Video Compression with 3-D Pose Tracking, PDE-based Image Coding, and Electrostatic Halftoning

Christian Schmaltz and Joachim Weickert

Mathematical Image Analysis Group, Faculty of Mathematics and Computer Science, Building E1 7, Saarland University, 66041 Saarbrücken, Germany
{schmaltz,weickert}@mia.uni-saarland.de

Abstract. Recent video compression algorithms such as the members of the MPEG or H.26x family use image transformations to store individual frames, and motion compensation between these frames. In contrast, the video codec presented here is a model-based approach that encodes fore- and background independently. It is well-suited for applications with static backgrounds, i.e. for applications such as traffic or security surveillance, or video conferencing. Our video compression algorithm tracks moving foreground objects and stores the obtained poses. Furthermore, a compressed version of the background image and some other information such as 3-D object models are encoded. In a second step, recent halftoning and PDE-based image compression algorithms are employed to compress the encoding error. Experiments show that the stored videos can have a significantly better quality than state-of-the-art algorithms such as MPEG-4.

1 Introduction

Due to the huge amount and increasing resolution of videos that are created and viewed each day, video compression remains a topic of ongoing research. Most popular video compression algorithms such as the ones from the MPEG and H.26x family calculate the motion of pixel blocks to estimate the appearance of these blocks from nearby frames. In addition to the estimated displacement, an approximation of the reconstruction error is stored. In the decoding phase, this information is used to reconstruct individual frames. For a detailed introduction to general video codecs, we refer to the survey by Sullivan and Wiegand [19], or the overview of Abomhara *et al.* [1].

Apart from general purpose video compression algorithms, there are also video compression codecs using *model-based* coding schemes. The idea behind these schemes is to compress fore- and background independently. This concept is fundamentally different from standard video compression algorithms, and thus has different advantages and drawbacks. For example, model-based coding typically requires previous knowledge to distinguish fore- and background region. Furthermore, the video sequence must have a fairly static background. However,

several views of the same scene can often be encoded very efficiently, and generating intermediate frames is often much easier. Examples in which these conditions are fulfilled include traffic or security surveillance, or video conferencing.

Since the seminal work by Forchheimer and Fahlander [6], different approaches to model-based video coding have been pursued. We will give a short summary of the ideas presented in this context, but refer to [14] and [23] for a more detailed overview of the field.

In [20], Toelg and Poggio propose an approach that uses a small set of example images containing a human face with different facial expressions. With the help of a pose estimation algorithm, a novel view or facial expression is constructed from these example images. Vieux *et al.* employ a similar approach for their "Orthonormal Basis Coding" in [21].

These approaches are based on 2-D example images, but there are also methods that utilise full 3-D models. In [10], a partial description of a model-based coding which builds upon the MPEG-4 standard is presented. However, this method requires manual interaction, and it is not specified how the necessary texture is stored. Although these questions are answered in the work by Granai *et al.* [7], both methods only explain how to compress the foreground, and ignore the background. In [2], motion compensated temporal interpolation is used to estimate the background onto which the 3-D object model is projected. Various extensions to model-based coding have been proposed, e.g. for varying illumination conditions [5] or for different facial expressions [4, 13]. The latter was even included into MPEG-4 as facial animation parameters [12].

The model-based video compression codec we propose differs fairly much from these existing approaches. It combines three state-of-the-art algorithms from apparently unrelated fields, namely 3-D pose tracking, PDE-based image compression, and halftoning. As illustrated in Section 4, their combination makes it possible to beat the results of MPEG-1, and even of MPEG-4. In contrast to many other model-based coding algorithms, our approach is not specialised to faces or other specific objects. Thus, it is applicable for different kinds of videos.

Our paper is structured as follows: Section 2 explains our baseline video compression algorithm (*MB*), which is extended to an algorithm with residual coding (*MB+DH*) in Section 3. We continue with an evaluation of both approaches in Section 4 and conclude the paper with a summary in Section 5.

2 Our Baseline Codec (MB)

Before we explain the steps of our codec in detail, lets us give an overview of our algorithm: First, we track the moving objects in the video. As a second step, the tracking results are used to estimate the colour of each vertex of the object model. Thirdly, the background is reconstructed, if necessary, and compressed. Finally, all data is saved and compressed using PAQ [11], a general purpose entropy coder. To reconstruct a frame of the video, the object model is simply projected onto the loaded background image using the pose tracked while encoding. We denote this *m*odel *b*ased codec by *MB*.

