ADDIS ABABA UNIVERSITY
ADDIS ABABA INSTITUTE OF TECHNOLOGY
Department of Electrical and Computer Engineering

A Thesis Submitted to Addis Ababa Institute of Technology, Addis Ababa University, In Partial fulfillment for the Degree of Master of Science in Computer Engineering.

Performance Evaluation of Compression and Security Algorithms for Audio-Video Streaming and Implementation of Selected Algorithms.

Advisor

Dr. Kumudha Raimond

By
Andargachew Gobena

April 28, 2011

ADDIS ABABA UNIVERSITY
ADDIS ABABA INSTITUTE OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Performance Evaluation of Compression and Security Algorithms for Audio-Video Streaming and Implementation of Selected Algorithms

By

Andargachew Gobena

ADDIS ABABA INSTITUTE OF TECHNOLOGY

APPROVAL BY BOARD OF EXAMINERS

_____	_____
Chairman	Signature
Dr. Kumudha Raimond _____	
Advisor	Signature
_____	_____
Internal Examiner	Signature
_____	_____
External Examiner	Signature

Declaration

I, the undersigned, declare that this thesis work is my original work, has not been presented for a degree in this or any other universities, and all sources of materials used for the thesis work have been fully acknowledged.

Andargachew Gobena _____

Name Signature

Place: Addis Ababa

Date of submission: April 28, 2011

This thesis has been submitted for examination with my approval as a university advisor.

Dr. Kumudha Raimond _____

Advisor's Name Signature

Acknowledgement

First and foremost, I would like to express my deep and sincere gratitude to my advisor Dr. Kumudha Raimond. I could not have imagined having better advisor and mentor for my MSC study. She provided great encouragement and guide, without which this thesis work would have been lacking.

My thanks go to my family, specially my sister, Mebrate Mekonen, for the great care throughout my MSC study.

My special thanks go to Natnael Chuta and Wehib Abubeker.

Table of Contents

List of tables

List of Figures

Abbreviations

3G	Third Generation of Mobile Telephony
3GPP	3rd Generation Partnership Project
3GS	Third Generation of Mobile Systems
AAC	Advanced Audio Codec
AAC	Advanced Audio Codec
AC3	Audio Compression 3
ADC	Analogue to Digital Converter
AES	Advanced Encryption Standard
AIFF	Audio interchange File Format
ALE	Apple Lossless Encoder
ALG	Algorithm
AMR	Adaptive Multi Rate
ASCII	America Standard Code for Information Interchange
AU	Audio
AVC	Advanced Video Codec
AVI	Audio Video Interleaved
BMP	(Windows) Bit Map Image
B-PICTURE	Bi-directionally predicted Pictures
CD	Compact Disk
CDMA	Code Division Multiple Access
CIF	Common Interchange Format
CL	Command Line
CODEC	Compressor - Decompressor
CPU	Central Processing Unit
CS&Q	Coarser sampling and/or quantization
DAC	Digital To Analogue Converter
DCT	Discrete cosine transform
DMC	Digital Media Converter
DSP	Digital Signal Processing
DVD	Digital Video Disc or Digital Versatile Disc

EC	Embedded Compression
ECG	Electrocardiogram
EPS	Encapsulated Postscript
EXR	OpenEXR image format
FLAC	Free Lossless Audio Codec
FLV	Flash Video
FPS	Frames Per Second
GB	Giga Bytes
GHZ	Giga Hertz
GIF	Graphics interchange format
GNU	GNU is Not Unix
GPL	General Public License
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HD	High Definition
IBT	Insignificant Bit Truncation
I-PICTURE	Intra Coded Pictures
ISO/IEC	International Organization for Standardization/ International Electrotechnical Commission
ITU	International Telecommunication Union
JBIG	Joint Bi-level Image Experts Group
JDK	Java Development kit
JMF	Java Multimedia Framework
JPEG	Joint Photographic Expert Group
KB	Kilo bytes
KBPS	Kilo-Bits-Per-Second
KHZ	Kilo Hertz
L	Lossy codec
LL	Lossless codec
LPC	Linear Predictive Coding
LSB	Least Significant Bit
LZOP	Lempel-Ziv-Oberhumer Packer

LZW	Lamperl-Ziv-Welch
M4A	MPEG 4 Audio
MB	Mega bytes
MNG	Multiple-image Network Graphics
MP3	MPEG layer 3
MPEG	Moving pictures Expert group
NTSC	National Television System Committee
PAL	Phase Alternating Line
PC	Personal computers
PCM	Pulse Coded Modulation
PCX	Picture Exchange Format
PDF	Portable Document Format
PKI	Public Key Infrastructure
PNG	Portable Network Group
PPI	Pixel Per Inch
P-PICTURE	Predicted Pictures
PSD	Photoshop Data File
PSNR	Peak To Peak Signal To Noise Ratio
RAM	Random Access Memory
RGB	Red Green Blue
RIFF	Resource Interchange File Format
RLE	Run Length Encoding
RMS	Root Mean Square
SBT	Significant Bit Trancation
SWF	ShockWave Flash
TGA	Targa Image File
TIFF	Tagged Image File Format
TV	Television
VCD	Video Compact Disc
VCEG	Video Coding Experts Group
VoIP	Voice over IP
VQ	Vector Quantization

VR	Virtual Reality
WAR	Windows Archiver
WAV	Waveform Audio
WMA	Windows Media Audio
WMV	Windows Media Video

Abstract

Audio, Image and Video are multimedia data that are ubiquitous in the digital world. The raw forms of such data produced by capturing devices contain some unnecessary details that can be removed so that only usable/observable contents are maintained for storage. Other contents are temporally or spatially redundant so that long repetitions can be represented in a more storage efficient way. The process of reducing the size of data is called compression. The compressed data are reprocessed to create the original or near original data through a process of decompression. For multimedia data, there are numerous algorithms and programs to perform the compression and decompression process that are created, developed and maintained by different commercial and non-commercial organizations converting the processed data into custom formats. The diversity in storage formats makes it difficult for users to definitively adapt a uniform standard for data management. Processing time, storage requirement, and bandwidth consumption of the data vary among the representations. This thesis work independently treats Audio, Image and Video data and carries out performance evaluation based on defined set of parameters. Data files have been prepared and subjected to compression and decompression programs and values for the performance parameters are registered under uniform test environment for each data type. The observations point out the difficulty in ultimately deciding the best algorithms in each test case due to the trade off that exists between the defined parameters. So many real life applications have been considered and optimal algorithms have been suggested based on affordable tradeoffs. A new algorithm for image compression has been designed, implemented and tested. Security test has also been done to observe the effect of applying security features to the data. Securing data imposes additional processing time and has little effect on the storage size.

Key words: Compression, Decompression, Codec, Audio, Image, Video, Multimedia, Performance, Evaluation, Lossy, Lossless, Security.

Chapter 1

Introduction

1.1 Background Information

The advancement in digital technology has paved a way for the use of computers in almost all aspects of life. The real world data that humans deal with are analogue in form such as temperature, sound, voltage and everything we biologically sense in our environment. In order to be processed with the help of computers, these data need to be converted into digital form. The amount of digital space required to store the converted data depends on the future processing and utilization of the data. For instance, sound generated during speech can be typed in and stored in a text format for reporting purpose or can be recorded in audio format for later playback and/or editing. Audio, image and video data are generally referred to as multimedia data when represented for later playback (audio and video) or display (image). Multimedia data take relatively huge computer storage space that makes it expensive to manage huge amount of such information content. In the era of the internet, where data heavily traverse computer networks, efficient use of available data communication infrastructure for timely delivery from source to destination is a compelling requirement.

To reduce the storage size and bandwidth requirement, it is necessary to keep multimedia data as small as possible, preserving usable contents. Due to the characteristics of human visual and hearing sensations, these data in their original form can be subjected to a process where undetectable contents can be removed and smaller data size can be attained. For instance, a source of sound may generate range of frequencies that are beyond or under the hearing threshold of the human auditory system. Those undetectable ranges are obvious candidates for removal during storage. Similarly, color variations in digital images that are not normally detected by the eye can be manipulated for efficient

storage and transmission. Compression is the process of converting (any) input data and generating a lesser (compressed) one for better storage. The compression process may not necessarily remove contents, but it can represent the data in a different format that reduces the size. Decompression is the process where the original (or near original) data is recovered by reversing the compression process.

A lossless compression preserves all the contents from the original data but attains a lesser size by changing the output format. A lossy compression is a compression where some contents from the original data are irreversibly removed. In both cases, the corresponding decompressor processes the output from the compressor to recover the intact form of the input to the (lossless) compressor or an approximation of the input to the (lossy) one. Since a compressor is always paired with an associated decompressor, the term codec is used to refer to either or both of them.

Unlike lossless compressors, where there is no loss of quality, in the case of using lossy compressors, the intended purpose of the output data dictates the level of lost quality that can be tolerated due to the compression algorithm. For instance, Data for entertainment purposes can tolerate losses in the output data while in medical imaging applications, the alteration of certain contents due to the approximation made in the recovery process can result in wrong pathology analysis.

Securely storing and transmitting multimedia data is also important in certain applications. Archiving such sensitive data or transmitting surveillance of secret scenes needs to be secured to protect the confidentiality of system owners.

1.2 Problem Description (Motivation, Statement of the Problem)

As mentioned in section 1.1, multimedia data consumes a huge amount of memory and requires a large amount of transmission bandwidth in its 'raw' format. There are several compression algorithms used to alleviate these limitations that are used in various multimedia applications such as

Voice over IP (VOIP), Video conferencing and others. Storage and transmission constraints can limit the use of multimedia applications in countries where there are poor communication infrastructures. Securing the stored and streamed data poses additional area of concern in such cases to avoid or be aware of data alteration and to keep the integrity of the data. For systems that use codec software programs, selection of appropriate products to meet various needs is a challenge. Some users or systems may require fast processing and others may prioritize better storage or bandwidth efficiency. To this end, comparison of as many codecs as possible and parameterization of the codecs are solution providers in the process of selecting a product for use.

1.3 Objective of the Thesis

1.3.1General Objective

The objective of this thesis work is to study existing compression/decompression and security algorithms for multimedia streaming and make evaluation of their performance and security.

1.3.1 Specific Objective

- Review existing compression algorithms for multimedia data.
- Study the security implications of the algorithms.
- Evaluate the performance of the Algorithms.
- Implementing a chosen algorithm(s).

1.4. Methodology

Literature survey: A thorough study will be made on literatures relating to multimedia data representation, general compression philosophy, multimedia specific compression and security algorithms.

Survey of communication infrastructure: A partial survey will be made on locally available network technology in most applications relevant to the topic.

Performance evaluation: The compression and security algorithms are evaluated based on processing time, transmission time and possible area of application.

Selection and implementation of some algorithms: Criterion will be set for selecting and implementing one of the tested algorithms.

1.5 Scope

In this thesis work, performance evaluation tests have been done on several compression, decompression and security algorithms. The tests are conducted on stored audio, image and video data in a uniform environment for each data type. However, the codecs are not applied on the fly (in real time). An ultimate comparison of the codecs would have served as a benchmark if the algorithms are implemented under the same implementation environment. But in this thesis work, black box testing has been done on off-the-shelf software packages still providing a ground up on which preferences can be made.

1.6 Contribution

This thesis work makes multimedia algorithm comparison over a wide range of formats, data types (audio, image, and video) and compression types. Testing methodology has been proposed in two categories. In the first category, approach I, several algorithms from literatures are implemented and the results compared. The result from [11] has pointed out that the proposed algorithm has unpredictable effect on the recovered data after decompressing the compressed data. The authors have shown that the level of quality compromise is more sever on the background color than the objects in the test image. The result from this thesis work shows no such variation.

In the second category, approach II, performance evaluation tests have been done for the three data types mentioned above, by applying software

packages for the test codecs that use either lossy or lossless logic under similar test environment.

This thesis work proposes using general encryption tools for multimedia data. The use of this kind of security has little effect on the performance of the codecs but it is possible to gain the benefits of securing sensitive multimedia data.

Users of the test multimedia algorithms can complement their data management process by optimizing the tradeoffs between the evaluation parameters.

1.7 Thesis outline

The rest of the thesis is organized as follows. In chapter two literatures related to this thesis work are reviewed. It has been observed that researchers are still working on ways of processing multimedia data with better performance than ever. The results of performance evaluation done by several literatures have also been reviewed. Chapter three discusses basic digital signal processing (DSP) concepts. In addition, the various kinds of data representations such as audio, image and video data are discussed. List of file formats and processing algorithms have been provided and with brief definition for each. In chapter four, principles of some compression and decompression algorithms have been discussed. The chapter covers the theoretical philosophy behind data compression and how they work. Chapter five is concerned with performing the actual performance evaluation tests and result analysis. In this part of the document, testing environment setup, methodology used, test categories, and the difficulties encountered during the entire work period have also been discussed. Finally, chapter six provides summery and conclusion for the tests performed in chapter 5.

Chapter 2

Literature review

2.1 Literature Survey

Among the data types stored in digital computers, multimedia data are among the most resource intensive ones. This type of contents can hinder or limit their use without proper processing before storing them for future use. Compression-decompression (Codec) algorithms are used to minimize their size for cost effective management of such data. Over the years, codecs have been developed by researchers and organizations to make them suitable for large scale use. The developed algorithms vary in the processing time, percentage of storage savings and quality of the processed data. Since the existing algorithms still incur poor performance during processing, delay while transmission over low bandwidth data communication networks, and costly memory space, several researches are underway to device algorithms that mitigate these drawbacks.

The ubiquitous nature of multimedia data such as audio, image and video, has drawn significant attention in the world of computing. Developers of software products implement multimedia processing software in variety of ways to meet a specific need of their custom products or targeting a market area. Better ways of representing multimedia signals and processing them in an efficient way is an ever increasing area of research in the scientific community.

Tarif Riyad Rahman and Miftahur Rahman[1] proposed two lossy compression algorithms. Their algorithm works by combining several groups of bytes in to a lesser group of bytes, eliminating the least significant bit information, which is assumed during recovery of the data. The error depends on the number of bits assumed. In particular, the first algorithm combines two bytes into one byte by combining the four most significant bits of each raw data and eliminating the four least significant bits of each byte. The second algorithm is similar but combines four raw

data bytes into three bytes and eliminates two bits of information from each input byte. The authors have mentioned that, while the results of the first algorithm is expected to be of a better quality recovered content, the opposite was observed in some test images. But in this experiment, the result is as expected and consistent with theoretical expectations; i.e. the more bits are lost, the lesser the quality and vice versa.

An Inter-Frame coding algorithm based on vector quantization (VQ) is mentioned by Meihua Gu and Ningmei Yu[2]. They have discussed three inter coding modes according to the motion type of the image block. Not-encode mode is adopted by static block, residual mean coding mode is adopted by the smooth motion block, VQ and mean encode mode are adopted by general motion block. The motion type of the image block is determined by the moving threshold TH and smoothing threshold T. They have claimed that the quality and bit rate can meet the visual and bandwidth requirements in the application of video telephony communications.

Jaemoo Kim[3], in the paper "A Lossless Embedded Compression Using Significant Bit Truncation For High Definition (HD) Video Coding", addressed the fact that increasing the image size of a video sequence aggravates the memory bandwidth problem of a video coding system. It has been mentioned in the paper that despite many embedded compression (EC) algorithms proposed to overcome this problem, no lossless EC algorithm able to handle HD size video sequences has been proposed thus far. A lossless EC algorithm for HD video sequences and related hardware architecture is proposed. The proposed algorithm consists of two steps. The first is a hierarchical prediction method based on pixel averaging and copying. The second step involves significant bit truncation (SBT) which encodes prediction errors in a group with the same number of bits so that the multiple prediction errors are decoded in a clock cycle.

In the paper "Simultaneous Arithmetic Coding and Encryption Using Chaotic Maps," [4] Kwok-Wo Wong, Qiuzhen Lin, and Jianyong Chen proposed a simultaneous compression and encryption scheme. Traditionally, source coding and encryption are performed one after another to reduce the data volume while maintaining information secrecy. A typical example is the compression of a private photo using Joint Photographic Expert Group (JPEG) format and then the encryption of the compressed file using the Advanced Encryption Standard (AES). However, there is an increasing interest in simultaneous compression and encryption. This can be achieved by either embedding compression into encryption algorithms or adding cryptographic sense in compression schemes. In this finding, the authors proposed a scheme for the simultaneous compression and encryption of message sequences with multiple symbols. Compression is achieved by iterating a multisegment piecewise linear chaotic map, whereas encryption is realized by changing the chaotic map model continuously using a secret key, without affecting the essence of arithmetic coding.

From usage point of view, there are several challenges that are faced by end users and applications that make use of the data. One such challenge is compatibility of the various multimedia data representations. Some formats of the same data are readable by some applications and not by others. A universally usable data format is one necessity for multimedia processing applications.

Another challenge faced by applications or users is the performance of the processing algorithms. As these kinds of data consume huge amount memory space relative to text and database files, special focus is worth given to the processing time and transfer of the data over networks (which depends on the size of actual data represented in digital form). In this regard, several people have compared subsets of multimedia codecs with

respect to measurable parameters of processing time and storage requirements.

Jeff Gilchrist has made an extensive testing benchmark on state of the art lossless compression algorithms [11] for text, exe, sound, and image files. He has collected these compression algorithms from the respective developers of the tools and evaluated the compression time, decompression time and compression ratio. In his image test, he has taken a file already compressed with JPEG and applied several lossless archiving tools and tabulated the results. His other tests include a WAVeform audio (WAV) file and Tagged Image File Format (TIFF) image files with similar procedures.

Based on Gilchrist experiment, among the three JPEG files used in the test, zip 2.3 has mostly been the best in compression and decompression time while Allume JPEG b2 resulted in best compression ratio. The result shows that the compressing and decompressing varies for different tools based on the color composition of the test images, i.e. no single tool is best for each of the evaluation parameters. With respect to audio, he has tested speech and music files and found that Monkey's Audio tool is the best in compression ratio and overall (compression, decompression and transferring aver a network) and Lempel-Ziv-Oberhumer Packer (LZOP) to be the best at compressing and decompressing the input audio files. Test of eight TIFF images resulted in ERI32 5.1fre tool to be the best in compression ratio and overall and LZOP is best at compression and decompression times.

Video codec comparison by the CS MSU Graphics & Media Lab Video Group [15] has done the test on several frames extracted from movie scenes. The comparison focused on the effect of the drop-frame features of lossy codecs from several sources. The group has defined a logarithmic comparison parameter called peak-to-peak signal to noise ratio (PSNR)

$$d(x,y) = 10 * \log_{10} \frac{255^2 * n^2}{\sum_{i=1,j=1}^{n,n} (x_{ij} - y_{ij})^2} \quad (2\text{-}1)$$

Where, (x,y) are pixel coordinates and 'n' is the frame/image dimension.

Based on this parameter, they have made extensive analysis on the test frames with respect to PSNR and visual analysis. This parameter is similar to the root mean square value defined in chapter 5 of this document since they are based on the square of the differences between original and recovered data sets. However, as shown in (equation 2-1), the PSNR is inversely related to the quality (differences between original and recovered files). In some of the conclusions, it was mentioned that better PSNR does not necessarily mean a better visual result. This is an indication of the fact that the characteristics of the sense of vision cannot ultimately be represented as a numeric quantity. So the numerical value of the measure of the quality can only be useful for entry comparison and should be followed by the actual observer's vision for final quality decisions. In [16] and [17], the authors have made comparison tests for the most common audio formats (five and three, respectively). In [16] it was mentioned that, sound quality should not be the reason to rule out any codec among the ones tested (MP3, OGG, AAC, WMA, musepack). But this decision is valid only if we take the codecs mentioned and we intend to use the contents for entertainment purposes. In [17], the codecs used are for MP3, OGG and WMA and showed that, WMA is superior in quality.

Conclusion

As the survey of the literatures reveals, there are numerous researches going on to optimize the processing time and storage requirement with more emphasis on the later. Lossless algorithms have very limited gain in storage size of compressed data and since they are based on the simple logic of exploiting the repetition available in most data streams, they have

little promise for innovation. Lossy algorithms are promising areas for the creation of better performance codecs for multimedia data and most researches are focused on creating a better solution by exploiting the possibility of removing or wisely preserving imperceptible contents in the original content. From the survey, we can see that while some researches focus on the creation of new way of processing multimedia digital data, others focus on making comparison tests to reveal the nature and performance of available algorithms.

This thesis work complements the above mentioned comparison tests by using the best lossless archivers from [11] and proceeding with lossy audio, image and video codecs. In addition, computationally simple algorithms have been designed and implemented for lossy image processing. The comparison tests performed in this thesis work measure several evaluation parameters (processing time, storage size, transmission time), design test methodology and propose viable selection criteria for variety of real life applications.

Chapter 3

Digital Signal Processing

3.1 Introduction [12]

DSP, as the term suggests, is the processing of signals by digital means. A signal in this context can mean a number of different things. Historically the origins of signal processing are in electrical engineering, and a signal here means an electrical signal carried by a wire or telephone line, or perhaps by a radio wave. More generally, however, a signal is a stream of information representing anything from stock prices to data from a remote-sensing satellite. The term "digital" comes from "digit", meaning a number (we count with our fingers - our digits), so "digital" literally means numerical. A digital signal consists of a stream of numbers, usually (but not necessarily) in binary form. The processing of a digital signal is done by performing numerical calculations.

3.1.1 Analog and Digital Signals [12]

In many cases, the signal of interest is initially in the form of an analog electrical voltage or current, produced for example by a microphone or some other type of transducer. In some situations, such as the output from the readout system of a compact disc (CD) player, the data is already in digital form. An analog signal must be converted into digital form before DSP techniques can be applied. An analog electrical voltage signal, for example, can be digitized using an electronic circuit called an analog-to-digital converter (ADC). This generates a digital output as a stream of binary numbers whose values represent the electrical voltage input to the device at each sampling instant.

3.1.2 Signal Processing [12]

Signals commonly need to be processed in a variety of ways. For example, the output signal from a transducer may well be contaminated with unwanted electrical "noise". The probes attached to a patient's chest when

an Electrocardiogram (ECG) is taken measure tiny electrical voltage changes due to the activity of the heart and other muscles. The signal is often strongly affected by "mains pickup" due to electrical interference from the mains supply. Processing the signal using a filter circuit can remove or at least reduce the unwanted part of the signal. Increasingly nowadays, the filtering of signals to improve signal quality or to extract important information is done by DSP techniques rather than by analog electronics.

3.2 Multimedia Data acquisition

3.2.1 Sampling [45] [46]

In order to perform any form of processing by digital computers, the signals must be reduced to discrete samples of a discrete-time domain. The operation that transforms a signal from the continuous time to the discrete time is called sampling, and it is performed by picking up the values of the continuous-time signal at time instants that are multiple of a quantity T, called the sampling interval. The quantity Fs = 1/T is called the sampling rate.

During the digital-to-analog conversion, the numbers will be converted again into voltages. By close examination of Fig.3-1 we will see that the waveform resulted from the digital-to-analog conversion won't be perfect, as it won't have all the points from the original analog signal, just some of them. In other words, the digital-to-analog converter (DAC) will connect all the points captured by the ADC; any values that existed originally between these points will be suppressed. The sampling is such that if the sampling rate is too high, the output quality will be close to perfection, but a lot of storage space is needed to hold the generated data (i.e., the generated file will be very big); if the sampling rate is too low, the output quality will be bad. Nyquist Theorem states that the sampling rate on analog-to-digital conversions must be at least two times the value of the highest frequency we want to capture.

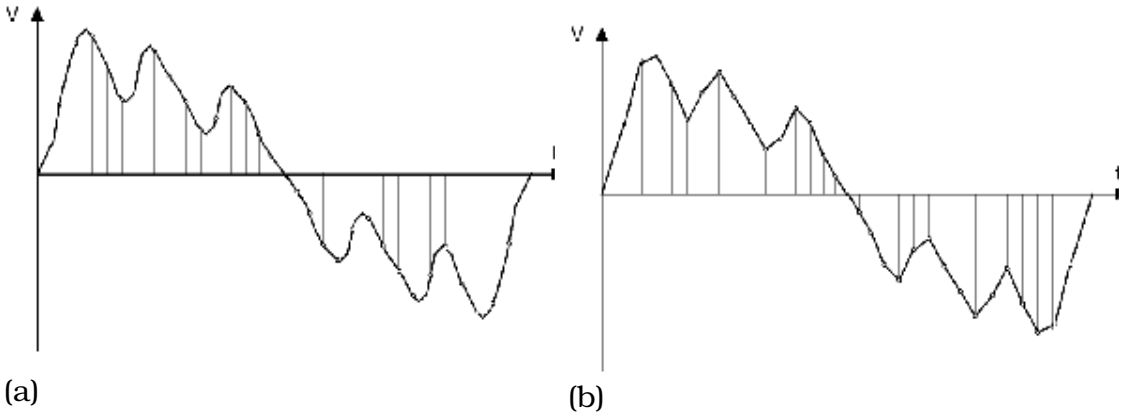


Figure 3-1: Analogue signal. (a) before ADC and (b) after DAC

3.2.2 Encoding [46]

The value of each sampled point will be stored on a fixed-length variable. If this variable uses eight bits, this means it can hold values from 0 to 255 (2^8 = 256). If this variable uses 16 bits, this means it can hold values from 0 to 65,535 (2^16 = 65,536). And so on.

