AUTOMATIC TWO-LAYER VIDEO OBJECT PLANE GENERATION SCHEME AND ITS APPLICATION TO MPEG-4 VIDEO CODING

Kumi Jinzenji, Shigeki Okada, Hiroshi Watanabe, Naoki Kobayashi

NTT Cyber Space Labs.
1-1 Hikari-no-oka Yokosuka-Shi
Kanagawa, 239-0847 JAPAN
E-mail: {kumi, shige, hiroshi, kobayashi}@nttvdt.hil.ntt.co.jp

ABSTRACT

The new video coding standard MPEG-4 provides content-based functionality and low bit-rate video compression. We focused on the "sprite coding" supported MPEG-4 Version 1 Main profile in order to achieve "VHS quality video on 2B (128kbps)" for narrow-band transmission such as the Internet. Automatic VOP (Video Object Plane) generation technologies are being studied as one of the most important issues of MPEG-4 object coding. This paper proposes a two-layer VOP generation scheme with some core algorithms such as GME (Global Motion Estimation), foreground moving object extraction, and background sprite generation. This paper also describes a shape information reduction method for foreground objects. Using this method, shape information is compressed by 90%. Experiments are conducted on VOP generation and video coding with MPEG-4. Coding efficiency is 3-4 times higher than that of typical existing video coding schemes at the same subjective image quality.

1. INTRODUCTION

A new standard called MPEG-4 is currently being developed [1]. It will provide content-based functionality and very low bit-rate video compression. We are studying very low bit-rate video compression to achieve VHS quality on 2B (128kbps) for narrow-band use such as Internet applications. One solution can be found in the "Sprite Coding" supported in the MPEG-4 Version 1 Main profile. Sprite coding is extremely effective for video sequences with camera operation [2-4]. Some studies have introduced automatic sprite generation schemes, but they did not discuss foreground object extraction. For the background content-based video representation, object segmentation is the core technology for object-based video representation. Most existing technologies offer only manual/semi-automatic video segmentation [5][6]. Fully automatic video sequence decomposition to create VOP has not been achieved. Some studies offered automatic video object segmentation, but they place tight limits on the types of video sequences that are supported. Moreover, multiple object extraction and their correspondence over several frames remain unrealized [7-9].

This paper presents a new algorithm that can automatically generate VOPs and describes video coding experiments that use the MPEG-4 Version 1 Main profile. To prevent the multiple object correspondence problem, the two-layer VOP is proposed. The two-layer VOP consists of the foreground moving object and the background sprite. The "foreground object" covers all regions not identified as the background. Consider the example of a "soccer game", the foreground object covers players, the ball and any kind of moving object in the sequence. (Most existing technologies treat these small objects as independent foreground objects.) This paper also proposes original technologies, such as GME, background sprite generation, and foreground moving object extraction. The GME algorithm uses a four-parameter affine model suitable for camera operation. GM is calculated from the characteristics of the motion vector distribution in the feature space. Clusters in the feature space are examined to identify the camera operation, and the most prevalent camera operation is selected as GM. This is important in generating the sprite, because the sprite can become blurred and distorted if non-camera motion is selected as GM. To generate a clear sprite, the temporal median and overwriting methods are used. The foreground object is extracted by using the difference image between the original image and the image extracted from the sprite. It provides robust against GM error. This paper also describes a shape information reduction method for the foreground object. The moving objects automatically extracted do not obtain smooth boundaries and also contain many small regions that increase the volume of shape information. A video coding experiment using a two-layer VOP generated by proposed the algorithm is conducted. A comparison is made with a typical coding scheme from the viewpoints of subjective image quality and coding efficiency.

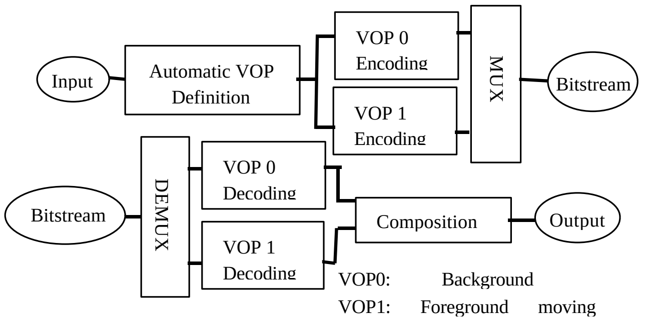

Fig.1 Concept of two-layer VOP: foreground moving object and background sprite.

