International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:5, 2007

# Enhanced Frame-based Video Coding to Support Content-based Functionalities

Prabhudev Hosur and Rolando Carrasco

***Abstract*****—** This paper presents the enhanced frame-based video coding scheme. The input source video to the enhanced frame-based video encoder consists of a rectangular-size video and shapes of arbitrarily-shaped objects on video frames. The rectangular frame texture is encoded by the conventional frame-based coding technique and the video object's shape is encoded using the contour-based vertex coding. It is possible to achieve several useful content-based functionalities by utilizing the shape information in the bitstream at the cost of a very small overhead to the bitrate.

***Keywords*****—** Video coding, content-based, hypervideo, interactivity, shape coding, polygon.

## I. Introduction

OVER the years, the digital video technology has been evolving quickly in terms of the ways video content is produced, delivered, and consumed. As a result, users are not content with only the compression efficiency in a video coding technology; they are now demanding more features. To meet the user demands, the new video coding schemes are required support content-based interactivity, allow content-based video indexing and retrieval, cope up with bandwidth and bit error rates of transmission networks, and achieve backward compatibility with existing video coding schemes in addition to providing a very good compression efficiency. For this purpose, the new video encoders require the source input video to be in such a form that the video content can be easily identified and characterized.

The input video source to the object-based video coding approach in MPEG-4 standard [1] is in the form of arbitrarily shaped video objects and their shapes as shown in Fig. 1(a). The MPEG-4 object-based video facilitates user-interactivity with individual objects. However, the decoders based on earlier MPEG standards are not capable of decoding the MPEG-4 object-based video bitstreams. Furthermore, the accurate segmentation of semantically meaningful objects from a rectangular video still remains a challenging problem. On the other hand, inaccurate segmentation of video objects may adversely affect the compression efficiency [2].

In the sub-picture coding [3], a video frame is partitioned into one or more user-defined non-overlapping rectangular sub-pictures and a remaining background picture (see Fig. 1(b)). It is possible to assign different error protection and quality levels to different sub-pictures based on their importance by using sub-picture coding. However, the rectangular sub-pictures do not generally represent the true video objects with which a user would like to interact.

Some video transmission schemes suggest encapsulating the MPEG-2 and the MPEG-4 video streams within the MPEG-2 transport stream so as to provide the value-added services such as content-based interactivity [4]. Although these schemes provide the backward compatibility with the existing conventional MPEG-2 decoders, they have the disadvantage of substantially increasing the cost of the decoders in order to support more than one MPEG standard.

(a) (b)

Fig. 1. Examples of input source video formats for (a) the MPEG-4 object-based coding, and (b) the sub-picture coding.

The objectives of the proposed enhanced frame-based video encoder (EFBE) are the following: 1) to serve as a simple extension of the conventional frame-based video encoder architectures and to provide backward compatibility, 2) to achieve nearly the same performance in terms of video quality and compression ratio as compared to the conventional frame-based video coding performance, and 3) to address the user demands for content-based functionalities. The architecture, the features and the performance of the enhanced frame-based coding scheme are described in the following sections.

## II. Proposed Enhanced Frame-Based Video Coding

In the proposed enhanced frame-based coding, the input source video consists of a rectangular-size video and shapes of arbitrarily-shaped objects on video frames. We define the *object of interest* (OOI) in a rectangular frame video as the arbitrarily-shaped semantically meaningful video object which is of interest to a user. There are three steps involved in the process of obtaining the coded representation of video using the enhanced frame-based video coding scheme:1) pre-processing, 2) encoding shape and texture, and 3) post-processing. In the first step, an OOI is identified and its shape information is obtained in the form of either an outline contour sketch or a segmentation mask. The outline contour sketch of an OOI can be obtained by marking the outline of OOI on the screen manually; this way the shape information can be easily generated by a user. Several techniques have been proposed to obtain the segmentation masks using chroma-key [5], automatic segmentation [6] and semi-automatic segmentation techniques [7]. For the proposed EFBE, the boundaries of segmented video objects need only represent an approximate outline of the area belonging to a video object and there is no requirement for segmentation to be accurate. In the second step, the rectangular frame texture and the video object's shape are encoded. The block diagram of the EFBE is shown in

