IMAGE COMPRESSION-CS3IA16

Alfie Sargent

BSc Robotics, jy013677@reading.ac.uk

ABSTRACT

The following paper details the development cycle of an image compression program using MATLAB and its in built functions. The use of colour channel segmentation, image sampling, image quantitation and interpolation has been used to reduce the size of an image but also attempt to limit the reduction in quality that the human eye can perceive. Two different original images have been used in the experiment, one greyscale and one colour to showcase the potential that the compression technique has across a range of image types. The overall conclusion to the experiment is positive, with the images being compressed and the quality of the image not being severely affected.

1.INTRODUCTION

The task provided tasks the user with designing an image compression program that is capable of reducing the size of an image whilst preserving as much of the images quality as possible using any interface the user desires; this experiment was conducted in MATLAB.

Image compression techniques have been used for many years to reduce the size of images but also to maintain their quality. The two types of image compression techniques lossless and lost. Lossless compression is where no data is lost in the reconstruction of the compressed image. Lossy data compression is where data is lost in the reconstruction of the compressed image but it does not necessarily mean that "visible" data is lost due to how the human eye operates. The following experiment describes how lossy compression has been used for both colour and black and white images.

2.METHODOLGY

2.1 Techniques used

The human eye is capable of seeing a vast array of wavelengths of light, ranging from 390 to 700 nm of light. The cone cells in your eye (the photoreceptor cells responsible for detecting different colour wavelengths of light) are split into 3 different types; L, M and S types (see Figure 1). Each type of cone is more responsive to certain wavelengths of light; the L cone is more responsive to green, M more responsive to red and S more responsive to blue.
This experiment aims to use this information to compress image sizes so that, even though lossy compression is taking place, the human eye is not capable of accurately determining if the image is of a lower quality or not.

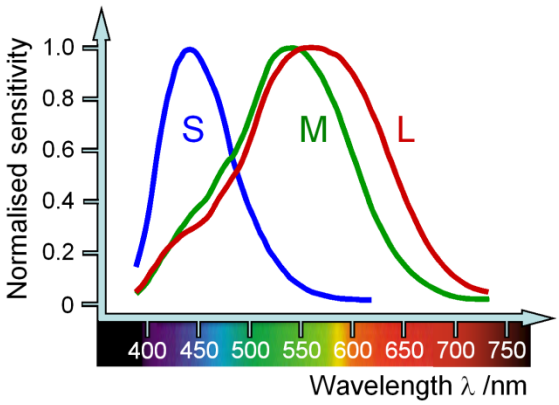


Figure 1: This figure illustrates the range of wavelength in which each specific cone operates in the human eye.

2.2 Image Sampling & Image Quantisation

Image sampling is a technique used in image compression to greatly reduce the image size by essentially reducing the number of bits that are allocated to each pixel in an image (please see Figure 2). Although this initially sounds like a bad thing, many images have a superfluous amount of bits allocated to each pixel; too many that the human eye can easily distinguish. For instance, a lot of images, especially close up images, do not rely on certain colours to have as high of a bit count in their respective pixels and therefore, a certain colour channel can be down sampled and as a result compress the image significantly. The only real disadvantage is the reduction in quality once the image has been magnified (if done properly and for the right image).

However, a technique known as interpolation in conjugation with Image sampling & image quantisation can provide much better results.

Figure 2: This image shows a bad example of image sampling being used. Here, the entire image has been down sampled instead of selective down sampling and consequently the entire image appears pixelated and blurred.

2.3 Interpolation

The interpolation technique used in this experiment is the bi-linear interpolation technique. Essentially, this technique relies on the relationship that neighbouring pixels have with each other and take an average of the pixels around them. The more severe you want the interpolation, the more pixels are averaged. Figure 3 shows how interpolation is used in this experiment, where the average of the neighbouring pixels colour channel is used to determine a mean.

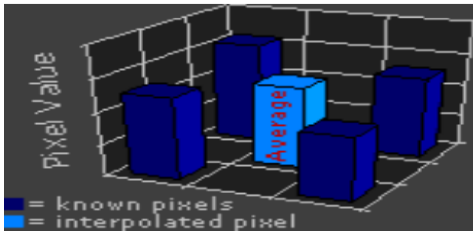


Figure 3: Interpolation is used to average the pixel values of the surrounding 2x2 pixels around it to come up with a final interpolated value. This technique creates a much smoother final product then NNI (nearest neighbour interpolation).

