPML documents are compressed and protected by ECCs.

At the decoder end, the MPML trees are built up using the MPML documents. From the trees, the MPML parser can extract the synchronization information for all multicast layers at the VOP and the bit-plane levels. Using this information, the MPML decoder applies the same error detection mechanism as discussed in Section 3.2. When errors are detected at a multicast layer, the packet is abandoned and the next resynchronization point is searched from the MPML trees. Then, the decoder can resume the normal decoding process.

4.3. Simulation results

In this test, we simulate the resynchronization problem for video multicast in IP environment, where typical bit rate ranges from 100 kbps to 10 Mbps. The "Foreman" QCIF sequence at a frame rate 6 frames/s is encoded into the base and

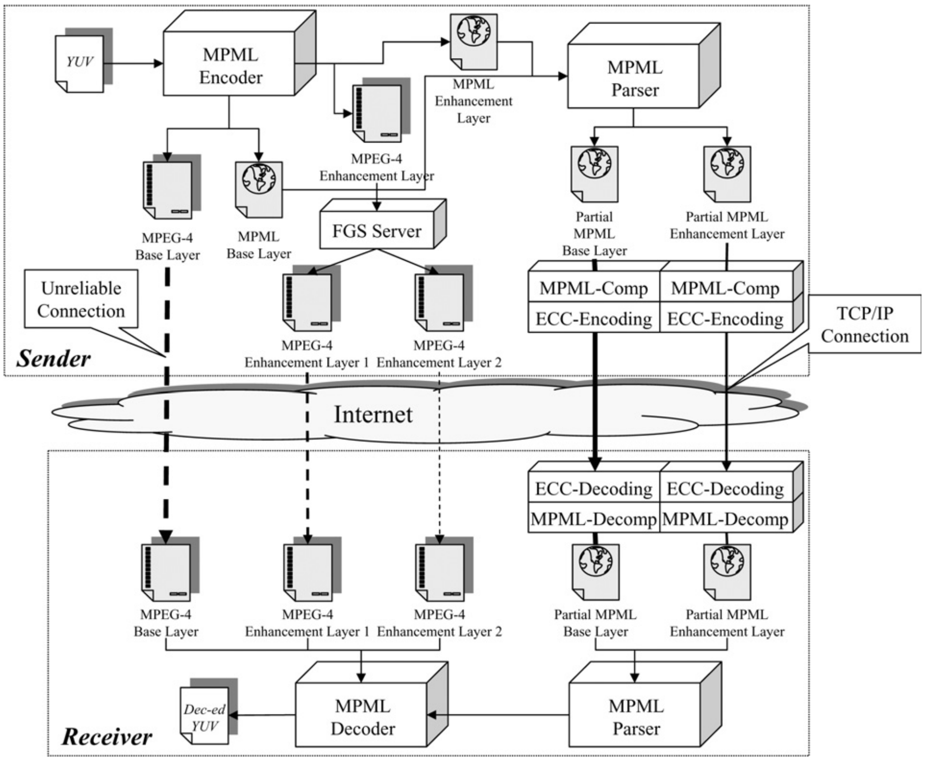


Fig. 24. Illustration of the MPML FGS-based multicast system.

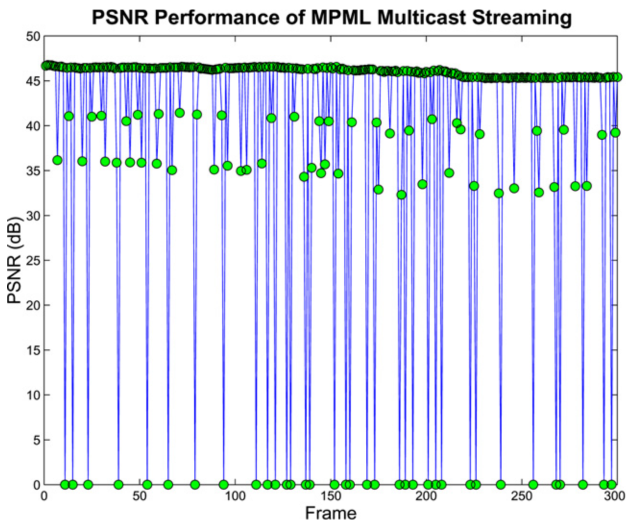


Fig. 25. PSNRs of reconstructed frames in the MPML FGS multicast system.

