ABSTRACT

JOSHI, KANISHKA S. Design and Performance Evaluation of some Conditional Replenishment Schemes for Video. (Under the direction of Associate Professor Injong Rhee).

Conditional Replenishment is an inter-frame video compression method that uses correlation across images to reduce video transmission bitrates. This method works by encoding only those portions of the image that have changed in the subsequent image. All state of the art video encoding techniques use Conditional Replenishment, in Residual or Non-Residual form, for achieving high compression by exploiting the temporal redundancy between images. In this thesis we first examine a simple block based, Non-Residual, Conditional Replenishment scheme based on JPEG-2000 and JPEG-LS. The work shows that better (w.r.t the project goal of achieving a low complexity, low bitrate and high quality video encoding scheme) compression results are obtained using lossless compression mode, than using lossy compression mode for such a Non-Residual Conditional Replenishment process. It was also seen that Conditional Replenishment resulted in better quality video at the same bitrate than the one that was achieved without Conditional Replenishment. We see that JPEG-LS is more suited for lossless Conditional Replenishment than JPEG-2k because it shows similar compression performance, but works tens of times faster than JPEG-2k. The work also confirms the compression improvement for sparse histogram images using Histogram Packing and discusses the design, implementation and performance of several variants of this Simple Block Based Conditional Replenishment technique such as ones based on Longest Prefix String Matching, Longest Prefix String Matching using Gray Codes, Motion Estimation based Residual Conditional Replenishment. The work also confirms the finding that residual encoding based on Sum of Absolute Differences Motion

Estimation, that has proved to be so effective a technique in lossy video encoding, does not directly help in lossless video encoding. The results show that for natural image sequences, the codec based on Simple Block Based, Non-Residual, Conditional Replenishment followed by Histogram Packing followed by JPEG-LS gives bitrate and encoding time performance, comparable to the best lossless codecs reported, and for artificial images such as images obtained by desktop capture, Conditional Replenishment followed by ZLIB gives the most competitive results. The work has direct applicability in the implementation of low computation, lossless video encoding schemes such as those used for server-based computing, and video streaming.

Design and Performance Evaluation of some Conditional Replenishment Schemes for Video.

by

Kanishka S. Joshi

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science in

Computer Science

Raleigh, North Carolina

2007

Approved By:

Dr. Douglas Reeves	Dr. Matthias Stallmann

Dr. Injong Rhee
Chair of Advisory Committee

Biography

Kanishka Joshi was born in Mumbai, India on 29th Nov 1981, and lived all his life in Pune, India before coming to the United States. In June 2003, he completed his bachelors degree in Computer Engineering from the University of Pune and following his bachelors, he joined a startup named Codito Technologies where he was involved in implementing a parallelized H.264, MPEG2 and a Motion-JPEG encoder, on an on-chip multiprocessing system, so as to do real time SD encoding. After coming to NC State in August 2004, he worked for Dr. Matthias Stallmann on a parallelized branch and bound knapsack solver in Spring 2005. After spending one year at NC State, in May 2005, as part of the co-op program through NC State, he joined another startup based in Columbia, MD, named FastVDO LLC, and was responsible for parallelizing their H.264 encoder and decoder to get it to do real time 720p HD. Here he also got a chance to work on video streaming. Motivated to learn more about networking and communication issues, after he rejoined NC State in Jan 2006, he took coursework geared towards Networking, Error Control Coding and Cryptography. It was during this time that he got the opportunity to work for Dr. Injong Rhee on the implementation of a block based Conditional Replenishment scheme based on JPEG-2000, the extension of which is this thesis.

After the MS, he plans to join Cisco Systems as a Software Engineer. He also hopes to pursue a part time Masters in Operations Research. His area of interests are mainly Design and Analysis of Algorithms, Digital Signal Processing, Data Compression and recently Networking Protocols/evaluation and Communications. He hopes to gather competence in Statistics and Cryptography. His hobbies mainly motivated by philosophical curiosities, include Logic, Physics, Economics, Sociology, Medicine and Psychology.

Acknowledgements

Acknowledgements primararily go to Dr. Injong Rhee, who provided not only the idea that would become a thesis, but also provided with a solid guidance, financial support, intellectual challenges and much needed intellectual freedom that provided a wonderful environment for continuous learning, being productive and staying focussed. It was clearly here that I learned the foremost importance of hard work in doing experimental research. Further acknowledgements go to Dr. Matthias Stallmann, who exposed me to graph theory, game theory, economics, integer and linear programming. I also thank Dr. Jon Doyle for giving me some vital education lessons, specifically, not taking things for granted. I am also thankful to Dr. Steffen Heber and Dr. Robert Hartwig, whose clarity of exposition was refreshing. I also thank Dr. Douglas Reeves for taking the time off to be on my defense committee. I express my gratitude to Dr. David Thuente and the entire administrative staff who helped me in every administrative problem I have fallen into with understanding and amazing patience and sincerety. Without saying, I am indebted to my parents, Sharad Joshi and Vrishali Joshi whose austerity and non-authoritative approach to my upbringing has been crucial for my motivation throughout my lifetime. I am also thankful to my brother, Abhijit and his wife Gauri for their encouragement and financial support. I am also very grateful to Sangtae Ha and Sangjoon Park for their encouragement, as well as coffee.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview of Video Communication Systems

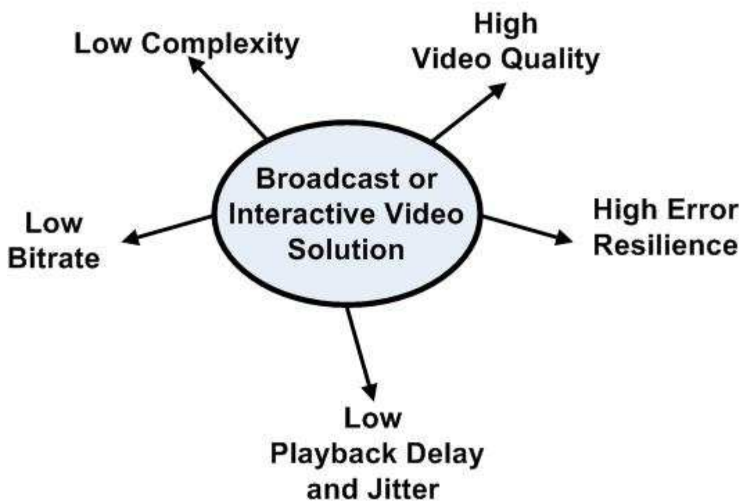


Figure 1.1: Typical Design Requirements of Video Communication Systems

Figure 1.1 shows the different requirements from a typical video communication system. These constraints are typically contradictory [1], in the sense that trying to optimize for one is generally unoptimal for the others, with the notable exception of attempting to achieve low complexity and high quality at the same time. It is believed that finding a specific solution to a distributive or interactive video communications problem has to be based on a compromise in terms of these constraints; video quality, bitrate, playback delay, robustness against channel errors, and the associated implementation complexity. Analyzing these tradeoffs and proposing solutions to various video communication problems, is the field of video communications [1].

1.2 Project Goals

The project was motivated by a desire to seek a high quality, computationally lightweight, and low bitrate video encoding scheme, and in the process evaluate the performance of the proposed schemes.

1.3 Literature Survey and Scientific Contribution of this work

Lossy video coding is now a maturing subject, albeit still not an exact science. It is largely predicated on the structure of image coding, which is a more mature subject. While no global optimization approach yet exists for the problem of image coding, in practice image coding has now stabilized into a 3-stage process of transform, quantization, and entropy coding. Video coding goes a step further, and incorporates a motion estimation/compensation loop, giving it both temporal and spatial decorrelation. This technique of exploiting both the temporal and spatial correlation in a group of frames is known as Conditional Replenishment, which can be done either as a residual-error based scheme or as replenishment of entire blocks that have changed across images, which we call as Non-Residual based scheme. While the follow-on stages of quantization and entropy coding have seen improvements in recent standard designs, it is primarily the subject of advanced spatial and temporal decorrelation that has been the focus of recent designs, especially the latest international standard: ITU/H.264 — ISO/IEC/MPEG-4 Pt. 10 (AVC) [4]. While attention is being given to seek optimality vis-a-vis bitrate-distortion, relatively less attention has been given to the time complexity of these schemes. The reason lossy coding has gained popularity is because it is possible to achieve great deal of compression at a relatively small cost of quality, and thus is the de-facto way to do video compression for applications such as televisions etc.

On the other hand, although lossy video coding has been studied extensively in recent literature, lossless video coding has been given relatively less attention especially in the literature because of its very high bandwidth demands rendering it impractical for everyday use. Lossless video encoding has applications in medical and scientific applications, studio quality archival purposes [5], and extremely high quality video playback.

However, although the field of lossless video encoding has not been extensively dealt with in scientific literature, there is a cottage industry around the development of proprietary lossless video codecs. Several companies and individual people have developed their own codecs. To name a few they are: Alparysoft lossless codec, MSU, CorePNG, YULS, FFV1, Flash Screen Video, HuffYUV, Lagarith, MSU lossless Codec, Pegasus Lossless JPEG, Sheer Video, VMWare Video, CamStudio Screen Codec, DosBox Capture Codec, Fraps, Microsoft Screen Codec, TechSmith Screen Capture Codec, Flash Video Encoder etc.

A study undertaken by the MSU Graphics and Media Lab (Video Group), in March 2007 [7], compares several of these existing lossless video encoders, and leads to the conclusion that amongst the codecs that they evaluated, in Video Capture and Video Editing area, the overall clear winner is Lagarith [29]. In Maximum Compression area the overall winner is YULS and the most balanced and flexible codec is FFV1 (relatively good speed and high compression for various presets). Their study shows that typically, the compression ratio achieved by the best lossless codec is around 3.15 [7], which can be more than the compression ratio achieved by H.264 (Best Quality). We also compare some of these codecs including the MSU codec with our codec, and confirm their findings. Through our results we see that our codec performs better than Lagarith for some of the sequences, but we note that MSU consistently gives better bitrates than the codec presented in this work. However, MSU takes much more time to do the encoding than the codec presented herein. We thus claim that the codec presented in this work is amongst the best low-computation-time and low-bitrate lossless codecs around as of today. MSU is a proprietary codec and the algorithm has not been released. Lagarith is quite fast as we also confirm, but not always better than the work presented herein. In terms of achieving a good competitive balance of compression rate and encoding speed, this work does present a fast lossless Conditional Replenishment based lossless codec.

In the design of video encoding schemes, considerable effort has also gone into the design of high quality but low delay, real time transmission of desktop images, leading to the development of remote desktop applications. These applications typically employ a server-client based approach and is called as Server based computing. Such a Server-based computing (SBC) model is fast becoming an increasingly popular approach for delivering computational services with reduced administrative costs and better resource utilization. Nieh et. al. in [8], examine how effectively SBC architectures support multimedia applications. Typically, the approach used for remote desktop applications, is to encode only the blocks that have changed across consecutive images, and then transfer this encoded data over the channel. As noted before, this approach is called as Conditional Replenishment. In this thesis we first classify video encoding schemes in their ability to do Residual Conditional Replenishment and Non-Residual Conditional Replenishment, and evaluate the performance of each. Several proprietary codecs developed, fall into the categories that would be evaluated in the thesis.

Amongst existing Server-based video encoding applications, Virtual Network Computing (VNC) is a popular application for encoding of desktop screens and transferring them over the network, typically for remote desktop use. The VNC standard proposes the use of the following approaches to encode data and is known as the RemoteFrameBuffer(RFB) Protocol.

1. Raw : The raw encoding simply sends width*height pixel values. All clients are required to support this encoding type. Raw encoding minimizes processing time.

2. CopyRect: The Copy Rectangle encoding principle is to copy updates from other areas of the screen if they already exist there. The only data sent is the location of a rectangle from which data should be copied to the current location. Copyrect could also be used to efficiently transmit a repeated pattern. The VNC Project however does not have a scheme to decide efficiently which areas of the screen are the most likely candidates for thie copyrect.

3. RRE: The Rise-and-Run-length-Encoding is basically a 2D version of run-length encoding (RLE). In this encoding, a sequence of identical pixels are compressed to a single value and repeat count. In VNC, this is implemented with a background color, and then specifications of an arbitrary number of subrectangles and color for each. This is an efficient encoding for large blocks of constant color.

4. CoRRE: This is a minor variation on RRE, using a maximum of 255x255 pixel rectangles. This allows for single-byte values to be used, reducing packet size. This is in

general more efficient, because the savings from sending 1-byte values generally outweighs the losses from the (relatively rare) cases where very large regions are painted the same color.

5. Hextile: Here, rectangles are split up in to 16x16 tiles, which are sent in a predetermined order. The data within the tiles is sent either raw or as a variant on RRE. Hextile encoding is usually the best choice for using in high-speed network environments (e.g. Ethernet local-area networks).

6. Zlib: Zlib uses zlib data compression library to compress raw pixel data. This encoding achieves good compression. Support for this encoding is provided for compatibility with VNC servers that might not understand Tight encoding which is more efficient than Zlib in nearly all real-life situations.

7. Tight: Like Zlib encoding, Tight encoding uses zlib library to compress the pixel data, but it pre-processes data to maximize compression ratios, and to minimize CPU usage on compression. Also, JPEG compression may be used to encode color-rich screen areas. Tight encoding is usually the best choice for low-bandwidth network environments (e.g. slow modem connections).

In [8], Nieh et. al. report that typically, using Server Based Computing, a compression gain of 2 or more is generally achieved at an acceptable quality. In our findings in this thesis, we will also see that we achieve similar compression performance using Non-Residual, block based Conditional Replenishment using Zlib. We would also demonstrate the performance of a scheme quite similar in spirit to VNC Tight encoding. The VMWare Video Codec, uses this VNC RemoteFramebuffer protocol [9]. The codec SBB-NR-CR-ZLIB presented in this thesis is very similar to VMware Video Codec and VNC-Zlib encoding.

Further, a high quality image compression system, has been described by Nakao et. al. in [2] wherein they use JPEG-2000 to encode the blocks that have changed across consecutive images. The codec SBB-NR-CR-JPEG-2000 presented in the second chapter of this thesis, is comparable with the codec presented by Nakao. et. al [2]

In [5], Memon and Sayood investigate lossless video coding techniques by extending the 2-D prediction based methods used in lossless image coding into 3-D, thus considering the frames in groups in the temporal domain. This paper describes that the standard block based difference technique for motion compensation that has been so successfully used in lossy encoding does not work well when used with lossless compression since the statistical properties of the residuals are different from still images. A similar 3-D approach has been proposed by Brunello et. al. [10], and another similar but embedded approach has been proposed by [13]. Our work confirms the findings leading to a similar conclusion that residual encoding might not be always useful. Implementing an improved variant of the motion estimation error metric, based on minimizing the prediction error of the lossless image encoding scheme, instead of using SAD, as proposed by Memon et. al. [5] is part of future work. Also, the optimal prediction idea proposed by [10] is part of the future work, although in this thesis we do work on a 3-D prediction, but our work cannot be considered optimal. Also, the optimal prediction considered in [10] leads to around 10% improvement in bitrate, which thus if applied to the codecs based on Motion Estimation that we present, would still lead to a bitrates obtained less than those obtained by the most efficient codec considered in this work.

In [12], Alzina et. al. discuss a novel method based on a fixed database model using longest prefix matching with the elements in the database(dictionary), followed by enhanced run length coding, followed by lossless encoding, and they propose lossy extensions to it. Through this work, we also attempt a similar longest prefix matching motion estimation, however, our results do not compare favorably against theirs. Discussion about why such a result was obtained would be considered in the thesis.

To the best of the author's knowledge, no equivalent study of the schemes presented herein and a classification of video encoding schemes as given herein has been reported in the literature to date. Thus, through this work, we make contribution to the literature for lossless encoding of video by describing the design and performance gains/losses that can be achieved through use some block based Conditional Replenishment schemes. We independently confirm the findings of [5], [12], and propose directions of future research. We also confirm the findings of [3] of the gains achieved by Histogram Packing, and propose a schematically simple and lightweight video encoding scheme that outperforms several other propreitary schemes. We also make the contribution of classifying existing video encoding schemes into categories based on Conditional Replenishment.

1.4 Approach and Organization of the thesis

The thesis starts with a model of Conditional Replenishment where in, there is no motion estimation (and correspondingly no motion compensation) and Conditional Replenishment is instead done with a very simple boolean-block-based-pel differentiator, which checks, per block, if any pixel/pel in the block is different from the corresponding pixel/pel in the block of the previous image.

If even one pel differs in the entire block, across any one of the components, then that entire block of pixels is tagged that it needs to be Conditionally Replenished. That block is marked present in the "text image", and the pixel-data in the block is put into the corresponding location in the "video image".

This process is repeated for every block in the current image. At the end of this process we get a "text image" signifying a map of presence of blocks that have been Conditionally Replenished, and the "video image" which contains the data necessary for reconstructing the image at the decoder. We call this process as Simple Block Based, Non-Residual, Conditional Replenishment (SBB-NR-CR). Now, once this scheme for doing Conditional Replenishment has been established, we will experiment with its components. Applying this technique to the images that make up a video sequence, make these schemes video encoding schemes.

Figure 1.2 shows the schemes that are presented in this work. In this figure, the schemes marked with "FW" mean future work, and the schemes that are numbered are discussed in this work. The organization of the thesis is as follows:

1. The first section of the thesis would start off with describing how Simple Block Based, Non-Residual, Conditional Replenishment (SBB-NR-CR), followed by lossy/lossless image coding leads to a compression gain. This chapter will then put forth an argument for the lossless encoding of the "Text image" created after SBB-NR-CR. Then the chapter would discuss how SBB-NR-CR followed by lossy JPEG-2000 of the "Video Image" leads to results worse than the ones achieved with SBB-NR-CR followed by lossless JPEG-2000 encoding of the "Video Image". By being convinced of this fact, an alternative to JPEG-2000, that being JPEG-LS, would be considered as a probable candidate for fast lossless image coding. It would be confirmed by experiments that JPEG-LS is computationally lighter than lossless JPEG-2000, and thus would be the

compression scheme of our choice for doing SBB-NR-CR followed by lossless encoding of the "Video Image". This scheme would be called as SBB-NR-CR-JPEG-LS.

2. In this second section, we would work on an attempt to improve the compression gains achieved in SBB-NR-CR-JPEG-LS. Most of the results of this work are in the negative in the sense of not leading to any compression improvement, except for the improvement achieved by online Histogram Packing for sparse histogram images, and the compression gain achieved for some images using ZLIB instead of JPEG-LS. The chapter considers the motivation for trying out the following approaches and explains the theory behind them.

   (a) CleanImage Lossless Encoding Schemes

      i. CleanImage JPEG-LS: This is also called Motion JPEG-LS.

      ii. CleanImage ZLIB: This is also called Motion ZLIB.

   (b) SBB-NR-CR-Lossless using ZLIB: In this scheme, the "text" and "video" image is encoded using ZLIB. This scheme is similar to Tight-VNC-ZLIB.

   (c) Non-Residual Lossless Schemes (SBB-NR-CR-Lossless) based on Histogram Packing

      i. SBB-NR-CR-OfflineHistogramPacked-JPEG-LS: Image obtained after SBB-NR-CR subjected to offline Histogram Packing followed by JPEG-LS encoding.

      ii. SBB-NR-CR-OnHistogramPacked-JPEG-LS: Image obtained after SBB-NR-CR subjected to Online Histogram Packing, followed by JPEG-LS encoding.

      iii. SBB-NR-CR-ZLIB: Encoding of both, Video image and Text Image using ZLIB.

      iv. SBB-NR-CR-OfflineHistogramPacked-ZLIB: Image obtained after SBB-NR-CR subjected to offline Histogram Packing, followed by ZLIB encoding.

      v. SBB-NR-CR-OnHistogramPacked-ZLIB: Image obtained after SBB-NR-CR subjected to Online Histogram Packing, followed by ZLIB encoding.

   (d) Non-Residual Lossy Schemes (SBB-NR-CR-Lossy) with Intra Frame/Key Frame refresh

(e) Residual Lossless Schemes (SBB-R-CR-Lossless) based on Longest Prefix Matching Motion Estimation and Gray Code Variant

i. SBB-R-CR-LPME-ZLIB: Longest Prefix Matching Motion Estimation (LPME) using XOR (addition modulo 2), followed by ZLIB encoding of the bitstream.

ii. SBB-R-CR-Gray-LPME-ZLIB: LPME using Gray Codes.

(f) Residual Lossless Schemes (SBB-R-CR-Lossless) based on SAD Motion Estimation and Histogram Packing

i. SBB-R-CR-SAD-WithSign-JPEG-LS: Simple Block Based, Residual Conditional Replenishment. The Residual is found out using Motion Estimation with sign, followed by JPEG-LS.

ii. SBB-R-CR-SAD-WithoutSign-JPEG-LS: Simple Block Based, Residual Conditional Replenishment. The Residual is found out using Motion Estimation without sign, followed by JPEG-LS.

iii. SBB-R-CR-SAD-WithSign-ZLIB: Simple Block Based, Residual Conditional Replenishment. The Residual is found out using Motion Estimation with sign, followed by ZLIB.

iv. SBB-R-CR-SAD-WithoutSign-ZLIB: Simple Block Based, Residual Conditional Replenishment. The Residual is found out using Motion Estimation without sign, followed by ZLIB.

v. SBB-R-CR-SAD-WithSign-OnlineHistogramPacked-JPEG-LS: SBB-R-SAD-WithSign undergoing Online Histogram Packing, followed by JPEG-LS.

vi. SBB-R-CR-SAD-WithoutSign-OnlineHistogramPacked-JPEG-LS: SBB-R-SAD-WithSign undergoing Online Histogram Packing, followed by JPEG-LS.

vii. SBB-R-CR-SAD-WithSign-OnlineHistogramPacked-ZLIB: SBB-R-SAD-WithSign undergoing Online Histogram Packing, followed by ZLIB.

viii. SBB-R-CR-SAD-WithoutSign-OnlineHistogramPacked-ZLIB: SBB-R-SAD-WithSign undergoing Online Histogram Packing, followed by ZLIB.

(g) Residual Lossy Schemes (SBB-R-CR-Lossy) with an Intra/Key Frame:
This thesis does not cover these schemes and comparison with the codecs presented herein is part of future work. This is the category into which the MPEG-2, H.264 schemes would fall into.

3. This section will give the Benchmarking Results i.e. performance comparison of each of the schemes outlined here with reference to bitrate and encoding times, comparing them with codecs such as H.264 High Profile, Fidelity Range Extensions, with QP=1, i.e. highest quality that can be achieved using H.264, with AlparySoft's Lossless codec, with MSU, with Lagarith, with Techsmith Screen Capture codec which is used by the Camtasia desktop grabbing tool and the Lossless Adobe Flash Codec.

4. The fourth section would be the conclusions section, which would discuss that lossless SBB-CR is a good way to do SBB-NR-CR especially if high quality but low playback delay and real time encoding is desired. Further, at relatively similar computational cost, a lossy SBB-NR-CR leads to a poorer quality video, and lossy SBB-NR-CR does not lead to significantly less bitrate because of additional content in the video image (because of the error in the reconstructed image). This work further shows that SBB-NR-CR-HistPacked-JPEG-LS gives the best performance amongst all the schemes tested and for all the sequences tested when the sequence is made up of natural images. The work further shows that SBB-NR-CR-ZLIB can give good performance in comparison with Lossless Flash Video Encoder for sequences with no natural images, and containing high frequency components. This work shows that the promise of LPME-ZLIB and SAD based lossless video encoding cannot be achieved using the techniques as has been used here and that it needs some more careful design. The work also shows that SBB-NR-CR-OnlineHistPacked-JPEG-LS is a method that is competitive to the best lossless video codecs known.

5. The fifth section would describe directions of future work, and scope for parallelization.

1.5 Definitions

This section describes the definitions of various terms used in the thesis.

1. Compression Rate $= \frac{Image\ Size\ After\ Compression\ (in\ unit\ U)}{Image\ Size\ before\ compression\ (in\ unit\ U)}$

2. Bits per Pixel (bpp) β = compression Rate * δ bits/pixel, $\delta = 24$ bits/pixel, for our input RGB images.

3. Bitrate@α frames per second (fps) = (Image Height in pixels * Image Width in pixels * β * α fps) bits per second(bps).

4. SBB-NR-CR : Simple Block Based, Non-Residual, Conditional Replenishment

5. SBB-R-CR : Is an abbreviation for SBB-NR-CR-R, meaning Simple Block Based, Residual, Conditional Replenishment.

6. PSNR : The phrase peak signal-to-noise ratio, often abbreviated PSNR, is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. Because many signals have a very wide dynamic range, PSNR is usually expressed in terms of the logarithmic decibel scale (db). The power of a signal $x(t)$ is usually represented by $x^2(t)$ in the spirit of electrical power which states that the amount of work that is done by an electric current is $P = I^2R$, or $P = V^2/R$. Thus if the signal $x(t)$ is a measure of the current or the voltage at time t, then its power is approximated to be $x^2(t)$. If we consider two images g and g', where g' is a noisy approximation of g, then the Mean Squared Error (MSE) is given as :

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{i=0}^{n-1}||g(i,j) - g'(i,j)||^2$$

$$PSNR = 10.log_{10}(\frac{MAX_g^2}{MSE}) = 20.log_{10}(\frac{MAX_g}{MSE})$$

where MAX_g is the highest intensity value in the intensity range of g depending on the bit depth. For color images with three RGB values per pixel, the definition of PSNR is the same except the MSE is the sum over all squared value differences divided by 3mn.

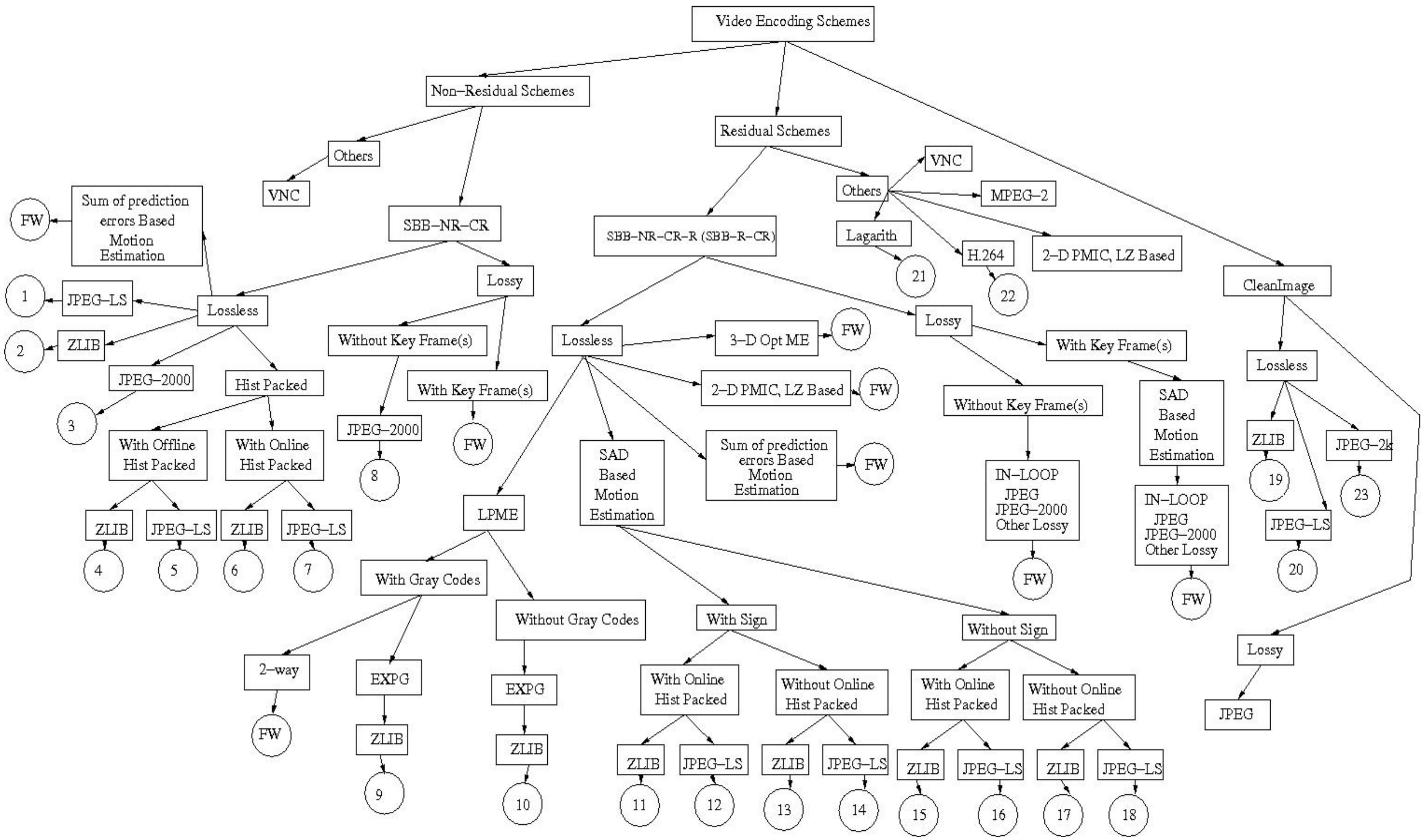


Figure 1.2: Classification of Video Encoding Schemes based on Replenishment type

Chapter 2

Simple Block Based, Non-Residual, Conditional Replenishment

2.1 Simple Block Based, Non-Residual, Conditional Replenishment (SBB-NR-CR) Based Video Encoding System, Architecture Design

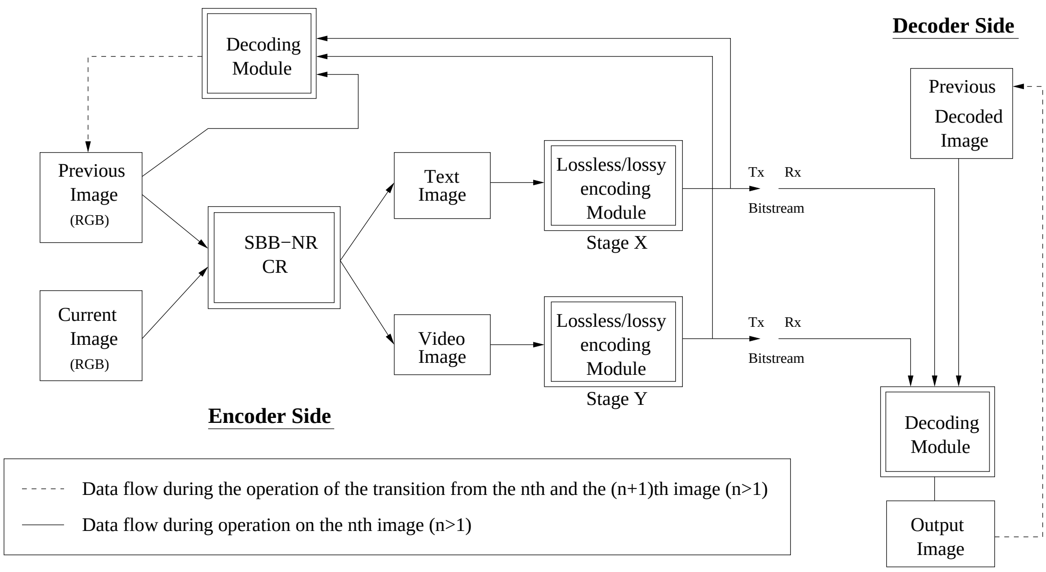


Figure 2.1: (SBB-NR-CR) Non-Residual Conditional Replenishment Based Video Encoding System Block Diagram

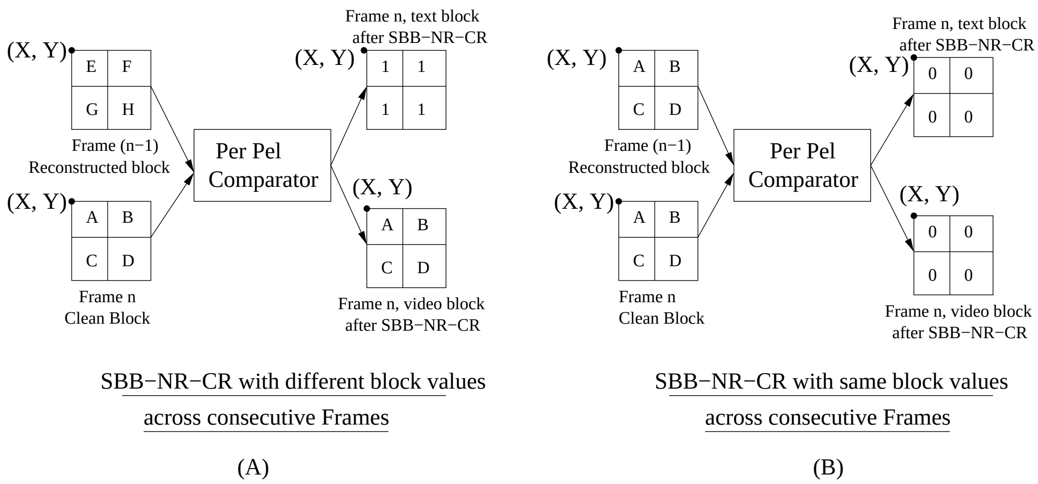


Figure 2.2: Simple Block Based-Non-Residual Conditional Replenishment (SBB-NR-CR) Encoding Process

Figure 2.1 shows the Conditional Replenishment encoding/decoding scheme block diagram. For all frames(n) with $n > 1$, we use the encoding scheme as shown in Figure 2.1, and the first frame is transferred over the channel as it is without any compression. The Simple Block Based, Non-Residual, Conditional Replenishment (SBB-NR-CR) module as shown in the Figure 2.1 operates using the technique shown in Figure 2.2. The principle behind the operation of SBB-NR-CR is as follows: Consider a block in the frame n located at position (X, Y). Lets assume without loss of generality, that this block is made up of pels A, B, C and D. Now, consider the block in the previous reconstructed frame i.e. in reconstructed $(n-1)^{th}$ frame located at the same position (X, Y). Then, each pel from the block ”ABCD” is compared with the block ”EFGH” from the previous frame. Even if 1 pel differs in value, then the entire block ”ABCD” is carried forward, as it is, to the ”video” image, and the ”text” image is marked with 1s as shown in Figure 2.2 (a). Figure 2.2 (b) shows what is to happen when the block in the previous frame i.e. $(n-1)^{th}$ reconstructed frame is exactly same as the block in the n^{th} frame at position (X, Y). In this case, the text image is marked with 0s and the video image is also marked with 0. This process is called Simple Block Based Non-Residual Conditional Replenishment, and this represents the SBB-NR-CR block shown in Figure 2.1.

The SBB-NR-CR process takes two images as input, the current image and the previous reconstructed image and outputs a ”text image” and a ”video image”. The next step is encoding these two images. Stage X and Stage Y as shown in Figure 2.1 are used to accomplish this goal. We can do lossy as well as lossless encoding of these images. The dotted line in the figure is used to show that the previous image during the encoding of the $(n+1)^{th}$ image is to be constructed from the existing $(n-1)^{th}$ frame and the decoded/reconstructed differential/residual data, because doing so guarantees that the reconstructed images on the decoder side has the same quality (PSNR) as could have been observed after the decoding of an image on the encoder side.

2.2 Encoding of "Text" image and "Video" image

We will discuss the following approaches for encoding the "text" and "video" image data

1. Lossy text scheme: This leads to a very minor reduction in bitrate over lossless text.
2. Lossless text and lossy video scheme: Results show that for the image sets considered, lossy video leads to an overall bitrate higher than the one achieved by lossless SBB-NR-CR, despite the ability of SBB-NR-CR lossy encoding to encode at arbitrary bitrates.
3. Lossless test and lossless video, which is the scheme that gave the best results for SBB-NR-CR.

We would use the wavelet based JPEG-2000[1] as the image encoding engine to compress the text image and the video image. An analysis of JPEG-2000 parameters was made and the encoding was parameterized on the parameters that showed the best performance.[2]

2.2.1 Lossy Text

In this scheme, we use JPEG-2000 in the lossy mode to compress the "text" image obtained after SBB-NR-CR. Because the text acts like a map for the "video" image data, having a lossy "text" leads to misconstruction of the "video" reconstructed image. It is seen that because the "text" image is made up of only 0s and 1s, and is more amenable to run length coding than the video image, it is compressed to very small files, even when

[1] See Appendix

[2] See Appendix for our choice of parameter selection

using the lossless entropy coding in the JPEG-2000. Doing a lossy compression for this text portion, leads only to a degradation of quality at tiny bitrate savings. Consequently the idea of doing a lossy text encoding was dropped.

2.2.2 Lossless Text, Lossy Video

The motivation for checking the performance of lossless text and lossy video was that lossy encoding allows us to compress a single image at an arbitrary rate. These numbers were collected by running SBB-NR-CR-JPEG-2000 Quarterscreen Forest sequence and LG sequence and then finding the mean over all these sequences. On average, for quarter screen video sequences, it can be seen from Table 2.1 that lossy SBB-NR-CR performs worse than lossless SBB-NR-CR for our sample image set. Note that here, the input set is only Quarterscreen, and we would need to take more measurements before we can say whether lossless video is better/worse than lossy video. In this section, we will talk about why we have observed such a result. The next section about lossless video, will put forth an argument to decide the winner amongst lossless video and lossy video.

Table 2.1: Comparison of Quarterscreen Video SBB-NR-CR encoding schemes

Scheme	Overall Compression Rate	Bits Per Pixel	PSNR	Bitrate@30fps(bps)
Text Lossless, Video@0.3	0.2	4.99	48.45 48.65 48.65	138,018,816
Text,Video Lossless	0.06	1.44	INF INF INF	39,813,120
No CR; Lossy@0.06	0.06	1.44	31.40 31.53 31.53	39,813,120

Reason for poorer performance of lossy SBB-NR-CR compared to lossless SBB-NR-CR

This effect can be explained by looking at Figure 2.3. This figure shows how lossy encoding of the previous video image, can lead to video image blocks that would be absent had we done lossless encoding for the previous video image. As has been shown in Figure 2.3, if the reconstructed block in the $n-1^{th}$ frame is with pel values "EFGH", and the clean block in the current frame is say "ABCD", then when we pass these two images to

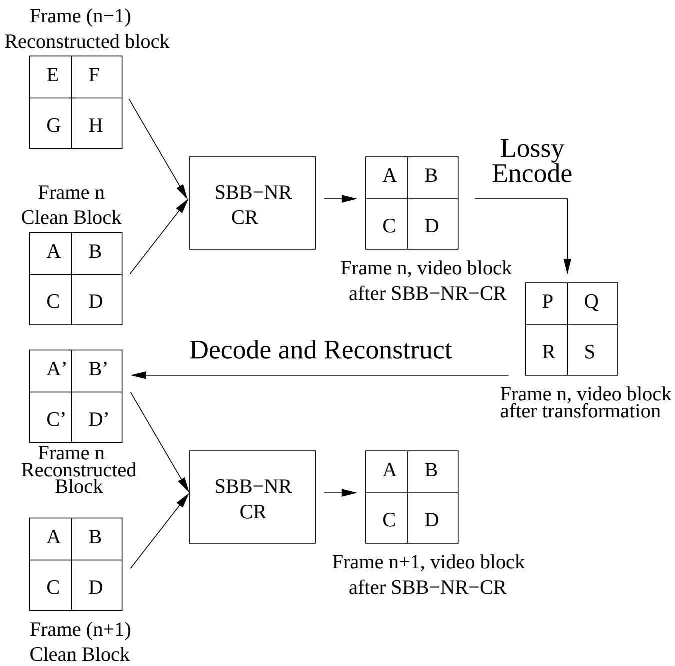


Figure 2.3: Effect of lossy encoding on video after SBB-NR-CR

SBB-NR-CR, the output would be the block "ABCD" assuming that the block ABCD, and EFGH differ in at least 1 pel value. We need to make this assumption because we are trying to demonstrate why SBB-NR-CR lossless performs better than SBB-NR-CR lossy. If we do not make this assumption, then after SBB-NR-CR we would know that this block has not changed across consecutive frames, and we not bother encoding it. Hence, we assume that block ABCD, and block EFGH differ in at least 1 pel value. If we attempt to encode this video image after SBB-NR-CR using lossy encoding, like JPEG-2000 in lossy mode, then, after transformation by JPEG-2000, the block would appear to be say "PQRS". After decoding this PQRS block using JPEG-2000, we are going to see some loss because this block was encoded lossily. Consequently therefore, if we now reconstruct the block, we would get the block A'B'C'D' which could be different than the block "ABCD". Here we consider the case in which it IS different, which is a reasonable assumption. Further now, if the next frame, i.e. $(n+1)^{th}$ frame contains that same block, in the same coordinates, then after SBB-NR-CR, the block "ABCD" would be encoded again. Note that had we encoded the previous video image losslessly, then after reconstruction, we would have got back the "ABCD" block, leading to the absence of that block in the video image of the SBB-NR-CR

(a) (b)

Figure 2.4: (a)Ringing Effect for Lossy encoding (b) Lossless encoding

for the $(n+1)^{th}$ frame.

Figure 2.4 shows two images, in which one was encoded losslessly using JPEG-2000, and the other was encoded lossily. It can be seen that the Figure 2.4 (a) has much perceptual (and PSNR as well) poorer quality than 2.4 (b). It can be noted that at sharp edges there is a smearing effect occurring. This effect is the ringing effect which comes as an artifact introduced by DWT subject to lossy quantization. Note that DWT reversible, if not subjected to a quantization loss, when reconstructed/decoded leads to a perfect reconstruction and leads to no ringing effects. This is because of using reversible mother wavelets for the reversible JPEG-2000. In the SBB-NR-CR encoding of the next frame, this newly reconstructed image would be used as the reference as explained before. An example of how a lossy encoding of the previous video image led to distortion in the reconstructed previous image, thereby causing stray blocks in the current video image is shown in Figure 2.5.

As a consequence of this ringing effect, the video part image size keeps increasing. The video blocks can reduce only under the condition if a new block undergoes no quantization loss in the lossy encoding process. In such cases, the block is made of low frequency components, and is not affected by the quantization. However, whenever the block contains high frequency components, which is not so atypical for unnatural sequences such as desktop grabs, then that leads to error propagation, and leads to loss in compression efficiency. With reference to Figure 1.2, we are discussing about Scheme 8, and comparing it with Scheme 3. This error propagation explains the results seen in Table 2.1, that the lossless text and lossless video scheme leads to a better overall rate as well as better quality.

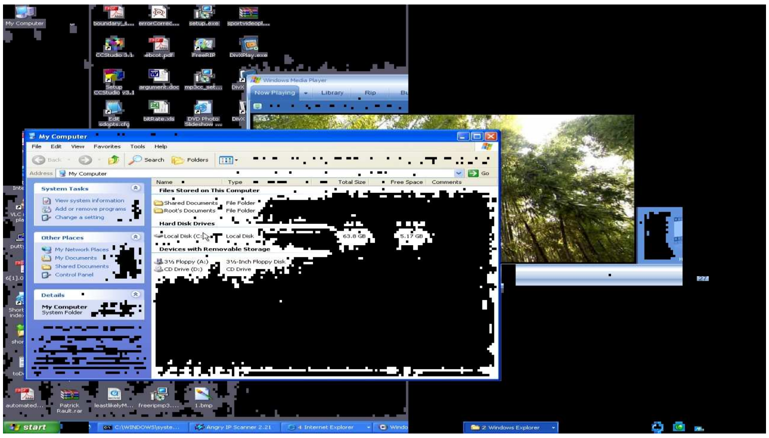


Figure 2.5: Effect of ringing effect in the previous "Video" image on the current "Video" image

In order to convince ourselves whether a SBB-NR-CR-Lossy is really worse than SBB-NR-CR-Lossless, we would need to first measure the performance of SBB-NR-CR-Lossless. This is done in the next section.

