Graduate School Form 30
Updated 1/15/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Delaram Amiri

Entitled
BILATERAL AND ADAPTIVE LOOP FILTER IMPLEMENTATIONS IN
3D-HIGH EFFICIENCY VIDEO CODING STANDARD

For the degree of Master of Science in Electrical and Computer Engineering

Is approved by the final examining committee:

Mohamed El-Sharkawy Chair	
Brian King	
Paul Salama	
Maher Rizkalla	

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Mohamed El-Sharkawy

Approved by: Brian King 8/13/2015

Head of the Departmental Graduate Program Date

BILATERAL AND ADAPTIVE LOOP FILTER IMPLEMENTATIONS IN 3D-HIGH EFFICIENCY VIDEO CODING STANDARD

A Thesis

Submitted to the Faculty

of

Purdue University

by

Delaram Amiri

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

December 2015

Purdue University

Indianapolis, Indiana

ACKNOWLEDGMENTS

Thanks to Fraunhofer Institute for providing 3D-HEVC software [1] and Poznan University of Technology, Poznan, Poland and Nagoya University, Japan [2] for providing multiview test sequences [3]. Thanks to my adviser, Dr. El-Sharkawy, professor of Electrical and Computer Engineering at IUPUI for his guidance in my research. Thanks to my committee members for giving me feedbacks to improve my thesis and research. Thanks to my mother, Maryam Bizhannia, for her love and support in every step of my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ALF	Adaptive Loop Filter
CB	Coding Blocks
CTU	Coding Tree Units
DF	Deblocking Filter
FBF	Fast Bilateral Filter
HEVC	High Efficiency Video Coding
PSNR	Peak Signal to Noise Ratio
PU	Prediction Unit
SAO	Sample Adaptive Offset
TU	Transform Blocks

ABSTRACT

Amiri, Delaram. MSECE, Purdue University, December 2015. Bilateral and Adaptive Loop Filter Implementations in 3D-High Efficiency Video Coding Standard. Major Professor: Mohamed El-Sharkawy.

In this thesis, we describe a different implementation for in loop filtering method for 3D-HEVC. First we propose the use of adaptive loop filtering (ALF) technique for 3D-HEVC standard in-loop filtering. This filter uses Wienerbased method to minimize the Mean Squared Error between filtered pixel and original pixels. The performance of adaptive loop filter in picture based level is evaluated. Results show up to of 0.2 dB PSNR improvement in Luminance component for the texture and 2.1 dB for the depth. In addition, we obtain up to 0.1 dB improvement in Chrominance component for the texture view after applying this filter in picture based filtering. Moreover, a design of an in-loop filtering with Fast Bilateral Filter for 3D-HEVC standard is proposed. Bilateral filter is a filter that smoothes an image while preserving strong edges and it can remove the artifacts in an image. Performance of the bilateral filter in picture based level for 3D-HEVC is evaluated. Test model HTM- 6.2 is used to demonstrate the results. Results show up to of 20 percent of reduction in processing time of 3D-HEVC with less than affecting PSNR of the encoded 3D video using Fast Bilateral Filter.

1. INTRODUCTION

High Definition (HD) videos are increasingly finding their way into technology and using HD videos in different applications is growing rapidly.

There is a challenge for technology experts that how can they transmit videos without sending all the information and that was the time concept of video compression became an important subject. Compression is beneficial because it can decrease usage of data storage space or transmission capacity. On the other hand, compressed data should be decompressed to be used which needs extra processing time and/or expensive hardware for fast decompression.

Video compression can be either lossy or lossless. In lossless transmission, typically bits are reduced by eliminating statistical redundancy meaning that the amount of wasted spaces and repeated information will be reduced without losing data. Lossy video compression finds the unimportant information and removes it. Lossy video compression encodes an approximation of data content using reduced amount of original data. The amount of possible data reduction in lossy compression is more considerable than lossless compression techniques most of the times. Lossy compression used in multimedia data (e.g. audio, images and video) has different applications, such as streaming media and voice over IP in Internet telephony.

Since lossy video and image compression encodes an approximation of data, quality of reconstructed multimedia data after decompression is not as high as the original data. Therefore, improving the quality of encoded video with minimum cost in complexity of hardware design and minimum running time is an essential challenge. In this thesis we first implement an algorithm that can improve the quality of multiview test sequences while running the encoder faster.

One of the methods to measure the numerical measures is PSNR (peak signal to noise ratio) to compare the compression techniques. This method improves the PSNR encoded by 3D HEVC video compression standard up to 2.1 dB in depth view.

On the other hand, adding new algorithm to the anchor test model increases the running time. So, improving the quality of encoded video without increasing encoder running time is a big challenge for designers. Therefore, we modify the implemented method to decrease the running running time with minimal changes in quality of encoded video. In other words, we implement an optimal method that improves the quality of encoded video with faster process compared to the anchor test model. Results indicates up to 20 percent of reduction in encoding process.

For better understanding, with a 2013-model laptop using a single core and single thread of an Intel Core i5-3210M CPU processor clocked at 2.50 GHz for encoding an anchor 3D HEVC HM 6.2 encoder version the following is observed. To test the anchor test model, we encoded a multiview test sequence, meaning that 3 videos out of 7 videos with their corresponding depth data are encoded. This type of multiview test sequences was recorded as medium detailed scene with 7 cameras of 5cm spacing. Encoding 8 frames in each video of 3 videos with 8 frames of corresponding depth data approximately requires 2729.279 seconds (e.g. 45 minutes and 48.279 seconds). Therefore, encoding multiview test sequences because of their large amount of data compression is very slow. One solution to decrease the encoding time is to use supercomputers with high speed processors and CPUs which can be very expensive.

On the other hand, improving software algorithms in encoder/decoder for higher speed can be cheaper and more efficient. In this thesis we implement an algorithm that can reduce the running time up to 20 percent (e.g. 545.85 seconds or 9 minutes and 9.85 seconds reduction for the mentioned test condition) reaching 2183.4232 seconds (36 minutes and 36.38 seconds) of encoding processing time and improve the quality of the encoded multiview test sequence at the same time.

In this chapter, the general requirements for the development of video coding features are described including an overview of previous video coding standards and overview of HEVC as a new standard and explanation of how HEVC encoder works. In order to deeply discuss in the subject without overload of definitions, some of the terms will be used in this chapter and will be defined in coming sections and chapters. Sections of loop filter and deblocking filter are discussed briefly and will be explained more in forthcoming sections. Aim and scope of the thesis is explained at the end of introduction and proposed algorithm with the results and application considerations are covered in the next chapters.

1.1 Video Compression Standards

Since the early 1990s, the improvement of video coding standards has been directed toward two purposes:real-time video communication and distribution or broadcast of video content. The main design of all major video compression standards since H.261 follows what is defined as block-based hybrid video coding method. In hybrid coding, each block of a picture can be coded as intra picture (considered as coded in an intra coding mode), without using other pictures of the video sequence as a reference. It is temporally predicted (defined as an inter coding mode), where the prediction signal is formed by a block of an already coded picture.

The recent method is defined as motion-compensated prediction and represents the concept of utilizing the large amount of temporal redundancy in video sequences.

The complete intra-coded block is processed using transform coding for exploiting spatial redundancy. First, a linear transform is applied to the input signal which obtains the transform coefficients, then transform coefficients are coded together with side information such as coding modes and motion parameters. Although all considered standards pursue the same basic design, but they are different in various features which leads to an improved coding efficiency from one standard to the next generation. After H.264/MPEG-4, AVC was the second video coding standard that was jointly developed by ITU-T VCEG and ISO/IEC MPEG [4], [5]. High Efficiency Video Coding (HEVC) is the name of the current joint standardization project. A common question is what is the benefit of using HEVC over the previous standards.

Generally, it is hard to judge the benefit of a video compression algorithm on a tool-by-tool basis, as the acceptable algorithm is reflected by an appropriate combination of tools. For example, introduction of larger block structures has impact on motion vector compression, but should be followed by fusion of larger transform structures as well. Using a configuration for HEVC that is close to settings of H.264/MPEG-4 AVC, the bit-rate savings ranges from around 30% to nearly 67% in HEVC, depending on the video sequence. The average bit-rate reduction over all the sequences tested was 49.3%. The subjective benefit for HEVC seems to exceed the benefit measured using PSNR, and the benefit is greater for low bit rates, higher-resolution video content and low-delay application encodings [6].

1.2 HEVC as a New Standard

The High Efficiency Video Coding (HEVC) standard is the recent joint video project of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) standardization organizations,working together known as the Joint Collaborative Team on Video Coding (JCT-VC) [7]. JCT-VC is tasked to work with 3D HEVC extensions; and for work on 3D video topics for multiple standards including 3D video extensions for HEVC in particular, they formed a team known as the Joint Collaborative Team on 3D Video (JCT-3V). JCT-3V has developed a 3D extension to HEVC to support 3D- displays using a similar coding process as HEVC standard.

The wide coverage of high definition (shortly as HD) video content and displays is developing toward variety of applications of UHD video services, as well as increasing applications of high-quality video services to mobile devices resulted in request for a new video coding standard in 2009 called HEVC. A complete video coding system of HEVC is sectioned to four blocks [8]:

Source

Source of video sequence is the output in a digital form. It may be followed by the following processing time.

Pre-Processing

Operation on the uncompressed video sequence such as changing color format, color correction and modification or denoising is considered as a part of this coding block.

Encoding

Transforming the input video to a coded bit-stream. The purpose of encoding is to form a compressed version of the input video sequence to be ready for transmission through a channel to a receiver.

Transmission

Packaging the bit-stream to a correct format and transmission over the channel. Transmission includes transmitting and receiving from the receiver side, along with channel coding for loss protection and security purposes. In some cases feedback from from the receiver to the transmitter is needed to control encoder configuration for real time requirements. Feedback can be useful for the applications that retransmission request in the case of lost data.

Decoding

Transforming the received bit-stream to a reconstructed video. Video encoding can be lossy compression to reach the desired transmission bit-rate which makes the decoded video to be an approximation of the input video. If unrecoverable transmission losses happen, decoder uses methods to recover the corrupted video sequence.

Post-Processing

Operation on the reconstructed video in the decoder side of coding is for enhancement or adaptation of the sequence for the display. Post processing block includes color correction, re-sampling or trimming or special modifications and effects can be done.

Display

Presentation of the video to be viewed. Video sequence should be converted to a correct color format to be displayed. Output timing of frames are of the video is an essential task of reach the intended visual effectiveness.

The processing stages described above shows a simplified video coding system. Since all the implementation and focus is on encoder side of coding system we will describe encoding block of HEVC in forthcoming section.

The encoder stage of a coding system first transforms the input video sequence to a bit-stream that has packet oriented structure in which encoded data is hierarchically organized. The encoder should generate a bit-stream to the application needs such as selection of encoding tools.

The optimization of the encoder control decision is an important task of encoder design. On the picture level, selecting the coding tool and bit allocation for regions of frames should be chosen carefully. Two separate encoders using the same video coding features may construct different video qualities [8].

HEVC kept the main hybrid coding structure of previous video compression standards such as H.264/AVO. An important difference between HEVC and previous standards is the use of adaptive quadtree structure based on a coding tree unit (CTU) instead of a macroblock in previous standards.

In general, the quadtree coding architecture is defined by splitting the image into blocks and units. A CTU (coding tree units) contains coding tree blocks (CTB with the size of 64×64, 32×32, or 16×16) and syntax defining coding data and block subdivision. In other words, a block in hybrid coding contains an array of samples and sizes, while a unit contains information regarding one luma and corresponding chroma blocks along with syntax related to the blocks and units that needs to be coded. CTBs are divided into smaller partitions called coding units (CUs). Note that sub partition of CTB into four smaller areas are called quadtree. These sub partions are called coding units (CUs).This subdivision results in coding unit (CU) leaves with coding blocks (CB). Each CU is divided into prediction units (PUs) for either inter or intra prediction with various sizes of 64×64 down to 4×4. In addition, each CB is divided in to prediction blocks (PB) and transform blocks (TB) Sub partition of The prediction residual is then coded using transform units (TUs with the size of 4×4, 8×8, 16×16 and 32×32). This adaptive size method is especially suitable for high resolutions, such as $4k \times 2k$ which is a desired resolution for some HEVC applications. Figure 1.1 shows an example of CB and TB quadtree structure.

Moreover, partitioning modes for dividing a CB into PBs are demonstrated in Figure 1.2. All partitioning modes specifying how to split a CB into PBs are shown in Figure 1.2.

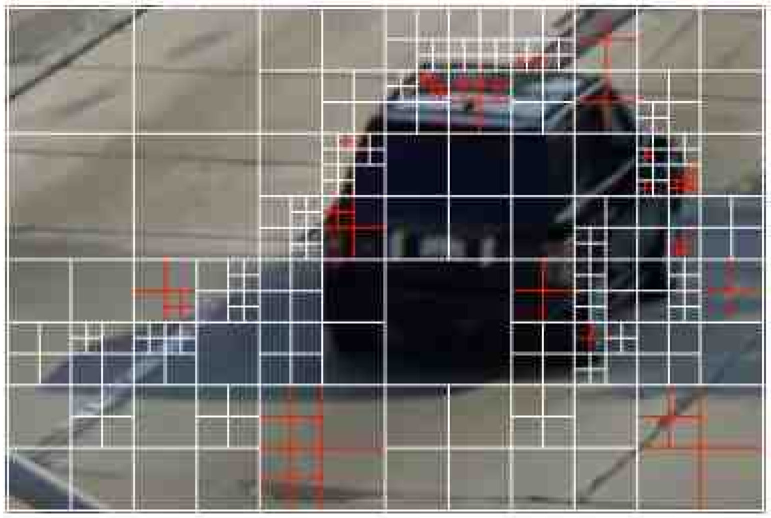


Fig. 1.1. Details of $4k \times 2k$ sequence Traffic showing the coding block in white and nested transform block in red structure after using quad-tree partitioning [12]

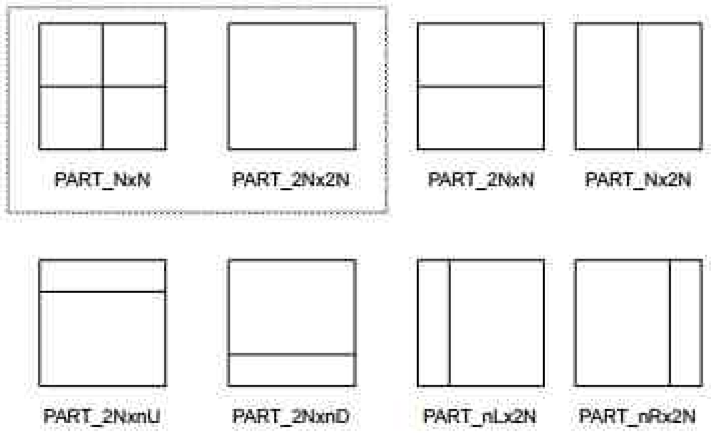


Fig. 1.2. Prediction block division has Inter-coded CUs using all modes while intra-coded CUs can use only the first two modes [9].

The HEVC encoder, uses simple tree-pruning algorithms to estimate the optimal partitioning in a distortion sense [10], [11]. The HEVC standard uses a block-based hybrid coding architecture, combining motion-compensated prediction and transform coding with high-efficiency entropy coding. In contrast to previous video coding standards, it applies a flexible quad-tree coding block partitioning structure that enables the efficient use of large and multiple sizes of coding blocks. In addition, HEVC employs developed intra prediction and coding, adaptive motion parameter prediction and coding, and improved version of Context-adaptive binary arithmetic coding (CABAC) entropy coding. Block-diagram of the HM encoder is shown in Figure 1.3. The input video is first divided into blocks called coding tree units (CTUs), that is broadly equivalent to that of macroblocks in previous standards. The coding unit (CU) applies a region using the same prediction mode (intra, inter or skip) and it is demonstrated by node leaf of a quadtree structure. The prediction unit (PU) uses a region sharing equal prediction information. The transform unit (TU), used by another quadtree, defines an area having the same transformation information [12].

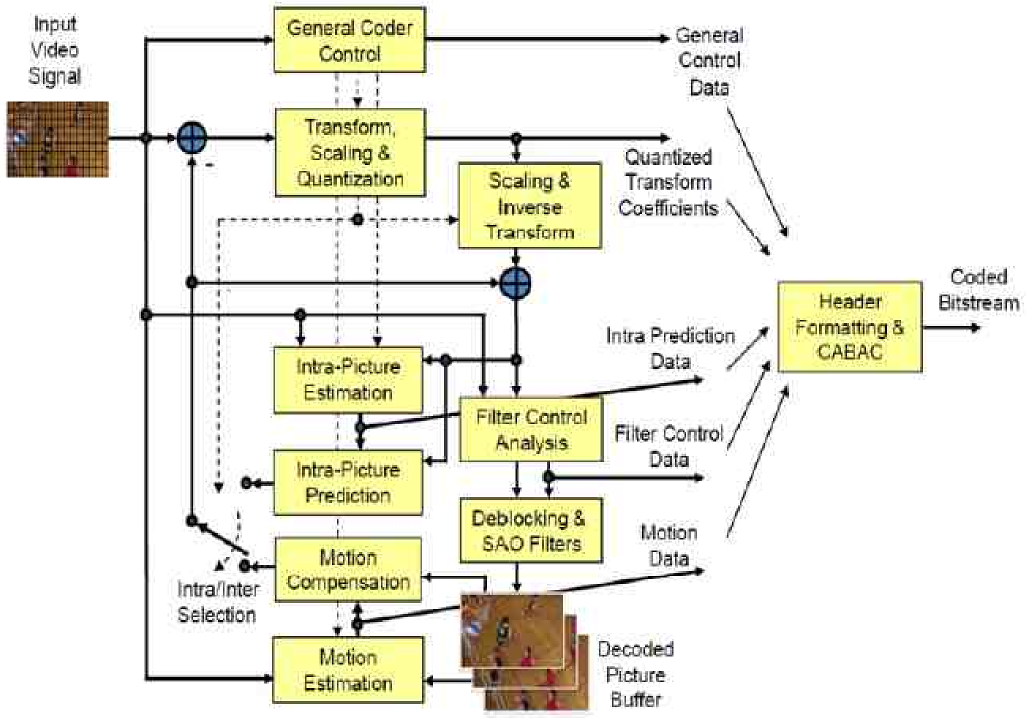


Fig. 1.3. Block Diagram of HEVC Encoder [9].

1.3 Video Coding Layer and Encoder of HEVC

The HEVC standard is designed for different purposes such as being efficient in coding process, ease of transport system integration and data loss resilience, as well as applying parallel processing design in system implementation. Later on, the main specifications of the design to reach the mentioned aims along with the encoding process that would generate the bitstream is described [9]. The video coding layer of HEVC applies the same hybrid concept (inter/intra picture prediction and 2-D transform coding) used in previous video compression standards since H.261. HEVC is a hybrid video encoder using both motion-compensated prediction and transform coding with high-efficiency entropy coding, creating a bit stream as shown in Figure 1.3. Encoding algorithm starts with:

First, divide picture into block-shaped areas, with the exact block partitioning would be signaled to the decoder. The first picture called the reference picture is coded using intra-picture prediction. Intra picture uses some prediction of data within the same picture (or random access point of frame) and it finds the closest region to the block.

Intra picture prediction is used to eliminate correlation within regions of the same picture. The main idea of intra prediction is that texture of a frame region is similar to the regions within the same frame.Intra prediction is applied when there is no picture for inter prediction or inter prediction is less efficient than intra prediction.

Remaining frames of a video or between random access points will be inter picture coded. Inter picture method is typically used for most blocks. Inter picture tries to find a block similar to the one encoding on a previously encoded pictures, referred to as a reference picture.

The basic idea of inter prediction is that a significant region of a picture content including objects moves in a scene. From frame to a frame only small changes between the scene content are seen due to motion. If this motion is used for prediction, the scene can be represented using motion vectors and a prediction error signal. The motion vectors shows how picture regions should move from the reference picture to the current picture to make the prediction. The prediction error signal has the part of the picture that was not possible to be described by the motion vector model.

The encoding process of inter-picture prediction includes choosing data related to the reference picture and motion vector (MV) to be done for predicting each block. The encoder and decoder apply exactly the same inter-picture prediction signals using motion compensation (MC) including the MV and mode decision data.

The best motion parameters are chosen and encoded using merge mode and adaptive motion vector prediction (AMVP) mode, in which motion predictors are selected and explicitly coded among several candidates.

For better efficiency of motion-compensated prediction, non-cascaded interpolation structure with 1D FIR filters are used. A filter with seven or eight taps is used to generate the samples of half-pel and quarter-pel luma samples. A 4-tap filter is applied for chroma interpolation.

