COMP9517: Computer Vision

2021 T3 Lab 1 Specification

Maximum Marks Achievable: 2.5

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

The lab files should be submitted online. Instructions for submission will be posted closer to the deadline. **Deadline for submission is Week 3, Wednesday 29 September 2021, 23:59:59.**

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.

Materials: The sample image to be used in all the questions of this lab is available in WebCMS3. You are required to use OpenCV 3+ with Python 3+.

Submission: Question 4 is assessable **after the lab**. Submit your source code as a Jupyter notebook (.ipynb) with output images (.png) in a single zip file via Moodle by the above deadline. The submission link will be announced in due time.

The sample image **Cat.png** is to be used for all four questions. The output image of some questions may be used as the input to other questions.

1. Contrast Stretching

Contrast in an image is a measure of the range of intensity values and is the difference between the maximum and minimum pixel values. The full contrast of an 8-bit image is 255 (max) – 0 (min) = 255. Anything 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 in I. Then the contrast-stretched image O is given by the function:

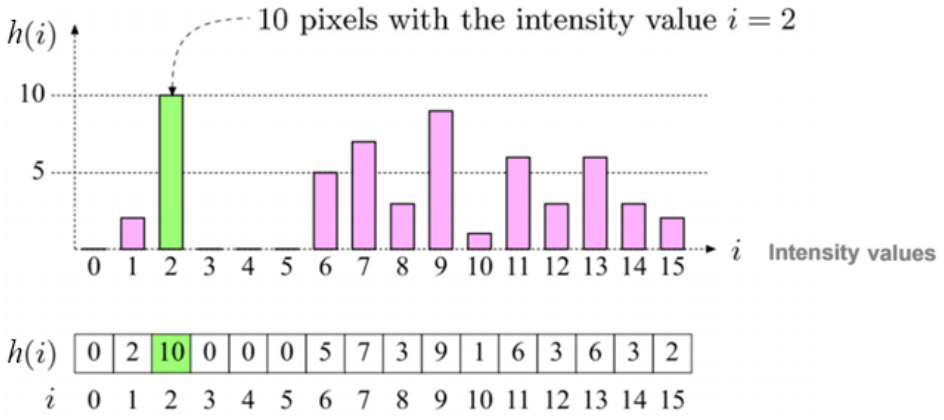
$$O(x, y) = (I(x, y) - c)\left(\frac{b - a}{d - c}\right) + a \qquad (1)$$

比例

Question 1: Write an algorithm that performs contrast stretching as per Equation (1) above. Read the given gray-scale image and run your algorithm to see whether it indeed improves the image quality. Also write an algorithm that finds the coordinates of the minimum pixel value and the coordinates of the maximum pixel value in an image. Run it on both the input image and the output image and look up the values at these pixels to confirm whether your contrast stretching algorithm works correctly.

2. Intensity Histogram

The histogram of an image shows the frequency of pixel 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 with that gray level.



Question 2: Write an algorithm that computes and plots the histogram of an image. Do not use the built-in OpenCV functions for computing the histogram but write your own code to perform this task. Then run your algorithm on the given input image and the contrast-stretched output image from Question 1 and visually compare the histograms.

3. Image Edges

Edges are an important source of semantic information in images. They occur in human visual perception at divisions between areas of different intensity, colour, or texture. A gray-scale image can be thought of as a 2D landscape with areas of different intensity living at different heights. A transition between areas of different intensity in an image I means there must be a steep slope, which we formalise as the gradient (vector):

$$\nabla I = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) \tag{2}$$

As the image I is discrete, we need to approximate the continuous derivatives $\partial I/\partial x$ and $\partial I/\partial y$ by finite differences. Simple examples of convolution kernels that perform finite differencing are the Sobel filters defined as follows:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Question 3: Write an algorithm that computes the two Sobel images $\partial I/\partial x \approx I * S_x$ and $\partial I/\partial y \approx I * S_y$ from the given image. Do not use the built-in OpenCV functions for computing the Sobel images but write your own code to perform this task. You may verify the output of your own algorithm by comparing with the output of the built-in functions.

