National Chiao Tung University

EECS International Graduate Program

Thesis

使用視訊品質估計與LMRC通道編碼之視訊串流最佳化研究

Optimization of Video Streaming Using Video Quality Estimation and LMRC Channel Coding

Student: Philip Tovstogan

Advisor: Prof. Hsu-Feng Hsiao

October 2015

使用視訊品質估計與LMRC通道編碼之視訊串流最佳化研究

Optimization of Video Streaming Using Video Quality Estimation and LMRC Channel Coding

研 究 生: 杜斐利　　Student: Philip Tovstogan

指導教授: 蕭旭峰　　Advisor: Prof. Hsu-Feng Hsiao

國 立 交 通 大 學

電機資訊國際學位學程

碩 士 論 文

A Thesis

EECS International Graduate Program

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Electrical Engineering and Computer Science

October 2015

Hsinchu, Taiwan, Republic of China

中華民國一百零四年六月

使用視訊品質估計與LMRC通道編碼之視訊串流最佳化研究

學生:杜斐利　　指導教授:蕭旭峰

國立交通大學電機資訊國際學位學程

摘要

在易發生錯誤的通道環境上傳輸視訊串流，會因為封包的遺失而影響到接收者在觀賞時的影片品質，為了處理不穩定通道上的封包遺失問題，通常可透過回饋資訊或是重傳封包的方式來解決。但這種方法並不適用於串流環境之下。另一種保護方式即是通道編碼，透過增加冗餘的編碼資料來確保接收者在封包遺失的狀況下，仍有機會可以還原所有的資料。在噴泉碼中，Luby Transform 編碼是很頻繁被使用的通道編碼技術，因此我們將會把重點放在這類編碼上面。

高效率視訊編碼(HEVC或是H.265)是目前最新的視訊編碼標準，而其中的視訊資料擁有不對等重要性的特性，參考畫格(I)假如發生錯誤，錯誤的效應將會傳播到多數後面的畫格，反之假如錯誤發生在非參考的畫格(P,B)上，錯誤的效應只會對附近的畫格造成影響。

透過不對等資料保護，相比於較為不重的資料位元，我們盡可能減少在較為重要的資料位元上的封包遺失，進而提升傳輸影片的畫面品質，而對於兩者之間的權衡，是一個具有挑戰性的優化問題。

在這篇論文中，我們將會把焦點放在分層多權重無比例編碼-這項結合傳統Luby Transform 編碼以及不對等資料保護並且擁有較佳表現的編碼方式上面。因此，我們設計了一個系統，可以根據影片的內容以及通道狀況，動態調整並選擇合適的編碼參數。為了達成這個目的，我們發展了一個數學模型，可以用來估計預期影片的品質下降量並且在準確度上有很好的表現，透過最小化品質的下降量，我們將可以計算出合適的編碼參數給設計的系統使用，達到提升影片品質的目的。

Optimization of Video Streaming Using Video Quality Estimation and LMRC Channel Coding

Student: Philip Tovstogan Advisor: Prof. Hsu-Feng Hsiao

Department of Electrical Engineering and Computer Science

National Chiao Tung University

Abstract

Video streaming over error-prone channels is a subject to a packet loss, which impacts video quality on a receiver side. Ways of dealing with packet loss include feedback and retransmission of lost packets, what is not applicable for streaming environment that may have no feedback channel (cellular networks). Another way to protect data is **channel coding**, which adds some redundant data to a stream to guarantee that receiver can decode all data, even if something is lost. Fountain codes, particularly **LT codes** are one of frequently used channel coding mechanisms, therefore we will focus on them.

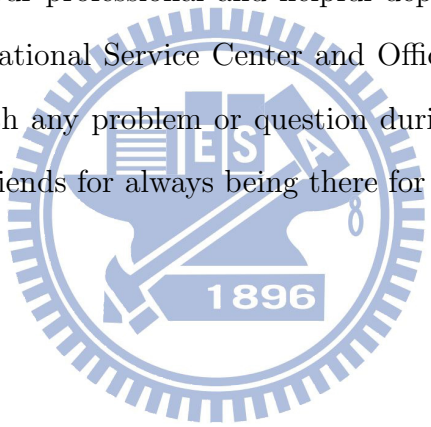
High efficiency video coding (HEVC, H.265) is latest standard in video coding, and it has property of unequal importance of the video data. Reference (I) frame errors will propagate over several frames forward, while errors in non-reference frames (P, B) will be mostly localized near that frame. By using **unequal error protection (UEP)** mechanism, packet loss can be further reduced for more important bits (MIB) at the expense of less important ones (LIB), thus increasing quality of transmitted video. The challenge of choosing optimal trade-off between MIB and LIB layers is an optimization problem with set of challenges on its own.

In this paper we focus on Layer-Aligned Multi-Priority Rateless Codes (LMRC) - channel coding scheme that utilizes UEP and coding sections overlapping that yields better performance comparing to pure LT codes. As a result, we design a system that dynamically chooses optimal channel coding parameters depending on the video content and channel condition. For this purpose we develop a model that allows estimation of expected video degradation of a transmitted video with good accuracy. By minimizing degradation we are able to calculate optimal parameters for a system.

Acknowledgement

First and foremost I would like to express huge gratitude to my supervisor prof. Hsu-Feng Hsiao for being an amazing teacher and a person. He was always helpful and resourceful, answered all the questions that I had and gave directions when I was lost. I would also like to thank all my colleagues, especially Kuan-Ju Tseng, Lian-En Hong and Chin-Han Chen from Network and Information Processing (NAIP) Lab for providing a great working environment and giving insights for problem solving, and helping me out during the times when I was stuck.

I also want to thank our professional and helpful department staff, especially Angel Yu and Aileen Yu, International Service Center and Office of International Affairs staff for always helping me with any problem or question during my master studies. And of course thanks to all my friends for always being there for me.



Contents

List of Figures

List of Tables

Chapter 1

Introduction

HEVC standard [1] is the latest video standard adopted in 2013, successor to H.264. Video streaming is slowly transitioning to this new standard, but the question of protecting data sent over the error-prone networks remains the same.

Let us consider the flow of video streaming scenario. We have raw video being generated by camera or screen capture that is encoded into HEVC stream or using any other codec. For the scope of this paper we are only discussing HEVC, but some concepts described in this paper can be applied to other codecs.

So compressed video stream is supposed to be sent over the network. It consists of Network Abstraction Layer (NAL) units which usually contain either one video frame or video slice worth of information; or some additional information needed to successfully decode video. Size of NAL packets can vary significantly, but eventually all of them are split into transport-level packets before being sent over the network. So if there is some event in the network that can lead to loss of the packets, it is going to happen on the transport level. Size of packets is usually dictated by network's MTU (maximum transmission unit). For example, in Ethernet networks MTU is equal to 1500 bytes, while for WLAN networks it can be as large as 8KB.

If we transmit video straight after encoding, some packets will be lost what will lead to artifacts on the receiver side after decoding and error concealment that is usually part of decoder. TCP layer has retransmission mechanism, but timing might already be too late if this packet was part of the frame that was already shown on the receiver side; if client will

actually wait for retransmitted packets, waiting for retransmission will introduce pauses and stuttering in the video playback reducing stream quality.

Usually various channel coding techniques are applied to video before it being sent over the network. The main goal is to add redundant data to raise probability of successfully decoding video stream on the client side with minimal overhead.

One of the most popular and efficient channel codes are **Reed-Solomon Codes** [2], but they have drawback of having complex encoding and decoding algorithms what makes them not very suitable for video streaming scenario. Another set of codes is called **Fountain Codes** or **Rateless Erasure codes** [3]. First practical implementation of them are **LT codes** [4]. Subsequently, **Raptor codes** [5] and **Online codes** [6] were also introduced. Currently channel coding area is rich with multiple versions of many coding schemes and new papers are continuously being published.

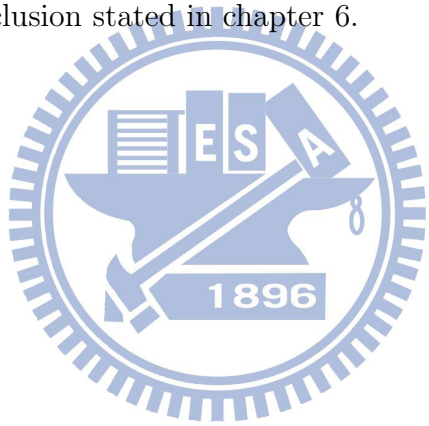
Most of fountain codes provide equal error protection (EEP) to all of the data. However video data has different importance depending on the role of the frame. In HEVC standard (including preceding versions) there are reference frames - frames that are encoded using internal spatial redundancy, and frames that are referencing other ones (taking advantage of temporal redundancy with other frames). If reference frame is lost, then all frames that are referencing it will have artifacts present; however if frame that has no references to it is lost - artifacts will only be present in that particular frame. So naturally reference frames need to be protected stronger even at possible expense of other frames. This concept is called unequal error protection (UEP) and was introduced to LT codes by several papers. Most notable of them include **Expanding Window Fountain Codes** [7], **Sliding Window Raptor codes** [8] and others [9], [10], [11], [12], [13].

This paper is continuation of work on class of rateless erasure codes called **Layer-Aligned Multi-Priority Rateless Codes** [14]. Original paper had introduced this class of codes with their practical applications to SVC extension of H.264 standard [15]. Moreover, accurate prediction model to calculate loss probability of individual coded packets belonging to each importance layer was proposed.

In this paper we will introduce a system that will allow to choose optimal channel

coding parameters to maximize transmitted video quality. In contrast to [14] we are focusing on HEVC standard. Two major contributions of this paper include model that can be used to estimate quality degradation of HEVC video after transmission through the error-prone channel and system that uses above-mentioned model to carefully choose set of parameters for channel coding scheme to minimize quality degradation of transmitted video.

The rest of the paper is organized in the following structure: in chapter 2 we present background for the research, go through related work from literature in chapter 3, present our method in chapter 4, and in chapter 5 we provide experimental results and comparison to other models with conclusion stated in chapter 6.



Chapter 2

Background

2.1 LT Codes

LT codes [4] were introduced as channel coding method that can generate potentially limitless number of coded symbols with relatively simple encoding and decoding process. Imagine you have section of data that you want to encode. First it is split into N message symbols/blocks (MS/MB) of same size. Then number of coded symbols/blocks (CS/CB) can be generated from set of MBs according to encoding algorithm:

1. Pick degree d according to degree distribution function $\Omega(d)$
2. Uniformly choose d message symbols to be encoded
3. Apply XOR operation to all of them and you will get data of coded symbol
4. Repeat previous steps until you have generated N' coded symbols

Most important things that affect performance are degree distribution function $\Omega(d)$ and amount of coded symbols N'. Overhead ϵ is defined the following way:

$$\epsilon = \frac{N'}{N} - 1$$

Easiest decoding algorithm that can run in linear time is called belief-propagation (BP) decoding that is described below:

1. After receiving a coding block: if it's degree is not 1, put it into buffer

2. If it's degree is 1 we have just recovered one message symbol. Go through all coded symbols in the buffer, if any of those has this message symbol inside, XOR it with MB effectively removing it from CB data. If any of CBs from buffer were reduced to degree 1, treat them as newly received symbol and repeat all of the steps with it being a received MB. When there are no more CBs that can have their degree reduced in this way, put original MB to Message Queuing Area (MQA)

3. If all messages have been recovered, finish the decoding process. If there are no more input symbols and there are still unrecovered message symbols, report failure.

Decoder needs to know which MBs are encoded in each CB, so this information should either be passed via the network, or pseudo-random number generators should be synced on both receiver and transmitter sides.

There are several papers dedicated to the search for an optimal degree distribution. In the original paper [4] **Ideal Soliton** and **Robust Soliton** distributions were proposed. Ideal soliton distribution is described by the following probability mass function:

$$p(k) = \begin{cases} \frac{1}{N} & \text{if } k = 1 \\ \frac{1}{k(k-1)} & \text{if } k = 2 \dots N \end{cases} \tag{2.1}$$

For robust soliton distribution extra set of parameters are introduced. Let δ be allowable loss probability for message symbol and M to be location of another spike in the probability distribution. Define $R = N/M$.

$$t(i) = \begin{cases} \frac{1}{iM} & \text{if } i = 1 \dots M-1 \\ \frac{\ln R/\delta}{M} & \text{if } i = M \\ 0 & \text{if } i = M+1 \dots N \end{cases} \tag{2.2}$$

$t(i)$ is summed with $p(k)$ and then re-normalised to produce robust soliton distribution.

