Image Compression in MATLAB

Tamanna Gaur, Aakriti Khanna

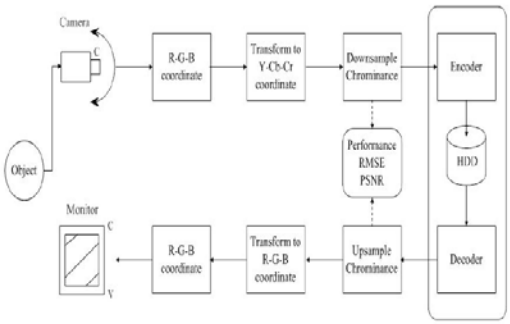
Abstract— In recent years, the development and demand of multimedia product grows increasingly fast, contributing to insufficient bandwidth of network and storage of memory device. Therefore, the theory of data compression becomes more and more significant for reducing the data redundancy to save more hardware space and transmission bandwidth. In computer science and information theory, data compression or source coding is the process of encoding information using fewer bits or other information-bearing units than an un-encoded representation. Compression is useful because it helps reduce the consumption of expensive resources such as hard disk space or transmission bandwidth.

Index Terms— Compression, MATLAB modeling, bandwidth encoded, redundancy.

1 Introduction

Image compression is an application of data compression that encodes the original image with few bits. The objective of image compression is to reduce the redundancy of the image and to store or transmit data in an efficient form. Fig 1.1 shows the block diagram of the general image storage system.

Image compression means reducing the size of the image or video file without downgrading the quality of the f perceived as an unsatisfactory image . For images anc the reduction in the file size is achieved by removin dundancies or repetitions involved in the image or v and preserving as much original information as pos this project, we have used Matlab to demonstrate ima pression of a 640 X 480 24-bit per pixel JPEG color i work on various aspects of image processing such as and displaying the images using imread and imsh mand. Converting the given image YCbCr color spa sampling and up-sampling using linear interpolation or column replication and converting the image into l mat and measuring the MSE values between the orig reconstructed images. All these mentioned aspects ar when it comes to image compression as these are t operations that would be performed on a given imag is to be compressed.



2 Procedure Section

2.1 Read and display the image in MATLAB along with its R, G and B bands

This is the basic operation whose purpose is to read the image file and display the same. We have made use of 'imread' command to read the original image Landscape.jpg and 'imshow' command to display the same image. Also the use of 'subimage' command has been made to depict the scale or axis (X and Y) for the image. In order to display the R, G and B components separately, for instance, to display the Red component, we make the G and B components zero in the original image (:,:,2:3) = 0 such that only the Red component of the image is displayed. The same procedure is then employed in a similar fashion to achieve the display of Green and Blue components respectively.

2.2 Conversion from RGB to YCbCr and displaying the individual Y, Cb, Cr components

'rgb2ycbcr' is the command used in Matlab to convert the original image in RGB format to its YCbCr form. In order to show the 'Y' component of the image, we have made the 'Cb' and 'Cr' components equal to zero such that the Matlab command being YCbCr(:,:,2:3) = 0 thereby displaying only the 'Y' component out of the three components. Similarly, the 'Cb' and 'Cr' components are individually displayed in the same manner by making the required changes to the above components

2.3 Sub-sampling the image using 4:2:0 and displaying both 'Cb' and 'Cr' bands:

Subsampling is basically a process of removing the pixels in an image by reducing the size of the image. The chroma samples from every even numbered rows and columns are removed by the Matlab code. Thus in this process, the 'Cb' and 'Cr' components with values equal to zero are discarded or ignored and only the remaining values of the 'Cb' and 'Cr' components with a non-zero value are displayed. This resulting image is the sub-sampled image with individual 'Cb and 'Cr' bands respectively.

2.4 Up-sampling using Linear Interpolation and displaying the 'Cb' and 'Cr' bands:

- *Tamanna Gaur is currently pursuing masters degree program in electronics and instrumentation engineering in YMCA University OF Science and Technology, India, PH-9711225827. E-mail:tamanna.tg@gmail.com*
- *Aakriti Khanna is currently pursuing masters degree program in electronics and instrumentation engineering in YMCA University OF Science and Technology, India, PH-9718873339. E-mail: aaki0502@gmail.com*

Each missing pixel value ('Cb' and 'Cr' component) in an odd numbered row is calculated as the average of its adjacent two pixels in the same row; whereas in an even numbered row, each missing pixel value is calculated as the average of the pixels in the same column from the two adjacent rows. So now, due to this process, we cannot calculate the average of the last row and the last column. In order to include this into the calculation, it is achieved by copying the second last row and the second last column into the last row and the last column respectively. We have used the Matlab command 'subimage' used to display the up-sampled 'Cb' and 'Cr' bands.

