Providing Efficient Support for Lossless Video Transmission and Playback

Ali Şaman Tosun, Amit Agarwal, Wu-chi Feng
The Ohio State University
Department of Computer and Information Science
Columbus, OH 43210
{tosun,agarwal,wuchi}@cis.ohio-state.edu

Abstract

As networking technologies evolve, the ability to support the low-latency transmission of lossless video for applications such as scientific visualization or medical imaging will continue to become more important. With all the research focused on lossless and near-lossless image compression, little gains in compression performance have been achieved over the last decade. In this paper, we introduce a system that provides comparable compression ratios to current lossless compression techniques but makes it more amenable to network transmission and playback. Working with computer graphics researchers, this system provides the appropriate tradeoff of constraints and performance for their needs.

1 Introduction

As networking and computing technologies continue to advance, the ability to support advanced video-based application that deliver high-quality video are now possible. With the large amount of research focused on the efficient coding, storage, retrieval, and transmission of digital video, we are currently able to support DCT-based video streaming across networks fairly well. We are, however, still unable to support applications that require more stringent bounds on the quality of the video such as medical imaging, advanced scientific visualization, and computer graphics.

Unlike *lossy* DCT-based video compression algorithms that transform the data into the frequency domain so that it is more compressible, lossless image compression techniques attempt to predict the value of each pixel based on some of the surrounding pixels and then entropy encoding the difference to the predicted value. More importantly, it has been shown that there is little advantage to temporal compression of these sequences [7]. Because of this, the compression ratios achieved by lossless image compression standards are much less than that of DCT-based algorithms (typically 2:1 for lossless and 25:1 for DCT-based). Thus, the coding and transmission of lossless video can be extremely resource hungry.

The efficient support for lossless video-based applications is extremely difficult because they require low-latency delivery of video while requiring much larger bandwidth, generally considered two mutually exclusive goals. In our discussion with these researchers and scientists, two main themes manifested themselves. First, they need the ability to have low-latency visualizations so that they can understand what is happening in the simulation data at a higher-layer (that is, general understanding). Second, they need the ability to query the data to be able to extract meaningful information from the data accurately. Note that the image data is not necessarily confined to 24-bits per pixel.

In this paper, we introduce a lossless video compression technique that provides comparable performance to current lossless image compression techniques *but*, more importantly, makes the data more amenable to network transmission. The main purpose of this approach is to provide (i) a smaller-sized MPEG video sequence that supports the low-latency requirements, allowing the researchers to understand the data at a higher-level and (ii) a lossless or loss-bounded differential file that allows the original image data to be reconstructed, if necessary [1]. Our results will show that we can achieve lossless and loss-bounded image compression that has comparable performance to lossless video compression algorithms but also supports low-latency delivery (through the smaller MPEG file). The experiments use two video sequences, one from medical imaging and the other from a scientific visualization program.

The rest of the paper is organized as follows. In section 2, we describe some of the background and related work necessary for the understanding of this paper. In section 3, we describe our proposed approach. In section 4, we provide experimental results for both lossless and loss-bounded compression using the previously mentioned image data. In

[1]Clearly this is achievable only if the MPEG decompression algorithms are the same. We will elaborate more on this later in the paper

section 5, we conclude and give future research directions.

Contributions of this work: The main contribution of this work is the design and analysis of a lossless video system that provides scientists with high-level, low-latency understanding of scientific computations while preserving the important lossless characteristics that they desire. In particular, our work compares and contrasts several lossless video compression algorithms (including LOCO which serves as the basis for JPEG2000) using lossless video data that is used by scientists and graphics researchers at Ohio State. Our goal *is not* to create a better lossless video compression algorithm, but making lossless video more available and usable for scientists.

2 Background and Related Work

We break the background and related work section into two main parts: lossless and loss-bounded image compression techniques, and video compression algorithms.