By doing this, the image sampling is less obvious because the mean of each down-sampled pixel is used. However, although this does mean that the overall image quality is increased, you can see where drastically different colours "bleed" into other sections of the image when magnified as shown in Figure 4. This "bleeding" effective is not obvious to the human eye and as a result has been used here.

Figure 4: The above figure illustrates how interpolation can affect an image but is only visible when zoomed in drastically. This image has been zoomed in 10 times so that the interpolation can easily be seen. Certain colours, in this instance red and green, are "bleeding" into the image. However, this selective interpolation results in a high quality image that, too the human eye is not distinguishable from the original.

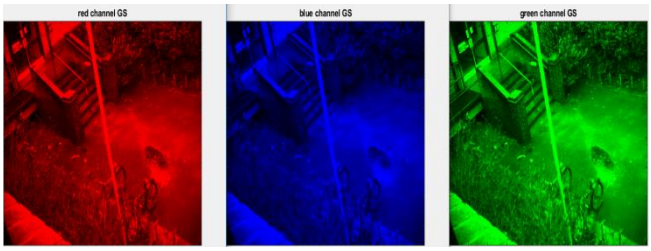
2.4 Selective Image sampling

The colour images that have been used in this experiment have a varying range of dominant colours per pixel. However, the general dominant colours out of the three primary colours (red, green and blue) are the red and green due to the yellow (which is a mixture of green and red) colours of the brick and the green colours of the grass. There is a small amount of blue in the sky but for the majority of the image there is a noticeable lack of blue. As a result, the intensity of the down-sampling in the blue sector of the images is much larger than the down-sampling of the pixels in the other two colours. Therefore, the size of the image can be greatly reduced by a huge factor and no noticeable difference can be seen due to the imperfections of the human eye.

3. RESULTS & DISCUSSION

3.1 Grey scale Images

For the image compression used for grey scale, the image was initially read into the program and then split into its respective RGB colours (see Figure 5).



```
%******************DISPLAY OF THE RGB VALUES OF THE IMAGE*******************

%output only the red channel of the image A
figure ();
image(A(:,:,1)); title('red channel GS');
col = zeros(256,3); %creating an array that will be filled with zeroes.
col(:,1) = 0:1/255:1;
colormap(col); caxis([0 255]);%setting the colormap of the image to the red spectrum.

%output only the green channel of the image A
figure ();
image(A(:,:,2)); title('green channel GS');
col = zeros(256,3);
col(:,2) = 0:1/255:1;
colormap(col); caxis([0 255]);%setting the colormap of the image to the green spectrum

%output only the blue channel of the image A
figure ();
image(A(:,:,3)); title('blue channel GS');
col = zeros(256,3);
col(:,3) = 0:1/255:1;
colormap(col); caxis([0 255]);%setting the colormap of the image to the blue spectrum.
```

Figure 5: The grey scale image was separated into red, green and blue values so that they could be processed later on in the program. The functions colormap and caxis have been used here to ensure that the right colour values are being output for the image.

Once this was done, each RGB colour was down sampled equally to ensure that no distortion was prevalent in the grey scale image (see Figure 6).


```
%*********************DOWNSAMPLING OF THE IMAGE*****************************

%this is the downsampling section for all of the "colours" in the image.
%The greyscale image means that although the downsampling can be a set
%standard across the whole image, and therefore it is "easier" to
%downsample the image, it does mean that some quality is lost because you
%cannot choose what less dominant colours you would downsample more.
for width_1=1:width/2
for height_1=1:height/2
    blue(height_1,width_1) = mean(mean(A((height_1-1)*2+1:height_1*2,(width_1-1)*2+1:width_1*2, 3)));
end
end

for width_1=1:width/2
for height_1=1:height/2
    green(height_1,width_1) = mean(mean(A((height_1-1)*2+1:height_1*2,(width_1-1)*2+1:width_1*2, 3)))
end
end

for width_1=1:width/2
for height_1=1:height/2
    red(height_1,width_1) = mean(mean(A((height_1-1)*2+1:height_1*2,(width_1-1)*2+1:width_1*2, 3)));
end
end
```

Figure 6: The down sampling for the greyscale image was done equally for each colour channel so that no distortion can occur, the downside of using this technique for a black and white image. The code shown is a formula to calculate the mean number of pixels in the horizontal and vertical direction that will be down sampled.

