Engineering Assignment Coversheet

Please note that you:

- Must keep a full copy of your submission for this assignment
- Must staple this assignment
- Must NOT use binders or plastic folders except for large assignments

Student Number(s)
819870 807780
Group Code (if applicable): 2

Assignment Title:	transform coding in image and video
Subject Number:	ELEN90051
Subject Name:	ad communication system
Student Name:	Yuetao Wu & Junbin Zhong
Lecturer/Tutor:	
Due Date:	

For Late Assignments Only

Has an extension been granted? Yes / No (circle)

A per-day late penalty may apply if you submit this assignment after the due date/extension. Please check with your Department/coordinator for further information.

Plagiarism

Plagiarism is the act of representing as one's own original work the creative works of another, without appropriate acknowledgment of the author or source.

Collusion

Collusion is the presentation by a student of an assignment as his or her own which is in fact the result in whole or in part of unauthorised collaboration with another person or persons. Collusion involves the cooperation of two or more students in plagiarism or other forms of academic misconduct.

Both collusion and plagiarism can occur in group work. For examples of plagiarism, collusion and academic misconduct in group work please see the University's policy on Academic Honesty and Plagiarism: http://academichonesty.unimelb.edu.au/

Plagiarism and collusion constitute cheating. Disciplinary action will be taken against students who engage in plagiarism and collusion as outlined in University policy. Proven involvement in plagiarism or collusion may be recorded on my academic file in accordance with Statute 13.1.18.

STUDENT DECLARATION

Please sign below to indicate that you understand the following statements:

I declare that:

- This assignment is my own original work, except where I have appropriately cited the original source.
- This assignment has not previously been submitted for assessment in this or any other subject.

For the purposes of assessment, I give the assessor of this assignment the permission to:

- Reproduce this assignment and provide a copy to another member of staff; and
- Take steps to authenticate the assignment, including communicating a copy of this assignment to a checking service (which may retain a copy of the assignment on its database for future plagiarism checking).

Student signature Yuetao Wu Date 19/05/2018

Advanced Communication Systems, ELEN90051

Workshop week 4: Transform coding in image and video

Junbin Zhong 819870 18-Mar-18

Yuetao Wu 807780 18-Mar-19

5. How JPEG works (20 marks)

a) Download the test image "KIsealion_640.bmp" from the LMS. How big is the file? How many bits are used to store each pixel? Set the current directory in MATLAB to be the same directory as the one in which you stored the image.

The file is 900k Bytes, so there are 7200k bits involved. The image is 640x480, so 307200 pixels are involved. Hence, the number of bits per pixel is 7200k/307.2k=24 bits/pixel, which is not surprising because 8 bits per color channel are assigned for a pixel, and there are three channels, R, G and B.

b) Use the imread operation to load the image into an array in MATLAB:

>> i_array = imread('KIsealion_640.bmp', 'bmp');

c) Use the size function to obtain the dimensions of the array. Explain the numbers you obtain.

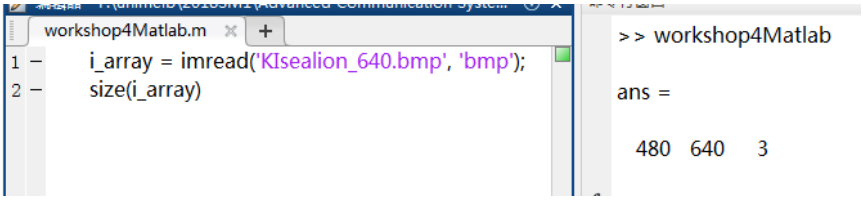

Fig.1 Size of BMP

The number 480 indicates the number of pixels per column, and the number 640 indicates the number of pixels per row. The number 3 indicates the number of bytes per pixel, and each byte represents the value for one of the R, G and B channels.

d) Now save the image to JPEG form using different "quality" settings in imwrite. The quality parameter can be anything from 1 to 100, so to save at highest quality you would use a command like:

>> imwrite(i_array, 'KIsealion_Q100.jpg', 'jpg', 'quality', 100);

e) Repeat this for some different quality settings, then open the output image files (outside of MATLAB; just double-click them) and look at the resulting images. What do you see? Compare the file sizes with the quality settings you used to create them and the visual quality of the resulting images. What quality setting1 is the smallest necessary to provide what you would describe as a "good" or "acceptable" quality image? What quality setting gives distortion which is visible, but not too extreme?

