RECEIVER-BASED RATE-DISTORTION OPTIMIZED INTERACTIVE STREAMING FOR SCALABLE BITSTREAMS OF LIGHT FIELDS

Chuo-Ling Chang and Bernd Girod

Information Systems Laboratory, Department of Electrical Engineering, Stanford University
{chuoling,bgirod}@Stanford.EDU

ABSTRACT

A receiver-based rate-distortion optimized framework to interactively stream scalable bitstreams of light fields is proposed. Based on the desired viewpoints, the predicted future buffer state, the network characteristics, and the target transmission rate, the receiver customizes the data request in order to minimize the distortion in the frames to be rendered. Experimental results show that the proposed framework improves the rendering quality by $1.0 \sim 2.6$ dB over a heuristic scheme at the same rate. Correspondingly, to achieve the same rendering quality the proposed framework saves $30\% \sim 40\%$ of the rate over the heuristic scheme.

1. INTRODUCTION

A *light field* [1][2] captures the outgoing radiance from a particular scene, at all points in 3-D space and in all directions. In practice, a light field can be represented as a set of 2-D camera views. A novel view of the scene from an arbitrary viewpoint can be rendered by appropriately combining pixels from the acquired camera views.

The uncompressed size for a large light field can exceed tens of gigabytes. Even with efficient compression, downloading the entire data set can be prohibitive. It is also redundant if the user only navigates part of the scene. An attractive alternative is to interactively stream the necessary data stored at the sender to the user at the receiver instead of requiring the user to download the entirety before viewing it.

In [3], Ramanathan et al. extend the work of Chou and Miao for rate-distortion optimized packet scheduling of video and audio data [4] to streaming of light fields. In their framework, the compressed data set has been assembled into packets, typically with various lengths, before the optimization for packet scheduling takes place. Therefore, the packet content cannot adapt to the user request interactively. Additionally, accurate rate control is difficult to achieve due to the predetermined various packet lengths.

Packetization of scalable multimedia bitstreams is also addressed, e.g., by the PZW (Packetized Wavelet Zerotree) compression scheme [5]. Rate-distortion optimality of packetization as well as resiliency to packet losses are further discussed by Wu et al. [6]. These schemes pack the scalable bitstreams into packets with desired length, hence provide exact rate control. However, they do not support applications where the previously transmitted data can be reused and therefore need to be considered.

More recently, the JPIP standard is proposed for interactive viewing of JPEG2000 images in a client-sever manner [7]. The client identifies a region and maximum resolution of interest. The sever responds by rate-distortion optimally sequencing the JPEG2000 coded data while taking previous responses into account. However, it does not explicitly consider the handling of packet delay and losses encountered during the viewing sessions.

In [8], we present a sender-based rate-distortion optimized framework for interactive streaming of light fields coded as scalable bitstreams, where the sender carries out the optimization process. The computational burden at the sender may hamper the implementation in media servers that serve streams to hundreds of clients simultaneously. In this paper, we propose a receiver-based framework that moves the optimization task to the receiver as well as relaxes several limitations in the previous work.

The remainder of the paper is organized as follows: In Section 2, we briefly introduce the light field rendering and coding scheme adopted in this work. In Section 3, we describe the proposed receiver-based light field streaming framework. Finally, experimental results comparing with a heuristic scheme as well as the sender-based scheme are presented in Section 4.

2. LIGHT FIELD RENDERING AND CODING

We adopt a light field rendering scheme based on the principles proposed for *unstructured lumigraph rendering* [9]. To render a novel view, we choose the N_r camera views in the data set which are potentially most similar to the novel view as the *reference views*. For our light field data sets, we only consider the difference in viewing angles for the measurement of similarity [8].

We assume that a 3-D geometry model of the target scene is available [9]. The geometry model facilitates warping of each reference views to the desired novel viewpoint. The resulting N_r versions of the novel view are then combined by taking a weighted average. The weight associated with each reference view is inversely proportional to the difference of the viewing angles between the novel view and the reference view. Note that the proposed streaming framework is also applicable to generic light field rendering methods that do not incorporate a geometry model [1].

