International Journal of Pattern Recognition
and Artificial Intelligence
Vol. 20, No. 8 (2006) 1139–1158
© World Scientific Publishing Company

A HIGH VISUAL QUALITY SPRITE GENERATOR USING INTELLIGENT BLENDING WITHOUT SEGMENTATION MASKS

I-SHENG KUO and LING-HWEI CHEN*

Department of Computer and Information Science
National Chiao Tung University, 1001 Ta Hsueh Rd.
Hsinchu, Taiwan 30050, R.O.C.
**lhchen@cc.nctu.edu.tw*

The sprite generator introduced in MPEG-4 blends frames by averaging, which will make places, that are always occupied by moving objects, look blurred. Thus, providing segmented masks for moving objects is suggested. Several researchers have employed automatic segmentation methods to produce moving object masks. Based on these masks, they used a reliability-based blending strategy to generate sprites. Since perfect segmentation is impossible, some ghost-like shadows will appear in the generated sprite. To treat this problem, in this paper, an intelligent blending strategy without needing segmentation masks is proposed. It is based on the fact that for each point in the generated sprite, the corresponding pixels in most frames belong to background and only few belong to moving objects. A counting schema is provided to make only background points participate in average blending. The experimental result shows that the visual quality of the generated sprite using the proposed blending strategy is close to that using manually segmented masks and is better than that generated by Lu-Gao-Wu method. No ghostlike shadows are produced. Furthermore, a uniform feature point extraction method is proposed to increase the precision of global motion estimation, the effectiveness of this part is presented by showing the comparison results with other existing method.

Keywords: Sprite generation; background mosaic; feature point extraction; global motion estimation; blending; MPEG-4.

1. Introduction

A sprite, which is also referred as "background mosaics",[2,3,12,13] is an image that is formed by collecting backgrounds of a video sequence in a scene. The sprite provides a panoramic view of the scene and is efficient to represent the backgrounds. The technique of extracting backgrounds and obtaining a sprite is named as "sprite generator" and is adopted as an efficient tool in the MPEG-4[4] video standard. The coding of a sprite, instead of coding all backgrounds in each frame of a scene, can achieve very low bit-rate with good quality.

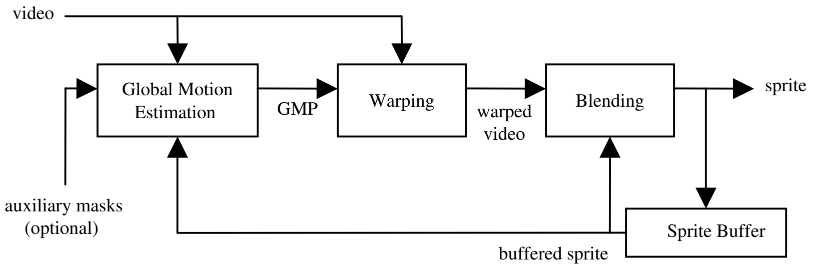


Fig. 1. The sprite generator proposed by MPEG-4.

MPEG-4 VM[5] had provided a framework of sprite generator as shown in Fig. 1. The framework contains three parts: global motion estimation (GME), frame warping and frame blending. The GME aims at finding the spatial location variation, which is caused by the camera motion, of the background in the current frame relative to the current sprite. The camera motion can be represented by parameters of some geometric models often denoted as global motion parameters (GMP). Gradient descent based algorithms[10] are widely used in the estimator. These algorithms usually need a good initial guess to avoid the solution being local optimum, and an error function is required to evaluate the performance of different GMPs. The squared error between the current frame and the current sprite is usually employed as the error function. In order to raise speed, only some points are selected as feature points and involved in the computation of the error function.[6–8,11] The selection of feature points affects the estimation accuracy, especially when the number of feature points is small. Thus, how to select few representative feature points is an important issue. Most existing methods[6–8,11] take those points with higher variations in spatial or temporary domain; some important ones may be lost with moderate variations and some moving object points with higher variations may be included. To avoid this disadvantage, in this paper, a uniform feature point extractor is provided to increase the estimation precision for GMPs.

After obtaining GMPs, the current frame is first geometrically transformed (also called warped) to a warped one with the same camera view as the current sprite.

The warped frame is then blended into the current sprite by a blending strategy. In MPEG-4's framework, simple averaging is employed as the blending function. This will make some places, that are always occupied by moving objects, blurred. To avoid the disadvantage, providing manually-segmented masks has been suggested. The segmentation masks are used to distinguish moving objects from background such that moving objects will not be involved in blending and the quality of the generated sprite can be significantly improved. However, segmenting masks manually is impractical.

Several sprite generators have been proposed,[6–8,11,14] most followed the MPEG-4's framework. Smolić *et al.*[11] proposed a hierarchal long-term global motion estimator and a reliability-based blending strategy to generate sprite. Watanabe and

Jinzenji[14] presented a sprite generator with two-pass blending and automatic foreground object extraction. In the first pass of sprite blending, a provisional sprite is constructed using the temporal median. Then the foreground objects are extracted automatically based on the provisional sprite. A difference image of the current frame from the provisional sprite is calculated, it is used to classify the pixels of the current frame into those of foreground and background. Then the current frame is divided into blocks, and each block is classified as either a foreground or a background according to the number of foreground pixels inside the block. The foreground blocks are excluded in the second pass of sprite blending. There are two disadvantages. One is that getting a perfect segmentation is impossible since a good threshold is needed in block classification, and the block used as the classification unit will roughly make segmentation. The other is that the additional pass doubles the blending time.

Lu *et al.*[6–8] used a more precise segmentation method proposed by Meier and Ngan[9] to obtain moving object masks. Based on the obtained masks, the reliability-based blending strategy inspired from Smolić *et al.*[11] is developed to generate sprite.

Note that to avoid sprite being blurred, pixels of moving objects must be excluded from being blended into the sprite. If the segmentation is perfect, the averaging blending provided in MPEG-4 can achieve excellent quality; otherwise, the generated sprite will be blurred around moving object boundary due to which some pixels of moving objects are considered as background. However, a perfect segmentation is impossible, the above mentioned methods[6–8,11] use the reliability-based blending concept provided in Ref. 11 to solve this problem. In the reliability-based blending strategy, a frame is divided into reliable, unreliable and undefined regions according to the segmented masks. Pixels denoted as objects in the segmented masks are classified as undefined pixels, and pixels near mask borders or frame borders (within a given distance) are classified as unreliable ones. The remaining pixels are classified as reliable ones. The reliable and unreliable pixels are average-blended separately, and the blended pixels with the highest reliability are chosen into the sprite.[8] The undefined pixels do not contribute to the sprite blending. The given distance from the mask border must be large enough to cover all segmentation faults, or the generated sprite will have ghost-like shadows in some places. However, it is hard to decide the distance automatically. In this paper, we will provide a sprite generator to avoid above-mentioned problems.

The proposed method is based on the MPEG-4's framework shown in Fig. 1, but the demand of segmentation masks is removed. A uniform feature point extractor with object point removing is proposed. With the proposed feature points, the precision of estimated global motion parameters are increased significantly. A new blending strategy that does not need segmentation masks is also proposed. The moving objects are excluded from blending by a counting schema. The proposed method provides higher visual quality of the generated sprite than those existing methods, and the average PSNR of reconstructed backgrounds is increased slightly.

The rest of the paper is organized as follows. Section 2 describes the proposed sprite generator. The uniform feature point extractor and the novel intelligent blender are presented. Section 3 shows the experimental results and some comparisons with existing methods. Conclusions are presented in Sec. 4.