For the first step of our codec, we employ the 3-D pose tracking algorithm explained in [16], as it reports one of the best tracking results in the HumanEva-II benchmark [18]. Assuming that the necessary data (a projection matrix of each view, an (uncoloured) 3-D object model, and a pose initialisation in the first frame) are known, we find the pose of the free-form surface consisting of rigid parts interconnected by n predefined joints as minimiser of the cost function

$$E(\boldsymbol{\chi}) = -\sum_{i=0}^{\ell} \int_{\Omega} \Big(Pv_{i,\boldsymbol{\chi}}(\boldsymbol{x}) \log p_{i,\boldsymbol{\chi}}(\boldsymbol{x}) \Big) \mathrm{d}\boldsymbol{x} \ . \tag{1}$$

Thereby, the pose $\boldsymbol{\chi} \in \mathbb{R}^{6+n}$ consists of the 3-D position and orientation of the object model, as well as of the n joint angles (or other internal parameters) searched for. The index i runs over the background ($i = 0$) and all ℓ model components. The set Ω denotes the 2-D image domain, while the function $p_{i,\boldsymbol{\chi}}(\boldsymbol{x})$ models the appearance of the i-th component. These appearances are estimated and adapted while tracking. The indicator function $Pv_{i,\boldsymbol{\chi}}(\boldsymbol{x})$, which is 1 if the i-th model component is visible at the image point $\boldsymbol{x}$ and 0 otherwise, ensures that occlusions are taken into account in an adequate way. Even if model components belong to different object models, all occlusions are automatically handled correctly. Thus, even tracking multiple mutually occluding object is possible. This is favourable for our codec in case of multiple moving foreground objects.

Equation 1 is minimised with a modified gradient descent: The object model is projected onto the image plane, and the resulting silhouette points are displaced depending on to which region they fit better. This displacement is then transferred to the 3-D pose of the object. These steps are repeated until convergence, and the pose initialisation of the next frame is obtained by extrapolation.

In the second step of the *MB* codec, we estimate the appearance of the object model. As the tracking algorithm requires an (uncoloured) object model, we already know to which image point each vertex of the object is projected in each frame. Thus, to estimate the colour of each vertex, we simply average the colour at the projected vertex position over all frames in which this vertex is visible. This simple estimation is far from being perfect, though. Consequently, the obtained video quality should improve significantly if a better estimation is used.

In the third step, we reconstruct the background image, if necessary. This step is easy in our setting as we know which parts of the background are occluded after the tracking step.

Then, we employ the PDE-based image compression algorithm from [17] to encode the background image. We chose this algorithm as is reports better a compression quality than JPEG 2000. Moreover, it is related to the approach to store the residual image introduced in the next section.

The basic idea behind the algorithm from [17] is to store only a small subset of all image points, while the remaining points are reconstructed using edge-enhancing anisotropic diffusion (EED) [22], i.e. by computing the steady-state $\inf_{t\to\infty} u(\boldsymbol{x}, t)$ of the evolution equation

$$\partial_t u = \operatorname{div}(g(\boldsymbol{\nabla} u_\sigma \boldsymbol{\nabla} u_\sigma^\top)\boldsymbol{\nabla} u) \ . \tag{2}$$

Here, $u = u(\boldsymbol{x}, t)$ is the image value of the point $\boldsymbol{x}$ at time t, $u_\sigma := K_\sigma * u$ denotes the image convolved with a Gaussian K_σ with standard deviation σ, and g is the Charbonnier diffusivity function $g(s^2) := \frac{\lambda}{\sqrt{\lambda^2+s^2}}$ with contrast parameter λ. The diffusion tensor $g(\boldsymbol{\nabla} u_\sigma \boldsymbol{\nabla} u_\sigma^\top)$ is a symmetric 2×2 matrix with eigenvectors parallel and orthogonal to $\boldsymbol{\nabla} u_\sigma$, and corresponding eigenvalues $g(|\boldsymbol{\nabla} u_\sigma|^2)$ and 1. Since EED smoothes along edges, while reducing smoothing across them, this diffusion process is able to create sharp edges.

