5-2016

Toward a General Parametric Model for Assessing the Impact of Video Transcoding on Objective Video Quality

Nawaf Omar N. Alsrehin
Utah State University

TOWARD A GENERAL PARAMETRIC MODEL FOR ASSESSING THE IMPACT OF VIDEO TRANSCODING ON OBJECTIVE VIDEO QUALITY

by

Nawaf Omar N. Alsrehin

A dissertation submitted in partial fulfillment
of the requirements for the degree

of

DOCTOR OF PHILOSOPHY

in

Computer Science

Approved:

Stephen W. Clyde, Ph.D. Major Professor	Nicholas Flann, Ph.D. Committee Member
Curtis Dyreson, Ph.D. Committee Member	Amanda Lee Hughes, Ph.D. Committee Member
Bedri Cetiner, Ph.D. Committee Member	Mark R. McLellan, Ph.D. Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2016

ABSTRACT

Toward A General Parametric Model for Assessing the Impact of Video Transcoding on Objective Video Quality

by

Nawaf Omar N. Alsrehin, Doctor of Philosophy

Utah State University, 2016

Major Professor: Dr. Stephen W. Clyde
Department: Computer Science

Video transcoding can cause degradation to an original video. Currently, there is no general model that assesses the impact of video transcoding on video quality. Such a model could play a critical role in evaluating the quality of the transcoded video, and thereby optimizing delivery of video to end-users while meeting their expectations.

The main contribution of this research is the development and substantiation of a general parametric model, called the Video Transcoding Objective-quality Model (VTOM), that provides an extensible video transcoding service selection mechanism, which takes into account both the format and characteristics of the original video and the desired output, i.e., viewing format with preferred quality of service. VTOM represents a mathematical function that uses a set of media-related parameters for the original video and desired output, including codec, bit rate, frame rate, and frame size to predict the quality of the transcoded video generated from a specific transcoding.

VTOM includes four quality sub-models, each describing the impact of each of these parameters on objective video quality, as well as a weighted-product aggregation function that combines these quality sub-models with four additional error sub-models in a single function for assessing the overall video quality.

I compared the predicted quality results generated from the VTOM with quality values generated from an existing objective-quality metric. These comparisons yielded results that showed good correlations, with low error values.

VTOM helps the researchers and developers of video delivery systems and applications to calculate the degradation that video transcoding can cause on the fly, rather than evaluate it statistically using statistical methods that only consider the desired output. Because VTOM takes into account the quality of the input video, i.e., video format and characteristics, and the desired quality of the output video, it can be used for dynamic video transcoding service selection and composition.

A number of quality metrics were examined and used in development of VTOM and its assessment. However, this research discovered that, to date, there are no suitable metrics in the literature for comparing two videos with different frame rates. Therefore, this dissertation defines a new metric, called Frame Rate Metric (FRM) as part of its contributions. FRM can use any frame-based quality metric for comparing frames from both videos.

Finally, this research presents and adapts four Quality of Service (QoS)-aware video transcoding service selection algorithms. The experimental results showed that these four algorithms achieved good results in terms of time complexity, success ratio, and user satisfaction rate.

(196 pages)

PUBLIC ABSTRACT

Toward A General Parametric Model for Assessing the Impact of Video Transcoding on Objective Video Quality

Nawaf Omar N. Alsrehin

The ultimate goal of any video delivery system is to allow any user to watch any video of any kind on any display device over any type of network with a desired output, i.e., viewing codec with preferred quality of service. This could theoretically require 10^{32} video transcoding functions that convert any original video to any desired output. Guaranteeing a required format and preferred quality of service of the perceived video requires selecting or composing a set of transcoding functions that satisfy the requested format and preferred quality of service. An effective way to accomplish this is by allowing the selection and composition mechanisms to take place based on a model that accurately assesses the impact of each transcoding function on video quality. Using such a model, each user will receive the requested video based on the required format and preferred quality of service.

The main contribution of this research is the development and substantiation of such a model, called Video Transcoding Objective-quality Model (VTOM), that provides an extensible video transcoding service selection mechanism, which takes into account both the format and characteristics of the original video and the desired output. VTOM represents a mathematical function that uses a set of media-related parameters for the original video and desired output, including codec, bit rate, frame rate, and frame size to predict the quality of the transcoded video generated from a specific transcoding function.

VTOM includes four quality sub-models, each describing the impact of one of these parameters on objective video quality, as well as a weighted product aggregation function that combines these quality sub-models with four additional error sub-models in a single tool for assessing the overall video quality. I compared the predicted results from the VTOM with quality values generated from an objective video quality metric. These extensive comparisons yielded results that showed good correlations, with low error values.

Because no suitable metrics exist in the literature that evaluate the relative quality of two videos that have different frame rates, this research presents and develops such a metric, called Frame Rate Metric (FRM). FRM uses any frame-based objective quality metric to compare two videos. This research also presents a strategy that helps in evaluating the relative quality of two videos that have different frame sizes.

This research also presents and adapts four QoS-aware video transcoding service selection algorithms. Each of them selects the "best-fit" video transcoding service from a pool of available ones. This selection satisfies the requested format and desired quality of service. The evaluation results showed the effectiveness and the efficiency of these candidate algorithms.

As a consequence from this dissertation, the researchers and developers of video delivery systems and applications have a model to calculate the degradation that each transcoding function can cause rather than statistically evaluate it. Current statistical methods consider only the desired quality of service. This can lead us to conclude that VTOM can improve video transcoding selection and composition algorithms.

I dedicate this work to the soul of my brother, Baha'a Omar Alsrehin.

ACKNOWLEDGMENTS

This dissertation would not have been possible without the mercy of Almighty, Omnipotent and Omniscient, to whom I express my gratitude.

First, I would like to express my sincere gratitude to **Dr. Stephen W. Clyde**, my advisor, for his patience, motivation, immense knowledge, tremendous support, constant guidance, constructive feedback, periodical extensive discussions, and invaluable understanding that have enabled me to complete this dissertation. His guidance has continually helped me in the research and writing of this dissertation. I could not have imagined having a better advisor and mentor for my Ph.D. study. I hope that he will support me morally and technically for the rest of my life.

I would like to thank the respected members of my dissertation committee, **Dr. Nicholas Flann**, **Dr. Curtis Dyreson**, **Dr. Amanda Lee Hughes**, and **Dr. Bedri Cetiner**, for their helpful discussions concerning my work. Their insightful comments on my Ph.D. proposal greatly helped in improving my work. I am also grateful to them for carefully reading my dissertation and providing helpful feedback.

I gratefully acknowledged the funding sources that made my Ph.D. work possible. I was funded by Yarmouk University and the Computer Science Department at Utah State University.

I am grateful for the support and assistance from **Dr. Al Forsyth** and the English Writing Center at Utah State University for reviewing my dissertation.

I would like to thank my family: my parents (**Omar Alsrehin** and **Najah Alzoubi**), my brothers (**Ala'a** and **Dea'a**), and my sisters (**Ghada**, **Manar**, and **Nowar**) for their love, encouragement, constant prayers (doas), guidance, and unconditional affection that made all of this possible.

Last but not the least, I would especially like to thank my wife (**Nosaybah Alzoubi**) for her patience and unending support during the most challenging periods of my Ph.D. research. Thanks to my daughters: **Ayah**, **Yarah**, **Zainah**, **Hala**, and **Saba**, who have missed me a lot during my study.

Nawaf Omar Alsrehin

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

QoS	Quality of Service
IoTs	Internet of Things
PSNR	Peak Signal-to-Noise Ratio
SSIM	Structural Similarity
VSSIM	Video Structural Similarity
MS-SSIM	Multi-Scale Structural Similarity
QCIF	Quarter Common Intermediate Format
UHD	Ultra High Definition
SUHD	Super Ultra High Definition
VLD	Variable Length Decoder
DCT	Discrete Cosine Transform
IQ	Inverse Quantized
IDCT	Inverse Discrete Cosine Transform
MC	Motion Compensation
FM	Frame Memory
PCA	Principal Component Analysis
ACO	Ant Colony Optimization
SOA	Service Oriented Architecture
PCC	Pearson Correlation Coefficient
RMSE	Root Mean Square Error
TCF	Temporal Correlation Factor
MOS	Mean Opinion Score
DMOS	Difference Mean Opinion Score
QoE	Quality of Experience
GOP	Group of Pictures

LSLO	Local Selection and Local Optimization
IQA	Image Quality Assessment
VQA	Video Quality Assessment
LCM	Least Common Multiple
SAD	Sum of Absolute Differences
FR	Full Reference
RR	Reduced Reference
NR	No Reference
VTOM	Video Transcoding Objective-quality Model

CHAPTER 1

INTRODUCTION AND MOTIVATION

1.1. Introduction

The unprecedented growth of ubiquitous communication infrastructure and cloud-based video delivery systems, the increased number of end-user devices ranging from desktop computers to smart phones, and the improvements in display characteristics, computational power, and storage have all facilitated the general public being able to create, capture, and access video content at any time and from anywhere. Cisco® predicts that there will be more than 50 billion devices connected to the Internet of Things (IoTs) by 2020 [1]. In 2013, video traffic accounted for 66% of all consumer Internet traffic, and Cisco® predicts that this will grow to 79% by 2018 [2]. Moreover, Cisco® predicts that Wi-Fi and mobile devices will account for 61% of Internet traffic by 2018, from 44% in 2013.

However, delivering video content to the end-users based on their requirements, i.e., required video format, and preferences, i.e., desired QoS, is a challenge. Storing, transcoding, and transmitting video content are still expensive processes due to the massive amount of data required for digital video, the heterogeneity of end-user devices and QoS demands, and the variety of video formats.

Another challenge is transmitting video content over network routes that have limited or unpredictable bandwidth and are subject to congestion problems. This challenge is especially evident for the "last mile," which is the link in a network route that connects directly to the end-user's device. The "last mile" is often over a mobile telecommunication link, like 4G, or a wireless network.

A third challenge is that consumer eyes are better trained than ever, and they expect higher quality with faster services. These trends lead us to believe that the demand for a responsive video delivery is going to increase in the near future.

A video is a media file that has a video codec or format and video characteristics. A video coding format defines the structure of the video's image and audio data. Some popular formats are H.264, MPEG-

4 part 2, MJPEG (Motion JPEG), WMV, and DivX, but there are over 20 in common use today [3]. A codec is a piece of software that can encode video data into a particular format or decode a video from that format. A codec's name is often used as a synonym for the format with which it works. Video characteristics include frame size, i.e., width and height, frame rate, bit rate, and audio sampling rate and all characterize a video [4]. However, since the audio portion of the video forms only a small part of the data, it is often not considered in QoS-related decisions. Therefore, without compromising the proposed model, I have excluded it from this research.

For this research, quality is a measure of how a transcoded video looks compared to the original. This measure can be evaluated either objectively or subjectively [5]. Objective evaluation techniques, such as Mean Square Error (MSE) [6], are mathematical models that approximate expert judgments. Since they are mathematical models, a computer program can automatically calculate them. Subjective evaluations, on the other hand, require expert judgments. Video quality can be affected by changing the codec or adjusting any of the video characteristics mentioned above. Objective video quality is a measure of a quality that is evaluated using an objective-quality metric. This research uses three full-reference[1] objective-quality metrics to evaluate the quality of the transcoded videos. See Appendix B for more detail.

Desired QoS represents the end-user preferences, which includes bit rate, frame rate, and frame size. Calculating the desired QoS is an interesting human-factors and device management problem, but it is outside the scope of this research.

Besides storing and transmitting video content, video delivery systems convert the original video, which was captured in some specific format and at a certain quality, to match the end-user's required viewing format and desired QoS. A *video transcoder* is a piece of software that does this conversion [7]. To convert both video format and characteristics, a transcoder typically uses a two-step approach. First, it decodes the incoming video into a raw uncompressed format. Then, it re-encodes the uncompressed format into the required output format and characteristics that meet the desired QoS. See Figure 1.1.

---

[1] Full-Reference (FR) metrics are metrics that compute the quality difference by comparing the original video signal against the transcoded video signal, in which every pixel from the source is compared against the corresponding pixel at the transcoded video. In FR metrics, both the original and transcoded videos should be available [5].

Video transcoding may occur entirely before the streaming begins, e.g., Amazon Elastic Transcoder®, or it may be integrated into the streaming process, e.g., Akamai Media Content Delivery®. This research focuses on video transcoding, and it does not matter whether the conversion occurs before streaming or in a pipeline with the streaming. Chapter 2 and Appendix A provide more information about video transcoding techniques and types.

A video delivery system may have thousands of different transcoding functions, but even then, those will only represent a fraction of possible mappings. For example, assume that there are 20 different coding formats. Also, assume that there are over 5 million different frame sizes (ranging up to 4096x2160), 10 different frame rates, and approximately 10 million different bit rates. This means there are 5^{14} possible quality settings or levels. Together, there would be 10^{16} possible combinations of formats and qualities and 10^{32} possible mappings. In reality, most of these combinations and mappings are not interesting. Still, it is likely for an end-user to request a format and quality for which the delivery system has no transcoder that provides the desired mapping exactly. In such cases, a composition of multiple transcoding functions is necessary.

A *transcoder* is a software implementation of one or more transcoding functions. It is common for a single transcoder to handle many different mappings that all have the same input and output formats. Given a video in the format f_1 and a required output format f_2, a *compatible transcoder* is a transcoder that accepts a video with a format of f_1 as input and directly generates a video of format f_2. In other words, a compatible transcoder for f_1 and f_2 implements a set of video transcoding functions that differ in terms of

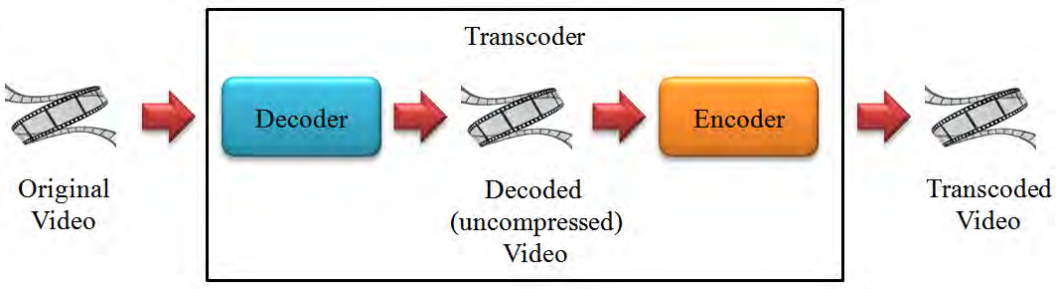


Figure 1.1. Video transcoding.

the input and output video characteristics. If the video delivery system has a compatible transcoder that matches the input video and end-user requirement, then it would select a transcoding function from that transcoder whose output closely matches the end-user preferences, i.e., desired QoS.

If the video delivery system does not have a compatible transcoder, the system will need to compose two or more transcoding functions implemented in different transcoders. The first transcoding function would convert the original video into an intermediate format that would then become the input for the next transcoding function. The output of that function would be either the desired output or the input to a third transcoding function, and so forth until the requested video content satisfies both the requested format and desired QoS. With a composition of transcoding functions, each function in the sequence can degrade the quality of the original video content.

Even with a single transcoding function, assessing the potential degradation to quality is a challenging task for three reasons. First, each video characteristic can introduce loss of information, albeit in a different way, which alters the end-user's perception of the video. For example, reducing the frame rate might affect the motion level, while reducing the bit rate might affect the frame quality and thereby the overall quality. Second, the video characteristics are not independent of each other. For example, a change in a frame rate or a frame size may necessitate a change in a bit rate. Third, there are lower limits to the video characteristics, beyond which the video becomes unwatchable. For example, reducing the frame rate below 5 frames per second (fps) might interrupt the perception of smooth motion. There are also upper limits for specific formats, but this is not considered a significant source of degradation.

To address the above challenges, this research develops a general parametric model, called the Video Transcoding Objective-quality Model (VTOM), that assesses the impact of video transcoding on objective video quality. VTOM also aims to assess the quality of the transcoded video that is generated from a single transcoding function. In addition, VTOM helps not just in the selection of a single transcoding function. It also provides the foundation that can be extended, in a future work, to help in composing a set of transcoders. This extension also can help in assessing the quality of the transcoded video that is generated from composing a set of transcoding functions. Chapter 8 describes the future work in more detail.

Generally, parametric models predict the perceived quality based on a set of parameters that are related to the encoding or transcoding distortions, video content, motion level, and/or transmission process [8]. Parametric models are easy to implement since there is no need to fully access the original or transcoded videos. The VTOM uses a set of media-transcoding parameters, which includes the video format and characteristics mentioned above.

To evaluate the VTOM and its sub-models, I measured the degree of correlation between the predicted quality that the VTOM generates and the quality measures from an existing objective-quality metric, called Multi-Scale Structural Similarity (MS-SSIM) [9]. I computed the correlation by using the Person Correlation Coefficient (PCC), and I measured the degree of prediction by using the Root Mean Square Error (RMSE). Appendix B provides more detail about this evaluation strategy.

1.2. A Typical Example for Video Delivery with Transcoding

Figure 1.2 shows an example that makes use of video transcoding selection and composition processes in a distributed cloud environment. This example may be considered typical of video delivery systems, such as YouTube®[2], Netflix®[3], Akamai®[4], or Hulu®[5], where a video is transcoded in different steps according to the end-user's viewing capabilities and the bandwidth of network route, particularly the "last mile."

In this example, user A uploads video content to the cloud using a personal computer. Then users B, C, and D request and play that video using a laptop, a smart TV, and a smart phone, respectively. If the original video format that is uploaded by user A is not supported by user B, C, or D's device or playback software, then this video must be transcoded to a new format that is supported by user B, C, or D's device. Furthermore, users B, C, and D may have different network limitations and therefore different desired QoS's.

[2] https://www.youtube.com/
[3] http://www.netflix.com/
[4] https://www.akamai.com/
[5] http://www.hulu.com/

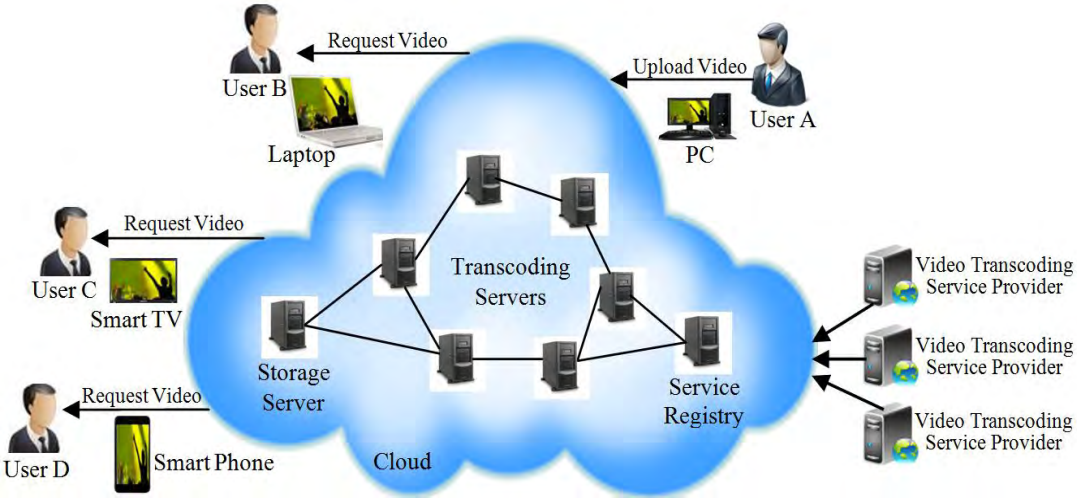


Figure 1.2. An example of video delivery system that makes use of video transcoding selection and composition processes in a distributed cloud environment.

Imagine that user A uploads a video in the MPEG-2 format with a frame rate of 30 fps, a frame size of 1920x1080, and a bit rate of 6.0 Mbps. User B may want that video in the MJPEG format with 25 fps as a frame rate, 1280x720 as a frame size, and 4.4 Mbps as a bit rate based on his device and network bandwidth capabilities. However, user C may want that video in the H.264 format at 20 fps as a frame rate, 640x480 as a frame size, and 1.8 Mbps as a bit rate. Similarly, user D may want that video in a third format and at another QoS. To optimize transmission while still satisfying user expectations, a video delivery system will transcode the original video three times, one for each of these users.

If the cloud-based video delivery system has compatible transcoders, i.e., from MPEG-2 to H.264, then the problem becomes selecting a transcoding function whose output best matches the desired QoS. If the cloud-based video delivery system does not have any compatible transcoders, then the system needs to compose multiple transcoding functions.

Assume that the cloud-based video delivery system shown in Figure 1.2 has transcoders that convert from MPEG-2 to MJPEG (type T_1) and MJPEG to H.264 (type T_2), but does not have any transcoder that directly converts MPEG-2 to H.264. Each transcoder might have hundreds of different transcoding functions, each with different QoS specification values.

For user B, the delivery system must select a transcoding function from T_1 whose output best matches the desired QoS. For user C, the delivery system must select and compose transcoding functions from types T_1 and T_2, where the output of the first transcoding function becomes an input to the second. It is conceivable that scenarios exist where the delivery system must compose three or more transcoding functions to generate a video in the required format and desired QoS, but I believe that such situations are rare and do not alter the fundamental problem of selection and composition.

When a single transcoding function or a composition of functions is necessary, each transcoding function can degrade the quality of the original video. Degradation is a loss of information that negatively affects the end-user experience, so the goal of the selection and composition is to keep that degradation as low as possible.

1.3. Contributions

The example illustrated in Figure 1.2 demonstrates the need for a general model that assesses the impact of video transcoding on video quality. To accomplish this, VTOM provides an extensible video transcoding service selection mechanism that takes into account the format and characteristics of the input video, the format required by the video playback software, and the desired QoS. For convenience, we refer to these as the *transcoding parameters*.

To measure the impact of each of these parameters on objective video quality, I developed four quality sub-models, one for each of them. In addition, to measure the weight, i.e., affect, of each of these parameters in the overall quality, I developed four additional error sub-models. VTOM combines these quality and error sub-models in a single tool to predict the overall quality of the transcoded video generated from a single video transcoding function. Specifically, VTOM represents a weighted product aggregation function where the estimated perceived quality is computed as a result of a direct mathematical function based on the values of these transcoding parameters. This mathematical function generates a value between 0 and 1, the higher the value, the better the quality.

To develop the VTOM, I used a set of raw, uncompressed videos from the VQEG Phase 1 2010 video data set [10] to generate a set of encoded videos. Then I transcoded these encoded videos to generate

a new set as a training set of transcoded videos by changing the video codec and reducing the values of each of the bit rate, frame rate, and frame size during video transcoding. After that, I used the MS-SSIM as an objective quality metric to compare the encoded and transcoded videos and recorded the quality results. I aggregated these quality results in different ways to explore the impact of changing each of these parameters on objective video quality. I used these aggregated results as reference values to develop the VTOM and its sub-models. Chapter 5 and Appendix C describe the exploring phase in more detail. Chapter 6 and Appendix D present the development phase in more detail.

To evaluate the VTOM, I used another set of raw, uncompressed videos from the VQEG Phase 1 2010 video data set [10] to generate a new set of encoded videos. Then I followed similar steps described above to generate a new set as a testing set of transcoded videos. After that, I used the MS-SSIM to compare the encoded and transcoded videos and recorded the quality results. I compared these quality results with the predicted quality values generated from the VTOM. These extensive comparisons yielded results that showed good correlations with low error values. Chapter 6 and Appendix D present and describe the evaluation process in more detail.

In accomplishing the above main contribution, this dissertation provides the following specific contributions:

1. It explores and analyzes the impact of video encoding on objective video quality. I used nine original, raw, uncompressed videos from the VQEG phase 1 2010 video data set [10] to generate around 189 encoded videos. I encoded these original videos using three different video codecs, i.e., H.264, MPEG-4, and FLV, at seven different bit rate values ranging from 2 to 20 Mbps. I used the following three full-reference objective quality metrics to evaluate the quality of the encoded videos: Peak Signal-to-Noise Ratio (PSNR) [11], Structural Similarity (SSIM) [12], and MS-SSIM [9]. See Appendix B for more detail about these quality metrics. The quality evaluation results showed that the H.264 codec encodes the original videos in a better quality than the MPEG-4 and FLV codecs at a given bit rate. Chapter 5 describes this encoding and its quality results in more detail.

2. It explores, analyzes, and models the impact of each of the following on objective video quality in video transcoding: a) changing the video codec, b) reducing the bit rate, c) reducing the frame rate, and d) reducing the frame size. I used five original, raw, uncompressed videos from the VQEG phase 1 2010 video data set [10] using the three codecs mentioned above, three different frame rates, i.e., 25, 20, and 15 fps, three different frame sizes, i.e., 1440x900, 1280x720, and 640x420, and seven different bit rates, i.e., 2, 4, 6, 8, 10, 15, and 20 Mbps, to generate around 1155 transcoded videos. Chapter 5 and Appendix C present and describe the exploring and analysis steps in more detail. Chapter 6 and Appendix D present and describe the modeling step in more detail. This modeling step generates four parametric quality sub-models, each of which handles changing or reducing a specific transcoding parameter. To evaluate these models, I used four different videos from the same video data set mentioned above to generate in total 924 transcoded videos as a testing set. I compared the predicted quality results that are generated from each of these quality sub-models with quality values generated from MS-SSIM. These extensive comparisons yielded results that showed high correlations, with low error values.
3. It presents and substantiates four parametric error sub-models that assess the error in evaluating the impact of changing or reducing each of the above transcoding parameters on objective video quality in video transcoding. I used the error sub-models to calculate the weight of each transcoding parameter in the overall quality. To develop and evaluate these error sub-models, I used the testing set mentioned above that generates 924 transcoded videos. The evaluation results showed high correlations between the results that are generated from the error sub-models and the actual error values that I got from the difference between the quality sub-models' results and the actual quality values that I got from using the MS-SSIM metric [9]. In addition, the evaluation results showed low error values. Moreover, the evaluation results showed that normalizing the error generated results that decreased the PCC values and increased the RMSE of the VTOM. Chapter 4 formally defines the error in

assessing the quality. Chapter 6 and Appendix D present and describe these error sub-models in more detail.

4. It presents and develops a new metric, called the Frame Rate Metric (FRM), for evaluating the relative quality of two videos that have different frame rates. FRM uses any frame-based quality metric for comparing frames from both videos. FRM represents an initial step toward developing a more robust metric. Chapter 5 describes this metric in more detail. It also presents a strategy that helps in evaluating the relative quality of two videos that have different frame sizes. This strategy uses any frame-based objective quality metric for comparing frames from both videos.
5. It presents and adapts four QoS-aware video transcoding selection algorithms [13]. Each of them selects the "best-fit" video transcoding function from a pool of available ones. This selection satisfies the requested format and comes as close as possible to satisfy the desired QoS. The evaluation results showed the effectiveness and the efficiency of these candidate algorithms in terms of time complexity, success ratio, user satisfaction rate, and recall and precision. Chapter 7 describes these algorithms and the evaluation process and results in more detail.
6. It presents a general model for a cloud-based video distribution system [13] [14]. This system contains three sub-systems: a) a cloud-based video management system, b) a cloud-based video transcoding system, and c) a cloud-based video streaming system. Chapter 7 provides more detail on these systems.
7. As a technical contribution, it provides an implementation of a set of 780 video transcoding functions that handle different combinations of different values of video transcoding parameters. This implementation is based on Java® SDK 1.8 and Xuggler© 4.5.

As a consequence of this dissertation, the researchers and developers of video delivery systems can use VTOM to calculate the degradation of video transcoding function dynamically rather than statistically evaluate it. Thus, VTOM can improve video transcoding selection and composition algorithms. Also, it can help video delivery systems meet end-user requirements and preferences, while optimizing

transmission. Moreover, it opens opportunities for researchers to propose extensions to VTOM or similar models that consider additional video characteristics and end-user requirements or preferences.

The results from this research lead us to believe that further subjective studies would be of great interest and value. Also, considering additional characteristics, requirements, and preferences could prove to be very beneficial to a wide range of video-based systems and applications.

CHAPTER 2

BACKGROUND AND RELATED RESEARCH

To appreciate the complexities of selecting an appropriate video transcoding function or a sequence of functions, it is important to understand the fundamentals of video transcoding. This chapter summarizes the key concepts related to video transcoding techniques. Appendix A provides more detail on specific transcoding types. This chapter also presents state-of-the-art-research that is related to the problem domain organized into the following four areas: a) multimedia selection and composition, b) models for perceptual video quality estimations, c) the impact of video encoding on subjective and objective video qualities, and d) the impact of video transcoding artifacts on subjective and objective video qualities.

2.1. Video Transcoding

The recent evolution of multimedia services and applications has accelerated the development of video transcoding technology for four different reasons. First is the increased number of video display formats, ranging from Quarter Common Intermediate Format (QCIF) for small handheld devices to Super Ultra High Definition (SUHD) or 8K for large screens. Second is the increased number of end-user devices that can consume the same video content. For example, in an effort to provide a representative set of devices, the Xamarin Test Cloud®[6] currently supports over 1,000 different devices, and that number is growing by 100 every month [15]. Third is the increased range of network bandwidths, ranging from less than 1 Mbps to more than 1000 Mbps for wireless and wired networks [16]. Fourth is the increased number of video formats that define the structure of the video's image and audio data [3]. The release of a new video format in the market has the potential problem of ensuring interoperability with other existing formats. No one would start a video delivery business based on a single video format, because users are very reluctant to lock themselves into using a single format. Therefore, supporting diverse video transcoding capabilities is a critical requirement for any video delivery system.

[6] https://xamarin.com/

As mentioned earlier, the improvements in the end-user devices in terms of processing power, display characteristics, computational power, and storage have all facilitated the general public being able to create and capture high-quality video content. For example, iPhone 6s plus® supports video capturing and playing at 4K resolution, and records SLO-MO video in 1080p HD at 120 fps, or in 720p at 240 fps [17]. These videos are not suitable for transmission over networks with limited bandwidth, such as those currently available via cell phone technology. Video transcoding can solve the problem by reducing the amount of data that needs to be transmitted by changing the video format and characteristics. For this reason, video transcoding is often referred to as either *video adaptation* or *video repurposing* [18].

As mentioned earlier and shown in Figure 1.1, the most popular approach to video transcoding is a two-step approach in which the decoder first converts the original compressed video into an intermediate uncompressed format, and then the encoder converts this intermediate format into a compressed target format [7]. See Figures 2.1 and 2.2. These figures show a simple architecture for a decoder and encoder, called *cascade video transcoder*, which takes as input a set of transcoding parameters, such as video format, i.e., codec, spatial resolution, i.e., frame size, temporal resolution, i.e., frame rate, and bit rate [19]. I used this simple architecture in this research, which allows transcoding the original video without introducing a significant distortion in the image quality [20].

In the cascade video transcoder architecture, the Variable Length Decoder (VLD) first decodes the original bit stream. Then the decoded Discrete Cosine Transform (DCT) coefficients are inversely quantized (IQ_1) and transformed (IDCT), producing a copy of the original coded pixel. Motion Compensation (MC) is performed on the intra-coded images to help in estimating the motion of the enter-

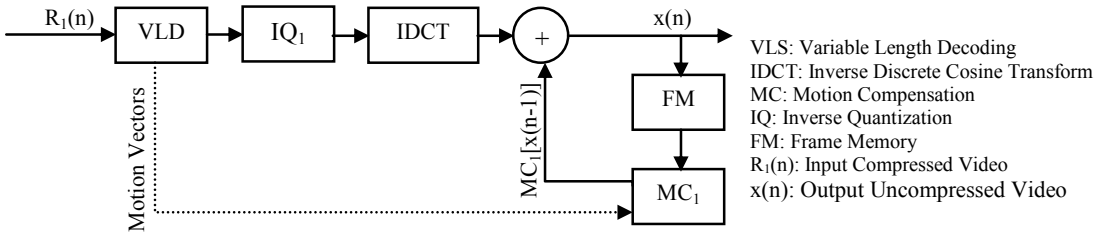


Figure 2.1. The video decoder [19] [20].

coded images and then the decoded frames are stored in the Frame Memory (FM) in order to keep a local copy of the frames.

The decoder accepts a compressed bit stream $R_1(n)$ as an input and produces an uncompressed video stream x(n) as an output. The uncompressed video is then re-encoded with new encoding parameters defined by the target bit stream. Figure 2.2 shows that the video encoder performs even more complex operations than video decoding. Video encoding consists of a set of other operations, such as discrete cosine transformation, quantization, variable length coding, and motion compensation [21] [20].

Video transcoding is usually done in the following cases [7] [22]:

- To allow a target device (or playback software) that has limited storage capacity, supports different format, or has limited network bandwidth to display any original video content.
- To convert incompatible or obsolete data to a better-supported or modern format.
- To convert multimedia content that is originally designed for a particular device, platform, user, or format, to be suitable for another device, platform, user, display capability, processing power, or even network bandwidth.
- To reduce the diversity in communication technology and infrastructure along with enhancing the ways users consume video content.
- To cope with heterogeneous communication environments, networks, user devices, user demands, and applications.

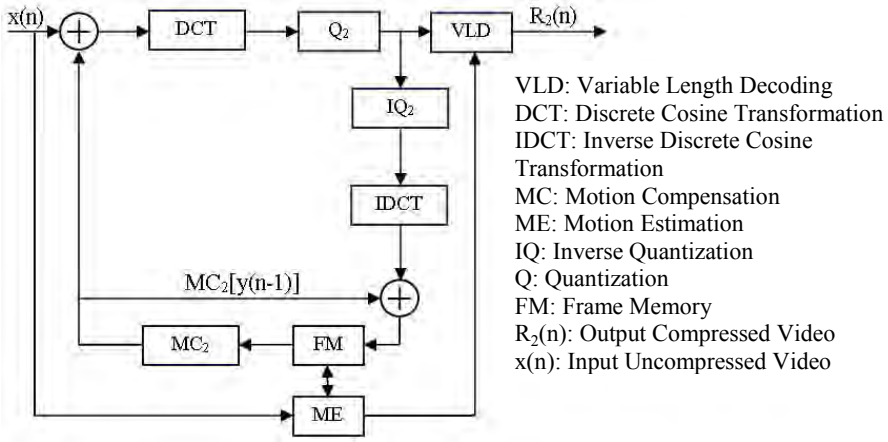


Figure 2.2. The video encoder [19] [20].

2.2. Video Transcoding Techniques

Based on the parameters that a transcoder accepts, it provides several functions, including adjustment of bit rate, frame size, frame rate, and video codec conversion [23]. Some of them can take place in different techniques, such as [18]:

- Conversion of video codec, e.g., MPEG-2 to H.264.
- Conversion of transcoding parameters:
    - Bit rate, e.g., from 2Mbps to 800 kbps.
    - Frame rate, e.g., from 30 fps to 25 fps.
    - Frame size, e.g., from 1920x1080p to 1280x720.

Below are brief descriptions of some of these techniques [18] [21] [23]:

2.2.1. Homogenous Transcoding

In homogenous transcoding, the format of the output video is the same as the input video [21]. This type of transcoding only adjusts the frame rate, bit rate, and frame size.

2.2.2. Heterogeneous Transcoding

In heterogeneous transcoding, the video content is transcoded between different video codecs, such as MPEG-2 to MPEG-4, MJPEG to H.264, etc [21]. In addition to the codec conversion, this technique of transcoding may additionally adjust any format parameters that are combined with other transcoding functions whenever needed, such as picture type conversion, frame rate conversion, bit rate conversion, frame size or resolution conversion, or the direction of Motion Vector (MV) to match the device, player, or bandwidth requirements.

In contrast to homogenous transcoders, the heterogeneous transcoders require more computation and add more complexity due to the asymmetry between the decoder and encoder [21]. The heterogeneous transcoders perform a full decoding up to the pixel domain and store all the embedded information, such as motion information, picture and macroblock types to be reused at the encoder side. Previous research has

made efforts to develop and enhance heterogeneous transcoding. Liange et al. [24] presented a heterogeneous video transcoding technique that transcodes between MPEG-4 and H.263.

2.2.3. Bit rate Transcoding

Bit rate transcoding, also referred to as *transrating*, was one of the first transcoding applications [21]. This type of transcoding is required in transmission of video content over heterogeneous low-bandwidth wireless networks when the capacity of the outgoing channel is less than that of the incoming channel [18]. More specifically, this type of transcoding is required when transmitting high-quality video content to multiple wireless clients through a channel with varied capabilities [18]. There is a trade-off between reducing the size of the video bitstream and keeping that stream in high-quality content.

One of the ways bit rate transcoding implementation is done is by ignoring the high-frequency signals, since high-frequency coefficients have a low impact on quality. This avoids inverse quantization and re-quantization, but needs to be carefully implemented [21]. Another way this is done is by controlling the bit rate in video encoding. This could be implemented by increasing the quantization step size to match the target bit rate. Thus, increasing the quantization step size reduces the number of nonzero quantized coefficients to be encoded, resulting in a higher compression ratio and decreasing the amount of bits in the outgoing stream [21] [23]. A third way to perform bit rate transcoding is by re-encoding the video. This is done by reusing the original motion vectors and mode decisions that are embedded in the bitstream. This eliminates error drift as the reference frames are reconstructed and the residual information is recompressed [21].

2.2.4. Spatial Resolution Transcoding

Spatial resolution can be done by reducing or down-scaling the frame size. To maintain high-quality video content, bit rate transcoding could be combined with spatial resolution transcoding or any other transcoding techniques, especially if the stream needs to be transmitted over resource-limited wireless network devices that have limited display capabilities, such as screen size, resolution depth, and color components [18]. Increasing the number of end-user devices that have different display sizes raises the need to develop multimedia content adapted to each of these devices.

For spatial resolution transcoding, frame size from the incoming stream could be down-scaled. This could be done in either the frequency domain or the spatial domain [21]. Spatial resolution transcoding has the following two advantages [18]: a) the size of the stream is reduced, which can be easily delivered to resource-limited wireless devices, and b) there is no need to scroll up and down in order to watch the video.

2.2.5. Temporal Resolution Transcoding

Temporal resolution transcoding is a very useful technique for low-power devices and low-bandwidth wireless links. Temporal resolution transcoding or frame rate conversion could be done by dropping or skipping some frames from the original video stream. However, removing frames from a video stream requires readjusting the decoder buffer controller to avoid underflow and overflow [21]. Temporal resolution transcoders reduce the size of the stream to accommodate resource-limited devices and bandwidth-limited networks. However, reducing the frame rate might degrade the motion of the video content and thereby the overall quality of the perceived video [19].

2.2.6. Error Resilience Transcoding

Usually wireless channels are more error-prone than wired ones. Thereby, delivering video content in wireless channels can affect the quality of video signals [21]. Therefore, there is a need to match the compressed stream to the channel capabilities, especially when using a mixed-transport environment of wired and wireless channels. To address this, error resilience techniques are usually used to adapt the video signal based on the channel capabilities along the transport link [21]. By using error resilience transcoding techniques, the quality of the delivered video content may be improved, even with the existence of errors [18]. This could be done by injecting the error resilience features into the video stream in the network node that is connected to a wireless network. These error resilience features help in adapting the video stream to the channel, while maintaining the quality of the video within an acceptable range [21]. A detailed description of some of these techniques can be found in [25].

2.3. Related Research

2.3.1. Multimedia Selection and Composition

For the multimedia service selection problem, Qi et al. [26] proposed a multimedia service selection method based on the weighted Principal Component Analysis (PCA) algorithm. They successfully investigated the correlation between QoS properties for multimedia services. In addition, they successfully simplified the selection process by reducing the number of QoS properties and eliminating the correlation between them. However, they did not measure their proposed selection algorithm in terms of QoS assurance. In addition, they found that the time complexity for their proposed algorithm could be measured in polynomial time.

For the transcoder selection problem, Hossain and Saddik [22] proposed a multimedia transcoding service selection process that uses the Ant Colony Optimization (ACO) algorithm for selecting the most suitable multimedia transcoding services for the desired composition process. They statistically evaluated each transcoding service by using the standard-deviation normalization [27], and then assigned a weight to each of them. Finally, they applied the ACO algorithm for the selection and composition processes. Hossain and Saddik [22] used only average transcoding delay and frame rate as QoS properties for the selection process. In addition, their proposed algorithm had more overhead than genetic and Dijkstra-based (traditional) algorithms. Moreover, it had long convergence time.

Xiaohui and Nahrstedt [28] proposed a fully decentralized service composition framework, called the SpiderNet. This framework supports a distributed multimedia service composition process with a statistical QoS assurance. The prototype implementation and simulation results showed the feasibility and efficiency of the SpiderNet. The QoS properties cover both the application and network levels. However, the video transcoding composition process requires special handling due to the sequential dependency of the video transcoding services. In addition, they do not consider frame rate and frame size as QoS properties. Moreover, they did not investigate the quality degradation that is caused by these video transcoding services.

Moissinac [29] proposed a semantic-based automatic discovery and composition approach for multimedia adaptation services. Moissinac [29] focused on developing a semantic description of the basic

adaptation services. Service composition based on semantic description of multimedia services is limited to the opportunity of adding a complete description to all known categories of multimedia services.

Li et al. [30] proposed a heuristic algorithm, named Greedy-EF, to solve the multimedia service composition problem in overlay networks. This composition process finds the proper service paths and routes the data flows through, so that the resource requirements and the QoS constraints of the applications are satisfied. The simulation results showed that their proposed approach can achieve the desired QoS assurance as well as load balancing in multimedia service overlay networks. However, they considered only the response time and the availability of the services as QoS requirements. In addition, their QoS properties covered only the application level.

Hossain [31] proposed a QoS-aware service composition approach for distributed video surveillance in which Hossain [31] used the ant-based algorithm to solve the multi-constraints QoS routing problem. Hossain [31] validated the proposed approach through implementation and simulation. The implementation results showed the quality of the transcoded results in terms of PSNR. The performance results and satisfaction rates of the proposed approach are shown through the simulation.

For the video transcoding service selection and composition problem in a distributed cloud environment, Alsrehin [14] proposed a QoS-aware model that selects and composes the best video transcoding services that satisfy the end-user requirements and preferences. This model uses an aggregation function to evaluate the QoS for each transcoding service and for each viewer request to explore the best composition path. In addition, Alsrehin [14] adapted two algorithms, based on the Simulated Annealing (SA) and Genetic Algorithm (GA), as candidate solutions to help in the composition process. Alsrehin [14] implemented a prototype of the proposed algorithms and conducted experiments using small, medium, and large scale graphs of video transcoders and sample viewer requests to measure the performance and the quality of the results. The experimental results showed that the SA outperforms the GA in terms of performance and success ratio for a small scale graph, while the GA outperforms the SA algorithm in terms of performance for medium and large scale graphs. However, the proposed approaches have some limitations, such as low performance.

Alberto et al. [32] introduced how the Service Oriented Architecture (SOA) paradigm can be applied to the context-aware multimedia communications. In addition, they presented a scoring function for selecting video codec in case of selecting transcoding functions taking into account different quality assessment metrics. Alberto et al. [32] defined a new quality analyzer model to assign a score to each transcoding service. They have some limitations, such as a) the evaluation strategy focused only on the audio codec; b) the composition process is based on the quality or the compression ratio for each audio codec, while the general video transcoding composition process handles the selection of the best implementation from various implementations of the same codec; and c) the evaluation results are based only on a single video/audio source with specific configurations. Evaluating their approach based on a set of video/audio sources with different configurations might help in generating more general results.

In summary, most of the research in the video transcoding selection and composition has focused on developing algorithms and techniques to select and compose a set of transcoding services that are either physically distributed in the network or virtually distributed in the cloud. The selection and composition should satisfy the end-user requirements and preferences. For evaluating these selection and composition algorithm, researchers have focused on evaluating the satisfaction rate or the success ratio with insufficient evaluation of the quality of the perceived video. In addition, these studies statistically evaluate each video transcoding service based on specific values of the desired QoS. I believe that statistically evaluating the weight of each service using the standard-deviation normalization does not reflect the exact degradation that each transcoding service can cause. Also, ignoring the characteristics of the original video and focusing only on the desired QoS specifications might generate inappropriate weight value for the transcoding service and might affect the quality of the perceived video. Moreover, given an existing pool of video transcoding services, adding more services to this pool every time requires re-calculating their weights. This leads us to conclude that the statistical methods are not extensible and are inappropriate for video transcoding service selection and composition algorithms.

2.3.2. Models for Perceptual Video Quality Estimations

Many researchers have developed quality-based models for assessing the impact of video encoding on quality [33]. These models handle only video encoding and consider some parameters that are related to the encoding process, such as bit rate and frame rate. Gonzalez et al. [32] provided a proof of concept for a context-aware multimedia service selection and composition approach using quality assessment. Specifically, they presented a scoring function for selecting different service implementations particularized for selecting video transcoding functions. This scoring function handles the perceptual quality and compression ratio. The quality assessment can be used for measuring the perceptual quality, while the compression ratio is calculated using the number of bits in the original and encoded videos. However, they provided just a proof of concept with insufficient data to evaluate the proposed function. In addition, the experiments focused only on the audio part of the video.

Joskowicz et al. [8] [34] also made an attempt to develop a general parametric model for perceptual quality estimation, in which they presented and analyzed 10 different parametric models for quality estimation proposed by different groups of authors and organizations. A performance comparison was conducted between their proposed general model and other models, using standard Reduced Reference (RR) models with a large set of videos, and subjective tests using a reduced set of videos. For the comparison, Joskowicz et al. [8] used the H.264 codec for the encoding process, Video Quality Metric (VQM) [35] as an objective quality model, CIF and VGA as display sizes with bit rates range from 100kbps to 6Mbps and frame rates range from 5 to 25 fps. Their model showed a better performance for both encoding and transmission degradations, with 0.89 as the PCC based on subjective tests.

Yen-Fu et al. [36] provided subjective studies and analytical models for the perceptual video quality with variations of frame rates and quantization step sizes using five different videos in the CID resolution and 30 fps as an original frame rate. They generated 90 processed (encoded and decoded) videos from the original videos and asked the subjects to give an overall rating for each video in the range of 0 to 100. The impact of constant frame rate was modeled and fits 0.95 with PCC and 0.013 with RMSE. In addition, they provided a model for assessing the impact of constant Quantization Step size (QS) on perceived quality that fits well (with a subset of measured data) with PCC = 0.99 and RMSE = 0.045.

However, they considered only frame rate and quantization step size variations. Additional studies are needed to consider other factors, such as spatial resolution.

Yen-Fu et al. [37] proposed a novel quality metric for compressed video considering both frame rate and quantization artifacts. Their model is a function of PSNR and a Temporal Correlation Factor (TCF). It uses the average PSNR of frames included in the video. Their model has only two parameters and correlates very well with subjective rating obtained in their subjective tests. These parameters are video content dependent. However, their model depends on PSNR and considers only frame rate and quantization artifacts.

Yen-Fu et al. [38] proposed a model for assessing the impact of frame rate on perceptual quality. Their model is based on the sigmoid function and uses the Mean Opinion Score (MOS) at the maximum frame rate for each video. Their model uses a single parameter that is highly correlated with the normalized frame difference, the weighted sum of normalized frame difference, and the motion vector magnitude. However, their model depends on the Mean Opinion Score (MOS) for each video and considers only the frame rate.

Zinner et al. [39] introduced a framework for Quality of Experience (QoE) management for video streaming systems based on the H264/SVC video codec. Zinner et al. [39] conducted a measurement study and quantified the influence of video resolution, scaling method, video frame rate, and video content types on the QoE by means of SSIM and VQM full-reference metrics. After that they discussed the trade-off between these different parameters and their influences on the QoE. They showed that these objective metrics are able to distinguish between quality levels for different resolutions and frame rates. In addition, the results showed that SSIM and VQM can be used to quantify the behavior of different video content on the QoE. Furthermore, they showed that videos with a lower resolution perform better than videos with a lower frame rate with respect to VQM.

2.3.3. Impact of Video Encoding and Transcoding

Vidyut et al. [40] presented the design and implementation of a dynamic video transcoding server. They conducted a systematic evaluation of the following encoding parameters: Q-scale, color depth, and

frame rate on CPU usage, bandwidth usage, and energy consumption of the transcoding server. The measurements showed that reducing image quality to a Q-scale of 10 can dramatically decrease the CPU cycles and streaming throughput. The measurements also show that 50% scaling down on frame width and height effectively reduces the throughput to approximately 50%. In addition, the measurements show that by using a Q-scale greater than 5, the energy consumption might be reduced to more than 50% at the transcoding server.

Goldmann et al. [41] presented the results of a comprehensive study on how the transcoding artifacts influence the subjective study. Furthermore, they analyzed how specific encoding parameters, such as drift error, Group of Pictures (GOP), and number of B-frames influence the strength of these artifacts and thus the quality. They generated 84 different test videos from the DVB [42] and MPEG [41] datasets to perform the subjective quality evaluation. Fifteen non-expert human subjects took part in the experiment. They used the MPEG-2 video codec to encode the original, raw videos. They considered the cascaded pixel-domain and the open-loop transcoding architectures. As a result, the cascaded pixel-domain architecture achieved mostly "good" quality, while the open-loop transcoding architecture achieved "bad" and "fair" quality. Furthermore, they concluded that the largest influence of the subjective quality results is video content dependent. For example, for some videos, the special artifacts seem to have the largest influence, while the temporal artifacts are dominant for other videos.

For GOP experiments conducted in [41], Goldmann et al. [41] observed that the loss of subjective quality depends largely on the visual characteristics of the video and transcoding type, i.e., cascaded pixel-domain or open-loop transcoding. For open-loop transcoding, the video with high spatial details and a small amount of global motion suffers less than the video with lower spatial details and high individual object motion. For pixel-domain transcoding, the subjective quality stays between "good" and "excellent".

For the experiment that evaluates the effect of the number of B-frames (M) on quality in [41], Goldmann et al. [41] observed that the influence of M on the quality of the pixel-domain transcoding is greater than on the open-loop transcoding.

CHAPTER 3

PROBLEM FORMULATION

3.1. Definitions

Definition 3.1 (**Video Format**): A video format is the structure of the video's image and audio data. Some popular formats are H.264, MPEG-4 part 2, MJPEG (Motion JPEG), WMV, FLV, and DivX, but there are over 20 in common use today [3]. In this dissertation, the letter f, sometimes with prefixes, e.g. $x.f$, superscripts, e.g., f', or subscripts, e.g., f_{in}, represents a specific video format.

Definition 3.2 (**Video Codec**): A video codec is a piece of software that can encode video data into a particular format and decode a video from that format. A codec's name is often used as a label or synonym for the format it encodes and decodes.

Definition 3.3 (**Video Characteristics**): A video's characteristics consist of three measures: bit rate, frame rate, and frame size. Here, the letter c with optional prefixes, subscripts, or superscripts represents a set of video characteristics, and $c.br$, $c.fr.$, and $c.fs$ represent the above three measures, respectively. The frame size further consists of a width, $c.fs.w$, and a height, $c.fs.h$. When considering quality, it is useful to consider a frame size's aspect ratio, $c.fs.ar$, which is $c.fs.w \,/\, c.fs.h$. A viewer's desire QoS also specified as a set of video characteristics.

Definition 3.4 (**Video Characteristics Domain**): A video characteristic domain, $f.C$, is the set of all possible characteristic for some format, f. Since bit rate, frame rate, and frame size are metrics with discrete measures and since they all have lower and upper limits for a specific f, $f.C$ is a finite domain.

Definition 3.5 (**Video**): A video is a media file, consisting of an image stream and typically one or more audio streams. As mentioned, earlier, this research ignores the audio streams since they are only a small part of the file's data and since changes to the audio only have a modest impact on bandwidth requirements and on the user's experience. So, for this research, a video, v, is an image stream that consists of a format, $v.f$, and characteristics, $v.c$.

Definition 3.6 (**Video Transcoding Function**): A video transcoding function is a data processing function that converts input video into a new transcoded video with a different format, characteristics, or both. Formally, let $t(v_{in}) = v_{out}$ be a transcoding function that transcodes v_{in} to generate v_{out}, where $t.f_{in} = v_{in}.f, t.c_{in} = v_{in}.c,\ t.f_{out} = v_{out}.f$, and $t.c_{out} = v_{out}.c$. So, $t.f_{in}$ is the input format accepted by t, $t.c_{in}$ are the input characteristics, $t.f_{out}$ is the format of the generated output, and $t.c_{out}$ are its characteristics. Finally, a transcoding function takes time to execute and therefore introduces a delay into the streaming of a video. The anticipated delay of t is represented as $t.d$.

Definition 3.7 (**Video Transcoder**): As mentioned earlier, a video transcoder is a piece of software that implements one or more video transcoding functions. From a theoretical perspective, the grouping of transcoding functions into transcoders is purely an optimization or organizational convenience and does not change the fundamental nature of the selection process. Consider two extremes: a) placing each transcoding function in its own transcoder and b) having all transcoding functions in one large transcoder. Regardless of selection algorithm, the complexity of selection process with these two extremes would be the same. Similarly, partitioning the functions into transcoders according to any criteria would not change the fundamental selection problem. So, for simplicity and without loss of generality, a transcoder, T, is a group of transcoding functions such that all its functions have the same input format, $T.F_{in}$, and the same output format, $T.F_{out}$. Furthermore, *T* must include functions for all possible inputs in the characteristic domain corresponding to the input video format. More formally,

$$T = \{t \text{ is a transcoding function} \mid t.f_{in} = T.F_{in} \wedge t.f_{out} = T.F_{out}\} \text{ and}$$

$$\forall c \in T.F_{in}.C \; \exists t \in T : c = t.c_{in} \quad (3.1)$$

This definition is consistent with how most transcoders are implemented; they can convert from one format to another for any possible set of input video characteristics.