Inter or intra prediction eliminates the correlation within frames or between frames of a video sequence. The difference between prediction signal and the the current block contains the regions of the original signal that could not be predicted using inter or intra prediction methods. Prediction stage helps to reduce the correlation of signals but still it can be compressed. The difference between the original block and its prediction (inter or intra prediction) is transformed using a linear spatial transform.

Transformation on the remaining part of residual signal makes the resulted coefficients to be less correlated. In this method, Karhunen-Loave Transform (KLT) is used for the optimum decorrelation and energy compaction of the residual signal [13].The transform coefficients are scaled and quantized. The quantization process is functions like a mapping to covert the signal amplitudes to a predefined set of quantized values.

Quantization is definitely a nonlinear- lossy operation that can not be inverted. Careful control setting of quantizer in a current frame and the coded frames in a video sequence the amount of removed unrelated information can be optimized.

Transform coefficients are entropy coded, and transmitted afterwards with the prediction information. In-loop filtering in encoder design is used to enhance the reconstruction quality of the picture for display. Since it is placed within a loop, the improvement affects both the quality of current output pictures and reference pictures for prediction unit. Therefore loop filtering has a strong effect on the overall performance of the video coding scheme.

From the decoder perspective after receiving bit stream, inverse scaling creates quantized transform coefficients. Next, quantized transform coefficients are inverse transformed to find the residual signal. The residual signal is then added to the prediction, and the result of that addition will be sent to into one or two loop filters to smooth out artifacts induced by block-wise processing and quantization.

The final result picture representation is stored in a picture buffer to be used for the prediction of further pictures. Video material to be encoded by HEVC is expected to be the input as progressive scan imagery. Context adaptive binary arithmetic coding (CABAC) is used for entropy coding. HEVC applies similar CABAC scheme to that of H.264/MPEG-4 AVC. Note that CABAC in HEVC has multiple improvements for its throughput speed (especially for parallel-processing coding designs) and its compression performance, and to decrease its memory requirements for implementation.

1.4 Overview of 3D HEVC as an Extension to HEVC Standard

Stereoscopic displays are among the one of many 3D displays that can be viewed with special glasses. In contrast, auto-stereoscopic displays emit view-dependent pixels and do not require glasses for viewing [14]. JCT-3V has developed 3D extension to HEVC to support auto-stereoscopic displays. PSNR is the relation of the mean squared error between the original and the reconstructed video.

Since the quality of 3D displays is essential for us and PSNR is a common used quality measure in video coding, we are interested to improve the PSNR in 3D-HEVC standard.

3D HEVC encoding process starts with splitting the signal into rectangular blocks called coding tree units (CTUs), predicted from previously decoded data. In encoding process, block transformation based on integer approximation of discrete cosine transformation, quantization and coding of the transform coefficients are applied. 3D-HEVC encoding process uses block-based hybrid coding architecture, combining motion-compensated prediction and transform coding with high-efficiency entropy coding. Therefore, visible discontinuities in the reconstructed images, known as blocking boundaries, at the block boundaries are formed. Block-transform coding of the prediction error followed by coarse quantization is the major reason of blocking artifacts [15]. One of the approaches to improve the blocking artifacts is in-loop filtering. Since the filter is placed within the loop, improving artifacts effects not only the quality of the output pictures but also the reference pictures for prediction when coding current pictures. Therefore, loop filters have an essential impact on the performance of the video coding scheme [8].

1.5 In Loop Filtering

In loop ltering is a part of HEVC standard to remove artifacts from an image. Two types of loop filter can be used: the first type uses linear filters on a region of a picture or a complete picture. In the selected region the filter can be applied as a convolution with the finite impulse response (FIR) in the spatial domain or multiplication in the frequency domain. Filter parameters can be defined on the encoder side according to optimization solution for filter coefficients such as using a Wiener Filter method. The second type is applied on a local spatial domain of a picture. Based on the local features the filter is applied on small set of samples. HEVC deblocking filter uses the same concept. In addition to H.264/AVC which includes an in-loop Deblocking Filter, HEVC employs a Deblocking Filter similar to the one used in H.264/AVC. A deblocking filter operates on block boundary structures that are made by block-wise motion compensated prediction and block-wise coding of residual signal. Quantization of residual results in visibility of edges in these blocks. In order to reduce the impact of deblocking filter is applied on a region across prediction and transform blocks. The strength of the debocking filter is based on the amount of blockiness in the boundaries.

In loop filtering in 3D-HEVC is the same as HEVC adding SAO and ALF as two new filters compared to H.264/AVC. These filters reduce the block artifacts caused by prediction, transform, and quantization process in coding. Since, in-loop filtering is used as a feedback for the reference picture, reference pictures will have better quality for motion- compensated prediction.

In this thesis, an adaptive loop filtering in picture based for both depth and texture view used for loop filtering and their PSNRs is reviewed. Adding ALF filter, not only improves the average PSNR in luminance component of texture and depth but also PSNR improvement in chrominance components is observed which results in better quality compared to anchor test model.

On the other hand, a fast bilateral filter in picture based for texture view is implemented and added to in-loop filtering block to reduce the artifacts in encoding process. This implementation is discussed in detailed in next chapter and it is recommended to apply both Adaptive Loop Filter and Fast Bilateral Filter together for an optimal solution of running time and quality in 3D HEVC.

1.6 Deblocking Filter

In encoding process, the input video is first divided into blocks called coding tree units (CTUs), that is broadly equivalent to that of macroblocks in previous standards. The coding unit (CU) applies a region using the same prediction mode (intra, inter or skip) and it is demonstrated by node leaf of a quadtree structure. The prediction unit (PU) uses a region sharing equal prediction information. The transform unit (TU), used by another quadtree, defines an area having the same transformation information [12]. Blocking artifacts are known as objectionable visible discontinuities. To remove these blocks and improve the quality of the video, adaptive low-pass filters are applied relative to the boundary level of change. Block based boundaries are caused by different types of blocks such as TUs, CUs and PUs. Deblocking filter is based on a decision made to turn on or off. After deciding to use the filter, encoder applies strong or weak filtering based on the pixel gradients over the boundaries. Thresholds to determine whether to apply the filter or not are based on Quantization Parameter table.

1.7 Aim of the Thesis

The main purpose of this thesis is to reduce the processing time of 3D HEVC encoder by implementing Fast Bilateral Filter and improve the video quality by implementing Adaptive Loop Filter. A second aim is to evaluate the impact of proposed algorithm on the quality and changes of running time of the output and compare their results to the results of the original algorithm of 3D HEVC.

In this thesis, we first implement adaptive loop filter in picture based in loop filtering design. PSNR and running time of proposed method is evaluated in the first section of Chapter two. We will see that using ALF improves the quality (that is defined by PSNR) and increases the running time. In the second section, we implement fast bilateral filter with adaptive loop filter to decrease the running time of encoding compared to implementation of ALF alone. Results demonstrate the improvement of quality compared the anchor test model. We will show that this implementation is an optimal design for in loop filtering block in the sense of quality and running time.

Note that improving the quality of a video is not costless, cost of this implementation is higher bitrate and more bandwidth for transmitting the signal. In other words, bitrate control what is related to video quality and the file size.

Implemented algorithm is evaluated relative to bitrate changes and the quality versus the bitrate is demonstrated in graphs and efficiency of mentioned algorithm is discussed in results section.

2. DESIGN DESCRIPTION OF FILTERS

2.1 Overview of Adaptive Loop Filter in HEVC

2.1.1 ALF Core Techniques

This section describes ALF core techniques employed in HTM 6 in 3D-HEVC, the filter shape of ALF is a combination of nine-7-tap cross shape and three-3-tap rectangular shape, as showed in Figure 2.2. Every square in Figure 2.2 is equivalent to a pixel. In other words, 19 pixels are used to form a filtered shape with corresponding value for the sample of 9 pixel positions, and coefficient coding. As can be seen from Figure 2.2, the filter shape is symmetric which results in finding the coefficients for 9 samples and use the same coefficients for the remaining samples, except the middle sample, that will decrease the number of coefficient calculation to half [16]. Other ALF shapes are discussed in [17], where the concept of using diamond filters is shown. Diamond shapes can be selected among 5×5, 7×7 or 9×7 depending on the rate distortion cost [17]. In other words, 13 to 39 filter coefficients need to be calculated using diamond shape, while implemented symmetric filter shape in Figure 2.2 only requires 9 filter coefficients. This filter shape reduces the number of multiplications and complexity significantly resulting in decrease of running time compared to using diamond shapes.

In our implementation, after deblocking filter and sample adaptive offset filters, ALF is applied to the reconstructed signal. The filter is adaptive because filter coefficients are signaled in the bitstream. In the coding process, coefficients are calculated based on the original picture content an distortion of the reconstructed image [18]. The Wiener filter minimizes the mean square error between the desired images, also known as the original image, and the filtered image, which is the input of the ALF filter, leading to higher PSNR. Figure 2.1.1 shows the implementation of ALF in HEVC encoder. ALF filter is applied after Deblocking Filter and Sample Adaptive Offset Filter to remove artifacts. Since the output of ALF filter will affect the encoding process, improving PSNR in an image will result in higher PSNR of the video sequence.

ALF can be used for the sequence of coded pictures of a view which is referred to as the texture view and/or corresponding depth information referred to as depth view. The ALF can be used in picture based level in which picture-level filter coefficients are in a picture-level header.

In the picture-based ALF, filter coefficients are derived using the current picture [19]. Although this encoding process needs to access the entire current picture multipasses to derive the best coefficients, which leads to slower process of filtering the image compared to LCU-based filtering. On the other hand, results shown better improvement in PSNR for picture based ALF compared to LCU-based method. The following setting are applied to the ALF for the highest PSNR achieved by 3D-HEVC. First, ALF LCU-based or picture-based are to be picture based. Second, ALF is enabled for both texture and depth view.

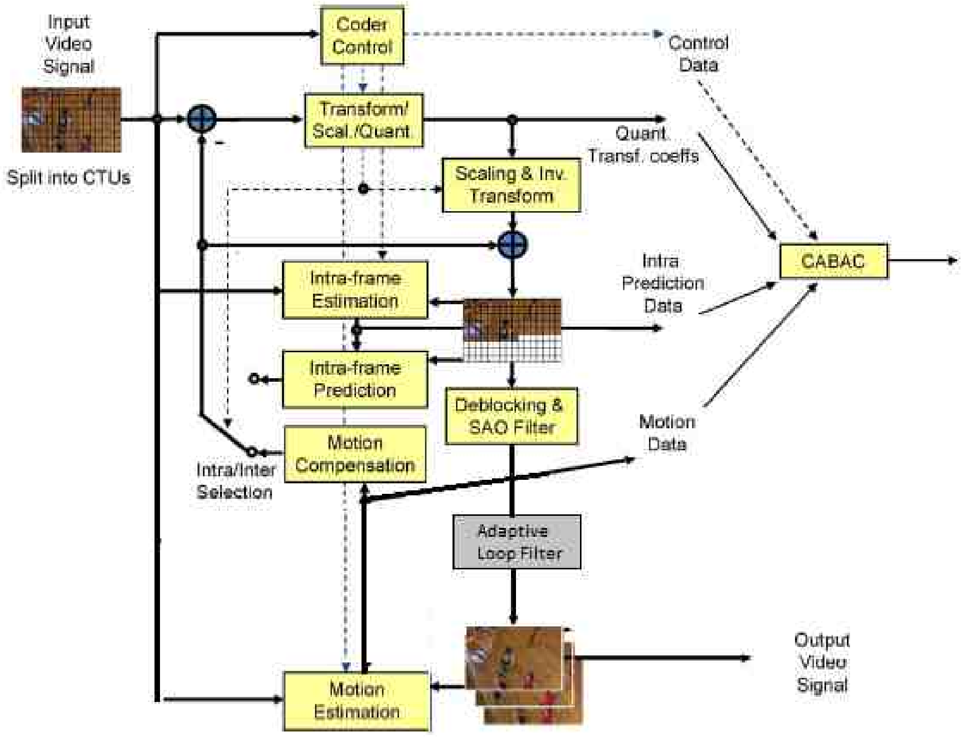


Fig. 2.1. Block Diagram of HEVC Encoder with ALF filter [9]

2.1.2 Wiener Filter Equation

To implement ALF filter, several notations are introduced:

$$\text{Filtered signal: } f[k], \text{ where } k = (x, y) \text{ is a pixel position} \tag{2.1}$$

$$\text{Original signal: } t[k] \tag{2.2}$$

$$\text{FIR filter with N coefficients: } \{h_0, h_1, h_2, \ldots, h_{N-1}\} \tag{2.3}$$

In order to form the filter shape of Figure 2.2 on the to-be-filtered signal, an offset variable k_i in Equation 2.4 is defined.

$$\text{Filter shape: } \{k_0, k_1, k_2, \ldots, k_{N-1}\}, \text{ where } k_i = (x_i, y_i) \tag{2.4}$$

Filtered signal is the result of multiplying the filter coefficients with the filtered shape of the image.

$$\text{Filtered signal: } f[k] = \sum_{i=0}^{N-1} h_i t[k + k_i] \tag{2.5}$$

The Wiener filter deals with Minimum Sum of Squared error between the desired signal and the filtered signal, SSE equation can be written as:

$$\text{Sum of squared error: } SSE = \sum_k \{f[k] - d[k]\}^2 \tag{2.6}$$

After substituting $f[k]$ in Equation 2.6, SSE is:

$$SSE = \sum_k \sum_{i=0}^{N-1} (h_i \cdot t[k + k_i] - d[k])^2 \tag{2.7}$$

In order to minimize SSE, the partial derivatives of SSE with respect to h_i are calculated and the derivatives will equalized to zero in Equation 2.7. That is,

$$\frac{\partial (SSE)}{\partial (h_i)} = 0$$

So this implies,

$$\frac{\partial \left(\sum_k \left(\sum_{j=0}^{N-1} (h_j t[k + k_j] - d[k])^2 \right) \right)}{\partial (h_i)} = 0.$$

Hence,

$$\sum_k \left(\sum_{j=0}^{N-1} \sum_{i=0}^{N-1} 2t[k+k_j](h_i t[k+k_i] - d[k])^2 \right) = 0. \tag{2.8}$$

Substitution in Equation 2.8,

$$\sum_k \left(\sum_{j=0}^{N-1} \sum_{i=0}^{N-1} t[k+k_j]t[k+k_i] \right) = \sum_k (\sum_{j=0}^{N-1} t[k+k_j]d[k]) \tag{2.9}$$

Considering index i as the number of columns and j as the number of rows, Equation 2.9 can be written as a multiplication of matrixes:

$$T \cdot \vec{h} = \vec{D} \tag{2.10}$$

where $T = [a_{ij}]$, $a_{ij} = \sum_k (t[k+k_j]t[k+k_i])$, $\vec{h} = [h_j]$ and $\vec{D} = \sum_k t[k+k_j]d[k]$
As can be seen from Figure 2.2, there are 19 filter coefficients which will be replaced by h_j s . These coefficients are called c_j s and j ranges from 0 to 18 . Since the filter coefficients are symmetric, we need to find c_0 to c_9 by column elimination,

$$T \cdot \vec{c} = \vec{D} \tag{2.11}$$

where $T = [a_{ij}]$, $a_{ij} = \sum_k (t[k+k_j]t[k+k_i])$, $\vec{h} = [h_j]$, $\vec{D} = \sum_k t[k+k_j]d[k]$ and $i, j = \{0, 1, \cdots, 9\}$.
The Wiener-Hopf equations in a matrix form can be derived, as shown in Equation 2.11. In this equation, there are two terms in the left side of equation. The first term is the auto-correlation matrix showed as T related to the to-be-filtered signal. Second term is the coefficient vector showed as $\vec{c}$ in Equation 2.11. The right side of Equation 2.11 is the cross-correlation vector showed as $\vec{D}$. Cross Correlation between the original signal and the to-be-filtered signal. In order to calculate the optimal filter coefficients, Gaussian elimination method is used to solve Equation 2.11 [19].

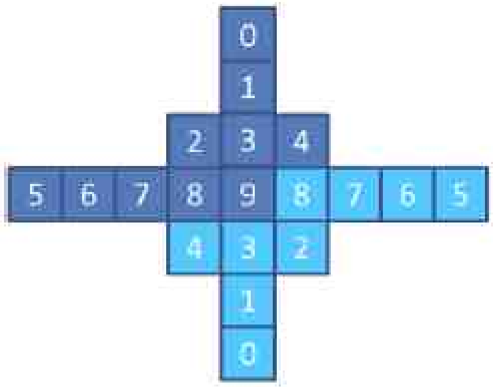


Fig. 2.2. ALF filter shape in HM 6 test model [19]

2.1.3 Virtual Boundary Processing

As can be seen from Figure 2.2, ALF filter shape consist of 9×7 cross with 3×3 square. Therefore, ALF is formed from 7 taps in the vertical and 9 taps in the horizontal direction, with a total of nineteen taps for this filter. In other words, the memory bandwidth requires 19 times of the number of samples in one image in the case that is no local buffer. For ALF implementation on chip, to prevent the increase in memory bandwidth, six line buffers can be invested. Note that line buffers need large chip areas design, especially for high resolution videos. To avoid the implementation of large chips, virtual boundary (VB) processing [20] is introduced for ALF. The idea of VB processing is the fact that during filtering one pixel on one region of a VB, pixels on the other region of the virtual boundary cannot be used. Meaning that, if ALF filter is placed at the center of the pixel to be filtered, passes a VB, process of filtering will be modified to prevent applying pixels on the other side of the virtual boundary. The concept of virtual boundary and ALF filtering process are shown in the Figure 2.3. C9 is the coefficient for the to-be-filtered pixel in the original image and it is in the center of ALF filter. Virtual boundary is shown in the figure as a red line in the center. In this figure, C0 to C9 are original filter coefficients and C0'-C9 are updated filter coefficients after passing the VB. In the case that ALF is filtering a pixel and it is not crossing the VB, original filter coefficients (C0-C9) are used to filter the pixels. On the other hand, assuming that adaptive loop filter shape of the to-be-filtered pixel crosses the virtual boundary, updated filter coefficients (C0-C9) will be calculated. Updated filter coefficients are chosen depending on placement of the to-be-filtered pixel in the original image and will be applied in the filtering process. Mentioned method does not need local line buffer in pixel filtering process. Considering the deblocking filter in HEVC, virtual boundary of luma component is located 4 pixels above the horizontal LCU boundary. Moreover, the virtual boundary of the the chroma component is placed 2 pixels above the horizontal LCU boundary [19].

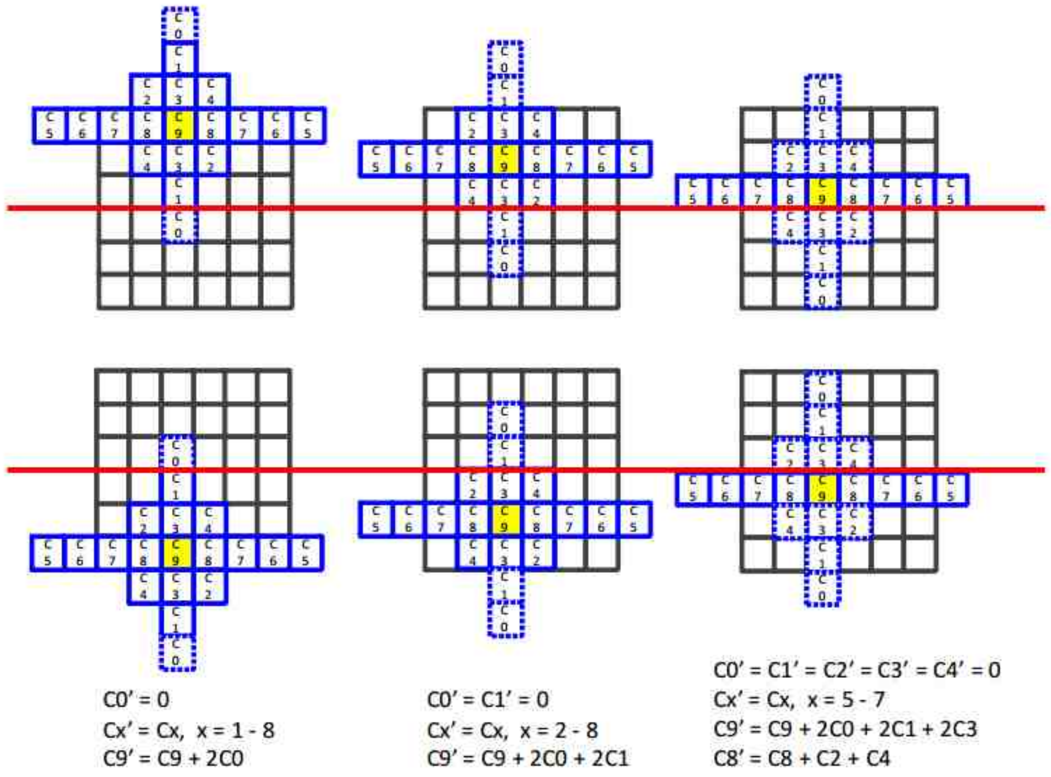


Fig. 2.3. Virtual Boundary Processing of ALF [19]

2.1.4 Syntax Design

Information to filter coefficient parameters and data related to signaling on or off the filter are coded and transmitted through with the bitstream. As can be seen in Figure 2.4, picture level header known as APS contains the information related to the filter coefficient parameters. Slice data stores the data related to the filter on/off control flags with LCUs. Since the filter coefficient parameters should be signaled for both luminance and chrominance components, picture level on and off control flags along with the number of luma filters and the calculated adaptive loop filter coefficients are stored in the coefficient parameters. Maximum of sixteen luma filters, one chroma component Cb filter, and one Cr filter in each image can be signaled. Adaptive loop filter on or off control flags are applied for better choice of adaptation depending on the situation to use the ALF filter or not. Furthermore, on or off control flags on slice-level or LCU level filter can be used. Slice header includes filter on or off control flags for luminance and two components of chrominance. In some cases, the referenced APS of the slice is lost which is considered as a slice parsing problem. In order to solve this challenge, signaling slice level filter on or off control flag is applied where the coding system will decide to apply ALF or not [21]. Now we consider two modes of slice level on or off flag. While the flag is signaled to be on, LCU-level filter on/off control flags are interleaved in slice data and coded with LCUs. If the flag is signaled as off, LCU-level filter on/off control flags will not be coded and all slice LCUs are considered off for adaptive loop filter [19].

