VIDEO-OBJECT SEGMENTATION USING MULTI-SPRITE BACKGROUND SUBTRACTION

Dirk Farin[1]*, Peter H. N. de With*[2]*, and Wolfgang Effelsberg*[3]

[1] Univ. of Technol. Eindhoven
[2] LogicaCMG / Univ. of Technol. Eindhoven
5600 MB Eindhoven, Netherlands
d.s.farin@tue.nl

[3] Dept. of Computer Science IV
University of Mannheim
68131 Mannheim, Germany

ABSTRACT

The background subtraction algorithm is a frequently-used object segmentation technique because of its algorithmic simplicity. However, we show that for general rotational camera-motion, it is impractical or even impossible to use a single background image. As a solution, we propose to use multi-sprite backgrounds which enables processing of arbitrary rotational camera-motion. This paper describes a complete video-object segmentation system employing multi-sprites. The system generates object masks and background sprites that are compatible with the MPEG-4 object-oriented video-coding tools. Good segmentation results are also obtained for sequences, which cannot be processed with ordinary background images.

1. INTRODUCTION

The background subtraction algorithm is frequently used as a segmentation algorithm, since it provides good results at low algorithmic complexity. The algorithm detects foreground objects by subtracting the pure background image from the current video input [1]. In the simplest case, the background image is recorded separately, but it can also be artificially synthesized by observing a scene for a longer time [2, 3]. If the camera is allowed to rotate or zoom, camera-motion has to be compensated during construction of the scene-background image. The choice of an appropriate camera-motion model depends on the degrees of freedom of camera-motion. For arbitrary rotation and zoom, the 8-parameter perspective motion model defined by

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + 1}\ ,\ y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + 1} \quad (1)$$

is the mostly used model. Note that the affine motion model, which is often used as an approximation, cannot handle camera pan and tilt and is only practical for very small pan and tilt angles at large focal lengths.

Since the perspective motion model is a plane-to-plane mapping [4], warping one image plane onto another, the maximum field of view that can be represented in a single composed background image is limited to $180°$. Furthermore, the area of the projected background image increases quickly, such that only small rotation angles can be processed efficiently (see Fig. 1a). This limitation is a key problem to all algorithms that use the perspective motion model for background sprite generation.

An alternative representation for background images would be to use spherical or cylindrical image mosaics [5] for the background image, instead of a planar mapping. However, this approach has several disadvantages compared to using the perspective motion model, as is shown below.

- For the generation of spherical/cylindrical mosaics, it is required to know internal camera parameters like focal length and the principal point of the camera. Even though these parameters can be estimated from the parameters of the perspective motion model, the estimation is difficult, since the calculation is numerically sensitive.
- The estimation of the transformation parameters for cylindrical and spherical mosaics requires complicated non-linear optimization techniques. Moreover, these techniques are computationally expensive since transcendental functions are required.
- The obtained cylindrical/spherical background is not compliant with the MPEG-4 video coding standard, since MPEG-4 only supports the perspective transformation model.

To solve the above-mentioned problems with the perspective motion model and still avoid the problems of cylindrical/spherical mosaics, we propose to distribute the background scene content over a set of several independent background images, which is in the following called a *multi-sprite*. The input sequence is partitioned into segments and an independent sprite is generated for each segment such that the total area of all sprites is minimized. This not only results in lower memory requirements and faster execution time, but also enables to process scenes with arbitrary camera-motion (see Fig. 1b).

Our multi-sprite based segmentation system comprises four processing steps. First, camera-motion is estimated using a feature-based robust estimator (Sect. 2). Subsequently,