Definition 3.8 (**Compatible Transcoders**): Given an input video format, $v.f$, and some required output format, $v'.f$, a transcoder, *T*, is a compatible transcoder if and only if $v.f = T.F_{in}$ and $v'.f = T.F_{out}$. For convenience, the predict $comp(T, v.f, v'.f)$ represents this condition.

Definition 3.9 (**Video Delivery System**): For this dissertation, a video delivery system, VDS, is a finite set of transcoders, $VDS.TS$, with a set of possible output formats, $VDS.FS$, a set of source videos, $VDS.VS$, and a mechanism for selecting or composing a set of transcoding functions from among the transcoders given a viewer request. See Definition 3.10.

Definition 3.10 (**Viewer Request**): A viewer request, Q, consists of a selected video, $Q.v \in VDS.VS$, a viewing format, $Q.f \in VDS.FS$, and a desired QoS, $Q.c \in Q.f.C$. A viewer request can be thought of as a set of requirements for the delivery of a video. $Q.v$ and $Q.f$ are hard requirements that have to be met for the request to be satisfied. $Q.c$, however, is a soft requirement, meaning that the VDS should try to select a transcoding function or a composition of functions that results in video with characteristics that come as close to $Q.c$ as possible, while minimizing the degradation in quality.

Definition 3.11 (**Video Quality**): In this dissertation, video quality is a measure of how a transcoded video y looks compared to the original video x. Video quality can be evaluated either objectively or subjectively [5]. A video quality for a transcoded video y generated from x and measured using a particular metric m is represented as y. $quality_{m,x}$.

Definition 3.12 (**Objective Video Quality**): In this dissertation, objective video quality is a measure of the quality of a transcoded video that is evaluated using objective quality metrics, such as PSNR, SSIM, and MS-SSIM (see Appendix B), compared to an original video. Because these metrics compare the quality of a transcoded video to an original video, they are referred to as full-references metrics. PSNR measures the quality in decibel loss, typically between 30 and 50 dB. The SSIM, MS-SSIM, and VTOM measure the quality as a ratio between 0 and 1, with 1 being perfect quality. For these metrics, the quality of any video with respect to itself is always 1.

Definition 3.13 (**Video Degradation**): Video degradation is a measure of loss of information caused by video transcoding and is defined as (1 - quality) for the SSIM, MS-SSIM, and VTOM. More specifically, degradation for a transcoded video y generated from a video x based on a particular metric m is $y.degradation_{m,x} = 1 - y.quality_{m,x}$.

Definition 3.14 (**Distance**): A distance between two video characteristics c_1 and c_2, expressed as $d(c_1, c_2)$, is a weighted sum of the absolute difference between their measures, which calculated as follows:

$$d(c_1, c_2) = w_{br} * |c_1.br' - c_2.br'| + w_{fr} * |c_1.fr' - c_2.fr'| + w_{fs} * |c_1.fs' - c_2.fs'| \quad (3.2)$$

where w_{br}, w_{fr}, and w_{fs} are weights (between 0 and 1) for the characteristics that represent the importance of each of them to the viewer request. The br', fr', fs' are the normalized values of the characteristics using a method described in Chapter 7.

3.2. Problem Formulation

This research addresses the problem of modeling the impact of video transcoding on objective video quality to facilitate the selection and composition of transcoding functions. The selection and composition problem can be broken down into two sub-problems: a) a base case requiring a single transcoding function and b) a case requiring the composition of two transcoding functions. More complex cases that require more than two transcoding functions are recursive instances of the later. For the formulization of these cases, let VDS be a video delivery system and Q be a viewer request.

3.2.1. Case 1 – A Single Video Transcoding Function

For selecting a single transcoding function, let's assume that the VDS contains a compatible transcoder for Q, i.e.,

$$\exists\, T \in VDS.TS : comp(T, Q.v.F, Q.f) \quad (3.3)$$

In this case, VDS needs to select a single transcoding function, t, from among all of the compatible transcoders that comes closest to provide the desired quality of service, i.e., $d(t.c_{out}, Q.c) \approx 0$, with the least video degradation. Given (3.3), the problem of selecting t can be expressed as follows:

$$\forall\, t' \in T : t(Q.v) = y \wedge t'(Q.v) = y' \wedge$$
$$((y.degradation_{m,Q.v} \leq y'.degradation_{m,Q.v}) \wedge \big(d(y.c, Q.c) \leq d(y', Q.c)\big) \quad (3.4)$$

To allow the selection to take into account the video degradation, it is necessary to estimate the degradation caused by t without running t. This involves the following three sub-problems: a) finding a mathematical function that models the impact of transforming the codec and three characteristics on

objective video quality, b) finding a mathematical function for the weight value for each transformation in the overall video quality, and c) combining all of these mathematical functions into a single function that measures the overall quality.

3.2.2. Case 2 – Composing Two or More Video Transcoding Functions

Composing two transcoding functions is necessary and possible when VDS does not contain a compatible transcoder for Q, i.e.,

$$\forall\, T\ \in VDS.TS : \neg comp(T, Q.v.f, Q.f) \quad (3.5)$$

but it does contain two transcoders whose composition provide the requested output format, i.e.,

$$\exists T^1, T^2 \in VDS.TS\ \exists f' \in VDS.FS : comp(T^1, Q.v.f, f')\ \wedge comp(T^2, f', Q.f) \quad (3.6)$$

To formalize the problem, let $t_i \in T^1$ and $t_j \in T^2$ be transcoding functions that the VDS will compose to provide the requested output format, $Q.f$, with the goal of closely approximating the desired quality of service, i.e., $d(t_j.c_{out}, Q.c) \approx 0$, with the least video degradation, i.e., the combined video degradation of t_i and t_j is minimized. Composing t_i and t_j can be represented as $t_j\big(t_i(v_{in})\big) = v_{out}$, where the transcoded video from t_i, i.e., $t_i(v_{in})$, will be the input to t_j to generate v_{out}. Given (3.5) and (3.6), the problem of composing two transcoding functions that come closest to provide the desired quality of service with the least video degradation can be expressed as a selection of $t_i \in T^1$ and $t_j \in T^2$ such that

$$\forall t_i' \in T^1\ \ \forall\, t_j' \in T^2 :\ t_j(t_i(Q.v)) = y \wedge\ t_j'(t_i'(v_{in}) = y'\ \wedge$$

$$(y.degradation_{m,Q.v} \leq\ y'.degradation_{m,Q.v} \wedge d(y.c, Q.c) \leq d(y'.c, Q.c) \quad (3.7)$$

The problem of estimating the degradation caused by composing t_i and t_j without running any of them consists of the following two sub-problems: a) finding the best function that measures the degradations caused by any single transcoding functions (Case 1), and b) finding the best function that aggregates the degradations caused by both t_i and t_j . The former requires solving multiple Case-1 selection problems and is on the order of $n * m$, where n is the number of transcoding functions that accept $Q.v.f$ as input and m is the number of transcoding functions that generate $Q.f$ as output. Therefore, once a model that assesses the impact of a single transcoding function on objective video quality exists, composing a set of transcoding functions is relatively a straight forward extension. This dissertation develops such a model,

called VTOM, that assesses the impact of a single transcoding function on objective video quality. VTOM provides the foundation that can be extended, in a future work, to help in composing a set of transcoding functions. The next chapter describes the VTOM in more detail.

CHAPTER 4

VIDEO TRANSCODING OBJECTIVE-QUALITY MODEL (VTOM)

Different parametric models have been proposed to predict the quality of the perceived video based on different encoding parameters. However, most of them apply to specific applications, codecs, networks, or display sizes. Unlike these models, VTOM predicts the quality of the transcoded video based on different transcoding parameters that are based on combinations of different codecs, display sizes, bit rates, and frame rates. In addition, instead of using a traditional quality assessment metric, I developed and evaluated the VTOM using the MS-SSIM [9] metric because it represents one of the best full-reference metrics based on perceived quality [9] [33]. VTOM is an objective RR[7] model, because it uses the format and characteristics of the input video but does not rely on expert judgment or use the content of the input video.

As mentioned earlier, VTOM combines four quality sub-models with four additional error sub-models in a single function as a weighted product aggregation function for assessing the overall video quality. This chapter provides a general description about calculating the quality values that I used as reference values to develop these sub-models. A discussion and evaluation of these sub-models will come later in Chapters 5 and 6, as well as in Appendixes C and D.

4.1. Definitions

Definition 4.1 (**Video Transcoding Objective-quality Model** (**VTOM**)): The quality of video y relative to x measured by VTOM is defined as follows:

$$y.quality_{VTOM,x} = \alpha_{x.f,y.f}(x.c.br)^{w_c} * \beta_{x.f}(x.c.br, y.c.br)^{w_{br}} * \gamma_{x.f}(x.c.br, x.c.fr, y.c.fr)^{w_{fr}} * \delta_{x.f}(x.c.br, x.c.fs, y.c.fs)^{w_{fs}} \quad (4.1)$$

In this aggregated function, $\alpha_{x.f,y.f}$ is a sub-model function that measures the quality generated by changing the codec from $x.f$ to $y.f$; $\beta_{x.f}$ is a sub-model function that measures the quality generated by

[7] Reduced Reference (RR) metrics are metrics that compute the quality based on extracted information about some features from the original and distorted video signals. So, there is no need to fully access both videos [5]. See Appendix B for more detail.

reducing the bit rate from $x.c.br$ to $y.c.br$; $\gamma_{x.f}$ is a sub-model function that measures the quality generated by reducing the frame rate from $x.c.fr$ to $y.c.fr$; and $\delta_{x.f}$ is a sub-model function that measures the quality generated by reducing the frame size from $x.c.fs$ to $y.c.fs$. The exponent values, w_c, w_{br}, w_{fr}, and w_{fs}, represent the weight values of each of these quality sub-model functions.

4.2. Quality Sub-Models

To assess the impact of changing the codec or characteristics on objective video quality, it is necessary to create a set of test videos that can be used for benchmarking. Let $\mathcal{S}$ be an initial set of raw, uncompressed, test videos, $\{v_i, \dots, v_n\}$, that vary in content and motion and let B be a representative set of bit rates, $\{b_1, .., b_m\}$, measured in Mbps, where $|M| = m$. Let $E(v, b)$ be an encoding function that encodes a raw video at b bit rate. Now, a set of test videos, S, can be defined as:

$$S = \bigcup_{v \in \mathcal{S}} \bigcup_{b \in B} E(v, b) \qquad (4.2)$$

The test videos in S can be treated as original videos, even though they are generated via an encoding function, because VTOM only models the change in quality not the quality of the original video. In fact, it is beneficial to have test videos with the same content and motion but at different bit rates to fully analyze the impact of certain kinds of transcodings.

4.2.1. Assessing Video Quality Generated by Changing the Video Codec

The calculation of the α functions begins with transcoding each video in S to two different formats, keeping the characteristics the same using a set of transcoding functions, τ_c, defined as follow:

$$\tau_c = \{t | \exists v \in S : \ t.f_{in} = v.f \ \wedge \ t.f_{out} \neq v.f \wedge \ t.c_{in} = t.c_{out}\},$$
$$such\ that\ \forall x \in S\ \exists t_t t_2 \in \tau_c : t_1.f_{out} \neq t_2.f_{out} \qquad (4.3)$$

The next step is the application of every $t \in \tau_c$ to every $v \in S$ and calculation of the quality of $t(v)$ relative to v to produce $|\mathcal{S}|$ data points for each input/output format pair and bit rate, and $|M|$ data points for each input/output format pair (f_1, f_2). The latter data points form a set of tuples, $Q^c_{f_1,f_2}$, defined as follows:

$$Q^{c}_{f_1,f_2} = \bigcup_{b\in M}\{(b,q) \mid q = \frac{\sum_{x\in S, t\in\tau_C:\, t.f_{in}=f_1 \wedge t.f_{out}=f_2 \wedge t(x).c.br=b}\ t(x).quality_{MSSSIM,x}}{|S|}\} \quad (4.4)$$

For every input/output format pair, I computed an α_{f_1,f_2} function that approximates $Q^{c}_{f_1,f_2}$ using non-linear regression. In the end, I produced a set of α functions; one for every (f_1, f_1) pair, based on the formats represented in VDS.VS.

4.2.2. Assessing Video Quality Generated by Reducing the Bit Rate

The calculation of the β functions begins with selecting some test cases for how to change the bit rate. A minimal set of changes, expressed as a percentage of the input bit rate, is $P_{br} = \{70\%, 60\%, \text{and } 50\%\}$. It is not necessary to try percentages more than 100% because increasing the bit rate should not decrease the quality. The next step is to define a set of transcoders, τ_{br}, that affect these changes on the test set, keeping the format and other characteristics the same as follows:

$$\tau_{br} = \left\{ \begin{array}{c} t \mid \exists v \in S\ \exists p \in P_{br} : \\ (t.f_{in} = v.f) \wedge (t.f_{out} = v.f) \wedge \\ (t.c_{in}.br = v.c.br) \wedge (t.c_{out}.br = v.c.br * p)\ \wedge \\ (t.c_{in}.fr = v.c.fr)\ \wedge\ (t.c_{out}.fr = v.c.fr) \\ \wedge (t.c_{in}.fs = v.c.fs) \wedge (t.c_{out}.fs = v.c.fs) \end{array} \right\} \quad (4.5)$$

The next step is the application of every $t \in \tau_{br}$ to every $v \in S$ and calculation of the quality of $t(v)$ relative to v to produce $|M|$ data points for each input format. These data points form a set of tuples, Q^{br}_{f}, defined as follows:

$$Q^{br}_{f} = \bigcup_{b_{in}\in M} \left\{ (b_{in}, b_{out}, q) \left| \Gamma_{\mathrm{br}} = \left\{ t \in \tau_{br} \left| \begin{array}{c} t.f_{in} = f\ \wedge t.f_{out} = f\ \wedge \\ x.c.br = b_{in} \wedge t(x).c.br = b_{out} \end{array} \right. \right\} \right. \right\}$$
$$\wedge\ q = \frac{\sum_{x\in S, t\in\Gamma_{\mathrm{br}}}\ t(x).quality_{MSSSIM,x}}{|S||\Gamma_{\mathrm{br}}|} \quad (4.6)$$

For every input format, f, I computed a β_f function that approximates Q^{br}_{f} using non-linear regression. In the end, I produced a set of β functions; one for every format represented in VDS.VS.

4.2.3. Assessing Video Quality Generated by Reducing the Frame Rate

The calculation of the γ functions begins with selecting some test cases for how to change the frame rate. A minimal set of changes, expressed as a percentage of the input frame rate, is $P_{fr} =$

$\{83.3\%, 66.6\%, \text{and } 50\%\}$. The next step is to define a set of transcoders, τ_{fr}, that will affect these changes on the test set, keeping the format and other characteristics the same as follows:

$$\tau_{fr} = \left\{ \begin{array}{c} t \mid \exists v \in S \; \exists p \in P_{fr} : \\ (t.f_{in} = v.f) \wedge (t.f_{out} = v.f) \wedge \\ (t.c_{in}.br = v.c.br) \wedge (t.c_{out}.br = v.c.br) \; \wedge \\ (t.c_{in}.fr = v.c.fr) \wedge (t.c_{out}.fs = v.c.fr * p) \wedge \\ (t.c_{in}.fs = v.c.fs) \\ \wedge (t.c_{out}.fs = v.c.fs) \end{array} \right\} \quad (4.7)$$

The next step is the application of every $t \in \tau_{fr}$ to every $v \in S$ and calculation of the quality of $t(v)$ relative to v to produce $|M|$ data points for each input format. These data points form a set of tuples, Q_f^{fr}, defined as follows:

$$Q_f^{fr} = \bigcup_{b \in M} \left\{ (b, fr_{in}, fr_{out}, q) \middle| \Gamma_{\text{fr}} = \left\{ t \in \tau_{fr} \middle| \begin{array}{c} t.f_{in} = f \; \wedge t.f_{out} = f \; \wedge \\ x.c.br = t(x).c.br \\ \wedge \; x.c.br = b \wedge \; x.c.fr = fr_{in} \\ \wedge \; t(x).c.fr = fr_{out} \end{array} \right\} \right\}$$
$$\wedge \; q = \frac{\sum_{x \in S, t \in \Gamma_{\text{fr}}} t(x).quality_{MSSSIM,x}}{|S||\Gamma_{\text{fr}}|} \quad (4.8)$$

For every input format, f, I computed a γ_f function that approximates Q_f^{fr} using non-linear regression. In the end, I produced a set of γ functions; one for every format represented in VDS.VS.

4.2.4. Assessing Video Quality Generated by Reducing the Frame Size

The calculation of the δ functions begins with selecting some test cases for how to change the frame size. A minimal set of changes, expressed as a percentage of the input frame size, is $P_{fs} = \{62.5\%, 44.4\%, 14.8\% \}$, see (6.11) for how to calculate these percentages. The next step is to define a set of transcoders, τ_{fs}, that will affect these changes on the test set, keeping the format and other characteristics the same as follows:

$$\tau_{fs} = \left\{ \begin{array}{c} t \mid \exists v \in S \; \exists p \in P_{fs} : \\ (t.f_{in} = v.f) \wedge (t.f_{out} = v.f) \wedge \\ (t.c_{in}.br = v.c.br) \wedge (t.c_{out}.br = v.c.br) \; \wedge \\ (t.c_{in}.fr = v.c.fr) \wedge (t.c_{out}.fr = v.c.fr) \wedge \\ (t.c_{in}.fs = v.c.fs) \; \wedge \; (t.c_{out}.fs = v.c.fs * p) \end{array} \right\} \quad (4.9)$$

The next step is the application of every $t \in \tau_{fs}$ to every $v \in S$ and calculation of the quality of $t(v)$ relative to v, producing $|M|$ data points for each input format. These data points form a set of tuples, Q_f^{fs}, defined as follows:

$$Q_f^{fs} = \bigcup_{b \in M} \left\{ (b, fs_{in}, fs_{out}, q) \middle| \Gamma_{fs} = \left\{ t \in \tau_{fs} \middle| \begin{array}{c} t.f_{in} = f \ \wedge t.f_{out} = f \ \wedge \\ x.c.br = t(x).c.br \\ \wedge \ x.c.br = b \wedge \ x.c.fs = fs_{in} \\ \wedge \ t(x).c.fs = fs_{out} \end{array} \right\} \right.$$
$$\left. \wedge \ q = \frac{\sum_{x \in S, t \in \Gamma_{fs}} t(x).quality_{MSSSIM,x}}{|S||\Gamma_{fs}|} \right\} \quad (4.10)$$

For every input format, f, I computed a δ_f function that approximates Q_f^{fs} using non-linear regression. In the end, I produced a set of δ functions; one for every format represented in VDS.VS.

4.3. Assessing the Weight

Generally, I defined the weight, w_i, for each a quality sub-model i as follows:

$$w_i = 1 - Error_i \quad (4.11)$$

where $Error_i$ is the error in expecting the quality value based on the quality sub-model i. To assess these error values; I developed four error sub-models, one for each quality sub-model.

4.3.1. *Assessing the Error Generated by Codec Conversion*

The calculation of the $Error_{f_1,f_2}^{\alpha}$ functions begins with using the quality of $t(v)$ relative to v for every $v \in S$ and $t \in \tau_c$ calculated above to produce $|S|$ data points for each input/output format pair and bit rate, and $|M|$ data points for each input/output format pair. These data points form a set of tuples, E_{f_1,f_2}^{α}, defined as follows:

$$E_{f_1,f_2}^{\alpha} = \bigcup_{b \in M} \left\{ (b, e) \middle| e = \frac{\sum_{x \in S, t \in \tau_C : t.f_{in} = f_1 \ \wedge \ t.f_{out} = f_2 \wedge x.c.br = b} |t(x).quality_{MSSSIM,x} - t(x).quality_{\alpha_{f_1,f_2}(b),x}|}{|S|} \right\} \quad (4.12)$$

For every input/output format pair, I computed the $Error_{f_1,f_2}^{\alpha}$ function that approximates E_{f_1,f_2}^{α} using non-linear regression. In the end, I produced a set of $Error_{f_1,f_2}^{\alpha}$ functions; one for every possible input/output format pair based on the formats represented in VDS.VS.

4.3.2. *Assessing the Error Generated by Reducing Bit Rate*

The calculation of the $Error_f^{\beta}$ functions begins with using the quality of $t(v)$ relative to v for every $v \in S$ and $t \in \Gamma_{\text{br}}$ calculated above to produce $|M|$ data points for each input format. These data points form a set of tuples, E_f^{β}, defined as follows:

$$E_f^{\beta} = \bigcup_{b_{in} \in M} \{(b_{in}, b_{out}, e) \mid e = \frac{\sum_{x \in S, t \in \Gamma_{\text{br}}} |t(x).quality_{MSSSIM,x} - t(x).quality_{\beta_f(br_{in}, br_{out}),x}|}{|S||\Gamma_{\text{br}}|}\} \quad (4.13)$$

For every input format, I computed the $Error_f^{\beta}$ function that approximates E_f^{β} using non-linear regression. In the end, I produced a set of $Error_f^{\beta}$ functions; one for every possible input format based on the formats represented in VDS.VS.

4.3.3. *Assessing the Error Generated by Reducing Frame Rate*

The calculation of the $Error_f^{\gamma}$ functions begins with using the quality of $t(v)$ relative to v for every $v \in S$ and $t \in \Gamma_{\text{fr}}$ calculated above to produce $|M|$ data points for each input format. These data points form a set of tuples, E_f^{γ}, defined as follows:

$$E_f^{\gamma} = \bigcup_{b \in M} \{(b, fr_{in}, fr_{out}, e) \mid e = \frac{\sum_{x \in S, t \in \Gamma_{\text{fr}}} |t(x).quality_{MSSSIM,x} - t(x).quality_{\gamma_f(b, fr_{in}, fr_{out}),x}}{|S||\Gamma_{\text{fr}}|}\} \quad (4.14)$$

For every input format, I computed the $Error_f^{\gamma}$ function that approximates E_f^{γ} using non-linear regression. In the end, I produced a set of $Error_f^{\gamma}$ functions; one for every possible input format based on the formats represented in VDS.VS.

4.3.4. *Assessing the Error Generated by Reducing Frame Size*

The calculation of the $Error_f^{\delta}$ functions begins with using the quality of $t(v)$ relative to v for every $v \in S$ and $t \in \Gamma_{\text{fs}}$ calculated above to produce $|M|$ data points for each input format. These data points form a set of tuples, E_f^{δ}, as follows:

$$E_f^{\delta} = \bigcup_{b \in M} \{(b, fs_{in}, fs_{out}, e) \mid e = \frac{\sum_{x \in S, t \in \Gamma_{\text{fs}}} |t(x).quality_{MSSSIM,x} - y.quality_{\delta_f(b, fs_{in}, fs_{out}),x}|}{|S||\Gamma_{\text{fs}}|}\} \quad (4.15)$$

For every input format, I computed the $Error_f^{\delta}$ function that approximates E_f^{δ} using non-linear regression. In the end, I produced a set of $Error_f^{\delta}$ functions; one for every possible input format based on the formats represented in VDS.VS.

CHAPTER 5

EXPLORING THE IMPACT OF VIDEO TRANSCODING ON OBJECTIVE VIDEO QUALITY

Modeling the impact of video transcoding requires exploring its impact on objective video quality. To explore its impact, this chapter first describes the video data set used in the exploring, modeling, and testing phases. Second, it explores the impact of each of the following on objective video quality: a) video encoding, b) changing the video codec, c) reducing the frame rate, d) reducing the frame size, and e) reducing the bit rate. Third, it provides a general discussion of these results.

5.1. Video Data Set

I used the VQEG Phase 1 2010 video data set, which contains sets of raw, uncompressed HDTV videos, each of which has the following characteristics [10]:

- 10-second length with no audio content
- 1080p progressive scan with 1920x1080 frame size
- Around 995 Mbps bit rate
- Around 1.15 Giga byte raw-video file size
- UYVY422 pixel type
- 29.97 fps frame rate

I selected a set that contains 10 videos based on the following three criteria: a) variety of video content, i.e., natural scenes, sports, movies, and animations, b) a range of scene source material, i.e., complex content and high motion to simple content and low motion, and c) variety of color and brightness components. Figure 5.1 shows the first frame from each of the selected videos.

5.2. Exploring the Impact of Video Encoding

I performed the following steps to explore the impact of video encoding on objective video quality:

Figure 5.1. The first frame from each video used in the exploring, modeling, and testing phases.

1) Selecting a set of video codecs to be used in video encoding. This set includes H.264, MPEG-4, and FLV codecs. The selection criteria are based on the codecs that are commonly used by researchers and have high efficiency and flexibility in compression.
2) Selecting the original videos. In this step, I used nine original, raw, uncompressed videos from the data set described above. Specifically, I used the following videos as input: vqeghd1_csrc11, vqeghd1_csrc12, vqeghd1_csrc14, vqeghd1_src01, vqeghd1_src02, vqeghd1_src03, vqeghd1_src04, vqeghd1_src05, and vqeghd1_src06.
3) Selecting the values of the bit rates. These values are 2, 4, 6, 8, 10, 15, and 20 Mbps.

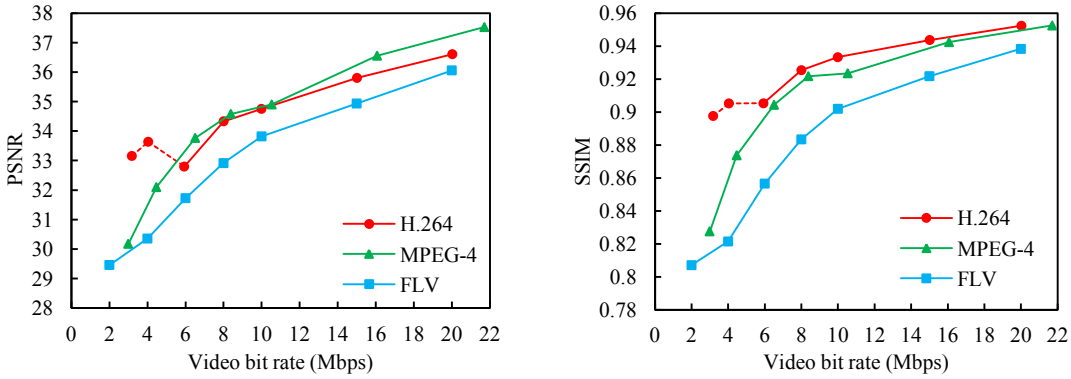
4) Implementing the video encoding functionality. In this step, I implemented the encoder for each codec mentioned above using Java® SDK 1.8 and Xuggler© 4.5.
5) Generating a new set of encoded videos. In this step, I encoded the original, raw, uncompressed videos using the encoding functionality and the codecs mentioned above at different values of bit rates that are specified above. The total number of videos generated as output is 189, i.e., 9 original videos ∗ 7 different values of bit rates ∗ 3 different codecs.
6) Assessing the impact of video encoding on objective video quality. I used the PSNR [11], SSIM [12], and MS-SSIM [9] quality metrics for this assessment step. More detail about these metrics can be found in Appendix B. In this step, I compared each pair of videos, the original and encoded, using these metrics. Each of the above metrics is frame-based, so I compared each pair of frames from both the original and encoded videos and then took the average of all comparisons as a final result between any two videos.
7) Aggregating the above assessment results. In this step, I aggregated the above quality assessment results based on the bit rate for each codec.

Video encoding represents compressing the original, raw, uncompressed videos to generate new compressed videos based on the values of the encoding parameters described above. For example, given the original, raw, uncompressed video, vqeghd1_csrc11, I encoded this video using the H.264, MPEG-4, and FLV codecs at different bit rates ranging from 2 to 20 Mbps. During this encoding, I kept the frame rate and size the same as in the original videos. Any original video was used to generate a new set that includes at least 21 encoded videos, i.e., 3 different codecs ∗ 7 different bit rates. Figure C.1 in Appendix C shows the naming scheme that I used to name the encoded videos. Table C.1 shows some examples of these names.

Figure 5.2 shows the video encoding quality evaluation results in terms of PSNR, SSIM, and MS-SSIM for the three codecs mentioned above. Each point represents the average of the quality values of all the nine videos used at a given bit rate. Notice that there are solid and dashed curves for the H.264 codec, solid curves represent the average of nine videos while the dashed curves represent the average of only six videos. This is because the H.264 codec is unable to encode some of these videos at lower bit rates, i.e.,

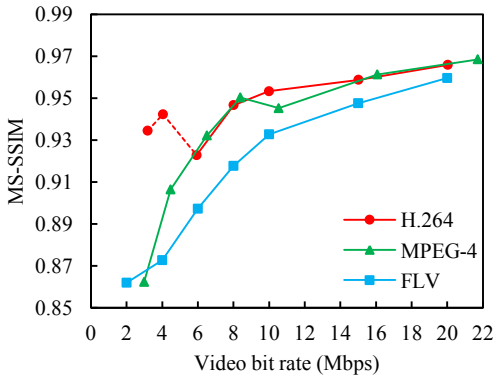
lower than 6 Mbps. Figure 5.2 also shows that whenever the value of the bit rate is increased, the quality is increased. In addition, it shows that the FLV codec achieves the lowest quality results. Moreover, it shows that the H.264 and MPEG-4 codecs achieve close quality results. The H.264 codec achieves the highest quality in terms of SSIM.

As a conclusion, the H.264 codec encodes the original, raw, uncompressed videos in a better quality than the MPEG-4 and FLV codecs at a given bit rate.



(a) PSNR

(b) SSIM



(c) MS-SSIM

Figure 5.2. The video quality evaluation results for exploring the impact of video encoding on objective video quality for the H.264, MPEG-4, and FLV codecs. The evaluation results are in terms of (a) PSNR, (b) SSIM, and (c) MS-SSIM.

5.3. Exploring the Impact of Changing the Video Codec

For exploring the impact of changing the codec on objective quality in video transcoding, I performed steps similar to those described above, but in this case to perform the following six different mappings: a) from H.264 to MPEG-4, b) from H.264 to FLV, c) from MPEG-4 to H.264, d) from MPEG-4 to FLV, e) from FLV to H.264, and f) from FLV to MPEG-4. The total number of videos used as input for each mapping is 35, i.e., 5 videos ∗ 7 different bit rates.

During this transcoding, I kept the bit rate, frame rate, and frame size the same as in the original videos. The only thing that I changed in this transcoding was the codec. The total number of videos generated as output for each mapping, based on the input videos, is 35.

Figure 5.3 shows the quality evaluation results for exploring the impact of changing the codec on objective video quality in terms of PSNR, SSIM, and MS-SSIM. Generally, it shows that when the bit rate is increased, the quality is increased. In addition, it shows that the conversion from FLV to MPEG-4 achieves the highest quality in terms of PSNR and SSIM. The transcoding from H.264 to MPEG-4 and the transcoding from MPEG-4 to H.264 achieve the next highest quality. Moreover, it shows that all the other conversions, i.e., FLV to H.264, MPEG-4 to FLV, and H.264 to FLV, have close quality results at a given bit rate. Table C.2 in Appendix C shows more detailed results.

5.4. Exploring the Impact of Reducing the Frame Rate

For exploring the impact of reducing the frame rate on objective video quality in video transcoding, I performed steps similar to those described above, but in this case to explore the impact of reducing the frame rate. I selected a set of new frame rates that includes 15 fps, 20 fps, and 25 fps to be used during video transcoding. The total number of videos used as input for each codec is 35, i.e., 5 videos ∗ 7 different bit rates. Figure C.2 in Appendix C shows the grouping structure of these videos. During this transcoding, I kept the codec, bit rate, and frame size the same as in the original videos. The only thing I changed in this transcoding was the frame rate. The total number of videos generated as output for each codec, based on the input videos, is 105, i.e., 35 input videos ∗ 3 different frame rates.

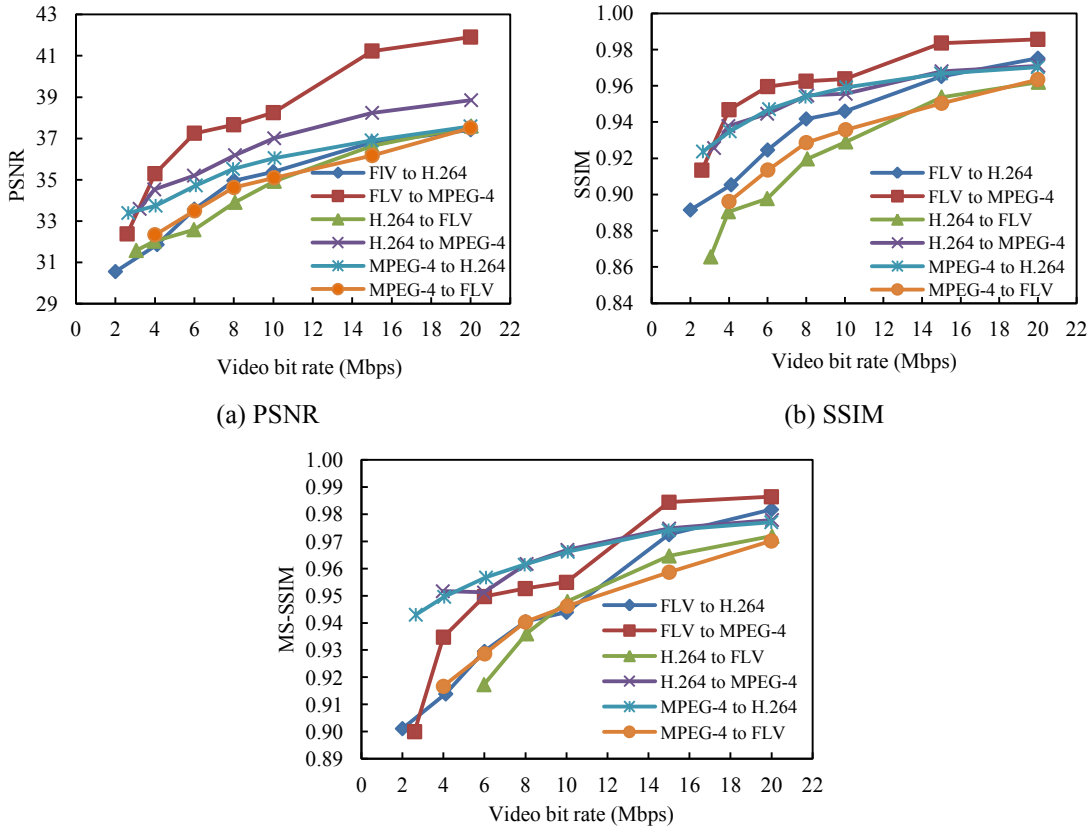


Figure 5.3. The video quality evaluation results for exploring the impact of changing the codec on objective video quality in terms of (a) PSNR, (b) SSIM, and (c) MS-SSIM.

As part of this research, and because no objective quality metric exists that compares two videos that have different frame rates, it was necessary to develop such a metric, called Frame Rate Metric (FRM). FRM uses any frame-based metric to compare frames from both videos. In this research, I used the above metrics, i.e., PSNR, SSIM, and MS-SSIM, to compare each pair of video frames from the original and transcoded videos.

FRM finds the Least Common Multiple (LCM) between the frame rates of the original and transcoded videos. After that, it finds the value at which the two frames from the original and the transcoded videos intersect, and then it multiplies this intersection value with the result of the objective quality metric that is generated by comparing these frames together. Finally, it calculates the weighted

average for all these comparisons. Algorithm 1 describes the FRM metric in more detail. Figure 5.4 shows an example that depicts how FRM calculates the quality based on two videos that have different frame rates.

The example illustrated in Figure 5.4(a) shows two videos that are at different frame rates, v_1 is at 10 fps and v_2 is at 7 fps. v_2 is generated by transcoding v_1 and reducing the frame rate from 10 fps to 7 fps in the transcoding. Each video is represented as a sequence of boxes; each box has a different color and represents a frame. Also, I numbered the boxes from 1 to 10 for v_1 and from 1 to 7 for v_2.

Figure 5.4(b) shows that the LCM between 10 and 7 is 70. Then I divided 70 over 10 and 7 to get 7 and 10 as results, respectively. Figure 5.4(c) shows how I used these numbers to calculate the weight. This weight represents the intersection between each frame in v_1 with its corresponding frame in v_2. Figure 5.4(c) also shows how I used these numbers to calculate the quality between these two videos. Frame 1 from v_1 is compared with frame 1 from v_2 using an objective quality metric and the result of this comparison is multiplied by 7. Then frame 2 from v_1 is compared with frame 1 from v_2 using an objective quality metric and the result of this comparison is multiplied by 3, and so on. The values of these comparisons are added together and then divided by the total summation of all the intersection values.

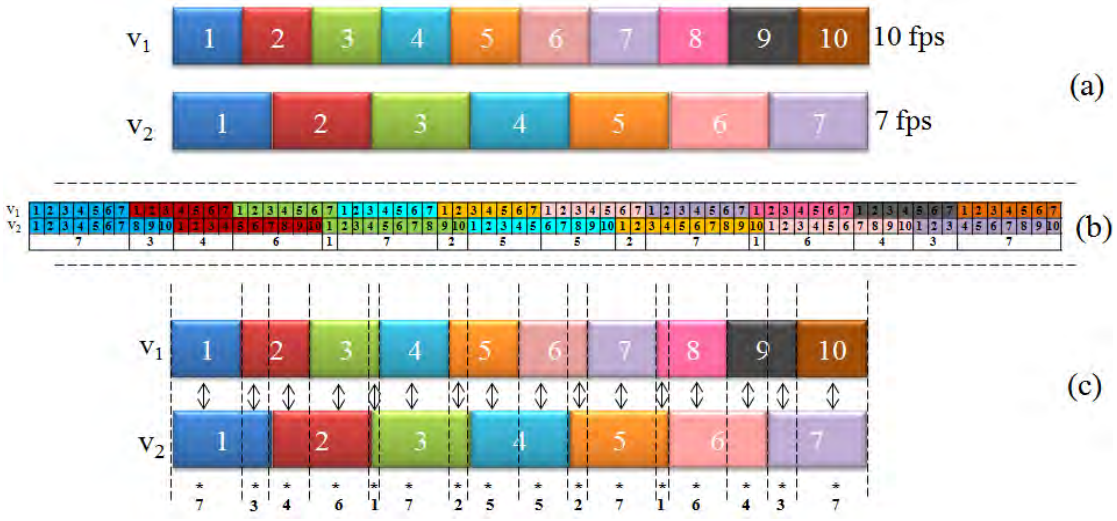


Figure 5.4. An example that shows how FRM compares two videos that have different frame rates.

Algorithm 5.1 FrameRateMetric(video v_1, video v_2) // the proposed new metric

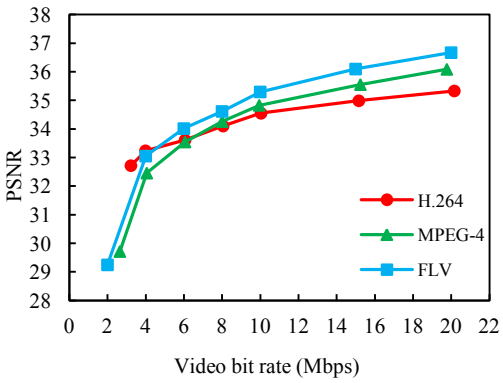
```
1:  size1 ← num_of_frames(v1);
2:  size2 ← num_of_frames(v2);
3:  for (i = 1 to size1) do
4:      for (j = 1 to size2) do
5:          frame_i ← v1.getFrame(i);
6:          frame_j ← v2.getFrame(j);
7:          if (intersect(frame_i, frame_j))
8:              weight ← calcualteWeigth(frame_i, frame_j);
9:              quality ← calculateObjectiveQuality(frame_i, frame_j);
10:             results.add(weight*quality);
11:             weights.add(weight);
12:         end if
13:     end for
14: end for
15: for (i = 0 to results.size − 1) do
16:     sumQ += results.get(i);
17: end for
18: for (i = 0 to weights.size − 1) do
19:     sumW += weights.get(i);
20: end for
21: return sumQ/sumW;
```

In Algorithm 5.1, the $intersect(frame_i, frame_j)$ method checks if the two frames intersect or not. If so, the $calcualteWeigth(frame_i, frame_j)$ method calculates the intersection value, i.e., weight, between these two frames. Then the $calculateObjectiveQuality(frame_i, frame_j)$ method compares these two frames using any frame-based objective quality metric and returns the quality value. After that, it stores the result of multiplying the intersection value with the quality value in any data structure for later use. This process is repeated for all the frames that are available in both videos. Finally, the results of multiplying the intersection values with the quality values are added together and then divided by the summation of all the intersection values. The returned result from Algorithm 1 represents the overall quality between any two videos that have different frame rates.

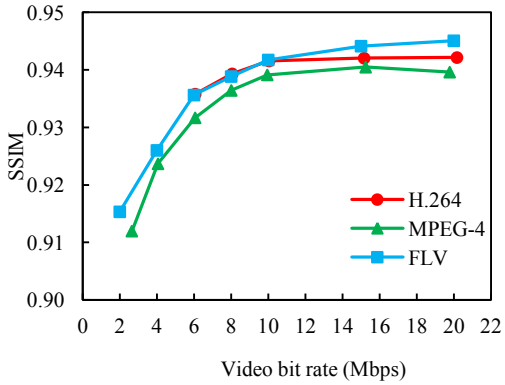
Figure 5.5 shows the quality evaluation results for exploring the impact of reducing the frame rate from 29.97 fps to 25 fps in terms of PSNR, SSIM, and MS-SSIM for the H.264, MPEG-4, and FLV codecs. Most of the codecs achieve high quality when the bit rate is increased. In addition, after a specific bit rate

value, the quality does not improve a lot in terms of SSIM and MS-SSIM. For example, Figure 5.5 (b) shows that after 10 Mbps bit rate value, there is no big improvement in the quality for most of the codecs.

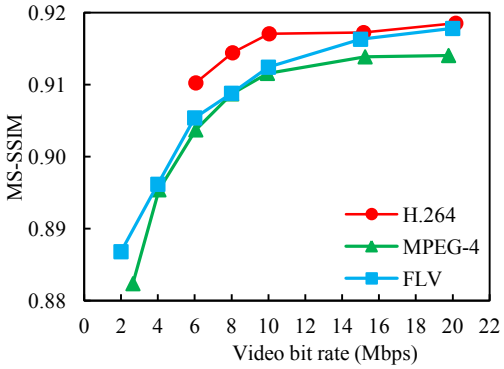
Figure 5.5(a) and Figure 5.5(b) show that the FLV codec achieves the highest quality for most cases in terms of PSNR and SSIM, respectively. Figure 5.5(c) shows that the H.264 codec achieves the highest quality in terms of MS-SSIM. In addition, the FLV and MPEG-4 codecs have close results in terms of MS-SSIM. As a conclusion, all the above codecs have close quality results and the difference in quality is not too big.



(a) PSNR (from 29.97 fps to 25 fps)



(b) SSIM (from 29.97 fps to 25 fps)

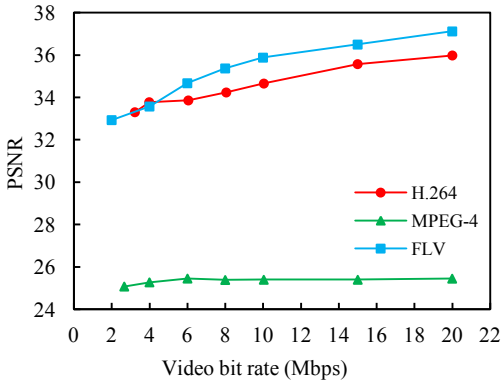


(c) MS-SSIM (from 29.97 fps to 25 fps)

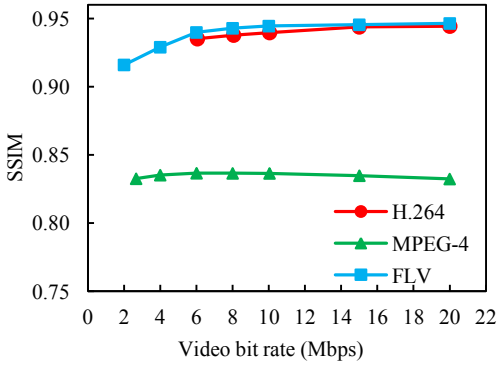
Figure 5.5. The quality evaluation results for exploring the impact of reducing the frame rate on objective quality for the H.264, MPEG-4, and FLV codecs. The frame rate reduction is from 29.97 fps to 25 fps. The quality evaluation results are in terms of (a) PSNR, (b) SSIM, and (c) MS-SSIM.

Figure 5.6 shows the quality evaluation results for exploring the impact of reducing the frame rate from 29.97 fps to 20 fps in terms of PSNR, SSIM, and MS-SSIM for the H.264, MPEG-4, and FLV codecs. The H.264 and FLV codecs achieve better quality results than MPEG-4. In addition, there is no big difference in the quality between the H.264 and FLV codecs. Figure 5.6(a) shows that the H.264 and FLV codecs achieve high quality in terms of PSNR when the bit rate is increased. For the MPEG-4 codec, the improvement in the quality is very small when the bit rate is increased.

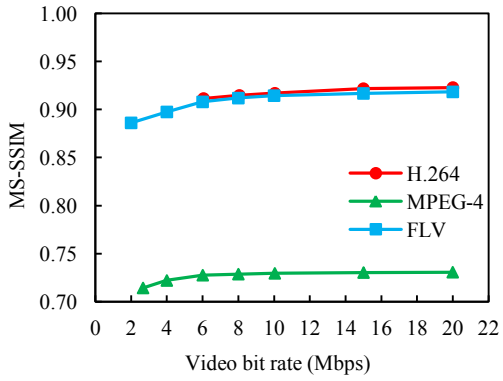
Figure 5.6(b) shows that the H.264 and FLV codecs achieve small improvements in the quality in terms of SSIM when the bit rate is increased. For the MPEG-4 codec, there is a small improvement in the quality when the bit rate is increased. After 10 Mbps bit rate, the quality is decreased by a small amount.



(a) PSNR (from 29.97 fps to 20 fps)



(b) SSIM (from 29.97 fps to 20 fps)



(c) MS-SSIM (from 29.97 fps to 20 fps)

Figure 5.6. The quality evaluation results for exploring the impact of reducing the frame rate on objective quality for the H.264, MPEG-4, and FLV codecs. The frame rate reduction is from 29.97 fps to 20 fps. The quality results are in terms of (a) PSNR, (b) SSIM, and (c) MS-SSIM.

Figure 5.6(c) shows that the H.264 and FLV codecs achieve small and almost the same improvement in the quality in terms of MS-SSIM when the bit rate is increased. For the MPEG-4 codec, there is an improvement in the quality when the bit rate is increased from 2 to 8 Mbps. When the bit rate is higher than 8 Mbps, there is almost no improvement in the quality. One factor that affects the quality is the number of dropped frames. In the FRM, if this number is increased, this means that the frame-by-frame comparison strategy compares two frames, from the original and transcoded videos, with different frame content. Increasing the bit rate of any of these frames does not enhance this comparison and thereby the overall quality.

Figure 5.7 shows samples of different frame dropping approaches for the example shown in Figure 5.4. Figure 5.7(a) shows that the last three frames, i.e., frames 8, 9, and 10, from v_1 were dropped in video transcoding to generate v_2. In this situation, to evaluate the quality of v_2 generated from transcoding v_1, the comparison is done by comparing frame 1 from v_1 with frame 1 from v_2 and multiplying the result by 7, frame 2 from v_1 with frame 1 from v_2 and multiplying the result by 3, frame 2 from v_1 with frame 2 from v_2 and multiplying the result by 4, and then frame 3 from v_1 with frame 2 from v_2 and multiplying the result by 6, and so on. Finally, it divides the summation of all these comparisons by the summation of all the weights as a final result.

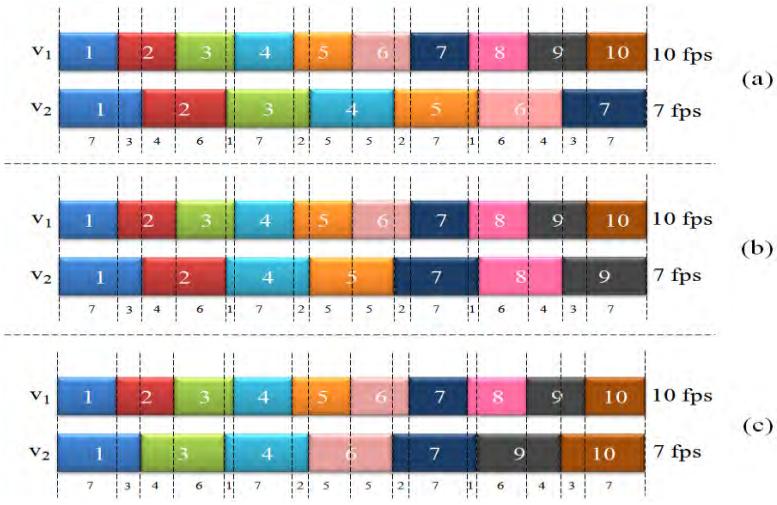


Figure 5.7. Samples of different frame dropping approaches.

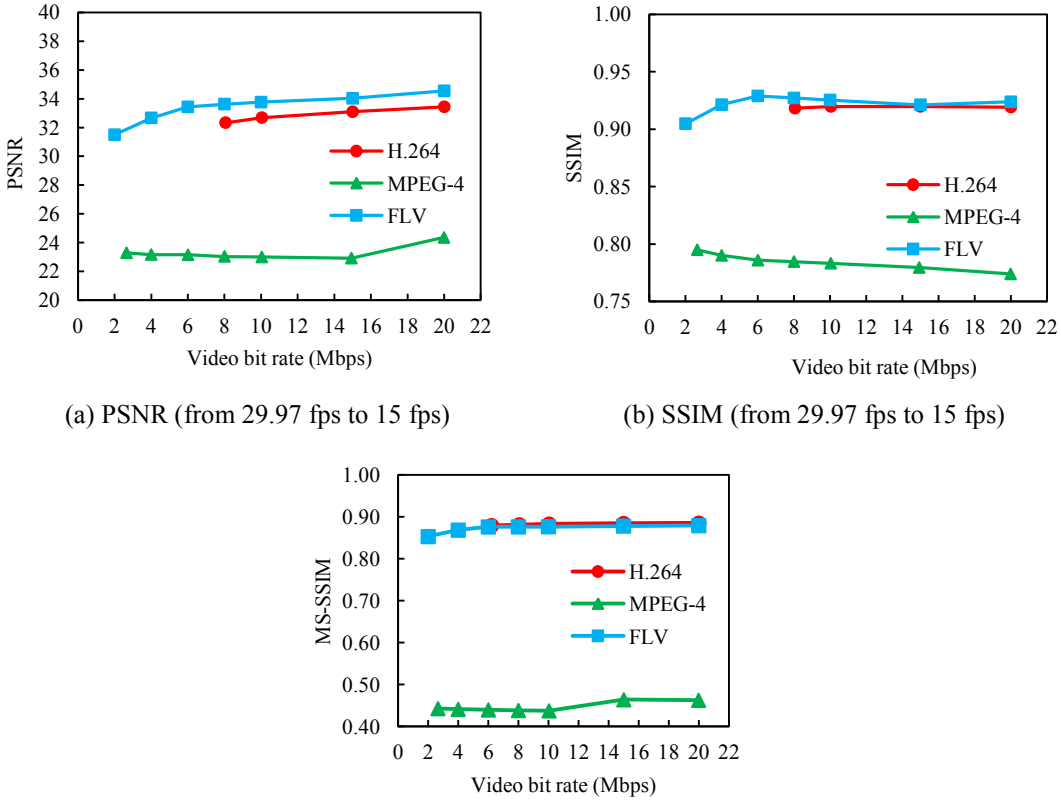
In this type of comparison, FRM compares frames with different frame content, such as frame 10 from v_1 with frame 7 from v_2, because they appeared on the same timestamp. This means that however the bit rate is increased in the transcoding, the quality value that is returned from PSNR, SSIM, and MS-SSIM does not be improved a lot. In other words, the comparison between two frames with different frame content does not increase the value that is returned from the objective metrics that I used. I believe that this type of comparison is fair. However, it needs more investigation and to be evaluated using subjective tests.

Figure 5.7(b) shows that frames 3, 6, and 10 were dropped in video transcoding, while Figure 5.7(c) shows that frames 2, 5, and 8 were dropped in video transcoding. Notice that in each of these approaches, the percentage of frame rate reduction is 30%, while in this exploring phase; the percentage of frame reduction is different (16% for frame rate reduction from 29.97 to 25, 33 % for frame rate reduction from 29.97 to 20 and 50% for frame rate reduction from 29.97 to 15). As a conclusion, when the number of dropped frames is increased, the quality is decreased.

