MODEL-BASED FREE-VIEWPOINT VIDEO: ACQUISITION, RENDERING, AND ENCODING

Christian Theobalt, Gernot Ziegler, Marcus Magnor and Hans-Peter Seidel
MPI Informatik, Saarbrücken, Germany
{theobalt,gziegler,magnor,hpseidel}@mpi-sb.mpg.de

Abstract— **In recent years, the convergence of computer vision and computer graphics has put forth free-viewpoint video as a new field of research. The goal is to advance traditional 2D video into an immersive medium that enables the viewer to interactively choose an arbitrary viewpoint in 3D space onto the a scene while it plays back. In this paper we give an overview of a system for reconstructing, rendering and encoding free-viewpoint videos of human actors. It employs a hardware-accelerated marker-free optical motion capture algorithm from multi-view video streams and an a-priori body model to reconstruct shape and motion of a moving actor. Real-time high-quality rendering of the moving person from arbitrary perspectives is achieved by applying a multi-view texturing approach from the video frames. We also present a predictive encoding as well as a 4D-SPIHT wavelet compression mechanism that both exploit the 3D scene geometry for efficient encoding of the multi-view texture images.**

I. INTRODUCTION

We currently witness the convergence of the fields of computer graphics and computer vision. This development has been motivated by the idea of creating photo-realistic visualizations of real-world scenes in a computer not by designing models of shape and appearance, but by reconstructing these models from photographic or video data of the real world. One novel field of research that has been spawned by this convergence is the field of free-viewpoint video. In traditional video, the viewpoint onto a scene has been determined by the director and cannot be changed by the viewer. The goal of free-viewpoint video is to enable the viewer to interactively choose an arbitrary viewpoint onto the playback of a dynamic scene that was previously recorded in the real world. The algorithmic challenges to make such a technology reality are manifold and involve the acquisition of multi-view video (henceforth MVV) streams, the reconstruction of the recorded dynamic scenes and the efficient encoding of the 3D video footage. In this paper we describe our algorithmic solutions to these problems that lead to the development of a model-based free-viewpoint video system of human actors [1]. Our approach applies a model-based marker-less human motion capture algorithm to reconstruct free-viewpoint videos from MVV footage. The reconstructed 3D video can then be played back in real-time from any arbitrary viewpoint. Photo-realistic surface appearance of the actor is attained by creating dynamic MVV textures from the

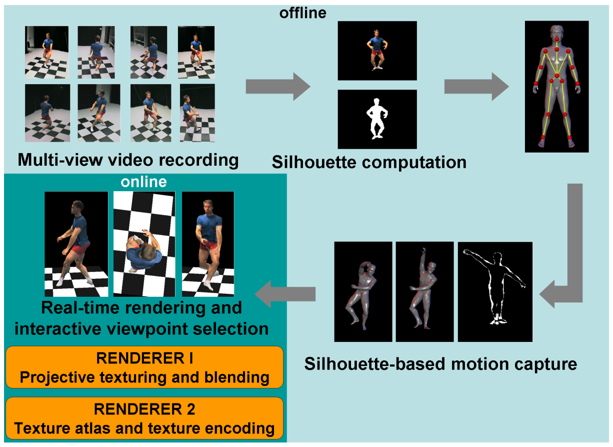
offline
Multi-view video recording
Silhouette computation
online
Real-time rendering and interactive viewpoint selection
RENDERER I
Projective texturing and blending
RENDERER 2
Texture atlas and texture encoding
Silhouette-based motion capture

Fig. 1. *Overview of the free-viewpoint video acquisition and rendering system.*

input video footage. We also developed algorithmic solutions for efficient compression of the surface textures.

II. RELATED WORK

A variety of different approaches have been proposed in the literature that aim at transforming 2D video and television into an immersive 3D medium. Free-viewpoint video is one category of 3D video in which the viewer shall be given the flexibility to interactively position himself at an arbitrary virtual location in a 3D. But the term 3D video also comprises other techniques, such as depth-image-based [2] or panoramic video [3].

