UNSUPERVISED OBJECT-BASED SPRITE CODING SYSTEM FOR TENNIS SPORT

Ching-Yeh Chen, Shao-Yi Chien, Yi-Hau Chen, Yu-Wen Huang, and Liang-Gee Chen

DSP/IC Design Lab
Graduate Institute of Electronics Engineering and Department of Electrical Engineering
National Taiwan University
1, Sec. 4, Roosevelt Rd., Taipei, Taiwan

ABSTRACT

Sprite coding is a new objected-based coding technology proposed by MPEG-4 video standard. In this paper, we propose an unsupervised sprite coding system for sport videos, for example, tennis sequences. Our system can provide several important functions. First, the sprite of the background can be generated without any pre-processing masks in our system. Second, it can automatically segment the foreground and background in a video sequence. Third, it can provide the masks of the foreground objects and the tennis ball. The experimental results show that our system has a very good performance and the coding gain of our system compared with MPEG-4 Advanced Simple Profile is 2.5dB at the low bit rate (270Kbps) and is 2dB at the ultra low bit rate (70Kbps). It can be used in the sport video coding at low bit rate and provides a object-based sport video sequence.

1. INTRODUCTION

The concept of the sprite is proposed by MPEG-4 video standard [1]. A sprite(mosaic) is an image that collects all related information belonging to a video object in a video sequence. For a still object, like background, a sprite can give a panoramic view of the scene and can well represent the associated video object planes (VOPs). According to this property, sprite coding is proposed and it has several advantages. First, sprite coding is an efficient video coding with parametric motion models at very low bit rate. Second, sprite coding is also a kind of object-based coding. It is to say that sprite coding can provide various functions and options to users.

There are also many applications of sprite. For example, we can use the sprite of the background to segment the foreground objects in a video sequence. The sprite of the background can support the MPEG-7 descriptors about the background in a video sequence. The camera motion provided by sprite coding is also an important descriptor in MPEG-7. Moreover, there are much more intelligent information provided by sprite coding. There are several sprite coding systems in previous researches[2] [3]. However, their performances are not good and cannot provide a object-based sport video sequence. At the same time, there are some difficulties when we implement a sprite coding system. For example, if we want to collect the information of the background, we have to recognize the information of the foreground first. However, if we want to recognize the information of the foreground, the related information of the background is needed. That is, it is an egg-and-chicken problem. Another task is the algorithm to generate the sprite of the background, where global motion estimation and compensation are involved. There is a lack of a appropriate algorithm of global motion estimation and compensation to be applied in a sprite coding system.

In this paper, we propose a complete sprite coding system for sport videos, for example, tennis sequences. And we overcome the above-mentioned difficulties by means of a two pass method. A defective sprite is generated in the first pass. In the second pass, the defect is compensated and then we use the compensated sprite to segment the foreground objects. Our proposed system can automatically generate the sprite of the background without any pre-processing masks. The masks of the foreground objects and the tennis ball can be generated automatically in our system. Our system can provide high quality in PSNR of the foreground and in subjective view of the background at very low bit rate. The structure of this paper is as follows. In Sec. 2, we would introduce the algorithm in our sprite coding system. In Sec. 3, we would show the experimental results and compare it with MPEG-4 Advanced Simple Profile. And a conclusion is given in Sec. 4.

2. ALGORITHM

Our target is to set up a sprite coding system which can achieve following requirements. First of all, the sprite of the background can be generated automatically without any pre-processing masks in our system. Second, it can use the related information in a sprite coding system to generate the masks of the foreground objects automatically. Third, since we focus on the application of sport video sequences, tennis sport, this system also has to get the information of the tennis ball. Therefore, the flowchart of the proposed sprite coding system is shown in Fig. 1. There are two passes in our system. The sprite of the background is generated in the first pass, which is represented with the thick line. The dotted line shows the second pass to compensate the defect in the sprite of the background and segment the masks of the foreground objects. The processing steps are as following.

