

CMP3108M Image Processing Assessment Item 1

1 Task 1 – Pre-processing Image

Task 1 consists of multiple operations of pre-processing the image. Initially, the enhancement technique used was histogram equalisation, where the intensity values are transformed so that the number of values in each bin in the histogram is approximately the same. This method was not very effective as the shadows around the edges of the image proved difficult, this made some items around the edges hard to see clearly. When binarized it could not detect some objects, so it was not used. The function 'imadjust' was used; it saturates the bottom 1% and top 1% of all pixel values, making the washers and screws darker than the background (except one washer in the bottom right of the image), so they were distinguishable when binarizing the image. Another function 'imsharpen' was used with the parameters of 'radius = 1' and 'amount=1', this is where the contrast along the edges where different colours meet, such as edges of objects, is increased, allowing for more distinct edges in the image which makes it better for edge detection operators in the later tasks, but can introduce image noise if the amount is too high. Even with the 'amount' parameter set to 1, slight image noise is introduced when binarizing the image (Figure 5) but can be removed with morphological operations.

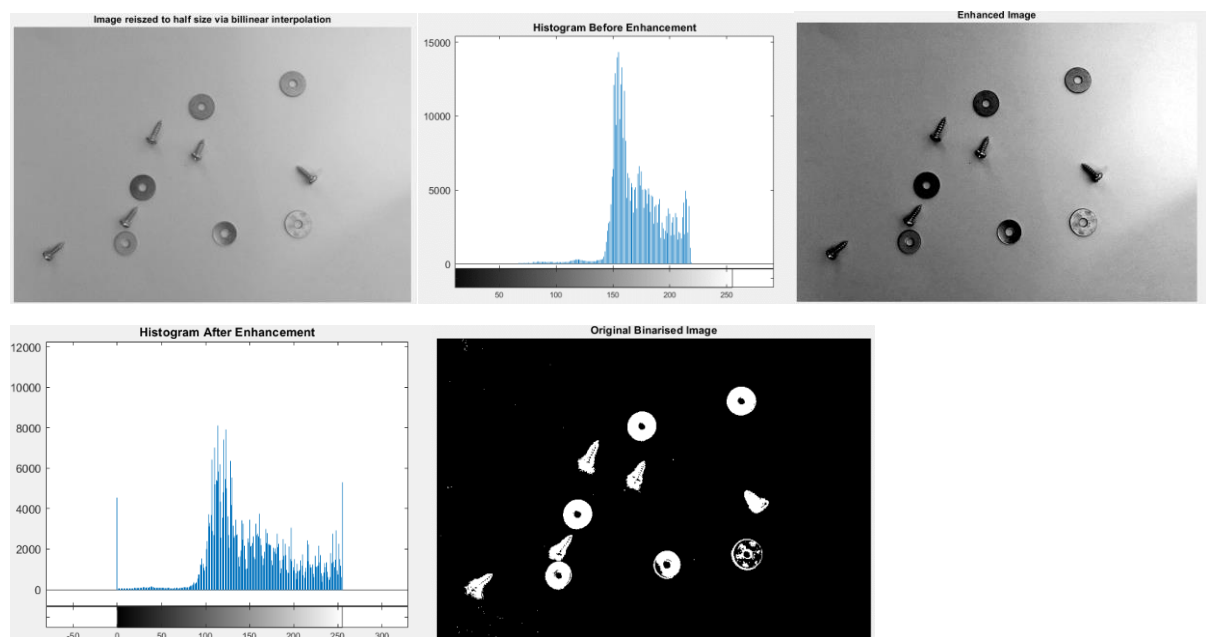


Figure 1-5 – Resized image (Top Left), Histogram before enhancement (Top Middle), Enhanced image (Top Right), Histogram after enhancement (Bottom Left) and Binarized Image (Bottom Middle).

2 Task 2 – Edge Detection

The 'sobel' algorithm was used for edge detection of the objects. Several different algorithms such as 'canny', 'prewitt' and 'roberts' were also tested. The 'canny' edge detector picked up too much visual noise from the sharpening enhancement technique and the lighting of the image, so it performed poorly. The enhancement technique could have been changed to

accommodate this algorithm to minimise the effect of shadows and lighting. The ‘roberts’ algorithm was used and performed sub-par, not detecting some edges of the short screws and washers, leaving gaps between objects, so it was not used. The ‘sobel’ and ‘prewitt’ algorithms both performed similarly by adequately detecting the edges of all washers and short screws, there was minimal difference between the two algorithms thus ‘sobel’ was chosen. Some noise was detected in both algorithms due to the sharpening in the preprocessing stage, but it was removed later with morphological operations.

