MULTI-VIDEO COMPRESSION IN TEXTURE SPACE

Gernot Ziegler, Hendrik P. A. Lensch, Naveed Ahmed, Marcus Magnor, Hans-Peter Seidel

MPI Informatik, Saarbrücken, Germany

ABSTRACT

We present a model-based approach to encode multiple synchronized video streams depicting a dynamic scene from different viewpoints. With approximate 3D scene geometry available, we compensate for motion as well as disparity by transforming all video images to object textures prior to compression. A two-level hierarchical coding strategy is employed to efficiently exploit inter-texture coherence as well as to ensure quick random access during decoding. Experimental validation shows that attainable compression ratios range up to 50:1 without subsampling. The proposed coding scheme is intended for use in conjunction with Free-Viewpoint Video and 3D-TV applications.

1. INTRODUCTION

Recent advances in imaging technology have made the acquisition of multiple synchronized videos of real-world dynamic scenes economically feasible. This new media modality is useful, e.g., to re-display the recorded action from an arbitrary perspective (3D-TV, Free-Viewpoint Video) [1, 2]. Although only a sparse set of cameras might be used, the resulting data amounts are still tremendous. In uncompressed state, the data expenditure equals to one video stream per recording camera. Available bandwidth as well as DVD storage capacity currently make distribution of such uncompressed media impossible. While standardized video compression techniques could be applied, these cannot exploit the high degree of redundancy inherent in multiple video recordings of the same object. Especially, no inter-stream coherence would be exploited. In addition, synchronized, random access of video frames is not easily possible.

Using multiple images as texture is a relatively new research area. The benefits of having multiple images available to render view-dependent effects have first been shown by Debevec et al. [3]. Compression of multi-view image data in the texture domain has been explored by a few researchers. Nishino et al. apply eigenvector decomposition to a number of textures created from images and a 3D model of the object [4]. Wood et al. investigate vector quantization and principal function analysis to compress local reflection characteristics [5]. Magnor et al. make use of a 4D wavelet decomposition to exploit textural coherence also between textures [6]. All previous work was concerned with encoding multiple textures of a static object. This paper, in contrast, presents an efficient approach to compress multi-video image data of a dynamic scene in the texture domain.

Since the differences in the recorded frames are mainly due to disparity as well as object motion, we separate the recorded information into the object's shape and its surface texture. Object movement and disparity are compensated in texture space. Differences between texture maps are much smaller than between the original images. As a result much higher compression rates can be achieved. To efficiently exploit redundancy, we propose a two-level hierarchical coding approach. First, we construct one average texture map for each time step from all camera images. All average textures are then averaged again to yield one base texture. We encode the residual texture differences between the base texture and each time-step average texture as well as the difference between average textures and each corresponding camera view texture using a shape-adaptive wavelet coder. Our experimental results with 8 synchronous video streams show that usable compression ratio ranges up to 50:1 without YUV subsampling, which includes the possibility to render arbitrary viewpoints.

2. OVERVIEW

Several computing steps are necessary to transform the raw multi-view camera recordings into a surface mesh with motion parameters and corresponding texture maps. We demonstrate the principles on the basis of capturing human actors.

The motion parameters are obtained by the free-viewpoint video approach by Caranza et al. [2]. Eight synchronized videostreams of a human actor are captured by calibrated cameras and seperated into fore- and background pixels. Motion parameters are estimated for each frame in each stream by silhouette matching.

Using the camera and motion parameters the input frames are resampled into a texture atlas, i.e., a different texture map is created for each time step and camera while the texture atlas parameterization itself is constant. The resampling is carried out using graphics hardware and is ex-

plained in more detail in Section 3.

The resulting texture maps are then compressed. After generating a two step average (first over all camera views in one time step, then over all time steps within the given time range), differential images are computed with respect to the average images. Since this is done at both levels, the redundancy between the multiple views of one timestep is removed before the temporal coherence is considered.

All resulting images are then compressed using a shape-adaptive wavelet algorithm which exploits the fact that not all areas in the texture maps are actually used. The wavelet encoder utilizes decoding prediction to avoid propagating errors in the dependent two step pixel decoding.

3. 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. A projection of the mesh into the 2D plane can be performed by minimizing texture stretch and distortion on the surface, as for example proposed in [7, 8, 9, 10, 11]. For general meshes it is necessary to introduce cuts on the surface in order to bound the distorsion. Those cuts may partition the surface into distinct patches.

Our texture atlas is constructed by projecting the triangles of one patch orthogonally onto a plane defined by the average surface normal. The selection of patches ensures a minimum sampling density of the surface: 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 texture as demonstrated in Figure 1a) and c).

3.1. Hardware-Accelerated Resampling

Given the animation parameters, i.e., the tranformation matrices for each body part, each original vertex is moved to its transformed 3D position. Exploiting graphics hardware we can now easily map each frame of the input streams into the texture domain using a vertex program: the final vertex position is set to the 2D texture atlas coordinates of the vertex while the transformed 3D vertex position is used to compute the image position in the current frame which defines how to texture the currently rendered triangle. 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 1d).