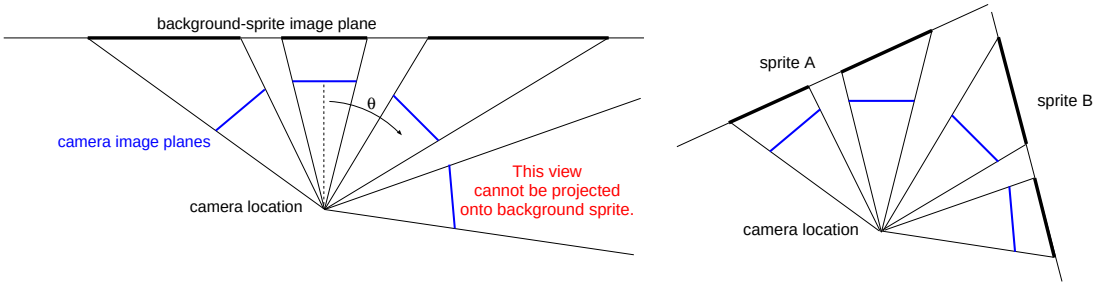


Fig. 1. (a) Top-view of projecting the input frames onto the background sprite plane with a horizontally rotating camera. The more the camera rotates away from the frontal view ($|\theta|$ increases), the larger the projection area on the sprite plane. For $|\theta| \geq 90^\circ$, the projection ray does not intersect the sprite plane. Hence, only 180° field of view can be covered with one sprite. (b) The use of several sprites reduces the required sprite size and allows to cover larger viewing angles.

the obtained motion parameters are used to find a multi-sprite partitioning of the scene background (Sect. 3). The background sprites are synthesized (Sect. 4), and finally, the object masks are obtained using background subtraction and a Markov-Random-Field regularization (Sect. 5).

2. CAMERA-MOTION ESTIMATION

Camera-motion is estimated using a combination of short-term and long-term prediction. A short-term prediction is computed for each pair of consecutive input frames using a feature-based motion estimator. This algorithm is robust against fast camera-motion, but it is not accurate enough for synthesizing a background sprite. Therefore, a second motion-estimation algorithm is applied subsequently in the background synthesis step to optimize the alignment between the input images and the background sprite (Sect. 4).

To estimate the transformation H_i between input frames i and $i+1$, we use the Harris detector to locate corners in both images. Correspondences between corners in the two images are established by evaluating the cross-correlation of a small window around each corner position. Each correspondence gives a reliable motion-vector, since corners can be tracked easily [6]. If all of the motion-vectors would be part of the background motion, the motion parameters of the camera model as specified in Eq. (1) could be determined using a least-squares solution of an overdetermined equation system. However, inevitably, some motion-vectors will be part of foreground objects. These foreground vectors have to be excluded from the computation of the camera-motion parameters. This is achieved with a *Least-Trimmed Squares* (LTS) regression algorithm [7], which sorts out a fixed percentage of outliers, such that the error of the fitted model to the remaining inlier data is minimized (see Fig. 3).

3. MULTI-SPRITE PARTITIONING

As described earlier, for arbitrary camera-motion, it is not feasible to construct a single background image for the complete scene. A possible solution would be to omit the construction of background sprites completely and instead synthesize the background for each input frame independently, by warping all input frames to the current frame's coordinate system. However, this approach would be very computational complex, since the camera parameter refinement and background image synthesis (see Sect. 4) would have to be computed for each input frame over the whole sequence. Furthermore, no sprites appropriate for MPEG-4 object-based video coding would be created.

The concept of multi-sprites combines the advantages of background sprites with the applicability to general camera motion, faster computation, and less memory requirements. The idea is to partition the input sequence into a set of segments and compute a background sprite for each of the segments. Specifically, we search for the partitioning that results in the minimum sum over the covered areas of the generated sprites. This results not only in minimum memory requirements, but also in minimum computation time. More precisely, let $P = \big((1, p_1 - 1), (p_1, p_2 - 1), (p_2, p_3 - 1), \ldots, (p_{n-1}, N)\big)$ be a partitioning of the video sequence of length N into n segments. Furthermore, let $S_{a;b}$ denote the image area required for the sprite synthesized from the frame range a to b. In fact, $S_{a;b}$ is defined slightly different to incorporate further constraints like enforcing that the background sprite resolution may not decrease during the construction process. For further details, see [8]. The optimal sequence partitioning P^* is defined as the one which minimizes the total sprite area:

$$P^* = \arg\min_P \sum_{(a,b)\in P} S_{a;b}. \tag{2}$$

The minimization problem can be computed efficiently with an iterative algorithm. For each image k, we compute the minimum cost c_k ($c_0 = 0$) of a partitioning ending in image k as

$$c_k = \min_{i\in[1,k]} \{c_{i-1} + S_{i;k}\}. \tag{3}$$

The index i denotes the beginning of the last segment in the partitioning. For each image, we store the i for which the minimum was reached. Tracing back these stored k-values,

