ABSTRACT

NAYAK, DEBANJANA. An Evaluation of Video Traffic Models for 3D Video. (Under the direction of Harry Perros.)

Over the past few years, video-based applications have become immensely popular. We have witnessed a phenomenal increase in video streaming, such as Amazon Instant Video, Netflix, Hulu, and YouTube, and in video calling services, such as, Skype and Facetime. Large businesses and organizations regularly utilize video conferencing applications such as Cisco's TelePresence and WebEx for face-to-face collaborations across various geographic regions. In addition, the ever-increasing population of multimedia users has resulted in an exponential increase of bandwidth requirements. Consumers are more conscious of and demand a higher quality and performance of video-based products and, therefore, there is a compelling need for continuous improvement in the field of multimedia technologies and communication infrastructure.

In order to deploy an efficient and reliable transport network, we need to understand the traffic characteristics associated with the various multimedia services that run on top of the network. Specifically, accurate traffic models of video are required in order to dimension the capacity of a network, since forecasts show that video will dominate the Internet traffic. A video traffic model is a stochastic model that generates the size of each successive encoded video frame. Such video traffic models are used in performance evaluation studies for dimensioning a network, decide how much bandwidth should be allocated to a video stream so that the required QoS metrics are met, and create synthetic loads for real-life network testing.

A large variety of stochastic models have been proposed for various different types of video such as MPEG-1, MPEG-2, H.264 AVC and H.264 SVC, etc. over a period of several

years. However, there is a lack of traffic models for three-dimensional (3D) video, which is becoming popular. Multiview encoded video is used to support 3D video applications. The statistical characteristics of multiview video are significantly different from the traditional single-view video and therefore existing video traffic models are no more applicable for this type of video. To the best of our knowledge, there are only two models for multiview video that have been proposed in the literature, namely, the Poisson Hidden Markov Model (P-HMM) due to Rossi, Chakareskia, Frossard and Colonnese, and the Markov Modulated Gamma (3D-MMG) model due to Tanwir and Perros.

In this thesis, we implemented and compared the two models using several traces from the Arizona State University's video traces library. The implementation of the P-HMM model was by no means trivial. The comparisons involve Q-Q plots and the autocorrelation function of the frame sizes along with QoS metrics of the resulting packet traces estimated by simulation. The comparison results show that the model generated traces for both the models are quite similar to the original trace in terms of frame size distribution and ACF, though the P-HMM model slightly under-estimates the frame size distribution. The 3D-MMG model predicted the three QoS metrics: 95^{th} percentile of the end-to-end delay, jitter and packet loss rate closer to the actual trace as compared to the P-HMM model. On the other hand, the P-HMM model was better able to predict traffic intensity. Also, the complexity and execution time of the P-HMM model is very high compared to the 3D-MMG model.

An Evaluation of Video Traffic Models for 3D Video

by
Debanjana Nayak

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Computer Science

Raleigh, North Carolina

2015

APPROVED BY:

Harry Perros Committee Chair	David Thuente

Lina Battestilli

BIOGRAPHY

The author was born and brought up in Kolkata, India. After completing her school work from the same city, she joined and eventually graduated from the West Bengal University of Technology, West Bengal, India. She has also worked, as a software developer, for Tech Mahindra Ltd., a renowned IT firm. Currently, she is pursuing her Master's degree from the North Carolina State University, Raleigh, USA. Throughout her academic and professional career, she has received several accolades and awards. She is a cheerful and fun-loving person, who loves to meet new people and travel to new places. She is quite fond of reading and cooking. She loves animals and volunteers for a few animal shelters from time to time.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Over the past few years, video applications have become immensely popular. In today's world, a typical viewer receives digital television with high definition services and a multitude of channels to choose from. We have witnessed a phenomenal increase in the popularity of high definition DVDs, Blue-ray discs and video streaming, such as Amazon Instant, Netflix and Hulu. An enormous portion of the modern population uploads and downloads videos via sites like YouTube and many more join them every day. Recording and sharing of videos using mobile phones is prevalent. There is an increasing demand of video calling services over the Internet through applications like Skype and Facetime. Large businesses and organizations frequently utilize video conferencing applications such as Cisco's TelePresence [34] and WebEx [38] for face-to-face collaborations across various geographic regions.

The ever-increasing population of multimedia users has resulted in an exponential increase of bandwidth requirements. Consumers are more conscious of and demand a higher quality and performance of video-based products and, therefore, there is a compelling need

for continuous improvement in the field of multimedia technologies and communication infrastructure.

In order to deploy an efficient and reliable transport network, we need to understand the traffic characteristics associated with the various multimedia services that run on top of the network. Specifically, accurate traffic models of video are required in order to dimension the capacity of a network, since forecasts show that video will dominate the Internet traffic.

Network capacity is typically designed to allow a target level of Quality of Service. End-to-end delay, jitter and packet loss rate are the most important parameters typically used to evaluate the QoS of a network. An ideal scenario for dimensioning a network is to carry out live experiments over real networks and real video sources. However, such a process is quite expensive and the results obtained may not be extended to more general cases. The alternative approach would be to model a network using modeling techniques, such as mathematical analysis and simulation, which necessitates accurate traffic models. Trace-driven traffic models are popular as they present an actual traffic load, but they are static and they provide only a point representation of the workload space [31]. Another disadvantage of using traces is the difficulty in adjusting parameters and extending the trace if there is a need to continue the simulation beyond the number of packets in the trace file. On the other hand, statistical and mathematical traffic models provide a superior understanding of various traffic characteristics, because of their stochastic nature. Consequently, different realizations that may represent different scenarios can be studied by varying the different model parameters. In other words, a traffic trace provides insight on a particular traffic source, but a traffic model sheds light on different traffic sources of a similar type.

Before an in-depth understanding of 3D video traffic models can be achieved, one must have a working knowledge of 2D and 3D video coding. Section 1.1 provides a short description of the video coding process, while section 1.2 discusses some of the popular video coding standards. Section 1.3 presents a brief survey of the various video traffic models that have been researched and published till date. Because of their prevalence, variable bit rate (VBR) video traffic models have been focused here primarily. Section 1.4 presents a brief description of Markov processes and Hidden Markov Models (HMM), which form the basis of both the video traffic models for 3D video, proposed in the literature so far. Finally, section 1.5 provides a brief summary of the 3D video traffic models that have been proposed and published till date and forms the main topic of discussion for the thesis.

1.1 Video Coding

Video coding is the process of reducing the amount of data required to represent a digital video signal, prior to transmission or storage. Video data may be represented as a series of still image frames. The sequence of frames contains spatial and temporal redundancy. Video encoding algorithms take advantage of this redundancy to compress video, encoding only the difference between frames.

A video encoder consists of three main functional units: a prediction model, a spatial model and an entropy encoder [27]. A block diagram of a video encoder is shown in figure 1.1. The input to the prediction model is an uncompressed (raw) video sequence. The goal of

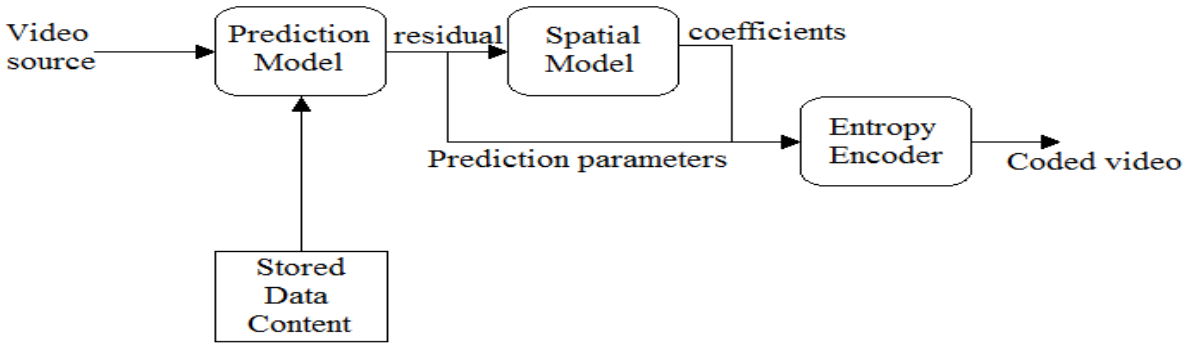


Figure 1.1: Block diagram of an encoder

the prediction model is to reduce redundancy by forming a prediction of the data and subtracting this prediction from the current data. There are two types of prediction models: temporal prediction and spatial prediction. In temporal prediction, the prediction is formed from previously coded frames while spatial prediction uses the previously coded image samples within the same frame. The output of the prediction model is a residual frame, created by subtracting the prediction from the actual current frame, and a set of model parameters indicating the intraprediction type or describing how the motion was compensated, also known as the motion vector.

The residual frame is given as an input to the spatial model that makes use of the similarities between local samples in the residual frame to reduce spatial redundancy. Transform coding, such as discrete cosine transform (DCT) is widely used for reducing the spatial redundancy in video coding. The transform converts the residual samples into another

domain in which they are represented by transform coefficients. The coefficients are quantized to remove insignificant values, leaving a small number of significant coefficients that provide a more compact representation of the residual frame. The output of the spatial model is a set of quantized transform coefficients.

The parameters of the prediction model, i.e. the intraprediction modes or the interprediction modes and motion vectors, and of the spatial model, i.e. the transform coefficients, are further compressed by the entropy encoder. The entropy encoder removes statistical redundancy in data. For example, it represents commonly occurring vectors and coefficients with short binary codes. Variable length coding such as the Huffman coding and arithmetic coding are two commonly used entropy coding techniques. The entropy encoder produces a compressed bit stream that may be transmitted or stored. A compressed sequence consists of coded prediction parameters, coded residual coefficients and header information.

Once the data are compressed, the bit stream is packetized and sent over the Internet. The video decoder reconstructs a video frame from the compressed bit stream. The coefficients and prediction parameters are decoded by the entropy decoder after which the spatial model is decoded to reconstruct the residual frame. The decoder uses the prediction parameters together with previously decoded image pixels to create a prediction of the current frame and the frame itself is reconstructed by adding the residual frame to this prediction.

1.2 Video Coding Standards

Many different techniques for video coding have been proposed and researched. Hundreds of research papers have been published describing innovative compression schemes. However, commercial video applications use a limited number of standardized techniques for video compression, which simplify interoperability between different manufacturers. Various standards have been developed for video encoding, such as, H.261, H.263, MPEG-1, MPEG-2, MPEG-4 and H.264.

The Moving Picture Experts Group (MPEG) was started in 1988 as a working group with the aim of defining standards for digital compression of audiovisual signals. MPEG has produced several standards for video compression which run over IP.

The MPEG layer hierarchy, shown in figure 1.2, consists of six different layers. A block is an 8×8 matrix of pixels or corresponding DCT information that represents a small chunk of brightness (luma) or color (chroma) within the frame. To ease the computation, four blocks (16×16 pixels) are grouped together to form a macroblock. A single row of macroblocks in a video frame is called a slice. These slices are then grouped to form a video frame or a picture. Successive video frames are considered together as a group of pictures (GOP), which represents an independent unit in the video scene. The sequence layer is comprised of a sequence of GOPs. A sequence layer can be thought of as a video scene or shot.

The MPEG-2 coding standard has three frame types that are organized in several possible ways within a GOP. Intra, or I, frames carry a complete video picture. They are

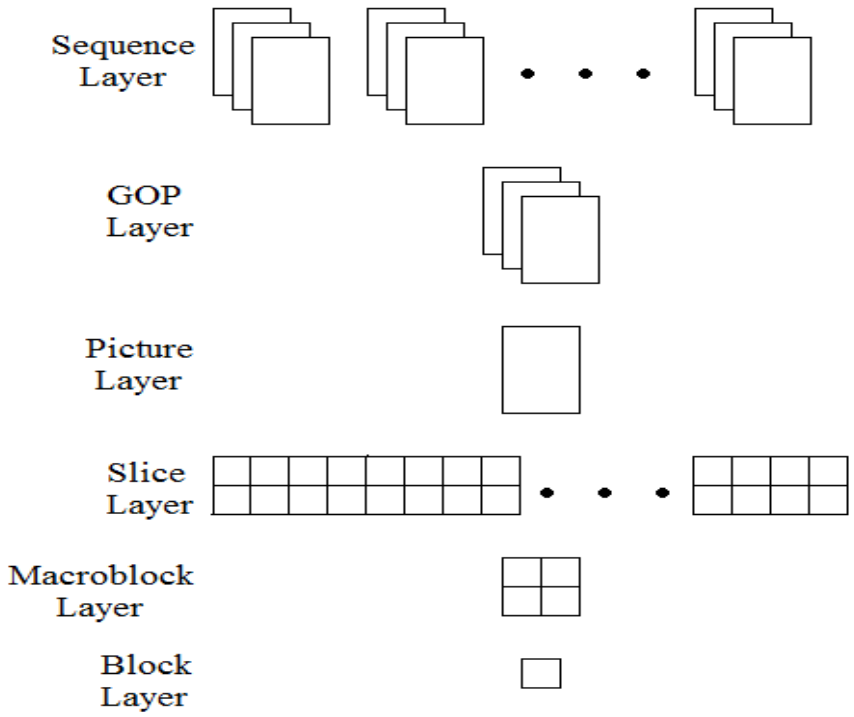


Figure 1.2: The MPEG video hierarchy

coded without reference to other frames and might use spatial compression but do not use temporal compression. That is, no information from other frames is used in the compression. Therefore, this frame can be decompressed even if other frames in the GOP are lost. Predictive-coded, or P, frames predict the frame to be coded from a preceding I or P frame using temporal compression. P frames can provide increased compression compared to I frames, with a P frame typically 20-70% the size of an associated I frame [12]. Finally, bidirectionally predictive-coded, or B, frames use the previous and next I frame or P frame as

their reference points for motion compensation. B frames provide further compression, which is typically 5-40% the size of an associated I frame.

The frame order within a GOP depends on the interrelationships between frames, i.e. which frame is used as the reference for another frame. The interdependencies between the frames of a GOP are shown in figure 1.3. The I frame provides direct reference for the B frames immediately preceding it. It also provides reference for the first P frame in the GOP. As a result, the I frame directly or indirectly is the source of all the temporal encoding within a GOP. The first P frame within a GOP takes reference from the I frame; subsequent P frames take reference from the preceding P frame. Finally, a B frame takes bidirectional prediction from both P and I frames. Many possible GOP structures exist and depend on the source video signal's format or on any bandwidth constraints on the encoded video stream (which determine the required compression ratio), and possible constraints on the encoding or decoding delay.

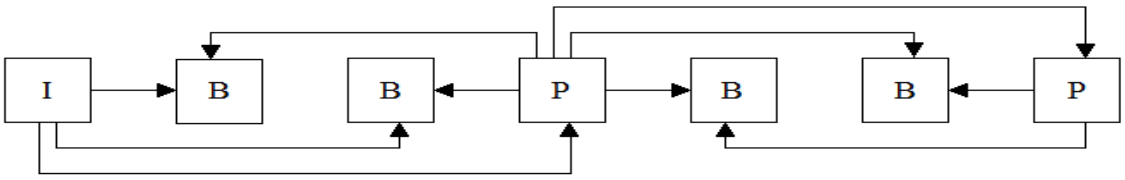


Figure 1.3: Interframe dependencies within a GOP

Due to dependencies between frames, their display order is not the same as their transmission order. Figure 1.4 shows the encoder input order or decoded display order. In figure 1.4(a), frame I_1 provides reference for frame P_4. In turn, both frames I_1 and P_4 provide reference for frames B_2 and B_3. To decode the B frames, the decoder must have already decoded the associated reference frames. So, the decoder will need to receive and decode frames I_1 and P_4 before it can decode frames B_2 and B_3. Figure 1.4(b) shows the corresponding transmission order.

B_{14} B_{15} I_1 B_2 B_3 P_4 B_5 B_6 P_7 B_8 B_9 P_{10} B_{11} B_{12} P_{13} B_{14} B_{15} $I_{New\ GOP}$

(a)

I_1 B_{14} B_{15} P_4 B_2 B_3 P_7 B_5 B_6 P_{10} B_8 B_9 P_{13} B_{11} B_{12} $I_{New\ GOP}$ B_{14} B_{15}

(b)

Figure 1.4: GOP encoding and transmission orders

H.264/MPEG-4 advance video coding (AVC) represents a big leap in video compression technology with typically a 50% reduction of the average bit rate for a given video quality compared to MPEG-2. Block transforms in conjunction with motion compensation and prediction are still the core of the encoder as in previous standards, but a

number of new encoding mechanisms have been added, which improved the performance significantly. The most important of these mechanisms, germane to the purpose of the discussion here, is perhaps the introduction of Hierarchical B frames. Hierarchical B frames are an important new concept that was first introduced in H.264 AVC using generalized B frames and was later found to be the best method to build the scalable video coding (SVC) extension. In the classical B frame prediction structure, each B frame is predicted only from the preceding I or P frame and from the subsequent I or P frame. Other B frames are not referenced since this is not allowed by video standards. The hierarchical B frame structure allows B frames for the prediction of B frames. This can be seen in figure 1.5.

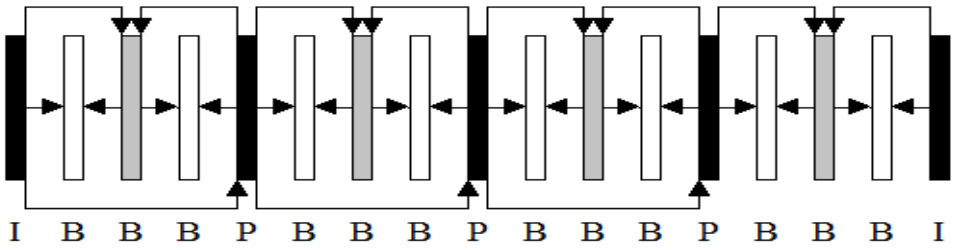


Figure 1.5: Hierarchical B frames

1.2.1 H.264 MVC (Multi-View Coding)

Three-dimensional (3D) video has become very popular during the last few years due to the advancement in display technology, signal processing, circuit design and networking infrastructure. One common characteristic of many of these systems is that they use multiple camera views of the same scene, often referred to as multiview video (MVV), which is implemented by simultaneously capturing the video streams of several cameras. Since this approach creates large amounts of data to be stored or transmitted to the user, efficient compression techniques are essential for realizing such applications. An easy solution for this would be to encode all the video signals independently using a video codec such as H.264 AVC. However, MVV contains a large amount of inter-view statistical dependencies, since all cameras capture the same scene from different viewpoints.

The basic approach of most multiview coding (MVC) schemes is to exploit not only the redundancies that exist temporally between the frames within a given view, but also the similarities between frames of neighboring views. However, it is required for the compressed multiview (MV) stream to include a base view bit stream, which is coded independently from all other views and is compatible with decoders for single-view profiles. This allows the two-dimensional (2D) version of the content to be easily extracted and decoded. For example, in television broadcast, legacy receivers should be able to extract and decode the base view, while newer 3D receivers can decode the complete 3D bit stream.

