Proc. International Conference on Image Processing ICIP-2002, Rochester, New York, Sept. 2002. © 2002 IEEE

GEOMETRY REFINEMENT FOR LIGHT FIELD COMPRESSION

Prashant Ramanathan, Eckehard Steinbach, Peter Eisert and Bernd Girod

Information Systems Laboratory
Stanford University
{pramanat,steinb,eisert,bgirod}@Stanford.EDU

ABSTRACT

In geometry-aided light field compression, a geometry model is used for disparity-compensated prediction of light field images from already encoded light field images. This geometry model, however, may have limited accuracy. We present an algorithm that refines a geometry model to improve the over-all light field compression efficiency. This algorithm uses an optical-flow technique to explicitly minimize the disparity-compensated prediction error. Results from experiments performed on both real and synthetic data sets show bit-rate reductions of approximately 10% using the improved geometry model over a silhouette-reconstructed geometry model.

1. INTRODUCTION

Image-based rendering has emerged as an important new alternative to traditional image synthesis techniques in computer graphics. With image-based rendering, scenes can be rendered by sampling previously acquired image data, instead of synthesizing them from light and surface shading models and scene geometry. Light field rendering [1, 2] is one such image-based technique that is particularly useful for interactive applications.

A light field is a 4-D data set which can be parameterized as a 2-D array of 2-D light field images. For photo-realistic quality, a large number of high-resolution light field images is required, resulting in extremely large data sets. For example, the light field of Michelangelo's statue of *Night* contains tens of thousands of images and requires over 90 Gigabytes of storage for raw data [3]. Compression is therefore essential for light fields.

Currently, the most efficient techniques for light field compression use *disparity compensation*, analogous to motion compensation in video compression. In disparity compensation, images are predicted from previously encoded reference images. Disparity or depth values are either specified for a block of pixels, or inferred from a geometry model [4, 5, 6].

In this paper, we consider disparity-compensated light field compression using an explicit geometry model. A geometry model can be an efficient method of specifying the depth values required for disparity-compensated prediction. The geometry models used may be of limited accuracy for several reasons. The model may be generated from image data using error-prone computer vision techniques. Even an accurate model must be represented digitally in a finite number of bits, and therefore, some degree of approximation is necessary. The result of this geometry inaccuracy is reduced compression efficiency for the compression algorithm [7].

In this paper, we describe a method of refining the geometry model to reduce the disparity-compensated prediction error, and improve compression efficiency. Our algorithm is similar to the Sliding Textures approach [8, 9], differing in only a few details. One of the main contributions of this paper is to apply these ideas to the problem of light field compression. In Section 2, we review the basics of geometry-based disparity-compensated light field compression. In Section 3, we present our method for geometry refinement. We present our results in Section 4.

2. GEOMETRY-BASED DISPARITY-COMPENSATED LIGHT FIELD COMPRESSION

Disparity compensation is used in most current light field compression algorithms [4, 5, 6]. The underlying idea of disparity-compensated prediction is that a pixel in a light field image can be predicted from corresponding pixels in one or more other light field images. This prediction requires a depth value for a given pixel. In a light field, the recording geometry is known, which means that by specifying the depth, it is possible to establish correspondence between pixels in two different views. This pixel correspondence allows for the prediction of pixels of one view from another. We assume that the imaged surface in both views look similar, which is true for Lambertian, unoccluded surfaces.

In a geometry-based prediction scheme, depth values are inferred from an explicit geometry model by rendering the model. The reference images that are used to predict a particular image must be defined. We follow the hierarchical coding structure described by Magnor and Girod [4]. Here, each image is predicted from two to four reference images. The order in which images are encoded is also defined, so that images are predicted from images that are already encoded.

3. PROPOSED ALGORITHM FOR GEOMETRY REFINEMENT

In this section we present an algorithm that modifies the geometry model for better compression performance. We explain the optical-flow-based shape refinement method, and the iterative regularized least-squares method used to find the solution.

