Correlating Video Quality Metrics to User Experience: an Event-based Approach 2012 Yongfeng Huang

Master's Thesis

Correlating Video Quality Metrics to User Experience: an Event-based Approach

Yongfeng Huang

Division of IT Convergence Engineering (Autonomics)

Pohang University of Science and Technology

2012

Correlating Video Quality Metrics to User Experience: an Event-based Approach

Correlating Video Quality Metrics to User Experience: an Event-based Approach

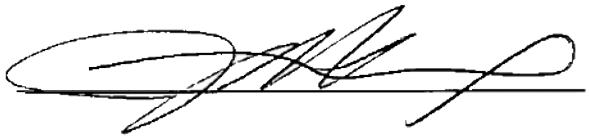
by

Yongfeng Huang
Division of IT Convergence Engineering (Autonomics)
Pohang University of Science and Technology

A dissertation submitted to the faculty of the Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Master of Science in the Division of IT Convergence Engineering (Autonomics).

Pohang, Korea
June 22, 2012

Approved by



Academic Advisor: James Won-Ki Hong

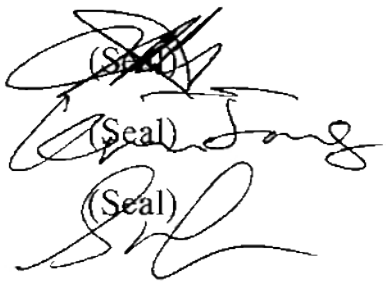
Correlating Video Quality Metrics to User Experience: an Event-based Approach

Yongfeng Huang

The undersigned have examined this thesis/dissertation and hereby certify that it is worthy of acceptance for a master's degree from POSTECH.

June 22, 2012

Committee Chair	James Won-Ki Hong	(Seal)
Member	Jong Kim	(Seal)
Member	Young-Joo Suh	(Seal)



MITCE 20101030 Yongfeng Huang, “Correlating Video Quality Metrics to User Experience: an Event-based Approach”
Division of IT Convergence Engineering (Autonomics), 2012, 49P, Advisor: James Won-Ki Hong, Text in English.

ABSTRACT

Multimedia services have become an important part of today’s network offering. How to guarantee user’s perception of delivered video quality is a critical problem for all content providers, especially when the underlying transportation uses a shared network infrastructure.

In this thesis, we present an event-based model to correlate video quality metrics to user experience, as part of VIDAR - a comprehensive VIDeo quality Analyzer in Real-time. Our model maps objective frame-level metrics to perceivable video defects with correlated user experience, using machine learning techniques.

Through experiments, our model shows good classification accuracy of defect events while keeping the training time reasonable. Furthermore, we have developed user perception models for each event, which also show good prediction of the user experience when subjected to the specific type of events.

Contents

List of Figures

List of Tables

1 Introduction

Multimedia services have become an important part of today's network offering. Therefore, how to guarantee user's perception of delivered video quality is a critical problem for all content providers, especially when the underlying transportation uses a shared network infrastructure. In order to improve user's Quality of Experience (QoE), it is important to have a good understanding of the different parameters that affect the perceptual quality of displayed content. These parameters include underlying network-related factors (packet loss, delay, jitter, etc.), application-related factors (frame rate, bit rate, codec types, loss recovery technique, etc.), and human perception-related factors (light, user's preference for video content, etc.). In this thesis, we are interested in the effect of network-related factors on user experience, specifically with respect to network packet loss, because delay or jitter can be translated into losses in the playback buffer.

In order to assess user's perception of video quality, many existing work rely on measurable objective metrics. In particular, the peak signal-to-noise ratio (PSNR)/the mean squared error (MSE), structure similarity (SSIM) [1], and video quality metric (VQM) [2] are popular. However, all of them are full-reference metrics, which means they need the original frame for reference. Although these approaches give quantifiable measurements of video frame quality (measured as deviation from the original source frame), they generally do not consider temporal factors and the related influence of human perceptions.

In contrast to these objective metrics, user experience is generally measured in terms of QoE, which is obtained via direct user feedback. A common method of evaluation QoE is the mean opinion score (MOS), ranging from 1 (bad) to 5 (excellent) [3]. However, as a metric, MOS is not readily measurable and its reliance on human feedbacks makes it difficult for content providers to estimate user's QoE.

To date, mapping from frame quality metrics to user experience is remains an important open problem. This problem is challenging because such a mapping should not only correlate the objective metrics to users' perception, but also augment biological and

psychological factors of human perception.

In this thesis, we present the video perception model of VIDeo quality Analyzer in Real-time (VIDAR) project [4], which uses a machine learning approach to evaluate user experience of video services under varying network conditions. Machine learning (ML) has been widely used in literature to establish correlation among metrics that are either complex to mathematical model or with many sensitive parameters that are difficult to tune in isolation. It is important to mention that for a ML approach to work well, two crucial conditions must be met: 1) there must exists a strong correlation between the input parameters (called features in ML lingo) and the output result; 2) the number of features must be small and sensitive. In surveying the existing literature on video quality assessment using ML techniques (see Related Work in Chapter 2), we find that existing work attempt to build very complex models that attempts to incorporate many factors (network, codec, content, and human) under a single ML mechanism, resulting in solutions that are both difficult to train and to apply in practice. In contrast, we take a more focused and decomposable approach to the mapping procedure. First, we observe that human experience is formulated by recalling and evaluating memorable events. In the case of video perception, these events are usually some form of visible defects. Second, we observe that different type of defects generates different degree of experience from the user. From these two observations, we propose an event-based mapping methodology, whereby we attempts to first classify the type of defects the users will experience based on frame level metrics, and then to correlate the intensity of dissatisfaction associated with each specific events. With this methodology, we are able decompose the entire mapping problem into sub-problems of event classification, and intensity mapping per event class. In doing so, we meet the two crucial conditions under which ML techniques can be successfully applied.

Our perception model targets on the following two main problems:

- Because frame level objective metrics do not consider human perception factors, how to estimated user MOS from them?

- Because our model aims to be applied in practical operations, how to ensure that it is simple and fast?

Accordingly, our contributions are as follows:

First, our eSSIM aggregator group frame quality metrics into defective events, which are series of defective frames in close play-time proximity. Then, multi-class classification is applied to automatically determine the types of defects a human viewer will experience. Because of the diverse defect patterns produced by stochastic packet losses and the varied length of defective events, traditional time series classification by measuring similarity among series are not suitable to our problem. Therefore, we extract statistical features from the time series data that best represent the characteristics of the defective events.

Second, to ensure that our model is simple and fast, we focus on key features that are sensitive for classification. Furthermore, optimization processes are applied to the machine learning techniques, including parameters selection for radial basis function (RBF) kernel of support vector machine (SVM), and methods selection for aggregating binary classifiers.

Third, for each type of defect events, we design a user model (ML-based or statistical correlation) to estimate user experience (MOS) about an event.

Through experiments of both eSSIM aggregator and the user model in Chapter 4, we show that our perception model can provide good estimate of user MOS from frame level objective metrics, while keeping the overall process simple and practical to operate on.

The rest of the thesis is organized as follows. Chapter 2 provides a brief overview of H.264/AVC, related work on objective metrics, as well as that on mapping from objective metrics to user MOS. Chapter 3 presents the overview of VIDAR, specifically our user model based on different types of defect events. In Chapter 4, we show the result of validation. We give our conclusion in Chapter 5.

2 Related Work

In this chapter, several objective video quality assessment methods are briefly introduced, and related work on inferring user MOS from these objective metrics are reviewed. Then, we give a comparison among multiple supervised ML techniques, and explain why k-NN and SVM are selected as our candidate algorithms. Finally, H.264/AVC, which is chosen as the video technology, is summarized.

2.1 Objective Video Quality Assessment Methods

Objective video quality assessment (VQA) methods can be categorized as full-reference (FR), reduced-reference (RR), and no-reference (NR) depending on whether methods require access to full information, partial information, or no information of transmitted videos, respectively [6]. Because FR methods compare the source video and the transmitted video frame-by-frame, exact temporal and spatial alignment of the two videos are necessary. By having access to full information, FR methods usually give better result in terms of accuracy than RR and NR. However, they are impractical for real-time scenarios, because of their larger bandwidth requirement for transferring full video information. In contrast, NR gives the worst accuracy because they predict the expected distortion without any information of transmitted videos, although they are more flexible because of the same reason. RR, as a trade-off method, can reduce both additional bandwidth requirement and assumption on the video characteristics, by extracting features from the transmitted video.

Among the different objective VQA methods, PSNR/MSE is simple and popular. However, it does not consider human perception and therefore have been shown to have poor result in practice [7]. VQM on the other hand aggregates seven independent features based on luminance or chrominance components into a single metric, and haven shown good correlation with user's perception of video quality [2]. However, it is processing intensive, and consequently is not suitable for real-time analysis. Another popular metric

is SSIM [1], which is computed frame by frame on the luminance component of the video. It has been shown to be a good metric for still image quality assessment, but it does not consider temporal factors or content features of video.

We choose SSIM as our objective metric, and remedy its shortcoming with our subjective model (Vidi model) in VIDAR [4]. As SSIM is FR, an estimated SSIM (eSSIM) metric is introduced in the R3 model of VIDAR, as an RR method, that produces an approximate of SSIM.

2.2 From Objective Metrics to User Experience (MOS)

Objective metrics can provide concrete information at the video frame level (MSE, PSNR, etc.) and at the network level (packet loss, delay, etc.). Here, we discuss how objective metrics can be utilized to assess user experience that is generally subjective.

Since objective metrics do not consider natural visual characteristics or perceptual characteristics of human visual system (HVS), many work attempts to map objective metrics to human experience by incorporating human perception factors.

A simple perceptual metric based on MSE is proposed as follows [8]:

$$MOS_p = 1 - k(MSE), \tag{1}$$

where k is derived from the spatial edge strength. Spatial edges give a good estimate of the amount of detail in a region and are related to object boundaries, surface crease, and other important visual events [9]. However, MOS_p is only designed to predict MOS of compressed video without considering distortion caused by varying network conditions.

PSNR is normalized in the MOS scale by scaling factor a and shift factor b, which are obtained by applying an affine minimum MSE estimator [10]:

$$\widehat{MOS_{PSNR}}[n] = a \cdot PSNR[n] + b, \tag{2}$$

where n is the frame index. After normalization, several simple human perception rules are applied to correct this result. For example, gradual scene changes are compensated by multiplying the minima with $k > 1$, which is obtained by linear minimum mean square estimation applied to all tested sequences.

The above techniques are representative of the type of statistical techniques used. In summary, statistical models rely on a very small number (usually 1 or 2) of salient perception characteristics that has strong correlation with human vision systems. For more comprehensive treatment of multiple perception factors, machine learning (ML) techniques are sometimes used for QoE prediction [11]. Both support vector machine (SVM) and decision tree, as popular ML algorithms, are used to predict user's perception of video quality. Data instances used in these algorithms are made up of four parameters, including video spatial information, video temporal information, frame rate and bitrate, while the output is just simple "YES" or "NO". Clearly, simply accepted or reject cannot express user's QoE comprehensively. These approaches typically use application-level metrics as indicator metrics.

In [29], authors explored the use of temporal quality fluctuations as an important factor towards effective video quality assessment. Considering the spatial factors, support vector regression is applied to find the factors' interaction to the overall video quality. However, their temporal factor is obtained by simply computing the average spatial quality of a certain number of worst frames. The disadvantage of their method is that both the spatial and temporal factors are based on frames, and they did not consider content factors, i.e. scene change.

Some works also try to predict user perception of video quality from network-level metrics using ML algorithms. Neural network-based model is designed to optimize the QoE, with a random packet loss on the link [12]. However, because they calculated the QoE of the received videos using the PSNR and SSIM, the same problem for PSNR and SSIM still exists, which is that they did not consider temporal or content-related factors. The same disadvantage is for [30]. A neural network was trained to viewer responses on the ITU-R 9-point quality scale, when a single 10-s sequence was subjected to different bandwidth, frame rate, packet loss rate, and I-block refresh rate. In [33], two modeling approaches are studied to predict packet loss visibility in MPEG-2 video: one classifies each packet loss as visible or not; the other one predicts the probability that a packet loss is visible. However, without considering HVS factors, simple classification of visible

packet loss and invisible one cannot reflect user perception in detail.