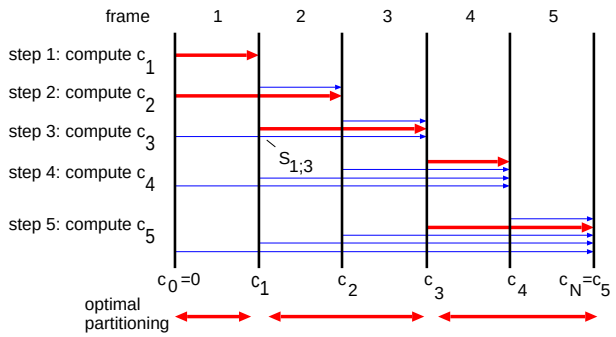


Fig. 2. Determining the best sequence partitioning.

starting at frame N, results in the optimal partitioning with respect to total sprite size. The computation process is illustrated in Fig. 2. Each state c_k is assigned the minimum coding cost up to frame k. The arrows represent the cost for the sprite built from the covered frames. The sprite that gives the minimum cost in each c_k is marked with a bold arrow. Tracing back the bold arrows from the last node (c_N) gives the optimal partitioning with minimum cost (in this case, three sprites will be built using frames 1–1;2–3;4–5).

4. BACKGROUND SYNTHESIS

After the input sequence has been partitioned into a set of segments $l \in \{1, \ldots, n\}$, a pure background sprite s_l is computed for each segment. Assume that sprite construction begins with frame r, then we obtain the geometric transformation T_i, mapping a frame i onto r, by a concatenation of the inter-frame transforms: $T_i = H_r \cdot H_{r+1} \cdots H_{i-1}$. However, using the camera parameters obtained in the short-term prediction step to compensate camera-motion, would lead poor background quality, since the concatenation of the transforms quickly leads to an accumulation of errors. Hence, the parameters of the transformations T_i are refined by registering each input frame f_i to the background mosaic image $\hat{s}_l$. The Levenberg-Marquardt gradient-descent algorithm is applied to minimize the inter-frame difference

$$\Delta = \sum_{x,y} \rho\big(\hat{f}_i(T_i(x,y)) - s_l(x,y)\big), \tag{4}$$

where ρ is an error limiting M-estimator function. The parameters from the short-term prediction give a good initialization, leading to robust and fast convergence. Since the reference image for registration is the current background sprite computed so far, transformation errors cannot accumulate. The input images are processed sequentially and new pixels are added to the sprite only in sprite areas that have not been covered yet. This background update policy is important to prevent error accumulation, but it does not remove foreground objects from the background sprite $\hat{s}_l$. Hence, in a further processing step, the sprite image is recomputed with a background estimation algorithm, using the refined geometric transformations. In order to obtain a background image without foreground objects, a temporal median filter is applied [3]:

$$s_l(x,y) = \text{median}_i\big\{f_i(T_i(x,y))\big\} \tag{5}$$

Alternatively, the background image can also be synthesized using the algorithm described in [2], which gives better results in complex scenes.

5. OBJECT SEGMENTATION

Foreground object masks are obtained by computing the change detection mask of the input frames w.r.t. the camera-motion compensated background sprite. We apply the algorithm proposed in [9], which classifies a pixel (x,y) as changed *c* or unchanged *u* based on the decision rule

$$\frac{p(d(x,y)|H_u)}{p(d(x,y)|H_c)} \gtrless_c^u \frac{\Pr(Q_c^{x,y})}{\Pr(Q_u^{x,y})}, \tag{6}$$

where $d(x,y) = f_i(x,y) - s_l(T_i^{-1}(x,y))$ denotes the difference between the input frame and the motion compensated background sprite, and H_u, H_c denote the unchanged or changed hypotheses. $\Pr(Q_u^{x,y}), \Pr(Q_c^{x,y})$ denote the *a-priori* probabilities that pixel (x,y) in change detection mask Q is unchanged or changed, respectively. The *a-priori* probabilities are modeled using a Gibbs/Markov random field $\Pr(Q) = Z^{-1}\exp\{-E(Q)\}$, where Z is the normalization constant and $E(Q)$ comprises the energy terms for second-order cliques in the 3×3 neighborhood around the pixel. For further details, see [9, 10].