2.1 Lossless/Loss-bounded Image Compression

There has been considerable research in the area of lossless image compression in the last three decades. While many standard compression techniques such as Huffman, Arithmetic, Liv-Zempel(LZ), and Liv-Zempel-Welch (LZW) can be employed, lossless image compression techniques attempt to take advantage of the spatial properties of the image to aid in compression. A number of image-specific lossless compression algorithms and systems have been introduced recently. These algorithms include the JPEG lossless coder, SUNSET, universal context modeling, FELICS [4], CALIC [10], LOCO-I, SICLIC and many others. Despite all the effort, there has not been much of an improvement in the compression ratio which typically varies from 1.5:1 to 3:1, depending on the image. Even with such low compression ratios, the gains can have substantial impact, given the size of the original image.

Lossless image compression algorithms typically comprise of two distinct and independent components: modeling and coding. The modeling part can be formulated as an inductive inference problem in which an image is observed pixel by pixel in some defined order (usually in raster-scan). The goal then is to infer the next sample value from a selected region of pixels that are close to it.

For this paper, we focus on the use of the LOCO-I image compression algorithm since it has been standardized as the algorithm for JPEG 2000 [6]. LOCO-I provides for both lossless and near-lossless compression of continuous-tone images. Since LOCO-I has been designed for still-image data, it does not take into account spectral redundancy in the case of multi-spectral images and temporal redundancy in the case of video sequences. It is based on a simple fixed context model - combining simplicity with the compression potential of context models. Based on the context for the current pixel the current pixel value is predicted using a primitive edge detector (because of complexity constraints). The context for conditioning the current prediction residual is built out of the differences between the pixel values in the context. These differences represent the local gradient thus capturing the local activity. The model is tuned for efficient performance in conjunction with an extended family of Golomb-type codes, which are adaptively chosen and and embedded alphabet extension for coding of low-entropy image regions. Interband CALIC [9] is one step to take into account the above two resulting in modest gains in compression. Another approach is SICLIC [1] which is an inter-color coding algorithm similar to the LOCO-I algorithm. It does both inter and intra color encoding taking into account the spectral redundancy component in the image.

2.2 Video Compression Algorithms

For brevity, we assume that the readers are somewhat familiar with the details of the MPEG video compression standard [3] and simply highlight the relevant parts for this paper. First, the data is lossy transformed from the RGB color space to the YUV color space with the U and V channels being subsampled at a 2:1 ratio in both spatial dimensions. Second, MPEG uses a discrete cosine transform (DCT) to separate the data into its frequency components, making it easier to run-length compress. After the DCT is performed, the resultant coefficients are quantized. Larger quantization values mean that more of the coefficients will have the same value, increasing the compression ratio. Smaller quantization values mean that less of the coefficients will have the same value, decreasing the compression ratio. As a result, the DCT and quantization steps will also introduce error in the compression of video data. Finally, we note that MPEG has varying frame types to take advantage of inter-frame redundancy by using predictive coding coupled with motion compensation.

The MPEG-2 video standard is very similar in flavor to the original MPEG-1 video compression standard but provides for higher quality video such as HDTV video. The actual differences between these algorithms is limited to fairly minor items such as providing for interlaced video with fields and different zig-zag ordering of coefficients when interlacing is used.

3 Proposed Method

To aid scientists in the visualization and understanding of scientific data, we propose a multi-layer lossless video compression algorithm that supports low-latency viewing of time-varying data and supports lossless query retrieval