Most of the effort to understand the visual impact of network-level metrics, i.e. packet losses, has focused on the goal of understanding the average quality of typical videos subjected to average packet loss rate, by ignoring that packet losses may have different visual effects. For instance, in [31], authors studied video conferencing using the average judgment of consumer observers to examine the relative importance of bandwidth, latency and packet loss. The impact of packet loss on the MOS of real-time streaming media is studied for Microsoft Windows Media encoder 9 videos [32].

Compared to the existing methods that predict user's QoE from either application-level metrics or network-level metrics, our approach is event-based, which is motivated by the observation that human experience is largely event-based. The viewers react and evaluate their experience by recollecting their reactions to past events. In this way, we can decompose the complex problem of user QoE assessment into sub-problems of predicting QoE of specific types of events. Consequently, our sub-problem shows stronger correlation between user QoE and a small set of indicator metrics, and can achieve higher accuracy with smaller training data set.

2.3 Machine Learning Techniques

Since ML techniques are extensively used in our methodology, we now give a summary of supervised machine learning techniques. We do not consider unsupervised ML techniques as they are only preferred when training is not possible.

In essence, supervised ML [13] is the process of learning from supplied instances to produce general hypotheses, and then make predictions about future instances. Basic steps in the processing include: collecting dataset, data pre-processing, feature construction, algorithm selection, training, and evaluation with test data. According to the result of evaluation, we need repeat this process in order to tune each stage of the process as to improve accuracy.

Supervised ML algorithms can be classified into logic-based algorithms, perceptron-based techniques, statistical learning algorithms, and SVM [13]. Among logic-based

algorithms, decision tree is a popular one. Each node in a decision tree represents a feature of instances, and each branch represents a value that the node can assume. However, logic-based algorithms, like decision trees, usually cannot perform well with numerical features. Artificial neural network (ANN) [13], as a popular algorithm of perceptron-based techniques, has been applied to many real-world problems, but can be very inefficient with the presence of irrelevant features. Naïve Bayesian network (NB) [13] is the most well known representative of statistical learning algorithms. The major advantage of NB is its short computational time for training, but NB is only applicable in specific problem scenarios, because it assumes that it can discriminate between classes with a single probability distribution. Another popular algorithm under the category of statistical method is k-nearest neighbor (k-NN) [13], which is based on the principle that instances close to each other have similar properties. Although k-NN is very sensitive to irrelevant features, its tuning is straight forth as there is one single parameter (the number of nearest neighbors) to adjust. Finally, as one of the most recent supervised ML technique, SVM has been shown to perform well when dealing with multi-dimension and continuous features, as well as when a nonlinear relationship among the input and output features.

Considering the pros and cons of each ML technique, no single one can uniformly outperform other algorithms over all datasets. The preferable approach is to test several candidate algorithms on the dataset, and then choose one with the highest accuracy. Therefore, for our problem specifically, we choose k-NN (statistical) and SVM (multi-dimensional) as the chosen techniques for classifying defect events.

2.4 Overview of H.264/AVC

H.264/AVC was developed by ITU-T Video Coding Experts Group together with ISO/IEC Moving Picture Experts Group in 2003. It is the current state-of-the-art video codec standard by achieving a significant improvement in rate-distortion efficiency relative to existing standards [5]. In H.264/AVC, a video is composed of a sequence of frames, and a frame consists of several slices, which are made up of consecutive

macroblocks. H.264/AVC is also a temporal compression scheme, which means that besides conventional spatial compression within each frame, it also considers redundant probabilities among frames. To be specific, three slice types are defined. Intra slice (I slice) is coded using only spatial information within the slice. In addition to the coding scheme of I slice, predictive slice (P slice) is also coded using inter prediction with at most one previous reference slice. Bi-predictive slice (B slice) is similar to P slice, but uses inter-prediction with two reference slices. In general, all of the slices in a frame are coded with the same slice type, termed I-, P-, and B-frames.

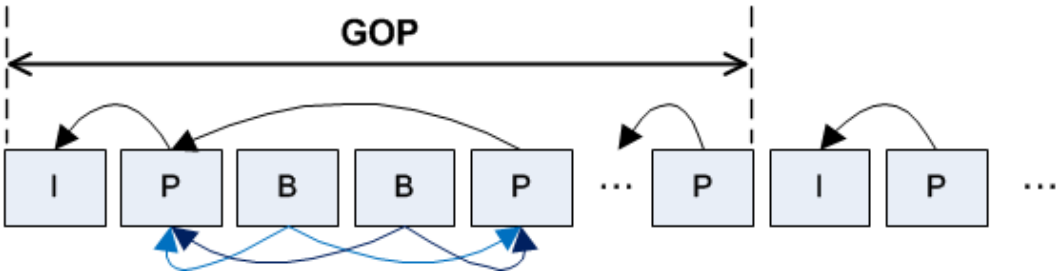


Figure 1. Example of Frame Reference Relationships

Because both intra-frame prediction and inter-frame prediction are used during the coding process, several problems may occur during the decoding process, if network packets are lost. First, H.264/AVC is subject to error propagation, which means that the loss of a single packet containing one or more slices, affects not only frames these slices belong to, but also frames using these slices as a reference. However, this error propagation is limited to a group of pictures (GOP), which starts with an I-frame followed by P- and B-frames. Usually, the encoder can set the maximum and minimum GOP sizes, however the actual GOP size depends on the content of videos. For example, a GOP may start after a scene change where frames change greatly in a short time. Second, H.264/AVC is also sensitive to the content of video, meaning that the more things change over time, the more difficult it is to compress the video while maintaining high quality and minimizing bandwidth and storage. More specifically, a scene change or a fast moving of object has negative impact on bandwidth and video quality. A detail

exploitation of these characteristics can be found in VIDAR's R3 model [4].

3 Proposed Approach

In this chapter, first, we will give an overview of VIDAR project, including its components and their functionalities. Then, we will focus on the construction of event-based eSSIM aggregator, and the way the user model predicts user MOS.

3.1 Overview of VIDAR Project

VIDAR is a video quality analyzer in real-time. It takes as input the network QoS conditions observed at the client side, and estimates the impact on the quality of video frames and user's perception of video quality. Multiple aspects that have an effect on the user experience are considered, including varying network conditions, codec behavior, error recovery techniques, and video content features. In our investigation, we have focused on packet loss. VIDAR is made up of three models: the R3 model, the Vidi model, and the user model (Figure 2).

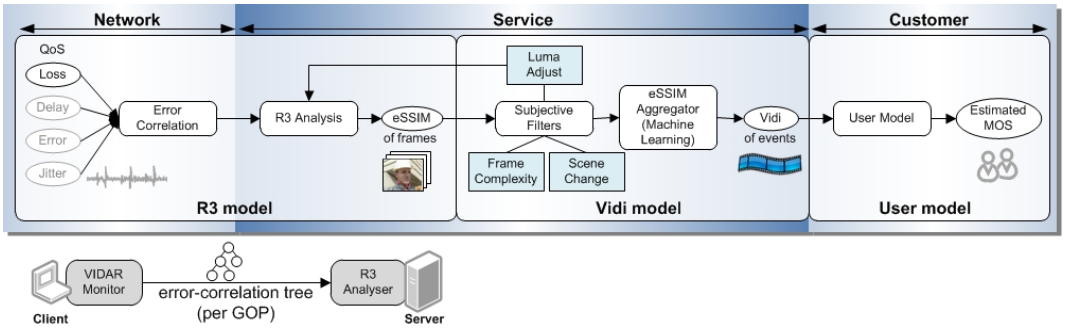


Figure 2. Process flow of VIDAR framework

The R3 model relies on a lightweight client-side monitor and a server-side R3 analyzer. The client-side monitor is a modified video decoder that can generate error-correlation trees and send it to the server-side R3 analyzer. Error-correlation trees are the key part of the R3 model, which is a result of packet loss in our case. Besides, error-correlation trees are constructed based on the frame reference strategies, so they are codec dependent in

fact. By analyzing these trees, it is possible to estimate SSIM without access to transmitted videos [4]. Our R3 model has three salient advantages:

- The amount of information transferred between the client and the server is minimized;
- Client side can be kept lightweight, ideal for power-constrained mobile devices;
- Better preservation of network-video causalities.

The Vidi model relates eSSIM of frames to temporal events of perceived video defects. The design rationale behind this model is as follows: a good estimation of image quality is not sufficient to evaluate perceived video quality on its own. This is because user perception is video content-dependent. For instance, it is known that minor to moderate defects in dark scenes are much less noticeable to the viewers [1] and residual image can hide defects in frames immediately following a scene change [14]. In Vidi model, we consider four key content features: luminance, frame complexity, scene change, and motion. A combination of these factors can be used to describe different classes of video content (e.g. news, sports, action, drama, etc.). The Vidi model takes into account of these features through the use of Vidi filters. The filters modify the eSSIM values and help the eSSIM aggregator to generate defective events, recorded as a series of video index (Vidi) metrics.

The user model maps the Vidi metrics to user MOS. The mapping approach depends on that how different types of defective events affect user QoE and their intensity. Although we do not provide a mapping of Vidi metrics to MOS in this thesis, we show how Vidi metrics can be used to analyze viewer's perception and the impact of different defects on user's MOS.

3.2 eSSIM Aggregator

As shown in Figure 2, eSSIM aggregator takes as input eSSIM series that have been passed through subjective content filters, and its output are classified events with their intensity, termed as Vidi metrics. ML techniques are applied to classify different types of defect events. In the following part, we will give an overview of the process flow of

eSSIM aggregator, and then each process is explained in detail.

3.2.1 Overview of eSSIM Aggregator

The construction of eSSIM aggregator is motivated by the following observations:

- Human perception is content-dependent;
- Human experience is event-based as we react and evaluate our experience by recollecting past events.

Basically, event is a time-related concept, and in this thesis, we consider event as a consecutive series of video frames, which is a short sequence video for users. Under the effect of varying network conditions, an error propagation, a series of dropped frames or duplicated frames may occur in transmitted videos that are coded by H.264/AVC. Finally, these errors result in different types of defect events in the videos that are perceived by users.

According to our experiential investigation, defect events of transmitted videos can be categorized as follows:

- ***Distortion***: frames containing perceivable distortions.
- ***Glitch***: short sequence of distortion, which is slightly perceivable.
- ***Freezing***: series of duplicate frames, or frames dropped and duplicated in interleaving patterns.
- ***Discontinuity***: two frames not in consecutive order are played back-to-back.

Through our initial experiments, we find that these four defective types are not mutually exclusive and some types often co-exist in the same event, i.e., freezing is always followed by discontinuity. Therefore, we combined freezing and discontinuity as a single defect event in the experiment.

In our previous work [4], these defect events are generated from a modified aggregated eSSIM by passing eSSIM through the content feature filters (Figure 2). Then, a specific class label for each event has to be tagged by users manually, which is impractical and prone to human errors. Therefore, this motivates us to develop an approach to classify defect events automatically by using ML techniques.

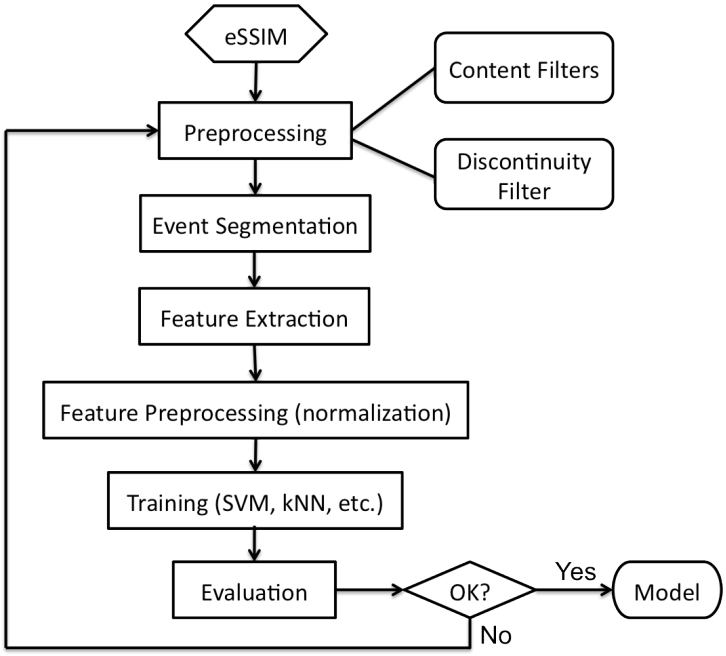


Figure 3. Process flow of eSSIM aggregator

The process flow of the eSSIM aggregator is shown in Figure 3. Over all, these processes can be grouped into two parts: 1) dataset preparation for ML techniques, including eSSIM data preprocessing, event segmentation, feature extraction from events, and feature preprocessing; 2) ML techniques and their optimization, including reducing computation complexity and training time. Since we have multiple types of defect events, multi-class classification techniques are used. Among our candidate binary classification algorithms, kNN can be directly extended to multi-class classification, while a combination of several binary SVM classifiers has to be used to solve a given multi-class problem.

