AN EFFICIENT MODEL FOR QUALITY VIDEO COMPRESSION

A PROJECT REPORT

Submitted by

MANICHELLA.R (810015205039)

NANDHINI. S (810015205048)

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

ANNA UNIVERSITY, TIRUCHIRAPALLI – 620024

MARCH, 2019

UNIVERSITY COLLEGE OF ENGINEERING -BIT

CAMPUS

TIRUCHIRAPPALLI-620024

BONAFIDE CERTIFICATE

Certified that this project report "**AN EFFICIENT MODEL FOR QUALITY VIDEO COMPRESSION"** is the bonafide work of "**MANICHELLA R (810015205039)** and **NANDHINI S (810015205048)"** who carried out the project work under my supervision.

SIGNATURE

Dr.D.Venkatesan

HEAD OFTHEDEPARTMENT

Assistant Professor
Department of Information Technology
University College of Engineering,
Anna University-BIT Campus,
Tiruchirappalli -620 024.

SIGNATURE

Mrs.S.Usha

SUPERVISOR

Assistant Professor,
Dept. of Information Technology
University College of Engineering
Anna University- BIT Campus,
Tiruchirapalli. -620 024.

Submitted for the project Viva voce examination held on ……

InternalExaminer **ExternalExaminer**

DECLARATION

We hereby declare that the work entitled "**AN EFFICIENT MODEL FOR QUALITY VIDEO COMPRESSION** " is submitted in partial fulfilment of the requirement for the award of the degree in B.Tech, in University College of Engineering, BIT Campus, Tiruchirapalli. It is the record of our own work carried out during the academic year 2015-2019 under the supervision and guidance of **Mrs. S.USHA**, Department of Information Technology, University College of Engineering, BIT Campus, Tiruchirapalli. The extent and source of information are derived from the existing literature and have been indicate through the dissertation at the appropriate places.

MANICHELLA.R(8100150205039)

NANDHINI.S(810015205048)

I certify that the declaration made above by the candidate is true.

Signature of the Guide,

Mrs. S.USHA,
Assistant Professor,
Dept. of Information Technology,
Anna University,
Tiruchirapalli - 620024.

ACKNOWLEDGEMENT

We would like to thank our Honourable Dean **Dr.T.SENTHIL KUMAR**, Professor for having provided us with all required facilities to complete our project without hurdles.

We would also like to express our sincere thanks to honourable **Dr. D. VENKATESAN**, Assistant Professor, Head of the Department of Information Technology, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of this projectwork.

We would like to express our sincere thanks to the Project Co-ordinators **Mrs.V.M.PRIYADHARSHINI**, Assistant Professor, Department of Information Technology and **Mr.M.PRASANNAKUMAR**, Teaching Fellow, Department of Computer Science and Engineering for their kind support.

We would like to thank and express our deep sense of gratitude to our project guide **Mrs. S. USHA**, Assistant Professor, Department of Information Technology, University College of Engineering, BIT Campus, Tiruchirapalli, for her valuable guidance throughout the project.

We also extend our thanks to all other teaching and non-teaching staff for their encouragement and support.

We thank our beloved parents and friends for their full support in the moral development of this project.

TABLE OF CONTENTS

ABSTRACT

The huge usage of digital multimedia via communications, wireless communications, Internet, Intranet and cellular mobile leads to incurable growth of data flow through these media. The researchers go deep in developing efficient techniques in these fields such as compression of data, image and video. Recently, video compression techniques and their applications in many areas (educational, agriculture, medical …) cause this field to be one of the most interested fields. Wavelet transform is an efficient method that can be used to perform an efficient compression technique. In this proposed work takes video as input, from that given video is converted into snapshots or frames using Key Frame Technique. The converted frames into Motion Detection using background frame matching for pixel calculation. This method is very efficient method of comparing image pixel values in subsequent still frames captured after every 0.5 seconds from the video. Two frames are required to detect movement. First frame is called reference frame and the second frame, which is called the input frame contains the moving object. The two frames are compared and the differences in pixel values are determined. Finally Frame Difference Approach, wavelet transform is an efficient method that can be used to perform an efficient video compression technique. This work can be concentrated on the calculation of frame near distance (difference between frames). Histogram Matching and Discrete Cosine Transform are used for decompressing the video. The proposed work provides better video quality and storage reduction.

CHAPTER 1

INTRODUCTION

1.1 VIDEO COMPRESSION STANDARDS

During the '80s and '90s, Discrete Cosine Transform (DCT) based compression algorithms and international standards were developed to alleviate storage and bandwidth limitations imposed by digital still image and motion video applications. Today there are three DCT-based standards that are widely used and accepted worldwide: · JPEG (Joint Photographic Experts Group) · H.261 (Video codec for audiovisual services) · MPEG (Motion Picture Experts Group) Each of these standards is well suited for particular applications: JPEG for still image compression, H.261 for video conferencing, and MPEG for high-quality, multimedia systems.

1.2 VIDEO COMPRESSION TECHNOLOGY

Video compression technology is a set of techniques for reducing and removing redundancy in video data. The compressed video must have a much smaller size compared to the uncompressed video. This allows the video to be saved in a smaller file or sent over a network more quickly. The video compression efficiency is related to the video bitrate for a given resolution and frame rate. The compression is more efficient if it results in lower bitrates.Video compression may be lossy, in which case the image quality is reduced compared to the original image. For lossy compression, the goal is to develop compression techniques that are efficient and result in perceptually lossless quality. In effect, even though the compressed video is different from the original uncompressed video, the differences are not easily visible to the human eye.

Video data may be represented as a series of still frames, or fields for interlaced video. The sequence of frames will almost certainly contain both spatial and temporal redundancy that video compression algorithms can use. Most video compression algorithms use both spatial compression, based on redundancy within a single frame or field, and temporal compression, based on redundancy between different video frames.

1.2.1 SPATIAL COMPRESSION

Spatial compression techniques are based on still image compression. The most popular technique, which is adopted by many standards, is the transform technique. In this technique, the image is split into blocks and the transform is applied to each block. The result of the transform is scaled and quantized. The quantized data is compressed by a lossless entropy encoder and the output bitstream is formed from the result. The most popular transform algorithm is the Discrete Cosine Transform (DCT) or its modifications. There are many other algorithms for spatial compression such as wavelet transform, vector coding, fractal compression, etc.

1.2.2 TEMPORAL COMPRESSION

Temporal compression can be a very powerful method. It works by comparing different frames in the video to each other. If the video contains areas without motion, the system can issue a short command that copies that part of the previous frame, bit-for-bit, into the next one. If some of the pixels are changed (moved, rotated, change brightness, etc.) with respect to the reference frame or frames, then a prediction technique can be applied. For each area in the current frame, the algorithm searches for a similar area in the previous frame or frames. If a similar area is found, it’s subtracted from the current area and the difference is encoded by the transform coder. The reference for the current frame area may also be obtained as a weighted sum of corresponding areas from previous and consecutive frames. If consecutive frames are used, then the current frame must be delayed by some number of frame periods.

1.2.3 BLOCK TRANSFORMATION

The image compression techniques used in JPEG and in most video compression algorithms are "lossy." That is, the original uncompressed image can't be perfectly reconstructed from the compressed data, so some information from the original image is lost. Lossy compression algorithms attempt to ensure that the differences between the original uncompressed image and the reconstructed image are not perceptible to the human eye.

The first step in JPEG and similar image compression algorithms is to divide the image into small blocks and transform each block into a frequency-domain representation. Typically, this step uses a discrete cosine transform (DCT) on blocks that are eight pixels wide by eight pixels high. Thus, the DCT operates on 64 input pixels and yields 64 frequency-domain coefficients. The DCT itself preserves all of the information in the eight-by-eight image block. That is, an inverse DCT (IDCT) can be used to perfectly reconstruct the original 64 pixels from the DCT coefficients. However, the human eye is more sensitive to the information contained in DCT coefficients that represent low frequencies (corresponding to large features in the image) than to the information contained in DCT coefficients that represent high frequencies (corresponding to small features). Therefore, the DCT helps separate the more perceptually significant information from less perceptually significant information. Later steps in the compression algorithm encode the low-frequency DCT coefficients with high precision, but use fewer or no bits to encode the high-frequency coefficients, thus discarding information that is less perceptually significant. In the decoding algorithm, an IDCT transforms the imperfectly coded coefficients back into an 8x8 block of pixels.

The computations performed in the IDCT are nearly identical to those performed in the DCT, so these two functions have very similar processing requirements. A single two-dimensional eight-by-eight DCT or IDCT requires a few hundred instruction cycles on a typical DSP. However, video compression

algorithms must often perform a vast number of DCTs and/or IDCTs per second. For example, an MPEG-4 video decoder operating at CIF (352x288) resolution and a frame rate of 30 fps may need to perform as many as 71,280 IDCTs per second, depending on the video content. The IDCT function would require over 40 MHz on a Texas Instruments TMS320C55x DSP processor (without the DCT accelerator) under

1.3. VIDEO COMPRESSION PROCESSING FUNCTION

As mentioned earlier, the JPEG, H.261, and MPEG video compression standards are all based on the DCT. In addtion to being DCT-based, many processing functions and compression principles are common to these standards. The basic compression scheme for all three standards can be summarized as follows: divide the picture into 8x8 blocks, determine relevant picture information, discard redundant or insignificant information, and encode relevant picture information with the least number of bits. Common functions to all three standards are:

- DCT
- Zig-Zag Scanning
- Quantization
- Entropy Coding
- Motion Estimation

1.3.1 DCT & ZIG-ZAG SCANNING

The Discrete Cosine Transform is closely related to the Discrete Fourier Transform (FFT) and, as such, allows data to be represented in terms of its frequency components. Similarly, in image processing applications the two dimensional (2D) DCT maps a picture or a picture segment into its 2D frequency components. For video compression applications, since the variations in the block tend to be low, the great majority of these transformations result in a more compact representation of the block. The block energy is packed into the corresponding lower frequency bins.

The DCT component at coordinates (0,0) is referred to as the DC bin. All other components are referred to as AC bins.

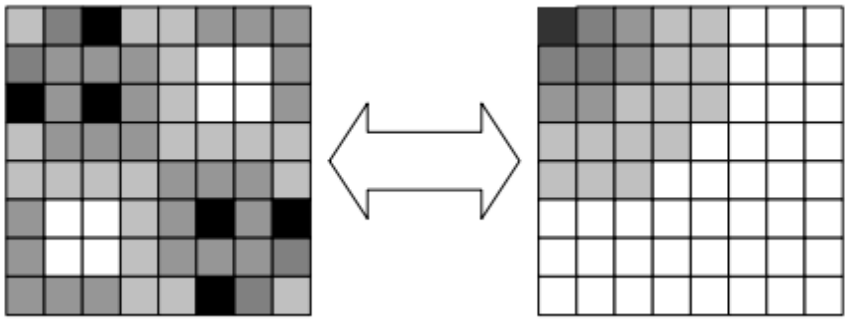


Figure 1.1 The DCT operation. (a) Original picture. (b) Corresponding DCT mapping of (a)

Since the mapping is from lower to higher frequencies in the horizontal and vertical directions, zig-zag scanning of the resulting 2D frequency bins clusters packets of picture information from low to high frequencies into a 1D stream of bins.

