7060 Image Sensors & Processing

Assignment 3

May 2019

(Assignment Value: 13%)

Overview

This assignment is intended to give you some hands-on experience with morphological filters, edge detection and line finding, and to review some of the morphological techniques covered in recent lectures.

IMPORTANT:

Do not use routines from the image processing toolbox, or parts thereof (unless approved in class/tutorials) for Exercises 3A to 3C. You may of course use some of these toolbox functions to **test** your code so long as they are not present in your submitted solutions.

Assignment Submission

Assignments are to be submitted via MyUni as a ZIP archive file containing the key MATLAB functions and a Word or PDF document summarising your results with comments on the performance of the processing steps employed.

Source Materials

Source code, test functions and example imagery for this assignment are located on the MyUni website (https://www.myuni.adelaide.edu.au).

Exercise 3A – (4%) – Morphological HitorMiss

Simplified versions of the convex hull hand thinning operations described in class can be constructed from multiple evaluations of the so-called Hit or Miss morphological matching function using 3x3 templates.

Modify the supplied function *hitormiss_3x3.m* such that when given a binary image and a 3x3 matching template the function returns those points in the image which match the template. Here the 3x3 templates are represented as arrays of -1,0 and 1's, for example:

$$\begin{bmatrix} 0 & 1 & 1 \\ 0 & -1 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Where 1 = "set pixel", -1 = "unset pixel" and 0 = "don't care" (note: this notation is slightly different to that used in class but matches that used in MATLAB). In this example $\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$ would be the template for set pixels and $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$ for the unset pixels. so a binary pattern such as $\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$ would be considered a match.

Uising the above 3x3 template against the binary image:

```
1 1 1 1 1 1
0 0 0 1 1 1
0 0 0 1 1 1
1 1 1 1 1 1
```

there would be a match at the highlighted position (2nd row, 3rd column).

Morphological HitorMiss is defined as follows:

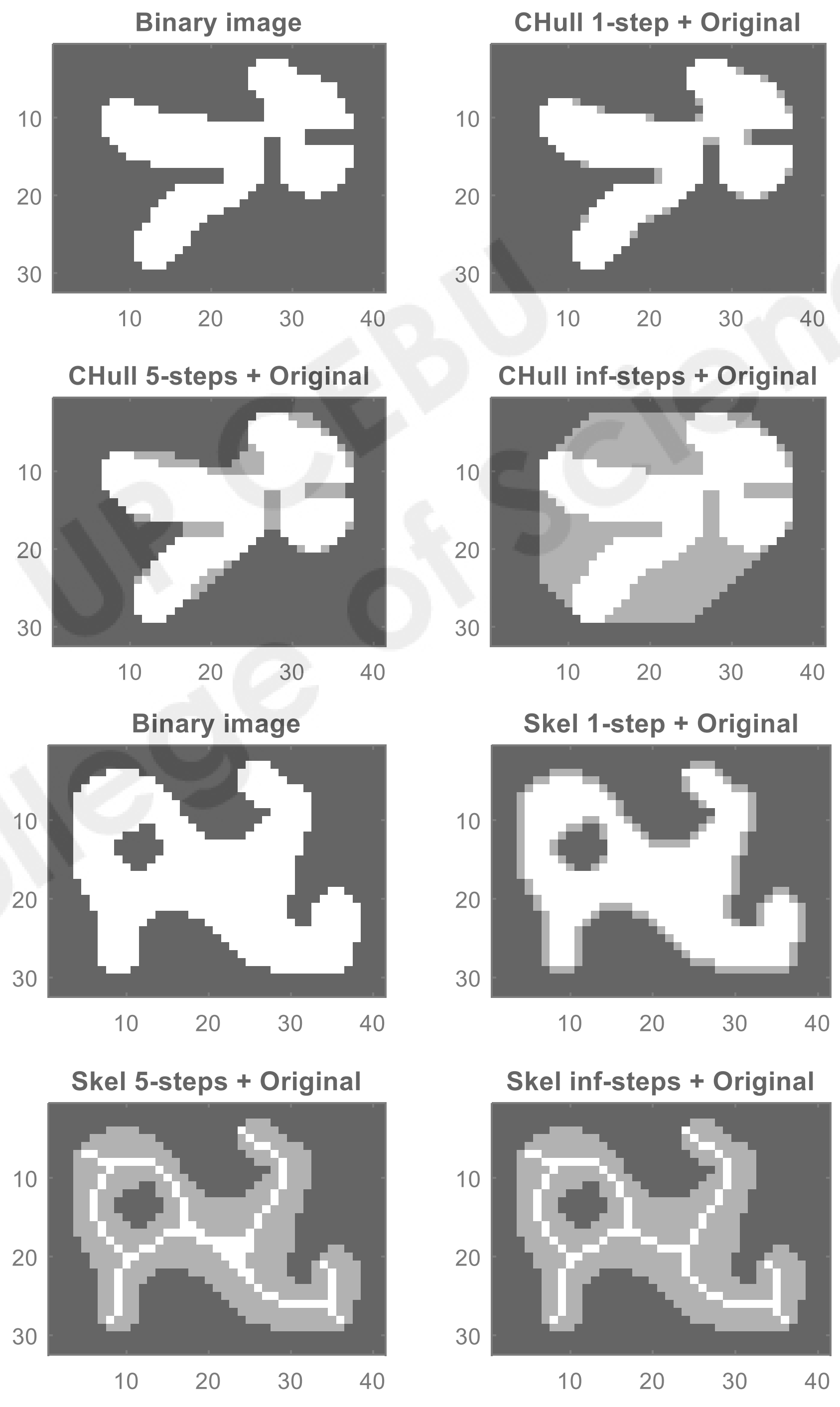
$$\text{hitormiss}(A, B_1, B_2) = \left(erosion(A, B_1)\right) \cap \left(erosion(A^{\subset}, B_2)\right)$$