Another factor is the distance between frames in the transcoded video. If the distance becomes greater, this means that the frame content is usually completely different, especially in high-level motion videos.

Figure 5.8 shows the quality evaluation results for exploring the impact of reducing the frame rate from 29.97 fps to 15 fps on objective video quality in terms of PSNR, SSIM, and MS-SSIM for the H.264, MPEG-4, and FLV codecs. The H.264 and FLV codecs achieve better quality results than MPEG-4. In addition, there is no big difference in the quality results between the H.264 and FLV codecs, especially in terms of SSIM and MS-SSIM. Also, it shows that there is a gap between the quality results that are generated using the FLV and H.264 codecs and the quality results that are generated using the MPEG-4 codec.

Figure 5.8(a) shows that the FLV codec achieves the highest quality in terms of PSNR. Then the H.264 and MPEG-4 codecs achieve the second and third highest quality, respectively. In addition, it shows that the quality for the FLV and H.264 codecs is increased when the bit rate is increased. Moreover, it shows that for the MPEG-4 codec, the quality is decreased by a very small amount when the bit rate is increased up to 15 Mbps. After this bit rate value, the quality is increased when the bit rate is increased.



(a) PSNR (from 29.97 fps to 15 fps)

(b) SSIM (from 29.97 fps to 15 fps)

(c) MS-SSIM (from 29.97 fps to 15 fps)

Figure 5.8. The video quality evaluation results for exploring the impact of reducing the frame rate on objective video quality for the H.264, MPEG-4, and FLV codecs. The frame rate reduction is from 29.97 fps to 15 fps. The quality results are in terms of (a) PSNR, (b) SSIM, and (c) MS-SSIM.

Figure 5.8(b) shows that the FLV codec achieves a small improvement in the quality in terms of SSIM when the bit rate is increased from 2 to 6 Mbps. After 8 Mbps bit rate, the quality is decreased by a small amount when the bit rate is increased. At 15 Mbps bit rate, the quality is increased by a small amount when the bit rate is increased. In addition, it shows that for the MPEG-4 codec, the quality is decreased by a small amount in terms of SSIM when the bit rate is increased.

Figure 5.8(c) shows that the H.264 and FLV codecs achieve the highest quality in terms of MS-SSIM. In addition, it shows that there is a small improvement in the quality for the H.264 and FLV codecs when the bit rate is increased. For the MPEG-4 codec, the quality starts with a very small improvement

when the bit rate is increased. When the value of the bit rate is increased from 10 to 15 Mbps, the quality is noticeably improved.

Figures C.5, C.6, and C.7 in Appendix C show the quality evaluation results for exploring the impact of reducing the frame rate on objective video quality aggregated based on the H.264, MPEG-4, and FLV codecs, respectively.

5.5. Exploring the Impact of Reducing the Frame Size

For exploring the impact of reducing the frame size on objective video quality in video transcoding, I performed steps similar to those described above, but in this case to explore the impact of reducing the frame size. I selected a set of new frame sizes that includes 1440x900, 1280x720, and 640x480. These sizes represent 62.5%, 44.4%, and 14.8% of the original frame size, i.e., 1920x1080, respectively. The total number of videos used for each codec as input is 35, i.e., 5 different videos $*$ 7 different bit rates. Figure C.3 in Appendix C shows how I grouped these videos. During this transcoding, I kept the codec, bit rate, and frame rate the same as in the original videos. The only thing that I changed in this transcoding was the frame size. The total number of videos generated as output, based on the input videos, is 105, i.e., 35 input videos $*$ 3 different frame sizes.

Because no objective quality metric exists that compares two frames with different frame sizes, I proposed a comparison strategy by down-scaling the size of each frame from the original video to match the size of each frame from the transcoded video. I used the bi-cubic interpolation algorithm [43] to down-scale the frame size. I decided to perform down-scaling for the following four reasons: a) I believe that down-scaling the frames, i.e., images, is an easier process than up-scaling; it throws away data in a more or less intelligent manner; b) down-scaling frames generates more accurate and precise frames with fewer on-screen errors and artifacts than up-scaling; c) it is more realistic to the assessment process to compare frames from the transcoded video with the down-scaled frames from the original video than up-scaling the frames from the transcoded video and comparing them with frames from the original video; and d) for video transmission via the internet, it is common to down-scale the frame size to the optimal output resolution to save bandwidth and storage space.

Figure 5.9 describes the general structure of this down-scaling strategy. First, I extracted a frame from the original video and its corresponding, i.e., transcoded, frame from the transcoded video. Second, I down-scaled the size of the frame extracted from the original video to match the size of the frame extracted from the transcoded video. This down-scaling procedure is done using the bi-cubic interpolation algorithm [43]. Third, I compared these two frames using any frame-based objective quality metric, i.e., PSNR, SSIM, or MS-SSIM.

Figures 5.10, 5.11, and 5.12 show the evaluation results of reducing the frame size from 1920x1080 to 1440x900, from 1920x1080 to 1280x720, and from 1920x1080 to 640x480, respectively, in terms of PSNR, SSIM, and MS-SSIM for the H.264, MPEG-4, and FLV codecs. Generally, they show that whenever the bit rate is increased, the quality is increased. In addition, they show that the H.264 codec achieves the highest quality. Moreover, they show that the MPEG-4 and FLV codecs achieve the second and third highest quality, respectively, with close quality results.

Figures C.8, C.9, and C.10 in Appendix C show the quality evaluation results for exploring the impact of reducing the frame size on objective video quality aggregated based on the H.264, MPEG-4, and FLV codecs, respectively.

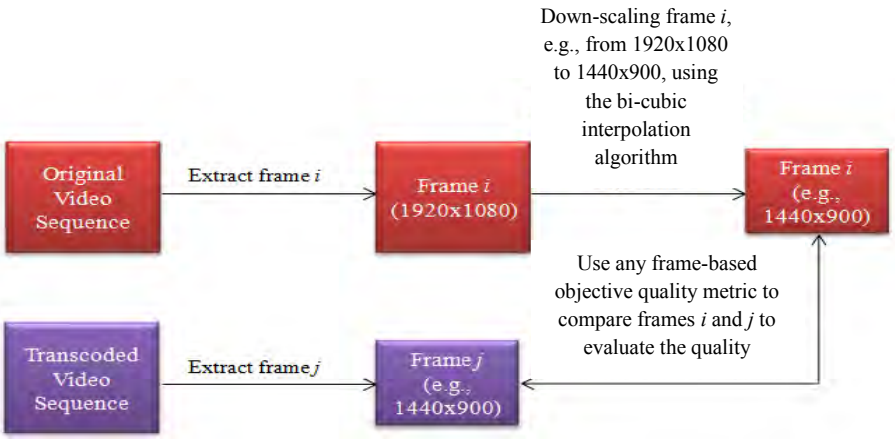
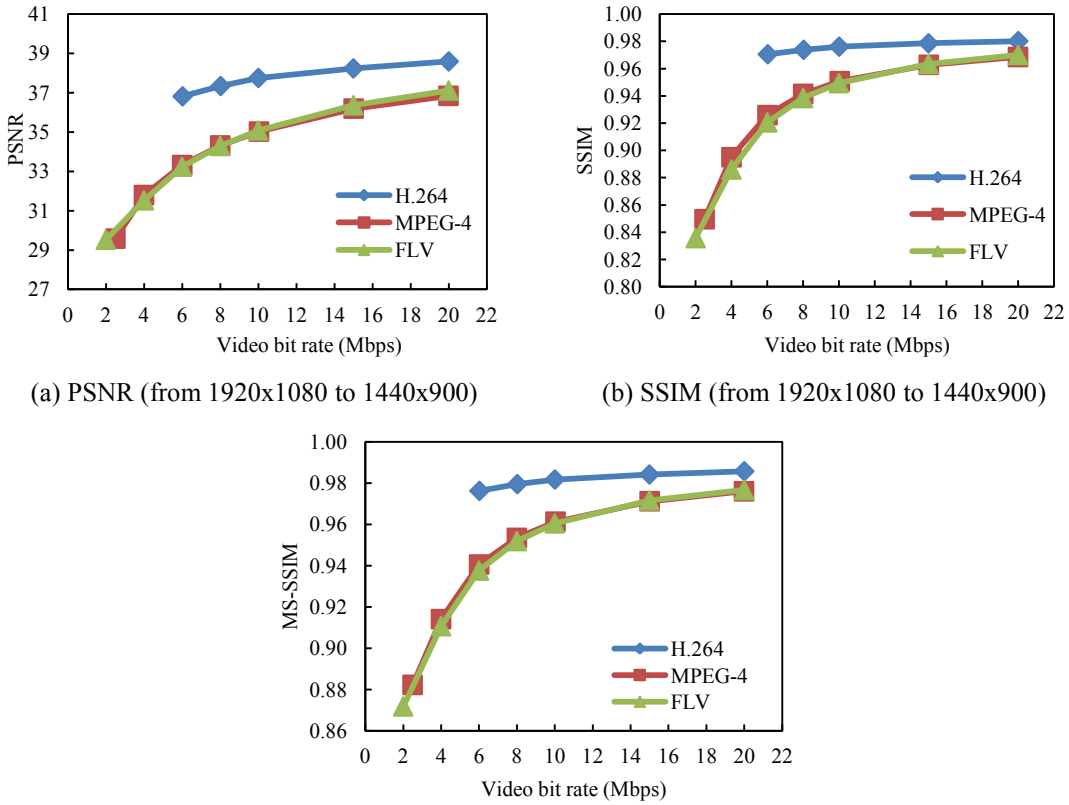


Figure 5.9. The general structure that shows the down-scaling strategy using the bi-cubic interpolation algorithm [43].



(a) PSNR (from 1920x1080 to 1440x900)

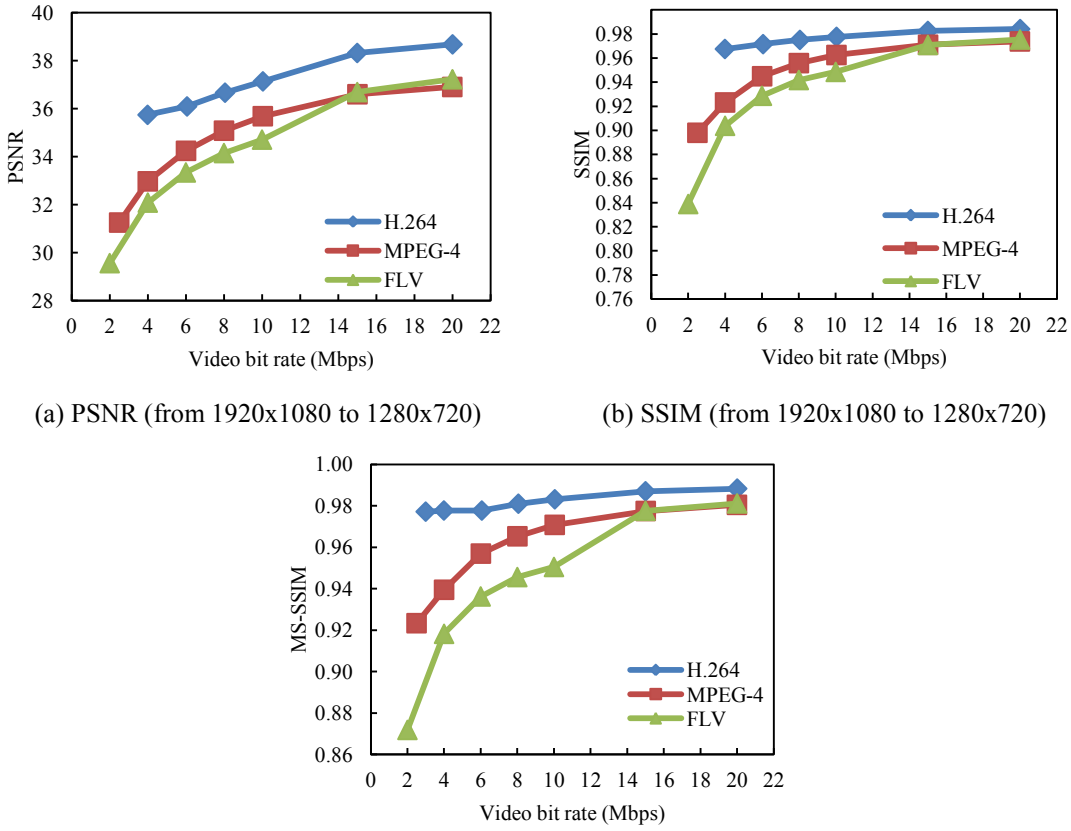
(b) SSIM (from 1920x1080 to 1440x900)

(c) MS-SSIM (from 1920x1080 to 1440x900)

Figure 5.10. The video quality evaluation results for exploring the impact of reducing the frame size on objective video quality for the H.264, MPEG-4, and FLV codecs. The frame size reduction is from 1920x1080 to 1440x900. The quality results are in terms of (a) PSNR, (b) SSIM, and (c) MS-SSIM.

5.6. Exploring the Impact of Reducing the Bit Rate

For exploring the impact of reducing the bit rate on objective video quality in video transcoding, I performed steps similar to those described above, but in this case to explore the impact of reducing the bit rate. I selected a set of bit rate reduction percentages, which includes 30%, 40%, and 50% from the original bit rate. The total number of videos used as input for each codec is 35, i.e., 5 different videos $*$ 7 different bit rates. Figure C.4 in Appendix C shows how I grouped these videos. In addition, it shows the names of some of the original and transcoded videos.



(a) PSNR (from 1920x1080 to 1280x720)

(b) SSIM (from 1920x1080 to 1280x720)

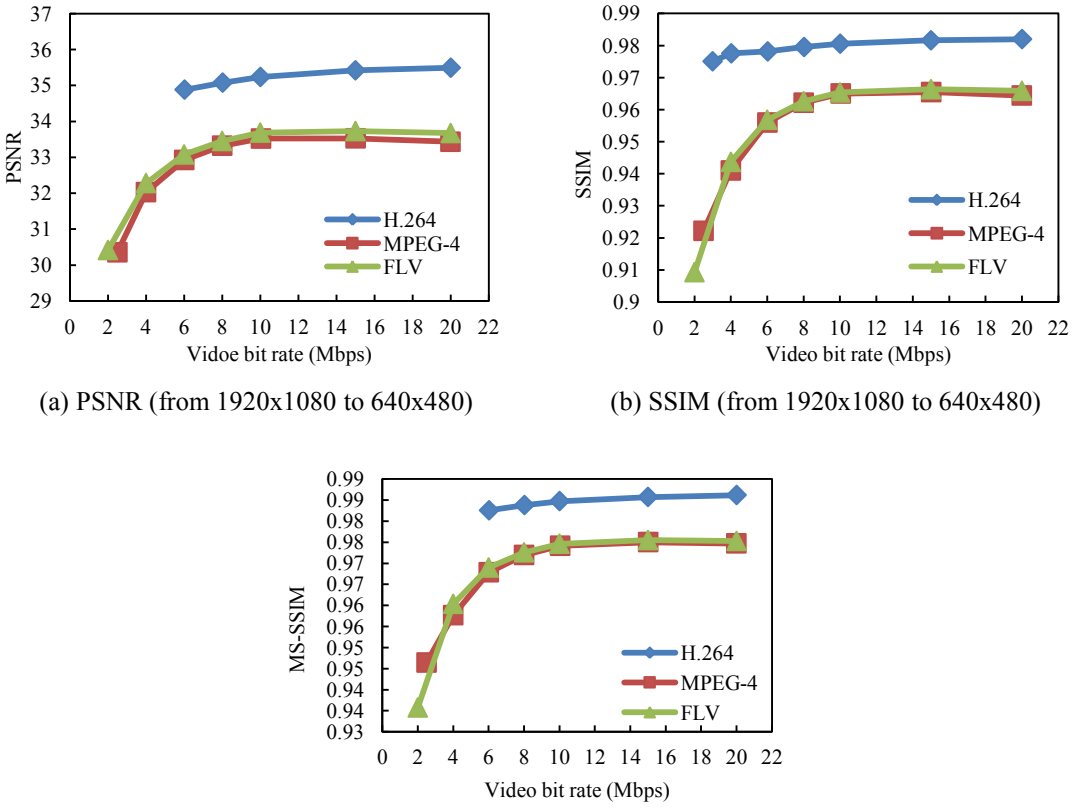
(c) MS-SSIM (from 1920x1080 to 1280x720)

Figure 5.11. The video quality evaluation results for exploring the impact of reducing the frame size on objective video quality for the H.264, MPEG-4, and FLV codecs. The frame size reduction is from 1920x1080 to 1280x720. The quality results are in terms of (a) PSNR, (b) SSIM, and (c) MS-SSIM.

During this transcoding, I kept the codec, frame rate, and frame size the same as in the original video. The only thing that I changed in this transcoding was the bit rate. The total number of videos generated as output for each codec, based on the input videos, is 105, i.e., 35 input videos $*$ 3 different bit rate reduction percentages.

Figures 5.13, 5.14, and 5.15 show the quality evaluation results for exploring the impact of reducing the bit rate on objective video quality for the H.264, MPEG-4, and FLV codecs. The reduction percentages are 30%, 40%, and 50% from the original bit rate. The quality results are in terms of PSNR, SSIM, and MS-SSIM. They show that whenever the bit rate is increased, the quality is increased. In addition, they show that the H.264 codec achieves the highest quality. The FLV and MPEG-4 codecs

achieve the second and third highest qualities, respectively. Moreover, the difference in the quality between the FLV and MPEG-4 codecs is slightly small; while the difference in the quality between the H.264 and the other two codecs, i.e., FLV and MPEG-4, is slightly bigger. This difference is increased when the reduction percentage of the bit rate is increased. See Figure 5.15.



(a) PSNR (from 1920x1080 to 640x480)

(b) SSIM (from 1920x1080 to 640x480)

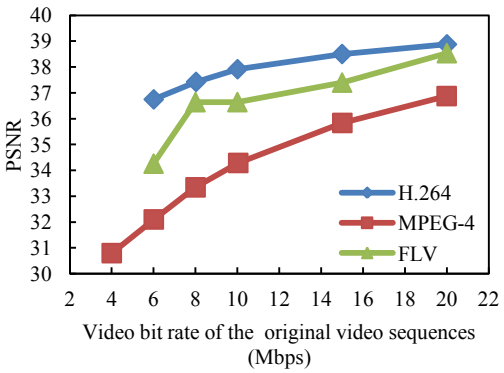
(c) MS-SSIM (from 1920x1080 to 640x480)

Figure 5.12. The video quality evaluation results for exploring the impact of reducing the frame size on objective video quality for the H.264, MPEG-4, and FLV codecs. The frame size reduction is from 1920x1080 to 640x480. The quality results are in terms of (a) PSNR, (b) SSIM, and (c) MS-SSIM.

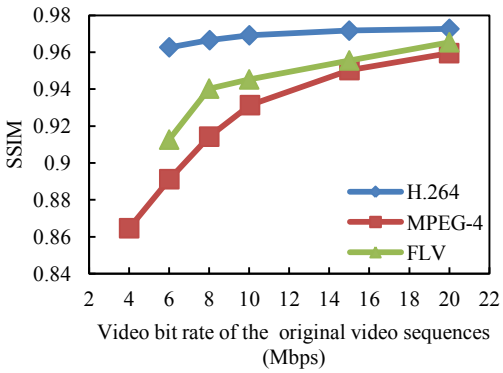
Figures 5.13, 5.14, and 5.15 also show that there is no big improvement in the quality in terms of SSIM and MS-SSIM for the H.264 codec when the bit rate is increased. In addition, they show that for the

FLV and MPEG-4 codecs, there is an improvement in the quality in terms of PSNR, SSIM, and MS-SSIM when the bit rate is increased.

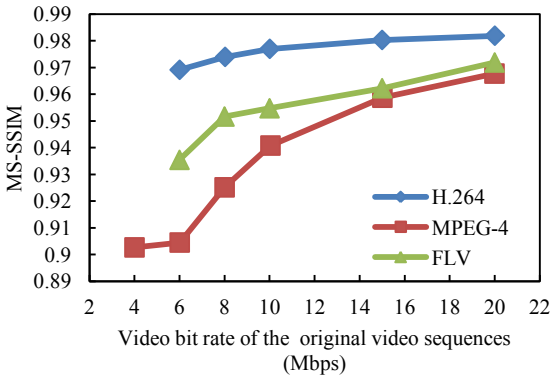
Figures C.11, C.12, and C.13 in Appendix C show the quality evaluation results for exploring the impact of reducing the bit rate on objective video quality aggregated based on the H.264, MPEG-4, and FLV codecs, respectively.



(a) PSNR (30% reduction in the bit rate)

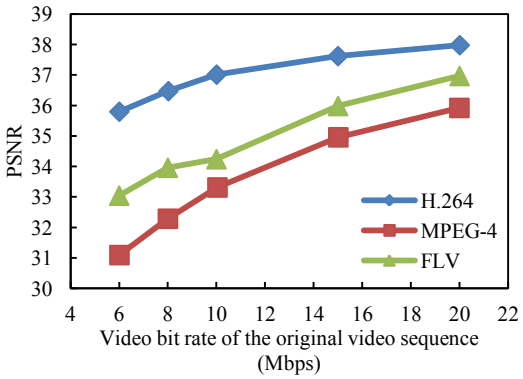


(b) SSIM (30% reduction in the bit rate)

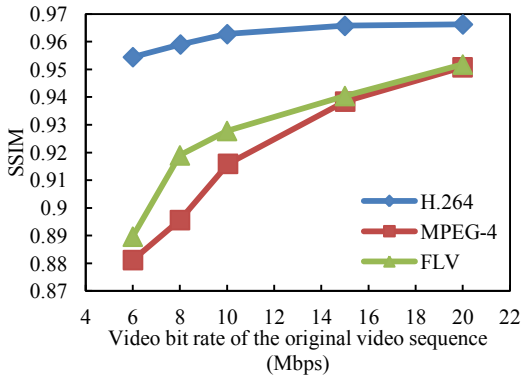


(c) MS-SSIM (30% reduction in the bit rate)

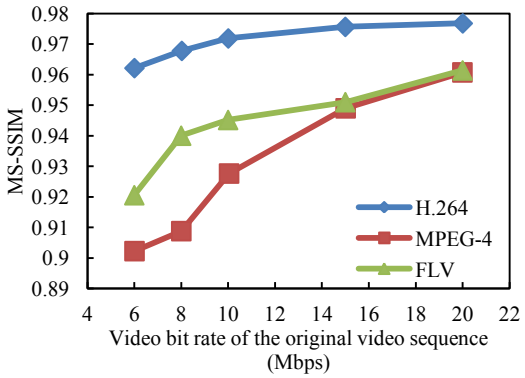
Figure 5.13. The quality evaluation results for exploring the impact of reducing the bit rate on objective quality for the H.264, MPEG-4, and FLV codecs. The reduction percentage is 30% from the original bit rate. The quality results are in terms of (a) PSNR, (b) SSIM, and (c) MS-SSIM.



(a) PSNR (40% reduction in bit rate)

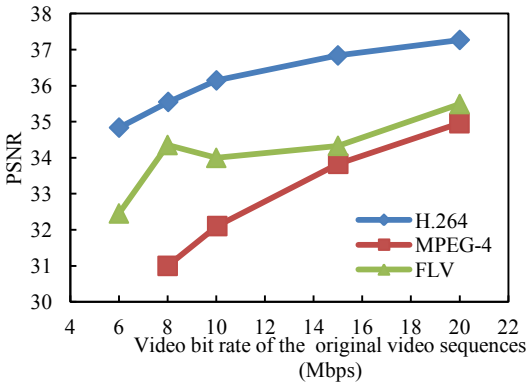


(b) SSIM (40% reduction in bit rate)

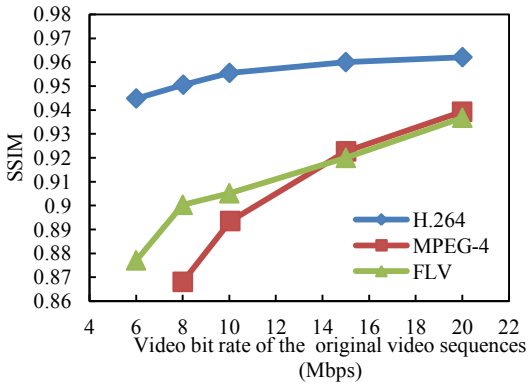


(c) MS-SSIM (40% reduction in bit rate)

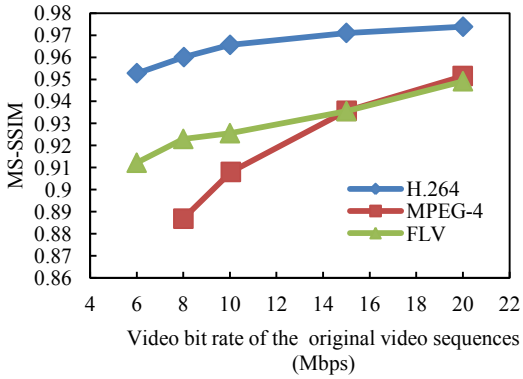
Figure 5.14. The quality evaluation results for exploring the impact of reducing the bit rate on objective quality for the H.264, MPEG-4, and FLV codecs. The reduction percentage is 40% from the original bit rate. The quality results are in terms of (a) PSNR, (b) SSIM, and (c) MS-SSIM.



(a) PSNR (50% reduction in bit rate)



(b) SSIM (50% reduction in bit rate)



(c) MS-SSIM (50% reduction in bit rate)

Figure 5.15. The quality evaluation results for exploring the impact of reducing the bit rate on objective quality for the H.264, MPEG-4, and FLV codecs. The reduction percentage is 50% from the original bit rate. The quality results are in terms of (a) PSNR, (b) SSIM, and (c) MS-SSIM.

CHAPTER 6

MODELING THE IMPACT OF VIDEO TRANSCODING ON OBJECTIVE VIDEO QUALITY

6.1. Overview

This chapter presents the major components of the VTOM, which includes four quality sub-models and four error sub-models. Each of these quality sub-models represents modeling the impact of one of the following on objective video quality: a) changing the codec, b) reducing the bit rate, c) reducing the frame rate, and d) reducing the frame size. Each of the error sub-models represents modeling the error in assessing the impact of one of the above on objective video quality using its corresponding quality sub-model.

To develop each of these four quality sub-models, I used five different videos and their explored results described in the previous chapter as a training set, which includes vqeghd1_csrc11, vqeghd1_csrc12, vqeghd1_csrc14, vqeghd1_src01, and vqeghd1_src03.

To test each of these four quality sub-models, I used four different videos as a testing set, which includes vqeghd1_src02, vqeghd1_src04, vqeghd1_src06, and vqeghd1_src07. To evaluate each of the quality sub-models, I compared the predicted quality results generated from each quality sub-model with quality values generated from MS-SSIM.

To develop and test each of these four error sub-models, I used the following four videos as training and testing sets: vqeghd1_src02, vqeghd1_src04, vqeghd1_src06, and vqeghd1_src07. To evaluate each of the error sub-models, I compared the predicted error results generated from each error sub-model with the difference between the actual and predicted quality results. The actual quality results are generated from using MS-SSIM, and the predicted quality results are generated from each of the quality sub-models.

For each of the above parameters and for each codec, I developed a mathematical function that assesses the quality of the transcoded video that is generated by using a specific transcoding function based on the format and characteristics of the original video and the requested values of the video transcoding parameters. For each of the above parameters and for each codec, I developed a mathematical function that

assesses the error in evaluating the quality of the transcoded video that is generated by using one of the quality sub-models. This chapter starts first by describing the quality sub-models. Then it will present and describe the error sub-models. Finally, it will depict the combination and evaluation strategies of the VTOM.

6.2. Modeling the Impact of Changing the Video Codec

To model the impact of changing, i.e., mapping, the codec on objective video quality, I addressed each mapping individually by providing a function that handles that mapping. This section describes one mapping, i.e., from H.264 to MPEG-4. Other mappings will be described in Appendix D.

6.2.1. H.264 to MPEG-4 Transcoding

Based on exploring the impact of changing the codec from H.264 to MPEG-4 on objective quality described in the previous chapter, I modeled the quality of the transcoded video that is generated from this transcoding function at a given bit rate, br in Mbps, as follows:

$$y.quality_{\alpha_{H.264,MPEG4}(x.br),x} = 0.9841 + \left(\frac{log_2(x.br+5.245)}{e^{log_2(x.br+5.245)} * x.br^{0.6581}}\right) \quad (6.1)$$

where x and y are the original and transcoded videos, respectively. The codec for video x is H.264 and for video y is MPEG-4. The value of this function ranges from 0 to 1; the higher the value, the better the quality.

To test this quality sub-model, I performed the following steps:

1. Encoded each video in the testing set described above using the H.264 codec and at different values of bit rates that range from 2 to 20 Mbps. The total number of encoded videos generated from this video encoding and for each codec is 28, i.e., 4 different videos in the testing set $*$ 7 different bit rates.
2. Transcoded these encoded videos to generate a new set of transcoded videos. In this step, I first decoded the encoded videos using the H.264 codec that was used in the encoding step. Second, I re-encoded them using the MPEG-4 codec. During this transcoding, I kept the bit rate, frame rate, and frame size the same as in the encoded videos. The only thing that I changed in this

transcoding was the codec. The total number of transcoded videos generated from this transcoding is 28.

3. Assessed the impact of changing the codec on objective quality using the MS-SSIM quality metric to generate the actual quality values. This assessment was done by comparing each pair of frames from both the original and transcoded videos and then taking the average of all these comparisons as a final result between any two videos.
4. Assessed the impact of changing the codec using (6.1) to generate the predicted quality values.
5. Evaluated the proposed quality sub-model using the PCC and RMSE by comparing the actual and the predicted quality values described above.

Figure 6.1 shows the quality evaluation results for modeling the impact of changing the codec on objective quality. These evaluation results are for each video in the testing set and the reference values, represented by black curves, that represent the average of the actual quality values for all videos used in the training set in terms of MS-SSIM at a given bit rate. In addition, it shows the model results that are represented by dashed-red curves. Moreover, Figure 6.1 shows high correlation results between the predicted quality values generated from these quality sub-models and both the reference quality values and the quality values generated from using MS-SSIM for each video in the testing set.

Table 6.1 shows the evaluation results for the H.264 to MPEG-4 codec conversion quality sub-model in terms of PCC and RMSE. It shows high PCC values between the predicted quality results generated from this quality sub-model and the quality results generated for each video in the testing set using the MS-SSIM. It also shows very low RMSE values. In addition, it shows the evaluation results for all the other codec conversion quality sub-models.

6.3. Modeling the Impact of Reducing the Bit Rate

To model the impact of reducing the bit rate on objective quality, I developed a model for each of the H.264, MPEG-4, and FLV codecs. This section describes the model that handles the H.264 codec. The models that handle the MPEG-4 and FLV codecs will be described in Appendix D.

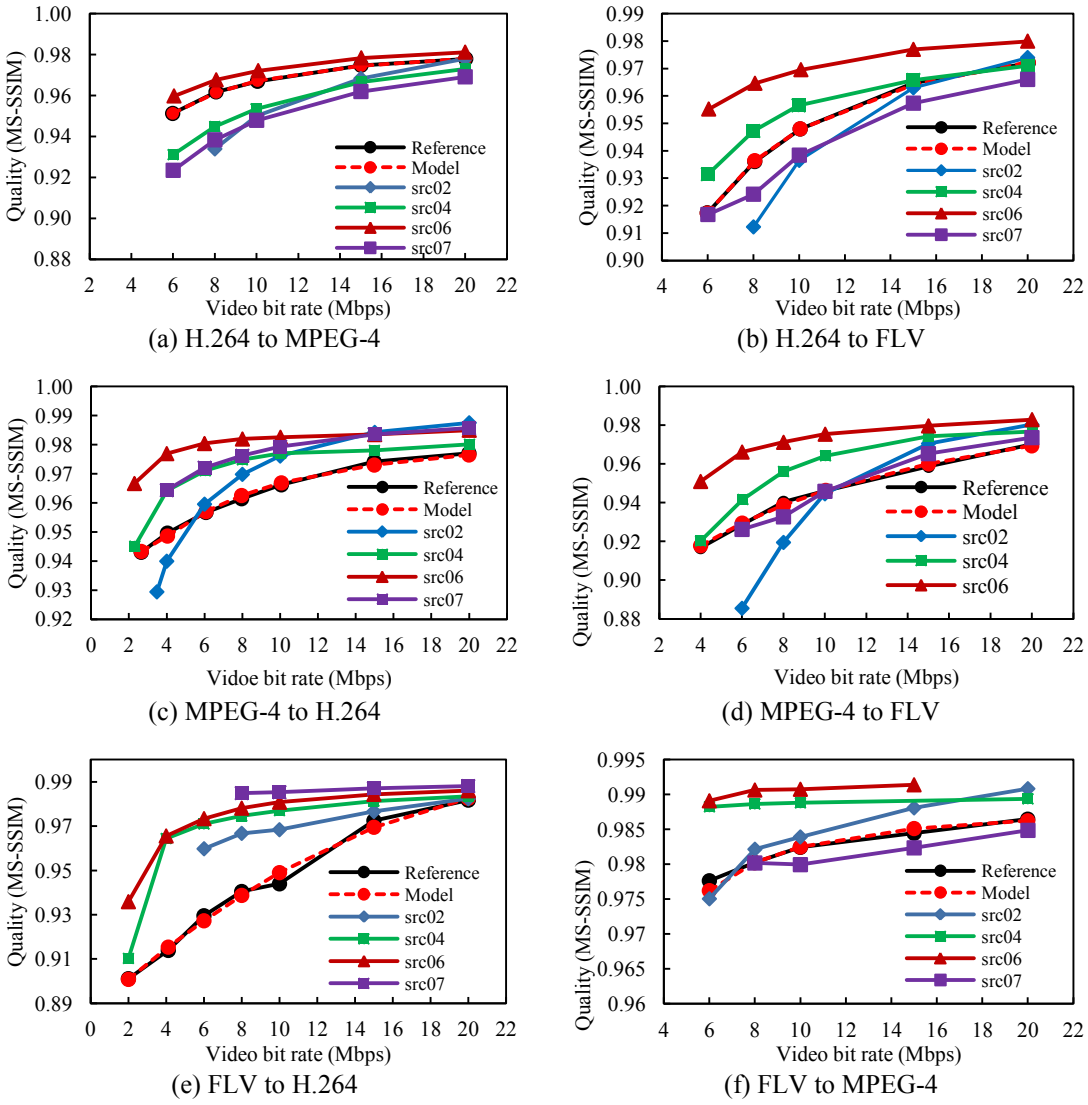


Figure 6.1. The video quality evaluation results for modeling the impact of changing the codec on objective video quality. These results are for each video in the testing set and the reference values that represent the average of the actual video quality values for all videos used in the training set in terms of MS-SSIM at a given bit rate. It also shows the results of these quality sub-models. The codec mappings are (a) from H.264 to MPEG-4, (b) from H.264 to FLV, (c) from MPEG-4 to H.264, (d) from MPEG-4 to FLV, (e) from FLV to H.264, and (f) from FLV to MPEG-4.

Table 6.1. The evaluation results in terms of PCC and RMSE for the quality sub-models that assess the impact of changing the codec on objective video quality.

Seq. #	H.264 to MPEG-4 model		H.264 to FLV model		MPEG-4 to H.264 model	
	PCC	RMSE	PCC	RMSE	PCC	RMSE
vqeghd1_src02	0.99960	0.01646	0.99997	0.01540	0.98413	0.01030
vqeghd1_src04	0.99787	0.01384	0.99913	0.00847	0.89817	0.01030
vqeghd1_src06	0.99982	0.00548	0.99615	0.02345	0.97975	0.01905
vqeghd1_src07	0.99785	0.01962	0.95360	0.00950	0.99710	0.01309
Average	**0.99879**	**0.01385**	**0.98722**	**0.01421**	**0.96479**	**0.01319**
Seq. #	MPEG-4 to FLV model		FLV to H.264 model		FLV to MPEG-4 model	
	PCC	RMSE	PCC	RMSE	PCC	RMSE
vqeghd1_src02	0.97568	0.02250	0.99325	0.02129	0.99349	0.00272
vqeghd1_src04	0.96260	0.01314	0.77474	0.03090	0.99611	0.00741
vqeghd1_src06	0.95015	0.02864	0.85254	0.03534	0.96720	0.00881
vqeghd1_src07	0.99344	0.00430	0.99461	0.03077	0.97061	0.00200
Average	**0.97047**	**0.01715**	**0.90379**	**0.02957**	**0.98185**	**0.00524**

6.3.1. H.264 Video Codec

Based on exploring the impact of reducing the bit rate on objective quality described in the previous chapter, I modeled the quality of the transcoded video that is generated from this transcoding function as follows:

$$y.quality_{\beta_{H.264}(x.c.br_{in},y.c.br_{out}),x} = a_1 + \left(\frac{log_2(br_{out}+0.002)*a_2}{log_2(br_{in}+3.47985)}\right) \quad (6.2)$$

where x and y are the original and transcoded videos, respectively. The codec for videos x and y is H.264. The bit rates for videos x and y are br_{in} and br_{out} in Mbps, respectively. The model coefficients are a_1 and a_2, calculated as follows:

$$a_1 = -0.444 * x^2 + 0.26293 * x + 0.88885 \quad (6.3)$$

$$a_2 = 0.7699 * x^2 - 0.496 * x + 0.14459 \quad (6.4)$$

the value of x represents the reduction percentage in the bit rate, calculated as follows:

$$x = \left(1 - \frac{br_2}{br_1}\right) \quad (6.5)$$

The value of the function described by (6.2) ranges from 0 to 1; the higher the value, the better the quality. The relationship between the model coefficients, i.e., a_1 and a_2, and the feature x is plotted in Figure 6.2, which shows high correlations between the equations, which calculate the model coefficients a_1 and a_2 based on the values of x, and the model coefficients.

To test this model, I performed steps similar to those described above, but in this case to test modeling the impact of reducing the bit rate on objective video quality for the H.264 codec. I selected 30%, 40%, and 50% as percentages for the bit rate reduction. The total number of encoded videos generated from video encoding is 28, i.e., 4 videos $*$ 7 different bit rates. During video transcoding, I kept the codec, frame rate, and frame size the same as in the original videos. The only thing that I changed in this transcoding was the bit rate. The total number of transcoded videos generated from this transcoding is 84, i.e., 28 input videos $*$ 3 reduction percentages.

Figure 6.3 shows the quality evaluation results for modeling the impact of reducing the bit rate on objective video quality for the H.264 codec. These evaluation results are for each video in the testing set and the reference values, shown using black curves, that represent the average of the actual quality values for all videos used in the training set in terms of MS-SSIM at a given bit rate. In addition, it shows the model results that are represented by dashed-red curves. Moreover, Figure 6.3 shows high correlation results between the predicted quality values generated from this quality sub-model and both the reference quality values and the quality results generated from using MS-SSIM for each video in the testing set.

Table 6.2 shows the evaluation results of modeling the impact of reducing the bit rate on objective video quality for the H.264 codec. It shows high PCC values between the predicted quality results

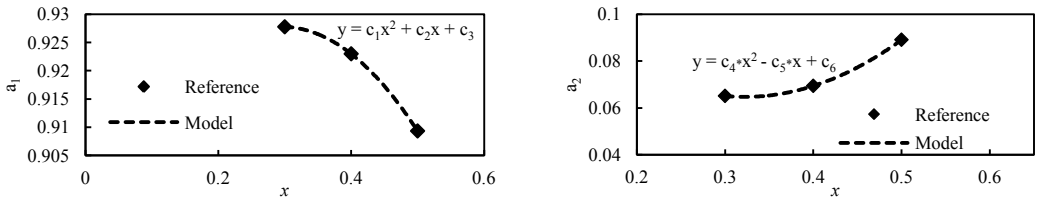
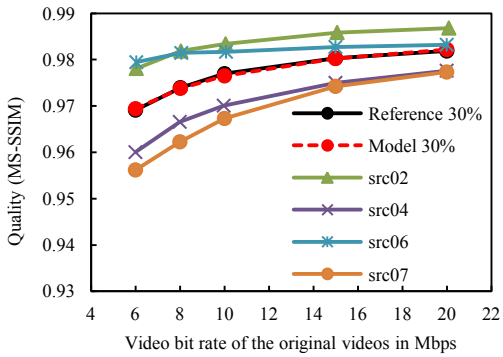


Figure 6.2. The relationship between the model coefficients (a) a_1 and (b) a_2 and the feature x for modeling the impact of reducing the bit rate on objective video quality for the H.264 codec.

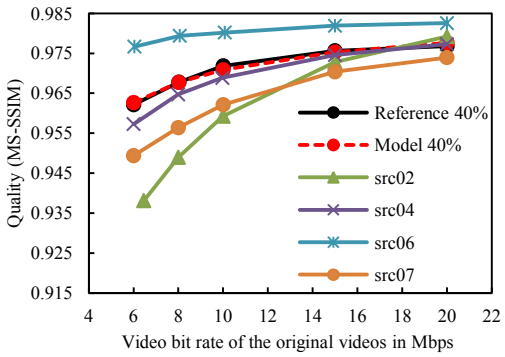
generated from this quality sub-model and the quality results for each video in the testing set generated using MS-SSIM. In addition, it shows very low RMSE values.

6.4. Modeling the Impact of Reducing the Frame Size

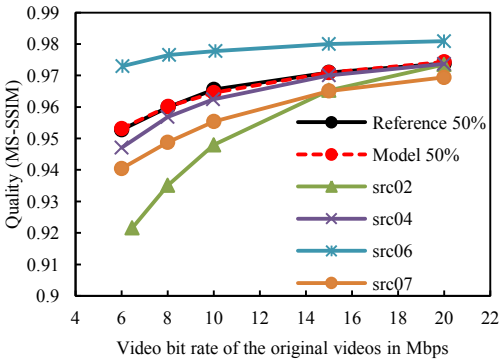
To model the impact of reducing the frame size on objective video quality, I developed a model for each of the H.264, MPEG-4, and FLV codecs. This section describes the model that handles the H.264 codec. The models that handle the MPEG-4 and FLV codecs will be described in Appendix D.



(a) Modeling the 30% bit rate reduction



(b) Modeling the 40% bit rate reduction



(c) Modeling the 50% bit rate reduction

Figure 6.3. The video quality evaluation results for modeling the impact of reducing the bit rate on objective video quality for the H.264 codec. These results are for each video in the testing set and the reference values that represent the average of the actual video quality values for all the videos used in the training set in terms of MS-SSIM at a given bit rate. It also shows the model results. The reduction percentages are (a) 30%, (b) 40%, and (c) 50% from the original bit rate.

Table 6.2. The evaluation results in terms of PCC and RMSE for the quality sub-model that assesses the impact of reducing the bit rate on objective video quality for the H.264 codec.

Seq. #	30% reduction		40% reduction		50% reduction	
	PCC	RMSE	PCC	RMSE	PCC	RMSE
vqeghd1_src02	0.99986	0.01246	0.99978	0.01526	0.99975	0.02025
vqeghd1_src04	0.99947	0.00679	0.99905	0.00290	0.99877	0.00322
vqeghd1_src06	0.98098	0.00613	0.99210	0.00981	0.99192	0.01378
vqeghd1_src07	0.99811	0.00953	0.99837	0.00913	0.99880	0.00929
Average	**0.99460**	**0.00873**	**0.99732**	**0.00928**	**0.99731**	**0.01164**

6.4.1. H.264 Video Codec

Based on exploring the impact of reducing the frame size on objective video quality for the H.264 codec described in the previous chapter, I modeled the quality of the transcoded video that is generated from this transcoding function and by reducing the frame size from fs_1 to fs_2 as follows:

$$y.quality_{\delta_{H.264}(x.c.br,x.c.fs_{in},y.c.fs_{out}),x} = 1 - \left[a_1 + \left(\frac{\left(\left\lceil\frac{fs_{out}.w}{fs_{in}.w}\right\rceil * (log_2(x.c.br + a_2))\right)}{\left(\left\lceil\frac{fs_{out}.h}{fs_{in}.h}\right\rceil * (log_2(x.c.br)^{a_3}) * (e^{log_2(x.c.br + a_4)})\right)}\right)\right] \quad (6.6)$$

where x and y are the original and transcoded videos, respectively. The codec for videos x and y is H.264. The frame size fs can be further described by $fs.w$ and $fs.h$ for width and height, respectively. The model coefficients are a_1, a_2, a_3, and a_4, calculated as follows:

$$a_1 = -0.4549 * x^2 + 0.3762 * x - 0.065 \quad (6.7)$$

$$a_2 = -1359.39481 * x^2 + 1024.407 * x - 190.52608 \quad (6.8)$$

$$a_3 = 156.34138 * x^2 - 112.4538 * x + 21.75114 \quad (6.9)$$

$$a_4 = -1417.29546 * x^2 + 1109.498 * x - 206.36664 \quad (6.10)$$

the value of x represents the relationship between the sizes of the frames, calculated as follows:

$$x = \frac{fs_2.h * fs_1.h}{fs_2.w * fs_1.w} \quad (6.11)$$

The value of the function described by (6.6) ranges from 0 to 1; the higher the value, the better the quality. The relationship between the model coefficients, i.e., a_1 to a_4, and the feature x is plotted in Figure

6.4, which shows high correlations between the equations, which calculate the model coefficients a_1 to a_4 based on the values of x, and the model coefficients.

To test this quality sub-model, I performed steps similar to those described above, but in this case to test modeling the impact of reducing the frame size on objective video quality for the H.264 codec. I selected three different new frame sizes: 1440x900, 1280x720, and 640x480. The total number of encoded videos generated from video encoding is 28. During video transcoding, I kept the codec, bit rate, and frame rate the same as in the original videos. The only thing that I changed in this transcoding was the frame size. The total number of transcoded videos generated from this transcoding is 84, i.e., 28 input videos $*$ 3 different frame sizes. Because the frames from both the original and transcoded videos have different sizes, I used the bi-cubic interpolation algorithm to down-scale the frame sizes as described in the previous chapter.

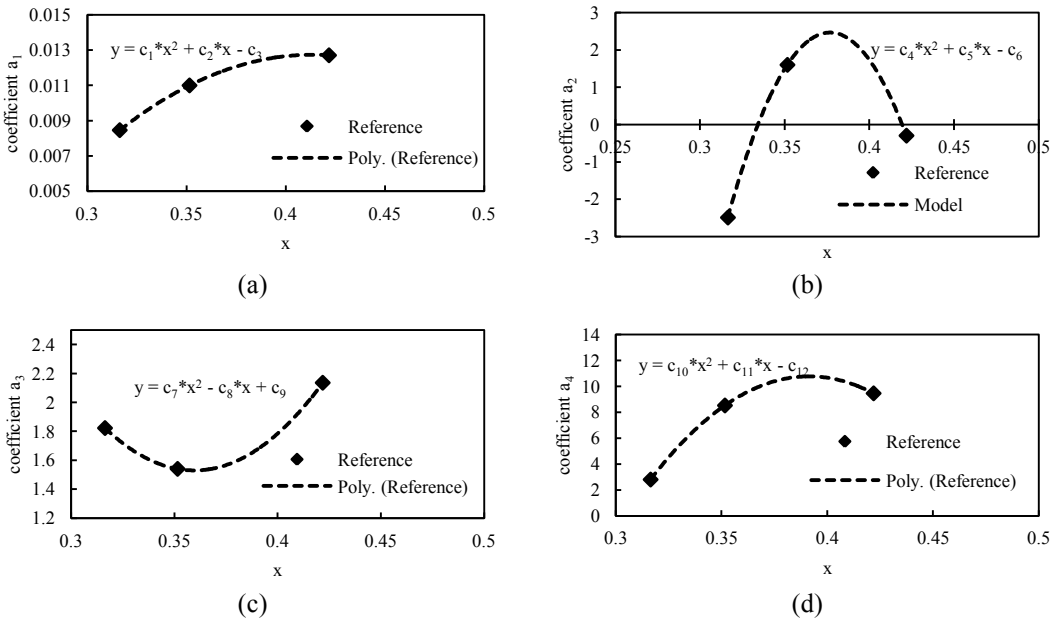
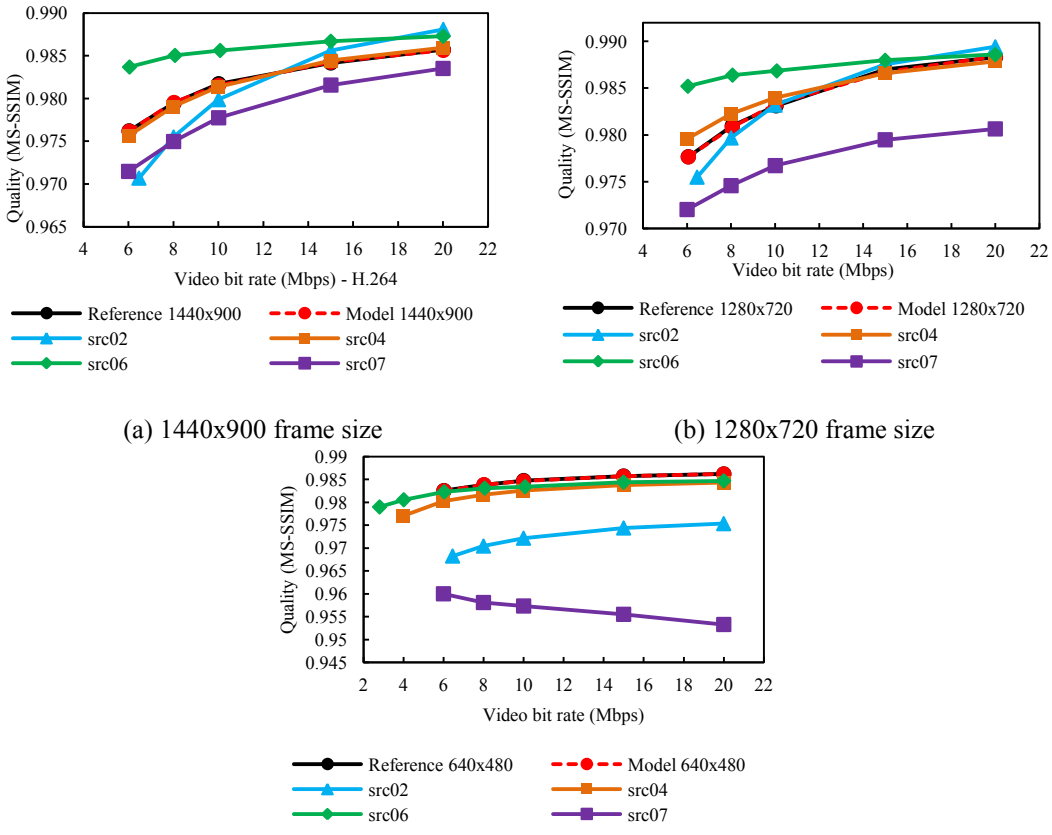


Figure 6.4. The relationship between the model coefficients (a) a_1, (b) a_2, (c) a_3, and (d) a_4 and the feature x for modeling the impact of reducing the frame size on objective video quality for the H.264 codec.

Figure 6.5 shows the quality evaluation results for modeling the impact of reducing the frame size on objective video quality for the H.264 codec. These evaluation results are for each video in the testing set and the reference values, shown using black curves, that represent the average of the actual quality values for all videos used in the training set in terms of MS-SSIM at a given bit rate. In addition, it shows the model results that are represented by dashed-red curves. Moreover, Figure 6.3 shows high correlation results between the predicted quality values generated from this quality sub-model and both the reference quality values and the quality results generated from using MS-SSIM for each video in the testing set, except for the vqeghd1_src07 video in case of 640x480 frame size.



(a) 1440x900 frame size

(b) 1280x720 frame size

(c) 640x480 frame size

Figure 6.5. The video quality evaluation results for modeling the impact of reducing the frame size on objective video quality for the H.264 codec. The quality results are for each video in the testing set and the reference values that represent the average of the actual video quality values for all the videos used in the training set in terms of MS-SSIM. The frame size reductions are: (a) from 1920x1080 to 1440x900, (b) from 1920x1080 to 1280x720, and (c) from 1920x1080 to 640x480.

Table 6.3 shows the evaluation results of modeling the impact of reducing the frame size on objective video quality for the H.264 codec. It shows high PCC values between the predicted quality results generated from this quality sub-model and the quality results for each video in the testing set generated by using the MS-SSIM metric. In addition, it shows very low RMSE values, except for the vqeghd1_src07 video in case of 640x480 frame size.

6.5. Modeling the Impact of Reducing the Frame Rate

To model the impact of reducing the frame rate on objective video quality, I developed a model for each of the H.264, MPEG-4, and FLV codecs. This section describes the model that handles the H.264 codec. The models that handle the MPEG-4 and FLV codecs will be described in Appendix D.