In conclusion, we investigated multi-class classification for the defect events. To improve the accuracy while keeping the classification processing efficient, several optimization steps are applied (e.g., parameter selection for the kernel of SVM).

3.2.2 Preprocessing for eSSIM Raw Data

Considering that human perception is content-dependent, eSSIM raw data has been passed through four content filters, including luminance, frame complexity, scene change and motion (Figure 1). However, to improve the accuracy of classification, some hints injected into raw data can also be helpful. Among the four defective types, all of distortion, glitch and freezing have their own mark in the eSSIM raw data: distortion may exist in a series of frames with lower eSSIM values (the range of original eSSIM value is [0, 1]. 1 shows a perfect match between the original frame and the transmitted frame. The nearer 0, the heavier the distortion, but 0 means that the frame is duplicated or dropped.), glitch may exist in a shorter series of frames with not very low value, and freezing usually happens with a sequence of duplicated frames (eSSIM=0). The difference between discontinuity and the other three types is that only discontinuity requires the comparison between two frames. Based on our experience of experiments, discontinuity always happens with freezing. To be precise, there are two situations:

- ***Frames dropped***: if the frame before dropped frames and the frame after them have distinguishing content, discontinuity may be perceived.
- ***Frames duplicated***: when a sequence of frames duplicate the frame before them, and the frame after them has a distinct content, discontinuity may be perceived.

To let the original eSSIM have discontinuity characteristic, we introduce a discontinuity mark:

$$eSSIM_{disc} = ssim(frame_b, frame_a) - 1, \tag{3}$$

where function *ssim*() calculates the structure similarity between the frame ($frame_b$) before and one ($frame_a$) after duplicated or dropped frames. Since the value range of SSIM is [0, 1], $eSSIM_{disc}$ has a range of [-1, 0]. Therefore, the closer the value is to 0, the less difference there is between $frame_b$ and $frame_a$.

According to two situations mentioned above, we give two modification strategies, respectively:

- ***Frames dropped***: replace the original *eSSIM* of dropped frames with $eSSIM_{disc}$.

- ***Frames duplicated***: keep *eSSIM* of duplicated frames at 0, and add an $eSSIM_{disc}$ after duplicates.

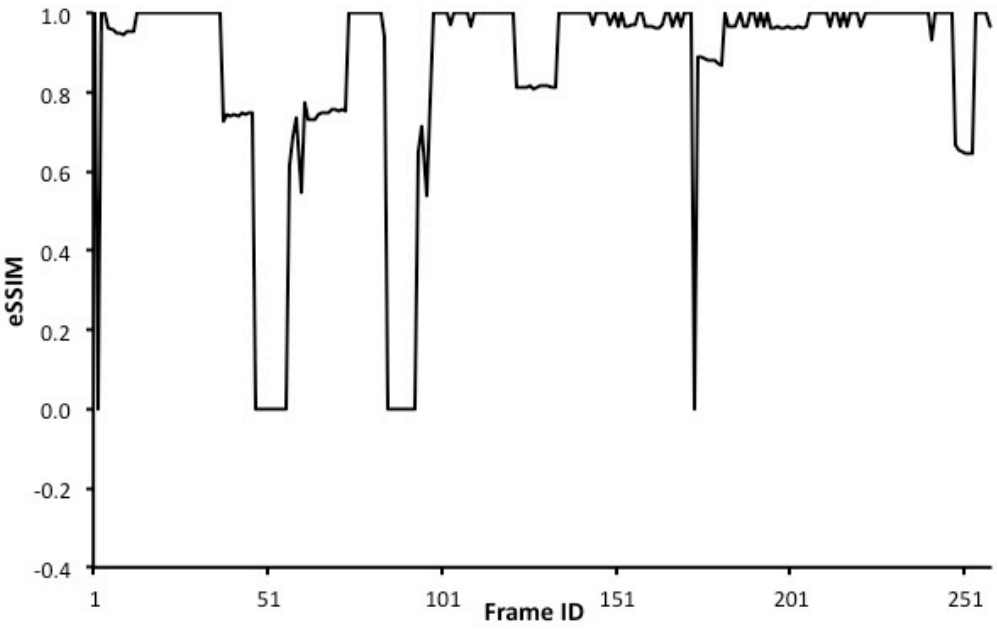


Figure 4. eSSIM example without discontinuity filter

Compared to Figure 5, *eSSIM* of the same transmitted video without discontinuity filter is shown in Figure 4. Two issues are worth mentioning. First, the process of $eSSIM_{disc}$ is done at the server side, meaning that $frame_b$ and $frame_a$ can be extracted from the original video. Second, because for human vision system, the first few frames immediately following a scene change are not perceived [14], hence our process work on eSSIM values that have already passed through the scene change filter.

3.2.3 Event Segmentation

The design of event-based classification is motivated by the observation that human experience is event-based as we react and evaluate our experience by collecting past events. At the same time, this motivation also shows us that the length of event (in this

thesis, the length of event means the number of frames in an event) can be scalable and the boundary between events can be coarse. But event segmentation is a key part to generate event instances for both ML training and testing. Therefore, according to our experimental experience (in our experiment, we set frame rate of video to 30 frames per second, and bitrate to 800 kilobits per second.), we set the following rules to extract event instances from video eSSIM sequences:

- The minimum length of a defect event is 10.
- The maximum length of a defective event is 100, about 3 seconds.
- The minimum length of boundary between two events is 10.
- When *eSSIM* >= 0.95, distortion is too small to be perceived. Therefore, we regard these frames as normal ones with *eSSIM* = 1.
- The first 10 frames of each video are ignored, because they can be counted as frames after a scene change.

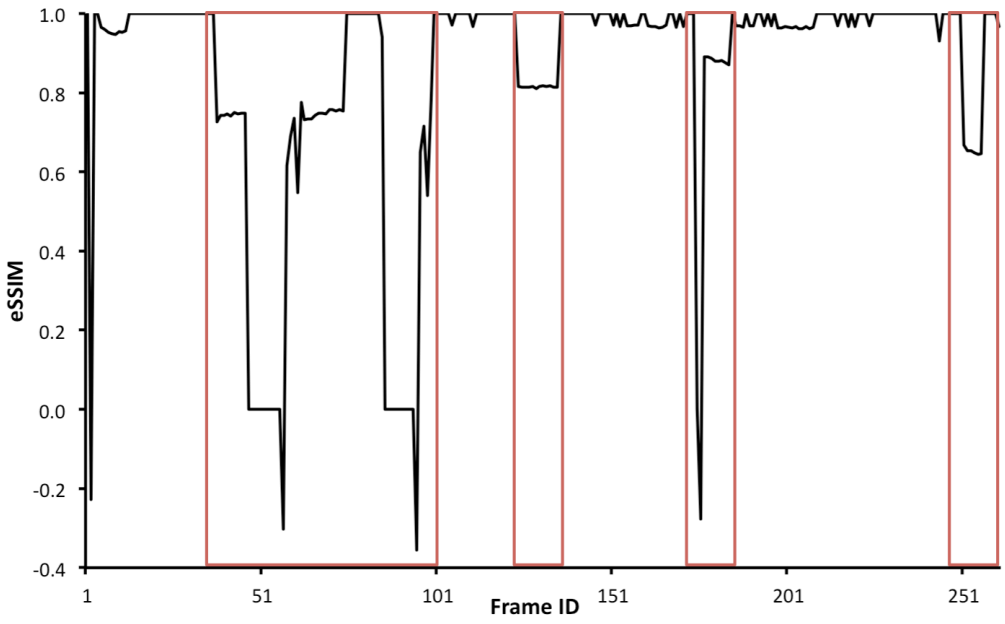


Figure 5. An example of event segmentation

An example of event segmentation in a transmitted *Foreman* video is shown in Figure 5. The X-axis is the frame id of this video, and the Y-axis is eSSIM value of each frame. From this eSSIM sequence, we can extract four defect events, which are marked by the red frame. According to our segmentation scheme, the first few frames are ignored although there is a frame dropped among them. Besides, in the first defect event, we can see two series of duplicated frames, but we integrate them as a single event because the boundary between them is too short to be distinguished by users.

3.2.4 Feature Extraction

As shown in Figure 5, we can consider a defective event as a time series data. Traditional classification algorithms on time series data usually use a distance measure, i.e. Euclidean Distance. But it is impractical for our problem. First, the time series data is very long (high dimensionality), so it slows down the speed of computation. Second, it is sensitive to handle time series with missing or noisy data if actual points are used as input. Third, it cannot process time series data with different length, because it usually requires a one-to-one alignment. But for our case, defect events do not have a uniformed start point and do not follow a specific pattern, as we can see in Figure 5. There are many reasons for these characteristics of our data, including packet-loss model used in our experiment and the loss rate we set, size of a group of pictures (GOP), and event segmentation methods.

Therefore, to provide a method that can handle time series data with varying length, be robust to missing or noisy data, and ensure the computing efficiency, we find a path on using statistical measures to extract features from event data. Basically, the set of features extracted from defective events should follow several considerations:

- First, it can best capture the global characteristic of event data;
- Second, it can discern similarity and difference between events;
- Third, it should be calculated in a normalized way, since the length of events can be different.

Here are the features we extract from event data:

1) ***μ*:**

$$\mu = \frac{\sum_1^n eSSIM_i}{n}. \quad (4)$$

where n is the total number of frames in an event, $eSSIM_i$ is the *eSSIM* value (range from -1 to 1) of the ith frame in the event. If μ is close to 1, the distortion is not heavy, meaning that glitch may happen.

2) ***δ*:**

$$\delta = \sqrt{\frac{\sum_1^n eSSIM_i^2}{n} - \mu^2}. \quad (5)$$

3) ***Minimum*:** the minimum *eSSIM* value of an event. If there are frames dropped or duplicated in an event, the minimum *eSSIM* can be an indicator of how severity the discontinuity is.

4) ***Defective ratio*:**

$$ratio = \frac{N_{eSSIM<0.95}}{n}. \quad (6)$$

where $N_{eSSIM} < 0.95$ is the number of frames with *eSSIM* < 0.95. It shows the severity of distortion in terms of the number of frames. 0.95 is selected because frames with *eSSIM* > 0.95 have no distortion perceivable, according to our experiment.

5) ***Severity of dropped frames and duplicated frames*:**

$$severity = \frac{N_{eSSIM\leq 0.0}}{n}. \quad (7)$$

where $N_{eSSIM} <= 0.0$ is the number of frames with *eSSIM* <= 0.0. It indicates the severity of freezing.

6) ***Skewness*:** it is a measure of the asymmetry of the probability distribution of event data. Negative *skewness* shows that the mass of distribution is concentrated on the right of the distribution, and vice versa [20]. Therefore, both *skewness* and the following *kurtosis* are calculated based on the probability distribution of event data. According to our experimental experience, the probability distribution is generated based on a range of varying unit:

[-1.0, 0.0, 0.1, 0.5, 0.8, 0.9, 0.95, 0.98, 1.0]

7) ***Kurtosis*:** it is a measure of whether the data are peaked or flat, relative to a normal distribution [21]. Both *skewness* and *kurtosis* represent some characteristics of *eSSIM* distribution in an event.

By applying these statistical measurements to defective event dataset with high dimension and different length, a new dataset with a limited number of features (low dimension) are generated.