where $A^{\subset}$ represents the compliment (inversion) of the binary image A, and B_1 and B_2 are the structure elements (patterns) for "set" (1) and "unset" (0) pixels respectively.

In order to implement this you will first need to implement the function *erosion_3x3* which takes an image and a 3x3 structure element and performs morphological erosion. This function is located inside *hitormiss_3x3.m*.

Two special test case example of this function which implement very simple forms of the morphological convex hull and skeletonisation processes shown in class have been provided to you (*chull_3x3.m* and *skel_3x3.m* – these are called from *chull_test.m* and *skel_test.m*). Make sure your HitorMiss solution generate results identical to those shown overleaf.

Remember to write up your results in the assignment report. Do NOT use any methods from the image processing toolbox in your solution.



Above: *Example output from* ***chull_test.m*** *and* ***skel_test.m*** *based on a working solution for hitormiss_3x3*

Exercise 3B – (4%) – Laplacian of Gaussians Edge Detection

You have been tasked implement an edge detector as part of a robotic vision system to analyse machine parts. Modify the supplied function *log_edge.m* so that it implements a Laplacian of Gaussians edge detector as described in class. The function is called as follows:

E=log_edge(I,opsize)

Here I is the input image and opsize is the size of the LoG convolution filter to be used and (ie. opsize by opsize pixels). The output image E is a binary image where the background and edge points are denoted by 0 and 1 respectively. The expression for the LoG convolution filter from the lecture notes is:

$$\nabla^2 G = -\left(\frac{\left(x^2+y^2\right)-2\sigma^2}{\sigma^4}\right)e^{-(x^2+y^2)/2\sigma^2}$$

Here sigma is proportional to `floor((opsize-1)/3)/2`. The detected edge points should be thresholded using a suitable threshold value. The one given in the lectures was 75% of the mean absolute of the filtered image, that is

