CompE565, Fall 2021
Homework 1: Basic Digital Image Processing Operations

Prepared by
Elena Pérez-Ródenas Martínez
E-mail: eperezrodenasm3836@sdsu.edu
Electrical and Computer Engineering
San Diego State University

Contents

List of Figures

List of Tables

1 Introduction

Storage and transmission bandwidth are an essential part of multimedia communication systems. Image compression is based on encoding an original image with only a few bits. In order to do this, the redundancies must be reduced and the storage and transmission of data must be efficient.

In our everyday lives we consume different types of multimedia content but before we can watch it there is a whole process to be done. Years ago we used analogical methods that have been improved with digital techniques providing better quality with less bandwidth. Image and video coding in analog technology was designed for basic types of video, speech signals... However, with digital representation the transmission and storage can provide high quality of different types of images and videos as long as they have enough bandwidth available. [1]

Over the years, standards for digital image and video coding have been set so that we can interpret pictures coding. Today, they offer us a wide variety of visual communication with a common language. Watching television, talking on the phone, making videoconferences, being able to work on a computer... This couldn't have been possible without image and video coding. [8]

The domain of this work is to work with basic digital image processing operators such as being able to read and display images, converting from RGB to YCbCr and the other way around, subsampling and upsampling bands using different formats, measuring MSE between images and being able to comment on the similarities and differences of the results obtained.

The problem addressed is that images with high quality can be really heavy and we should be able to reduce them using different techniques so that we can transfer more data with less bandwidth.

In the big picture, this work is of great importance because thanks to these basic digital image processing operations and using them in the most effective way we are able to minimize the size of the data without losing quality. On a large scale, this is crutial for the evolution of multimedia communication systems.

2 Procedural Section

2.1 Read and display the image using Matlab

In order to read an image using Matlab we can use the function *imread* with the complete route to the file we want to read and the format as parameters. After that, we can display it with the function *imshow*. We will set the title below.

```
I=imread("C:\Users\Elena Pérez-Ródenas\Desktop\SDSU\MULTIMEDIA COMMUNICATION
SYSTEMS\Flooded_house.jpg","jpg")
figure(1)
imshow(I)
title("Original image")
```

2.2 Display each band (Red, Green and Blue) of the image file

An image is a 3-dimensional array where the third dimension is the component (being 1 red, 2 green and 3 blue)[7]. So we can just choose the different bands and show them.

```
Red=I(:,:,1)
Green=I(:,:,2)
Blue=I(:,:,3)
figure(2)
```

```
imshow(Red)
title("Red Band")
figure(3)
imshow(Green)
title("Green Band")
figure(4)
imshow(Blue)
title("Blue Band")
```

2.3 Convert the image into YCbCr color space

In order to convert a RGB image into a YCbCr one we can use the command *rgb2ycbcr*

```
ycbcr=rgb2ycbcr(I)
figure(5)
imshow(ycbcr)
title("YCbCr Image")
```

2.4 Display each band separately (Y, Cb and Cr bands)

As we did with the RGB image, we select each band and show it.

```
Y=ycbcr(:,:,1)
Cb=ycbcr(:,:,2)
Cr=ycbcr(:,:,3)
figure(6)
imshow(Y)
title("Y Band")
figure(7)
imshow(Cb)
title("Cb Band")
figure(8)
imshow(Cr)
title("Cr Band")
```

2.5 Subsample Cb and Cr bands using 4:2:0 and display both bands

Format 4:2:0 means that 4 is the width, 2 is the number of samples in the upper line and 0 samples in the lower line (so it will copy the samples from the line above). But this is a Lossy Compression so we may lose quality. Sometimes when we combine the colors the patterns can even disappear. [2]. In order to subsample Cb and Cr bands using 4:2:0 we will use the following code reducing x and y pixels by half:

```
Cb420=Cb(1:2:end, 1:2:end)
Cr420=Cr(1:2:end, 1:2:end)
figure(9)
imshow(Cb420)
```

```
title("Subsample Cb using 4:2:0")
figure(10)
imshow(Cr420)
title("Subsample Cr using 4:2:0")
```

2.6 Upsample and display the Cb and Cr bands using

2.6.1 Linear interpolation

In order to upsample Cb and Cr bands with linear interpolation, we will create a new matrix called ycbcrlin with the same Y component and Cb and Cr with 4:2:0. To obtain the points in the middle we will do $(i, i+1) = \frac{(i,i)+(i,i+2)}{2}$ We have to make sure that the size of the matrix is accurate with the corresponding endings.
After that, we will obtain the remaining row points doing $(i+1, :) = \frac{(i,:)+(i+2,:)}{2}$

```
ycbcrlin(:,:,1)=ycbcr(:,:,1)
ycbcrlin(1:2:end,1:2:end,2)=Cb420(:,:)
ycbcrlin(1:2:end,1:2:end,3)=Cr420(:,:)
ycbcrlin(1:2:end,2:2:end-1,2:3)=(double(ycbcrlin(1:2:end,1:2:end-2,2:3))+
    double(ycbcrlin(1:2:end,3:2:end,2:3)))/2
ycbcrlin(2:2:end-1,:,2:3)=(double(ycbcrlin(1:2:end-2,:,2:3))+
    double(ycbcrlin(3:2:end,:,2:3)))/2
cb420lin=ycbcrlin(:,:,2)
cr420lin=ycbcrlin(:,:,3)
figure(11)
imshow(cb420lin)
title("Cb Upsampling with linear interpolation")
figure(12)
imshow(cr420lin)
title("Cr Upsampling with linear interpolation")
```

2.6.2 Simple row or column replication

In order to upsample the image with row or column replication we will first replicate the pixels (i, i) in $(i, i+1)$. After that, we will copy the complete row i in row $i+1$.

```
ycbcrrep=ycbcr
ycbcrrep(1:2:end,2:2:end+1,2:3)=ycbcrrep(1:2:end,1:2:end,2:3)
ycbcrrep(2:2:end+1,:,2:3)=ycbcrrep(1:2:end,:,2:3)
figure(13)
imshow(ycbcrrep(:,:,2))
title("Cb Upsampling with row or column replication")
figure(14)
imshow(ycbcrrep(:,:,3))
title("Cr Upsampling with row or column replication")
```

2.7 Convert the image into RGB format

We just have to convert again both images into RGB format with the command *ycbcr2rgb*.

```
rgblin=ycbcr2rgb(ycbcrlin)
rgbrep=ycbcr2rgb(ycbcrrep)
```

2.8 Display the original and reconstructed images (the image restored from the YCbCr coordinate)

```
figure(15)
imshow(rgblin)
title("RGB image upsampled by linear interpolation")
figure(16)
imshow(rgbrep)
title("RGB image upsampled by row or column replication")
```

2.9 Comment on the visual quality of the reconstructed image for both the upsampling cases

This section will be responded on the results section.

