Layer-Optimized Streaming of Motion-Compensated Orthogonal Video

Wenjie Shen

Master Thesis
Stockholm, Sweden 2013

XR-EE-KT 2013:008
Layer-Optimized Streaming of
Motion-Compensated Orthogonal Video

Wenjie Shen
School of Electrical Engineering
KTH Royal Institute of Technology, Stockholm
wenjies@kth.se

August 2013
Abstract

This report presents a layer-optimized streaming technique for delivering video content over the Internet using quality-scalable motion-compensated orthogonal video. We use Motion-Compensated Orthogonal Transforms (MCOT) to remove temporal and spatial redundancy. The resulting sub-bands are quantized and entropy coded by Embedded Block Coding with Optimized Truncations (EBCOT). Therefore, we are able to encode the input video into multiple quality layers with sequential decoding dependency. A layer-distortion model is constructed to measure the trade-off between expected streaming layer and expected distortion. Due to the sequential dependency among streaming layers, we build a cost function with concave properties. With that, we develop a fast algorithm to find the optimal transmission policy at low computational complexity. The experiments demonstrate the advantages of expected distortion and computational complexity for challenging streaming scenarios.
Contents

1 Introduction ......................................................... 3
  1.1 Background .................................................. 3
    1.1.1 Video delivery over the Internet ..................... 3
    1.1.2 Challenges in video streaming ....................... 5
  1.2 Decoding Dependencies ....................................... 6
  1.3 Literature Review .......................................... 7
    1.3.1 Non rate-distortion optimized streaming .......... 7
    1.3.2 Rate-Distortion optimized streaming ............ 7
  1.4 Motivation and contribution ............................... 9
  1.5 Outline ..................................................... 9

2 Motion-Compensated Orthogonal Video .......................... 11
  2.1 Motion-Compensated Orthogonal Transform ................. 12
  2.2 Adaptive Spatial Transform ................................ 12
  2.3 Embedded Block Coding with Optimized Truncations ...... 13
  2.4 Packetization .............................................. 15

3 Layer-Optimized Streaming ...................................... 17
  3.1 Preliminaries .............................................. 17
  3.2 Layer-Distortion Model .................................... 18
    3.2.1 Motivation of layer-distortion model ............ 18
    3.2.2 Layer-distortion trade-off ....................... 19
  3.3 Optimal Streaming Layer .................................. 21
    3.3.1 Cost function ...................................... 21
    3.3.2 Properties of cost function ..................... 21
    3.3.3 Fast algorithm ..................................... 22

4 Experimental Results .......................................... 23
  4.1 Experimental Setup ........................................ 23
  4.2 Reference Scheme .......................................... 24
    4.2.1 Cost function ...................................... 24
    4.2.2 Iterative sensitivity adjustment algorithm .... 26
    4.2.3 Smooth rate control ................................ 26
Chapter 1

Introduction

Multimedia information on the Web continues to increase with the explosive growth of the Internet. Because of the high volume of video, delivering video content is especially a challenging problem under a limited bandwidth constraint. Video content delivery over the best effort network can be classified into two approaches, via file downloading or via streaming. The former approach requires the entire video to be downloaded before the user can start viewing. In contrast, the latter approach enables simultaneous delivery and playback of the video. Therefore, video streaming is of great interest to the multimedia communication industry [1], meeting short delay and low storage requirements.

In this chapter, Sec. 1.1 first gives the background of video streaming in the Internet environment. Sec. 1.1.1 summarizes the framework of our video streaming system. Sec. 1.1.2 outlines the basic problems in video streaming. We discuss the decoding dependencies of several video coding algorithms in Sec. 1.2. This will motivate our proposed strategy. Sec. 1.3 introduces several video streaming schemes and their advantages and disadvantages, where the rate-distortion optimized streaming is of great interest. The following Sec. 1.4 summarized the motivation and contribution of this report. Finally, Sec. 1.5 gives an outline of each chapter’s individual work.

1.1 Background

1.1.1 Video delivery over the Internet

Streaming of video content can be classified into two categories - stored video and live video. Storage of video content is quite widely used in DVD and video on demand. Since the video content is pre-encoded, high coding efficiency can be achieve. Live broadcast of sport events and interactive applications require the video content to be encoded in real time, and demand low end-to-end delay. Video streaming over the Internet is normally the streaming of stored video content.
Fig. 1.1 shows the basic framework of video streaming over Internet. First, the video sequences are divided into groups of pictures. Then the encoder encodes each group of pictures (GOP) into a bitstream. The bitstream is then packed into IP packets. For each packet, the streaming server allocates a set of transmission opportunities. At each transmission opportunity, the streaming server schedules several packets into a sending buffer. The selected packets in the sending buffer are then sent to the client during the time interval between successive transmission opportunities. These packets travel through a lossy forward channel and arrive at the client under some delay. The client then sends feedback to the streaming server through a backward channel.

At the client, those received packets form a shortest decodable bitstream. Finally, the decoder reconstructs the video sequences from this bitstream. Note, the reconstruction quality relies not only on the number of received packets, but also on the decoding dependencies among them. These decoding dependencies among packets will be discussed in Sec. 1.2.

In general, a streaming server is a specialized application which is able to detect the connection speeds of users, handle large traffic loads and broadcast live events. As we have mentioned before, the streaming server listens to the feedback from the client and schedules packets at each transmission opportunity. In other words, a streaming server decides when and which packets to send with the knowledge of current network condition and feed-
Layer-optimized Streaming of Motion-Compensated Orthogonal Video

back. Examples of streaming media software are Helix Universal Server from RealNetWorks [2], Apple Quicktime Streaming Server [3] and Macromedia Communication Server [4].

A practical video streaming system has more complex structure than the framework described above. In general, there are six key areas in video streaming - Video compression, application-layer QoS control, continuous media distribution services, streaming servers, media synchronization mechanisms, and protocols for streaming media [5]. However, this simple framework is adequate because our interest concentrates on the streaming server. At the same time, video compression is also stated in detail for its crucial importance for our proposed streaming scheme design.