Next, the down sampled images were displayed, saved (see Figure 7) and then decompressed using bi-linear interpolation (see Figure 8).

```
%******************DISPLAY THE DOWNSAMPLED RGB IMAGE ***********************

%output the down sampled blue channel of the image A
figure();
image(blue); title('blue channel GS, down-sampled');
col = zeros(256,3);
col(:,3) = 0:1/255:1;
colormap(col); caxis([0 255]);

figure();
image(red); title('red channel GS, down-sampled');
col = zeros(256,3);
col(:,1) = 0:1/255:1;
colormap(col); caxis([0 255]);

figure();
image(green); title('green channel GS, down-sampled');
col = zeros(256,3);
col(:,2) = 0:1/255:1;
colormap(col); caxis([0 255]);

figure();
image(bluecol); title('blue channel colour, down-sampled');
col = zeros(256,3);
col(:,3) = 0:1/255:1;
colormap(col); caxis([0 255]);

%saving the downsampled images, this is the collective lossy compression of
%the images.
imwrite(blue, 'downsampled_blue_GS.bmp','bmp');
imwrite(red, 'downsampled_red_GS.bmp','bmp');
imwrite(green, 'downsampled_green_GS.bmp','bmp');
imwrite (bluecol, 'downsampled_blue_colour.bmp', 'BMP');
```

Figure 7: Here we can see the down sampled images being displayed in the program before being saved into memory so that they can be analysed later if necessary.

```
%******************RECONSTRUCTION OF THE IMAGE*****************************

% reconstruct a complete image from reduced blue, green and red
% it is then interpolated linearly. Use of the interpolation function is
% shown here.

%interpolation for the GS image
[ih,iw] = size(blue);
A(:,:,3) = interp2([1:width/iw:width],[1:height/ih:height]',blue,[1:width],[1:height]','linear');

[ih,iw] = size(red);
A(:,:,1) = interp2([1:width/iw:width],[1:height/ih:height]',red,[1:width],[1:height]','linear');

[ih,iw] = size(green);
A(:,:,2) = interp2([1:width/iw:width],[1:height/ih:height]',green,[1:width],[1:height]','linear');
```

Figure 8: Interpolation has been used to take the average of the neighbouring pixels around it. The use of the MATLAB function interp2 has been used to bilinearly interpolate the pixels and smooth the image.

Lastly, the decompressed image was shown and the compression ratio between the original image and the compressed RGB images was calculated and displayed (see Figure 9).

```
%********************DISPLAY THE RECONSTRUCTED IMAGE************************

image(A); title('compressed image GS final');

image (B); title ('compressed image colour final');

figure (); title ('Original images');
subplot (1,2,1); imshow (A); title ('final decompressed GS');
subplot (1,2,2);imshow(B); title ('final decompressed colour');

%********************COMPRESSION RATIO ***************************************

%getting the size of the original image in the three dimensions of data
%that is is in; 768*1024*3
[xogs,yogs,zogs]=size(A);

%Getting the size of the three indivdually compressed colour channels.
[xrgs,yrgs,zrgs]=size(red);
[xbgs,ybgs,zbgs]=size(blue);
[xggs,yggs,zggs]=size(green);

%multiplying the x,y and z values to get the size of the image overall
sizeo=xogs*yogs*zogs;
sizec=((xrgs*yrgs*zrgs)+(xbgs*ybgs*zbgs)+(xggs*yggs*zggs));

%formula for the compression ratio which is then printed.
CompressionratioGS=(sizec/sizeo);
fprintf("compression ratio for GS: %0.4f\n", CompressionratioGS);
   %*********END OF THE RGB TEST **************
```

Figure 9: The final decompressed image was output and then a simple calculation was done to calculate the compression ratio of the original image in comparison to the compressed file. To do this, the size of each image was put into an array and the x, y and z values multiplied and then compared.

