IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)
e-ISSN: 2278-2834,p- ISSN: 2278-8735.Volume 9, Issue 6, Ver. III (Nov - Dec. 2014), PP 105-119
www.iosrjournals.org

Evaluation of H.264/AVC Coding Elements and New Improved Scalability/Adaptation Algorithm/Methods

Sanusi Muhammad (Dr), *Student Member, IEEE,* Abdul Hamid Sadka(Prof), *Senior Member, IEEE*
Department of Electronic& Computer Engineering., Brunel University Uxbridge, UK.

Abstract: *In this paper, we evaluated the major coding elements/parameters of H.264/AVC and their significant differences with the prior standards. The design parameters of H.264/AVC codec are experimented with and analysed. The influences of the parameters and other coding elements on scalable bitstream and their level of adaptation over multi-channel networks are also evaluated. Objective and real-time analysis are employed in the evaluation. To enhance the flexibility and robustness of codec adaptation over heterogeneous networks, Qp Constraint Algorithm is developed that reveals bits reduction up to 100kbits depending on the current bitrates. The usage of slice groups is implemented and the result shows that, the implementation of slice groups can support flexible adaptation and data protection. We also propose and experimentally evaluate methods for multi-channels adaptation.*

***Key-Words*:** *Coding Elements, H.264/AVC, Scalability, Adaptation, Evaluation*

I. Introduction

H.264/AVC is the most recent video coding standard [1]. Advances in video technology along with rapid developments of new devices, network infrastructure, processing power and storage capacity are enabling an increasing number of video applications. Hence, the needs for more improvements to further achieve better flexible scalability and adaptation.

Video coding for telecommunications applications has evolved through the development of the ITU-T H.261, MPEG-2, H.262 and H.263 video coding standards. H.263+ evolved later as an enhancement to H.263 and then H.263++. In early 1998, the Video Coding Experts Group (VCEG-ITU-T SG16 Q.6) issued a call for proposals on a project call H.26L, with the aim of doubling the coding efficiency with a given bitrate in comparison with other standards for a wide range of applications. The first draft design was adopted in October 1999. In December of 2001, VCEG and the Moving Picture Expert Group (MPEG – ISO/IEC JC I/SC 29/WG II) formed a Joined Video Team (JVT) with the charter to conclude the new video coding design draft for formal approval submission as *H.264/AVC* [1] in March 2003. In the design of the *H.264/AVC*, various coding parameters were improved or introduced to enhance the efficiency and scalability gain over the prior coding standards. In this work, we evaluated the achieved objective, subjective and real-time performances of the improved and the introduced parameters of *H.264/AVC*. This established knowledge will enable the adaptation of a specific set of coding parameters for a particular performance impact on scalability and adaptation. New methods and adaptation algorithms are also proposed.

H.265/HEVC (High Efficiency Video Coding) is now the most recent standard to be released in July 2013, an upgrade to H.264. It is designed for the next generation networks/devices and applications ranging from high quality video streaming on mobile devices to ultra-high resolution displays. HEVC can reduce bandwidth requirement and bitstream size by 50% providing equivalent or enhanced quality over the current *H.264/AVC* [2].

1.1. Methodology

The new and introduced coding parameters are evaluated. The evaluation is based on experiments to determine their objective and subjective quality performance. Real time performances are also conducted over a heterogeneous simulator setup as in Figure 1. A coded parameter bitstream output is passed into the network simulator through *SITL* (System in the Loop) which is a standard tool [3] that receives external packets and translates them into Opnet simulation packets. Results are collected at the end of the simulations. Apart from section one which is the introduction and methodology, the remaining parts are organised as follows: section two reviewed and evaluated *H.264/AVC* design parameters/elements and their comparisons with the prior standards, section three presents proposed adaptation methodologies and section four describes the introduced adaptation techniques/simulation results. In section five and six, acknowledgement and conclusion are presented followed by the references.

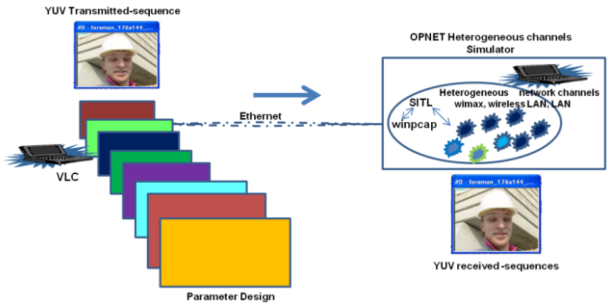


Figure 1 : Simulations Setup over Heterogeneous Networks

II. H.264/AVC Standard Design Elements and Parameters

In an attempt to address the need for flexibility and customisability, the *H.264/AVC* design covers a Video Coding Layer (*VCL*) and Network Abstraction Layer (*NAL*). *VCL* represents the video content and the *NAL* formats the *VCL* representation of video and provides header information in a style suitable for a variety of transport layers or storage media. The design and inclusion of *NAL* in *H.264/AVC* has supported a variety of mappings to the system. This has not been achieved prior to the inclusion of the *NAL* units. Some key features and concepts of the *NAL* are byte stream, packet format uses of *NAL* units, *NAL* units, access units and parameter sets described in this section.

Some codec design (e.g., H.320 and MPEG-2 codec) requires the delivery of the entire or partial *NAL* unit stream as an ordered stream of bytes or bits within which the locations of *NAL* units boundaries require to be identifiable from patterns within the coded data itself.

According to the byte stream format in *H.264/AVC*, each *NAL* unit is prefixed by a specific pattern of three bytes called a *start code* prefix. The boundaries of the *NAL* unit can then be identified by searching the coded data for the unique *start code* prefix pattern. The use of emulation prevention bytes guarantees that start code prefixes are unique identifiers of the start of a new *NAL* unit.

The *VCL* design follows a block-based hybrid coding approach as in prior video standards since H.261 [1, 4]. In block-based video coding, each coded picture is represented in block-shaped units of associated luma and chroma samples called macroblocks (*MBs*). The main source-coding algorithm is a hybrid inter-picture to exploit and reduce temporal statistical dependencies and transform coding of the prediction residual to exploit and minimise spatial statistical dependencies. There is no single video coding component in the *VCL* that provides the majority of the significant enhancement in compression efficiency in relation to prior standards. It is rather a plurality of smaller enhancements that sum up to the significant achieved gain. Figure 4 illustrates the main elements for a macroblock coding process in the *H.264/AVC* standard.
In the design of *H.264/AVC*, a parameter set contains information that is expected to rarely change and offers the decoding of a large number of *VCL NAL* units. Two types of parameter sets are designed [1, 4]:

- Sequence parameter sets which apply to a series of consecutive coded video pictures called a coded video sequence.
- Picture parameter sets which apply to the decoding of one or more individual pictures within a coded video sequence.

The sequence and picture parameter set mechanism decouples the transmission of infrequently changing information from the transmission of coded representations of the values of the samples in the video pictures. Every *VCL NAL* unit contains an identifier that refers to the content of the relevant picture parameter set, and each picture parameter set contains an identifier that refers to the content of the relevant sequence parameter set. In this approach, a small amount of data (the identifier) can be used to refer to a larger amount of information (the parameter set) without repeating the information within each *VCL NAL* unit.
Sequence and parameter sets can be sent well ahead of the *VCL NAL* units that apply to, and can be repeated to provide robustness against data loss. In some applications, parameter sets may be sent within the channel that carries the *VCL NAL* units (in-band transmission). In some other applications, it can be better to transport the parameter sets “out-of-band” using a more reliable transport mechanism than the video channel itself (Figure 2).

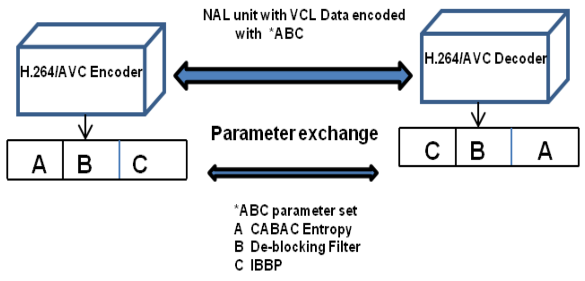


Figure 2: Out of band parameter set exchange using reliable channels

H.264/AVC Extension has obtained a large improvement in coding efficiency with an enhanced degree of supported scalability in comparison to the scalable profiles of prior video codec standards. Robustness to data errors and flexibility of transmission over heterogeneous network environments is also supported for several design aspects of *H.264/AVC* Extension. Some of the other major improvements in the *H.264/AVC* Standard include the following highlighted features [1] in section 2-2.8.

