Frame Interpolation for Cloud-Based Mobile Video Streaming

Muhammad Usman, *Student Member, IEEE,* Xiangjian He, *Senior Member, IEEE,* Kin. M. Lam, *Senior Member,* Min Xu, *Member, IEEE,* Syed Mohsin M. Bokhari, *Member, IEEE and* Jinjun Chen, *Senior Member, IEEE*

***Abstract*— Cloud-based High Definition (HD) video streaming is becoming more and more popular day by day. On one hand, it is important for both end users and large storage servers to store their huge amount of data on different servers at different locations from security and mobile availability point of views, especially for end users having small amount of storage in their mobile devices. On the other hand, it is becoming a big challenge for network service providers to provide constant and reliable connectivity to the network users. There have been many studies over cloud-based video streaming for Quality of Experience (QoE) for services like Youtube. Packet losses and bit errors are very common in transmission networks, producing annoying effects such as frame freezing and blocky artifacts, which affects the user feedback over cloud-based media services. To cover up packet losses and bit errors, Error Concealment (EC) techniques are usually applied at decoder/receiver side to estimate the lost information. This paper proposes a time efficient and quality oriented EC method. The proposed method considers H.265/HEVC based Intra-encoded videos for the estimation of whole Intra-frame loss. The unsliced mode of H.265 is targeted for the proposed approach. The main emphasis in the proposed approach is the recovery of Motion Vectors (MVs) of a lost frame in real-time. The search to find the optimum MV is performed in parallel in nearby four sub-blocks in the reference frame. To boost-up the search process for the lost MVs, a bigger block size and searching in parallel are both considered. The simulation results clearly show that our proposed method outperforms the traditional Block Matching Algorithm (BMA) by approximately 2.5 dB and Frame Copy (FC) by up to 12 dB at a Packet Loss Rates of 1%, 3% and 5% with different Quantization Parameters (QPs). The computational time of the proposed approach outperforms the BMA by approximately 1,788 seconds. The proposed technique can readily be applied for real-time cloud-based HD video streaming.**

***Index Terms*—High Definition, Quality of Experience, Packet Retransmission, Error Concealment, Block Matching Algorithm, Motion Vectors, H.265/HEVC, Intra Frame, Packet Loss Rate, Quantization Parameter.**

Manuscript submitted on September 01, 2015 and asterisk indicates corresponding author.

Muhammad Usman, Prof. *Xiangjian He, Dr. Min Xu and Associate Prof. Jinjun Chen are with Global Big Data Technologies Center (GBDTC), School of Computing and Communications, University of Technology Sydney, Australia. (e-mail: Muhammad.Usman-2@student.uts.edu.au, Xiangjian.He@uts.edu.au, Min.Xu@uts.edu.au, Jinjun.Chen@uts.edu.au).

Dr. Syed Mohsin Matloob Bokhari is with Department of Electronic Engineering, University of Engineering and Technology Peshawar, Pakistan (e-mail: Mohsin.Bokhari@uetpeshawar.edu.pk).

Prof. Kin Man Lam is with Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: enkmlam@polyu.edu.hk).

I. INTRODUCTION

THERE is a tremendous growth in the usage of streaming videos nowadays. It has been reported in [1] that the global Internet consumer video traffic will be 80 percent of all Internet consumer traffic in 2019, which is an increase of 64 percent from what was reported in 2014. In the case of mobile-users, the increase will be 10-fold from 2014 to 2019. This report indicates a huge amount of video streaming that may lead to degradation in QoE for end users.

The cloud services are becoming more and more popular day by day on both computing and storage aspects and producing a major impact on current IT and research industry [2]. Today's companies are shifting IT activities to third party storage servers and computing platforms. The most famous cloud-based media services include Youtube, Dailymotion, Microsoft Azure and Putlocker. A general framework of cloud-based media streaming is shown in Fig. 1. Cloud computing provides not only high computation power but also huge amount of storage space, so that it enables end users to utilize best services without having an expensive and complex architecture at their side [3]. However, cloud architecture suffers from transmission degradations, especially transmission errors and security threats in cloud networks [4]. The QoE is becoming a criterion in quality testing for end users to rate media streaming services provided by clouds [5].

Nowadays, videos are becoming more attractive in resolution, and we now see HD and Ultra HD (UHD) techniques. To process and store such high resolution videos, cloud computing provides an effective platform for computing, storage and transmission. There are many media servers which are actually cloud data centers. Some well-known cloud based media servers are MediaFire, YouTube, DailyMotion, PutLocker and many more.

Current video processing standards such as H.264 and H.265 are dealing with immense amount of data, which ultimately demands either parallel or distributed processing platforms. In the case of cloud computing, many Video Service Providers (VSPs) can rent out the distribution architecture from Cloud Service Providers (CSPs) to deliver video streams to a large number of mobile end users [6, 7].

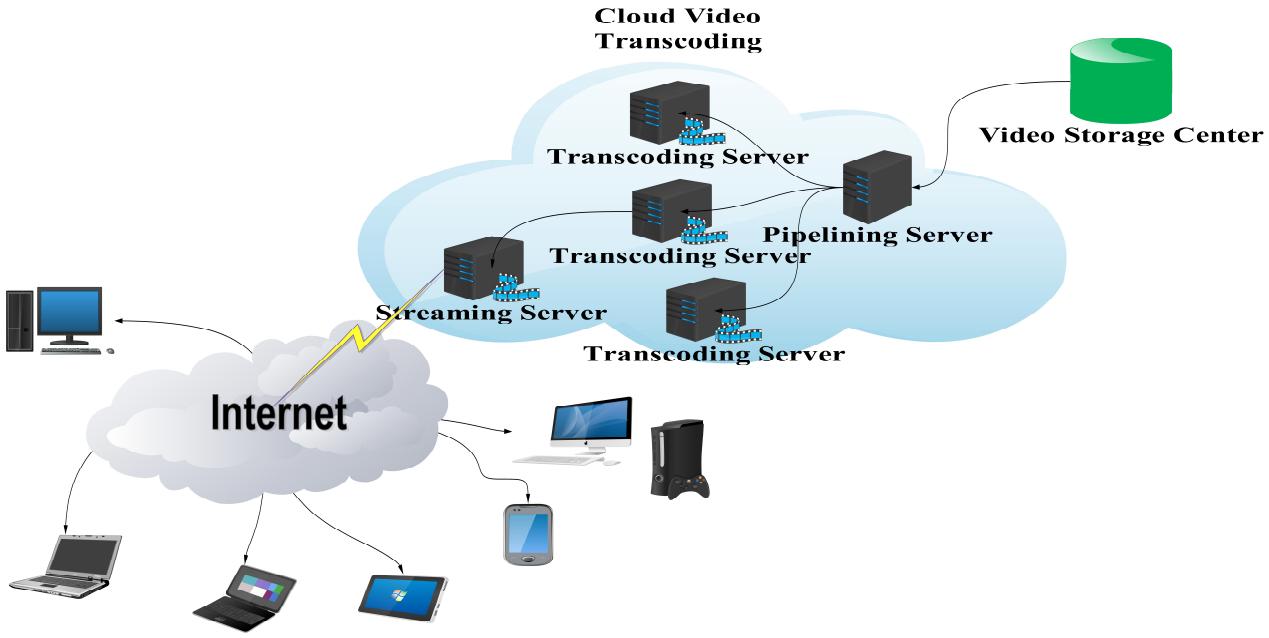


Fig. 1. The generic architecture of cloud-based media streaming

The main advantage of such approaches is the computing power, which is available any time at the VSP side while the drawback is extra management and cost.

In order to tackle with QoE, a VSP needs to intelligently deal with both cloud and network service providers. The HD video streaming is always bandwidth sensitive, so it leads to many challenges 1) Distribution of data at different locations demands proper management of available sources, 2) Different geographical locations will be connected through Internet, which is based on best effort service 3) As the Internet is a combination of low and high speed networks, which may cause data buffering at many locations during data transmission, out of order delivery or packet loss or drops may occur 4) Video streaming applications run on connectionless protocols, which does not provide guaranteed and safe delivery of transmitted packet [8].

In the case of cloud server-based media streaming, unreliable transmission channels are the biggest issue. The current video coding standard H.265/HEVC provides approximately half bit rate with same visual quality as compared to previous standard H.264/AVC [9]. As H.265 encoded videos are highly compressed, little packet loss or smaller bit errors may still lead to huge quality degradations. There are mainly two modes of encoding in H.264 and H.265 i.e. Intra and Inter. In case of Intra mode, each video frame is treated independently and known as Intra or I-frame. On the other hand, Inter mode is a combination of I, P (Predictive) and B (Bidirectional) frames. In this mode, video frames are always encoded as a Group Of Pictures (GOP). In each GOP, there is always one I-frame, which is always the first frame. Then there are P-frames which occur at regular intervals and always dependent on either previous I or P-frame. The B-frames occur mostly and are dependent on previous and upcoming I and P-frames. Unlike Intra encoded videos, such errors can easily propagate to subsequent frames in inter encoded videos, if they are not handled appropriately. To protect the binary encoded streams, the Error Resilience (ER) techniques are usually applied at encoder side. The ER techniques may protect the binary streams from content modification but cannot fight against the packet drops. The EC techniques are used at decoder side, are very popular to deal with such problems and can estimate the lost information without demanding any change in decoder architecture [10].

