Event-based Estimation of User Experience for Network Video Streaming

Yongfeng Huang*, Jin Xiao*, James Won-Ki Hong*, Ahmed Mehaoua†, Raouf Boutaba‡

*Division of IT Convergence Engineering, POSTECH, Pohang, South Korea
email:{xgjonathan,jinxiao,jwkhong}@postech.ac.kr

†Laboratorie d'Informatique Paris Descartes, Paris Descartes University, Paris, France
email:ahmed.mehaoua@parisdescartes.fr

‡School of Computer Science, University of Waterloo, Canada
email:rboutaba@cs.uwaterloo.ca

***Abstract*—In managing multimedia services, it is important to understand how network performance affects user experience. The model presented in this paper aims to estimate user perception of video quality based on defect events, which are automatically classified by machine learning techniques. The underlying principle of our model is that human experience is event-based and there is a strong correlation between defective events and user MOS. Through experiments, we show that our model can detect different types of defect events with good accuracy even under small data set, and we find that indeed different defect event types affect user experience with different sensitivity.**

***Index Terms*—video quality management, H.264/AVC, QoE, machine learning**

I. 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 and how are related. 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 paper, we are interested in the effect of network-related factors specifically 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 most cases, including the peak signal to noise ratio (PSNR)/the mean squared error (MSE), structure similarity (SSIM) [1], and video quality metric (VQM) [2], metric is computed frame by frame, which means they need the original frame for reference. However, these approaches have the advantage of granting a clear and precise understanding of the correlations among metrics. One major disadvantage of these objective metrics is that human perception factors are not well represented.

Compared to these objective metrics, subjective perception of video quality is generally measured in terms of QoE, which are generally obtained via direct user feedback. A common way of such subjective measurements is the mean opinion score (MOS), which ranges 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 based on measurable metrics. Considering the cons and pros of both objective and subjective measurements, mapping from objective metrics to subjective ones is a good way of assessing user's QoE. More specifically, such a mapping should not only base on objective metrics, but also takes the particularities of human perception into consideration.

In this paper, we present the video perception model of VIDeo quality Analyzer in Real-time (VIDAR) project [4], 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 machine learning techniques. In summary, our model focuses on the following two problems:

- Because conventional objective metrics do not consider human perception factors, how to improve the accuracy of estimated user MOS based on these metrics?
- Because our model is applied to real-time streaming multimedia, how to ensure that it is simple and fast?

Accordingly, our contributions are as follows: first, to improve the accuracy of estimated user MOS. Based on the event-based nature of human experience, we group frame quality metrics into defective events, which are series of defective frames in close played time proximity. Then, multi-class classification is applied to automatically determine the types of defects a human viewer will experience. Because of the complex 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 time series data that best represent the characteristics of different defective events. Second, to ensure that our model is simple and fast, we only focus on a few key features. At the same time, optimization processes are applied in our machine learning techniques, including parameters selection for radial basis function (RBF) kernel of support vector machine (SVM), and methods selection for aggregating binary classifiers. Through experiments, 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 fast.

The rest of the paper is organized as follows. Section II 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. Section III presents the overview of VIDAR, specifically our user model. In section IV, we explain the classification algorithms used on defective events. We show our simulation results in section V and conclusion in section VI.

II. Related Work and Background

A. 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.

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].

B. 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]. FR and RR give better result in terms of accuracy, but are impractical for real-time scenarios, while NR gives the worst accuracy, but is suitable for real-time implementation.

Among the different FR models to assess video quality, PSNR is simple and popular. However, it has poor performance against human perception according to investigation [7]. VQM shows good correlation with user's perception of video quality [2]. However, due to its high requirement of processing capacity, it is not suitable for real-time scenarios. 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].

C. Inferring user MOS from objective metrics

Objective metrics that are related to user perception of video quality can be classified into application-level metrics (MSE, PSNR, etc.) and network-level metrics (packet loss, delay, etc.). Here, we start with related work on predicting user MOS from application-level metrics.

