SPIE Conference Proceedings Visual Communications and Image Processing (VCIP2000)
Perth, Australia, pp. 14-22, June 2000

Model-based Coding of Multi-Viewpoint Imagery

Marcus Magnor[a] and Bernd Girod[b]

[a]Telecommunications Laboratory, University of Erlangen-Nuremberg
91058 Erlangen, Germany

[b]Informations Systems Laboratory, Stanford University
Stanford, CA 94305-9510

ABSTRACT

A compression scheme for calibrated images depicting a static scene from arbitrary viewpoints is presented. 3-D scene geometry is reconstructed, and view-dependent texture maps are generated from all images. Texture is wavelet-coded using the SPIHT coding scheme extended to 4-D, exploiting correlations within as well as between texture maps. During decoding, all texture maps are simultaneously and progressively reconstructed. The coder provides 3-D scene geometry and multiple texture maps, enabling the use of graphics hardware to accelerate the rendering process. Three image sets acquired from real-world objects are used to evaluate the model-based coding scheme. Coding efficiency is shown for geometry approximations of different accuracy.

Keywords: Image-based Rendering, Compression, Geometry, Texture, SPIHT

1. INTRODUCTION

Photorealistic, fast rendering constitutes a fundamental research objective of computer graphics. Conventional rendering systems rely on object geometry in conjunction with texture and illumination information to render scene appearance. Scene geometry is commonly modelled by polygon meshes approximating object surfaces. Steadily increasing demand for high-performance 3-D graphics has led to the development of specialized graphics hardware to accelerate polygon rendering. Today, *geometry-based rendering* can rely on very powerful hardware support: low-cost, consumer-market video game consoles are capable of rendering up to 3 million polygons per second.[1]

While graphics hardware can accelerate the rendering process, rendering quality strongly depends on scene characteristics. Polygon meshes approximate object surfaces as piecewise planar surface patches. For complex objects, large numbers of polygons are necessary to obtain sufficiently accurate geometry descriptions. Despite computationally expensive global illumination calculations, geometry-based rendering results of natural scenes often exhibit an artificial and sterile impression.

In recent years, a novel approach to rendering real-world scenes has attracted considerable attention. In *Image-based Rendering (IBR)*, information on scene appearance is directly exploited. Conventional images capturing natural scene appearance from multiple viewpoints serve as the basis of the rendering process. While IBR performance is largely independent of scene complexity, rendering quality depends on the number of views available. To attain photorealistic rendering results, many thousands of images must be acquired, stored and transmitted, calling for efficient compression.[2–4]

A number of different IBR techniques have been proposed to interpolate intermediate views from recorded image data, ranging from solely image-based *light field rendering*[5] over the *lumigraph*[6] using additional depth information to *view-dependent texture mapping (VDTM)*[7,8] which applies multiple textures to an approximate 3-D geometry model. As IBR-supporting hardware does not exist, light field as well as lumigraph rendering require high-end workstations to achieve interactive rendering rates and leave little computational resources for image data decoding. In *view-dependent texture mapping*, an approximate 3-D scene geometry model must be available. It is textured using multiple texture maps. Because polygons are textured, VDTM can exploit conventional rendering-accelerating graphics hardware, leaving time to decode texture information during rendering. With regard to compression,

Correspondence: magnor@LNT.de; www.LNT.de/~magnor

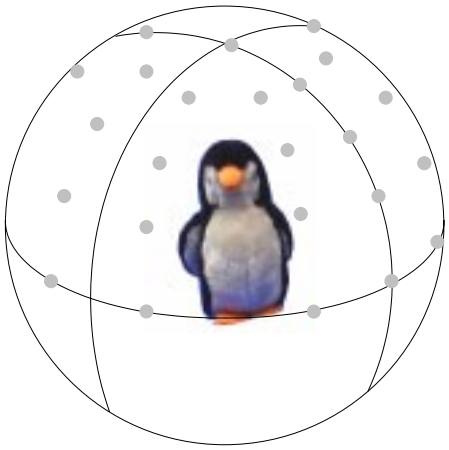

Figure 1. Image recording geometry: camera positions are arranged on a half-sphere around the object.

