University of Stuttgart

Faculty of Computer Science, Electrical Engineering and Information Technology

Masterarbeit Nr. 3254

Memory-efficient Lossless Video Compression Using Temporal Extended JPEG-LS and On-line Compression

Debasish Chanda

Course of Study:	INFOTECH
Examiner:	Prof. Dr. Sven Simon
Supervisor:	M.Sc. Zhe Wang
Commenced:	February 01, 2011
Completed:	October 31, 2011
CR-Classification:	B.3.0, E.4, I.3.1, I.4.2

Institut für Parallele und
Verteilte Systeme
Abteilung Parallele Systeme
Universitätsstraße 38
D-70569 Stuttgart

Acknowledgement

I would like to thank Zhe Wang and Simeon Wahl for their continuous supervision and guidance with rigorous reviews and inspiring suggestions throughout my work on this thesis. I would also like to thank professor Sven Simon for providing me the opportunity to contribute to such a research field, making one of my most important achievements.

I am eternally grateful to my parents for their constant encouragement and support, without which this work would not have been completed.

Abstract

Use of temporal predictors in lossless video coders play a significant role in terms of compression gain, but comes with a cost of significant memory requirement since this approach requires to save at least one frame in buffer for residue calculation. An improvement to standard JPEG-LS based lossless video coding algorithm is proposed in this work which requires very small amount of memory comparing to the regular approach keeping the computational complexity low. To obtain a higher compression, a combination of spatial and temporal predictor model has been used where appropriate mode is selected adaptively on a pixel based analysis. Using only one reference frame, the context based temporal coder performs its calculation regarding mode selection and prediction error calculation with already reconstructed pixels. This method eliminates the overhead of transmitting the coding mode in the decoder side. The need for storage space to save the only reference frame is further reduced by introducing on-line lossy compression on that frame. Relevant pixels from the stored reference frame are obtained by partial on-the-fly decompression. The combination of temporally extended context based prediction and on-line compression achieves a significant gain in compression ratio comparing to standard frame-by-frame JPEG-LS video coding keeping the memory requirement low, making it usable as a lightweight lossless video coder for embedded systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Need for a new video coder

In digital media systems, there is a constant need for improvement of video coders since the concept of image and video encoding is considered as one of the essential concepts which has a profound impact in today's commercial, technical and social life. The continuous application of advanced mathematics and other statistical analysis into the media science creates a variety of possibilities of encoding performance enhancements in various fields of application. In general, amount of compression becomes highly scalable depending upon the implementation of appropriate mathematical and algorithmic models in relevant applications, making things interesting from a research point of view. On the other hand, over the last few decades, digital image and video coding systems have transformed from pure academic research works into massive commercial use which also made coding standards evolve more according to specific fields of interests. Many technologies and standards are available today for such wide range of applications including medical imaging, digital HDTV broadcasting, multimedia database service, video on demand and so forth; and still growing. This combination of persistent commercial demand and compelling academic topic is sufficient to make it evolve even further.

Among the available primary coding techniques[10][11], lossless video coders have specific needs in both research and commercial purpose where cost of storage and transmission becomes less significant comparing to the need for available details in video frames. Example applications can be divided into 3 basic catagories: (1)molecule and medical imaging, image archiving, military purposes where details of every pixel is required for a successful analysis, (2)Computerized applications where analysis is done by something other than human eyes and where the term "perceptually lossless" does not make any sense, and (3)applications where subject is to be encoded repeatedly (studios for example) where lossy encoding means an accumulation of errors over each iteration of encoding. These applications require a lossless reconstruction of data since tiny difference between pre and post compressed media is enough to alter the application's reliability of outcome.

Over past few decades the use of embedded systems has grown exponentially since it offers a very good integration of a multi modular device designed to perform a specific task. The lossless video coders are also subject to be embedded into other systems as a compression tool for storage and retrieval. Unlike standalone machines, embedded systems always suffer from lack of space and higher memory cost and therefore, things get more challenging for lossless video coders. So memory efficiency becomes an important issue considering the complexity and cost.

This work aims at designing a lossless video coder with competitive speed and compression performance, taking memory constraints of embedded systems into consideration. The achievement of this work will bridge the gap between high performance lossless video coders and lightweight coders with much lower compression gain. Designing such a system is challenging because higher compression performance of lossless video requires a larger amount of memory. The term lossless consumes most of the freedoms of being memory efficient since it can not lose information which is the case for lossy coders with a very high compression ratio. Still, usage of proper analysis into the problem and adoption of relevant solutions are done and tested for better compression gain with memory efficiency.

1.2 Organization of this thesis

This work is separated into some basic parts. Chapter 1 (this) is the introductory chapter where the motivation and the project objective is described. Some background theory and concepts relevant to this work are described in Chapter 2. In Chapter 3, a theoretical proposal of the proposed model is introduced with relevant explanation for the approached sub-solutions. The outline of the model is also presented in that chapter. Detailed description of the two basic modular upgrade over basic model are explained in the following two chapters; where Chapter 4 focuses the improvement on the predictor model and Chapter 5 explains the proposed way of obtaining memory efficiency. The desired outcome of the proposed system is analyzed in using several test inputs and performance in terms of compression ratio and memory efficiency is measured using several approaches in Chapter 6. Detailed environment setup and implementation strategy are also included there. Finally Chapter 7 illustrates the overall outcome of this thesis, pointing out the possibilities of foreseeable improvements.

Chapter 2

Background

Before jumping into details, some background theories and concepts are to be reviewed to have a through understanding of the proposed approach. Since this system deals with several compression algorithms, outline of these are also an issue to discuss. This chapter is dedicated to serving this purpose and idea behind usage of some of the procedures are explained later in Chapter 3.

2.1 Video compression and lossless video coders

The term video compression refers to the reduction of data amount that represents the digital video. Practically video is a sequence of images or frames that interpret some temporal events. Video compression is the method to compress the video data exploiting the intra and/or inter frame redundancy of those sequences. The goal of such compression is to save space and to reduce amount of bandwidth in video storage and communications. Over the last few decades, usage of digital video has raised exponentially and thus video compression has become one of the key topics among the academic researchers and commercial users.

Like the other multimedia compression strategies, video compression can also be lossy and lossless. As the name depicts, lossless videos are those where each pixel of any frame can be reconstructed without losing any information. Generally video codecs use lossy algorithms as these give better compression and amount of small losses in a video are hard to recognize by human eyes. But there are cases where detailed analysis of frames are to be done for further usage, and storage or transmission cost becomes less significant comparing to need of details. As a result, it has been an important issue in various applications where videos are subject to further processing, intensive editing and archiving.

As the working ground of the proposed video coder the lossless mode JPEG-LS is used and therefore a detailed description of this coding is presented in the next section. Unless otherwise stated, this description is based upon [1] and [4].

2.2 JPEG-LS as a Lossless Encoder

JPEG-LS[4] standard is aimed for a very high compression ratio maintaining the implementation complexity very low. The core of this standard is based upon the LOCO-I (LOw COmplexity LOssless COmpression for Images)[1] algorithm which combines design simplicity and compression gain in a very well formed manner. This approach offers both lossless and near lossless approaches of compression for continuous tone images. High compression gain is achieved using a simple and less complex implementation of the universal context modeling paradigm[5].

In terms of compression ratio, it is within the range of few percentages comparing to very sophisticated algorithms (CALIC[16] for example) and all of these done keeping the computational complexity very low. The structural simplicity of the design is very useful for the researchers to add up extra features. As a result, it has become a very suitable tool for image/video compression researchers around the world. For the same reason, JPEG-LS has been selected as the foundation of this work as well.

As we said in the previous section, video is a sequence of still images. Because of its competitive performance and low computational complexity, a natural first step of this thesis is to investigate the compression of video on a frame-by-frame basis using JPEG-LS as a framework. In later chapters we will try to improve compression performance / memory efficiency within this framework using lossy coders. Therefore it is important now to have a basic understanding of JPEG-LS.

2.2.1 Basic block diagram

JPEG-LS consists of two fundamental blocks, namely Modeling and Coding. Figure 2.1 shows the simplified modular block diagram separating working entities into those modeler and encoder parts.

2.2.2 JPEG-LS modular description

The modeling of JPEG-LS is performed using the template illustrated in Figure 2.2. Here, x denotes the current pixel for which prediction value is being generated and a, b, c, d are the neighboring samples relative to x upon which the prediction modeling is depended. For the rest of this section, the letter notations a, b, c, d and x will denote both the positions and the magnitudes of pixels of their respective positions for simplicity. To work on the current pixel x, pixel values from left and top of that point are used for modeling and prediction as the scanning order supports the fact that those helping pixel values are already known for that certain point of time. Only

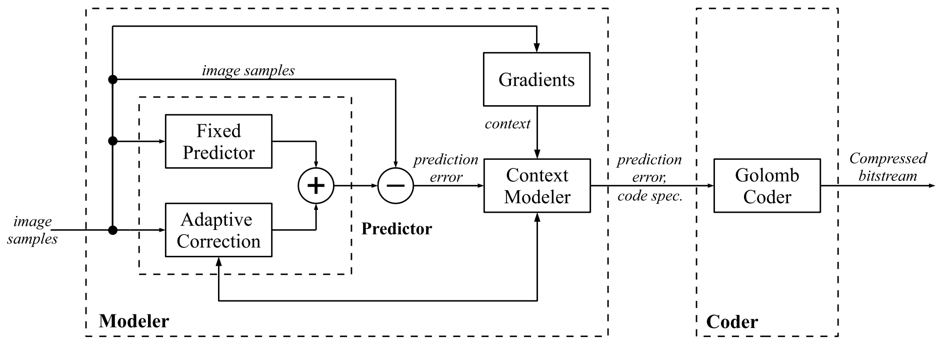

Figure 2.1: Simplified JPEG-LS block diagram with regular mode

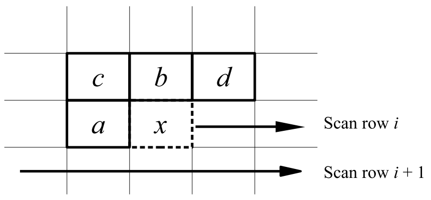

Figure 2.2: Context Modeling Template.

problem arises for the samples of very first row and column. For JPEG-LS, upper neighbors ($b,\ c,\ d$) for the first row are assumed zero and the for the vertical order positions, value of a and d are assigned as the values of b.

Prediction

In this step, a value x'_t is guessed denoting the actual pixel value x_t. This guessing is performed using the values obtained from the previous samples x'_{t-1} from the same image. In general, this guess or prediction may be performed using any kind of fixed scheme and should be able to perform edge detection for best results. However, accuracy of predicted value comes with a price of calculation complexity which opposes the main goal of JPEG-LS. Still, a simpler but effective version of edge detection is made possible to use in JPEG-LS. A fixed predictor loaded with a very basic vertical and horizontal edge check serves as the predictor in this standard. To be more specific, the model predicts the value (*PError*) using the following equation:

$$PError = \begin{cases} min\,(a,b) & \text{if } c \geq max\,(a,b) \\ max\,(a,b) & \text{if } c \leq min\,(a,b) \\ a+b-c & \text{if otherwise} \end{cases} \tag{2.1}$$

This model is a simple combination of three fixed predictors and a dynamic switch is used to choose the best possible one guessing the probable occurrences of edges. It tries to take the value of a if horizontal edge is detected and tries to predict a value close to b in case of a vertical one. In absence of any possible occurrence of edge, the value a+b-c is selected as the predicted value. In absence of corners, this ensures the expected flat behavior since it calculates a possible similar value depending upon the neighboring samples. This predictor model is also known as the Median Edge Detector (MED) predictor because of its behavior of combination of edge detection and median value calculation for possible non-edge pixels.

In practice, this model does not deliver the accuracy comparing to other state of the art models, but it ensures an optimal value considering the amount of computational complexity and hardware requirement. Still some more correction of this predicted value is done later in terms of context modeling which is discussed in the next section.

Context modeling

A simple context model is used in JPEG-LS to extend its efficiency by error value correction and encoding parameter optimization. Data on the context of a specific pixel is used to improve the probabilistic model to correct the predicted pixel magnitude. The context model used here is established using quantized local gradients. The whole process is divided into three fundamental steps: (1)Parameterization, (2)Context determination, and (3)Adaptive correction.

1. Parameterization: Two sided geometrical distribution (TSGD) centered at zero is capable of modeling the statistical data of residuals obtained from a fixed predictor used upon continuous toned images very well[2]. According to this model, the probability of any value of prediction error ϵ is proportional to $\theta^{|\epsilon|}$. Here θ is the controller of the exponential decay rate; mathematically $\theta \in (0, 1)$. But in context based error signals, a DC value is typically present; which is obvious because of the presence of possible bias in prediction step and other statistical constraints, presence of non-integer values for instance. To control this phenomenon, addition of a new parameter μ makes things more suitable for the model. If μ is capable of taking non integer values, it can capture the context based prediction errors very efficiently. If we split this prediction offset value into a whole bias (R) and a fractional shift (s) part so that $\mu = R - s$, we can state the TSGD parametric class P for each context as

$$P_{\theta,\mu}(\epsilon) = \frac{(1-\theta)}{(\theta^{1-s} + \theta^{s})}\theta^{|\epsilon - R + s|}, \quad \epsilon = 0, \pm 1, \pm 2, ... \tag{2.2}$$

In the predictor, R is an integer adaptive term which is to be tuned out by the prediction model. This tuning out tends to keep the average residual values within the range of 0 and -1 in the probability distribution model.