2. The Proposed Sprite Generator

In the proposed sprite generator, a local-to-global GME is provided with a novel feature point extraction method to get GMPs. With the estimated GMPs, each input frame is warped. An intelligent blending strategy is then presented to blend the warped frame to form a sprite. The details of the proposed generator are described as follows.

2.1. *Global motion estimation*

The aim of global motion estimation is to obtain an accurate estimation of camera motion between the current frame and a reference image, e.g. the current sprite. In this paper, we take the perspective transformation to model camera motion as follows:

$$x' = \frac{m_1x + m_2y + m_3}{m_7x + m_8y + 1}, \quad y' = \frac{m_4x + m_5y + m_6}{m_7x + m_8y + 1}, \tag{1}$$

where (x, y) and (x', y') denote the coordinates of a pixel before and after the camera motion respectively. $m_1, m_2, \ldots, m_8$ are the transformation parameters also referred to as global motion parameters (GMPs).

The GME can be described by a minimization problem:

$$P^* = \arg\min_P E(I, I', T_P) \tag{2}$$

where I and I' are the current frame and the reference image respectively. T_P is the transformation function with global motion parameters P. P^* is the estimated parameters. $E(I, I', T_P)$ is an error function defined by user. The estimation registers pixels in the current frame into the reference image by finding the parameters which minimize the error between the current frame and the reference image. In this paper, the squared error is chosen as the error function, that is,

$$E(I, I', T_P) = \sum_{x,y \in I} \left(I(x, y) - I'(T_P(x, y))\right)^2. \tag{3}$$

To solve the minimization problem, the Levenberg–Marquardt algorithm[10] based on the gradient descent method is used. Since the gradient descent method has a risk of being trapped into a local minimum, a good starting point called an "initial guess" should be provided.

To generate a sprite from a video sequence, the first frame is copied into the sprite directly, and the camera motions between the following frames and the first frame must be estimated by the GME. At the beginning of the sequence, it is easy

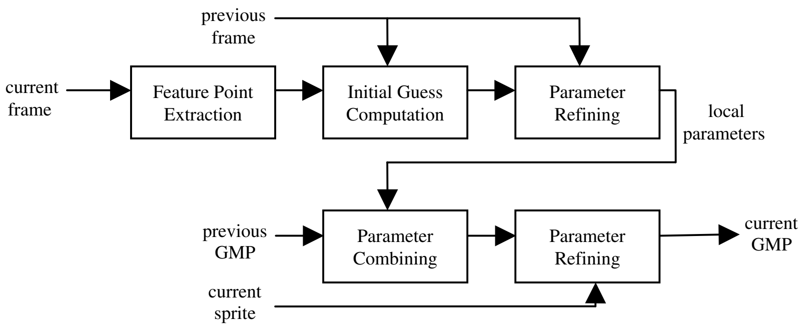


Fig. 2. The two-staged GME method.

to find a good initial guess due to which the camera does not move too far. It becomes hard to find a good initial guess when the camera motion of the current frame relative to the current sprite is large. This problem is solved by a two stage GME schema shown in Fig. 2. The first stage estimates the motion parameters called the local parameters between the current frame and the previous frame, i.e. the frame before the current frame. Finding an initial guess of the local parameters is easy because the variation of camera motion between two successive frames is small. Based on the estimated local parameters, the initial guess of global motion parameters can be computed in the second stage by combining the local parameters and the global motion parameters of the previous frame. Then the gradient descent method is employed to estimate the global motion parameters. The details will be described in the following paragraphs.

2.1.1. *Feature point extraction*

The iterative minimization of the gradient descent method is time consuming. To reduce the time complexity, only some selected feature points in the current frame are employed while computing the registration error.[11] In order to avoid the aperture problems described in Ref. 1, the Hessian value,[1] defined by

$$H(x,y) = \left(\frac{d^2I(x,y)}{dx^2} \cdot \frac{d^2I(x,y)}{dy^2} - \left(\frac{d^2I(x,y)}{dxdy}\right)^2\right), \tag{4}$$

is employed to find feature pixels in many previous researches.[6–8,11] Those points with Hessian values being local maximum or minimum are considered as feature points.[11] An example of using Hessian value to extract feature points is shown in Fig. 3. The grayscale of each pixel in Fig. 3(b) represents the absolute Hessian value of the corresponding pixel in Fig. 3(a).

Conventional methods[6–8,11] choose pixels with largest absolute Hessian values as feature points. However, the distribution of pixels with large absolute Hessian values does not spread uniformly, as shown in Fig. 3(b). Figure 3(c) shows the feature points extracted by these methods. The extracted feature points are concentrated

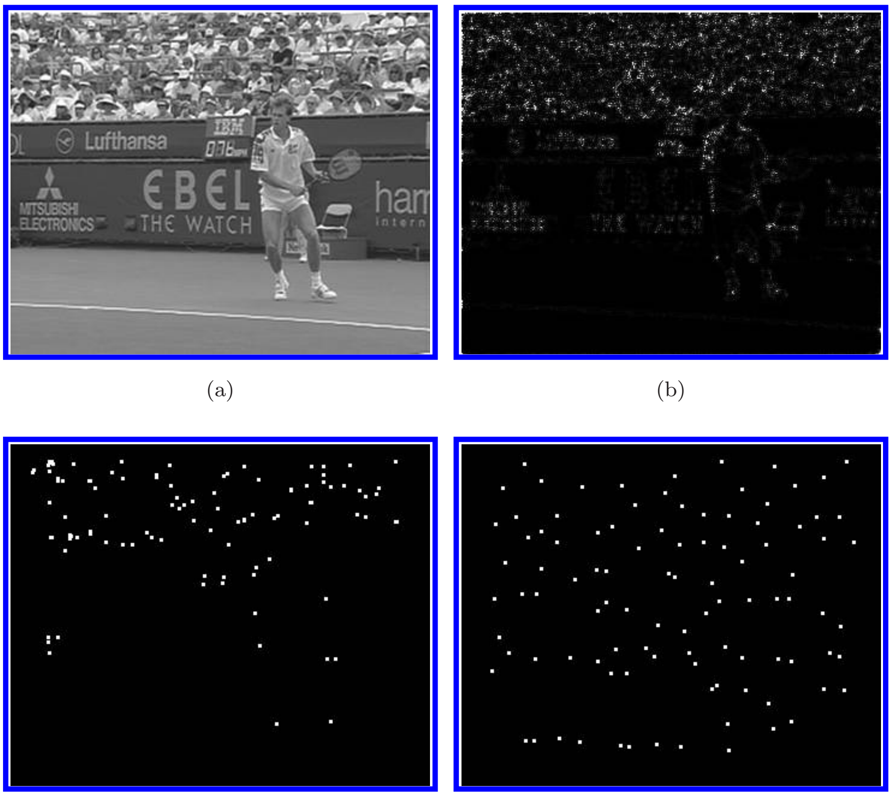
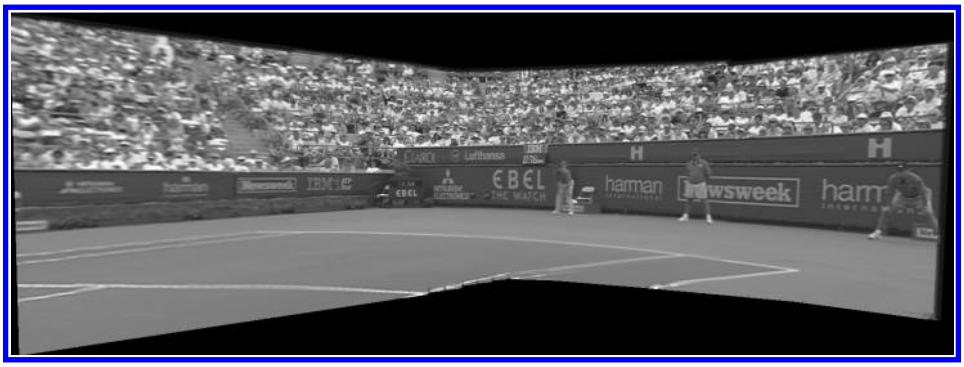


