Multimed Tools Appl
DOI 10.1007/s11042-010-0469-3

Automatic MPEG-4 sprite coding—Comparison of integrated object segmentation algorithms

Alexander Glantz · Andreas Krutz · Thomas Sikora · Paulo Nunes · Fernando Pereira

Abstract Sprite coding, as standardized in MPEG-4 Visual, can result in superior performance compared to common hybrid video codecs. We consider sprite coding, which significantly increases the objective as well as the subjective quality of coded video content. The main challenge of this approach is the segmentation of the foreground objects in a preprocessing step. We evaluate automatic object segmentation methods based on global motion estimation and background sprite generation. The objects are coded using the MPEG-4 Visual Main Profile and compared with the Advanced Simple Profile.

Keywords MPEG-4 Visual · Global motion estimation · Background modeling · Object segmentation · Sprite coding

A. Glantz (✉) · A. Krutz · T. Sikora
Technische Universität Berlin, Communication Systems Group, Berlin, Germany
e-mail: glantz@nue.tu-berlin.de

A. Krutz
e-mail: krutz@nue.tu-berlin.de

T. Sikora
e-mail: sikora@nue.tu-berlin.de

P. Nunes
Instituto Superior de Ciências do Trabalho e da Empresa,
Instituto de Telecommunicações, Lisbon, Portugal
e-mail: paulo.nunes@lx.it.pt

F. Pereira
Instituto Superior Técnico, Instituto de Telecommunicações, Lisbon, Portugal
e-mail: fp@lx.it.pt

Published online: 19 January 2010

1 Introduction

Sprite coding has been developed and standardized some years ago in MPEG-4 Visual/Part 2 [24, 26]. MPEG-4 Visual defines so-called static and dynamic sprites for background modeling. Static sprites are built offline in a preprocessing step. These are possibly large still images describing panoramic background. For each frame in the sequence, a set of eight motion parameters is transmitted that defines the transformation necessary for reconstruction from the background model. The static sprite image is transmitted before the first image. Other than that, dynamic sprites are built online at the decoder from currently decoded video object planes. Dynamic sprites are then used as reference frames in motion compensation. In this work, only static sprite coding is utilized.

The main idea of the sprite coding approach is to segment the video content into foreground and background objects in a preprocessing step. For the background object, a model, i.e. a so-called background sprite image, is generated, which contains all background pixels of a given part of the sequence. Global motion estimation (GME) techniques initiate the sprite generation process and motion parameters are transmitted as side information to the receiver. The background sprite image and the foreground object sequence are coded separately. At the receiver, a background sequence is reconstructed using the background sprite image and the global motion parameters. The background sequence is merged with the transmitted foreground object sequence to build a representation of the original sequence. Figure 1 illustrates the principle of sprite coding. This approach is very promising, as has been outlined in [27]. It has been shown that even the state-of-the-art video coding standard H.264/AVC [31] can be outperformed using this technique [17, 21].

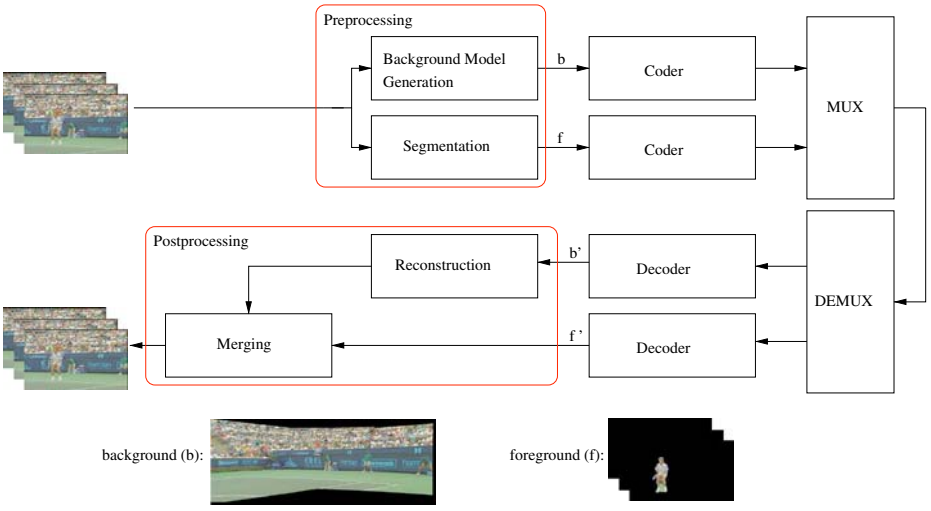


Fig. 1 Sprite-based coding principle

However, to our knowledge no commercial implementation of the MPEG-4 Visual standard has used the sprite coding approach. This is due to the fact that for best results, the preprocessing techniques—i.e. global motion estimation, background modeling and foreground object segmentation—have to be very sophisticated. For global motion estimation, a number of methods based on motion vector fields and pixel-based methods have been proposed. A great deal of these GME techniques has been presented and compared in [12]. Research on sprite generation for video sequences has provided the community with a number of techniques as well, including the work by Farin et al. [7, 8, 10] and Kunter et al. [13, 19, 20].

The biggest challenge in sprite coding is the segmentation step. The MPEG-4 Visual standard assumes that the separated foreground and background objects of the video content are already known. No integrated segmentation step has been defined. The work by Mech et al. uses short-term global motion compensation for automatic object segmentation [23]. Another approach is the use of background models in a background subtraction step for automatic foreground object segmentation [9]. Inspired by this work we developed a new automatic object segmentation algorithm for sequences with a moving camera. Other than in [9], no common sprites are used for background subtraction but a model that we call local background sprites, i.e. a model, that uses the techniques from global background sprite generation but minimizes distortion introduced in one single sprite image for a whole sequence by computation of one local sprite for every frame of the sequence. In this work, we compare a set of integrated automatic foreground object segmentation algorithms in terms of coding performance with the MPEG-4 Visual Advanced Simple Profile (which does not include sprite coding).

This paper is organized as follows. Section 2 describes the pixel-based global motion estimation algorithm that is used for all test scenarios in this paper. Section 3 introduces background modeling techniques including the new algorithm for local background sprite generation. Section 4 outlines all the automatic foreground object segmentation algorithms that are compared in Section 5 in terms of coding efficiency. The last section summarizes this paper.

2 Global motion estimation

The performance of sprite coding critically depends on the accurate estimation of the background motion. Therefore, it is important to apply a global motion estimation technique with very accurate estimation of the higher-order motion parameters. The global motion estimation algorithm used in this work is based on Dufaux et al. [6] and Krutz et al. [13], respectively. It is a gradient descent-based approach using feature tracking as initialization and image pyramid decomposition for estimation refinement. A simplified block diagram of the algorithm is depicted in Fig. 2. The algorithm decomposes the frames that are subject to motion estimation into an image pyramid. It contains two downsampled versions and the original frames. Other than in [6], it comprises an upsampled representation as well. The algorithm then performs a gradient descent step in every layer of the image pyramid and takes the parameters from the step before as initialization. Additionally, the first gradient descent step is initialized translationally by a feature tracking method.

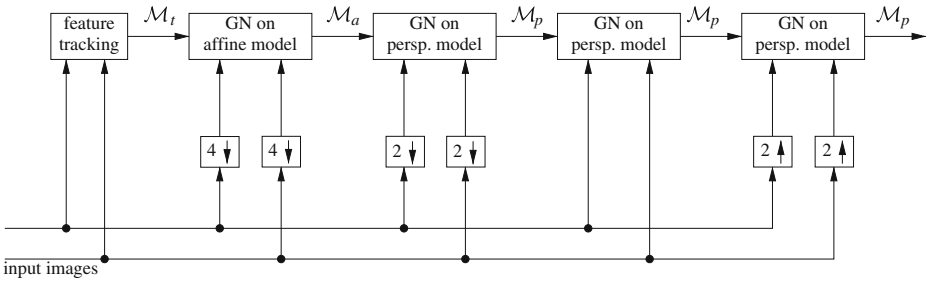


Fig. 2 Gauss–Newton-based (GN) gradient descent approach for global motion estimation using feature tracking for parameter initialization and image pyramid decomposition for estimation refinement

2.1 Motion models and warping

Camera movement introduces displacement between consecutive frames of a video sequence. This displacement can be described by a motion model. The simplest case is translational movement, i.e. the displacement in x and y direction, respectively. The translational motion model $\mathcal{M}_t$ can be described mathematically by

$$\mathcal{M}_t : \begin{aligned} x' &= x + m_0 \\ y' &= y + m_1 \end{aligned}$$

where $(x, y)^T$ describe the coordinates of the current pixel and $(x', y')^T$ the coordinates of its transformed version, respectively. m_0 and m_1 are the displacement values in x and y direction.

For modeling rotation, scaling and shearing, the translational motion model has to be extended. The so-called affine motion model

$$\mathcal{M}_a : \begin{aligned} x' &= m_0 x + m_1 y + m_2 \\ y' &= m_3 x + m_4 y + m_5 \end{aligned}$$

defines the affine transformation between pixels at position $(x, y)^T$ in the current frame and pixels at $(x', y')^T$ in another frame. The parameters m_2 and m_5 describe translation, m_0 and m_4 describe scaling and m_1 and m_3 describe rotation and shearing, respectively.

Assuming, the content of the frames can be approximated by a plane, the affine model can be extended by parameters m_6 and m_7 yielding the perspective motion model

$$\mathcal{M}_p : \begin{aligned} x' &= \frac{m_0 x + m_1 y + m_2}{m_6 x + m_7 y + 1} \\ y' &= \frac{m_3 x + m_4 y + m_5}{m_6 x + m_7 y + 1} \end{aligned} \tag{1}$$

Therefore, the affine motion model $\mathcal{M}_a$ is a special case of the perspective model $\mathcal{M}_p$. The algorithm presented here uses the models presented above for global motion estimation.

Knowing the higher-order motion model parameters for the displacement between two arbitrary frames, the motion can be compensated. This process is called warping, e.g. using the parameters from (1) a 3×3 matrix, often called homography, can be formed that transforms the pixels at $(x, y)^T$ in the current frame to generate a prediction of another frame:

$$\begin{pmatrix} x' \cdot z' \\ y' \cdot z' \\ z' \end{pmatrix} = \mathbf{M} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{2}$$

The same parameters can be used for warping into the other direction. For that, the transformation matrix $\mathbf{M}$ has to be inverted.