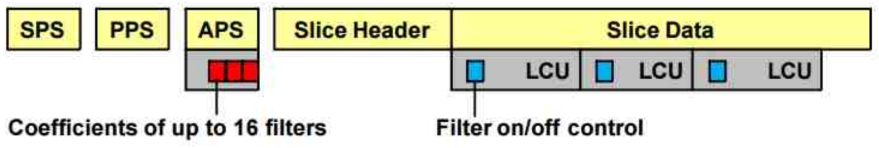


Fig. 2.4. Locations of ALF parameters in the bitstream [19]

2.1.5 Filter Coefficients

As discussed previously, filter coefficients are calculated by solving Wiener-Hopf in Equation 2.11. These coefficients are floating numbers, and they can have a large range. Note that implementation of a variable with a floating point value with a large range in syntax design is difficult [22]. To avoid this difficulty, the filter coefficient fractional part and the filter coefficient range must limit to a certain number. As an example, filter coefficients in the fractional part is limited to 8-bits in HEVC HM-7.0. Therefore calculated filter coefficients are quantized with maximum of 8 bits. For non-center filter coefficients (i.e. coefficients other than C_9 the filter coefficient) range in $[-1.0, 1.0)$. For the filter coefficient located in the center i.e. C_9 usually has larger value than the other coefficients ranges in $[0.0, 2.0)$. Considering mentioned two conditions, filter coefficients can be showed using integers in software implementations. For hardware applications, a single ten bit register is sufficient to save and process a single filter coefficient. Note that coefficients near the center position can possibly have large values, when compared to others. Since correlation between neighbor pixels is higher compared to further coefficients, larger values for near coefficients are calculated and chosen. Consider that reduction of C_9 distribution can be achieved using prediction calculated by the other coefficients applying Equation 2.12 [19]:

$$\Delta C_9 = C_9(2^8 - \sum_{j=0}^{8} 2C_j) \tag{2.12}$$

Where ΔC_9 shows its prediction error, and 28 is representation of 1.0 with 8-bit fractional precision.

To exploit different distributions of coefficients, filter coefficients are calculated by applying K^{th} order Exponential Golomb codes. As shown in Table 2.1.5, different K values are chosen for different ALF coefficients. The shortest codeword for zero value of coefficient is chosen. The second shortest codeword for coefficient magnitude one is chosen, and so on. Each sign related to non zero coefficient is demonstrated by one sign bit.

Table 2.1.
K Values For Different Filter Coefficients

Filter Coefficient	Coefficient Value	K
C_0	-256 to 255	1
C_1	-256 to 255	2
C_2	-256 to 255	3
C_3	-256 to 255	4
C_4	-256 to 255	3
C_5	-256 to 255	1
C_6	-256 to 255	3
C_7	-256 to 255	3
C_8	-256 to 255	5
C_9	0 to 511	0

2.1.6 Testing Adaptive Loop Filter in 3D-HEVC

ALF was an option for 3D HEVC in test model 6. However, it was removed after version HM-7. In this thesis, we will see the impact of using ALF in quality of a video by measuring and comparing PSNR of a video using modified test model and anchor test model. The performance of the proposed method is illustrated under different bases on the resulting image quality and changes in processing time. The peak signals to noise ratio shows the quality of decoded video relative to the original video, and PSNR is defined as:

$$PSNR = 10\log_{10}\left(\frac{R^2}{MSE}\right)$$

Where R is the maximum fluctuation in the original picture. As an example, if the input is a floating point picture type, then R is one. In the case of 8-bit unsigned integer picture type, R is 25, and so on.

The *MSE* is:

$$MSE = \frac{\sum_m \sum_n \mid I_1(m,n) - I_2(m,n) \mid^2}{MN}$$

M and N are the number of rows and columns in the input images, I_1 and I_2 shows the original and filtered signal.

For implementation of filters, we used HM 6.2.The aim of the HM encoder is generally to employ a reference for HEVC encoder implementation. In order to test the technologies and develop the encoder or decoder, HM encoder is beneficial as a test platform.

The HM, implemented in C++, is not for providing commercialized HEVC encoder. The HM encoder runs significantly slowly and it needs a large computer cluster for implementation. Note that the HM encoder possibly runs slower than ideal in a lot of cases. Recently, some encoding time considerations have been received while developing HM software.

Previously, version 0.7 of HM required maximum of 100 hours to encode a 10 second video. After improving the running time, up to 20 hours reduction was observed in later versions. The HM encoder is built based on a rate-distortion quantization process optimization. The JCT-VC common test conditions introduced a few configurations for encoder to be applied in tests. These configurations are [12]:

All intra (AI)

All images are encoded with I slices.

Random access (RA)

A pyramidal structure with a randomly selected picture approximately every one second is used as a picture reordering. Mentioned configuration simulates what we use in a broadcasting environment.

Low-delay with B slices (LB)

In this mode, just the first frame is encoded using I slices and random selection of pictures and picture reordering are not used. This configuration simulates what we use in a videoconferencing environment.

The 3D video test set consisted of 4 sequences with the video component in 4:2:0 chroma format: 4 with a progressive HD resolution of 1024 by 768 luma/depth samples with 30 fps. Multiview test sequences called: Balloons, Kendo, Newspaper and Lovebird1 are shown in table 2.1.6 and their features are described.

Table 2.2.
Multi view test sequences with feature description [23]

Sequence Name	Resolution	Frame Rate	Sequence Length	Video Format	Stereo Type	Description
Balloons	1024×768	30 Hz	500 frames	YUV4:2:0 uncoded	multiview, 7 cameras with 5cm spacing	indoor, studio type, complex object motion (reflections, transparency), moving camera, medium detail, medium complex depth structure, studio light
Kendo	1024×768	30 Hz	400 frames	YUV4:2:0, uncoded	multiview, 7 cameras with 5cm spacing	Indoor, studio type, complex object motion (fog, reflections, transparency), moving camera, high detail, medium complex depth structure, studio light
Newspaper	XGA 1024×768	30 Hz	300 frames	YUV4:2:0, uncoded	multiview, 9 cameras with stereo distance	Indoor, studio, soap opera type of scene, simple object motion, no camera motion, high detail, medium complex depth structure, studio light
Lovebird1	XGA 1024×768	30 Hz	300 frames	YUV4:2:0, uncoded	multiview, 12 cameras with stereo distance	Outdoor, quiet dialogue in movie type of scene, simple object motion, no camera motion, high detail, complex depth structure, natural light

2.1.7 Results

All 4 sequences were evaluated in a test scenario: with ALF implementation in picture-based level, video and depth component of 3 views {V0, V1, V2} were coded and the changes in PSNR is evaluated. Test model of HTM 6.2 was used for ALF implementation. Tables 2.3, 2.4, 2.5, and 2.6 show the results of average improvement in PSNR of Y, U and V components using adaptive loop filtering method in picture-based for both depth and texture view.

In this implementation, Inter-view is restricted to the given QP for the independent view and $QP + 3$ for all dependent views. The QP offset ΔQPD for the depth QPD in relation to the Video QP ($QPD = QP + \Delta QPD$) was fixed based on subjective assessments and varies from $\Delta QPD = 0$ for video $QP = 51$ at lowest quality up to $\Delta QPD = 9$ for video $QP \leq 32$ at high quality. With these settings, the same encoder configuration is used for all sequences and rate points [24]. Different value of QP was tested for texture inter-view including 30, 35 and 40 with corresponding inter-view QP depth of 39, 42 and 45.

Luminance(Y component) is a weighted sum of the three colors of light used in color displays such as televisions which are Red, Green and Blue and it is related to the brightness of a video. In scientific terms, the brightness of light is measured in terms of luminance. Chrominance (U and V components) is the signal used in video systems to show the color information of the picture. Research studies shows that the human eye and brain are more sensitive to changes in brightness detail than color [25].

This concept means that improvement in luminance component is more important than chrominance and will have more effective for human visual system. ALF filter shows higher improvement in luminance component compared to chrominance. Results show up to 2.1 dB improvement in depth and 0.2 dB improvement in texture of luminance component of test sequences.

In all test sequences, ALF has better improvement in luminance component of depth compared to the texture which concludes that ALF is useful for the applications that accuracy of depth map is important and ALF can significantly enhance the depth of the video.

There is a question of why ALF can improve the quality of a video. Recall formula of PSNR defined in section 2.1.6. There is a logarithmic relation between PSNR and *MSE* of original and filtered image. A decrease in *MSE* results in an increase in logarithmic increase in PSNR. Previously, we discussed that the main purpose of ALF filter, which is based on Wiener Filter, is to minimize *MSE*. Therefore, it makes sense that we observe an increase in PSNR of a video using ALF.

The Balloons video sequence has less details compared to Kendo, Newspaper and Lovebird1. We see that the improvement in Balloons texture is more significant, which makes sense because we more details leads to having more edges and higher frequencies in an image and it is more difficult for a filter to reduce *MSE* between original and filtered image with higher details. So generally, ALF can be more efficient in the sense of higher quality and less running time changes for less detailed video sequences.

Table 2.3 shows that Balloons sequence with medium details can reach up to 0.7 dB improvement in luminance depth and 0.2 dB improvement in texture. On the other hand, chrominance component shows maximum of 0.1 dB improvement in all tested QPs. Also, average bitrate changes shows maximum of 13% and minimum of 0.4% increase which is the result of improving the quality of videos. As can be seen, encoder running time increases in Balloons sequence by increasing QP. The lowest encoding time with the highest PSNR for Ballooons in Table 2.3 is observed while inter-view QP of texture equals to 35 and QP in depth equals to 42. Balloons has the lowest increase in running time among all the four sequences because of less details.

Now we will take a closer and detailed look at the changes of PSNR relative to the changes of bitrate. Graph 2.5 shows average of Y-PSNR on Y-axis and average bitrate in X-axis. We discussed previously that three videos (Video1, Video2, Video3)

Table 2.3.
Average PSNR changes in Luminance and Chrominance components with different QP Values for Balloons video sequence

	Balloons					
	Y (dB)	U (dB)	V (dB)	Inter-view QP	Bitrate %	Encoder Time %
Texture	0.24	0.07	0.07	30	0.4%	1%
Depth	0.72	0	0	39	5.5%	
Texture	0.24	0.06	0.1	35	0.8%	1%
Depth	0.49	0	0	42	6.3%	
Texture	0.17	0.12	0.12	40	12.12%	4%
Depth	0.67	0	0	45	13.39%	

among the multi-view videos were encoded with their corresponding depths (Depth1, Depth2, Depth3). 6 sub-graphs in 2.5 show higher quality while using ALF filter compared to the anchor test model.

Comparison of subjective video quality ALF in Balloons video sequence is showed in the Figure 2.6 with corresponding depth view in Figure 2.7. In this experiment the inter-view QP is 30 for texture and 39 for the depth and results of encoded images can be seen. Kendo video sequence in 2.4 shows that encoder running time can vary from 1 to 4 percent and highest PSNR and lowest running time is observed while QP equals to 35. As we discussed earlier, since Kendo has higher details compared to Balloons, less PSNR improvement is observed. Bitrate in depth view of Kendo test sequence is increased up to 27 percent in and minimum os 0.1 percent is observed in texture view. Detailed changes of PSNR relative to the changes of bitrate is shown in Figure 2.8. Results indicate that when bitrate is not very high or very low the quality improvement can be in maximum value and higher improvement is shown compared to the video but note that higher PSNR improvement results in higher bitrate. Comparison of subjective video quality ALF in Kendo video sequence is showed in the Figures 2.9 with their corresponding depth view in Figure 2.10 while inter-view QP is 30 for the texture and 39 for the depth.

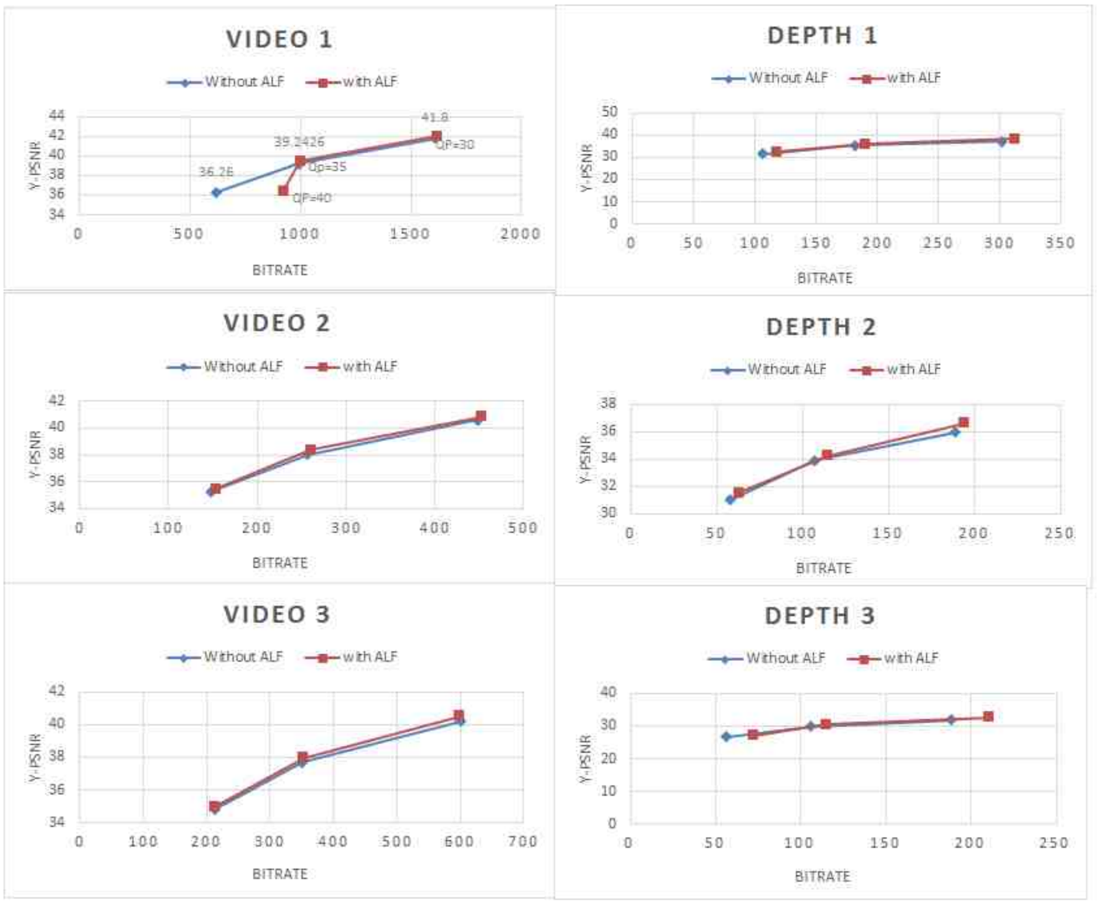


Fig. 2.5. Average PSNR in three encoded videos with corresponding depth views relative to bitrate for Balloons video sequence

Lovebird1 in table 2.5 the highest improvement of Y PSNR in depth among all the test sequences. Approximately, 0.1 dB improvement in luminance of texture is seen. Highest quality improvement is reached for inter-view QP texture of 35 and depth QP of 42. Lovebird1 shows the lower Detailed changes of PSNR relative to the changes of bitrate is shown in Figure 2.11. Results indicate that when bitrate is not very high or very low the quality improvement can be in maximum value and higher improvement is shown compared to the video but note that higher PSNR improvement results in higher bitrate. Comparison of subjective video quality ALF in Lovebird1 video sequence is showed in the Figures 2.12 with their corresponding depth view in Figure

Table 2.4.
Average PSNR changes in Luminance and Chrominance components with different QP Values for Kendo video sequence

	Kendo					
	Y (dB)	U (dB)	V (dB)	Inter-view QP	Bitrate %	Encoder Time %
Texture	0.15	0.03	0.07	30	0.1%	1%
Depth	0.72	0	0	39	13.6%	
Texture	0.19	0.06	0.08	35	1.6%	1%
Depth	0.8	0	0	42	14.7%	
Texture	0.09	0	0.1	40	0.2%	4%
Depth	1.9	0	0	45	27.2%	

Table 2.5.
Average PSNR changes in Luminance and Chrominance components with different QP Values for Lovebird1 video sequence

	Lovebird1					
	Y (dB)	U (dB)	V (dB)	Inter-view QP	Bitrate %	Encoder Time %
Texture	0.16	0.04	0.05	30	0.34%	2%
Depth	1.04	0	0	39	1.7%	
Texture	0.16	0.06	0.08	35	1.2%	3%
Depth	2.3	0	0	42	8.9%	
Texture	0.13	0.06	0.03	40	1.7%	2%
Depth	2.1	0	0	45	8.3%	

2.13 while inter-view QP is 30 for the texture and 39 for the depth where one of thee coded videos is shown as an example. The last detailed video test sequence is shown in Table 2.6. Newspaper video sequence reaches the highest quality while inter-view QP texture of 35 and depth QP of 42. Bitrate percentage increases from 0.1 percent in texture to 12 percent in depth view. Detailed changes of PSNR relative to the changes of bitrate is shown in Figure 2.14. Results indicate that not very significant Y-PSNR improvement compared to the previous video test sequences is achieved because this

Table 2.6.
Average PSNR changes in Luminance and Chrominance components with different QP Values for Newspaper video sequence

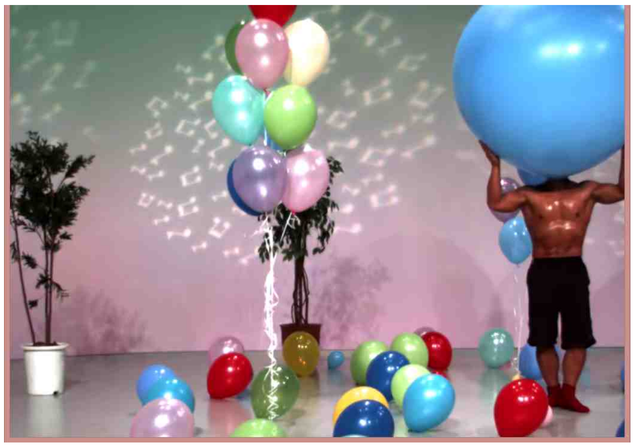
	Newspaper					
	Y (dB)	U (dB)	V (dB)	Inter-view QP	Bitrate %	Encoder Time %
Texture	0.15	0.05	0.05	30	0.1%	2%
Depth	1.79	0	0	39	7.3%	
Texture	0.15	0.04	0.05	35	0.08%	5%
Depth	2.0	0	0	42	10.3%	
Texture	0.12	0.06	0.05	40	3.6%	2%
Depth	0.5	0	0	45	12.2%	

test sequence is more detailed than the previous sequences. A significant change is observed for the depth of third coded video while for a constant bitrate around 150 bits/second around 2.5 dB improvement is reached. Comparison of subjective video quality ALF in Balloons video sequence is demonstrated in the Figure 2.15 with the depth view in Figure 2.16 while inter-view QP is 30 for the texture and 39 for the depth where one of thee coded videos is shown as an example. QP is a quantization parameter to quantize transform coefficients in encoder. The higher the value of QP, the higher the compression and distortion. We can see that for higher QPs, there will be more distortion in images, so using ALF will not improve the quality of video.