2.2. Entropy Coding

An advanced entropy coding technique known as arithmetic coding is included in *H.264/AVC*. Arithmetic coding was an optional feature of H.263, a more effective use of this technique is included in *H.264/AVC* to create a very powerful entropy coding algorithm known as Context Adaptive Binary Arithmetic Coding (*CABAC*). The other optional entropy coding used in *H.264/AVC* is Context Adaptive Variable Length Coding (*CAVLC*). Both *CABAC* and *CAVLC* use context adaptation to enhance performance relative to prior standards. Section 4.4 presents the simulations results and analysis for *CABAC*/*CAVLC* algorithms performances.

2.3. Deblocking Filtering

Blocking artifacts are generated from the block-based video coding. These are initiated from the prediction and the residual difference coding stages of the decoding procedure. To improve the resulting picture quality, deblocking filtering technique is used. The objective and subjective quality can be improved from a well designed de-blocking filter. The de-blocking filter is built further than that of H.263+, as the *H.264/AVC* filter design is brought within the motion-compensated prediction loop. This design enhancement can be used in inter-picture prediction to provide the ability to predict other pictures.

2.4. SP/SI Switching Pictures

In the *H.264/AVC* design, a new feature consisting of picture types that support the precise synchronisation of the decoding process of some decoders without penalising all decoders with the loss of efficiency resulting from sending I-pictures. This can enable switching a decoder among representations of the video content that used dissimilar data rates, recovery from data losses or errors as well as enabling trick modes such as fast-forward and fast-reverse etc. Figure 7 (right) shows switching among two scalable techniques t+s+q/t+s. [7] can be referred for details on t+s+q/t+s scalability techniques.

2.5. Macroblocks, Slices and Slice Groups

Macroblocks are the basic building blocks for which the reconstruction procedure is specified. A picture is partitioned into fixed-size *MBs* that each covers a rectangular video area of 16x16 samples of the luma component and 8x8 samples of each of the two chroma components. This design of partitioning into *MBs* has been used into all previous video coding standards (since H.263) of the ITU-T and ISO/IEC JTC1 [1, 5].

A Slice is a sequence of *MBs* which are processed in the order of a raster scan when not using flexible macroblock ordering (FMO) discussed in the next paragraph. A video picture may be split into one or several slices as illustrated in Figure 3. Hence, a picture in *H.264/AVC* is a collection of one or more slices. Slices are self-contained in the sense that given the active sequence slices can be rightly decoded without the use of data from other slices provided that utilised reference pictures are identical at encoder and decoder. When applying the deblocking filter across the slice boundaries, some information from other slices may be required and this depends on the deblocking filter type [1].

Slice -0
Slice -1
Slice -N

Figure 3 : Slices as subdivision of a picture when FMO is not used.

FMO restyled the approach where pictures are partitioned into slices and *MBs* by employing the concept of slice groups. Every slice group is a set of *MBs* defined by a *MB to slice group map*, which is specified by the content of the picture parameter set and header information of the slices. The *MB* to slice group map contains a slice group identification number for each *MB* in the picture, identifying which slice group the related *MB* belongs to. Each of the slice group can be partitioned into one or more slices, such that a slice is a sequence of *MBs* within the same slice group that is processed in the order of a raster scan within the set of *MBs* of a particular slice group.

Several techniques are adopted using FMO. In the techniques, a picture can be split into several *MBs* scanning patterns such as interleaved slices, a dispersed *MB* allocation, a box out slice pattern allocation, one or more foreground slice groups or left-over slice group pattern. Details about these techniques can be found in [4, 8]. The techniques are utilised for different applications gain, for instance the right-hand side *MB* to slice group pattern has been demonstrated for error concealment in video conferencing applications where slice group-1 and slice group-2 are transmitted in separate packets and one is lost. This is also demonstrated to support flexible scalability where different layers are coded with different FMO and slice group mapping as in (section 4.8) simulation results.

2.6. Macroblocks Encoding and Decoding Process

As in the prior standards, *H.264/AVC* follows the same coding process except for the elements described in this paper. All luma and chroma samples of a slice are either spatially or temporally predicted or both predictions applied.

The resulting prediction residual is encoded with transform coding. In *H.264/AVC*, each component of the prediction residual signal is subdivided into smaller 4x4 blocks [1]. An integer transform is employed to transform each block and the coefficients are then quantised and encoded using one of the entropy methods in section 2. Figure 4 illustrates the block diagram of the *VCL* of a macroblock. The input video signal is split into macroblocks, the relation of macroblocks to slice groups and slices is selected and then each macroblock from each slice is processed as illustrated in the diagram. This design allows for an efficient parallel processing of macroblocks when there are several slices in the video sequence.

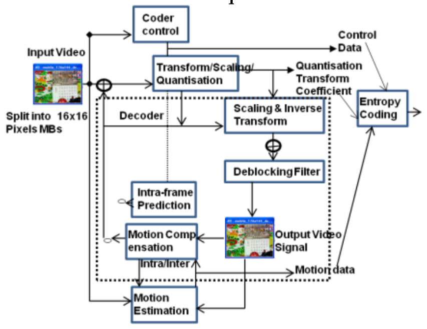


Figure 4: *H.264/AVC* macroblock basic coding structure

2.7. Intra-frame Prediction

Depending on the slice-coding type, a macroblock can be transmitted in one of the available slice-coding methods. The slice-coding types of intra-coding supported are denoted as intra_4x4 or intra_16x16 together with chroma prediction and *I_PCM* prediction modes.

The intra_4x4 mode is based on predicting each 4x4 luma block separately and is well appropriate for coding of regions of a picture with significant detail. The intra_16x16 mode on the other hand does prediction of the whole 16x16 luma block and is more appropriate for coding extremely smooth regions of a picture. An extra to these luma prediction modes which is a separate chroma prediction is then applied. The *I_PCM* coding mode which is an alternative to Intra_4x4 and Intra_16x16 supports the encoder to simply bypass the prediction and transform coding processes and instead directly send results of the encoded samples. The *I_PCM* mode allows the following processes:

- It supports the encoder to exactly represent the results of the samples.
- It provides a means to precisely represent the values of anomalous picture content without significant data expansion.
- It allows placing a hard limit on the number of bits a decoder must handle for a macroblock without harming the coding efficiency.

In the previous video coding standards such as H.263+ and MPEG-4 Visual where intra prediction has been adopted in the transform domain, intra prediction is always in the spatial domain in *H.264/AVC* standard. This is done in *H.264/AVC* by referring to neighbouring samples of previously coded blocks which are above or/and left to the block to be predicted. This can introduce error propagation in environments where transmission errors propagate due to motion compensation into inter-coded macroblocks. Hence, a constrained intra coding mode can be signalled that supports prediction only from intra-coded neighbouring macroblocks.

2.8. Transformation, Scaling and Quantisation

The transform coding of the prediction residual in *H.264/AVC* is designed similar to previous video coding standards. However, the transformation in *H.264/AVC* is applied to 4x4 blocks, and instead of a 4x4 discrete cosine transform (DCT), a separable integer transform with similar properties as a 4x4 DCT is employed. The transform matrix is represented as follows:

$$A=\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix}$$

As the inverse transform is defined by precise integer operations, inverse-transform mismatches are avoided. The basic transform coding process is very similar to that of preceding standards. At the encoder, the process includes a forward transform, zig-zag scanning, scaling and rounding as the quantisation process followed by entropy coding. At the decoder, the inverse of the encoding process is performed except for the rounding. [4] describes in more details the aspects of the *H.264/AVC* transform. In *H.264/AVC*, smaller size transforms are used for the following reasons 1) to improve prediction process for inter and intra. Hence the residual signal will possess less spatial correlation which means the transform has less to offer with regards to spatial correlation. This also indicates that a 4x4 transform is essentially as efficient in removing statistical correlation as a larger transform. 2) The smaller transform contains visual benefits resulting in less noise around edges known as ringing artifacts. 3) Less computations and wordlength are required. The *H.264/AVC* standard involves only adds and shifts specification such that encoder and decoder mismatch is avoided. This has been a problem with earlier 8x8 DCT standards. Simulations results in section 0 show the performance of 4x4 and 8x8 transformations types.

