EBU723U – Image and Video Processing – 2016/17

Coursework report and exercises

Name: _____SHI Yuqi_____

Username: _____JP2013212998_____

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 EBU723U 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:

1. Open the file retrieved from the given path
2. Ignore the comments in the file
3. Judge if the file is encoded in ASCII or binary by the magic identifiers
4. Read the image width, height and color levels information and store them in a matrix

   The dimension of the matrix is determined by height and width.

5. Recover the image from the matrix: imshow () function (ps. If the image is encoded in ASCII, we need to divide each pixel grey level by 255 to maintain the number between 0 and 1)

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

1. Input Matrix information to write in file (containing dimension, grey levels and mode)
2. Determine the image's encoding method by defining its magic identifiers
3. Open the file in read mode and print out the matrix information
4. Store the grey level information in each pixel
5. Output the file of outascii.pgm or outbinary.pgm

What is the difference between the identifiers P3 and P6?

P3 and P6 are magic identifiers for PPM image.

If it is "P3" then the image is given as ASCII text, the numerical value of each pixel ranges from 0 to the maximum value given in the header. The lines should not be longer than 70 characters.

If it is "P6" then the image data is stored in byte format, one byte per colour component (r,g,b). "P6" PPM files can only be used for single byte colours. "P6" image files are obviously smaller than "P3" and much faster to read.

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?

Y component represents the luminance of an image

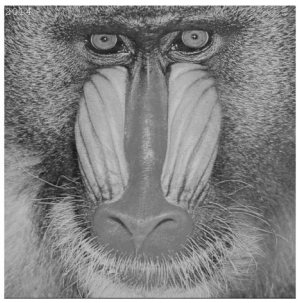
Use boxes to display the results for the colour to grey-scale conversion: imshow(YUV);

Lena colour (RGB)

Lena grey

Baboon colour (RGB)

Baboon grey



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

The transformation between space RGB and space YUV are linear. For one YUV vector is matched with one RGB vector.

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.140 \\ 1 & -0.395 & -0.581 \\ 1 & 2.032 & 0 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix}$$

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

YUV channel

U channel

V channel

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

The colours are distorted in the previous picture.

YUV encodes a colour image taking into account properties of the human eye that allow for reduced bandwidth for Chroma components without perceptual distortion.

For people is more sensible to luminance than the chrominance factor, so we compress the colour factor and the previous picture is distorted in colour.

YUV model is used for TV while the RGB is created for TV, when we show YUV image on PC it distorted.

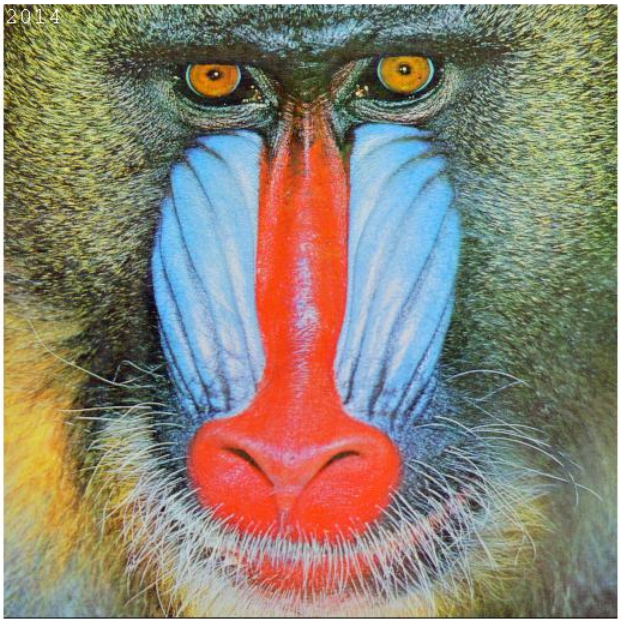
Based on the formula for the RGB to YUV conversion, derive the formula for the YUV to RGB conversion.

R=1.164*(Y-16) +1.596*(V-128);

G=1.164*(Y-16)-0.813*(V-128)-0.391*(U-128);

B = 1.164*(Y-16) + 2.018 *(U-128);

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

2 horizontal

2 vertical

2 vertical and 8 horizontal

4 vertical and 4 horizontal

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

When we want to recover the signal from the sampled signal, the aliasing problem may arise when the points we own can be recovered as various kinds of wave form, we don't know which is the original wave form then the aliasing problem arises.It always happen when the sample frequency is not high enough.

We need to choose a larger sample frequency, which is more than twice of the highest frequency of the signal. Or we can decrease the highest sample frequency.

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?