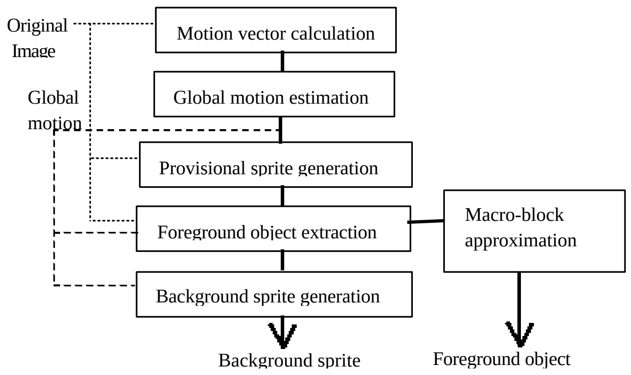
Original Image
Global motion
Motion vector calculation
Global motion estimation
Provisional sprite generation
Foreground object extraction
Background sprite generation
Macro-block approximation
Background sprite
Foreground object

Fig. 2 Flowchart of two-layer VOP generation.

Section 2 covers automatic two-layer VOP generation schemes including GME, sprite generation, and object extraction. Video coding experiments are described in Section 3. Section 4 concludes this paper.

2. AUTOMATIC TWO-LAYER VOP GENERATION

2.1 Two-layer Video Object

Figure 1 shows the concept of the two-layer VOP. This paper eliminates the issue of object correspondence among frames by treating all moving object regions as the foreground object.

Figure 2 shows a flowchart of the automatic two-layer VOP generation algorithm that consists of four parts: GME, background sprite generation, foreground object extraction and macro-block approximation of foreground shape.

2.2 GME for Sprite Generation

For sprite generation, GM should reflect only camera operation. This is the most significant difference from the typical GME algorithm which is tries to minimize the error between the original image and the predicted image with GM. To avoid non-camera movement (outlier), two techniques are proposed for GME: use of motion vector distribution in the feature space and cluster candidate selection. Figure 3 outlines the GME algorithm.

Camera motion can be described using the Hermart transform (four parameters affine) as:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} a' & b \\ -b & a' \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} c \\ d \end{pmatrix} \qquad (1)$$
$$a = a' + 1$$

where *(u,v)* is the motion vector calculated in each macro-block, *(x,y)* is the position of the pixel, and *{a,a',b,c,d}* is the set of GM parameters to be calculated. *a* and *a'* are scaling parameters, *b* denotes rotation, *c* and *d* denote translation.

First, the motion vector for each macro-block is calculated using the block-matching algorithm. Figure 4 shows the distribution of the motion vectors and their partial derivatives in the feature space. Motion vector distribution should strongly reflect camera operation.

Partial derivatives (see equation (2) and (3)) of the motion vectors are calculated for each macro-block.

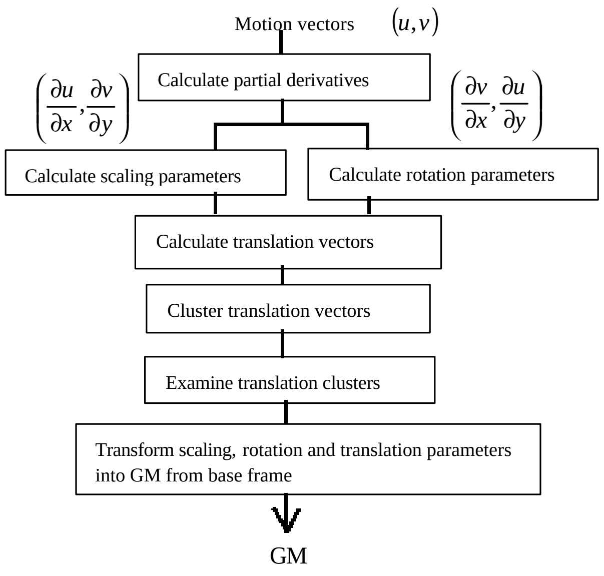
Motion vectors (u,v)
Calculate partial derivatives
$\left(\frac{\partial u}{\partial x},\frac{\partial v}{\partial y}\right)$
$\left(\frac{\partial v}{\partial x},\frac{\partial u}{\partial y}\right)$
Calculate scaling parameters
Calculate rotation parameters
Calculate translation vectors
Cluster translation vectors
Examine translation clusters
Transform scaling, rotation and translation parameters into GM from base frame
GM

Fig. 3 Overview of algorithm for GME.