2.2.3 Lossless Text, Lossless Video

This section is divided into two sub-parts. In the first part we measure the performance of SBB-NR-CR with lossless text and lossless video in comparison with Clean-Image compression using JPEG-2000, which is called as MJPEG when enclosed in an AVI header. Once we are through with this test we would have an average compression rate that was achieved by the SBB-NR-CR-Lossless. In order to see the performance of SBB-NR-CR-LossyWithoutKeyFrame with SBB-NR-CR-Lossless, we subject SBB-NR-CR-Lossy to encode at the same rate as was observed by the lossless scheme, and we intend to check its quality. We are following this methodology, because it is possible to encode lossily at ANY arbitrary rate, and thus if we wanted to really measure which one of these two schemes is better (w.r.t the project goal), we must see how good/bad the quality of the SBB-NR-CR-Lossy scheme at the rate of the lossless is. This is covered in second part of this section.

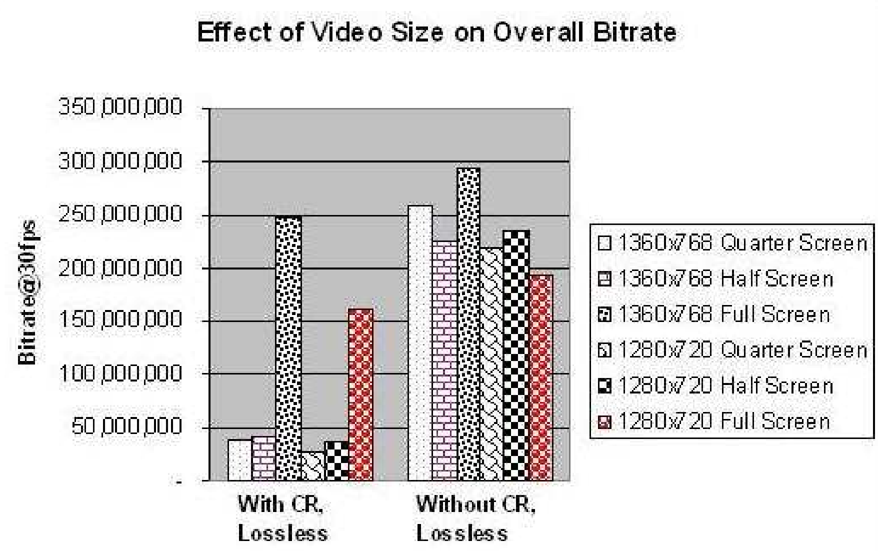


Figure 2.6: Bitrate@30fps comparison of schemes using JPEG-2k, with and without SBB-NR-CR

Rates achieved with and without SBB-NR-CR

This section presents a comparative study of lossless SBB-NR-CR and lossless encoding without Conditional Replenishment, for 6 different image data sets. Three data sets are of the same size, but differ in the amount of video data present in them. The data sets used the ForestFight and LGE video running on the screen along with other activity on the desktop like reading PDF documents, browsing the Internet etc. The first image set is of size 1280x720, and the second is of size 1360x768.

It can be seen from Figure 2.6, that using lossless CR gives us better rates, and consequently bitrates compared to the ones achieved without CR. This happens because of the high correlation between consequent images, which leads to very small video part of the image, and thus leads to good overall rate. It can also be seen that as the percentage of the video portion in the image increases, the margin of advantage that the Conditional Replenishment scheme had, reduces. These images were sequences that were around 4000 frames, and the rate shown here is the average rate achieved over a typical desktop like usage with varying proportions of video content on the screen. The video playing on the screen carried the Forest sequence and the LG sequence. The exact values of the rates achieved are as given in the following Table 2.2.

Table 2.2: Rate comparison of schemes using JPEG-2k, with and without CR

ImageSize	Input type	Compr. Rate using SBB-NR-CR	Bitrate@30fps using SBB-NR-CR	Compr. Rate CleanImage	Bitrate@30fps CleanImage
1360x768	Quarter Screen Video	0.051	38,353,305	0.344	258,696,806
1360x768	Half Screen Video	0.565	42,489,446	0.2984	224,404,439
1360x768	Full Screen Video	0.33	248,168,448	0.388	292,252,188
1280x720	Quarter Screen Video	0.0435	28,864,512	0.33	218,972,160
1280x720	Half Screen Video	0.056	37,158,912	0.354	234,897,408
1280x720	Full Screen Video	0.2411	160,042,106	0.2897	192,231,014

SBB-NR-CR-Lossy-WithoutKeyFrame Vs SBB-NR-CR-Lossless

In this section, we will see the performance of the SBB-NR-CR Lossy scheme in comparison with SBB-NR-CR-Lossless scheme. Adding Keyframes and lossy SBB-NR-CR with Keyframes, would still cause the increase in video content as was explained in the previous section. Exactly how much would it make a difference is not considered in the thesis and is part of future work. Looking at Table 2.2, for each of the sequences in that table, we know mean rates achieved by the SBB-NR-CR-Lossless. We now subject SBB-NR-CR-LossyJPEG-2000 to encode at this rate and then measure the PSNRs. As said before, we are following this methodology, because it is possible to encode lossily at ANY arbitrary rate, and thus if we wanted to really measure which one of these two schemes is better, we must see how good/bad the quality of the SBB-NR-CR-Lossy scheme at the rate of the lossless is. The input data set is the same as was considered in the measurements taken for Table 2.2, and the following results are observed. Note carefully how the Av. Compr Rate shown in Table 2.3 is exactly the same compression rate as shown for SBB-NR-CR-Lossless in Table 2.2. With reference to Figure 1.2 we are referring here to Scheme 23 and Scheme 3.

Table 2.3: PSNR and Bitrate of the Lossy-Video Scheme

ImageSize	Input type	PSNR (Red)	PSNR (Green)	PSNR (Blue)	Av. Compr Rate	Bitrate@30fps (bps)
1360x768	Quarter Screen Video	29.85	29.85	29.85	0.051	38,353,305
1360x768	Half Screen Video	36.82	37.01	36.45	0.056	42,113,433
1360x768	Full Screen Video	48.13	48.17	48.17	0.33	248,168,448
1280x720	Quarter Screen Video	30.41	30.64	30.34	0.0435	28,864,512
1280x720	Half Screen Video	31.47	31.50	31.50	0.056	37,158,912
1280x720	Full Screen Video	49.24	49.26	49.26	0.24119	160,042,106

Thus, by looking at these numbers, we can say now, that if we try doing SBB-NR-CR-Lossy-WithoutKeyFrame at the same rate at which we try doing SBB-NR-CR-Lossless, then we achieve quality of images as shown above. Note that the bitrate would be the same. It can be seen from Table 2.3 that for QuarterScreen Video and HalfScreen video, the SBB-NR-CR-Lossy scheme performs quite poorly in terms of quality. For Fullscreen ”video” sequences, the quality is sufficiently good, but is still lossy.

Thus, in the preceding sections, we have seen how SBB-NR-CR-Lossless compares with SBB-NR-CR-Lossy. It can be seen that for QuarterScreen and FullScreen ”Video” image sequences SBB-NR-CR-Lossless outperforms SBB-NR-CR-Lossy, but for FullScreen images, SBB-NR-CR-Lossy gives good quality images. Because the goal of the project is to seek a high-quality scheme, we now drop the idea of SBB-NR-CR-Lossy schemes because we have seen that for SBB-NR-CR, lossless image encoding performs better than lossy image encoding on average over different kinds of input images.

2.3 Comparison of JPEG-2000 and JPEG-LS results

2.3.1 Comparison of Compression Rates

It was seen in the previous section that a lossless scheme for doing SBB-NR-CR is better than a lossy scheme without Keyframes, both in terms of bitrate and quality. This section will compare the two codecs, JPEG-2000 and JPEG-LS on the basis of bitrates obtained by each in encoding output of the SBB-NR-CR. Table 2.4 shows the bitrates obtained for the same input images as in the previous section.

Table 2.4: JPEG-2000 Vs JPEG-LS Bitrates

ImageSize	Input type	SBB-NR-CR JPEG-2000 Average BitRate@30fps (bps)	SBB-NR-CR JPEG-LS Average BitRate@30fps (bps)	SBB-NR-CR JPEG-LS Minimum BitRate@30fps (bps)	SBB-NR-CR JPEG-LS Maximum BitRate@30fps (bps)
1280x720	Quarter Screen Video	28,864,512	39,134,448	0	159,252,480
1280x720	Half Screen Video	37,158,912	38,856,243	0	177,129,317
1280x720	Full Screen Video	160,042,106	150,847,470	0	325,638,000
1360x768	Quarter Screen Video	38,353,305	55,662,181	0	307,238,640
1360x768	Half Screen Video	42,511,374	34,651,099	0	316,998,720
1360x768	Full Screen Video	248,168,448	300,767,146	0	538,450,329

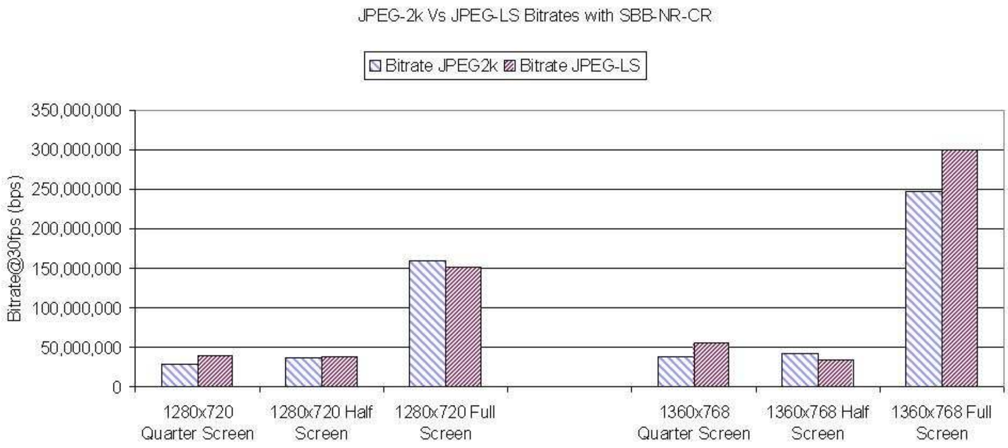


Figure 2.7: Comparison of JPEG-LS and JPEG-2000

Figure 2.7 shows data described in the Table 2.4 in the form of a graph. It can be seen from the results obtained above, that SBB-NR-CR-JPEG-LS and SBB-NR-CR-JPEG-2k-lossless are comparable in terms of the bitrates achieved.

2.3.2 Comparison of encoding time

In order to see whether JPEG-LS offers any advantage in terms of encoding time in comparison with JPEG-2000, we measured the time it took for doing Conditional Replenishment by each method, for a small input sample. Table 2.5 shows the average encoding time for the text portion and the video portion images, created during the Conditional Replenishment process.

Table 2.5: JPEG-2000 Vs JPEG-LS Encoding Timings

Image Type	Text Enc. Time Average (sec)	Vid Enc. Time Average (sec)
QuarterScreen JPEG-2k	0.888	0.941
QuarterScreen JPEG-LS	0.01804	0.04009
HalfScreen JPEG-2k	0.908	0.941
HalfScreen JPEG-LS	0.01408	0.03999
FullScreen JPEG-2k	0.829	1.218
Fullscreen JPEG-LS	0.01531	0.18733

It can be seen from this that JPEG-LS performs 10-20 times faster than JPEG-2k for our input image sequences.

2.4 Conclusion for SBB-NR-CR

From the previous sections, it was seen that SBB-NR-CR is an effective way to do video compression. It was seen that depending on the quantity of ”video” data across images, compression can be achieved ranging from 3 times to 10 times using SBB-NR-CR-JPEG-LS. We also found out for the video sequences that we tested, that SBB-NR-CR-Lossless performs better on average(quality as well as encoding time) than SBB-NR-CR-Lossy, meaning that we achieve better compression on average for the sequences tested, doing Non-Residual Conditional Replenishment in a lossless way than doing it in a lossy way. We also see that the JPEG-LS implementation encodes significantly faster compared to the JPEG-2k implementation.

As mentioned before, it should be noted at this point that in our SBB-NR-CR-Lossy scheme we do not test what happens when we insert a key frame after every specific number of frames. We have only noted that for the data set that we considered, doing SBB-NR-CR-Lossy-Without-Key-Frame is worse than SBB-NR-CR-Lossless-Without-Key-Frame. Inserting a key frame and checking the performance has been excluded from the thesis and remains a task of the future. As mentioned before, the task of the project was to seek a low bitrate, high quality and computationally lightweight scheme, and because we have seen that SBB-NR-CR-Lossless gives unmatchable quality, and is better in terms of bitrate (on average) than SBB-NR-CR-Lossy (even WITH Keyframes), we opt for working on lossless schemes. We have also seen that lossless image encoding schemes based on JPEG-LS outperform JPEG-2k in terms of encoding time. As a consequence we will pursue this thread of doing SBB-NR-CR-Lossless, and attempt to improve on the performance of the this scheme and propose variants.

Chapter 3

Design of Simple Block Based Conditional Replenishment Variants

In the previous chapter we discussed the SBB-NR-CR scheme. In this chapter we will discuss variants of this scheme, with the goal of attempting to improve the compression rate. We will discuss the motivations for, and the design of each of the following schemes. This chapter describes the techniques and the next chapter will discuss the results. These schemes can be seen in Figure 1.2.

1. CleanImage Lossless Encoding Schemes
   (a) CleanImage JPEG-LS: This is also called Motion JPEG-LS.
   (b) CleanImage ZLIB: This is also called Motion ZLIB.
2. SBB-NR-CR-Lossless using ZLIB: In this scheme, the ”text” and ”video” image is encoded using ZLIB. This scheme is similar to Tight-VNC-ZLIB.
3. Non-Residual Lossless Schemes (SBB-NR-CR-Lossless) based on Histogram Packing
   (a) SBB-NR-CR-OfflineHistogramPacked-JPEG-LS: Image obtained after SBB-NR-CR subjected to offline Histogram Packing followed by JPEG-LS encoding.
   (b) SBB-NR-CR-OnHistogramPacked-JPEG-LS: Image obtained after SBB-NR-CR subjected to Online Histogram Packing, followed by JPEG-LS encoding.

(c) SBB-NR-CR-ZLIB: Encoding of both, Video image and Text Image using ZLIB.

(d) SBB-NR-CR-OfflineHistogramPacked-ZLIB: Image obtained after SBB-NR-CR subjected to offline Histogram Packing, followed by ZLIB encoding.

(e) SBB-NR-CR-OnHistogramPacked-ZLIB: Image obtained after SBB-NR-CR subjected to Online Histogram Packing, followed by ZLIB encoding.

4. Non-Residual Lossy Schemes (SBB-NR-CR-Lossy) with Intra Frame/Key Frame refresh

5. Residual Lossless Schemes (SBB-R-CR-Lossless) based on Longest Prefix Matching Motion Estimation and Gray Code Variant

   (a) SBB-R-CR-LPME-ZLIB: Longest Prefix Matching Motion Estimation (LPME) using XOR (addition modulo 2), followed by ZLIB encoding of the bitstream.

   (b) SBB-R-CR-Gray-LPME-ZLIB: LPME using Gray Codes.

6. Residual Lossless Schemes (SBB-R-CR-Lossless) based on SAD Motion Estimation and Histogram Packing

   (a) SBB-R-CR-SAD-WithSign-JPEG-LS: Simple Block Based, Residual Conditional Replenishment. The Residual is found out using Motion Estimation with sign, followed by JPEG-LS.

   (b) SBB-R-CR-SAD-WithoutSign-JPEG-LS: Simple Block Based, Residual Conditional Replenishment. The Residual is found out using Motion Estimation without sign, followed by JPEG-LS.

   (c) SBB-R-CR-SAD-WithSign-ZLIB: Simple Block Based, Residual Conditional Replenishment. The Residual is found out using Motion Estimation with sign, followed by ZLIB.

   (d) SBB-R-CR-SAD-WithoutSign-ZLIB: Simple Block Based, Residual Conditional Replenishment. The Residual is found out using Motion Estimation without sign, followed by ZLIB.

   (e) SBB-R-CR-SAD-WithSign-OnlineHistogramPacked-JPEG-LS: SBB-R-SAD-WithSign undergoing Online Histogram Packing, followed by JPEG-LS.

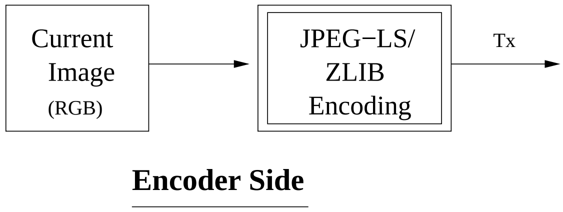


Figure 3.1: Clean Image Lossless Encoding

(f) SBB-R-CR-SAD-WithoutSign-OnlineHistogramPacked-JPEG-LS: SBB-R-SAD-WithSign undergoing Online Histogram Packing, followed by JPEG-LS.

(g) SBB-R-CR-SAD-WithSign-OnlineHistogramPacked-ZLIB: SBB-R-SAD-WithSign undergoing Online Histogram Packing, followed by ZLIB.

(h) SBB-R-CR-SAD-WithoutSign-OnlineHistogramPacked-ZLIB: SBB-R-SAD-WithSign undergoing Online Histogram Packing, followed by ZLIB.

7. Residual Lossy Schemes (SBB-R-CR-Lossy) with an Intra/Key Frame:
This thesis does not cover these schemes and comparison with the codecs presented herein, and with its very similar variants such as MPEG-2/H.264 is part of future work.

3.1 CleanImage Lossless Encoding Schemes

These schemes are characterized by encoding the image directly without subjecting them to SBB-NR-CR. In this thesis we discuss the following CleanImage encoding schemes.

3.1.1 CleanImage JPEG-LS

In this technique, we do not perform SBB-CR, but rather the original image (called as Clean Image here) is encoded directly using JPEG-LS. The schematic block diagram is as shown in Figure 3.1.

3.1.2 CleanImage ZLIB

In this technique, we do not perform SBB-CR, but rather the original image (called as Clean Image here) is encoded directly using ZLIB. The schematic block diagram is as shown in Figure 3.1.

3.2 SBB-NR-CR-ZLIB

In this scheme, the text and the video image as shown in Figure 2.1 are compressed by the ZLIB compression library. The Flash Screen Video Codec from Adobe is a 16x16 Block Based codec, and the similarity of the performance of SBB-NR-CR-ZLIB with the performance of the Adobe video codec leads us to guess that Adobe FLV is based on SBB-NR-CR-ZLIB, but we have no way of verifying because Adove FLV is not an open format.

3.3 Non-Residual Lossless Schemes (SBB-NR-CR-Lossless) based on Histogram Packing

The basic tool that is used in designing point operations[1] (and many other operations as well) is the Image Histogram. The histogram H_g of the digital image g is a plot or graph of the frequency of occurrence of each gray level in g[2]. Hence, H_g is a one-dimensional function with domain $\{I_{min}, ..., I_{max}\}$, and range extending from 0 to NM (for the image of size NxM). The histogram is given explicitly by $H_g(k) = J$ if the image g contains exactly J occurrences of the gray level (Intensity Value) k. The algorithm to compute the image histogram involves a simple counting of gray levels, which can be accomplished even as the image is scanned.[15]

In [16], Armando J. Pinho discusses the methods that can be used to improve the compression performance of lossless compression of images with sparse histograms. Histogram packing is a known preprocessing method capable of producing improvements in compression rate if applied prior to lossless compression of images having sparse histograms. [16] There are two ways of performing this Histogram Packing in the literature, that being Offline Histogram Packing, and the other being Online Histogram Packing.

Paulo J.S.G and Armando J. Pinho discuss in [17], that the Histogram Packing method described in [16] can be best understood in terms of its effect on the total image

[1] Image Processing Operations that are applied to individual pixels/pels

[2] Note that in this thesis we deal with continuous tone images, i.e. an image whose components(as measured by device) have more than 1 bit per sample. A continuous tone image is one where each color at any point in the image is reproduced as a single tone, and not as discrete halftones

variation. The packing transformation reduces the total variation in the image, yielding a smaller variation that is easier to compress. We now discuss these two schemes:

3.3.1 Offline Histogram Packing

Offline Histogram Packing is obtained through the construction of an one-one order preserving mapping, h, from the image intensity values, I, into a contiguous subset of $\mathbb{N}_0$. Let the range of the intensity I in the image g, be I_{min} to I_{max}. Further, let I_{α_0} be the smallest image intensity value that has non-zero occurrence in the image g, I_{α_1} be the second smallest intensity has non-zero occurrence in the image g and in general I_{α_i} be the i^{th} smallest intensity value with an occurrence non-zero number of times in g, and $I_{\alpha_{N-1}}$ the highest such intensity. If the image is made up of more than one component, then this same method is applied to every component separately. It suffices to describe the algorithm for just 1 component, because we can apply the same method for every component separately. The mapping h from non-zero intensities to $\mathbb{N}_0$ is as follows:

$$h = (I_{\alpha_0} \mapsto 0, I_{\alpha_1} \mapsto 1, I_{\alpha_2} \mapsto 2, I_2 \mapsto 2, ..., I_{\alpha_{N-2}} \mapsto N-2, I_{\alpha_{N-1}} \mapsto N-1) \tag{3.1}$$

Thus, when this mapping is applied to an image g, a new image $g_{ofhp} = h(g)$ is obtained which is more compression friendly than g [16].

In order to apply the mapping h to an image g, we must need to know apriori the histogram of the image. In this implementation, Offline Histogram Packing is done in a two-pass process. The first pass creates the histogram of the image g for each channel present in the image (In our case it is, R, G, and B), and the second pass packs the histogram using the mapping h for each channel. At the end of this process, we have an image g_{ofhp} i.e. g after Offline Histogram Packing (ofhp), that has total variance less than that of g. [17]

Note that this mapping needs extra side information for the inverse mapping. The inverse function is needed to be computed in order to reconstruct the original image from the histogram packed image. One way this mapping function h can be reversed is by building a map of the histogram as it existed *before* the Histogram Packing. For every image intensity I_j, where $I_{min} \leq I_j \leq I_{max}$, that exists in a component c of an image g, we mark a '1' in the mapping, and all the intensities that are not present are marked with zero. We apply this mapping for every I_j Thus this function is described as follows:

$$
\begin{aligned}
f(I_j) &= 0 \text{ if NumOccur}(I_j{=}0) \\
&= 1 \text{ if NumOccur}(I_j{\neq}0) \quad \forall I_{min}{\leq}I_j{\leq}I_{max}
\end{aligned}
$$

Once the decoder has this mapping f(·), as soon as it starts reconstructing the image, for every intensity that it sees in the histogram packed image, all it needs to do is to interpret this intensity I_k, as the I_k^{th} smallest value from I_{min} with a non-zero bin count, in the original histogram before Histogram Packing. Thus, if the decoder sees a '3' as an example, it should check f(·) to find the 3rd smallest element with a nonzero bin, and substitute that intensity value instead of the 3.

3.3.2 Online Histogram Packing

Offline Histogram packing assumes an apriori knowledge of the image histogram before the packing process begins. As has been noted before, this can mean doing a one-pass over the entire image and build the histogram. An alternative technique called Online Histogram Packing has been proposed by Armando Pinho in [18]. Similar to Offline Histogram Packing, this is also a preprocessing method, which has a variance reducing property. In [18], Pinho demonstrates that this method can give better performance than Offline Histogram Packing when used with JPEG-LS. The preprocessing technique can be explained as follows:

Let us assume that the preprocessor is going to handle sample x_t, and that it has already found n different intensity values given by $I^t = I_0, I_1, I_2, ..., I_{n-1}$. Let us also assume, without loss of generality, that $I_i < I_j, \forall_{i<j}$. Then the intensity mapping, h^t, that is used to map the k^{th} smallest sample I_k is given by:

$$h^t = (I_0 \mapsto 0, I_1 \mapsto 1, I_2 \mapsto 2, I_2 \mapsto 2, ..., I_k \mapsto k, I_{n-1} \mapsto n-1) \tag{3.2}$$

which maps ascending sorted intensity values into ascending sorted contiguous indexes. If $x_t \in I^t$, then symbol $h^t(x_t)$ is generated by the preprocessor. On the other hand, if $x_t \notin I^t$, then the symbol that is generated by the preprocessor is n, i.e. the value that is not present in the mapping table so far. In this case, this new intensity value is recorded. When the decoder is reconstructing the original image from the Online Histogram Packed image, it can identify that it needs to read a value from the recorded list, as soon as it hits upon an element which is not present in the mapping created so far. At the preprocessing side, the occurrence of a new value also implies the rearrangement of the intensity-value mapping h^t such that $I_i < x_t < I_{i+1}$, then the new mapping is:

$$h^{t+1} = (..., I_i \mapsto \beta, x_t \mapsto \beta + 1, I_{i+1} \mapsto \beta + 2, ...) \tag{3.3}$$

This Online Packing is performed for every component in the image, and the values that are found not to exist in I^t at stage t, are recorded per component.

The packing is reversed using a similar approach at the decoder side. If the symbol to be reverse-mapped corresponds to some index in use, then ordinary inverse mapping is performed. If not, then an intensity value is read from the list of intensities (the one that was created during the packing phase) and the mapping table is rearranged accordingly.

The application of these two packing methods as a preprocessing step in our Conditional Replenishment process, leads to the following schemes:

1. SBB-NR-CR-OfflineHistogramPacked-JPEG-LS
2. SBB-NR-CR-OfflineHistogramPacked-ZLIB
3. SBB-NR-CR-OnlineHistogramPacked-JPEG-LS
4. SBB-NR-CR-OnlineHistogramPacked-ZLIB

Each of the schemes listed above can be understood by looking at the schematic block diagram as shown in Figure 3.2, wherein the video image obtained after SBB-NR-CR is subjected to histogram Packing, and is then encoded either using JPEG-LS or using ZLIB. Both the histogram Packing schemes generate data that is required for successful decoding. This is also shown in the Figure 3.2. In the decoder, the decoder reads the encoded image, decodes it either using JPEG-LS or ZLIB depending on its encoding format, and then performs histogram unpacking so as to get the reconstructed image.

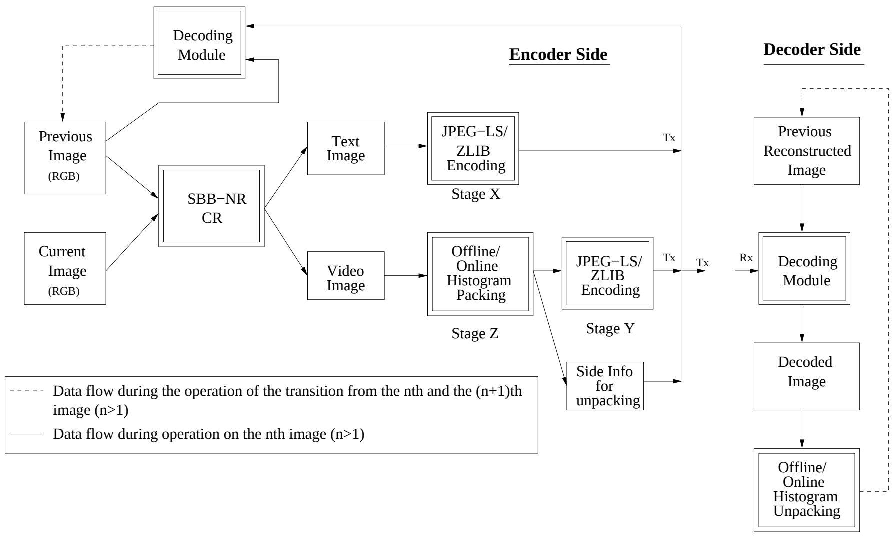


Figure 3.2: SBB-NR-CR-Histogram Packing Based Video Encoding Scheme

3.4 Non-Residual Lossy Schemes (SBB-NR-CR-Lossy) with Intra Frame/Key Frame refresh

In the previous chapter we discussed how the lossy encoding of the ”video” sequence led to an increase in bitrate in spite of the ability of lossless encoding to encode at arbitrary bitrate per frame. An important question then is what would happen if we were to introduce an intra frame in the SBB-NR-CR-Lossy scheme, and the answer to that is as follows : If α_k is the number of bits generated after k frames of SBB-NR-CR-Lossless, and if β_k is the number of bits collected after k frames of SBB-NR-CR-Lossy, we have seen for our test sequences that $\alpha_k < \beta_k$. If we now decide to introduce an intra frame to stop the error propagation in SBB-NR-CR-Lossy, we would have at the end of (k+1) sequences, the bits collected to be α_{k+1} and β_{k+1}. given by the formula:

$$\begin{aligned}&\alpha_{k+1} = \alpha_k + \gamma_{k+1}\\&\text{and, } \beta_{k+1} = \beta_k + \delta_{k+1}\\&\text{and, } \beta_{k+2} = \beta_{k+1} + \kappa_{k+2}\end{aligned}$$

where, γ_{k+1} is the number of bits for frame (k+1) after SBB-NR-CR-Lossless, δ_{k+1} is the number of bits for Intra frame (k+1) encoded losslessly or lossily. and κ_{k+2} is the number of bits for frame (k+2) after SBB-NR-CR-Lossy Depending on the values of γ_{k+1}, δ_{k+1} and

κ_{k+2}, we would expect to see different results.

Note that such a scheme based on SBB-NR-CR-Lossy encoding using an Intra frame refresh, can lead to a high quality video as well, if the encoding is near-lossless. This video encoding scheme has been left out of the schemes discussed herein, and we decided to focus on lossless schemes as was said in the first chapter. It remains a task of the future to see where exactly this scheme stands in the quality/bitrate schema.

3.5 Residual Lossless Schemes (SBB-R-CR-Lossless) Based on Longest Prefix Matching Motion Estimation and Gray Code Variant

In order to improve the compression efficiency of the Conditional Replenishment process, it was thought that the Conditional Replenishment scheme, if given a residual nature, will lead to a residual error magnitude constrained to a smaller range than the original pel range itself.[3], which could thus lead to smaller standard deviation, and consequently smaller bit encodings for each error, thereby leading to smaller sized bitstreams. We call such schemes as SBB-R-CR, signifying the residual nature of this scheme.

3.5.1 Related Work

In [12], Alzina, Szpankowski and Grama discuss a dictionary based pattern matching approach to image and video compression and propose lossy extensions to it. The central theme of a dictionary based method such as Lempel-Ziv algorithm is the notion of repetitiveness. Lossy extensions of Lempel Ziv consider approximate repetitiveness. If a portion of data recurs in an approximate sense, then subsequent occurrences can be stored as direct or indirect references to the first occurrence. This approximate recurrence of data may not be contiguous. Their theme of exploiting approximate repetitiveness is uniformly applicable to multimedia data from various sources such as text, images, video, and even audio. Such approximate repetitiveness can be hidden in various forms for different types

[3] Note that because we are dealing with continuous tone images, and each image intensity is represented from amongst the set of $\mathbb{N}_0$, the error is also represented by elements from $\mathbb{N}_0$. Had we been dealing with $\mathbb{R}$, then the although the error variance could in a small range, because of the binarization of such an error, there is a possibility of the residual needing exactly as many bits as the original value, or even more than the original pel value.

of media. Therefore, they also consider different distortion measures. This is in contrast with current state-of-the-art, where a different approach is used for compressing each of these types of data. Using the theoretical underpinning of Luczak and Szpankowski " A Suboptimal Lossy Data Compression Based on Approximate Pattern Matching, Atallah, Génin and Szpankowski implemented a lossy scheme for image compression. This scheme, called Pattern Matching Image Compression was based on one-dimensional pattern matching enhanced with some salient features. It was shown that for high quality images PMIC scheme is competitive with JPEG and wavelet image compression up to 1 bpp.

While in this thesis, we do not follow the Lempel-Ziv based pattern matching, nor their 2D-PMIC, we nevertheless borrow the idea of approximate repetitiveness from them, and search for a longest prefix match in the previous image. Thus in our scheme we consider the previous reconstructed frame as the only data present in the database, and then perform longest prefix matching motion estimation on this image. As a disclaimer, our method does not use any techniques from [12], and we make no claims of confirming/not confirming their findings. The results that we obtained from our scheme based on longest prefix matching motion estimation, were far less than satisfactory, and this section will cover the possibility of why this Longest Prefix Matching motion estimation gave such poor results.

In this section, we will first describe the motivations that led us to choose the design for the Longest Prefix Matching Motion Estimation, as was chosen.

3.5.2 Choice of Distance Metric for Approximate String Matching

Approximate String matching can be done by a variety of difference metrics such as Minimum Edit Distance, Episode distance, Longest common subsequence, Minimum Hamming distance etc. The metric is used for choosing the best match from a given text, for a given search pattern. If we wanted to achieve lossless encoding, a scheme based on Hamming distance would need us to encode the position of each bit that in different in the best-match-block and the search pattern. Consequently, after evaluating approximate string matching techniques such as based on Minimum Edit Distance, Episode distance, Longest common subsequence etc, we settled for a simple longest-prefix-match based scheme, purely because this scheme satisfied the required properties of having being a residual scheme, and also that it was very easily achievable by using the XOR operator. If XORing a, and b gives c, then by examining the leading zeros (MSB) of c, we can deduce the length of the longest prefix match. This scheme by virtue of its property of not needing to encode specific bit positions, and its simplicity was chosen over others.

3.5.3 The 2-D Longest Prefix Matching Motion Estimation Algorithm

Figure 3.3 shows the implemented version of the longest prefix based motion estimation. Following are the meanings of the various labels found in the Figure 3.3

1. P : Is the pattern for which a best match is being searched for, and has length PATTERN_SIZE . P is a group of bytes from the block in the current image.
2. T : is the total search window in which a suitable match for P is being searched for and has length TEXT_SIZE. T is a group of bytes picked up from the previous image.
3. W : is the current window in T that corresponds to a value of shift
4. Shift : Is an integer value that describes the horizontal displacement P in units of bytes.
5. E : is the error Vector, which comprises of the difference of the best matched W and P.

Fig.3.3 shows the different stages in the operation of the 1-D longest prefix motion estimation. Steps (a) to (e) show 5 possible states in the motion estimation process. Following is the description of these state transitions:

1. We start with the state (a) and check for the longest prefix match. This matching is done on a bit-to-bit basis, between the pattern window and the text window. The number of bits that match are say $\alpha 0$ in (a). In order to speed up the motion estimation, we take the decision that every subsequent shift happens in multiples of 8 bits, i.e 1 byte. This is a design decision to simplify the encoding and speed up the search process.
2. If the current value of longest Prefix Matched length is greater than any previous, then we replace the error vector E, by the difference

3. Thus we reach state (b), after 1 shift from state(a). In state (b) we again check for the longest prefix match. We carry on the same process, (TEXT_SIZE - PATTERN_SIZE) times, and finally reach state (e).

At the end of the longest prefix matching motion estimation process, we would have reached a minima in the sense that all other shifts lead to lower prefix length of matched data. We now need to encode the following information for a given pattern P. The data that need to be encoded to guarantee perfect reconstruction at the decoder side, from the transmitted bitstream are the following:

1. Sign of the error vector
2. The unsigned error Vector E (error)
3. The length in bits of the error vector E . (PATTERN_SIZE - Longest Prefix Match Length)
4. The sign of the motion Vector in X direction(signMvX),
5. The Motion Vector X (mvX) , which is the value of shift
6. The sign of the motion vector in Y direction (signMvY)
7. The Motion vector Y (mvY), which is zero, in our method, because we primarily work on a 1-D Longest Prefix Motion Estimation algorithm

3.5.4 Error distribution and Encoding Selection

Figure 3.4 shows the probability mass function of the first 30 error bins (to base 10). This plot also shows the distributions for some geometric functions, and the optimal pdf for the Exp-Golomb code. It is well known that if the elements in a set occur with a certain type of a geometric probability distribution, then the Huffman code for that data set becomes the Unary code, and this code is considered to be optimal (i.e. with minimum average length) for that distribution. This can be understood from observing Figure 3.5. It can be seen in this figure that for the two geometric distributions, the Huffman tree constructed leads to Unary prefix codes. The downside of Unary codes is that the code grows too quickly. It means that the code will grow to size $(n + 1)$ bits for encoding n.

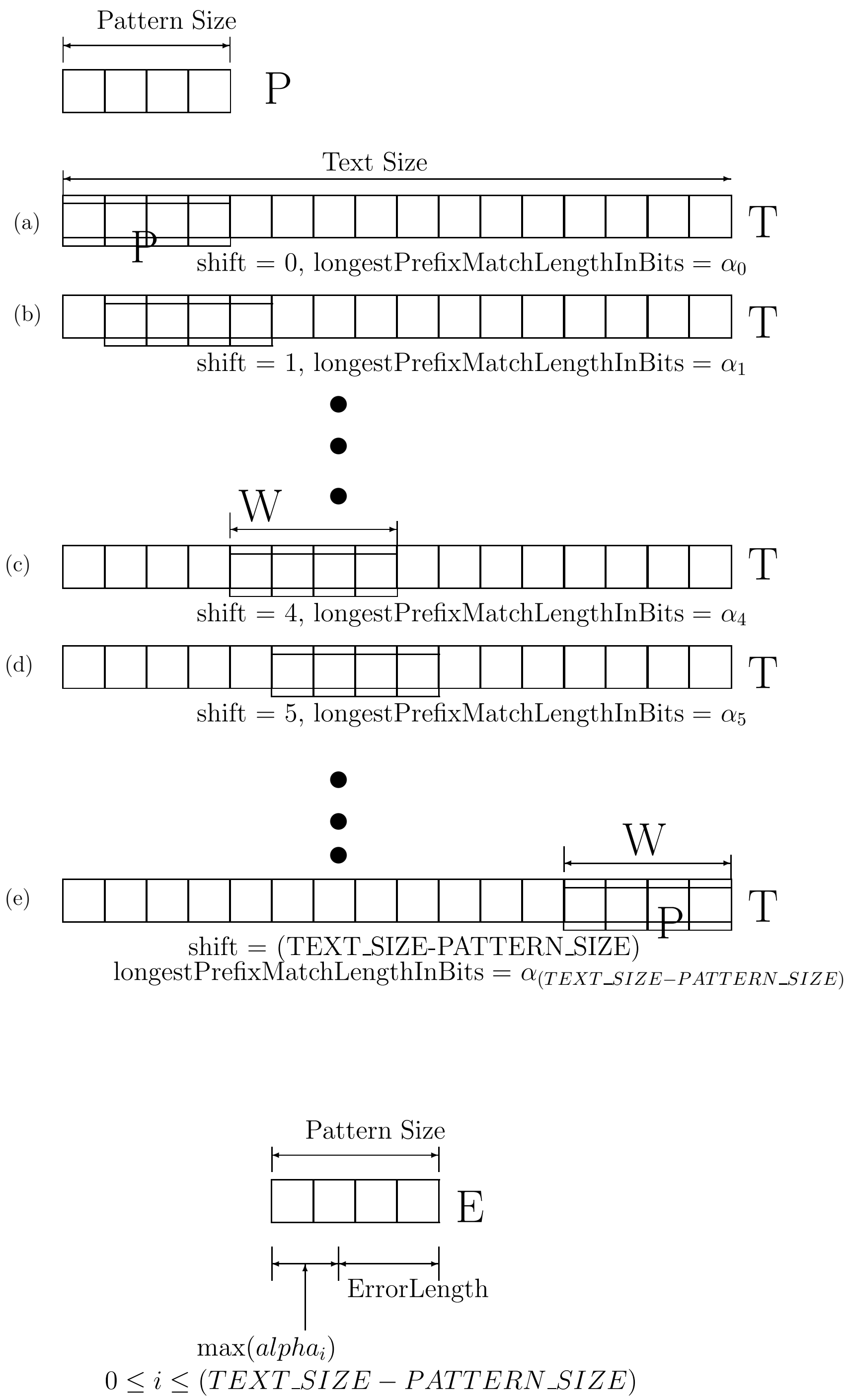


Figure 3.3: Longest Prefix Motion Estimation Working

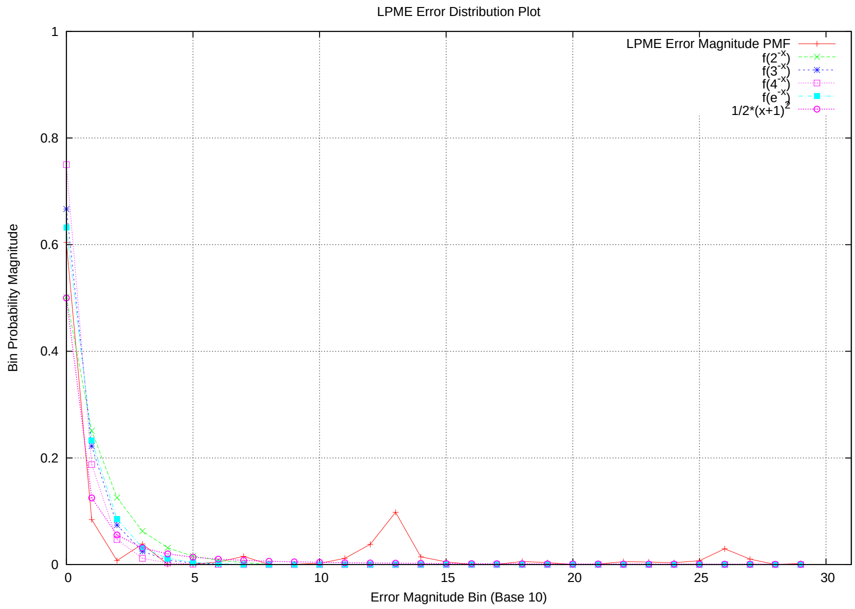


Figure 3.4: LPME Error Distribution

Looking at the error graph and plots of some geometric distributions, we can see that our error plot is more narrow but fatter tailed. For any distribution for which the Huffman tree is not completely skewed the optimal code is not a Unary Code.

In order to have larger tails, a multi-resolution approach to coding has been considered which has come to be known as the Golomb Code [21] in which the set of $\mathbb{N}$ is grouped into groups of size M each, and this code is denoted as Golomb-M. See Figure 3.6. Golomb Codes have the property that groups with smaller numerical values have shorter codes which are assigned in Unary, and the elements that fall within a group have a fixed length code. It is seen that the codeword length of Golomb codes grows at a much smaller rate than Unary codes. The Golomb codes are also what is known as, Canonical Huffman Codes.

Golomb-Rice codes is a special case of Golomb coding first described by, and independently invented by, Robert F. Rice. In these codes, the parameter M for Golomb codes is given by $M = 2^k$. Context based Golomb Rice coding is used as the entropy encoding stage in a number of lossless image compression including JPEG-LS and audio data compression methods. Finding the best m is a non-trivial task in general.