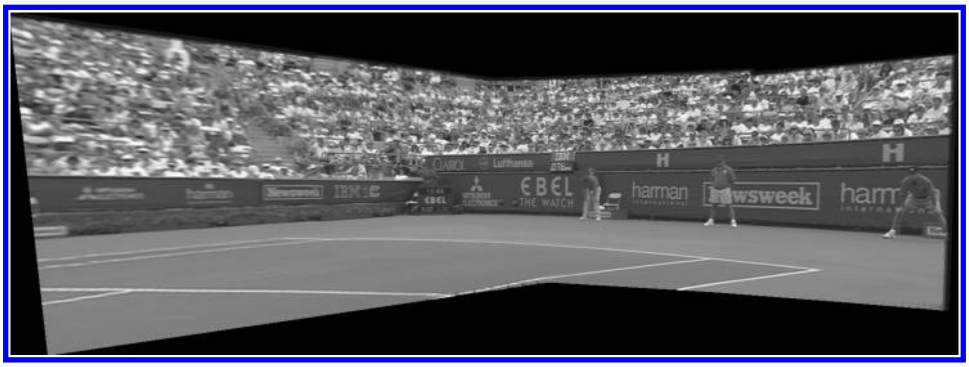
Fig. 3. Feature point extraction based on Hessian value. (a) Original image. (b) Absolute Hessian values of (a). (c) Feature points extracted by conventional methods. (d) Feature points extracted by the proposed method.

in the half-upper of the image. This will degrade the accuracy of the estimated parameters because the registration will be focused only on the half-upper of the image. This degradation will become more serious, when the number of feature points is small. The sprite generated based on these feature points is shown in Fig. 4(a). Although the half-upper of the sprite looks well, the white lines in the half-bottom of the sprite are not fitted correctly such that they look blurry [see Fig. 4(c)]; the reason is that no white line points are considered as feature points. To overcome this problem, the feature points must be selected uniformly. A uniform feature point extraction method is proposed and described as follows.

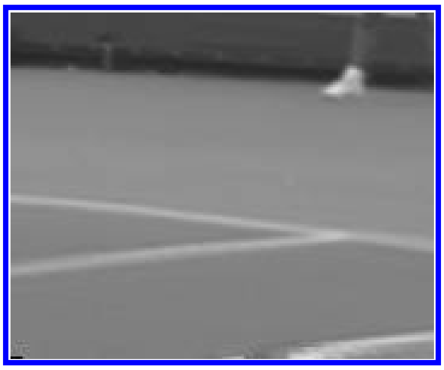
For an image of width W and height H, its border area of width B is excluded first. The rest is divided into 256 nonoverlapping blocks. For each block, the gray value variance is calculated to test its homogeneity. The block will be classified as a homogeneous one if its variance is smaller than a preset threshold T_V. The feature points are extracted uniformly in the nonhomogeneous blocks to avoid the



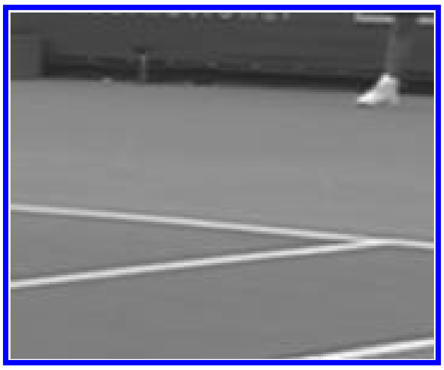
(a)



(b)



(c)



(d)

Fig. 4. Two examples to show the sprites generated using different feature points with the same number. The sprite generated using feature points extracted by (a) conventional methods, (b) the proposed method. (c) A close look of the white line in (a). (d) A close look of the white line in (b).

aperture problem. Suppose that we want to extract N feature points from K non-homogenous blocks, N/K pixels with largest absolute Hessian values are chosen in each non-homogeneous block. Feature points extracted from Fig. 3(a) using the proposed method are shown in Fig. 3(d). In contrast to the result of using the conventional method (see Fig. 3(c)), the distribution of feature points using the proposed method is uniform. Several points on the white line in the half-bottom of the frame are extracted as feature points, which will let the white line register well and significantly improve the visual quality of the generated sprite. However, from Fig. 3(d), we also find some points on the player located which will reduce the accuracy of estimated GMPs. As mentioned previously, we should avoid taking moving objects as feature points. Since the motions of moving objects usually differ from the motion of background, this provides us a clue to remove these outliers.

Traditional translation-based motion estimation is applied on each feature point to find the motion vector relative to the previous frame. In order to reduce the searching time, a global translation is found based first on some selected feature points, then a full search around the global translation for each feature point is performed. A 17×17 block centered at each selected feature point is used to find the global translation. A fully-searched motion estimation with a large search window (64×64) is processed on the 17×17 block. To raise up the searching speed, only 100 pixels with the largest absolute Hessian values among all feature points found previously are employed. The occurrences of the estimated 100 motion vectors are counted, and the motion vector with the highest occurrence is considered as the global translation. The motion vectors of all feature points are found by searching around the global translation with a smaller search window (17×17).

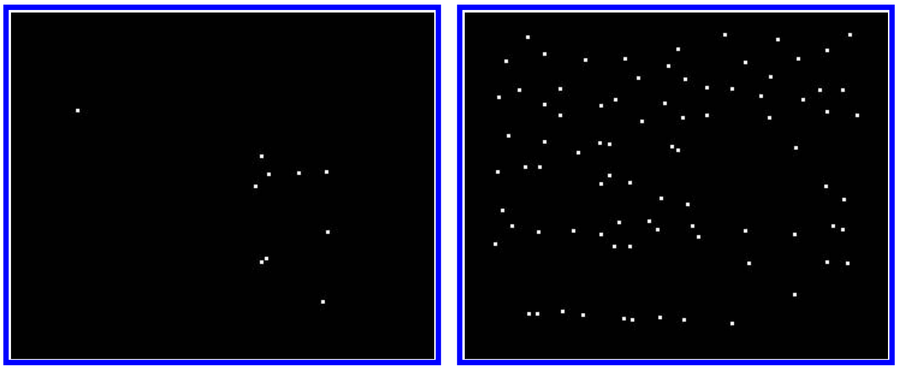
Let (dx, dy) be the motion vector estimated, the feature point is considered as an outlier if its mean-squared-error (MSE) between the original and the motion-estimated blocks is larger than a preset threshold T_O, i.e.

$$\frac{1}{N_B}\sum_{x,y\in B}\left(I(x,y) - I'(x+dx, y+dy)\right)^2 > T_O \tag{5}$$

where B is the block centered at the feature point and N_B is the number of pixels in the block, $I(x,y)$ and $I'(x,y)$ are the current frame and the previous frame, respectively. Since objects are assumed to have different motions from the background, their best motion vectors are usually not around the global translation (a rough approximation of the background motion), and their MSEs are likely to be higher with inaccurate motion vectors. They will be considered as outliers in Eq. (5).

Figure 5(a) illustrates the object pixels found from the feature points shown in Fig. 3(d). The feature points on the player are detected successfully. These object pixels are removed from the original feature points and the final feature points are shown in Fig. 5(b).