Figure 1.6 shows the frame prediction structure for MVC video that consists of two views: left and right. The left view, being the base view, is coded independently from the

right view. The right view is coded with reference to the left view. Each view is transmitted as a separate video stream. The MV encoder receives N temporally synchronized video streams and generates one bit stream. The MV decoder receives the bit stream, decodes and outputs the N video signals.

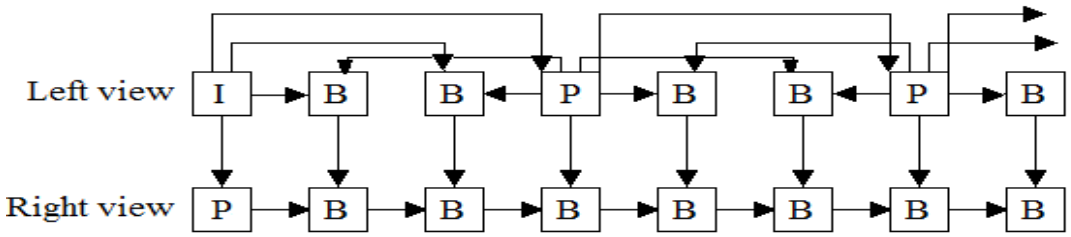


Figure 1.6: Inter-view prediction in MVC [36]

1.3 Video Traffic Modeling

A video traffic model is a stochastic model that generates the size of each successive encoded video frame. The parameters of the model are obtained from a given frame trace. Video traffic models are used in performance evaluation studies to dimension a network, decide how much bandwidth should be allocated to a video stream so that the required QoS metrics are met, and create synthetic loads for real-life network testing.

A large number of video traffic models have been proposed in the literature in the last 20 years, focusing on different types of videos and encoding formats and using different modeling techniques. A survey of some of the variable bit-rate (VBR) video traffic models, pertinent to our discussion here, can be found in [32] [33]. The authors in [33] classified the VBR video traffic models into the following five groups:

I. Autoregressive models

II. Models based on Markov processes

III. Self-similar and fractional autoregressive integrated moving average models

IV. Wavelet models

V. Other approaches

I. *Autoregressive (AR) models:* In such models, the next frame size in a video sequence is predicted as an explicit function of p previous ones. These video traffic models use different types of AR processes based on the type of video and encoding scheme (viz. I/B/P frames). In general, they appear to capture the autocorrelation behavior of compressed video, which is an essential element when modeling compressed video sources. The coefficients of these models are simple to estimate from empirical data. However, it is not possible to find a single AR model that can capture different statistical characteristics. Hence, there is no single video model that is suitable for all video sequences and all purposes. Similar observations were made in the survey papers on AR models [1] [2].

II. *Models based on Markov processes:* In such models, Markov processes/chains are used to represent bit rate regimes or frame/GOP sizes. Markovian models appear to be more accurate than AR models. There are two main factors that distinguish these models from each

other. First factor is the modulated process, which is dependent on the frame size distribution of the video traffic. Authors have used many different types of frame size distributions, i.e. gamma, AR, Bernoulli, normal, lognormal, geometric and uniform. The second factor that differentiates these models is how the state space of the Markov process is determined and how the transition probabilities are calculated. Some models use bit rate or frame size ranges for determining the number of states, while others use scene activity levels. In all cases, the states are dependent on the bit rate variation in the video traffic. However, the same transition probabilities cannot be used for all types of video. The transition probabilities are very much dependent on the type of video traffic. A wide variety of Markovian models exist in the literature for different types of video and for different video coding standards. Markov-modulated models have attracted much attention and research over the years [4] [6] [17] [18]. Perhaps the most successful model in this category is a Markov-modulated gamma model [30], which has also formed the basis of one of the 3D video traffic models that form our discussion here.

III. *Self-similar and fractional autoregressive integrated moving average (FARIMA) models*: A process is said to be self-similar if the observations for that process appears "similar" regardless of the duration of sampling interval. Such models capture the long-range dependence (LRD) of compressed video traffic. LRD is the phenomenon where observations of an empirical record are significantly correlated to observations that are far removed in time. In this category, the fractional ARIMA or FARIMA models [11] [14] [16] have received much attention, while we also have a discrete-time statistically self-similar system [23]. The ACF structure of VBR video traffic exhibit both LRD and SRD properties. If the

video is highly correlated for a large number of lags, then a model that captures LRD like self-similar process may be a good option. The main drawback of such models is their computational complexity. Also, they fail to capture the SRD properties of video traffic.

IV. *Wavelet models:* In such models, wavelet transform techniques are used to capture both LRD and short-range dependence (SRD) properties of video traffic. Only a few models have been published in this category, e.g. a wavelet/AR(1) model [37], a simple wavelet model [19], a hybrid wavelet/time domain model [7], etc. It appears that wavelet-based models are good for modeling the coexistence of SRD and LRD behavior in video traffic. These results show that a key advantage of using wavelets is their ability to reduce the complex temporal dependence so significantly that the wavelet coefficients only possess the SRD. However, such models require understanding of digital signal processing tools and techniques. Also, they require several trials to optimize the different parameters involved.

V. *Other approaches:* Different models such as the seasonal ARIMA (SARIMA) model [31], M/G/∞ process [15] and the transform-expand-sample (TES) models [13] [20] [21] [22] fall in this category. The SARIMA is an extension of the ARIMA model used for series that exhibit periodic or seasonal behavior. Such models allow easy adjustments of traffic parameters. The M/G/∞ process, though not Markovian, exhibits SRD and captures the marginal distribution of a video sequence. The TES-based models utilize the TEStool, which is an interactive software environment for modeling autocorrelated time series using a class of stochastic processes called TES. TES processes are designed to fit both the marginal distribution and the ACF of the empirical data simultaneously. The main drawback of all

TES-based models is that they require access to the TES tool and they have high computational complexity.

1.4 Hidden Markov Models (HMM)

In this section, we briefly review Hidden Markov Process (HMM) used in one of the two video models that were evaluated and compared in this thesis. A stochastic process $\{X_t\}$, $t = 1, 2, 3, \dots$, with state space $S = \{1, 2, 3, \dots, N\}$ is Markovian if for every n and all states $s_1, s_2, \dots$ where $s_n \in S$, it satisfies the Markov property:

$$P[X_n = s_n | X_{n-1} = s_{n-1}, X_{n-2} = s_{n-2}, \dots, X_1 = s_1] = P[X_n = s_n | X_{n-1} = s_{n-1}]$$

In simple words, the current state of a Markov process depends only on its previous state, and not on previous states. A Markov process switches from one state to another following a set of transition probabilities associated with the first state, and may even make a transition back to the same state as previous. These transition probabilities are usually represented through the transition matrix P, whose each element p_{ij} gives the probability that the Markov process switches from state s_i to s_j. If we denote the actual state at time t as q_t, then we can define p_{ij} as,

$$p_{ij} = P\left[q_t = s_j \middle| q_{t-1} = s_i\right], \qquad 1 \leq i, j \leq N, \qquad \text{such that,}$$

$$p_{ij} \geq 0 \text{ and } \sum_{j=1}^{N} p_{ij} = 1.$$

The above stochastic process is an *observable* Markov model since the output of the process at each instant of time is a state that is observable. On the other hand, in a Hidden Markov Model (HMM), the actual state that the system occupies at each time t is hidden, and instead we can only see an observation that is a result of the state the system is in. An HMM is a doubly embedded stochastic process with an underlying Markov process that is not observable or hidden, but can only be observed indirectly through a state-dependent stochastic process that produces the sequence of observations. The underlying Markov process of an HMM is typically a discrete process, i.e., a Markov chain. A detailed study of the theoretical aspects of this type of statistical modelling has been presented in [26].

An HMM is characterized by the following elements:

i.) *The number of states in the model, N*. Although the states are hidden, for many practical applications there is often some physical significance attached to the states or to the sets of states of the model. For example, in case of video traffic models states are used to represent ranges of bit rates or ranges of frames or GOP sizes of a video sequence. Generally, the states are interconnected in such a way that any state can be reached from any other state, in a single step. However, other interconnections of states are also possible and often become necessary. For many real-life applications, often connections do not exist between two states, due to some physical constraint.

ii.) *The number of distinct observation symbols per state M.* The observation symbols correspond to the physical and observable output of the system being modeled. As stated earlier, in case of video traffic models states are used to represent ranges of bit rates or ranges of frames or GOP sizes of a video sequence. Thus, each frame size or GOP size from the corresponding range may be considered as an observation symbol for that state. In many applications, the observations may not be discrete, but continuous, and can be represented using processes such as Bernoulli, Poisson, AR, gamma, etc.

iii.) *The one-step transition matrix P.* Similar to general Markov model, the transitions between the various states of an HMM are also described by the state transition matrix, notated previously as $P = (p_{ij})$, where $p_{ij} \geq 0$.

iv.) *The observation symbol probability distribution $b_j(k)$.* This the probability distribution that we will observe the k^{th} symbol, denoted by v_k, when the HMM is in state s_j, that is,

$$b_j(k) = P\left[v_k \text{ at } t \middle| q_t = s_j\right], \qquad 1 \leq j \leq N, \qquad 1 \leq k \leq M.$$

We define the matrix B as $B = \{b_j(k)\}$. As discussed earlier, the observation symbol probability distribution may be discrete or continuous.

v.) *The initial state distribution $\pi = \{\pi_i\}$.* This is the probability that the system is in state i at time t=1, i.e.,

$$\pi_i = P[q_1 = s_i], \qquad 1 \leq i \leq N$$

Often, for convenience, the following compact notation is used to indicate the complete parameter set of the model.

$$\lambda = (P, B, \pi)$$

1.4.1 The Three HMM Problems

There are three different, but related, problems that typically arise in real-world applications. They are described briefly below.

Problem 1: Given an observation sequence O_1, O_2, ..., O_T, compute the probability that it came from a given λ (Forward Probability Computation).

Let the system be in state s_i at time t. The probability of the system shifting to state s_j at t+1 is given by the one-step transition probability p_{ij}. The probability that of the M possible observed values (symbols), the k^{th} one is observed given that the system shifted from state s_i to s_j is $p_{ij}b_j(k)$. Since the state s_i can be any one of the states from 1 to N, the probability of observing the k^{th} value, given that the system shifts to state s_j at time t+1 is the sum of all probable paths to state s_j, i.e., $\left[\sum_{i=1}^{N} p_{ij}\right] b_j(k)$.

Now, the only remaining unknown is the joint probability of having observed the sequence from O_1 through O_t and being in state s_i at time t, given λ. Representing this

quantity by $\alpha_t(i)$ and representing the k^{th} observed value at time t+1 by O_{t+1}, we have:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) p_{ij}\right] b_j(O_{t+1}), t \in (1, T-1], j \in [1, N]$$

To start off this induction, the initialization step for calculating $\alpha_t(i)$ is obtained as follows. The probability of the system being in state s_i at time 1 is given by π_i and the probability of observing O_1 is then given by $b_i(O_1)$. Thus,

$$\alpha_1(i) = \pi_i b_i(O_1), i \in [1, N].$$

$\alpha_T(i)$ obtained from the induction step represents the probability of observing the sequence from O_1 through O_T, and ending up in state s_i. Thus, the total probability of observing the sequence O_1 through O_T, given λ is

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Problem 2: Given an observation sequence, compute the probable states the system passed through (Backward Probability Computation).

The quantity $\beta_t(i)$ is defined as the probability that the sequence of observations from O_{t+1} through O_T are observed starting at state s_i at time t for a given λ. It is calculated in the

same way as $\alpha_t(i)$, but in the backward direction. We have:

$$\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) p_{ij}\, b_j(O_{t+1}), t = T-1, \ldots, 1.$$

For t = T we have $\beta_T(i) = 1$, i = 1, 2, …, N. To calculate the probability that the system was in state s_i at time t given O and λ, we observe that $\alpha_t(i)$ accounts for the observation sequence from O_1 through O_t and $\beta_t(i)$ accounts for O_{t+1} through O_T, and both account for the state s_i at time t. So the required probability is given by $\alpha_t(i)\beta_t(i)$. Introducing the normalizing factor from problem 1, $P(O|\lambda)$, we have

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)}$$

At each time t, the state with the highest γ is the most probable state at time t. A better way to obtain the most probable path of states that give rise to an observation sequence O, is to use dynamic programming.

Problem 3: Given an observation sequence O, compute the most probable λ.

The term p_{ij} can be calculated as the ratio of the number of transitions made from state s_i to state s_j over the total number of transitions made out of state s_i. We have from problem 2 that $\gamma_t(i)$ is the probability of being in state s_i at time t. Extending this, the probability of being in state s_i at time t and in state s_j at time t+1 can be calculated as follows:

$$\xi_t(i,j) = \frac{\alpha_t(i)p_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)}$$

1.4.2 Generating an Observation Sequence

Given appropriate values of N, M, P, B and π, the HMM can be used as a generator to give an observation sequence

$$O = O_1, O_2, \ldots, O_T$$

Here, each observation O_t is one of the symbols from the symbol space V and T is the number of observations in the sequence. The observations are created using the following steps:

1. Choose an initial state s_i according to the initial state distribution π. This can be done using the procedure for sampling from an empirical distribution. Suppose we have three states s_1, s_2 and s_3, and let π_1, π_2 and π_3 be respectively the initial state probabilities for the three states. We generate a pseudo-random number r in [0,1]. If $r \leq \pi_1$, then we select s_1. If $\pi_1 \leq r \leq \pi_1 + \pi_2$, then we select s_2. Otherwise, we select s_3.
2. Set t = 1.

3. Set $O_t = v_k$ according to the symbol probability distribution in state s_i, i.e. $b_i(k)$. This can also be done with the help of the sampling procedure described above.
4. Transit to a new state s_j according to the state transition probability distribution for state s_i, i.e. p_{ij}. The new state can again be chosen with the help of the procedure explained above.
5. Set t = t+1. Return to step 3 if t < T. Otherwise terminate the procedure.

1.5 Video Traffic Models for MVC or 3D Video

A large variety of stochastic models have been proposed for various different types of video such as MPEG-1, MPEG-2, H.264 AVC and H.264 SVC etc. over a period of several years. A brief study of these models has been presented in section 1.3. However, there is a lack of traffic models for multiview coding (MVC) video and three-dimensional (3D) video, which is a special case of MVC video. Multiview (MV) video provides several different views of a given scene and it gives viewers the perception of depth.

Although many traffic models have been developed for H.264 AVC and SVC video, there is very little research done on the modeling of MVC video. Previous studies [25] [35] on 3D video transport have mainly concentrated on the encoding formats, network and transport layer protocols and traffic characteristics of 3D video representation formats. To the best of our knowledge, there are only two models for multiview video that have been proposed in the literature so far. The first model, hereafter referred to as Poisson Hidden

Markov Model (P-HMM), is by Rossi et. al., [28], and it is based on a Hidden Markov Model. The second model, hereafter referred to as Markov Modulated Gamma Model (3D-MMG), is by Tanwir and Perros, [33], and it is based on a Markov modulated gamma process.

The P-HMM model is based on a Hidden Markov Model (HMM) where the states of the model correspond to different video activity levels. The duration of a video activity level is assumed to be Poisson distributed. That is, the distribution of the time the Markov chain (MC) spends in each state is not geometric, as is typically the case, but it is Poisson. This type of HMM is referred to in the literature as a Poisson Hidden Markov Model (P-HMM). It was assumed that the different views are correlated and thus they have the same variation in each activity level. That is, the corresponding GOPs of each view belong to the same state of the MC.

As in the P-HMM model, the 3D-MMG model also assumes that there is a strong correlation between the frames of the left view and the corresponding frames in the right view. Thus, the right view cannot be modeled independently. Otherwise, the correlation between the views will be lost. As a result, the base or the left view is modeled independently using the Markov-modulated gamma (MMG) model proposed in [30]. The right view is then modeled with the left view as reference in a manner, very similar to the model for SVC video presented in [39].

In this thesis, both models were implemented and their performances were compared extensively using several traces from the Arizona State University's video traces library. The

comparisons involve Q-Q plots and the autocorrelation function of the frame sizes along with QoS metrics of the resulting packet traces estimated by simulation.

1.6 Organization of the Thesis

The thesis is organized as follows. The two video traffic models that have been published till date for MVC or 3D video are described in detail in chapter 2. The implementation details and the evaluation results are presented in chapter 3. Finally, the conclusion and future research is given in chapter 4.

Chapter 2

Video Traffic Models for 3D Video

In this chapter, we discuss, in detail, both the video traffic models that have been proposed in literature, so far, for multiview video coding (MVC) or three-dimensional (3D) video. For convenience and clarity of discussion, we have divided this chapter into two sections. Section 2.1 reviews the P-HMM model as proposed by Rossi et. al. in [28]; while section 2.2 does the same for the 3D-MMG model by Tanwir and Perros in [33].

2.1 The Poisson Hidden Markov Model (P-HMM)

In [28], the authors have come up with a new stochastic model that characterizes traffic generated by a multiview video coding (MVC) variable bit rate (VBR) source. It is a Poisson Hidden Markov Model (P-HMM) with two stochastic layers – the first layer consists of a non-stationary chain that models the evolution of the video activity and the second layer represents the frame sizes of each of the MVC encoded views. The model is quite complex

and in order to maintain lucidity of discussion, this section has been further divided into the following sub-sections. Section 2.1.1 provides a review of the traffic model and its underlying concept. Section 2.1.2 discusses the different parameters involved in the model and their estimation procedure. Section 2.1.3 describes the implementation of the model.

2.1.1 Traffic Model

The goal of the proposed model is to be able to characterize the frame sizes of an MVC compressed stream, given its GOP structure. Although different rate control algorithms [24] can be implemented in an H.264 MVC encoder [5], in this paper the authors concentrate on Variable Bit Rate (VBR) traffic only, due to their widespread use. In VBR operating mode, the bit-rate of the MVC encoded streams depends on the corresponding video activity level and varies accordingly. Consequently, the model should be able to replicate the stochastic non-stationary nature of the process. The authors have assumed that the video activity level varies in time according to a Poisson distribution, i.e. the amount of time for which the video activity level stays in a particular range can be described by a Poisson distribution. As mentioned earlier, the paper introduces a Poisson Hidden Markov Model (P-HMM) with two stochastic layers – the first (hidden) layer is a discrete time Markov chain whose states represent different video activity levels and the second layer represents the frame size sequence corresponding to a given activity level. The authors state that their model is heavily inspired by [9] [29].

The authors assumed that the activity level of the video content always changes at the beginning of a GOP. That is, the transitions from one state of the HMM to another can only occur when a new GOP starts. In real MVC encoded traces, however, the activity level may change at any point of a GOP. This is a common assumption and it is also made in the 3D-MMG model reviewed in this chapter. Views are correlated and thus they have the same variation in activity levels. In view of this, the corresponding GOPs of both views belong to the same state of the HMM.

