COMP9517: Computer Vision

2021 Term 3

Assignment Specification

Maximum Marks Achievable: 10

This assignment is **worth 10% of the total course marks**.

The assignment files should be submitted online.

Instructions for submission will be posted closer to the deadline.

Deadline for submission is Week 4, Wednesday 6 October 2021, 23:59:59.

Deliverables: You must submit the following items:

1. A report explaining the approach you have taken in Tasks 1, 2, and 3. The report can be written in Word, LaTeX, or a similar text processor, but must be submitted as a PDF and include your name and zID on the first page. It must also show the sample input images and the intermediate and final output images obtained. No templates are provided, but the report must be no more than 5 A4 pages, and must be single column, use 11 points Times New Roman or similar font, and have 2.5 cm margins.
2. Output images as separate files (in .png or similar lossless format).
3. Source code in the form of a Jupyter notebook (.ipynb).

Submission: Submit all deliverables in a single zip-file via Moodle. Instructions for submission will be posted closer to the deadline. To avoid confusion for the tutor/marker, who may handle many submissions, put your zID in all file names (e.g. **zID_Report.pdf** for the report and **zID_Notebook.ipynb** for the notebook).

Software: You are required to use OpenCV 3+ with Python 3+ and submit your code as a Jupyter notebook (see deliverables above and requirements below).

Objectives: This assignment is aimed at familiarisation with basic image processing methods. It also introduces you to common image processing and analysis tasks using OpenCV.

Learning Outcomes: After completing this assignment, you will have learned how to:

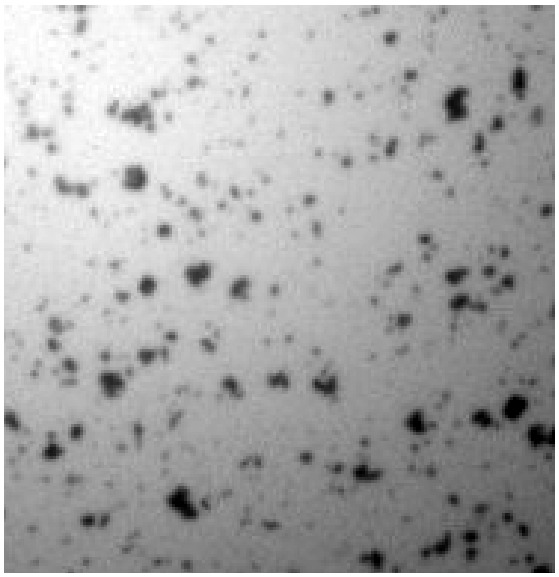
1. Open and read image files.
2. Display and write image files.
3. Perform mathematical operations on images.
4. Apply basic image filtering operations.

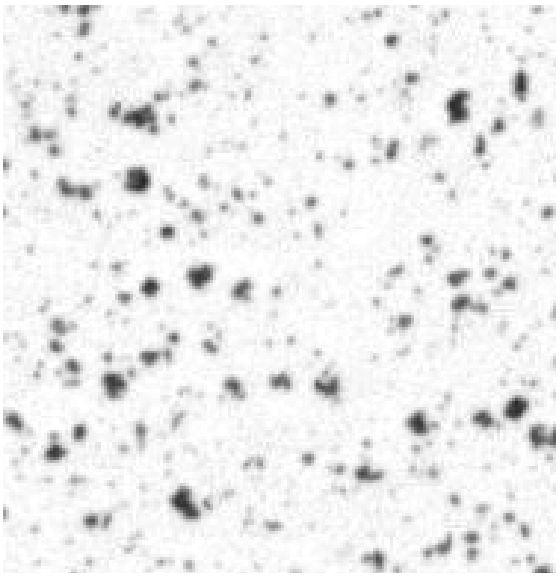
5. Perform image adjustment and restoration.

Description: Many computer vision applications use image processing techniques to suppress artifacts and enhance relevant features to prepare images for quantitative analysis.

The goal of this assignment is to create an image processing algorithm that can open a digital image, perform a sequence of pixel manipulation operations (step-by-step as listed under **Tasks** below), in order to remove background shading artifacts.