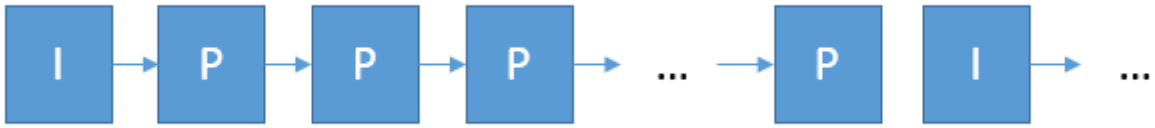


Figure 2.1: IPP..P GOP structure

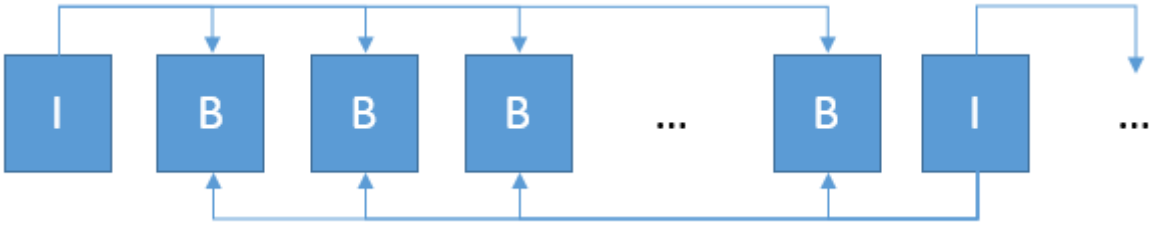


Figure 2.2: IBB..B GOP structure

As number of message symbols in one section will increase, required overhead that will yield same MB loss probability will decrease. Thus, higher amount of message symbols gives better performance.

2.2 HEVC standard

HEVC is a video standard, it was ratified on April 2013 and is essentially a successor to H.264. Each frame is labeled as either I, P or B frame according to GOP (Group of Pictures) structure. I frames are intra-coded, in other words, only spatial redundancy is taken advantage of, no other frames are referenced. P or B frames usually reference other frames, so because of temporal redundancy being used for coding, their size is smaller, however any errors in the reference frames will propagate over all P an B frames referencing them. P stands for predictive frame and B for bi-predictive frame, and that what their roles were usually in H.264 standard. However with introduction of HEVC these restrictions are relaxed and you can find P frames referencing more than one frame.

In HEVC coding referencing pattern of frames is defined by GOP structure. One of simplest GOP structures is IPP..P. First frame of GOP is labeled I and all other frames are P frames, each one referencing previous one (see fig. 2.1). In case of any of P frames is corrupted, artifacts will propagate until the end of GOP.

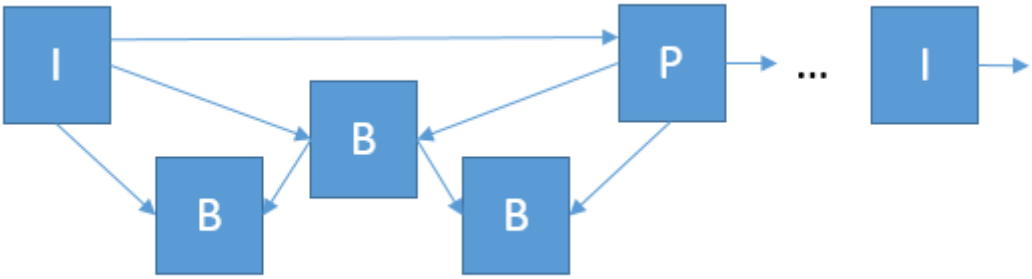


Figure 2.3: H-B GOP structure of size 4

Another simple GOP structure is IBB..B. It works exactly like IPP..P, however frames following I frame are B, and each of them is referencing I frame in the beginning of this and next GOP (see fig. 2.2). It has smaller bitrate because of bi-prediction, also if any of B frames is lost, it doesn't affect other frames. If any of I frames is corrupted, it will lead to artifacts or undecodability of all neighboring B frames both from left and right.

The most common structure used in HEVC is called **Hierarchical-B** (fig. 2.3). First, I frame is not inserted in the beginning of each GOP, but only once every N_{ref} frames. This period is also called intra-refresh period. Instead, in the beginning of each GOP (except first) there is a P frame referencing starting frame of previous GOP. Inside of the GOP, which size is restricted to be an integer that is a power of 2 (2, 4, 8, 16, etc.) there are B frames that are referencing each other resembling hierarchical structure. For example, for GOP size of 4, 2nd B frame is referencing current and next GOPs starting P frame; 1st B - current starting and middle (2nd) B; and 3rd B - middle (2nd) B and starting P of next GOP. If GOP size is larger, hierarchical structure continues in similar way down. This structure combines strengths of both P and B frame structures in flexible and controllable way.

Though HEVC decoders can try to conceal errors if some part of NAL unit is corrupted or lost, it will usually introduce noticeable spatial distortion. Thus, for this paper we will focus on temporal distortion and use concealment strategy of displaying latest successfully decoded frame. In addition, temporal distortion usually deteriorates subjective quality less, because human eyes tend to notice spatial distortion more.

Chapter 3

Related Work

3.1 Layer-Aligned Multi-Priority Rateless Codes (LMRC)

These codes were introduced in [14] and two most important characteristics they possess is sliding window (SW) method and unequal error protection (UEP).

Let us briefly go over the encoding and decoding process to introduce major differences between LMRC and LT codes.

First, assume that data in the section is separated by importance into N layers, and layer sequence repeats each section (see fig. 3.1). After first section was encoded, window moves by the length of the 1st layer thus now including 1st layer of second section. Then

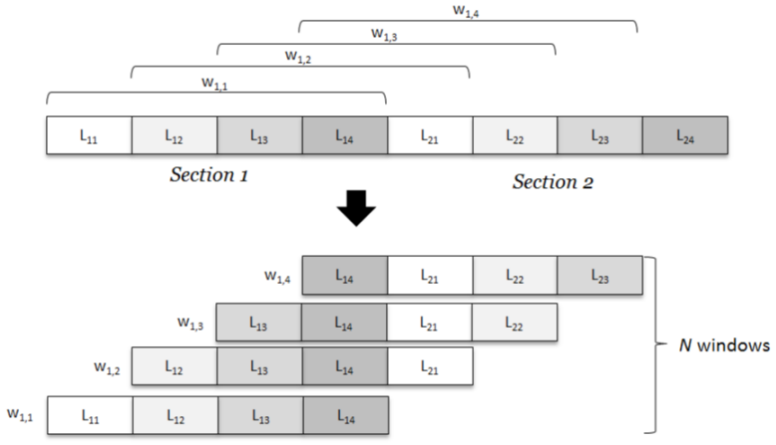


Figure 3.1: Sliding window (LMRC codes)

window moves by length of layer 2 and so forth. This method introduces possibility of cascade decoding, where packet from 1st layer of 1st section can indirectly help to decode packet from 3rd layer of 2nd section, in case they both are coded in some CB with message symbol from 1st layer of 2nd section. Another positive thing is concept of virtual section size - effective section size used for decoding is actually larger than size of one section.

UEP is achieved by slight modification of encoding algorithm. It is described below:

1. Pick degree d according to degree probability distribution $\Omega(d)$
2. Choose d MBs from N different layers. The probability of choosing a certain MB from video layer j is p_j, where $1 \leq j \leq N$. If there are n_j MBs from layer j in one section, then probabilities should be normalized: $\sum_{j=1}^{N} n_j p_j = 1$
3. The coded symbol is formed as result of XOR operation between the chosen message symbols

Another notion for defining importance of each layer includes weighting factors ω_j. In this case probability to pick each individual message symbol is calculated in the following way:

$$p_j = \frac{\omega_j}{\sum_{i=1}^{N} n_i \omega_i} \tag{3.1}$$

Decoding process is the same as for LT codes, the information about mapping of message symbols to coded symbols still needs to be passed. Layers with higher weight will be chosen as source for message symbols more often, thus increasing probability of their successful decoding.

Main contribution of [14] is analytical model that was derived using and-or trees, introduced in [16] by M. Luby et. al. Process of its derivation is similar to original one for simple LT codes, we will not go through it in this overview. For complete derivation see [14] and [16]. The final result is the iterative formula that can be used to calculate probabilities of message symbol loss for each layer:

$$p_{i,j} = e^{-\mu\gamma_\omega k p_j N(-1+q_i)} \tag{3.2}$$

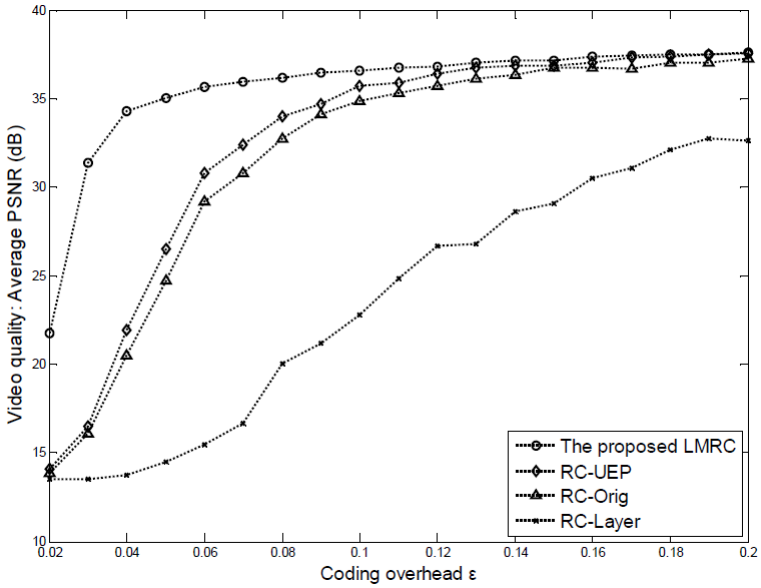


Figure 3.2: Performance of LMRC

$p_{i,j}$ is loss probability for message symbol from layer j in and-or tree with height $2i$, $\gamma_\omega = 1 + \epsilon$, k is number of message symbols in one section, p_j is the above-mentioned probability to pick individual message symbols from layer j, N is number of layers and q_i is probability for an and-node at level i to be evaluated as 0 and it can be calculated using following formula:

$$q_i = \sum_{d=0}^{k-1} (A_{d+1} (\sum_{\forall \boldsymbol{d}: \sum_{i=1}^{N} d_i = d} wdp(\boldsymbol{d}) \Psi_{\boldsymbol{d}}))$$

$$A_d = \frac{d\Omega_d}{\Omega'(1)}$$

$$\Psi_{\boldsymbol{d}} = 1 - \prod_{j=1}^{N} (1 - p_{i-1,j})^{d_j}$$

A_d is probability for an edge to connect to and-node of degree d, $wdp(\boldsymbol{d})$ is probability for a coded block with layer vector $\boldsymbol{d}$ to exist. In other words, $wdp(\boldsymbol{d})$ is probability to select d_i message blocks layer i for all $1 \leq i \leq N$. It can be calculated using the multivariate version of Wallenius non-central hyper-geometric distribution [17]. $\Psi_{\boldsymbol{d}}$ is a probability for an and-node with fixed vector $\boldsymbol{d}$ to be evaluated as 0.

Performance evaluation of LMRC (see fig. 3.2) is performed using video of small

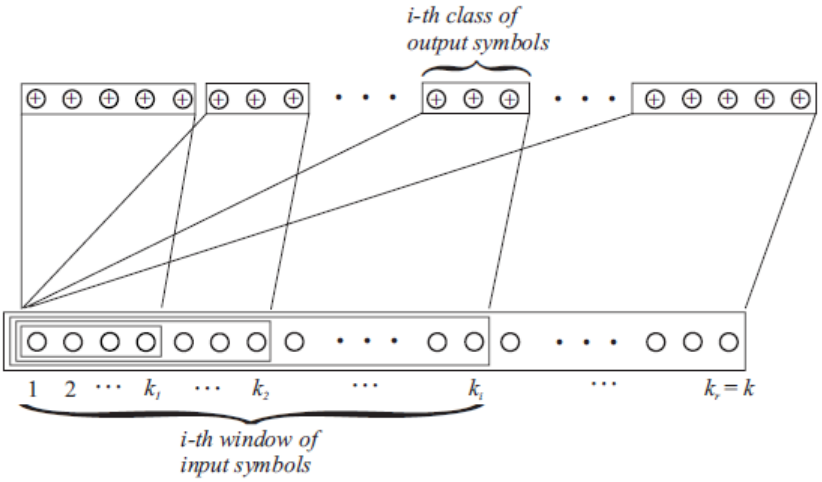


Figure 3.3: EWF codes encoding

resolution (325x288) what results in very small packet size that is not applicable to real world situations. Also choice of optimal weighting factor for MIB layer is performed using exhaustive search. In this paper we will use more appropriate high resolution HEVC-encoded videos and our method will provide faster search for a optimal weighting factor value.

3.2 Expanding Window Fountain codes

Expanding Window Fountain (EWF) codes [7] use expanding windows to cover the section with standard LT encoding performed inside windows. Before generating CB, one of expanding windows is selected, and all MBs in it are used for CB generation. Thus, MBs in the start of the section will be present in more CBs than MBs located in the end of the section.

Structure used in EWF codes has N layers, all of them start in the beginning of the section, all of MBs of layer i are also contained in layer $i+1$, and layer N covers the whole section (see fig. 3.3).

Also define probabilities to choose each layer: p_i, $\sum_{i=1}^{N} p_i = 1$. LT encoding process is altered in the following way: after we chose degree for CB, layer is chosen according to layer probability distribution. Then, all MBs from the layer are uniformly picked to be included in CB.