This concludes the description of our baseline codec, which is often sufficient to yield a reasonable reconstruction. However, tracking failures or model inaccuracies can sometimes result in a bad video quality. Thus, we introduce an algorithm that can correct such problems in the next section.

3 Video Codec with Residual Coding (*MB+DH*)

Our enhanced codec explained in this section is an extension of the *MB* codec. It additionally encodes the residual images, i.e. the error of each frame compressed by our baseline codec. This residual image is stored as set of pixels between which inpainting with homogeneous diffusion is performed. Therefore, we compute the steady-state of the linear diffusion equation [8]

$$\partial_t u = \Delta u = \operatorname{div}(\boldsymbol{\nabla} u) \ . \tag{3}$$

Let us start by considering only the first frame of a grey-valued video. When inpainting with homogeneous diffusion, we know that the interpolation points should be distributed according to the magnitude of the Laplacian of a smoothed version of the image [3]. Thus, we can employ a dithering algorithm to obtain the inpainting mask. In [3], the Floyd-Steinberg algorithm was used for dithering, while we compare the performance of four different dithering algorithm. Two representative results are shown in Figure 1. In these experiment, we use 500 mask points, but results are similar for other numbers: Independent of the image and the amount of presmoothing, the electrostatic halftoning algorithm from [15] performs best. Thus, we chose this algorithm in our codec.

The basic idea behind the electrostatic halftoning algorithm is to model black dots as negatively charged particles, while the pixels are positively charged [15]. Consequently, particles repel each other, but are attracted to dark image areas. Let us denote the grey value at position $\boldsymbol{x}$ by $u(\boldsymbol{x}) \in [0, 1]$. Then, the charge of the pixel $\boldsymbol{x}$ is equal to $1 - u(\boldsymbol{x})$. When choosing the charge of the particles is such a way that the total amount of positive and negative charges is equal, the particles are automatically bound to the image domain. The final halftoning result is then obtained as the steady-state of this particle system. Adding all forces acting on each particle results in the update equation

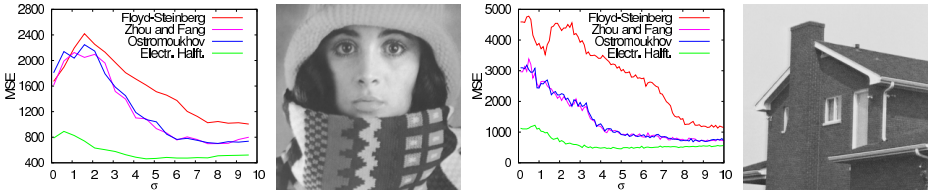

Fig. 1. Evaluation of dithering algorithms in the context of image interpolation. The mask points are obtained by dithering the (scaled) absolute value of the Laplacian of the smoothed input image, where a Gaussian with standard deviation σ is used for smoothing. The graphs show the results for the two images "trui" and "house".

$$\boldsymbol{p}_n^{k+1} = \boldsymbol{p}_n^k + \tau\Big(\sum_{\substack{\boldsymbol{x}\in\Omega \\ \boldsymbol{x}\neq\boldsymbol{p}_n}} \frac{1-u(\boldsymbol{x})}{|\boldsymbol{x}-\boldsymbol{p}_n^k|}\boldsymbol{e}_{n,x} - \sum_{\substack{m\in\mathcal{P} \\ m\neq n}} \frac{1}{|\boldsymbol{p}_m^k-\boldsymbol{p}_n^k|}\boldsymbol{e}_{n,m}\Big), \tag{4}$$

where $\boldsymbol{p}_n^k$ is the position of the n-th particle at time k, $\tau = 0.1$ serves as an artificial time step parameter, and $\mathcal{P}$ denotes the set of all particles. The two vectors $\boldsymbol{e}_{n,m}$ and $\boldsymbol{e}_{n,x}$ denote unit vectors between the n-th and m-th particle, and between the n-th particle and the pixel $\boldsymbol{x}$, respectively.

The optimal standard deviation σ of the Gaussian used to smooth the residual image before dithering varies with the image and the number of mask points. Thus, we try different standard deviations in our codec and take the σ for which the best approximation is obtained. As σ is not needed to decompress the video, this does not increase the final file size. Furthermore, we restrict the domain of the dithering algorithm to a region containing the foreground region and points close to it, as inaccuracies in the background region are easier solved by storing an improved version of the background image.