The trail for 3D video applications was paved by algorithms from image-based rendering that aim at reconstructing novel renderings of a scene from input images [4]. These techniques have motivated novel research that draws from experience in computer vision and computer graphics to explicitly create 3D video systems. In depth-image-based approaches, novel viewpoints of a scene are reconstructed from color video and depth maps [2]. In [5] and [6] dynamic 3D scene geometry is reconstructed via stereo algorithms from multiple video cameras, and during playback the viewer can attain novel viewpoints in between the recording imaging sensors. In [7] a shape-from silhouette method is applied to reconstruct dynamic scenes from multiple video streams. Applying light-field based methods for free-viewpoint video has also been considered [8].

Whereas 3D video provides interactivity only on the viewer's side, in 3D TV the full pipeline from acquisition

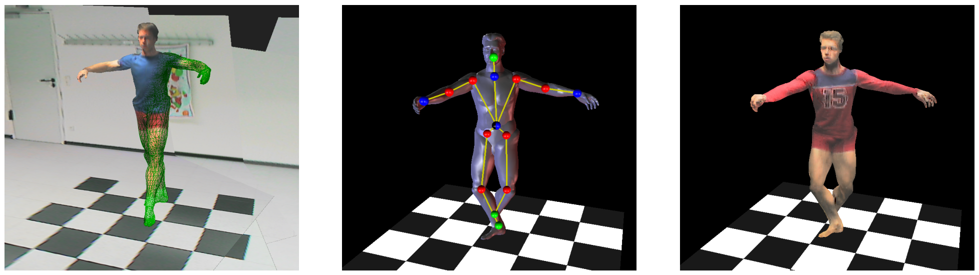

Fig. 2. *Blend between rendered body model and underlying triangle mesh (l); body model with kinematic structure (m); rendered free-viewpoint video (r).*

to display needs to run in real-time. A 3D TV system for a restricted set of novel viewpoints based on multiple video cameras for recording and multiple projectors for display has been presented in [9].

We propose a model-based system for free-viewpoint videos of human actors that employs a marker-less motion capture approach to estimate motion parameters. A comprehensive review of computer vision based motion capture algorithms can be found in the survey paper [10].

Whereas acquisition and rendering of free-viewpoint videos is one aspect, encoding of the 3D videos is essential for efficient transmission over broadcasting channels [11].

III. THE BIG PICTURE

Our free-viewpoint video acquisition and rendering system consists of: a multi-camera system for recording, a model-based marker-free optical motion capture algorithm for reconstruction, and a real-time renderer for display of the free-viewpoint videos [1]. The system consists of an off-line and an online component (Fig. 1). The off-line part involves the recording of the MVV footage of the actor with a set of synchronized video cameras as well as the estimation of the person's motion parameters. The (optional) texture compression is also performed off-line. Both, for estimating motion parameters, as well as for rendering the 3D video from a novel viewpoint, we employ an a-priori human body model (Fig. 2). The optical motion capture algorithm uses silhouettes of the moving person and an error measure computed in graphics hardware to robustly estimate motion parameters. In an initialization step, the shape and proportions of our a-priori model are adapted to the appearance of the recorded person.

The online component of the system provides a real-time renderer with an interaction interface for the user. Two different rendering systems are provided. Both renderers display the body model in the sequence of captured poses. Renderer I textures the model with the back-projected and rendered segmented video frames. Renderer II, on the other hand, applies time-varying texture maps on the human body model that have been created in a pre-processing step. The *texture atlas*, a parameterization function, allows our surface texture compression schemes to efficiently exploit scene geometry and temporal coherence [12].

IV. MULTI-VIEW VIDEO RECORDING

The video sequences used as inputs to our system are recorded in our multi-view video studio [13]. IEEE1394 cameras are placed in a convergent setup around the center of the scene. The video sequences used in our experiments are recorded from 8 static viewing positions arranged at approximately equal angles and distances around the center of the room. All cameras are synchronized and record at a resolution of 320x240 pixels and a frame rate of 15 fps (maximum frame rate with external trigger). The cameras are calibrated into a global coordinate system. In each video frame, the silhouette of the person in the foreground is computed via background subtraction.

V. THE MODEL

In our system we apply a generic human body model consisting of 16 individual body segments. Each segment's surface is represented via a closed triangle mesh. The model's kinematics are defined via 17 joints that connect the body segments and form a hierarchical skeleton structure. 35 pose parameters are needed to completely define the pose of the body. In total, more than 21000 triangles make up the human body model (Fig. 2).

The generic model does not, in general, have the same proportions as its human counterpart. To be able to adapt model size and proportions to the recorded person, each segment can be individually scaled, and its surface deformed. The parameters controlling model stature and build are derived from the model overlap with silhouette images of an initialization pose. They are kept fixed during actual motion capture.