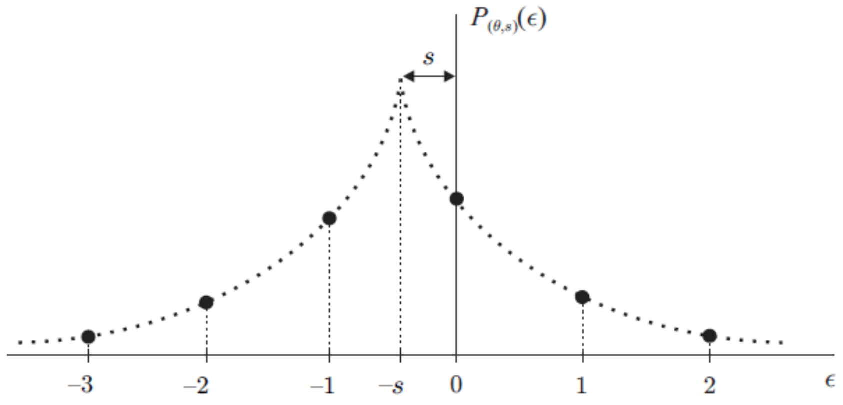

Figure 2.3: Two-sided geometric distribution[1].

Using this theory, the procedure of error correction is introduced in JPEG-LS. After prediction error correction the Equation 2.2 is further reduced to

$$P_{\theta,\mu}(\epsilon) = \frac{(1-\theta)}{(\theta^{1-s}+\theta^{s})}\theta^{|\epsilon+s|}, \quad \epsilon = 0, \pm1, \pm2, ... \tag{2.3}$$

This equation is illustrated in Figure 2.3 which shows a TSGD centered at zero. Considering the X axis value as s, if $s = 1/2$, P resembles as bi-modal distribution. This figure shows the dependency upon context parameters very well. The coding scheme used in JPEG-LS matches this new reduced variable range and so this value is further used in the encoding module.

2. Context determination: In JPEG-LS context parameter is defined using the surrounding properties of the working pixels. These properties are obtained using the local gradients that captures the statistical behavior of the prediction error. Mathematically, the gradients are defined as $g1 = d-b$, $g2 = b-c$ and $g3 = c-a$.

Each gradient value obtained from the neighboring pixels are quantized for further model size reduction. The quantized values are then converted into some smaller numbers of equal probabilities regardless of the context parameter values. These numbers are indexed maintaining an integer range of T such as: -T, . . . , -1, 0, 1, . . . , T keeping the symmetry of the numbers, which defines the “regions” in context term. As a result, this offers a total of $(2T+1)^3$ numbers of equal probable regions to be modeled. In practice, this number can be also reduced by handling the sign of the number more carefully. For negative context numbers, the sign of the error values are inverted before encoding. As the decoder measures the context parameters in the same manner, this sign toggling is captured casually while decoding; resulting no errors in the decoder side. So just by taking care of the context values of the opposite signs, the total number of context becomes $[(2T+$

$1)^3 + 1]/2$, almost half the size we needed before. In JPEG-LS the value of T is set to 4, resulting a total of 365 contexts. As the storage requirement for implementation of such models is equivalent to this number, this leads to a quite optimized outcome in terms of space complexity.

3. Adaptive correction: The predictor model used in JPEG-LS is fixed MED predictor which introduces some bias considering the actual value. As discussed before, the adaptive part of the predictor is designed to cancel the integer part of the offset (R) using the context parameters.

Typically the value of R can be estimated using the median of the prediction errors obtained in a particular point of scanning / encoding. However, this general procedure requires additional storage for saving required data for exact median value calculation. As a result, a smarter approach is adopted to calculate the value equivalent to R. In practice, using the cumulated sum of the prediction error (D) and the total number of occurrence of that particular context (N) gives an usable average value that resembles that fixed offset. Using this, a correction value (*C'*) is calculated as

$$C' = \lceil (D/N) \rceil \tag{2.4}$$

and added to the error value obtained from that relevant pixel magnitude.

In spite of being simple, this approach comes with some two major setbacks as well. First problem comes due to the division of the values as this cost severe computational effort. And secondly, this value of estimated average can deviate from accuracy in occurrence of unexpectedly "large" prediction error magnitudes. These problems can be avoided by changing the equation (2.4) into

$$D = N.C + B \tag{2.5}$$

Here, B remains in a specific range ($-N < B \leq 0$). Values of, B and C are stored and updated maintaining the constraints of that range in each context. In practice, B is added to the corrected error value, followed by arbitrary number of addition of subtraction with N until the resultant value is in that required range. The update of C is performed considering that number of addition/subtraction of N in the previous step.

In JPEG-LS, the above mentioned procedure of adaptive correction is modified further. Here, the number of addition/subtraction is limited to one per update and if required, the value of B is forced to be in the appropriate range as discussed above. In Listing 2.1, the pseudo-code of this bias computation is shown where *B'* and *C'* is substituted by the estimated values of B and *C*. Initial values of B and C are set to zero and a division free update is possible using this procedure.

Listing 2.1: Bias cancellation procedure

```
B = B + error;  // Prediction error accumulation
```

```
N = N + 1;        // Context occurrence counter

/* Update bias register and error correction value */
if (B <= -N){
    C = C - 1;
    B = B + N;
}
else if (B > -N){
    C = C + 1;
    B = B - N;

    if ( B > 0 )
      B = 0;
}
```

Here we can see that if the value of C fails to remain in desired range, it is incremented/decremented, followed by further adjustment of B with respect to the change of C. If this adjustment still fails to keep B in range, it is forced to do so. This controlled update prepares C to be equivalent to the value of R which is used to correct the prediction error. For the record, C is also clamped within the range of (-128, 127) for space requirement reduction.

To sum up, by using this approach, the adaptive correction of the prediction error is now estimated just by storing and updating few variables for each context doing some basic calculation maintaining a very low memory requirement.

Coding

JPEG-LS standard uses limited length Golomb-Rice codes for regular mode coding. As seen in the previous section, the corrected prediction residuals resemble the TSGD (Figure 2.3), as in presence of more zero values around the center and larger values in two sides. Golomb-Rice coding handles such distributions very efficiently and with an extra advantage of independence of code tables, which is common among the similar prefix coders.

The working procedure of Golomb coders is very simple as well. To encode, it divides the positive integer value N(which has to be encoded) in two main parts, q and r with the help of another tunable non-zero positive integer parameter M. q is the quotient parameter obtained by dividing N by M ($\lfloor N/M \rfloor$) and unary coding is applied to it. Then the remainder part r is calculated by doing a mod operation of N and M ($N\%M$) and thereafter coded by using binary representation. Code length of r is kept as $\lfloor \log M \rfloor$ bits if $N \leq 2^{M'} - M$ or simply kept as M bits if otherwise happens where $M = \lceil \log M \rceil$. In JPEG-LS, the parameter M is tuned as $M = 2^k$ to coupe properly with the TSGD model, yielding a code length of $k + 1 + \lfloor N/2^k \rfloor$

Theoretically, The parameter space in TSGD is divided into some spatial regions and for each region there is a magnitude of M yielding the minimum possible code length for that specific region using Golomb-Rice code. In JPEG-LS, this rule is used in a much simplified form. For each of 365 contexts, along with other parameters for context updates, the summation of prediction errors are also stored in a dedicated register A and value of k can be defined in a single line code

$$for\,(k = 0; (N << k) < A; k + +)\,;$$

This on-line adaptive context depended selection of Golomb parameter k ensures a very good code length optimization, resulting nearly similar performance of arithmetic coding[1].

2.2.3 Summary

To summarize, the lossless approach of JPEG-LS uses the combination of MED prediction, context-based error correction and context-based adaptive Golomb coding to produce a very good compression gain. Its lightweight but efficient fixed predictor is capable of edge detection in an optimal way. The context model uses the image data in a very efficient way using state of the art statistical method. By using only the already scanned adjacent pixel densities, this model offers a well optimized adaptive correction of the prediction residuals which keeps the sum-of-error of the whole image lower. Comparing to the complexity and usability of this model, the memory requirement and number computational units are kept very low with some clever modifications in implementation model. The coding model, which is based on the extension of the Golomb coder family, also takes the best advantage of the context model, resulting significant reduction in average code length.

Along with lossless mode of JPEG-LS as the base of the proposed system, usage of lossy codecs are also subject of discussion to complete the system structure; which is the topic of the next section.

2.3 Lossy coders

Unlike the lossless coders, lossy coders compromise the quality of the image to obtain better compression. The amount of loss or distortion is depended upon the efficiency of the encoding algorithm and requirement of the user application.

Typically these encoders use some transform coding where a block of image data are transformed into another domain to find out relevant similarities. On that transformed domain, redundancy can be used in a greater scale and be quantized thereafter. Another approach for coding lossy images is performed by calculating residue based on some prediction algorithm and quantizing the error value. These quantized values are then coded or entropy coded depending upon the coding criteria. The common procedure of "quantization" is responsible for the loss of the image quality, as well as the provider of a higher compression gain.

The proposed design consists of lossy coders to improve the memory module for its higher compression capabilities. This design supports adoption of arbitrary lossy coders in the relevant module. However, the overall outcome of the system is still a subject of discussion which will be covered in the later chapters. To test this proposed approach, three widely available lossy encoders are selected, namely JPEG, JPEG2000 and near lossless mode of JPEG-LS. Concise description of each of these are given below.

2.3.1 JPEG

JPEG[26] (acronym for Joint Photographic Expert Group) is one of the most popular lossy compression standard for continuous tone still images. The term "joint" in JPEG refers to the co-operation of three international standard organizations, namely International Organization for Standardization (ISO), and the International Electrotechnical Commission (IEC) and International Telegraph and Telephone Consultative Committee (CCITT). The goal of this group was to standardize the image compression format in a common platform which was a requirement in late 1980s and JPEG compression format is one of the greatest achievements of that effort that can be seen today.

This standard defines several compression formats to satisfy the requirements of wide variety of use cases. Among those, the format known as the "Baseline JPEG", a subset of lossy DCT (Discrete Cosine Transform) based mode of operation matches the relevance to be integrated to the proposed system for its simple lossy coding capabilities and unmatched popularity. A concise description of this standard is therefore described below.

Outline of Baseline JPEG

The basic layout of baseline JPEG encoder is illustrated in Figure 2.4. A brief description of each of the basic blocks follows:

Encoder model The encoder model of JPEG is responsible for transformation of the input images so that the encoding procedure becomes more abstract and suitable for further processing. Here, the input image is partitioned into non-overlapping 8X8 pixel blocks. Values of each block is then converted into signed integer and a block based DCT operation is performed thereafter. This transformation converts each block into values with representation of spatial frequencies. As a result, for each block, the pixel values with lower spatial frequencies are separated from those with the higher ones. Since the most of the usable image data are stored in the lower frequency

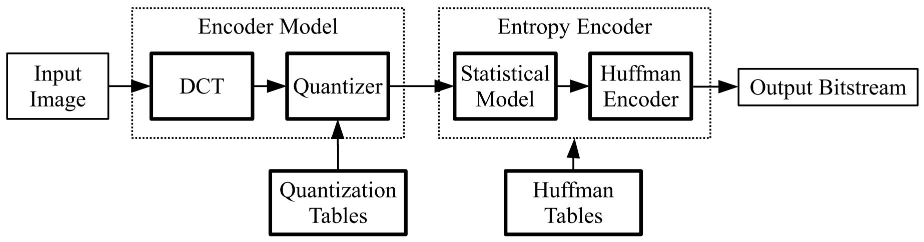

Figure 2.4: JPEG encoder model

zone, these can be easily identified by this transformation and separated from the data with higher frequencies, which becomes close to zero after the transformation.

This transformation eventually separates the important data into less significant ones, which offers the opportunity to discard some of the visually redundant details, which is done by quantization. This is performed using a predefined quantization table. However, custom tables are also possible to use for specific applications which gives the freedom to choose the final image quality by the user group. This standard becomes a "lossy" one because of this adoption of selection through quantization, resulting a very good compression ratio. The data in the output matrix are named as the quantized DCT coefficients, that are subject to be processed further for entropy coding.

Entropy encoder Within DCT matrix blocks the important data are located in the upper left corner keeping the less important ones on the other side. To get the best out of this structure these values are scanned in a zig-zag order to preserver the importance of the pixels in a sequential 1 dimensional order. The very first coefficient in each block is called DC coefficient which has the value with the highest magnitude. This value is coded using a DCPM (Differential Pulse Code Modulation) by comparing to the DC coefficient of the previous block, which generally contains a small or no difference in magnitude comparing to the present one. The other 63 coefficients, known as AC coefficients, with smaller (mostly zero or close to zero) values are coded using run length coding technique.