So, if we are using an 8-bit ADC, the lowest value will be zero and the highest value will be 255. Likewise, when a 16-bit ADC converter is used, the lowest value will be zero and the highest value will be 65,535.

What the ADC does is to divide the "y" axis in "n" possible parts between the maximum and the minimum values of the original analog signal, and this "n" is given by the variable size. If the variable size is too small, what will happen is that two sampling points close to each other will have the same digital representation, thus not corresponding exactly to the original value found on the original analog signal, making the analog waveform available at the DAC output to not have the best quality. Hence, the highest the variable size, the better the quality will be, but more storage space will be needed. Using a 16-bit variable will require twice the storage space if an 8-bit variable was used, but the quality will be far better.

3.2.3 Processing

Once the analogue signal has been captured and represented in a digital form, it is stored on a digital media such as CD or on the local computer's hard disk. The processing stage starts by reading the binary data from a storage media to perform the necessary signal processing operations. One such processing is to decrease the size of the input digital media file by applying codec functions generating lesser size output data.

3.2.4 Decoding

After the processing stage, the result is ready to be output to several different targets. In this case, video frame sequences are sent to display devices like the computer's monitor or Television (TV) screen and audio signals, after being converted to analogue signal, are sent to a speaker. The decoding mostly gives out the resemblance of the encoded signal with varying level of approximation based on both the encoding, processing and decoding stage design.

3.3 Audio Data Representations [47]

Sound consists of audible variations in air pressure. Microphones convert variation in air pressure into a varying voltage. To represent sound digitally, we must convert this varying voltage into a series of numbers representing its amplitude. This is what constitutes the analog-to-digital conversion. Audio data consisting of such numbers is said to be in pulse code modulation format (PCM). Audio data is often stored in other formats, usually in order to compress it, but it starts off in PCM format.

The air pressure variation, and therefore the corresponding voltage produced by a microphone, is continuous in two-dimensions. That is, the values vary continuously, and they exist at every point in time. However, a digital system such as a computer cannot directly represent a continuous signal. Instead, it must measure the signal at a finite set of discrete times. This is sampling stage. Furthermore, it must make use of a finite number

of discrete amplitude levels. This is known as quantization. The number of levels used is known as the resolution. The resolution is usually expressed in bits, that is, as the base-2 logarithm of the actual number. A system with a resolution of 8 bits makes use of 2^8 = 256 levels. A system with 16 bit resolution makes use of 2^16 = 65,536 levels. The sampling rate and resolution determine the quality of the digital representation of the sound. "CD-quality" sound has a resolution of 16 bits and a sampling rate of 44,100 samples per second.

Several ways of processing audio data are created to compress the large audio data with varying level of quality and processing time. Some common ones are discussed below. The detailed technical discussion of these and other formats mentioned in this thesis work is beyond scope.

MP3 (MPEG Layer 3) [18]: MP3 uses two compression techniques to achieve its size reduction ratios over uncompressed audio - one lossy and one lossless. First it throws away what humans can't hear anyway (or at least it makes acceptable compromises), and then it encodes the redundancies to achieve further compression. However, it's the first part of the process that does most of the grunt work, requires most of the complexity. In brief, the MP3 encoding process can be subdivided into a handful of discrete tasks:

- Breaking the signal into smaller component pieces called “frames," each typically lasting a fraction of a second.
- Analyzing the signal to determine its "spectral energy distribution." On the entire spectrum of audible frequencies, find out how the bits will need to be distributed to best account for the audio to be encoded. Because different portions of the frequency spectrum are most efficiently encoded via slight variants of the same algorithm, this step breaks the signal into sub-bands, which can be processed independently for optimal results.

- The encoding bitrate is taken into account, and the maximum number of bits that can be allocated to each frame is calculated. For instance, if we are encoding at 128 Kilo-bits-per-second (kbps), we have an upper limit on how much data can be stored in each frame (unless the encoding uses variable bitrates). This step determines how much of the available audio data will be stored, and how much will be left out.

- The frequency spread for each frame is compared to mathematical models of human psychoacoustics, which are stored in the codec as a reference table. From this model, it can be determined which frequencies need to be rendered accurately, since they'll be perceptible to humans, and which ones can be dropped or allocated fewer bits, since we wouldn't be able to hear.

- The bitstream is run through the process of Huffman coding, which compresses redundant information throughout the sample. The Huffman coding does not work with a psychoacoustic model, but achieves additional compression via more traditional means.

- The collection of frames is assembled into a serial bitstream, with header information preceding each data frame. The headers contain instructional "meta-data" specific to that frame.

AAC (Advanced Audio Codec)[19][20]: is an audio codec standardized by the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) Moving Picture Experts Group (MPEG) committee as an improved but non-backward-compatible alternative to MP3. Like MP3, AAC is intended for high-quality audio (like music). Advanced audio codec plus is a version of the AAC coder that is used to provide enhanced audio quality at high frequencies.

It can encode audio at any bitrate. (MP3 is limited to a fixed number of bitrates, with an upper bound of 320 kbps.) AAC can encode up to 48 channels of sound, although not practically used.

Dolby Digital/AC3 (Audio Codec 3)[21]: the Dolby Digital format is a digital audio encoding standard developed by Dolby Labs. For consumer electronics, the Dolby Digital format relies on a compression algorithm called Audio Coding 3 (AC3). It uses six independent audio channels:

- A central speaker: generally placed atop or above the screen, for reproducing dialogue.
- Two audio tracks for the front speakers, for accentuating the context of the sound coming from the central speaker.
- Two channels for the rear speakers, used for reproducing noise and the sound environment, in order to create ambiance.
- A channel for low frequencies (a subwoofer) for amplifying special effects (like explosions and earthquakes).

AMR (Adaptive Multi-Rate) [22]: this audio codec is a patented audio data compression scheme optimized for speech coding. AMR was adopted as the standard speech codec by 3rd Generation Global Partnership Project (3GPP). AMR is also a file format for storing spoken audio using the AMR codec.

M4A (MPEG layer 4 Audio) [23]: M4A is an audio file which is compressed using MPEG-4 technology which is an algorithm with lossy compression. It is primarily associated with "MPEG-4 Audio Layer" and files in this extension are the audio layer of MPEG-4 movies (non-video). It aims to overtake mp3 and become the new standard in audio compression. It is very similar to mp3 in many ways but developed to have better quality in a same or even lesser file size. M4A format was first introduced by Apple. The format type is also known as the Apple lossless Encoder (ALE).

OGG-Vorbis [24]: Ogg Vorbis is a lossy audio compression format which is totally open source and with a free license. It was developed to replace all the proprietary formats currently used for audio compression (MP3, WMA, etc.).

WMA (Windows Media Audio) [25]: it is an audio data compression technology developed by Microsoft. The name can be used to refer to its audio file format or its audio codecs.

AIFF (Audio Interchange File Format) [26]: it is a file format used by Macintosh computers and Silicon Graphics Incorporated to store and transmit high-quality audio data such as music. It can store monaural, stereo or multi-channel sound as used for soundtracks. AIFF files are uncompressed. Though the AIFF file was designed for Macintosh computers, the format can be read by personal computers (PC) as well, just as wave files can be read by Macintosh computers.

FLAC (Free Lossless Audio Codec) [27]: is a file format for audio data compression. As the name suggests, it is a free (fully open to the public to be used for any purpose free of cost) and lossless (incurs no loss of information) audio Codec (program capable of encoding and decoding digital data).

AU [28]: the standard audio file format used by Sun, UNIX and Java.

3.4 Image data representation [48]

Digital imaging is the art of making digital images – photographs, printed texts, or artwork - through the use of a digital camera or image machine, or by scanning them as a document. Each image is compiled of a certain amount of pixels, which are then mapped onto a grid and stored in a sequence by a computer. Every pixel in an image is given a tonal value to determine its hue or color.

In digital imaging, the tonal value of each pixel is represented in binary code. The binary digits for each pixel are called "bits," which are read by the computer to determine the analog display of the image. The number of pixels-per-inch (ppi) is a good indicator of the resolution, which is the ability to distinguish the spatial detail of the digital image.

The bit-depth and pixel measurement of the picture relate to the colors viewable in the image, and determines the size of the image file on a computer. Images with only two pixel shades – black and white – are binary. Grayscale images are typically displayed in 8-bit mode, which are 256 shades of gray. In digital imaging 24-bit mode, which represents true color, is generally the maximum available mode due to monitor limitations. Both of these ranges extend beyond the sensitivity of the human eye.

The dynamic range of an image is the number of shades of gray or color that can be included in that image. It is the range of tone between the darkest and lightest colors. A higher dynamic range brings with it more potential shades represented but does not necessarily correlate with the amount of tones that are reproduced. An image may have a broad dynamic range, but a smaller amount of tones represented. Likewise, in digital imaging an image may have more tones, but not as wide of a dynamic range. This can have an effect on the details within the image.

There are a wide range of options for storing digital images on computers. Some common ones are briefly listed below.

GIF (Graphics Interchange Format) [29]: This format supports up to 8 bits per pixel thus allowing a single image to reference a palette of up to 256 distinct colors. The colors are chosen from the 24-bit Red-Green-Blue (RGB) color space. It also supports animations and allows a separate palette of 256 colors for each frame. GIF images are compressed using

the Lempel-Ziv-Welch (LZW) lossless data compression technique to reduce the file size without degrading the visual quality.

JPEG (Joint Photographic Expert Group) [30]: This is a lossy compression method performed using discrete cosine transformation, where some data from the original picture is lost. In the JPEG compression method, compression is done by splitting the original image into minute pixel blocks, which are halved again and again to achieve the desired amount of compression. When a JPEG file is created or an image is converted from another format to JPEG, the user must specify the desired quality of the image. If the user chooses to create the highest quality image possible, the file size will be large.

TIFF (Tagged Image File Format) [31] [32]: The TIFF image file format is based on the lossless compression method, allowing for an image to be stored without losing even a percent of its original data. TIFF gives users the option to choose the LZW compression technique, officially supported by the GIF image format. This technique is characterized by allowing decrease of the image file's size with the data compression execution. It is a highly flexible and platform-independent format.

BMP (Bit Map Image) [33][34]: Bitmap is a home Windows raster format, which is used practically for all possible raster data storage. All BMP versions were designed for computers with Intel processors. The current format version is device independent (that means that Bitmap determines the pixel's color without reference to display device) and makes possible to record images of a different quality level (to the point of 32 bits).

MNG(Multiple-image Network Graphics): An extension of the .PNG image format that uses indexed colors and supports image animations; similar to an animated .GIF file.

PNG (Portable Network Graphics) [35]: It is a file format for image compression that, in time, is expected to replace the GIF that is widely

used on today's Internet. Owned by Unisys, the GIF format and its usage in image-handling software involve licensing or other legal considerations. (Web users can make, view, and send GIF files freely but they can't develop software that builds them without an arrangement with Unisys.) The PNG format, on the other hand, was developed by an Internet committee expressly to be patent-free. It provides a number of improvements over the GIF.

TGA (Truevision Advanced Raster Graphics Adapter)[36]: is a raster graphics file format. It is used to store and inter-change deep pixel images, paint, and image manipulation programs that need to store image data of up to 32 bits per pixel. The TGA format is most extensively used by the animation and video game industry. This is because their output is on the television screen and computer monitors, and not the printed page (TGA's color depths are not suited for the press). TGA is also popular for still-video editing, the reason being TGA's ability to be stored in a digital frame buffer.

PCX (Personal Computer eXchange)[37]: PCX is an image file format that uses a simple form of run-length encoding. It's lossless. Nowadays it has been largely replaced by GIF, JPEG and PNG which support better compression.

PSD [38]: PSD is a proprietary file format used to create and edit images in Adobe Photoshop. It is a widely accepted format as it supports all available image types – Bitmap, RGB, Duotone, Grayscale and Indexed Color. The biggest advantage of using the PSD format is that it can save images in layers (image, text, shape, etc.). In other words, the image may be opened with Adobe Photoshop anytime at a later date and each individual layer, or part, may be adjusted and edited independently to achieve better effects.

EXR [39]: Raster image stored in the OpenEXR format, a high dynamic-range (HDR) image file format developed by Industrial Light & Magic; supports multi-layer images, lossy and lossless compression, and 16-bit and 32-bit pixels; used for storing deep raster images for high-quality graphics; used by raster graphics editing programs and imaging applications.

3.5 Video Data Representations

A video data is a sequence of frames/images displayed to a computer or other screens. The frames are displayed at a speed such that it creates the illusion of motion. Frame rate describes the number of frames displayed per second and affects the smoothness of the perceived motion; the higher the frame rate the smoother the motion appears to be. Video codecs operate both on frame by frame basis and exploits redundancy that appears between sequential frames. Most video data also interleaves audio tracks to be played back with the frames. Some common video file formats and/or codecs are briefly discussed below:

3GP (or 3GPP - 3rd Generation Global Partnership Project)[27]: 3GP is in a multimedia file format. It was defined by 3GPP for use by high-speed wireless networks, or 3G cell phones. The 3GP file format stores video as MPEG-4 and H.263 files, and audio streams as AAC and AMR audio files.

3G2 [27]: 3G2 is a multimedia format developed and designed by the 3rd Generation Partnership Project 2 for transmission of multimedia files over the internet, and for use by mobile phone systems. 3GP specifications are based on Global System for Mobile Communications (GSM), whereas 3G2 specifications are based on Code Division Multiple Access (CDMA).

The 3G2 file, with the file extension '.3g2', is a multimedia file format, which means it stores both audio and video files. 3G2 can also store video streams in H.264 and MPEG-4 formats. Audio files are stored in AAC and MP3 formats.

AVI (Audio Video Interleave) [27]: is a multimedia container format. Developed by Microsoft, it is a format for audio and video data on a computer. AVI is actually a type of Resource Interchange File Format (RIFF), which divides a file's data into chunks, or blocks. Video clarity of AVI files mostly depends on the codecs used, and the type and level of compression.

MPEG [27]: (pronounced M-peg) stands for Moving Picture Experts Group, which is an acronym for the group that develops these standards. MPEG develops international standards for video and audio encoding, compression and decompression, processing, or their combination. MPEG also refers to the family of standards developed by the group. The family of standards produced by MPEG so far includes, but not limited to, MPEG-1, MPEG-2 and MPEG-4. MPEG is a non-commercial standard supported by most popular web browsers.

MOV [27]: also called QuickTime, is a multimedia container file format created by Apple Inc. A MOV file can contain video, graphics, animation, 3D and virtual reality (VR) content, or text (for example for subtitles). A QuickTime (.mov) file consists of several tracks, each containing data of one of the following – video, audio, effects, subtitles, and so on. These tracks are set in a hierarchal structure consisting of objects called atoms. These atoms either contain media data or edit data. This particular structure of separate atoms for containing and editing data makes QuickTime an ideal format for editing. MOV stores video and audio files in Apple's own proprietary compression technology.

SWF (ShockWave Flash) [40]: The file format SWF, a partially open repository for multimedia and especially for vector graphics, originated with FutureWave Software and has come under the control of Adobe. Intended to be small enough for publication on the web, SWF file extension can contain animations or applets of varying degrees of interactivity and function. SWF file extension is also sometimes used for

creating animated display graphics and menus for digital video disk (DVD) movies, and television commercials.

WMV (Windows Media Video)[27]: Windows Media Video is a compressed video file format developed by Microsoft when MPEG-4 did not have a standard version. The most important aspect of a WMV file is that its format allows large video files to be compressed while retaining their high quality. They may be of any size and may be compressed to match any bandwidth (connection speed). The codec allow streaming, playing, and downloading of content via the Internet. Another important feature of a WMV codec is that it is close sourced. In other words, it cannot be manipulated or edited.

DivX [27]: DivX, with file extension '.divx', is a digital media format developed by DivX, Inc. The DivX is a popular format due to its ability to compress lengthy video segments to a fraction of their original size while still retaining the original image quality. Based on MPEG-4, DivX uses a lossy compression scheme.

FLV (Flash Video File) [41]: is the name of a file format used to deliver video over the Internet using Adobe Flash Player (initially produced by Macromedia).

H.263 [27]: H.263 is a member of the H.26x family of video coding standards designed by International Telecom Union (ITU). It was developed in 1995 as an improvement over H.261, the previous ITU codec.

H.264 [27]: H.264 is a standard for video compression. Also known as MPEG-4 Advanced Video Coding) (AVC), it was jointly written by International Telecom Union - Video Coding Experts Group (ITU – VCEG) and ISO/IEC MPEG in 2003. The two groups maintain the codec together so as to have identical technical content. Hence, the codec is most often referred to as H.264/AVC.

NTSC (National Television System Committee) [42]: is the color encoding conversion standard that is used in most of the North and South Americas and Japan for television, video and DVD playback. It was created as a standard in the United States in 1941 for black and white television broadcasting.

PAL (Phase Alternating Line) [43]: is the color encoding conversion standard that is used in much of Europe, Africa, Asia, Australasia and the Middle East for television, video and DVD playback.

Sorenson [44]: A lossy QuickTime codec that offers a higher quality than the previously used Cinepak codec. The Sorenson compressor is frequently used to compress video from video cameras. A Sorenson compressor uses both temporal and spatial compression techniques. Sorenson compressor was jointly created by companies Sorenson Vision and Terran Interactive.

Xvid [27]: Xvid is a video codec that follows the MPEG-4 standard. Designed by a group of volunteer programmers specially to compete with the DivX standard, it is an open source and free video codec used by programmers all over the world. Xvid has been released under the GNU GPL license.

Chapter 4

Survey of Multimedia formats and algorithms

4.1 Lossy vs. Lossless Compression [12]

Data transmission and storage cost money. The more information being dealt with, the more it costs. In spite of this, most digital data are not stored in the most compact form. Rather, they are stored in whatever way makes them easiest to use, such as: America Standard Code for information Interchange (ASCII) text from word processors, binary code that can be executed on a computer, individual samples from a data acquisition system, etc. Typically, these easy-to-use encoding methods require data files about twice as large as actually needed to represent the information. Data compression is the general term for the various algorithms and programs developed to address this problem. A compression program is used to convert data from an easy-to-use format to one optimized for compactness. Likewise, an uncompression program returns the information to its original form.

4.2 Data Compression Strategies [12]

Table 4-1 shows two different ways that data compression algorithms can be categorized. In (a), the methods have been classified as either lossless or lossy. A lossless technique means that the restored data file is identical to the original. This is absolutely necessary for many types of data, for example: executable code, word processing files, tabulated numbers, etc. It is a disaster to misplace even a single bit of this type of information. In comparison, data files that represent images and other acquired signals do not have to be kept in perfect condition for storage or transmission. All real world measurements inherently contain a certain amount of noise. If the changes made to these signals resemble a small amount of additional noise, no harm is done. Compression techniques that allow this type of degradation are called lossy. Lossy techniques are much more effective at

compression than lossless methods. The higher the compression ratio, the more noise added to the data.

Table 4-1: Compression Classifications [12]

Lossless	Lossy
Run Length	CS&Q
Huffman	JPEG
Delta	MPEG
LZW	

(a)

Method	Group Size: *Input*	*Output*
CS&Q	Fixed	fixed
Huffman	fixed	variable
Arithmetic	fixed	variable
Run-length, LZW	variable	variable
	variable	fixed

(b)

Images transmitted over the World Wide Web are an excellent example of why data compression is important. For example, if we need to download a digitized color photograph over a computer's 33.6 kbps modem and the image is not compressed; it will contain about 600 KB of data. If it has been compressed using a lossless technique (such as used in the GIF format), it will be about one-half this size, or 300 KB. If lossy compression has been used (a JPEG file), it will be about 50 KB. Hence, the download times for these three equivalent files are 142 seconds, 71 seconds, and 12 seconds, respectively.

The second way of classifying data compression methods is shown in Table 4-1b. Most data compression programs operate by taking a group of data from the original file, compressing it in some way, and then writing the compressed group to the output file. For instance, one of the techniques in this table is short for coarser sampling and/or quantization (CS&Q). Suppose we are compressing a digitized waveform, such as an audio signal that has been digitized to 12 bits. We might read two adjacent samples from the original file (24 bits), discard one of the samples completely, discard the least significant 4 bits from the other sample, and then write the remaining 8 bits to the output file. With 24 bits in and 8 bits out, we have implemented a 3:1 compression ratio using a lossy algorithm.

4.2.1 Run-Length Encoding (RLE) [12]

Data files frequently contain the same character repeated many times in a row. For example, text files use multiple spaces to separate sentences, indent paragraphs, format tables & charts, etc. Digitized signals can also have runs of the same value, indicating that the signal is not changing. For instance, an image of the nighttime sky would contain long runs of the character or characters representing the black background. Likewise, digitized music might have a long run of zeros between songs. RLE encoding is a simple method of compressing these types of files.

Figure 4-1 illustrates run-length encoding for a data sequence having frequent runs of zeros. Each time a zero is encountered in the input data, two values are written to the output file. The first of these values is a zero, a flag to indicate that run-length compression is beginning. The second value is the number of zeros in the run. If the average run-length is longer than two, compression will take place. On the other hand, many single zeros in the data can make the encoded file larger than the original.

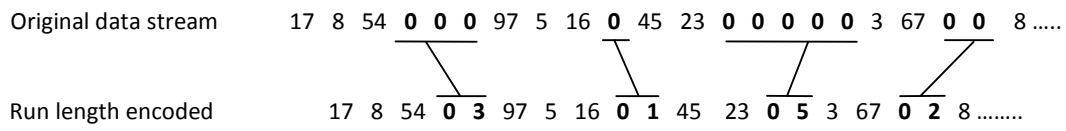


Figure 4-1: Example of run-length encoding [12].

4.2.2 Huffman Encoding [12]

This method is named after D.A. Huffman, who developed the procedure in the 1950s. Figure 4-2 shows a histogram of the byte values from a large ASCII file. More than 96% of this file consists of only 31 characters: the lower case letters, the space, the comma, the period, and the carriage return. This observation can be used to make an appropriate compression scheme for this file. To start, let's assign each of these 31 common characters a five bit binary code: 00000 = "a", 00001 = "b", 00010 = "c", etc. This allows 96% of the file to be reduced in size by 5/8. The last of the

five bit codes, 11111, will be a flag indicating that the character being transmitted is not one of the 31 common characters. The next eight bits in the file indicate what the character is, according to the standard ASCII assignment. These results in 4% of the characters in the input file requiring 5+8=13 bits. The idea is to assign frequently used characters fewer bits, and seldom used characters more bits. In this example, the average number of bits required per original character is: 0.96×5 + 0.04×13 = 5.32. In other words, an overall compression ratio of: 8 bits/5.32 bits, or about 1.5:1.

Huffman encoding takes this idea to the extreme. Characters that occur most often, such the space and period, may be assigned as few as one or two bits. Infrequently used characters, such as: !, @, #, $ and %, may require a dozen or more bits. In mathematical terms, the optimal situation is reached when the number of bits used for each character is proportional to the logarithm of the character's probability of occurrence.

A clever feature of Huffman encoding is how the variable length codes can be packed together. Imagine receiving a serial data stream of ones and zeros. If each character is represented by eight bits, you can directly separate one character from the next by breaking off 8 bit chunks. Now consider a Huffman encoded data stream, where each character can have a variable number of bits. The separation of one character from the next involves proper selection of the Huffman codes that enable the correct separation.

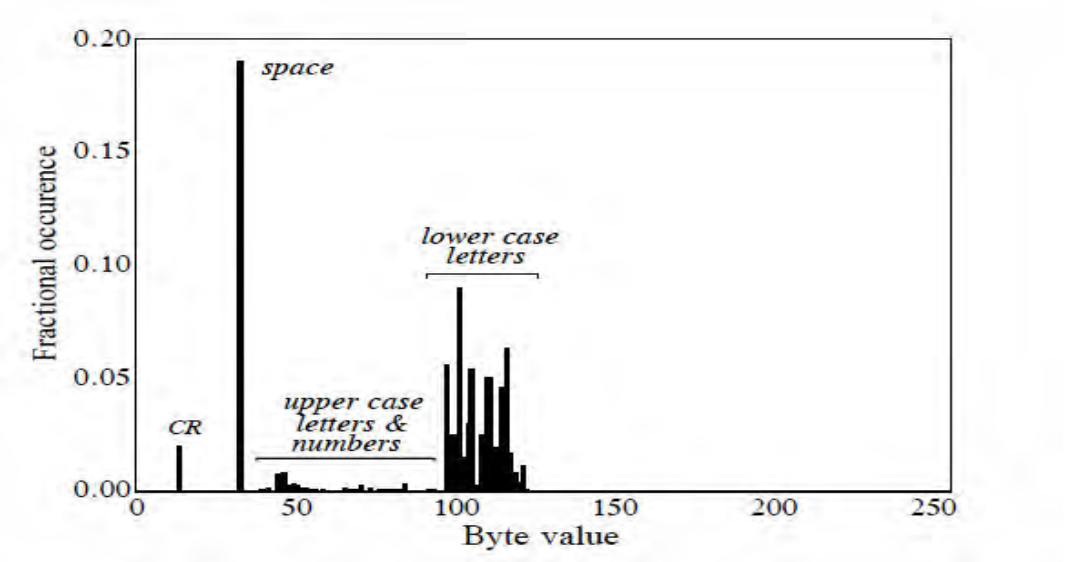


Figure 4-2: Histogram of text [12]

Figure 4-3 shows a simplified Huffman encoding scheme. The characters A through G occur in the original data stream with the probabilities shown. Since the character A is the most common, we will represent it with a single bit, the code: 1. The next most common character, B, receives two bits, the code: 01. This continues to the least frequent character, G, being assigned six bits, 000011. As shown in this illustration, the variable length codes are resorted into eight bit groups, the standard for computer use.