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \equiv a' \qquad (2)$$

$$\frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x} \equiv b \qquad (3)$$

Each partial derivative creates a significant cluster on a line written by equations (2) and (3) in each feature space (see Figure 4-(a) and (b)). Here, all blocks with smooth intensity gradation are removed from the target blocks in the GME process, because such motion vectors are not accurate and often concentrate around zero. The centroid of each cluster yields scaling and rotation parameters. In this way, *a'* and *b* are detected.

Equation (1) can be transformed into

$$\begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} u \\ v \end{pmatrix} - \begin{pmatrix} a' & b \\ -b & a' \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}. \qquad (4)$$

Translation vectors in each macro-block are calculated using Equation (4). The translation vectors create several clusters in the feature space. These centroids of the clusters are the candidates of translation parameters *c* and *d*. Here, suppose that the translation vector $(c_{outlier}, d_{outlier})$ reflects outlier motion, while (c_{camera}, d_{camera}) is camera motion. The absolute difference at each pixel between the original and predicted images using a certain translation parameter candidate is calculated. The number of pixels whose value exceeds a certain threshold Th is calculated. If p_{ideal} is the pixel number for ideal camera motion, while $p_{outlier}$ is for outlier motion. p_{ideal} is always larger than any $p_{outlier}$. This principle can be used for the cluster candidate selection needed for translation parameter detection. First, an absolute difference image between the original image and the predicted image is calculated for each candidate to examine. Then, in each candidate, the number of pixels in the absolute difference image less than a certain threshold Th is counted. The centroid having the largest number of pixels is selected as the translation parameter. Here, threshold Th is experimentally determined. These detected

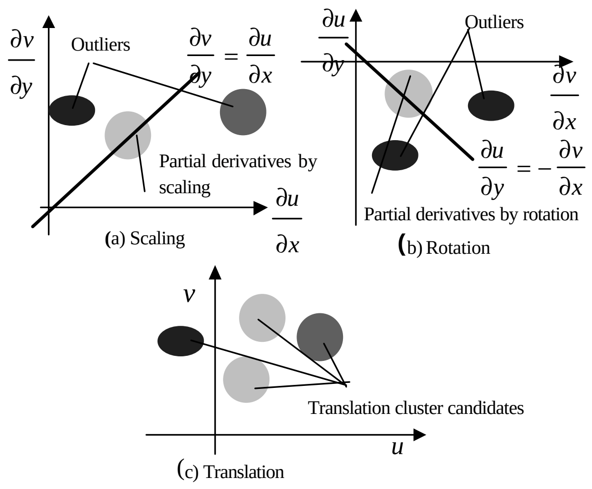

Fig. 4 Partial derivatives distribution of motion vectors.

parameters *{a,a',b,c,d}* are then transformed into GM from a preset base frame.

2.3 Background Sprite Generation

All frames are aligned to the basic coordinate of the base frame using the GM calculated by the proposed algorithm in the previous section. The pixel intensity of each coordinate is interpolated with the nearest four points in the original frame.

We note that there are several integration methods, such as temporal average, temporal median, and over-writing. From the viewpoint of subjective sprite quality, the over-writing method is the best. However, it cannot create a sprite for the entire background, because the moving objects can not be eliminated on the top frame. The median method is free of the moving object problem when the pixels of moving objects are fewer than half of the temporally aligned pixels. If that condition is fulfilled, the median method can create the entire background sprite. Accordingly, we use the temporal median and over-writing as integration methods.

To create a clear sprite, we use three steps: provisional sprite generation, foreground object extraction, and final sprite generation. First, aligned frames are integrated to form a provisional sprite using the temporal median method, so that moving objects are eliminated from the sprite. Second, background images are generated using differentiation of the provisional sprite and the original image as is explained in the next section. The background images have no moving objects. Aligned background images are integrated to form the final sprite by overwriting.

