SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING
MYSORE-570006
Department of Computer Science and Engineering

Data Compression Project
Mars Video Compressor & Decompressor

Submitted by

Name	Roll No	USN
Asif.M	11	4jc10cs019
Akshay Shetty	05	4jc10cs009
Parashivamurthy.C	69	4jc11cs408
Raghu.D	72	4jc11cs414

Guidance of
Smt. Trisiladevi C.Nagavi
Asst. Professor
Dept of CS&E,SJCE Mysore-06.

Affiliated to
Visvesvaraya Technological University
Belgaum

Contents

1 Abstract

Todays generation uses a broad range of applications involving high quality videos and audios. Since uncompressed videos need a very high bit-rates consuming a lot of bandwidth. This necessitates the need for video and audio compression exponentially from its original uncompressed format and the encoded format should be capable of decoding the uncompressed file exactly as it is (lossless) or almost as it is with some tolerable error (lossy).

2 Problem Definition

Mars video compressor is a program which compress a given video file of any format to a compressed file, of format '.mat' which is used to send via network or used to store in the memory. Using the compressed file, we can regenerate the given video file with some tolerable loss of data (lossy compression).

We are adopting the JPEG (Joint Photographer Expert Group) image compression technique on each frame of the video given and compressing the each frames and storing it as an intermediate file, and using this file we inverse all the module which is applied in the compression part and will get 'n' frames of the video and we convert these frames to a video.

Scope: Mars video compressor is not typical video compressor but it compress the video frames and restore it, i.e., a video is split to all its frames and the compression is employed on its each frames separately and stores to a '.mat' file. So the compression ratio is dependent on size of the individual frames.

Since we are using MATLAB are program is dependent upon MATLAB compiler but it is OS platform independent.

3 Software Requirement Specification

3.1 Functional Requirements

3.1.1 Compression Phase

• Program should accept any type of video file and should compress it
• Exception should be handles while reading an input video file path and also input file type.
• All the intermediate Procedure and its progress to be displayed so that user should not feel that the program is struct or not responding to his input file
• Output of the compression phase should be displayed. i.e., path of the intermediate file should be displayed.
• Compression ratio should be calculated properly and should be displayed at the end of compression.

3.1.2 Decompression Phase

• Program should accept the proper input file and decompress it.
• All the Exceptions to be handles while reading the input file path and file type.
• Progress of Decompression should be displayed.
• Result of the Decompression should be a video file.
• Resultant video should be saved in the Hard drive of the system so that it can be played from other video players and can be used later.
• Program should give user a choice to Play the Result video.
• Result video should be in any supported formate, use preferable formate such as .avi
• Frame-rate of the Resulting video should be same as the original video.
• Height and Width of the output video frames should be same as the original video frames.

3.2 Non-functional Requirements

• Program should be interactive with facilitated GUI.
• Ease of use should be flexible.
• Presentation should be good and easy to understand.
• Program is open source hence program should be written with the comments explaining the function properly.
• Documentation is done using IEEE standard with 12pt font size,oneside paper style and A4 Paper size.

3.3 Hardware and Software Requirements

3.3.1 Hardware Requirements

A System which supports the following:

- **Processor:** Intel Pentium based System.
- **Processor speed:** 250MHz to 3.0GHz.
- **RAM:** 128MB to 8GB.

3.3.2 Software Requirements

- **Operating System:** Windows, Linux and Mac OS.
- **Matlab:** Matlab R2008b and above are supported.
- **PDF Reader:** To view the report.
- **LaTeX:** To prepare the documentation.

4 Design Documentation

4.1 Program Orientation

Since the program is developed on the Matlab software and Matlab supports both Procedure oriented and Object Oriented Programming. Our Mars video Compressor & Decompressor uses both procedure oriented and Object oriented programming patterns.

Object Oriented: Reading of video file is done using videoreader module which stores the video formate in class type. i.e, the video read is stored as a class object of 'video Reader' class. and also in Decompression part the video is regenerated from the compressed file is stored in using video writer class object.

Procedure Oriented: Most part of the program uses module oriented that is we call procedure in our main program. There are different modules used in the main code. Some are built-in and soome are user defined modules. Modules are stored as different file in Matlab current directory with the extension of '.m'.