The grey scale images that have been compressed in this exercise produce a different result to the colour images. Due to the image obviously having no dominant colours, the entire image was simply down sampled and as a result, the image is still very clear but the overall down sampling of each channel needed to be identical to stop the image from being distorted in one colour spectrum. However, although no dominant colour could be down sample, each channel could be down sampled only slightly and still produce a drastic change in image size.

Furthermore, due to no dominant colours in a grey scale image, no bleeding can be seen in the image, as shown in Figure 10 and 11.

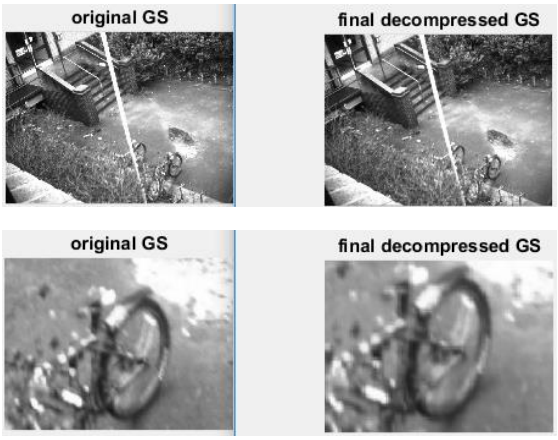


Figure 10 & 11: The final comparison between the original grey scale image and the final decompressed image. The image is only 75% of the size of the original image but doesn't appear, at first glance, to have any discernible difference in quality. However, once magnified, the image can be seen to be slightly lower resolution. Moreover, the human eye does not actually see that huge a difference at normal magnification.

3.2 Colour Images

The image compression technique used here is the same technique used for the greyscale images but slightly different and the results produced are much more effective. Due to the image actually having colour this time, the picture can be analysed and then dominant colours can be seen. Dependent upon the analysis, certain spectra of colour can be sampled more severely then others. In the case of the image used in this exercise, the colour blue is the least dominant and as a result, the down sampling can be more severe without much obvious loss in quality.

The code and procedure followed for the colour compression is the same as the greyscale technique but with a few differences such as the sampling of the colours is only done in the blue spectrum and much more severely as shown in Figure 12 (please see Appendix 1 for more details on the code used). Here, the severity of the sampling is shown and may initially seem like the image has been obviously reduced in quality but once the interpolation pass is completed and the final image put together it is hard to discern any noticeable change.

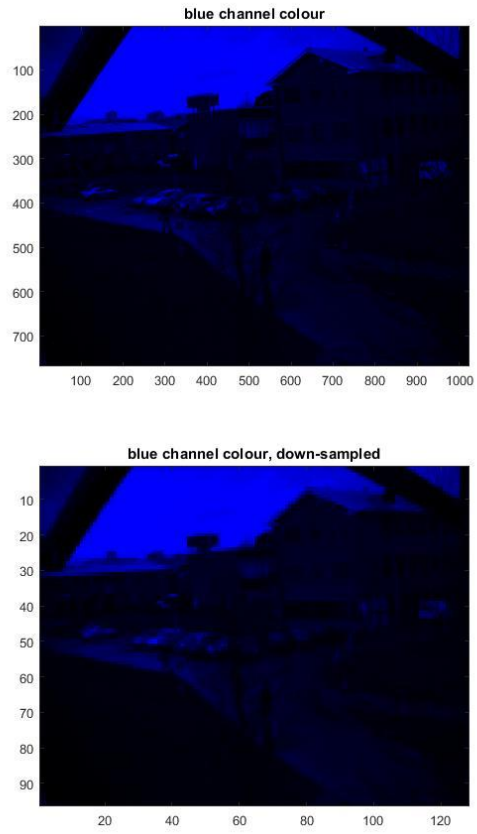


Figure 12: This Figure shows us the severity of the down sampling. The factor has been changed from 2 in the greyscale to a factor of 8, so the image quality in the blue spectrum has been greatly reduced.

Once the colours had been separated, then down sampled and the saved to memory, the image was then reconstructed using interpolation. Again, very much the same as the greyscale technique described in the previous section.

