INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 13, Number 6, Pages 879–897

MATHEMATICAL MODELS FOR QUALITY ANALYSIS OF MOBILE VIDEO

SONGQING ZHAO, HONG JIANG, CHAO LIANG, SHERIF SHERIF, AND AHMED TARRAF

Abstract. With the explosive growth of mobile video applications, analysis of video quality becomes increasingly important because it is an important Key Performance Indicator (KPI) for Quality of Experience (QoE). In this paper, a framework for non-reference video quality analysis is proposed and applied to Video Telephony (VT) in LTE networks. Three metrics, blockiness, blur and freezing, are used to estimate the MOS. Blockiness is detected by taking the H.264 codec features into account, blur is estimated by utilizing the percentage of noticeable blurred edges in each frame, and freezing is evaluated by using a sigmoid function to mimic the effect of different freezing duration on the Human Visual System (HVS). Furthermore, the three metrics are combined into one objective MOS by considering different weighting factors and using the linear curve fitting. Above 90% correlation is achieved between the objective MOS score and subjective MOS score.

Key words. Mathematical models, video quality analysis, quality of experience, mean opinion score, blockiness, blur, freezing, human visual system, modeling.

1. Introduction

As smartphones and tablets are increasingly being used by more and more people and cellular network capacity is being significantly improved with more 4G LTE network deployment, mobile data traffic is growing explosively. Among the data traffic, video traffic is playing a big role. In a recent study [5], mobile video traffic was already 51 percent of the entire mobile data traffic by the end of 2012. It forecasts that mobile video will increase 16 times between 2012 and 2017 and two-thirds of the world's mobile data traffic will be video by 2017. As a consequence the analysis of video quality is becoming increasingly important because it is an important Key Performance Indicator (KPI) for Quality of Experience (QoE). In recent years, QoE research and development has gained significant attraction in both academia and industry since the first international workshop on Quality of Multimedia Experience (QoMEX) was held in 2009 and more and more video quality models are recommended by Video Quality Expert Group (VQEG) to the ITU-T standardization process [25].

Fig. 1 is a typical architecture for Video Telephony (VT) over all-IP based LTE network. The User Equipment (UE) of the originator sends out the original video through the uplink indicated by blue lines, and the UE terminator receives the video through the downlink indicated by red lines. The real-time video for VT application is usually encoded in H.264 and transmitted through the Real-time Transport Protocol (RTP). During transmission, especially over the air interface between UE and base station, IP packets may experience network impairments such as packet loss, delay, and jitter, which are crucial factors in causing degradation to the quality of the received video. Video packets may be lost due to network congestion or they may be discarded by the video decoder if they arrive at the

Received by the editors July 22, 2016.
2000 *Mathematics Subject Classification.* 35R35, 49J40, 60G40.

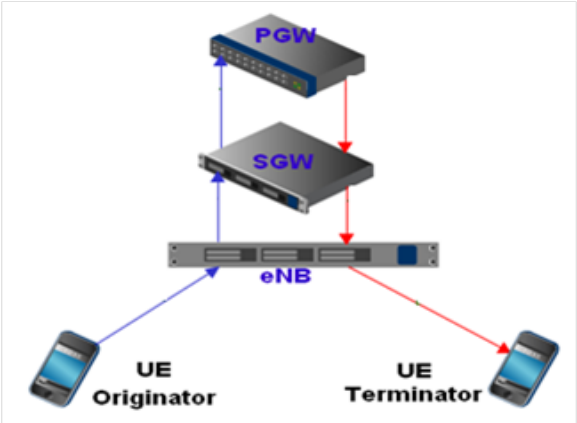


FIGURE 1. Components in LTE network.

terminator UE with a delay so large that it exceeds the video de-jitter buffer's limit. When the terminator UE does not receive the video packets for certain time duration, it will use the latest decoded frame for display, resulting in freezing or jerky video. When a few video packets in one Group of Picture (GOP) are lost, the video decoder will produce some impaired frames even after error concealment, which cause annoying blockiness and blur. Once there is an impaired frame, the subsequent frames until the next I frame will be also adversely affected due to the error propagation property in H.264 codec. It is worthwhile to mention that although conceptually the blockiness occurs on the regular 8x8 block boundaries, and has clear block edges, this notable feature is diminished due to the de-blocking filter and error concealment at the video decoder, making blockiness detection more challenging.

There exist subjective video quality and objective video quality. Subjective video quality can be evaluated according to the standard P.910 [11]. Essentially, a group of people are asked to watch the video in a specific environment with certain lighting requirements and to give their scores to the video, according to their liking, using a certain scale, e.g. 1-5. After averaging the scores over the audience, each video is provided with a Mean Opinion Score (MOS). However, this process to obtain the subjective MOS is both time and resource consuming, especially when a lot of video sequences need to be evaluated. The purpose of objective video quality metrics [23] is to use artificial intelligence to replace people evaluating the video. Such a predicted MOS approximates the subjective MOS by detecting and estimating the impairments most sensitive to the Human Visual System (HVS), e.g. freezing/jerkiness, blockiness, and blur, and combining these impairments into one MOS.

In general, there are three categories for objective video quality analysis. The first category is Full Reference (FR) video quality, in which the analysis is performed by comparing the received and decoded video with the original reference video. The most used FR metric is Peak Signal-to-Noise Ratio (PSNR) because of its computational simplicity and simple mathematical formula. However, it correlates poorly with the subjective ratings [3] [22]. Thus many new FR metrics are developed that better correlate with subjective ratings than PSNR does, such as structure similarity (SSIM) [22] and visual signal-to-noise ratio (VSNR) [3], and the latest FR video quality standard for multimedia application is ITU-T J.247 [10]. Another

category is Reduced Reference (RR) video quality, which analyzes the received and decoded video using auxiliary information about a limited number of features extracted from the original reference video. The auxiliary information may be embedded into the video packet header or transmitted through the ancillary data channel [7] [13]. The latest RR video quality standard is ITU-T J.246 [9]. The third category is No Reference (NR) video quality analysis in which only the received video is analyzed because no information about the original video is available. As described in [8] by Hemami et al. NR video quality analysis is more attractive and more practical since the original reference video source or video information is not available in many scenarios. At the same time, NR video quality analysis is most challenging due to the lack of reference video. In this paper, we will focus on the NR video quality analysis.

1.1. Related Work. The related work about NR video quality can be divided into another three categories. The first category is based on the analysis of the received bit-stream without decoding it into video frames. Various features of the video stream, such as quantization parameter, motion vector information, macroblock information, frame rate, frame type, frame size, packet loss, are extracted directly from H.264 coded bit-stream at the receiver, and then applied in a MOS prediction model such as neural networks [1] [4] [12] [19] [27]. Another category is based on the analysis of decoded video. In [17], Shen et al. apply natural video statistics, 2D/3D curvelet and cosine transforms of decoded video, and investigate the relation between video quality, specific video characteristics, and motion vectors. However, this method has a high computational complexity. The final category is based on analysis of the combined bit-stream and decoded video, which is called a hybrid model. The hybrid NR video quality analysis is a very promising method and has the highest accuracy among all the NR video quality methods since it has more information available for predicting MOS as illustrated in [18] [20] [28]. Obviously, the complexity of hybrid model is higher than a model just based on bit-stream itself or decoded video itself. The decoded video NR video quality has higher accuracy than bit-stream based NR video quality since a bit-stream based method is still performed before decoding and ignores the impact of video decoder such as error concealment.

