# MULTIPLE SPRITES AND FRAME SKIPPING TECHNIQUES FOR SPRITE GENERATION WITH HIGH SUBJUCTIVE QUALITY AND FAST SPEED

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

DSP/IC Design Lab, Graduate Institute of Electronics Engineering,
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
{shoayi, bluecity, yuwen, lgchen}@video.ee.ntu.edu.tw

## ABSTRACT

Sprite is an image collecting information of a video object through a video sequence. It can be used for efficient video coding, video summary, browsing, and editing. In this paper, three new techniques for sprite generation are proposed. Boundary matching and multiple sprites techniques can improve the subjective quality with refining the positions of the warped frames and generating more than one sprites. Frame skipping technique can skip redundant frames that contain only little new information when the camera revisits a scene several times to accelerate the sprite generation process. Experimental results show that these techniques can be employed independently and can improve the subjective quality as well as reduce 47.68% - 17.22% runtime of sprite generation. They can be applied with any sprite generation algorithms.

## 1. INTRODUCTION

Sprite (mosaic) is an image constructed from information of a video object in all frames of 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) in all frames. With parametric motion models, it can be exploited as an efficient video coding tool [1], which is also a part of MPEG-4 video standard [2]. Sprite coding has been proved to have good coding efficiency, which can give good subjective quality with very low bit rate [1][3], and the sprite can be also used for video summary and video database applications.

The key operation of sprite generation is global motion estimation. In the verification model of MPEG-4 and many other algorithms, global motion estimation is achieved with differential approach, where the differential of error with respect to the motion parameters need to be calculated at all pixels of every frame [4][5]. Several iterations are also required, which makes the computation load enormous. To improve the subjective quality of sprite, a frame matching algorithm is further introduced in [6], whose computational intensity becomes too high to be applied in real applications. On the other hand, a feature points hierarchical global motion estimation algorithm is proposed in [7]. The required computation is much lower; however, for real applications, the fast algorithm of sprite generation is still needed. Moreover, the subjective quality of sprite is not satisfied. The limitation of motion models with six, eight, and twelve parameters leads to many artifacts.

In this paper, several novel techniques for improving the subjective quality of sprite and accelerating the generation process are proposed. The subjective quality can be improved with boundary matching and multiple sprites technique (Multi-Sprite), where multiple sprites can be created to overcome the limitation of motion models. In addition, the sprite generation process can be accelerated with frame skipping technique (FS-Sprite). It can skip those frames captured when the camera revisit a scene that has been registered in the sprite before. These techniques can be applied with any sprite generation algorithm.

This paper is organized as follows. First, the proposed techniques, including boundary matching, Multi-Sprite, and FS-Sprite, are described in Sec. 2. Next, the experimental results are shown in Sec. 3. Finally, Sec. 4 gives the conclusion.

## 2. PROPOSED ALGORITHM

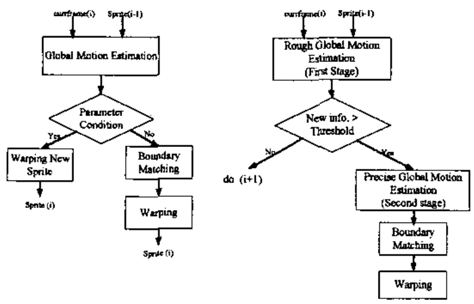
In this section, the techniques to improve the subjective quality are first proposed, which contains boundary matching and Multi-Sprite. Next, FS-Sprite is described, which can accelerate the sprite generation process. These techniques can be applied simultaneously to have both advantages at the same time.

### 2.1. Boundary matching

Inaccuracy of global motion estimation often leads discontinuity artifacts in the sprite, which are unacceptable for human vision. However, this effect occurs in every sprite generation algorithm since there is no perfect global motion estimation. In order to eliminate these artifacts, a method named boundary matching is proposed. After being warped, the new aligned pieces to be integrated into the sprite can be found. Before these pieces are added into the sprite, the positions of them are further refined with boundary matching algorithm: they are shifted vertically in the sprite, and the refined positions are those having the minimum boundary matching errors. The motion parameters are also refined.

