Multi-View Coding for Image-Based Rendering Using 3-D Scene Geometry

Marcus Magnor, *Member, IEEE*, Prashant Ramanathan, *Student Member, IEEE*, and Bernd Girod, *Fellow, IEEE*

***Abstract*—To store and transmit the large amount of image data necessary for Image-based Rendering (IBR), efficient coding schemes are required. This paper presents two different approaches which exploit three–dimensional scene geometry for multi-view compression. In texture-based coding, images are converted to view-dependent texture maps for compression. In model-aided predictive coding, scene geometry is used for disparity compensation and occlusion detection between images. While both coding strategies are able to attain compression ratios exceeding 2000:1, individual coding performance is found to depend on the accuracy of the available geometry model. Experiments with real-world as well as synthetic image sets show that texture-based coding is more sensitive to geometry inaccuracies than predictive coding. A rate-distortion theoretical analysis of both schemes supports these findings. For reconstructed approximate geometry models, model-aided predictive coding performs best, while texture-based coding yields superior coding results if scene geometry is exactly known.**

***Index Terms*—Geometry coding, image-based rendering (IBR), light field compression, model-based coding, multi-view coding analysis, multi-view compression.**

I. Introduction

FROM Internet museum tours and virtual city sightseeing to three–dimensional (3-D) product presentations and computer games, Image-based Rendering (IBR) techniques can be used to create photo-realistic representations of remote real-world or computer-generated places and objects [1]–[9]. Visual quality thereby depends on the number of scene images available, and since hundreds to thousands of images are typically necessary to obtain convincing rendering results [10], efficient multi-view coding techniques are needed to store IBR data, or to transmit multi-view imagery over a network, such as the public Internet.

In recent years, a number of multi-view compression schemes have been developed specifically for use in conjunction with image-based rendering applications. Among the various coding techniques employed are vector quantization [3], discrete cosine transform (DCT) coding [11], wavelet coding [12]–[16] predictive image coding [17]–[19], as well as approaches based on video coding standards [15]. To achieve interactive rendering frame rates, these coders are designed to feature fast decoding performance which, however, limits coding efficiency to a range from about 20:1[3] to 300:1[19]. For storage and Internet transmission, higher compression ratios are desirable. Here, block-adaptive coding and hierarchical disparity compensation techniques have been shown to yield compression ratios exceeding 800:1 [20].

Coding efficiency, decoding speed, and rendering quality can be increased considerably if 3-D scene geometry information is available [8], [21], [22]. This paper describes two different ways in which knowledge of scene geometry can be employed to encode multi-view imagery. In texture-based coding, scene geometry is used to convert images to view-dependent texture maps prior to compression [5], [23], [24], [8], [25]. These view-dependent texture maps exhibit greater inter-map correlation than the original images, making them more amenable to coding. In model-aided predictive coding, on the other hand, images are successively estimated by employing scene geometry to predict new views from already encoded images [26]. The residual prediction error between the image estimate and the actual image recording is additionally encoded [27]. By predicting multi-view images in a hierarchical fashion, the image data is available in a multiresolution representation during decoding. Since the presented coding schemes aim at highest compression performance for data storage and transmission purposes, the light field data is decoded off-line prior to using the images for rendering.

The paper is organized as follows. After outlining multi-view image data acquisition, the process for reconstructing and encoding the geometry model is explained. Both the model-aided and the progressive texture-based coding schemes are described. Coding results for real-world as well as synthetic image sets are presented. For both codecs, the influence of scene geometry accuracy on coding efficiency is experimentally investigated. A theoretical analysis of coding efficiency in the presence of geometry and image noise concludes the paper.

II. Multi-View Image Acquisition

To investigate both multi-view coding schemes described in the following sections, image sets have been acquired from three stuffed toy animals (*Garfield*, *Mouse*, and *Penguin*, Fig. 1). Calibrated images are captured using a computer-controlled turntable and a digital camera on a lever arm, acquiring object appearance at 384×288-pixel resolution and 24-bit RGB color information per pixel. Image recording positions are distributed on a hemisphere around the object with an angular step size of $\sim\ 11.25^\circ$ in horizontal and vertical direction, yielding a total of 32×8 images around the object, plus one additional image from the zenith perspective. Before and after recording each object, a calibration body is recorded from the

Manuscript received November 2001; revised June 2003.

M. Magnor is with the Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany (e-mail: magnor@mpii.de).

P. Ramanathan and B. Girod are with Stanford University, Stanford, CA 94305-9515 USA (e-mail: pramanat@stanford.edu; bgirod@stanford.edu).

Digital Object Identifier 10.1109/TCSVT.2003.817630

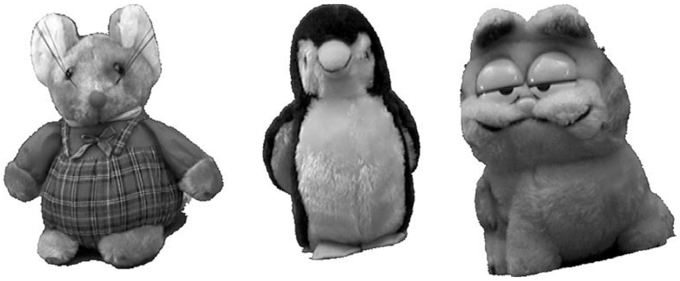


Fig. 1. Real-world test objects. Multi-view imagery is recorded from stuffed toy animals.

same viewpoints as the object images. As the turntable and the camera arm are driven by stepper motors, image recording positions can be reproduced with sufficiently high accuracy for calibration. For calibration, an analysis-by-synthesis method is employed [28]: A synthetic computer model of the calibration body is automatically fit to the calibration images to determine extrinsic as well as intrinsic camera parameters. The model is repeatedly rendered while an optical flow-based optimization algorithm varies camera parameters until the rendered model matches the actual image of the calibration object.

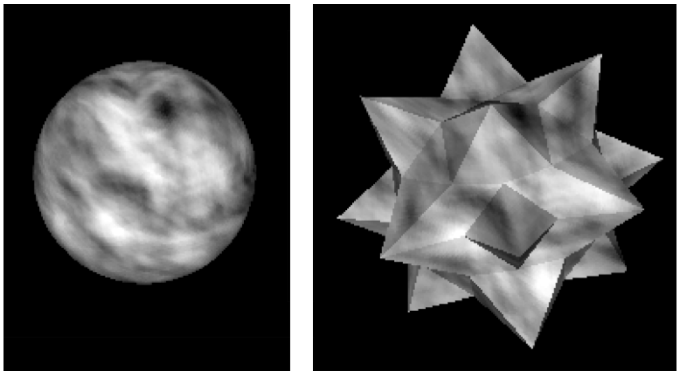
To evaluate coding performance for the case of exactly known geometry, synthetic multi-view data sets are also used (Fig. 2). In the *Sphere* data set, the geometry model of a sphere approximation consisting of 8192 triangles is rendered from multiple viewpoints. The *Star* geometry model consists of 128 vertices approximating a sphere of which 18 vertices protude by 50% to form the spikes. Both geometry models are gray-scale textured using a synthetic texture map exhibiting an inter-pixel intensity correlation of 0.98. The synthetic objects have diffusely reflecting surfaces (Lambertian reflection), while a directional light source illuminates the objects always from the direction of the camera (eye-light). Both synthetic image sets consist of 257 images, rendered from viewpoints spaced corresponding to the real-world image sets.

III. 3-D Geometry Reconstruction and Geometry Coding

In geometry-based multi-view coding, scene geometry must be encoded in addition to image data. This section describes the procedure for reconstructing 3-D scene geometry from images, in the case of real-world images, and for encoding the geometry, for both real-world and synthetic data sets.

A. Geometry Reconstruction

While for the computer-generated *Sphere* image set exact 3-D geometry is available, for the real-world objects geometry must be inferred from the recorded images. To reconstruct 3-D scene geometry directly from calibrated multi-view image data, several volumetric algorithms have been proposed [29]–[32]. In contrast to methods that rely on distinct image features [33]–[35], volumetric reconstruction does not require the explicit identification of correspondences between images. For the presented experiments, the *Multi-Hypothesis Volumetric Reconstruction* (MHVR) algorithm is used to derive 3-D scene geometry [32]. The algorithm is based on discretizing the space containing the object into volume elements (*voxels*). A voxel model is constructed directly from multiple calibrated images (Fig. 3). Reconstruction accuracy depends on image calibration precision, object surface characteristics, and voxel size. In the presented work, the discretized volume is comprised of 200^3 to 300^3 voxels, where the spatial extent of each voxel corresponds to approximately 1 pixel when projected into the image plane. The reconstruction of a 300^3-voxel model takes several hours using all 257 images on a conventional PC.



(a) (b)

Fig. 2. Synthetic test image data. The diffusely reflecting models of (a) a *Sphere* and (b) a *Star* are illuminated from the camera direction.

To eliminate the influence of nonmodelled background on coding performance measurements, the reconstructed voxel models are used to segment the *Garfield*, *Mouse*, and *Penguin* images. For the synthetic *Sphere* image set, no segmentation is necessary as the object silhouette exactly matches the geometry model projection. Exact object geometry is available for the *Sphere* images, whereas for the real-world image sets only finite-precision geometry models can be reconstructed.

B. Rate-Constrained Geometry Coding

A reconstructed volumetric model typically consists of millions of voxels. To efficiently compress and process object geometry, a triangle-mesh description of the object's surface is desirable. The *Marching Cubes* algorithm [36] is therefore used to triangulate the voxel model surface (Fig. 3). The resulting mesh still contains hundreds of thousands of triangles, however, many more than are necessary to represent object geometry at the level of accuracy of the reconstructed model. Consequently, the *Progressive Meshes* (PM) algorithm [37] is employed to reduce the number of triangles until the maximum distortion of the resulting mesh corresponds to half the size of a voxel (Fig. 3). This way, triangle mesh accuracy is matched to the original reconstructed voxel model, and the number of triangles in the mesh is reduced to approximately 10 000 triangles.