2.4 Foreground Moving Object Extraction

Figure 5 overviews foreground object extraction. First, a GM image is extracted from the provisional sprite with GM. Next, the absolute difference image is created from the GM image and original image. Next, the absolute difference image is divided into foreground image and background image using a certain threshold. The final sprite is generated from the background images, while foreground images are approximated into macro-blocks to reduce the volume of shape information.

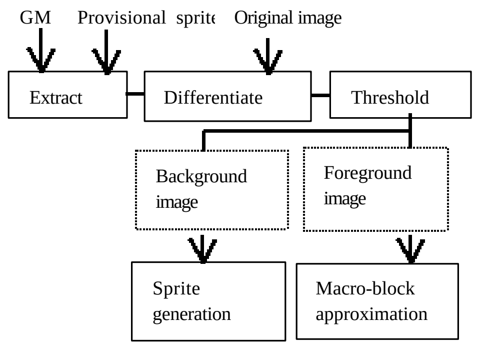

Fig. 5 Overview of foreground object extraction.

2.5 Shape Information Reduction

The initial foreground image has complex shape and small regions that occupy a large volume of shape information. Moreover, shape search area is limited to plus-minus 16 pixels so that most macro-blocks are coded in the INTRA mode. Two types of shape coding modes are provided in MPEG-4; lossy and lossless. The shape in the most lossy mode is the "macro-block" itself. When the foreground object pixels occupy more than half the macro-block, the entire macro-block is filled with "255", otherwise with "0". Here, "255" value is given to the foreground pixel, while "0" value is given to the background pixel. The most lossy shape compresses the data size by 80-90 % compared to the information created by the automatically extracted object's "original" lossless shape.

The most lossy shape approximation yields the highest level of the compression, but it cause shape erosion that degrades subjective image quality. Accordingly, we propose another macro-block-based approximation algorithm to reduce shape information without degrading image quality (see Figure 6). From several experiments, we found that the foreground area should be less than 10 % of the frame if the proposed sprite coding method is to be more efficient than existing coding schemes. Accordingly, the algorithm is controlled by setting the foreground area ratio. Two types of macro-block-based approximation are proposed. First, when a macro-block contains more than th1 pixels, the value "255" is given to all pixels in the macro-block. Second, if any of the surrounding eight macro-blocks are filled with 255 value pixels, and the target macro-block has more than the2 (th2<th1) pixels, all pixels in the target macro-block are given the value of 255. These two thresholds are gradually increased till the foreground rate is less than 10 %. Finally, the macro-block approximated foreground shape is coded using the lossless mode. Here, the thresholds Th1 and Th2 are experimentally determined.

3. MPEG-4 Visual Coding Experiment

We conducted several coding experiments using MPEG-4 Version1 Main Profile and video sequences with camera operation. Two types of coding mode were compared, sprite mode and rectangle mode, from the viewpoints of subjective image quality and coding efficiency. The rectangle mode is the basic of many existing coding schemes such as H.26X. Coding conditions were QP=12, SIF, 30 frames/sec.

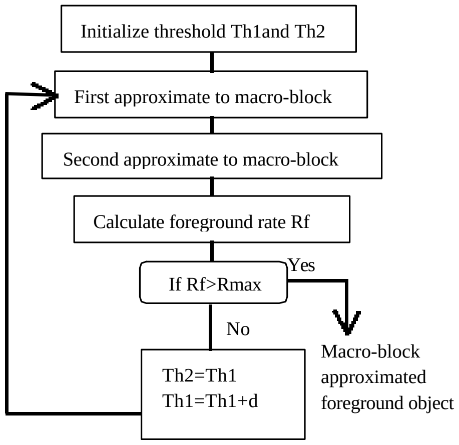
Initialize threshold Th1and Th2
First approximate to macro-block
Second approximate to macro-block
Calculate foreground rate Rf
If Rf>Rmax
Yes
No
Th2=Th1
Th1=Th1+d
Macro-block approximated foreground object

Fig. 6 Macro-block approximation for shape information.

Figure 7 shows coding results. Figure 8 shows examples of background sprite, foreground object, and a composed frame. Note that "bits per second" is written in the graph to compare sprite and rectangle modes. The coding efficiency of the sprite mode is 3-4 times higher than that of the rectangle mode. The sprite mode image has almost the same quality as the rectangle mode image at the same QP. The new method provides enough compression and image quality to allow "VHS on 2B".