6.5.1. H.264 Video Codec

Based on exploring the impact of reducing the frame rate on objective video quality for the H.264 codec described in the previous chapter, I modeled the quality of the transcoded video that is generated by reducing the frame rate from fr_{in} to fr_{out} as follows:

$$y.quality_{\gamma_{H.264}(x.c.br, x.c.fr_{in}, y.c.fr_{out}), x} = \frac{a_1 * (y.c.fr_{out}/x.c.fr_{in}) * x.c.br}{a_2 * x.c.br^{a_3}} \quad (6.12)$$

where x and y are the original and transcoded videos, respectively. The codec for videos x and y is H.264. The frame rates for videos x and y are fr_{in} and fr_{out}, respectively. The model coefficients are a_1, a_2, and a_3, calculated as follows:

Table 6.3. The evaluation results in terms of PCC and RMSE for the quality sub-model that assesses the impact of reducing the frame size on objective video quality for the H.264 codec.

Seq. #	1440x900		1280x720		640x480	
	PCC	RMSE	PCC	RMSE	PCC	RMSE
vqeghd1_src02	0.99981	0.00368	0.99757	0.00147	0.99975	0.01260
vqeghd1_src04	0.99972	0.00035	0.99954	0.00119	0.99998	0.00216
vqeghd1_src06	0.99884	0.00470	0.99614	0.00452	0.99343	0.00155
vqeghd1_src07	0.99735	0.00369	0.99978	0.00668	-0.97095	0.02794
Average	**0.99893**	**0.00310**	**0.99826**	**0.00346**	**0.50555**	**0.01106**

$$a_1 = -6.62819 * x^2 + 9.74811 * x - 4.51634 \quad (6.13)$$

$$a_2 = -4.14883 * x^2 + 4.97475 * x - 2.19671 \quad (6.14)$$

$$a_3 = 0.13222 * x^2 - 0.17909 * x + 1.05092 \quad (6.15)$$

the value of x represents the relationship between the two frame rates, calculated as follows:

$$x = \frac{y.c.fr_2}{x.c.fr_1} \quad (6.16)$$

The value of the function described by (6.12) ranges from 0 to 1; the higher the value, the better the quality. The relationship between the model coefficients, i.e., a_1 to a_3, and the feature x is plotted in Figure 6.6, which shows high correlations between the equations, which calculate the model coefficients a_1 to a_3 based on the values of x, and the model coefficients.

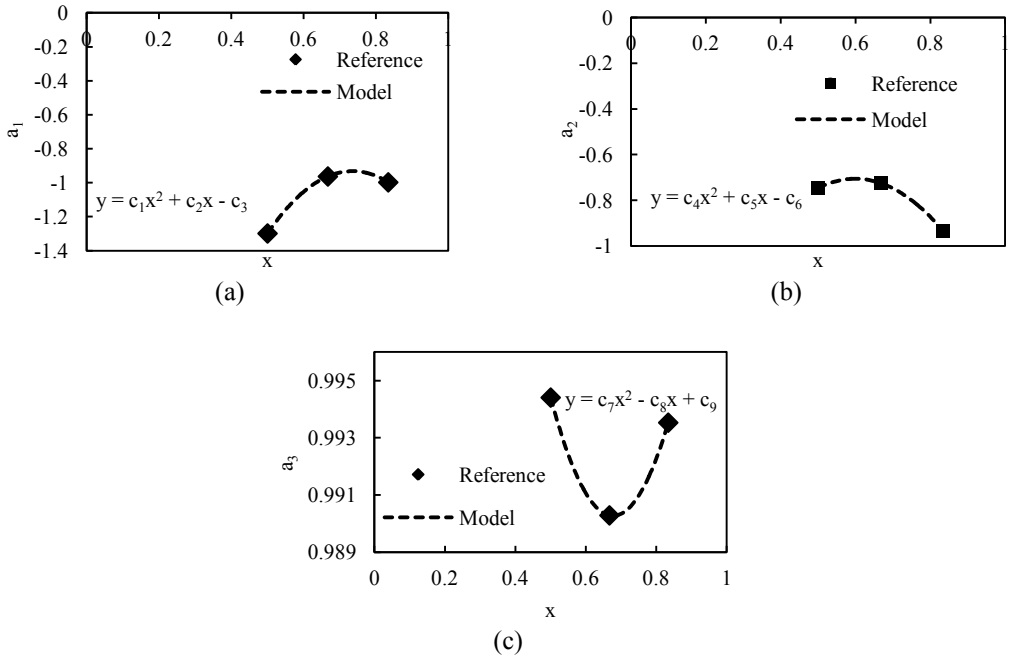


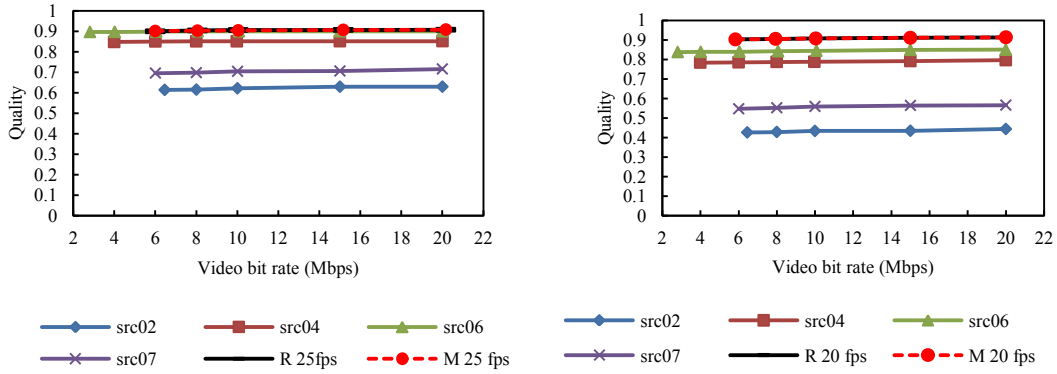
Figure 6.6. The relation between the model coefficients (a) a_1, (b) a_2, and (c) a_3 and the feature x for modeling the impact of reducing the frame rate on objective video quality for the H.264 codec.

To test this sub-model, I performed steps similar to those described above, but in this case to test modeling the impact of reducing the frame rate on objective video quality for the H.264 codec. The frame rate reductions are from 29.97 fps to 25 fps, 20 fps, and 15 fps for all the videos used in the testing set. The total number of encoded videos generated from the video encoding is 28. During video transcoding, I kept the codec, bit rate, and frame size the same as in the original videos. The only thing that I changed in this transcoding was the frame rate. The total number of transcoded videos generated from this transcoding is 84, i.e., 28 input videos $*$ 3 different frame rates.

Figure 6.7 shows the quality evaluation results for modeling the impact of reducing the frame rate on objective video quality for the H.264 codec. These quality results are for each video in the testing set and the reference values, labeled by "R", that represent the average of the actual quality values for all the videos that are used in the training set in terms of MS-SSIM at a given bit rate. It also shows the model results, labeled by "M".

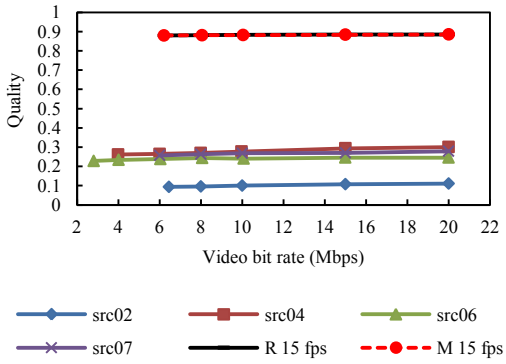
Figure 6.7(a) shows that two videos from the testing set, i.e., src04 and src06, have results that are close to the model results. The other two videos, i.e., src02 and src07, have results that are slightly farther from the model results.

Figures 6.7(b) and (c) show that the quality results of the videos that are in the testing set start to move away from the model results. This is unexpected and indicates that this quality sub-model needs to consider other factors, such as video content and motion level. In addition, the proposed metric, i.e., FRM, needs to be evaluated subjectively. However, all the videos that are used in the testing set have high correlation results with the model results as shown in Table 6.4. This table shows high PCC values between the quality results generated from this quality sub-model and the quality results for each video in the testing set generated from using MS-SSIM. Also, it shows that the RMSE is increased when the reduction of the frame rate is increased.



(a) (from 29.97 to 25 fps)

(b) (from 29.97 to 20 fps)



(c) (from 29.97 to 15 fps)

Figure 6.7. The video quality evaluation results for modeling the impact of reducing the frame rate on objective video quality for the H.264 codec. The quality results are for each video in the testing set and the reference values that represent the average of the actual video quality values for all the videos used in the training set in terms of MS-SSIM at a given bit rate. The frame rate reductions are (a) from 29.97 to 25 fps, (b) from 29.97 to 20 fps, and (c) from 29.97 to 15 fps.

Table 6.4. The evaluation results in terms of PCC and RMSE for the quality sub-model that assesses the impact of reducing the frame rate on objective video quality for the H.264 codec.

Seq. #	From 29.97 to 25 fps		From 29.97 to 20 fps		From 29.97 to 15 fps	
	PCC	RMSE	PCC	RMSE	PCC	RMSE
vqeghd1_src02	0.96944	0.28259	0.95016	0.47459	0.99559	0.78074
vqeghd1_src04	0.99380	0.05338	0.97530	0.11806	0.99216	0.60104
vqeghd1_src06	0.69477	0.00537	0.98677	0.06286	0.85144	0.63921
vqeghd1_src07	0.95779	0.20039	0.97334	0.34999	0.97642	0.61474
Average	**0.90395**	**0.13543**	**0.97139**	**0.25137**	**0.95390**	**0.65894**

6.6. Modeling the Error

As mentioned in Chapter 4, the error represents the error in assessing the quality of the transcoded video generated from using any of the quality sub-models mentioned above and in Appendix D. This is interpreted as the lower the error value, the better the expectation.

This section describes the error sub-models for the codec conversions. Other error sub-models for assessing the error in evaluating the impact of the following on objective video quality will be described in Appendix D: a) reducing the bit rate, b) reducing the frame rate, and c) reducing the frame size.

6.6.1. Modeling the Error for the Video Codec Conversions

I represented the error in assessing the quality of the transcoded video y generated by transcoding video x and changing the codec during this transcoding at a given bit rate, br in Mbps, as follows:

$$Error^{\alpha}_{f_1,f_2}(br) = \frac{Log_2(br*c_1)*c_2}{(br^{c_3}+c_4)} \quad (6.17)$$

where c_1 to c_4 are constants and their values are specified, based on each codec conversion, in Table 6.5. The codec for the original video x is f_1 and for the transcoded video y is f_2. This error sub-model is for all the codec conversions, except from FLV to MPEG-4. The FLV to MPEG-4 codec conversion error sub-model is represented using a different function at a given bit rate, br in Mbps, as follows:

$$Error^{\alpha}_{FLV,MPEG4}(br) = \frac{Log_2(br+13.24194)*1.26922}{(e^{log_2(br*3.72561)}+\sqrt{(br*2597.46256)})} \quad (6.18)$$

Figure 6.8 shows the predicted and reference error values for the following codec mappings: a) from H.264 to MPEG-4, b) from H.264 to FLV, c) from MPEG-4 to H.264, d) from MPEG-4 to FLV, e) from FLV to H.264, and f) from FLV to MPEG-4. The predicted error values, labeled by "M", are

Table 6.5. The values of the constants c_1 to c_4 for the error sub-models that assess the error in modeling the impact of changing the codec on objective video quality.

$x.f_1$	$y.f_2$	c_1	c_2	c_3	c_4
MPEG-4	H.264	910.35809	0.00662	0.76065	1.73597
H.264	MPEG-4	0.23347	5.36705	2.64298	24.82882
H.264	FLV	910.35813	0.00772	1.02595	-1.66021
FLV	H.264	909.96502	2.06561	2.71721	455.33794
MPEG-4	FLV	38.43219	0.00115	0.31490	-1.38388

generated from the codec conversion error sub-models and are calculated using (6.17) and (6.18) described above. The reference error values, labeled by "R", represent the average of the absolute difference between the actual quality values generated from MS-SSIM and the predicted quality values generated from the codec conversion quality sub-models described above at a given bit rate, br in Mbps, for all the videos used in the training set.

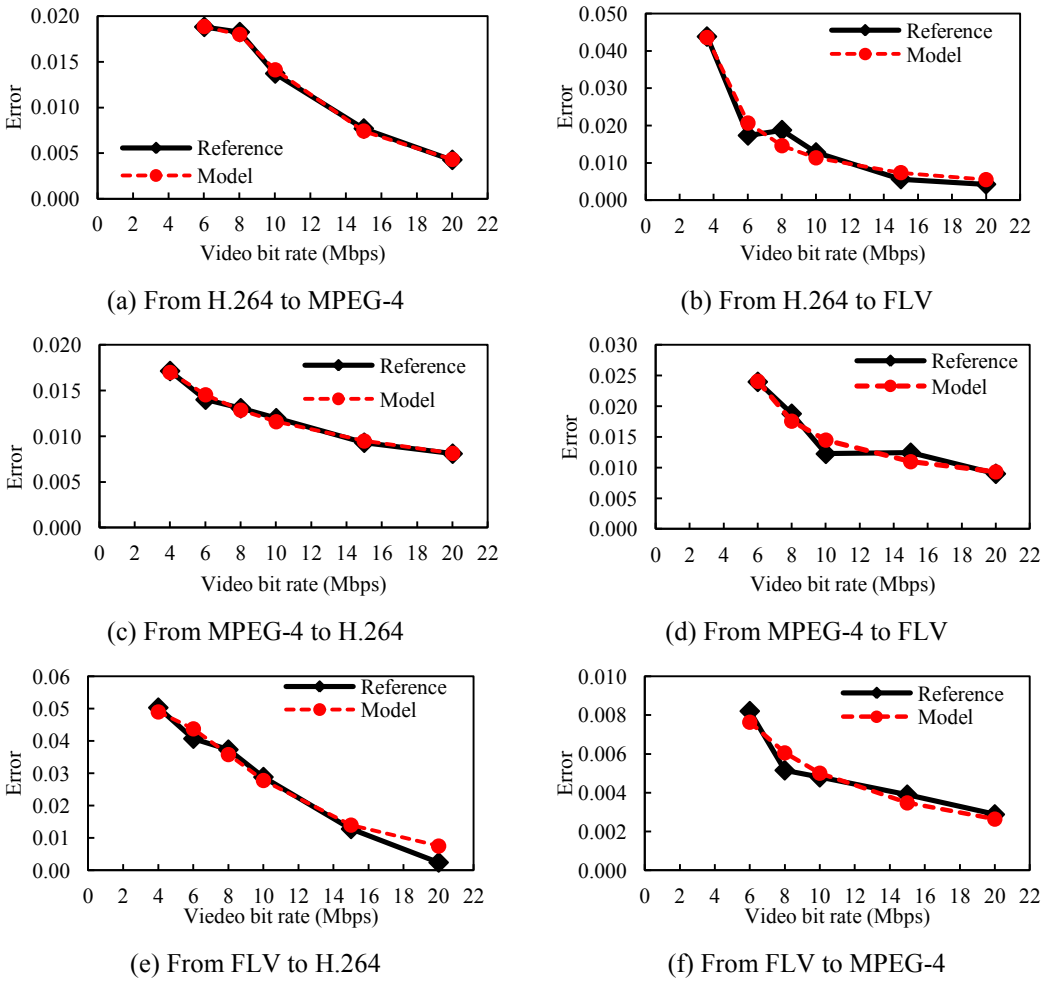


Figure 6.8. The reference and predicted error values that represent the error in assessing the impact of changing the codec on objective video quality. The black curves represent the reference error values and the red curves represent the predicted error values generated from the proposed error sub-models. The reference error values represent the average of the differences between the actual video quality values generated from MS-SSIM and the predicted video quality values generated from the proposed codec conversion quality sub-models for all the videos in the training set at a given bit rate. The codec mappings are (a) from H.264 to MPEG-4, (b) from H.264 to FLV, (c) from MPEG-4 to H.264, (d) from MPEG-4 to FLV, (e) from FLV to H.264, and (f) from FLV to MPEG-4.

Table 6.6 shows the evaluation results in terms of PCC and RMSE for these error sub-models that assess the error in evaluating the impact of changing the codec on objective video quality. These results show high PCC values between the reference and predicted error values, with very low RMSE values.

Table 6.6. The evaluation results in terms of PCC and RMSE for the error sub-models that assess the error in modeling the impact of changing the codec on objective video quality.

Codec Conversion	PCC	RMSE
From H.264 to MPEG-4	0.99907	0.00025
From H.264 to FLV	0.98281	0.00242
From MPEG-4 to H.264	0.99493	0.00030
From MPEG-4 to FLV	0.96939	0.00132
From FLV to H.264	0.99013	0.00178
From FLV to MPEG-4	0.95599	0.00053

6.7. General Models

As previously mentioned, VTOM combines the four quality sub-models with the four error sub-models in a single weighted product aggregation function that measures the overall quality of the perceived, i.e., transcoded, video. VTOM typically presents a mathematical function that estimates the perceived quality based on different parameters. Thereby, the quality estimation is computed as the result of a direct mathematical function.

To combine all of the above sub-models, i.e., quality and error sub-models, to generate more general models, I addressed each mapping individually. This means that I provided six general models; each handles a specific mapping. This section describes one mapping, i.e., from MPEG-4 to H.264. Other mappings will be described in Appendix D.

6.7.1. MPEG-4 to H.264 General Model

Based on the definitions described in Chapters 3 and 4, I modeled the quality of the transcoded video y generated by changing the codec from MPEG4 to H.264, reducing the bit rate from br_1 to br_2, reducing the frame rate from fr_1 to fr_2, and reducing the frame size from fs_1 to fs_2 for any original video x as follows:

$$y.quality_{VTOM,x} =$$

$$\alpha_{MPEG4,H.264}(x.c.br)^{w_c} * \beta_{H.264}(x.c.br_{in}, y.c.br_{out})^{w_{br}} * \gamma_{H.264}(x.c.br, x.c.fr_{in}, y.c.fr_{out})^{w_{fr}} *$$

$$\delta_{H.264}(x.c.br, x.c.fs_{in}, y.c.fs_{out})^{w_{fs}} \quad (6.19)$$

where x and y represent the original and transcoded videos, respectively. Each of the above quality terms represents a quality sub-model described above and in Appendix D. The exponent values of each of the above terms represent the weight of each term in the whole function. For example, w_c represents the weight of the codec conversion quality sub-model, calculated as follows (using (4.11) and (6.17)):

$$w_c = 1 - \left(Error^{\alpha}_{MPEG4,H.264}(br)\right) = 1 - \left(\frac{Log_2(br*910.35809)*0.00662}{(br^{0.76065}+1.73597)}\right) \quad (6.20)$$

I calculated the other weight values using a similar way. Generally, I calculated the weight based on (4.11) described in Chapter 4 and substitute the error function for each transcoding parameter. To test this general model, and any other general mapping from the VTOM, I performed the following steps:

1. I used the following four original, raw, uncompressed videos as a testing set: vqeghd1_src02, vqeghd1_src04, vqeghd1_src06, and vqeghd1_src07 to generate a new set of encoded videos using the above three codecs at bit rate values ranging from 2 to 20 Mbps.
2. I implemented 780 transcoding functions to generate 780 transcoded videos as a testing set with different combinations of formats and values of the transcoding parameters. During video transcoding, I modified the values for all the video transcoding parameters for the six codec conversions, i.e., mappings, mentioned earlier. Specifically, for each codec conversion, I reduced the bit rate by 30% and 40% from the original bit rate, I reduced the frame rate from 29.97 fps to 25 fps and 20 fps, and I reduced the frame size from 1920x1080 to 1440x900 and 1280x720.
3. I used the above transcoding functions to generate a new set of transcoded videos based on above selections.
4. I used the MS-SSIM objective quality metric to compare the original and transcoded videos and then I recorded these quality results.
5. I used the four quality sub-models to assess the impact of using each of the above transcoding parameters, and then I recorded the results. In addition, I recorded the results I got from using

the four error sub-models to assess the error in estimating the quality values generated from the above quality sub-models.

6. I combined the recorded results generated from the quality and error sub-models, e.g., based on (6.19), to generate a final result as a quality of the transcoded video based on the input video and transcoding function used. This result represents the result returned from VTOM.
7. Finally, I evaluated the VTOM by measuring the degree of correlation between the recorded results I got from using the MS-SSIM and the results I got from using the VTOM. I calculated the degree of correlation using the PCC and the degree of prediction using the RMSE.

Table 6.7 describes a sub-set from the above 780 transcoding function that evaluates the MPEG-4 to H.264 general model. This subset includes around 130 transcoding functions; each function has specific transcoding values. Each row in Table 6.7 represents a summary of the results of 10 of these functions for a specific video in the testing set. These transcoding functions reduce the values of all the transcoding parameters from the original values to new specific values. The values of the original bit rates are ranging from 2 to 20 Mbps. As a result, this table shows that the MPEG-4 to H.264 general model achieves in average a high PCC value, i.e., 0.93, and a low RMSE value, i.e., 0.13.

6.8. Discussion, Observations, and Further Experiments

This section provides a general discussion about the results of further experiments that I

Table 6.7. The evaluation results for the MPEG-4 to H.264 general model.

Seq. #	video codec		Frame rate		Frame size		Reduction in bit rate	PCC	RMSE
	From	To	From	To	From	To			
vqeghd1_src02	MPEG-4	H.264	29.97	25	1920x1080	1280x720	30%	0.83	0.21
vqeghd1_src04	MPEG-4	H.264	29.97	25	1920x1080	1280x720	30%	0.98	0.02
vqeghd1_src06	MPEG-4	H.264	29.97	25	1920x1080	1280x720	30%	0.86	0.07
vqeghd1_src07	MPEG-4	H.264	29.97	25	1920x1080	1280x720	30%	0.90	0.14
vqeghd1_src02	MPEG-4	H.264	29.97	20	1920x1080	1280x720	40%	0.97	0.36
vqeghd1_src04	MPEG-4	H.264	29.97	20	1920x1080	1280x720	40%	0.94	0.04
vqeghd1_src06	MPEG-4	H.264	29.97	20	1920x1080	1280x720	40%	0.98	0.02
vqeghd1_src07	MPEG-4	H.264	29.97	20	1920x1080	1280x720	40%	0.96	0.29
vqeghd1_src02	MPEG-4	H.264	29.97	25	1920x1080	1440x900	40%	0.82	0.21
vqeghd1_src04	MPEG-4	H.264	29.97	25	1920x1080	1440x900	40%	0.99	0.03
vqeghd1_src06	MPEG-4	H.264	29.97	25	1920x1080	1440x900	40%	0.93	0.09
vqeghd1_src07	MPEG-4	H.264	29.97	25	1920x1080	1440x900	40%	0.95	0.13
Average								**0.93**	**0.13**

conducted. In addition, it presents some observations during this research.

6.8.1. Normalizing the Error

As I mentioned in Chapter 4, the weight of each QoS parameter, i.e., transcoding parameter, represents how each of them affects the overall quality. For each transcoding parameter, I calculated its weight by using the error sub-model for that parameter and using (4.11) in Chapter 4. The error represents how successfully I predict the results of each quality sub-model. This is interpreted as the more error, the less accurate quality results generated from the proposed quality sub-models. In contrast, the less error, the more accurate quality results generated from these quality sub-models.

Normalizing the error is done by using the error of each parameter to calculate its weight and making the summation of all the weights generated from these errors equals to 1. The normalization is done based on the following steps:

1) Calculate the error, $Error_i$, for each parameter i using its corresponding error sub-model as described above and in Appendix D.

2) Sum up all the error values for all the parameters as follows:

$$sum = \sum_{i=1}^{4} Error_i \quad (6.21)$$

3) Normalize the error as follows:

$$Error'_i = \frac{Error_i}{sum} \quad (6.22)$$

4) Continue normalizing the error as follows:

$$Error''_i = 1 - Error'_i \quad (6.23)$$

5) Sum up the new normalization of the errors as follows:

$$sum' = \sum_{i=1}^{4} Error''_i \quad (6.24)$$

6) Calculate the weight as follows:

$$weight_i = \frac{Error''_i}{sum'} \quad (6.25)$$

As a result, normalizing the error decreased the PCC and increased the RMSE values, which generates results that start to deviate from the reference results. Using the normal weight, i.e., as described

in (4.11), increased the PCC by 0.2% and decreased the RMSE by 36.2%. Therefore, I concluded from this experiment that using the normal weight enhances the results of the VTOM.

6.8.2. Importance of the QoS Parameters

To find out how important each transcoding parameter is to the overall video quality, I tried to remove one transcoding parameter each time from the VTOM, and then I calculated the overall PCC and RMSE again. I started by removing the frame rate, this decreased the PCC by 0.3% and increased the RMSE by 16.9%. Removing the frame size decreased the PCC by 0.8% and decreased the RMSE by 1.5%. Removing the bit rate decreased the PCC by 0.4% and decreased the RMSE by 1.7%. Finally, removing the codec increased the PCC by 0.7% and increased the RMSE by 1.7%. I concluded from these results that all these transcoding parameters are important factors to the VTOM.

6.8.3. Transcoding

Here are some observations that are related to video encoding and transcoding:

- The H.264 and MPEG-4 codecs are unable to encode the vqeghd1_src01 original, raw, uncompressed video at lower than 6 Mbps. I believe that this is because of the video content type and motion level.
- The H.264 and MPEG-4 codecs are unable to encode the vqeghd1_src05 original, raw, uncompressed video at higher than 10 Mbps. I believe that this is because of the video content type, which is animation.
- The MPEG-4 codec is unable to reduce the frame rate from 29.97 fps to less than 15 fps for the videos generated from vqeghd1_src01 and vqeghd1_src03 when the bit rate is more than 15Mbps.
- When reducing the frame rate, the H.264 codec reduced the length of the videos more than the MPEG-4 and FLV codecs. Generally, the H.264 codec reduced the length from 10 seconds to around 6-7 seconds, MPEG-4 reduced the length from 10 seconds to 9 seconds, and FLV kept the length the same, at 10 seconds.

CHAPTER 7

QOS-AWARE VIDEO TRANSCODING SELECTION ALGORITHMS

Independent of any selection or composition algorithms, each transcoding function requires evaluation. Previous research uses statistical methods to evaluate these functions based only on the characteristics of the output video generated from each transcoding function [22]. Theses statistical methods use the standard-deviation normalization, which transforms data in a more efficient way than decimal normalization, using the mean and standard deviation [27]. This is done by picking up the most balanced function instead of one that is strong in one characteristic, e.g., delay, and weak in another, e.g., frame rate, [27]. To understand these statistical methods, this chapter provides a detailed description along with an example on how they work. It also presents my adaptation to generate four different QoS-aware video transcoding selection algorithms. Each of these algorithms selects the best transcoding function that meet the end-users requirements, i.e., required format, and comes as close as possible to satisfy their preferences, i.e., desired QoS. These algorithms statistically evaluate each transcoding function using the standard-deviation normalization introduced above. I evaluated these algorithms in terms of time complexity, user satisfaction rate, success ratio, recall, and precision. The evaluation results show the effectiveness and efficiency of these algorithms.

7.1. Introduction

When an end-user wants to watch a cloud-based video through a viewer (video playback software), the viewer needs to request a stream of that video. As mentioned in Chapter 3, this request includes a) the required video format, and b) a desired QoS that specifies a hoped-for video quality and a tolerable delay. As mentioned before, the desired QoS includes frame size, frame rate, bit rate, but when dealing with streaming, a desired QoS may also include a constraint on the amount of delay that the viewer is willing to tolerate in the stream. Even though the delay does not directly characterize a video's quality, it does characterize a video stream and it affects the end-user's experience. So, in these selection algorithms, I limit the desired QoS to be the bit rate, frame rate, frame size, i.e., width, height, and aspect ratio, and average transcoding delay.

Before presenting these four selection algorithms, this chapter first provides formal definitions for the function fitness, best fit, and weighted set used in these selection algorithms. These definitions are based on the definitions mentioned before. Second, it gives an overview of state-of-the-art research that is related to the QoS-aware video transcoding selection process. Third, it explains how I adapted these algorithms from other domains as candidate solutions to the "best-fit" transcoding function selection problem. Finally, it presents a general model for video delivery system independent of any particular transcoding function selection algorithm. I used this model as a framework to evaluate the four selection algorithms.

7.2. Definitions

Definition 7.1 (**Transcoding Function Fitness**): Informally, the transcoding function fitness is a value that represents how much degradation in video quality the function can cause. Formally, the transcoding function fitness is defined as follows:

$$(v, Q, t) = \begin{cases} \gamma(Q, t), & if\ v.f = t.f_{in}\ and\ Q.f = t.f_{out} \\ , & otherwise \end{cases} \quad (7.1)$$

where v is an input video, Q is a viewer request, and t is a transcoding function. is a function that computes a fitness value. Section 7.5 introduces the four candidate algorithms for calculating the value of the function . The value of $\gamma(Q, t)$ is a non-negative real number that combines the fitness with respect to video degradation and transcoding delay, the lower the value, the better it is.

Definition 7.2 (**Quality of Service**): A QoS, denoted in formulas as qos, is a specification consisting of frame size ($qos.fs$), frame rate ($qos.fr$), bit rate ($qos.br$) and a tolerated delay ($qos.d$). $v.c$ matches a qos, written as $v.c \approx qos$, if and only if $v.c.fs = qos.fs$, $v.c.fr = qos.fr$, and $v.c.br = qos.br$. As with video characteristics, we can further describe the desired frame size in a QoS in terms of width, $qos.fs.w$, height, $qos.fs.h$, and aspect ratio, $qos.fs.ar$. Generally, each specification $q_i \in qos$ is represented as $qos.q_i$

Definition 7.3 ("**Best-fit**"): The "best-fit" means the closest choice from a set of available ones based on specific criteria. Selecting the "best-fit" transcoding function, from a set of compatible transcoders, means selecting a transcoder with a transcoding function, based on the viewer's request and the requested video, such that $t(v, Q, t)$ has the lowest value. Formally, I can define the "best-fit", $Best(v, Q, T)$, as follows:

$$Best(v, Q, T): \forall t_i \forall t_j \; t_i \in T \wedge t(v, Q, t_i) \leq t(v, Q, t_j) \quad (7.2)$$

where v is an input video file, Q is a viewer request, and T is a transcoder.

Definition 7.4 (**The Weighted Set**): $WS = \{w_1, w_2, \dots, w_m\}$, where w_i is the viewer request's weight value for the video characteristic $qos.q_i$, where $(1 \leq i \leq m)$. The sum of these weights should be one. Each element in this set indicates how important its corresponding QoS property is to the viewer. For example, Table 7.1 shows a sample of the WS for a viewer request.

7.3. Related Work

Although selecting a "best-fit" transcoder has been an open problem to date, there are similar selection problems in other domains, like web services, that have been heavily investigated [44]. The problem of selecting the most appropriate web services from a pool of available ones that best match the end-user requirements and preferences has received a considerable attention in recent years [44]. The problem of web service selection shares many of the same concerns found in the multimedia service selection. However, it is not easy to directly apply web service selection approaches to the multimedia domain for two reasons: a) the rich semantic and complex internal structure of multimedia content itself, which is a combination of different forms, e.g., video, audio, or images, makes the process of storing, transcoding, transporting, and receiving them expensive; and b) the dynamic characteristics of multimedia

Table 7.1. An example of the viewer's weighted set.

QoS Properties	Bit rate	Frame rate	width	height	delay	aspect ratio
Weight	0.1	0.6	0.1	0.1	0.05	0.05

applications, such as the continuous flow of multimedia streams, makes the real-time processing requirement difficult [22].

Optimization algorithms like linear programming, dynamic programming, and Dijkstra-based algorithms have been proposed as solutions to the web service selection problem [22]. Yan Gao et al. [45] applied dynamic programming to solve the web service selection problem based on interface matching and to dynamically select the optimum Web services for composite services. However, their approach has some limitations, such as the complexity of runtime decisions. Rathore and Suman [46] proposed a Local Selection and Local Optimization (LSLO) approach based on linear programming for optimal candidate service selection for composition. In spite of the advantages of their approach, there are also some limitations. For example, they considered only the positive QoS properties. Avoiding negative QoS properties may result in inappropriate selections that might violate end-user expectations.

Hao Gao et al. [47] defined a novel similarity measure between user request and available services for web service selection. They calculated a similarity score for generating the ranking list. In addition, Hao Gao et al. [47] proposed a flexible matchmaking approach that enables the user to present the negotiable preference. After identifying the candidate services by using the matchmaking approach, i.e., either matching or not, they selected the service that has the maximum similarity value, using both categorical and numerical values. They provided a case study to illustrate their approach, but had insufficient data to evaluate the effectiveness and efficiency. In addition, they did not analyze the time complexity for their proposed selection approach.

For cloud-based service selection, Len et al. [48] divided selection techniques into three types, one of which is multi-criteria decision making technique, which is relevant to a transcoding function selection. Also, they considered cloud services as similar to web services, but in a different category. In addition, they described the differences between web services and cloud-based services. Building on their definitions, I considered video transcoding functions as cloud-based services and consider the transcoding function selection problem as a multi-criteria decision-making problem.

The cosine similarity and Euclidian distance are successfully used for such problems in information retrieval and data mining research [49] [50]. In addition, using similarity measures for service

selection, based on explicitly stated preferences, is a useful approach in service selection applications [51]. Therefore, for these reasons, I decided to adopt them in the transcoding function selection problem domain.

7.4. A General Model for a Cloud-based Video Distribution System

Figure 7.1 shows a general model for a cloud-based VDS, consisting of the following three main sub-systems: a) a cloud-based service management system, b) a cloud-based video transcoding system, and c) a cloud-based video streaming system. Here I will focus on the service management system, specifically, on the service selection process.

In this model, all available transcoders and their transcoding functions, which are provided by the service providers, are captured in a Service Registry. Step 1, which only needs to occur once after the Service Registry is loaded, normalizes $t.qos$ for every transcoding function t in every transcoder T in the Service Registry, using the standard-deviation normalization, $norm(qos.q_i)$, as in [27] as follows:

$$norm(qos.q_i) = \begin{cases} 2, & if\ (qos.q_i - \mu(qos.q_i)) > 2 * \delta(qos.q_i)) \\ 0, & if\ ((qos.q_i - \mu(qos.q_i)) < -2 * \delta(qos.q_i)) \\ \left(\frac{qos.q_i - \mu(qos.q_i)}{2 * \delta(qos.q_i)}\right) + 1, & otherwise \end{cases} \quad (7.3)$$

where $qos.q_i$ is the value of the QoS property i where $q_i \in qos$. The QoS properties are $qos.fs.w$,

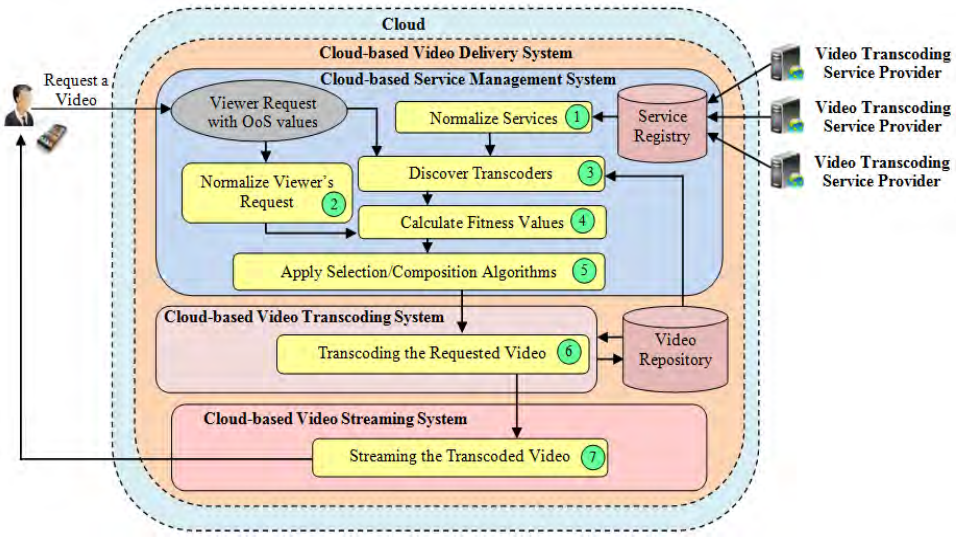


Figure 7.1. A general model for a cloud-based VDS.

$qos.fs.h$, $qos.fs.ar$, $qos.fr$, $qos.br$, and $qos.d$. The mean and standard deviation of the QoS property i is represented as $\mu(qos.q_i)$ and $\delta(qos.q_i)$, respectively.

I calculated the mean value for each QoS property i for each video transcoder T that contains m transcoding functions as follows:

$$\mu(qos.q_i) = \frac{1}{m} * \sum_{j=1}^{m}(t_j.qos.q_i) \quad (7.4)$$

Also, I calculated the standard deviation value for each QoS property i for each video transcoder T that contains m transcoding functions as follows:

$$\delta(qos.q_i) = \sqrt{\frac{1}{m}\sum_{j=1}^{m}((t_j.qos.q_i) - \mu(T.c.q_i))^2} \quad (7.5)$$

For the QoS properties that are better with smaller values, e.g., delay, $norm(qos)$ is further transformed into $norm'(qos.q_i)$ as follows [27]:

$$norm'(qos.q_i) = 2 - norm(qos.q_i) \quad (7.6)$$

It is obvious that this step has been completed before the viewer submits a request, so I decoupled its computation time from the computation time of the four candidate algorithms.

The normalization step is required because each QoS property has a different unit. For example, the delay in milliseconds, while the bit rate in kilobits per second. In addition, some properties are better with smaller values while others are better with bigger values.

For the remainder of this chapter, I will use t' to represent a transcoding function t where $t.qos$ has been replaced with $norm(t.qos)$.

Every time a viewer submits a request, Q, for a specific video, v, the system needs to normalize $Q.qos$ using (7.3). Below I use Q' to be a revised request with $Q.qos$ replaced by $norm(Q.qos)$. This process is Step 2 in Figure 7.1.

Step 3 uses Q' to discover compatible transcoders among the transcoders that are available in the Service Registry, according to Definition 3.8. In other words, Step 3 finds a set of transcoders, $CT = \{T_i\}$, such that $comp(T_i, v, Q)$ is true. Service discovery is a very important step in the service management system. In this chapter, I decided to focus on the service selection process and keep the service discovery step as simple as possible. I used a simple searching mechanism to discover all the compatible transcoders

from the Registry. For example, when the end-user wants to watch a cloud-based video that is originally encoded using H.264 through his playback software that supports DivX format, the viewer needs to request a stream for that video in DivX format. Thus, I used all the transcoders that transcode from H.264 to DivX as compatible transcoders in the selection process.

In Step 4, one algorithm from the candidate algorithms is used to compute $ŧ(v, Q', t')$ for every t' in every $T_i \in CT$. Step 5 then selects the "best-fit" according to Definition 7.3. In other words, it chooses a t' with a $ŧ(v, Q', t')$ value that is less than or equal to any other values.

Once a transcoding function has been selected in Step 5, then the processes, represented by Steps 6 and 7, perform the actual transcoding and streaming of the transcoded video.

7.5. The Candidate Algorithms

Step 4 shown in Figure 7.1 represents the actual computation for the fitness value, γ. In this section, I will explain my adaptations to propose four different algorithms from related research areas that compute the fitness value, γ. Then in the next section I compare their effectiveness.

The candidate algorithms are the Normalized Similarity (NS) algorithm, Normalized Euclidian Distance (NED) algorithm, Weighted Normalized Similarity (WNS) algorithm, and Weighted Normalized Euclidian Distance (WNED) algorithm. I believe that using any of these algorithms would represent an advancement for the transcoding function selection problem.

7.5.1. Normalized Similarity (NS) Service Selection Algorithm

This algorithm is an adaptation of the cosine similarity measure [50]. As such, it finds the fitness between each normalized transcoding function, $t\,.qos$, and a normalized viewer request, $Q\,.qos$. Algorithm 7.1 describes the NS algorithm as follows: Given a set of compatible transcoding functions, $CT = \{t_1, t_2, \dots, t_n\}$, and a viewer request, $Q = \{f, QoS\}$, I can define the fitness, $\gamma(Q\,.qos, t_i.qos)$, between each normalized transcoding function, t_i, where $(1 \le i \le n)$ and Q in the NS algorithm using (7.7).

An Example: This example facilitates understanding the NS algorithm and illustrates the proposed cloud-based video distribution model shown above. It also describes the statistical method that uses the

standard-deviation normalization. Assume there are 10 compatible transcoding functions that convert from H.264 to WMV1 but with different QoS specifications, see Table 7.2. The FR, BR, FS.W, FS.H, and D are the frame rate, bit rate, frame width, frame height, and delay, respectively. I computed the last column AR as FS.W/FS.H. The rows labeled M and SD are the mean and standard deviation, respectively. The last row is the viewer request, $Q.qos$.

Algorithm 7.1: Normalized Similarity (NS) algorithm.

Input:

1) The normalized values for the QoS properties for all the transcoding functions that are available in the compatible transcoders set, i.e., $CT = T_i = \{t'_1, t'_2, \dots, t'_n\}$.
2) A viewer request, i.e., $Q = \{f, qos\}$.

Output: The "best-fit" transcoding function.

Step 1: Normalization

1) Normalize all the QoS properties for the viewer request, $Q.qos$, using (7.3) to generate $Q'.qos$.
2) For the QoS properties that are better with smaller values, use (7.6) for further normalization.

Step 2: Calculate the fitness value, $\gamma(Q'.qos, t'_i.qos)$, between each normalized transcoding function, $t_i.qos$, and the normalized viewer request, $Q.qos$, as follows:

$$\gamma(Q'.qos, t'_i.qos) = 1 - \frac{\sum_{s \in QoS}((t'_i.qos.q_s) \times (Q'.qos.q_s))}{\sqrt{\sum_{s \in QoS}(t'_i.qos.q_s)^2} \times \sqrt{\sum_{s \in QoS}(Q'.qos.q_s)^2}} \qquad (7.7)$$

where q_s represents the QoS value for its corresponding QoS property for the normalized transcoding function, t_i, and the normalized viewer request, $Q.qos$.

Step 3: Return the transcoding function that has the lowest fitness value.

Table 7.2. Video transcoding service functions with different QoS values.

t_i	BR (Kbps)	FR (fps)	FS.W (pixel)	FS.H (pixel)	D (ms)	AR
1	15.48	8	128	72	1.1	1.78
2	19.35	10	128	72	1.15	1.78
3	23.22	12	128	72	1.2	1.78
4	92.9	12	256	144	1.3	1.78
5	116.13	15	256	144	1.35	1.78
6	241.92	18	320	200	1.6	1.60
7	268.8	20	320	200	1.65	1.60
8	386.59	23.97	320	240	1.8	1.33
9*	**387.1**	**24**	**320**	**240**	**1.85**	**1.33**
10	483.36	29.97	320	240	2.35	1.33
M	203.49	17.29	249.6	162.40	1.54	1.61
SD	174.14	7.14	87.70	71.56	0.39	0.20
Q.qos	388	24	320	230	1.87	1.39

I calculated the average transcoding delay for each transcoding function by averaging the time (in milliseconds) needed to transcode one video frame for all video frames that are available in a certain video. For example, if I have a video stream that has n video frames, I calculated the average transcoding delay, $delay(t_k)$, for the transcoding function t_k as follows:

$$delay(t_k) = \frac{\sum_{i=1}^{n} vf_i}{n} \quad (7.8)$$

where vf_i is the time needed to transcode one video frame, i.

Table 7.3 shows the normalized values for the 10 transcoding functions. The last row represents the Q , which is the normalized value for the viewer request, $Q.qos$. These values were computed using (7.3). I used (7.6) to further normalize the delay, i.e., D column.

Using the NS algorithm, I calculated the fitness value between each transcoding function, $t\,.qos$, and $Q'.qos$. Table 7.4 shows the results. For example, consider the normalized values of the first transcoding function, $t_{\,1}$, and the viewer request, Q . To calculate the fitness value using the NS algorithm, I want to re-write (7.7) for simplicity as follows:

$\gamma(Q'.qos, t'_1.qos) = 1 - (\frac{A}{B \times C})$, where

$$A = (0.460 * 1.53) + (0.349 * 1.47) + (0.307 * 1.401) + (0.368 * 1.472) + (1.553 * 0.574) + (1.416 * 0.464) = 3.737$$

$$B = \sqrt{0.460^2 + 0.349^2 + 0.307^2 + 0.368^2 + 1.553^2 + 1.416^2} = 2.232$$

Table 7.3. The QoS values after normalization.

t'_i	N-BR	N-FR	N-W	N-H	N-D	N-AR
1	0.460	0.349	0.307	0.368	1.553	1.416
2	0.471	0.489	0.783	0.368	1.490	1.416
3	0.482	0.629	0.783	0.368	1.426	1.416
4	0.682	0.629	1.151	0.871	1.299	1.416
5	0.749	0.839	1.151	0.871	1.235	1.416
6	1.110	1.049	1.335	1.263	0.917	0.978
7	1.188	1.189	1.335	1.263	0.854	0.978
8	1.526	1.467	1.335	1.542	0.663	0.321
9*	**1.527**	**1.470**	**1.335**	**1.542**	**0.599**	**0.321**
10	1.804	1.888	1.335	1.542	0.000	0.321
$Q'.qos$	1.530	1.470	1.401	1.472	0.574	0.464

$$C = \sqrt{1.53^2 + 1.470^2 + 1.401^2 + 1.472^2 + 0.574^2 + 0.464^2} = 3.029$$

$$\gamma(Q'.qos, t'_1.qos) = 1 - \left(\frac{3.737}{2.232 * 3.029}\right) = 0.4472$$

I followed the same way to calculate the fitness values for the other transcoding functions. Using these fitness values present in Table 7.4, the "best-fit" transcoding function that has the minimum fitness value will be selected, which turns out to be t_9.

7.5.2. Normalized Euclidean Distance (NED) Service Selection Algorithm

This algorithm is an adaptation of the Euclidean distance [49]. As such, it finds the distance between each normalized transcoding function, $t\,.qos$, and a normalized viewer request, $Q\,.qos$. Algorithm 7.2 describes the NED algorithm as follows: Given a set of compatible transcoding functions, $CT = \{t_1, t_2, \ldots, t_n\}$, and a viewer request, $Q = \{f, QoS\}$. I can define the fitness, $\gamma(Q\,.qos, t_i.qos)$, between each normalized transcoding function, t_i, and the normalized request, Q , in the NED algorithm using (7.9).

7.5.3. Weighted Normalized Similarity (WNS) Service Selection Algorithm

This algorithm is an adaptation to the cosine similarity measure [50] with the values of the weighted set. As such, it finds the fitness between each normalized transcoding function, $t\,.qos$, and a normalized viewer's request, $Q\,.qos$. Algorithm 7.3 describes the WNS algorithm as follows: Given a set

Table 7.4. The fitness values between the normalized transcoding functions and the normalized viewer request.

t'_i	$\gamma(Q', t'_i)$
1	0.4472
2	0.3498
3	0.3223
4	0.197
5	0.1642
6	0.0421
7	0.0316
8	0.0021
9*	**0.0017**
10	0.0257

of compatible transcoding functions, $CT = \{t_1, t_2, \dots, t_n\}$, a viewer request, $Q = \{f, QoS\}$, and a weighted set, $WS = \{w_1, w_2, \dots, w_m\}$. I can define the fitness value, $\gamma(Q'.qos, t_i.qos)$, between each normalized transcoding function, t_i, and the normalized request, Q', based on the weighted set, WS, in the WNS algorithm using (7.11).

Algorithm 7.2: Normalized Euclidian Distance (NED) algorithm.

Input:

1) The normalized values for the QoS properties for all the transcoding functions that are available in the compatible transcoders set, i.e., $CT = T_i = \{t'_1, t'_2, \dots, t'_n\}$.
2) A viewer request, i.e., $Q = \{f, qos\}$.

Output: The "best-fit" transcoding function.

Step 1: Normalization

1) Normalize all the QoS properties for the viewer request, $Q.qos$, using (7.3) to generate $Q'.qos$.
2) For the QoS properties that are better with smaller values, use (7.6) for further normalization.

Step 2: Calculate the fitness value, $\gamma(Q'.qos, t'_i.qos)$, between each normalized transcoding function, $t_i.qos$, and the normalized viewer request, $Q.qos$, as follows:

$$\gamma(Q'.qos, t'_i.qos) = \sqrt{\sum_{s \in QoS}(t_i.qos.q_s - Q.Qos.q_s)^2} \quad (7.9)$$

where q_s represents the QoS value for its corresponding QoS property for the normalized transcoding function, $t_i.qos$, and the normalized viewer request, $Q.qos$.

Step 3: Return the video transcoding function that has the lowest fitness value.

Algorithm 7.3: Weighted Normalized Similarity (WNS) service selection algorithm.

Input:

1) The normalized values for the QoS properties for all the transcoding functions that are available in the compatible transcoders set, i.e., $CT = T_i = \{t'_1, t'_2, \dots, t'_n\}$.
2) A viewer request, i.e., $Q = \{f, qos\}$.
3) A set of weights for the QoS properties, i.e., WS, that satisfies the following:

$$\sum_{s=1}^{m} w_s = 1 \ (i.e., \ W_{br} + W_{fr} + W_d + W_w + W_h + W_{ar} = 1) \quad (7.10)$$

Output: The "best-fit" transcoding function.

Step 1: Normalization

a) Normalize all the QoS properties for the viewer request, $Q.qos$, using (7.3) to generate $Q'.qos$.
b) For the QoS properties that are better with smaller values, use (7.6) for further normalization.

Step 2: Calculate the fitness value, $\gamma(Q.qos, t_i.qos)$, between each normalized transcoding function, $t'_i.qos$, and the normalized viewer request, $Q'.qos$, as follows:

$$\gamma(Q'.qos, t'_i.qos) = 1 - \frac{\sum_{s \in QoS}((w_s \times t\prime_i.qos.q_s) \times (w_s \times Q'.qos.q_s))}{\sqrt{\sum_{s \in QoS}(w_s \times t\prime_i.qos.q_s)^2} \times \sqrt{\sum_{s \in QoS}(w_s \times Q'.qos.q_s)^2}} \quad (7.11)$$

where q_s represents the QoS value for its corresponding QoS property for the normalized transcoding function, $t'_i.qos$, and the normalized viewer request, $Q'.qos$. The w_s represents the set of weights in the WS.

Step 3: Return the transcoding function that has the lowest fitness value.

7.5.4. Weighted Normalized Euclidean Distance (WNED) Service Selection Algorithm

This algorithm is an adaptation to the Euclidean distance [49] with the values of the weighted set. As such, it finds the distance between each normalized transcoding function, $t'.qos$, and a normalized viewer request, $Q'.qos$. Algorithm 7.4 describes the WNED algorithm as follows: Given a set of compatible transcoding functions, $CT = \{t'_1, t'_2, \dots, t'_n\}$, a viewer request, $Q = \{f, QoS\}$, and a weighted set, $WS = \{w_1, w_2, \dots, w_m\}$. I can define the fitness value, $\gamma(Q'.qos, t'_i.qos)$, between each normalized transcoding function, t'_i and the normalized viewer request, Q , based on the weighted set WS in the WNED algorithm using (7.12).

Algorithm 7.4: Weighted Normalized Euclidian Distance (WNED) algorithm.

Input:

1) The normalized values for the QoS properties for all the transcoding functions that are available in the compatible transcoders set, i.e., $CT = T_i = \{t'_1, t'_2, \dots, t'_n\}$.
2) A viewer request, i.e., $Q = \{f, qos\}$.
3) A set of weights, i.e., WS, for the QoS properties that satisfies (7.10).

Output: The "best-fit" transcoding function.

Step 1: Normalization

a) Normalize all the QoS properties for the viewer request, $Q.qos$, using (7.3) to generate $Q'.qos$.
b) For the QoS properties that are better with smaller values, use (7.6) for further normalization.

Step 2: Calculate the fitness value, $\gamma(Q.qos, t_i.qos)$, between each normalized transcoding function, $t'_i.qos$, and the normalized viewer request, $Q'.qos$, as follows:

$$\gamma(Q.qos, t_i.qos) = \sqrt{\sum_{s \in QoS}((w_s \times t_i.qos.q_s) - (w_s \times Q'.Qos.q_s))^2} \quad (7.12)$$

where q_s represents the QoS value for its corresponding QoS property for the normalized transcoding function, $t'_i.qos$, and the normalized viewer request, $Q'.qos$. The w_s represents the set weights in the WS set.

Step 3: Return the transcoding function that has the lowest fitness value.

7.6. Evaluation and Discussion

This section presents the evaluation of these four candidate algorithms in terms of success ratio. Then, it analyzes the time complexity for each of them. After that, it describes in detail calculating the user satisfaction rate. Finally, it depicts the further evaluation in terms of recall and precision along with a discussion regarding limitations and some possible extensions of this work.

7.6.1. Evaluation Setup

For evaluation, I created 472 transcoding functions that convert from H.264 to WMV1, but with different QoS specifications. These specifications cover most of the common frame sizes, frame rates and bit rates. I also generated five different viewer requests and evaluated the QoS assurance in terms of success percentage and user satisfaction, based on the above five different queries.