Since objective metrics, including MSE, PSNR and SSIM, consider neither natural visual characteristics nor perceptual characteristics of human visual system (HVS), a lot of work has been done to remedy these objective metrics by mapping with subjective factors.

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

$$MOS_p = 1 - k(MSE)$$

Where k is derived from the spatial edge strength, since 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].

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]:

$$\widehat{MOS}_{PSNR}[n] = a \times PSNR[n] + b$$

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.
Besides related work above where basic statistical techniques are used, machine learning (ML), as a more advanced statistical method, is applied to build accurate and adaptive QoE prediction model [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 bit rate, while the output is just simple "YES" or "NO". Obviously, simply accepted or not cannot express user's QoE comprehensively.
Besides approaches mentioned above that are based on application-level metrics, some work also try to predict user's QoE from network-level metrics using ML algorithms. Neural network-based reasoner is designed to optimize the QoE, in terms of PSNR, with a random packet loss on the link [12].
Compared to 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, our approach can give better prediction of user's QoE, with human perceptual characteristics considered.

D. Machine Learning Techniques

Since we have defined defective event classes, and generated a set of event data with class labels, we will only give a brief summary of supervised machine learning techniques in this part, but not unsupervised ones.
In general, supervised ML 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 go through all the steps repeatedly and adjust anyone that can improve the accuracy.
Briefly, 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), 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. Nave Bayesian network (NB) 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 also considered to be partial, because it assumes that it can discriminate between classes by a single probability distribution. Another popular algorithm under the category of statistical method is k-nearest neighbor (kNN), which is based on the principle that instances close to each other have similar properties. Although kNN is very sensitive to irrelevant features, it is simple to use with only one single parameter (the number of nearest neighbors) to set. Finally, as the newest supervised ML technique, SVM has been shown to perform much better when dealing with multi-dimension and continuous features, as well as situations with a nonlinear relationship between 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 simplest approach is to test several candidate algorithms on the dataset, then choose one with the highest accuracy. Therefore, for our problem with numerical features, we choose kNN (the simplest) and SVM (usually has the best performance) as our candidates.

III. Overview of Our Approach

A. Overview of VIDAR

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. VIDAR is made up of three models: the R3 model, the Vidi model, and the user model (Figure 1).
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. By analyzing these trees, it is possible to estimate SSIM (eSSIM) without access to transmitted videos [4].
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 generates 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 paper, we show how Vidi metrics can be used to analyze viewer's perception and the impact of different defects on user's MOS.

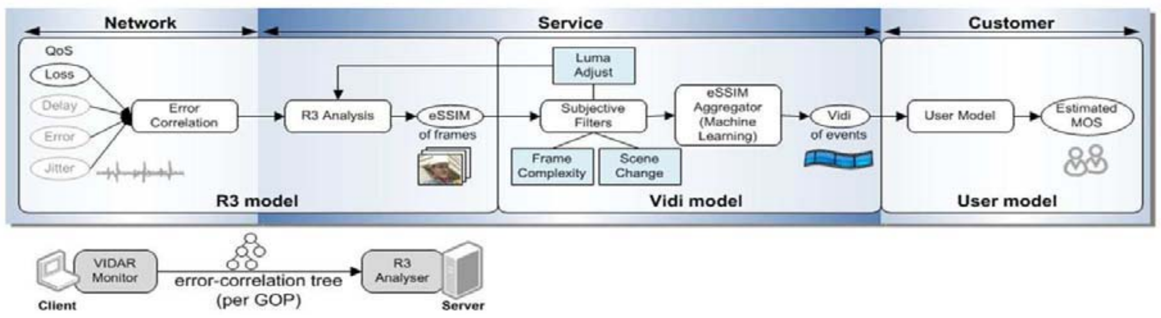

Fig. 1. Process flow of VIDAR framework

B. Overview of event-based classification

According to our experiential investigation, defective events of transmitted video can be categorized as follows:

- *Distortion*: frames containing perceivable distortions.
- *Glitch*: short sequence of distortion.
- *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.