2.5 Up-sampling using Row-Column Replication and displaying the 'Cb' and 'Cr' bands:

For odd-numbered rows, the missing pixels are filled up by copying the pixels from the previous columns. Then, in order to complete the even-numbered rows, the odd-numbered rows are added to the subsequent even-numbered rows. In this process, the numbers of rows are doubled. The 'Cb' and 'Cr' bands are displayed using the 'subimage' command in Matlab

2.6 Conversion of the image into RGB format:

We made use of the Matlab command 'ycbcr2rgb' to convert the linearly interpolated and row-column replicated up-sampled (YCbCr) images into RGB format.

sampled (YCbCr) images into RGB format.

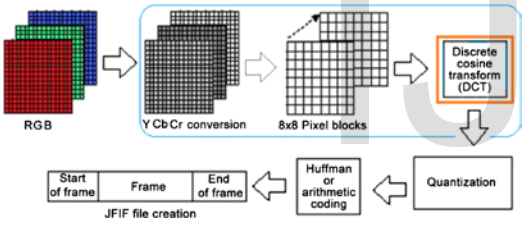


Fig 1.2 Conversion of image into RGB

MATLAB SIMULATION TOOL

The Image Processing Toolbox in Matlab is a collection of functions that extend the capabilities of the MATLAB's numeric computing environment. The toolbox supports a wide range of image processing operations, including:

- Geometric operations
- Neighborhood and block operations
- Linear filtering and filter design
- Transforms
- Image analysis and enhancement
- Binary image operations
- Region of interest operations

- MATLAB stores the read image in 3 x 2D matrix form. To read an image
  imfinfo('landscape.jpg')
  RGBImage = imread('landscape.jpg','jpg');
  imshow(RGBImage);
  size(RGBImage)
- To show the picture.
  figure() and subplot()
- To convert an image

YCbCr_Image= rgb2ycbcr (RGBImage);
Gray_Image = rgb2gray (RGBImage);

