COMPRESSION OF LIGHTFIELD RENDERED IMAGES USING COSET CODES

A. Jagmohan, A. Sehgal, N. Ahuja

University of Illinois, Urbana-Champaign
{jagmohan, asehgal, n-ahuja}@uiuc.edu

ABSTRACT

Image-based rendering (IBR) and lightfield rendering (LFR) techniques aim to represent a 3D real-world environment by densely sampling it through a set of fixed viewpoint cameras. Remote digital walkthroughs of the 3D environment are facilitated by synthesizing novel viewpoints from the captured view-set. The large amount of data generated by the dense capture process makes the use of compression imperative for practical IBR/LFR systems. In the present paper, we consider the design of compression techniques for streaming of IBR data to remote viewers. The key constraints that a compression algorithm for IBR streaming is required to satisfy, are those of random access for interactivity, and precompression. We propose a compression algorithm based on the use of coset codes for this purpose. The proposed algorithm employs H.264 source compression in conjunction with LDPC coset codes to precompress the IBR data. Appropriate coset information is transmitted to the remote viewers to allow interactive view generation. Results indicate that the proposed compression algorithm provides good compression efficiency, while allowing client interactivity and server precompression.

1. INTRODUCTION

Digital walkthroughs of remote 3D real-world environments find applications in fields such as remote surveillance, video conferencing and virtual malls/museums. The remote user typically controls a virtual viewpoint which he/she changes in the 3D environment creating a trajectory along which the scene can be continuously televiewed as a series of 2D views. A key challenge is to represent the 3D environment using as few parameters as possible, so as to minimize the amount of information that needs to be transmitted to allow the user to teleview the scene from any arbitrary viewpoint.

A traditional approach to this problem has been to parameterize the geometrical and reflectance characteristics of the constituent objects of the 3D scene and use these parameters in conjunction with appropriate illumination models to represent the 3D scene. Then, these parameters can be appropriately transformed to synthesize the 2D view of the scene from an arbitrary viewpoint. Examples of such approaches can be found in [1, 2, 3]. However, finding a set of geometrical and reflectance parameters that can characterize an arbitrary 3D scene photo-realistically has proven to be a difficult problem to solve.

An alternative class of approaches for this problem, which has emerged during the last decade, is that of image-based rendering (IBR) or lightfield rendering (LFR) [4, 5]. In these approaches, the 3D scene is imaged using a fixed set of camera positions. As the user-viewpoint changes dynamically, the required views are synthesized directly as transformations of the captured image set, denoted $\{C_i\}_{i\in\{1,\ldots,n\}}$. The two key steps in an IBR system, as shown in Fig. 1, are (1) *Representation* of the real-world environment by densely sampling the plenoptic function through the captured image set $\{C_i\}$, and (2) *Virtual-view generation* i.e. synthesis of a novel view, based on the desired user-viewpoint, by appropriately transforming the captured images.

In the present paper, we consider the design of compression techniques for streaming of IBR data to remote viewers. Fig. 2 shows the scenario under consideration. The IBR representation, in the form of the captured views $\{C_i\}_{i\in\{1,\ldots,n\}}$, is stored at a central server, and the viewers are located at remote clients separated from the server through a network. As we shall see, a compression algorithm for IBR streaming is required to satisfy the conflicting constraints of allowing random access for interactive viewing, while allowing precompression of the captured data. These constraints can be satisfied by independently coding the captured images, for instance, but independent coding is wasteful in terms of coding efficiency.

In this work, we present the design of a compression algorithm for streaming IBR data, which allows random access and precompression without sacrificing compression efficiency. We show that the problem of compression for IBR streaming belongs to the class of problems which can be formulated as variants of the Wyner-Ziv decoder side-information problem in information theory [6]. The proposed compression framework employs H.264 source compression in conjunction with LDPC coset codes to precompress the IBR data. Appropriate coset information is transmitted to the clients to allow interactive view generation. To illustrate the efficacy of the proposed approach, we present

0-7803-8104-1/03/$17.00 ©2003 IEEE

compression performance results for streaming of the standard Dragon lightfield, and the standard Tsukuba multiview stereo set. The proposed precompression algorithm is shown to significantly outperform independent coding in terms of rate-distortion (R-D) performance, and is shown to be within 1-2.5 dB of the compression performance achievable with on-the-fly compression.

