Improving Compression Ratios for High Bit-Depth Grayscale Video Formats

An Ho, Alan George, Ann Gordon-Ross
NSF CHREC Center, ECE Department, University of Florida
327 Larsen Hall, 968 Center Drive, Gainesville, FL 32611
352-392-5225
{aho,george,ann}@chrec.org

***Abstract*—Since increasing demand for high bit-depth video places large demands upon resources, such as communication bandwidth as well as memory and storage capacity, research into improving the compression ratio (CR) for these videos is critically important. Most conventional video encoders are not amenable to high bit-depth format, so this paper presents novel preprocessing methods designed to improve CR of high bit-depth grayscale video by transforming raw data such that the video can be compressed using conventional encoders. We present five preprocessing methods: filtering, region of interest (ROI), factoring, SuperFrame, and bit-stream splitting (BSS). Results show tradeoffs for each method, with respect to CR and data quality. The greatest increases in CR are obtained using SuperFrame, BSS, and factoring, and combining these methods increases CR even further. With our focus upon tradeoffs between CR and data quality, our new methods, results, and analysis enable system designers to select appropriate preprocessing method(s) based upon their specific system and mission requirements.**

TABLE OF CONTENTS

1. INTRODUCTION

Increasing demand for higher data quality is prevalent in many computing domains, especially aerospace and defense domains, where this increase enables improved data analysis for critical and sensitive applications, such as surveillance systems, object tracking, space science, etc. One way to improve the video quality is to increase the video format's bit-depth from the typical 8-bit consumer video to a higher bit-depth (e.g., any video containing more than 8 bits per pixel, such as 14- or 16-bit), which vastly increases the resolution and data information per video frame. However, the tradeoff for increased bit-depth is increased resource requirements, such as memory storage, computational demands, communication bandwidth, etc., which may preclude usage in highly resource-constrained systems, such as satellites and drones. For example, 100 frames of 8-bit 256×256 video contains (100 × 256 × 256 × 8) = 52,428,800 bits of raw data. A video with an equivalent number of frames and the same resolution but in 16-bit format contains (100 × 256 × 256 × 16) = 104,857,600 bits, which doubles the memory requirements as compared to the 8-bit video.

One common method to mitigate this increased resource requirement is to compress the raw data to a smaller size using existing video encoders, such as H.264 [1], MJPEG [2], or VP8 [3]. An encoder's effectiveness is measured using the compression ratio (CR), which is the uncompressed data size divided by the compressed data size. Lossless encoders retain all existing raw data exactly when the compressed data is decoded, but retaining this information severely limits attainable CR. Lossy compression enables larger CRs, but the decoded data are not exactly the same as the encoded data. This loss can be measured using the Root Mean Square Error (RMSE), which is a common metric used to quantify data loss. Lower RMSE values represent higher decoded video quality (less data loss), with lossless compression achieving an RMSE of zero (the decoded data are identical to the original raw data).

A major challenge in using existing encoders for high bit-depths is that these encoders typically only support 8-bit video formats, and thus they are not amenable to high bit-depth videos. Even though a few existing encoders support high bit-depth grayscale video (e.g., JPEG-LS [4], FFV1 [5], JPEG2000 [6], FFVhuff [7]), an in-house analysis of these encoders showed low CR, ranging from 1.0 to 1.85.

There are several potential solutions for improving the CR, including architecting new encoders. However, architecting new encoders can be an immense undertaking for a special-purpose need, which may explain the limited availability of suitable encoders [8]. We propose a more practical solution that uses preprocessing methods to alter the video's format, creating processed data that are more amenable to existing encoders. In this work, we propose and evaluate five video preprocessing methods to transform high bit-depth videos, making the processed data amenable to existing encoders. Our preprocessing methods include: filtering, region of interest (ROI), factoring, SuperFrame, and bit-stream splitting (BSS). These methods offer different tradeoffs between CR and RMSE to enable system designers to

978-1-4673-7676-1/16/$31.00 ©2016 IEEE

choose an appropriate method based on application and system requirements [9].

Filtering is a preprocessing method that attempts to reduce the noise in the data, which reduces the video's entropy. Entropy is a measure of the quantity of information contained in an image, where lower entropy results in a higher potential CR, and thus filtering can theoretically improve a video's CR. We evaluated several noise-reduction filtering algorithms, including Gaussian, Median, Average, and Wiener. Since filtering modifies the content of the video, filtering is inherently lossy, and thus the processed data already contains some loss in quality.

ROI identifies the critical video-frame regions and maintains these regions' qualities using lossless compression, while using lossy compression on the remainder of the video frame in order to isolate the data loss to non-critical data. This selective region compression achieves higher overall CR while maintaining the quality of the critical data and varying the quality of the non-critical data.

Factoring reduces the bit-depth by attempting to remove only bits that provide little information with respect to the overall data quality. Factoring is similar to the quantization step in JPEG encoding [10].

SuperFrame is a method that converts a large stream of video frames into a single (supersized) frame, making the data more amenable to popular image compression algorithms, such as JPEG2000. Using SuperFrame enables compression of all pixel values within a single supersized frame, which transforms temporal redundancy into spatial redundancy and increases CR potential.

BSS is a novel video preprocessing method that splits the video's bits into smaller 8-bit partitions that can be compressed using existing 8-bit encoders (e.g., H.264, MJPEG, VP8). For example, a 16-bit video can be split into two separate 8-bit videos, each containing the same number of frames, where one partition contains the upper bytes of each frame and the other partition contains the lower bytes.