VI. MOTION CAPTURE

Since any form of visual markers in the scene would necessarily change its natural appearance, we developed a marker-less human motion capture method to acquire free-viewpoint videos based on our a-priori model. In our method, the individualized geometry model automatically tracks the motion of a person by optimizing the 35 joint parameters for each time step. This is achieved by matching the projected body model to the segmented silhouette images of the person in each of the input camera views so that the model performs the same movements as the human in front of the cameras.

The pose parameters are determined by means of a hierarchical non-linear optimization procedure. The number of pixels of the projected model silhouettes and the input silhouettes that do not overlap is our comparison measure. Conveniently, the exclusive-or (XOR) operation between the rendered model silhouettes and the segmented video-image silhouettes yields exactly those pixels. The sum of these XOR pixels over all camera perspectives forms our error measure that can be efficiently evaluated in graphics hardware.

For numerical optimization of the pose parameters we employ a standard non-linear optimization method, such as Powell's method. To efficiently avoid local minima and to obtain reliable model pose parameter values, the parameters are not all optimized simultaneously. Instead, the model's hierarchical structure is exploited. Model parameter estimation is performed in descending order with respect to the individual segments' impact on silhouette appearance and their position along the model's kinematic chain. First, the position and orientation of the torso are varied to find its 3D location. Next the arms and legs are fitted using a joint parameterization for their lower and upper parts. Finally, the hands and the feet are regarded.

To avoid local, sub-optimal error minima for the arms and legs a limited regular grid search precedes the optimization search. This procedure accelerates convergence and effectively avoids local minima. Inter-penetrations between limbs are prevented by incorporating a collision check based on bounding boxes into the parameter estimation.

The motion parameters as well as the body deformation parameters are saved in our proprietary free-viewpoint video file format that serves as input for the real-time renderer.

Recently, we have enhanced the basic silhouette-based tracking system by implementing it as a client-server application using 5 CPUs and GPUs [14]. We also included texture information into the tracking process by deriving pose corrections from 3D corrective flow fields reconstructed from optical flow [15].

VII. RENDERING WITH PROJECTIVE TEXTURES

A high-quality 3D geometry model is now available that closely matches the dynamic object in the scene over the entire length of the sequence. Renderer I displays the free-viewpoint video photo-realistically by rendering the model in the sequence of captured body poses and by projectively texturing the model with the segmented video frames. Time-varying cloth folds and creases, shadows and facial expressions are faithfully reproduced, lending a very natural, dynamic appearance to the rendered object (Fig. 2). To attain optimal rendering quality, the video textures need to be processed off-line prior to rendering: Since the final surface texture at each time step consists of multiple images taken from different viewpoints, the images need to be appropriately blended in order to appear as one consistent object surface texture. Also, local visibility must be taken into account, and any adverse effects due to inevitable small differences between model geometry and the true 3D object surface must be countered efficiently. For appropriate blending of the input camera views, per-vertex blending weights need to be computed and the visibility of each vertex in every input camera view needs to be determined. If surface reflectance can be assumed to be approximately Lambertian, view-dependent reflection effects play no significant role. Thus, the weights are computed independent of the output view in such a way that the camera seeing a vertex best gets the highest blending weight. This is achieved by assigning the reciprocal of the angle between the vertex normal and a camera's viewing direction as blending weight to each camera's texture fragment. An additional rescaling function is applied to these weights that allows for the flexible adjustment of the influence of the best camera on the final texture.

The 0/1-visibility of each vertex in each input camera view is precomputed and saved as part of the free-viewpoint video file. Since the silhouette outlines do not always exactly correspond to the projected model outlines in each camera view, we apply an extended visibility computation from a set of displaced camera views to avoid projection artifacts.

Finally, while too generously segmented video frames do not affect rendering quality, too small outlines can cause annoying untextured regions. To counter such artifacts, all image silhouettes are expanded by a couple of pixels prior to rendering.

During rendering, the color from each texture image is multiplied by its vertex-associated normalized blending weight and its 0/1-visibility in the programmable fragment stage of the graphics board. The final pixel color is the sum of the scaled texture colors.

Optionally, Renderer I can also reproduce view-dependent lighting effects by means of view-dependent rescaling of the view-independent blending weights.