In this work, we use a simple coding scheme that encodes each camera view as an independent image. However, the proposed streaming framework can be extended to schemes where the coherence among views is exploited [10]. A multi-level 2-D shape-adaptive discrete wavelet transform (SA-DWT) is first applied to decompose each view into wavelet coefficients. To encode these coefficients, the SPIHT (Set Partitioning in Hierarchical Trees) [11] algorithm is chosen to provide a scalable bitstream so that different reconstruction qualities can be obtained by truncating the coded bitstream at different lengths. To render a novel view, the truncated bitstreams of its reference views available at the decoder are decoded into reconstructed wavelet coefficients by the inverse SPIHT algorithm. Then the inverse 2-D SA-DCT is applied to reconstruct the reference views.

3. INTERACTIVE LIGHT FIELD STREAMING

We consider the scenario where the N camera views, $C_n, n = 0, \cdots, N-1$, in a light field are independently encoded as N scalable bitstreams stored at the sender. Assuming that the light field renderer at the receiver renders a frame after every time duration T_d, we define a sequence of *assigning instants*, $\{a_k\}, k = 0, 1, \cdots$, where $a_{k+1} = a_k + T_d$. At a_k, the desired viewpoint of the frame, F_k, to be rendered, is assigned based on the user navigating behavior. Given the maximum allowable delay, D_{max}, from a_k to the time the data used for rendering F_k is decoded, a sequence of *decoding instants*, $\{d_k\}$, is defined as $d_k = a_k + D_{max}$.

We also define a sequence of *requesting instants*, $\{r_i\}, i = 0, 1, \cdots$, where $r_{i+1} = r_i + T_r$. At r_i, the receiver initiates a request for certain data from the sender. The request is optimized such that the resulting distortion in the reconstructed frames after receiving the requested data is minimized. It takes T_p processing time to generate the request, and the request is then fragmented into N_p requesting packets all issued at $r_i + T_p$ through the backward channel of the network. Upon receiving a requesting packet, the sender transmits a responding packet containing the requested data to the receiver through the forward channel. The N_p requesting packets from the request as well as the corresponding responding packets are indexed by $iN_p + q$ where $q = 0, \cdots, N_p - 1$.

At the beginning of each rendering session, the camera parameters of each camera view and the 3-D geometry model of the target scene are transmitted to the receiver. Both the backward and the forward channel are modelled as an independent time-invariant packet erasure channel with random delays. Each packet is delayed or lost independently from other packets. The cumulative distribution function of the round trip time (RTT) is denoted as $F_{RTT}(\tau) = Pr\{RTT \leq \tau\}$ [4]. Additionally, we denote the target forward payload transmission rate by C_f, measured in bit-per-second. Given C_f and T_r, the maximum amount of data that can be requested at each requesting instant is determined by $L = \lfloor T_r \frac{C_f}{8} \rfloor$ in bytes.

3.1. Packet Buffer and Decoding Buffer

A requesting packet as well as the content of a responding packet can be described by two N-vectors, $\mathbf{s}$ and $\mathbf{e}$. The elements of $\mathbf{s}$ and $\mathbf{e}$, denoted by s_n and e_n, represent the preceding and the ending position (in bytes) of the bitstream for C_n contained in the packet respectively. In other words, the bitstream segment for C_n starts at byte $s_n + 1$ and ends at byte e_n of the bitstream.

The data received at the receiver are decoded only at $\{d_k\}$, as opposed to upon each packet arrival. Between d_{k-1} and d_k, the receiver continuously receives packets and keeps them in the *packet buffer* which holds up to N_{pb} most recently received packets. In addition, the *decoding buffer* stores the bitstreams for the N_{db} most recently referred views. The decoding buffer state can be described with an N-vector, $\mathbf{b}$, where the element b_n denotes the length (in bytes) of the bitstream for C_n in the decoding buffer.