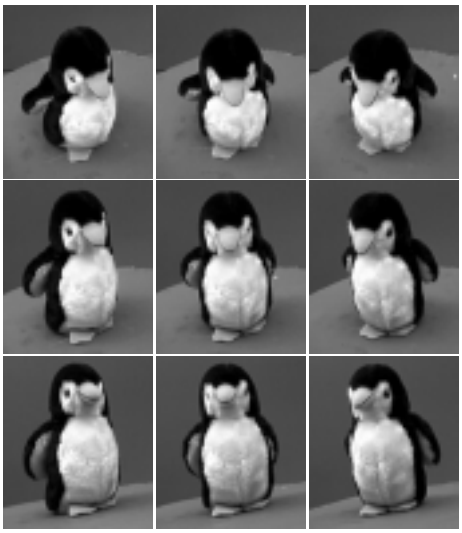

Figure 2. Image set: natural object appearance is captured from multiple viewpoints.

texture maps offer a distinct advantage over conventional images: as object surface points are mapped to fixed texture coordinates, texture-mapped images generated from exact scene geometry exhibit no disparity, and texture maps show higher inter-correlations than the original image recordings.

The model-based coder described in this paper extends the concept of *view-dependent texture mapping*. A high-resolution 3-D scene model is reconstructed from calibrated images. Scene geometry is approximated and compressed. An efficient mapping between geometry surface and texture plane is presented. After texture maps are generated from all scene images, texture is coded exploiting correlations within as well as between texture maps using subband coding in conjunction with a SPIHT coding scheme extended to 4 dimensions.[9,10] During decoding, all texture maps are progressively reconstructed. Model-based coding performance is evaluated with regard to model accuracy using three image sets acquired from real-world objects.

2. GEOMETRY RECONSTRUCTION

To investigate the proposed coding scheme, image sets of three stuffed animals are recorded. A turntable and a digital camera on a computer-driven lever arm are used to capture object appearance from multiple viewpoints (Figs. 1,2).

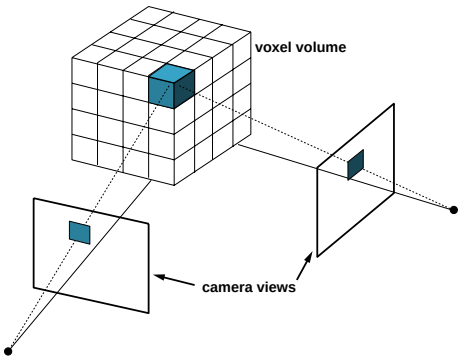

Figure 3. Volumetric geometry reconstruction[11]: the solid voxel volume enclosing the object is projected into all images; voxels are iteratively removed until the remaining voxels are consistent with all images.

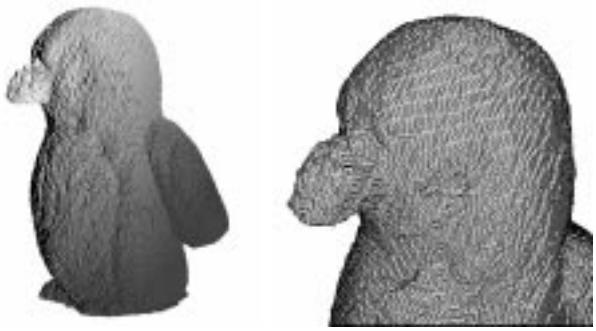

Figure 4. Reconstructed object geometry: the 3-D volume model consists of numerous cube-shaped voxels; triangulating the surface yields hundreds of thousands of triangles.

Figure 5. Image segmentation: the reconstructed geometry model is used to segment object pixels from background.

Prior to reconstructing geometry, recording positions and camera parameters must be calibrated. A cube of known size and texture is imaged from the same camera positions as the object images. A computer model of the textured cube is matched to the calibration images, and the internal and external camera parameters are calculated.[12]

Object geometry must be directly inferred from recorded images. A robust scheme to reconstruct 3-D geometry of a static scene from multiple camera views is described in.[11] The algorithm discretizes the volume enclosing the scene and projects voxels into all images (Fig. 3). For each voxel, color hypotheses are generated and checked for consistency. A voxel is removed if none of its color hypotheses is in accord with all image projections. Iteration over all voxels ends when no further voxel has to be removed (Fig. 4).

Because the voxel model represents the highest-resolution geometry model available, object pixels are segmented from background by projecting the model silhouette into all images (Fig. 5). Coder evaluation is performed with regard to the segmented images.

Triangulating the voxel model surface yields several hundred thousand triangles, so scene geometry must be approximated for efficient coding.