Main difference between LMRC and EWF codes is that in EWF encoding algorithm you make a decision about layer once every CB. For LMRC you decide on the layer each time you pick MB, so all CBs are more uniform in this regard when in each CBs from EWF codes all MBs are belonging to one of importance layers.

Moreover because of nature of expanding windows, layers in EWF codes are always placed from most important to least important order. In LMRC codes importance is assigned using weighting factors, so if needed, importance of first layer can be lower than second layer. This property makes LMRC codes more flexible.

Because of huge similarity of LMRC and EWF codes, and flexibility of our system we can replace LMRC module with EWF module and compare their performance in chapter 5.

3.3 Randomized Expanding Reed-Solomon (RE-RS) Codes

These codes are introduced in [18]. They have similar applications with this paper, however their approach uses modified version of Reed-Solomon codes to provide efficient encoding of video stream. We will use this paper for performance comparison in chapter 5.

Each GOP of the video is encoded in one section and there is no overlapping. Each frame is split into message symbols with constant size typically containing one video slice. Traditional approach of RS coding is to generate coded symbols for each frame individually and transmit them. Method proposed in this paper involves two key concepts: **randomization** and **expanding**.

Assume that there are K message symbols, $N - K$ coded symbols are going to be generated. Moreover, before encoding process starts, $2^m - 1 - N$ padding symbols filled with zeros are added to create full-length RS code (m is number of bits in the symbol). Overhead μ is calculated in the following way: $\mu = \frac{N-K}{K}$. Downside of encoding each frame individually is that number of MBs per frame is pretty low for code to have good

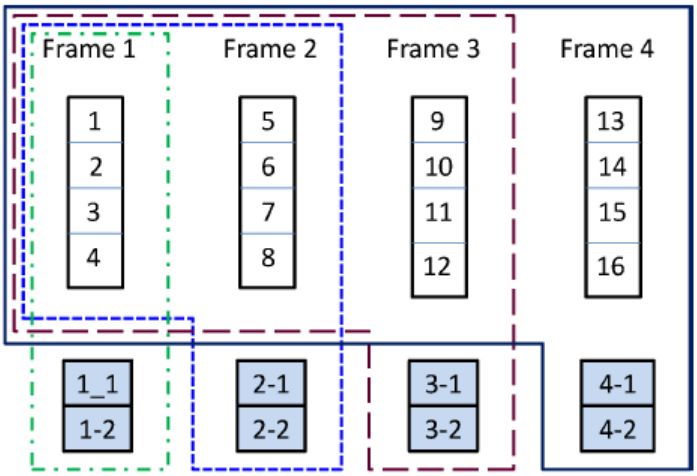


Figure 3.4: RE-RS scheme

performance, in addition CBs from one frame provide no information that can be used to decode other frames.

To keep overhead more or less constant over all frames, number of CBs to generate is calculated using following function ($S(k)$ is number of MBs in frame with index k):

$$R(i) = \begin{cases} \lceil \mu S(1) \rceil & \text{if } i = 1 \\ \lceil \mu \sum_{k=1}^{i} S(k) \rceil - \sum_{k=1}^{i} R(k) & \text{if } i > 1 \end{cases} \tag{3.3}$$

First key concept is to use MBs from previous frames for CB generation (see fig. 3.4). Thus, if there was error in one of earlier frames which is currently propagating, if we will receive enough CBs from next frames, we will be able to successfully decode that previous frame, and though it was already played back, we can stop error propagation by using updated pixel buffer values that is used in reference decoding.

To improve the performance of RS code, before encoding all MBs are shuffled with zero-padding MBs. It increases probability of having full-rank parity matrix at the time of decoding process. Still, mapping of the shuffle performed needs to be the same at both encoder and decoder sites, thus should be either transmitted over network or synchronised.

Full encoding process looks like this:

1. RS packets are allocated to each frame using equation 3.3

2. Video packets of current and previous frames are collected, padding packets are

generated if number of packets is less then $2^m - 1 - R(i)$

3. All $2^m - 1 - R(i)$ packets are randomly reordered.

4. $R(i)$ CBs are generated

5. Video packets of current frame and generated CBs are transmitted to receiver.

6. Repeat previous steps for all frames in GOP

Decoding process for RE-RS codes is described below:

1. After receiving packets for current frame they are reordered together with zero-padding MBs using the same map as at the encoder site

2. By multiplying reordered video packets with parity-check matrix, parity-check equations are generated

3. Parity-check equations of current frame are combined with equations from previous ones

4. If system of equations can be solved and it will allow successful decoding of previously lost frames, it is done and reference buffer is updated if necessary. If not, then current frame will be decoded and any artifacts if present, will be attempted to be concealed.

5. Repeat previous steps for the next frame until end of GOP

Regarding experimental results, optimal value of overhead μ and its relation with packet loss rate was found using exhaustive search ($\mu = 4p_{loss}$). For their simulations they used packet size of 400 bytes and $m = 10$.

One of advantages of RE-RS codes is that there is no section-based delay in streaming, i.e. section doesn't need to be completely encoded before being sent over the network. However encoding and decoding process time can grow near the end of the section because of large amount of MBs used and nature of RS encoding/decoding. Also low number of MBs in the beginning of GOP limits potential performance of RE-RS codes.

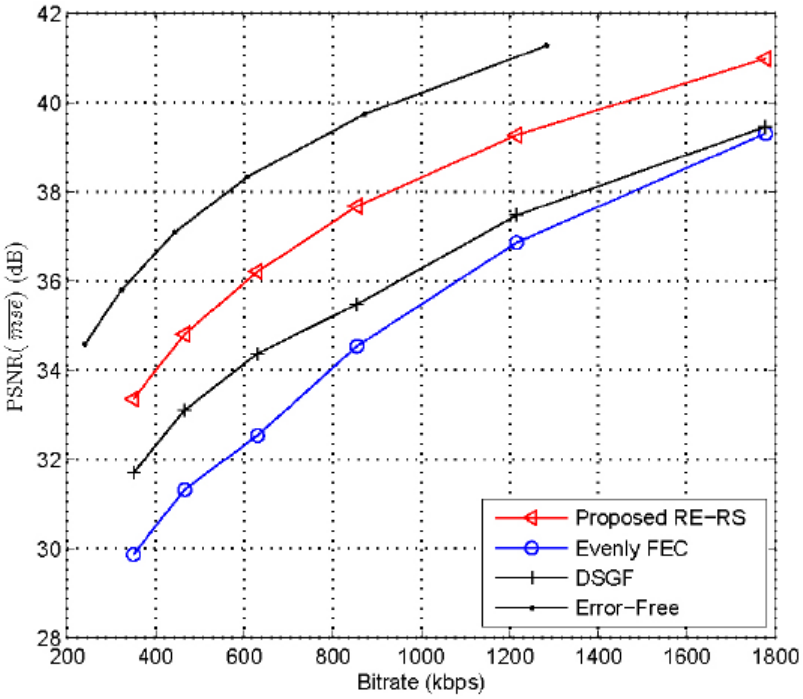


Figure 3.5: Performance of RE-RS codes

3.4 Video Distortion Estimation

In [19] a model to estimate end-to-end video distortion was proposed. It considers advanced error concealment strategy used by decoder and its accuracy is good enough for its purposes. However because it covers both encoding and error concealment process after transmission, it cannot be used in the scenario that we are considering with channel coding included. Also it doesn't take possible UEP into account, so that's another point why it is not applicable to our scenario.

Thus we need to develop model that will allow us to predict video degradation after it being channel coded, transmitted and its errors concealed. We are using PFC (previous frame copy) error concealment method.

Chapter 4

Theory

Our goal is to be able to estimate video quality degradation after it being encoded with LMRC, transmitted over the network, decoded and its errors been concealed.

The generated video stream is first split into video segments. Segment's size is determined by how often the video content significantly changes (e.g. scene change) and channel condition changes. Once per each video segment it's data is analysed to provide information to our system, so longer it is, less computations are required. But it shouldn't be too long, or the estimation accuracy of the system will suffer, if video content and channel condition varies too much. Typically, video segment is of the same size as channel coding section, however it can be any integer multiple of it.

We introduce 3 main modules of our system with different functionality and responsibility (see fig. 4.1).

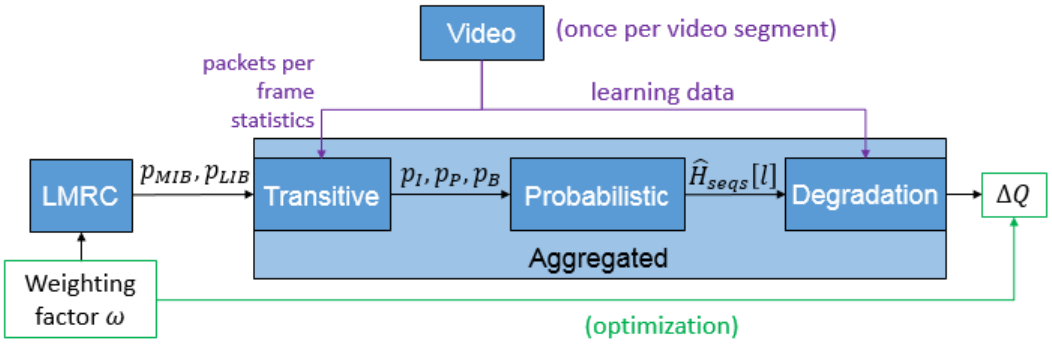


Figure 4.1: System Architecture

Transitive model uses video frame size statistics to provide means to calculate loss probability for each type of frame depending on packet loss rate after channel coding. Video data needs to be processed only once to gather statistics about frame size, and if it should not change much in the next segment, data from previous segment can be kept.

Probabilistic model is responsible for calculating estimated frequency of encountering sequences of undecodable frames depending on the GOP structure used and each loss probability for each frame type. This information is used by **degradation model** to estimate quality degradation of the decoded and error-concealed video.

All three models combined can be abstracted as **aggregated model** that analyses video segment and can estimate video degradation depending on weighting factor used in LMRC and packet loss rate (channel condition). Thus it is used to solve the optimization problem of choosing weighting factor that will minimize video quality degradation or in other words maximize received video quality.

So, the complete system flow includes splitting video into segments, packet loss rate estimation from the network, then usage of the aggregated model in conjunction with LMRC model to choose optimal weighting factor and finally applying LMRC channel coding to the video and transmitting it over the network. Ideally, for each video segment the whole process should be repeated, unless video content and channel condition doesn't change. Moreover, reduced complexity estimation (introduced in section 4.3.2) can be used to reduce amount of calculations done per segment without significant loss of performance.

4.1 Transitive Model

We will assume that if any packet from the frame is lost, then the whole frame is undecodable. This assumption will allow us to disregard any spatial distortions that may be present because of partial decoding of the frame, and focus on the temporal distortions. In other words we are considering worst-case scenario when the whole frame cannot be decoded.

Because packet loss occurs on the network layer, which is abstracted from the appli-

cation layer, and we are interested in frame loss rate, we need to devise a schema that will allow us to make this conversion with good accuracy. That is the goal of transitive model.

Packet loss rate used by transitive model is output of the LMRC model which uses coded packet loss rate as its input. So naturally we require loss probability of the coded packets first. In the systems without feedback channel (e.g. mobile networks) value for packet loss rate can be estimated from uplink channel (from device to base station) or using channel quality measure (CQI for WiMAX and LTE networks).

Let us assume that size of the frames in the encoded video is changing with more or less constant variance, so most of the frames of same type can be guaranteed to be less than certain size (i.e. all P frames are less than one size value, but of course it is different for I frames). This viability of this assumption is confirmed in the chapter 5.

Thus, the conversion is not difficult to perform – probability of successfully receiving the whole frame is equal to the probability of successfully receiving all of the packets. Then probability of loss can be calculated as follows:

$$p = 1 - (1 - p_{msg})^S$$

Where S is the maximum estimated size of the frame divided by network packet size, which means how many network packets are there in the frame. Here we are overestimating actual number of packets per frame, thus because actual probabilities will be less, we are using worst-case scenario for video-quality estimation.

Now we introduce frame loss probabilities for each different frame type:

$$p_I = 1 - (1 - p_{msg,I})^{S_I} \tag{4.1}$$

$$p_P = 1 - (1 - p_{msg,P})^{S_P} \tag{4.2}$$

$$p_B = 1 - (1 - p_{msg,B})^{S_B} \tag{4.3}$$

For example for the video sequence named Park Scene with resolution of 1080p after

analysing frame sizes, we get that I frame is less than 75KB on average, P - 3.0KB, B - 1.5KB. So, for the typical network packet size or MTU of 1.5KB, we can calculate the values of S_I, S_P and S_B:

$$S_I = 50, S_P = 2, S_B = 1$$

Depending of importance layer assignment to different frame types message symbol recovery probability will be different. So for example, if I frame packets are assigned to MIB layers, and others to LIB, then $p_{msg,I} = p_{MIB}$, $p_{msg,P} = p_{msg,B} = p_{LIB}$.