Exponential-Golomb-Codes, are codes where the parameter M is not constant

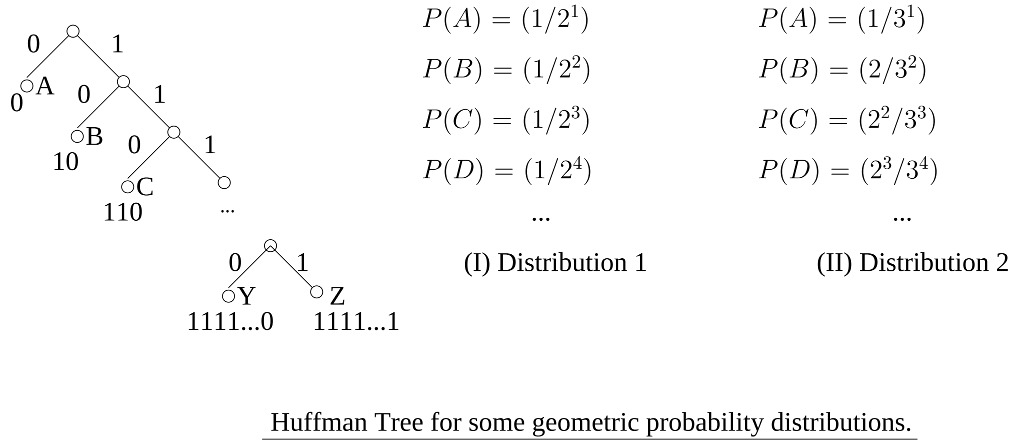


Figure 3.5: Huffman Tree for some Geometrically Distributed Symbols

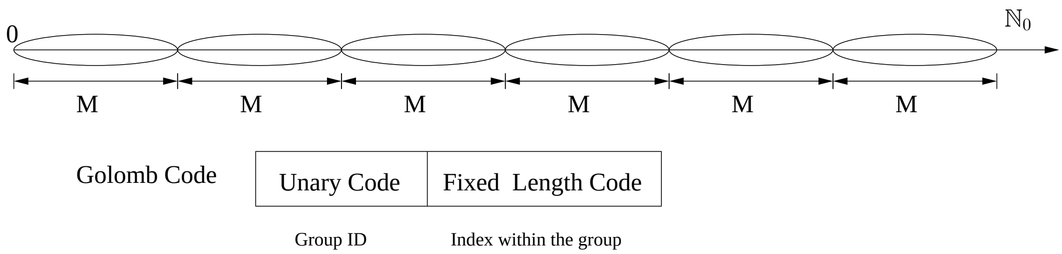


Figure 3.6: Golomb Codes

across the entire number line for $\mathbb{N}$, but instead has the nature as is shown in the Figure 3.7. Exp-Golomb Codes were first proposed by Teuhola in the context of run-length coding schemes, in 1978 [22], in which the code contains two parts – Unary code followed by fixed length code with size M varying in powers of 2. Exp-Golomb codes have a fatter tail than geometric distributions, and thus grow slower than Unary codes. The implied source pdf related to the Exp-Golomb code (or the UVLC) is given by $p(x) = 1/(2*(x+1)^2)$[23]

Table 3.1 shows the first entries 15 of the Exp-Golomb code. The construction of a codeword for a given index i is relatively simple and consists of two steps:

1. Extract the symbol γ representing the most significant bit (MSB) of $i+1$, i.e., γ $=\lfloor log(i+1)\rfloor$, and put it in an Unary representation of zeros. This part forms the prefix of the codewords shown in the MSB column of Table 3.1.

2. Append the γ least significant bits (LSB) of the binary representation of (i+1) to the

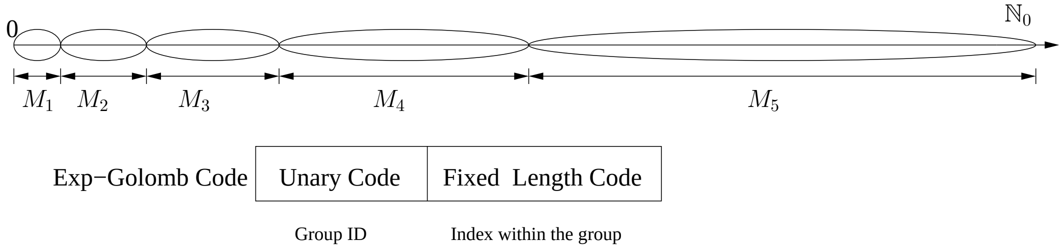


Figure 3.7: Exp-Golomb Codes

Table 3.1: Exponential Golomb Codes

Input	Input+1 in Binary	Exp Golomb Prefix(MSB)	Exp Golomb Suffix(LSB)	Exp Golomb Code	Segment	Offset
0	1	-	1	1	0	0
1	10	0	0	010	1	0
2	11	0	1	011	1	1
3	100	00	00	00100	2	00
4	101	00	01	00101	2	01
5	110	00	10	00110	2	10
6	111	00	11	00111	2	11
7	1000	000	000	0001000	3	000
8	1001	000	001	0001001	3	001
9	1010	000	010	0001010	3	010
10	1011	000	011	0001011	3	011
11	1100	000	100	0001100	3	100
12	1101	000	101	0001101	3	101
13	1110	000	110	0001110	3	110
14	1111	000	111	0001111	3	111
...						

prefix. The corresponding suffix bits are shown in the LSB column of 3.1

Exp-Golomb codes are variable length prefix codes, and are uniquely decodable. In fact, Golomb codes, and Golomb-Rice codes, and Exp-Golomb codes are a type of Canonical Huffman encoding.

3.5.5 LPME+EXPG+ZLIB encoding scheme

Thus from the preceding discussion it was seen that using a combination of longest prefix matching motion estimation, followed by the entropy coding using unsigned exponential Golomb codes, followed by the dictionary based ZLIB might be useful for improving the compression performance of the SBB-CR.

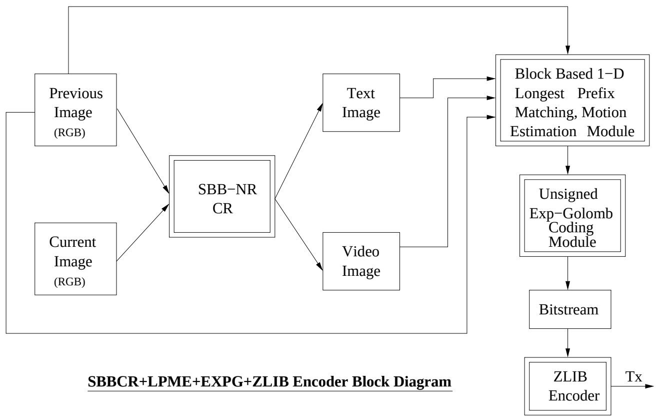


Figure 3.8: LPME+EXPG+ZLIB Conditional Replenishment System Block Diagram

Figure 3.8 shows the block diagram of the new concatenated encoding Conditional Replenishment scheme. Like in the SBB-NR-CR scheme, as was shown in Figure 2.1, this scheme first passes the current and the previous image through a SBB-NR-CR module. A block that changes across frames even in 1 pel value qualifies for the video frame. As before, the text frame is a map of the presence of information in the video frame. The text and video RGB images are then fed to the 1-D Longest Prefix Motion estimation module as was shown before. For every block which is present in the video image, the following elements need to be encoded . They are :

1. Row and Column of the block present in the video image.
2. The encoded Motion vectors, mvX and MvY with their sign and magnitude for each group of bytes group of bytes in the block (previously called as Text referred by T in the Figure 3.3)
3. The longest prefix matched least error data (E) with the sign and magnitude.

We pass the above entities to be encoded by the unsigned-Exp-Golomb coder. It can be observed that by property of the Exp-Golomb codes, we do not need to encode, the length of every code. We thus avoid encoding the "length" overhead of the Run-Length-Codes, and the Context switching overhead of the variable length codes, because we believe that we don't need it for the error distribution observed. It will however be seen in the results section that this scheme fares the poorest of all the schemes that have been tried. This brings up the possibility of an incorrect choice in choosing Exp-Golomb as the coding scheme.

Further, an important reason for the poor performance of this scheme is the strength of the longest prefix matching motion method. The better the match, the smaller the error, and smaller the error smaller its Exp-Golomb code and smaller the bitstream.

The LPME algorithm as outlined in the previous section if leads to an encoding with the length of the encoding bigger than the length of the Pattern block, then we use a "SKIP" mode and the block is put as it is without the Exp-Golomb of its best match error.

It was seen that such SKIP blocks were many in a stream, thus indicating that the tail is growing fatter for further errors. The task of implementing a more aggressive longest prefix motion estimation module is a task of the future.

The bitstream thus obtained after LPME and SKIP blocks, is then passed through ZLIB encoding so as to get the final compressed bitstream.

The next section will discuss the comparison of this scheme with SBB-CR and the compression gains achieved, and the timing overhead in doing the LPME+EXPG+ZLIB encoding.

3.5.6 Improving Longest Prefix Matching performance using Gray Codes

A Gray code is an encoding of numbers so that adjacent numbers have a single digit differing by 1. The term Gray code is often used to refer to a "reflected" code, or more specifically still, the binary reflected Gray code because the code is constructed using a "reflection". [19]

The figures shown in 3.10 were taken from [19]. Figure 3.10 (a) shows the 5-bit reflected Gray codes and Binary Codes. It can be observed that across all bit planes, on average, the Gray codes have lesser transitions than Binary Codes[19]. As can be seen from Figure 3.10 (a) (a), for this 5-bit code, the average number of transitions for Binary code $= (1+3+7+15+31)/5 = 11.4$, from 3.10 (a) (c), the average number of transitions for the reflected Gray code as given by : $(1+2+4+8+16)/5 = 6.2$. This feature has the property that consecutive numbers when encoded in the reflected Gray code have a higher probability of being similar than Binary Codes. Thus it is expected that using Gray codes might give a better LPME prefix match depending on the statistics of the input images.

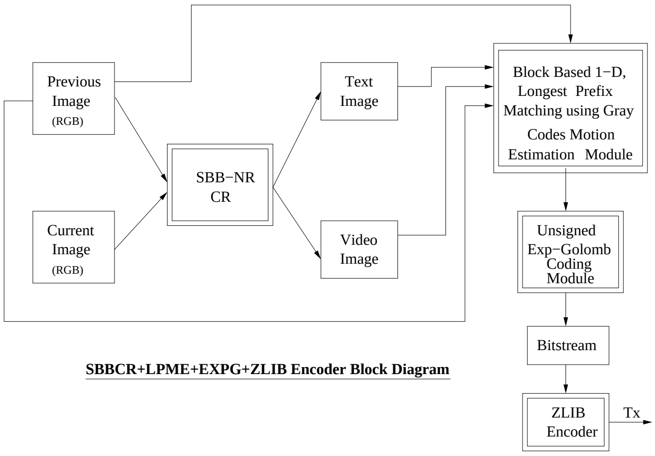
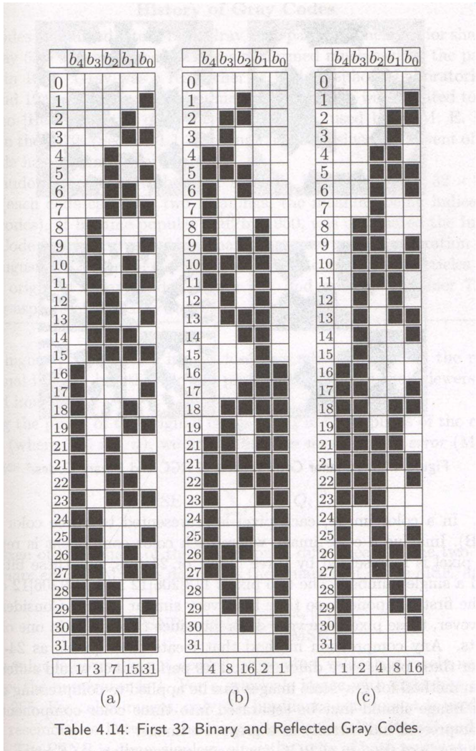


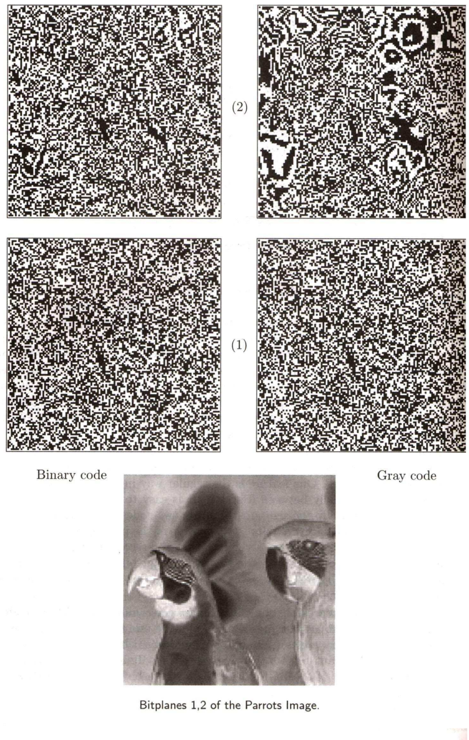
Figure 3.9: GRAY+LPME+EXPG+ZLIB Conditional Replenishment System Block Diagram

A demonstration of this property can be seen in the parrot images as shown in Figure 3.10 (b), (c) and (d). Here it is seen that the bit-planes 3,4,5,6, 7, 8 in reflected Gray Codes show much better correlation to the image than the bit-planes in binary. Further, it can be noted that the bit-planes towards the MSB have lower transitions in Gray Code than the Binary code, and is roughly half the number of transitions in a bit-plane for Binary Codes. This property means that across two consecutive images, which have a higher chance of similar pixel values, the Gray code longest prefix matching would perform better then the longest prefix matching using binary. Thus, the SBB-R-GRAY-LPME-ZLIB encoding scheme has block diagram as shown in Figure 3.9. The conversion of Binary-Gray and Gray-Binary was done using the simplification as shown in the appendix.

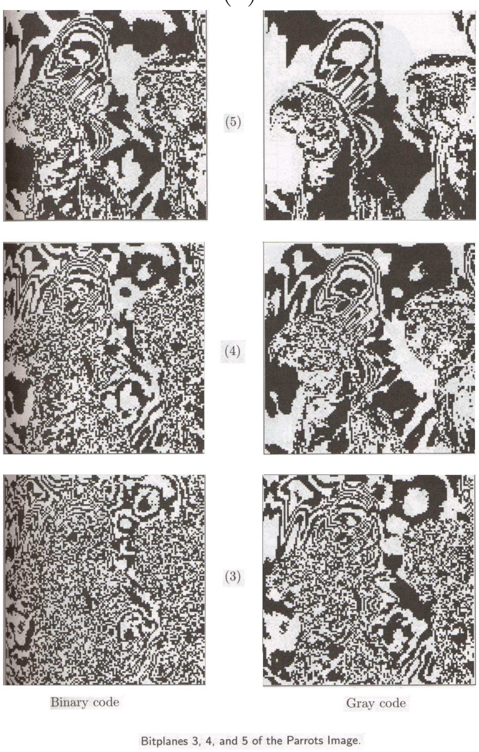
As we will see in the results section, although SBB-R-GRAY-LPME-ZLIB gives us so small a performance improvement for the natural images sequences that we tried and it takes such a long time it is not a useful scheme for real time video encoding. The Appendix shows how Binary Codes are converted to Reflected Gray codes and how Gray codes are converted back to binary. This is done by the encoder and the decoder respectively.



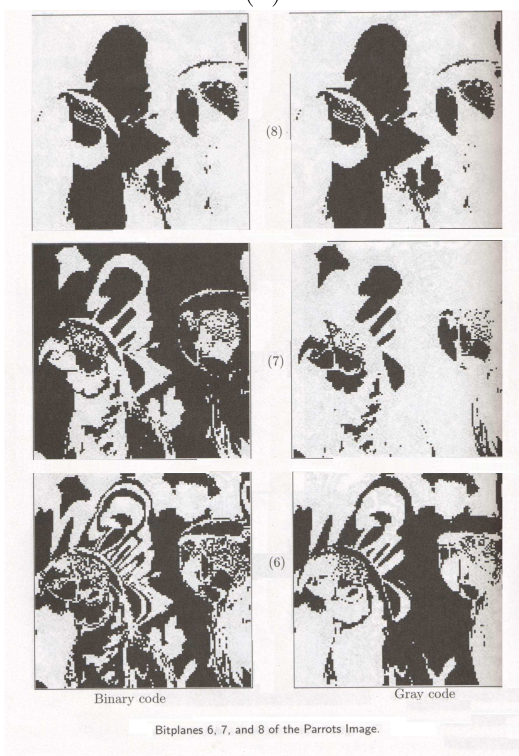
(a)



(b)



(c)



(d)

Figure 3.10: (a) Binary, Gray Comparison (b) bit-planes 1, 2 (c) bit-planes 3, 4, 5 (d) bit-planes 6, 7, 8

3.6 Residual Lossless Schemes (SBB-R-CR-Lossless) based on Block Based Motion Estimation and Histogram Packing

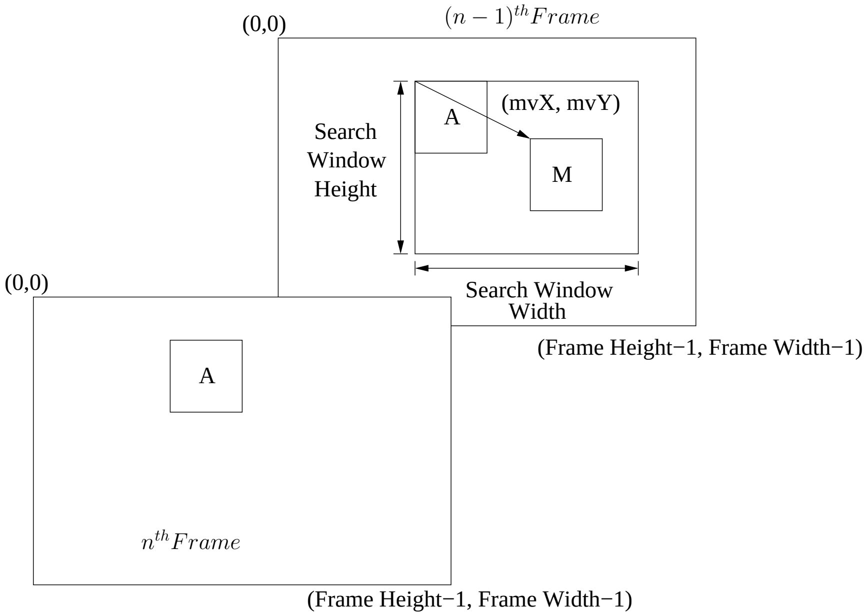


Figure 3.11: Motion Estimation Brute Force Technique

Successive video frames may contain the same objects (still or moving). Motion estimation examines the movement of objects in an image sequence to try to obtain vectors representing the estimated motion. In block motion estimation (BMC), the frames are partitioned in blocks of pixels. For the given frame under consideration, for each block in that frame, a search is made in the previous image(s) for the best matching block. This search area is called as the Motion Estimation Window. Figure 3.11 shows an example of motion estimation being done for a block at position A. The best match for this block A is obtained by searching in the search window in the previous frame. Different error measures can been used for finding the best match. Among others, the sum of absolute differences (SAD) and the minimum squared error (MSE) are commonly used. The block that has the minimum SAD or MSE is considered to be the best match for the block at position A. This best match is labeled as M in Figure 3.11, and the displacement of this block from the top left corner of the block A, is called the motion vectors for the best matching block, and are labeled as mvX and mvY representing displacement in the X direction and Y direction respectively.

After a best matching block M is found for a given block A, the next task is to create a residual block. This residual block can be found out with and without sign. It should be noted that in our SBB-R-CR scheme, we are going to subject the residual to an image compression method. This is unlike subjecting it to an in-loop Fourier/wavelet transform and were we using a frequency/wavelet domain transform, we could have made this residual as an integer instead of our insistence of fitting the residual into a byte. Doing the SBB-R-CR-Lossy in loop with an image encoding transform, and comparing that technique with MPEG/H.264 etc is part of future work, as is discussed in the future work section. The following sections describe how the residual can be obtained for further lossless image compression, with and without sign.

3.6.1 Residual With Sign

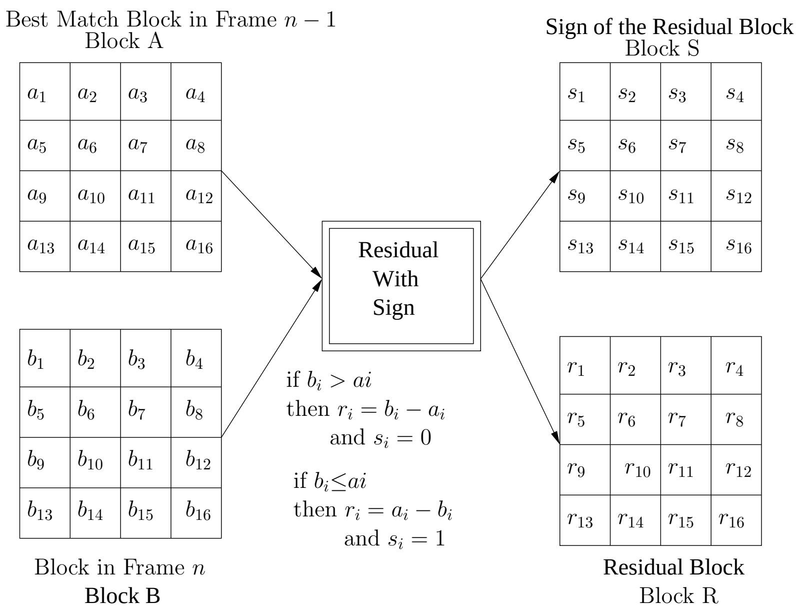


Figure 3.12: Residual With Sign

Figure 3.12 shows how the residual block is computed for a block B. If the block B is a 4x4 block of pels $\{b_1, b_2, ..., b_{16}\}$ and if block A is the best matching block using a metric such as SAD, then the residual block (R) with sign (S) is calculated using the following formula:

$$\begin{array}{lll} \text{if } b_i > a_i & \text{then} & r_i = (b_i - a_i),\ s_i{=}0 \\ \text{if } b_i \leq a_i & \text{then} & r_i = (a_i - b_i),\ s_i{=}1 \end{array}$$

Thus, if we carry out this process for each block in the image, then we get the residual image and sign image with the same dimensions as the input and reference image. This process if carried out as a stage after SBB-CR leads to the scheme which we call as SBB-R-CR-SAD-WithSign. The sign image and the residual image obtained here, can be compressed either by JPEG-LS or by ZLIB. Figure 3.13 shows the schematic block diagram of this Conditional Replenishment process.

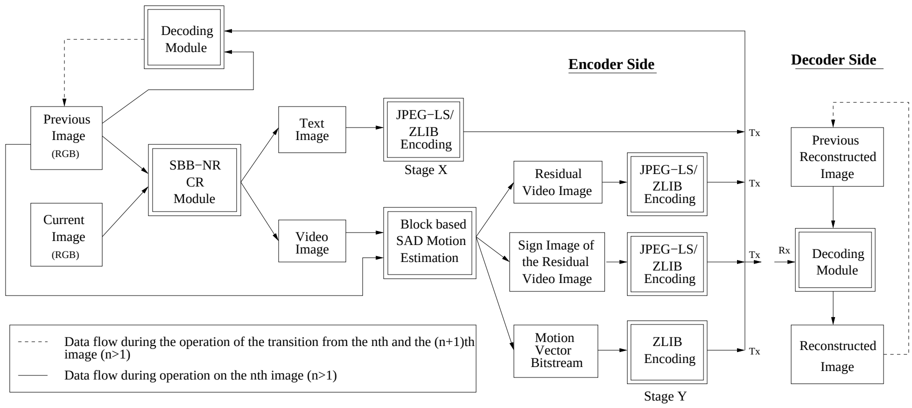


Figure 3.13: SBB-R-CR-SAD Based Motion Estimation With Sign, Block Diagram

Online Histogram Packing of SAD Residual Image With Sign

In order to see the effect of Histogram Packing on the residual image that we obtain after SAD based motion estimation, we pass it through an Online Histogram Packing step, leading thus to a scheme as shown in Figure 3.14. In order for the decoder to successfully unpack the image packed by the encoder, the encoder needs to transmit some side information as was discussed in the Histogram Packing section.

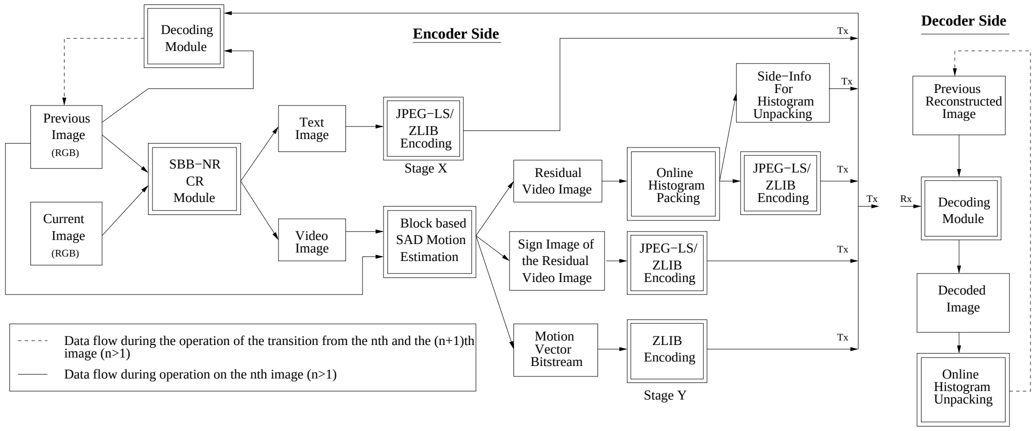


Figure 3.14: SBB-R-CR-SAD Based Motion Estimation With Sign, Histogram Packed, Block Diagram

3.6.2 Residual Without Sign

This section describes how the residual block after motion estimation can be obtained without creating a sign block. [25] mentions how the prediction residual error can be fit in N bits, where N denotes the bit depth of the image pel. This method is directly applicable in the calculation of the residual without the need for the additional sign block. Our results show that the performance of this scheme is similar to the performance of the image with sign, although both of these schemes lead to very different residual images. The idea for calculating the residual without sign can be understood by looking at Figure 3.15 The idea behind this approach is noticing the fact that when the residual is being calculated, it is the subtraction of two intensities b_i and a_i. If N is the bit depth of b_i and a_i then both b_i and a_i, are in the range [0,$2^N - 1$]. Consequently, if r_i = b_i - a_i, then the range of r_i is between $[-2^N + 1, 2^N - 1]$.To encode such a symbol directly using the natural Binary Code, it would take N+1 bits. This is exactly the spill over bit that was considered in the "sign" image. However, we can encode this r_i using just N bits because the least value that b_i can take is 0, and the thus the least value of the residual ${r_i}^{min}$ = 0 - a_i = -a_i. Also, the highest value that b_i can take is $2^N - 1$, and thus the highest value of the residual is ${r_i}^{max}$ = -a_i + $2^N - 1$. Noting this fact, we can try the following tricks:

1. We can encode the residual r_i as a positive offset from -a_i because we have seen that the residual error is never less than -a_i. As noted before, this offset will fit in N bits.

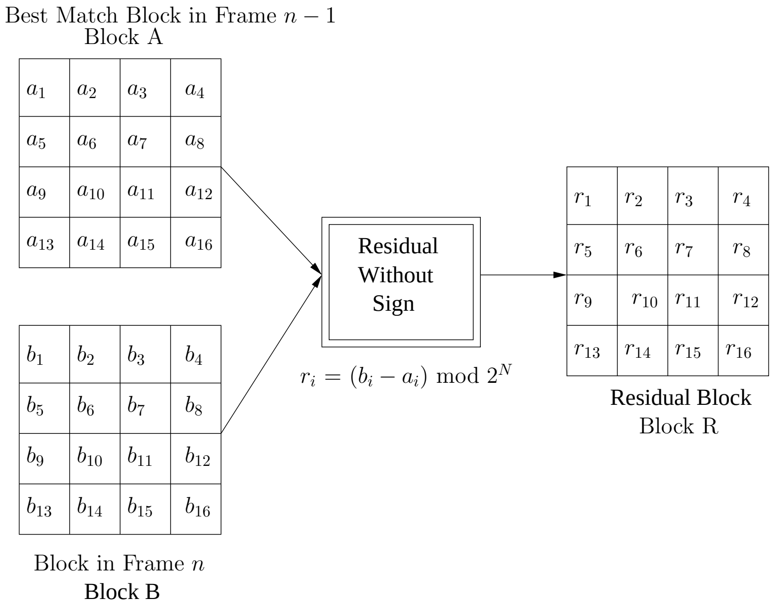


Figure 3.15: Residual Without Sign

However this means that the new residual $r\prime_i$ is given as:

$$r\prime_i = (0\text{-}(\text{-}a_i)) + r_i$$

$$r\prime_i = a_i + b_i \text{ - } a_i = b_i$$

Thus we would get a new residual value that is exactly equal to the original pel value b_i. This we already know fits in N bits, and thus we have achieved no change in variation, and thus this trick is a pointless exercise.

2. The other trick is then, to encode the error $r\prime_i$ as :

$$r\prime_i = (b_i \text{ - } a_i) \bmod 2^N$$

Thus, if the $(b_i$ - $a_i)$ is negative, then it would be stored as 2^N-$(a_i$ - $b_i)$. The upshot of doing it in such a way is that at the decoder side in order to reconstruct b_i from the residual pel $r\prime_i$ and the previous image pel a_i is

$$b_i^{recon} = (r\prime_i + a_i) \bmod 2^N$$

Thus, if we carry out this process for each block in the image, then we get the residual image and sign image with the same dimensions as the input and reference image. This process if carried out as a stage after SBB-CR leads to the scheme which we call as SBB-R-CR-SAD-WithoutSign. The sign image and the residual image obtained here, can be compressed either by JPEG-LS or by ZLIB. Figure 3.16 shows the schematic block diagram of this Conditional Replenishment process.

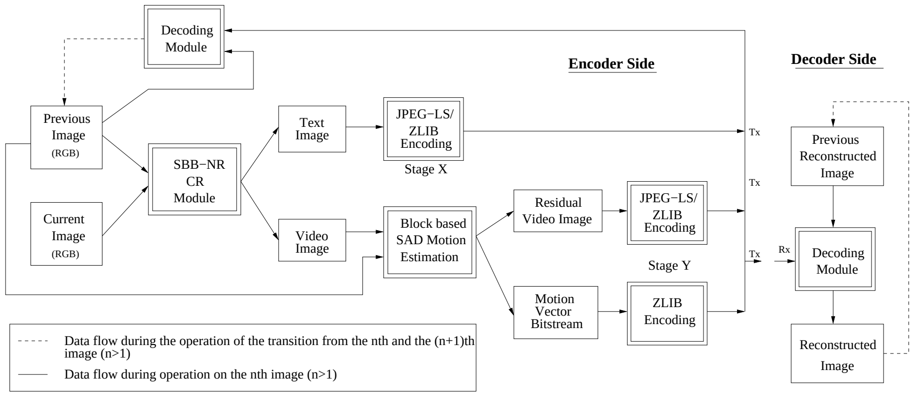


Figure 3.16: SBB-R-CR-SAD Based Motion Estimation Without Sign, Block Diagram

Online Histogram Packing of SAD Residual Image Without Sign

In order to see the effect of Histogram Packing on the residual image that we obtain after SAD based motion estimation, we pass it through an online Histogram Packing step, leading thus to a scheme as shown in Figure 3.17. In order for the decoder to successfully unpack the image packed by the encoder, the encoder needs to transmit some side information as was discussed in the Histogram Packing section.

In the results section we will describe how these schemes compare with each other and with the other schemes discussed so far.

The main disadvantage of block motion estimation is that it introduces discontinuities at the block borders (blocking artifacts). These artifacts appear in the form of sharp horizontal and vertical edges which are easily spotted by the human eye and produce ringing effects (large coefficients in high frequency sub-bands) in the Fourier-related transform used for transform coding of the residual frames. Although JPEG-LS does not work with frequency transforms, it was seen that residual image obtained after a SAD based

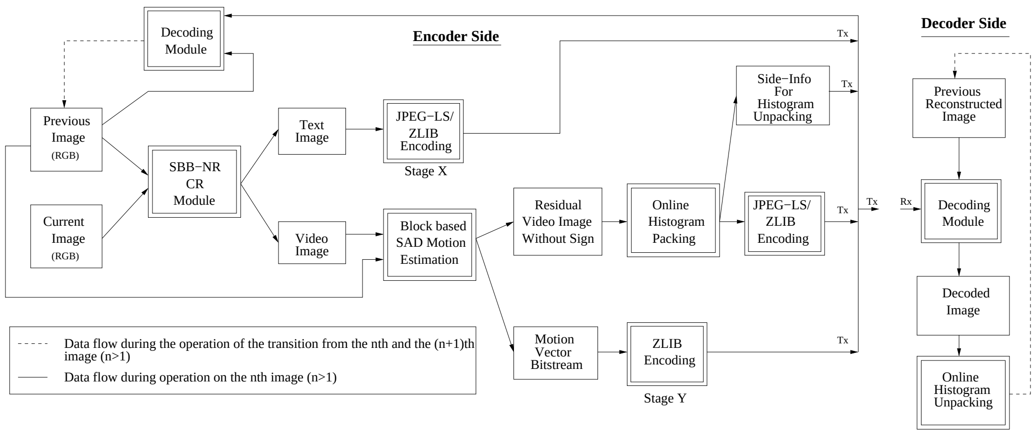


Figure 3.17: SBB-R-CR-SAD Based Motion Estimation Without Sign, Histogram Packed, Block Diagram

motion estimation was more often than not, less JPEG-LS friendly than the image that was present before SAD motion estimation. This phenomenon was also observed by Memom and Sayood in [5], and we confirm their findings that SAD based motion estimation might not directly help even in JPEG-LS/ZLIB based lossless schemes.

3.7 Residual Lossy Schemes (SBB-R-CR-Lossy) with an Intra/Key Frame

SBB-R-CR-Lossless considered above was subjected to lossless encoding once the residual was found. However if one subjects it to in-loop transforms such as a Discrete Cosine Transform, then we would obtain the MPEG-2 like codec. In this thesis we not directly consider such a scheme, but we consider the performance of H.264 which is DCT based scheme with a very similar residual approach.

H.264/MPEG-2/MPEG-4 can be thought to employ an encoding model very similar to SBB-R-CR-Lossy by reasoning as follows: It was described in the SBB-NR-CR scheme, that if for a block in the current frame, the block having the same top-left coordinates in the previous frame has exactly the same pels, then such a block would lead to a 0 residue, and would be tagged with 0s in the text frame. In H.264, if we are trying to find the best match for a block in a frame, and that best match happens to lead to zero residue, then this block is encoded with (mvX, mvY) and a SKIP block flag. If mvX and mvY are zero (meaning that the best match was the block having the same topleft coordinates, then, we can encode mvX, mvY and SKIP as 0,0,1 in the bitstream. Thus this SKIP scheme is a SBB-R scheme but one which does not operate with a ”text” image and the ”video” image. It was said in the abstract that all video encoding schemes are Conditional Replenishment based schemes, and the purpose of this paragraph was to put forth the reason for saying so. It is only a matter of being not the same ”type” of Conditional Replenishment scheme (i.e. in H.264 Conditional Replenishment is not achieved through the use of ”Text” image, but rather through use of SKIP blocks and zero motion vectors).

Another thing we should consider is that for in the SBB-NR-CR schemes, blocks that don’t match with the block exactly at the same coordinates in the previous image, were considered for SBB-R-CR Motion Estimation, just like it is done in H.264 except that in SBB-R-CR it occurs as a 2 stage process instead of a 1 stage process in H.264.

If we use frequency transform encoding methods to encode this block residual, then it is well known that this can achieve a high degree of compression, but this normally leads to error propagation, which has to be compensated with the introduction of a clean frame called ”Intra Frame” or Key Frame.

Such a scheme based on SBB-R-CR-Lossy encoding using an Intra Frame/Key Frame refresh can lead to a high quality video as well, if the encoding is near-lossless. This video encoding scheme has been left out of the schemes discussed herein, and we decided to focus on lossless SBB-R-CR schemes. We mention this as a category so that we could classify existing video encoding schemes into groups.

Chapter 4

Results

4.1 Bitrate Comparison

The results were obtained by running a program written by the author on a Windows XP machine with DirectX installed. The encoder program has the ability to encode directly by grabbing the desktop screen with whatever is playing on it, provided Hardware Acceleration is turned off. The program can also encode image files independently stored on the hard drive, and the decoder program can decode them back to the drive. This chapter will list first the results obtained for the following sequences. Each sequence is a set of RGB frames, of dimension 1280x800.

1. LGE Sequence, 1280x800x3, 90 frames
2. Forest Sequence, 1280x800x3, 90 frames
3. Highway Sequence, 1280x800x3, 80 frames
4. Highway Sequence with Sparse Histogram, 1280x800x3, 30 frames
5. Quarterscreen Foreman, with 1280x800x3, 100 frames
6. NoVideo Sequence, with 1280x800x3, 100 frames
7. DuckTales Sequence, with 1280x800x3, 100 frames

In order to see the performance of Histogram Packing, first 30 frames of the highway sequence were modified by changing its histogram through contrast stretching. The

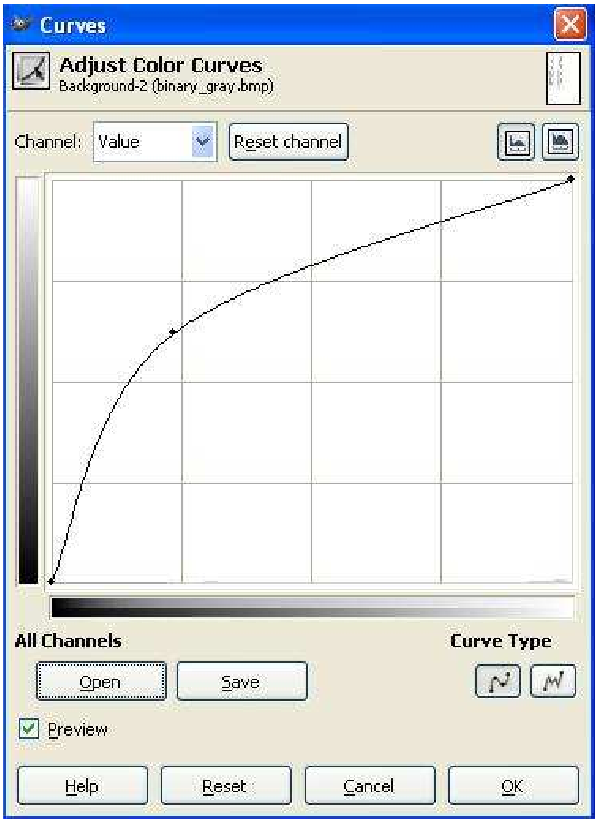


Figure 4.1: Gamma Compression Curve used for Creating Sparse Histograms

contrast stretching was performed in GIMP using a curve as shown in Figure 4.1. A function of the sort $f(x_i) = \text{round}((M^{(1-\gamma)}) * (x_i^{\gamma}))$, where x_i is the image intensity value, and $[0,\ M]$ is the range of x_i, and $\gamma < 1$, leads to a curve as shown in Figure 4.1 and is called as the gamma compression curve. Applying such a function to the image leads to a sparse histogram because it can be seen that the smaller values of x_i are mapped to higher values, and higher values remain more or less unchanged. This leads to an image histogram with majority of the bins in the higher intensities, and thus makes the histogram sparse, as well as more amenable to Histogram Packing.

In the results that follow, the bitrate is calculated as follows:

1. Compression Rate = $\frac{Image\ Size\ After\ Compression\ (in\ unit\ U)}{Image\ Size\ before\ compression\ (in\ unit\ U)}$

2. Bits per Pixel (bpp) β = compression Rate * δ bits/pixel, δ = 24 bits for RGB input images.

3. Bitrate@α frames per second (fps) = (Image Height in pixels * Image Width in pixels * β * α fps) bits per second(bps).

4.1.1 Discussion of the Bitrate Results

In this section, the bitrate results that are obtained for every sequence are plotted using the standard deviation errorbars, with 95% confidence intervals. This means that the standard deviation error shown in the diagram is 95% of the standard deviation error for the data under consideration. Further it should be noted that for the schemes such as Lagarith, Alparysoft, TechSmith, Adobe Flash Video Encoder and MSU we did not get the exact bitrate per frame, and measured only the final file size after the entire encoding. As a consequence, these codecs have been marked only by their average bitrate and would be seen without any errorbars.

It is seen that for the LGE, Forest and Highway Sparse sequences, which carry natural image data, the SBB-NR-CR-JPEG-LS performs nearly as well as H.264 FRExt with QP=1. It can be seen for these sequences that SBB-NR-CR-JPEG-LS performs better than Lagarith, Adobe FLV, TechSmith, AlparySoft and all other schemes based on residual Conditional Replenishment such as SBB-R-LPME-ZLIB, and SAD based residual schemes. Note however that for the Highway sequence, Lagarith and AlparySoft perform significantly better than SBB-NR-CR-JPEG-LS. What can be said by looking at the results is that if the video sequence carries more natural image data, then the codec based on JPEG-LS performs better. Also, looking at the results for sequences such as NoVideo and QuarterScreen Foreman, we can realize that for those kinds of image sequences with less natural image content, codecs based on dictionary compression such as ZLIB perform much better than the codecs based on JPEG-LS.

For the DuckTales sequence, which is an animation sequence, it can be seen that SBB-NR-CR-JPEG-LS performs better than Lagarith, but not as good as H.264. This good performance of the SBB-NR-CR-JPEG-LS can be attributed to the ability of JPEG-LS to encode smooth regions efficiently using the run-mode.

It is also seen that for sparse histogram image sequences, such as the Highway Sparse Histogram Sequence here, the codec based on Histogram Packing, SBB-NR-CR-Histogram Packed-JPEG-LS works the best amongst the codecs tested, and that is much better than SBB-NR-CR-JPEG-LS.

It can also be noted looking at the figures presented in this section, that schemes based on SAD based motion estimation fare quite poorly compared to the Non-Residual schemes. This effect has also been observed by Memom and Sayood in [5]. We also see that SBB-R-CR-WithSign and SBB-R-CR-WithoutSign are comparable in terms of the bitrate achieved. It is also seen that Histogram Packing although has influence on the performance of SAD based schemes, it does not improve the bitrate to such an extent so as to be competitive with the Non-Residual based schemes.

Further, it can be noted thus that we could achieve a bitrate gain over uncompressed video ranging from 3 to 15 depending on the type of image sequence.