1. In *Sprite Generator*, the sprite of the background in a video sequence is generated.
2. Use the information of the sprite to reconstruct the background in a frame in *Background Reconstruction*.
3. Calculate the difference between the original and reconstructed frame in *Difference Comparison*.
4. According to the information of the difference, the defect in the sprite of the background is recovered and a compensated sprite is gotten in *Update Sprite*.
5. Repeat the step 2 but use the compensated sprite.

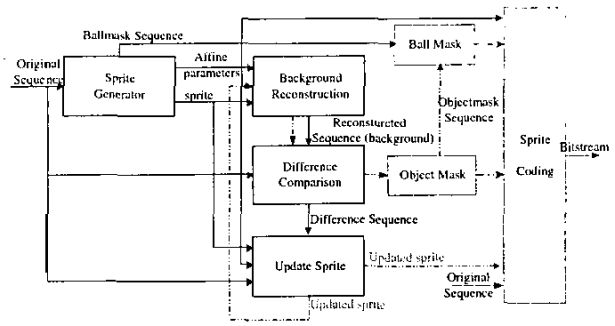

Fig. 1. Block diagram of the proposed sprite coding system.

6. Repeat the step 3.
7. The masks of the foreground objects are segmented in accordance with the information of the difference in *Object Mask*.
8. The mask of the tennis ball is calculated in *Ball Mask*.
9. All related information are compressed into a bitstream in *Sprite coding*.

In the successive subsections, we would introduce the five major components, *Sprite Generator*, *Update Sprite*, *Object Mask*, *Ball Mask* and *Sprite coding* in our proposed system.

2.1. Sprite Generator

The core operation of *Sprite Generator* is a feature-point based global motion estimation[4]. There are three steps in this algorithm. First, we select some feature points which have larger Hessian values,

$$HessianValue = (\frac{\partial I(x,y)}{\partial x})(\frac{\partial I(x,y)}{\partial y}) - (\frac{\partial^2 I(x,y)}{\partial x \partial y})^2 \quad (1)$$

to represent the frame. The second step is motion estimation, and affine model,

$$x' = a_0 x + a_1 y + a_2 \quad (2)$$
$$y' = a_3 x + a_4 y + a_5, \quad (3)$$

is used to describe the global motion. We use the previous global motion parameters as prediction to calculate the motion vectors of these feature points in the sprite. Simultaneously, in order to avoid the selected feature points belonging to the foreground objects and get more accurate global motion parameters, the motion vector of the selected feature point is neglected if the absolute difference of this feature point is larger than a threshold. Finally, we use these motion vectors to estimate the global motion parameters by the LMS algorithm. This process is repeated iteratively until it converges or the number of iterations is larger than ten. When this process stops, the final global motion parameters are gotten. According to these global motion parameters, we warp the current frame to a warped frame in the sprite domain and only record the new information in the warped frame.

2.2. Update Sprite

Because we cannot recognize the information of the foreground objects before the sprite of the background is generated, there is a defect appeared in the sprite of the background. Therefore, we have to compensate this defect to get the true background. After *Difference Comparison*, there are two major categories in the difference frame. The foreground objects in the current frame belong to a category, and the region warped by the defect is the other. That is to say, if we warp the difference frame to the warped frame in the sprite domain, the former is warped to different positions and the latter is warped to the same place, the defect in the sprite. By this property, we make a judgement, $J(x,y)$, to detect the region of the defect in the sprite of the background.

$$J(x,y) = \begin{cases} Defect & \text{if } P(x,y) \geq \frac{2}{3} max(P(x,y)) \\ Background & \text{otherwise} \end{cases} \quad (4)$$

where (x,y) is the position of a pixel in the sprite domain, and $P(x,y)$ is the number of times, where the corresponding position in the difference frame of the pixel (x,y) is a pixel of some object. After we find the region of the defect, we select the temporal median value of each pixel in the region of the defect to make up for the defect in the sprite of the background. In order to reduce the memory size, a hierarchial method is used. A median is selected in every ten frames, and then we choose the median in the set of these medians. In this way, only ten percent memories are required compared with the original method.