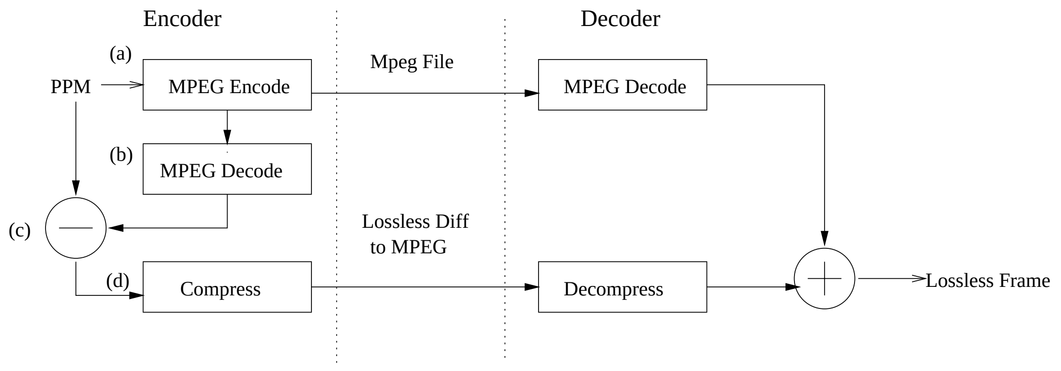


Figure 1. Compression and Decompression using proposed method

of the original image data. The basis of our approach is to use MPEG to compress the video stream to get an *in the ball park* video that provides the scientist with a high-level, real-time view of the visualization. After the scientist understands the high-level view of the visualization and wants to look at actual values of pixels (which may represent physical phenomena such as pressure) we augment the data with a lossless differential that provides the original data back to the user (i.e. is lossless). For clarity, we refer to the lossless differential as an *enhancement* and use *differential* to mean that a frame is differentially coded with respect to a previous frame.

3.1 Reasons For/Against Using MPEG

In the general scenario where millions of people may want access to the video, this type of multi-layering may not be possible. The main reason behind this is that different MPEG implementations may result in different image data, mostly due to shortcuts taken to optimize for speed. Some of these differences include:

- Different inverse-DCT algorithms being implemented in the decoder. Depending on the conformance level, the decoder may implement a two-pass one-dimensional DCT (column/row), or may implement a one-pass two-dimensional DCT [8, 2].
- Tweaks to optimize performance at the cost of accuracy. In order to maximize the decoding rate, a decoder may not perform all the calculations. One such example is using the Berkeley MPEG-1 decoder [5] with half-pixel motion estimation searches. If the motion vectors in the B-frames are both half pixel in size, the Berkeley decoder will not completely average all four pixels that make up the motion compensated pixel. Their results show that there is very little visual difference.

However, all is not lost. The researchers that we are working with are perfectly willing to use the *same* decoder on all machines. The only caveat being that the machines may not necessarily be the same. We have found that under these assumptions, our approach results in lossless delivery of the video. We will discuss this point further in the experimentation section.

3.2 Lossless Video Overview

The encoding phase of our proposed approach consists of four main steps: (a) MPEG video compression, (b) MPEG video reconstruction, (c) differential image calculation, and (d) differential entropy encoding. In the first step, we encode the video frames using quantization levels of 1 for the I, P, and B frames. Using a low quantization level, ensures that the artifacts introduced during the DCT transformation remain small. This is particularly important when dealing with computer generated data that contain many high frequency components. In the second step, we decode the video frame to determine the resultant video frame. This decoded frame is then used to create the enhancement image from the original frame in the third step. In the final step, we lossless compress the enhancement video frame. As we will show in the experimental section, *Huffman compression or LZ and not LOCO-I* will be the most effective in compressing the enhancement because it provides similar levels of compression and provides much faster decoding speed. A block-level diagram of the compression stages is shown in Figure 1.

Viewing the compressed data occurs in several phases as well. Initially, when the scientist is trying to understand the simulation, the highly compressed MPEG-file is streamed to the user. If the user find an interesting phenomena that he/she wants to look at in greater detail, the lossless compressed enhancement is transmitted to the user. The enhancement is then decoded and added back to the MPEG video frame to create the original video data without loss. In many cases, it is expected that the user will not need to see the lossless compressed video frame, resulting in a large savings of network bandwidth.