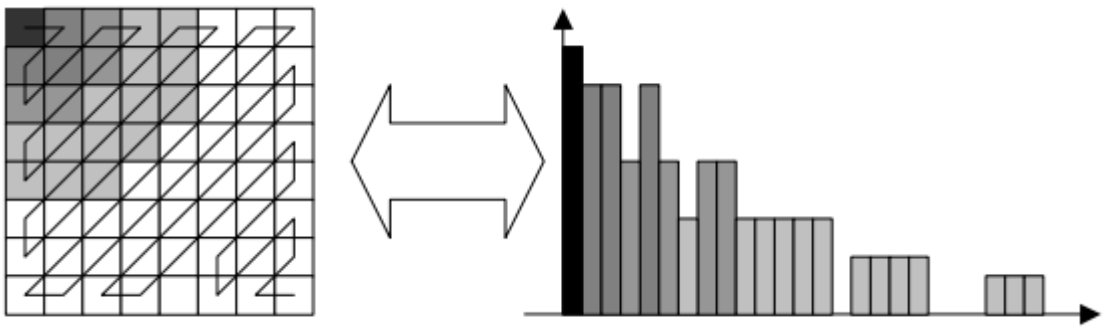


Figure 1.2 Zig-zag scanning

1.3.2 QUANTIZATION

Quantization is the primary source of data loss in DCT based image compression algorithms. Quantization reduces the amount of information required to represent the frequency bins by converting amplitudes that fall in certain ranges to one in a set of quantization levels. For simplicity, all the standard image compression algorithms use linear quantization where the step size quantization levels are constant. Quantization in the frequency domain has many advantages over directly quantizing the pixel values. Quantization of the pixel values results in a visual artifact called "contour" distortion where small changes in amplitude in a gradient area cause step-sized changes in the reconstructed amplitude. Except for the DC bin, quantization error for each of the frequency bins average out to zero over the 8 x 8 block.

1.3.3 ENTROPY CODING

Entropy coding is a loss-less compression scheme based on statistical properties of the picture or the stream of information to be compressed. Although entropy coding is implemented slightly different in each of the standards, the basic “entropy coding” scheme consists of encoding the most frequently occurring patterns with the least number of bits. In, this manner, data can be compressed by an additional factor of 3 or 4.

Entropy coding for video compression applications is a two step process: Zero Run-Length Coding (RLC) and Huffman coding. RLC data is an intermediate symbolic representation of the quantized bins which utilizes a pair of numbers. The first number represents the number of consecutive zeros while the second number represents the value between zero-run lengths. For instance the RLC code (5,8) represents the sequence (0,0,0,0,0,8) of numbers. Huffman coding assigns a variable length code to the RLC data, producing variable length bitstream data. This requires Huffman tables which can be pre-computed based on statistical properties of the image (as it is in JPEG) or can be pre-determined if a default table is to be used (as it is in H.261 and MPEG). In either case, the same table is used to decode the bitstream data. As mentioned

above, frequently occurring RLC patterns are coded with the least number of bits. At this point the digital stream, which is a representation of the picture, has no specific boundaries or fixed length. This information can now be stored or appropriately prepared for transmission.

1.3.4 MOTION ESTIMATION

In general, successive pictures in a motion video sequence tend to be highly correlated, that is, the pictures change slightly over a small period of time. This implies that the arithmetic difference between these pictures is small. For this reason, compression ratios for motion video sequences may be increased by encoding the arithmetic difference between two or more successive frames. In contrast, objects that are in motion increase the arithmetic difference between frames which in turn implies that more bits are required to encode the sequence. To address this issue, motion estimation is utilized to determine the displacement of an object Motion estimation is the process by which elements in a picture are best correlated to elements in other pictures (ahead or behind) by the estimated amount of motion. The amount of motion is encapsulated in the motion vector. Forward motion vectors refer to correlation with previous pictures. Backward motion vectors refer to correlation with future pictures.

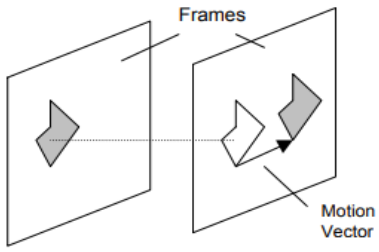


Figure 1.3 Motion Estimation.

An efficient motion estimation algorithm increases frame correlation, which in turn minimizes pixel arithmetic difference. Resulting in not only higher compression ratios but also in higher quality decoded video sequences. Motion estimation is an extremely computationally intensive operation difficult to implement in real-time. For this reason, a variety of motion estimation algorithms have been implemented by the industry.

1.4 THE JPEG COMPRESSION ALGORITHM

The JPEG algorithm was designed to efficiently compress continuous-tone still images. In addition to its use in still image compression, JPEG has also been adapted for use with motion video sequences. This adaptation (commonly called motion-JPEG) uses the JPEG algorithm to individually compress each frame in a motion video sequence.

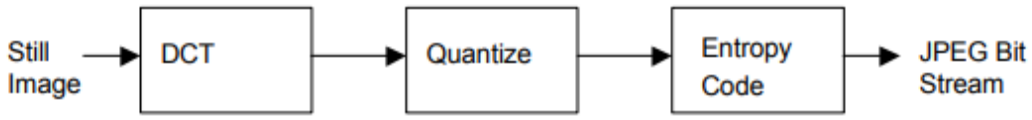


Figure 1.4 JPEG Encoding

Each color component of a still image is treated as a separate gray-scale picture by JPEG. Although JPEG allows any color component separation, images are usually separated into Red, Green, Blue (RGB) or Luminance (Y), with Blue and Red color differences (U=B-Y, V=R-Y). Separation into YUV color components allows the algorithm to take advantage of the human eye's lower sensitivity to color information. The U and V color components are commonly recorded at a lower bandwidth and sub-sampled to one-half in the horizontal dimension (called 4:2:2), or one-half in both the horizontal and vertical (called 4:2:0). JPEG partitions each color component picture into 8x8 pixel blocks of image samples. An 8x8 DCT is applied to each block. For quantization, JPEG uses quantization matrices. JPEG allows a different quantization matrix to be specified for each color component. Using quantization matrices allow each frequency bin to be quantized to a different step size. Generally the lower frequency components are quantized to a small step size and the high frequency components to a large step size. This takes advantage of the fact that the human eye is less sensitive to high frequency visual noise, but is more sensitive to lower frequency noise, manifesting itself in blocking artifacts. Modification of the quantization matrices is the primary method for controlling the quality and compression ratio in JPEG. Although the quantization step size for any one of the frequency components can be modified individually, a more common technique is to scale all the elements of the

matrices together. The final stage of compression is the zig-zag scanning and entropy coding. Although JPEG allows for two different types of entropy coders, nearly all JPEG implementations use the Huffman coding option. The JPEG standard allows for the use of user-definable Huffman coding tables. To decompress a JPEG image each operation is performed in reverse order.

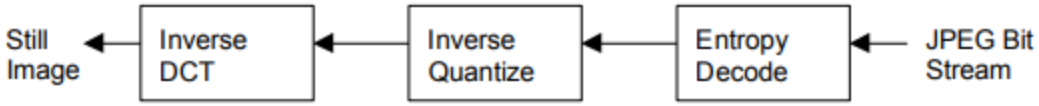


Figure 1.5 JPEG Decoding

1.5. THE H.261 COMPRESSION ALGORITHM

Video conferencing and video telephony are the intended applications for the H.261 compression algorithm. For these applications, representation of limited motion video (taking heads) is a key component. To allow for low-cost implementations, H.261 fixes many of the system parameters. Only the YUV color component separation with the 4:2:0 sampling ratio is allowed by the standard. In addition, H.261 allows for only two frame sizes, CIF (352x288) and QCIF (176x144).

As with the JPEG standard, each color component picture is partitioned into 8x8 pixel blocks of picture samples. Instead of coding each block separately, H.261 groups 4 Y blocks, 1 U block, and 1 V block together into a unit called a macroblock. The macroblock is the basic unit for compression.

CHAPTER 2

LITERATURE SURVEY

2.1 A STUDY OF SUBJECTIVE VIDEO QUALITY AT VARIOUS FRAME RATES

This paper presents a new video database (BVI-HFR), which contains content with a variety of frame rates from 15Hz to 120Hz, that can be used to demonstrate the benefits and limitations of higher frame rates, as well as investigating the role that frame rates play from capture to delivery. A characterization of the video database using low-level descriptors is also provided, which establishes that it successfully spans a variety of scene types and motions, and compares well to existing video databases. Subjective evaluations performed on the video database, have demonstrated a significant relationship between frame rates and perceived quality, up to 120Hz. They also confirm that the relationship between frame rate and perceived quality is content dependent.

2.2 A STUDY OF SUBJECTIVE VIDEO QUALITY AT VARIOUS SPATIAL RESOLUTIONS

In this paper we present the BVI-SR video database, which contains 24 unique video sequences at a range of spatial resolutions up to UHD-1 (3840p). These sequences were used as the basis for a large-scale subjective experiment exploring the relationship between visual quality and spatial resolution when using three distinct spatial adaptation filters (including a CNN-based super-resolution method). The results demonstrate that while spatial resolution has a significant impact on mean opinion scores (MOS), no significant reduction in visual quality between UHD-1 and HD resolutions for the super-resolution method is reported. A selection of image quality metrics were benchmarked on the subjective evaluations, and analysis indicates that VIF offers the best performance.

2.3 DOWN-SAMPLING BASED VIDEO CODING USING SUPER-RESOLUTION TECHNIQUE

It has been reported that oversampling a still image before compression does not guarantee a good image quality. Similarly, down-sampling before video compression in low bit rate video coding may alleviate the blocking effect and improve peak signal-to-noise ratio of the decoded frames. When the number of discrete cosine transform coefficients is reduced in such a down-sampling based coding (DBC), the bit budget of each coefficient will increase, thus reduce the quantization error. A DBC video coding scheme is proposed in this paper, where a super-resolution technique is employed to restore the down-sampled frames to their original resolutions. The performance improvement of the proposed DBC scheme is analyzed at low bit rates, and verified by experiments.