The main idea behind the EC is to estimate the video packet, lost during transmission. The lost video packet may contain parts of a video frame or one complete frame, depending upon the encoding parameters. The EC techniques utilize the correlation between the frames of the same shot in a video sequence. Usually, there are two types of correlations in video streams, namely Spatial and Temporal. Based upon these two categories, EC techniques can be classified into three types which are Spatial Error Concealment (SEC), Temporal Error Concealment (TEC) and a combination of both. SEC techniques are mostly used for concealment of Intra frames (I-frames) while TEC techniques are mostly used for concealment of Inter frames (I, B and P-frames). As Inter frames also involve I-frames, TEC techniques can also be applied to Intra-frames [11].

In this paper, we propose a new algorithm for concealment of whole frame loss of Intra encoded videos, stored on cloud-based media servers. The proposed algorithm conceals the Intra frames lost during the media streaming with following contributions:

- Our proposed algorithm does not require any changes in decoder architecture. Rather it works as a separate processing module, which is called by decoder, whenever required. The existing work for EC mostly focuses on performing such tasks at cloud side using H.264 [12]. The main drawback of the existing approach is the processing overload at the streaming server, which may cause delays and slow down the

performance of the streaming server. Our proposed approach runs at user side and reduces the processing load of cloud-based media server by utilizing the processing capability of end-user mobile devices.

- Our proposed algorithm performs in parallel, so it can be implemented on the microprocessors of user mobile devices. These parallel computations are threshold based and are not running all the time iteratively, so they produce minimum computational overhead and hence can easily support real-time HD video streaming. The experimental results show that our proposed algorithm produces less computational time as compared to traditional Boundary Matching Algorithm (BMA) [14].
- The proposed approach has also considered power saving. The user mobile devices always run on chargeable batteries. The duration and amount of stored charge decrease as the processing power increases [13]. The threshold-based parallel processing decreases the computational time and ultimately saves the stored charge.
- The experimental results show that our proposed algorithm produces better visual quality in terms of average PSNR. The experimental results also show an improvement of approximately 2.5 dB from the traditional BMA [14] and approximately 12 dB from Frame Copy (FC) [15].

The remainder of this article is organized as follows. The related work of EC is presented in Section II. In Section III, the proposed algorithm is introduced. The experimental setup, results and comparison are presented in Section IV. Finally, the paper is concluded in Section V.

II. RELATED WORK

To cover up video packet delays and drops during transmission, EC techniques play an important role. EC techniques always run at receiver side and work as a separate module, which does not demand any change in decoder architecture. This section summarizes classic error concealment techniques.

A. Spatial Error Concealment

The Spatial Error Concealment (SEC), also known as Intra EC, is pixel based approach and use information from neighborhood blocks to estimate the missing ones [16].

In [17], the texture modeling is used to perform geometric interpolation to estimate the lost pixels. Though it produces better results, is suitable for large block sizes only. The mean square estimation is used in combination with probability distribution function to restore the missing block of pixels [18]. The technique produces fine quality at the cost of computational complexity.

The directional edge analysis is used to estimate the edges of missing areas in a video frame [19]. This approach works well with sufficient information from nearby edges but does not produce better results in heavy losses. The relevant edges in the presence of digital drop out errors are estimated through canny edge detector [20]. This approach produces better results in linear motion only. To estimate whole frame lost, pixel-wise disparity matching technique is used in [21]. This technique works well for whole frame loss with affordable reconstruction quality but does not support consecutive frame losses.

B. Temporal Error Concealment

The Temporal Error Concealment (TEC), also known as Inter EC, uses the correlation between video frames to find out the MVs of lost block of pixels [16].

The boundary matching score in combination with object detection is used to estimate the MVs of lost blocks in a frame [22]. This approach works well in the presence of multiple objects in a video frame. The dynamic programming is used to recover the MVs of lost blocks in a frame iteratively [23]. This approach produces better results at the cost of computational time. In [12], the MVs of neighborhood frames are used to estimate a corrupted block in a frame. This approach is simple and has simpler computations but does not produce better results in slow motion videos. The auto-regressive modeling is used to refine the estimated MVs of lost blocks in a video frame [24]. This approach only supports videos having linear motions only.

The depth information in a 3D video is used to estimate the MVs of lost blocks in a frame of the video in [25]. The lost MVs are reconstructed using previous Intra-frames [26]. Instead of using depth information, the Motion Vector Extrapolation (MVE) algorithm is used to estimate the lost MVs [27]. The approaches in [25-27] produce better reconstruction quality but support 3D videos only.

C. Hybrid Error Concealment

The Hybrid Error Concealment (HEC) techniques are either a combination of techniques from the SEC and TEC domains or techniques from other domains of signal and image processing to perform concealment. The Multiple Description Coding (MDC) is combined with the SEC and TEC domain techniques to produce better concealment results [28]. This technique produces better results but is limited to random packet loss only. To support random packet losses, the MDC is combined with Set Partitioning in Hierarchical Trees (SPIHT) to restore missing blocks [29]. To restore the region of interest, videos are transmitted in two streams, having high and low resolutions respectively [30]. The techniques presented in [29, 30] produce better reconstruction quality at the cost of availability of high bandwidth.

III. FRAME INTERPOLATION

In this section, we present our proposed technique. Motivated by the BMA and FC, we propose an EC technique based on multiple threading. The proposed EC technique, named as Frame Interpolation (FI) is illustrated in Fig. 2. For each lost Intra-frame, the MVs are derived first. The lost MVs are estimated through a Motion Estimation (ME) scheme. The main time consuming process during video encoding is always the ME. The ME consumes 40% to 80% of computational time [31]. In our approach, the main emphasis is to reduce computational complexity and time, as we cannot expect a high computational power in any end-user mobile device. This reduction in complexity of computations ultimately reduces

computational time, which helps in supporting real-time processing. The ME process is performed by using two frames at a time, i.e., the previous and the next frame. The blocks in the previous and the next frames represent the tail and head of estimated MVs respectively, i.e., the starting point of any MV is known as tail while the ending point of any MV is known as head as shown in Fig. 2. The BMA is used as a base algorithm for further enhancement of the ME process. The estimated MVs then help to interpolate the missing/lost frame. The interpolated frames are passed through an adaptive filter to remove the false estimation noises, if any. The details of the FI are discussed in the following subsections.

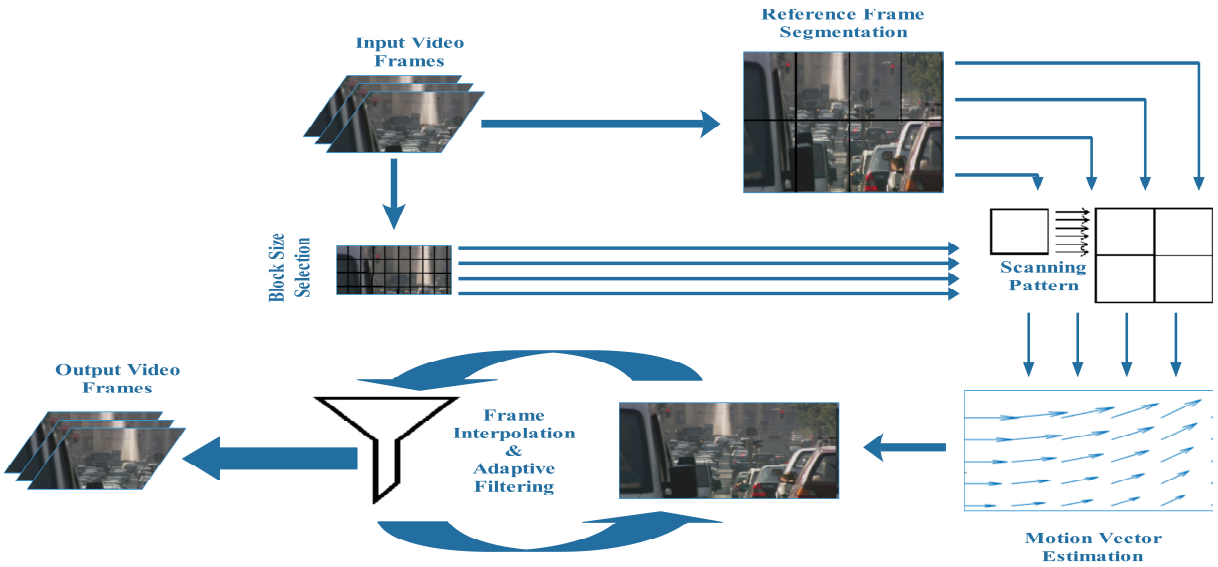


Fig. 2. Block diagram of the proposed algorithm

A. *Block Size Selection*

Before starting the procedure of the EC, it is important to divide the current frame into blocks but there is no specific criterion for that. This division totally depends upon the nature of information and applications. The sizes and shapes of the segmented blocks need to be considered when the current frame is divided into blocks.

