A new adaptive colorization filter for Video decompression

Vaughan H. Lee*, Yuko Roodt† and Willem A. Clarke†
Hypervision Research lab
School of Electrical Engineering
University of Johannesburg, P. O Box 524, Auckland Park, South Africa
vaughan.h.lee@gmail.com*, {yukor†, willemc†}@uj.ac.za

***Abstract*—HD content is more in demand and requires a lot of bandwidth. In this paper, a new real-time adaptive colorization filter for HD videos is presented. This approach reduces the required bandwidth by reducing non-key frames in the HD video sequence to grayscale and colourizing these frames at the decompression stage. Additionally this technique determines the frame status based on the image information.**

Index Terms: HD video compression, adaptive colorization, graphics processing unit (GPU)

I. INTRODUCTION

The multimedia industry requires elaborate compression schemes to reduce the massive volume of data representing multimedia content, in particular video streams. In video streams, there is a significant amount of redundant data which is removed. The reduction achieved is not sufficient to successively merge high definition (HD) graphic content into more consumer applications. However, there is a demand for HD multimedia content [1]. HD multimedia content caters for better realism, by providing a sharper more clear image. The consumer demand for HD content can therefore be expected to continue to increase. The growing demand for HD video content present challenges such as transmission latencies due to the volume of data. The large volumes of data unnecessarily burden networks. HD video content has significantly more data than standard definition (SD) and enhanced definition (ED) content. The volume of video data in particular should be further compressed to better align to consumer trends. Some high bandwidth applications stimulating the HD trend are digital television (HDTV), digital video by satellite (DVS), 3D-Video games and 3D blue-ray DVDs.

HD video sequences require significant processing power to decode [3]. The graphics processing units (GPUs) are increasingly becoming available in every personal computer. Modern GPUs have a parallel architecture, specifically designed for image processing. The modern GPU is programmable, presenting itself as additional processing hardware for applications. To this end, the GPU is used as a work horse for a growing amount of applications that suit single instruction multiple data (SIMD) paradigm.

The most relevant research is briefly discussed. Shen et al. implement motion compensation for a video decoder using GPU [2]. In this work, the video decoding tasks are divided such that a portion of tasks are executed on the CPU and the remainder on the GPU in a cascading manner. The implementation achieved real-time decoding at 720p, which is 1280 by 720. The GPU tasks were motion compensation (MC) from precomputed motion vectors, colour space conversion (CSC) and finally rendering the frame. The experimental results showed that the trap video sequence at $720p$ video sequence achieved $31.3fps$. Other experimental results given showed that lower quality video sequences with lower quality achieved a faster frame rate.

The entire motion compensation algorithm, as well as the motion estimation architecture cannot be offloaded to the GPU. Appropriately, motion vector estimators are computed on the CPU, while the GPU refines the pre-calculated predictors. In this research, Schwalb et al. report the latency is reduced [6]. Their contribution has two interesting outcomes, a small diamond search mapped to a parallel implementation for execution on the fragment processing unit. The second contribution asserts that CPU and GPU could be used together simultaneously. The theoretical performance gain anticipated was not achieved due to the high number of random memory accesses.

In work more closely related Kumar and Mitra made considerable effort to explore a new approach to improving compression of video sequences [10]. In this work, a video sequence is compressed by dropping the chrominance channels of certain frames. A full colour frame consists of three 2-*D* matrices, where the matrices represent pixel intensity. The compression achieved by dropping two of these channels every so often achieves additional compression. At the decoder side, the luminance only frames are colorized. Their approach using colorization to further compress a video sequence is integrated into MPEG-2 as a module, making their work interesting. A penalty noted by Kumar in this approach is the processing latency, which is considerable.

In our work, a new adaptive compression scheme is presented and a process illustration is given in Figure 1. The proposed scheme reduces the required HD video bandwidth while maintaining acceptable colour. The video sequences considered in this work are full HD video sequences, that is 1920 by 1080. Here the encoder adaptively encodes frames in sub-sampled colour or grayscale, a decision based on the image entropy of the current frame.

