A FAST AND HIGH SUBJECTIVE QUALITY SPRITE GENERATION ALGORITHM WITH FRAME SKIPPING AND MULTIPLE SPRITES TECHNIQUES

Shao-Yi Chien[*], *Ching-Yeh Chen, Wei-Min Chao, Chih-Wei Hsu, Yu-Wen Huang, and Liang-Gee Chen*

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

ABSTRACT

Sprite coding, which is a new coding tool in MPEG-4, can achieve high coding efficiency with high subjective quality at low bit rate. Many sprite generation algorithms have been proposed; however, the computational intensity is very high and the quality is not good enough because of the limitation of simple motion models. In this paper, a novel sprite generation algorithm is proposed with several new techniques. A frame skipping technique can generate the sprite with only several important frames to accelerate the process with similar subjective quality. In addition, a boundary matching and multiple sprites techniques can overcome the limitation of simple motion models to achieve high subjective quality with little computation overhead. Experiments show the proposed algorithm is 46 times faster than the algorithms in MPEG-4 VM and have high subjective quality. These techniques can be also applied with other sprite generation algorithms.

1. INTRODUCTION

A sprite is an image that collects information belonging to a video object in a video sequence. For still video objects, such as background, the associated video object planes (VOPs) in every frame are very suitable to be represented with a sprite, which is called sprite coding and is supported in MPEG-4 video standard as an efficient coding tool [1]. Sprite coding has been proven to have good coding efficiency, which can achieve high subjective quality with very low bit rate [2]. The most important part of sprite coding is the generation of sprite information, where global motion estimation and compensation are involved.

Many sprite generation algorithms have been proposed. In the verification model of MPEG-4 and several other algorithms, gradient descent based global motion estimation is applied [3][4]. In these algorithms, the gradient of error with respect to the motion parameters needs to be calculated in whole frame in each iteration, which introduces enormous computation. In [5], a full search image matching algorithm is further included as well as the gradient descent algorithm, whose high computational intensity makes sprite coding infeasible for real applications. A feature points hierarchical global motion estimation based algorithm is proposed in [6]. The computation load is lower; however, for real applications, a fast sprite generation algorithm is still required. Besides, the subjective qualities of the sprites generated with these algorithms are not satisfied. Many artifacts exist, which are caused by the limitation of the simple motion model with six, eight, or twelve motion parameters.

In this paper, a novel fast and high subjective quality sprite generation algorithm is proposed. A frame skipping technique is proposed to generate the sprite information with only several important frames, not all frames, to speed up the process, which is named as frame skipping sprite (FS-Sprite). In addition, a multiple sprites technique is also proposed in this paper, which can divide a sprite into many small sprites to overcome complicated camera motion with a simple motion model. It is named as Multi-Sprite. Furthermore, a boundary matching technique is also proposed to improve the subjective quality with small computation load.

This paper is organized as follows. First, the proposed algorithm, including FS-Sprite, Multi-Sprite, and boundary matching 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, an original sprite generation algorithm is first described. Then the algorithm is accelerated with FS-Sprite, and the subjective quality of the sprite is improved with boundary matching and Multi-Sprite. Note that FS-Sprite, Multi-Sprite, and boundary matching technique can be applied indepently.

2.1. Original sprite generation algorithm

The flowchart of the original sprite generation algorithm is shown in Fig. 1. It is based on global motion estimation with feature points.

2.1.1. Select feature points

In this algorithm, the first step is the selection of a set of N feature points (typically N=1000), whose motion vectors are more reliable. They are points that have largest Hessian value [6]:

$$\left[\left(\frac{d^2I(x,y)}{dx^2}\right)\cdot\left(\frac{d^2I(x,y)}{dy^2}\right)-\left(\frac{d^2I(x,y)}{dxdy}\right)^2\right].$$

[*] This work is partly supported by SiS education foundation.

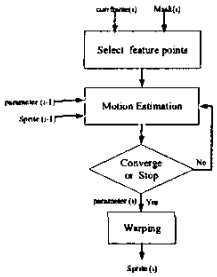


Fig. 1. Flowchart of the original sprite generation algorithm.

Note that, in order to improve the correctness of global motion estimation, these feature points need to be dispersed evenly in the frame. The frame is divided into four parts, and the feature points are respectively chosen in every part.