To comprehend the model better, let us denote the number of traffic sequences generated from all the views as N_{view}. Typically, since one such sequence is generated from one view, they are numerically equivalent. The number of frames in one GOP of each view is denoted as N_{GOP}. Thus, the total number of frames in one GOP across all the views can be referred as $N_f = N_{view} \times N_{GOP}$. The number of video activity levels, or rather, the number of states in the HMM is denoted as N_s. Since, the first or hidden layer of the model is assumed to be a P-HMM, the amount of time that the model stays in state i is described by a Poisson distribution, with a single parameter λ_i, and it is given by the expression

$$d_i[k] = \frac{e^{-\lambda_i}\lambda_i^k}{k!}$$

where the model is currently in state i and will continue in the same state for the next k GOPs. The transition probabilities that define the model transitions from one state to another is given by the state transition matrix Π, each element of which is denoted by π_{ij}, which is the

probability that the model will transit from state i to state j. In a traditional HMM, the model may stay in the same state i for a geometrically distributed number of slots. However, in a P-HMM, the state duration follows a Poisson distribution and the model never returns to the same state after transition out of the state. Consequently, in this model, the state transition matrix Π has elements $\pi_{ii} = 0$ for every state i.

The second layer of the model characterizes the frame sizes of an entire GOP, with respect to the state it currently pursues in the first layer. Let us consider that the following random vector describes the set of frame sizes of every frame in the n^{th} GOP of the compressed stream.

$$x[n] = [x_0[n], \ldots\ldots, x_{N_f - 1}[n]]$$

The vector x[n] is generated by a multivariate probability mass function (pmf), which, in turn, depends on the current state of the hidden layer of the model, i.e., if the first layer of the model is in state i, the set of frame sizes of the corresponding GOP can be described by the vector x[n], generated according to the pmf $b_i[x[n]]$, i = 1, . . . , N_s. A GOP can be imagined as a series of N_f frames, each occupying its designated position as defined by an MVC encoder [5]. The frame size of each such frame is described by a separate pmf b_i [·], where [·] gives the position of the frame in the GOP. Each pmf b_i [·] comprises of a different number of bins depending on the frame (namely I, P, or B frame) of the compressed picture to be generated. The authors explain that such a design is motivated by the attempt to avoid imposing a fixed probability distribution (e.g., Gaussian, gamma, etc.) for the frame sizes,

but instead to adapt the distribution shape to the actual MVC sequences. Moreover, the decision to use a different number of bins for different frame types helps adaptively control the model's complexity (i.e., the parameter set) according to the desired accuracy performance.

2.1.2 Parameter Estimation

The above-described model is extremely complicated, with a lot of parameters in play, and hence their estimation forms a crucial part of the model. The model is a member of the wide HMM family [26], for which several popular parameter estimation algorithms are available. However, the most widely used estimation procedure, in case of HMMs, is perhaps the Expectation-Maximization (EM) algorithm [8]. The version of the EM algorithm proposed in [29] is suitable for P-HMMs, but displays numerical instability when applied to long data sequences [10]. In this paper, the authors claim to extend the method of [10] to the case of non-stationary hidden state durations. The version of the EM procedure, employed in this case, enables stable parameter estimation for long data sequences. This model is heavily influenced and closely follows the deductions and derivations in [26], for inclusion of explicit state duration density in HMMs. The authors have, however, adapted the method of [26] to accommodate a Poisson density of state durations.

Let us now discuss the algorithm in view of video traffic characterization. Suppose the observed video traffic consists of N GOPs. Hence, it can be represented as $y_0^{N-1} \stackrel{\text{def}}{=}$

$\{y[n]\}_{n=0}^{N-1}$, where y[n] denotes the set of frame sizes in the n^{th} GOP. Let θ denote the parameter set of the P-HMM model, defined as $\theta \stackrel{def}{=} \{\Pi, \lambda_1, b_1[\cdot], \pi_1, \ldots, \lambda_{N_s}, b_{N_s}[\cdot], \pi_{N_s}\}$, where π_i denotes the probability that the model will begin in state i. The algorithm itself consists of two iterative computational steps. The first step, also called the Expectation step, computes the auxiliary likelihood function $Q\left(\theta\middle|\theta^{(m)}\right) = E\left\{\log(\text{Prob}\{S, x, \theta\}) \middle| y, \theta^{(m)}\right\}$, where $\theta^{(m)}$ represents the m^{th} estimate of the parameter set and S is a plausible sequence of states. The second step, also called as the Maximization step, maximizes the likelihood function, i.e., $\theta^{(m+1)} = \arg\max_{\theta} Q\left(\theta\middle|\theta^{(m)}\right)$. If we consider the three classical problems for HMMs discussed in chapter 1, we will realize that we are basically dealing with the third problem. The EM algorithm initially assumes certain values for each of the parameters. Then, it executes the Expectation step, where it estimates the probabilities of all probable sequences of states, which means it needs to calculate $\gamma_t(i)$ and $\xi_t(i, j)$. After that, it goes on to the Maximization step where it re-estimates the parameters based on the probabilities calculated during the Expectation step. The algorithm continues to iterate between the two steps until convergence of the parameter set is achieved.

Before we describe the algorithm in detail, let us first enumerate the commonly used variables in the EM algorithm, employed in this paper. In the first step, the model needs to be trained on an observed video traffic sequence of a known number of GOPs N. The GOPs are numbered as n = 0, 1, 2,…, N-1, where n represents the n^{th} GOP. The model views each GOP as a unique time instant, such that the traffic sequence is of length T. If t is the t^{th} time instant, then n=0 corresponds to t=0, n=1 to t=1, n=2 to t=2, and so on until n=N-1to t=T-1.

Another interesting variable is k, which simply denotes that the model will stay in the current state for k future consecutive time instants or GOPs. Consequently, the model continues in the current state for k+1 consecutive time instants or GOPs, starting from the present instant.

The model also requires, as an input, the number of states present in the traffic sequence being used to train the model, viz. N_s. Here, each state represents a level of video activity, which is essentially the average frame size of the frames considered to be in that state. The states are again numbered as i = 1,2,…, N_s, where i represents the i^{th} state.

Another important parameter is $b_i[y[n]]$, which gives the probability that the n^{th} GOP of the observed sequence is in state i. Now, as discussed earlier, a GOP can be imagined as a series of N_f frames, each occupying its designated position as defined by an MVC encoder [5]. Hence, $b_i[y[n]]$ becomes the joint probability that each frame in the n^{th} GOP is in state i, where $b_i[y[n]] = b_i[f_{n_1}] \times \ldots \times b_i[f_{n_{N_f}}]$, where f_{n_x} is the frame in the n^{th} GOP occupying the x^{th} position in the GOP. That is, the frame size distribution of each frame is assumed to be independent of the other frame sizes in the same or adjacent GOPs.

The computations of the EM algorithm, employed in this case, can be reduced to three steps as follows:

(i) the computation of forward probabilities, namely α's

(ii) the computation of backward probabilities, namely γ's and ξ's

(iii) the estimation of the parameter set

The forward and the backward probabilities, mentioned above, are essentially the conditional probabilities of the state sequence given the observed sequence of frame sizes. Steps (i) and

(ii) correspond to the Expectation step and step (iii) corresponds to the Maximization step of the EM algorithm.

2.1.2.1 Poisson State Duration Distribution

Earlier, we discussed the nature of the state durations, which follow a Poisson distribution, with parameter λ_i, i = 1, 2, … , N_s, i.e., $d_i[k] = \frac{e^{-\lambda_i}\lambda_i^k}{k!}$. In an ideal case, the model has to be trained on an infinite traffic sequence, so that no restriction on time and hence on the variable k is imposed. But in reality, any video traffic has a finite length, which restricts k to a maximum value, viz., $k_{max} = N - n - 1$, after the n^{th} GOP. Consequently, $d_i[k]$needs to be modified accordingly and it may be rewritten as

$$d_i[k] = \begin{cases} 1 - D_i[k-1] & k = N - n - 1 \\ \frac{e^{-\lambda_i}\lambda_i^k}{k!} & k < N - n - 1 \end{cases}$$

where $D_i[k]$ denotes the cumulative distribution of the state duration time when in state i.

2.1.2.2 Calculation of the Forward Probabilities

The authors defined the following two forward probabilities, $\alpha_n(i,k)$ and $\alpha_n(i)$, for n = 0, 1, 2, … , N-1.

$$\alpha_n(i,k) \stackrel{\text{def}}{=} \begin{cases} P\left(s_n = i, \dots, s_{n+k} = i, s_{n+k+1} \neq i \middle| y_0^n, \theta^{(m)}\right), & k < N-n-1 \\ P\left(s_n = i, \dots, s_{N-1} = i \middle| y_0^n, \theta^{(m)}\right), & k = N-n-1 \end{cases},$$

where s_n denotes the state occupied by the model in the n^{th} GOP. Thus, $\alpha_n(i,k)$ essentially describes the probability that the model stays in the i^{th} state from the n^{th} GOP until the $(n+k)^{th}$ GOP and then makes a transition to a different state. The obvious exception is the case when n + k = N – 1 or, $k = N-n-1$. In this case, the traffic sequence comes to an end after the $(n+k)^{th}$ GOP and there is no transition after that. $\alpha_n(i,k)$ is conditioned on the observed sequence from the beginning until the n^{th} GOP and the m^{th} estimate of the parameter set.

$$\alpha_n(i) \stackrel{\text{def}}{=} P\left(s_n = i \middle| y_0^n, \theta^{(m)}\right),$$

where s_n denotes the state occupied by the model in the n^{th} GOP. Thus, $\alpha_n(i)$ essentially describes the probability that the model is in the i^{th} state when in the n^{th} GOP, given the observed sequence from the beginning until the n^{th} GOP and the m^{th} estimate of the parameter set.

Both the α's are being calculated using Algorithm 1, described here.

Algorithm 1: Computing the forward probabilities $\alpha_n(i,k)$ and $\alpha_n(i)$

1: for n = 0 → N - 1 do

2: for i = 1 → N_s do

3: for k = 0 → N - n - 1 do

4: if n = 0 then

5: $\alpha_0(i,k) \propto \pi_i d_i[k] b_i[y[0]]$

6: else

7: $\alpha_n(i,k) \propto b_i[y[n]](\alpha_{n-1}(i,k+1) + \sum_{\substack{j=1, \\ j\neq i}}^{N_s} \alpha_{n-1}(j,0)\pi_{ji} d_i[k])$

8: end if

9: end for

10: end for

11: Normalize all the $\alpha_n(i,k)$'s using the formula given below

12: for i = 1 → N_s do

13: $\alpha_n(i) = \sum_k \alpha_n(i,k)$

14: end for

15: end for

The normalization coefficient for $\alpha_n(i,k)$ is calculated by summation over i and k, i.e.,

$$\text{Normalization Coefficient for } \alpha_n(i,k) = \sum_{i,k} \alpha_n(i,k)$$

The above algorithm is pretty straightforward given that one understands the underlying concept. Both the forward and the backward probabilities, presented later on, are calculated in a sequential order. In case of the forward probabilities, we calculate the $\alpha_n(i,k)$ and $\alpha_n(i)$ iteratively for n = 0 first, then we do the same for n = 1 and so on until n = N-1. This is because these probabilities calculated for each GOP are dependent on those calculated for the previous GOP.

Now, each $\alpha_n(i,k)$ can be calculated as

$$\alpha_n(i,k) = P(\text{at the present instant the model is in state i}) \times P(\text{observation, if at the present instant the model is in state i}) \times P(\text{model continues to be in state i for k future consecutive GOPs})$$

The calculation of the $\alpha_n(i,k)$'s can be clearly distinguished into the following two cases.

Case I: (n = 0)

For n = 0, the model is being initialized and there is no prior information available. So, the probability that the model begins in state i and continues in that state for k future consecutive GOPs is given by

$\alpha_n(i,k)$

$= P(\text{at the present instant the model is in state i})$

$\times P(\text{observed traffic sequence, if at the present instant the model is in state i})$

$\times P(\text{model continues to be in state i for k future consecutive GOPs})$

or,

$$\alpha_0(i,k) \propto \pi_i \times b_i[y[0]] \times d_i[k]$$

The symbol $\propto$ means that the left hand side is proportional to the right hand side of the equation and thus to obtain the actual value of $\alpha_n(i,k)$, the right hand side needs to be normalized.

Case II: ($n \neq 0$)

For the rest of the GOPs, the calculation is not so simple. For $n \neq 0$, we have to consider all possible sequences of states that the model might have followed from t = 0 in order to reach the n^{th} GOP. Hence, we have to calculate the summation of the probabilities of all such possible sequences of states in order to obtain $\alpha_n(i,k)$. Instead of calculating this repetitively for each GOP, the algorithm maintains a matrix of $\alpha_n(i,k)$'s and it just needs to lookup the required probabilities, thus reducing its time complexity. Such an algorithm is called a dynamic algorithm. Also, because of the dynamic nature, the α's calculated for each GOP are dependent on those calculated for the previous GOP.

While $n \neq 0$, the calculation of $\alpha_n(i,k)$'s require two major considerations and can again be clearly distinguished into two sub-cases.

Case a:

After the end of the (n-1)th GOP and before the beginning of the nth GOP, the model may have made a transition from a state j to a state i, in which case the probability that the model stays in state i for k future consecutive GOPs is:

$$P\big(\text{for the } (n - 1)^{th} \text{ GOP the model is in state j and does not continue in that state any more}\big) \times P(\text{model transits from state j to state i}) \times P(\text{model continues in state i for k future consecutive GOPs}) = \alpha_{n-1}(j, 0)\pi_{ji}d_i[k]$$

Case b:

The model may have stayed in the same state i for both (n-1)th and nth GOPs, in which case the probability that the model pursues state i for k future consecutive GOPs become

$$P(\text{the model stays in the } (n-1)^{th}, n^{th} \text{ and k future consecutive GOPs in state i}) = \alpha_{n-1}(i, k+1)$$

Hence, for $n \neq 0$,

$\alpha_n(i,k)$

$= P(\text{at the present instant the model is in state i})$

$\times P(\text{observed traffic sequence, if at the present instant the model is in state i})$

$\times P(\text{model continues to stay in state i for k future consecutive GOPs})$

or,

$$\alpha_n(i,k) = P(\text{observed traffic sequence, if at the present instant the model is in state i}) \times (P(\text{Case a}) + P(\text{Case b}))$$

or,

$$\alpha_n(i,k) \propto b_i[y[n]](\alpha_{n-1}(i,k+1) + \sum_{\substack{j=1, \\ j\neq i}}^{N_s} \alpha_{n-1}(j,0)\pi_{ji}d_i[k])$$

The calculation of $\alpha_n(i)$'s is pretty simple. $\alpha_n(i,k)$ is the probability that the model occupies state i during the n^{th} GOP and will continue to do so for k future consecutive GOPs. Hence, $\alpha_n(i)$ is simply the summation of such $\alpha_n(i,k)$'s for all possible values of k. Consequently,

$$\alpha_n(i) = \sum_k \alpha_n(i,k)$$

2.1.2.3 Calculation of the Backward Probabilities

The authors have defined the following two backward probabilities, $\gamma_n(i,k)$ and $\xi_n(i,j,k)$, for n = 0, 1, … , N-1.

$$\gamma_n(i,k) \stackrel{\text{def}}{=} P\left(s_n = i, \dots, s_{n+k} = i, s_{n+k+1} \neq i \middle| y_0^{N-1}, \theta^{(m)}\right),$$
$$n = 0, \dots, N-1; \quad k < N-n-1,$$

where s_n is the state occupied by the model in the n^{th} GOP. Thus, similar to $\alpha_n(i,k)$, $\gamma_n(i,k)$ essentially describes the probability that the model is in the i^{th} state from the n^{th} GOP until the $(n+k)^{th}$ GOP and then makes a transition to a different state, conditioned on the entire observed sequence and the m^{th} estimate of the parameter set.

$$\xi_n(i,j,k) \stackrel{\text{def}}{=} P\left(s_{n-1} = i, s_n = j, \dots, s_{n+k} = j, s_{n+k+1} \neq j \middle| y_0^{N-1}, \theta^{(m)}\right),$$
$$n = 1, \dots, N-1; \quad k < N-n-1,$$

where s_n is the state occupied by the model in the n^{th} GOP. Thus, $\xi_n(i,j,k)$ essentially describes the transition probability of the model from a state i in the $(n-1)^{th}$ GOP to a state j in the n^{th} GOP, where it continues until the $(n+k)^{th}$ GOP and then makes a transition again to a different state, conditioned on the entire observed sequence and the m^{th} estimate of the

parameter set. Since no transition can take place at t = 0, or at the beginning of the traffic sequence, $\xi_0(i, j, k)$ is not defined. Hence, they are calculated until n = 1.

Usually, backward probabilities are denoted by β as in [26]. However, the authors note that they deviated from this notation due to numerical issues. In [26], a β backward probability is defined as the likelihood of the observed sequence and is calculated similar to α but in a backward direction, and then γ and ξ are calculated by means of α and β. In this paper, the authors have directly calculated the γ and ξ in a backward iteration in order to avoid numerical issues arising due to the length of the traffic.

The γ's and ξ's are calculated using Algorithm 2, described below, where δ_i^j is the Kronecker function, i.e.,

$$\delta_i^j = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}$$

Algorithm 2: Computing the backward probabilities $\gamma_n(i, k)$ and $\xi_n(i, j, k)$

1: for n = N – 1 → 1 do
2: for i = 1 → N_s do
3: for k = N – n – 1 → 0 do
4: if n = N – 1 then
5: $\gamma_{N-1}(i, k) = \alpha_{N-1}(i)$
6: else
7: if k ≠ 0 then

8: $\gamma_n(i,k) = \xi_{n+1}(i,i,k-1)$

9: else

10: $\gamma_n(i,0) = \sum_{\substack{j=1,\\ j\neq i}}^{N_s} \sum_{k'=0}^{N-n-2} \xi_{n+1}(i,j,k')$

11: end if

12: end if

13: end for

14: end for

15: for $i = 1 \rightarrow N_s$ do

16: for $j = 1 \rightarrow N_s$ do

17: for $k = N - n - 1 \rightarrow 0$ do

18: $\xi_n(i,j,k) = \frac{\alpha_{n-1}(i,0)\pi_{ij}d_j[k]+\delta_i^j\alpha_{n-1}(i,k+1)}{\sum_{l=1}^{N_s}(\alpha_{n-1}(l,0)\pi_{lj}d_j[k]+\delta_l^j\alpha_{n-1}(l,k+1))} \times \gamma_n(j,k)$

19: end for

20: end for

21: end for

22: end for

23: for $i = 1 \rightarrow N_s$ do

24: for $k = N - 1 \rightarrow 0$ do

25: if $k \neq 0$ then

26: $\gamma_0(i,k) = \xi_1(i,i,k-1)$

27: else

28: $\gamma_0(i,0) = \sum_{\substack{j=1,\\ j\neq i}}^{N_s} \sum_{k'=0}^{N-2} \xi_1(i,j,k')$

29: end if

30: end for

31: end for

The algorithm calculates the γ's and ξ's for n = N-1 first, then for n = N-2, and so on until n = 0 for γ's and n = 1 for ξ's. This is because, similar to the calculation of α's, the backward probabilities calculated for each GOP are dependent on those calculated for the next GOP. Consequently, the backward probabilities become conditional probabilities given the entire observed sequence.

