COMP9517: Computer Vision

2025 T2 Lab 1 Specification

Maximum Marks Achievable: 2.5

This lab is **worth 2.5% of the total course mark**.

The lab must be submitted online.
Instructions for submission will be posted closer to the deadline.
Deadline for submission is Week 3, Friday 20 June 2025, 18:00:00 AET.

Objective: This lab revisits important concepts covered in the Week 1 and Week 2 lectures and aims to make you familiar with implementing specific algorithms.

Software: You are required to use OpenCV 3+ with Python 3+ and submit your code as a Jupyter notebook (see coding and submission requirements below). In the tutor consultation session this week, your tutors will give a demo of the software to be used, and you can ask any questions you may have about this lab.

Materials: The images to be used in this lab are available via WebCMS3.

Submission: All code and requested results are assessable after the lab. Submit your source code as a Jupyter notebook (.ipynb file) that includes all output and answers to all questions (see coding requirements at the end of this document) by the above deadline. The submission link will be announced in due time.

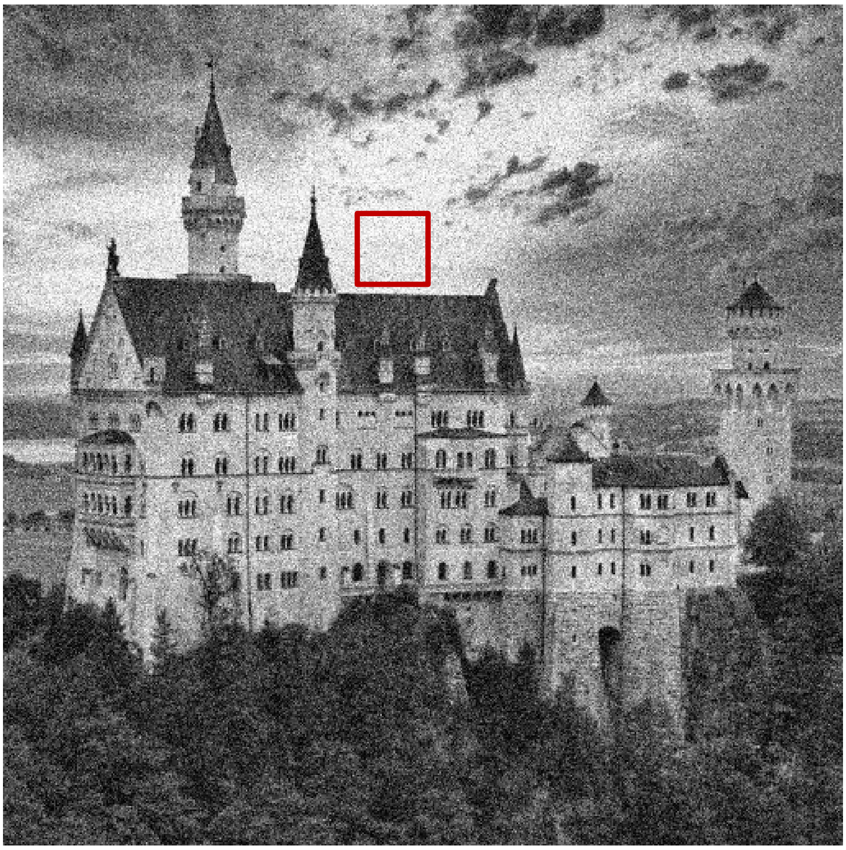
Task 1 (0.75 mark)

While on holiday in Europe, you visit an old castle and decide to take a nice picture, only to discover that your camera for some reason produces very noisy images. You remember from the computer vision course that you can still create a nice picture simply by averaging many images, so you take 10 images from the same viewpoint (see the given .zip file).

In your notebook, calculate the average image of the 10 noisy images and display the result. Also answer the following questions in your notebook:

Theoretically, how much noise reduction should you be able to achieve this way? That is, by what factor should the standard deviation of the noise drop?

Practically, how much noise reduction did you actually achieve here? Note that this requires measuring the standard deviation of the pixel values within a region containing only random noise and no other image structures. Perhaps the best region is the brightest part of the sky as indicated by the red box in the example below. Measure the standard deviation within that region in a single noisy image and within the same region in the averaged image. Report the two standard deviations and the ratio of the former to the latter.



Task 2 (0.75 mark)

Given the average image, you decide to create derived versions of it, using different image filters. One interesting filter is the difference of Gaussians (DoG).

Let us define two approximations of 2D Gaussian filters:

$$h_1 = \frac{1}{16} \cdot \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 2 & 4 & 2 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad h_2 = \frac{1}{256} \cdot \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

This notation means that, to get the actual kernel values, each value in the matrix must be multiplied by the factor in front of it. This is because the kernels must be normalized (the kernel values must sum to 1 to avoid blowing up the pixel values in the filtered image).

As you can see, h_1 is a 'narrower' filter (the values in the outer rows and columns are all 0, so they do not actually count, and the filter is effectively 3 x 3 pixels) and h_2 a 'wider' filter (5 x 5 pixels). These two filters can be combined to create a DoG filter:

$$\text{DoG} = h_1 - h_2$$

The convolution of an image f with the DoG filter can be written as:

$$g = f * \text{DoG} = f * (h_1 - h_2)$$

In other words, you first calculate a new kernel, being the difference $(h_1 - h_2)$, and then you convolve the image f with it to get the result image g.

From the image processing lecture you remember that convolution is a distributive operation. This means the convolution could equivalently be written as:

$$g = f * h_1 - \ f * h_2$$

In other words, you first convolve the image f with h_1, and separately with h_2, and then you subtract the two resulting filtered images to get g.

Take your average image from Task 1 and calculate its DoG-filtered version using the two different approaches. Keep in mind that either approach involves subtraction, and thus the resulting pixel values could be positive or negative, so make sure to use data types that can handle this. In your notebook, display the result images of the two different DoG-filtering approaches. Also answer the following questions in your notebook:

Which of the two approaches is computationally faster? And why?

Are the corresponding pixel values in the two result images exactly the same? In other words, if you subtract the two images, is the result 0 everywhere? If not, why not?

Task 3 (1 mark)
Another way of representing images is through their frequency spectrum. This can be obtained by applying the 2D Discrete Fourier Transform (DFT). Since in this lab the images are 512 x 512 pixels, meaning their dimensions are powers of 2, the fastest way to calculate the 2D DFT is to apply the 2D Fast Fourier Transform (FFT).

Take your average image from Task 1, and one of the DoG-filtered versions from Task 2 (it does not matter which one) and calculate the 2D FFT of each. More specifically, in both cases, calculate the magnitude of the 2D FFT. You may need to adjust the brightness/contrast of the magnitude images a lot to see their contents more clearly. In your notebook, display the magnitude of both images side by side and explain the key difference.

Coding Requirements

Make sure that in your Jupyter notebook, the input images are readable from a location specified as an argument, and all output images and other requested results are displayed in the notebook environment. All cells in your notebook should have been executed so that the tutor/marker does not have to execute the notebook again to see the results.

Released: 9 June 2025