**EBU6307 – Visual Computing – 2024/25**

Coursework report and exercises

Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

QM student number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

BUPT student number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

DISCLAIMER

* The students are only allowed to use the provided Anaconda environment.
* All codes will be tested in the provided Anaconda environment.
* All codes should be executable in the terminal. That is, python <EXERCISE\_NAME>.py should be able to produce all the results.
* Outputs of the implementation must be generated in the **\results** folder
* Zero mark will be given if your code is not executable in the provided Anaconda environment.
* Zero mark will be given if another student’s name and ID are written in your own dataset, and **the submitted coursework will be reported as a plagiarism case**.
* If the submitted file does not follow the structure below, **10 marks will be deducted**.
* Make your folder structure as below:

EBU6307\_FIRSTNAME\_FAMILYNAME\_QMSTUDNETNUMBER

├── inputs

├── results

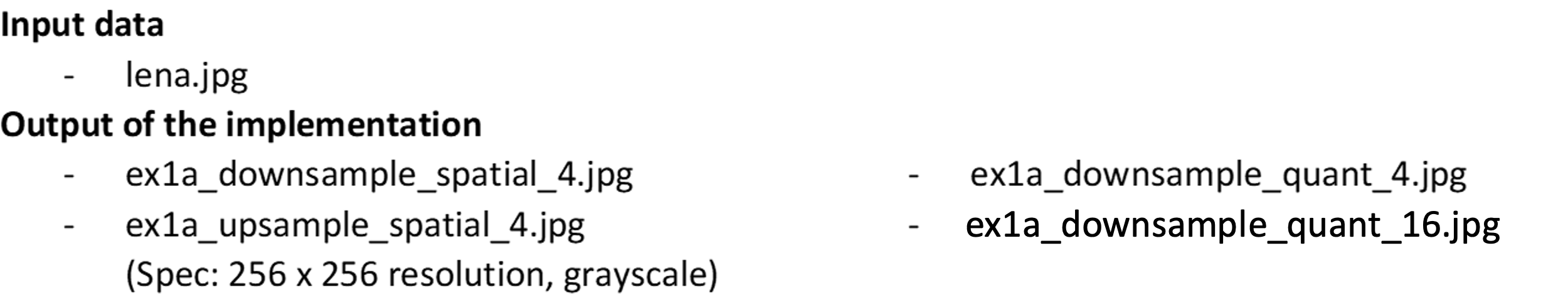
└── codes

* In the final coursework submission, submit 1) your report and the single 2) zip file of the above folder (EBU6307\_FIRSTNAME\_FAMILYNAME\_QMSTUDNETNUMBER) to the QMplus.
* QMplus submission example:
  + EBU6307\_SALMAN\_HALEEM\_1911XXXXX.pdf
  + EBU6307\_SALMAN\_HALEEM\_1911XXXXX.zip

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **DO NOT WRITE IN THIS TABLE (TA-only)** | | | | | |
|  | **1** | **2** | **3** | **4** | **Total** |
| **(a)** | **/10** | **/10** | **/10** | **/10** |  |
| **(b)** | **/10** | **/10** | **/10** | **/10** |
| **(c)** |  | **/10** | **/10** | **/10** |
| **(d)** | **-** | **/10** |  |  |
|  |  | **/10** |  |  |  |
|  |  | **/10** |  |  |  |
| **Total** | **/20** | **/60** | **/30** | **/30** | **/140** |

**Exercise 1: Data acquisition, filtering and getting familiar with Python/Scipy/Numpy/OpenCV  
Exercise 1 (a): Image resolution (Spatial and Intensity)**

**--------------------------------------------------------------------------------------------------------------------------------------**



**--------------------------------------------------------------------------------------------------------------------------------------**

**Description**

Implement spatial scaling and intensity scaling functions using a grayscale lena.jpg as an input. You need to get the original dimensions of the image.

**Spatial Scaling:**

1. Define the spatial scaling function without using OpenCV.
2. Downsample the original image by factor of 4 i.e. if original dimension is 512 x 512, the downsampled dimension should be 128 x 128
3. Take the downsampled image from ii as input and upsample back to original dimension as i.