2.3. Object Mask

After the process of *Update Sprite*, there is only one category in the difference frame. This category includes the foreground objects in the current frame. However, there are still two objects in this category corresponding to the two players in the game. The job of *Object Mask* is to separate the foreground objects into the two different objects. The first step is temporal continuity. In *Difference Comparison*, some information is lost after eliminating noise. We can recover these information by temporal continuity. Next, a connected component algorithm is applied to separate the masks of the foreground objects into two mask frames, and then we use the temporal similarity to classify them. At last, the rectangular mask are used to replace the irregular shape mask to achieve a higher compression ratio.

2.4. Ball Mask

In *Sprite Generator*, some pixels are selected as the candidates for the tennis ball. These pixels have very larger Hessian values and the chrominance of these pixels are in a certain range. The tennis ball does not stay at the same place in continuous several frames; therefore, we can delete these candidates which stay at the same position in successive frames. On the other hand, the information of the tennis ball does not appear in the sprite of the background so the corresponding position of the tennis ball in the sprite domain does not have a very large Hessian value, but the pixel of the tennis ball in the current frame does. According to this property, there are some candidates which can be deleted. Finally, there are two candidates for the tennis ball in each frame, in average.

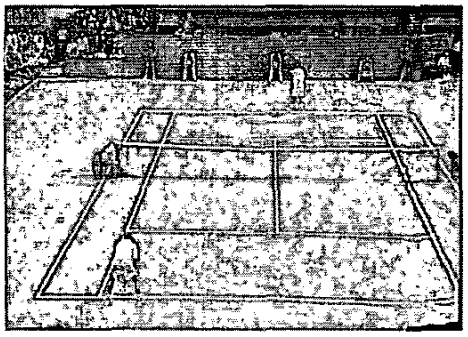

Fig. 2. The sprite of Sampras.

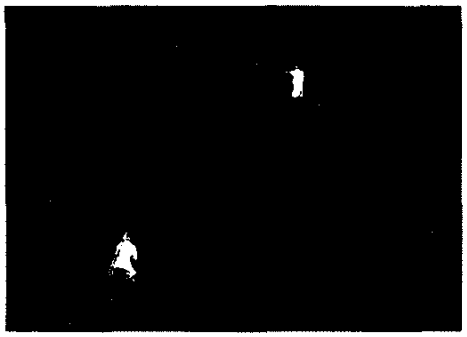

Fig. 3. The defect in the sprite of Sampras.

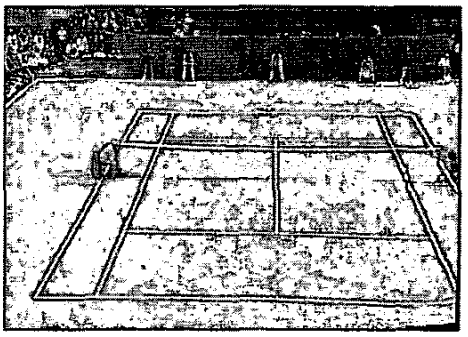

Fig. 4. The compensated sprite of Sampras.

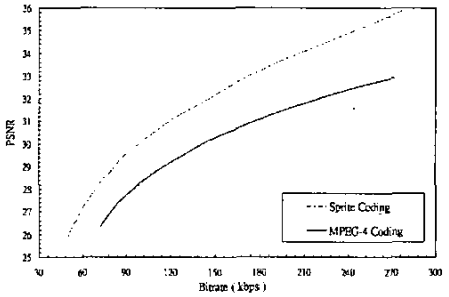

Fig. 5. The comparison between the RD curve of the foreground objects in our system and that of the whole frame in MPEG-4 Advanced Simple Profile.

