**EBU6230 – Image and Video Processing – 2022/23**

**Coursework report and exercises**

Name: \_\_\_\_Zhengxiao Wu\_\_\_\_\_

Username: \_\_\_\_\_200978936\_\_\_\_\_\_\_

# Exercise 1 (a)

**Reading/writing PGM/PPM images**: The first step towards image and video processing is reading images from a file and write them to a file. There exist different standards that store the information in different formats; so before opening an image, knowledge of the standard is necessary.

Two widely used image formats are PPM and PGM. The PGM format is a greyscale file format designed to be easy to manipulate. A PGM image represents a greyscale graphic image. For most purposes, a PGM image can just be thought of as an array of integers. The name "PGM" is the acronym of "Portable Grey Map." The name "PPM" is the acronym for "Portable Pixel Map." Images in this format (or a precursor of it) were once also called "portable pixmaps." It is a highly redundant format, and contains a lot of information that the Human Visual System (HVS) cannot even discern. However, as for PGM, PPM is very easy to write and analyse.

The goal of the first part of today’s lab is to become comfortable with these two formats. You will implement functions to read and to write PPM and PGM images. The final demonstration of the implemented software will be done using the well-known test images: LENA, BABOON, PEPPERS, etc. You can find PPM and PGM versions of these images in the EBU6230 QMplus pages. The writing function must add as a comment in the header: “image created by *your\_name*”.

Include in your submission the file resulting from reading the images provided and writing them back in their original format.

Summarize in 5 points the operations necessary to read a PGM/PPM image:



Summarize in 5 points the operations necessary to write a PGM/PPM image:



What is the difference between the identifiers P3 and P6?

# Exercise 1 (b)

**Format conversions:** in this part of the lab, the images will be converted from colour to grey scale; in other words a PPM image will be converted to the PGM format. You will implement a function called “BUPT\_format\_converter” which transforms images from colour to grey-scale using the following YUV conversion:

Y = 0.257 \* R + 0.504 \* G + 0.098 \* B + 16

U = -0.148 \* R - 0.291 \* G + 0.439 \* B + 128

V = 0.439 \* R - 0.368 \* G - 0.071 \* B + 128

Note swap of 2nd and 3rd rows, and sign-change on coefficient 0.368

What component represents the luminance, i.e. the grey-levels, of an image?

Use thee boxes to display the results for the colour to grey-scale conversion.

Lena grey

Lena colour (RGB)

Baboon grey

Baboon colour (RGB)

Is the transformation between the two colour-spaces linear? Explain your answer.

Display in the box the Lena image converted to YUV 3 channels format.

Are the colours of the previous picture distorted? If yes why?

Based on the formula for the RGB to YUV conversion, derive the formula for the YUV to RGB conversion.

Use the formula you derived at the previous step to convert the YUV image back to the original RGB format. Display the result in the box.

# Exercise 1 (c)

**Sub-sampling**: The HVS is incapable of perceiving certain details in an image. Therefore high compression ratios can be achieved by exploiting the characteristics of the HVS, thus discarding what has a low visual relevance. However, this process can introduce distortions due to the compression. A simple way to exploit the characteristics of the HVS to give compression is to sub-sample an image. A drawback of this approach is that it is possible to incur the well-known problems of a discrete representation, such as aliasing. This part of the lab covers some simple sub-sampling operations.

Implement a function that sub-samples grey level images by a factor n, with n a multiple of 2. The function should be able to sub-sample independently in the horizontal and in the vertical direction or in both directions at the same time.

Display the results of sub-sampling the image Lena using the following factors: 2 horizontal, 2 vertical, 2 vertical and 8 horizontal, 4 vertical and 4 horizontal. Include the files of the results in the submission.

Box for the 4 images

Describe, using your own words, the aliasing problem and how to avoid it, as applied to signal processing

Given a scene sampled by a ccd sensor with minimum horizontal sampling frequency 10cm-1, what is the maximum horizontal frequency in the image that can be correctly represented?

