COMP9517: Computer Vision

2023 T3 Lab 1 Specification

Maximum Marks Achievable: 2.5

This lab is **worth 2.5% of the total course mark**.

The lab files should be submitted online.
Instructions for submission will be posted closer to the deadline.
Deadline for submission is Week 3, Wednesday 27 September 2023, 18:00:00.

Objectives: This lab revisits important concepts covered in the Week 1 and Week 2 lectures and aims to make you familiar with implementing specific algorithms.

Preliminaries: As mentioned in the first lecture, we assume you are familiar with programming in Python or are willing to learn it independently. You do not need to be an expert, as you will further develop your skills during the course, but you should at least know the basics. If you do not yet know Python, we assume you are familiar with at least one other programming language such as C, in which case it should be relatively easy to learn Python.

To learn or brush up your Python skills, see several free online resources listed at the end of this document. Especially if you already know C or similar languages, there is no need to go through all the linked resources in detail. Just quickly learn the syntax and the main features of the language. The rest will follow as you go.

For implementing and testing computer vision algorithms, we use OpenCV in this course. OpenCV is a library of programming functions mainly for computer vision. The library is cross-platform and licensed as free and open-source software under Apache License 2. It also supports training and execution of machine/deep learning models. Originally written in C, with new algorithms developed in C++, it has wrappers for languages such as Python and Java. As stated above, in this course we will focus on programming in Python. See the links below for OpenCV tutorials and documentation.

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 first 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.

Materials: The sample 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) which 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.

1. Contrast Stretching

Contrast is a measure of the range of intensity values in an image and is defined as the difference between the maximum pixel value and minimum pixel value. The maximum possible contrast of an 8-bit image is 255 (max) – 0 (min) = 255. Any value less than that means the image has lower contrast than possible. Contrast stretching attempts to improve the contrast of the image by stretching the range of intensity values using linear scaling.

Assume that I is the original input image and O is the output image. Let a and b be the minimum and maximum pixel values allowed (for an 8-bit image that means $a = 0$ and $b = 255$) and let c and d be the minimum and maximum pixel values found in I, respectively. Then the contrast-stretched image O is given by the function:

$$O(x, y) = (I(x, y) - c)\left(\frac{b - a}{d - c}\right) + a \tag{1}$$

Task (0.75 mark): Write an algorithm that performs contrast stretching as per Equation (1) above. Read the given image **Oakland.png** and execute your algorithm to see whether it indeed improves the contrast. Notice that this is a colour image, which has three channels (R, G, B), so you need to somehow apply your algorithm to the three channels.

There are different possibilities here and we consider three of them:

- The most straightforward is to apply the algorithm to each of the image channels (R, G, B) individually. That is, the mapping function (1) is calculated for each channel separately and may be different for each channel depending on its c and d values.

- Alternatively, you could convert the colour image to a gray-level image, for example using the formula Y = 0.299R + 0.587G + 0.114B, then calculate the mapping function (1) on the gray-level image (Y), and apply it to the channels (R, G, B) of the original colour image. In this case, the same mapping function is applied to all channels.

- Finally, you could convert the colour image to a different colour space, such as HSV, then calculate the mapping function (1) on the value (V) channel, and apply it to the channels (R, G, B) of the original colour image. Here again, a single mapping function is calculated and applied to all image channels.

In your notebook, execute your algorithm and show the input image and output image next to each other, for each of the three approaches. Also briefly discuss in a comment in your notebook which approach yields the best contrast-stretched colour image and provide reasons for why that approach works better than the other two.

2. Histogram Calculation

The histogram of an image shows the counts of the intensity values. It gives only statistical information about the pixels and removes the location information. For a digital image with L gray levels, from 0 to $L-1$, the histogram is a discrete function $h(i) = n_i$ where $i \in [0, L-1]$ is the ith gray level and n_i is the number of pixels having that gray level.

Task (0.5 mark): Write an algorithm that computes and plots the histogram of an image and also reports the minimum pixel value and the maximum pixel value in the image. Then execute your algorithm to compare the histograms and extreme values before and after contrast stretching of image **Oakland.png** for each of the three approaches in the previous task.

More specifically, for the first contrast-stretching approach, show the histogram and extreme values for each of the three channels (R, G, B) of both the input image and the output image. For the second approach, show the histogram and extreme values of only the gray value representation (Y) of both the input image and the output image after conversion. For the third approach, show the histogram and extreme values of only the value channel (V) of both the input image and the output image after conversion.

To facilitate visual comparison, present the histograms of the input image and corresponding output image side by side in each case.

3. Image Thresholding

A crucial first step for quantitative analysis of objects (or regions) of interest in images, is to identify which pixels belong to the objects (the relevant pixels) and which belong to the background (the irrelevant pixels). This task is called image segmentation.

The simplest technique to perform this task is thresholding. Here, a pixel is considered to belong to an object if its value is above the threshold, and to the background if its value is lower than or equal to the threshold.

While an optimal threshold for each image could be selected manually by the user, this is undesirable in applications that require full automation. Fortunately, several automatic thresholding techniques exist, as discussed in the lecture.

Task (0.75 mark): Write an algorithm that can threshold an image using the three different thresholding methods discussed in the lecture: Otsu, IsoData, Triangle. Apply your algorithm to the images **Hardware.png** (the objects are the dark nuts and bolts), **Nuclei.png** (the objects are the bright cell nuclei), and **Orca.png** (the object of interest is the Orca).

In your notebook, show the results in table form to facilitate visual comparison of all images. For example, one table row per input image, successively showing the input image, its histogram, and the thresholding results using the three methods.

Also briefly discuss the differences in the results in your notebook and provide explanations (based on the histograms or otherwise) why for some images one thresholding method may work better than others, while for other images it may be the other way around, or perhaps in some cases none of the methods work well. Present some general rules of thumb for which thresholding methods are best for what kind of images.

4. Edge Detection

Edges are an important source of semantic information in images. A gray-scale image can be thought of as a 2D landscape with areas of different intensities corresponding to different heights. The edges are the transitions from one such area to the next.

The Laplacian is a second-order derivative operator that can be used to find edges. It emphasizes pixels in areas of strong intensity changes and de-emphasizes pixels in areas with slowly varying intensities. The edges are the zero-crossings in the Laplacian image.

One example (there exist several) of a 3×3 pixel convolution kernel that approximates the Laplacian operator is the following:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Task (0.5 mark): Write an algorithm that computes the Laplacian image of an input image using the above kernel. Apply the algorithm to the image **Laplace.png**.

Notice that the calculations may produce negative output pixel values. Thus, make sure you use the right data types for the calculations and for the output image, and the right intensity mapping to display the output image.

Coding Requirements

Make sure that in your Jupyter notebook, the input images are readable from the 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.

Free Online Python Resources:

W3Schools Python Tutorial
https://www.w3schools.com/python/

LearnPython.Org
https://www.learnpython.org/

Python For Beginners
https://www.python.org/about/gettingstarted/

Harvard's Introduction to Programming with Python
https://cs50.harvard.edu/python/

Google's Python Class
https://developers.google.com/edu/python/

FreeCodeCamp's Python in 4 Hours Full Course on YouTube (40M Views)
https://www.youtube.com/watch?v=rfscVS0vtbw

Free OpenCV Resources:

About OpenCV
https://opencv.org/about/

OpenCV Tutorials
https://docs.opencv.org/4.x/d9/df8/tutorial_root.html

OpenCV Wiki
https://github.com/opencv/opencv/wiki

OpenCV Documentation
https://docs.opencv.org/

Released: 20 September 2023