Uer define modules used in the programs are ZigzagMx(), RunlengthEncoding(), AntiZigzagMx() and RunlengthDecoding(). And many more predefined modules are used such dct2(), idct2(), rgb2ycbcr(), ycbcr2rgb() and etc.,

4.2 Context Diagram

In Fig.1 the flow of the video file across the different modules of Software is shown. As explained earlier the video file is split into its frames and stored in a directory and each images are extracted from this directory and performed different operation modules such as DCT and Encoding and its reconstructed to frames back in other end of the module and stored in a structure and these frames are used to regenarate the video. The video is outputed in '.avi' formate.

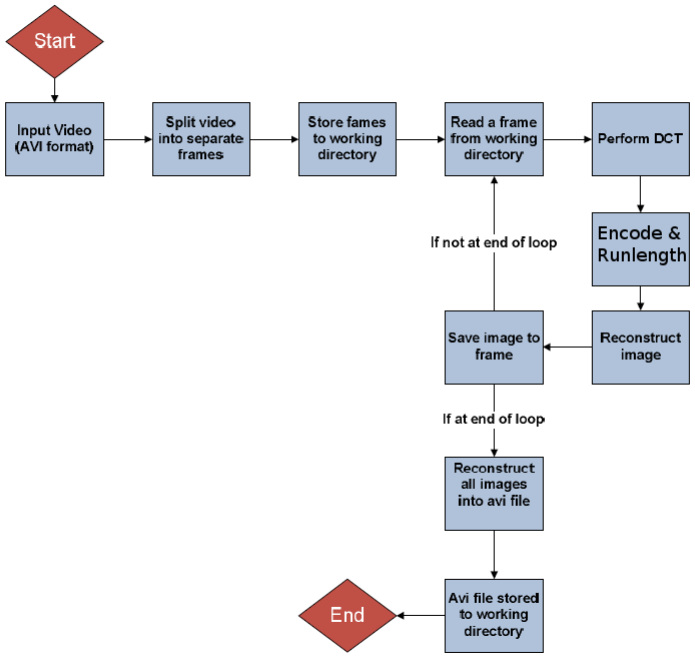

Figure 1: Context Diagram of compression procedure for a Frame

Phase-I : In phase-I i.e., Compression part of the software, Each frames are considered and converted into YCbCr formate where Y gives the Luminous and Cb & Cr gives the Chrominous part of the image. These are considered in macro blocks and send to the different modules such as DCT function, Quantization, Zigzag traversal and Runlength Encoding and the data is Stored as "Intermediate file" i.e., our Compressed file.

Phase-II : In phase–II i.e., Decompression part of the softwar, Encoded Runlength is loaded and all the information about each frames are extracted and these data are operated in different inverse modules such as Runlength Decoding, Anti-Zigzag Traversal and inverse DCT function and we obtain a YCbCr formated frames and these are converted back to RGB and stored in a Structure and finally formated to a avi video file.

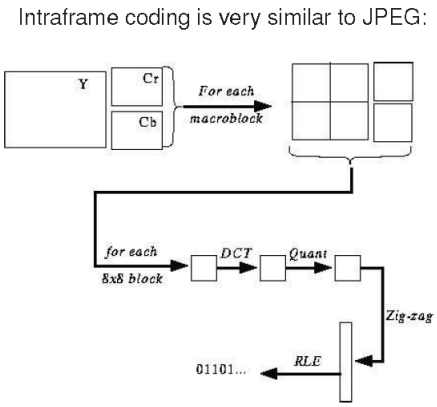

Figure 2: Flow of modules in phase-I

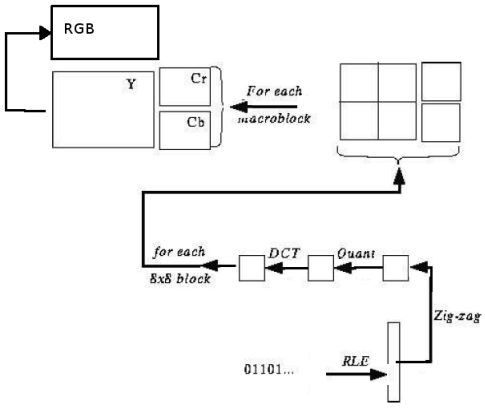

Figure 3: Flow of modules in phase-II

5 Implementation

Implementation should be straight forward. Program should accept the input, The input accept should be error free and if user enters an input which causes an error in the system that should be specified. And the input should be processed as explained in context free diagram and shoould satisfies each and every Functional requirement explained in this document and data structure should be effective enough for the dynamic processes created. Display the results in a good presentation form so that user has no ambiguity in the result obtained. As problem defined the program should be well defined to take any type of video and compress it and when Decompressed it should result the same video file with .avi formate.