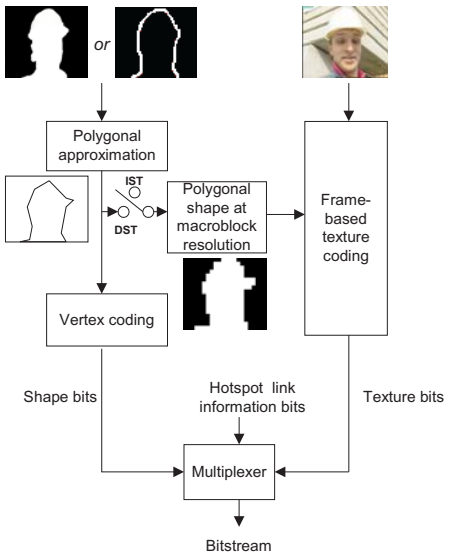



Fig. 2. The enhanced frame-based video encoder.

Fig. 2. The post-processing step involves associating the shape information with the texture of the object it represents. The shape of an OOI identifies the area belonging to the OOI on the rectangular frame video so that the area behaves as a hotspot on the video frame. This hotspot is associated with a link to another destination similar to a hyperlink in the World Wide Web. This link can be to either another video (which may or may not contain links) or an HTTP object such as a web page, file, or CGI script. The tracking as well as the link information is multiplexed into the bitstream along with texture and shape information.

In this paper, we focus mainly on the second step to propose a new video encoding scheme to enable content-based functionalities whatever the methods used for pre-processing and post-processing.

### *A. Encoding shape and texture*

As shown in Fig. 2, the major components of the proposed EFBE are frame-based texture coding and shape coding. The shape coding is performed in two steps: 1) polygonal approximation of shape boundary, and 2) vertex coding. The texture and shape bits are multiplexed into a single bitstream. The EFBE has two modes of operation: independent-shape-texture (IST) mode and dependent-shape-texture (DST) mode. The *IST/DST* switch allows the switching between the two modes. In the IST mode, the rectangular frame texture and video object's shape are encoded independently. Whereas in the DST mode, the shape information is utilized to adjust the quantization parameter of texture macroblocks.

*1) Shape coding:* Unlike the MPEG-4 which uses bitmap-based techniques for shape coding, the proposed EFBE employs contour-based technique which lends itself to the semantic shape characterization. The shape contour in the form of either the segmentation mask's boundary or the outline contour sketch of OOI is approximated by a polygon using *sequential method* such that the distance between the polygon and the contour is less than or equal to a given tolerable approximation

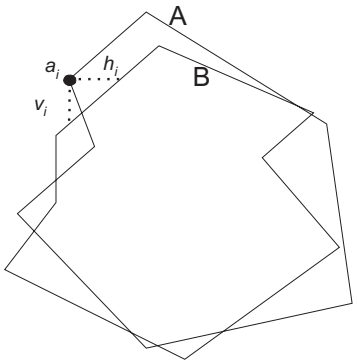



Fig. 3. The horizontal and vertical distances of the vertex $a_i$ belonging to polygon $A$ from polygon $B$.

error $\delta$. In sequential method, the contour is scanned from an initial point on the curve to determine the longest possible edge having the approximation error less than or equal to $\delta$. The process is repeated with the end point of the current edge as the next starting point. The vertices of the polygon are coded using the object adaptive vertex encoding method described in [8]. The amount of shape distortion is controlled by varying the value of $\delta$; the larger the value of $\delta$, the higher the shape distortion. Lossless shape coding is achieved by setting $\delta = 0$.

The distance between the polygonal approximations of OOI shape in the current and the reference frame is used to detect the amount of temporal variation in shape. The distance between two polygons is computed as follows. Let $A$ and $B$ be the polygonal approximations of the current and the reference OOI shape. Let $h_i$ and $v_i$ be the horizontal and vertical distance of $i$th vertex of $A$ from $B$ (see Fig. 3). Then the distance of $A$ from $B$ is defined as

$$D_{AB} = \max_i \ (d_i), \tag{1}$$

where $d_i = min(h_i, v_i)$. For the objects of interest such as *talking-head*, the variation in OOI shape over a sequence of contiguous frames is usually very small. Therefore when the lossy shape coding is desired, the shape information is not transmitted with every frame. Instead, the polygonal approximation of the current OOI shape is coded and transmitted only if the distance between the polygonal approximations of the current and reference OOI shapes is greater than or equal to a threshold $T$. Otherwise no shape information is transmitted in the current frame. At the decoder, the most recently decoded shape is used to identify the OOI if no shape information is present in the current frame.