3.1. Optical-flow-based equation

A light field image may be predicted from one or more reference light field images that have been previously encoded, according to the hierarchical structure discussed in the previous section. We call the image and pixel to be predicted the *target* image and pixel, and the images and pixels from where they are predicted the *reference* images and pixels. The intensity of a given target pixel is predicted from the corresponding pixel in a reference image. Any given target pixel corresponds to a specific 3-D point along the viewing ray $\mathbf{l}$ in 3-D space. If we allow this line $\mathbf{l}$ to be parameterized by the depth t, we obtain the line equation, in world coordinates,

$$\mathbf{l}(t) = \begin{bmatrix} l_x(t) \\ l_y(t) \\ l_z(t) \end{bmatrix}. \qquad (1)$$

Note that this line is a function of the intrinsic and extrinsic camera parameters of the target view, as well as the pixel position in the image. By projecting this line into the reference view, we obtain a 2-D line $\mathbf{e}$. This so-called *epipolar line*

$$\mathbf{e}(t) = \begin{bmatrix} e_X(t) \\ e_Y(t) \end{bmatrix}, \qquad (2)$$

also parameterized by the original depth parameter t, is a function of the pixel position in the target image, and the camera parameters of the target and reference images.

Specifying a depth value t fixes the point in 3-D space as well as in the reference view on the epipolar line. Thus, we obtain a corresponding reference pixel for the target pixel. We assume that the true value of t will result in the same intensity for the corresponding target and reference pixels. An error in the depth t will result in a prediction error, denoted by the difference in intensity ΔI. This is a function of the intensity value at the target pixel I_t and at the reference pixel $I_r(t)$ given by

$$\Delta I = I_t - I_r(t_0) = I_r(t) - I_r(t_0) \qquad (3)$$

where t_0 is the current (inaccurate) depth value, and t is the correct depth value that results in no prediction error.

If we assume the intensity gradient

$$\mathbf{g} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} \qquad (4)$$

to be locally constant over this region in the reference image, we obtain the familiar optical flow equation

$$\Delta I = \mathbf{g}^T(\mathbf{e}(t) - \mathbf{e}(t_0)) \qquad (5)$$

However, this equation only relates prediction error ΔI to the depth parameter t for a given pixel. We need to further relate this to the parameters of the geometry model using the relation

$$t = h(\mathbf{p}) \qquad (6)$$

where $\mathbf{p}$ is the vector of geometry parameters and h is a nonlinear multivariate function that maps the geometry parameters to the depth for a given pixel.

We now describe the mapping function h for our problem. For the triangle mesh geometry model that we use, the geometry parameters are the positions of each of the vertices in the model. In addition, we restrict the movement of these vertices to one degree of freedom, radially from the center of the model.

For a particular target pixel, the corresponding 3-D line $\mathbf{l}$ intersects the geometry at exactly one triangle face. Therefore, the depth parameter t is determined by the three vertices that define this triangle face. By characterizing this triangle as an infinite plane defined by its three vertices, we obtain a differentiable function h that describes the depth parameter t in terms of the geometry parameters $\mathbf{p}$. Note that we assume that other vertices will not affect this pixel, through occlusion for example, and that the pixel will not move off this triangle. Both of these assumptions are supported by the restriction that changes in the geometry parameters will be small, enforced by regularization in the least-squares solution.

Combining (5) and (6), we obtain

$$\Delta I = \mathbf{g}^T(\mathbf{e}(h(\mathbf{p})) - \mathbf{e}(h(\mathbf{p}_0))) \qquad (7)$$

where $\mathbf{p}_0$ denotes the current geometry configuration. Both $\mathbf{e}$ and h are non-linear functions of the parameter vector $\mathbf{p}$. We can linearize this equation, and the resulting equation will be valid locally around $\mathbf{p}_0$. If

$$\mathbf{e}(h(\mathbf{p})) \approx C_1\mathbf{p} + C_2, \qquad (8)$$

then

$$\mathbf{e}(h(\mathbf{p})) - \mathbf{e}(h(\mathbf{p}_0)) \approx C_1\Delta\mathbf{p} \qquad (9)$$

where $\Delta\mathbf{p} = \mathbf{p} - \mathbf{p}_0$ and C_1 is a matrix of size $2 \times N$, with N as the number of geometry parameters.

Substituting (9) into (7), we obtain the following equation

$$\mathbf{g}^T(C_1\Delta\mathbf{p}) \approx \Delta I \qquad (10)$$

for each pixel and a corresponding reference view.