The entropy coder uses Huffman algorithm with predefined or user defined Huffman table. To perform optimized encoding the DC and AC coefficient coding procedure use separate tables for their different statistical behavior. The outcome of the coder is a series of encoded bit stream. These are saved into storage media along with embedded quantization Huffman tables that are to be used by the decoder whenever needed.

Although JPEG suffers from the limitations of losing image details, it offers the users to choose the quality factor which defines the amount of loss of quality and more importantly the amount of final file size. As a result users have the ability to tune the required quality comparing to the estimated savings of storage space. The decoding procedure of JPEG is the reverse of encoding flow with simple inversion of each of the blocks. Detailed description of this standard along with available file formats are available on [26] and [31] for further understanding.

2.3.2 JPEG2000

Another codec used for the lossy compression in this work is the JPEG2000[14], developed by the JPEG community (see previous section) to outperform the capabilities of standard JPEG. Adoption of advanced transformation algorithms and coding structures, in conjunction with various dimensions of usability has made it a very suitable standard for users of various domains.

Basic working procedure A basic block diagram of JPEG2000 encoder and decoder is illustrated in Figure 2.5. A very brief description of the working procedure of this standard relating to the block diagram is stated below.

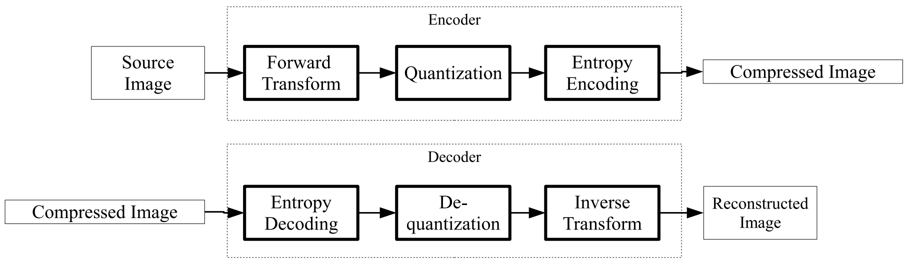

Figure 2.5: JPEG2000 encoder and decoder

Preprocessing The sample image has to be undergone some preprocessing before getting into the transformation engine. The image samples are level shifted to have an weighted average of zero. JPEG2000 allows an optional inter-component transform after level shifting. For multi component images, this helps the separate components to docorrelate efficiently. After that, the image is split into equal sized tiles. These tiles are the unit component to be encoded separately giving the advantage of distribution of needed memory for encoding larger images.

Wavelet transformation The tiles are then undergone the Discrete Wavelet Transformation(DWT). This transformation is used to perform similar task as done by the Discrete Cosine Transformation (DCT) performed in JPEG. In practice, DWT performs better in case of discontinuation of image properties than DCT. This results in fewer artifacts for sharp fluctuations in images. Depending upon the lossy or lossless mode of operation, reversible and irreversible DWT is performed in JPEG2000. The outcome of this phase are DWT coefficients that are to be processed further.

Quantization Quantization of the DWT done to have a better compression with the cost of loss of final quality. Like JPEG, the amount of losing details can also be tuned by choosing a parameter called "quantization step": higher step value yields better compression with lower quality images. This is the only irreversible step where information can be lost.

Entropy coding After quantization, the DWT coefficients are Data to be encoded are rearranged into 2 dimensional arrays called code blocks of equal sizes, and data in each code block are assumed as accumulation of bit planes. The term bit plane refers to the bits having the similar magnitudes in every samples. Each bit planes are encoded generating a series of binary symbols that are encoded using entropy arithmetic coding with the help of an adaptive probability model. The outcome of this stage is binary bit stream which are to be packetized and thereafter the final compressed output is obtained, known as the JPEG2000 codestream.

Summary In spite of having several advantages over JPEG, JPEG2000 is still is not a replacement of the original JPEG due to its heavy computational complexity. For the proposed system in this thesis, the expected lower amount of artifacts in higher compression ratio might get useful to obtain a better memory efficient model; which is to be figured out in later part of this report. The detailed description of JPEG2000 can be obtained from [15] and [14] for further consideration.

2.3.3 Near lossless mode of JPEG-LS

JPEG-LS standard is equipped with a near lossless mode[4] of operation which also is a candidate as a lossy coder in the proposed system. Near lossless mode is a specific type of compression which sets a bound in magnitudes in the amount of introduced loss. Unlike the lossy one, near lossless coders ensures that no pixel difference between the original and encoded ones cross a certain amount of difference[7]. This provides a control over final image quality by the users.

In JPEG-LS, the near lossless mode is operated by a variable, *NEAR* for example, which denotes the magnitude of tolerable errors by number. *NEAR*=0 represents the lossless and any positive integer number sets it to the quantization tolerance within $\pm NEAR$ range. Detailed description of JPEG-LS is present in previous section, and as for its usability in the proposed system, amount of memory efficiency with respect to compression ratio with near lossless mode is used as an addition of the regular lossy compression algorithms, making it an interesting topic to proceed and therefore included in the model.

2.4 Summary

Lossy, lossless and near lossless codecs described above are the key components of the proposed design that are used during the rest of the work done in this thesis. In the next chapter, a detailed discussion about design and development analysis of the proposed system is presented.

Chapter 3

Improvement over frame-by-frame JPEG-LS video coder

Designing lightweight lossless video coder with low memory cost introduces several challenges due to its conflicting requirements. A theoretical analysis of possible challenges with their solutions are described step by step in this chapter. Each possible issue in developing such a system is discussed here and suitable solutions for those issues are proposed keeping the overall goal of improving the compression gain over frame-by-frame JPEG-LS based video coder while maintaining high memory efficiency in mind. Before starting this analysis, a brief description of the target environment and the requirement is provided in the following section.

3.1 Target applications

The target applications of this work are the ones that handle videos with high frame rate with mostly static background. This scenario is very common where properties of a moving particle is the subject to detailed analysis which is captured in a high frame rate keeping a high image quality. Example applications may include molecular level particle analysis, medical and astronomical pattern analysis and so forth. This is useful for any kind of motion based research area where tiny movement of the subject is a matter of deeper evaluation.

In general, such applications come as a part of a bigger system used for the relevant purpose. For example, for medical imaging applications, the lossless encoder has to be integrated into the sensor chip to save storage space on the external server. Therefore, the encoder has to be developed to be usable for smaller scale embedded systems. This requirement enhances the need for more optimization in terms of computational complexity and memory efficiency.

The following section discusses the key design considerations, which are (1)Baseline encoder selection as a framework, (2)Enhancement of compression gain, and (3) Memory efficiency achievement for the target environment.

3.2 Problem analysis

3.2.1 Issue 1: Baseline encoder selection

Keeping the target environment discussed in Section 3.1 in mind, use of any lossless video coders gets expensive. Typically, the compression ratio of lossless coders are not as high as the lossy ones. Since videos are sequence of still images, standard image compression algorithms can also be used for video coding. Such intra frame encoding techniques are very popular as lossless video coders. Example encoders include lossless mode of JPEG, JPEG2000, JPEG-LS, CALIC etc. Among these, JPEG-LS and CALIC serves particularly well in terms of compression gain for lossless operations. Both of these use adaptive context based approach to remove the most of the redundancies an image could have in lossless mode. In practice, CALIC has a highly sophisticated computational mechanism that makes it the perform slightly better[1] than JPEG-LS in terms of compression gain. But as for the target of this work, the system has to be light-weight also. As discussed in Chapter 2, JPEG-LS on the other hand provides nearly the same performance (ranging within a few percentages) comparing to CALIC keeping computational complexity very low. Considering the target environment, JPEG-LS is a better candidate for the job.

Other types of lossless video coders use the temporal redundancy along with the spatial ones. H.264 is widely used these days which also provides a lossless mode. But encoders like these are computationally heavyweight since these consist of state of the art computational mechanism including motion estimation and compensation. Moreover, to achieve high performance, these coders use several reference frames simultaneously. This introduces a gigantic space complexity for lossless images which conflicts with the fundamental goal of memory efficiency of this work.

Considering the competitive facts about being less complex and more memory efficient, JPEG-LS algorithm stands out to be a very good choice as the framework for the lossless video coder framework. Its simple but effective context based design extracts a very good compression while keeping low memory requirement, which matches the objective of being efficient in compression gain with a lightweight structure.

3.2.2 Issue 2: Enhancement of compression gain

To obtain a higher compression ratio, utilization of the vast amount of inter frame redundancy of the high speed sequence are needed, therefore, temporal prediction has to be introduced to the system. Since the target video has mostly stationary backgrounds, temporal prediction would be very handy to enhance compression efficiency. Several works have already been done to improve compression efficiency over plain JPEG-LS utilizing temporal redundancy. Among those, the idea behind [17] and [3] are very suitable primarily because the temporal extension of the MED predictor of JPEG-LS has been achieved using only one past frame as reference. In [17], a combination of four prediction algorithms including bilinear interpolation and motion compensation, creating some computational overhead. Moreover, to represent the selected predictor choice that has been used for coding any block, some extra bits have to be encoded along with the image data. This increases the average code length of the encoded file.

On the other hand, the approach stated in [3] adopts context based adaptive prediction with a separate temporal module. This design offers a remarkably good compression gain (14.1% and 15.9% gain over static JPEG-LS in two test samples) for lossless videos with complexity of a usable range. However, the temporal extension used in the proposed system is a slightly modified version of this idea. The design in [3] is equipped with a near lossless mode of prediction. Since the goal of this work is to develop a lossless video coder, this near lossless mode has no relevance to the fully lossless framework. Therefore, this specific part has been ommitted from the proposed model, saving some computational load.

3.2.3 Issue 3: Memory efficiency in lossless coders

Lossless coders suffer from large memory requirements because of the fundamental criterion of being lossless. Since no distortion or loss is tolerable in such formats, the file size remains relatively large depending on the amount of details or noise, as well as the dimension of the image. This situation is also true for the proposed system where one frame has to be stored to be used as the reference. To make the system memory efficient, the reference frame storage procedure is further improved by adoption of lossy coding before saving. The concept used here is to save the reference frame after lossy encoding, and decode the compressed image partially whenever needed for the current frame's temporal prediction on the fly. Since the reference frame is lossy compressed before storing, the amount of memory required to save the whole frame is much lower than saving a lossless one. Clearly, this approach will require less memory to support temporal prediction and make the system more memory efficient.

3.3 The proposed system

Summing up all the issues in Section 3.2, the target system is an extended version of standard JPEG-LS algorithm since it offers a good compression of lossless images with a very low computational complexity. But in order to achieve better compression ratio for a high speed image sequence, temporal prediction is introduced. The temporal predictor model chosen for the job is equipped with context based bias cancellation and coding length optimization. This method ensures a very good compression gain [3]. On the other hand inter-frame prediction techniques usually increase the memory requirements due to the need for storage of a complete previous frame locally for predicting the next frame. To solve this problem, the storage module for reference frame is to be replaced by a lossy compression based storage system where frames are coded using lossy encoder before storing and decoded partially whenever needed for the current frame prediction. The simplified block diagram of the proposed system is illustrated in Figure 3.1.

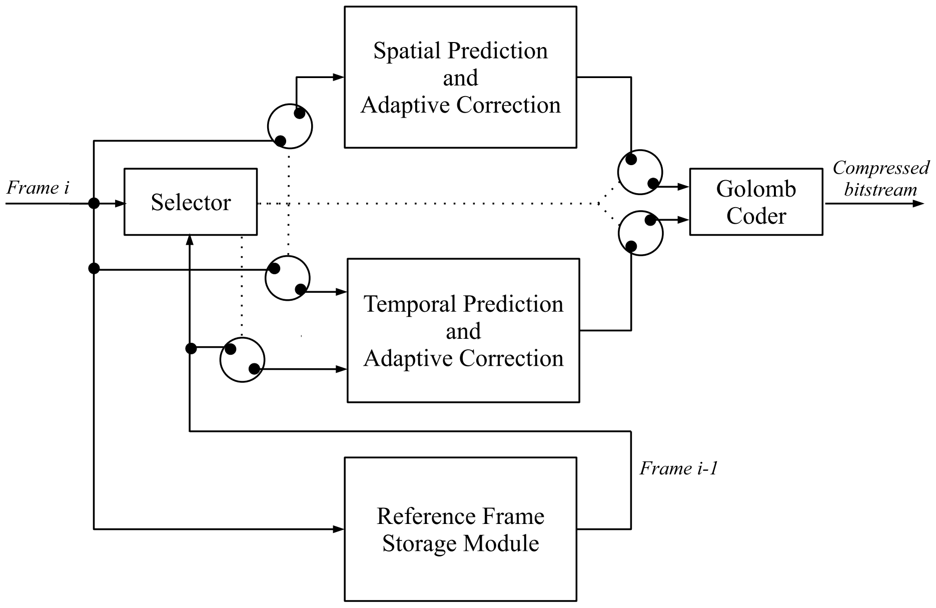
Spatial Prediction and Adaptive Correction
Frame i
Selector
Golomb Coder
Compressed bitstream
Temporal Prediction and Adaptive Correction
Frame i-1
Reference Frame Storage Module

Figure 3.1: Block diagram of the proposed model

As seen in Figure 3.1, the work-flow of the proposed model can be divided into five basic blocks. Measuring the intra and intra frame gradients, the **Selector** enables the appropriate prediction mode for encoding the current pixel. Depending upon the chosen prediction method, either the current frame data is passed through the **Spatial Prediction and Adaptive Correction** block or both the current and reference frame data is sent to the **Temporal Predictor and Adaptive Correction** block. The context based spatial predictor used here is similar to the one used in standard JPEG-LS (see Section 2.2.2). The temporal predictor also supports context based prediction bias cancellation and Golomb parameter specification. Detailed description of the working principal of the prediction model is explained later in Chapter 4.

