High Visual Quality Sprite Generator using Automatic Segmentation and Intelligent Blending

I-Sheng Kuo* and Ling-Hwei Chen**

Department of Computer and Information Science, National Chiao Tung University,
1001 Ta Hsueh Rd., Hsinchu, Taiwan

ABSTRACT

This paper presents a sprite generator using automatic segmentation and intelligent blending. Unlike the sprite generator introduced in MPEG-4, which requires manually segmentation masks provided by user, the proposed sprite generator segments a video frame automatically into the background and moving objects called masks first. The segmentation masks increase the fidelity of the generated sprite. The automatically segmented masks are then used to generate sprite. The experimental result shows that the generated sprite using the automatically segmented masks close to the generated sprite using manually segmented masks. An intelligent blending method is also proposed to remove ghostlike shadows in the generated sprite caused by imperfect segmentation. The intelligent blending removes the shadows effectively without PSNR degradation, and the fidelity of the generated sprite is raised.

Keywords: sprite generation, sprite coding, background mosaic, global motion estimation, MPEG-4

1. INTRODUCTION

A sprite 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[1] 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. Many sprite generation algorithms[2-6] have been proposed, most of them contain two parts: global motion estimator (GME) and frame blender. First, the global motion estimator uses a geometric model to estimate the spatial location variation of two frames caused by camera motions. After the effects of camera motions are eliminated, a blender is then provided to mix all frames into a single image, that is, a sprite.

The aim of global motion estimator is to obtain an accurate estimation of camera motions. The camera motions can be represented by parameters of some geometric models often denoted as global motion parameters (GMP). Gradient descent based algorithms are widely used in the estimator. These algorithms usually need a good initial guess to avoid the solution being local optimum not global one. In this paper, we will provide an efficient method to get a good initial guess.

Before the frames are mixed into a single image, they must be in the same camera position. The estimated global motion parameters which represent the camera motions between two frames A and B are used to register the two frames. Frame B is first geometrically transformed (also called warped) via estimated global motion parameters into a frame B' with the same camera view as that of frame A. Then the transformed frame B' and frame A are mixed by a blending algorithm, e.g., averaging.

The motion of background comes from camera motion, like zooming, panning, and rotation. To eliminate the effect of camera motion, the motion must be modeled and estimated. A good camera model describes the camera motion accurately with low computational complexity; it is usually modeled by a geometric transformation with a set of

* sasami@debut.cis.nctu.edu.tw
** lhchen@cc.nctu.edu.tw

Visual Communications and Image Processing 2005, edited by Shipeng Li,
Fernando Pereira, Heung-Yeung Shum, Andrew G. Tescher, Proc. of SPIE Vol. 5960
(SPIE, Bellingham, WA, 2005) · 0277-786X/05/$15 · doi: 10.1117/12.633436

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 04/26/2014 Terms of Use: http://spiedl.org/terms

parameters. The affine transformation and perspective transformation are usually selected as the camera models. Both of them are defined by two equations with a set of parameters and as follows:

$$\text{Affine:}\quad x' = m_1 x + m_2 y + m_3\,,\quad y' = m_4 x + m_5 y + m_6\,, \tag{1}$$

$$\text{Perspective:}\quad x' = \frac{m_1 x + m_2 y + m_3}{m_7 x + m_8 y + 1}\,,\quad y' = \frac{m_4 x + m_5 y + m_6}{m_7 x + m_8 y + 1}\,, \tag{2}$$

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 which describes the camera motion. We can see that the affine transformation with six parameters is a special case of the perspective transformation with $m_7=m_8=1$. The perspective transformation can describe more complicated camera motions. However, the higher computational complexity of the perspective transformation limits its usability. The affine transformation is quite simple, but it has a lower accuracy. Both of them are adopted as standard tools in MPEG-4[1]. In this paper, the perspective transformation is selected since the sprite is usually generated first before coding the video sequence itself and real-time processing is not required.

MPEG-4 introduces an offline sprite generator[7] with three major parts: global motion estimation (GME), warping, and blending. Fig. 1 shows a flowchart of the sprite generator proposed by MPEG-4. The global motion estimation estimates the camera motion between the current frame and the sprite built so far and finds a set of parameters, P, of the used camera model which is named as global motion parameters (GMP). Then the current frame is warped toward the sprite by the transformation of the camera model, i.e., $I'(x,y)=I(x,y)$ where I and I' are the original frame and the warped frame respectively. (x,y) are one pixel's coordinates in the original frame and the corresponding warped coordinates are defined as $(x',y')=T_P(x,y)$, where T_P is the transformation with parameters P. After warping the frame, the sprite is updated by blending the warped frame into the previous sprite. The sprite is blended by simple averaging in MPEG-4's system.

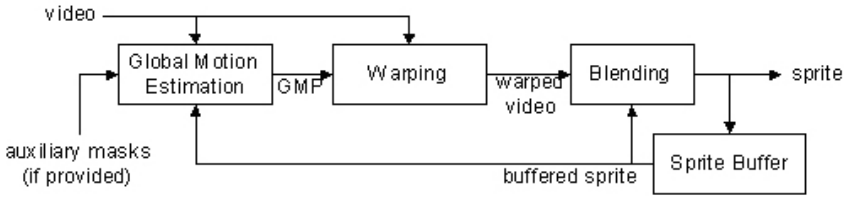


Figure 1: The sprite generator proposed by MPEG-4.

The video sequence is inputted frame by frame into the sprite generator. The sprite buffer holds the sprite generated so far and provides the current sprite to the GME process as the reference image.