1.2. Main Contribution and Motivation. By considering a compromise between accuracy and complexity, we focus, in this paper, on NR video quality based on received decoded video analysis. The novelty of this paper is that it introduces a new method for computing an objective MOS by combining NR metrics for freezing/jerkiness, blockiness and blur. Furthermore, we use a new content-agnostic method for detection and estimation of blockiness, and new edge ratio for the detection of blur. For freezing/jerkiness detection, we use a novel detection algorithm by using a histogram of inter-frame pixel differences and also taking into consideration of shaking disturbance to prevent false detection.

The purpose of our VT video quality analysis over LTE networks is to ensure that the end user has a good QoE and is satisfied with the VT service. The VT videos in this paper are obtained in the various network situations. Our analysis will provide the quantitative results for these videos. Thus we will know how the network conditions affect the VT quantitatively, which is also very useful information for VT service providers and helps them improve VT service over LTE. Three most critical impairments, i.e. blockiness, blur and freezing/jerkiness will be analyzed using our novel adaptation for the state-of-the-art image processing techniques. The

metrics for these impairments are used for predicting the objective MOS. Finally, a polynomial curve fitting model is adopted based on the video database from practical VT testing over LTE network.

The paper is organized as follows. First, the algorithm for blockiness estimation and the corresponding implementation are described and examples are given. Second, the algorithm for blur estimation and the corresponding implementation is described and examples are given. Third, freezing/jerkiness detection and estimation method and model are described. Fourth, the MOS prediction model is described and its performance is given. Finally we summarize the paper with future work in the conclusion section.

2. BLOCKINESS DETECTION AND ESTIMATION

2.1. Algorithm Description. First of all, the blockiness algorithm and the blur algorithm in the following section will deal with grayscale image. Since our decoded video image is RGB format, we need convert it into gray image by computing the luminance value from RGB values as follows.

$$Y = 0.2989 * R + 0.587 * G + 0.114 * B. \tag{1}$$

In [24], Wang et al. proposed a NR blockiness estimation method by averaging the differences across block boundaries, which are located at multiple of 8 columns and rows due to the 8x8 DCT-based coding blocks. In [15], Muijs et al. introduced the normalized gradient metric to estimate the blockiness. However, these methods are content dependent, i.e., the blockiness metric value could be very different for the different image content with similar blockiness level, or the blockiness metric value could be similar for one image (simple texture and some blockiness) and another image (complex texture and no blockiness). Thus we propose a novel content independent blockiness estimation method by explicitly detecting the blockiness perceived by HVS and calculating the severity of blockiness. H.264 is a block transform based codec. Therefore, we use the method in [15] to verify that blockiness indeed occurs at multiple of 8 columns (vertical blockiness) and rows (horizontal blockiness). In the following, we only describe the details for vertical blockiness detection, and the detection of horizontal blockiness can be performed in a similar fashion.

Let I denote a grayscale image with M rows and N columns, and $I(i, j)$ denote the pixel value at ith row and jth column, where $1 \leq i \leq M$ and $1 \leq j \leq N$. The horizontal difference between two neighbor columns is computed as

$$D_H(i, j) = I(i, j + 1) - I(i, j). \tag{2}$$

In the areas with a lot of texture, D_H will be big, which value is comparable with the one when blockiness occurs. In order to solve this issue, we define the following horizontal neighbor average difference and normalized horizontal difference.

$$D_{H,ave,left}(i, j) = \frac{1}{p_2 - p_1 + 1} \sum_{k=j-p_2}^{j-p_1} D_H(i, k), \tag{3}$$

$$D_{H,ave,right}(i, j) = \frac{1}{p_2 - p_1 + 1} \sum_{k=j+p_1}^{j+p_2} D_H(i, k), \tag{4}$$

$$D_{H,norm}(i,j) = \frac{D_H(i,j)}{\min\left(D_{H,ave,left}(i,j),\, D_{H,ave,right}(i,j)\right) + \epsilon}, \tag{5}$$

where j is only considered for multiple of 8 as in [24], p_1 and p_2 are parameters to take into consideration of deblocking filter. By checking the blockiness area, we notice that when blockiness occurs, the value of D_H at block boundary is large, and its value close to block boundary is also not small when deblocking filter is used, but the value of D_H away from block boundary is small. Taking this fact into account, we set p_1 and p_2 to 2 and 6 (but not 1 and 7), respectively. If $D_{H,ave,left}$ and $D_{H,ave,right}$ are smaller than $D_{ave,thresh}$ (we set it to 3 after checking the blockiness areas), they will be set to 0 in order to magnify $D_{H,norm}$ since they are the candidates for blockiness in this situation. ϵ stands for a very small value (e.g. 1e-6) in order to avoid to be divided by 0. The initial binary image $I_{V,bin}$ for vertical blockiness edge is obtained as follows

$$I_{V,bin}(i,j) = \begin{cases} 1 & D_H(i,j) > D_{thresh}\, and\, D_{H,norm}(i,j) > D_{norm,thresh} \\ 0 & else \end{cases}, \tag{6}$$

where again j is only considered for multiple of 8, D_{thresh} is set to 5 and $D_{norm,thresh}$ is set to 1000 in our case. For other columns, $I_{V,bin}(i,j)$ equals to 0 since they are not the places of vertical blockiness edge.

Existing work in literature stops at Eq. (2) or Eq.(5). However, we will utilize the definition in Eq. (6) and perform further processing to arrive at a new metric in order to be content independent.

In the initial $I_{V,bin}$, we get the initial vertical blockiness edge segments. Due to the use of deblocking filter, we observe that some blockiness segments that belong to the same blockiness edge have a gap between them. Thus the blockiness edge becomes discontinuous in most cases. In order to solve this issue, if the gap of two neighbor segments is less than some threshold (we use 4 pixels in our case), we will connect them. It is like the dilation operation in binary morphology. After this connection processing, we will consider the segments which has length less than 8 pixels as noise since the blockiness length must be greater than 8.

After the above processing, by testing the different content images, we also notice that some vertical blockiness edge segments come from the image content itself. One way to reduce them is to consider the fact that there should be at least one corresponding horizontal blockiness edge around one vertical blockiness edge since they are blocks. After this processing, we will get the final $I_{V,bin}$. Similarly final $I_{H,bin}$ can be obtained by the same processing. The blockiness metric is calculated as