7.6.2. Success Percentage

I calculated the success percentage as follows:

$$SR = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases} \quad (7.13)$$

where SR is the success percentage, which is either 0 or 1. It is 1 if the retrieved transcoding function x from the candidate algorithm belongs to the class A, or 0 if it does not belong to a class A. Class A is a predefined class, which contains the most relevant transcoding functions that are close enough to the viewer request based on the expert judgment. This was done by having an expert look at the above 472 transcoding functions, and determines the acceptable transcoding functions for each viewer request and then add them to the class A.

As a result, the success ratio of these candidate algorithms, based on the above five queries that I created, is 1, which means that all these algorithms achieved high success ratio results.

7.6.3. Complexity Analysis

Let N denote the total number of transcoders that are available in the Service Registry and let M be the maximum number of transcoding functions that satisfy the hard constraints, i.e., the total number of transcoding functions that belong to the group of compatible transcoders. Based on Figure 7.1, in Step 1, normalizing the QoS properties for all the available transcoders using (7.3) takes $O(N)$. As I described earlier, I decoupled the computation time for this step from the candidate algorithms' computation time. Therefore, I will not consider this time in the candidate algorithm's overall computation time. In Step 2, normalizing the viewer request using (7.3) takes $O(1)$. In Step 3, discovering the compatible transcoders requires $O(log(N))$. In Step 4, For NS and NED algorithms, calculating the fitness between each

transcoding function and the viewer's request using (7.7) or (7.9) takes $O(M)$. Therefore, based on this analysis, the time complexity for the NS and NED algorithms is $O(log(N) + M + 1) = O(log(N) + M)$, which means that the service selection using the NS or NED algorithms is done in linear time.

For the WNS and WNED algorithms, calculating the fitness value between each transcoding function and the viewer's request in Step 4 using (7.11) or (7.12) takes $O(M)$. Therefore, based on this analysis, the time complexity for the WNS and WNED algorithms is $O(log(N) + M + 1) = O(log(N) + M)$, which means that the service selection using the WNS or WNED algorithms is also done in linear time.

7.6.4. User Satisfaction Rate

This experiment focuses on evaluating the candidate algorithms in terms of user satisfaction rate, which can be defined as follows:

$$\mathrm{S_r} = \frac{\sigma}{\omega} * 100\% \quad (7.14)$$

where S_r represents the user satisfaction rate, σ represents the total number of QoS criteria that are completely satisfied, i.e., exact values, and ω represents the total number of QoS criteria that are available, i.e., the total number of QoS criteria is 6. As a result, the user satisfaction rate for all the candidate algorithms is 100%. These results show that these algorithms have high user satisfaction rates based on the five different viewer requests mentioned above.

7.6.5. Further Evaluation

Because the four candidate algorithms achieved the same results in terms of success ratio and user satisfaction rate, I performed further experiments to evaluate the performance of these candidate algorithms in terms of classification. In other words, how the candidate algorithms are strongly classify the transcoding functions. I calculated the QoS assurance in terms of average recall and average precision based on the above five queries as follows:

$$recall = \frac{\varphi}{\tilde{\jmath}} \quad (7.15)$$

$$precision = \frac{\varphi}{\mathrm{Y}} \quad (7.16)$$

where φ is the total number of transcoding functions retrieved from the proposed algorithms and in a class A, based on a specific threshold. $\Im$ is the total number of transcoding functions that are in the class A, based on expert judgment, and Y represents the total number of transcoding functions that are retrieved. Class A is a predefined class, which contains the most relevant services that are close enough to the viewer request.

Figure 7.2 shows the evaluation results of these candidate algorithms based on the above five queries that I created. I measured these candidate algorithms in terms of how they successfully retrieve the transcoding functions that are belong to the class A. I also established a reference set of "best-fit" choices by having an expert look at the above 472 transcoding functions and determine the acceptable transcoding functions for each viewer's request, i.e., the transcoding functions that belong to the class A. It is obvious that the WNED algorithm achieves the best result in terms of average recall (85%), the NED algorithm achieves the best result in terms of average precision (69%). For the WNS and WNED algorithms, I choose equal weights that satisfy (7.10) for all the QoS properties. In case of different weights, as shown in Table 7.1, the returned result have in average a good recall (80%) and a low precision (14.6%). Since recall is more important for this problem, I conclude that the WNED algorithm is the best algorithm from among these four.

In case of different weight values, i.e., WS set, and based on the discussion above, I got low

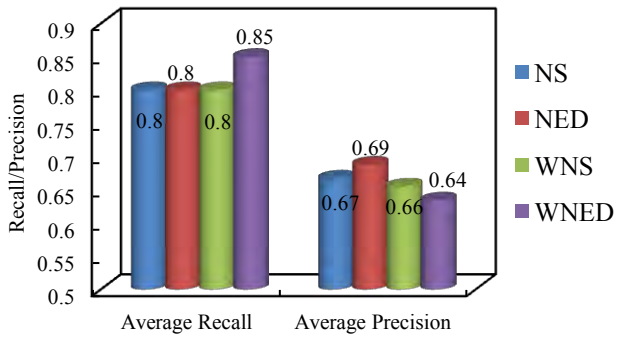


Figure 7.2. Evaluation results of the four candidate algorithms in terms of average recall and average precision.

precision values after applying the WNS and WNED algorithms. For this, I tried to find the best set of weight values that is a combination of different weight values for all QoS properties, which returns the highest recall and precision. For WNED, I started by assigning an equal weight for all the QoS properties as a base state. Then, I incremented the weight of one of the QoS properties and decremented the weight of the others (this set of weights should satisfy (7.10)), to generate a new state that might be a better state than the base state. After certain steps, sometimes the results I got represent a better state than the base case and sometimes a worse. I followed the induction principle [52] to prove that the new state is either a better or a worse than the base state.

For the bit rate test, I started by assigning an equal weight, i.e., 0.1666, set A, for all the QoS properties. After few increment and decrement steps, I got the highest recall value, i.e., 100%, and the highest precision value, i.e., 40%, when the weight value for the bit rate is 0.47, i.e., set B, and the weight value for all the other QoS properties is 0.106, i.e., set B. Therefore, I can prove by induction that this new state is better than the base state, i.e., set B is better than set A.

For the aspect ratio test, I started by assigning an equal weight, i.e., 0.1666, set A, for all the QoS properties. After few increment and decrement steps, I got the highest recall value, i.e., 100%, and the highest precision value, i.e., 40%, when the weight value for the aspect ratio is 0.37, i.e., set B, and the weight value for all the other QoS properties is 0.126, i.e., set B. Therefore, I can prove by induction that this new state is better than the base case, i.e., set B is better than set A.

For the width and height tests, I started by assigning an equal weight, i.e., 0.1666, set A, for all the QoS properties. After few increment and decrement steps, I got the highest recall value, i.e., 100%, and the highest precision value, i.e., 30.8%, when the weight value for the width and height is 0.19, i.e., set B, and the weight value for all the other QoS properties is 0.155, i.e., set B. Therefore, I can prove by induction that this new state is better than the base case, i.e., set B is better than set A.

For the frame rate test, I started by assigning an equal weight, i.e., 0.1666, set A, for all the QoS properties. After few increment and decrement steps, I got the highest recall value, i.e., 100%, and the highest precision value, i.e., 26.7%, when the weight value for the frame rate is 0.22, i.e., set B, and the

weight value for all the other QoS properties is 0.156, i.e., set B. Therefore, I can prove by induction that this new state is better than the base case, i.e., set B is better than set A.

For the delay test, I followed similar steps as described above. Unfortunately, I found that all the new states are worse than the base case, i.e., the base case is when all the weights are the same and it is better than all the new states.

There are two limitations associated with these selection algorithms. Firstly, the WNS and WNED algorithms depend on the viewer's assigned weight, i.e., WS, which is sometimes difficult to successfully assign them. Secondly, the successful use of these candidate algorithms depends on a large number of transcoding functions that cover most or all of the QoS properties. Studying these two limitations will be a future work.

CHAPTER 8

SUMMARY, CONCLUSION, AND FUTURE WORK

Guaranteeing a desired quality of the perceived, i.e., transcoded, video requires selecting or composing a set of transcoding functions that satisfy the requested viewing format and desired QoS. An effective way to accomplish this is by allowing the selection and composition mechanisms to take place based on a model that assesses the impact of video transcoding on video quality. This dissertation took the necessary steps to introduce, present, develop, and substantiate a general parametric model, called VTOM, that provides an extensible transcoding selection mechanism that takes into account the format and characteristics of the input video, the format required by the video playback software, and the desired QoS.

This selection mechanism uses VTOM for assessing the impact of video transcoding on objective video quality. As criteria for selection and quality evaluation, VTOM uses a set of media-related transcoding parameters including codec, bit rate, frame rate, and frame size. This research used the following three full-reference objective metrics: PSNR [11], SSIM [12], and MS-SSIM [9], to measure the quality of the transcoded videos given an original video and to develop, test, and evaluate the VTOM.

VTOM includes four quality sub-models, each describing the impact of one of these parameters on objective video quality. VTOM combines these quality sub-models with four additional error sub-models that I developed in a single weighted product aggregation function for assessing the overall quality of the transcoded video that is generated from a single transcoding function based on the above transcoding parameters. VTOM typically presents a mathematical function that estimates the perceived video quality based on different parameters. Therefore, the quality estimation is computed as the result of a direct mathematical function.

To develop the four quality sub-models, I used the results from exploring the impact of changing codec, and reducing each of bit rate, frame rate, and frame size, which include more than 2310 videos. In addition, I used the MS-SSIM [9] metric to generate the reference results. To test these quality sub-models, I used, as a testing set, four original, raw, uncompressed videos and encoded them at different bit rates using three different video codecs. Then I transcoded these encoded videos to generate more than 1510

transcoded videos. To evaluate these quality sub-models, I used the PCC and RMSE metrics that are widely used to evaluate image and video quality models. The conducted experiments showed that all of these sub-models achieved high PCC values with low RMSE values.

To develop the four error sub-models, I used the results of the videos that are available in the above testing set as training and testing sets. I used these error sub-models to calculate the weight for each of the above quality sub-models in the overall video quality. The evaluation results showed high PCC with very low RMSE values between the predicted and reference error values. The predicted error values are generated from the error sub-models. The reference error values represent the average of the absolute difference between the actual quality values generated from MS-SSIM and the predicted quality values generated from the quality sub-models at a given bit rate. In addition, I performed another experiment to calculate the weight based on normalizing the error, showing that normalizing the error generated results that decreased the PCC values and increased the RMSE values of the VTOM.

To evaluate the VTOM, I compared the predicted video quality results that are generated from VTOM with quality values generated by using MS-SSIM [9]. These extensive comparisons yielded results that showed high PCC values, with low RMSE values.

In the future, I plan to add video content type and motion level to the parameters that characterize the quality. Estimating the motion level could be done by calculating the average value of the Sum of Absolute Differences (SAD) that estimates the motion level in each video. Also, it has been shown that video content affects the perceived video quality [8].

Other factors that were not evaluated may also affect the perceived video quality, such as the available bandwidth (causing re-buffering), GOP size and structure, packet loss and concealment strategy, video codec-specific configurations, delay and jitter, video filters at receiver, and display type.

As a consequence of VTOM, the researchers and VDS developers have a model to calculate the degradation that each transcoding function can cause on the fly rather than evaluate it statistically. This can lead us to conclude that VTOM advances the field of video transcoding and video quality models and can improve transcoding service selection and composition algorithms that guarantee the required format and QoS, while optimizing transmission.

In addition, VTOM paves the road for researchers to develop and improve video transcoding selection and composition algorithms that might generate more accurate results. Also, it opens opportunities for researchers to propose extensions to VTOM or similar models that consider additional video characteristics and end-user preferences or expanding their ranges, which could prove to be very beneficial to a wide range of video-based systems and applications. Moreover, the substantiated results from this research lead us to believe that further subjective studies would be of great interest and value and represent a future direction of this research. VTOM also provides the foundation that can be extended, in a future work, to help in composing a set of selected transcoding functions and thereby assessing the quality of the transcoded video that is generated from composing these functions.

There exist likely benefits of the VTOM in different areas, such as cloud-based video delivery systems and applications, video conferencing, video surveillance systems, adaptive video transcoding and streaming, live streaming, smart phone content adaptation, Multimedia Message Service (MMS), home theater PC, on-demand video transcoding applications, and real-time video monitoring systems.

As part of this research and because no objective metric exists in the literature that evaluates the relative quality of two videos that have different frame rates, it was necessary to develop such a metric, called Frame Rate Metric (FRM). FRM uses the LCM and any frame-based quality metric to calculate the weighted average of comparing frames from the original and transcoded videos. I believe that the proposed metric represents an initial step toward developing a more robust one. However, it needs more evaluation using subjective tests and I left this for a future work.

To find out how important each transcoding parameter is to the overall video quality, I tried to remove one parameter each time from VTOM, and then re-calculate the overall PCC and RMSE values again. I found that all of them are important to the overall quality. In addition, I found that the frame size has the highest impact on the PCC, while the frame rate has the highest impact on the RMSE.

As a technical contribution, this dissertation provides an implementation of 780 transcoding functions that handle different combinations of different values of transcoding parameters. This implementation is based on Java® SDK 1.8 and Xuggler© 4.5. I plan to make this implementation and the video database that is generated from this research publicly available to the research community to

facilitate comparative evaluations of newer objective models and advance the state-of-the-art research in perceptual video quality systems.

In addition, this research presents a general model for a cloud-based video distribution system [14]. This model contains three sub-systems and has the following five benefits: a) it allows handling different distributions of transcoding functions and different end-user requests in the cloud; b) it helps in developing and evaluating new selection and composition algorithms; c) it provides a clear separation between video content and transcoding functions; d) it generates more powerful applications with more complex functionalities; and e) it allows video transcoding providers to add more transcoding functions to the cloud and video transcoding consumers that are distributed in different sites to consume these functionalities.

This research also presents a transcoding selection mechanism, which introduced four candidate algorithms: NS, NED, WNS, and WNED [13]. I showed that they are $O(log(N) + M)$, where N is the total number of choices and M is the maximum number of choices that satisfy the hard constraints. Then I conducted an experiment using a database that contains 472 transcoding functions, sample viewer requests, and expert opinions about the "best-fit" choice for each request. I showed that all of the above algorithms achieved high results in terms of success ratio and user satisfaction rate. In addition, I performed further experiments to evaluate these algorithms in terms of classification. I showed that the WNED algorithm achieved the highest result in terms of recall with a value of 85%, while the NED achieved the highest result in terms of precision with a value of 69%.

Possible future directions from this research would be composing the selected transcoding functions to satisfy complex viewer requirements and preferences. In addition, it would be possible in the future to build a complete cloud-based video delivery system, which includes both video transcoding and streaming sub-systems, based on the viewer requirements and preferences. Moreover, I could do a more thorough evaluation with a test bed constructed using input from multiple experts for the QoS-aware video transcoding service selection process.

REFERENCES

[1] D. Evans, "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything," Cisco Internet Business Solutions Group (IBSG), April 2011.

[2] Cisco, "Cisco Visual Networking Index: Forecast and Methodology, 2013–2018," 10 June 2014. [Online]. Available: http://www.cisco.com. [Accessed 11 January 2015].

[3] Wikipedia, "Video Codec," 1 July 2014. [Online]. Available: http://en.wikipedia.org/wiki/Video_codec. [Accessed 11 January 2015].

[4] J. Cabasso, "Determining Video Quality," November 2008. [Online]. Available: http://www.aventuracctv.com/PDF/ATI_Video_Quality.pdf. [Accessed 24 January 2015].

[5] Wikipedia, "Video Quality," 9 March 2015. [Online]. Available: http://en.wikipedia.org/wiki/Video_quality. [Accessed 21 March 2015].

[6] Wikipedia, "Mean Squared Error," 16 March 2016. [Online]. Available: https://en.wikipedia.org/wiki/Mean_squared_error. [Accessed 24 March 2016].

[7] Wikipedia, "Transcoding," 6 January 2015. [Online]. Available: http://en.wikipedia.org/wiki/Transcoding. [Accessed 11 January 2015].

[8] J. Joskowicz, R. Sotelo and J. C. L. Ardao, "Towards a General Parametric Model for Perceptual Video Quality Estimation," *IEEE Trans. Broadcast.,* vol. 59, no. 4, pp. 569-579, December 2013.

[9] Z. Wang, E. P. Simoncelli and A. C. Bovik, "Multi-scale Structural Similarity for Image Quality Assessment," in *Proc. 37th IEEE Asilomar Conf. on Signals, Systems and Computers*, Pacific Grove, CA, November 9-12, 2003.

[10] VQEG Group, "Report on the Validation of Video Quality Models for High Definition Video Content," 2010.

[11] Wikipedia, "Peak signal-to-noise ratio," 14 December 2015. [Online]. Available: https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio. [Accessed 23 March 2016].

[12] Z. Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Trans. Image Process.,* vol. 13, no. 4, pp. 600 - 612, April 2004.

[13] N. O. Alsrehin and S. W. Clyde, "QoS-Aware Video Transcoding Service Selection Process," *Journal of Media & Mass Communication,* vol. 1, no. 2, pp. 61-68, December 2015.

[14] N. O. Alsrehin, "QoS-aware Video Transcoding Service Composition Process in a Distributed Cloud Environment," *Int. Journal of Innovative Research in Computerand Communication Engineering,* vol. 3, no. 5, pp. 3726-3748, May 2015.

[15] Xamarin, [Online]. Available: http://xamarin.com. [Accessed 24 January 2015].

[16] G. Coulouris, J. Dollimore, T. Kindberg and G. Blair, Distributed Systems: Concepts and Design, 5th ed., Pearson, 2011.

[17] Apple, [Online]. Available: http://www.apple.com/iphone-6s/. [Accessed 15 March 2015].

[18] M. S. Hossain, "Towards a Biological-inspired Framework for Multimedia Service Management," Ph.D. dissertation, Dept. Elect. and Comp. Eng., University of Ottawa, Ottawa, Canada, 2009.

[19] H. Sun, X. Chen and T. Chiang, Digital Video Transcoding for Transmission and Storage, CRC Press, 2005.

[20] F. Jokhio, T. Deneke, S. Lafond and J. Lilius, "Bit Rate Reduction Video Transcoding with Distributed Computing," in *Proc. 20th Euromicro Int. Conf. on Parallel, Distributed and Network-based Processing*, Germany, 2012.

[21] S. Moiron, M. Ghanbari and P. Assunção, "Video Transcoding Techniques," *Studies in computational intelligence,* vol. 231, pp. 245-270, 2009.

[22] M. S. Hossain and A. E. Saddik, "QoS Requirement in the Multimedia Transcoding Service Selection Process," *IEEE Trans. Instrum. Meas.,* vol. 59, no. 6, pp. 1498-1506, 2010.

[23] I. Ahmad, X. Wei, Y. Sun and Y.-Q. Zhang, "Video Transcoding: An Overview of Various Techniques and Research Issues," *IEEE Trans. Multimedia,* vol. 7, no. 5, pp. 793-804, October 2005.

[24] Y. Liang, F. Chebil and A. Islam, "Compressed Domain Transcoding Solutions for MPEG-4 Visual Simple Profile and H.263 Baseline Videos in 3GPP Services and Applications," *IEEE Trans. Consum. Electron.,* vol. 52, no. 2, pp. 507 - 514, 2006.

[25] A. Vetro, J. Xin and H. Sun, "Error-Resilience Video Transcoding for Wireless Communications," *IEEE Wireless Communications,* vol. 12, pp. 14-21, August 2005.

[26] W. D. J. C. Lianyong Qi, "Weighted principal component analysis-based service selection method for multimedia services in cloud," *Springer Computing,* vol. 98, no. 1, pp. 195-214, January 2016.

[27] J. L. a. V. Issarny, "QoS-aware Service Location in Mobile Ad-Hoc Networks," in *Proc. 5th Int. Conf. on Mobile Data Managment (MDM)*, 2004.

[28] X. Gu and K. Nahrstedt, "Distributed multimedia service composition with statistical QoS assurance," *IEEE Trans. Multimedia,* vol. 8, no. 1, pp. 141-151, 2006.

[29] J.-C. Moissinac, "Automatic Discovery and Composition of Multimedia Adaptation Services," in *Proc. the 4th Int. Conf. on Advances in Multimedia*, Chamonix, France, April 2012.

[30] W. Li, Y. Wang, C. Li, S. Lu and D. Chen, "A QoS-aware service selection algorithm for multimedia service overlay networks," in *Proc. the Int. Conf. on Parallel and Distributed Systems*, Hsinchu, 2007.

[31] M. S. Hossain, "QoS-aware service composition for distributed video surveillance," *Multimedia Tools and Applications,* vol. 73, no. 1, pp. 169-188, November 2014.

[32] A. J. Gonzalez, J. Alcober, R. M. d. Pozuelo, F. Pinyol and K. Z. Ghafoor, "Context-aware multimedia service composition using quality assessment," in *Proc. IEEE Int. Conf. on Multimedia and Expo.*

(ICME), Barcelona, Spain, July 2011.

[33] S. Chikkerur, V. Sundaram, M. Reisslein and L. J. Karam, "Objective Video Quality Assessment Methods: A Classification, Review, and Performance Comparison," *IEEE Trans. Broadcast.,* vol. 57, no. 2, pp. 165 - 182, February 2011.

[34] J. J. a. J. C. L. A. Ardao, "Combining the effects of frame rate, bit rate, display size and video content in a parametric video quality model," in *Proc. of the 6th Latin America Networking Conf.*, Quito, Ecuador, 2011.

[35] "Video Quality Metric (VQM) Software," Institute for Telecommunication Sciences, [Online]. Available: http://www.its.bldrdoc.gov/resources/video-quality-research/software.aspx. [Accessed 9 August 2015].

[36] Y.-F. Ou, W. Lin, H. Zeng and Y. Wang, "Perceptual Quality of Video with Periodic Frame Rate and Quantization Variation–Subjective Studies and Analytical Modeling," *Computing Research Repository (CoRR),* vol. 1, 2014.

[37] Y.-F. Ou, Z. Ma and Y. Wang, "A Novel Quality Metric for Compressed Video Considering both Frame Rate and Quantization Artifacts," in *Video Processing and Quality Metrics*, Scottsdale, AZ, 2009.

[38] Y.-F. Ou, T. Liu, Z. Zhao, Z. Ma and Y. Wang, "Modeling the Impact of Frame Rate on Peceptual Quality of Video," in *Proc. 15th IEEE Int. Conf. on Image Processing (ICIP)*, San Diego, CA, October, 2008.

[39] T. Zinner, O. Hohlfeld, O. Abboud and T. Hossfeld, "Impact of frame rate and resolution on objective QoE metrics," in *Proc. 2nd Int. Workshop on Quality of Multimedia Experience (QoMEX)*, Trondheim, Norway, June 2010.

[40] V. Samanta, R. Oliveira, A. Dixit, P. Aghera, P. Zerfos and S. Lu, "Impact of Video Encoding Parameters on Dynamic Video Transcoding," in *Proc. the 1st Int. Conf. on Communication Systems Software and Middleware*, New Delhi, India, January 2006.

[41] L. Goldmann, F. D. Simone, F. Dufaux, T. Ebrahimi, R. Tanner and M. Lattuada, "Impact of Video Transcoding Artifacts on the Subjective Quality," in *Proc. the 2nd Int. Workshop on Quality of Multimedia Experience (QoMEX)*, Trondheim, Norway, June 2010.

[42] U. H. Reimers, "DVB—The Family of International Standards for Digital Video Broadcasting," *Proc. IEEE,* vol. 94, no. 1, pp. 173-182, January 2006.

[43] Wikipedia, "Bicubic Interpolation," 19 May 2015. [Online]. Available: https://en.wikipedia.org/wiki/Bicubic_interpolation. [Accessed 8 August 2015].

[44] D. G. Joseph and M. Moghaddam, "Service Selection in Web Service Composition: A Comparative Review of Existing Approaches," in *Web Services Foundations*, Springer New York, 2014, pp. 321-346.

[45] Y. Gao, J. Na, B. Zhang, L. Yang and Q. Gong, "Optimal Web Services Selection Using Dynamic Programming," in *Proc. of the 11th IEEE Symp. on Computers and Communications (ISCC'06)*, 2006.

[46] R. Maya and S. Ugrasen, "Web Service Selection Algorithm for Dynamic Service Composition using LSLO Approach," in *Proc. Int. Conf. on Informatics, Electronics & Vision (ICIEV)*, Dhaka, Bangladesh, May 2013.

[47] H. Gao, J. Yan and Y. Mu, "Web Service Selection based on Similarity Evaluation," in *Proc. IEEE Int. Conf. on Services Computing*, Washington, D.C, 2011.

[48] L. Sun, H. iDong, F. h. Hussain, O. K. Hussain and E. Chang, "Cloud service selection: State-of-the-art and future research directions," *Journal of Network and Computer Applications,* vol. 45, pp. 134 - 150, October 2014.

[49] Wikipedia, "Euclidean Distance," 17 November 2014. [Online]. Available: http://en.wikipedia.org/wiki/Euclidean_distance#Squared_Euclidean_distance. [Accessed 6 11 2014].

[50] Wikipedia, "Cosine similarity," 17 January 2016. [Online]. Available: https://en.wikipedia.org/wiki/Cosine_similarity. [Accessed 4 March 2016].

[51] H. Wang, S. Shao, X. Zhou, C. Wan and A. Bouguettaya, "Web Service Selection with Incomplete or Inconsistent User Preferences," in *Service-Oriented Computing*, Springer Berlin Heidelberg, 2009, pp. 83-98.

[52] Wikipedia, "Mathematical Induction," 27 October 2014. [Online]. Available: http://en.wikipedia.org/wiki/Mathematical_induction. [Accessed 23 December 2014].

[53] BOSCH, "Bosch Security Systems Asia Pacific," BOSCH, [Online]. Available: http://www.bosch.com/en/com/home/index.php. [Accessed 2015].

[54] I. Shin and K. Koh, "Hybrid Transcoding for QoS Adaptive Video-on-Demand Services," *IEEE Trans. Consum. Electron.,* vol. 50, no. 2, pp. 732-736, 2004.

[55] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, "A View of Cloud Computing," *Communications of the ACM,* vol. 53, no. 4, pp. 50-58, April 2010.

[56] F. Jokhio, A. Ashraf, S. Lafond, I. Porres and J. Lilius, "Cost-Efficient Dynamically Scalable Video Transcoding in Cloud Computing," TUCS Technical Report, Finland, 2013.

[57] Wikipedia, "List of video transcoding software," 31 July 2014. [Online]. Available: https://en.wikipedia.org/wiki/List_of_video_transcoding_software. [Accessed 2015 October 2015].

[58] A. Maheshwari, A. Sharma, K. Ramamritham and P. Shenoy, "TranSquid: Transcoding and Caching Proxy for Heterogenous E-Commerce Environments," in *Proc. 12th IEEE Int. Workshop Research Issues in Data Engineering*, San Jose, California, USA, 26 Febraury to 1 March, 2002.

[59] Wikipedia, "H.264/MPEG-4 AVC," 4 October 2015. [Online]. Available: https://en.wikipedia.org/wiki/H.264/MPEG-4_AVC. [Accessed 12 October 2015].

[60] Wikipedia, "MPEG-4," 29 September 2015. [Online]. Available: https://en.wikipedia.org/wiki/MPEG-4. [Accessed 14 October 2015].

[61] R. Koenen, "MPEG-4: a Powerful Standard for Use in Web and Television Environments," [Online]. Available: http://www.w3.org/Architecture/1998/06/Workshop/paper26/. [Accessed 14 October

2015].

[62] Wikipedia, "Flash Video," 13 September 2015. [Online]. Available: https://en.wikipedia.org/wiki/Flash_Video. [Accessed 14 October 2015].

[63] Wikipedia, "Human Visual System Model," 7 January 2015. [Online]. Available: https://en.wikipedia.org/wiki/Human_visual_system_model. [Accessed 11 July 2015].

[64] K. Seshadrinathan, R. Soundararajan, A. C. Bovik and L. K. Cormack, "Study of Subjective and Objective Quality Assessment of Video," *IEEE Trans. Image Process.,* vol. 19, no. 6, pp. 1427 - 1441, February 2010.

[65] ITU-R Recommendation BT.500-11, "Methodology for the subjective assessment of the quality of of television pictures," International Telecommunication Union, 2012.

[66] ITU-T recommendation P.910 Tech., "Subjective video quality assessment methods for multimedia applications," Union, Internation Telecommunication, 2008.

[67] Recommendation ITU-R BT.500-11, "Methodology for the Subjective Assessment of the Quality of Television," International Telecommunication Union, June 2002.

[68] K. Seshadrinathan and A. C. Bovik, "Motion Tuned Spatio-Temporal Quality Assessment of Natural Videos," *IEEE Trans. Image Process.,* vol. 19, no. 2, pp. 335-350, February 2010.

APPENDICES

APPENDIX A

VIDEO TRANSCODING TYPES

This appendix provides more description about different types of video transcoding along with the video codecs that I used in this research. This classification follows similar style and structure that is described in [18] [19].

A.1. Video Transcoding Types Based on Content Variation

Based on the content variation, video transcoding is classified into the following approaches [18]:

A.1.1. Static Transcoding

In static transcoding, multiple versions of the same original video content are transcoded in advance and then stored in the server side. These versions are transcoded based on the common formats or codecs, network bandwidths, devices, and end-user requirements and preferences. When the end-user makes a video request with a specific format requirement and QoS preference values, the system will select the appropriate version among the existing transcoded ones, without any modification. The advantage of this approach is reducing the processing and the delivery time, so there is no need to wait for video transcoding to be completed on the original video, because the desired version is already available. On the other hand, there is a need for huge storage capabilities in order to store different versions of the same video content. In addition, whenever a new format, device, or display size appears on the market, a new version should be generated. Moreover, managing this huge amount of video content requires more resource management in terms of indexing, processing, accessing, retrieving, recovery, and storage [18].

A.1.2. Dynamic Transcoding

In static transcoding, a set of transcoded versions is generated and stored in advance, while in the dynamic transcoding, video transcoding occurs "on the fly"; upon the end-user's request [18]. When an end-user requests a video to be played on his device, that request is sent to the server, where upon the system interprets the request and extracts the required format and QoS, then it transcodes the original requested video "on the fly" to generate a new version that matches the end-user requirement and preferences, then

finally it sends the stream back to the end-user. As a result, the end-user receives a video that matches his requirements and preferences. The advantage of dynamic transcoding is that there is no need to store more than one version of each original video. However, a scalable system should be provided in order to handle huge end-user requests at the same time. Bosch [53] presented dynamic transcoding technology that provided immediate access to high-quality video content for the first time at “Security Essen 2012".

A.1.3. Hybrid Transcoding

In hybrid transcoding, a mix of static and dynamic transcoding is performed. This approach was initially introduced by Shin and Koh [54]. The static approach is used for frequently-accessed video content while the dynamic approach is used when the video content is accessed less frequently.

A.2. Video Transcoding Types Based on Architectures

Video transcoding can be done at any point between video content creation and video content consumption. This can take place at either the client's side, the server's side, or in the cloud. Video transcoding types can be classified based on this architecture into: a) client-based transcoding, b) server-based transcoding, or c) cloud-based transcoding. Each possesses strengths and weaknesses.

A.2.1. Client-based Video Transcoding

In the client-based video transcoding architecture, transcoding is done completely at the client's device or playback software. In this situation, the server delivers the entire requested video without any modification, while the user's device or playback software performs the real video transcoding based on its capabilities in terms of processing power, supporting format, network bandwidth, power-saving settings, displaying size, storage capacity, and other device-specific factors. Client-based transcoding has some advantages: a) avoiding transporting large amounts of device and user-profile information or metadata to the server for transcoding; b) tolerating failure when the server is unable to determine the device capabilities from the request; and c) moving the transcoding overhead from the server side. On the other hand, it has some weaknesses, such as: a) bottlenecking the network when requesting high-quality video content, e.g., UHD at 30 fps, to be played at low quality, e.g., QCIF frame size and at 15 fps; and b) the

transcoding speed and quality depend on the capabilities of the client's device. Opera software® is an example of client-based transcoding [18].

A.2.2. Server-Based Video Transcoding

In the server-based transcoding architecture, the transcoding is done completely at the server side. When the server receives the client's request, which contains information about the client's device (or playback software) and its capabilities in terms of media format, network bandwidth, screen size, and storage; the server performs the transcoding and generates new media content that matches the request. Both static (off-line) and dynamic ("on the fly") transcoding are supported by this architecture [18].

However, dynamic transcoding is not useful here because the transcoding is a computer-intensive operation and the end-user should wait until the transcoded content becomes ready, which depends on different client characteristics, different user preferences, and the communication protocol used in the content delivery [18]. Server-based transcoding provides different video formats and content for each original video using a static transcoding approach, and the video content that best matches the client's request is selected and sent to the client. IBM WebSphere® Transcoding Publisher, BEA Weblogic®, and AvantGo© are examples of server-based transcoding [18].

There are some advantages of using server-based transcoding, such as: a) the network is utilized with the video content that matches the client's device and network bandwidth; b) the server offers more processing power than the client devices; and c) the provided video content and quality are controlled by the content creators. On the other hand, the drawbacks of server-based video transcoding are a) heavy server-side applications may slow down the server; and b) the provided video contents and quality might not best match the client's preferences.

A.2.3. Cloud-based Transcoding

In the cloud-based video transcoding architecture, the transcoding is done in the cloud and "on the fly" based on the client's request. This is known as on-demand video transcoding. In this architecture, the client requests a video content based on his device and network capabilities, and then the original video content is transcoded "on the fly" using the cloud infrastructure capabilities and the transcoded video

content is sent back to the client. The transcoded content might be stored or cached for future use, which reduces the need for re-transcoding and avoids maintaining multiple variant copies of the same video content. This architecture supports both dynamic and hybrid video transcoding, addresses heterogeneity issues, and reduces overall end-to-end response time [18].

On-demand video transcoding is a computer-intensive operation. Therefore, transcoding a large number of on-demand videos requires a large number of cloud-based transcoding servers. Similarly, a large amount of disk space is required to store multiple transcoded versions for each source video. Infrastructure as a Service (IaaS), such as Amazon Elastic Computing Cloud® (EC2)[8] and Amazon Simple Storage Service® (S3)[9], provide computing and storage resources under the pay-per-use business model [55].

EC2 provides the computing resources through Virtual Machines (VM) by dynamically creating scalable clusters of servers, while S3 provides the storage resources. EC2 can be used to virtually create scalable clusters of video-transcoding servers that hold thousands of video-transcoding functions, and similarly S3 can be used to store both the original source video and the multiple transcoded versions. In a cloud environment, video transcoding can be performed in several different ways. For example, it is possible to map the entire video stream to a dedicated VM in order to transcode the entire stream, or split the video streams into smaller segments and independently transcode each of them in different VMs [56]. Cloud-based video transcoding has been well investigated and improved in the recent years.

Software as a Service (SaaS) is a model where the customers can access the services via the internet without paying attention to how these services are maintained. The service providers are responsible for maintaining these services. Amazon Elastic Transcoder®[10] is a video-transcoding service provider that provides video transcoding functionality in the cloud. There are over 30 such providers today [57]. Many of the available video-transcoding services provide the same functionality, i.e., format conversion, but with different QoS values.

Video transcoding is a computationally-intensive operation, and due to the weaknesses of the client- and server-based video transcoding architectures, it may not be suitable for performing the

[8] http://aws.amazon.com/ec2/
[9] http://aws.amazon.com/s3/
[10] https://aws.amazon.com/elastictranscoder/

transcoding at the client or the server sides. Therefore, the cloud-based video transcoding architecture is the best solution for the successful distribution of video content and the most commonly used these days [56].

In the cloud-based video transcoding approach, the client's request might not be satisfied using one transcoding service. Hence, two or more transcoding services are composed. For example, to transcode MPEG-2 to H.264 where there is no direct conversion from MPEG-2 to H.264, the original video could first be transcoded from MPEG-2 into one or more intermediate formats, e.g., to MPEG-3, before converting into the required format using one transcoding function, then the output could be transcoded using another transcoding function, to obtain the H.264 format. This composition approach helps in [22]:

- Satisfying more complicated transcoding requests even with a limited number of available transcoding functions.
- Generating a final transcoded content that would be exactly the same as if the transcoding had been performed in one step.
- Distributing the computations among different resources, which in turn generates more powerful applications with more complex functionalities, because the functionality offered by individual resources is limited.
- Cloud-based video transcoding approach has a number of advantages, such as [18] [58]:
- Providing a clear separation between video content creation and content transcoding.
- Allowing the computation to be distributed among different computation resources in the cloud.
- Improving the opportunity for composing multiple transcoding functions.
- Improving the client-perceived latency and reduces overall end-to-end response time.
- Scaling the transcoding properly with the number of clients.

A.3. Video Codecs

A video format is the structure of the video’s image and audio data. Some popular formats are H.264, MPEG-4 part 2, FLV, MJPEG (Motion JPEG), WMV, and DivX, but there are over 20 in common use today [3]. A video codec is a piece of software that can encode video data into a particular format and

decode a video from that format. A codec's name is often used as a label or synonym for the format it encodes and decodes.

The format of the compressed data usually follows some video-compression specifications or standards. If the compression is lossy, a lower quality than the original is generated as a consequence of this compression phase. However, the available video codecs have different quality loss. The next sub-sections will describe the ones that are used in this research.

A.3.1. H.264

H.264 or MPEG-4 Part 10, Advanced Video Coding (MPEG-4 AVC) [59] is currently one of the most commonly used video codecs for recording, compression, and transmission of video content [59]. H.264/AVC developed by the International Telecommunications Union, Video Coding Experts Group (VCEG), and Moving Picture Experts Group (MPEG). It is widely used in the HTML5 Internet standard. Microsoft® supports H.264 in Internet Explorer 9® browsers, and Google® announced that they will support it in Google Chrome™ browser [59].

H.264 uses the Real Time Transport Protocol (RTTP) as a transport protocol. It has a wide range of multimedia applications over heterogeneous network and it is suitable for high-quality video streaming at lower data rate, especially in the limited wireless communication environments and devices [18]. The standardization of the first version of H.264/AVC was completed and approved in 2003 [59]. H.264 compresses video much more efficiently than other video codecs and contains a number of new features that provide more flexibility for applications and a wide variety of network environments [59].

A.3.2. MPEG-4 Part 2

MPEG-4 Part 2, or MPEG-4 for convenience, was introduced in late 1998 and designated as a standard for a group of audio and video coding formats. The ISO/IEC Moving Picture Experts Group (MPEG) (ISO/IEC JTC1/SC29/ WG11) agreed upon MPEG-4 under the formal standard ISO/IEC 14496 – Coding of audio-visual objects [60].

MPEG-4 is used for AV compression for web and wireless media streaming, voice and television applications, and applications that facilitate integrating different media content coming from different

channels to the same multimedia scene, such as video conferencing. It preserves a compatibility with major existing video codecs, such as: MPEG-1, MPEG-2, and ITU-T H.263.

MPEG-4 adds more features, such as: a) supporting different types of media content (2D and 3D); b) coding at different levels of quality; c) adapting media content to different environments and bandwidths; and d) adding protection and intellectual property to the original media content [61].

A.3.3. FLV

FLV is a Flash Video [62], which represents a video codec and a container format. It is used to deliver video content over the Internet using Adobe Flash Player® version 6 and newer. It usually contains material encoded with codecs following the Sorenson Spark, Sorenson H.263, ITU-T H.263, or VP6 video compression formats [5]. Sorenson Spark is the first video codec supported in Flash Player. In this research, I used Sorenson H.263 as FLV video codec supported by Xuggler© and FFmpeg© with some flash extensions.

Flash Video is the standard for web-based video streaming over Real Time Messaging Protocol (RTMP). YouTube®, Hulu®, Yahoo!® and many other news providers support Flash Video. Most of the operating systems support Flash Video via the Adobe Flash Player® and Web browser plug-ins or via one of several third-party programs [5].

APPENDIX B

VIDEO QUALITY ASSESSMENTS

This appendix describes the video quality assessment techniques classified into objective and subjective [5]. In addition, it briefly describes the three objective quality assessment metrics that I used through this research.

B.1. Video Quality Assessment

Typically, digital videos pass through several processing phases before they reach to the end-users, who are most often human observers. These processing phases mostly degrade the quality of the video that pass through it. Although some of them (for example, in the end-user devices) attempt to improve the quality.

One of the fundamental video processing phases is the transcoding phase. This phase affects the quality of the perceived video and introduces degradation. This affect comes from the asymmetry between the values of the transcoding parameters at both the decoder and encoder sides.

Generally, evaluating video quality plays an important role in maintaining the QoS requirements, evaluating the performance of digital video processing and transmission systems, improving the configuration parameters for video compression, and finally designing optimal video management systems.

Humans can judge the quality of an image or video by using prior knowledge derived from viewing millions of time-varying daily basis images and videos [5]. Human Visual System (HVS) is used by image and video processing experts to deal with biological and psychological vision processes that are not yet fully understood [63]. However, it is used as the ultimate standard to evaluate and assess Image Quality Assessment (IQA) and Video Quality Assessment (VQA) algorithms that are developed and in progress. Algorithms and techniques for VQA can be classified into two types, objective and subjective [5].

B.2. Objective Quality Techniques

Objective evaluation techniques are mathematical models that approximate expert judgments. Since they are mathematical models, a computer program can automatically calculate them. The robustness

of any proposed objective quality assessment metric can be determined by testing it with respect to a variety of video content and metrics, and then measuring the degree of correlation and consistency with the subjective results by using the following two common metrics: PCC and RMSE [33]. These objective evaluation techniques are classified based on the amount of information available about the original and the distorted signals into the following:

B.2.1. Full Reference (FR) Methods

To compute the quality of the perceived video using the FR metrics, the original and the distorted, e.g., transcoded, video signals should be available. This calculation can be done by computing the difference between the original video signals against the distorted video signals. Typically, every pixel from the source is compared against the corresponding pixel at the distorted video, which leads to a frame-by-frame comparison.

In spite of the higher expenses of the computational effort using the FR metrics, they are usually the most accurate [5]. Developing a general full-reference VQA algorithm that works across a range of distortion types requires two assumptions a) the reference video is a perfect and distortion free, b) each test video is a distorted version from the reference one [5].

B.2.2. Reduced Reference (RR) Methods

The RR metrics are used when some reduced information about the original and distorted video signals is available. RR can be done by extracting some features from both videos and compare them to give a quality score. RR is more efficient than FR metrics due to the practicality of their assumptions [5]. I considered the VTOM as a RR model.

B.2.3. No-Reference (NR) Methods

When the original video signal is absent or it is impossible to practically provide it, the NR metrics try to assess the quality of a distorted video without any reference to the original signal. The NR metrics may be less accurate than FR or RR approaches, but are more efficient to compute [5].

B.3. Subjective Quality Techniques

Subjective evaluation techniques, on the other hand, require expert judgments. These techniques are the only reliable methods to evaluate video quality [64] by asking the human subjects for their opinion on the quality of a video that is perceived by HVS. For subjective video quality methods, there are a set of procedures that are described in ITU-R recommendation BT.500 [65] and ITU-T recommendation P.910 [66] that might vary depending on what kind of system that will be tested.

These subjective methods are crucial for evaluating the performance of objective methods. The results of the objective methods are usually compared for the degree of correlation and consistency with the results of the perceptual quality measures that are obtained from the subjective methods [33].

The main idea for the subjective methods is allowing a group of viewers to watch videos and then their opinions are recorded and averaged by calculating the MOS or Difference Mean Opinion Score (DMOS) to evaluate the quality of each video [5].

Subjective studies enable the performance of the objective methods and algorithms to be improved to achieve the ultimate goal of matching the human perception [64]. The best way to evaluate the performance of any video quality model is usually done by calculating the correlation between the model results and the results obtained with subject tests [8]. However, subjective studies are cumbersome, time consuming, expensive, difficult to implement, have to be undertaken manually, and impractical for most applications due to the human involvement in the evaluation process.

To evaluate the performance of a video quality model, different sets of parameters should be considered. These sets include bit rate, frame rate, frame size, video content, and transmission artifact during packet loss. Therefore, very large subjective tests should be performed [8]. For example, using only 5 video clips that are encoded using 5 different video codecs, at 5 different bit rates, with 5 different frame rates, and 5 different frame sizes (display size), will results in $5^5 = 3125$ videos.

Subjective evaluation sessions have some limitations, such as a) typically, no more than 30 videos can be presented at the same subjective session with no more than a one-hour session duration [8], and b) at least 15 subjects are needed for each video to obtain an appropriate confidence in the statistical results [67]. So, around 1562 subjective evaluation sessions should be performed to cover all the above 3125 videos.

Because of the above limitations for subjective evaluation methods, I decided to use objective evaluation methods to evaluate the quality of the transcoded videos generated from the above exploring and modeling phases. I used the following three Full-Reference (FR) metrics: PSNR [11], VSSIM [12], and MS-SSIM [9], which will be described in more detail in the next sections.

B.4. Peak Signal-to-Noise Ratio (PSNR)

PSNR [11] uses MSE [6] to calculate the spatial distortion. They are widely used because they are very simple to calculate, have a clear physical meaning, and are mathematically convenient in the context of optimization [64]. MSE is computed by averaging the square intensity difference of original, i.e., reference, and distorted, e.g., transcoded, image pixels as follows:

$$MSE = \frac{1}{m*n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \quad \text{(B.1)}$$

where I and K are the original and the distorted images, respectively. The width and height of the image is represented by m and n, respectively. PSNR is computed as follows [11]:

$$PSNR = 10\ log_{10}(\frac{{MAX_I}^2}{MSE}) \quad \text{(B.2)}$$

where MAX_I is the maximum possible pixel value of the image I, when the pixels are represented using 8 bits per sample, this is 255. Typical values for the PSNR in lossy image and video compression are between 30 and 50 dB, when the bit depth is 8 bits, where higher is better [11]. In spite of their widely used, they are not very well matched to the perceived visual quality [33].

To calculate the PSNR between two videos, I used the average of the PSNR values for all the frame-by-frame comparisons between the original and distorted videos.

B.5. Structural Similarity (SSIM)

SSIM metric is introduced by Wang et al. [12] and motivated based on the assumption that the HVS is highly adaptable for extracting structural information and its sensitivity to the structural distortion. This structural distortion is used to estimate the perceptual distortion for still images.

The structure of the objects in the image is independent of the illumination. However, it is hard to separate the structure of the image from the illumination influence. Wang et al. [12] defined the structural

information of the image as a local luminance and contrast that represent the structure of the objects in the scene [12]. In order to calculate the luminance, $l(x, y)$, between two non-negative discrete image signals x and y, Wang et al. [12] used the mean intensity, μ_x, as follows:

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \quad \text{(B.3)}$$

$$l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x{}^2 + \mu_y{}^2 + C_1} \quad \text{(B.4)}$$

where C_1 is a constant included to avoid instability when $\mu_x{}^2 + \mu_y{}^2$ (the denominator) is very close to zero and defined as $C_1 = (K_1 L)^2$ where L is the dynamic range of the pixel values (255 for 8-bit grayscale images) and K_1 is a small constant $K_1 \ll 1$.

In order to calculate the contrast, $c(x, y)$, Wang et al. [12] used the standard deviation (the square root of variance) as an estimate of the signal contrast, σ_x, as follows [68]:

$$\sigma_x = \left[\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2\right]^{1/2} \quad \text{(B.5)}$$

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x{}^2 + \sigma_y{}^2 + C_2} \quad \text{(B.6)}$$

where C_2 is a constant and defined as $C_2 = (K_2 L)^2$ and $K_2 \ll 1$.

In order to calculate the structural similarity, $s(x, y)$, Wang et al. [12] used the correlation coefficient, σ_{xy}, between x and y as follows [68]:

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y) \quad \text{(B.7)}$$

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \quad \text{(B.8)}$$

Finally, Wang et al. [12] combined the luminance, contrast, and structural components together and name the resulting similarity measure the Structural Similarity (SSIM) index between signals x and y as follows:

$$SSIM(x, y) = [(l(x, y)]^{\alpha}.[(c(x, y)]^{\beta}.[s(x, y)]^{\gamma} \quad \text{(B.9)}$$

where $\alpha > 0$, $\beta > 0$, and $\gamma > 0$ are parameters used to adjust the relative important of the above three components. To simplify the above expression, they set $\alpha = \beta = \gamma = 1$ and $C_3 = C_2/2$. The above index can be re-written as follows:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x{}^2+\mu_y{}^2+C_1)(\sigma_x{}^2+\sigma_y{}^2+C_2)} \quad \text{(B.10)}$$

To calculate the overall SSIM for the entire image, Wang et al. [12] used the mean SSIM for all the windows in the two image signals as follows:

$$MSSIM(x,y) = \frac{1}{M}\sum_{i=1}^{M} SSIM(x_i,y_i) \quad \text{(B.11)}$$

where x_i and y_i are the image contents at the i^{th} local window, and M is the number of local windows in the image. To calculate the SSIM between two videos, the average of the $MSSIM(x,y)$ values for all the frame-by-frame comparisons between the original and distorted videos are calculated as follows:

$$VSSIM(x,y) = \frac{1}{n}\sum_{i=1}^{n} MSSIM(x_i,y_i) \quad \text{(B.12)}$$

where x and y are the original and distorted videos, n represents the total number of frames.

B.6. Multi-Scale Structural Similarity (MS-SSIM)

Multi-scale structural similarity [9] is an extension of SSIM paradigm and, with other metrics, results in a high correlation with perceived video quality [33]. MS-SSIM was developed by Zhou Wang et al. [9] based on the assumption that a single-scale method may be appropriate for specific settings for subjective evaluation. The perceivability of image details depends on a) the sampling density of the image signal, b) the distance of the image plan to the observer, and c) the perceptual capability of the observer's visual system [9]. MS-SSIM is a convenient way to incorporate image details at different resolutions [9].

MS-SSIM is a full-reference image quality metric where the reference and the distorted images are applied as input, and then iteratively applies low pass filter and down-samples the filtered images by a factor of 2. This process iterates $M-1$ iterations starting at the original images as scale 1 till scale M, which is obtained after M-1 iterations. At each scale, the contrast components $c(x,y)$, and the structure components $s(x,y)$ between the reference image x and the distorted image y are calculated using (B.6) and (B.8) equations (refer to s and c above in the SSIM metric), respectively. While the luminance component is calculated only at scale M, to produce the overall MS-SSIM using equation (B.13). The overall process is depicted in Figure B.1.

$$MSSSIM(x,y) = [l_M(x,y)]^{\alpha M}.\prod_{j=1}^{M}[c_j(x,y)]^{\beta j}[s_j(x,y)]^{\gamma j} \quad \text{(B.13)}$$

To calculate the MS-SSIM between two videos, I used the average of the MSSSIM values for all the frame-by-frame comparisons between the original and distorted videos as follows:

$$MS - SSIM(x, y) = \frac{1}{n}\sum_{i=1}^{n} MSSSIM(x_i, y_i) \qquad \text{(B.14)}$$

where x and y are the original and distorted videos, n represents the total number of frames.

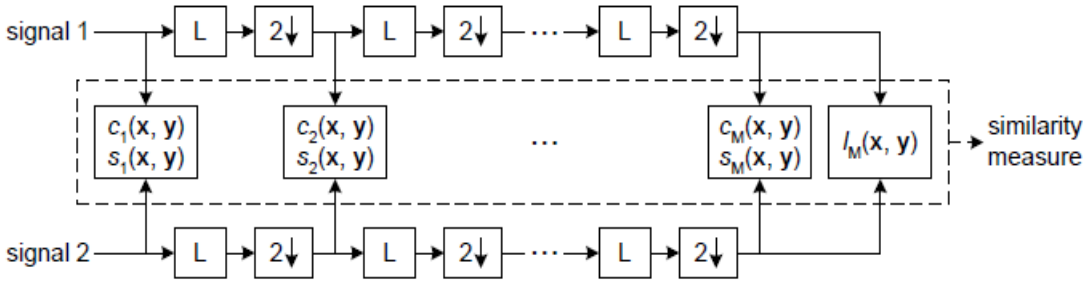


Figure B.1. Multi-scale structural similarity (MS-SSIM) measurement system. L: low-pass filtering; 2↓: down-sampling by 2 [9].

APPENDIX C

NAMING SCHEMAS AND EXPLORING THE IMPACT OF VIDEO TRANSCODING ON OBJECTIVE VIDEO QUALTY

C.1. Naming Scheme for the Encoded Videos

"S_IN_vqeghd1_A_B_C_D_E" represents the naming scheme that I used to name the encoded videos, where S represents a sequence number, A represents a video source name and number, B represents a video codec, C represents a bit rate, D represents a frame rate, and E represents a frame size. For example, "2_IN_vqeghd1_csrc11_H.264_4000kbps_29.97_1920x1080.mp4" indicates that the original, raw, uncompressed video is "vqeghd1_csrc11", which is used to generate a new video that is encoded using the H.264 codec at 4000 kbps as a bit rate, at 29.97 fps as a frame rate, and at 1920x1080 as a frame size.

I used mp4 as a file container for the H.264 and MPEG-4 codecs, and the flv as a file container for the FLV codec. Table C.1 shows samples of these names.