A quantisation parameter (*Qp*) is used for determining the quantisation of transform coefficients in *H.264/AVC*. The parameter can take values between 0 and 51. These values are arranged so that an increase in 1 in *Qp* means an increase of quantisation step size by approximately 12% and an increase of 6 means an increase of quantisation step size by exactly a factor of 2. In comparison with prior standards, the *Qp* takes values between 0 and 31 (MPEG-4). The wide range of values in *H.264/AVC* supports adaptable and flexible quantisation level for the transform coefficients.

III. Proposed Streaming Methods

The design of every parameter provides a particular amount of robustness and efficiency. Hence, parameter design will affect the efficiency and adaptation of a produced scalable stream. The following methods are proposed in the implementation of different parameter bitstream.

3.1. Multi-bitstreams to Multi-channels

In this design method, robust and flexible scalable streams are provided for adaptation to heterogeneous networks. A number of encoder states are generated from one encoder and one processor. The encoder states are of different performance delivering an automatic adaptation to multi-channels. Every one of the encoder state output produces unique scalability adaptation characteristics. Figure 5 illustrates this technique. Several network conditions can be tolerated from any of the adaptable bitstream and decoders can switch to from one layer/bitstream to another layer/bitstream by the use of SP slice discussed in section 2.4. This type of technique will require a high-speed processor and can be suitable and effective for specific applications such as broadcasting, Internet protocol networks or other real-time applications where un-guaranteed channels are experienced. Simulations results in section 4.1 show an example performance of this method.

3.2. Multi-bitstreams on a Dedicated Server

A *dedicated server* can be used to host the several parameter streams with different scalability and adaptation levels. An algorithm allows monitoring and processing of network conditions. At any condition of the network, a suitable stream can be retrieved from the server and then broadcast into the network. This method reduces the bandwidth requirements and processor complexity in comparison with the above method (multi-encoder states).

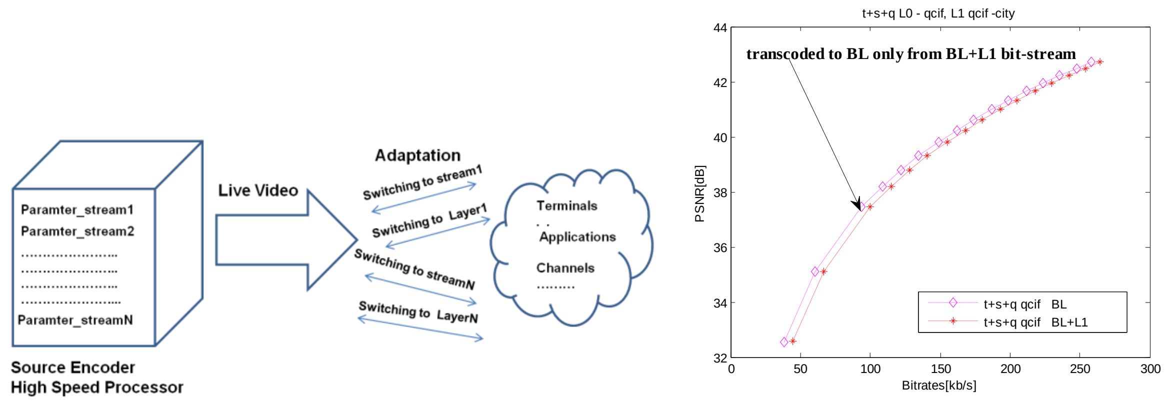


Figure 5 : **Left**: Multi-bitstream to Multi-channel Adaptation **Right**: Switching between bitstream with SP/SI

IV. Introduced adaptation techniques/coding-elements with simulations and results

In this section, we introduce a number of scalability adaptation techniques and simulation results. Also, a number of the described elements and parameters above are simulated and results are presented.

4.1. Multi-bitstreams to Multi-Channels scalability adaptation method

This method is described in section 3. An experiment is conducted for three encoder states X1, X2 and X3. Each of the encoder states is designed for a distinct parameter identifier. The parameter configuration and usage is defined in

TABLE - **I** for each of the encoder states. The Header sizing is altered for the three sets of encoders, the layer quantisation, Hierarchical *GOP* structure, the number of quality layers and the coding sequence structure. These elements influence the encoder performance and efficiency and hence the encoders produce variable scalability outputs for adaptation to multi-channels capability and dynamic resources. Figure 6 presents objective quality performances from an experiment with city standard ITU sequence. Figure 7 presents the distinct scalability provided by each of the encoder states. The overall X1, X2 and X3 states scalabilities will be provided on a multi-network environment improving adaptation and reducing the amount of delays and loss of data. The negative impact of this method is that a high-speed processor is required to process the multiple encoders at the same time.

4.2. Qp Constraint Algorithm (QCA)

H.264/AVC adapts the quantisation parameter within the video sequence frames and slice units. The *Qp* parameter automatically changes for rate control with a reference to a minimum value. This happens at rate-control in regulating the bitstream rates. However, constant values for the minimum *Qp* do not achieve the set target rates at most and especially at higher bit-rates. This situation does also affect the ELs in achieving an optimised performance due to their reference from BL. The BL uses this set minimum *Qp* but does not consider the effect of this on the higher layers (ELs) performance. The higher layers that reference from the BL will perform better if considered in this *Qp* setting. To achieve the target bitrates and optimum performance, the currently computed bits need to be considered prior to assigning the minimum *Qp* for BL control. Experiments are conducted which proved that the *Qp* value can be restricted for a certain value to support the achievement of the target rate and for better scalability control. Experimental results in

TABLE - II, TABLE - III and

TABLE **- IV**, show the application of this technique for several scalability techniques. The experiments reveal that the *Qp* restriction is required at higher rates > 32kbit/s. At target rates >= 300kbits, a maximum *Qp* Delta of 8 is required to achieve the target rates. The pseudo-code below outlines an algorithm *Qp Constraint Algorithm* for a dynamic selection and restriction of minimum *Qp* value during BL rate control.

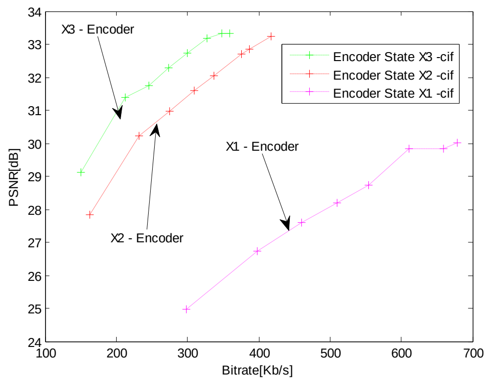
```
If $R_t$ <= minRt {
    $Q_p = Q_{pcurrent} + MinDeltaQ_p$
} else
{
    $Q_p = Q_p + MaxDeltaQ_p$}
```

Where R_t is the computed target rate and $MaxDeltaQ_p$, $MinDeltaQ_p$ are the maximum (8) and minimum (2) quantisation level respectively. minRt defines a target rate of 32kbits/sec and $Qp_{\text{-current}}$ is the last *Qp* used.
In

TABLE **- IV**, the higher ELs do not improve in the bits reduction. This is because ELs below EL0 do not reference from the BL only. They reference from the immediate above ELs. Hence in a multi-layer bitstream, this algorithm is implemented where the BL is needed or the next Enhancement Layer (EL0).

TABLE - I : X1, X2 and X3 coder states distinct design configuration

Notation	**Definition**
Qp	Quantisation parameter value (incremented by 2 for X2 and by 3 for X3)
Seq	Type of sequence structure used (X1 = *IPPP*, X2= *IBBP*, X3 = *IBBP*)
Quality	Number of quality layers used (X1 = 2, X2 = 3, X3 =3)
SEI	Supplemental Enhancement Information (True for X1,X2 and X3)
Map	Type of *MB* map to slice group (same for X1, X2 and X3)
Header_size	Reduced number of headers (X1 = 7%, X2 = 40%, X3 = 67%)
CGS	Coarse Grain Scalability (X2 and X3)
MGS	Medium Grain Scalability (X1)
GOP	Group of Pictures (X1=4, X2= 16 and X3=64)



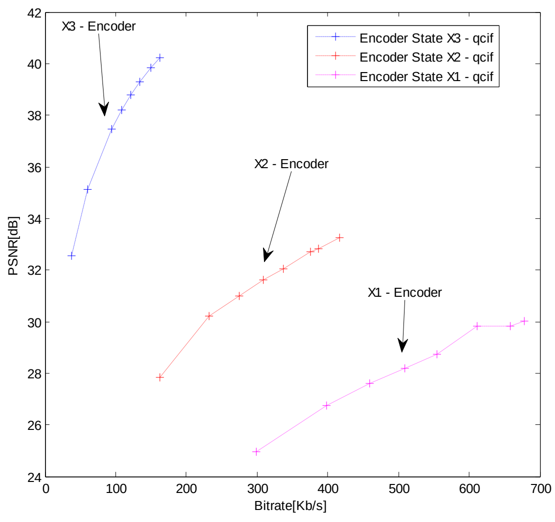


Figure 6 : Encoder States X1, X2 and X3 objective quality performance. Left: CIF city sequence. Right: QCIF city sequence.