5.1 Algorithm

5.1.1 Compresssion Phase

• Reading a video, which involve storing all the attribute of the video such as Duration, Number of Frames, Width, Height, File-format, Frame rate, File name and Path, which are used in other modules of program.
• Splitting the video into frames and storing them in a directory so that we can extract the frames individually for our later use, each frames are stores as an image of format PNG or JPG.
• By taking an individual frames image we apply the following modules.
• Each RGB frames are converted to YCbCr format where Y indicate luminous part of the image and Cb and Cr indicate two different chrominous part of the image.
• Discrete cosine transform (DCT) is applied on all the mentioned blocks of the image that is Y,Cb and Cr block of the image so has to get maximum zeros in the matrices.
• Quantization by value is applied on the resultant matrices after DCT module.
• Zigzag traversal is employed on the resultant matrices of quantization which intern gives an arrays for each matrix, which contains frequently repeated zeros in large numbers.
• Run-length encoding is employed on the arrays obtained after zigzag which gives two column matrices, where 1st column represent the element and 2nd column is used to store the run length of corresponding element. These two column matrix reduces the array length drastically because of large number repeated zeros.

• Storing all Run-length encoded matrices of each frames of the video to a '.mat' file which is the compressed file, that has to be used in decompressed phase to get the encoded video back.

5.1.2 Decompresssion Phase

• Load the '.mat' file which is given as the input.
• Extracting run-length encoded matrices representing Y,Cb and Cr part of all the frames of the video and during extraction we also extract the useful parameter of the video such as number of frames and height and width of the frames.
• Inverse run-length or run-length decoding is applied on these extracted two column matrices which results in an array for each matrices.
• Inverse zigzag or anti-zigzag employed on the arrays obtained in the previous module, this result to a matrix which is of the size equal to the size of the original frame.
• Inverse DCT is applied on the matrices which results Y, Cb and Cr matrices of the frames.
• Convert Y, Cb & Cr to RGB to get the original frames back.
• Writing an video into the file using all frames obtained.
• Playing video on user demand.

5.2 Data Structure

Mars Video compressor & decompressor takes the video file as input, result the compressed intermediate '.mat' file and reconstructed video file as output. To access the video in Matlab we use **VideoReader** class to read video and store it as the variable and **VideoWriter** class to Write the video file. Since compression is employeed on the each frames we consider each frame as image and images are stored in the form of **Matrix**. Matrix of different type are used in the program to store different intermediate data. **uint8** is used while considering the RGB image and all the intermediate data matrix are stored in **double** type. and Compressed Runlength encoded data is stored as **single** type. To store the different length of Runlength encoded data or decoded data of several image it is not feasible to use matrices or array vector hence we use **Structure**, which is powerfull data structure supported by Matlab. To store the file name and path of the different file **String** type should be used so that any string operation can be done easily.

5.3 Error Handling

Durring implementation many errors can be occured and from the requirement analysis we have a list of few errors to be handled durring implementations.

Errors by User's Input:

• Error to be handled when user select file path which does not exist. Proper Error should be displayed to user.
• When user select input file which is not the suitable file type for the program then Error messages to be displayed. and should ask user to select the desired type of file.
• Exception should be handle when the wrong input file is try to load or open in the program.

Errors while storing the Intermediate Data:

• Error may occured when the Encoded data is of different length for different frames and programmer is trying to store them using array or matrix.
• To eradicate the above error program can use Structure as the data type.
• Errors to be handled while storing intermediate compressed data for the video.
• Only Runlength information should be stored as the intermediate file and should use the data type which consume less bits to store the data.

Errors on Result video:

• Exception can occured when the video file given is corrupted and unable to open in the program.
• Video file to be stored as the result of the decompression with no error or with no corruption.
• Error while playing the resultant video file to be handled.
• Any error durring the execution of the program should be reported to the user with proper messages.
• Program must be error free as much as possible so use exception handling on the blocks where error can occur.