$$t = \frac{3}{4}\text{mean}\left|(\nabla^2 G)\otimes I\right|$$

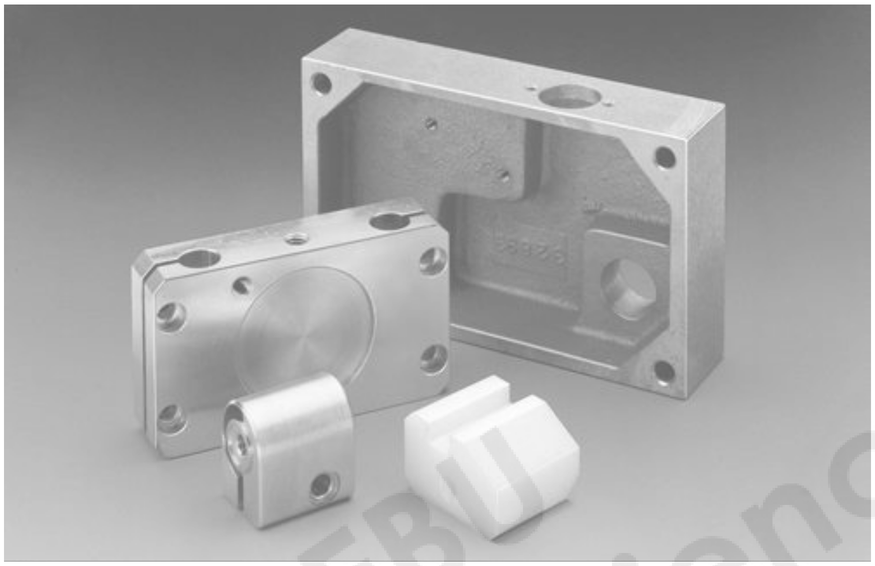
IMPLEMENTATION NOTE: For filters larger than 12x12 pixels the above expression typically provides a good threshold. However, for smaller filter sizes this threshold tends to be too small resulting in additional spurious edges. As a result, a correction term has been provided for you in the code to partly deal with this (see variable *threshK* in *log_edge.m*).

In the supplied code the Laplacian of Gaussians $(\nabla^2 G)$ filter has already been constructed. From here you will have to:

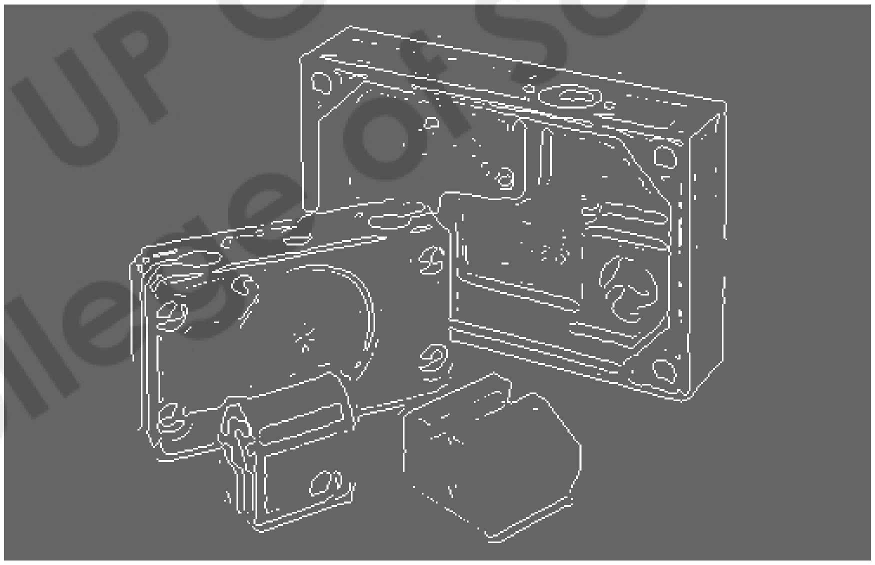
1. Convolve $(\nabla^2 G)$ with I
2. Compute the threshold *t* shown above (and multiply by the given scale factor *threshK* in the supplied code)
3. For each pixel location where $(\nabla^2 G)\otimes I$ is positive:
    a. Test to see of the value of $(\nabla^2 G)\otimes I$ immediately to the left is negative. If so then a zero crossing is present. If the sum of the absolute of the two values is greater than *t* then this pixel is an edge pixel.
    b. repeat step (a) for pixels to the right, above and below the current one.

The edge image, the gradient and filter gradient image should be returned in variables E, F and G respectively.