If you sub-sample an image, why do you have more problems from aliasing?

Paste below a clear example of artefacts generated by aliasing. For this task you can use your own choice of image. Use the box below for the image and comments.

# Exercise 2 (a)

**Quantize:** Quantization is the process of approximating the continuous values in the image data with a finite set of discrete values. The input of a quantizer is the original data and the output is one among the finite number of levels. This process is an approximation, and a good quantizer is one which represents the original signal with minimum loss (quantization error). In this lab, you will work with a scalar uniform quantizer applied to grey-scale images.

Implement a function that uniformly quantizes grey level images. The function will allow the reduction of the number of grey level values by a given factor n (a power of 2).   
*Note*. To visualize the image, you need to re-map it in the 8-bit-per-pixel representation. Show the results in the boxes below.

Lena, quantization factor 2

Baboon, quantization factor 8

Peppers, quantization factor 128

Peppers, quantization factor 32

Is quantization a reversible process? Can you recover what you discarded? Briefly explain.

Write the results back to PGM/PPM files using the function you created. Make sure that your writing function allocates the correct number of bits per pixel. What is the size of the files compared with the original? Given the results, what is a typical application field for quantization? Include in your submission the output files and comment on the results.

# Exercise 2 (b)

**Histograms:** This part of the lab is dedicated to image processing using histograms. A histogram is a statistical representation of the data within an image. The histogram can be represented as a plot of the frequency of occurrence of each grey level. This representation shows the distribution of the image data values. By manipulating a histogram, it is possible to improve the contrast in an image and the overall brightness or to segment different areas of the image by applying one or more thresholds to the histogram itself.

Implement a function to output the histogram values of a given grey level image. Display in the boxes the resulting histograms.

|  |  |  |
| --- | --- | --- |
| Lena | Baboon | Peppers |

If you normalize the values of the histogram so that they sum to 1, what does the value of a bin represent?

# Exercise 2 (c)

**Equalize:** Equalization is one of the possible image processing algorithms implemented using histograms. Histogram equalization allows a user to enhance the contrast of images. Histogram equalization employs a monotonic, non-linear mapping which re-assigns the intensity values of pixels in the input image such that the output image contains a uniform distribution of intensities (i.e. a flat histogram).

Implement a function that equalizes grey-scale images based on their histogram. The input is a given grey level image; the output is the derived image with uniform intensity distribution.

Display in the boxes the equalized images and their histograms.

|  |  |  |
| --- | --- | --- |
| Lena equalized | Baboon equalized | Peppers equalized |

|  |  |  |
| --- | --- | --- |
| Lena histogram after equalization | Baboon equalized after equalization | Peppers equalized after equalization |

Are the distributions really uniform? Explain your results.

Show an example of the successful application of histogram equalization to image enhancement. You can use an appropriate image of your choice

Original image

Enhanced image

Comment on the results of the previous step.

# Exercise 2 (d)

**Histogram modelling:** Histogram modelling techniques are effective tools for modifying the dynamic range and contrast of an image. Unlike [contrast stretching](http://homepages.inf.ed.ac.uk/rbf/HIPR2/stretch.htm), histogram modelling operators may employ non-linear and non-monotonic transfer functions to map between [pixel intensity values](http://homepages.inf.ed.ac.uk/rbf/HIPR2/value.htm) in the input and output images. In the first part of this lab you will model the histogram of a grey-scale image.

Implement a point to point operation that maps the input grey level image into an output image which has a predefined frequency distribution. The algorithm is not given explicitly in the lecture slides, you are supposed to derive it. Use as input histogram the histogram of an image A and model the histogram of another image B according to the input.

(A = Lena) (B=Peppers)

Image A

Histo of A

Image B

Histo of B

B after Modelling

Histo of B after modelling

Use as input histogram an approximation of the exponential distribution**.**

Image Peppers

Histo of Peppers

Peppers after modelling

Histo after modelling

Write in the box the formulation of your algorithm.