$$F_{Blockiness} = \frac{1}{2}(TotalLengthV + TotalLengthH), \tag{7}$$

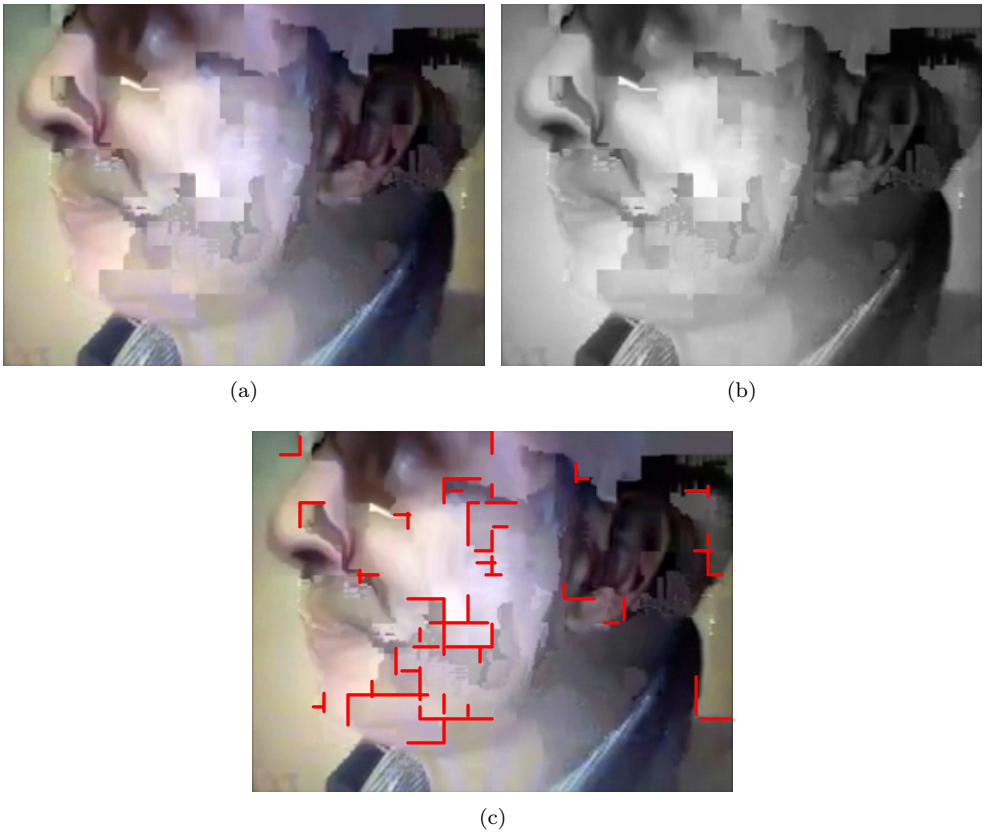
where $TotalLengthV$ is the total length of vertical blockiness segments in the final $I_{V,bin}$, and $TotalLengthH$ is the total length of horizontal blockiness segments in the final $I_{H,bin}$.

Above described blockiness detection and estimation algorithm is summarized as follows.

Step 1: initialize all parameters (p_1, p_2, $D_{ave,thresh}$, D_{thresh}, $D_{norm,thresh}$, Segment distance, Minimum segment length),

Step 2: (vertical blockiness) calculate the absolute difference D_H for each pair of neighbor columns in grayscale image,

Step 3: compute the average of D_H around the columns of multiples of 8 ($D_{H,ave,left}$ and $D_{H,ave,right}$) and process them according to $D_{ave,thresh}$,



(a) (b)

(c)

FIGURE 2. Blockiness algorithm verification (a) Original image with blockiness impairment, (b) Corresponding grayscale image, (c) Original image overlapped with blockiness detection results.

Step 4: calculate the normalized difference $D_{H,norm}$,

Step 5: binarize $D_{H,norm}$ according to D_{thresh}, $D_{norm,thresh}$ to compute the initial $I_{V,bin}$,

Step 6: connect segments and remove segments in the binary image $I_{V,bin}$ according to segment distance and minimum segment length,

Step 7: do the similar steps (step 2 to step 6) for horizontal blockiness and obtain the binary image $I_{H,bin}$,

Step 8: further process the binary image $I_{V,bin}$ to obtain the final $I_{V,bin}$, i.e. for each segment in $I_{V,bin}$, if any segment is found in $I_{H,bin}$ which is within 4 pixels around it, then keep it, otherwise, consider it as noise and discard it,

Step 9: obtain the final $I_{H,bin}$ by similar processing with step 8,

Step 10: from final $I_{V,bin}$ and $I_{H,bin}$, calculate the total length of vertical blockiness edges and horizontal blockiness edges respectively and get the blockiness metric of this image.

2.2. Implementation and Discussion. We take one image with blockiness from a saved video at the receiver device for the video telephony application over LTE networks to verify the effectiveness of our algorithm. Fig. 2 (a) is the color image which is perceived by the end user. It has blockiness impairment due to the video

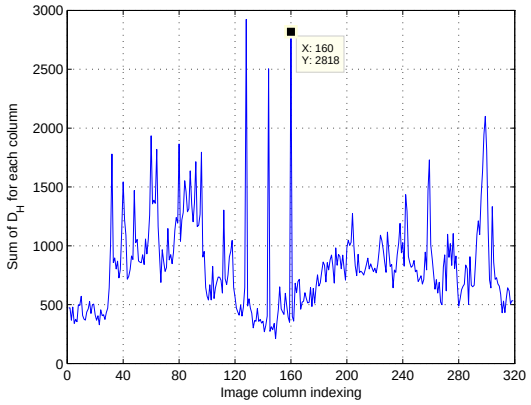


FIGURE 3. Summation of D_H for each column.

packet loss over LTE networks. Fig. 2 (b) is the corresponding grayscale image which will be processed by our blockiness algorithm. This is the second section.

After calculation of DH for Fig. 2 (b), Fig. 3 is the plot for the summation of DH for each column, where we can clearly see that the blockiness peaks happen at the columns of multiple of 8. We label one peak which corresponds to the column of 160 and is in the middle of the image. Going back to the middle of image Fig. 2 (a), we indeed see the severe blockiness. And the rows have the similar phenomenon. Thus this confirms that the blockiness indeed occurs at the multiple of 8 columns and rows if there is any.

Fig. 4 (a) is the initial vertical binary image $I_{V,bin}$ for Step 5, where we see that a lot of segments are very small, which are not blockiness segments, and some blockiness segments are broken into several parts. By connecting the close segments and discarding the small ones after connection (i.e. Step 6), we get the processed $I_{V,bin}$ as Fig. 4 (b). With the similar process for horizontal part, we get the initial $I_{H,bin}$ as Fig. 4 (c) and processed $I_{H,bin}$ as Fig. 4 (d). Putting Fig. 4 (b) and Fig. 4 (d) together, we get Fig. 4 (e). Since one vertical blockiness segment should have one corresponding horizontal blockiness segment around it, we get the final blockiness detection results as Fig. 4 (f) after discarding the single vertical/horizontal segments (Step 8 and Step 9).

In order to easily see the effectiveness of our blockiness detection algorithm, we superimpose the final results in Fig. 4 (f) on top of original image and create Fig. 2 (c), where we can see that most of blockiness is successfully detected. Some of blockiness is not detected because the error concealment changes the features of some blockiness. For example, in Fig. 2 (c), the blockiness in the top-right part is not detected because the vertical edges at those places are not at multiple of 8 due to error concealment process and hence discarded by our algorithm. However, as we can see from Fig. 2 (c), the missing blockiness detection rate is very small. More importantly, our blockiness detection algorithm works well for different image content and different level of blockiness. Finally, the blockiness metric will be calculated according to Eq. (7) and be used for the MOS score prediction.