Below is an image (on the left) from a biological experiment in which the researchers wanted to measure several properties of the dark particles. But these measurements are impeded by the shading in the background. Thus, the shading needs to be removed (as on the right).





Tasks: This assignment consists of three tasks, described below. Each task depends on the previous one, so you need to do them in the given order.

<u>Task 1 (4 marks): Background Estimation</u>

Open the given image **Particles.png** which, like the example shown above, has a background shading pattern that needs to be removed. The first step to get rid of the shading is to make an accurate estimate of the background of the image.

Write an algorithm that performs the following two steps. First, it should create a new image, let us call it image A, with the same size (number of pixel rows and columns) as the input image, which we call I. Second, the algorithm should go through the pixels of I one by one, and for each pixel (x,y) it must find the **maximum** gray value in a neighbourhood centered around that pixel, and write that maximum gray value in the corresponding pixel location (x,y) in A. The resulting image A is called a **max-filtered** image of input image I.

The neighbourhood is a square of size N x N pixels, where N is a free parameter of the algorithm and typically has an odd value, with the pixel (x,y) under consideration being the center pixel. Manually try different values of N and mention in your report what is the smallest value of N that visually causes the dark particles in I to disappear altogether in image A. Also explain why this value of N, and not smaller values, causes the particles to disappear, and what is the effect of taking larger values than this smallest value.

The max-filtering causes the gray values in A to be higher than the actual background values in I, so a correction is needed. Extend your algorithm to create another image, which we call image B here, of the same size as I and A. Now let the algorithm go through the pixels of A one by one, and for each pixel (x,y) find the **minimum** gray value in an N x N neighbourhood centered around that pixel, and write that minimum gray value to (x,y) in B. The resulting image B is called a **min-filtered** image of the image A.

In your report, include image B computed from **Particles.png**.

Task 2 (2 marks): Background Subtraction

Now that your algorithm can estimate the background B of an image I, removing the shading artifacts from I can be done simply by subtracting B pixel by pixel from I, resulting in the output image O. Extend your algorithm to perform this subtraction.

Notice that the pixel values in B are equal to or larger than the corresponding pixel values in I, which means I – B will have negative values. Explain in your report what is theoretically the most negative value we can expect when subtracting two 8-bit/pixel images. In the pixelwise subtraction process, add this value (magnitude) to ensure all pixels in O are positive.

In your report, include image O computed from **Particles.png**.

Task 3 (4 marks): Algorithm Extensions

Open the given image **Cells.png** which, like **Particles.png**, has a background shading pattern that needs to be removed. There are three main differences between the two images: 1) the sizes of the images are different; 2) the sizes of the objects (cells versus particles) are different; 3) in **Cells.png** the objects are bright and the background is dark, whereas in **Particles.png** the objects are dark and the background is bright.

Ensure your algorithm can deal with input images of arbitrary size. Dealing with larger objects in images is a matter of changing the value of parameter N. But as you will see, your algorithm will not be able to remove the shading from **Cells.png**. To make your algorithm work for that image, you need to reverse the max-filtering and min-filtering.

Extend your algorithm with another free parameter (or rather a flag) named M. If the user sets M = 0, the algorithm should perform max-filtering (image I to A), then min-filtering (image A to B), then subtraction (O = I – B) with correction (see Task 2). And if the user sets M = 1, the algorithm should perform first min-filtering, then max-filtering, then subtraction.

In your report, explain why M = 0 (and not M = 1) works for **Particles.png**, and conversely why M = 1 (and not M = 0) works for **Cells.png**. Also mention what is a good value of N for **Cells.png** and why. And include images B and O computed from **Cells.png** in your report.

Coding Requirements

For all tasks, implement all mentioned operations yourself, and **do not use library functions** from OpenCV (such as `cv2.dilate()`, `cv2.erode()`, `cv2.morphologyEx()`), the Python Imaging Library (PIL) (such as `MinFilter()`, `MaxFilter()`), and Scikit-Image Library (such as `white_tophat()`, `black_tophat()`). 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 should be displayed in the notebook environment. In other words, 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