VIII. RENDERING AND ENCODING USING VIDEO TEXTURES

For encoding purposes we resort to a different texture representation and rendering algorithm. Renderer II represents surface textures as an angle-dependent video texture, or multi-view video texture, that has been created beforehand using a parameterization of the body model. During display Renderer II unpacks the MVV textures from a compressed code-stream.

Since the texture changes only slightly with the viewing angle and over time, MVV textures actually comprise correlated 4D data volumes. In the following we detail how the MVV textures are constructed and describe two methods for their efficient encoding. No new compression method has to be provided for the 3D model and motion capturing data, as compressed bit-streams already are specified as sub-standards of the MPEG4 AFX specification [16].

IX. TEXTURE ENCODING

A. Texture Atlas

The mapping from the input streams into texture space is done by defining a texture atlas which maps the surface of the 3D model into the 2D domain. The problem of creating a texture atlas is closely related to the problem of surface parameterization. For general meshes it is necessary to introduce cuts on the surface in order to bound the distortion. Those cuts may partition the surface into distinct patches.

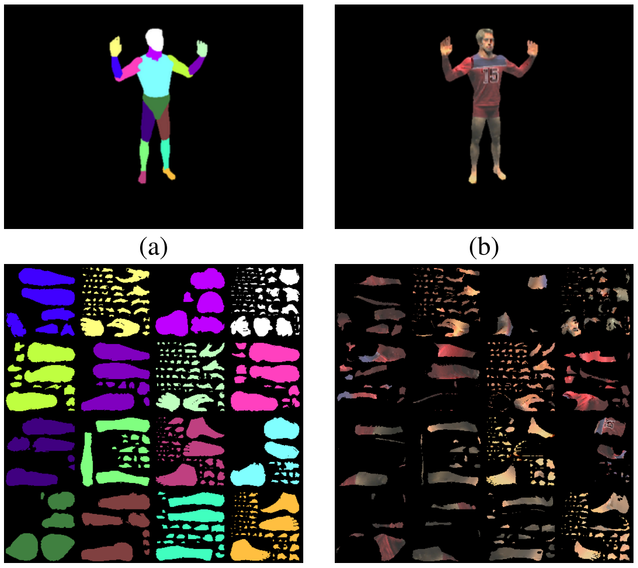

Fig. 3. *Resampling into the texture atlas: (a) Color-coded body parts. (c) Corresponding regions in texture space. (b) Original frame. (d) Resampled frame considering visibility (512x512).*

Our method resembles the approach presented in [17], in the way that our texture atlas is constructed by projecting the triangles of one patch orthogonally onto a plane defined by the average surface normal. Starting with one arbitrary seed triangle, neighboring triangles are added to the patch until the triangle normal deviates too much from the average normal. If one patch cannot grow any further another seed triangle starts a new patch. A separate texture atlas is constructed for each body part which are then joined into a single atlas as demonstrated in Figure 3a) and c).

Vertex programs on graphics hardware assist in the texture resampling process. In order to compute visibility, we perform a traditional shadow mapping approach with the camera position used as the light source. All non-visible texels will be rendered black as can be seen in Figure 3d).

B. Compression Overview

Two compression approaches can be used:

- Predictive Texture Encoding [12]: A two step average (first over all camera angles, then over all time steps) is generated, then differential textures are computed with respect to the average textures. All resulting textures are then compressed using a shape-adaptive wavelet algorithm. The format grants random access over 2 differential image additions. .
- 4D-SPIHT coding [18]: All camera-specific textures comprise one large 4D array of texel data. For random access and acceptable decoding time they are chopped into blocks with an equal number of camera angles and time duration. Unused texels are exploited to reduce entropy. Afterwards, a modified SPIHT algorithm in four dimensions provides the necessary, lossy data compression. The SPIHT-specific data structure allows early termination of decoding to fulfill time constraints in real-time playback.

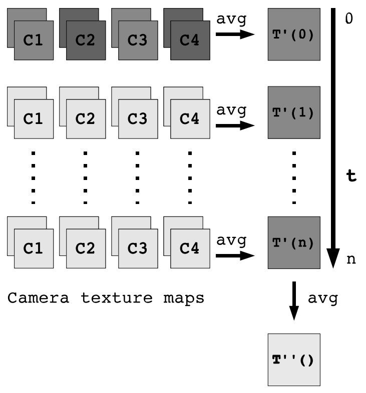

Fig. 4. *The averaging scheme for texture encoding.*

C. Predictive Encoding Scheme

Encoding the video streams now means encoding the resulting texture maps of each time step t and camera c. We experienced a strong coherence inbetween the camera-specific textures of one timestep as well as over time. The proposed encoding reflects this coherence in a two level encoding.