Fs =$10cm^{-1}$, Fmax = Fs / 2, so the maximum horizontal frequency is $5cm^{-1}$

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

The number of samples reduces when we do subsampling operation, the resolution is then becomes worse, there would definitely 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.

factor(8,8)

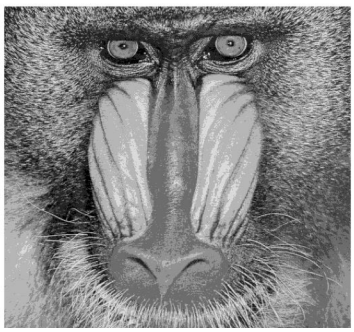
Exercise 2 (a)

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 32

Peppers, quantization factor 128

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

No, quantization is not a reversible process since we are setting some specific integer steps in quantization, some information is discarded.

We cannot recover from the outcome, because the position information is unknown

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.

The size of the files become smaller, since we discard some colour level in quantization step, the file is compressed comparing to the original one.

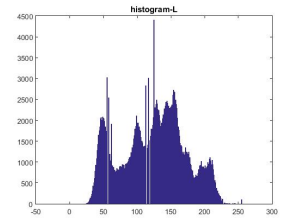
If we use the factor N, the new file size becomes 1/N.

Exercise 2 (b)

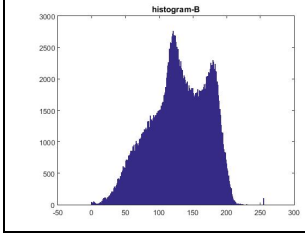
Histograms: This part of the lab is dedicated to image processing using histograms. A histogram is a statistical representation of the data within an image. The histogram can be represented as a plot of the frequency of occurrence of each grey level. This representation shows the distribution of the 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.

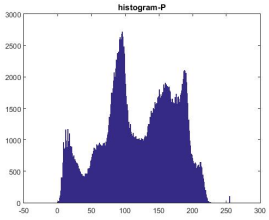
Lena



Baboon



Peppers



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

It presents the probability for each pixel in the image.

Exercise 2 (c)

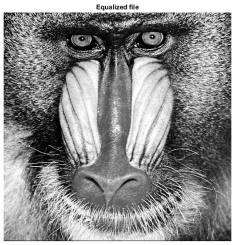
Equalize: Equalization is one of the possible image processing algorithms implemented using histograms. Histogram equalization allows a user to enhance the contrast of images. Histogram equalization employs a monotonic, non-linear mapping which re-assigns the intensity values of pixels in the input image 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.

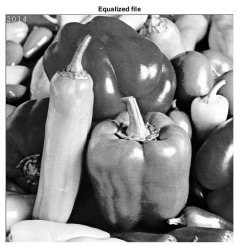
Display in the boxes the equalized images and their histograms.

Lena equalized

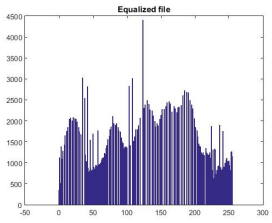
Baboon equalized



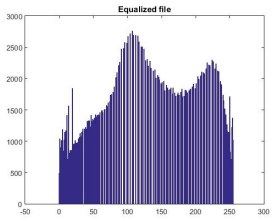
Peppers equalized



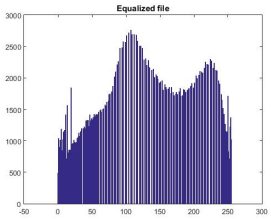
Lena histogram after equalization



Baboon histogram after equalization



Peppers histogram after equalization



Are the distributions really uniform? Explain your results.

It is not uniform. But from the range 0 to 255, the grey levels are more distributed and the probability are nearly uniform.

To achieve enhance the overall contrast of the image, we need to increase the dynamic range of the pixel grey values.

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.

The equalization step equalize the distance of each histogram line which makes the grey level colour distributed more smooth.