At d_k, we repeatedly update the decoding buffer by the packets in the packet buffer in the ascending order of their indices. A bitstream segment in the packet can update the decoding buffer only if it satisfies all of the following three conditions. First, the segment belongs to one of the N_{db} most recently referred camera views (including those for rendering F_k). Second, every bit in the bitstream preceding the segment is already in the decoding buffer, i.e., $s_n \leq b_n$. Third, the segment extends the bitstream already in the decoding buffer, i.e., $e_n > b_n$. After d_k, the packets are still kept in the packet buffer since they may be needed in later decoding instants, for instance, for those bitstream segments contained in the packets that later satisfy the above conditions.

3.2. Rate-Distortion Optimized Data Request

We define the rate for coding C_n as the length of the bitstream decoded to reconstruct C_n. The distortion is defined as the mean squared reconstruction error of the reconstructed C_n. During SPIHT-encoding for C_n, rate and distortion are recorded whenever there is a change in the distortion due to the rate increment, resulting in a sequence of $(R_n^{j_n}, D_n^{j_n})$ pairs with $j_n \in \{0, \cdots, j_n^{max}\}$ indexing the recording order. $R_n^{j_n}$ is expressed in bytes but is not necessarily a integer. Here we first assume that these R-D pairs are transmitted to the receiver before the rendering session. The first R_n^0 bytes of a bitstream contain essential header information for the bitstream to be decodable. Therefore, we assume that these initial segments are also transmitted before the rendering session. The receiver can always reconstruct a view at the lowest quality when no subsequent bitstream segments are received.

To render F_k, we denote the rendering weight for C_n by w_k^n. Note that w_k^n is nonzero only if C_n is a reference view for F_k. To optimize the request at r_i, we assume it is given that the first $\hat{b}_n$ bytes of the bitstream for C_n, where $\hat{b}_n \in \{R_n^{j_n}\}$, are already in the decoding buffer by the time the responding packets update the decoding buffer. In general, the decoding buffer state in the future is a variable. Prediction of the state is discussed in Section 3.4.

We define that r_i is associated with a window of frames to be rendered, from $F_{m_1(i)}$ to $F_{m_2(i)}$, and hence a set of camera views that are needed to render these frames. The indices of this set of views are denoted as $\mathcal{Z}_i = \{n : \bigcup_{m_1(i) \leq k \leq m_2(i)} w_k^n \neq 0\}$, and only the bitstreams of these views are requested at r_i. The simplest case would be associating r_i only to the most recently assigned frame, i.e., $m_1(i) = m_2(i) = \max\{k : a_k \leq r_i\}$. We can take into account more bitstreams in the request by using a smaller $m_1(i)$ to additionally consider the previously assigned frames, as well as using a larger $m_2(i)$ along with prediction of the user navigating behavior to consider the possible future frames.

The task of rate-distortion optimized data request is to determine the length of the additional bitstream segment for each C_n where $n \in \mathcal{Z}_i$ to be requested in order to minimize the expected total distortion in the rendered frames up to frame $F_{m_2(i)}$, subject to the target forward transmission rate constraint.

We assume that the distortion in a reconstructed camera view is proportional to that of the corresponding wavelet coefficients, and warping a particular view to a different viewpoint preserves the distortion originally in the view. We further assume that the reconstruction error in one view is uncorrelated to that in any other view, and hence the distortions from different reference views are additive in the rendered frame. Based on these assumptions, minimizing the expected total distortion is equivalent to:

$$\text{minimize} \sum_{k=0}^{m_2(i)} F_{RTT}(d_k - r_i - T_p) \sum_{n \in \mathcal{Z}_i} (w_k^n)^2 D_n^{j_n} \tag{1a}$$

$$\text{subject to} \sum_{n \in \mathcal{Z}_i} (R_n^{j_n} - \hat{b}_n) \leq L \tag{1b}$$

$$R_n^{j_n} \geq \hat{b}_n, \quad n \in \mathcal{Z}_i \tag{1c}$$

Note that the distortion in F_k is weighted by $F_{RTT}(d_k - r_i - T_p)$ since a responding packet contributes to the distortion reduction in F_k only if it arrives by d_k. The unknown variable in (1) are j_n for every $n \in \mathcal{Z}_i$. j_n determines the final bitstream length $\hat{R}_n = R_n^{j_n}$ and its corresponding distortion $\hat{D}_n = D_n^{j_n}$ for C_n.