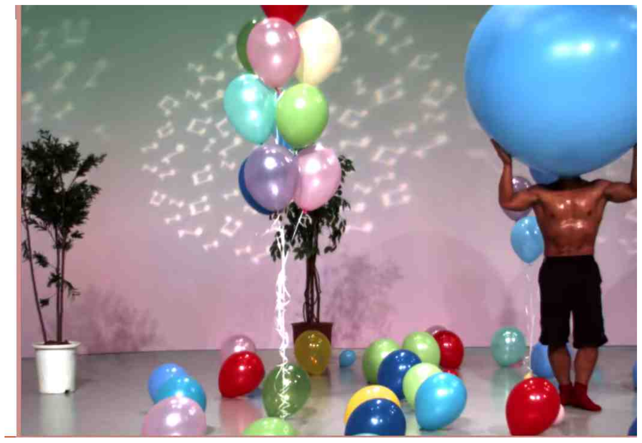
All four test sequences show that best Quantization Parameter for the highest PSNR is while inter-view QP texture is 35 and depth QP is 42. As can be seen, where the general encoder running time increased by 1 to 5%, showing that picture-based ALF can make the process of encoding slower but improve PSNR in luminance and chrominance components.

In the next section we propose an algorithm to make encoding process faster and more efficient for the applications that quality and time processing is an important issue.

(a) Input picture of Balloons test sequence.

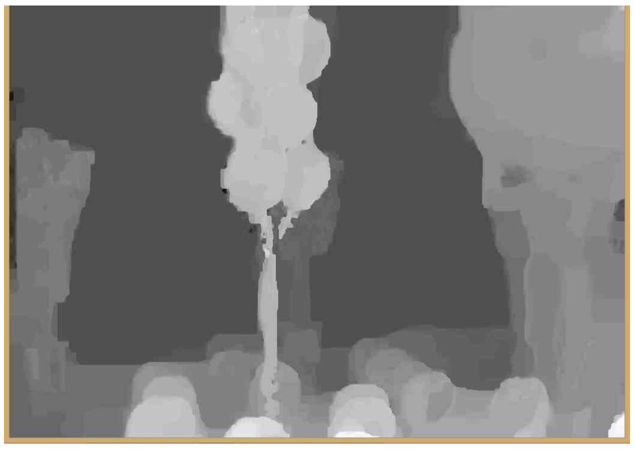


(b) Reconstructed picture using ALF.

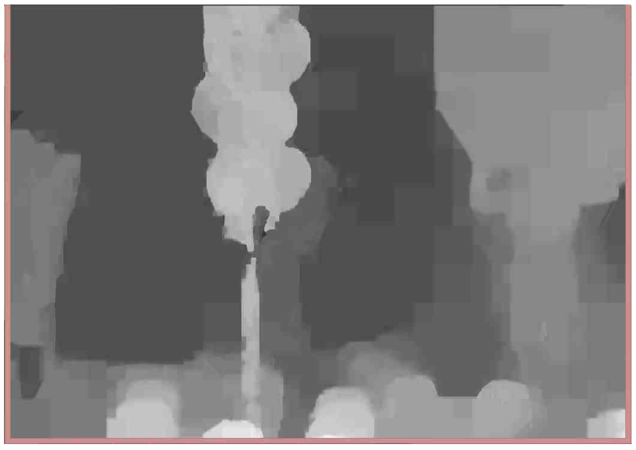


(c) Reconstructed picture using original software.

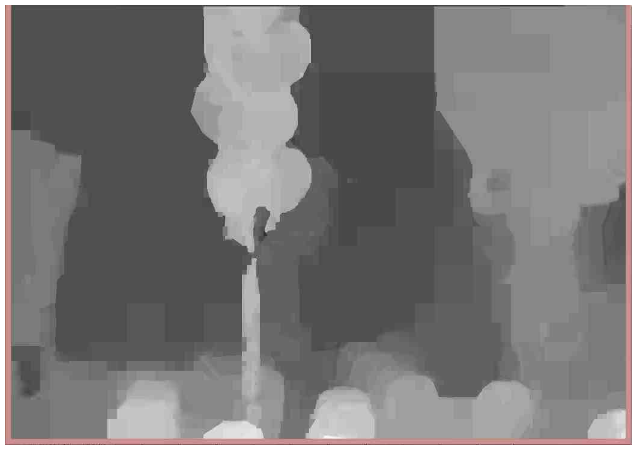
Fig. 2.6. Comparison of subjective video quality ALF in Balloons video sequence.



(a) Input depth view of Balloons test sequence.



(b) Reconstructed depth view using ALF.



(c) Reconstructed depth view using original software.

Fig. 2.7. Comparison of subjective depth view quality ALF in Balloons video sequence.

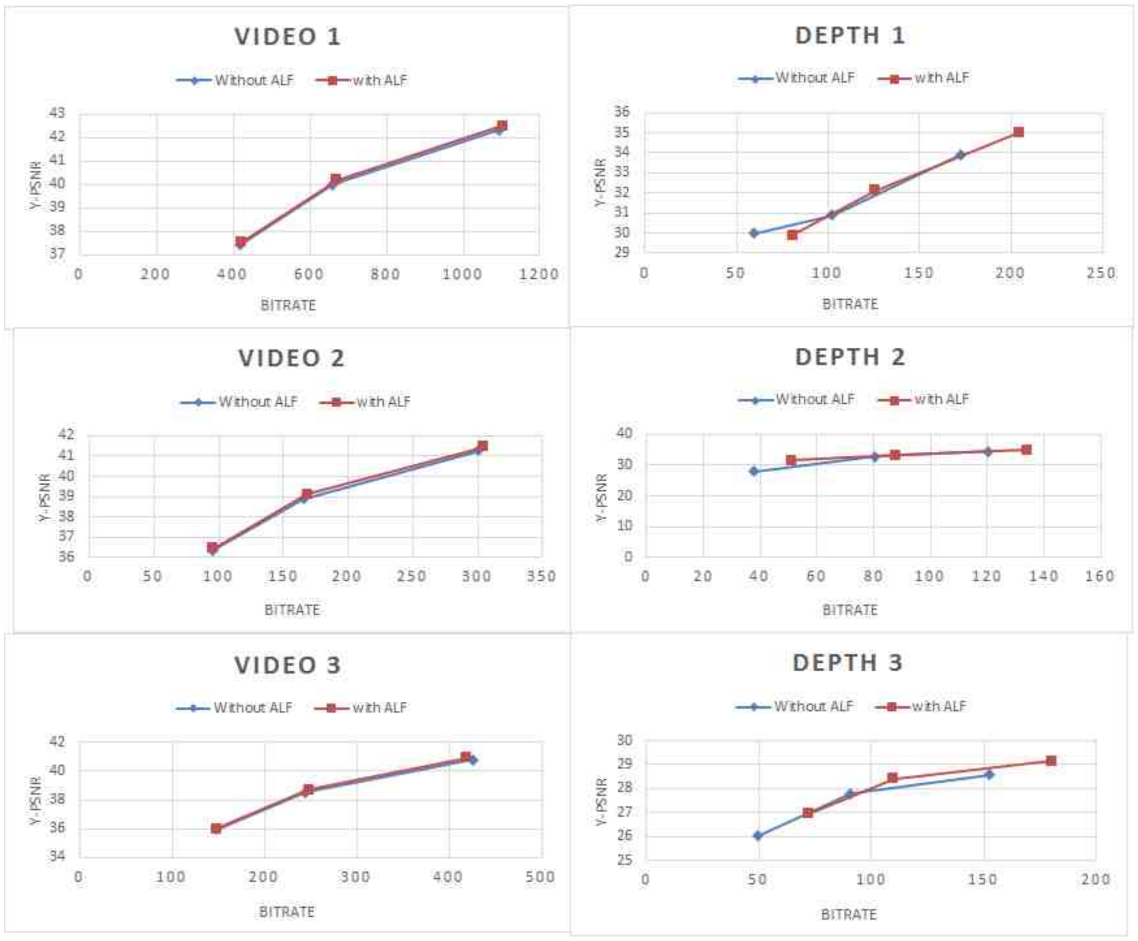
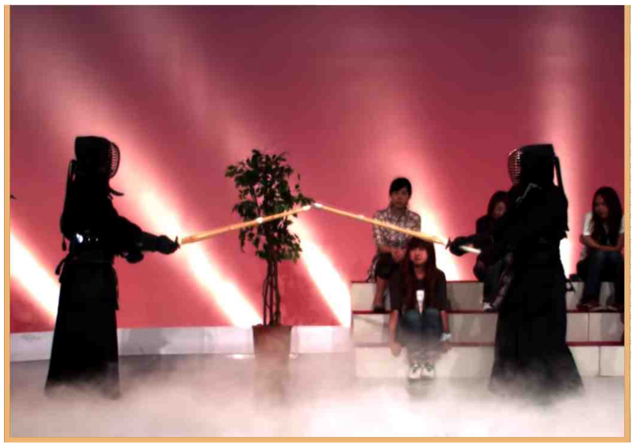
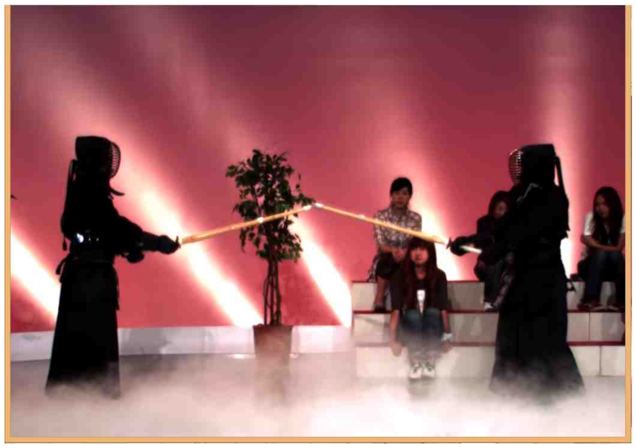


Fig. 2.8. Average PSNR in three encoded videos with corresponding depth views relative to bitrate for Kendo video sequence

(a) Input picture of Kendo test sequence.

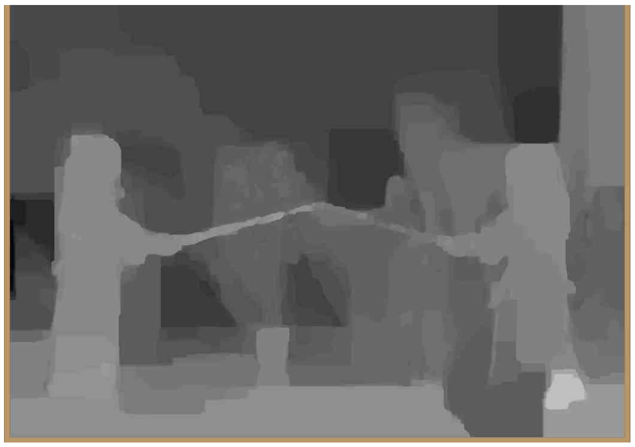


(b) Reconstructed picture using ALF.

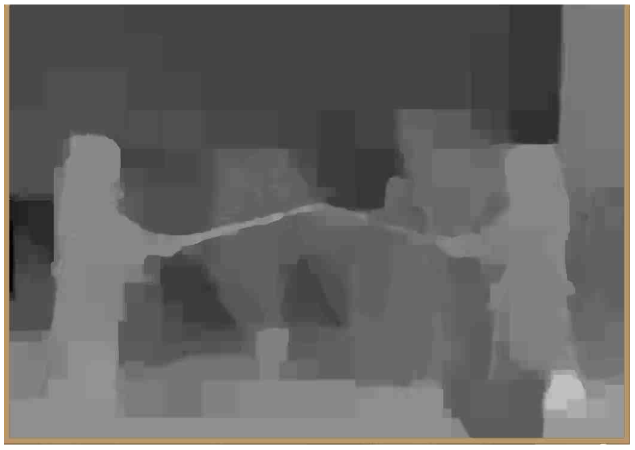


(c) Reconstructed picture using original software.

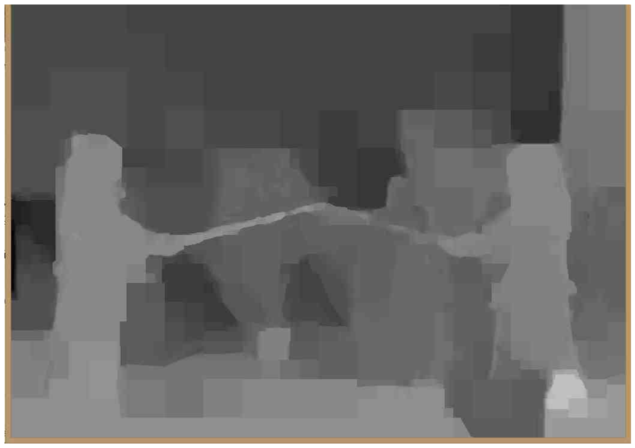
Fig. 2.9. Comparison of subjective video quality ALF in Kendo video sequence.



(a) Input depth view of Kendo test sequence.



(b) Reconstructed depth view using ALF.



(c) Reconstructed depth view using original software.

Fig. 2.10. Comparison of subjective depth view quality ALF in Kendo video sequence.

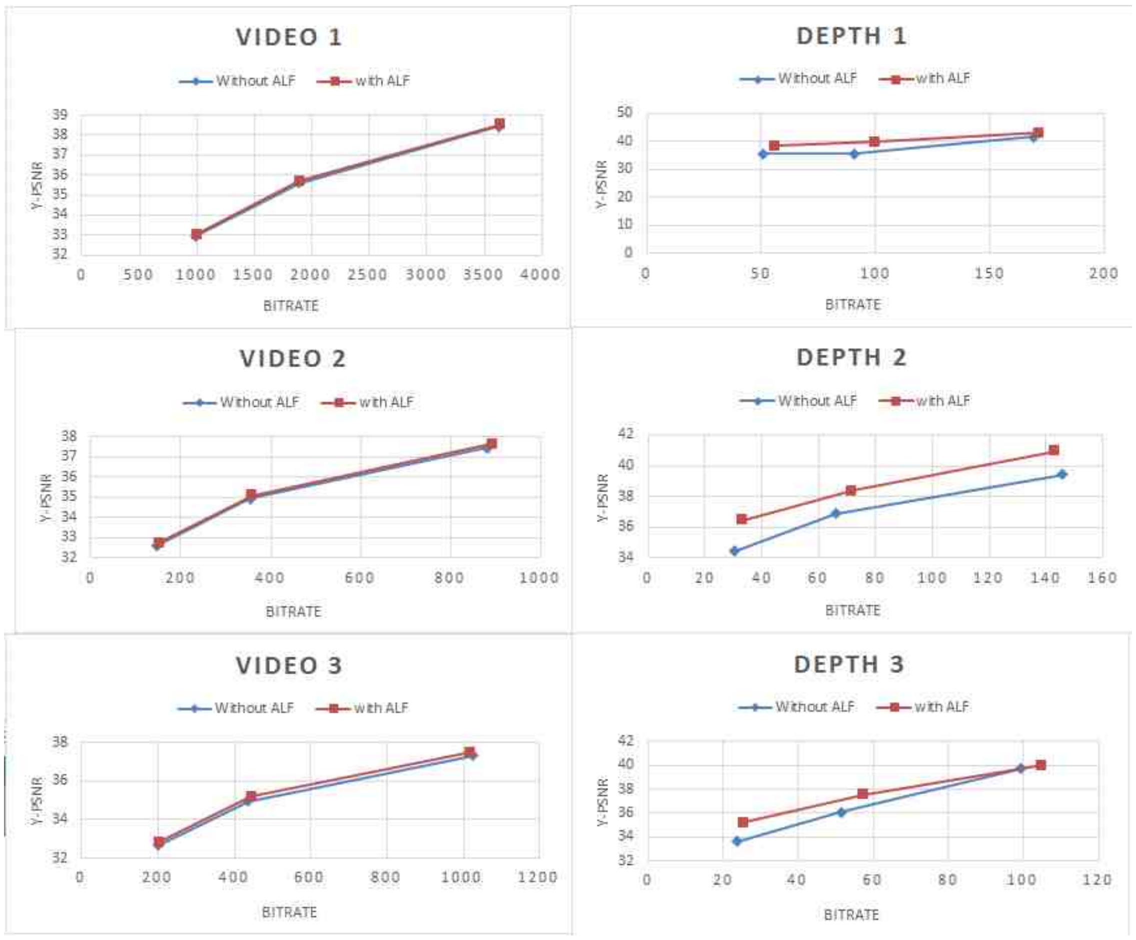


Fig. 2.11. Average PSNR in three encoded videos with corresponding depth views relative to bitrate for Lovebird1 video sequence

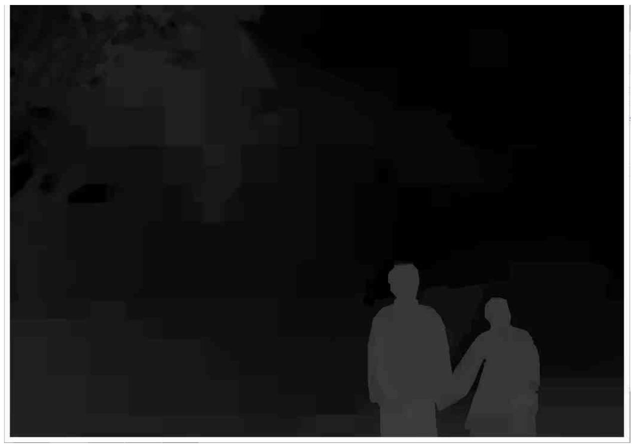
(a) Input picture of Lovebird1 test sequence.

(b) Reconstructed picture using ALF.

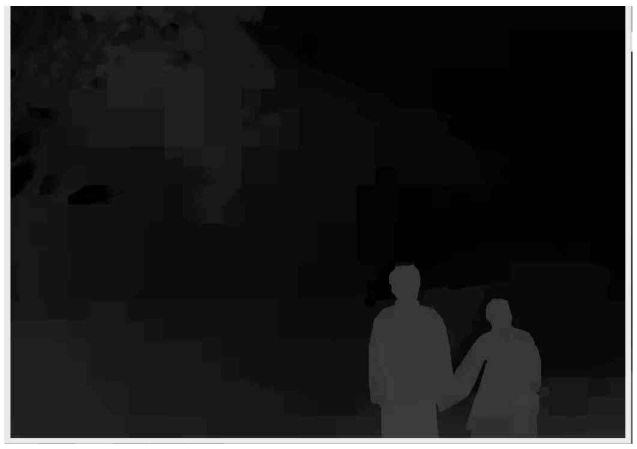
(c) Reconstructed picture using original software.

Fig. 2.12. Comparison of subjective video quality ALF in Kendo video sequence.

(a) Input depth view of Lovebird1 test sequence.



(b) Reconstructed depth view using ALF.



(c) Reconstructed depth view using original software.

Fig. 2.13. Comparison of subjective depth view quality ALF in Lovebird1 video sequence.