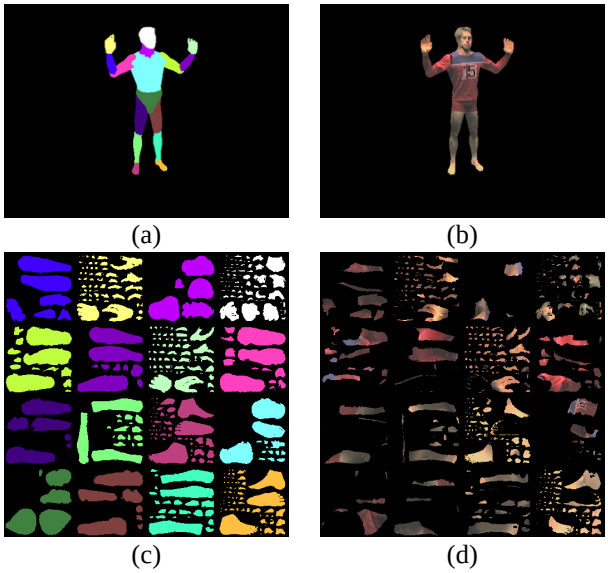

Fig. 1. *Resampling into the texture atlas: (a) Color coded body parts. (c) Corresponding regions in texture space. (b) Original frame (320x240). (d) Resampled frame considering visibility (512x512).*

4. SHAPE-ADAPTIVE WAVELET ENCODING

Encoding the video streams now means encoding the resulting texture maps of each time step t and camera c. We experienced a strong coherence of the texture maps of the different camera images 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 bit 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 images 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):

$$T'(x,y,t) = \frac{\sum_c p(x,y,t,c) \cdot vis(x,y,t,c)}{\sum_c vis(x,y,t,c)} \quad (1)$$

$$vis'(x,y,t) = \bigvee_c vis(x,y,t,c) \quad (2)$$

$$T''(x,y) = \frac{\sum_t T'(x,y,t) * vis'(x,y,t)}{\sum_t vis'(x,y,t)} \quad (3)$$

See Figure 2 for a graphical sketch of the process.

After computing the average images T' and T'' all images (also the input textures) are converted from RGB to YUV color space in which all further processing is performed. Then, only the T'' image is encoded using a shape-adaptive wavelet encoding (BISK) [12].

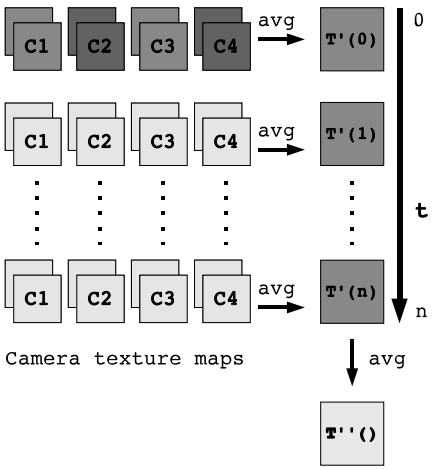

Fig. 2. *The averaging scheme.*

Afterwards differential images $R(t,c)$ 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 images and thus degrade encoding quality.

More specific, we first compute for each timestep the difference images $R'(t)$ 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.

```
for all timesteps t:
  average all camera images c → T'(t)
average over all timesteps → T''
encode, decode T'' → T̂''
for all timesteps t:
  T̂'' − T'(t) → R'(t)
  encode, decode R'(t) → R̂'(t)
  reconstruct T̂'' + R̂'(t) → T̂'(t)
  for all camera views c:
    T(t,c) − T̂'(t) → R(t,c)
    encode R(t,c)
```

Fig. 3. *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 images $\hat{T}''$ and $\hat{R}'(t)$ respectively. The overall encoding pipeline is listed in Figure 3. Instead of averaging the entire stream by a single T'' one may also define shorter time spans and compute several T''.

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

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

Furthermore, the object shape and the motion parametes are included. The data stream is currently comprised of invidual files for each image component. No container format or supplemental information is currently included.

5. DECODING

Given the aforementioned encoding pipeline, decoding the streams is straight forward and explained in Figure 4.

```
decode T̂''()
for all timesteps t:
  decode R̂(t)
  reconstruct:  T̂''() + R̂(t) → T̂(t)
  for all camera views c:
    decode R̂(t,c)
    reconstruct:  T̂'(t) + R̂(t,c) → T̂(t,c)
    write/apply T̂(t,c)
```

Fig. 4. *Decoding.*

6. THE SHAPE MASK

As mentioned earlier, visibility information is represented by a bitmask such that only the visible texels are considered during encoding and decoding. The shape mask is however implicitly defined by the object's mesh and the motion parameters. Thus, the shape mask does not need to be represented explicitly and can be omitted during transmission if the object shape is given.

7. RESULTS

The verification of the encoder results was achieved by comparing a 350 frames long rendering of the model animation (the dancer scene used in [2]) with the original, recorded camera images. Only the Y-values of the rendering were considered, but we expect similar results for the U and V values.