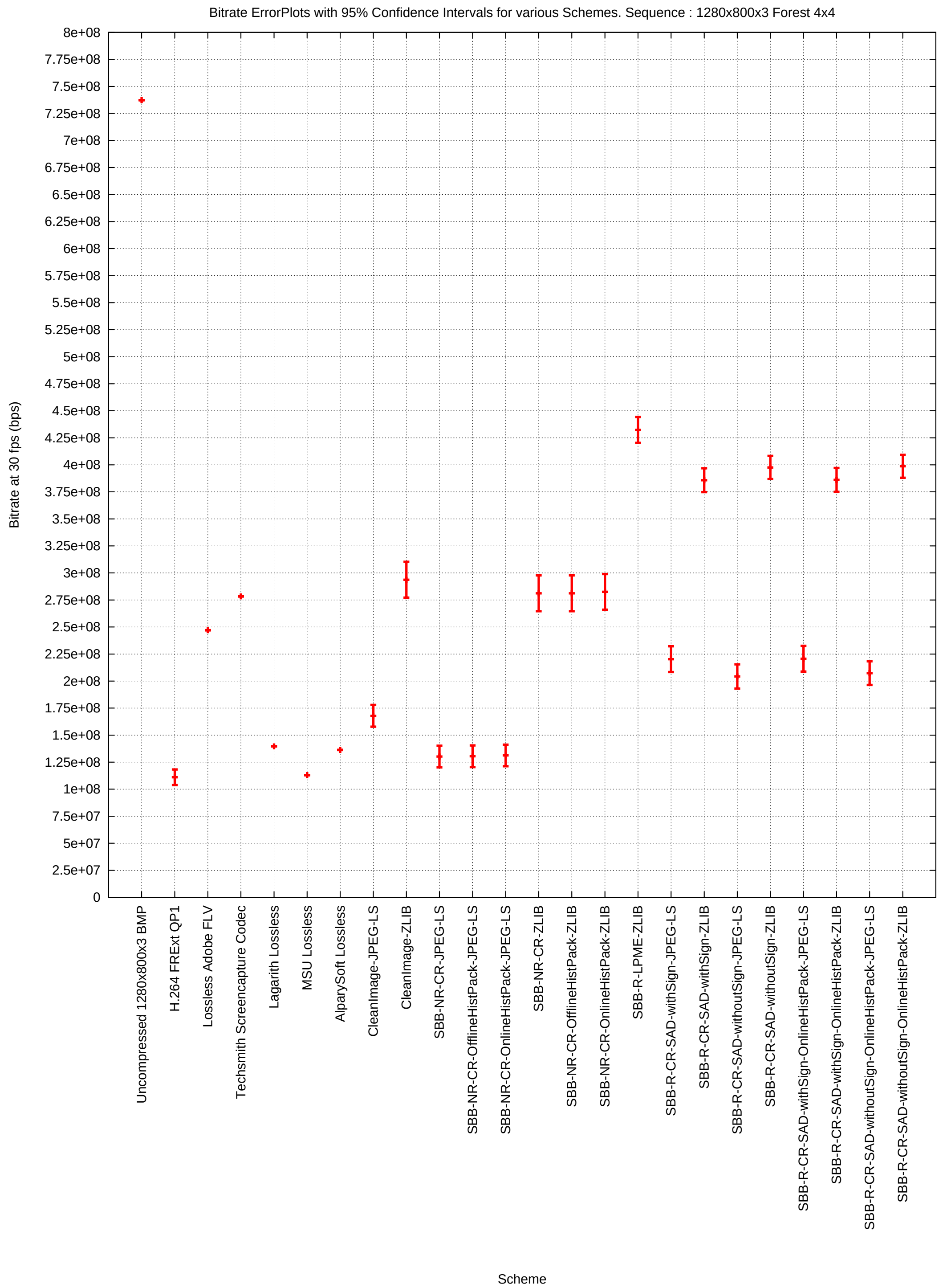


Figure 4.2: Bitrate@30 fps For Forest Sequence with 4x4 SBB-CR

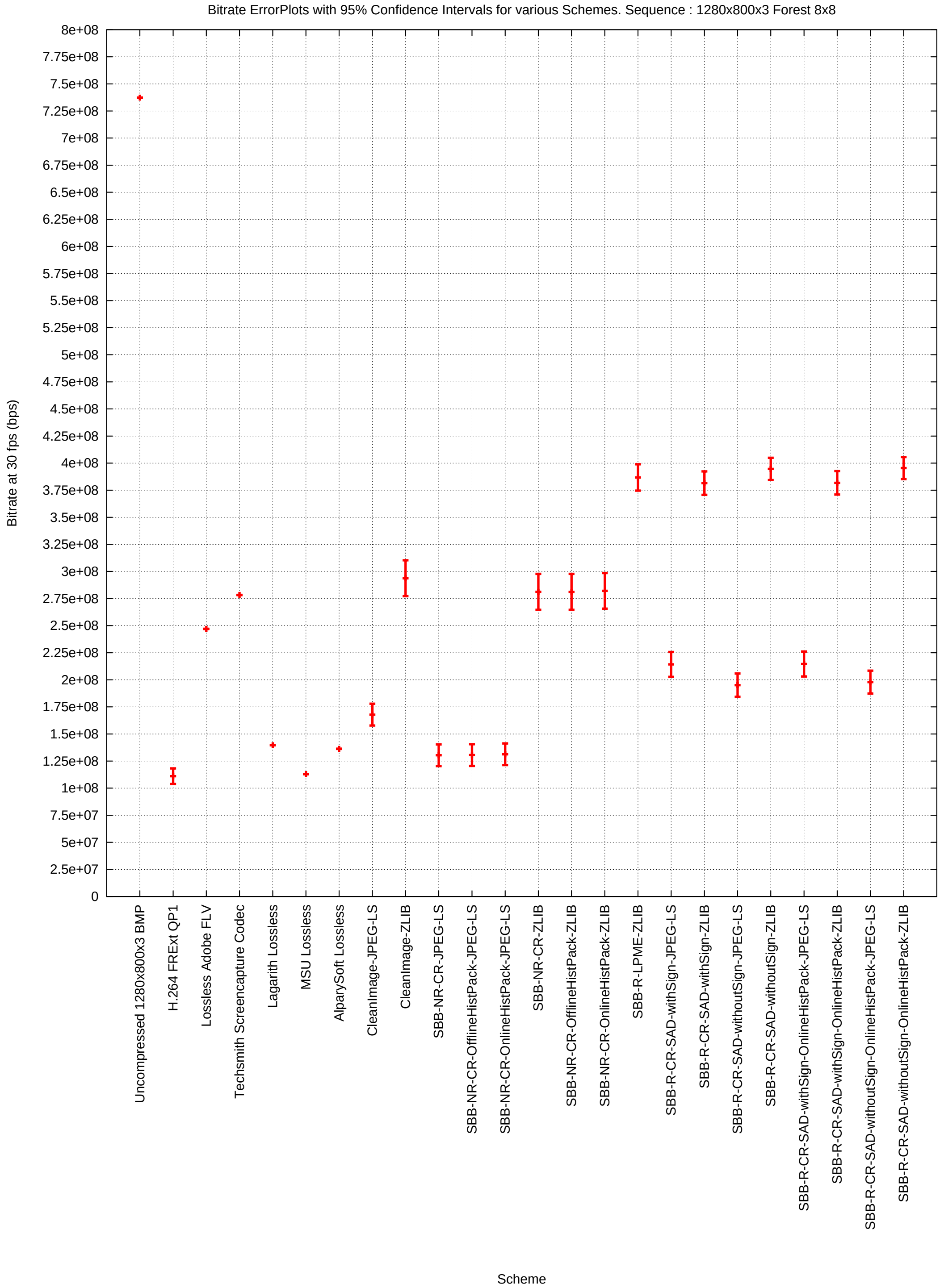


Figure 4.3: Bitrate@30 fps For Forest Sequence with 8x8 SBB-CR

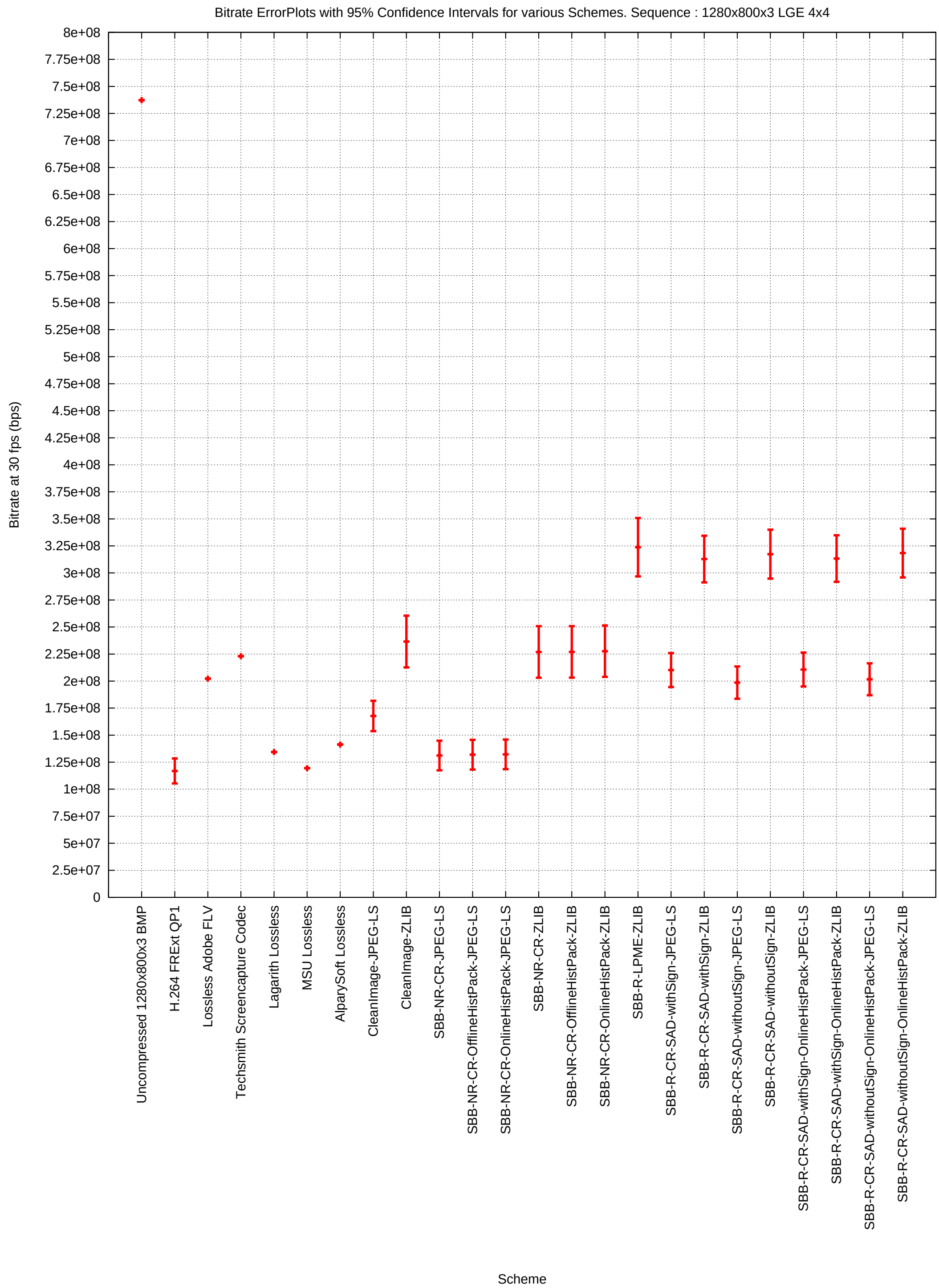


Figure 4.4: Bitrate@30 fps For LGE Sequence with 4x4 SBB-CR

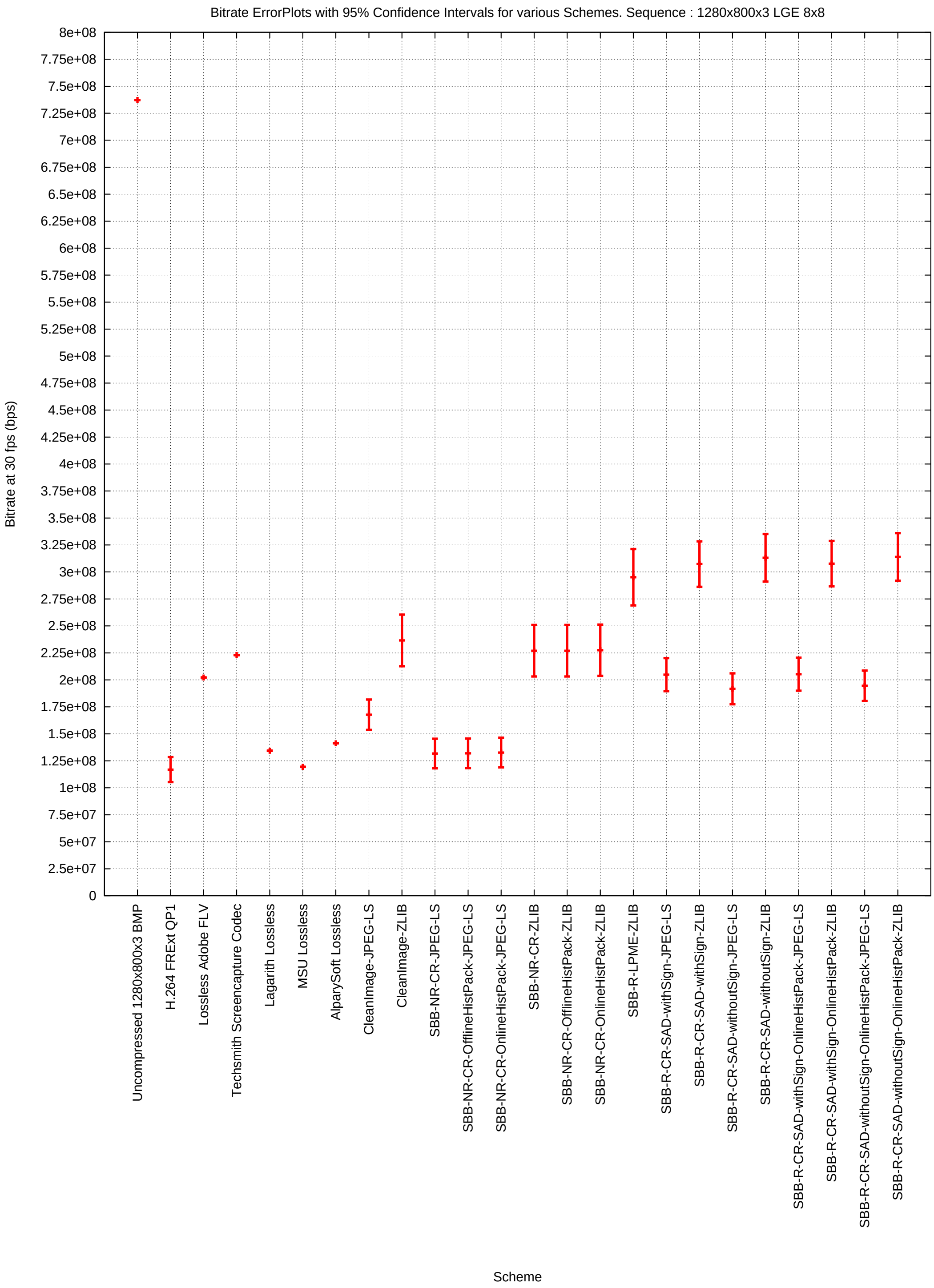


Figure 4.5: Bitrate@30 fps For LGE Sequence with 8x8 SBB-CR

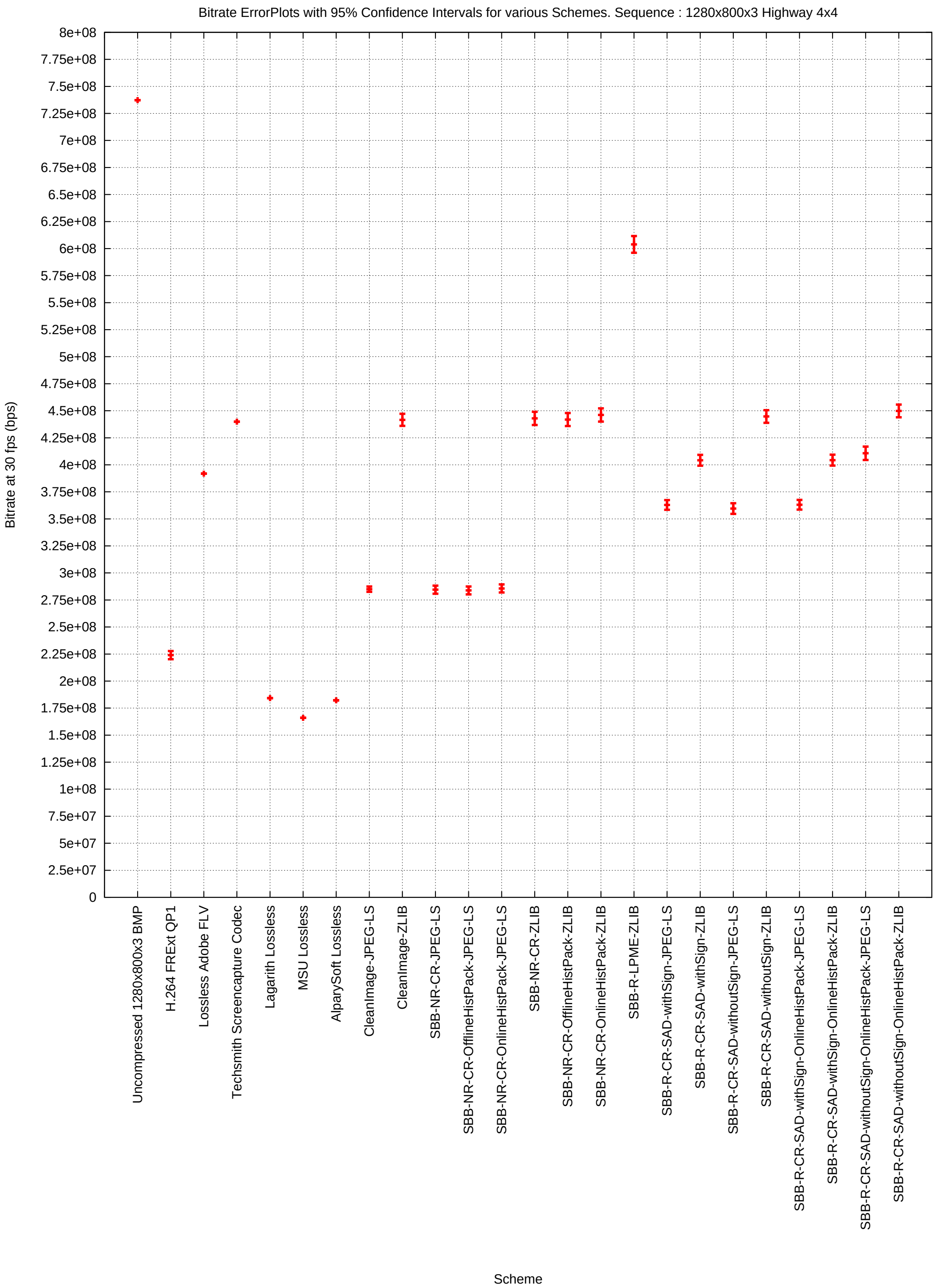


Figure 4.6: Bitrate@30 fps For Highway Sequence with 4x4 SBB-CR

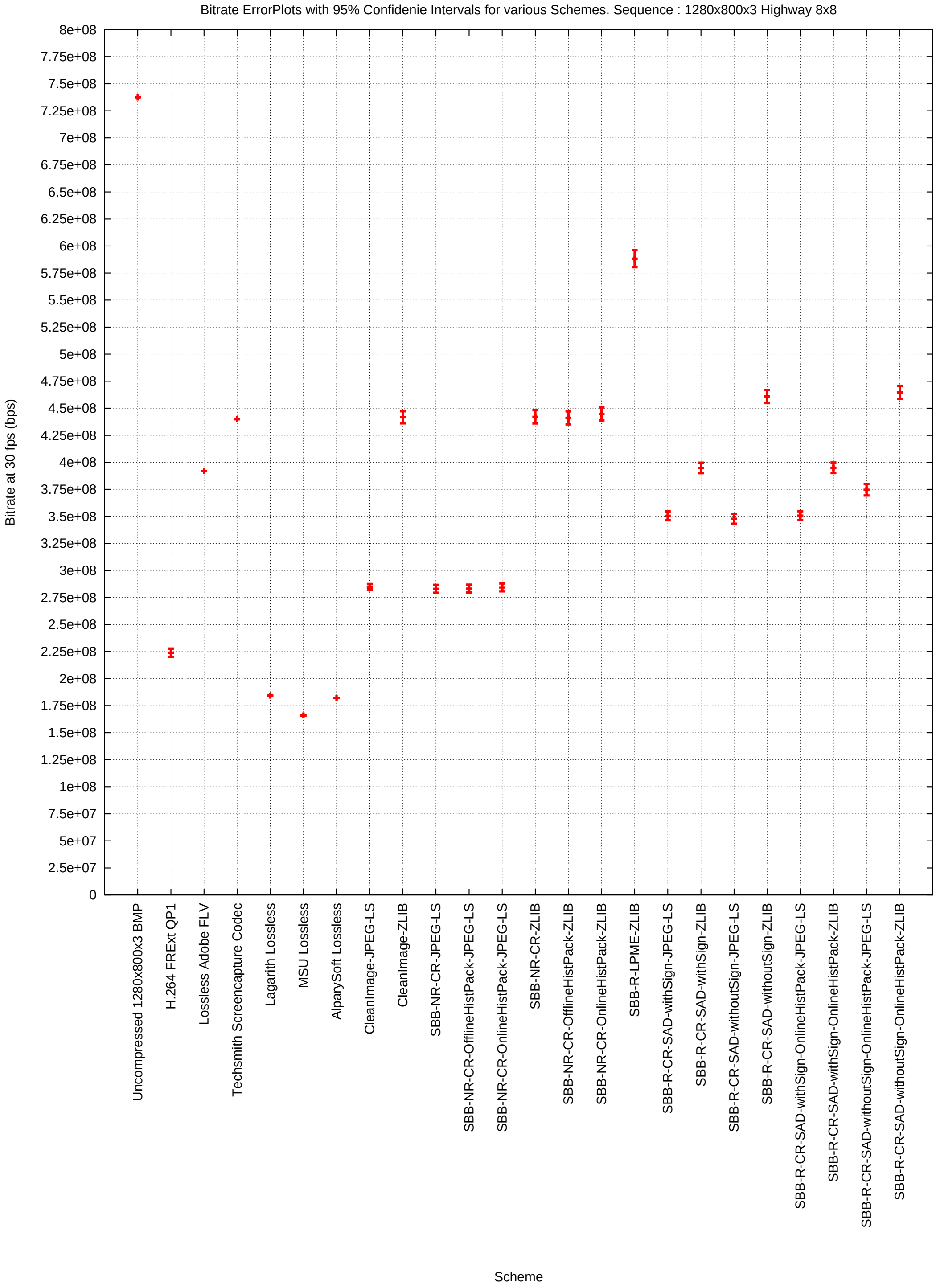


Figure 4.7: Bitrate@30 fps For Highway Sequence with 8x8 SBB-CR

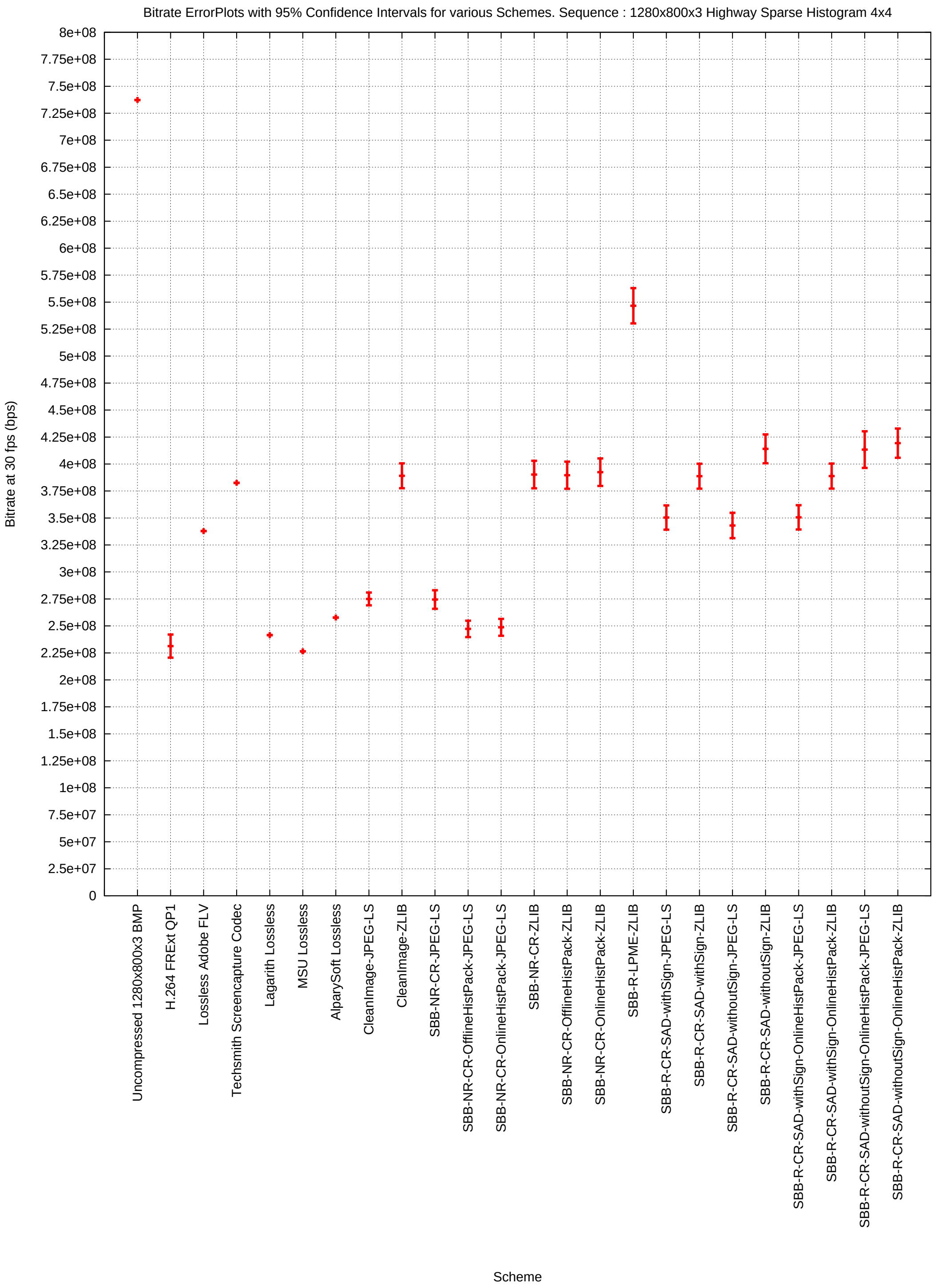


Figure 4.8: Bitrate@30 fps For Highway Sparse Sequence with 4x4 SBB-CR

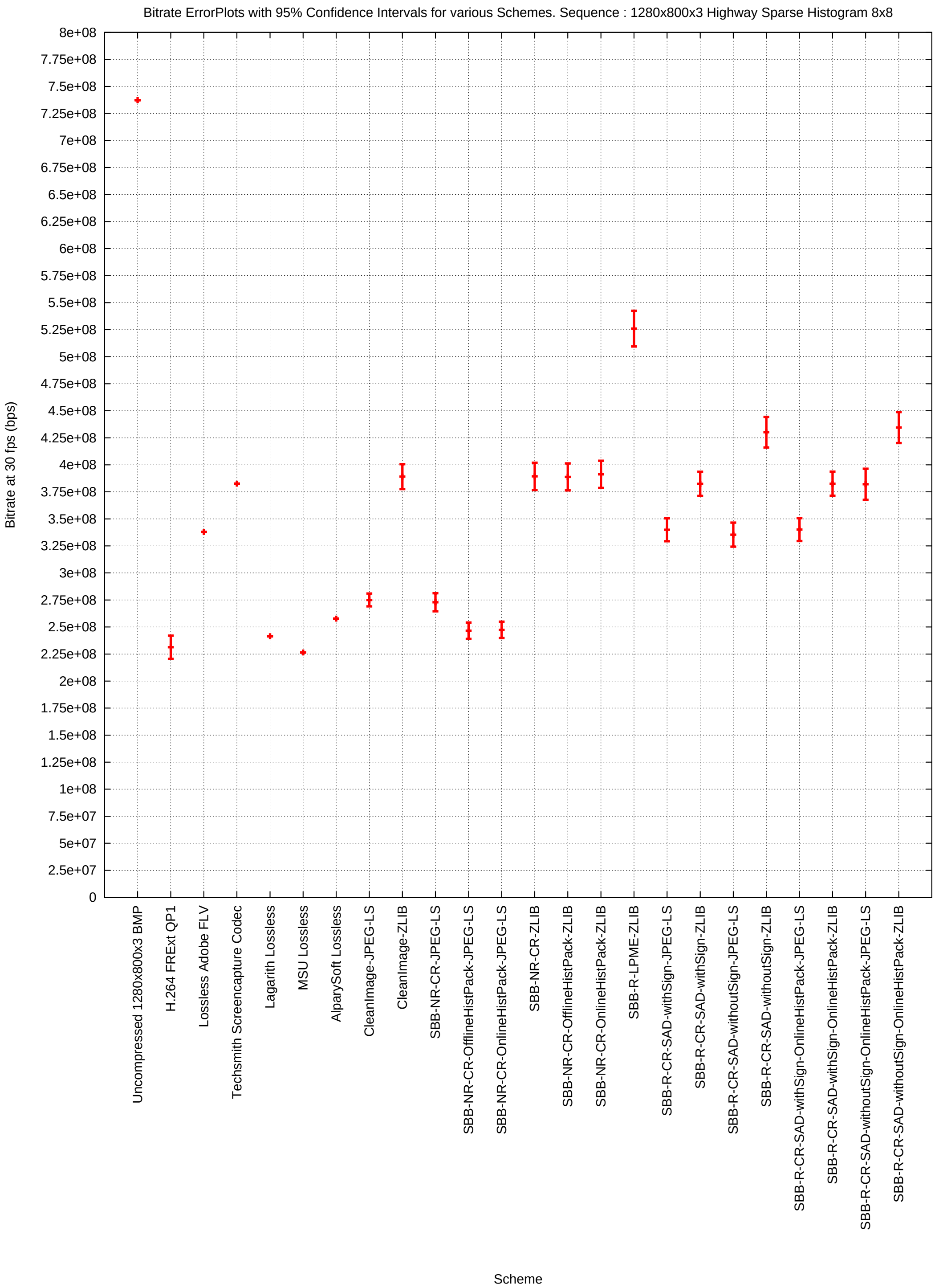


Figure 4.9: Bitrate@30 fps For Highway Sparse Sequence with 8x8 SBB-CR

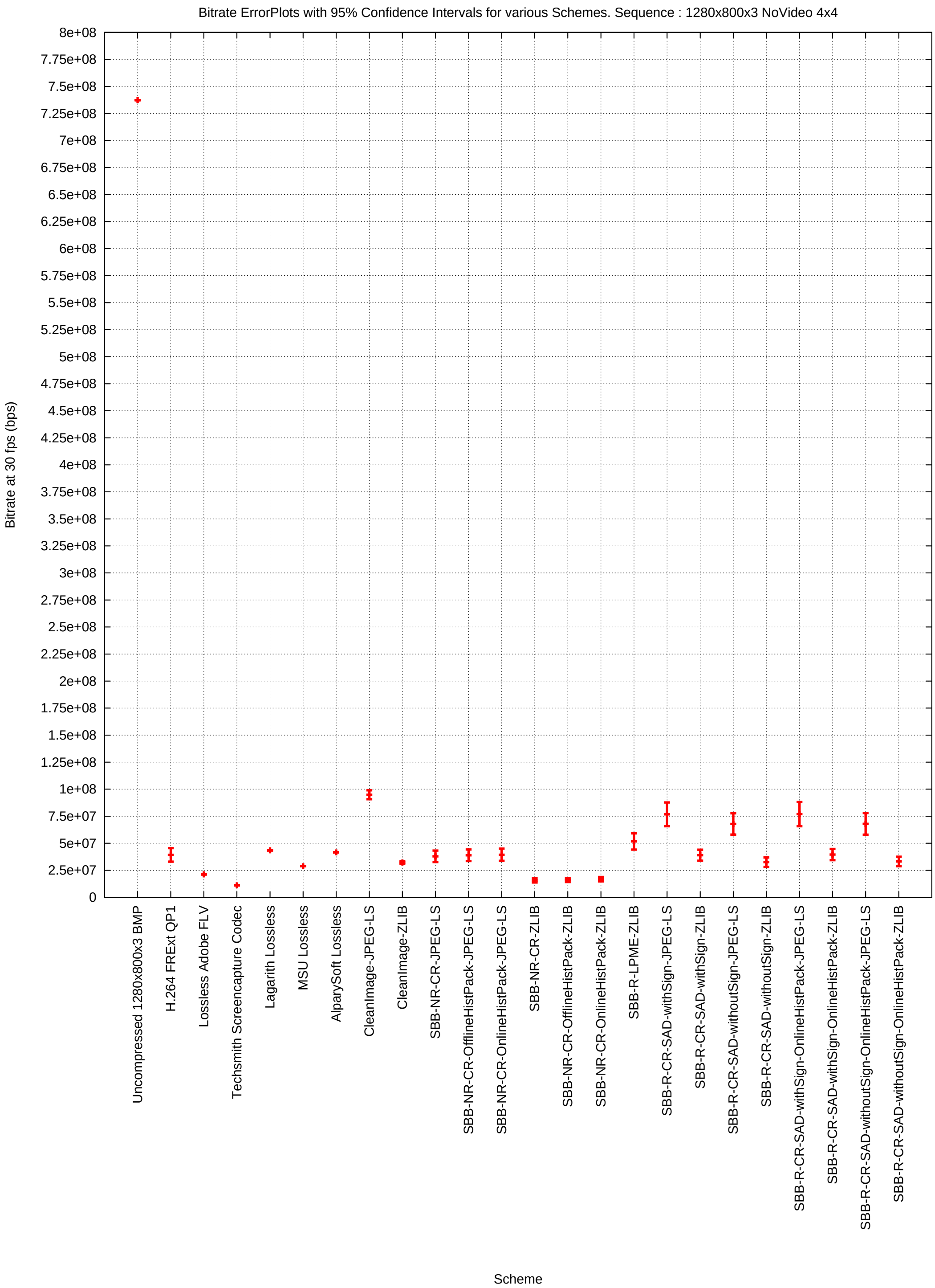


Figure 4.10: Bitrate@30 fps For NoVideo Sequence with 4x4 SBB-CR

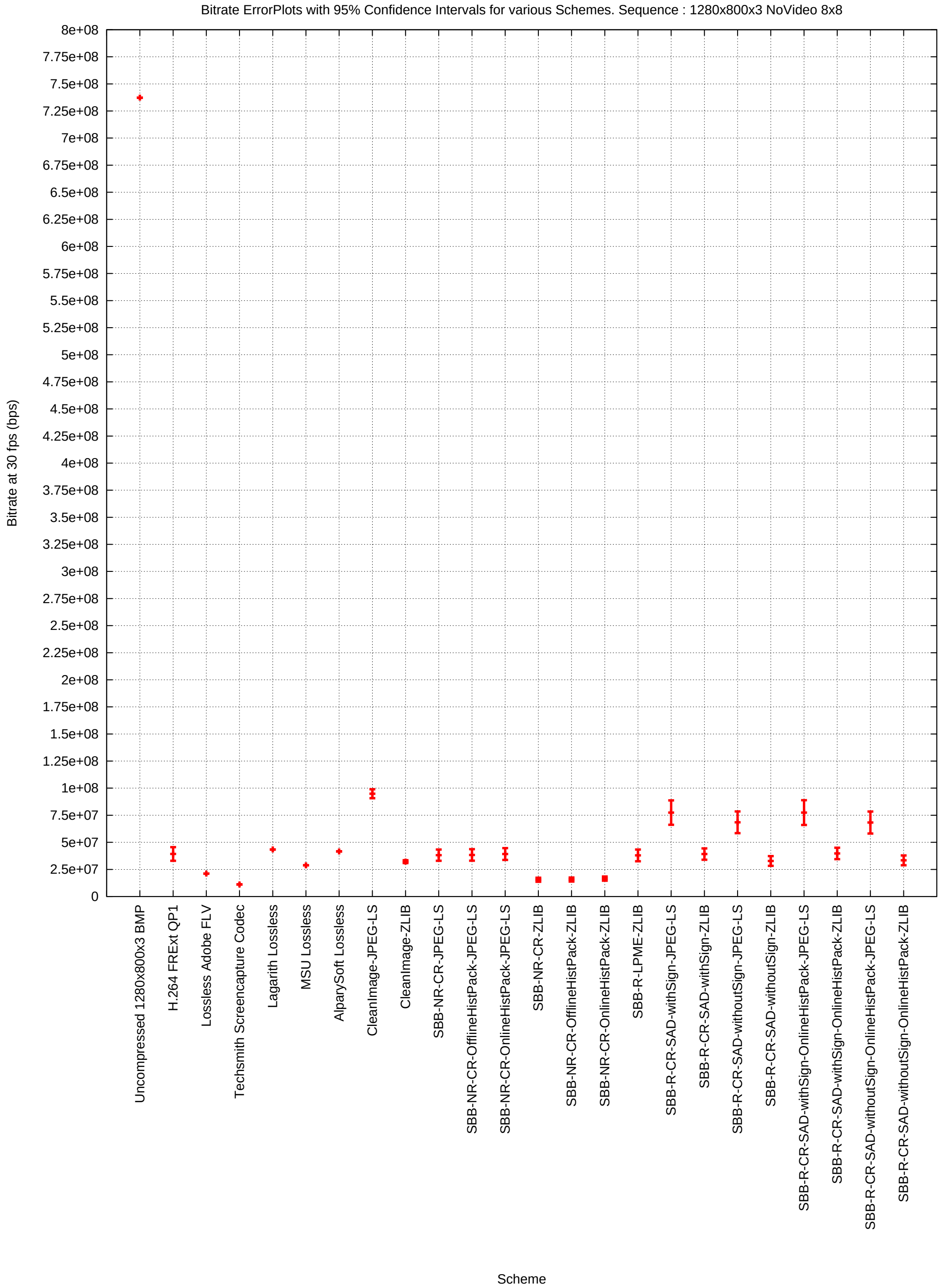


Figure 4.11: Bitrate@30 fps For NoVideo Sequence with 8x8 SBB-CR

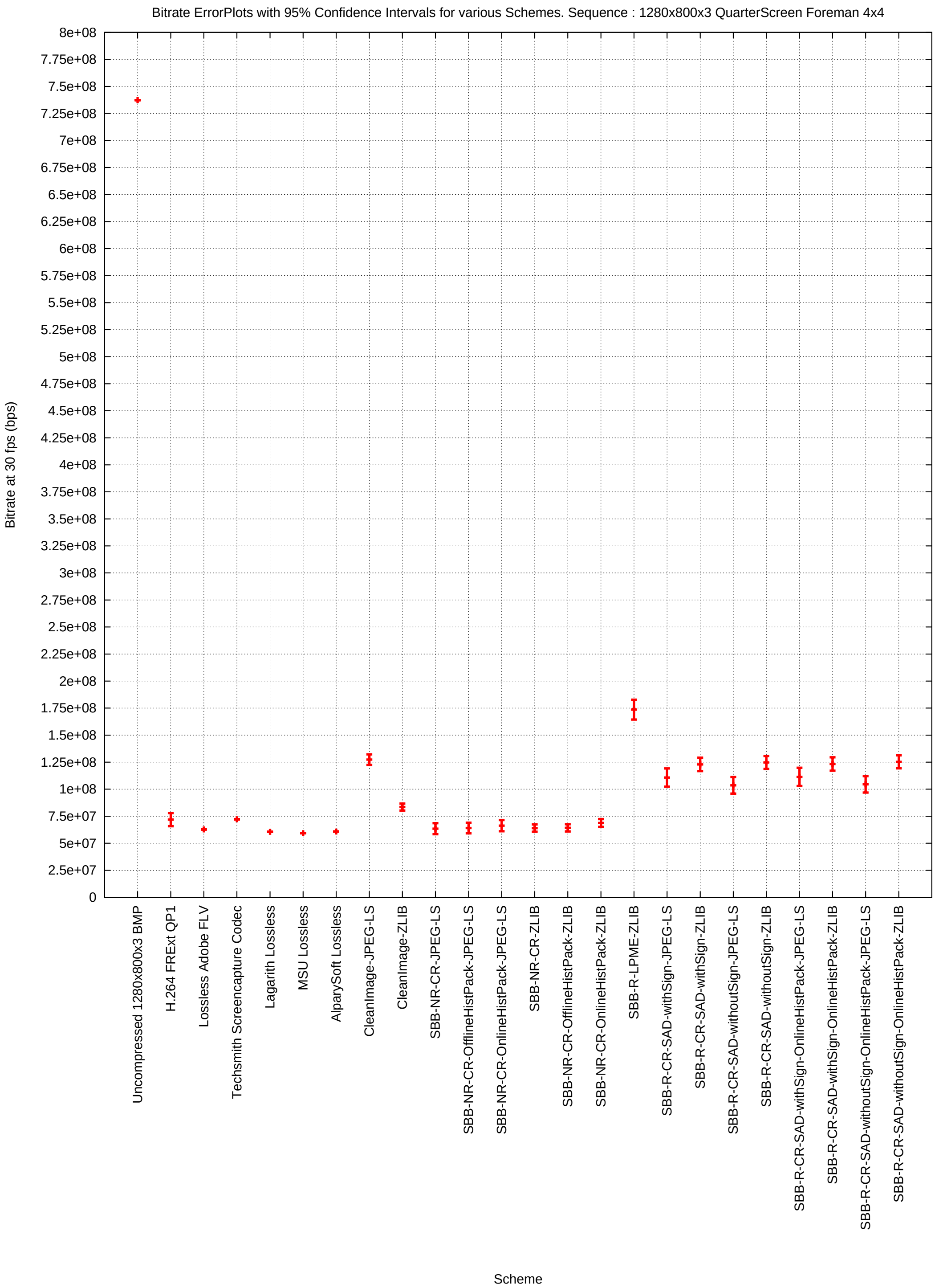


Figure 4.12: Bitrate@30 fps For QuarterScreenForeman Sequence with 4x4 SBB-CR

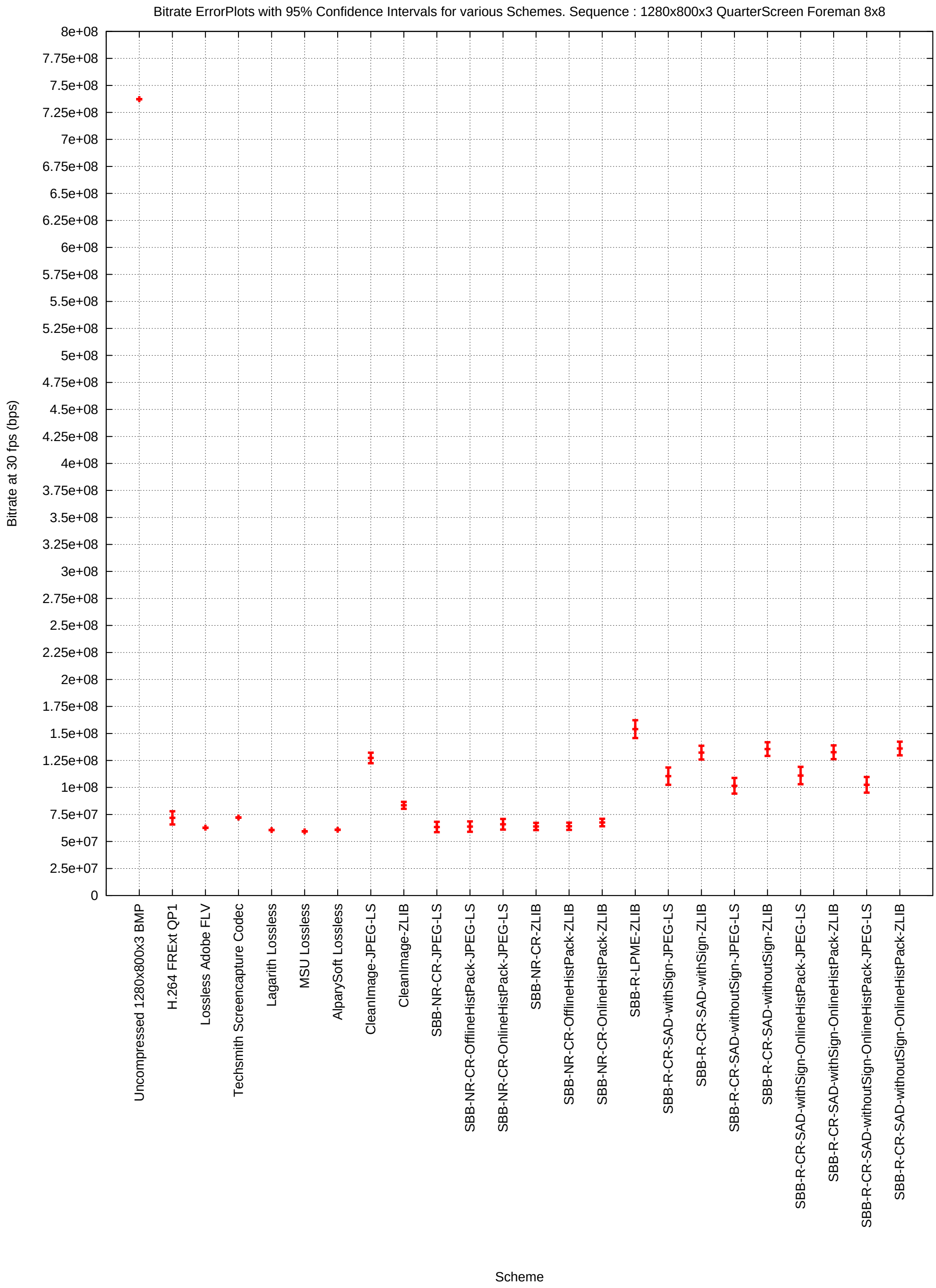


Figure 4.13: Bitrate@30 fps For QuarterScreenForeman Sequence with 8x8 SBB-CR

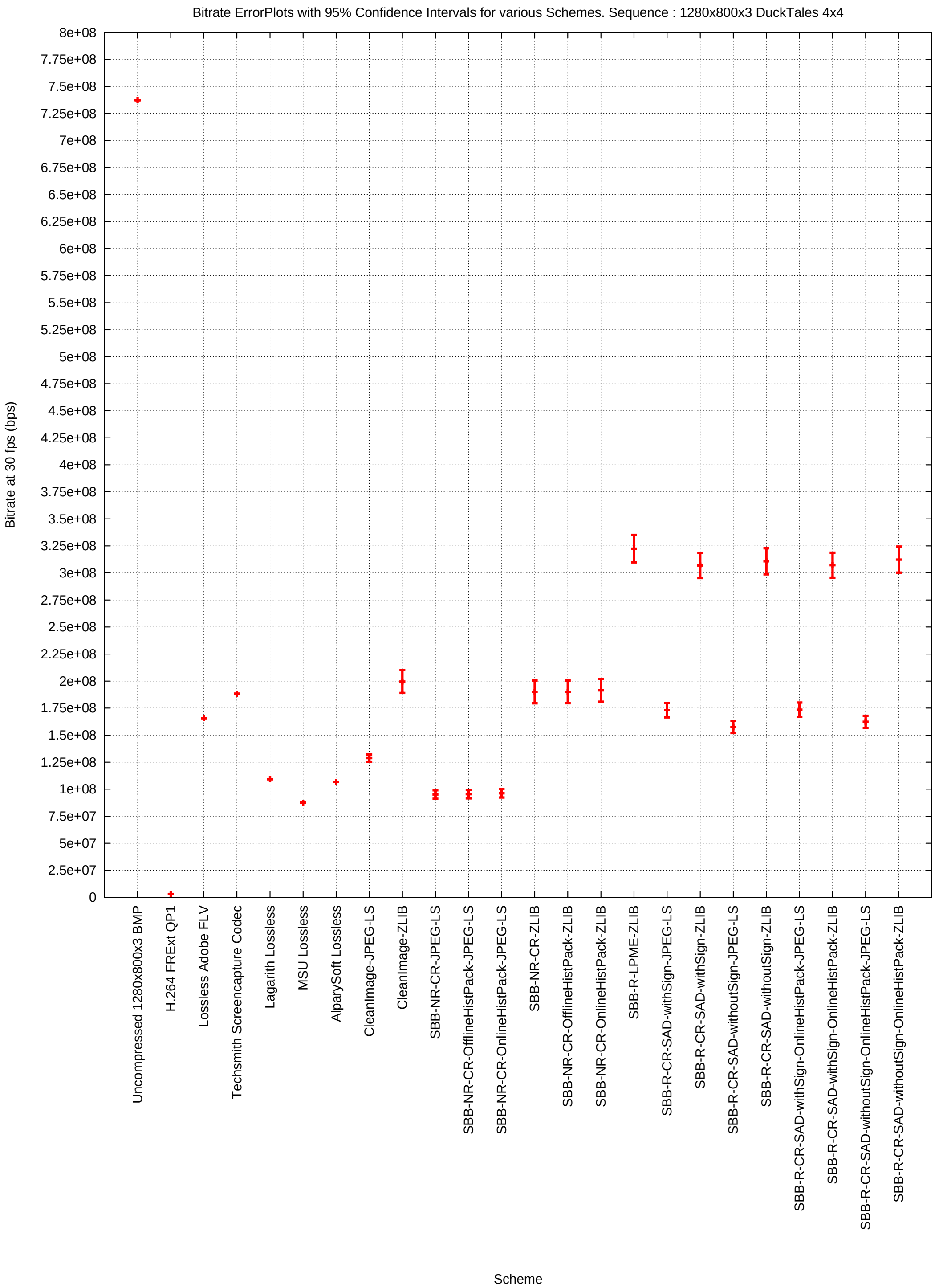


Figure 4.14: Bitrate@30 fps For DuckTales Sequence with 4x4 SBB-CR

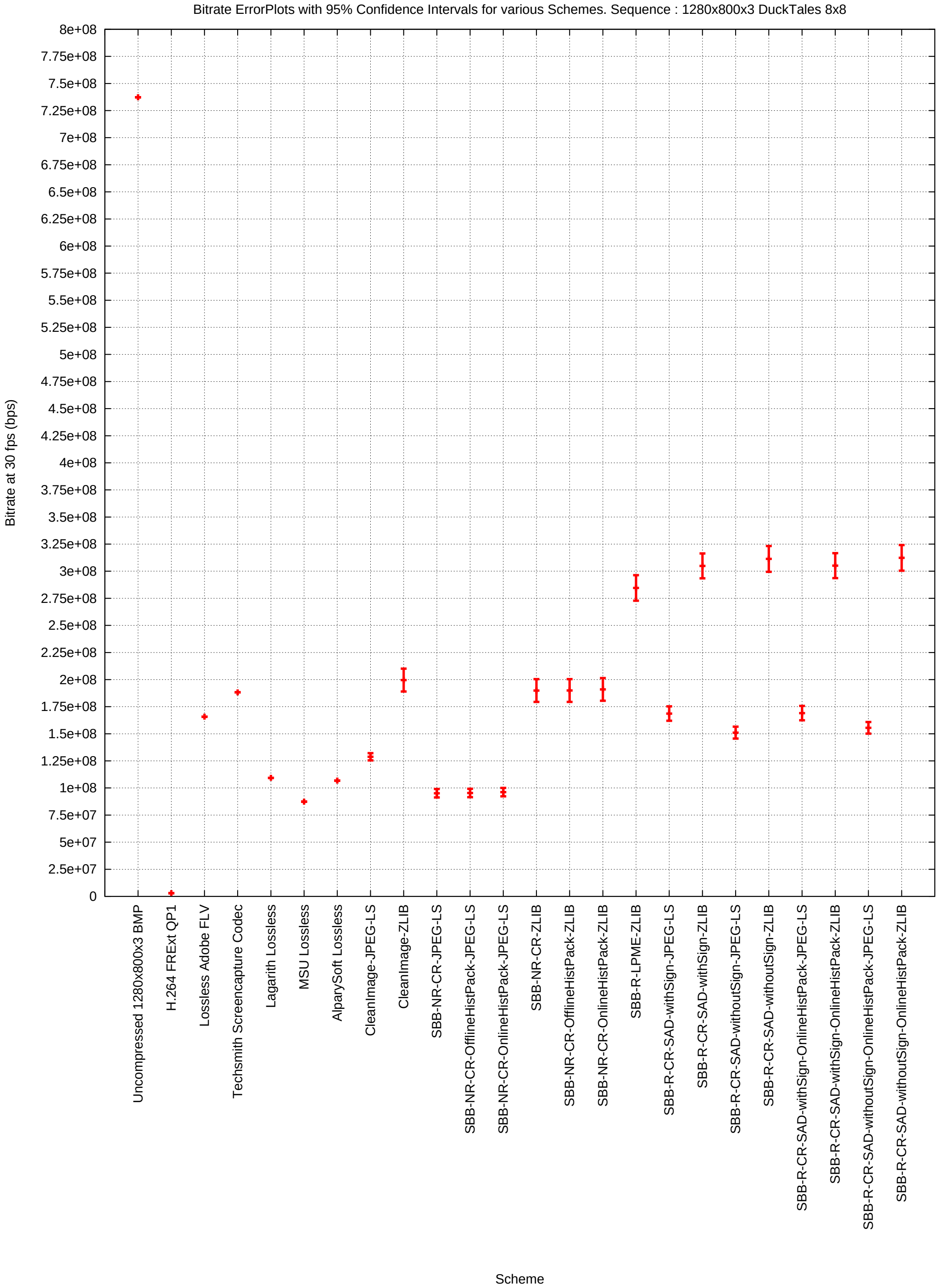


Figure 4.15: Bitrate@30 fps For DuckTales Sequence with 8x8 SBB-CR

4.2 Time Evaluation

4.2.1 Discussion of the Time Results

In this section, the encoding time results per frame, that are obtained for every sequence, are plotted using the standard deviation errorbars, with 95% confidence intervals. This means that the standard deviation error shown in the diagram is 95% of the standard deviation error for the data under consideration. Further it should be noted that for the schemes such as Lagarith, Alparysoft, TechSmith, Adobe Flash Video Encoder and MSU we did not get the exact encoding time per frame, and measured only the final time it took for the entire encoding process. As a consequence, these codecs have been marked only by their average encoding time and would be seen without any errorbars.

It can be seen from the Figures in this section, that H.264 FRExt with QP=1, takes the longest time for encoding each image, followed by the MSU codec. The Lagarith lossless codec consistently outperforms SBB-NR-CR-HistPacked-JPEG-LS in terms of encoding time, but does not always give better bitrates than SBB-NR-CR-HistPacked-JPEG-LS, as was seen in Forest, LGE and DuckTales sequences. In the next section we will compare all of these codecs with their bitrates and encoding times together.