As Fig. 1 shows, the MPEG-4's sprite generator requires an auxiliary segmentation masks. The segmentation masks are binary images which describe whether a pixel of its corresponding frame belongs to the foreground or not. These masks are created manually. The auxiliary masks can be used in global motion estimation to raise the estimation accuracy, furthermore, these masks can also be used to avoid moving object attending average blending. However, to generate the masks manually is impractical. Thus, developing a method to automatically segment moving objects is necessary.

MPEG-4 uses a simple averaging as the blending strategy. The averaging strategy derives good quality only if the background is segmented perfectly. The generated sprite will be blurred if some foreground objects are misclassified as backgrounds. To overcome the above-mentioned problems, in this paper, a two pass sprite generator is proposed. In the first pass, a coarse sprite is generated without using masks. In the second pass, based on the generated coarse sprite, an automatic segmentation method is provided to create moving object masks first. And then an intelligent blending strategy is presented. Based on the segmented masks and the intelligent blending strategy, a high visual quality sprite is generated finally.
The reminder of the paper is structured as follows. Section 2 describes the first pass of the proposed sprite generator. The second pass of the proposed generator is described in Section 3. Section 4 shows the experimental results and some comparisons with existing methods. Conclusions are made in Section 5.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 04/26/2014 Terms of Use: http://spiedl.org/terms

2. THE FIRST-PASS OF SPRITE GENERATION

The proposed sprite generator is based on the MPEG-4's work shown in Fig. 1. Fig. 2 shows the flowchart of the proposed generator. The generation process contains two passes. The first pass generates a coarse sprite by a global motion estimation algorithm, the bilinear interpolation warping[1], and the averaging blending strategy[7]. The second pass takes the generated coarse sprite and the original frame as inputs to generate a segmentation mask, and generates the final sprite with the generated segmentation mask. A new blending strategy noted as 'counting-averaging' replaces the averaging blending in the second pass to improve the visual quality of generated sprite.

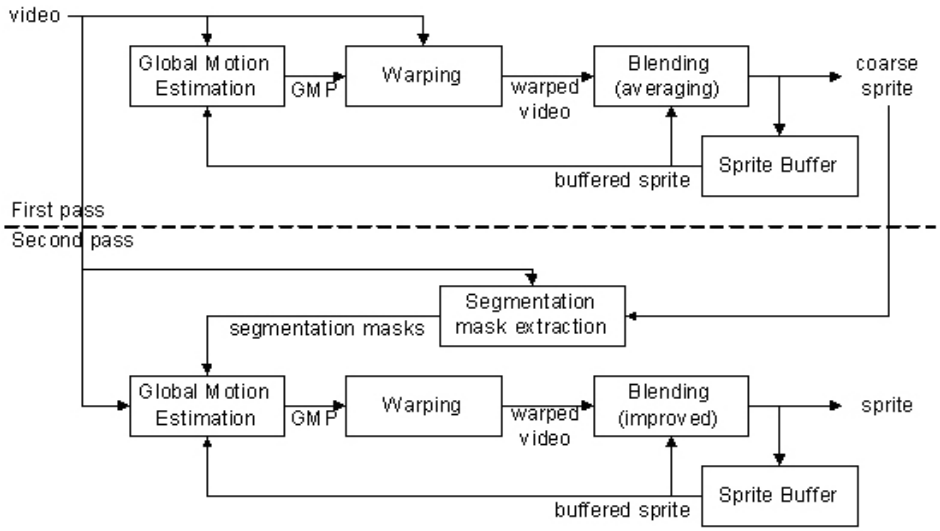


Figure 2: The proposed sprite generator.

The details of the proposed generator are described as follows.

2.1. Global motion estimation

The GME aims at finding a set of GMP between the current frame and the reference image, i.e., the current sprite. The global motion parameters determine the corresponding position in the reference image for every pixel in the current frame. The GME can be described by a minimization equation:

$$P^* = \arg\min_P E(I, I', T_P) \tag{3}$$

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:

$$E(I, I', T_P) = \sum_{x,y \in I} \left(I(x,y) - I'(T_P(x,y)) \right)^2 \tag{4}$$

To solve the minimization problem, the Levenberg-Marquardt algorithm[8] (LM-algorithm) based on the gradient descent method is used. Since the gradient descent method has a risk of being trapped into a local minimum, to get ride of the risk, a good starting point called an 'initial guess' should be provided. The GME is divided into two stages. Fig. 3 illustrates the block diagram of the proposed two stage GME schema. The first stage makes a good initial guess and estimates the motion parameters called the 'local parameters' between the current frame and the previous frame by the

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 04/26/2014 Terms of Use: http://spiedl.org/terms

LM-algorithm. The estimated local parameter is combined with the GMP of the previous frame as the initial guess of the second stage and the global parameter is estimated with the LM-algorithm.

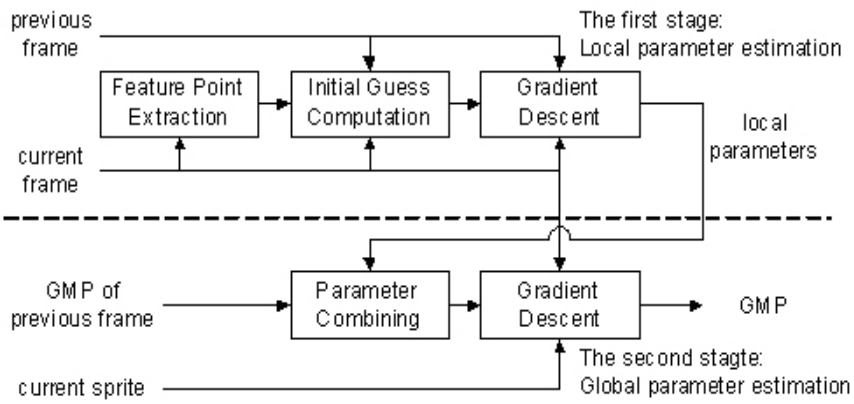


Figure 3: The proposed two-stage GME method.