In our previous work [4], these defective events are generated from a modified aggregated eSSIM by passing eSSIM through the content feature filters (Figure 1). 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 defective events automatically by using ML techniques.

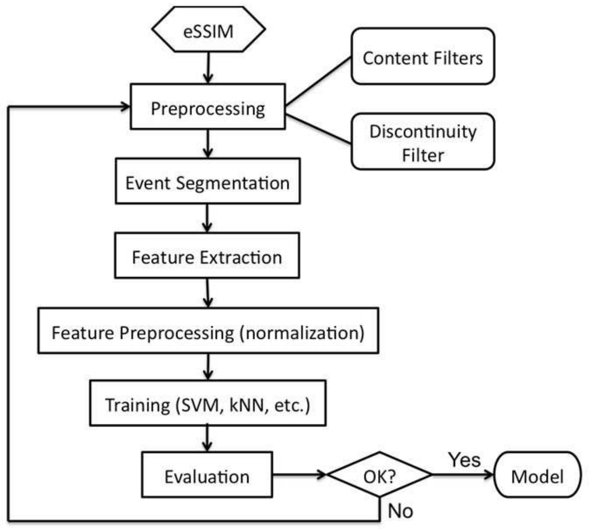

Fig. 2. Process flow of eSSIM aggregator (machine learning)

The basic steps of our classification approach are shown in Figure 2. Since we have multiple types of defective events, multi class techniques are used in our research. Among our candidate binary classification algorithms, both ANN and kNN can be directly extended to do multi-class classification, while a combination of several binary SVM classifiers has to be used to solve a given multi-class problem.

According to 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 in the experiment.

IV. Classification of Defective Events

In this section, we present multi-class classification for the defective events (Figure 2). To improve the accuracy while keeping the classification processing efficient, several optimization steps are applied (i.e. parameter selection of SVM kernel).

A. 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:

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

The 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.

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.

B. 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 paper, 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 bits per second.), we set the following rules to extract event instances from video eSSIM sequences: an event has minimum length of 10 and maximum of 100. The boundary frames between two events must be 10 or more.

C. Feature Extraction

We 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, and it also requires a 1 to 1 alignment. But for our case, events do not have a uniformed start point and do not follow a specific pattern. There are many reasons for these characteristics of our data, including packet-loss model used in our experiment, 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.

The features we extract from event data are: 1) mean (μ); 2) standard deviation (σ); 3) minimum;
4) defective ratio:

$$ratio = \frac{N_{eSSIM<0.95}}{n},$$

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.
5) severity of dropped frames and duplicated frames:

$$severity = \frac{N_{eSSIM\leq 0.0}}{n},$$

where $N_{eSSIM\leq 0.0}$ is the number of frames with $eSSIM \leq 0.0$. 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. 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.
By applying these statistical measurements to defective event dataset with high dimension and different length, a new dataset with a limited number of features are generated.

D. 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 kNN to calculate distance between two instances, skewness and kurtosis can affect the result significantly. Therefore, before next step, our dataset is normalized to range [-1, 1].

E. 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 is accompanied by a large mean value. The dependence among several features often unduly influences the accuracy of classification [15]. Compared to SVM, kNN is very sensitive to noisy data or irrelevant features [13]. Therefore, a greedy forward selection of features is applied to kNN. 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. Afterwards, the input subsets with three, four, and more features are evaluated. Finally, the best one is selected from all the combinations.

F. Multi-class classification (SVM)

SVM is widely used and usually performs well in many fields. Its advantages 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. The performance of SVMs is dependent on a number of important factors: preprocessing data, kernel selection, parameters setting of the SVM and the kernel. We discussed preprocess data prior, we will focus on kernel selection and parameters setting.