While better geometry accuracy can increase image coding efficiency, it inevitably also increases geometry coding bit rate. To determine the point of best overall coding performance, the geometry model must be encodable at different levels of accuracy and with correspondingly different bit rates. A number of progressive mesh-coding algorithms are known that allow trading off geometry reconstruction accuracy versus geometry coding bit rate [37]–[40]. Unfortunately, these algorithms encode only mesh connectivity in a progressive fashion, while

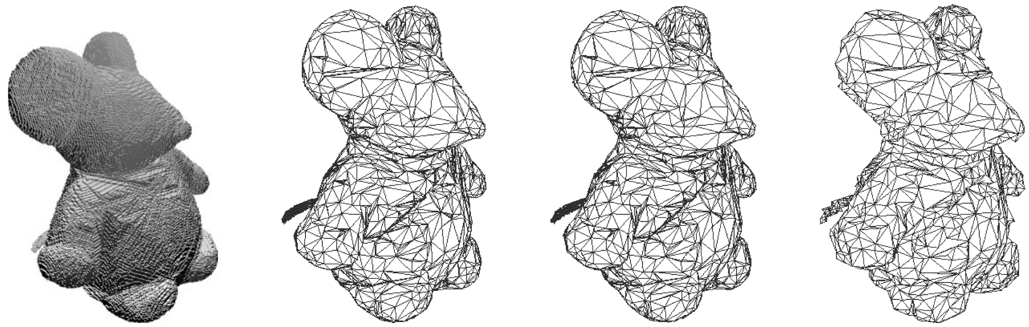


Fig. 3. A voxel model is reconstructed from the *Mouse* image set, the surface is triangulated, the triangle mesh reduced, and the resulting geometry model is EMC encoded. By decoding only a fraction of the bitstream, the accuracy of the reconstructed geometry model can be continuously varied, allowing us to trade coding bit rate for geometry accuracy. Model accuracy is expressed as the number of bits n used to quantize vertex coordinates. The maximum deviation of a vertex from its original position is $2^{-(n+1)}$ relative to overall bounding-box size. From left to right: reconstructed voxel model; triangulated surface, decimated mesh; 8-bit accuracy 95 183 bits; and 5-bit accuracy 33 894 bits.

vertex coordinates are encoded with fixed accuracy. Especially at coarse geometry resolution, most bit rate is spent for expressing precise vertex positions, even though reconstructed geometry accuracy would require substantially less accurate vertex coordinates.

C. Embedded Mesh Coding

The *Embedded Mesh Coding* (EMC) algorithm progressively encodes mesh connectivity as well as vertex coordinates simultaneously [41]. The geometry coding scheme is based on vertex connectivity and an oct-tree representation of vertex coordinates by introducing multiple resolution levels. This way, EMC allocates available coding bit rate evenly between mesh connectivity and vertex positional information.

The EMC algorithm encodes vertex coordinates and mesh connectivity simultaneously at continuously increasing level of detail. The complete bitstream contains all information to faithfully reconstruct original vertex positions and connectivity. But EMC coding can also be interrupted at any point, yielding a truncated bitstream that still allows the reconstruction of an approximate geometry model. During decoding, the number of triangles and vertices as well as vertex positional accuracy increases continuously, yielding increasingly accurate object geometry. The decoding process can also be stopped at any point to obtain approximate geometry representations.

By using EMC in conjunction with multi-view coding schemes, geometry coding bit rate can be continuously varied, which enables optimal allocation of bit rate between geometry and image information. Because the EMC algorithm directly encodes vertex connectivity instead of triangle connectivity, EMC can be used to encode polygonal meshes of arbitrary topology and dimension, regardless of whether the mesh is orientable, regular, manifold, or nonmanifold.

IV. Model-Aided Predictive Coding

Given camera parameters and 3-D scene geometry, disparity between images can be compensated and obscured image areas can be detected. The *Model-Aided Coder* (MAC) relies on successively predicting image appearance by disparity compensation and occlusion detection on a pixel basis [27]. The images are hierarchically ordered for encoding, yielding a multiresolution representation of the multi-view image set during decoding and rendering.

A. Model-Aided Prediction

In model-aided coding, an image is predicted by warping multiple reference images [42]. First, the geometry model is rendered for the image viewpoint that is to be predicted. Each image pixel is assigned its corresponding point on the surface of the 3-D model by determining the triangle index and the barycentric coordinates within the triangle (Fig. 4). The geometry model is then rendered for all reference image positions. For each pixel in the prediction image, the corresponding pixels in the reference images are sought using the pixel's triangle index and its barycentric coordinates. This way, pixels that are not visible in a reference image are automatically detected. A partially occluded image region is predicted only from those reference images that depict the respective region, and coinciding pixel predictions are averaged. Because multiple reference images are used for prediction, the number of completely invisible regions is small. These regions are filled by interpolation using a multiresolution pyramid of the predicted image estimate: Also known as push–pull interpolation [4], lower resolution versions of the image are used to look-up the mean color value of the local neighborhood around an unpredicted pixel position.

B. Hierarchical Image Coding Order

As image recording positions are distributed on the hemisphere around the scene (Fig. 5), they can be expressed in spherical coordinates with the origin at the scene's center. To efficiently exploit similarities among the images, and at the same time span the entire light field recording hemisphere early on during decoding, it was experimentally found that highest coding efficiency is achieved by encoding the images in the following hierarchical order [20], [43]: The image closest to the zenith of the hemisphere and four images evenly spaced around the equator are intra-encoded using the block-DCT scheme familiar from still-image compression (images A in Fig. 5). For each image, the DCT coefficients' quantization parameter Q is individually adjusted to ensure that the reconstructed image meets a preset minimum reconstruction quality $q_{\min}$. The five intra-encoded images are arranged into four groups, each

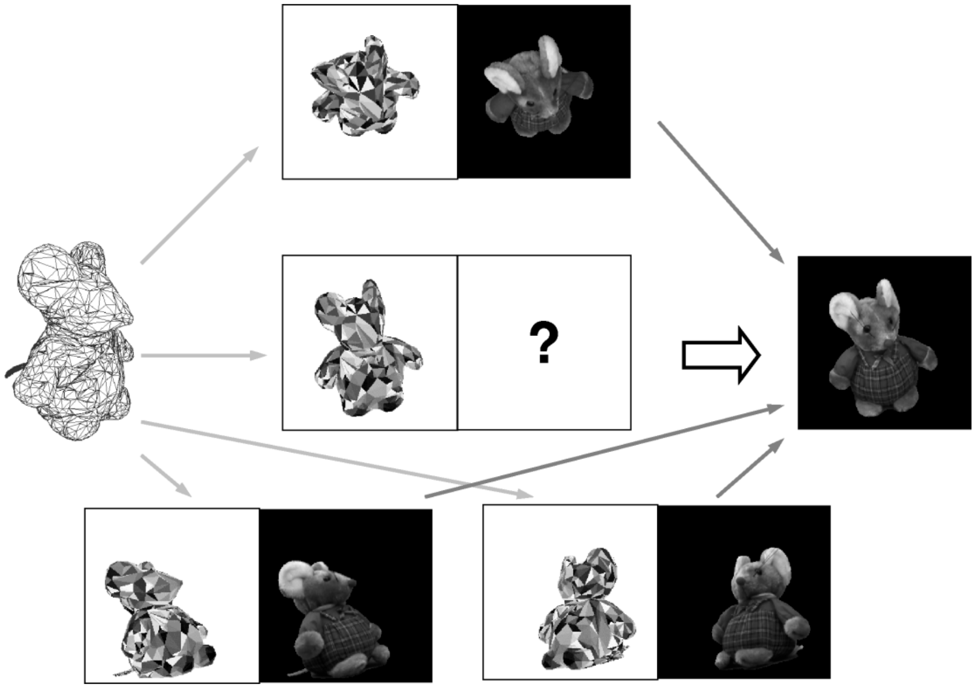


Fig. 4. Model-aided image prediction detects occlusions and performs disparity compensation from multiple reference frames.

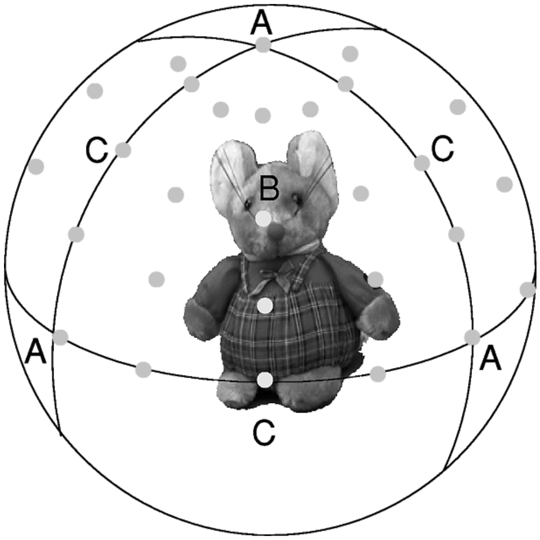


Fig. 5. Hierarchical multi-view coding order. Image recording positions are projected onto the hemisphere around the scene. The images closest to the zenith and four images along the equator are intra-encoded (images A). The image closest to the center of each quadrant (image B) is predicted from the quadrant's corner images, and mid-side images (images C) are predicted from the central and two corner images. Each quadrant is then subdivided and treated likewise until all images have been encoded.

consisting of the polar and two equatorial images, subdividing the hemisphere into four quadrants. In each quadrant, the image closest to the central position (image B in Fig. 5) is predicted by model-aided disparity compensation (Section IV-A), using the three corner images as reference. After prediction, the residual prediction error is DCT-encoded if image quality does not meet the desired reconstruction quality $q_{\min}$. Next, the three images closest to the mid-positions of the quadrant's sides (images C in Fig. 5) are predicted likewise using the just-encoded center image and the two closest corner images as reference. After all quadrants have been considered, each quadrant is divided into four smaller subregions: the center image B and the polar A image form together with each of the two mid-latitude images C two triangular sub-quadrants, while the center image B in conjunction with each of the equatorial A images and the two closest C images represent two more quadrangular subquadrants. Because each sub-quadrant's corner images are already encoded (the center image B, one corner image A, and one or two mid-side images C), these images are available for prediction of the center and side images of each subdivided region. After the center and mid-side images of all subquadrants have been encoded, each subquadrant is again subdivided in the above-described fashion. Quadrant subdivision continues recursively until all images are encoded. This way, an image pyramid of ever-decreasing mutual recording distances is established.

As all images are predicted from previously encoded images, a multilevel hierarchy is established among the image data. The hierarchical coding order provides short prediction distances to yield best prediction results, and during decoding, multiple resolution levels of the image set can be progressively accessed. For image-based rendering, the multiresolution representation allows adjusting rendering quality to available computational resources. To eliminate ghosting artifacts during rendering, model-aided prediction can also be used to supplement multi-view imagery by providing disparity-compensated intermediate image estimates [44], or by directly warping in-between viewpoints [21].