2.10 Measure MSE between the original and reconstructed images (obtained using linear interpolation only). Comment on the results

We will measure both the Cb and Cr bands between the original image I and the one after linear interpolation.

```
[m,n]=size(I)
n=n/3
double MSECb
double MSECr
MSECb=0
MSECr=0
for i=1:1:m
        for j=1:1:n
                MSECb=MSECb+(double(I(i,j,2))-double(rgblin(i,j,2))).^2
                MSECr=MSECr+(double(I(i,j,3))-double(rgblin(i,j,3))).^2
        end
end
MSECb=MSECb/(m*n)
MSECr=MSECr/(m*n)
```

The results will be commented on the results section.

2.11 Comment on the compression ratio achieved by subsampling Cb and Cr components for 4:2:0 approach. Please note that you do not send the pixels which are made zero in the row and columns during subsampling

First we calculate the size of the original picture adding the size of the 3 bands. Then we calculate the same after the 4:2:0 subsampling. Finally, we calculate the compression ratio with the following code:

```
size1 = size(Y,1)*size(Y,2)+size(Cb,1)*size(Cb,2)+size(Cr,1)*size(Cr,2)
size2 = size(Y,1)*size(Y,2)+size(Cb420,1)*size(Cb420,2)+size(Cr420,1)*size(Cr420,2)
ratio=size1/size2
```

The results will be commented on the results section.

3 Results

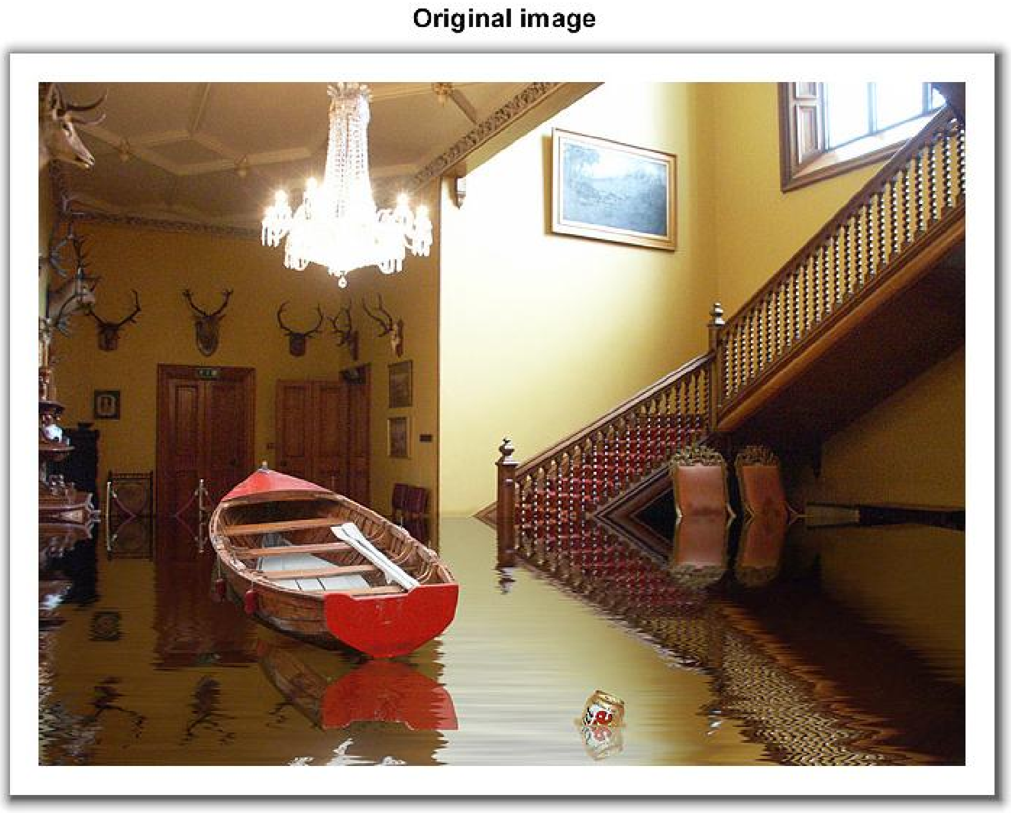
3.1 Read and display the image using Matlab

As a result of the command *imread* we read the image from the file specified by the route, inferring the format of the file from its contents. [3]

imshow(I) displays the grayscale image I in a figure. *imshow* uses the default display range for the image data type and optimizes figure, axes, and image object properties for image display. [4]

This is the image shown:

Figure 1: Original image



3.2 Display each band (Red, Green and Blue) of the image file

As a result of the commands above, we obtain these three images:

Figure 2: Red Band

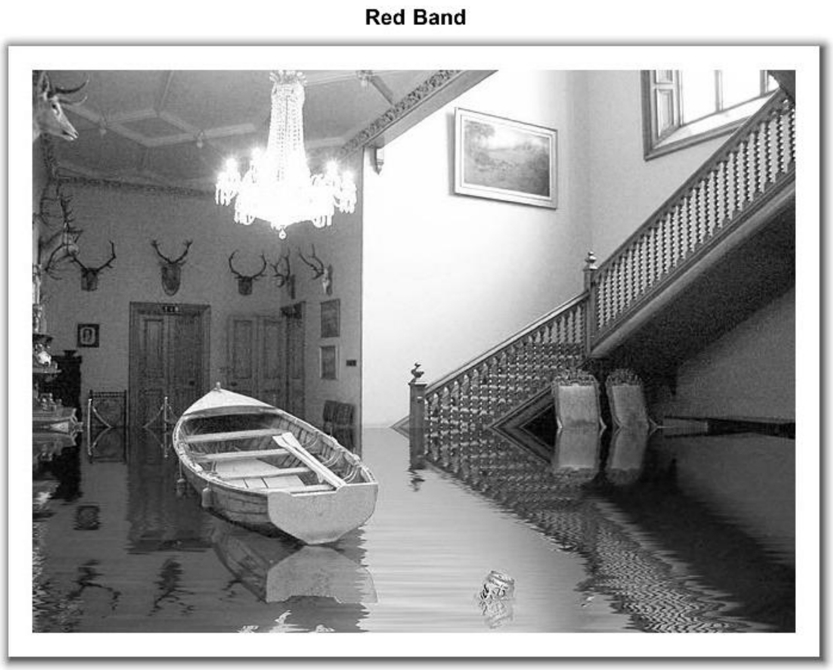


Figure 3: Green Band

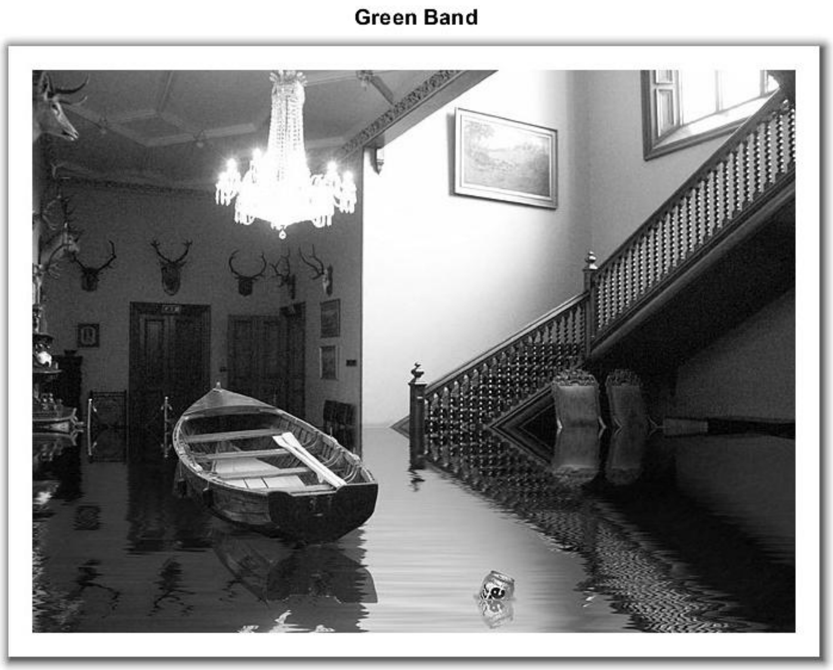
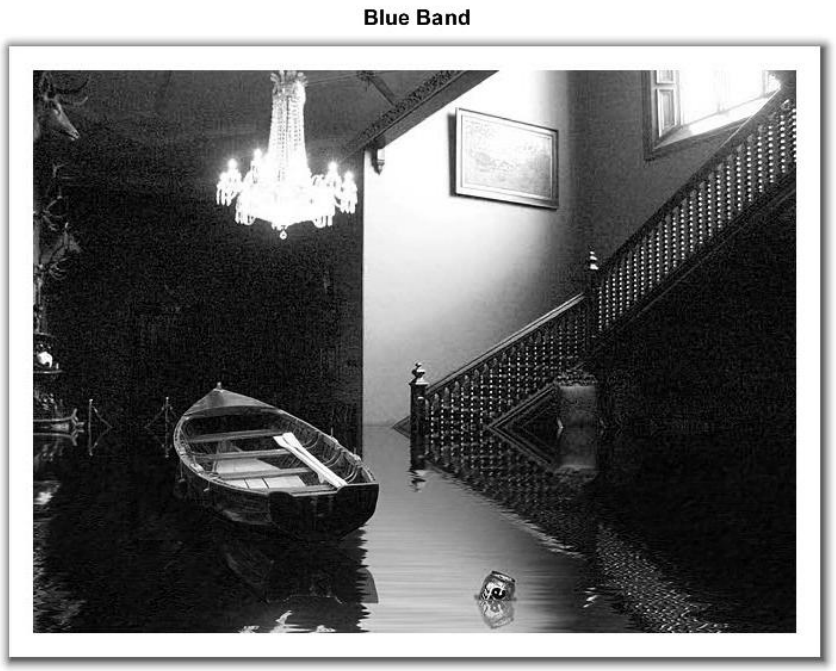


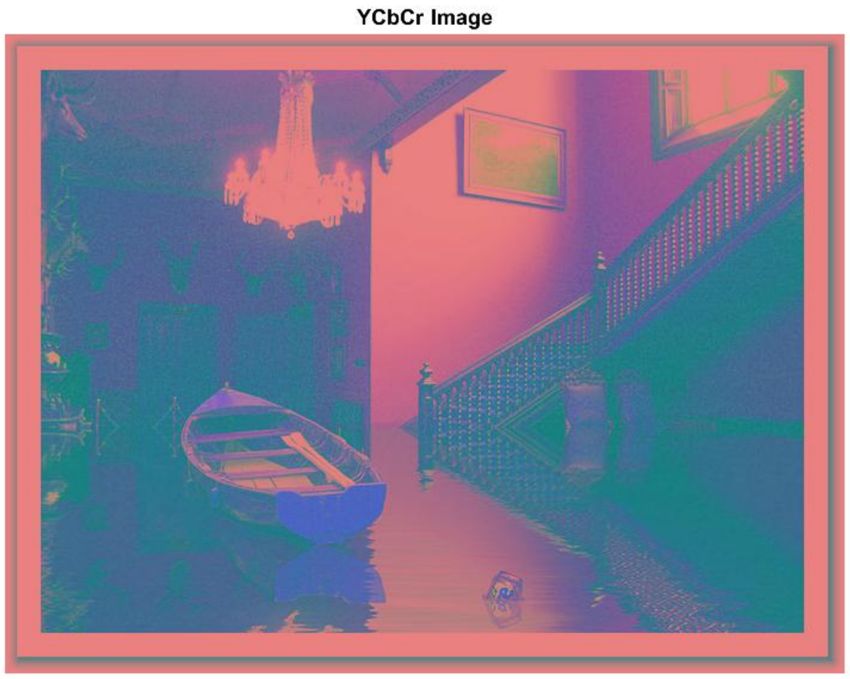
Figure 4: Blue Band



3.3 Convert the image into YCbCr color space

The command *rgb2ycbcr* converts the red, green, and blue values of an RGB image to luminance (Y) and chrominance (Cb and Cr) values of a YCbCr image. [5]

Figure 5: YCbCr Image



3.4 Display each band separately (Y, Cb and Cr bands)

These are the images obtained:

Figure 6: Y Band

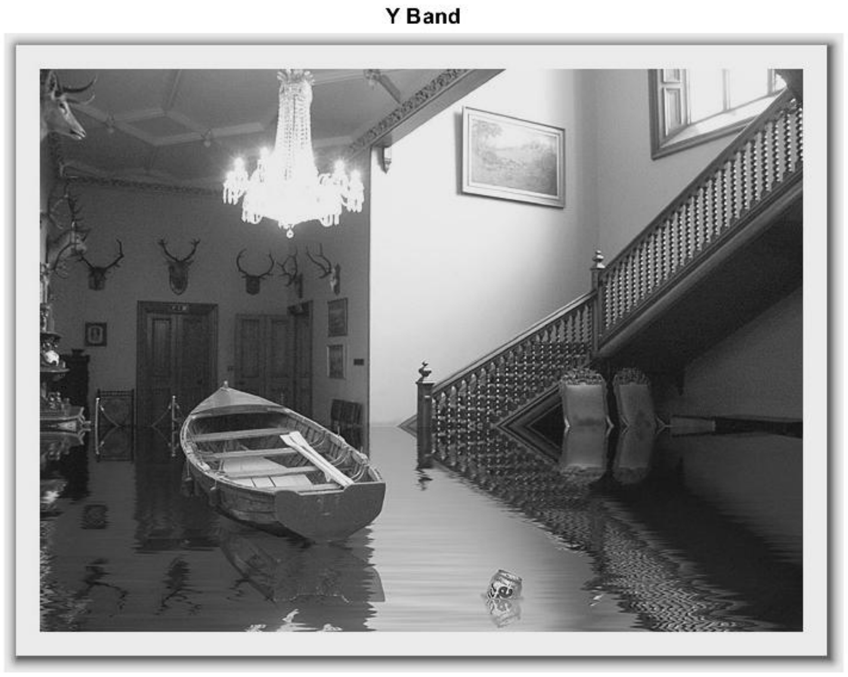


Figure 7: Cb Band

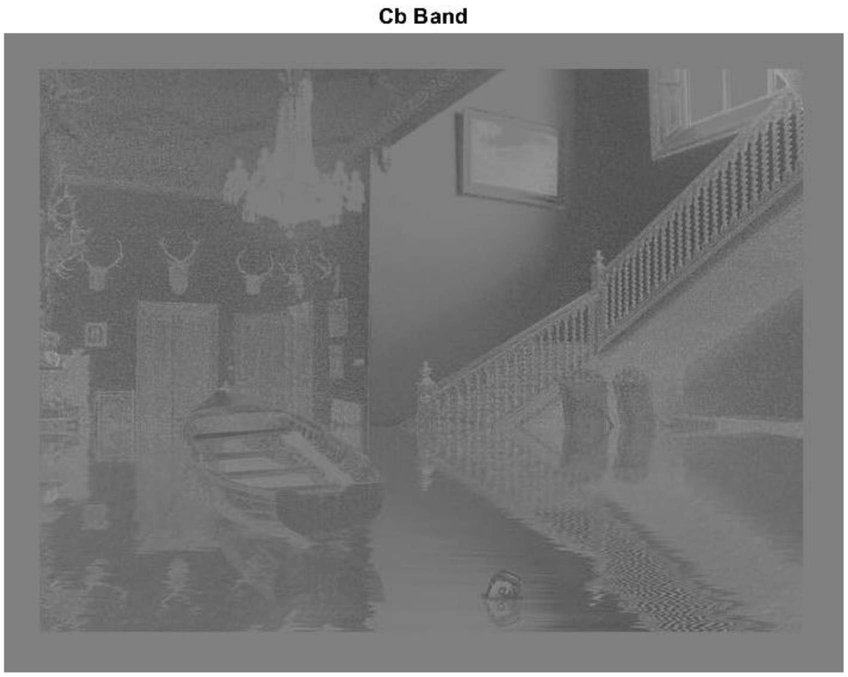
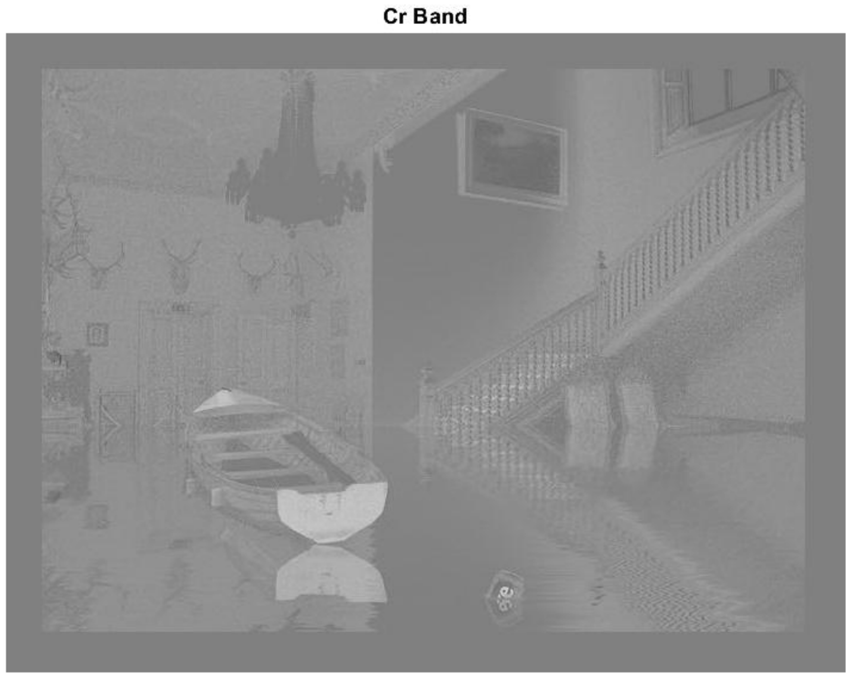


Figure 8: Cr Band



3.5 Subsample Cb and Cr bands using 4:2:0 and display both bands

As we can see in the images some patterns are lost although we have won space.