2.4 REDUCED COMPLEXITY SUPER RESOLUTION FOR LOW-BIT RATE VIDEO COMPRESSION

Evolving video applications impose requirements for high image quality, low bitrate, and/or small computational cost. This paper combines state-of-the-art coding and superresolution (SR) techniques to improve video compression both in terms of coding efficiency and complexity. The proposed approach improves a generic decimation-quantization compression scheme by introducing low complexity single-image SR techniques for rescaling the data at the decoder side and by jointly exploring/optimizing the downsampling/upsampling processes. The enhanced scheme achieves improvement of the quality and system's complexity compared with conventional codecs and can be easily modified to meet various diverse requirements, such as effectively supporting any off-the-shelf video codec, for instance H.264/Advanced Video Coding or High Efficiency Video Coding. Our approach builds on studying the generic scheme's parameterization with common rescaling techniques to achieve 2.4-dB peak signal-to-noise ratio (PSNR) quality improvement at low-bitrates compared with the conventional

codecs and proposes a novel SR algorithm to advance the critical bitrate at the level of 10 Mb/s. The evaluation of the SR algorithm includes the comparison of its performance to other image rescaling solutions of the literature. The results show quality improvement by 5-dB PSNR over straightforward interpolation techniques and computational time reduction by three orders of magnitude when compared with the highly involved methods of the field. Therefore, our algorithm proves to be most suitable for use in reduced complexity downsampled compression schemes.

2.5 ADAPTIVE DOWNSAMPLING VIDEO CODING WITH SPATIALLY SCALABLE RATE-DISTORTION MODELING

Downsampling video coding, whereby downsampled frames are encoded, provides improved perceptual quality in rate-constrained situations. This method shows considerable advantages over other approaches, particularly in wide-spreading high-definition video formats. This paper provides a comprehensive analysis of downsampling video coding. The study proposes a spatially scalable rate-distortion (RD) model, comprising quantization-distortion and quantization-rate models, and develops an optimal encoding frame size determination framework. The proposed method achieves a gain up to 2.3 dB peak signal-to-noise ratio (PSNR) at 1 Mb/s when compared with conventional full frame size coding. The RD performance is close to the optimal scenario, in which the ideal frame size is obtained by heuristically performing downsampling coding in various allowable sizes.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

A Quantization-Resolution Optimization (QRO) module which applies perceptual quality metrics and machine learning techniques to generate reliable resolution adaptation decisions.The employment of a CNN-based super resolution model to reconstruct full spatial resolution content, trained specifically for compressed content.The integration of the ViSTRA framework with HEVC reference software.

3.1.2 LIMITATIONS OF EXISTING SYSTEM

- Failure to store images.
- Security problem.
- This procedure is required due to the fact that the HEVC does not inherently support the encoding of different spatial resolutions.

3.2 PROPOSED SYSTEM

In this proposed work takes video as input, from that given video is converted into snapshots or frames using Key Frame Technique. The converted frames into Motion Detection using background frame matching for pixel calculation. This method is very efficient method of comparing image pixel values in subsequent still frames captured after every 0.5 seconds from the video. Two frames are required to detect movement. First frame is called reference frame and the second frame, which is called the input frame contains the moving object. The two frames are compared and the differences in pixel values are determined. Finally Frame Difference Approach, wavelet transform is an efficient method that can be used to perform an efficient video compression technique. This work can be concentrated on the calculation of frame near distance (difference between frames). Histogram Matching and Discrete Cosine Transform are used for decompressing the video.

3.2.1 ADVANTAGES OF PROPOSED SYSTEM

- The images are represented as digitized samples containing visual (color and intensity information at each spatial and temporal location.
- Visual information at each sample point may be represented by the values of the three basic color components RGB color space.
- A video signal can be sampled in either frames (progressive) or fields (interlaced).

CHAPTER 4

SYSTEM SPECIFICATIONS

4.1 HARDWARE REQUIREMENTS

- **Processor** : Pentium IV 2.4 GHz.
- **Hard Disk** : 80 GB.
- **Monitor** : 15 VGA Color.
- **Mouse** : Logitech.
- **RAM** : 4 GB.

4.2 SOFTWARE REQUIREMENTS

- **Operating system** : Windows 8/8.1
- **Front End** : Net beans 8.0.2/Java

4.3 SOFTWARE SPECIFICATION

4.3.1 FRONT END –NETBEANS / JAVA

4.3.1.1 INTRODUCTION

NetBeans is an Integrated Development Environment (IDE) for Java. NetBeans allows applications to be developed from a set of modular software components called modules. NetBeans runs on Windows, Mac OS, Linux and Solaris. In addition to Java development, it has extensions for other languages like PHP, C, C++, HTML and JavaScript. Applications based on NetBeans, including the NetBeans IDE, can be extended by third party developers.

4.3.1.2 NETBEANS PLATFORM

The NetBeans platform is a framework for simplifying the development of Java swing desktop applications. The NetBeans IDE bundle for Java SE contains what is needed to start developing NetBeans plug-ins and NetBeans platform based applications; no additional SDK is required.

Applications can install modules dynamically. Any application can include the update centre module to allow users of the application to download digitally signed upgrades and new features directly into the running application. Reinstalling an upgrade or a new release does not force users to download the entire application again.

The platform offers reusable services common to desktop applications, allowing developers to focus on the logic specific to their application. Among the features of the platform are:

- User interface management
- User settings management
- Storage management
- Window management
- Wizard framework
- NetBeans visual library
- Integrate development tools

4.3.1.3 NETBEANS IDE

NetBeans IDE is an open source integrated development environment. NetBeans IDE supports development of all Java application types out of the box. Among other features is an Ant-based project system, Maven support, Refactoring, Version Control.

4.4 THE JAVA PLATFORM

4.4.1 JAVA

Java acts as the front end, which drives its syntax from C and object-oriented features from C++. The main feature is platform independent. Java is

popular among Internet programmers. It expends the universe of objects that can move about freely in cyberspace. Java can be used to create two types of programs, application and applets. An application is a program that runs on the computer, under the operation system of that computer.

4.4.2 THE BYTE CODE

The output of java compiler is not executable code, it is byte code. It is a set of instructions to be executed by java run-time system called java virtual machine (JVM) it is an interpreter for byte code. Some of the java Buzzwords.

- **Simple**
- **Secure**
- **Portable**
- **Object-oriented**
- **Robust**
- **Multithreaded**
- **Architecture-neutral**
- **Interpreted**
- **High performance**
- **Distributed**
- **Dynamic**

4.4.3 SIMPLE

It inherits the C syntax and many of the object oriented features of C++. It is the consolidated form of both.

4.4.4 PORTABLE

Programs to be dynamically downloaded to all various types of platforms connected to the Internet, some means of generating portable executable code is needed.

4.4.5 SECURE

With a help of java-compatible web browser, you can safely download java applets without fear. It is possible with the help of java execution environment and not allowing it access to other parts of computer.

4.4.6 OBJECT-ORIENTED

It organizes a program around its data and a set of well-defined interfaces to that data. Some of its concepts are as follows:

- **Abstraction**
- **Encapsulation**
- **Inheritance**
- **Polymorphism**
- **Exception Handling**

4.4.7 ABSTRACTION

It's the way of hiding the internal details but showing the external behavior. Example is car without the details of how it was made.

4.4.8 ENCAPSULATION

It is the mechanism that binds together code and the data it manipulates, and keeps both safe from outside interference and misuse. In java the basics of encapsulation is class, which is shared by the objects. Objects are sometimes called instances of a class.

4.4.9 INHERITANCE

Inheritance is the process by which one object acquires the properties of another object. It supports the concept of hierarchical classification for example golden retriever is part of classification dog, which in turn is part of mammal class, which is under the class animal without in each object would need to define its characteristics explicitly.

4.4.10 POLYMORPHISM

It is a concept that allows one interface to be used for general class of actions. Consider one example as stack. It is of three types one used for integer values one for floating values one for characters. The algorithms used are same in non-object oriented programming you would create three types of routines and with the help of polymorphism the general stack of routines is enough.

4.4.11 EXCEPTION HANDLING

Java provides very powerful exception handling mechanism. Exception handling minimizes the runtime loss. Java supports 100% Object Oriented exception handling mechanism, in fact java itself is 100% OPPs. With the help of this feature various runtime errors are trapped and handled.

4.4.12 ROBUST

Java frees from having to worry about many of the programming errors. It checks the code at compile time and also at runtime. For example java eliminates the memory management problem by managing memory allocation and reallocation, exceptional condition such as division by zero can be resolved with the help of exception handling.

4.4.13 MULTITHREADED

Multitasking is a concept introduced in operating systems such as UNIX, Windows etc., Multitasking is allowed only with 32 Bit Operating Systems. Multitasking supports concurrent execution of multiple tasks. The Operating System take care of the scheduling algorithms and switching logics. Now these powerful concepts in Operating Systems can be adopted in programming languages such as VB, JAVA etc., This technique is called as multithreading. Here program is divided into independent units called threads. Each thread can be executed concurrently. Java provides easy mechanism for multithreading. These are the programs that do many things simultaneously.

4.4.14 ARCHITECTURAL-NEUTRAL

Architectural Neutrality means Hardware independent. Cent percent Architectural Neutrality is not possible but we can develop tools to archive this with in very short period when compared with other languages. The main goal of it is that write once, run anywhere, any time.

4.4.15 DISTRIBUTED

Java is designed for the distributed environment of the Internet, because it handles TCP/IP protocols. It is possible that two different computers to execute procedures remotely. Java revived these interfaces in package called Remote Method Invocation (RMI). Even the java libraries can also be distributed.

Event Driven programming model is a powerful one. We can not predict the flow of execution before any user interaction. Systems Flow is completely controlled by the user's action called events. Java provides event driven programming model, by its powerful libraries in AWT-Event and Swing – Event packages. Applets are event-driven programs. There are several types of events.

The most common events are those generated by the mouse, the keyboard, and various control events are supported by java.awt.event package.

4.4.16 PLATFORM-INDEPENDENT AND PORTABLE

The most significant contribution of Java over other languages is its portability. Java programs can be easily moved from one computer system to another, anywhere and anytime. Changes and upgrades in operating systems, processors and system resources will not force any changes in Java programs. Secondly, the sizes of the primitive data types are machine-independent.

4.4.17 ROBUST AND SECURE

Java is a robust language. It provides many safeguards to ensure reliable code. It has strict compile time and run time checking for data types. Java also incorporates the concept of exception handling which captures series of errors and eliminates any risk of crashing the system. Security becomes an important issue for a language that is used for programming on Internet.

4.4.18 MULTITHREADED AND INTERACTIVE

Multithreaded means handling multiple tasks simultaneously. Java supports multithreaded programs. This means that we need not wait for the application to finish one task before beginning another. For example, we can listen to an audio clip while scrolling a page and at the same time download an applet from a distant computer. This feature greatly improves the interactive performance of graphical applications.The Java runtime comes with tools that support multi-process synchronization and construct smoothly running interactive systems.Java programs support functions written in other languages such as C and C++. These functions are known as native methods. This facility enables the programmers to use the efficient functions available in these languages.Native methods are linked dynamically at runtime.