We use the 233 JPEG images from LIVE image quality database in [29] to verify our blockiness algorithm's effectiveness and compare it with the existing blockiness metric in [24]. The 233 JPEG images are DCT block-based compressed

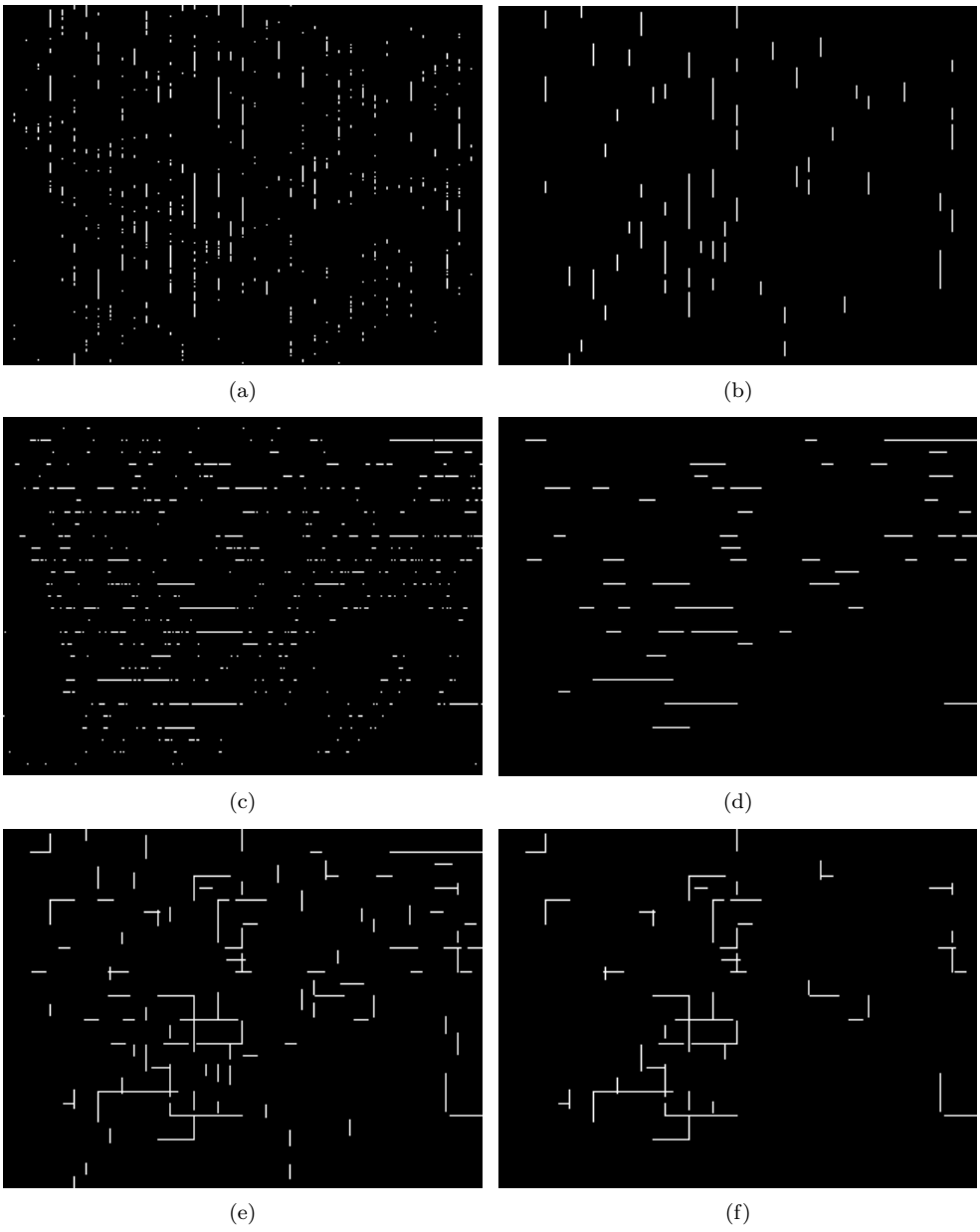


FIGURE 4. Blockiness detection process (a) Initial $I_{V,bin}$, (b) Processed $I_{V,bin}$, (c) Initial $I_{H,bin}$, (d) Processed $I_{H,bin}$, (e) Putting (b) and (d) together, (f) Final $I_{V,bin}$ and $I_{H,bin}$.

images which have the similar compression properties with H.264 compression. This database has different level of image quality. Among them, most of distorted images have blockiness impairments and some of distorted images have blur impairments. Also it provides the subjective MOS for each image (scale is from 0 to 100). Thus it is a good database to verify our blockiness algorithm. For the same database, in [24], it proposed one simple blockiness metric in eq. 2, i.e. the average differences

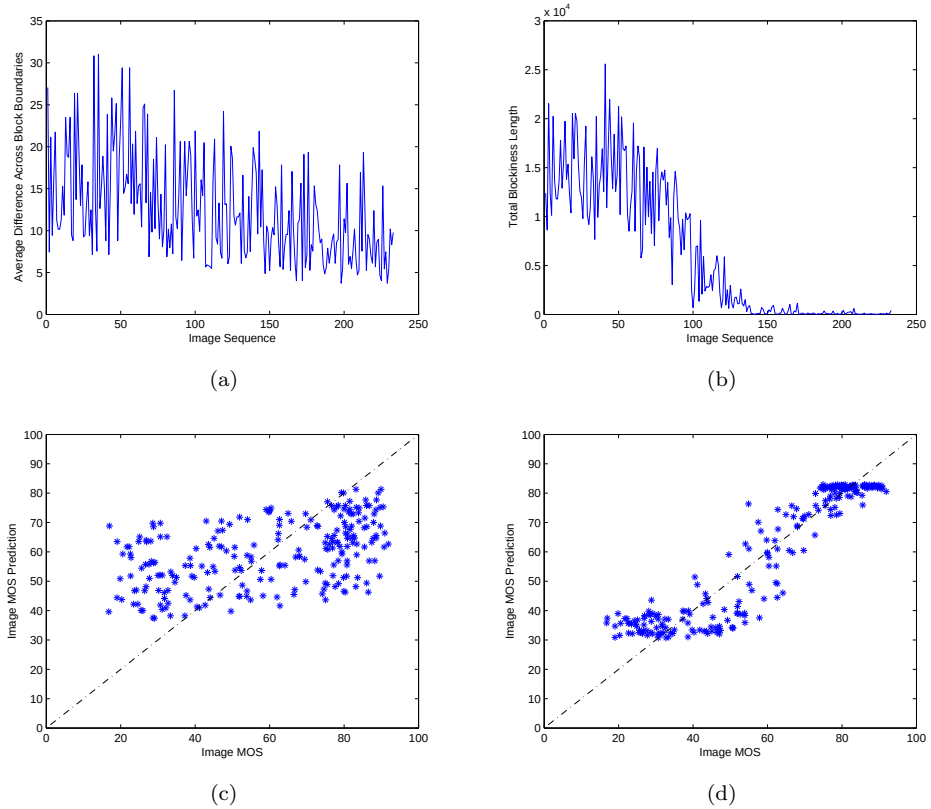


FIGURE 5. (a) average difference across block boundaries in [24] (b) total blockiness length in eq. (7) (c) image MOS prediction just using blockiness metric in [24] (d) image MOS prediction using our blockiness metric.

TABLE 1. Blockiness metric comparison

Metrics	Pearson	Spearman	RMSE
Metric in [24]	0.4861	0.4873	19.844
Our metric	0.9341	0.8891	8.1048

across block boundaries (multiple of 8). We will compare our blockiness metric with the one in [24].

First the 233 JPEG images are sorted in the ascending order for the subjective MOS. Then the blockiness metric in [24] is plotted in Fig. 5 (a) and our blockiness metric in eq. (7) is plotted in Fig. 5 (b). Bigger metric indicates more blockiness. We can see that our metric is decreasing much better than the one in [24] when the image quality is increasing. Also we use the logistic function in [21] to predict the image MOS using both metrics respectively. The results are shown in Fig. 5 (c) and (d) and the performance in terms of Pearson correlation coefficient, Spearman rank-order correlation coefficient, Root mean square error is presented in Tab. 1. As we can see, the prediction using our blockiness metric is much better than the one using the blockiness metric in [24].