Due to nature of channel coding there will definitely some padding overhead (frame will occupy integer amount of network packets). We can calculate it assuming uniform distribution of each packet padding overhead.

$$\epsilon' = packet\_size/2 \times fps/bitrate \tag{4.4}$$

This overhead is inevitable for any LT-based channel codes and it should be accounted for in the experiments. Estimated value of padding using equation 4.4 overhead is very accurate comparing to actual value of overhead from channel encoded video (average relative error = 1.01E-3).

Packet loss rate that is used by transitive model should be obtained via channel coding model (in our case, LMRC) and thus loss probabilities of each frame type can be used by probabilistic model.

4.2 Probabilistic Model

4.2.1 Definitions

Let's define the binary indicator function I_{frames} that tells if a frame is decodable or not in the video stream segment with length N:

$$I_{frames}[n] = \begin{cases} 1 & \text{if frame } n \text{ is decodable} \\ 0 & \text{if frame } n \text{ is not decodable} \end{cases} \tag{4.5}$$

The frame may be undecodable due to loss of it's data when it was sent through a channel, or if it references another undecodable frame.

The video is divided into GOPs of equal and fixed length (denoted by G) and we will assume that video slice length N is multiple of G ($N_G = N/G$ is the number of GOPs in it). We will also transform function $I_{frames}[n]$ into function that takes GOP index n_G and frame index g within that GOP - $I_{GOP,frames}[n_G, g]$:

$$I_{GOP,frames}[n_G, g] = I_{frames}[Gn_G + g], 0 \leq n_G < N_G, 0 \leq g < G \tag{4.6}$$

$I_{GOP,frames}$ gives us information about sequences of 0's and their length. We will name sequence of consecutive 0's in I_{frames} a 0-seq for shortness of notation. 0-seq index identifies it in the video segment and starts with 1. Let's define $L_{seqs}[i]$ as a function that takes 0-seq index as input and outputs length of corresponding 0-seq, and $L_{GOP,seqs}[n_G, i]$ a function that takes GOP index and 0-seq index inside of GOP. Important thing is that n_G indicates the index of GOP, **where the sequence has ended**.

Notation-wise if we don't explicitly indicate value for index of the function, then we are interested in all values of it. For example $I_{GOP,frames}[0, 3] = 0$ is a scalar and indicates decodability of frame 3 in GOP 0. If we are using index g instead of its value in notation, it means that we are interested in all its values - $I_{GOP,frames}[0, g] = \{1100\}$ is an array of $I_{GOP,frames}[0, g]$ for $0 \leq g < G$. Similarly $I_{GOP,frames}[n_G, g] = \{\{1100\}, \{1110\}\}$ shows all values for $I_{GOP,frames}[n_G, g]$, $0 \leq n_G < N_G$, $0 \leq g < G$.

Now we will show how to calculate values for introduced functions on the example. Assume that we have GOP structure IPPP, $G = 4$, $N = 16$:

$$I_{frames}[n] = \{1100111000001100\}$$
$$I_{GOP,frames}[n_G, g] = \{\{1100\}, \{1110\}, \{0000\}, \{1100\}\}$$
$$L_{seqs}[i] = \{2, 5, 2\}$$
$$L_{GOP,seqs}[n_G, i] = \{\{2\}, \{\}, \{5\}, \{2\}\}$$

Now we count number of occurrences of each 0-seq length in each GOP using $L_{GOP,seqs}$ and construct new function $H_{GOP,seqs}$ by assigning to $H_{GOP,seqs}[n_G, l]$ the number of occurrences of 0-seqs with length l in $L_{GOP,seqs}[n_G, i]$. Continuing with our example it means that $H_{GOP,seqs}[0, 2] = 1$ because there is only one 0-seq with length 2 in $L_{GOP,seqs}[0, i] = \{2\}$, $H_{GOP,seqs}[0, 3] = 0$ because there are no 0-seqs with length 3, and so forth. Notationwise, we will skip $H_{GOP,seqs}[n_G, 0]$ because it is always zero, and also skip all the trailing zeros. Thus for our example here are all values of $H_{GOP,seqs}$:

$$H_{GOP,seqs}[n_G, l] = \{\{0, 1\}, \{\}, \{0, 0, 0, 0, 1\}, \{0, 1\}\}$$

Similarly we construct $H_{seqs}[l]$ which is a function that represents total number of occurrences of length l in $L_{seqs}[i]$ (0-seqs with length l in I_{frames}). In our example 0-seqs with length 2 show up in total 2 times in L_{seqs}, so $H_{seqs}[2] = 2$. For all values of l in our example $H_{seqs}[l] = \{0, 2, 0, 0, 1\}$.

Our goal is to calculate $\hat{H}_{seqs}[l]$ which is the estimation of $H_{seqs}[l]$. To do this we first calculate $\hat{H}_{\overline{GOP},seqs}[l]$ which is defined as average of $H_{GOP,seqs}[n_G, l]$ excluding GOPs where 0-seqs with length l cannot physically end. For example, 0-seq with length 1 can belong to any GOP, but 0-seq with length $G+1$ cannot belong to (end in) first GOP - maximun length of 0-seq from first GOP is G. More precisely, $\hat{H}_{\overline{GOP},seqs}[l]$ is an expectation of $H_{GOP,seqs}[n'_G, l]$ with n'_G being a random variable with uniform distribution over indices

Symbol	Definition
N	Length of video segment
G	Size of GOP
N_G	Number of GOPs in video segment
g	Index of frame inside GOP
n	Index of frame inside video segment
n_G	Index of GOP inside video segment
l	Length of 0-seq
i	Index of 0-seq

Table 4.1: Symbol definitions

of GOPs where 0-seq with length l can end ($\lceil l/G \rceil - 1 \leq n'_G < N_G$):

$$\hat{H}_{\overline{GOP},seqs}[l] = \frac{1}{N_G - \lceil l/G \rceil + 1} \sum_{n_G=\lceil l/G \rceil - 1}^{N_G - 1} H_{GOP,seqs}[n_G, l] \tag{4.7}$$

Then $\hat{H}_{seqs}[l]$ can be calculated in following manner:

$$\hat{H}_{seqs}[l] = (N_G - \lceil l/G \rceil + 1)\hat{H}_{\overline{GOP},seqs}[l] \tag{4.8}$$

We will be taking closer look at several common GOP structures used: all frames are P and in the beginning of GOP we have an I (IPP..P), all B frames with I in the beginning (IBB..B) and most popular and used for HEVC hierarchical B structure shown in fig. 2.3 (Hierarchical-B, H-B).

Our goal is to calculate $\hat{H}_{seqs}[l]$ and we will be using $\hat{H}_{\overline{GOP},seqs}[l]$ as an estimation target for IPP..P and IBB..B structures, and for H-B structure we will derive formula for $\hat{H}_{seqs}[l]$ directly. Because notation of $\hat{H}_{G\bar{O}P,seqs}[l]$ is cumbersome, we will use a shorthand notation for it $\hat{H}[l] = \hat{H}_{\overline{GOP},seqs}[l]$.

All symbols that will be often used and were introduced in probabilistic model are summarized in table 4.1

4.2.2 Derivation of $\hat{H}_{seqs}[l]$

IPP..P

Assume that that there is I frame at the beginning of each GOP and only P frames in the GOP (see fig. 2.1). We want to derive $\hat{H}[l]$ for this particular case. Let us look at particular GOP and calculate the occurrence probability of sequence of each length. We assume that the probability of actual I frame loss is constant and equal to p_I and p_P for a P frame respectively.

The only way we can have a sequence of length 1, or single frame undecodable is in the end of GOP ($Pr = p_P$), so it will not render other frames undecodable. Also, we need all other frames decodable in this GOP ($Pr = (1 - p_I)(1 - p_P)^{G-2}$). We don't care about previous GOP, but we need to know that the I frame in the next GOP is decodable ($Pr = (1 - p_I)$) otherwise there will be more consecutive zeros. By multiplying probabilities of these independent events we can get the value for $\hat{H}[1]$:

$$\hat{H}[1] = (1 - p_I)^2(1 - p_P)^{G-2}p_P$$

We can extend this concept to get occurrence probabilities of sequence of length l, where $1 \leq l \leq G - 1$. To get it we need first I frame and next $G - 1 - l$ P frames not lost, next $(G - l)$th P frame lost, and we don't care about next $l - 1$ P frames because anyway there are undecodable due to them referencing $(G - l)$th P frame:

$$\hat{H}[l] = (1 - p_I)^2(1 - p_P)^{G-1-l}p_P, 1 \leq l \leq G - 1$$

For convenience, let us define $\hat{H}[0]$ as probability of no frames lost in particular GOP including the I frame of next GOP:

$$\hat{H}[0] = (1 - p_I)^2(1 - p_P)^{G-1}$$

To get the sequence of length exactly G in particular GOP we have two cases: our GOP either is or isn't the first GOP of the segment. In the first case we need to have

its I frame lost ($Pr = p_I$) and I frame of next GOP not lost ($Pr = 1 - p_I$). Probability of this case occurring is $Pr = 1/N_G$. For the second case except for the things already mentioned, we also need all frames from previous GOP to be decodable ($Pr = \hat{H}[0]$). Because $\hat{H}[0]$ includes probability of decodability of this GOP's I frame, which is not the case, we transform the meaning of term $(1 - p_I)$ from $\hat{H}[0]$ to represent the decodability of next GOP's I frame and now everything is taken into account. Adding probabilities yields following equation:

$$\hat{H}[G] = (\frac{1}{N_G}(1 - p_I) + (1 - \frac{1}{N_G})\hat{H}[0])p_I$$

We can use recursive relation to derive values of $\hat{H}[l]$ for values of $l > G$ (except multiples of G). To obtain the sequence of length l we need the sequence of length $l - G$ from previous GOP, I frame of this GOP lost ($Pr = p_I$) and I frame of GOP after it not lost (it is already counted in $\hat{H}[l - G]$ as I frame of this GOP), so:

$$\hat{H}[l] = p_I\hat{H}[l - G], l > G$$

Using the same logic as for derivation of $\hat{H}[G]$ we can calculate probabilities for multiples of G. For k I frames lost first case is when they are in the beginning of segment, so we don't need to account for $\hat{H}[0]$, second case is all next GOPs. The only thing is that the number of possible GOPs to have 0-seq will reduce as l increases.

$$\hat{H}[kG] = (\frac{1}{N_G - k + 1}(1 - p_I) + (1 - \frac{1}{N_G - k + 1})\hat{H}[0])p_I^k$$

Now we can unwrap the recursive relation and get the closed-form formula for $\hat{H}[l]$:

$$\hat{H}[l] = \begin{cases} (\frac{G}{N-l+G} + (1 - \frac{G}{N-l+G})(1 - p_I)(1 - p_P)^{G-1})(1 - p_I)p_I^{l/G} & \text{if } l \bmod G = 0 \\ (1 - p_I)^2(1 - p_P)^{G-1-l \bmod G}p_I^{\lfloor l/G \rfloor}p_P & \text{else} \end{cases} \quad (4.9)$$

IBB..B

Now let us consider similar structure, but with B frames instead of P frames (probability of B frame loss is denoted as p_B) (see fig. 2.2).

First let's again consider occurrence probability of the sequence of length 1. We definitely need both I frames at the beginning of this GOP and next GOP not to be lost ($Pr = (1-p_I)^2$), or the sequence will be covering whole GOP. Then the actual sequence can only be got by loss of single B frame ($Pr = p_B$) while 2 neighboring frames are decodable. They both can be B frames ($Pr = (1-p_B)^2$) and there are $G-3$ positions for this to be the case; or one of them can be I frame, another B ($Pr = 1-p_B$), and there are only 2 way for this to happen. Combining everything, we get $\hat{H}[1]$:

$$\hat{H}[1] = (1-p_I)^2 p_B((G-3)(1-p_B)^2 + 2(1-p_B))$$

Extending this way of thinking to values of $l \leq G-2$ we still have two cases of sequence being surrounded by B frames, or being in the beginning or the end of GOP. Loss probability of frames in sequence is now p_B^l, number of different positions for first case is now $G-2-l$, other values stay the same:

$$\hat{H}[l] = (1-p_I)^2 p_B^l((G-2-l)(1-p_B)^2 + 2(1-p_B)), l \leq G-2$$

Simplifying the equation we get:

$$\hat{H}[l] = (1-p_I)^2 p_B^l(1-p_B)((G-2-l)(1-p_B) + 2), l \leq G-2$$

To get the sequence of length $G-1$ we need to have all B frames in GOP lost, and both I frames not lost:

$$\hat{H}[G-1] = (1-p_I)^2 p_B^{G-1}$$

For higher values of l we need I frame to be lost. But as soon as it is the case, it affects all B frames to both left and right of it and length of sequence becomes $2G-1$., unless it is in the beginning of video slice ($l = G$). So actually B frames loss don't influence values

of $\hat{H}[l]$, $l > G-1$. It is solely determined by number of I frames lost in consecutive GOPs and their position. If their number is k, then length of sequence is $Gk+G-1$ if it doesn't include first GOP ($Pr = \frac{N_G-k}{N_G-k+1}$) and Gk if it includes ($Pr = \frac{1}{N_G-k+1}$), probability of loss is $(1-p_I)^2 p_I^k$. Occurrence of sequences of other lengths is 0.