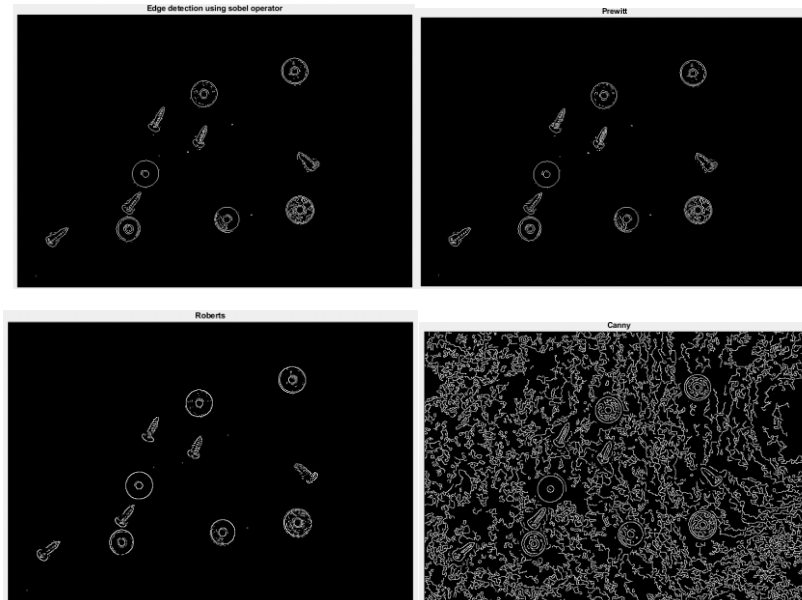


Figure 6-9 – Sobel edge detector (Top Left), Prewitt edge detector (Top Right), Robert's edge detector (Bottom Left) and Canny edge detector (Bottom Right).

3 Task 3 – Simple Segmentation

The edge-detected image using the ‘sobel’ operator was used to perform edge-based segmentation of the washers and screws. The first step was to create structuring elements; these are matrices used to define the neighbourhood used to probe or analyse an input image (Gonzalez and Woods, 2008). Two disk structuring elements were made with radii 1 and 2 using the ‘strel’ function. The image was dilated (grows/thickens objects) using the ‘imdilate’ function with the disk structuring element with a radius of 1, this was to ensure that all edges were filled in but not hugely dilated that the objects started touching and to minimise the increase in the size of the noise (Figure 10). Next, with the edges now completed, the holes within large regions of the washers and screws can be filled with the function ‘imfill’ with the ‘holes’ parameter (Figure 11). The image now contains filled-in regions, but the edges of objects are somewhat jagged and there are still some unconnected pixels surrounding some objects, so the disk structuring element with radius 2 was used for erosion. This is where the objects are shrunk or thinned, attempting to remove the features of jagged edges and noise, most of which was removed except one pixel which was left (Figure 12). Finally, the function ‘bwareaopen’ was used, which removes all connected regions with fewer than X pixels which was defined as 4, thus the singular pixel was removed, and the objects were properly segmented as shown in Figure 13.