6 Result analysis

6.1 Snap shots of Output

Figure 4: Selecting Compression or Decompression Option

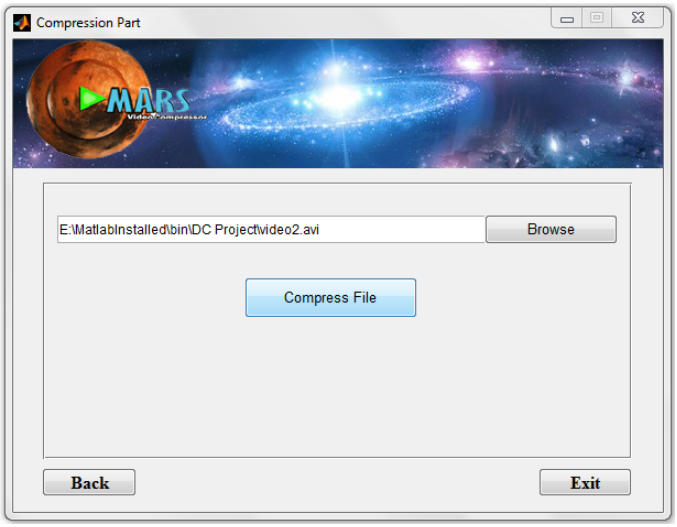

Figure 5: Browse The Desired Video file

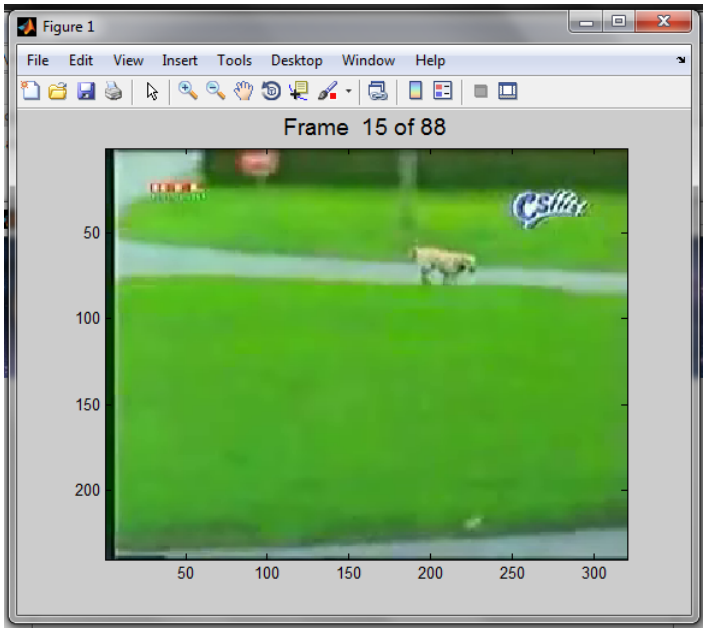

Figure 6: Displaying the Splitting of Frames

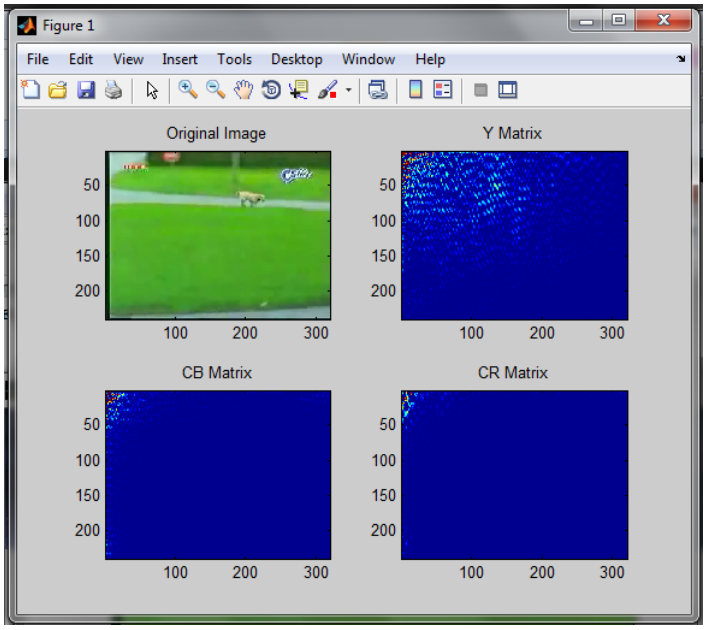

Figure 7: Displaying the Different Images Processes

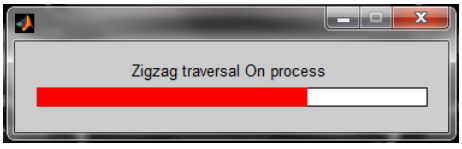

Figure 8: Wait bar for Zigzag Encoding Process

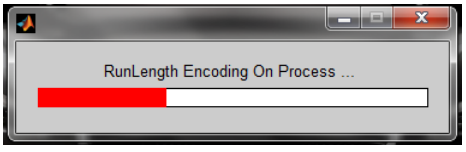

Figure 9: Wait bar for Runlength Encoding Process

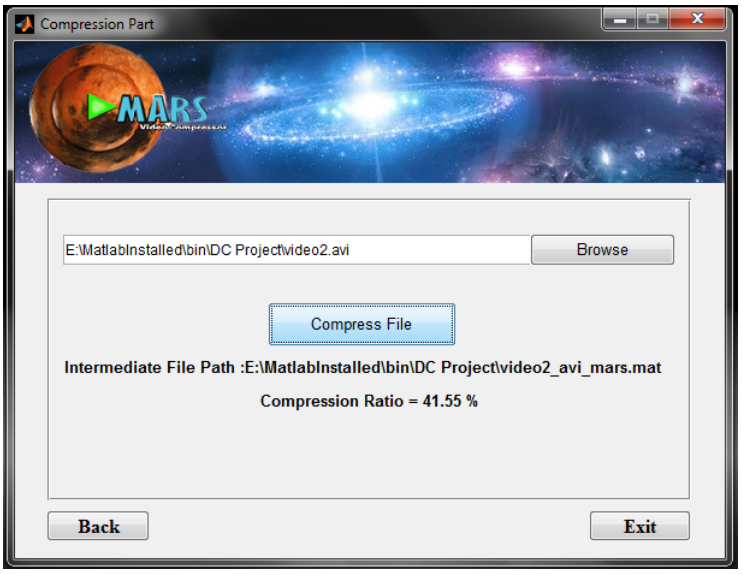

Figure 10: Displaying the Compression Ratio and Compressed File Path

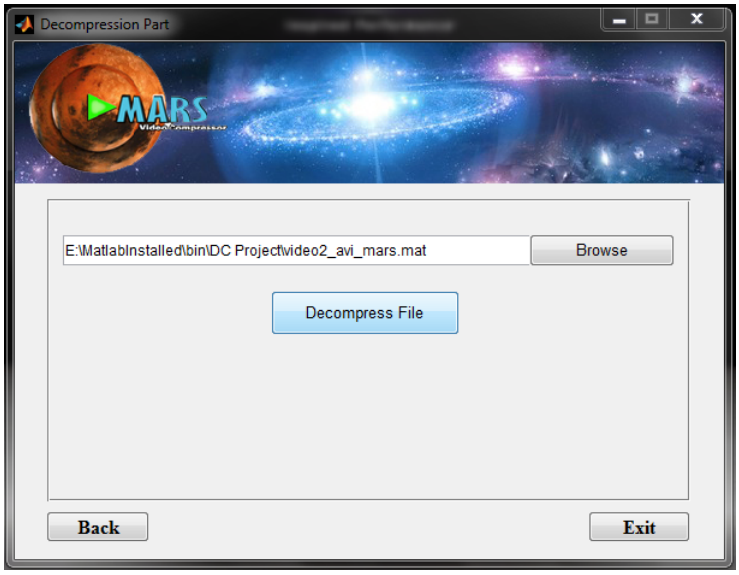

Figure 11: Browsing the compressed file for Decompression

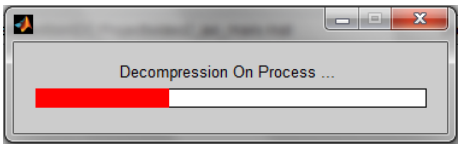

Figure 12: Wait bar showing Decompression Part

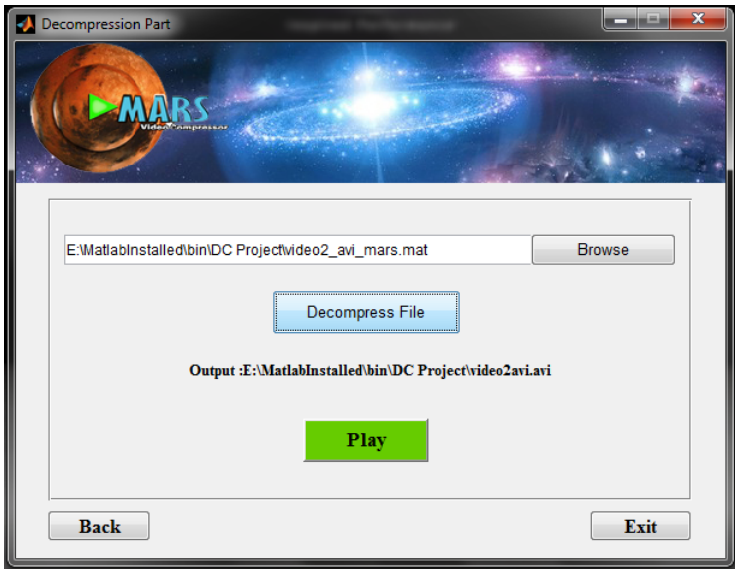

Figure 13: Displaying the Decompressed Video File Path and Video Play Option

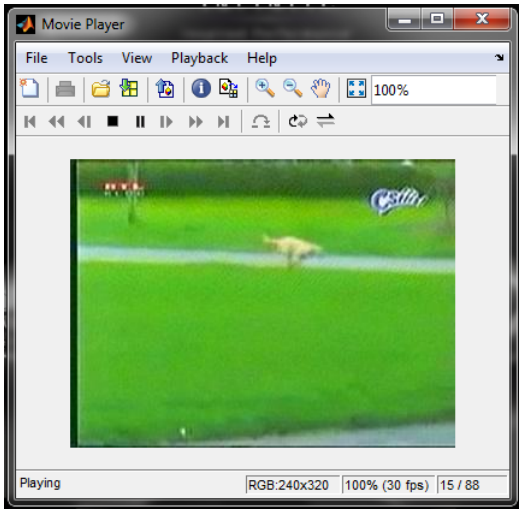

Figure 14: Matlab video player playing Decompressed File

6.2 Analysis on Output

In Fig.5 user is selecting a video file from his computer and video is under goes different procedures and all the operation results are displayed in further figures. After compression of the given video file the compressed file is displayed to the user with its compression ratio (In fig.10).