1.1.2 Challenges in video streaming

Video streaming over the Internet is difficult since the Internet only provides best-effort service but no guarantees on bandwidth, jitter and loss. Therefore, these are the three fundamental problems in video streaming. All these parameters are unknown and dynamic. The goal of video streaming is to design a system that delivers reliably quality video over the Internet.

Bandwidth limitation

The bandwidth between a sender and a receiver is often unknown and time-varying. Congestion occurs when packets are sent faster than the available bandwidth. In other words, more packets are lost during travelling through the forward channel thus the video quality drops. On the other hand, if packets were sent slower than the available bandwidth, then the residual bandwidth, which can be used to improve the video quality, is wasted. The solution for this problem is to have efficient bandwidth estimation [6] and match the sending rate with the available bandwidth.

Jitter

The sending order is scheduled by the streaming server and packets are sent consecutively during the transmission interval. Since the inter-arrival time between successive packets varies, the received packets are not in the same order as they are at the streaming server. Jitter refers to the variation of travelling time delays. Some packets may be received very late due to delay jitter, then the reconstructed video may experience jerks. Having a playout buffer [7] at the receiver is an efficient way to solve this problem.

Packet loss

Depending on the particular network, different types of losses may occur. The Internet is usually afflicted by packet loss, where an entire packet is
lost while wireless channels are typically afflicted by bit or burst errors. The error control mechanisms can be basically grouped into three categories [8]: forward error concealment (FEC), error concealment by postprocessing and interactive error concealment. Retransmission is a typical interactive error concealment method which relies on a dialogue between the source and destination. In this report, layered coding (a forward error concealment technique) and retransmission are the only error control techniques used in the following discussion.

1.2 Decoding Dependencies

The decoding dependencies among packets play an important role in streaming policy design. Regardless of which type of transform is performed, there are no directed cycles among encoded frames, thus a directed acyclic graph (DAG) is able to describe the decoding dependencies among video packets.

Let each node or vertex in the graph represent a video packet and each directed edge illustrates the dependency between two connected nodes. A destination node is connected to its origin by a directed edge. In other words, the origin is the ancestor of the destination node. If a packet has more than one ancestor packets, it can only be decoded when all of its ancestors are decoded successfully.

Examples of DAGs are shown in the Figure below. Fig. 1.2(a) shows the typical embedded encoding with sequential dependency. Take JPEG 2000 for example, if we receive packet 3, but packet 2 is lost, then this packet 3 becomes useless because it cannot be decoded without packet 2.

![Directed acyclic dependency graphs](image)

Figure 1.2: Directed acyclic dependency graphs

Fig. 1.2(b) shows typical hierarchical decoding dependencies of predictive coding. We take H.264/AVC for example, where three kinds of video packets in a GOP, namely an I packet, P packets and B packets. If a B packet is
received while its P packet is lost, this B packet becomes useless because it can only be decoded if the P packet has been received.

1.3 Literature Review

Video streaming systems can be classified into two categories: rate-distortion optimal and non rate-distortion optimal. Non rate-distortion optimized streaming systems are usually of low complexity while rate-distortion optimized streaming systems in most cases offer better performance. In this section, we will introduce these two video streaming systems and point out their advantages and disadvantages.

1.3.1 Non rate-distortion optimized streaming

A reliable error control method that is widely used in communication industry is to incorporate error detection with Automatic Repeat reQuest (ARQ) [9]. Based on this simple technique, Podolsky et al. [10] defines a "soft ARQ" framework for streaming delay-constrained media with retransmission at fixed rate. It is not rate-distortion optimized. With the assumption of instantaneous feedback, this soft ARQ method is able to derive an optimal transmission policy of layered data over a lossy channel. It also finds that the optimal strategy is time-invariant under fixed network conditions. The limitation is that the soft ARQ method do not take the network delay and layer distortion information into account. These simplification do not reflect the properties of the Internet property and lead to inefficient use of layer information. Furthermore, when allowing more retransmission opportunities per frame, the transmission policy space grows exponentially. This consequence challenges the complexity if extended this system to be more sophisticated.

1.3.2 Rate-Distortion optimized streaming

Basically, the streaming server of the rate-distortion optimized system aims at minimizing the end-to-end reconstruction error under the constraint of rate limitation over the entire video presentation:

\[
\min D(R) \\
\text{s.t. } R \leq R_{\text{max}},
\]

(1.1)

where \( D \) denotes the overall distortion and \( R \) denotes the total number of bytes transmitted.

To our knowledge, the general work of rate-distortion optimized streaming is done by Chou and Miao [11]. Results show that rate-distortion optimized streaming systems have a steady-state gain of 2 – 6 dB or more.
Layer-optimized Streaming of
Motion-Compensated Orthogonal Video

over non rate-distortion optimal systems. The contribution of [11] is that the rate-distortion optimization of the whole video presentation is simplified to the error-cost optimized transmission of a single packet. Furthermore, the same method can be applied to other scenarios, thus it is easily extended to various transmission scenarios. Although the derived policy is near-optimal, this isolated optimization method has been proofed to be quite close to the operational rate-distortion optimization. And the computational complexity is considerably lower. Meanwhile, this near-optimal policy also brings a vital disadvantage. The rate control mechanism of this system utilizes a bisection search method to smooth the instant rate. Therefore, once a near-optimal policy is derived instead of the optimal one, there is a high chance that the error propagates and leads to the failure of obtaining a desired instant rate.

Several follow-up works have been done based on above framework. To our knowledge, most of them carry on the rate-distortion optimized framework and apply it to other specific scenarios, such as receiver-driven transmission over best-effort networks [12], multiple access networks [13] [14], differentiated services networks [15] and wireless networks [16].

