TDT4195: Visual Computing Fundamentals

Digital Image Processing - Assignment 1

October 12, 2017

Aleksander Rognhaugen
Department of Computer Science
Norwegian University of Science and Technology (NTNU)

- **Delivery deadline: October 27, 2017** by 23:00.
- **This assignment counts towards 4 % of your final grade.**
- You can work on your own or in groups of two people.
- Deliver your solution on *Blackboard* before the deadline.
- Please upload your report as a PDF file, and package your code into a single ZIP archive. Not following this format may result in a score deduction.
- The programming tasks may be completed in the programming language of your choice, however, it might be a good idea to select one that supports matrix and image processing operations, e.g. MATLAB or Python with NumPy. *The lab computers at ITS-015 support Python and MATLAB.*
- The delivered code is taken into account with the evaluation. Ensure your code is documented and as readable as possible.
- For each programming task you need to give a brief explanation of what you did, answer any questions in the task text, and show any results, e.g. images, in the report.

Objective: Gain experience with (a) how to process images on a computer by using basic intensity transformations, and (b) how spatial filtering works by implementing two-dimensional convolution.

Digital Image Processing Assignments Overview

The image processing assignments in this course are meant to supplement the lectures by providing theoretical and practical experience with the material.

There are three assignments in total and together they are worth 15 % of your final grade. The assignments aim to show you how to look at images from a computational perspective, and introduce you to a variety of techniques to manipulate them. This include extracting or highlighting specific regions of interest, or improving the quality of the image overall.

The assignments can be quite difficult and may at times be programming and maths heavy. For this reason, I recommend you to start working on the assignments early. Seek out help whenever you get stuck by making use of the assignment forum and the computer lab sessions.

- **Assignment 1** (worth 4 %) introduces you to intensity transformations and spatial filtering.
- **Assignment 2** (worth 5 %) shifts the focus from spatial to frequency filtering using the convolution theorem.
- **Assignment 3** (worth 6 %) gives an introduction to image segmentation and mathematical morphology.

To make the transition to programming with images a bit easier, we have made available a *getting started* guide for Python (with NumPy) and MATLAB. This guide walks through the basics of how to create and manipulate tensors[1], with a focus on vectors and matrices. Additionally, the guide demonstrates how to read and write images, manipulate pixels, inspect image histograms, traverse images in scanline, and more.

A link to the guide can be found below:
`https://github.com/vicolab/tdt4195-public`

[1] For our purposes, tensors are arrays of numbers organised in a regular grid with an arbitrary number of axes.

Task 1: Theory Questions [1.5 points]

a) **[0.5 points]** Histogram equalisation is an algorithm that creates a transformation $\mathcal{T}_{heq}$ by exploiting an image histogram. Explain in your own words:

i) How can we see that an image has low contrast when looking at its histogram?

ii) What effect does $\mathcal{T}_{heq}$ have when applied on an image?

iii) What happens when histogram equalisation is applied multiple times on the same image in sequence?

b) **[0.3 points]** Perform histogram equalisation by hand on the 4-bit ($2^4 = 16$ intensity values) 3×4 image in Figure 1. Your report must include all the steps you did to compute the transformation, the transformed image, and its histogram.

12	3	1	9
3	0	4	3
2	6	15	2

Figure 1: An image with intensities in the $[0, 15]$ range (4-bit). Each square represents a pixel, where the number within is the intensity value.

c) **[0.2 points]** The convolution operator is associative. What does this mean, and how is this property beneficial for convolution?

Hint: The same property is convenient for affine transformations as well.

d) **[0.5 points]** An $M \times M$ kernel is *separable* if it can be decomposed into an $M \times 1$ and $1 \times M$ vector whose outer product $\otimes$ form the original kernel. To determine if a kernel is separable we can compute its *rank*[2]. A separable kernel has rank equal to 1. The rank of a matrix can be found by counting the number of nonzero rows remaining after *Gaussian elimination*. Answer the following questions in your own words. Please show the steps you took when calculating your answers.

i) Determine if the convolution kernel (a) and (b) in Figure 2 are separable by computing their rank. If a kernel is separable, also provide the separated row and column vector.

ii) Why is it helpful to have convolution kernels that are separable?

[2]The column (or row) rank of a matrix $\boldsymbol{A}$ is the maximal number of linearly independent columns (or rows) of $\boldsymbol{A}$.

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} \qquad \begin{bmatrix}1 & 1 & 1\\ 1 & -8 & 1\\ 1 & 1 & 1\end{bmatrix}$$

(a) (b)

Figure 2: Two convolution kernels: (a) is a Gaussian kernel, and (b) approximates the Laplacian $\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$ (includes diagonal entries).

Task 2: Greyscale Conversion [0.5 points]

Converting a colour image to a greyscale representation can be done by taking the average of the red (R), green (G), and blue (B) channels, as seen on the left-hand side of Equation 1. However, because different colours have different wavelengths it follows that each colour has a slightly different contribution to the image. Thus, to preserve the luminance, or lightness, of the original colour image a weighted average is typically used instead. One such weighted average – used by the sRGB colour space – can be seen on the right-hand side of Equation 1.[3]

$$\text{grey}_{i,j} = \frac{\text{R}_{i,j} + \text{G}_{i,j} + \text{B}_{i,j}}{3} \qquad \text{grey}_{i,j} = 0.2126\text{R}_{i,j} + 0.7152\text{G}_{i,j} + 0.0722\text{B}_{i,j} \quad (1)$$

a) **[0.3 points]** Implement *two* functions that manually convert a RGB colour image to a greyscale representation. The first function must implement the averaging method, while the second must implement the luminance-preserving method (see Equation 1).

b) **[0.2 points]** Apply your functions on a couple of colour images and show the result in your report. Briefly discuss the perceived differences between the output of your two functions.

