A

REPORT ON

TERM PROJECT ON COMPRESSION TECHNIQUES

MULTIMEDIA COMPUTING

CS F401 / BITS F447Submitted By

PRANGAV SINGHAL 2014A8PS332P

GAURANG BANSAL 2014A7PS128P



First Semester 2016-17  
Birla Institute of Technology & Science, Pilani

**TABLE OF CONTENTS**

**No. Title**

1. INTRODUCTION
2. SUMMARY
3. IMPLEMTATION
4. RESULTS
5. REFERENCES

**INTRODUCTION**

Image data is an important part of any multimedia system. To interpret image data, it is stored in discrete colour values. Many formats are followed to represent image data. Few of them are RGB, YCbCr, HSV, YUV, YIQ etc.

**RGB**

In this format, image data is stored in three different streams of discretised intensity values of the Red, Green and Blue colours. The range of the colours depend on the quantization value and also the number of bits available per stream.

**YCbCr**

In this format the three streams used are of luminance and chrominance values instead of colour intensity values. Y is the luminance value, Cb and Cr are chrominance values.

**HSV**

HSV stands for Hue, Saturation and Value. This format follows a cylindrical coordinate system to define the colour values. Hue is the value which defines the colour, Saturation defines the intensity of the colour and value defines the brightness or lightness of the colour.

**YUV and YIQ**

These are colour spaces used by different video formats. It is basically a transpose of the RGB colour space. The RGB vector for a particular colour value is pre-multiplied by a 3X3 matrix to get the YIQ and YUV values.

These formats define different ways to represent the colour values that are to be displayed to the viewer. However, image files can be very large to be stored in raw form. To overcome this problem several compression techniques are employed.

Different colour models have some effect on the compression. Each colour model have different set of colours available and hence the quantisation value is also different. For a smaller quantization number, less compression is achieved, but quality is better. For a larger, compression achieved is better, but the quality is decreased.

**Summary**

Our system Compares different compression ratios and Different image sets that can be built using different Colour model. It is Colour model integration where all different dimension of each colour model can be varied for finding out the corresponding Results.

It also demonstrates the effect of the difference in the Run Length code for different type of scanning of the Direct cosine transforms for the images. The scanning patterns used are: zig-zag scanning, scanning along y axis, scanning along x axis.

The system also implements the effect of different colormap and how the number of colors impact the compression of the image. We have used all the standard maps.

So basically our program is a fully functioning image processing compression unit which is very efficient to calculate the different compressions that can be achieved using

* Different Colour Maps
* Different types of Color Models
* Different Compression on different frames of a model
* Run Length Encoding
* Quantisation
* Number of Color used
* Types of Images

All these parameters have been kept in mind to have this implementation in complete.

**Implementation**

The system has been implemented on MATLAB. It consists of the following files:

* **MyCompress.m**

This is the driver functions which accepts the path of the image file as an argument and gives the user a choice between different colour models and compression to choose from.

* **CompressionRatio.m**

This functions calculates the compression ratio of different images by the taking a ratio of their respective file sizes.

* **RGBCompr.m, YIQCompr.m, YUVCompr.m, HSVCompr.m, YCBCRCompr.m, NTSCCompr.m, GrayCompr.m**

These files convert the image to the respective colour spaces and split them into different streams for compression.

* **RedFrame.m, Greenframe.m and BlueFrame.m**

These files the main compression functions. For the respective streams, first the direct cosine transform is taken, then it is quantized according to the quality factor accepted from the user. **Quantization is the main lossy step in this process**. Then, this is decoded by inverse DCT and saved as a different image file.

* **RedFrame\_2.m, Greenframe\_2.m and BlueFrame\_2.m**

These files are similar to the other compression functions. However, these replace the lower right quarter of the DCT obtained for the image with zeroes to gain a higher compression. These files also demonstrate the benefit of this action by employing Run Length Coding using different types of scans: zig-zag scan, scan along x axis, scan along y axis.

* **colormap1.m**

This file uses the colormap function of MATLAB to demonstrate a compression technique by limiting the number of colours available. It uses different colormaps available with MATLAB and shows the compression ratio.