4.4.19 THREADING TECHNIQUES

Unlike most other computer languages, Java provides built-in support for multithreaded programming. A multithreaded program contains two or more parts that can run concurrently. Each part of such a program is called a thread, and each thread defines a separate path of execution. Thus, multithreading is a specialized form of multitasking.

However, there are two distinct types of multitasking: process-based and thread-based. In a thread-based multitasking environment, the thread is the smallest unit of dispatch able code. This means that a single program can perform two or more tasks simultaneously. Thus, process-based multitasking deals with the "big picture", and thread-based multitasking handles the details.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

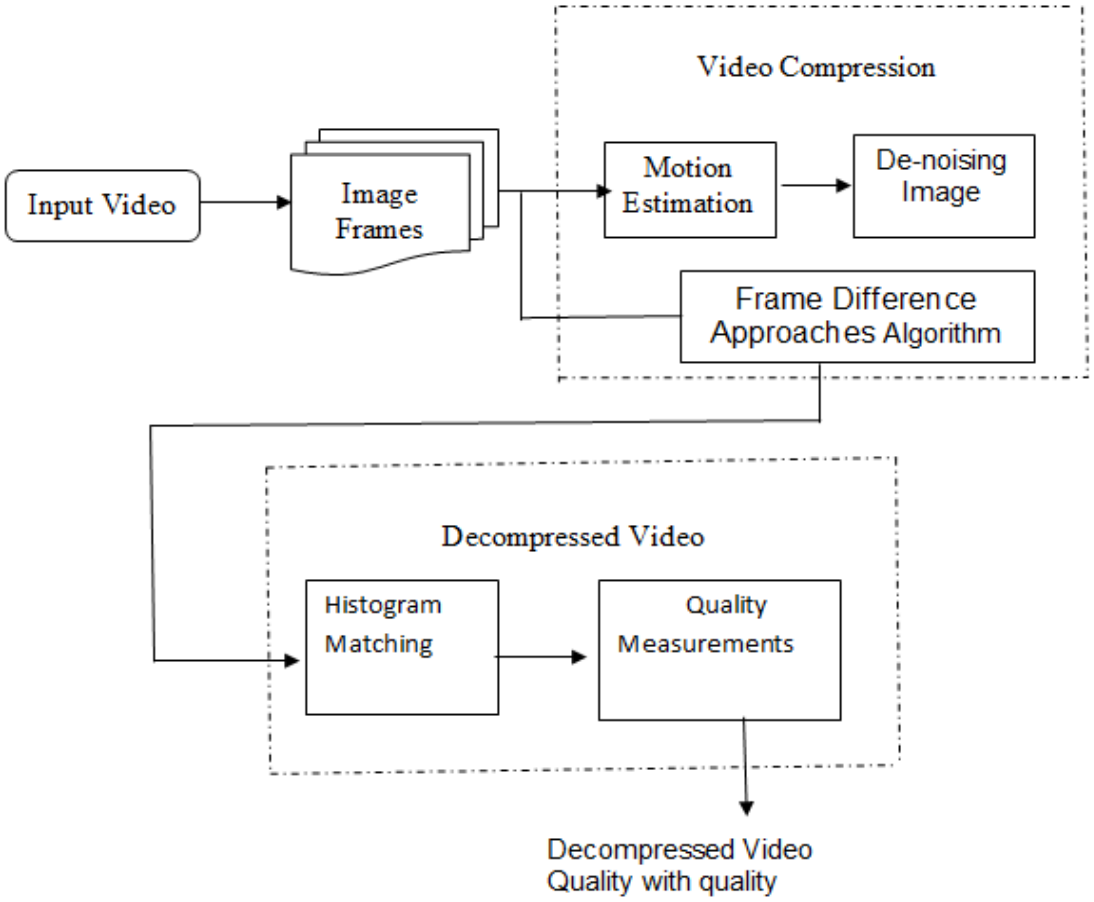


Fig 5.1 Video compression and Decompression

Video as input, from that given video is converted into snapshots or frames using Key Frame Technique. The converted frames into Motion Detection using background frame matching for pixel calculation. This method is very efficient method of comparing image pixel values in subsequent still frames captured after every 0.5 seconds from the video. Two frames are required to detect movement. First frame is called reference frame and the second frame, which is called the input frame contains the moving object. The

two frames are compared and the differences in pixel values are determined. Finally Frame Difference Approach, wavelet transform is an efficient method that can be used to perform an efficient video compression technique.

CHAPTER 6

SYSTEM IMPLEMENTATION

6.1 MODULES

- CS Camera Module
- CS Video Encoder Module
- Rate Controller Subsystem Module
- Adaptive Parity-based Transmission

6.2 MODULE DESCRIPTION

6.2.1 CS CAMERA MODULE

This module deals with compressed sensing image capture, where the camera can be either a traditional CCD or CMOS imaging system, or it can be a single-pixel camera. The image samples are obtained by combining random set of pixels and summing the intensity through photodiode and finally the obtained samples are stored for the CS video encoder module.

6.2.2 CS VIDEO ENCODER MODULE

In this module, the samples obtained from the camera are received and the compressed video frames are generated. The number of samples and the sampling matrix is decided in this module. The number of samples is generated based on the input from the rate controller module and the sampling matrix is preselected by the sender and receiver.

6.2.3 RATE CONTROLLER SUBSYSTEM MODULE

Rate Controller module plays an important role in congestions control. The sending node needs to take two main factors into account. First, the sender needs to regulate its rate in such a way as to allow any competing transmission at least as much bandwidth as it needs to attain a comparable video quality as itself. Second, the sender needs to regulate its rate to make sure that packet losses due to buffer overflows are reduced. The end-to-end RTT (Round Trip Time) and the estimated

sample loss are provided as input in this module and give the optimal sample rate. The optimal sample rate is then sent to the Video Encoder module.

6.2.4 ADAPTIVE PARITY BASED TRANSMISSION

For a fixed number of bits per frame, the perceptual quality of video streams can be further improved by dropping error samples that would contribute to image reconstruction with incorrect information, which shows the reconstructed image quality both with and without including samples containing errors. It assume that the receiver knows which samples have errors, they demonstrate that there is a very large possible gain in received image quality if those samples containing errors can be removed. The adaptive parity with compressed sensing for image transmission is studied, where the transmitted samples constitute an unstructured, random, incoherent combination of the original image pixels, in CS, unlike traditional wireless imaging systems, no individual sample is more important for image reconstruction than any other sample. Instead, the number of correctly received samples is the only main factor in determining the quality of the received image.

CHAPTER 7

SYSTEM TESTING

7.1 TYPES OF TESTING

7.1.1 UNIT TESTING

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It is often done by the programmer to test that the unit he/she has implemented produces expected output against given input.

7.1.2 INTEGRATION TESTING

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware is tested in integration testing if software and hardware components have any relation. It may fall under both white box testing and black box testing.

7.1.3 FUNCTIONAL TESTING

Functional testing is the testing to ensure that the specified functionality required in the system requirements works. It falls under the category of black box testing.

7.1.4 SYSTEM TESTING

System testing is performed to ensure that the software still works by putting in different environments. System testing is done with full system implementation and environment. It falls under the category of black box testing.

7.1.5 WHITE BOX TESTING

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called a structural testing and glass box testing. White box testing is used for verification.

7.1.6 BLACK BOX TESTING

Black box testing is attesting technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing. it is used for validation.

7.1.7 ACCEPTANCE TESTING

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the expected. It falls under the category of black box testing.

7.2 TEST RESULTS

All the above types of testing were performed and no defect is found.

CHAPTER 8

CONCLUSION

8.1 CONCLUSION

An efficient video compression approach based on frames difference approaches are developed that concentrated on the calculation of frame near distance. Many factors are applied in the selection of meaningful frames, in which eliminate the similar frames. The implemented system passes into many steps; pre-processing, frame extraction, frame selection, frame reordering, 2DDWT, then video construction. Different types of videos are introduced to test the system. The output compressed video is in a good quality and good performance as well as it has a specific compression ratio.

8.2 FUTURE ENHANCEMENT

The importance of security will be used in the next issue where encrypt DCT (Discrete Cosine Transform) coefficients, MV (Motion Vector) and other parameters in video encoding will be discussed. In addition it will dissolve the issues of rate–distortion (R– D) optimized error-resilient scheme and channel losses.

APPENDIX –I SOURCE CODE

Home.java:

```
package videocompression;
import javax.swing.UIManager;
import com.birosoft.liquid.LiquidLookAndFeel;
public class Home extends javax.swing.JFrame {
    public Home() {
        initComponents();
    }
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jPanel2 = new javax.swing.JPanel();
        jLabel1 = new javax.swing.JLabel();
        jPanel3 = new javax.swing.JPanel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();


setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("DCT Image Compression and Decompression");
        setResizable(false);


jPanel2.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.bord
```

```
er.BevelBorder.RAISED));

        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
        jLabel1.setText("Data Compression and Decompression");

        javax.swing.GroupLayout jPanel2Layout = new
javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGap(78, 78, 78)
                .addComponent(jLabel1)
                .addContainerGap(109, Short.MAX_VALUE))
        );
        jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()
                .addContainerGap(24, Short.MAX_VALUE)
                .addComponent(jLabel1)
                .addGap(23, 23, 23))
        );
```

```
jPanel3.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.bord
er.BevelBorder.RAISED));

        jLabel2.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
        jLabel2.setText(">   Image Compression ");
        jLabel2.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        jLabel2.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jLabel2MouseClicked(evt);
            }
        });

        jLabel3.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
        jLabel3.setText(">   Image  Decompression");
        jLabel3.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        jLabel3.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jLabel3MouseClicked(evt);
            }
        });

        jLabel4.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
        jLabel4.setText(">   Video Frame Conversion");
        jLabel4.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        jLabel4.addMouseListener(new java.awt.event.MouseAdapter() {
```

```
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jLabel4MouseClicked(evt);
        }
    });

    javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
    jPanel3.setLayout(jPanel3Layout);
    jPanel3Layout.setHorizontalGroup(
        jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel3Layout.createSequentialGroup()
            .addGap(118, 118, 118)
            .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel4)
                .addComponent(jLabel3)
                .addComponent(jLabel2))
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
    );
    jPanel3Layout.setVerticalGroup(
        jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel3Layout.createSequentialGroup()
                .addGap(37, 37, 37)
                .addComponent(jLabel4)
                .addGap(34, 34, 34)
                .addComponent(jLabel2)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,  43,
Short.MAX_VALUE)
                .addComponent(jLabel3)
                .addGap(61, 61, 61))
        );

        javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAI
LING, false)
                    .addComponent(jPanel3,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jPanel2,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
        );
        layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(18, 18, 18)
                .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jPanel3,      javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addContainerGap())
        );

        pack();
    }// </editor-fold>

    private void jLabel2MouseClicked(java.awt.event.MouseEvent evt) {
```

```
        new ImageCompression().setVisible(true);
        this.setVisible(false);
    }

    private void jLabel3MouseClicked(java.awt.event.MouseEvent evt) {
        new ImageDecompression().setVisible(true);
        this.setVisible(false);
    }

    private void jLabel4MouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        new HomePage().setVisible(true);
    }
    public static void main(String args[]) {
        try {
            LiquidLookAndFeel.setLiquidDecorations(true, "mac");
            LiquidLookAndFeel.setPanelTransparency(true);

javax.swing.UIManager.setLookAndFeel("com.birosoft.liquid.LiquidLookAndFeel");
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Home().setVisible(true);
            }
```

```
        });
    }

    // Variables declaration - do not modify
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JPanel jPanel3;
    // End of variables declaration

}
```