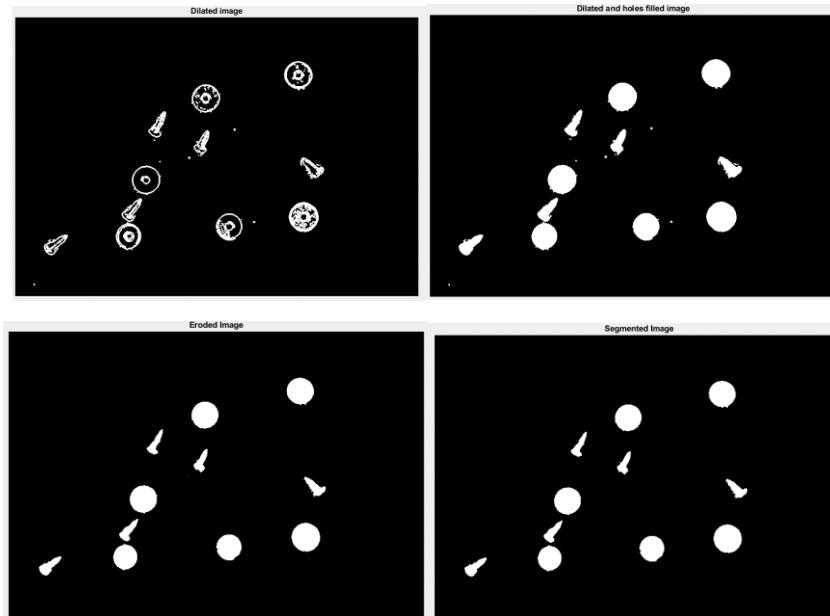


Figure 10-13 – Dilated Image (Top Left), Holes Filled (Top Right), Eroded Image (Bottom Left), Final Segmented Image (Bottom Right).

4 Task 4 – Object Recognition

For object recognition, the segmented image was passed through a function 'bwlabel' which creates a matrix of labels for the components in the image that are connected with corners or edges. The labels are then passed to another function 'regionprops' which gets the properties such as perimeter, area, or circularity of the different labelled regions in the label matrix and creates a table, allowing us to set thresholds for certain metrics so objects can be classified. Each region in the table was looped through and if the circularity was greater than the set threshold of 0.8, the label would be set to 1, a washer, or else it was set to 2, a short screw. Finally, the objects were coloured using the 'label2rgb' method.

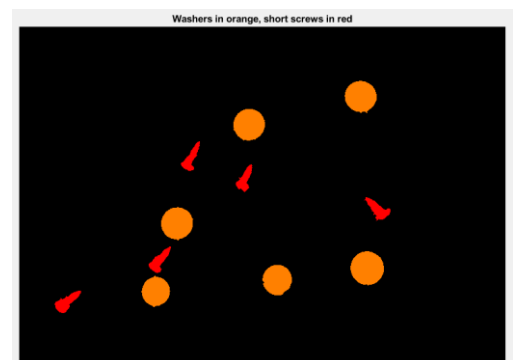


Figure 14 - Object Recognition.

5 Robust Method

Firstly, the image was resized using bilinear interpolation and sharpened with a radius of 2 and an amount of 9, allowing the edges of each object to be more defined, this did add noise to the image. The image was binarized and dilated using a square structuring element of size 3, this thickened the edges but also increased the size of the noise so the 'bwareaopen' function was used again to remove connected objects less than 60 pixels. The 'bwmorph' function was used to thicken object edges by adding pixels until they were connected, with the majority parameter also used to fill some unconnected edges of objects. These functions did not have much of an effect on this image specifically but did help segment other images. The holes were filled and then the image was eroded using a rectangular structuring element to decrease some spiked edges around objects. The median filter was applied to smooth

object edges while keeping their shape. Finally, the same process was used as above where two thresholds were set, with the circularity and area of the regions calculated using 'regionprops', each region was labelled and then coloured using 'label2rgb'. The resulting image is shown in Figure 20, where two short screws are misclassified.