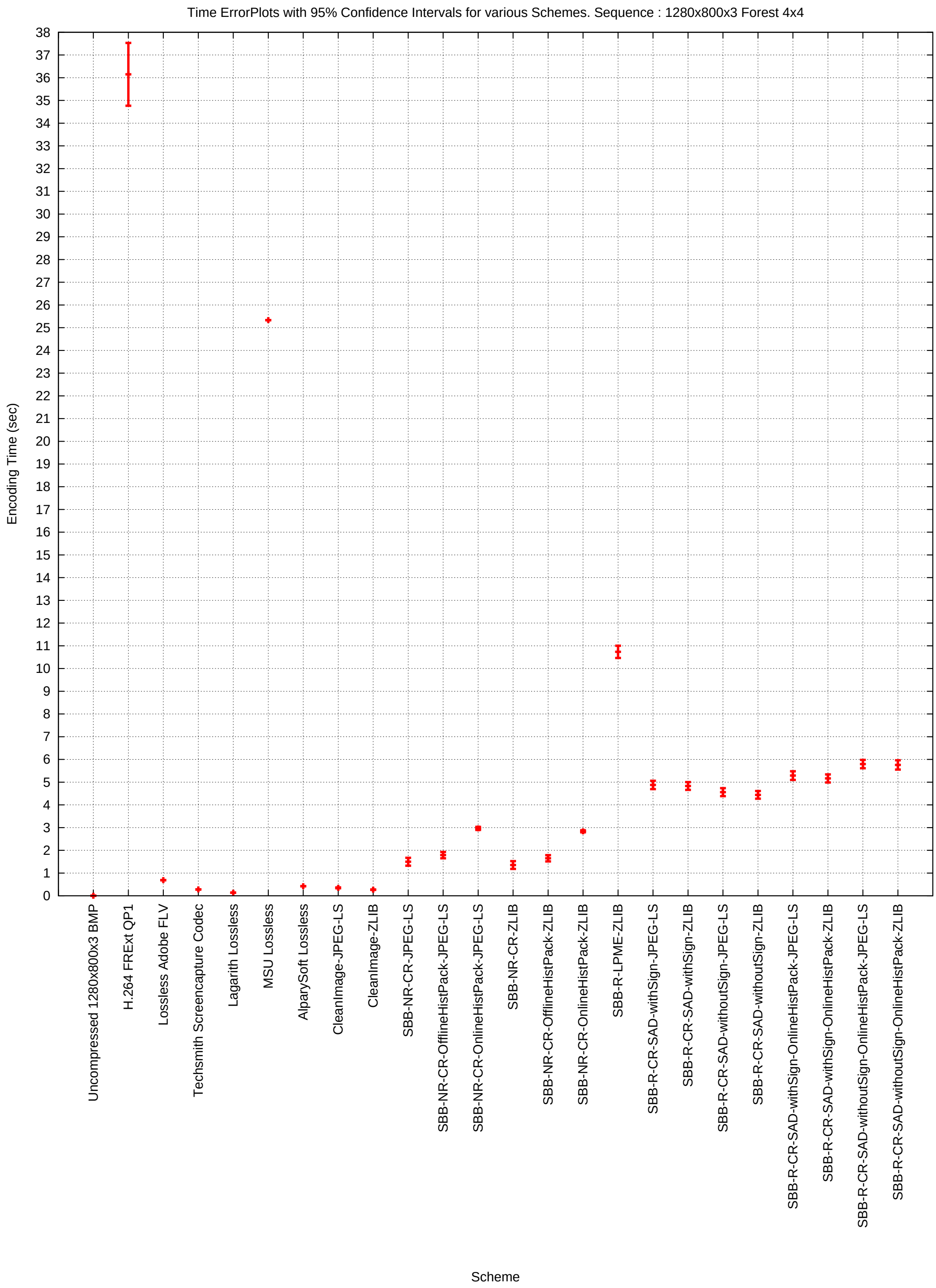


Figure 4.16: Encoding Time For Forest Sequence with 4x4 SBB-CR

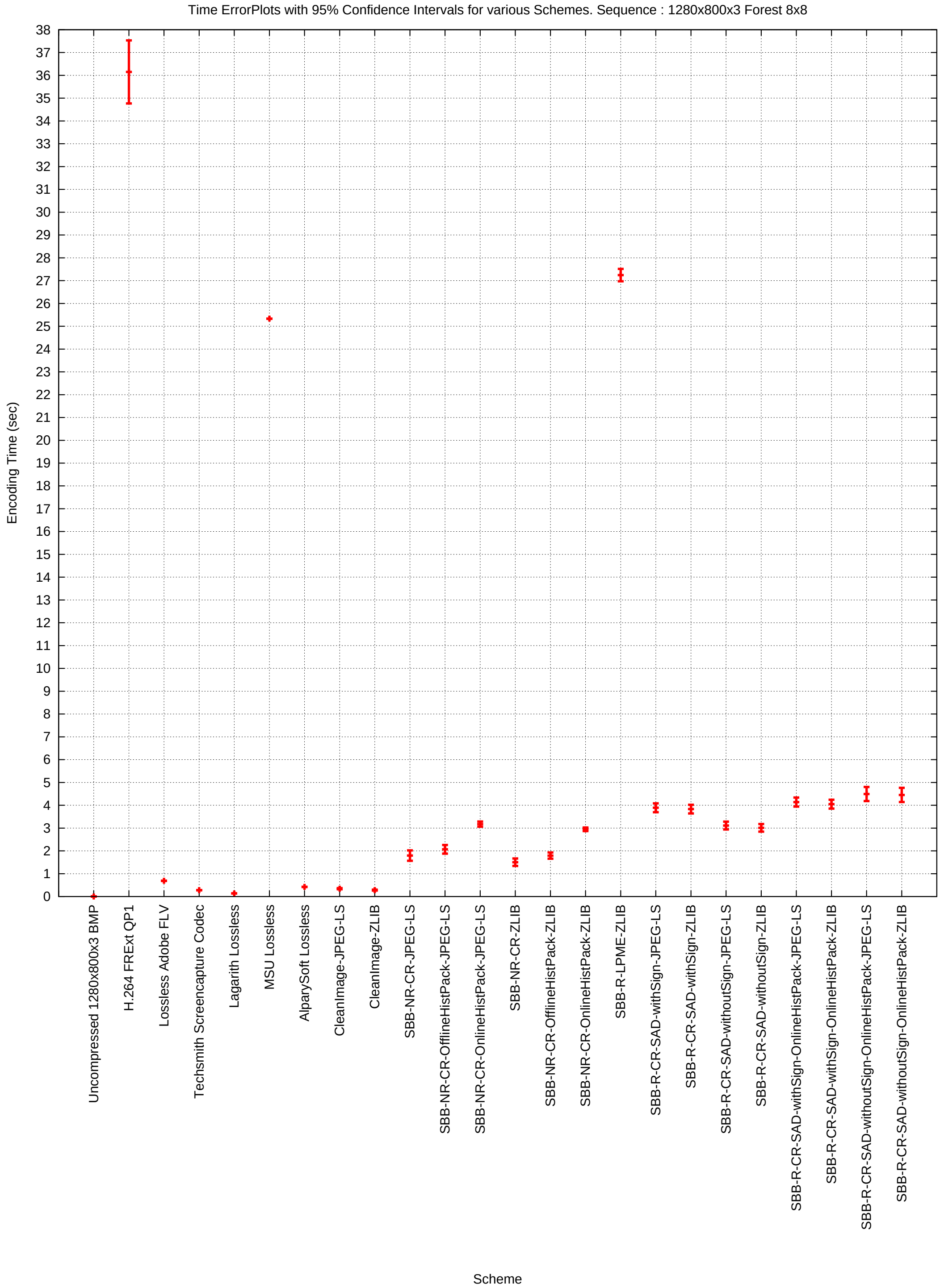


Figure 4.17: Encoding Time For Forest Sequence with 8x8 SBB-CR

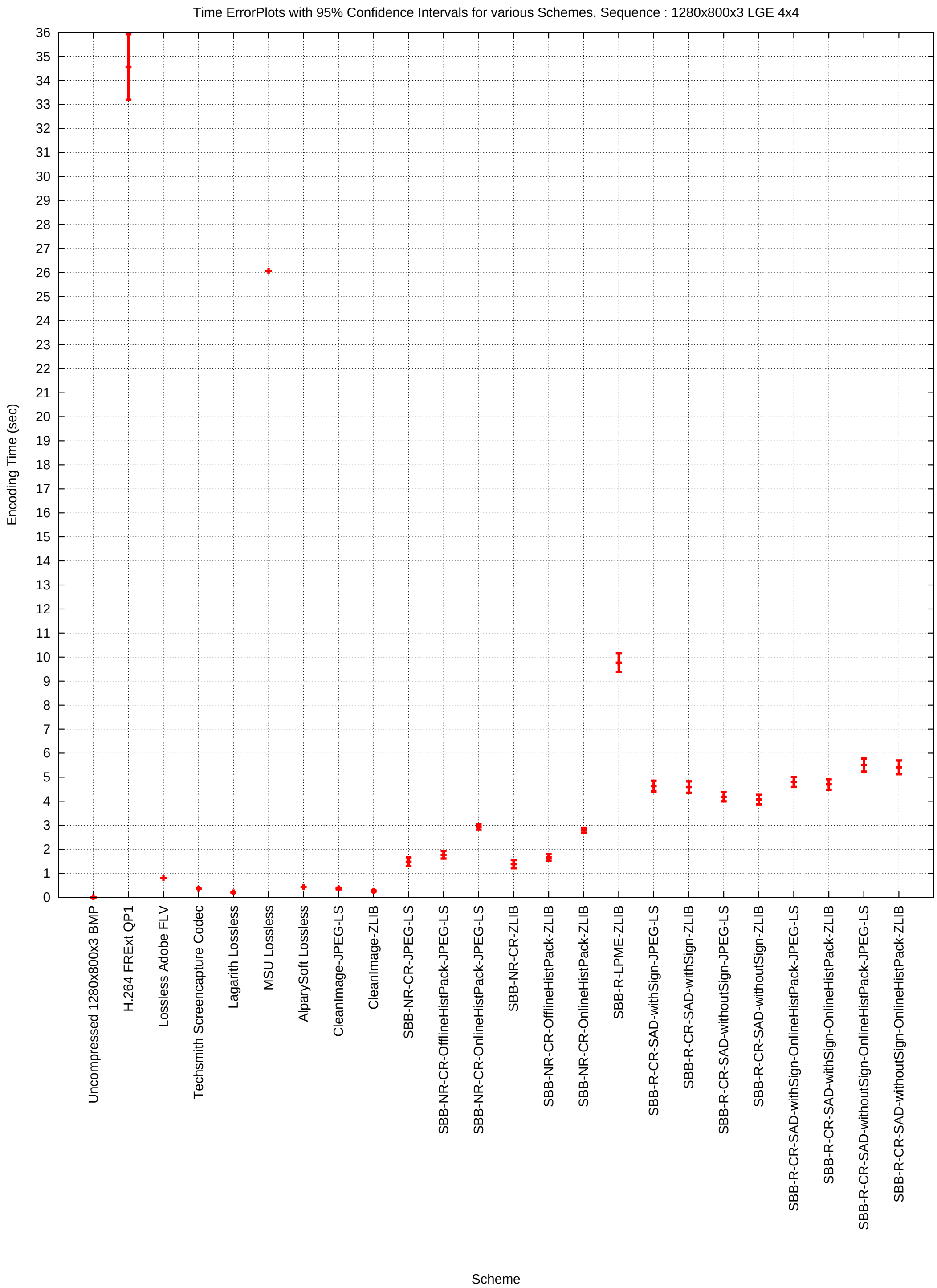


Figure 4.18: Encoding Time For LGE Sequence with 4x4 SBB-CR

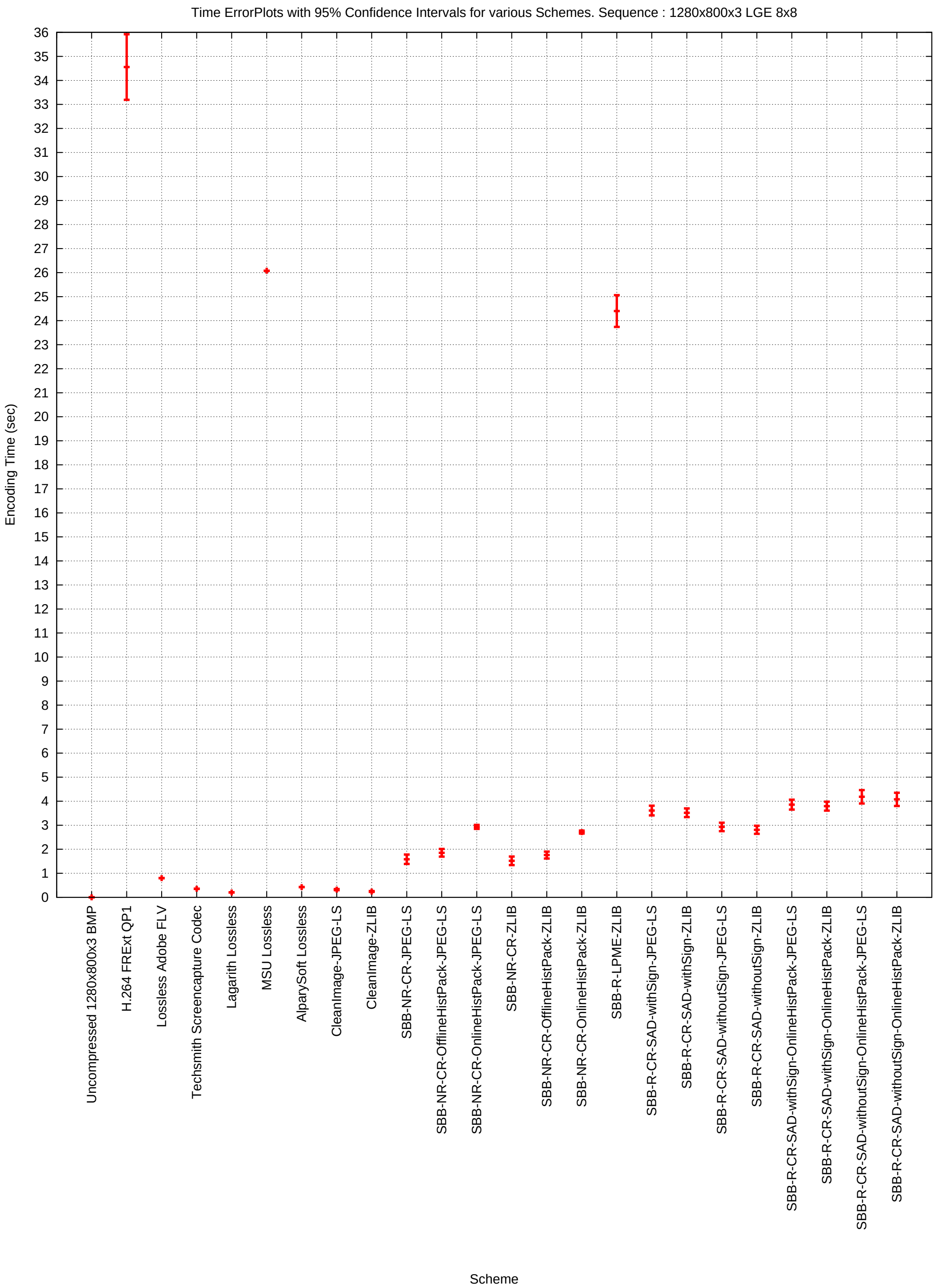


Figure 4.19: Encoding Time For LGE Sequence with 8x8 SBB-CR

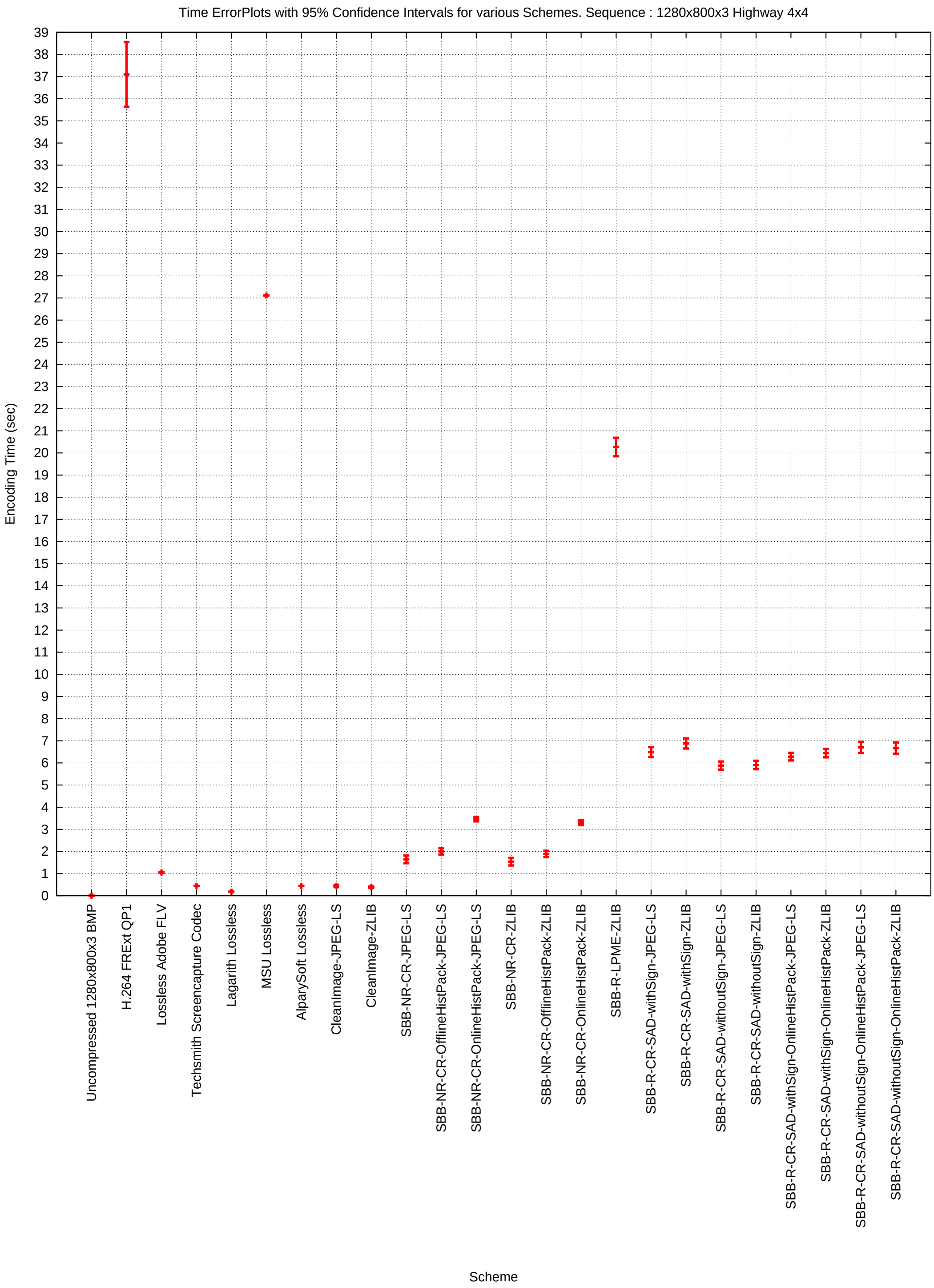


Figure 4.20: Encoding Time For Highway Sequence with 4x4 SBB-CR

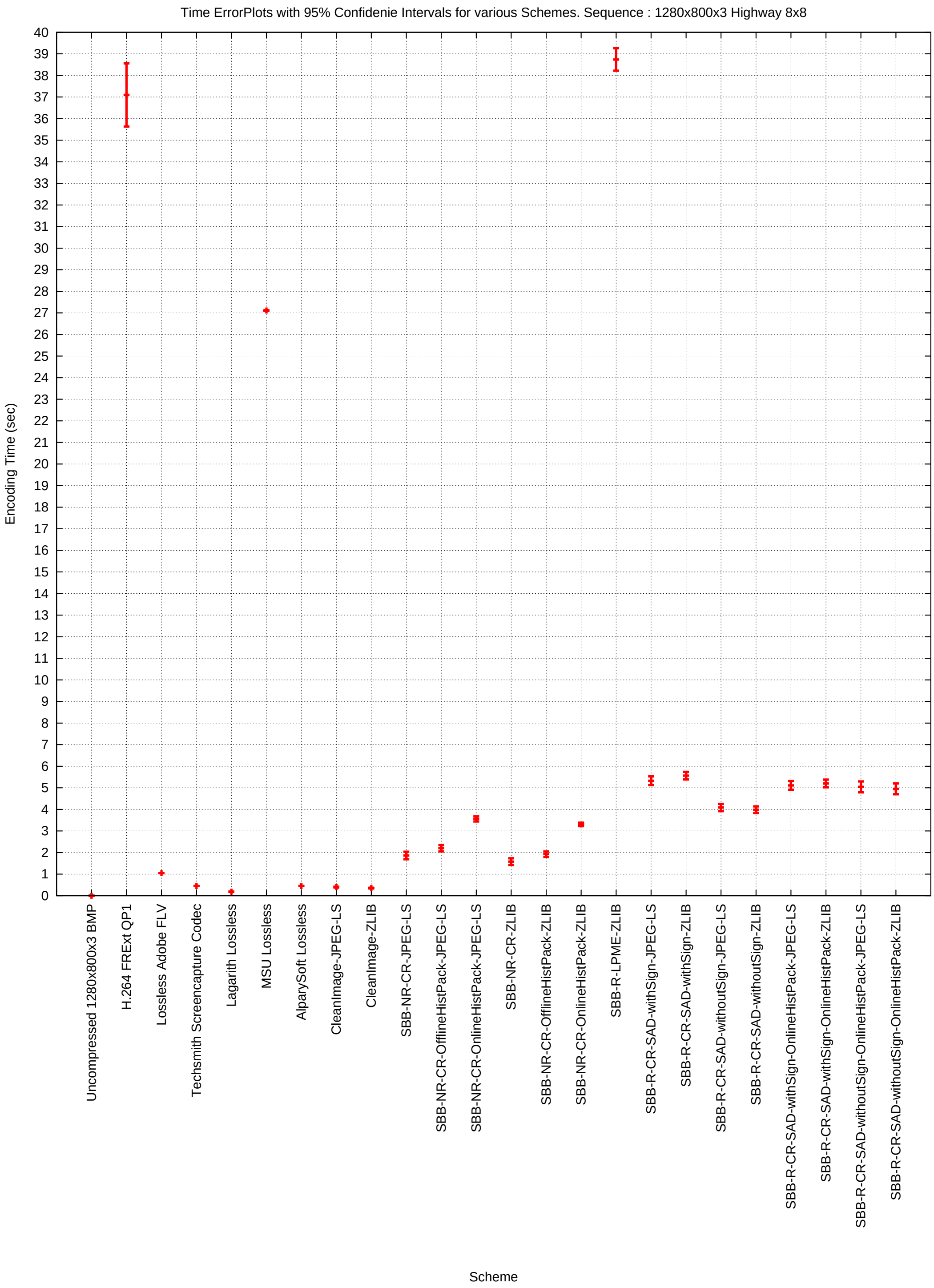


Figure 4.21: Encoding Time For Highway Sequence with 8x8 SBB-CR

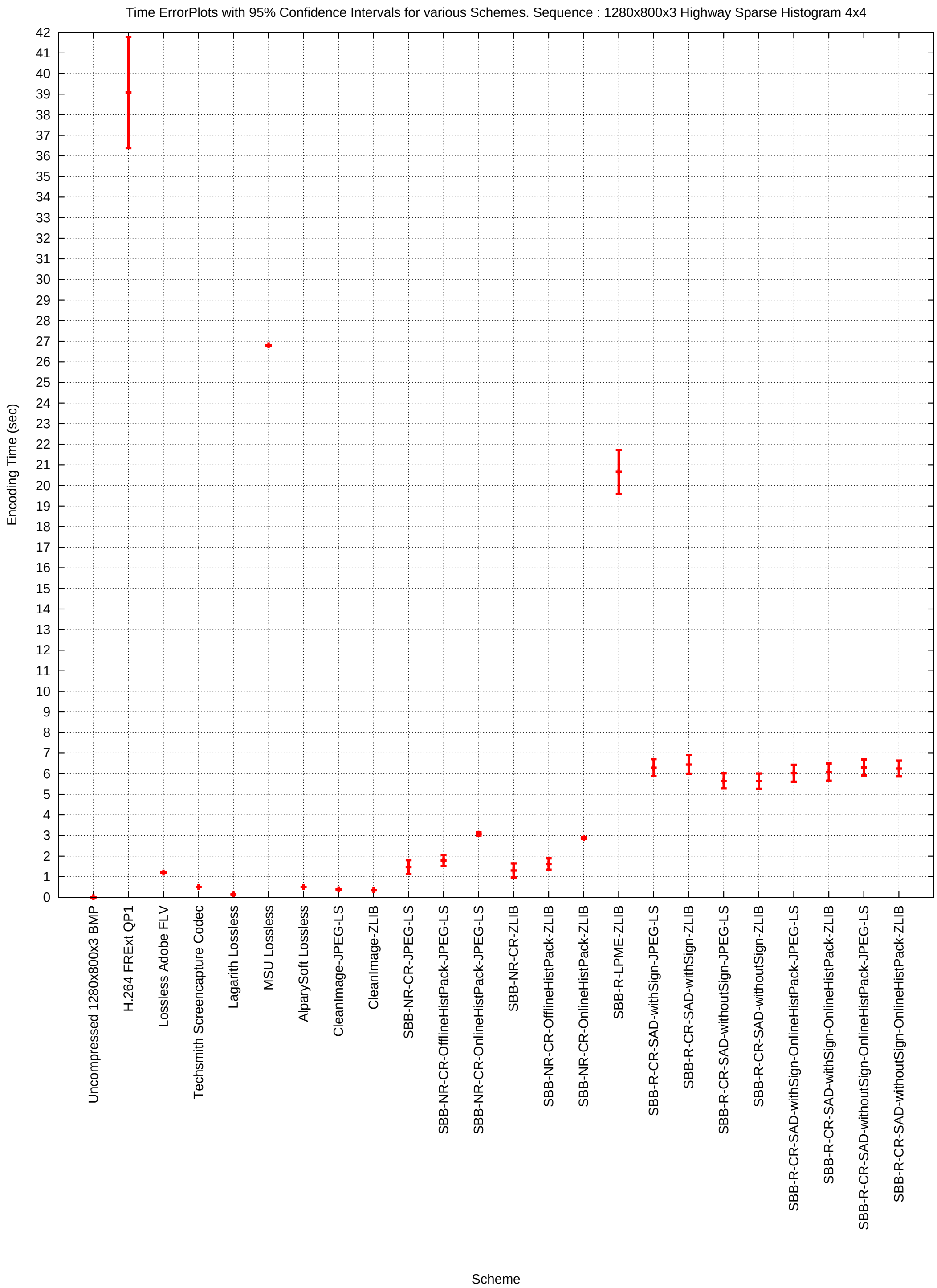


Figure 4.22: Encoding Time For Highway Sparse Sequence with 4x4 SBB-CR

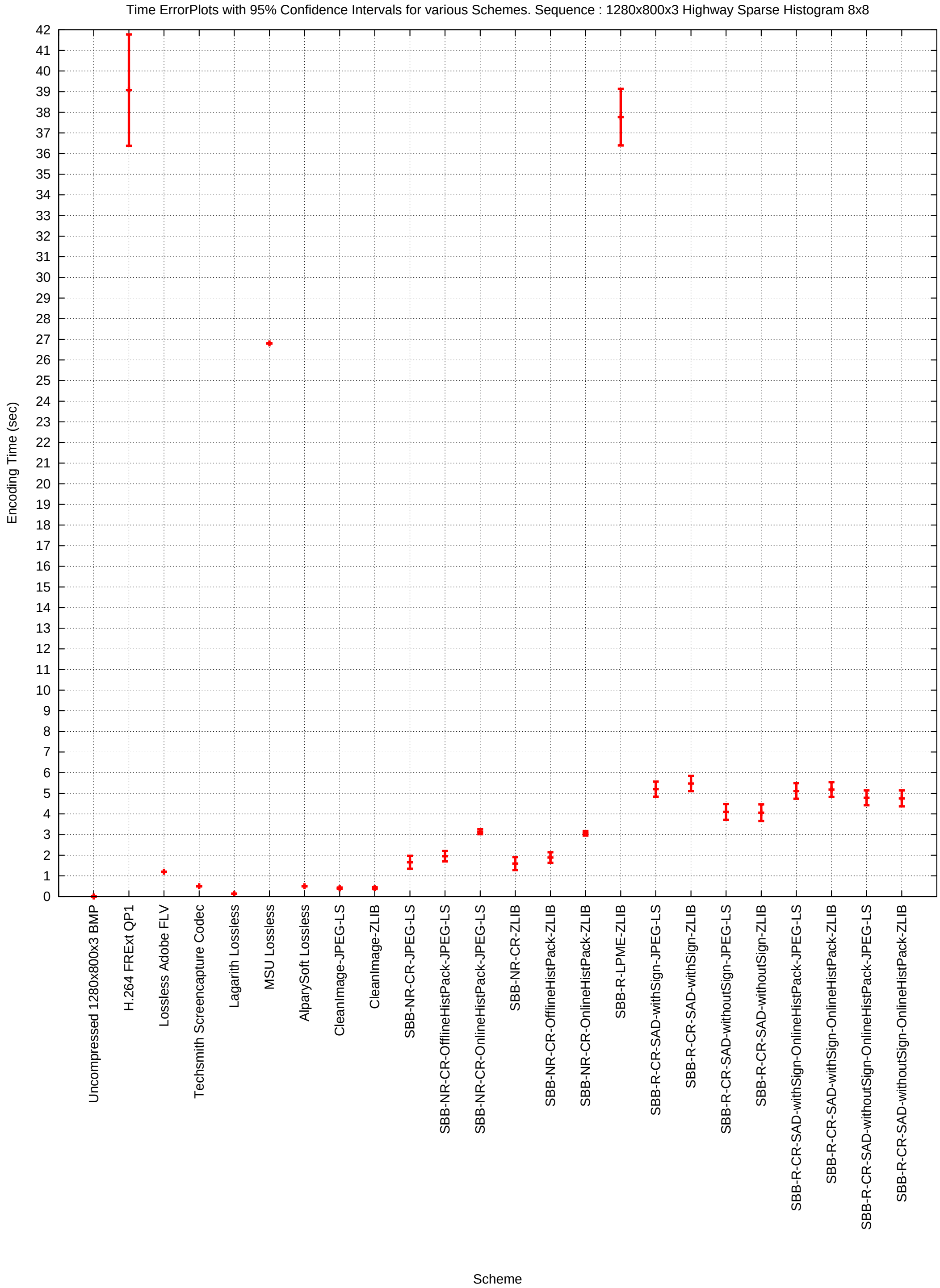


Figure 4.23: Encoding Time For Highway Sparse Sequence with 8x8 SBB-CR

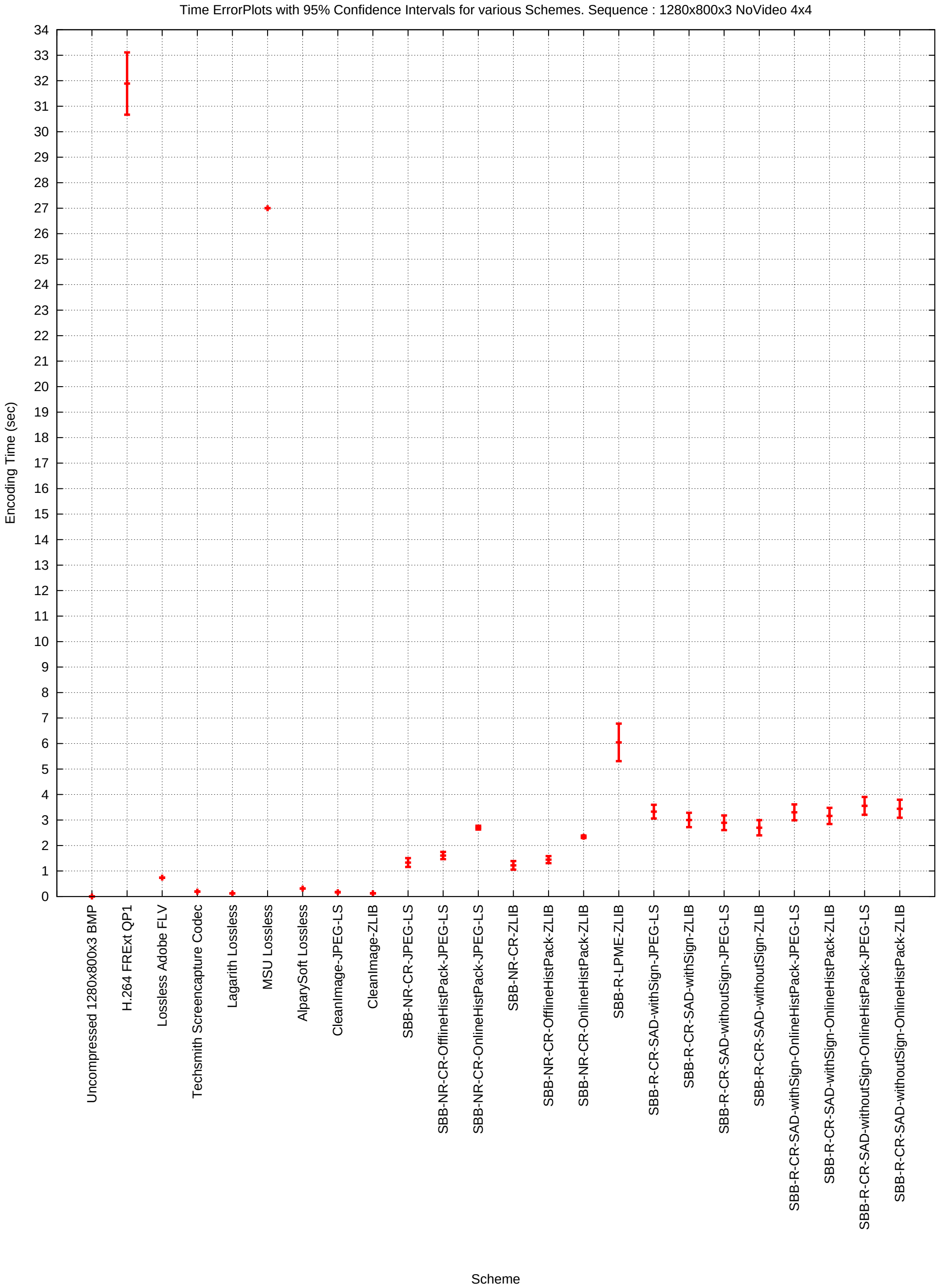


Figure 4.24: Encoding Time For NoVideo Sequence with 4x4 SBB-CR

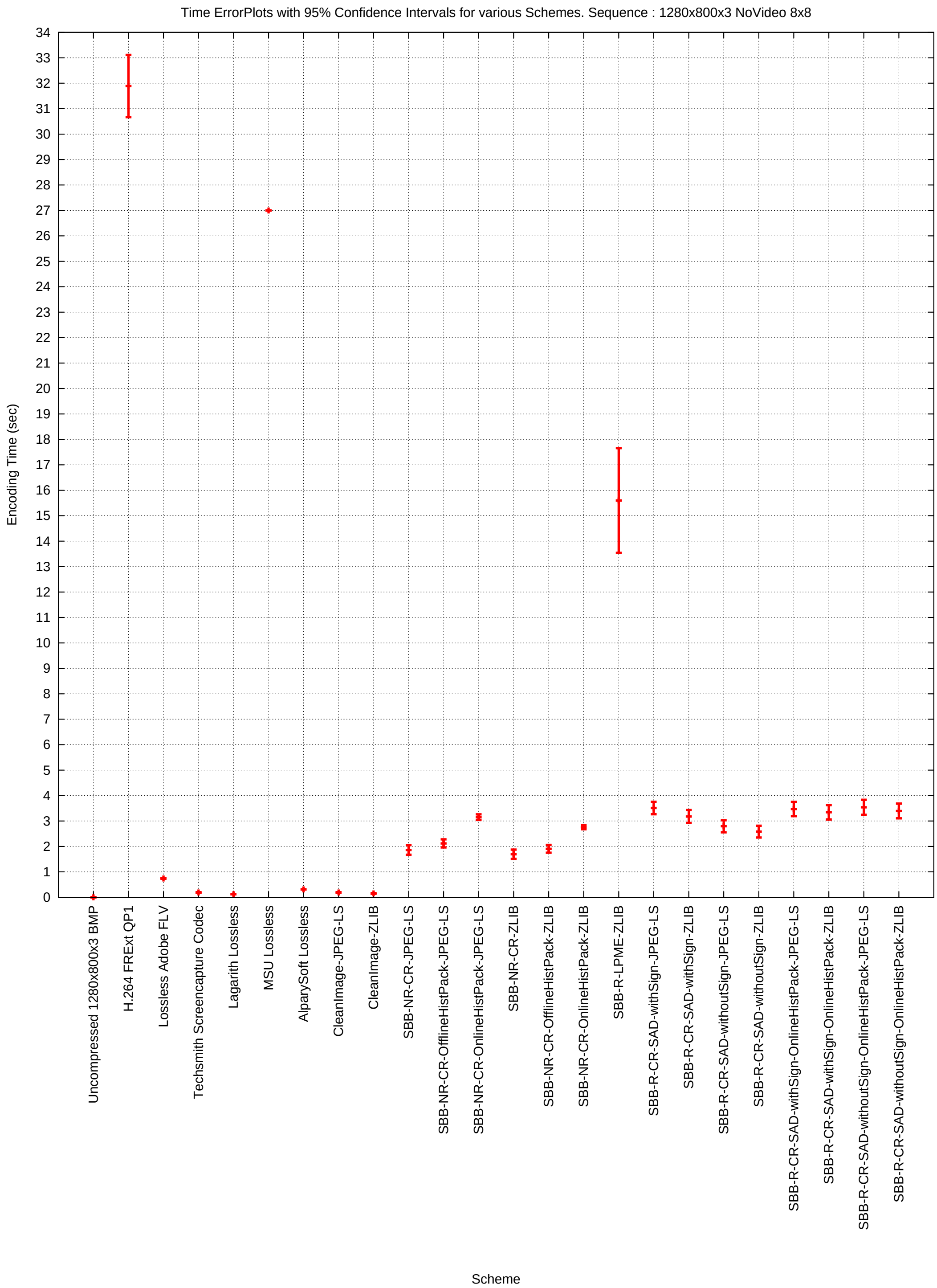


Figure 4.25: Encoding Time For NoVideo Sequence with 8x8 SBB-CR

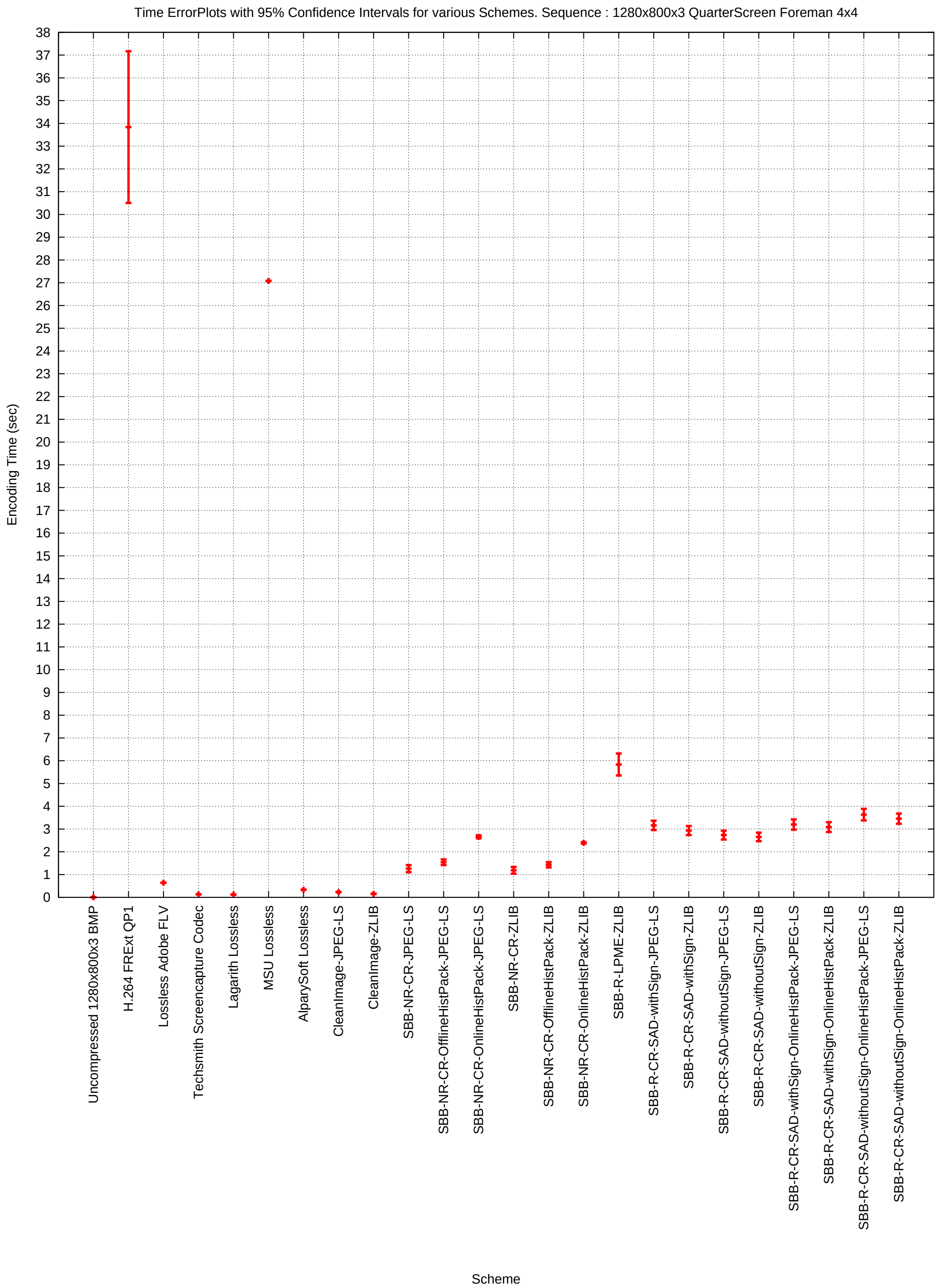


Figure 4.26: Encoding Time For QuarterScreenForeman Sequence with 4x4 SBB-CR

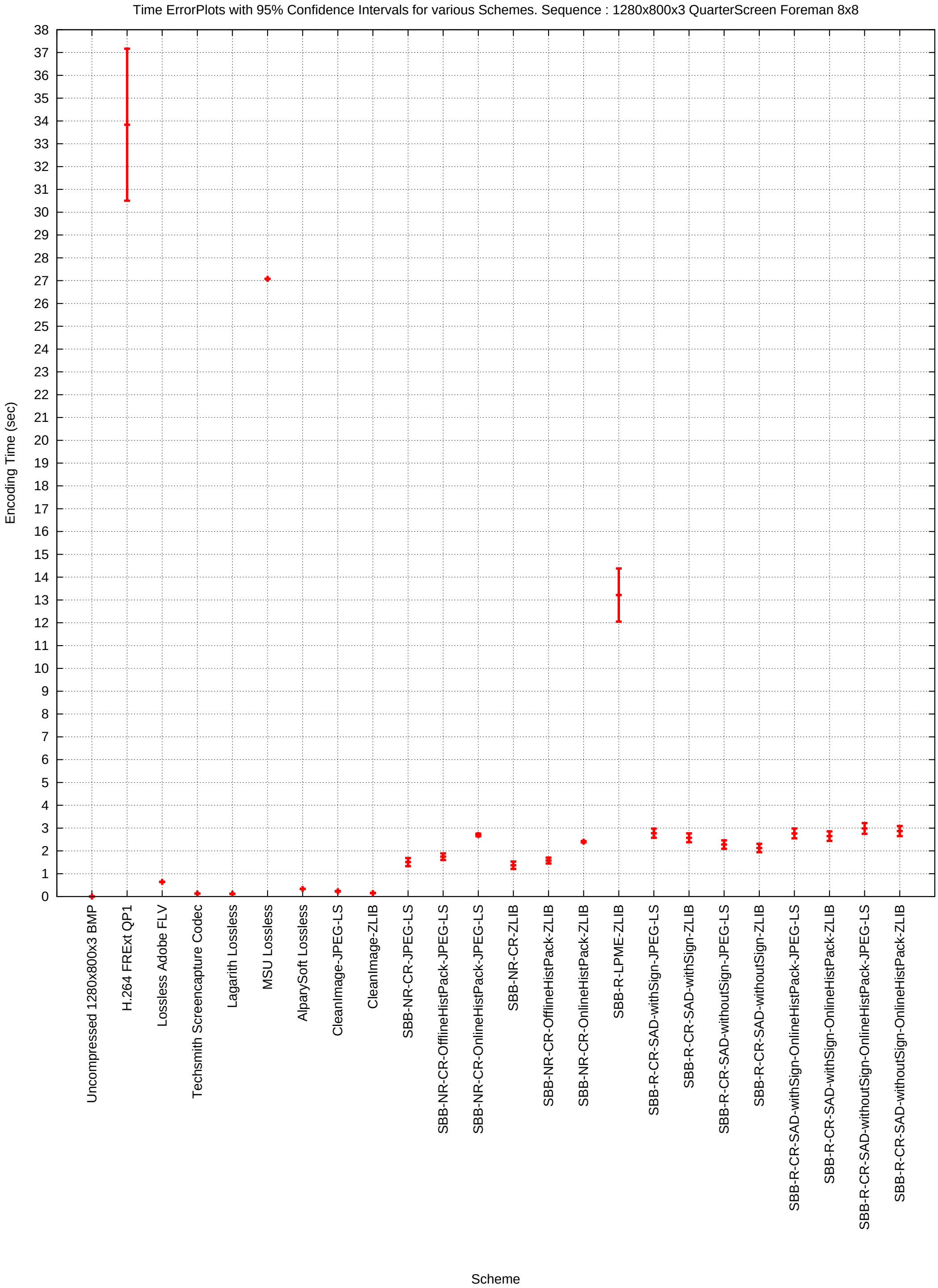


Figure 4.27: Encoding Time For QuarterScreenForeman Sequence with 8x8 SBB-CR

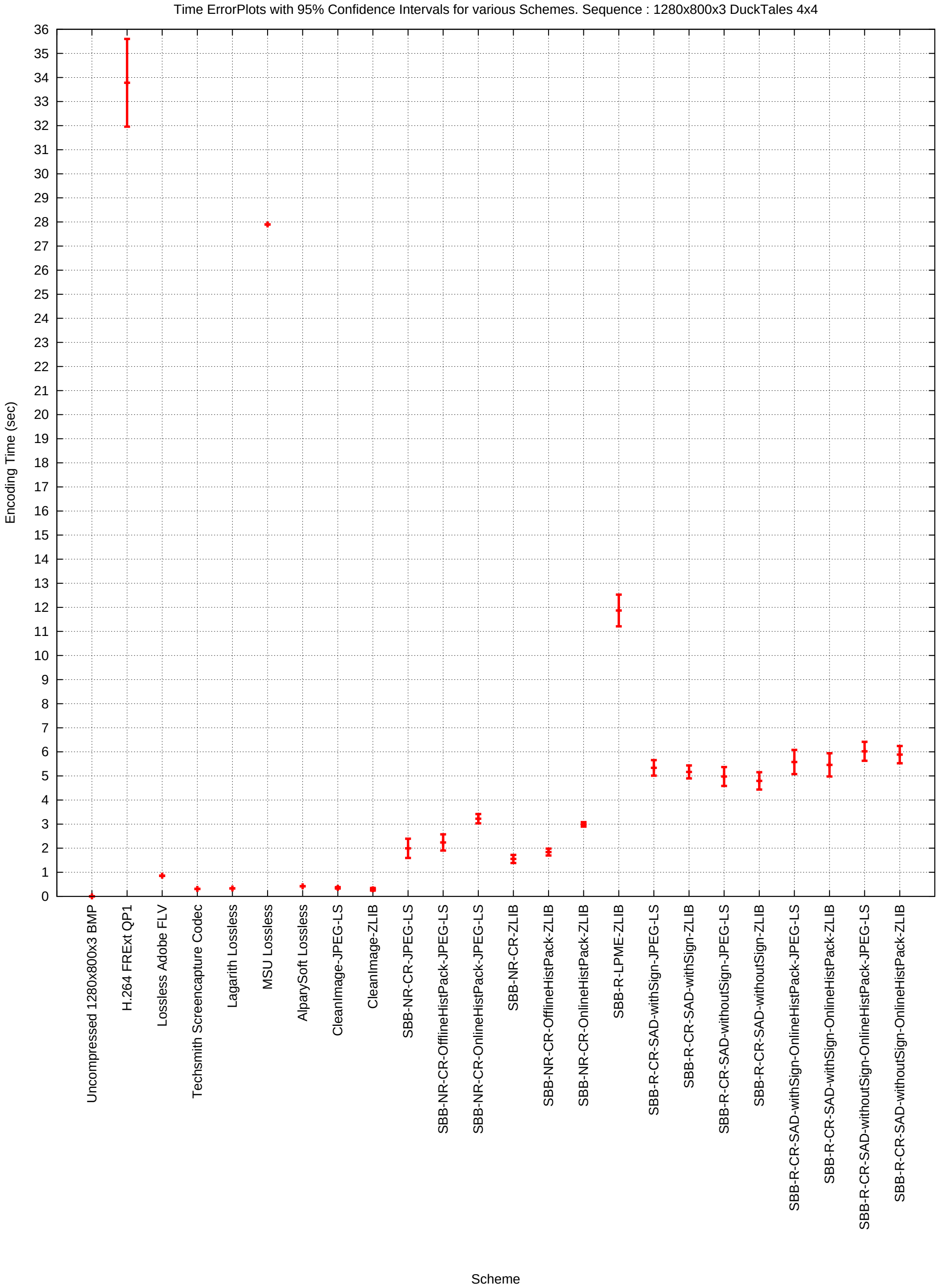


Figure 4.28: Encoding Time For DuckTales Sequence with 4x4 SBB-CR

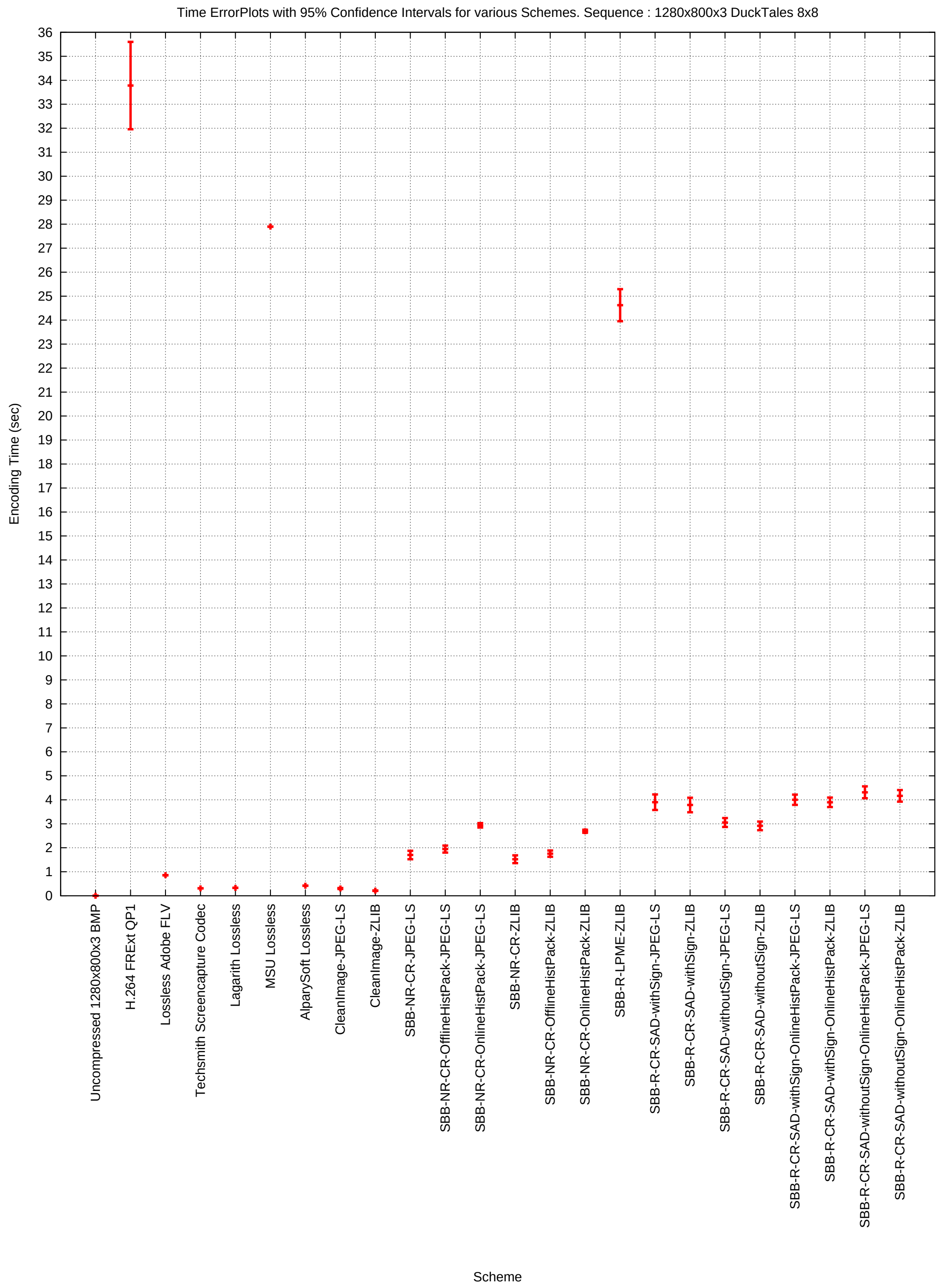


Figure 4.29: Encoding Time For DuckTales Sequence with 8x8 SBB-CR

4.3 Comparison of SBB-R-LPME-ZLIB And SBB-R-GRAY-LPME-ZLIB

In the previous charts, we have not included SBB-R-GRAY-LPME-ZLIB. The reason for doing that can be explained that SBB-R-GRAY-LPME-ZLIB takes so much more time to encode 1 image, that it was ruled out as a candidate for a low bitrate low complexity coding scheme. As the following data indicates, very little gain in compression was observed by using Gray Codes, but the time taken by the encoding was in order of magnitude more than the time achieved by SBB-R-LPME-ZLIB as shown in Figure 4.30

