Computer Vision Assignment 1 (CLab1) S1-2024

0 Overview

This assignment has **five** tasks in total, is graded out of **25** marks, and is worth **15%** of your final mark for this course.

0.1 Objectives

The goal of this assignment is to develop and assess proficiency at low-level image processing in Python, such as basic image I/O, image manipulation, and image filtering.

0.2 Permitted Python libraries

The Python libraries that you may use in this assignment are

- OpenCV (cv2);
- NumPy;
- Matplotlib;
- scikit-image (skimage); and
- Pillow (PIL).

Use of other Python libraries will result in a score of 0 for the task in which the library was used.

0.3 Advice

1. Before writing your report, we recommend you watch the video "how to write a good lab report" on Wattle. The markers do not want to just see a collection of experimental results, but rather a coherent explanation and interpretation of the results.
2. The requirements for submission are at the end of this document. Please ensure your submission meets the requirements.
3. The report is to be uploaded to the Wattle site before the due time. This course does not allow late submissions. That is, any submission after the deadline will receive a mark of 0.
4. This is an individual assignment. All students must work individually when coding and writing the report.

0.4 Academic Integrity

You are expected to comply with the university policy on academic integrity and plagiarism. Please ensure you cite appropriately any resources that you use (lecture notes, papers, online documents, code), and complete all tasks independently. All academic integrity violations will be reported and can result in significant penalties. More importantly, working through the problems yourself will help you learn the material and will set you up for success in the final exam.

(a) image1.jpg

(b) image2.jpg

(c) image3.jpg

Figure 1: Images for Task 1.

1 Task 1: Basic Image I/O (2 marks)

Given the three images shown in Figure 1, which can be downloaded from Wattle:

1. Save them to JPG image files of size 1024 columns $\times$ 720 rows, named "image1.jpg", "image2.jpg" and "image3.jpg". (0 marks)
2. Using image1.jpg, develop code and functions that do the following tasks:
   (a) Read this image from its JPG file and resize it to 384 columns $\times$ 256 rows. (0.2 marks)
   (b) Separate the colour image into red, green, and blue (RGB) channels, and display each separately as a grayscale image. (0.2 marks per channel, 0.6 marks in total)
   (c) Compute and display a histogram for each of the grayscale images. (0.2 marks per histogram, 0.6 marks in total)
   (d) Apply histogram equalisation to the individual red, green, and blue channels and then merge the three channels into a colour image. Display the three new histograms after equalisation and the new colour image. (0.15 marks per histogram and image, 0.6 marks in total)

Note: You are **NOT** allowed to use any built-in function for histogram equalisation in this task, you must write your own function. However, you may use built-in functions to check your answer.

(a) (b)

Figure 2: Images for Task 2.

2 Task 2: Colour Space Conversion (3 marks)

Use the two images in Figure 2 to study colour space conversion from RGB to HSV. You can download them from Wattle.

1. Based on the formulation of RGB-to-HSV conversion:
   (a) Write your own function `cvRGB2HSV` that converts the RGB image to HSV colour space. (1 mark)
   (b) Read in Figure 2(a) and convert it with your function, and then display the H, S, and V channels in your report. (0.5 marks)
2. Compute the average hue (H) values of the five colour regions in Figure 2(b) with your function and Python's built-in (scikit-image) function `rgb2hsv`.
   (a) Print both results under the corresponding regions. (0.5 marks)
   (b) Explain how to distinguish and divide the five regions, and how to calculate the average hue value. More efficient solutions will obtain higher marks. (1 mark)
   Note: The elements of both colourmaps are in the range 0 to 1.

3 Task 3: Image Denoising via a Gaussian Filter (5 marks)

In this task, you are asked to:

1. Read in image4.jpg. Crop a square image region corresponding to the central facial part of the image, resize it to 512×512, and save this square region to a new grayscale image. Display the two images. (0 marks)
2. Add Gaussian noise to the new 512×512 image, with zero mean and a standard deviation of 15. The intensity values of the noised image should be clipped to be within [0, 255]. (0.5 marks)
   Hint: Make sure your input image range is within [0, 255]. You may need the `np.random.randn` function in Python; take a look at its default settings before use.
3. Compute and display two intensity histograms side by side, one before adding the noise and one after adding the noise. Describe what you can observe from the histograms and explain the reasons why they are similar/different. (1 mark)
4. Implement your own function that performs a 7×7 Gaussian filtering operation. (1.5 marks)
   Your function interface is:
   ```
   my_Gauss_filter()
       input:  noisy_image, my_7x7_gauss_kernel
       output:  output_image
   ```
5. Apply your Gaussian filter to the noisy image from step 2 and display the smoothed images and visually verify their noise-removal effects. (0.5 marks)
6. One of the key parameters to choose for the task of image filtering is the standard deviation of your Gaussian filter. Test and compare several different Gaussian kernels with different standard deviations. (1 mark)
   Note: In doing this task you **MUST NOT** use any built-in image filtering functions (e.g., `cv2.filter2D`). In other words, you are required to code your own 2D filtering code, based on the original mathematical definition for 2D convolution. However, you *are* allowed to generate the 7×7 Gaussian *kernel* using built-in functions.
7. Compare your result with a built-in 7×7 Gaussian filter function (e.g., `cv2.GaussianBlur`) and show that the two results are nearly identical. (0.5 marks)