Figure 4(b) shows the sprite generated using the proposed uniform feature point extraction method, the same number of feature points as Fig. 4(a) is used. A close view of the white lines in Fig. 4(b) is shown in Fig. 4(d). From this figure, we can see



(a) (b)

Fig. 5. An example for outlier removing. (a) Detected object pixels in Fig. 3(d). (b) The feature points after removing outliers from Fig. 3(d).

that those white lines in the generated sprite are registered very well. The average PSNR of the reconstructed backgrounds using the nonuniform and uniform feature points are both 26.25 dB. Although the PSNR is the same, the proposed method achieves much better visual quality.

2.1.2. *Gradient descent method*

Each feature point (x, y) in the current frame and its corresponding point (x', y') in the reference image form a feature point pair, which are used to find the initial guess for camera motion. The corresponding point is defined to be the motion-estimated point of the feature point, i.e. $(x', y') = (x + dx,\ y + dy), (dx, dy)$ is the motion vector.

As mentioned previously, the perspective transformation expressed in Eq. (1) has eight parameters $m_1, m_2, \ldots, m_8$. By substituting each pair of (x, y) and (x', y') into Eq. (1) respectively, two equations will be built. Thus, four feature point pairs are sufficient to solve the eight parameters. However, in practice, the corresponding points found by motion estimation are not precise enough to provide a correct solution. Instead of using four feature point pairs, all feature point pairs found are applied to form an over-determined set of equations. The initial guess is computed by solving the set of equations under the sense of the least-squared error.

Let us recall the minimization problem described in the beginning of this section. An initial guess is required for the gradient descent method to find the global/local motion parameters. The Levenberg–Marquardt method[10] is employed here. The error function required in the gradient descent method, is slightly different from Eq. (3). Only the errors of the feature points are counted in the error function, that is,

$$E(I, I', T_P) = \sum_{(x,y)\in\text{feature\_points}} \left(I(x, y) - I'(T_P(x, y))\right)^2. \tag{6}$$

The gradient descent method is applied in the estimation of the local parameters and the global parameters. While estimating the local parameters, the reference image is defined as the previous frame. And the reference image is defined as the current sprite in the case of estimating the global parameters.

2.2. *Warping*

The current frame is warped toward the sprite using the same method described in MPEG-4's system. Let I be the current frame, I_W be the warped frame and defined as

$$I_W(x, y) = I_F(T_P(x, y)), \tag{7}$$

where T is the transformation and P is the estimated global motion parameters.

Since the transformed coordinates are not integers, bilinear interpolation[4] is applied while generating the warped frame.

2.3. *Intelligent blending*

The precision of segmentation mask affects the quality of the generated sprite. Although the unreliable region around segmentation mask boundary reduces the segmentation error in the reliability-based blending, some errors still cannot be covered. Figure 6 shows two frames with segmentation errors. The left foot of the player in both frames is not completely segmented; this will leave some ghostlike shadows after blending. A close view of the shadows in the blended sprite is shown in Fig. 7(a). Increasing the distance from the mask border given in the reliability-based blending schema may solve this problem, but other problems will occur. More segmentation errors can be covered using a larger distance, but it also increases the number of pixels being classified as unreliable. Thus the opportunity of an unreliable pixel being replaced by a reliable one is decreased. The reliable and unreliable pixels are blended by averaging separately. If an unreliable pixel is not replaced by a reliable one, the blending acts like the normal averaging. Figure 7(b) shows a close view of the blended sprite using a larger distance. We can see that the right border, which is the boundary of reliable and unreliable regions, is blurred. Thus, providing a suitable distance is a hard job. To avoid this problem, an intelligent blending strategy without requiring segmentation masks is proposed here.

The proposed intelligent blending strategy is based on a fact that for a series of pixels in video frames corresponding to the same location of a sprite, most pixels will be in the background; only few pixels are moving objects. Since objects are moving, those object pixels will come from different positions of an object, or even different objects, their intensities will have larger variation. On the contrary, intensities of those background pixels for the same background point in the real world should be similar, and can be found by counting their occurrence.

Figure 8 shows a flowchart of the proposed intelligent blending schema. Let X be the incoming pixel, S be the current sprite pixel. A candidate pixel C is used to

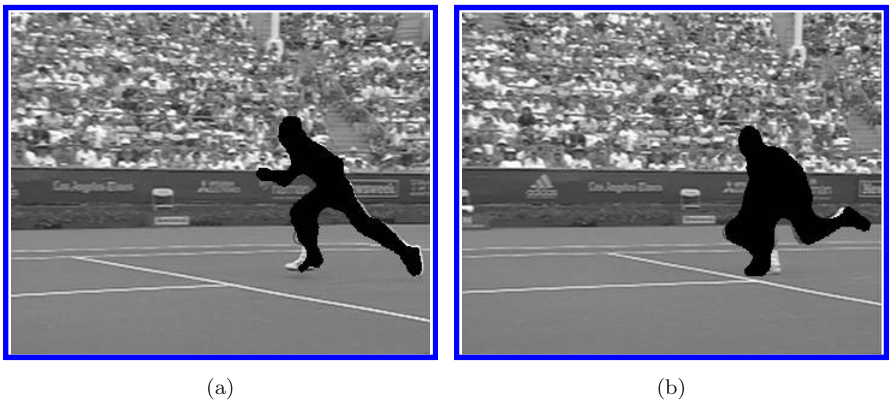


Fig. 6. Segmentation errors in player's feet. (a) Frame #255 with left foot incompletely segmented. (b) Frame #258 with both feet incompletely segmented.

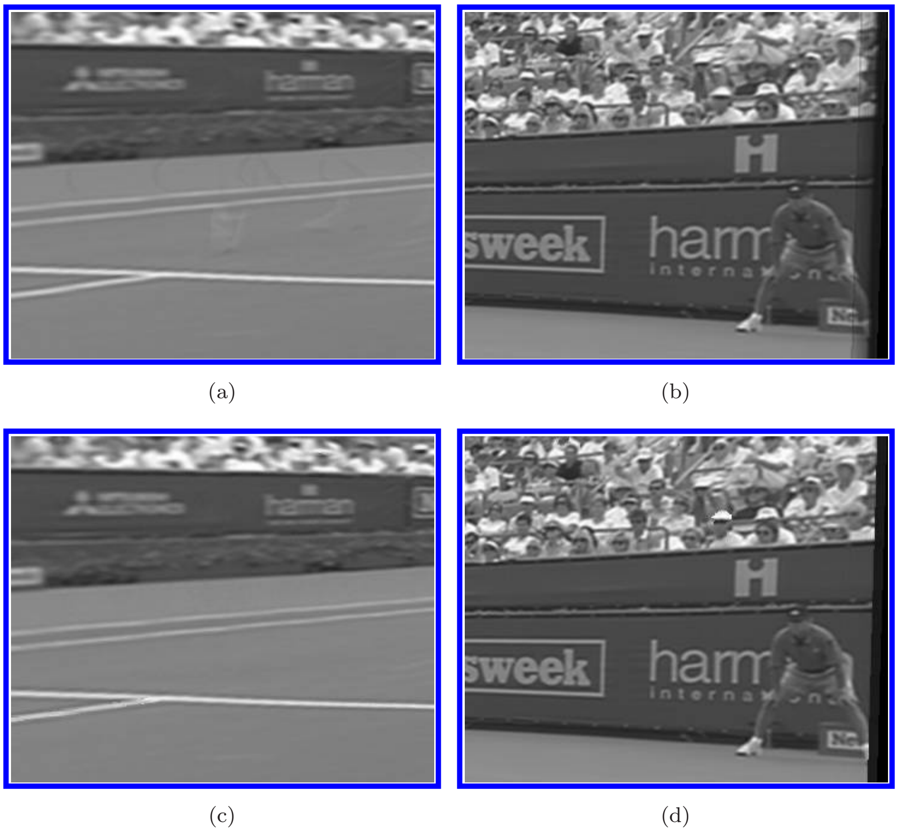


Fig. 7. Two examples to show the blended sprites using different methods. (a) The first example of the generated sprite based on the reliability-based blending. (b) The second example of the generated sprite based on reliability-based blending. (c) The first example of the generated sprite based on intelligent blending. (d) The second example of the generated sprite based on intelligent blending.