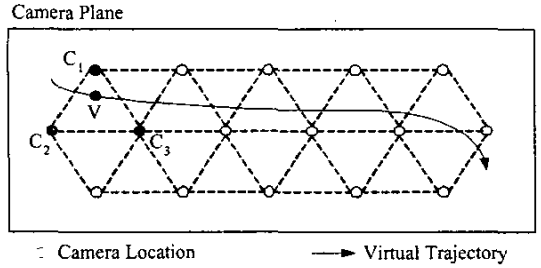

Fig. 1. Simplified IBR system under consideration. The cameras are setup as a 2D planar array, with optical axis perpendicular to the camera plane. The camera locations triangulate the view-plane. Synthesis of virtual view V, along the desired user trajectory, is done using the triangle vertices C_1, C_2, C_3.

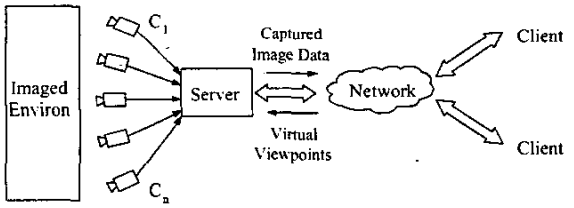

Fig. 2. IBR streaming scenario under consideration. The IBR representation is stored at the server. The viewers are located at remote clients, and virtual-view generation is performed at the clients. The clients communicate the desired user trajectories to the server, which responds with appropriate captured views.

2. PROBLEM DEFINITION

We consider the problem of compression for the simplified IBR system shown in 1. The cameras are setup in a 2-D coplanar array, with the optical axes of the cameras aligned perpendicular to the camera plane. The set of camera views is denoted as $\{C(x_i, y_i)\}_{i\in\{1,\ldots,n\}}$, where $C(x_i, y_i)$ denotes the view captured by the camera with optical centre located at the location (x_i, y_i) on the camera plane. In the sequel we simplify the notation by denoting $C(x_i, y_i)$ as C_i. As shown in Fig. 1, the set of camera views $\{C_i\}$ defines a triangulation of the camera plane. We denote the set of indices of the three camera views constituting the vertices of the triangle enclosing a point on the camera plane with coordinates (x_i, y_i), as $\Delta(x_i, y_i)$.

A given virtual trajectory traversed by a viewer, denoted $\mathcal{T}$, requires the generation of a sequence of virtual views $[V_i] \equiv [V(x_i, y_i)]$ for virtual camera centers (x_i, y_i) on the camera plane. As shown in Fig. 1, we assume for the sake of simplicity that a virtual view V_j is synthesized by transforming the camera views which constitute the vertices of the triangle in which the virtual camera center lies i.e. $V_j = f_s(C_{i_1}, C_{i_2}, C_{i_3})$ where $f_s(\cdot,\cdot,\cdot)$ is an appropriate synthesis function, and $i_1, i_2, i_3 \in \Delta(x_j, y_j)$. This simplified IBR system constrains the allowed virtual viewpoints, and the quality of the synthesized virtual views. However, the proposed compression algorithm can be easily extended to more complicated representations and synthesis functions.

We consider the problem of compression for the streaming scenario shown in Fig. 2. The representation $\{C_i\}$ is stored at the central server. Multiple viewers are located at remote clients $\mathcal{U}_1, \ldots, \mathcal{U}_m$. A given client $\mathcal{U}_k$ dynamically informs the server of the virtual trajectory the viewer wishes to traverse, $\mathcal{T}_k = [(x_{k,t}, y_{k,t})]$. The server, in response, transmits a sequence of camera views with indices $S = [i_t]$, with $i_t \in \Delta(x_{k,t}, y_{k,t})$, such that the viewer can synthesize the desired trajectory. In the sequel, we denote the t^{th} element of the sequence S by $S(t)$. Note that the process of virtual-view generation occurs at the client.