Among all of those follow-up work, some have improved the above framework to be more sophisticated. For example, Chakareski et al. [17] takes self congestion into consideration. Based on the current transmission rate of the optimization algorithm, it computes the transmission policy in network bottleneck links with instantaneous rate control and dynamic update of delay and loss probabilities. This extension provides smoother output rate compared to that of conventional rate-distortion optimized streaming.

In [18], Kalman found that the rate-distortion performance can be improved significantly by associating packets with multiple deadlines. It is reasonable for the case of H.264/AVC compressed video. Because of the use of bi-directional prediction, decoders could recover from late packet arrivals through a method called accelerated retroactive decoding. Experimental results proofed that this extended work outperforms the original rate-distortion optimized streaming in most cases. However, this improvement can hardly be applied on embedded encoded video packets where all the frames within a GOP have identical decoding deadlines.

In all, Chakareski et al. [19] have shown the importance of rate-distortion optimization for streaming over the Internet. This paper compares rate-distortion optimized streaming with simple ARQ techniques for lossy and lossless traces. Results show that rate-distortion optimized streaming is an efficient way to deliver video contents especially over lossy networks. We will introduce the fundamental rate-distortion optimized streaming scheme of [11] in the experiments to compare with the proposed scheme.
1.4 Motivation and contribution

As mentioned in the above chapter, the state-of-art approach has already found a near-optimal solution to this rate-distortion optimal streaming problem for video packets with arbitrary decoding dependencies. However, finding the optimal streaming policy is NP-hard and the performance degrades when taking rate overshooting into account. Since the computational complexity is affected by the decoding dependencies among packets, embedded coding with sequential dependency is a good candidate for efficient streaming. Furthermore, sequential dependencies permit smooth instant rates.

Therefore, in order to generate video packets with sequential dependency, we use a subband video coding scheme to obtain quality scalable layers with sequential dependency. Since packets are transported over a lossy network, and overshooting the bandwidth is not favored. Hence, a layer-distortion model is built to measure the tradeoff between rate and distortion.

This report has two main contributions. First, by exploiting sequential dependency, we lower the computational costs. We will see from an analysis that the computational complexity is significantly lower. Second, the output rate we achieve adapts better to the channel bandwidth, thus the channel bandwidth is utilized more efficiently.

1.5 Outline

This chapter provides the background for video streaming and introduces some streaming systems. The motivation of the proposed scheme is given after comparing some existing streaming systems. Now we present the major ideas in this report as follows:

Chapter 2 introduces a subband video coding scheme to generate quality-scalable motion-compensate orthogonal video. Motion-compensated orthogonal transforms followed by adaptive spatial transforms are performed to exploit both temporal and spatial redundancies in order to obtain high energy compaction. The resulting subbands are then quantized and entropy coded into quality layers by EBCOT. In the packetization part, we show how to package the side information together with the data.

Chapter 3 shows the framework of our proposed scheme and how to derive the cost function. In our proposed scheme, we assume that the network random packet loss is known and the chance of packet loss increases immediately when overshooting occurs. Currently sent packets has no effect on subsequent ones, i.e., the impulse drop of packets happens in the present transmission opportunity and all transmission opportunities are independent from each other. In order to make full use of the sequential dependencies to develop a simple and efficient algorithm, a layer-distortion model is built to speed up the packet selection at each transmission opportunity. A fast
algorithm to find the optimal solution is also provided.

Chapter 4 compares experimental results of the proposed scheme to a reference scheme, which is briefly introduced in Sec. 4.2. By testing on a long video sequence, we display the performance of both schemes in terms of instant output rate, actual packet loss, PSNR over packet loss and computational complexity.

Chapter 5 concludes the work of this report.
Chapter 2

Motion-Compensated
Orthogonal Video

In this chapter, we introduce motion-compensated orthogonal video. Energy compaction and sequential decoding dependency are the two main properties. To generate motion-compensated orthogonal video, we build on previous work.

Predictive coding, such as H.264/AVC which offers excellent video compression efficiency, is the current state of art standard in video coding and used for TV broadcasting and HD-DVD [20]. Because of the bi-prediction mechanism, the coded pictures heavily depend on the relationship among successive pictures. Therefore, this prediction mechanism introduces a high risk of error propagation. In a packet loss environment, this might be suboptimal. On the other hand, subband video coding has the property of energy concentration and conservation while avoiding error propagation. Thus, it is more suitable for packet-based networks like the Internet [21] [22]. Furthermore, the hierarchical decoding dependencies of H.264/AVC coding also brings challenge for streaming policy design. In contrast, a sequential decoding dependency among packets enables the potential to lower the complexity for streaming policies. This motivates us to utilize subband video coding to get both efficient compression and a simple decoding dependency. Our subband video coding scheme named motion-compensated orthogonal video coding is shown in Fig. 2.1.

Figure 2.1: Motion-compensated orthogonal video coding.


2.1 Motion-Compensated Orthogonal Transform

Optimal energy compaction is known to be reachable by the Karhunen Loeve Transform (KLT). However, the KLT is signal dependent, which means that we have to store the transform basis in order to recover the original signal through inverse transform. This is especially a waste of bit rate. In order to lower extra bit rate for side information, motion-compensated transforms are a good choice. A $l_2$-norm preserving motion-compensated transform, also known as Motion-compensated orthogonal transform (MCOT) [23], enables the MSE calculation to be easily performed in the frequency domain. Further, MCOT is an approximation of the KLT, but with considerably fewer side information needed to store.

![Diagram of motion-compensated orthogonal transform](image)

Figure 2.2: Motion-compensated orthogonal transform.

Here we give a brief introduction to the effect of input and output of this subband coding system:

Let the size of one Group of Picture (GOP) in the original image sequence be denoted by $G$. An $n$-level MCOT is used for each GOP such that $G = 2^n$. As shown in Fig. 2.2, $I_1 \ldots I_G$ is a group of pictures, we obtain $G$ temporal subbands of frequency coefficients and a set of motion vectors. Due to the energy compaction and conservation properties of MCOT, the energy of the input frames is accumulated in the first temporal low-band $L_1$ while the energies in the $H_1 \ldots H_{G-1}$ temporal high-bands are relatively small.