3. GEOMETRY CODING

Approximate geometry containing fewer triangles can be obtained by reducing the initial surface mesh.[13] However, mesh topology cannot be controlled during reduction. Because a very efficient texture map parametrization can be found for octahedron-based geometry models (Sec. 4), approximate geometry models are generated starting from an initial octahedral mesh (Fig. 6): the octahedron is placed at the center of the voxel model, and each vertex is

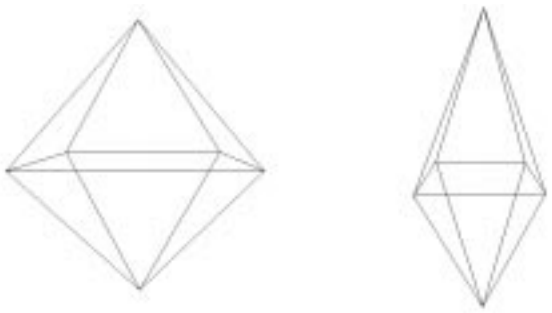

Figure 6. An octahedron is used to approximate object geometry by moving its vertex positions to the voxel volume's surface.

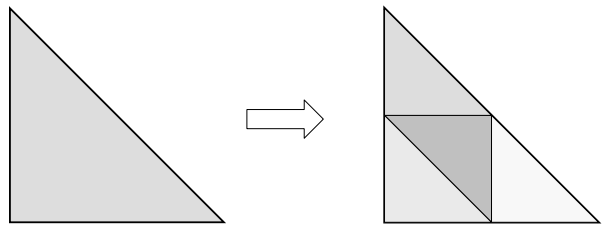

Figure 7. Geometry triangles are subdivided by inserting new vertices at edge midpoints.

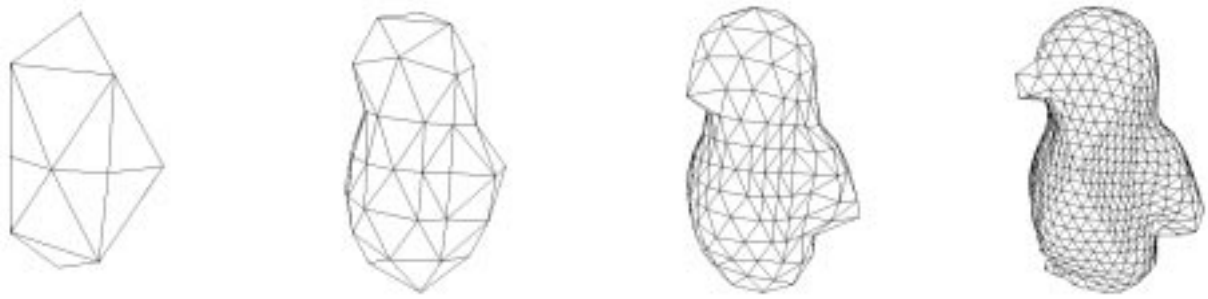

Figure 8. Approximate geometry models of the *Penguin* consisting of 32, 128, 512 and 2048 triangles.

moved in radial direction until it hits the model surface, yielding a very coarse geometry approximation. The mesh is then refined by inserting new vertices at edge midpoints, subdividing each triangle into four new triangles (Fig. 7). Vertex normals are calculated from adjacent triangles, and the new vertices are again moved to the voxel model's surface to obtain a better geometry approximation. By repeatingly subdiving triangles and refining vertex positions, increasingly accurate geometry models are obtained (Fig. 8). After n subdivisions, the resulting mesh consists of 2^{3+2n} triangles. Approximate geometry is coded using the *Embedded Mesh Coding*[14] algorithm.

4. TEXTURE MAP GENERATION

To attain best coding efficiency, a texture map must completely fill a rectangular region while preserving polygon connectivity. Both conditions can be met by suitably mapping an *octahedron* onto a square planar region (Fig. 9). The octahedron's closed 2-D surface is cut along 4 edges originating from one vertex, and the four-fold split vertex is assigned to the texture map's 4 corners. The 4 equatorial vertices are mapped to border midpoints, and the last vertex is assigned to the center. By subdividing texture-map triangles in the same way as previously described for geometry triangles (Sec. 3), one obtains texture maps for more accurate geometry meshes (Fig. 10).

While these texture maps fill the entire support, geometry triangle size and shape is not observed. To fit texture triangle size to the actual 3-D model, vertex positions in the texture map are optimized. The geometry model's total surface area A_{tot}^{geo} is determined, and each triangle is assigned the ratio of its size A_i^{geo} to total surface area. The texture triangles A_i^{txt} are normalized with regard to total texture map size A_{tot}^{txt}. The energy function

$$E = \sum_{i=0}^{N} \left(1 - \frac{R_i^{geo}}{R_i^{txt}}\right)^2 \tag{1}$$

over all N triangles with

$$\begin{aligned} R_i^{geo} &= A_i^{geo}/A_{tot}^{geo} \\ R_i^{txt} &= A_i^{txt}/A_{tot}^{txt} \end{aligned}$$

Figure 9. An octahedron can be mapped onto the texture plane without holes, completely filling a square region.