*2) Texture coding:* The basic steps of texture coding in the EFBE are essentially the same as those in a typical rectangular frame-based encoder. These basic steps consist of dividing a video frame into an array of basic units called macroblocks and processing each macroblock by applying discrete cosine transform, quantization and variable length coding. In fact, the texture coding block in the EFBE can be any rectangular frame-based video encoder (e.g., MPEG-1, MPEG-2 or MPEG-4 (simple profile) video encoder). In our implementation of EFBE, we have used MPEG-4 (simple profile) encoder for frame-based video texture coding.

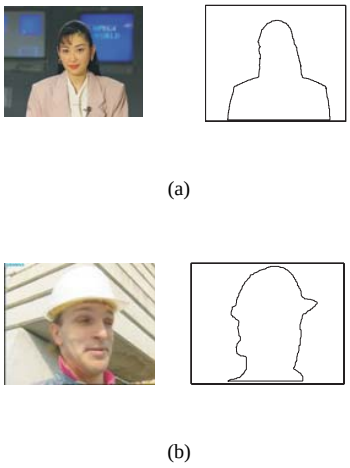
In the DST mode, the shape information is used to adjust the texture coding parameters. The main idea is to encode the texture in the region belonging to the polygonal approximation of the OOI shape with a finer quantization as compared to the rest of the video frame. The quantizer values used for the two regions are embedded in the header information which is appended to the shape bitstream. At the decoder, the decoded shape information is utilized to correctly identify the region belonging to video object on the video frame and the quantizer information in the bitstream header is utilized for decoding the frame texture.

*3) Multiplexing shape and texture bits:* The proposed enhanced frame-based video coding scheme involves the multiplexing texture bits with the additional bits consisting of the shape bits and the bits that are generated by the post-processing stage for tracking and linking the objects of interest. In order to achieve backward compatibility with the conventional frame-based coding, we need to place the additional bits into the bitstream such that the conventional frame-based decoders would simply ignore these additional bits and decode the texture bits as usual. The proposed enhanced frame-based video decoders would utilize the additional bits to provide the content-based functionalities. We employ the user data packet insertion scheme for MPEG-4 is described in [9] for this purpose. We combine the shape bits, additional header bits and the link information bits into user data and place the user data into the bitstream generated by the frame-based texture coding block of the EFBE.

## III. Features of the EFBE

The architecture of the proposed EFBE is designed such that it can be implemented as a simple extension of an existing frame-based encoder architecture. The only additional complexity that EFBE adds to a frame-based encoder is that required for shape coding. When a EFBE bitstream is received by an existing conventional frame-based video decoder, the shape information present in the bitstream would simply be ignored and only rectangular frame texture would be decoded and displayed. Since the shape and texture are encoded independently in the IST mode operation of EFBE, the IST mode provides the backward compatibility with the conventional frame-based video decoders. However, the DST mode of EFBE does not support backward compatibility with the existing conventional frame-based decoders because the texture coding is dependent on the shape information in this mode of operation.

The embedded shape information in the bitstream of the EFBE can be utilized to support several content-based functionalities. At the receiver, the shape information of a video object in the bitstream facilitates the identification of the region belonging to the object as a hotspot on the rectangular frame. A hyperlink can be provided for the object on the rectangular frame when a user activates the object by clicking on the hotspot. Thus the decoded video functions almost like a Web page, allowing people to interact with the picture on the screen. Such a video bitstream in which a link and content information is associated with a region on a video frame is generally referred to as HyperVideo [10]. This kind of application does not require the shape of OOI to be accurate; therefore input shape information to the encoder can be in the form of inaccurate segmentation mask generated by simpler and practically viable segmentation techniques or rough contour sketch obtained by manually tracing the boundaries of OOI. Furthermore, lossy shape coding can be employed to achieve higher compression.