4 Experimentation

In this section, we compare and contrast the performance of our algorithm with other algorithms such as LOCO-I for lossless video compression and transmission. We begin with a description of the experimental setup including the data used, the different architectures used, and the different decoders that were used. We then describe the comparison algorithms that were implemented. Finally, we present our experimental data.

4.1 Experimental Setup

In this subsection, we briefly highlight the software and setup that we used for the experiments.

4.1.1 Input data

For our experiments, we used two different data sets. The first data set is a scientific visualization simulation that shows the movement of clouds and winds around the earth. The original data set consists of frames that are 1920x1035 pixels in size. Due to the limitations of the MPEG-1 video coder, we extracted a 352x240 pixel subregion within the data. In addition, we extracted 16 frames from the data set to use in our experiments. We refer to this data set as the *wind* data set. The second data set is a visualization of a rotating head. We had 40 images that show the 360 degree camera rotation around the head. We refer to this data set as the *head* data set. Sample images from these data sets are shown in Figure 2.

4.1.2 Architectures used

For our experiments, we used three different architectures. These included an SGI Octane, a Linux-based PC, and a Sun UltraSparc running Solaris. The purpose of the various machines is to test the lossless properties across different architectures and to verify that they result in the same image.

4.1.3 MPEG encoders and decoders used

We used two publicly available MPEG encoders and decoders for our experiments. The first set consists of the Berkeley MPEG tools *mpeg_encode* (version 1.5) and *mpeg_play* (version 2.3)[2]. The second set of software consists of the MPEG Software Simulation Group tools *mpeg2encode* and *mpeg2decode*[3].

[2]These are available via the web at www.bmrc.berkeley.edu

[3]These are available via the web at www.mpeg.org

4.1.4 Lossless compression algorithms used

For our experiments, we coded several basic compression algorithms and were able to gather some publicly available image coding algorithms. The code for the comparison includes:

LOCO-I: a publicly available LOCO-I encoder and decoder from the Univ. of British Columbia

MPEG + LOCO-I enhancement: implements the MPEG video plus enhancement frames with respect to the MPEG stream using LOCO-I compression.

MPEG + Huffman enhancement: implements the MPEG video plus enhancements that are lossless compressed using Huffman compression.

MPEG + LZ enhancement: implements the MPEG video plus enhancements that are lossless compressed using *compress* which is an adaptive Lempel-Ziv compression.

LOCO-I Differential: uses modified LOCO-I encoder to do differential coding of video frames (from a previous frame). The purpose of this algorithm is to show the effect of temporal encoding.

4.2 Experiments

For our research, we are interested in three main questions:

- What overall compression ratio is achievable with the various algorithms?
- How do they compare in terms of compression and decompression time?
- What happens when different architectures and decoders are used?

In the rest of this section, we will describe the experiments that we ran and the resulting data that was obtained.

4.2.1 Compression Performance

To test the compression performance of the algorithms, we ran the sample data sets through the various algorithms and plotted the resulting compression ratios (on a frame-by-frame basis). The results for lossless compression, loss-bounded compression with bound 2 and loss-bounded compression with bound 4 are given in Figure 3, Figure 4, and Figure 5, respectively. As shown by the figures, the compression ratio achieved by the various algorithms is very similar. The *wind* figures show that there is some advantage to doing differential compression using LOCO-I. However,