The following charts show what was observed as the performance of SBB-R-GRAY-LPME-ZLIB compared to SBB-R-LPME-ZLIB. Figure 4.31 shows the bitrate comparison of LPME and GRAY-LPME. As the graph indicates, using GRAY codes does not give any significant change in bitrate. Closer inspection of the data reveals that some compression gain does occur but it is on the order of a maximum of a some small kilobytes per image which is in the order of a few megabytes.

Thus from these two Figures, we conclude that SBB-R-CR-GRAY-LPME is a video encoding scheme that fares worse than SBB-R-CR-LPME, and thus was excluded from the previous analysis of bitrates and encoding times. This scheme is however of academic interest, in understanding that some other approach needs to be tried, to utilize the promise of Gray Codes. One approach that has to be tried and is a small extension to this work is trying out the 2-way Gray Code Matching.

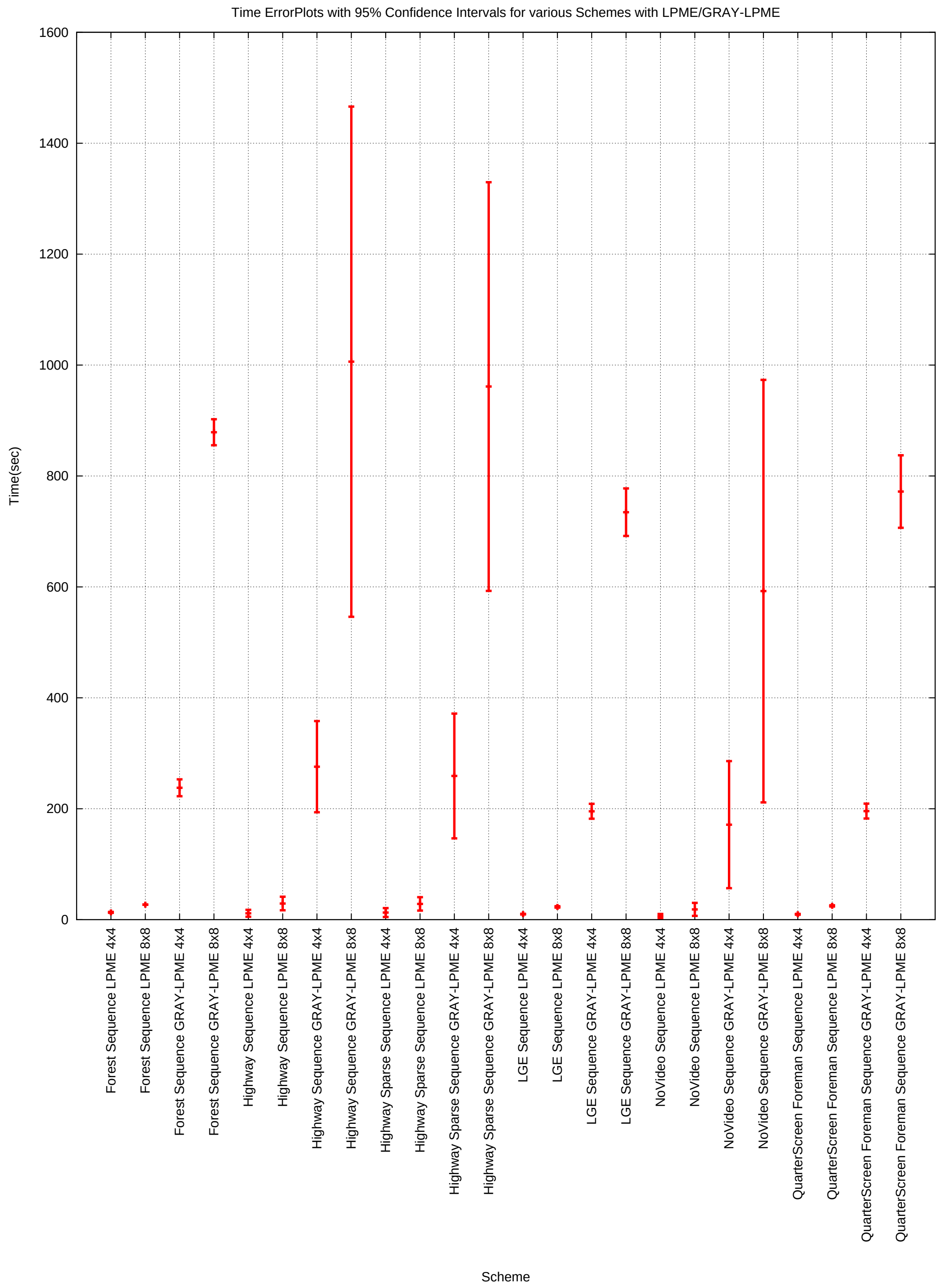


Figure 4.30: Encoding Time Plots for LPME and GRAY-LPME

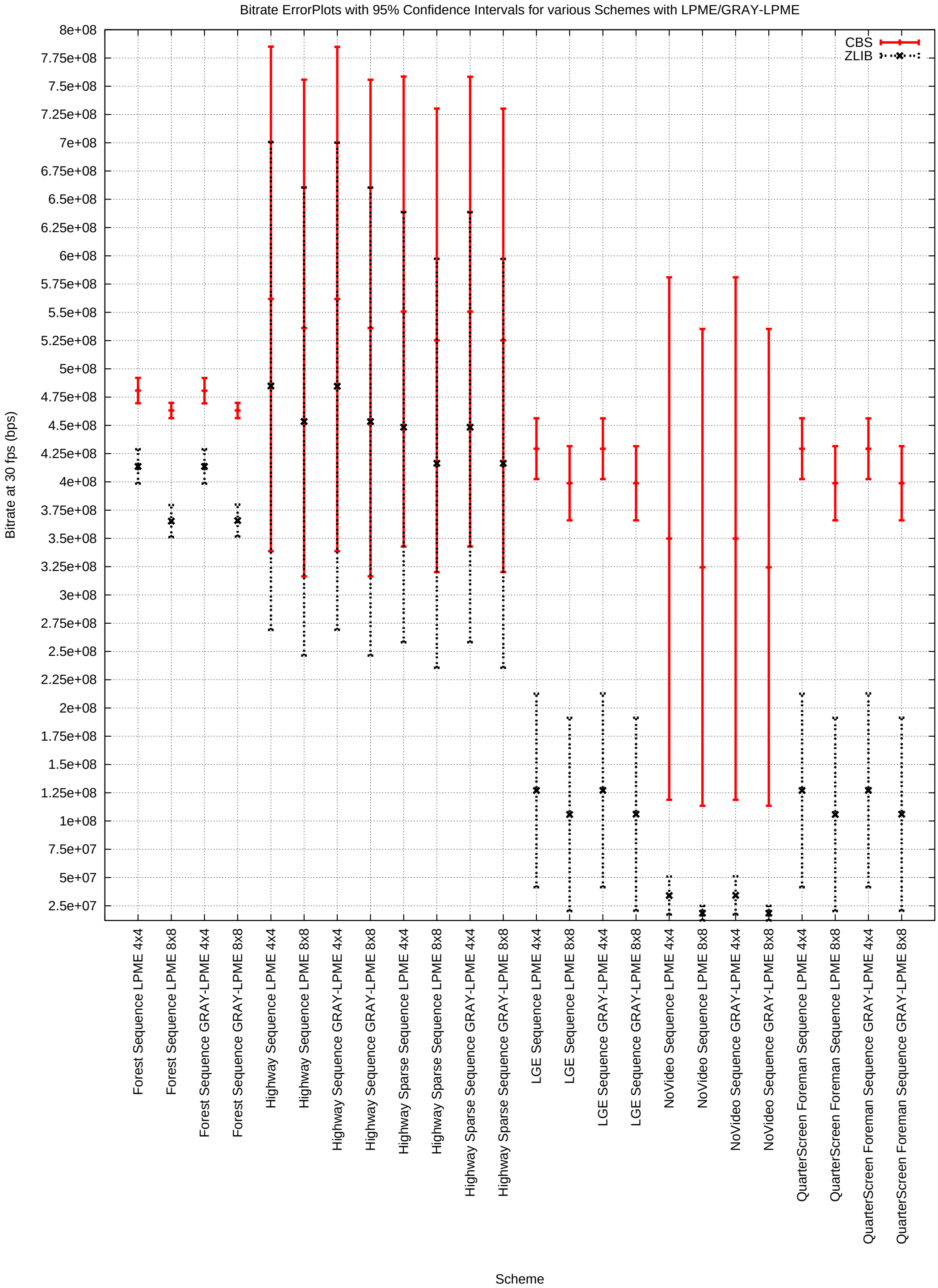


Figure 4.31: Bitrate Plot for LPME and GRAY-LPME

4.4 Bitrate Vs Encoding Time

In order to compare all of these codecs together, 2-D errorplots of these sequences for Bitrate Vs Time were obtained. From these graphs, we can conclude that the SBB-NR-CR-JPEG-LS, and SBB-NR-CR-HistPacked-JPEG-LS give the smallest bitrate and the smallest encoding times from amongst all the codecs designed herein, for natural image sequences. For artificial images such as those occurring in the NoVideo sequence, the best results were obtained by SBB-NR-CR-ZLIB in terms of bitrate and encoding time. For the DuckTales sequence, which is an animation sequence, it can be seen that SBB-NR-CR-JPEG-LS gives lesser bitrate than Lagarith.

It is also seen that H.264 FRExt although always betters SBB-NR-CR-JPEG-LS in terms of bitrate, it takes a lot of time for encoding, and thus is not suitable for real time video encoding. Similar is the case with MSU. It can be seen that Lagarith and AlparySoft are fast as well as achieve high compression. The SBB-NR-CR-JPEG-LS/SBB-NR-CR-ZLIB codec compare well with Lagarith and Alparysoft for a variety of image sets. Thus from these results we can see that the codecs attempted in this thesis satisfy the required goal of the project which was to design a high quality and fast Conditional Replenishment codec, and are infact rate competitive with the highest rate lossless codecs reported to date.