C. Coding Performance

The performance of the model-aided coder has been assessed using multi-view imagery from real-world scenes as well as a computer-generated image set (Section II). For encoding, the images are converted to $\mathrm{YC_bC_r}$ color space, and the chromi-

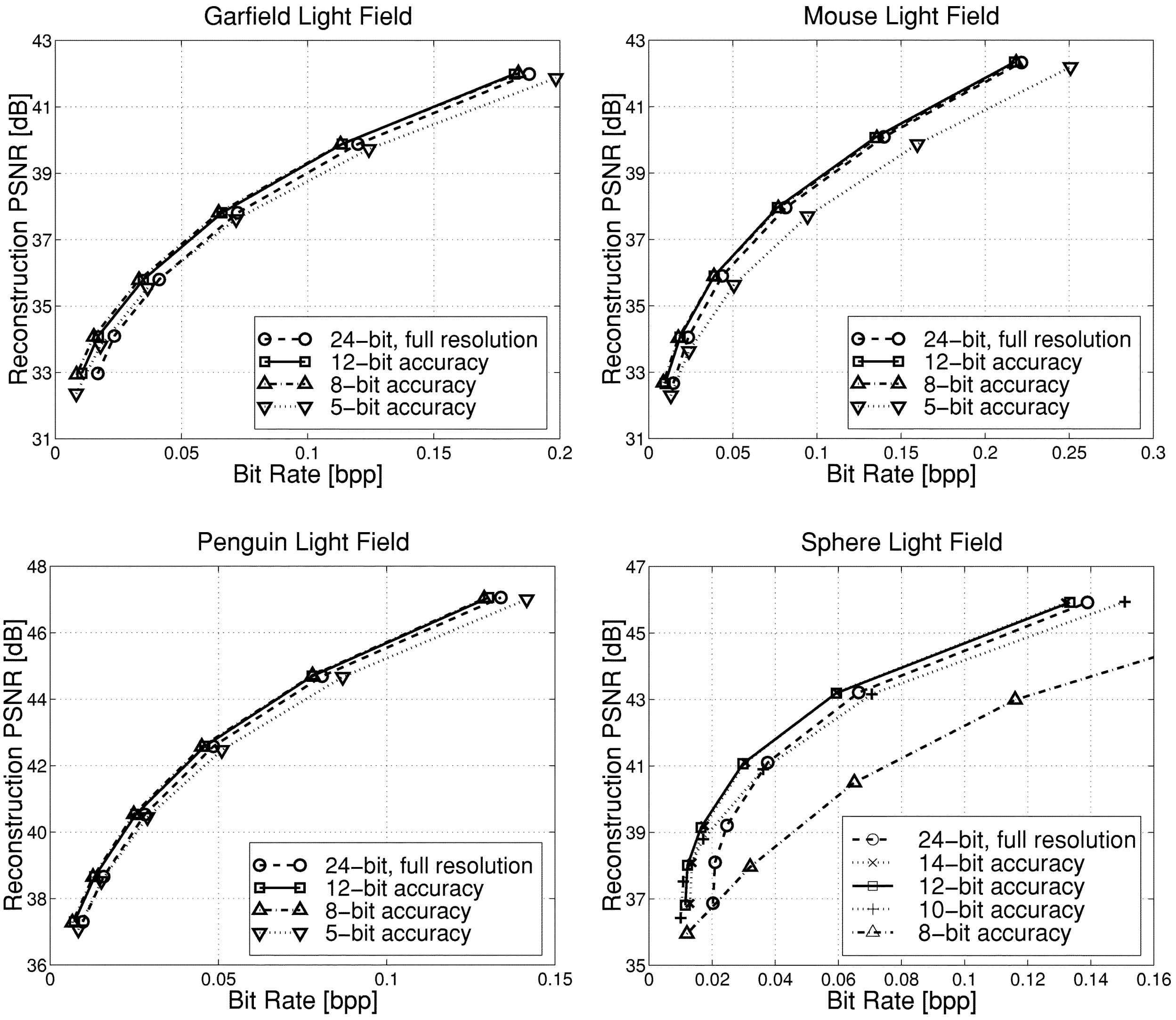


Fig. 6. Model-aided coding performance. Geometry-model accuracy is expressed in number of bits used to quantize vertex positions along each dimension.

nance components are downsampled by a factor of 2 both horizontally and vertically. Fig. 6 illustrates the rate-distortion performance of the model-aided coder for the multi-view image sets. Reconstruction quality is expressed as the peak-signal-to-noise ratio (PSNR) averaged over all image pixels, whereas bit rate is given in bits per pixel (bpp). EMC geometry coding bit rate is included in all curves.

For the *Garfield* image set, coding bit rate ranges from 0.0085 bpp at 32.9 dB reconstruction PSNR up to 0.182 bpp at 42.0 dB. Reconstruction quality of the *Mouse* images varies from 32.7 dB at 0.087 bpp to 42.3 dB at 0.218 bpp. The *Penguin* light field requires between 0.0067 bpp at 37.3 dB and 0.129 bpp at 47.0 dB. The synthetic *Sphere* images, finally, are encoded with 0.01 bpp at 36.4 dB and 0.133 bpp at 45.9 dB PSNR.

Different geometry approximations are evaluated to determine best bit-rate allocation. Except for the *Sphere* object, best coding performance is obtained if object geometry is approximated with 8-bit vertex positional accuracy. This corresponds to the resolution of the reconstructed volumetric models which consist of $\approx 2^8$ voxels along one side. More accurate vertex positions do not improve prediction performance but only increase geometry bit rate. The same approximate geometry model is optimal at high as well as at very low coding bit rates, indicating that the benefits from better geometry accuracy outweigh the bit-rate savings from using a more approximate geometry representation even at high compression.

In contrast to the real-world test objects, the synthetic *Sphere* image set shows optimal coding results at 10- to 14-bit accurate geometry. Because the original 8192-triangle *Sphere* geometry model is available, prediction quality can be expected to improve with approximation accuracy until the exact rendering geometry is recovered at 24-bit accurate vertex positions. At low bit rates, 10-bit accurate sphere geometry achieves best coding results, while improved prediction quality using 12- and 14-bit approximate models subsequently over-compensates geometry coding bit rate at higher reconstruction quality.

D. Reconstruction Quality

Fig. 7 depicts an image of the *Mouse* image set that is encoded at different bit rates using the 8-bit accurate geometry model. At 0.136 bpp overall coding bit rate and 40.1 dB PSNR,

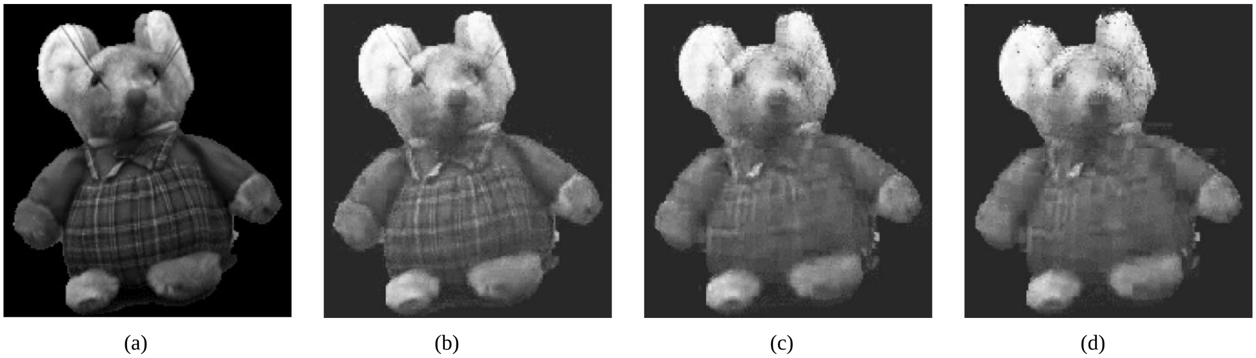


Fig. 7. Image from the *Mouse* set, MAC encoded with 8-bit accurate geometry at different bit rates. (a) Original image. (b) 0.136 bpp at 40.1 dB. (c) 0.018 bpp at 34.0 dB. (d) 0.009 bpp at 32.7 dB.

the reconstructed *Mouse* image shows only marginal differences from the original image set, visible merely in the finely textured pants. When increasing the compression ratio, the pants pattern washes out and becomes blurry. At 34.0 dB PSNR, corresponding to 0.018 bpp, the fine whiskers cannot be resolved anymore. At the lowest reconstruction PSNR of 32.7 dB, the silhouette of the depicted object still shows up as a crisp outline. The mosquito artifacts outside the object's silhouette are caused by the block-based DCT scheme applied to encode the residual error.

V. Texture-Based Coding

By transforming images of a 3-D object into texture maps, disparity-induced differences between the images can be eliminated. If nonvisible regions are suitably interpolated, texture maps exhibit higher inter-map correlation than images because only reflection properties of the object surface can cause any remaining variation between texture maps generated from different viewpoints. Texture-based coding is inspired by view-dependent texture mapping techniques developed in IBR research [23], [8], [25].

In *Progressive Texture-based Coding* (PTC), reconstructed 3-D object geometry is used to convert images to view-dependent texture maps. After undefined regions in the texture maps are suitably interpolated, a wavelet coding scheme is employed to encode the texture information while simultaneously exploiting texture correlation within as well as between texture maps. The progressive coding technique continuously increases attainable reconstruction quality with available bit rate. Because PTC is based on texture information, only the image regions within the projected geometry model silhouette are encoded. Decoding of the progressive bitstream can be interrupted at any point of time which allows trading rendering quality for frame-rate and, thus, adapting rendering performance to available computational resources.

A. Texture Map Generation

Object surfaces are most commonly described by piecewise planar triangle meshes in computer graphics. While surface texture can be easily parameterized over each triangle separately, individual triangle-texture patches are very inefficient to encode as correlation across adjacent triangles cannot be exploited. To achieve more efficient texture coding results, a connectivity-preserving mapping of object surface onto a rectangular texture map is necessary, preserving texture correlation. In the following, a suitable triangle mesh parameterization is described for objects that exhibit sphere-like topology.

B. Geometry Model Generation for Texture Mapping

To parameterize the closed surface of a volumetric object such that a planar two–dimensional (2-D) texture map can be generated, the surface must be cut once or several times, depending on the body's topological genus. For objects having genus 0, which are topologically equivalent to a sphere, a simple rectangular surface parameterization can be obtained by starting from the simple shape of an *octahedron*, Fig. 8. By opening the octahedron along two edges from pole to pole and along two edges on the equator and only mildly distorting the shape of the individual triangles, the octahedron's eight triangles are mapped onto a planar square region while preserving triangle connectivity for eight of the 12 octahedral edges [25]. In this manner, an unambiguous, connected parameterization of the entire octahedral surface is obtained, yielding a rectangular, completely filled texture map.

To approximate the volumetric geometry model, the octahedron is placed at the center of the voxel model. Each vertex is moved along its normal direction until it lies on the voxel model surface, obtaining an octahedron-based coarse approximation of the object's 3-D shape (Fig. 8). More accurate geometry is obtained by subdividing the triangles, inserting new vertices at the edge midpoints to create four new triangles for each previous triangle. Vertex normals are inferred from the orientation of the adjacent triangles, and the new vertices are relocated to the voxel model surface. By repeated triangle subdivision, increasingly accurate triangle meshes are obtained (Fig. 8), so that after n subdivisions, the mesh consists of 2^{3+2n} triangles [25].