**Intensity Scaling:**

1. Define the intensity scaling function without using OpenCV.
2. Downsample the original image upto 4 intensity levels
3. Downsample the original image upto 16 intensity levels

**Note.** You are not allowed to use OpenCV library except image I/O.

**Display the results in the boxes below:**

|  |  |  |
| --- | --- | --- |
| Original image | Spatial scaling (ii) | Spatial scaling (iii) |
|  | 女人戴着帽子  描述已自动生成 |  |
| Original image | Intensity scaling (ii) | Intensity scaling (iii) |
|  |  |  |

**Discuss the results (in 10 lines):**

Your discussion here (Font: Calibri, 11 pt)

**Exercise 1 (b): Image translation, rotation and shearing**



**Description**

Implement image translation, rotation and shearing. First you need to create a synthetic image i.e. red outlined rectangle using pillow (PIL) package and importing Image and ImageDraw functions which can take width and height of user defined choice. You can then put width=600 pixels and height=400 pixels, Outline width=5, and background colour as white. Red colour rgb code=(255,0,0) and white colour rgb code is (255,255,255).

1. Implement the image translation function and translate by tx=50, ty=30
2. Implement the image rotation function and rotate by 30 degrees
3. Implement the image shearing function and shear the image by a) and b)

**Note.** You are not allowed to use OpenCV library except image I/O

**Display the results in the boxes below:**

|  |  |  |
| --- | --- | --- |
| Original image | Image Translation (i) | Image Rotation (ii) |
| 形状, 矩形  描述已自动生成 | 形状, 矩形  描述已自动生成 | 形状, 多边形  描述已自动生成 |
|  | Image Shearing (iii) | Image Shearing (iii) |
| 形状, 矩形  描述已自动生成 | 形状  描述已自动生成 | 多边形  中度可信度描述已自动生成 |

**Discuss the results (in 10 lines):**

Your discussion here (Font: Calibri, 11 pt)

**Exercise 2 (a): Convolution gaussian filtering**

**--------------------------------------------------------------------------------------------------------------------------------------**

**Input data**

* lena.jpg

**Output of the implementation**

* ex2a\_gf\_5\_1.jpg
* ex2a\_gf\_5\_10.jpg
* ex2a\_gf\_21\_1.jpg
* ex2a\_gf\_21\_10.jpg

**--------------------------------------------------------------------------------------------------------------------------------------**

**Description**

Implement gaussian filtering based on convolution function using a grayscale lena.jpg (you can convert coloured lena into grayscale) as an input. See the results with varying the window size (kernel size), , and the standard deviation, .

**Note.** You are not allowed to use OpenCV library except image I/O.

**Display the results in the boxes below:**

|  |  |  |
| --- | --- | --- |
| Original image |  |  |
|  |  | 女人戴着帽子  描述已自动生成 |
| Original image |  |  |
|  | 女人戴着帽子  描述已自动生成 | 女人的照片  低可信度描述已自动生成 |

**Discuss the results (in 10 lines):**

Your discussion here (Font: Calibri, 11 pt)

**Exercise 2: Low Vision, Image filtering, Image features**

**Exercise 2 (b): Image Aliasing**

**--------------------------------------------------------------------------------------------------------------------------------------**

**Input data**

* Squares.jpg

**Output of the implementation**

* ex2b\_Alias.jpg
* ex2b\_Sharp.jpg

**--------------------------------------------------------------------------------------------------------------------------------------**

**Description**

1. Using a Squares.jpg as an input and implement anti-aliasing filter
2. Using a Squares.jpg as an input and implement image sharpening filter on anti-aliased image

**Note.** You are not allowed to use OpenCV library except image I/O.

**Display the results in the boxes below:**

|  |  |
| --- | --- |
| Original image | Anti-alias |
| 图片包含 游戏机, 物体, 建筑, 地板  描述已自动生成 |  |
| Original image | Image sharpening |
|  |  |

**Discuss the results (in 10 lines):**

Your discussion here (Font: Calibri, 11 pt)

**Exercise 2 (c): SIFT (Scale Invariant Feature Transform) and feature matching**

**--------------------------------------------------------------------------------------------------------------------------------------**

**Input data**

* sift\_input1.jpg
* sift\_input2.jpg

**Output of the implementation**

* ex2c\_sift\_input1.jpg
* ex2c\_sift\_input2.jpg
* ex2c\_matches\_least10.jpg
* ex2c\_matches\_most10.jpg

**--------------------------------------------------------------------------------------------------------------------------------------**

**Description**

Run the SIFT descriptor and nearest neighbour (NN) feature matching using the function provided by OpenCV.

