Sprite Coding in Object-based Video Coding Standard: MPEG-4

Hiroshi Watanabe
Waseda University
29-7 Bldg., 1-3-10 Nishi-Waseda, Shinjuku-ku, Tokyo 169-0051 JAPAN
Tel: +81-3-5273-9104 Fax: +81-3-5286-3832
E-mail: hiroshi@giti.waseda.ac.jp

and

Kumi Jinzenji
NTT Cyberspace Labs.
1-1 Hikari-no-oka Yokosuka-shi, Kanagawa 239-0847 JAPAN

ABSTRACT

This paper outlines a new standard video compression technology: MPEG-4. We propose a very efficient coding scheme using the one of the compression tools of MPEG-4 "Sprite coding". We first introduce MPEG-4 visual "Profiles", which are sets of tools (technical elements) that realize certain types of applications. Among these "Profiles", "Main Profile" is capable of using "Sprites" i.e. a unified background image derived from a sequence having camera motion. We propose a new technique to split foreground moving objects from the background "Sprite" automatically for more efficient video compression than is possible with conventional technologies. We call the algorithm the "Two-layer video object plane (VOP) generation scheme". The two-layer VOP generation scheme has several core algorithms such as GME (Global Motion Estimation), foreground moving object extraction, and background sprite generation. The foreground object is MPEG-4 object coded in the main profile, while the background sprite is coded using MPEG-4 sprite coding. We call this coding scheme "sprite mode"; MPEG-4 simple profile coding is called "normal mode". Experiments are conducted on VOP generation and video coding with MPEG-4. We compare sprite mode to normal mode. The coding efficiency of sprite mode is several times higher than that of normal mode at the same objective image quality if the foreground ratio is within 10-15%.

Keywords: MPEG-4, Video Coding, Profile, Sprite, Video Object Extraction, Very low bit-rate coding

1. INTRODUCTION

MPEG-4 Visual[1] is an object-based video coding standard that employs state-of-the art video compression technology for representing 3D video objects in a scene. A frame based video sequence is decomposed to several video object planes (VOPs). These VOPs can be coded in different ways to suit their characteristics. For example, an object that stays in front of a background can be coded using an arbitrary shape. In addition, MPEG-4 adopts a new variable length code to achieve the error robustness desired for mobile and Internet applications. The standard is suitable for mobile videophones, which are a practical application in the next generation mobile communication system, IMT2000.

In this paper, we first introduce MPEG-4 visual "Profiles", which are sets of tools (technical elements) collected to achieve certain types of applications. Among these "Profiles", "Main Profile" is capable of using "Sprite" i.e. a unified background image derived from a sequence having a camera motion. Next, we propose an automatic two-layer video object extraction algorithm to achieve higher compression ratios than conventional video coding methods using MC+DCT. This algorithm is based on the two layer video object model. A video sequence is divided into the foreground and background object. The background object is represented as a sprite image. Unlike existing algorithms, the proposed algorithm offers object correspondence and entirely automatic processing [3-11]. The proposed algorithm generates foreground objects and the background sprite, and then compresses them by object coding and sprite coding of MPEG-4 Main profile, respectively. We compare the result to the result achieved with MPEG-4 simple profile without VOP structure.