2.5. Sprite coding

Sprite coding is a component to compress all related information in a video sequence into a bitstream. In *Sprite coding*, MPEG-4 object coding tools are applied to compress the information of the foreground objects and the sprite of the background. The information of the tennis ball is compressed by the DPCM technique. Furthermore, the sprite points in MPEG-4 video standard are used to replace the global motion parameters in order to preserve the precisions of the parameters in the coding system.

3. EXPERIMENTAL RESULTS

This section would show the experimental results of our proposed sprite coding system. The test platform is a PC with a P4 1.8 GHz processor. The test sequence is a tennis sequence, Sampras, in CIF format with 162 frames. In the following subsections, we also show the experimental results of *Sprite Generator*, *Update Sprite* and the comparison of rate distortion curves and the reconstructed frames between our system and MPEG-4 Advanced Simple Profile with searching range 16.

3.1. Sprite Generator and Update Sprite

The experimental result of *Sprite Generator* is shown in Fig. 2. In the sprite of the background, there are apparent two foreground objects corresponding to the two players in the game. The reason is that we cannot recognize the information of the foreground objects before the sprite of the background is generated. In Fig. 3, we show the position of the defect detected in *Update Sprite* and Fig. 4 is the compensated sprite after the process of *Update Sprite*. There is no foreground objects in the compensated sprite of the background. That is to say, we overcome the egg-and-chicken problem about the segmentation of the foreground and background, and successfully generate a sprite without any pre-processing masks. Besides, our sprite has a good quality in subjective view.

3.2. Rate Distortion

In Fig. 5, we show the comparison of rate distortion between our proposed sprite coding system and MPEG-4 Advanced Simple Profile with searching range 16. There are three major components of the bitstream in our system. They are the information of the foreground objects, the global motion parameters and the information of the tennis ball. The bit rate of the sprite of the background can be neglected, because in a real video sequence, there are thousands of frames in a scene. That is to say, the information of the sprite would be reused in very many frames. The bit rate of the sprite of the background is distributed to each frame averagely so that the bit rate of the sprite of the background is small enough to neglect it. Similarly, the PSNR is calculated by only considering the part of the foreground objects excluding the information of the background.

According to Fig. 5 at low bit rate (270Kbps, 30fps), the coding gain of our proposed system compared with MPEG-4 Advanced Simple Profile is 2.5dB. At the ultra low bit rate (70Kbps,

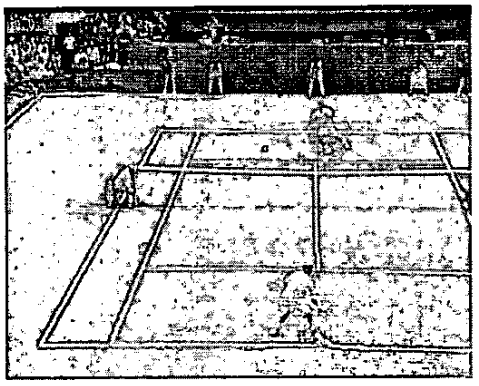

Fig. 6. The reconstructed frame 51 in Sampras by our proposed system at 160Kbps and the PSNR is 32.61 dB.

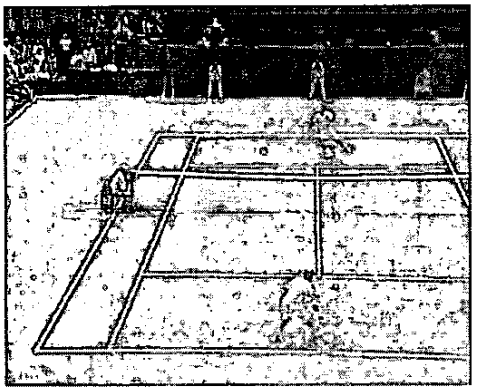

Fig. 7. The reconstructed frame 51 in Sampras by MPEG-4 Advanced Simple Profile at 160Kbps and the PSNR is 30.62 dB.

