COMP9517: Computer Vision

2025 T2 Lab 4 Specification

Maximum Marks Achievable: 2.5

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

The lab files must be submitted online.

Instructions for submission will be posted closer to the deadline.

Deadline for submission is Week 7, Monday 14 July 2025, 18:00:00 AET.

Objective: This lab revisits important concepts covered in the Week 5 lectures and aims to make you familiar with implementing specific algorithms.

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 tutor consultation session this week and next, you can ask any questions you may have about this lab.

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 file) that includes all outputs (see coding requirements at the end) by the above deadline. The submission link will be announced in due time.

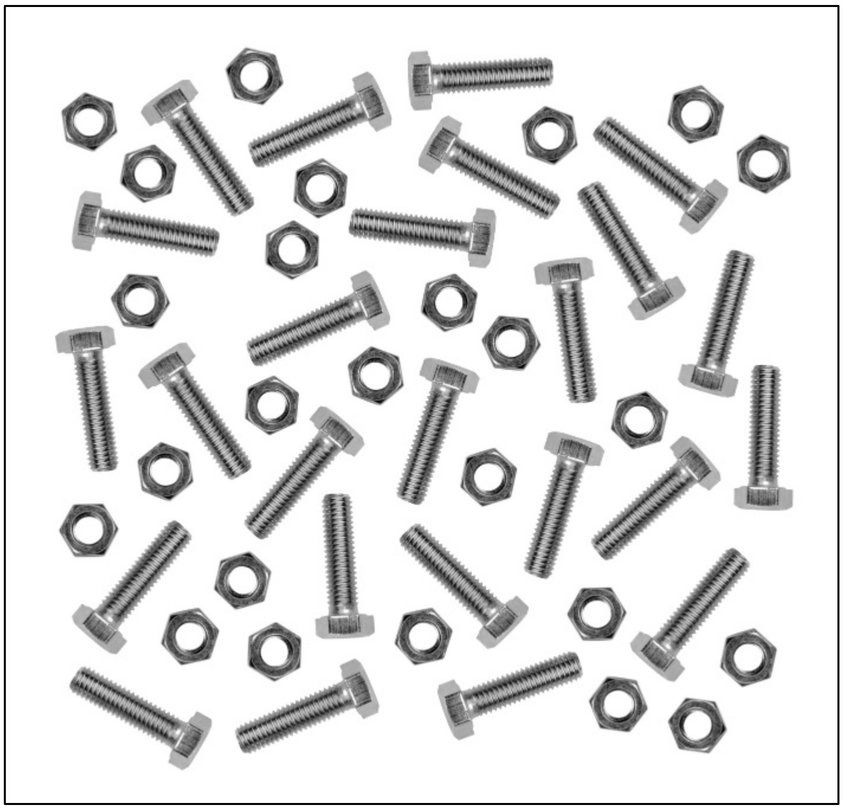
Task (2.5 marks)

In many computer vision applications, image segmentation is a crucial first step towards quantitative image analysis. As discussed in the lectures, there are many possible image segmentation methods, ranging from very simple to very complicated.

Despite its simplicity and limitations, intensity thresholding is still a very popular image segmentation method. This is because imperfections in the segmentation results can often be filtered out by postprocessing using binary morphological operators.

The goal of this lab is to write an algorithm that performs image segmentation and subsequent analysis based on just intensity thresholding, binary morphological processing, and applying some basic postprocessing rules to the segmented objects.

The scenario is as follows. A manufacturer of nuts and bolts wants to develop a quality control pipeline that fully automatically checks whether each batch contains the right number of nuts and bolts before packaging. Each batch must contain exactly N nuts and N bolts, where N is a number set by the human operator of the pipeline. In our case, N = 25. To this end, each batch is imaged (see below for an example) and checked automatically.


#Nuts = 25, #Bolts = 25, All good

You are responsible for developing the algorithm that performs the task. More specifically, for each batch image, the algorithm must count the number of nuts and the number of bolts, report both numbers, and indicate whether the batch needs correction and how.

For example, if the number of nuts and bolts are both N, no further action is needed, and the algorithm must indicate "All good". But if the number of nuts and/or bolts is less than N, the algorithm must indicate "Add X nuts" and/or "Add Y bolts", where X and Y are the numbers needed to get to N. Conversely, if the number of nuts and/or bolts is larger than N, the algorithm must indicate "Remove X nuts" and/or "Remove Y bolts" to get to N.

In your code, specify N at the beginning and then use it throughout the algorithm, so that it can be easily set by the user as needed. Also, if you use any operations with (hyper)parameters, define these at the beginning of your code, too.

Your algorithm must work fully automatically without any user input or interaction during operation. It must be able to handle all images provided in this lab (see WebCMS3) with the same (hyper)parameter values (not different values per individual image).

Hints: Convert the input image to grayscale (if needed), apply intensity thresholding (using either an automatically calculated or a manually fixed threshold), use binary morphology operators to fill any holes or filter out noise pixels in the segmentation, use the connected components algorithm or something similar to identify and count the separate regions, use a size threshold to determine whether a region is a nut or a bolt, and implement some rules to calculate and print out what action is needed (as described above).

Coding Requirements
Make sure that in your Jupyter notebook, the input images are readable from the location specified as an argument, and all 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.

If your algorithm has any (hyper)parameters, their values must be specified at the beginning of your code, so that users can easily change and experiment with these parameters.

Released: 30 June 2025