The desired virtual trajectory for a given client, thus defines the sequence S in which the camera views are to be transmitted to the client. Compression for IBR streaming implies the compression of the transmitted view sequence S, prior to transmission to the client. The two chief constraints required to be satisfied by the compression algorithm are that it should provide interactivity and precompression. The requirement of interactivity stems from the real-time nature of the virtual walk-through process—the client can dynamically change the desired virtual trajectory. This constraint implies that the view sequence to be transmitted can only be known during transmission. The requirement of precompression implies that the camera view-set $\mathcal{V}$ is required to be compressed completely, *prior to* transmission. This is because, in a typical remote IBR application, there are expected to be a large number of clients interacting with the server at any given time. This makes it computationally infeasible for the server to perform real-time, on-the-fly compression of the transmitted view sequence for each individual client.

We now evaluate the suitability of conventional IBR compression approaches for the problem of IBR streaming. The simplest coding methodology is to independently compress each camera view during precompression. This allows interactivity since, during transmission, the server can trans-

mit each compressed camera view as and when it is required by the remote viewer. However, independent coding does not remove the large amount of inter-view redundancy which results from the dense sampling of the real-world environment, and hence provides poor compression efficiency. At the other extreme are approaches that seek to completely eliminate the IBR inter-view redundancy, through the use of hierarchical coding of the view-set [5, 8], or through the use of multi-dimensional transform coding [9]. These approaches provide good compression performance, but they do so at the expense of interactivity. This is because these approaches introduce decoding dependencies between multiple views in the IBR view-set, random access of individual view-data, required for interactive client trajectories, is severely constrained.

An alternative is to use predictive coding to compress the transmitted view sequence S. Conventional predictive coding encodes each source symbol with respect to previously encoded symbols. Thus, for the case of IBR streaming, conventional one-step predictive coding would require the transmission of the innovation sequence $\tau_t = C_{S(t)} - E[C_{S(t)}|\widehat{C}_{S(t-1)}]$ at transmission instant t, where $\widehat{C}_k$ denotes the client reconstruction of the camera view C_k, and $E[\cdot]$ denotes the expectation operator. The client decoder would then reconstruct the required camera view as $\widehat{C}_{S(t)} = \widehat{\tau}_t + E[C_{S(t)}|\widehat{C}_{S(t-1)}]$, where $\widehat{\tau}_t$ denotes the received innovation sequence at the decoder. The key problem encountered in using conventional predictive coding for IBR streaming is that the trajectory uptil transmission instant $(t-1)$ should be known prior to the process of compression, since the generation of the innovation symbol τ_t requires knowledge of the decoder predictor $\widehat{C}_{S(t-1)}$. This conflicts with the requirement of precompression, which dictates that the server compress the view-set information before the trajectory is known. To see the extent of this problem, we note that the decoder reconstruction at time instant t is a function of the decoder predictor $\widehat{C}_{S(t-1)}$, because $\tau_t \neq \widehat{\tau}_t$, in general, because of lossy source coding. Thus the cardinality of the set of potential predictors for a given camera view, during precompression, is of the same order as the cardinality of the set of possible view-set sequences culminating in the transmission of the camera view. This makes precompression using predictive coding infeasible.

The goal of the present paper is to present the design of a compression algorithm for IBR streaming that provides good compression efficiency while satisfying the requirements of precompression and interactivity.

3. PROPOSED COMPRESSION METHODOLOGY

In this section we show how the problem of compression for IBR streaming can be formulated as a decoder side-information problem. We briefly describe the architecture of the proposed encoder for precompression of the view-set data at the server.