We store the position of the points in the inpainting mask K using the JBIG file format [9], which is a lossless compression algorithm for binary images. The grey-values of the mask points are quantised uniformly before entropy coding.

For colour videos, we compute a grey-valued variant of the difference image to find the inpainting mask. Thereby, different colour models are possible. According to our experiments, the results are very similar, though. Thus, we simply average the red, green, and blue colour channels to get a grey-valued variant of the difference image.

In the remaining frames, we initialise the dithering process with the inpainting mask from the previous frame. This requires much fewer iterations of electrostatic halftoning. In addition to speeding up the computations, this allows to store the particle movements relative to the last frame instead of the particle positions. This reduces the amount of data that must be encoded if the number of particles is reasonable. While it is trivial to obtain the particle motion when using electrostatic halftoning, this is a difficult or even impossible problem with

	HumanEva-II S4			Cart		
Codec	frame size	file size	MSE	frame size	file size	MSE
MPEG-1	656 × 480	2019733	187.25	496 × 368	202847	48.53
MPEG-4	656 × 490	537404	210.58	500/496 × 380	112182	31.52
MB	656 × 490	161223	102.57	500 × 380	68721	52.38
MB+DH (400/200)	656 × 490	494246	48.09	500 × 380	109137	41.55
MB+DH (100)	656 × 490	194513	76.53	500 × 380	83355	47.04
MB+DH (500)	656 × 490	612452	43.66	500 × 380	196465	32.77
MB+DH (1000)	656 × 490	1256973	30.27	500 × 380	353026	26.64

Table 1. Overview over the frame and file sizes, as well as the mean square errors (MSE) of results with MPEG-1, MPEG-4, and the proposed algorithms. The numbers in parenthesis state the number of additional points stored per frame. In line 4, this number was chosen in such a way that the file size is similar to MPEG-4. Note that the codecs from the MPG-family cropped some of the video material.

other dithering algorithms. This is another reason why it is advantageous to use the electrostatic halftoning algorithm as dithering step in our codec.

Finally, all data is compressed with the same entropy coder as in the baseline codec. We denote this *m*odel-*b*ased codec which stores the *d*ifference image by *h*alftoning as *MB+DH*.

To reconstruct the video, we first execute the steps explained for our baseline codec. Afterwards, the inpainting mask is loaded (first frame) or reconstructed using the stored particle motion and the particle locations in the preceding frame. The loaded values of the error image are interpolated and added to the frame. Thereby, we use Dirichlet boundary conditions to ensure that the difference image is zero at the boundary.

4 Experiments

In this section, we compare the performance of our codecs against MPEG-1 and MPEG-4. We created the MPEG-1 videos with the program "mpeg_encode" using three P-frames between successive I-frames, and three B-frames between other frames, i.e. the pattern "IBBBPBBBPBBBPBBB". This is a common pattern which typically allows a strong compression. The quantisation levels for all types of frames were set to the lowest possible value (31) to obtain a compression ratio which is as close as possible to the one of our approach. To create the MPEG-4 videos, we used the Linux program "mencoder", the codec "msmpeg4v2", the AVI container format, and the smallest possible variable bit rate (4000 bits per second). Nevertheless, we could neither create MPEG-1 nor MPEG-4 videos which are as small as the ones from our baseline codec MB (see Table 1). Note that MPEG-1 cropped the original frames, while MPEG-4 replaced a part of the frames in one sequence by a black boundary.

In our first experiment, we encode the sequence S4 of the HumanEva-II tracking benchmark [18]. Figure 2 illustrates an example frame from the resulting