Hint: The data type of your variables greatly impact the result of arithmetic operations.

Task 3: Intensity Transformations [0.5 points]

A greyscale intensity, or pixel brightness, transformation $\mathcal{T}$ maps the intensity values $p \in [p_0, p_k]$ in the original image into a new intensity range $q \in [q_0, q_k]$:

$$q = \mathcal{T}(p) \Leftrightarrow J_{i,j} = \mathcal{T}(I_{i,j}) \quad (2)$$

a) **[0.2 points]** Implement a function that takes a greyscale image and applies the following intensity transformation: $\mathcal{T}(p) = p_k - p$. Include the following in your report:

[3] More information can be found here: `https://en.wikipedia.org/wiki/Greyscale`.

i) In your own words, explain what this transformation does. What would you call it?

ii) Apply the function on an image of your choice and show the result.

Hint: Remember, p_k is the highest value a pixel can take, e.g. 15 for a 4-bit image.

A basic definition of the gamma transformation can be seen in Equation 3. It adjusts the overall intensity of an image by squeezing pixel intensities to either low or high intensities.

$$\mathcal{T}(p) = cp^{\gamma}, \qquad c > 0 \wedge \gamma > 0 \tag{3}$$

b) **[0.3 points]** Implement a function that takes a greyscale image and applies the gamma transformation from Equation 3. You can let $c = 1$. For this task, it is important that you normalise the image, so that the intensity values lie $\in [0, 1]$, before applying the gamma transformation[4]. Include the following in your report:

i) In your own words, explain what happens with the image intensity values when $\gamma > 1$ and $\gamma < 1$?

ii) Test out different γ-values on a greyscale image and show the result.

Task 4: Spatial Convolution [1.5 points]

Convolution $f * h$ is an operation on two functions f and h. When f and h are defined on spatial variables, the operation is called spatial convolution. Formally, for functions $f(x, y)$ and $h(x, y)$ of two discrete variables x and y, two-dimensional convolution is defined as:

$$(f * h)(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f(i, j)h(x - i, y - j) \tag{4}$$

Convolution is commutative – that is, $f * h = h * f$ – however, to simplify explanations we will let f be the image and h be the convolution filter/kernel/mask. We can think of two-dimensional spatial convolution as sliding a kernel across the width and height of an image. For each pixel in the image, a linear combination between the kernel, centred on the image pixel, and a local neighbourhood around the image pixel is evaluated. The result is assigned to the output image.

a) **[0.5 points]** Implement a function that takes a greyscale image and an arbitrary linear convolution kernel and performs two-dimensional spatial convolution. Assume that the size of the convolution kernel is odd numbered, e.g. 3×3, 5×5, or 7×7. You *must* implement the convolution procedure yourself from scratch. You are not required to implement procedures for adding and removing padding. In your own words, explain how you implemented convolution your report.

[4] Assuming we have an 8-bit image, with intensity values $\in [0, 255]$, then all we have to do is change the data type to `float` and divide by 255.

Equation 5 shows two convolution kernels that are used for smoothing images. The one on the left is a 3×3 averaging kernel, while the one on the right is a 5×5 Gaussian kernel approximated using binomial coefficients.

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \qquad \frac{1}{256}\begin{bmatrix}1 & 4 & 6 & 4 & 1\\ 4 & 16 & 24 & 16 & 4\\ 6 & 24 & 36 & 24 & 6\\ 4 & 16 & 24 & 16 & 4\\ 1 & 4 & 6 & 4 & 1\end{bmatrix} \tag{5}$$

b) **[0.2 points]** Try the convolution function you made in task (a) by displaying the result of the following in your report:

i) Convolve a *greyscale* image with the smoothing kernels in Equation 5.

ii) Convolve a *colour* image with the smoothing kernels in Equation 5.

Hint: To run convolution on a colour image, convolve each channel separately and layer them afterwards.

The Sobel operator consists of a pair of directional convolution kernels and is used to measure the gradient of an image. The convolution kernels for the horizontal and vertical direction can be seen in Equation 6. They can be applied separately on an image $\boldsymbol{I}$, producing $\boldsymbol{I_x}$ and $\boldsymbol{I_y}$. However, they can also be combined to measure the magitude of the gradient at each point: $|\nabla| = \sqrt{\boldsymbol{I_x}^2 + \boldsymbol{I_y}^2}$.

$$\begin{bmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{bmatrix} \qquad \begin{bmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{bmatrix} \tag{6}$$

c) **[0.8 points]** Using your convolution implementation from (a), carry out the following and record the result in your report:

i) Use the convolution kernels in Equation 6 to approximate the horizontal and vertical gradient of a greyscale image.

ii) Compute the magnitude of the gradients $|\nabla|$.

iii) In your own words, explain what $|\nabla|$ tell us about the image.

Hint: Convolution with Sobel kernels may yield negative numbers. This means that your implementation must output images with signed numbers.