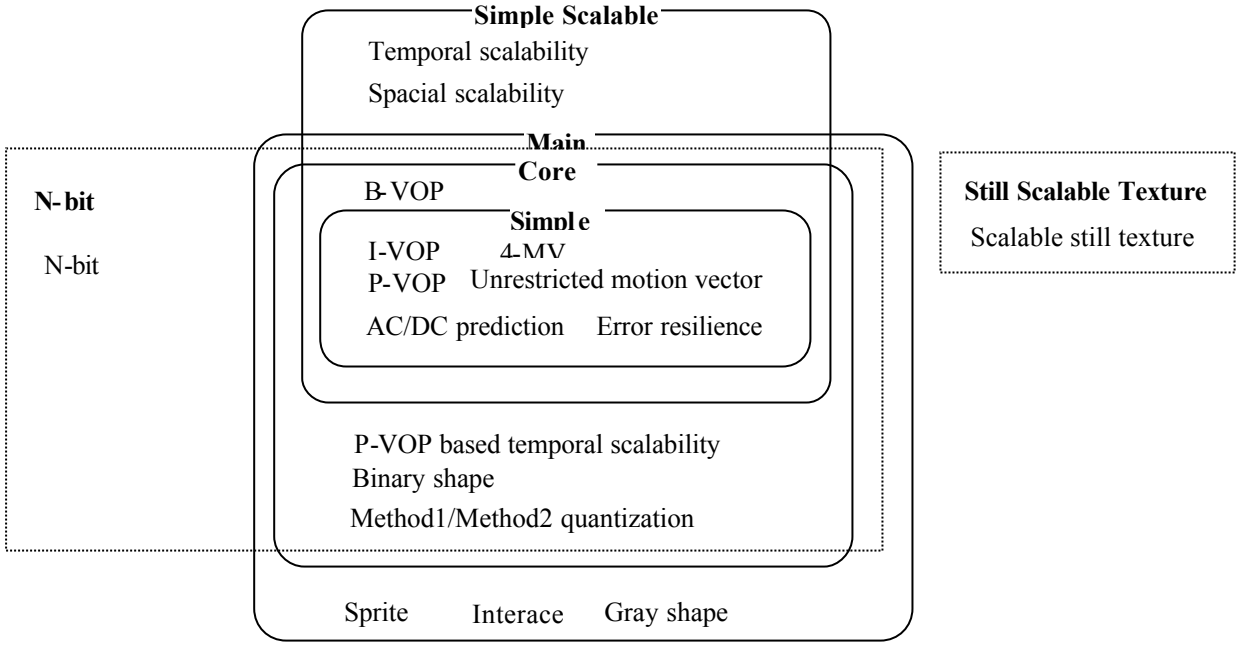

Fig1. MPEG-4 visual profile and visual tools.

2. MPEG-4 VISUAL PROFILE

MPEG-4 follows the concept of "Profile" in which the lower layer profile is a subset of a higher layer profile. This means that the higher layer decoder can decode all of the bitstream created by the lower layer encoder. This hierarchical model is also called "onion ring model." In MPEG-4, "Profile" contains "Object Type", a set of tools for several targeted applications. The list of "MPEG-4 Visual Profiles" and the related "Object Types" are as follows.

(1)Simple Profile: Simple Object Type

(2)Core Profile: Simple Object Type, Core Object Type

(3)Simple Scalable Profile: Simple Object Type, Simple Scalable Object Type

(4)N-bit Profile: Simple Object Type, Core Object Type, N-bit Object Type

(5)Main Profile: Simple Object Type, Core Object Type, Main Object Type,
Scalable Still Texture Object Type

(6)Scalable Still Texture Profile: Scalable Still Texture Object Type

"Simple Profile" is targeted for applications of mobile communication and the Internet. "Core Profile" is for PC applications. "Main Profile" is designed for higher resolution images. "Simple Scalable Profile" is suitable for applications that experience time-varying transmission channel environments. "N-bit Profile" is mainly used for satellite surveillance applications.

Many new coding techniques have been introduced to achieve new coding functions in MPEG-4. Intra, Predicted and Bi-directionally predicted Video Object Planes are the basic approaches to cope with arbitrary shaped images that differ from the conventional square ones. I-VOP and P-VOP can be used in "Simple Profile" where as B-VOP can be used in "Core Profile". Binary/Grayscale alpha plane can represent levels of transparency when two or more objects overlap. Binary alpha is used in "Core Profile" where as grayscale alpha can be used in "Main Profile". For enhanced error robustness, "Simple Profile" offers re-synchronization marker, reversible variable length code (VLC) and data partitioning. "Sprite" is one of the coding tools in "Main Profile". The MPEG-4 standard was design on the assumption that "Sprite" is provided in a certain way. How to generate "Sprite" lies outside the scope of MPEG-4. Wavelet still image coding can provide a graceful degradation for scalable image representation. In the next section, we will focus on MPEG-4 "Sprite" coding used in combination with automatic "Sprite" generation.