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. 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 (Section V) with optimized parameters, linear kernel gives the worst result (accuracy: 74.45%±0.63%), while RBF kernel performs the best (accuracy: $85.46\% \pm 0.29\%$) and polynomial kernel is in the second place (accuracy: $83.88\% \pm 0.36\%$). The reason for linear kernel's poorest performance is that our instances are not linearly separable even in the higher dimensional feature space. Therefore, we choose RBF kernel for our classification. RBF kernel is shown as the following:

$$k(\overrightarrow{x}, \overrightarrow{x_l}) = exp(-\gamma \| \overrightarrow{x} - \overrightarrow{x_l} \|^2),$$

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

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 is the soft margin constant or the penalty factor. It controls how strict every instance can be classified correctly. 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, and overfitting occurs when γ is too large. To choose the best combination of C and γ, a common approach is using a grid search, which has time complexity of $O(n^2)$. Since we require our model to support real-time streaming, grid search is not feasible. However, we observe that the best combination of C and γ are in fact located in a local area. The result of a SVM with linear kernel can therefore serve as a baseline [16]:

$$log\sigma^2 = logC - log\widetilde{C},$$

where $\widetilde{C}$ is the penalty parameter of the SVM with a linear kernel. The baseline with best combinations of C and γ is therefore located as defined by the above equation, when $\widetilde{C}$ is optimized for the linear SVM. Accordingly, we obtain our combination of C and γ based on the following procedure: 1) search for the optimal $\widetilde{C}$ of the SVM with linear kernel; 2) search for the best C and γ combination that satisfy the equation based on the optimal $\widetilde{C}$. The time complexity is thus trimmed down to $O(n)$.

Combination of binary classifiers: to meet the requirement of classifying multiple classes, it is common to combine several binary classifiers. There are two popular strategies of combination: one versus all (OVA) and one-versus-one (OVO). OVA combines M (the number of classes) binary classifiers. For each classifier, it choose a class as the positive class, and the remaining as the negative class. OVO combines $M(M-1)/2$ binary classifiers. For each classifier, it select two from all the classes mutually exclusively: 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. Given the relatively small class size and presence of biased sample sizes across the classes [17], we select OVO as the combination method for our multi-class classification.

G. Multi-class classification (kNN)

kNN 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, kNN can be very sensitive to noisy instances or imbalanced features. Two decisions need to be made when use kNN: choosing k and the distance function. A smaller k is more sensitive to noisy instances, but can fit classes with small areas. The distance function minimizes the distance between instances of the same class, and maximizes the distance between instances of different classes. We use a grid search to find the best k and the best distance function ∈ Minkowsky, Manhattan, Chebychev, Euclidean) [13].

V. Experiment

In this section, we show our experiment results. First, we show our data set for classification and that how it 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 classification method is related to user MOS.

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

TABLE I
Original Videos

Name	Scene Cuts	Motion	Content
Foreman	0	Medium	Portrait and landscape
Mother and Daughter	0	None	Portrait
Football	0	Fast	Sports
Bus	0	Medium	Moving vehicle
inception	6	Fast	Varied

FFmpeg, and plays the video with VLC media player. Five original videos are used for the experiment (Table I). During encoding, we set two different sizes (12 and 24) of GOP for each video, since the varying GOP affects the severity of defects in videos [4]. Besides, to simulate different network packet loss situations, we apply two packet loss models for the experiment: Bernoulli uniform loss model and Glibert-Elliott (GE) burst loss model. 2% packet loss rate is set to both of them. Through the experiment, we find that generally GE model creates severer defects than the uniform model, and GE model is more likely to introduce freezing into the transmitted video, while glitch defects are more probably caused by the uniform model.