However, the main change here is that only one spectrum of colour is actually being adjust and as a result the quality of the image is still very good. On the other hand, "bleeding" can be seen when the image is magnified, as shown in Figure 13, where the more dominant colours start to cross over into certain parts of the image where blue colour is more prevalent.

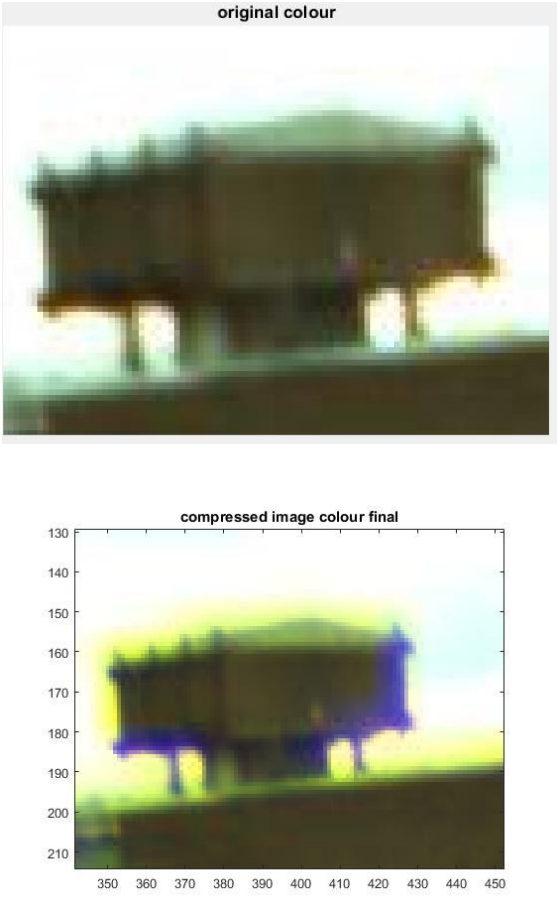


Figure 13: From the images shown above, the bleeding affect that selective down sampling has on colour images can be seen. The image has been zoomed in, onto a rooftop, and the blue of the sky is being distorted by the red and green colours of the rick of the building. However, the human eye is not able to process such small imperfections accurately in an image and therefore this bleeding effect doesn't really affect the quality of the image too much and the result is a highly compressed image.

The final results of the comparison between the original and the decompressed image can be seen in Figure 14, where there is no discernible difference in quality and the image has been greatly reduced in size.

Figure 14: The final comparison between the original and the decompressed image can be seen here. Although there is some bleeding of colours around the top of the image, overall, the compression has worked very well.

3.3 Compression Ratio

The compression ratio was calculated by getting the size of the original image in an array of its x, y and z values and then getting each individually compressed colour channels size array as well. These numbers were then multiplied to get an overall size for the image and then the size of the collective compressed file was divided by the size of the original file to get a ratio.

The greyscale compression ratio was basically just a factor of how much image sampling was done on the image, in this case, the best ratio discovered was 0.25, where the image is significantly reduced but the quality of the image isn't greatly reduced (relative to the human eye).

The colour compression ratio was a little bit different. The only channel that was actually being compressed was the blue channel and as a result, the compression ratio was significantly smaller, only a value of 0.68. This was due to two thirds of the colour data not being compressed at all in the image and only the blue portion. However, the blue channel was greatly reduced to an image of only 12544 bytes in comparison to 786432 bytes; a huge reduction.

4. CONCLUSION & FUTURE DEVELOPMENTS

4.1 Future Developments

Although Fourier transformations and discrete cosine transformations could have been used to great effect, if the "colour" technique was pursued further, I would have liked to have explored the possibility of using YCbCr by rotating the RGB colours into a different position by using an inbuilt MATLAB function, a simple coordinate transformation. Thus, Chroma subsampling could be done, this is the practice of encoding images by implementing less resolution
for chroma information than for luma information, taking advantage of the human visual system's lower acuity for colour differences than for luminance. This would have made a greatly reduced image with almost no "visible" reduction in quality for the image. However, this would only really be applicable for the colour images. For the grey scale images, as stated previously, the possibility of using a discrete cosine transformation and quantization. This would have provided basically no "visible" drop in quality but a hugely reduced file size.