The encoding is shape-adaptive, i.e., all black texels will be encoded by a shape mask, and only the visible texels $p(x,y,t,c)$ for which $vis(x,y,t,c) = 1$ need to be considered for each frame.

At first, the average $T'(x,y,t)$ of all camera textures at one timestep is computed, and from this the average over all frames $T''(x,y)$ (the parameters x and y are left out in the remainder of this paper). See Fig. 4 for a graphical sketch of the process.

After computing the average textures T' and T'' all textures (also the input textures) are converted from RGB to YUV color space in which all further processing is performed. Then, only the T'' texture is encoded using a shape-adaptive wavelet encoder (Binary Set Splitting with k-d Trees) [19].

Afterwards differential textures $R(t,c)$ to the original texture maps are extracted for each frame and camera. Weighting the input in Equations 1-3 with the visibility vis when computing the averages avoids affecting the distance from the valid pixel values to the computed average by non-visible pixels. This would otherwise increase the entropy of the differential textures and thus degrade encoding quality.

More specific, we first compute for each timestep the difference textures $R'(t)$ between the overall T'' and the timestep average $T'(t)$. In order to get back to the original textures we further compute for each camera and timestep $R(t,c) = p(t,c) - (T'' + R'(t))$. Again all $R'(t)$ and $R(t,c)$ are encoded using shape-adaptive wavelet encoding.

Since the applied wavelet encoding is lossy the decoded result would be strongly influenced by the decoding error in T'' and $R'(t)$. We avoid this influence by computing all differences with respect to the once encoded and decoded textures $\hat{T}''$ and $\hat{R}'(t)$ respectively. Instead of averaging the entire stream by a single T'' one may also define shorter time spans and compute several T''.

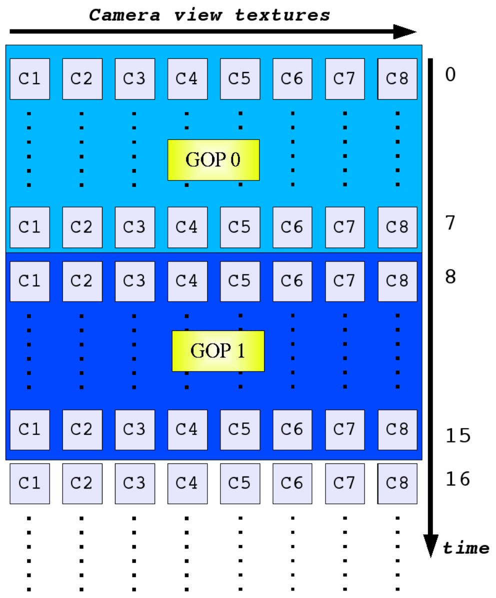

Fig. 5. *Group of Pictures arranged from MVV textures.*

As the final result, the encoded data stream contains BISK-encoded files for the YUV-planes and bitmasks of:

- $\hat{T}''$: the average of all timestep averages.
- $\hat{R}'(t)$: the difference textures to the timestep average.
- $\hat{R}(t,c)$: the difference textures of each camera texture.

D. 4D-SPIHT Encoding

1) Data grouping and Filling: Compression of a data block commences when all necessary camera images are available as textures.

After resampling, we group the texture maps into blocks of spatial and temporal coherency, yielding four-dimensional data blocks of YUV samples. The block division corresponds to the GOP (group of picture) block structure commonly used in MPEG video formats, and allows for limited random access as long as the whole 4D block containing a certain texture is decoded, see Figure 5. U and V values can optionally be subsampled, but we currently work with reduced bitrates for these color components, see the SPIHT encoder below.

Unused texels (currently: black pixels) in these 4D blocks are now filled with averages of the surrounding valid texels, see Fig. 6 for an example. This ensures best possible data compression under the subsequently applied algorithm, as described in [20].