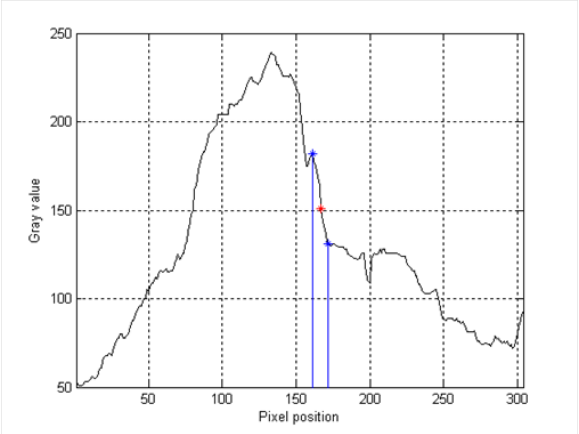


FIGURE 6. Edge width calculation.

3. BLUR DETECTION AND ESTIMATION

3.1. Algorithm Description. The HVS is very sensitive to the image edges. In [14], Marziliano et al. describe a method to compute the edge width. When the value of edge width is small, the edge appears being sharp. On the contrary, when the value of edge width is large, the edge appears as blurred. An image will be perceived to be annoyingly blurred if there are a lot of blurred edges in the image. In [14], the metric for measuring the amount of blurring is the average edge width for all strong vertical edges in one image. The blur metric in [14] is improved by Ferzli er al. in [6] by taking the Just Noticeable Blur (JNB) concept into account. Based on [6], Narvekar et al. in [16] uses a Cumulative Probability of Blur Detection (CPBD) as the blur metric. Following the analysis in [16], we find that their proposed blur metric is essentially calculated from the percentage of sharp edges among all the edges. However, we find that it is not necessary to compute the blur detection probability and CPBD in order to arrive at that percentage. And we can easily obtain that percentage right after computing the edge width. Thus our blur detection and estimation algorithm will utilize the edge width and JNB information to compute the percentage of JNB as our blur metric.

We first apply the Sobel vertical edge detection to identify the edge locations, where edge width is calculated as in [14]. A general method to compute edge width is illustrated in Fig. (6). The black curve represents the pixel values for one horizontal line in the image. The red point is a detected vertical edge point using the Sobel vertical operator. Intuitively, the magnitude of the slope at the red edge point is locally maximum, which means that the gradient at the red point is also locally maximum. Since the HVS is very sensitive to the large gradient and the corresponding point will be perceived as edge point. The left blue point is the local maximum closest to red point and the right blue point is the local minimum closest to the red point. At these two points, the two slopes are equal to zero and HVS will not be sensitive to the two points. Then the edge width for the red edge point is calculated as the horizontal length between the two blue points.

After computing the edge width for all the edge points, we utilize the concept of JNB, that is, the edge point is regarded as blurred point if the edge width is greater than a threshold (e.g. 5 pixels), and otherwise it is regarded as sharp point. Our blur metric is the percentage of blurred edge points among all the edge points, that

is,

$$F_{Blur} = \frac{number\,of\,blurred\,edges}{total\,number\,of\,edges}. \tag{8}$$

We can process the horizontal edges using the similar way as above for vertical edges. However, as in [6] and [14], we also find that such processing does not help improve accuracy of blur estimation, on the contrary, it will increase the computational complexity. Thus we only process the vertical edges. Also we notice that around the image borders there are usually some edges, which the HVS will not pay attention to. Hence we will remove a small part of the image (8 pixel width or height) close to the four image borders before doing any process.

The algorithm for blur detection and estimation described above can be summarized as follows.

Step 1: remove a small part of image close to each image border,

Step 2: convert color image to grayscale image,

Step 3: find vertical edges using the vertical Sobel operator,

Step 4: calculate the edge width for each vertical edge, if the edge width is bigger than JNB, it is blurred edge.

Step 5: compute the percentage of blurred edges and determine the blur metric.

3.2. Implementation and Discussion. We take one blurred image from our saved video at the receiver for the video telephony application over LTE networks. The blurred image could be due to video packet loss over LTE air interface in severe channel condition and the error concealment method of the video decoder cannot fully recover the corrupted image.

Fig. 7 (a) is the original blurred color image, Fig. 7 (b) is the corresponding grayscale image, Fig. 7 (c) is the vertical edge image after using vertical Sobel operator, and Fig. 7 (d) is the original color image with edge information. Specifically, the blues points are all the vertical edge points in Fig. 7 (c), and the red points are the edge points which edge width is greater than the threshold (5 pixel). We can clearly see that most blue points are overlapped by red points, which means most of edges are blurred edges.

Fig. 8 is the CDF graph for the edge width, where we can see that the percentage of blurred edges (edge width is greater than 5 pixel) is 76.8% (1-0.232). And this value is our blur metric in Eq. (8).

The latest blur estimation method is presented in [16] with a good performance. As we mentioned before, the key idea in [16] is to compute the percentage of blurred edge and we do not find the necessity of computing the cumulative probability of blur detection. We use the same dataset used in [16] to compare the effectiveness and efficiency. The dataset is originally from LIVE image database [29] and consists of 174 Gaussian blurred images which are generated using a 2-D Gaussian kernel of standard deviation ranging from 0 to 15. After we get the blur metric, as in [16] we also use the logistic function in [21] to predict the image MOS, which is compared with the subjective image MOS (scale is from 0 to 100) from LIVE database. The performance in terms of Pearson correlation coefficient, Spearman rank-order correlation coefficient, Root mean square error and Computational time is presented in Tab. 2.

We use the released software in [30] to get the CPBD metric performance and in the same code we just use our metric computation function instead of CPBD metric computation function to get our metric performance. The computational time in Tab. 2 is the average metric computation time for 174 Gaussian blurred

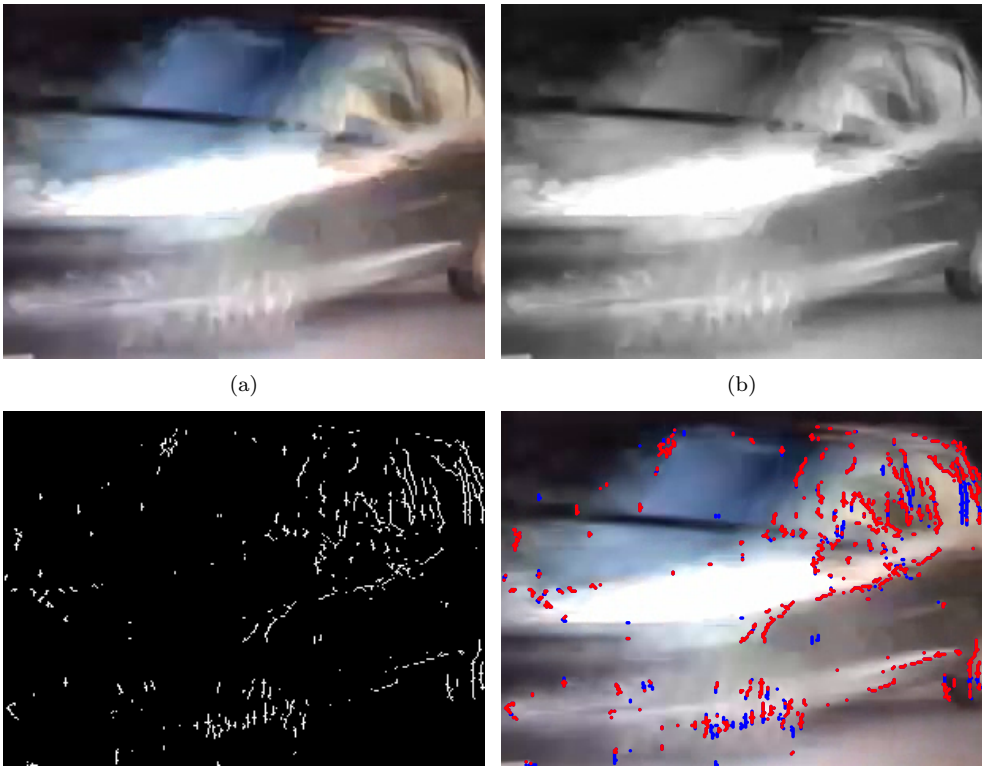


FIGURE 7. Verification of the blur algorithm (a) Original image with blur impairment, (b) Corresponding grayscale image, (c) Edges after vertical Sobel operator, (d) Original image overlapped with blue edges (c) and red edges with edge width greater than 5 pixels.