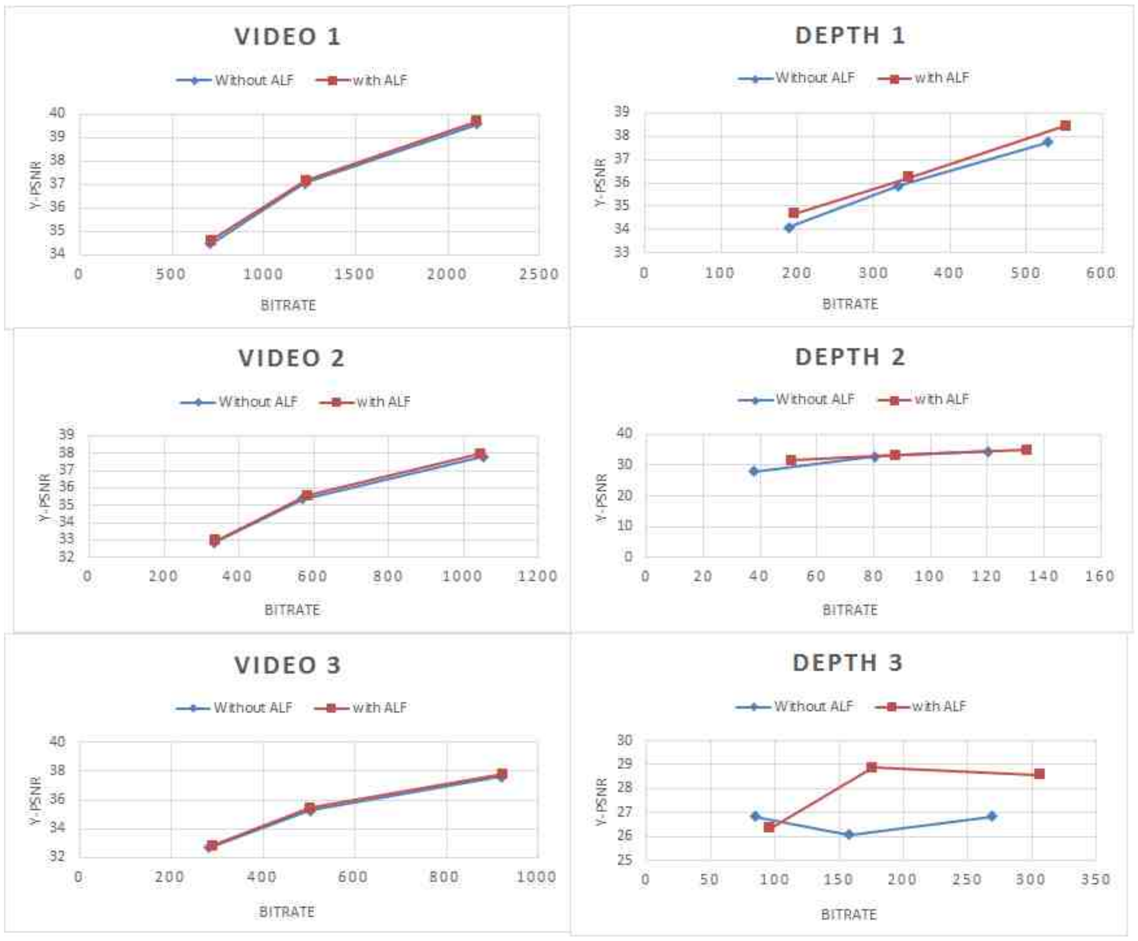
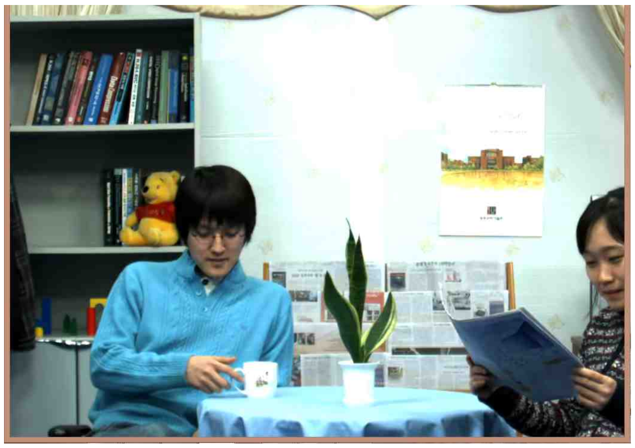
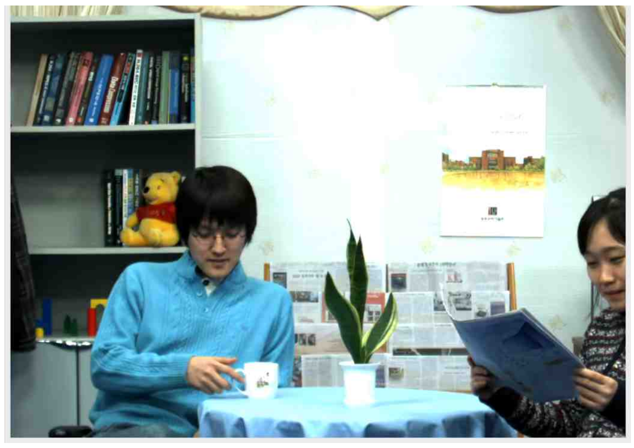


Fig. 2.14. Average PSNR in three encoded videos with corresponding depth views relative to bitrate for Newspaper video sequence

(a) Input picture of Newspaper test sequence.

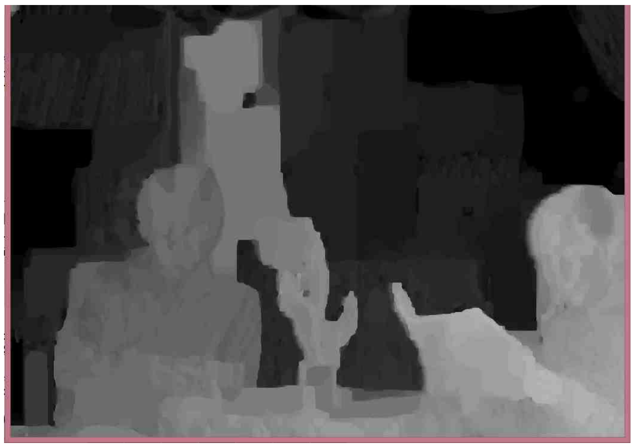


(b) Reconstructed picture using ALF.

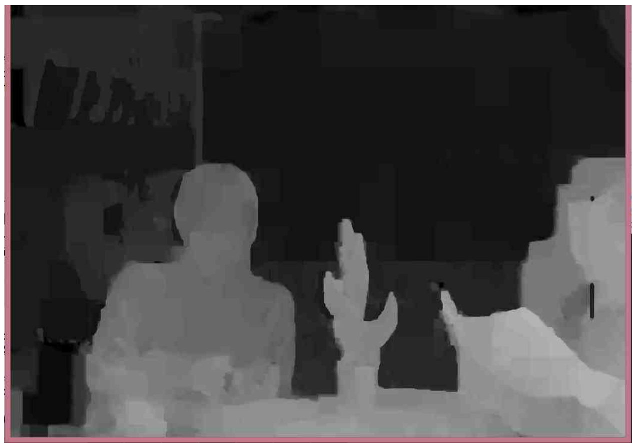


(c) Reconstructed picture using original software.

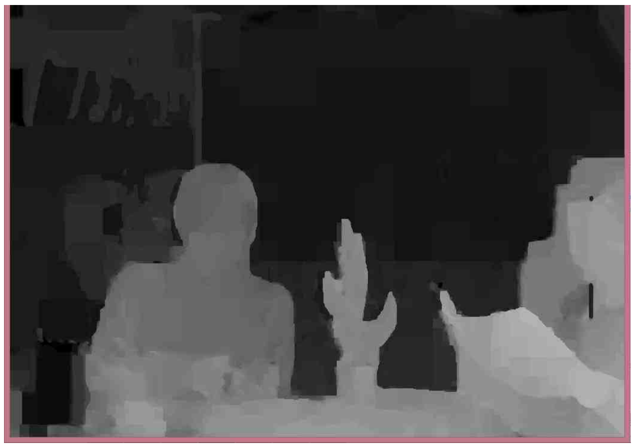
Fig. 2.15. Comparison of subjective video quality ALF in Newspaper video sequence.



(a) Input depth view of Newspaper test sequence.



(b) Reconstructed depth view using ALF.



(c) Reconstructed depth view using original software.

Fig. 2.16. Comparison of subjective depth view quality ALF in Newspaper video sequence.

2.2 Fast Bilateral Filter

2.2.1 Overview of Deblocking Filter in HEVC

Blocking is an artifact that is cause by two sources: Inter predicted using motion compensation or intra predicted from a spatial neighborhood result in forming blocks of picture. Intra or inter prediction effects block based parts where the boundaries of the neighboring blocks are not significantly related. Therefore, the blocks of picture after prediction becomes visible.In addition,block transform process of transform coding in HEVC leads to forming blocking artifacts. Increase in amount of quantization makes the block boundaries between neighboring transform more visible due to reconstruction of the independent neighboring blocks [8]. Deblocking filter is used to reduce the artifacts resulting from independent coding blocks. This filter first filters the vertical edge using horizontal filter. Afterwards, a horizontal edge is filtered using vertical filtering. Deblocking filtering is applied to 8×8 block boundaries. A questionable design process of deblocking filtering is to decide whether or not to filter a special block boundary and to decide on choosing either a normal or a strong filter to be applied. Deblocking is applied on a block boundary when all of the following conditions are fulfilled: 1) the block boundary is either a prediction unit or transform unit boundary; 2) the boundary strength is greater than zero; and 3) changes of signal on both sides of a block boundary is below a threshold [15]. According to Figure 2.17 if Equation 2.13 is satisfied, filtering for the first four lines is turned on and strong/weak filter selection process is applied:

$$dp0 + dq0 + dp3 + dq3 < \beta \tag{2.13}$$

On the other hand if conditions 2.14 and 2.15 are met, strong filter is used for the first four lines,

$$\begin{aligned} &2(dp0 + dq0) < (\beta/4), \\ &\mid p3_0 - p0_0 \mid + \mid q0_0 - q3_0 \mid < (\beta/8) \\ &\text{and} \mid p0_0 - q0_0 \mid < (2.5t_C) \end{aligned} \tag{2.14}$$

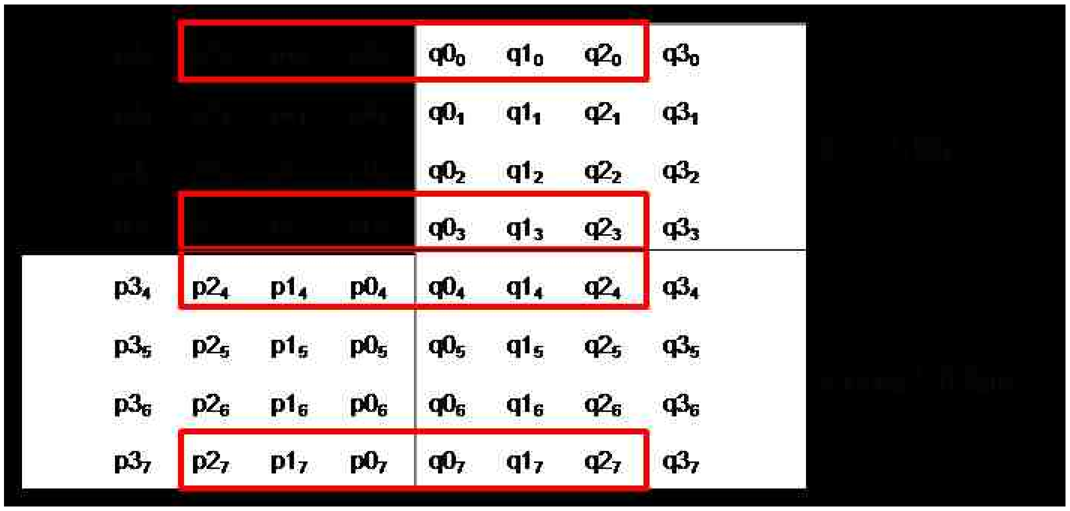

q0₀ q1₀ q2₀ q3₀
q0₁ q1₁ q2₁ q3₁
q0₂ q1₂ q2₂ q3₂
q0₃ q1₃ q2₃ q3₃
p3₄ p2₄ p1₄ p0₄ q0₄ q1₄ q2₄ q3₄
p3₅ p2₅ p1₅ p0₅ q0₅ q1₅ q2₅ q3₅
p3₆ p2₆ p1₆ p0₆ q0₆ q1₆ q2₆ q3₆
p3₇ p2₇ p1₇ p0₇ q0₇ q1₇ q2₇ q3₇

Fig. 2.17. Red boxes represent pixels involving in filter on/off decision and strong/weak filter selection [26]

$$\begin{aligned} &2(dp3 + dq3) < (\beta/4), \\ &\mid p3_3 - p0_3 \mid + \mid q0_3 - q3_3 \mid < (\beta/8) \\ &\text{and} \mid p0_3 - q0_3 \mid < (2.5t_C) \end{aligned} \tag{2.15}$$

If conditions 2.16 and 2.17 are met, strong filter is used for filtering of the second 4 lines. Otherwise, weak filter is used.

$$\begin{aligned} &2(dp4 + dq4) < (\beta/4), \\ &\mid p3_4 - p0_4 \mid + \mid q0_4 - q3_4 \mid < (\beta/8) \\ &\text{and} \mid p0_4 - q0_4 \mid < (2.5t_C) \end{aligned} \tag{2.16}$$

$$\begin{aligned} &2(dp7 + dq7) < (\beta/4), \\ &\mid p3_7 - p0_7 \mid + \mid q0_7 - q3_7 \mid < (\beta/8) \\ &\text{and} \mid p0_7 - q0_7 \mid < (2.5t_C) \end{aligned} \tag{2.17}$$

Where,

$$dp0 = \mid p2_0 - 2p1_0 + p0_0 \mid$$

$$dp3 = \mid p2_3 - 2p1_3 + p0_3 \mid$$

$$dp4 = \mid p2_4 - 2p1_4 + p0_4 \mid$$

$$dp7 = \mid p2_7 - 2p1_7 + p0_7 \mid$$

$$dq0 = \mid q2_0 - 2q1_0 + q0_0 \mid$$

$$dq3 = \mid q2_3 - 2q1_3 + q0_3 \mid$$

$$dq4 = \mid q2_4 - 2q1_4 + q0_4 \mid$$

$$dq7 = \mid q2_7 - 2q1_7 + q0_7 \mid$$

Threshold β and t_C used in filter on/off decision, strong and weak filter selection and weak filtering process are derived based on value of two neighboring blocks with common block edge [26].

2.2.2 Overview of Fast Bilateral Filter

This section describes Fast Bilateral Filter method used for 3D-HEVC for in-loop filtering implementation. Bilateral filter is a nonlinear filter that smoothes an image during which it preserves the strong edges. Since Bilateral Filter can eliminate the artifacts in an image, it can be used as a part of in-loop-filtering block in 3D-HEVC. In Bilateral Filter, a pixel is replaced by a weighted average of its neighbors. The idea is that since neighbor pixels are probably correlated, due to small variation in an image, it is reasonable to find the mean of the near pixels. In contrast, noise corrupting near pixels are less correlated, so impact of noise will be reduced by averaging the neighbors. This Bilateral Filtering process can be slow [27]. Bilateral Filter outputs each pixel as a weighted average of its neighbors. Table 2.7 summarizes the notations for this filter. In bilateral filter, the weight assigned to each neighbor decreases with the distance in the image plane and the distance on the intensity axis. Using a

Table 2.7.
Notation Used for Bilateral Filter

S	spatial domain
$\parallel x \parallel$	L_2 norm of vector x
$\otimes$	convolution operator
G_σ	1D Gaussian
I^b	result of the bilateral filter
R	range domain
$p \in S$	pixel position
g_{σ_s,σ_r}	3D Gaussian
σ_s, σ_r	Gaussian parameters(space and range)
W^b	normalization factor

Gaussian filter with a one dimensional G_σ as a decreasing function, and considering an image$I$, the result I^b of the bilateral filter is defined by:

$$I_p^b = \frac{1}{W_p^b} \sum_{q \in S} G_{\sigma_s}(\parallel p - q \parallel)\, G_{\sigma_r}(\mid I_p - I_q \mid)\, I_q \tag{2.18}$$

$$W_p^b = \sum_{q \in S} G_{\sigma_s}(\parallel p - q \parallel)\, G_{\sigma_r}(\mid I_p - I_q \mid) \tag{2.19}$$

The parameter σ_s shows the size of pixels surrounding a filtered pixel, and σ_r shows how much a neighbor pixel is down-weighted due to the a intensity difference. W^b normalizes the sum of the weights. A signal processing approach for the Fast Bilateral Filter is used to approximate Bilateral Filter function. In this method, a homogeneous intensity allows us to obtain the normalization term W_p^b as a homogeneous component. both sides of Equation 2.18 by W_p^b . Rewriting the equations using vectors:

$$\begin{pmatrix} W_p^b I_p^b \\ W_p^b \end{pmatrix} = \sum_{q \in S} G_{\sigma_s}(\parallel p - q \parallel)\, G_{\sigma_r}(\mid I_p - I_q \mid) \begin{pmatrix} I_q \\ 1 \end{pmatrix} \tag{2.20}$$

Where G_{σ_s} and G_{σ_r} are Gaussian functions, I the input image, S is the spatial domain, and I^b the filtered image after applying the bilateral filter. As we know, the Bilateral Filter replaces each pixel with a weighted neighbor pixels called W_q:

$$\begin{pmatrix} W_p^b I_p^b \\ W_p^b \end{pmatrix} = \sum_{q \in S} G_{\sigma_s}(\parallel p - q \parallel)\, G_{\sigma_r}(\mid I_p - I_q \mid) \begin{pmatrix} I_q W_q \\ W_q \end{pmatrix} \tag{2.21}$$

Considering calculation of $(W_q I_q, W_q)$ to each pixel q, we define the filtered pixels as linear combinations of their neighbor pixels. Basic idea of using this method is similar to homogeneous coordinates used in geometry projective. Adding a new coordinate linearize the calculations. Dividing by W^b is done at the last step. Vector (WI, W) as a two dimensional vector is called the homogeneous intensity [27].

As mentioned previously, this method introduces a new coordinate called intensity of the signal to keep track of the intensity and improve the accuracy of filtering an image. Extra dimension ξ and the intensity I for each pixel is used. We consider the Kronecker symbol $\delta(\xi)$ ($\delta(0) = 1$, $\delta(\xi) = 0$ otherwise) and the intensity ranging within the R . Now, we rewrite Equation 2.21 using $\delta(\xi I_q)$ so that the terms are canceled when $\xi \neq I_q$,

$$\begin{pmatrix} W_p^b I_p^b \\ W_p^b \end{pmatrix} = \sum_{q \in S} \sum_{\xi \in R} G_{\sigma_s} (\| p - q \|) \, G_{\sigma_r} (| \, I_p - \xi \, |) \, \delta (\xi - I_q) \begin{pmatrix} I_q W_q \\ W_q \end{pmatrix} \tag{2.22}$$

We rewrite the Equation 2.22:

$$\begin{pmatrix} W_p^b I_p^b \\ W_p^b \end{pmatrix} = \sum_{q \in S} \sum_{\xi \in R} G_{\sigma_s} (\| p - q \|) \, G_{\sigma_r} (| \, I_p - \xi \, |) \begin{pmatrix} \delta (\xi - I_q) \, I_q W_q \\ \delta (\xi - I_q) \, W_q \end{pmatrix} \tag{2.23}$$

The product $G_{\sigma_s} G_{\sigma_r}$ defines a separable Gaussian kernel on $S \times R$. The right side of the formula shows the convolution of Gaussian kernel and (wi, w) :

$$(w^b i^b, w^b) = g_{\sigma_s, \sigma_r} \otimes (wi, w) \tag{2.24}$$

Where g_{σ_s, σ_r} shows the separable Gaussian kernels of $G_{\sigma_s} G_{\sigma_r}$ on $S \times R$.

Fast Bilateral algorithm is summarized as the following. In this filter, image I, Gaussian parameters σ_s and σ_r and sampling rates s_s and s_r are the inputs for this algorithm and output is the filtered image I^b. In the following, w_d is the down sampled weights and i_d is the down sampled intensity.

1. Initialize all $w_d i_d$ and w_d values to 0.

2. Compute the minimum intensity:

$$I_{min} \triangleq \min_{(x,y) \in S} I(X, Y)$$

3. For each pixel (X, Y) in spatial domain S with an intensity $I(X, Y)$ in the range R,

   a) Find the homogeneous vector (wi, w):

$$(wi, w) \triangleq (I(X, Y), 1)$$

   b) Calculate the down-sampled pixel coordinates,

$$(x, y, \xi) \triangleq \left(\left[\frac{X}{S_s} \right], \left[\frac{Y}{S_s} \right], \left[\frac{I(X, Y) - I_{min}}{S_r} \right] \right)$$

c) Update the down-sampled $S \times R$ space

$$\begin{pmatrix} w_d i_d(x,y,\xi) \\ w_d(x,y,\xi) \end{pmatrix} \triangleq \begin{pmatrix} w_d i_d(x,y,\xi) \\ w_d(x,y,\xi) \end{pmatrix} + \begin{pmatrix} wi \\ w \end{pmatrix}$$

4. Convolve $(w_d i_d, w_d)$ with a 3D Gaussian g whose parameters are σ_s/s_s and $\sigma r/s_r$

$$(w_d^b i_d^b, w_d^b) \triangleq (w_d i_d, w_d) \otimes g$$

5. For each pixel $(X,Y) \in S$ with an intensity $I(X,Y) \in R$

$$I^b(X,Y) \triangleq \frac{W^b I^b(X,Y)}{W^b(X,Y)}$$

All the above steps lead the fast bilateral filter to keep track of intensity and use a 3D Gaussian considering intensity as an additional coordinate to the pixel position. In this method, down-sampled signal is convolved with the Gaussian kernel instead of the whole image without affecting the quality of the image. Mentioned fact helps the process of filtering to be faster than bilateral filtering. For evaluation of Fast Bilateral Filter technique in a professional environment, an eight mega pixel picture is processed. In this implementation, a running time of 2.5s is obtained. An image with DVD resolution, which is $576 \times 320 \simeq 0.2$ mega pixel, a running time of about 57ms near the 40ms is required to reach real-time performances, which is a 25Hz frame rate. If we run fast bilateral filter on machine with an AMD Opteron 252 at 2.6MHz and 1MB of cache, the running time is 38ms. In other words, FBF algorithm software implementation can reach real-time video processing [27].