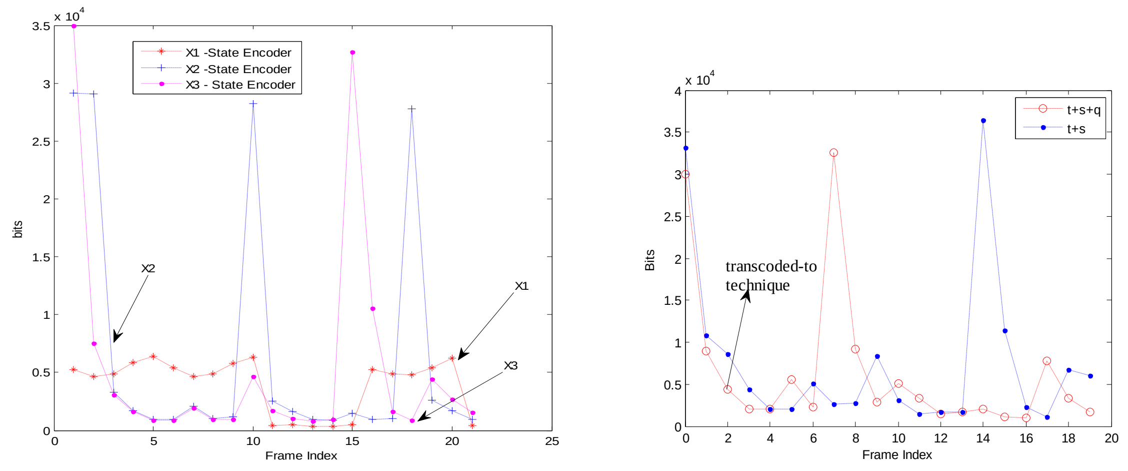


Figure 7 : Left- Encoder X1, X2 and X3 states distinct scalabilities **Right-** Switching between bitstream with SI/SP slice

TABLE - II : QCA Performance for temporal+quality scalability- foreman sequence

Constraint Q_p		**Qp Constraint Algorithm**	**non-usage of Qp Constraint Algorithm**
	Target bits (kbit/s)	**Actual Bits(kbit/s) / PSNR(dB) / Q_p used**	**Actual Bits(kbit/s) / PSNR(dB) / Q_p used**
(L0)	32	31.26 / 30.37 / 29	47.10 / 33.35 / 35.76
(L1)	32	69.52 / 33.72 / 36.67	51.65 / 33.36 / 36.36
(L0)	400	176.69 / 39.62 / 40.25	226.67 / 42.35 / 20.69
(L1)	400	182.67 / 39.61 / 40.44	230.86 / 42.35 / 26.95

TABLE - III: QCA Performance for temporal+spatial scalability- foreman sequence

Constraint Q_p		**Qp Constraint Algorithm**	**non-usage of Qp Constraint Algorithm**
	Target bits(kbit/s)	**Actual Bits(kbit/s) / PSNR(dB) / Q_p used**	**Actual Bits(kbit/s) / PSNR(dB) / Q_p used**
(L0)	318	46.13 / 33.21 / 21.42	173.80 / 40.63 / 22.09
(L1)	350	302.95 / 34.88 / 33.16	349.06 / 33.88 / 35.00
(L0)	350	46.13 / 35.39 / 20.84	186.84 / 41.00 / 21.52
(L1)	382	327.86 / 35.39 / 32.63	388.82 / 34.49 / 34.00
(L0)	414	46.13 / 33.21 / 19.83	212.13 / 41.66 /20.51
(L1)	446	354.64 / 35.81 / 31.71	413.16 /34.51 /34
(L0)	446	46.13 / 33.21 / 19.39	223.52/41.95/20.6
(L1)	478	363.35 / 35.94 / 31.29	424.93 / 34.54 /33.94

TABLE - IV: QCA Performance for temporal+quality+spatial scalability- foreman sequence

Constraint Q_p		**Qp Constraint Algorithm**	**non-usage of Qp Constraint Algorithm**
	Target bits(kbit/s)	**Actual Bits(kbit/s) / PSNR(dB) / Q_p used**	**Actual Bits(kbit/s) / PSNR(dB) / Q_p used**
(L0)	320	252.40 / 35.14 / 35.14	279.15 / 37.02 / 24.76
(L1)	320	272.71 / 35.45 / 28.81	283.35 / 37.02 /29.00
(L2)	352	555.28 / 29.25 /38.01	492.00 / 29.63 / 37.40
(L3)	352	590.93 / 29.23 / 38.72	513.06 / 29.62 /37.83
(L0)	384	252.40 / 35.14 / 27.48	323.27 / 37.89 / 23.66
(L1)	384	287.88 / 35.79 / 27.72	327.26 / 37.88 / 29.00
(L2)	416	605.77 / 29.69 / 37.09	560.77 / 30.00 / 36.62
(L3)	416	642.03 / 29.67 / 37.81	582.26 / 29.97 / 37.05

4.3. Quantiser-Step-Size (QSS) Evaluation

An experiment is conducted with standard ITU city sequence to determine the quality and scalability performance of variable quantisation step-size in the *H.264/AVC* encoder. The experiment uses the temporal+quality+spatial scalability technique. It can be deduced from the results in

TABLE **- V** that, bitrates reduction is obtained when *Qp* is incremented. The amount of reduction tends to be uniform as the *Qp* increment gets higher. The incremented value from 28 up to 40 yields a reduction of ~7 bits while increments from 22 to 26 yield better bits and variable reduction.

TABLE - V : QSS Performance with temporal+quality+spatial scalability

Quantisation Step	**bitrates [kbit/s]**	**PSNR [dB]**
22	181.70	40.50
24	143.70	39.23
26	115.63	38.08
28	95.38	36.94
30	78.25	35.75
32	65.14	34.55
34	54.88	33.37
36	46.14	32.14
38	39.01	30.88
40	33.86	29.68

However, this depends on the video sequence and the produced coefficients. The objective picture quality tends to reduce with higher *Qp* values and therefore careful consideration should be made in deciding the *Qp* values especially at lower bit rates.

4.4. Entropy Algorithms Performances Evaluation

Experimental results in Figure 9 show the objective quality performance of *CABAC*/*CAVLC* algorithms. Their performance reveals slight differences at bitrates while generally better performance is observed within the video sequence coded with *CABAC* algorithm. The subjective quality performance of the two algorithms shows the same quality performance [7]. Although *CABAC* is more efficient in bits reduction and hence in adaptation, its processing time is found to be higher than *CAVLC* as shown in TABLE - VI.
The Scene Complexity Index (SCI) is smaller for *CABAC* due to fewer bits. The SCI of a picture is the product of its bitrates and average quantiser step size in the corresponding frame [6]. This is expressed in (7-1).

$$SCI = 1/GOP\ [IQ_I + \textstyle\sum_{j=1}^{m} Pj\,Qpj + \sum_{j=1}^{m} BjQBj] \qquad (7\text{-}1)$$

Where I, P and B are the target bit-rates for the I, P, B frames, and Q_I, *QP*, Q_B are their average quantiser step sizes respectively. Applications and channel conditions will allow a suitable structure based on the calculated complexities.
Figure 8 presents the real time simulation results for *CABAC*/*CAVLC* algorithms. The simulation is conducted with BUS sequence for 150 frames using temporal+spatial+quality scalability. Generally, in the generated results, *CABAC* experience less delay than *CAVLC*. *CABAC* has shown better compression efficiency and adaptation performance. The subjective quality performance of the *CABAC*/*CAVLC* algorithms reveals a good quality picture [7].