Our proposed scheme ensures that maximal compression is achieved when frames contain minor differences. However rapidly changing scenes will likely be more frequently coded in

Telkom SA is acknowledged for their role as sponsor to this research.

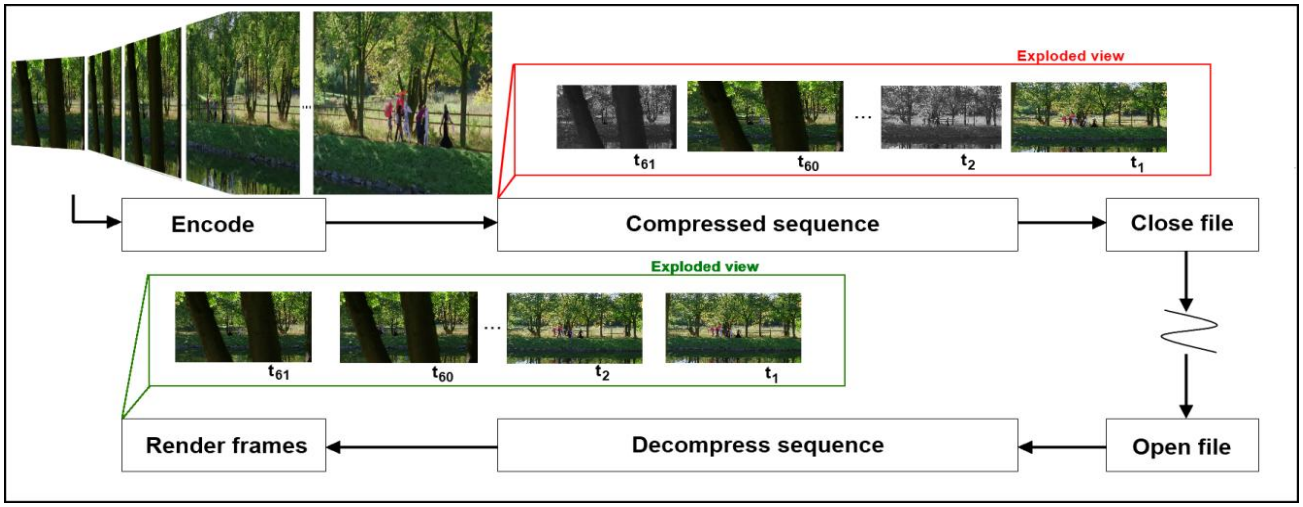

Figure 1: Video Colourization Illustration

colour, making the approach robust to video content. The primary contribution of this research is at the decoder side. The grayscale frames are colorized at the decoder side, without use of any particular group of pictures scheme. A new colorization filter is used to colourise the frames, enhanced with adaptive filter updates. The colorization filter enhancement is received through motion compensation (MC). The MC is used to update the filter, allowing the filter to change to the content of the video. Finally, the approach makes optimal use of the CPU and the GPU for better utilisation of system resources. The paper is organised as follows. The theory section briefly introduces video compression. The colorization framework is discussed followed by the adaptive colorization algorithm. The proposed compression and decompression scheme is implemented, a discussion details the operation thereof. The experimental results are closely examined followed by concluding remarks.

II. THEORY

A. *Overview to Video Compression*

Video compression is the reduction in the volume of data representing the video information of interest. There are various types of compression schemes, primarily lossy or lossless compression. Lossy compression is usually more practical as better compression ratios are achieved, however information that is discarded cannot be reconstructed. While in lossless compression the original information can be exactly reconstructed from the compressed sequence. There are temporal and spatial redundancies in video, which are removed in compression schemes. Temporal refers to those redundancies which exist between consecutive frames, while spatial redundancies exist within a frame. The combined use of several techniques are employed to remove these redundancies achieve good results.

One compression approach is to apply a transform to a macroblock, with the desired effect of removing the correlation between the pixels. The coefficients which emerge after applying the transform can then be quantised and encoded, thus discarding data points that contains the least amount information.