To reduce the time complexity of the LM-algorithm, only some selected feature points in the current frame are employed while computing the registration error[6]. In order to avoid the aperture problem[9], those pixels with intensities being local maximum and local minimum are considered as feature points[6]. The feature points can be located by finding the pixels with larger Hessian values[10]. The Hessian value of a pixel (x,y) with intensity $I(x,y)$ is defined as

$$\left(\frac{d^2 I(x,y)}{dx^2} \cdot \frac{d^2 I(x,y)}{dy^2} - \left(\frac{d^2 I(x,y)}{dxdy} \right)^2 \right). \tag{5}$$

An image and its Hessian value are showed in Fig. 4. We can see that the pixel with large Hessian values (the white points in Fig. 4(b)) centralized in the high frequency part of the image. This will degrade the estimated GMP. The feature points should spread evenly over the entire image and avoids the homogenous regions.

For a $M \times N$ image, the image is divided into 256 non-overlapping regions, each of which has size $(M/16)\times(N/16)$. The intensity variance of each region is computed. Regions with variances larger than a pre-given threshold T_V are considered as non-homogenous regions. Suppose that we have K non-homogenous regions. For each non-homogenous region, H/K pixels with largest Hessian values in the region are considered as feature points. H=2000 is taken in the paper.

(a) (b)

Figure 4: An image and its Hessian value. (a) The image. (b) The Hessian value.

By observing Figure 4(b), we can find that not only the pixels in background have larger Hessian values but also the pixels in foreground objects. The precision of the global motion estimation will be affected if the pixels of the foreground objects are selected as feature points. The motion vectors of foreground objects usually differ from the motion vector of the background, which can be noted as the global translation (GT). The quality of the motion vectors of the objects will be degraded if we make a smaller search window around the GT.

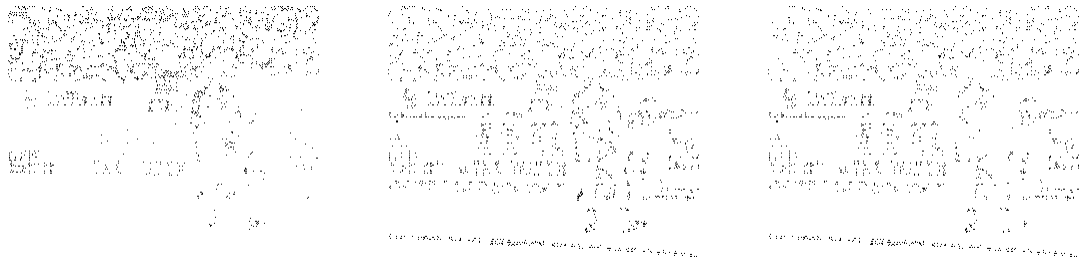
Downloaded From: http://proceedings.spiedigitallibrary.org/ on 04/26/2014 Terms of Use: http://spiedl.org/terms

We select 100 feature points which have the largest Hessian value and find the translations of them by full search within a 64×64 search window. Then the GT can be found by finding the most frequent translation. Then the motion vectors of all feature points are found by a full search within a smaller search window 17×17 centralized at the GT. Both the GT and the motion vectors of objects are estimated with a 17×17 block centralized at the feature point. 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.,

$$\frac{1}{N_B}\sum_{x,y\in B}\left(I(x,y)-I'(x+dx,y+dy)\right)^2 > T_O \tag{6}$$

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, they may not have a good motion vector around the global translation (a roughly approximation of the background motion), and will be considered as outliers.

Fig. 5 shows the results of extracting 2000 feature points from the original frame shown in Fig. 4(a) using different methods. Fig.5(a) shows the result of the traditional method which selects the pixels with the largest H Hessian values as features points. We can see that no feature points exist at the lower-left part of the image. The feature points selected by the proposed method are shown in Fig. 5(b). The figure shows that not only the pixels in the dedans but also the pixels on the white line in the lower part of the image are selected. The feature points selected by the proposed method spread uniformly. Fig. 5(c) shows the effect of outlier removing. We can see that many of the feature points inside the man are removed by testing Eq. (6).



(a) (b) (c)

Figure 5: The extracted feature points by different methods. (a) The result of taking H pixels with the largest Hessian values. (b) The result of the proposed method. (c) The result of the proposed method with outlier removing.

Each feature point in the current frame and its corresponding point 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 as the motion-estimated point of the feature point, i.e., if the location of the feature point is (*x*,*y*), the location of the corresponding point in the previous frame is (*x*+*dx*,*y*+*dy*), where (*dx*,*dy*) is the motion vector.

As mentioned previously, the perspective transformation expressed in Eq. (2) has eight parameters $m_1 \ldots m_8$. By substituting the location of feature point (*x*,*y*) and the location of the corresponding point (*x*',*y*') in Eq. (2) respectively, two equations are built. Since the perspective transformation has eight parameters, theoretically, four feature point pairs are sufficient to solve the parameters. In practical, 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[10].