Fig.2 JPG with quality 1 Fig.3 JPG with quality 30

Quality	1	5	10	20	30	40	50	100
Size(KB)	6	7	8	11	13	16	18	167
visual	BAD	BAD	SO-SO	SO-SO	GOOD	GOOD	GOOD	PERFECT

The smallest quality setting is 30, which gives a good quality image. The qualities from 10 to 20 give visible but not extreme distortion.

f) Take a subset of the (original) image, and plot a 1-D horizontal slice through it for one of the 3 colour components (say, "R")2.

>> figure

>> plot(i_array(150,1:640,1), 'r');

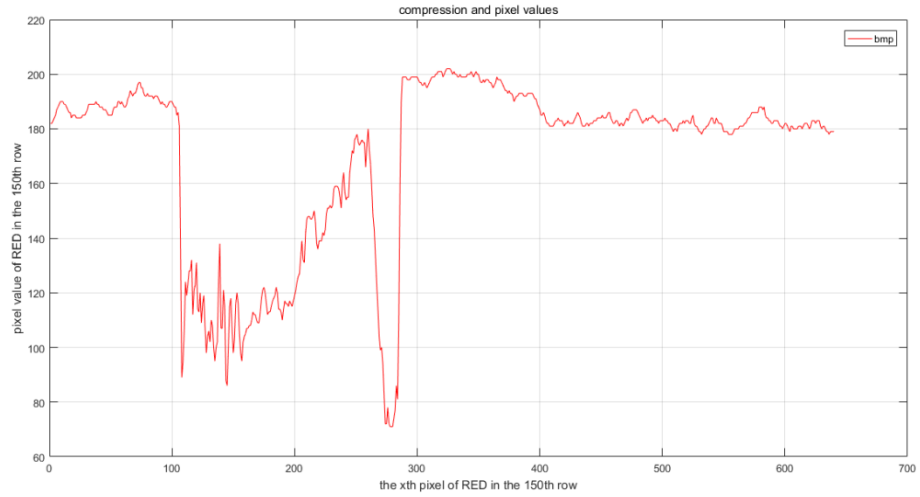

Fig.4 pixel values of red in 150th row in bmp

g) Repeat this for the corresponding pixels in the most highly compressed versions of the test image. (That is, load the compressed files back into MATLAB in new arrays, and make the same plot in a different colour. You'll need to use the "hold on" command to keep an earlier plot on the same figure.) How do you explain the results you obtain, in terms of what you know about how JPEG compression works? You may wish to change the location of the slice (i.e. other than line 150) and the number of samples plotted (i.e. less than 640) to help you see what is going on.