# Exercise 3 (a)

**Negatives:** We are used to the negative of an image in analogue image processing. It is possible to generate a negative from a digital image too. In the last part of today’s lab you will solve a simple exercise on negative images.

Write a function that inverts the grey level of a PGM image (i.e. it creates the negative of the image).

Negative of Baboon

Negative of Peppers

Negative of Lena

Perform the same task with PPM images and comment on the results.

Negative of Lena

Negative of Peppers

Negative of Baboon

Your comments:

# Exercise 3 (b)

**Rotation and translation.** Image processing toolboxes allow a user to rotate, translate and skew images. These are very useful operations for image composition, for example. The first exercise will cover the implementation of two such transformations.

Write a function *BUPT\_transform* that takes as input an image *I*, rotates it with an angle θ1 and skews it with a second angle, θ2.

Write the matrix formulation for image rotation (define all variables).

Write the matrix formulation for image skewing (define all variables).

Create and paste below a PGM image containing your name written in Arial font, point 72, uppercase letters.

Your image

Rotate the image you created by 30, 60 120 and -50 degrees clockwise and display the results below.

-50

120

60

30

Skew the same image by 10, 40 and 60 degrees and display the results below.

60

40

10

During the development process have you experienced the problem of regular patterns of black pixels in the image? If so, explain how you solved the problem. Otherwise imagine what could have generated these artefacts and how you would have worked around them.

Rotate the image by 20 degrees clockwise and then skew the result by 50 degrees. Display the result in ‘a’.

Skew the image by 50 degrees and then rotate the result by 20 degrees clockwise. Display the result in ‘b’.

|  |  |
| --- | --- |
| a | b |

Analyse the results when you change the order of the two operators i.e. R(S(*I*)) and S(R(*I*)), where R is the rotation and S is the skew. Are the results of (a) and (b) the same? Why?

# Exercise 4 (a)

**Noise and PSNR:** This part of the lab introduces error metrics for image quality evaluation. Two common metrics are the Mean Square Error (MSE) and the Peak Signal to Noise Ratio (PSNR). The MSE is the cumulative squared error between the processed image and the original image. The PSNR makes use of the MSE. The smaller the MSE, the smaller the error is. The larger the PSNR, the smaller the error is. You will add different amounts of random noise to the test images and measure their MSE and PSNR.

Write the formulas of MSE and PSNR.

Can the PSNR return a negative value? Explain your answer.

Create a function that can add (i) salt and pepper noise and (ii) Gaussian noise to a PGM image and compute PSNR and MSE. Show the results in the box and write under the box the values of MSE and PSNR comparing the original with the corrupted one.

Peppers Gaussian noise

σ =5% of the range.

Baboon Gaussian noise

σ =10% of the range.

Baboon salt and pepper noise

Peppers salt and pepper noise

Lena Gaussian noise

σ =1% of the range.

Lena salt and pepper noise

Lena Gaussian noise

σ =5% of the range.

Peppers Gaussian noise

σ =2% of the range.

Baboon Gaussian noise

σ =7% of the range.

# Exercise 4(b)

**Up-sampling:** Scaling-up an image (up-sampling) requires the filling of the new positions given the original pixels. This filling can be obtained by interpolation. Different interpolation techniques can be used. The choice depends on the quality we want to achieve and on the computation resources we have available. The nearest-neighbour interpolation is the simplest and fastest technique, but it is also a technique achieving low quality results. Bilinear interpolation is computationally more intensive, but it achieves higher quality results.

Implement the function *BUPT\_up* that increases the resolution of images by a given factor (also a non-integer one). The up-sampling should be achieved using the nearest neighbour as well as the bilinear interpolation. The function will be able to up-sample independently in the horizontal and in the vertical direction or in both directions simultaneously.

Up-sample the image Lena using nearest neighbour interpolation. Display a blow-up of the image Lena obtained by up-sampling the original image with factor 4.5. The image should clearly show the type of artefact obtained using the nearest neighbour interpolation. Use the box below to display the image and discuss the results.