Figure 9: Unsample Cb Band 4:2:0

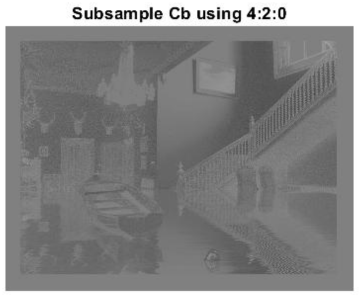


Figure 10: Unsample Cr Band 4:2:0

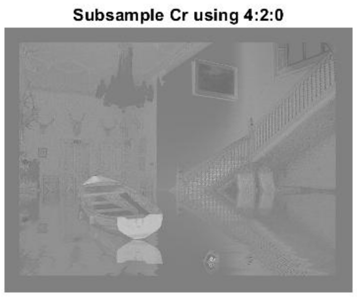
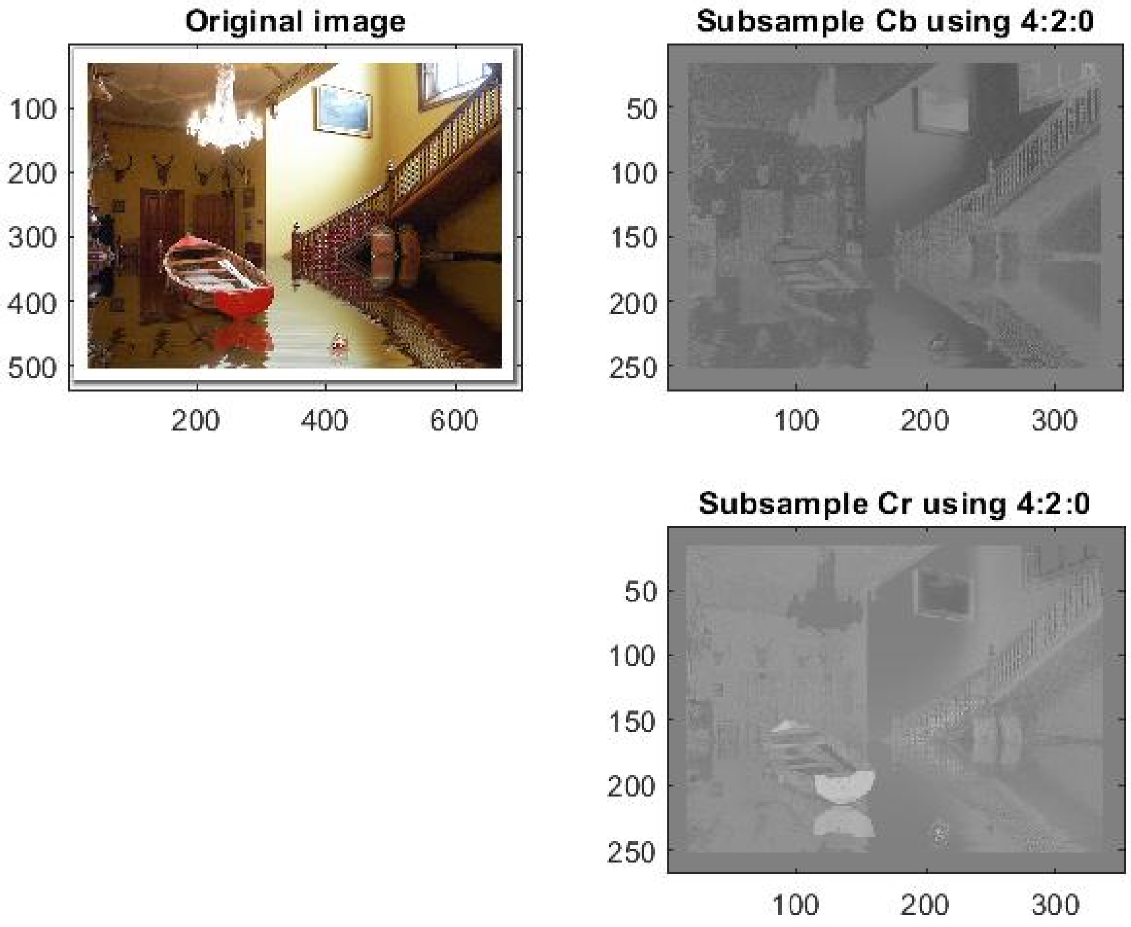


Figure 11: Comparison among original image, and subsamples Cb and Cr using 4:2:0



3.6 Upsample and display the Cb and Cr bands using

3.6.1 Linear interpolation

These are the results of the images after upsampling Cb and Cr bands using linear interpolation:

Figure 12: Cb Upsampling with linear interpolation

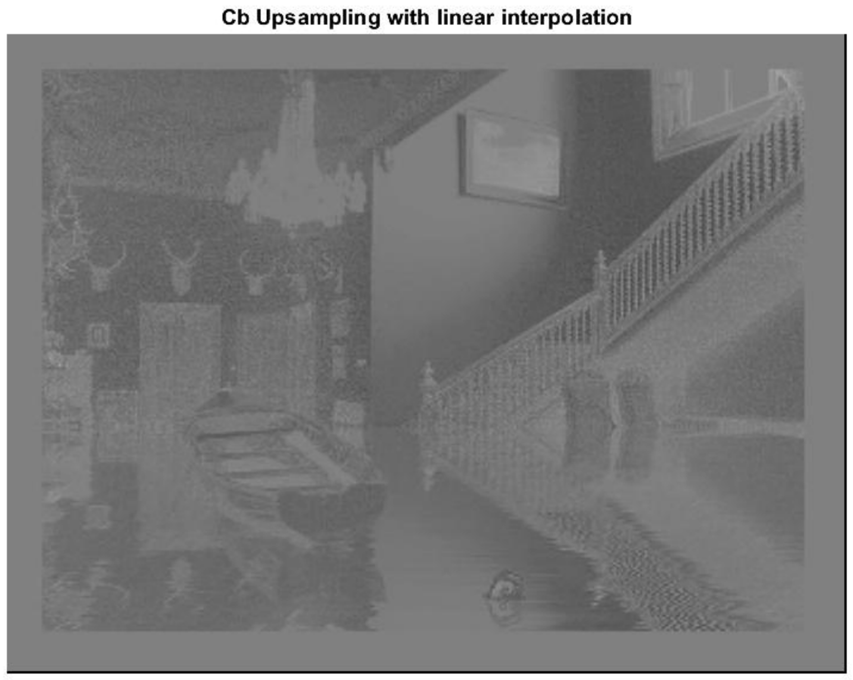


Figure 13: Cr Upsampling with linear interpolation

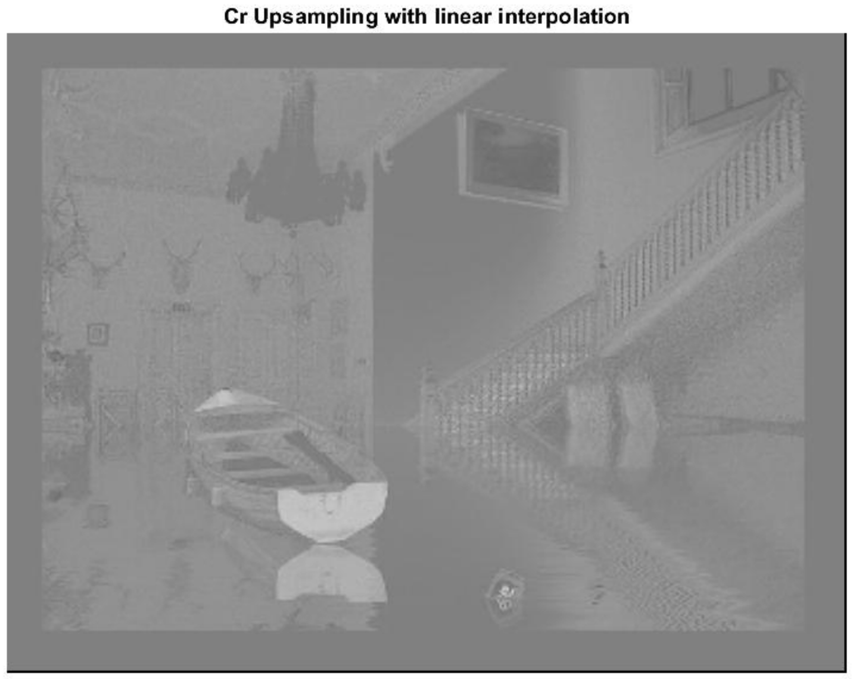
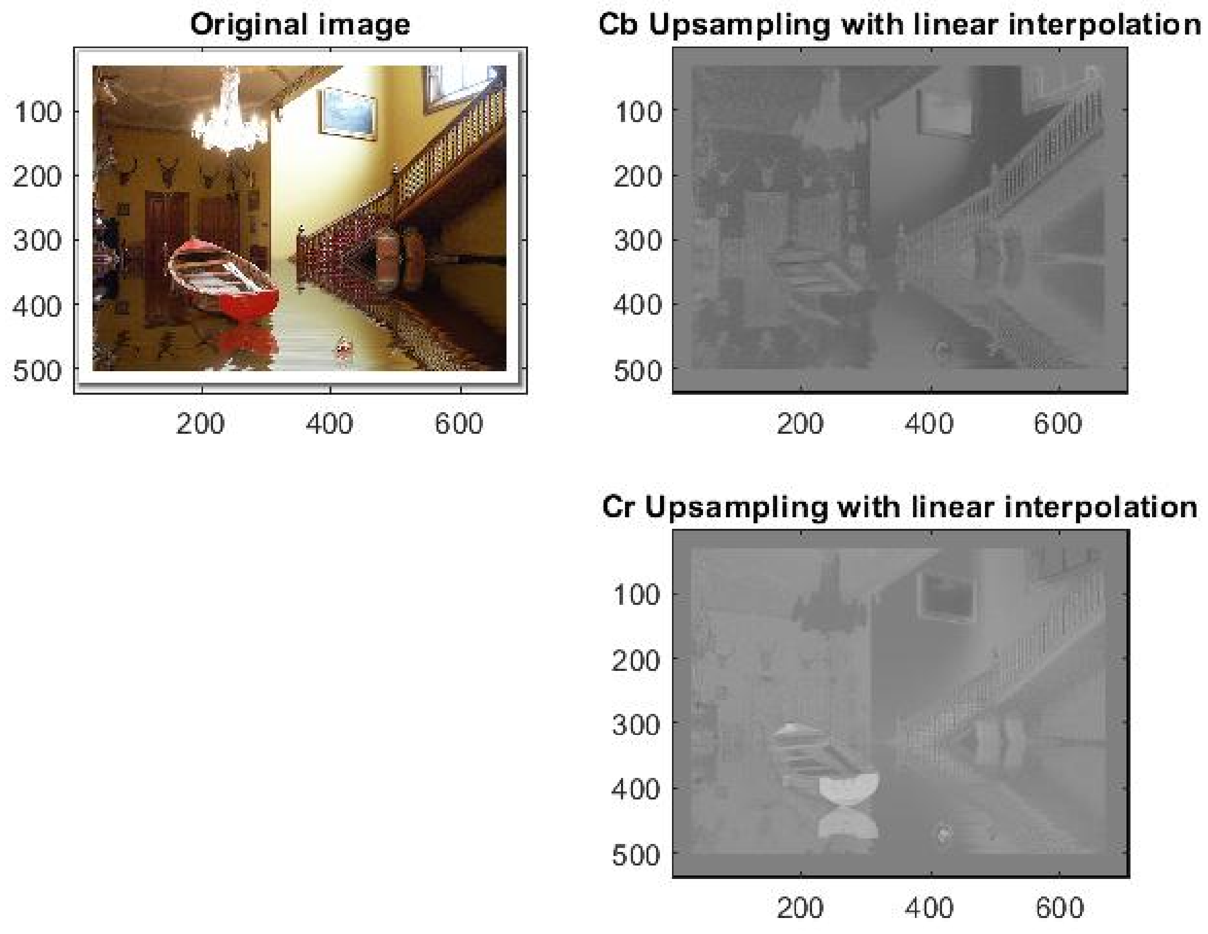


Figure 14: Comparison among original image, and upsamples Cb and Cr using linear interpolation


Original image
Cb Upsampling with linear interpolation
Cr Upsampling with linear interpolation
100
200
300
400
500
200
400
600

3.6.2 Simple row or column replication

These are the images obtained:

Figure 15: Cb Upsampling with row or column replication

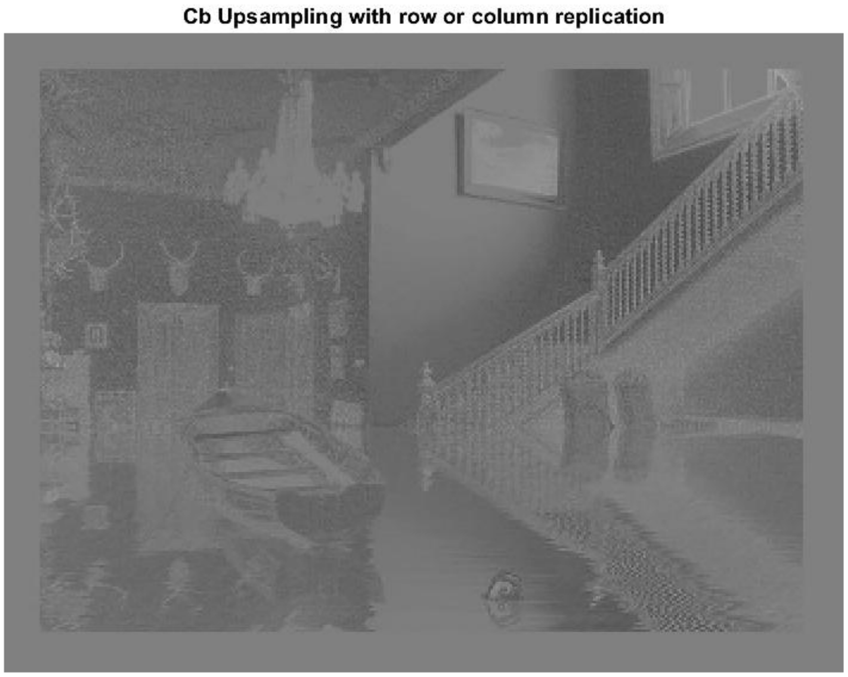
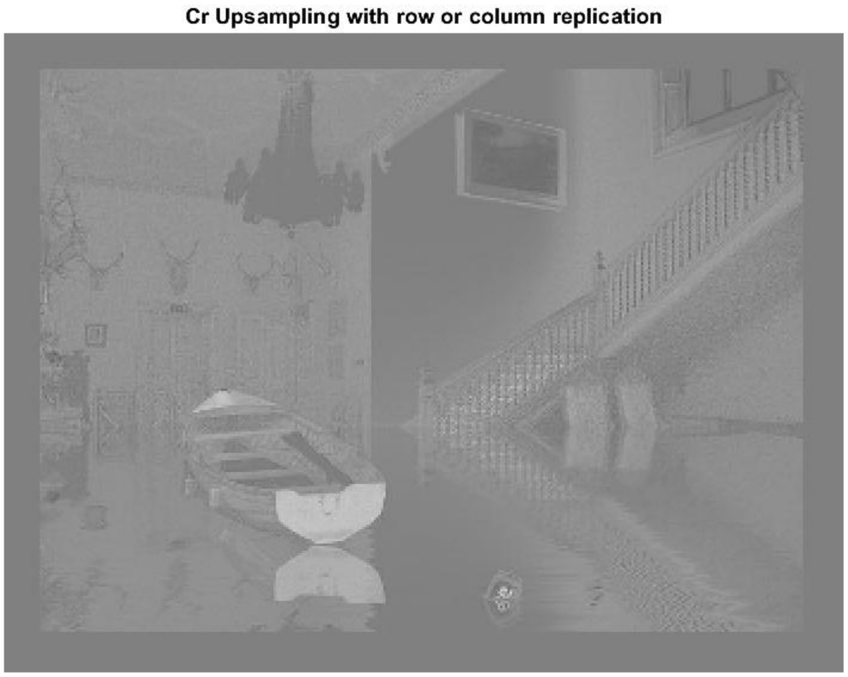


Figure 16: Cr Upsampling with row or column replication



3.7 Convert the image into RGB format

After the command we will have both images converted rgblin and rgbrep.