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 LM-algorithm is employed here. The error

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 04/26/2014 Terms of Use: http://spiedl.org/terms

function required in the gradient descent method, is in slightly different from Eq. (4). Only the errors of the feature points are counted in the error function, that is,

$$E(I, I', T_P) = \sum_{(x,y) \in feature\_points} \left(I(x,y) - I'(T_P(x,y)) \right)^2 \tag{7}$$

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. The reference image is defined as the current sprite in the case of estimating the global parameters.

2.2. Warping and blending

The current frame is warped toward the sprite using the same method in MPEG-4's system. Since the transformed coordinates are not integers, bilinear interpolation is applied while generating the warped frame. The coarse sprite is built by blending the warped current frame into the current sprite. The simple averaging strategy is employed in the blending process. Let X, S_C and S_U be the intensities of the current frame, the current sprite and the updated sprite respectively. The averaging strategy can be expressed as:

$$S_U = \frac{N_C * S_C + X}{N_C + 1} \tag{8}$$

where N_C is the number of pixels blended in the current sprite.

3. THE SECOND-PASS OF SPRITE GENERATION

The blending strategy employed in the second pass requires segmentation masks to distinguish objects from backgrounds. Good segmentation masks separate the objects and backgrounds accurately. Furthermore, using good segmentation masks in GME, we can avoid selecting object points as feature points, thus the accuracy of estimated global motions will be increased. The blending error caused by wrongly blending the objects can be reduced. In the MPEG-4's system[7], the segmentation masks are given manually. However, generating the segmentation masks for every frame manually is time consuming and impractical. Developing an automatic generation method is necessary for a real system.

In the existing blending methods, objects will be blended into the sprite if segmentation masks are not provided. Objects are defined as moving things. An object which does not move in the whole scene is considered as a part of the background. Since the sprite is blended by averaging the gray values of pixels of the same location in the real world, those pixels, which some objects ever pass through, must get blurred in the blended sprite. This phenomenon provides a good starting point for automatic segmentation mask generation.

3.1. Frame segmentation

Fig. 6(a) and Fig. 6(b) shows the original frame and its reconstructed frame, which is built from the generated sprite in the first pass of the proposed method. We can see that the player in Fig. 6(a) leaves some shadows along his moving path. The shadows are caused by blending the player into the sprite incorrectly. Despite of the shadow areas, the reconstructed frames still carry most of background information.

By subtracting the original frame by the reconstructed frame, we can get an image of the moving objects. In order to remove the effect of peak noise, the block difference is applied instead of the pixel difference. The pixel difference D is defined as the magnitude of the difference between the original frame I and the reconstructed background R, i.e., $D=|I-R|$. A threshold T_1 is set to find out the candidates of object pixels. Pixels with D value larger than T_1 are considered as candidates. For each candidate, a 5×5 block B centered on the candidate is taken. The block difference D_B, is defined as

$$D_B = \sum_{(i,j) \in B} D(i,j) \tag{9}$$

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 04/26/2014 Terms of Use: http://spiedl.org/terms

The candidate with block difference larger than a preset threshold T_2 is considered as an object pixel. The two-stage thresholding technique computes the block differences only for those pixels with higher possibility to be objects. It reduces the complexity of computing block difference for each pixel.

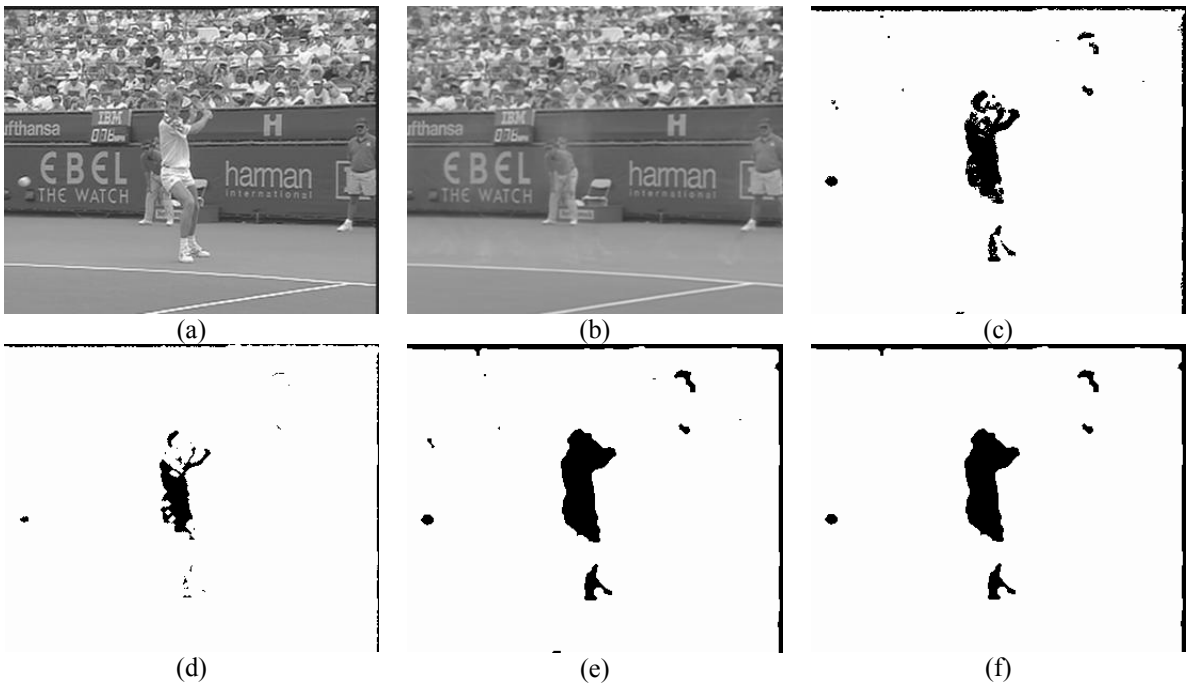


Figure 6: The generation of a segmentation mask. (a) The original image. (b) The reconstructed background. (c) The object pixels extracted by two-stage thresholding. (d) The seed image. (e) The base image. (f) The generated segmentation mask.