Image Compression.java:

```
/*
 * Copyright (c) 1994, 2004, Oracle and/or its affiliates. All rights reserved.
 * ORACLE PROPRIETARY/CONFIDENTIAL. Use is subject to license terms.
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 *
 */
```

Image Decompression.java:

package videocompression:

```
import java.awt.Color;
import java.awt.Container;
import java.io.ByteArrayInputStream;
import java.io.ByteArrayOutputStream;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.UnsupportedEncodingException;
import java.util.*;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.zip.DataFormatException;
import java.util.zip.GZIPInputStream;
import java.util.zip.Inflater;
import javax.swing.JFileChooser;
import javax.swing.filechooser.FileNameExtensionFilter;
import org.apache.commons.io.IOUtils;


public class ImageDecompression extends javax.swing.JFrame {

    private String outputString;
```

```
public ImageDecompression() {
    initComponents();
}


@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jPanel1 = new javax.swing.JPanel();
    jLabel2 = new javax.swing.JLabel();
    txtCF = new javax.swing.JTextField();
    jButton1 = new javax.swing.JButton();
    jButton2 = new javax.swing.JButton();
    jLabel3 = new javax.swing.JLabel();
    txtDS = new javax.swing.JTextField();
    jButton3 = new javax.swing.JButton();
    jLabel4 = new javax.swing.JLabel();
    txtDSSize = new javax.swing.JTextField();
    jLabel5 = new javax.swing.JLabel();
    txtCFSize = new javax.swing.JTextField();
    jLabel6 = new javax.swing.JLabel();
    jLabel7 = new javax.swing.JLabel();
    jPanel2 = new javax.swing.JPanel();
    jLabel1 = new javax.swing.JLabel();
```

```
setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setTitle("Image Decompression");
        setResizable(false);


jPanel1.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.bord
er.BevelBorder.RAISED));

        jLabel2.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
        jLabel2.setText("Decompressed Image File");

        txtCF.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N

        jButton1.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
        jButton1.setText("Browse");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });

        jButton2.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
        jButton2.setText("Open");
        jButton2.setEnabled(false);
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
        jButton2ActionPerformed(evt);
    }
});

jLabel3.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
jLabel3.setText("Select Compressed Image File");

txtDS.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N

jButton3.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
jButton3.setText("Decompress");
jButton3.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jButton3ActionPerformed(evt);
    }
});

jLabel4.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
jLabel4.setText("Decompressed File Size");

txtDSSize.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N

jLabel5.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
jLabel5.setText("Compressed File Size");

txtCFSize.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
```

```
        jLabel6.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
        jLabel6.setForeground(new java.awt.Color(0, 153, 0));

        jLabel7.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
        jLabel7.setForeground(new java.awt.Color(255, 51, 51));
        jLabel7.setText("< Back");
        jLabel7.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        jLabel7.addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                jLabel7MouseClicked(evt);
            }
        });

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addContainerGap()
```

```
.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel3)
                    .addComponent(jLabel5)
                    .addComponent(jLabel4)
                    .addComponent(jLabel2)))
                .addGroup(jPanel1Layout.createSequentialGroup()
                    .addGap(72, 72, 72)
                    .addComponent(jLabel7)))
            .addGap(26, 26, 26)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
                .addComponent(jLabel6,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 225, Short.MAX_VALUE)
                .addComponent(txtCFSize,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 225, Short.MAX_VALUE)
                .addComponent(txtDSSize,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 225, Short.MAX_VALUE)
                .addComponent(txtDS,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 225, Short.MAX_VALUE)
                .addComponent(txtCF,
javax.swing.GroupLayout.Alignment.LEADING,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE,                    225,
javax.swing.GroupLayout.PREFERRED_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
                .addComponent(jButton1,
javax.swing.GroupLayout.DEFAULT_SIZE, 105, Short.MAX_VALUE)
                .addComponent(jButton3,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
            .addComponent(jButton2,    javax.swing.GroupLayout.DEFAULT_SIZE,
84, Short.MAX_VALUE)
            .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADIN
G)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(46, 46, 46)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
```

```
                .addComponent(txtCF,
javax.swing.GroupLayout.PREFERRED_SIZE,                    31,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel3)
                .addComponent(jButton1,
javax.swing.GroupLayout.PREFERRED_SIZE,                    30,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
                .addComponent(txtDS,
javax.swing.GroupLayout.PREFERRED_SIZE,                    34,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton3,
javax.swing.GroupLayout.PREFERRED_SIZE,                    31,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jButton2,
javax.swing.GroupLayout.PREFERRED_SIZE,                    31,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel2))
            .addGap(30, 30, 30)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
                .addComponent(txtDSSize,
javax.swing.GroupLayout.PREFERRED_SIZE,                    33,
```

```
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel5))
            .addGap(25, 25, 25)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.BASELINE)
                .addComponent(txtCFSize,
javax.swing.GroupLayout.PREFERRED_SIZE,                    28,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jLabel4))
            .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignme
nt.LEADING)
                .addComponent(jLabel6, javax.swing.GroupLayout.DEFAULT_SIZE,
29, Short.MAX_VALUE)
                .addComponent(jLabel7))
            .addContainerGap())
        );


jPanel2.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.bord
er.BevelBorder.RAISED));

        jLabel1.setFont(new java.awt.Font("Tahoma", 1, 18)); // NOI18N
        jLabel1.setText("Image Decompression");
```

```
        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGap(192, 192, 192)
                .addComponent(jLabel1)
                .addContainerGap(261, Short.MAX_VALUE))
        );
        jPanel2Layout.setVerticalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGap(19, 19, 19)
                .addComponent(jLabel1)
                .addContainerGap(19, Short.MAX_VALUE))
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
```

```
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEA
DING)
                .addComponent(jPanel1,
javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(jPanel2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addContainerGap())
      );
      layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
            .addContainerGap(13, Short.MAX_VALUE)
            .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(18, 18, 18)
            .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE)
```

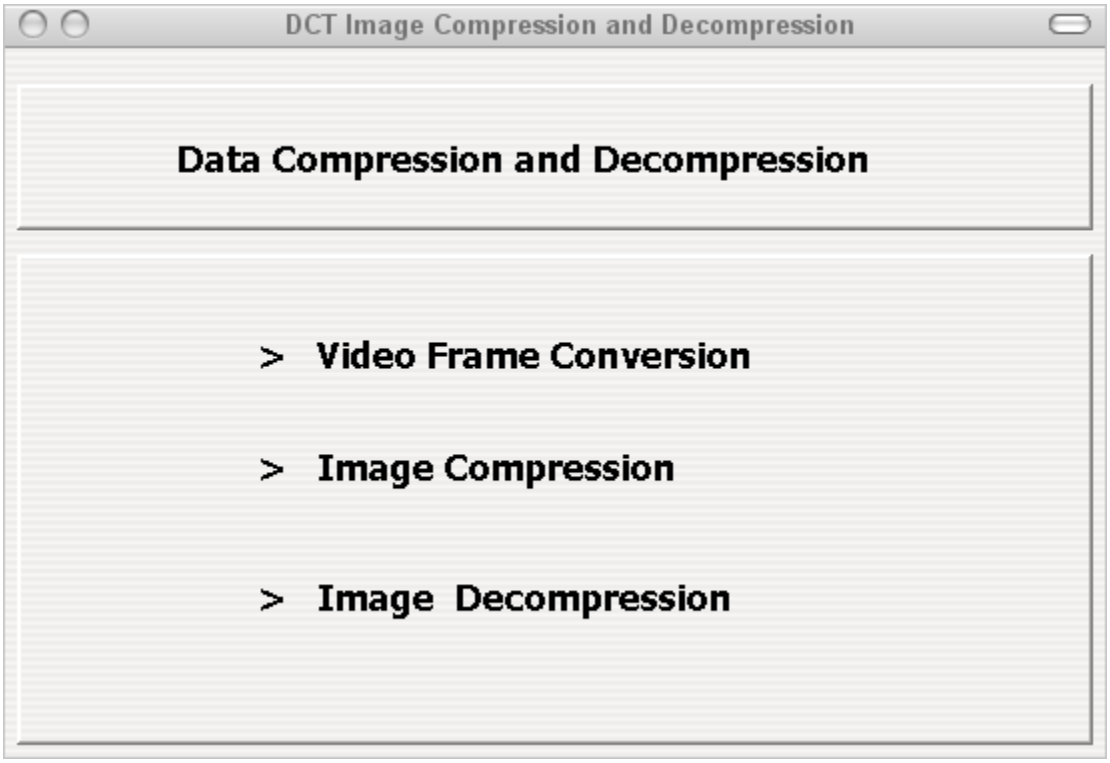
```
                .addContainerGap())
        );

        pack();
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        JFileChooser fc = new JFileChooser(new File("").getAbsolutePath()+"\\Brain
Images");
        fc.setFileFilter(new                    FileNameExtensionFilter("Compressed
File","cmp","jpeg"));
        fc.showOpenDialog(rootPane);
        File f = fc.getSelectedFile();
        txtCF.setText(f.getAbsolutePath());
        StringBuffer s = new StringBuffer(f.getAbsolutePath());
        int lindex = s.lastIndexOf(".cmp");
        String path = s.substring(0, lindex);
        txtDS.setText(path);
        jButton2.setEnabled(true);
    }

    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            Runtime.getRuntime().exec("explorer \"" + txtDS.getText().trim() + "\"");
        } catch (IOException ex) {

Logger.getLogger(ImageCompression.class.getName()).log(Level.SEVERE,   null,
```