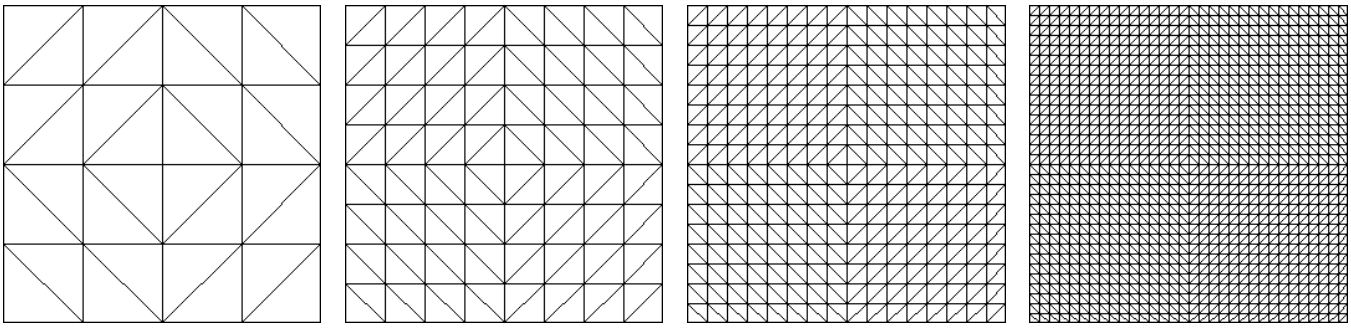

Figure 10. By triangle subdivision, texture maps can be generated for more accurate geometry models.

yields a quality measure for the texture map's vertex distribution. The function's singularity at $R_i^{txt} = 0$ prevents triangles from flipping and avoids triangle overlap.

Except for the 4 corner vertices, all texture vertices are considered for optimization. Movement of border vertices is restricted along the corresponding side, while 8 directions are tested for free vertices. Each vertex is moved by a small amount, sizes of affected triangles are determined, and the cost functional in Eq. (1) is evaluated. Vertex position is updated according to the lowest energy value. All vertices are repeatedly tested until no more positional changes occur.

The resulting texture triangles have about the same relative size as the triangles of the geometry model. Fig. 11 depicts optimized texture maps for different model approximations of the *Penguin* data set. Scene images can now be transformed to texture maps.

5. TEXTURE CODING

Mapping image pixels onto the texture plane results in an uneven distribution of texture information: occluded triangles cause empty texture map regions, triangles seen at a grazing angle lead to sparsely filled areas, while for face-on triangles, neighboring image pixels can fall onto the same texture map element (texel) (Fig. 12).

To render from arbitrary viewpoints, missing texture information must be interpolated. Coinciding pixel mappings can be avoided by choosing a suitably large, high-resolution texture map. As a result, one needs to code substantially more texels than there are object pixels in the original images. However, the amount of information contained in the texture maps is the same as in the segmented object images.

To eliminate the adverse effect of large texture maps on compression efficiency, a subband pyramid scheme is applied. By appropriate interpolation, texture-map holes can be filled without introducing additional high-frequency components, leaving subband-coding efficiency unchanged.

For coding, the 2-D texture maps are arranged in a two-dimensional array. The resulting 4-D data structure is used to interpolate missing texture information from nearby pixels within the same texture map as well as from pixels in adjacent texture maps (Fig. 13). The interpolation method is adapted to the following subband decomposition scheme

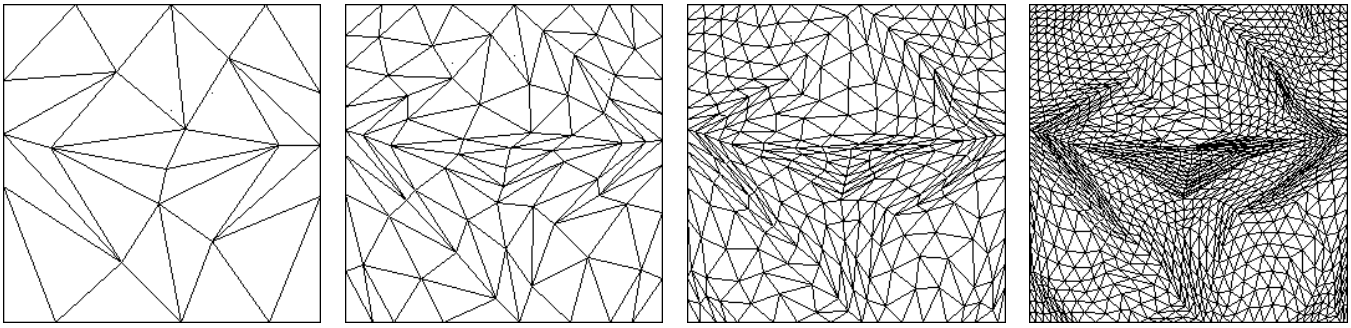

Figure 11. During texture map optimization, vertices are moved to preserve relative triangle size.