When uncompression occurs, all the eight bit groups are placed end-to-end to form a long serial string of ones and zeros. In Figure 4-3, each code consists of two parts: a number of zeros before a one, and an optional binary code after the one. This allows the binary data stream to be separated into codes without the need for delimiters or other marker between the codes.

Letter	Probability	Huffman code
A	.154	1
B	.110	01
C	.072	0010
D	.063	0011
E	.059	0001
F	.015	000010
G	.011	000011

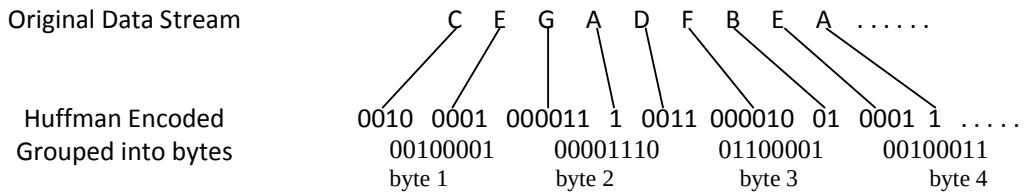


Figure 4-3: Huffman Encoding [12].

The uncompression program looks at the stream of ones and zeros until a valid code is formed, and then starting over looking for the next character. The way that the codes are formed insures that no ambiguity exists in the separation.

A more sophisticated version of the Huffman approach is called arithmetic encoding. In this scheme, sequences of characters are represented by individual codes, according to their probability of occurrence. This has the advantage of better data compression, say 5-10%. Run-length encoding followed by either Huffman or arithmetic encoding can also be used.

4.2.3 Delta Encoding [12]

In mathematical computation, the Greek letter delta (Δ) is used to denote the change in a variable. The term delta encoding, refers to several techniques that store data as the difference between successive samples (or characters), rather than directly storing the samples themselves. Figure 4-4 shows an example of how this is done. The first value in the delta encoded file is the same as the first value in the original data. All the following values in the encoded file are equal to the difference (delta) between the corresponding value in the input file, and the previous value in the input file.

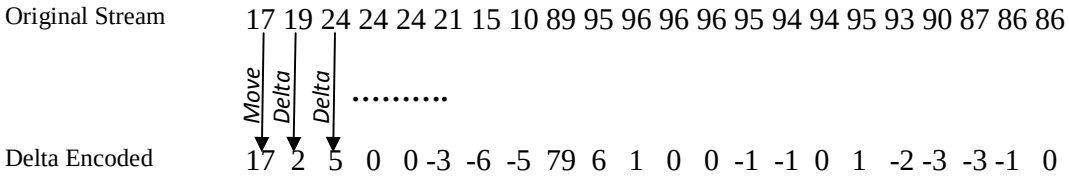


Figure 4-4: An example of delta encoding [12].

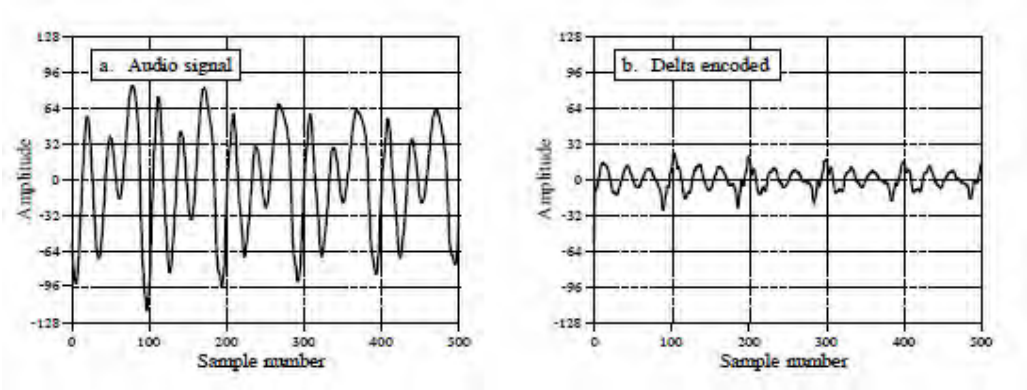


Figure 4-5: Example of delta encoding [12].

Delta encoding can be used for data compression when the values in the original data are smooth, that is, there is typically only a small change between adjacent values. This is not the case for ASCII text and executable code; however, it is very common when the file represents a signal. For instance, Fig. 4-5a shows a segment of an audio signal, digitized to 8 bits, with each sample between -127 and 127. Figure 4-5b shows the delta encoded version of this signal. The key feature is that the delta encoded signal has lower amplitude than the original signal. In other words, delta encoding has increased the probability that each sample's value will be near zero, and decreased the probability that it will be far from zero. This uneven probability is just the thing that Huffman encoding needs to operate. If the original signal is not changing, or is changing in a straight line, delta encoding will result in runs of samples having the same value. This is what run-length encoding requires. Correspondingly, delta

encoding followed by Huffman and/or run-length encoding is a common strategy for compressing signals.

The idea used in delta encoding can be expanded into a more complicated technique called Linear Predictive Coding (LPC). The idea behind LPC is that, imagine the first 99 samples from the input signal have been encoded, and we are about to work on sample number 100. Then we need to predict the most likely value for sample 100 based on the first 99 samples. In delta encoding, the most likely value for sample 100 is the same as the previous value, sample 99. This expected value is used as a reference to encode sample 100. That is, the difference between the sample and the expectation is placed in the encoded file. LPC expands on this by making a better guess at what the most probable value is. This is done by looking at the last several samples, rather than just the last sample.

4.2.4 LZW Compression [12]

LZW compression is named after its developers, A. Lempel and J. Ziv, with later modifications by Terry A. Welch. It is a technique for general purpose data compression due to its simplicity and versatility. LZW also performs well when presented with extremely redundant data files, such as tabulated numbers, computer source code, and acquired signals.

LZW compression uses a code table, as illustrated in Fig. 4-6. A common choice is to provide 4096 entries in the table. In this case, the LZW encoded data consists entirely of 12 bit codes, each referring to one of the entries in the code table. Uncompression is achieved by taking each code from the compressed file, and translating it through the code table to find what character or characters it represents. Codes 0-255 in the code table are always assigned to represent single bytes from the input file. For example, if only these first 256 codes were used, each byte in the original file would be converted into 12 bits in the LZW encoded file, resulting in a 50% larger file size. During uncompression, each 12 bit code would be

translated via the code table back into the single bytes. Of course, this wouldn't be a useful situation.

The LZW method achieves compression by using codes 256 through 4095 to represent sequences of bytes. For example, code 523 may represent the sequence of three bytes: 231 124 234. Each time the compression algorithm encounters this sequence in the input file, code 523 is placed in the encoded file. During uncompression, code 523 is translated via the code table to recreate the true 3 byte sequence. The longer the sequence assigned to a single code, and the more often the sequence is repeated, the higher the compression achieved.

Although this is a simple approach, there are two major obstacles that need to be overcome: (1) how to determine what sequences should be in the code table, and (2) how to provide the uncompression program the same code table used by the compression program. The LZW algorithm exquisitely solves both these problems.

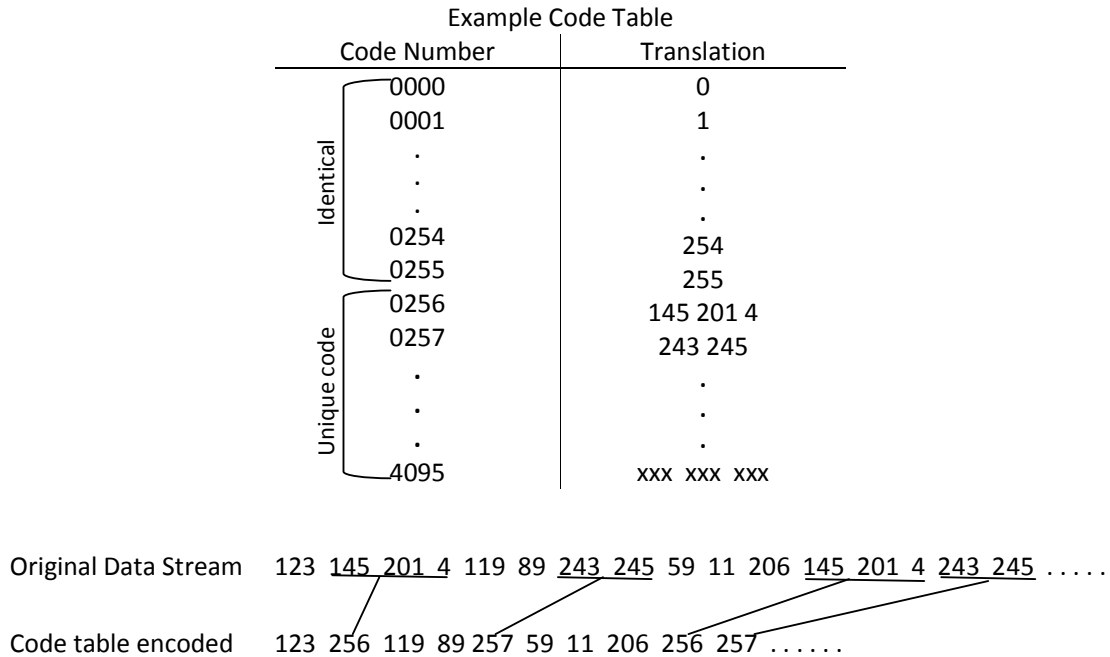


Figure 4-6: Example of code table compression [12].

When the LZW program starts to encode a file, the code table contains only the first 256 entries, with the remainder of the table being blank. This means that the first codes going into the compressed file are simply the single bytes from the input file being converted to 12 bits. As the encoding continues, the LZW algorithm identifies repeated sequences in the data, and adds them to the code table. Compression starts the second time a sequence is encountered. The key point is that a sequence from the input file is not added to the code table until it has already been placed in the compressed file as individual characters (codes 0 to 255). This is important because it allows the uncompression program to reconstruct the code table directly from the compressed data, without having to transmit the code table separately.

A flowchart of the LZW uncompression algorithm is shown in Fig. 4-8. Each code is read from the compressed file and compared to the code table to provide the translation. As each code is processed in this manner, the code table is updated so that it continually matches the one used during the compression. However, there is a small complication in the uncompression routine. There are certain combinations of data that result in the uncompression algorithm receiving a code that does not yet exist in its code table. This contingency is handled in boxes 4, 5 & 6.

The execution time of the compression algorithm is limited by searching the code table to determine if a match is present. As an analogy, imagine you want to find if a friend's name is listed in the telephone directory. The catch is, the only directory you have is arranged by telephone number, not alphabetical order. This requires you to search page after page trying to find the name you want. This inefficient situation is exactly the same as searching all 4096 codes for a match to a specific character string. The answer: organize the code table so that what you are looking for tells you where to look (like a partially alphabetized telephone directory). In other words, don't assign the 4096 codes to sequential locations in memory.

Rather, divide the memory into sections based on what sequences will be stored there. For example, suppose we want to find if the sequence: code 329 + x, is in the code table. The code table should be organized so that the "x" indicates where to starting looking. There are many schemes for this type of code table management, and they can become quite complicated.

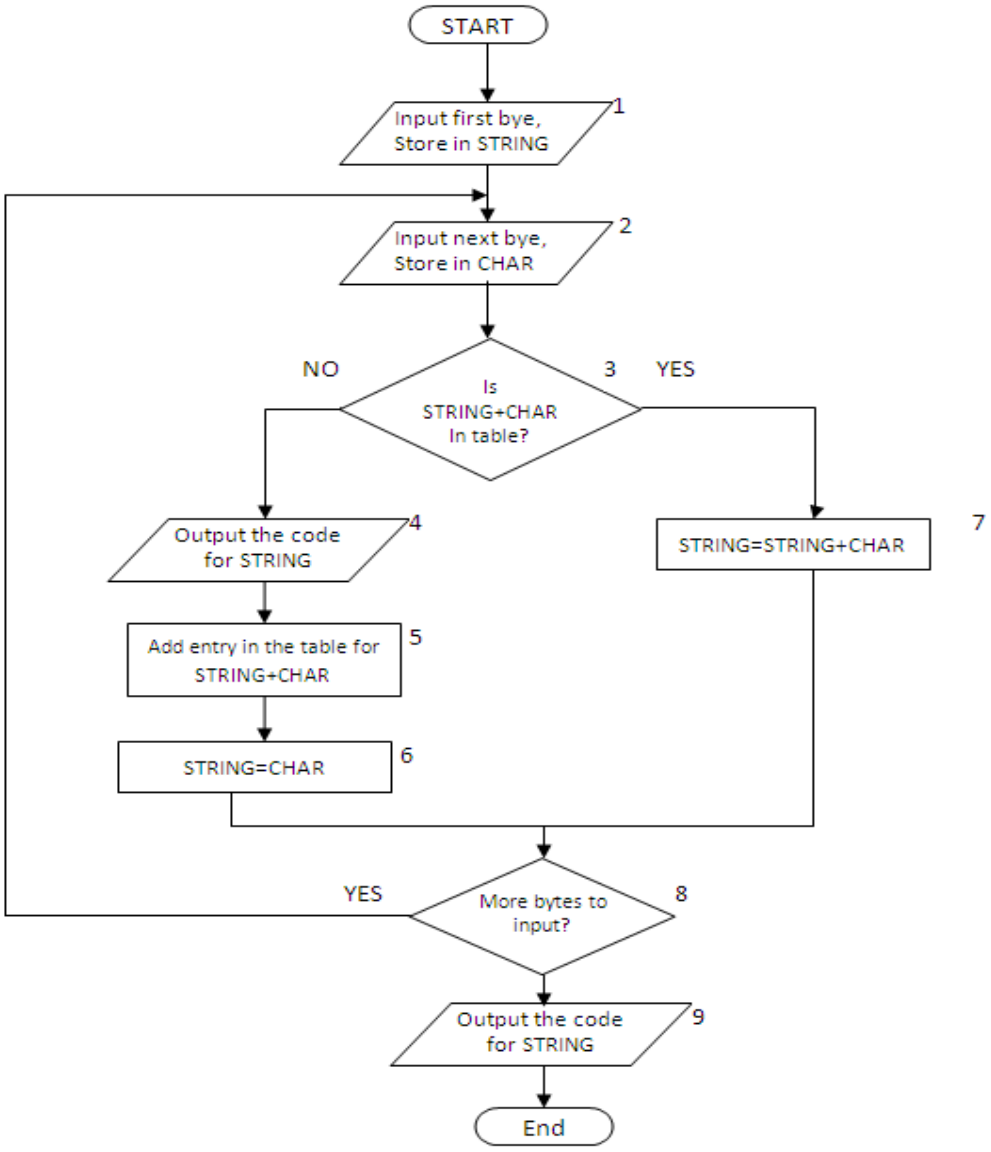


Figure 4-7: LZW compression flow chart [12].

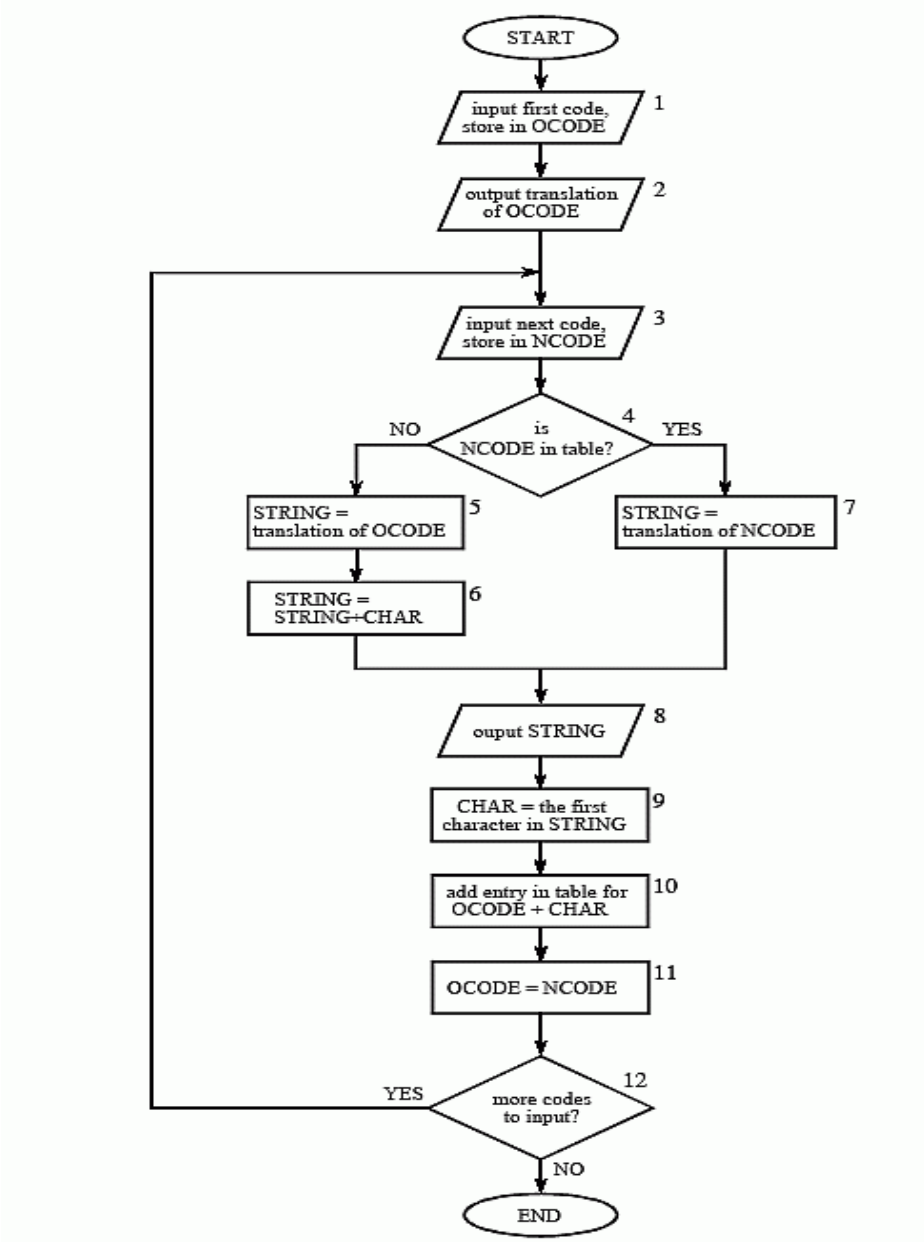


Figure 4-8: LZW uncompression flowchart [12]

4.2.5 JPEG (Joint Photographic Expert Group) [13]

JPEG works on either full-color or gray-scale images; it does not handle bi-level (black and white) images well. It doesn't handle color mapped images either; the image has to be pre-expanded into an unmapped full-

color representation. JPEG works best on "continuous tone" images. Images with many sudden jumps in color values will not compress well.

There are a lot of parameters to the JPEG compression process. By adjusting the parameters, we can trade off compressed image size against reconstructed image quality over a wide range. You can get image quality ranging from op-art (at 100x smaller than the original 24-bit image) to quite indistinguishable from the source (at about 3x smaller). Usually the threshold of visible difference from the source image is somewhere around 10x to 20x smaller than the original, i.e. 1 to 2 bits per pixel for color images. Grayscale images do not compress as much. In fact, for comparable visual quality, a grayscale image needs perhaps 25% less space than a color image; certainly not the 66% less that is naively expected.

JPEG defines a "baseline" lossy algorithm, plus optional extensions for progressive and hierarchical coding. There is also a separate lossless compression mode; this typically gives about 2:1 compression, i.e, about 12 bits per color pixel. Most currently available JPEG hardware and software handles only the baseline mode.

Here's the outline of the baseline compression algorithm:

1. Transform the image into a suitable color space. This is not applicable for grayscale, but for color images you generally want to transform RGB into a luminance/chrominance color space (YCbCr, YUV, etc). The luminance component is grayscale and the other two axes are color information. The reason for doing this is that you can afford to lose a lot more information in the chrominance components than you can in the luminance component: the human eye is not as sensitive to high-frequency chroma info as it is to high-frequency luminance

2. (Optional) Downsample each component by averaging together groups of pixels. The luminance component is left at full resolution, while the

chroma components are often reduced 2:1 horizontally and either 2:1 or 1:1 (no change) vertically. In JPEG-speak these alternatives are usually called 2h2v and 2h1v sampling.

This step immediately reduces the data volume by one-half or one-third. In numerical terms it is highly lossy, but for most images it has almost no impact on perceived quality, because of the eye's poorer resolution for chroma info. Downsampling is not applicable to grayscale data; this is one reason color images are more compressible than grayscale.

3. Group the pixel values for each component into 8x8 blocks. Transform each 8x8 block through a Discrete Cosine Transform (DCT). The DCT is a relative of the Fourier transform and likewise gives a frequency map, with 8x8 components. Thus we now have numbers representing the average value in each block and successively higher-frequency changes within the block. The motivation for doing this is that you can now throw away high-frequency information without affecting low-frequency information. (The DCT transform itself is reversible except for round off error.)

4. In each block, divide each of the 64 frequency components by a separate "quantization coefficient", and round the results to integers. This is the fundamental information-losing step. The larger the quantization coefficients, the more data are discarded. Note that even the minimum possible quantization coefficient, 1, loses some info, because the exact DCT outputs are typically not integers. Higher frequencies are always quantized less accurately (given larger coefficients) than lower, since they are less visible to the eye. Also, the luminance data is typically quantized more accurately than the chroma data, by using separate 64-element quantization tables.

5. Encode the reduced coefficients using either Huffman or arithmetic coding. (Strictly speaking, baseline JPEG only allows Huffman coding;

arithmetic coding is an optional extension.) Notice that this step is lossless, so it doesn't affect image quality.

6. Tack on appropriate headers, etc, and output the result. In a normal "interchange" JPEG file, all of the compression parameters are included in the headers so that the decompressor can reverse the process. These parameters include the quantization tables and the Huffman coding tables. For specialized applications, the spec permits those tables to be omitted from the file; this saves several hundred bytes of overhead, but it means that the decompressor must know a-priori what tables the compressor used. Omitting the tables is safe only in closed systems.

The decompression algorithm reverses this process. The decompressor multiplies the reduced coefficients by the quantization table entries to produce approximate DCT coefficients. Since these are only approximate, the reconstructed pixel values are also approximate, but if the design has done what it's supposed to do, the errors won't be highly visible. A high-quality decompressor will typically add some smoothing steps to reduce pixel-to-pixel discontinuities.

The JPEG standard does not specify the exact behavior of compressors and decompressors, so there's some room for creative implementation. In particular, implementations can trade off speed against image quality by choosing more accurate or faster-but-less-accurate approximations to the DCT. Similar tradeoffs exist for the downsampling/upsampling and color space conversion steps. (The spec does include some minimum accuracy requirements for the DCT step, but these are widely ignored, and are not too meaningful anyway in the absence of accuracy requirements for the other lossy steps.)

4.2.6 MPEG (Moving pictures Expert Group) [12]

MPEG is a compression standard for digital video sequences, such as used in computer video and digital television networks. In addition, MPEG also

provides for the compression of the sound track associated with the video. In addition to reducing the data rate, MPEG has several important features. The movie can be played forward or in reverse, and at either normal or fast speed. The encoded information is random access, that is, any individual frame in the sequence can be easily displayed as a still picture. This goes along with making the movie editable, meaning that short segments from the movie can be encoded only with reference to themselves, not the entire sequence.

The approach used by MPEG can be divided into two types of compression: within-the-frame and between-frame. Within-the-frame compression means that individual frames making up the video sequence are encoded as if they were ordinary still images. This compression is preformed using the JPEG standard, with just a few variations. In MPEG terminology, a frame that has been encoded in this way is called an intra-coded or I-picture.

Most of the pixels in a video sequence change very little from one frame to the next. Unless the camera is moving, most of the image is composed of a background that remains constant over dozens of frames. MPEG takes advantage of this with a sophisticated form of delta encoding to compress the redundant information between frames. After compressing one of the frames as an I-picture, MPEG encodes successive frames as predictive-coded or P-pictures. That is, only the pixels that have changed since the I-picture are included in the P-picture.

The main distortion associated with MPEG occurs when large sections of the image change quickly. In effect, a burst of information is needed to keep up with the rapidly changing scenes. If the data rate is fixed, the viewer notices "blocky" patterns when changing from one scene to the next. This can be minimized in networks that transmit multiple video channels simultaneously, such as cable television. The sudden burst of information needed to support a rapidly changing scene in one video

channel, is averaged with the modest requirements of the relatively static scenes in the other channels.

4.3 Compatibility of different file formats

High quality video files are huge. The higher the resolution, the bigger the file because it contains more information in order to achieve that high resolution. To play videos on the web or the computer, they have to be compressed, or made smaller. This is done by designing ways to take information out of the video signal. If we take information out, the resulting file is smaller but the loss of information degrades the picture quality. This process is called compression.

Video files can be compressed in a wide variety of ways by different kinds of software/codecs. This results in many different formats. Which format it is is indicated by the three letters (sometimes 4) that follow the file name.