In most cases, the transformed coordinates $(x', y')^T$ do not match full integer positions. Therefore, bicubic spline interpolation is used in the proposed algorithm.

2.2 Gradient descent

Gradient descent, also known as method of steepest descent, is a numeric optimization technique for finding the local minimum in a given function. For that, the algorithm iteratively moves along the negative gradient of the function. Since the gradient descent algorithm converges in a local minimum, a good initialization has to be provided to find a global minimum. The initialization procedure used herein is further described in Section 2.4. The gradient descent algorithm used in this work is based on the Gauss–Newton approach, i.e. a combination of Newton's method and Gaussian elimination.

Based on the perspective motion model described in (1), it is assumed that $I(x, y)$ is the current frame of size $M \times N$ and $I'(x', y')$ is its prediction from a reference frame I_{ref} generated by employing (2). Thus, a parameter vector

$$\mathbf{m} = (m_0, m_1, m_2, m_3, m_4, m_5, m_6, m_7)^T$$

is requested that lets $I'(x', y')$ be an optimal prediction of $I(x, y)$. Therefore, an error function is computed by subtraction of both frames. The gradient descent approach has to minimize this error function. The Gauss–Newton method uses the sum of squared differences (SSD) as error function:

$$\epsilon(\mathbf{m}) = \sum_{i=1}^{M} \sum_{j=1}^{N} \left(I'(x'_i, y'_j) - I(x_i, y_j) \right)^2 = \sum_{i=1}^{M} \sum_{j=1}^{N} e_{ij}^2 \tag{3}$$

where M and N describe the frame sizes and e_{ij} is the pixel difference at position $(i, j)^T$. The first derivative of a function yields its gradient. However, (3) depends on frames of a video sequence, i.e. a random process, and thus, cannot be described analytically. Therefore, it has to be approximated by a Taylor series. Assuming, the error function is described sufficiently well by a first order Taylor series, this yields

$$\epsilon(\mathbf{m} + d\mathbf{m}) = \epsilon(\mathbf{m}) + \nabla\epsilon(\mathbf{m})^T \cdot d\mathbf{m}$$

Setting its derivative to zero yields

$$\nabla\epsilon(\mathbf{m}+d\mathbf{m}) = \nabla\epsilon(\mathbf{m}) + \left.\frac{\partial^2\epsilon(\mathbf{m})}{\partial m_p \partial m_q}\right|_{\mathbf{m}} \cdot d\mathbf{m} = 0$$
$$\Leftrightarrow d\mathbf{m} = -H(\mathbf{m})^{-1} \cdot \nabla\epsilon(\mathbf{m})$$

where $H(\mathbf{m})$ is the so-called Hessian matrix containing the second partial derivatives of the error function. In every step k, the algorithm has to update the motion model parameters by computing $d\mathbf{m}$:

$$\mathbf{m}_{k+1} = \mathbf{m}_k + d\mathbf{m}_k \tag{4}$$

However, the computation of the second partial derivatives in the inverted Hessian matrix in every step is computationally intensive. Therefore, the Gauss–Newton method approximates $H(\mathbf{m})$ using the first partial derivatives. The Hessian matrix is

$$H(\mathbf{m}) = \left.\frac{\partial^2\epsilon(\mathbf{m})}{\partial m_p \partial m_q}\right|_{\mathbf{m}} \tag{5}$$

Equation (3) in (5) yields

$$\begin{aligned} H(\mathbf{m}) &= \left.\frac{\partial}{\partial m_p}\frac{\partial}{\partial m_q}\sum_{i=1}^{M}\sum_{j=1}^{N} e_{ij}^2\right|_{\mathbf{m}} \\ &= 2\cdot\sum_{i=1}^{M}\sum_{j=1}^{N}\left[\frac{\partial e_{ij}}{\partial m_p}\frac{\partial e_{ij}}{\partial m_q} + e_{ij}\frac{\partial^2 e_{ij}}{\partial m_p \partial m_q}\right]\Bigg|_{\mathbf{m}} \end{aligned} \tag{6}$$

The second summand in (6) is small compared to the first summand due to its multiplication by the error e_{ij}. Therefore, it can be disregarded and (6) can be simplified:

$$H(\mathbf{m}) = 2\cdot\sum_{i=1}^{M}\sum_{j=1}^{N}\left[\frac{\partial I'(x_i', y_j')}{\partial m_p}\frac{\partial I'(x_i', y_j')}{\partial m_q}\right]\Bigg|_{\mathbf{m}} \tag{7}$$

Furthermore, the gradient of the error function (3) is

$$\begin{aligned} \nabla\epsilon(\mathbf{m}) &= \nabla\sum_{i=1}^{M}\sum_{j=1}^{N} e_{ij}^2 = 2\cdot\sum_{i=1}^{M}\sum_{j=1}^{N} e_{ij}\nabla e_{ij} \\ &= 2\cdot\sum_{i=1}^{M}\sum_{j=1}^{N}\left[I'(x_i', y_j') - I(x_i, y_j)\right]\left.\frac{\partial I'(x_i', y_j')}{\partial m_p}\right|_{\mathbf{m}} \end{aligned} \tag{8}$$

The partial derivatives in (7) and (8) are computed using

$$\left.\frac{\partial}{\partial m_p} I'(x', y')\right|_{\mathbf{m}} = \left.\frac{\partial}{\partial m_p} I'\left(\frac{m_0 x + m_1 y + m_2}{m_6 x + m_7 y + 1}, \frac{m_3 x + m_4 y + m_5}{m_6 x + m_7 y + 1}\right)\right|_{\mathbf{m}}$$

for the perspective motion model $\mathcal{M}_p$ from (1), which yields the following equations

$$\begin{aligned}
\frac{\partial I'(x', y')}{\partial m_0} &= \frac{\partial I'(x', y')}{\partial x'} \cdot \frac{\partial x'}{\partial m_0} = I'_x \cdot \frac{x}{D} \\
\frac{\partial I'(x', y')}{\partial m_1} &= \frac{\partial I'(x', y')}{\partial x'} \cdot \frac{\partial x'}{\partial m_1} = I'_x \cdot \frac{y}{D} \\
\frac{\partial I'(x', y')}{\partial m_2} &= \frac{\partial I'(x', y')}{\partial x'} \cdot \frac{\partial x'}{\partial m_2} = I'_x \cdot \frac{1}{D} \\
\frac{\partial I'(x', y')}{\partial m_3} &= \frac{\partial I'(x', y')}{\partial y'} \cdot \frac{\partial y'}{\partial m_3} = I'_y \cdot \frac{x}{D} \\
\frac{\partial I'(x', y')}{\partial m_4} &= \frac{\partial I'(x', y')}{\partial y'} \cdot \frac{\partial y'}{\partial m_4} = I'_y \cdot \frac{y}{D} \\
\frac{\partial I'(x', y')}{\partial m_5} &= \frac{\partial I'(x', y')}{\partial y'} \cdot \frac{\partial y'}{\partial m_5} = I'_y \cdot \frac{1}{D} \\
\frac{\partial I'(x', y')}{\partial m_6} &= \frac{\partial I'(x', y')}{\partial x'} \cdot \frac{\partial x'}{\partial m_6} + \frac{\partial I'(x', y')}{\partial y'} \cdot \frac{\partial y'}{\partial m_6} \\
&= -\frac{x}{D} \cdot (I'_x x' + I'_y y') \\
\frac{\partial I'(x', y')}{\partial m_7} &= \frac{\partial I'(x', y')}{\partial x'} \cdot \frac{\partial x'}{\partial m_7} + \frac{\partial I'(x', y')}{\partial y'} \cdot \frac{\partial y'}{\partial m_7} \\
&= -\frac{y}{D} \cdot (I'_x x' + I'_y y')
\end{aligned}$$

where $D = m_6 x + m_7 y + 1$, $I'_x = \frac{\partial I'(x', y')}{\partial x'}$ and $I'_y = \frac{\partial I'(x', y')}{\partial y'}$, respectively.

Table 1 shows the Gauss–Newton based algorithm used herein, which is based on the work by Baker et al. [2]. The approach does not warp the reference frame in every loop using the current set of motion model parameters. Instead, it warps the current frame $I(x, y)$. Thus, the partial derivatives and the inverted Hessian matrix can be computed before entering the loop. In terms of computational complexity, this is very efficient. Variable T in Step 4.5 of the algorithm is a predefined threshold and $k_{\max}$ in Step 4.7 describes the maximum number of iterations.

Table 1 Fast Gauss–Newton algorithm [2]

Step 1	Compute the gradient of $I'(x', y') = I_{ref}$ (I'_x and I'_y)
Step 2	Compute the derivatives $\partial I'(x', y')/\partial m_p, \forall m_p \in \mathbf{m}$
Step 3	Compute the inverted Hessian matrix
Step 4.1	Warp frame $I(x, y)$ using the current parameter vector $\mathbf{m}$ and (2)
Step 4.2	Compute error frame between $I'(x', y')$ and the warped frame from Step 4.1
Step 4.3	Compute $\nabla\epsilon(\mathbf{m})$
Step 4.4	Update motion model parameters as described in (4)
Step 4.5	If $SSD_k < T$, stop the algorithm
Step 4.6	If $SSD_k > SSD_{k-1}$, stop the algorithm after undoing the update from Step 4.4
Step 4.7	If $k >= k_{\max}$, stop the algorithm
Step 4.8	Goto Step 4.1

2.3 Image pyramid decomposition

The algorithm for global motion estimation presented herein first generates a four-step image pyramid containing the original frame, two downsampled and an upsampled version. On the one hand, this decreases computational complexity, since the algorithm first approaches the local minimum in lower resolutions and then refines the estimation in higher resolutions. On the other hand, downsampling results in smoothing of the error function. Thus, the gradient descent approach converges faster. Although the results in lower resolutions are not optimal, they pose a good initialization for computation in the next layer of the image pyramid.