There are two problems while extracting the object pixels. First, the object regions are often ill-shaped with holes. Second, there are some small-sized regions which are mis-classified as objects. These problems can be solved using morphological processing and region selecting. Let O be a binary image representing the results of thresholding in the previous step. Pixels judged as objects will be set to one and others will be set to zero. Two binary images called seed and base images are generated. The seed image is produced by applying morphological erosion to O using a disk shaped structure element of radius 2, and the base image is produced by applying morphological dilation to O using the same shaped structure element of radius 5. The region selecting is applied on the base image. An object region is selected if any of its pixels have a value of one in the seed image. The segmentation mask is defined as the union of all regions selected.

Fig. 6(c)-Fig. 6(f) gives an example of generating a segmentation mask. Fig. 6(c) shows the candidates of object pixels generated by the two-stage thresholding. The seed and base images are shown in Fig. 6(d) and Fig. 6(e). The generated segmentation mask is produced by region selecting and shown in Fig. 6(f). The object pixels are colored black. Most part of the moving objects was extracted correctly, except two unclassified parts: the upper part of bat and the player's legs. The upper part of bat is nearly transparent hence the background is visible through the bat; the legs of the player have similar intensities to the background. Thus, both mis-classified parts do not affect the blending result. Moreover, the top and right borders are also classified as objects, this will eliminate the black line shadows in the generated sprite which will be discussed in the next subsection. Note that the tennis ball is also classified as an object.

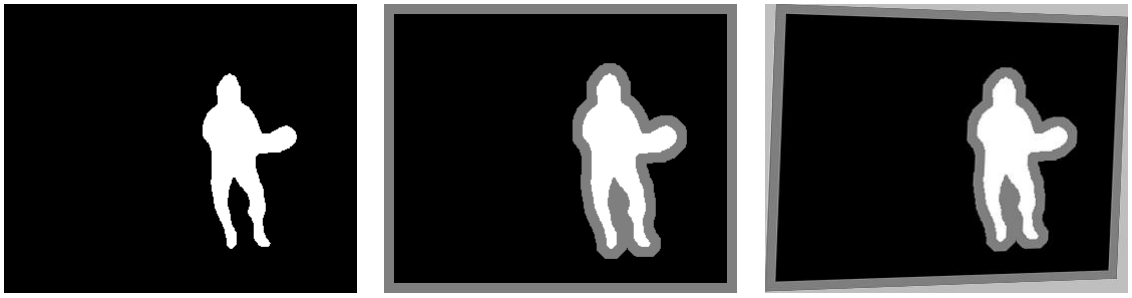
3.2. Global motion estimation in the second pass

The GME process in the second pass is similar to that described in the Subsection 2.1. The automatically generated segmentation masks are employed as a classification of object pixels. The feature points which are classified as object pixels are removed. The GMP of the second pass is estimated by the LM-algorithm. The accuracy of GMP is increased since the effect of object pixels is removed.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 04/26/2014 Terms of Use: http://spiedl.org/terms

3.3. Reliability-based blending

The traditional sprite generation methods produce the sprite by warping and averaging all frames of the video sequence. However, pixels belonging to the foreground objects will also be blended into the generated sprite; these will blur the generated sprite. A reliability-based blending method based on Smolic's work[6] is proposed to resolve this problem. A frame is split into reliable, unreliable and undefined regions by a reliability mask. Each region will be blended by different strategies. The proposed method divided the frame into 4 types: reliable, unreliable, object, and undefined. The reliable pixels are the pixels classified as background in the segmentation mask, excepting the pixels around the frame boundary which are classified as unreliable. The pixels classified as foreground objects in the segmentation mask are treat as object region. Since the warped frame and the sprite are not in rectangular shape, the pixels without holding any value are assigned to undefined. Fig. 7(a)-Fig. 7(c) shows a segmentation mask, the reliability mask extracted from the segmentation mask, and the warped reliability mask respectively. The black regions and the dark-grayed regions are the pixels classified as 'reliable' and 'unreliable' respectively. The white region is the pixels classified as 'object', and the undefined pixels are colored in light-gray.



(a) (b) (c)

Figure 7: The reliability mask establishment. (a) The segmented mask. (b) The reliability mask. (c) The warped reliability mask.

The reliability-based blending method blends the warped frame into the current sprite according to different blending strategies. Three blending strategies are proposed: average, replace, and discard. The average strategy blends the current frame into the current sprite by simple averaging which is the Eq. (8) specified in Subsection 2.2. The replace strategy simply replaces the value of the current sprite by the value of the current frame. The discard strategy discards the value of the current frame and the value in the current sprite keeps unchanged.

Table 1 lists the blending strategies applied in all combinations of reliability status of the current sprite and the current frame. If the types of one pixel in the current frame and the current sprite are the same, the average strategy is applied. If the pixel in the current frame is more reliable than that in the current sprite, the replace strategy is used. Otherwise the discard strategy is taken.