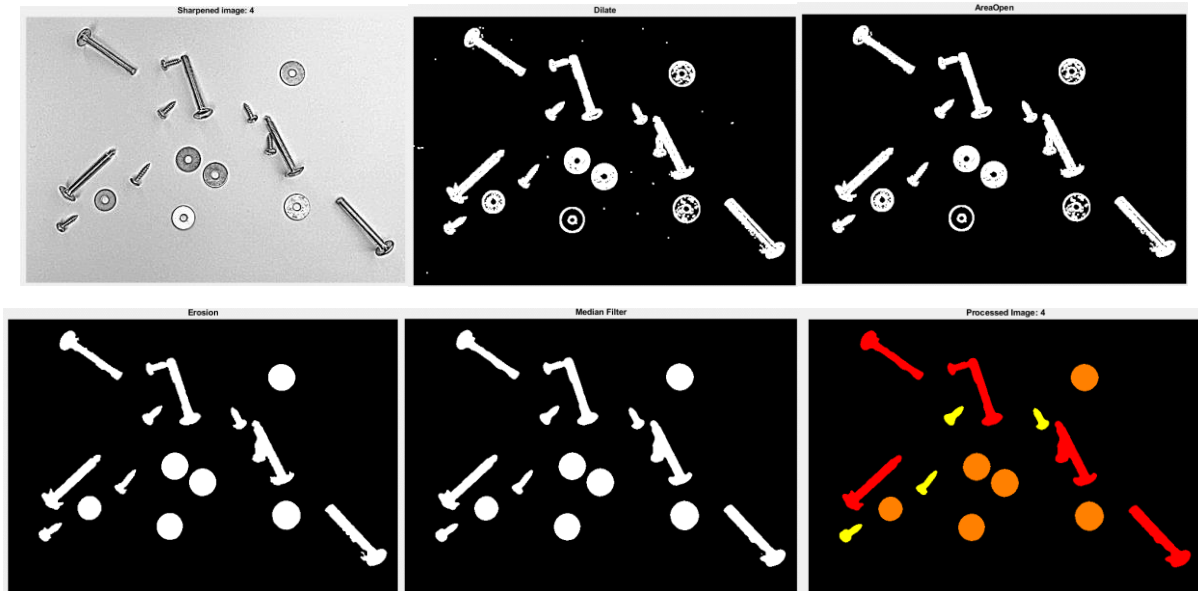


Figure 15-20 – Sharpened Image (Top Left), Dilated and Binarized Image (Top Middle), Image with noise removed (Top Right), Holes filled then eroded image (Bottom Left), Median Filter (Bottom Middle, Final Image (Bottom Right).

6 Task 6 - Performance Evaluation

The calculations for the metrics precision and recall are as shown, the dice score was calculated using a built-in MATLAB function 'dice'. The True Positives (TP) were calculated by checking the label for each region and if it matched with the ground truth image. The True Negatives (TN) were calculated by checking if the ground truth matched the predicted 0 labels for the processed image which was the background. False Positives (FP) were calculated by checking if the ground truth is not equal to the processed image label and the processed image is not equal to 0. False Negatives (FN) were calculated by checking if the ground truth was not equal to the processed image label and if the ground truth was not equal to 0. Then, the following equations were used to calculate the precision and recall of each image. The results of each image are shown below.

Image	Dice Score	Precision	Recall
IMG_01.jpg	0.9503	0.9582	0.9369
IMG_02.jpg	0.9539	0.9597	0.9596
IMG_03.jpg	0.9585	0.9597	0.9677
IMG_04.jpg	0.9238	0.9357	0.8864
IMG_05.jpg	0.9308	0.9447	0.9208
IMG_06.jpg	0.9180	0.9506	0.9098
IMG_07.jpg	0.9435	0.9543	0.9354
IMG_08.jpg	0.9524	0.9573	0.9595
IMG_09.jpg	0.9458	0.9526	0.9487
IMG_10.jpg	0.9567	0.9634	0.9478
Mean	0.9434	0.9536	0.9372
S.D	0.0143	0.0083	0.0253

$$Precision = \frac{TP}{FP + TP}$$


$$Recall = \frac{TP}{FN + TP}$$

7 References

MathWorks. (2023) *imsharpen Function - MATLAB & Simulink*, Available at: <https://uk.mathworks.com/help/images/ref/imsharpen.html> (Accessed: 15/01/2024).

R.C. Gonzalez and R.E. Woods. (2008) *Digital Image Processing*. Third Edition. London: Pearson Prentice Hall.

8 Appendix A



```

1  clear; close all;
2
3  % Task 1: Pre-processing -----
4  % Step-1: Load input image
5  I = imread('Assignment_Input\IMG_01.jpg');
6  figure, imshow(I)
7
8  % Step-2: Covert image to grayscale
9  I_gray = rgb2gray(I);
10 figure, imshow(I_gray), title("Image converted to greyscale");
11
12 % Step-3: Rescale image
13 J = imresize(I_gray, 0.5, 'bilinear');
14 figure, imshow(J)
15 title('Image reisized to half size via billinear interpolation');
16 % Step-4: Produce histogram before enhancing
17 figure, imhist(J, 256), title("Histogram Before Enhancement");
18 % Step-5: Enhance image before binarisation
19 enhancedJ = imadjust(J);
20 enhancedJ = imsharpen(enhancedJ, radius=1, amount=1);
21 figure , imshow(enhancedJ);
22 title('Enhanced Image')
23 % Step-6: Histogram after enhancement
24 figure, imhist(enhancedJ, 256), title('Histogram After Enhancement');
25 % Step-7: Image Binarisation
26 BW = imbinarize(enhancedJ, 'adaptive', 'ForegroundPolarity', 'dark', 'Sensitivity', 0.47);
27 BW = ~BW;
28 figure, imshow(BW)
29 title('Original Binarised Image');
30
31 % Task 2: Edge detection -----
32 edgeImg = edge(enhancedJ, "sobel");
33 figure, imshow(edgeImg);