Putting everything together yields the complete model for $\hat{H}[n]$:

$$\hat{H}[l] = \begin{cases} (1-p_I)^2 p_B^l (1-p_B)((G-2-l)(1-p_B)+2) & \text{if } 1 \le l \le G-2 \\ (1-p_I)^2 p_B^{G-1} & \text{if } l = G-1 \\ \frac{N_G-k}{N_G-k+1}(1-p_I)^2 p_I^k & \text{if } l = Gk+G-1, 1 \le k < N_G \\ \frac{1}{N_G-k+1}(1-p_I)^2 p_I^k & \text{if } l = Gk, 1 \le k \le N_G \\ 0 & \text{else} \end{cases} \tag{4.10}$$

Hierarchical-B

Now we will derive expression for $\hat{H}_{seqs}[l]$ for a Hierarchical-B structure (see fig. 2.3). GOP size should be a power of 2, and we will define intra-period length as N_{ref}. Intra-period is a period of insertion of I frame in the beginning of GOP. We will assume that N_{ref} is multiple of G and N is a multiple of N_{ref}. Also let's define $N_{G,ref} = N_{ref}/G$ that represents number of GOPs in intra-period.

Let us first examine the case of 0-seqs because of B frame losses. We will assume that closest P frames on the edges of GOP are decodable. Define $T = \log_2 G$ as a maximum depth of B layers. For example, if $G = 2$, it means that there is maximum depth of 1 layer consisting of one B frame. If $G = 8$, then there are 3 layers of B frames beneath P frames. Possible l values in this case are limited to 1, 3, 7, in general $2^t - 1$ frames undecodable, where $1 \le t \le T$.

Because probabilities of P frames to be undecodable are different for each GOP (like P frames in IPP..P structure), to get expected number of 0-seqs with length $1 \le l \le 2^T - 1$ we need to add values from each GOP (they will be different). Thus we need to calculate probabilities of P frames losses. Define $P_{allP}[n_G]$ as a probability of receiving I frame and

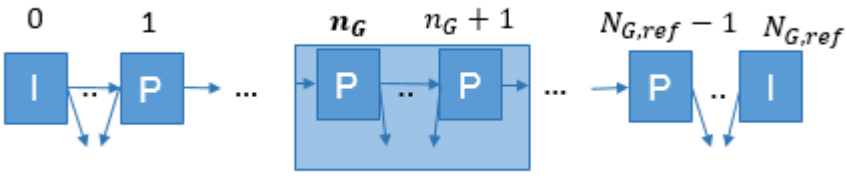


Figure 4.2: P frame indices in H-B

all P frames till n_G-th included (index starting with 0 for I-frame, see fig. 4.2).

$$P_{allP}[n_G] = (1 - p_I)(1 - p_P)^{n_G}, 0 \leq n_G < N_{G,ref}$$

Because $P_{allP}[n_G]$ will be used as an indicator that both non-B frames on the edges of GOP are received, we also need to introduce value $P_{allP}[N_{G,ref}]$ that includes receiving probability of next GOP's I frame that is required for the last GOP in segment.

$$P_{allP}[N_{G,ref}] = (1 - p_I)^2(1 - p_P)^{N_{G,ref}}$$

Now we can break down probability of having 0-seq of length l in n_G-th GOP, and sum them over all GOPs (don't forget that there are N/N_{ref} intra-periods):

$$\begin{aligned}
\hat{H}_{GOP,seqs}[n_G, l] &= \frac{N}{N_{ref}} P_{allP}[n_G + 1] P_B[l], 0 < l < G \\
\hat{H}_{seqs}[l] &= \frac{N}{N_{ref}} \sum_{n_G=0}^{N_G-1} P_{allP}[n_G + 1] P_B[l] = \\
&= \frac{N}{N_{ref}} (1 - p_I)(1 - p_P)(\frac{1 - (1 - p_P)^{N_{G,ref}}}{p_P} - p_I) P_B[l]
\end{aligned} \tag{4.11}$$

Where $P_B[l]$ is a probability of getting 0-seq with length l due to B frame loss in a GOP. It can be easily seen that if B frame in layer t from bottom ($T - t$ from top) is lost, when depth is T, it will lead to loss of $2^t - 1$ consecutive B frames.

$$P_B[2^t - 1] = 2^{T-t}(1 - p_B)$$

Also it is easy to calculate number of B frames on layer t from top: there are 2^t

different places where B frame can be lost.

Substituting expression for $P_B[l]$ into equation 4.11 yields complete formula to calculate $\hat{H}_{seqs}[l]$ for $l < G$:

$$\hat{H}_{seqs}[l] = \frac{N}{N_{ref}}(1-p_I)(1-p_P)(\frac{1-(1-p_P)^{N_{G,ref}}}{p_P} - p_I)2^{(T-t)}(1-p_B)^{(T-t)}p_B \quad (4.12)$$

Now for the values of $G \leq l < N_{ref}$: as soon as we lose at least one P frame, all P frames on right till I frame and B frames left of it till P frame will be undecodable. Assume that first lost P frame in intra-period has index n_G, then length of 0-seq will be $G(N_{G,ref} - n_G) + G - 1$ and probability will be equal to:

$$\begin{aligned}
\hat{H}_{seqs}[G(N_{G,ref} - n_G) + G - 1] &= \frac{N}{N_{ref}}(1-p_I)P_{allP}[n_G - 1]p_P \\
&= \frac{N}{N_{ref}}(1-p_I)^2(1-p_P)^{n_G-1}p_P, 0 < n_G < N_{G,ref} \\
\hat{H}_{seqs}[Gk + G - 1] &= \frac{N}{N_{ref}}(1-p_I)^2(1-p_P)^{N_{G,ref}-k-1}p_P, 1 \leq k < N_{G,ref}
\end{aligned} \quad (4.13)$$

And for the last part we need to consider what will happen when I frames are lost. If we have one I frame lost with all P frames from previous intra-period and next I frame received, length of 0-seq will be $N_{ref} + G - 1$. Important point here is that in case of I frame being lost in the beginning of the segment we cover frames lost in the end of previous segment (otherwise it severely complicates equations). This way of thinking with slight adjustments (number of possible positions is reduced) applies also to k I frames lost.

$$\begin{aligned}
\hat{H}_{seqs}[N_{ref} + G - 1] &= \frac{N}{N_{ref}}P_{allP}[N_{G,ref}]p_I \\
\hat{H}_{seqs}[kN_{ref} + G - 1] &= \frac{N - (k-1)N_{ref}}{N_{ref}}P_{allP}[N_{G,ref}]p_I^k
\end{aligned} \quad (4.14)$$

In case if at least one P frame is lost and next I frame is also lost, it yields slightly different equation, because we need 2 intra-periods to constrain on, which reduces number

of possible positions among intra-periods by 1.

$$\hat{H}_{seqs}[N_{ref}+l] = \frac{N-N_{ref}}{N_{ref}}\hat{H}_{seqs}[l]p_I$$
$$\hat{H}_{seqs}[kN_{ref}+l] = \frac{N-kN_{ref}}{N_{ref}}\hat{H}_{seqs}[l]p_I^k \tag{4.15}$$

Putting together equations 4.12, 4.13, 4.14 and 4.15 and making remaining substitutions yields complete model for Hierarchical-B structure:

$$\hat{H}_{seqs}[l] = \begin{cases} \frac{N}{N_{ref}}(1-p_I)(1-p_P)(\frac{1-(1-p_P)^{N_{G,ref}}}{p_P} - p_I)(2(1-p_B))^{T-t}p_B \\ \qquad\qquad l = 2^t - 1, 1 \le t \le \log_2 G, t \in Z \\ \frac{N-(k-1)N_{ref}}{N_{ref}}(1-p_I)^2(1-p_P)^{N_{G,ref}}p_I^k \\ \qquad\qquad l = N_{ref}k + G - 1, 1 \le k \le \frac{N}{N_{ref}}, k \in Z \\ \frac{N-k_2N_{ref}}{N_{ref}}(1-p_I)^2(1-p_P)^{N_{G,ref}-k_1-1}p_P p_I^{k_2} \\ \qquad\qquad l = Gk_1 + G - 1 + N_{ref}k_2, \\ \qquad\qquad 1 \le k_1 < N_{G,ref}, 0 \le k_2 < \frac{N}{N_{ref}}, k_1, k_2 \in Z \end{cases} \tag{4.16}$$

4.3 Degradation Model

4.3.1 Definitions

Let us denote by V encoded video as the sequence of frames, so $V[n]$ represents n-th frame in the video. Moreover, let us denote by V' concealed video. At the point in video where we have several consecutive frames lost, we introduce degradation function $D[l,i]$ which is defined as follows. Let $n_{last,i}$ be position of last frame that was decoded successfully for i-th 0-seq and $n_{last,i}+1$ is the first corrupted frame Also assume that the length of this i-th 0-seq is L_i.

$$D[l,i] = PSNR_{frame}(V[n_{last}+l], V'[n_{last}+l]), \text{ where } 1 \le l \le L_i$$

In other words, $D[l,i]$ tells us what is the value of PSNR between original video frame and the one that was used to conceal it after l frames lost into the i-th 0-seq. $D[1]$ corresponds to first lost frame, $D[2]$ to the second, and so forth.

We also define average degradation function as degradation, averaged over all 0-seqs, $D[l] = \sum_{i:l \leq L_i} D[l,i]$.

Our goal is to predict average degradation function, that will be used in conjunction of probabilistic model to estimate average degradation of the video. For the remainder of the section we assume that the concealment strategy is using the latest non-corrupted frame in placement of corrupted ones:

$$V'[n_{last,i} + l] = V[n_{last,i}], 1 \leq l \leq L_i,$$

4.3.2 Prediction of $D[l]$

After numerous observations it was evident that all degradations in different videos including average degradation can be modeled by some function. For example, different degradations for video sequence Cactus can be seen on figure 4.3. They were obtained by calculating PSNR between starting frame and frame Δ frames to the right. 10 different starting frames were chosen uniformly distributed along the whole video sequence that provide 10 different lines.

To predict PSNR drop pattern or $D[l]$ we introduce $\hat{D}[l]$ which is the estimation of $D[l]$. We will use non-linear prediction model to calculate $\hat{D}[l]$:

$$\hat{D}[l] = \frac{\alpha e^{-\gamma l}}{l^{\beta}} + \delta \tag{4.17}$$

This prediction model is based on shape of degradation observations and comparing to other models (polynomial, exponential, etc. - see fig. 4.2) achieves best fitting performance.

To estimate parameters α, β, γ and δ, which are video content dependent, we are using machine learning on the period of video with artificial data $\widetilde{D}_{(l)}^{(m)}$, where l corresponds to

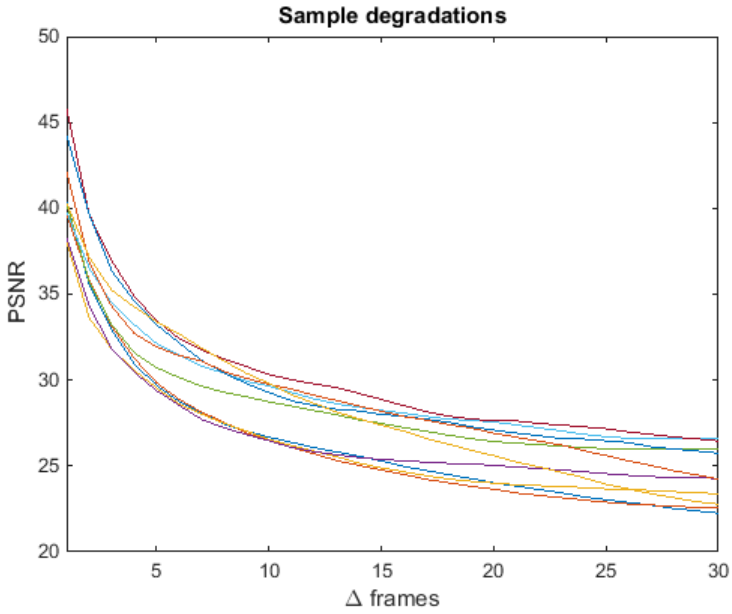


Figure 4.3: Degradation observations

Model	R^2
Proposed	0.9999
Polynomial (4th degree)	0.9901
Polynomial (3rd degree)	0.9807
Exponential	0.8743
Laplacian	0.2321
Rayleigh	0.0753

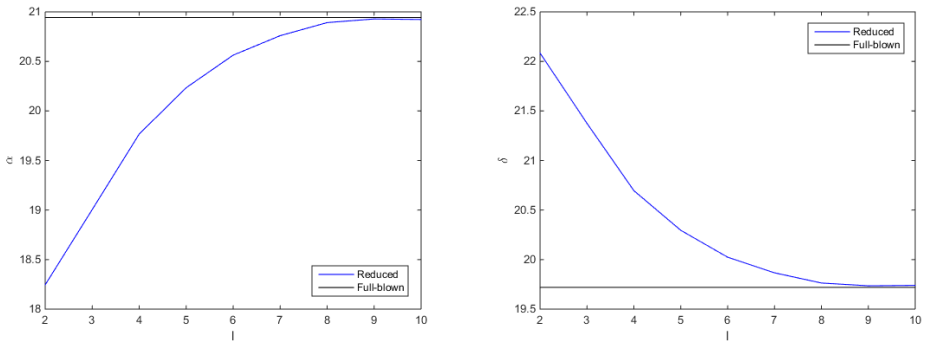
Table 4.2: Degradation fitting performance

argument of $D[l]$, and m is index of sample for that l. Basically, what we want to do is to get M samples for each value of $D[l]$ up to L from video slice V with total of N frames in it. We assume that content variations inside the video slice are minimal, so we can use one set of model parameters to successfully represent possible degradations.