As is evident from the algorithm itself, when $n \neq N-1$, γ calculated for the n^{th} GOP is dependent on the ξ calculated for the $(n+1)^{th}$ GOP. On the other hand, ξ calculated for the n^{th} GOP is dependent on the γ calculated for the same GOP, right before it. Hence, the backward probabilities calculated for each GOP essentially depend on those calculated for the next GOP, which in turn depend on those calculated for the next-to-next one and so on. Thus, similar to Algorithm 1, Algorithm 2 also takes a dynamic approach, but is a bit more complicated. Instead of calculating the γ's and ξ's for the same GOP, again and again, the algorithm just stores them into a matrix and looks them up whenever required.

The discussion of the above-described algorithm can be handled best if we divide it into cases, similar to the discussion on the calculation of α's.

Case I: (Calculation of γ's)

This calculation is simple if we take into account the definition of γ and the following cases.

$$\gamma_n(i,k) \stackrel{\text{def}}{=} P\left(s_n = i, \ldots, s_{n+k} = i, s_{n+k+1} \neq i \middle| y_0^{N-1}, \theta^{(m)}\right),$$

$$n = 0, \ldots, N-1; \quad k < N-n-1$$

Case a: (n = N-1)

The algorithm needs to be initialized at n = N-1, so that the calculation of both the backward probabilities can follow from there onwards. In this case the model stays in the i^{th} state from the n^{th} GOP until the $(n+k)^{th}$ GOP and then does not make a transition to a different state as the traffic sequence comes to an end. The probability that the model stays in state i in the last or $(N-1)^{th}$ GOP is $\alpha_{N-1}(i)$. Hence, we have,

$$\gamma_{N-1}(i,k) = \alpha_{N-1}(i)$$

Case b: ($n \neq N-1$)

In this case, we have the following two sub-cases.

Case i:

In this case, we consider the transition probability that the model transits from state i to a state j, after the end of the n^{th} GOP, where it stays for k' future consecutive GOPs, i.e. $\xi_{n+1}(i, j, k')$. Now, we need to take into account all possible values for k'.

$$k'_{max} = \text{No. of GOPs after the } (n+1)^{th} \text{ GOP until the end}$$

$$= N - 1 - (n+1) = N - n - 2$$

So, in this case, we need to take a summation over k', as follows

$$\sum_{k'=0}^{N-n-2} \xi_{n+1}(i, j, k')$$

Again, we need to consider all possible transitions from the state i, such that $(i \neq j)$. Hence,

$$\gamma_n(i, 0) = \sum_{\substack{j=1, \\ j \neq i}}^{N_s} \sum_{k'=0}^{N-n-2} \xi_{n+1}(i, j, k')$$

Case ii:

The model may not make a transition to a different state after the end of the n^{th} GOP and may continue in the same state i for k future consecutive GOPs, i.e. $k \neq 0$. Here, the

model makes a transition from a state i to the same state i, after the end of the n^{th} GOP, and then continues there for (k-1) future consecutive GOPs. Thus, we have

$$\gamma_n(i,k) = \xi_{n+1}(i,i,k-1)$$

Case II: (Calculation of ξ's)

ξ's are essentially the transition probabilities that the model transits from one state at the end of a GOP. The transition may be from state i to the same state i or to a different state j. If, we suppose, during the n^{th} GOP, the model is in state j, then we need to consider the fact that a transition from any of the N_s states could have brought the model to the state j. There are, obviously, N_s such possible transitions. Thus, the probability of a transition from a state i to the state j as the model moves from the $(n-1)^{th}$ GOP to the n^{th} GOP, such that it stays in state j for k future consecutive GOPs, can be denoted as,

$$\xi_n(i,j,k) = \left(\frac{P\left(\text{transition from a state i to state j , where } (i=j) \text{ or } (i\neq j)\right)}{\sum_{l=1}^{N_s} P\left(\text{transition from a state l to state j , where } (l=j) \text{ or } (l\neq j)\right)}\right) \times P(\text{model pursues state j for k future consecutive GOPs})$$

We should note, here, that such a transition does not signify a physical transition from a state to another, but a figurative one, which makes our calculations easy. As noted before, when the model makes a physical transition out of a particular state, it always enters a different one, since $\pi_{ii} = 0$ for each state i.

In this case, as well, the calculation is simplified if we take into account the following cases.

Case a: $(i \neq j)$

In this case, the model makes a transition from a state i to a different state j, where it stays for k future consecutive GOPs. The probability of such a transition is:

$$\begin{aligned} &P\big(\text{for the } (n \\ &-1)^{th} \text{ GOP the model is in state i and does not continue in that state anymore}\big) \\ &\times P(\text{model transits from state i to state j}) \\ &\times P(\text{model continues in state j for k future consecutive GOPs}) \\ &= \alpha_{n-1}(i,0)\pi_{ij}d_j[k] \end{aligned}$$

Case b: $(i = j)$

In this case, the model transits from a state i to the same state i, where it stays for k future consecutive GOPs. So the probability of such a transition can be given as,

$$\begin{aligned} &P\big(\text{model pursues state i during the } (n-1)^{th}, n^{th} \text{ and k future consecutive GOPs}\big) \\ &= \alpha_{n-1}(i, k+1) \end{aligned}$$

Hence, the probability that the model makes a transition from a state i to a state j (where, i = j or i ≠ j), as it moves from the $(n-1)^{th}$ GOP to the n^{th} GOP, can be given as,

$$P(\text{Case a}) + \delta_i^j P(\text{Case b})$$

$$= \alpha_{n-1}(i,0)\pi_{ij}d_j[k] + \delta_i^j \alpha_{n-1}(i,k+1)$$

Note that, when $i \neq j$, $\delta_i^j = 0$ and the stated probability becomes $\alpha_{n-1}(i,0)\pi_{ij}d_j[k]$. Whereas, when i = j, $\pi_{ij} = \pi_{ii} = 0$ and the stated probability becomes $\alpha_{n-1}(i,k+1)$.

Finally, we can state that,

$$\xi_n(i,j,k) = \left(\frac{P\left(\text{transition from a state i to state j, where } (i=j) \text{ or } (i \neq j)\right)}{\sum_{l=1}^{N_s} P\left(\text{transition from a state l to state j, where } (l=j) \text{ or } (l \neq j)\right)}\right) \times P(\text{model pursues state j for k future consecutive GOPs})$$

$$= \frac{\alpha_{n-1}(i,0)\pi_{ij}d_j[k] + \delta_i^j \alpha_{n-1}(i,k+1)}{\sum_{l=1}^{N_s}(\alpha_{n-1}(l,0)\pi_{lj}d_j[k] + \delta_l^j \alpha_{n-1}(l,k+1))} \times \gamma_n(j,k)$$

2.1.2.4 Estimation of the Parameter Set

The parameter set $\theta^{(m+1)}$, as defined earlier, can be calculated with the help of the forward and the backward probabilities, as follows.

I. Initial Probabilities: Initial probabilities can be represented as a vector of N_s elements, whose each element is of the form π_i, which is essentially the probability that the model will initiate in state i. Thus, we can state,

$$\pi_i = \text{P(model initializes in state i at n} = 0)$$

$$= \sum_k \text{P(model initializes in state i at n} = 0 \text{ and continues there for k future consecutive GOPs)}$$

or,

$$\pi_i = \sum_{k=0}^{N-1} \gamma_0(i,k)$$

Since, here, we are dealing with the first GOP; k can range from 0 to the entire traffic sequence.

Obviously, the sum of all the initial probabilities must be equal to 1, i.e. $\sum_{i=1}^{N_s} \pi_i = 1$, as the model needs to initialize in some state at t = 0 or n = 0.

II. Transition Probabilities: The transition probabilities π_{ij}, are estimated as follows.

$$\pi_{ij} = \frac{\#(\text{transitions made by the model from state i to state j})}{\sum_{j'} \#(\text{transitions made by the model from state i to a state j}')}$$

$$= \frac{\sum_{n,k} \text{P(the model makes a transition from state i to state j before the } n^{th} \text{ GOP and continues in state j for k future consecutive GOPs)}}{\sum_{j'} \sum_{n,k} \text{P(the model makes a transition from state i to a state } j' \text{ before the } n^{th} \text{ GOP and continues in state } j' \text{ for k future consecutive GOPs)}}$$

Basically, here, we examine every GOP and then calculate the probability that such a transition may happen at that time instant. During this calculation, we, also, need to consider the duration, for which the model pursues the new state after the transition. Thus, we need to calculate the probabilities of possible transition for all possible values of k and then use their sum towards the desired result. Consequently, the above equation can be represented as,

$$\pi_{ij} = \begin{cases} \dfrac{\sum_{n=1}^{N-1} \sum_{k=0}^{N-n-1} \xi_n(i,j,k)}{\sum_{\substack{j'=1, \\ j' \neq i}}^{N_s} \sum_{n=1}^{N-1} \sum_{k=0}^{N-n-1} \xi_n(i,j',k)}, & \text{if } (i \neq j) \\ 0, & \text{if } (i = j) \end{cases}$$

A transition cannot happen before the beginning of the traffic sequence. As a result, for this calculation, we cannot use the GOP n = 0. Thus, the corresponding summation is taken over n = 1 to n = N-1. As discussed earlier $k_{max} = (N-1) - n = N - n - 1$. Hence, the corresponding summation is taken over k = 0 to k = N-n-1. The model can transit from the state i to any state j', as long as ($j' \neq i$). Consequently, the corresponding summation is taken over $j' = 1$ to $j' = N_s$, such that ($j' \neq i$). Of course, $\pi_{ii} = 0$.

We should note here, that $\sum_{j=1}^{N_s} \pi_{ij} = 1$, since, as the model makes a transition from the state i, it must enter some state j.

III. Frame Size Distribution: As described earlier, as well, the size of each frame in the GOP is described by a separate pmf $b_i[\cdot]$, where $[\cdot]$ gives the position of the frame in the GOP. Thus, depending on the state i and the position of the frame, under consideration, within the GOP, we have a unique pmf. Consequently, we have $N_f \times N_s$ pmfs, in total. As mentioned earlier, each such pmf is an empirical distribution described by a histogram with a different number of bins, depending on the frame type (namely I, P, or B) of the compressed picture to be generated. During this step of the estimation process, each pmf is re-estimated. This means that the height of each bin of each pmf is recalculated.

Define the Kronecker function, $\delta_x^{y[n]}$, as,

$$\delta_x^{y[n]} = \begin{cases} 1, & \text{if the corresponding frame in } y[n] \text{ falls in the range that describes the bin } x \\ 0, & \text{otherwise} \end{cases}$$

where x denotes a bin of a particular pmf and y[n] represents the set of frame sizes in the n^{th} GOP.

Then each bin of each pmf is re-estimated as follows

$$b_i[x] = \frac{\sum_n P(n^{th} \text{ GOP is in } i^{th} \text{ state}) \delta_x^{y[n]}}{\sum_n P(n^{th} \text{ GOP is in } i^{th} \text{ state})}$$

$$= \frac{\sum_{n,k} P(n^{th} \text{ GOP is in } i^{th} \text{ state for all possible values of } k) \delta_x^{y[n]}}{\sum_{n,k} P(n^{th} \text{ GOP is in } i^{th} \text{ state for all possible values of } k)}$$

Basically, here, we examine every GOP and then calculate the probability that such a GOP may be present in the i^{th} state. During this calculation, we, also, need to consider the duration, for which the model stays in state i. Thus, we need to calculate the above-mentioned probabilities for all possible values of k and then use their sum towards the desired result. Consequently, the above equation can be written as,

$$b_i[x] = \frac{\sum_{n=0}^{N-1} \sum_{k=0}^{N-n-1} \gamma_n(i,k) \delta_x^{y[n]}}{\sum_{n=0}^{N-1} \sum_{k=0}^{N-n-1} \gamma_n(i,k)}$$

IV. State Duration (Lambda): State-dependent parameter, λ_i, of the Poisson-dstributed state duration is obtained as follows:

$$\lambda_i = \frac{\sum_{n=1}^{N-1} \sum_{k=0}^{N-n-2} \sum_{\substack{j=1, \\ j \neq i}}^{N_s} k\xi_n(j,i,k) + \sum_{k=0}^{N-1} k\gamma_0(i,k)}{\sum_{n=1}^{N-1} \sum_{k=0}^{N-n-2} \sum_{\substack{j=1, \\ j \neq i}}^{N_s} \xi_n(j,i,k) + \sum_{k=0}^{N-1} \gamma_0(i,k)}$$

The above expression is obtained by examining each possible value of the actual state duration, given by k, and then multiplying it by a weight, which denotes the probability of the occurrence of that particular value of k. There are two parts of the calculation of the duration – i.) The model begins in state i and stays in state i for k future consecutive GOPs. This occurs with probability $\gamma_0(i,k)$; ii.) The model makes a transition from a state j to the

state i and stays in state i for k future consecutive GOPs. This occurs with probability $\xi_n(j, i, k)$. Finally, the average is taken over the sum of the weights.

2.1.3 Implementation of the Model

First, the average frame size of each GOP is calculated by adding the individual frame sizes present within the GOP under consideration, and then dividing it by the number of frames present in each GOP. The GOPs are then classified to different states, based on their average frame sizes. Each state is described by the range or threshold of the average frame sizes of the GOPs it represents. The thresholds are selected, carefully, so as to ensure that each state comprises of approximately equal number of GOPs. Thus, the entire observed traffic sequence gets converted to a sequence of states that the model should be able to imitate.

Please note that during our recreation of the model, we used three states, i.e., $N_s = 3$, as suggested by the authors of the paper. The three states represent three different video activity levels, viz., low, medium and high.

The EM algorithm, as discussed above, requires a coarse estimation of each of the parameters, as input, which are obtained as follows.

I. Initial Probabilities: Each element, π_i, of the initial probabilities can be initialized to random positive values, such that $0 \leq \pi_i \leq 1$ and $\sum_{i=1}^{N_s} \pi_i = 1$. While recreating the model, we have set each such element as, $\pi_i = \frac{1}{N_s}$.

II. Transition Probabilities: Each element, π_{ij}, of the transitional probabilities can be initialized to random positive values, such that $0 \leq \pi_{ij} \leq 1$, $\sum_{j=1}^{N_s} \pi_{ij} = 1$. Of course, in case i = j, $\pi_{ij} = 0$, i.e., $\pi_{ii} = 0$.

Otherwise, in case $i \neq j$, the authors advocated the use of random positive values. However, in our recreation of the model, we have used a different approach. During the calculation of every element, π_{ij}, such that $i \neq j$, we have at first noted the number of times a transition occurs in the actual traffic sequence from the state i to the state j. After obtaining similar values corresponding to all such elements, we have calculated,

$$\pi_{ij} = \frac{\#\ (\text{transitions from the state i to the state j})}{\sum_{\substack{j'=1,\\ j' \neq i}}^{N_s} \#\ (\text{transitions from the state i to a state } j')}$$

The above-described approach helps us obtain an approximation of the transition probabilities, that is close to the actual estimate produced by the EM algorithm and thus, helps us reduce the number of iterations required by the algorithm to reach the point of convergence.

III. Frame Size Distribution: As we have discussed earlier, the number of bins in the pmfs of the model may vary, depending on the type of frames, in order to have a trade-off between performance and model complexity. The number of bins, for each frame type, that we have used in our experiments is similar to that used by the authors. The bins are placed in the interval between the minimum and the maximum observed frame size for each frame type. A coarse estimation of the pmfs is obtained by evaluating frame size histograms related to each state and each type of frame. Zero-valued bins are set to a low fixed value and the histograms are normalized accordingly.

IV. State Duration (Lambda): Each element, λ_i, is initialized as 1.

After the initial values of each of the various parameters have been set, as described above; these values are passed, as input, to the EM algorithm. As we have discussed earlier, the EM algorithm undergoes a number of iterations until it reaches the point of convergence. Each iteration consists of two steps, viz. Expectation and Maximization. Expectation, again, consists of two steps – (i) the computation of forward probabilities, namely α's; and, (ii) the computation of backward probabilities, namely γ's and ξ's. Maximization, on the other hand, consists of only one step – the estimation of the parameter set. The authors define the point of convergence as the iteration after which the difference between the log-likelihood of the two most recent iterates is less than 0.01. Convergence is assured by Jensen's inequality [10].

After convergence has been achieved by the EM process, we have a valid estimate of the various parameters involved, and the model can now be used to generate a synthetic

trace. The fundamental concept behind the generation of the synthetic trace is to first generate a sequence of states, as dictated by the various parameters of the model, and then to produce synthetic traffic for a GOP for each state of the sequence by means of the frame size pmfs. In this regard, the authors suggested the conversion of bins to the mean frame size of the interval they represent.

Algorithm 3, as described below, is used for the generation of synthetic traffic.

Algorithm 3: Synthetic Traffic Generation

1: Initial state i extracted according to the distribution defined by the initial probabilities, viz. $\pi_1, \pi_2, \ldots, \pi_{N_s}$

2: $n \leftarrow 0$

3: while $n < N$ do

4: Sample k from the Poisson distribution that corresponds to the current state, i.e., $d_i[k]$

5: if $n \geq N - k$ then

6: $k \leftarrow N - n - 1$

7: end if

8: for $j = 0$ to k do

9: generate a synthetic GOP $x[n+j]$ according to $b_i[\cdot]$

10: end for

11: $n \leftarrow n + k + 1$

12: Perform state transition according to the distribution defined by the Transition Probabilities, viz. Π

13: end while

2.2 The Markov Modulated Gamma Model (3D-MMG)

The authors discuss in [33] a stochastic model that characterizes traffic generated by a multiview video coding (MVC) variable bit rate (VBR) source. It is a Markov Modulated Gamma Model (3D-MMG) which defines the generation of frames for the base view as a separate process from that of the dependent views. It is based on the MMG model proposed in [30] and is very similar to the model for SVC video presented in [39].

In the 3D-MMG model, the authors assume that the base view is modeled independently, since it is encoded independently. The base view in MVC is, typically, the left view and the authors modeled it using the Markov-modulated gamma (MMG) model proposed in [30]. In this model, first, we partition the video frames into clips. A clip is a sequence of consecutive similar sized GOPs. For instance, if the following k GOPs G_{i+1}, G_{i+2}, … , G_{i+k} are in the same clip, starting from the i^{th} GOP, then G_{i+k+1} belongs to the same clip if its size is less than the average clip size or equal to the average clip size plus a threshold. The clips are then organized into shot classes. A shot class of length k is a union of k distinct but not necessarily consecutive clips. Each clip belongs to only one shot class. A predefined number of shots n is used for generating the synthetic trace. In our implementation, we used

n = 7 as it was the optimal number of shots obtained when experimenting with different number of shots. The GOP sizes were partitioned into these 7 shots. The successive partitioning boundaries of these shots increase in a geometric progression with a as the first term, and $b = ar^n$, as the $(n+1)^{th}$ term, where $r = e^{\frac{(\ln b - \ln a)}{n}}$, a and b are the smallest and the largest GOP sizes. Please note that, a clip size or a GOP size, typically, refers to the average frame size of the frames present in a clip or in a GOP.