For the division of a frame into blocks, there are two options determining the sizes of blocks. We could either consider the whole frame as one block or divide the frame into blocks of equal/unequal size. The first option demands large buffer memory and may slow down the performance of the system. This option also requests the whole frame to be processed although there may be specific regions that can easily be skipped based upon the repetitive information. The above drawbacks can be minimized using the second option, which is to segment a frame into number of blocks. Depending upon the nature of contents in a video frame, either the fixed or variable sized segmentation can be opted. In the proposed FI, we select the second option, and divide the frame into number of blocks of fixed size. During experiments, the size of 16×16 is proved to be an optimal size on average. As our algorithm is basically designed for HD and higher resolution videos, a larger block size creates shorter computational overhead. On contrast, smaller and variable sized divisions produce better fineness in results but at the cost of longer processing overhead and higher computational complexity.

The shape of a block can be square, rectangular, triangular and irregular. Out of all these possibilities, the square shape is widely adopted. One main reason for the popularity of square shape is the even resolution of test videos. The test videos are always available in even resolution as shown in Table II. Traditional BMAs such as Full Search (FS), Three Step Search (TSS) and Four Step Search (FSS) use square shape while others use diamond, cross or hexagonal shapes [32]. Computational time depends upon the number of pixels in a selected shape. A non-squared shape i.e., hexagonal or diamond keeps coordinate information of frame pixel compared to a square shape, and hence requires further processing, especially in a HD and higher resolution video. In square shape, only starting and ending point is required while in hexagonal or diamond, checks and breaks are required in rows and columns to create hexagonal or diamond shape. In our proposed algorithm, to reduce the amount of information, required during the processing, a square block shape is considered. Experiments show that using a square shape produces better results and requires less time as compared to other shapes.

In FI, each Input Video (IV) is read in frame by frame. If frame *N* is missing, then frame *N-1* and *N+1* are selected for the EC process. Otherwise frame is written to Output Video (OV) file. To process HD frames quickly, a block size of 16×16 is used to perform EC at a decoder side. The blocks of the current frame are denoted as Input Blocks (IBs) and the blocks of a reference frame are denoted as Reference Blocks (RBs).

B. Reference Frame Segmentation

The second step in any EC scheme is to divide each reference frame into areas, known as Search Windows (SWs). Like the division of the current frame, there is no specific criterion for such division and the division totally depends upon the nature of information and applications. As described for the current frame, two factors need to be taken into account when dividing a reference frame into SWs. These factors are the size and shape of a SW.

For the same reasons for determining the division of the current frame, in FI, we choose to divide each reference frame into multiple SWs of a fixed size. When determining the size of SWs, we assume that the motion of pixels is not very fast. Therefore, the SW size is selected to be four times of the IB size in FI. Thus, the SW size is 64×64. It has been proved that the SW size can be further reduced for more efficient computations if needed [33].

In terms of shape of SWs, traditional BMAs such as Full Search, Three Step Search and Four Step Search, the SW patterns are of square shape. However, it is not necessary to have square shape. The shape can be irregular or can be adaptive [34, 35]. However, irregular and adaptive shapes usually require extra processing time due to resolution variation from video to video. In order to be adapted to videos of any resolution, square shape is adopted in our proposed algorithm.

C. Scanning Pattern

After defining the block size and search window, the next step is how to find the best matching block in a reference frame. In BMA, the most famous search is Full search (FS). This technique is linear and tries to find the best match exhaustively. The main advantage of FS is its high accuracy while the drawback is its computational complexity and hence is inappropriate for real-time applications.

In our proposed technique, the FS is used with modifications. Unlike the traditional FS, our approach does not search every pixel in an SW. Our approach considers the assumption that pixels cannot move that far in consecutive frames. There is another possibility that more than one match are found during the matching process at different locations. To overcome this problem, we set a threshold and terminate the search process as soon as the first best or near match is found.

There are other state-of-art search techniques such as Three Step Search (TSS), Four Step Search (FSS), Two Dimensional Logarithmic Search (TDLS), Simple and Efficient Search (SES), Diamond Search (DS) and Adaptive Rood Pattern Search (ARPS), among which ARPS is the latest one [36]. These techniques are very attractive in terms of computational time. However, their searching accuracies do not beat the searching accuracy of FS [37]. To further speed up the searching process, in our proposed method, multiple threads based parallel processing is considered. Each SW is divided into N number of partitions, known as RBs. The size of each RB is equal to the IB size. A random value is set as an initial threshold. The block matching process is performed in parallel in each partition. The matching is performed pixel by pixel just like the FS. If the matching value is below the initial threshold, the search moves on to the next pixel in sequence until a matching value greater than the initial threshold is found to terminate the searching process. As soon as the search is terminated, the threshold value is replaced by the minimum value, found in the set of previously searched minimum values.

D. Motion Vector Estimation

The Motion Vector (MV) is considered as a key element in the ME process. It is used to represent a similarity match between a block in current and reference frame [9]. In other words, it defines the distance covered by a block between current and reference frame. To find out the best match, there are many mathematical techniques i.e. Sum of Absolute Differences (SAD), Sum of Squared Differences (SSD), Sum of Hamming distances (SHD), Minimum Absolute Differences (MAD), Locally scaled Sum of Absolute Differences (LSAD), Locally scaled Sum of Squared Differences (LSSD), Zero-mean Sum of Absolute Differences (ZSAD), Zero-mean Sum of Squared Differences (ZSSD), Normalized Cross Correlation (NCC) and Zero-mean Normalized Cross Correlation (ZNCC). These techniques involve operations like addition, subtraction, multiplication and division. The time complexity for addition and subtraction is N while for multiplication and division, it is N^2 [38].

In our proposed approach, we try to avoid multiplications and divisions to reduce the computational complexity and time. Among the above mentioned techniques, the simplest similarity measure is MAD, which involves difference operations only. During the search process, the MAD is calculated for each pixel until the difference is below a threshold. If the difference is not below the threshold in a row, the search process will continue and move to next row in the SW. This process is repeated until all rows in the SW are searched. After that the minimum value is searched from a set of calculated values. Then X-Y coordinates of the pixels, in the current and reference frame respectively, are found out. These X-Y coordinates determine the head and tail locations of the corresponding MV. After that another search is performed to find out a MV that has the minimum value from the set of calculated MVs. The MV, having minimum value is selected as the representing MV of the current block. The above process is a Forward Motion Estimation (FME) that is to find the Forward Motion Vector (FMV) for each IB. The same process as that shown from subsections *A* to *D* is performed for a Backward Motion Estimation (BME) to find the Backward Motion Vector (BMV) for each IB by considering frame $N+1$ and $N-1$ as the current and reference frames respectively.

E. Frame Interpolation

The frame or motion interpolation is a process in which intermediate animated video frames are generated between the existing ones to provide a clear, fluid or smooth motion. This feature is very common in many latest HD displays and media players, supplied by different companies such as LG, Mitsubishi, Panasonic, Phillips, Samsung, Sony, Toshiba, Sharp and many others. The technology is known as TruMotion in LG [39], Smooth in Mitsubishi [40], Intelligent Frame Creation (IFC) in Panasonic [41], Perfect Motion Rate in Phillips [42], Auto Motion Plus in Samsung [43],

MotionFlow in Sony [44], ClearScan in Toshiba [45] and AquoMotion in Sharp [46]. Although these techniques lead to crystal and clear visual quality, they request complex hardware. Furthermore, from marketing point of view, the manufacturers never reveal the technical flow process of their techniques.

To perform a simple frame interpolation between a pair of frames F_{N-1} and F_{N+1}, at time t_{N-1} and t_{N+1}, frame F_i can be inserted as:

$$t_i = \frac{(t - t_{N-1})}{(t_{N+1} - t_{N-1})} \tag{1}$$

$$F_{i,x,y} = (1 - t_i)F_{N-1,x,y} + t_i F_{N+1,x,y} \tag{2}$$

Let $V_{x,y}$ be the MV between blocks in the current frame and a reference frame As $V_{x,y}$ defines the motion vector of pixels between the current and reference frames, the tail coordinates can be defined as $[x_b, y_b]$ and head coordinates as $[x_f, y_f]$. The values of $[x_b, y_b]$ and $[x_f, y_f]$ can be found as:

$$[x_b, y_b] = [x, y] - t_i V_{x,y} \tag{3}$$

$$[x_f, y_f] = [x, y] + (1 - t_i)V_{x,y} \tag{4}$$

where the point $[x, y]$ represents a point in the interpolated frame [47]. This approach works pixel by pixel at whole frame level and consumes lots of processing time.

In our proposed algorithm, we use the above technique but at block level. In our proposed approach, the MVs are estimated at block level rather than pixel level, so our approach saves the computational time and does not demand the process at every pixel in a block. After estimating the MV for each block of missing frame, the missing blocks are interpolated by modifying (2) to:

$$B_{i,x,y} = (1 - t_i)B_{N-1,x_b,y_b} + t_i B_{N+1,x_f,y_f} \tag{5}$$

where B_i is the block of missing frame, B_{N-1} is the block in current frame and B_{N+1} is the block in reference frame. The final concealed frame can be found by:

$$CF = \frac{F_{i,x,y}}{2} \tag{6}$$

The main reason behind adopting the simple frame interpolation is its mathematical simplicity, less computational complexity and time. However, it is observed in experiments that in some interpolated frames, there are thin lines at the edges of reconstructed blocks because of placement of pixels at false locations.