We thoroughly evaluated each preprocessing method's CR and RMSE for different video scene types. Since these requirements can be mission-specific, for evaluation and comparison purposes, in this paper we target an average CR of 10 or more with an RMSE of 15 or less. Using a 16-bit lossless encoder FFV1 as a baseline, our results show that filtering resulted in small CR improvements of 1.01× to 1.04× with a large increase in RMSE. A simple quadrant-based ROI method that identified one quarter of the frame as critical data increased the overall CR by 2.10×, with an RMSE of zero for the ROI quadrant (i.e., perfect data quality), but large increases in RMSE for the other quadrants. SuperFrame using lossless JPEG2000 resulted in CR increases of 1.63×. BSS resulted in CR increases of 1.74× and 15.4× using lossless and lossy compression, respectively. Combining factoring and BSS resulted in a CR increase of 22.4× using lossy compression, with a small RMSE of less than 18.5.

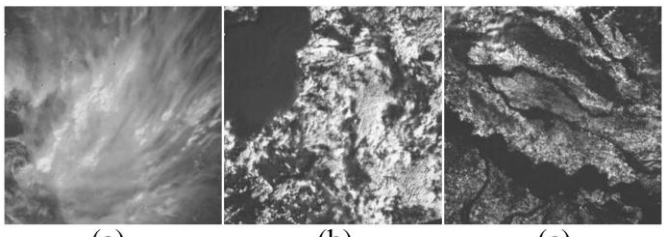


Figure 1: Representative OPIR frames/scenes from our simulated 14-bit video test set for varying cloud cover situations: (a) Cloud001, (b) Cloud002, and (c) NoCloud001

As aerospace missions demand higher bit-depth video, it is critical to improve data compression in order to be able to deliver video information efficiently. However, given the high data-integrity requirements of aerospace and defense applications, the compression methods used must meet critical CR and RMSE requirements for the mission. In this paper we present new methods, results, and tradeoff analyses with different preprocessing and video encoding methods, enabling designers to quickly evaluate and select an appropriate method given system constraints and application requirements.

2. BACKGROUND AND METHODOLOGIES

Given the disparity between expected and required data quality and between the level of information for consumer products (e.g., personal video camera, television data) and specialized space applications, vastly different video formats and processing methods may be required. In this section, we summarize these specialized requirements with respect to the high bit-depth video test set used in our evaluations, and the various metrics and tools used to compare and analyze the effectiveness of our proposed preprocessing methods.

Evaluated Video Test Set

For this study, we evaluate three different high bit-depth videos, provided by the Air Force Research Laboratory (AFRL) Space Vehicles Directorate, which are simulated, video data in Overhead Persistent Infrared (OPIR) imaging with 14-bit grayscale raw video in little endian format. Since the files are stored as raw video, even though the sensor data is recorded as 14-bit, each pixel value contains 16 bits, with two zero bits padded automatically at the most significant bit position [11]. A special characteristic of these videos is that the videos are recorded at a high frame rate of greater than 100Hz, whereas a consumer video would typically be recorded at 30Hz or 60Hz. This high frame rate adds to the memory, storage, and bandwidth requirements.

Figure 1 depicts a representative frame/scene from each of these three videos, showing various possible expected scenarios from space applications that are observing Earth terrain. The videos show different cloud cover variations,

with Figures 1a (Cloud001) and 1b (Cloud002) showing different types of cloud cover, and Figure 1c (NoCloud001) with no cloud cover. Cloud001 is a more uniform scene, with most of the video containing a small range of gray levels. Cloud002 and NoCloud001 show more complexities in the video, with more drastic changes within certain portions of the frame. This range of complexities will impact the effectiveness of the video compression, and vary the achievable CR and RMSE of each video, based on our different preprocessing methods.

Evaluation Metrics

We use several metrics to measure the effectiveness of video compression and our proposed preprocessing methods. CR quantifies the reduction in data size, which is the uncompressed file size divided by the compressed file size:

$$CR = \frac{uncompressed\ file\ size}{compressed\ file\ size} \quad (1)$$

Higher CR values indicate a smaller compressed file size, which reduces memory, storage, and communication bandwidth requirements.

RMSE is used to measure the quality of a video after compression, and calculates the difference in pixel values in the compressed file against the original pixel values from the raw, uncompressed file. RMSE is calculated as:

$$RMSE = \sqrt{\frac{1}{WHL}\sum_{x=0}^{W-1}\sum_{y=0}^{H-1}\sum_{z=0}^{L-1}[f'(x,y,z) - f(x,y,z)]^2} \quad (2)$$

where *W* and *H* are the width and height of the video frame, respectively, *L* is the number of frames in the video, $f'(x,y,z)$ is the compressed video pixel value at the *x,y* position within the *z-th* frame, and $f(x,y,z)$ is the raw video pixel value. However, RMSE does not distinguish between loss of noise and loss of information and simply represents the error, unlike other common metrics such as Peak Signal to Noise Ratio (PSNR), which attempts to quantify the visual effects of data loss and may not make sense for analysis applications.

Video can be compressed using either lossless or lossy compression. In lossless compression, the compressed file has an RMSE value of zero, meaning there are no changes in any of the pixel values (i.e., all data quality is retained). However, this zero RMSE requirement severely limits attainable CR. Lossy compression increases CR, but the tradeoff is reduced data quality. Lossy compression results in RMSE values greater than zero.

Tools

We used MATLAB as the primary tool to create and test our preprocessing methods. Filtering was performed using the *imfilter* MATLAB function, which allows the use of built-in filters, such as Average, Median, Gaussian, and Wiener.