* This work is partly supported by SiS Education Foundation.

0-7803-7304-9/02/$17.00 C2002 IEEE 




(a) (b)

Fig. 1. Flowchart of (a) Multi-Sprite and (b) FS-Sprite.

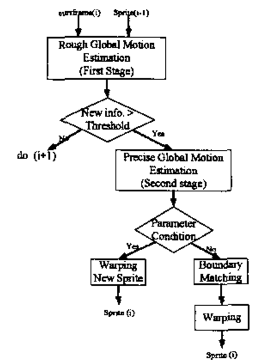



Fig. 2. Flowchart of Multi-Sprite and FS-Sprite hybrid scheme.

### 2.2. Multi-Sprite

In conventional sprite generation algorithms, only one sprite is generated. All frames in a video sequence are aligned into the sprite with respect to a fix coordinate system of a frame, which is often the first frame. For large camera motion, the subjective quality is degraded. For example, if the camera is zooming, and the current frame is warped with respect to the coordinate system of the first frame, although the resolution of the current frame is higher, it is downsampled to have the same resolution as the first frame, where some information of the current frame is lost. Note that, the resolution here is the pixel number used to represent a physical object in real world. Furthermore, for complex camera motion, such as sideway rotation, the depth of the scene is large, or the distance between the scene and the camera is changed, simple motion model with six, eight, or twelve parameters cannot be successful employed. In these cases, multiple sprites can be generated rather than only one sprite, which is names as Multi-Sprite. It can improve the quality of reconstructed frames without extra computation complexity overhead.

The flowchart of Multi-Sprite is shown in Fig. 1(a). After global motion estimation, the motion parameters are checked. If the camera motion is too large with respected to the current coordinate system to be handled with the motion model, a new sprite is created with respect to the coordinate system of the current frame; otherwise, the generation process remains the same. Note that the boundary matching technique is also integrated in Fig. 1(a), and any global motion estimation algorithm can be used with the proposed technique.

Affine model is taken as an example. It can be shown as the following equation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_2 & a_3 \\ b_2 & b_3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_1 \\ b_1 \end{bmatrix},$$

where *(x, y)* and *(x', y')* denote the coordinates of a point in the current frame and in the sprite, respectively. The condition of large camera motion can be set as:

$$a_2 \cdot b_3 < Th\_s \quad \text{or}$$
$$a_3 > Th\_r \quad \text{or}$$
$$b_2 > Th\_r,$$

where $Th\_s$ is the threshold of scaling, and $Th\_r$ is the threshold of rotation.

### 2.3. FS-Sprite

When the camera motion is small, or the camera is stopped to move, the camera revisits the same scene several times, and only a little new information in the current frame needs to be added into the sprite. The sprite generation procedure can be accelerated with skipping this kind of frames, and the quality of sprite can be still maintained. This technique is names as FS-Sprite, and the flowchart of FS-Sprite is illustrated in Fig. 1(b). The global motion estimation is divided into two stages: rough global motion estimation and precise global motion estimation. At the first stage, the rough motion parameters are estimated, which requires little computation. According to the rough motion parameters, the current frame can be skipped if

$$\frac{\#\{W(currframe(i)) - [W(currframe(i)) \cap sprite(i-1)]\}}{\#\{W(currframe(i))\}} < Percentage$$

where *currframe(i)* is the current frame at time *i*, *sprite(i-1)* is the sprite at time *(i-1)*, *W(currframe(i))* is the warped current frame, *Percentage* is a threshold of new information. If a lot of new information exists in the current frame, we continue to do the second stage, or this frame is skipped. For the skipped frame, the motion parameters are the rough parameters, and large portion of computation can be avoided. On the other hand, at the second stage, the precise global motion estimation is employed. For the reason that the rough motion parameters can be used as an initial guess of precise global motion estimation, the overhead of two-stage global motion estimation is small, compared with single stage global motion estimation.