Finally the optimized request is fragmented into N_p requesting packets, and each is responded by one responding packet from the sender. Ideally, the fragmentation is applied such that the bitstream segment for a view is entirely contained in a single packet to avoid dependencies among the N_p packets. However, we set a maximum packet length in order to avoid further packet fragmentation incurred in the network. Thus a bitstream segment too long to fit into one packet is divided into several segments, and each is requested in a different requesting packet.

3.3. Convex Optimization Approximation

The formulation in (1) is a combinatorial optimization problem. To reduce the complexity, we approximate the formulation by a convex optimization problem. Specifically, since the recorded $(R_n^{j_n}, D_n^{j_n})$ pairs are usually densely located and they approximately form a decreasing convex function they can be well fitted with a weighted sum of S terms of exponential functions, resulting in the continuous distortion-rate function $D_n(R_n)$:

$$D_n(R_n) = \sum_{s=1}^{S} c_{n,s} \cdot exp(-\lambda_{n,s} \cdot R_n), \quad c_{n,s} \geq 0 \qquad (2)$$

$D_n(R_n)$ is a convex function. In addition, it has analytically derivable gradient and Hessian, which greatly facilitate the optimization process. Furthermore, only $2S$ parameters instead of a long list of R-D pairs for each view now need to be transmitted. As a result, (1) can be replaced by:

$$\text{minimize} \sum_{k=0}^{m_2(i)} F_{RTT}(d_k - r_i - T_p) \sum_{n \in \mathcal{Z}_i} (w_k^n)^2 D_n(R_n) \qquad (3a)$$

$$\text{subject to} \sum_{n \in \mathcal{Z}_i} (R_n - \hat{b}_n) \leq L \qquad (3b)$$

$$\hat{b}_n \leq R_n \leq R_n^{j_n^{max}}, \quad n \in \mathcal{Z}_i \qquad (3c)$$

The formulation in (3) consists of a convex cost function with linear constraints, thus formulates a convex optimization problem that approximates the original problem in (1). It can be solved efficiently using standard convex optimization techniques. To conform with the original problem, the optimal solution of R_n should be rounded to the nearest $R_n^{j_n}$. However, for an efficient representation of the request, we round it to the nearest integer.

3.4. Decoding Buffer State Prediction

To request the bitstream segment of C_n, we choose to optimize the request at r_i according to the predicted decoding buffer state at decoding instant $d_{k(i,n)}$, i.e., use this prediction as the $\hat{b}_n$ in (3). $d_{k(i,n)}$ is defined as the first among the decoding instants at which C_n is needed , and by which the packets responding to the request issued at r_i is more probable to arrive the receiver than not, i.e.,

$$k(i,n) = \min\{k : w_k^n \neq 0, F_{RTT}(d_k - r_i - T_p) > 0.5\} \qquad (4)$$

We assume that there are N_u packets that have been previously requested but have not yet arrived at the receiver at r_i. The indices of these packets are denoted by $u(q)$, $q = 1, \cdots, N_u$. For each of these packets, the probability of its arrival by a particular decoding instant d_k given it has not yet arrived at r_i can be expressed by:

$$P_a(i,k,u(q)) = Pr\{RTT \leq d_k - r_{\lfloor u(q)/N_p \rfloor} - T_p \mid RTT > r_i - r_{\lfloor u(q)/N_p \rfloor} - T_p\} \qquad (5)$$

There are 2^{N_u} possible arrival combinations of these packets regarding to d_k, and each results in a possible decoding buffer state. Since packet delay and losses are assumed to be independent across packets, the probability of each possible decoding buffer state is simply the product of that of the outcome of each packet.

Ideally, optimization should be applied for each possible state at $d_{k(i,n)}$, and the request that results in the minimum expected distortion calculated over all possible arrival combinations is chosen. However, this can lead to a long processing time since N_u can become large. Observing that the probability distribution of the possible states is highly skewed, the unlikely states can therefore be neglected. For simplicity, we choose the most probable decoding buffer state at $d_{k(i,n)}$ as the prediction. Consequently, the packet with index $u(q)$ is predicted to arrive by $d_{k(i,n)}$ if and only if $P_a(i, k(i,n), u(q)) > 0.5$.