Table 1: Blending strategies.

current frame \ current sprite		pixel type			
		reliable	unreliable	object	undefined
pixel type	reliable	average	replace	replace	replace
	unreliable	discard	average	replace	replace
	object	discard	discard	average	replace

The blended sprite is the result of the sprite generation. If there still have unprocessed frames, the blended sprite is buffered and treated as the current sprite.

3.4. Improved blending

It is hard to get a prefect segmentation mask automatically. The pixels of foreground objects which are not correctly segmented still blend into the generated sprite, leaves ghostlike shadows. Fig. 11(a) and Fig. 11(b) show this quality degradation. There are several elliptic shadows caused by the bat crossing the middle part of the image, and two visible shadows caused by the shoes in the center of the image. To overcome the problem, we should avoid those pixels in the current frame with intensity very different from those of the corresponding pixels in the current sprite being blended. If

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 04/26/2014 Terms of Use: http://spiedl.org/terms

the average strategy is selected for a pixel according to Table 1, the intensity of pixel is further checked. The pixel is mixed into the sprite only if the following criteria met. The proposed criteria-added averaging strategy is noted as 'counting-averaging' strategy.

A candidate memory is used to store candidate background intensity for every pixel in the current sprite. For each pixel, two counters are used for the current sprite and the candidate. The counters hold the numbers of pixels that are used to compute the average intensities. The intensity difference between the current pixel and the current sprite is tested. If the intensity difference is smaller than a preset threshold, the current pixel is blended into the current sprite by averaging, and the counter of current sprite is increased by 1. If the test fails, the candidate counter is checked. The intensity of the current pixel is stored in the candidate memory directly if the counter is zero, this means that the current pixel is the first one with intensity very different from that of the current sprite. Otherwise another test is performed; it checks the difference between the current pixel intensity and the candidate memory. The current pixel will be blended into the candidate memory and the candidate counter is increased if the intensity difference is smaller than a preset threshold; else the candidate memory is replaced by the intensity of the current pixel

Based on the fact that the background appears more often than moving objects in any pixel, two counters are checked. If the candidate counter is larger than the sprite counter, the value stored in the candidate memory is believed to have a higher possibility to be the intensity of background. The sprite and sprite counter are replaced by the value of candidate memory and candidate counter respectively. The candidate memory is cleared and the candidate counter is set to zero. A pseudo code is listed in Table 2 to help understanding.

Table 2: The pseudo code of improved blending

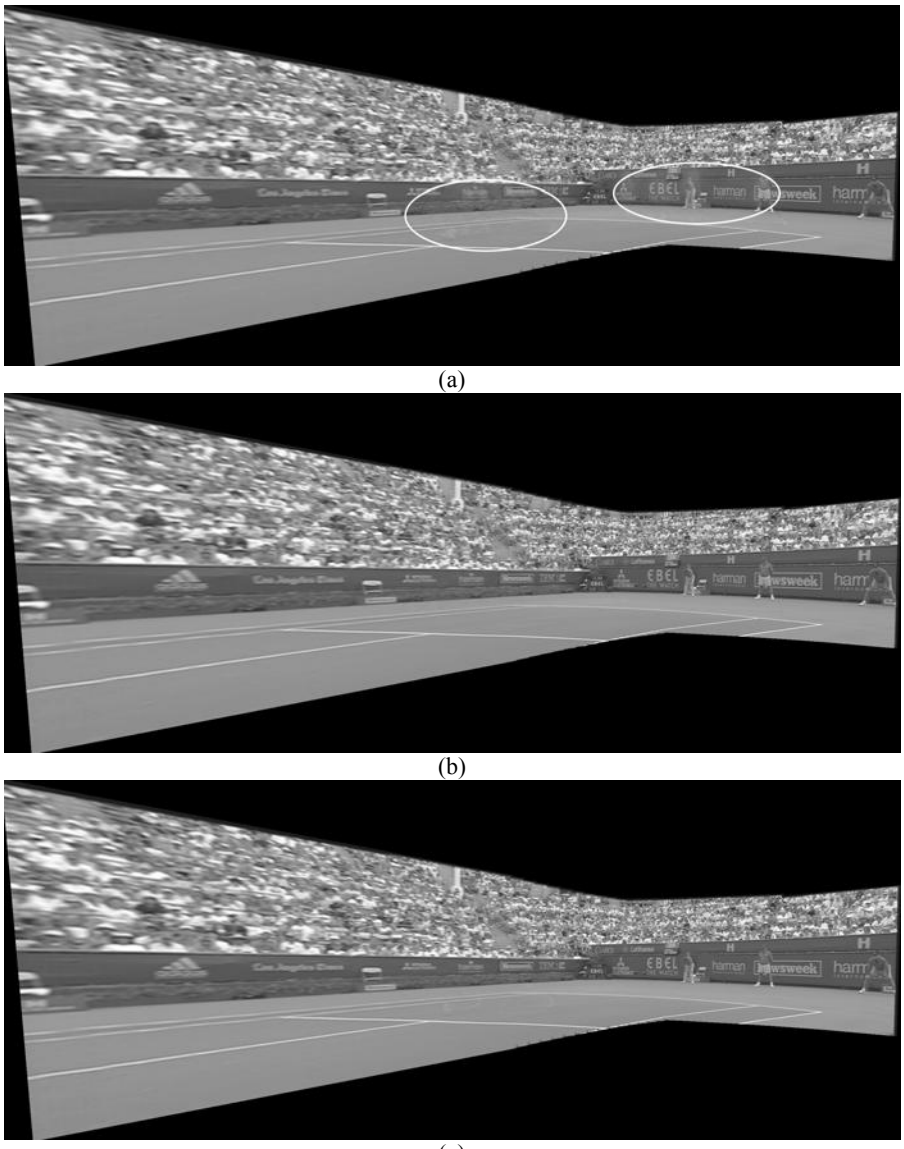
```
if abs(current_pixel-current_sprite)<T then
        blend the current pixel into the current sprite.
        sprite_counter is increased by 1.
else
        if cand_counter>0 and abs(current_pixel-candidate)<T then
                blend the current pixel into the candidate memory.
                cand_counter is increased by 1.
        else
                candidate=current_pixel
                cand_counter is increased by 1.
        end if
        if cand_counter>sprite_counter then
                replace the current sprite by the candidate.
                sprite_counter=cand_counter
                clear candidate and cand_counter.
        end if
end if
```

