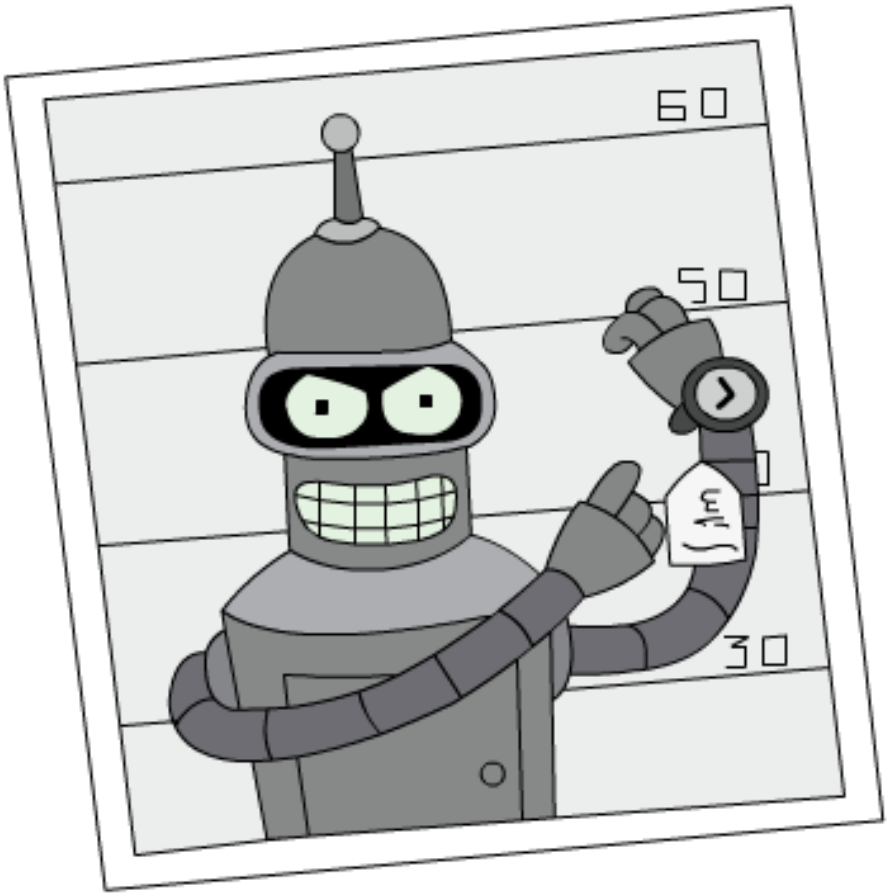
Convolutional filters and pointwise image operations

ECE4076/5176 Computer Vision
Lab 1 (Weeks 3,4)



Academic integrity Every lab submission will be screened for any collusion and/or plagiarism. Breaches of academic integrity will be investigated thoroughly and may result in a zero for the assessment along with interviews with the plagiarism officers at Monash University.

Late submission. Late submission of the lab will incur a penalty of 10% for each day late. That is with one day delay, the maximum mark you can get from the assignment is 7.2 out of 8, so if you score 7.5, we will (sadly) give you 7.2. Labs submitted with more than a week delay will not be assessed. Please apply for special consideration for late submission as soon as possible (*e.g.*, documented serious illness).

Lab specific restrictions/ allowances.

- You may **not** use any built-in opencv functions for this lab, other than those used for loading/ saving an image.
- You may use numpy for array handling and vectorising code (fewer for loops) is encouraged.
- You should use matplotlib to display the images and any intermediate results.
- You may use a generative AI tool or coding assistance[1], but must indicate the prompts you used, and explain in detail changes made to modify the code to complete the assignment.

Each lab is worth 8% and there are a number of sections and tasks with their own weighting. A task is only considered complete if you can demonstrate a working program and show an understanding of the underlying concepts. Note that later tasks should reuse code from earlier tasks.

This lab is about basic computer vision with handcrafted convolutional kernels. This serves as a basis of understanding how modern AI uses convolutional kernels to detect shapes and objects. In this lab, we will be applying convolutional kernels to a grayscale image (only one image channel). The following are the six tasks that you should deliver. Collectively, tasks 1-5 implement an edge detection technique (Canny edge detection). Task 6 requires an application of the Canny edge detector that you have created on another image.