3 MATLAB PROGRAM CODE

```
% Name: Tamanna Gaur
% email: tamanna.tg@gmail.com

%% Project Cleanup
clc
clear all
close all

%% PROBLEM 1 - Read and display the image using Matlab

%Project location to read the image
Input = imread('C:\Users\sony\Desktop\Mini Project\Assignment 1\Landscape.jpg','jpg');
figure(1)
subimage(Inputtitle('Original RGB image')

%% PROBLEM 2 - Display each band (Red, Green and Blue) of the image file

figure(2)
Red = Input;
Red(:,:,[2 3]) = 0;   Green = Input;
Green(:,:,[1 3]) = 0;   Blue = Input;
Blue(:,:,[1 2]) = 0
% Display the original image and R,G & B Band Seperately
subplot(2,2,1)
subimage(Input);
title('Original RGB Image');

subplot(2,2,2)
subimage(Red);
title('Red Component');

subplot(2,2,3)
subimage(Green);
title('Green Component');

subplot(2,2,4)
subimage(Blue);
```

```
title('Blue Component');

%% PROBLEM 3: Convert the image into YCbCr color space

YCbCr = rgb2ycbcr(Input
figure(3)
colormap(gray
subplot(2,2,1)
subimage(YCbCr);
title('YCbCr Image');

%% PROBLEM 4: Display each band separately (Y, Cb and Cr bands)

subplot(2,2,2)
Y = YCbCr(:,:,1
subimage(Y);
title('Y Component');

subplot(2,2,3)
Cb = YCbCr(:,:,2);
subimage(Cb);
title('Cb Component');

subplot(2,2,4
Cr = YCbCr(:,:,3);
subimage(Cr);
title('Cr Component');

%% PROBLEM 5: Subsample Cb and Cr bands using 4:2:0 and display them

YCbCr2 = YCbCr;
YCbCr2(1:2:479,2:2:640,2:3) = 0;

YCbCr2(2:2:480,:,2:3) = 0;

figure(4)
colormap(gray);

SubCb(:,:) = YCbCr2(1:2:480,1:2:640,2);
SubCr(:,:) = YCbCr2(1:2:480,1:2:640,3);

subplot(1,2,1)
subimage(SubCb(:,:));
title('Cb component after 4:2:0 Subsampling');

subplot(1,2,2)
subimage(SubCr(:,:));
title('Cr component after 4:2:0 Subsampling');

%% PROBLEM 6.1: Upsampling using Linear Interpolation

% To fill blank row between every Cb, Cr 4:2:0 subsampled row
% To fill blank column between every Cb, Cr 4:2:0 subsampled column

YCbCr3(:,:,1) = YCbCr2(:,:,1);
YCbCr3(1:2:479,1:2:639,2) = SubCb(:,:);
YCbCr3(1:2:479,1:2:639,3) = SubCr(:,:);

YCbCr3(1:2:479,2:2:638,2:3) = (double(YCbCr3(1:2:479,1:2:637,2:3))
+ double(YCbCr3(1:2:479,3:2:639,2:3)))/2;

YCbCr3(1:2:479,640,2:3) = YCbCr3(1:2:479,639,2:3);

YCbCr3(2:2:478,:,2:3) = (double(YCbCr3(1:2:477,:,2:3)) +
double(YCbCr3(3:2:479,:,2:3)))/2;

YCbCr3(480,:,2:3) = YCbCr3(479,:,2:3);

% Display the upsampled Cb and Cr components
figure(5)
subplot(1,2,1)
subimage(YCbCr3(:,:,2));
title('Cb Upsampling - Linear Interpolation');

subplot(1,2,2)
subimage(YCbCr3(:,:,3));
title('Cr Upsampling - Linear Interpolation');

%% PROBLEM 6.2: Simple Row or Column Replication

YCbCr4 = YCbCr2;

YCbCr4(1:2:479,2:2:640,2:3) = YCbCr4(1:2:479,1:2:639,2:3);

YCbCr4(2:2:480,:,2:3) = YCbCr4(1:2:479,:,2:3);

% Display the upsampled Cb and Cr components
figure(6)
subplot(1,2,1)
subimage(YCbCr4(:,:,2));
title('Cb Upsampling - Row or Column Replication');

subplot(1,2,2)
subimage(YCbCr4(:,:,3));
title('Cr Upsampling - Row or Column Replication');

%% PROBLEM 7: Convert the images to RGB format

Input2 = ycbcr2rgb(YCbCr3);
Input3 = ycbcr2rgb(YCbCr4);

%% PROBLEM 8: Display the original and reconstructed image

figure(7)

subplot(2,2,1)
subimage(Input);
title('Original Image')
```

```
subplot(2,2,2)
subimage(Input2);
title('Upsampled by Linear Interpolation')

subplot(2,2,3)
subimage(Input3);
title('Upsampled by Row - Column Replication')

%% PROBLEM 10: Calculate MSE between the original and re-constructed images

double MSECB
double MSECR;
MSECB = 0;
MSECR = 0;

% To calculate the value in both the summations in the formula
for row = 1:1:480
  for col = 1:1:640
    MSECB = MSECB + (double(YCbCr3(row,col,2)) - double(YCbCr2(row,col,2))).^2;
    MSECR = MSECR + (double(YCbCr3(row,col,3)) - double(YCbCr2(row,col,3))).^2;
  end
end

% Divide by N*M (480 * 640) and display
disp('MSE for Cb component is: ');
MSECB = MSECB / (640 * 480);
disp(MSECB);

disp('MSE for Cr component is: ');
MSECR = MSECR / (640 * 480);
disp(MSECR);
```

V Results

The following results were deduced:

Below is the original RGB image along with its respective Red Green and Blue components.