F. Adaptive Filtering

Sometimes it may happen that the concealed image contains blocky artifacts or thin hair-like lines. These types of distortions are very common in block based approaches. While smoothing down the image, it is very important to maintain the significant details without introducing blurring effects. In homogenous regions, the intensities of distorted pixels can easily be replaced by the average intensities of neighborhood pixels. Sometimes, the distortions occur at edges or boundaries of objects because averaging operations produce blurring and blocky artifacts in such cases. Therefore, the following two solutions are proposed for such cases. The solutions are either the neighborhood is selected adaptively or the filter is adaptive in operation. For first solution, the adaptive neighborhood can be formulized as:

$$\tilde{f}(x) = \frac{1}{M} \sum_{y \in N(x)} f(y) \tag{7}$$

where $f(y)$ represents an input image, $f(x)$ is the smoothed image, $N(x)$ is the adaptive neighborhood and M is the size of adaptive neighborhood. In (7), $N(x)$ consists of pixels in Ω, which are homogeneous and have intensities close to that of x, for each $x \in \Omega$. Therefore,

$$N(x) = \{y : R(x, y) \leq T, x, y \in \Omega\} \tag{8}$$

where R is represents the relationship between x and y, T is the real positive threshold and $\Omega \subset R^2$ [48]. The major problem in this approach is to find out the required neighborhood intelligently because if correct neighborhood is not selected, it will lead to false results. Another major problem is the computational complexity, as this approach involves division operations frequently.

For the second solution, the corrupted pixels need to be detected first before applying the adaptive filter. To declare a pixel as corrupted one, a difference of Δ_1 and $\Delta_{(1)}$ is used, where Δ_1 is the cumulated weighted distance allocated to the central pixel of filtering window and $\Delta_{(1)}$ is the output of weighted vector median filter. This difference shows the strength of impulsive noise. Let this difference is denoted be δ. The next step is to define a threshold value T_h for comparison purpose. The threshold value needs to be set carefully. If it is very large, it will pass out many corrupted pixels. In the case of very low value, uncorrupted pixels will also be declared as corrupted ones. The adaptive filter can be defined by:

$$AFO = \begin{cases} CF_{AMF}, & if\ \Delta_1 - \Delta_{(1)} > T_h \\ CF, & Otherwise \end{cases} \tag{9}$$

where AFO is the Adaptive Filter Output, CF is the concealed image pixel and CF_{AMF} is the output of Arithmetic Mean filter (AMF), which is computed from those pixels, declared as corrupted in the CF [49]. Although this approach is very simple and produces less computational complexity, it may produce false results, which are unavoidable at some times.

In our proposed algorithm, we choose and modify the adaptive filter approach. The main reason for choosing that approach is its less computational complexity. In thin hair-like lines, it is very obvious that there would be corrupted pixels in more than one row. Based upon that assumption, the filtering process may become speedy and produce less computational time if the filtering is performed in parallel. Another check is also performed for the detection of thin hair-like lines in homogeneous regions. If the hair-like line errors are in homogeneous regions, then intensity averaging is applied to save the computational time. In our proposed technique, the estimated frame is passed through the modified adaptive filter to remove line errors, if there are any. Another reason to use this modified filter is the presence of noise, which may or may

not be present in the estimated frame. In this case, the adaptive filter approach proves to be very quick compared to the adaptive neighborhood approach. The entire procedure of the FI algorithm is described in pseudo code in Algorithm 1.

Algorithm 1Frame Interpolation

Procedure: Conceal the Corrupted Input Video IV
for all frames in IV**do**
Read frame ***N-1***
Read frame ***N+1***
Compute forward motion estimation**FME**
Divide ***N-1*** into blocks **IBs**
Divide ***N+1*** into search windows **SWs**
Divide each **SW** into reference blocks **RBs**
whiletrue:
while true:
Match **IB** with $\mathbf{RB}_1$
Calculate the forward motion vector ***FMV***
$\boldsymbol{FMV}_1 \leftarrow \sum_{M=1}^{N} \boldsymbol{RB}_1$
end while
while true:
Match **IB** with $\mathbf{RB}_2$
Calculate the forward motion vector ***FMV***
$\boldsymbol{FMV}_2 \leftarrow \sum_{M=1}^{N} \boldsymbol{RB}_2$
end while
while true:
Match **IB** with $\mathbf{RB}_3$
Calculate the forward motion vector ***FMV***
$\boldsymbol{FMV}_3 \leftarrow \sum_{M=1}^{N} \boldsymbol{RB}_3$
end while
while true:
Match **IB** with $\mathbf{RB}_4$
Calculate the forward motion vector ***FMV***
$\boldsymbol{FMV}_4 \leftarrow \sum_{M=1}^{N} \boldsymbol{RB}_4$
end while
Assign best $\boldsymbol{FMV}_i \rightarrow$ **IB**
end while
Compute backward motion estimation **BME**
Divide ***N+1*** into **IBs**
Divide ***N-1*** into **SWs**
Divide each **SW** into **RBs**
whiletrue:
while true:
Match **IB**with$\mathbf{RB}_1$
Calculate the backward motion vector ***BMV***
$\boldsymbol{BMV}_1 \leftarrow \sum_{M=1}^{N} \boldsymbol{RB}_1$
end while
while true:
Match **IB** with $\mathbf{RB}_2$
Calculate the backward motion vector ***BMV***
$\boldsymbol{BMV}_2 \leftarrow \sum_{M=1}^{N} \boldsymbol{RB}_2$
end while
while true:
Match **IB** with $\mathbf{RB}_3$
Calculate the backward motion vector ***BMV***
$\boldsymbol{BMV}_3 \leftarrow \sum_{M=1}^{N} \boldsymbol{RB}_3$
end while
while true:
Match **IB** with $\mathbf{RB}_4$
Calculate the backward motion vector ***BMV***
$\boldsymbol{BMV}_4 \leftarrow \sum_{M=1}^{N} \boldsymbol{RB}_4$
end while
Assign best $\boldsymbol{BMV}_i \rightarrow$ **IB**
end while
Perform concealment based upon **FME**
Duplicate ***N-1*** as forward duplicate***FD***
whiletrue:
Calculate forward length**FL** of ***FMV***
Paste $\boldsymbol{N-1(IB)} \rightarrow \boldsymbol{FD}\left(\frac{\boldsymbol{FL}}{2}\right)$
end while
$\boldsymbol{FCF} = \boldsymbol{FD}$
Perform concealment based upon **BME**
Duplicate ***N+1*** as backward duplicate***BD***
whiletrue:
Calculate backward length**BL** of ***FMV***
Paste $\boldsymbol{N+1(IB)} \rightarrow \boldsymbol{BD}\left(\frac{\boldsymbol{BL}}{2}\right)$
end while
$\boldsymbol{BCF} = \boldsymbol{BD}$
Construct combinedconcealed frame ***CF***
$\boldsymbol{CF} \leftarrow \frac{(\boldsymbol{FD}+\boldsymbol{BD})}{2}$
Apply adaptive filter **AF**
whiletrue:
$$\boldsymbol{CF} = \begin{cases} \boldsymbol{CF}_{AMF}, & if\ \Delta_1 - \Delta_{(1)} > \boldsymbol{T_h} \\ \boldsymbol{CF}, & Otherwise \end{cases}$$
end while
end for
end procedure

IV. EXPERIMENTAL SETUP AND RESULTS

In order to evaluate the performance of the proposed technique, experiments are performed using HM 16.2 [50] and Matlab R2015a. The experiments are carried out using various video sequences of both High Definition (HD) and non-HD. Fifteen popular video sequences are used for simulation purposes. The details of experimental model and platform can be found in Table I.

TABLE I. Simulation environment

Hardware	CPU: Intel ® Core ™ i5-3470 CPU @ 3.20 GHz RAM: 8 GB
Software	H.265/HEVC Codec HM 16.2 Matlab R2015a
Video	Format: 4:2:0
QP	Varying between 10 to 37
PLR	1, 3, 5 %
Methods	Frame Copy, BMA and Proposed Method

The details of input video sequences, such as resolution, total number of frames, frame rate, Quantization Parameter (QP) and encoding bit rates are illustrated in Table II. All videos are encoded in unsliced and Intra only mode, and the first 150 frames are used from each sequence for experiments. The video sequences are encoded using different QPs ranging from 10 to 37 as shown in Table II. There are two main reasons to use different QPs. The first reason is to generate different bit rates for testing the efficiency of our proposed approach from reconstructed frame quality point of view. The second reason is to follow the same experimental setup, as used in [14], in order to test the efficiency of FI. To simulate the packet loss, we use H.265 RTP loss model proposed in [51], which is a well-known model and widely used in many research articles. In our experiments, three different Packet Loss Rates (PLRs), 1%, 3% and 5% are tested. We also implement the Frame Copy (FC) [15] and Block Matching Algorithm (BMA) [14] under the same experimental conditions, as proposed in [14, 15] for the comparison purpose.