Lots of different companies and organizations have designed programs to compress video. They each have their own way of doing it, so you have lots of different file formats. Some were created by Microsoft, some by Apple, some by standardizing agencies like the MPEG.

Once you compress a video, you have to have a program that reads it, or basically uncompresses it to play. The program that reads the video is called a video player. Video players are built to read particular video file formats and not others. This is where the incompatibility comes in.

In an audio or video editing program, the format affects how each command will work. When they design a program, they build it to work with certain formats only. So everybody is competing and the nature of the programs causes incompatibility.

The following terms are used to describe the various components in multimedia data processing:

File Format: a specific way to encode data that is to be saved as a file. The encoding is left up to the codecs.

Codec: a program/algorithm that encodes/decodes data to convert a file between different formats. The popular media codecs are generally for shrinking file size.

Lossy Codec: refers to a codec that sacrifices file quality for the sake of compression.

Lossless Codec: does not destroy any data, regardless of whether or not the data is necessary for the file's integrity.

Metadata: information about the file that is stored within the file itself – for example, when a picture was taken and what type of camera it was taken with, or the artist of an audio track.

Container: a file format that concerns itself more with how data is stored, and not necessarily coded.

Bitrate: the number of bits processed per second. To put things into perspective, mp3's generally have a bitrate of 128 kbit/s, while CD's generally have bitrates of around 1.4 Mbit/s.

4.4 Security of the data

Security of multimedia data implies, most importantly, the confidentiality and authenticity of the source. Audio stream of data may contain confidential conversation between communicating parties that needs to be encrypted. The revelation of the contents of this kind of data to the wrong person may cause severe damage to social or political system. In addition, data sources should be authentic, so that the sender of the data is known for sure and cannot be denied being as the source.

Authenticity implies that the source of the data is definitively identified and cannot be denied. This kind is commonly implemented with Public

Key Infrastructure (PKI). Keeping the confidentiality of data involves hiding the true content of the data except for the intended recipient. Confidentiality is realized by use of passwords in most cases. The integrity of data strives to flag to the data recipient if the content of the data has been altered in any way. Integrity of data is maintained by use of hashing functions of various complexities.

Most multimedia specific codec are not designed with security in mind as that would cause more processing inefficiencies. Some lossless archiving tools coded with general data compression goal provide an option for securing the data with encryption or hashing.

Two options are available for integrating security with compression algorithms [4]. The first option is to embed the logic of encryption with the compression algorithm. This makes the codec to be specific to applications that necessarily require security. The second option is to apply the compression and security logic separately one after another. Unlike the first option, this method gives the user the opportunity to choose compression only or compression followed by encryption. The option of encryption followed by compression is only applicable to lossless methods.

Chapter 5

Performance Evaluation and Result Analysis

5.1 Introduction

Selecting the 'best' algorithm and program for processing multimedia data can be difficult to generalize due to several compromises that can be made. Compromises can be made on one or more of the evaluation parameters such as the processing time, quality or size of the output data. Each of these parameters can have a range of values (both measurable and non-measureable) that allows several combinations to attain a satisfactory result. The choice of an algorithm and their associated configuration settings (quality, compression level, methodologies used etc) depend on the final purpose of the output data. Some applications may use the output data for entertainment purposes while others can use it for long time archiving or historical significance. As mentioned in the literature review chapter (Chapter 2), several variables are defined to compare the various implementations. Even using the same baseline logic and parameter settings, codec algorithms can vary in the measurable and observable characteristics depending on the programming experience, hardware platform and implementation tools (programming languages).

This thesis work evaluates the performance parameters on multimedia codecs using two independent approaches, Approach I and Approach II.

Approach I: In this approach, the algorithms are taken from literatures and some operational principles of standardized methods. Due to their defining principle of preserving the entire information content and are solely based on exploiting the repetitions available in data streams, innovation on lossless compression algorithms are less probable to provide better solution. Basic repetition removal methods have been implemented as part of all lossy methods of compression and hence, this approach considers only lossy methods. Most lossy algorithms (see chapter 4) involve complex mathematical operations that practically require

hardware module to handle those operations, or otherwise perform slow. But the ones implemented here use only integer operations and do not assume a hardware module to support them.

Three main algorithms are implemented in this approach, named here as Algorithm 1 (ALG1), Algorithm 2 (ALG2) [1] and Algorithm 3 (ALG3). ALG3 and its several variations are designed and implemented in this thesis work based on the logics used in JPEG and the first two algorithms. The algorithms are selected based on their simplicity and unavailability of the code implementations. Detailed designs, implementations and tests for the mentioned algorithms are given in Section 5.3.

Approach II: In this approach, off-the-shelf products have been used to encode the uncompressed formats of the test files. These products are well standardized commercial or open products and are treated separately in order to make a uniform comparison. Black-Box testing has been used without considering the details of how the algorithms work. The experiment in this approach is done on four types of multimedia data: music audio, speech audio, image and video data. The details of the tools used and procedures followed are discussed in section 5.4.

Design of implementation: The implementation in this work or any other codec design has a specific set of procedures to follow. First the raw multimedia data is prepared with relevant information registered. The data is then given as an input to a codec program to generate the required output file. This file can then be stored locally or transmitted over a wired or wireless network to a destination where the reverse process of decoding happens by a compatible codec to regenerate the original or near original data.

Alternative approaches used and implementation difficulties

Initially the idea of the work is to implement the codecs in an appropriate programming language and test the common ones across similar

programming environment. The programming was started with .NET. But due to .NET's luck of suitable class libraries for multimedia programming, a separate application called Aurigma for .NET has been used. This programming plug-in is available as a free trial to work with .Net. Aurigma libraries have been installed and imported into .Net and used to read several file formats and process them. For example it has libraries to read images and make operations on them. The problem with this implementation is that the program runs very slowly as part of the .Net and the add-on limitations. The program took several minutes (typically over 20 minutes) for twelve colored images of 320x320 pixel dimensions. The next alternative was to use Java Development Kit (JDK) and Java Multimedia Framework (JMF) and similar problem was encountered during implementation. In addition, java doesn't have unsigned data types which are necessary for image processing applications without longer bit variables. A better solution was to do the coding in Matlab and call the appropriate functions from Java or .Net. The limitation with this procedure was that, Matlab supports few file formats and library codes for most codecs are not available. The other problem was that, it has been realized that the complexity and diversity of the codecs hinder the custom implementation of each one.

As part of the final solution, an alternative approach has been used such that the algorithms for which no program has been found and easy to code are implemented with Matlab, but for well standardized and commonly used codecs, off-the-shelf software programs have been used. The software packages used are selected based on the number of file formats, codecs and platforms (if available) they support.

5.2 Evaluation parameters

The performance evaluation task begins by defining relevant metrics that sufficiently describe the time taken to encode and decode, the amount of

storage required, and the time to transmit over a network medium. The following are the parameters commonly used: [1] [11] [16] [17].

Compression ratio: This measures the saving in storage of the data and is computed as

$$\frac{\textit{Size in units of memory after compresson}}{\textit{Size in units of memory before compression}} \qquad (5\text{-}1)$$

Storage requirement: It is the exact memory size of the data after processing with a codec. It consists of the actual data output after compression and associated information that can be used by some codecs during decompression such as encoding table, quantization values etc.

Bandwidth requirement/ Transmission time: This is the time it takes to transmit the data over a network from one physical computing system to the other. Generally, network protocols are not concerned with the type of data and this parameter is directly proportional to the compressed size that is to be transmitted.

Encoding execution time: it is the amount of time required to process the input uncompressed data and generate the compressed format.

Decoding execution time: the time taken to uncompress the compressed data. Since decompression mostly reverses the encoding operation, the time of decoding is equal/proportional to the encoding time.

Quality (subjective): This is an observable parameter with the sense of hearing or vision to compare the output of the decoding process with the original input data.

Reconstruction Error: This parameter is defined here with respect to multimedia data to mathematically represent how well the codec preserves initial information. It is a computed parameter from the output of the decoding process with that of the test data input to the encoder. It is the

measure of the amount of information lost during the lossy processing and calculated as the root mean square (rms) error of the compared data points.

$$rms = \sqrt{\left(\sum_{p=1}^{3} \sum_{i=1,j=1}^{n,m} \left(x_{i,j,p} - y_{i,j,p}\right)^2\right) \Big/ (3 * m * n)} \qquad (5\text{-}2)$$

Where, x and y are the pixel values of the original and decompressed images, respectively

n and m are the height and width of the image, respectively

p is the color plane index

Some of the above listed parameters are closely related with one another. The bandwidth requirement of transferring a given file across a network is solely dependent on the size of the data to be transmitted. Hence, it is valid to assume that the better the compression ratio (the lower the computed value in equation 5-1), the lesser the bandwidth consumption.

Test Environment

For audio and image tests:

- Toshiba Satellite Series P205-S6247 Laptop Computer
- Intel® CPU, Dual core processor, (1.73GHZ,1.73GHZ)
- Intel® 8280, 1GB Motherboard
- 1GB RAM
- Mobile Intel® 945 Express Chipset Family Display Adapter
- Hitachi HTS543216L9A300 Hard Drive
- Pioneer DVD-RW DVR-K17LF
- Realtek High Definition Audio Driver
- Microsoft Windows XP Professional, Service Pack 3 Operating System.

Video tests:

The video test has been done on a separate computer platform with the following specifications:

- Toshiba Satellite Series A505-S6005 Laptop Computer
- Intel® core i3-330M processor (2.13 GHZ/2.13 GHZ)
- 4GB RAM
- NVIDIA GeForce 310M Display Adapter
- Fujitsu MJA2500BH G2 500GB Hard Disk
- TSSTcorp CDDVDW TS-L633Y CD/DVD writer
- Microsoft Windows 7 Home Premium (64-bits) Operating System.

The computer used in the video test is better in performance than the test computer used in the other tests. The reason for using this computer was to make the testing faster as the video data measures large values for the evaluation parameters.

5.3. Approach I

Two general categories are used in this approach. The first one is insignificant bits truncation (IBT) method as proposed in [11]. The least significant bits (LSB) of a byte represent a relatively smaller information share of the whole value so that these bits can be discarded and the remaining ones can be combined in a certain pattern to get a lesser storage size than the original data. At the decoding side, the received data bytes can be disassembled and the lost bits are replaced with zeros. Depending on the number of bits discarded from a byte, two algorithms were proposed, namely, ALG1 and ALG2 (see section 5.3.1). In ALG1, the lower four bits of a byte are discarded where as in ALG2, the logic discards two bits. The second approach, ALG3, is proposed in this thesis work with several sub-options; so that viable features from the first category and some working principle of JPEG are used to compress image files (see section 5.3.2).

Matlab has been used in the coding for the fact that it is well optimized and widely used programming environment for matrix data. As part of the procedure, the images were read and represented in a matrix form. The operations have been performed on each plane of the colored image and the results separately stored in three matrix variables, representing the RGB color components. The decompression is preformed separately on each of the component color planes and the final outputs combined to form the RGB color image. The inherent property of the algorithm is that, the process is lossy and the performance evaluation parameters are independent of the image contents or color compositions.

5.3.1 Insignificant 'bits' truncation

The logic behind this method is that, since the lower order bits of the encoded signal stores lesser information in the original signal than the higher order bites of the data bytes, we can conserve storage size, and hence transimission bandwidth requirment over a limited network infrastructure. In [11], the authors have shown the results of insignificant bit truncation (See ALG1 and ALG2 in the next subsection) using images taken by both low resolution and high resolution web cams. The authors have claimed that some results are unexpected and they did not provide an explanation. In this current experiment, the most popular image test file, lena.bmp (468KB size), has been used. The result shows qualitatively better images after decompressing the compressed images than mentioned in the literature.

5.3.1.1 Combination of two bytes into one byte (ALG1)

The schematics of the whole process is shown in Fig 5-1. ALG1 works in such a way that the four lower order bits (a3-a0 and b3-b0) of a byte are discarded so that the four most significant bits of two bytes can be represented by one byte (c7-c0). During decoding the reassembled bytes will have zeros in the four least siginificant bit positions. For the purpose of comparing the reconstruction error with standard compression

algorithm, an image converter tool, Pixillion Image Converter, has been used to generate the JPEG file and the difference has been computed.

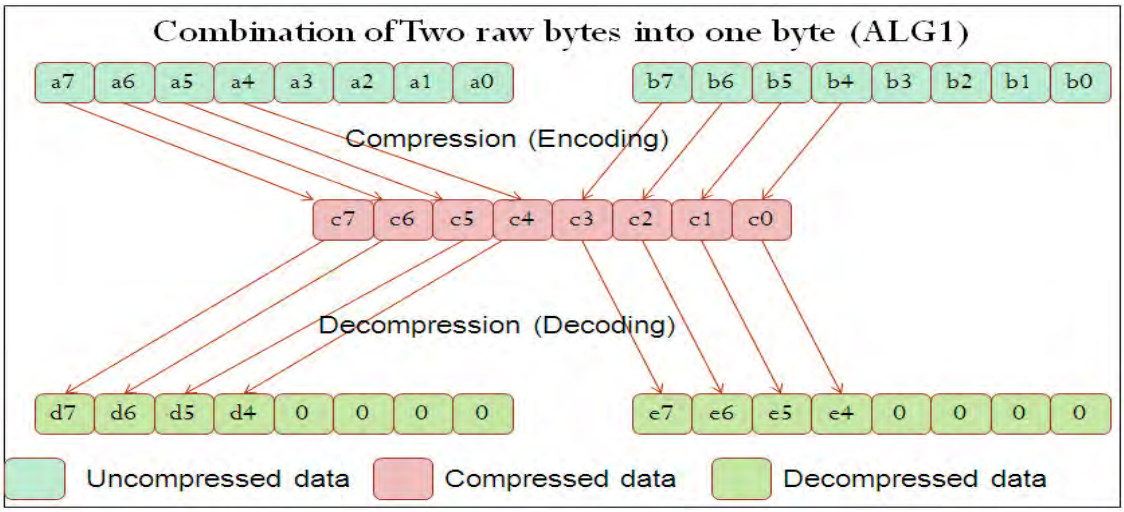


Figure 5-1: ALG1 Schematics

The graph in Fig. 5-3 shows the data points before and after compression (of the images shown in Fig. 5-2) midway along the row.

Figure 5-2 Result of ALG1

TOP Blue: Original Green: Recovered Red: JPEG

BOTTOM Red: Error with ALG1 Cyan: Error with JPEG

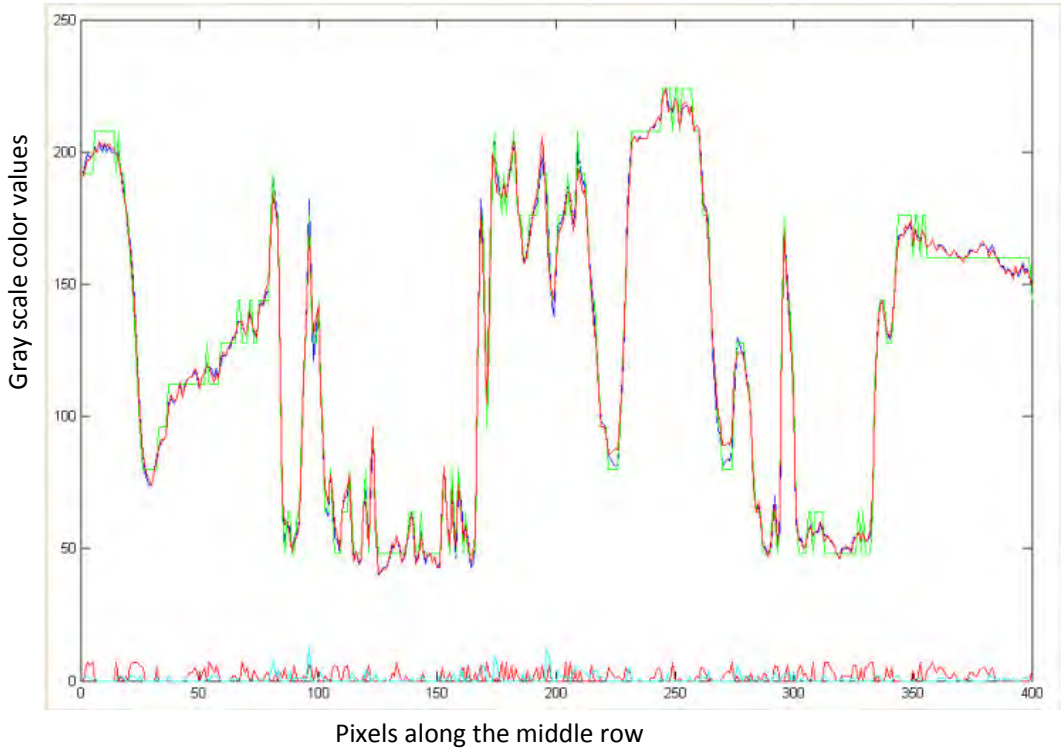


Figure 5-3: Graph showing data points from middle row of the test image for ALG1.

The bottom red curve shows the absolute value of the errors between original file and recovered file and the cyan shows the absolute error with respect to the JPEG version of the file. From the graph we can see that, relative to the magnitude of the color values, the error can be of an acceptable magnitude to attain 50% compression. The calculated value for the construction error in this case is 3.32 (Equation 5-2). The same image converted to JPEG resulted in a reconstruction error of 1.85.

5.3.1.2 Combination of four bytes into three bytes (ALG2)

In this implementation, the same uncompressed image as above, lena.bmp has been used to test the algorithm. The rest of the procedures are similar except that only two bits of a data byte are discarded to be replaced at the destination with zeros. The schematic of ALG2 is shown in Fig. 5-4.

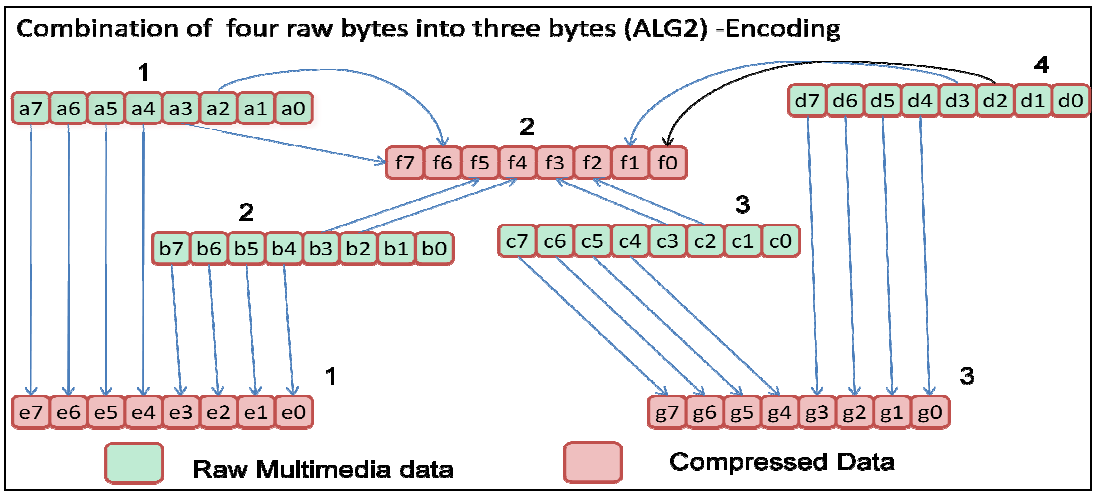


Figure 5-4(a): ALG2 Encoding Schematics

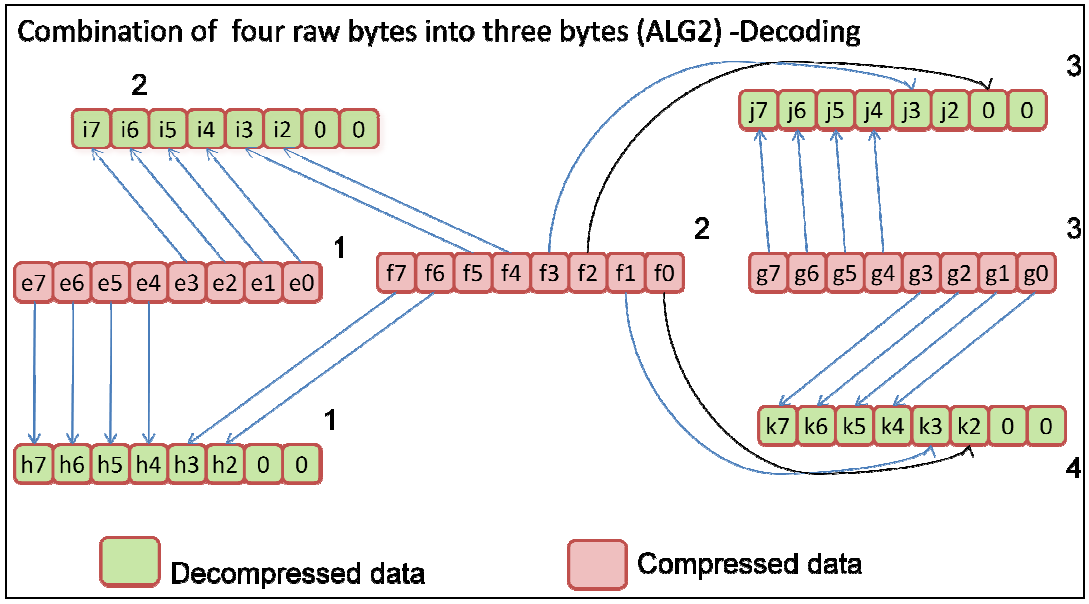


Figure 5-4(b): ALG2 Decoding Schematic

Figure 5-5: Result of ALG2

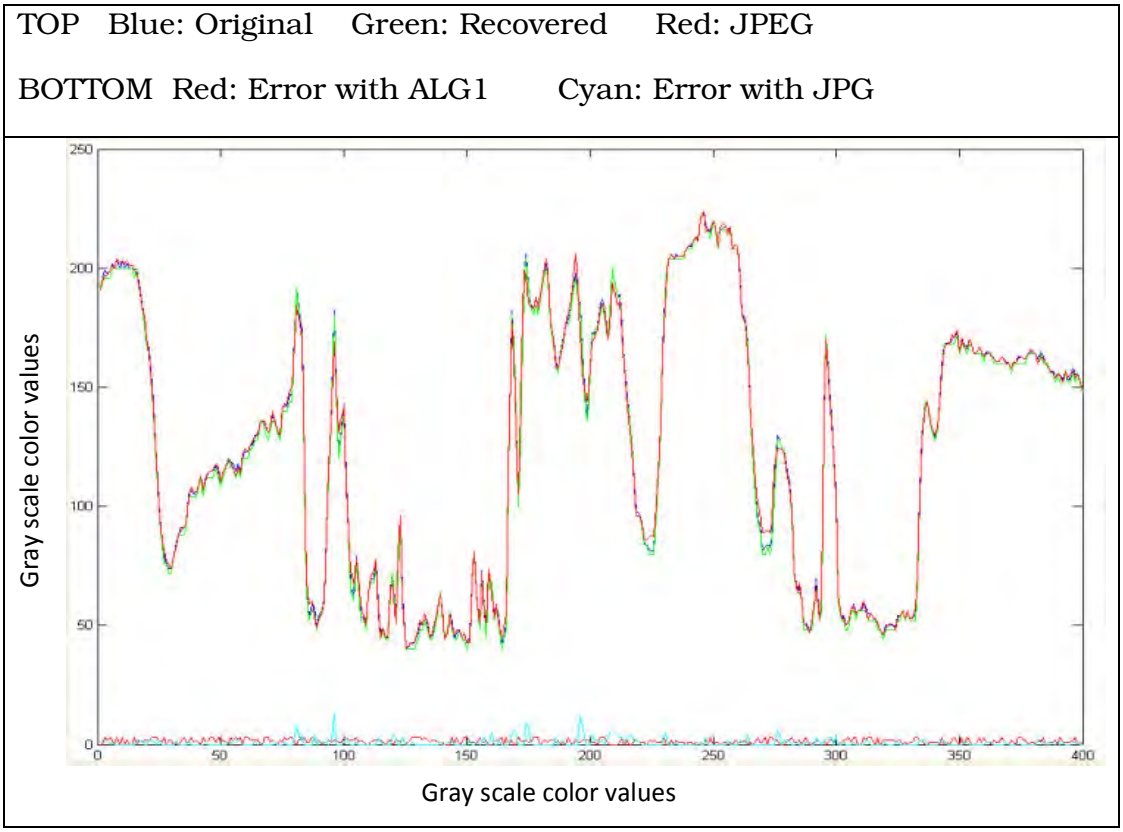


Figure 5-6: Graph showing data points from middle row of the test image for ALG2

The graph in fig 5-6 shows the data points before and after compression (of the images shown in Fig. 5-5) midway along the row. In a similar computation as in ALG1, the reconstruction error is 1.87. This implies a better visual quality than ALG1 at a cost of less compression ratio, i.e. 75%. The same computation using a JPEG compression resulted in a reconstruction error of 1.85.