I grouped the encoded videos based on both the original video and codec used in video encoding. I used the following naming scheme to name the group: "vqeghd1_X_Y", where X represents a video source name and number, and Y represents a codec. For example, vqeghd1_src03_MPEG4 is the group that contains a set of encoded videos that are generated by encoding the "vqeghd1_src03" original video using the MPEG-4 codec at different values of bit rates. Figure C.1 shows examples of both the group names and some encoded videos.

Table C.1. Samples of some of the encoded videos' names.

7_IN_vqeghd1_csrc11_H.264_20000kbps_29.97_1920x1080.mp4
6_IN_vqeghd1_csrc14_FLV_15000kbps_29.97_1920x1080.flv
4_IN_vqeghd1_src01_MPEG4_8002kbps_29.97_1920x1080.mp4

C.2. Naming Scheme for the Frame Rate Transcoding

I used the following naming scheme: "S_OUT_vqeghd1_A_B_C_D_E" to name the transcoded videos, where S represents a sequence number, A represents a video source name and number, B represents a codec, C represents a bit rate, D represents a frame rate, and E represents a frame size. Figure C.2 shows the grouping scheme of the encoded and transcoded videos that I used for exploring and modeling the impact of reducing the frame rate on objective video quality.

For example, "1_IN_vqeghd1_csrc11_FLV_2000kbps_2997fps_1920x1080.flv" represents the input encoded video, while "1_OUT_vqeghd1_csrc11_FLV_2000kbps_25fps_1920x1080.flv" represents its corresponding transcoded video by reducing the frame rate from 29.97 fps to 25 fps using the FLV codec at 2000Kbps as a bit rate, and at 1920x1080 as a frame size.

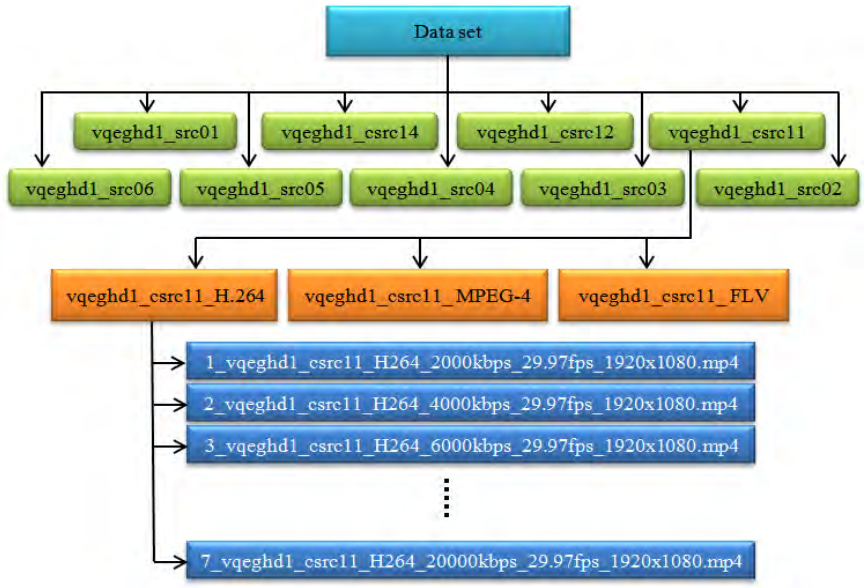


Figure C.1. The naming scheme for the encoded videos used for exploring the impact of video encoding on objective video quality.

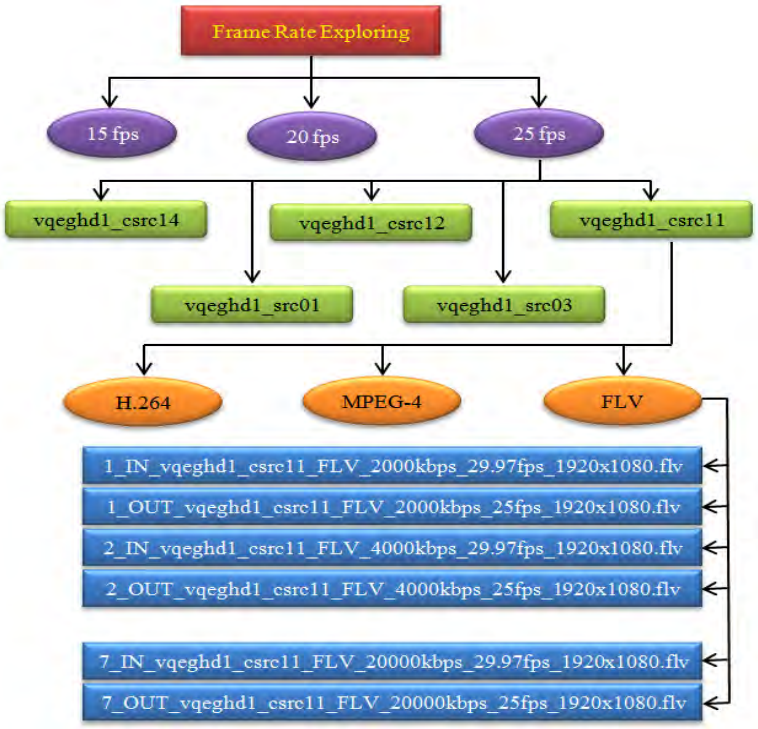


Figure C.2. The naming scheme for the encoded and transcoded videos used for exploring and modeling the impact of reducing the frame rate on objective video quality.

C.3. Naming Scheme for the Frame Size Transcoding

I used the following naming scheme: "S_OUT_vqeghd1_A_B_C_D_E" to name the transcoded videos, where S represents a sequence number, A represents a video source name and number, B represents a codec, C represents a bit rate, D represents a frame rate, and E represents a frame size. Figure C.3 shows the grouping structure of the encoded and transcoded videos that I used for exploring the impact of reducing the frame size on objective video quality.

For example, "1_IN_vqeghd1_csrc11_FLV_2000kbps_2997fps_1920x1080.flv" represents the input encoded video, while "1_OUT_vqeghd1_csrc11_FLV_2000kbps_29.97fps_1440x900.flv" represents its corresponding transcoded video by reducing the frame size from 1920x1080 to 1440x900 using the FLV codec at 2000Kbps as a bit rate, and at 29.97 as a frame rate.

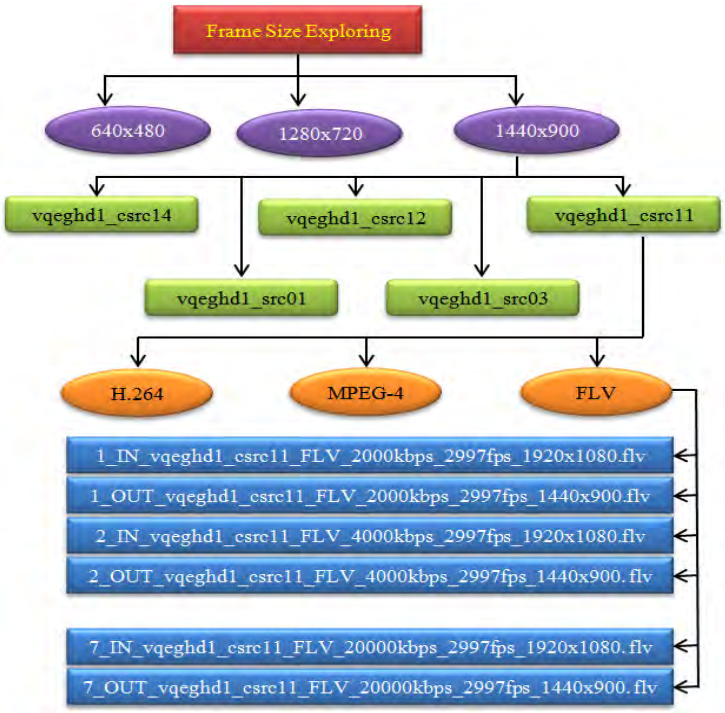


Figure C.3. The naming scheme for the encoded and transcoded videos used for exploring and modeling the impact of reducing the frame size on objective video quality.

C.4. Naming Scheme for the Bit rate Transcoding

I used the following naming scheme: "S_OUT_vqeghd1_A_B_C_D_E" to name the transcoded videos, where S represent a sequence number, A represents a video source name and number, B represents a codec, C represents a bit rate, D represents a frame rate, and E represents a frame size. Figure C.4 shows the grouping structure of the encoded and transcoded videos that I used for exploring and modeling the impact of reducing the bit rate on objective video quality.

For example, "2_IN_vqeghd1_csrc11_H.264_8000kbps_2997fps_1920x1080.mp4" represents the input encoded video, while "2_OUT_vqeghd1_csrc11_H.264_4000kbps_29.97fps_1920x1080.mp4" represents its corresponding transcoded video by reducing the bit rate 50% from the encoded bit rate using the H.264 codec at 29.97 fps as a frame rate and at 1920x1080 as a frame size.

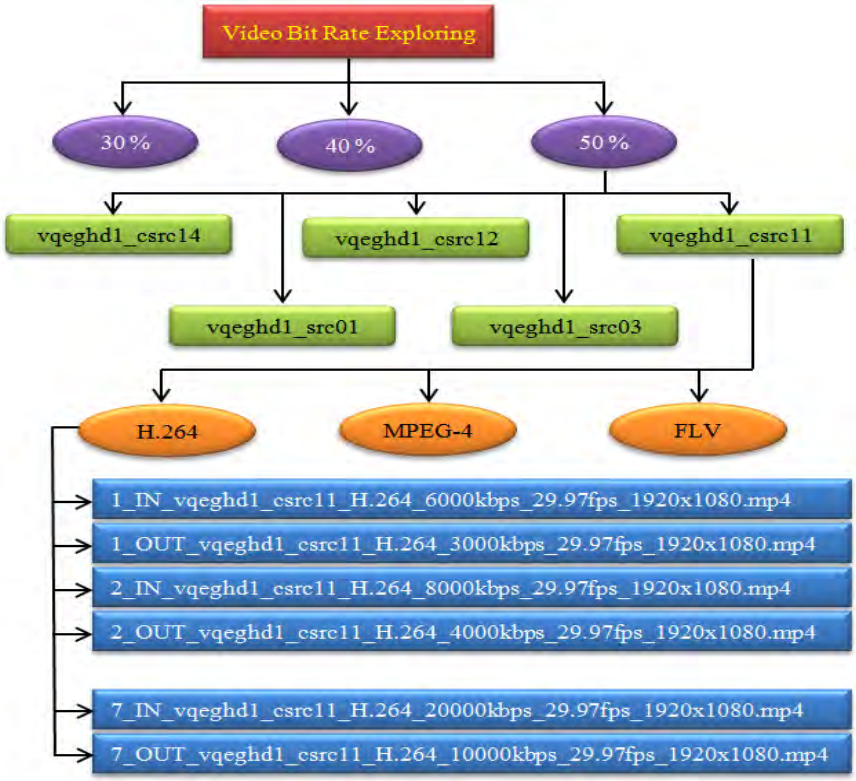


Figure C.4. The naming scheme for the encoded and transcoded videos used for exploring the impact of reducing the bit rate on objective video quality.

C.5. Video Quality Results for Changing the Video Codec

Table C.2 shows the video quality evaluation results for exploring the impact of changing the codec on objective video quality. It shows that the conversion from FLV to MPEG-4 achieves the highest quality in terms of PSNR, VSSIM, and MS-SSIM when the bit rate is higher than 15 Mbps. The transcoding from H.264 to MPEG-4 and the transcoding from MPEG-4 to H.264 achieve the highest quality when the bit rate is lower than 15 Mbps.

C.6. Video Quality Results for Reducing the Frame Rate

Figure C.5 shows the video quality evaluation results for the frame rate reductions from 29.97 fps to 25 fps, 20 fps, and 15 fps in terms of PSNR, VSSIM, and MS-SSIM for the H.264 codec. Figure C.5(a)

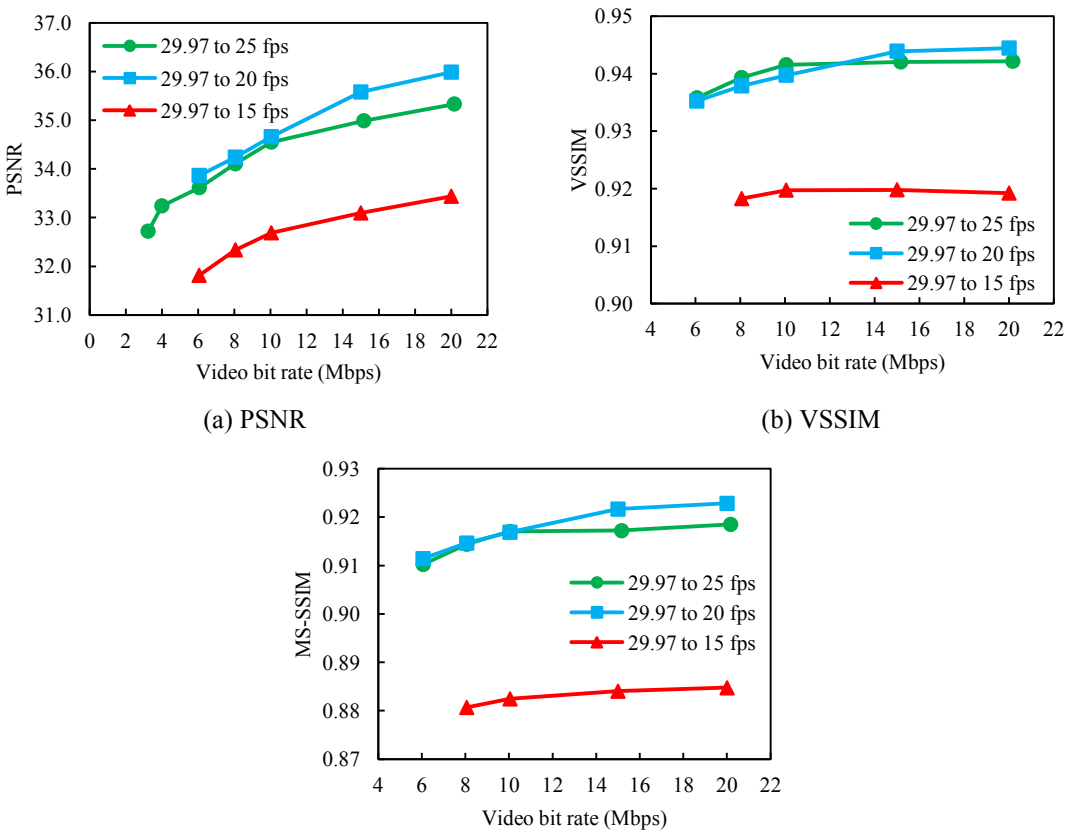
shows that the quality is generally increased in terms of PSNR when the bit rate is increased.

Table C.2. Video quality evaluation results for the video codec conversions.

(a) From FLV to H.264				(b) From FLV to MPEG-4			
Bit rate (Mbps)	PSNR	VSSIM	MS-SSIM	Bit rate (Mbps)	PSNR	VSSIM	MS-SSIM
2.0	30.55	0.8914	0.9012	2.6	32.38	0.9134	0.9000
4.0	31.86	0.9053	0.9139	4.0	35.29	0.9468	0.9348
6.0	33.57	0.9245	0.9296	6	37.25	0.9595	0.9497
8.0	34.95	0.9417	0.9405	8.0	37.66	0.9625	0.9527
10.0	35.38	0.9459	0.9440	10	38.25	0.9638	0.9550
15.0	36.81	0.9651	0.9725	15	41.23	0.9836	0.9844
20.0	37.42	0.9752	0.9818	20.0	41.91	0.9856	0.9865

(c) From H.264 to FLV				(d) From H.264 to MPEG-4			
bit rate (Mbps)	PSNR	VSSIM	MS-SSIM	Bit rate (Mbps)	PSNR	VSSIM	MS-SSIM
6.0	32.58	0.8977	0.9173	4.0	34.53	0.9378	0.9517
8.1	33.91	0.9196	0.9360	6.0	35.20	0.9445	0.9512
10.0	34.92	0.9290	0.9479	8.1	36.19	0.9547	0.9618
15.0	36.64	0.9537	0.9647	10.0	37.00	0.9555	0.9669
20.0	37.60	0.9619	0.9719	15.0	38.24	0.9680	0.9748
				20.0	38.85	0.9710	0.9779

(e) From MPEG4 to H.264				(f) From MPEG-4 to FLV			
bit rate (Mbps)	PSNR	VSSIM	MS-SSIM	bit rate (Mbps)	PSNR	VSSIM	MS-SSIM
2.7	33.39	0.9237	0.9431	4.0	32.34	0.8961	0.9167
4.0	33.74	0.9348	0.9496	6.0	33.49	0.9135	0.9287
6.1	34.72	0.9471	0.9567	8.0	34.62	0.9286	0.9404
8.0	35.52	0.9539	0.9615	10.0	35.10	0.9357	0.9462
10.1	36.05	0.9591	0.9662	15.0	36.17	0.9503	0.9587
15.0	36.90	0.9668	0.9741	20.0	37.51	0.9634	0.9702
20.0	37.58	0.9703	0.9771				



(a) PSNR

(b) VSSIM

(c) MS-SSIM

Figure C.5. The video quality evaluation results for exploring the impact of reducing the frame rate on objective video quality for the H.264 video codec. The reductions are from 29.97 fps to 25 fps, 20 fps, and 15 fps. The quality results are in terms of (a) PSNR, (b) VSSIM, and (c) MS-SSIM.

Also, Figure C.5(a) shows that the frame rate reductions from 29.97 fps to 25 fps and 20 fps achieve better quality than the frame rate reduction from 29.97 fps to 15 fps. In addition, it shows that there is no big difference in the quality between the frame rate reduction from 29.97 fps to 25 fps and the frame rate reduction from 29.97 fps to 20 fps. Moreover, it shows that reducing the frame rate from 29.97 to 20 achieves better quality than reducing the frame rate from 29.97 to 25. However, the difference is very small, especially when the bit rate is lower than 10 Mbps.

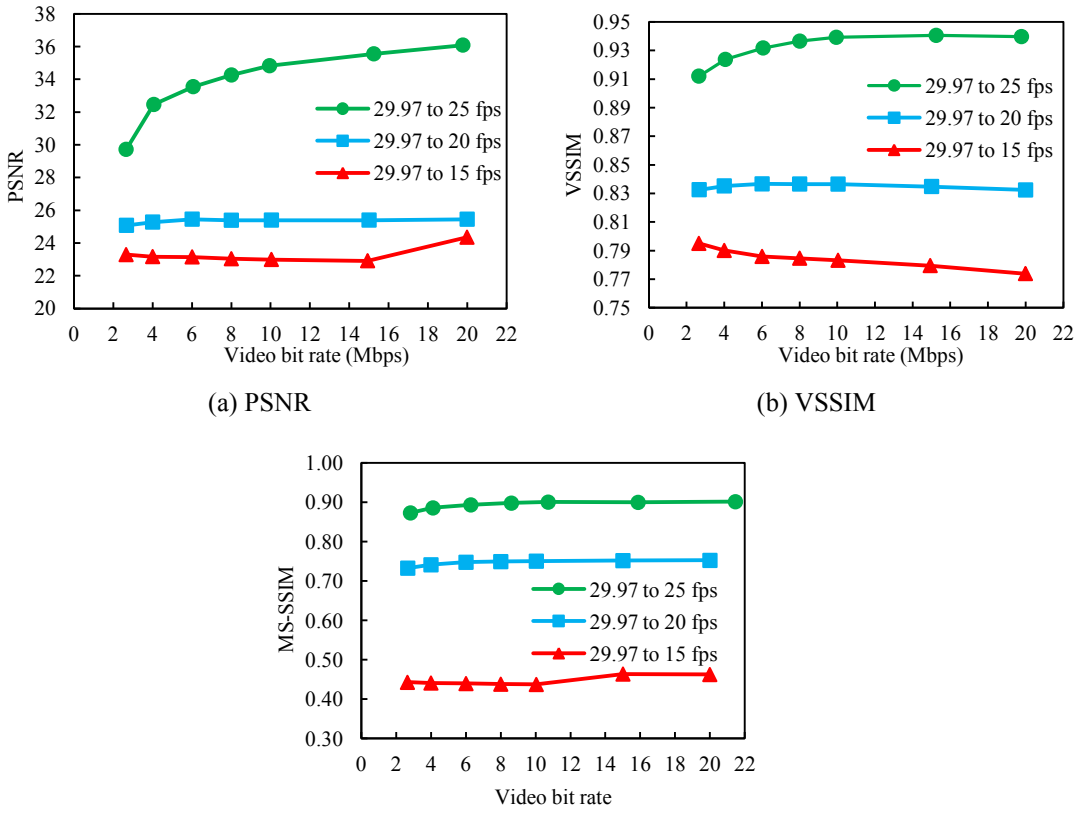
Figure C.5(b) shows that the frame rate reductions from 29.97 fps to 25 fps and 20 fps achieve better quality in terms of VSSIM than the frame rate reduction from 29.97 fps to 15 fps. Also, it shows that the quality is increased when the bit rate is increased. For the frame rate reduction from 29.97 fps to 15 fps,

there is no big improvement in the quality when the bit rate is increased. Moreover, it shows that there is no big difference in the quality between the frame rate reduction from 29.97 fps to 25 fps and the frame rate reduction from 29.97 fps to 20 fps.

Figure C.5(c) shows that the frame rate reductions from 29.97 fps to 25 fps and 20 fps achieve better quality in terms of MS-SSIM than the frame rate reduction from 29.97 fps to 15 fps. Also, it shows that the frame rate reduction from 29.97 fps to 25 fps and the frame rate reduction from 29.97 fps to 20 fps have close quality results when the value of the bit rate is lower than 10 Mbps; after that, there is a noticeable difference in the quality.

Figure C.6 shows the video quality evaluation results for the frame rate reductions from 29.97 fps to 25 fps, 20 fps, and 15 fps in terms of PSNR, VSSIM, and MS-SSIM for the MPEG-4 codec. Figure C.6(a) shows that for the frame rate reduction from 29.97 fps to 25 fps, the quality is increased in terms of PSNR when the bit rate is increased. For the frame rate reductions from 29.97 fps to 20 fps and 15 fps, there is almost no improvement in the quality when the bit rate is increased. In addition, it shows that the frame rate reduction from 29.97 fps to 25 fps achieves better quality than the frame rate reductions from 29.97 fps to 20 fps and 15 fps. Moreover, it shows that there is no big difference in the quality between the frame rate reduction from 29.97 fps to 20 fps and the frame rate reduction from 29.97 fps to 15 fps. Also, there is a noticeable difference in the quality between the frame rate reduction from 29.97 fps to 25 fps and the frame rate reductions from 29.97 fps to 20 fps and 15 fps when the bit rate is higher than 15 Mbps.

Figure C.6(b) shows that the frame rate reduction from 29.97 fps to 25 fps achieves better quality in terms of VSSIM than the frame rate reductions from 29.97 fps to 20 fps and 15 fps. Also, it shows that the quality is increased by a small amount when the bit rate is increased for the frame rate reduction from 29.97 fps to 25 fps. For the frame rate reduction from 29.97 fps to 20 fps, there is no big improvement in the quality when the bit rate is increased. For the frame rate reduction from 29.97 fps to 15 fps, the quality is slightly decreased when the bit rate is increased. In addition, it shows that there is a noticeable difference in the quality between the frame rate reduction from 29.97 fps to 20 fps and the frame rate reduction from 29.97 fps to 15 fps.



(a) PSNR

(b) VSSIM

(c) MS-SSIM

Figure C.6. The video quality evaluation results for exploring the impact of reducing the frame rate on objective video quality for the MPEG-4 video codec. The reductions are from 29.97 fps to 25 fps, 20 fps, and 15 fps. The quality results are in terms of (a) PSNR, (b) VSSIM, and (c) MS-SSIM.

Figure C.6(c) shows that the frame rate reduction from 29.97 fps to 25 fps achieves better video quality in terms of MS-SSIM than the frame rate reductions from 29.97 fps to 20 fps and 15 fps. Also, it shows that the quality is increased by a small amount when the bit rate is increased for the frame rate reduction from 29.97 fps to 25 fps and the frame rate reduction from 29.97 fps to 20 fps.

For the frame rate reduction from 29.97 fps to 15 fps, there is no big improvement in the quality in terms of MS-SSIM when the bit rate is increased within the range from 6 to 15 Mbps. There is a noticeable improvement in the quality when the bit rate is increased at values that are higher than 15 Mbps. In addition, for the frame rate reduction from 29.97 fps to 15 fps, the quality is decreased when the bit rate is

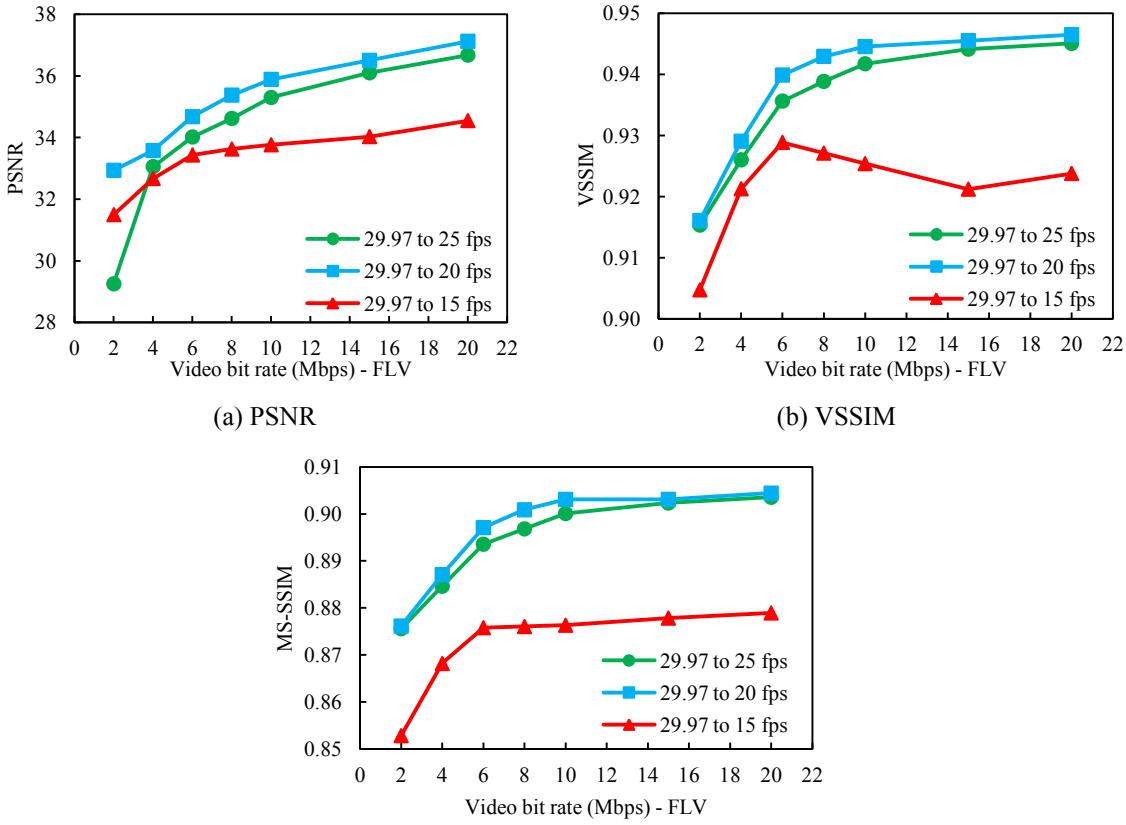
increased within the range from 4 to 6 Mbps, and the quality is increased when the bit rate is increased within the range from 2 to 4 Mbps.

Figure C.7 shows the video quality results for the frame rate reductions from 29.97 fps to 25 fps, 20 fps, and 15 fps in terms of PSNR, VSSIM, and MS-SSIM for the FLV codec. Figure C.7(a) shows that the quality is increased in terms of PSNR when the bit rate is increased for all the frame rate reductions, i.e., from 29.97 fps to 25 fps, from 29.97 fps to 20 fps, and from 29.97 fps to 25 fps. In addition, it shows that the frame rate reduction from 29.97 fps to 20 fps achieves better quality than the frame rate reduction from 29.97 fps to 25 fps, and the frame rate reduction from 29.97 fps to 25 fps achieves better quality than the frame rate reduction from 29.97 fps to 15 fps. Moreover, it shows that there is no big difference in the quality between the frame rate reduction from 29.97 fps to 25 fps and the frame rate reduction from 29.97 fps to 20 fps.

Figure C.7(b) shows that the frame rate reduction from 29.97 fps to 20 fps achieves better quality in terms of VSSIM than the frame rate reduction from 29.97 fps to 25 fps and the frame rate reduction from 29.97 fps to 15 fps. Also, it shows that the quality is increased when the bit rate is increased for the frame rate reduction from 29.97 fps to 25 fps and for the frame rate reduction from 29.97 fps to 20 fps. For the frame rate reduction from 29.97 fps to 15 fps, it shows that the quality is increased when the bit rate is increased within the range from 2 to 6 Mbps and from 15 to 20 Mbps, and the quality is decreased when the bit rate is increased within the range from 6 to 15 Mbps. In addition, it shows that there is no big difference in the quality between the frame rate reduction from 29.97 fps to 20 fps and the frame rate reduction from 29.97 fps to 25 fps. Moreover, it shows that there is a noticeable difference in the quality between the frame rate reductions from 29.97 fps to 20 fps and 25 fps, and the frame rate reduction from 29.97 fps to 15 fps.

Figure C.7(c) shows that the quality is increased in terms of MS-SSIM when the bit rate is increased for all the frame rate reductions, i.e., from 29.97 fps to 25 fps, from 29.97 fps to 20 fps, and from 29.97 fps to 15 fps. Also, it shows that the frame rate reductions from 29.97 fps to 25 fps and to 20 fps achieve better quality than the frame rate reduction from 29.97 fps to 15 fps. In addition, it shows that there is almost no difference in the quality between the frame rate reduction from 29.97 fps to 20 fps and the frame rate reduction from 29.97 fps to 25 fps. Moreover, it shows that there is no big difference in the

quality for the frame rate reduction from 29.97 fps to 15 fps when the bit rate is increased within the range from 6 to 15 Mbps.



(a) PSNR

(b) VSSIM

(c) MS-SSIM

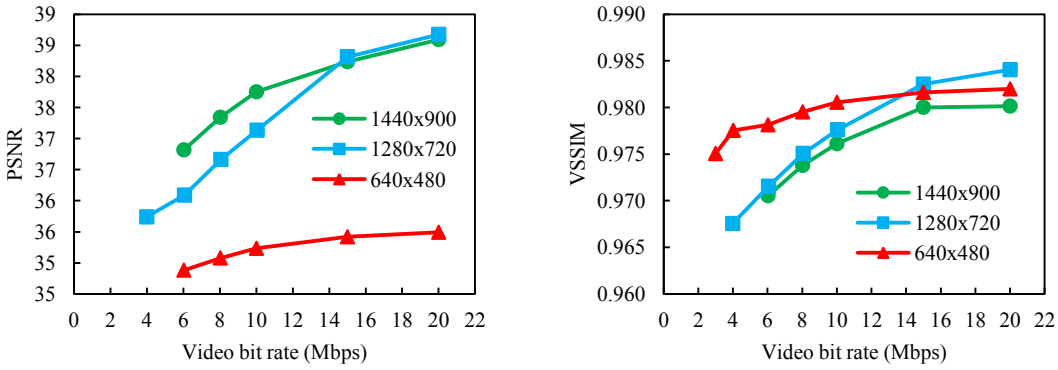
Figure C.7. The video quality evaluation results for exploring the impact of reducing the frame rate on objective video quality for the FLV video codec. The reductions are from 29.97 fps to 25 fps, 20 fps, and 15 fps. The quality results are in terms of (a) PSNR, (b) VSSIM, and (c) MS-SSIM.

C.7. Video Quality Results for Reducing the Frame Size

Figure C.8 shows the video quality evaluation results of the frame size reductions from 1920x1080 to 1440x900, 1280x720, and 640x480 in terms of PSNR, VSSIM, and MS-SSIM using the H.264 codec. Figure C.8(a) shows that the frame size reduction from 1920x1080 to 640x480 achieves the lowest quality in terms of PSNR. Also, it shows that the frame size reduction from 1920x1080 to 1440x900 achieves the

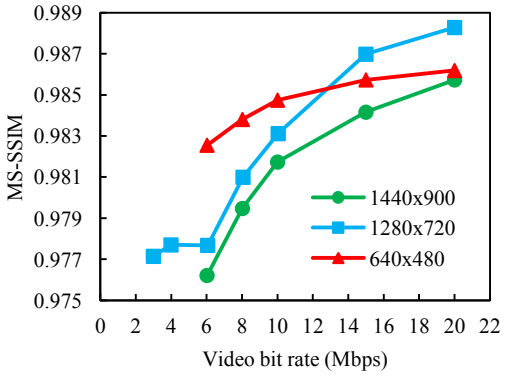
highest quality when the bit rate is lower than 15 Mbps. At higher bit rates, the frame size reductions from 1920x1080 to 1440x900 and from 1920x1080 to 1280x720 achieve almost the same quality results.

Figure C.8(b) shows different video quality results than Figure C.8(a); it shows that the frame size reduction from 1920x1080 to 640x480 achieves the highest quality results in terms of VSSIM when the bit rate is lower than 15 Mbps. At higher bit rates, the frame size reduction from 1920x1080 to 1280x720 achieves the highest quality results. In addition, it shows that the frame size reduction from 1920x1080 to 1440x900 achieves the lowest quality results when the bit rate is greater than 6 Mbps. Figure C.8(c) shows almost the same trend and values of the quality results as Figure C.8(b), but in terms of MS-SSIM.



(a) PSNR

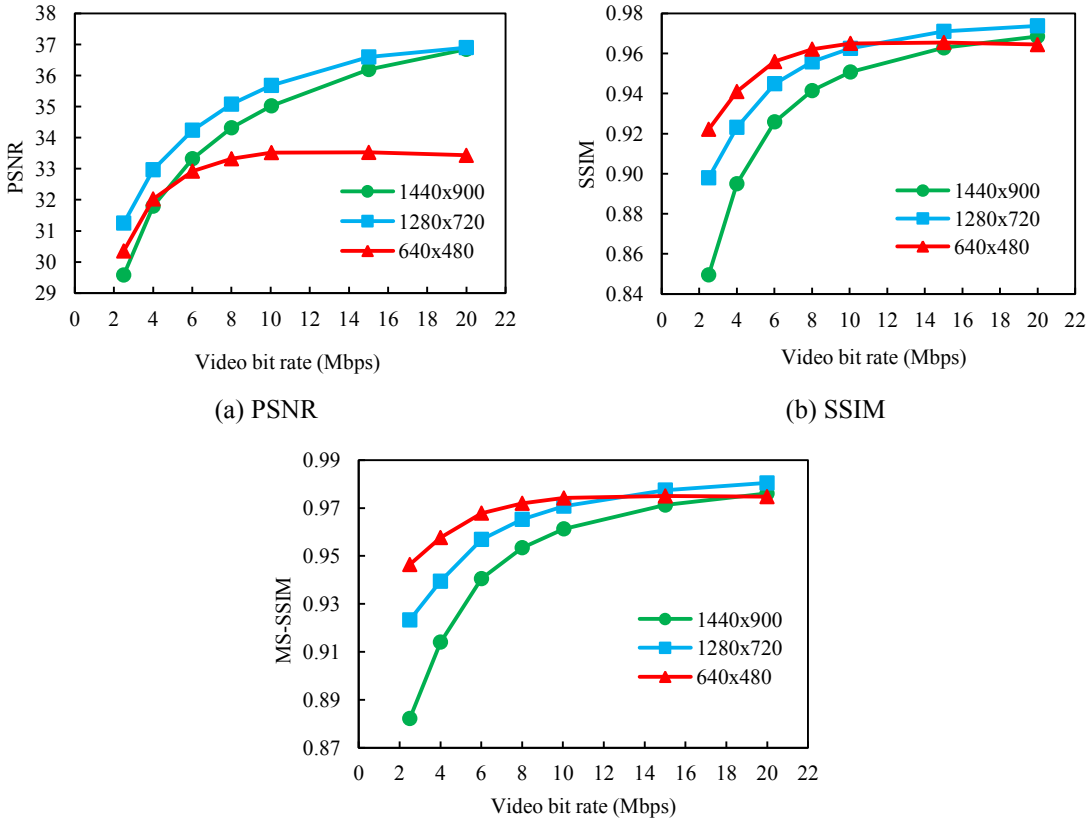
(b) VSSIM



(c) MS-SSIM

Figure C.8. The video quality evaluation results for exploring the impact of reducing the frame size on objective video quality for the H.264 video codec. The reductions are from 1920x1080 to 1440x900, 1280x720, and 640x480. The quality results are in terms of (a) PSNR, (b) VSSIM, and (c) MS-SSIM.

Figure C.9 shows the quality results for the frame size reductions from 1920x1080 to 1440x900, 1280x720, and 640x480 in terms of PSNR, VSSIM, and MS-SSIM for the MPEG-4 codec. Figure C.9(a) shows that the frame size reduction from 1920x1080 to 1280x720 achieves the highest quality results in terms of PSNR. The second highest quality results are for the frame size reduction from 1920x1080 to 1440x900. In addition, it shows that the frame size reduction from 1920x1080 to 640x480 achieves the lowest quality results.



(a) PSNR

(b) SSIM

(c) MS-SSIM

Figure C.9. The video quality evaluation results for exploring the impact of reducing the frame size on objective video quality for the MPEG-4 video codec. The reductions are from 1920x1080 to 1440x900, 1280x720, and 640x480. The quality results are in terms of (a) PSNR, (b) SSIM, and (c) MS-SSIM.

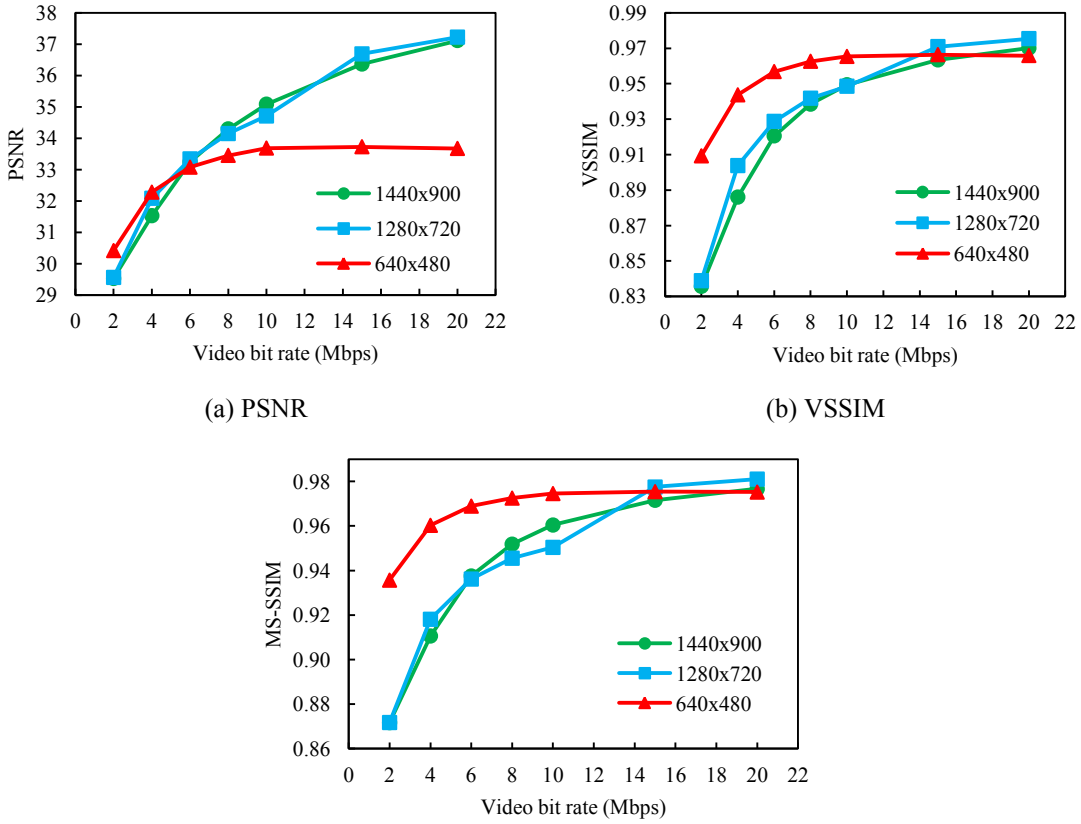
Figures C.9(b) and (c) show smooth increased curves in the quality in terms of VSSIM and MS-SSIM, respectively, when the bit rate is increased. They show that the video frame size reductions from 1920x1080 to 640x480, from 1920x1080 to 1280x720, and from 1920x1080 to 1440x900 achieve the first, second, and third highest quality results, respectively, in terms of VSSIM and MS-SSIM when the bit rate is lower than 13 Mbps. When the bit rate is higher than 13 Mbps, all the above frame rate reductions have very close quality results.

Figure C.10 shows the quality results for the frame size reductions from 1920x1080 to 1440x900, 1280x720, and 640x480 in terms of PSNR, VSSIM, and MS-SSIM for the FLV codec. Figure C.10(a) shows that the frame size reductions from 1920x1080 to 1280x720 and from 1920x1080 to 1440x900 achieve the highest and almost the same quality results in terms of PSNR when the bit rate is higher than 6 Mbps. Also, it shows that the video frame size reduction from 1920x1080 to 640x480 achieves the lowest quality when the bit rate is greater than 6 Mbps. In addition, it shows that there is no gap between all the video frame size reductions when the bit rate is less than 6 Mbps.

Figures C.10(b) and (c) show that the quality in terms of VSSIM and MS-SSIM, respectively, is increased when the bit rate is increased. Also, they show that the frame size reduction from 1920x1080 to 640x480 achieves the highest quality when the bit rate is lower than 15 Mbps. Moreover, the frame size reductions from 1920x1080 to 1280x720 and from 1920x1080 to 1440x900 have close results when the bit rate is lower than 15 Mbps. All of the above frame size reductions have close quality results when the bit rate is higher than 15 Mbps.

C.8. Video Quality Results for Reducing the Bit Rate

Figures C.11, C.12, and C.13 show the video quality evaluation results for the three bit rate reductions, i.e., 30%, 40%, and 50%, in terms of PSNR, VSSIM, and MS-SSIM for the H.264, MPEG-4, and FLV codecs, respectively.



(a) PSNR

(b) VSSIM

(c) MS-SSIM

Figure C.10. The video quality evaluation results for exploring the impact of reducing the frame size on objective video quality for the FLV video codec. The reductions are from 1920x1080 to 1440x900, 1280x720, and 640x480. The quality results are in terms of (a) PSNR, (b) VSSIM, and (c) MS-SSIM.

As expected, these figures show that the 30% reduction in the bit rate achieves the highest quality results in terms of PSNR, VSSIM, and MS-SSIM. The second and third highest video qualities are achieved at 40% and 50% bit rate reductions from the original bit rate.

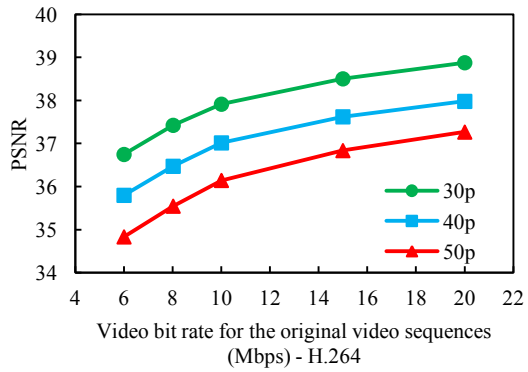
Also, they show that the difference in the quality results between the 30% and 40% reductions in the bit rate is almost the same as the difference in the quality results between the 40% and 50% reductions in the bit rate from the original bit rate.

Figure C.11 shows that the quality evaluation results are improved in a bigger amount when the bit rate ranges from 6 to 10 Mbps. For the bit rates that are range from 10 to 20 Mbps, the improvement is

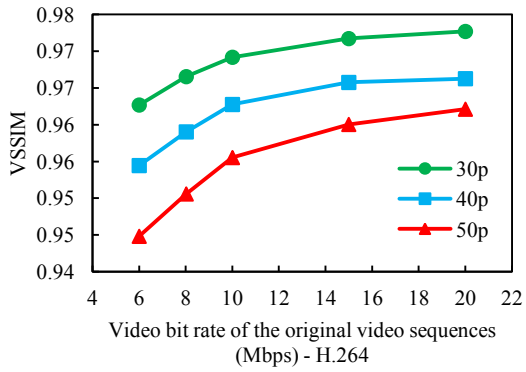
small. In addition, it shows that all the reductions in the bit rate have almost the same trend. Moreover, it shows that the H.264 codec can reduce the bit rate to 50% from the original bit rate when the original bit rate is at 6 Mbps.

Figure C.12 shows that the MPEG-4 codec is unable to reduce the bit rate for some of the original videos more than 50% when the bit rate of the original videos is less than 8 Mbps.

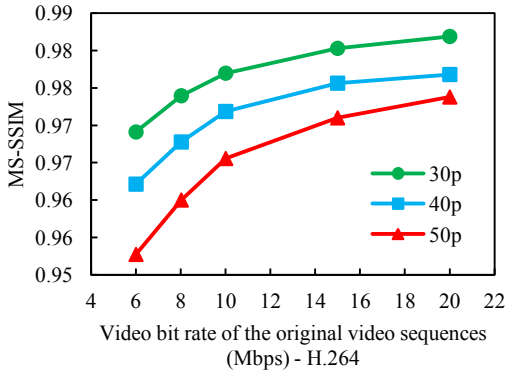
Figure C.13 shows that the FLV codec is able to reduce the bit rate for all the original videos at 50% reduction percentage from the original bit rate.



(a) PSNR



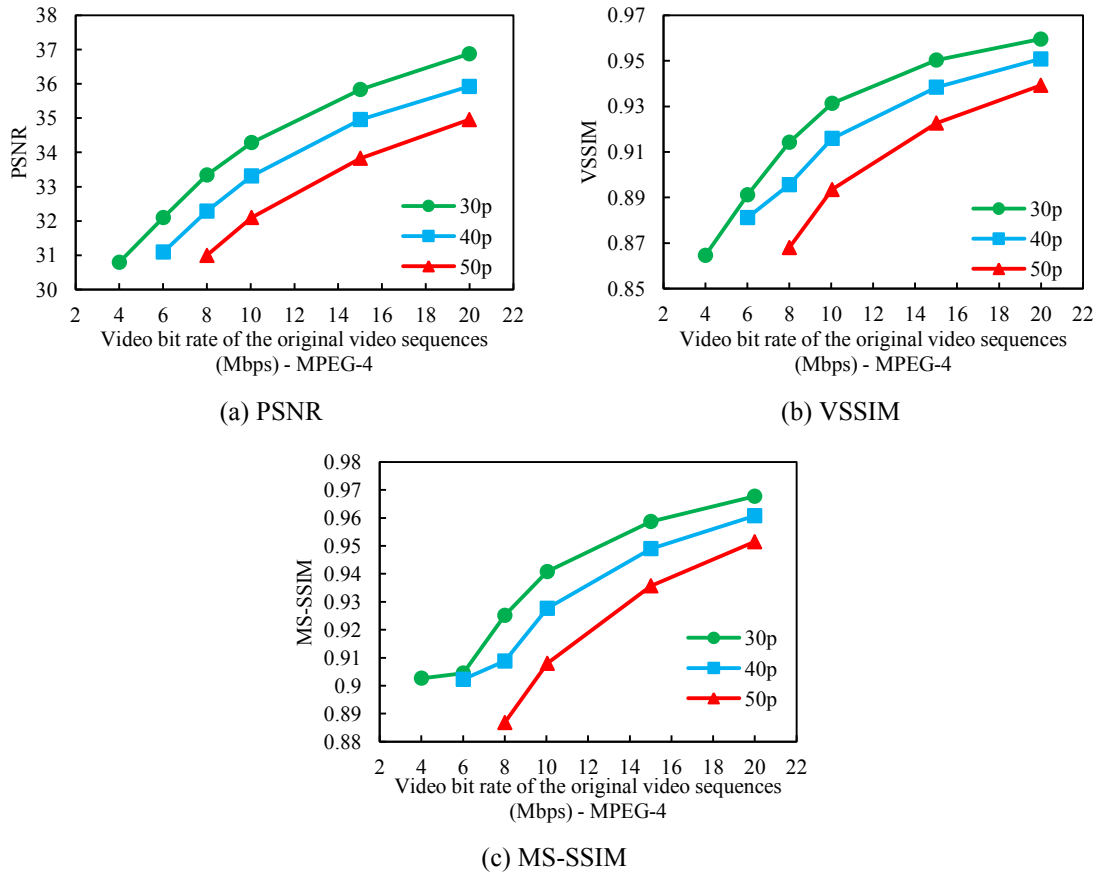
(b) VSSIM



(c) MS-SSIM

Figure C.11. The video quality evaluation results for exploring the impact of reducing the bit rate on objective video quality for the H.264 video codec. The reduction percentages are 30%, 40%, and 50% from the original bit rate. The quality results are in terms of (a) PSNR, (b) VSSIM, and (c) MS-SSIM. (For clarification, p means %.)

Figure C.13(a) shows that there is a small decrease in the quality when the value of the bit rate is increased within the range from 8 to 10 Mbps. Figures C.13(b) and (c) show that there is a small improvement in the quality when the value of the bit rate is increased within the same range, i.e., from 8 to 10 Mbps.

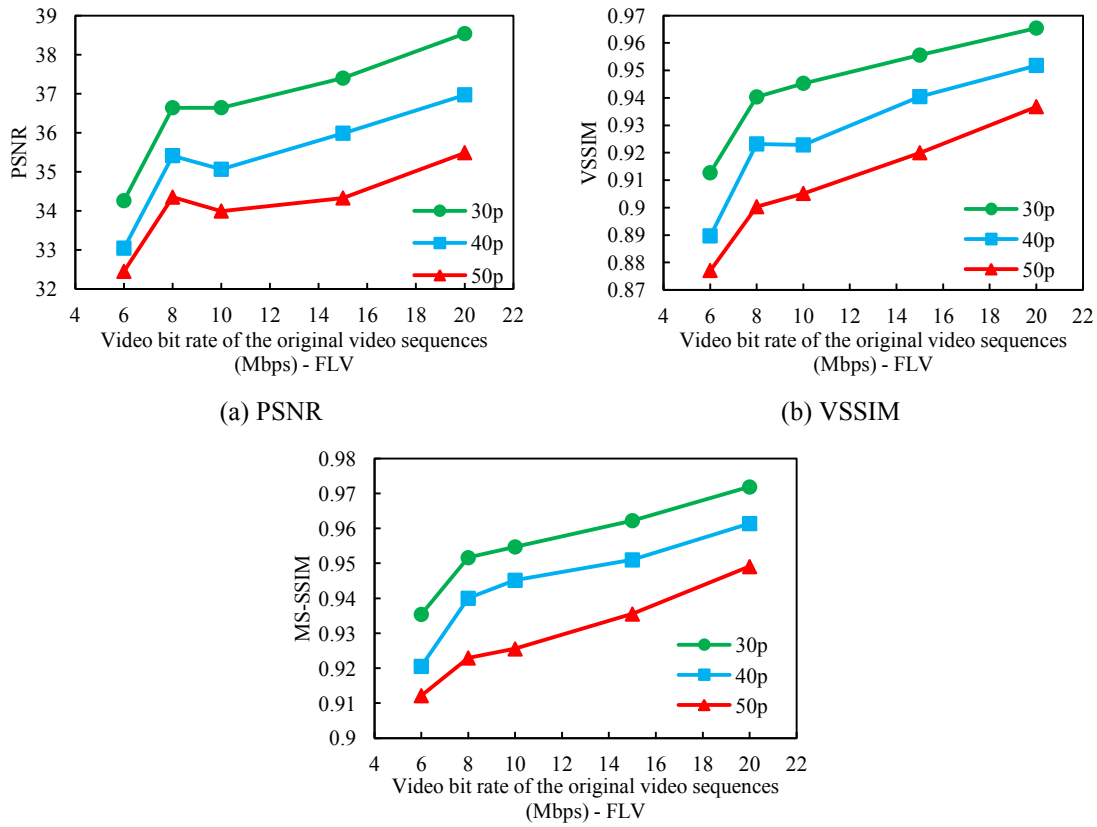


(a) PSNR

(b) VSSIM

(c) MS-SSIM

Figure C.12. Video quality evaluation results for exploring the impact of reducing the bit rate on objective video quality for the MPEG-4 video codec. The reduction percentages are 30%, 40%, and 50% from the original bit rate. The quality results are in terms of (a) PSNR, (b) VSSIM, and (c) MS-SSIM. (For clarification, p means %.)



(a) PSNR

(b) VSSIM

(c) MS-SSIM

Figure C.13. Video quality evaluation results for exploring the impact of reducing the bit rate on objective video quality for the FLV video codec. The reduction percentages are 30%, 40%, and 50% from the original bit rate. The quality results are in terms of (a) PSNR, (b) VSSIM, and (c) MS-SSIM. (For clarification, p means %.)