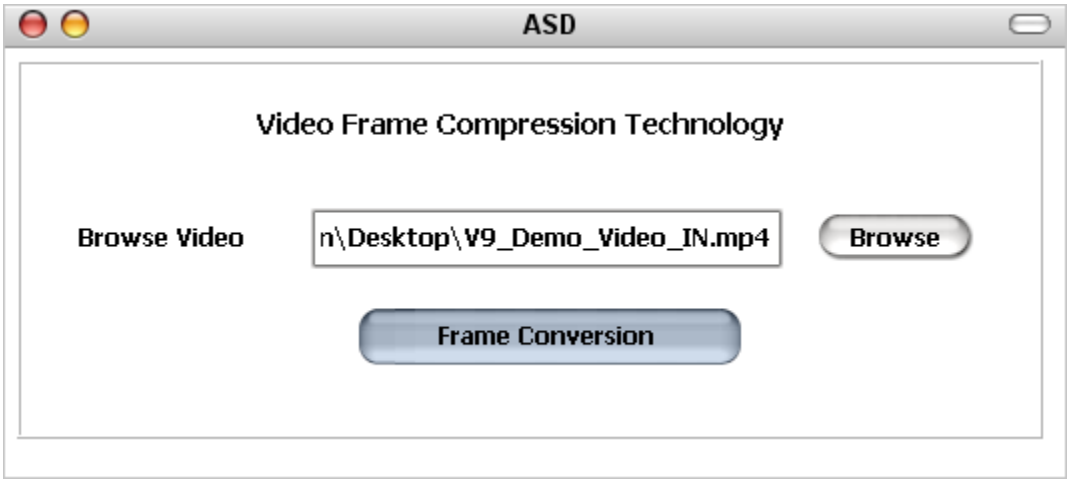
```
ex);
        }
}

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

        try {
            long c1= System.nanoTime();
            File f1 = new File(txtCF.getText());
            File f2 = new File(txtDS.getText());
            if (!f2.exists()) {
                f2.createNewFile();
            }
            FileInputStream fin = new FileInputStream(f1);
            byte readdata[] = new byte[fin.available()];
            fin.read(readdata);

            FileOutputStream fout = new FileOutputStream(f2);
            DCT d=new DCT(25);
            byte outdata[] = decompressImage(readdata);
            for (int i = 0; i < outdata.length; i++) {
                fout.write(outdata[i]);
            }
            txtDSSize.setText(f1.length() + " B");
            txtCFSize.setText(f2.length() + " B");
            fin.close();
            fout.close();
```

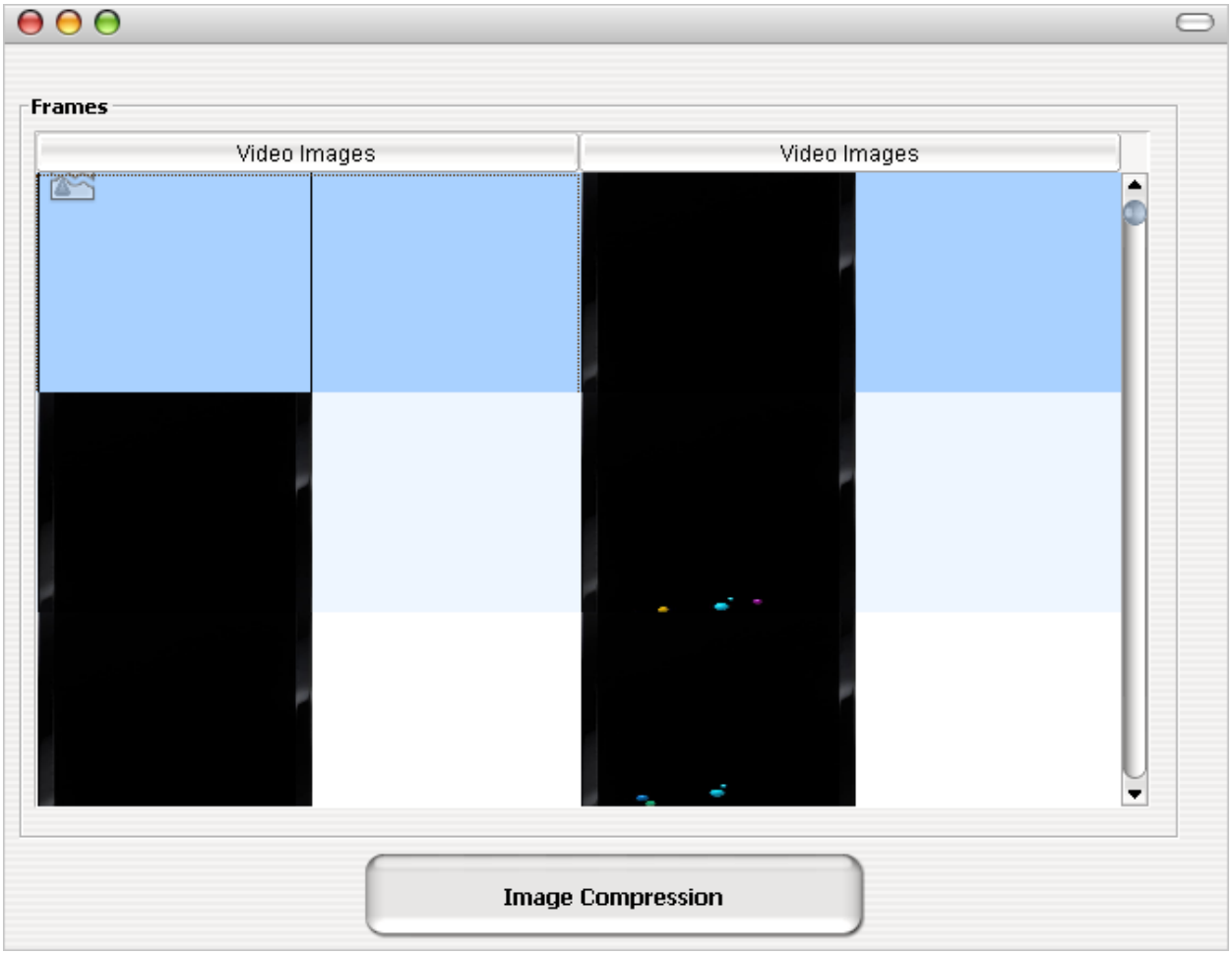
```
            jLabel6.setText("Decompression Done");
            // jButton4.setEnabled(true);
            long c2= System.nanoTime();
            long c3=(c2-c1)/1000000000;



        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    private void jLabel7MouseClicked(java.awt.event.MouseEvent evt) {
      new Home().setVisible(true);
      this.setVisible(false);
    }


    public static byte[] decompressImage(byte[] contentBytes) {
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        try {
//            NeuralDecompress.decompress();
            IOUtils.copy(new                                    GZIPInputStream(new
ByteArrayInputStream(contentBytes)), out);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
```

```
        return out.toByteArray();
    }
/*   public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {

            public void run() {
                new ImageDecompression().setVisible(true);
            }
        });
    }*/
    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JButton jButton3;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JTextField txtCF;
    private javax.swing.JTextField txtCFSize;
    private javax.swing.JTextField txtDS;
    private javax.swing.JTextField txtDSSize;}
```

APPENDIX-II SCREENSHOTS






Frames
Video Images
Video Images
Image Compression

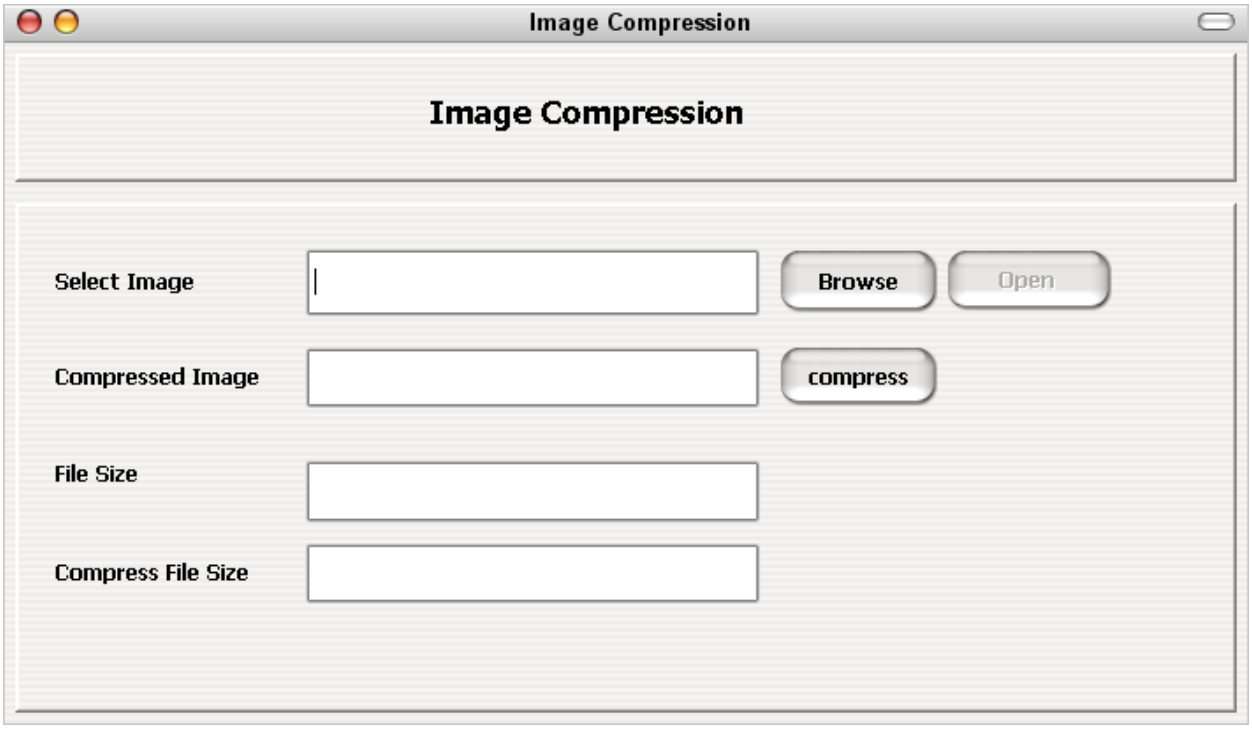
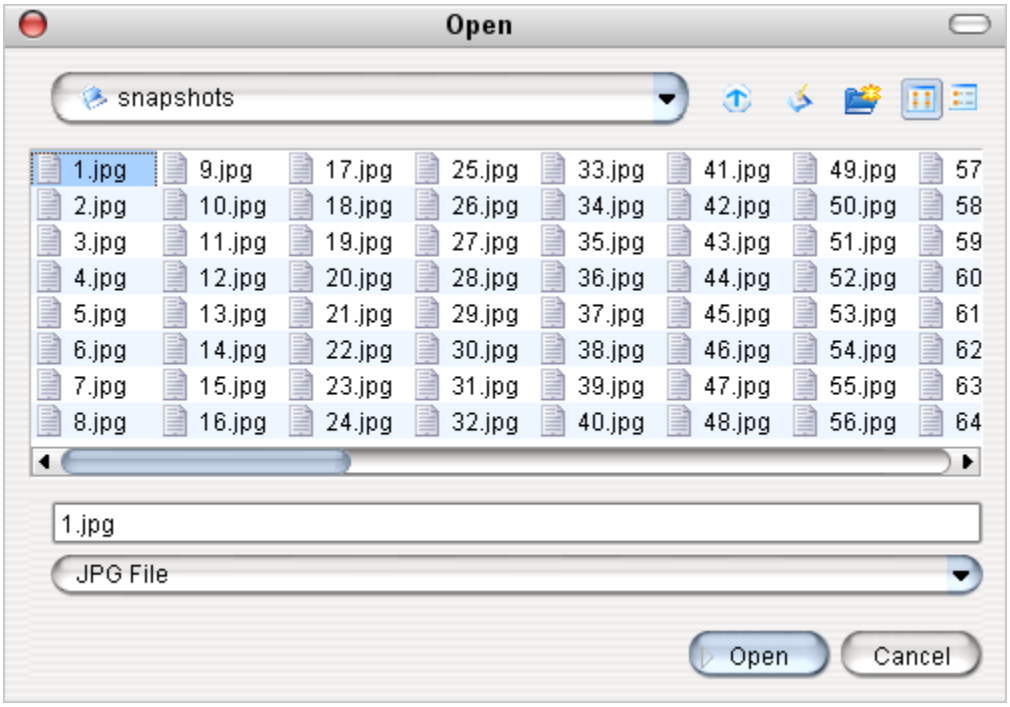