Tables III (appendix) and IV (appendix) summarize the simulation results for the BMA, FC and the FI in terms of average Peak Signal to Noise Ratio (PSNR) with different PLRs for selected HD and non-HD video sequences. The Mean Square Error (MSE) metric is used to calculate the distortion effects. The PSNR in simulations is based on the MSE value for each estimated block. The video sequences, used in simulations have variety of motions in terms of speed, ranging from slow to high. The videos are categorized into three main categories, i.e. moving objects with static camera, static objects with moving camera and moving objects with moving camera. Tables V and VI describe the average computational time with different PLRs, both for BMA and FI for selected HD and non-HD videos respectively. As shown in Table III and IV, FI produces better performance as compared to FC and BMA in terms of average PSNR by approximately 12 dB and 2.5 dB respectively. In Table V and VI, FI leads BMA in average computational time, by approximately 1,788 seconds. The two main motives behind the development of FI are an acceptable/affordable average PSNR and low computational time. The low computational time is particularly focused because the EC algorithms execute at end-user side. In the case of mobile users, the end-users may have low processing devices such as smart phones, tablets and personal laptops. These end-user devices have slow processors with limited number of hardware sources such as small amount of Random Access Memory (RAM) and limited number of temporary storage buffers with slow speed data buses. In real-time streaming, delays are always time critical, so the low processing time and minimum hardware resources requirements for the EC applications are always recommended.

The visual comparison between the proposed and reference algorithms is presented in Fig. 3 (appendix) under different PLRs and QPs with different video sequences. In Fig. 3,

TABLE II. Input video sequences

Sequence	Resolution	Total Number of Frames	Frame Rate	QP	Bit Rate
Blue_sky	1920×1080	250	25	27, 32, 37	23982.949, 14977.315, 9261.401
BQTerrace	1920×1080	600	60	27, 32, 37	81257.312, 43334.454, 23746.781
Cactus	1920×1080	500	50	27, 32, 37	49908.523, 27077.061, 14652.869
Kimono	1920×1080	240	24	27, 32, 37	13262.299, 7864.826, 4628.346
Rush_hour	1920×1080	500	25	27, 32, 37	6219.905, 3745.957, 2291.689
Tractor	1920×1080	761	25	27, 32, 37	22311.997, 12958.396, 7528.085
BasketballDrillText	832×480	500	50	26, 29, 38	13437.965, 9443.944, 3429.003
BasketballPass	416×240	500	50	21, 23, 30	5563.536, 4537.888, 2126.491
BlowingBubbles	416×240	500	50	21, 23, 30	10858.829, 8930.136, 4176.013
Flowervase	416×240	301	30	10, 11, 17	4416.958, 4067.218, 2544.066
	832×480	300	30	19, 22, 27	7426.966, 5610.571, 3492.112
Keiba	416×240	300	30	13, 17, 24	6285.877, 4526.310, 2434.555
	832×480	301	30	33, 37, 42	2483.947, 1558.250, 827.222
RaceHorses	416×240	300	30	17, 21, 29	7516.494, 5418.634, 2358.395
	832×480	300	30	27, 29, 37	11593.811, 9347.834, 3148.022

column (a) represents the lost frame, (b) represents the output of FC algorithm, (c) represents the output of BMA and (d) represents the output of FI. These sample images are taken from three different videos, having different resolutions with motion of camera and/or objects. The first sample is taken from an HD video, Rush_hour with QP 32, the second one is a non-HD video, Flowervase (832×480) with QP 22, and the last one is from a non-HD video BlowingBubbles with QP 30. It can be seen very clearly that there is no bigger and notable visual difference among the original, FC and FI technique while BMA is showing lots of blurred lines. The only notable difference is present in BlowingBubbles samples, which is red encircled. As there is no significant motion between consecutive video frames, the visual difference cannot be noticed much in the FC algorithm. However, in the case of higher PLRs, the FC algorithm may not perform well and may produce frame-freezing effects.

Fig. 4 (appendix) shows the performance graphs of FC, BMA and FI in terms of average PSNR under different PLRs. Fig. 5 shows the performance graphs of BMA and FI in terms of average computational time under different PLRs. In Figs. 4 and 5, first, second and third rows represent the average PSNRs and computational time for the HD video Rush_hour with QP 32, the non-HD videos Flowervase (832×480) with QP 22 and BlowingBubbles with QP 30 respectively. The first, second and third columns represent 1%, 3% and 5% PLRs respectively. It can be seen very easily that the average PSNR performance of FI is better than FC and BMA. FI outperforms BMA in average computational time, which proves its suitability for real-time processing.

V. CONCLUSION

In this paper, we have proposed a fast and quality oriented frame interpolation considering parallel processing for mobile end-user devices, having low processing power and hardware resources. The importance of less computational time and acceptable quality has been raised after the recent developments in H.265/HEVC standard. The computational time has gained more importance, with the arrival of HD and Ultra HD media streaming with 4K and 8K resolutions. However, no matter if the media streaming is live or stored-file-based (i.e. multimedia cloud servers), network channels can never give guarantee for no packet loss.

We have studied FC and BMA for the implementation and comparison purposes over H.265 platform. The proposed algorithm provides better results in terms of both average PSNR and computational time. With the continuously updating threshold, the number of searching points has been decreased and the process to find the matching block has become fast.

In our study, we have used various types of videos, having different types of motion. The experimental results have proven that our proposed scheme is independent of the nature and contents of video sequence. The experimental results have proven that the proposed algorithm gives better performance in terms of visual quality compared to FC and BMA. It has also been proven that the computational time is far better than the classical BMA approach and can easily be applied at low processing end-user devices.

The simulation results produced in this paper will act as a base for the development of real-time EC algorithms for end user devices with error-prone transmission channels. Future developments may take this study further to modify HEVC decoder in such a way, so that it should incorporate efficient EC methods.

ACKNOWLEDGMENTS

Xiangjian He is the corresponding author.

REFERENCES

[1] Cisco Systems, "Cisco Visual Networking Index: Forecast and Methodology", 2014-2019, May 27, 2015.

[2] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing", Communications of the ACM, Vol. 53, No. 04, pp: 50-58, April 2010.

[3] L. Wang, J. Zhan, W. Shi, and Y. Liang, "In Cloud, Can Scientific Communities Benefit from the Economies of Scale?", IEEE Transactionson Parallel and Distributed Systems, Vol. 23, No. 02, pp: 296-303, February 2012.

[4] X. Zhang, L. Yang, C. Liu and J. Chen, "A Scalable Two-Phase Top-Down Specialization Approach for Data Anonymization Using MapReduce on Cloud", IEEE Transactions on Parallel and Distributed Systems, Vol. 25, No. 02, February 2014.

[5] T. Hobfeld, R. Schatz, M. Varela, and C. Timmerer, "Challenges of QoE Management for Cloud Applications", IEEE Communications Magazine, Vol. 50, No.4,pp: 28-36, April 2012.

[6] J. He, Y. Wen, J. Huang and D. Wu, "On the Cost-QoE Tradeoff for Cloud-Based Video Streaming Under Amazon EC2's Pricing Models", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 24, No. 04, April 2014.

[7] A. Alasaad, K. Shafiee, H. Behairy and V. Leung, "Innovative Schemes for Resource Allocation in the Cloud for Media Streaming Applications", IEEE Transactions on Parallel and Distributed Systems, Vol. 26, No. 04, April 2015.

[8] J. He, D. Wu, Y. Zeng, X. Hei and Y. Wen, "Towards Optimal Deployment of Cloud-Assisted Video Distribution Services", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 23, No. 10, October 2013.

[9] G. Sullivan, J. Ohm, W. Han and T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 22, No. 12, December 2012.

[10] G. Carle and E. W. Biersack, "Survey of Error Recovery Techniques for IP-based Audio-Visual Multicast Applications", IEEE Network, Vol. 11, No. 06, pp: 24-36, November 1997.

[11] W. Kung, C. Kim and C. Kuo, "Spatial and Temporal Error Concealment Techniques for Video Transmission Over Noisy Channels", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 16, No. 7, July 2006.

[12] X. Liu, W. Yang and Z. Shen, "H.264/AVC Video Error Concealment Algorithm by Employing Motion Vector Recovery Under Cloud Computing Environment", The Journal of Supercomputing, Vol. 70, No. 03, pp: 1180-1199, December 2014.

[13] Y. Lin, E. ChuY. Lai and T. Huang, "Time-and-Energy-Aware Computation Offloading in Handheld Devices to Coprocessors and Clouds", IEEE Systems Journal, Vol. 09, No. 02, June 2015.

[14] J. Zhou, B. Yan and H. Gharavi, "Efficient Motion Vector Extrapolation for Error Concealment of H.264/AVC", IEEE Transactions on Broadcasting, Vol. 57, No. 01, March 2011.

[15] J. Nightingale, Q. Wang, C. Grecose and S. Goma, "The Impact of Network Impairment on Quality of Experience (QoE) in H.265/HEVC Video Streaming", IEEE Transactions on Consumer Electronics, Vol. 60, No. 02, May 2014.

[16] M. Usman, X. He, M. Xu and K. Lam, "Survey of Error Concealment Techniques: Research Directions and Open Issues", IEEE Picture Coding Symposium, pp: 233-238, June 2015.

[17] W. Kumwilaisak and C. Kuo, "Spatial Error Concealment with Sequence-Aligned Texture Modelling and Adaptive Directional Recovery", Journal of Visual Communication and Image Representation, Vol. 22, pp: 164-177, December 2010.

[18] J. Koloda, A. Peinado and V. Sanchez, "Kernel-Based MMSE Multimedia Signal Reconstruction and Its Application to Spatial ErrorConcealment", IEEE Transactions on Multimedia, Vol. 16, No. 06, October 2014.