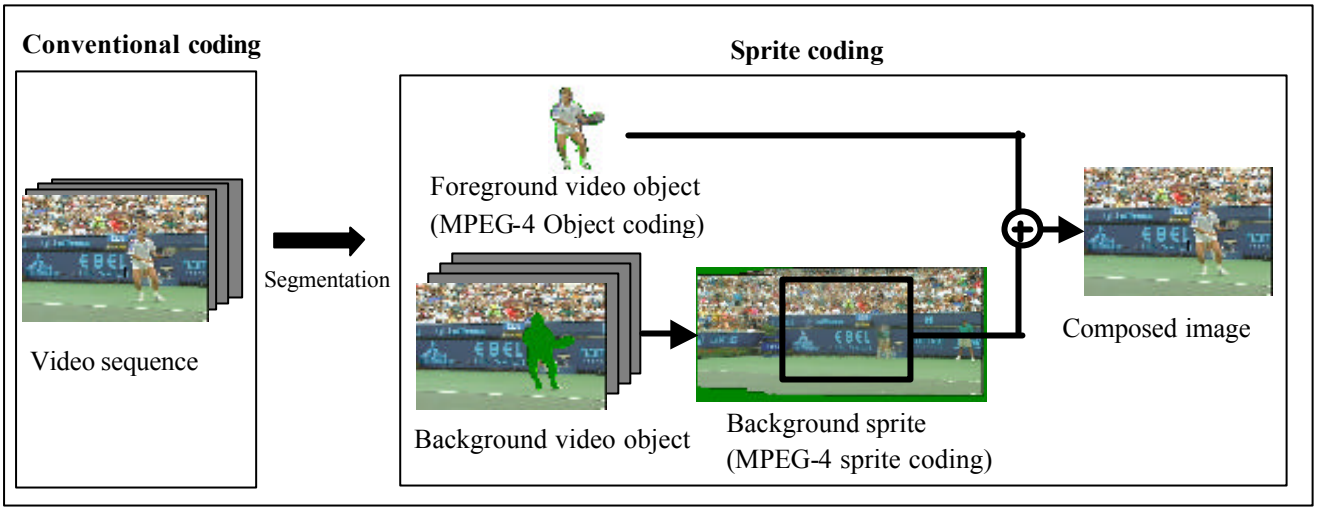

Fig2. Conventional coding vs. sprite coding.

3. SPRITE CODING

Figure2 shows the difference between conventional coding schemes and MPEG-4 object coding. The left side of the figure presents conventional coding. A video sequence is coded frame by frame using motion compensation to reduce frame redundancy. The right side of Figure 2 presents the typical MPEG-4 object coding process. A video sequence can be divided into several video objects. One of the objects is "Sprite". These video objects are assembled at the receiver site to recreate the image. Object extraction/segmentation lies outside the standardization issue.

Automatic "Sprite" generation from a video sequence is a key point in realizing the "Sprite" coding mode in MPEG-4. A "Sprite" can be viewed as a unified image when there is some camera motion such as panning or zooming. Most sport video scenes have such characteristics. First, we propose a new technique to separate moving foreground objects from panning and zooming video sequences. Next, foreground objects are coded by MPEG-4 arbitrary shaped video coding mode; "Sprite" is coded in still image coding mode. The background image in each frame can be reconstructed by geometric-projection from the "Sprite". This offers high coding efficiency, i.e. high compression ratio, since most of the background image in the video sequence can be represented by one large sprite image; temporal change is ignored.

3.1 Two-layer video object model

We presume a simple two-layer video object plane model. Each frame can be automatically separated to foreground and background VOPs. Fig.3 overviews

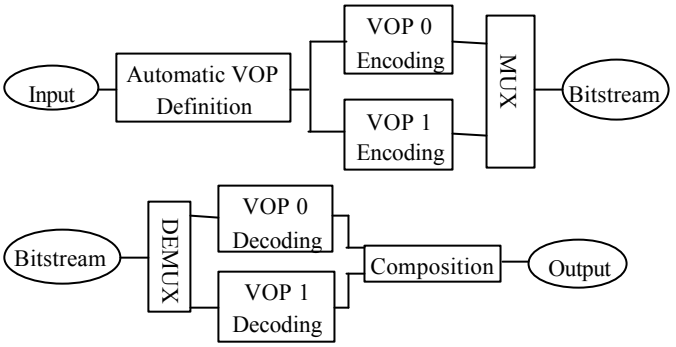

Fig3.Two-layer video object model.

the coding/decoding model proposed in this paper. On coding side, the image is first split into two layer video objects, namely the foreground object and the background sprite. The background sprite is the background and reflects the camera motion. The foreground object is any moving area not belonging to the background, and all such areas are treated as one object. For example in "soccer" image, the players, the referees and the ball are treated as one foreground object. The foreground object and the background sprite are independent video objects, and the former is converted to free shape code; the latter is subjected to sprite coding in the MPEG-4 Main profile. These isolated bit streams are multiplexed and sent as one. At the receiving side, the bitstream is demultiplexed and the video objects are decoded, superimposed and displayed. If the foreground area is small, the non-foreground area (the background area occupies