The exploitation of temporal redundancies require the use of motion compensation techniques, or 3*D* transforms. The basic principle of operation is that at the encoder the motion of pixels are predicted. The exact pixel motion is then compared to the predicted pixel motion, the resulting error together with the motion vectors are encoded. As a last example of a compression technique, sub-sampled formats are used. There are three components in colour images, intensity and 2 chroma (colour) components. The human vision system (HVS) cannot detect the colour differences as easily as intensity changes, thus some colour is discarded. In sub-sampled video format 4: 2: 2, there are 4 intensity samples for every 2 samples for both colour components.

B. *Frame Colorization*

The estimation and addition of chrominance components to luminance is a technique known as colorization. Colorization is interpreted as an image filter which transforms the input luminance (grayscale) image into a color image. The operational principle behind image colorization is described as; image features from the grayscale image are used to assist estimation of chrominance (chroma) values to be added to the grayscale image. Identifying image features using statistical methods are processor intensive, as a result image comparisons are the suitable alternative for real-time applications. An image comparison will require an image of the same type to be colorized, together with the desired output image. Image features are used to establish a loosely unique pattern between the input and the desired image. To increase certainty more features can be considered. This training input and desired output image are used to initialize the colorization filter, thereafter, sequential input images to the filter are then colorized in a similar manner.

III. DESIGN AND IMPLEMENTATION

A. *Proposed Systems Process Overview*

The process overview of the video compressor and decompressor developed in this work is depicted below in Figure 2. In Figure 2, first focusing on the raw video block depicted on the left is the uncompressed video sequence which is to be coded. The pixel motion in this video sequence is

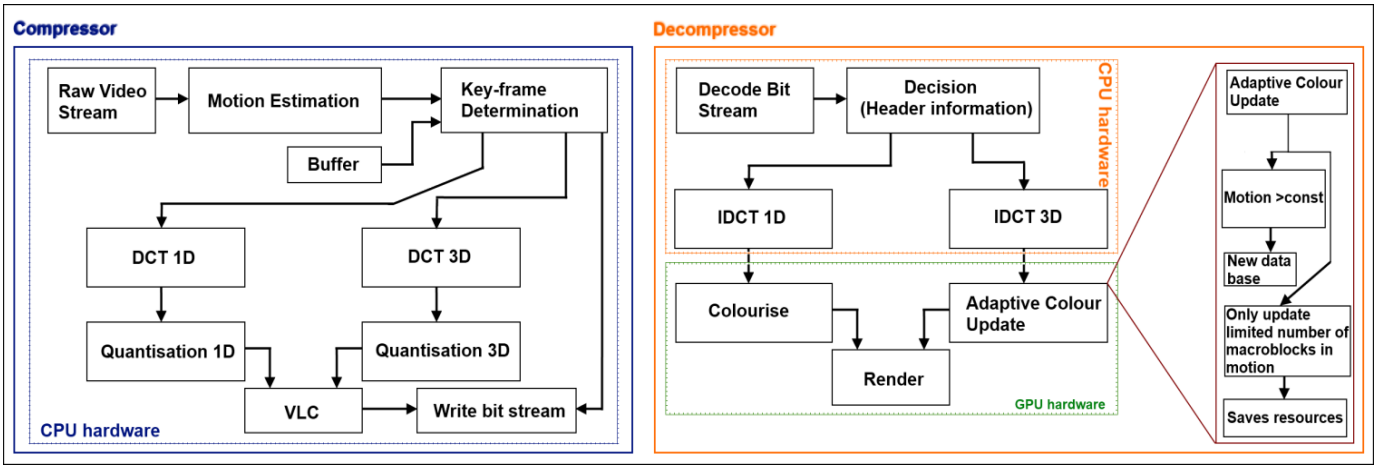

Figure 2: Process Overview

determined by the first function module on a forward frame basis, no backward prediction is used. Each frame is divided into macroblocks, where block motion is then detected. Once the blocks that are in motion are determined the frame entropy is calculated.

The entropy gives an indication of the information content in the image, which is used to determine the status of the frame. The frame status can either be regarded as a key frame or a non-key frame. The decision is based on the entropy difference between the past frame and the current frame, thus either encoding full color components or simply the luminance component of that frame only. Upon the decision outcome, the macroblocks that undergo motion are decorrelated using the discrete cosine transform (DCT) and then quantized.