Image Compression
Image Compression
Select Image
Browse
Open
Compressed Image
compress
File Size
Compress File Size


Open
snapshots
1.jpg 9.jpg 17.jpg 25.jpg 33.jpg 41.jpg 49.jpg 57
2.jpg 10.jpg 18.jpg 26.jpg 34.jpg 42.jpg 50.jpg 58
3.jpg 11.jpg 19.jpg 27.jpg 35.jpg 43.jpg 51.jpg 59
4.jpg 12.jpg 20.jpg 28.jpg 36.jpg 44.jpg 52.jpg 60
5.jpg 13.jpg 21.jpg 29.jpg 37.jpg 45.jpg 53.jpg 61
6.jpg 14.jpg 22.jpg 30.jpg 38.jpg 46.jpg 54.jpg 62
7.jpg 15.jpg 23.jpg 31.jpg 39.jpg 47.jpg 55.jpg 63
8.jpg 16.jpg 24.jpg 32.jpg 40.jpg 48.jpg 56.jpg 64
1.jpg
JPG File
Open
Cancel

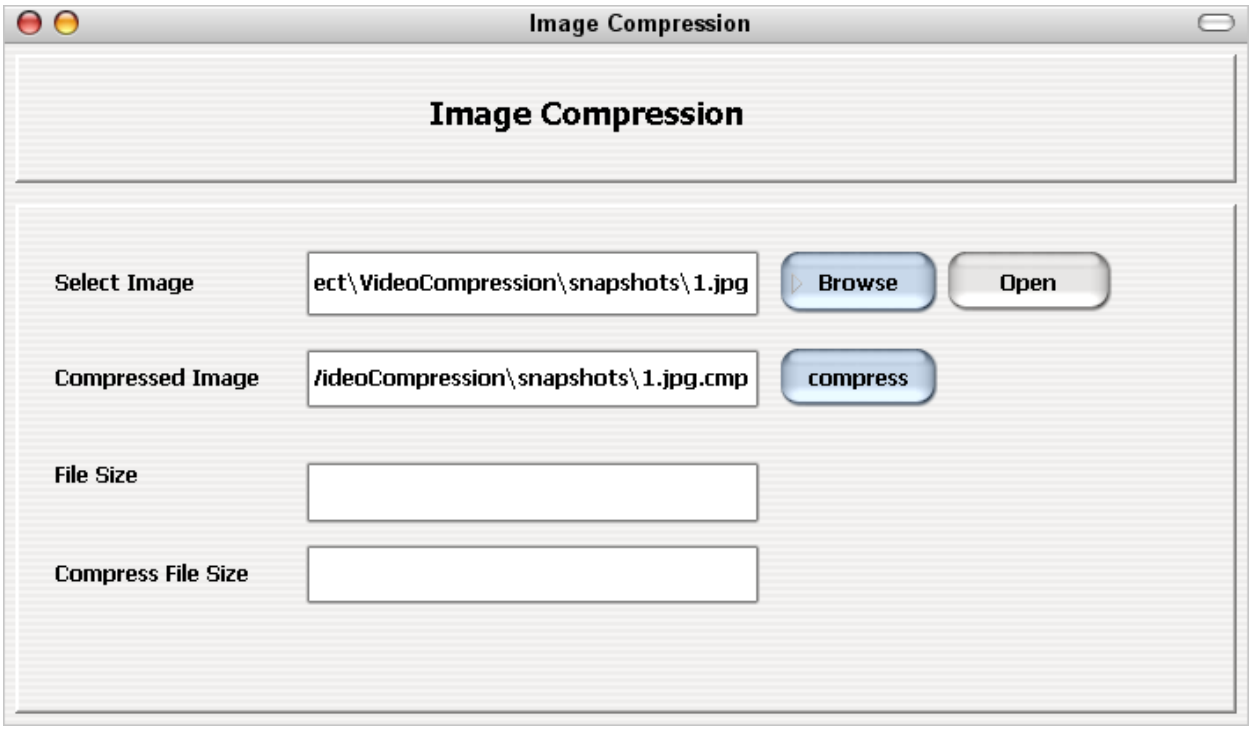


Image Compression
Image Compression
Select Image
ect\VideoCompression\snapshots\1.jpg
Browse
Open
Compressed Image
/ideoCompression\snapshots\1.jpg.cmp
compress
File Size
Compress File Size

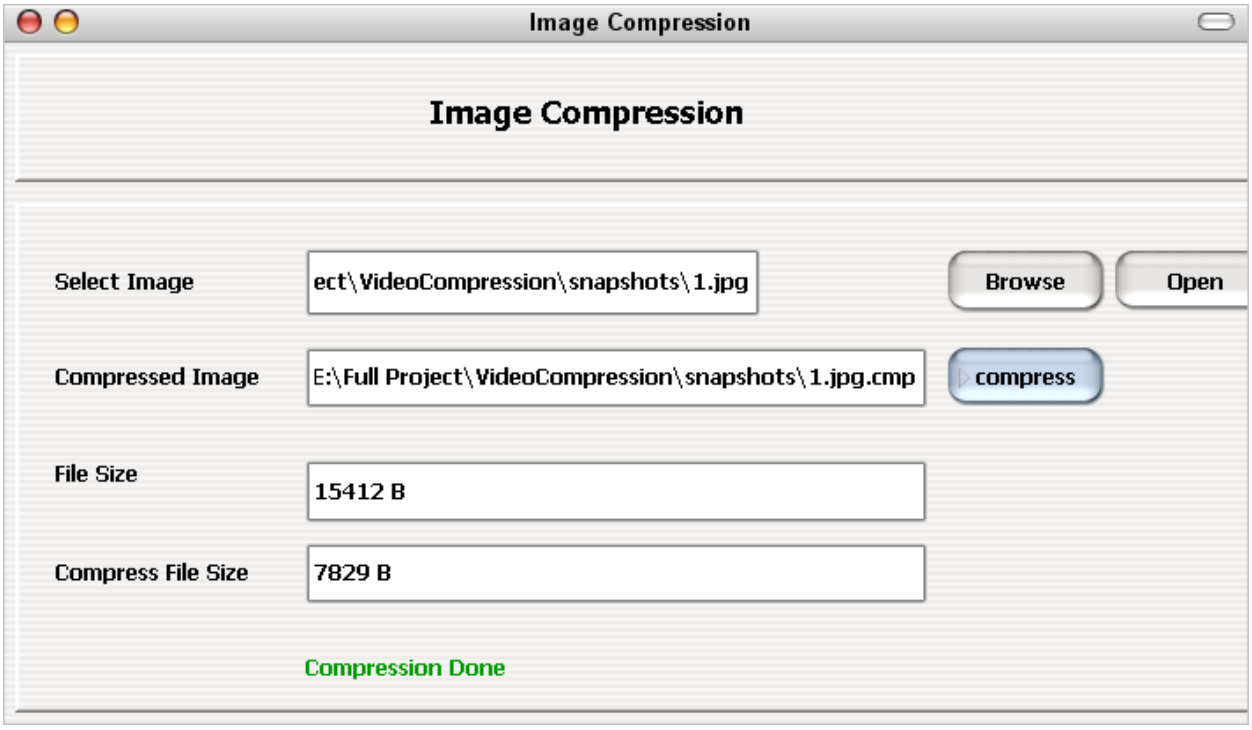
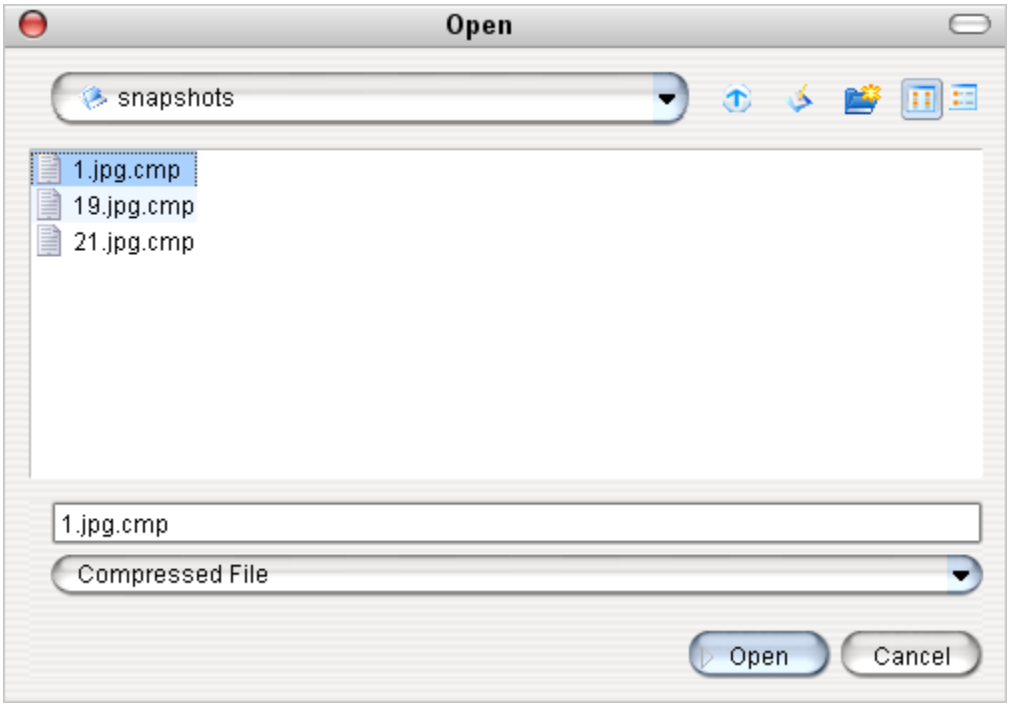


Image Compression
Image Compression
Select Image
ect\VideoCompression\snapshots\1.jpg
Browse
Open
Compressed Image
E:\Full Project\VideoCompression\snapshots\1.jpg.cmp
compress
File Size
15412 B
Compress File Size
7829 B
Compression Done