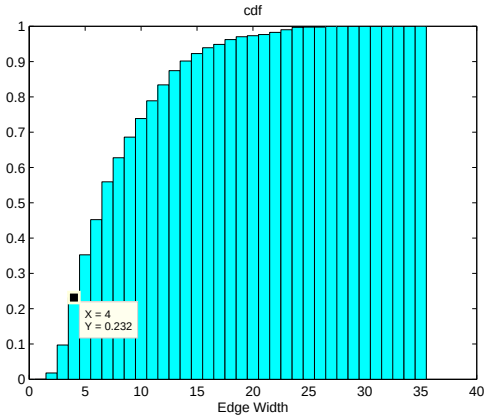


FIGURE 8. Edge width CDF.

TABLE 2. Blur metric comparison

Metrics	Pearson	Spearman	RMSE	Computational Time
CPBM metric	0.9256	0.9446	6.4688	4.86 sec
Our metric	0.9225	0.9432	6.5991	0.68 sec

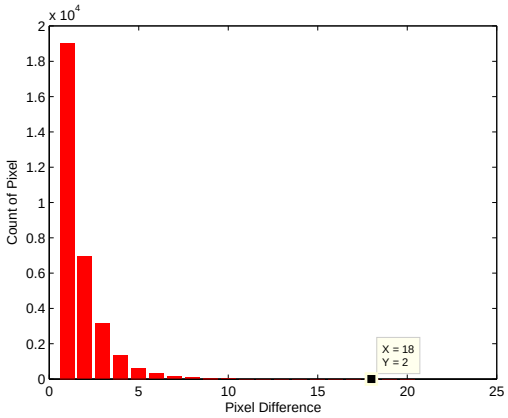


FIGURE 9. Histogram of pixel difference for two freezing frames.

images. Our metric algorithm is about 7 times faster than CPBD metric algorithm when it achieves the similar Pearson, Spearman, and RMSE performance.

4. FREEZING/JERKINESS DETECTION AND ESTIMATION

Freezing and jerkiness impairments are very annoying to end users. In order to detect and estimate them, we need the timing information for each frame. After we receive the video at UE, we will convert it to 30 fps uncompressed video. For mobile video telephony application, the frame rate will not exceed 30 fps and usually it is around 15 fps. Thus converting to 30 fps can catch all the frame information. With the constant 30 fps frame rate, each frame will display for 1000/30=33.3 ms and we get the timing information for each frame. The duplicate frames will be produced when converting from lower frame rat to 30 fps and the end user will have the same QoE for the video before and after conversion. We borrow the idea in [2] by Borer for freezing and jerkiness detection and estimation, and add the new process according to our practical need and problem.

Ideally when freezing happens, the difference of the frames during the freezing period is zero. However, in practice we notice that this is not the case. After changing the color image to grayscale image, we extract two freezing neighbor frames and compute the absolute difference of each pixel. And Fig. 9 is the corresponding histogram of pixel difference, where we can clearly see that there are still many pixels which have small difference. However, HVS will not perceive the small difference and this case will be experienced as a frame freeze. In order to deal with this case, we use a pixel difference threshold (in our case we use 15 for 0-255 gray image), and count all the pixels which have pixel difference greater than the threshold. If the count is less than the counting threshold (in our case we use 20, which needs adjustment for different video resolution), these two neighbor frames will be regarded

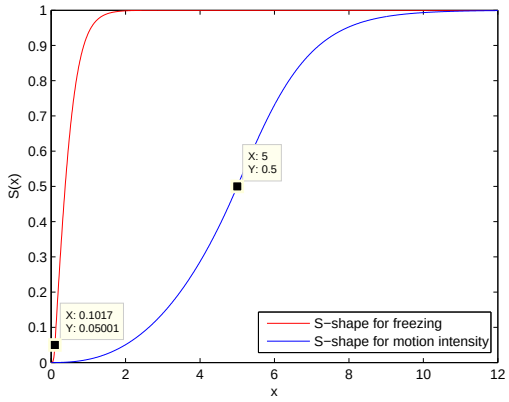


FIGURE 10. Sigmoid function for display time and motion intensity.

as freezing frames. In Fig. 9, the count number is only 2, so these two consecutive frames constitute a frame freeze.

Also we notice that during some long freezing time, some frames will shake a little bit. The freezing detection described above will regard these shaking frames as non-freezing frames, thus the long freezing will be divided into several parts. Usually for the shaking frames, the difference of neighbor frames is much smaller than the one of the natural frames. We use some threshold to distinguish them. If the sum of the total difference is less than 5000 pixels, they are shaking frames. Otherwise, they are natural frames. In order to solve this problem, we will connect the interrupted freezing frames if the number of detected different frames (actually they are shaking frames) between two consecutive freezing periods is less than some threshold (in our case we use 5 frames).

With above freezing detection and process, we can estimate the display time for each frame. Then the jerkiness metric is calculated as in [2].

$$F_{Jerkiness} = \frac{1}{T}\sum_{i=1}^{n} \Delta t_i \cdot \tau(\Delta t_i) \cdot (m_i), \tag{9}$$

where n is the total number of different non-freezing frames, T is the total video duration (in our case we choose 5 second for each freezing/jerkiness estimation), $\Delta t_i = t_{i+1} - t_i$ (t_i is the starting time for displaying the ith frame) is the display time for ith frame, m_i is the motion intensity for ith frame which is simply estimated by the root mean square of the neighbor frame pixel difference over the whole frame, both τ function and μ function are sigmoid function parameterized by (p_x, p_y, q) as in [2]

$$S(x) = \begin{cases} a \cdot x^b & if\, x \le p_x \\ \frac{d}{1+exp(-c\cdot(x-p_x))} + 1 - d & otherwise \end{cases}, \tag{10}$$

where $a = p_y / p_x^{(q \cdot p_x / p_y)}$, $b = q * p_x/p_y$, $c = 4q/d$, and $d = 2(1 - p_y)$. This sigmoid function polynomially increases from origin to (p_x, p_y), then exponentially increases from (p_x, p_y) to (inf., 1). q is the slope at the point (p_x, p_y).

In Eq. (9), when Δt_i (display time for ith frame) increases, jerkiness value will increase and more freezing/jerkiness will be perceived. The τ function and μ function are chosen from [2] and plotted in Fig. 10. When Δt_i is very small, $\tau(\Delta t_i)$

will be close to zero. Thus the jerkiness value will be also close to zero and the freezing/jerkiness will not be perceived. On the contrary, when Δt_i is very large, $\tau(\Delta t_i)$ will be close to 1 and usually in this situation $\mu(m_i)$ will be also close to 1. Thus the jerkiness value will be close to 1 and a lot of freezing/jerkiness will appear. The function τ is the red curve in Fig. 10 and has the parameters (0.12/1.18, 0.05, 1.5*1.18) as in [2] (in our case, we have QVGA size video which is close to CIF size). The function τ is to mimic the HVS for freezing. When the display time is very small, the function τ is close to zero. Then when display time increases, the function τ increases linearly. Finally when display time reaches to about 2 second, the function τ saturates to 1, which means HVS will notice jerkiness very much once the display time is beyond 2 second. the function τ is to take the motion intensity into account. When freezing happens, if the two frames before freezing and after freezing have a big motion, HVS will experience a lot of jerkiness. On the contrary, if the two frames have a small motion, HVS will experience less jerkiness. And the function μ will saturate to 1 if motion intensity reaches to around 10. As in [2], we also use the function μ with the parameters (5, 0.5, 0.25).