[19] H. Asheri, H. Rabiee, N. Pourdamghani and M. Ghanbari, "Multi-Directional Spatial Error Concealment using Adaptive Edge Thresholding", IEEE Transactions on Consumer Electronics, Vol. 58, No. 03, August 2012.

[20] M. Hoque, M. Pimentel, M. Hasan, K. Ahn and J. Kim, "Edge-Based Spatial Concealment of Digital Dropout Error in Degraded Archived Media", IET Electronics Letters, Vol. 50, No. 14, pp: 996-997, July 2014.

[21] L. Zhu, Y. Zhao, S. Wang and H. Chen, "Spatial Error Concealment for Stereoscopic Video Coding Based on Pixel Matching", The Journal of Supercomputing, Vol. 58, No. 01, pp: 96-105, October 2011.

[22] S. Yang, C. Chang and C. Chan, "An Object Based Error Concealment Technique for H.264 Coded Video", Multimedia Tools and Applications, July 2014.

[23] W. Lie, C. Lee, C. Yeh and Z. Gao, "Motion Vector Recovery for Video Error Concealment by Using Iterative Dynamic Programming Optimization", IEEE Transactions on Multimedia, Vol. 16, No. 01, January 2014.

[24] Y. Zhang, X. Xiang, D. Zhao and W. Gao, "Packet Video Error Concealment with Auto Regressive Model", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 22, No. 01, January 2012.

[25] B. Yan and J. Zhou, "Efficient Frame Concealment for Depth Image-Based 3-D Video Transmission", IEEE Transactions on Multimedia, Vol. 14, No. 03, June 2012.

[26] M. Yang, X. Lan, N. Zhang and P. Cosman, "Depth-Assisted Temporal Error Concealment for Intra Frame Slices in 3-D Video", IEEE Transactions on Broadcasting, Vol. 60, No. 02, June 2014.

[27] Y. Zhou, W. Xiang and G. Wang, "Frame Loss Concealment for Multi-View Video Transmission Over Wireless Multimedia Sensor Networks", IEEE Sensors Journal, Vol. 15, No. 03, March 2015.

[28] W.Tsai and J. Chen, "Joint Temporal and Spatial Error Concealment for Multiple Description Video Coding", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 20, No. 12, December 2010.

[29] J. Zhu and R. Dansereau, "Error-Resilient and Error Concealment 3-D SPIHT for Multiple Description Video Coding with Added Redundancy", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 22, No. 06, June 2012.

[30] H. Hadizadeh, I. Bajic and G. Cheung, "Video Error Concealment Usinga Computation Efficient Low Saliency Prior", IEEE Transactions on Multimedia, Vol. 15, No. 08, December 2013.

[31] W. Choi, B. Jeon and J. Jeong, "Fast Motion Estimation With Modified Diamond Search for Variable Motion Block Sizes", IEEE International Conference on Image Processing, Vol. 03, pp: 371-374, September 2003.

[32] https://en.wikipedia.org/wiki/Block-matching_algorithm#References

[33] Z. Pan, Y. Zhang and S. Kwong, "Efficient Motion and Disparity Estimation Optimization for Low Complexity Multiview Video Coding", IEEE Transactions on Broadcasting, Vol. 61, No. 02, June 2015.

[34] S. Goel, Y. Ismail and M. Bayoumi, "Adaptive Search Window Size Algorithm for Fast Motion Estimation in H.264/AVC Standard", 48thIEEE Midwest Symposium on Circuits and Systems, Vol. 02, pp: 1557-1560, August 2005.

[35] Y. Ismail, J. McNeely, M. Shaaban, H. Mahmoud and M. Bayoumi, "Fast Motion Estimation System Using Dynamic Models for H.264/AVC Video Coding", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 22, No. 01, January 2012.

[36] Y. Nie and K. Ma, "Adaptive Rood Pattern Search for Fast Block-Matching Motion Estimation", IEEE Transactions on Image Processing, Vol. 11, No. 12, December 2002.

[37] S. Metkar and S. Talbar, "Motion Estimation Techniques for Digital Video Coding", Springer briefs in Applied Sciences and Technology, 2013.

[38] R. Brent and P. Zimmermann, "Modern Computer Arithmetic", Cambridge University Press, 2010.

[39] http://www.lg.com/us/tv-audio-video/discoverlgtvs/picture_quality/index.jsp

[40] http://tv.toptenreviews.com/projection/dlp/mitsubishi/wd-65737-review.html

[41] http://www.panasonic.com/au/consumer/televisions-projectors-learn/engineer-interview/4k-intelligent-frame-creation.html

[42] http://www.philips.com.au/c-p/32HFL5573D_10/5000-series-32-inch-mediasuite-lcd-dvb-t2-t-c-mpeg-2-4

[43] http://www.samsung.com/au/consumer/tv-audio-video/television/

[44] https://www.sony.com.au/article/287566/section/product/product/kdl-46z4500

[45] http://www.toshiba.com/us/clearscan

[46] http://www.sharp.net.au/product-catalogue/products/LC70LE960X/

[47] http://monochrome.sutic.nu/2010/10/12/motion-interpolation.html

[48] Y. Zhang, W. Liu, I. Magnin and Y. Zhu, "Feature-Preserving Smoothing of Diffusion Weighted Images Using Nonstationarity Adaptive Filtering", IEEE Transactions on Biomedical Engineering, Vol. 60, No. 06, June 2013.

[49] B. Smolka, K. Malik and D. Malik, "Adaptive Rank Weighted Switching Filter for Impulsive Noise Removal in Color Images", Journal of Real-Time Image Processing, Vol. 10, No. 02, pp: 289-311, June 2015.
[50] https://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-16.2/
[51] S. Wenger, "NAL Unit Loss Software", JCT-VC Document, JCTVH0072, February 2012.

APPENDIX

TABLE III. Average PSNR for HD video sequences

Sequence	Algorithm	1% (PLR)			3% (PLR)			5% (PLR)		
		QP 27	QP 32	QP 37	QP 27	QP 32	QP 37	QP 27	QP 32	QP 37
BQTerrace	FC	25.2432	25.2179	25.3932	26.244775	26.69735	25.69185	26.91565	26.18165	26.6829
	BMA	29.4008	29.3551	29.0321	29.070575	28.4839	29.480625	28.0274	26.900088	28.317725
	Proposed	34.7003	34.2211	32.951	34.22555	33.42165	33.043225	32.858675	31.291813	32.107325
Cactus	FC	23.3833	23.6229	22.1943	21.88065	21.7775	21.017225	21.311938	20.702463	20.390463
	BMA	27.8169	27.6374	27.2405	26.89515	26.85805	26.97515	26.431775	26.135513	26.561313
	Proposed	30.2461	30.4243	30.3357	29.43545	29.441	29.616575	28.9046	28.6516	29.273163
Rush_hour	FC	21.3486	21.2988	19.9627	21.26185	21.073275	20.5269	22.092388	21.71125	20.785263
	BMA	29.1569	29.1554	27.4403	28.085825	29.27325	28.012875	28.426363	27.748975	28.1525
	Proposed	32.9946	32.9813	32.1557	32.070125	33.32635	32.542275	32.7078	31.651038	32.494275

TABLE IV. Average PSNR for non-HD video sequences

Sequence	Algorithm	1%			3%			5%		
		QP 21	QP 23	QP 30	QP 21	QP 23	QP 30	QP 21	QP 23	QP 30
BlowingBubbles	FC	26.6671	26.5818	25.4311	22.724525	22.135375	22.031975	22.0715	21.764688	20.8691
	BMA	29.2497	29.3069	29.3204	28.037075	27.913075	28.262125	26.258438	26.661538	25.49985
	Proposed	32.8957	32.8804	32.5554	31.75895	31.68	31.721725	29.12265	29.94855	28.298375
Flowervase (832×480)	FC	QP 19	QP 22	QP 27	QP 19	QP 22	QP 27	QP 19	QP 22	QP 27
		34.0446	33.2339	32.1672	33.6769	30.22825	29.103975	29.626575	28.188825	28.6421
	BMA	40.9063	40.7886	40.2935	41.5893	41.422175	39.75925	40.0074	39.830163	39.8129
	Proposed	43.6805	43.4677	42.6814	44.484875	44.7111	43.29595	44.163163	43.8247	42.906225
RaceHorses (832×480)	FC	QP 27	QP 29	QP 37	QP 27	QP 29	QP 37	QP 27	QP 29	QP 37
		15.6651	15.6167	15.657	14.8043	14.83665	14.90525	14.780538	14.863875	15.143825
	BMA	17.8541	17.8916	18.2004	17.4203	17.70035	17.747075	17.313088	17.378375	17.877313
	Proposed	20.0254	20.0742	20.41	19.58105	19.853775	19.889775	19.471438	19.60395	20.150625