4.2 Conclusion

In conclusion, the image compression experiment was a success; the images used had been significantly reduced in size and although the quality had been technically reduced, the manipulation of the human eyes cones had been used to make it seem like it was less prevalent. However, I do believe that the images could have used some more refined techniques in deciding what parts of the image are actually more prevalent then others by using a Fourier transform and a mask.

Furthermore, improvements could have been made to the code to make it more robust, reliable and efficient; in its current state it does take a while to compress the image. Moreover, more images could have been used to prove that the image compression used was more reliable across multiple different images.

Overall, I do believe that the experiment was a success.

5. APPENDICES

5.1 Appendix 1: Image Compression Source Code

```
%Clear all of the previous images
if any of the settings are
changed.
clear all; close all; clc

%********************DISPLAYING
OF THE
IMAGE*****************************
*
%Load the images from the provided
ppm's.
A = imread('Images\4-gs.ppm');
image(A); title('original
greyscale');
[height,width] = size(A(:,:,1));

B=imread('Images\1.ppm');
image(B);title('original colour');
[heightc,widthc]=size(B(:,:,1));

%subplot of the original images
figure (1); title ('Original
images');
subplot (1,2,1); imshow (A); title
('original GS');
subplot (1,2,2);imshow(B); title
('original colour');

%********************DOWNSAMPLING
OF THE
IMAGE*************************

%this is the downsampling section
for all of the "colours" in the
image.
%The greyscale image means that
although the downsampling can be a
set
%standard across the whole image,
and therefore it is "easier" to
%downsample the image, it does
mean that some quality is lost
because you
%cannot choose what less dominant
colours you would downsample more.
for width_1=1:width/2
for height_1=1:height/2
   blue(height_1,width_1) =
mean(mean(A((height_1-
1)*2+1:height_1*2,(width_1-
1)*2+1:width_1*2, 3)));%just a
standard formula for calculating
the mean amount of pixels both
horizontally and vertically that
will be downsampled.
end
end

for width_1=1:width/2
for height_1=1:height/2
   green(height_1,width_1) =
mean(mean(A((height_1-
1)*2+1:height_1*2,(width_1-
1)*2+1:width_1*2, 3)));
end
end

for width_1=1:width/2
for height_1=1:height/2
   red(height_1,width_1) =
mean(mean(A((height_1-
1)*2+1:height_1*2,(width_1-
1)*2+1:width_1*2, 3)));
end
end

for width_2=1:widthc/8
for height_2=1:heightc/8
   bluecol(height_2,width_2) =
mean(mean(B((height_2-
1)*8+1:height_2*8,(width_2-
1)*8+1:width_2*8, 3)));%just a
standard formula for calculating
the mean amount of pixels both
horizontally and vertically that
will be downsampled.
end
end
```

```
%*****************DISPLAY OF THE
RGB VALUES OF THE
IMAGE******************

%output only the red channel of
the image A
figure ();
image(A(:,:,1)); title('red
channel GS');
col = zeros(256,3); %creating an
array that will be filled with
zeroes.
col(:,1) = 0:1/255:1;
colormap(col); caxis([0
255]);%setting the colormap of the
image to the red spectrum.

%output only the green channel of
the image A
figure ();
image(A(:,:,2)); title('green
channel GS');
col = zeros(256,3);
col(:,2) = 0:1/255:1;
colormap(col); caxis([0
255]);%setting the colormap of the
image to the green spectrum.

%output only the blue channel of
the image A
figure ();
image(A(:,:,3)); title('blue
channel GS');
col = zeros(256,3);
col(:,3) = 0:1/255:1;
colormap(col); caxis([0
255]);%setting the colormap of the
image to the blue spectrum.

figure ();