A transition probability matrix lP is formed for the transition probabilities between the different shot classes. These shot classes are the states of the underlying Markov chain. The transition probabilities are computed from normalized relative frequency of transitions among shot classes as one sequentially traverses all GOPs in the original video i.e.,

$$lp_{ij} = P(\text{the next GOP belongs to shot j} \mid \text{the current GOP belongs to shot i})$$

$$= \frac{\#(\text{transitions from the shot i to the shot j})}{\sum_{j'=1}^{n} \#(\text{transitions from the shot i to any shot } j')}$$

Finally, each shot is sub-partitioned into three groups, one per type of frame. That is, the first group contains all the I frames in the shot, the second group contains all the P frames and the third group contains all the B frames. Thus, all frames are partitioned into 3n, i.e., 21 data sets as each shot is sub-partitioned based on the type of frame I, B or P. An axis-shifted gamma distribution is fitted to the frame sizes of each of the 21 data sets and its parameters are estimated from the data set it models. For the shift, we ignore 1% of the data points or

frame sizes and set the value of the shift at the one percentile. Thus, for the seven shot classes, there are seven different gamma distributions for each frame type I, B and P.

The right view is modeled as follows. We start by partitioning the GOPs of the right view into shot classes just like the left view. Let the resulting shot-class sequences of the GOPs of the base left view and the right view be $lSG = (lSG_1, lSG_2, \dots, lSG_m)$ and $rSG = (rSG_1, rSG_2, \dots, rSG_m)$ respectively, where m is the total number of GOPs in the video sequence. The state agreement probability $p(lSG = rSG)$ between the left view shot-class lSG and the right view shot-class rSG is defined as the ratio of the number of pairs (lSG_i, rSG_i) having identical values with the total number of pairs in them [39]. The state agreement probability quantifies the fraction of corresponding GOP pairs in the left and right view assigned to the same shot-class. Let lP be the transition matrix for the MMG model of the left view and let cP be a connection matrix [39]. We, now, describe how cP can be computed from the shot-class sequences lSG and rSG of the left and right view. Let $cSG = (<lSG_1, rSG_1>, <lSG_2, rSG_2>, \dots, <lSG_m, rSG_m>)$ be the connection sequence pairs between the left and right views. Let lf_i be the number of i entries in the left view shot-class sequence lSG, and cf_{ij} be the number of pairs $<lSG_k, rSG_k>$ in cSG, such that $lSG_k = i$ and $rSG_k = j$. The element cp_{ij} of the connection matrix cP is defined as,

$$cp_{ij} = \frac{\#(\text{i entries in lSG such that corresponding entry in rSG is j})}{\sum_{j'=1}^{n} \#(\text{i entries in lSG such that corresponding entry in rSG is } j')}$$

$$= \frac{\#(\text{i entries in lSG such that corresponding entry in rSG is j})}{\#(\text{i entries in lSG})}$$

$$= \frac{cf_{ij}}{lf_i}$$

The algorithm for generating synthetic video traces for both the left and the right views, viz., lSG′ and rSG′ is described in Algorithm 4.

Algorithm 4: Generation of Synthetic Trace

1: Draw the initial random state for the left view
2: Generate the frame sizes for one GOP in this state using the respective Gamma distributions for I, P and B frames
3: Generate the right view state using the connection matrix cP
4: Generate the frame sizes for one GOP in this state
5: while $i \leq m$ do
6: Generate the next state for left view using the transition matrix lP for left view
7: Generate the next state for right view using the connection matrix cP
8: Generate one GOP within each state for each view
9: i++
10: end while

Chapter 3

Evaluation of the Models

In this chapter, we evaluate and compare the two 3D models, presented in the previous chapter. In the literature, the accuracy of a video traffic model is evaluated by testing how close the distributions of the generated I, B and P frame sizes are to those in the original frame trace, which was used to develop the model. This is done using Q-Q plots. A Q-Q plot (Q stands for quantile) is a graphical method for comparing two probability distributions by plotting their quantiles against each other. The Autocorrelation Function (ACF) of the size of successive frames is also confirmed by comparing it with the original one. These comparisons are visual but they do provide some idea of the model's accuracy. However, they do not reveal any information about the QoS metrics of the model's resulting packet trace as the flow of packets is transmitted through a series of network elements such as switches and routers. Therefore, we have provided two sets of comparisons, i.e., (1) the Q-Q plots to compare the frame size distributions and ACF of successive frame sizes and, (2) the QoS metrics of the packetized video trace when transmitted over a tandem network, namely, the 95^{th} percentile of the end-to-end delay, the jitter, and the packet loss rate.

Both models were implemented and evaluated using the video traces given in table 3.1. These traces are available from the Arizona State University's video traces library. The videos are encoded with the JMVC 8.3.1 encoder. We have used these traces because they are the only MVC video traces that are well-documented and publicly available. The GOP size used is 16 with 7 B frames between I and P frames with the following pattern: IBBBBBBBPBBBBBBB. The 3D property of each video trace is achieved with the help of 2 views. All the videos are HD with 1920 ×1080 resolution.

TABLE 3.1: Video traces used for evaluation and comparison

Movie	**Frame Rate (frames/sec)**	**Total number of frames**	**Mean frame size (bytes)**	**Mean GOP size (bytes)**
Alice in Wonderland	24	102,368	6601	211249
IMAX Space Station	24	102,368	11357	363416
Monsters vs Aliens	24	102,368	6278	200864
Clash of the Titans	24	102,368	7108	227467

For convenience and clarity of discussion, we have organized this chapter as follows. Section 3.1 illustrates the accuracy of the models with the help of Q-Q plots and ACF; while section 3.2 presents the QoS metrics. Finally, section 3.3 provides a short discussion on the merits and demerits of the two models.

3.1 Q-Q Plots and Autocorrelation Function

The Q-Q plots for the left and right views of each of the video traces are given in figures 3.1, 3.2, 3.3 and 3.4 respectively. The Q-Q plots are for the combined pdf of all I, P and B frames. The P-HMM model was trained on the first 500 GOPs of the trace, because of its high computational cost. In view of this, the comparison results are presented for the first 500 GOPs only. Based on the Q-Q plots, we can say that the pdf of all the left view frame sizes is modeled accurately by both models. However, the P-HMM model slightly under-estimates the upper tail of the pdf distribution of all the frame sizes of right view. The ACF of frame sizes produced by both models for each of the video traces are given in figures 3.5, 3.6, 3.7 and 3.8 respectively. The ACF for both models is similar to the actual trace.

3.2 QoS Metrics

The QoS behavior of both models for each of the video traces was evaluated using the

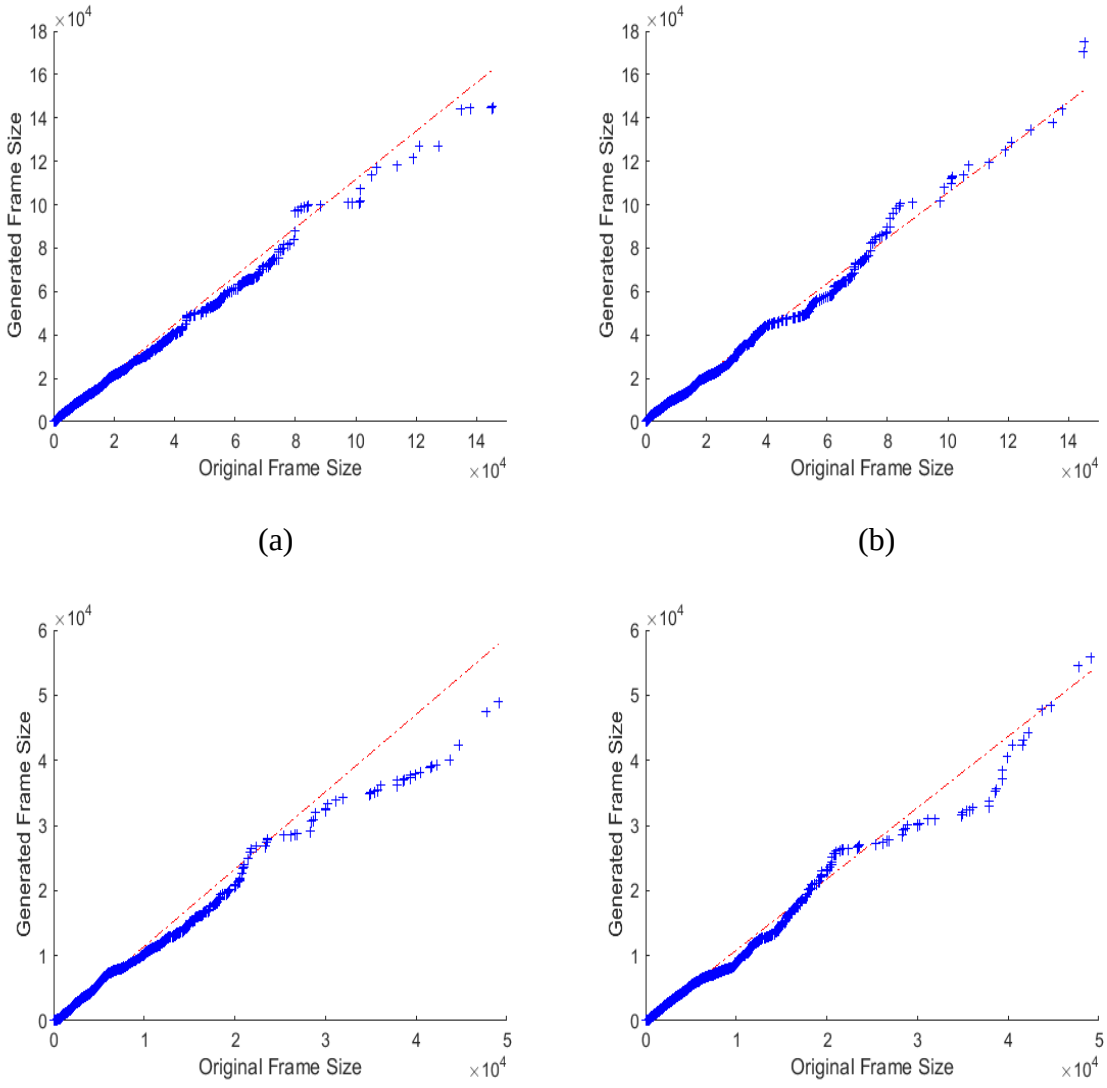


Figure 3.1: Q-Q plots for Alice in Wonderland: (a) Left view for P-HMM, (b) Left view for 3D-MMG, (c) Right view for P-HMM, (d) Right view for 3D-MMG

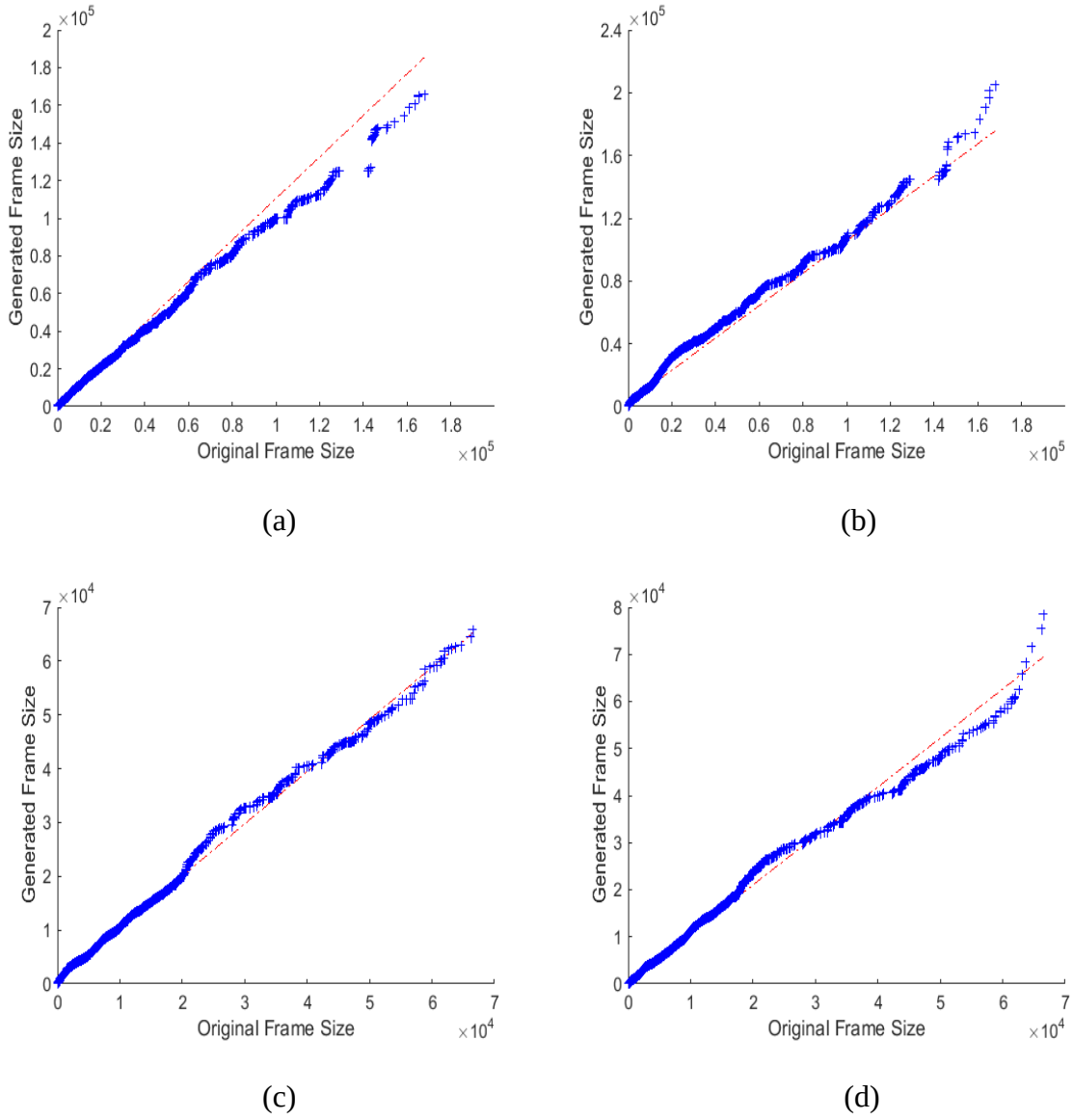


Figure 3.2: Q-Q plots for IMAX Space Station: (a) Left view for P-HMM, (b) Left view for 3D-MMG, (c) Right view for P-HMM, (d) Right view for 3D-MMG

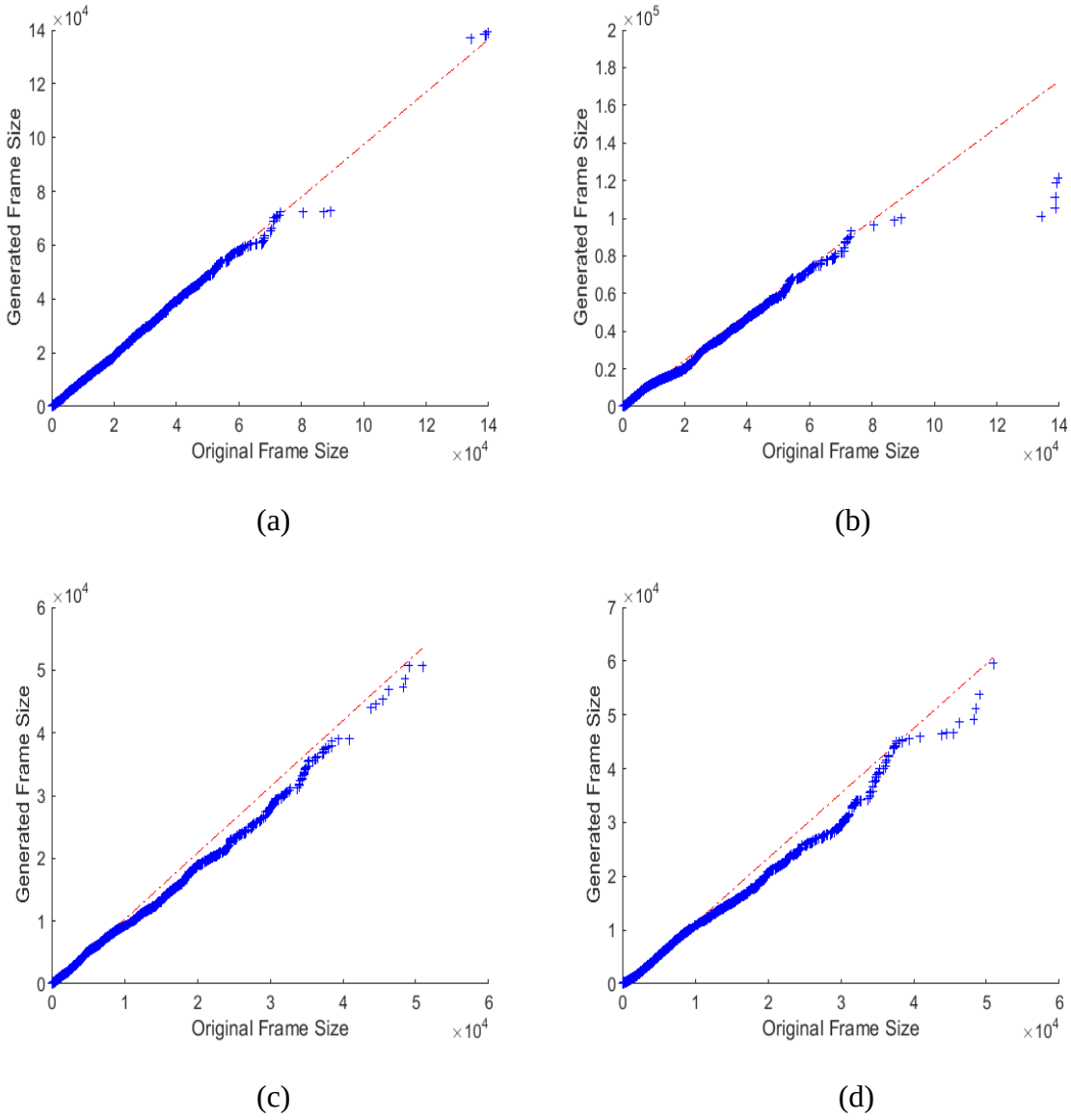


Figure 3.3: Q-Q plots for Monsters vs Aliens: (a) Left view for P-HMM, (b) Left view for 3D-MMG, (c) Right view for P-HMM, (d) Right view for 3D-MMG