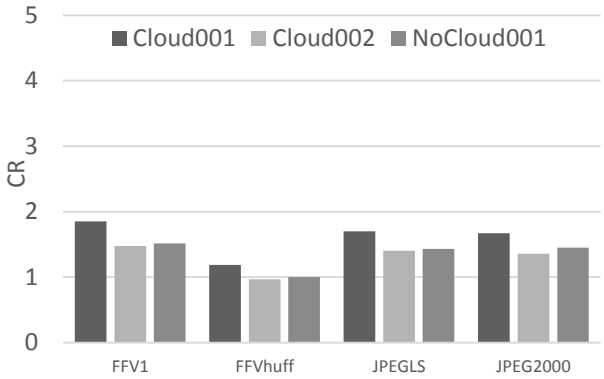


Figure 2: Baseline CR for each video using existing 16-bit encoders for lossless compressions.

MATLAB was also used to create the scripts for factoring, SuperFrame, and BSS. For our video encoders, we used the open-source application FFMPEG [7] running on a Windows 7 computer. FFMPEG contains a large number of supported video encoders, some of which we analyze in this paper, including FFV1, FFVhuff, JPEG-LS, LJPEG, x264, and JPEG2000.

3. CONVENTIONAL METHODS

Due to the focus on consumer product demands, few video encoders support high bit-depth in video compression, and thus leave limited options. Using the FFMPEG tool, the only encoders that support high bit-depth and lossless video compression without modification are FFV1, FFVhuff, JPEG-LS, LJPEG, and JPEG2000. FFV1, FFVhuff, and LJPEG only provide lossless compression, but JPEG-LS and JPEG2000 also support lossy compression in addition to lossless.

Figure 2 shows the CR results for our video test set using four of the five conventional lossless encoders for high bit-depth video. LJPEG is not shown due to complications in the FFMPEG tool, which resulted in incorrectly encoded videos, and thus a non-lossless compressed file in our tests. The results show a common trend across all of the encoders with respect to CR. Cloud001 had the highest CR, while Cloud002 and NoCloud001 both had similar CRs less than Cloud001. These results are expected and correlate well with our initial assessment of the video content shown in Figure 1. Cloud001 has a more uniform scene, thus higher CR than the more complex scenes in Cloud002 and NoCloud001 is expected.

Out of these four conventional encoders, FFV1 resulted in the highest CR and FFVhuff resulted in the lowest CR. In the best-case scenario, FFV1 achieved a CR of 1.85 with Cloud001, a CR of 1.48 with Cloud002, and a CR of 1.52 with NoCloud001, with an average CR of 1.62. Since FFV1 is the best conventional encoder, we designate FFV1 as our baseline for evaluating our proposed preprocessing methods.

4. PREPROCESSING METHODS

Since, even in the best-case scenario, the highest achievable CR using any conventional encoder is 1.85, which is far from our general goal of 10, we propose several new preprocessing methods to improve CR. In this section, we describe each of our proposed preprocessing methods and evaluate the methods against our baseline CR (Section 3).

Filtering

Filtering is a method of reducing noise in a video, which in turn reduces the entropy of the video. Entropy is the average of the information contained in each individual frame of the video [12,13]. In theory, by reducing the entropy of the video using noise reduction filtering, CR is expected to increase [14].

We evaluated several filters, such as Gaussian, Median, Average, and Wiener, however, the entropy of the video only reduced by a small amount and resulted in negligible CR improvements. Of these filters, Gaussian resulted in the lowest RMSE at kernel size of 5x5 and standard deviation of 0.50 pixels. Applying Gaussian to Cloud001 and compressing the resulting video using FFV1 only showed a 1.01× improvement over the baseline (CR of 1.86) with an RMSE of 15. Even at an RMSE over 100, CR increased by only 1.04×. Given these results, we concluded that filtering does not provide appreciable CR improvement.

ROI

ROI is a method designed to compress the vital or critical region of the video using lossless compression, and compressing the remaining non-critical region using lossy compression. Potential savings using ROI depend on the size of the critical region, with smaller critical regions providing a larger potential increase in the video's overall CR.

Since ROI results are highly dependent on the specific application situation, our implementation assumed a moderately sized ROI by dividing the video into four quadrants. We arbitrarily selected the first (upper left) quadrant as the ROI for lossless compression. We used JPEGLS as the video encoder, since JPEG-LS has both lossless and lossy compression modes. The amount of loss allowed was set using a compression setting—a parameter called the *quality factor*—that ranges from 0 to 128. A quality factor of 0 provides lossless compression, and a quality factor between 1 and 128 varies the degree of loss. Higher quality factors (i.e., more loss) have a higher potential CR.

Figure 3 shows the overall CR averaged over the video test set for our ROI method. The overall CR for a single video was calculated by summing the compressed size of each quadrant and comparing this total size to the original raw file size using Equation 1. The RMSE plotted is that of the lossy portion of the video. At our target RMSE of 15, CR increased by 1.81× (CR of 2.94) as compared to the baseline. Since the critical region is encoded with lossless compression, the target RMSE can be relaxed (i.e., increased). For example, at an RMSE of 50, CR increased by 2.06× (to a CR of 3.34) as compared to the baseline.

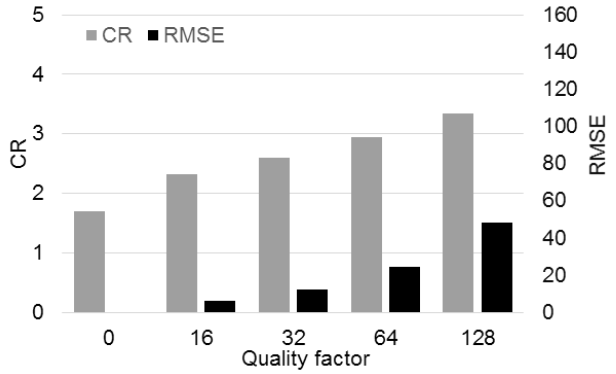


Figure 3: CR and RMSE for ROI, averaged over the video test set using JPEG-LS, for varying quality factors

Even though ROI showed a marked improvement in CR over filtering, in these tests, we manually designated the ROI as an arbitrary quadrant. In order to most effectively leverage ROI in a real world application, an automated method for selecting the ROI is required. This automation is application-dependent and is not within the scope of this paper.