```

Figure A.1 - Task 1 to 4 Code - Part 1/3

```

35 % Task 3: Simple segmentation -----
36 %Structuring element for morphological operations
37 se = strel('disk', 1);
38 se2 = strel('disk', 2);
39
40 %Dilate and fill holes
41 I_Dilated = imdilate(edgeImg, se);
42 I_Filled = imfill(I_Dilated, 'holes');
43 figure, imshow(I_Filled);
44 title('Dilated and holes filled image')
45
46 %Erode image to remove some noise around objects
47 I_Eroded = imerode(I_Filled, se2);
48 figure, imshow(I_Eroded);
49 title('Eroded Image')
50
51 %Removing objects less than 4 pixels
52 BW2 = bwareaopen(I_Eroded, 4);
53 figure, imshow(BW2)
54 title('Segmented Image')
55
56 % Task 4: Object Recognition -----
57
58 %Getting labels from image
59 [L, num] = bwlabel(BW2);
60 labels = label2rgb(L, 'prism', 'k', 'shuffle');
61 figure, imshow(labels);
62 title(['Different labelled regions = ', num2str(num)]);
63
64 %table of stats containing the perimeter, area and circularity of each
65 %labelled region.
66 stats = regionprops('table', L, 'Perimeter', 'Area', 'Circularity');
67

```

Figure A.2 – Task 1 to 4 Code – Part 2/3

```

55
56 % Task 4: Object Recognition -----
57
58 %Getting labels from image
59 [L, num] = bwlabel(BW2);
60 labels = label2rgb(L, 'prism', 'k', 'shuffle');
61 figure, imshow(labels);
62 title(['Different labelled regions = ', num2str(num)]);
63
64 %table of stats containing the perimeter, area and circularity of each
65 %labelled region.
66 stats = regionprops('table', L, 'Perimeter', 'Area', 'Circularity');
67
68 %Creating a threshold for the circularity of a washer
69 washerCircularityThreshold = 0.8;
70
71 %Loop through each value in circularity column
72 %If the value is greater than the threshold it is set to one colour
73 %else it is set to another colour.
74 for i=1:height(stats)
75     if stats.Circularity(i) > washerCircularityThreshold
76         L(L == i) = 2;
77     else
78         L(L == i) = 1;
79     end
80 end
81
82 %Labels for image
83 rgb_label = label2rgb(L, 'prism', 'k');
84 figure, imshow(rgb_label);
85 title('Washers in orange, short screws in red');
86
87

```

Figure A.3 – Task 1 to 4 Code – Part 3/3

9 Appendix B

```

1  clear; close all;
2
3  inputFolder_I = 'Assignment_Input';
4  outputFolder = 'Assignment_Output';
5  GT_Folder = 'Assignment_GT';
6
7  %making output folder
8  if ~exist(outputFolder, 'dir')
9      mkdir(outputFolder);
10 end
11
12 % Task 5: Robust method -----
13
14 %Get Image folders
15 imageFiles = dir(fullfile(inputFolder_I, '*.jpg'));
16 GT_Files = dir(fullfile(GT_Folder, '*.png'));
17
18 for i = 1:length(imageFiles)
19     %read in each image
20     I = imread(fullfile(inputFolder_I, imageFiles(i).name));
21     I_gray = rgb2gray(I);
22
23     J = imresize(I_gray, 0.5, "bilinear");
24
25     enhancedJ = imsharpen(J, radius=2, amount=9);
26
27     BW = imbinarize(enhancedJ, "adaptive", "ForegroundPolarity", 'dark', 'Sensitivity', 0.40);
28     BW = ~BW;
29
30     SEsq = strel('square', 3);
31     SRect = strel('rectangle', [3 3]);
32
33     %Morphological operations