30fps), the coding gain is 2dB. In Fig. 6 and Fig. 7, we show the reconstructed frame 51 in Sampras by our proposed system and MPEG-4 Advanced Simple Profile at 160Kbps, respectively. The PSNR of our system is 32.61dB and that of MPEG-4 Advanced Simple Profile is 30.62dB at 160 Kbps. In Fig. 8 and Fig. 9, we show the reconstructed frame 71 in Sampras by our proposed system and MPEG-4 Advanced Simple Profile at 72Kbps. The PSNR of our system is 28.28dB and that of MPEG-4 Advanced Simple Profile is 26.33dB. To be brief, our performance is much better than MPEG-4 Advanced Simple Profile either in the PSNR or in the subjective view and the coding gain is at least 2dB .

4. CONCLUSION

In this paper, we establish a complete and unsupervised sprite coding system for sport videos, for example, tennis sequences. Our proposed system can automatically generate the sprite of the background without any pre-processing masks by a two pass method. The foreground and background in a video sequence can be segmented automatically in our proposed system. The masks of the foreground objects and the tennis ball are also provided by this system. The performance of our proposed system is much better than MPEG-4 Advanced Simple Profile and the coding gain compared

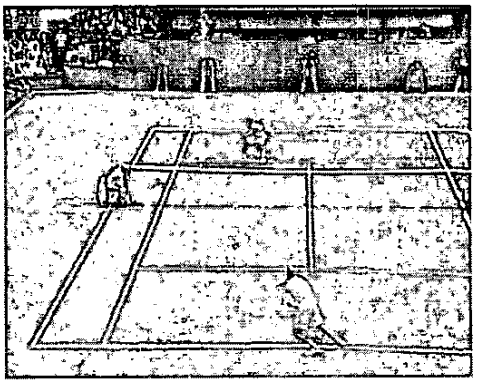

Fig. 8. The reconstructed frame 71 in Sampras by our proposed system at 72Kbps and the PSNR is 28.28 dB.

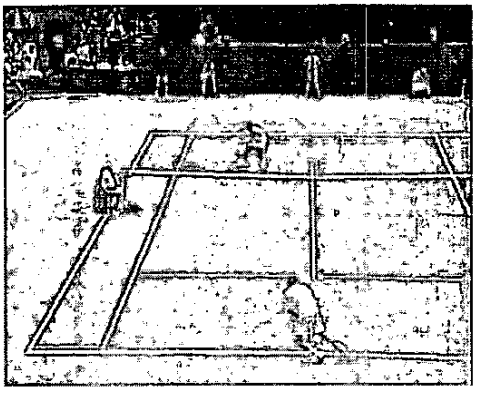

Fig. 9. The reconstructed frame 71 in Sampras by MPEG-4 Advanced Simple Profile at 72Kbps and the PSNR is 26.33 dB.

with MPEG-4 Advanced Simple Profile is at least 2dB at low bit rate.

5. REFERENCES

[1] MPEG Video Group, "The MPEG-4 video standard verification model version 18.0," *ISO/IEC JTC 1/SC 29/WG11 N3908*, 2001.

[2] K. Jinzenji, S. Okada, H. Watanabe, and N. Kobayashi, "Automatic two-layer video object plane generation scheme and its application to MPEG-4 video coding," in *Proc. of International Symposium on Circuits and Systems*, 2000, pp. 606–609.

[3] K. Jinzenji, S. Okada, N. Kobayashi, and H. Watanabe, "MPEG-4 very low bit-rate video compression by adaptively utilizing sprite to short sequences," in *Proc. of International Conference on Multimedia and Expo*, 2002, pp. 653–656.

[4] S.-Y. Chien, C.-Y. Chen, W.-M. Chao, C.-W. Hsu, Y.-W. Huang, and L.-G. Chen, "A fast and high subjective quality sprite generation algorithm with frame skipping and multiple sprites techniques," in *Proc. of International Conference on Image Processing*, 2002, pp. 193–196.