C. Texture Map Optimization

To map the refined geometry meshes onto the texture plane, the initial texture-map triangles are subdivided in the same way as the geometry triangle mesh (Fig. 9). Each texture-map triangle corresponds to one geometry triangle. The texture-map triangles are all identical, however, while the geometry triangles differ in size and shape. To minimize coinciding pixel mappings, which limits attainable reconstruction quality, relative texture-map triangle size is matched to their corresponding geometry triangle area by iteratively shifting texture-map vertex

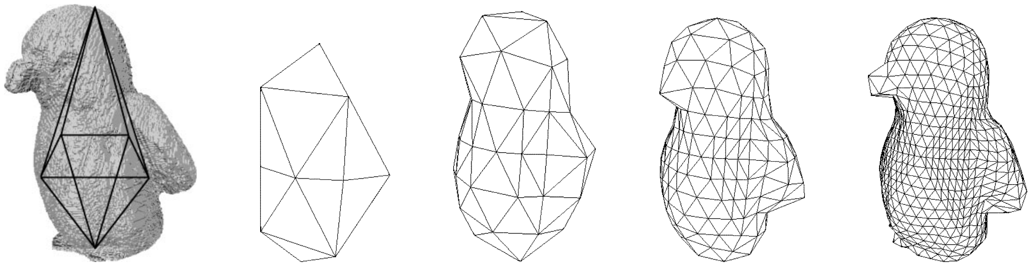


Fig. 8. Object geometry is initially approximated by fitting an octahedron to the voxel model. By triangle subdivision, refined *Penguin* geometry models consisting of 32, 128, 512, and 2048 triangles are generated.

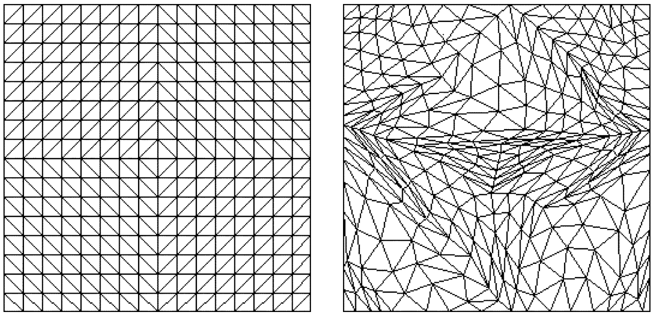


Fig. 9. Texture-map triangles are subdivided equivalent to their geometry mesh counterparts. Afterwards, the texture map is optimized to match the relative triangle area to the geometry triangle size.

positions [25]. The resulting optimized texture-map triangles have the same relative size as their corresponding geometry triangles. Fig. 9 depicts an optimized texture map. The coding gain achieved by optimizing texture maps depends on target bit rate and has been found to be more pronounced for finely textured objects.

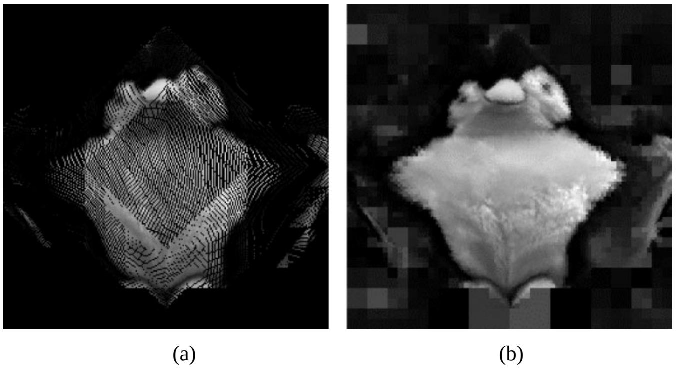
D. Image-to-Texture Map Conversion

A well-known problem in texture mapping is aliasing due to different resolutions in the image and the texture domain. To circumvent aliasing artifacts and to guarantee exact reconstruction, the texture domain is chosen significantly larger than the pixel area covered by the object in the images. Since many more texels are available than object pixels in the images, each texture map is only sparsely filled.

The object pixels in the image are inversely mapped onto the texture plane to determine the corresponding texels' color values. To convert multi-view images into view-dependent texture maps, the geometry model is rendered, and each pixel inside the projected model silhouette is assigned its corresponding geometry triangle and the relative coordinates within the triangle. Triangle number and coordinates determine the texel to which the color value of the image pixel is copied.

By mapping image pixels onto the texture plane, the texture maps are filled unevenly, as depicted in Fig. 10. Invisible triangles cause empty texture map regions, triangles seen at a grazing angle lead to sparsely filled areas, while for face-on triangles, different image pixels may be mapped onto the same texel. To avoid coinciding pixel mappings, the texture map is chosen suitably large.

By converting images into texture maps, substantially more texels are introduced than there are object pixels in the original images. To avoid adverse effects from large texture maps on compression efficiency, a wavelet coding scheme is applied to compress the texture information. Because wavelet coding relies on texture information represented in the frequency domain, bitstream size does not have to increase with map size in the spatial domain. In the following, a texture interpolation scheme is presented that allows filling in undefined texel values such that the wavelet coding bit rate remains minimal.



(a) (b)

Fig. 10. Mapping image pixels to the texture plane leads to unevenly distributed texture information. Missing texture information is interpolated from pixels within the texture map as well as from other maps. (a) Sparse texture map. (b) Interpolated texture map.

E. Sparse Texture Map Interpolation

The undefined texel values represent a large number of degrees of freedom that can be exploited to keep the bitstream size to a minimum by matching the texture interpolation to the wavelet coding scheme that follows. Because the four–dimensional (4-D) SPIHT wavelet coding method (Section V-F) performs best if high-frequency coefficients are small relative to low frequency coefficients, undefined texels must be interpolated subject to the constraint that the applied wavelet transform results in minimal high-frequency coefficient values and maximum low-frequency coefficients. For the 4-D Haar wavelet used, it can be shown that the mean value of the defined texels within the region of support must be assigned to the undefined texel positions in order to minimize the overall energy of the high-frequency coefficients [45]. Texture interpolation is performed in all four dimensions. 2×2 texels from 2×2 adjacent texture maps are considered, corresponding to the 4-D Haar basis function's region of support of 2^4 texels. Those texels within the 2^4-texel region which have been assigned a color value during image-to-texture mapping (Section V-D) are averaged, and the so-far undefined texels among the 16 texels are assigned this average value. To interpolate larger undefined texture regions, the texture-map array is downsampled by a factor of 2 along all four directions, assigning to each downsampled texel the previously determined 2^4-texel region average value. Undefined texels in the downsampled texture-map array are then again interpolated from local 2^4-texel regions by taking again the average of the defined texels. Downsampling and interpolation continues until all blank texture regions are filled. Fig. 10 depicts the sparse and the corresponding interpolated texture map for one frontal image of the *Penguin*. The blocky structures in the interpolated map result from the Haar wavelet's square region of support which for large undefined regions (such as the penguin's back in the example) shows up as blocks of uniform intensity, interpolated from texture maps that actually show the back side. Note, however, that these structures never become visible during rendering, since they are always on the invisible far side of the object. Instead, uniformly colored square areas which are aligned with the Haar basis functions keep wavelet coefficient coding bit rate at a minimum since they represent only low frequencies.

For image compression purposes, more efficient basis functions than the Haar wavelet are known [46]. Unfortunately, no optimal interpolation scheme could be derived for these more complex wavelet basis functions because of their overlapping region of support. The PTC codec therefore employs the simple Haar wavelet.

F. 4-D Wavelet Decomposition

Texture maps exhibit statistical properties very similar to conventional images. In addition, texels at the same coordinates in different texture maps display high correlation as they correspond to the same object surface point. The 4-D texture-map array allows exploiting intra-map as well as inter-map similarities by applying the one-dimensional Haar wavelet kernel separately along all four dimensions. After the entire texture-map array has been transformed, the resulting low-frequency coefficients, representing 1/16 of the original texture information, are again wavelet-transformed along all four dimensions. By repeated transformation of the low-frequency coefficients, a hierarchy of octave subbands is created. The resulting 4-D array of wavelet coefficients constitutes a joint multiresolution representation of all texture maps [45].

To compress the wavelet coefficient array, the *Set Partitioning in Hierarchical Trees (SPIHT)* codec [46] is modified to be applicable to the 4-D coefficient field [47]. In the modified SPIHT codec, wavelet coefficients are considered in order of importance, with large-magnitude coefficients encoded early on, and small coefficient values considered later in the bitstream. Reconstructed coefficients are at first coarsely represented and then gradually refined as bitstream decoding continues. Higher order statistical dependencies between wavelet coefficients in different subbands of the 4-D coefficient array are exploited to encode the positional arrangement of the magnitude-ordered coefficients. The extended 4D-SPIHT algorithm makes use of dependencies between subbands along all four dimensions. During decoding, all texture maps are jointly and progressively reconstructed. Bitstream decoding can be terminated at any arbitrary point to access a (lower quality) version of any texture map. Also, texture information is progressively reconstructed which allows continuous adjustment of reconstruction quality versus decoding speed.

G. Progressive Texture-Based Coding Performance

Progressive texture coding starts out by compressing the octahedron-based object geometry using the EMC algorithm (Section III-C). Object surface parameterization and texture-map optimization can be deduced from the encoded geometry model, so no additional mapping information needs to be encoded. For coder evaluation, 32×8 images of the *Garfield*, *Mouse*, *Penguin*, and *Sphere* data sets are transformed into 256×256-texel texture maps. For encoding, the texture maps are converted to $\mathrm{YC_bC_r}$ color space, and the chrominance components are downsampled by a factor of 2 in both dimensions. Each color channel is hierarchically decomposed into a 4-D pyramid of frequency subbands using the Haar wavelet (Section V-F). The coefficients are grouped into sets according to the 4-D SPIHT coding scheme, and the array is progressively encoded starting with the highest-magnitude coefficient. As many bits are written out to the bitstream as the preset bit rate allows. Coding ends after all three color channels have been encoded.

Fig. 11 depicts PTC performance for different object geometry approximations, denoted by the number of triangles. Since texture-based coding can encode only those image regions that fall within the geometry model's outline, reconstruction quality is measured as the average PSNR of the reconstructed object pixels only. Consequently, bit rate is measured with regard to the number of reconstructed object pixels only, expressed in *bits per object pixel* (bpop). In the examples, the object silhouette covers no more than $\approx 15\%$ of the total image size, i.e., approximately 1.5×10^4 pixels per silhouette. Note that this is considerably smaller than the texture map size of 2^{16} texels.

At 25.1-dB reconstruction PSNR, the *Garfield* images are encoded at 0.0066 bpop, whereas 0.174 bpop allow reconstructing the images at 31.3 dB mean object-pixel PSNR. For the *Mouse* images, bit rate ranges from 0.007 bpop at 24.3 dB up to 0.192 bpop at 29.9 dB. The *Penguin* images are encoded with 0.0076 bpop at 28.6 dB and 0.212 bpop at 36.6 dB. Coding bit rate for the *Sphere* data set ranges from 0.0087 bpop at 24.5 dB to 0.338 bpop at 38.4 dB.

H. Reconstruction Quality

Fig. 12 depicts decoded images of the *Garfield* data set. The applied 2048-triangle geometry approximation clips the original object along the projected model silhouette. At 31.3-dB reconstruction PSNR and 0.173 bits per object pixel, only very fine detail of the fur texture is lost. With increasing compression, blurriness increases and the Haar wavelet's nonoverlapping region of support causes a checkerboard effect of the textured surface. At 0.016 bpop and 26.9 dB PSNR, the reconstructed object texture is composed of irregularly shaped patches of uniform color.