3.2.5 Preprocessing for Features

During generation of above seven features, we have considered calculating them in a normalized way by dividing values by *n*. But both the range of *skewness* and *kurtosis* are still different from that of the others. This unbalanced situation can decrease the efficiency of classification. For instance, the Euclidean Distance function treats every dimension equally, and when we apply it in k-NN to calculate distance between two instances, *skewness* and *kurtosis* can affect the result more than the others do. On the other hand, except the unbalance of range of these seven features, there is also unbalanced mean among them. For instance, mean of *defective ratio* is much higher than that of *freezing severity*.

According to these two different types of unbalance, two types of normalization are applied to the feature dataset, respectively:

Middle-range Normalization: this filter normalizes the dataset with a certain middle value and a certain range for each feature (Figure 6). In our experiment, we set the middle value 0 and the range 2. Therefore, all features of the dataset are normalized to the range [-1, 1].

```
procedure MiddleRangeNormalization(normalMiddle, normalRange)
        Instance max = combined maximum of each feature
        Instance min = combined minimum of each feature
        currentRange = max – min
        currentMiddle = (max + min) / 2
        for instance in dataset
                newInstance = (instance-currentMiddle)/currentRange*normalRange+normalMiddle
        end for
end procedure
```

Figure 6. Middle-range normalization

Mean-std Normalization: this filter normalizes all features of the dataset with mean 0 and standard deviation 1 (Figure 7).

```
procedure MeanStdNormalization( )
        mean = average of each feature
        std = standard deviation of each feature
        for instance in dataset
                newInstance = (instance-mean)/std
        end for
end procedure
```

Figure 7. Mean-std normalization

3.2.6 Feature Reduction

Although we only extract seven features to limit the size of dataset, these features are not totally independent. For instance, it is very likely that a high *defective ratio* or *freezing severity* is accompanied by a small μ. The dependence among several features often unduly influences the accuracy of classification [15]. Compared to SVM, k-NN is very sensitive to noisy data or irrelevant features [13]. Therefore, a greedy forward selection of features is applied to k-NN. For the greedy forward selection algorithm, the best individual feature is selected first, and then the second one is chosen from the rest to

be combined with the best individual feature. After the best combination of two features is selected, we choose the third one from the rest to find the best performance of three features. Afterwards, the input subsets with four, and more features are evaluated. Finally, the best one is selected from all the combinations.

3.3 Multi-class Classification

The SVM, as a state-of-the-art classification method, is widely used and usually performs well in many fields. To be precise, the advantages of SVMs include that it performs well when there is a nonlinear relationship between the input and output, and that it is capable of dealing with high dimensional data. Therefore, we choose the SVM as one of our candidate techniques. To get good performance of SVMs, we need to make right decisions in the following steps: preprocessing data, kernel selection, parameters setting of the SVM and the kernel. Uniformed choices may result in severely reduced performance [22]. Since how to preprocess data has been introduced in previous parts, we will focus on kernel selection and parameters setting. Besides, because our problem requires multi-class classification and SVMs are just for binary classification, we need to choose the best approach to construct a multi-class model from binary SVMs.

3.3.1 Kernel Selection

A kernel function for SVMs is used to map instances of a dataset from the original feature space to a higher dimensional one. Then, a decision boundary is constructed in this high dimensional feature space to distinguish between two classes. According to the way of mapping, kernel functions can be classified into linear (i.e. linear kernel) and non-linear ones (i.e. polynomial kernel and radial basis function (RBF)). Based on our experiment with optimized parameters in the next chapter, linear kernel gives the worst result (accuracy: 74.45%±0.63%), while RBF kernel performs the best (accuracy: 85.46%±0.29%) and polynomial kernel is in the second place (accuracy: 83.88%±0.36%). The reason for linear kernel's poorest performance is that our instances are not linearly

separable even in the higher dimensional feature space. On the other hand, both polynomial and RBF kernel perform better because they can construct a non-linear and flexible boundary between two classes. Therefore, we choose RBF kernel for our classification. RBF kernel is shown as the following:

$$k(\vec{x}, \overrightarrow{x_i}) = exp(-\gamma \|\vec{x} - \overrightarrow{x_i}\|^2), \quad (8)$$

where $\gamma = 1/2\sigma^2$.

3.3.2 Parameters setting

For the SVM with RBF kernel, there are two parameters to set: *C* for the SVM and γ for the RBF kernel. *C*, termed as the soft margin constant or the penalty factor, controls how strict each instance to be classified correctly. When *C* is small, points close to the boundary are ignored. γ controls the flexibility of the decision boundary. When γ is small, the decision boundary is nearly linear. Underfitting happens when both *C* and γ are small. When γ increases, the flexibility of the decision boundary increases. But overfitting occurs when γ is too large (Figure 8). Therefore, we can find that the classification accuracy of SVM with RBF kernel is closely related to the setting of parameters.

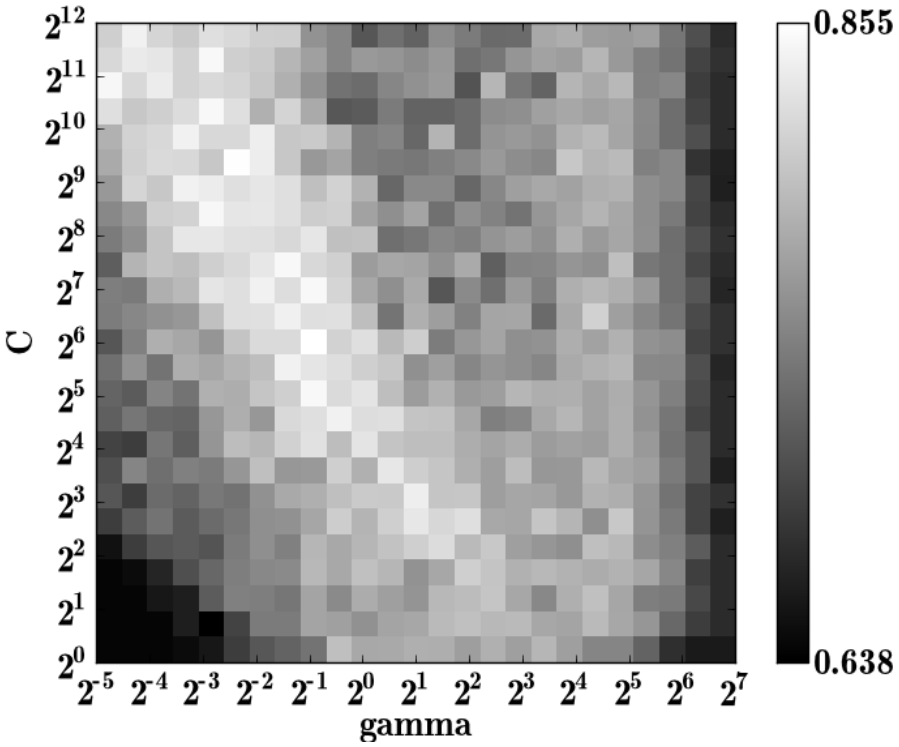


Figure 8. Grid search for C and gamma

To choose the best combination of C and γ, a common approach is using a grid search. The grid search is usually based on exponentially growing sequences of C and γ, i.e. in our experiment, as shown in Figure 8,

$$C \in \{2^0, 2^{0.5}, \ldots, 2^{11.5}, 2^{12}\},$$
$$\gamma \in \{2^{-5}, 2^{-4.5}, \ldots, 2^{6.5}, 2^7\}.$$

Therefore, the time complexity of a grid search is $O(n^2)$.

As what we mentioned in the first chapter, one basic requirement of our model is as fast as possible to meet the challenge of video streaming. Thus, the grid search, as an exhaustive and therefore potentially expensive method, is not suitable anymore. But according to Figure 8, we can see that the best combinations of C and γ are not randomly

distributed around all the area, but located in a restricted area. In fact, the result of a SVM with a linear kernel can serve as a baseline [16]:

$$\log \sigma^2 = \log C - \log \tilde{C}, \tag{9}$$

where $\tilde{C}$ is the penalty parameter of the SVM with a linear kernel. In Figure 8, the baseline with best combinations of C and γ located, can be defined by (9), when $\tilde{C}$ is set to the optimal value of penalty parameter for the linear SVM. Therefore, we can find best combinations of C and γ following this procedure:

- Search for the best $\tilde{C}$ of the SVM with a linear kernel.
- Fix $\tilde{C}$ from last step in (9), and search for the best combination of C and γ that satisfies (9).

Compared to the quadratic time complexity $O(n^2)$ of the grid search approach, the time complexity of this optimized method is $O(n)$.

3.3.3 Combination of Binary Classifiers

SVMs are originally designed for binary classification. To meet the requirement of classifying multiple classes, a common approach is using a combination of several binary classifiers. There are two popular strategies of combination:

1) **One-versus-all (OVA):** OVA combines M (the number of classes) binary classifiers. For each classifier, it chooses a class as the positive class, and the remaining as the negative class. For our problem, three binary classifiers are built: distortion vs. rest, glitch vs. rest, and freezing vs. rest.
2) **One-versus-one (OVO):** OVO combines M(M-1)/2 binary classifiers. For each classifier, it selects a pair of classes from all the classes without repeat: one as the positive class, and the other as the negative class. For instance, in our problem set, we have the following three OVO classifiers: distortion vs. glitch, distortion vs. freezing, and glitch vs. freezing. After these classifiers are constructed, a simple voting strategy is applied to evaluate the result. For each instance, we predict it in the class with the largest vote.

By comparison, OVA may be better than OVO, in terms of the time complexity, especially for the dataset with many types of classes. But OVO has been shown to perform better than OVA, in terms of the classification accuracy [23]. According to our experiment, we believe that one reason of the poor performance of OVA is the unbalanced dataset for its binary classifiers. In our case, the same problem for OVA exists. For instance, in our experiment, there are totally 238 defective events, but only 39 among them are *glitch*. Unbalanced datasets can present a challenge for training a classifier [17], in terms of the classification performance. Besides, there are only three classes in our experiment: glitch, distortion and freezing (we combine discontinuity with freezing, because discontinuity always happens after freezing). Therefore, both for OVA and OVO, only three binary classifiers are constructed. Considering these, we select OVO as the combination method for our multi-class classification.

3.3.4 Multi-class Classification with k-NN

k-NN is a method for classifying points (instances) in the feature space by comparing their relative distance. The basic principle behind it is that instances which are close to each other have similar features. Because of this, k-NN can be very sensitive to noisy instances or imbalanced features.

Two decisions need to be made when use k-NN: *k* selection and distance function selection. A smaller *k* is more sensitive to noisy instances, but it can deal with situations, like the region defining some class is so small that instances of other classes that surround the region win the majority vote. The objective of the distance function is to minimize the distance between instances of the same class, and maximize the distance between instances of different classes. Since the classification result is sensitive to the different combination of *k* and distance functions, we use a grid search to find the best one ($k \in \{1, 2, \ldots 9, 10\}$, and distance function $\in$ {Minkowsky, Manhattan, Chebychev, Euclidean}) [13].

3.4 User Model

In this part, we will investigate the user model of VIDAR project. Based on defect events that have been classified by the Vidi model, we will show that how these events, including *Distortion*, *Glitch* and *Freezing*, are related to user MOS, respectively.

3.4.1 Distortion

For *Distortion* defect, to get an accurate estimated user MOS (eMOS), we have to measure the intensity of defect events. Although μ is the main contributor to classify different types of defect events according to the next chapter, it is not enough to indicate the exact intensity of a *Distortion* event, because of the complexity of human vision system (HVS). Therefore, besides the features we extract from the event eSSIM sequence, we also try to add easily operated factors that have information about HVS. According to our experimental experience, different regions and different frames may be of different importance to the human observers. For instance, with similar eSSIM sequence, *Distortion* that occurs during a scene movement generally is less perceivable than that happens with the scene fixed. Besides, *Distortion* within frames with a low frame complexity (i.e. *Football*) is more perceivable than that with a high frame complexity (i.e. *Bus*). Considering these HVS-related reasons, two factors are introduced to our feature set to measure the exact *Distortion* intensity: frame complexity and motion speed [4]. As a result, we have four features for each *Distortion* event:

$$\{\mu, ratio, frame\ complexity, motion\ speed\}.$$

To estimate user MOS based on these four features, we select artificial neural network (ANN) as our approach, because the linear correlation coefficient between these features and user MOS is not clear, according to our experiment in Chapter 4. Besides, ANNs usually perform well when a nonlinear relationship exists between the input and output features.