* **CodingScheme.m, zigzag.m and runlength.m**

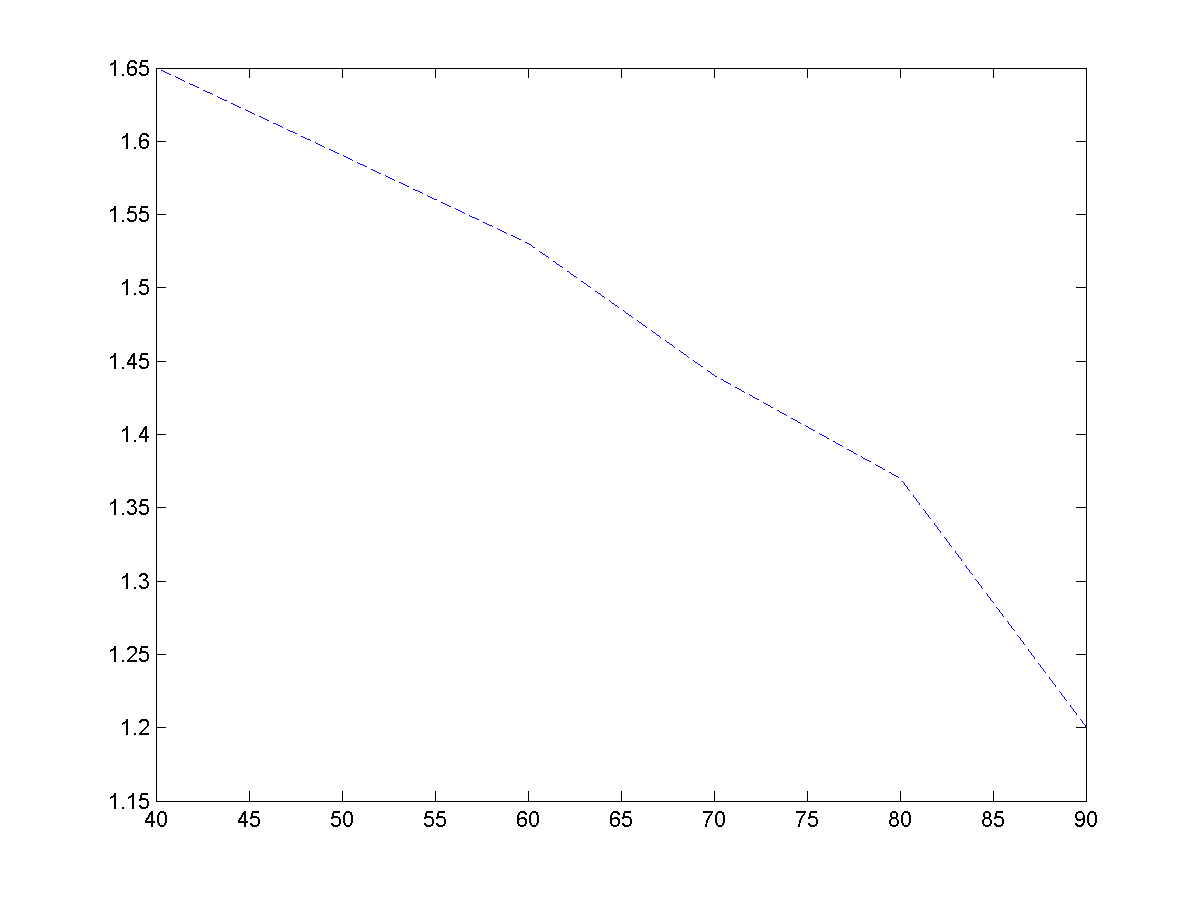
This file uses different way of coding bits an image which could provide us with different levels of compressions at a given stage of image.