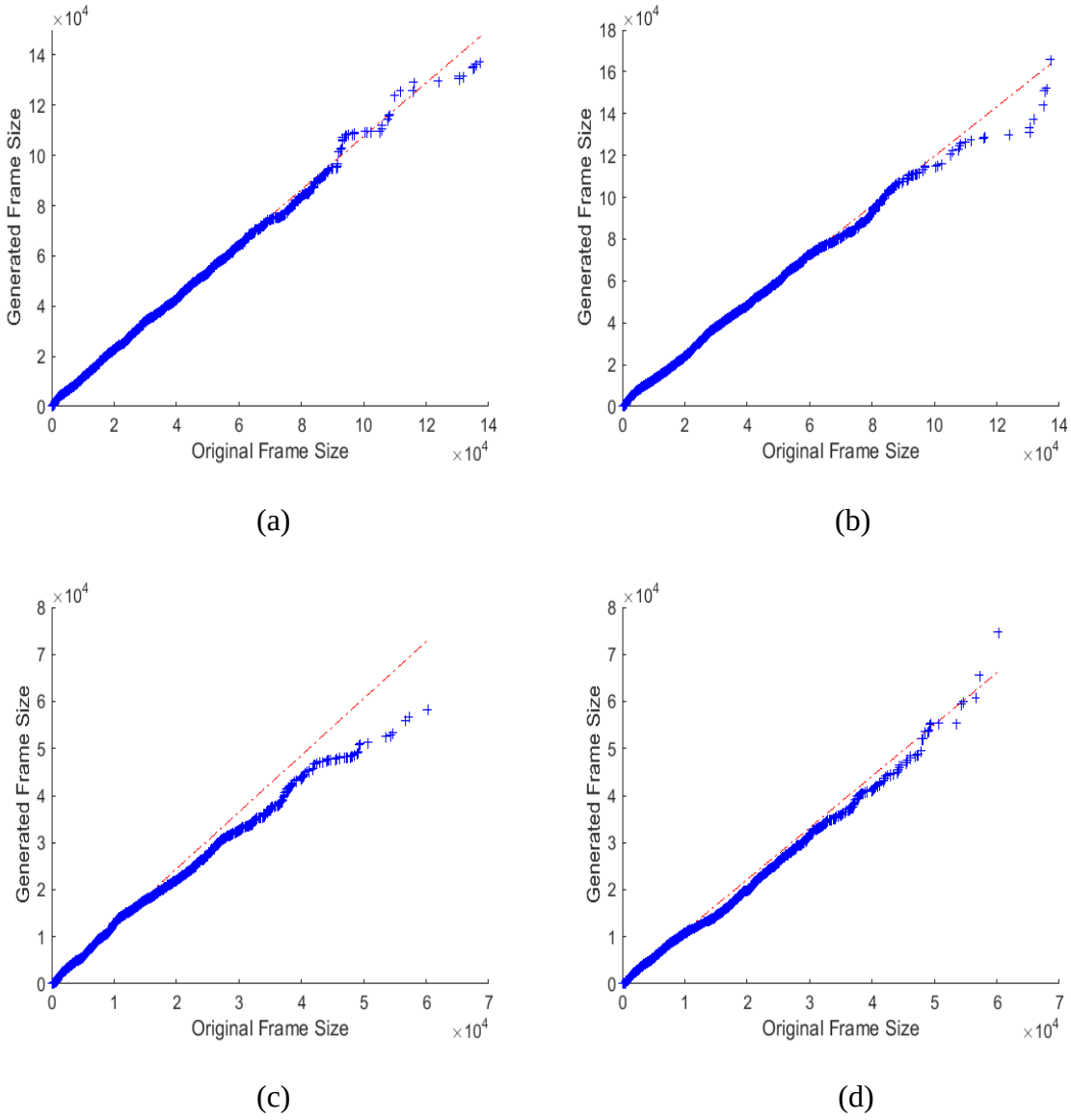


Figure 3.4: Q-Q plots for Clash of the Titans: (a) Left view for P-HMM, (b) Left view for 3D-MMG, (c) Right view for P-HMM, (d) Right view for 3D-MMG

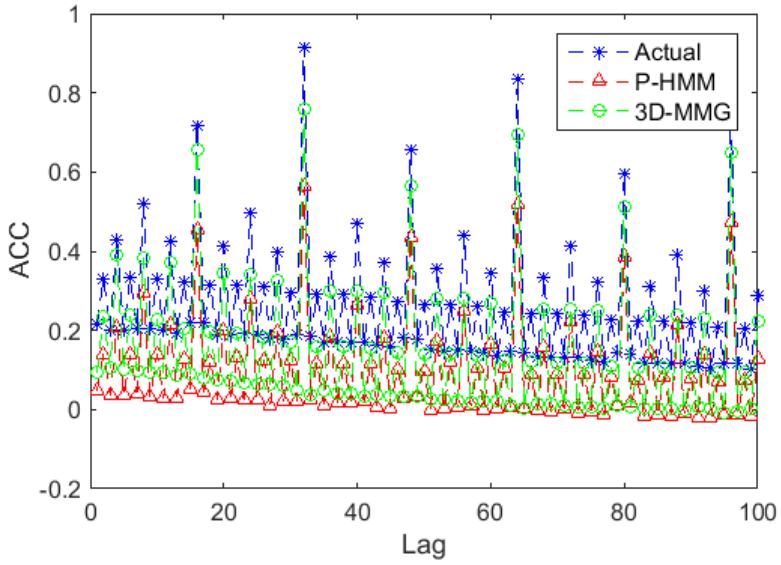


Figure 3.5: Auto-correlation function for both views for Alice in Wonderland

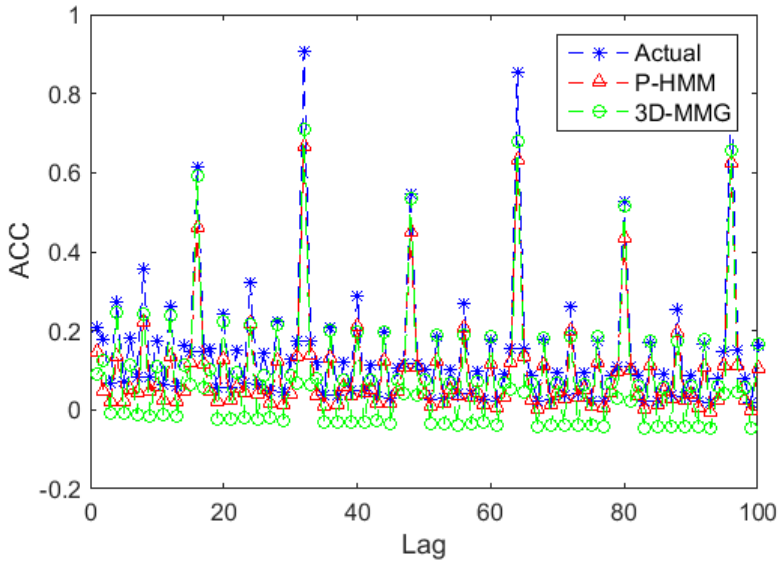


Figure 3.6: Auto-correlation function for both views for IMAX Space Station

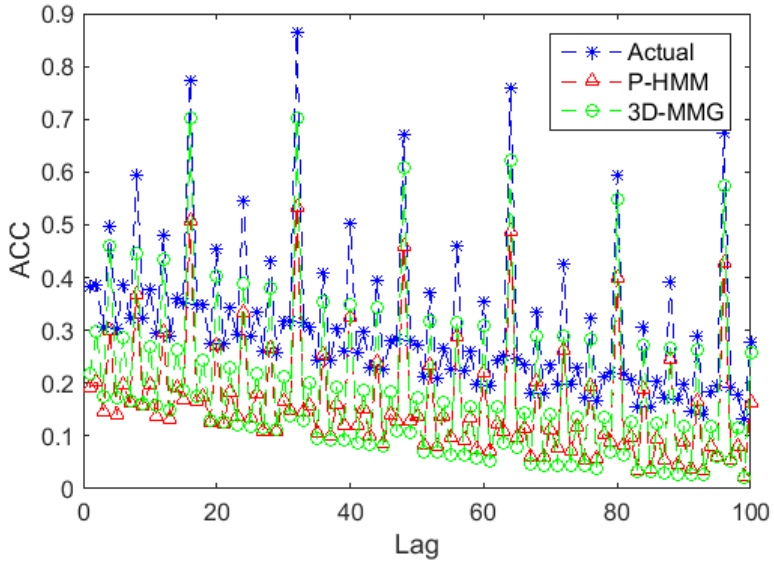


Figure 3.7: Auto-correlation function for both views for Monsters vs Aliens

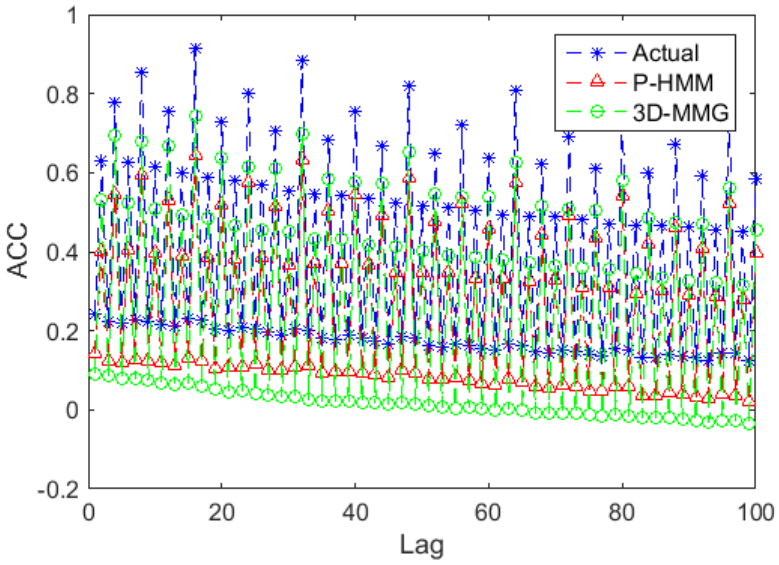


Figure 3.8: Auto-correlation function for both views for Clash of the Titans

Figure 3.9: The queueing network model

generated packet trace in a simulation model of the queueing network shown in figure 3.9. Specifically, the video model generates a frame sequence, which is converted into a sequence of IP packets. The H.264/AVC standard introduced the concept of network abstraction layer units (NALUs) which encapsulate the video frames. Each frame is preceded by a prefix NALU of 8 bytes. We assume that the video stream is transmitted using RTP over IPv4, and the maximum transfer unit (MTU) is 1500 bytes. The sender fragments NALUs larger than 1460 bytes to avoid fragmentation along the path. Each packet consists of the 12 byte RTP header, an 8 byte UDP header, and 20 byte IPv4 header. The resulting packet trace is a sequence of pairs of the arrival time of a packet and its packet length. The queueing model used in this paper consists of a sequence of 10 single-server queues, with each queue representing the queueing encountered by the packets of the video flow at the output port of each router along the path of the flow. The service time at each queue μ depicts the bandwidth allocated to the flow, and the amount of time it takes to serve a packet is equal to its (packet length)/μ. μ is same for all the servers in the queueing network since the same bandwidth is allocated to each link. The mean packet arrival rate λ is calculated from the

packet trace by calculating the number of bits per second for each second and then taking the average over the period of the entire trace. Each packet preserves its packet length throughout the queueing network.

Using a very efficient simulation technique proposed in [3], we estimated the 95^{th} percentile of the end-to-end delay, packet loss and jitter as a function of the service rate μ at each queue. We note that jitter was calculated as the average of the differences of the end-to-end delays of successive packets. These results were obtained by multiplexing traffic from both views. For comparison purposes, we also obtained similar results using the original traces. Figures 3.10, 3.11 and 3.13 give the 95^{th} percentile delay, jitter and packet loss rate for each of the video traces for both the models. Figure 3.12 illustrates the traffic intensity obtained by both the models for each of the video traces. We note that the confidence intervals were not plotted as they are very small and were not discernible in the graphs. The 95^{th} percentile of the end-to-end delay and jitter were obtained assuming that each queue in the queueing network has an infinite capacity. The packet loss results were obtained by varying the buffer size of each queue for a fixed bandwidth μ that corresponds to a traffic intensity of 0.5 for the original trace. All queues were assumed to have the same buffer size. The traffic intensity is a measure of the average occupancy of the server during a specified period of time, and it is defined as λ/μ, i.e., the mean arrival rate divided by the mean service rate or the bandwidth allocated. Please note that there is no significant difference between results from simulations with background traffic and from those without it.

In figure 3.10, we see that both models under-estimated the 95^{th} percentile of the end-to-end delay, though, the 3D-MMG model has values closer to the original trace. We note

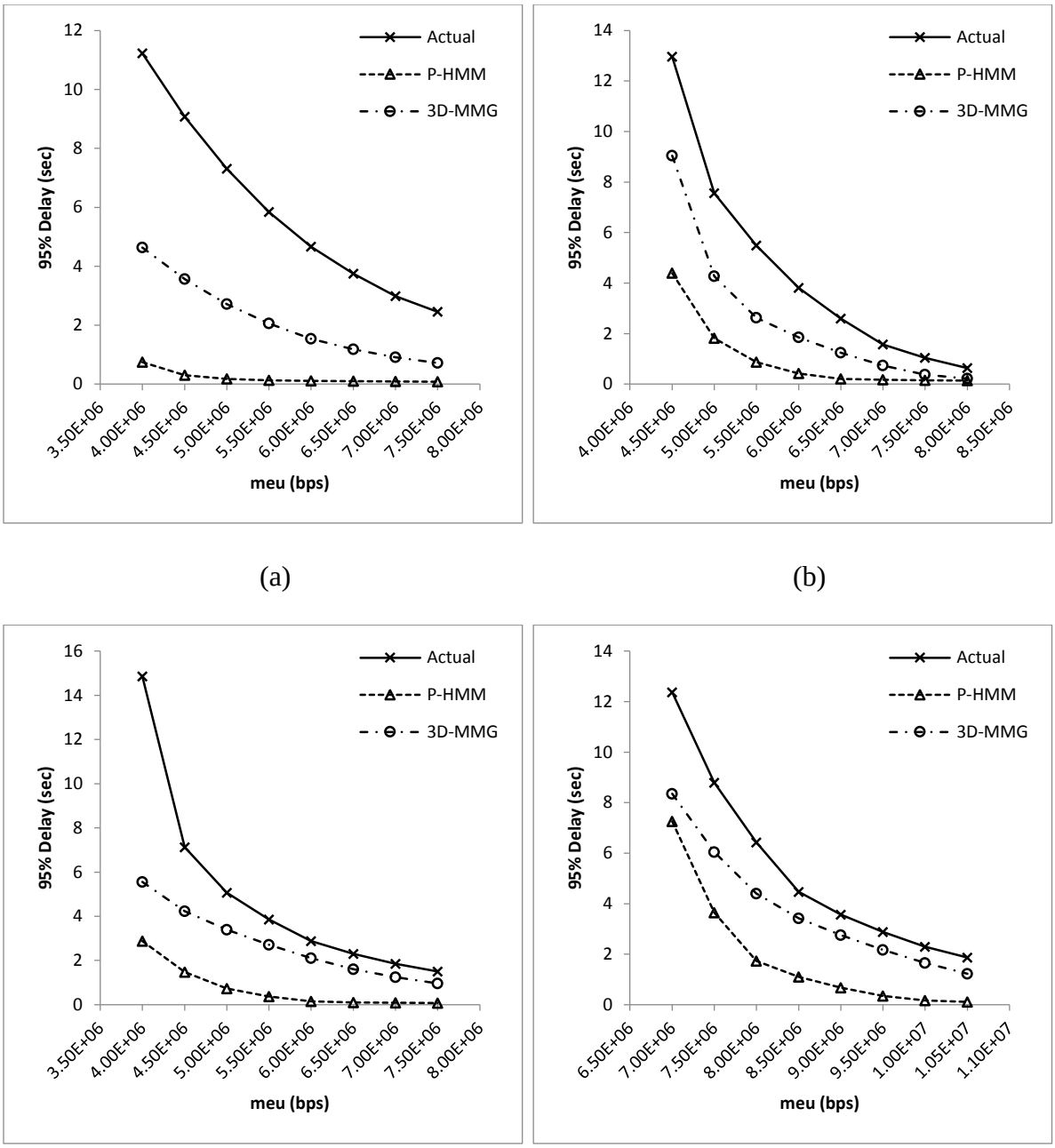


Figure 3.10: 95th percentile of the delay for (a) Alice in Wonderland, (b) IMAX Space Station, (c) Monsters vs Aliens, (d) Clash of the Titans