Let us introduce hop size $h = \left\lfloor \frac{N-L}{M} \right\rfloor$ which is equal to number of frames to jump when m is increased. Then we can use the following formula to gather data:

$$\widetilde{D}_{(l)}^{(m)} = PSNR_{frame}(V[mh], V[mh+l]), 1 \leq l \leq L, 0 \leq m < M$$

Now we have ML samples to learn model parameters for particular video slice. So we use machine learning to find optimal model parameters to minimize aggregated square



(a) α (b) δ

Figure 4.4: Reduced complexity estimation

error:

$$\arg\min_{\alpha,\beta,\gamma,\lambda} \sum_{l=0}^{L} \sum_{m=0}^{M-1} (\widetilde{D}_{(l)}^{(m)} - \hat{D}[l])^2$$

In real-time streaming full-blown prediction isn't usually feasible because of amount of computations needed, so we introduce reduced complexity prediction that can be used. After performing fitting for multitude of different videos and comparing parameter values, it was evident that γ and β have less variance comparing to α and δ (see table 5.4 in chapter 5). So to reduce amount of numerical operations we can use heuristic values for γ and β: γ_0 and β_0, and then we need to estimate only two parameters. Values for β_0 and γ_0 were obtained by calculating average over 6 different video sequences.

Moreover, it simplifies estimation process. For each segment only several frames from its beginning are required to do the estimation. Essentially a system of equations needs to be solved:

$$\begin{cases} \alpha F(1) + \delta = \Delta(1) \\ \alpha F(2) + \delta = \Delta(2) \\ \dots \\ \alpha F(l) + \delta = \Delta(l) \end{cases}$$

Where $F(l) = \frac{e^{-\gamma_0 l}}{l^{\beta_0}}$ is a constant for any particular l and $\Delta(l)$ is a PSNR between first and $(l+1)$th frame. So, minimum number frames to successfully calculate value for α and β is 3 yielding system of $l = 2$ equations and 2 variables. Number of equations can

be increased and solved approximately using linear algebra giving more accurate results (see fig. 4.4).

In case of different error concealment strategy used, degradation model should be revised, but concept is still applicable.

4.4 Aggregated Model

Our final goal is to estimate average quality difference between original video and degraded video as a function of weighting factor used in LMRC codes.

Degradation model allows us to estimate degradation function $\hat{D}[l]$, and probabilistic model allows us to estimate number of 0-seqs according to their length $\hat{H}_{seqs}[l]$. Assume that we know average quality of encoded video in terms of PSNR between it and original video Q_{enc}. Let's take one frame and combine Q_{enc} (PSNR between encoded video and original one) and Q_{deg} (PSNR between degraded video and encoded one): in worst case noise introduced by video encoding and error concealment is added, what leads us to following equation for a function to combine PSNR values $J(Q_{enc}, Q_{deg})$

$$PSNR = \log_{10} \frac{MAX^2}{\sigma^2}$$

$$\sigma_J = \sigma_{enc} + \sigma_{deg}$$

$$Q_J = J(Q_{enc}, Q_{deg}) = -2 \log_{10}(10^{-Q_{enc}/2} + 10^{-Q_{deg}})$$

Let us introduce joint degradation function that takes care of combining PSNR from two sources: encoding and degradation:

$$\hat{D}_J[l] = J(\hat{D}[l], Q_{enc})$$

Now for calculation of PSNR of degraded video we have to consider 2 classes of frames: degraded and non-degraded. Number of degraded frames can be calculated using proba-

bilistic model, where we know that there are $\hat{H}_{seqs}[l]$ 0-seqs with length of l frames:

$$\hat{N}_{deg} = \sum_{l=1}^{L} l\hat{H}_{seqs}[l]$$

Using degradation model we can calculate sum of PSNR of l-length 0-seq frames l: $\sum_{i=1}^{l} \hat{D}_J[i]$. Thus by summing PSNR of all frames and dividing it by number of frames we get average PSNR of video segment:

$$\hat{Q}_D = \frac{1}{N}(\sum_{l=1}^{L} \hat{H}_{seqs}[l] \sum_{i=1}^{l} \hat{D}_J[i] + (N - \sum_{l=1}^{L} l\hat{H}_{seqs}[l])Q_{enc}) \tag{4.18}$$

By subtracting $\hat{Q}_D$ from Q_{enc} we can get quality difference ΔQ:

$$\Delta Q = Q_{enc} - \hat{Q}_D = \frac{1}{N}(Q_{enc} \sum_{l=1}^{L} l\hat{H}_{seqs}[l]) - \sum_{l=1}^{L} \hat{H}_{seqs}[l] \sum_{i=1}^{l} \hat{D}_J[i]) \tag{4.19}$$

Using LMRC model we can calculate $p_{MIB}(\omega)$ and $p_{LIB}(\omega)$, transitive model allows us to get $p_I(\omega)$, $p_P(\omega)$ and $p_B(\omega)$, probabilistic model gives us $\hat{H}_{seqs}[l]$ with ω as parameter, and ultimately we are able to calculate $\hat{Q}_D$.

From numerous observations we can assume that $\hat{Q}_D(\omega)$ is concave function (see fig. 5.5 in chapter 5). It makes sense, because as ω increases, loss probability of MIB layer is decreasing while for LIB layer it is increasing. Thus there should be point of trade-off, where increase in LIB loss will overshadow gain from reduction of MIB loss, thus decreasing PSNR.

Aggregated model in conjunction with LMRC model gives us formula that can be used to calculate $\hat{Q}_D$ as a function of p_{loss} and ω. Thus we can use it to find value of ω that maximizes its value. Due to its concavity we can use gradient descent method to find maximum. Because of difficulty of deriving $\frac{d}{d\omega}\hat{Q}_D(\omega)$ we are going to use numerical approach ($\frac{d}{d\omega}\hat{Q}_D(\omega_0) \approx \frac{\hat{Q}_D(\omega_0+\delta)-\hat{Q}_D(\omega_0)}{\delta}$) to estimate value of derivative.

After the optimal value of ω is found, it is used to perform channel coding on video stream and it is transmitted through the network.

Chapter 5

Experimental Results

In this chapter, first we give a brief introduction of our experimental environment including the testing datasets and the metrics we used to evaluate the accuracy. We evaluate accuracy of each individual model and aggregated one. Also we perform performance comparison of our proposed method to RE-RS codes [18].

For the following experiments we are using 4 video sequences from JCT-VC provided sequences of different classes.

For HEVC encoding we are using HM software version 16.6 with low delay profile, base QP value of 32 if not specified otherwise.

5.1 Transitive model

One of the assumptions that we made for transitive model was about small relative variations of frame sizes in the same frame type. To verify it, we have used 4 video sequences to gather information about frame size variations. They were encoded using Hierarchical-B GOP structure with $G = 4$ and $N_{ref} = 32$. Summary of data gathered is

Name	Resolution	FPS	Frames
KristenAndSara	1280x720	60	600
FourPeople	1280x720	60	600
ParkScene	1920x1080	24	240
Cactus	1920x1080	50	500

Table 5.1: Video sequences used

Video	Frame	Mean	Std	Std/Mean	Packet	$S_{(frame)}$
Kristen And Sara	I	17.7K	0.3K	1.7%	150	118
	P	1.92K	0.48K	25%		13
	B	67, 184	27, 64	40%, 34%		2
Four People	I	26.6K	0.3K	1.2%	200	123
	P	3.12K	0.73K	23%		16
	B	165, 424	65, 136	39%, 32%		3
Park Scene	I	81.7K	5.8K	7.1%	500	164
	P	26.1K	2.9K	11%		53
	B	1.0K, 2.7K	0.3K, 0.5K	30%, 18%		3
Cactus	I	73.0K	2.4K	3.3%	900	82
	P	22.5K	1.5K	6.7%		25
	B	1.4K, 3.2K	0.4K, 0.6K	28%, 18%		4

Table 5.2: Frame size statistics

shown in table 5.2.

Thus, if we will choose $S_{(frame)}$ that will be bigger than actual amount of packets per frame, we are estimating PSNR for the worst-case scenario, so we can guarantee that actual PSNR will be higher than we estimated.

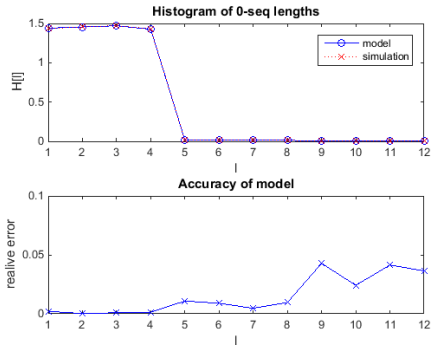
One of the important part of packaging strategy is decision of packet size, it will determine $S_{(frame)}$ for each frame, thus frame loss probabilities. General strategy is to choose balance between one that will make section size not too big (it determines playback delay, if number of message symbols per sections remains roughly constant) and one that will not be too small for header overhead to become noticeable.

5.2 Probabilistic model

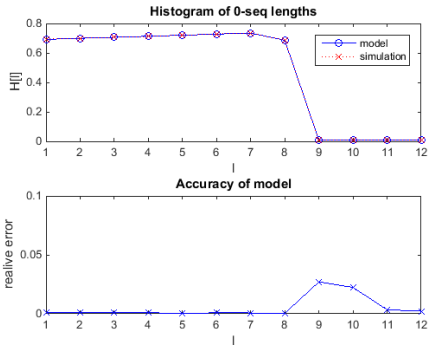
To verify accuracy of the probabilistic model, given set of frame loss probabilities we have simulated loss pattern (I_{frames}) for a sequence of length N, and compared probabilities from estimations to actual values from simulation. Scenarios used are: high and low loss. Number of simulations for one point is $1E+6$.

GOP structures used in the simulations are: IPP..P with GOP size of 4 and 8 (see fig 5.1); IBB..B $G \in \{4, 8\}$ (fig 5.2); and Hierarchical-B $N_{ref} = 32$, $G \in \{4, 8\}$ (fig 5.3)

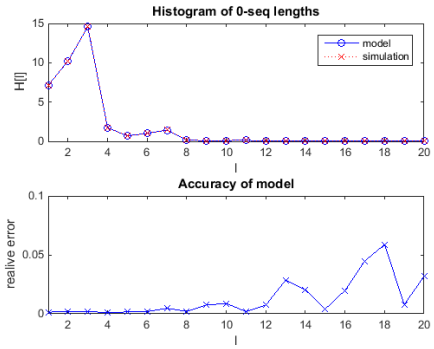
To evaluate accuracy of the model we are using relative error metric ($\frac{\|estimated - simulated\|}{simulated}$). Results show that relative accuracy is generally less than 0.05 and for low values with high



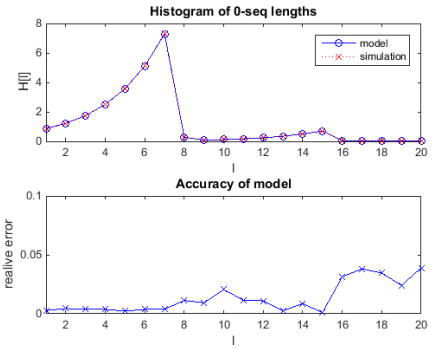
(a) $g = 4$, $p_I = 0.01, p_P = 0.01$



(b) $g = 8$, $p_I = 0.01, p_P = 0.01$

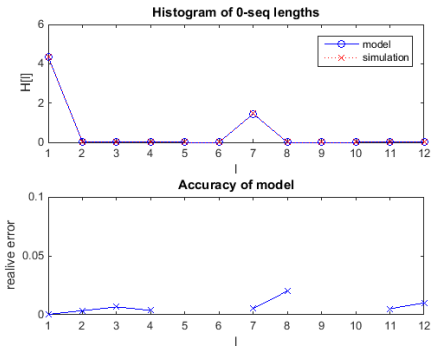


(c) $g = 4$, $p_I = 0.1, p_P = 0.3$

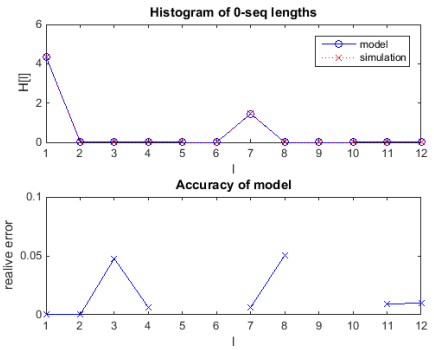


(d) $g = 8$, $p_I = 0.1, p_P = 0.3$

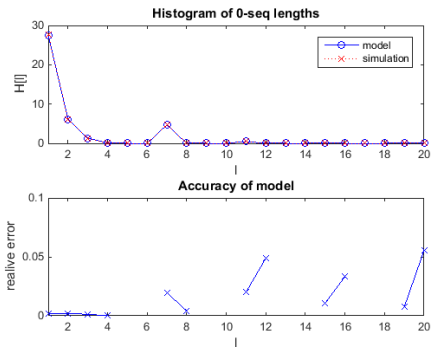
Figure 5.1: Probabilistic model, IPP..P structure