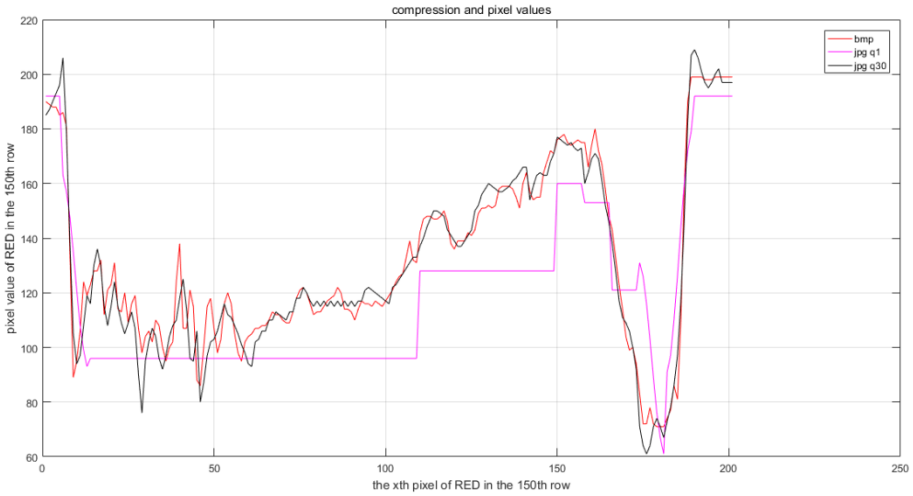

Fig.5 pixel values of red in 150th row in bmp, jpg with Q1, jpg with Q30

The JPG compression smoothes the high spatial frequencies so that only low frequency values are transmitted, which results in fewer transmitted bits. As shown in the picture, the low quality compression, where Q equals 1, has many flat intervals, while higher quality compression remains most of the high spatial frequencies and traces the original bmp data very well.

h) Plot the other two colour components ("G" and "B") along with "R" on the one graph. Can you relate the different levels of each colour to the appearance of the picture at the point where you've taken a slice through it? Do you see the same compression process carried out on each colour component?

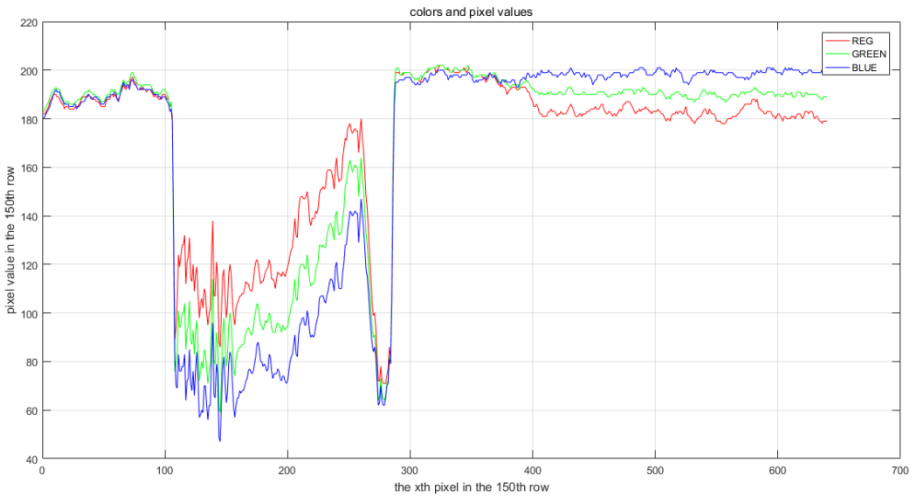

Fig.6 pixel values of RED, GREEN and BLUE in the 150th row in bmp

Fig.7 image segment around 150th row

The image shows a segment of the original picture from the 145th row to 155th row, which

graphically indicated the color distribution. From the 100th column to the 300th column, the segment contains the face of the sea lion, which is mainly brown. Hence, the major color component of this part is red, which is also shown in the graph above. The background sky is a mixture of grey and light blue, so the distribution of three different color channel is roughly even, with a slight advantage in the blue and green color.

