COMP9517: Computer Vision

2023 Term 2

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, Monday 19 June 2023, 18:00:00.

Deliverables: You must submit the following items:

1. A report explaining the approach you have taken in each of the tasks. 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 pages (A4), and must be single column, use 11 points Times New Roman or similar font, and have 2.5 cm margins.
2. Source code and results in the form of a Jupyter notebook (.ipynb) including all output (see coding requirements below).

Submission: Submit all deliverables in a single zip-file. 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).

Preliminaries: As stated in the course outline, 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 anyway 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. It would be most helpful to check (some of) these before coming to the first tutor consultation session (see next). 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 real-time computer vision. The library is cross-platform and licensed as free and open-source software under Apache License 2. It also supports model execution for machine/deep learning. 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 deliverables above and coding requirements below). In the first tutor consultation session, on Friday 9 June 2023 2-3 PM, your tutors will give a demo of the software to be used, and you can ask any questions you may have.

Objectives: This assignment is to get familiar 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 basic image processing operations on images.
4. Implement and apply various automatic thresholding techniques to images.

Description: In many computer vision applications, a crucial first step to allow quantitative analysis of an object (or region) of interest in images, is to identify which pixels belong to the object (the relevant pixels) and which belong to the background (the irrelevant pixels). This task is called binary image segmentation.

The simplest technique to perform binary image segmentation is intensity (gray-level) thresholding. 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 Week 1 lecture.

The goal of this assignment is to write image processing algorithms that can open a digital image and perform binary segmentation using various automatic thresholding techniques.

Tasks: This assignment consists of four tasks as described below.

<u>Task 1 (2.5 marks): Otsu Thresholding</u>

The first technique is called Otsu thresholding (named after the inventor). It defines the optimal threshold as the one that minimises the intra-class variance or, equivalently, maximises the inter-class variance of the pixel values in the two classes (object versus

background). See the lecture slides for more technical details.

Write an algorithm that can take an input image of any size and perform Otsu thresholding on it to produce a binary (segmented) image. Note that OpenCV or other Python libraries may already have existing functions for this, but you wouldn't learn anything from just using them. The task here is to write your own Otsu thresholding algorithm from scratch. In particular, write your own loops over the pixels to compute the pixel classes, standard deviations, and variances, and search for the optimal threshold value.

Task 2 (2.5 marks): Isodata Thresholding

The second technique is called isodata thresholding. It finds the optimal threshold by starting with a random threshold, computing the mean intensities (gray levels) of the two resulting classes of pixels, taking the mean of the two means as the new threshold, and repeating this process until convergence. See the lecture slides for more technical details.

Write an algorithm that can take an input image of any size and perform isodata thresholding on it. Here, the same comment applies as for Otsu thresholding: Do not use existing library functions that may be able to perform this task with one function call but write your own algorithm from scratch.

Task 3 (2.5 marks): Triangle Thresholding

The third technique is called triangle thresholding. It computes the optimal threshold from the intensity (gray-level) histogram of the image. More specifically, it calculates a straight line from the peak of the histogram to the extreme of the histogram (the highest gray level point) and finds the gray level for which the histogram deviates the most from the line. See the lecture slides for more technical details.

Write an algorithm that can take an input image of any size and perform triangle thresholding on it. Here again, the same comment applies as for the other tasks: Do not use existing library functions that may be able to perform this task with one function call but write your own algorithm from scratch. You are allowed to use an existing function to compute the histogram of the image.

Task 4 (2.5 marks): Compare Thresholding Techniques

Apply your Otsu, isodata, and triangle thresholding algorithms to the five images provided with this assignment and present the results in your report. Show the results in table form to allow easy visual comparison of all images. For example, per table row, show the input image, its histogram, and the three thresholding results.

In your report, discuss the differences in the results and provide explanations (based on the histograms or otherwise) why for some images one thresholding technique may work better

than others, while for other images it may be the other way around. Present some general guidelines for which thresholding techniques are best for which kinds of images.

Coding Requirements

For each task, implement the algorithm yourself, and do not use existing library functions (from OpenCV or any other packages) that can perform the task directly. Using these functions instead of your own implementation will result in deduction of points.

In future labs and in the group project you may use existing library functions, but in this assignment (like in the first lab) the goal is to learn how basic image processing operations work at the pixel level and get experience implementing them yourself. Of course, you may use existing functions to verify the results of your own implementations.

For this assignment (same as for the labs), 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: 5 June 2023