Regardless of the used prediction method, the **Golomb Coder** does the encoding of the error value using code specifications delivered from the used predictor. This coder model is similar to the one used in standard JPEG-LS (see 2.2.2). Along with the temporal extension of the predictor model, another feature added to JPEG-LS framework is the **Reference Frame Storage Module**. It contains a lossy encoder and decoder, a storage space and a controller to handle the runtime store/load operation. Detailed description of the reference frame storage unit is available in Chapter 5.

The approach outlined in Figure 3.1 is designed to (1)ensure higher compression gain over static JPEG-LS and (2)maintain a very low memory requirement. The detailed description of the two additions (inter-frame predictor and lossy storage unit) in JPEG-LS are provided in the following chapters.

Chapter 4

Predictor Model: Temporal extension of MED predictor

The predictor model of the proposed system is context based and is equipped with spatial and temporal components. The spatial part is similar to the MED predictor used in the standard JPEG-LS. The temporal part is depended upon the pixel values from the previous frame and so is a purely temporal predictor. The appropriate coding scheme is selected adaptively depending on the image characteristics or context of the applicable part of the image. The block diagram of the predictor model is illustrated in Figure 4.1. Brief working procedure of these sub modules of the model are stated in the following sections.

4.1 Selector

The adaptive selector figures out the appropriate coding scheme for the current pixel using neighboring pixels of the current and the past frame as demonstrated in Figure 4.2 and Figure 4.3. For the purpose, it uses the already constructed neighbors which is available for the decoder as well. It calculates the spatial and temporal variation (V_s and V_t respectively):

$$V_s = |R_d - R_b| + |R_b - R_c| + |R_c - R_a| \tag{4.1}$$

$$V_t = \left|R_a - R'_a\right| + \left|R_b - R'_b\right| + \left|R_c - R'_c\right| + \left|R_d - R'_d\right| \tag{4.2}$$

Here, the term V_s is the summation of the absolute local gradient values which shows the property of the current context using probable smoothness or roughness of the adjacent area of the present frame. On the other hand, V_t is a way to measure the change in the individual neighboring pixel magnitudes with respect to the previous frame which shows the temporal changing behavior. Then these two parameters are compared using a variation factor

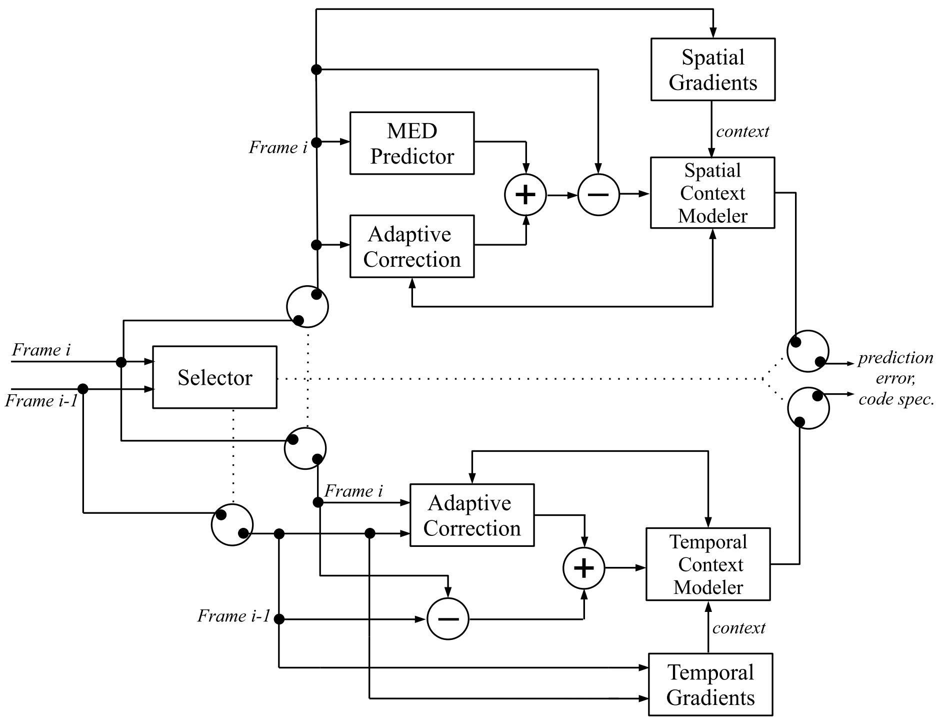

Figure 4.1: Temporal Prediction and Adaptive Correction block

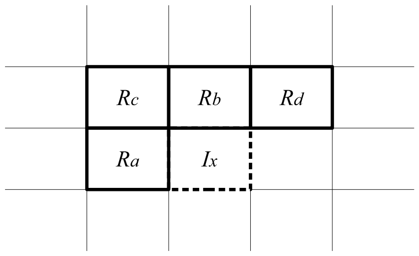

Figure 4.2: Current frame

I_x: Current sample
R_a, R_b, R_c and R_d: Reconstructed neighboring samples

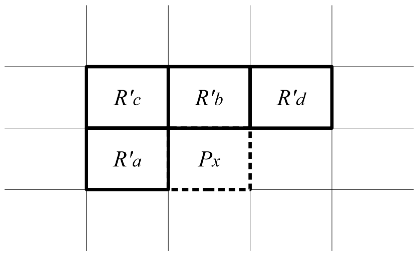

Figure 4.3: Past frame

P_x: Located in the same spatial location as I_x
R'_a, R'_b, R'_c and R'_d: Reconstructed values of equivalent location with respect to the present frames R_a, R_b, R_c and R_d

α to choose between these two predictors by measuring

$$Select_t = (V_t < \alpha.V_s) \tag{4.3}$$

If the boolean $Select_t$ is true, the temporal predictor is used. Another interpretation would be, if the temporal change with respect to the past frame is smaller than current frame's spatial change, the temporal predictor should be used. If this specific change is too large, the regular spatial predictor would yield better result or smaller code length. However, the choice of prediction mode can be partially controlled by changing α. If α is more than 1, the temporal predictor is chosen more often. Values less than 1 makes the selector to choose the spatial predictor more. For example, a zero value α will always run the model in the spatial mode since in that case the right hand side part of Equation (4.3) will always be false. This parameter α can be used as a tuner of the selection logic for testing. Figure 4.4 shows the basic step-by-step procedure of the predictor switching model.

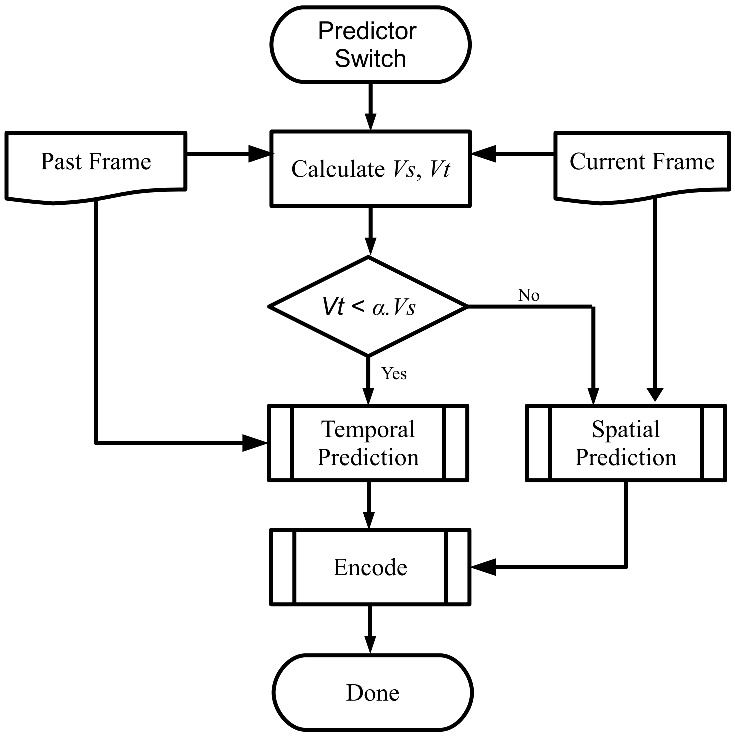

Figure 4.4: Predictor switching process

The adaptive selector logic defined by (4.1)- (4.3) enables the model to smartly exploit the temporal data in addition to the already existing spatial predictor. This kind of co-existing and clever switching uses the advantages of the both worlds in one system and generates better result in terms of overall compression ratio of the encoder. The detailed description of both the predictors are provided in the following sections.

4.2 Spatial predictor

The spatial predictor used in this model is the same MED predictor defined in Chapter 2 (see Section 2.2.2). This context based predictor uses the already generated/available neighboring pixel densities of the current frame to do the prediction, context generation and error coding. The working and neighboring pixel magnitudes and locations of the current frame are similar and therefore referenced as the Figure 4.2 for this section.

First of all, with the help of *a, b, c* and *d* it generates a predicted value I_x. This is achieved by including a simple but effective approach for edge detection. I_x is then used for generating prediction error value $Errval$ for further processing. Afterwards, with the help of the same neighboring pixel magnitudes, three local gradients D_i are calculated to generate a total of 365 contexts Q_i showing the working pixel's probable properties. This value Q_i is already optimized for model simplification with the help of quantization and sign reduction.

With the help of the context parameter, the bias generated in prediction error is reduced which is called bias cancellation. Moreover, the error coder also exploits the context to tune itself for average code length reduction. An extended version of Golomb family coder is used for coding the error signals and its tunable parameter uses the context parameter to enhance efficiency with few basic calculations, yielding a comparable result obtained by the use of more efficient arithmetic coders.

Implementation of this spatial predictor is also straightforward. The edge detection engine is kept very light-weight, with only three simple conditions for horizontal edge, vertical edge and smooth area determination and one line calculation of predicted value for each of those conditions. The context modeler uses few registers to aggregate few parameters like context frequency, accumulated error values to adjust the bias cancellation and code optimization. These registers contain these values for each of these 365 contexts and updates in each iteration. As a result, we get a context based spatial predictor with a very low computational and space complexity.

4.3 Temporal predictor

Unlike the spatial one, the temporal predictor exploits the redundancies obtained from the immediate past frame. For the rest of this section, reference values for both present and past frames are taken from Figure 4.2 and Figure 4.3 where each working and neighboring pixel subscripted as *a, b, c, d* and *x* shows both their locations and magnitudes.

The predicted value I_x in this case is the pixel magnitude of the same spatial location of the working pixel in current frame. The prediction error P_t

is calculated and optimized using a context based algorithm.Mathematically,

$$P_t = (I_x - P_x) + B\,(C_t) \tag{4.4}$$

where C_t is the bias context and B is the bias register that contains the respective bias value. Rest of the parameters are similar to the model shown in Figure 4.2 and Figure 4.2.

Context determination for error correction For the temporal predictor, the context parameter is defined using the changing properties of the individual pixels surrounding the candidate with respect to the previous frame. These properties are obtained by measuring the summation of the four inter-frame differences of the surrounding pixels.

$$C_t' = \left(R_a - R_a'\right) + \left(R_b - R_b'\right) + \left(R_c - R_c'\right) + \left(R_d - R_d'\right) \tag{4.5}$$

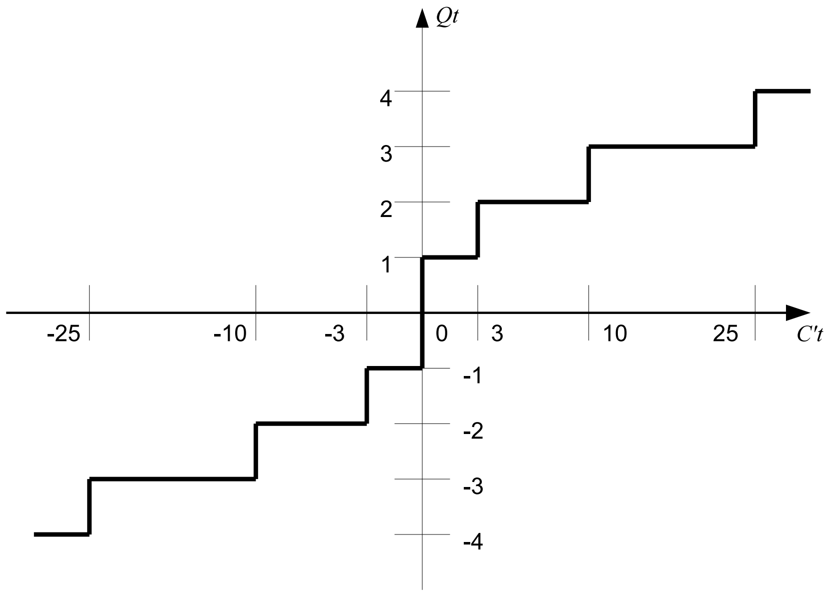

Figure 4.5: Quantizer model for bias cancellation context.
C_t': Temporal gradients
Q_t: Quantized value

This C_t' is then quantized into 7 levels (Figure 4.5) to obtain a concrete usable value and a further reduction of the model size is obtained by converting the quantized magnitudes into some smaller values. Here, the step size after reducing the sign is 4 ($=T$), and total number of temporal gradients is also 4. This yields the number of contexts as $[(2T + 1)^4 + 1]/2 = 3281$.This can not be called a small number, but it makes sense for proper bias cancellation for temporal predictor where such higher range of necessary gradients are to be considered.