A typical output from the edge detector should look similar (but not necessarily exactly the same) to the example given on the next page:



LoG Result 13x13 filter



ABOVE: An example LoG edge detection. NOTE: Your own results may differ from this depending on the size of the filter used and your threshold calculation.

Test your solution out on examples from the provided test scripts (see *log_test.m*) and images provided as well as some example images of your own choosing. Experiment with different sized filters and write up your observations in the assignment report.

Exercise 3C: Hough Transform Line Estimates (3%)

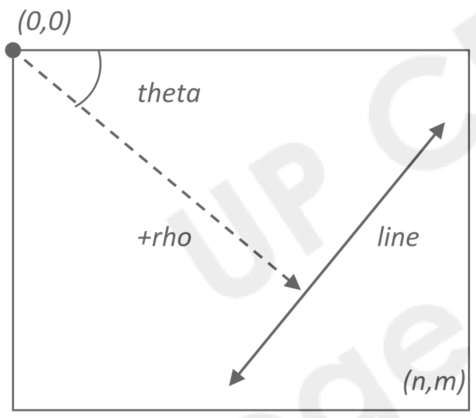
In the simple case considered here, the Hough parameterisation of a line is as follows:

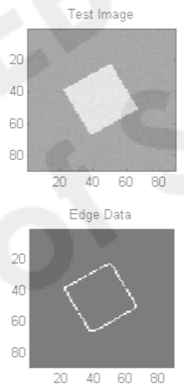
$$\rho = x\cos\vartheta + y\sin\vartheta$$

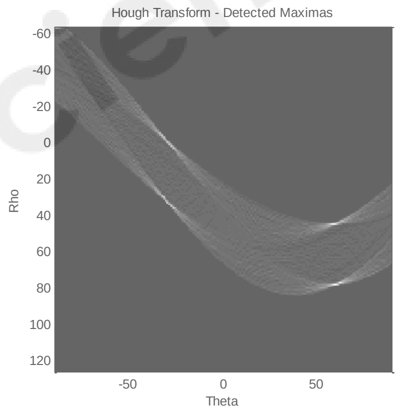
where the coordinate system is relative to the top-left (0,0) pixel of the image. A function *hough_transform.m* has been provided to compute a simple estimate of the Hough transform. For example:

H = hough_transform(E,[-64:126],[-90:90])

will produce a Hough estimate H of the provided edge image E for a range of rho values of -64 to 126 and -90 to 90 for theta. An example result is given below, including a visual illustration of the coordinate system. The peak at approximately rho=0, theta=30 corresponds to the bottom left edge of the example square.







Modify the function *extract_lines.m* so that given a Hough transform H, ranges for *rho* and *theta,* the function returns the 'best' N line estimates detected in the Hough transform. For example:

rt = extract_lines(H, E,[-64:126],[-90:90],4)

should return *rho* and *theta* estimates of the strongest (highest) 4 peaks in the transform data *H* for the box example. The value ***rt*** is a Nx2 array of *rho* and *theta* values (N rows by 2 columns that is).

A simple method for achieving this is as follows:

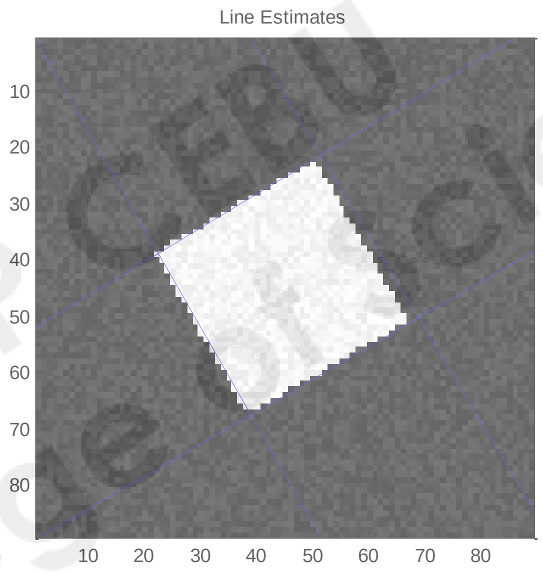
```
for k=1 to number of lines required
        1. Find the brightest 'peak' in the Hough transform
        2. Record the rho, theta position of the 'peak'
        3. Delete (supress) the region around the 'peak' in the Hough
           transform (so that it won't be picked up again as a peak on the
           next iteration).
End
```