5.3.2 Block Replacement (ALG3)

The idea behind this logic is that, most real photographic scenes do not change sharply in gray color values and we observe relatively smoother/continuous transition over a certain area. The process starts by dividing the image into blocks of certain number of pixels and processing each block separately. The result of this codec depends on the smoothness of the image (also referred to as texture) and the block size; the smoother the image and/or the smaller the block size, the better the result and the less compression ratio gain. The schematic in Fig. 5-7 shows the basic logic of the compression and decompression operations. Four options were considered and briefed as follows,

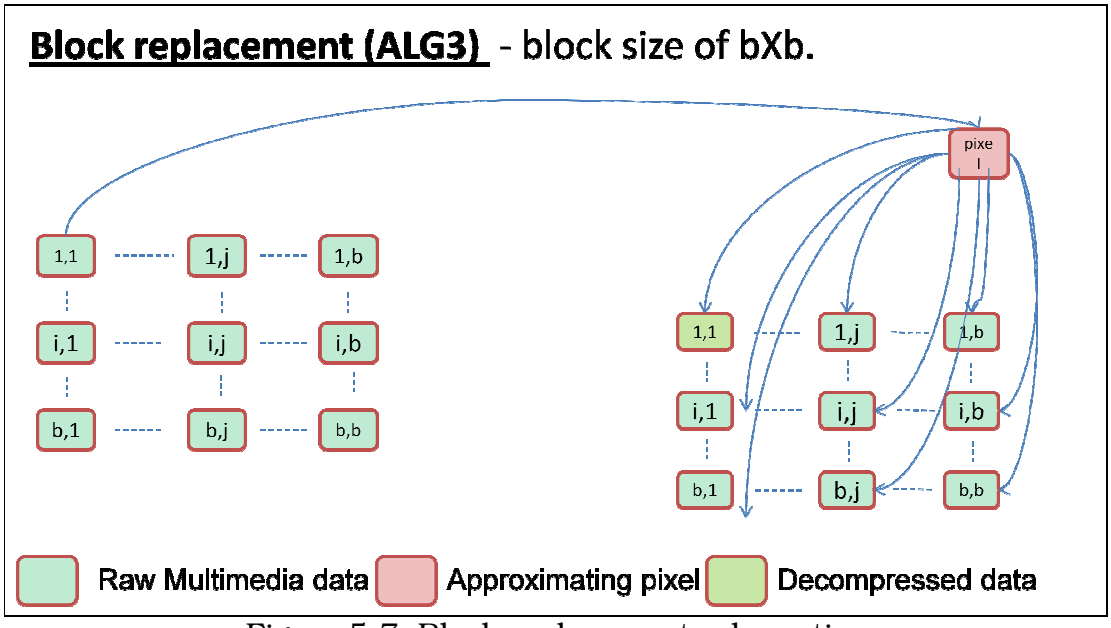


Figure 5-7: Block replacement schematics

Option 1: In this option, an input image is divided into blocks of size 2X2 pixels. Each of these blocks is represented by the minimum value of the neighboring pixels and stored in the compressed file. At the decoding stage, the pixels in a block are reconstructed having this single value. The resulting compression ratio is 25%, irrespective of the image characteristics.

Option 2: This option is a modification over option 1. The difference (offset) between the minimum value of a 2X2 pixel block and each neighboring pixel is computed and stored with the compressed file. Instead of encoding each value, the offsets are quantized as shown in Table 5-1. This is done due to the fact that the human vision system is sensitive to color variations only beyond a certain threshold. To test this threshold, several values were added to all pixels in the test image, while observing the similarity between the original image and the image after modification. A value 5 is subjectively assumed and is observed to provide acceptable resemblance between the modified image and the original one. The offsets are quantized into four values (0-3) and represented in binary by two bits as in Table 5-1.

Table 5-1: Offset Encoding

Offset ranges	Quantized value	Binary format
0-5	0	00
5-10	1	01
10-15	2	10
>15	3	11

The offsets are combined into a variable of type unsigned 8-bit integer so that four such offsets are packed into a byte of data. For instance, consider the 2X2 block of pixels in Fig. 5-2, taken from the test image.

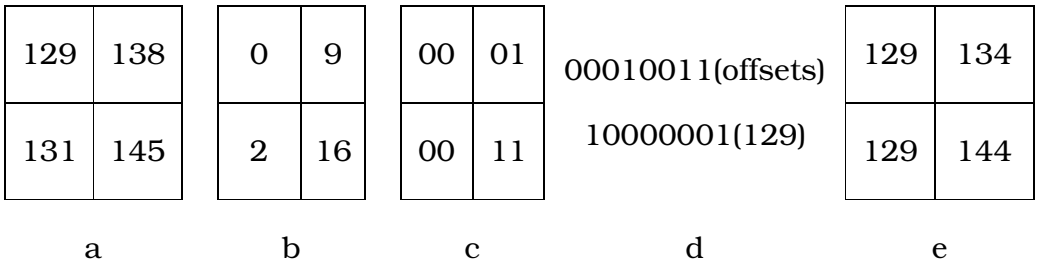


Figure 5-8: Example of option 2. a) Original block. b) Offset values. c) Binary format. d) The minimum value and the packed offsets e) reconstructed pixels

The block is represented by the minimum value of the group, which is 129. The offsets for each other pixels from this value are shown in (b) and the binary format is shown in (c). The minimum value 129 is stored with the offsets (c). At the decoding stage (e), the offsets are added to the minimum value to reconstruct each pixel (binary 00 is decoded as decimal 0, 01 as 5, 10 as 10 and 11 as 15 based on the quantization scheme in Table 5-1). Since the four pixels (each 1byte) are stored as two bytes, the resulting compression ratio is 50%, irrespective of the image characteristics.

Option 3: Similar to option 1, except that the block size is 3X3. The resulting compression ratio is 11% (100X1/9), irrespective of the image characteristics.

Option 4: This option is similar to option 2 with two main differences. First, a block is comprised of 9 pixels and the middle value of the 3X3 pixel block is taken as the representative value (unlike the minimum value of the block in option 2) and maps to the exact position during decompression. Second, the offsets are computed for the remaining 8 pixels and are encoded in 2bytes. As a result, the original 9 pixels (bytes) are represented with three bytes, giving a compression ratio of 33.33%.

The results from these tests are shown in fig 5-9. Fig. 5-10 shows the graph of data points from the original and decompressed image and their

errors for option3 and option 4. The graph also shows the data points and errors for JPEG-operated image for comparison.

The construction error and the compression ratios are shown in brackets in fig 5-9. We can see that, more pixel groupings result in more distortion of the image and loss of important details that may not be tolerable. Rather than simple replacement of the grouped pixels, the offset error 'recoding' has a smoothening effect which gives a better quality compression at the cost of lesser compression ratio.

Figure 5-9a: Original image.

Figure 5-9b: Result of ALG3. The values in brackets are in the form (reconstruction error, compression ratio).

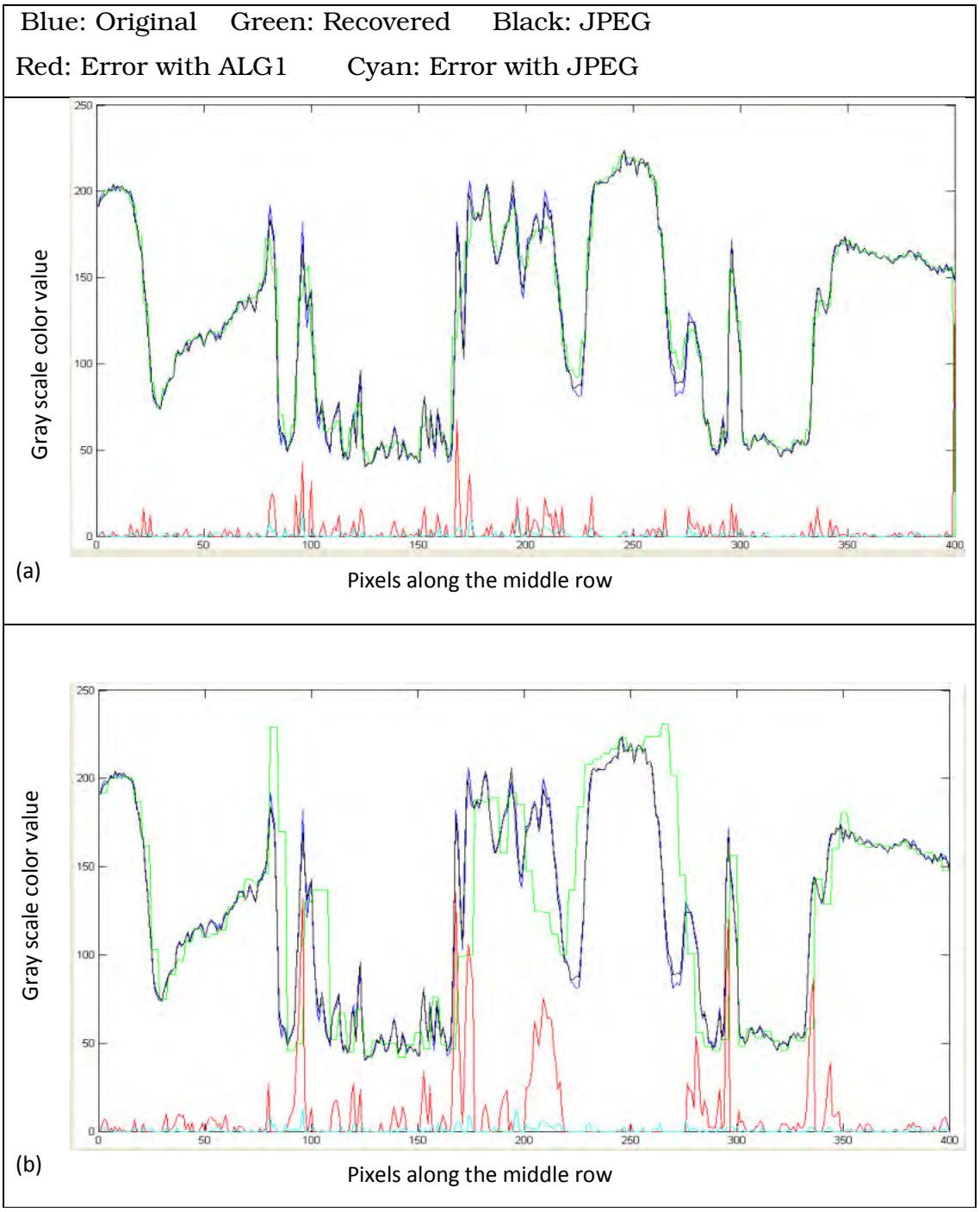


Figure 5-10: Graph showing data points from middle row of the test image for ALG3. a) Data points for Option3. b) Data points for option4

Table 5-2: Approach I test summary for image lena.bmp with uncompressed size of 468KB

Algorithm		encoding time (seconds)	Decoding time (Seconds)	Compressed size (KB)	Compression ratio (%)	Reconstruction error
ALG1		0.2190	2.7970	234	50	3.32
ALG2		2.4220	2.5310	351	75	1.87
ALG3	Option1	0.0160	0.3750	117	25	6.96
	Option2	7.8910	11.984	234	50	10.72
	Option3	≈0	0.2500	52	11	12.96
	Option4	5.6720	0.2350	156	33.33	18.43

Conclusion

The choice of an optimal algorithm depends on the intended purpose of the application. Since these methods result in a considerable loss of details compared to standard formats such as JPEG and TIFF, they are not best choices for accurate content retrieval systems or business class applications. The advantage with these methodologies is their simplicity. Unlike JPEG and other transform based compression techniques, the algorithms in this approach don't require floating point operations and can be executed by most legacy hardware. Otherwise, the compression shown above can be attained by other lossy codecs, as shown in section 5.4.3, by adjusting the parameters of speed, quality or the compression logic used. Maintaining compensations for error values (offsets) has a superior quality than simple block representation and the choice depends on the accumulated space requirement of the images to be processed.

As shown in Table 5-2, simple block replacement in option 3 of ALG3 is the fastest in processing time (both encoding and decoding) and the best in compression ratio. It would be appropriate in legacy and slow processor devices such as middle range cell phones and old computers. For devices over a network of limited bandwidth, this option, by virtue of least output size, can provide best download time. However, if quality and simplicity are required, then ALG2 is the ultimate choice as it gives better quality than the others by simple visual observation.

Images are one of the most digital media stored in massive amounts. If such an archiving is needed, while preserving reasonable quality, ALG1 is a choice as it preserves a quality near ALG2 and significantly saves storage size.

5.4 Approach II

This section presents an experiment of codecs comparison using products and standards developed for commercial use or as part of open source projects. The commercial applications used are unlicensed versions and no information was found as to whether these versions perform less or equal to the purchased ones. As a limitation, one of the applications (Digital Media Converter-DMC) limits the file length to be 3 minutes. For this reason, the files used for this music test and the speech audio test in the next section are 2 minutes and 30 seconds long to avoid boundary effects. Since such and other possible limitations are uniform, the tests done are valid. For the purpose of the comparison tests, four files shall be used as described in Table 5-3 below.

The experiments are carried out by applying lossless compressors, lossy compressors and/or lossless compressors followed by lossy ones. The option of compressing with a lossless (archiver) tool followed by a lossy codec is not a reasonable option for the fact that, once the archiver has compressed the multimedia data in a generic archive, the lossy multimedia codecs are left with no advantage of manipulating the characteristics of audio/image based on which they are basically designed. Furthermore, the formats created by the archiver(s) are not recognizable by the lossy multimedia algorithms, and hence, the procedure is not feasible.

5.4.1 Audio Music Test

The music test is performed on a WAV audio song by Tilahun Gessese (tilahun.wav). The first procedure was to test the best lossless

compressors (archivers) from the results of the experiments done in [11]. And next, other lossless and lossy codecs were applied on the original uncompressed file. There are numerous possibilities of hierarchically applying the algorithms one after the other. But a systematic approach has been used in such a way that test cases have been selected and applied by first using the lossy codecs followed by the lossless ones.

Table 5-3: Test categories, file descriptions

Test category	Description
Audio Music File	Tilahun.wav Uncompressed File Size: 25.2MB Resolution: 16 bits Bit Rate: 44.1KHZ Stereo type
Audio Speech File	DRdagnachewonradio.wav Uncompressed File Size: 25.2MB Resolution: 16 bits Bit Rate: 44.1KHZ Stereo type
Image File	Lena.bmp Uncompressed Dimension:400X400 Pixels File Size: 468.8 KB Color Depth: 24 bits RGB
Video File	Bigbang.avi Compressed File Size:26.2MB Frame dimension: 624 X 352X3 pixels Stereo Audio Frames per Second: 24fps

From a storage point of view, compressing an already compressed file would be an advantage. This is not possible for the fact that it would mean we can ultimately eliminate storage requirements for data in the extreme

case at a cost of enormous duration of time. For example, the file used in this experiment was first converted to MP3 and then converted to an AMR (Adaptive Multi rate) format which resulted in storage size of exactly the same as if the original file was directly converted to AMR. We can conclude from this fact that the codecs, when given an input in another compressed format, use an operation equivalent to first generating the uncompressed data and then apply their respective logic to create their own compact format. At the end of this section, it has been demonstrated that, hierarchically applying one codec after another is not worth doing from performance point of view.

The audio music test was first tried on a very high speed computer (core i3 processor and 4GB RAM). The registered time values were so short and overlap for most codecs. The values tabulated in this section were read from tests performed on a slower processor and lesser Memory Capacity computer, as listed in section 5-2. The advantage of the later choice is that, greater resolution of time readings could be made and error in stop-watch-time readings could be made minimal.

The result of the experiment is tabulated in Table A-1 in appendix A. The time measurements are done with a stop watch, for Graphical interface applications (GUI) like DMC 3.29 [49] and Switch Audio Converter [50], and from batch code, for command line (CL) tools. The output file sizes are read from the operating system file Properties window. To measure the reconstruction error, the encoded files are first converted back to the uncompressed format with the same tool used for encoding/compressing. Then, the original WAV files and the reconstructed Wav files are stored in Matlab array variables and root mean square (RMS) error is computed as formulated in equation 5-2. The compression ratios are computed using equation 5-1.

The data points are plotted in Fig. 5-11 and Fig. 5-12. Fig. 5-11 shows the graph for encoding and decoding time values. For the purpose of

readability, the compression ratio and reconstruction error values are plotted on a separate graph as shown in Fig. 5-12. The reconstruction error data points are multiplied by 100 to gain better resolution, since the plot for the real values (all less than 1) would not emphasize the variations compared to the much larger values of the compression ratio; the compression ratio is computed as a percentage by definition. The data points on horizontal axis in this and the rest of the tests are indexed into the referenced tables.

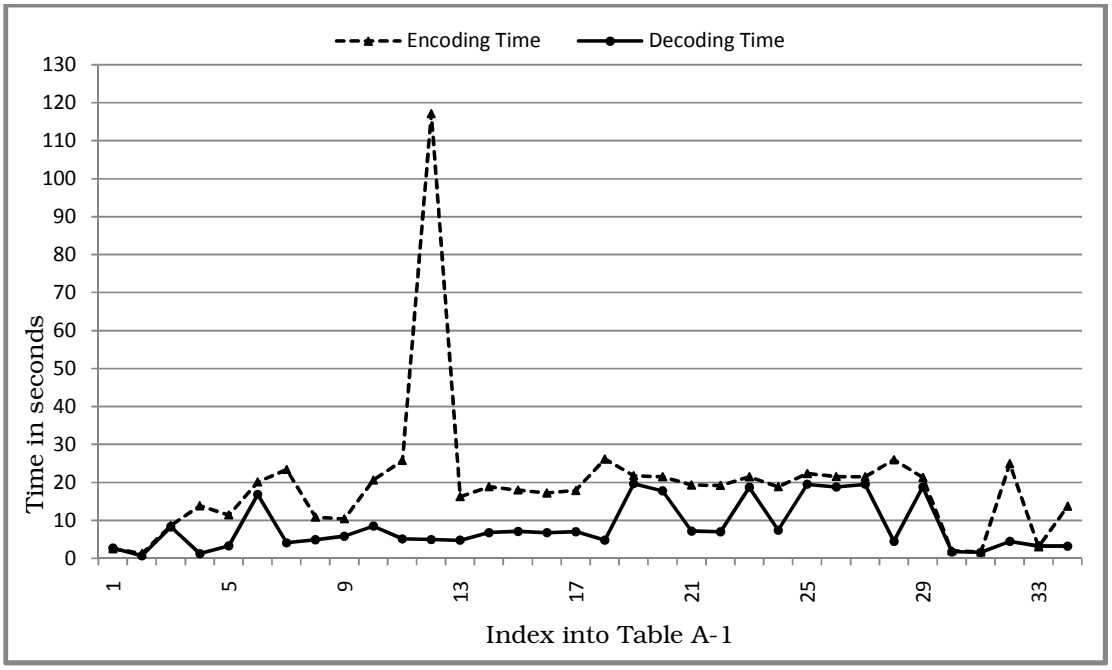


Figure 5-11: Plot of encoding and decoding times for Audio Music Test.

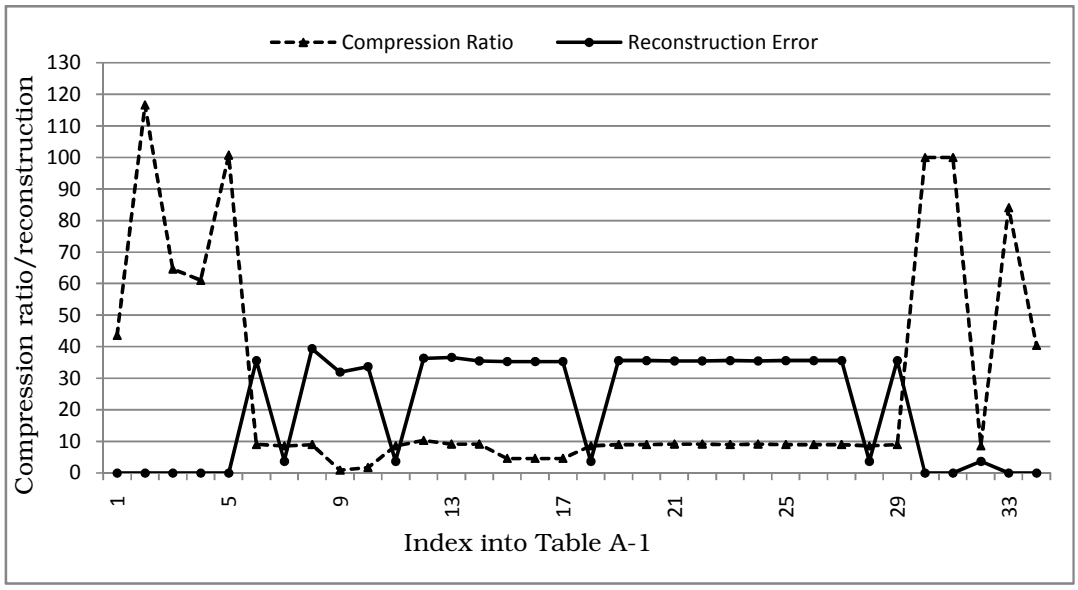


Figure 5-12. Plot of compression ratio and reconstruction error for Audio Music Test. The actual values are multiplied by 100 for the reconstruction error.

Conclusion

In Fig. 5-11 and Fig. 5-12, the lossless codecs are the points on the horizontal axis labeled 1 to 5, 30, 31, 33 and 34. From the graph, we can see that the fastest encoding and decoding codec corresponds to the data point 2, which is LZOP from Table A-1(1.25 and 0.688 seconds, respectively). But this codec has a negative impact on the compression ratio and we can see that the size of the 'compressed' file (29.4MB) is larger than the original file size (25.2MB). The data points 5, 30 and 31, corresponding to the codecs ZIP, AU and AIFF, respectively, provide no compression. Taking the need for compression into consideration, these codecs are of little importance for audio file. Considering encoding and decoding times with viable compression gain, Monkey audio archiver wins with the values 2.5 and 2.7 seconds, respectively. The problem with this codec is that, the resulting compressed file is not in a format ready to be played by most audio players. FLAC results in a much slower performance in encoding (13.7 seconds) and decoding (3.21 seconds). The compression

ratio of FLAC (40.48%) is slightly better than Monkey (43.65%). Hence, the bandwidth utilization, which depends solely on the transmitted file size, of FLAC and Monkey compressed files are almost the same. The intact quality of the results from the lossless codecs is confirmed from the zero reconstruction error in Table A-1 in and the graph in Fig. 5-12. In general, if maintaining the whole audio information content is necessary, i.e. if lossless processing is required, Monkey audio is preferable.

The data for lossy codecs are collected using two applications; DMC and Switch. Some codecs are tested that target various hardware platforms, shown in Table A-1, other than personal computers. The configuration settings each audio specific codecs are listed in the order of bit rate (in kbps), number of channels (stereo or mono) and sample rate (in khz). Another fact that has been observed is that, each version of the codecs that target different platforms have similar measurements. For example, there are eight tests for MP3 codec intended for various platforms and the measurements show similar performance parameter values for this group. The test has been done under the same set of codec settings (128 kbps, stereo channels, and 44.1 kHz) except when the codecs do not support one or more of them, in which case the nearest values were selected. It is apparent that the values of the evaluation parameters depend on these settings. Hence, the higher the bit rates in kbps, the higher the encoding and decoding time, and the higher the compression ratio(lower is best). The channel affects the parameters in such a way that, the more the channels the higher the processing time and the compression ratio will be (i.e. more bandwidth requirement).

With respect to the lossy codecs, the best compression ratio is attained using AMR codec with a compression ratio of less than 1%. The conversion time of this algorithm is also well acceptable as it is in near range of the lowest time recorded in the test. But the problem with this codec is that, the resulting audio is heavily compromised with respect to

quality (observed subjectively) and can't be used for entertainment purposes. The AMR file has been converted back to WAV file and the quality (subjective) is the same as playing the AMR file, implying that immense amount of audio information is permanently lost. One unexpected result from this experiment is that, the reconstruction error values for lossy codecs don't reveal the quality loss during compression. We can see that the reconstruction error for AMR is computed to be 0.32 while the value 0.39 is computed for a much better quality output from AAC. AMR can be used for archiving purposes or historical use and in a very poor networking infrastructure. Besides, the resulting compressed file is not played by all music players.

For non-networked playing, where there is not real need for transmission bandwidth, the encoding and decoding times are the only ones to-be-worried about parameters. Accordingly, AC3 has the fastest encoding time (10.83 seconds). WMA comes next in encoding performance but with added advantages. WMA is available in various flavors such as for desktop playing, internet streaming and for progressive download (playing while downloading). So, one may consider this added advantages when selecting an encoder. The draw back with WMA is that it is proprietary to Microsoft Corporation. AAC is the next best encoding codec that requires no license fee and can be used if free distribution of audio contents is needed. MP3, even though it encodes audio faster than AAC, requires a loyalty fee to be in the legal framework. Considering decoding time parameters, AAC has the best performance.

Compression ratio has a direct impact on the bandwidth requirement when considering networked access; the better the compression ratio, the lesser the bandwidth requirement. Most of the local networks are so crippled that, users hardly enjoy multimedia streaming over the internet. From the test codecs, AMR would be a choice in cases where users need the basic information content rather than high quality requirement.

Otherwise, the version of the WMA for progressive download is preferable. From the 'openness' perspective is the next best compression codec available.

We can see that, MP3 is out performed by the other algorithms based on the comparison parameters. Nonetheless, MP3 is the most popular codec used by diverse set of users because of the fact that it is the first high quality audio codec.