Factoring

Factoring reduces the number of bits in the video before compression. Our factoring method divides the pixel value by a factor number 2^n, where n represents the number of bits being reduced or removed. For example, factor numbers of 2, 4, 8, 16, etc. will reduce the number of bits by 1, 2, 3, 4, etc. In our MATLAB implementation, the program also defaulted to rounding the output value to the closest value. Using our 14-bit video test set and a factoring number of 64 would result in only 8 bits of the raw data being used in the compression, and a factoring number of 32 would use only 9 bits. The original bit-width can be attained by multiplying the pixel values of the decompressed video file by the same factor used before compression. Obviously this method results in some loss of data in the least significant bits, but the tradeoff is potentially higher CR.

We evaluated factoring using FFV1, since this lossless video encoder does not introduce additional RMSE, allowing us to evaluate just the RMSE introduced due to factoring. We varied the factor number from 1 to 64, using 64 as the maximum factor number since this value produces an 8-bit video from the original 14-bit video, which is already a considerable loss of data.

Figure 4 shows the CR and RMSE results for the video test set for varying factor numbers. Factoring achieved a CR as high as 4.43 for Cloud001, 2.84 for Cloud002, and 2.94 for NoCloud001, with an average CR of 3.40, which is a 2.10× increase over the baseline CR. RMSE values for factor

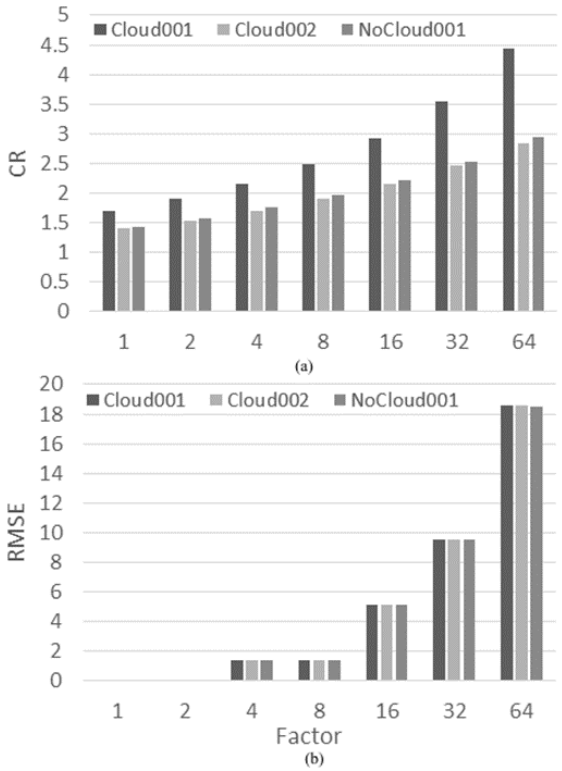


Figure 4: Factoring results with the video test set for varying factor numbers in terms of (a) CR and (b) RMSE

numbers of 2, 4, and 8 showed inconsistency, where the results did not match expected values. Factor numbers of 4 and 8 should have resulted in different RMSE values, and a factor number of 2 should not have resulted in 0 RMSE. This outcome is due to some error in the way that the original simulated video test set was created, whereas all the pixel values were either equal to 0 or 2 modulo 8. The results also show that the RMSE introduced by factoring was constant across all videos, indicating that RMSE lost due to factoring is not scene-dependent when observing most natural scenes. Factoring attains a CR comparable to that of ROI but at a much lower overall RMSE.

SuperFrame

JPEG2000 is a popular image-compression algorithm for use in various domains, especially space applications. For instance, the *Consultative Committee for Space Data Systems* (CCSDS) recommends using JPEG2000 for video compression for applications in which the data are stored locally for transmission at a later time [15]. However, since JPEG2000 is for image compression, using JPEG2000 for video requires compressing each frame in the video individually, thus JPEG2000 cannot take advantage of redundancy in the temporal, or time, domain like other video encoders can typically leverage.

SuperFrame is a method suggested by our sponsors (Alex Toussaint and Dr. Reed Weber at the AFRL Space Vehicles Directorate) to introduce more redundancy into the raw video data for improved JPEG2000 image compression. Figure 5 depicts the SuperFrame concept, where sequential

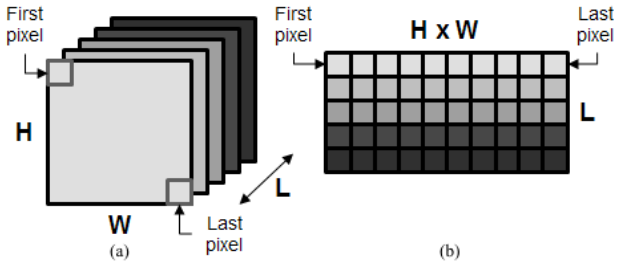


Figure 5: SuperFrame concept: Sequential video frames (a) are reorganized into a single SuperFrame (b) for frame-based compression