3.4.2 Glitch

Similarly, *Glitch* is a variant of the *Distortion* type in that it has a short length of distortion. Therefore, features used to indicate the intensity of *Glitch* event is similar to those of *Distortion*.

3.4.3 Freezing

According to our initial experiment, we observed that sometimes *Freezing* defect happened with *Distortion* in the same event, i.e. the first segmented event in Figure 5. Because we have discussed the intensity of *Distortion* in the previous part, we will only consider *Freezing* events without *Distortion* accompanied. However, as another part of our observation, *Freezing* is always followed by *Discontinuity* defect. Therefore, in this thesis, we combine *Freezing* and *Discontinuity* as a single type of defect event.

For this kind of *Freezing* events, calculating its intensity is quite straightforward. Two features are extracted from *Freezing* eSSIM sequences:

- **The time duration of *Freezing*:** it can be directly computed given the number of duplicated frames in sequence and the frame rate of the video itself. However, in this thesis, we set the same frame rate for all testing videos. Therefore, we only need to calculate the number of duplicated frames.
- **Discontinuity:** it is quantified based on the difference between two frames that are not adjacent in the original video, but are played back to back in the transmitted video. The intensity of discontinuity is computed by comparing the SSIM difference between these two frames.

4 Validation

In this chapter, we show our experiment results of both eSSIM aggregator and the user model. First, we show our experiment setup for event-based classification, and that how the event dataset is generated. Then, we compare the result of multi-class classification and investigate the effect of our optimized steps on the classification efficiency. Finally, we analyze that how our classified defect events are related to user MOS.

4.1 Validation of eSSIM Aggregator

4.1.1 Experiment Setup and Dataset Generation

As experiment setup, original videos are encoded at the server side into H.264/AVC using the x264 library [27] of VLC media player. Then, these videos are streamed to the client side through RTP over UDP. The client side decodes the streamed videos using FFmpeg [28], and plays the video with VLC media player.

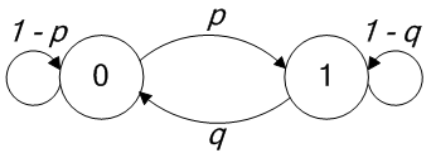


Figure 9. Gilbert-Elliot burst loss model

To simulate different network packet loss situations, two packet loss models are applied to our experiment:

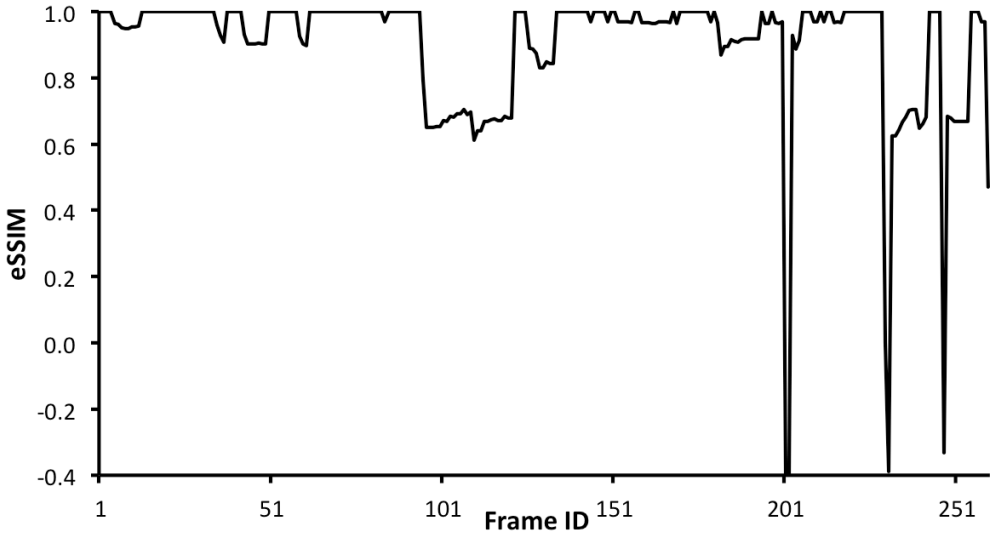
- **Bernoulli uniform loss model:** this model performs random loss patterns, and we set the loss rate to 2% for our experiment.
- **Glibert-Elliott (GE) burst loss model:** compare to uniform model's random loss pattern, GE model results in a burst of packet loss. Two parameters, p and q, control its loss probability and loss burstiness, respectively (Figure 9). In our

experiment, we set the loss rate to 2%, which is the same as the uniform model, and the burstiness rate to 45%.

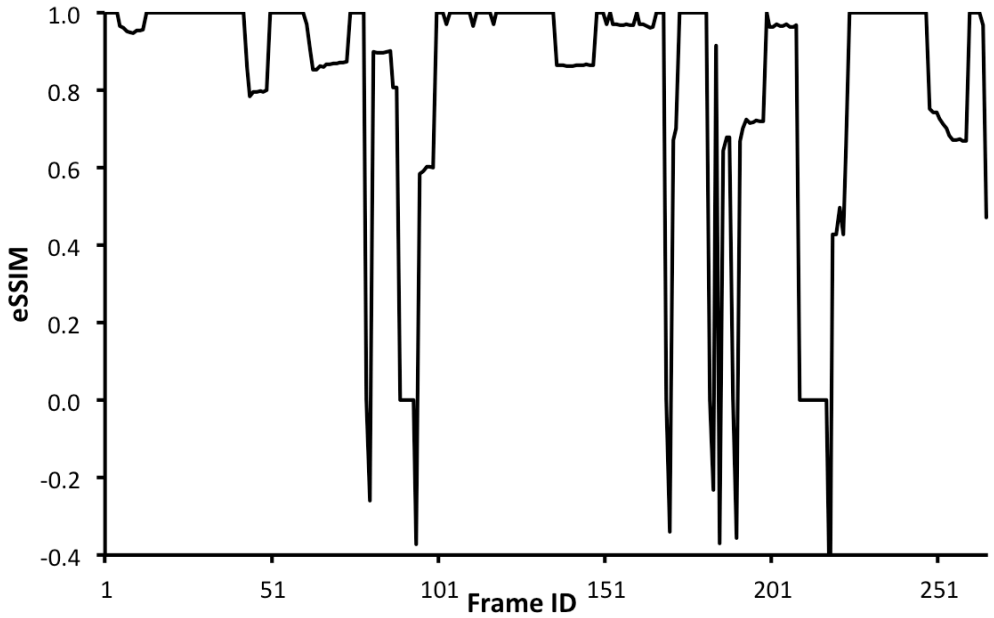
Through the experiment, we find that generally GE model creates severer defects than the uniform model. To be specific, as a result of burst loss, GE model is more likely to introduce *freezing* or heavy *distortion* into the transmitted videos that are coded by H.264/AVC. On the other hand, because of the random loss pattern, *glitch* defects are more likely caused by the uniform model. In Figure 10, we can see that, for the uniform model, there are only three frames dropped, and the distortion is not severe in terms of the number of frames with defects. However, for the GE model, we can find two series of consecutive duplicated frames, which may result in a *freezing* perception for users. Besides, there are also more frames discarded, and the ratio of frames with distortion is also higher.

During encoding at the server side, we set two different size (12 and 24) of GOP for each video, because the varying GOP size has a great influence on the severity of defects in transmitted videos. To be precise, with GOP size 12 and 2% uniform loss, a small number of frames are dropped and the sequence of consecutive distorted frames is not very long. However, with GOP size 24 and 2% uniform loss, more frames are discarded and the distortion is severer.

For the videos streamed to the client side, we use reference videos with lossless H.264 encoded from [24]. Considering user MOS is closely related to the content of videos, we select a range of videos with varying contents, including *Bus*, *Container*, *Flower*, *Football*, *Foreman*, *Mother & Daughter*, and *Stefan*. However, all of these videos have no scene cuts. Therefore, to investigate the effect of scene cut on user's perception of videos under the same network condition, we extract a short video from *Inception*, where there is six scene cuts. In Table 1, we show the detail information of these videos, including number of scene cuts, scene moving speed, object moving speed, average frame complexity, and the main content. The maximum length of these videos is 10 seconds of *Foreman*, while the minimum length is 3 seconds of both *Football* and *Stefan*.



(a) Foreman, GOP=12, 2% uniform loss model



(b) Foreman, GOP=12, 2% GE loss model

Figure 10. Comparison of two different models

Table 1. Detail information of streamed videos

Name	Number of Scene Cuts	Scene Moving Speed	Object Moving Speed	Average Frame Complexity	Content
Bus	0	Middle	Fast	27.95	Bus
Container	0	No	Slow	16.18	Ship with containers
Flower	0	Middle	Slow	22.14	Flower and house
Football	0	Fast	Fast	11.96	Football players
Foreman	0	Slow	Slow	18.73	Portrait and construction
Mother &Daughter	0	No	Slow	12.60	Portrait
Stefan	0	Slow	Middle	23.62	Tennis player
Inception	6	Fast	Fast	18.00	Portrait, construction and etc.

To focus on investigating the effect of varying network packet loss conditions on user MOS and keep the other parameters the same, we transcode all these videos with frame rate 30 (frames per second), and bit rate 800 (kilobits per second). A screenshot of each tested video is shown in Figure 11.

After getting eSSIM sequence of the transmitted video from R3 model, we pass it through the content and discontinuity filters of Vidi model, and then defect events are segmented from it. Each video is tested with different network packet loss conditions (uniform model and GE model) and different GOP size (12 and 24). Finally, 238 defect events are generated from these transmitted videos, including 151 distortion events, 39 glitch events, and 48 freezing events. These defect events are labeled manually by people with multimedia research experience. The unbalance between the number of *distortion* and *glitch* events shows that 2% packet loss for both the uniform model and GE model is

severe enough to create perceivable distortion.

Figure 11. Screenshots of tested videos

4.1.2 Multi-class Classification (SVM)

To compare the efficiency of SVM multi-class classification with different kernels and different parameter setting strategies, we construct three multi-class models based on sequential minimal optimization (SMO). SMO is an algorithm for efficiently solving the optimization problem that arises during the training of SVMs [25].

1) **SMO-G:** SMO with RBF kernel, and the best combination of C and γ is selected by Grid search (SMO-G). The range of C and γ for searching is as following:

$$C \in \{2^0, 2^{0.5}, \ldots, 2^{11.5}, 2^{12}\},$$
$$\gamma \in \{2^{-5}, 2^{-4.5}, \ldots, 2^{6.5}, 2^{7}\}.$$

As we mentioned in Chapter 3, when both C and γ are too small, underfitting will happen, and when γ is too large, overfitting will occur. Therefore, according to parameter setting experience of other related work, we restrict the range of C and γ as above.

2) **SMO-L:** SMO with Linear kernel (SMO-L). Compared to RBF kernel or polynomial kernel, one important difference between linear kernel and them is that only one parameter C, the penalty factor, needs to be set. Therefore, we use a fine granularity for searching the best C.

$$C \in \{2, 4, 6, \ldots, 198, 200\}$$

3) **SMO-O:** SMO with RBF kernel, and the best combination of C and γ is set by the Optimized method that is mentioned in Chapter 3 (SMO-O). First, we fix $\tilde{C}$ in Equation (9) with the best C selected by SMO-L. Then, compared to SMO-G, we use a fine granularity for searching the best combination of C and γ.

$$\gamma \in \{0.05, 0.10, \ldots, 1.95, 2.00\}$$

When γ is fixed with one value from the above range, we can calculate the corresponding C according to Equation (9).

For all three methods, we conduct a ten-fold cross-validation on the dataset. Cross-validation is used to assess how accurately a predictive model will perform in practice [26], because our dataset is small and therefore overfitting of classification can happen easily. Briefly, cross-validation works like this:

- Partition the dataset into complementary subsets
- Perform classification training on one subset
- Validate the training model on the other subsets

The implementation is based on Weka open source library [19]. Because cross-validation applies partition randomly, we test all three models several times to get the average result. In Table 2, we show both the classification accuracy and the time consumption of four multi-class classification models: SMO-G, SMO-L, SMO-O, and k-NN (the result of k-NN will be analyzed in the next part).