3.2. Least-Squares Solution

This equation may be derived for all the pixels in the light field that are to be predicted, and can be combined to form the matrix equation

$$A\Delta\mathbf{p} \approx \mathbf{b} \qquad (11)$$

where

$$A = \begin{bmatrix} (\mathbf{g}^T C_1)^{(1)} \\ (\mathbf{g}^T C_1)^{(2)} \\ \cdots \\ (\mathbf{g}^T C_1)^{(M)} \end{bmatrix} \qquad (12)$$

and

$$\mathbf{b} = \begin{bmatrix} (\Delta I)^{(1)} \\ (\Delta I)^{(2)} \\ \cdots \\ (\Delta I)^{(M)} \end{bmatrix} \qquad (13)$$

Because our linearized problem and our mapping function h are only valid for small $\Delta\mathbf{p}$, we must include regularization into the solution of our problem. This gives us the equation

$$\begin{bmatrix} A \\ \lambda I \end{bmatrix} \Delta\mathbf{p} \approx \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} \qquad (14)$$

where λ is the regularization constant. A larger value for λ means that the solution $\Delta\mathbf{p}$ will be smaller, and the problem will be more numerically stable. When λ is too large, however, it takes many iterations to converge to the solution. In our experiments, the value for λ is selected empirically.

This linearized problem can be solved using the least-squares approach. Since an equation is formed for each pixel predicted from a reference view, the number of rows of A can be large. In any particular equation, only three parameters are specified, therefore A is also sparse. We use the *LSQR* method [10], which is well-suited to large, sparse problems, in our implementation.

Once we obtain a new geometry model from the solution, we can again linearize the equations about the new operating point, and solve for the new change in geometry parameters. We can iteratively perform these two steps until we converge to the best geometry model.

4. RESULTS

Our experiments use both real and synthetic light field data sets. An initial approximate geometry model is created using the silhouette information from the light field image data. This silhouette-reconstructed geometry model is refined using the technique described in this paper to obtain the improved geometry model. For the synthetic light fields, we also have the true geometry models, which can serve as a useful reference point. We encode the light fields using each of these geometry models, and compare their relative rate-PSNR performance. The light field coder is described next.

4.1. Light Field Coder

The light field coder in our work uses block-based disparity-compensation both without and with an explicit geometry model [4, 5]. All images are divided up into 8×8 blocks. Each block is encoded in one of several modes: the INTRA mode, where DCT-based image compression is used for the block; the GEO mode, where an explicit geometry model is used to predict the block from reference images; the STD (standard) mode where a depth value is specified to predict the block from reference images; and the COPY mode, where a block from the same image location is simply copied from the reference image. For the STD mode, the depth values are quantized such that they correspond to approximately integer-pixel accuracy in the image plane. In the GEO and STD modes, a DCT-based residual encoder is used on the prediction error. Mode selection is based on a rate-distortion Lagrangian cost function

$$J = D + \lambda_M R \qquad (15)$$

where D is the sum-squared-error distortion of the block image, and R is the rate in bits for the block. The mode with the smallest Lagrangian cost is chosen. A rate-PSNR curve is obtained by varying the image quality, using the quantization parameter Q in the DCT intra and residual coders. The Lagrangian multiplier λ_M that is used to trade off rate versus distortion is adjusted according to the quantization parameter using the following equation commonly used in video compression, [11]

$$\lambda_M = 0.85 Q^2. \qquad (16)$$

4.2. Experiments

Four data sets were used in our experiments. The first two, *Star* and *Cube*, are synthetic light fields, each with 26 images of resolution 256×256. The last two, *Garfield29* and *Garfield288*, are light fields recorded from a real-world object, the same plush toy. *Garfield29* has 29 images, each of resolution 384×288, covering the frontal region of the object, while *Garfield288* has 288 images, each of resolution 192×144 covering the entire hemi-sphere of views.

For each of the data sets, we derive a geometry model that is consistent with the silhouette from each view. We begin with a 642-vertex subdivided icosahedron model that is larger than the object. In each view, for each vertex, if a vertex lies outside of the silhouette, we move it radially towards a center point so that it lies on the silhouette border. We thereby obtain a 642-vertex object that matches the silhouette in all views.