APPENDIX D

QUALITY AND ERROR SUB-MODELS

D.1. Modeling the Impact of Changing the Video Codec

This section describes the rest of the quality sub-models that assess the impact of changing the codec on objective quality in video transcoding. These models are a) from H.264 to FLV, b) from MPEG-4 to H.264, c) from MPEG-4 to FLV, d) from FLV to H.264, and e) from FLV to MPEG-4.

D.1.1. From H.264 to FLV Transcoding

I modeled the quality of the transcoded video, generated by changing the codec from H.264 to FLV, on objective quality at a given bit rate, br, as follows:

$$y.quality_{\alpha_{H264,FLV}(x.c.br),x} = 1 - (0.008519 + \left(\frac{log_2(br+5.07627)}{e^{log_2(br+3.858446)}*br^{0.304657}}\right)) \quad \text{(D.1)}$$

where x and y are the original and transcoded videos, respectively. The codec for video x is H.264 and for video y is FLV.

D.1.2. From MPEG-4 to H.264 Transcoding

I modeled the quality of the transcoded video, generated by changing the codec from MPEG-4 to H.264, on objective quality at a given bit rate, br, as follows:

$$y.quality_{\alpha_{MPEG4,H264}(x.c.br),x} = 1 - (0.0128 + \left(\frac{log_2(br*0.8756)}{e^{log_2(br*2.594)}+(br*4.3463)}\right)) \quad \text{(D.2)}$$

where x and y are the original and transcoded videos, respectively. The codec for video x is MPEG-4 and for video y is H.264.

D.1.3. From MPEG-4 to FLV Transcoding

I modeled the quality of the transcoded video, generated by changing the codec from MPEG-4 to FLV, on objective quality at a given bit rate, br, as follows:

$$y.quality_{\alpha_{MPEG4,FLV}(x.c.br),x} = 1 - (0.0129 + \left(\frac{log_2(br+11.16775)}{((log_2(br+13.1737))^{-0.13548})*e^{log_2(br+11.04114)}}\right)) \quad \text{(D.3)}$$

where x and y are the original and transcoded videos, respectively. The codec for video x is MPEG-4 and for video y is FLV.

D.1.4. From FLV to H.264 Transcoding

I modeled the quality of the transcoded video, generated by changing the codec from FLV to H.264, on objective quality at a given bit rate, br, as follows:

$$y.quality_{\alpha_{FLV,H264}(x.c.br),x} = (-0.0001 * (br\text{^}2)) + (0.0078 * br) + 0.8858) \quad \text{(D.4)}$$

where x and y are the original and transcoded videos, respectively. The codec for video x is FLV and for video y is H.264.

D.1.5. From FLV to MPEG-4 Transcoding

I modeled the quality of the transcoded video, generated by changing the codec from FLV to MPEG-4, on objective quality at a given bit rate, br, as follows:

$$y.quality_{\alpha_{FLV,MPEG4}(x.c.br),x} = 1 - (0.0106 + \left(\frac{(\ln(br))^{1.7506}}{(e^{log_2(br*3.4527)}*\sqrt{(br+2.8842)})}\right)) \quad \text{(D.5)}$$

where x and y are the original and transcoded videos, respectively. The codec for video x is FLV and for video y is MPEG-4.

The values of all the above quality sub-models, i.e., functions, range from 0 to 1, the higher the value, the better the quality. To test changing the codec in the above sub-models, I performed steps similar to those described in Chapter 6. The total number of encoded videos generated from each encoding process for each codec conversion is 28. During this transcoding, I kept the bit rate, frame rate, and frame size the same as in the original video. The only thing that I changed in this transcoding was the codec. The total number of transcoded videos generated from each transcoding for each codec conversion is 28. I calculated the reference values by following the same steps described in Chapter 4.

Figure 6.1 in Chapter 6 shows the quality evaluation results for modeling the impact of changing the codec on objective video quality for each video in the testing set and the reference values that represent the average of the actual quality values for all videos used in the training set in terms of MS-SSIM. Also, it shows the model results. Figure 6.1 shows high correlations between the predicted quality values generated

from these sub-models and the reference quality values. It also shows high correlations between the predicted quality results generated from these sub-models and the quality results generated from using MS-SSIM for each video in the testing set. Table 6.1 in Chapter 6 shows high PCC values between the predicted quality values generated from the proposed sub-models and the actual quality values generated from using MS-SSIM for each video in the testing set. It also shows low RMSE values.

D.2. Modeling the Impact of Reducing the Bit Rate

This section describes the rest of the quality sub-models that assess the impact of reducing the bit rate on objective video quality. These models are for the MPEG-4 and FLV codecs.

D.2.1. MPEG-4 Video Codec

I modeled the quality of the transcoded video, generated by reducing the bit rate from br_{in} to br_{out}, on objective video quality for the MPEG-4 codec as follows:

$$y.quality_{\beta_{MPEG4}(x.c.br_{in},y.c.br_{out}),x} = a_1 + \left(\frac{log_2(br_{out}*a_2)*\ln(br_{in}{}^{a_3})}{log_2(br_{in}*a_4)*\ln(br_{out}*a_5)}\right) \quad \text{(D.6)}$$

where x and y are the original and transcoded videos, respectively. The codec for videos x and y is MPEG-4. The bit rate for video x is br_{in} and for video y is br_{out} in Mbps ($br_{in} > br_{out}$). The sub-model coefficients are a_1, a_2, a_3, a_4, and a_5, calculated as follows

$$a_1 = -0.67613 * x^2 + 0.83722 * x - 0.02499 \quad \text{(D.7)}$$

$$a_2 = 3.20461 * x^2 - 2.28249 * x + 0.52697 \quad \text{(D.8)}$$

$$a_3 = 14.74001 * x^2 - 11.90797 * x + 3.54817 \quad \text{(D.9)}$$

$$a_4 = 0.24308 * x^2 - 0.36446 * x + 1.01875 \quad \text{(D.10)}$$

$$a_5 = 45.87801 * x^2 - 35.50992 * x + 7.61841 \quad \text{(D.11)}$$

The value of x represents the reduction percentage in the bit rate, calculated using (6.5). The value of (D.6) ranges from 0 to 1, the higher the value, the better the quality. The relationship between the model coefficients, i.e., a_1 to a_5, and the feature x is plotted in Figure D.1.

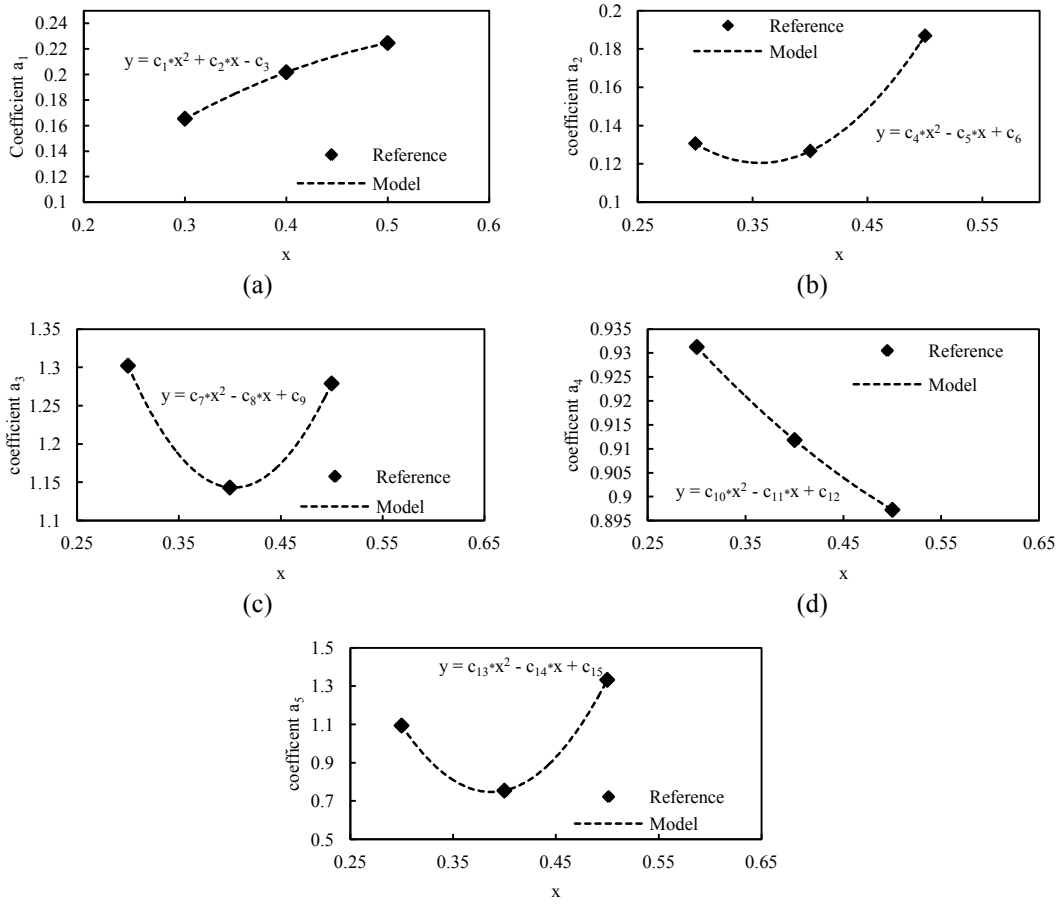


Figure D.1. The relationship between the model coefficients and the feature $\boldsymbol{x}$ for modeling the impact of reducing the bit rate on objective video quality for the MPEG-4 video codec. (a) a_1, (b) a_2, (c) a_3, (d) a_4, and (e) a_5.

Figure D.1 shows high correlation between the equations, which calculate the model coefficients a_1 to a_5 based on the values of x, and the model coefficients.

To test this model and the next one, i.e., for the FLV codec, I performed steps similar to those described in Chapter 6, but in this case to test modeling the impact of reducing the bit rate for the MPEG-4 and FLV codecs. I selected 30%, 40%, and 50% as reduction percentages for the bit rate. The total number of encoded videos generated from this encoding process for each codec is 28. During video transcoding, I kept the codec, frame rate, and frame size the same as in the original video. The only thing that I changed

in this transcoding was the bit rate. The total number of transcoded videos generated from this transcoding for each codec is 84.

Figure D.2 shows the quality evaluation results for modeling the impact of reducing the bit rate on objective video quality for the MPEG-4 codec for each video in the testing set and the reference values that represent the average of the actual quality values for all videos used in the training set in terms of MS-SSIM. Also, it shows the model results. Figure D.2 shows high correlations between the predicted quality values generated from this sub-model and the reference quality values. It also shows high correlations between the predicted quality results and the quality results generated from using MS-SSIM for each video in the testing set. Table D.1 shows high PCC values between the predicted quality values generated from this proposed sub-model and the actual quality values generated from using MS-SSIM for each video in the testing set. It also shows low RMSE values.

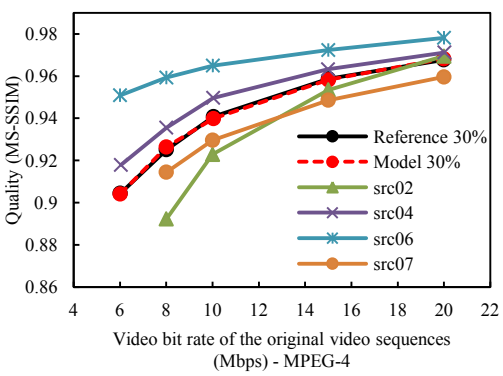
D.2.2. FLV Video Codec

I modeled the quality of the transcoded video, generated by reducing the bit rate from br_{in} to br_{out}, on objective video quality for the FLV codec as follows:

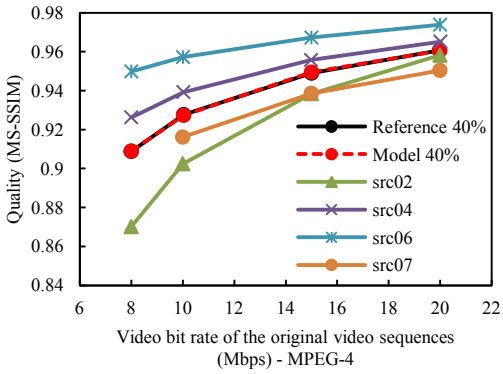
$$y.quality_{\beta_{FLV}(x.c.br_{in},y.c.br_{out}),x} = a_1 * \left(\frac{e^{(br_{out}*a_2)}}{e^{(br_{in}*a_3)}}\right) \quad \text{(D.12)}$$

Table D.1. The evaluation results in terms of PCC and RMSE for modeling the impact of reducing the bit rate on objective video quality for the MPEG-4 video codec.

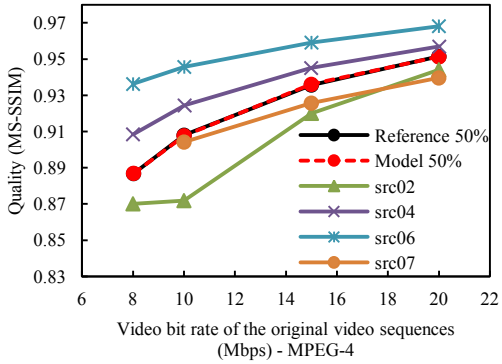
Seq. #	30% reduction		40% reduction		50% reduction	
	PCC	RMSE	PCC	RMSE	PCC	RMSE
vqeghd1_src02	0.99672	0.01916	0.99940	0.02407	0.99958	0.02648
vqeghd1_src04	0.99872	0.00884	0.99966	0.01461	0.99997	0.01470
vqeghd1_src06	0.99794	0.02899	0.99490	0.03789	0.99857	0.03435
vqeghd1_src07	0.99980	0.01012	0.99986	0.01061	0.99999	0.01041
Average	**0.99829**	**0.01678**	**0.99846**	**0.02180**	**0.99953**	**0.02149**



(a) 30% bit rate reduction



(b) 40% bit rate reduction



(c) 50% bit rate reduction

Figure D.2. The video quality evaluation results of modeling the impact of reducing the bit rate on objective video quality for the MPEG-4 video codec. These quality results are in terms of MS-SSIM for each video in the testing set and the reference values that represent the average of the actual video quality values for all the videos that are used in the training set at a given bit rate. It also shows the model results. The bit rate reduction percentages are (a) 30%, (b) 40%, and (c) 50% from the original bit rate.

In (D.12), x and y are the original and transcoded videos, respectively. The codec for videos x and y is FLV. The bit rate for video x is br_{in} and for video y is br_{out} in Mbps ($br_{in} > br_{out}$). The sub-model coefficients are a_1, a_2, and a_3, calculated as follows:

$$a_1 = -0.03448 * x^2 - 0.11390 * x + 0.96648 \quad \text{(D.13)}$$

$$a_2 = -0.07207 * x^2 + 0.11154 * x + 0.14674 \quad \text{(D.14)}$$

$$a_3 = -0.08722 * x^2 - 0.07835 * x + 0.15063 \quad \text{(D.15)}$$

The value of (D.12) ranges from 0 to 1, the higher the value, the better the quality. The relationship between the model coefficients, i.e., a_1 to a_3, and the feature x is plotted in Figure D.3, which shows high correlations between the equations, which calculate the model coefficients a_1 to a_3 based on the values of x, and the model coefficients.

Figure D.4 shows the quality evaluation results for modeling the impact of reducing the bit rate on objective video quality for the FLV codec for each video in the testing set and the reference values that represent the average of the actual quality values for all videos used in the training set in terms of MS-SSIM. Also, it shows the model results. Figure D.4 shows high correlations between the predicted quality values generated from this sub-model and the reference quality values. It also shows high correlations between the predicted quality results and the quality results generated from using MS-SSIM for each video in the testing set, except for "vqeghd1_src07" video, which shows inconsistent trend comparable with other videos and this inconsistency starts to increase when the reduction is increased. This inconsistency decreases the PCC values and increases the RMSE values.

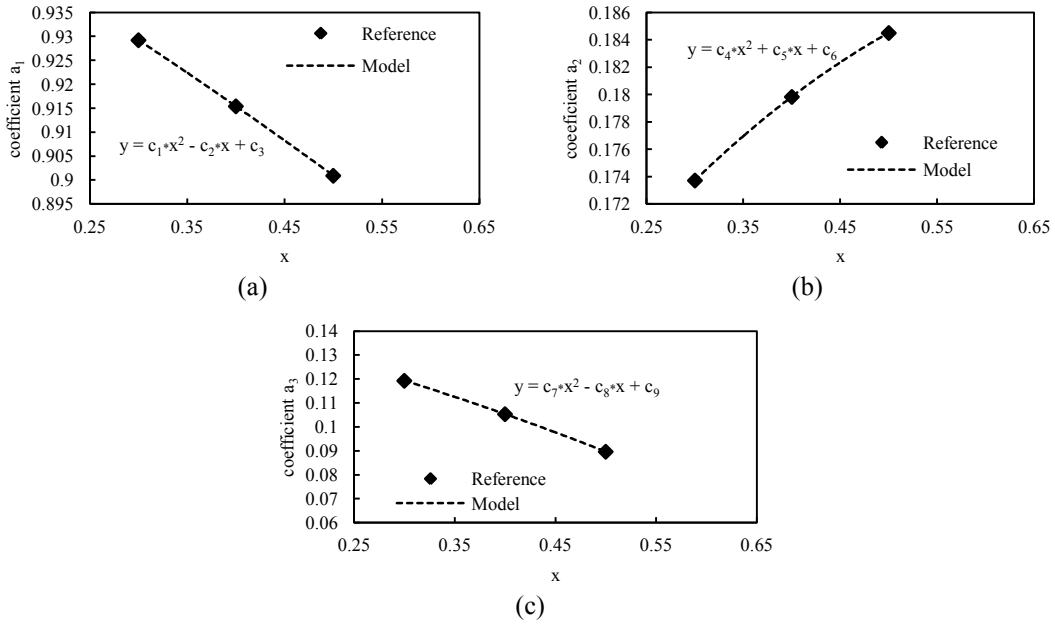
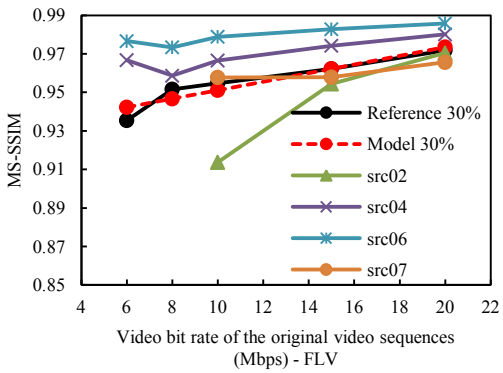
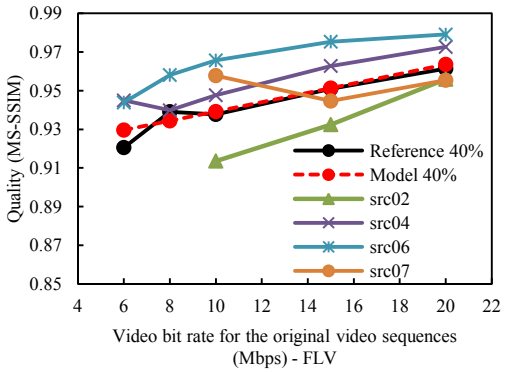


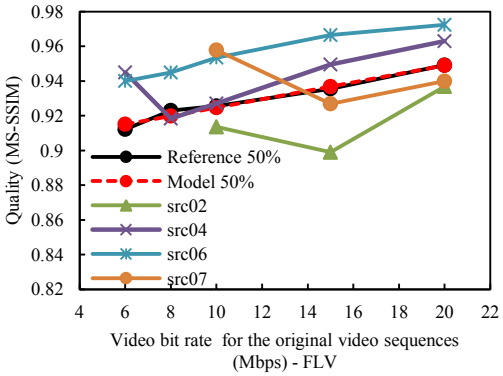
Figure D.3. The relationship between the model coefficients and the feature $\boldsymbol{x}$ for modeling the impact of reducing the bit rate on objective video quality for the FLV video codec. (a) a_1, (b) a_2, and (c) a_3.



(a) 30% bit rate reduction



(b) 40% bit rate reduction



(c) 50% bit rate reduction

Figure D.4. The video quality evaluation results of modeling the impact of reducing the bit rate on objective video quality for the FLV video codec. These quality results are in terms of MS-SSIM for each video in the testing set and the reference values that represent the average of the actual video quality values for all the videos that are used in the training set at a given bit rate. It also shows the model results. The bit rate reduction percentages are (a) 30%, (b) 40%, and (c) 50% from the original bit rate.

Table D.2 shows good PCC values between the predicted quality values generated from this proposed sub-model and the actual quality values generated from using MS-SSIM for each video in the testing set, with low RMSE values, except for "vqeghd1_src07" video.

Table D.2. The evaluation results in terms of PCC and RMSE for modeling the impact of reducing the bit rate on objective video quality for the FLV codec.

Seq. #	30% reduction		40% reduction		50% reduction	
	PCC	RMSE	PCC	RMSE	PCC	RMSE
vqeghd1_src02	0.96901	0.02215	0.99823	0.01888	0.612158	0.023812
vqeghd1_src04	0.88583	0.01525	0.96369	0.01047	0.730787	0.016282
vqeghd1_src06	0.92775	0.02543	0.92163	0.02136	0.980111	0.027191
vqeghd1_src07	0.87580	0.00649	-0.16490	0.01234	-0.57471	0.020648
Average	**0.91460**	**0.01733**	**0.67966**	**0.01576**	**0.437088**	**0.021983**

D.3. Modeling the Impact of Reducing the Frame Size

This section describes the rest of the quality sub-models that assess the impact of reducing the frame size on objective quality. These models are for the MPEG-4 and FLV codecs.

D.3.1. MPEG-4 Video Codec

I modeled the quality of the transcoded video, generated by reducing the frame size from fs_{in} to fs_{out}, on objective video quality for the MPEG-4 codec at a given bit rate, br, as follows:

$$y.quality_{\delta_{MPEG4}(x.c.br,x.c.fs_{in},y.c.fs_{out}),x} = 1 - \left[a_1 + \frac{\left(\frac{fs_{out}.w}{fs_{in}.w}\right)*(log_2(br+a_2)}{(\frac{fs_{out}.h}{fs_{in}.h})*(log_2(br)^{a_3*(e^{log_2(br+a_4)})})}\right] \quad \text{(D.16)}$$

where x and y are the original and transcoded videos, respectively. The codec for videos x and y is MPEG-4. The sub-model coefficients are a_1, a_2, a_3, and a_4, calculated as follows:

$$a_1 = 0.90584 * x^2 - 0.57133 * x + 0.10391 \quad \text{(D.17)}$$

$$a_2 = -152.96951 * x^2 + 108.39776 * x - 19.69767 \quad \text{(D.18)}$$

$$a_3 = 332.84962 * x^2 - 232.03934 * x + 41.55703 \quad \text{(D.19)}$$

$$a_4 = 127.68969 * x^2 - 110.93035 * x + 24.34879 \quad \text{(D.20)}$$

The value of x represents the relationship between the sizes of the frames, calculated using (6.11). The value of (D.16) ranges from 0 to 1, the higher the value, the better the quality. The relationship between the model coefficients, i.e., a_1 to a_4, and the feature x is plotted in Figure D.5, which shows high

correlations between the equations, which calculate the model coefficients a_1 to a_4 based on the values of x, and the model coefficients.

To test this model and the next one, i.e., for the FLV codec, I performed steps similar to those described in Chapter 6, but in this case to test modeling the impact of reducing the frame size for the MPEG-4 and FLV codecs. I selected the following new frame sizes: 1440x900, 1280x720, and 640x480. The total number of encoded videos generated from this encoding process is 28 for each codec. During this transcoding, I kept the codec, bit rate, and frame rate the same as in the original video. The only thing that I changed was the frame size. The total number of transcoded videos generated from this transcoding is 84 for each codec.

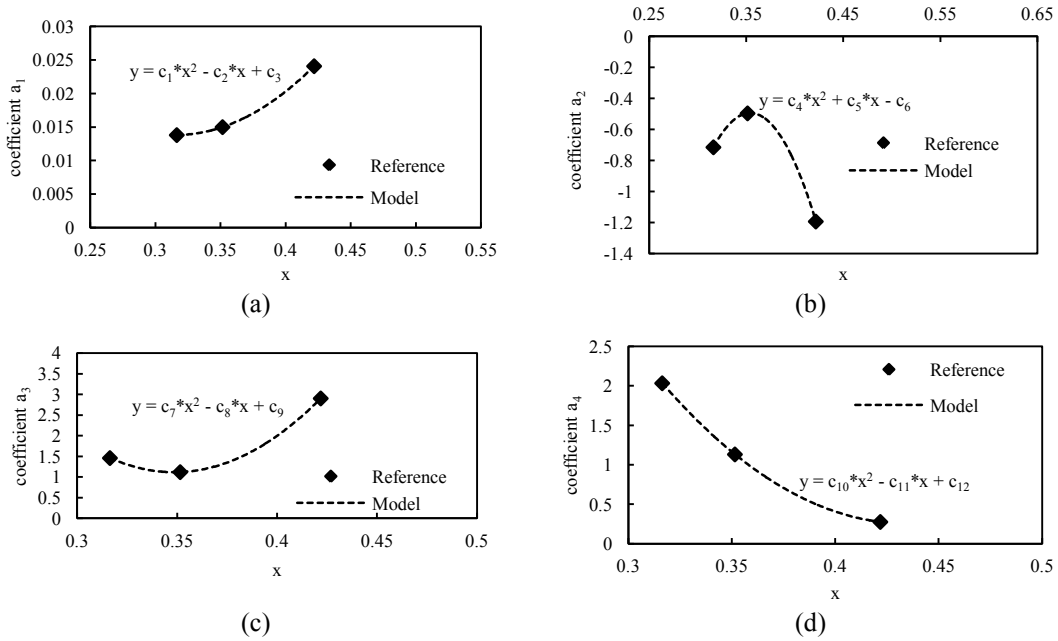
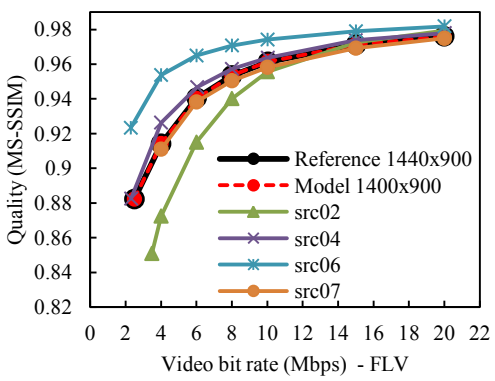


Figure D.5. The relationship between the model coefficients and the feature $\boldsymbol{x}$ for modeling the impact of reducing the frame size on objective video quality for the MPEG-4 video codec. (a) a_1, (b) a_2, (c) a_3, and (d) a_4.

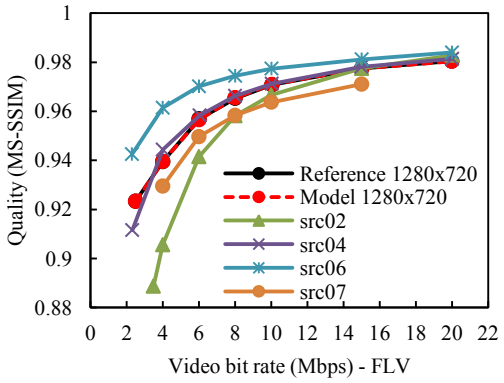
Figure D.6 shows the quality evaluation results for modeling the impact of reducing the frame size on objective video quality for the MPEG-4 codec for each video in the testing set and the reference values that represent the average of the actual quality values for all videos used in the training set in terms of MS-

SSIM. Also, it shows the model results. Figure D.6 shows high correlations between the predicted quality values generated from this sub-model and the reference quality values generated from averaging the actual quality values generated from using MS-SSIM for all the videos used in the training set. It also shows high correlations between the predicted quality results and the quality results generated from using MS-SSIM for each video in the testing set.

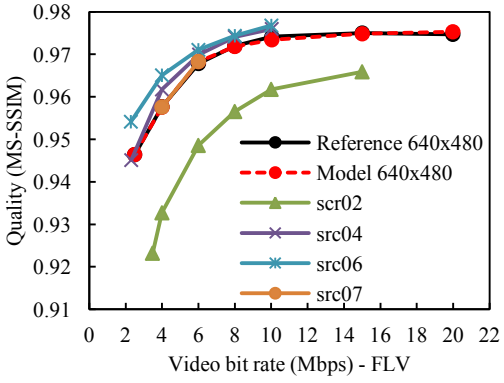
Table D.3 shows good PCC values between the predicted quality values generated from this proposed sub-model and the actual quality values generated from using MS-SSIM for each video in the testing set. It also shows low RMSE values.



(a) From 1920x1080 to 1440x900



(b) From 1920x1080 to 1280x720



(c) From 1920x1080 to 640x480

Figure D.6. The video quality evaluation results of modeling the impact of reducing the frame size on objective video quality for the MPEG-4 video codec. These quality results are in terms of MS-SSIM for each video in the testing set and the reference values that represent the average of the actual video quality values for all the videos that are used in the training set at a given bit rate. It also shows the model results. The new frame sizes are (a) 1440x900, (b) 1280x720, and (c) 640x480.

Table D.3. The evaluation results in terms of PCC and RMSE for modeling the impact of reducing the frame size on objective video quality for the MPEG-4 video codec.

Frame Size	1440x900		1280x720		640x480	
Seq. #	PCC	RMSE	PCC	RMSE	PCC	RMSE
vqeghd1_src02	0.99968	0.02781	0.99759	0.02215	0.99288	0.01946
vqeghd1_src04	0.99489	0.00563	0.98069	0.00534	0.87226	0.00631
vqeghd1_src06	0.98634	0.02573	0.98090	0.01266	0.83686	0.00488
vqeghd1_src07	0.99941	0.00257	0.99914	0.00763	1.00000	0.01861
Average	**0.99508**	**0.01544**	**0.98958**	**0.01194**	**0.92550**	**0.01232**

D.3.2. FLV Video Codec

I modeled the quality of the transcoded video, generated by reducing the frame size from fs_{in} to fs_{out}, on objective video quality for the FLV codec at a given bit rate, br, as follows:

$$y.quality_{\delta_{FLV}(x.c.br,x.c.fs_{in},y.c.fs_{out}),x} = 1 - \left[a_1 + \frac{\frac{fs_{out}.w}{fs_{in}.w} * (log_2(br+a_2))}{\frac{fs_{out}.h}{fs_{in}.h} * (log_2(br)^{a_3}) * (e^{log_2(br+a_4)})}\right] \quad \text{(D.21)}$$

where x and y are the original and transcoded videos, respectively. The codec for videos x and y is FLV. The sub-model coefficients are a_1, a_2, a_3, and a_4, calculated as follows:

$$a_1 = -4.5294 * x^2 + 3.66191 * x - 0.71518 \quad \text{(D.22)}$$

$$a_2 = 1177.39952 * x^2 - 916.0712 * x + 176.1566 \quad \text{(D.23)}$$

$$a_3 = 43.79609 * x^2 - 8.17365 * x - 1.54095 \quad \text{(D.24)}$$

$$a_4 = 1233.11183 * x^2 - 951.96648 * x + 183.49663 \quad \text{(D.25)}$$

The value of (D.21) ranges from 0 to 1, the higher the value, the better the quality. The relationship between the model coefficients, i.e., a_1 to a_4, and the feature x is plotted in Figure D.7, which shows high correlations between the equations, which calculate the model coefficients a_1 to a_4 based on the values of x, and the model coefficients.

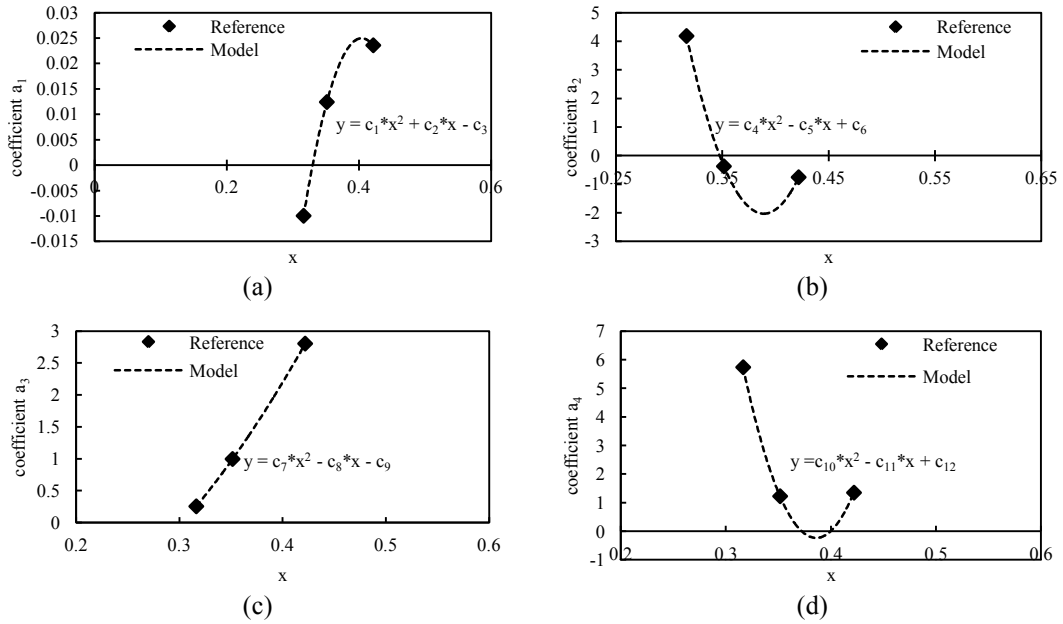
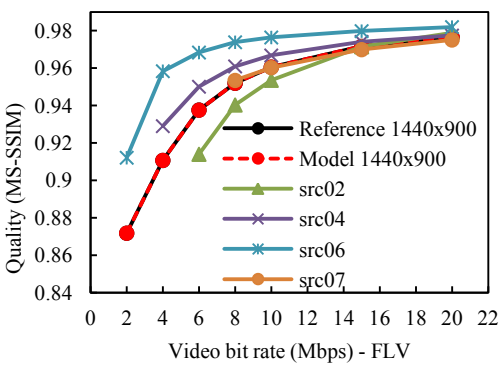


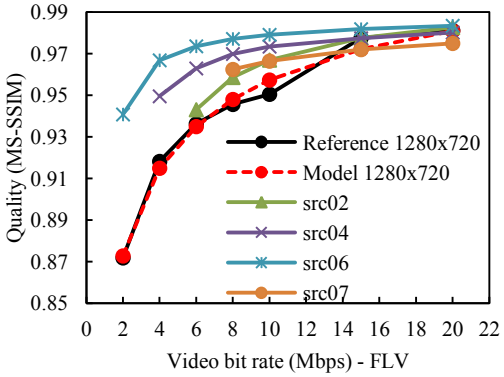
Figure D.7. The relationship between the model coefficients and the feature x for modeling the impact of reducing the frame size on objective video quality for the FLV video codec. (a) a_1, (b) a_2, (c) a_3, and (d) a_4.

Figure D.8 shows the quality evaluation results for modeling the impact of reducing the frame size on objective video quality for the FLV codec for each video in the testing set and the reference values that represent the average of the actual quality values for all videos used in the training set in terms of MS-SSIM. Also, it shows the model results.

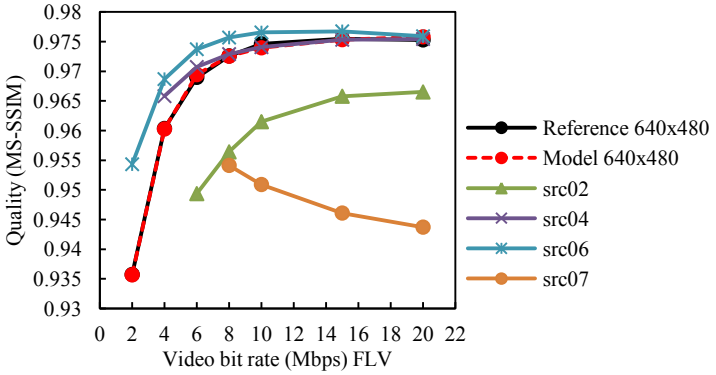
Figure D.8 shows high correlations between the predicted quality values generated from this sub-model and the reference quality values generated from averaging the actual quality values generated from using MS-SSIM for all the videos used in the training set. It also shows high correlations between the predicted quality results and the quality results generated from using MS-SSIM for each video in the testing set. Table D.4 shows good PCC values between the predicted quality values generated from this proposed sub-model and the actual quality values generated from using MS-SSIM for each video in the testing set. It also shows low RMSE values.



(a) From 1920x1080 to 1440x900



(b) From 1920x1080 to 1280x720



(c) From 1920x1080 to 640x480

Figure D.8. The video quality evaluation results of modeling the impact of reducing the frame size on objective video quality for the FLV video codec. These quality results are in terms of MS-SSIM for each video in the testing set and the reference values that represent the average of the actual video quality values for all the videos that are used in the training set at a given bit rate. It also shows the model results. The new frame sizes are (a) 1440x900, (b) 1280x720, and (c) 640x480.

D.4. Modeling the Impact of Reducing the Frame Rate

This section describes the rest of the quality sub-models that assess the impact of reducing the frame rate on objective video quality. These models are for the MPEG-4 and FLV codecs.

D.4.1. MPEG-4 Video Codec

I modeled the quality of the transcoded video, generated by reducing the frame rate from fr_{in} to fr_{out}, on objective video quality for the MPEG-4 codec at a given bit rate, br, as follows:

Table D.4. The evaluation results in terms of PCC and RMSE for modeling the impact of reducing the frame size on objective video quality for the FLV video codec

	1440x900		1280x720		640x480	
Seq. #	PCC	RMSE	PCC	RMSE	PCC	RMSE
vqeghd1_src02	0.99914	0.01221	0.98991	0.00775	0.99567	0.01411
vqeghd1_src04	0.99917	0.01015	0.97985	0.02138	0.99543	0.00233
vqeghd1_src06	0.95314	0.02847	0.96488	0.03822	0.99743	0.00806
vqeghd1_src07	0.99914	0.00140	0.99911	0.00903	-0.99074	0.02624
Average	**0.98765**	**0.01306**	**0.98344**	**0.01910**	**0.49945**	**0.01268**

$$y.quality_{Y(x.c.fr_{in},y.c.fr_{out}),x} = \frac{\left(\frac{y.fr_{out}}{x.fr_{in}}\right) + \log_2(br+a_1) + a_2}{a_3 * \ln(log_2(br+a_4)+br^{a_5}} \quad \text{(D.26)}$$

where x and y are the original and transcoded videos, respectively. The codec for videos x and y is MPEG-4. The model coefficients are a_1 to a_5, calculated as follows:

$$a_1 = 200.28718 * x^2 - 301.82862 * x + 112.94118 \quad \text{(D.27)}$$

$$a_2 = -9.23642 * x^2 + 14.01832 * x - 5.70813 \quad \text{(D.28)}$$

$$a_3 = -8.76139 * x^2 + 11.89591 * x - 3.19208 \quad \text{(D.29)}$$

$$a_4 = 638.80911 - 959.49887 * x + 359.24664 \quad \text{(D.30)}$$

$$a_5 = 0.0504 * x^2 - 0.1429 * x + 0.26 \quad \text{(D.31)}$$

The value of x represents the relationship between the two frame rates, calculated using (6.16). The value of (D.26) ranges from 0 to 1, the higher the value, the better the quality. The relationship between the model coefficients, i.e., a_1 to a_5, and the feature x is plotted in Figure D.9, which shows high correlations between the equations, which calculate the model coefficients a_1 to a_5 based on the values of x, and the model coefficients.

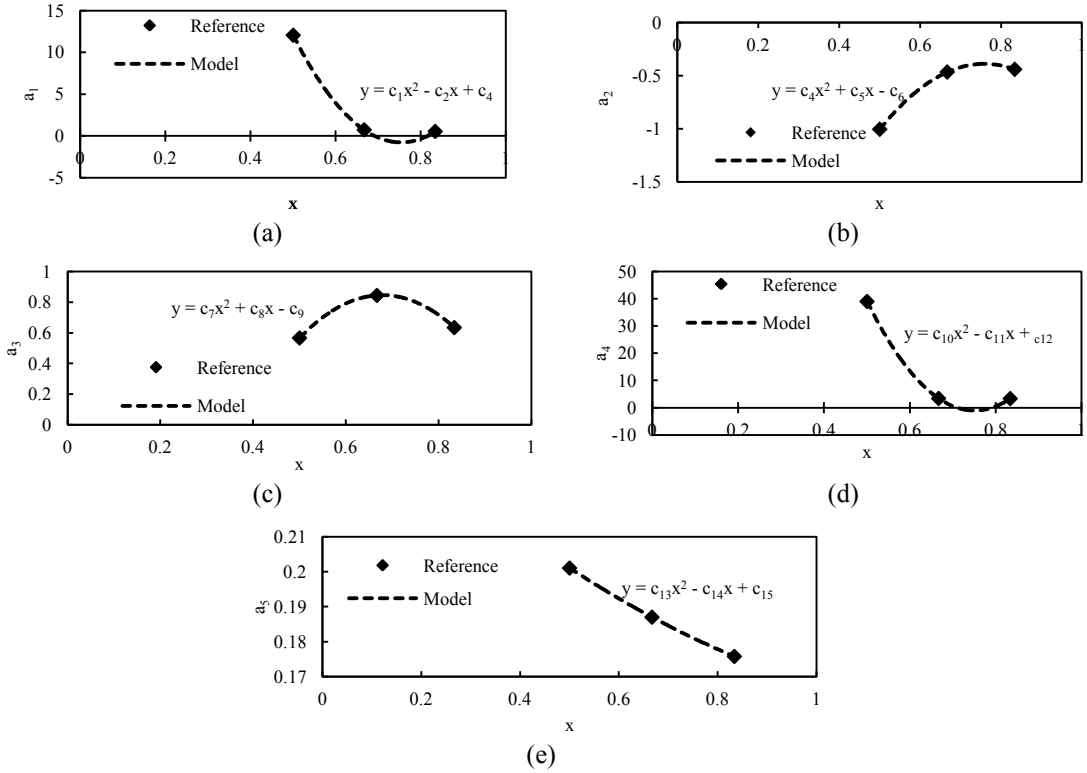


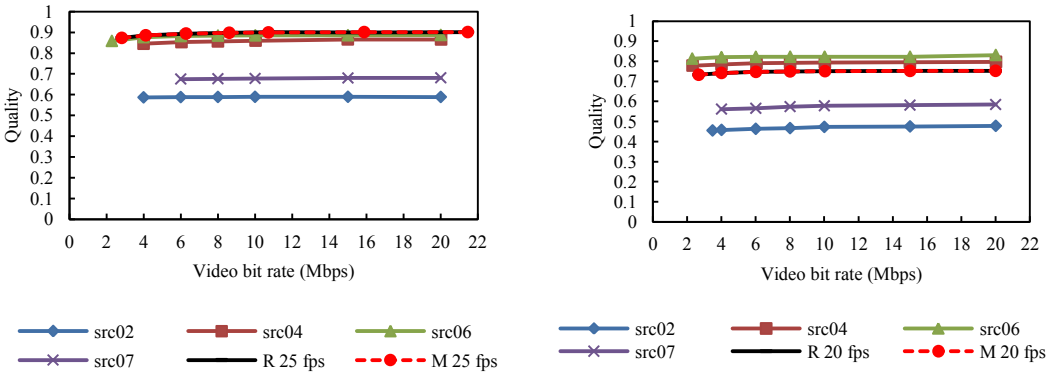
Figure D.9. The relationship between the model coefficients and the feature $\boldsymbol{x}$ for modeling the impact of reducing the frame rate on objective video quality for the MPEG-4 video codec. (a) a_1, (b) a_2, (c) a_3, (d) a_4 and (e) a_5.

To test this model and the next one, i.e., for the FLV codec, I performed steps similar to those described in Chapter 6, but in this case to test modeling the impact of reducing the frame rate for the MPEG-4 and FLV codecs. I selected the following new frame rates: 25 fps, 20 fps, and 15 fps. The total number of encoded videos generated from this encoding process is 28 for each codec. During this transcoding, I kept the codec, bit rate, and frame size the same as in the original video. The only thing that I changed was the frame rate. The total number of transcoded videos generated from this transcoding is 84 for each codec.

Figure D.10 shows the quality evaluation results for modeling the impact of reducing the frame rate on objective video quality for the MPEG-4 codec for each video in the testing set and the reference values (labeled by "R") that represent the average of the actual quality values for all the videos used in the training set in terms of MS-SSIM. It also shows the model results (labeled by "M"). The frame rate

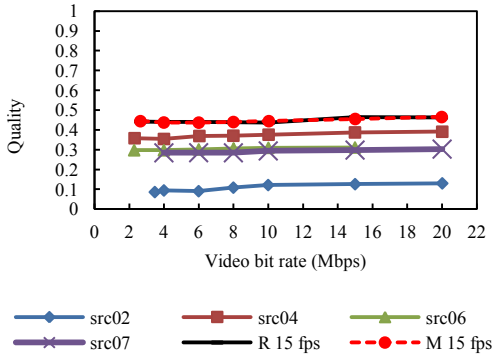
reductions are from 29.97 fps to 25 fps, 20 fps, and 15 fps for all the videos that are used in the training and testing sets.

Figure D.10(a) shows that two videos from the testing set, i.e., src04 and src06, have results that are close to the model results. The other two videos, i.e., src02 and src07, have results that are slightly farther from the model results. However, all of the videos that are used in the testing set have high correlations with the model results.



(a) From 29.97 fps to 25 fps

(b) From 29.97 fps 20 fps



(c) From 29.97 fps 15 fps

Figure D.10. The video quality evaluation results of modeling the impact of reducing the frame rate on objective video quality for the MPEG-4 video codec. These quality results are in terms of MS-SSIM for each video in the testing set and the reference values that represent the average of the actual video quality values for all the videos used in the training set at a given bit rate. It also shows the model results. The new frame rates are (a) 25 fps, (b) 20 fps, and (c) 15 fps.

Figure D.10(b) shows that two videos from the testing set, i.e., src04 and src06, have results that are close to the model results, whereas the other two videos, i.e., src02 and src07, have results that are slightly farther from the model results. Figure D.10(c) shows that three videos from the testing set, i.e., src04, src06, and src07, have results that are close to the model results, while src02 has results that are slightly farther from the model results. However, the quality values of three videos used in the testing set have high correlations with the predicted quality results generated from this model with low RMSE values, as shown in Table D.5.

D.4.2. FLV Video Codec

I modeled the quality of the transcoded video, generated by reducing the frame rate from fr_{in} to fr_{out}, on objective video quality for the FLV codec at a given bit rate br as follows:

$$y.quality_{\gamma(x.c.fr_{in},y.c.fr_{out}),x} = (\log_{10}\left(\frac{fr_{out}}{fr_{in}}\right) + \mathrm{a}_1) - (a_2 + \frac{log_2(br+a_3)}{e^{log_2(br+a_4)}*br^{a_5}}) \quad \text{(D.32)}$$

where x and y are the original and transcoded videos, respectively. The codec for videos x and y is FLV. The sub-model coefficients are a_1 to a_5, calculated as follows:

$$a_1 = 9.4292 * x^2 - 11.838 * x + 5.547 \quad \text{(D.33)}$$

$$a_2 = 1.9068 * x^2 - 2.3529 * x + 0.2168 \quad \text{(D.34)}$$

$$a_3 = 18.325 * x^2 - 23.442 * x + 6.7765 \quad \text{(D.35)}$$

$$a_4 = 99.53 * x^2 - 124.36 * x + 39.268 \quad \text{(D.36)}$$

$$a_5 = -8.737 * x^2 + 8.9851 * x - 0.3799 \quad \text{(D.37)}$$

Table D.5. The evaluation results in terms of PCC and RMSE for modeling the impact of reducing the frame rate on objective video quality for the MPEG-4video codec.

Seq. #	From 29.97 to 25 fps		From 29.97 to 20 fps		From 29.97 to 15 fps	
	PCC	RMSE	PCC	RMSE	PCC	RMSE
vqeghd1_src02	0.93969	0.30753	0.96088	0.28051	0.84790	0.33733
vqeghd1_src04	0.96605	0.03857	0.97656	0.04450	0.77092	0.07461
vqeghd1_src06	0.99586	0.01190	0.83628	0.07619	0.49163	0.13985
vqeghd1_src07	0.94884	0.22006	0.94527	0.17483	0.95955	0.15493
Average	**0.96261**	**0.14451**	**0.92975**	**0.14401**	**0.76750**	**0.17668**

The value of (D.32) ranges from 0 to 1, the higher the value, the better the quality. The relationship between the model coefficients, i.e., a_1 to a_5, and the feature x is plotted in Figure D.11, which shows high correlations between the equations, which calculate the model coefficients a_1 to a_5 based on the values of x, and the model coefficients.

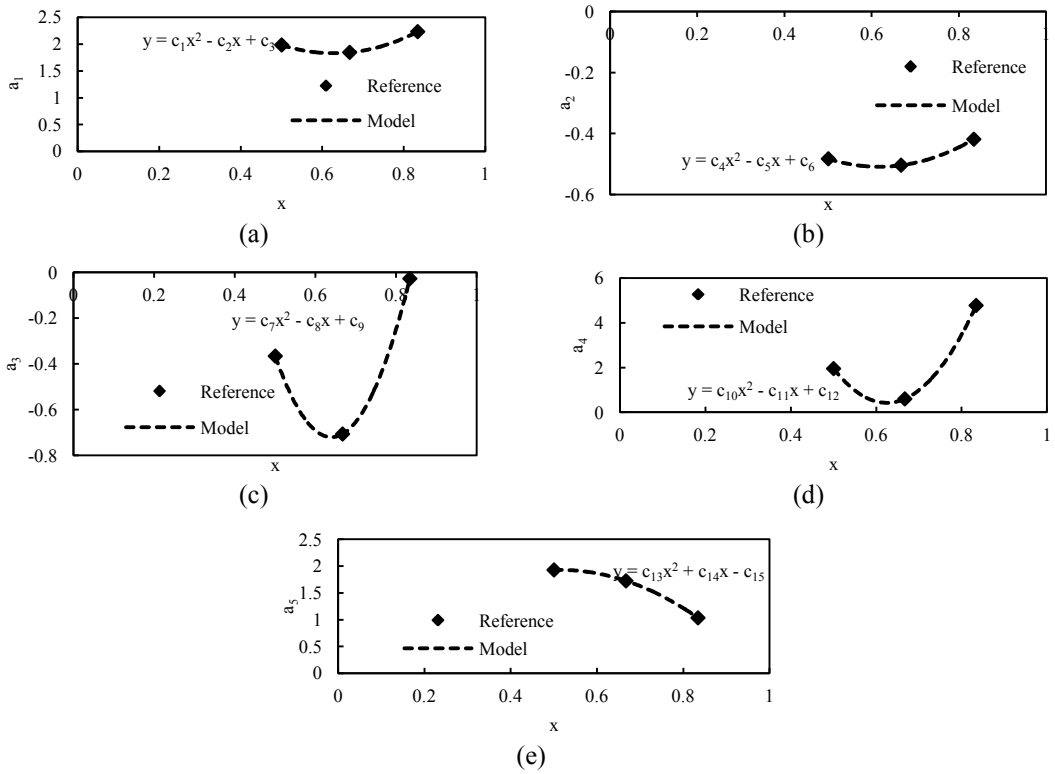
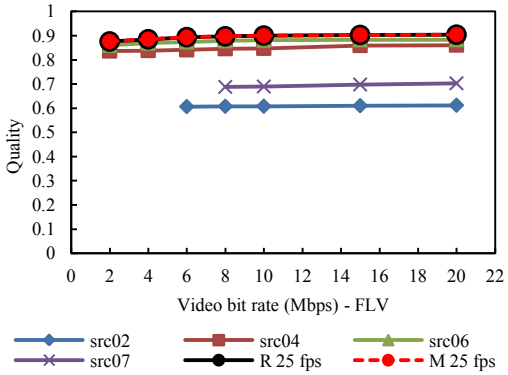
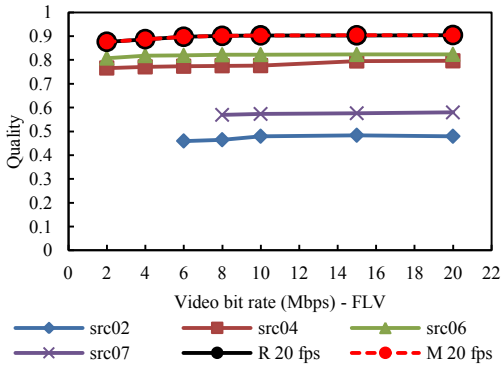


Figure D.11. The relationship between the model coefficients and the feature x for modeling the impact of reducing the frame rate on objective video quality for the FLV video codec. (a) a_1, (b) a_2, (c) a_3, (d) a_4 and (e) a_5.

