# IPA 2025 Lab Exercise 1 (15 Points)

**Exercise due: 19th of October 2025 23:55**

## How to submit your report?

Read this document carefully. Ask if you are unsure what to do; otherwise, use common sense to solve the problems. The hand-in for these laboratory exercises must be done using Moodle before the deadline (see http://vda.univie.ac.at/Teaching/IPA/25w/ for more information). You should submit a compressed (zip, 7z) file named *IPA_Lab1_YourLastname.7z*, containing the following:

- **A written report of the lab as a PDF document** (*YourLastname_Report.pdf*), including your results and, most importantly, a discussion of those results.
- **All Python code necessary to run your solution**, including **only** the images you used.
    - Feel free to define auxiliary functions where needed, but name your main files as specified in each task.
    - Please write your name and student ID as a comment at the beginning of each Python file and in the PDF report.
    - Include many comments in your source code, as this is required.
- Please do not send the entire original image folder back to us, as we already have it.

You may discuss the results and solutions with others, but you must submit your own work.

Feel free to use Jupyter Notebook if you find it easier.

**Warning!** Plagiarism will not be tolerated. If the submitted code does not run, we will reject your exercise completely. Be sure to check it before submission.

It is not necessary to include a copy of all the code in the PDF document, although key parts can be included if they are needed to explain a point.

Please find the images on the Moodle web page and follow the instructions.

# 1. Edge detection (2 points)

Consider a black image (0) with a single pixel of value 1 at the center. Sketch (using numerical values) what the gradient magnitude image (using $Eq.$ 1.) would look like if we use Prewitt kernels (Fig. 14. shown below) to compute $g_x$ and $g_y$ [watch the signs of $g_x$ and $g_y$].

Eq. 1

$$M(x,y) = \|\nabla f(x,y)\| = \sqrt{{g_x}^2 + {g_y}^2}$$

Scan your answer and include in the submission of your work.

| | | |
|---|---|---|
| $z_1$ | $z_2$ | $z_3$ |
| $z_4$ | $z_5$ | $z_6$ |
| $z_7$ | $z_8$ | $z_9$ |

| | | | |
|---|---|---|---|
| −1 | 0 | 0 | −1 |
| 0 | 1 | 1 | 0 |

Roberts

| | | | | | |
|---|---|---|---|---|---|
| −1 | −1 | −1 | −1 | 0 | 1 |
| 0 | 0 | 0 | −1 | 0 | 1 |
| 1 | 1 | 1 | −1 | 0 | 1 |

Prewitt

| | | | | | |
|---|---|---|---|---|---|
| −1 | −2 | −1 | −1 | 0 | 1 |
| 0 | 0 | 0 | −2 | 0 | 2 |
| 1 | 2 | 1 | −1 | 0 | 1 |

Sobel

a b c d e f g

**FIGURE 10.14** A 3 × 3 region of an image (the $z$'s are intensity values) and various masks used to compute the gradient at the point labeled $z_5$.

## 2. Comparing edge detection algorithms (3 points)

Use the image in Fig. 2. to compare the results of **Sobel**, **Prewitt** and of the **Canny edge detection** (you can use implementations from an image processing library, or write you own version of edge detectors). Write a script **`task2`** and discuss the results and the effects of the parameters used by the algorithms in detecting the blood vessels in the image in Fig. 2.

Fig.2. Angiogram Aortic - Kidney © Gonzales et. al

## 3 Moving Average thresholding (5 Points)

- Write a **function [g,ma] = movingThresh4e(f,n, pcnt)** to implement the moving average thresholding scheme discussed in section 10.3. Input: **f** is the image to be thresholded, **n** is the number of pixels used to compute the average (including the current pixel), and **pcnt** (a positive scalar) is such that if the value of the image at a point exceed **pcnt** percent of the moving average at that point, a 1 is output for **g** at that location (**pcnt** can be greater than 1). Otherwise a 0 is output for **g** at that location. Output **ma** is a 1-D array containing the values of all the moving average. (**Hints:** Consider using the zigzag approach discussed in the book chapter on Image compression and watermarking, by flipping every other row in the image (see function **fliplr** in numpy) and then converting the resulting matrix to a 1-D array. This will allow you to use function like **lfilter** in scipy.signal to perform the running average. At the end convert back to a 2-D array using function **reshape** in nympy)
- Read the image **text-spotshade.tif** (Fig. 3) and duplicate the result in Fig. 10.44(c).

To showcase write a script **task3.**

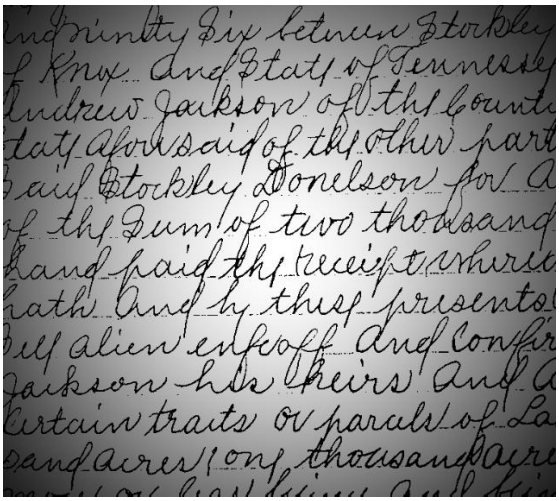

Fig.3. Text spot shade © Gonzales et. al

## 4. Otsu's Algorithm (5 points)

- Write a function **[g,sep,kstar]=otsuThresh(f)** that implements Otsu's thresholding algorithm following the 7 steps as shown in page 813 of the book (Section 10.3). (Hint. Organize your code based on the **seven steps listen in the algorithm**)
- Read the image **polymercell.tiff** (Fig. 4) and threshold using the **otsuThresh**. Display your results to confirm that you get the same result as in Fig. 10.36(d). Display the threshold value. Discuss the results.

To showcase write a script **task4.**

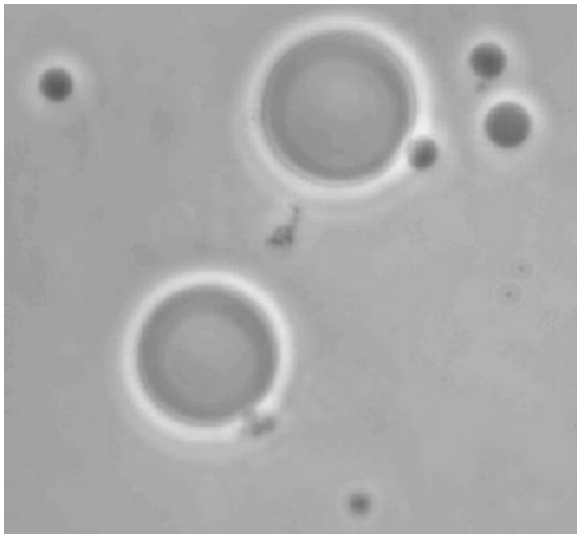

Fig.4. Polymer cell © Gonzales et. al