As can be seen in Fig. 2, the various gradient descent steps compute the motion parameters for different models, i.e. translational, affine, and perspective. Therefore, the parameters have to be adjusted between the steps. For transition from lower to higher resolution this means

$$\begin{pmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & 1 \end{pmatrix} \longrightarrow \begin{pmatrix} m_0 & m_1 & m_2 \cdot 2 \\ m_3 & m_4 & m_5 \cdot 2 \\ \frac{m_6}{2} & \frac{m_7}{2} & 1 \end{pmatrix}$$

For transition from higher to lower resolution this means

$$\begin{pmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & 1 \end{pmatrix} \longrightarrow \begin{pmatrix} m_0 & m_1 & \frac{m_2}{2} \\ m_3 & m_4 & \frac{m_5}{2} \\ m_6 \cdot 2 & m_7 \cdot 2 & 1 \end{pmatrix}$$

Downsampling an image results in information loss. Therefore, a low-pass filter is employed before downsampling to avoid aliasing. Here, a five-tap Le-Gall wavelet filter is used. Additionally, the upsampled version is generated by employing a seven-tap Daubechies filter for interpolation, which is a good approximation of the sinc function and therefore near the optimum [1, 4, 5].

2.4 Initialization using feature tracking

For the gradient descent approach in global motion estimation, a good initialization is essential. This is because the algorithm runs the risk of converging to a local minimum that differs from the global minimum. Looking at Fig. 2, one can clearly see that initialization is performed using the result from the step before. This is done on all hierarchical layers but the first. Since translational movement results in a large displacement of the global minimum of the error function, it is requested that the first layer is provided with a good translational initialization. The algorithm presented herein uses the feature tracking algorithm presented by Kanade, Lucas, Shi and Tomasi (KLT) [3, 22, 28, 30].

The KLT feature tracker is a window-based approach that minimizes the squared error differences between a current and a reference window similar to the gradient descent approach presented in Section 2.2. However, it uses the second partial derivatives in the Hessian matrix instead of an approximation by the first partial derivatives. Kanade et al. assume that the movement between consecutive frames is small and that it can therefore be approximated by a translational motion model. An

affine motion model is only used for verification of window correspondences after a certain temporal distance between them. Since in this work, feature tracking is needed for initialization, only the translational motion model is used.

Besides tracking of feature windows, their selection is an important issue. Shi and Tomasi define that *a good feature is one that can be tracked well* [28]. Thus, a feature window is good if the matrix

$$\mathbf{Z} = \begin{pmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{pmatrix},$$

which comprises its summed gradients, lies both above the noise level of the frame and is well-balanced. It is

$$g_x^2 = \sum_{i=0}^{P-1} \sum_{j=0}^{Q-1} g_x^2(i,\, j)$$

$$g_y^2 = \sum_{i=0}^{P-1} \sum_{j=0}^{Q-1} g_y^2(i,\, j)$$

$$g_x g_y = \sum_{i=0}^{P-1} \sum_{j=0}^{Q-1} g_x(i,\, j) \cdot g_y(i,\, j)$$

where $g_x(i,\, j)$ is the gradient of the window in x direction, $g_y(i,\, j)$ is the gradient in y direction and $P \times Q$ is the resolution of the feature window. The constraint concerning the noise level means that both eigenvalues of $\mathbf{Z}$ are large. Well-balanced means that the difference between the eigenvalues is not too large, e.g. two small eigenvalues indicate a homogeneous region and a large and a small eigenvalue indicate a unidirectional pattern. Only two large eigenvalues indicate salient features that can be tracked well. Therefore, a feature window is only used if its eigenvalues λ_1 and λ_2 lie above a predefined threshold λ_T.

The algorithm used for global motion estimation chooses the $N_{\max}$ best feature windows from a given reference frame. Then, the KLT tracker finds their correspondences in the current frame. Features can get lost due to occlusion or because they left the camera scope. Therefore, a set of motion vectors $N \leq N_{\max}$ is available for initialization. Afterwards, outliers are removed using the random sample consensus (RANSAC) algorithm [11]. Finally, a translational estimate of the displacement between current and reference frames is generated by computing the mean of the remaining motion vectors. This estimate represents the initialization for the first gradient descent step on the coarsest resolution of the image pyramid.

3 Background modeling

Background modeling means the description of the background of a video sequence. Application scenarios include video analysis for surveillance systems as well as panoramic image generation in state-of-the-art digital cameras. Here, background models are needed on the one hand for sprite-based video coding as defined in MPEG-4 Visual [24, 26]. On the other hand, for background subtraction in foreground object segmentation. Both application scenarios necessitate a model of the

background of a given sequence. In the following, three algorithms for background modeling are explained in detail. The first two algorithms, i.e. single and multiple background sprites, are based on the work done by Kunter et al. [13, 19, 20] and Farin et al. [7, 8, 10]. For the final experimental setup, these algorithms are used both for sprite-based video coding and foreground object segmentation in a background subtraction method. The third background modeling approach, i.e. local background sprites, is based on the work by Krutz et al. and is exclusively used for segmentation [14–16].

3.1 Single background sprites

A single background sprite models the background of a given sequence in one single image. This image is usually of large size and contains only the pixels from the background of the sequence. An example of a single sprite for the *Stefan* sequence is depicted in Fig. 3.

For the creation of a single sprite, an arbitrary reference frame is chosen. For best results, i.e. minimal distortion in the background sprite image, the reference frame is the center frame in terms of camera pan. All other frames of the sequence are warped into the coordinate system of the reference. Therefore, long-term higher-order motion parameters, i.e. parameters based on the motion models introduced in Section 2.1, are computed that describe this transformation [29]. First, short-term parameters are calculated using the global motion estimation approach presented in Section 2. These short-term parameters are then accumulated by simple matrix multiplication as shown in Fig. 4.

By transforming all frames into the coordinate system of the background sprite using long-term parameters, a stack of size $M \times N \times S$ is created where $M \times N$ is the resolution of the final background sprite and S is the number of frames in the sequence. The images in this stack are then blended together to generate the background sprite. Blending filters normally used are linear filters like the mean or non-linear filters like the median. However, even more complex approaches like Wiener filtering can be utilized to reach an optimal result. The sprite blending principle is shown in Fig. 5.

One problem in single background sprite generation is the choice of the reference frame. Depending on the amount of motion induced by the movement of the camera, severe geometrical distortion can be caused. In the example in Fig. 3 this problem has been minimized by the selection of an optimal reference frame in terms of distance to the borders of the sprite image. If the first frame of the sequence, i.e. the frame that has been mapped onto the right side of the sprite image, had been used, distortion

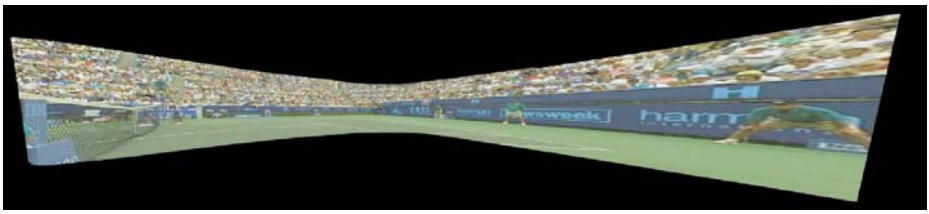


Fig. 3 Single background sprite, sequence *Stefan*, reference frame 253

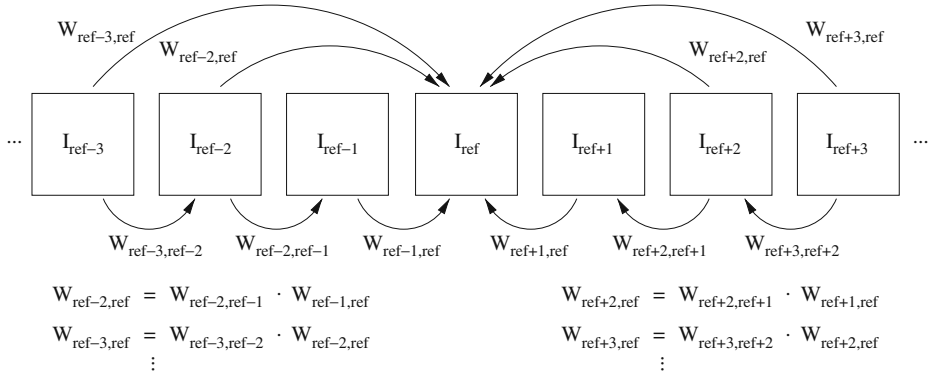


Fig. 4 Generation of long-term higher-order motion parameters [29]. The short-term parameters used for generation have been calculated using the approach presented in Section 2

on the left side would have been even higher. Additionally, the error in long-term motion parameters increases with growing distance to the reference frame. This is due to error accumulation by multiplication of short-term parameter matrices. This problem can as well be minimized by the right selection of the reference frame.

3.2 Multiple background sprites

As mentioned before, strong camera motion can result in severe geometrical distortion when generating a single background sprite image for a complete video sequence. This distortion causes decreased quality of the background model and additionally increases coding cost due to large image resolutions needed for representation. This can be seen in Fig. 3, where only a part of the sprite image is actually used for background content. In some cases where the camera pan exceeds an angle

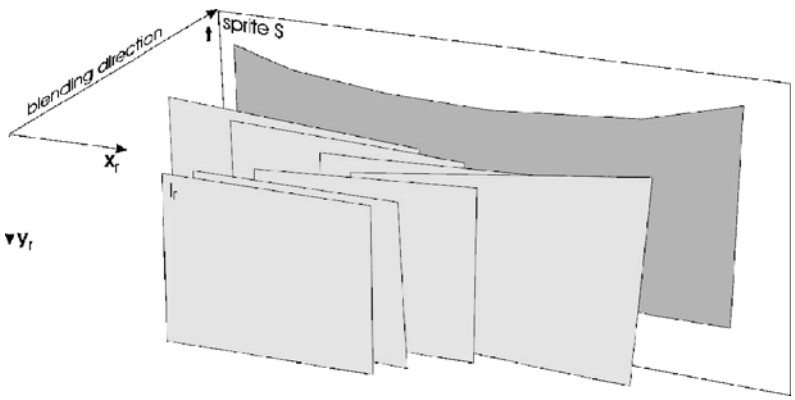


Fig. 5 Sprite blending principle: An image stack of spatially aligned frames is built by warping all frames into the coordinate system of the reference frame I_r. Blending is performed for all pixel values into the time direction [19]

of 90 degrees, generation of one single background sprite image is not even possible. These drawbacks have led to so-called multiple background sprites.