```

Figure B.1 – Task 5 to 6 Code – Part 1/4.

```

31     SRect = strel('rectangle', [3 3]);
32
33     %Morphological operations
34     I_Dilate = imdilate(BW, SRect);
35     I_Remove = bwareaopen(I_Dilate, 60);
36     I_Thicken = bwmorph(I_Remove, 'thicken');
37     I_Maj = bwmorph(I_Thicken, 'majority');
38     I_Filled = imfill(I_Maj, 'holes');
39     I_Open = imerode(I_Filled, SEsq);
40
41     I_Seg = medfilt2(I_Open, [6 6]);
42     % BW2 = imfilter(BW2, fspecial('average', 3));
43
44     I_Seg = bwareaopen(I_Seg, 30);
45
46     [L, num] = bwlabel(I_Seg);
47
48     stats = regionprops('table', L, 'Perimeter', 'Area', 'Circularity');
49     washerCircularityThreshold = 0.90;
50     longScrewAreaThreshold = 1500;
51
52     %no idea why, but if I use 1,2 and 3, they do not get classified
53     %correctly??
54     for j=1:height(stats)
55         if stats.Circularity(j) > washerCircularityThreshold
56             L(L == j) = 20;
57         elseif (stats.Area(j) > longScrewAreaThreshold)
58             L(L == j) = 21;
59         else
60             L(L == j) = 22;
61         end
62     end
63

```

Figure B.2 – Task 5 to 6 Code – Part 2/4.


```

63
64 %so I have to do it down here?? but it works in the other image
65 %task1-4?
66 L(L == 20) = 1; %washer
67 L(L == 21) = 3;%long screw
68 L(L == 22) = 2;%short screw
69
70
71 %label for processed img
72 rgb_label = label2rgb(L,'prism', 'k', 'shuffle');
73 figure, imshow(rgb_label);
74 title(['Processed Image: ', num2str(i)]);
75 %writing processed images to separate folder.
76 [~, baseName, ext] = fileparts(imageFiles(i).name);
77 outputFile = fullfile(outputFolder, [baseName '_processed' ext]);
78 imwrite(rgb_label, outputFile);
79
80
81 %Reading in ground truth images
82 GT = imread(fullfile(GT_Folder, GT_Files(i).name));
83 %Creating label for GT image
84 L_GT = label2rgb(GT, 'prism','k','shuffle');
85 figure, imshow(L_GT), title(['GT Image: ', num2str(i)]);
86
87
88 TP = sum((GT == L) & (GT == 1 | GT == 2 | GT == 3)); % True Positives
89 TN = sum(GT == L & GT == 0); % True Negatives
90 FP = sum(GT ~= L & L ~= 0); % False Positives\
91 FN = sum(GT ~= L & GT ~= 0); % False Negatives
92
93 %Convert to logical for dice score
94 logical_L = logical(rgb_label);
95 logical_GT = logical(L_GT);
96

```

Figure B.3 – Task 5 to 6 – Part 3/4.

```

80
81 %Reading in ground truth images
82 GT = imread(fullfile(GT_Folder, GT_Files(i).name));
83 %Creating label for GT image
84 L_GT = label2rgb(GT, 'prism','k','shuffle');
85 figure, imshow(L_GT), title(['GT Image: ', num2str(i)]);
86
87
88 TP = sum((GT == L) & (GT == 1 | GT == 2 | GT == 3)); % True Positives
89 TN = sum(GT == L & GT == 0); % True Negatives
90 FP = sum(GT ~= L & L ~= 0); % False Positives\
91 FN = sum(GT ~= L & GT ~= 0); % False Negatives
92
93 %Convert to logical for dice score
94 logical_L = logical(rgb_label);
95 logical_GT = logical(L_GT);
96
97 Dice_score = dice(logical_L, logical_GT);
98 Precision = TP/(TP+FP);
99 Recall = TP/(TP+FN);
100
101 disp(" ");
102 disp("Image " + i + ": ");
103
104 disp("Dice Score: " + Dice_score);
105 disp("Precision: " + Precision);
106 disp("Recall: " + Recall);
107
108 %figure, imshowpair(rgb_label, L_GT)
109 %title(['Dice Index = ' num2str(Dice_score)])
110
111 end
112

```

Figure B.4 – Task 5 to 6 – Part 4/4.