Exercise 2 (d)

Histogram modelling: Histogram modelling techniques are effective tools for modifying the dynamic range and contrast of an image. Unlike contrast stretching, histogram modelling operators may employ non-linear and non-monotonic transfer functions to map between pixel intensity values 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

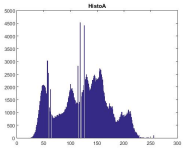
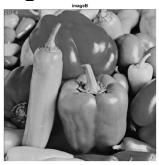
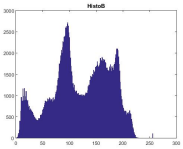


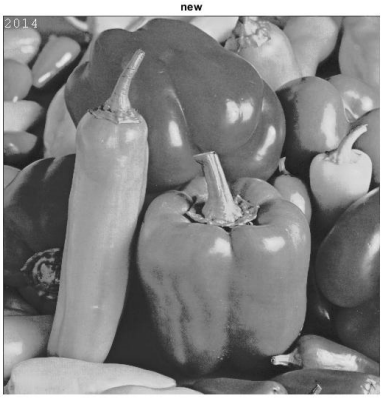
Image B



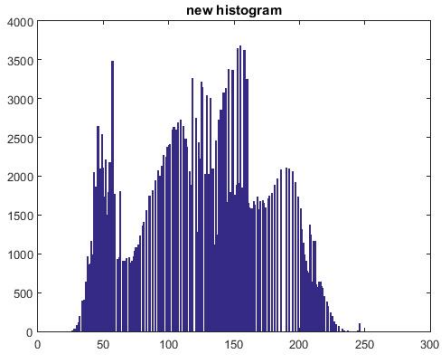
Histo of B



B after Modelling



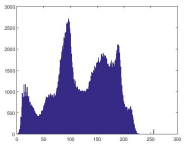
Histo of B after modelling



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

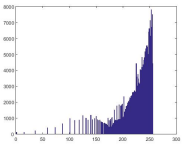
Image Peppers

Histo of Peppers



Peppers after modelling

Histo after modelling



Write in the box the formulation of your algorithm.

```
for i=-255:0
   e(i+256)=1.5^(i/16);
end;                                BUPT_histogram_modelling;
```

Exercise 3 (a)

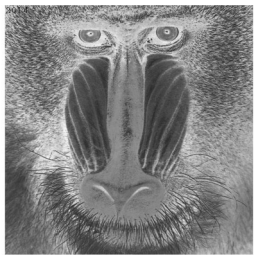
Negatives: We are used to the negative of an image in analogue image processing. It is possible to generate a negative from a digital image too. In the last part of today's lab you will solve a simple exercise on negative images.

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

Negative of Lina

Negative of Peppers

Negative of Baboon



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

Negative of Lina

Negative of Peppers

Negative of Baboon

Your comments:

We use ppmread and pgmread function to achieve the maximum grey levels in the image and

we used the max grey level value to subtract the original value in the previous pixel.

Two methods are applied in this method to identify the file types using function fileparts() to extract the extension and the strfind() function to compare two strings.

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 θ_1 and skews it with a second angle, θ_2.

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

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \quad R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Counter clockwise rotation around 3 axis

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

skewing X coordinates by angle a

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{\tan(a)} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

skewing Y coordinates by angle a

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{\tan(a)} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

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.

30

60

120

-50

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

10

40

60

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.

Yes, the problem always occurs in the image. We can solve this problem as follows.

When the image coordinate mapping is checked back, the current point is calculated outside or inside the original image. If it is outside, the X or Y position of the current pixel is judged, and the pixel values near the four boundaries are substituted.

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?

They are different from each other. Because when we change the transformation order, the position of the matrixes change as well which would definitely bring difference for the outcome.

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.

$$MSE = \frac{\sum [I(x, y) - I'(x, y)]^2}{NM}$$

$$PSNR = 20 \log_{10}\left(255/\sqrt{MSE}\right)$$

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

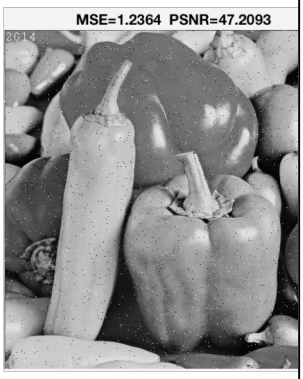
There is always some difference between the image and the one that after image compression. In order to measure the quality of the processed image, we usually use PSNR value to measure whether a process is satisfactory.

According to the previous equation, the root of the maximum of MSE is 255 that is to say, PSNR is always positive.

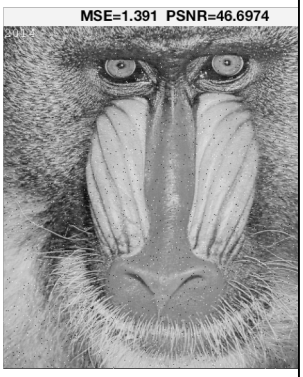
Create a function that can add (i) salt and pepper noise and (ii) Gaussian noise to a PGM image and compute PSNR and MSE. Show the results in the box and write under the box the values of MSE and PSNR comparing the original with the corrupted one.