2.2.3 Implementation of Bilateral Filter in 3D-HEVC

Figure 2.18 shows the encoder block diagram of this implementation, which contains four in-loop filtering blocks: fast bilateral filter (FBF), deblocking filter (DF), sample adaptive offset (SAO), and Adaptive Loop Filter (ALF). The encoder shown in Figure 2.18 picture is split into block-shaped regions. The reference picture is coded using intra-picture prediction which uses the closest data related to the region of the reference image. The encoding process of inter-picture prediction choses data related to the reference frame and motion vector (MV) to be done for predicting each block.

Afterwards, the residual between the original block and its prediction (inter or intra prediction) is transformed using a linear spatial transform. The transform coefficients are quantized, entropy coded, and sent with the prediction information to the decoder.

The quantized coefficients are inverse quantized and inverse transformed and would be reconstructed to be used in prediction unit. reconstructed signal is also sent to the in-loop filtering to be smoothed.

After in-loop filtering the picture will be saved in a picture buffer to be used for the prediction of further pictures. As can be seen, loop filter is functioning as a feedback for prediction unit. Since loop filter can remove the artifacts, it has an important role in the quality of the image and improving the quality of the reference picture results in improvement of dependent coded pictures. In this implementation, FBF is located at the first stage of filtering process to reduce artifacts for each picture. Note that DF uses predefined filters. In contrast, sample adaptive offset and adaptive loop filter exploit the original pixels of the current picture to decrease the mean difference between the original pixels and the reconstructed pixels. In other words, SAO filter is adaptive in the sense that it adds an offset and uses a finite impulse response (FIR) filter, with coded additional data signaling filter coefficients and the offset values. Adaptive loop filter is applied at the the end of post processing filtering of every picture. ALF is considered as a method to detect and remove artifacts formed by the previous coding blocks [19].

Strong Deblocking Filter affects three pixels using four pixels on each side of the block boundary. On the other hand, weak Deblocking Filter affects at most two pixels using three pixels on each side of the block boundary. This concept means that the more the use of the strong filter the slower the process of coding. In order to speed

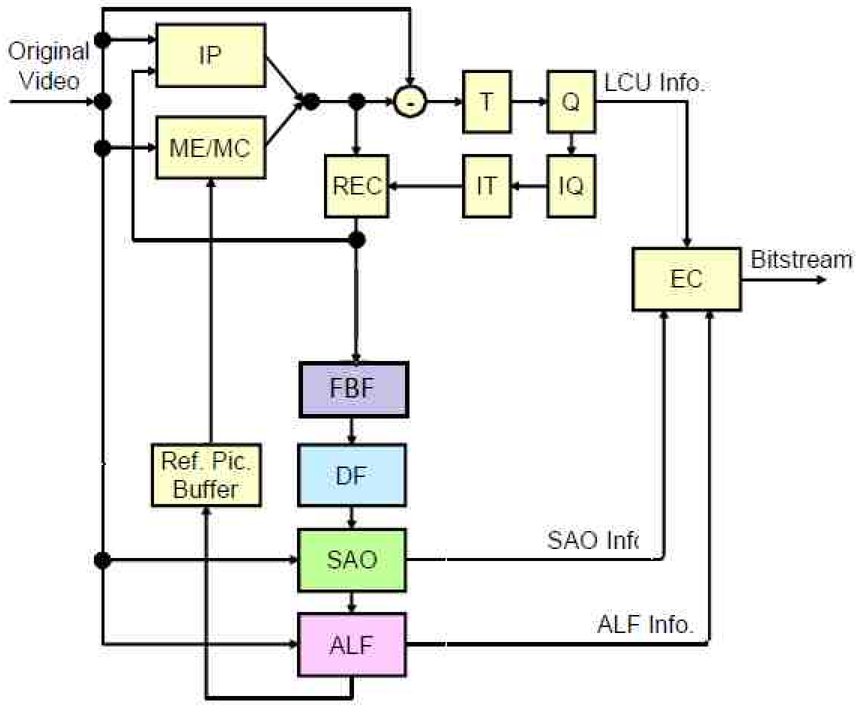


Fig. 2.18. HEVC Encoder Implementation with Fast Bilateral Filter and Adaptive Loop Filter [19].

up the coding process,removing the artifacts using Fast Bilateral Filter just before applying Deblocking Filter is an effective method. Fast bilateral filter replaces each pixel in the image with its weighted neighbor pixels and smoothes images without effecting the strong edges which causes neighbor pixels to be more correlated.

Moreover, block boundaries in images show larger variation in neighbor pixels. Therefore, Fast Bilateral Filter removes the block boundaries to some extend which causes thresholds of block boundaries taking place in weak Deblocking Filter most of the times. As discussed above, this process reduces choosing strong Deblocking Filter forcing the encoder time to reduce up to 20% with minimal change in PSNR and quality of the video. The concept of using bilateral filter was demonstrated previously in HEVC design [28]. In this implementation, the main idea is to implement an adaptive bilateral filter along with adaptive loop filter that can improve the quality with benefit of bitrate reduction.

2.2.4 Testing Fast Bilateral Filter in 3D-HEVC

Proposed algorithm uses FBF just before Deblocking Filter and it removes the artifacts as much as possible which helps choosing weak Deblocking Filter and it reduces using strong DF leading to faster process of encoding. Later, the Sample Adaptive Offset is applied and finally ALF is used to increase PSNR of video. PSNR is defined previously in Section 2.1.6.

For implementation of filters we used HM 6.2. The aim of the HM encoder is to provide a reference implementation of an HEVC encoder, useful as a test platform for evaluating technologies and for independent encoder or decoder development.

For more information please see Results section of Adaptive Loop Filter design. The 3D video test set consisted of 4 sequences with the video component in 4:2:0 chroma format: 4 with a progressive HD resolution of 1024 by 768 luma/depth samples with 30 fps. Multiview test sequences are called: Balloons, Kendo, Newspaper and Lovebird1 are shown in table 2.1.6 and their features were described.

2.2.5 Results

Encoder was evaluated in a test scenario: with FBF and ALF implementation in picture-based level, video and depth component of 3 views V0, V1, V2 were coded and the changes in PSNR and coding running time is evaluated. Test model of HTM 6.2 was used for implementation. Tables 2.8, 2.9, 2.10, and 2.11 show the results of average changes in PSNR of Y, U and V components using ALF in picture-based for both depth and texture view and Fast Bilateral Filter in pictured based for texture view. The 3D video of 4 multiview test sequences called: Balloons, Kendo, Newspaper and Lovebird1 described previously was used to test the results.

As can be seen from the Table 2.8, encoding time decreased up to 20 percent for test sequences. The most impact has on Balloons because Balloons has less details compared to Kendo, Lovebird1 and Newspaper. Since Bilateral Filter preserves strong edges, test sequences with more details will not be effected smoothing Bilateral Filter and more number of strong DFs will be used instead of weak ones. It means that processing time for more detailed images does not decrease as much as edgy images. Table 2.8 shows that texture luminance quality is reduced for 0.1 dB compared to use ALF alone. Chrominance component does not change at all for depth or texture view of video sequences.

Proposed algorithm in paper [28] uses an adaptive bilateral filter and adaptive loop filter in HEVC coding. Results show that proposed algorithm can reach maximum of 7 percent bitrate reduction, while PSNR improvement is maximum of 0.2 dB [28]. Our proposed method uses the concept of an approximation of bilateral filter known as fast bilateral filter instead of bilateral filter and extending to 3D-HEVC encoder. In other words, our method is used for both depth and the texture view of a multi-view test sequence, improving the PSNR up to 2 dB and maximum of 12 percent reduction in bitrate.

Implementation of using other non linear filters such as trilateral filter along with adaptive loop filter in HEVC encoder is demonstrated in [29]. Using trilateral filter and adaptive loop filter improves the Y-PSNR with a maximum of 0.1, while our proposed method improves the Y-PSNR up to 0.2 dB in texture and 2 dB in depth view. Proposed algorithm in [29] increases running time ranging from 20 percent to 33 percent while our proposed has a superiority in running time decrease, ranging from 2 to 20 percent depending on the quantization parameter and multiview test sequence. In our proposed method, a maximum of 20% reduction in encoding time is observed for Balloons while inter-view QP for texture is 30 and for the depth is 39. Results show that bitrate for discussed algorithm is decreased up to 1.8 percent and generally not much of change in bitrate is observed. In other words, this implementation is optimal in the sense of bitrate changes. Detailed changes of PSNR relative to the changes of

Table 2.8.
Average PSNR changes in Luminance and Chrominance components with different QP Values for Balloons video sequence

				Balloons		
	Y (dB)	U(dB)	V (dB)	Inter-view QP	Bitrate %	Encoder Time %
Texture	-0.1	0	0	30	0.1%	-20%
Depth	0.1	0	0	39	-1.8%	
Texture	-0.1	0	0	35	-0.6%	-10%
Depth	0.2	0	0	42	2.1%	
Texture	-0.1	0	0	40	-0.6%	-7%
Depth	0.5	0	0	45	1.5%	

bitrate is shown in Figure 2.19. Results indicate that Y-PSNR is improved compared using FBF with ALF in depth view compared to using ALF alone. Comparison of

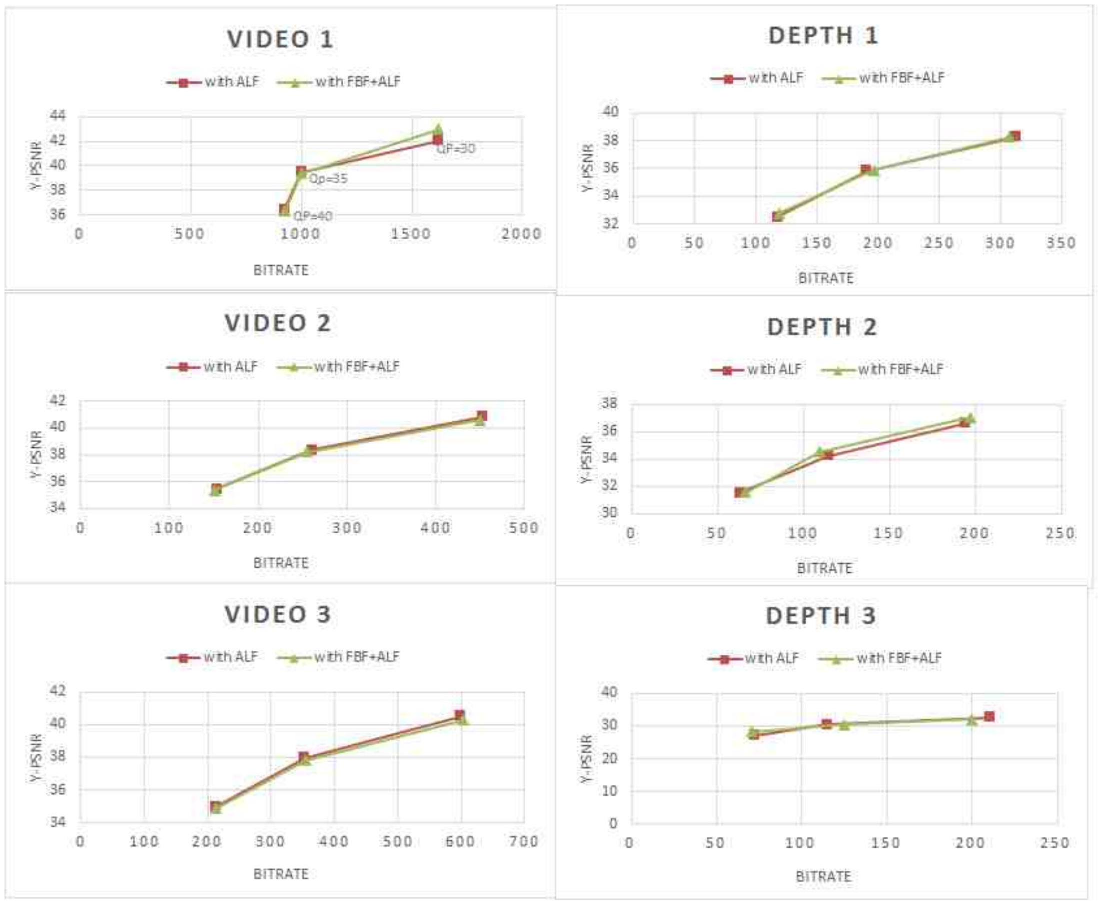


Fig. 2.19. Average PSNR in three encoded videos with corresponding depth views relative to bitrate for Balloons video sequence

subjective video quality ALF in Balloons video sequence is showed in the Figure 2.20 with corresponding depth view in Figure 2.21. In this experiment the inter-view QP is 30 for texture and 39 for the depth and results of encoded images can be seen. As can be seen from the figure, the color intensity is closer to the original input picture compared to the anchor test model. Therefore, this algorithm can increase the quality of the encoded videos. Also, in depth view we can see the block artifacts are much more reduced compared to the anchor test model. Kendo video sequence results is depicted in Table 2.9. Up to 0.1 dB improvement is observed compared to using ALF for in-loop filtering design. Maximum of 6 percent reduction in running time can be seen and there is not much of difference in reduction of encoder time relative to changes of QP. All the chrominance component of texture and depth stayed the same which shows that this method reduces encoding time without affecting the quality compared to design of ALF. Table 2.9 indicates that bitrate either does not change or it is reduced by 2 percent which means that this algorithm is optimal for Kendo test sequence.

Table 2.9.
Average PSNR changes in Luminance and Chrominance components with different QP Values for Kendo video sequence

				Kendo		
	Y (dB)	U (dB)	V (dB)	Inter-view QP	Bitrate %	Encoder Time %
Texture	-0.2	0	0	30	0.3%	-5%
Depth	0.1	0	0	39	-2.2%	
Texture	-0.1	0	0	35	-0.4%	-5%
Depth	-0.1	0	0	42	0.2%	
Texture	0.1	0	0	40	-1.5	%-6%
Depth	0.1	0	0	45	-0.3%	

Detailed changes of PSNR relative to the changes of bitrate is shown in Figure 2.22. Results indicate that when bitrate is not very high or very low the quality improvement can be in maximum value and higher improvement is shown compared to the video but note that higher PSNR improvement results in higher bitrate. Comparison of subjective video quality ALF in Kendo video sequence is showed in the Figure 2.23 with their corresponding depth view in Figure 2.24 while inter-view QP is 30 for the texture and 39 for the depth.

Previously, we discussed that using ALF improves PSNR of the images. Since FBF does not reduce the quality of an image it can be used as a way to process coding of pictures faster. In other words, this implementation is optimal because first it improves the quality of coding pictures compared to the original encoder and second, it process the encoding faster and third, it does not increase the bitrate.

Lovebird1 is demonstrated in Table 2.10. This test sequence does not show any changes in PSNR of texture or depth. Instead, maximum of 10 percent of encoding time reduction is observed while inter-view QP for texture is 40 and for the depth is 45. Bitrate changes is very small in Lovebird test sequence up to 1 percent bitrate reduction is observed.

Table 2.10.
Average PSNR changes in Luminance and Chrominance components with different QP Values for Lovebird1 video sequence

				Lovebird1		
	Y (dB)	U (dB)	V (dB)	Inter-view QP	Bitrate %	Encoder Time %
Texture	0	0	0	30	0.2%	-2%
Depth	0	0	0	39	0.4%	
Texture	0	0	0	35	0%	-7%
Depth	0	0	0	42	-0.3%	
Texture	0	0	0	40	-1%	-10%
Depth	0.1	0	0	45	0.9%	

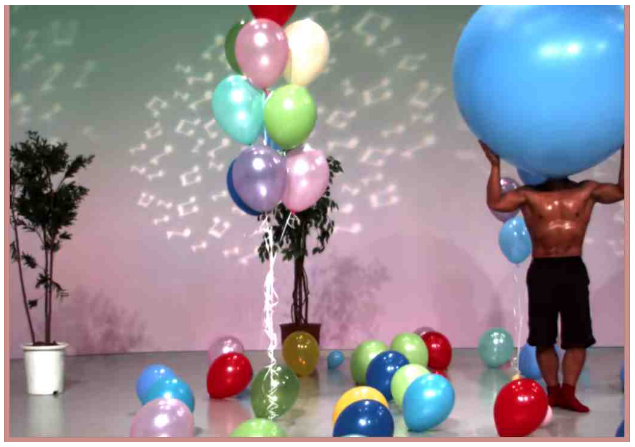
Detailed changes of PSNR relative to the changes of bitrate is shown in Figure 2.25. Results indicate that not much of changes in quality is observed compared to ALF implementation but bitrate can be reduced up to 1 percent in texture view. Comparison of subjective video quality of proposed algorithm in Lovebird1 video sequence is showed in the Figures 2.26 with their corresponding depth view in Figure 2.27 while inter-view QP is 30 for the texture and 39 for the depth where one of thee coded videos is shown as an example. As can be seen, the brightness and color of encoded video using proposed algorithm is more closer to the input video sequence compared to the anchor test model. In addition, the block artifacts in depth view after implementing the proposed algorithm is less visible than the original encoder. The Newspaper test sequence in Table 2.11 shows reduction in PSNR of the luminance component of texture and depth for 0.3 dB. On the other hand, PSNR for chrominance component does not change. This test sequence does not change very much in encoder time processing. A maximum of 3 percent of reduction in time while QP of inter-view for texture is 35 is observed.

Table 2.11.
Average PSNR changes in Luminance and Chrominance components with different QP Values for Newspaper video sequence

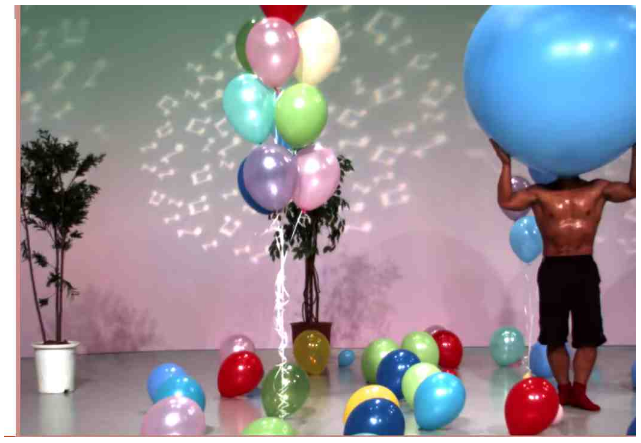
	Newspaper					
	Y (dB)	U (dB)	V (dB)	Inter-view QP	Bitrate %	Encoder Time%
Texture	-0.2	0	0	30	0.2%	-2%
Depth	-0.2	0	0	39	1.3%	
Texture	-0.2	0	0	35	-0.1%	-3%
Depth	-0.3	0	0	42	-10%	
Texture	-0.1	0	0	40	0.1%	-3%
Depth	-0.3	0	0	45	-12.3%	

Comparison of subjective video quality ALF in Balloons video sequence is demonstrated in the Figure 2.29 with the depth view in Figure 2.30 while inter-view QP is 30 for the texture and 39 for the depth where one of thee coded videos is shown as an example. Generally, using Fast Bilateral Filter with ALF reduces the running time of encoder from 2 to 20 percent compared to implementation of ALF by itself. In addition, PSNR can either be decreased or increased minimally for different test sequences compared to using ALF filter. This can be helpful for the applications that running time is essential. Although we observed reduction in the quality of the image compared to implementation of ALF alone, combination of ALF and FBF improves the quality compared to the anchor test model while it reduces the running time.

(a) Input picture of Balloons test sequence.

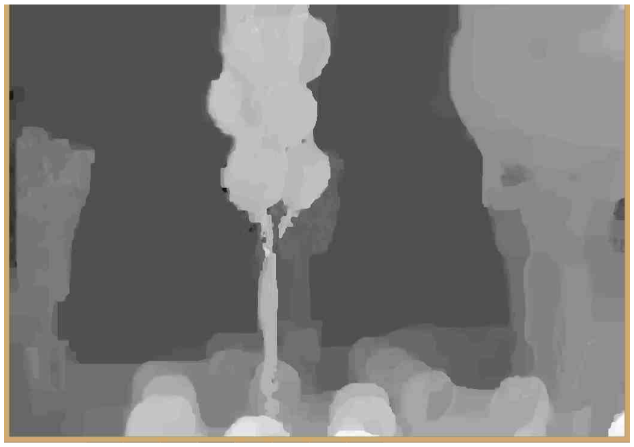


(b) Reconstructed picture using proposed algorithm.