The next test was to see if the hierarchical application of a lossy algorithm followed by a lossless one results in a better storage savings, and hence, lesser bandwidth requirements. For this purpose, the wave file used in the test was converted to MP3, OGG-Vorbis and AMR formats followed by Monkey Audio and WinRar. The data are shown in table 5-4.

As shown in Table 5-4, multi level compressions with the indicated options is not interesting with the goal of codecs being providing lesser file sizes. Even in the case of Monkey audio codec, the file size has significantly increased and simply should not generally be used. We can use this form of compression if we need the encryption options of some archiving tools.

Table 5-4: Hierarchical application of lossless codecs on the outputs of lossy ones.

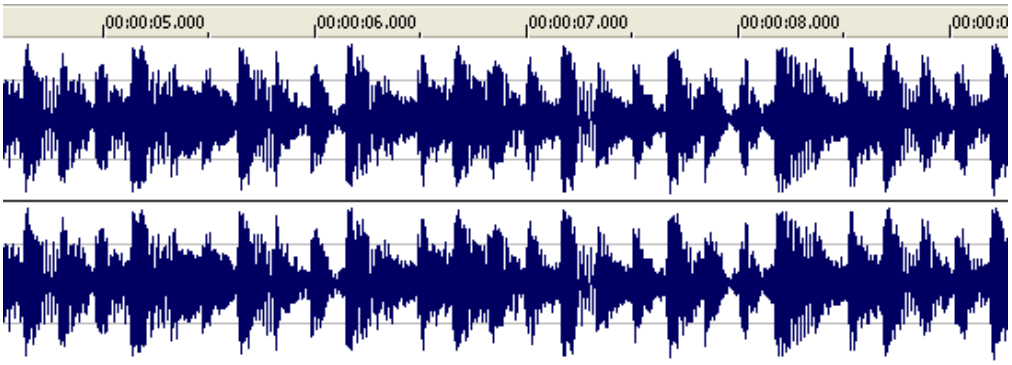
From	Size in MB	To	Size in MB	Compression ratio (%)
MP3	2.74	Monkey's ape	11.06	423.00
		WinRar	2.56	93.04
OGG	2.98	Monkey's ape	11.06	389.03
		WinRar	2.74	91.09
AMR	0.281	Monkey's ape	---	---
		WinRar	0.268	95.4

Based on the experiment, for example, a company in a music industry may need to archive all the songs in the world for historical significance,

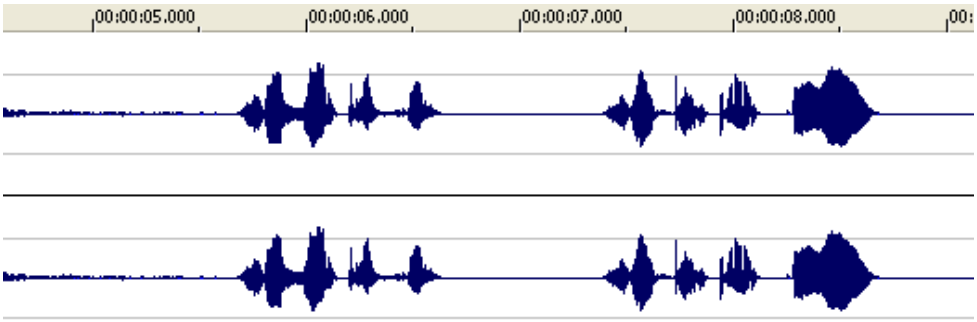
with less concern to the entertaining aspect of the data and can afford the long time of archiving. In this case an AMR format would be a de facto format that can save tera bytes of memory and the associated costs and physical space requirement. Considering the other case of a company which makes songs for entertainment purposes and nothing else, the use of MP3 is a choice if it needs to reach the widest audience. However, if the business or social goal expects mostly PC users and need internet accessibility, WMA is the best choice available as it is both a very high quality and excellent compression gain audio codec. The other side advantage with high quality (meaning preserving most information content) codec is that, search of audio content can be more accurate. WMA is also a choice in a low bandwidth network infrastructure and where the end users experience is heavily Windows Platform. For other platforms like Linux and Mac systems, MP3 is the next best alternative to OGG, where OGG player is not available. AAC, which is very similar to MP3, can be used where ever possible. But AAC has additional features such as supporting wider range of frequencies and security features. AAC is also more appropriate than MP3 for telephone conversation since it processes data on the fly and less delay than MP3.

5.4.2 Audio Speech Test:

Even though both speech and music are sensed by the ears and they do share common set of characteristics, there is an interesting difference between them that makes speech much more feasible for higher level of compression than music data. The waveforms of the music file tilahun.wav and an interview made on a radio program has been seen on Sony Sound Forge 9.0 audio application, as shown in Fig. 5-13. The amplitude to time graph shows that, the music file (a) has a considerable rate of change in amplitude and stays at zero for a much lesser amount of time than the speech sound (b). The same procedure has been used to measure the evaluation parameters as that of the Audio Music Test done in the previous section.



(a)



(b)

Figure 5-13: (a) Music signal pattern (b) speech signal pattern

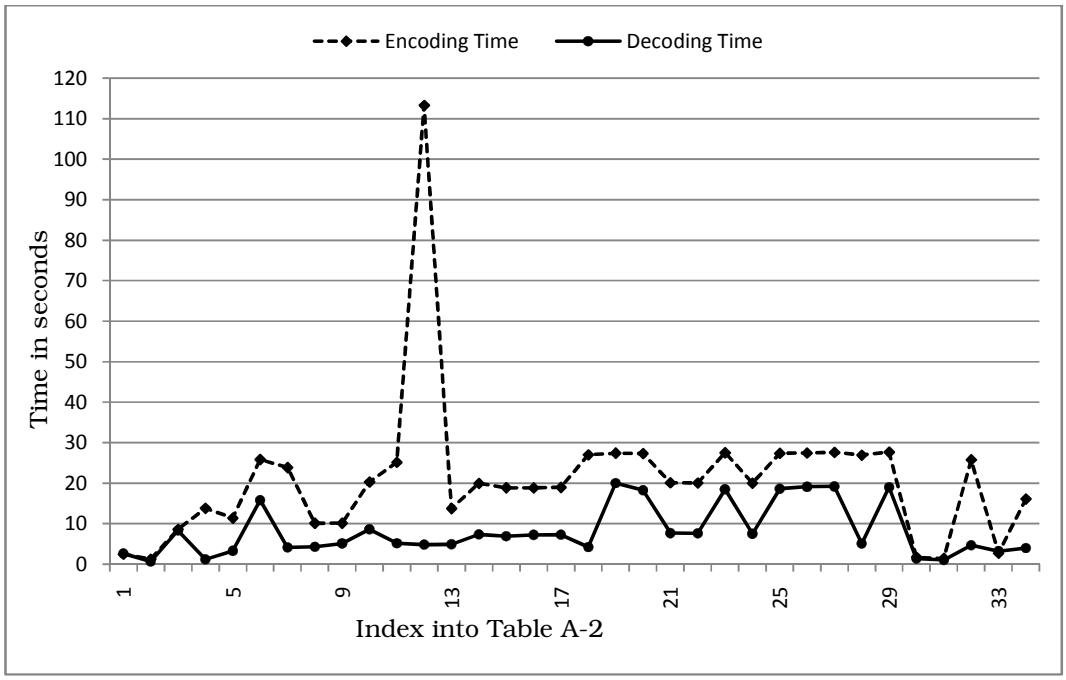


Figure 5-14: Plot of encoding and decoding time for speech test.

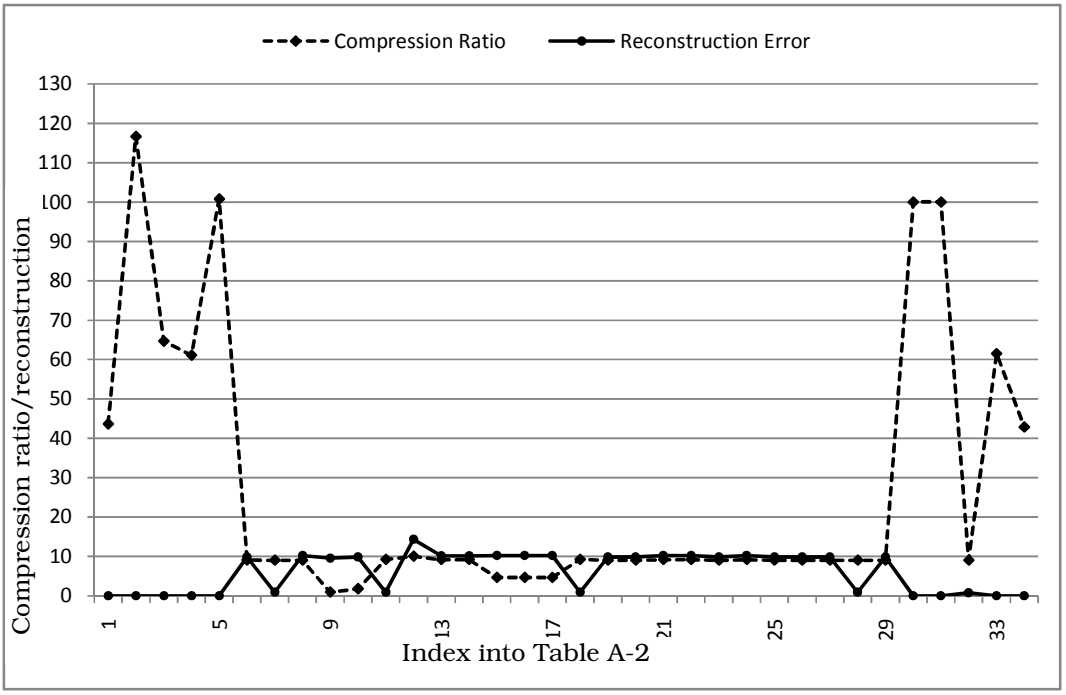


Figure 5-15: Plot of compression ratio and reconstruction error (multiplied by 100) for speech test

Conclusion

Comparing the graphs in Fig. 5-11 and Fig. 5-12 for Audio Music tests with the ones in Fig. 5-14 and Fig. 5-15 for Audio Speech Test , we can see that the amount of time for processing and the space requirement are very similar for the same duration of music and speech. The only visible difference is observed in the reconstruction error data points. Even though the numerical value of the reconstruction error doesn't reflect the operational qualities among the codecs, this value is an indication of the quality loss for music and speech data in the big picture. The reconstruction errors from Fig. 5-12 for music are higher than the ones shown in Fig. 5-15 for speech. But the quality requirement of a speech is not as compulsive as that of a music data. As shown in Fig. 5-13, music signals have lesser pauses and near zero values than speech signals. This fact indicates that, the loss of relatively small magnitude signal amplitudes may be relatively affordable resulting in a generally smaller reconstruction error values.

Unless the speech is required for business and professional class purposes, the AMR compression is a number one choice for content focused applications. As for other applications, the same comparison can be made as that of the music audio discussed in the previous section.

5.4.3 Image test

The image test has been done on the uncompressed version of popular picture of Lena (lena.bmp); this image has been selected as it is the most popular test image used in image processing algorithms. There are several conversion tools available for inter-conversion between the different image formats. Some of these are graphical user interface programs, command line tools and online applications. For this thesis work, work Image Convertor Plus [56] has been used as it has, in addition to graphical user interface, the most complete command line tool. The choice is done since it is convenient to run batch commands for mass conversion and the

program records the time of execution in its log files, along with several other details. In addition, the time measurement is more accurate than the stop watch way.

For each conversion test, 50 of the files are used to measure longer time than the very small execution time (in the order of milliseconds) for processing a single image file. This gives a better resolution of the time differences for the tests done. The data are populated in Table B-1 tabulated in appendix B. Each of the algorithms has different set of parameter settings and some of these parameters are applicable to some codecs and not to others. Fig. 5-16 show graph of the encoding and decoding times and Fig. 5-17 shows the plots of the compression ratio and the reconstruction error data points from the table in appendix B. The reconstruction error values have been computed by importing the input and output images into Matlab. The images were represented by matrix variables and equation 5-2 has been used. The computation has been done over the entire image pixels and color components.

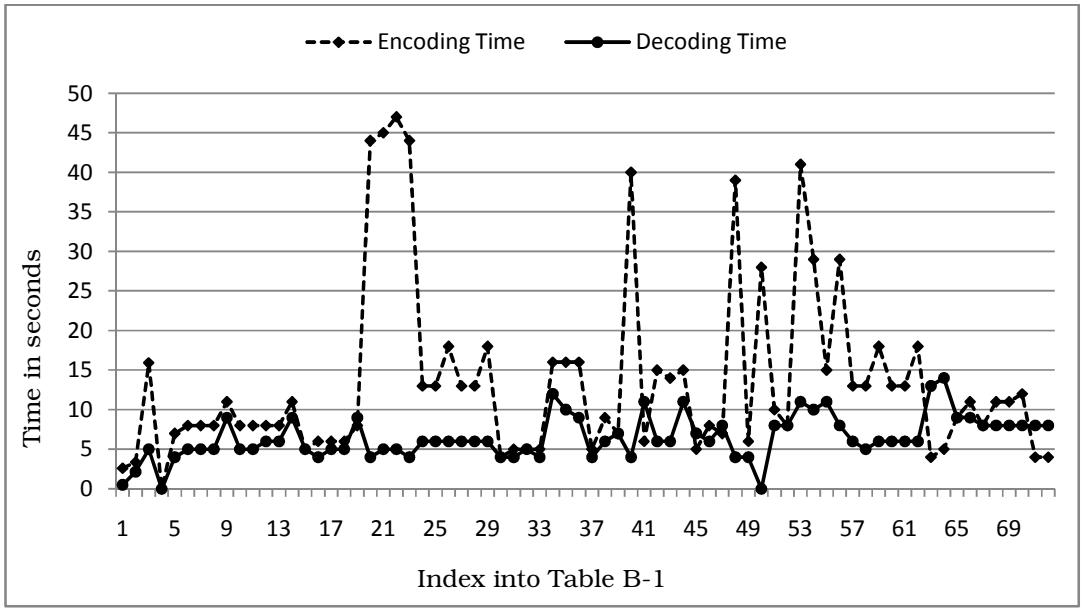


Figure 5-16: Plot of encoding and decoding time for image test

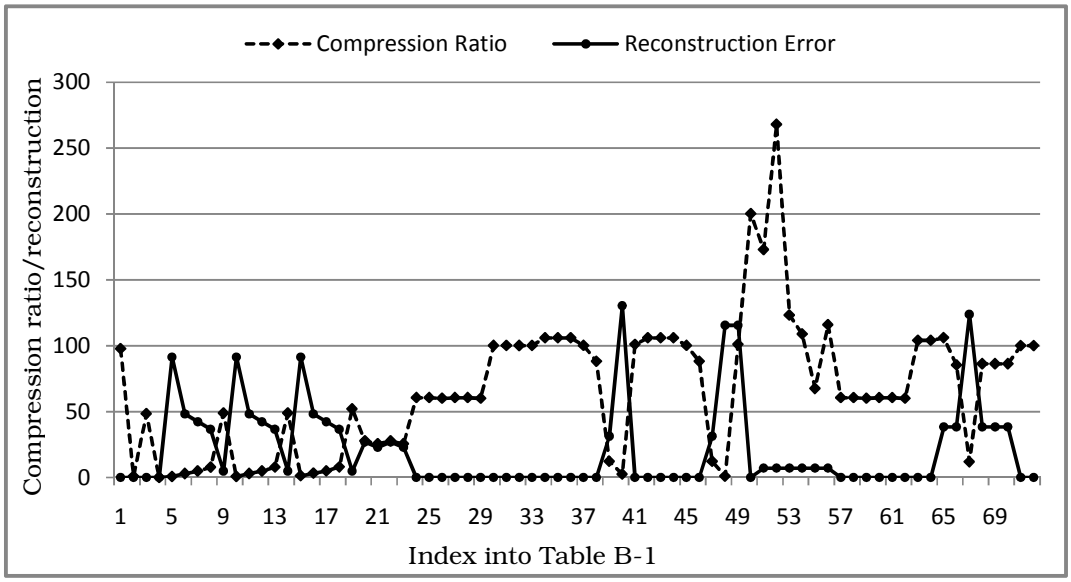


Figure 5-17. Plot of compression ratio and reconstruction error (multiplied by 10) for image test

Conclusion

The codecs tabulated in Table B-1 can be generally categorized into two types: Lossy and Lossless. Some of the codecs can operate in both modes by setting the values of some of the parameters. The lossless nature of the algorithms can be dictated from the value of the reconstruction error, which is zero. For lossy algorithms, this value is none zero.

Before making any form of comparison, some assumptions are made for convenience. Compared to processing time, compression ratio, and hence bandwidth requirement, is more of a challenge so that we can take out the algorithms with compression ratios greater that 50% (the value is intuitively assumed).

Considering the lossless image processing, on the course of the experiment, it has been found that the codec stuffIT has a property that mass conversion yields much better result than single file conversion. From the table, the mass conversion of the 50 images resulted in compression ratio of 0.85% and a single file conversion resulted in 40% compression. This has been seen only in this codec. In both cases the,

stuffIT has the best compression among the lossless compression codecs tested. Considering the compression and decompression times, noting a reasonable compression ratio, stuffIT is the best. Even though WinZIP (compression ratio of 85%) and LZOP (compression ratio of 97.77%) appears to have a faster processing time, the compression ratios of these codecs are too poor compared to that of stuffIT. But the WinZip, LZOP and stuffIT are general purpose archiving tools. Hence, the image specific formats are compared separately. Noting the need for reasonable compression, PNG is the fastest and best compression lossless codec for images. In addition, the use is not subjected to any payment, unlike GIF, and not as complicated as TIFF specification.

With respect to lossy processing, we need to consider one fact. Unlike audio, preserving the quality is vital in almost all imaging applications (except for artistic purposes – cartooning, for example) at least to the level where each object is identifiable by its color. Therefore, before making any comparisons, we shall wipe out the most visible poor quality codec and associated settings. Unlike the audio tests, in image test, the reconstruction error is found to reflect the quality of the output image from the codecs. In this test codecs resulting in a reconstruction error of more than 9.0 has been ignored. This leaves GIF, TIFF and JPEG and their setting for lossy image codec comparison. The fastest encoding and decoder of these codecs is JPEG setting the quality to 100, and without using optimized Huffman encoding (a lossless encoding stage following a lossy operation). The recorded time was (for processing 50 images) 6 seconds for encoding and 4 seconds for decoding. With respect to compression ratio, considering reasonable quality as discussed above, a variation of JPEG (setting quality to be 25 and using optimized Huffman encoding) is found to win the race with a value of 2.9%.

From the above analysis, we can choose the specific combination of codec and configuration settings to suit a wide variety of applications. We can

see that, stuffIT is an ideal tool for mass storage of image data and transmitting these data over a network, as it becomes most storage efficient, and hence, efficient bandwidth utilization, when processing large amount of data at the same time. For this reason, beyond a certain number of files, stuffIT can be more bandwidth and storage efficient than lossy codecs. In addition, for most accurate content retrieval system and medical image management, it is preferable as it is the best performance lossless codec. For lossless image specific codecs, PNG is preferable. PNG supported all the features available in GIF (except support for animation) and TIFF (except for storing multiple images in one file).

In applications where certain losses are permissible such as web pages, JPEG is a preferred choice. JPEG has the most diverse configuration settings that affect the processing time and compression ratio.

5.4.4 Video test

The test file used is 2 minutes duration of clip from the BigBang Series movie. The clip is a good quality movie in an AVI. AVI is a container format for any encoded video and audio data. The AVI file format stores the associated codec used for the data it contains. The uncompressed data would take up Giga Bytes of memory space. The uncompressed size can be calculated by multiplying the frame size (Width X Height), number of frames (frame per second X duration in seconds) and the color depth. Using the data in Table 5-3, the uncompressed sum of the video and audio data would be 1.79GB. As the test is performed on a relatively high quality compressed data, the test reveals the relative performance parameters of the various codecs.

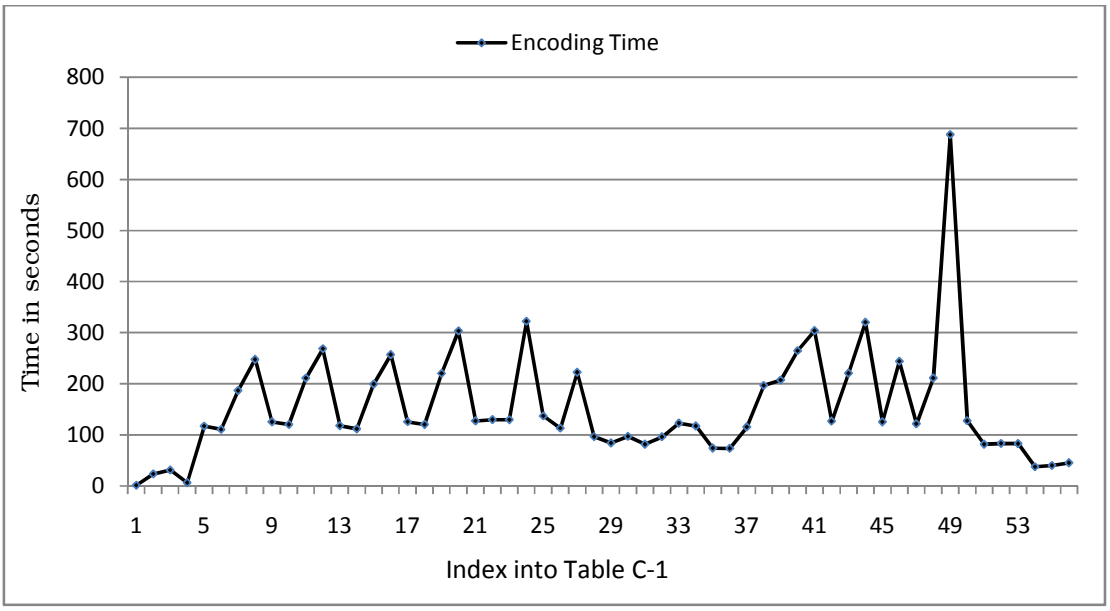


Figure 5-18: Plot of encoding time for video test

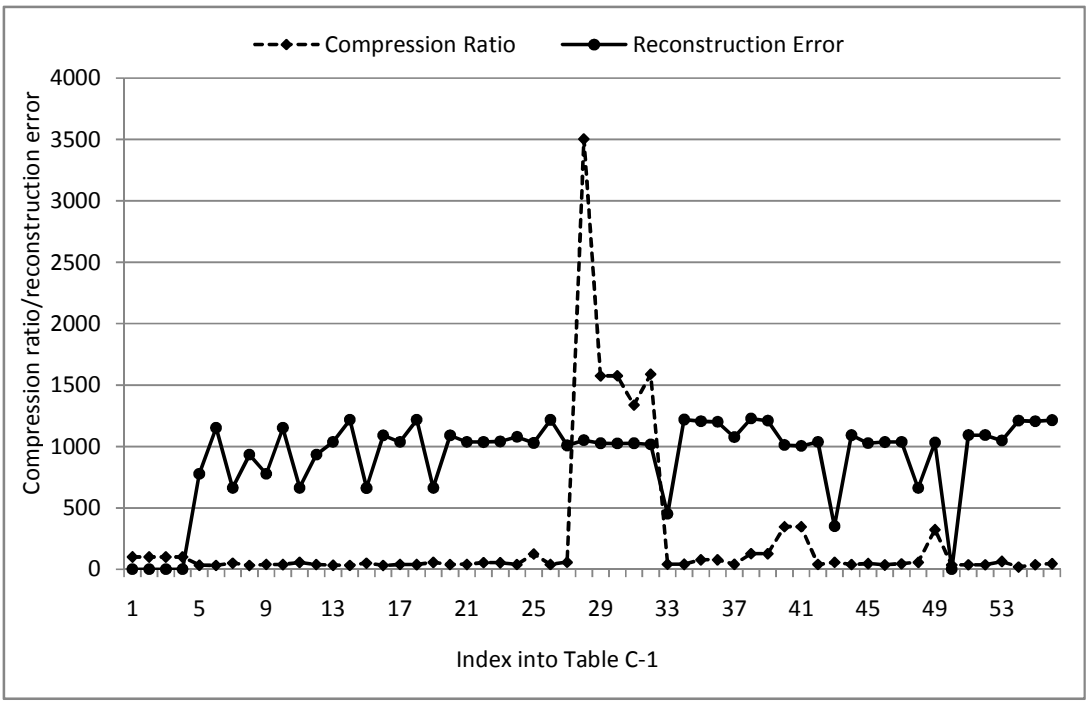


Figure 5-19: Plot of compression ratio and reconstruction error (multiplied by 100) for video test

The program used for the conversion is Digital Media Convertor 3.29[49]. The data for this test has been tabulated in Table C-1 in appendix C. In this test, the decoding time has not been tested. This would have been done if the original data was in an uncompressed format. The reconstruction error has been computed by comparing the first frame of the 56th second of the play time. This frame is one of the frames in a scene with motion of a person. The frames where captured from AVS Media Player for all the test outputs. Stop watch has been used to measure the encoding execution time. Since the computation of the reconstruction error is done on a single frame of the moving pictures, the same procedure as that of the image test has been used.