**Your comments:**

Up-sample the image Baboon using bilinear interpolation. Paste below a zoomed portion of the image Baboon obtained by up-sampling the original image with a factor 3.6. Discuss the artefacts obtained using bilinear interpolation.

**Your comments:**

Compare the nearest neighbour technique and the bilinear technique in terms of speed and accuracy. Which technique is faster? Why? What artefacts are more visually disruptive?

# Exercise 5(a)

**Low pass filtering:** Image filtering generates a processed image as a result of certain operations on the pixels of the original image. Each pixel in the output image is computed as a function of one or several pixels in the original image, usually located near the output pixel. The procedure is usually implemented by convolving a kernel with desired properties with the pixels of the input image. If the kernel is a Gaussian kernel, then the behaviour of the filter depends on the variance of the Gaussian.

Write a function BUPT\_lowpass that convolves an image with a Gaussian kernel.

(i) Write the formula of the kernel you used.

(ii) The 2D Gaussian kernel is separable: write the two separate equations for the rows and the columns, and discuss the advantages of using separable filters.

(iii) What is the relationship between σ and the cut-off frequency of the filter?

(iv) Given σ, what criterion should be used to choose the size of the kernel? Why?

**Your comments:**

Add Gaussian noise to the image Lena with noise power 50 dBm, then convolve the noisy image with Gaussian kernels with σ = 0.5, 1, 2, 4, 7, 10, respectively. Paste below the resulting images. Comment the results obtained with increasing values of σ.

σ=1

σ=2

σ=4

σ=10

σ=7

σ=0.5

**Your comments:**

Implement a rectangular-shaped filter (*BUPT\_rect*). Filter the noisy image Lena with a 5-by-5 and with a 7-by-7 kernel. Paste below the resulting images. Compare these results with those obtained with the Gaussian filter.

5-by-5 pixel

7-by-7 pixel

**Your comments:**

# Exercise 5(b)

**Edge Detection:** Edge detection is the process of identifying and locating discontinuities in an image. The discontinuities are sharp changes in pixel intensity which characterise object boundaries. Classical edge detectors convolve the image with a 2-D kernel designed to be sensitive to large gradient amplitudes. There exist a large number of edge detectors, each designed to be sensitive to certain types of edges.

Implement the Sobel, Roberts and Prewitt filters for grey level and colour images. In the case of colour images, you can apply separate filtering of each of the three RGB components. Paste below the images representing the absolute value of the gradient for the three filters. Comment on how you dealt with the borders.

Sobel Peppers (colour)

Sobel Lena (grey)

Roberts Peppers (colour)

Roberts Lena (grey)

Prewitt Peppers (colour)

Prewitt Lena (grey)

**Your comments:**

# Exercise 6

**LoG.** Laplacian filters are derivative filters used to find edges in images. Since derivative filters are very sensitive to noise, it is common to smooth the image before applying the Laplacian. For example, the image can be smoothed using a Gaussian filter. The two-step process involving Gaussian low-pass filtering followed by Laplacian filtering is called the Laplacian of Gaussian (LoG) operator and will be covered in the first part of the lab.

Implement the LoG operator as a parametric function of the variance, and display the results in the boxes.

Peppers σ = 3

Lena σ = 1

Try the effect of LoG filters using different Gaussian widths by changing the variances. What is the general effect after increasing the Gaussian width?

Baboon σ = 3

Baboon σ = 2

Baboon σ = 1

Add your comments here

Construct a LoG filter where the mask size is too small for the chosen Gaussian width (*i.e.* the LoG becomes truncated). What is the effect on the output? Define an empirical or analytical rule to determine how large a LoG mask should be in relation to the variance of the underlying Gaussian if severe truncation is to be avoided.

Lena result

σ =?

Kernel size = ?

Peppers result

σ =?

Kernel size = ?

Discussion