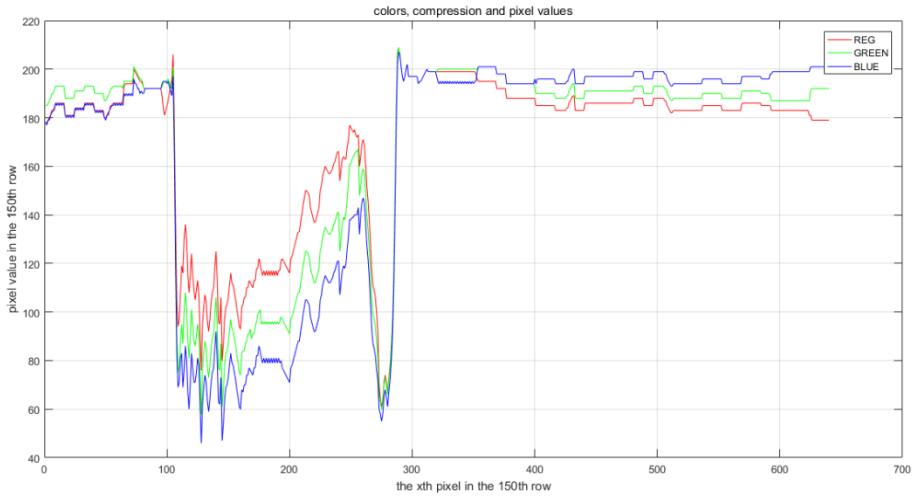

Fig.6 pixel values of RED, GREEN and BLUE in the 150th row in JPG with Q30

The same compression process is carried out, because the trends in each color are roughly the same.

6. Dependence on images (20 marks)

a) Choose a level of compression you found in 5e above, for which the image suffered some visible distortion, but the distortion was not too extreme. What value of "quality" did you use? What was the file size? How many bits per pixel (that is, file size divided by number of pixels) were required to represent the picture? How does this compare to the original?

Fig.7 JPG with quality 10

Quality	File size	Bits per pixel	Bits per pixel (origin)
10	8KB	0.21	24

b) Now repeat the process of loading a picture into MATLAB then writing it out at different compression rates (different values of "quality"), but use some of the additional test pictures: "Aerobatics_640.bmp", "Bowlcrowd_640.bmp", "Perth_640.bmp", "Heronwreck_640.bmp", "Obama_640.bmp" and "Stinson_640.bmp". (Use your experience with the first image to be selective in the number of "quality" values you try for each of these additional ones; don't try to do an exhaustive series of tests for every image. You also do not need to use all the above images if you think you understand what is going on and can answer the questions using just some of them. Include at least "Perth_640" and "Stinson_640", though.) For the same values of "quality", do you think the picture quality (or visible coding distortion) is the same for different pictures? How does it differ between pictures? Do you think you could save some pictures with a smaller number of bits per pixel (bpp) than others, without it appearing significantly worse? What characteristics of pictures make them "hard" to compress (that is, require more bits for a given quality), and what characteristics make them easier to compress? Can you relate this back to what you know about how JPEG works?

Fig.8 JPGs with Q10

The picture qualities are not the same for the same quality factor. Although the sea lion looks good with quality equal to 10, the Stinson and the Perth suffer from severe perceivable distortion in the background. This is due to the perceivable levels of human eyes are pretty high the medium-low regions. Hence, the compression of the gradually changing background is very noticeable. However, when the spatial frequency goes up as in the bowl crowd image, the distortion is less perceivable.

I think I could save some pictures with a smaller number of bits per pixel (bpp) than others, without it appearing significantly worse.

The low spatial frequencies in chrominance and the medium-low spatial frequencies make it

difficult to compression, while high spatial frequencies make it easy.
In the JPG compression, we assign more quantization levels to the more perceivable frequencies.

7. Inter-frame video coding (30 marks)

a) Test images "MonashRd1_640.bmp" and "MonashRd2_640.bmp" are two pictures of the same (moving) scene taken a very short time apart. (Think of them as being two consecutive frames of a movie.) Load these images into arrays in MATLAB. You will see that the resultant arrays are 2 (instead of 3) dimensional. Have a look at the images to see why.

Because they are black and white pictures, which only have one channel of luminance values, rather than 3 channels of RGB values.

b) Take the difference between these images. We need to convert the image array data to type int16 to avoid problems with negative numbers in uint8.

```
>> f1int8_array = imread('MonashRd1_640.bmp', 'bmp');
>> f2int8_array = imread('MonashRd2_640.bmp', 'bmp');
>> int16_array1=int16(f1int8_array);
>> int16_array2=int16(f2int8_array);
>> int16_diff = int16_array2 - int16_array1 + 128;
```

In the last line of the above sample MATLAB code, we add an offset of 128. Why do you think we do this?