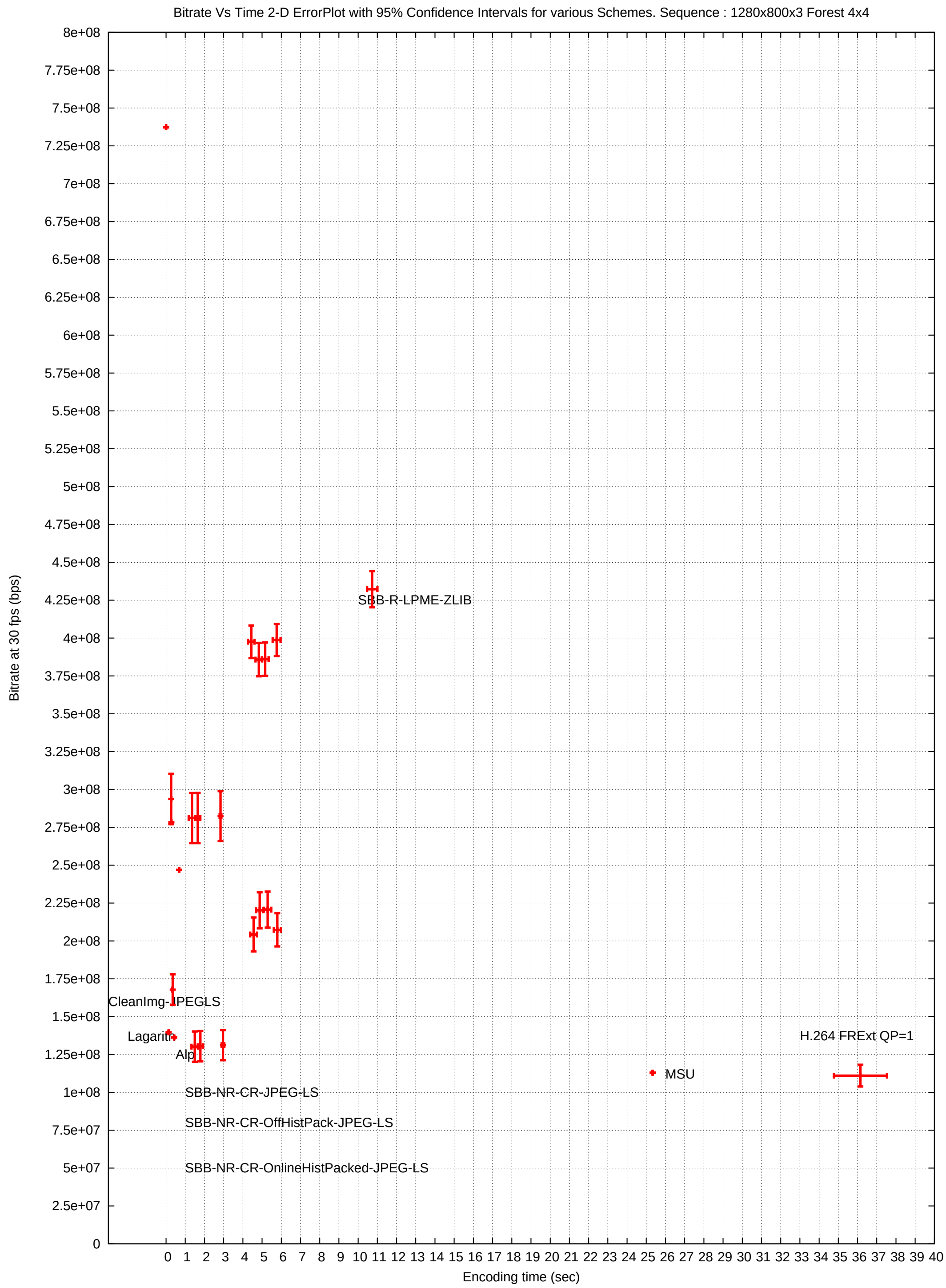


Figure 4.32: Bitrate@30 fps Vs Encoding Time For Forest Sequence with 4x4 SBB-CR

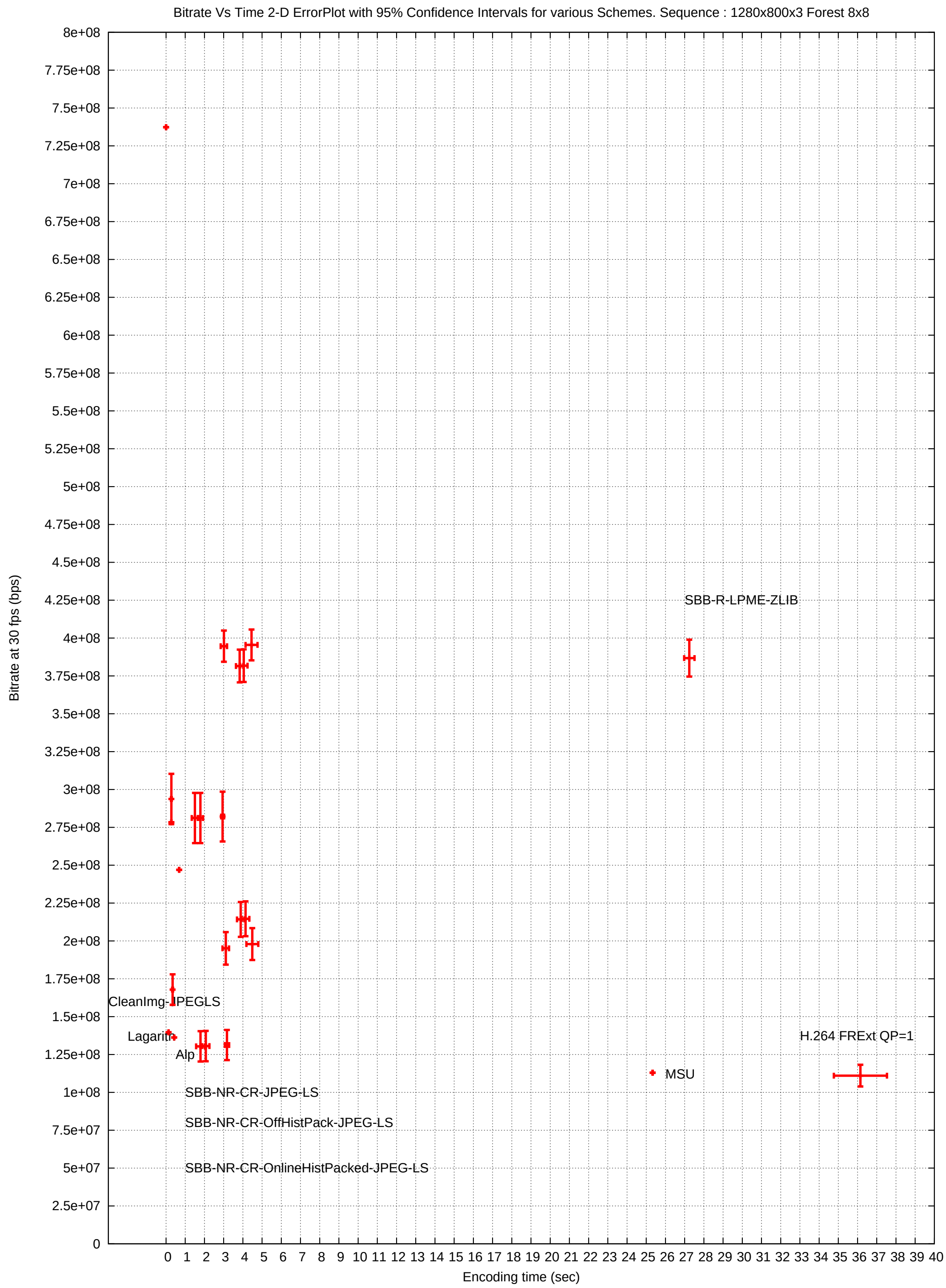


Figure 4.33: Bitrate@30 fps Vs Encoding Time For Forest Sequence with 8x8 SBB-CR

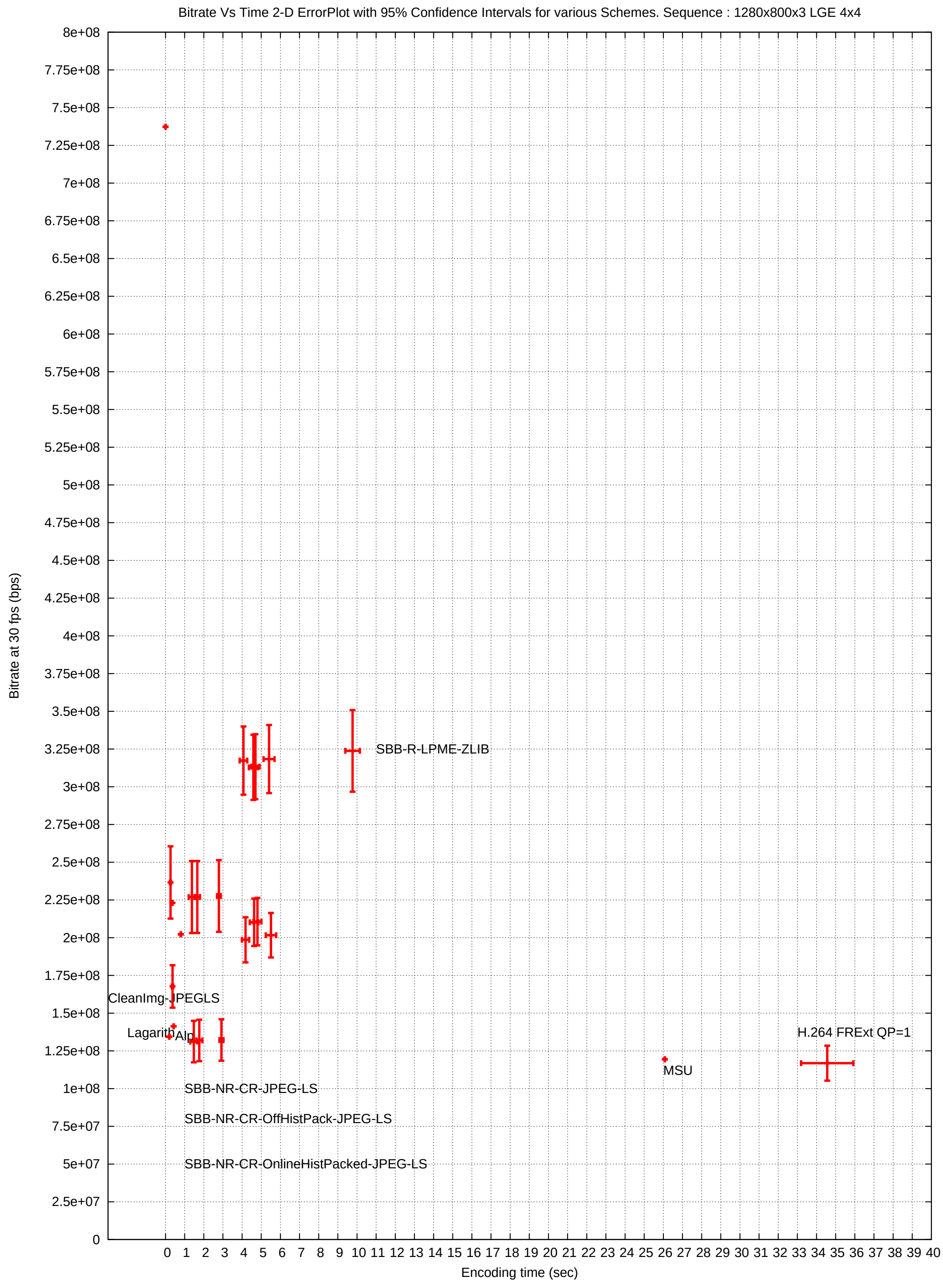


Figure 4.34: Bitrate@30 fps Vs Encoding Time For LGE Sequence with 4x4 SBB-CR

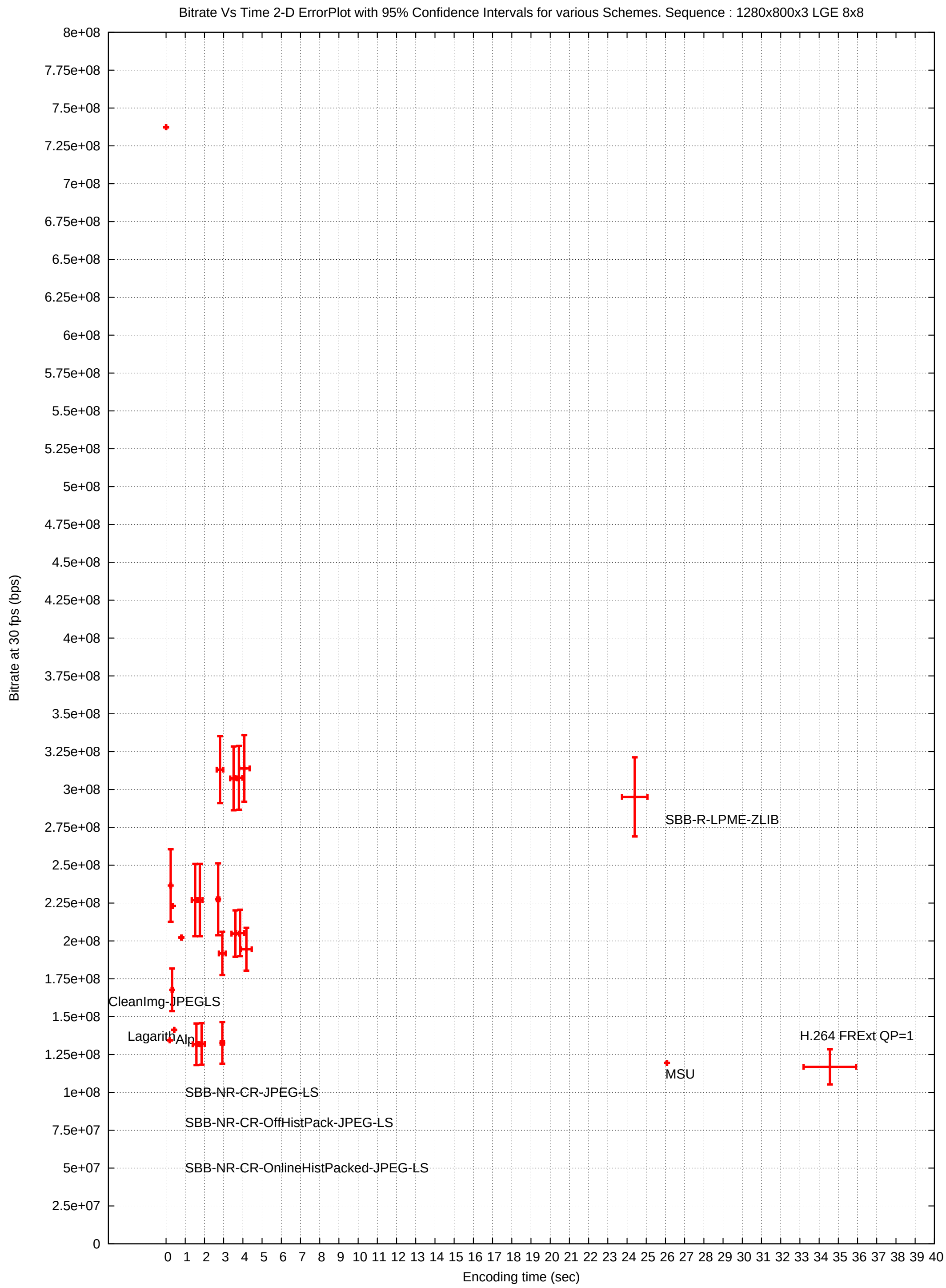


Figure 4.35: Bitrate@30 fps Vs Encoding Time For LGE Sequence with 8x8 SBB-CR

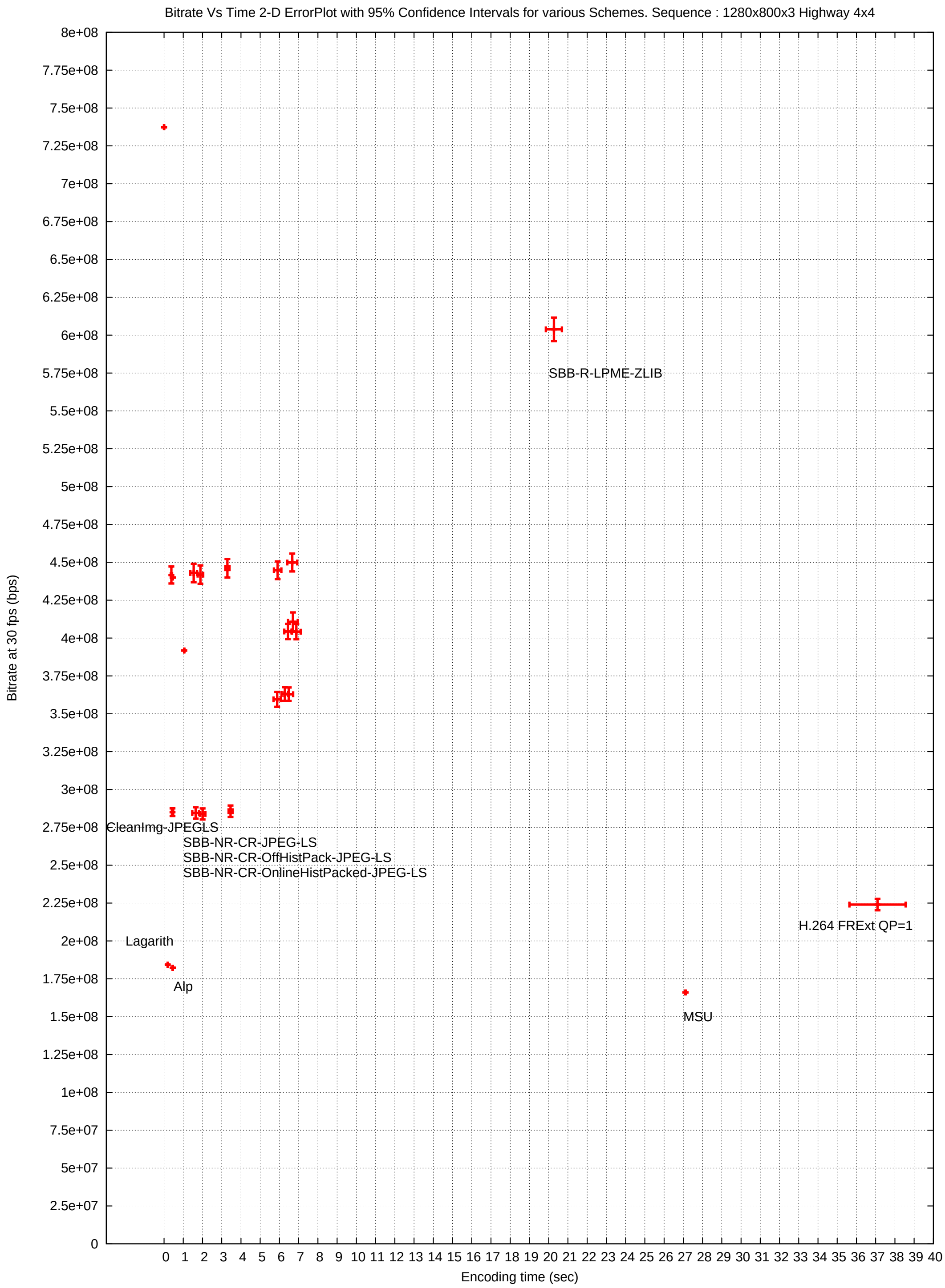


Figure 4.36: Bitrate@30 fps Vs Encoding Time For Highway Sequence with 4x4 SBB-CR

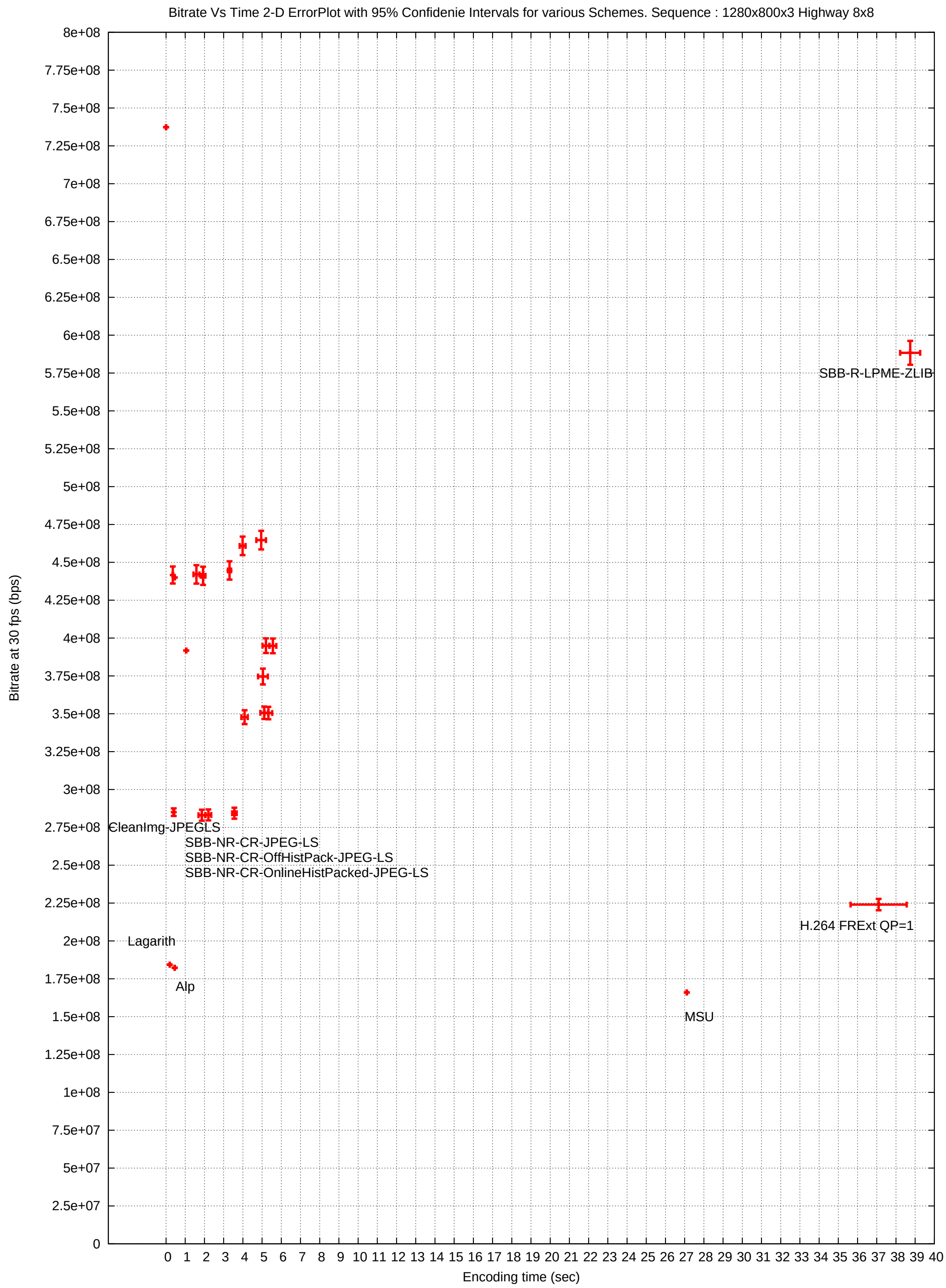


Figure 4.37: Bitrate@30 fps Vs Encoding Time For Highway Sequence with 8x8 SBB-CR

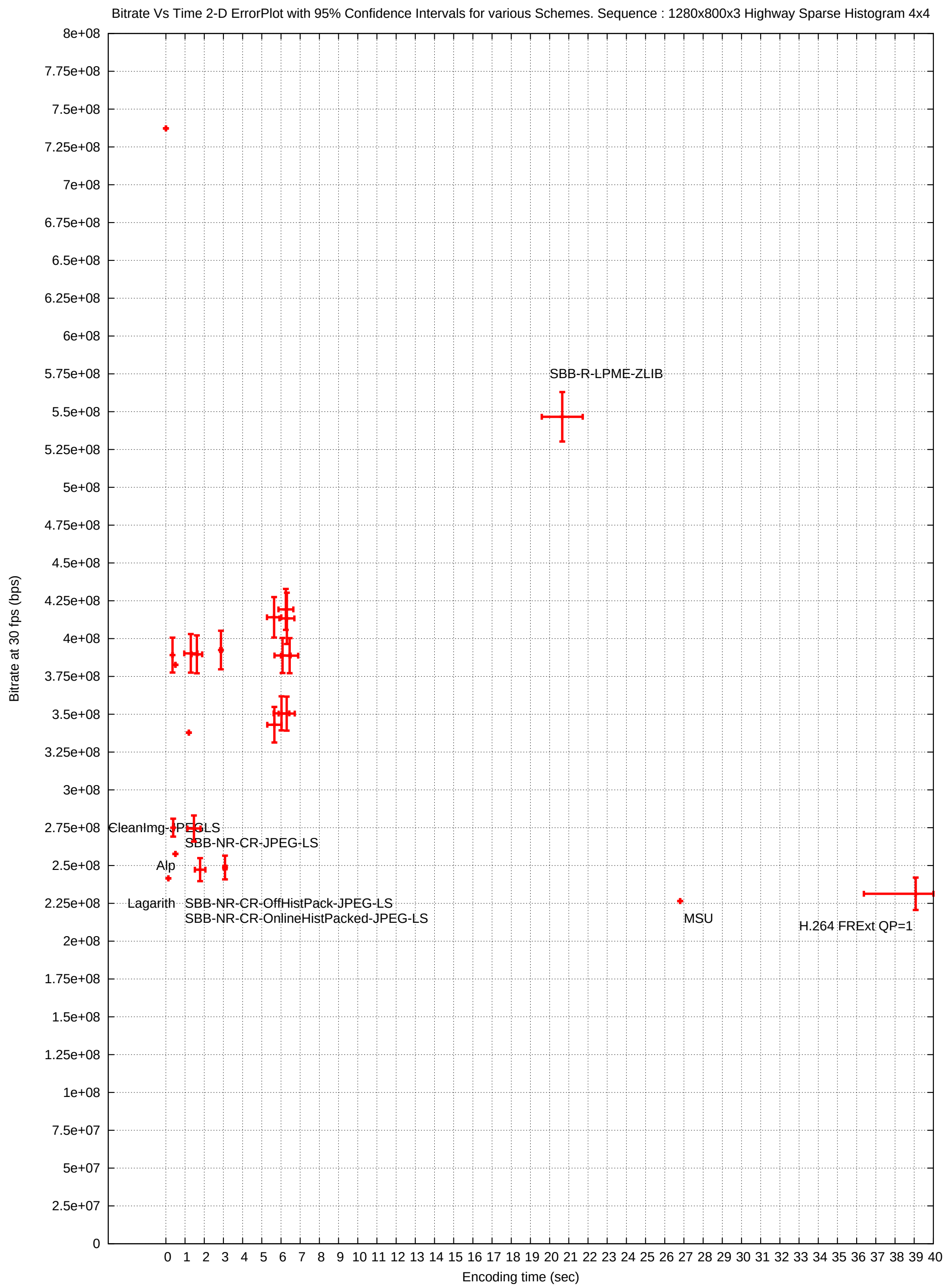


Figure 4.38: Bitrate@30 fps Vs Encoding Time For Highway Sparse Sequence with 4x4 SBB-CR

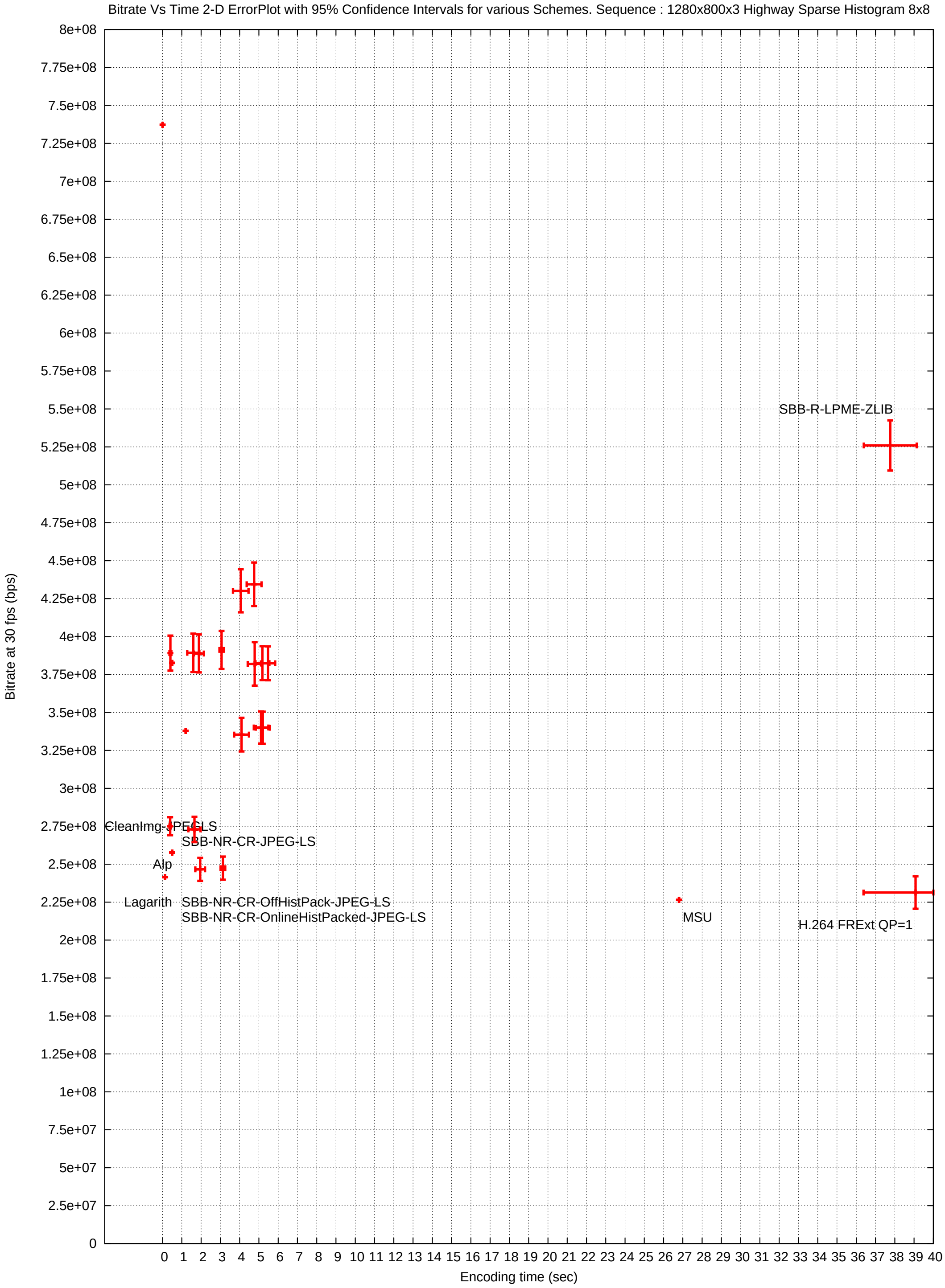


Figure 4.39: Bitrate@30 fps Vs Encoding Time For Highway Sparse Sequence with 8x8 SBB-CR

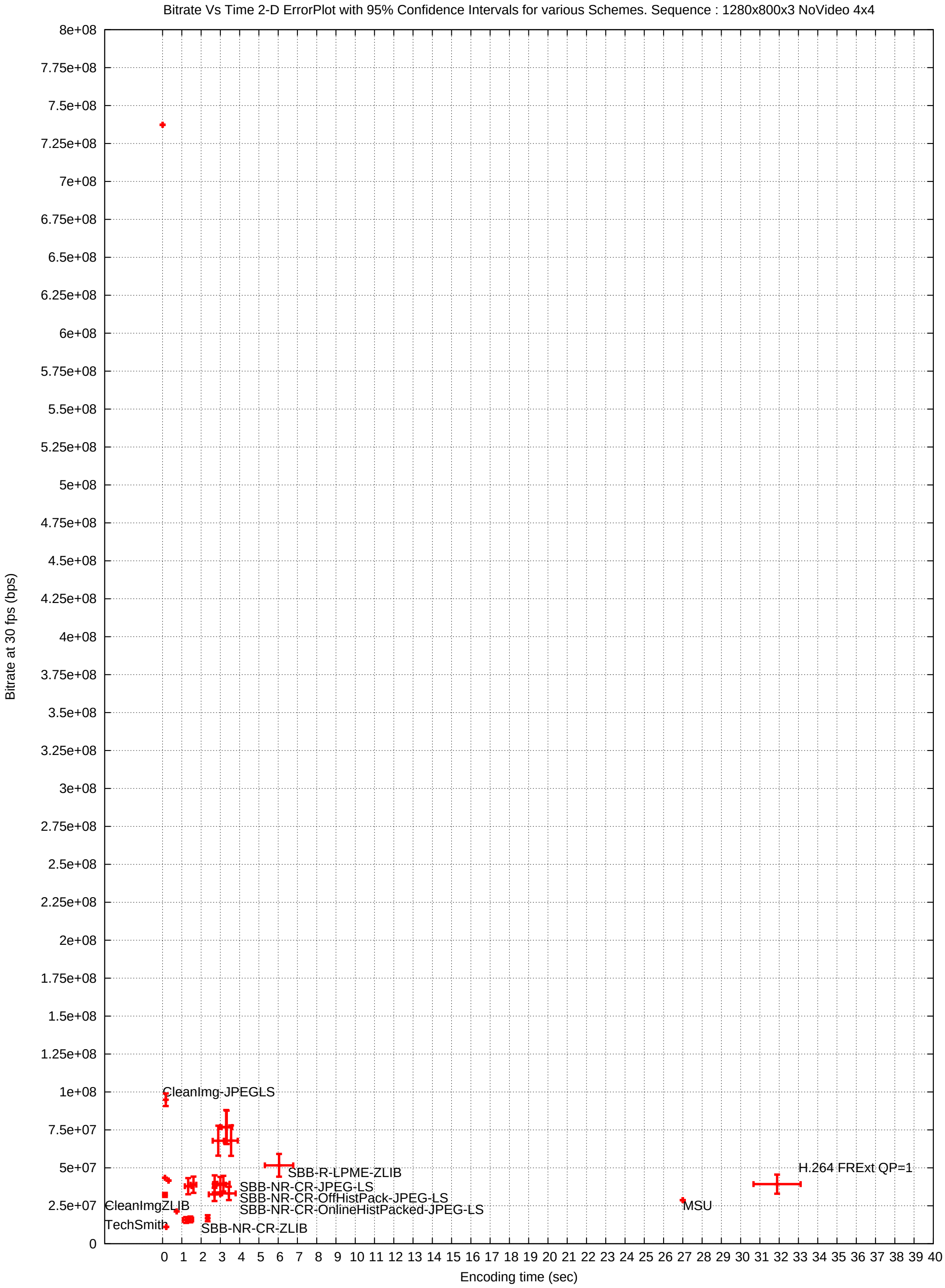


Figure 4.40: Bitrate@30 fps Vs Encoding Time For NoVideo Sequence with 4x4 SBB-CR

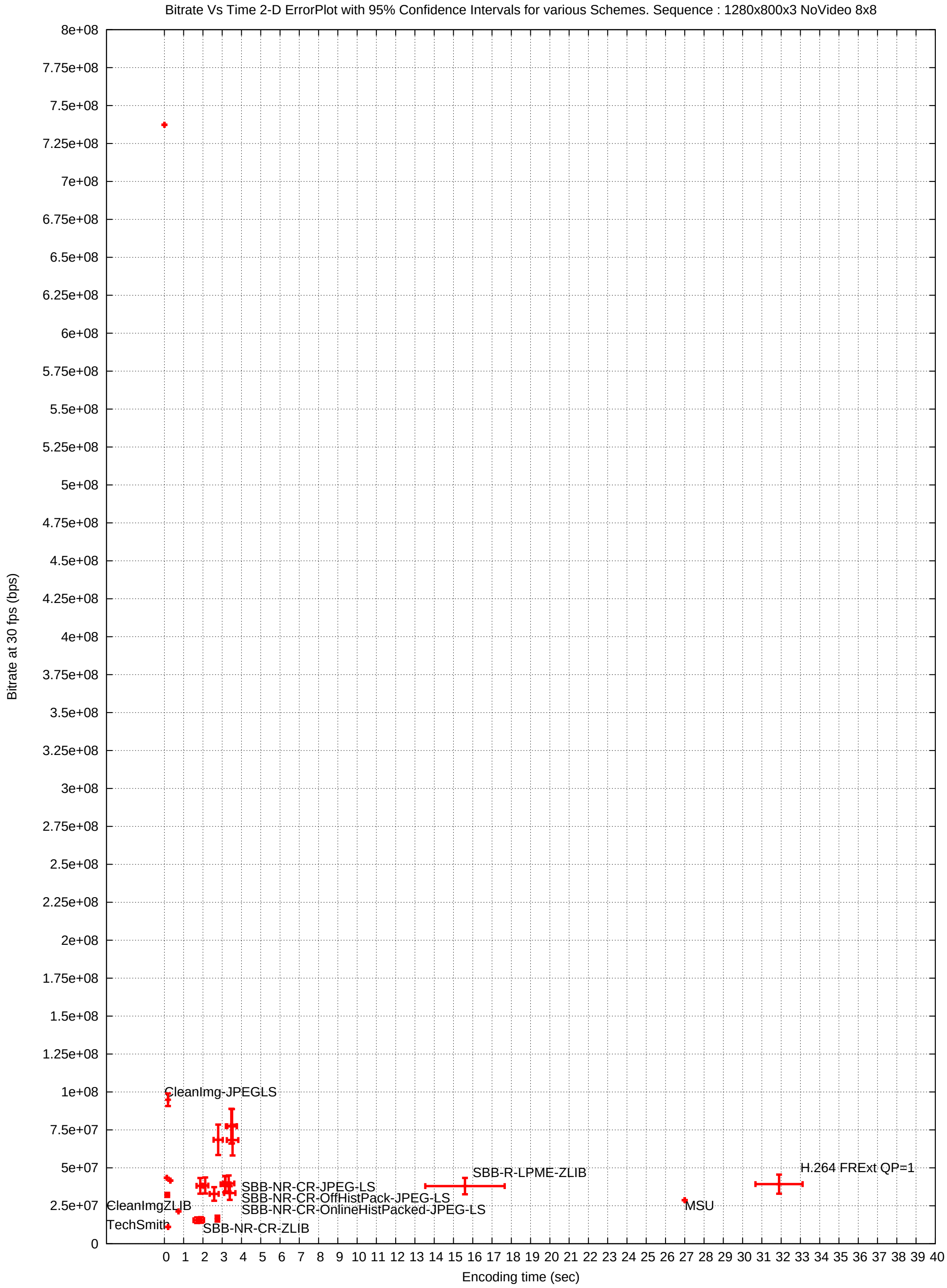


Figure 4.41: Bitrate@30 fps Vs Encoding Time For NoVideo Sequence with 8x8 SBB-CR

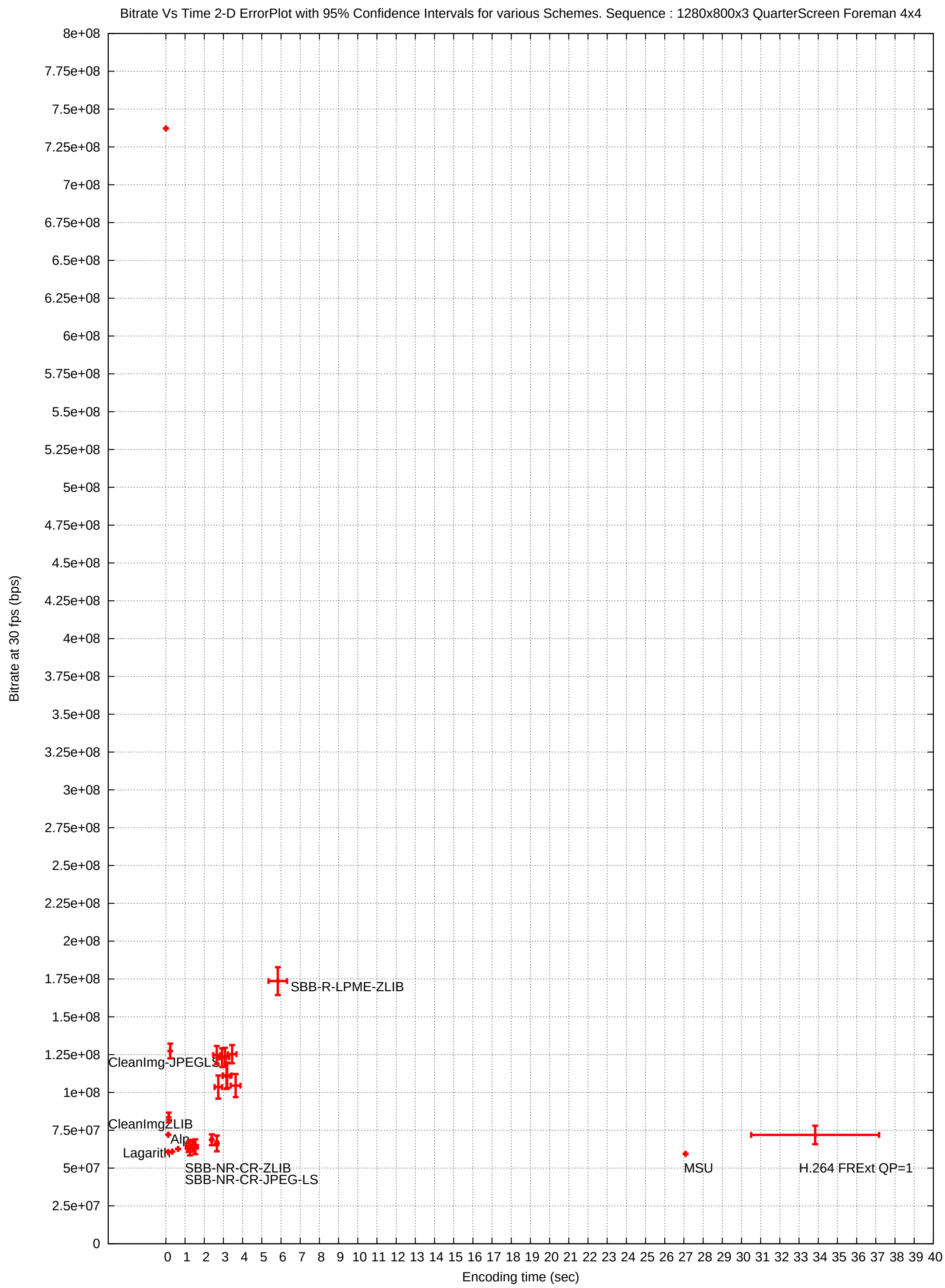


Figure 4.42: Bitrate@30 fps Vs Encoding Time For QuarterScreenForeman Sequence with 4x4 SBB-CR

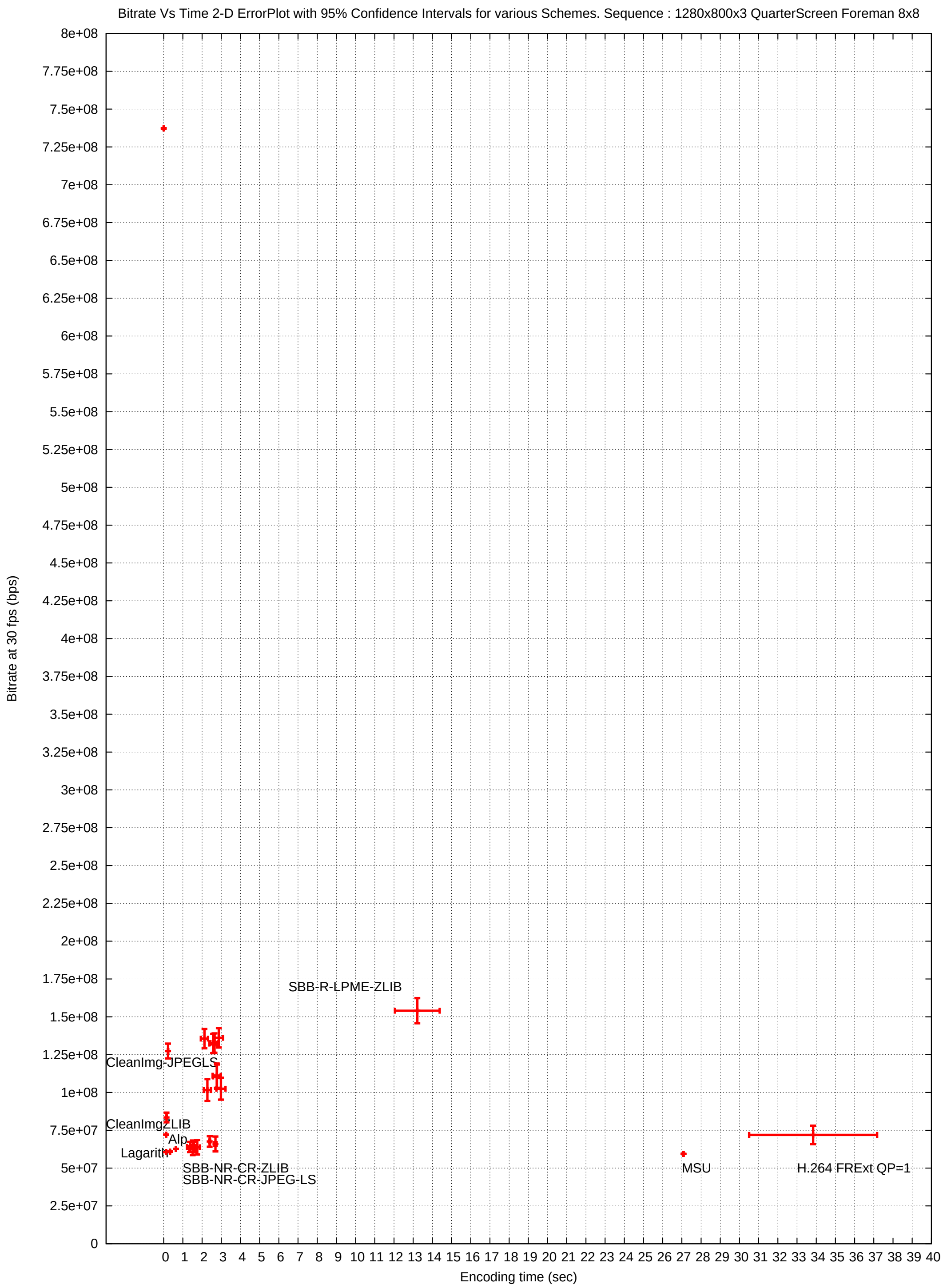


Figure 4.43: Bitrate@30 fps Vs Encoding Time For QuarterScreenForeman Sequence with 8x8 SBB-CR

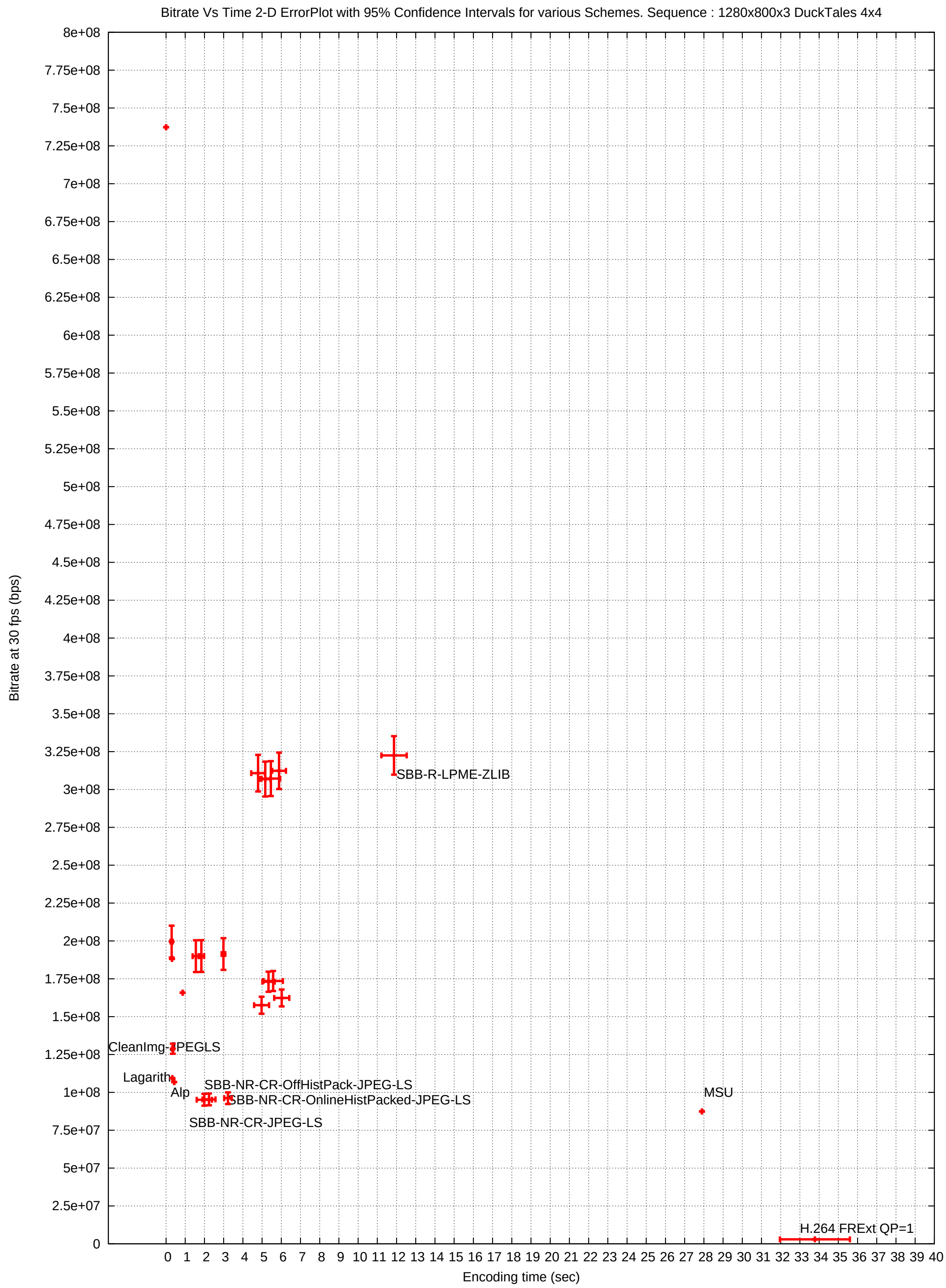


Figure 4.44: Bitrate@30 fps Vs Encoding Time For DuckTales Sequence with 4x4 SBB-CR

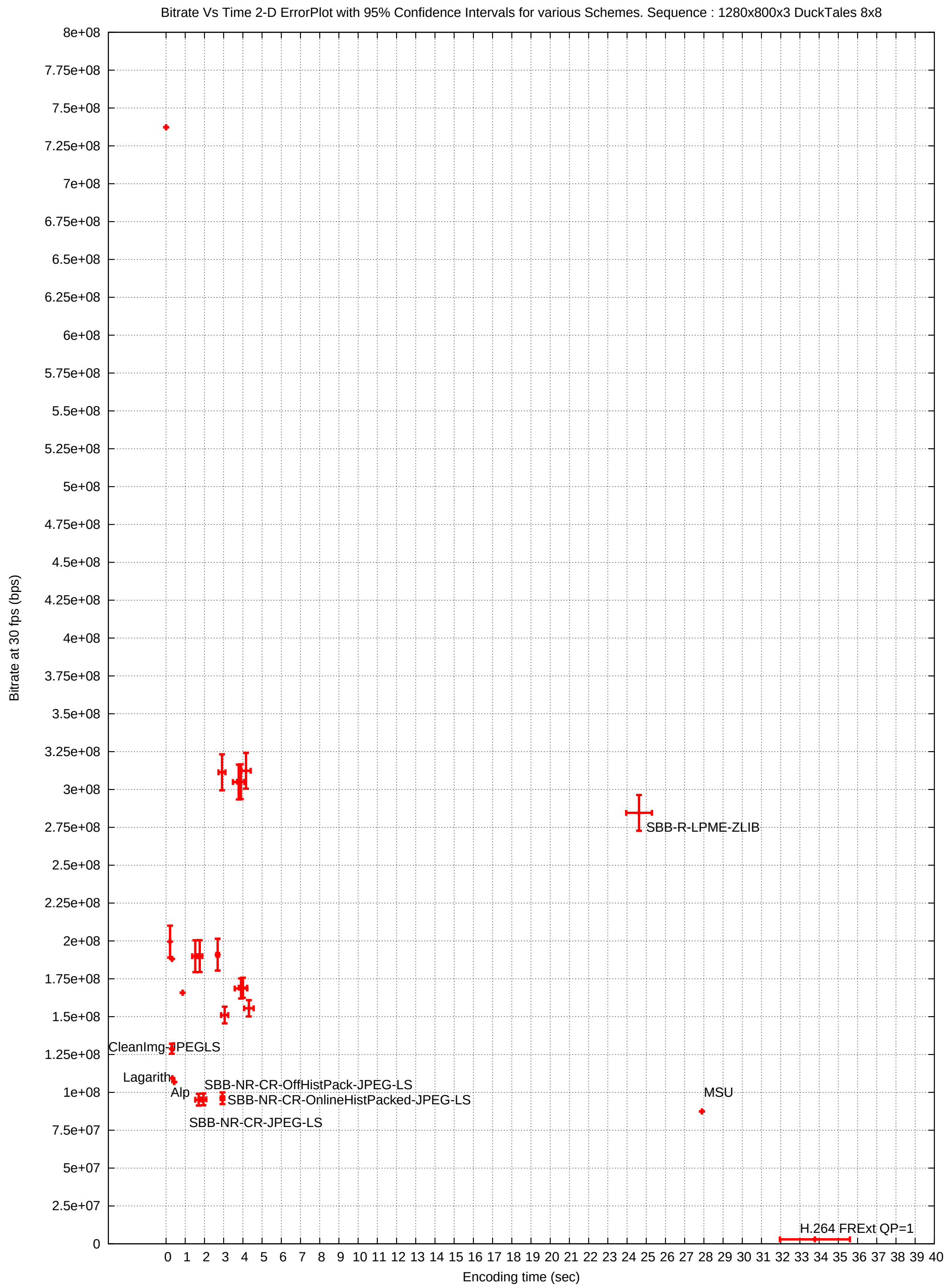


Figure 4.45: Bitrate@30 fps Vs Encoding Time For DuckTales Sequence with 8x8 SBB-CR

Chapter 5

Future Work

The schemes that are considered in this work are shown in Figure 1.2 and in that figure, the schemes marked with "FW" mean future work. In this section, we would describe those schemes in some further detail.

Amongst the Non-Residual schemes, future work includes a systematic study of the VNC protocol. We in this thesis rely on the study of Nieh et. al. [8], about the compression performance of VNC, but do not do those experiments about VNC ourselves.

Further, a scheme based on SBB-NR-CR-Lossy encoding using an Intra frame refresh, which too can lead to a high quality was left out of the Conditional Replenishment schemes discussed herein, because, we decided to focus on lossless schemes. It remains a task of the future to see where exactly this scheme stands in the quality/bitrate schema. A SBB-NR-CR lossless scheme in which a block is replaced by another block that has lower prediction error than itself, based on the proposal by Memon and Sayood in [5], is also part of future work.

Amongst schemes based on residuals/prediction, working on SBB-R-CR-Lossy, and comparing such schemes with MPEG-2, H.264 schemes is also a task of the future. It was argued in the thesis that SBB-R-CR-Lossy is very similar to what MPEG-2/H.264 schemes are. In order to verify this claim, further experiments need to be performed.

In the design of SBB-R-LPME-ZLIB, it was noted that in the SBB-R-Lossless-LPME-ZLIB approach, instead of using the Exp-Golomb Code, using a context based Golomb code might have improved the compression rate significantly. An important reason for the poor performance of this scheme is the strength of the longest prefix matching motion method. The better the match, the smaller the error, and smaller the error, smaller its Exp-Golomb code and smaller the bitstream. Further it was clearly seen that there were several SKIP blocks in the final bitstream, thus indicating a poor selection in using Exp-Golomb as the encoding method. It however remains an open problem as to how the context based coding can be done for longest prefix matches. Further, we saw that we could not utilize the promise of Gray Codes effectively. One straightforward future work is to extend the one way LPME using Gray Codes to two way matching.