Context determination for error coding The corrected value is then coded using similar extended Golomb family coding algorithm, which is used by the regular JPEG-LS standard. As the prediction error calculated this way can be modeled as Figure 2.3, the average coding length can also be reduced with the help of context based tuning[1]. For the purpose, the coding context is determined using equation (4.2) since it is the value V_t depicts the property of temporal change mathematically. For further simplification, the value V_t is quantized using a 5 level quantizer of [3, 10, 25, 50](Figure 4.6) and converted into smaller integers for simplicity in the model. This way, the size of the coding context remains 5.

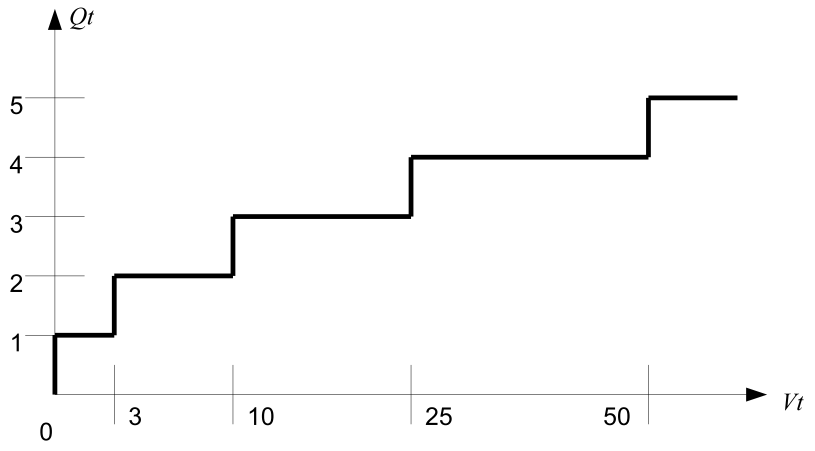

Figure 4.6: Quantizer model for error coding context.
V_t: Absolute temporal variation (4.2)
Q_t: Quantized value

The integration of these two adaptive context based correction and tuning yields significant reduction in average code length. Sample simulation results [3] shows this temporal predictor in combination with MED predictor in JPEG-LS gives almost 15% reduction in bits per pixel value comparing to regular JPEG-LS video coder. As for the predictor model for this work, in conjunction with the satisfactory compression gain, it is very much suitable in terms of computational effort, space requirement as in model cost.

Chapter 5

Improving memory efficiency by on-line compression of the reference frame

Memory efficiency is a key issue for video coders which exploit temporal correlations among the frames since it requires at least one past frame to be stored in the buffer. The predictor used in the proposed model also needs one reference frame to perform its operation. The amount of space required to store a complete lossless frame is considerably large and therefore becomes expensive for lightweight applications. In the proposed model, an effective approach is proposed to alleviate this problem by introducing on-line lossy compression based storage unit to save the reference frame.

5.1 Storage unit

The frame storage unit used in this model uses an intermediate lossy encoder and decoder package to store and load the frame to be saved. Before saving any image to the unit, the whole frame is compressed using a lossy image coder. And when this data is to be used for calculation later, it is decoded again to obtain the individual pixel magnitudes. For a lossy codec with compression ratio of N:1, this model will also reduce the memory requirement by a factor of N at most. A simplified block diagram of the proposed storage unit is illustrated in Figure 5.1.

5.2 Working procedure

Normally, a temporal video coder which uses only one past frame, deals with two images simultaneously. One of these is the operating(current) frame $Frame_i$ and other one is the reference(past) frame $Frame_{i-1}$ to deal

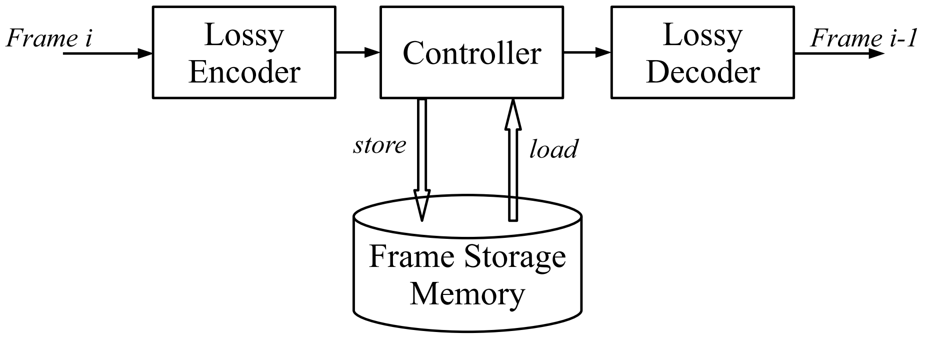

Figure 5.1: Reference Frame Storage Module block

with the run-time temporal calculations. For a totally sequential system, one set of calculation is done in one pixel scan and therefore rest of the pixels in $Frame_{i-1}$ are redundant for the computational module. So, the system saves the $Frame_{i-1}$ using the lossy encoder in the memory and thereafter decodes and loads only the necessary pixel values from that frame at that particular time of calculation and discards afterwards. The logical work-flow model is shown in Figure 5.2. Let's say a video consists of M number of frames, each with n number of pixels. After working on a particular frame, the system has to save this frame as the past frame for the next frame calculation. Before saving this, the frame is coded using a lossy encoder to save space in the memory module. During the next frame iteration this frame becomes the past frame $Frame_{i-1}$ and the current frame is $Frame_i$, which is scanned sequentially from top-left to bottom-right by the system. At a particular scanning point x, along with the current frame's pixel I_x, let's say that the pixel magnitude of the previous frame's point x is also needed, noted as P_x. At that particular time, the system checks if P_x is already decoded and ready for use. If not, it decodes that, uses for the calculation, and discards afterwards if it's not supposed to be needed on a later point. Except for the lossy codec, the implementation does not require heavy calculation, but offers a huge save in memory. The precise memory reduction is described in the following section.

5.3 Memory reduction

Case 1: Regular mode without lossy save In regular JPEG-LS based video coder with temporal predictor, one whole frame must be saved to use later. In this case, for a video with b bit depth, the amount of memory required B_c to save each frame can be noted as

$$B_c = b.nbits \tag{5.1}$$

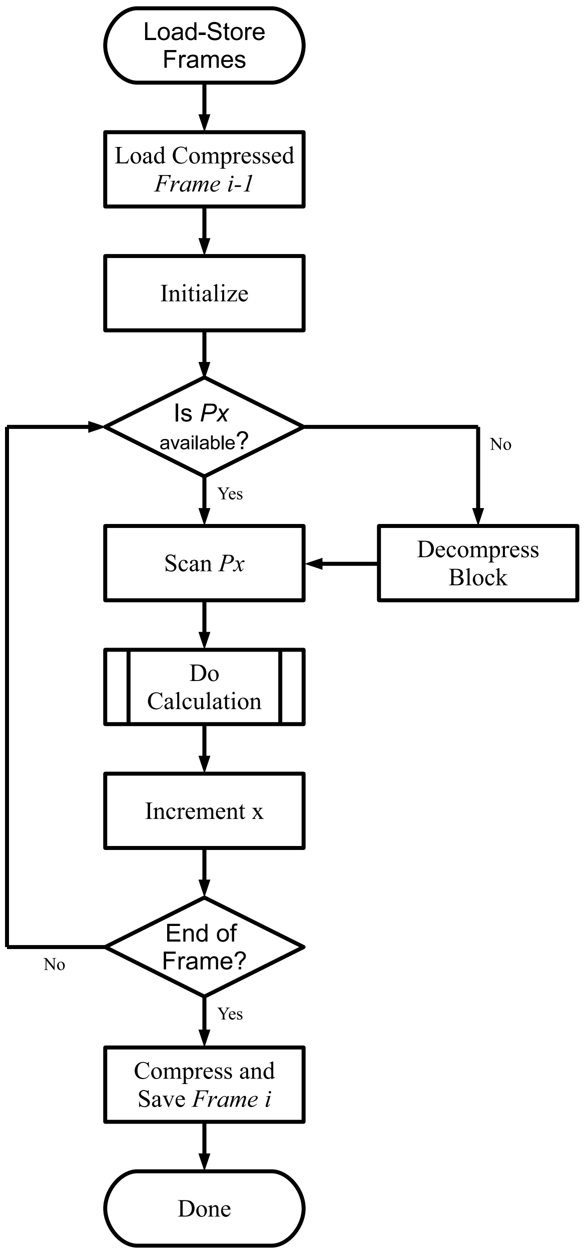

Figure 5.2: Work-flow of the memory unit.

Case 2: Using lossy save with decompression unit of 1 pixel If the lossy codec with a compression ratio of 1:R has the capability of decoding one pixel per time, the required memory B_n would be

$$B_{n1} = b.\left(\frac{n}{R} + 1\right) bits \tag{5.2}$$

Comparing to case 1, this method reduces the memory requirement almost by a factor of R. Amount of memory gain can be calculated using the following equation using case 1 as reference:

$$MemoryGain_{ni} = \frac{B_c - B_{ni}}{B_c} \tag{5.3}$$

which yields the gain for case 2 as:

$$MemoryGain_{n1} = \left(1 - \frac{1}{R} - \frac{1}{n}\right) \tag{5.4}$$

Case 3: Using lossy save with decompression unit of blocks In practice, lossy codecs encodes and decodes the images by blocks rather than by a pixel. So, for a typical case if the lossy coder decodes one block at a time with block size of *heightXwidth*, required memory is

$$B_{n2} = b.\left(\frac{n}{M} + heightXwidth\right) bits \tag{5.5}$$

And the amount of gain using (5.3) is:

$$MemoryGain_{n2} = \left(1 - \frac{1}{R} - \frac{heightXwidth}{n}\right) \tag{5.6}$$

5.4 Lossy codecs

The proposed memory efficient video compression framework in Figure 3.1 is compatible with a large variety of lossy image coders. For initial investigation perpose, we have tested several coders including JPEG, JPEG2000 and near lossless JPEG-LS. However it should be mentioned that the overall performance would require certain properties of the lossy coder used, such as scanning order, compression ratio and the amount of distortions introduced after compression which is relevant for the temporal prediction quality. The proposed system has a row major pixel per pixel scanning order like JPEG-LS. In this regard, most of the existing lossy compression algorithms are suitable to used in this system. As mentioned in the previous section, only drawback is that lossy encoders adopt block based compression system for the intermediate co related transformation mostly. As a result, a bunch of pixel values are decoded at a time, even though only one or few more neighboring pixels are needed for a particular operation. This causes a redundancy of the pixels that has to be saved or discarded for later use. If rest of the pixels are stored, it consumes memory, which of course contradicts the theory of memory efficient model of this work. And if we discard those, it saves memory, but introduces a huge computational burden. This way, each pixel in an image has to be decoded more than once, and for bigger block size, this number is unacceptable. So, this is always a trade-off between the cost for storage and computational burden. Still, a suitable approach to handle this problem would be to match the most with the JPEG-LS based scan. For instance, for the lossy decoder a row of pixels can be saved at a time discarding the rests. Because we know that the pixels in right positions (of the same row) are to be used for the following computations. Rest of the rows of that blocks will be used only after finishing all the vertical blocks of that image, so decoding this block on that much later point makes more sense.

Compression ratio is an important factor for choosing the type of lossy coders. As we can see in the previous section, the memory efficiency is directly proportional to R. But with the increase compression, quality of the image also falls which affects the temporal prediction badly. The prediction accuracy of such predictors goes down with the loss of image quality. As a result, with the use of a lossy coder with higher compression ability, the predictor model tends to choose the spatial predictor more often reducing the advantage of usage of temporal predictor and therefore reducing the compression ratio of the whole system. If a lossless codec is used for intermediate save, memory efficiency goes down, but results in a very good overall compression of the video. And usage of a lossy coder with high compression gain, less memory is needed, but compression gain of the video coder becomes less than before. Once again, this introduces another trade-off between the system's memory efficiency and compression gain.