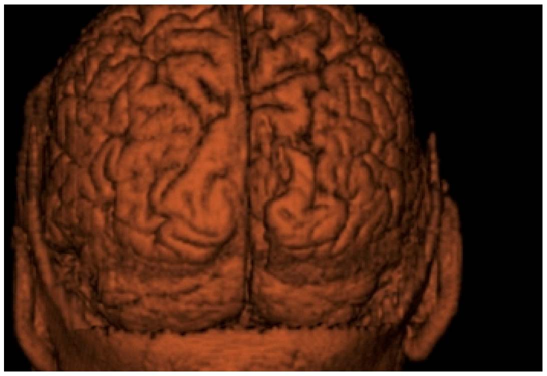
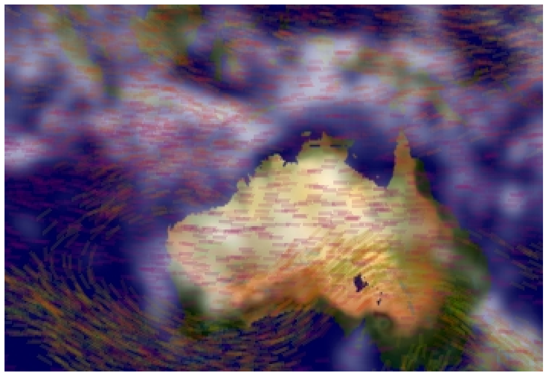
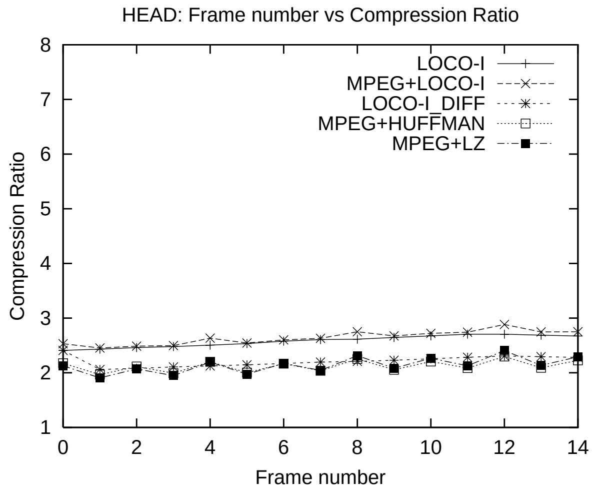



Figure 2. Sample head and wind images



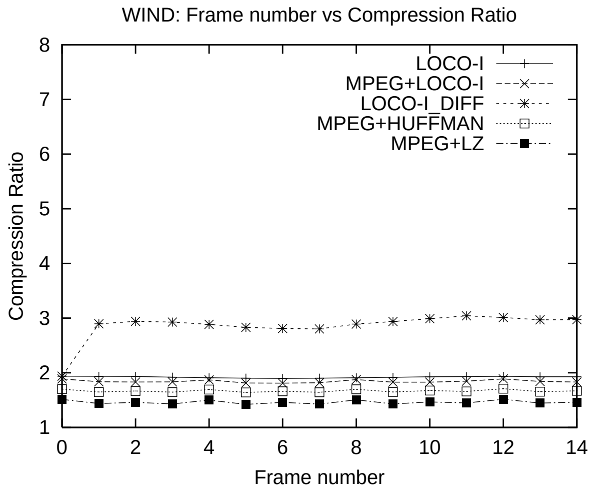


Figure 3. Compression ratios of frames for head and wind data for lossless compression

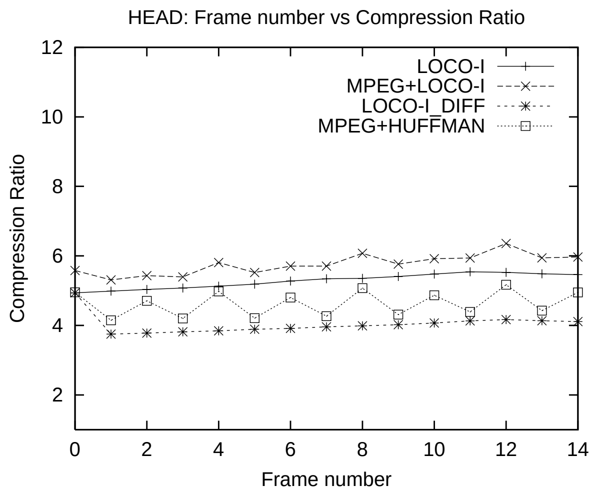
this advantage becomes a disadvantage when applied to the *head* data set. The reason for this is that the *wind* data set has a background that does not change and the actual wind vectors slowly change. As a result, the data between frames is very similar. The main disadvantage is that for sequences that have larger motion (as in *head*), performing differential lossless compression is terrible. The main reason is that the differential is "fixing" data from the previous frame, resulting in a higher entropy. This experiment verifies the previous finds why the application of differential coding for lossless video is difficult [7].