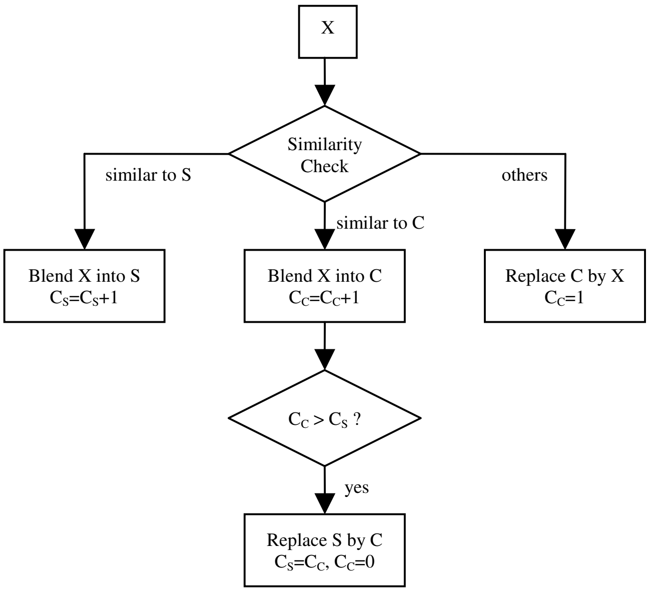


Fig. 8. Flowchart of the intelligent blending.

store a candidate of incoming background pixel. Two counters C_S and C_C are used to store the number of pixels being blended into S and C, respectively. Initially, S and C are undefined and both counters are set as zero. A similarity check is performed on the incoming pixel by calculating two absolute differences:

$$\begin{aligned} D_S &= |X - S| \\ D_C &= |X - C|. \end{aligned} \tag{8}$$

At the start of the blending, since S is undefined, it is set to be the gray value of the first incoming pixel and C_S is set to one. After filling S, the similarity check is conducted. If an incoming pixel is unlike S (i.e. D_S is greater than a preset threshold T) and C is undefined, C is set to the gray value of the incoming pixel and C_C is set to one; otherwise, if D_S is smaller than T and D_C, the incoming pixel is considered as a background pixel and is blended into S, C_S is increased by 1. If D_C is smaller than T and D_S, the incoming pixel is blended into the candidate C, C_C is increased by 1. Two counters are compared when a pixel is blended into the candidate C. If the candidate counter is larger than the sprite counter, the sprite and the sprite counter are replaced by the candidate and the candidate counter. Then the candidate and its counter are reset to undefined and zero, respectively. This replacement is based on the fact described before. Since the candidate appears more frequently than the current sprite, the candidate is more likely to be the background.

If both D_S and D_C are larger than T, the candidate is replaced by the incoming pixel, and the candidate counter is set to one. With a series of incoming object pixels, the candidate is continuously replaced by the incoming pixels until a background pixel is replaced into the candidate. If the background pixels appear continuously, the accumulation of candidate counter will begin.

The boundaries of frames are often unreliable and should be removed from blending.[11] In the proposed method, the effects of boundary are eliminated by counting the boundary pixels once. The pixels near the frame border within a preset distance W are defined as boundary pixels. The boundary pixels are checked and blended into the sprite or the candidate as normal pixels, but the corresponding counter is not increased. This will ensure the boundary pixels to be replaced quickly when normal pixels are input.

Two examples of the blending results using the proposed intelligent blender are shown in Figs. 7(c) and 7(d). In contrast to the results using reliability-based

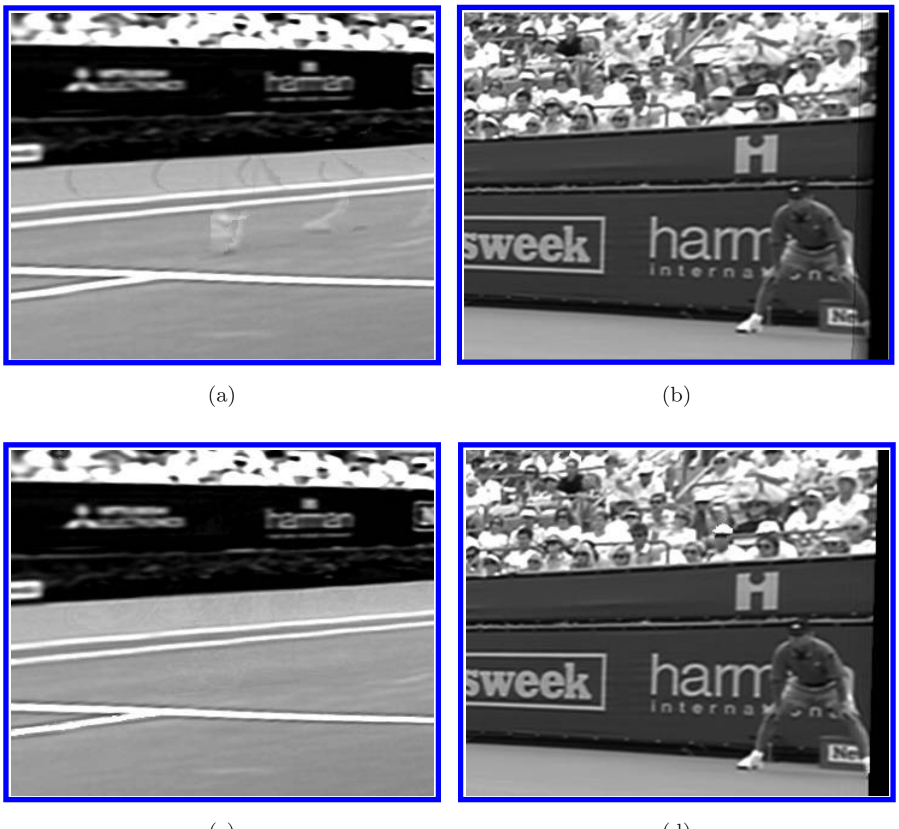
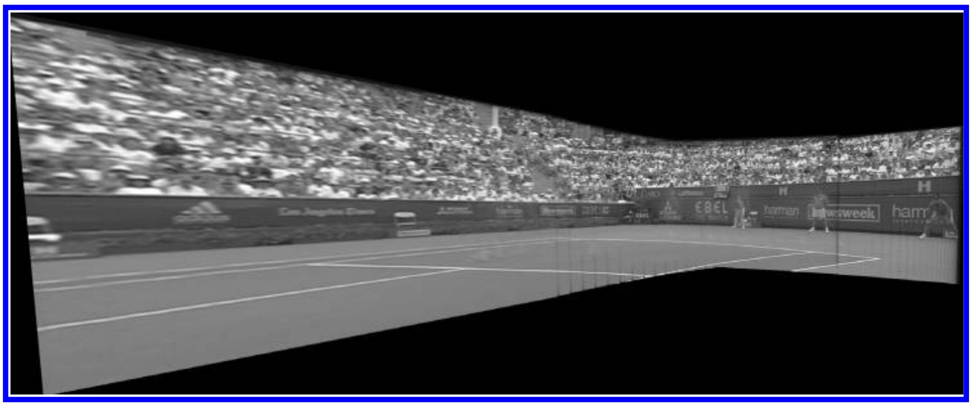


Fig. 9. Contrast-enhanced version of Fig. 7. (a) Contrast-enhanced version of Fig. 7(a). (b) Contrast-enhanced version of Fig. 7(b). (c) Contrast-enhanced version of Fig. 7(c). (d) Contrast-enhanced version of Fig. 7(d).