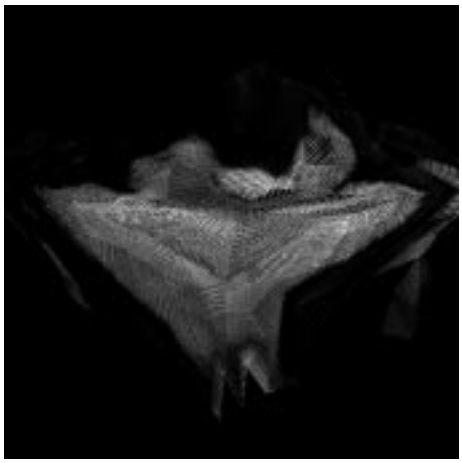

Figure 12. Mapping image pixels onto the texture plane leads to sparsely distributed texture information.

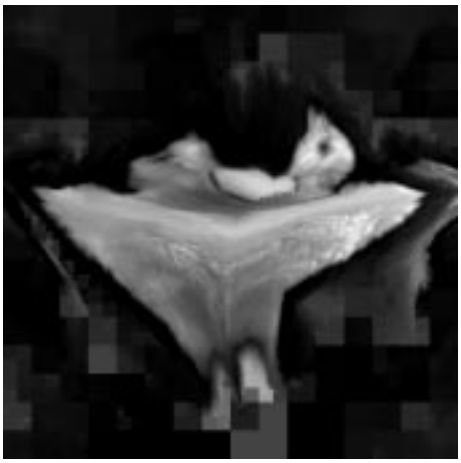

Figure 13. Missing texture information is interpolated from texels within the same texture map as well as from neighboring maps.

to avoid high-frequency coefficients. Subband coding can be straightforwardly extended to multiple dimensions, exploiting correlations within as well as between texture maps. A wavelet kernel is employed to decompose the texture-map array into octave bands along all 4 dimensions. To code the resulting wavelet coefficients, the *Set Partitioning in Hierarchical Trees (SPIHT)* algorithm is extended to 4 dimensions.[10] Coefficients are coded in order of importance, with large-magnitude coefficients coded early on, while small coefficient values are reproduced later in the bitstream. Correlations between wavelet coefficients in different subbands of the 4-D coefficient array are exploited to code the positional arrangement of the magnitude-ordered coefficients. The modified SPIHT coder also exhibits attractive decoding properties with regard to rendering applications: any texture map can be directly reconstructed, providing fast random access to arbitrary data segments, and texture information is progressively reconstructed, which allows continuous adjustment of reconstruction quality vs. decoding time.

6. RESULTS

The model-based coder is evaluated using three different image sets. Each test data set consists of 256 24-bit color images, recorded at 768×576-pixel resolution. The images cover an entire half-sphere around the object (Fig. 1).

Prior to coding, object pixels are segmented from background using the reconstructed voxel model (Fig. 5). As background cannot be coded using model geometry, only object pixels are considered for performance evaluation. Approximate geometry models with 32, 128, 512, 2048 and 8192 triangles are generated and coded as described in Sec. 3. Optimized texture mappings are found for all geometry models, Sec. 4. Texture map size is restricted to 256×256 texels due to limited computational resources, causing some coinciding pixel mapping and limiting maximum achievable reconstruction quality to ≈ 41 dB for the *Penguin*, ≈ 36 dB for the *Garfield*^{TM}, and ≈ 35 dB for the *Mouse* image set.

All images are converted to sparse texture maps. Missing texture information is interpolated by downsampling and averaging over known texture pixels. As is common in image compression, color is transformed to YUV space, and both chrominance components are downsampled by a factor of 2 prior to coding.

The interpolated texture maps are decomposed into subbands for each color channel separately. The Haar wavelet as well as a 5-tap QMF wavelet adopted from[10] are evaluated. The wavelet coefficients are coded using the modified SPIHT coder (Sec. 5). The progressive coding scheme allows decoding the bitstream up to different bit-rates, yielding different texture reconstruction qualities. Due to the interpolation scheme applied to fill missing texture, the Haar wavelet yields better texture reconstructions than the 5-tap QMF wavelet. While other wavelets are known to yield better SPIHT coding results, optimal interpolation of the irregularly distributed texture information proves non-trivial for more complex wavelets.