To serve this purpose, the whole 4D data block is first downsampled in a Laplacian 4D pyramid, all the way to the lowest resolution of 1x1, taking the different dimension extents into consideration (a division by two remains one if the result would be smaller than one). Afterwards, the pyramid is traversed backwards from the lowest to the highest resolution, and each unused (black) texel receives the color of its associated, average parent in the previous level. This way, it is ensured that all unused texels are filled with a color value that corresponds to the average of all valid texels in its support region.

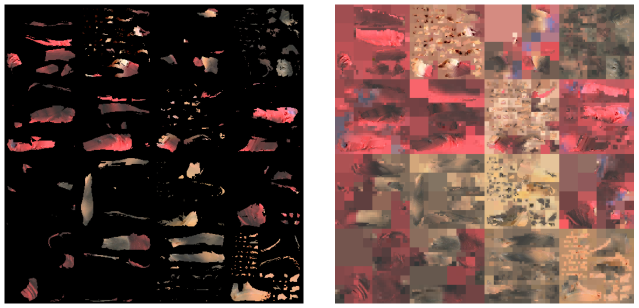
Fig. 6. *Left: input texture map. Right: Same map after filling operation.*

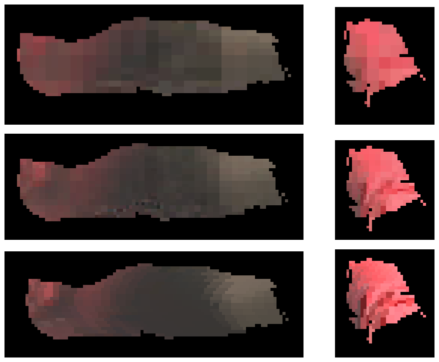
Fig. 7. *Two texture map patches. Decoding was performed with equal Y, U, V datarate, and using the fill feature. Top: 0.05 bpp; Middle: 0.25 bpp; Bottom: Encoder input.*

2) Wavelet Encoding: The following 4D wavelet transformation uses Haar wavelets. We take the 4D data block that was filled in the previous step, and sequentially apply a 1D Haar wavelet tranform in all four dimensions until even the texture dimension sizes have been reduced to 2. Finally, compression commences. The compression algorithm is based on the widely-used SPIHT algorithm, although in a new adaptation, making it suitable for 4D data. It is based on work done in [20]. The encoder is currently able to handle a 4D data block with pairs of equal dimensions (e.g. max(s,t,u,v) = {8,8,512,512}, that is, 8 timesteps of 8 cameras at 512 x 512 resolution). The SPIHT encoder very much resembles its classic 2D counterpart [21], with one exception: Since the 4D dimensions are of different size, the codec has to be able to detect boundary conditions that generate a different number of descendants in the wavelet data traversal.

3) Decoding: Most decoders will probably extract several time steps at once, since time steps are usually read sequentially. The bit mask can only be applied if it has been transmitted to the decoder. If this is not the case, shadow casting must be applied for masking if multi-view interpolation is intended (as noted in the next section).

Figure 7 shows example output from the reference decoder. Notice the typical wash-out effect of wavelet compression. The outer contours were transmitted in an additional shape-mask.

E. The Shape Mask

Texture shape masks are not needed if the original 3D model is available; only visible texels will be accessed in the rendering phase, since only those were exposed to the camera in the given view. Therefore it suffices to provide the renderer with the unmasked texture maps. If, on the other side, *intermediate views* shall be rendered, then the texel visibility mask needs to be applied to the texture maps before rendering One way is to store it in the bitstream, which requires some extra storage space. The other approach is to reconstruct the relevant masks from the 3D model in the decoder, using the visibility algorithm that was originally used by the encoder. This improves compression, but requires a minimum of two extra rendering passes.

X. RESULTS AND CONCLUSION

Our free-viewpoint video system can robustly reconstruct such complex motion as that of expressive jazz dance(Fig. 2). Fitting times of 1 s per frame are achievable (for details see [1], [14]). With both rendering algorithms highly realistic renditions of the scene from novel viewpoints are obtained, preserving subtle details in surface appearance.

A novel rendering method enables photo-realistic display of free-viewpoint videos using texture streams that have been compressed in texture space. A straightforward way of texture compression would be to encode the input video frames using MPEG2. Applying our predictive encoder as well as our 4D-SPIHT encoder in the texture domain has the advantage that the codecs can benefit from structural scene information during compression. With the predictive scheme, we obtain compression rates in the range of 8:1 to 50:1. The 4D-SPIHT encoder enables compression rates between 20:1 and 280:1, it also allows for random access into the bitstream. Which codec and which compression rates are suitable depend on the target application and the tolerable PSNR value (see [12], [18] for PSNR ranges).