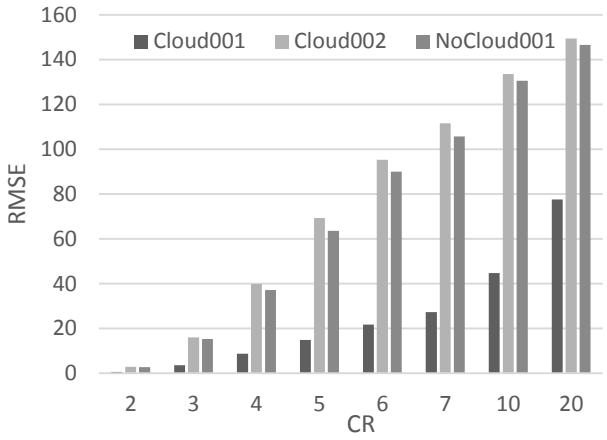


Figure 6: RMSE versus CR using the normal JPEG2000 image compression method

video frames (Figure 5(a)) are rearranged into a single combined frame (Figure 5(b)). The figure depicts this concatenation using different shades of gray to show the video frame layout in the SuperFrame. Each horizontal line in a SuperFrame contains all of the pixels for a single video frame. For example, Cloud001 has a frame size of 256×256 with 4,224 total frames. The combined SuperFrame would be a single frame of size 4224×65536. This SuperFrame can then be compressed using JPEG2000 as if the video is a normal (big) image. If an entire video produces a SuperFrame larger than JPEG2000 can process, multiple SuperFrames can be created, each no larger than the maximum size that JPEG2000 can compress.

JPEG2000 supports both lossless and lossy compression, where the lossiness is varied by specifying a desired CR. Figures 6 and 7 depict the RMSE for the video test set with varying CR values, using JPEG2000 in lossy compression mode. Figure 6 shows the normal method for using JPEG2000 for video compression, where all frames are individually compressed. Figure 7 shows the SuperFrame method. At a CR of 20, SuperFrame had an RMSE of less than 20 for all of the videos, with an RMSE as low as 4.8 for Cloud001. At the same CR, using the normal JPEG2000 image compression method, the RMSE was as high as 150, with a best case RMSE of 80 for Cloud001. Results showed

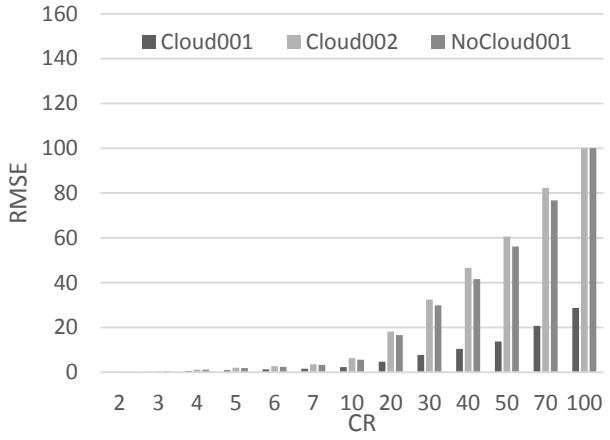


Figure 7: RMSE versus CR using a SuperFrame with JPEG2000

that SuperFrame could significantly increase CR for some videos, such as Cloud001, which achieved a CR above 50, a 30× increase over the baseline, with an RMSE value below our targeted maximum of 15.

Using JPEG2000 in the lossless compression mode, SuperFrame resulted in a CR of 3.00 for Cloud001, 2.44 for Cloud002, and 2.49 for NoCloud001, with an average 1.63× improvement as compared to the baseline. These results show that it is possible to increase the lossless CR by using SuperFrame without sacrificing any data, unlike filtering, ROI, or factoring. Thus, SuperFrame is a more ideal method for applications requiring absolutely no data loss.

Bit-Stream Splitting (BSS)

Due to the fact that most consumer video encoders use 8-bit format, advancements in video compression techniques focus on 8-bit video, with popular encoders such as H.264/H.265 and VP8/VP9[1]. Our novel method of BSS allows any 8-bit video encoder to be used with higher bit-depth videos.

Figure 8 represents the basic flow of compressing a high bit-depth video using BSS. Even though BSS can be used for any bit-depth, we describe the concept using our 14-bit grayscale video test set (Section 2), which in raw file format uses 16-bits for each pixel. The video is split into two parts, resulting in two video files, one with the upper bytes and one with the lower bytes. Each resulting 8-bit video file can be compressed using any 8-bit video encoder, which could not have been used on the original 14-bit video. To obtain the original video, the two compressed files are decompressed and merged back into a single video file.

Figure 9 shows the CR for various lossless encoders using BSS. The results show that x264 (an open-source

[1] Even though H.265 and VP9 reported support high bit-depth formats, at the time of testing, these implementations in FFMPEG were not functional.