**Results**

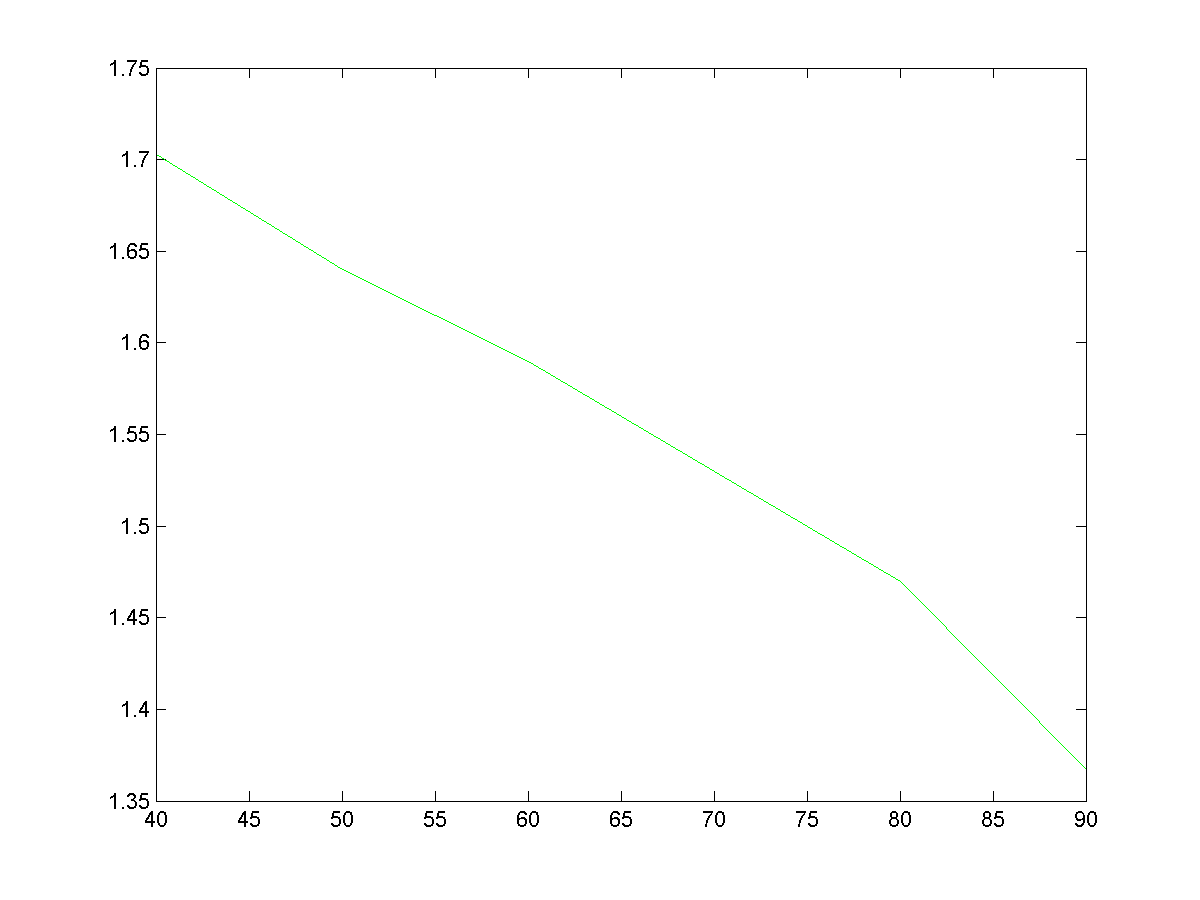
The projects clearly shows that the compression of an image decreases as the quality factor desired is increased.

The following shows how the compression of image varies as we increase the components of a particular dimension. Graph is **Compression Ratio vs Quality of Corresponding Frame**