Table 5
File size comparison of MPEG-4 bitstreams and MPML documents (in bytes)

VOP	MPEG-4 bitstreams		MPML documents		Compressed MPML documents		Overhead ratio (%)	
	Base layer (*A*)	Enhancement layer (*B*)	Base layer	Enhancement layer	Base layer (*C*)	Enhancement layer (*D*)	*C*/*A*	*D*/*B*
50	82,114	1,409,025	25,471	56,264	2,357	3,041	2.87	0.22
75	124,654	2,139,426	37,636	85,079	3,535	4,562	2.84	0.21
100	174,113	2,884,958	50,181	113,591	4,562	6,083	2.62	0.21
125	207,757	3,604,487	62,726	142,750	5,702	7,603	2.74	0.21
150	255,011	4,352,072	75,272	171,756	6,843	9,124	2.68	0.21
175	310,030	5,092,053	87,817	200,915	7,983	10,644	2.57	0.21
200	390,804	5,816,448	100,362	231,746	9,124	12,165	2.33	0.21
225	460,868	6,645,805	112,908	260,029	10,264	13,686	2.22	0.21
250	508,654	7,674,289	125,453	289,302	11,405	14,826	2.24	0.19
275	547,811	8,712,697	137,998	318,232	12,545	16,309	2.29	0.19
300	584,382	9,725,557	150,543	347,618	13,686	17,791	2.34	0.18

Compressed MPML documents mean MPML documents after compression and ECC protection.

enhancement layers by the MPEG-4 FGS encoder. The bit rates of the base and the enhancement layers are 128 and 512 kbps, respectively. These bitstreams are transmitted by the multicast server in Fig. 24.

Both the base layer bitstream and enhancement layers are corrupted at a BER of 10^{-3} using the two-sate Gilbert model [29,34,35]. When errors are detected, the decoded data are discarded. In traditional multicast approaches, it is difficult for the decoder to synchronize the decoding of all the subscribed layers of the next VOP. In contrast, the proposed MPML decoder can easily access the correct positions at all the layers of the next VOP by parsing the MPML trees. Fig. 25 shows the PSNR performances of the reconstructed frames. The PSNR curve deteriorates significantly at the VOPs containing transmission errors. However, their effects are transient and the PSNRs of next VOPs quickly approach to those of the error-free reconstruction.

Table 5 compares the file sizes of MPEG-4 bitstreams and MPML documents. In multicast applications, the MPML document is customized to contain only the header nodes for VOPs and bit-planes, and thus it requires only a small amount of overheads. Moreover, the size of the customized MPML document becomes even smaller after the compression and the ECC protection. We can see that the compressed MPML documents only introduce less than 3.0% and 0.3% overheads for the base layer and the enhancement layer, respectively. Using these small overheads, the MPML document can provide an effective solution to the synchronization problem for video multicasting.

5. Conclusion

In this research, we developed the MPML description format to describe MPEG-4 video contents. As an XML designed specifically for MPEG-4 bitstreams, MPML not only provides an alternative format for describing the semantics of the MPEG standard, but also is equipped with friendly support for random access, portability, interoperability, flexibility, and extendibility in various video applications.

A text-coded MPML document improves readability and interoperability for cross-platform applications but demands a significant amount of overhead. Since the document is textual and of a tree structure, it is possible to develop a compression algorithm to reduce the amount of overhead. We developed an efficient compression algorithm for MPML documents, which provides a much higher coding gain than Winzip and a few XML-aware coders.

In real-world applications, the MPML description can be adaptively tailored with a different degree of details to meet the requirements of a specific problem. In this work, we demonstrated how to use MPML to tackle the error-resilient problem and the multicast resynchronization problem, which are critical in video delivery over unreliable transmission channels. It was shown that the MPML-based decoding algorithm provides elegant solutions to these two problems by using the random access support of MPML.