For each of the four light fields, we refine this silhouette-reconstructed geometry object to obtain an improved geometry object. Typically, we use anywhere from 250 to 1000 iterations and a regularization constant $\lambda = 10^{10}$. Both of these quantities are determined empirically based on the results.

Figure 1 shows the results of our algorithm for the *Star* light field. Figure 1(a) shows the true geometry, Figure 1(b) shows the silhouette-reconstructed geometry, and Figure 1(c) shows the refined geometry. Figure 2 illustrates the geometry results for the real-world *Garfield29* light field. Figure 2(a) shows only face of the object from one image of the light field. Figure 2(b) shows the silhouette-reconstructed geometry, and Figure 2(c) shows the refined geometry.

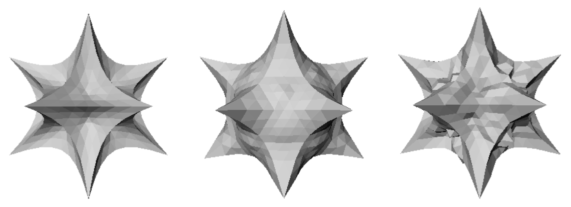
(a) True Geo. (b) Silhouette Geo. (c) Improved Geo.

Fig. 1. Geometry models for the *Star* light field. The *near-exact constrained geometry* (not pictured) is visually identical to the true geometry.

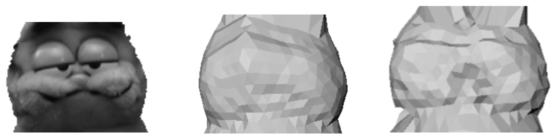
(a) Light field image (b) Silhouette Geo. (c) Improved Geo.

Fig. 2. Magnified portion of light field image and geometry models for the *Garfield29* light field.

For all of the light field data sets, we compare the efficiency of our light field coder using the silhouette-reconstructed geometry versus the improved geometry. For the two synthetic light fields, we can compare with the results of the true geometry model as well. Our algorithm constrains the set of possible improved geometry outcomes, since it uses only 642 vertices and constrains the positions of these vertices to be in the same directions as the original subdivided icosahedron vertices. To understand the possible effect of these constraints, we create another geometry model that is subject to these constraints, but fit as close as possible to the exact geometry model. We call this our *near-exact constrained geometry*. This geometry model represents the best possible geometry result under the constraints that we have placed on the algorithm.

Figures 3 and 4 show the Rate-PSNR curves using the various geometry models for the *Star* light field and the *Garfield29* light field, respectively. The bit-rate for the geometry models is not included. Since we have a regular icosahedron arrangement of vertices, where only the 642 vertex radii must be specified, this bit-rate will be negligible compared to the overall bit-rate for the light field. The PSNR is measured over the entire image.

Due to space considerations, we do not show the curves for the other data sets. In all cases, we see a bit-rate reduction of approximately 10% using the improved geometry instead of the silhouette-reconstructed geometry. This corresponds to an increase of approximately 1 dB in PSNR. The results for the synthetic data sets indicate that there still exists a large performance gap between the improved geometry and the exact geometry. The results for the near-exact constrained geometry show, however, that only another 10% is possible using our constrained arrangement. In other words, our improved geometry realizes 50% of the gain possible under our constrained arrangement.

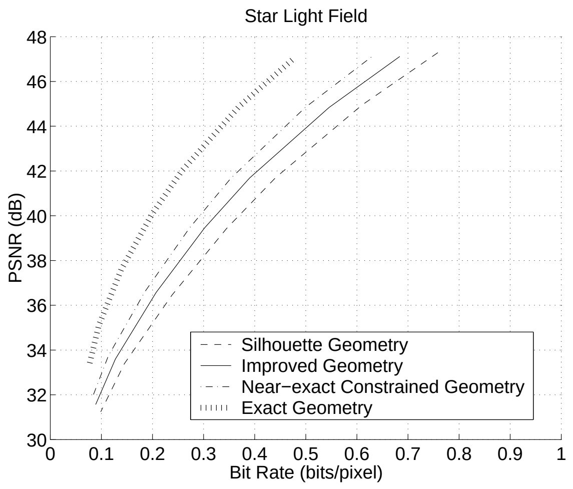

Fig. 3. Rate-PSNR for *Star* Light Field. We see a 10% bit-rate reduction using the improved geometry over the original silhouette-reconstructed geometry. The near-exact constrained geometry shows us the best possible result for our constrained arrangement. There is still a large performance gap from the exact geometry results.