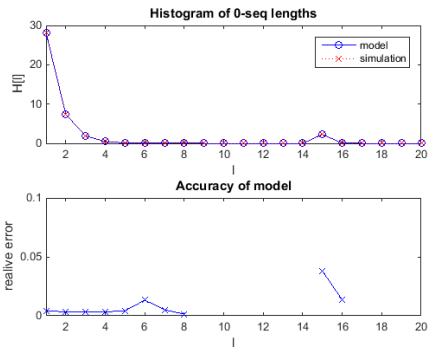
(a) $g = 4$, $p_I = 0.01, p_B = 0.01$



(b) $g = 8$, $p_I = 0.01, p_B = 0.01$

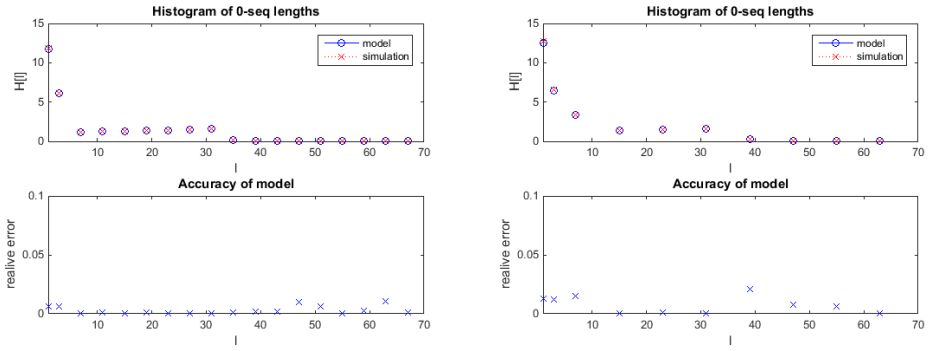


(c) $g = 4$, $p_I = 0.1, p_B = 0.3$

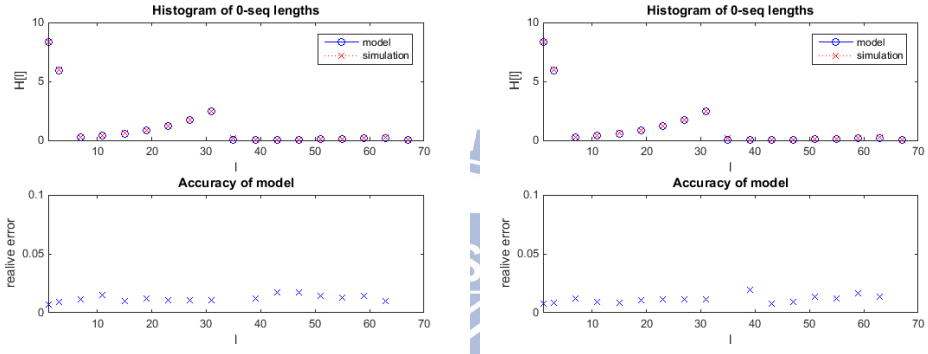


(d) $g = 8$, $p_I = 0.1, p_B = 0.3$

Figure 5.2: Probabilistic model, IBB..B structure



(a) $g = 4$, $p_I = 0.01, p_P = 0.05, p_B = 0.02$ (b) $g = 8$, $p_I = 0.01, p_P = 0.05, p_B = 0.02$



(c) $g = 4$, $p_I = 0.1, p_P = p_B = 0.3$ (d) $g = 8$, $p_I = 0.1, p_P = p_B = 0.3$

Figure 5.3: Probabilistic model, H-B structure

probability is less than 0.01. Moreover increase in number of simulations makes relative error converge to 0.

Thus we can make a conclusion that our probabilistic model is accurate.

5.3 Degradation model

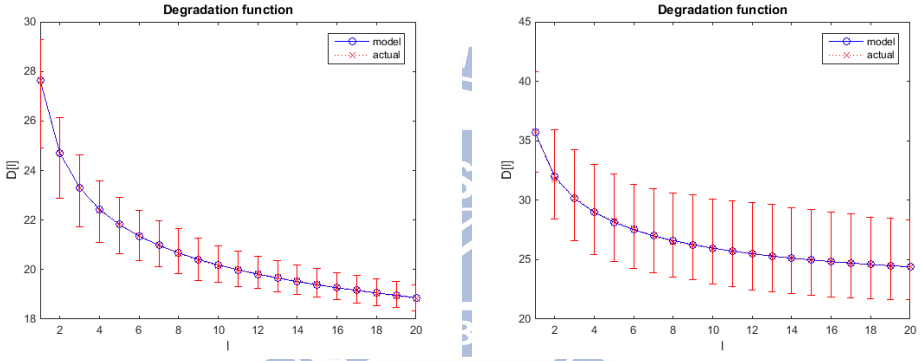
Degradation model experiments involve estimation of goodness of the fit for model (see equation 4.17). We are using 4 video sequences introduced earlier to see how different is the degradation function for each of those (fig. 5.4). Red points on top and bottom indicate maximum and minimum PSNR value for that particular l, red cross shows average value.

Blue line shows fitting curve that had its parameters trained using number of sample points for each l of 10 and maximum value for l of 20, so in total there are 200 points. We were using non-linear regression to find curve parameters using gradient descent (MAT-

LAB function `fitnlm`). In live scenario number of sample points can be reduced to slightly larger than 4 (because we have 4 parameters) and maximum length can be defined by point where probability of occurrence is low enough.

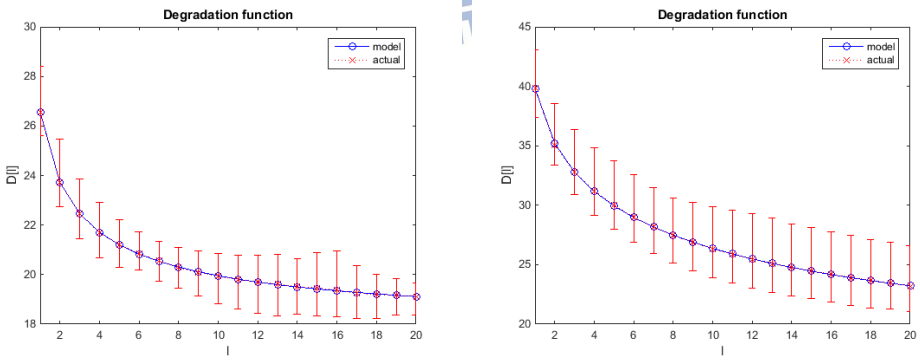
Video	Amount of motion	Accuracy (R^2)
KristenAndSara	low	0.99990
FourPeople	low	0.99982
Cactus	medium	0.99927
ParkScene	high	0.99984

Table 5.3: Accuracy of degradation model



(a) ParkScene

(b) FourPeople



(c) Cactus

(d) KristenAndSara

Figure 5.4: Degradation model

Table 5.4 shows how model parameters vary for different videos. This statistics was obtained from fitted parameter values over 4 videos, and it helped to devise reduced complexity estimation algorithms by fixing 2 out of 4 parameters and estimating other two.

We have also calculated heuristic values (γ_0, β_0) averaging fitted values over 6 different video sequences (additional are Johnny 720p and Kimono 1080p) for degradation function

Parameter	Estimated Value	Standard Deviation
α	15.98	6.04
β	0.39	0.09
γ	0.00621	0.00487
δ	16.62	4.29

Table 5.4: Degradation parameters statistics for different videos

Parameter	Value
β_0	0.39432
γ_0	0.0062167

Table 5.5: Degradation model: heuristic values

that can be used for reduced complexity estimation for any video. Lack of necessity to train a full model for specific video is compensated by reduced accuracy of the resulting model, however the trade-off is good enough for it to be viable strategy in real-time systems.

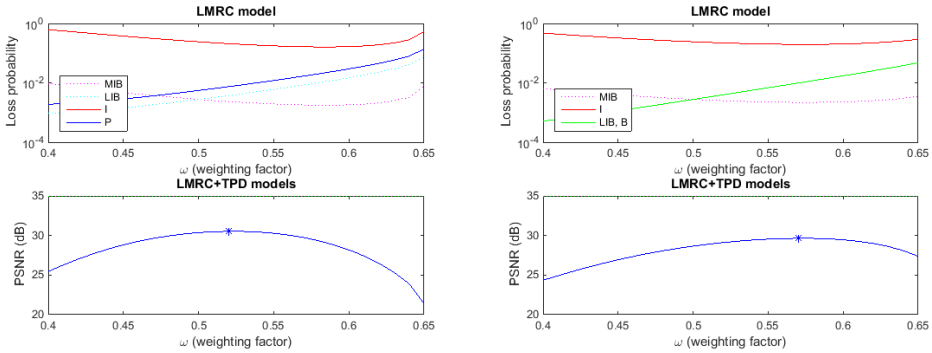
Simulation results show that our choice for a degradation function is valid and accurate, moreover we have derived heuristic degradation model that can be used instead of training video-specific one at the cost of whole system performance.

5.4 Aggregated model

To verify accuracy of aggregated model we have performed multiple experiments to measure values of estimated PSNR and average of simulated PSNR and comparing optimal value of ω for these two cases.

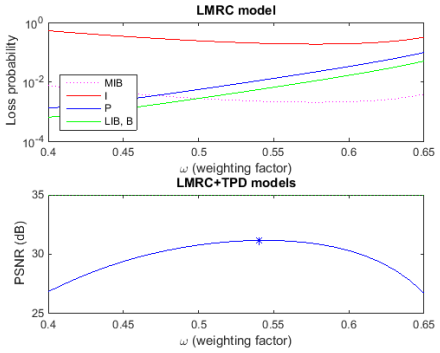
Figure 5.5 demonstrates how frame loss probabilities and estimated PSNR are changing depending on ω. The video used for that experiment is Cactus, for GOP structure we were using IPP..P ($G = 32$), IBB..B ($G = 32$) and Hierarchical-B with $N_{ref} = 32$ and GOP size values of 4 and 8. For HEVC encoding we were using main profile with QP equal to 32 which resulted in average PSNR of encoded video of 34.49 dB (it is indicated with dotted green line in bottom figure for each structure). For LMRC overhead we are using $\epsilon = 0.04$.

Another set of experiments shows how accurate is our aggregation model. For this

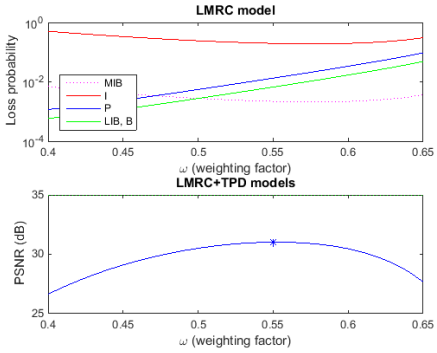


(a) IPP..P

(b) IBB..B



(c) H-B, $G = 4$



(d) H-B, $G = 8$

Figure 5.5: Aggregated model

Video	ϵ	ω (est)	ω (sim)
Four People	0.08	0.543	0.54
	0.12	0.547	0.54
Kristen And Sara	0.08	0.549	0.55
	0.12	0.551	0.55

Table 5.6: Estimation vs Simulation

experiment we used frame sizes from HEVC encoded video, used appropriate packetization schema, simulated LMRC encoding and decoding for packets and depending on which packets were lost, we marked those frames undecodable. Then we have applied error concealment strategy and calculated PSNR between original and concealed video. The goal is to see how estimated and actual PSNR values are changing according to ω. We used two video sequences (Kristen And Sara and Four People) and two values for overhead: $\epsilon = 0.08$ and $\epsilon = 0.12$ with 0.01 packet loss rate. Simulations were repeated 1E+4 times and PSNR value was averaged. As you can see from table 5.6, optimal value of ω found using estimated PSNR (proposed model) is located very close to point of maximum actual PSNR achieved what proves that our model is a viable and accurate.