Figure 1.3: Original image and its individual R, G and B bands

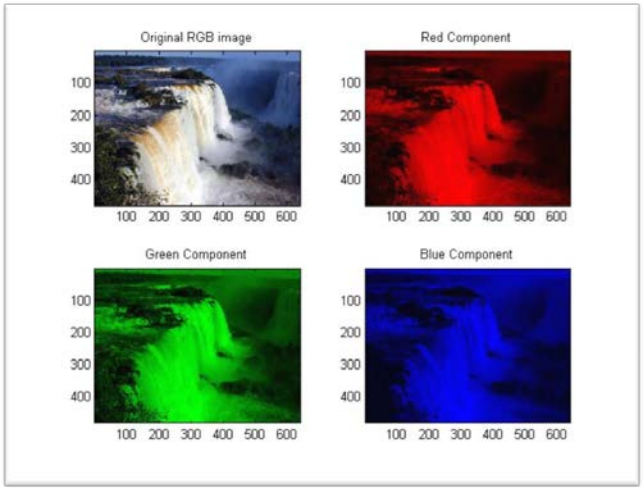
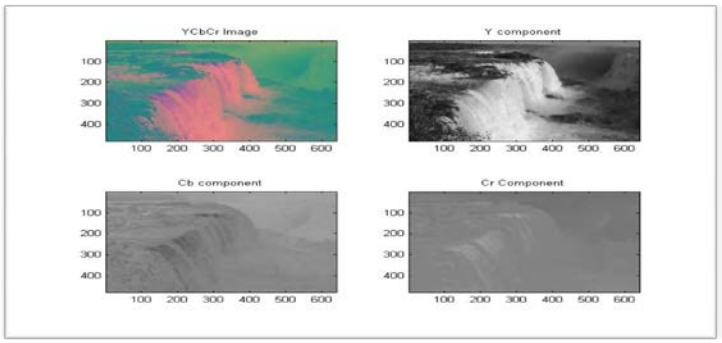


Figure 1.4: YCbCr image and its individual Y, Cb and Cr bands



There are two luminance samples for every Cb and Cr sample in 4:2:0 YCbCr but, there are no chrominance samples in alternative rows and columns.

Also a noticeable change is that the matrix size of the 4:2:0 sub-sampled Cb and Cr components is diminished by a factor of almost half i.e. to 320X280 compared to the individual Cb and Cr components whose matrix size is 640X480.

Hence a compression ratio of 2:1 is achieved in this process.

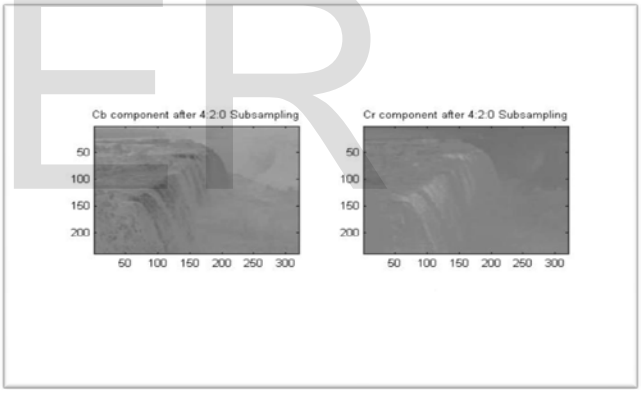


Figure 1.5: Cb and Cr components after 4:2:0 subsampling

Below are the up-sampled images of the individual Cb and Cr components using the linear interpolation method for Upsampling.

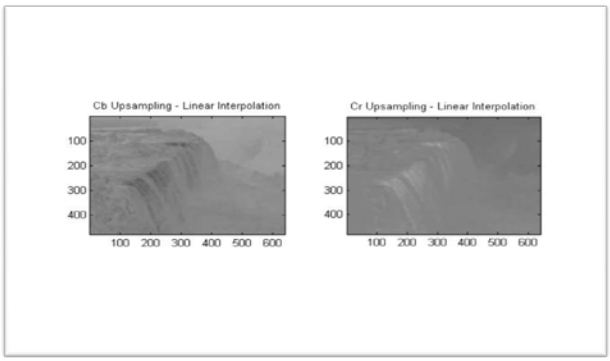


Figure 1.6: Cb and Cr upsampling using Linear Interpolation method

Following are the up-sampled images of the individual Cb and Cr components using row-column replication method of subsampling.

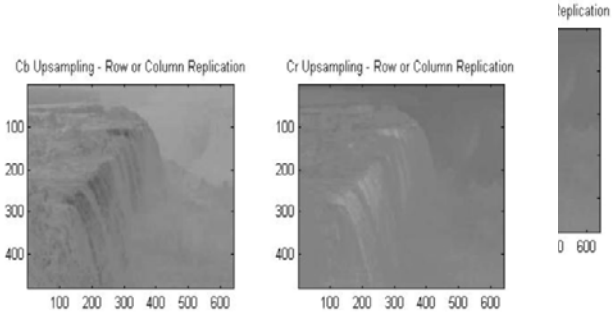


Figure 1.7: Cb and Cr upsampling using Row-Column replication method

Below are the reconstructed images (images restored from YCbCr) for Linear Interpolation and Row-Column replication respectively besides the original image.

Overall, the reconstructed image using Linear Interpolation has more contrast (the image is darker) compared to the image using Row-Column replication.