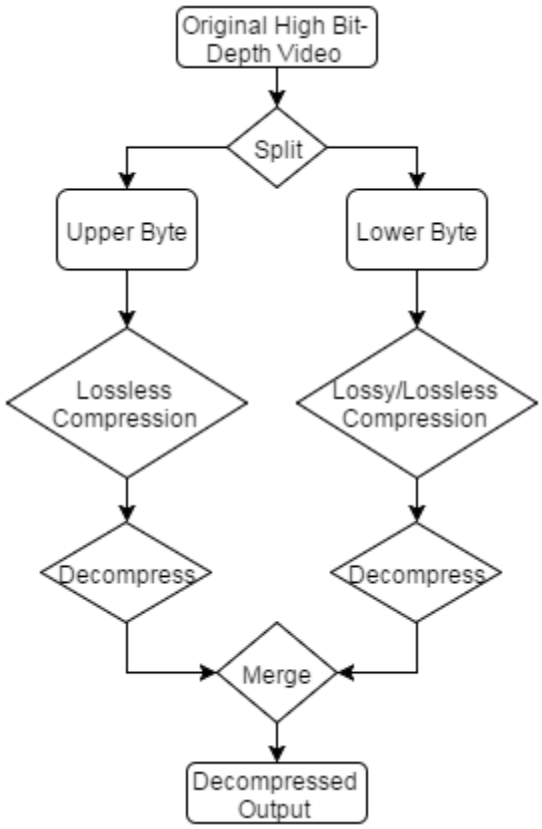


Figure 8: BSS conceptual flow

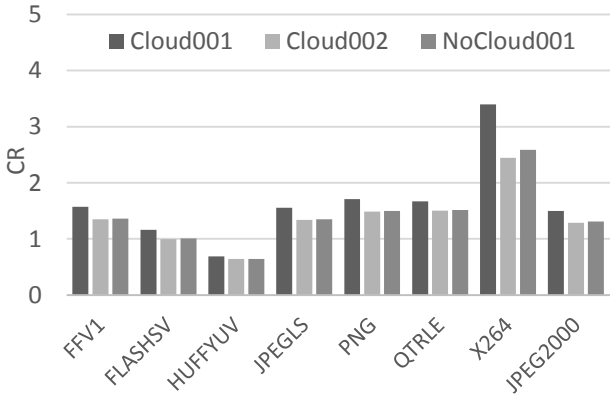


Figure 9: CR from lossless compression with BSS

implementation of H.264) provided the best CR, achieving a CR as high as 3.40 for Cloud001, 2.45 for Cloud002, and 2.60 for NoCloud001. With an average CR of 2.82, BSS increased the CR by 1.74× as compared to the baseline. BSS also outperformed SuperFrame in lossless compression.

Splitting the bits also enables *targeted* lossy compression. Since the upper byte contains more relative information about the video data as compared to the lower byte, lossless compression can be applied to the upper byte and lossy compression to the lower byte.

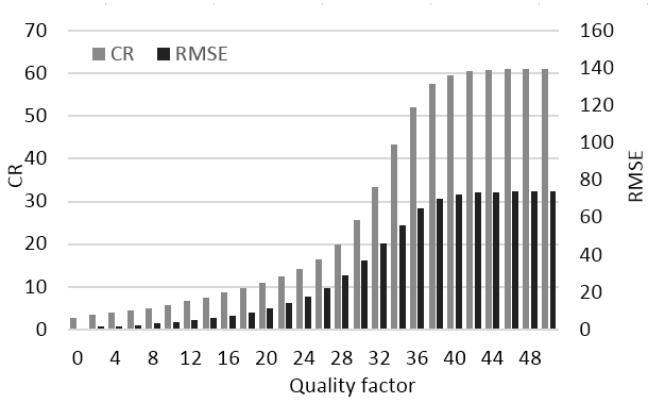


Figure 10: Lossy CR and RMSE using x264 with BSS. Results are averaged over the video test set

Figure 10 shows the CR and RMSE using x264 with BSS, averaged over the video test set. Even though there are other 8-bit encoders that support lossy compression, such as VP8/VP9, the CCSDS recommends using MPEG-4 part 10, an H.264 implementation, for certain space video applications that use 8-bit video format [15]. The results in the figure show that x264 with BSS achieved an average CR of 25 at our target RMSE of 15, a 15.4× improvement as compared to the baseline.

5. Combination of Methods

Given the individual evaluation of our five preprocessing methods, the results showed that factoring, SuperFrame, and BSS achieved the best CR improvement over the baseline using both lossless and lossy compression. Since these methods do not have to be orthogonal, we evaluated several combinations of these methods to achieve even higher CR.

BSS and Factoring

Results showed that, out of all our preprocessing methods, BSS using x264 achieved the best lossless CR, and factoring provided good lossy CR with acceptable RMSE (Section 4). Thus, we evaluated the combination of BSS and factoring by applying factoring to the raw video test set before applying BSS with x264 lossless compression.

Figure 11 shows the CR and RMSE results for the video test set using this combination of BSS and factoring, which shows similar constant RMSE trends as did the factoring results in Figure 4(b) across different videos. This similarity is expected, since we are using lossless compression with BSS, which does not introduce additional data loss beyond the data loss incurred by factoring. However, this combination resulted in greater CR, as depicted in Figure 11(a). In the best-case scenario, Cloud001 has a CR of 36.3 with an RMSE of 18.5, an 8.19× improvement in CR as compared to factoring alone. On average, using this combination increased CR by 22.4×, with an RMSE of only 18.5, which is a CR more than double our targeted CR of 10 while keeping the RMSE close to 15.

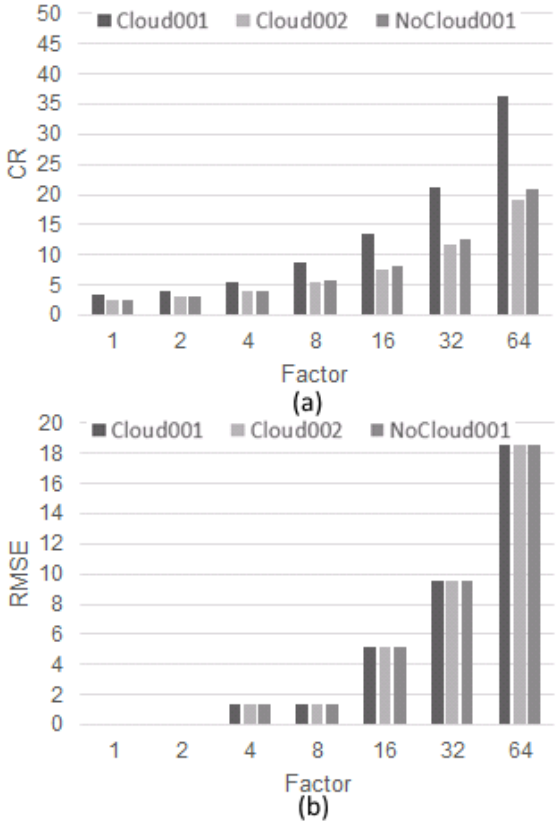


Figure 11: (a) CR and (b) RMSE results when combining factoring and lossless BSS using x264 on the video test set with varying factor values