2.2 Adaptive Spatial Transform

As MCOTs are temporal transforms for video sequences, it is necessary to further spatially decomposed each temporal subband to exploit the spatial correlation within the resulting temporal subbands.

The simplest spatial transform is a $2 \times 2$ Haar transform, and orthogonality also holds. However, the Haar transform is data independent so that
high energy concentration is not ensured. Since the energy at high-bands is considerably lower when compared to that of the low-band, effects on the energy compaction ratio can be negligible. The Haar transform has very low complexity. We apply this simple transform on $G-1$ temporal high-bands $H_1 \ldots H_{G-1}$. For the temporal low-band, this low compaction ratio is not favored. Thus, we use an adaptive spatial transform called Type-2 transform [24] for the first low-band $L_1$. It efficiently moves the spatial energy of the first temporal subband to the spatial low band pixels.

Fig. 2.3 shows an example of the motion-compensated orthogonal transform and the adaptive spatial transform. A group of 4 pictures is extracted from the video sequence foreman.qcif. A two-level MCOT and a two-level adaptive spatial transform are successively performed on this GOP.

Figure 2.3: Illustration of video images after applying temporal and spatial space transform: (a) original video images of foreman with a resolution of 176 * 144, (b) temporal subbands after MCOT, (c) spatial subbands after adaptive spatial transform.

2.3 Embedded Block Coding with Optimized Truncations

The above transforms remove both temporal and spatial redundancy among consecutive pictures. The next goal is to obtain a quality-scalable video. Embedded coding is known to produce scalable output. An interesting algorithm of which is called Embedded Block Coding with Optimized Truncations (EBCOT). EBCOT exhibits a set of rich features, including rate scalability and SNR scalability [25]. Of all these features, SNR scalability is our target. So we choose this algorithm to quantize the resulting subbands into quality layers.
In EBCOT, the smallest coding unit is called a code-block, usually with the size of $16 \times 16$. First, each subband is divided into several code-blocks. EBCOT algorithm is built on fractional bit-plane coding. It encodes each block to generate its own independent fractional bitstream. Each fractional bitstream comprises of many shorter sub-streams. To make it easier, we call this sub-stream a ‘chunk’. So each bitstream is made up of chunks. Then the compressor extracts some chunks from all code-blocks according to a predefined coding style [26] to form a quality layer. In this way, each code block assigns an incremental contribution to each quality layer [27]. The truncation points of a bitstream are at the end-points of those chunks.

![R-D curves of B1 and B2](image1.png)

![Overall R-D curve](image2.png)

Figure 2.4: Example of PCDR-opt: (a) R-D curves of two independent code-blocks $B_1$ and $B_2$; (b) overall R-D curve

In order to generate rate-distortion optimized quality layers, a post-compression rate-distortion optimization (PCDR) algorithm is used to help collecting incremental contributions from all code-blocks into quality layers. [28] presents an integer-based PCDR algorithm to accelerate processing.

To briefly display how PCDR works, here we give a simple example. Suppose there are only two code-blocks $B_1$ and $B_2$ for the resulting subband and Fig. 2.4(a) shows their independent rate-distortion (R-D) curves. The overall R-D curve is a set of truncation points that fall on the convex hull of all the possible combinations. In other words, a truncation point on the overall R-D curve represent a pair of truncation points of block $B_1$ and $B_2$ that minimize the overall distortion under a certain rate constraint. The
PCDR algorithm is basically trying to find a common R-D slope $\lambda$ for both code-blocks. The total rate $R_1 + R_2$ and the corresponding overall distortion $D_1 + D_2$ is a rate-distortion optimal point on the convex hull of the overall R-D curve (shown in Fig. 2.4(b)). By setting a constraint $R_{\text{max}}$, we can always find a corresponding slope $\lambda'$ on the R-D curve. The set of optimal truncation points are found by matching individual code-block R-D curves at the point where exactly slope $\lambda'$ is reached. In this way, global rate-distortion optimality is preserved and the dependency between each consecutive layer is sequential.

According to the above, EBCOT is able to produce as many quality layers as possible. For packetized media, our interest focuses on the relation between quality layers and packets. We will conclude that the optimal solution is to have an equal amount of quality layers and packets. The necessity and sufficiency are proofed below:

Assume a quality layer is deployed into several video packets. If one of those packets is lost and all the rest are received, this quality layer can not be decoded. Then all the received packets waste of that quality layer are wasted. Therefore, a layer is better to be preserved in one video packet. On the other hand, if a video packet contains several consecutive layers, then those consecutive layers can be regarded as one layer. Because receiving or losing this video packet always benefits or degrades the distortion by the same amount. The amount of decreased distortion is the sum of those layer’s contribution. In all, it is most efficient to have exactly one layer in each packet. An example of a layer-distortion relationship can be found in Fig. 3.2.

For our interested scenario, the size of IP packets is fixed. Thus, the ideal solution is to have each quality layer length to be exactly the same as the packet size. Due to the discrete nature of truncation points, meeting the exact packet size can not be guaranteed. A suboptimal solution is to find a set of truncation points whose total size is not larger than and closest to a given packet size.

### 2.4 Packetization

After performing EBCOT, one GOP is encoded into a long bitstream with side information (i.e., a set of motion vectors). This side information is of great importance, without which decoding can not be performed correctly. So we need to include it in the packets to enable the receiver to reconstruct the video. Considering the need for accuracy and the small data volume of motion vectors, they are compressed without loss through entropy coding (e.g. Huffman coding). The total compressed motion coefficients consume up to several thousands (normally hundreds of bytes). The maximum length of the IP packet is 1500 Bytes, so we can not guarantee to pack those
coefficients into a single packet. We can divide it into several parts and pack those into several packets or spread them together with the data packets. Although these motion coefficients are vital for reconstruction, receiving them alone do not benefit for the video reconstruction. The compressed motion vectors occupy around 2% of total rate, which is a small amount. So we pack those entropy coded motion vectors together with the coefficient bitstream in a packet to simplify streaming.

