COMP9517: Computer Vision

2025 T3 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 3 October 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 hiking in The Blue Mountains, you decide to take a nice picture, only to discover that your camera is defective and 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. The best region for this is the sky.

Measure the standard deviation within some rectangular region in the sky in one of the single noisy images 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:

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

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

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

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 correctly. 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)

From the lecture on image processing in the frequency (Fourier) domain, you know that DoG is a high-pass filter. This means that the output of the filter is an image containing only the higher frequencies contained in the input image.

From the lecture on image processing in the spatial domain, you know that high frequencies can be used to sharpen an image using a technique called unsharp masking.

Take one of the DoG-filtered images from Task 2 (it does not matter which one) and multiply

it with a factor of 5. Then add the resulting image to your average image from Task 1.

You may need to adjust the brightness/contrast of the final image a bit to make the intensities look similar to those of the average image from Task 1.

Visually comparing the output image of Task 3 with the output image of Task 1, you will notice that indeed edges look a bit sharper now (higher local contrast across the image). In your notebook, show the two images side by side, and answer the following questions:

In addition to edge sharpness, what else has been amplified in the output image of Task 3, which we were trying to get rid of in Task 1?

And what is the theoretical explanation for why this happens? Hint: the DoG filter is an approximation of the (inverted) Laplacean filter.

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: 22 September 2025