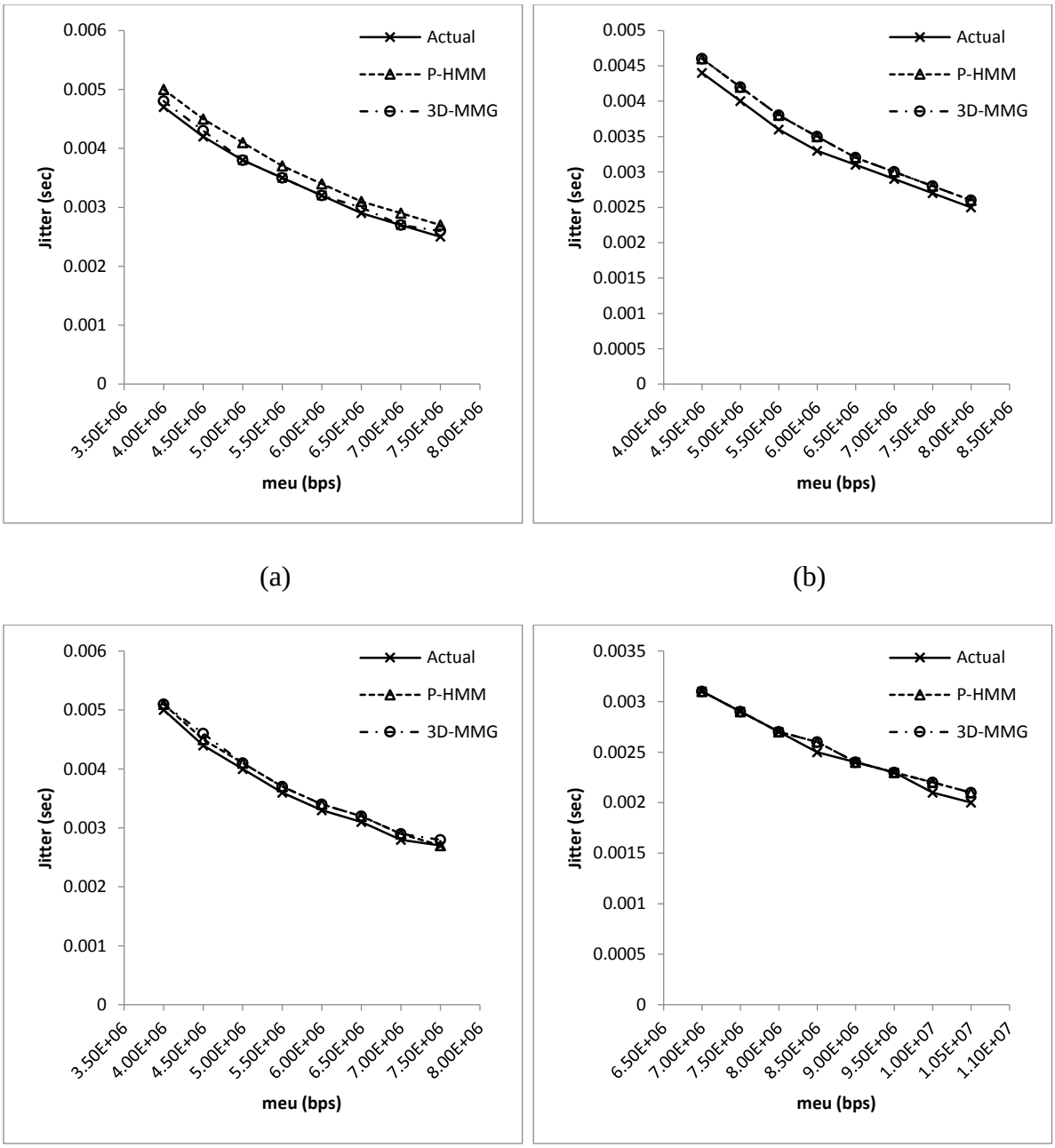
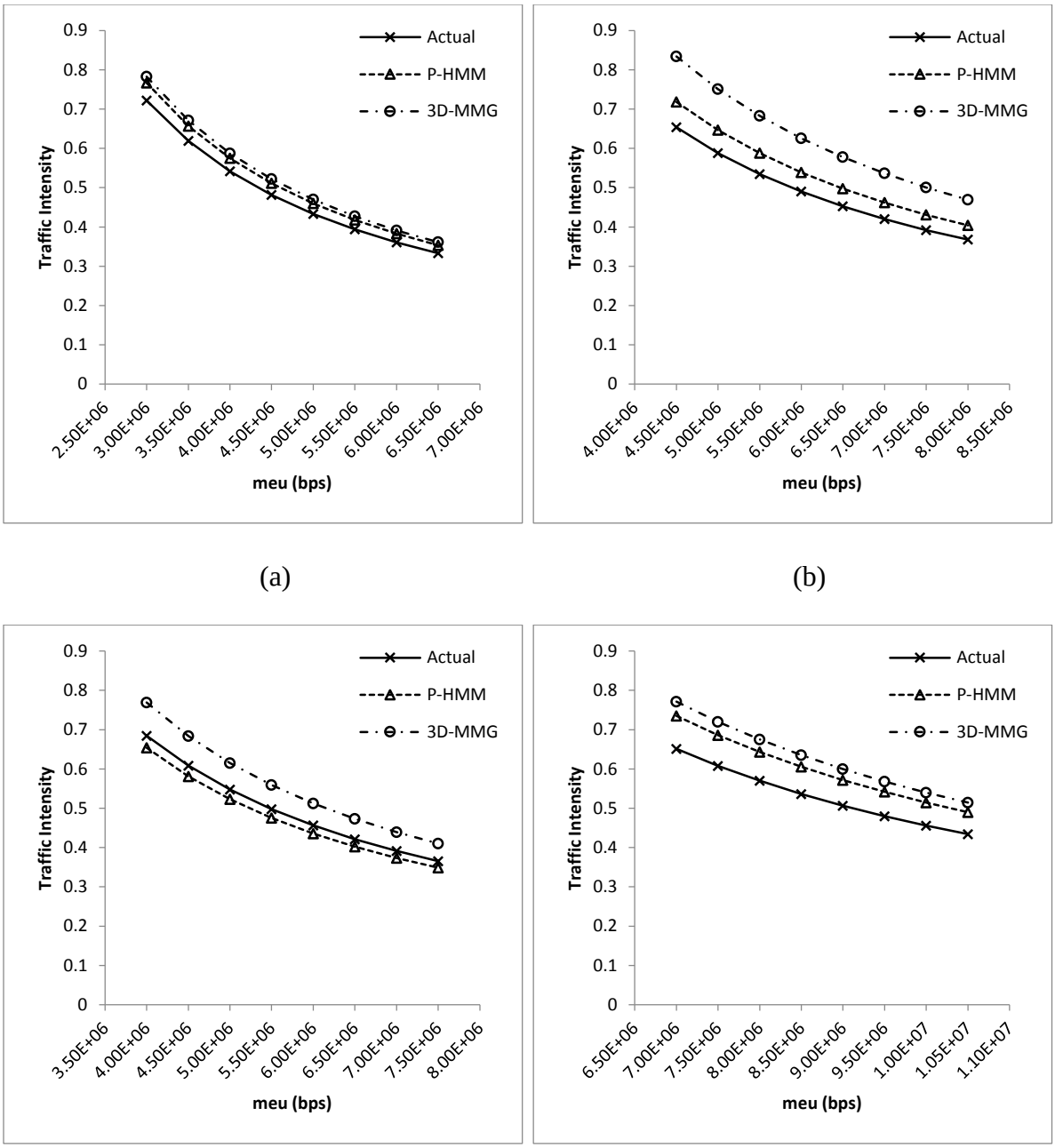


Figure 3.11: Jitter for (a) Alice in Wonderland, (b) IMAX Space Station, (c) Monsters vs Aliens, (d) Clash of the Titans



(a) (b)

(c) (d)

Figure 3.12: Traffic Intensity for (a) Alice in Wonderland, (b) IMAX Space Station, (c) Monsters vs Aliens, (d) Clash of the Titans

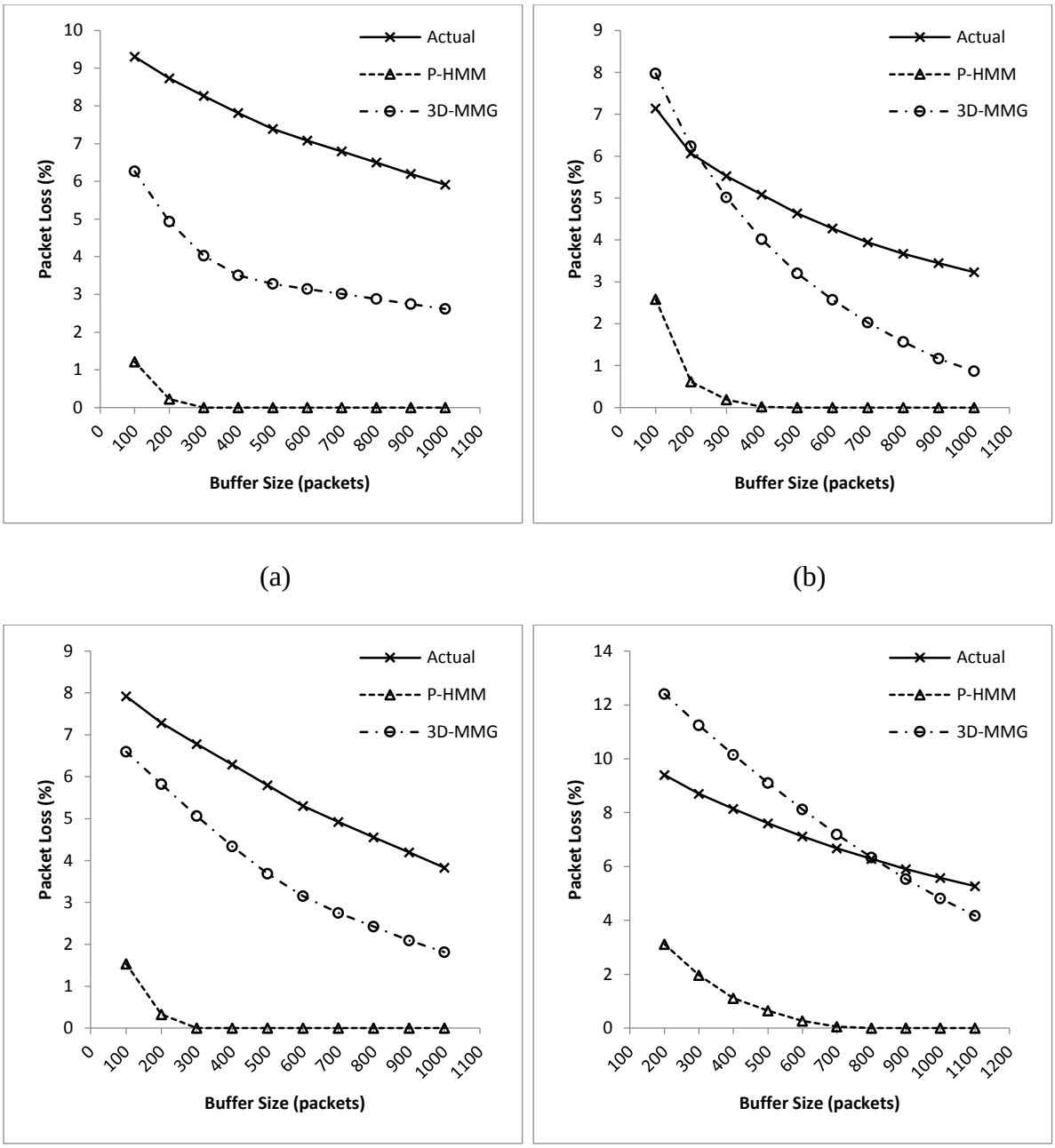


Figure 3.13: Packet Loss for (a) Alice in Wonderland, (b) IMAX Space Station, (c) Monsters vs Aliens, (d) Clash of the Titans

that the difference between actual and predicted values is higher at lower bandwidth values. In figure 3.11, the jitter is predicted accurately by both models. As is apparent from figure 3.12, the 3D-MMG model slightly over-estimates the traffic intensity, while the P-HMM model is able to imitate the trend of the original trace more efficiently. Finally, in figure 3.13, we see that the 3D-MMG model is able to predict a trend of packet loss which follows that of the original trace. The P-HMM model largely under-estimates the packet loss.

Some more results are presented for the 3D-MMG model, where the model was trained on the entire trace, consisting of 102,368 frames. Figures 3.14 and 3.15 present the 95^{th} percentile of the end-to-end delay and the packet loss rate respectively for such a scenario. We could not train the P-HMM model for the entire trace as it has a very high computational complexity. Hence, these figures do not present similar results for the P-HMM model. In figure 3.14, we see that the 3D-MMG model was able to predict values for the 95^{th} percentile of the end-to-end delay even closer to the original trace in such a case. Figure 3.15 displays a similar outcome for the packet loss rate. Jitter and traffic intensity, for such a scenario, are not provided as they are quite close to those presented earlier.

3.3 Merits and Demerits

The Q-Q plots and the ACF, provided in section 3.1, reveal that both the models are efficient and can effectively reproduce a sequence of frame sizes that closely resembles the original trace on which they are trained. However, each has certain merits and demerits.

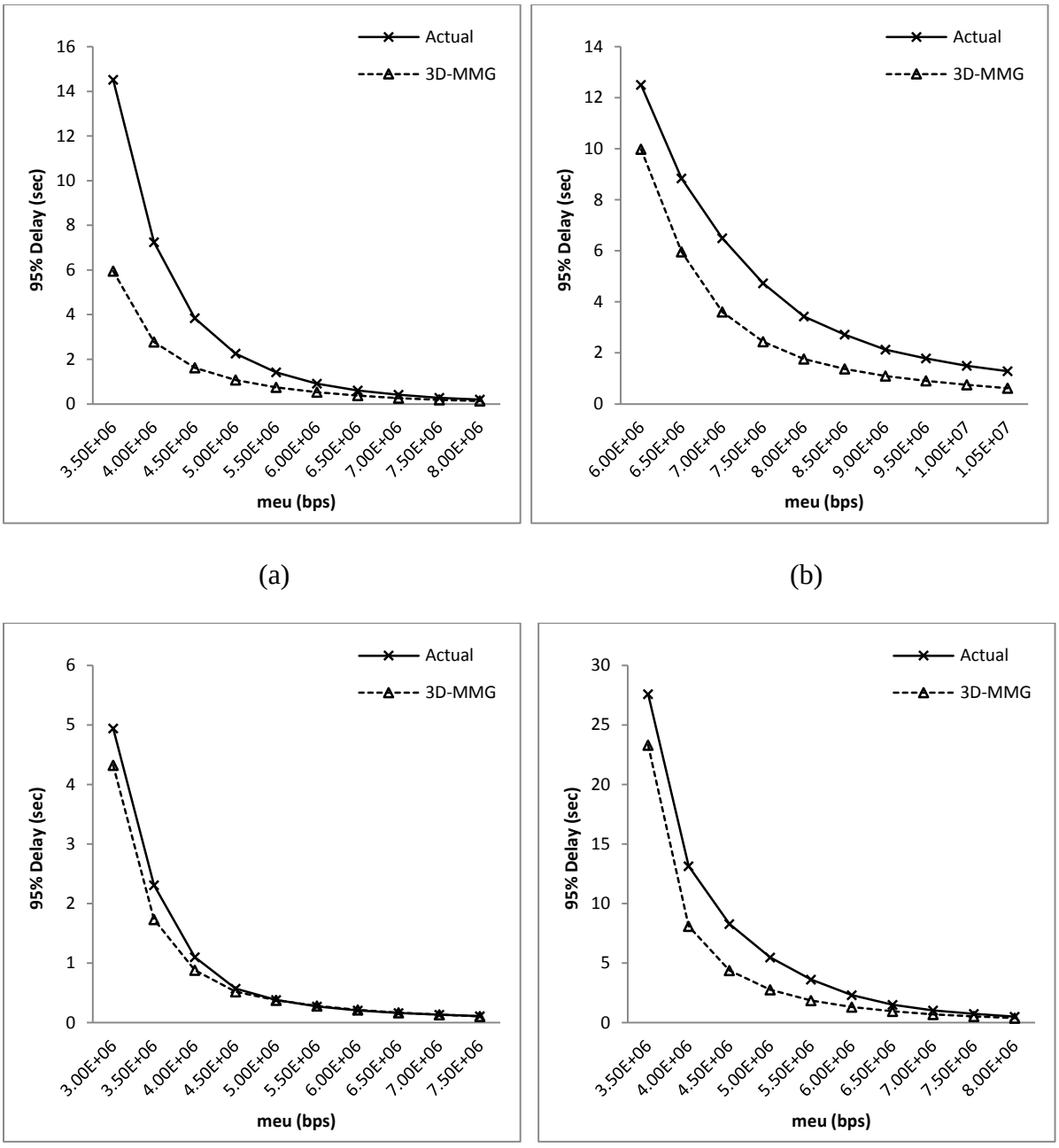


Figure 3.14: 95th percentile of the delay for the 3D-MMG model only for (a) Alice in Wonderland, (b) IMAX Space Station, (c) Monsters vs Aliens, (d) Clash of the Titans

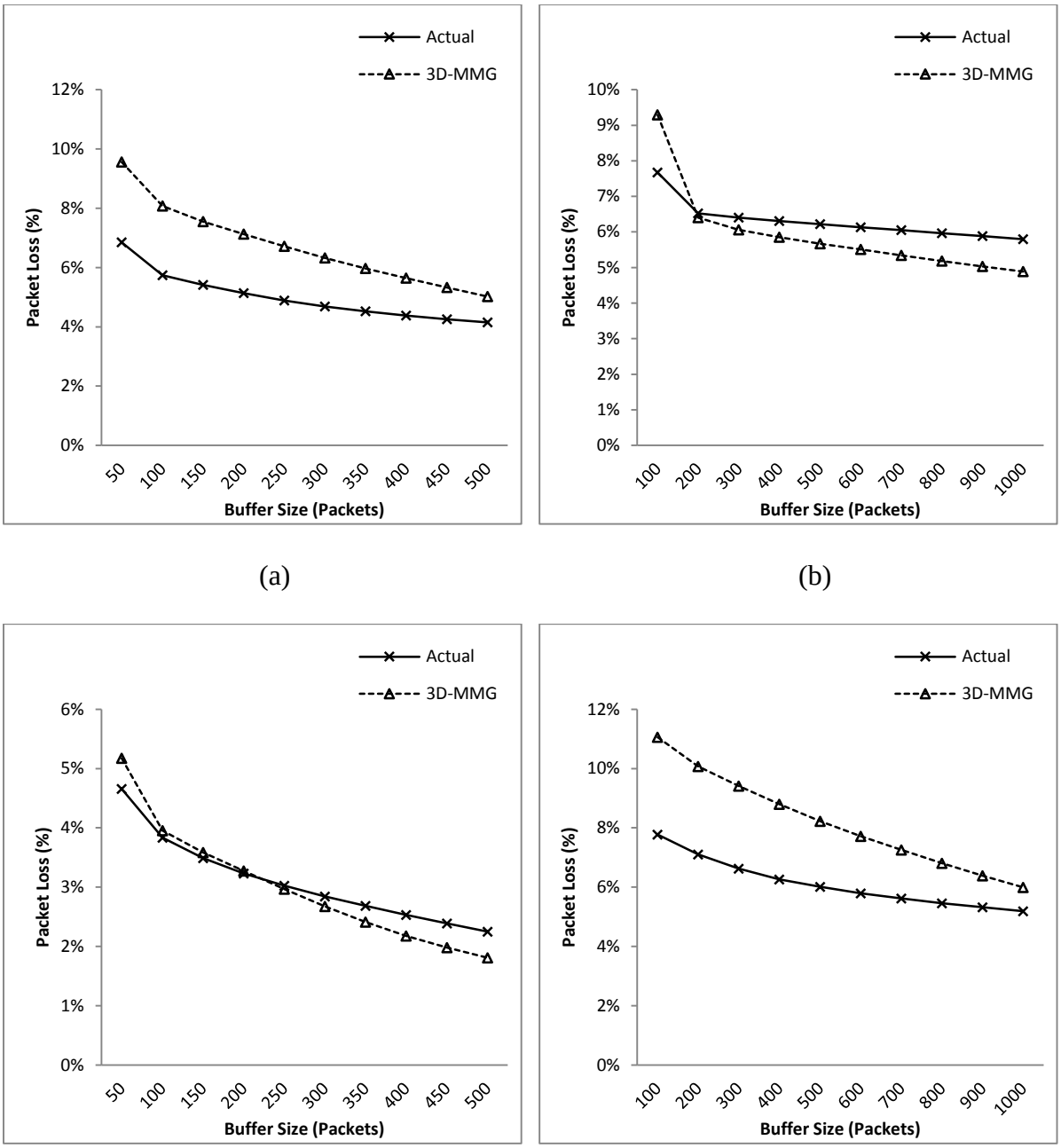


Figure 3.15: Packet Loss for the 3D-MMG model only for (a) Alice in Wonderland, (b) IMAX Space Station, (c) Monsters vs Aliens, (d) Clash of the Titans

1. Structure of the Model & Underlying Concept: The P-HMM model is extremely complicated, with a lot of parameters in play. The fundamental logic and mathematics behind it is complex, which makes the model computationally heavy as well. As we discussed earlier, the initial coarse estimates that need to be provided to the EM process at the beginning, especially the initial and transitional probabilities, must be refined to some extent. This reduces the amount of computations and iterations required by the EM algorithm. In some cases, this also avoids certain numerical instabilities.

On the other hand, the 3D-MMG model is comparatively simple and does not involve as many parameters as the P-HMM model. It is conceptually simple, can be easily executed and is not that heavy computationally.

2. Space Complexity of the Algorithm: The P-HMM model has a high space complexity. Each step of the m^{th} estimation of the EM estimation procedure depends on the probabilities calculated in the previous step. As a result, huge matrices of probabilities need to be maintained at every step, ranging to even four dimensions. The m^{th} estimate of the parameter set needs to be maintained for the $(m+1)^{th}$ estimation step of the EM algorithm, and so on. In this model, the space complexity is dictated by the largest matrix involved, which contains the backward probabilities, $\xi_n(i,j,k)$, and hence, can be stated approximately as $O(q^2r^2)$, where q is the number of GOPs present in the trace being used to train the model, and r is the number of states the model assumes in its estimation procedure.