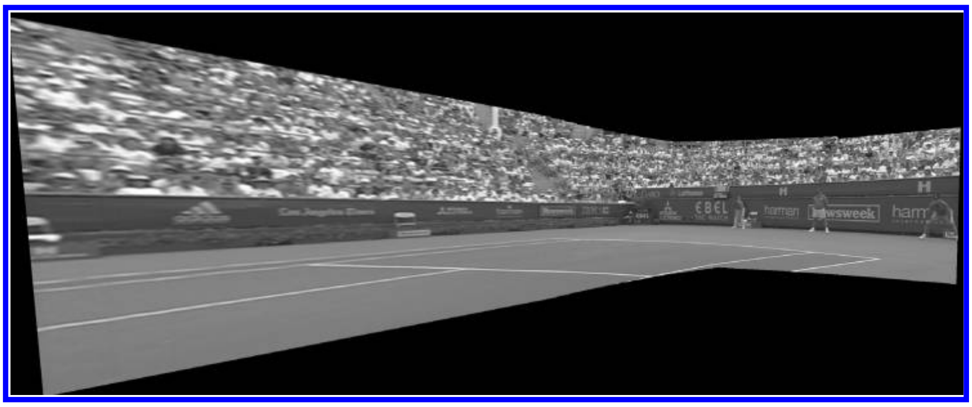
blender shown in Figs. 7(a) and 7(b), the ghostlike shadows are eliminated and the sprite border is clear and sharp. In order to examine the results easier, a contrast-enhanced version of Fig. 7 is provided in Fig. 9.

3. Experimental Results

The aim of sprite generation is to reconstruct the background from the generated sprite perfectly. The quality of the reconstructed background for each frame is often measured by PSNR. In most cases, the PSNR is a good measurement to describe the quality. However, the PSNR is fooled in seldom cases. A comparison in visual quality of the generated sprites is also performed.

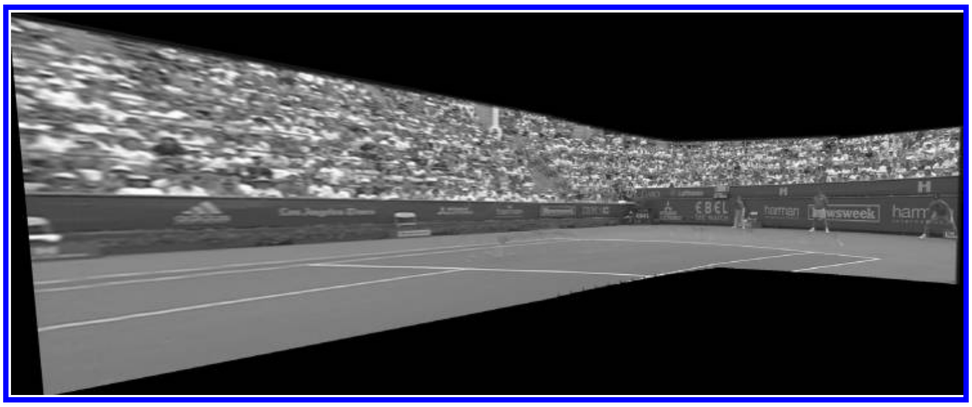


(a)

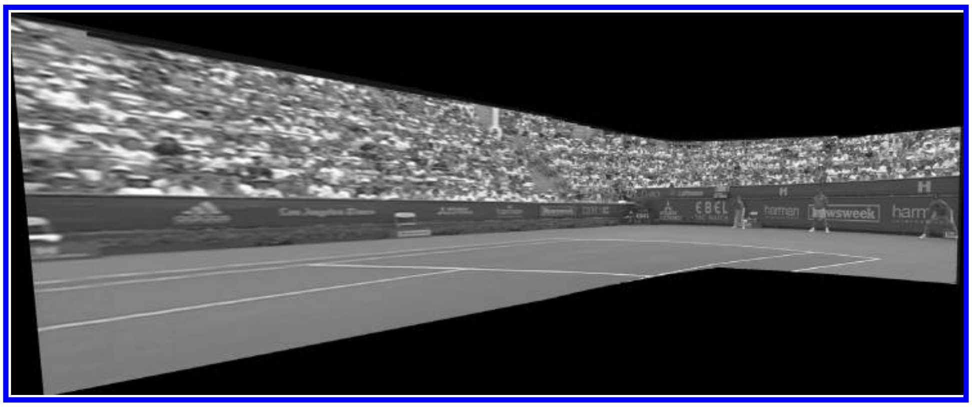


(b)

Fig. 10. The generated sprites. (a) Sprite generated using averaging blending without referring to segmentation masks. (b) Sprite generated using averaging blending based on manually segmented masks. (c) Sprite generated using reliability-based blending. (d) Sprite generated using intelligent blending.



(c)



(d)

Fig. 10. (*Continued*)

The GMPs are estimated using the two-staged GME with the proposed uniform feature points described in Sec. 2.1. Then sprites are mixed by different blending strategies with the same GMPs. Backgrounds are reconstructed from the generated sprites and their PSNRs are calculated. Pixels of moving objects are excluded when calculating the PSNR since we are measuring the qualities of the reconstructed backgrounds. The qualities of generated sprites are measured by computing the averaging PSNR of the reconstructed backgrounds. The generated sprites are shown in Fig. 10. Figure 10(a) is generated by the averaging blending strategy employed in the MPEG-4 VM without segmentation masks. The result using the averaging blending strategy with manually segmented masks is shown in Fig. 10(b). Figure 10(c) shows the sprite generated using the reliability-based blending strategy based on the rough segmentation masks extracted via the method developed by Lu *et al.*[6] Finally, the sprite generated using the proposed intelligent blender is