Actual p_{loss}	0.03	**0.05**	0.07
QP=32	36.53	**36.63**	36.39
QP=28	39.47	**39.65**	39.27
QP=24	40.03	**40.42**	39.93

Table 5.7: p_l mismatch performance

To evaluate performance of our system in case when value of channel packet loss rate estimation is not accurate, we performed another set of experiments, where packet loss rate for simulations was different from one used as input for estimation model. Video sequence used is Kristen And Sara, $\epsilon = 0.2$, for estimation value of p_{loss} is 0.05, in reality it is larger. Average PSNR values for actual videos with 1E+4 simulations are shown in Table 5.7

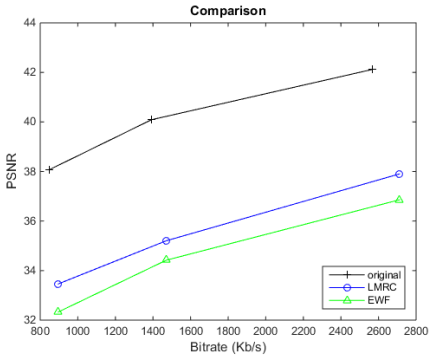
Also we have performed several experiments to see how using fully trained degradation model versus low-complexity estimation impacts both ω and PSNR estimation. Values of ω found using both methods were the same with threshold of 0.005 for gradient descent convergence, so we can safely say that performance is the same, but average time dropped from 1.4sec to 0.2sec in the MATLAB simulation, time of efficient implementation in C/C++ should be even less.

5.5 Comparison

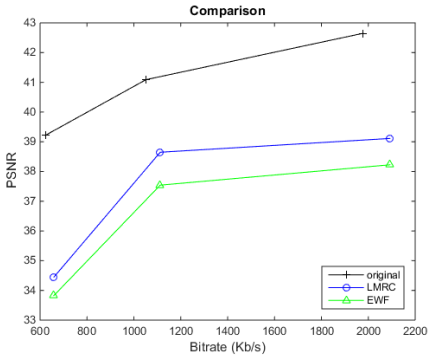
In this section we are comparing performance of our proposed method to Expanding Window Fountain codes used instead of LMRC in our system and Randomized Expanding Reed-Solomon codes [18].

For comparison of LMRC and EWF codes in the context of our system (see fig. 5.6) we have tested the performance in constrained bandwidth scenario ($\epsilon = 0.08, p_{loss} = 0.02$) using 4 video sequences introduced earlier. Aggregated model determines optimal weighting factor, channel coding is performed and we calculate average PSNR of degraded and concealed video over multiple simulations.

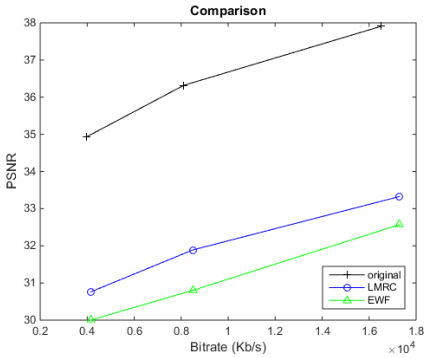
Another set of experiments was performed for comparison with RE-RS codes. We are using 2 scenarios: $p_{loss} = 0.05$ and $\epsilon = 0.2$; $p_{loss} = 0.1$ and $\epsilon = 0.4$. These overhead



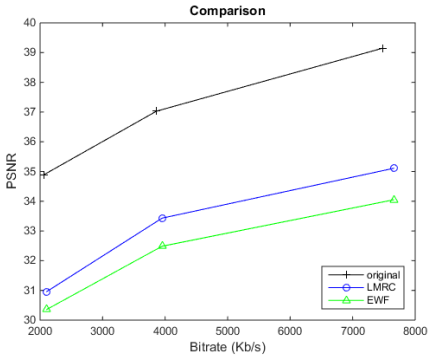
(a) Four People



(b) Kristen And Sara

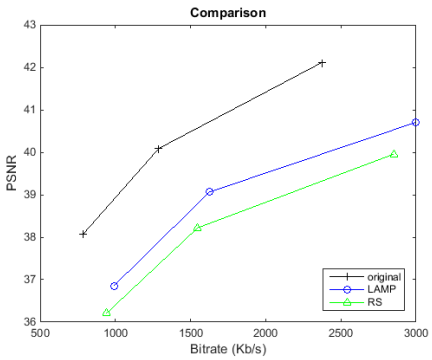


(c) Cactus

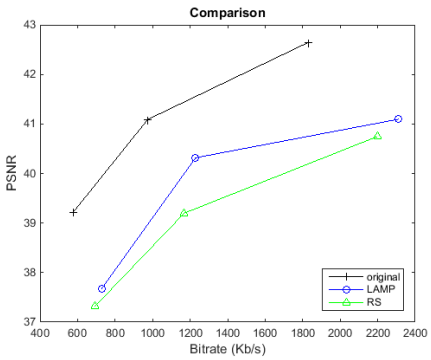


(d) Park Scene

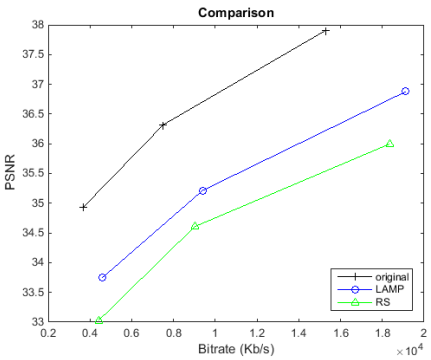
Figure 5.6: Comparison: EWF codes



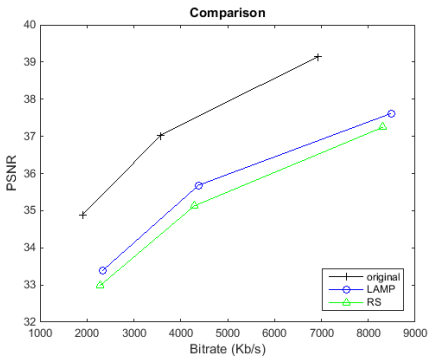
(a) Four People



(b) Kristen And Sara

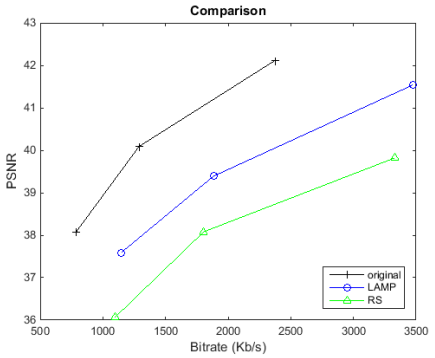


(c) Cactus

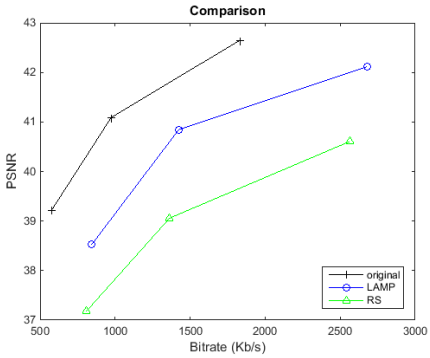


(d) Park Scene

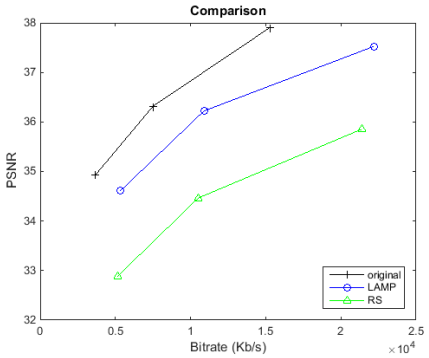
Figure 5.7: Comparison: $p_{loss} = 0.05$, $\epsilon = 0.2$



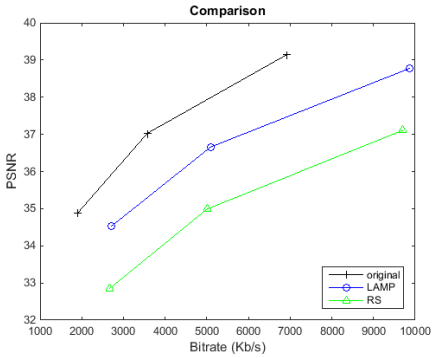
(a) Four People



(b) Kristen And Sara



(c) Cactus



(d) Park Scene

Figure 5.8: Comparison: $p_{loss} = 0.1$, $\epsilon = 0.4$

values are claimed to be optimal for RE-RS performance from the paper. All videos are encoded with 3 different QP values: 32, 28 and 24 and H-B GOP structure with $G = 4$ and $N_{ref} = 32$. Results can be seen in figures 5.7 and 5.8.

While both LMRC and EWF codes have UEP parameter that our proposed system is optimizing, the goal of LMRC vs RE-RS codes is to compare system end-to-end performance. In [18] general streaming scenario is considered, but there is no UEP parameter to optimize, RE-RS codes have UEP by design. Thus because contribution of the original paper was optimization of overhead given packet loss rate of the channel, to make a fair comparison, we are using the same value of the overhead and show that our system is able to achieve higher performance.

Moreover simulation results show that performance of our proposed method performs better than RE-RS codes for both constrained ($\epsilon = 0.2$) and large ($\epsilon = 0.4$) bandwidth. Moreover in very constrained bandwidth scenario ($\epsilon = 0.08$) LMRC outperforms EWF codes used with proposed system. The only downside for our method is that transmission

is performed section by section, so video streaming delay is inevitable. However if it is allowed by service QoE, our method achieves better video quality and what is more important - is adaptable to different GOP structures and channel loss rate. Channel coding performance of EWF codes is lower than LMRC codes, because we have only changed channel coding module and model for estimation and experiments. However lower margin between performance difference on EWF and LMRC compared to RE-RS and LMRC proves that proposed system is flexible and can be extended to support other channel codes as long as it has a model.

Chapter 6

Conclusion

In this paper we have proposed a dynamic system for channel coding of video stream that is sensitive to channel condition and video content and is capable of fine-tuning channel coding parameters in response to those changes. Moreover we have introduced **probabilistic** and **degradation** models. Probabilistic model provides a closed-form formulas to calculate estimated frequency of 0-seq occurrences depending on GOP structure used, when degradation model estimates impact of those on video quality (PSNR). We also provided reduced complexity methods of calculating degradation model that can be used in live streaming environment. These two models in conjunction can estimate video quality degradation after transmission and it is used in the proposed system for choice of optimal channel coding parameters.

Experiment results verified accuracy of each model individually as well as of whole system. Comparison of our proposed method to other methods in literature showed superiority of our method for the multiple scenarios.

Bibliography

[1] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 22, pp. 1649–1668, Dec 2012.

[2] I. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society of Industrial and Applied Mathematics*, vol. 8, p. 300–304, 06/1960 1960.

[3] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," *SIGCOMM Comput. Commun. Rev.*, vol. 28, pp. 56–67, Oct. 1998.

[4] M. Luby, "Lt codes," in *Foundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on*, pp. 271–280, 2002.

[5] A. Shokrollahi, "Raptor codes," *Information Theory, IEEE Transactions on*, vol. 52, pp. 2551–2567, June 2006.

[6] P. Maymounkov, *Online Codes*. PhD thesis, New York University, 2002.

[7] D. Sejdinovic, D. Vukobratovic, A. Doufexi, V. Senk, and R. Piechocki, "Expanding window fountain codes for unequal error protection," *Communications, IEEE Transactions on*, vol. 57, pp. 2510–2516, September 2009.

[8] P. Cataldi, M. Grangetto, T. Tillo, E. Magli, and G. Olmo, "Sliding-window raptor codes for efficient scalable wireless video broadcasting with unequal loss protection," *Image Processing, IEEE Transactions on*, vol. 19, pp. 1491–1503, June 2010.

[9] Y. Cao, S. Blostein, and W.-Y. Chan, "Unequal error protection rateless coding design for multimedia multicasting," in *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*, pp. 2438–2442, June 2010.

[10] N. Rahnavard, B. Vellambi, and F. Fekri, "Rateless codes with unequal error protection property," *Information Theory, IEEE Transactions on*, vol. 53, pp. 1521–1532, April 2007.

[11] D. Vukobratovic and V. Stankovic, "Unequal error protection random linear coding strategies for erasure channels," *Communications, IEEE Transactions on*, vol. 60, pp. 1243–1252, May 2012.

[12] A. Dimakis, J. Wang, and K. Ramchandran, "Unequal growth codes: Intermediate performance and unequal error protection for video streaming," in *Multimedia Signal Processing, 2007. MMSP 2007. IEEE 9th Workshop on*, pp. 107–110, Oct 2007.

[13] K. M. Elliadka and R. Morelos-Zaragoza, "Precoding by priority: A UEP scheme for raptorq codes," *CoRR*, vol. abs/1402.4246, 2014.

[14] H.-F. Hsiao and Y.-J. Ciou, "Layer-aligned multipriority rateless codes for layered video streaming," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 24, pp. 1395–1404, Aug 2014.

[15] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the h.264/avc standard," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 17, pp. 1103–1120, Sept 2007.

[16] M. G. Luby, M. Mitzenmacher, and M. A. Shokrollahi, "Analysis of random processes via and-or tree evaluation," in *In Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 364–373, 1998.

[17] K. T. Wallenius, *Biased sampling: The non-central hypergeometric probability distribution.* PhD thesis, Stanford University, 1963.

[18] J. Xiao, T. Tillo, and Y. Zhao, "Real-time video streaming using randomized expanding reed-solomon code," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 23, pp. 1825–1836, Nov 2013.

[19] Q. Cheng and D. Agrafiotis, "End to end video distortion estimation with advanced error concealment considerations," in *Visual Communications and Image Processing Conference, 2014 IEEE*, pp. 303–306, Dec 2014.