For the implementation and simulation purpose, JPEG, JPEG2000 and near lossless mode of JPEG-LS are used for the lossy save. Implementation and detailed analysis and performance measurements using the proposed model are presented in the next chapter.

Chapter 6

Implementation and experimentation results

In the last two chapters, the improvement potential of the frame-by-frame JPEG-LS based video coder has been proposed using temporal extension of the prediction model while maintaining the memory efficiency by performing on-line compression as a pre-processing step in the memory block. In this chapter, extensive simulations have been performed in order to check performance and usability of the proposed model. A brief outline of the implementation procedure is provided in the following section to understand the overall experiment's environment.

6.1 Implementation outline

To test the proposed model's outcome, the software model of the system illustrated in Figure 3.1 has been implemented. Development of the model is primarily based on standard C language in Microsoft Windows platform. The implementation is performed following three basic steps to maintain a disciplined and error free model generation:

Step 1: Implementation of baseline JPEG-LS Since this model is an improvement over standard JPEG-LS, the first step of the implementation is also to develop the baseline JPEG-LS encoder model. Implementation and simulations in this chapter are based upon 8 bit grayscale lossless video samples. The implementation is based upon the regular mode of operation of the baseline JPEG-LS encoder since regular mode is mainly operational for lossless compression of natural images. Other than that, the detailed design description has been followed from [4] to maintain the standard and future extendibility. The outcome of this encoder has been tested comparing standard encoded image for accuracy measurement.

Step 2: Implementation of the temporal predictor The temporal predictor model described in chapter 4 is then developed and added to the standard JPEG-LS model. The temporal predictor of the proposed model is also context based like the prediction unit of baseline JPEG-LS. Therefore, integration of the proposed prediction model into the baseline JPEG-LS based framework is logically convenient. One drawback of the proposed temporal prediction unit is that the number of contexts used for bias cancellation and error coding is too few, specifically 4 and 5. Although such small number of contexts provide much better result over usage of only regular MED predictor model, intermediate test results has shown some further improvements in adoption of larger number of contexts for bias cancellation. As a result, context numbers for this particular part are increased for a better outcome of the overall model.

Step 3: Implementation of on-line memory module In Chapter 5, a modular description of the memory model was presented. Such a model is to be implemented next and integrated to the model. Since implementation of this part includes the usage of few lossy encoders and decoders, namely JPEG, JPEG2000 and Near lossless mode of JPEG-LS, this turned out to be a rather lengthy process than expected. For limited time span for this particular implementation part of the project, lossy codecs were adopted from open source implementations. Since this work merely depends upon thorough understanding of all these lossy codec standards, adoption of external sources does not affect the system's outcome or reliability. However, a detailed analysis and implementation of baseline JPEG encoder and decoder has been done and integrated to the system for integration ability verification.

Step 4: Integration After all the modular development mentioned above, the whole system was integrated to be used as a video coder for further usage and evaluation purpose. This integration step made this model to be used for sequence of gray-scale bitmap image sequences to be simulated as a lossless video input. The constraints of these input files are similar to standard JPEG-LS and outcome of the system is also a series of files that represent the encoded video.

The system is built keeping further research and development in mind, several tunable parameters are kept open for later adjustment and evaluation. For example, the parameter α (stated in chapter 4) that controls the choice bias between spatial and temporal predictor for a particular scenario, is left open. It has been set to 1.0 for the simulation purpose. The choice of lossy coder is also left open which is something to be changed whenever needed depending upon the user group and their specific needs.

A decoder of the proposed system is also implemented to check the ac-

Sample	Frame Resolution	Number of Frames
salesman	360 X 288	449
container	352 X 288	300
tennis	352 X 240	151
mother and daughter	352 X 288	300

Table 6.1: Used samples

curacy and usability of the system. The bit level match between the input files of the encoder and the outputs of the decoder has proven the expected lossless behavior of the model. The inputs and outcomes of the software model are further subject to be analyzed and discussed for the success of the proposed design, which is performed if the following sections.

6.2 Test parameters

6.2.1 Test input samples

The outcome of the system has been tested using four distinct gray-scale lossless video samples:

All samples are taken from [23] and [24]. Although, samples with similar names are available in several sources, similarities in file types are required to get a comparable result. For example, while testing the sample “salesman” during the implementation of the temporal predictor model, significant difference in result has been encountered comparing to the results from [3] because of the difference in resolution and possible earlier down sampling. Appropriate samples have been selected considering their resolution, frame rate and file type. After selection, frames from the sample videos are separated and converted into a certain color space using a video post production tool Avysinth[25]. For the record, the sample “tennis” was already separated into frames from the source, therefore, no manual extraction was needed.

Each of these frames were then encoded using the frame-by-frame JPEG-LS encoder to introduce a ground for comparison. Afterwards, these samples are encoded using the proposed encoder model to test the improvement of performance in terms of gain in compression ratio and reduction in memory.

6.2.2 Performance measurement terms

Calculation of compression gain The gain in compression ratio of the system is calculated using Equation (6.1) where $Bitrate_{JPEG-LS}$ and $Bitrate_{proposed}$ denotes the calculated bit rate of frame-by-frame JPEG-LS encoded video and bit rate of the proposed model respectively.

$$Gain_{proposed} = \frac{Bitrate_{JPEG-LS} - Bitrate_{proposed}}{Bitrate_{JPEG-LS}} \times 100 \qquad (6.1)$$

The resultant term in Equation 6.1 gives a clear indication of amount of relative improvement in overall compression potential.

Calculation of memory efficiency Since the memory efficiency using on-line compression method corresponds closely to the used lossy compression codec in the system, this efficiency term can be calculated simply by measuring the compression ratio of that used lossy compression algorithm for the test sample.

6.2.3 Tunable parameters of the lossless framework

The α parameter The spatiotemporal predictor model introduced in Chapter 4 works by selecting one of the prediction submodules (intra or inter frame predictor) depending on the runtime behavior of the current and the reference frame. However, an input variable α, can be used to make some biasness for the choice criteria of the prediction algorithm. If the value if α is set to greater than 1, the system tends to choose the temporal predictor more often. The opposite tendencies are visible for values near to zero. For the test and simulation purpose, it is set to 1.

Frame save option Since the proposed model uses several lossy codecs for the memory module, the parameter *FRAMESAVEOPTION* can be set to to choose one of the available codecs. For $FRAMESAVEOPTION = 0$, the lossless mode of frame save is performed, keeping the raw data in the buffer for next frame processing.

6.2.4 Tunable parameters of the lossy codecs

Three lossy codecs are used in the memory module as described in Chapter 5 to reduce the amount of storage space for saving the lossless reference frame. These are the baseline implementation of JPEG, JPEG2000 and near lossless mode of JPEG-LS. Required description of these three have been briefly discussed in Chapter 2, where further references have been provided. The compression ratio of each of these codecs are tunable for convenience, and for the test purpose, the outcome of the proposed system are also subjects of analysis with the variation of the compression ratio of the lossy coders. A brief description of tunable parameters of these codecs from an implementation point of view is given below.

JPEG The JPEG encoder and decoder model used in this system is a open source implementation [19] of standard JPEG algorithm. The compression performance of this system can be tuned using a integer number N_{JPEG}, ranging from 0 to 100. With bigger N_{JPEG}, quality of the reconstructed image gets higher decreasing the amount of compression.

JPEG2000 Unlike JPEG, the tunable parameter in JPEG2000 used in the system uses N_{j2k} which represents approximate amount of resulting compression ratio. The used model is taken from [21] and integrated to the proposed model for test purpose.

Near lossless mode of JPEG-LS In this mode, the amount of compression range is set by choosing *NEAR* parameter which sets the bound of maximum possible range of change in pixel magnitudes. Therefore, higher value of *NEAR* results in higher memory efficiency and lower quality reconstructed image. The software model of this codec is developed using C and integrated to the lossless coder.

6.3 Analysis of video compression gain over compression ratio of lossy coders

The primary goal of the proposed lossless video compression algorithm is to reduce the amount of data needed to represent a fully reconstructible video file. To test the system's outcome in terms of compression gain, the sample video files are compressed using both the proposed model and frame-by-frame JPEG-LS video coder to measure the $Gain_{proposed}$ using Equation (6.1). Any positive value of computed $Gain_{proposed}$ would represent its improvement over standard JPEG-LS.

To have a thorough test result, each sample was tested enabling all three lossy codecs (by varying *FRAMESAVEOPTION*)for the memory module. Variation of the lossy tuning was done for a deeper analysis of the system's behavior with respect to the quality of the reference frame. The detailed outcome of the implemented system has been illustrated in Figure 6.1 through Figure 6.12 for a detailed analysis. In these plots, behavior of the proposed system over change of lossy compression (by tuning N_{jpeg}, N_{j2k} or *NEAR* as described in the previous section) for all three lossy compression based implementations has been illustrated. The first 50 frames from each of the four samples were subject to this test.