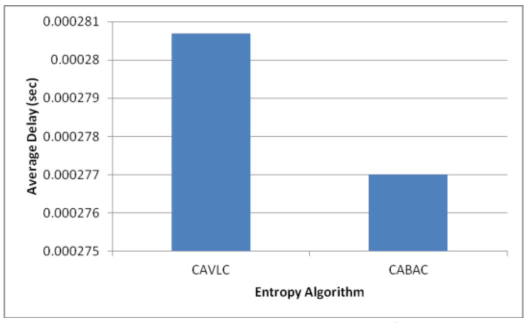
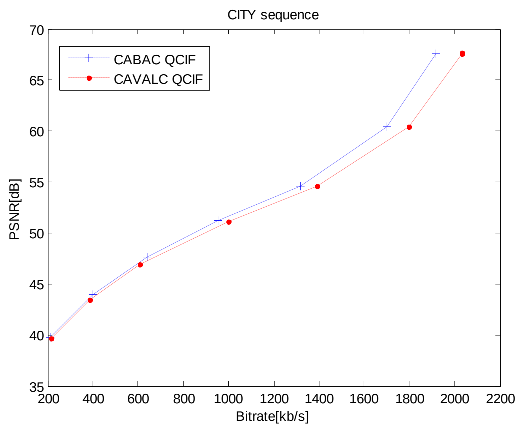


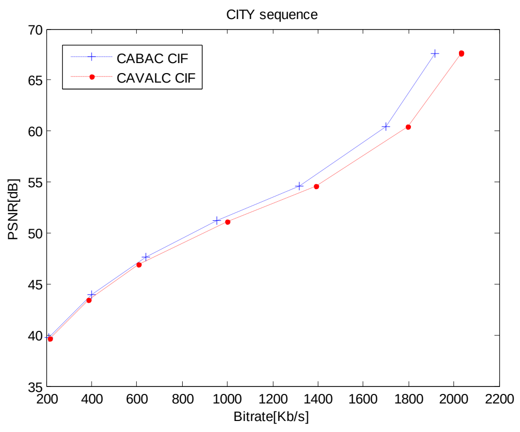
Figure 8: *CABAC*/*CAVLC* real-time performances

TABLE - VI : CABAC/CAVLC Complexity Analysis.

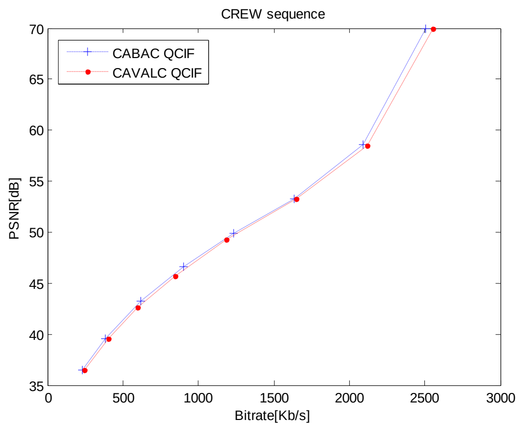
Algorithm	**Encoding time**	**SCI**	**Sequence**
CABAC	5min, 11.83sec	271	Bus
CAVLC	4min, 15.24 secs	289	
CABAC	10min, 5.9sec	335	Football
CAVLC	10min, 3.63 secs	338	
CABAC	10.79 min	333	Crew
CAVLC	5m, 25.06 secs	335	
CABAC	7min, 35.40 secs	286	Mobile
CAVLC	8min, 4.94 secs	292	
CABAC	3.54 min	168	City
CAVLC	3min, 22.25 secs	175	



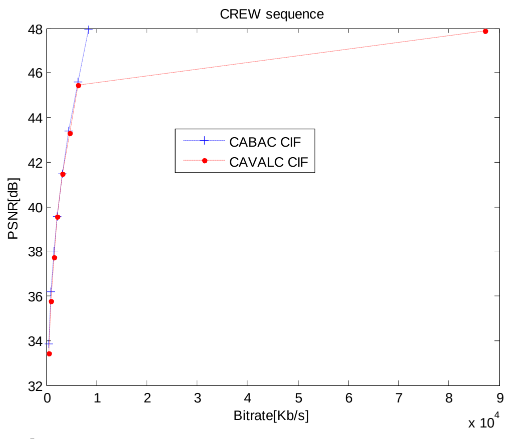
a. *CABAC/CAVLC*, City- L0, QCIF



b. *CABAC/CAVLC*, City-L3, CIF



c. *CABAC/CAVLC*, Crew-L0, QCIF

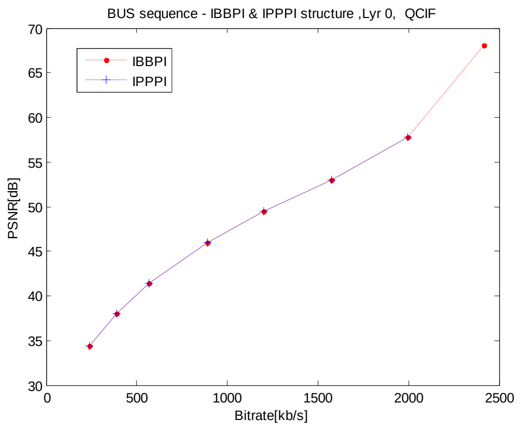


d. *CABAC/CAVLC*, Crew- L3, CIF

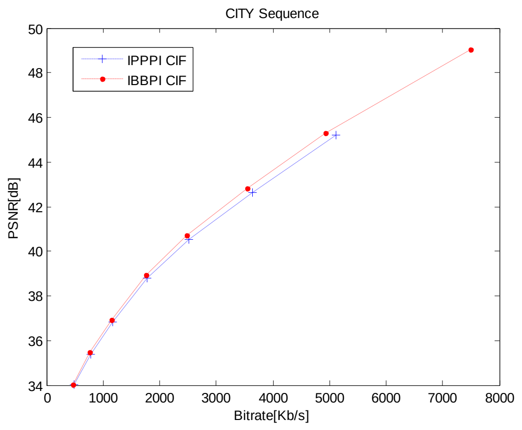
Figure 9: Performance of *CABAC/CAVLC* algorithms for temporal+spatial+quality scalability

4.5. Sequence Structure Performance Evaluation

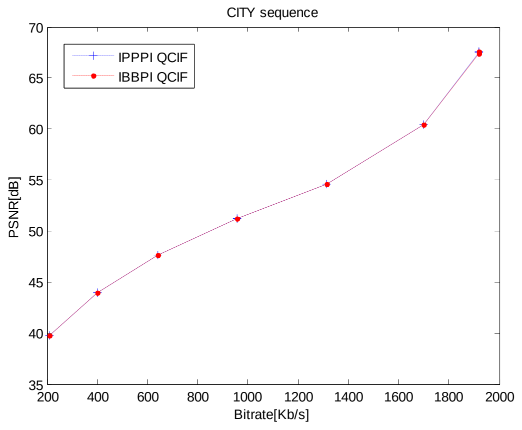
Figure 11 presents real-time simulations results for *IBBP/IPPP* coded video sequences. The average delay statistics show that, *IBBP* structure encountered less delay time. Experimental results in Figure 10 indicate that *IBBP* coding pattern achieves better bits reduction than that of *IPPP* particularly at bitrates >1kbit/s. However, the encoding complexity is higher in the *IBBP* sequence for high motion video sequence (football) as shown in TABLE - VII while its Scene Complexity Index (SCI – described in section 4.4) value is lower. This value has proved the bits reduction of IBBP over *IPPP* video sequence. In reconstruction, *IPPP* is cheaper than *IBBP* sequence. This is due to an additional prediction reference that *B-frames* adopted.