In these figures, we also see that using MPEG and Huffman compression on the enhancements (between the MPEG and actual frame data) results in compression performance on par with using LOCO-I only or using MPEG and LOCO-I compressed enhancements. Finally, we note that with looser error bounds on the compression that the ability to compress the data becomes easier.

In Table 1, we have graphed the overall compression ratio for the various algorithms and loss-bounds. Compression ratio is computed by dividing the size of the original images by the size of compressed images. For algorithms which also have an MPEG part the size of compressed images is sum of size of MPEG and size of compressed differences. We implement lossbounded compression using huffman by mapping many nodes to one node without violating lossbound and then we compress the differences using this smaller tree. We don't have lossbounded compression results for LZ because LZ is lossless compression and we don't have a practical lossbounded LZ compression.

4.2.2 Algorithm Speed

So far, we have shown that the proposed approach results in similar compression to applying LOCO-I to each frame individually. In this section, we compare the performance of our approach to LOCO-I. In Table 2, we show the time required to compress and decompress the two data sets. Here, we see that MPEG compression is the most expensive op-



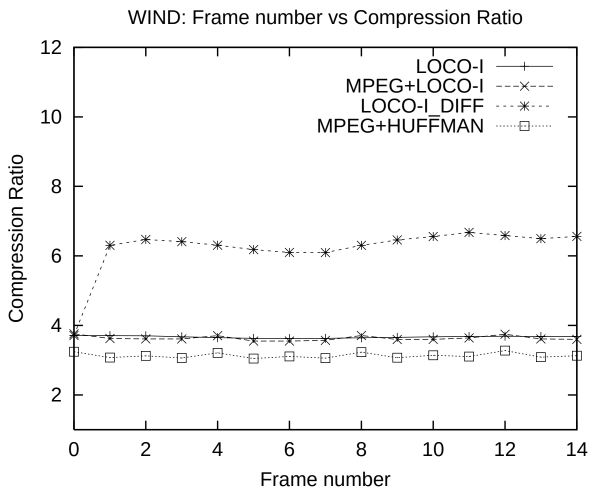
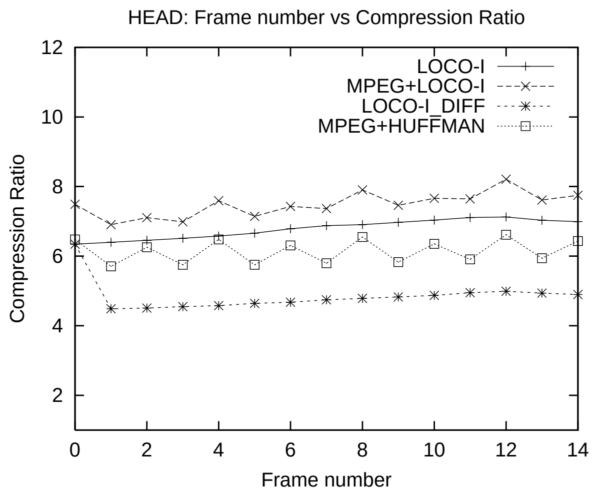


Figure 4. Compression ratios of frames for head and wind data for loss-bounded compression with bound 2