The quantization level and the maximum entropy difference influence the bit rate, however these values are not related. The entropy is adjusted automatically based on information content, while the bit rate is specified. These bits are then variable length encoded, finally ready for transmission or storage. The decoder is the side that decompresses and finally renders the video frames is depicted on the right of the encoder in Figure 2. The decoder has several function modules, the first determines the inverse variable length codes (IVLC). The status of the frame is then determined from the header information. The status of the frame determines which function module to follow. In the case of key frame status, the inverse discrete cosine transform (IDCT 3D) is applied for all 3 channels, followed by updating of the colorization filter database. The colorization filter is reinitialized during this function. The colorization filter is configured to receive new non-key frames of similar content to the update.

This approach renders the colorization filter as adaptive, thus limiting the error associated to pixel drift and feature movement. In the second case, where the frame is determined as a non-key frame, color components will need to be estimated by the colorization filter. The colorization filter module is written for GPU hardware, in section III.E the filter is described in detail. Once the non-key frame is colorized, the image is rendered to the screen.

B. *Encoder Architecture*

The motion for each macroblock is determined by comparing the current frame I_t to the next frame I_{t+1}. The macroblocks are compared by calculating the mean absolute difference (MAD),

$$MAD = M(I_t - I_{t+1}) = |I_t - I_{t+1}| \quad (1)$$

The macroblocks that do inhabit motion are then replaced with the updated pixels. Thus only the macroblocks that are in motion are decorrelated. The decorrelated macroblocks are then quantised and encoded, effectively a difference update.

The proposed compression scheme intends on delivering a variable bit rate data stream. The entropy of an image reveals the average information content per symbol, effectively entropy is a function of occurrence probabilities of source symbols. To calculate the entropy, the source symbol count for each of the 256 quantization levels (seen as source symbols), where 0 is black and 255 is white. The probability (ρ) of occurrences of each of these quantisation levels are then,

$$p_S(k) = \frac{h_f(k)}{M.N} \quad (2)$$

where the subscript I denotes the image and the information content per source symbol (Q) is,

$$Q_f(k) = log_2\left(\frac{1}{p_f(k)}\right) = -log_2\left(p_f(k)\right) \quad (3)$$

The entropy (H) is determined using,

$$H_f(k) = \sum_{k=1}^{k_{max}} e_f(k) = \sum_{k=1}^{k_{max}} p_f(k)Q_f(k) \quad (4)$$

When summing the entropy per quantisation level for a given image 77% of the total image entropy describes most significant information [9]. The experimental value used to determine the entropy threshold is determined experimentally as 95%, should the entropy change be less than this the difference will change enough content significantly to effect the colorization process. This is determined as,

$$Decision = \begin{cases} Key\ if\ I_{t-1} > 1.05I_t \ \| \ I_{t-1} < 0.95I_t \\ Non-key\ otherwise \end{cases} \quad (5)$$

The final step is bit stream encoding, traditional run-length encoding (RLE) was selected. RLE presents an effective yet quick enough implementation, there are numerous techniques that are more efficient, however these distract from the scope of the proposed scheme.

C. Decoder Architecture

The decompressor cannot be simplified by determining the inverse of the compressor techniques. The crucial proposal of our work is based on the adaptive colorization, which is not simply the inverse of the compressor. The bit stream is decoded, in a sense unpacking the data stream. The frame header contains information to reduce the computational overhead at the decoder. It should be noted that the header can contain significant bit representation and cannot be compressed with ease, only critical information is necessary here. The frame status is retrieved from the frame header, which directs the stream to the desired function module. Once the approximated stream has been recovered through the IDCT, the stream is prepared for offloading to the GPU. A note worthy consideration is the time penalty in pushing data between the CPU and the GPU, effectively nullifying any potential processing gain. The decompressor is designed with this consideration, once the correlated difference image data is prepared, the data is not transferred back the CPU for additional processing.

The colorization filter should be designed with stream processors in mind, careful consideration must be given to the architecture. The GPU program can be simplified to a kernel function, which is applied to every pixel element. Further consideration is given to the number of control structures which severely retard the performance. The use of a neighborhood does reduce the performance, however the sacrifice is less than approaching the dimensionality problem. The neighborhood is constructed by shifting the image, then performing the quick smallest first match test. The shifting of the image pixels is structured, all check positions are predetermined.