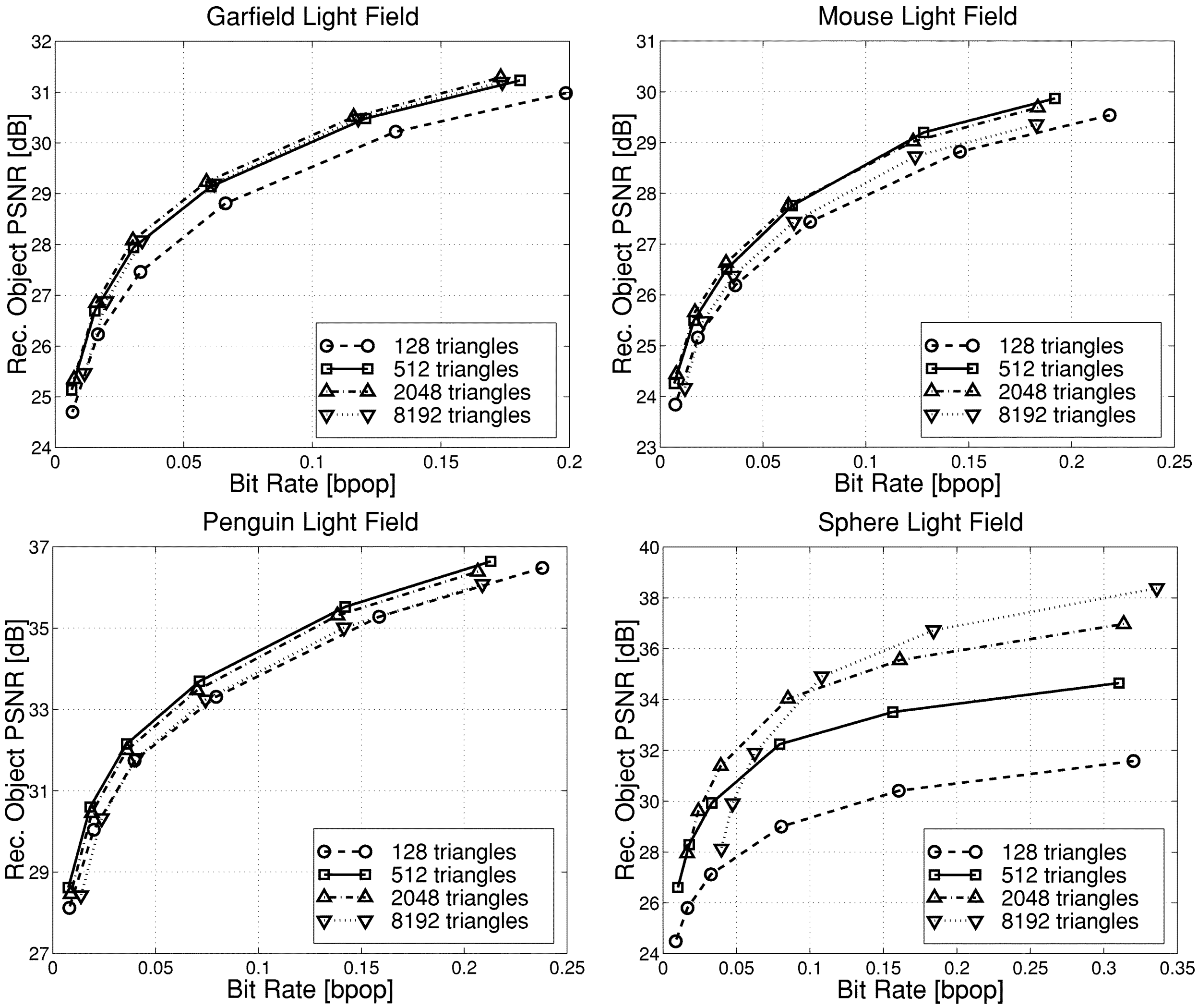


Fig. 11. Rate-distortion curves of progressive texture-based coding (PTC) for different geometry approximations: Reconstruction quality and bit rate (bpop) are expressed in relation to the number of reconstructed object pixels.

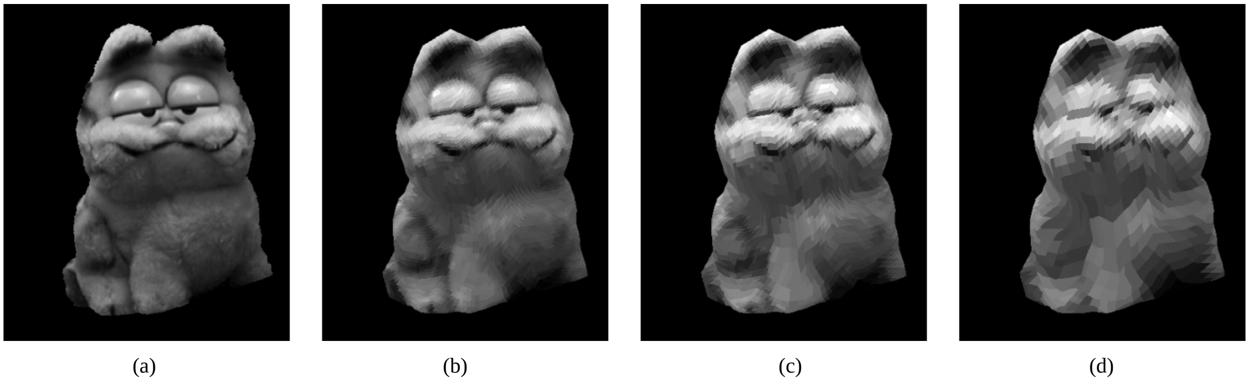


Fig. 12. Image from the *Garfield* image set, PTC encoded at different bit rates using the 2048-triangle geometry model. (a) Original image. (b) 0.173 bpop at 31.3 dB. (c) 0.059 bpop at 29.2 dB. (d) 0.016 bpop at 26.9 dB.

VI. Coder Comparison

To objectively compare model-aided and progressive texture-based coding performance, in the following coding bit rate as well as reconstruction quality are restricted to the pixels within the projected geometry silhouette. In Fig. 13, which depicts rate-distortion performance of the PTC and MAC coding schemes, reconstruction PSNR is measured only with respect to pixels that lie inside the model silhouette, and coding bit rate is expressed in *bits per reconstructed object pixel* (bpop), corresponding to Section V-G. The image sets are encoded using the same octahedral geometry models for MAC and PTC. The MAC rate-distortion curves in Fig. 13 therefore differ from Fig. 6.

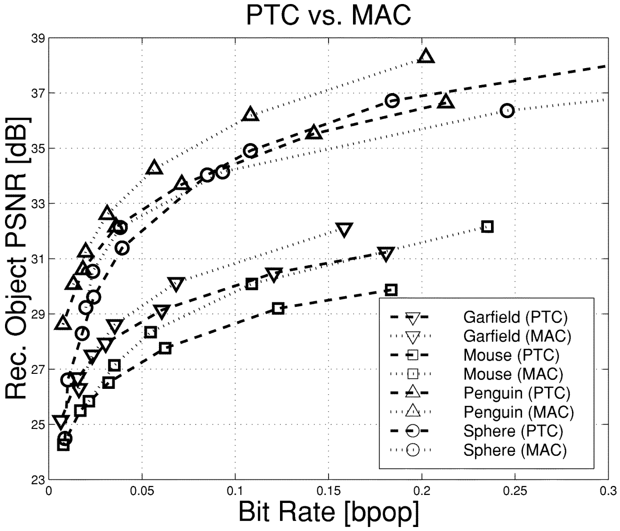


Fig. 13. Comparison of model-aided and progressive-texture coding performance.

The results depicted in Fig. 13 indicate that real-world objects whose geometry models exhibit only limited accuracy are more efficiently encoded using model-aided compression. At high reconstruction quality, the model-aided codec requires about 40% less bit rate to encode the *Garfield*, *Mouse* or *Penguin* data set than the progressive texture codec does. Only at the lowest bit rates does PTC perform equally well as MAC. For the synthetic *Sphere* image set, however, progressive-texture coding yields better coding results at medium to high reconstruction quality, and model-aided coding performs only slightly better at low bit rates.

To investigate the performance differences of PTC and MAC for real-world and synthetic images, additional experiments are conducted. For a broader experimental basis and to investigate the influence of object shape, the *Star* test data set is introduced as a second synthetic image set, Fig. 2. The same texture map as for the *Sphere* images is used to texture the *Star* geometry. As these experiments focus on image-data coding performance, the following coding results do not include geometry coding bit rate. Again, 256 × 256-texel maps are used for progressive-texture coding.

Fig. 14 illustrates that if exact geometry is used for PTC and MAC coding, the progressive texture-based coder performs significantly better than model-aided coding for both test data sets. Coding gains are especially impressive for the *Star* image set for which PTC achieves up to 80% better compression at medium to high reconstruction quality. At low bit rates, MAC performs only slightly better than PTC for the *Sphere* object.

To systematically examine the influence of small geometry inaccuracies on PTC and MAC coding performance, the exact *Sphere* and *Star* geometry models are gradually distorted. Due to the volumetric reconstruction algorithm's implicit shape-from-silhouette approach, the reconstructed geometry models tend to be slightly too small. Therefore, the vertices of the synthetic data sets are displaced randomly by a small amount along the radial direction toward the object's center. Vertex displacements parallel to the object's surface do not significantly alter overall model geometry and are not considered. Geometry distortion is expressed as mean radial vertex displacement in unit pixels when projected into the image plane. The radial displacement steps are chosen such that the depicted curves in Fig. 14 correspond to a mean silhouette mismatch of 1/16th, 1/8th, 1/4th, 1/2th, and 1 pixel.

Fig. 14 shows that PTC and MAC react differently to increasing geometry error. While at medium to high bit rates, PTC coding suffers a 1–2-dB decrease in reconstruction quality from one geometry approximation level to the next coarser level, the MAC encoder shows only marginal loss in reconstruction quality for small geometry deviations. Even at the greatest geometry distortion level, MAC performance degrades only by about 2 dB when compared to exact geometry. The experimental results indicate that the progressive-texture coding scheme is very sensitive even to small errors in object geometry.

To summarize, the texture-based coding scheme PTC is observed to yield superior compression results if exact 3-D scene geometry is available. For approximate geometry, however, coding efficiency drops quickly, and predictive MAC coding attains better compression performance. In the next section, this empirical result is investigated on a signal-theoretical basis.

VII. Theoretical Analysis

The effect of geometry inaccuracy on the efficiency of the two geometry-based coding schemes can be analyzed theoretically. A statistical signal processing framework for this analysis has been developed [48], similar to that for motion-compensation errors in video compression [49], [50]. The theoretical analysis qualitatively explains the experimental results from the previous section. The main ideas and results of the analysis are summarized here. The reader is referred to the Appendix for more details.

In this analysis, a simple planar geometry is considered, pictured in Fig. 15. With certain camera model assumptions, the multi-view image data set can be considered equivalent to a set of view-dependent texture maps, allowing for easy comparison of the texture-based and prediction-based schemes.