According to Table 2, we can see that SMO-G performs the best in terms of the

classification accuracy, but its time consumption is much higher than the other models. The best performance shows the advantage of grid search, but it also explains why an optimized method to find the best combination of C and γ is possible. In fact, according to our multiple tests of SMO-G, the combination of C and γ with the best accuracy varies for each test. The following are some examples in the format of (C, γ):

$$(2^5, 2^1), (2^6, 2^{-1}), (2^8, 2^{-2}), (2^9, 2^{-3}), (2^{10}, 2^{-3})$$

But we can find that the product of C and γ is either 2^6 or 2^5. This result just coincides with Equation (9). Actually, Equation (9) can be converted to:

$$2\times\gamma\times C = \tilde{C}, \tag{10}$$

where $\gamma = 1/(2\times\sigma^2)$. As we fix $\tilde{C}$ according to the best result of SMO-L, the product of C and γ can be constant. Depending on the granularity for searching the best combination, the product may fluctuate slightly, like 2^6 or 2^5 in our case.

Among all the SVM-based models, SMO-L gives the worst classification accuracy, but its time consumption is the least. The reason is that the computational complexity of linear kernel is lower than that of RBF kernel, and there is only one parameter C to set.

As an optimized model, SMO-O shows almost the same accuracy as SMO-G, and keeps its time consumption at the same level as SMO-L. Therefore, in terms of both classification accuracy and time consumption, SMO-O performs the best.

Table 2. Classification result

	SMO-G	**SMO-L**	**SMO-O**	**k-NN**
Accuracy	85.46%± 0.29%	74.45%± 0.63%	84.57%± 0.29%	83.00%± 0.33%
Time (s)	445.2±16.0	66.7±2.7	94.5±4.5	31.6±3.2

For the above experiment, we set parameters for each binary classifier uniformly to keep it easy to control. Now we investigate each one separately to see what affects the classification accuracy. According to OVO combination strategy of binary classifiers, there are three binary SVM classifiers generated in our case, including *Distortion* vs.

Glitch, *Distortion* vs. *Freezing*, and *Glitch* vs. *Freezing*. Each one is with RBF kernel, and grid search is applied to search for the best setting of parameters.

Table 3. Classification result of each binary classifier

	Distortion* vs. *Glitch	***Distortion* vs. *Freezing***	***Glitch* vs. *Freezing***
Accuracy	92.04%±0.24%	85.38%±0.32%	97.18%±0.21%

In Table 3, we can find that *Glitch* vs. *Freezing* has the best accuracy, which is near 100%. This means that *Glitch* and *Freezing* events can be distinguished from each other easily. In fact, according to our experimental experience, generally *Freezing* events are caused by a series of duplicated frames, like the *Freezing* event found in Figure 12, while *Glitch* event is caused by a short sequence of slightly distorted frames. Therefore, some features in our feature set have distinguishable values for these two types of defect event. For instance, *Freezing* usually has a lower value of feature μ because of duplicated frames with eSSIM = 0, while *Glitch* has a higher value of μ.

As for *Distortion* vs. *Glitch*, it has a relatively high classification accuracy. One important factor that affects the accuracy, we believe, is that user's perception of video quality is highly content-related. To be precise, the following points can be factors of video content:

- **Frame complexity:** the frame complexity has a direct correspondence to how perceivable a distortion can be. For instance, the second *Distortion* event in Figure 12 (frame complexity of *Foreman*: 18.73) and the *Glitch* event in Figure 13 (frame complexity of *Flower*: 22.14) have similar eSSIM value. However, we observe significant difference in defect visibility, and that is why we label one as *Distortion* and the other as *Glitch*. According to our experience, videos with higher frame complexity can reduce the severity of distortion, while distortion is more perceivable in videos with lower frame complexity.
- **Content types of video:** different types of content have different influence on user's perception of video quality. In fact, it depends on which object users focus

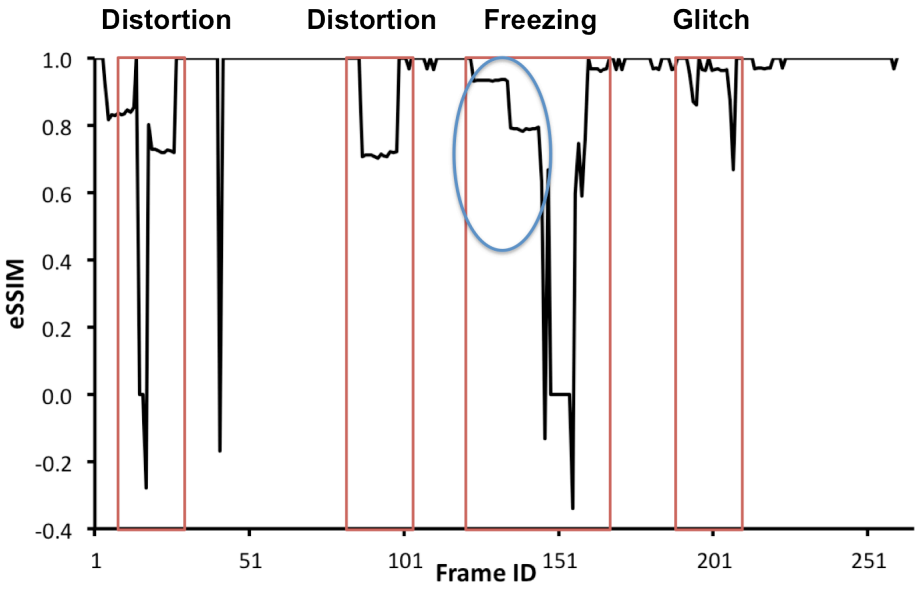


Figure 12. Event examples (Foreman, GOP=12, 2% GE loss model)

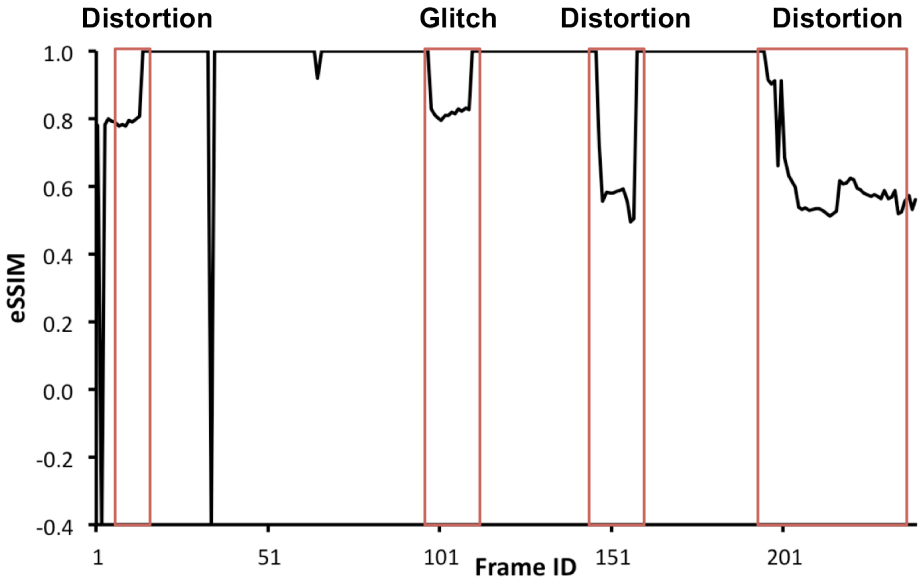


Figure 13. Event examples (Flower, GOP=12, 2% uniform loss model)

on. For instance, the main object of *Foreman* is the person who is talking, while the object of *Flower* is just the ground full of flowers (Figure 11). For distortion with similar eSSIM values, it is more perceivable when it occurs in *Foreman* than that in *Flower*.

Finally, for *Distortion* vs. *Freezing*, it shows the worst classification accuracy. The main reason, according to our analysis of the dataset, is that *Freezing* may happen accompanied by *Distortion* in the same event. For example, in Figure 12, although we labeled it as a *Freezing* defect, distortion is also observed in this event. The area marked with the blue ellipse shows where the distortion is. This coexistence characteristic in the dataset makes the binary classifier difficult to distinguish between *Freezing* and *Distortion*.

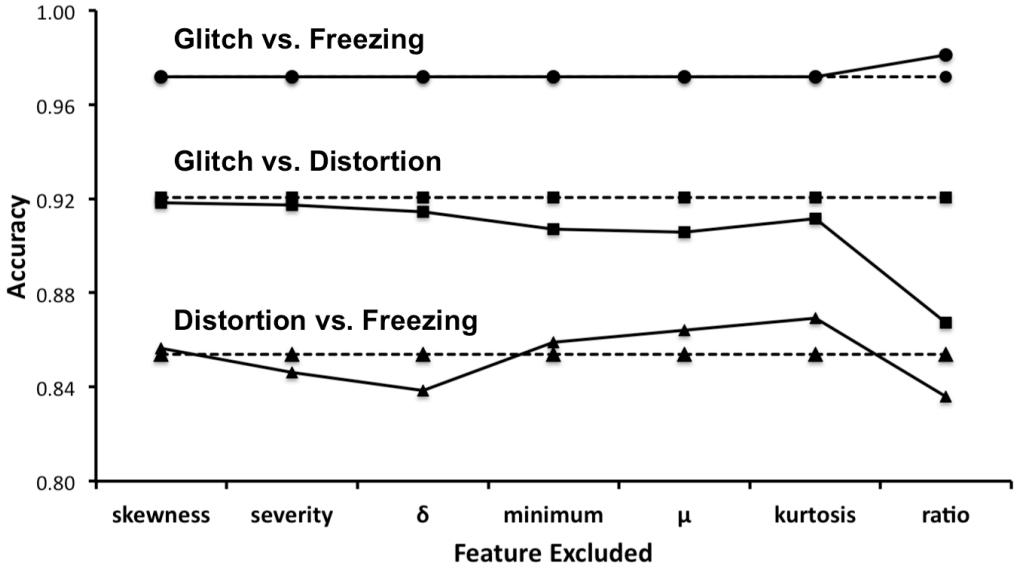


Figure 14. Feature sensitivity under different binary classifiers

Now, we study how each binary SVM classifier is sensitive to all seven features. For each test, we remove one from the feature set, and use the remaining as the dataset for classification. In Figure 14, the dashed line shows the classification result without any

feature removed, while the solid line is the classification accuracy after each feature is removed. As we can see in this figure, without feature *skewness*, all three binary classifiers retain the same accuracy, which means that *skewness* is a redundant feature for our multi-class classification. In contrast, the removal of feature *severity*, δ and *ratio* has a negative influence on the classification accuracy of both *Distortion* vs. *Freezing* and *Distortion* vs. *Glitch*. It means that these features can provide certain useful information for multi-class models. Finally, the removal of feature *minimum*, μ and *kurtosis* increases the accuracy slightly. The reason is that both *Distortion* and *Freezing* have similar value of these features. On the other hand, the removal of these features decreases the accuracy of *Distortion* vs. *Glitch*, meaning that they are important for this binary classifier. To be precise, *Glitch* defect usually does not have *minimum* or μ for small value, while dropped or duplicated frames with eSSIM = 0 may exist in *Distortion* event.

4.1.3 Multi-class Classification (k-NN)

For multi-class classification with k-NN, we use grid search to select the best combination of k and distance function:

$$k \in \{1, 2, \ldots 9, 10\},$$

$$\text{distance function} \in \{\text{Minkowsky, Manhattan, Chebychev, Euclidean}\}.$$

Combined with grid search, a greedy forward feature selection approach is applied to find that which features contribute to the classification the most. Through experiment, we find that the best combination of k and distance function is (5, Manhattan), and the result is shown in Table 2. Compared to the other SVM-based multi-classifiers (SMO-G, SMO-L, SMO-O), k-NN still has a relatively good performance in terms of the classification accuracy, while it shows the best time efficiency because of its simple mechanism of classification. The result of feature selection is shown in Figure 15. As a single feature, *ratio* is the most important contributor to the classification. After combined with *minimum* that is selected from the remaining features, the accuracy improves significantly. Finally, the best accuracy happens with this feature combination:

{*ratio*, *minimum*, μ, *severity*}.