BSS and SuperFrame

Since SuperFrame showed good improvement in both lossless and lossy CR, we combined BSS and SuperFrame to achieve even higher CRs. Figure 12 compares the CR and RMSE results from the video test set using BSS separately, SuperFrame separately, the combination of BSS and factoring, and the combination of BSS and SuperFrame.

The results show that the combination of BSS and SuperFrame has a higher RMSE at CR less than 20 compared to just using BSS and SuperFrame separately, and the combination of BSS and factoring. However, at higher CR values, the BSS and SuperFrame combination had lower RMSE than BSS alone. In the case of Cloud001 (Figure 12(a)), the BSS and SuperFrame combination showed higher RMSE than using SuperFrame alone. Figures 12(b) and 12(c), however, show that for Cloud002 and NoCloud001 the combination of BSS and SuperFrame had the lowest RMSE across these combinations. These results indicate that CR, and thus best method in combination, is dependent on the particular video scenes.

Since our targeted RMSE is 15 or lower, Figure 13 depicts a closer look at the achievable CR while maintaining an RMSE in this range. This figure presents the same results as in Figure 12, but with the RMSE axis limited to 20 to more

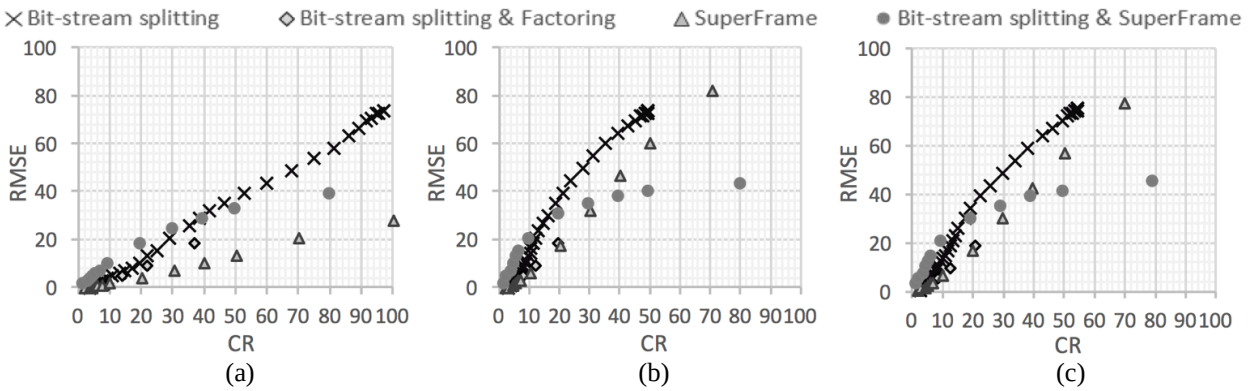


Figure 12: Comparison between BSS and SuperFrame applied separately versus combined BSS with factoring and BSS with SuperFrame on the video test set: (a) Cloud001, (b) Cloud002, (c) NoCloud001

clearly evaluate trends. As observed, the combination of BSS and SuperFrame does not perform as well as just using either BSS or SuperFrame separately. Figure 13 also shows that SuperFrame achieved the highest CR while remaining below our target value of RMSE for the entire video test set, even though trends in Figure 12 indicated that BSS and SuperFrame in combination performed better than SuperFrame separately for CRs higher than 20.

6. CONCLUSIONS

The demand for high bit-depth video is ever increasing for critical and sensitive applications, which places increased demand on system resources. Since most conventional video encoders do not support these formats, this paper presents novel preprocessing methods to address the need for higher compression ratios (CRs). Our results and analysis reveal that the largest CR improvements for high bit-depth video can be achieved using several of our proposed preprocessing methods: SuperFrame, bit-stream splitting (BSS), and factoring, or a combination of these. Since large CR increases the tradeoff in loss of data quality, our analysis enables system designers to select the most appropriate preprocessing method, or combination of methods, based on the specific application requirements. Our preprocessing methods enable new and more complex algorithms in resource-constrained environments. For example, since BSS transforms the data to a format suitable for any existing 8-bit encoder, this method enables a designer to consider a large range of highly developed consumer-oriented encoders instead of relying on a more specialized, application-specific encoder.

Future work includes measuring the resource cost needed to run our proposed preprocessing methods, and the additional processing overhead that these methods demand upon the compression speed. These overheads are another critical issue that must be considered by system designers when selecting preprocessing method(s), since compression for a given mission and high bit-depth video leads to requirements in compression rate and quality as well as compression speed.

ACKNOWLEDGEMENTS

This work was supported by the I/UCRC Program of the National Science Foundation, under Grant Nos. EEC-0642422 and IIP-1161022, and the industry and government members of CHREC. The authors gratefully acknowledge the AFRL Space Vehicles Directorate for the evaluated video test set used in this research, as well as their strong support and interaction on this project. We also gratefully acknowledge Mr. Prashant Awasthi, former student of CHREC, for his contributions with filtering and factoring.