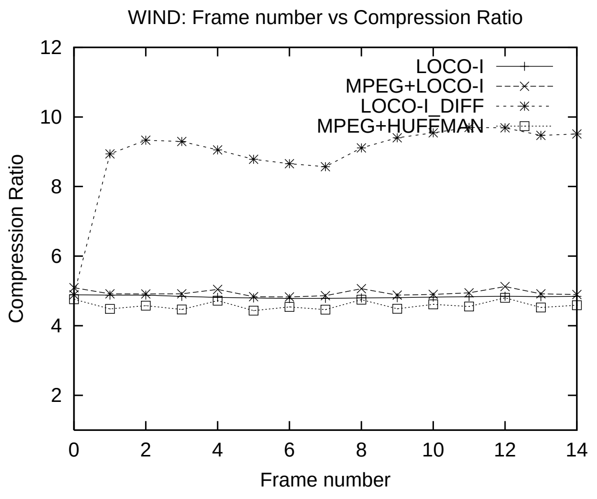


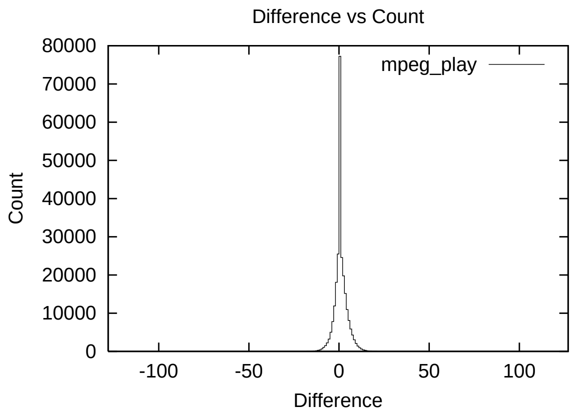
Figure 5. Compression ratios of frames for head and wind data for loss-bounded compression with bound 4

Table 1. Compression Ratio

Data	Compression	MPEG+ LOCO-I	MPEG+ HUFFMAN	MPEG+ LZ	LOCO-I	LOCO-I DIFF
Head	lossless	2.03	1.74	1.72	2.57	2.20
	bound-2	3.50	3.04	N/A	5.27	4.01
	bound-4	4.07	3.63	N/A	6.77	4.81
Wind	lossless	1.53	1.40	1.25	1.91	2.82
	bound-2	2.59	2.32	N/A	3.66	6.09
	bound-4	3.19	3.04	N/A	4.82	8.69

Table 2. Encoding/Decoding Performance

Algorithm	Encoding Time (Sec) Head	Encoding Time (Sec) Wind	Decoding Time (Sec) Head	Decoding Time (Sec) Wind
intra-frame LOCO-I	2.89	3.63	2.71	2.89
MPEG differences	3.03	3.74	2.84	3.03
MPEG	14.30	19.75	0.42	0.43
Huffman	1.92	2.20	0.76	0.78
LZ	0.79	0.98	0.42	0.51



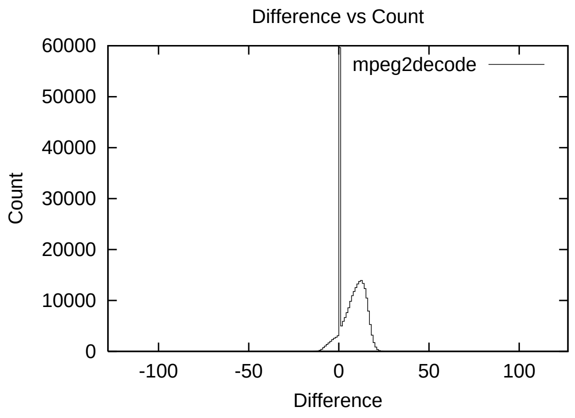


Figure 6. Differences when decoded using mpeg_play and mpeg2decode

eration, due mostly to the expensive motion compensation step. We also see, however that the *Compress* program on Unix has the best speed (probably due to heavy optimization), one-fourth to one-sixth of LOCO-I but typically has lower compression ratio. MPEG and Huffman compression result in a total decompression time that is one-half to one-third that of LOCO-I. As a result, we will be able to deliver the same quality video with less overhead at the client. The reason LOCO-I is so slow is that performs a number of operations per pixel in both the encoding and decoding phases. These steps in encoding include: predicting the value based on neighboring pixels, calculating a prediction based on gradient information between neighboring pixels, and adaptively coding the value using Golomb codes. The decoding must perform these steps in the reverse order.