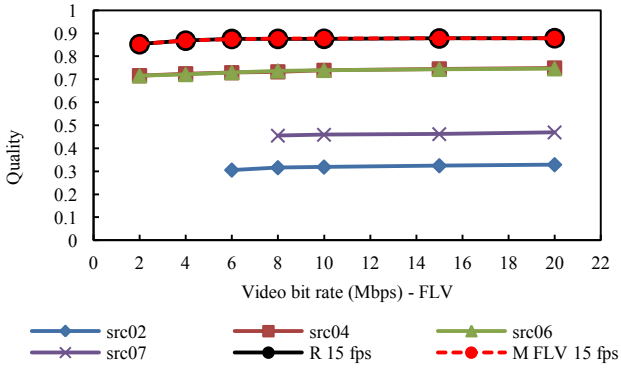
Figure D.12 shows the quality evaluation results for modeling the impact of reducing the frame size on objective video quality for the FLV codec for each video in the testing set and the reference values that represent the average of the actual quality values for all videos used in the training set in terms of MS-SSIM. Also, it shows the model results.



(a) from 29.97 to 25 fps



(b) from 29.97 to 20 fps



(c) from 29.97 to 15 fps

Figure D.12. The video quality evaluation results for modeling the impact of reducing the frame rate on objective video quality for the FLV video codec. These quality results are in terms of MS-SSIM for each video in the testing set and the reference values that represent the average of the actual video quality values for all the videos used in the training set at a given bit rate. It also shows the model results. The new frame rates are (a) 25 fps, (b) 20 fps, and (c) 15 fps.

Figure D.12 shows good correlations between the predicted quality values generated from this sub-model and the reference quality values generated from averaging the quality results generated from using MS-SSIM for each video in the training set. It also shows good correlation between the predicted quality values and the quality results generated from using MS-SSIM for each video in the testing set. Figure D.12 also shows that two videos from the testing set, i.e., src04 and src06, have results that are close to the model results. The other two videos, i.e., src02 and src07, have results that are slightly farther from

the model results. However, all the videos that are used in the testing set have high correlation results with the model results, as shown in Table D.6.

Table D.6. The evaluation results in terms of PCC and RMSE for modeling the impact of reducing the frame rate on objective video quality for the FLV video codec.

Seq. #	From 29.97 to 25 fps		From 29.97 to 20 fps		From 29.97 to 15 fps	
	PCC	RMSE	PCC	RMSE	PCC	RMSE
vqeghd1_src02	0.92641	0.29037	0.92464	0.39128	0.97770	0.55826
vqeghd1_src04	0.87602	0.04708	0.73457	0.11731	0.84779	0.13862
vqeghd1_src06	0.99720	0.01850	0.95845	0.07686	0.88488	0.13943
vqeghd1_src07	0.96158	0.20645	0.96875	0.32866	0.89348	0.41620
Average	**0.94030**	**0.14060**	**0.89660**	**0.22853**	**0.90096**	**0.31313**

D.5. Error Sub-Models for Reducing the Bit Rate

This section describes the error sub-models that assess the error in predicting the impact of reducing the bit rate on objective video quality. These sub-models are for the H.264, MPEG-4, and FLV codecs.

D.5.1. H.264 and MPEG-4 Video Codecs

I modeled the error in assessing the impact of reducing the bit rate from br_1 to br_2 on objective video quality for the H.264 and MPEG-4 codecs as follows:

$$Error_f^{\beta}(x.c.br_1, y.c.br_2) = a_1 + \left(\frac{log_2(br_2+a_2)*a_3}{log_2(br_1+a_4)}\right) \quad \text{(D.38)}$$

where $f \in \{H264, MPEG4\}$ and a_1 to a_4 are the sub-model coefficients, calculated for the H.264 codec as follows:

$$a_1 = 1.17442 * x^2 - 0.80705x + 0.17973 \quad \text{(D.39)}$$

$$a_2 = 48.287 * x^2 - 36.115 * x + 6.48031 \quad \text{(D.40)}$$

$$a_3 = -1.4976 * x^2 + 1.0039 * x - 0.2162 \quad \text{(D.41)}$$

$$a_4 = 78.62853 * x^2 - 75.62253 * x + 20.84327 \quad \text{(D.42)}$$

The value of x represents the relationship between the two bit rates, calculated using (6.5). For the MPEG-4 codec, the sub-model coefficients are calculated as follows:

$$a_1 = -3.2977 * x^2 + 2.461 * x - 0.3766 \quad \text{(D.43)}$$

$$a_2 = -105.49186 * x^2 + 72.1984 * x - 13.37489 \quad \text{(D.44)}$$

$$a_3 = 4.325 * x^2 - 3.2479 * x + 0.51 \quad \text{(D.45)}$$

$$a_4 = 139.26 * x^2 - 116.18 * x + 27.109 \quad \text{(D.46)}$$

D.5.2. FLV Video Codec

I modeled the error in assessing the impact of reducing the bit rate from br_1 to br_2 on objective video quality for the FLV codec as follows:

$$Error_{FLV}^{\beta}(x.c.br_1, y.c.br_2) = a_1 * \left(\frac{e^{(br_2 * a_2)}}{e^{(br_1 * a_3)}}\right) \quad \text{(D.47)}$$

where a_1 to a_3 are the sub-model coefficients, calculated as follows:

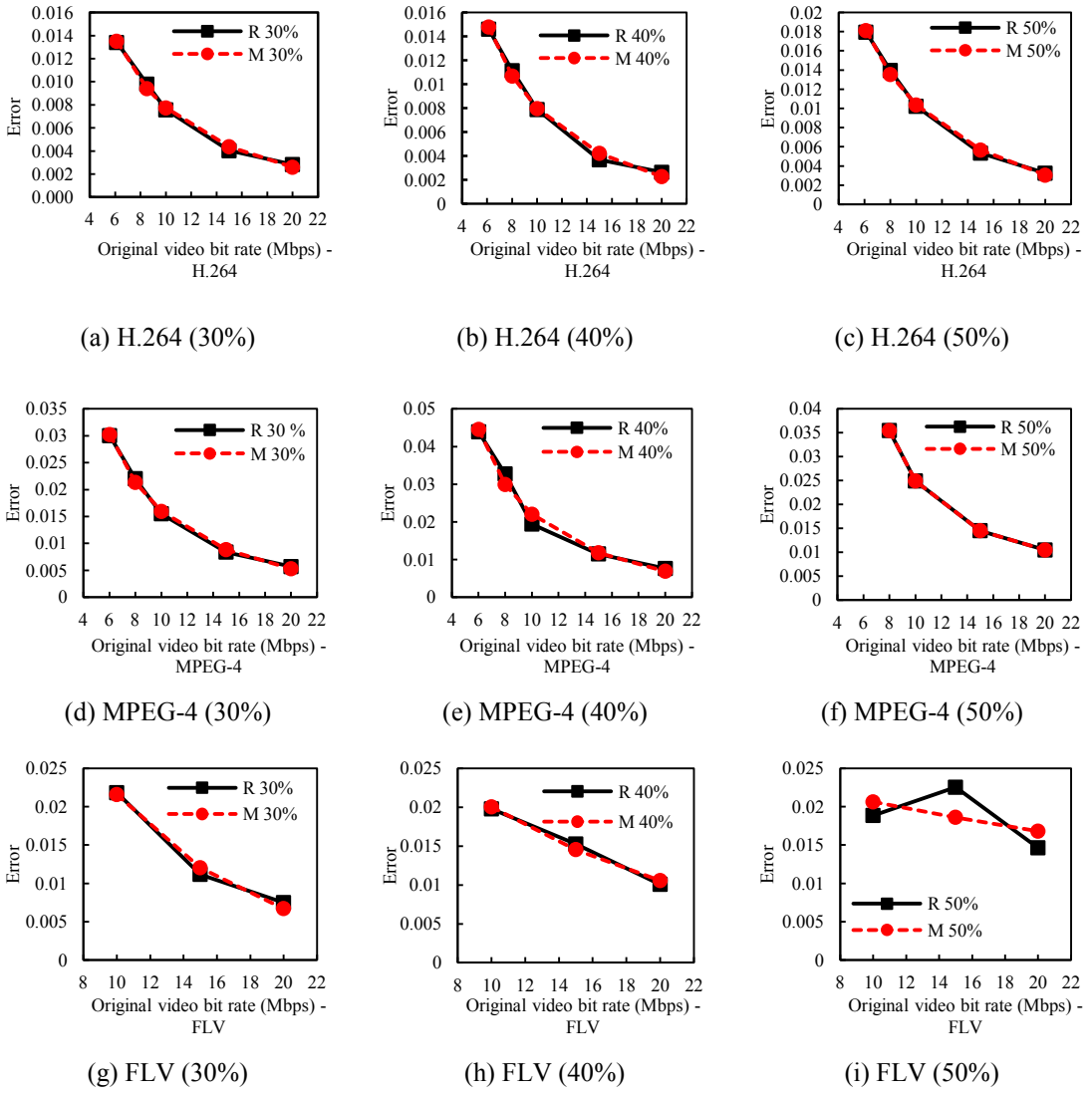
$$a_1 = 0.9333 * x^2 - 0.967 * x + 0.2755 \quad \text{(D.48)}$$

$$a_2 = -112.27882 * x^2 + 79.6291647 * x - 11.56980285 \quad \text{(D.49)}$$

$$a_3 = -56.711 * x^2 + 37.576 * x - 4.5025 \quad \text{(D.50)}$$

Figure D.13 shows the error values that represent the error in assessing the impact of reducing the bit rate on objective video quality for the H.264, MPEG-4, and FLV codecs at a given bit rate. The expected error values are generated from the proposed error sub-models (labeled by "M"). The reference error values (labeled by "R") represent the average of the difference between the actual quality values generated from MS-SSIM and the predicted quality values generated from the proposed bit rate quality sub-models described above at a given bit rate. The bit rate reduction percentages are 30%, 40%, and 50% from the original bit rate for all these codecs. Figure D.13 shows high correlation between these reference and expected error values.

Table D.7 shows the evaluation results in terms of PCC and RMSE for assessing the error in modeling the impact of reducing the bit rate on objective video quality for the H.264, MPEG-4, and FLV codecs. These evaluation results are generated by comparing the reference and expected error values, which show high PCC and very low RMSE values.



(a) H.264 (30%) (b) H.264 (40%) (c) H.264 (50%)

(d) MPEG-4 (30%) (e) MPEG-4 (40%) (f) MPEG-4 (50%)

(g) FLV (30%) (h) FLV (40%) (i) FLV (50%)

Figure D.13. The error values that represent the error in assessing the impact of reducing the bit rate on objective video quality for the H.264, MPEG-4, and FLV video codecs at a given bit rate. The black curves (labeled by "R") represent the reference error values and the red curves (labeled by "M") represent the predicted error values generated from the proposed error sub-models. The reference error values represent the average of the differences between the actual video quality values generated from MS-SSIM and the predicted video quality values generated from the proposed bit rate quality sub-models. The bit rate reduction percentages are 30%, 40%, and 50% from the original bit rate for the (a) to (c) H.264, (d) to (f) MPEG-4, and (g) to (i) FLV codecs.

Table D.7. The evaluation results in terms of PCC and RMSE for assessing the error in modeling the impact of reducing the bit rate on objective video quality for the H.264, MPEG-4, and FLV codecs.

Reduction Percentage in the bit rate	H.264 sub-model		MPEG-4 sub-model		FLV sub-model	
	PCC	RMSE	PCC	RMSE	PCC	RMSE
30%	0.99754	0.00027	0.99864	0.00046	0.99383	0.00068
40%	0.99674	0.00036	0.99142	0.00177	0.99143	0.00052
50%	0.99875	0.00027	0.99994	0.00011	0.50895	0.00278

D.6. Error Sub-Models for Reducing the Frame Rate

This section describes the error sub-models that assess the error in predicting the impact of reducing the frame rate on objective video quality. These models are for the H.264, MPEG-4, and FLV codecs.

D.6.1. H.264 Video Codec

I modeled the error in assessing the impact of reducing the frame rate from fr_1 to fr_2 on objective video quality for the H.264 codec at a given bit rate, br, as follows:

$$Error^{\gamma}_{H264}(\,x.c.br, x.c.fr_1, y.c.fr_2) = \frac{(fr_2/fr_1)*br*a_1}{a_2*(br^{a_3})} \quad \text{(D.51)}$$

where a_1 to a_3 are the sub-model coefficients, calculated as follows:

$$a_1 = -8.96860 * x^2 + 13.52600 * x - 5.9039 \quad \text{(D.52)}$$

$$a_2 = -28.516 * x^2 + 25.842 * x - 6.7921 \quad \text{(D.53)}$$

$$a_3 = 0.24085 * x^2 - 0.32317 * x + 1.12076 \quad \text{(D.54)}$$

The value of x represents the relationship between the two frame rates, calculated using (6.16).

D.6.2. MPEG-4 Video Codec

I modeled the error in assessing the impact of reducing the frame rate from fr_1 to fr_2 on objective video quality for the MPEG-4 codec at a given bit rate, br, as follows:

$$Error^{\gamma}_{MPEG4}(\,x.c.br, x.c.fr_1, y.c.fr_2) = \frac{(fr_2/fr_1)+log_{10}(br+a_1)+a_2}{(a_3*\ln(log_{10}(br+a_4))+br^{a_5})} \quad \text{(D.55)}$$

where a_1 to a_5 are the sub-model coefficients, calculated as follows:

$$a_1 = 905.09206 * x^2 - 1378.07499 * x + 519.78394 \quad \text{(D.56)}$$

$$a_2 = -18.40403 * x^2 + 26.808659 * x - 10.95724 \quad \text{(D.57)}$$

$$a_3 = 26.911 - 12.064 * x - 2.0045 \quad \text{(D.58)}$$

$$a_4 = 615.071 * x^2 - 929.1683 * x + 349.6824 \quad \text{(D.59)}$$

$$a_5 = -8.2925 * x^2 + 11.485 * x - 3.4392 \quad \text{(D.60)}$$

D.6.3. FLV Video Codec

I modeled the error in assessing the impact of reducing the frame rate from fr_1 to fr_2 on objective video quality for the FLV codec at a given bit rate, br, as follows:

$$Error_{FLV}^{\gamma}(x.c.br, x.c.fr_1, y.c.fr_2) = \frac{\left(log_2((fr_2/fr_1)+a_1)*log_2(br+a_2)\right)*a_3}{\left(e^{log_2(br+a_4)}*br^{a_5}\right)*a_6} \quad \text{(D.61)}$$

where a_1 to a_6 are the sub-model coefficients, calculated as follows:

$$a_1 = 10.96238 - 19.43307 * x + 10.86636 \quad \text{(D.62)}$$

$$a_2 = -7.82795 * x^2 + 6.582759 * x - 0.655307 \quad \text{(D.63)}$$

$$a_3 = 104.8019 * x^2 - 146.298 * x + 51.24557 \quad \text{(D.64)}$$

$$a_4 = -4.58816 * x^2 + 0.685149 * x + 11.24952 \quad \text{(D.65)}$$

$$a_5 = -0.59 * x^2 + 0.4353 * x - 0.5717 \quad \text{(D.66)}$$

$$a_6 = 154.028 * x^2 - 202.609 * x + 66.96729 \quad \text{(D.67)}$$

Figure D.14 shows the error values that represent the error in assessing the impact of reducing the frame rate on objective video quality for the H.264, MPEG-4, and FLV codecs at a given bit rate. The expected error values are generated from the proposed error sub-models (labeled by "M"). The reference error values (labeled by "R") represent the average of the difference between the actual quality values generated from MS-SSIM and the predicted quality values generated from the proposed frame rate quality sub-models described above at a given bit rate. The frame rates are reduced from the original frame rate, i.e., 29.97 fps, to 25 fps, 20 fps, and 15 fps for all these codecs. Figure D.14 shows high correlation between these reference and expected error values.

Table D.8 shows the evaluation results in terms of PCC and RMSE for assessing the error in modeling the impact of reducing the frame rate on objective video quality for the H.264, MPEG-4, and

FLV codecs. These evaluation results are generated by comparing the reference and expected error values, which show high PCC and very low RMSE values.

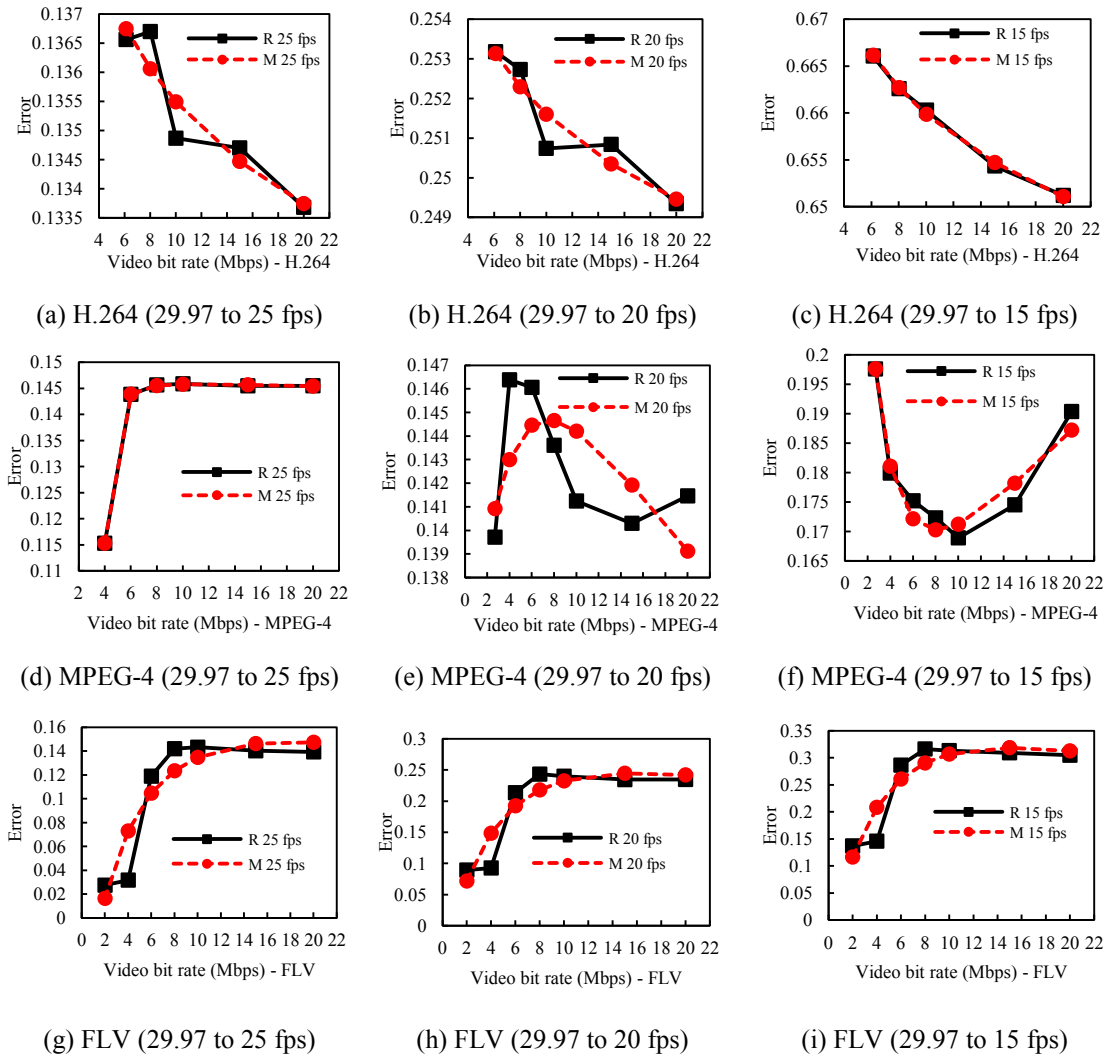


Figure D.14. The error values that represent the error in assessing the impact of reducing the frame rate on objective video quality for the H.264, MPEG-4, and FLV video codecs at a given bit rate. The black curves (labeled by "R") represent the reference error values and the red curves (labeled by "M") represent the predicted error values generated from the proposed error sub-models. The reference error values represent the average of the differences between the actual video quality values generated from MS-SSIM and the predicted video quality values generated from the proposed frame rate quality sub-models. The frame rate reductions are from 29.97 fps to 25 fps, from 29.97 fps to 20 fps, and from 29.97 fps to 15 fps for the (a) to (c) H.264, (d) to (f) MPEG-4, and (g) to (i) FLV codecs.

Table D.8. The evaluation results in terms of PCC and RMSE for assessing the error in modeling the impact of reducing the frame rate on objective video quality for the H.264, MPEG-4, and FLV video codecs.

fr_{from}	fr_{to}	H.264 sub-model		MPEG-4 sub-model		FLV sub-model	
		PCC	RMSE	PCC	RMSE	PCC	RMSE
29.97	25	0.93132	0.00042	0.99995	0.00011	0.92163	0.01906
29.97	20	0.93801	0.00049	0.54007	0.00219	0.91640	0.02596
29.97	15	0.99874	0.00027	0.96610	0.00250	0.92235	0.02895

D.7. Error Sub-Models for Reducing the Frame Size

This section describes the error sub-models that assess the error in modeling the impact of reducing the frame size on objective video quality. These models are for the H.264, MPEG-4, and FLV codecs.

D.7.1. H.264 Video Codec

I modeled the error in assessing the impact of reducing the frame size from fs_1 to fs_2 on objective video quality for the H.264 codec at a given bit rate, br, as follows:

$$Error^{\delta}_{H264}(x.c.br, x.c.fs_1, y.c.fs_2) = \frac{log_2(fs_2.w+a_1)*log_2(fs_2.h+a_2)*\sqrt{br}*a_3}{log_2(fs_1.w+a_4)*log_2(fs_1.h+a_5)*br^{a_6}} \quad \text{(D.68)}$$

where a_1 to a_6 are the sub-model coefficients, calculated as follows:

$$a_1 = 10.96238 * x^2 + 19.43307 * x - 10.86636 \quad \text{(D.69)}$$

$$a_2 = -7.82795 * x^2 + 6.582759 * x - 0.655307 \quad \text{(D.70)}$$

$$a_3 = 104.8019 * x^2 + 146.298 * x - 51.24557 \quad \text{(D.71)}$$

$$a_4 = -4.58816_{10} * x^2 + 0.685149 * x - 11.24952 \quad \text{(D.72)}$$

$$a_5 = -0.59 * x^2 - 0.4353 * x + 0.5717 \quad \text{(D.73)}$$

$$a_6 = 154.028 * x^2 + 202.609 * x - 66.96729 \quad \text{(D.74)}$$

The value of x represents the relationship between the sizes of the two frames from the original and transcoded videos, calculated using (6.11).

D.7.2. MPEG-4 Video Codec

I modeled the error in assessing the impact of reducing the frame size from fs_1 to fs_2 on objective video quality for the MPEG-4 codec at a given bit rate, br, as follows:

$$Error^{\delta}_{MPEG4}(\,x.c.br, x.c.fs_1, y.c.fs_2) = \frac{log_2(fs_2.w * a_1) * log_2(fs_2.h * a_2) * (br * a_3)}{log_2(fs_1.w * a_4) * log_2(fs_1.h * a_5) * br^{a_6}} \quad \text{(D.75)}$$

where a_1 to a_6 are the sub-model coefficients, calculated as follows:

$$a_1 = -15.892 * x^2 + 10.626 * x + 0.0534 \quad \text{(D.76)}$$

$$a_2 = -16.834 * x^2 + 11.344 * x - 0.094 \quad \text{(D.77)}$$

$$a_3 = -13.39 * x^2 + 9.3576 * x - 1.519 \quad \text{(D.78)}$$

$$a_4 = 12.26054 * x^2 + -8.12581 * x + 3.49709 \quad \text{(D.79)}$$

$$a_5 = 13.30693 * x^2 + -8.82508 * x + 3.62465 \quad \text{(D.80)}$$

$$a_6 = -30.416 * x^2 + 17.334 * x - 0.2093 \quad \text{(D.81)}$$

D.7.3. FLV Video Codec

I modeled the error in assessing the impact of reducing the frame size from fs_1 to fs_2 on objective video quality for the FLV codec at a given bit rate, br, as follows:

$$Error^{\delta}_{FLV}(\,x.c.br, x.c.fs_1, y.c.fs_2) = \frac{log_{10}(fs_2.w + a_1) * log_{10}(fs_1.w + a_2) * (br^{a_3})}{log_{10}(fs_2.h * a_4) * log_2(fs_1.h * a_5) * log_2(br^{a_6})} \quad \text{(D.82)}$$

where a_1 to a_6 are the sub-model coefficients, calculated as follows:

$$a_1 = 28.25304 * x - 6.93638 \quad \text{(D.83)}$$

$$a_2 = 28.38778 * x - 6.98101 \quad \text{(D.84)}$$

$$a_3 = 323.17874 * x^2 - 224.41641 * x + 37.83325 \quad \text{(D.85)}$$

$$a_4 = 308.82029 * x^2 - 179.25734 * x + 27.63372 \quad \text{(D.86)}$$

$$a_5 = 267.04910 * x^2 - 150.15878 * x + 22.59876 \quad \text{(D.87)}$$

$$a_6 = 323.17874 * x^2 - 224.41641 * x + 37.83325 \quad \text{(D.88)}$$

Figure D.15 shows the error values that represent the error in assessing the impact of reducing the frame size on objective video quality for the H.264, MPEG-4, and FLV codecs at a given bit rate. The

expected error values are generated from the proposed error sub-models (labeled by "M"). The reference error values (labeled by "R") represent the average of the difference between the actual quality values generated from MS-SSIM and the predicted quality values generated from the proposed frame size quality sub-models described above at a given bit rate. The frame sizes are reduced from the original size, i.e., 1920x1080, to 640x480, 1280x720, and 1440x900 for all these codecs. Figure D.15 shows high correlation between these reference and expected error values.

Table D.9 shows the evaluation results in terms of PCC and RMSE for assessing the error in modeling the impact of reducing the frame size on objective video quality for the H.264, MPEG-4, and FLV codecs. These evaluation results are generated by comparing the reference and expected error values, which show high PCC and very low RMSE values.

Table D.9. The evaluation results in terms of PCC and RMSE for assessing the error in modeling the impact of reducing the frame size on objective video quality for the H.264, MPEG-4, and FLV codecs.

fs_{from}	fs_{to}	H.264 sub-model		MPEG-4 sub-model		FLV sub-model	
		PCC	RMSE	PCC	RMSE	PCC	RMSE
1920x1080	640x480	0.73656	0.00234	0.96049	0.00106	0.81297	0.00069
1920x1080	1280x720	0.94689	0.00026	0.99220	0.00098	0.99018	0.00196
1920x1080	1440x900	0.98317	0.00022	0.99000	0.00144	0.98399	0.00233

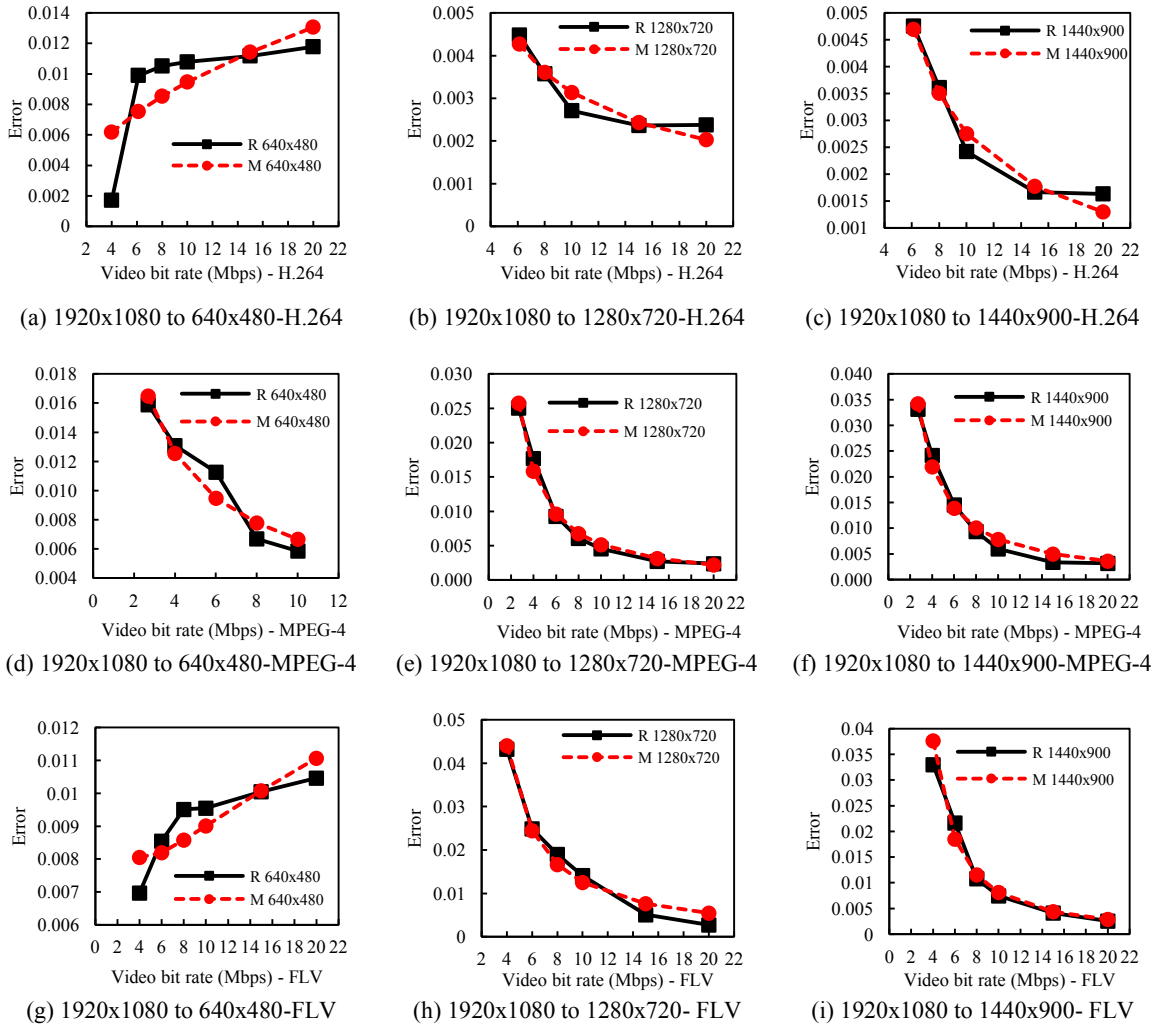
D.8. General Models

This section describes the rest of the general models that include: a) H.264 to MPEG-4, b) FLV to H.264, c) FLV to MPEG-4, d) H.264 to FLV, and e) MPEG-4 to FLV general models.

D.8.1. H.264 to MPEG-4 General Model

I modeled the quality of the transcoded video y generated by changing the codec from H.264 to MPEG-4, reducing the bit rate from br_{in} to br_{out}, reducing the frame rate from fr_{in} to fr_{out}, and reducing the frame size from fs_{in} to fs_{out} for any original video x as follows:

$$y.quality_{VTOM,x} = (y.quality_{\alpha_{H264,MPEG4}(x.c.br),x})^{w_c} * (y.quality_{\beta_{MPEG4}(x.c.br_{in},y.c.br_{out}),x})^{w_{br}} * (y.quality_{\gamma_{MPEG4}(x.c.br,x.c.fr_{in},y.c.fr_{out}),x})^{w_{fr}} * (y.quality_{\delta_{MPEG4}(x.c.br,x.c.fs_{in},y.c.fs_{out}),x})^{w_{fs}} \quad \text{(D.89)}$$



(a) 1920x1080 to 640x480-H.264 (b) 1920x1080 to 1280x720-H.264 (c) 1920x1080 to 1440x900-H.264

(d) 1920x1080 to 640x480-MPEG-4 (e) 1920x1080 to 1280x720-MPEG-4 (f) 1920x1080 to 1440x900-MPEG-4

(g) 1920x1080 to 640x480-FLV (h) 1920x1080 to 1280x720- FLV (i) 1920x1080 to 1440x900- FLV

Figure D.15. The error values that represent the error in assessing the impact of reducing the frame size on objective video quality for the H.264, MPEG-4, and FLV video codecs at a given bit rate. The black curves (labeled by "R") represent the reference error values and the red curves (labeled by "M") represent the predicted error values generated from the proposed error sub-models. The reference error values represent the average of the differences between the actual video quality values generated from MS-SSIM and the predicted video quality values generated from the proposed frame size quality sub-models. The frame size reductions are from 1920x1080 to 640x480, from 1920x1080 to 1280x720, and from 1920x1080 to 1440x900 for the (a) to (c) H.264, (d) to (f) MPEG-4, and (g) to (i) FLV video codecs.

D.8.2. FLV to H.264 General Model

I modeled the quality of the transcoded video y generated by changing the codec from FLV to H.264, reducing the bit rate from br_{in} to br_{out}, reducing the frame rate from fr_{in} to fr_{out}, and reducing the frame size from fs_{in} to fs_{out} for any original video x as follows:

$$y.quality_{VTOM,x} = (y.quality_{\alpha_{FLV,H264}(x.c.br),x})^{w_c} * (y.quality_{\beta_{H264}(x.c.br_{in},y.c.br_{out}),x})^{w_{br}} * (y.quality_{\gamma_{H264}(x.c.br,x.c.fr_{in},y.c.fr_{out}),x})^{w_{fr}} * (y.quality_{\delta_{H264}(x.c.br,x.c.fs_{in},y.c.fs_{out}),x})^{w_{fs}} \quad \text{(D.90)}$$

D.8.3. FLV to MPEG-4 General Model

I modeled the quality of the transcoded video y generated by changing the codec from FLV to MPEG-4, reducing the bit rate from br_{in} to br_{ouy}, reducing the frame rate from fr_{in} to fr_{out}, and reducing the frame size from fs_{in} to fs_{out} for any original video x as follows:

$$y.quality_{VTOM,x} = (y.quality_{\alpha_{FLV,MPEG4}(x.c.br),x})^{w_c} * (y.quality_{\beta_{MPEG4}(x.c.br_{in},y.c.br_{out}),x})^{w_{br}} * (y.quality_{\gamma_{MPEG4}(x.c.br,x.c.fr_{in},y.c.fr_{out}),x})^{w_{fr}} * (y.quality_{\delta_{MPEG4}(x.c.br,x.c.fs_{in},y.c.fs_{out}),x})^{w_{fs}} \quad \text{(D.91)}$$

D.8.4. H.264 to FLV General Model

I modeled the quality of the transcoded video y generated by changing the codec from H.264 to FLV, reducing the bit rate from br_{in} to br_{out}, reducing the frame rate from fr_{in} to fr_{out}, and reducing the frame size from fs_{in} to fs_{out} for any original video x as follows:

$$y.quality_{VTOM,x} = (y.quality_{\alpha_{H264,FLV}(x.c.br),x})^{w_c} * (y.quality_{\beta_{FLV}(x.c.br_{in},y.c.br_{out}),x})^{w_{br}} * (y.quality_{\gamma_{FLV}(x.c.br,x.c.fr_{in},y.c.fr_{out}),x})^{w_{fr}} * (y.quality_{\delta_{FLV}(x.c.br,x.c.fs_{in},y.c.fs_{out}),x})^{w_{fs}} \quad \text{(D.92)}$$

D.8.5. MPEG-4 to FLV General Model

I modeled the quality of the transcoded video y generated by changing the codec from MPEG-4 to FLV, reducing the bit rate from br_{in} to br_{out}, reducing the frame rate from fr_{in} to fr_{out}, and reducing the frame size from fs_{in} to fs_{out} for any original video x as follows:

$$y.quality_{VTOM,x} = (y.quality_{\alpha_{MPEG4,FLV}(x.c.br),x})^{w_c} * (y.quality_{\beta_{FLV}(x.c.br_{in},y.c.br_{out}),x})^{w_{br}} * (y.quality_{\gamma_{FLV}(x.c.br,x.c.fr_{in},y.c.fr_{out}),x})^{w_{fr}} * (y.quality_{\delta_{FLV}(x.c.br,x.c.fs_{in},y.c.fs_{out}),x})^{w_{fs}} \quad \text{(D.93)}$$

Tables D.10, D.11, D.12, D.13, and D.14 show the evaluation results for the H.264 to MPEG-4, FLV to H.264, FLV to MPEG-4, H.264 to FLV, and MPEG-4 to FLV general parametric models, respectively, in terms of PCC and RMSE. I compared the actual quality values generated from MS-SSIM with the predicted quality values generated from each of the above general models. These comparisons yield results that show high PCC and low RMSE values. Figure D.16 shows an aggregation of all these results. It shows that the MPEG-4 to FLV general model achieves the highest PCC values, while the H.264 to MPEG-4, MPEG-4 to FLV, and MPEG-4 to H.264 general models achieve the lowest RMSE values.

Table D.10. The evaluation results for the H.264 to MPEG-4 general model.

Seq. #	video codec		Frame rate		Frame size		Reduction in bit rate	PCC	RMSE
	From	To	From	To	From	To			
vqeghd1_src02	H.264	MPEG-4	29.97	25	1920x1080	1440x900	30%	0.85	0.18
vqeghd1_src04	H.264	MPEG-4	29.97	25	1920x1080	1440x900	30%	1.00	0.08
vqeghd1_src06	H.264	MPEG-4	29.97	25	1920x1080	1440x900	30%	1.00	0.12
vqeghd1_src07	H.264	MPEG-4	29.97	25	1920x1080	1440x900	30%	0.67	0.10
vqeghd1_src02	H.264	MPEG-4	29.97	20	1920x1080	1280x720	40%	0.94	0.18
vqeghd1_src04	H.264	MPEG-4	29.97	20	1920x1080	1280x720	40%	1.00	0.15
vqeghd1_src06	H.264	MPEG-4	29.97	20	1920x1080	1280x720	40%	1.00	0.19
vqeghd1_src07	H.264	MPEG-4	29.97	20	1920x1080	1280x720	40%	0.89	0.08
vqeghd1_src02	H.264	MPEG-4	29.97	25	1920x1080	640x480	40%	0.82	0.12
vqeghd1_src04	H.264	MPEG-4	29.97	25	1920x1080	640x480	40%	0.99	0.14
vqeghd1_src06	H.264	MPEG-4	29.97	25	1920x1080	640x480	40%	1.00	0.18
vqeghd1_src07	H.264	MPEG-4	29.97	25	1920x1080	640x480	40%	0.83	0.04
Average								**0.92**	**0.13**

Table D.11. The evaluation results for the FLV to H.264 general model.

Seq. #	video codec		Frame rate		Frame size		Reduction in bit rate	PCC	RMSE
	From	To	From	To	From	To			
vqeghd1_src02	FLV	H.264	29.97	20	1920x1080	1280x720	30%	0.91	0.37
vqeghd1_src04	FLV	H.264	29.97	20	1920x1080	1280x720	30%	0.97	0.04
vqeghd1_src06	FLV	H.264	29.97	20	1920x1080	1280x720	30%	0.86	0.04
vqeghd1_src07	FLV	H.264	29.97	20	1920x1080	1280x720	30%	0.93	0.29
vqeghd1_src02	FLV	H.264	29.97	25	1920x1080	1440x900	30%	0.91	0.20
vqeghd1_src04	FLV	H.264	29.97	25	1920x1080	1440x900	30%	0.99	0.06
vqeghd1_src06	FLV	H.264	29.97	25	1920x1080	1440x900	30%	0.99	0.12
vqeghd1_src07	FLV	H.264	29.97	25	1920x1080	1440x900	30%	0.98	0.12
vqeghd1_src02	FLV	H.264	29.97	25	1920x1080	1280x720	40%	0.85	0.18
vqeghd1_src04	FLV	H.264	29.97	25	1920x1080	1280x720	40%	0.97	0.04
vqeghd1_src06	FLV	H.264	29.97	25	1920x1080	1280x720	40%	0.91	0.10
vqeghd1_src07	FLV	H.264	29.97	25	1920x1080	1280x720	40%	0.89	0.12
Average								**0.93**	**0.14**

Table D.12. The evaluation results for the FLV to MPEG-4 general model.

Seq. #	video codec		Frame rate		Frame size		Reduction in bit rate	PCC	RMSE
	From	To	From	To	From	To			
vqeghd1_src02	FLV	MPEG-4	29.97	20	1920x1080	1280x720	30%	0.71	0.21
vqeghd1_src04	FLV	MPEG-4	29.97	20	1920x1080	1280x720	30%	0.97	0.11
vqeghd1_src06	FLV	MPEG-4	29.97	20	1920x1080	1280x720	30%	1.00	0.18
vqeghd1_src07	FLV	MPEG-4	29.97	20	1920x1080	1280x720	30%	0.95	0.13
vqeghd1_src02	FLV	MPEG-4	29.97	25	1920x1080	1440x900	30%	0.90	0.20
vqeghd1_src04	FLV	MPEG-4	29.97	25	1920x1080	1440x900	30%	0.99	0.08
vqeghd1_src06	FLV	MPEG-4	29.97	25	1920x1080	1440x900	30%	0.99	0.12
vqeghd1_src07	FLV	MPEG-4	29.97	25	1920x1080	1440x900	30%	1.00	0.12
vqeghd1_src02	FLV	MPEG-4	29.97	20	1920x1080	1280x720	40%	0.87	0.19
vqeghd1_src04	FLV	MPEG-4	29.97	20	1920x1080	1280x720	40%	0.93	0.15
vqeghd1_src06	FLV	MPEG-4	29.97	20	1920x1080	1280x720	40%	1.00	0.20
vqeghd1_src07	FLV	MPEG-4	29.97	20	1920x1080	1280x720	40%	0.94	0.12
Average								**0.94**	**0.15**

Table D.13. The evaluation results for the H.264 to FLV general model.

Seq. #	video codec		Frame rate		Frame size		Reduction in bit rate	PCC	RMSE
	From	To	From	To	From	To			
vqeghd1_src02	H.264	FLV	29.97	20	1920x1080	1280x720	30%	0.96	0.30
vqeghd1_src04	H.264	FLV	29.97	20	1920x1080	1280x720	30%	0.98	0.08
vqeghd1_src06	H.264	FLV	29.97	20	1920x1080	1280x720	30%	0.96	0.13
vqeghd1_src07	H.264	FLV	29.97	20	1920x1080	1280x720	30%	0.97	0.20
vqeghd1_src02	H.264	FLV	29.97	25	1920x1080	1440x900	30%	0.89	0.17
vqeghd1_src04	H.264	FLV	29.97	25	1920x1080	1440x900	30%	0.99	0.11
vqeghd1_src06	H.264	FLV	29.97	25	1920x1080	1440x900	30%	0.92	0.16
vqeghd1_src07	H.264	FLV	29.97	25	1920x1080	1440x900	30%	0.89	0.09
vqeghd1_src02	H.264	FLV	29.97	25	1920x1080	1280x720	40%	0.95	0.14
vqeghd1_src04	H.264	FLV	29.97	25	1920x1080	1280x720	40%	0.97	0.14
vqeghd1_src06	H.264	FLV	29.97	25	1920x1080	1280x720	40%	0.97	0.17
vqeghd1_src07	H.264	FLV	29.97	25	1920x1080	1280x720	40%	0.71	0.07
Average								**0.93**	**0.15**

Table D.14. The evaluation results for the MPEG-4 to FLV general model.

Seq. #	video codec		Frame rate		Frame size		Reduction in bit rate	PCC	RMSE
	From	To	From	To	From	To			
vqeghd1_src02	MPEG-4	FLV	29.97	25	1920x1080	1280x720	30%	0.96	0.17
vqeghd1_src04	MPEG-4	FLV	29.97	25	1920x1080	1280x720	30%	0.98	0.10
vqeghd1_src06	MPEG-4	FLV	29.97	25	1920x1080	1280x720	30%	0.99	0.13
vqeghd1_src07	MPEG-4	FLV	29.97	25	1920x1080	1280x720	30%	0.96	0.11
vqeghd1_src02	MPEG-4	FLV	29.97	20	1920x1080	1280x720	40%	0.98	0.26
vqeghd1_src04	MPEG-4	FLV	29.97	20	1920x1080	1280x720	40%	0.98	0.06
vqeghd1_src06	MPEG-4	FLV	29.97	20	1920x1080	1280x720	40%	0.99	0.09
vqeghd1_src07	MPEG-4	FLV	29.97	20	1920x1080	1280x720	40%	1.00	0.18
vqeghd1_src02	MPEG-4	FLV	29.97	25	1920x1080	1440x900	40%	0.97	0.17
vqeghd1_src04	MPEG-4	FLV	29.97	25	1920x1080	1440x900	40%	0.98	0.10
vqeghd1_src06	MPEG-4	FLV	29.97	25	1920x1080	1440x900	40%	0.98	0.14
vqeghd1_src07	MPEG-4	FLV	29.97	25	1920x1080	1440x900	40%	0.94	0.09
Average								**0.98**	**0.13**

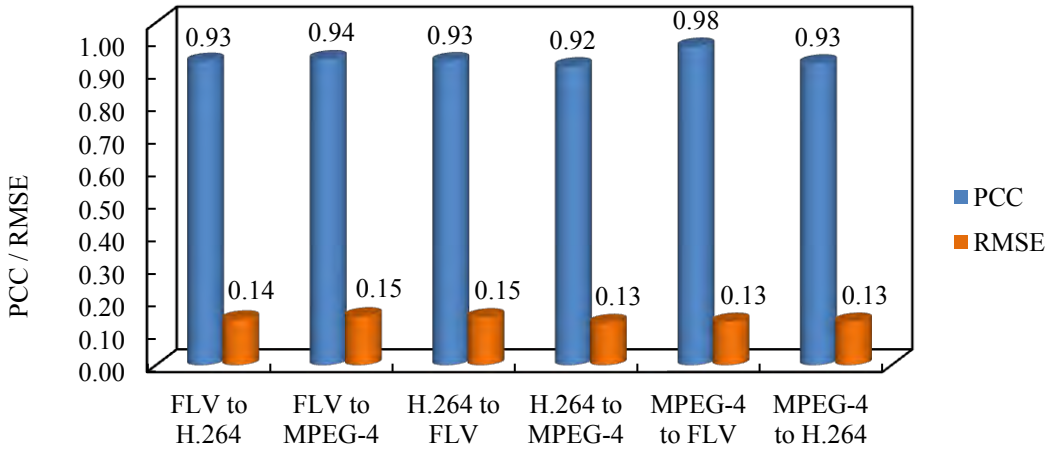


Figure D.16. The evaluation results of the VTOM general models in terms of PCC and RMSE.

APPENDIX E

CURRICULUM VITAE

Nawaf O. Alsrehin

(May 2016)

CONTACT INFORMATION

- Email: nawwaf981@yahoo.com, n_alsrehin@yu.edu.jo
- Address: 39 Aggie Village Apt. E, Logan UT, 84341, USA.
- Phome: USA +1(435)213-7698, Jordan +962-2-7381555

EDUCATION

- Ph.D., Computer Science, Utah State University, Logan, UT, 2016, GPA 3.92/4.0.
- M.Sc., Computer Information Systems, Yarmouk University, Irbid, Jordan, 2006, GPA 88.1%, ranking as 3rd out of 17.
- B.S., Computer Science, Yarmouk University, Irbid, Jordan, 2003, GPA 71.1%.

RESEARCH INTERESTS

Multimedia, Multimedia Communications, Computer Vision, Image Processing, Video Transcoding and Streaming, Multimedia Services Selection and Composition, MapReduce for Multimedia Applications, Video Quality Assessments, Cloud-based Video Transcoding, Natural Language Processing, and Information Retrieval.

HONORS AND AWARDS

- Best Oral Presentation Award, in the Second International Conference on Multimedia Communication Technology (ICMCT 2015), Hong Kong, September 19-21, 2015.
- A Scholarship from Yarmouk University to pursue my Ph.D. in computer science, September 2011 to August 2015.

- Graduate Student Travel Grants, RGS Graduate Student Travel Award, and Student Travel Award from: School of Engineering - Dean Office, USU Graduate School, USU Computer Science Department, Utah State University, July 2015, Travel funding to attend the 2nd International Conference on Multimedia and Communication Technologies (ICMCT 2015). Hong Kong, September 19-20, 2015.

PUBLICATIONS

- Nawaf O. Alsrehin and Stephen W. Clyde, "QoS-Aware Video Transcoding Service Selection Process," Journal of Media & Mass Communication, Vol. 1, No. 2, pp. 61-68, December, 2015. doi: 10.12720/jmmc.1.2.61-68.
- Nawaf O. Alsrehin, QoS-aware Video Transcoding Service Composition Process in a Distributed Cloud Environment, International Journal of Innovative Research in Computer and Communication Engineering, Volume 3, pp 3726-3748, 30 May 2015; doi:10.15680/ijircce.2015.0305001.
- Sameh Ghwanmeh, Ghassan Kanaan, Riyad Al-Shalabi, and Nawaf Alsrehin, "Automatic Query Expansion for Arabic Text Retrieval System Based on an Association Clustering Thesaurus", Information Technology Journal Vol. 4, Issue 4, pp 476-483, 2005, ISSN 1812-5638.

POSTERS

- Nawaf O. Alsrehin and Stephen W. Clyde, "VTOM: Toward A General Parametric Model for Assessing the Impact of Video Transcoding on Objective Video Quality", GrTS Research Week, Utah State University, April 2016, USA.
- Nawaf O. Alsrehin and Stephen W. Clyde, "QoS-Aware Video Transcoding Service Selection Process", Computer Science Department, Utah State University, Spring 2014, Fall 2015.
- Nawaf O. Alsrehin and Stephen W. Clyde, "QoS-Aware Video Transcoding Process", Computer Science Department, Utah State University, Industry Day, Spring 2014.

TRAINING AND CERTIFICATIONS

- November 13, 2012. Membership certificate of the Golden Key International Honour Society by Utah State University, Logan, UT, USA.
- August 13 - 22, 2012, International Teaching Assistant workshop (ITA) given by the Intensive English Language Institute at Utah State University, Logan, UT, USA.
- December 26, 2009, Using Interactive Smart Boards and Tools in Collaborative Learning given by Ketab Technology company in Yarmouk University, Irbid, Jordan.
- June 28, 2009 – July 29, 2009, Electronic Learning using Moodle given by Queen Rania Center for Jordanian Studies and Community Service in Yarmouk University, Irbid, Jordan.
- June 10 – 12, 2008. Design and Evaluation of Innovation Policy in Developing Countries given by Jordan Innovation Center, Royal Scientific Society, and United Nations University in Princes Somayah University, Amman, Jordan.

TEACHING EXPERIENCE

- January 2012 to May 2016, Graduate teaching/research assistant, computer science department, Utah State University, Logan UT, USA.
- August 2013 to December 2013, Graduate instructor, CS2410 Graphical User Interface Development in Java, computer science department, Utah State University, Logan UT, USA.
- May 2015 to July 2015. Lab Instructor, CS1410: C++ Programming Language, computer science department, Utah State University, Logan UT, USA.
- February 2008 to August 2011, Full time lecturer, Yarmouk University, Faculty of Information Technology and Computer Science, computer information system department, Jordan-Irbid.
- September 2006 to August 2007, Full time lecturer, King Saud University, Faculty of Information Technology and Computer Science, computer science department, Al-Riyadh, Saudi Arabia.
- February 2006 to August 2006 and September 2007 to January 2008, Part-time lecturer in computer science department, Jordan University of Science and Technology (Irbid – Jordan),

Yarmouk University (Irbid – Jordan), Arab Open University (Amman – Jordan), and Al-Albayt University (Al-Mafraq – Jordan).

PERSONAL/SCIENTIFIC SKILLS

- Excellent Hardware Maintenance.
- Excellent Various Software Installations. Operating System (OS): (MS-DOS, Windows 98, NT, 2000, XP, Vista, 7, Server, Advanced Server, Mandrake Linux 8.2.).
- Programming Languages: (Assembly, FORTRAN, Prolog, Pascal, C++, Visual Studio.NET, C#, Java and AspectJ, MATLAB)
- Web Programming development (XHTML, HTML, XPath, XQuery, XML)
- Database Management Systems: (Advanced Access 2000, 2002, Oracle10g, Workbench, DataLog, Pig-Latin, Hadoop)
- Web Page development (ASP.NET, Ioinic, AngularJS)
- Object Oriented Methodologies (Modeling Languages): UML, Booch, and Coad Methodology.
- Modeling Tools: Rational Rose, Smart Draw, Meta Mill, Request Brow, Visual Paradigm UML 8.3.
- Geographic Information System (GIS) Software Tools: ArcView 3.2, ArcGIS.
- Multimedia Software: Adobe Flash CS5.5, Adobe Photoshop CS5, Adobe Illustrator CS3. SQL tools MySQL 5.2 CE, OQL, SQL server.
- Image Processing applications using MATLAB.
- Parallel Computing: MPI-C++ under UNIX. Scripting Languages: PostScript, JavaScript.

COURSES THOUGHT

- Computer Skills I and II
- Programming using C++
- Programming using Visual Basic.NET
- Introduction to Computer Science

- Introduction to Information System
- Introduction to Computer Information System
- HTML Programming
- Software Engineering Lab (Rational Rose, Request Pro., and N-Unit)
- Data Structure
- Discrete Mathematics
- Multimedia Systems Lab (Adobe Flash CS3, Photoshop CS3, Illustrator CS3)
- Information Retrieval