If the feature points based global motion estimation is take as an example [7], at the first stage, only *P* feature points (typically *P*=200 - 500 when the total number of feature points is 1000) are used to roughly calculate the motion parameters while all feature points are used at the second stage. Note that, among all feature points, the motion vectors of the *P* feature points have been calculated at the first stage and do not need to be recalculated at the second stage.

### 2.4. Hybrid scheme

Since all these three techniques described above are independent of each other, they can be employed simultaneously as the hybrid scheme shown in Fig. 2. FS-Sprite is applied first to avoid redundant computation of global motion estimation for frames containing only little new information. Next, Multi-Sprite is applied to detect if a new sprite is required and the coordinate system of the current frame is adopted. Finally, after the current frame is warped, boundary matching technique can be applied to refine the position of the new pieces in the sprite.

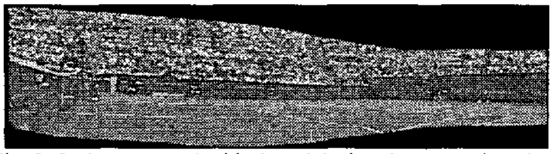
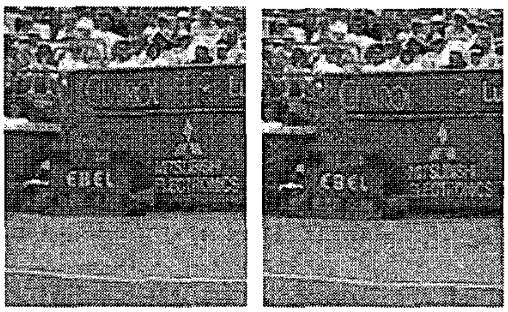


Fig. 3. Sprite generated with the original sprite generation algorithm.



(a) (b)

Fig. 4. A part of reconstructed frames of the original algorithm (a) without boundary matching and (b) with boundary matching at *Stefan* #99.

## 3. EXPERIMENTAL RESULTS

The experimental results of the proposed techniques are shown in this section. In these experiments, an algorithm, which is similar to [7], is implemented as the original sprite generation algorithm, and the experiments are carried out by use of the CIF test sequence *Stefan* with its mask sequence, and the total frame number is 300. Figure 3 shows the sprite obtained with the original sprite generation algorithm. There are some discontinuous lines in the sprite because the motion parameters are not accurate enough.

### 3.1. Boundary matching

In Fig. 4, the reconstructed frames of the original algorithm and the original algorithm with boundary matching are presented. It is obvious that the discontinuity in Fig. 4(a) is not satisfied. In Fig. 4(b), after boundary matching technique is employed, these discontinuous artifacts are depressed, and the subjective quality is improved. Note that for the sake of clear comparison, only a part of the reconstructed frames are shown.

### 3.2. Multi-Sprite

Table 1 shows the certain frames where new sprites are formed with different *Th_s* and *Th_r*. If Th_s=0.7 and Th_r=0.04, for example, four sprites are generated, and they start to be generated at frame 1, frame 245, frame 264, and frame 298, respectively. The sprites generated with Multi-Sprite technique are shown in Fig. 5. Some black regions exist in the sprites since the information of those regions are not available after the new sprites are generated. It will not influence the reconstructed frames since there must be foreground objects covering those regions. Multi-Sprite has better subjective performance especially when the frame has serious deformation or resolution changing. The former one is shows in Fig. 6, where the subjective quality of the reconstructed frame of Multi-Sprite is higher; the latter one is shown in Fig. 7, where the high resolution of the frame can be kept. Moreover, although the quality is improved,

Table 1. The first frames of multiple sprites.

| The First Frames | | Th_s 0.5 | 0.6 | 0.7 | 0.75 | 0.8 |
|---|---|---|---|---|---|---|
| Th_r | 0.05 | 1<br>248<br>270 | 1<br>248<br>270<br>298 | 1<br>248<br>270<br>296 | 1<br>248<br>270<br>294<br>300 | 1<br>248<br>270<br>292<br>296 |
| | 0.04 | 1<br>245<br>264 | 1<br>245<br>264 | 1<br>245<br>264<br>298 | 1<br>245<br>264<br>297 | 1<br>245<br>264<br>295<br>299 |