Lena salt and pepper noise

Peppers salt and pepper noise

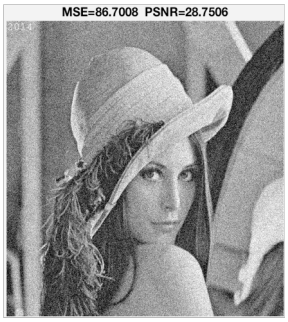


Baboon salt and pepper noise



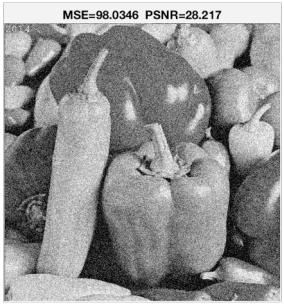
Lena Gaussian noise

σ =1% of the range.



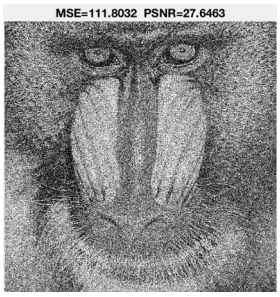
Peppers Gaussian noise

σ =2% of the range.



Baboon Gaussian noise

σ =7% of the range.

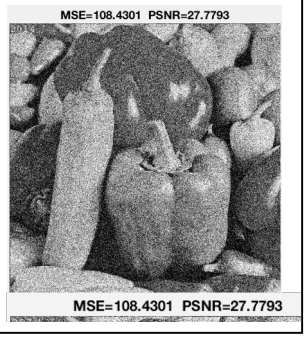


Lena Gaussian noise

σ =5% of the range.

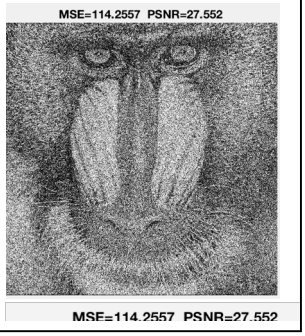
Peppers Gaussian noise

σ =5% of the range.



Baboon Gaussian noise

σ =10% of the range.

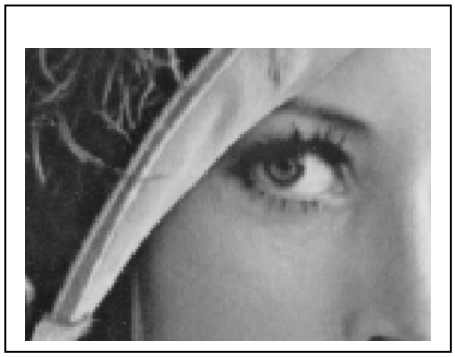


Exercise 4(b)

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.

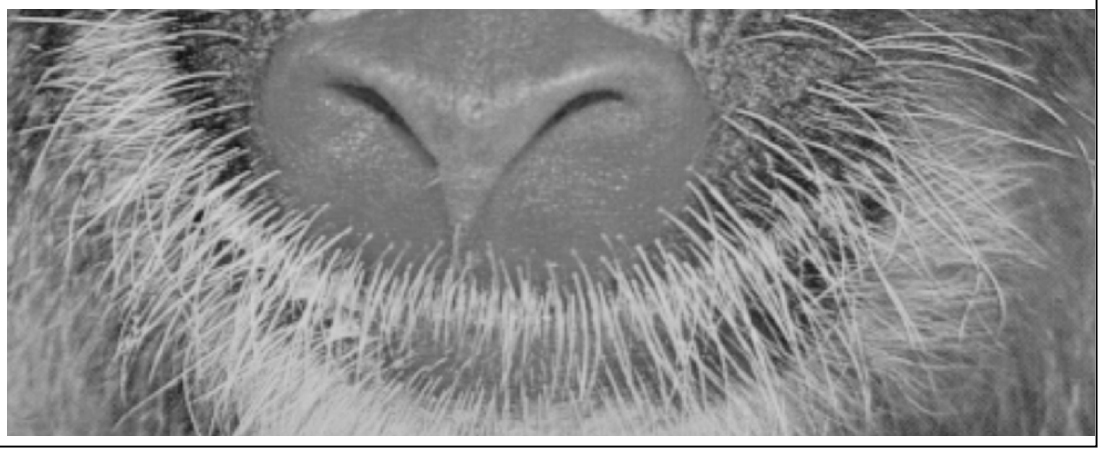


Your comments:

There are obvious mosaic sawtooth at the edge of the hat. That is because nearest-neighbour method because only uses one value to calculate.

It chooses the value of the nearest while don't consider the other neighbour points

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:

It is more smooth and show more details than the lena one. Bilinear interpolation uses the closet 4 neighbours instead only one which decrease the tooth blur.