2.1.2. Motion estimation

The second step is motion estimation. Affine model is used to describe the motion vector, which can be shown as the following equation:

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

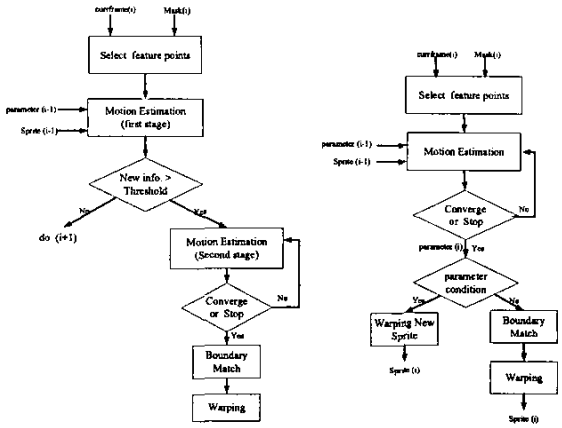
where (x, y) and (x', y') denote the coordinates of a point in the current frame and in the sprite, respectively. Because the feature points are Hessian points, the block matching algorithm in an L-neighborhood (typically L=4) around the feature point is applied. Initially, the corresponding position (x', y') of each feature point in the sprite of the current frame is predicted with the motion parameters of the last iteration or the previous frame. Then a new optimal matching position is searched around the predicted position. This motion vector prediction procedure can reduce the computation load by decreasing the search range and improve the correctness of motion vectors. Next, the motion parameters are calculated by LMS algorithm. This process is repeated iteratively until the process is convergent or the number of iterations is larger than ten.

2.1.3. Warping

According to the motion parameters and the affine model, the current frame is warped to update the sprite. Since (x', y') in the sprite usually do not correspond to integer grid in the current frame, a bilinear interpolation has to be applied.

2.2. FS-Sprite

When the camera motion is slow, or the camera is not moving, only little new information in the current frame needs to be added into the sprite. Some frames can be skipped to accelerate the sprite generation procedure with similar subjective quality. The flowchart of FS-Sprite is illustrated in Fig. 2(a). The motion estimation stage is divided into two stages. At the first stage, only P feature points (typically P=200~500) are used to roughly calculate motion parameters. According to the rough motion parameters, the percentage of new information that does not exist in the sprite can be estimated. If the percentage is larger than a threshold, *Percentage*, we continue to do the second stage, or this frame is skipped. When the current frame is skipped, the



(a) (b)

Fig. 2. Flowchart of (a) FS-Sprite and (b) Multi-Sprite.

motion parameters of this frame are the rough motion parameters, and large portion of computation is also skipped.

At the second stage, all feature points are used to calculate the motion parameters. The procedure is the same as that described in Sec. 2.1.2. Note that, among all feature points, the motion vectors of the P feature points have been calculated in the first stage and do not need to be recalculated in this stage.

2.3. Boundary matching

In a lot of experiments, we found the discontinuous lines or texture caused by the inaccuracy of global motion estimation is obvious and unsatisfied for human vision. In order to eliminate the discontinuous lines, a method called boundary matching is proposed. After warping, the new pieces to be added into the sprite can be found. The positions of these new pieces in the sprite are further refined. They are shifted vertically in the sprite, and then the refined position, which has the minimum boundary matching error, can be located. The motion parameters are also refined according to the optimal position.

2.4. Multi-Sprite

In the conventional sprite generation algorithm, only one sprite is generated. Every frame is warped into the sprite with respect to the coordinate system of the first frame. When the camera motion is large, the subjective quality of the sprite is degraded. For example, if the camera is zooming in, 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 than the first frame, the resolution is lowered to be the same with the first frame. Furthermore, for complex camera motion, simple motion model with six, eight, or twelve parameters cannot be successfully applied. In order to improve the quality of the reconstructed frame without extra computing complexity overhead, multiple sprites can be generated instead of only one sprite, which is named as Multi-Sprite. The flowchart of Multi-Sprite is shown in Fig. 2(b). Two thresholds are set for scaling and rotation to detect the resolution changing and deformation when the current frame is warped to the sprite domain. After motion parameters are derived, if the product of scale motion