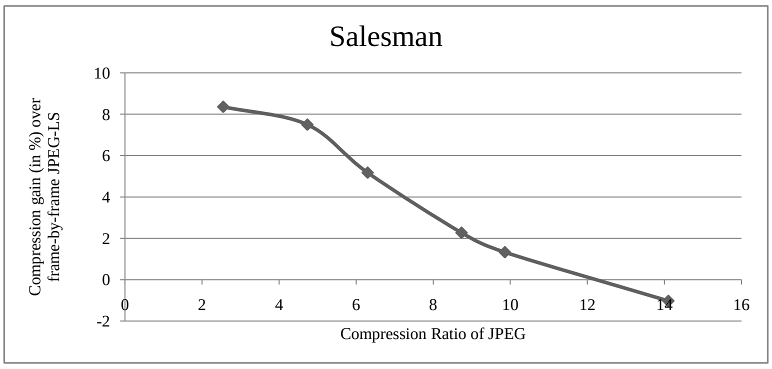

Figure 6.1: Analysis on JPEG on-line save

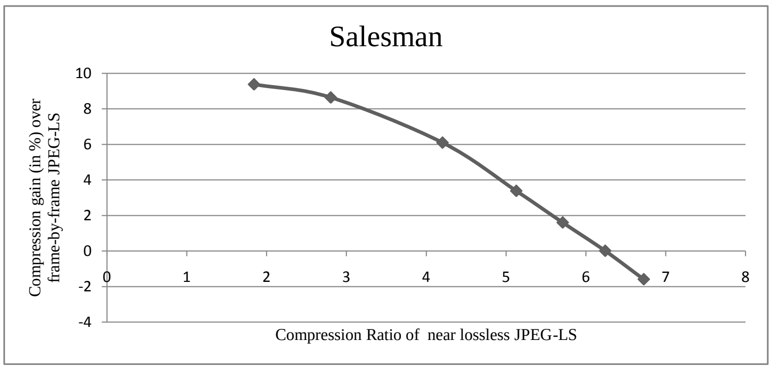

Figure 6.2: Analysis on JPEG-LS near lossless on-line save

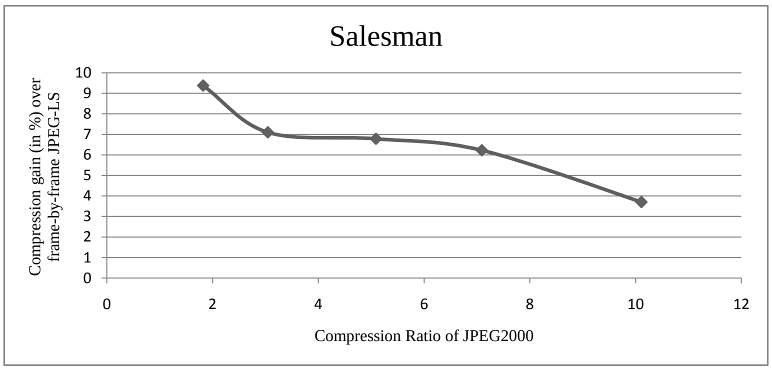

Figure 6.3: Analysis on JPEG2000 on-line save

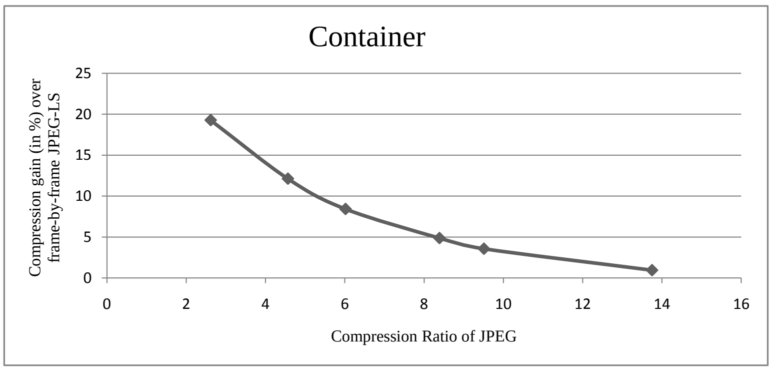

Figure 6.4: Analysis on JPEG on-line save

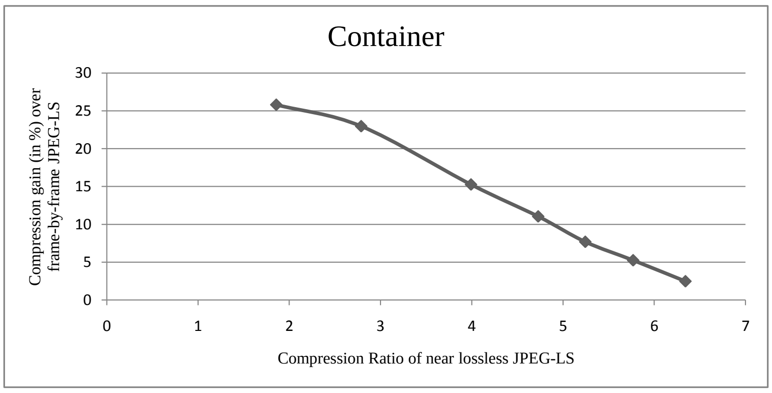

Figure 6.5: Analysis on JPEG-LS near lossless on-line save

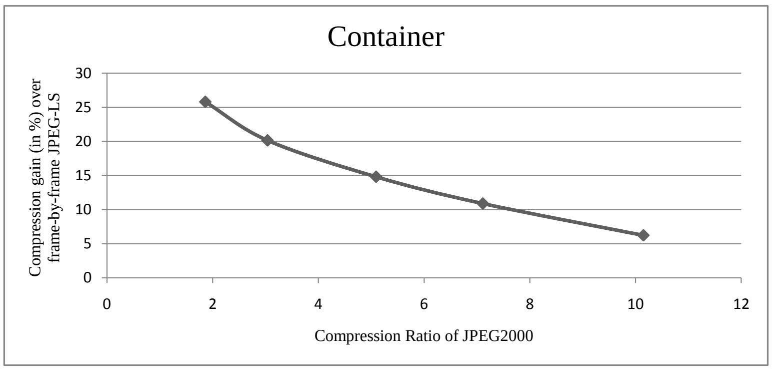

Figure 6.6: Analysis on JPEG2000 on-line save

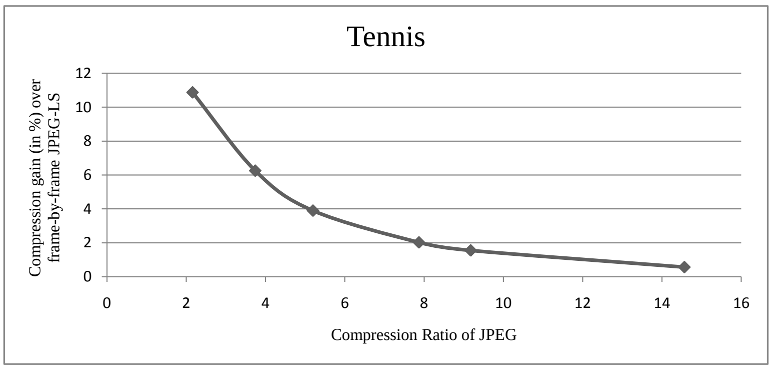

Figure 6.7: Analysis on JPEG on-line save

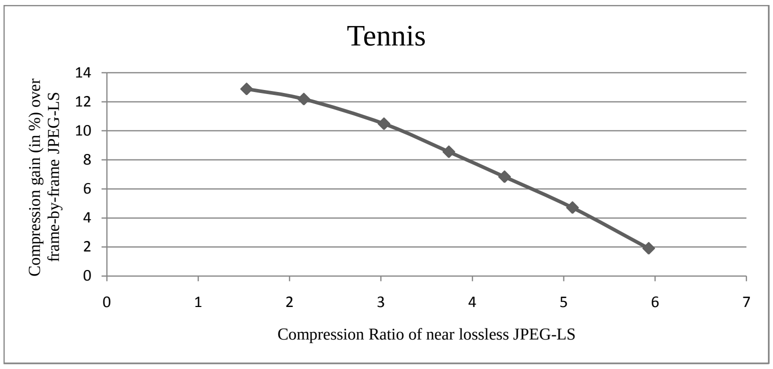

Figure 6.8: Analysis on JPEG-LS near lossless on-line save

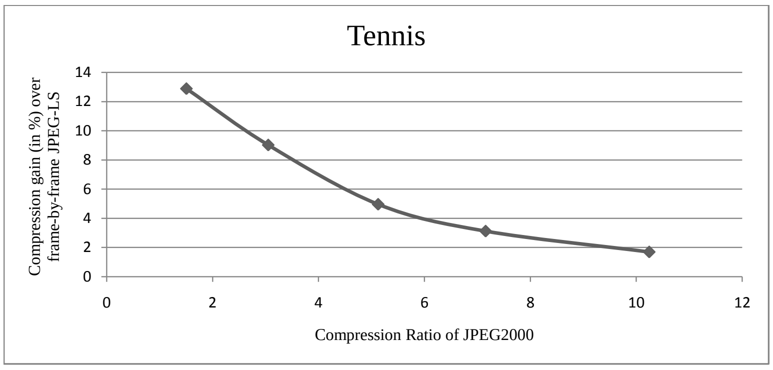

Figure 6.9: Analysis on JPEG2000 on-line save

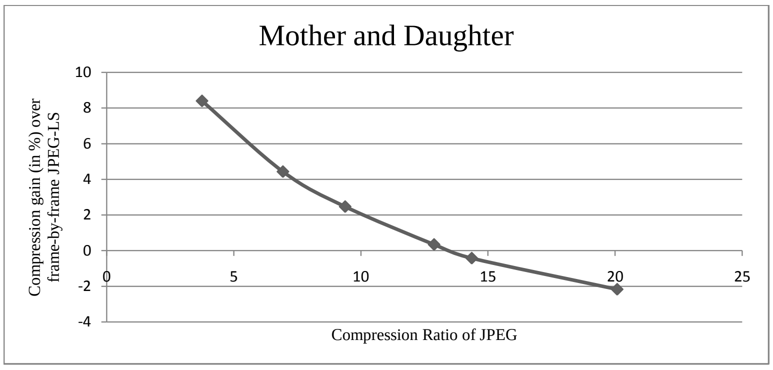

Figure 6.10: Analysis on JPEG on-line save

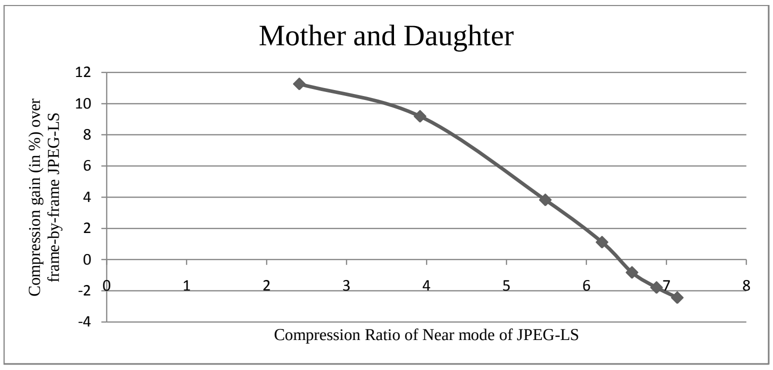

Figure 6.11: Analysis on JPEG-LS near lossless on-line save

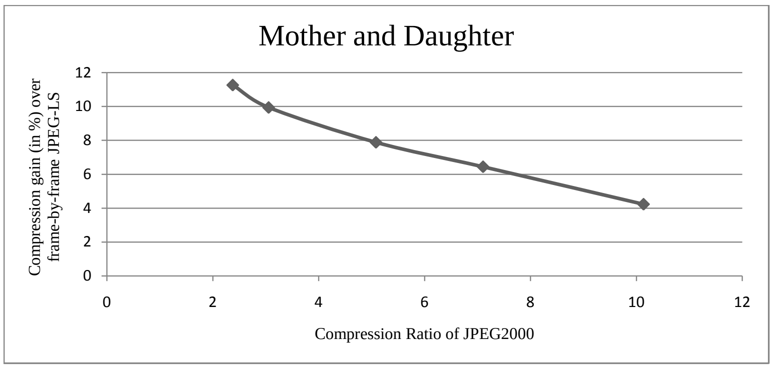

Figure 6.12: Analysis on JPEG on-line save

From the plots in Figure 6.1 to Figure 6.12, the domination of positive values of $Gain_{proposed}$ shows a clear indication of better compression performance over frame-by-frame JPEG-LS video coders. However, the amount of magnitude of $Gain_{proposed}$ varies with the change of compression parameter of the lossy codecs. The common behavior seen in all the charts is that, with the increase of lossy codec's compression ratio, the outcome potential of the proposed system in terms of compression gain gets lower. The highest gain of the system is observed when the least or no amount of loss in intermediate reconstruction of lossy images has occurred. For example, for all four samples peak $Gain_{proposed}$ is observed whenever $NEAR$=0 in Near lossless mode of JPEG-LS or $N_{j2k} = 1$ in JPEG2000 is chosen. The amount of highest $Gain_{proposed}$ ranges between roughly 9% (Figure 6.3) to 25% (Figure 6.5 and Figure 6.6). However, the peak gain in case of usage of JPEG save does not incline to such a high point for any of the samples, making it inferior in higher video compression gain outcome.

In the higher compression zone of the lossy coders, both JPEG and JPEG2000 based systems offers better gain than near lossless JPEG-LS based systems. For all four samples, $Gain_{proposed}$ reaches to a point of 0 within range of compression ratio roughly between 6 to 7 for near JPEG-LS based approach. For JPEG2000 based systems this range remains between approximately 6 to 13 depending upon the video samples. But JPEG based implementation offers such positive $Gain_{proposed}$ giving a compression ratio of about 12 (Figure 6.1) at least, and this number gets bigger for other samples. As a result, a constant higher memory efficiency is observed in case of JPEG based approach comparing to the other two implementations.

To summarize, the proposed system offers better compression of lossless video comparing to standard frame-by-frame JPEG-LS based video coders regardless of whatever intermediate lossy compression algorithm is used. Among the three lossy codecs used for memory efficiency, JPEG performs best in higher compression ratio (of JPEG) zone, offering a good reduction of in-processor space requirement. In the similar zone, the near lossless mode of JPEG-LS performs the worst of all three. On the other hand, in higher compression gain of video (lower compression ratio of lossy coders) region, both JPEG2000 and near lossless JPEG-LS based implementations offer superior video compression comparing to the standard JPEG based approach. Since computational overhead of JPEG2000 is much higher then near lossless JPEG-LS, the latter one would be the better choice of lossy coder whenever a higher range of video compression is demanded.

6.4 Analysis of performance gain over similar memory efficiency

Introduction of the on-line compression of reference frame as described in Chapter 5, has been designed to save the amount of memory required for storing one past frame for temporal prediction. The amount of compression ratio extracted from the lossy codec is the measurement of the term memory efficiency for the proposed model. As seen on the test runs (Figure 6.1 through Figure 6.12) of all four samples, positive amount of $Gain_{proposed}$ has been seen using various compression ratio of different lossy algorithms.

The magnitude of the outcome of the system depends upon the amount of applied compression ratio of the lossy codec (or memory efficiency for this system). From all the tests, a gradual decrease of $Gain_{proposed}$ is observed with the increase of compression in frame storage, making the memory efficiency inversely proportional to system's compression potential. Mathematically,memory efficiency of this proposed model ($MemoryEfficiency_{proposed}$) can be defined as,

$$MemoryEfficiency_{proposed} = \frac{EfficiencyFactor}{Gain_{proposed}} \tag{6.2}$$

Here, the term *EfficiencyFactor* depends upon variables like the behavior of the used lossy codec, information of the samples and overall frame handling procedure of the proposed model. Measurement of quantitative memory efficiency is tricky since it is delivered as an outcome of the lossy compression rather than input of the system. However, by fixing the target system's requirement of memory efficiency, the compression of the overall outcome can be analyzed for further analysis.

To perform this analysis, a two step approach is followed. On the first step, a fixed amount of compression ratio of the intermediate lossy encoder is set by tuning the lossy parameter of the respective lossy encoder. Afterwards the amount of $Gain_{proposed}$ is calculated for that specific amount of generated loss on the previous step. This method is performed using all four test samples to get an overview of the system's outcome over a specific memory efficiency measurement.

For the test purpose the amount of $MemoryEfficiency_{proposed}$ is set to around 5 by tuning the lossy parameters N_{jpeg}, N_{j2k} and *NEAR* for JPEG, JPEG2000 and near lossless mode of JPEG-LS respectively. Table 6.2 shows the exact amount of compression gained over the raw input file along with relevant lossy parameters. Here, the term *CR* is the amount of compression ratio comparing to the size of the respective samples.