**Task 1.**

Run SIFT descriptor for two images (sift\_input1.jpg) and (sift\_input2.jpg).  
Draw the keypoints on (ex2c\_sift\_input1.jpg) and (ex2d\_sift\_input2.jpg).  
**Task 2.**Perform the feature matching using NN for →.   
Show the worst 10 matches (ex2c\_matches\_least10.jpg) and best (nearest) 10 matches (ex2c\_matches\_most10.jpg) for →.

Display the results:

|  |  |
| --- | --- |
| ex2c\_sift\_input1.jpg | ex2c\_sift\_input2.jpg |
|  |  |
| ex2c\_matches\_least10.jpg | ex2c\_matches\_most10.jpg |
|  |  |

**Discuss the results (in 10 lines):**

Your discussion here (Font: Calibri, 11 pt)

**Exercise 2 (d): Canny Edge Detection**

**--------------------------------------------------------------------------------------------------------------------------------------**

**Input data**

* building.jpg

**Output of the implementation**

* ex2d\_canny1.jpg
* ex2d\_canny2.jpg
* ex2d\_canny3.jpg
* ex2d\_canny4.jpg
* ex2d\_canny5.jpg
* ex2d\_ canny6.jpg

**--------------------------------------------------------------------------------------------------------------------------------------**

**Description**

Convert building.jpg into grayscale and perform

1. Perform Gaussian Smoothing using sigma=1.4
2. Perform Sobel operator and calculate gradient magnitude(you are not allowed to use OpenCV library or scipy function)
3. Perform the non-maxima suppression (you are not allowed to use OpenCV library or scipy function)
4. Apply double threshold to detect strong, weak and non edges (use the range 0.5<weak edge<1 and strong edge>1) (you are not allowed to use OpenCV library or scipy function)
5. Perform edge tracking by converting weak edge to strong edge if connected to strong edge. (you are not allowed to use OpenCV library or scipy function)
6. Determine edge map using OpenCV canny edge detection function and compare your results

**Display the results in the boxes below:**

|  |  |  |  |
| --- | --- | --- | --- |
| Original image | i | ii | iii |
| 建筑的摆设布局  描述已自动生成 | 建筑的摆设布局  描述已自动生成 | 建筑与房屋的城市空拍图黑白照  中度可信度描述已自动生成 | 建筑的设计  低可信度描述已自动生成 |
| Original image | iv | v | vi |
| 建筑的摆设布局  描述已自动生成 | 图片包含 户外, 田地, 建筑, 站  描述已自动生成 | 夜晚的月亮  中度可信度描述已自动生成 | 黑板上的字  描述已自动生成 |

**Discuss the results (in 10 lines):**

Your discussion here (Font: Calibri, 11 pt)

**Exercise 2 (e): fourier transform for high pass and low pass filtering**

**--------------------------------------------------------------------------------------------------------------------------------------**

**Input data**

* cat.png

**Output of the implementation**

* ex2e\_fourier.jpg
* ex2e\_restored.jpg
* ex2e\_low pass\_30.jpg
* ex2e\_highpass\_30.jpg

**--------------------------------------------------------------------------------------------------------------------------------------**

**Description**

1. Implement the apply\_fourier\_transform function using np.fft.fft2, save into fft\_image variable, then shift the zero frequency component to the centre using np.fft.fftshift and save to fft\_shift. Return both variables. Display fft\_shift
2. Implement inverse fourier using np.fft.ifftshift on fft\_shift, save it in ifft\_shift and then restor image using np.,fft.ifft2(ifft\_shift).real
3. Implement the high\_pass\_filter with cutoff frequency of 30. This can be done by zero out frequencies from centre to half of fft\_shift. Restore image using inverse fourier function in ii.
4. Implement the low\_pass\_filter with cutoff frequency of 30. This can be done by zero out frequencies from half to full length of fft\_shift. Restore image using inverse fourier function in ii.

**Display the results in the boxes below:**

|  |  |  |
| --- | --- | --- |
| Original image | Fourier transform after zero shift | Restored image after high pass filter |
| 灰色的小猫  描述已自动生成 | 图片包含 街道, 游戏机, 桶, 淋浴  描述已自动生成 | 黑暗的猫  描述已自动生成 |
| Original image | Restored image | Restored image after low pass filter |
| 灰色的小猫  描述已自动生成 | 灰色的小猫  描述已自动生成 | 猫躺在人的眼睛  描述已自动生成 |

**Discuss the results (in 10 lines):**

Your discussion here (Font: Calibri, 11 pt)

**Exercise 2 (f): Hough Tranform**

**--------------------------------------------------------------------------------------------------------------------------------------**