5. MOS PREDICTION

For video quality evaluation, the final result is MOS. Based on the three important impairment estimations described above, we are able to predict the MOS. First, we need the subjective MOS database. The video samples come from video telephony testing in different conditions over LTE networks. Video sources have 8 different contents in order to have diverse scenes from low motion to high motion. They are Calendar, Container, Foreman, News, Mother & daughter, Car, Grandpa, and Football [31]. In order to have different level of impairments, video telephony is run in different LTE network conditions, such as near cell, middle cell, edge cell, different fading conditions (slow speed, high speed), different mobility (intra handoff, inter handoff), and different interference. From a large set of received video, we are able to choose different video content with different level of impairments. For our own created dataset, we need subjective MOS. We adopt Absolute Category Rating (ACR) subjective test method, which is no-reference method. We use the EyeOne Display 2 calibration tool to calibrate the Samsung SyncMaster 24 inch monitor for brightness, black level and white point conforming to the standard specified in [11]. Then we assemble a team of people, give them some instruction in the beginning, and ask them to evaluate the videos and provide a score from 1 to 5 (1: bad, 2: poor, 3: fair, 4: good, 5: excellent) for each video. We deleted some outlier subjective MOS and calculated the mean of the remaining usable ones for each video sample. Eventually we got the subjective MOS for 94 video samples. Each video sample has QVGA size, and is 5 second long and encoded in H.264 with bit rate about 350 kbps. This dataset includes 14 instances of Calendar, 9 of Container, 10 of Foreman, 17 of News, 13 of Mother & daughter, 10 of Car, 10 of Grandpa, and 11 of Football. Four videos belong to [1 1.5), 13 videos belong to [1.5 2.5), 34 videos belong to [2.5 3.5), and 43 videos belong to [3.5 4.5). Since videos are QVGA size and about 350 kbps, none of them is given the score of 5 (excellent).

After we compute the estimation metrics for the three important impairments described in previous sections, we need further processing before using them to predict the subjective MOS. After we get the blockiness and blur metrics for each image in the 5 second long video, the blockiness and blur impairments will be smoothed out if we just use the mean value for all the images. Instead we use 75th percentile value among all the values for both blockiness and blur respectively. In

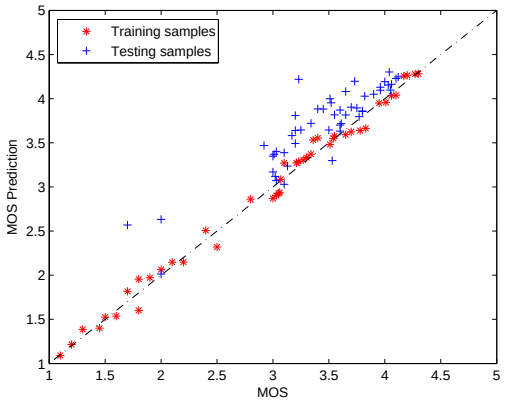


FIGURE 11. MOS prediction.

order to make our prediction model simpler, we can normalize all the metrics to the range of [0 1]. Blur metric is already in the range of [0, 1]. For blockiness metric, we use the sigmoid function as Eq. (10) (we choose (20, 0.1, 0.08) parameters) to map the blockiness metric measured as Eq. (7) to [0, 1]. Jerkiness metric is already processed based on the video sequences and is in the range of [0, 1]. Thus it does not need to be further processed. Then we use some weighting factors for each impairment metric considering the relative importance to HVS and integrate them into one final metric as follows.

$$F = w_1 \cdot F_{Jerkiness} + w_2 \cdot F'_{Blockiness} + w_3 \cdot F'_{Blur}, \tag{11}$$

where $F'_{Blockiness}$ and F'_{Blur} are the processed blockiness metric and blur metric from $F_{Blockiness}$ and F_{Blur} as described above, and w_1, w_2, w_3 are weighting factors. We choose $w_1 = 0.55$, $w_2 = 0.4$, $w_3 = 0.25$ considering that HVS will feel more annoying in the sequence of jerkiness, blockiness and blur. In general, the predicted MOS will decrease when F value increases, that is, it will be large for small F, and it will be small for large F.

The last step is to choose an appropriate model to compute the objective MOS based on the final metric F in Eq. (11). We evaluated several models for our dataset, such as model given by (7) in [24], model in the VQEG report [21], BP (back propagation) neural network model and polynomial curve fitting model. Both neural network model and polynomial curve fitting model give us a very good performance (Pearson correlation is above 90%). Considering the simplicity of polynomial curve fitting, we choose it as our final model to predict the subjective MOS. We use the fourth order polynomial function, that is,

$$MOS(F) = a_4F^4 + a_3F^3 + a_2F^2 + a_1F + a_0. \tag{12}$$

Among our 94 video samples, we choose 44 samples where the subjective MOS scores cover the whole score range [1 5] for the curve fitting model (12). That is, for each video, i, of the 44 samples, whose subjective MOS score is MOS_i, we calculate the metric F_i according to Eq. (11), where $i = 1, ..., 44$. Then the values (F_i, MOS_i), $i = 1, , 44$, are used for the curve fitting in (12). And we get coefficient values as $a_4 = 210.62$, $a_3 = -233.55$, $a_2 = 80.82$, $a_1 = -15.25$, $a_0 = 4.62$. Then we use the remaining 50 samples for testing. For our model (12), if the output value is lower than 1, we will let it be 1. And if the output value is bigger than 5, we will let

it be 5. The predicted MOS using Eq. (12) and corresponding subjective MOS for each sample are potted in Fig. (11). The red points are for training samples, which have the Pearson correlation 0.9950 and the blue points are for testing samples, which have the Pearson correlation 0.9010. Overall, the Pearson correlation for all samples is 0.9633.

6. Conclusion

In this paper, we propose a NR video quality analysis framework based on a received decoded video obtained from our VT testing over LTE networks in various channel and network conditions. Effective blockiness and blur detection and estimation algorithms are proposed. Also our provided examples visually verify the effectiveness of our novel algorithms. Freezing/jerkiness detection and estimation take the subtle change between freezing frames and shaking distortion during freezing period into account, and mainly use parameterized sigmoid function to mimic the impact of freezing and motion intensity on HVS. After further post-processing for the blockiness and blur metrics and combining three metrics into one weighted final metric, the predicted MOS is obtained by polynomial curve fitting model which is trained and tested through our video database from practical VT testing over LTE networks, and above 90% Pearson correlation is achieved.

Although our video database is from specific VT testing with certain bit rate and frame resolution, our NR video quality analysis can be easily extended to other bit rates and frame resolutions through some parameters adjustment and further training with the new database. The VT application in this paper is over LTE networks, however, our video quality analysis techniques can also be applied into other networks again with some parameters adjustment. Our current method is still offline analysis. Our future work will include extend it to real-time analysis by acquiring the decoded frame through the UE in real time and transporting it to the high-performance computer for further analysis. Furthermore, hybrid NR video quality model by combing the bit-stream analysis with our current decoded video analysis is our ongoing work.