For the generation of multiple background sprites, the camera motion is analyzed first. The next step is to split the video sequence into partitions depending on the motion analyzed before. Afterwards, a background sprite is generated independently for every partition as described in Section 3.1.

For analysis, we exploit the fact that for common camera setups a transformation matrix $\mathbf{W}_{0,n}$, i.e. the homography describing the motion between a reference frame I_0 and another frame I_n, can be decomposed into a product of intrinsic and extrinsic camera parameter matrices

$$\begin{aligned}\mathbf{W}_{0,n} &= \mathbf{F}_n \mathbf{R}_{0,n} \mathbf{F}_0^{-1} \\ &= \frac{1}{\alpha_{0,n}} \begin{pmatrix} r_{00} & r_{01} & f_0 r_{02} \\ r_{10} & r_{11} & f_0 r_{12} \\ r_{20}\dfrac{\alpha_{0,n}}{f_0} & r_{21}\dfrac{\alpha_{0,n}}{f_0} & r_{22}\alpha_{0,n} \end{pmatrix}\end{aligned} \tag{9}$$

where $\mathbf{R}_{0,n}$ is the rotation matrix between frame I_0 and I_n, and $\mathbf{F}_n$ and $\mathbf{F}_0$ contain focal length values of both views. After computation of the focal length ratio $\alpha_{0,n} = \frac{f_0}{f_n}$, we calculate the focal length of the reference frame as the median of all solutions from (9). This is done by exploiting orthogonality and constant vector norm constraints for the matrices $\mathbf{W}_{0,n}$. Knowing all focal lengths, the rotation angles can finally be computed using trigonometrical properties of the center points for every image [20]. Figure 6 shows the basic principle for the creation of multiple background sprites for a sequence with a large camera pan. Figure 7 shows the multiple background sprites generated with the approach presented above for the *Stefan* video sequence.

3.3 Local background sprites

A local background sprite (LBS) specifies a model of the background. Other than general background sprites one model is built for every frame and not one model for the whole video sequence. Only the local temporal neighborhood of each reference frame is taken into account for sprite generation. The dimensions of a local background sprite match those of the reference frame. The idea is to minimize distortion in background regions. When a background frame is reconstructed from a

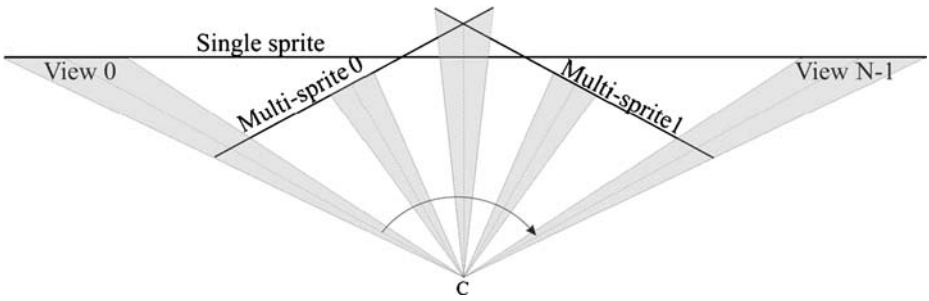


Fig. 6 Example for the partitioning of a video sequence into multiple background sprites for panning camera with constant focal length [19]

(a) Frames 0-244

(b) Frames 245-261

(c) Frames 262-299

Fig. 7 Multiple background sprites, sequence *Stefan*

general background sprite, distortion can be severe. This is due to accumulated errors in the global motion estimation, non-ideal interpolation and the double mapping into the coordinate system of the background sprite and back. The algorithm for modeling local background sprites for a given video sequence is depicted in Fig. 8.

For global motion estimation, the hierarchical gradient descent approach based on the Gauss–Newton method as presented in Section 2 is applied. Since the short-term displacement between two frames I_p and I_q is used several times while creating all local background sprites for a video sequence, the motion parameters are computed in a preprocessing step. This means for a sequence with n frames the set T of transformation matrices

$$T = \{\mathbf{W}_{0,1}, \mathbf{W}_{1,2}, \ldots, \mathbf{W}_{n-2,n-1}\}$$

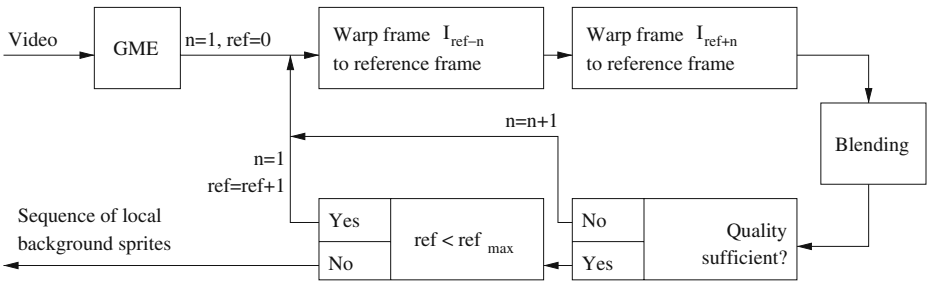


Fig. 8 Modeling the background by means of local background sprites

and its inverted correspondences

$$T_{inv} = \left\{\mathbf{W}_{0,1}^{-1}, \mathbf{W}_{1,2}^{-1}, \ldots, \mathbf{W}_{n-2,n-1}^{-1}\right\}$$

are computed where $|T| = |T_{inv}| = n - 1$, $\mathbf{W}_{p,q}^{-1} = \mathbf{W}_{q,p}$ and

$$\mathbf{W}_{p,q} = \begin{pmatrix} m_{0,p,q} & m_{1,p,q} & m_{2,p,q} \\ m_{3,p,q} & m_{4,p,q} & m_{5,p,q} \\ m_{6,p,q} & m_{7,p,q} & 1 \end{pmatrix}$$

is the transformation matrix between frames I_p and I_q.

For every reference frame, a local background sprite is built. The algorithm iteratively transforms neighboring frames into the coordinate system of the reference. This produces a dynamically growing image stack of size $M \times N \times S_t$ where M and N are the dimensions of the reference frame and $S_t = 2t + 1$ is the depth of the stack in step t. In step $t = 0$, the stack only contains the reference frame. This approach can be seen in Fig. 9.

For the transformation of an arbitrary frame into the reference's coordinate system, the short-term motion parameters from the preprocessing step are accumulated to generate long-term parameters, which can be seen in Fig. 4. The global motion estimation can only compute the displacement between two frames by approximation. Due to existing small errors and the accumulation, the error in the long-term parameters grows larger with increasing temporal distance to the reference frame. Hence, the long-term parameters are used as initialization for another gradient descent step to reduce this error.

In every step t, the images in the stack are merged together to build a preliminary local background sprite of size $M \times N$. For this purpose, a so-called blending filter is used, which here is a median filter. The median returns the middle value of an ordered dataset, in this case a set of luminance and chrominance values, respectively. The advantage compared to using a mean filter is its robustness to outliers. Additionally, the median is always an element of the set itself and does not, therefore, produce new values.

By successively adding temporally neighboring frames, the foreground objects in the preliminary local background sprites are removed step by step. This is due to the area behind the foreground objects that is disclosed because of their movements. This can be seen in Fig. 10. Depicted are the preliminary local background sprites

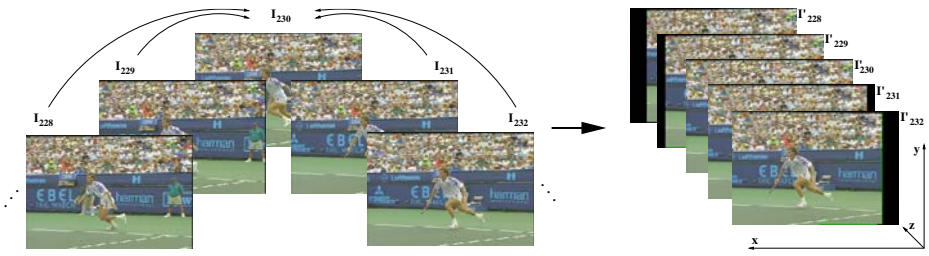
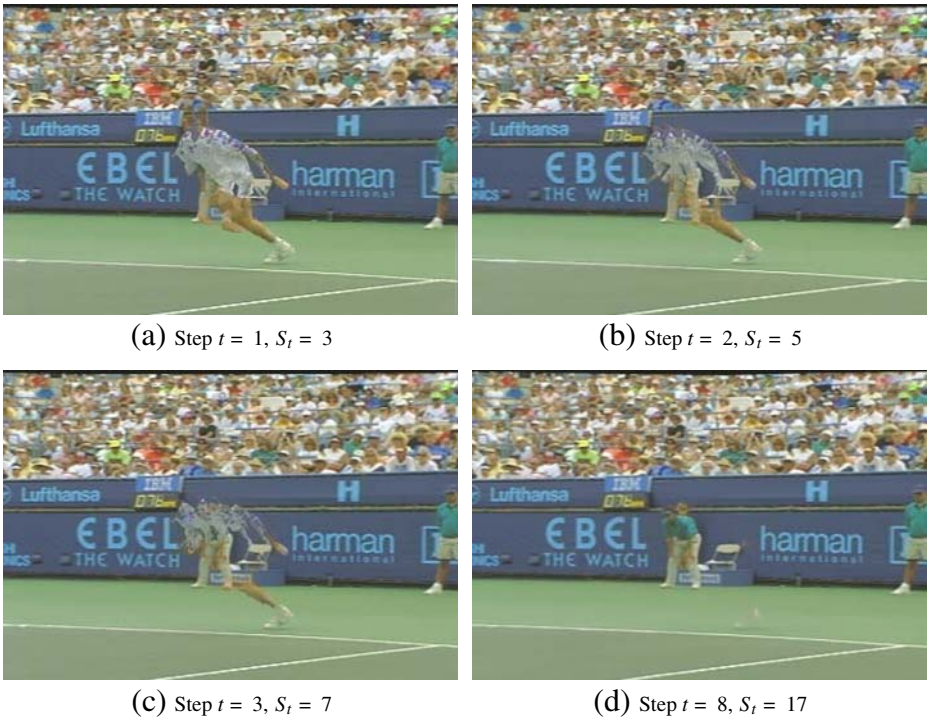


Fig. 9 Creation of an image stack for the generation of a local background sprite, sequence *Stefan*, reference frame 230



(a) Step $t = 1$, $S_t = 3$ (b) Step $t = 2$, $S_t = 5$

(c) Step $t = 3$, $S_t = 7$ (d) Step $t = 8$, $S_t = 17$

Fig. 10 Preliminary local background sprites, sequence *Stefan*, reference frame 230

for the *Stefan* sequence for various steps t. One can clearly see that the foreground object has nearly completely vanished after eight blending steps.