Further, in the SAD Motion Estimation based residual schemes, followed by lossless compression (SBB-R-CR-Lossless), implementing an improved variant of the motion estimation error metric, based on minimizing the prediction error of the lossless image encoding scheme, instead of using SAD, as proposed by Memon et. al. [5] is part of future work. Also the optimal 3-D prediction as mentioned in [10] has to be tried in the context of SBB-R-SAD-Lossless based schemes.

It also remains a task of the future to extend these schemes to using other schemes such as those based on Arithmetic encoding. The SBB-R-CR-LPME version was passed through arithmetic encoding as well as other schemes such as those based on Discrete Markov Coding, but it did not lead to any significant changes in bitrate, and thus those results are not included herein. It remains a task of the future to see which entropy codec will give the best performance for the schemes discussed herein.

In [25], Roman Starosolski proposes a fast lossless compression method that is 2 times faster than JPEG-LS and is comparable in terms of compression achieved by JPEG-LS. Using this method instead of JPEG-LS has the ability to improve the compression speed.

Also, the SBB-NR-CR Schematic as shown in Figure 2.1 is amenable to parallelization. The SBB-NR-CR operation can be done in parallel at a block level, and inside a block, the per-pel comparison can also be done in parallel. Figure 5.1 shows the way we can achieve the parallelization of SBB-NR-CR schematic using threads and semaphores. Each frame is divided into blocks, and the operation of entire frames can be pipelined.

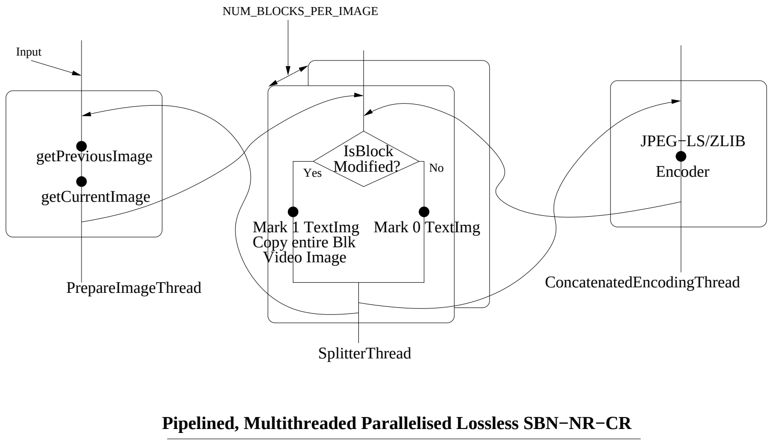


Figure 5.1: Parallelized, Pipelined SBB-NR-CR

Chapter 6

Conclusion

The work shows that better (w.r.t the project goal of achieving a low complexity, low bitrate and high quality video encoding scheme) compression results are obtained, on average, using lossless compression mode, than using lossy compression mode for the SBB-NR-CR based video encoding scheme. It was also seen that Conditional Replenishment resulted in better quality video at the same bitrate than the one that was achieved without Conditional Replenishment. We further see that JPEG-LS is more suited for lossless Conditional Replenishment than JPEG-2k because it shows similar compression performance, but works tens of times faster than JPEG-2k. Further, the work also confirmed the compression improvement for sparse histogram images using Histogram Packing and discussed the design, implementation and performance of several variants of this Simple Block Based Conditional Replenishment technique such as ones based on Longest Prefix String Matching, Longest Prefix String Matching using Gray Codes, Motion Estimation based Residual Conditional Replenishment. The work also confirmed the finding that residual encoding based on Sum of Absolute Differences Motion Estimation, that has proved to be so effective a technique in lossy video encoding, does not directly help in lossless video encoding. The results show that for natural image sequences, the codec based on Simple Block Based, Non-Residual, Conditional Replenishment followed by Histogram Packing followed by JPEG-LS gives bitrate and encoding time performance, comparable to the best lossless codecs reported, and for artificial images such as images obtained by desktop capture, Conditional Replenishment followed by ZLIB gives the most competitive results. The work has direct applicability in the implementation of low computation, lossless video encoding schemes such as those used for server-based computing, and video streaming.

Bibliography

[1] L. Hanzo, P.J Cherriman, J. Streit, "Modern Video Compression and Communications over Wireless Channels : From Second to Third Generation Systems, WLANs and Beyond", Dept. of Electronics and Computer Science, University of Southampton, UK.

[2] Masayuki Nakao, Masaki Kita, Masayuki Miyama, "High Quality Image Compression System for Low Delay and Real-time Wireless Transmission", *SID Symposium Digest of Technical Papers – June 2006 – Volume 37, Issue 1, pp. 1896-1899*

[3] Armando J. Pinho, "An Online Preprocessing Technique for Improving the Lossless Compression of Images With Sparse Histograms", *Signal Processing Letters, IEEE, Nov. 2001, Volume 9, Issue 1, Jan 2002 Page(s): 5 - 7*

[4] Topiwala Pankaj, *http://www.norbertwiener.umd.edu/FFT/FFT06/06_Titles-and-abstracts/pankaj-topiwala.html*

[5] Nasir Memom, Khalid Sayood, "Lossless Compression of Video Sequences", *IEEE Transactions on Communications, Vol. 44, NO. 10. October 1996*

[6] Bruno Carpentieri,Marcelo J. Weinberger,Gadiel Seroussi, "Lossless Compression of Continuous Tone Images" *HP Labs Technical Reports, HPL-2000-163*

[7] *http://compression.graphicon.ru/video/codec_comparison/lossless_codecs_2007_en.html*

[8] Jason Nieh, S. Jae Yang, "Measuring the Multimedia Performance of Server-Based Computing", *Proceedings of the Tenth Workshop on Network and Operating System Support for Digital Audio and Video*

[9] *http://wiki.multimedia.cx/index.php?title=VMNC*

[10] Brunello, D.; Calvagno, G.; Mian, G.A.; Rinaldo, R., "Lossless Video Coding using Optimal 3-D prediction", *Image Processing. 2002. Proceedings. 2002 International Conference on Volume 1, Issue , 2002 Page(s): I-89 - I-92 vol.1*

[11] R. Oami and M. Ohta, "Efficient lossless video coding compatible with MPEG-II", *Proc. IEEE International Conference in Communications, Vol. 2, pp. 901-905, IEEE, 1998.*

[12] Marc Alzina, Wojciech Szpankowski, Ananth Grama, "2D-Pattern Matching Image and Video Compression", *Data Compression Conference 1998*

[13] Abhayaratne G.C.K, D.M.Monro, "Embedded to lossless coding of motion compensated prediction residuals in lossless video coding", *Proceedings of the SPIE - The International Society for Optical Engineering, v 4310, 2000, p 175-85*

[14] Michael D. Adams, "The JPEG-2000 Still Image Compression Standard (2000)", *citeseer.ist.psu.edu/adams00jpeg.html*

[15] Alan C. Bovik, *"Handbook of Image and Video Processing", pp 3-17.*

[16] Armando J. Pinho, "A Comparison of methods for improving the lossless compression of images with sparse histograms", *Proceedings of the IEEE Int. Conf on Image Processing, ICIP 2002, Rochester, NY, September 2002*

[17] Paulo J.S.G. Ferreira, Armando J. Pinho, "Histogram Packing, total variation, and lossless image compression", *Signal Processing XI - Theories and Applications, Proc. of the 11th European Signal Processing Conf., EUSIPCO-2002, September 2002, Toulouse, France, vol II, pp. 498-501.*

[18] Armando J. Pinho, "An online preprocessing technique for improving the lossless compression of images with sparse histograms", *IEEE Signal Processing Letters, Vol. 9, No. 1, January 2002.*

[19] David Salomon, *"Data Compression: The Complete Reference, 3rd Edition", pp. 262-270.*

[20] David Salomon, *"Data Compression 3rd Edition", pp 262-270.*

[21] Solomon W. Golomb "Run Length Encodings", *IEEE tranasctions on information theory, pp. 399-401, Oct. 1966.*

[22] J. Teuhola, "A Compression Method for Clustered Bit-Vectors", *Information Processing Letters, Vol. 7, pp. 308-311, Oct. 1978.*

[23] ITU - Telecommunications Standardization Sector STUDY GROUP 16 Question 6, Video Coding Experts Group (VCEG), 15th Meeting: Pattaya, Thailand, 4-6 Dec., 2001

[24] *http://www.zlib.net/zlib_tech.html*

[25] Roman Starosolski, "Simple Fast and Adaptive Lossless Image Compression Algorithm", *Volume 37 , Issue 1 (January 2007), Pages: 65 - 91*

[26] Signal Processing and Multimedia Group, University of British Columbia, SPMG/JPEG-LS Implementation, version 2.2, 1999, *ftp://ftp.se.netbsd.org/pub/NetBSD/packages/distfiles/jpeg_ls_v2.2.tar.gz*

[27] Weinberger, M. J.; Seroussi G.; Sapiro G., "LOCO-I: A low complexity, context-based, lossless image compression algorithm". *Proceedings DCC'96, IEEE Computer Soc. Press, Los Alamitos, California, 1996, pp. 140-9.*

[28] Weinberger M. J., Seroussi G., Sapiro G., "The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS" *IEEE Transactions on Image Processing, August 2000, Vol. 9(8), pp. 1309-24*

[29] Ben Greenwood, Lagarith Lossless Codec, *http://lags.leetcode.net/index.htm*

Appendix

Chapter 7

Appendix

7.1 Zlib

Zlib is a free compression library written written by Jean-loup Gailly (compression) and Mark Adler (decompression). To quote from the authors site, Zlib's compression method, an LZ77 variant called deflation, emits compressed data as a sequence of blocks. Various block types are allowed, one of which is stored blocks, which are simply composed of the raw input data plus a few header bytes. In the worst possible case, where the other block types would expand the data, deflation falls back to stored (uncompressed) blocks. Thus for the default settings used by deflateInit(), compress(), and compress2(), the only expansion is an overhead of five bytes per 16 KB block (about 0.03%), plus a one-time overhead of six bytes for the entire stream. Even if the last or only block is smaller than 16 KB, the overhead is still five bytes. In the absolute worst case of a single-byte input stream, the overhead therefore amounts to 1100% (eleven bytes of overhead, one byte of actual data). For larger stream sizes, the overhead approaches the limiting value of 0.03% [24].

Typical compression ratios achieved by Zlib are in the range 2:1 to 5:1. Zlib can be used directly as a library through the DLL provided by the authors. For our implementation, we will discuss how we have used zlib for our purposes.

7.2 Lagarith

Lagarith is a lossless video codec intended for editing and archiving. Lagarith offers better compression than codecs like Huffyuv, Alparysoft, and CorePNG. There are a few lossless codecs that can compress better than Lagarith, such as MSU and FFV1; however Lagarith tends to be faster and stabler than these codecs. Lagarith is able to operate in several colorspaces - RGB24, RGB32, RGBA, YUY2, and YV12. For DVD video, the compression is typically only 10-30% better than Huffyuv. However, for high static scenes or highly compressible scenes, Lagarith significantly outperforms Huffyuv. Lagarith is able to outperform Huffyuv due to the fact that it uses a much better compression method. Pixel values are first predicted using median prediction (the same method used when "Predict Median" is selected in Huffyuv). This results in a much more compressible data stream. In Huffyuv, this byte stream would then be compress using Huffman compression. In Lagarith, the byte stream may be subjected to a modified Run Length Encoding if it will result in better compression. The resulting byte stream from that is then compressed using Arithmetic compression, which, unlike Huffman compression, can use fractional bits per symbol. This allows the compressed size to be very close to the entropy of the data, and is why Lagarith can compress simple frames much better than Huffyuv, and avoid expanding high static video. Additionally, Lagarith has support for null frames; if the previous frame is mathematically identical to the current, the current frame is discarded and the decoder will simply use the previous frame again.

7.3 JPEG-LS

JPEG-LS was developed with the aim of providing a low complexity "near lossless" image compression standard that could be able to offer better compression efficiency than lossless JPEG. JPEG-LS is now the standard of the JPEG committee for lossless and near-lossless compression of still images. Part 1 of this standard was finalized in 1999; and when released, Part 2 of this standard will introduce extensions such as arithmetic coding.

The core of JPEG-LS is based on the LOCO-I algorithm [27][28], that relies on prediction, residual modeling and context-based coding of the residuals. Most of the low complexity of this technique comes from the assumption that prediction residuals follow a two-sided geometric distribution (also called a discrete Laplace distribution). The standard describes a low-complexity predictive and adaptive image compression algorithm with entropy coding using a modified Golomb-Rice family. These context based Golomb-Rice codes, are known to be approximately optimal for geometric distributions. In the JPEG-LS algorithm, we use 3 neighboring pixels for nonlinear prediction, and 4 pixels for modeling. JPEG-LS utilizes the bias cancellation method, also it detects and encodes in a special way smooth image regions. If the smooth region is detected we enter the, so called, run-mode and instead of encoding each pixel separately we encode, with a single codeword, the number of consecutive pixels of equal intensity. We used the SPMG/UBC implementation [26]. In this implementation, some code parts are implemented in 2 variants: one for images of depths up to 8 bits and the other for image depths 9-16 bits.

Besides near lossless compression, JPEG-LS also provides a lossy mode where the maximum absolute error can be controlled by the encoder. Compression for JPEG-LS is generally much faster than JPEG 2000 and much better than the original lossless JPEG standard.

7.4 Binary Code to Gray Code Conversion

Table 7.1 shows the 4-bit Gray codes and their binary equivalent. The construction of Gray codes is done by "reflection" and can be seen in [20]. Given this truth table, we can deduce the logical expression that can achieve this conversion with the use of the Karnaugh Map SOP (Sum of Products method). This method is a way of grouping elements such that one variable can be eliminated, for example, the expression, $C = \overline{A}B + AB$ can be simplified by C = $(A + \overline{A})B$ and, $(A + \overline{A})$ is a tautology, and thus C = B. This idea is used in the grouping done in K-Maps. K-Maps are ordered using Gray codes instead of binary because Gray codes have the property of only 1 bit changing across consecutive numbers, and thus grouping across consecutive cells can lead to a simplification, which was thought of by Karnaugh. Figure 7.1 shows the K-Map simplification of the truth table given in Table 7.1

Simplifying the K-MAP yields:

This thus leads to the logic diagram as shown in Figure 7.2.

Table 7.1: Truth table for 4 bit Gray code to 4 bit Binary Code

Binary Code				Gray Code			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

$$
\begin{aligned}
G_3 &= B_3 \\
G_2 &= B_3 \oplus B_2 \\
G_1 &= B_2 \oplus B_1 \\
G_0 &= B_1 \oplus B_0
\end{aligned}
$$

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

G_0

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

G_1

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

G_2

$B_3B_2 \backslash B_1B_0$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

G_3

Figure 7.1: Binary Gray KMAP

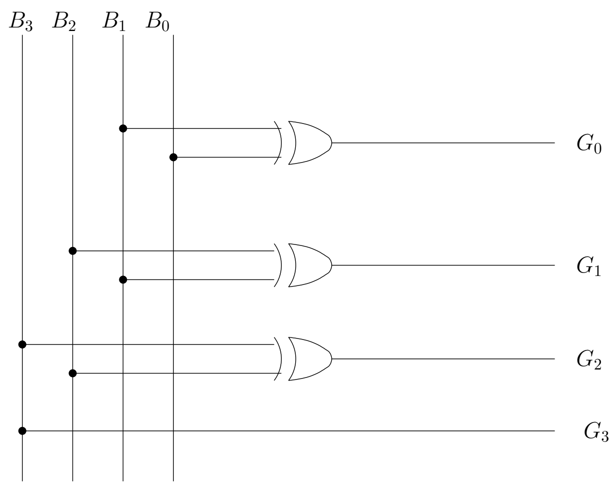


Figure 7.2: Binary to Gray Conversion Logic Diagram

7.5 Gray Code to Binary Code Conversion

Table 7.2 shows the 4-bit Gray codes and their binary equivalent. The construction of Gray codes is done by "reflection" and can be seen in [20]. Given this truth table, we can

Table 7.2: Truth table for 4 bit Gray code to 4 bit Binary Code

Gray Code				Binary Code			
G_3	G_2	G_1	G_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1

deduce the logical expression that can achieve this conversion with the use of the Karnaugh Map SOP (Sum of Products method). This method is a way of grouping elements such that one variable can be eliminated, for example, the expression, $C = \overline{A}B + AB$ can be simplified by C = $(A + \overline{A})B$ and, $(A + \overline{A})$ is a tautology, and thus C = B. This idea is used in the grouping done in K-Maps. K-Maps are ordered using Gray codes instead of binary because Gray codes have the property of only 1 bit changing across consecutive numbers, and thus grouping across consecutive cells can lead to a simplification, which was thought of by Karnaugh. Figure 7.3 shows the K-Map simplification of the truth table given in Table 7.2

Simplifying the K-MAP yields:

$$
\begin{aligned}
B_3 &= G_3 \\
B_2 &= G_3 \oplus G_2 \\
B_1 &= G_3 \oplus G_2 \oplus G_1 \\
B_0 &= G_3 \oplus G_2 \oplus G_1 \oplus G_0
\end{aligned}
$$

This thus leads to the logic diagram as shown in Figure 7.4.

G_3G_2 \ G_1G_0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

B_0

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	1	1
10	1	1	0	0

B_1

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

B_2

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

B_3

Figure 7.3: Gray Binary KMAP

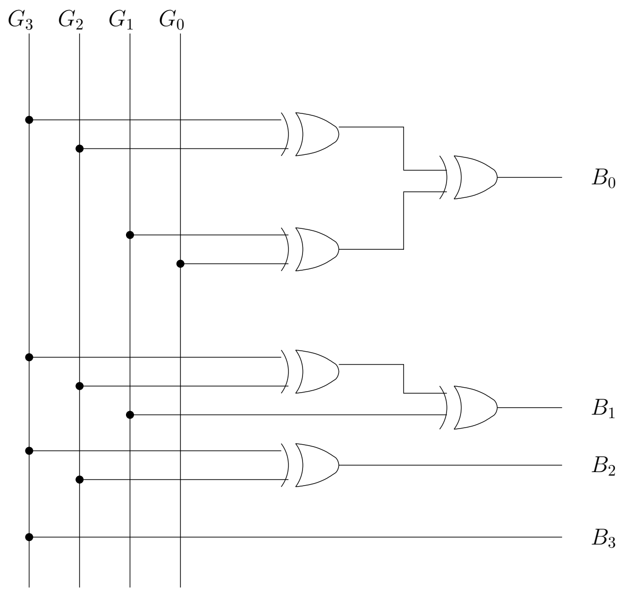


Figure 7.4: Gray To Binary Conversion Logic Diagram

7.6 Searching for the right parameters for JASPER-JPEG-2000 encoding

JPEG 2000 includes a lossless mode based on a special integer wavelet filter (biorthogonal 3/5). JPEG 2000's lossless mode runs more slowly and has often worse compression ratios than JPEG-LS on artificial and compound images. JPEG 2000 fares better than the UBC implementation of JPEG-LS on digital camera pictures. JPEG 2000 is also scalable, progressive, and more widely supported.

The JPEG-2000 standard supports lossy and lossless compression of single-component (e.g., grayscale) and multi-component (e.g., color) imagery. In addition to this basic compression functionality, however, numerous other features are provided, including:

1. Progressive recovery of an image by fidelity or resolution
2. Region of interest coding, whereby different parts of an image can be coded with differing fidelity
3. Random access to particular regions of an image without needing to decode the entire code stream
4. A flexible file format with provisions for specifying opacity information and image sequences
5. Good error resilience. Due to its excellent coding performance and many attractive features,

JPEG 2000 has a very large potential application base. Some possible application areas include image archiving, Internet, web browsing, document imaging, digital photography, medical imaging, remote sensing, and desktop publishing.[14]

7.6.1 Definitions

1. Box : A building block defined by a unique block type and block width. Some boxes may contain other boxes
2. Box Contents : Refers to the data wrapped within the box structure.

3. Box type : Specifies the kind of data that will be stored within the box.
4. Code-Block : A rectangular grouping of wavelet coefficients from the same sub-band of a tile component.
5. Color Component : A component from the codestream that functions as an input to a color transformation system. e.g. Red component, or Y (Luma) component
6. Decomposition Levels : A collection of wavelet sub-bands where each coefficient has the same span with respect to the original samples . These include the HL, LH , HH and for the lowest resolution decomposition levels the, the LL sub-band. Only the LL sub-band can be further decomposed.
7. Discrete Wavelet Transform (DWT) : A transformation that iteratively transforms one signal to two or more filtered and decimated signals corresponding to different frequency bands. This transform operates on spatially discrete samples.
8. Image : The set of all components.
9. ICT : Irreversible Color Transform
10. JP2 file : The name of the file format that is described by the JPEG-2000 standard. Structurally the JP2 file is just a contiguous sequence of boxes.
11. Layer : A collection of coding-pass compressed data from one, or more code blocks of a tile component. Layers have an order for encoding and decoding that must be preserved.
12. MCT : Multi-Component Transform.
13. Packet : A part of the bitstream comprising of a packet header and the coded data from one layer of one decomposition level of one component of a tile.
14. Packet Header : Portion of the packet that describes the layer, decomposition level, component and the code-block segment lengths.
15. Precinct : A sub-division of a tile component, within each resolution used for limiting the size of the packets.

16. Reference Grid : A rectangular array of points to which the images, components, tiles, sub-bands etc. are associated.

17. Tile : A rectangular array of points on the reference grid, registered with and offset from the reference grid origin and defined by a base width and height.

18. Tile Component : All the samples of a given component in a tile. There is a tile component for every component and every tile.

7.6.2 Parameters

A subset of the parameters that JASPER provides for JPEG-2000 encoding are as follows:

1. File Format

2. Reconstructed Image bit depth

3. Tile Size

4. Color Space (multi-component Color Transform)

5. Reversible or irreversible Wavelet Transforms (Lossless or Lossy correspondingly)

6. Number of wavelet resolution levels: This allows us to examine lower frequencies at increasingly finer resolutions, and thereby packing more energy into fewer wavelet coefficients.

7. Precinct size

8. Code-Block Size(The effect of using different code block sizes has no effect on the bpp for a given rate. Code Block size affects the SNR and resolution scalability, and is a part of the EBCOT algorithm)

9. Coefficient Quantization step size

10. Perceptual weights

11. Block Coding Parameters

    (a) Magnitude Coding refinement method

(b) MQ code termination method

12. Progression order

13. Number of Quality layers

14. Region of interest Coding Method

15. Lazy encoding (bypass the arithmetic coding) : in which we put raw bits instead of arithmetic entropy coded bits, and increase error resilience (by avoiding the propagation of error).

In this section, we will see how varying some parameters costs us in terms of bits per pixel, image quality, program execution time. We will also diagram the percentage distribution of different modules in the overall runtime.

File Format

JPEG-2000 allows us to create two types of output file formats, JP2 and JPC (Code Stream format). The JP2 format is useful if we want accurate color representation. For the purposes of our BPP and PSNR analysis, we will use the JP2 file format.

Choosing a Tile Size

In order to find out how tile size is related to the PSNR and BPP, we use the following algorithm.

1. We fix up a resolution level (4 possibilities), and keep other parameters constant.

2. We then choose a tile size.

3. We then plot PSNR Vs Bits Per Pixel(BPP) charts for this chosen tile size.

4. We repeat steps 2 to 4 for all tile sizes

It can be seen from figure 7.5 (c) that when we fix up all other parameters, and vary the tile sizes, then, higher the tile size, larger is the PSNR. Note that larger the PSNR, better is the image quality. Thus, it can be seen that larger tile size leads to better quality. Note that 7.5 (c) is NOT using the multi-component Color transform, and is using thus the reversible color transform. Figure 7.5(d) is the plot that is obtained with MCT, called the ICT (Irreversible color transform) lossy color encoding, and we can see that ICT encoded JPEG 2000 has lower variation in PSNR compared to No MCT versions, and also lesser PSNRs hinting that they lead to more loss.

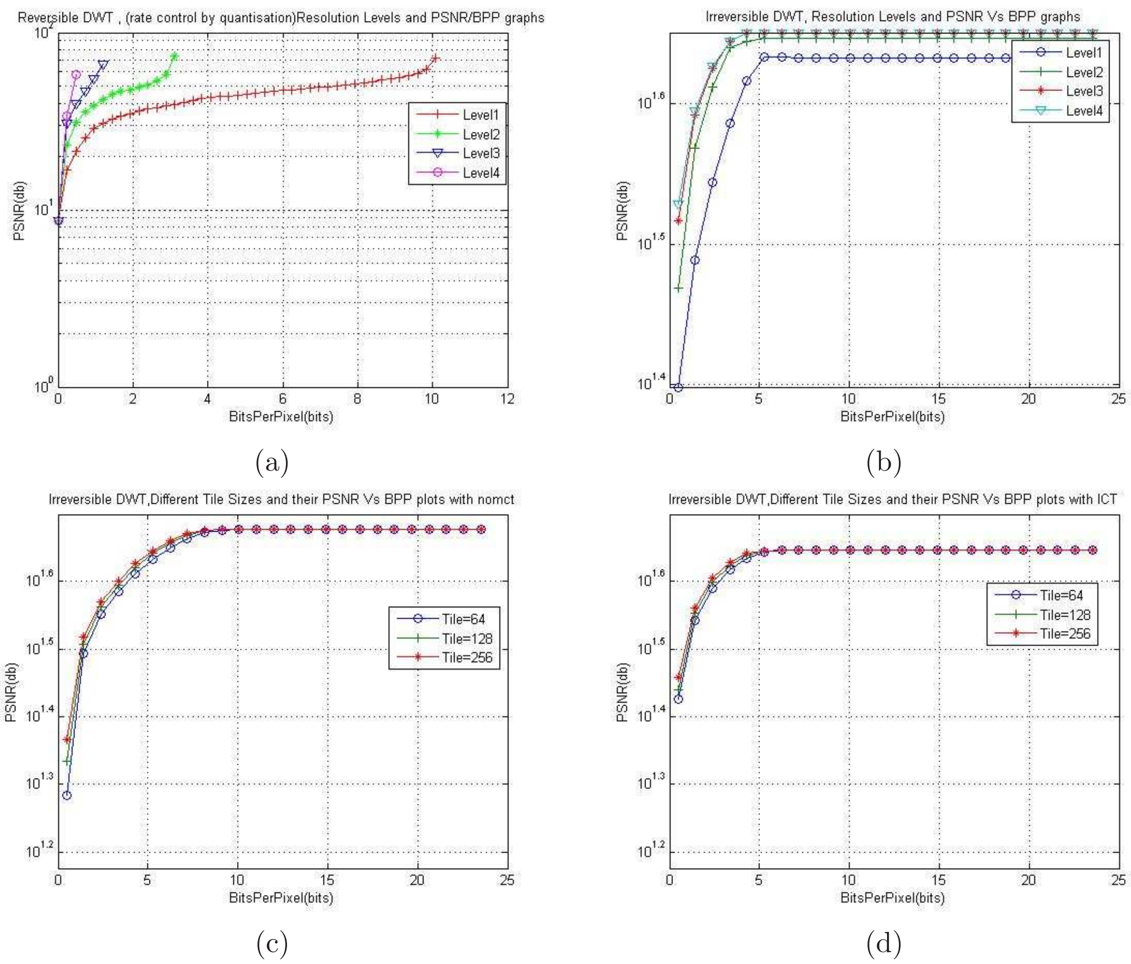


Figure 7.5: (a) Reversible(Lossless) DWT, Resolution Levels and PSNR Vs BPP graph (b) Irreversible(Lossy) DWT , Resolution Levels and PSNR Vs BPP graph (c) Tile size plots with no MCT(fixed Level) (d) Tile plots with ICT (fixed Level)

Lossy and Lossless encoding

While encoding, if the compression rate is not specified as an input parameter, then lossless compression is employed by JASPER. Lossless compression uses the integer, reversible 5/3 mother wavelets. Also, we simply cannot use the irreversible transform to do lossless coding but note that we can use the reversible transform to do lossy coding. This is because, in the second case, the loss could be made to occur through quantization. Figure 7.5 (a) is a plot where we use a reversible DWT. However it should be noted that the plot is actually showing a lossy nature even when we are using the reversible DWT. The loss occurs because we have explicitly asked JASPER to encode at a particular rate, so the required bpp was achieved by appropriate scalar quantization by JASPER. As can be seen in the plot, the PSNR curve has a sharp cutoff at a fixed BPP threshold α and the PSNR goes to infinity for all $bpp > \alpha$, meaning that lossless encoding happens above this threshold. The threshold is different for different levels. The graph for an irreversible DWT is shown in figure 7.5 (b), keeping tilesize, codeblocksize fixed. It was seen that lossless encoding gives a compression of about 10bpp for our test images.

Choosing a DWT resolution level

Graph 1(a) shows that higher the allowed DWT resolution, better is the encoded image quality. We therefore would want to go for a higher resolution encoding. The tradeoff is that it takes longer time to encode, higher the resolution level desired.

multi-component Color Transform

Initially, images have to be transformed (from the RGB color space) to the well known YCbCr color space or to the RCT space (reversible component transform) leading to three components. In lossless mode, a very rough, but reversible approximation of this color space conversion is used. The chrominance components can be, but do not have to be down-scaled in resolution; in fact, since the wavelet transformation already separates images into scales, downsampling is more effectively handled by dropping the finest wavelet scale. This step is called multiple component transformation in the JPEG 2000 language since its usage is not restricted to the RGB color model. The figure 7.5(c) and (d) show how using the MCT can affect our quality and variance of quality. If the -O noMCT is not stated as an explicit parameter, then the ICT (irreversible color transform is applied on the data). Profiling indicates that this is not a very heavy operation but it consumes around 4-5% of the overall computation in the encoding of an image.

Choosing an encoding rate

The choice of choosing a rate(reverse of compression ratio) is mostly dependent on the bandwidth that is available. The tradeoff is with the quality(PSNR) and the total number of cycles of encoding. The following two graphs are useful in determining the relationship of BPP and rate.

1. Reversible DWT, lossy encoding PSNR vs BPP and all levels (Refer Figure 7.5 (a))

2. Irreversible DWT, lossy encoding PSNR vs BPP and all levels (Refer Figure 7.5 (b))

Choosing a code-block size

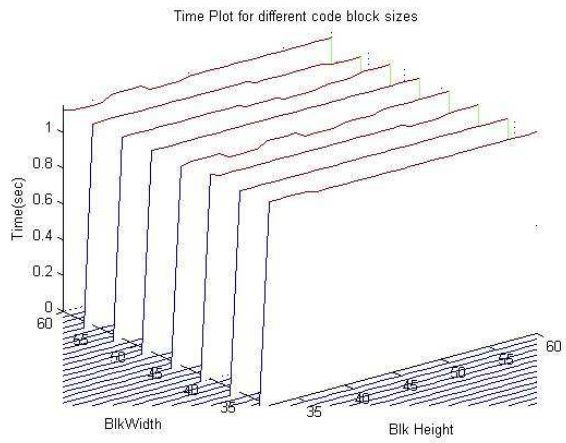


Figure 7.6: Encoding Time(sec) for the Different Code Block Sizes

It was seen from a number of experiments that changing the code-block-size has no effect on the PSNR, for a fixed rate, which is because, the purpose of the code-blocks is not for affecting PSNR when the entire bitstream is available. This is a feature that allows SNR and resolution scalability, meaning that we could test how code-blocks affect our work, when we truncate the bitstreams to arbitrary lengths, and we can check how code blocks affect the ROI coding and the overall PSNR. In those cases, when the bitstream is truncated, code-block should affect the PSNR. That however is out of the scope of this analysis because we would not deal with such scenarios wherein we need to do partial decoding. Figure 7.6is a graph of code block sizes vs the encoding time, keeping all other parameters constant, which also shows that the code-block size affects the overall encoding time upto a very small deviation.

Precincts Size

The quantizer indices for each sub-band are partitioned into code blocks. Code blocks are rectangular in shape, and their nominal size is a free parameter of the coding process, subject to certain constraints, most notably that the nominal width and height of a code block must be an integer power of two, and the product of the nominal width and height cannot exceed 4096.

In tier-2 coding, code blocks are grouped into what are called precincts. Since code blocks are not permitted to cross precinct boundaries, a reduction in the nominal code block size may be required if the precinct size is sufficiently small. In our analysis we are keeping the precinct to be the default size. The code block is also a error resilience tool. The idea being that bit errors in a code block will be restricted to that code block only. This is used to control the size of the packet, and is useful in rate control. Here we are primarily concerned about the bare-bones version with no concern for rate control (achieved through EBCOT). We therefore do not test this parameter.

Lazy encoding

Measurements were done on a sequence of images with the lazy encoding ON and OFF. It was seen that the lazy encoding gave us slightly longer JPEG-2k bitstreams (meaning higher bitrates), but instead was a little faster in the overall encoding time.

7.6.3 Bitrates

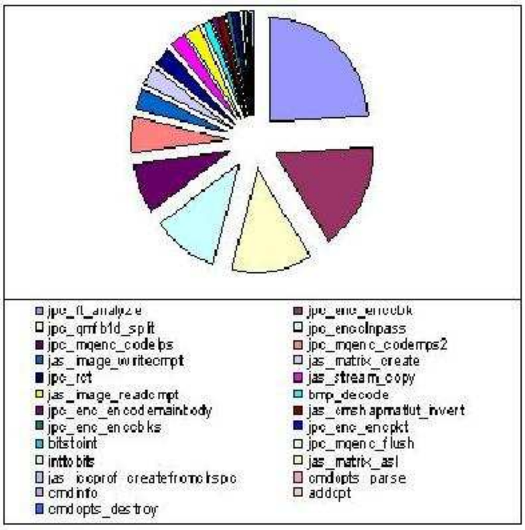
For a typical channel with a bandwidth of 3-400 Mbps, we have the following bitrates achievable. Looking at the discussion about resolution levels, code block size, tile size, rates, we choose the default resolution level, a decent tile size, a fixed code-block length and see that the following bitrates and PSNRs are achieved using JPEG-2000 encoding. In Table 7.6.3 sections are separated by two-coupled-horizontal lines, and the first section is related to lossy encoding, while the second to lossless encoding.

Table 7.3: Different JPEG-2000 parameters and Bitrates

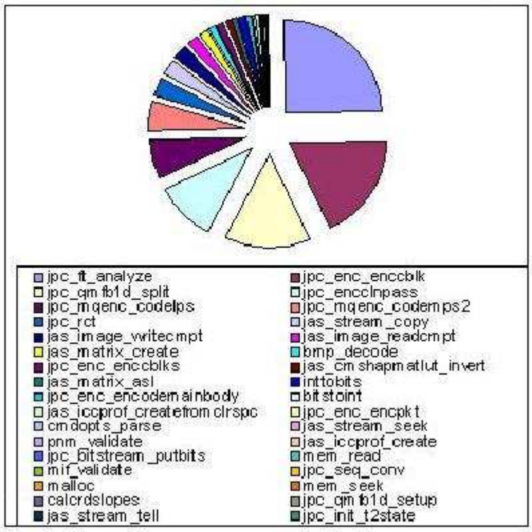
Rate	BPP	PSNR	Level	TileSize	CdBlk	Lazy	ImgSz bits (bpp=24)	Cmprsd img bits/image	Bitrate @5fps	Bitrate @15fps	Bitrate @30fps
0.1	2.4	38.56	6	256	64x64	ON	22118400	2211840	11059200	33177600	66355200
0.15	3.6	41.5	6	256	64x64	ON	22118400	3317760	16588800	49766400	99532800
0.2	4.8	44.04	6	256	64x64	ON	22118400	4423680	22118400	66355200	132710400
0.25	6	45.92	6	256	64x64	ON	22118400	5529600	27648000	82944000	165888000
0.3	7.2	47.12	6	256	64x64	ON	22118400	6635520	33177600	99532800	199065600
0.35	8.4	47.60	6	256	64x64	ON	22118400	7741440	38707200	116121600	232243200
0.4	9.6	47.631	6	256	64x64	ON	22118400	8847360	44236800	132710400	265420800
0.45	10.8	47.63	6	256	64x64	ON	22118400	9953280	49766400	149299200	298598400
0.5	12	47.63	6	256	64x64	ON	22118400	11059200	55296000	165888000	331776000
0.55	13.2	47.63	6	256	64x64	ON	22118400	12165120	60825600	182476800	364953600
0.6	14.4	47.63	6	256	64x64	ON	22118400	13271040	66355200	199065600	398131200
0.65	15.6	47.63	6	256	64x64	ON	22118400	14376960	71884800	215654400	431308800
0.7	16.8	47.63	6	256	64x64	ON	22118400	15482880	77414400	232243200	464486400
0.75	18	47.63	6	256	64x64	ON	22118400	16588800	82944000	248832000	497664000
0.8	19.2	47.63	6	256	64x64	ON	22118400	17694720	88473600	265420800	530841600
0.85	20.4	47.63	6	256	64x64	ON	22118400	18800640	94003200	282009600	564019200
0.9	21.6	47.63	6	256	64x64	ON	22118400	19906560	99532800	298598400	597196800
0.95	22.8	47.63	6	256	64x64	ON	22118400	21012480	105062400	315187200	630374400
1	24	47.63	6	256	64x64	ON	22118400	22118400	110592000	331776000	663552000
0.1	2.4	38.54	6	256	64x64	OFF	22118400	2211840	11059200	33177600	66355200
0.15	3.6	41.56	6	256	64x64	OFF	22118400	3317760	16588800	49766400	99532800
0.2	4.8	44.083	6	256	64x64	OFF	22118400	4423680	22118400	66355200	132710400
0.25	6	46.00	6	256	64x64	OFF	22118400	5529600	27648000	82944000	165888000
0.3	7.2	47.14	6	256	64x64	OFF	22118400	6635520	33177600	99532800	199065600
0.35	8.4	47.59	6	256	64x64	OFF	22118400	7741440	38707200	116121600	232243200
0.4	9.6	47.62	6	256	64x64	OFF	22118400	8847360	44236800	132710400	265420800
0.45	10.8	47.62	6	256	64x64	OFF	22118400	9953280	49766400	149299200	298598400
0.5	12	47.62	6	256	64x64	OFF	22118400	11059200	55296000	165888000	331776000
0.55	13.2	47.62	6	256	64x64	OFF	22118400	12165120	60825600	182476800	364953600
0.6	14.4	47.62	6	256	64x64	OFF	22118400	13271040	66355200	199065600	398131200
0.65	15.6	47.62	6	256	64x64	OFF	22118400	14376960	71884800	215654400	431308800
0.7	16.8	47.62	6	256	64x64	OFF	22118400	15482880	77414400	232243200	464486400
0.75	18	47.62	6	256	64x64	OFF	22118400	16588800	82944000	248832000	497664000
0.8	19.2	47.62	6	256	64x64	OFF	22118400	17694720	88473600	265420800	530841600
0.85	20.4	47.62	6	256	64x64	OFF	22118400	18800640	94003200	282009600	564019200
0.9	21.6	47.62	6	256	64x64	OFF	22118400	19906560	99532800	298598400	597196800
0.95	22.8	47.62	6	256	64x64	OFF	22118400	21012480	105062400	315187200	630374400
1	24	47.62	6	256	64x64	OFF	22118400	22118400	110592000	331776000	663552000
0.41	10.05	INF	6	256	64x64	ON	22118400	9263899.064	46319495.32	138958486	277916971.9
0.41	10.051	INF	6	256	64x64	ON	22118400	9263899.064	46319495.32	138958486	277916971.9

7.6.4 Profiling Analysis

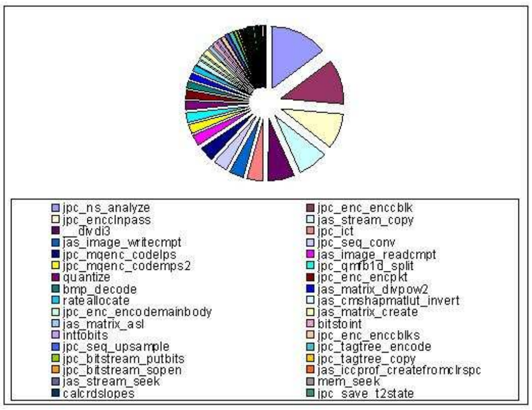
In order to measure the performance of the different modules in JASPER, we did profiling using gprof and collected the profile of every module in an execution. The aggregate analysis was done over multiple images that were part of a video sequence. Figure 4 is an indicator of how Jasper behaved over multiple images. In order to see what these percentages



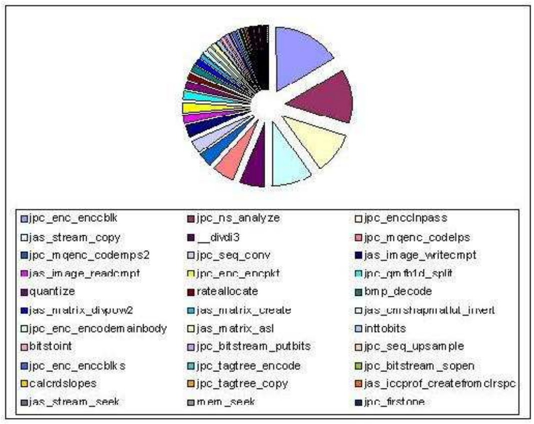
(a)



(b)



(c)



(d)

Figure 7.7: (a) Lossless with MCT (b) Lossless without MCT (c)Lossy with MCT (d) Lossy with no MCT

mean, we must know how gprof works. The output from gprof gives no indication of parts of the program that are limited by I/O or swapping bandwidth. This is because samples of the program counter are taken at fixed intervals of the program's run time. Therefore, the time measurements in gprof output say nothing about time that the program was not running. For example, a part of the program that creates so much data that it cannot all fit in physical memory at once may run very slowly due to thrashing, but gprof will say it uses little time. On the other hand, sampling by run time has the advantage that the amount of load due to other users won't directly affect the output timings that we get. Consequently, because gprof works on sampling the program counter every few units of time, the profile that we get is not affected by the thrashing etc , which is a desirable property.

In general it can be seen that the Lossless DWT variants, spend most of their time (around 25%) doing the jpc_ft_analyze, and the lossless variant spends most of its time in the jpc_ns_analyze, followed by the block encoding. The functions given above are described in the table that follows. Also, DWT uses a sub-band coding technique called Quadrature Mirror Filter-Image Filter Bank (QMFB), and is represented by a module (jpc_qmfb.c).

Category	Encoder Routines
Utilities	Bitstoint() inttotobits() jas_malloc() Pgx_wordtoint() pgx_inttoword()
Image Transfer	jas_image_writecmpt() jas_image_readcmpt()
Coefficient Bit Modeling	jpc_encsigpass() jpc_encrefpass() jpc_encclnpass()
Wavelet Transforms	jpc_qmfb1d_split() jpc_ft_analyze() jpc_ns_analyz()
Arithmetic Entropy Encoding	jpc_mqenc_codelps() jpc_mqenc_codemps2()

7.6.5 Conclusion

1. Higher the wavelet decomposition level better the PSNR.

2. Compression Rate affects PSNR, and under our conditions, i.e. with entire bitstream

available with no errors, and decoder decoding the entire stream, the code-block size doesnt.

3. We saw how changing the tile size directly affects the PSNR. Higher the tile size, better the PSNR with the best case being that the entire image is one tile size.

Thus, in our SBB-NR-CR-JPEG-2000, we use the parameters as shown in Table 7.6.3