3.1. Decoder side-information formulation

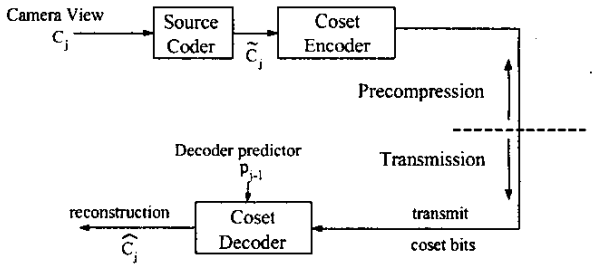

Fig. 3. Precompression and transmission of IBR camera view C_j using coset codes. For trajectory-independent reconstruction $\widehat{C}_j = \tilde{C}_j$.

Consider the communication of a M-dimensional source with memory, $\{X_i\}_{i=1}^{\infty}, X_i \in \mathbb{R}^M$, using one-step predictive coding. Given the decoder reconstruction of source symbol X_{k-1} (denoted $\widehat{X}_{k-1}$), the encoder communicates X_k by transmitting the innovation $T_k = X_k - E[X_k|\widehat{X}_{k-1}]$. Now consider the case where the decoder reconstruction $\widehat{X}_{k-1}$ belongs to a *reconstruction set* $\mathcal{R}_k$, and consider that the encoder has knowledge of $\mathcal{R}_k$, but not of $\widehat{X}_{k-1}$, while encoding source symbol X_k. We denote this scenario as predictive coding with one of a multiplicity of predictors. As we have shown previously, this scenario is of interest in several practical video coding applications including multiple description coding [10] and joint source-channel coding [11]. This scenario can be formulated as a variant of the Wyner-Ziv decoder side-information paradigm in information theory, with the predictor $\widehat{X}_{k-1}$ serving as the side-information which is available only to the decoder. Accordingly, code constructions based on coset codes, analogous to the classical information-theoretic constructions for the Wyner-Ziv problem, can be used for predictive coding in this scenario [6].

The problem of compression for IBR streaming can be formulated as one of predictive coding with one of a multiplicity of predictors as follows. The source word to be encoded is the camera view C_j which is to be compressed during precompression. The reconstruction set consists of all possible decoder predictors, $\mathcal{R}_j = \{\widehat{C}_{p,k} : p \in \mathcal{N}(j), \mathcal{T}_k \in \Gamma\}$, where $\mathcal{N}(j)$ denotes the set of camera views that are connected to C_j through triangulation edges, and Γ denotes the set of possible trajectories which include the view C_j. Note that $|\mathcal{R}_j| = |\mathcal{N}(j)| \cdot |\mathcal{T}_k|$, illustrating the unsuitability of conventional predictive coding for the problem at hand. During precompression the server has knowledge of $\mathcal{R}_j$,

but not of the precise decoder predictor, which will only be available during transmission.

Fig. 3 shows how a coset code construction can be used for the problem of compression for IBR streaming. During precompression, each camera view C_j is source-coded to generate $\tilde{C}_j$, and coset indices, defined on an appropriate channel code, are generated for the view. Note that this process does not introduce dependencies between multiple camera views and thus allows random access of camera views for interactivity. During transmission, appropriate coset indices for view C_j are transmitted and can be decoded at the client with the help of the side-information present at the client. It is important to note that the coset decoder can ensure the reconstruction of C_j is identical to the source coder output, i.e. $C_j^e = \tilde{C}_j = \widehat{C}_j$, thereby ensuring *trajectory-independent reconstruction*, where the server knows precisely the decoder reconstruction of each camera view C_j. The utility of such a reconstruction will be discussed in the next section.

3.2. Proposed Encoder Architecture

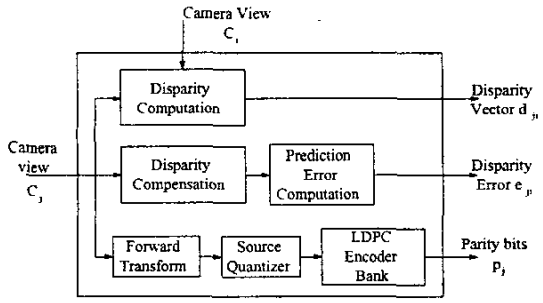

Fig. 4. A simplified representation of the proposed encoder architecture used for precompressing an IBR camera view C_j. Disparity compensation vectors and error vectors with respect to each neighboring view are computed and stored during precompression.