Open
snapshots
1.jpg.cmp
19.jpg.cmp
21.jpg.cmp
1.jpg.cmp
Compressed File
Open
Cancel

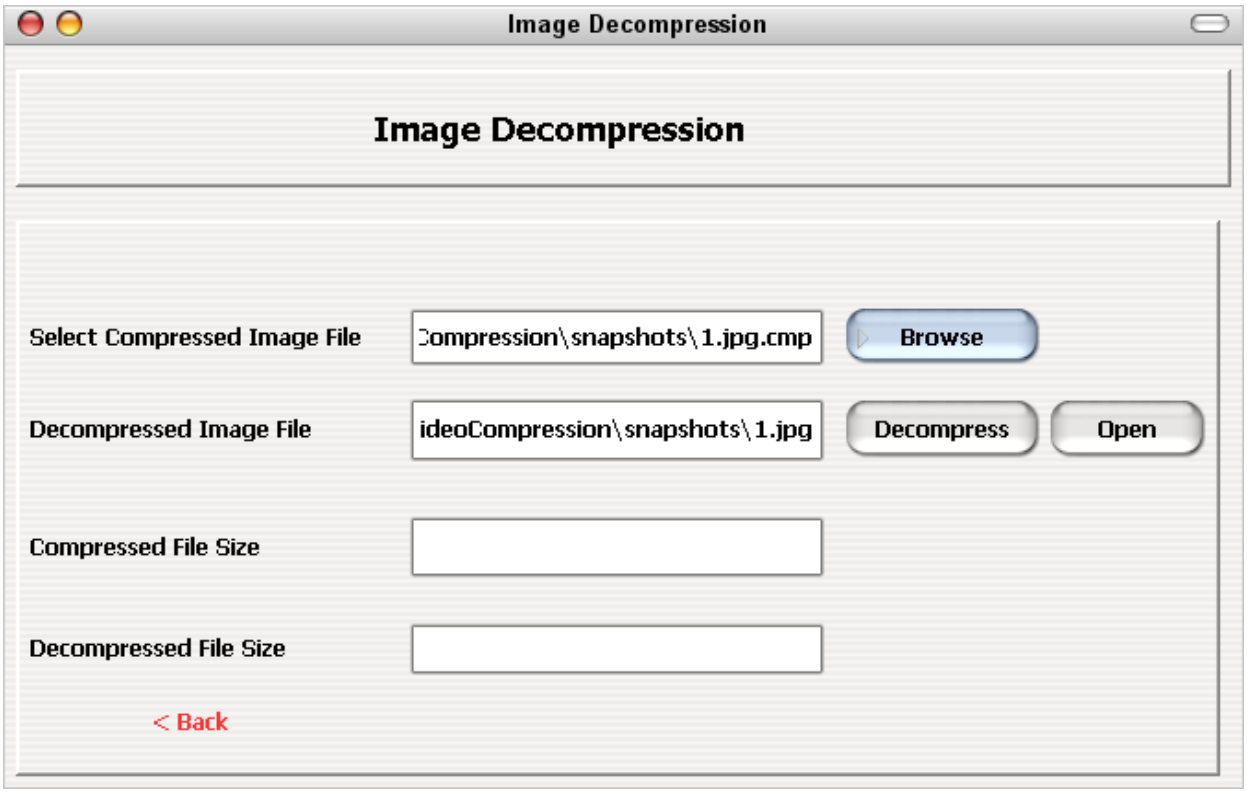


Image Decompression
Image Decompression
Select Compressed Image File
ompression\snapshots\1.jpg.cmp
Browse
Decompressed Image File
ideoCompression\snapshots\1.jpg
Decompress
Open
Compressed File Size
Decompressed File Size
< Back

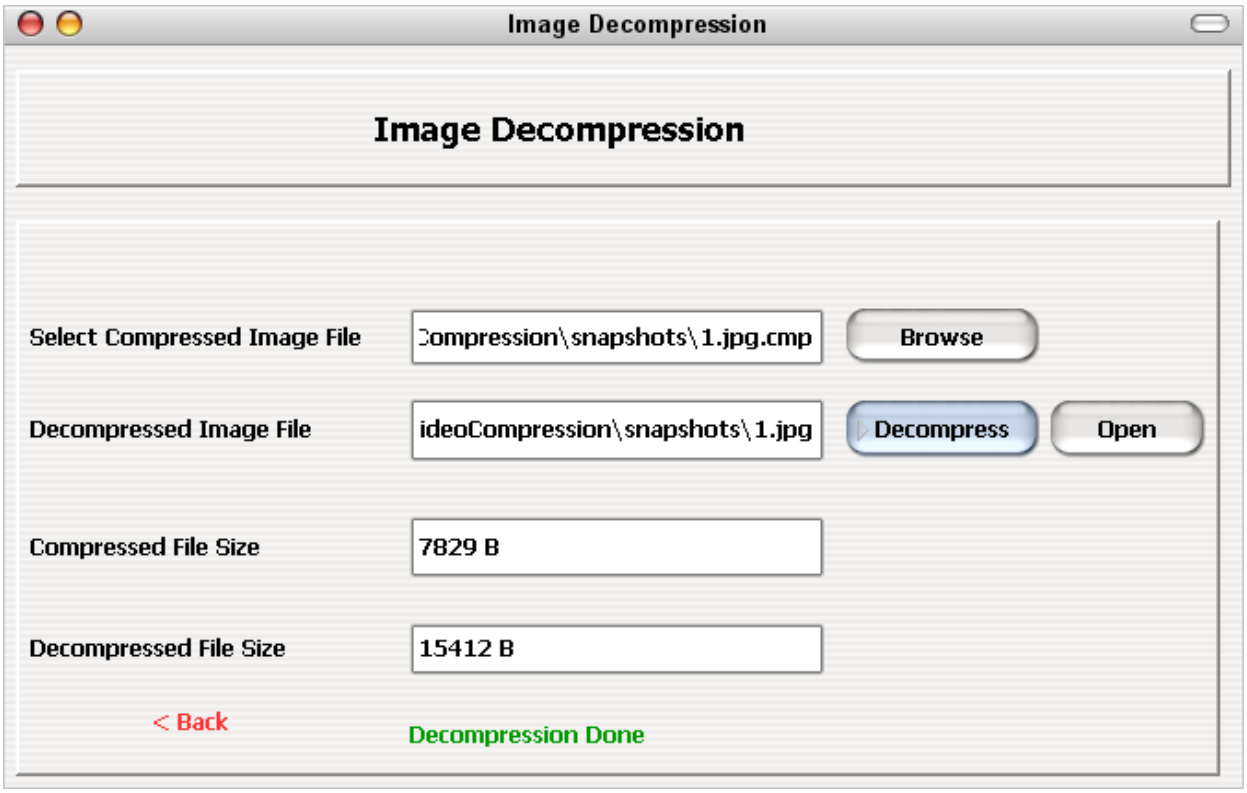


Image Decompression
Image Decompression
Select Compressed Image File
ompression\snapshots\1.jpg.cmp
Browse
Decompressed Image File
ideoCompression\snapshots\1.jpg
Decompress
Open
Compressed File Size
7829 B
Decompressed File Size
15412 B
< Back
Decompression Done

REFERENCES

[1] A. Mackin, F. Zhang, and D. R. Bull, “A study of subjective video quality at various frame rates,” in IEEE International Conference on Image Processing, Sept 2015, pp. 3407–3411.

[2] A. Mackin, M. Afonso, F. Zhang, and D. R. Bull, “A study of subjective video quality at various spatial resolutions,” in IEEE International Conference on Image Processing, 2018.

[3] M. Shen, P. Xue, and C. Wang, “Down-sampling based video coding using super-resolution technique,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 21, no. 6, pp. 755–765, 2011.

[4] G. Georgis, G. Lentaris, and D. Reisis, “Reduced complexity superresolution for low-bitrate video compression,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 26, no. 2, pp. 332–345, 2016.

[5] R. Wang, C. Huang, and P. Chang, “Adaptive downsampling video coding with spatially scalable rate-distortion modeling,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 24, no. 11, pp. 1957–1968, 2014.

[6] J. Dong and Y. Ye, “Adaptive downsampling for high-definition video coding,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 24, no. 3, pp. 480–488, 2014.

[7] Y. Li, D. Liu, H. Li, L. Li, F. Wu, H. Zhang, and H. Yang, “Convolutional neural network-based block up-sampling for intra frame coding,” IEEE Transactions on Circuits and Systems for Video Technology, pp. 1–1, 2018.

[8] Z. Ma, M. Xu, Y.-F. Ou, and Y. Wang, “Modeling of rate and perceptual quality of compressed video as functions of frame rate and quantization stepsize and its applications,” IEEE Transactions on Circuits and Systems for Video Technology, vol. 22, no. 5, pp. 671–682, 2012.

[9] Q. Huang, S. Y. Jeong, S. Yang, D. Zhang, S. Hu, H. Y. Kim, J. S. Choi, and C.-C. J. Kuo, “Perceptual quality driven framerate selection (PQD-FRS) for high-frame-rate video,” IEEE Transactions on Broadcasting, vol. 62, no. 3, pp. 640–653, 2016.

[10] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super resolution using deep convolutional networks,” IEEE Transactions on pattern analysis and machine intelligence, vol. 38, no. 2, pp. 295–307, 2016.

[11] J. Kim, J. Kwon Lee, and K. Mu Lee, “Accurate image super resolution using very deep convolutional networks,” in IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1646–1654.

[12] F. Zhang, A. Mackin, and D. R. Bull, “A frame rate dependent video quality metric based on temporal wavelet decomposition and spatiotemporal pooling,” in IEEE International Conferenceon Image Processing, Sept 2017, pp. 300–304.

[13] A. Mackin, M. Afonso, F. Zhang, and D. R. Bull, “SRQM: a video quality metric for spatial resolution adaptation,” in Picture Coding Symposium (PCS), 2018.

[14] M. Afonso, F. Zhang, A. Katsenou, D. Agrafiotis, and D. Bull, “Low complexity video coding based on spatial resolution adaptation,” in IEEE International Conference on Image Processing. IEEE, 2017, pp. 3011–3015.

LIST OF FIGURES

LIST OF ABBREVIATIONS

DCT	Discrete Cosine Transform
JPEG	Joint Photographic Experts Group
MPEG	Motion Picture Experts Group
IDCT	Inverse Discrete Cosine Transform
RLC	Run Length Coding
RGB	Red,Green,Blue
MOS	Mean Opinion Scores
DBC	Down Sampling Based Coding
SR	SuperResolution
HEVC	High Efficiency Coding
PSNR	Peak Signal Noise Ratio
IDE	Integrated Development Environment
JVM	Java Virtual Machine
RMI	Remote Method Invocation
RTT	Round Trip Time
MV	Motion Vector