Afterwards video coding of the samples are performed using all three lossy codecs with their relevant lossy tune parameter determined in the Table 6.2. The outcome of the system is stated in Table 6.3. Here, BR_{ref} is the bit rate in bpp (bits per pixel) of the frame-by-frame JPEG-LS based

Sample	Sample size (bytes)	JPEG		Near losssless JPEG-LS		JPEG2000	
		N_{jpeg}	CR	$NEAR$	CR	N_{j2k}	CR
salesman	47,036,342	82	5.22	5	5.12	5	5.09
container	12,919,258	75	5.1	7	5.23	5	5.09
tennis	30,736,200	81	5.13	10	5	5	5.11
mother and daughter	30,736,200	91	5.16	3	5.48	5	5.08

Table 6.2: Complession ratio matching of lossy module

encoded samples which is used as the reference point to measure the system's performance. *BR* is the bit rate of the proposed system after encoding using the proposed model. Then this *BR* for a particular implementation is compared with BR_{ref} to calculate the *Gain(%)* in output compression ratio of the proposed model. For the record, this *Gain(%)* = $Gain_{proposed}$ defined in Equation (6.1) using each of the lossy algorithm based implementation.

Sample	BR_{ref}	Using JPEG		Using Near losssless JPEG-LS		Using JPEG2000	
		BR	$Gain(\%)$	BR	$Gain(\%)$	BR	$Gain(\%)$
salesman	4.39	4.06	7.49	4.22	3.85	4.06	7.51
container	4.38	3.90	10.93	4.03	8.04	3.70	15.46
tennis	5.19	4.93	5.03	4.95	4.68	4.84	8.07
mother and daughter	3.36	3.16	5.83	3.25	3.15	3.13	6.85
Average	3.36	3.16	5.83	3.25	3.15	3.13	6.85

Table 6.3: Outcome of the proposed model

From Table 6.2 and Table 6.3, the performance of the proposed system is seen with fixed memory efficiency with a factor of approximately 5. Keeping the memory reduction constant, an average bit rate of the compressed files results in compression gain of 5.83%, 3.15% and 6.85% over standard frame-by-frame based JPEG-LS video coder. This shows a very good compression outcome from the proposed model keeping the memory requirement very low (as a factor of approx. 5), which is the primary goal of this work.

As for the comparison of the lossy codecs, all three implementations deliver better compression over JPEG-LS frame based approach. However, as seen in Table 6.3, the average amount of Gain(%) ($Gain_{proposed}$ in other words) is the highest for the JPEG2000 based implementation. For all four test samples, this specific implementation results higher compression gain over other two with similar memory requirements. The only exception can be seen for the sample "salesman" where $Gain_{proposed}$ is comparable with the implementation with standard JPEG. But reflecting on the unmatched dominance in compression performance in other three samples and significantly higher amount of average compression, it can be concluded that the proposed model performs best in terms of compression ratio with JPEG2000 based memory save option.

The next best choice of lossy codec for the proposed system would be JPEG since it gives better compression over near lossless JPEG-LS based system for all four samples. The amount of average $Gain_{proposed}$ is also fairly higher than near lossless JPEG-LS based implementation to call it better in terms of compression performance.

Chapter 7

Discussions and Conclusion

7.1 Summary of the whole work

In the presented work, a new method for memory efficient lossless video compression has been proposed. The framework of the proposed design is based upon standard JPEG-LS encoder with temporal extension of the MED predictor for higher compression and introduction of on-line lossy compression of the reference frame for memory reduction. These two additions in JPEG-LS based video coding structure offer a very good compression of lossless video with low memory requirement. The improvement of compression performance has been verified comparing with regular frame-by-frame JPEG-LS based video encoder as seen in the previous chapter. The prediction model used in this approach uses a combination of context based spatial and temporal predictor where context parameters control the prediction bias cancellation and error coding optimization for both the predictors (see Chapter 4). This prediction model plays the key role for achievement of such higher compression as observed during the testing phase. Moreover, the temporal predictor part uses only one past frame for error calculation keeping a very low memory requirement. Need for space to store one frame is further reduced using the new on-line lossy compression method (see Chapter 5) which resulted in higher memory optimization while encoding. The amount of memory reduction with respect to compression gain has been measured using three implementations based upon JPEG, JPEG2000 and near lossless JPEG-LS algorithms. Reflecting on the test results, all these three implementations delivered better compression gain over frame-by-frame JPEG-LS video coder, making the design a successful one. However, the amount of compression of these three implementations differ depending upon the amount of used memory reduction. As seen in the results, to obtain higher video compression from the system, use of JPEG2000 or near lossless JPEG-LS based implementation delivered relatively better result than the one with JPEG. However, use of near lossless mode of

JPEG-LS over JPEG2000 would be a better choice for higher amount video compression gain since it has a significantly lower computational complexity comparing to JPEG2000. On the other hand, in the higher amount of memory reduction region, JPEG based implementation performed much better than the other two. The outcome of the system based upon JPEG2000 has been relatively stable throughout the various memory optimization range. This observation is verified when average compression gain has been measured and compared among all three implementation keeping the memory reduction amount in a similar mid range (around 5). From the latter analysis, a rough comparison of the three implementations has been made which ranks JPEG2000 and JPEG as the first and the second choice of suitable lossy compression algorithms for the on-line memory reduction module of the proposed model. To sum up, near lossless JPEG-LS based implementation would be the most suitable choice for systems with high compression gain requirement. In contrast, systems with JPEG codec for on-line frame storage delivers usable amount of video compression in higher memory efficiency region. The outcome of JPEG2000 based models has a more stable compression outcome comparing the other two, which makes it more suitable for standard implementation.

7.2 Limitations of the design

In spite of delivering good performance regarding both compression gain and memory efficiency, the proposed system lacks some features. One of these would be the range of video categories it can handle to extract good compression. Since it has been designed to work with videos with relatively stationary backgrounds, it does not promise a very good temporal prediction in case of video samples with moving backgrounds or with very large moving objects. In such cases the intra-frame predictor is chosen more often, resulting almost similar compression comparing frame-by-frame JPEG-LS video coder model. Occurrence of such samples however does not deliver worse compression than regular JPEG-LS since temporal predictor is not used much in such cases.

The introduction of lossy compression increases the computational load of the model since it includes the encoding and decoding procedure of a full image for processing one frame. Increase in computational load depends upon the choice of used lossy codec. This problem can be reduced by using any lossy compression method with relatively lower requirement of computational effort. Such convenient change in choice of lossy coders is possible due to the model's adaptability of arbitrary lossless or lossy coders to be used in the memory module. Another option to avoid this problem is to store the raw frame for future reference. This will however take away the "memory efficient" feature from the system, but will deliver very good compression of video. No lossy compression on reference frame also means no noise in the reconstructed image, and this is the reason behind such increase in compression gain.

7.3 Future works

As seen in the previous chapter, the choice of the lossy coder for on-line frame save plays a significant role in memory efficiency. So, the framework of the proposed lossless video coder model has been tested with JPEG, JPEG2000 and near lossless mode of JPEG-LS as the lossy codec for memory module. However, the implemented framework can also be tested with other available lossy codecs in persuit of better memory optimization and compression gain.

In order to make the system usable for samples with non-stationary backgrounds and larger moving objects, the motion estimation and compensation methods[29][20] can be introduced. Since adoption of this feature would result in higher computational complexity, some lower performance lightweight version of the motion estimation and compensation algorithms can be integrated with the framework to check the adaptability of the system in terms of compression gain and computational complexity.

Although the predictor model used in the proposed design is good enough to deliver a very good compression, its performance might be improved even further by combining other existing prediction algorithms [6] [7] with the proposed model.

Bibliography

[1] M. J. Weinberger, G. Seroussi, and G. Sapiro, *"The LOCO-I Lossless Image Compression Algorithm: Principles and Standardization into JPEG-LS"*, Proc. IEEE Transactions on Image Processing, Vol. 9, No. 8, August 2000.

[2] A. Netravali and J. O. Limb, *"Picture coding: A review"*, Proc. IEEE, vol. 68, pp. 366406, 1980.

[3] K. H. Yang and A. F. Faryar, *"A contex-based predictive coder for lossless and near-lossless compression of video"*, Proc. ICIP 2000, Vol. I, Vancouver, BC, Canada, September. 2000, pp.144-147.

[4] *"JPEG-LS; Lossless and near-lossless coding of continuous tone still images (JPEG-LS)"*, ISO/IEC JTC 1/SC 29/WG 1 FCD 14 495, July 1997.

[5] M. J.Weinberger, J. Rissanen, and R. Arps, *"Applications of universal context modeling to lossless compression of gray-scale images"*, Proc. IEEE Trans. Image Process., vol. 5, no. 4, pp. 575586, Apr. 1996.

[6] X. Wu and N. Memon, *"Context-based, adaptive, lossless image coding"*, IEEE Trans. Communications, vol. 45, no. 4, pp. 437-444, Apr. 1997.

[7] R. Ansari, N. Memon and E. Ceran, *"Near-lossless Image Compression Techniques"*, Journal of Electronic Imaging. 7(03):486-494, July 1998.

[8] G. C. K. Abhayaratne and D. M. Monro, *"Embedded to lossless coding of motion compensated prediction residuals in lossless video coding"*, ISO/IEC 14495-1 and ITU Recommendation T.87, 1999.

[9] *"Information technology Lossless and near-lossless compression of continuous-tone still images Baseline"*, Proc. SPIE (4310) Visual Commumications and Image Processing (VCIP), Jan. 2001.

[10] ISOISO/IEC Document 13818-2, *"Information technology - Generic coding of moving pictures and associated audio information: Video"*, International Standard, 2000.

[11] T. Wiegand, G. J. Sullivan, G. Bjontegaard and A. Luthra, *"Overview of the H.264/AVC Video Coding Standard"*, in IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, No. 7, pp. 560-576 July 2003.

[12] P. G. Howard and J. S. Vitter, *"Fast and efficient lossless image compression"*, Proc. IEEE DCC 93. Los Alamitos, CA: IEEE Comput. Soc. Press, 1993, xiii+505, pp. 351360.

[13] Gary J. Sullivan, Pankaj Topiwala and Ajay Luthra, *"The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions"*, SPIE Conference on Applications of Digital Image Processing XXVII, Special Session on Advances in the New Emerging Standard: H.264/AVC, August, 2004.

[14] *"Information TechnologyJPEG2000 Image Coding System"*, ISO 15444-1 Final Committee Draft (FCD) Version 1.0 March, 2000.

[15] A. N. Skodras, C. A. Christopoulos and T. Ebrahimi, *"JPEG2000: The Upcoming Still Image Compression Standard"*, Proc. 11th Portuguese Conference on Pattern Recognition (RECPA00D 20; invited paper), pp.359-366, May. 2000.

[16] Xiaolin Wu and Memon, N., *"CALIC-a context based adaptive lossless image codec"*, Proc. 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing, 1996, pp. 18901893.

[17] D. Brunello, G. Calvagno, G.A. Mian, R. Rinaldo, *"Lossless compression of video using temporal information"*, Proc. IEEE Transactions on Image Processing, vol. 12. no. 2, pp.132-139, Feb. 2003.

[18] Sonja Grgic, Marta Mrak, Mislav Grgic, *"Comparison of JPEG Image Coders"*, Proc. 3rd International Symposium on Video Processing and Multimedia Communications, VIPromCom-2001, Zadar, Croatia, 2001, pp. 79-85.

[19] The independent JPEG Group, `http://www.ijg.org/`

[20] J. Jain and A. Jain, *"Displacement Measurement and Its Application in Interframe Image Coding"*, IEEE Trans. Comm., vol. 29, no. 12, pp. 1799-1808, 1981.

[21] The OpenJPEG Library, `http://www.openjpeg.org/`

[22] N.D. Memon and K. sayood, *"Lossless compression of video sequences"*, IEEE Trans. Communications, vol. 44, no. 10, pp.1340-1345, Oct. 1996.

[23] Arizona State University online repository `http://trace.eas.asu.edu/yuv/`

[24] xipf.org test media repository, `http://trace.eas.asu.edu/yuv/`

[25] Avysinth homepage, `http://avisynth.org/mediawiki/Main_Page`

[26] Gregory K. Wallace, *"The JPEG Still Picture Compression Standard"*, Multimedia Engineering, Digital Equipment Corp., submitted for publication in IEEE Transactions on Consumer Electronics, DECEMBER 1991.

[27] A. N. Netravali and B.G. Haskel, *"Digital Pictures: Representation, Compression, and Standards"*, 2nd Ed., Plenum Press, 1995.

[28] G. J. Sullivan and R. L. Baker, *"Motion compensation for video compression using control grid interpolation"*, Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., 1991, vol. 4, pp. 27132716.

[29] B. Martins and S. Forchhammer, *"Lossless compression of video using motion compensation"*, in Proc. IEEE DCC 98, xvi+589, Los Alamitos, CA, USA, 1998, p. 560.

[30] M. Chan, Y. Yu, and A. Constantinides, *"Variable size block matching motion compensation with applications to video coding"*, Proc. IEEE, vol. 137, no. 4, pt. 1, pp. 205212, Aug. 1990.

[31] ITU and CCITT, *"Information Technology Digital Compression and Coding of Continuous-Tone Still Images Requirements and Guidelines"*, ISO/IEC IS 10918-1, ITU-T T.81, SEPTEMBER 1992.

Declaration

All the work contained within this thesis, except otherwise acknowledged, was solely the effort of the author. At no stage was any collaboration entered into with any other party.

(Debasish Chanda)