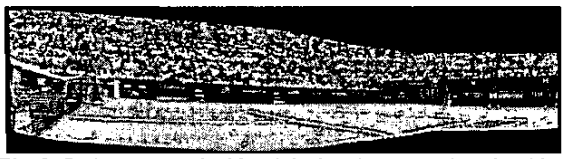
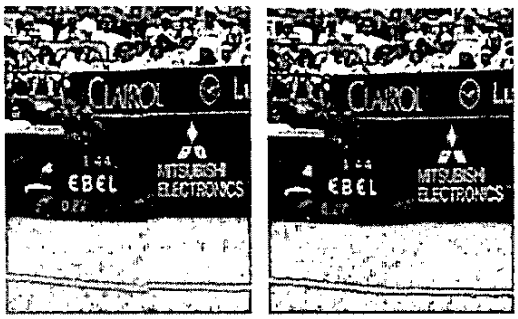


Fig. 3. Sprite generated with original sprite generation algorithm.



(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.

parameters, $a_2 \cdot b_3$, is smaller than the threshold *Th_s*, or the rotation parameters of motion parameter, a_3 or b_2, is larger than the threshold *Th_r*, a new sprite is formed with this frame as the first frame; otherwise, this frame is warped to the original sprite. If a new sprite is generated, it is also used for next frame, or the original sprite is used.

3. EXPERIMENTAL RESULTS

3.1. Original Sprite

The experiments have been carried out using the CIF test sequence *Stefan* with its mask sequence. Fig. 3 shows the sprite obtained with original sprite generation algorithm. There are some discontinuous lines in sprite because the motion parameters are not accurate enough.

3.2. Boundary Match

In Fig. 4, the reconstructed frame of the original algorithm and the original algorithm with boundary matching are shown. Obviously, the discontinuity in Fig. 4(a) is not satisfied. In Fig. 4(b), after boundary matching algorithm is applied, these discontinuous lines become smoother, and the subjective quality is improved. Note that only a part of the reconstructed frames are shown for clear comparison.

3.3. FS-Sprite

Table 1 shows the run time and the total number of frames that are actually warped to sprite. The test platform is a PC with a AMD K-7 800MHz processor and a general C++ compiler. The test sequence is *Stefan* in CIF format with 300 frames. To be brief, the original algorithm with boundary matching can speed up the sprite generation up to 36 times compared with the method in MPEG-4 Video VM [3]. FS-Sprite (also with boundary matching) can further reduce 47.68% ~ 17.22% computation with different parameters *P* and *Percentage*. To achieve the same quality of original algorithm with boundary matching, *P* should larger than 400, and *Percentage* should be smaller than 8%. At this choice, 21.85% run time can be reduced, and it is 46 times faster than MPEG-4 Video VM, where the fast sprite generation proposed in MPEG-4 part 7 is only 7 times faster [7]. Fig. 5 shows the sprite generated with FS-Sprite, and Fig. 6 shows the comparison of the reconstructed frame with original algorithm

Table 1. Comparison of run time.

Method	Time (sec), No. of warped frame			
Original algorithm	149, 300			
With boundary matching	151, 300			
VM Sprite	5462, 300			
FS-Sprite	P			
Percentage	200	300	400	500
24%	79, 37	82, 36	84, 38	85, 37
18%	90, 74	91, 73	94, 72	95, 74
12%	101, 114	103, 116	104, 118	107, 118
10%	107, 136	109, 137	110, 138	110, 136
8%	116, 170	116, 167	118, 167	119, 167
7%	120, 183	121, 184	122, 185	123, 185
6%	122, 195	123, 195	124, 195	125, 195

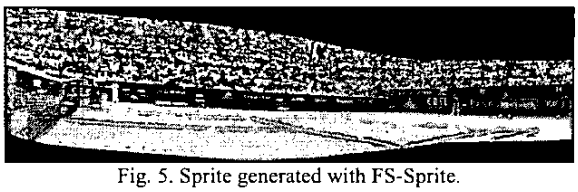


Fig. 5. Sprite generated with FS-Sprite.

(a) (b)

Fig. 6. Reconstructed frames of (a) the original algorithm with boundary matching and (b) FS-Sprite with boundary matching at *Stefan* #186.