(a)

(b)

Fig. 4. The source videos used in our experiments: a) the first frame in *Akiyo* and the corresponding shape contour of the OOI, and b) the third frame in *Foreman* and the corresponding shape contour of the OOI,

The annotation of video hyperlinks in the form of small icons containing the polygonal approximation of object contour can be displayed at the bottom of the rectangular video for a user to identify the hot spots in a frame. Here, the annotations give an idea about the current, previous and next scenes to a user through the display of polygonal approximations that provide semantic description of the objects in the scenes. This will allow a user to easily search for a scene in a video. Furthermore, during fast-forward or fast-reverse operations, only shape can be decoded and displayed instead of decoding the entire rectangular frame texture.

If the boundaries of the shape mask perfectly match the true boundaries of OOI (as in the case of shape information obtained through the standard blue-screen technique in studio environments), one may choose the lossless shape coding by setting the polygonal approximation error $\delta = 0$. This allows the user to extract the OOI from the rectangular frame and overlay the OOI on a background picture of his own choice to display a video scene which is different from the one present in the bitstream.

The DST mode of the EFBE is capable of providing additional content-based functionalities such as assigning different quality, compression, and error protection levels to the regions of interest and the remaining areas in a video frame. For example, the region belonging to an OOI can be encoded at a higher quality by using finer quantization during texture coding and the remaining region in the frame can be encoded at a relatively lower quality to balance the bit budget.

## IV. Experimental results

The conventional frame-based video encoder (CFBE) in the MPEG-4 Verification Model (VM) software [1] is used as the basis to implement the proposed EFBE. We incorporated the

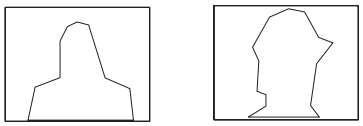


Fig. 5. Polygonal approximations of the original contours shown in Fig. 4 for given tolerable approximation error $\delta = 10$.

TABLE I
COMPARISON OF PERFORMANCES OF CFBE AND THE IST MODE OF EFBE. WE SET $\{\delta = 10, T = 5\}$ FOR THE *Akiyo*, AND $\{\delta = 10, T = 0\}$ FOR THE *Foreman* DURING SHAPE CODING IN THE IST MODE OF THE EFBE.

| Coding scheme | Quantizer ($Q$) | CFBE | EFBE (IST) | |
|---|---|---|---|---|
| | | Av. Bits/frame | Av. Bits/frame | |
| | | Texture | Texture | Shape |
| *Akiyo* | 8 | 102961 | 102961 | 21 |
| | 16 | 5924 | 5924 | 21 |
| *Foreman* | 8 | 176584 | 176584 | 220 |
| | 16 | 16941 | 16941 | 220 |

following modifications to VM software: 1) addition of our shape coding module, 2) adjustment of quantization step based on shape information in the DST mode, and 2) multiplexing the user data consisting of encoded shape information and the link information of hotspots with the texture and header data.

We use the first 100 frames of the 30Hz CIF-size *Akiyo* and *Foreman* video sequences and associated OOI shape contours in our experiments. Fig. 4 shows a sample frame and associated OOI shape contour in the original videos. The video sequences are encoded at 10 frames/sec; so there are 34 coded frames in the bitstream.

First, we compare the performance of the IST mode of the EFBE with the performance of the CFBE. Since the texture coding in the IST mode of EFBE is the same as that in the CFBE, both the encoders yield the same video quality. Therefore, we compare only the bitstream size. The following encoder settings are used. A fixed quantization step of $Q$=16 is used during texture coding. For the EFBE, we need to specify two additional parameters namely $\delta$ and $T$ associated with shape coding. In our experiments, we set $\delta = 10$ and we use $T = 0$ for *Akiyo* and $T = 5$ for the *Foreman*. The Fig. 5 shows the polygonal approximation of OOI in a frame for $\delta = 10$. The bitrates are listed in Table I. Since $T = 5$ is used for coding the shape of *Akiyo*, we observed in our tests that the OOI shape was encoded in only four out of the 100 frames and the total number shape bits in the entire bitstream was 720; thus the average number of shape bits per frame is $720/34 \simeq 21$, which is a negligibly small value as compared to the average number of texture bits. In case of the *Foreman*, the shape of OOI in each frame is encoded because $T = 0$. We observe that the shape bits are 0.125% and 1.29% of the texture bits for $Q = 8$ and $Q = 16$, respectively. However, this small additional overhead of shape bits in the EFBE bitstream as compared to the CFBE bitstream is greatly justified by the benefit achieved in terms of several useful content-based functionalities that the shape information enables.

