Computer Vision 2025
Assignment 1: Image Filtering

1 Overview

For this assignment you will research, implement and test some image filtering operations. Image filtering by convolution is a fundamental step in many computer vision tasks and you will find it useful to have a firm grasp of how it works. For example, later in the course we will come across Convolutional Neural Networks (CNNs) which are a combination of convolutional filters learned to solve specific tasks. The main aims of the assignment are:

- to understand the basics of how images are stored and processed in memory;
- to gain exposure to several common image filters, and understand how they work;
- to get practical experience implementing convolutional image filters;
- to test your intuition about image filtering by running some experiments;
- to report your results in a clear and concise manner.

This assignment relates to the following ACS CBOK areas: abstraction, design, hardware and software, data and information, HCI and programming.

2 Setup

In this assignment, you will write a report and some code. While you are required to submit both, *you will be graded based primarily on your report.*.

There are many software libraries that can perform image filtering supporting various languages and environments. For this assignment you should use python, and to that end we provide code stubs and a jupyter notebook (ipynb file) to get you started. You *must* use this setup so that we can verify code and/or provide help if you get your setup wrong.

The notebook can be uploaded and run in a google colab environment, with all processing and file management happening on a remote server, mostly invisible to you. Google colab provides a web interface/environment for your python code development that looks a lot like the jupyter notebook interface. This will be the easiest way to do the work offline.

If you prefer the code be able to run locally, you need to set up python and jupyter notebooks on your local computer. The easiest way to do this is to install anaconda, a nicely packaged version of python that comes with Jupyter notebooks built-in. You run jupyter or jupyter-lab as a server on your local machine and then communicate with that through a web-browser which provides the interface to the IDE and the language. There may be additional packages you have to install, such as opencv, which do not come as part of anaconda. Familiarize yourself with the anaconda package management as later assignments will require you to install deep learning packages as well.

More detailed instructions on setups is provided in a separate document available through the MyUni course pages, and will also be discussed in workshops. There are, of course, also lots of online tutorials about Colab, Jupyter Notebooks and Anaconda.

3 Instructions

This assignment is divided into several tasks. For each task, you should write a corresponding section in your report that addresses all questions posed. In most cases, this will be one or more images and a *small* amount of text and maybe code fragment to explain them and describe what you did. **Make sure to read through the whole of the assignment before tackling the tasks in order.**

Task 1: Loading and displaying an image (10%)

Create or download several (minimum of 3 and maximum of 5) test images that you will apply your filters to. Choose your images so that they will test different aspects of your filters. Think about this carefully when you choose these images. Looking at some of the filtering examples covered in the lecture might help in isolating different aspect of the filtering technique in question. Include at least one grayscale image and one RGB image. Show each image in your report, and explain why you have chosen it in light of the exercise. It is important thus to read the tasks before carefully choosing the images.

1. Write some code to read an image from file and display it. Apply it to your test images and then apply some simple point processes such as those described in Lecture 1, and observe the results. What happens when a processed pixel value becomes < 0 or > 255, and what effect does this have on later processing?

2. Repeat the above for a RGB test image, but this time, apply the point process only to one channel of the image (red, green or blue). Display the resulting RGB image.

Task 2: Image processing (30%)

Now that you can load and display images, let's try some operations on them. Note that there are stubs for each of these functions in `a1code.py`

1. Implement a crop() function to create a new image that is a small crop from the larger image.

2. Implement a resize() function that works by *sampling* from the original image.

3. Implement a change_contrast() function to enhance the image contrast.

4. Implement a greyscale() to convert an RGB image into monochrome

5. Implement a binary() function that thresholds an image based on an intensity threshold. What do you observe when you change the threshold of the binary function?

Apply all these functions with different parameters on your own test images and discuss the results.

Task 3: Image filtering (30%)

See section 3.2 of the textbook [1] for background information on this question.

1. Implement in stages 2D convolution filter for RGB images:

- Using the definition of 2D convolution from week 1, implement the convolution operation in the function conv2D().
- In the function conv, extend your function conv2D to work on RGB images, by applying the 2D convolution to each channel independently.

2. Use the gauss2D function provided in a1code.py to create a Gaussian kernel, and apply it to your images with convolution. You will obtain marks for trying different tests and analyzing the results, for example:

   - try varying the image size, and the size and variance of the filter.
   - subtract the filtered image from the original.

   What do you observe and why?

3. Define a horizontal and vertical Sobel edge filter kernel and test them on your images. You will obtain marks for testing them and displaying results in interesting ways, for example:

   - apply them to an image at different scales.
   - considering how to display positive and negative gradients.
   - apply different combinations of horizontal and vertical filters, such as applying the vertical Sobel filter to the output of the horizontal Sobel filter. What do you see?

Task 4: Image sampling and pyramids (30%)

NOTE: image sampling will be covered in Lecture 2. See sections 3.5.1 and 3.5.2 of the textbook [1] for this question.

1. Apply your resize() function to reduce an image to 0.5 x height and 0.5 x width (i.e. down-sample the image to half size)

   Repeat the above procedure, but apply a Gaussian blur filter to your original image before downsampling it. How does the result compare to your previous output, and to the original image? Why?

2. Create a Gaussian pyramid as described in week2's lecture on an image.

   Apply a Gaussian kernel to an image I, and resize it with ratio 0.5, to get I_1. Repeat this step to get I_2, I_3 and I_4.

   Display these four images in a manner analogs to the example shown in the lectures.

Task 5: Implement a blob detector (Bonus 20%)

NOTE: mage filtering lectures, particularly Lecture 2, have covered the details related to this question. This is a bonus question for students to get opportunities to recover lost marks in the other parts of the assignment. Note that the overall marks will be capped at 100%.

1. Create a Laplacian of Gaussian (LoG) filter in the function 'LoG2D()' and visualize its response on your images. You can use the template function (and hints therein) for the task if you wish.

2. Modify parameters of the LoG filters and apply them to an image of your choice. Show how these variations are manifested in the output.

3. Repeat the experiment by rescaling the image with a combination of appropriate filters designed by you for these assignment. What correlations do you find when changing the scale or modifying the filters?

4. How does the response of LoG filter change when you rotate the image by 90 degrees? You can write a function to rotate the image or use an externally rotated image for this task.

4 Assessment

Hand in your report and supporting code/images via the MyUni submission page for the Assignment. Upload two files:

1. `report.pdf`, a PDF version of your report

2. `code.zip` a zip file containing your code and images

Make sure you include all the code you've written and your test images. Your mark will be based primarily on your report. The code will be used to check your work and ensure it has been done independently and not plagiarized.
The assignment is worth 20% of your overall mark for the course. It is due on Monday, March 31 at 11.59pm.

Ravi Garg
March 2025.

References

[1] Richard Szeliski, *Computer Vision: Algorithms and Applications*, Springer, 2011. `http://szeliski.org/Book/`