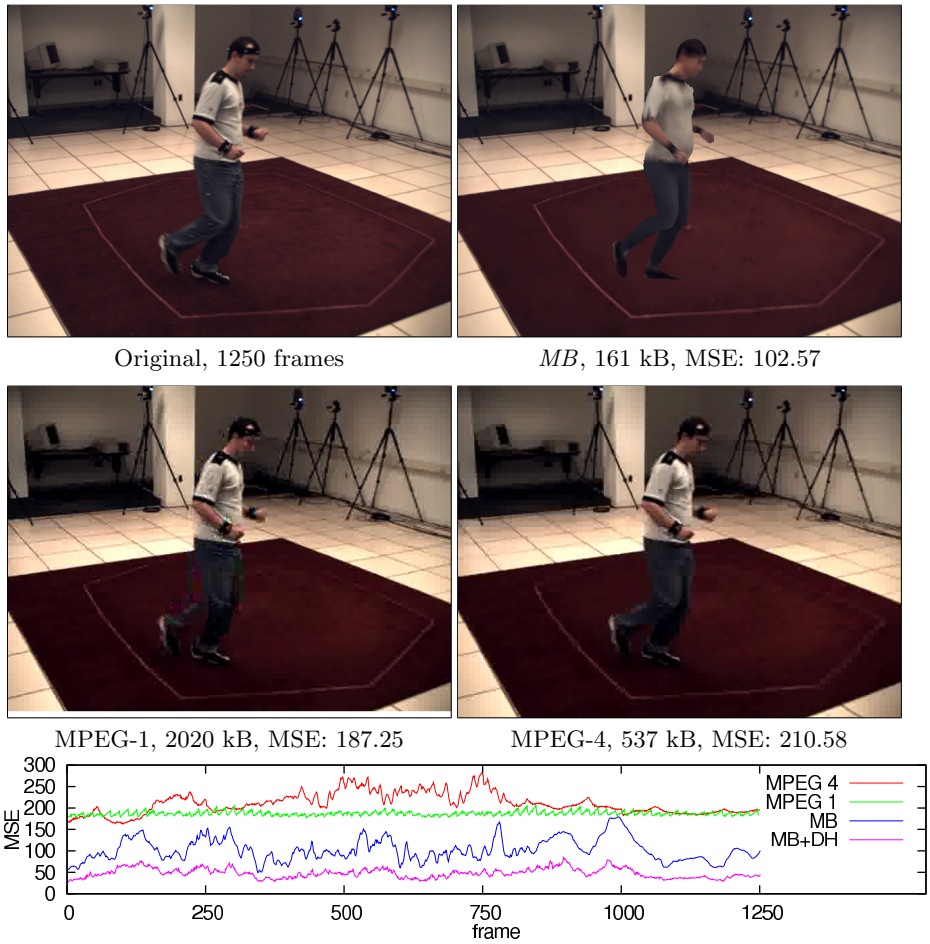

Fig. 2. Comparison of our *MB* codec against MPEG-1 and MPEG-4 using the HumanEva-II sequence S4. In the graph showing the mean square error in each frame, the result of our *MB+DH* codec with 400 point per frame is shown as comparison. The corresponding files sizes are denoted below the images, which show frame 500.

Fig. 3. Magnifications of the experiment from Figure 2. Note the block artifacts when using MPEG-1 or MPEG-4. One can see that our approach has sharp boundaries in object and background region, and that our simple model colouring algorithm is far from being perfect. The better result of our algorithm *MB+DH* with 400 additional points, which tries to reduce this problem, is shown on the right.