That is because we want to increase the brightness so that the details in the image are more perceivable to the eye. Otherwise, the differential image will be black for most of the area.

c) Convert the difference array back to an 8 bit array and export it as an image:

```
>> uint8_diff = uint8(int16_diff);
>> imwrite(uint8_diff,'DiffImage.bmp','bmp');
```

d) Open the file and have a look at it on the computer screen. What do you see, and can you explain it?

Fig.9 differential image

This is the difference between the two frames. The grey area indicates there is no change in those areas, while the scattered dots and lines indicate differences.

e) Now try compressing the offset difference array (uint8_diff in the above example) using JPEG coding, and adding the compressed picture back to the first image (int16_array1 in the above example code), before outputting it as a bitmap file and viewing it. (Hint: be careful to manage data types, and don't forget to remove the offset (128) when you combine the pictures back

together.) You will find that you need a high quality setting with JPEG to get a reasonable reconstruction3, and you will see artefacts in the image even with quality=100.
Although we are not focused on coding efficiency here (for the reasons discussed in the footnote), does the process you have carried out suggest a means of saving bits when encoding a sequence of similar video frames?
Some objects in images "MonashRd1" and "MonashRd2" are almost identical. However, because they have moved, we end up with some large difference signals that need to be encoded. This seems a bit wasteful, because we already have information about these objects in the earlier picture. (The objects are there, but in a different place to the second frame.) How could we describe the changes between our consecutive pictures, in a way that might mean we need fewer bits to describe the second one?


Fig.10 MonashRd2_640.bmp


Fig.11 reconstructed MonashRd2

Yes.
To save bits when coding a sequence of similar frames, we can exploit the difference between the two frames. After that, we can encode only the differential image, which contain less high spatial frequency components. To reconstruct the original image, we can add up the differential image and the preceding image.
We can describe the changes between two consecutive images with a motion vector. We can segment an image into blocks, and then, for one particular block in the preceding image, we search a similar block in the neighborhood of the following image. After that, we represent the location change with a vector, which indicates the shifting of objects in two consecutive images. In this way, we can save bits by finding and not encoding similar objects in the two images.

8. Applications (20 marks)
Though in a more sophisticated form, interframe coding such as you have been exploring in part 7 gives significant bit rate savings in many situations. For which of the following applications do you think it would be suitable, and why?
a) Very high quality applications such as Digital Cinema.
b) Video editing (where video sequences are spliced and joined).
c) Videoconferencing
d) Television broadcasting

It is not suitable for digital cinemas, because the screen is so big that even a slight distortion in a block will be noticeable. In addition, people pay money to enjoy high resolution movies, it is not practical to compression a blue ray movie.

It is not suitable for video editing, because JPG exploits the similarity in consecutive frames. When different frames are put together, it becomes less predictable.
It is suitable for video conference, because we don't care the details in people's face or other high spatial frequency components, as long as people can communicate effectively with each other.
It is suitable for TV broadcast, because it is real-time communication, which requires higher speed to maintain the consecutive images.