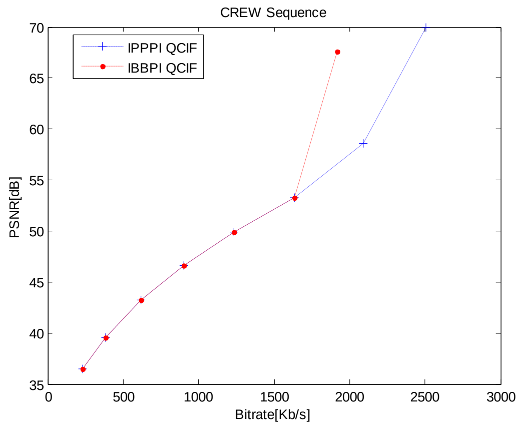
a. *IBBP/IPPP*, Bus – L0, QCIF



b. *IBBP/IPPP*, City – L3, CIF



c. *IBBP/IPPP*, City – L0, QCIF



d. *IBBP/IPPP*, Crew – L0, QCIF

Figure 10: *IBBP/IPPP* Sequence Structure Performance

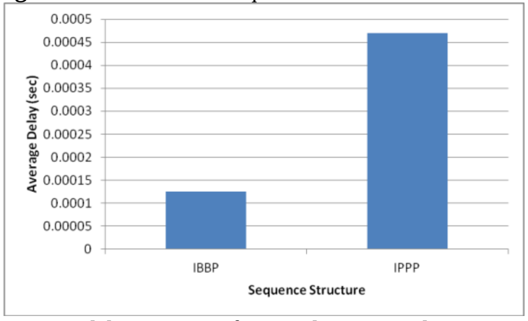


Figure 11: *IBBP/IPPP* Sequence- Average delay statistics from real-time simulations over heterogeneous networks.

TABLE - VII : *IBBP/IPPP*- **Left**: encoding time and SCI. **Right**: Sequence decoding time

Structure	Encoding Time	SCI	Sequence
IBBP	3.57 min	261.5	Foreman
IPPP	3.82 min	321.5	Foreman
IBBP	3.80 min		Foreman
IPPP	3.87 min		Foreman
IBBP	3.35 min		Foreman
IPPP	3.83 min		Foreman
IBBP	10.27 min	275	Football
IPPP	8 .40 min	275	Football
IBBP	10.23 min		Football
IPPP	8.12 min		Football
IBBP	3.47min	203	Harbour
IPPP	4.44min	205.3	Harbour
IBBP	3.99min		Harbour
IPPP	4.13min		Harbour

Structure	Layer	Decoding time(secs)	Sequence
IBBP	0	0.86	Harbour
IPPP	0	0.78	Harbour
IBBP	1	5.28	Harbour
IPPP	1	5.02	Harbour
IBBP	2	5.34	Harbour
IPPP	2	5.09	Harbour
IBBP	3	5.29	Harbour
IPPP	3	5.21	Harbour
IBBP	0	0.74	Foreman
IPPP	0	0.74	Foreman
IBBP	1	4.7	Foreman
IPPP	1	4.5	Foreman

4.6. Hierarchical Group of Pictures (GOP) Structure Performance Evaluation

A video sequence is divided into units of pictures called *GOP*. Figure 13 shows that the encoded video sequence with a *GOP* allocated number of pictures provides a more efficient compression and higher bitrates savings on the scalable stream. This is due to better prediction within the frames of a *GOP* unit. A higher level of scalability is also indicated from experiments in Figure 12 for a *GOP* with a higher number of pictures. Figure 12 indicates that *GOP* units with 64 pictures will generate 14 scalable layers with variable frame rates, bitrates and quality levels on to the network and *GOP* units with 32 pictures will produce 11-12 scalable layers within the stream. *GOP* from graphical results represents the number of pictures allocated in a *GOP*.

TABLE - VIII

TABLE **- VIII** presents variable *GOP* with a variable number of pictures with their encoding time and SCI. *GOP* with a higher number of pictures requires more processing time but less SCI (section 4.4) value. This is due to the additionally produced scalability layers which slightly incur a computational cost but support flexible scalability adaptation. Figure 16 shows the scalability variations for a variable *GOP* structure. The hierarchical structure with a higher number of pictures generates a stream with reduced bits. The subjective quality performance of different *GOP* structures shows a good quality picture [7].

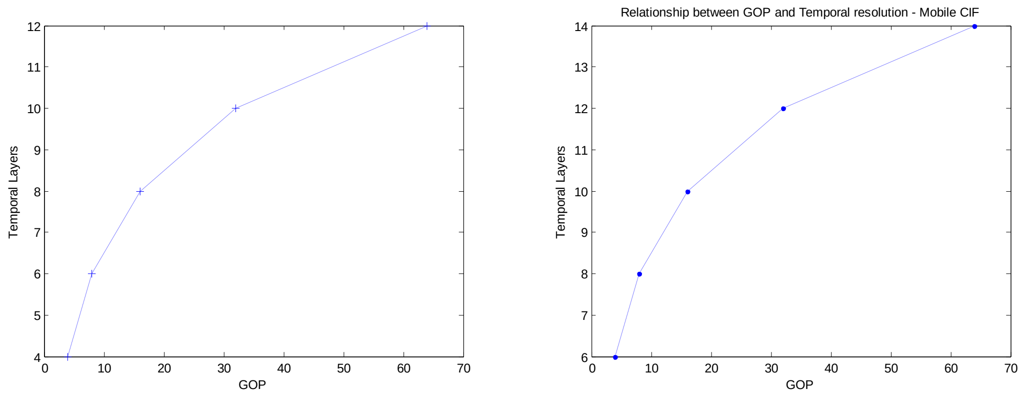


Figure 12: Relationship between *GOP* and Temporal Scalability –**Left**: Mobile-L0, QCIF **Right**: Mobile-L3,CIF

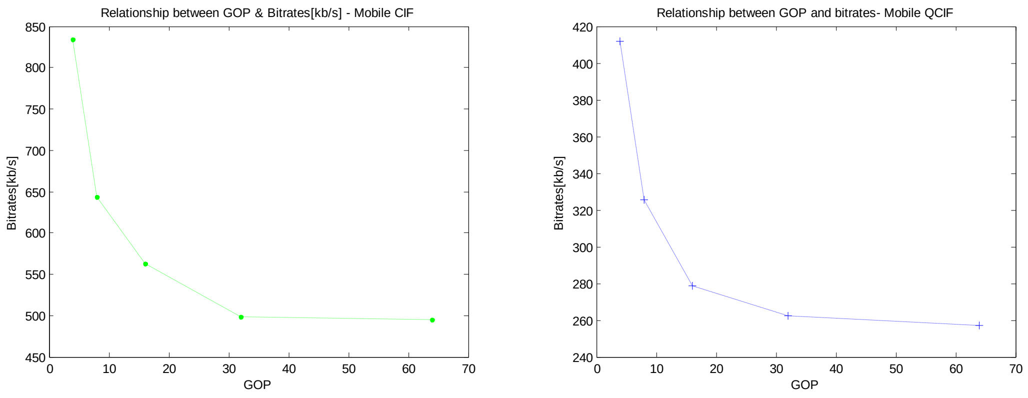


Figure 13: Relationship between *GOP* and Bitrates –**Left**: Mobile-L3, CIF **Right**: Mobile-L0, QCIF

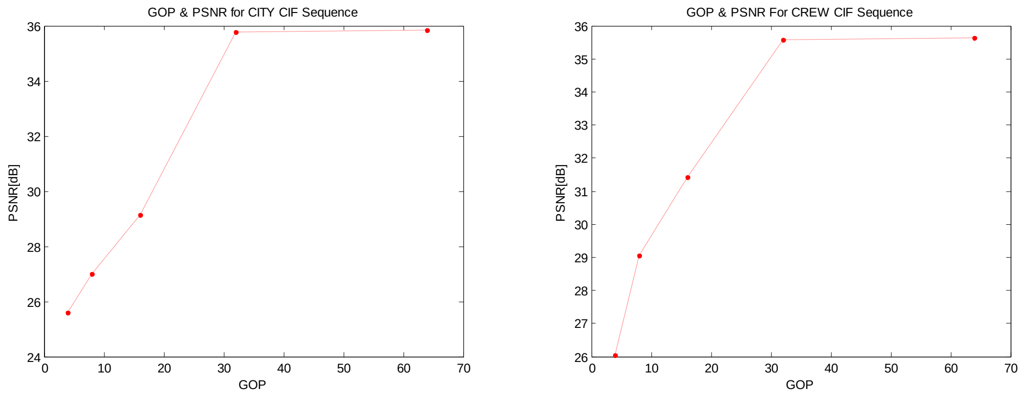


Figure 14: Relationship between *GOP* & PSNR- **Left**: City- L3, CIF **Right**: Crew- L3, CIF

TABLE - VIII: Variable *GOP* structure encoding time and SCI

GOP size	**Encoding time (min)** Foreman/harbour	**SCI** Foreman/harbour	**Scalable layers** Foreman/harbour
2	1.73/1.68	2128/2128	6/6
4	2.58/2.74	1872/1872	10/10
8	3.17/3.2	1038/1038	13/13
16	3.57/3.78	549/549	18/18
32	4.05/4.06	255/255	22/22
64	4.15/4.34	222/222	26/26

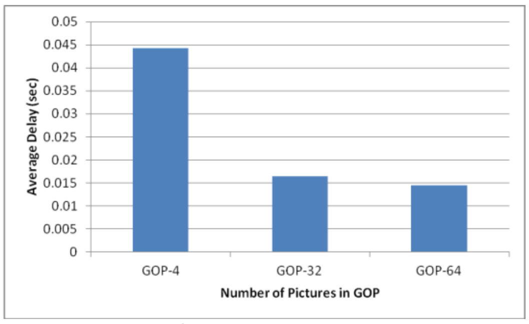


Figure 15 : *GOP* structures: Average delay statistics from real-time simulations over heterogeneous networks.