Typically, the presence of a line will appear as an elongated blob in the Hough transform, possibly with several very nearby maxima in the transform estimate. As a result, a 'peak' may not always appear as a simple 'local maxima' but as a small local cluster. The deletion of the peak may therefore actually require the deletion of several points (eg. a 5x5 region) in a neighbourhood around the peak to prevent multiple detections of the same line.

Once you have implemented this, to check your results, use the provided function *display_lines.m.* For example, if ***I*** was the original image used to construct ***H*** and ***rt*** above, then:

display_lines(I,rt)

should display the image *I* along with the 4 strongest line segments. An example is given below:



Above: Illustrations of the above steps (from the lecture notes) – depending on the version of MATLAB you use you may see different colored lines in your plots.

Test your code using the *extractlines_test.m* script provided and on other example imagery of your own choosing. Write up your results in your assignment report.

Q: How well does the line finder work when the image contains many line structures?

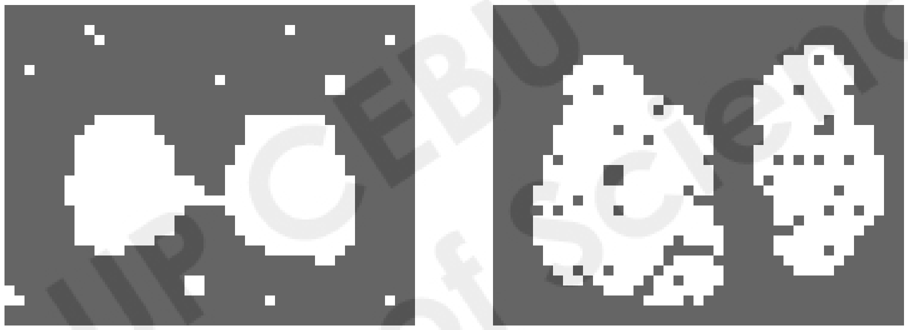
Exercise 3D – (2%) – Written Questions

Question 3C(1) (0.5%):

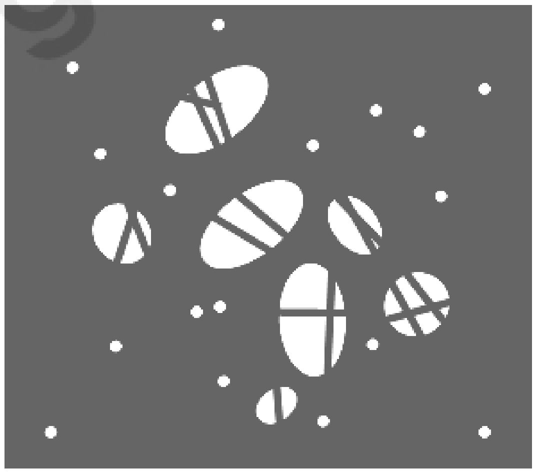
Explain how the CFAR detector works (use a diagram etc if required) and explain under what conditions does it make sense to use a local thresholding scheme such as this as opposed to a global one (such as iso-thresh).

Question 3C(2) (1.5%): A binary image contains a number of thresholded objects including several which have been accidentally fragmented during segmentation (See below). Some smaller regions are also present.

A) (0.5%) Using the morphological filters discussed in class, suggest how you would clean up the following two binary images containing 2 shapes each:



B) (0.5%) Suggest how the small regions can be removed without adversely affecting the line-like structures of the larger regions using a combination of morphological filters and some simple logic.



Above: *A thresholded image containing fragmented large regions and smaller isolated regions (here the black and white regions represent 0 and 1respectively).*

C) (0.5%) Conversely, what would we do if we wanted to remove the fragmentation of the large regions?

(NOTE: in all cases explain your reasoning in your answers)