Appendix

Part 5:

```
i_array = imread('KIsealion_640.bmp', 'bmp');

% read the KIsealion_640.bmp image
bmp_size = size(i_array);
%{
imwrite(i_array, 'KIsealion_Q100.jpg', 'jpg', 'quality', 100);
imwrite(i_array, 'KIsealion_Q50.jpg', 'jpg', 'quality', 50);
imwrite(i_array, 'KIsealion_Q40.jpg', 'jpg', 'quality', 40);
imwrite(i_array, 'KIsealion_Q30.jpg', 'jpg', 'quality', 30);
imwrite(i_array, 'KIsealion_Q20.jpg', 'jpg', 'quality', 20);
imwrite(i_array, 'KIsealion_Q10.jpg', 'jpg', 'quality', 10);
imwrite(i_array, 'KIsealion_Q5.jpg', 'jpg', 'quality', 5);
imwrite(i_array, 'KIsealion_Q1.jpg', 'jpg', 'quality', 1);
%}

figure(1);

% plot the image in R channel
plot(i_array(150,100:300,1), 'r');
grid on;
hold on;
xlabel('the xth pixel of RED in the 150th row');
ylabel('pixel value of RED in the 150th row');
title('compression and pixel values');
jpg_Q1 = imread('KIsealion_Q1.jpg', 'jpg');
% read the jpeg version of the image
plot(jpg_Q1(150,100:300,1), 'm');
jpg_Q30 = imread('KIsealion_Q30.jpg', 'jpg');
plot(jpg_Q30(150,100:300,1), 'k');
legend('bmp','jpg q1','jpg q30');
```

```
figure(2);
plot(i_array(150,1:640,1), 'r');
grid on;
hold on;
xlabel('the xth pixel in the 150th row');
ylabel('pixel value in the 150th row');
title('colors and pixel values');
plot(i_array(150,1:640,2), 'g');
plot(i_array(150,1:640,3), 'b');
legend('REG','GREEN','BLUE');
% plot the R G B channel of the image

figure(3);
imshow(i_array(145:155,1:640,1:3));

figure(4);
plot(jpg_Q30(150,1:640,1), 'r');
grid on;
hold on;
xlabel('the xth pixel in the 150th row');
ylabel('pixel value in the 150th row');
title('colors, compression and pixel values');
plot(jpg_Q30(150,1:640,2), 'g');
plot(jpg_Q30(150,1:640,3), 'b');
legend('REG','GREEN','BLUE');
```

Part 6:

```
i_array2 = imread('Perth_640.bmp', 'bmp');
% read the image.
bmp_size2 = size(i_array2);

imwrite(i_array2, 'Perth_Q100.jpg', 'jpg', 'quality', 100);
imwrite(i_array2, 'Perth_Q20.jpg', 'jpg', 'quality', 20);
imwrite(i_array2, 'Perth_Q10.jpg', 'jpg', 'quality', 10);
imwrite(i_array2, 'Perth_Q1.jpg', 'jpg', 'quality', 1);
% rewrite the image in different quality (100,20,10,1)

i_array3 = imread('Stinson_640.bmp', 'bmp');
bmp_size3 = size(i_array3);
% load the image
imwrite(i_array3, 'Stinson_Q100.jpg', 'jpg', 'quality', 100);
imwrite(i_array3, 'Stinson_Q20.jpg', 'jpg', 'quality', 20);
imwrite(i_array3, 'Stinson_Q10.jpg', 'jpg', 'quality', 10);
```

```
imwrite(i_array3, 'Stinson_Q1.jpg', 'jpg', 'quality', 1);
% rewrite the image in different quality (100,20,10,1)

i_array4 = imread('BowlCrowd_640.bmp', 'bmp');
bmp_size4 = size(i_array4);
% load the image

imwrite(i_array4, 'BowlCrowd_Q100.jpg', 'jpg', 'quality', 100);
imwrite(i_array4, 'BowlCrowd_Q20.jpg', 'jpg', 'quality', 20);
imwrite(i_array4, 'BowlCrowd_Q10.jpg', 'jpg', 'quality', 10);
imwrite(i_array4, 'BowlCrowd_Q1.jpg', 'jpg', 'quality', 1);

% rewrite the image in different quality (100,20,10,1)
```

Part 7

```
f1int8_array = imread('MonashRd1_640.bmp', 'bmp');
f2int8_array = imread('MonashRd2_640.bmp', 'bmp');
% load the image

int16_array1=int16(f1int8_array);
int16_array2=int16(f2int8_array);

int16_diff = int16_array2 - int16_array1 + 0;

uint8_diff = uint8(int16_diff);
% change it to 16bits to avoid negetive value
%imwrite(uint8_diff,'DiffImage.bmp','bmp');

imwrite(uint8_diff, 'DiffImage_Q100.jpg', 'jpg', 'quality', 100);
diffint8_array =  imread('DiffImage_Q10.jpg', 'jpg');
int16_arraydiff=int16(diffint8_array);
int16_combine = int16_arraydiff + int16_array1;
uint8_combine = uint8(int16_combine);
imwrite(uint8_combine,'CombImage.bmp','bmp');
```