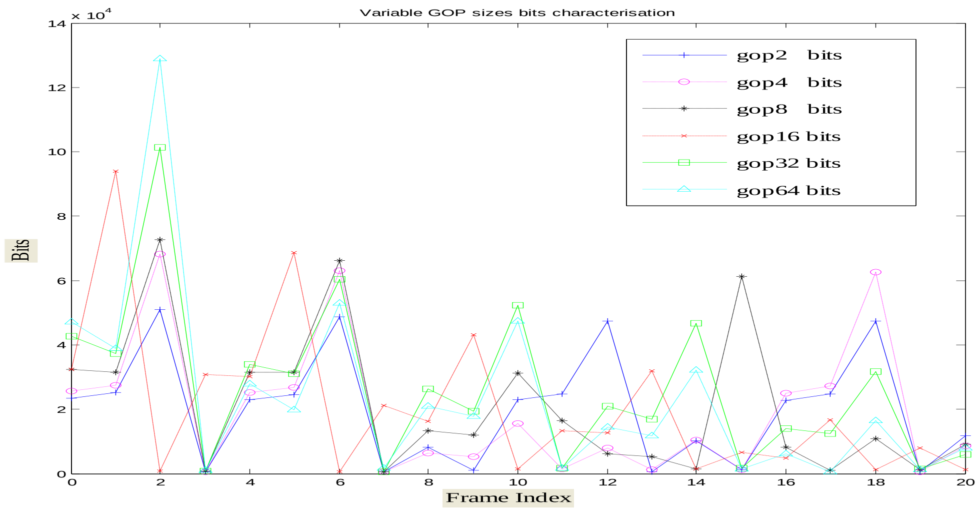
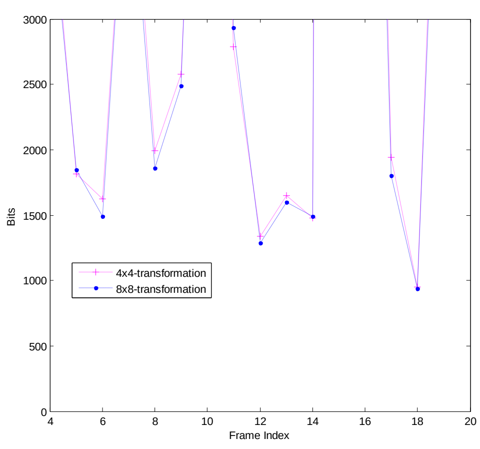


Figure 16: Variable *GOP* structure bits characterisation- football sequence at 224kbit/s, 130frames QCIF.

4.7. I_PCM coding, intra_8x8 and intra_4x4 transformation performances

I_PCM coding and intra predictions are discussed in section 0-2.8 Experimental results in Figure 17 show the scalability characteristics of 4x4/8x8 transformations. 8x8 transformation is shown to be less expensive although 4x4 is adopted in *H.264/AVC* design for the reasons discussed in section 0. Also, *I_PCM* coding performance is shown where intra coding performs better in scalability adaptation. TABLE - IX shows the 4x4/8x8 transformations quality performances and *I_PCM* coding adaptation and quality characteristics.



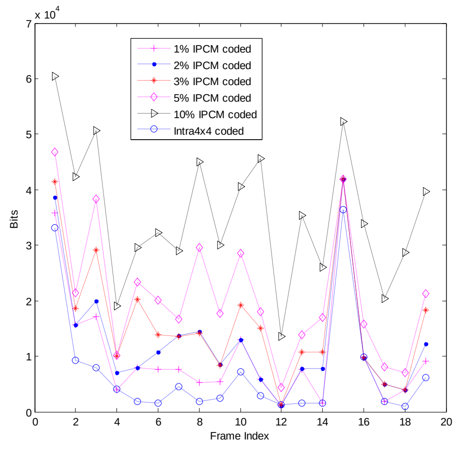


Figure 17: Scalability performance - **Left**- 4x4/8x8 transformation **Right**- *I_PCM* and intra coding

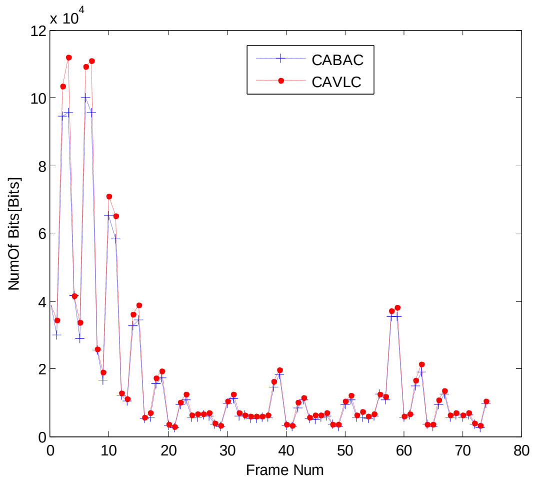


Figure 18: Scalability variation in *CABAC* = 245kbit/s, 34.5dB and *CAVLC*=262kbit/s, 34.5dB – BUS Sequence, 75 frames.

TABLE - IX: **Left**- 4x4/88 transformations quality performance. **Right**- *I_PCM* coding quality performance

Layer/ Target rate (kbit/s)	**4x4_Transformation** Achieved Bitrates(kbit/s)/ PSNR[dB]	**8x8_Transformation** Achieved Bitrates(kbit/s)/ PSNR[dB]	**Layer/ Target rate (kbit/s)**	**Achieved Bitrates(kbit/s)/ PSNR[dB]**	**% coded with I_PCM**
L0/32	35.59 /35.74	35.28 / 35.70	L1/32	479.93 / 44.33	10%
L1 /32	45.86 / 35.56	46.30 / 35.60	L0 /32	134.15 / 33.74	5%
L0/128	78.22 / 39.63	78.20 / 39.65	L1 /32	88.45 / 34.62	1%
L1/128	96.94 / 37.81	97.55 / 37.73	L0 /32	72.87 / 34.70	2%
			L0 /32	93.13 / 34.25	3%

4.8. Slice Groups and Flexible Macroblock Ordering (FMO) Implementation Performances

In Figure 19 - Figure 21, the implementation of slice groups with crew and mobile standard sequences is shown. The results establish that the use of a flexible slice structure can be used to improve the adaptation flexibility.

The left and right results show the slice group implementation for layer0 (L0) and layer3 (L3) respectively. This means that different complexity levels can be adopted in distinct layers of the scalable stream, thereby improving the flexibility of the adaptation. In the results, variable bitrates, quality and bitstream sizes are produced for each of the available packet/layer within the bitstream. These distinct layers representing the video sequence can support variable channels and applications given their diversity and limitations.

Acknowledgement

The authors thank the ITU-T VCEG, ISO/IEC MPEG and ISO/IEC Joint Video Team. Thanks also to all the referenced authors in this paper.