There are 157 defective events generated by the Vidi model with preprocessing, including 87 distortion events, 27 glitch events, and 43 freezing events. These defective events are labeled manually. We compare four multi-class classification methods in terms of both accuracy and time:

- *SMO-G*: sequential minimal optimization (SMO) with RBF kernel, and the best combination of C and γ is set by Grid search (SMO-G). the range of C and γ for searching is the same as that in last section. SMO is an algorithm for efficiently solving the optimization problem which arises during the training of SVMs [18].
- *SMO-L*: SMO with Linear kernel (SMO-L). Since for linear kernel, only C needs to be set. We use a fine granularity, C $\in$ 2, 4, ..., 198, 200, for search the best C.
- *SMO-O*: SMO with RBF kernel, and the best combination of C and γ is set by the optimized method introduced in Section IV-F. First, we fix $\widetilde{C}$ with the best C selected by SMO-L. Then, compared to SMO-G, we use a fine granularity, $\gamma \in$ 0.05, 0.10, ..., 1.95, 2.00, for searching the best combination of C and γ.
- *kNN*: we use a grid search for the best combination of k and distance function.

For all these four methods, we conduct a ten-fold cross-validation on the data set. The implementation is based on Weka open source library [19]. The classification result is shown in Table II. According to this table, SMO-G performs the best in terms of accuracy, but its running time is much higher than the others. On the other hand, SMO-O gives almost the same accuracy as SMO-G, while it requires a reasonable running time. It is worth noting that with the best performance in terms of time, kNN also shows a relative high accuracy. But as a lazy-learning algorithm, the executing time of kNN is much higher than its training time, especially when the data set is huge.

For the above experiment, we set parameters for each binary classifier uniformly to make it simple and fast. Now we investigate them separately. According to Table III, the highest accuracy of "Glitch vs. Freezing" shows that glitch and freezing can be distinguished from each other easily. The reason for the lower accuracy of "Distortion vs. Glitch", we believe, is that classification between them is content-related. For instance, considering the same amount of error while decoding macro blocks, error on the main object (i.e., mother's face in Mother and Daughter) is more perceivable than that on the background. On the other hand, the reason for the lowest accuracy of "Distortion vs. Freezing" is that distortion and freezing happen together in the same event sometimes.

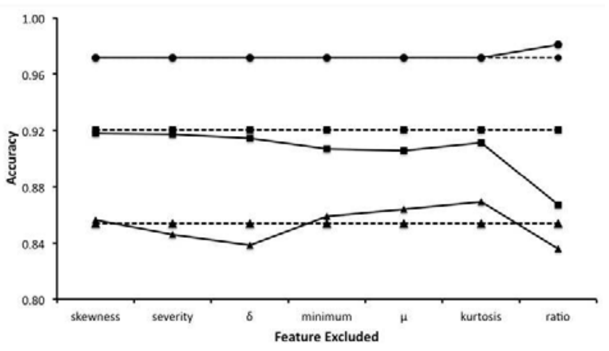

Fig. 3. Feature Sensitivity Under Different Binary Classifiers

In Figure 3, we study how each binary classifier is sensitive to seven features. The dashed line shows the classification result without any feature removed, while the solid line is the classification result after each feature is removed. As we can see, the remove of any feature cannot improve the accuracy of all binary classifiers at the same time, which means all seven features are necessary for multi-class classification. However, we do see that some classifier is sensitive to feature remove. For instance, "Distortion vs. Glitch" is sensitive to the feature *ratio*. In fact, it is reasonable since glitch usually has a quite small *ratio* compared to distortion. Therefore, *ratio* must be a key metric to distinguish glitch from distortion.

Now we examine how well the defective events predict user MOS. We conducted user tests with 5 people. 30 defective events were selected randomly from the data set as the testing set, including 10 glitch, 10 distortion and 10 freezing, respectively. Then, we prepared play lists for each person by arranging these defective events in a random order. Every user was asked to provide a real valued MOS for each event. The average MOS for each type of defect was calculated (Figure 4). According to our definition of defective events, not surprisingly, distortion causes the worst perception of video quality (2.11 ± 0.67), while glitch is not too perceivable (3.48 ± 0.34). It is worth mentioning that although discontinuity after freezing is the main reason for the low average

TABLE II
CLASSIFICATION RESULT

	SMO-G	SMO-L	SMO-O	kNN
Accuracy	$85.46\% \pm 0.29\%$	$74.45\% \pm 0.63\%$	$84.57\% \pm 0.29\%$	$83.00\% \pm 0.33\%$
Time (s)	445.2 ± 16.0	66.7 ± 2.7	94.5 ± 4.5	31.6 ± 3.2