**Input data**

* HoughTransformLines.jpg

**Output of the implementation**

* ex2f\_edgemap.jpg
* ex2f\_houghtransform1.jpg
* ex2f\_detectedline1.jpg
* ex2f\_houghtransform2.jpg
* ex2f\_ detectedline2.jpg

**--------------------------------------------------------------------------------------------------------------------------------------**

**Description**

1. Create the edge map via Sobel operators (you are not allowed to use OpenCV library or scipy function)
2. Create the custom defined Hough Transform method to show the Hough Transform space i.e. ( (you are not allowed to use OpenCV library or scipy function)
3. Determine the detected lines using the threshold=0.5 (you are not allowed to use OpenCV library or scipy function)
4. Repeat ii and iii using OpenCV function

**Display the results in the boxes below:**

|  |  |  |  |
| --- | --- | --- | --- |
| Original image | Edge Map | Hough Transform Space ii | Detected Lines iii |
| 图片包含 游戏机, 黑暗, 水, 风筝  描述已自动生成 | 图片包含 游戏机  描述已自动生成 | 黑暗中的光  描述已自动生成 | 黑暗里有星球  中度可信度描述已自动生成 |
|  |  | Hough Transform Space iv | Detected Lines iv |
|  |  | 在黑暗中  描述已自动生成 | 图片包含 水, 线, 黑暗, 滑雪  描述已自动生成 |

**Discuss the results (in 10 lines):**

Your discussion here (Font: Calibri, 11 pt)

**Exercise 3 (a): Stereo matching – window-based matching**

**--------------------------------------------------------------------------------------------------------------------------------------**

**Input data**

* teddy\_im2.png, teddy\_im6.png

**Output of the implementation**

* ex3a\_w\_3.png
* ex3a\_w\_11.png

**--------------------------------------------------------------------------------------------------------------------------------------**

**Description**

Window-based stereo matching is a local stereo matching method that estimate the disparity of pixel in the left image, where the left and right images**,** and are rectified. Window-based stereo matching can be performed as three steps:

1. Matching cost computation:

2. Cost aggregation:

3. Disparity computation:

where is a pixel coordinate and is a window where its centre is.

Perform the window-based stereo matching with varying the window size of as follows:

* window size:
* window size:

**Note.** You are not allowed to use OpenCV library except image I/O.

Display the results:

|  |  |
| --- | --- |
| ex3a\_w\_3.png | ex3a\_w\_11.png |
| 图片包含 物体, 照片, 桌子, 蛋糕  描述已自动生成 | 图片包含 桌子, 照片, 蛋糕, 食物  描述已自动生成 |

**Discuss the results (in 10 lines):**

Your discussion here (Font: Calibri, 11 pt)

**Exercise 3 (b): Stereo matching – Adaptive support-weight**

**--------------------------------------------------------------------------------------------------------------------------------------**

**Input data**

* teddy\_im2.png, teddy\_im6.png

**Output of the implementation**

* ex3b\_aw\_3.png
* ex3b\_aw\_11.png

**--------------------------------------------------------------------------------------------------------------------------------------**

**Description**

Adaptive support-weight based stereo matching is a local stereo matching method that estimate the disparity of pixel in the left image, where the left and right images**,** and are rectified. Window-based stereo matching can be performed as three steps:

1. Matching cost computation:

2. Cost aggregation:

3. Disparity computation:

where is a pixel coordinate and is a window where its centre is. Unlike window-based stereo matching, Adaptive support-weight based stereo matching employs the weight that measures the intensity and spatial similarity between pixelandin the left image.

Perform the Adaptive support-weight based stereo matching with varying the window size of as follows (hint: you can modify the bilateral filter in Exercise 2 (b) to measure the weight) :

* window size:
* window size:

**Note.** You are not allowed to use OpenCV library except image I/O.

Display the results:

|  |  |
| --- | --- |
| ex3b\_aw\_3.png | ex3b\_aw\_11.png |
| 图片包含 蛋糕, 物体, 照片, 桌子  描述已自动生成 | 图片包含 蛋糕, 桌子, 照片, 片  描述已自动生成 |

**Discuss the results (in 10 lines):**

Your discussion here (Font: Calibri, 11 pt)

**Exercise 3 (c): Mean-shift Tracking**

**--------------------------------------------------------------------------------------------------------------------------------------**

**Input data**

* ebu6304\_chaplin.mp4  
  (Spec: 720 × 960 (height × width) resolution, 40sec, 25 fps)