It is possible to evaluate the quality of the background model in every step subjectively. However, an automatic evaluation criterion is desirable that stops the generation of the local background sprite when its quality is good enough and the foreground objects are removed sufficiently. Therefore, an approach for automatic quality evaluation is applied.

A possible measure for the difference between two frames is the root mean square error (RMSE). The RMSE between a reference frame $I_{\text{ref}}(x, y)$ and its preliminary local background sprite $I_{\text{bs},t}(x, y)$ in step t is defined by

$$\text{RMSE}_t = \sqrt{\frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (I_{\text{ref}}(i, j) - I_{\text{bs},t}(i, j))^2}$$

where M and N are the dimensions of the reference frame and the preliminary local background sprite. The preliminary local background sprite in step $t = 0$ is the reference frame itself so that $I_{\text{bs},t=0} = I_{\text{ref}}$ and $\text{RMSE}_{t=0} = 0$. Since the foreground objects vanish step by step, the RMSE value increases successively. However, the difference of the RMSE values in two consecutive steps

$$\Delta \text{RMSE}_t = \text{RMSE}_t - \text{RMSE}_{t-1} \tag{10}$$

decreases. When the foreground objects are completely eliminated, the values in (10) change only marginally.

At the beginning, foreground objects are still present in the preliminary local background sprite. Change in these regions leads to high values of $\Delta\,\mathrm{RMSE}_t$. After several steps, most of the foreground is eliminated, which leads to lower values of $\Delta\,\mathrm{RMSE}_t$. It holds true that

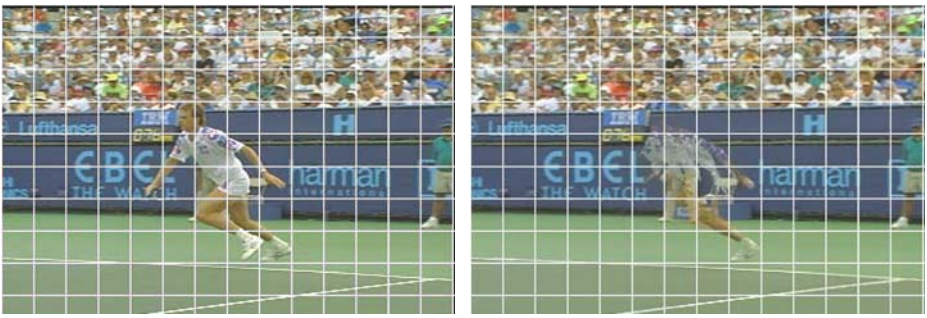
$$\Delta\,\mathrm{RMSE}_a >= \Delta\,\mathrm{RMSE}_b$$

for $a <= b$. The value $\Delta\,\mathrm{RMSE}_t$ can be interpreted as a measure for the information about the background that has been added to the local background sprite in step t.

However, using the measure $\Delta\,\mathrm{RMSE}_t$ is not without problems when only small foreground objects are present since small objects only take a minor percentage of the whole frame. The influence of errors in these regions on the measure is small compared to the sum of errors in the background regions. In this case, plotting RMSE_t and $\Delta\,\mathrm{RMSE}_t$ against time produces very flat curves, which make a decision on the quality of the preliminary local background sprite very difficult. Therefore, we define matrices containing the blockwise calculated value $\Delta\,\mathrm{RMSE}_t$, which we call dRMSE matrices.

Reference frame and preliminary local background sprite are divided into blocks of fixed size, which can be seen in Fig. 11. In this work, various block sizes between 10×10 and 40×40 have been tested. The problem with small block sizes is that the profile of the dRMSE matrices is of high frequency. With large block sizes, the unwanted effect of averaging of large regions sets in again. Here, a fixed block size of 25×25 has shown to be useful. The value RMSE_t is then calculated for every block independently. Therefore, no averaging over the whole frame takes place. Distinct areas in the preliminary local background sprite can now be evaluated independently. Furthermore, the difference to the block values in the step before is computed using (10).

Figure 12 shows the corresponding matrices for the example in Fig. 10. At the beginning, the plot of the matrix is very wavy. With increasing steps t the matrix flattens successively. This means, the more temporally neighboring frames are transformed into the local background sprite's coordinate system the less information



(a) Reference frame (b) Preliminary LBS, $t = 2$

Fig. 11 Partitioning of images into blocks of size 25×25, sequence *Stefan*, reference frame 230

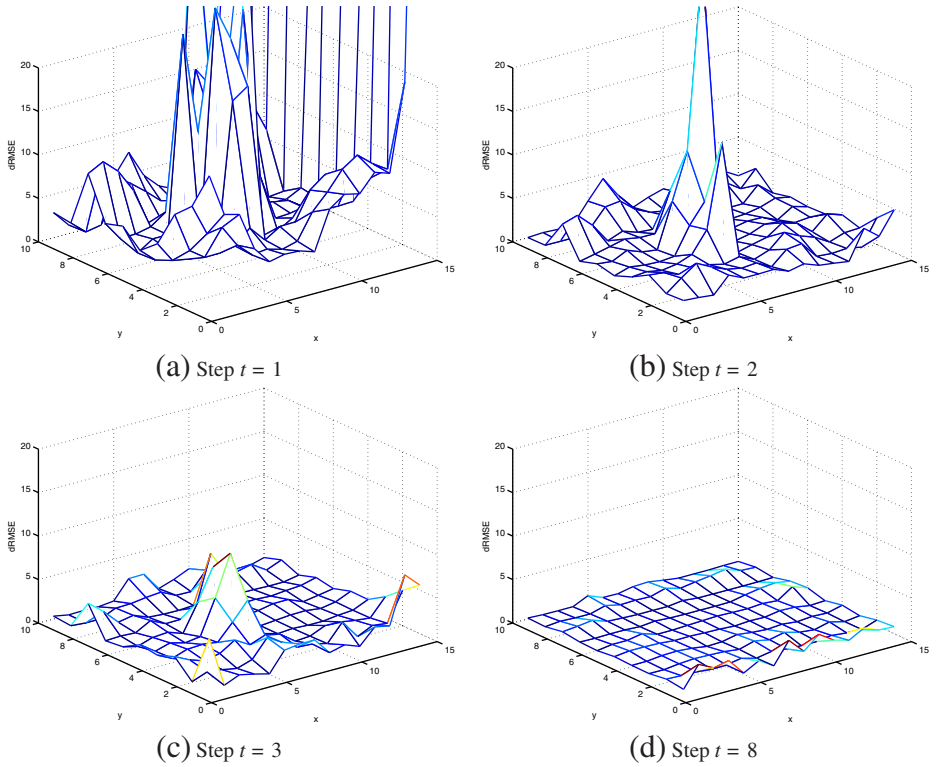


Fig. 12 dRMSE matrices using blocks of size 25×25, sequence *Stefan*, reference frame 230. This example corresponds to the preliminary local background sprites from Fig. 10

about the background of the reference frame is gained. The peak in the middle of the matrices in Fig. 12 results from the moving tennis player and the area behind him that is disclosed. However, the RMSE in the background regions doesn't change significantly. After 8 blending steps, i.e. after 16 frames and the reference frame have been blended (cf. Fig. 12d), the matrix is nearly flat in all regions. This corresponds to the results in Fig. 10.

The quality of the preliminary local background sprites is now assessable in a very differentiated way. Assuming that the generation of the local background sprite is to be aborted when there is no more information added in any region, meaning the matrix presented is flat in every region, one possible way to stop the local background sprite generation is evaluating the maximum value of the dRMSE matrices. The generation is aborted if the maximum value falls below a threshold of 5.0.

4 Foreground object segmentation

In this section we introduce three automatic foreground object segmentation algorithms that are based on global motion estimation and background model generation.

All algorithms generate an error frame for a given reference that is processed using the segmentation algorithm previously published by Krutz et al. [18]. Sections 4.1 to 4.3 describe the algorithms for error frame generation. Section 4.4 deals with the actual segmentation algorithm that is common for all approaches.

4.1 Error frame generation using short-term motion compensation

The first approach for automatic foreground object segmentation is based on earlier work by Mech et al. [23]. Figure 13 illustrates its processing chain. Here, global motion compensation (GMC) means that the background motion of two consecutive frames of a video sequence is estimated and compensated. When an error frame E_i is calculated by applying GMC using the reference frame I_{ref} and one of its neighbors $I_{\text{ref}-1}$ or $I_{\text{ref}+1}$, the background pixel values are ideally removed completely. However, the foreground object regions from both frames that are involved appear in the error frame. This results in non-optimal segmentation since for moving foreground objects their region is extended. Therefore, two error frames are computed, i.e. E_1 using the reference frame I_{ref} and its predecessor $I_{\text{ref}-1}$ and E_2 using the reference frame and its successor $I_{\text{ref}+1}$. The segmentation algorithm presented in Section 4.4 is then applied for both error frames to create two preliminary binary masks B_1 and B_2. Assuming both preliminary binary masks contain oversegmented foreground regions, a better estimate of the correct foreground region is then found by combining both masks using a logical AND operation. This generates the final binary foreground object mask B_{ref}.