TABLE V. Average computational time of HD video sequences

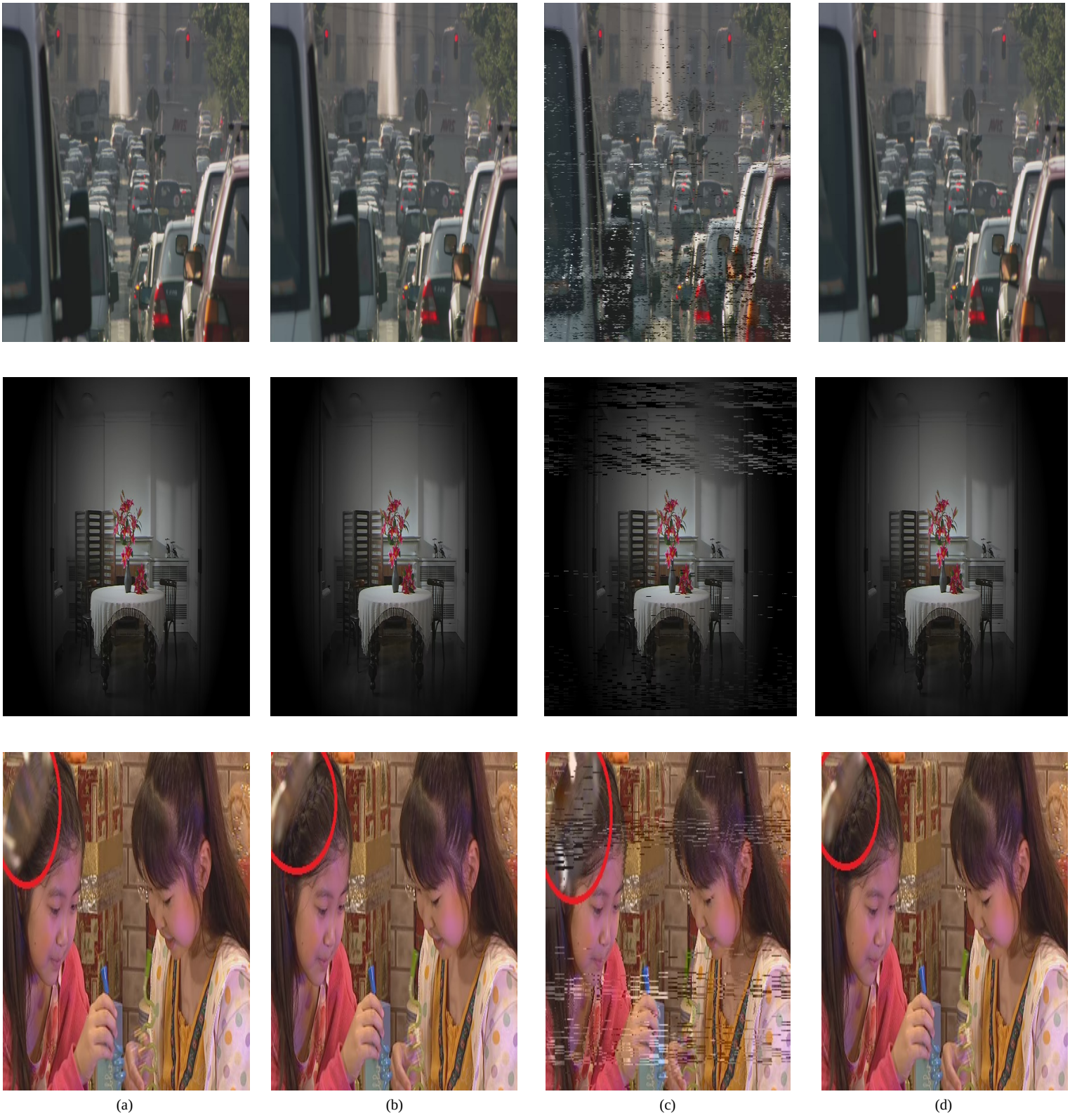
Sequence	Algorithm	1% (PLR)			3% (PLR)			5% (PLR)		
		QP 27	QP 32	QP 37	QP 27	QP 32	QP 37	QP 27	QP 32	QP 37
BQTerrace	BMA	1748.53045	1734.26683	1744.72494	1856.03563	1817.64689	1806.66057	3383.6576	3415.26973	3364.38238
	Proposed	12.477895	12.49347	12.589352	12.5099425	12.5323258	12.7712035	12.5826791	12.6544354	12.7165948
Cactus	BMA	1812.3262	1815.75855	1813.47535	1865.3222	1826.64524	1841.92017	3657.9894	3474.168	3522.71973
	Proposed	12.6025201	12.715382	12.738267	12.65332	12.7874503	12.8112125	12.6038569	12.8153914	12.7828589
Rush_hour	BMA	1455.26441	1451.42148	1444.10877	1456.64139	1460.31246	1477.69298	2757.98097	2776.21874	2714.77782
	Proposed	12.676578	14.848038	13.888899	12.6868988	14.7329763	14.3541953	12.2904226	14.8744468	14.4716305

TABLE VI. Average computational time of non-HD video sequences

Sequence	Algorithm	1% (PLR)			3% (PLR)			5% (PLR)		
BlowingBubbles	BMA	QP 21	QP 23	QP 30	QP 21	QP 23	QP 30	QP 21	QP 23	QP 30
		79.921818	63.481073	102.890182	71.508576	71.1960463	71.3787865	141.588345	156.473912	161.44053
	Proposed	1.508997	1.516878	1.518721	1.51949075	1.50877525	1.5139575	2.0109785	2.01243133	2.01781233
Flowervase (832×480)	BMA	QP 19	QP 22	QP 27	QP 19	QP 22	QP 27	QP 19	QP 22	QP 27
		398.591966	448.765788	420.146287	508.33831	526.452203	520.880732	1321.21549	1305.40621	1308.0575
	Proposed	3.132107	3.15418	3.131794	3.13594075	3.1475675	3.1352845	4.19657217	4.19241667	4.19702067
RaceHorses (832×480)	BMA	QP 27	QP 29	QP 37	QP 27	QP 29	QP 37	QP 27	QP 29	QP 37
		379.176938	488.472659	296.197198	508.487726	471.103954	476.644417	1333.08302	1221.59437	1161.13726
	Proposed	3.119176	3.149487	3.148898	3.1265075	3.13520525	3.13109775	4.17790533	4.17792767	4.16943

Fig. 3. Visual comparisons

Original FC BMA Proposed



(a) (b) (c) (d)