3.8 Display the original and reconstructed images (the image restored from the YCbCr coordinate)

Figure 17: Original image

Figure 18: RGB image upsampled by linear interpolation

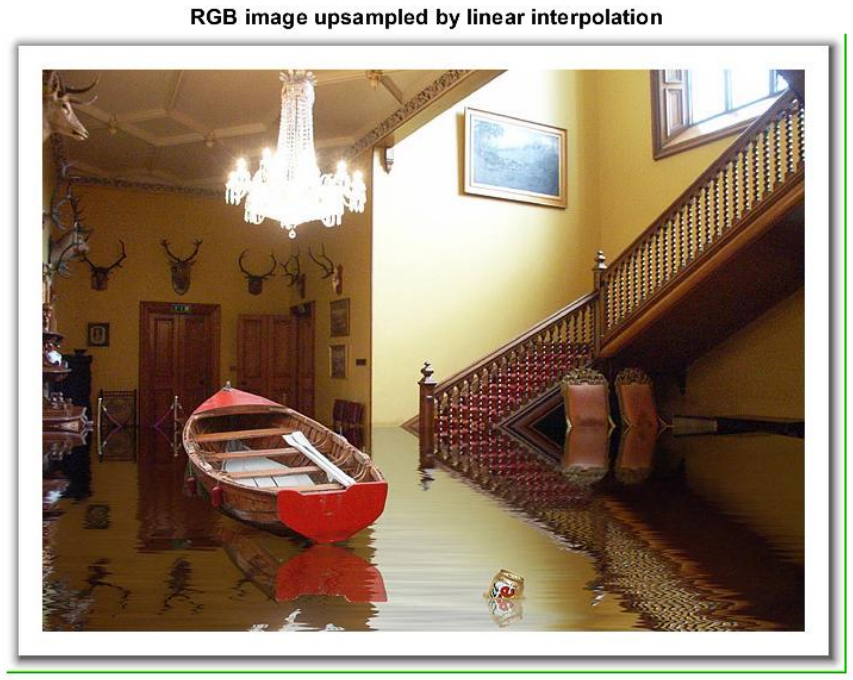
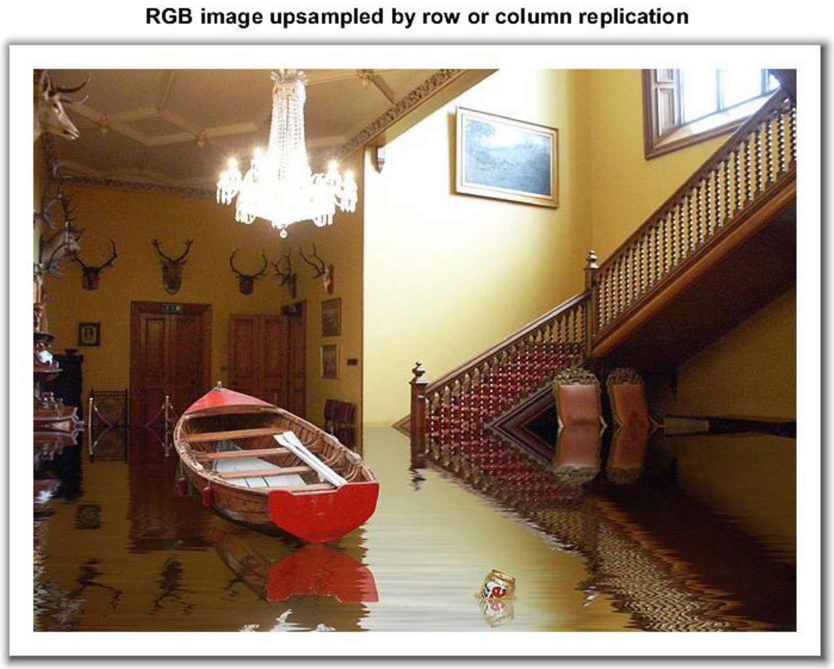
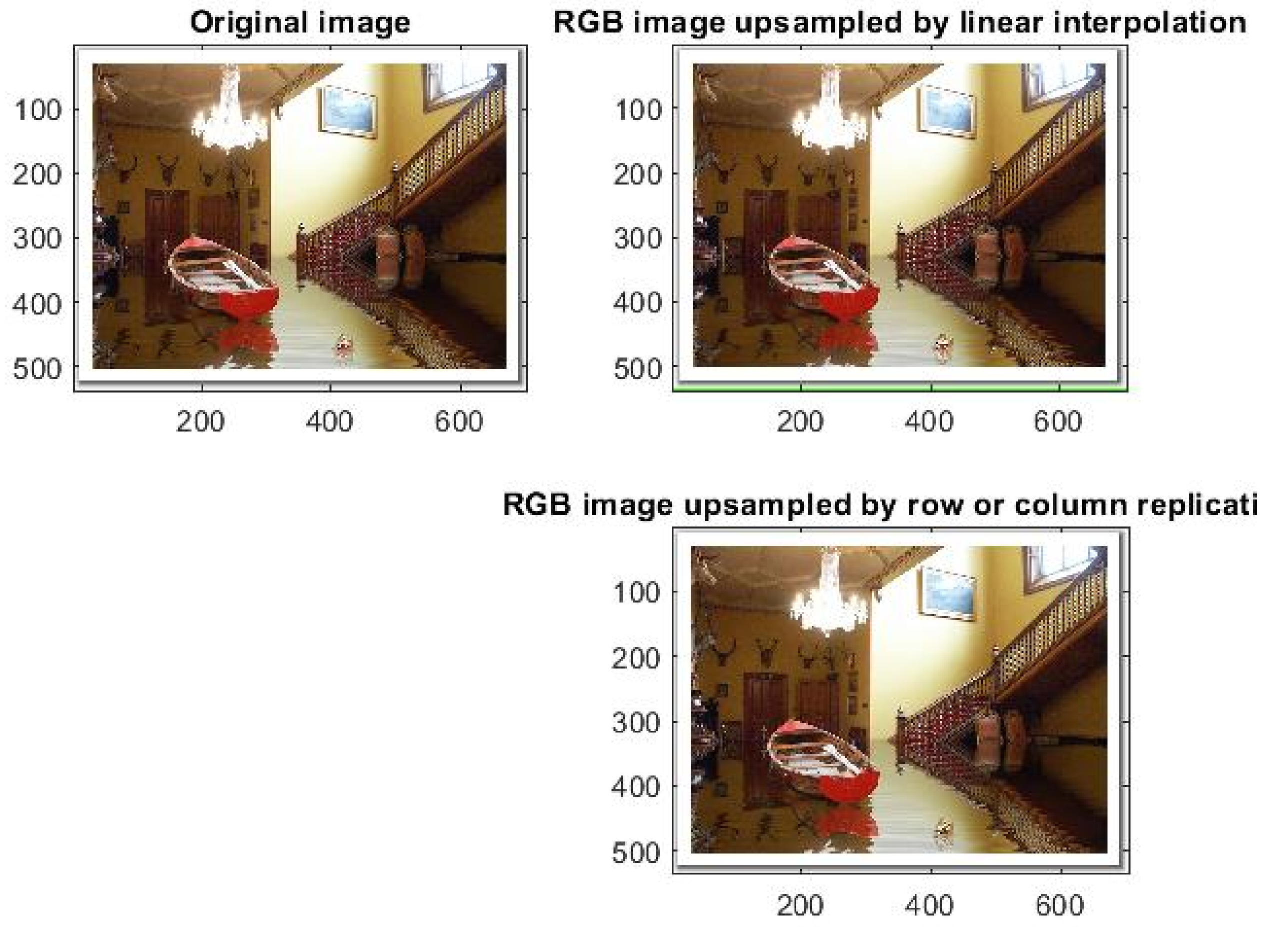


Figure 19: RGB image upsampled by row or column replication



3.9 Comment on the visual quality of the reconstructed image for both the upsampling cases

Figure 20: Comparison among original image and reconstructed images


Original image
RGB image upsampled by linear interpolation
RGB image upsampled by row or column replicati
100
200
300
400
500
200
400
600

As we can see, the visual quality of the reconstructed images for both upsampling cases is really good. We can hardly tell the difference between the original one and the reconstructed ones. This is a good sign, in fact, 4:2:0 only stores and transmits one color channel per line and the other channel is recovered from the previous line. Besides, 4:2:0 schemes halve the bandwidth compared to no chroma subsampling. [6]

3.10 Measure MSE between the original and reconstructed images (obtained using linear interpolation only). Comment on the results

The MSEs obtained are 5.1833 for the Cb band and 256.5679 for the Cr band.
This matches with the fact that we don't really see much difference between the original picture and the reconstructed one and corroborates the fact that this technique in this particular case is really effective.

3.11 Comment on the compression ratio achieved by subsampling Cb and Cr components for 4:2:0 approach. Please note that you do not send the pixels which are made zero in the row and columns during subsampling

The compression ratio obtained is 2. This makes sense because when we subsample an image with 4:2:0 format we reduce the pixels to the half. That is why the size of the original picture is 1132032 and of the subsampled one is just 566016.

4 Conclusion

To sum up, it's so good to know different techniques for image compression as well as the operations to get to do them. This is because in some cases we can reduce significantly the size of the pictures without losing quality at all (at least for the human eye). Of course, every technique is not good for every case, that's why we have to try and study the ones that are better for the different types of images.
From my point of view this assignment is a really good way to help students understand more in depth some of the techniques and to be more aware of what happens before we see multimedia content in our everyday lives.

5 References

[1] https://link.springer.com/chapter/10.1007/978-3-642-18750-6$_1$5.

[2] https://www.youtube.com/watch?v=fkz2-JVaYDk.

[3] https://www.mathworks.com/help/matlab/ref/imread.html.

[4] https://www.mathworks.com/help/images/ref/imshow.html.

[5] https://www.mathworks.com/help/images/ref/rgb2ycbcr.html.

[6] https://en.wikipedia.org/wiki/Chroma$_subsampling4 : 2 : 0$.

[7] COMPE-565. *Basic Matlab Commands for Digital Image Processing.*

[8] Gary J. Sullivan. *Visual Coding Standards: A New World of Visual Communication.*