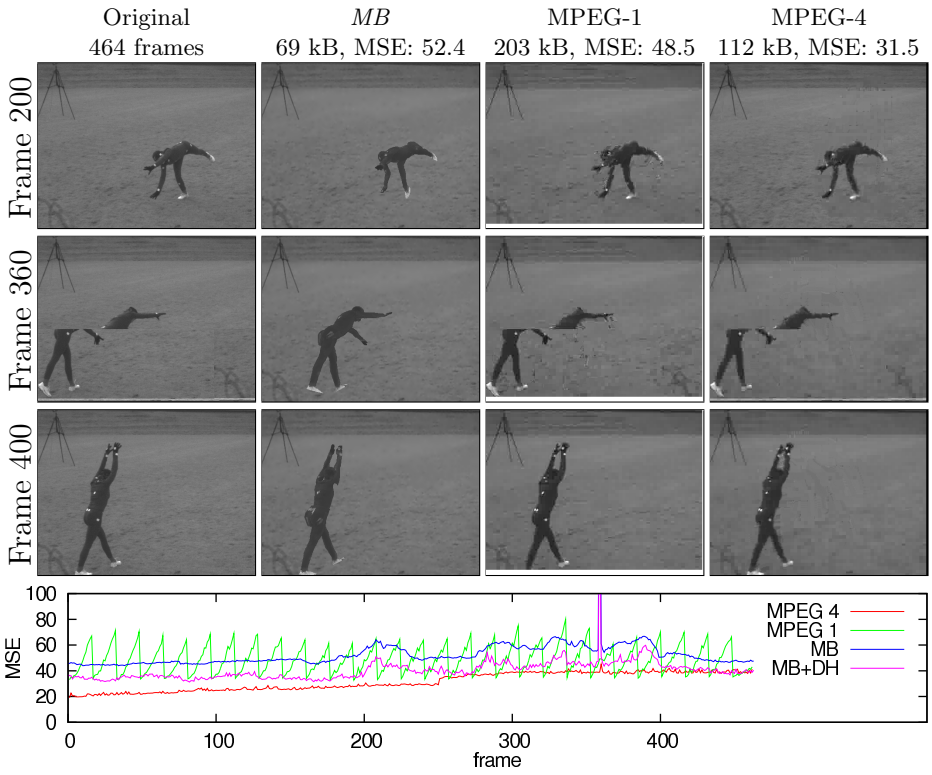

Fig. 4. Comparison of our method *MB* against MPEG-1 and MPEG-4 using the sequence "Cart". The jump with MPEG-4 in frame 250 and the sawtooth pattern with MPEG-4 are due to the different frame types. While our codec *MB* ignores the corruption in frame 360, MPEG-1 and MPEG-4 encode the original frame.

videos, as well as a graph showing the mean square error (MSE) each method obtained per frame. The video created with our approach is considerably smaller than those of the other methods. Nevertheless, its error is always below that of MPEG-1 and MPEG-4, even though the result of the tracking approach is inaccurate in some frames.

Figure 3 shows magnifications of the results depicted in Figure 2. We see that our *MB* codec creates sharp boundaries, while the approaches from the MPG family generate blocky results. Due to the rather poor performance of our simple model colouring approach from in Section 2, the *MB* codec yields suboptimal results at the sleeves, though. As shown on the right, this is improved by the additional information stored. A more accurate representation of the object model should significantly boost the performance of our algorithm, though.

Furthermore, we encode the sequence "Cart", in which a person performs a cart wheel; see Figure 4. This sequence is much more challenging than the first video for our codec due to several reasons: First of all, the background is very

noisy, which deteriorates the results of the diffusion-based image compression approach. Moreover, the object model is often not able to represent the complex movement performed by the actor, e.g. due to muscle contractions or missing joint angles. Additionally, the lower side of the feet are visible in many frames. Since the feet are not included in the object model, the human is partly seen from the inside, which results in wrong colours. Finally, this sequence is shorter than the HumanEva-II sequence, resulting in a larger overhead for object model and background. Due to these reasons, the *MB* algorithm is worse than MPEG-4 for this sequence; see Figure 4. However, we still beat MPEG-1 in most frames even though the file created with our approach is significantly smaller.

The high error of our codec in frame 360 is due to the fact that this input frame is corrupted. While MPEG-1 and MPEG-4 encode the corrupted frame, the codec *MB* stores a "corrected" version. This may even be seen as advantage of our algorithm, since it automatically corrected the corrupted frame.

Table 1 also shows results of the codec *MB+DH*. In particular, we compare our codec against MPEG-4 with similar file sizes. For the HumanEva-II sequence S4, this results in 400 additional particles, while 200 additional particles are used for the "Cart" sequence. We are still slightly worse than MPEG-4 in the "Cart" sequence, but one can see that we clearly outperform MPEG-4 in the HumanEva-II sequence S4: Even though the video created with MPEG-4 is 8% larger, its MSE is about 4.4 times as large as the one with our approach.

5 Summary

We have demonstrated how to combine recent state-of-the-art methods from PDE-based image compression, 3-D pose tracking, and halftoning into a model-based video compression codec. Our algorithms show promising results that can beat those of MPEG-1, and even of MPEG-4. Moreover, we are optimistic that the performance of our approach can be significantly improved when the appearance of the moving foreground objects is estimated more accurately, or is even known in advance. This would not only enhance the approximation obtained by projecting the object model, but can also help to improve the results of the tracking algorithm. A detailed evaluation is part of our future work.