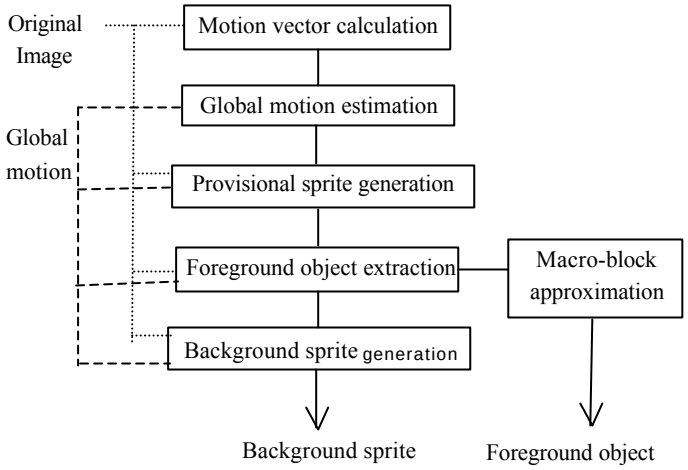

Fig.4 Automatic two-layer VOP generation algorithm.

most of the moving image) can be expressed as one sprite (= static image). Therefore, compression efficiency is expected to be higher than that possible with conventional methods based on MC+DCT.

Figure 4 overviews the flow of the two-layer VOP extraction algorithm. The algorithm consists of five parts: motion vector calculation, global motion estimation, provisional sprite generation, foreground object extraction and background(final) sprite generation.

3.2 Global motion detection

To generate the sprite, GM should faithfully reflect the camera motion. The typical GM calculation [4][5] yields the MSE (Mean Square Error) between the GM predicted image and the original image [4][5]. Because GM is calculated as the average value of local motion, it does not accurately reflect camera motion. Though documents [10][11] propose a GM calculation method, they do not exclude non-camera motion (outliers). To calculate GM that well reflects camera motion, this paper clarifies the relationship between camera motion and local motion vectors (parallel motion model), and proposes a way to select GM from local motion vector groups.

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

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

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

Each partial derivative creates a significant cluster on a line written by Eq.(2) and (3) in each feature space. 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

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

Subtracting the scaling and rotation effect from the block-based motion vector according to Eq.(4), yields the transition parameters. Block-based transition parameters are clustered to identify the median values, which well reflect the camera motion.

3.3 Background sprite generation

Once camera motion in the sequence is obtained, each frame image can be superimposed considering its size and location. Next, median filtering in the temporal direction is performed to remove the foreground object. Images without foreground object are super- imposed to generate one "Sprite" for the sequence. Applying scaling, rotation and transition parameters to the "Sprite" yields the background image of each frame. To generate a high quality sprite, this paper takes advantage of a conventional temporal median method[10], the overwriting method. Two kinds of sprite are generated: provisional sprite and final sprite.

Using the GM calculated from the base frame in the above subsection, we transform the shape of each image and map them on the lattice points of the base coordinate system. We can see that multiple pixels overlap each other. We calculate their median value and take it as the value of that coordinate. This generates the sprite without moving area. However, the sprite generated by the temporal median method tends to be a little fuzzy. We, therefore, use this median-value-based sprite as a temporary sprite to extract the foreground. The overwriting method pastes the pixels as they are on the base frame and so generates a high quality sprite. However, it has a

Background sprite

Original image

Foreground object

Composed image

Fig.5 Generated sprites and foreground objects.

problem in that the foreground of the top frame and the foreground object at the edge of each frame remain. Accordingly, after extracting the moving object, we treat the remaining part of the image as the background image and paste the pixels on the base coordinates one by one to generate the final high quality sprite.

3.4 Foreground object extraction

Finally, this background image is subtracted from the input image to obtain the foreground object.

At first, we calculate the difference image between the original image and the image extracted using the temporary sprite. We binarize this difference image by thresholding and split it into a foreground candidate image and a background image. The foreground candidate image is processed by the foreground shape macro block approximation method (described in the next section), to yield the final foreground image.

Because the automatically generated foreground object has complicated contours (many acnodes), its shape is very difficult to predict accurately. This triggers intra coding which increases coding volume. To sidestep this problem, this paper proposes a method that approximates object shape by using macro blocks. Foreground contours are approximated by macro-blocks.

This paper proposes two phase macro block approximation. At first, the macro block (contains more than threshold Th1 foreground pixels) is regarded as foreground. All other macro blocks are regarded as background. We then focus on the background macro block adjacent to the foreground macro block from the first macro-block approximation phase. If more than threshold Th2 (Th2<Th1) of foreground pixels are included in that macro block, we regard it as foreground.

Another merit of macro block approximation is the lower processing cost in software decoder implementation. As shape information encoding and padding are not needed, this method is suitable for applications that require real time encoding.