On the other hand, the 3D-MMG model does not require to maintain a lot of parameters. Its space complexity is determined by the transition matrix lP or the connection matrix cP and can be stated as $O(r^2)$ approximately.

3. Time Complexity of the Algorithm: The P-HMM model is extremely complicated and comprises of several nested loops, resulting in even five iterations running concurrently, each one within the other, leading to a time complexity of $O(q^2r^3)$ approximately, where q is the number of GOPs present in the trace being used to train the model, and r is the number of states the model assumes in its estimation procedure. In this model, the time complexity is dictated by the following nested control structure.

for $n = N - 1 \rightarrow 1$ do

  for $i = 1 \rightarrow N_s$ do

    for $j = 1 \rightarrow N_s$ do

      for $k = N - n - 1 \rightarrow 0$ do

$$\xi_n(i,j,k) = \frac{\alpha_{n-1}(i,0)\pi_{ij}d_j[k]+\delta_i^j\alpha_{n-1}(i,k+1)}{\sum_{l=1}^{N_s}(\alpha_{n-1}(l,0)\pi_{lj}d_j[k]+\delta_l^j\alpha_{n-1}(l,k+1))} \times \gamma_n(j,k)$$

      end for

    end for

  end for

end for

On the other hand, the 3D-MMG model is far less complicated and consumes much less time. Its time complexity can be stated as O(q) approximately. This is because, here, the time complexity is dictated by the following segment of the algorithm.

```
while i ≤ m do
    Generate the next state for left view using the transition matrix lP for left view
    Generate the next state for right view using the connection matrix cP
    Generate one GOP within each state for each view
    i++
end while
```

On average, the P-HMM model requires around 3 hours of CPU time to be trained on a trace containing only 16,000 frames; while the 3D-MMG model consumes around 850 milliseconds to be trained on a trace containing 102,368 frames. These times are obtained on using a system with the given configurations:

Processor: Intel(R) Xeon(R) CPU E5645 @2.40GHz 2.40GHz (2 processors)

Installed Memory (RAM): 4.00GB

System Type: 64-bit Operating System

4. Range of Frame Sizes: Both the P-HMM and the 3D-MMG models are restricted by the frame sizes of the frames, present in the traces used to train the models. Neither model can successfully predict any frame size beyond the range dictated by the training video sequence.

As a result, they are not effectively scalable. However, in all fairness, all video traffic models, which have been published in literature until now, have similar drawbacks.

5. Number of Views: A 3D video usually consists of two views, namely the left and the right views, to produce the perception of depth. However, sometimes, more views may be required for the same purpose. Typically, the left view acts as the base view, which is encoded independently; while the remaining views are encoded on the basis of the left view. The P-HMM model is more general than the 3D-MMG model, in the sense that it can accommodate as many views as required. The model incorporates the parameter, N_{view}, which represents the number of views and can acquire any user-defined value. Whereas, the 3D-MMG model, as has been published until now, can only accommodate two views. However, this shortcoming can be overcome easily. As discussed earlier, in this model, we use the transition matrix lP to predict the states or shot classes of the independent view or the left view, while we use the connection matrix cP to predict the states or shot classes of the dependent view or the right view. Similarly, we may introduce an additional connection matrix for every additional dependent view we may want to incorporate. However, this may increase the model complexity, and also the model's accuracy is not known.

6. Numerical Instability: The P-HMM model, as described by the authors, can be safely assumed to have converged if the difference between the log likelihoods of the two most recent iterates or estimation steps is smaller than 0.01. However, during experiments, it has been often observed that the model becomes unstable and the log likelihood of consecutive

estimation steps starts oscillating between two values. This happens when a large number of GOPs have very close average frame sizes and the corresponding individual frame sizes are nearly equal. However, the model divides them into two significant fractions and assigns one to a state i and the other to the state i+1. The reason behind the oscillations is that a large number of frames are equally likely to fall in either state and hence, two very different estimates of the parameter set are valid. Consequently, the model cannot deterministically converge on any one of them .This occurrence indicates that the model performs accurately only when the corresponding pmfs defined by each state are very different from each other. Hence, the video trace, used to train the model, must possess considerable variations in its activity level, so that each state can be represented by drastically different pmfs for corresponding frames. The model is probably not suitable for traffic that does not demonstrate much variation.

Chapter 4

Conclusion

In this thesis we evaluated and compared the only two models proposed for H.264 MVC video for 3D video traffic, namely, the Poisson Hidden Markov Model (P-HMM) due to Rossi et. al. [28], and the Markov Modulated Gamma model (3D-MMG), due to Tanwir and Perros [33]. The results showed that the model generated traces were quite similar to the original trace in terms of frame size distribution and ACF, though the P-HMM model slightly under-estimated the frame size distribution, especially, for the right view. The 3D-MMG model predicted the three QoS metrics: 95^{th} percentile of the end-to-end delay, jitter and packet loss rate closer to the actual trace as compared to the P-HMM model. On the other hand, the P-HMM model was better able to predict traffic intensity. Also, the complexity and execution time of the P-HMM model is very high compared to the 3D-MMG model. This limits the size of the trace that the model can be trained on. As the synthetic trace generated by the model is highly dependent on the training sequence, if the training sequence is small and does not have enough activity, the model will also result in generating a similar trace with less activity that does not represent the actual traffic very accurately.

For future work, we plan to develop a new model for 3D video traffic using an Autoregressive Hidden Markov Model (HMM-AR). Quite a few Markov-modulated AR models have been proposed in literature, so far, but have been designed specifically for 2D video traffic only. Hence, we plan to explore HMM-AR models for characterizing 3D video traffic.

An HMM-AR model is a Hidden Markov Model (HMM) which modulates the parameters of an underlying Autoregressive (AR) model. In the case of video modelling, the AR model estimates the frame sizes of the video frames being generated. Specifically, each state of the HMM is associated with a different set of parameters that define the underlying AR process.

In an AR process, the current value is a function of a weighted linear combination of past values. An AR process is generally described as,

$$x(n) = c + \sum_{i=1}^{p} a_i x(n-i) + e(n),$$

where x(n) is the value generated at the n^{th} instant; c is a constant; $a_1, a_2, \ldots, a_p$ are the AR coefficients, with p being the order of the AR process. The sequence e(n) consists of independent and identical distributed (i.i.d.) random variables, known as residuals (or errors), that give the AR its stochastic nature. The residuals are uncorrelated and they are normally distributed with zero mean and variance σ^2. Thus, an AR is defined by the parameters: c, a_i's,

p and σ. In an HMM-AR model, these parameters take on a different value in each state of the HMM.

As an example, let us consider an HMM(3)-AR(2) model. This means that the model is described by an HMM with three states, and the underlying AR process is of order 2, i.e., the current value predicted by such an AR function depends on the previous two values. Thus, there are three different sets of parameters for each state and while in a given state s_i the next value is predicted using the AR(2) with parameters associated with this state. Consequently, we end up with the following three different AR functions corresponding to each state:

$$x(n) = c^{(1)} + a_1^{(1)}x(n-1) + a_2^{(1)}x(n-2) + e^{(1)}(n)$$

$$x(n) = c^{(2)} + a_1^{(2)}x(n-1) + a_2^{(2)}x(n-2) + e^{(2)}(n)$$

$$x(n) = c^{(3)} + a_1^{(3)}x(n-1) + a_2^{(3)}x(n-2) + e^{(3)}(n)$$

The superscripts (1), (2) and (3) of each of the parameters denote the state to which the corresponding AR function belongs.

In case of a video traffic model, the AR function of an HMM-AR generates the values of frame sizes. However, such frame sizes will be very similar to each other and a single AR function may not be suitable to predict all of I, P and B frames, which are very different from each other. A separate AR process for each of them may not be appropriate either as these would fail to capture the correlations between the I, P and B frames as well as the different

views. In such a case, we may require a Vector Autoregressive Hidden Markov Model (HMM-VAR). A VAR model is a multivariate generalization of the univariate AR model, which allows for more than one evolving variable. Each such variable is not only dependent on its own previous values but also on those of the other variables. For example, a VAR(1) with two variables can be described as,

$$\begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \begin{bmatrix} x_1(n-1) \\ x_2(n-1) \end{bmatrix} + \begin{bmatrix} e_1(n) \\ e_2(n) \end{bmatrix},$$

where x_1 and x_2 represent the two variables; c_1 and c_2 are constants; and $e_1(n)$ and $e_2(n)$ are the residuals that add the stochastic nature to the respective variables. $a_{1,1}$, $a_{1,2}$, $a_{2,1}$ and $a_{2,2}$ are the VAR coefficients that are organized as a matrix. The diagonal elements denote the dependency of each variable on its own previous values, i.e., $a_{1,1}$ represents the nature of dependency of x_1 on itself. The other elements denote the dependency of a variable on the previous values of the other variables, i.e., $a_{1,2}$ represents the nature of dependency of x_1 on x_2.

Let us now consider another example, an HMM(3)-VAR(1) model with two variables. This means that the model is described by an HMM with three states, and the underlying VAR process is of order 1, i.e., the current value of either variable predicted by such a VAR function depends on its own previous value and also that of the other variable. Thus, there are three different sets of parameters for each state and while in a given state s_i the next value is predicted using the VAR(1) with parameters associated with this state.

Consequently, we end up with the following three different VAR functions corresponding to each state:

$$\begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix} = \begin{bmatrix} c_1^{(1)} \\ c_2^{(1)} \end{bmatrix} + \begin{bmatrix} a_{1,1}^{(1)} & a_{1,2}^{(1)} \\ a_{2,1}^{(1)} & a_{2,2}^{(1)} \end{bmatrix} \begin{bmatrix} x_1(n-1) \\ x_2(n-1) \end{bmatrix} + \begin{bmatrix} e_1^{(1)}(n) \\ e_2^{(1)}(n) \end{bmatrix}$$

$$\begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix} = \begin{bmatrix} c_1^{(2)} \\ c_2^{(2)} \end{bmatrix} + \begin{bmatrix} a_{1,1}^{(2)} & a_{1,2}^{(2)} \\ a_{2,1}^{(2)} & a_{2,2}^{(2)} \end{bmatrix} \begin{bmatrix} x_1(n-1) \\ x_2(n-1) \end{bmatrix} + \begin{bmatrix} e_1^{(2)}(n) \\ e_2^{(2)}(n) \end{bmatrix}$$

$$\begin{bmatrix} x_1(n) \\ x_2(n) \end{bmatrix} = \begin{bmatrix} c_1^{(3)} \\ c_2^{(3)} \end{bmatrix} + \begin{bmatrix} a_{1,1}^{(3)} & a_{1,2}^{(3)} \\ a_{2,1}^{(3)} & a_{2,2}^{(3)} \end{bmatrix} \begin{bmatrix} x_1(n-1) \\ x_2(n-1) \end{bmatrix} + \begin{bmatrix} e_1^{(3)}(n) \\ e_2^{(3)}(n) \end{bmatrix}$$

The superscripts of each of the parameters denote the state to which the corresponding VAR function belongs. The number of variables forming the vector of the VAR process, the order of the VAR function, the number of states of the HMM-VAR, etc. are all subjects of future research.

An HMM-AR model can be used to predict the size of successive I frames. The rest of the frame sizes in a GOP of the 3D video traffic can be obtained using the same method as in the 3D-MMG model. An HMM-VAR model can be seen as a more general version of the HMM-AR model, whereby the I frame and the corresponding P frame, in the right view, can both be estimated at the same time. The remaining frames in the GOP can be estimated as in the 3D-MMG model. It is anticipated that the HMM-AR and HMM-VAR models will improve in terms of accuracy of a video traffic model.

REFERENCES

[1] A. A. Alheraish, "Autoregressive video conference models," *International Journal of Network Management,* vol. 14, no. 5, pp. 329-337, 2004.

[2] A. A. Alheraish, "A comparison of AR full motion video traffic models in B-ISDN," *Computers and Electrical Engineering Journal,* vol. 31, no. 1, pp. 1-22, January 2005.

[3] B. Anjum and H. Perros, "Bandwidth estimation for video streaming under percentile delay, jitter and packet loss constraints using traces," *Computer Communications Journal,* vol. 57, p. 7384, February 2015.

[4] T. P. -C. Chen and T. Chen, "Markov modulated punctured autoregressive processes for video traffic and wireless channel modeling," *Packet Video,* April 2002.

[5] Y. Chen, Y. Wang, K. Ugur, M. M. Hannuksela, J. Lainema and M. Gabbouj, "The emerging MVC standard for 3D video services," *EURASIP Journal on Advances in Signal Processing,* vol. 2009, p. 786015, 2009.

[6] G. Chiruvolu, T. K. Das, R. Sankar, et. al. "A scene-based generalized Markov chain model for VBR video traffic," *Proceedings of the IEEE Conference on Communications (ICC 98),* vol. 1, pp. 554-558, June 1998.

[7] M. Dai, Y. Zhang and D. Loguinov, "A unified traffic model for MPEG-4 and H.264 video traces," *IEEE Transactions on Multimedia,* vol. 11, no. 5, pp. 1010-1023, August 2009.

[8] A. P. Dempster, N. M. Laird and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc. B,* vol. 39, no. 1, pp. 1-38, 1977.

[9] N. D. Doulamis, A. D. Doulamis, G. E. Konstantoulakis and G. I. Stassinopoulos, "Efficient modeling of VBR MPEG-1 coded video sources," *IEEE Trans. Circuits Syst. Video Technol.,* vol. 10, no. 1, pp. 93-112, February 2000.

[10] Y. Ephraim and N. Merhav, "Hidden Markov processes," *IEEE Trans. Inf. Theory,* vol. 48, no. 6, pp. 1518-1569, June 2002.

[11] M. W. Garrett and W. Willinger, "Analysis, modeling and generation of self-similar VBR video traffic," *Proceedings of the ACM SIGCOMM,* pp. 269-280, August 1994.

[12] J. Greengrass, J. Evans and A. C. Begen, "Not all packets are equal, part I: streaming

video coding and SLA requirements," *IEEE Internet Computing,* vol. 3, no. 1, pp. 70-75, 2009.

[13] J. R. Hill and B. Melamed, "TEStool: a visual interactive environment for modeling autocorrelated time series," *Performance Evaluation,* vol. 24, no. 1-2, pp. 3-22, 1995.

[14] C. Huang, M. Devetsikiotis, I. Lambadaris, et. al. "Modeling and simulation of self-similar variable bit rate compressed video: a unified approach," *Proceedings of the ACM SIGCOMM,* pp. 114-125, August 1995.

[15] M. Krunz and A. Makowski, "Modeling video traffic using M/G/infinity input processes: a compromise between Markovian and LRD models," *IEEE Journal of Selected Areas in Communications,* vol. 16, no. 5, pp. 733-748, June 1998.

[16] H. Liu, N. Ansari and Y. Q. Shi, "Modeling VBR video traffic by Markov-modulated self-similar processes," *IEEE 3rd Workshop on Multimedia Signal Processing,* pp. 363-368, September 1999.

[17] A. Lombardo, G. Morabito and G. Schembra, "An accurate and treatable Markov model of MPEG-video traffic," *Proceedings of the 17th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '98),* vol. 1, pp. 217-224, March 1998.

[18] D. M. Lucantoni, M. F. Neuts and A. R. Reibman, "Methods for performance evaluatin of VBR video traffic models," *IEEE/ACM Transactions on Networking,* vol. 2, no. 2, pp. 176-180, April 1994.

[19] S. Ma and C. Ji, "Modeling heterogenous network traffic in wavelet domain," *IEEE/ACM Transactions on Networking,* vol. 9, no. 5, pp. 634-649, October 2001.

[20] A. Matrawy, I. Lambadaris and C. Huang, "MPEG-4 traffic modeling using the transform expand sample methodology," *4th IEEE International Workshop on Networked Appliances (IWNA '02),* pp. 249-256, 2002.

[21] B. Melamed and D. Pendarakis, "A TES-based model for compressed Star Wars video," *Proceedings of the IEEE GLOBECOM,* pp. 120-126, 1994.

[22] B. Melamed, D. Raychaudhuri, B. Sengupta, et. al. "TES based video source modeling for performance evaluation of integrated networks," *IEEE Transactions on Communications,* vol. 42, no. 10, pp. 2773-2777, 1994.

[23] R. Narasimha and R. M. Rao, "Discrete-time self-similar systems and stable distributions: applications to VBR video modeling," *IEEE Signal Processing Letters*, vol. 10, no. 3, pp. 65-68, March 2003.

[24] S. Park and D. Sim, "An efficient rate-control algorithm for multiview video coding," *IEEE 13th International Symposium on Consumer Electronics, 2009*, pp. 115-118.

[25] A. Pulipaka, P. Seeling, M. Reisslein and L. J. Karam, "Traffic and statistical multiplexing characterization of 3D video representation formats," *IEEE Transactions on Broadcasting*, vol. 59, no. 2, pp. 382-389, 2013.

[26] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257-286, February 1989.

[27] I. R. Richardson, The H.264 Advanced Video Compression Standard, 2nd ed., Wiley, 2010.

[28] L. Rossi, J. Chakareskia, P. Frossard and S. Colonnese, "A poisson hidden Markov model for multiview video traffic," *IEEE/ACM Transactions on Networking*, February 2014.

[29] M. Russell and R. Moore, "Explicit modeling of state occupancy in hidden Markov models for automatic speech recognition," *Proc. ICASSP, 1985*, vol. 10, pp. 5-8.

[30] U. K. Sarkar, S. Ramakrishnan and D. Sarkar, "Modeling full-length video using Markov-modulated gamma-based framework," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 638-649, 2003.

[31] A. Tamimi, R. Jain and C. So-In, "SAM: a simplified seasonal ARIMA model for mobile video over wireless broadband networks," *10th IEEE International Symposium on Multimedia (ISM '08)*, pp. 178-183, 15-17 December 2008.

[32] S. Tanwir and H. Perros, "A survey of VBR video traffic models," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 4, pp. 1778-1802, 2013.

[33] S. Tanwir and H. Perros, VBR video traffic models, Wiley-ISTE, 2014.

[34] "CiscoTelepresence". http://www.cisco.com/c/en/us/products/collaboration-endpoints/index.html.

[35] A. Vetro, A. M. Tourapis, K. Muller and T. Chen, "3D-TV content storage and

transmission," *IEEE Transactions on Broadcasting,* vol. 57, no. 2, pp. 384-394, 2011.

[36] A. Vetro, T. Wiegand and G. J. Sullivan, "Overview of the stereo and multiview video coding extensions of the H.264/MPEG-4 AVC standard," *Proceedings of the IEEE,* vol. 99, no. 4, pp. 626-942, April 2011.

[37] X. Wang, S. Jung and J. S. Meditch, "VBR broadcast video traffic modeling - a wavelet decomposition approach," *Proceedings of the IEEE GLOBECOM '97,* vol. 2, pp. 1052-1056, November 1997.

[38] "CiscoWebEx". http://www.webex.com/.

[39] W. Zhou, D. Sarkar and S. Ramakrishnan, "Traffic models for MPEG-4 spatial scalable video," *GLOBECOM 2005,* pp. 256-260, 2005.