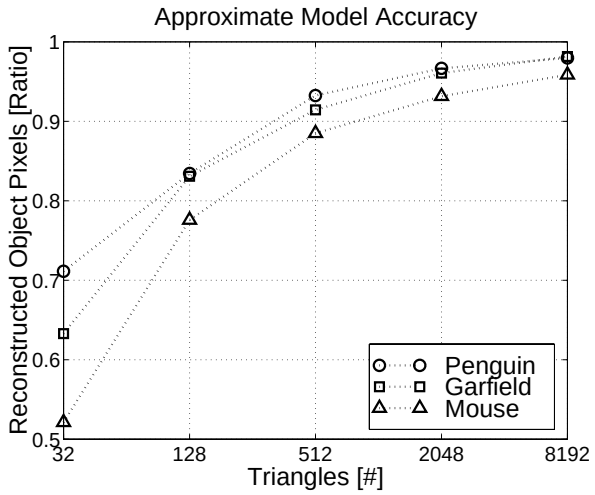

Figure 14. Percentage of reconstructable object pixels for different model approximations.

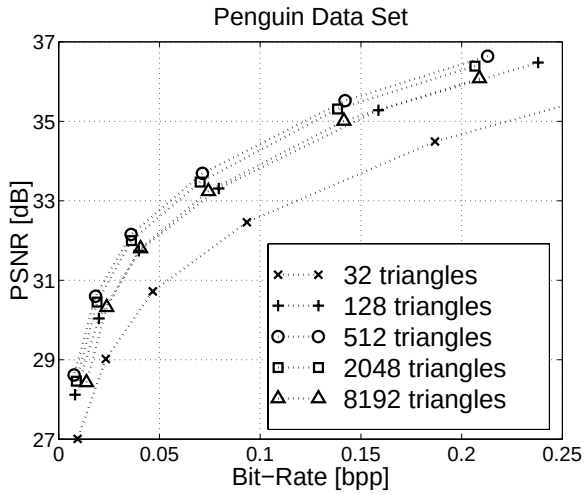

Figure 15. Rate-distortion performance for the *Penguin* data set.

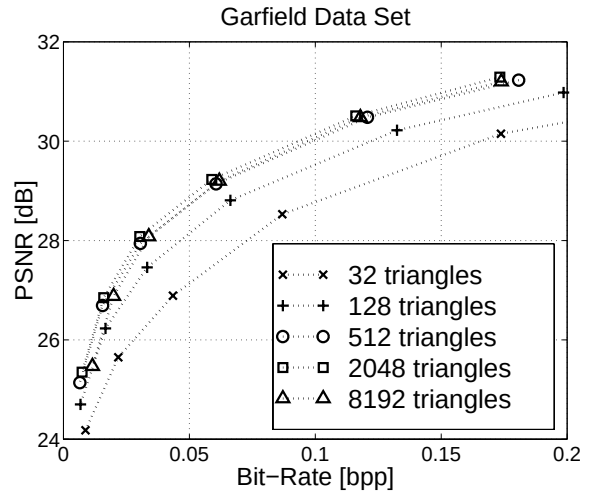

Figure 16. Rate-distortion performance for the *Garfield*^{TM} data set.

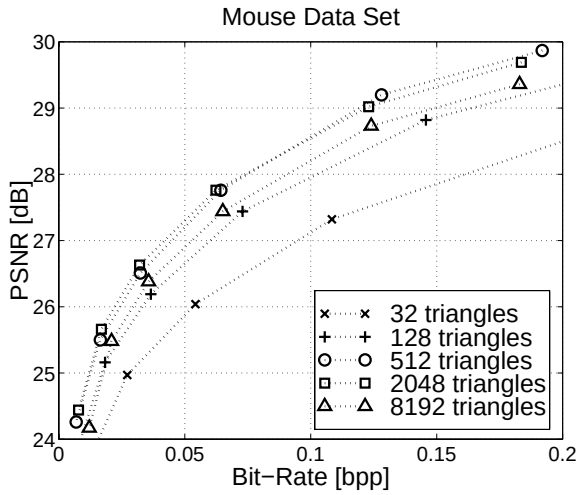

Figure 17. Rate-distortion performance for the *Mouse* data set.