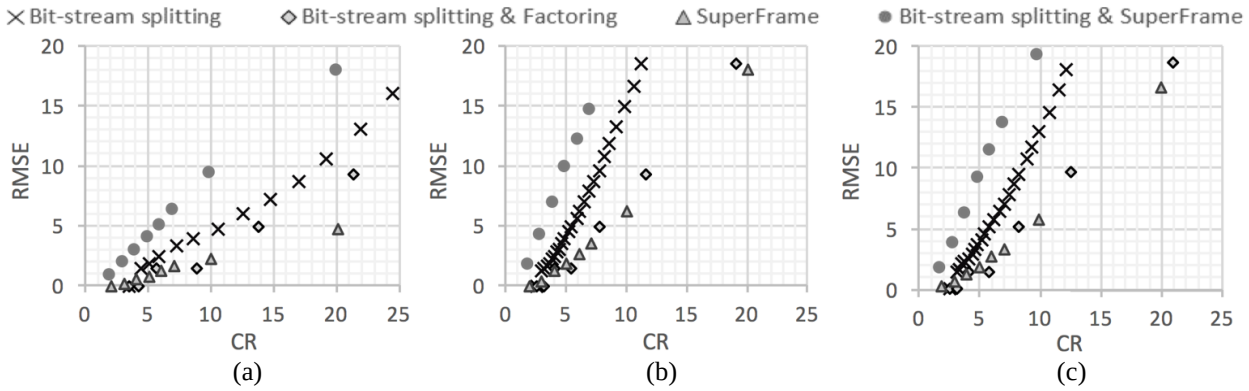


Figure 13: A more detailed comparison (zoomed in view) of results in lower left of Figure 12 on the video test set: (a) Cloud001, (b) Cloud002, and (c) NoCloud001

REFERENCES

[1] *ITU-T H.264*, February 2014.

[2] *ITU-T Recommendation T.81*, The International Telegraph and Telephone Consultative Committee, September, 1992.

[3] Feller, C.; Wuenschmann, J.; Roll, T.; Rothermel, A.; Inst. of Microelectron., Ulm Univ., Ulm, Germany, "The VP8 video codec – overview and comparison to H.264/AVC", Consumer Electronics – Berlin (ICCE-Berlin), 2011 IEEE International Conference, Berlin, 6-8 Sept. 2011, 57-61.

[4] Information Technology-Lossless and near-lossless compression of continuous-tone images-Baseline. International Telecommunication Union (ITU-T Recommendation T.87). ISO/IEC 14495-1, 1998.

[5] FFV1 Video Codec Specification [Online]. Available: http://ffmpeg.org/~michael/ffv1.html

[6] ISO/IEC 15 441-1: *Information Technology-JPEG 2000 Image Coding System-Part 1: Core Coding System*, 2000.

[7] FFMPEG [Online]. Available: http://www.ffmpeg.org

[8] S. Yu, Q. Qiao, L. Luo, and Y. Yang, "Increasing compression ratio of low complexity compressive sensing video encoder with application-aware configurable mechanism," *2014 International Conference on Communication and Signal Processing*, 2014.

[9] Y. Bao, B. Stukken, J. Stals, C. Chen and L. Claesen, "Quantitative comparison of lossless video compression for multi-camera stereo and view interpolation applications," *2015 IEEE 13th International New Circuits and Systems Conference (NEWCAS)*, 2015.

[10] W. Pennebaker and J. Mitchell, *JPEG still image data compression standard.* New York: Van Nostrand Reinhold, 1993.

[11] T. Ito, Y. Bandoh, T. Seishi and H. Jozawa, "A coding method for high bit-depth images based on optimized bit-depth transform," *2010 IEEE International Conference on Image Processing*, 2010.

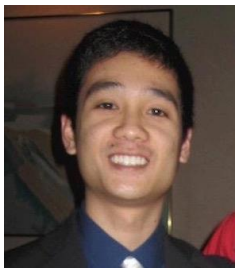
[12] H. Bai, A. Wang and A. Abraham, "Entropy analysis on multiple description video coding based on pre- and post-processing," *2012 12th International Conference on Hybrid Intelligent Systems (HIS)*, 2012.

[13] P. Symes and P. Symes, *Digital video compression.* New York: McGraw-Hill, 2004.

[14] A. Langi and W. Kinsner, "Compression of aerial ortho images based on image denoising," *Proceedings of Data Compression Conference – DCC '96,* 1996.

[15] CCSDS 706.1-G-1, Motion Imagery and Applications, CCSDS, 2010.

BIOGRAPHY



An Ho *is an M.S. student in ECE at the University of Florida. He received his B.S. degree in EE from the University of Florida. He is a research assistant in the on-board processing group in the NSF CHREC Center at Florida.*

Alan D. George *is Professor of ECE at the University of Florida, where he serves as Director of the NSF Center for High-performance Reconfigurable Computing known as CHREC. He received the B.S. degree in CS and M.S. in ECE from the University of Central Florida, and the Ph.D. in CS from the Florida State University. Dr. George's research interests focus upon high-performance architectures, networks, systems, services, and applications for reconfigurable, parallel, distributed, and fault-tolerant computing. He is a Fellow of the IEEE.*

Ann Gordon-Ross *received her B.S. and Ph.D. degrees in Computer Science and Engineering from the University of California, Riverside (USA) in 2000 and 2007, respectively. She is currently an Associate Professor of ECE at the University of Florida and is a member of the NSF Center for High-Performance Reconfigurable Computing (CHREC). Her research interests include embedded systems, computer architecture, low-power design, reconfigurable computing, dynamic optimizations, hardware design, real-time systems, and multi-core platforms.*