We must note that these experiments assumed that camera motion had little zoom operation, so that a background image quality, which is extracted from the sprite, is supposed to "subjectively" equal to the original image. As an objective measurement, PSNR of the normal mode is apparently much better than sprite mode.

4. CONCLUSION

We focused on the "sprite coding" supported MPEG-4 Version 1 Main profile to achieve very low bit-rate video compression. This paper proposed a two-layer VOP generation scheme with core algorithms such as GME, foreground moving object extraction, and background sprite generation. This paper also described a shape information reduction method to handle moving objects. Using this method, shape information is compressed by 80-90 %. Experiments on VOP generation and video coding were conducted using MPEG-4. We found that the proposed two-layer VOP generation method yields a coding efficiency that is 3-4 times higher than those typical existing video coding schemes at the same subjective image quality. In the future, we will apply the proposed scheme to realize the "VHS quality video on 2B" service for narrow-band transmission infrastructures such as the Internet.

AKNOWLEGDEMENT

The authors wish to thank *Ms Noriko Yonehara* for her assistance during the computer simulations.

REFERENCES

[1] Final Draft of International Standard, ISO/IEC 14496-2.

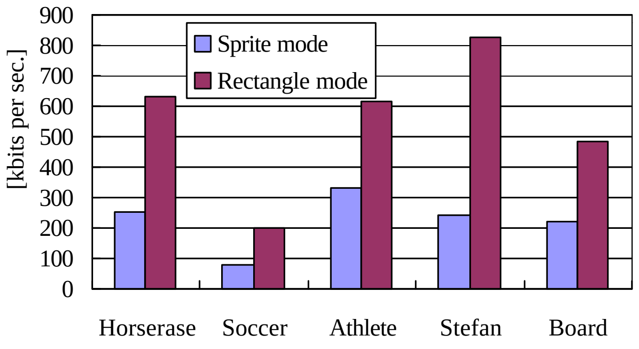

Fig. 7 Coding results with MPEG-4.

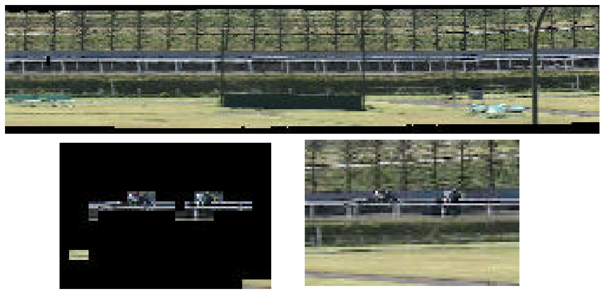

Fig.8 Coding image examples(Horserace). A part of sprite(upper), foreground object(lower left), and composed image(lower right).

[2] M. Irani, S. Hsu, and P. Anandan, "Video Compression Using Mosaic Representation," Signal Processing: Image Communication, Vol. 7, pp. 529-552, 1995.

[3] K. Jinzenji, H. Watanabe, N. Kobayashi, " Global Motion Estimation for Static Sprite Production and Its Application to Video Coding," IEEE ISPACS'98, pp.328-332, November 1998.

[4] K. Jinzenji, S. Takamura, H. Watanabe, N. Kobayashi, "Automatic VOP Production Scheme for Very Low Bit Rate Coding," PCS'99, pp.299-302, 1999.

[5] M. Kass, A. Witikin, D. Terzopoulos, "SNAKES: Active Contour Models," Proc. 1st ICCV, pp.259-268, 1987.

[6] J. G. Choi, S. Lee, S. Kim, "Spatio-Temporal Video Segmentation Using a Joint Similarity Measure," IEEE Trans. on CSVT, Vol. 7, No. 2, April 1997.

[7] A. Neri, S. Colonnese, G. Russo, "Automatic Moving Objects and Background Segmentation by Means of Higher Order Statistics," IEEE ISCAS'97, June 1997.

[8] R. Mech, M. Wollborn, "A Noise Robust Method for Segmentation of Moving Objects in Video Sequence," IEEE ICASSP'97, April 1997.

[9] T.Meier, K.N. Ngan, "Automatic Segmentation of Moving Objects for Video Object Plane Generation," IEEE Trans. on CSVT, Vol. 8, No. 5, September 1998.