We have discussed in the previous section that each layer should be contained in one video packet. Here a fixed length packet is used and we denote the target packet length by $N_p$. Due to the high importance of motion vectors, we always put them in the very first packets together with the data of each GOP. We define a budget $N_m$ of each packet for the motion vectors that $N_m < 0.5 \cdot N_p$. As the data volume of motion vectors is much smaller than that of the bitstreams, only a very few packets contain motion vectors. The rest of the budget is used for the coefficients and byte stuffing, if necessary.
Chapter 3

Layer-Optimized Streaming

In this chapter, we show how to take advantage of motion-compensated orthogonal video to make the best decision for a lossy network.

Our streaming scheme is shown in Fig. 3.1. First, we specify the environment of our video streaming framework by considering random packet loss and extra packet loss caused by overshooting. A layer-distortion model is constructed in Section 3.2. Section 3.3 reveals a fast algorithm to find the optimal solution to this proposed streaming framework.

3.1 Preliminaries

Noting that packets encoded within the same GOP have an identical decoding time stamp. Thus, we grant the group of video packets within the same GOP identical transmission opportunities from the set of transmission opportunities $\Phi = \{t_i, i = 1, \ldots, N\}$. $T = |t_{i+1} - t_i|$ is the time interval between two successive opportunities. At each transmission opportunity, our scheme schedules $K$ packets and includes them in the sending buffer for streaming. The packets are delivered over a lossy network with a pre-known packet loss ratio $\varepsilon$. The packet loss rate $\varepsilon$ comprises the random packet loss rate $\varepsilon_0$ and $\varepsilon_c$ caused by server overshooting. In our model, we aim at achieving the smallest possible expected distortion. However, increasing sending rate and decreasing distortion conflicts with each other when packet loss is present. In particular, the loss rate increases if the sending rate exceeds the
Layer-optimized Streaming of
Motion-Compensated Orthogonal Video

bandwidth limitation. Therefore, there is a tradeoff between sending rate and overall distortion. Our scheme allows retransmissions for lost packets by using feedback from the client. We set the decoding deadline (or decoding time stamp) of each GOP as $T_d$. Packets that are received later than this deadline will be discarded.

### 3.2 Layer-Distortion Model

![Directed acyclic dependency graph](image)

(a) directed acyclic dependency graph

![Layer distortion relationship](image)

(b) layer distortion relationship

Figure 3.2: Layer-distortion relationship.

#### 3.2.1 Motivation of layer-distortion model

Due to the energy compaction properties of MCOT and the bit-plane coding of EBCOT, energy is more concentrated in the packets at the beginning and the dependency between successive packets is sequential. In our case, in particular, the current packet can only be decoded if all earlier packets have been successfully received.

Therefore, sending the packets in natural order (i.e., from the base layer to the highest layer) is obviously an optimal streaming policy without feedback and retransmission. To adapt to a changing bandwidth and to avoid a possible network congestions, the streaming server is designed to adjust the output rate by scheduling an appropriate number of packets in the sending buffer. To accomplish this, a layer-distortion model is introduced to measure the relationship between expected streaming layer and reconstruction quality.
3.2.2 Layer-distortion trade-off

Generally speaking, the packet loss rate becomes relatively high when the output rate is higher than the current bandwidth. Once a packet is lost, our scheme has to retransmit it due to the decoding dependency. In such situations, it is desirable to decrease the expected streaming layers (to reduce the risk of packet loss). On the other hand, the coding distortion could be lower when sending more layers. In this situation, it is desirable to increase the expected streaming layers to reduce the coding distortion.

![Diagram](image)

Figure 3.3: Current streaming layer $l_c$ and expected streaming layer $L$

We are now in a position to introduce the measures that reflect the trade-off between expected streaming layers and distortion. We define the contribution of each layer $D_s(l)$ as the distortion decrement if layer $l$ is decoded on time. For instance, as shown in Fig. 3.3, at a transmission opportunity, the streaming server has already sent 1 to $l_c - 1$ layers, and is about to send from the current layer $l_c$ to the expected streaming layer $L$.

We use the mean square error $D_{mse}$ to determine the expected reconstruction error per pixel after receiving $L$ layers:

$$E\{D_{mse}(l_c, L)\} = D_c(l_c) - E\{D_c(l_c, L)\}.$$  \hspace{1cm} (3.1)

In (3.1), we see that $E\{D_{mse}(l_c, L)\}$ comprises the accumulated contributions $D_c(l_c)$ from the lowest layer $l = 1$ to the current one $l_c$, and the expected distortion decrement $D_e(l_c, L)$ when receiving the next $L - l_c + 1$ layers.

![Diagram](image)

Figure 3.4: Accumulated contributions up to the current streaming layer $l_c$

The first term $D_c(l_c)$ indicates the accumulated contributions of the sent layers (Fig. 3.4). This is equivalent to the reconstruction error up to the current layer

$$D_c(l_c) = D_s(0) - \sum_{i=1}^{l_c-1} D_s(i),$$ \hspace{1cm} (3.2)
where $D_s(0)$ denotes the distortion if none of the layers is decoded.

$$E\{D_e(l_c, L)\} = \sum_{j = l_c}^{L} D_s(j)P(j) = \sum_{j = l_c}^{L} D_s(j)(1 - \varepsilon(L))^{j-l_c+1}, \quad (3.3)$$

where $P(j)$ is the probability of receiving the $j$-th layer and $\varepsilon(L)$ is the packet loss rate.