D. Proposed Adaptive Colorization Algorithm

The proposed algorithm requires initialization, a process whereby the image features in the search parameter are updated as well as the corresponding chroma values. Initialization occurrence is frequent enough to change with the data stream, as determined through the image entropy calculation. Our proposed algorithm is adaptive to the stream content, ensured by considering a percent shift rather than an experimentally determined value.

The algorithm proposes two types of updates. The first is a complete frame for initialization. A complete frame is used when motion exceeds several blocks, but the primary purpose is for new scenes. A completely new scene for initialization is a natural approach to ensure optimal operation of the colorization filter. The new scene update ensures that stationary scenes are greatly compressed.

The scenes that contain high motion threaten to undermine any additional compression gained. The second initialization approach is utilized to minimize the inefficient complete filter re-initialization caused through high motion scenes which require more frequent updates. Here the update container is accessed through motion detected blocks. The blocks determined to be in motion are encoded in grayscale as well as the corresponding chroma values. This approach limits the neighborhood search.

E. Proposed Algorithm Implementation

The inputs into the filter are $2D$ textures, the colorization filter update content must be contained herein. The frames to be colorized are naturally $2D$ images, requiring no significant changes. The use of textures greatly simplify the GPU programming model; adapting the representation of the updates for the colorization filter presented mapping difficulties. The primary intention is to reduce the bit rate of video streams, without excessive computational complexity.

The features used in our colorization filter were the locations of each pixel in the input image, together with the intensity values of those pixels. The corresponding chrominance values for each pixel is also used. The intensity values are limited to 256 quantization levels only, which introduces a problem of non-uniqueness between intensity and the possible corresponding chroma values. Additionally, the database grows to a very large size. In order to limit searching through a large database as well as the likelihood of selecting the incorrect chroma combination an assumption is asserted. The colorization filter assumes that the images into the filter are similar to the image set used to train the colorization filter. This assumption is only valid when the input images are those sequential images of a video. In asserting the assumption, certainty about the more probable combination of chroma can be increased by constructing a neighborhood surrounding each pixel. The values within this neighborhood are tested when estimating the chrominance component, thus limiting the search database. The limited search region reduces the otherwise unavoidable problem of dimensionality.

A single adaptive colorization filter was proposed, with two types of updates. These updates as discussed ensure a closely correlated processed output image even in the particularly difficult scenarios of high motion. The first update type referred to as database initialization which completely resets the database. The implementation of this module is given in pseudocode,

```
Start database_initialisation:
   desired = Input
   Υ^A_{i,j} = luma(Input)
   get features, for ∀ ρ
   configure database
End module
```

Figure 3: Type 1 Colorization Update, Database Initialization

```
Start update_container:
  B = recieve nonkey frame
  if current ρ == 0
    no ρ update
  else update this pixel in database → feature update
  end
End module
```

Figure 4: Type II Colorization Update, Update Container

The update container which is the motion corrected values must be updated is shown in Figure 4.

The colorization filter is implemented as a function given in pseudocode in Figure 5, here the colorization filter has a training input image A and the desired colourful image A'. A neighbourhood $\mathcal{N}$ is constructed around each pixel ρ to have chrominance determined. The pixels ρ in an image $\mathcal{S}$, $\mathcal{N} = \{\mathcal{N}_k \mid \forall\, k \in \mathcal{S}\}$ where $\mathcal{N}_k$ is the set of pixels neighbouring pixel ρ where ρ in $\mathcal{N}_k$. The luminance Υ from the sequential video images A is transferred to the output processed image. The chrominance components, C_r and C_b are estimated as the first smallest match in the neighbourhood.