Further reading material: `https://setosa.io/ev/image-kernels/`.

4 Task 4: Sobel Edge Filter (5 marks)

1. Implement your own 3×3 Sobel filter. Do not use any built-in edge detection filter. The implementation of the filter should be clearly presented with your code. Test your filter on sample images (image2.jpg, image4.jpg), first converting them to grayscale, and visualise the results. (1.5 marks)
   (a) Compare your result with the built-in Sobel edge detection function. (0.5 marks)
   (b) Briefly explain the purpose of the Sobel filter and how it achieves this. (1 mark)
2. Investigate the accuracy of the Sobel filter for predicting the orientation of edges. If it is used for edge detection, what might cause inaccuracies in the estimation of edge orientations? Look at the histogram of the gradient orientation and find out the major edge orientations in the image. Discuss whether the gradient orientation histogram aligns with the real expected edges for the image or not. (2 marks)
   Note: You can use the built-in Sobel function for this discussion.

5 Task 5: Short Response Questions (5 marks)

1. Table 1 is a separable filter.
   (a) What does it mean to be a separable filter? (0.5 marks)
   (b) Write down the separate components of the filter in Table 1. (1 mark)
2. In this assignment, we have so far looked at applying convolution kernels to single-channel (grayscale) images.
   (a) Propose an approach for extending convolutions to multi-channel inputs with multi-channel outputs, where the number of input and output channels may differ. (1.5 marks)
   (b) Discuss why these more generalised convolution operators may be useful in computer vision. (1 mark)
   (c) Design and write down a 3×3 kernel that discards the green channel and switches the red and blue channels before averaging over the kernel window. (1 mark)
   Note: The single-channel 3×3 averaging filter is given in Table 2 for reference. If the designed filter has repeated elements, it is acceptable to write down only the unique part and indicate how it is to be repeated.

4	4	6
4	4	6
6	6	9

Table 1: 3×3 separable filter.

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Table 2: Single-channel 3×3 averaging filter.

6 Submission Quality (5 marks)

As well as its correctness (the "what"), your report is also judged on its presentation quality (the "how"). Marks are allocated for:

- Clear and concise interpretation and presentation (2 marks)
- Correct report formatting (2 marks)
- Code quality, including clarity and commenting (1 mark)

7 Submission Requirements

7.1 Files

Upload a single ZIP file by the due date. You must use the following file name: A1-uXXXXXXX.zip, replacing uXXXXXXX with your university ID. Your ZIP file must contain:

1. A PDF of your report.
2. A folder named "code" that includes all your *.py files.

7.2 Report

At the top of the first page, include the assignment title, your name and your university ID. The report may be written in LaTeX or other word processing software, but must be exported to PDF. Handwritten submissions are not permitted. Brief explanations on how you solve the problems are expected.

7.2.1 Formatting Instructions.

Use:

1. numbered headings, as given in this document;
2. numbered answers, corresponding to those in this document;
3. Times New Roman font;
4. single-spaced font;
5. 12pt font for the body text;
6. references at the end, where needed; and
7. images where appropriate to explain your answers.

If not responding to a particular question or task, still include the associated heading and question number, but leave the content blank. For example, your report might look like:

1 Task 1: Basic Image I/O

1.1 Question 1

< Blank >

1.2 Question 2

Documentation, observations, results, analysis, etc.

. . . etc.

7.2.2 Figures and Tables.

These are critical for communicating your results clearly. Always refer to these from the body text of the report (e.g., "As shown in Figure 2, the results indicate that. . ."). Use:

1. descriptive captions;
2. axis labels;
3. a legend;
4. appropriate axis scaling (e.g., perhaps a log scale is more informative?);
5. appropriate annotations;
6. 10pt Times New Roman font;
7. a sufficient size for visibility; and
8. your own material, or public-domain and cited images only.