4.2 Error frame generation using single and multiple background sprites for background subtraction

Single and multiple background sprites, as introduced in Sections 3.1 and 3.2, can be used as background model for background subtraction-based foreground object segmentation. This approach has been outlined before, e.g. by Farin et al. [9], and turned out to be very promising. However, the error frames created using background subtraction of reconstructed sprite image and original frame occasionally result in oversegmented regions in the background area. This is due to the fact that depending on the amount of camera motion the mapping of pixel content into the coordinate system of the background sprite images causes distortion. Additionally, non-optimal interpolation and short-term global motion estimation errors increase this negative effect. Thus, this approach is combined with the short-term global

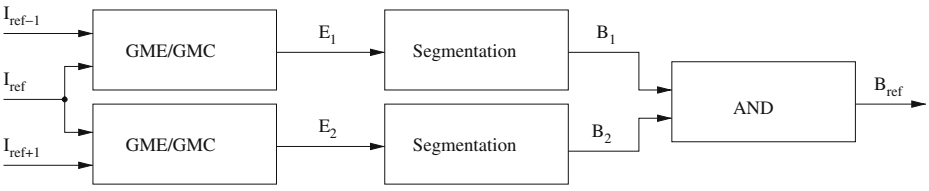


Fig. 13 Short-term motion compensated error frame generation and foreground object segmentation (*Algorithm1*) based on Mech et al. [23]

motion compensation technique outlined in Section 4.1, i.e. one processing chain from *Algorithm1*. A block diagram of this approach is shown in Fig. 14. The algorithm first performs a global motion estimation step to estimate the short-term motion between every two successive frames of the sequence. This is done as presented in Section 2. The estimated motion parameters are then used to generate a binary foreground mask for a given reference frame I_{ref} by combination of two approaches. The first approach, i.e. the upper processing chain in Fig. 14, generates a preliminary binary foreground mask B_1 by global motion compensation between the reference frame and its predecessor $I_{\text{ref}-1}$ to have an exact removal of background regions in the error frame E_1. The second approach, i.e. the lower processing chain in Fig. 14, generates a preliminary binary foreground mask B_2 by background sprite generation, reconstruction and background subtraction to have an exact contour of the foreground object area in the error frame E_2. Both approaches use the same error frame segmentation algorithm. The binary masks B_1 and B_2 are afterwards combined using a logical AND operation.

4.3 Error frame generation using local background sprites for background subtraction

Using common background sprites, i.e. single or multiple background sprites, for foreground object segmentation in a background subtraction method can lead to non-optimal results. This is due to the fact that, besides the mapping of pixel content into the coordinate system of the sprite image, a second mapping has to be performed for background frame reconstruction. This problem is illustrated in Fig. 15. This second mapping may result in distortion that is due to errors in motion estimation and non-ideal interpolation. If the background frame representation is distorted, background regions in the final binary object mask may be oversegmented. *Algorithm2* in Section 4.2 tries to solve this problem by combination with a parallel processing chain that uses short-term motion compensation for exact background removal in the binary mask.

However, the quality of the background model can be enhanced even more if local background sprites are used (cf. Section 3.3) for background subtraction. Therefore, *Algorithm3* uses this model for background representation. The algorithm is depicted in Fig. 16. The algorithm first performs a short-term global motion estimation between every two successive frames of the sequence to generate a set of short-term homographies. These are then used to generate a local background sprite, as

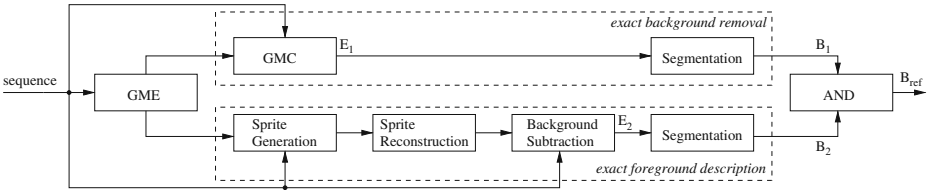


Fig. 14 Foreground object segmentation combining short-term global motion compensation for exact background removal and single/multiple sprite-based background subtraction for exact foreground contour description (*Algorithm2*)

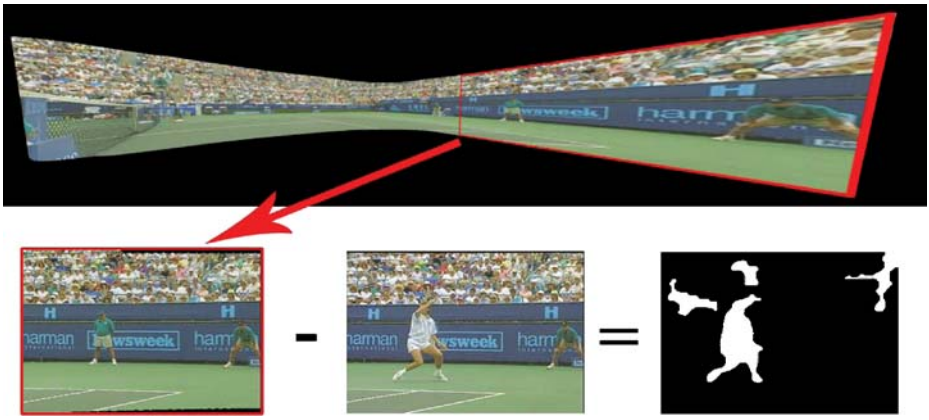


Fig. 15 Illustration of the pixel mapping problem, which occurs when using common background sprites for foreground object segmentation: For reconstruction of the background of a single frame, a pixel mapping needs to be performed causing distortion. When used in a background subtraction algorithm, this can result in non-optimal segmentation

described in Section 3.3, for a given reference frame I_{ref}. The subsequent background subtraction produces a very exact error frame E_{ref} that is afterwards processed using the segmentation approach that is common to the other algorithms to produce a binary foreground object mask B_{ref}.

4.4 Error frame processing for foreground object segmentation

The foreground object segmentation approach relies on precomputed error frames, e.g. using one of the approaches from Sections 4.1 to 4.3. The algorithm itself has been published before by Krutz et al. [18] and is listed in Table 2.

The algorithm assumes significant difference between foreground and background regions in the error frames (Step 1). The generation of the final binary foreground object mask is then realized in several steps. First, the absolute values of the pixels in the error frame are computed (Step 2).

Assuming, distortion in the background regions is high-frequency, the error values in these areas can be smoothed using a low-pass filter. However, common low-pass filters, e.g. Gaussian filtering, do not differentiate between high frequencies caused by noise and high frequency caused by object borders. Therefore, the anisotropic

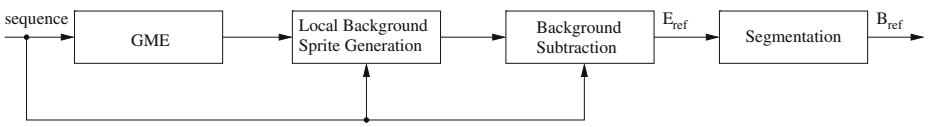


Fig. 16 Foreground object segmentation based on background subtraction using local background sprites (*Algorithm3*)

Table 2 Anisotropic diffusion-based segmentation algorithm

Step 1	Compute error frame E_{ref} using one of the approaches from Sections 4.1 to 4.3.
Step 2	Compute absolute pixel values of error frame E_{ref}.
Step 3	Perform anisotropic diffusion-based low-pass filtering.
Step 4	Intensity rescaling.
Step 5	Create preliminary binary mask using weighted mean thresholding.
Step 6	Morphological operation 1: Remove small objects.
Step 7	Morphological operation 2: Closing.
Step 8	Morphological operation 3: Fill holes.
Step 9	Morphological operation 4: Remove small objects.

diffusion-based low-pass filter proposed by Perona et al. is used [25] (Step 3). The algorithm is based on the diffusion equation, also known as heat equation

$$I_t = \text{div}(\rho(x, y)\nabla I(x, y)) \tag{11}$$

where $\rho(x, y)$ is the diffusion coefficient and $\nabla I(x, y)$ is the gradient of frame $I(x, y)$. If $\rho(x, y)$ is constant, I_t is simplified to the isotropic diffusion equation. Assuming the position of object borders is known, a setting of $\rho(x, y) = 1$ inside homogeneous regions and $\rho(x, y) = 0$ at object borders would be requested. However, these positions are not known. Therefore, an estimate is necessary. The absolute value of the gradient of $I(x, y)$ can be such an estimate. Then, $\rho(x, y)$ is a function $g(\cdot)$ of this estimate:

$$\rho(x, y) = g(|\nabla I(x, y)|)$$

A non-negative monotonically decreasing function has to be chosen for $g(\cdot)$ with $g(0) = 1$. This means that for regions with small gradient values, i.e. $|\nabla I(x, y)| \to 0$, it is $g(|\nabla I(x, y)|) \to 1$. A stronger filtering in homogeneous regions would be the result. Here,

$$\rho(x, y) = \frac{1}{1 + \left(\frac{|\nabla I(x,y)|}{\kappa}\right)^2} \tag{12}$$

is used, where κ is a predefined constant. Equation 12 in (11) yields

$$I_t = \text{div}\left(\frac{1}{1 + \left(\frac{|\nabla I(x,y)|}{\kappa}\right)^2}\nabla I(x, y)\right)$$

After generation of a filtered error frame, its values are normalized by intensity value rescaling to [0, 1] (Step 4). Then, a preliminary binary mask is calculated by thresholding (Step 5). The threshold t is computed as a weighted mean using

$$t = \text{mean}(I_{t,\text{norm}}) + \varsigma \cdot (\max(I_{t,\text{norm}}) - \text{mean}(I_{t,\text{norm}}))$$

where $I_{t,\text{norm}}$ is the filtered and normalized error frame and ς is a constant for weighting the threshold.

Finally, the preliminary binary mask is processed using a set of morphological operators. First, small objects are removed (Step 6), i.e. regions smaller than a predefined threshold. Then, a closing operation is performed to link neighboring objects

(Step 7). Assuming foreground objects contain no holes, these are filled (Step 8). At the end, small objects with a threshold larger than the first removal operation are removed again (Step 9). This step concludes the segmentation algorithm.

5 Experimental evaluation

For the experimental evaluation, we considered six test sequences that are listed in Table 3. First, background sprite images have been generated for all test sequences using the approaches presented in Sections 3.1 and 3.2. However, multiple sprites have only been generated for *Biathlon* and *Stefan*, since the camera pan in all other sequences is too narrow to make this worthwhile. In the multiple sprites-case, the sequence *Biathlon* is divided into four partitions (frames 0 to 9, 10 to 22, 23 to 46, and 47 to 199) and *Stefan* is divided into three partitions (frames 0 to 244, 245 to 261, and 262 to 299). Some examples of background sprite images are depicted in Fig. 17.