The set of texture maps is treated as a random process, with correlation both within a texture, and between textures of different views. The correlation between view-dependent texture maps depends upon the geometry error, which is modeled as a random quantity. The texture-based and prediction-based coding schemes are modeled as different transformations that seek to de-correlate the set of texture maps, allowing them to be encoded efficiently. The two schemes are compared by considering the optimal independent encoding efficiency of the resulting set of transformed images.

Fig. 16 shows the numerical results from the analysis for an arrangement similar to that in the previous sections. The coding efficiency, described by rate difference, is plotted against geometry accuracy. For rate difference, in bits per pixel, a more negative number indicates better coding efficiency. For geometry accuracy, described by the logarithm of the variance of the geometry error, a smaller number indicates a more accurate geometry.

When the geometry model is approximate and not very accurate, the prediction-based scheme out-performs the texture-based scheme. Here, coding performance is mainly affected by the amount of correlation between the texture maps, and the

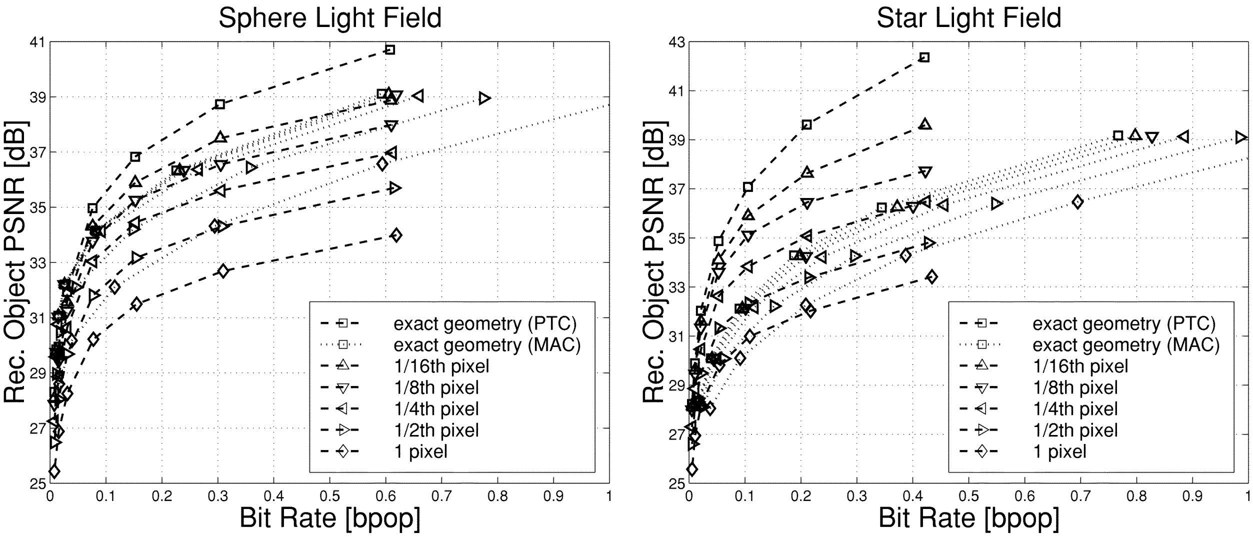


Fig. 14. Coding performance of MAC and PTC for two synthetis image data sets using different geometry approximations. Geometry accuracy is denoted as mean radial deviation of vertex positions, expressed in pixels.

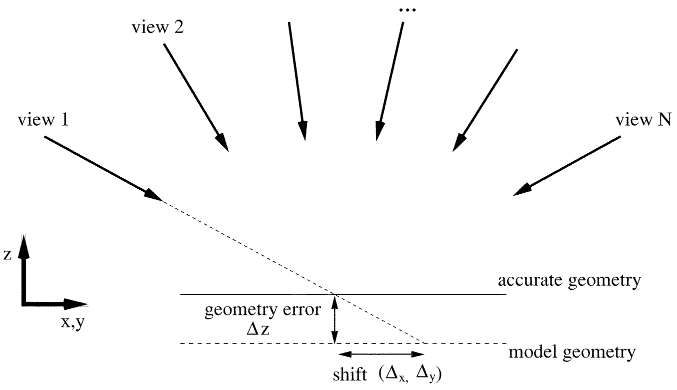


Fig. 15. Object geometry is modeled as a planar surface in the $x\ y$ plane, with vertical error Δz. Parallel projection is assumed. The projection of the texture from the accurate geometry surface onto the model geometry surface results in a texture shift by (Δ_x, Δ_y). This shift is dependent on the geometry error as well as the camera direction.

prediction-based scheme is better able to encode the texture maps. For high geometry accuracy, the texture maps are highly correlated, therefore, image noise dominates the analysis instead. The prediction-based scheme is not as efficient as the texture-based scheme in the presence of uncorrelated noise. These results qualitatively agree with the observations from the experimental results.

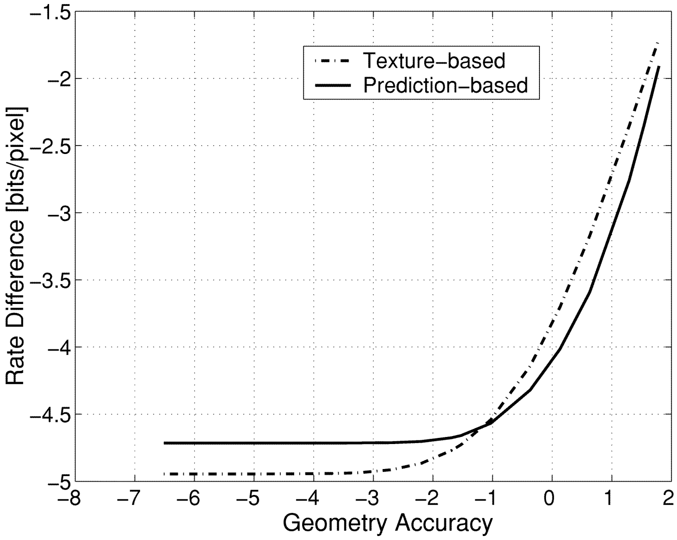


Fig. 16. Rate-difference (bits/pixel) versus geometry accuracy $(\log_2(\sqrt{12}\sigma)$ for the prediction-based scheme and the texture-based scheme. σ is the square root of the variance of the geometry error. For high geometry accuracy, toward the left of the plot, the curve for the texture-based scheme is below the curve of the prediction-based scheme, indicating greater efficiency. Conversely, for less accurate geometry, on the right side, the prediction-based scheme performs better. In this graph, the ratio of noise variance to signal variance is $\alpha = 0.001$.

VIII. Conclusion

In this paper, two different coding schemes for multi-view image data were presented that rely on reconstructed, actively acquired, or *a-priori* known 3-D geometry of the depicted scene. Both coding strategies apply 3-D scene geometry in different ways to either successively predict images and encode the prediction error, or to convert multi-view imagery into view-dependent texture maps prior to progressive wavelet coding. Compression ratios exceeding 2000:1 are observed for both coding schemes. In direct comparison, model-aided predictive coding is found to be more efficient for real-world images. For computer-generated image sets, however, progressive texture coding achieves significantly better compression than model-aided compression. Experiments with synthetic images show that texture-based coding is more susceptible to small errors in object geometry than the prediction-based scheme. A rate-distortion theoretical analysis is presented, explaining the relationship between geometry accuracy and coding efficiency for both schemes. The analysis confirms that for accurate geometry, the texture-based scheme is more efficient than the prediction-based approach, whereas for less accurate geometry, the prediction-based scheme is more efficient.

APPENDIX

A. Signal and Encoding Model

In this analysis [48], a simplified model for the geometry of the object is employed: a planar surface. A 2-D texture signal $v(x,y)$ exists on this surface, viewed by N cameras that have direction vectors $\mathbf{r_1}, \mathbf{r_2}, \ldots, \mathbf{r_N}$. This arrangement is illustrated in Fig. 15.

In addition, the surface is assumed to be Lambertian. All non-Lambertian view-dependent effects are modeled as noise. The camera that images the scene uses parallel projection and suffers no band-limitation restriction due to imaging resolution limits.

As a result of these assumptions, the image signal is essentially the same whether in the texture or image plane. Therefore, prediction between images can be performed in the texture plane instead of the image plane, making the comparison between the texture-based and prediction-based approaches simpler.

As illustrated in Fig. 15, the geometry error is modeled as an offset Δz of the planar surface from its true position. When the image is back-projected from view i onto the inaccurate geometry, this results in a texture c_i that is a shifted version of the original texture signal v given by the equation

$$c_i(x,y) = v(x - \Delta_{x_i}, y - \Delta_{y_i}) + n_i(x,y) \tag{1}$$

where n_i is the additive noise component that represents all non-Lambertian view-dependent effects.

The shift, which depends only upon the camera's viewing direction $\mathbf{r_i} = [r_{ix}\ r_{iy}\ r_{iz}]^T$ and the geometry error Δz, is described by the equation

$$\begin{bmatrix} \Delta_{x_i} \\ \Delta_{y_i} \end{bmatrix} = \Delta z \begin{bmatrix} \frac{r_{ix}}{r_{iz}} \\ \frac{r_{iy}}{r_{iz}} \end{bmatrix}. \tag{2}$$

The shift is a linear operation that can be represented in the frequency domain as the transfer function

$$D_i(\omega_x, \omega_y) = e^{-j(\omega_x \Delta_{x_i} + \omega_y \Delta_{y_i})}. \tag{3}$$

The set of texture images, given in vector form $\mathbf{c} = [c_1\ c_2\ \cdots\ c_N]^T$, represents the image data that is to be encoded. Both the prediction-based and the texture-based scheme take a linear transform of $\mathbf{c}$ to produce the set of images $\mathbf{e} = [e_1\ e_2 \cdots e_N]^T$. This linear transformation matrix T appears in the transformation equation $\mathbf{e} = T\mathbf{c}$. The model assumes that the components $e_1, e_2, \ldots, e_N$ are then independently encoded.

Using the transform T, the correlation between images can be reduced and therefore the entire image data set can be encoded more efficiently. The analysis centers on the effects of using different transformation matrices. For the texture-based scheme, the transform matrix describes a 2-D Haar wavelet subband decomposition. In the prediction-based scheme, the difference between the image to be predicted and a linear combination of its reference images is taken; the transform matrix describes this prediction structure.

The block diagram in Fig. 17 illustrates the relationship between v, $\mathbf{c}$, and $\mathbf{e}$.

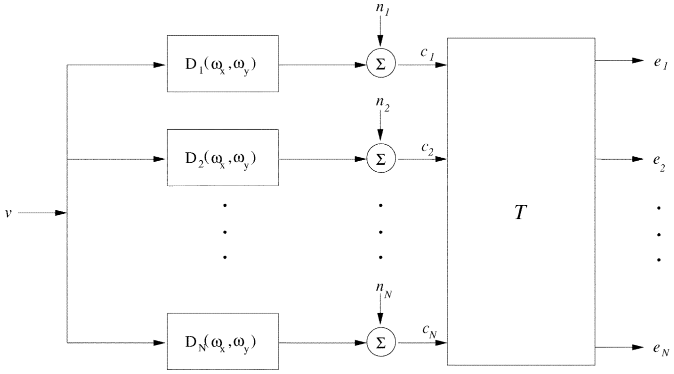


Fig. 17. Signals c_i are produced by shifting the original texture signal v by (Δ_{x_i}, Δ_{y_i}) (described by the transfer function $D_i(\omega_x, \omega_y)$) and adding signal-independent noise n_i. The signals c_i are linearly transformed (matrix T) to give the signals e_i. Each signal e_i is then independently encoded.