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

After that, compute the gradient magnitude image:

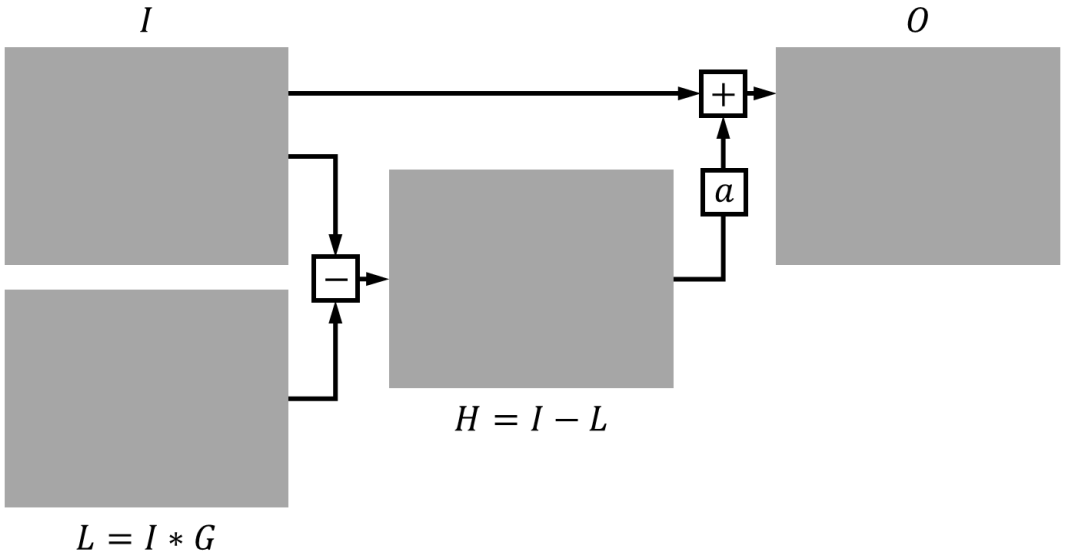
$$\|\nabla I\| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \tag{3}$$

In other words, create a new output image having the same size as the input image and the Sobel images, and then for every pixel in the output image compute the value as the square root of the sum of the squared value of the Sobel image $\partial I/\partial x$ and the squared value of the Sobel image $\partial I/\partial y$ at that pixel position.

4. Image Sharpening

Sometimes an image is not as sharp as it could be, and fine details look a bit blurred. An image can be sharpened by using specific filters that enhance the high-frequency content in the image. One possible technique to accomplish this is called unsharp masking.

The figure below gives a schematic overview of this technique. In words, it takes the input image I and convolves it with a Gaussian kernel G with standard deviation (also called the scale) σ, resulting in a slightly blurred image L. Next, the blurred image L is subtracted pixelwise from the input image I, resulting in image H, in which the high frequencies are enhanced. Then, each pixel in H is multiplied by a constant factor a, and the resulting image is added pixelwise to the input image I, resulting in output image O.



Question 4 (2.5 marks): Write an algorithm that implements the above technique. Apply it to the given image using, for example, $\sigma = 1.0$ pixels and $a = 1.25$. The differences between I and O are subtle but the latter should be visibly sharper.

Notice that this technique may produce output pixel values outside the range $[0,255]$. Thus, make sure you use the right data types for the computations. Also, you can apply contrast stretching (which you implemented in answer to Question 1) to force the contrast in I and O to be in the same range $[0,255]$, so you can properly compare the two images visually.

Coding Requirements
For all tasks, implement the required algorithms yourself, and **do not use library functions** from OpenCV (or any other packages) for these tasks. Using these functions instead of your own implementation will result in deduction of points.

In your Jupyter notebook, the input image should be readable from the location specified as an argument, and all output images and other requested results should be 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.

Released: 21 September 2021