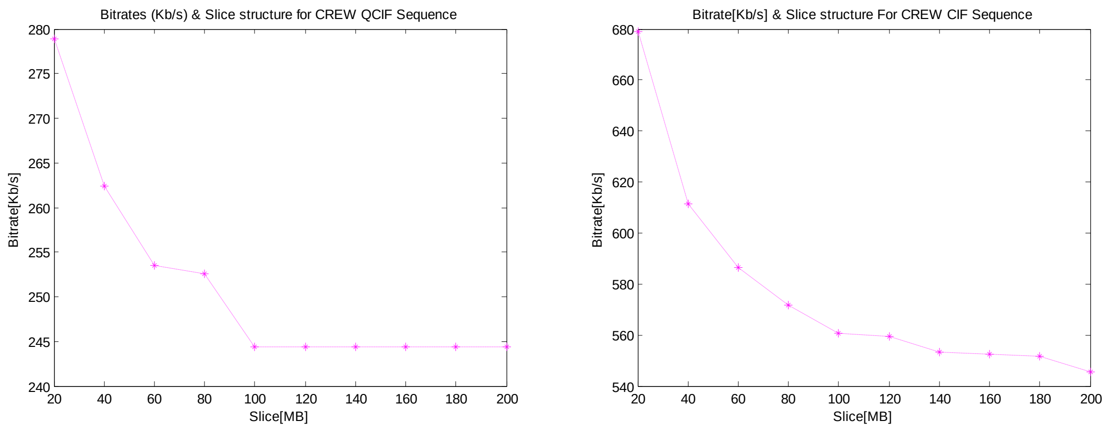


Figure 19: Bitrates & slice structure, crew sequence **Left**- L0. **Right**- L3

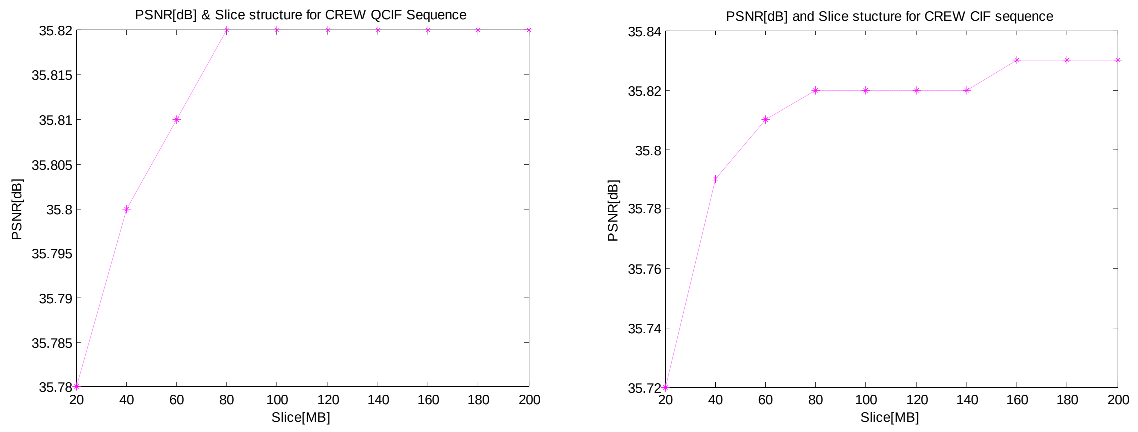


Figure 20: PSNR and slice structure, crew sequence **Left**- L0 Right- L3

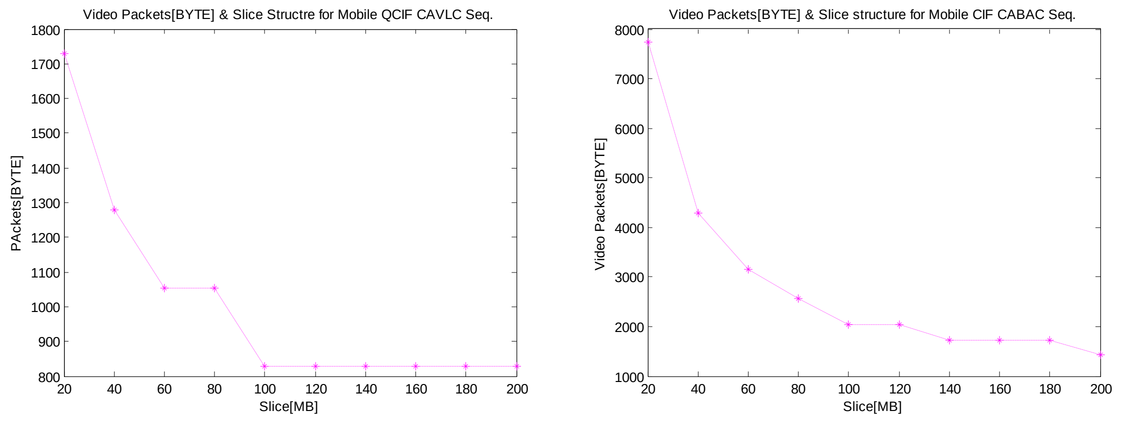


Figure 21: Bitsream size and slice structure, mobile sequence, **Left**- L0 **Right**- L3

V. Conclusion

In this paper, we evaluated the major *H.264/AVC* coding elements. The significant differences with the prior standards are discussed which include: 1) Adaptive deblocking filter with the motion compensation vectors 2) Enhanced and Context Adaptive Compression Algorithms 3) use of small block size transform 4) support for the implementation of flexible slice structure for video adaptation and error resilience.

To enhance the adaptation capability of *H.264/AVC*, *Qp Constraint Algorithm(QCA)* is introduced. *QCA* algorithm constraints the *Qp* parameter value during rate control at a bitrate ≥300kb/s. The algorithm gained bit reduction up to 100kbits depending on the current computed bitrate.

We also implemented the slice group usage for *H.264/AVC* in which the results show that, an efficient usage of *slice group* will support bitstream adaptation. It is shown from experimental results with several and different video sequence types that, an optimum performance is obtained by assigning 200 MBs/Slice.

The use of different priority bitstreams for multi-channel adaptation is proposed and experimented. Experimental results show that, *multi-bitstream* encoding provide support for *multi-channels resources*

adaptation. However, a high speed processor is required. This method is particularly efficient for video broadcasting or systems where multi-channel resources and applications are involved.
Several real-time simulations over heterogeneous networks and experimental results show that:

1. Better adaptation and scalability is achieved by coding with *IBBP* than *IPPP* sequence structure over multi-channel networks. Real-time simulations over multi-channel networks show that *IBBP* and *IPPP* structures experience an average delay of $1.2x10^{-4}$sec and $4.7x10^{-4}$sec respectively. Additional decoding complexity is experienced in coding with *IBBP* structure. The use of higher speed processor can lessen the complexity.
2. A video sequence coded with a *GOP* structure of higher number of pictures can achieve better coding efficiency and adaptation/scalability over multi-channel networks. An average delay of $4.4x10^{-2}$sec, $1.6x10^{-2}$sec and $1.5x10^{-2}$sec is obtained for *GOP* structure of size 4, 32 and 64 respectively. Additional coding complexity is incurred on large *GOPs* up to 1min encoding time depending on the *GOP* structure used and the processor specification.
3. Better efficiency and adaptation over heterogeneous network is shown when coding with *CABAC* algorithm. An average delay of $2.8x10^{-4}$sec and $2.7x10^{-4}$sec for *CAVLC* and *CABAC* algorithms respectively is obtained. An additional encoding complexity is obtained in *CABAC* algorithm up to 5min depending on the video sequence, processor speed and bitrates.
4. *Intra4x4* coded sequence show better adaptation than *I_PCM* coding type. Different percentage levels of *I_PCM* coded video show less efficiency. *I_PCM* coding should only be used on selected required applications.
5. *4x4 Transformation* shows better adaptation and scalability characteristics than *8x8 Transformation* in the Enhancement Layer video data. This characteristic feature is a reversed case on the Base Layer video data.

References

[1] T Wiegand, GJ Sullivan, G Bjontegaard, A Luthra. Overview of the H. 264/AVC video coding standard, Circuits and Systems for Video Technology, IEEE Transactions on. 13 (2003) 560-576.
[2] V. Video. H.265 Products, URL: http://www.vanguardvideo.com/h265.php. (2013).
[3] O Modeller. OPNET Technologies Inc, Bethesda, MD.URL: http://www.opnet.com. (2009).
[4] IEG Richardson. H. 264 and MPEG-4 video compression. 2003, John Wiley&Sons Ltd., New Jersey.
[5] H Schwarz, D Marpe, T Wiegand. Overview of the scalable video coding extension of the H. 264/AVC standard, Circuits and Systems for Video Technology, IEEE Transactions on. 17 (2007) 1103-1120.
[6] M Ghanbari, Standard Codecs: Image compression to advanced video coding, (2003).
[7] S. Muhammad, *Scalable and Network Aware video coding for advanced communications over heterogeneous networks*, PhD Thesis Submitted to SED, Computer & Electronic Engineering, Brunel University, London. (2013).
[8] JV Team. of ISO/IEC MPEG and ITU-T VCEG, reference software to committee draft, JVT-F100 JM10. 2 (2004).