At each r_i, a *predictive buffer* is created at the receiver. The buffer first duplicates the current state of the decoding buffer as the initial state. To further predict the decoding buffer state for C_n at $d_{k(i,n)}$, the predictive buffer is sequentially updated, in the same way as the decoding buffer, by the packets already in the packet buffer and the not-yet-arrived packets with $P_a(i, k(i,n), u(q)) > 0.5$. Finally, rate-distortion optimized data request in (3) is carried out with $\hat{b}_n$ being the predictive buffer state.

To account for the unavoidable prediction error of the decoding buffer state, we make a simplified assumption that the bitstream segment for C_n does not contribute to any distortion reduction in F_k if $\hat{b}_n$ does not exactly match the actual state b_n at d_k. Therefore, $D_n(R_n)$ in (3a) is further weighted by the probability that the predictive state $\hat{b}_n$ actually occurs at each d_k.

4. EXPERIMENTAL RESULTS

Experimental results are shown for the data set *Buddha* and *Bust*. There are $N = 281$ views in *Buddha* and $N = 339$ for *Bust*. We choose the parameters $N_r = 4$, $N_p = 4$, $S = 5$, $N_{db} = 16$, $N_{fb} = 200$, $T_d = 100$ ms, $T_r = 50$ ms, $T_p = 50$ ms, $D_{max} = 200$ ms, and only the luminance component is considered. The probability density function of the packet delay, given the packet is not lost, is shifted-gamma distributed with a 10 ms shift, 40 ms mean and 300 ms^2 variance, and the packet loss rate is set to 2%, for both the forward and the backward channel.

Each parameter of the fitting exponential functions is quantized and coded with 20 bits, resulting in a 25-byte overhead per camera view that is transmitted before the rendering session, along with the initial segment (typically around 10 bytes) of each bitstream. We experiment with two navigation trajectories, one for each data set. They both contain 100 frames and simulate the viewing experience of the scene being rotated in various directions.

We choose the target forward payload rate, C_f, from 64 up to 256 kbps. The maximum packet length is set to 1000 bytes. The payload in the requesting packet is entropy-coded with a backward payload rate typically within 1% of C_f. Performance is evaluated using the rate-PSNR curve. The rate is represented as the average forward payload rate, which slightly deviates from C_f since

the losses of the requesting packets result in some idle time in the forward transmission. The PSNR values are computed using the frames rendered from the original camera views as the reference.

We first compare the proposed optimized scheme with a heuristic scheme. The heuristic scheme share the same receiver-based framework as the optimized scheme, except that the amount of the additional bitstream segments to request is determined differently. In both schemes, a request is associated only with the most recently assigned frame. In the heuristic scheme, the forward rate is equally distributed among the needed bitstreams that are not in the predictive buffer (except for the initial segments). If all the needed bitstreams are in the predictive buffer, the rate is equally distributed among all of them. The performances are shown in Figure 1. The optimized scheme performs consistently better than the heuristic scheme, with a PSNR gap of $1.0 \sim 1.8$ dB at the same rate for *Buddha* and $0.5 \sim 0.9$ dB for *Bust*. Correspondingly, to achieve the same rendering quality the optimized scheme can save $20\% \sim 35\%$ of the rate over the heuristic scheme.

We further consider the cases where a request is additionally associated with the next frame in the future, with two prediction scenarios. The first assumes perfect prediction of the reference views and the corresponding weights. The second assumes partial prediction that successfully predicts the reference views but distributes equal weights among them. The partial prediction scenario, which is more practically achievable, provides an additional gain of $0.3 \sim 0.8$ dB over the single-frame case. The performance loss of assuming equal weights is about $0.2 \sim 0.4$ dB.