B. Statistical Model

The texture signal v is assumed to be a *wide-sense stationary* random process with *power spectral density (PSD)* $\Phi_{vv}(\omega_x, \omega_y)$. If the set of transfer functions that represent the shift in the texture map is denoted by the column vector

$$D = \begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_N \end{bmatrix} = \begin{bmatrix} e^{-j(\omega_x \Delta_{x1} + \omega_y \Delta_{y1})} \\ e^{-j(\omega_x \Delta_{x2} + \omega_y \Delta_{y2})} \\ \vdots \\ e^{-j(\omega_x \Delta_{xN} + \omega_y \Delta_{yN})} \end{bmatrix} \tag{4}$$

then the power spectrum of signal $\mathbf{c}$ is

$$\Phi_{cc} = D\, \Phi_{vv}\, D^H + \Phi_{nn} = \Phi_{vv}\, DD^H + \Phi_{nn} \tag{5}$$

where Φ_{nn} represents the power spectrum of the noise vector $\mathbf{n} = [n_1\ n_2\ \cdots\ n_N]^T$ and the superscript H denotes the complex-conjugate transpose of the matrix. Note that both Φ_{cc} and Φ_{nn} are matrices of size $N \times N$.

The shift amount of each texture map can be considered a stochastic quantity based on the random variable Δz, resulting in the following expression:

$$\Phi_{cc} = \Phi_{vv}\, E\{DD^H\} + \Phi_{nn} \tag{6}$$

where $E\{DD^H\}$ is defined by (7), shown at the top of the next page.

$P(\omega)$ represents the one–dimensional (1-D) Fourier transform of the continuous pdf of the random variable Δz, $\Omega = [\omega_x\ \omega_y]^T$ and $\mathbf{a_i} = [r_{ix}/r_{iz}\ r_{iy}/r_{iz}]^T$.

The power spectrum of the transformed signal $\mathbf{e}$ is

$$\Phi_{ee} = T\, \Phi_{cc}\, T^H = \Phi_{vv}\, T\, E\{DD^H\}\, T^H + T\, \Phi_{nn}\, T^H. \tag{8}$$

C. Performance Measure

To compare the texture-based and prediction-based approaches, the rate difference equation

$$\Delta R_i = \frac{1}{8\pi^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} \log_2 \left(\frac{\Phi_{eei}(\omega_x, \omega_y)}{\Phi_{cci}(\omega_x, \omega_y)} \right) d\omega_x\, d\omega_y \tag{9}$$

which represents the savings in bits per sample from encoding the signal e_i instead of the signal c_i, is used. This formula is

$$E\{DD^H\} = \begin{bmatrix} 1 & P(\Omega^T(\mathbf{a_1}-\mathbf{a_2})) & \dots & P(\Omega^T(\mathbf{a_1}-\mathbf{a_N})) \\ P(\Omega^T(\mathbf{a_2}-\mathbf{a_1})) & 1 & \dots & P(\Omega^T(\mathbf{a_2}-\mathbf{a_N})) \\ \vdots & \vdots & \ddots & \vdots \\ P(\Omega^T(\mathbf{a_N}-\mathbf{a_1})) & P(\Omega^T(\mathbf{a_N}-\mathbf{a_2})) & \cdots & 1 \end{bmatrix} \tag{7}$$

based on optimal encoding of a stationary, Gaussian signal at high rate [51].

The bit-rate savings is averaged over the entire set of images to obtain the overall rate difference

$$\Delta R = \frac{1}{N}\Sigma_{i=1}^{N} R_i. \tag{10}$$

D. Numerical Results

In evaluating the rate-difference equation for various geometry accuracies, 256 views arranged in a regular fashion over the hemisphere, similar to the arrangement in the experiments, are used. For the random geometry error Δz, a zero-mean Gaussian probability density function (pdf) with variance σ^2 is assumed.

All noise components are considered to be independent of one another. In addition, the noise signal spectrum is assumed to have the same shape as the image signal spectrum. The results do not change significantly even if a flat noise spectrum is used instead, indicating that the shape of the noise spectrum is not critical. The noise and image spectrums are related by the equation $\Phi_{nni}(\omega_x, \omega_y) = \alpha\ \Phi_{vv}(\omega_x, \omega_y)$, where α is the ratio of noise signal variance to image signal variance. The value $\alpha = 0.001$ is used in the numerical evaluation presented.

In Fig. 16, the rate difference for the prediction- and texture-based schemes in bits per pixel is plotted versus geometry accuracy. Geometry accuracy is expressed as $\log_2(\sqrt{12}\sigma)$, a quantity used in motion-compensation analysis for video compression, where σ is the square root of the variance of the geometry error Δz. There are two main regions of the curve that are interesting: the high geometry accuracy regime to the left and the low geometry accuracy regime to the right.

In the low geometry accuracy region, the misalignment between textures due to the texture-map shifts dictates the performance of the coding schemes. The prediction-based scheme predicts from views that have similar viewing direction, therefore the relative misalignment is smaller than in the texture-based scheme which encodes all views simultaneously. A similar geometric argument is used in [52] to explain the experimental results. A bit-rate difference of approximately 1/2 bits/pixel is observed between the prediction- and texture-based approaches. This corresponds to approximately a 3-dB difference in image quality.

For very high geometry accuracy, there is very little misalignment due to geometry error, so, instead, noise dominates the analysis. The texture-based coding scheme employs an orthogonal transform that preserves the noise variance. In the case of the prediction-based scheme, subtracting two or more independent noise signals increases the noise variance, and therefore degrades performance. This is why the texture-based scheme outperforms the prediction-based scheme by 1/4 bits/pixel, corresponding to 1.5 dB in image quality.

It should be noted that the analysis makes several assumptions about the formation of the light field images, and further assumes that these signals are Gaussian and are encoded at high rate. In light of these assumptions, comparisons between the theoretical and empirical results can only be qualitative, and apply only to the higher bit rates of the experimental results. However, for synthetic as well as real-world data sets with approximate geometry, the experimental and theoretical results agree that the prediction-based scheme is more efficient. For the synthetic multi-view data sets, with high geometry accuracy, both theory and experiments suggest that the texture-based scheme is more efficient.

References

[1] S. E. Chen, “Quicktime VR – An image-based approach to virtual environment navigation,” in *Proc. ACM Conf. Computer Graphics (SIGGRAPH’95)*, Los Angeles, CA, Aug. 1995, pp. 29–38.

[2] L. McMillan and G. Bishop, “Plenoptic modeling: An image-based rendering system,” in *Proc. ACM Conf. Computer Graphics (SIGGRAPH’95)*, Los Angeles, CA, Aug. 1995, pp. 39–46.

[3] M. Levoy and P. Hanrahan, “Light field rendering,” in *Proc. ACM Conf. Computer Graphics (SIGGRAPH’96)*, New Orleans, LA, Aug. 1996, pp. 31–42.

[4] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen, “The lumigraph,” in *Proc. ACM Conf. Computer Graphics (SIGGRAPH’96)*, New Orleans, LA, Aug. 1996, pp. 43–54.

[5] P. Debevec, C. Taylor, and J. Malik, “Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach,” in *Proc. ACM Conf. Computer Graphics (SIGGRAPH’96)*, New Orleans, LA, Aug. 1996, pp. 11–20.

[6] R. Szeliski and H.-Y. Shum, “Creating full view panoramic mosaics and environment maps,” in *Proc. ACM Conf. Computer Graphics (SIGGRAPH’97)*, Los Angeles, CA, Aug. 1997, pp. 251–258.

[7] H.-Y. Shum and L.-W. He, “Rendering with concentric mosaics,” in *Proc. ACM Conf. Computer Graphics (SIGGRAPH’99)*, Los Angeles, CA, Aug. 1999, pp. 299–306.

[8] D. Wood, D. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. Salesin, and W. Stuetzle, “Surface light fields for 3D photography,” in *Proc. ACM Conf. Computer Graphics (SIGGRAPH-2000)*, New Orleans, LA, July 2000, pp. 287–296.

[9] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, “Unstructured lumigraph rendering,” in *Proc. ACM Conf. Computer Graphics (SIGGRAPH’01)*, Los Angeles, CA, Aug. 2001, pp. 425–432.

[10] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum, “Plenoptic sampling,” in *Proc. ACM Conf. Computer Graphics (SIGGRAPH’00)*, New Orleans, LA, Aug. 2000, pp. 307–318.

[11] G. Miller, S. Rubin, and D. Ponceleon, “Lazy decompression of surface light fields for precomputed global illumination,” in *Proc. Eurographics Rendering Workshop’98*, Vienna, Austria, Oct. 1998, pp. 281–292.

[12] I. Ihm, S. Park, and R. K. Lee, “Rendering of spherical light fields,” in *Proc. 5th Pacific Conf. Computer Graphics and Applications*, Seoul, Korea, Oct. 1997, pp. 59–68.

[13] P. Lalonde and A. Fournier, “Interactive rendering of wavelet projected light fields,” in *Proc. Graphics Interface’99*, Kingston, ON, Canada, June 1999, pp. 107–114.

[14] Y. Wu, L. Luo, J. Li, and Y.-Q. Zhang, “Rendering of 3D wavelet compressed concentric mosaic scenery with progressive inverse wavelet synthesis (PIWS),” in *Proc. SPIE Visual Communications and Image Processing (VCIP-2000)*, vol. 1, Perth, Australia, June 2000, pp. 31–42.

[15] J. Li, H. Y. Shum, and Y. Q. Zhang, "On the compression of image based rendering scene," in *Proc. IEEE Int. Conf. Image Processing (ICIP-00)*, vol. 2, Sept. 2000, pp. 21–24.

[16] I. Peter and W. Strasser, "The wavelet stream: Interactive multi resolution light field rendering," in *Proc. 12th Eurographics Workshop Rendering*, London, U.K., June 2001, pp. 262–273.

[17] H. Aydinoglu and M. Hayes, "Compression of multi-view images," in *Proc. IEEE Int. Conf. Image Processing (ICIP'94)*, vol. 2, Nov. 1994, pp. 385–389.

[18] C. Zhang and J. Li, "Compression and rendering of concentric mosaics with reference block codec (RBC)," in *Proc. SPIE Visual Communications and Image Processing (VCIP-2000)*, vol. 1, Perth, Australia, June 2000, pp. 43–54.

[19] X. Tong and R. M. Gray, "Interactive view synthesis from compressed light fields," in *Proc. IEEE International Conference on Image Processing (ICIP-2001)*, Oct. 2001, pp. 85–88.