The areas like the vegetation (green shrubs/green color) is darker in the linearly interpolated image as compared to the row-column replicated image. The green colored stretch in the landscape at the top of the image and the golden-yellow colored water at the source of the waterfall is more dominant in the linearly interpolated image.

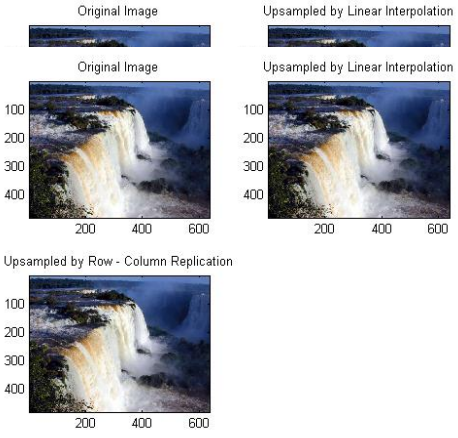


Figure 1.8: Reconstructed images using Linear Interpolation and Row-Column Replication

Mean squared Error (MSE) values

The Mean Squared Error (MSE) calculation between the original and the reconstructed image for the Cb and the Cr component in Matlab is found to be as follows:

MSE value for the **Cr** component is 11783

MSE value for the **Cb** component is 13746

4 Conclusion

The DCT-based image compression such as JPEG performs very well at moderate bit rates; however, at higher compression ratio, the quality of the image degrades because of the artifacts resulting from the block-based DCT scheme.

However, the current data compression methods might be far away from the ultimate limits. Interesting issues like obtaining accurate models of images, optimal representations of such models, and rapidly computing such optimal representations are the grand challenges facing the data compression community. Image coding based on models of human perception, scalability, robustness, error resilience, and complexity are a few of the many challenges in image coding to be fully resolved and may affect image data compression performance in the years to come.

In this project, we have conducted a few operations on a 640X480 24 bit/pixel JPEG color image using Matlab. The operations such as converting the image into YCbCr color space, sub-sampling using 4:2:0 and up-sampling using linear interpolation and row-column replication and converting the image into RGB format. However, the main focus of the project involved sub-sampling of the image using 4:2:0 and its reconstruction using linear interpolation and row-column replication methods. The 4:2:0 sub-sampling operation yields a compression ratio of 2:1. From viewing the reconstructed image and the original image, we can infer that the up-sampling operations have successfully reconstructed the original image.

Acknowledgment

I would like to take this opportunity to express my profound gratitude and deep regard to my project guide Dr. Anju Gupta for their exemplary guidance, valuable feedback and constant encouragement throughout the duration of project. Their valuable suggestions were of immense help in completing this project.

I would also like to give my sincere gratitude to all my friends and colleagues especially Ms.Aakriti Khanna who filled in the survey, without this the research work would be incomplete.

References

[1] S. Kumar, "Topic 1: Multimedia Communication Fundamentals"

[2] Haines, Richard F.; Chuang, Sherry L. (1 July 1992). The effects of video compression on acceptability of images for monitoring life sciences experiments (Technical report). NASA. NASA-TP-3239, A-92040, NAS 1.60:3239. Re-

trieved 13 March 2016. The JPEG still-image-compression levels, even with the large range of 5:1 to 120:1 in this study, yielded equally high levels of acceptability

[3] JPEG File Layout and Format

[4] N. Ahmed, T. Natarajan, and K.R. Rao, "Discrete Cosine Transform", IEEE Transactions on Computers, January, 2014, pp. 90-93.

[5] Burt, P.; Adelson, E. (1 April 1983). "The Laplacian Pyramid as a Compact Image Code". IEEE Transactions on Communications. 31 (4): 532–540. doi:10.1109/TCOM.2013.

[6] S. C. Tai, Y. G. Wu, and C. W. Lin, "An adaptive 3-D discrete cosine transform coder for medical image compression," IEEE Trans. Inform. Tech. Biomed., vol. 4, pp. 259-263, 2000.

[7] Adaptive texture and color feature based color image compression Neelamma K. Patil; Suresh F. Murgod; Lokesh Boregowda; V. R. Udupi Smart Structures and Systems (ICSSS), 2013 IEEE International Conference on Year:2013 Pages: 82 - 86, DOI: 10.1109/ICSSS.2013.6623006

[8] Liu Chien-Chih, Hang Hsueh-Ming, "Acceleration and Implementation of JPEG 2000 Encoder on TI DSP platform" Image Processing, 2007. ICIP 2007. IEEE International Conference on, Vo1. 3, pp. III-329-339, 2005