References

[1] S. Argyropoulos, A. Raake, M. Garcia, and P. List, NO-REFERENCE BIT STREAM MODEL FOR VIDEO QUALITY ASSESSMENT OF H.264/AVC VIDEO BASED ON PACKET LOSS VISIBILITY, in IEEE Int. Conf. on Acoustics, Speech and Signal Process. May, 2011, pp. 1169-1172.

[2] S. Borer, A MODEL OF JERKINESS FOR TEMPORAL IMPAIRMENTS IN VIDEO TRANSMISSION, in Second International Workshop on Quality of Multimedia Experience, 2010, pp. 218-223.

[3] D. M. Chandler and S. S. Hemami, VSNR: A wavelet-based visual signal-to-noise ratio for natural images, IEEE Trans. Image Processing, vol. 16, no. 9, pp. 2284-2298, Sep. 2007.

[4] J. Choe, K. Lee, and C. Lee, No-reference video quality measurement using neural networks, in 16th Int. Conf. on Digital Signal Processing, July, 2009, pp. 1-4.

[5] Cisco White Paper: Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012-2017.

[6] R. Ferzli and L. J. Karam, A no-reference objective image sharpness metric based on the notion of just noticeable blur (JNB), IEEE Trans. Image Process., vol. 18, no. 4, pp. 717-728, Apr. 2009.

[7] I. P. Gunawan and M. Ghanbari, Reduced-reference video quality assessment using discriminative local harmonic strength with motion consideration, IEEE Trans. Circuits Syst. Video Technol., vol. 18, no.1, pp. 71-83, Jan. 2010.

[8] S. Hemami, A. Reibman, No-reference image and video quality estimation: Applications and human-motivated design, Signal Processing: Image Communication, 25(7): 469-481, 2010.

[9] ITU-T Recommendation J.246, Perceptual visual quality measurement techniques for multimedia services over digital cable television networks in the presence of a reduced bandwidth reference, International Telecommunication Union, Geneva, Switzerland, 2008.
[10] ITU-T Recommendation J.247, Objective perceptual multimedia video quality measurement in the presence of a full reference, International Telecommunication Union, Geneva, Switzerland, 2008.
[11] ITU-T Recommendation P.910, Subjective video quality assessment methods for multimedia applications, International Telecommunication Union, Geneva, Switzerland, 2008.
[12] C. Keimel, M. Klimpke, J. Habigt and K. Diepold, NO-REFERENCE VIDEO QUALITY METRIC FOR HDTV BASED ON H.264/AVC BITSTREAM FEATURES, in 18th IEEE Int. Conf. Image Processing, 2011, pp. 3325-3328.
[13] L. Ma, S. Li and K. N. Ngan, Reduced-Reference Video Quality Assessment of Compressed Video Sequences, IEEE Trans. Circuits Syst. Video Technol., vol. 22, no.10, pp. 1441-1456, Oct. 2012.
[14] P. Marziliano, F. Dufaux, S. Winkler, and T. Ebrahini, Perceptual blur and ringing metrics: Applications to lPEG2000, Signal Proc.: Image Comm., vol. 19, pp. 163-172, 2004.
[15] R. Muijs, and I. Kirenko, A No-Reference Blocking Artifact Measure for Adaptive Video Processing, in European Signal Processing Conference, 2005.
[16] N. D. Narvekar and L. J. Karam, An Improved No-Reference Sharpness Metric Based on the Probability of Blur Detection, Int. Workshop on Video Processing and Quality Metrics for Consumer Electronics (VPQM), Jan. 2010.
[17] J. Shen, Q. Li, and G. Erlebacher, No-reference Video Quality Assessment for Noisy, Blurred, and MPEG-2 Natural Videos, submitted to IEEE Trans. Image Process.
[18] N. Staelens, I. Sedano, M. Barkowsky, L. Janowski, K. Brunnstrom and P. L. Callet, STANDARDIZED TOOLCHAIN AND MODEL DEVELOPMENT FOR VIDEO QUALITY ASSESSMENT - THE MISSION OF THE JOINT EFFORT GROUP IN VQEG, in third Int. Workshop on Quality of Multimedia Experience, 2011, pp. 61-66.
[19] N. Staelens, N. Vercammen, Y. Dhondt, B. Vermeulen, P. Lambert, R. V. Walle, and P. Demeester, VIQID: A NO-REFERENCE BIT STREAM-BASED VISUAL QUALITY IMPAIRMENT DETECTOR, in second Int. Workshop on Quality of Multimedia Experience, 2010, pp. 206-211.
[20] O. Sugimoto, S. Naito, S. Sakazawa and A. Koike, OBJECTIVE PERCEPTUAL VIDEO QUALITY MEASUREMENT METHOD BASED ON HYBRID NO REFERENCE FRAMEWORK, in 16th IEEE Int. Conf. Image Processing, 2009, pp. 2237-2240.
[21] VQEG, Final report from the Video Quality Experts Group on the validation of objective models of video quality assessment, 2000.
[22] Z. Wang, A. C. Bovik, H. R. Sheihk, and E. P. Simoncelli, Image quality assessment: From error visibility to structural similarity, IEEE Trans. Image Processing, vol. 13, no. 4, pp. 600-612, Apr. 2004.
[23] Z.Wang, H. R. Sheihk, and A. C. Bovik, Objective video quality assessment, in The Handbook of Video Databases: Design and Application, B. Furht and O. Marqure, Eds. Boca Raton, FL: CRC Press, Sep. 2003, pp. 1041-1078.
[24] Z. Wang, H. Sheikh, and A. Bovik, No Reference Perceptual Quality Assessment of JPEG Compressed Images, in Proc. IEEE Int. Conf. Image Processing, Rochester, NY, 2002, pp. 477-480.
[25] A. Webster, P. Corriveau, V. Baroncini, and M. Pinson, Standardization Trends in Video/Multimedia Quality Assessment, in tutorial IEEE Int. Conf. Image Processing, Orlando, FL, 2012.
[26] S. Winkler and P. Mohandas, The evolution of video quality measurement: From PSNR to hybrid metrics, IEEE Transactions on Broadcasting, vol. 54, no. 3, pp. 660-668, September 2008.
[27] F. Yang, S. Wan, Q. Xie, and H. Wu, No-Reference Quality Assessment for Networked Video via Primary Analysis of Bit Stream, IEEE Trans. Circuits Syst. Video Technol., vol. 20, no.11, pp. 1544-1554, Nov. 2010.
[28] S. Zhao, H. Jiang, Q. Cai, S. Sherif and A. Tarraf, Hybrid Framework for No-reference Video Quality Indication Over LTE Networks, the 23rd Wireless and Optical Communication Conference, May 2014.
[29] H. R. Sheikh, A. C. Bovik, L. Cormack and Z. Wang, LIVE image quality assessment database, 2003, http://live.ece.utexas.edu/research/quality.

[30] N. D. Narvekar and L. J. Karam, CPBD Sharpness Metric Software, http://ivulab.asu.edu/Quality/CPBD.
[31] http://media.xiph.org

Alcatel-Lucent USA, 600 Mountain Ave, Murray Hill, NJ 07974
E-mail: Songqing.Zhao@alcatel-lucent.com

Alcatel-Lucent USA, 600 Mountain Ave, Murray Hill, NJ 07974
E-mail: Hong.Jiang@alcatel-lucent.com