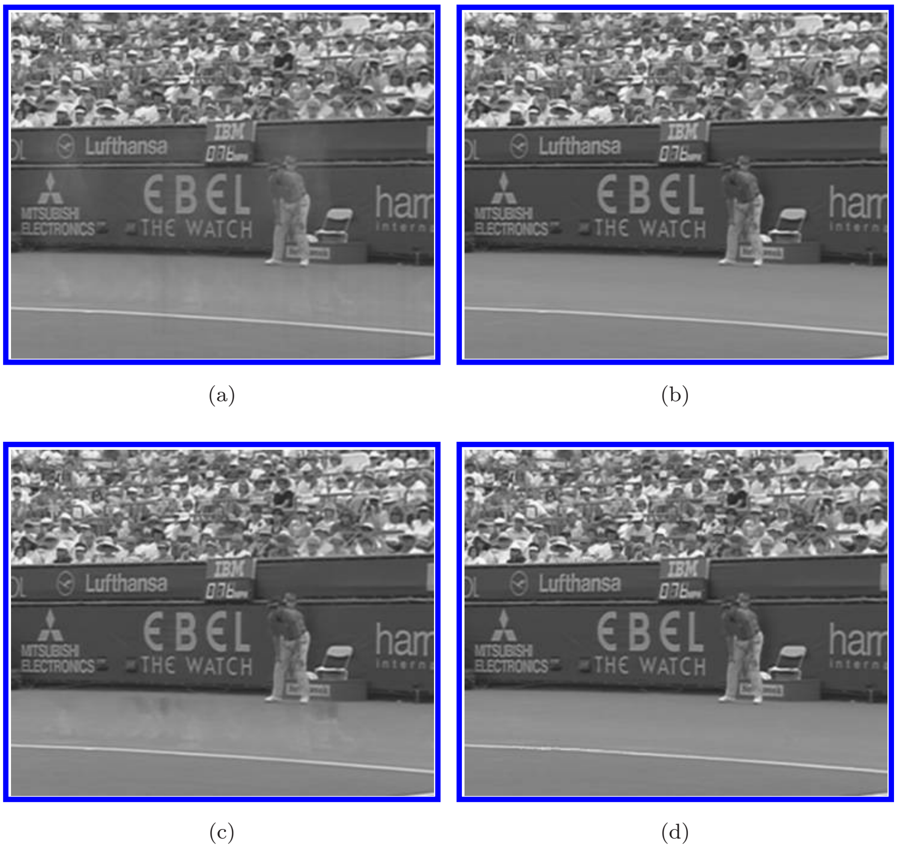


Fig. 11. The reconstructed backgrounds. (a) Averaging without segmentation masks. (b) Averaging with manually segmented masks. (c) Reliability-based blending. (d) Intelligent blending.

shown in Fig. 10(d). Figure 11 shows one of the reconstructed backgrounds using three different strategies respectively.

Since all moving objects are blended, the sprite generated by averaging blending will have shadows in several places, which are obvious in a reconstructed frame shown in Fig. 11(a). However, if perfect manual masks are provided, the shadows are eliminated completely and the averaging blending can achieve excellent results, as shown in Fig. 11(b). Most shadows can be removed using the reliability-based blending except some ill-segmented parts shown in the half-bottom of Fig. 11(c). The reconstructed background using the proposed intelligent blending is shown in Fig. 11(d). We can see that the sprite generated by our method is perceptually the same as the result using the average blending with perfect manual masks provided.

A quantitative comparison in PSNR is performed and illustrated in Fig. 12. The results of the average blending with and without manually segmented masks are plotted in dash-dotted and dotted lines, respectively. The result without masks is degraded by the shadows of moving objects and the frame borders, and has low

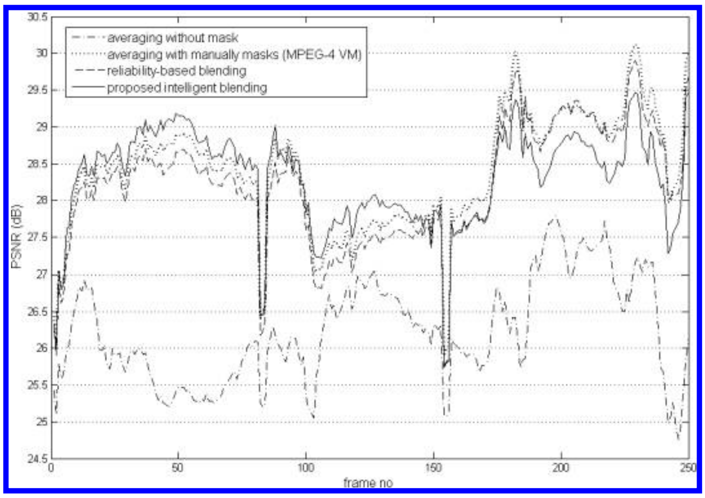


Fig. 12. PSNR comparison of different blending strategies.

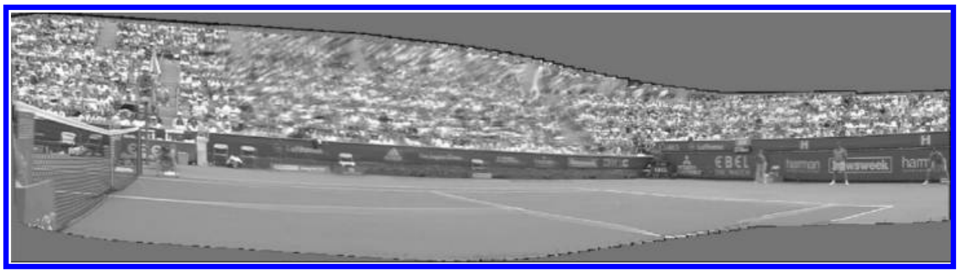


Fig. 13. The generated sprite of Lu *et al.*'s work.

average PSNR of 26.23 dB. With manually segmented masks, the averaging blending shows superior results not only in the visual quality but also in the measured PSNR. The average PSNR is 28.38 dB and is the best result in our tests.

The reliability-based and intelligent blending strategies are plotted in dashed, and normal lines, respectively. The reliability-based blending has average PSNRs 28.20 dB. The proposed intelligent blending has average PSNRs 28.29 dB, which is slightly higher than that of reliability-based blending and close to that of the average blending with perfect masks. These experiments show that the proposed method can generate high visual quality sprite without needing any segmentation mask.

The result of Lu *et al.*'s sprite generator,[8] which is shown in Fig. 13, is also quoted as a comparison. In contrast to the result of the proposed generator shown in Fig. 10(d), the proposed generator can generate better results. The average PSNR of Lu's generator is 23.1 dB, which is much lower than that of the proposed generator (28.29 dB). To make the comparison clearer, we take close views of three parts from the generated sprites in Fig. 13 and show them in Fig. 14. From Part 1 shown in Figs. 14(a) and 14(b), we can see that the generated sprite using[8] skews seriously.