Table 2. 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	1 248 270	1 248 270 298	1 248 270 296	1 248 270 294 300	1 248 270 292 296
	0.04	1 245 264	1 245 264	1 245 264 298	1 245 264 297	1 245 264 295 299

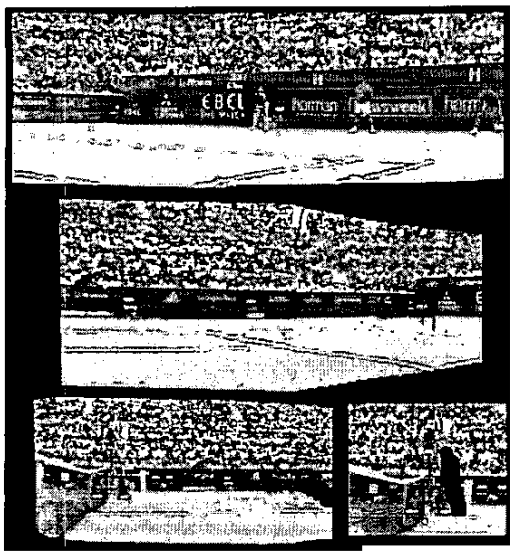


Fig. 7. Sprite generated with Multi-Sprite.

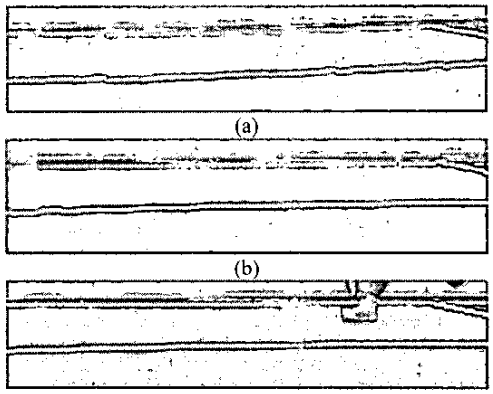
with boundary matching and FS-Sprite (P=500, *Percentage*=6%) with boundary matching. The quality of FS-Sprite is the same, sometimes even better because less discontinuity effects occur when some frames are skipped.

3.4. Multi-Sprite

For different *Th_s* and *Th_r*, Table 2 shows the frames where new sprites are formed. For example, if *Th_s*=0.7 and *Th_r*=0.04, four sprites are generated, and the first sprite starts to be generated at frame 1, the second starts to be generated at frame 245, the third starts at frame 264, and the fourth starts at frame 298. Fig. 7 shows the sprites generated with Multi-Sprite. There are some black regions in sprites because the information of these regions do not valid after we create a new sprite, namely, there should be a foreground object covering these regions when the frame is reconstructed, thus it will not influence the final reconstructed results. Multi-Sprite has better subjective performance especially when the frame has serious deformation or resolution changing. The former one is shows in Fig. 8, where the subjective quality of the reconstructed frame of Multi-Sprite is higher; the latter one is shown in Fig. 9, where the high resolution of the frame can be kept. Moreover, although the quality is improved with Multi-Sprite, the calculating time of Multi-Sprite is almost the same. The computation complexity overhead is quite small.

4. CONCLUSION

In this paper, a new sprite generation algorithm is proposed. The sprite can be generated fast with frame skipping based algorithm, FS-Sprite. Besides, high subject quality can be achieved with multiple sprites technique, Multi-Sprite, and boundary matching with only little computation overhead. Experiments show the proposed algorithm is 46 times faster than that proposed in MPEG-4 VM and can give high subjective quality reconstructed frames.

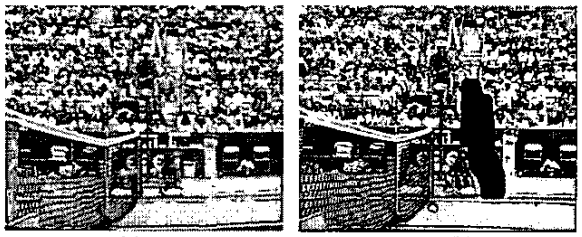


(a)

(b)

(c)

Fig. 8. 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. 9. Reconstructed frames of (a) the original algorithm with boundary matching and (b) Multi-Sprite at *Stefan* #298.

5. REFERENCES

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

[2] K. Jinzenji, S. Okada, H. Watanabe, N. Kobayashi, "Automatic two-layer vide 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.

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

[4] 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.

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

[6] 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.

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