Worksheet 5
Segmentation

---

1. Segmentation by Interactive Thresholding.

Chapter 1.6

As mentioned in class, if the image histogram is bi-modal then we can easily segment the image.

(a) Run the script "**Thresh1**" and look at the histogram of the image provided. Now choose what you think is a suitable threshold and see if this correctly segments the image. There should be (for a perfect segmentation, but that may not be possible)

(i) No isolated white pixels or small groups on the background
(ii) No holes in the objects.

(b) Repeat with different thresholds until you find the optimal choice.

(c) Now repeat with an obviously *incorrect* threshold and make sure you understand what you see.

---

2. Segmentation using the Otsu Method

Otsu's method works like the above example, but it is automatic since the threshold is computed 'optimally' for each image histogram.

(a) Run the script "**Thresh2**" which applies Otsu's method (choosing the threshold automatically).

(b) Compare the threshold found with your choice above.

(c) Look at the "Separation Measure" which can be between 0.0 (no separation) and 1.0 (perfect separation). Is it reasonable?

(d) Perhaps have a go on some other images. Perhaps not.

3. Segmenting Noisy Images – The Wrong Way

This activity demonstrates the futility of trying to segment a noisy image by thresholding.

(a) Run the script "**Thresh2a**" and note the form of the histogram which is not bi-modal. So how can we choose the threshold? With difficulty. Choose a few different thresholds and convince yourself this approach cannot work.

4. Segmenting Noisy Images – Pre-processing using Smoothing

Here we shall first smooth the noisy image using a mean filter to give a bimodal histogram. Then we shall use Otsu's method to find the optimal threshold and then we use this to segment.

(a) Run the script "**Thresh3**" and start with a kernel size of 3.

(b) Evaluate the segmentation result by (i) eyeballing the segmented image, looking for isolated white or black pixels, (ii) looking at the Separability Measure

(c) Repeat for increasing kernel sizes. What is the effect of increasing the kernel size on the segmentation of the images, as judged by eyeballing the segmented image?

(d) Does it really work?

5. Putting it all together – Labelling and Measuring the Segmented Objects

This uses the same approach as the last example. We have added extra function calls to label the objects in the segmented image and to measure their areas (number of

pixels in each object). Count the objects in the initial image, I make it 10.

(a) Run the script "**Thresh4**", increasing the smoothing kernel sizes starting with 3.

(b) Note down the number of objects segmented and keep an eye on their sizes as you increase the kernel size (You may want to consider just one object). You should find two 'disaster' segmentations.

(c) Can you see a pattern in the object size changing as you increase the kernel size? Can you explain this?

(d) Can you think of a way of automating setting the kernel size. Hint – count the objects.

You might like to try some other images, here's some suggestions.

Remember the above scripts **corrupt the image by adding Gaussian noise** of a huge amount (0.45). You may want to reduce this to something more realistic, like 0.05. Or remove it altogether.

(a) "circlesBrightDark.png".

(b) "eight.tif"

(c) "moon.tif"

(d) Your own grey-scale images.