Finally, we compare the proposed receiver-based framework with a sender-based framework modified from [8]. In the sender-based framework, the request from the receiver indicates the most recent desired view-point along with the indices of the packets in the packet buffer. Optimization of the response is performed at the sender. The optimized allocation now has to be signalled to the receiver as overhead of the responses, hence has to be included in the average forward rate. As shown in Figure 1, the sender-based framework outperforms the receiver-based one for the single-frame case. This is mostly due to that the former only have to cope with the random packet delay and losses in the forward channel, whereas the latter has to deal with the randomness in both channels. Nevertheless, the receiver-based framework removes the computational burden at the server, hence allows the server to simultaneously serve a large number of receivers. In addition, the sender is not involved in calculating the rendering weights as well as predicting the user behavior, thus can support different rendering methods and applications at the receiver.

5. CONCLUSIONS

We propose a receiver-based rate-distortion optimized framework to interactively stream scalable bitstreams of light fields. Based on the viewpoints assigned by the user and prediction of the future viewpoints, the predicted receiver buffer state, the network characteristics, and the target forward payload rate, the receiver performs rate-distortion optimized data request as a convex optimization process. Experimental results show that rate-distortion optimization together with prediction of the future frame improves the rendering quality by $1.0 \sim 2.6$ dB over the heuristic scheme at the same bit-rate. Although the receiver-based framework in general performs worse than the sender-based framework, it allows the sender to simultaneously serve a large number of receivers.

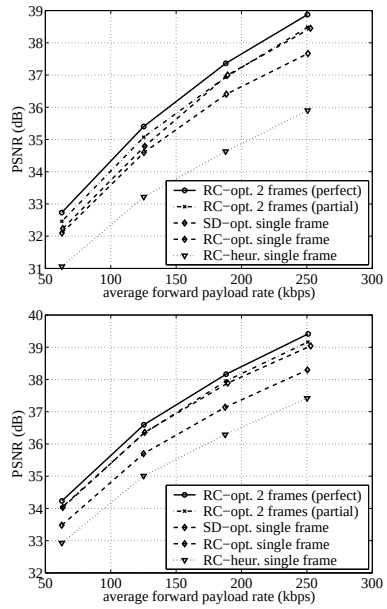

Fig. 1. (top)*Buddha* (bottom) *Bust*

6. REFERENCES

[1] M. Levoy and P. Hanrahan, "Light field rendering," in *Computer Graphics (Proceedings SIGGRAPH 96)*, Aug. 1996, pp. 31–42.

[2] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph," in *Computer Graphics (Proceedings SIGGRAPH 96)*, Aug. 1996, pp. 43–54.

[3] P. Ramanathan, M. Kalman, and B. Girod, "Rate-distortion optimized streaming of compressed light fields," in *Proc. IEEE Int. Conf. on Image Processing 2003*, Barcelona, Spain, Sept. 2003.

[4] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *(submitted) IEEE Trans. Multimedia*.

[5] J. Rogers and P. Cosman, "Wavelet zerotree image compression with packetization," *IEEE Signal Processing Lett.*, vol. 5, pp. 105–107, May 1998.

[6] X. Wu, S. Cheng, and Z. Xiong, "On packetization of embedded multimedia bitstreams," *IEEE Trans. Multimedia*, vol. 3, no. 1, pp. 132–140, Mar. 2001.

[7] D. Taubman and R. Rosenbaum, "Rate-distortion optimized interactive browsing of JPEG2000 images," in *Proc. IEEE Int. Conf. on Image Processing 2003*, Barcelona, Spain, Sept. 2003.

[8] C.-L. Chang and B. Girod, "Rate-distortion optimized interactive streaming for scalable bitstreams of light fields," in *SPIE Visual Communications and Image Processing, San Jose, USA, Jan. 2004*.

[9] C. Buehler, M. Bosse, L. McMmillan, S. Gortler, and M. Cohen, "Unstructured lumigraph rendering," in *Computer Graphics (Proceedings SIGGRAPH 01)*, Aug. 2001, pp. 425–432.

[10] B. Girod, C.-L. Chang, P. Ramanathan, and X. Zhu, "Light field compression using disparity-compensated lifting," in *Proc. IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 2003*, Hong Kong, China, Apr. 2003.

[11] A. Said and W. A. 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.

[1]This work was supported by Grant No. ECS-0225315 of the National Science Foundation and the Max Planck Institute. The *Buddha* and *Bust* data sets used for the work are courtesy of Intel Research.