When user want to decompress the compressed file he selects the compressed file from the system and give it as the input to the decompressor phase and the decompression module resluts to a video file which is same as the original video file used to get the compressed file which user selected in decompression phase. The Resultant video file path is displayed to the user from which user can browse through his system and can play the video file outside our program i.e., using different video players. Even our program gives user an option to play the resultant video file using Matlab's built in video player.

We have collected few sample videos of different properties to check the compatibility of our program on these video files. The result and comment on the output obtained from these videos are written below with the file properties of the different video files we have used yet.

video.mpg:
- Duration: 2.08 Sec
- Resolution: RGB 320 x 240
- No. of Frames: 32
- Objective: To Test the compression on a moderate scattered pixels
- Result: **Passed**
- Comment: *Compression ratio is comparitively less.*

video1.mp4:
- Duration: 1.03 Sec
- Resolution: RGB 1280x720
- No. of Frames: 31
- Objective: To Test the compression on a higher resolution.
- Result: **Passed**
- Comment: *Compression Ratio is good but the time taken is more during compression.*

video2.avi:
- Duration: 3.52 Sec
- Resolution: RGB 320x240
- No. of Frames: 88
- Objective: To Test the compression on a modarate number of frames.
- Result: **Passed**
- Comment: *Compression Ratio is Good upto 70%*

video3.avi:
- Duration: 4.24 Sec
- Resolution: RGB 320x240
- No. of Frames: 106
- Objective: To Test the compression on a more number of frames.
- Result: **Passed**
- Comment: *Compression Ratio is moderate upto 40% and time is comparitively moderate.*

7 Conclusion

The result of Mars video compressor & Decompressor is shown above using figures and theorital analysis.Our Program is working fine for all the different type of video files and compressing it to a file of extension '.mat' and using this '.mat' file we can successfully regenerate the original video file with some tolerable loss of data (lossy compression).

Mars compressor is upto the all Requirement speficified in this document and it is reviewed and tested with some sample video file and all the files are showing the positive result on this software.

References:

Book:

Data Compression - Khalid Sayood

Websites:

1. http://www.mathworks.in/help/matlab/ref/videoreaderclass.html
2. http://www.mathworks.in/help/images/performing-distinct-block-operations.html
3. http://www.mathworks.in/matlabcentral/newsreader/view_thread/300238
4. http://www.cs.sfu.ca/CourseCentral/365/mark/material/notes/Chap4/Chap4.2/Chap4.2.html
5. http://en.wikipedia.org/wiki/JPEG
6. http://www.dspguide.com/ch27/6.htm
7. http://www.mathworks.in/help/matlab/ref/videowriterclass.html