Conclusion

The lossless codecs are the fastest in compressing the video data. These are general archiving tools that are not specifically designed for multimedia operations. Even though they perform faster than any of the video specific codecs, they provide no real compression gain. The best compression ration attained was 99.42% by WinZip. Hence, there is no need for using lossless alternatives for video data. One exception to this is if transmission of the data is required where firewall settings block the video data in the original format.

The lossy codecs have range of execution time and visual qualities. The fastest encoder for the test done was the Windows Media Video codec customized for internet streaming (37.75 seconds). This codec also resulted in the best compression (18.20%). The quality of the output is affected in such a way that pictures are blurred and not as sharp as the original frames. For desktop playing, the fastest (73.84 seconds) codec was found to be the MPEG 1 video codec used in VCD (PAL) format.

Among the desktop playing formats, the best compression ratio was registered for 3GP format using H.263 codec with the value of 31.00%. The distortion in the output is visible with respect to change in the height

and width of the original video. H.263 supports only predefined dimensions. The next best compression ratio (31.23) has been obtained for 3GP format using the Xvid codec. The visual quality of the output from this codec is similar to the original data. However, Xvid has the slowest performance as it has an encoding time of relatively large value (247.85 seconds).

Irrespective of the format used, there is a trend in the codecs with respect to the values of the measured parameters. Xvid is generally the slowest encoder but good compression ratio and visual quality. WMV generally has good performance (encoding time) in encoding video data.

When selecting the optimal codec for a specific application several factors are affecting the choice to be made. Some codecs have resulted in a significantly high value of compression ratio (bad compression gain) and still applicable in real system. One such codec is the NTSC which is used in television broadcasting. The nature of television broadcasting is that, contents are continuously streamed to the receiver and there is little concern for bandwidth utilization in the analogue channel used in the system as long as the channel is carrying all the frequencies in the transmitted signal. The sound quality also needs to be taken into consideration with the moving pictures' (frames') quality.

Some of the computed reconstruction error values were relatively very high on the outputs of some codecs that only support specific frame sizes (for example, H.263 -11.501%). The values indicated for reconstruction error in Fig.5-19 shows the computed value as in equation 5-2 multiplied by 100 for clear observation.

5.4.5 Bandwidth Test

In places where there is poor networking infrastructure for data transmission, it is necessary to keep the file size as small as possible. For this test, the video file has been used to study the theoretical assertion of

the direct relationship between file size and transmission time over a fixed network media. The test has been done by storing the output files from the video test in section 5.4.4 on the test computer and a batch code has been written to copy the files one by one to another computer that has been used in the rest of the experiments. The time measurement is done from the log file of the batch process using a java code. The transmission media was a wireless network (802.11b wireless standard) between the two laptop computers used in this thesis work. The data has been tabulated in Appendix D. The graph in Fig. 5-20 shows the relationship between the time taken to transmit a given data and its size for the test cases in the video test in section 5.4.4. It is always the case that, more data size takes more time for transmitting a given data.

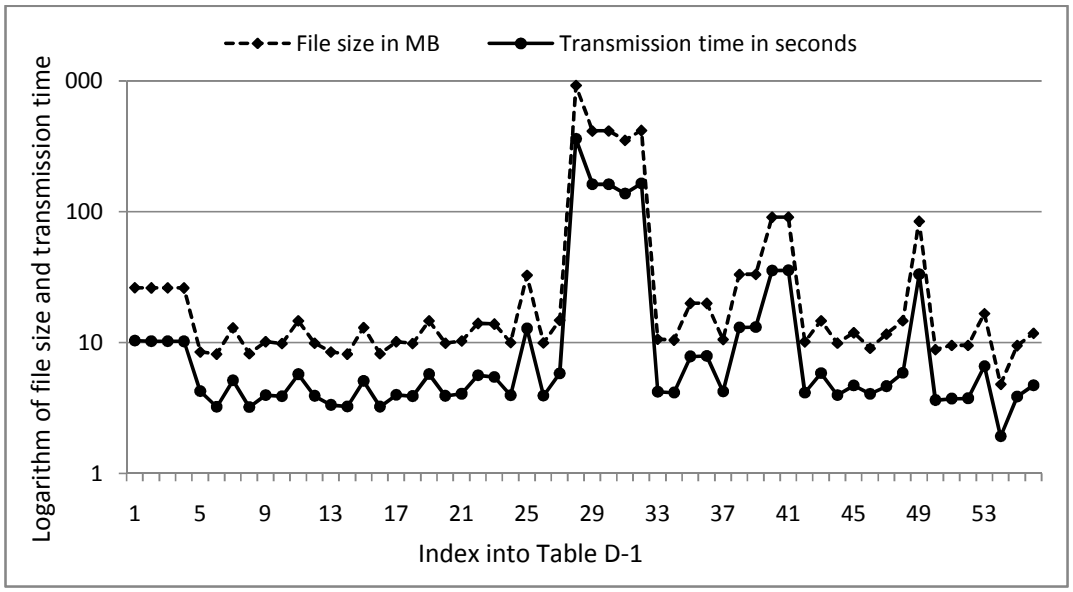


Figure 5-20: Graph of memory size and transmission time for the data in Appendix D. The vertical axis is shown in logarithmic scale.

Conclusion

From the experiment done in this section, we can see that the amount of time required to transmit data over a given networking media depends on the size of the data in units of memory. Hence, for effective bandwidth

utilization or to use a limited bandwidth channel, the amount of data to be transmitted needs to be kept as small as possible.

5.4.6 Security Test

In the process of securing any form of data, no data is expected to be lost in the process. The multimedia specific codecs used in this thesis work do not provide security support during the compression and decompression. Therefore, the lossless archivers have been used on the video data to compare the performance parameters by testing cases with and without encryption. The video data is selected for the fact that, by virtue of its large size, the worst case results can be observed.

Three of the tools used in this thesis work have encryption options during archiving by supplying a password. A password length of 10 characters is used in all the cases. The data points are tabulated in Table 5-5.

Table 5-5: Data table for Security test

Archiver	StaffIT	WinRar	WinZip
Type	LL	LL	LL
Parameters	Pwd=0123456789	Pwd=0123456789	Pwd=0123456789
Compression time (sec)	46.83	69.64	8.41
Encryption time (sec)	47.20	72.02	10.27
Decompression (sec)	55.93	2.68	2.06
Decryption time (sec)	55.07	4.89	3.47
Output size of compression (MB)	26.13	26.15	26.12
Output size of encryption (MB)	26.13	26.15	26.12
Compression ratio of compression (%)	99.48	99.54	99.42
Compression ratio of encryption (%)	99.48	99.54	99.42

Legend:

MB=Megabytes

Pwd=password

LL=Lossless

Conclusion

Several conclusions can be drawn from Table 5-5. WinZip is the fastest in compression and decompression with or without encryption applied. We can see from the compression ratio values that, the output memory sizes of compression with or without security applied are almost identical. The table shows the compression ratio values in units of Mega Bytes (MB); the actual values differ by some number of bytes. The encryption and decryption times are larger in value than the compression and decompression times with no security, respectively. This execution time difference is attributed to the additional mathematical processing time taken during encryption and decryption.

Chapter 6

Summary and Conclusion

6.1 Summary

This thesis work has been concerned with the performance evaluation of multimedia encoding and decoding algorithms (or codec, as it is mostly referred to). The evaluation has been done on defined set of parameters, namely, Compression ratio, Storage requirement, Bandwidth requirement, Encoding execution time, Decoding execution time, Quality and Reconstruction Error. These parameters are defined in section 5.2. The experiment has been done in two categories: Approach I and Approach II.

In approach I, some algorithms have been selected from literatures and implemented in Matlab for image data. The performance parameters are computed using the same tool (Matlab).

In approach II, four types of multimedia data have been used for testing; Audio music, Audio speech, Image and video data. Though music and speech are both audio contents, the natures of the signals are different. Music is rich in pitches and melody and speech is a plain conversation with longer pauses and repeating bursts. The test audio codecs have similar performance parameters for both music and speech samples. But the quality compromise is better affordable for speech than music which is mainly used for entertainment purposes. The test has been done for both PC use and branded hardware targets. There are cases where a codec has versions for several targets. For each codec, the computed performance parameters have similar values when tested for all the target players. The best codecs for the defined parameters have been identified with possible areas of applications. For image and video codecs, similar set of parameters have been computed and conclusion has been made based on the values. The relationship between the memory size and bandwidth requirement have been tested using the outputs of the video test by

measuring the amount of time taken to transmit each of the files across an ad-hoc wireless network between two computers kept behind walls.

Finally, security test has been done using three lossless archiving tools to see the impact of encryption on the performance parameters.

6.2 Conclusion

The tests performed in this thesis work have been done in several categories on various multimedia data. The major target of the work is to make comparison of multimedia algorithms with respect to measurable parameters such as processing time and compression ratios. In addition to these measureable quantities, qualitative observation of the test algorithms was done. Multimedia data are meant for human to view (such data as images and movies) and listen (as in music and speech). The measureable parameters such as compression time, decompression time and compression ratios are values that definitively give an insight of how the algorithms work. The quality measure is difficult to quantitatively define due to the nature of the data involved. For instance, taking the image tests, the codecs use special data processing scheme to attain the required level of performance and quality. Most of these codecs manipulate the similarity in neighboring pixel values and operate on blocks of the input image. Consider, for example, the whole of the input and encoded/compressed output images processed by a number of codecs. Some codecs may result in a lesser loss of pixel values on some blocks while other codecs may result in lesser loss on other blocks depending on the color composition in the processed blocks and the internal logic (i.e. the constants values used in the mathematical model). So, the single value of the reconstruction error computed over the whole image may not necessarily reflect the visual quality of the recovered image after compression of the original image.

The timing parameters, compression and decompression times, are affected by the processes running on the test computer. To minimize the

effect of computing platforms such as operating system task scheduling, caching, program initialization and other factors, the experiments were made with a fresh copy of the operating system installed on the test computers. Furthermore, only software necessary for the experiments were installed and the tests were repeated until a stable value has been obtained.

The difficulty with multimedia data mainly arises when they are used in a distributed environment and on low memory devices such as cell phones. Even, wired networks built in most low economy countries are poor. This is due to the large number of users using the backbone network and low profile networking devices used to built each segment of the overall network infrastructure. In such cases, the amount of data transmitted needs to be kept as small as possible. It is apparent from the data collected in the experiments that the parameters are dependent on each other. There is always a tradeoff between processing time, quality and compression ratio. But the relationship is not simple to generalize. For example, a slower codec does not always result in best compression. Further complication in the relationship between the performance parameters results due to the fact that the programs to implement the codecs are done by irrespective organization. Programming skill and logic used by each codec affects the execution time of the programs in an unpredictable way. To minimize these effects to some extent, for each test, the most comprehensive software encoder and decoder software products have been used, as mentioned in the respective tests in chapter 5.

In performance tests spanning such a wide range of data types and large number of algorithms, it is imperative to use black-box testing approach. So, all the results attained in this thesis work uses the codecs as-is.

The major observations and conclusions are:

- The audio tests were made for both music and speech data. The result shows the performance parameters are similar for both. This implies that the pauses common in speech data is not manipulated in any other way than the much lesser pauses seen in music data. That advantage can only be used by the user of the data in deciding the affordable loss of quality. For instance, AMR is the best compression audio codec but results in a much poorer quality. But one can use this codec to store and network-transmit for communicating the contents of audio data over the internet.

- In the case of image test, several observations have been made. One interesting fact is that, mass processing of large number of images can better be processed with lossless compressors than lossy codecs, which are supposed to provide better compression gain at the cost of losing quality. StuffIT archiver proved to be a choice in such a case. JPEG is the most flexible and has better performance with selected parameter settings; hence, is recommended in most situations.

- In the case of video codec test, the comparative selection of 'best' codec is even more complicated due to the combined effect of the moving pictures/frames and the associated audio content. Not all video formats support all audio formats. Therefore, the data sets can be more important than any decision made based on the data.

- The bandwidth test proves the fact that it is always true that, keeping other factors the same, more size of data always takes more time to transmit over a network media.

- Finally, the security test has shown that, applying encryption for securing the data affects the performance parameters. Accordingly, encryption increases the processing time during coding and

decoding and has little impact on the output memory size, and hence the bandwidth requirement.

In conclusion, for such diverse types of data and codecs, black box performance evaluation is a reasonable approach. Implementation and test of each codec can be feasible with large number of programmers, signal processing experts and software engineers. Such a test would provide an ultimate comparison under the most uniform implementation environment possible.

References

[1] Tarif Riyad Rahman and Miftahur Rahman, *Compression Algorithms for Audio-Video Streaming*, International conference on Intelligent systems, Modelling and Simulation, Department of Electrical Engineering and Computer Science (EECS), North South University Dhaka, Bangladesh, pp. 187-192, 2010.

[2] Meihua Gu, *An Inter-Frame Coding Algorithm based on Vector Quantization*, Lei Zhu College of Electronic Information Xi'an Polytechnic University Xi'an, China, Ningmei Yu, Dongfang Wang Department of Electronics Engineering Xi'an University of Technology Xi'an, China, pp.150-154, 2009.

[3] Jaemoon Kim and Chong-Min Kyung, *A Lossless Embedded Compression Using Significant Bit Truncation for HD Video Coding*, 2010

[4] Kwok-Wo Wong, Qiuzhen Lin, and Jianyong Chen, *Simultaneous Arithmetic Coding and Encryption Using Chaotic Maps*.

[5] Fatih Porikli, Faisal Bashir, and Huifang Sun, *Compressed Domain Video Object Segmentation.*

[6] Wang Wenda, Hao Yanling, *A Fast Wavelet Based Low Memory Embedded Block Coding Algorithm*, College of Automation, Harbin Engineering University, Harbin, Heilongjiang, 150001, China. IEEE computer society, Pp.615-619.

[7] Joachim Dahl, Jan Østergaard, Tobias Lindstrøm Jensen, Søren Holdt Jensen, *Compression of Image Sequences using the Structural Similarity Index Measure*, Department of Electronic Systems Aalborg University Aalborg, Denmark., Data compression conference.pp.133-142, 2009.

[8] Yassin M. Y. Hassan, Mohammed F. A Ahmed and Tarik K. Abdulhamid, *Image adaptive selective encryption of vector quantization index compression.* pp.1277-1280 ,ICIP 2009.

[9 Siu-Kei Au Yeung, Shuyuan Zhu, and Bing Zeng, *Partial video encryption based on alternating transforms*, IEEE SIGNAL PROCESSING LETTERS, VOL. 16, NO. 10, pp.893-896 OCTOBER 2009

[10] By Steven W. Smith, Ph.D. *The Scientist and Engineer's Guide to Digital Signal Processing,* http://www.dspguide.com/, November 20, 2010.

[11] Jeff Gilchrist, "Archive comparison Test", http://compression.ca, August 10, 2007

[12] The file extensions resource, http://www.fileinfo.com/extension/mng , Dec. 25, 2010.

[13] Introduction to JPEG, http://www.faqs.org/faqs/compression-faq/part2/section-6.html, March 20, 2011.

[14] The file extensions resource, http://www.fileinfo.com/, Dec. 25, 2010

[15] CS MSU Graphics & Media Lab Video Group, "Video codecs comparison test", MOSCOW, 15 MAY 2003, parts 1-4.

[16] Lossy Audio format comparison, http://www.bobulous.org.uk/misc/lossy_audio_2006.html, November 13, 2011.

[17] Audio format comparison, http://www.icydog.net/audioformats, November 15, 2010.

[18] By Scot Hacker, MP3: the definitive guide; Chapter 2-How MP3 works: inside the codec, http://oreilly.com/catalog/mp3/chapter/ch02.html, November 15, 2010.

[19] Advanced Audio codec (AAC), http://www.althos.com/tutorial/MPEG-tutorial-advanced-audio-codec-aac.html, March 20, 2011.

[20] By Mark Pilgrim, A gentle introduction to video encoding, part 3: lossy audio codecs, http://diveintomark.org/archives/2008/12/30/give-part-3-lossy-audio-codecs#aac , December 20, 2010

[21] By Jeff, Introduction to the Dolby Digital Format, http://en.kioskea.net/contents/audio/dolby-digital-ac3.php3, January 20, 2011.

[22] Adaptive Multi-Rate audio codec,

http://en.wikipedia.org/wiki/Adaptive_Multi-Rate_audio_codec, January 20, 2011.

[23] By Ian, Difference between MP3 and M4A, http://www.differencebetween.net/technology/difference-between-mp3-and-m4a/, January 20, 2011.

[24] Openformats.org, Ogg Vorbis (OGG), http://www.openformats.org/en69, February 17, 2011.

[25] Windows Media Audio, http://en.wikipedia.org/wiki/Windows_Media_Audio, February 17, 2011.

[26] By R. Kayne, What is AIFF? http://www.wisegeek.com/what-is-aiff.htm, November 20, 2010.

[27] My Format Factory, http://www.myformatfactory.com, January 13, 2011.

[28]. Audio File Formats http://www.nch.com.au/acm/formats.html, November 24, 2010.

[29] Graphics Interchange Formats, http://en.wikipedia.org/wiki/Graphics_Interchange_Format, January 15, 2011.

[30] JPEG, http://www.tech-faq.com/jpeg.html, November 10, 2010.

[31] TIFF, http://www.ntchosting.com/multimedia/tiff-tagged-image-file-format.html, November 20, 2010.

[32] The TIFF file format, http://www.prepressure.com/library/file-formats/tiff, January 10, 2011.

[33] Bitmap (Microsoft Windows Bitmap), http://www.image-formats.com/bitmap-microsoft-windows-bitmap/, February 15, 2011.

[34] Bitmap, http://en.wikipedia.org/wiki/Bitmap, January 26, 2011.

[35] PNG (Portable Network Graphics), http://searchsoa.techtarget.com/definition/PNG, January 26, 2010.

[36] What is TGA? http://www.myformatfactory.com/TGA, January 13, 2011.

[37] What is PCX? http://www.coolutils.com/Formats/PCX, February 20, 2011.

[38] PSD, http://www.myformatfactory.com/PSD, January 13, 2011.

[39] EXR File Extension, http://www.fileinfo.com/extension/exr, Dec. 25, 2010

[40] What is SWF file extension? http://www.rersoft.com/swf-file-extension/, March 4, 2011.

[41] What is FLV file, http://www.rersoft.com/flv-video-file/, March 4, 2011.

[42] What is NTSC? http://www.wizbit.net/cd-dvd_production_faqs_what_is_NTSC.htm, March 20, 2011.

[43] What is PAL? http://www.wizbit.net/cd-dvd_production_faqs_what_is_PAL.htm, March 20, 2011.

[44] Sorenson Compressor, http://www.webopedia.com/TERM/S/Sorenson_compressor.html, March 20, 2011.

[45] By Davide Rocchesso, The sampling theorem, http://www.faqs.org/docs/sp/sp-11.html, August 8, 2010.

[46] By Gabriel Torres, How Analog-to-Digital Converter (ADC) Works, http://www.hardwaresecrets.com/article/How-Analog-to-Digital-Converter-ADC-Works/317/2, August 7, 2010.

[47] By Bill Poser, Audio Data, http://billposer.org/Linguistics/Computation/LectureNotes/AudioData.html, September 16, 2010.

[48] By Erika Peterson, What is Digital Imaging?, http://www.wisegeek.com/what-is-digital-imaging.htm, August 27, 2010.

[49] Digital Media Converter 3.27, http://www.deskshare.com/audio-video-converter.aspx, November 3, 2010.

[50] Switch audio converter, http://www.nch.com.au/switch/index.html, November 8, 2010.

[51] Monkey Ape, http://www.monkeysaudio.com, November 8, 2010.

[52] LZOP103w, www.lzop.org, November 10, 2010.

[53] WinRar, http://www.rarlab.com, November 10.2010.

[54] WinZip, http: //www.winzip.com/downwz.htm, November 12, 2010.

[55] WinZip command line, http://www.winzip.com/downcl.htm, November 12, 2010.

[56] Image converter plus, http://www.imageconverterplus.com/download, November 19, 2010.

[57] StuffIT image converter, http://www.stuffit.com/win-deluxe.html, November 10, 2010.

Appendix A

Audio Music and Speech Comparison test

This section describes the legend and parameters of the table of data for Music Audio and Speech Audio tests in section 5.4.1 and section 5.4.2, respectively. The table index is given in the first column, header 'No'. The 'Type' column tags whether the compression used in the codecs is lossy or lossless. The 'Parameter' column specifies the values of several configurable settings during the operation performed by the codecs. As can be seen from Table A-1 and Table A-2, lossy audio specific codecs (those that are developed specifically for audio data) numbered 6-29 and 32 are parameterized by three variables. The first value, given in kbps, is the bitrate measured in kilo-bits-per-second. The bitrate is the number of bits that represents one second of the audio. The second parameter is the channel number; stereo is dual channel and mono is a single channel audio. The third parameter is the sample rate that represents the number of samples of audio taken in one second. The lossless audio codecs 30, 31, 33 and 34 parameterized by similar variables except in the first one, which is bit length, instead of bitrate for lossy audio specific codecs. Being lossless the bit length is the same as the test file, which is 16 bits. This means that, each sample is represented by a 16 bit variable. Monkey Audio (No1), is a lossless compressor for audio data. Even thought the same parameter is preserved in the output file as the lossless codecs mentioned above, the program were executed with extra high compression, considering the importance of network bandwidth efficiency. The tools numbered 2-5 are general archiving programs executed with no parameter settings; they do not have audio specific parameter settings. As for the codecs listed in the 'codecs' column, some of them are basically the same but are supposed for a target hardware plate form. For example, mp3 is available in several forms such as for PC stand alone playing, for Apple devices, for blackberry devices etc.