Second, segmentation has been performed using all test sequences from Table 3 and all given algorithms, i.e. *Algorithm1* based on short-term global motion compensation (cf. Section 4.1), *Algorithm2* based on background subtraction using single and multiple sprites (cf. Section 4.2), and *Algorithm3* based on background subtraction using local background sprite modeling (cf. Section 4.3). Additionally, for the sequences *Biathlon* and *Stefan*, groundtruth masks for the foreground objects were available. Therefore, we also used these as an ideal segmentation case for sprite coding to evaluate the influence of correct segmentation in terms of coding performance.

Last, the sequences were coded using the sprite coding approach presented in the introduction. Therefore, the MPEG-4 Visual reference coder software MoMuSys (Mobile Multimedia Systems) has been used applying the Main Profile (MP) for sprite coding. Additionally, all sequences were coded as one rectangular video object using the Advanced Simple Profile (ASP) for comparison. The quantization parameter (QP) has been kept constant ($QP_{bg} = 14$) for the background model for all test sequences. The foreground object sequences have been coded using one of several quantization parameters, i.e. $QP_{fg} \in \{7, 10, 14, 21, 28, 31\}$, as is the case for the ASP. It has to be stated explicitly that the choice of QP_{bg} is purely random, i.e. no optimization in terms of setting the best combination of QP_{bg} and QP_{fg} has been done. The prediction structure has been set to IPPP with a GOP size of 16 for sprite coding as well as for the ASP to ensure comparability. Quarter-pel motion vector accuracy has as well been enabled for both profiles.

Table 3 Test sequences used for comparison of MPEG-4 Visual sprite coding using various automatic foreground object segmentation approaches

Sequence	Source	Resolution	Frames	FPS [Hz]
Allstars	ZDF (German television)	352 × 288	250	25
BBC-Pan12	BBC (documentary *Planet Earth*)	720 × 576	185	25
BBC-Pan13	BBC (documentary *Planet Earth*)	720 × 576	110	25
Biathlon	ARD (German television)	352 × 288	200	25
Mountain	BBC (documentary *Planet Earth*)	352 × 192	100	25
Stefan	MPEG	352 × 240	300	30

(a) Single sprite for *BBC-Pan13*

(b) Multiple sprite 3/4 for *Biathlon*

(c) Single sprite for *All stars*

Fig. 17 Examples of background sprite images generated using the approaches from Sections 3.1 and 3.2

Figure 18 shows rate-distortion results for all test sequences used. It can be seen in all curves that the segmentation approach *Algorithm3*, i.e. segmentation based on background subtraction using local background sprites, outperforms all other segmentation techniques. Additionally, when multiple background sprites are used (cf. Figs. 18d, f), the performance is better than compared to using one single background sprite image as a model.

For the sequence *BBC-Pan12* (cf. Fig. 18b), the sprite coding approach performs worse than coding the sequence using MPEG-4 ASP. This can be explained with the content of the sequence. It shows a group of monkeys wading through water, which is moving and takes a large part of the video frame. Dynamic textures, e.g. water, have to be assumed as foreground objects in a video, since their movement differs from the global motion. Such a dynamic texture cannot be modeled correctly by a static background sprite. However, all segmentation approaches presented in this work define the water as background. Therefore, the decoded video frames differ strongly from the original sequence in these parts after reconstruction of background sprite

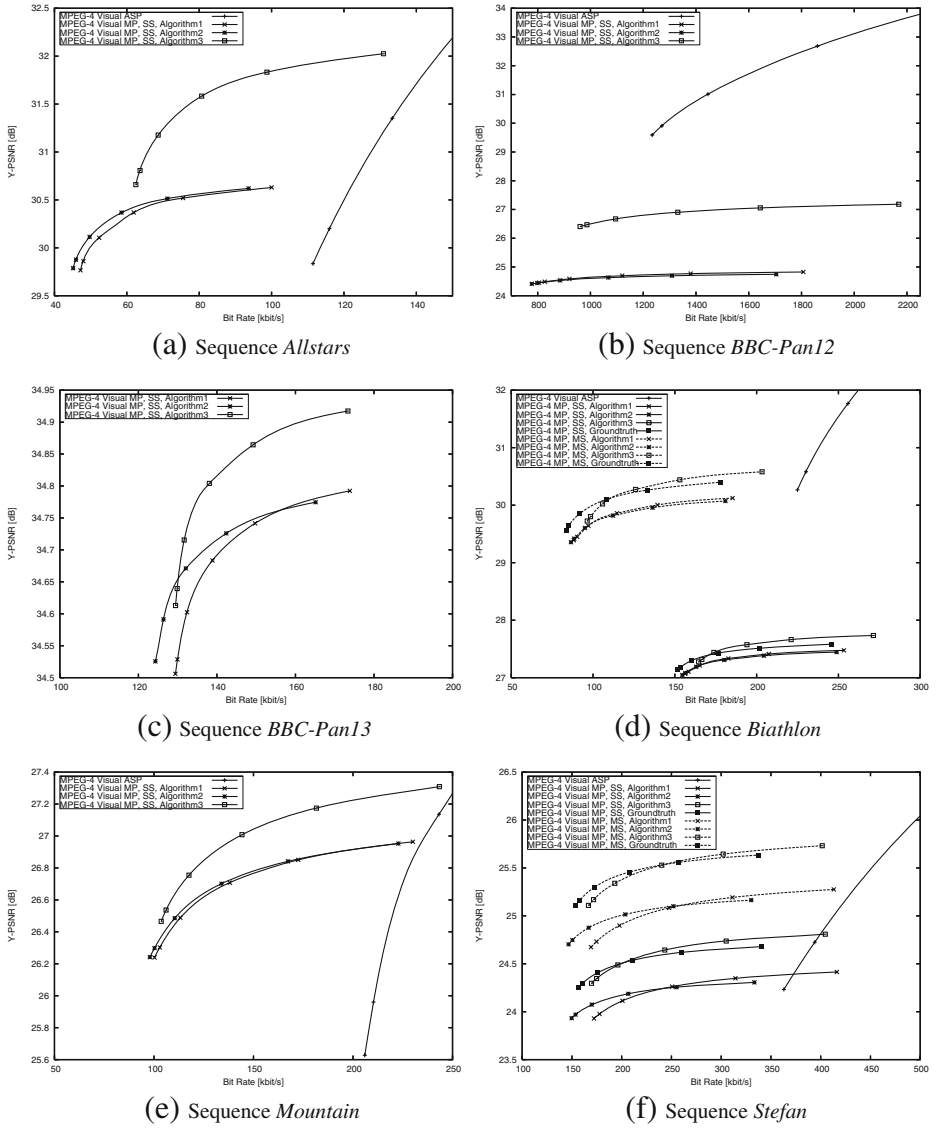


Fig. 18 Comparison of rate-distortion performance for coding using MPEG-4 Visual ASP with MPEG-4 Visual MP (sprite coding) applying different types of background sprites and segmentation algorithms

image and foreground object sequence. This results in a lower PSNR compared to MPEG-4 ASP, which correctly reconstructs the dynamic texture.

In Fig. 18c, the rate-distortion curves for the test sequence *BBC-Pan13* are shown. Here, no curve for coding the sequence with the MPEG-4 ASP is depicted, since the bit rate needed for coding this sequence using the ASP is in the range of about 800

kbit/s to 1,200 kbit/s, depending on the coarseness of quantization. The reason for the low bit rate range when using sprite coding compared to the ASP is the content of the sequence. The foreground objects, i.e. a group of flying birds, are very small. Therefore, the background model already shows nearly the complete sequence. In other words, the bit rate needed for coding the foreground object sequence is only one–two times the bit rate needed for coding the background model. E.g. for the sequence *Allstars*, the bit rate for the foreground object sequence is three-11 times higher than that needed for the background model, depending on the combination of quantization and segmentation approach. Therefore, the *BBC-Pan13* sequence is an excellent example of the capabilities of the sprite coding principle.

The usage of a single background sprite image for the sequence *Biathlon* (cf. Fig. 18d) did not perform as well as expected. It turned out that the generation of the background model introduced some errors. This can as well be explained with the content of the sequence. Figure 17c shows one of its multiple sprites, which gives an impression of its content. Most part of it is snow, i.e. homogeneous content. In background sprite generation, long-term motion compensation is performed. This means that a mapping of pixel content between temporally remote frames has to be done. This is challenging when large homogeneous areas are present and may lead to bad results, as is the case here. Additionally, the camera moves very fast, which makes background sprite image generation even harder. Due to these problems, the quality of the background model for this sequence is not very good, which leads to a bad rate-distortion performance.

6 Summary

We have proposed an automatic object segmentation algorithm, i.e. *Algorithm3*, for video sequences with a moving camera towards automatic object representation for sprite coding as standardized in MPEG-4 Visual. We compared this approach to using several other segmentation methods and showed that it always performs better than these. Additionally, it was shown that the new segmentation approach performs comparable to manually segmented groundtruth masks, meaning an almost perfect solution in terms of rate-distortion performance has been found. Using this approach, automatic sprite coding could become applicable.