4.2.3 Effect of Architecture and Different MPEG Players

In order to study the effect of different architectures, we first compared the displayed images of mpeg_play on Linux, Solaris, and SGI operating systems and observed that they are the same, as expected. This was accomplished by writing out all frames in raw R, G, B and comparing them pixel by pixel. While the three architectures returned the same results, we are investigating other operating system and architecture configurations. As a result, we believe that our algorithm can be used in a heterogeneous environment with multiple operating systems and architectures.

With regard to different MPEG players, we observed that for the same MPEG file, the displayed images of mpeg_play and *mpeg2decode* are not the same since they implement the DCT transformation using different algorithms. Because our enhancement images that create the lossless video depend on the decoded images, it is important that same decoder is used in compression and decompression of lossless video. The differences when the image is decoded using mpeg_play without the enhancements and mpeg2decode without enhancements are given in figure 6. As shown in the figure, the MPEG-2 player when configured to use all operations, results in a slightly different image. We are still trying to determine the exact cause of these differences. From our observations, the MPEG-2 player resulted in a slightly duller image.

5 Acknowledgements

The authors wish to thank the reviewers for their comments on the initial draft. This work was supported in part by grants from the National Science Foundation, the Department of Energy, and Ameritech.

6 Conclusion

In this paper we proposed an MPEG based lossless video compression algorithm with support for bandwidth-efficient transmission. The goal of this work was to transform the lossless video data into a format that makes it more amenable to network transmission while achieving similar compression ratios to the lossless image standards. By using MPEG video, we can ensure that low-latency playback is possible in order to understand the simulation at a high-level. By using a Huffman compressed or LZ compressed enhancement frame that allows the original image to be losslessly reconstructed, we can provide better decompression speeds, allowing for a more interactive viewing experience for the scientists. Future work will continue to examine the effect of different libraries, operating systems, and architectures on the lossless properties.

References

[1] R. Barequet and M. Feder. Siclic: A simple inter-color lossless image coder. In *Data Compression Conference (DCC'99)*, pages 501–510, March 1999.

[2] A. L. C. Loeffler and G. Moschytz. Practical fast 1-d dct algorithms with 11 multiplications. In *Proc. Int'l conf. on Acoustics, Speech and Signal Processing (ICASSP'99)*, pages 988–991, 1989.

[3] L. G. D. Mpeg: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, April 1991.

[4] P. Howard and J. Vitter. Fast and efficient lossless image compression. In *Proceedings of 1993 Data Compression Conference*, pages 351–360, March 1993.

[5] B. S. K. Patel and L. A. Rowe. Performance of a software mpeg video decoder. In *Proc. of the conference on Multimedia '93*, August 1993.

[6] G. S. Marcelo J. Weinberger and G. Sapiro. The loco-i lossless image compression algorithm: Principles and standardization into jpeg-ls. *To apppear in IEEE Trans. on Image Processing*.

[7] N. D. Memon and K. Sayood. Lossless compression of video sequences. *IEEE Trans. on Communications*, 44(10), October 1996.

[8] Z. Wang. Fast algorithms for the discrete w-transform and for teh discrete fourier transform. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, AASP-32:803–816, 1984.

[9] W.-k. C. Xiaolin Wu and N. Memon. Lossless interframe image compression via context modeling. In *Data Compression Conference*, pages 378–387, 1998.

[10] X.Wu and N. Memon. Context-based, adaptive, lossless image coding. *IEEE Trans. on Communications*, 45(4):437–444, April 1997.