References

[1] Y. Wang, S. Wenger, J. Wen, A.K. Katsaggelos, Error resilient video coding techniques, IEEE Signal Process. Mag. 17 (2000) 61–82.
[2] W3C Candidate Recommendation, Extensible Markup Language (XML) 1.0 (2nd ed.). Available from: <http://www.w3.org/TR/REC-xml>, 2004.
[3] B. Girod, N. Färber, Feedback-based error control for mobile video transmission, Proc. IEEE 87 (10) (1999) 1707–1723.
[4] X. Li, M.H. Ammar, S. Paul, Video multicast over the Internet, IEEE Network 13 (2) (1999) 46–60.
[5] D. Wu, Y.T. Hou, W. Zhu, Y. Zhang, J.M. Peha, Streaming video over the Internet: approaches and directions, IEEE Trans. Circuits Syst. Video Technol. 11 (3) (2001) 282–300.
[6] S. McCanne, V. Jacobson, M. Vetterli, Receiver-driven layered multicast, ACM SIGCOMM Comput. Commun. Rev. 26 (4) (1996) 117–130.
[7] X. Sun, Z. Shi, C.-C.J. Kuo, XML-based MPEG-4 video representation, streaming and error resilience, in: Proceedings of SPIE 16th Annual International Symposium on AeroSense 4736, 2002, pp. 139–150.
[8] X. Sun, Z. Shi, C.-C.J. Kuo, XML-based MPEG-4 video representation and error resilience, Proc. IEEE Int. Symp. Circuits Syst. 2 (2002) 676–679.
[9] X. Sun, C.-C.J. Kuo, An MPEG-4/XML FGS approach to multicast video synchronization, in: Proceedings of IS & T/SPIE 15th Annual Symposium, Electronic Imaging, 5018, 2003, pp. 284–295.
[10] X. Sun, C.-C.J. Kuo, Multicast video synchronization via MPEG-4 FGS/XML representation, Proc. IEEE Int. Symp. Circuits Syst. 2 (2003) 820–823.
[11] X. Sun, C.-C.J. Kuo, Performance evaluation of MPML-based error resilient video transmission, in: Proceedings of SPIE 17th Annual International Symposium on AeroSense, 5108, 2003, pp. 263–274.
[12] X. Sun, C.-C.J. Kuo, MPML-based error resilient wireless video transmission, Proc. SPIE ITCOM 5241 (2003) 111–122.
[13] M. Kim, S. Wood, L. Cheok, Extensible MPEG-4 Textual Format (XMT), in: Proceedings of ACM Workshops on Multimedia, 2000, pp. 71–74.
[14] Web3D Consortium, X3D: The Virtual Reality Modeling Language—International Standard ISO/IEC 14772:200x. Available from: <http://www.web3d. org/TaskGroups/x3d/specification>, 2001.
[15] W3C Recommendation, Synchronized Multimedia Integration Language 1.0 Specification. Available from: <http://www.w3.org/TR/REC-smil>, 1998.
[16] W3C Recommendation, Scalable Vector Graphics 1.0 Specification. Available from: <http://www.w3.org/TR/SVG/>, 2001.
[17] S. Chang, T. Sikora, A. Puri, Overview of the MPEG-7 standard, IEEE Trans. Circuits Syst. Video Technol. 11 (6) (2001) 688–695.
[18] T. Sikora, The MPEG-7 visual standard for content description—an overview, IEEE Trans. Circuits Syst. Video Technol. 11 (6) (2001) 696–702.
[19] P. Salembier, J.R. Smith, MPEG-7 multimedia description schemes, IEEE Trans. Circuits Syst. Video Technol. 11 (6) (2001) 748–759.
[20] J.H. Coombs, A.H. Renear, S.J. DeRose, Markup systems and the future of scholarly text processing, Commun. ACM 30 (11) (1987) 933–947.
[21] R. Price, Beyond SGML, in: Proceedings of the 3rd ACM Conference on Digital Libraries, 1998, pp. 172–181.
[22] MPEG-4 Video Group, Information Technology—Generic Coding of Audio-visual Objects—Part 2: Visual (ISO/IEC FDIS 14496-2, N2502), ISO/IEC JTC1/SC29/WG11, 1998.
[23] MPEG-4 Video Group, MPEG-4 Video Verification Model Version 18.0 (N3908), ISO/IEC JTC1/SC29/WG11, 2001.
[24] H. Liefke, D. Suciu, XMill: an efficient compressor for XML data, in: Proceedings of the ACM International Conference on Management of Data, 2000, pp. 153–164.
[25] J. Cheney, Compressing XML with multiplexed hierarchical PPM models, Proceedings of IEEE Data Compression Conference, 2001, pp. 163–172.