Here we model the packet loss rate $\varepsilon$ as the sum of the random packet loss rate $\varepsilon_0$ and $\varepsilon_c$ which is caused by server overshtooting. Further, we assume that $\varepsilon_c(L)$ is proportional to the difference between the current bandwidth $W_c$ and the streaming output rate $R_c$ at each transmission opportunity

$$\varepsilon(L) = \varepsilon_0 + \mu \cdot (R_c - W_c)$$
$$= \varepsilon_0 + \mu \cdot \left( \frac{N_p \cdot (L - l_c + 1)}{T} - W_c \right), \quad W_c \leq R_c, \quad (3.4)$$

where $\mu$ is a non-negative factor. Note that we set $\varepsilon_c(L) = 0$ if $W_c > R_c$.

Moreover, we use feedback messages as a trigger for the retransmission of lost packets. We check the latest received feedbacks at each transmission opportunity. We do not retransmit any packet without knowing its feedback. If the feedback indicates any lost packet, we integrate its distortion decrement $D_s(l_{nak})$ in (3.3) and consider it as an extra layer.
3.3 Optimal Streaming Layer

3.3.1 Cost function

With above layer-distortion model, we are able to find the optimal streaming layer for each transmission opportunity by minimizing the distortion function in (3.1). As the accumulated contributions $D_c(l_c)$ are constant at each transmission opportunity, we essentially need to find the optimal layer $L$ to maximize the term $E\{D_e(l_c, L)\}$. We define it as our cost function

$$J(l_c, L) = E\{D_e(l_c, L)\}. \quad (3.5)$$

![Figure 3.6: Relationship between layer and expected distortion decrement $D_e(l_c, L)$.](image)

3.3.2 Properties of cost function

Now we study the cost function. The accumulated distortion $\sum_{j=l_c}^{L} D_s(j)$ is generally concave due to the non-negativity of the term $D_s(j) \geq 0$. $D_s(j)$ is monotonically decreasing as shown in Fig. 3.2(b). The term $(1 - \varepsilon(L))$ is also a monotonically decreasing function of layer $L$ due to the packet loss by overshooting. As the probability term $(1 - \varepsilon_c(L))^{j-l_c+1}$ is non-negative and varying between 0 and 1, it will not affect the concavity of the accumulated distortion. In general, it has a concave shape as shown in Fig. 3.6. Generally, it has been shown that it is a concave function when $\sum_{j=l_c}^{L} D_s(j)$ is concave and above probability term is a monotonically decreasing function of the number of lost packets [29] [30].

For a concave function, the optimal streaming layer $L_{\text{opt}}$ can be obtained
by maximizing the cost function

\[
\max_L J(l_c, L) \\
\text{s.t. } L \leq l_M,
\]

where \( l_M \) is the maximum encoded layer in the set of quality layers \( \Lambda \).

### 3.3.3 Fast algorithm

To solve this concave problem, we use a steepest descent search algorithm as illustrated in Fig. 3.7. We use the slope \( \lambda(l) \) of each layer \( l \) to measure the steepness and apply bisection search to find the extreme. With this method, we can achieve a fast algorithm with low computational complexity to find the optimum. It requires logarithmic complexity \( O(\log_2 \Delta_l) \), where \( \Delta_l = |l_M - l_c| \). Note, if the maximum encoded layer \( l_M \) is on the left side of the extreme, the cost function is monotonically increasing. In this case, we set \( L_{opt} = l_M \).

---

1. Initialize: Define \( l_1^e = l_c \) and \( l_2^e = l_M \) as two endpoints, \( l_{\text{mid}} = \frac{l_c + l_M}{2} \) as the middle-point, start range \([l_1^e, l_2^e]\);
2. Compute the slope of the middle-point \( \lambda(l_{\text{mid}}) \);
3. While \( |l_2^e - l_1^e| \geq 2 \) or \( \lambda(l_{\text{mid}}) \neq 0 \)
   1. If \( \lambda(l_{\text{mid}}) > 0 \), set \( l_1^e = l_{\text{mid}} \);
   2. If \( \lambda(l_{\text{mid}}) < 0 \), set \( l_2^e = l_{\text{mid}} \);
   3. Create new range by \([l_1^e, l_2^e]\);
   4. Update the middle-point and compute the slope of the middle-point;
4. Optimal layer \( L_{opt} = l_{\text{mid}} \).

Figure 3.7: Algorithm for finding the optimal layer \( L_{opt} \).
Chapter 4

Experimental Results

In this chapter, we illustrate the testing environment, i.e. video source and channel parameters. To have a comparison, a reference scheme is introduced briefly in Sec. 4.2. It is a rate-distortion optimized streaming scheme. In Sec. 4.3, we show the performance of both schemes in these four areas: instant output rate, actual packet loss rate, reconstruction quality over packet loss and computational complexity.

4.1 Experimental Setup

To avoid being limited by a particular video sequence and to provide robust test results independent of a video sequence, we evaluate our layer-optimized streaming method over 10 video test sequences, including foreman, carphone, news, mother-daughter, big-buck-bunny, container, hall, pairs, highway, and silent. To simulate the client randomly switching video content, we concatenate 10 videos into a long sequence for testing. Therefore, in total 2400 frames (240 frames from each test sequence) are used for simulation with a resolution of $352 \times 288$ (CIF) at 25 fps. We set the GOP size at 8 frames with a code block size of $16 \times 16$.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test video sequences</td>
<td>Long concatenated video</td>
</tr>
<tr>
<td>Frame rate</td>
<td>25 fps</td>
</tr>
<tr>
<td>Resolution</td>
<td>$352 \times 288$ (CIF)</td>
</tr>
<tr>
<td>GOP size</td>
<td>8 frames</td>
</tr>
<tr>
<td>GOP duration</td>
<td>$T_g = 320$ ms</td>
</tr>
<tr>
<td>Total frames</td>
<td>2400 frames</td>
</tr>
<tr>
<td>Test video duration</td>
<td>96 seconds</td>
</tr>
</tbody>
</table>