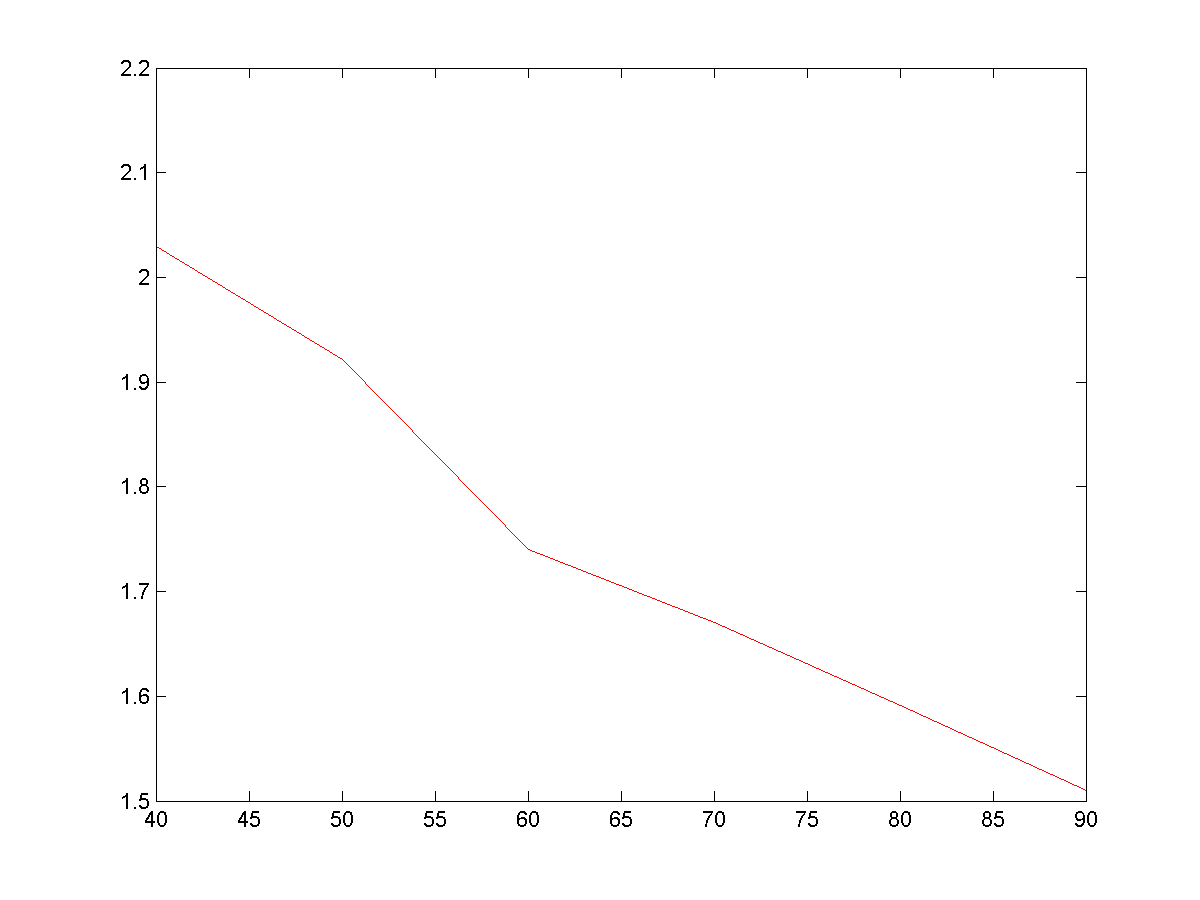
Changing Blue Frame Quality (Red and Green are kept Constant).



Changing Y Frame Quality (Cb and Cr are kept Constant).

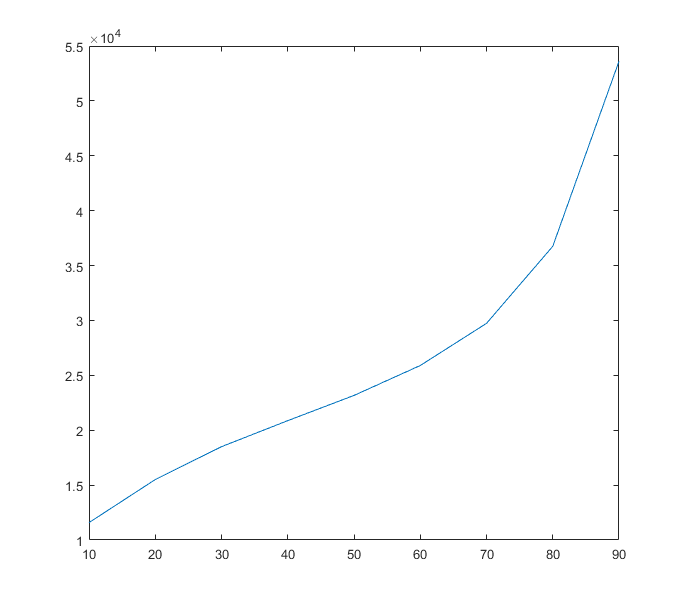


Changing Red Frame Quality (Blue and Green are kept Constant).