Function Colorization start:
  $output_{i,j} = \Upsilon^{B}_{i,j}$
  given Υ_B, *for* $\forall\, \rho$ *in* A *do*:
  if $\rho_{i,j} \in \Upsilon_B == \Upsilon^{A}_{i,j}$
    $output_{i,j} = \rightarrow feature\ output$
  else *construct neighbourhood,*
    $for\ \forall\ \rho_{i,j}\ in\ neighborhood\ \mathcal{N}_k\ do$:
      if $\rho_{i,j} \approx neighbourhood\ \mathcal{N}^{k}_{m,n}$,
        $output_{i,j} = neighbourhood\ \mathcal{N}^{k}_{m,n} \rightarrow feature\ output$
      else if $\rho_{i,j} \neq any\ \rho_{i,j}\ in\ neighbourhood\ \mathcal{N}^{k}$
        $average\ pixel\ not\ matched \rightarrow feature\ update$
      end
  end
End Colorization

Figure 5: Colorization Filter

Finally the proposed algorithm is,

Start proposed algorithm:
  if *Status* $\neq$ *TRUE*
    Initiate update_container
    Call Colourisation function
  else
    Initiate database_initialisation
    Call Colourisation function
  end
Loop algorithm for $\forall\, \mathbf{I_i}$ **in video**

Figure 6: Proposed Algorithm Adaptive Colourisation Filter

where the sub modules have been discussed and are separated for the purpose of this discussion.

IV. Experimental Results and Discussion

A. Experimental Overview

The video sequences used in the experiments are from a standard collection of HD 1920p sequences. The system specifications are 3GHz Athlon x2550 with 4Gb RAM and a Nvidia 260 GTX GPU on which the experiments were performed. The proposed algorithm was used to encode the video and decode the video. The decoded video was then compared to the original uncompressed video. The experimental parameters were the colorization neighborhood size and the threshold for blocks in motion used by the update container module. The neighborhood size was set at 2, and update blocks in motion threshold was set at 2 also. These values were varied, increasing these values slowed the encoder speed.

B. Experimental Results

The experiments show that good reconstruction is achievable. The cross-correlation between the original sequence and the decoder output is averaged and shown in Table 1 together with the peak signal to noise ratio (PSNR). The cross-correlation is desirable near 1, where 1 is exact. The PSNR should be as large as possible. The sunflower sequence was not colorized very well, this has been attributed to the large motion in the first 100 or so frames.

Table 1: Cross-correlation and PSNR of several video sequences

Video sequence	Cross-correlation mean ρ	PSNR mean
Touch Downpass	0.8663	24.5228
Sun Flower	0.5551	15.1920
Crowd Run	0.8749	16.2258
Snow Mountain	0.9574	26.0402
Controlled Burn	0.9628	23.1794

A plot of several frames is shown in Figure 7.

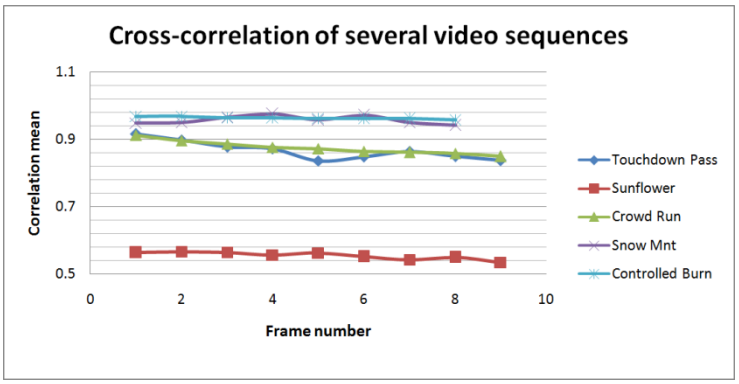

Figure 7: Frame cross-correlation for several frames of different sequences

The decoder frame rate (fps) achieved was not desirable, shown in Table 2. The frame rate for the decoding modules is indicated separate from the colorization frame rate. The decoder implementation was slowed down greatly by the IDCT module. The IDCT module was not pruned for real-time, thus negatively effecting the performance of the decoder. The colorization frame rate is well above real-time with the slowest rate of 89.2 fps. The desired bit rate was not achieved, shown in the last column of Table 2, showing the bit rate per second. The bit rate not being reduced effectively is attributed to the encoder. The encoder implemented has proven not to be effective as desired,

Table 2: Frame and bit rate of experimental set

Video sequence	fps	Colourization fps	Mbps
Touch Downpass	2.271	138.85342	101.19168
Sun Flower	2.414	99.90049	93.69600

Crowd Run	2.471	101.68966	104.93952
Snow Mountain	2.079	89.23637	73.05216
Controlled Burn	2.324	113.96565	86.13888

The bit rate when compared to the uncompressed stream is sufficiently good.