Table 4.1: Test video sequences setup

For the network, we set the parameter $\mu = 0.005$ in (3.4) such that an
overshoot of 200 kpbs will lead to a packet loss rate $\varepsilon = 1$. We assume that forward and backward channels are symmetric. A Gamma distribution is used to model forward and backward delays. Therefore, the round trip delay is also Gamma distributed. The other parameters are listed in the table below. Any packet that is received later than its decoding deadline will simply be discarded.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forward trip delay</td>
<td>$\Gamma(\kappa, \mu)$</td>
</tr>
<tr>
<td>Backward trip delay</td>
<td>$\Gamma(\kappa, \mu)$</td>
</tr>
<tr>
<td>Round trip delay</td>
<td>$\Gamma(2\kappa, \mu)$</td>
</tr>
<tr>
<td>Random packet loss rate</td>
<td>$\varepsilon_0$</td>
</tr>
<tr>
<td>Packet loss by overshooting</td>
<td>$\mu \cdot (R_c - W_c)$</td>
</tr>
</tbody>
</table>

Table 4.2: Network setup

The streaming server allocates $N$ transmission opportunities to each GOP. Since each GOP duration is $T_g$, the transmission interval is then $T_g/N$. We set the decoding deadline (or decoding time stamp) of each GOP as $T_d$. Packets that are received after the decoding deadline will be discarded.

<table>
<thead>
<tr>
<th>Variables</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmission opportunity</td>
<td>$N = 8$</td>
</tr>
<tr>
<td>Transmission interval</td>
<td>$T_g/N$</td>
</tr>
<tr>
<td>Decoding deadline</td>
<td>$T_d$</td>
</tr>
</tbody>
</table>

Table 4.3: Streaming server setup

4.2 Reference Scheme

A general solution to the rate-distortion optimal problem for various scenarios has been proposed in [31]. It uses a statistical model to measure the possible packet loss and network delays. The algorithm to find an optimal streaming policy can apply to packets with arbitrary decoding dependencies and results show that its performance is very close to the operational distortion-rate function.

4.2.1 Cost function

In this section, we introduce a rate-distortion optimized streaming scheme that can be generally applied to video packets regardless of packet dependency. Although this model can handle a variety of scenarios, we do not enumerate all the cases. In order to be comparable with our proposed scheme, the algorithm below is specifically derived for sequentially dependent video
Layer-optimized Streaming of
Motion-Compensated Orthogonal Video

packets. Moreover, sender-driven streaming with feedback over a best-effort
network scenario is discussed here. Only retransmission is taken into ac-
count to compensate random packet loss. [11] argues that the problem of
rate-distortion streaming of an entire presentation can be reduced to the
problem of error-cost optimized transmission of an isolated packet. A fast
practical algorithm is proposed for a single packet. The result is then used by
a general purpose iterative descent algorithm for locally optimal streaming
of a group of packets.

Streaming a single packet

First, we show how to compute the optimal streaming policy for a single
video packet. Every single packet is given a set of transmission opportunities
Φ = {t_i, i = 1, ..., N} equally. T = |t_{i+1} - t_i| is the constant duration
between two successive transmission opportunities. At each transmission
opportunity, the packet can either be sent or not, so the candidate policy
for one packet is 2^N. Taking into account the packet loss and delay, a Markov
decision process is used to calculate the error probability (ε_π) and expected
cost (ρ_π) of each policy. Here, the cost represents how many times a pa-
cket has been transmitted. For a given multiplier λ', dynamic programming [31]
or a branch and bound algorithm [32] (denoted as X1 algorithm below) is
used to compute the optimal choice π to minimize the expected Lagrangian.

\[ J_π = ε_π + λ'ρ_π \] (4.1)

Streaming a group of video packets

Then, we show how to utilize the above algorithm to solve the problem of
streaming a group of video packets. Suppose a group of L packets with
sequential dependency, π = {π_1, ..., π_L} is the transmission policy for all
packets in the group, whereas π_l is the policy for packet l. Then the total
expected distortion and cost can be written as

\[ D(π) = D_0 - \sum_{l=1}^{L} D_l \prod_{k=1}^{l} (1 - ε(π_k)), \] (4.2)

\[ R(π) = \sum_{l=1}^{L} B_l ρ(π_l), \] (4.3)

where \( D_l \) denotes the distortion decrement if packet \( l \) is decoded on time,
\( D_0 \) denotes the reconstruction error if no layer is decoded, \( B_l \) represen-
t the packet size.

Then the cost function of the group of packets can be written as

\[ J_π = D_0 - \sum_{l=1}^{L} [D_l \prod_{k=1}^{l} (1 - ε(π_k)) - λB_l ρ(π_l)]. \] (4.4)
4.2.2 Iterative sensitivity adjustment algorithm

Finding an optimal solution to (4.4) is shown to be NP-hard. To solve this problem, an iterative descent algorithm defined as Iterative Sensitivity Adjustment (ISA) (equivalent to Sensitivity Adaption (SA) in [31]) is proposed. The key is to minimize the objective function $J(\pi_1, \ldots, \pi_L)$ one variable at a time while keeping the others constant. The iteration is stopped upon convergence. The sufficiency of the X1 algorithm and its use with the ISA is proved to produce a near-optimal performance [11].

4.2.3 Smooth rate control

The ISA solved the optimization problem for a given $\lambda$. The rate control mechanism is to adjust $\lambda$ through bisection search until a desirable instant rate is achieved. We do the rate control at each transmission opportunity by rerunning the ISA. The sending history has to be taken into account to maintain the global rate-distortion optimization.

4.3 Performance Comparison

In this section, we compare the performance of both schemes in above mentioned four categories. The first comparison is the fitness to bandwidth variation. Since the bandwidth adaptation capability affects packet loss, we show the actual packet loss rate in the second experiment. The third experiment compares the reconstruction quality over packet loss rate, which is an overall quality comparison of both schemes. In the last experiment, the computational complexity is assessed.