The reconstructed texture maps are used in conjunction with the corresponding geometry model to render the object from the original images' recording positions. Coding performance is evaluated by comparing the rendering result to the original image recording. To quantify reconstruction quality, the luminance component's *Peak-Signal-to-Noise-Ratio (PSNR)* is measured over all reconstructed object pixels and averaged over all images in the data set. Because approximate geometry cannot reconstruct all segmented object pixels (Fig. 14), coding bit-rate is evaluated in relation to the number of reconstructed object image pixels and expressed in *bits per (reconstructed object) pixel (bpp)*.

Rate-distortion performance of the model-based coder is depicted in Figs. 15,16,17. Bit-rate includes texture-map as well as geometry coding. Coding efficiency depends on model accuracy: while coarse model geometry requires a lower bit-rate, texture maps show greater differences due to residual disparity. Best bit-rate allocation between geometry and texture is attained using 512 triangles to approximate *Penguin* geometry, while 2048 triangles yield best coding performance for the *Garfield*^{TM} and *Mouse* data sets. At high bit-rates, geometry coding bit-rate is negligible, yet the most accurate model consisting of 8192 triangled performs worse than coarser approximations, especially for the *Mouse* and *Penguin* data sets; higher geometry accuracy corresponds to increased model surface

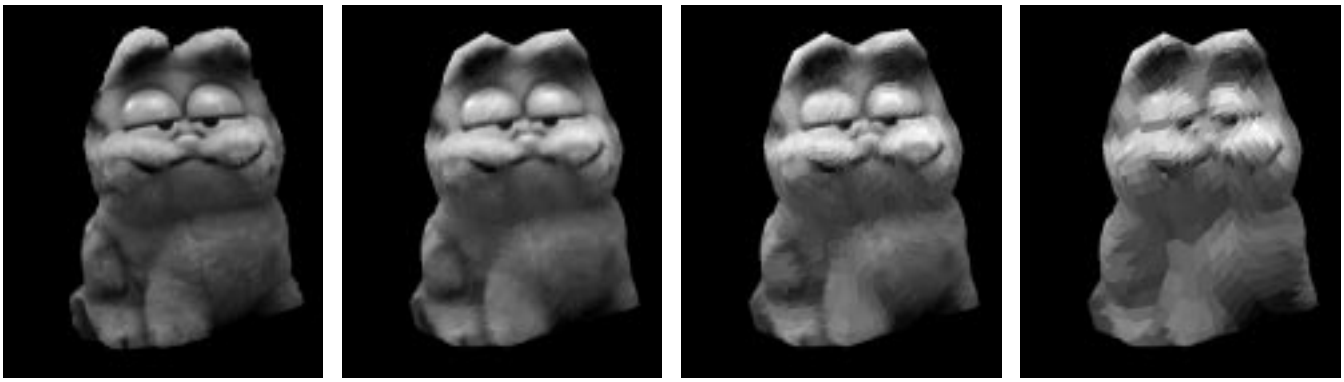

Figure 18. Original *Garfield*^{TM} image and reconstructions at 0.17 bpp, 0.06 bpp and 0.016 bpp, rendered from 2048 triangles.

area, which is mapped onto the constant-sized texture map, causing more coinciding pixel mappings while increasing coded object area.

Reconstruction quality for the *Garfield*^{TM} data set can be judged from Fig. 18. At low bit-rates, texture artifacts become apparent, caused by the Haar wavelet used to compress the texture maps. While approximate geometry cannot reproduce the exact silhouette, moderately accurate approximations suffice to yield smooth contour appearance (Fig. 19).

7. CONCLUSION

A codec for multi-view images has been presented that uses approximate scene geometry to generate texture maps. Texture maps are disparity-compensated representations of the original image data, yielding increased compression efficiency. Approximate geometry is sufficient to achieve high compression at moderate reconstruction quality. A progressive coding scheme allows trading reconstruction quality vs. decoding time and features random access to texture information during decoding. The decoder provides 3-D scene geometry and multiple texture maps which can be directly used for conventional geometry-based rendering.

Due to the applied texture map interpolation scheme, the subband decomposition is performed using the Haar wavelet. Enhanced coding performance is expected if more sophisticated interpolation schemes allow using multi-tap wavelet kernels. Larger texture map sizes can be applied to avoid coinciding pixel mapping which requires greater computational resources for coding, as complete texture must be kept in local memory for 4-D subband decomposition. On the decoding side, hardware requirements remain minimal: the bitstream can be easily stored and transmitted, e.g., over the Internet, and the compressed data fit into local memory. Without decoder optimization, 256×256 texels are currently reconstructed from the bitstream at 0.12 bits per pixel in less than 1 second on an SGI-O2.