**Output of the implementation**

* ex3c\_meanshift\_track\_chaplinface.mp4   
  (Spec: 720 × 960 (height × width) resolution, 40sec, 25 fps)

**--------------------------------------------------------------------------------------------------------------------------------------**

**Description**

Using OpenCV, perform the mean-shift tracking with initial bounding box. (Don’t need to utilize entire video, just 4 seconds from the beginning would be sufficient)

1. Initial bounding box that locates the chaplin face   
   (Output: ex3c\_meanshift\_track\_chaplinface.mp4)

**Display the results:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ex3c\_meanshift\_track\_chaplinface.mp4 | | | | |
| 1st frame | 20th frame | 40th frame | 60th frame | 90th frame |
| 商店外的西装笔挺的男子黑白照  中度可信度描述已自动生成 | 穿着西装笔挺的男子黑白照  中度可信度描述已自动生成 |  | 图片包含 人, 建筑, 室内, 女人  描述已自动生成 | 图片包含 人, 建筑, 室内, 天花板  描述已自动生成 |

**Discuss the results (in 10 lines):**

Your discussion here (Font: Calibri, 11 pt)

**Exercise 4: Deep Learning Project**

**--------------------------------------------------------------------------------------------------------------------------------------**

**Description**

The lab is concerned with the design and development of the deep learning based model for depth estimation in stereo images. We have extracted the Middlebury dataset (2021 Mobile stereo datasets with ground truth <https://vision.middlebury.edu/stereo/data/scenes2021/>). The model should be capable of take a pair of stereo images as an input and estimate the depth based on trained deep learning network. The expected tasks are mentioned as follows:

1. Design and development of deep learning driven pipeline which takes multiple pairs of stereo images as an input with disparity maps as their labels. You are highly encouraged to use standard architectures (e.g. AlexNet, GoogleNet, ResNet etc.)
2. Implementation of a model which takes a pair of stereo images as an input and calculate the disparity map based on trained model
3. Model evaluation using 5-fold cross validation based on disparity error between original disparity map and predicted disparity map.

In this lab, you are encouraged to use GenAI to generate your code. However, it would totally your idea for choice of deep learning model and how to integrate different Input/Output blocks.

**--------------------------------------------------------------------------------------------------------------------------------------**

**Input data**

* 2021 Mobile stereo datasets with ground truth

**Output of the implementation**

* Generate architecture of deep learning pipeline from python functions training the stereo images (ex4a\_architecture.png)
* Generate 3 examples of predicted disparity map as an output for estimated depth. Also mention which test images have been used to create those disparity maps (ex4b\_name of test image\_disparitymap.png)
* Generate table representing cross validation results in terms of disparity error in each fold of test images (ex4c\_crossvalidation.csv)

**--------------------------------------------------------------------------------------------------------------------------------------**

**Dataset description**

Each dataset consists of 2 views taken under several different illuminations and exposures. The folder contains 24 scenes (image pair, disparities, calibration file). The files are organized as follows:

SCENE{1,2,3}/ -- scene imaged from 1-3 viewing directions

im0e{0,1,2,...}.png -- left view under different exposures

im1e{0,1,2,...}.png -- right view under different exposures

calib.txt -- calibration information

im{0,1}.png -- default left and right view (typically ambient/L0/im{0,1}e2.png)

disp{0,1}.pfm -- left and right GT disparities

Calibration file format with explanation:

Here is a sample calib.txt file:

1. cam0=[1758.23 0 953.34; 0 1758.23 552.29; 0 0 1]

cam1=[1758.23 0 953.34; 0 1758.23 552.29; 0 0 1]

cam0,1: camera matrices for the rectified views, in the form [f 0 cx; 0 f cy; 0 0 1], where f: focal length in pixels. cx, cy: principal point

1. doffs=0 (x-difference of principal points, doffs = cx1 - cx0 (here always == 0))
2. baseline=111.53 (camera baseline in mm)
3. width=1920

height=1080

width, height: image size

1. ndisp=290 (a conservative bound on the number of disparity levels; the stereo algorithm MAY utilize this bound and search from d = 0 .. ndisp-1)
2. isint=0
3. vmin=75

vmax=262

vmin, vmax: a tight bound on minimum and maximum disparities, used for color visualization, the stereo algorithm MAY NOT utilize this information

To convert from the floating-point disparity value d [pixels] in the .pfm file to depth Z [mm] the following equation can be used: Z = baseline \* f / (d + doffs)

**--------------------------------------------------------------------------------------------------------------------------------------**

**Output of the implementation**