V. CONCLUSION

In this work an adaptive colorization filter was proposed. The filter aims to reduce the bit rate (improve compression) through dropping chroma channels of specific frames completely. The chroma channels dropped are estimated at the decoder side by means of image comparisons. Further we propose that only frames determined to be key frames should be encoded, instead of using a predetermined pattern. The information content in each frame was determined through entropy. The information content determines when a frame is encoded as a key frame or non-key frame.

The experimental results show that correlation between the actual video sequence and that decoded can vary quite significantly. The correlation is acceptable, however the overall subjective assessment reveals that a deblocking method should always be implemented. The proposed filter can partially correct errors associated with IDCT, but a deblocking filter should still be applied. It was also confirmed that the proposed method cannot be applied to all types of sequences, too much scene change would drastically effect the bit rate.

The colorization algorithm performance speed is well within real-time requirements while our specific encoder lacking speed. A more efficient decoder would correct for the slow decoding speed. Future work will aim to improve the decoder to performance through use of a fast IDCT or other fast transform. Future work will also use a variety of image filters together with the colorization filter to improve the image sharpness and clarity. A super resolution filter would work well with a database of past images, which can complement the colorization process.

ACKNOWLEDGEMENTS

V.H.L thanks the researchers at HPCV Lab for weekly discussions.

REFERENCES

[1] Ali Aghagolzadeh, Saeed Meshgini, Mehdi Nooshyar, Mehdi Aghagolzadeh, "A Novel Video Compression Technique for Very Low Bit-Rate Coding by Combining H.264/AVC Standard and 2-D Wavelet Transform", ICSP2008 Proceedings.

[2] Guobin Shen, Guang-Ping Gao, Shipeng Li, Heung-Yeung Shum, and Ya-Qin Zhang," Accelerate Video Decoding With Generic GPU", IEEE Transactions on circuits and systems for video technology, vol. 15, no. 5, may 2005.

[3] Cebrail Taşkin and Serdar Kürşat Sarikoz, "An Overview of Image Compression Approaches", The Third International Conference on Digital Telecommunications, 2008.

[4] M.C. Kung, Oscar C. Au, P.H.W. Wong, Chun Hung Liu, "Block Based Parallel Motion Estimation Using Programmable Graphics Hardware", ICALIP 2008.

[5] P.C.Shenolikar, S.P.Narote, "Different Approaches for Motion Estimation", International conference on Control, automation, communication and energy conservation, 4th-6th June 2009.

[6] Martin Schwalb, Ralph Ewerth, and Bernd Freisleben, Member, IEEE, "Fast Motion Estimation on Graphics Hardware for H.264 Video Encoding", IEEE Transactions on multimedia, Vol. 1. No. 1, January 2009.

[7] Mauricio Alvarez, Esther Salam´ı, Alex Ram´ırez and Mateo Valero, "HD-VideoBench. A Benchmark for Evaluating High Definition Digital Video Applications", IEEE, 2007.

[8] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, David H. Salesin, "Image Analogies", 2001.

[9] Markos Mentzelopoulos and Alexandra Psarrou, "KeyFrame Extraction Algorithm using Entropy Difference", MIR'04, Oct 15-16, 2004, NY.

[10] Ritwik Kumar, Suman K. Mitra, "Motion Estimation based Color Transfer and its Application to Color Video Compression", 2008.

[11] Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra, "Overview of the H.264/AVC Video Coding Standard", IEEE Transactions on circuits and systems for video technology, Vol 13, No. 7, July 2003.

[12] Roberto R. Osorio and Javier D. Bruguera, "High-Throughput Architecture for H.264/AVC CABAC Compression System", IEEE Transactions on circuits and systems for video technology, Vol 16, No. 11, Nov 2006.