3.5 Video Coding simulation

In the previous section, we applied the two-layer VOP generation algorithm to moving images and generated video objects. We converted the foreground objects of several images to MPEG-4 object code, and converted the background sprites to same profile sprite code. We then processed the original moving image using the MPEG-4 Simple profile code (One VOP). For convenience, the former (latter) is referred to as sprite (normal) mode. The code volume with sprite mode is the sum of the foreground object code and the background sprite code. We used five video sequence including “stefan”: standard video sequence for simulation trials. These video sequences contain camera motion and some moving objects. Quantization parameter (QP) was constant at 15 throughout the sequence. Foreground ratio was set to 10 – 15 percent of the image size.

Figure 5 shows the examples of the sprite and foreground moving object so generated. Figure 6 presents the coding results. When the foreground ratio is within 10 - 15%, the coding efficiency of sprite mode is more than twice that of normal mode. This is noticeable in “Stefan”, which includes a large audience area. The other images have more even backgrounds.

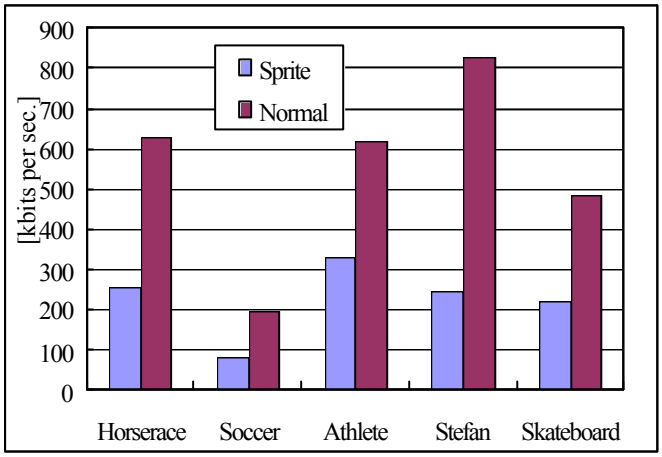

Fig.6 Coding simulation result.

Even if a part of a moving area is not extracted as foreground, its loss does not impact image quality.

4. CONCLUSION

We overviewed the new video coding standard MPEG-4 visual profile and object types. We then proposed a new video coding algorithm that well utilizes the characteristics of MPEG-4 visual tools such as sprite and video objects. The automatic two-layer VOP generation algorithm was proposed and applied to low bit rate video coding. In simulations, the proposed algorithm was found to provide a dramatic increase in compression rates, 50% to 25%, compared with the normal MPEG-4 mode. It offers the same quality video when the foreground object size is around 10 to 15% of the image.

However, video sequences do not always satisfy these conditions. How to judge which shot should undergo sprite coding, a seamless decoding method, and how to allocate the code to each video object are future tasks.

REFERENCES

[1] ISO/IEC 14496-2, "Information Technology – Coding of audio-visual objects– Part 2: Visual' (1999).

[2] "MPEG-4 Video Verification Model Version 15.0,"ISO/IECJTC1/SC29/WG11 PEG98/N3093.

[3] K. Jinzenji, S. Okada, H. Watanabe, N. Kobayashi, "Automatic Two-layer Video Object Plane Generation Scheme And Its Application to MPEG-4 Video Coding," IEEE ISCAS2000, pp.606-609, May 2000.

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

[5] J. Wang and E. Adelsen, "Representing Moving Images with Layers," IEEE Trans. on IP, Vol. 3, No. 5, pp.v625-638, September 1994.

[6] M. Lee, W. Chen, C. Lin, C. Gu, T. Markoc, S. Zabinsky, R. Szeliski, "A Layered Video Object Coding System Using Sprite and Affine Motion Model," IEEE Trans. on CSVT, Vol. 7, No. 1, February 1997.

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

[8] 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.

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

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

[11] 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.

[12] H. Jozawa, K. Kamikura, A. Sagata, H. Kotera, and H. Watanabe: "Two-stage motion compensation using adaptive global MC and local affine MC," IEEE Trans. Circuit & Systems for Video Tech., Vol. 7, No. 1, pp. 75-85, January 1997.

[13] J. Bergen, P. Anandan, K. Hana, R. Hingorani, "Hierachical Model-Based Motion Estimation," ECCV '92, pp. 237-252, May 1992.

[14] A. Akutsu, T. Tonomura, H. Hamada, "Videostyler: Multi-dimensional Video Computing for Eloquent Media Interface," ICIP ' 95, pp. 330-333, September 1995.

[15] K. Jinzenji, S. Ishibashi, and H. Kotera, "Algorithm for Automatically Producing Layered Sprites by detecting Camera Movement," IEEE International Conference on Image Processing '97 (ICIP'97), pp. 767-770, October 1997.

[16] 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.