image(B(:,:,1)); title('red
channel colour');
col = zeros(256,3); %creating an
array that will be filled with
zeroes.
col(:,1) = 0:1/255:1;
colormap(col); caxis([0
255]);%setting the colormap of the
image to the red spectrum.

figure ();
image(B(:,:,2)); title('green
channel colour');
col = zeros(256,3);
col(:,2) = 0:1/255:1;
colormap(col); caxis([0
255]);%setting the colormap of the
image to the green spectrum.

figure ();
image(B(:,:,3)); title('blue
channel colour');
col = zeros(256,3);
col(:,3) = 0:1/255:1;
colormap(col); caxis([0
255]);%setting the colormap of the
image to the blue spectrum.

%*****************DISPLAY THE
DOWNSAMPLED RGB IMAGE
**********************

%output the down sampled blue
channel of the image A
figure();
image(blue); title('blue channel
GS, down-sampled');
col = zeros(256,3);
col(:,3) = 0:1/255:1;
colormap(col); caxis([0 255]);

figure();
image(red); title('red channel GS,
down-sampled');
col = zeros(256,3);
col(:,1) = 0:1/255:1;
colormap(col); caxis([0 255]);

figure();
image(green); title('green channel
GS, down-sampled');
col = zeros(256,3);
col(:,2) = 0:1/255:1;
colormap(col); caxis([0 255]);

figure();
image(bluecol); title('blue
channel colour, down-sampled');
col = zeros(256,3);
col(:,3) = 0:1/255:1;
colormap(col); caxis([0 255]);


% figure (); title('Image in YCbCr
Color Space');
% subplot (3,2,1);imshow(YCBCRB);
title ('YCrCb 1');
% subplot (3,2,2);imshow(YCBCRB);
title ('YCrCb 2');
% subplot (3,2,3);imshow(YCBCRC);
title ('YCrCb 3');

%saving the downsampled images,
this is the collective lossy
compression of
%the images.
imwrite(blue,
'downsampled_blue_GS.bmp','bmp');
imwrite(red,
'downsampled_red_GS.bmp','bmp');
```

```
imwrite(green,
'downsampled_green_GS.bmp','bmp');
imwrite (bluecol,
'downsampled_blue_colour.bmp',
'BMP');
%****************RECONSTRUCTION
OF THE
IMAGE*****************************

% reconstruct a complete image
from reduced blue, green and red
% it is then interpolated
linearly. Use of the interpolation
function is
% shown here.

%interpolation for the GS image
[hes,wis] = size(blue);
A(:,:,3) =
interp2([1:width/wis:width],[1:hei
ght/hes:height]',blue,[1:width],[1
:height]','linear');

[hes,wis] = size(red);
A(:,:,1) =
interp2([1:width/wis:width],[1:hei
ght/hes:height]',red,[1:width],[1:
height]','linear');

[hes,wis] = size(green);
A(:,:,2) =
interp2([1:width/wis:width],[1:hei
ght/hes:height]',green,[1:width],[
1:height]','linear');

%interpolation for the colour
image
[hes,wis] = size(bluecol);
B(:,:,3) =
interp2([1:widthc/wis:widthc],[1:h
eightc/hes:heightc]',bluecol,[1:wi
dthc],[1:heightc]','linear');

%******************DISPLAY THE
RECONSTRUCTED
IMAGE**********************

image(A); title('compressed image
GS final');

image (B); title ('compressed
image colour final');

figure (); title ('Original
images');
subplot (1,2,1); imshow (A); title
('final decompressed GS');
subplot (1,2,2);imshow(B); title
('final decompressed colour');

%******************COMPRESSION
RATIO
**********************************
**

%getting the size of the original
image in the three dimensions of
data
%that is is in; 768*1024*3
[xogs,yogs,zogs]=size(A);

%Getting the size of the three
indivdually compressed colour
channels.
[xrgs,yrgs,zrgs]=size(red);
[xbgs,ybgs,zbgs]=size(blue);
[xggs,yggs,zggs]=size(green);

%multiplying the x,y and z values
to get the size of the image
overall
sizeo=xogs*yogs*zogs;
sizec=((xrgs*yrgs*zrgs)+(xbgs*ybgs
*zbgs)+(xggs*yggs*zggs));

%formula for the compression ratio
which is then printed.
CompressionratioGS=(sizec/sizeo);
fprintf("compression ratio for GS:
%0.4f\n", CompressionratioGS);
   %********END OF THE RGB TEST
**************
```