[20] M. Magnor and B. Girod, "Data compression for light field rendering," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 338–343, Apr. 2000.

[21] H. Schirmacher, W. Heidrich, and H.-P. Seidel, "High-quality interactive lumigraph rendering through warping," in *Proc. Graphics Interface 2000*, Montreal, QC, Canada, May 2000, pp. 87–94.

[22] B. Girod and M. Magnor, "Two approaches to incorporate approximate geometry into multi-view image coding," in *Proc. IEEE Int. Conf. Image Processing (ICIP-2000)*, vol. 2, Sept. 2000, pp. 5–8.

[23] P. Debevec, Y. Yu, and G. Boshokov, "Efficient view-dependent image-based rendering with projective texture-mapping," Univ. California, Berkeley, CA, Rep. CSD-98–1003, vol. 20, 1998.

[24] D. Cohen-Or, Y. Mann, and S. Fleishman, "Deep compression for streaming texture intensive animations," in *Proc. ACM Conf. Computer Graphics (SIGGRAPH'99)*, Los Angeles, CA, Aug. 1999, pp. 261–268.

[25] M. Magnor and B. Girod, "Model-based coding of multi-viewpoint imagery," in *Proc. SPIE Visual Communication and Image Processing (VCIP-2000)*, vol. 1, Perth, Australia, June 2000, pp. 14–22.

[26] S. M. Seitz and C. M. Dyer, "View morphing," in *Proc. ACM Conf. Computer Graphics (SIGGRAPH'96)*, New Orleans, LA, Aug. 1996, pp. 21–30.

[27] M. Magnor, P. Eisert, and B. Girod, "Model-aided coding of multi-viewpoint image data," in *Proc. IEEE Int. Conf. Image Processing (ICIP-2000)*, vol. 2, Sept. 2000, pp. 919–922.

[28] P. Eisert, "Model-based camera calibration using analysis by synthesis techniques," in *Proc. Vision, Modeling, and Visualization (VMV'02)*, Erlangen, Germany, Nov. 2002, pp. 307–314.

[29] S. Seitz and C. Dyer, "Photorealistic scene reconstruction by voxel coloring," in *Proc. IEEE Computer Vision and Pattern Recognition (CVPR'97)*, June 1997, pp. 1067–1073.

[30] K. Kutulakos, "Shape from the light field boundary," in *Proc. IEEE Computer Vision and Pattern Recognition (CVPR'97)*, June 1997, pp. 53–59.

[31] K. Kutulakos and S. Seitz, "A theory of shape by space carving," in *Proc. IEEE Int. Conf. Computer Vision (ICCV'99)*, vol. 1, Sept. 1999, pp. 307–314.

[32] P. Eisert, E. Steinbach, and B. Girod, "Automatic reconstruction of stationary 3-D objects from multiple uncalibrated camera views," *IEEE Trans. Circuits Syst. Video Technol. (Special Issue on 3D Video Technology)*, vol. 10, pp. 261–277, Mar. 2000.

[33] O. Faugeras, *Three-Dimensional Computer Vision*. Cambridge, MA: MIT Press, 1993.

[34] R. Koch and L. Van Gool, Eds., *3D Structure from Multiple Images of Large-Scale Environments*. Berlin, Germany: Springer-Verlag, 1998, vol. 1506, Lecture Notes in Computer Science.

[35] M. Pollefeys, R. Koch, M. Vergauwen, and L. van Gool, *Metric 3D Surface Reconstruction from Uncalibrated Image Sequences*. Berlin, Germany: Springer-Verlag, 1998, vol. 1506, Lecture Notes in Computer Science, pp. 139–154.

[36] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *Proc. ACM Conf. Computer Graphics (SIGGRAPH'87)*, Anaheim, CA, July 1987, pp. 163–169.

[37] H. Hoppe, "Progressive meshes," in *Proc. ACM Conf. Computer Graphics (SIGGRAPH'96)*, New Orleans, CA, Aug. 1996, pp. 99–108.

[38] J. Popović and H. Hoppe, "Progressive simplicial complexes," in *Proc. ACM Conf. Computer Graphics (SIGGRAPH'97)*, Los Angeles, CA, Aug. 1997, pp. 217–224.

[39] G. Taubin, A. Gueziec, W. Horn, and F. Lazarus, "Progressive forest split compression," in *Proc. ACM Conf. Computer Graphics (SIGGRAPH'98)*, Orlando, FL, July 1998, pp. 123–132.

[40] R. Pajarola and J. Rossignac, "Compressed progressive meshes," Georgia Inst. Technol., Atlanta, GA, Rep. GIT-GVU-99–05, 1999.

[41] M. Magnor and B. Girod, "Fully embedded coding of triangle meshes," in *Proc. Vision, Modeling, and Visualization (VMV'99)*, Erlangen, Germany, Nov. 1999, pp. 253–259.

[42] S. Chen and L. Williams, "View interpolation for image synthesis," in *Proc. ACM Conf. Computer Graphics (SIGGRAPH'93)*, Anaheim, CA, Aug. 1993, pp. 279–288.

[43] M. Magnor and B. Girod, "Hierarchical coding of light fields with disparity maps," in *Proc. IEEE Int. Conf. Image Processing (ICIP'99)*, vol. 3, Oct. 1999, pp. 334–338.

[44] W. Heidrich, H. Schirmacher, H. Kück, and H.-P. Seidel, "A warping-based refinement of lumigraphs," in *Proc. 6th Int. Conf. Central Europe Computer Graphics and Visualization*, Plzen, Czech Republic, Feb. 1998, pp. 102–109.

[45] M. Magnor, *Geometry-adaptive Multi-View Coding Techniques for Image-based Rendering*. Aachen, Germany: Shaker-Verlag, 2001.

[46] A. Said and W. Pearlman, "A new, fast and efficient image codec based on set partitioning in hierarchical trees," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp. 243–250, June 1996.

[47] M. Magnor, A. Endmann, and B. Girod, "Progressive compression and rendering of light fields," in *Proc. Vision, Modeling, and Visualization (VMV-000)*, Saarbrücken, Germany, Nov. 2000, pp. 199–203.

[48] P. Ramanathan and B. Girod, "Theoretical analysis of geometry inaccuracy for light field compression," in *Proc. IEEE Int. Conf. Image Processing ICIP-02*, Sept. 2002, pp. 229–232.

[49] B. Girod, "The efficiency of motion-compensating prediction for hybrid coding of video sequences," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 1140–1154, Aug. 1987.

[50] ——, "Efficiency analysis of multihypothesis motion-compensated prediction for video coding," *IEEE Trans. Image Processing*, vol. 9, pp. 173–183, Feb. 2000.

[51] N. S. Jayant and P. Noll, *Digital Coding of Waveforms*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[52] M. Magnor and B. Girod, "Sensitivity of image-based and texture-based multi-view coding to model accuracy," in *Proc. IEEE Int. Conf. Image Processing (ICIP-2001)*, vol. 3, Oct. 2001, pp. 98–101.

Marcus Magnor (S'99–M'02) received the B.A. degree from the University of Würzburg, Würzburg, Germany, in 1995, the M.S. degree in physics from the University of New Mexico, Albuquerque, in 1997, and the Ph.D. degree in electrical engineering from the University of Erlangen, Erlangen, Germany, in 2000.

Having spent 2001 as Research Associate at Stanford University's Computer Graphics Laboratory, he is currently heading the Independent Research Group Grahics-Optics-Vision at the Max-Planck-Institut für Informatik, Saarbrücken, Germany. His research interests include image- and video-based rendering, immersive television, 3-D imaging and acquisition, as well as multimedia communications.

Prashant Ramanathan (S'01) received the B.A.Sc. degree in systems design engineering in 1997 from the University of Waterloo, Waterloo, ON, Canada, and the M.S. degree in electrical engineering in 1999 from Stanford University, Stanford, CA, where he is currently working toward the Ph.D. degree.

He is currently working on compression and rate-distortion optimized streaming for image-based rendering. His other research interests include image and video compression, multimedia streaming, computer graphics, and computer vision.

Bernd Girod (F'98) received the M. S. degree in electrical engineering from Georgia Institute of Technology, Atlanta, in 1980, and the Doctoral degree "with highest honors" from the University of Hannover, Hannover, Germany, in 1987.

He is Professor of Electrical Engineering in the Information Systems Laboratory of Stanford University, Stanford, California. He also holds a courtesy appointment with the Stanford Department of Computer Science and he serves as Director of the Image Systems Engineering Program at Stanford. His research interests include networked media systems, video signal compression and coding, and 3-D image analysis and synthesis. Until 1987, he was a member of the research staff at the Institut für Theoretische Nachrichtentechnik und Informationsverarbeitung, University of Hannover, working on moving image coding, human visual perception, and information theory. In 1988, he joined Massachusetts Institute of Technology, Cambridge, first as a Visiting Scientist with the Research Laboratory of Electronics, then as an Assistant Professor of Media Technology at the Media Laboratory. From 1990 to 1993, he was Professor of Computer Graphics and Technical Director of the Academy of Media Arts in Cologne, Germany, jointly appointed with the Computer Science Section of Cologne University. He was a Visiting Adjunct Professor with the Digital Signal Processing Group at Georgia Institute of Technology in 1993. From 1993 until 1999, he was Chaired Professor of Electrical Engineering/Telecommunications at the University of Erlangen-Nuremberg, Germany, and the Head of the Telecommunications Institute I, co-directing the Telecommunications Laboratory. He served as the Chairman of the Electrical Engineering Department from 1995 to 1997, and as Director of the Center of Excellence "3-D Image Analysis and Synthesis" from 1995 to 1999. He was a Visiting Professor with the Information Systems Laboratory of Stanford University during the 1997/1998 academic year. As an entrepreneur, he has worked successfully with several startup ventures as founder, investor, director, or advisor. Most notably, he was a co-founder and Chief Scientist of Vivo Software, Inc., Waltham, MA (1993-1998); after Vivo's aquisition, from 1998 to 2002, he was Chief Scientist of RealNetworks, Inc.; and, since 1996, an he has been an outside Director of 8 × 8, Inc. He has authored or coauthored one major textbook, two monographs, and over 250 book chapters, journal articles, and conference papers in his field, and he is the holder of about 20 international patents.

Prof. Girod has served as on the Editorial Boards or as Associate Editor for several journals in his field, and is currently Area Editor for Speech, Image, Video and Signal Processing of the IEEE TRANSACTIONS ON COMMUNICATIONS. He has served on numerous conference committees, e.g., as Tutorial Chair of ICASSP-97 in Munich, Germany, and ICIP-2000 in Vancouver, BC, Canada, as General Chair of the 1998 IEEE Image and Multidimensional Signal Processing Workshop in Alpbach, Austria, and as General Chair of the Visual Communication and Image Processing Conference (VCIP) in San Jose, CA, in 2001. He was a member of the IEEE Image and Multidimensional Signal Processing Committee from 1989 to 1997 and was named a Distinguished Lecturer for the year 2002 by the IEEE Signal Processing Society. Together with J. Eggers, he was a recipient of the 2002 EURASIP Best Paper Award.