Besides, we can also see that δ, *skewness* and *kurtosis* introduce some noisy information to k-NN.

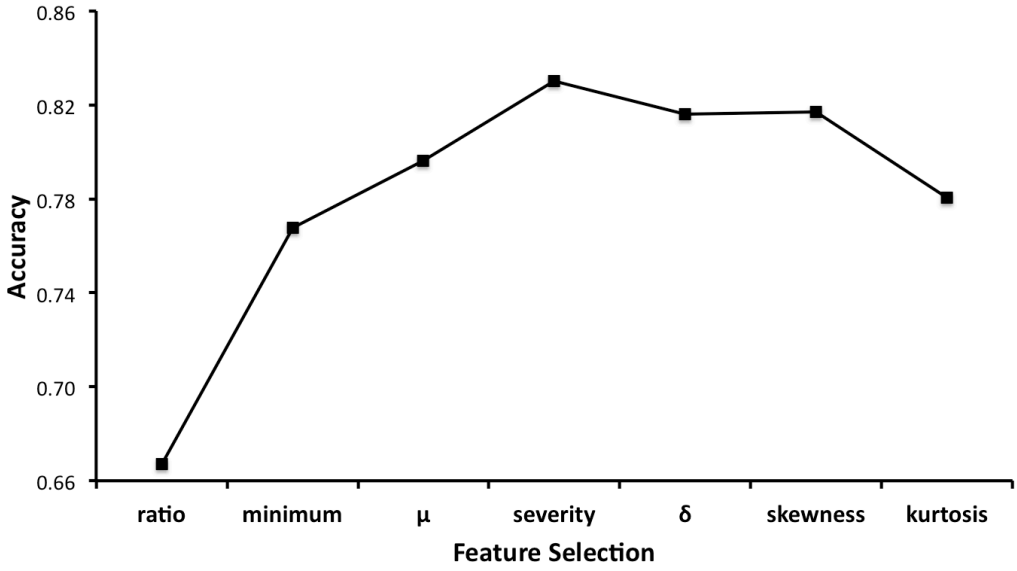


Figure 15. Greedy forward feature selection for k-NN

4.2 Validation of User Model

In this part, first, we will show that how we conduct the subjective testing for user MOS of defect events. Then, based on these subjective data, we will validate our models proposed for estimating user MOS of each type of defect, respectively.

4.2.1 Subjective Testing

We adopted a single stimulus procedure to obtain subjective quality ratings for different defect events. The choice of a single stimulus paradigm is well suited to state-of-the-art multimedia applications, including IPTV, Internet streaming, etc. In addition, it significantly reduces the amount of time required to conduct the study, compared to a double stimulus study. Considering that it is impractical for human

subjects to indicate the quality of the video on a continuous scale, and we also need relatively fine gradations, the MOS with the range 1 to 10 is used instead of 1 to 5(Table 4).

Table 4. MOS with the range 1 to 10

MOS	**1, 2**	**3, 4**	**5, 6**	**7, 8**	**9, 10**
Quality	Bad	Poor	Fair	Good	Excellent
Impairment	Very annoying	Annoying	Slightly annoying	Perceptible but not annoying	Imperceptible

We select 173 defect events, with length less than 3 seconds, from the dataset that has been classified by eSSIM aggregator, including different types of defect and videos with different content (Figure 11). We prepare a playlist for each subject by arranging these 173 events in a random order. To make sure that subjects give their MOS with a consistent baseline, we repeat the first 20 defect videos and insert them into the middle of the whole playlist, and we compare the difference between the first 20 and their repeats randomly distributed to see how big the drift is. If the drift is high, we will reject this experiment result and conduct a new one. In addition, considering that the playlist is long, we insert segments of the original videos without any defect into the playlist randomly, to serve as fixture of reference for the viewer. Because subjects have no experience of judging the quality of videos, we also show the viewer samples with different types of defect videos before the test.

VLC is used as the video player, and all the videos are displayed at their native resolution to prevent any defect due to the performance of software or hardware.

4.2.2 Distortion

Among all the defect events used in the subjective test, there are 116 Distortion events. As what we expect, the MOS of Distortion is in the range of 1 (very annoying) and 5 (slightly annoying), which is just consistent with its definition we give in Chapter 2. The

number of Distortion events for MOS from 1 to 5 is 15, 34, 39, 13, 15, respectively. Now, based on the five-category dataset, we apply ANN to see how the four intensity-related features indicate the subjective user MOS. In Figure 16, we show the classification result

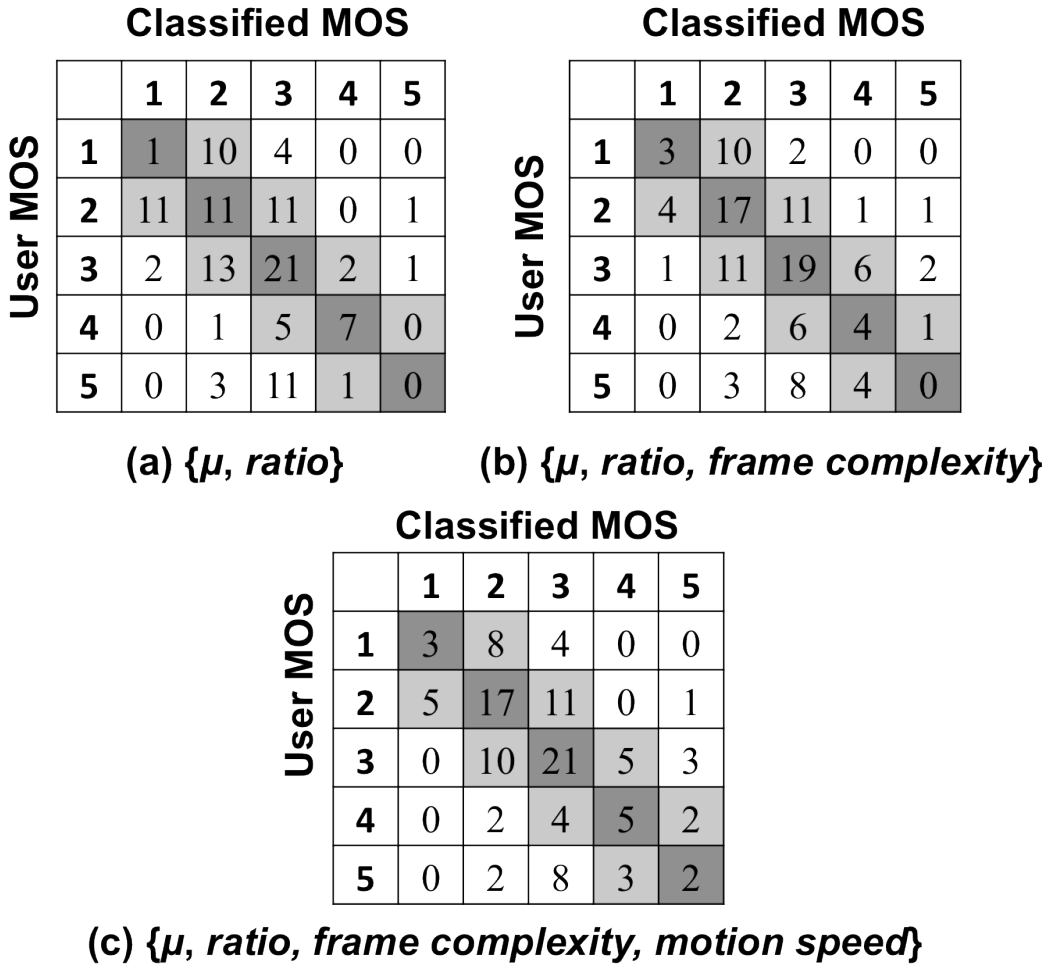


Figure 16. Classification result of user MOS on Distortion events

of ANN for datasets with different features. For each table, the first column is user MOS, while the first line is classified MOS. The dark grids show the number of events that have been classified correctly, while the others are the number of events misclassified. For

instance, according to the first line in Table (a), we know that there are 15 *Distortion* events with MOS = 1, and only one of them is classified correctly, while 10 and 4 among them are classified as events with MOS = 2 and 3, respectively. Based on these three tables, the exact classification accuracy for each one is 34%, 37% and 41%. As we can see, as we insert HVS-related features one by one, the result is getting better. However, the absolute accuracy is still very low. The following are basic reasons:

- HVS is so complicated that we cannot include all the factors, especially subject's point of interest on the video. For instance, generally users focus more on the character (i.e. *Foreman*) than the background. Therefore, *Distortion* on the character is more perceivable than that on the background, even with the same eSSIM sequence. However, it is still difficult to track user's point of interest on the video.
- For our subjective test, usually subjects can distinguish between good videos without any defect and bad videos with obvious distortion. However, it is difficult for users to tell the exact intensity of *Distortion*. For instance, for a video with slightly distortion, users may hesitate to set it as MOS=5 or MOS=6.

Considering the above reasons, we believe that classification results with a drift by ±1 are reasonable. For instance, for Distortion events with subjective user MOS=3, it is ok for ANN to classify them as events with MOS=2 or 4. Therefore, in Figure 16, the gray grids show the number of events that can be considered as correct classification results, under a drift by ±1. Then, the accuracy for these three feature sets is 79%, 83% and 83%, respectively.

4.2.3 Glitch

Because 2% network packet loss can easily result in a severe distortion in the transmitted video, Glitch happens not frequently as Distortion. Therefore, we only collected 26 Glitch events from the subjective test. As Glitch is defined as a short sequence of distortion, the expected user MOS of Glitch event should be usually higher

than that of Distortion defect, and our experiment just proves that. The user MOS for Glitch ranges from 6 (slightly annoying) to 9 (not annoying, even imperceptible), with a distribution of the number of events as the following: 1 for MOS=6, 6 for MOS=7, 13 for MOS=8, and 6 for MOS=9. With the same ANN applied, we got the result in (a) of Figure 17, which has the same format as Figure 16. Without a drift by ±1, the accuracy is 16 out of 26, while the classification result can be improved to 23 out of 26, considering the drift.

4.2.4 Freezing

There are totally 31 *Freezing* events used in the subjective experiment, and the range of user MOS for these events is from 3 to 7 (2 for MOS=3, 8 for MOS=4, 12 for MOS=5, 5 for MOS=6, and 4 for MOS=7). The intensity of *Freezing* is indicated by two features: the number of duplicated frames and discontinuity. Without distortion-combined *Freezing* events included, these two features are straightforward enough to indicate the intensity of *Freezing*. Therefore, we believe that it is practical to use ANN with such a small dataset. (b) of Figure 17 shows the classification result: 16 out of 31 without drift, and 30 out of 30 with ±1 drift considered.

4.2.5 Summary

In summary, for each type of defect, we extract most intensity-related features as input of ANN. However, because of the complexity of HVS and the limited judge granularity of human subjects, it is difficult to get a high classification accuracy. But with a ±1 drift, our feature-based method has a good performance on estimating user MOS.

On the other hand, our experiment also shows that three types of defect event have a distinguishable average user MOS (Figure 18). To be precise, *Distortion* can have a great influence on user's perception of video quality, while *Glitch* is the least perceptible, and *Freezing* without distortion shows a relatively high effect.