To acquire an indication of the maximum quality achievable with the rendered result, we mapped the texture maps back to the 3D model without using any compression pipeline inbetween. This resulted in a maximum limit for the achievable image quality, called *ideal reconstruction*.

We encoded then the texture maps at various bitrates, and mapped the decoded results back to the 3D model to gain an impression on how much the renderings were affected. The results are shown in in Figure 5.

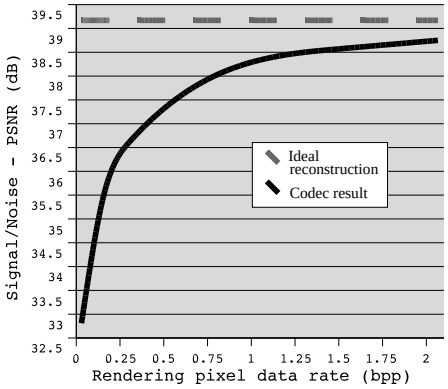

Fig. 5. *Decoding results for the dancer scene.*

As expected, the quality rating (measured in the PSNR of the two renderings' Y-values) improved with increasing bitrate. It showed for our test case that a bitrate increase from 1.5 to 2.1 bpp did not provide any substantial gain in image quality; it levelled out at approximately 1 dB below the ideal reconstruction quality.

A compression rate of 1 bpp can safely be used without sacrifices in reproduction quality. The compression ratios of 50:1 to 8:1 compare well to MPEG standard compression, with the strong advantage that arbitrary views can be rendered from the input data. The areas used in the texture maps and the rendered images compare at a ratio of 3.1:1.

8. CONCLUSIONS

We have presented a coding approach for a new data modality, synchronous multi-video image data, needed for Free-Viewpoint Video as well as 3D-TV applications. Our method relies on having an (approximate) 3D geometry model of the dynamic scene available that can be used to map the video images to textures. By working in texture space, we compensate for disparity between camera views as well as for object motion over time. By following a two-level hierarchical coding approach, we enable random access to any view by decoding just two texture difference maps.

Compression ratios up to 250:1 have been verified experimentally, with the usable range below 50:1, all without YUV subsampling. This compares well to MPEG standard compression, with the additional advantage of reproducing arbitrary viewpoints. The next steps will include accelerating decoding to allow for real-time rendering of the scene from the compressed bit stream. To further improve coding efficiency, we are going to investigate how to retrieve surface reflectance characteristics in addition to the diffuse reflection component currently exploited, and find estimates on the optimal area ratio between texture map and rendered image.

We will also investigate if U and V values should be subsampled prior to wavelet encoding, or if a reduced bitrate suffices.

9. REFERENCES

[1] C. Fehn, P. Kauff, M. Op de Beeck, F. Ernst, W. Ijsselsteijn, M. Pollefeys, E. Ofek L. Van Gool, and Sexton I., "An evolutionary and optimised approach on 3D-TV," *Proc. International Broadcast Conference (IBC'02)*, pp. 357–365, Sept. 2002.

[2] J. Carranza, C. Theobalt, M Magnor, and H.-P. Seidel, "Free-viewpoint video of human actors," *ACM Trans. on Computer Graphics*, vol. 22, no. 3, July 2003.

[3] P. Debevec, C. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach," *Computer Graphics (SIGGRAPH 96 Proceedings)*, pp. 11–20, Aug. 1996.

[4] K. Nishino, Y. Sato, and K. Ikeuchi, "Eigen-texture method: appearance compression based on 3d model.," *Proceedings of IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 618–624, June 1999.

[5] D. Wood, D. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. Salesin, and W. Stuetzle, "Surface light fields for 3D photography," *Proc. ACM Conference on Computer Graphics (SIGGRAPH-2000)*, New Orleans, USA, pp. 287–296, July 2000.

[6] 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.

[7] S. Marschner, *Inverse rendering for computer graphics*, Ph.D. thesis, Cornell University, 1998.

[8] D. Piponi and G. D. Borshukov, "Seamless texture mapping of subdivision surfaces by model pelting and texture blending," in *Proceedings of ACM SIGGRAPH 2000*, July 2000, Computer Graphics Proceedings, Annual Conference Series, pp. 471–478.

[9] P. V. Sander, J. Snyder, S. J. Gortler, and H. Hoppe, "Texture mapping progressive meshes," in *Proceedings of ACM SIGGRAPH 2001*, Aug. 2001, Computer Graphics Proceedings, Annual Conference Series, pp. 409–416.

[10] Olga Sorkine, Daniel Cohen-Or, Rony Goldenthal, and Dani Lischinski, "Bounded-distortion Piecewise Mesh Parameterization," in *IEEE Visualization*, 2002.

[11] Xianfeng Gu, Steven J. Gortler, and Hughes Hoppe, "Geometry images," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 355–361, July 2002.

[12] J. E. Fowler, "Shape-adaptive coding using binary set splitting with *k*-d trees," in *IEEE International Conference on Image Processing*, 2004, submitted.

[13] H. Danyali and A. Mertins, "Fully scalable texture coding of arbitrarily shaped video objects," *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'03)*, pp. 393–396, Apr. 2003.