[26] X. Sun, Design and Applications of MPEG Video Markup Language (MPML), University of Southern California, 2004.
[27] S. Lin, D.J. Costello, Jr., Error Control Coding: Fundamentals and Applications, Prentice-Hall Inc., Eglewood Cliffs, NJ, 07632, 1983.
[28] J.-G. Kim, J. Kim, C.-C.J. Kuo, Corruption model of loss propagation for relative prioritized packet video, Proc. SPIE Appl. Digital Image Process. XXIII 4115 (2000) 214–224.
[29] E.N. Gilbert, Capacity of a burst-noise channel, Bell Syst. Tech. J. 39 (1960) 1253–1265.
[30] B.P. Crow, I. Widjaja, J.G. Kim, P.T. Sakai, IEEE 802.11 wireless local area networks, IEEE Commun. Mag. 35 (9) (1997) 116–126.
[31] MPEG-4 Video Group, Description of Error Resilient Core Experiments (N1996), ISO/IEC JTC1/SC29/WG11, 1998.
[32] S. Aign, K. Fazel, Temporal & spatial error concealment techniques for hierarchical MPEG-2 video codec, Proc. IEEE Int. Conf. Commun. 3 (1995) 1778–1783.
[33] R. Yan, F. Wu, S. Li, R. Tao, Y. Wang, Y. Zhang, Error robust coding for the FGS enhancement bitstream, in: Proceedings of the 3rd International Conference on Information, 2001.
[34] J.R. Yee, E.J. Weldon Jr., Evaluation of the performance of error-correcting codes on a Gilbert channel, IEEE Trans. Commun. 43 (8) (1995) 2316–2323.
[35] S. Gringeri, R. Egorov, K. Shuaib, A. Lewis, B. Basch, Robust compression and transmission of MPEG-4 video, in: Proceedings of the 7th ACM International Conference on Multimedia, 1999, pp. 113–120

Biography of Dr. Xiaoming Sun. Xiaoming Sun received the B.S. degree in Computer Science from the Beijing University of Science and Technology, China, in 1993, two M.S. degrees in Computer Science from the Peking University in 1996 and from the University of Southern California in 2000, respectively, and the Ph.D. degree in Electrical Engineering from the University of Southern California in 2004. From 2004, he joined ESS Technology, Inc. as a senior software engineer and project developer. His research interests are in the areas of digital video processing, multimedia compression and embedded multimedia system design.

Biography of Dr. Chang-Su Kim. Chang-Su Kim received the B.S. and M.S. degrees in control and instrumentation engineering in 1994 and 1996, respectively, and the Ph.D. degree in electrical engineering in 2000, all from Seoul National University (SNU), Seoul, Korea. From 2000 to 2001, he was a Visiting Scholar with the Signal and Image Processing Institute, University of Southern California, Los Angeles, and a Consultant for InterVideo Inc., Los Angeles. From 2001 to 2003, he was a Postdoctoral Researcher with the School of Electrical Engineering, SNU. In August 2003, he joined the Department of Information Engineering, the Chinese University of Hong Kong as an Assistant Professor. His research topics include video and 3-D graphics processing and multimedia communications. Dr. Kim has published more than 70 technical papers in international conferences and journals.

Biography of Dr. C.-C. Jay Kuo. Dr. C.-C. Jay Kuo received the B.S. degree from the National Taiwan University, Taipei, in 1980 and the M.S. and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, in 1985 and 1987, respectively, all in Electrical Engineering. He is with the Department of Electrical Engineering, the Signal and Image Processing Institute (SIPI) and the Integrated Media Systems Center (IMSC) at the University of Southern California (USC) as Professor of Electrical Engineering and Mathematics. His research interests are in the areas of digital media processing, multimedia compression, communication, and networking technologies, and embedded multimedia system design. Dr. Kuo is a Fellow of IEEE and SPIE. He received the National Science Foundation Young Investigator Award (NYI) and Presidential Faculty Fellow (PFF) Award in 1992 and 1993, respectively.

Dr. Kuo has guided about 60 students to their Ph.D. degrees and supervised 15 postdoctoral research fellows. Currently, his research group at USC consists around 40 Ph.D. students and five postdoctors (visit website http://viola.usc.edu), which is one of the largest academic research groups in multimedia technologies. He is a co-author of about 100 journal papers, 600 conference papers and seven books. Dr. Kuo is Editor-in-Chief for the *Journal of Visual Communication and Image Representation*, and Editor for *the*

Journal of Information Science and Engineering and *the EURASIP Journal of Applied Signal Processing*. He was on the Editorial Board of the *IEEE Signal Processing Magazine* in 2003–2004. He served as Associate Editor for *IEEE Transactions on Image Processing* in 1995–1998, *IEEE Transactions on Circuits and Systems for Video Technology* in 1995–1997 and *IEEE Transactions on Speech and Audio Processing* in 2001–2003.