The system produces convincing results on complex test scenes both in terms of rendering and reconstruction. In future, we plan to extend the system to automatically derive more advanced surface appearance models.

REFERENCES

[1] J. Carranza, C. Theobalt, M. Magnor, and H.-P. Seidel, “Free-viewpoint video of human actors,” in *Proc. of ACM SIGGRAPH*, 2003, pp. 569–577.

[2] C. Fehn, “Depth-image-based rendering (dibr), compression and transmission for a new approach on 3d-tv,” in *Proc. Stereoscopic Displays and Applications*. to appear, 2004, p. nn.

[3] D. Kimber, J. Foote, and S. Lertsithichai, “Flyabout: spatially indexed panoramic video,” in *ACM Multimedia*, 2001, pp. 339–347.

[4] H.-Y. Shum and S. Kang, “A review of image-based rendering techniques,” in *Proc. of IEEE/SPIE VCIP*, 2000, pp. 2–13.

[5] K. Cheung, T. Kanade, J.-Y. Bouguet, and M. Holler, “A real time system for robust 3D voxel reconstruction of human motions,” in *Proc. of CVPR*, vol. 2, 2000, pp. 714–720.

[6] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski, “High quality video view interpolation using a layered representation,” in *Proc. of ACM SIGGRAPH*, 2004, pp. 600–608.

[7] T. Matsuyama and T. Takai, “Generation, visualization, and editing of 3D video,” in *Proc. of 3DPVT’02*, 2002, p. 234ff.

[8] M. Droese, T. Fujii, and M. Tanimoto, “Ray-space interpolation based on filtering in disparity domain,” in *Proc. 3D Image Conference*, 2004.

[9] W. Matusik and H.-P. Pfister, “3d tv: A scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes.” in *Proc. of ACM SIGGRAPH 2004*, 2004, pp. 814–824.

[10] T. B. Moeslund and E. Granum, “A survey of computer vision-based human motion capture,” *CVIU*, vol. 81, no. 3, pp. 231–268, 2001.

[11] S. Wuermlin, E. Lamboray, O. Staadt, and M. Gross, “3D video recorder,” in *Proc. of Pacific Graphics*. IEEE, 2002, pp. 325–334.

[12] G. Ziegler, H. Lensch, N. Ahmed, M. Magnor, and H.-P. Seidel, “Multi-video compression in texture space,” in *Proc. of ICIP’04*, 2004, to appear.

[13] C. Theobalt, M. Li, M. Magnor, and S. H.-P., “A flexible and versatile studio for synchronized multi-view video recording,” in *Proc. of Vision, Video and Graphics*, 2003, pp. 9–16.

[14] C. Theobalt, J. Carranza, M. A. Magnor, and H.-P. Seidel, “A parallel framework for silhouette-based human motion capture,” in *Proc. of VMV*, 2003, pp. 207–214.

[15] C. Theobalt, J. Carranza, M. Magnor, and H.-P. Seidel, “Enhancing silhouette-based human motion capture with 3d motion fields,” in *Proc. Pacific Graphics*. IEEE, 2003, pp. 185–193.

[16] http://www.chiariglione.org/mpeg/.

[17] O. Sorkine, D. Cohen-Or, R. Goldenthal, and D. Lischinski, “Bounded-distortion Piecewise Mesh Parameterization,” in *IEEE Visualization*, 2002.

[18] G. Ziegler, H. P. A. Lensch, M. Magnor, and H. P. Seidel, “Multi-video compression in texture space using 4D SPIHT,” in *Proc. of MMSP*, 2004, to appear.

[19] J. E. Fowler, “Shape-adaptive coding using binary set splitting with *k*-d trees,” in *IEEE ICIP*, 2004, to appear.

[20] M. Magnor, P. Ramanathan, and B. Girod, “Multi-view coding for image-based rendering using 3-D scene geometry,” *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 11, pp. 1092–1106, Nov. 2003.

[21] A. Said and W. A. Pearlman, “A new fast and efficient image codec based on set partitioning in hierarchical trees,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 243–250, 1996.