Fig. 4 shows a simplified block diagram of the proposed encoder. During precompression at the server, for each camera view C_j, the encoder generates and stores: (1) disparity compensation information with respect to each C_i, $i \in \mathcal{N}(j)$, and (2) multiple sets of coset indices of C_j, each generated using one of multiple coset codes.

Disparity information for the view C_j is generated by using a quad-tree block-based disparity compensation algorithm, akin to the motion compensation algorithm specified in the H.264 video coding standard. Disparity compensation vectors are generated with respect to each neighboring view of C_j, and are denoted $d_{ji:i\in\mathcal{N}(j)}$. In addition to the disparity compensation vectors, the encoder also generates disparity error vectors, $e_{ji:i\in\mathcal{N}(j)}$, by computing the residual between C_j and the disparity compensated neighboring view. The disparity error computation uses the reconstruction $C^e_{i:i\in\mathcal{N}(j)}$, which is precisely known even during precompression, because of the trajectory-independent reconstruction property. The disparity error vectors are transform coded, quantized and entropy coded, prior to storage, for compression. The transmission of disparity error vectors offsets the coding efficiency loss due to the non-availability of VLC codes for compression of coset information [10].

Coset information for view C_j is generated as described in [10]. The view is transform coded using the H.264 forward transform. The transform coefficients are quantized using the H.264 scalar quantizer, and the quantization indices are then input to a bank of systematic GF(2) LDPC encoders. The parity bits p_j of the systematic LDPC code represent the coset indices for C_j. In practice, multiple sets of parity bits are generated using LDPC codes with different rates.

During transmission, at transmission instant t, the server is required to communicate the camera view $C_{S(t)}$. The server accomplishes this by transmitting the disparity compensation vector $d_{S(t)S(t-1)}$, the disparity error vector $e_{S(t)S(t-1)}$ and an appropriate set of parity bits $p_{S(t)}$ to allow the decoder to correctly reconstruct the desired view. The amount of coset information required for successful decoding, i.e. the parity bit-set required to be transmitted, is precisely known if trajectory-independent reconstructions are used.

4. RESULTS

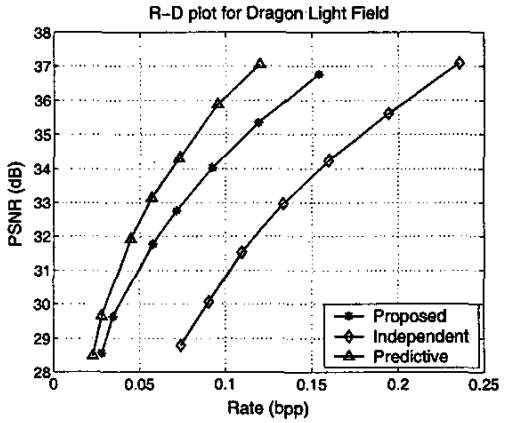

Fig. 5. Compression performance for Dragon lightfield. The proposed approach is within 1-2 dB of the omniscient predictive coding bound.

Fig. 5 shows compression performance results for one

slab of the standard Dragon lightfield set. The representation of the set consists of a 32×32 planar array of cameras arranged such that the focal plane of all the cameras is parallel to the camera-plane. The resolution of each camera view is 256×256. Several random trajectories with sequence length $|S| = 50$ were used for traversal of the camera-plane, and the R-D performance over the trajectories was appropriately averaged to generate the results. The performance of the proposed encoder was compared to: (1) independent coding of the camera views using the H.264 independent coding mode, and (2) omniscient predictive coding, in which the traversed trajectory was assumed to be known a-priori, and the H.264 coding standard was used to compress the trajectories—this serves as an upper bound on the achievable compression performance.

As Fig. 5 shows, the proposed approach outperforms independent coding by $3 - 4$ dB. More significantly, the performance of the proposed approach is only $1 - 2$ dB worse than that of the omniscient predictive coding bound. This demonstrates that the proposed approach provides interactivity and allows precompression at the expense of only a small performance loss in terms of compression performance.