References

1. Antonini M, Barlaud M, Mathieu P, Daubechies I (1992) Image coding using wavelet transform. IEEE Trans Image Process 1(2):205–220
2. Baker S, Matthews I (2001) Equivalence and efficiency of image alignment algorithms. In: Computer vision and pattern recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE computer society conference, vol 1, pp I-1090–I-1097
3. Baker S, Matthews I (2004) Lucas-Kanade 20 years on: a unifying framework. Int J Comput Vis 56(3):221–255
4. Calderbank AR, Daubechies I, Sweldens W, Yeo B-L (1997) Lossless image compression using integer to integer wavelet transforms. In: Image processing, 1997. Proceedings, international conference on Oct 1997, vol 1, pp 596–599
5. Daubechies I (1992) Ten lectures on wavelets. Society for Industrial Mathematics, Philadelphia

6. Dufaux F, Konrad J (2000) Efficient, robust, and fast global motion estimation for video coding. IEEE Trans Image Process 9(3):497–501
7. Farin D (2005) Automatic video segmentation employing object/camera modeling techniques. Ph.D. thesis, Eindhoven University of Technology
8. Farin D, de With PHN, Effelsberg W (2004) Minimizing MPEG-4 sprite coding-cost using multi-sprites. In: SPIE visual communications and image processing
9. Farin D, de With PHN, Effelsberg WA (2004) Video-object segmentation using multi-sprite background subtraction. In: Multimedia and expo, 2004. ICME '04. 2004 IEEE international conference, Jun 2004, vol 1, pp 343–346
10. Farin D, Haller M, Krutz A, Sikora T (2008) Recent developments in panoramic image generation and sprite coding. In: Proc. IEEE 10th workshop on multimedia signal processing, Oct 2008, pp 64–69
11. Fischler MA, Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun ACM 24(6):381–395
12. Haller M, Krutz A, Sikora T (2009) Evaluation of pixel- and motion vector-based global motion estimation for camera motion characterization. In: Proceedings of the international workshop on image analysis for multimedia interactive services (WIAMIS 2009). London, UK, ISBN 978-1-4244-3610-1, pp 49–52
13. Krutz A, Frater M, Kunter M, Sikora T (2006) Windowed image registration for robust mosaicing of scenes with large background occlusions. In: Proc. IEEE international conference on image processing, Oct 2006, pp 353–356
14. Krutz A, Glantz A, Frater M, Sikora T (2008) Local background sprite generation. In: International workshop on local and non-local approximation in image processing, LNLA 2008. IEEE, EURASIP, Tampere International Center for Signal Processing (TICSP), EUSIPCO2008, IEEE, EURASIP, Lausanne, Switzerland, Aug 2008
15. Krutz A, Glantz A, Sikora T, Nunes P, Pereira F (2008) Automatic object segmentation algorithms for sprite coding using MPEG-4. In: Grgic S, Grgic M (ed) 50th international symposium ELMAR-2008, September, Proceedings, Zadar, Croatia, vol 2 of 2. Department of Wireless Communications, Faculty of Electrical Engineering and Computering, University of Zagreb, Croatia, ELMAR, Zadar, ELMAR, Croatian Society Electronics in Marine, Zadar, pp 459–462
16. Krutz A, Glantz A, Borgmann T, Frater M, Sikora T (2009) Motion-based object segmentation using local background sprites. In: Proceedings of the IEEE international conference on acoustics, speech and signal processing (ICASSP 2009), Taipei, Taiwan, Apr 2009
17. Krutz A, Kunter M, Dröse M, Frater M, Sikora T (2007) Content-adaptive video coding combining object-based coding and H.264/AVC. In: Picture coding symposium, Lisbon, Portugal, Nov 2007
18. Krutz A, Kunter M, Sikora T, Mandal M, Frater M (2007) Motion-based object segmentation using sprites and anisotropic diffusion. In: Proc. eighth international workshop on image analysis for multimedia interactive services WIAMIS '07, Jun 2007, pp 35–35
19. Kunter M (2008) Advances in sprite-based video coding—towards universal usability. Ph.D. thesis, Technische Universität Berlin
20. Kunter M, Krutz A, Mandal M, Sikora T (2007) Optimal multiple sprite generation based on physical camera parameter estimation. In: Visual communications and image processing, VCIP, electronic imaging 2007, San José, CA, USA, Jan 2007
21. Kunter M, Krutz A, Dröse M, Frater M, Sikora T (2007) Object-based multiple sprite soding of unsegmented videos using H.264/AVC. In: IEEE int. conf. on image processing (ICIP 2007), San Antonio, Texas, USA, Sep 2007
22. Lucas BD, Kanade T (1981) An iterative image registration technique with an application to stereo vision. In: International joint conference on artificial intelligence, Aug 1981, pp 674–679
23. Mech R, Wollborn M (1997) A noise robust method for segmentation of moving objects in video sequences. In: Proc. IEEE international conference on acoustics, speech, and signal processing ICASSP-97, vol 4, pp 2657–2660
24. Pereira F, Ebrahimi T (2002) The MPEG-4 book. Prentice Hall, Upper Saddle River
25. Perona P, Malik J (1990) Scale-space and edge detection using anisotropic diffusion. IEEE Trans Pattern Anal Mach Intell 12(7):629–639
26. Sikora T (1997) The MPEG-4 video standard verification model. IEEE Trans Circuits Syst Video Technol 7(1):19–31
27. Sikora T (2005) Trends and perspectives in image and video coding. In: Proc. of the IEEE, vol 93, pp 6–17

28. Shi J, Tomasi C (1994) Good features to track. In: Proc. CVPR '94. IEEE computer society conference on computer vision and pattern recognition, Jun 1994, pp 593–600
29. Smolic A, Sikora T, Ohm J-R (1999) Long-term global motion estimation and its application for sprite coding, content description, and segmentation. IEEE Trans Circuits Syst Video Technol 9(8):1227–1242
30. Tomasi C, Kanade T (1991) Detection and tracking of point features. Tech. Rep., Technical Report CMU-CS-91-132, Carnegie Mellon University, Apr 1991
31. Wiegand T, Sullivan GJ, Bjontegaard G, Luthra A (2003) Overview of the H.264/AVC video coding standard. IEEE Trans Circuits Syst Video Technol 13(7):560–576

Alexander Glantz received the Dipl.-Ing. degree in computer engineering from the Technische Universität Berlin, Germany.

He is currently working toward the Ph.D. degree at the Communication Systems Group at Technische Universität Berlin, Germany.

He has been involved in the European Networks of Excellence (NoE) VISNET II and PetaMedia. His main research interests include multimedia signal processing, hybrid video coding, rate-distortion theory, object-based video coding and global motion estimation. He is a student member of the IEEE.

Andreas Krutz received the Dipl.-Ing. degree in electrical engineering from the Technische Universität Berlin, Germany in 2006. He was a student research assistant at the University College, University of New South Wales, Australian Defence Force Academy, Canberra, Australia, where he started to research in fields of image processing and video coding.

Andreas Krutz is currently a research assistant at the Communication Systems Group at Technische Universität Berlin and he works towards his Ph.D. During this time, he has been involved in a number of European Networks of Excellence (NoE), i.e. 3DTV, VISNET II, Kspace, PetaMedia. His research interests are topics in image and video processing, analysis, and coding. He is a member of the IEEE.

Thomas Sikora is professor and director of the Communication Systems Group at Technische Universität Berlin, Germany.

He received the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering from Bremen University, Bremen, Germany, in 1985 and 1989, respectively. In 1990, he joined Siemens Ltd. and Monash University, Melbourne, Australia, as a Project Leader responsible for video compression research activities in the Australian Universal Broadband Video Codec consortium. Between 1994 and 2001, he was the Director of the Interactive Media Department, Heinrich Hertz Institute (HHI) Berlin GmbH, Germany. Dr. Sikora is co-founder of 2SK Media Technologies and Vis-a-Pix GmbH, two Berlin-based start-up companies involved in research and development of audio and video signal processing and compression technology.

Dr. Sikora has been involved in international ITU and ISO standardization activities as well as in several European research activities for a number of years. As the Chairman of the ISO-MPEG (Moving Picture Experts Group) video group, he was responsible for the development and standardization of the MPEG-4 and MPEG-7 video algorithms. He also served as the chairman of the European COST 211ter video compression research group. He was appointed as Research Chair for the VISNET and 3DTV European Networks of Excellence. He is an Appointed Member of

the Advisory and Supervisory board of a number of German companies and international research organizations. He frequently works as an industry consultant on issues related to interactive digital audio and video.

Dr. Sikora is a Member of the German Society for Information Technology (ITG) and a recipient of the 1996 ITG Award. He has published more than 150 papers related to audio and video processing. He was the Editor-in-Chief of the IEEE Transactions on Circuits and Systems for Video Technology. From 1998 to 2002, he was an Associate Editor of the IEEE Signal Processing Magazine. He is an Advisory Editor for the EURASIP Signal Processing: Image Communication journal and was an Associate Editor of the EURASIP Signal Processing journal.

Paulo Nunes received the B.S., M.Sc. and Ph.D. degrees in electrical and computer engineering from the Instituto Superior Técnico (IST), Universidade Técnica de Lisboa, Portugal, in 1992, 1996, and 2007, respectively.

He is currently a Professor at the Information Science and Technology Department, Instituto Superior de Ciências do Trabalho e da Empresa (ISCTE), Portugal, and a member of the Research Staff of the Telecommunications Institute, Portugal. His current research interests include object-based and frame-based rate control and video coding.

Dr. Nunes has participated in various European projects and has acted as project evaluator for the European Commission. He acts often as reviewer for various conferences and journals and has contributed more than 30 papers.

Fernando Pereira received the B.S., M.Sc. and Ph.D. degrees in electrical and computer engineering from the Instituto Superior Técnico (IST), Universidade Técnica de Lisboa, Portugal, in 1985, 1988, and 1991, respectively.

He is currently a Professor with the Electrical and Computer Engineering Department, Instituto Superior Técnico (IST), Universidade Técnica de Lisboa, Portugal, where he is responsible for the participation of IST in many national and international research projects. He acts often as project evaluator and auditor for various organizations. His areas of interest are video analysis, processing, coding and description, and interactive multimedia services. He is an Area Editor of Signal Processing: Image Communication.

Dr. Pereira is or has been an Associate Editor of the IEEE TRANSACTIONS OF CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, the IEEE TRANSACTIONS on IMAGE PROCESSING, the IEEE TRANSACTIONS ON MULTIMEDIA, and the IEEE Signal Processing Magazine. He is a member of the IEEE Signal Processing Society Image and Multiple Dimensional Signal Processing Technical Committee and of the IEEE Signal Processing Society Multimedia Signal Processing Technical Committee. He was an IEEE Distinguished Lecturer in 2005. He has been a member of the Scientific and Program Committees of many international conferences and has contributed more than 200 papers. He has been participating in the work of ISO/MPEG for many years, notably as the head of the Portuguese delegation, Chairman of the MPEG Requirements Group, and chairing many Ad Hoc Groups related to the MPEG-4 and MPEG-7 standards.