Fig. 4. PSNR comparisons

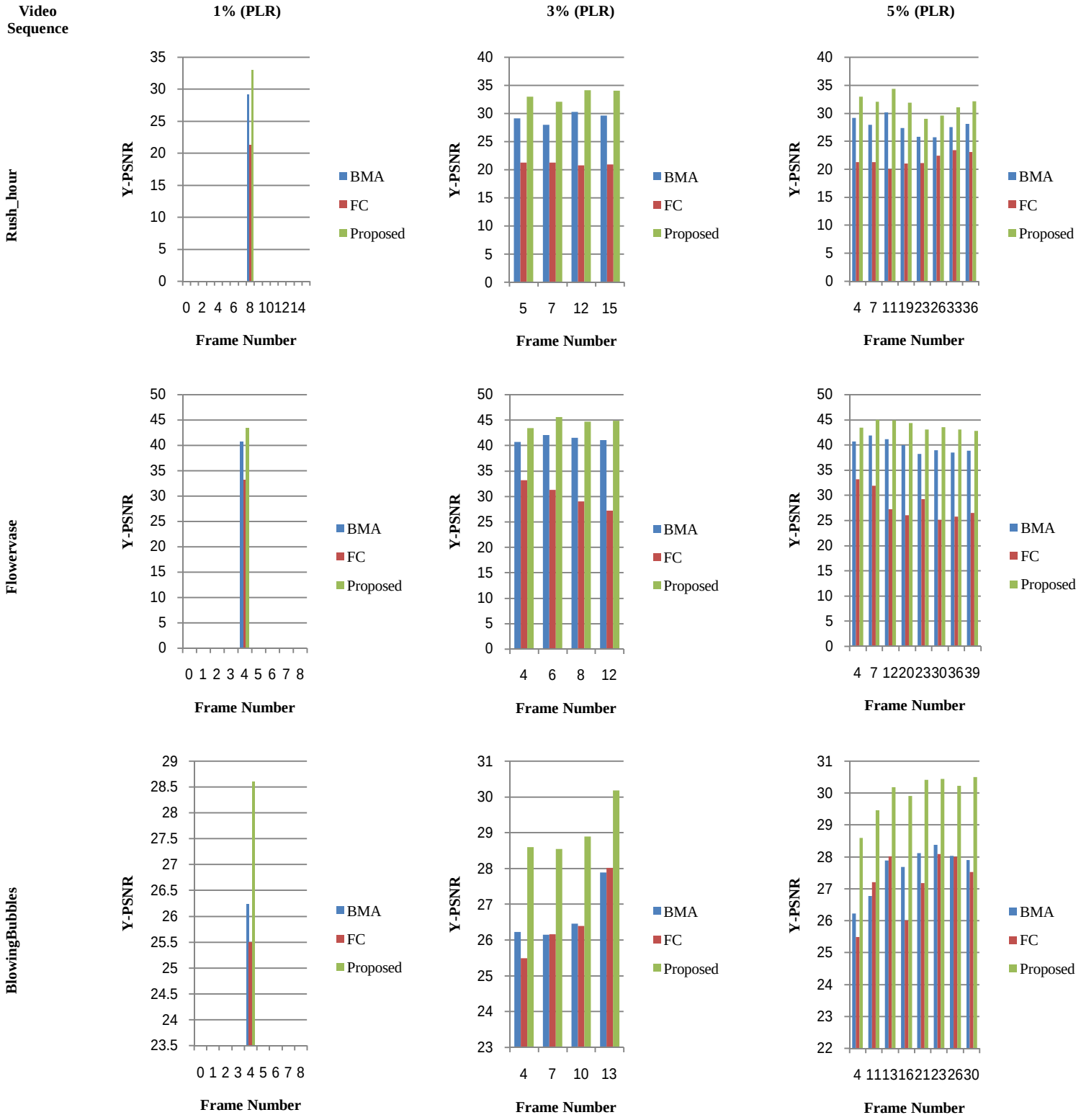
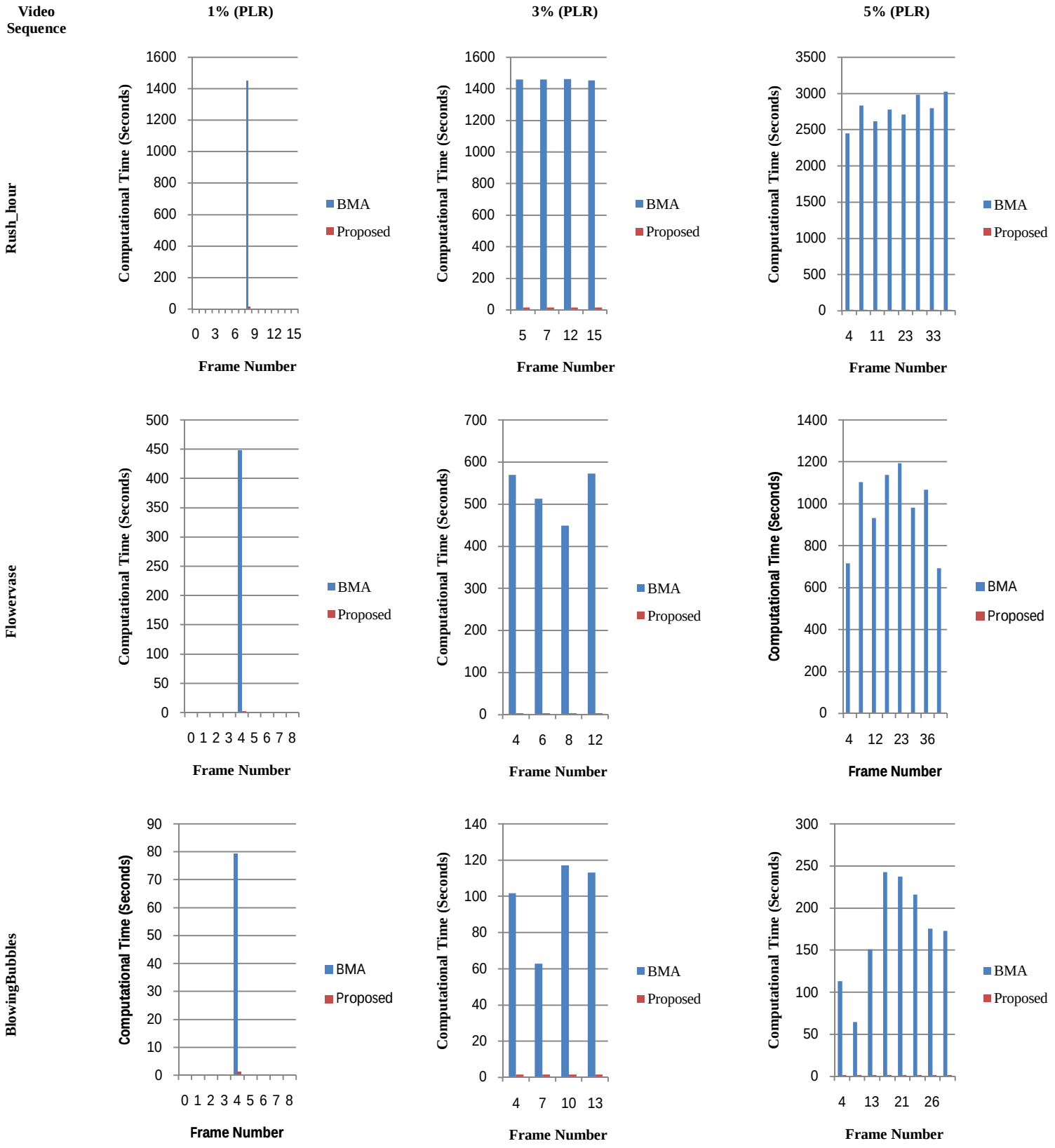
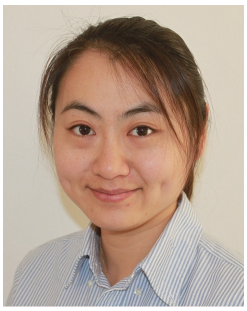


Fig. 5. Computational time comparisons



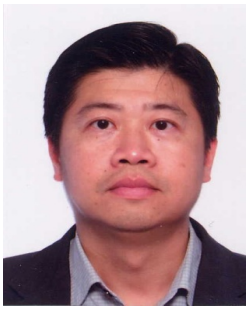
Muhammad Usman received the B.Sc degree in computer software engineering from University of Engineering and Technology (UET) Peshawar, Pakistan, in 2008 and the M.Sc degree in computer systems engineering from University of Engineering and Technology (UET), Taxila, Pakistan in 2012. Currently, he is pursuing the Ph.D. degree from the School of Computing and Communications, University of Technology Sydney, Australia. He was a lecturer in the department of electrical and computer engineering at Kohat University of Science and Technology (KUST), Kohat, Pakistan from 2009 to 2012 and from 2012 to 2014, he was a lecturer in the department of electrical engineering at University of Engineering and Technology (UET) Peshawar, Pakistan. His current research interests include audio, image and video processing, compression, communication, fast video encoding algorithms and error concealment techniques.

Xiangjian He received a B.S. in mathematics from Xiamen University in 1982, an MS in applied mathematics from Fuzhou University in 1986, and a Ph.D. in computing sciences from the University of Technology, Sydney, Australia, in 1999. From 1982 to 1985, he was with Fuzhou University. From 1991 to 1996, he was with the University of New England. Since 1999, he has been with the University of Technology, Sydney. He is currently a full professor and the director of Computer Vision and Pattern Recognition Laboratory at the University of Technology, Sydney.



Min Xu received the B.E. degree from University of Science and Technology of China, in 2000, M.S degree from National University of Singapore in 2004 and Ph.D degree from University of Newcastle, Australia in 2010. She is currently a Senior Lecturer at University of Technology, Sydney, Australia. Her research interests include multimedia content analysis, multimedia affective computing, social multimedia, pattern recognition and computer vision.

Syed Mohsin Matloob Bokhari received the B.E. degree in Computer Systems Engineering from the National University of Sciences and Technology (NUST), Rawalpindi, Pakistan, in 2004, and the M.Sc. degree in Computer Engineering from the Lahore University of Management Sciences (LUMS), Lahore, Pakistan, in 2006. He received his Ph.D. degree in Electrical and Electronics Engineering from the University of Bristol, Bristol, U.K. in 2012.From 2006 to 2008 he worked as a Senior Software Engineer in Streaming Networks (Pvt). Ltd., Islamabad, Pakistan. Since 2012, he is working as an Assistant Professor in Department of Electronics Engineering in the University of Engineering & Technology, Peshawar, Pakistan. His research interests include image and video processing, compression, and communication.



Kin-Man Lam received the Associateship in electronic engineering with distinction from The Hong Kong Polytechnic University (formerly called Hong Kong Polytechnic) in 1986, the M.Sc. degree in communication engineering from the Department of Electrical Engineering, Imperial College of Science, Technology and Medicine, London, UK, in 1987, and the Ph.D. degree from the Department of Electrical Engineering, University of Sydney, Sydney, Australia, in August 1996. From 1990 to 1993, he was a lecturer at the Department of Electronic Engineering, The Hong Kong Polytechnic University. He joined the Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, as an assistant professor in October 1996, became an associate professor in 1999, and has been a professor since 2010. He has been a member of the organizing committee and program committee of many international conferences. Currently, he is the Vice President of Member Relations and Development of the Asia-Pacific Signal and Information Processing Association (APSIPA) and the Director of Membership Services of the IEEE Signal Processing Society. He is an Area Editor of IEEE Signal Processing Magazine and an Associate Editor of Digital Signal Processing, APSIPA Transactions on Signal and Information Processing, and EURASIP International Journal on Image and Video Processing. He is also an Editor of HKIE Transactions. His current research interests include human face recognition, image and video processing, and computer vision.

Jinjun Chen is an Associate Professor from Faculty of Engineering and IT, University of Technology Sydney, Australia. He holds a PhD in Computer Science and Software Engineering (2007) from Swinburne, a master degree in engineering (1999) and a bachelor degree in applied mathematics (1996) from Xidian University, China. His research interests include cloud computing, social computing, green computing, service computing, e-science, workflow management. He has published more than 100 papers in high quality journals and conferences, including TOSEM, TSE and ICSE. He received IEEE Computer Society Outstanding Leadership Awards (2008-2009, 2010-2011), Vice Chancellor Research Award, and many other awards.