Table A-1: Music Audio Test data table

No	Type	Codec	Parameters	Compression time (Sec)	Decompression time (MB)	Output size (MB)	Compression ratio	Reconstruction Error
1	LL, CL	Monkey 4.06 [51]	Extra high	2.5	2.7	11	43.65	0.00
2	LL, CL	LZOP103w [52]	None	1.25	0.688	29.4	116.67	0.00
3	LL, CL	StafffT 14.0.0.18 [57]	None	8.61	8.31	16.3	64.68	0.00
4	LL, CL	WAR3.5 [53]	None	13.82	1.21	15.4	61.11	0.00
5	LL, CL	ZIP15.0 [54][55]	None	11.4	3.32	25.4	100.79	0.00
6	L,DMC,GUI	MP3	128kbps,stereo, 44.1khz	20.09	16.79	2.29	9.09	0.36
7	L,DMC,GUI	AAC	128kbps,stereo, 44.1khz	23.37	4.1	2.17	8.61	0.04
8	L,DMC,GUI	AC3(Dolby Digital)	128kbps,stereo, 44.1khz	10.83	4.91	2.28	9.05	0.39
9	L,DMC,GUI	AMR-NB	12.2kbps, mono, 8000HZ	10.4	5.81	0.234	0.93	0.32
10	L,DMC,GUI	AMR-WB	23.85kbps, mono, 16000HZ	20.59	8.49	0.446	1.77	0.34
11	L,DMC,GUI	M4A	128kbps, stereo, 44.1khz	25.74	5.13	2.18	8.65	0.04
12	L,DMC,GUI	OGG-Vorbis	128kbps, stereo, 44.1khz	117.16	4.97	2.61	10.36	0.36
13	L,DMC,GUI	Windows Media Audio 9	128kbps,stereo,44khz	16.2	4.76	2.32	9.21	0.37
14	L,DMC,GUI	WMA(for desktop)	128kbps,stereo, 48khz	18.85	6.79	2.32	9.21	0.35
15	L,DMC,GUI	WMA(for internet streaming)	64kbps,stereo, 48khz	17.97	7.11	1.17	4.64	0.35
16	L,DMC,GUI	WMA(for progressive download)	64kbps,stereo, 48khz	17.2	6.74	1.17	4.64	0.35
17	L,DMC,GUI	WMA (for portable devices)	64kbps,stereo, 48khz	17.86	7.03	1.17	4.64	0.35
18	L,DMC,GUI	Apple MPEG 4 AAC	128kbps,stereo, 44.1khz	26.14	4.78	2.18	8.65	0.04
19	L,DMC,GUI	Blackberry Audio MP3	128kbps,stereo, 44.1khz	21.76	19.64	2.28	9.05	0.36
20	L,DMC,GUI	Cell Phone Audio MP3	128kbps,stereo, 44.1khz	21.47	17.74	2.28	9.05	0.36
21	L,DMC,GUI	Mobile WMA Profiles WMA9.2	128kbps,stereo,48khz	19.3	7.19	2.32	9.21	0.35
22	L,DMC,GUI	Microsoft Zune Audio WMA9.2	128kbps,stereo, 48khz	19.19	6.98	2.32	9.21	0.35

No	Type	Codec	Parameters	Compression time (Sec)	Decompression time (MB)	Output size (MB)	Compression ratio	Reconstruction Error
23	L,DMC,GUI	Microsoft xbox Audio MP3	128kbps,stereo, 44.1khz	21.47	18.66	2.28	9.05	0.36
24	L,DMC,GUI	Creative Zen or MuVo Audio wma9.0	128kbps,stereo, 44.1khz	18.84	7.39	2.32	9.21	0.35
25	L,DMC,GUI	Archos Audio MP3	128kbps,stereo, 44.1khz	22.3	19.5	2.28	9.05	0.36
26	L,DMC,GUI	iRiver Audio mp3	128kbps,stereo, 44.1khz	21.53	18.78	2.28	9.05	0.36
27	L,DMC,GUI	Cowon MP3	128kbps,stereo, 44.1khz	21.49	19.47	2.28	9.05	0.36
28	L,DMC,GUI	Epson AAC Audio AAC	128kbps,stereo, 44.1khz	25.96	4.51	2.17	8.61	0.04
29	L,DMC,GUI	Epson mp3 audio	128kbps,stereo, 44.1khz	21.26	18.8	2.28	9.05	0.36
30	LL,Sw,GUI	au	16bits,stereo, 44.1khz	1.97	1.69	25.2	100.00	0.00
31	LL,Sw,GUI	Aiff	16bits,stereo, 44.1khz	1.63	1.57	25.2	100.00	0.00
32	LL,Sw,GUI	Aac	128kbps,stereo, 44.1khz	24.95	4.45	2.17	8.61	0.04
33	LL,Sw,GUI	Flac (fastest)	16bits,stereo, 44.1khz	3	3.25	21.2	84.13	0.00
34	LL,Sw,GUI	Flac (best)	16bits,stereo, 44.1khz	13.7	3.21	10.2	40.48	0.00

Table A-2: Speech Audio Test data table

no	Type	codec	parameters	compression time (Sec)	decompression time (Sec)	output size (MB)	compression ratio	Reconstruction Error
1	LL, CL	Monkey 4.06 [51]	Extra high	2.5	2.7	11	43.65	0.000
2	LL, CL	LZOP103w [52]	None	1.25	0.688	29.4	116.67	0.000
3	LL, CL	StafflT 14.0.0.18 [57]	None	8.61	8.31	16.3	64.68	0.000

no	Type	codec	parameters	compression time (Sec)	decompression time (Sec)	output size (MB)	compression ratio	Reconstruction Error
4	LL, CL	WAR3.5 [53]	None	13.82	1.21	15.4	61.11	0.000
5	LL, CL	ZIP15.0 [54][55]	None	11.4	3.32	25.4	100.79	0.000
6	L,DMC,GUI	MP3	128Kbps,stereo, 44.1KHZ	25.84	15.83	2.29	9.09	0.099
7	L,DMC,GUI	AAC	128Kbps,stereo, 44.1KHZ	23.87	4.17	2.28	9.05	0.009
8	L,DMC,GUI	AC3(Dolby Digital)	128Kbps,stereo, 44.1KHZ	10.12	4.3	2.28	9.05	0.103
9	L,DMC,GUI	AMR-NB	12.2Kbps, mono, 8000HZ	10.14	5.16	0.234	0.93	0.096
10	L,DMC,GUI	AMR-WB	23.85Kbps, mono, 16000HZ	20.3	8.62	0.446	1.77	0.099
11	L,DMC,GUI	M4A	128kbps, stereo, 44.1KHZ	25.16	5.2	2.33	9.25	0.009
12	L,DMC,GUI	OGG-Vorbis	128kbps, stereo, 44.1KHZ	113.27	4.84	2.53	10.04	0.144
13	L,DMC,GUI	Windows Media Audio 9	128kbps,stereo,44KHZ	13.76	4.94	2.32	9.21	0.102
14	L,DMC,GUI	WMA(for desktop)	128kbps,stereo, 48KHZ	19.94	7.34	2.32	9.21	0.102
15	L,DMC,GUI	WMA(for internet streaming)	64kbps,stereo, 48KHZ	18.89	6.93	1.17	4.64	0.103
16	L,DMC,GUI	WMA(for progressive download)	64kbps,stereo, 48KHZ	18.83	7.24	1.17	4.64	0.103
17	L,DMC,GUI	WMA (for portable devices)	64kbps,stereo, 48KHZ	18.97	7.27	1.17	4.64	0.103
18	L,DMC,GUI	Apple MPEG 4 AAC	128kbps,stereo, 44.1KHZ	27	4.22	2.33	9.25	0.009
19	L,DMC,GUI	Blackberry Audio MP3	128kbps,stereo, 44.1KHZ	27.43	20.05	2.28	9.05	0.099
20	L,DMC,GUI	Cell Phone Audio MP3	128kbps,stereo, 44.1KHZ	27.33	18.3	2.28	9.05	0.099
21	L,DMC,GUI	Mobile WMA Profiles WMA9.2	128kbps,stereo,48KHZ	20.11	7.7	2.32	9.21	0.103
22	L,DMC,GUI	Microsoft Zune Audio WMA9.2	128kbps,stereo, 48khz	20.05	7.68	2.32	9.21	0.103
23	L,DMC,GUI	Microsoft xbox Audio MP3	128kbps,stereo, 44.1KHZ	27.51	18.48	2.28	9.05	0.099
24	L,DMC,GUI	Creative Zen or MuVo Audio wma9.0	128kbps,stereo, 44.1KHZ	20.01	7.54	2.32	9.21	0.103
25	L,DMC,GUI	Archos Audio MP3	128kbps,stereo, 44.1KHZ	27.37	18.66	2.28	9.05	0.099
26	L,DMC,GUI	iRiver Audio mp3	128kbps,stereo, 44.1KHZ	27.46	19.14	2.28	9.05	0.099
27	L,DMC,GUI	Cowon MP3	128kbps,stereo, 44.1KHZ	27.6	19.21	2.28	9.05	0.099
28	L,DMC,GUI	Epson AAC Audio	128kbps,stereo, 44.1KHZ	26.93	5.15	2.28	9.05	0.009

no	Type	codec	parameters	compression time (Sec)	decompression time (Sec)	output size (MB)	compression ratio	Reconstruction Error
29	L,DMC,GUI	Epson mp3 audio	128kbps,stereo, 44.1KHZ	27.66	19	2.28	9.05	0.099
30	LL,Sw,GUI	au	16bits,stereo44.1khz	1.73	1.46	25.2	100.00	0.000
31	LL,Sw,GUI	Aiff	16bits, stereo,44.1khz	1.43	1.04	25.2	100.00	0.000
32	LL,Sw,GUI	Aac	128kbps,stereo,44.1khz	25.79	4.68	2.28	9.05	0.008
33	LL,Sw,GUI	Flac (fastest)	16bits, stereo,44.1khz	2.72	3.24	15.5	61.51	0.000
34	LL,Sw,GUI	Flac (best)	16bits, stereo,44.1kbps	16.11	4.03	10.8	42.86	0.000

Legend:

LL-Lossless

L-Lossy

GUI-Graphical User Interface programs

CL-Command line tools are available and used

DMC=Digital Media Convertor (version 3.27) [49]

Sw=Switch audio converter (version 1.29) [50]

Appendix B

Image Comparison Test

Table B-1 shows the data for the Image Comparison Test done in section 5.4.3. The 'Type' column tags whether the codecs are lossy or lossless. The 'codecs' column lists the various codecs tested in the experiment. The 'parameter' column specifies the settings used for each test. As can be seen in the table, all settings are not applicable for all the codecs. For most of the codecs, several combinations of settings have been intuitively selected and used. Command line tools are used for all the tests in this case. The product used for this test is Image Converter Plus, version V8.0.181, [56].

Table B-1: Image Test data table

no	type	codec	parameters	compression time (Sec)	decompression time (Sec)	output size (KB)	compression ratio	Reconstruction Error
1	LL,CL	Monkey	not applicable					
2	LL,CL	LZOP	none	2.61	0.5	458.35	97.77	0.00
3	LL,CL	StaffIT	none	3.11	2.16	3.97	0.85	0.00
4	LL,CL	WAR	none	15.92	5	227.08	48.44	0.00
5	LL,CL	ZIP	none	7.67	2.08	400.00	85.32	0.00
6	L,ICP,GUI	JPEG	Q=1,inter=Y,Op_Huff	7	4	3.15	0.67	91.47
7	L,ICP,GUI	JPEG	Q=25,inter=Y,Op_Huff	8	5	13.60	2.90	48.22
8	L,ICP,GUI	JPEG	Q=50,inter=Y,Op_Huff	8	5	22.57	4.81	42.37
9	L,ICP,GUI	JPEG	Q=75,inter=Y,Op_Huff	8	5	36.54	7.80	36.58
10	L,ICP,GUI	JPEG	Q=100,inter=Y,Op_Huff	11	9	228.99	48.85	4.83
11	L,ICP,GUI	JPEG	Q=1,inter=N,Op_Huff	8	5	3.17	0.68	91.47
12	L,ICP,GUI	JPEG	Q=25,inter=N,Op_Huff	8	5	13.63	2.91	48.22
13	L,ICP,GUI	JPEG	Q=50,inter=N,Op_Huff	8	6	22.58	4.82	42.37
14	L,ICP,GUI	JPEG	Q=75,inter=N,Op_Huff	8	6	36.56	7.80	36.58
15	L,ICP,GUI	JPEG	Q=100,inter=N,Op_Huff	11	9	229.00	48.85	4.83
16	L,ICP,GUI	JPEG	Q=1,inter=Y,WO_Op_Huff	5	5	5.81	1.24	91.47
17	L,ICP,GUI	JPEG	Q=25,inter=Y,WO_Op_Huff	6	4	14.98	3.20	48.22
18	L,ICP,GUI	JPEG	Q=50,inter=Y,WO_Op_Huff	6	5	23.26	4.96	42.37
19	L,ICP,GUI	JPEG	Q=75,inter=Y,WO_Op_Huff	6	5	37.16	7.93	36.58
20	L,ICP,GUI	JPEG	Q=100,inter=Y,WO_Op_Huff	8	9	243.82	52.01	4.83
21	L,ICP,GUI	GIF	Color=255,dither=Y,Op_LL=N	44	4	129.89	27.71	26.92
22	L,ICP,GUI	GIF	Color=255,dither=N,Op_LL=N	45	5	119.78	25.55	22.74
23	L,ICP,GUI	GIF	Color=255,dither=Y,Op_LL=Y	47	5	129.89	27.71	26.92

no	type	codec	parameters	compression time (Sec)	decompression time (Sec)	output size (KB)	compression ratio	Reconstruction Error
24	L,ICP,GUI	GIF	Color=255,dither=N,Op_LL=Y	44	4	119.78	25.55	22.74
25	LL,ICP,GUI	PNG	CompL=5,dither=Y	13	6	283.92	60.56	0.00
26	LL,ICP,GUI	PNG	CompL=0,dither=Y	13	6	283.92	60.56	0.00
27	LL,ICP,GUI	PNG	CompL=9,dither=Y	18	6	281.68	60.09	0.00
28	LL,ICP,GUI	PNG	CompL=5,dither=N	13	6	283.92	60.56	0.00
29	LL,ICP,GUI	PNG	CompL=0,dither=N	13	6	283.92	60.56	0.00
30	LL,ICP,GUI	PNG	CompL=9,dither=N	18	6	281.68	60.09	0.00
31	LL,ICP,GUI	TGA	RLE=N,dither=Y	4	4	469.31	100.11	0.00
32	LL,ICP,GUI	TGA	RLE=Y,dither=Y	5	4	469.45	100.14	0.00
33	LL,ICP,GUI	TGA	RLE=N,dither=N	5	5	469.31	100.11	0.00
34	LL,ICP,GUI	TGA	RLE=Y,dither=N	5	4	469.45	100.14	0.00
35	LL,ICP,GUI	TIFF	compT=LZW,CompL=0,dither=Y	16	12	497.10	106.04	0.00
36	LL,ICP,GUI	TIFF	compT=LZW,CompL=5,dither=Y	16	10	497.10	106.04	0.00
37	LL,ICP,GUI	TIFF	compT=LZW,CompL=9,dither=Y	16	9	497.10	106.04	0.00
38	LL,ICP,GUI	TIFF	CompT=none,dither=Y	5	4	469.52	100.15	0.00
39	LL,ICP,GUI	TIFF	CompT=flate,dither=Y	9	6	413.14	88.13	0.00
40	L,ICP,GUI	TIFF	CompT=jpeg,dither=Y	7	7	57.08	12.18	31.28
41	L,ICP,GUI	TIFF	CompT=jbig,dither=Y	40	4	11.55	2.46	130.50
42	LL,ICP,GUI	TIFF	CompT=packbits,dither=Y	6	11	473.43	100.99	0.00
43	LL,ICP,GUI	TIFF	compT=LZW,CompL=0,dither=N	15	6	497.10	106.04	0.00
44	LL,ICP,GUI	TIFF	compT=LZW,CompL=5,dither=N	14	6	497.10	106.04	0.00
45	LL,ICP,GUI	TIFF	compT=LZW,CompL=9,dither=N	15	11	497.10	106.04	0.00
46	LL,ICP,GUI	TIFF	CompT=none,dither=N	5	7	469.52	100.15	0.00
47	LL,ICP,GUI	TIFF	CompT=flate,dither=N	8	6	413.14	88.13	0.00

no	type	codec	parameters	compression time (Sec)	decompression time (Sec)	output size (KB)	compression ratio	Reconstruction Error
48	L,ICP,GUI	TIFF	CompT=jpeg,dither=N	7	8	57.08	12.18	31.28
49	L,ICP,GUI	TIFF	CompT=jbig,dither=N	39	4	4.28	0.91	115.54
50	L,ICP,GUI	TIFF	CompT=packbits,dither=N	6	4	473.43	100.99	115.54
51		EPS	none	28	0	938.76	200.25	0.00
52	L,ICP,GUI	EXR	CompT=RLE	10	8	811.10	173.01	7.11
53	L,ICP,GUI	EXR	compT=none	8	8	1256.57	268.04	7.11
54	L,ICP,GUI	EXR	compT=zips	41	11	577.94	123.28	7.11
55	L,ICP,GUI	EXR	compT=zip	29	10	510.48	108.89	7.11
56	L,ICP,GUI	EXR	compT=piz	15	11	316.55	67.52	7.11
57	L,ICP,GUI	EXR	compT=pxr24	29	8	543.44	115.92	7.11
58	LL,ICP,GUI	MNG	compL=5,dither=Y	13	6	283.97	60.57	0.00
59	LL,ICP,GUI	MNG	compL=0,dither=Y	13	5	283.97	60.57	0.00
60	LL,ICP,GUI	MNG	compL=9,dither=Y	18	6	281.74	60.10	0.00
61	LL,ICP,GUI	MNG	compL=5,dither=N	13	6	283.97	60.57	0.00
62	LL,ICP,GUI	MNG	compL=0,dither=N	13	6	283.97	60.57	0.00
63	LL,ICP,GUI	MNG	compL=9,dither=N	18	6	281.74	60.10	0.00
64	LL,ICP,GUI	PCX	dither=Y	4	13	487.71	104.03	0.00
65	LL,ICP,GUI	PCX	dither=N	5	14	487.71	104.03	0.00
66	L,ICP,GUI	PDF	compT=lzw,pdfa=N	9	9	497.13	106.04	38.41
67	L,ICP,GUI	PDF	compT=flate,pdfa=N	11	9	400.19	85.36	38.41
68	L,ICP,GUI	PDF	compT=ccitt_fax,pdfa=N	8	8	55.43	11.82	123.78
69	L,ICP,GUI	PDF	compT=lzw,pdfa=Y	11	8	404.32	86.25	38.41
70	L,ICP,GUI	PDF	compT=flate,pdfa=Y	11	8	404.32	86.25	38.41
71	L,ICP,GUI	PDF	compT=ccitt_fax,pdfa=Y	12	8	404.32	86.25	38.41

no	type	codec	parameters	compression time (Sec)	decompression time (Sec)	output size (KB)	compression ratio	Reconstruction Error
72	LL,ICP,GUI	PSD	RLE=Y	4	8	468.96	100.03	0.00
73	LL,ICP,GUI	PSD	RLE=N	4	8	468.96	100.03	0.00

Legend:

LL=Lossless

L=Lossy

Y=Yes

N=No

Q=Quality

Inter=interleaved

Op_Huff=Optimized Huffman encoding is used.

WO_Op_Huff= without Optimized Huffman.

Op_LL=Optimal Lossless compression

CompL=Compression Level

CompT=Compression Type

RLE=Run Length Encoding

PDFA= PDF Level A compliance standard

GUI=Graphical User Interface

CL=Command Line Tool.

ICP=Image Converter Plus 8.0.18 [56].

Appendix C

Video Comparison Test

Table C-1 shows the data for the Video Comparison Test in section 5.4.4. The 'No' column is the index into the table. The 'Type' column tags whether the codecs are lossy or lossless. The 'File Format' column lists the file format used to contain the combined video and audio data. The 'Video codec' column lists the video specific codes used in the experiment that applies to the frames (also known as moving pictures). The 'audio codec' column lists the codecs used to encode the audio data in the output video files. Similar configuration settings have been used for both codecs in the test except where these settings are no supported. For video codecs, the settings are 24 frames per second, 512 kbps bitrate, and the original dimensions of the frames have been preserved (624X352 pixels). For audio codecs, the settings are 128kbps, stereo channel and 44.1 kHz sample rate. The exception for video codec is H.263 in all the file formats it has been used. It only supports preset dimensions for the frames (the default value of 352 X 288 pixels has been used). The exception for the audio codec is the AMR codec where the default values of 12.2 kbps bitrate and 8000 kHz of bitrate has been used.

Table C-1: Video Test data table

No	Type	Format	Video codec	Audio codec	Compression time (Sec)	Output size (MB)	Compression ratio	Reconstruction Error
1	LL,CL	--	Monkey 4.06 [51]	NA	1.49	26.19	99.72	0
2	LL,CL	--	LZOP103w [52]	NA	23.77	26.13	99.48	0
3	LL,CL	--	StaffIT 14.0.0.18 [57]	NA	31.40	26.15	99.54	0
4	LL,CL	--	WAR3.5 [53]	NA	6.55	26.12	99.42	0
5	L,DMC,GUI	3GP	MPEG 4	AMR-NB	117.22	8.46	32.21	7.76
6	L,DMC,GUI		H.263	AMR-NB	110.82	8.14	31.00	11.50
7	L,DMC,GUI		H.264	AMR-NB	186.92	12.95	49.30	6.62
8	L,DMC,GUI		XviD	AMR-NB	247.85	8.20	31.23	9.33
9	L,DMC,GUI		MPEG 4	MP4	125.64	10.13	38.58	7.76
10	L,DMC,GUI		H.263	MP4	120.48	9.82	37.37	11.50
11	L,DMC,GUI		H.264	MP4	211.17	14.64	55.74	6.62
12	L,DMC,GUI		XviD	MP4	268.76	9.87	37.57	9.33
13	L,DMC,GUI	3G2	MPEG 4	AMR-NB	118.02	8.46	32.21	10.36
14	L,DMC,GUI		H.263	AMR-NB	111.87	8.14	31.00	12.17
15	L,DMC,GUI		H.264	AMR-NB	199.56	12.99	49.45	6.59
16	L,DMC,GUI		XviD	AMR-NB	257.12	8.20	31.23	10.90
17	L,DMC,GUI		MPEG 4	MP4	125.94	10.13	38.58	10.36
18	L,DMC,GUI		H.263	MP4	120.50	9.82	37.37	12.17
19	L,DMC,GUI		H.264	MP4	220.50	14.64	55.74	6.62
20	L,DMC,GUI		XviD	MP4	303.52	9.87	37.57	10.90

No	Type	Format	Video codec	Audio codec	Compression time (Sec)	Output size (MB)	Compression ratio	Reconstruction Error
21	L,DMC,GUI	AVI	MPEG 4	MP3	127.67	10.27	39.08	10.35
22	L,DMC,GUI		Microsft MPEG-4 Video Codec V2	MP3	129.96	13.98	53.24	10.34
23	L,DMC,GUI		DivX 3 (MS-MPEG-4 V3 codec)	MP3	129.76	13.83	52.64	10.39
24	L,DMC,GUI		XviD Encoder	MP3	322.42	9.99	38.05	10.77
25	L,DMC,GUI		MJPEG compressor	MP3	137.17	32.65	124.30	10.27
26	L,DMC,GUI		H.263	MP3	113.43	9.94	37.85	12.15
27	L,DMC,GUI		H.264	MP3	222.59	14.79	56.32	10.07
28	L,DMC,GUI		Huffman Lossless Compressor	MP3	96.69	920.37	3503.80	10.51
29	L,DMC,GUI	DV-AVI Type 1	NTSC	PCM	84.27	413.43	1573.92	10.24
30	L,DMC,GUI	DV-AVI Type 1	PAL	PCM	97.29	413.88	1575.63	10.23
31	L,DMC,GUI	DV-AVI Type 2	NTSC	PCM	81.96	351.14	1336.78	10.24
32	L,DMC,GUI	DV-AVI Type 2	PAL	PCM	96.38	417.28	1588.57	10.16
33	L,DMC,GUI	FLV	FFmpeg Flash Video (FLV)	MP3	123.20	10.60	40.34	4.50
34	L,DMC,GUI	MPEG 1 Custom	MPEG-1 Video	MP2	117.62	10.44	39.74	12.17
35	L,DMC,GUI	MPEG 1 VCD (NTSC)	MPEG -1 Video	MP2	74.31	19.94	75.90	12.04
36	L,DMC,GUI	MPEG 1 VCD (PAL)	MPEG -1 Video	MP2	73.84	19.92	75.85	11.99
37	L,DMC,GUI	MPEG 2 Custem	MPEG 2 Video	MP2	115.78	10.53	40.07	10.75
38	L,DMC,GUI	MPEG 2 SVCD (NTSC)	MPEG 2 Video	MP2	196.83	33.17	126.28	12.26
39	L,DMC,GUI	MPEG 2 SVCD (PAL)	MPEG 2 Video	MP2	207.38	33.13	126.12	12.09
40	L,DMC,GUI	MPEG 2 DVD (NTSC)	MPEG 2 Video	MP2	264.86	90.75	345.47	10.11
41	L,DMC,GUI	MPEG 2 DVD (PAL)	MPEG 2 Video	AC3	304.03	90.78	345.60	10.04
42	L,DMC,GUI	MPEG 4	MPEG-4	MP3	127.06	10.14	38.60	10.35
43	L,DMC,GUI		H.264	MP3	220.92	14.66	55.82	3.50

No	Type	Format	Video codec	Audio codec	Compression time (Sec)	Output size (MB)	Compression ratio	Reconstruction Error
44	L,DMC,GUI		XviD	MP3	320.62	9.87	37.59	10.90
45	L,DMC,GUI	Quick Time MOV	MPEG-4	MP4	125.66	11.86	45.16	10.26
46	L,DMC,GUI		XviD Encoder	MP4	244.15	9.03	34.36	10.34
47	L,DMC,GUI		H.263	MP4	121.80	11.56	44.03	10.34
48	L,DMC,GUI		H.264	MP4	211.34	14.65	55.76	6.60
49	L,DMC,GUI		Sorenson Video v1	MP4	687.66	84.41	321.35	10.30
50	L,DMC,GUI	SWF	FFmpeg Flash Video (FLV)	MP3	127.70	8.82	33.58	0.00
51	L,DMC,GUI	WMV Custom	WMV Video 9	WMA 9.2	81.97	9.50	36.15	10.92
52	L,DMC,GUI		WMV 10 Advanced Profile	WMA 9.2	83.40	9.50	36.15	10.92
53	L,DMC,GUI	WMV for Desktop	WMV	WMA	83.17	16.63	63.31	10.48
54	L,DMC,GUI	WMV for internet streaming	WMV	WMA	37.75	4.78	18.20	12.09
55	L,DMC,GUI	WMV for progressive download	WMV	WMA	40.27	9.47	36.07	12.03
56	L,DMC,GUI	WMV for Portable Devices	WMV	WMA	45.63	11.77	44.81	12.13

Legend:

LL-Lossless

L-Lossy

GUI-Graphical User Interface programs

CL-Command line tools are available and used

DMC=Digital Media Convertor (version 3.27) [49]

Appendix D

Bandwidth Test

The bandwidth test has been done by transferring each of the outputs from the video comparison test. Table D-1 shows the file size and the transmission time over an ad-hoc wireless network.

Table D-1: Bandwidth Test data table

No	File size in MB	Transmission time in seconds
1	26.19	10.33
2	26.13	10.22
3	26.15	10.23
4	26.12	10.20
5	8.46	4.24
6	8.14	3.23
7	12.95	5.13
8	8.20	3.21
9	10.13	3.97
10	9.82	3.88
11	14.64	5.73
12	9.87	3.91
13	8.46	3.34
14	8.14	3.24
15	12.99	5.09
16	8.20	3.23
17	10.13	3.98
18	9.82	3.89
19	14.64	5.75
20	9.87	3.91
21	10.27	4.06
22	13.98	5.62
23	13.83	5.45
24	9.99	3.95
25	32.65	12.80
26	9.94	3.92
27	14.79	5.80
28	920.37	359.05
29	413.43	161.89
30	413.88	161.75
31	351.14	137.38
32	417.28	164.33

No	File size in MB	Transmission time in seconds
33	10.60	4.20
34	10.44	4.15
35	19.94	7.82
36	19.92	7.89
37	10.53	4.22
38	33.17	13.03
39	33.13	13.09
40	90.75	35.47
41	90.78	35.56
42	10.14	4.13
43	14.66	5.84
44	9.87	3.97
45	11.86	4.70
46	9.03	4.05
47	11.56	4.63
48	14.65	5.87
49	84.41	33.20
50	8.82	3.63
51	9.50	3.73
52	9.50	3.75
53	16.63	6.58
54	4.78	1.92
55	9.47	3.86
56	11.77	4.71