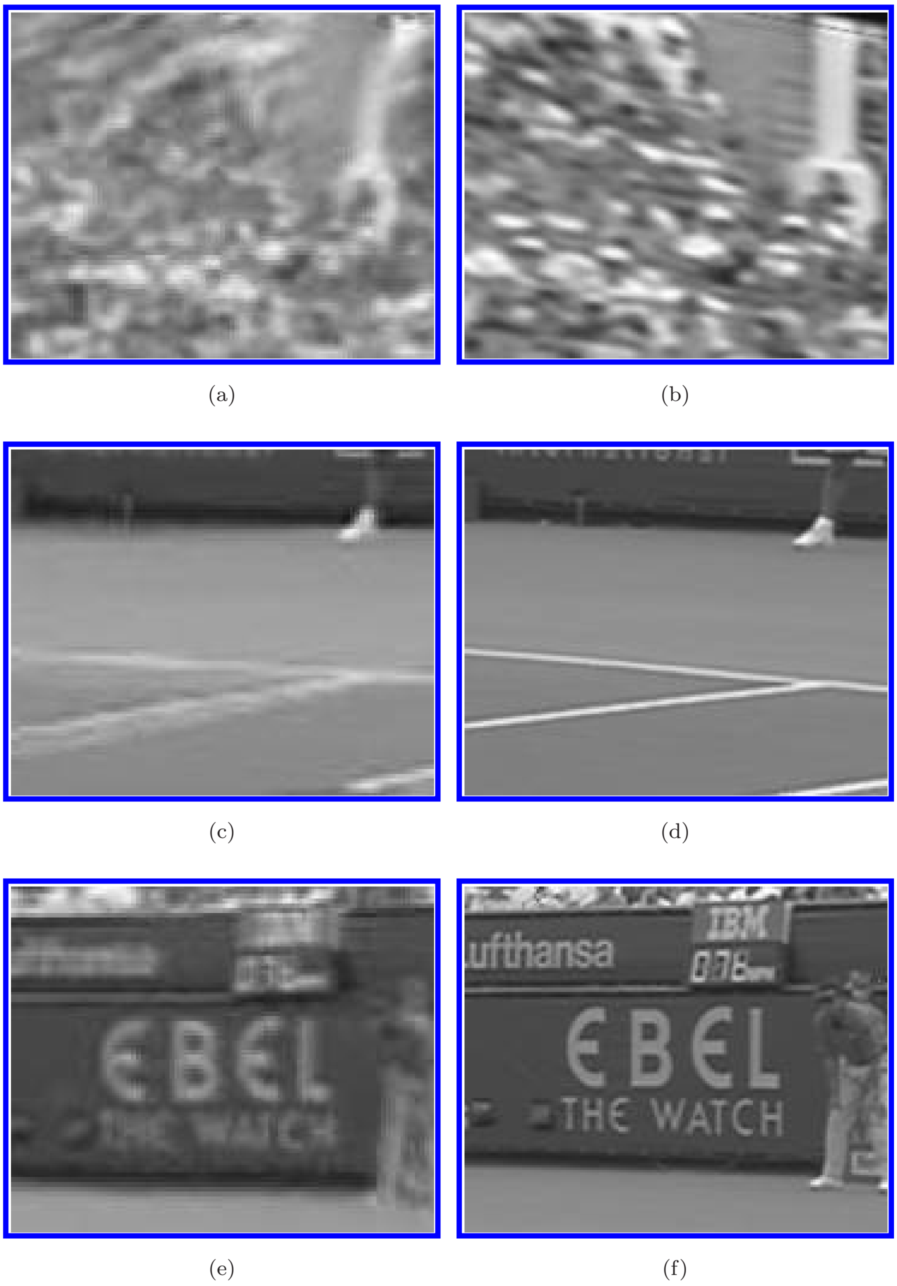


Fig. 14. Close views of the generated sprites. (a) Part 1 of Lu *et al.*'s work.[8] (b) Part 1 of the proposed method. (c) Part 2 of Lu *et al.*'s work.[8] (d) Part 2 of the proposed method. (e) Part 3 of Lu *et al.*'s work.[8] (f) Part 3 of the proposed method.

Figures 14(c) and 14(d) show Part 2, the white lines in Fig. 14(d) are registered well, but in Fig. 14(c) are not. The above two faults are due to the inaccuracy of the estimated GMP, and do not exist in the result of our generator. Part 3 shown in Figs. 14(e) and Fig. 14(f) also demonstrates that our method is superior to Lu *et al.*'s.

4. Conclusions

An effective sprite generator without segmentation masks is proposed. The method is based on MPEG-4's framework. A uniform feature point extraction with object removing is introduced to increase the precision of estimated global motion parameters. In order to remove the demand of segmentation masks, an intelligent blending strategy is proposed. It counts the occurrence of pixels and chooses the pixels with higher count to be blended into the sprite. The ghostlike shadows in the sprite generated by conventional reliability-based blending are eliminated, and the average PSNR is increased slightly.

Acknowledgments

The authors would like to thank the anonymous reviewers for their many valuable suggestions which have greatly improved the presentation of the paper.

This work is supported in part by National Science Council and Taiwan Information Security Center at NCTU.

References

1. A. Azarbayejani, T. Starner, B. Horowitz and A. Pentland, Visually controlled graphics, *IEEE Trans. Patt. Anal. Mach. Intell* **15** (1993) 602–605.
2. F. Dufaux and F. Moschieni, Background mosaicing for low bit rate coding, *Proc. IEEE Int. Conf. Image Processing* **1** (September 1996), pp. 673–676.
3. M. Irani and P. Anandan, Video indexing based on mosaic representations, *Proc. IEEE* **16**(5) (1998) 905–921.
4. ISO/IEC MPEG Video Group, MPEG-4 video international standard with amd. 1, ISO/IEC 14496-2 (July 2000).
5. ISO/IEC MPEG Video Group, MPEG-4 video verification model version 18.0, N3908 (January 2001).
6. Y. Lu, W. Gao and F. Wu, Fast and robust sprite generation for MPEG-4 video coding, in *Proc. IEEE Pacific-Rim Conf. Multimedia*, Beijing, China (October 2001), pp. 118–125.
7. Y. Lu, W. Gao and F. Wu, Sprite generation for frame-based video coding, *in Proc. IEEE Int. Conf. Image Processing*, Vol. 3, Thessaloniki, Greece (October 2001), pp. 473–476.
8. Y. Lu, W. Gao and F. Wu, Efficient background video coding with static sprite generation and arbitrary-shape spatial prediction techniques, *IEEE Trans. Circuits Syst. Vid. Technol.* **13**(5) (2003) 394–405.
9. T. Meier and K. N. Ngan, Video segmentation for content-based coding, *IEEE Trans. Circuits Syst. Vid. Technol.* **9**(8) 1999 1190–1203.

10. J.J. Moré, The Levenberg–Marquardt algorithm: implementation and theory, *Numerical Analysis*, ed. G. A. Watson, Lecture Notes in Mathematics, Vol. 630 (Springer Verlag, 1977), pp. 105–116.
11. A. Smolić, T. Sikora and J.-R. Ohm, Long-term global motion estimation and its application for sprite coding, content description, and segmentation, *IEEE Trans. Circuits Syst. Vid. Technol.* **9** (1999) 1227–1242.
12. R. Szeliski, Video mosaics for virtual environments, *IEEE Comput. Graph. Appl.* **16**(2) (1996) 22–30.
13. R. Szeliski and H. Y. Shum, Creating full view panoramic mosaics and environment maps, *Proc. ACM Conf. SIGGRAPH 97* (August 1997), pp. 251–258.
14. H. Watanabe and K. Jinzenji, Sprite coding in object-based video coding standard: MPEG-4, in *World Multiconf. Systemics, Cybernetics and Informatics (SCI) 2001*, *Proc.* **IV** (July 2001), pp. 420–425.

I-Sheng Kuo received the B.S. and M.S. degrees in electrical engineering from National Sun Yat-sen University, Kaohsiung, Taiwan, in 1997 and National Cheng Kung University, Tainan, Taiwan, in 1999, respectively. He is currently a Ph.D. student in the Department of Computer and Information Science at National Chiao Tung University, Hsinchu, Taiwan.

His research interests include image processing, video/image compression, MPEG-4 video and H.264.

Ling-Hwei Chen received the B.S. degree in mathematics and the M.S. degree in applied mathematics from National Tsing Hua University, Hsinchu, Taiwan in 1975 and 1977, respectively, and the Ph.D. in computer engineering from National Chiao Tung University, Hsinchu, Taiwan in 1987.

From August 1977 to April 1979, she worked as a research assistant in the Chung-Shan Institute of Science and Technology, Taoyan, Taiwan, and from May 1979 to February 1981, she worked as a research associate in the Electronic Research and Service Organization, Industry Technology Research Institute, Hsinchu, Taiwan. From March 1981 to August 1983, she worked as an engineer in the Institute of Information Industry, Taipei, Taiwan. She is now a Professor in the Department of Computer and Information Science at the National Chiao Tung University.

Her current research interests include image processing, pattern recognition, video/image compression and multimedia steganography.

This article has been cited by:

1. I-SHENG KUO, LING-HWEI CHEN. 2009. A FAST MULTISPRITE GENERATOR WITH NEAR-OPTIMUM CODING BIT-RATE. *International Journal of Pattern Recognition and Artificial Intelligence* **23**:02, 331-353. [Abstract] [References] [PDF] [PDF Plus]