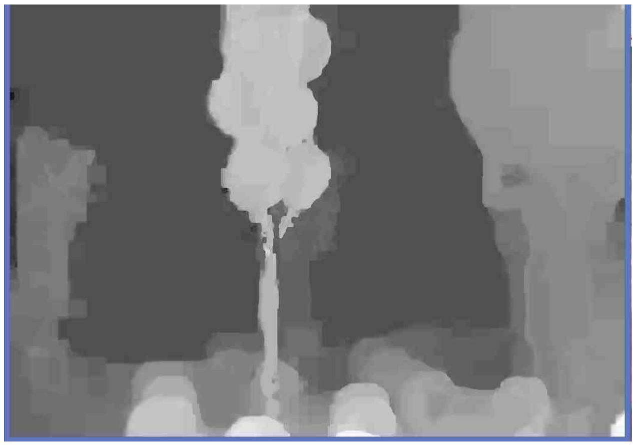


(c) Reconstructed picture using original software.

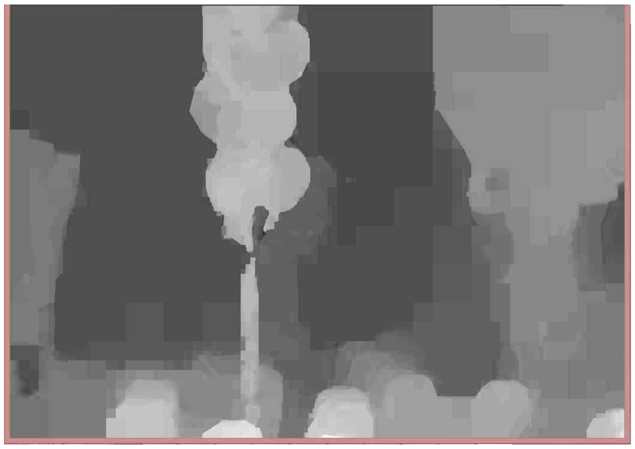
Fig. 2.20. Comparison of subjective video quality ALF in Balloons video sequence.



(a) Input depth view of Balloons test sequence.



(b) Reconstructed depth view using proposed algorithm.



(c) Reconstructed depth view using original software.

Fig. 2.21. Comparison of subjective depth view quality of proposed algorithm in Balloons video sequence.

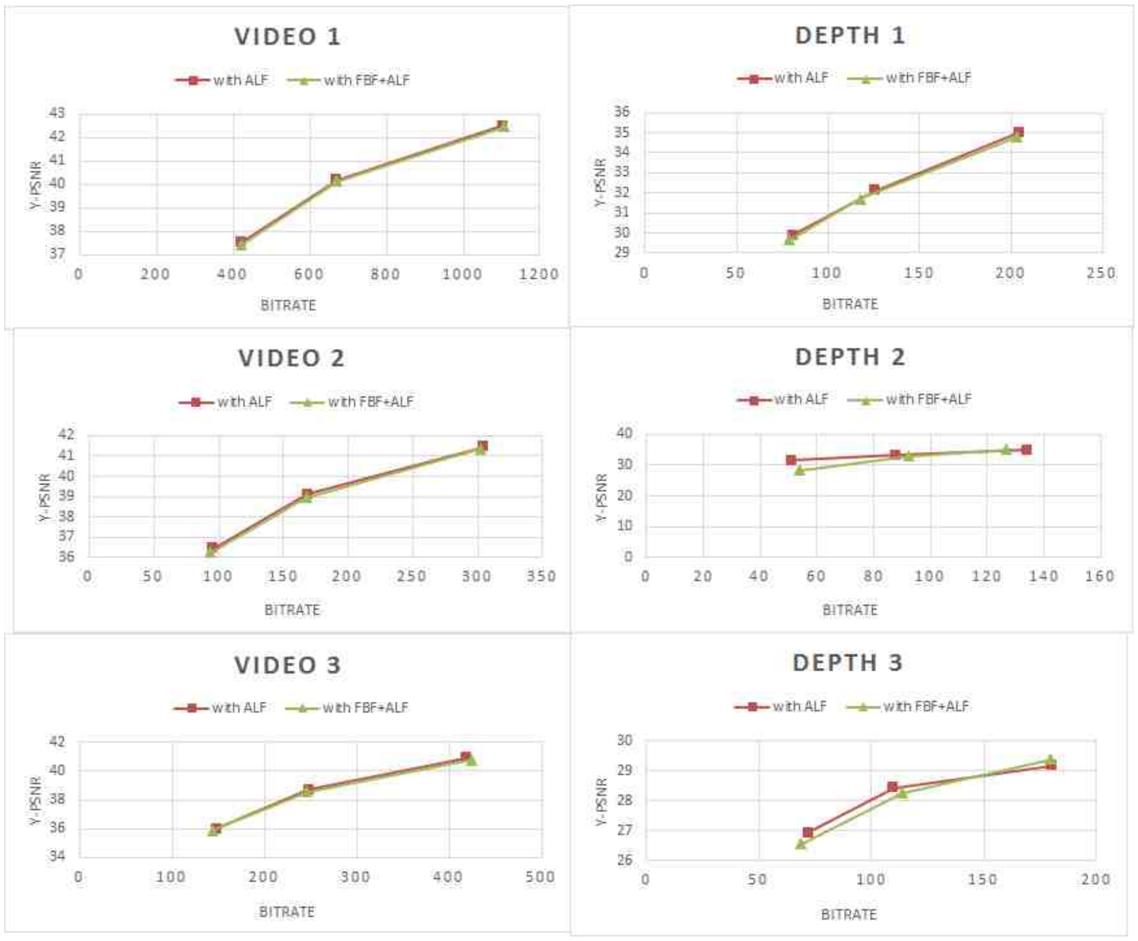
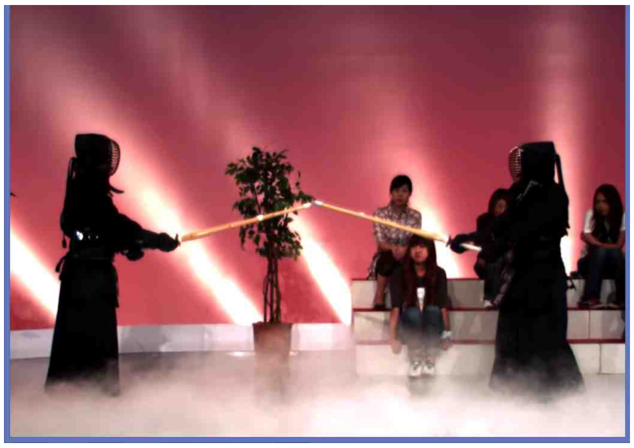
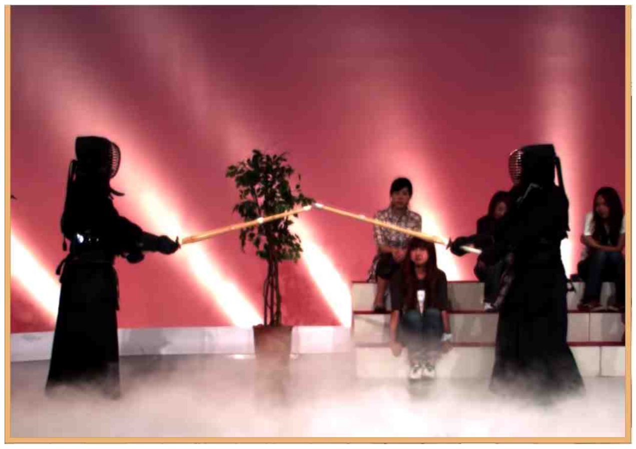


Fig. 2.22. Average PSNR in three encoded videos with corresponding depth views relative to bitrate for Kendo video sequence

(a) Input picture of Kendo test sequence.

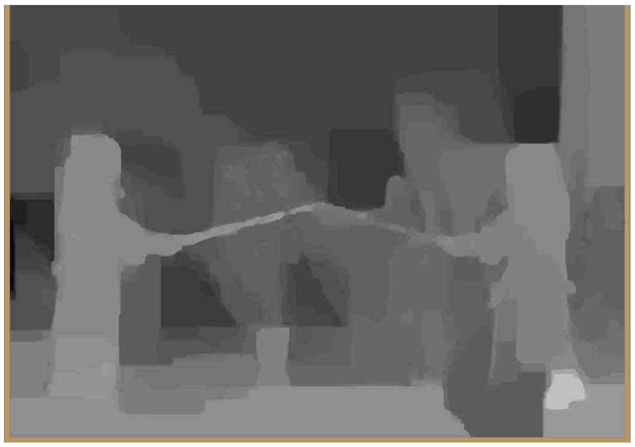


(b) Reconstructed picture using proposed algorithm.

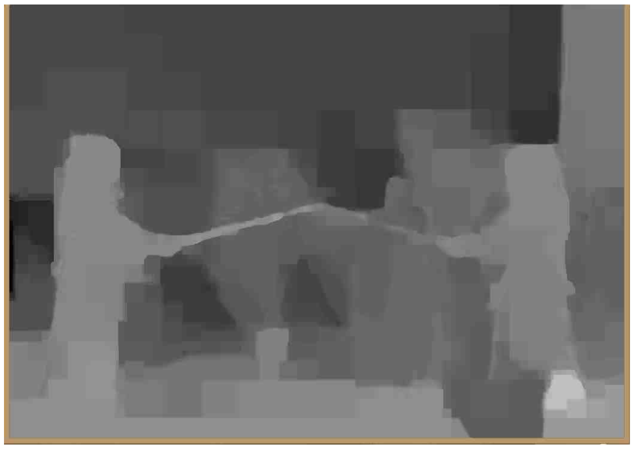


(c) Reconstructed picture using original software.

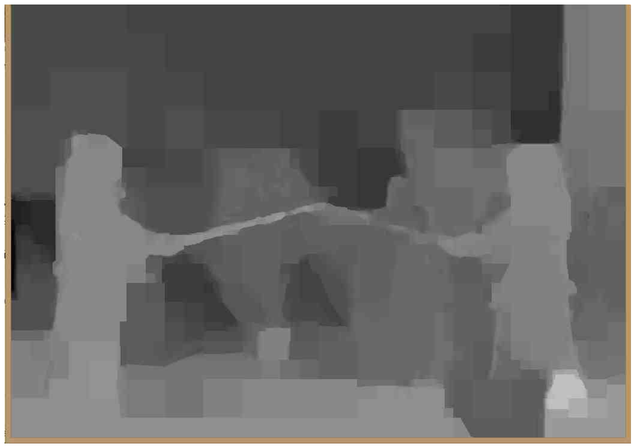
Fig. 2.23. Comparison of subjective video quality of proposed algorithm in Kendo video sequence.



(a) Input depth view of Kendo test sequence.



(b) Reconstructed depth view using ALF.



(c) Reconstructed depth view using original software.

Fig. 2.24. Comparison of subjective depth view quality ALF in Kendo video sequence.

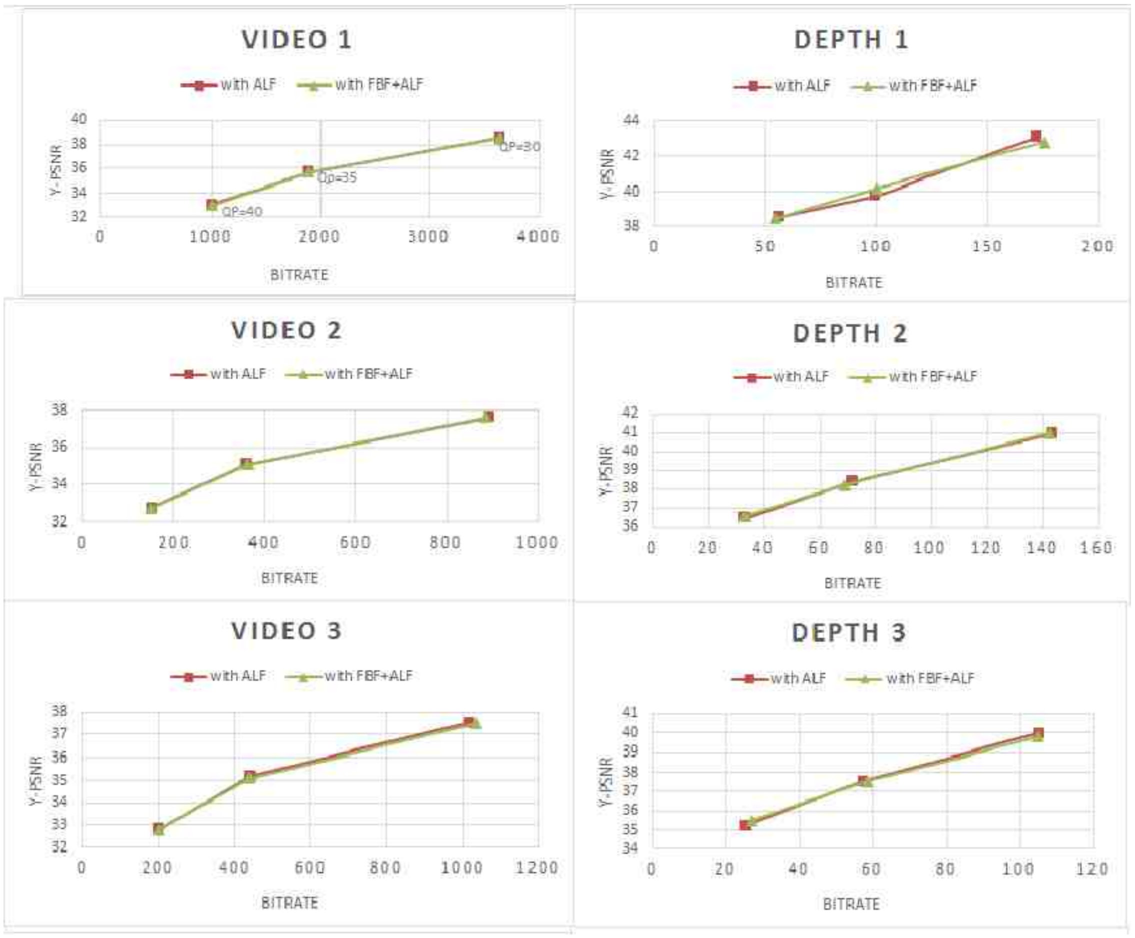
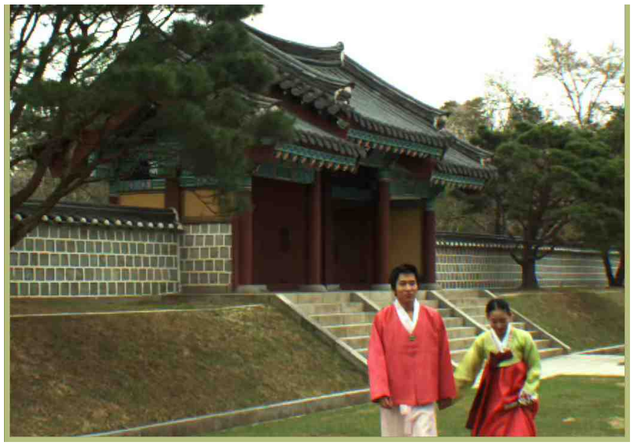
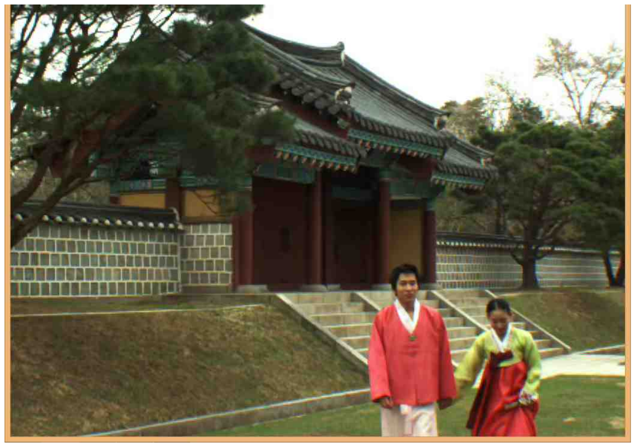


Fig. 2.25. Average PSNR in three encoded videos with corresponding depth views relative to bitrate for Lovebird1 video sequence.

(a) Input picture of Lovebird1 test sequence.

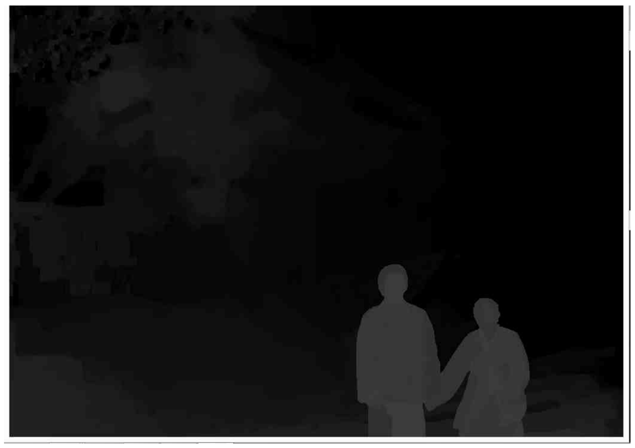


(b) Reconstructed picture using proposed algorithm.

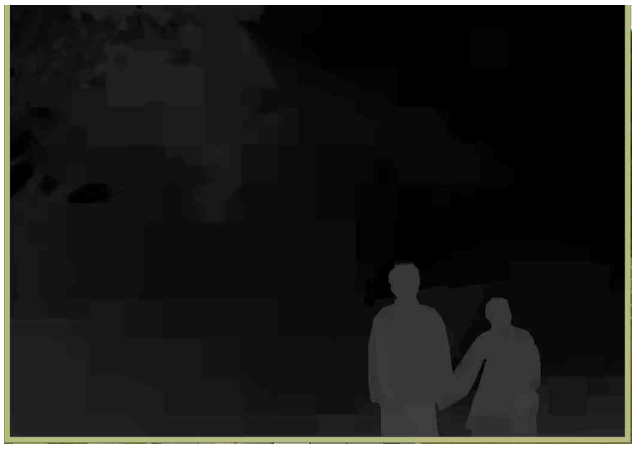


(c) Reconstructed picture using original software.

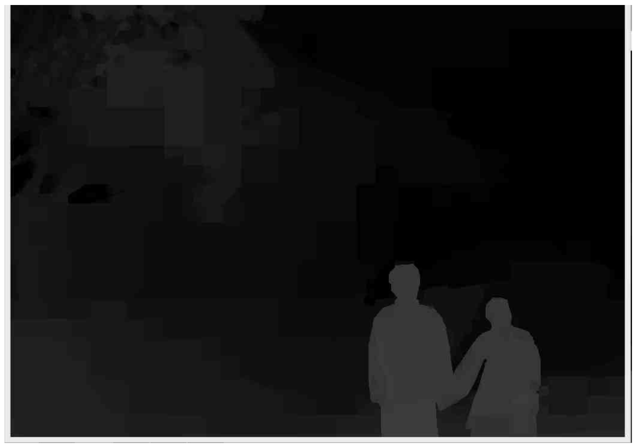
Fig. 2.26. Comparison of subjective video quality of propose algorithm in Kendo video sequence.



(a) Input depth view of Lovebird1 test sequence.



(b) Reconstructed depth view using proposed algorithm.



(c) Reconstructed depth view using original software.

Fig. 2.27. Comparison of subjective depth view quality of proposed algorithmin Lovebird1 video sequence.