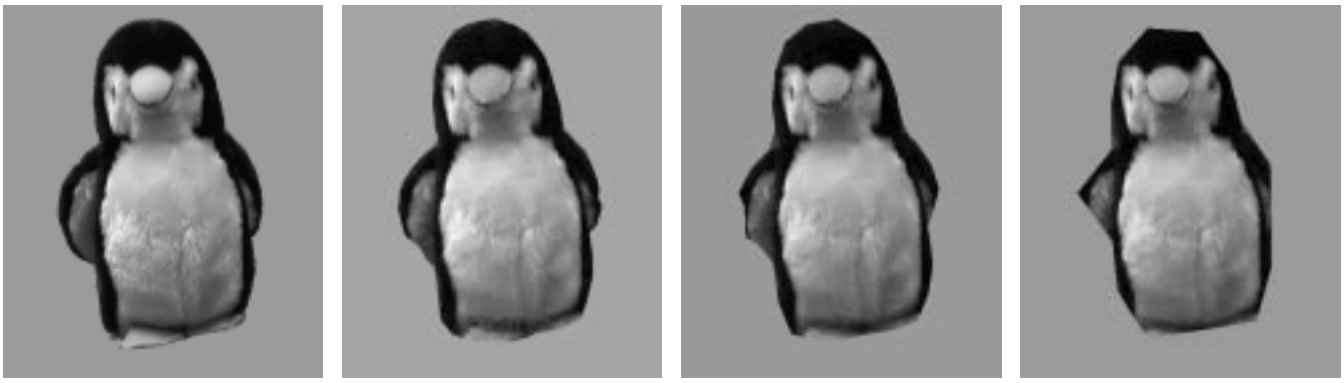

Figure 19. Original *Penguin* segmentation and reconstructions from 2048, 512 and 128 triangles at 0.14 bpp.

As approximate scene geometry consisting of some thousand triangles is rendered in negligible time using standard graphics hardware, the presented model-based coder promises to enable image-based rendering at interactive frame rates on low-end computer harware.

REFERENCES

1. SEGA Enterprises, Ltd., *Dreamcast^{TM} Video Game Console Manual*, 2000. http://www.sega.com/console/index.shtml.
2. M. Magnor and B. Girod, "Adaptive block-based light field coding," *Proc. International Workshop on Synthetic-Natural Hybrid Coding and Three Dimensional Imaging (IWSNHC3DI'99),* Santorini, Greece , pp. 140–143, Sept. 1999.
3. M. Magnor and B. Girod, "Hierarchical coding of light fields with disparity maps," *Proc. International Conference on Image Processing (ICIP-99),* Kobe, Japan , pp. 334–338, Oct. 1999.
4. M. Magnor and B. Girod, "Data compression for light field rendering," *IEEE Trans. Circuits and Systems for Video Technology* , Apr. 2000.
5. M. Levoy and P. Hanrahan, "Light field rendering," *Computer Graphics (SIGGRAPH '96 Proceedings)* , pp. 31–42, Aug. 1996.
6. S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The Lumigraph," *Computer Graphics (SIGGRAPH '96 Proceedings)* , pp. 43–54, Aug. 1996.
7. P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach," in *SIGGRAPH 96 Conference Proceedings*, H. Rushmeier, ed., Annual Conference Series, pp. 11–20, ACM SIGGRAPH, Addison Wesley, Aug. 1996.
8. P. Debevec, Y. Yu, and G. Boshokov, "Efficient view-dependent image-based rendering with projective texture-mapping," Technical Report CSD-98-1003, University of California, Berkeley, May 1998.
9. J. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients," *IEEE Trans. Signal Processing* **41**, pp. 3445–4362, Dec. 1993.
10. A. Said and W. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits and Systems for Video Technology* **6**, pp. 243–250, June 1996.
11. P. Eisert, E. Steinbach, and B. Girod, "Multi-hypothesis volumetric reconstruction of 3-D objects from multiple calibrated camera views," *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP'99)* Phoenix, USA , pp. 3509–3512, Mar. 1999.
12. R. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," *IEEE Journal of Robotics and Automation* **RA-3**, pp. 323–344, Aug. 1987.
13. H. Hoppe, "Progressive meshes," *Computer Graphics (SIGGRAPH '96 Proceedings)* , pp. 99–108, Aug. 1996.
14. M. Magnor and B. Girod, "Fully embedded coding of triangle meshes," *Proc. Vision, Modeling, and Visualization (VMV'99),* Erlangen, Germany , pp. 253–259, Nov. 1999.