4. 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 of visual quality of the generated sprite is performed for completeness.

Fig. 8 shows the generated sprite of the video sequence 'stefan' by different methods. Fig.8(a) is generated without using segmentation masks, i.e., the first pass sprite generation proposed in Section 2. The masks used to generate Fig. 8(b) are obtained automatically by the proposed segmentation schema described in Section 3.1. Mmanually segmented masks also employed as an comparison and the results are shown in Fig. 8(c). Fig.9 shows one of the reconstructed frames using three different methods respectively. Like we stated before, the sprite generated without using masks contains shadows (see the circled parts in Figure 8(a)), caused by wrongly blending the player into sprite. These shadows are successfully removed in the sprite generated using the masks generated automatically by our method. The sprites generated with automatically or manually segmented masks are perceptually the same by human eyes.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 04/26/2014 Terms of Use: http://spiedl.org/terms



(a) (b) (c)

Figure 8: The sprite generated by: (a) one pass without using mask; (b) two pass using automatically generated masks; (c) two pass using manually generated masks.

(a) (b) (c)

Figure 9: The reconstructed frames: (a) one pass without using mask; (b) two pass using automatically generated masks; (c) two pass using manually generated masks.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 04/26/2014 Terms of Use: http://spiedl.org/terms

The PSNR of the reconstructed frames are illustrated in Fig. 11. The results of generation without masks, with proposed automatically segmented masks, and with manually segmented masks are shown in thick line, thin line, and dotted line respectively. As we expected, the sprite generated without masks performs worst. The segmentation masks improve the quality of the reconstructed frames about 0.5 to 1.0dB in PSNR. Using masks generated automatically by the proposed method perform closely to that using the manually segmented masks.

The performance using the proposed counting-averaging blending strategy is also tested. Fig. 10 shows the reconstructed backgrounds using different blending strategies. Fig. 10(a) is generated by the traditional averaging strategy and Fig. 10(b) is generated using the proposed counting-averaging strategy. Two contrast-enhanced version of Fig. 10(a) and Fig. 10(b) are shown in Fig. 10(c) and Fig. 10(d) to express the effect of shadow eliminating. It shows that the counting-averaging strategy eliminates the shadows perfectly. A comparison in PSNR is also illustrated in Fig. 12. The averaging strategy is plotted in solid lines and the counting-averaging strategy is plotted in dotted line. The comparison shows that the novel counting-averaging strategy improves the visual quality without decreasing the PSNR.

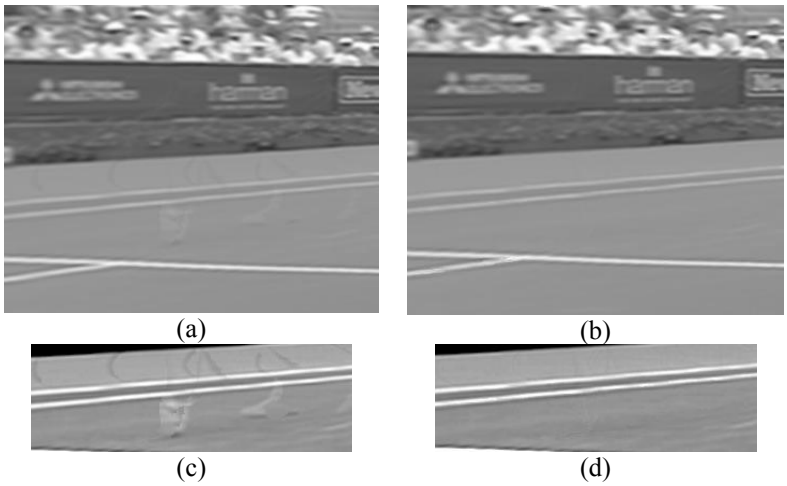


Figure 10: Reconstructed frames with different blending methods. (a) Reliability-based averaging. (b) Counting-averaging. (c) Part of (a), contrast enhanced. (d) Part of (b), contrast enhanced.

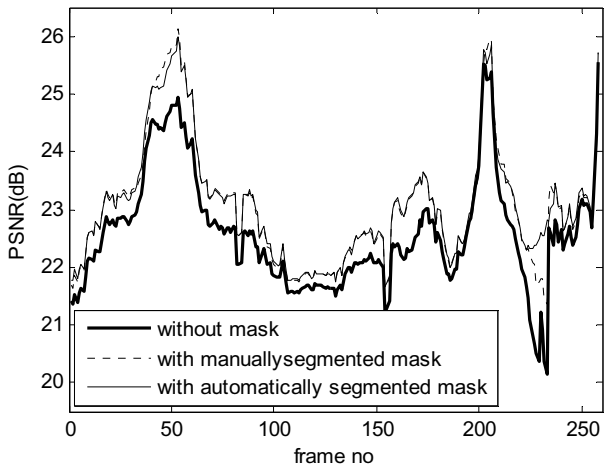


Figure 11: PSNR comparison using different segmentation masks.