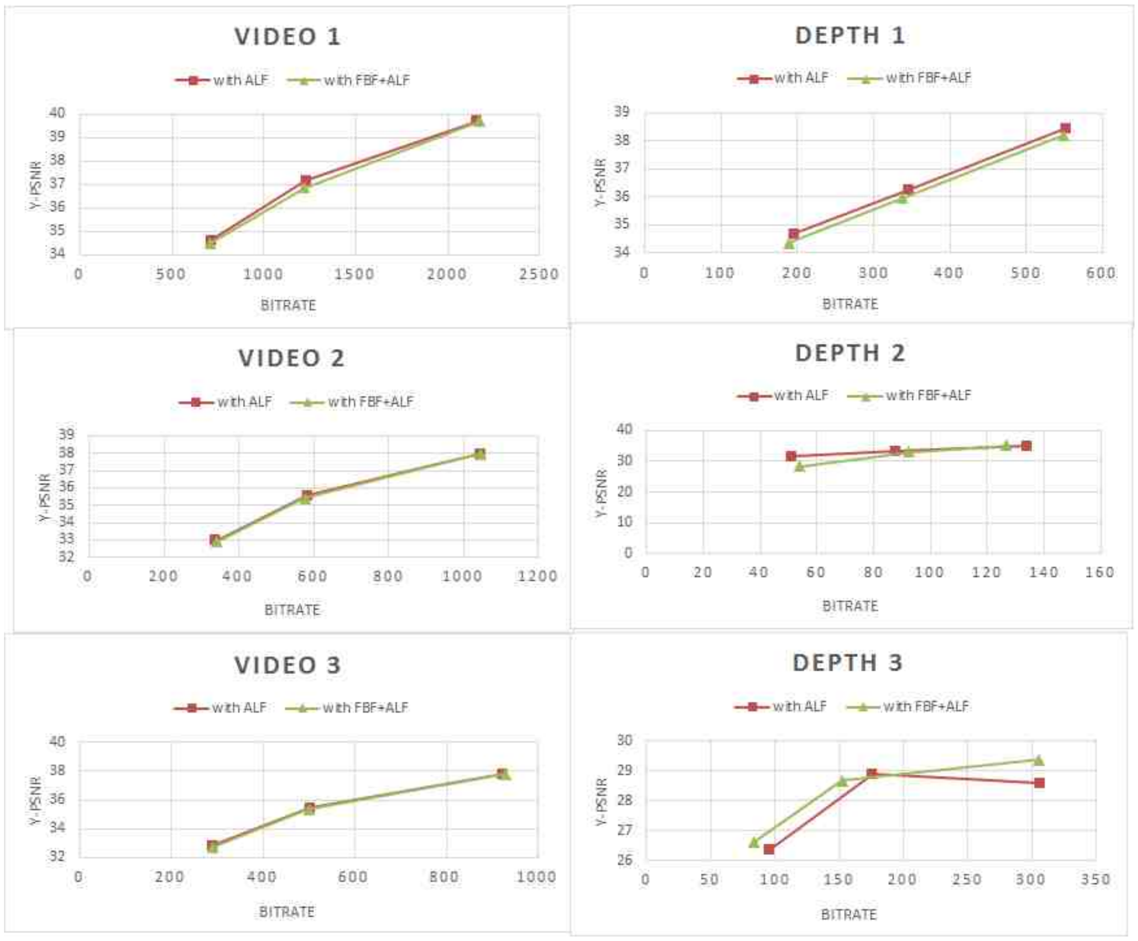
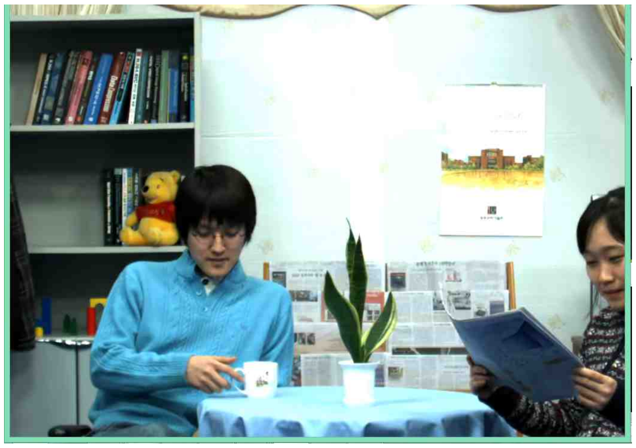
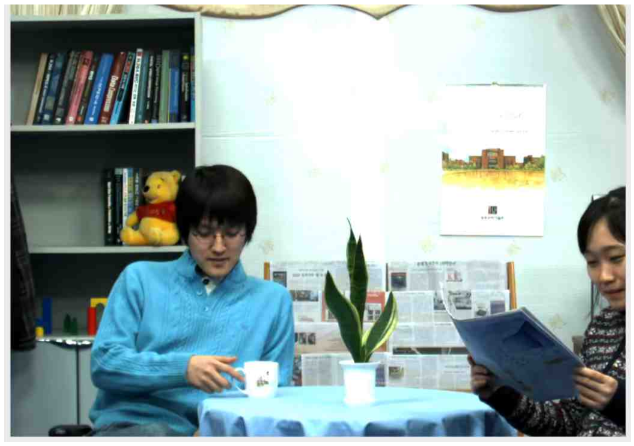


Fig. 2.28. Average PSNR in three encoded videos with corresponding depth views relative to bitrate for Newspaper video sequence

(a) Input picture of Newspaper test sequence.

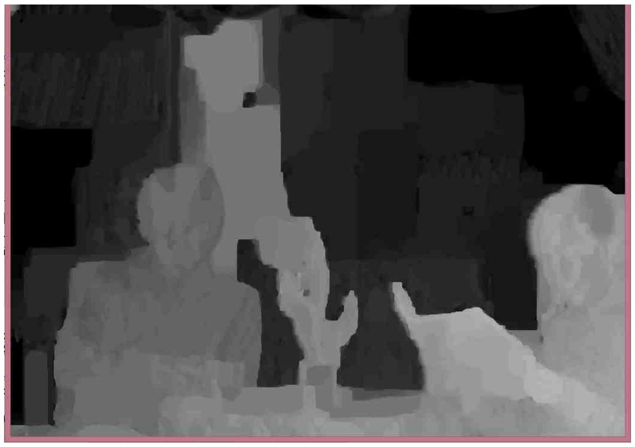


(b) Reconstructed picture using proposed algorithm

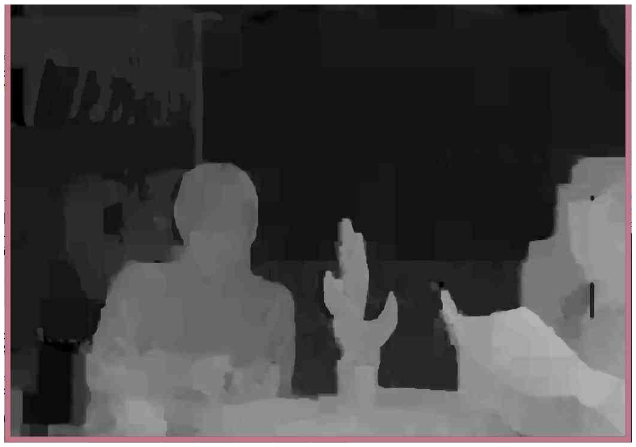


(c) Reconstructed picture using original software.

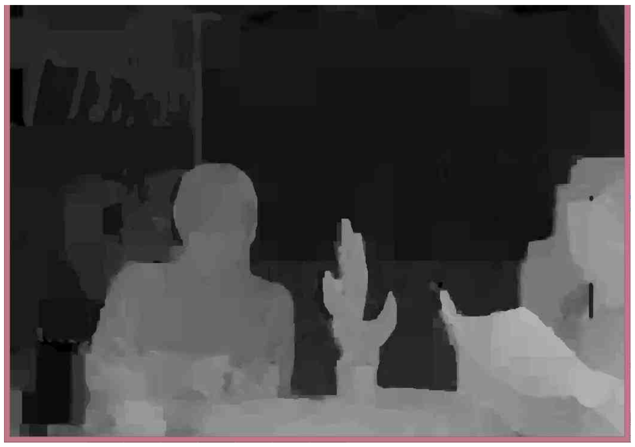
Fig. 2.29. Comparison of subjective video quality of proposed algorithm in Newspaper video sequence.



(a) Input depth view of Newspaper test sequence.



(b) Reconstructed depth view using proposed algorithm.



(c) Reconstructed depth view using original software.

Fig. 2.30. Comparison of subjective depth view quality of proposed algorithm in Newspaper video sequence.

3. APPLICATION CONSIDERATION

The results show that the proposed algorithm is adaptive in the sense that based on the application different settings of Bilateral Filter and Adaptive Loop Filter can be used. Three settings are suggested for different applications.

- If a low complexity is needed where quality (higher PSNR) and processing time is not an important issue for an application such as small devices the Adaptive Loop Filter and Fast Bilateral Filter can be signaled off. As discussed previously, the design of original test model with just two in-loop filters including Sample Adaptive Offset and Deblocking Filter, which is shown in Figure 1.3. This design has the lowest complexity with the minimum quality compared to other settings.

- If the quality is the most important aspect where running time and the complexity is not an essential issue, we suggest signaling the Adaptive Loop Filter to be on and Fast Bilateral filter to be off shown in Figure 2.1.1.

  Above design has three in-loop filtering which first filters an image using SAO and then DF removes the artifacts and at the end ALF minimizes the Minimum Square Error to increase PSNR of the video. ALF filter has higher impact on the depth and it has been observed up to 2dB improvement in the quality of encoded video. As discussed, this setting has higher complexity compared to the first setting, in Section 2.1.3, we have discussed that ALF can be designed using virtual boundary to avoid the memory bandwidth increase.

Note that implementation of ALF requires larger chip design and longer processing time because of finding the optimum solution for the coefficients which increases the running time resulting in more complexed design of encoder compared to the previous settings. This fact will make the maximum quality with 1-5 percent of increase in running time of encoding multiview video sequences

- If both high quality and low running time are important, signaling FBF and ALF filter to be on is suggested. This setting lets the running time to decrease up to 20 percent compared to the second setting.
  Although setting can decrease the quality of video up to 0.1 dB in luminance component of texture compared to using ALF alone, but still the PSNR will increase compared to the anchor test model which was discussed in the first setting. In-loop filtering design of mentioned procedure includes first using FBF to remove artifacts due to block partitioning and then filtering images using DF and SAO and using ALF as the last stage of in-loop filtering. Using FBF and ALF filter compared to the previous settings lead the encoder to be more complex. Although this design has the highest complexity, it has the optimal running time and optimal PSNR. So, using ALF and FBF together is recommended for in-loop-filtering design. Implementation of the algorithm for HEVC encoder was shown in Figure 2.18.

4. CONCLUSION AND FUTURE WORK

The work described in this thesis is concerned with the development adaptive in-loop filtering design for 3D-HEVC. The idea of designing new in-loop-filtering was considered because in-loop filtering is the last stage of encoding in HEVC and improvement in quality of an image can improve the total performance. As we mentioned previously, since the filter is placed within a loop, the enhancement not only affects the quality of the output pictures but also the reference pictures. Loop filters have a strong effect on the overall performance of the video coding scheme.

In this thesis, a new model consisting of design of two filters was proposed to reach the optimal running time and quality. An investigation of Fast Bilateral Filter and Adaptive Loop Filter was presented together to implement 3D-HEVC encoder. Interesting features of the proposed algorithm was described and the method was shown to be effective and improved in PSNR compared to the anchor test model.

To reach the maximum PSNR, it is suggested to use the Adaptive Loop Filter. Adaptive Loop Filter is based on Wiener Filter. The main purpose of ALF is to minimize the Mean Square Error. Since MSE is related to PSNR directly as MSE decreases, PSNR is increased logarithmic. Using Adaptive Loop Filter reached up to 2.1 dB of improvement in luma component of depth and up to 0.8 dB improvement in texture. Chroma component does not show much of difference in PSNR. Running time of ALF filter increases 1 to 5 percent compared to anchor test model which can be neglected if real time encoding is not very important subject.

Later on, design of Fast Bilateral Filter with Adaptive Loop Filter was proposed for in-loop filtering in encoder. This design seems to be the optimal solution since the running time decreases up to 20 percent and PSNR decreases up to 0.1 dB compared to using ALF filter.

Fast Bilateral Filter is an approximation of Non-linear Bilateral Filter that keeps track of intensity. In this method, intensity is considered a homogeneous coordinate added to the pixel position to be used in 3D-Gaussian filtering. Fast Bilateral Filter smooths and image and removes the block artifacts of an image which can reduce using of strong Deblocking Filter and uses weak Deblocking Filter resulting in faster processing of encoding. Strong DF uses at most three pixels in the neighbor of a to-be-filtered pixel while weak DF changes at most two neighbor pixels which is the reason of fast encoding in the proposed method.

Although the results have showed the effectiveness of the in-loop filtering design, it could be further developed in a number of ways:

Extending the algorithm to run on the decoder

The proposed algorithm was implemented and tested on the HEVC encoder. It is recommended to implement the proposed algorithm and observe the changes for the decoder.

Using a Block based Fast Bilateral Filter

The Fast Bilateral Filter which was used for this implementation was picture based, which means that it filters the whole image. It is suggested to implement a block based Bilateral Filter. This method seems to be faster than picture-based method since the whole image will not be filtered.

Considering Implementation of other nonlinear filters

The proposed method uses Bilateral Filter as a main idea to implement a Fast Bilateral Filter. It is recommended to implement other types of nonlinear filters for different applications. For example, non linear filters can be designed to improve PSNR of reconstructed video even better than implementation of ALF.

REFERENCES

REFERENCES

[1] F. Institute, *HEVC HM test model*, Last Date Accessed: July 29, 2015, https://hevc.hhi.fraunhofer.de/svn/svn_3DVCSoftware/.

[2] *Nagoya University 3D video test sequence*, Last Date Accessed: July 29, 2015, http://www.fujii.nuee.nagoya-u.ac.jp/multiview-data/.

[3] *Draft of 3D-HEVC Standard*, Last Date Accessed: July 29, 2015, http://phenix.int-evry.fr/jct3v/doc_end_user/current_document.php?id=1878.

[4] I. Rec, "H. 264 & iso/iec 14496-10 avc, advanced video coding for generic audiovisual services," *ITU-T, May*, 2003.

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

[6] J.-R. Ohm, G. J. Sullivan, H. Schwarz, T. K. Tan, and T. Wiegand, "Comparison of the coding efficiency of video coding standards including high efficiency video coding (hevc)," *Circuits and Systems for Video Technology, IEEE Transactions*, vol. 22, no. 12, pp. 1669–1684, 2012.

[7] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, and T. Wiegand, "High efficiency video coding (hevc) text specification draft 8," *JCTVC-J1003, July*, 2012.

[8] M. Wien, *High Efficiency Video Coding.* Springer, 2015.

[9] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *Circuits and Systems for Video Technology, IEEE Transactions*, vol. 22, no. 12, pp. 1649–1668, 2012.

[10] P. Chou, T. Lookabaugh, R. M. Gray *et al.*, "Optimal pruning with applications to tree-structured source coding and modeling," *Information Theory, IEEE Transactions*, vol. 35, no. 2, pp. 299–315, 1989.

[11] G. J. Sullivan and R. L. Baker, "Efficient quadtree coding of images and video," *Image Processing, IEEE Transactions*, vol. 3, no. 3, pp. 327–331, 1994.

[12] F. Bossen, B. Bross, K. Suhring, and D. Flynn, "Hevc complexity and implementation analysis," *Circuits and Systems for Video Technology, IEEE Transactions*, vol. 22, no. 12, pp. 1685–1696, 2012.

[13] A. K. Jain, *Fundamentals of digital image processing.* Prentice-Hall, Inc., 1989.

[14] G. J. Sullivan, J. M. Boyce, Y. Chen, J.-R. Ohm, C. A. Segall, and A. Vetro, "Standardized extensions of high efficiency video coding (hevc)," *Selected Topics in Signal Processing, IEEE Journal*, vol. 7, no. 6, pp. 1001–1016, 2013.

[15] A. Norkin, G. Bjontegaard, A. Fuldseth, M. Narroschke, M. Ikeda, K. Andersson, M. Zhou, and G. Van der Auwera, “Hevc deblocking filter,” *Circuits and Systems for Video Technology, IEEE Transactions*, vol. 22, no. 12, pp. 1746–1754, 2012.

[16] C.-Y. Tsai, C.-Y. Chen, T. Yamakage, I. S. Chong, Y.-W. Huang, C.-M. Fu, T. Itoh, T. Watanabe, T. Chujoh, M. Karczewicz *et al.*, “Adaptive loop filtering for video coding,” *Selected Topics in Signal Processing, IEEE Journal*, vol. 7, no. 6, pp. 934–945, 2013.

[17] X. Zhang, R. Xiong, S. Ma, and W. Gao, “Adaptive loop filter with temporal prediction,” in *Picture Coding Symposium (PCS), 2012.* IEEE, 2012, pp. 437–440.

[18] M. T. Pourazad, C. Doutre, M. Azimi, and P. Nasiopoulos, “Hevc: The new gold standard for video compression: How does hevc compare with h. 264/avc?” *Consumer Electronics Magazine, IEEE*, vol. 1, no. 3, pp. 36–46, 2012.

[19] C.-Y. Chen, C.-Y. Tsai, Y.-W. Huang, T. Yamakage, I. S. Chong, C.-M. Fu, T. Itoh, T. Watanabe, T. Chujoh, M. Karczewicz *et al.*, “The adaptive loop filtering techniques in the hevc standard,” in *SPIE Optical Engineering+ Applications.* International Society for Optics and Photonics, 2012, pp. 849 913–849 913.

[20] C.-Y. Chen, C. Fu, C. Tsai, Y. Huang, S. Lei, S. Esenlik, M. Narroschke, T. Wedi, I. Chong, and M. Karczewicz, “Non-ce8. c. 7: Single-source sao and alf virtual boundary processing with cross9x9,” *Joint Collaborative Team on Video Coding (JCT-VC) of ISO/IEC MPEG and ITU-T VCEG, JCTVC-G212*, 2011.

[21] S. Esenlik, M. Narroschke, and T. Wedi, “Syntax refinements for sao and alf,” *Joint Collaborative Team on Video Coding (JCT-VC) of ISO/IEC MPEG and ITU-T VCEG, JCTVC-G566*, 2011.

[22] C. Chen, C. Tsai, C. Fu, Y. Huang, and S. Lei, “Non-ce8: Constrained alf coefficients,” *Joint Collaborative Team on Video Coding (JCT-VC) of ISO/IEC MPEG and ITU-T VCEG, JCTVC-G214*, 2011.

[23] *Multi-view Test Sequences description*, Last Date Accessed: July 29, 2015, http://sp.cs.tut.fi/mobile3dtv/stereo-video/.

[24] K.-R. Muller, H. Schwarz, D. Marpe, C. Bartnik, S. Bosse, H. Brust, T. Hinz, H. Lakshman, P. Merkle, F. H. Rhee *et al.*, “3d high-efficiency video coding for multi-view video and depth data,” *Image Processing, IEEE Transactions*, vol. 22, no. 9, pp. 3366–3378, 2013.

[25] F. D. IV, *The Use and Misuse of Color in Web Pages*, Last Date Accessed: July 29, 2015, http://nemesis.lonestar.org/reference/internet/web/color/ntsc_primer.html.

[26] I.-K. Kim, K. McCann, K. Sugimoto, B. Bross, and W.-J. Han, “Hm9: High efficiency video coding (hevc) test model 9 encoder description,” in *9th JCT-VC Meeting, Switzerland*, 2012, pp. 10–11.

[27] S. Paris and F. Durand, “A fast approximation of the bilateral filter using a signal processing approach,” in *Computer Vision–ECCV 2006.* Springer, 2006, pp. 568–580.

[28] M. Naccari and F. Pereira, “Adaptive bilateral filter for improved in-loop filtering in the emerging high efficiency video coding standard,” in *Picture Coding Symposium (PCS), 2012*. IEEE, 2012, pp. 397–400.

[29] A. Kesireddy, “A new adaptive trilateral filter for in-loop filtering,” Ph.D. dissertation, Purdue University, 2014.

[30] *How to Build your First HEVC stream*, Last Date Accessed: July 29, 2015, https://codesequoia.wordpress.com/2012/11/04/make-your-first-hevc-stream/.

APPENDIX

APPENDIX. PROCESS OF BUILDING HEVC STREAM

The main idea of this thesis is to encode the video with HM-6.2 and produce the HEVC stream. In this process Adaptive Loop Filter is implemented and the resulting frames are compared to the frames of encoding HEVC Test Model anchor (HM-6.2). HM is provided as a source code to build on different platforms.Windows based of Visual Studio Command Prompt (2010) is the tools used. In addition, a Linux version of software is available for use. System should have few YUV mutiview test sequences and as input and YUV Player to view the encoded video. The following steps show encoding the video with HM-6.2 [30].

1. **Download the source code**

   Download the source code with the desired version of 3D HEVC [1].

2. **Build it!**

   Start Visual studio command prompt. Change the directory to encoder located in trunk/build directory and build HM vc10.This version is latest version of visual studio(2010) offered by HM test model.

   ```
   % msbuild /p:Configuration=Release HM vc10.sln
   ```

3. **Encode it!**

   You can find the sample of configuration files in cfg folder. After copying any configuration file to the folder where the TAppEncoder.exe is located, you can run the encoder. Note that in configuration file settings related to the name of the input yuv file and other settings such as QP can be set. TAppEncoder.exe is located in Trunk/bin/vc10/win32/release. Enter the following command in Command Prompt 2010 window:

   ```
   %TAppEncoder.exe -c test.cfg
   ```

4. **Output in YUV player** Using the above steps we can encode a YUV input file and it creates out.hevc, which is the HEVC stream. The final output is a YUV file which is decoded from the HEVC stream and the output of calculating PSNR and other information can be stored in a text file.