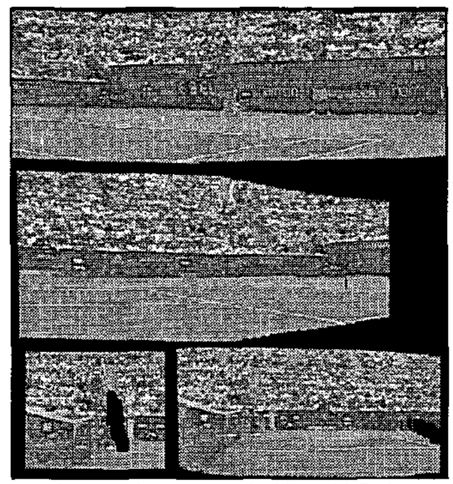
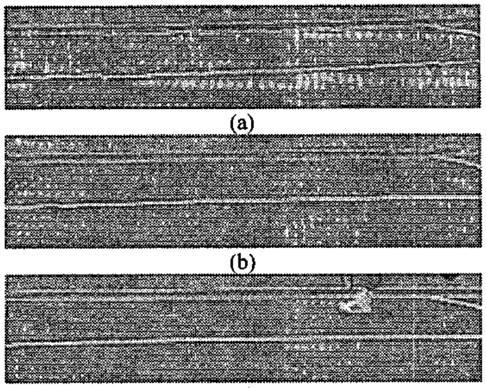


Fig. 5. Sprites generated with Multi-Sprite.

the calculating time of Multi-Sprite is almost the same. The computation complexity overhead is quite small.

### 3.3. FS-Sprite

Figure 8 shows the runtime of the original sprite generation algorithm accelerated with FS-Sprite technique. The test platform is a PC with a AMD K7-800MHz processor and a general C++ compiler. It shows that the runtime decreases when *P* is smaller and *Percentage* is higher. The runtime of MPEG-4 Video VM [4] on the same platform required 5462 s to generate the sprite, where the original algorithm needs 149 s, and it requires 151 s if boundary matching technique is employed. The original algorithm with boundary matching is 36 times faster compared with the method in MPEG-4 VM. If FS-Sprite technique is applied (also with boundary matching), 47.68% - 17.22% runtime can be further saved with different parameters *P* and *Percentage*. To achieve the same quality of the original algorithm, *P* should larger than 400, and *Percentage* should be smaller than 8%, and 21.2% runtime can be saved. With these parameters, the sprite generation process can be accelerated to be 46 times faster than MPEG-4 VM, where the fast sprite generation proposed in MPEG-4 part 7 is only 7 times faster [8]. The sprite generated with FS-Sprite is hard to be distinguished from that of the original algorithm. The comparison of the reconstructed frames with and without FS-Sprite technique (*P*=500, *Percentage*=6%) is shown in Fig. 9. The subjective quality with

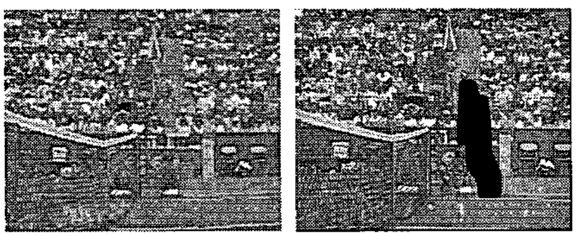


(a)

(b)

(c)

Fig. 6. A part of reconstructed frames of (a) the original algorithm with boundary matching, (b) Multi-Sprite, and (c) original sequence at *Stefan* #267.



(a) (b)

Fig. 7. Reconstructed frames of (a) the original algorithm with boundary matching and (b) Multi-Sprite at *Stefan* #298.