6. RESULTS

As an example, we applied our segmentation algorithm on the well-known *Stefan* sequence. Since the camera makes a wide horizontal pan, it is impossible to combine the complete scene background in a single background image. Consequently, our algorithm partitioned the sequence into four segments (Fig. 4). The obtained segmentation masks are shown for every 50^{th} frame in Fig. 5. Comparable results are obtained with other sequences.

7. CONCLUSIONS

Using multi-sprite background images releases the restriction to small camera rotation angles, which is a consequence of using the perspective motion model for background sprite construction. In fact, multi-sprites allow general camera-motion, including variable sprite resolution because of camera zoom. Furthermore, multi-sprites reduce the computational complexity and memory requirements of the segmentation system. Finally, a multi-sprite partitioning improves the coding efficiency of background sprites [8], while the obtained sprites are still compliant with the MPEG-4 video coding tools.

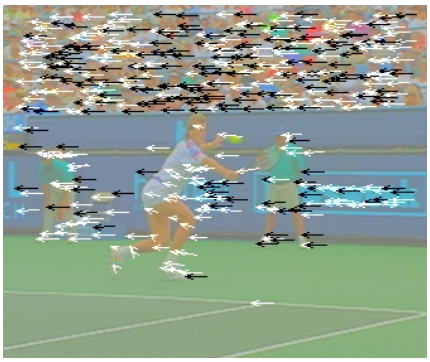


Fig. 3. Motion-vectors determined in the short-term prediction step. Inlier vectors that were used to compute the camera-motion parameters are drawn black.

8. REFERENCES

[1] R. J. Quan and M. I. Sezan, "Video background replacement without a blue screen," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 1999, vol. 4, pp. 143–146.

[2] Dirk Farin, Peter H. N. de With, and Wolfgang Effelsberg, "Robust background estimation for complex video sequences," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2003, vol. 1, pp. 145–148.

[3] M. Massey and W. Bender, "Salient stills: Process and practice," *IBM Systems Journal*, vol. 35, no. 3 & 4, pp. 557–573, 1996.

[4] Richard Hartley and Andrew Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2000.

[5] Richard Szeliski and Heung-Yeung Shum, "Creating full view panoramic image mosaics and environment maps," in *Proc. SIGGRAPH*, 1997, pp. 251–258.

[6] Jianbo Shi and Carlo Tomasi, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, June 1994, pp. 593–600.

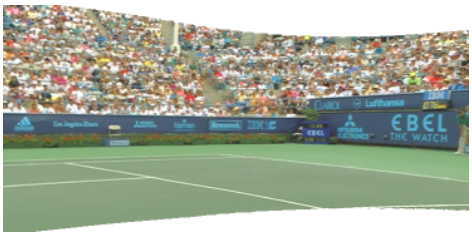
[7] P. J. Rousseeuw and K. Van Driessen, "Computing LTS regression for large data sets," *Institute of Mathematical Statistics Bulletin*, vol. 27, no. 6, 1998.

[8] Dirk Farin, Peter H. N. de With, and Wolfgang Effelsberg, "Minimizing MPEG-4 sprite coding-cost using multi-sprites," in *SPIE Proc. Visual Communications and Image Processing*, 2004.

[9] Til Aach and Andre Kaup, "Bayesian algorithms for adaptive change detection in image sequences using markov random fields," *Signal Processing: Image Communication*, vol. 7, pp. 147–160, 1995.

[10] Dirk Farin, Thomas Haenselmann, Stephan Kopf, Gerald Kühne, and Wolfgang Effelsberg, "Segmentation and classification of moving video objects," in *Handbook of Video Databases: Design and Applications*, Borko Furht and Oge Marques, Eds., pp. 561–592. CRC Press, Sept. 2003.

(a) sprite 1 (frames 1–242)



(b) sprite 2 (frames 243–255)

(c) sprite 3 (frames 256–293)

Fig. 4. Background sprites generated for the *Stefan* sequence. Sprite 4 (frames 294–300) is not shown.

(a) 1 (b) 50 (c) 100 (d) 150

(e) 200 (f) 250 (g) 300

Fig. 5. Object segmentation results for the *Stefan* sequence.