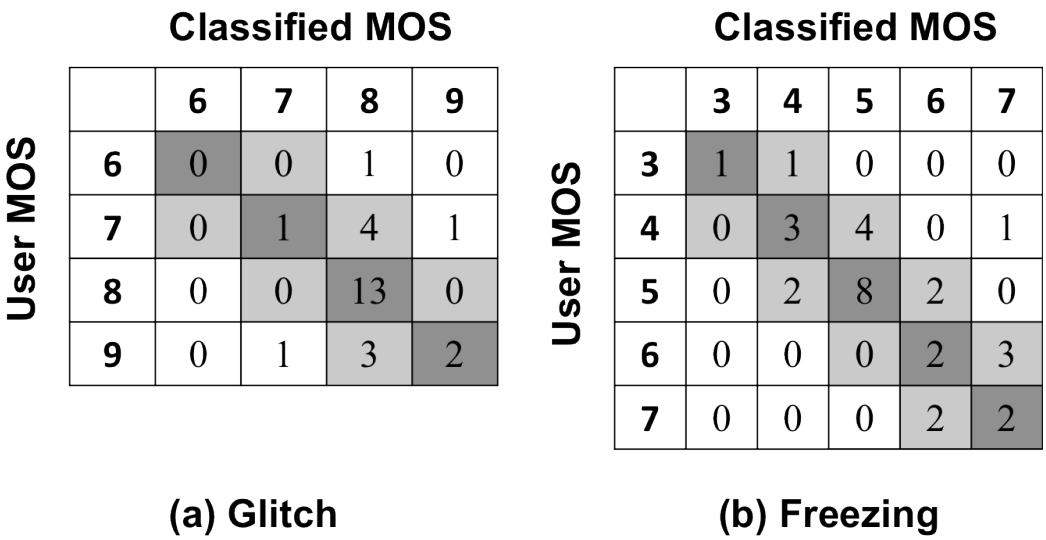


Figure 17. Classification result of Glitch and Freezing

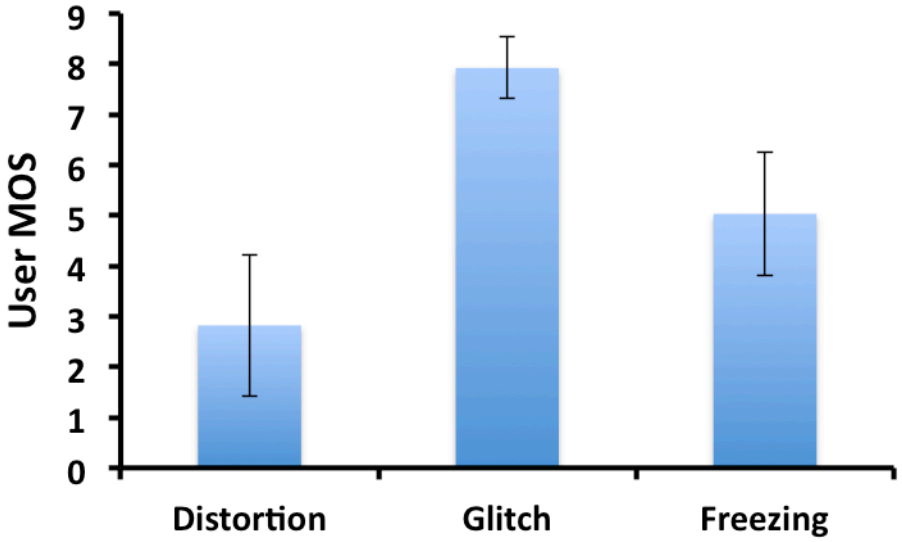


Figure 18. Average user MOS of defect events

5 Concluding Remarks

5.1 Summary

In this thesis, we presented an event-based model to correlate video quality metrics to user experience. This model is a part of VIDAR project, which uses an analytical approach to evaluate user experience of video services under varying network conditions. The perception model we propose is a method to map objective frame-level metrics to perceivable video defects with distinct user MOS characteristics, using ML techniques.

5.2 Contributions

Our contribution of this thesis composes of two parts: eSSIM aggregator and the user model.

eSSIM aggregator is designed to aggregate frame-level metrics to event-level metrics, and provide a classification of these defect events automatically. To make the whole approach practical for video streaming, we extract important features of defect events instead of using the original high-dimension eSSIM sequence. Considering that user experience of video quality is greatly affected by human vision system, we design several content filters, including scene change filter and discontinuity filter, to remedy the original objective metrics. Feature preprocessing and feature reduction are applied to improve the efficiency of classification. To classify different types of defect events, we construct a multi-class classification model from several binary SVM classifiers, with carefully kernel selection and optimized parameter-setting approach.

As for the user model, we analyze the intensity of different defect events, based the classification result of eSSIM aggregator. Features, which are highly related to user experience of video quality, are used to indicate the intensity.

Through experiments, eSSIM aggregator can provide a high classification accuracy of defect events while keeping the time consumption within a reasonable range. Based on

the good performance of eSSIM aggregator, the user model shows a strong correlation between the intensity of different types of defect events and the user MOS.

5.3 Future Work

For future work, because our current user MOS estimation is at the event level, which is a short sequence of video, an aggregator will be applied to predict user MOS of long videos. For this long-video user MOS estimation, a number of psychological theories may be used for modeling the complicated human perception system.

Besides, we need to conduct more comprehensive subjective tests and propose a reference model of user perception, considering the complexity of human vision system. To generate a reference model, we need find common response among users, and identify key user-specific features that specialize reference model to individual users (or groups of like-minded users) MOS.

References

[1] Z. Wang, L. Lu, and A. C. Bovik, "Video quality assessment based on structural distortion measurement," Signal Process.: Image Commun., vol. 19, no. 2, pp. 121-132, Feb. 2004.

[2] M. H. Pinson ans S. Wolf, "A new standardized method for objectively measuring video quality," IEEE Trans. Broadcast., vol. 50, no. 3, pp. 312-322, Sep. 2004.

[3] ITU-T Recommendation P.800, "Methods of subjective determination of transmission quality," 1996.

[4] A. Kwon, J. Xiao, S.-S. Seo, J. W.-K. Hong, and R. Boutaba, "The impact of network performance on perceived video quality and user experience in H.264/AVC," in IEEE/IFIP Network Operations and Management Symposium (NOMS), pp. 1061-1067, April 2012.

[5] T. Wiegand, G.-J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560-576, July 2003.

[6] ITU-T Recommendation J. 1443m, "User requirements for objective perceptual video quality measurement in digital cable television," May 2000.

[7] A. M. Eskicioglu and P. S. Fisher, "Image quality measures and their performance," IEEE Transactions on Communications, vol. 43, no. 12, pp. 2959-2965, December 1995.

[8] A. Bhat, I. Richardson, and S. Kannangara, "A new perceptual quality metric for compressed video," IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 933-936, April 2009.

[9] X. Ran and N. Farvardin, "A perceptually motivated three-component image model

– Part 1: Description of the model," IEEE Trans. on Image Processing, vol. 4(4), pp. 401-415, 1995.

[10] O. Nemethova, M. Ries, M. Zavodsky, and M. Rupp, "PSNR-based estimation of subjective time-variant video quality for mobiles," MESAQUIN, 2006.

[11] V. Menkovski, A. Oredope, A. Liotta, and A. Cuadra, "Predicting quality of experience in multimedia streaming," In Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia, pp. 52-59, 2009.

[12] S. Latre, P. Simoens, B. De Vleeschauwer, W. Van De Meerssche, F. De Turck, B. Dhoedt, P. Demeester, S. Van Den Berghe, and E. Gilon de Lumley, "An autonomic architechture for optimizing QoE in multimedia access networks," Computer Networks, vol. 53, pp. 1587-1602, July 2009.

[13] S. B. Kotsiantis, "Supervised machine learning: a review of classification techniques," Informatica, vol. 31, pp. 249-268, 2007.

[14] A. R. Reibman and D. Poole, "Predicting packet-loss visibility using scene characteristics," in Packet Video 2007, pp. 308-317, November 2007.

[15] S. Markovitch and D. Rosenstein, "Feature generation using general construction functions," Machine Learning, vol. 49, pp. 59-98, 2002.

[16] S. S. Keerthi, and C.-J. Lin, "Asymptotic behavior of support vector machine with Gaussian kernel," Neural Computation, vol. 15, pp. 1667-1689, July 2003.

[17] F. Provost, "Learning with imbalanced data sets 101," In AAAI 2000 workshop on imbalanced data sets, 2000.

[18] J. Platt, "Fast training of support vector machines using sequential minimal optimization," Advances in Kernel Methods – Support Vector Learning, 1998.

[19] Weka, http://www.cs.waikato.ac.nz/~ml/weka/.

[20] Skewness, http://en.wikipedia.org/wiki/Skewness.

[21] Kurtosis, http://en.wikipedia.org/wiki/Kurtosis.

[22] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," Technical report, Department of Computer Science, National Taiwan University, 2003.

[23] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multi-class support vector machines," IEEE Transactions on Neural Networks, vol. 12, pp. 415-425, 2002.

[24] YUV CIF reference videos, http://www2.tkn.tu-berlin.de/research/evalvid/cif.html.

[25] J. Platt, "Fast training of support vector machines using sequential minimal optimization," Advances in Kernel Methods – Support Vector Learning, 1998.

[26] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," the fourteenth international joint conference on artificial intelligence, pp. 1137-1143, 1995.

[27] X264, http://en.wikipedia.org/wiki/X264.

[28] FFmpeg, http://ffmpeg.org/.

[29] M. Narwaria, W. Lin, and A. Liu, "Low-complexity video quality assessment using temporal quality variations," IEEE Transactions on Multimedia, vol. 14, pp. 525-535, 2012.

[30] S. Mohamed and G. Rubino, "A study of real-time packet video quality using random neural networks," IEEE Trans. Circuits Syst. Video Technol., vol. 12, no. 12, pp. 1071-1083, 2002.

[31] G. W. Cermak, "Videoconferencing service quality as a function of bandwidth, latency, and packet loss," Verizon Laboratories, T1A1.3/2003-026, 2003.

[32] B. Chen and J. Francis, “Multimedia performance evaluation,” AT&T Tech. Mem., Feb. 2003.

[33] S. Kanumuri, P. C. Cosman, A. R. Reibman and V. A. Vaishampayan, “Modeling packet-loss visibility in MPEG-2 Video,” IEEE Transactions on Mulitmedia, vol. 8, pp. 341-355, 2006.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor, Prof. James Won-Ki Hong, for the continuous support of my master study and research. Besides my supervisor, I would like to thank the rest of my thesis committee: Prof. Jong Kim and Prof. Young-Joo Suh, for their insightful comments and questions.

Besides, I would like to offer my sincerest gratitude to my thesis mentor, Dr. Xiao Jin, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I would also like to thank Arum Kwon and Oussama Stiti for their cooperation and helping me during the implementation of my thesis project.

I thank my fellows in DPNM Lab: Joon-Myung Kang, Byung-Chul Park, Sung-Su Kim, Sin-Seok Seo, Jae-Yoon Chung, Jian Li, Yeong-Rak Choi, Yoon-Seon Han, Hyeok-Soo Choi, Taehyun Kim, Do Le Quoc, Tae-Yeol Jeong, Jong-Hwan Hyun and Dong-Woo Kwon, for all the fun we have in the last two years.

My sincere thank also goes to professors and researchers of ITCE: Prof. Hector E. Roman, Prof. Jamal Deen, Prof. Donhee Ham, Prof. Meyya Meyyappan, Prof. Gianaurelio Cuniberti, Prof. Raouf Boutaba, Prof. Ahmed Mehaoua, and Dr. Hui Wang, for their creative thoughts, nice lectures, and impressive personalities.

I would also like to thank secretaries of our lab and ITCE: Hye-Jeong Jang, Yu-Jin Kim, Dong-Woo Ryoo, In-Tae Lee, Jong-Su Park, Hye-Ri Kim, Yun-Mi Jeong, and Min-Ji Kim, for their patient and careful help.

I would like to show my sincere gratitude to all my friends in POSTECH, including Chinese, Korean, Vietnamese, Iranian, Tunisian, Indian, Pakistan and Russian. Because of you, my life in POSTECH becomes more colorful.

Finally, I am especially grateful to my father and mother for their noisy or soundless support. I love you.

Curriculum Vitae

Name:	Yongfeng Huang (黄勇峰)
Address:	RIST Bldg. 4, Rm. 4405, POSTECH, San 31, Hyoja-dong, Nam-gu, Pohang, Korea (790-784)
Email:	xgjonathan@postech.ac.kr

EDUCATION

September 2006 – July 2010:

B.S. in Department of Computer Science and Technology

Xi'an Jiaotong University, Xi'an, China

September 2010 – July 2012:

M.S. in Division of IT Convergence Engineering (Autonomics)

Pohang University of Science and Technology, Pohang, Korea

PUBLICATION

1. Do Quoc Le, Hamid Faryabi, **Yongfeng Huang**, Duc Minh Le, M. Jamal Deen and Hong June Park, "Developing smart garment for monitoring vital signs," 3rd International Symposium on IT Convergence Engineering, Pohang, Korea, 2011. (**Best Poster Award top 10%**)