COMP9517: Computer Vision

2023 T2 Lab 3 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 5, Friday 30 June 2023, 18:00:00.**

Objective: This lab revisits important concepts covered in the lectures of Week 4 and aims to make you familiar with implementing specific algorithms.

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

Submission: The tasks are assessable after the lab. Submit your source code as a Jupyter notebook (.ipynb) which includes all output (see coding requirements below) by the above deadline. The submission link will be announced in due time.

Image Segmentation

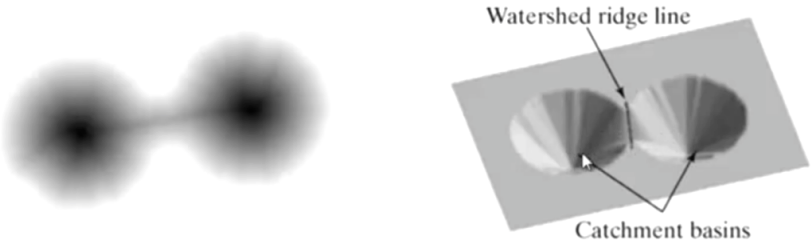
The goal of image segmentation is to assign a label to each pixel in an image, indicating whether it belongs to an object (and which object) or the background. It is one of the key research topics in computer vision and there are many different approaches: interactive segmentation, semantic segmentation, instance segmentation, and more.

In this lab the MeanShift clustering algorithm and the Watershed algorithm will be used to perform unsupervised image segmentation.

MeanShift is a clustering algorithm that assigns pixels to clusters by iteratively shifting points towards the modes in the feature space, where a mode is a position with the locally highest number of data points (highest density). A visualisation can be seen here.

Watershed is a transformation that aims to segment the regions of interest in a grayscale image. This method is particularly useful when two regions of interest are close to each other (that is, their edges touch). It treats the image as a topographic map, with the intensity of each pixel representing the height. For instance, dark areas are considered to be 'lower' and act as

troughs, whereas bright areas are ‘higher’ and act as hills or a mountain ridge.

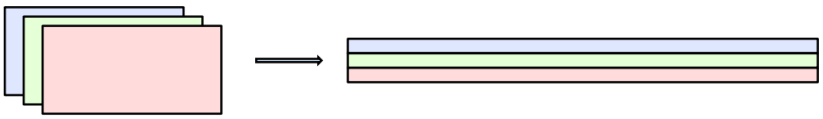


Visualising the Watershed: The left image can be topographically represented as the image on the right. Adopted from Agarwal 2015.

Task 1 (0.5 mark): Use the MeanShift algorithm for image segmentation.

Hint: Use MeanShift clustering from Scikit-learn.

Step 1. Once you have read an image into Numpy arrays, extract each colour channel (R, G, B) so you can use each as a variable for classification. To do this you will need to convert the colour matrices into a flattened vector as depicted below.



Step 2. Then you can use the new flattened colour sample matrix (e.g. 10,000 x 3 if your original image was 100 x 100) as your variable for classification.

Step 3. Use the MeanShift `fit_predict()` function to perform a clustering and save the cluster labels, which we want to observe.

Show the segmented image for each of the provided sample images. Hint: To further improve the segmentation results, you may use binary morphological operations (see lecture slides).

Task 2 (1 mark): Use Watershed transformation for image segmentation.

Hint: Use Watershed segmentation from Scikit-image.

Step 1. Convert the image to grayscale. Then use an appropriate threshold value to convert the grayscale image into a binary image (objects versus background). Hint: Use built-in functions for thresholding.

Step 2. Calculate the distance transform of the binary image. Hint: Visualising this step may help you understand how the algorithm works. Plot the result of the distance transform to see what is happening under the hood.

Step 3. Generate the Watershed markers by finding the ‘pixel clusters’ furthest away from the background. This can be somewhat confusing, so make sure to check the example code on the page linked above. Hint: Experiment with different local search region sizes in this step and

the threshold value in Step 1 above for good segmentation results.

Step 4. Perform Watershed on the image. This is the part where the image is 'flooded' and the 'water level' increases in the 'catchment basins' based on the markers found in Step 3.

Show the segmented image for each of the provided sample images. Hint: To further improve the segmentation results, you may use binary morphological operations (see lecture slides).

Task 3 (1 mark): Compare MeanShift and Watershed segmentation results.

Step 1. Write an algorithm that counts the number of objects in the segmented image. Hint: The number of objects is the number of connected components. You may use existing library functions or write your own to uniquely label and count the connected components.

Step 2. Apply the algorithm from Step 1 for the MeanShift segmentation and the Watershed segmentation of each image and record the results in a table in your notebook. The precise table format in your notebook does not matter, as long as it is easily readable, like this:

Image	**#Objects MeanShift**	**#Objects Watershed**
Balloons		
Balls		
Brains		

Step 3. Based on these results, briefly discuss in your notebook which method performs best for which image, and what could be the explanation for this.

Coding Requirements and Suggestions

In your Jupyter notebook, the input images 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 need to execute the notebook again to see the results.

References

Segmentation using Watershed Algorithm in Matlab

Meanshift Algorithm for the Rest of Us (Python)

Released: 23 June 2023