References

1. Abomhara, M., Khalifa, O.O., Zakaria, O., Zaidan, A., Zaidan, B., Rame, A.: Video compression techniques: An overview. Journal of Applied Sciences 10(16), 1834–1840 (2010)
2. Artigas, X., Torres, L.: A model-based enhanced approach to distributed video coding. In: Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2005). Article No. 1127 (2005)
3. Belhachmi, Z., Bucur, D., Burgeth, B., Weickert, J.: How to choose interpolation data in images. SIAM Journal on Applied Mathematics 70(1), 333–352 (2009)
4. Eisert, P., Girod, B.: Facial expression analysis for model-based coding of video sequences. In: Proc. Picture Coding Symposium. pp. 33–38. Berlin (1997)

5. Eisert, P., Girod, B.: Model-based coding of facial image sequences at varying illumination conditions. In: Proc. 10th Image and Multidimensional Digital Signal Processing Workshop. pp. 119–122. Alpbach (1998)
6. Forchheimer, R., Fahlander, O.: Low bit-rate coding through animation. In: Proceedings of Picture Coding Symposium. pp. 113–114 (March 1983)
7. Granai, L., Vlachos, T., Hamouz, M., Tena, J.R., Davies, T.: Model-based coding of 3D head sequences. In: Proc. 3DTV Conference. IEEE Computer Society Press (2007)
8. Iijima, T.: Basic theory on normalization of pattern (in case of typical one-dimensional pattern). Bulletin of the Electrotechnical Laboratory 26, 368–388 (1962), in Japanese
9. ISO/IEC: Information technology – lossy/lossless coding of bi-level images (2001), ISO/IEC 14492. Latest corrections in 2004
10. Javůrek, R.: Model based facial video sequences coding. In: Radioelektronika 2003 – Conference Proceedings. pp. 115–118 (2003)
11. Mahoney, M.: Data compression programs. http://mattmahoney.net/dc/ (2009), last visited November 30, 2009
12. Pandzic, I.S., Forchheimer, R. (eds.): MPEG-4 Facial Animation: The Standard, Implementation and Applications. Wiley, New York (2003)
13. Pardàs, M., Bonafonte, A.: Facial animation parameters extraction and expression detection using hidden markov models. In: Signal Processing: Image Communication. vol. 17, pp. 675–688 (2002)
14. Pearson, D.E.: Developments in model-based video coding. Proceedings of the IEEE 83(6), 892–906 (1995)
15. Schmaltz, C., Gwosdek, P., Bruhn, A., Weickert, J.: Electrostatic halftoning. Computer Graphics Forum 29(8), 2313–2327 (Dec 2010)
16. Schmaltz, C., Rosenhahn, B., Brox, T., Weickert, J.: Localised mixture models in region-based tracking. In: Denzler, J., Notni, G., Süße, H. (eds.) Pattern Recognition. Lecture Notes in Computer Science, vol. 5748, pp. 21–30. Springer, Berlin (2009)
17. Schmaltz, C., Weickert, J., Bruhn, A.: Beating the quality of JPEG 2000 with anisotropic diffusion. In: Denzler, J., Notni, G., Süße, H. (eds.) Pattern Recognition. Lecture Notes in Computer Science, vol. 5748, pp. 452–461. Springer, Berlin (2009)
18. Sigal, L., Balan, A.O., Black, M.J.: HUMANEVA: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. International Journal of Computer Vision 87(1/2), 4–27 (Mar 2010)
19. Sullivan, G.J., Wiegand, T.: Video compression – from concepts to the H.264/AVC standard. Proceedings of the IEEE 93(1), 18–31 (January 2005)
20. Toelg, S., Poggio, T.: Towards an example-based image compression architecture for video-conferencing. Tech. Rep. AIM-1494, Massachusetts Institute of Technology, Cambridge, MA, USA (1994)
21. Vieux, W.E., Schwerdt, K., Crowley, J.L.: Face-tracking and coding for video compression. In: Christensen, H.I. (ed.) Computer Vision Systems. pp. 151–161. No. 1542 in Lecture Notes in Computer Science, Springer, Berlin (1999)
22. Weickert, J.: Theoretical foundations of anisotropic diffusion in image processing. Computing Supplement 11, 221–236 (1996)
23. Yao, Z.: Model-based Coding – Initialization, Parameters Extraction and Evaluation. Ph.D. thesis, Department of Applied Physics and Electronics, Umeå University, Sweden (January 2005)