- Task 1: Implement Gaussian blur
- Task 2: Calculate image gradients
- Task 3: Calculate gradient magnitude
- Task 4: Calculate gradient orientation
- Task 5: Non-maxima Suppression and thresholding

[1] I recommend writing the convolution and non-maxima suppression/ thresholding manually, as you will be required to perform tasks like these (and hand calculations) in the final exam, without assistance, to demonstrate your understanding of the underlying algorithms.

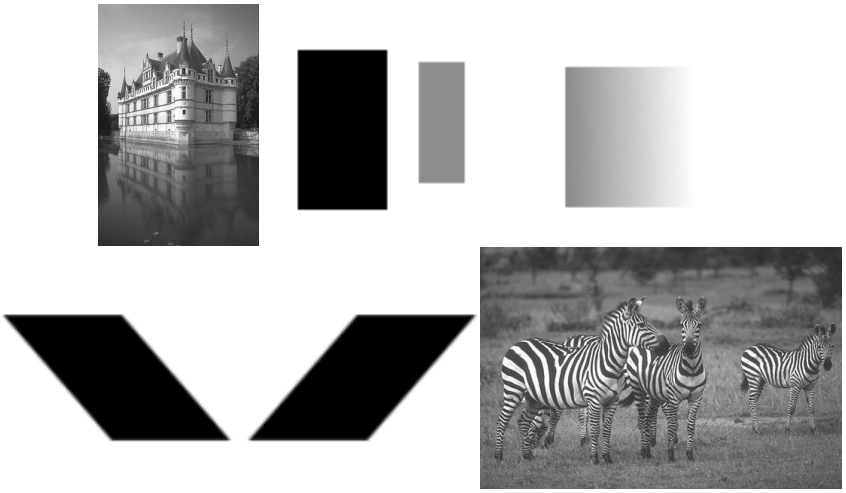


Figure 1: Test images for tasks 1-5

- Task 6: Find the helipad

References:

- Canny edge detector overview
- Non-maximum suppression for task 5

Resources:

- Lab1_student_template.ipynb - please complete the lab tasks in this template notebook and use the markdown spaces provided to report your findings/ describe your approach.
- Test images are located in the lab1 folder. These should be used to test your code.

Task 1. Implement Gaussian blur

Write a program that performs Gaussian blur on an input image using the following 5x5 kernel, where B is the blurred version of input image A. You code should not use a pre-existing conv function. Show results on screen.

$$\mathbf{B} = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * \mathbf{A}$$

Task 2. Calculate image gradients

Write a program that does the following in order. You should reuse your Task 1 code

1. Load an image from the hard drive as a grayscale (single channel) image.
2. Blur the image using a 5x5 Gaussian filter
3. Calculate the gradient of the blurred image in the x and y directions using a 3x3 Sobel filter
4. Check the x and y gradient values to make sure they are correct

Task 3. Calculate gradient magnitude

Extend your program to calculate the gradient magnitude. Gradient magnitude G is a function of the gradients in the x and y directions (G_x, G_y)

$$G = \sqrt{G_x^2 + G_y^2}$$

Show the gradient magnitude on screen as a grayscale image

Task 4. Calculate gradient orientation

Extend your program to calculate the gradient orientation as follows

$$\Theta = arctan(\frac{G_y}{G_x})$$

Note that gradient orientations should be rounded to the nearest 45-degrees (Compass directions: North, North-East, East etc). Rounding can be performed using a combination of integer arithmetic and the np.floor() function. Display the gradient orientations on screen to check that they are correct. Use colours to illustrate the different orientations.

Task 5. Non-maxima Suppression and thresholding

Extend your code to perform non-maxima suppression in order to "thin" the edges. A detailed summary is provided in the references above. The core idea is to zero any pixel that is not greater in terms of gradient magnitude than both pixels on either side of its gradient orientation.

Threshold the non-maxima suppression results. The final output should be a black and white (binary) image showing the edge pixels. Test your code against the test images.

Task 6. Find the helipad

Finally, use the code above to automatically locate the helipad in the test image provided. Return the row and column of the centre of the helipad, and plot an x on the image to show where this point lies. For this task, marks will be awarded based on how close your algorithm gets to the true centre of the helipad.