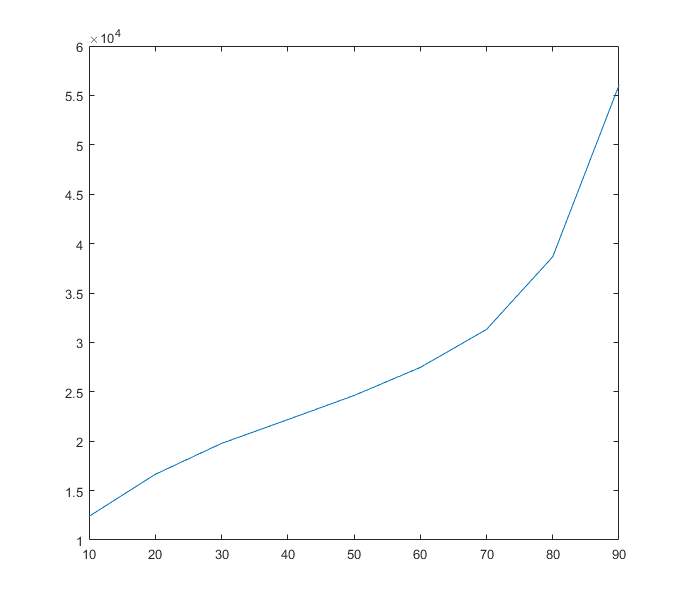


The comparisons for different scanning clearly show that the zig zag scanning of the DCT matrix results in the maximum compression by resulting in a smaller Run Length Code.

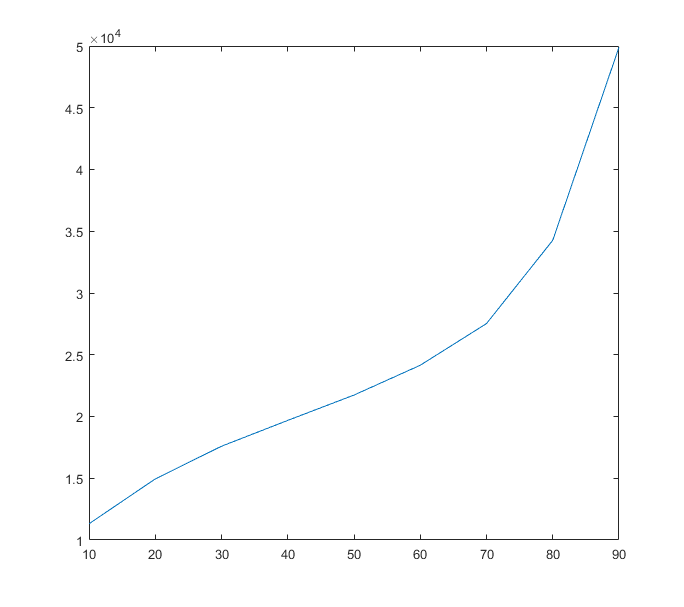
The following shows how the length of the run length code varies as we increase the quality factor of a particular domain for different scans for the red frame of Lenna.png.

Graphs are **Code Length (y axis, in order of 10000) vs Quality Factor (x axis)**

**Zig Zag scanning**



**Scanning along Y axis**



**Scanning along x axis**

**References**

* <https://in.mathworks.com/matlabcentral/fileexchange/38524-jpeg-compression-for-grayscale-images?s_tid=srchtitle>
* <https://in.mathworks.com/matlabcentral/fileexchange/27078-zig-zag-scan/content/zigzag.m>

### [Digital Image Processing Using MATLAB](http://users.nik.uni-obuda.hu/vamossy/GepiLatas2007/Segedlet/digital%20image%20processing%20using%20matlab%20(gonzalez).pdf)

### Still Image and Video Compression with MATLAB by S. Thyagarajan

* Research paper :- http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7574988