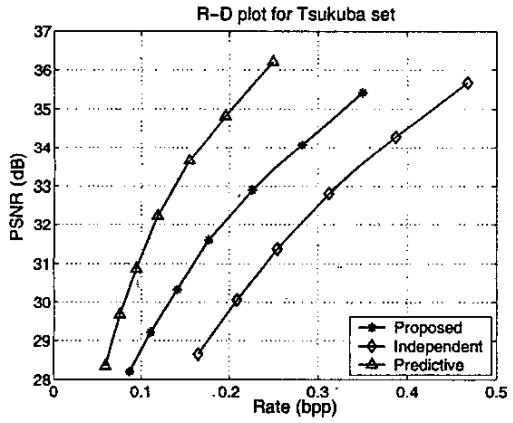

Fig. 6. Compression performance for Tsukuba multi-view set.

Fig. 6 shows similar results for the Tsukuba multi-view dataset. The Tsukuba dataset consists of a coplanar array of twenty-five camera viewpoints with resolution 384×288, and images a scene with a significantly larger field-of-view of interest than in the Dragon lightfield. As can be seen from the figure, the performance of the proposed approach is $2 - 2{:}5$ dB better than that of independent coding, and $2 - 3$ dB worse than that of the predictive coding bound for the Tsukuba view-set.

5. CONCLUSIONS

We have presented a compression algorithm for IBR streaming, which employs the transmission of coset information to provide good compression efficiency, while allowing client interactivity and server precompression. The proposed algorithm has been shown to yield promising results for standard IBR data-sets. The proposed encoder is a specific manifestation of our recently proposed state-free video coding paradigm [7], thereby allowing easy incorporation of error-resilience and scalable compression. The performance of the proposed codec can be further enhanced by utilizing appropriate depth-estimation techniques at the decoder which eliminate the need for explicitly transmitting disparity information. Simple R-D optimized algorithms can be developed to determine the source coding rate and the transmission rate for coset information. Finally, further research in improving the time-complexity of coset-code decoding would be fruitful.

6. REFERENCES

[1] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International Journal of Computer Vision*, vol. 47, pp. 7–42, 2002.

[2] R. Koch, "3-d surface reconstruction from stereoscopic image sequences," in *International Conference of Computer Vision*, 1995, pp. 109–114.

[3] K.N. Kutulakos and S.M. Seitz, "A theory of shape by space carving," *International Journal of Computer Vision*, vol. 38, pp. 199–218, 2000.

[4] M. Levoy and P. Hanrahan, "Light field rendering," in *Proc. SIGGRAPH 96*, 1996, pp. 31–42.

[5] D. Aliaga, T. Funkhouser, D. Yanovsky, and I. Carlbom, "Sea of images," *Proc. IEEE Visualization*, pp. 331–338, 2001.

[6] A. Jagmohan, A. Sehgal, and N. Ahuja, "Predictive encoding using coset codes," in *IEEE International Conference on Image Processing*, 2002, vol. 2, pp. 29–32.

[7] A. Sehgal, A. Jagmohan, and N. Ahuja, "A state-free causal video encoding paradigm," in *IEEE International Conference on Image Processing*, 2003, vol. 1, pp. 605–608.

[8] M. Magnor and B. Girod, "Data compression for light-field rendering," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, pp. 338–343, Apr. 2000.

[9] M. Magnor, A. Endmann, and B. Girod, "Progressive compression and rendering of light fields," *Proc. Vision, Modeling, and Visualization*, pp. 199–203, 2000.

[10] A. Jagmohan, A. Sehgal, and N. Ahuja, "Wyze-pmd based multiple description video codec," in *Proc. IEEE Intl. Conf. Multimedia Expo*, 2003, vol. 1, pp. 569–572.

[11] A. Sehgal and N. Ahuja, "Robust predictive coding and the wyner-ziv problem," in *IEEE Data Compression Conference*, 2003, pp. 103–112.