4.3.1 Instant output rate

In this first experiment, we test the instant output rate of both schemes. We set the packet length to 128 Bytes and the network bandwidth fluctuates around 500 Kbps. The network random packet loss rate $\epsilon_0 = 10\%$. The forward and backward trip delay is modeled by a Gamma distribution with average delay $T$ and variance $T^2/2$. The decoding deadline $T_d$ is $2 \times T_G$ and the result is shown in Fig. 4.1.

We see that our proposed scheme adapts to the network bandwidth better. The reference scheme sometimes overshoots or under-uses the bandwidth. This is caused by the ISA. In some cases, it can not find the optimal transmission policy. This leads to either overshooting or under-use. Consequences are extra packet loss and under-use of available bandwidth.
4.3.2 Actual packet loss rate

In this second experiment, we compare the actual packet loss rate of both schemes. From the previous experiment, we know that the reference scheme does not guarantee efficient use of network bandwidth. In our model, the probability of packet loss is higher if the sending rate overshoots the available bandwidth. We can see the consequences of overshooting by comparing the actual packet loss rate, as shown in Fig. 4.3. The parameters for testing are: packet length - 512 Bytes, network bandwidth - from 100 to 500 Kbps, Gamma distribution - average delay $T$ and variance $T^2/2$, decoding deadline $- T_d = 2 \times T_G$. The random packet loss rate varies from 5% to 30%.

The results show that for our proposed scheme, the actual packet loss rate $\epsilon$ is equal to the random packet loss rate $\epsilon_0$. That means, the proposed scheme does not incur extra packet loss rate. For the reference scheme, the ISA sometimes fails to find an optimal transmission policy. This leads to a significant additional packet loss rate by overshooting.

4.3.3 PSNR over packet loss

This experiment compares the overall performance. We use PSNR to evaluate the reconstruction video quality. We test the reconstruction quality for several packet loss rates and channel bandwidths. For rigorous testing,
we change the parameters of the Gamma distribution and set different decoding deadlines. As mentioned before, the assumption is that the forward and backward channels are symmetric and that they have the same Gamma parameters. By setting different parameters $\kappa$ and $\nu$, we can obtain different average delays and variances of the round trip. We set the decoding deadline as an integer multiple of the GOP duration such that $T_d = n \frac{F}{n}$, where $F = 25$ fps is the frame rate and $n$ the integer. The performance is measured by reconstruction quality over random packet loss rate for different conditions of bandwidths (we use constant bandwidth for this test).

As shown in Figs. 4.4 and 4.5, our proposed scheme and the reference scheme have a similar performance at high bandwidth (300 – 700 kbps) as both use a rate-distortion framework. Our proposed scheme outperforms the reference scheme when the bandwidth is low (100 kbps). The main reason is that the ISA can not guarantee to find the optimal solution for all transmission opportunities. This leads to more overshooting at low bandwidth. Therefore, extra packet losses may happen which result in additional degradations. On the other hand, when the bandwidth is sufficiently high (300 – 700 kbps) and the random packet loss rate is low, our performance is slightly better than that of the reference. However, when the packet loss rate becomes higher (up to 30%), the performance of the reference scheme is slightly better than with our scheme. This is due to the fact that the reference implementation allows more retransmission policies than ours, which is more favorable when the packet loss rate is very high. However, with increasing number of transmission opportunities, we can improve the performance of our implementation, which will be addressed in the next experiment.

4.3.4 Computational complexity

In the last experiment, we evaluate the computational complexity of both schemes. For our proposed scheme, as stated in Sec. 3.3, the complexity is $O(\log_2 \Delta_l)$ for each transmission opportunity. If we have $N$ transmission opportunities, the complexity increases linearly. Hence, the complexity for one GOP is $O(N \log_2 \Delta_l)$. For the reference scheme, as discussed in [32], the computational complexity is highly dependent on the number of transmission opportunities. Using dynamic programming [31] for a single packet, it requires $O(N 2^N l_M)$ operations. The ISA needs at least $O(N 2^N l_M)$ operations to find the optimal solution for a single transmission opportunity. On the other hand, if applying a branch and bound (B&B) algorithm [32], the complexity for a single packet can be reduced to $O(N)$. However, for a group of packets, it is much slower than the ISA.

Additionally, as stated in [31], increasing the number of transmission opportunities $N$ will improve the overall performance. As shown in Fig. 4.2, the performance of our scheme is improved by increasing the number of transmission opportunities $N$ to 40. It outperforms the reference scheme.
at both low and high packet loss rates. For our scheme, the complexity increases linearly with $N$. However, increasing the transmission opportunities to $N=40$ is not practical for the reference scheme, as the computational complexity increases significantly.

![Figure 4.2: Effect of transmission opportunities](image-url)
Figure 4.3: Actual packet loss rate for various channel bandwidths
Layer-optimized Streaming of Motion-Compensated Orthogonal Video

(a) average delay = $T$, variance = $T^2/2$, deadline = 640 ms

(b) average delay = $T$, variance = $T^2$, deadline = 640 ms

Figure 4.4: Performance with decoding deadline at 640 ms
Layer-optimized Streaming of
Motion-Compensated Orthogonal Video

Figure 4.5: Performance with decoding deadline at 1280 ms
Chapter 5

Conclusions

In this report, layer-optimized video streaming for orthogonal video is proposed. We use motion-compensated orthogonal transforms to encode the input video into multiple quality layers with sequential decoding dependency. With that, we construct a layer-distortion model and derive a cost function based on the trade-off between expected streaming layer and expected distortion. Due to the sequential decoding dependencies among the layers, the cost function is concave. Therefore, we develop a fast algorithm to find the optimal transmission policy at low computational complexity. The experimental results show that our layer-optimized streaming outperforms the Iterative Sensitivity Adjustment algorithm in terms of reconstruction quality and computational complexity.
Bibliography