TABLE III
RESULTS BY BINARY CLASSIFIERS

	Dist. vs. Glitch	Dist. vs. Freeze	Glitch vs. Freeze
Accuracy	92.04%	85.38%	97.18%

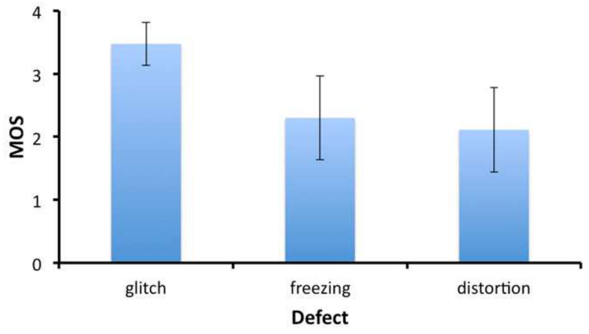

Fig. 4. MOS of Each Type of Defect

MOS (2.3 ± 0.66), we find that freezing is accompanied by distortion for some cases, and it can also cause a lower MOS. In summary, the approximate 85% accuracy of SMO-O shows that the features extracted from eSSIM series are able to capture both similar and different characteristics of defective events even under a rather small training set. This is encouraging as the requirement on large training data is one of the major drawbacks of ML-based approaches. Lastly, we find that different defect events impact user experience at different sensitivity. Therefore, it makes sense to deal with the mapping of defects to MOS from a per event class perspective.

VI. Conclusion

In this paper, we have presented an automatic event-based video quality analyzer using ML techniques. Our model considers the event-based nature of human experience, which is a different perspective compared to other known ML approaches. To make our design practical to implement, we have stressed on performance efficiency, including feature selection, parameter optimization, and method selection. In our investigation, we find that it is paramount to select key features that bear strong correlation with the defective event being classified, and the applying subjective filters is a method for data pre-process. For future work, we will conduct more experiments and focus more on predicting user MOS.

Acknowledgements

This research was supported by World Class University program funded by the Ministry of Education, Science and Technology through the National Research Foundation of Korea (R31-10100) and Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (2011-0020518).
The authors would also like to thank Arum Kwon for her guidance and Oussama Stiti for his technical support.

References

[1] Z. Wang, L. Lu, and A. C. Bovik, "Video quality assessment based on structural distortion measurement," *Signal Processing: Image Communication*, vol. 19, no. 2, pp. 121–132, February 2004.

[2] M. H. Pinson and S. Wolf, "A new standardized method for objectively measuring video quality," *IEEE Transactions on Broadcasting*, vol. 50, no. 3, pp. 312–322, September 2004.

[3] ITU-T, "Methods of subjective determination of transmission quality," ITU-T Recommendation P.800, 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), mini-conference*, 2012.

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

[6] ITU-T, "User requirements for objective perceptual video quality measurement in digital cable television," ITU-T Recommendation J.1443m, 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," in *IEEE International Conference on Acoustics, Speech and Singal Processing*, April 2009.

[9] X. Ran and N. Farvardin, "A perceptually motivated three-component image model - part 1: Description of the model," *IEEE Transactions on Image Processing*, vol. 4, no. 4, pp. 401–415, April 1995.

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

[11] V. Menkovski, A. Oredope, A. Liotta, and A. Cuadra, "Predicting quality of experience in multimedia streaming," in *International Conference on Advances in Mobile Computing and Multimedia*, 2009.

[12] S. Latre, P. Simoens, B. D. Vleeschauwer, W. V. D. Meerssche, F. D. Turck, B. Dhoedt, P. Demeester, S. V. D. Berghe, and E. D. de Lumley, "An autonomic architecture 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*, 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*, 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," in *Advances in Kernel Methods - Support Vector Learning*, 1998.

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