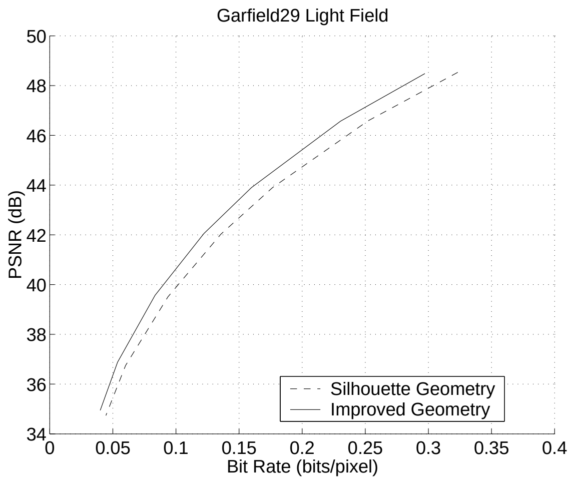

Fig. 4. Rate-PSNR for *Garfield29* Light Field. We see a 10% bit-rate reduction using the improved geometry over the original silhouette-reconstructed geometry for this real-world light field.

5. CONCLUSIONS

We have presented an algorithm to automatically refine the geometry model used for disparity-compensated light field compression. This improved geometry model reduces the disparity-compensation prediction error and improves the compression efficiency. Our experiments show bit-rate savings of approximately 10% using the refined geometry model over the silhouette-reconstructed geometry model. These experiments were performed on both real and synthetic light field data sets.

Results from the synthetic data sets indicate that the algorithm may be improved significantly by relaxing some of the geometric constraints in the algorithm.

6. REFERENCES

[1] Marc Levoy and Pat Hanrahan, "Light field rendering," in *Computer Graphics (Proceedings SIGGRAPH96)*, August 1996, pp. 31–42.

[2] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen, "The lumigraph," in *Computer Graphics (Proceedings SIGGRAPH96)*, August 1996, pp. 43–54.

[3] Marc Levoy, Kari Pulli, et al., "The Digital Michelangelo project: 3D scanning of large statues," in *Computer Graphics (Proceedings SIGGRAPH00)*, August 2000, pp. 131–144.

[4] Marcus Magnor and Bernd Girod, "Data compression for light field rendering," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 3, pp. 338–343, April 2000.

[5] Marcus Magnor, Peter Eisert, and Bernd Girod, "Model-aided coding of multi-viewpoint image data," in *Proceedings of the IEEE International Conference on Image Processing ICIP-2000*, Vancouver, Canada, September 2000, vol. 2, pp. 919–922.

[6] Xin Tong and Robert M. Gray, "Coding of multi-view images for immersive viewing," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing ICASSP 2000*, Istanbul, Turkey, June 2000, vol. 4, pp. 1879–1882.

[7] Marcus Magnor, *Geometry-Adaptive Multi-View Coding Techniques for Image-based Rendering*, Ph.D. thesis, University Erlangen-Nuremberg, Germany, 2001.

[8] Peter Eisert, Eckehard Steinbach, and Bernd Girod, "Automatic reconstruction of stationary 3-D objects from multiple uncalibrated camera views," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 2, pp. 261–277, March 2000.

[9] Eckehard Steinbach, Peter Eisert, and Bernd Girod, "Model-based 3-D shape and motion estimation using sliding textures," in *Proceedings Vision, Modelling and Visualization 2001*, Stuttgart, Germany, November 2001.

[10] Christopher C. Paige and Michael A. Saunders, "LSQR: An algorithm for sparse linear equations and sparse least squares," *ACM Transactions on Mathematical Software*, vol. 8, no. 1, pp. 43–71, March 1982.

[11] Gary J. Sullivan and Thomas Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, vol. 15, pp. 74–90, November 1998.