FS-Sprite is almost the same, even better than the original one because less discontinuity artifacts occur when some frames are skipped.

### 3.4. Hybrid scheme

When all the three techniques are applied, the first frames of multiple sprites are very similar to those in Table 1. The runtime with different Multi-Sprite conditions are almost the same. Moreover, with different FS-Sprite parameter $P$ and *Percentage*, the results of Multi-Sprite are quite the same, and the runtime is very similar to that with FS-Sprite only. These results show that Multi-Sprite and FS-Sprite techniques are independent of each other. With FS-Sprite, the first frames of multiple sprites will not change. With Multi-Sprite, the runtime will not increase. These techniques can perform well with each other and can keep their improvements of sprite generation.

## 4. CONCLUSION

In this paper, three novel techniques to improve the performance of sprite generation algorithm are proposed. Boundary matching can alleviate the discontinuity artifacts caused by inaccurate global motion estimation. Moreover, Multi-Sprite technique can generate more than one sprites to deal with complex camera motion and resolution changing and improve the subjective quality of reconstructed frames. Finally, FS-Sprite technique can skip redundant frames for sprite generation to accelerate the process. These techniques are effective and can be employed independently with any sprite generation algorithms.

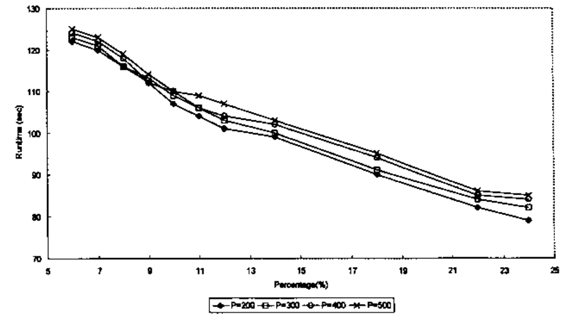
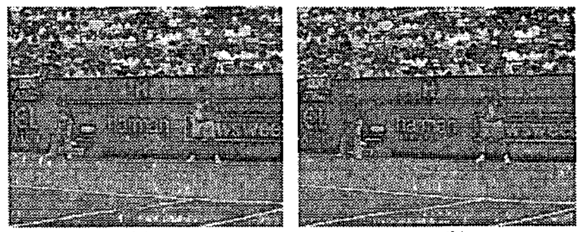



Fig. 8. Runtime analysis of FS-Sprite.



(a) (b)

Fig. 9. Reconstructed frames of (a) the original algorithm with boundary matching and (b) FS-Sprite with boundary matching at *Stefan* #186.

## 5. REFERENCES

[1] M. Irani, S. Hsu, and P. Anandan, "Video compression using mosaic representations," *Signal Processing: Image Communication*, vol. 7, no. 4-6, pp. 529-552, Nov. 1995.

[2] MPEG Video Group, *Text of 14496-2, Part 2: Visual*, ISO/IEC JTC1/SC 29/WG11 N4350, 2001.

[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," in *Proc. of IEEE International Symposium on Circuits and Systems 2000*, vol. 3, pp. 606-609, 2000.

[4] MPEG Video Group, *The MPEG-4 Video Standard Verification Model 18.0*, ISO/IEC JTC1/SC 29/WG11 N3908, 2001.

[5] M.-C. Lee, W.-G. Chen, C.-L. B. Lin, C. Gu, T. Markoc, S. I. Zabinsky, and R. Szeliski, "A layered video object coding system using sprite and affine motion model," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 1, Feb. 1997.

[6] H. Nicolas, "New methods for dynamic mosaicking," *IEEE Transactions on Image Processing*, vol. 10, no. 8, Aug. 2001.

[7] A. Smolic, T. Sikora, and J.-R. Ohm, "Long-term global motion estimation and its application for sprite coding, content description, and segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 8, Dec. 1999.

[8] MPEG Video Group, *Text of 14496-7 PDTR (Optimized Visual Reference Software)*, ISO/IEC JTC1/SC29/WG11 N4057, 2001.