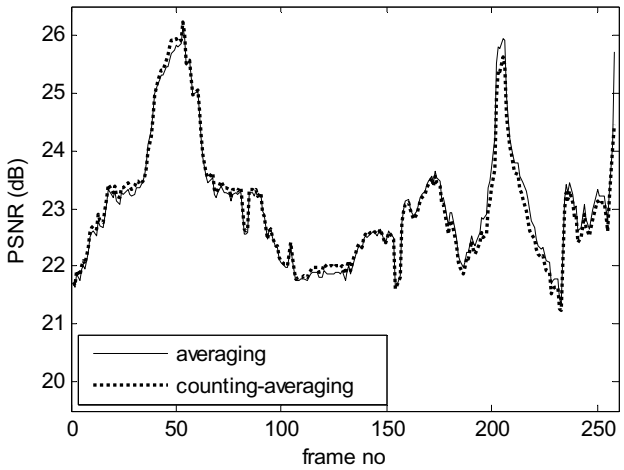


Figure 12: PSNR comparison of different blending strategies.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 04/26/2014 Terms of Use: http://spiedl.org/terms

5. CONCLUSIONS

An effective sprite generation method is proposed. The method includes an accurate GME schema, an object segmentation schema, and a novel frame blending strategy. The proposed GME schema applies a robust gradient descent approach with a new way of finding the initial guess. The novel blending strategy increases the visual quality of the reconstructed background by eliminating the ghostlike shadows effectively, without degrading the PSNR. The segmentation masks generated by the proposed segmentation schema can be employed into the GME and the blending step, these will increase the PSNR by 0.5 to 1.0dB.

ACKNOWLEDGEMENTS

The research was supported in part by the National Science Council under contract NSC-92-2213-E-009-101 and the MediaTek incorporation, Taiwan.

REFERENCES

1. ISO/IEC MPEG Video Group, *MPEG-4 video international standard with amd. 1*, ISO/IEC 14496-2, Jul 2000.
2. M. Irani, P. Anandan, J. Bergen, R. Kumar, and S. Hsu, "Efficient representations of video sequences and their applications," Signal Processing: Image Communication, special issue on Image and Video Semantics: Processing, Analysis, and Application, **vol. 8**, pp. 327–351, May 1996.
3. M. Irani and P. Anandan, "Video indexing based on mosaic representations," *Proc. IEEE*, vol. 86, issue 5, pp. 905–921, May 1998.
4. R. Szeliski, "Video mosaics for virtual environments," IEEE Computer Graphics and Applications, **vol. 16**, pp. 22–30, March 1996.
5. A. Smolić, and J. R. Ohm, "Robust global motion estimation using a simplified M-Estimator approach," *Proc. ICIP'2000, IEEE International Conference on Image Processing* , Vancouver, Canada, Sep. 2000.
6. A. Smolić, "Long-term global motion estimation and its application for sprite coding, content description, and segmentation," IEEE Trans. Circuits and system for video technology, **vol. 9**, pp. 1227-1242, Dec. 1999.
7. ISO/IEC MPEG Video Group, *MPEG-4 video verification model version 18.0*, N3908, Jan. 2001.
8. J.J. Moré, "The Levenberg-Marquardt algorithm: implementation and theory," Numerical Analysis, ed. G. A. Watson, Lecture Notes in Mathematics 630, Springer Verlag, pp. 105-116, 1977.
9. A. Azarbayejani, T. Starner, B. Horowitz, and A. Pentland, "Visually controlled graphics," IEEE Trans. Pattern Anal. Machine Intell., **vol. 15**, pp. 602–605, June 1993.
10. E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, LAPACK User's Guide, Third Edition, SIAM, Philadelphia, 1999.
11. M. Lee, W. Chen, C. B. Lin, C. Gu, T. Markoc, and R. Szeliski, "A layered video object coding system using sprite and affine motion model," IEEE Trans. on Circuits and System for Video Technology, **vol. 7**, pp. 130–145, Feb. 1997.
12. T. Sikora, "The MPEG-4 video standard verification model," IEEE Trans. on Circuits and System for Video Technology, **vol. 7**, pp. 19–31, Feb. 1997.
13. E. T. Kim and H. M. Kim, "Fast and robust parameter estimation method for global motion compensation in the video coder," IEEE Trans. on Consumer Electronics, **vol. 45**, no.1, Feb. 1999.
14. F. Dufaux and J. Konrad, "Efficient, robust, and fast global motion estimation for video coding," IEEE Trans. on Image Processing, **vol. 9**, no. 3, Mar 2000.
15. Y. Lu, W. Gao, and F. Wu, "Efficient background video coding with static sprite generation and arbitrary-shape spatial prediction techniques," IEEE Trans. Circuits and Systems for Video Technology, **vol. 13**, no. 5, May, 2003.
16. J. Y. A. Wang and E. H. Adelson, "Representing moving images with layers," IEEE Trans. on Image Processing, **vol. 3**, pp. 625-638, Sept. 1994.
17. H. G. Musmann, M. Hotter, and J. Ostermann, "Object-based analysis-synthesis coding of moving images," EUROSIP Signal Processing: Image Community, **vol. 1**, no. 2, pp. 117–138, Oct. 1989.

Downloaded From: http://proceedings.spiedigitallibrary.org/ on 04/26/2014 Terms of Use: http://spiedl.org/terms