We present the comparison of the performance of the DST mode of the EFBE with that of the CFBE using the *Foreman* video. A fixed quantization step of $Q$=16 is used for all the macroblocks in CFBE. Whereas in the DST mode operation of EFBE, a lower quantizer ($Q$=8) is used for OOI region and higher quantizer ($Q$=31) is used for the remaining part of the frame to achieve nearly the same texture bits as that required by CFBE. For the shape coding in the DST mode of the EFBE, we set $\{\delta = 10, T = 0\}$. The quality and bitrate for the first Intra-frame encoded by the two encoders are presented in Table II. A higher PSNR for the OOI region is achieved at the cost of lower PSNR for the background region as compared to the overall PSNR obtained with the CFBE. The EFBE requires additional 128 bits for coding the shape of the OOI in this frame.

TABLE II
COMPARISON OF PERFORMANCE OF CFBE AND THE DST MODE OF EFBE FOR INTRA-CODING OF FIST FRAME OF THE *Foreman*

| Coding scheme | Quantizer ($Q$) | Bits | | PSNR in dB |
|---|---|---|---|---|
| | | Texture | Shape | |
| CFBE | 16 | 34248 | - | 30.11 |
| EFBE (DST) | 8 (OOI)<br>31 (Background) | 34532 | 228 | 31.85 (OOI)<br>29.11 (Background) |

## V. CONCLUSIONS

The architecture and design of the proposed enhanced frame-based video encoder is presented. The main aim of the proposed encoder is to provide an enhancement to the conventional frame-based coding. It is possible to achieve several useful content-based functionalities by embedding the coded representation of a video object's contour along with the coded texture in the bitstream,. The overhead of additional bits required for shape coding is less than 2% of the total bits of the conventional frame-based coding in our experimental results.

## REFERENCES

[1] MPEG Video Group, "MPEG-4 Verification Model (VM) Version 14.0," *ISO/IEC JTC1/SC29/WG11 N2932*, Oct. 1999.
[2] L. Cheng and M. E. Zarki, "The Analysis of MPEG-4 Core Profile and Its System Design," *Proceedings of the IEEE International Conference on Multimedia Technology and Applications (MTAC)*, Nov. 2001.
[3] M. M. Hannuksela, Ye-Kui Wang, and M. Gabbouj, "Sub-picture video coding for unequal error protection," *XI European Signal Processing Conference (EUSIPCO)*, France, 03-06 Sep. 2002.
[4] E. Asbun and Siu-Wai Wu, "Implementation of an MPEG-4 enhanced digital TV set-top terminal" *International Conference on Consumer Electronics (ICCE)*, Jun. 2001, pp. 144-145.
[5] K. Panusopone and X. Chen, "A Modified Chroma Key Technique," *Thirty-Third Asilomar Conference on Signals, Systems and computers 1999*, pp. 735-739, vol. 1, 1999.
[6] T. Meier and K. N. Ngan , "Automatic Segmentation of Moving Objects for Video Object Plane Generation," *IEEE Trans. Circuits Syst. Video Technol.*, pp. 525-538, vol. 8, Sep. 1998.
[7] C. Toklu, M. Tekalo and T. Erdem , "Semi-automatic Video Object Segmentation in the presence of occulsion," *IEEE Trans. Circuits Syst. Video Technol.*, pp. 624-629, vol. 104, Jun. 2000.
[8] K. J. O'Connell, "Object-adaptive vertex-based shape coding method," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, no. 1, Feb. 1997, pp. 251-255.
[9] S. T. Worrall, A. H. Sadka, P. Sweeney, and A. M. Kondoz, "Backward compatible insertion of user defined data into MPEG-4," *IEE Electronics Letters*, vol. 36, no. 12, Jun. 2000, pp. 1036-1037.
[10] F. Bota, F. Corno, and L. Farinetti, "Hypervideo: a Parameterized Hotspot Approach," *TICWI2002: IADIS International Conference WWW/Internet 2002*, Nov. 2002