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?

The nearest neighbour one is faster for it reference to less neighbours what the artefacts that use bilinear technique is more detailed.

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:

(i)

$$G_\sigma(k,l) = \frac{1}{2\sigma^2\pi} e^{-\frac{k^2+l^2}{2\sigma^2}}$$

(ii)

$$G_\sigma(k,l) = G_\sigma(k)G_\sigma(l) \qquad G_\sigma(k) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{k^2}{2\sigma^2}}$$

(iii)

The larger sigma means the more concentrated to the cut-off frequency.

The computational complexity can decrease by the window size of the image and the filter.

(iv)

Kernel size should fit for box size considering that it is used to calculate the pixels pass the Gauss filter.

Because the box has a side length decided by XY kernel size field and its height is decided by the value in Z kernel size field.

Add Gaussian noise to the image Lena with noise power 50, 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=0.5

σ=1

σ=2

σ=4

σ=7

σ=10

Your comments:

The image is more blurred when the sigma increases, that is because the noise in the filter affects the information.

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:

Rectangular-shaped filter can give a clearer output image than Gauss filter. Besides, it is obvious that the 5-by-5 kernel act better than 7-by-7 kernel.

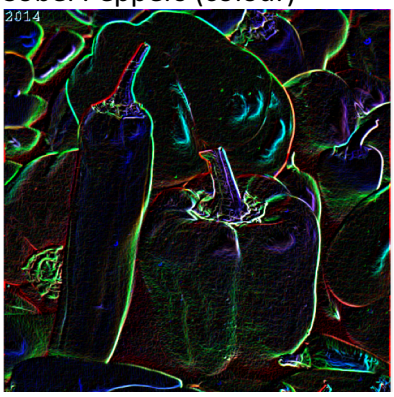
Exercise 5(b)

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.

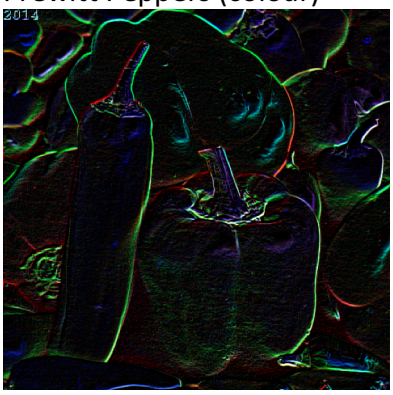
Sobel Lena (grey)

Sobel Peppers (colour)



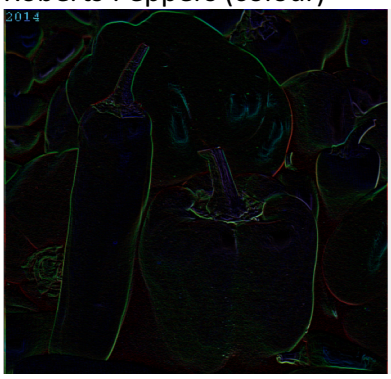
Prewitt Lena (grey)

Prewitt Peppers (colour)



Roberts Lena (grey)

Roberts Peppers (colour)



Your comments:

Robert operator positioning is more accurate, but not that smooth, so the noise is more sensitive. Prewitt operators and Sobel operators are all first order differential operators, while the former is the average filter, the latter is the weighted average filter and detection of image edges may be more than 2 pixels. Both of them have a better detection effect on the gray gradient low noise image, but the image processing effect is not ideal.

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.

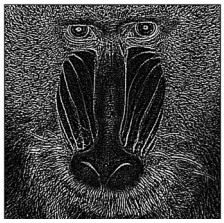
Implement the LoG operator as a parametric function of the variance, and display the results in the boxes.

Lena σ = 1

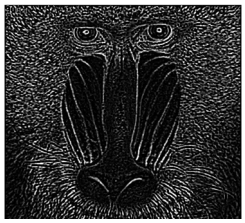
Peppers σ = 3

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 = 1



Baboon σ = 2



Baboon σ = 3

Add your comments here

The Laplacian filter does not provide information about edge direction, The magnitude of the gradient provides the information about the strengthen of the edge.

When we increase the sigma, the number of the edges decrease but they become a bit thicker since more information is ignored.

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 =3

Kernel size = 2*2

Peppers result

σ =3

Kernel size = 7*7

Discussion

When the value of sigma increases, then we should decrease the size of kernel to avoid truncating.

Big sigma and large kernel size own the same effect of smoothing the images, while the drawback is that they blur the image.

So we need to use small kernel size to compensate the effect brought by large sigma.