Report for Image Processing Assessment

Timothy Smith
Email: 25796944@students.lincoln.ac.uk
Student ID: 25796944

January 14, 2024

1 Pre-processing

1.1 Grayscale Conversion

The first step of the process is grayscale conversion. For the purposes of shape based object detection, colour is often a variable that only complicates processing and can be removed to simplify the process.

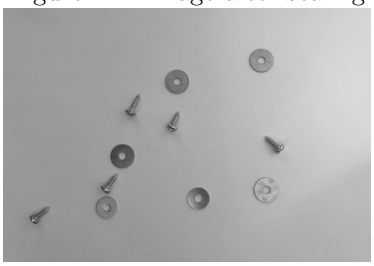
The code for grayscale conversion is already provided in `Task1to4.m` as given which uses the built-in function `rbg2gray`.

1.2 Size Reduction

Reducing the size of an image can be an excellent way to improve the performace of an object detection algorithm, by simply reducing the amount of data required to be processed. This has to be done within reason, of course, since reducing the image too far will cause of loss of information. Here to produce the image shown in figure 1.1 the built in function `imresize` is used. The image is scaled by half using bilinear interpolation as required using the code below:

```
I_scaled = imresize(I_gray, 0.5, 'bilinear');
```

Figure 1.1: Image after scaling



1.3 Enhancement

Note to self: this is subject to improvement.

In order to enhance the image we use the built-in function `imadjust`. This consists of simply passing the scaled image as an argument:

```
I_adjusted = imadjust(I_scaled);
```

Figure 1.2: Histogram prior to enhancement

Figure 1.3: Histogram after enhancement

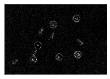
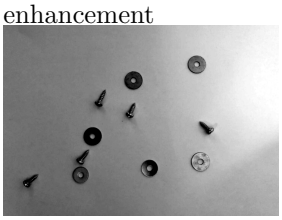


Figure 1.4: Image after enhancement



1.4 Binarisation

To binarise the image we call the built in function `im2bw`:

```
I_binarised = imadjust(I_enhanced); <<< FIX
```

blah blah splitting blah blah XXX binarisation.

2 Edge Detection

My approach here was to try a variety of edge detection methods and; see which one produced the most promising results.

The best looking turned out to be Canny, likely because XXX.

3 Simple Segmentation

We XXX the image to reduce noise and artefacts.

4 Object Recognition

Advantage is taken of the XXX method and we consider the circularity (I think that's the word) of the various objects.

5 Robust Method

Since we are required to avoid manually creating thresholds as much as possible, binarisation does not seem to be a robust segmentation method. This isn't an issue better enhancement either, since some of the washers and screws are lighter in the images than some of the background thanks to the unfavourable lighting conditions.

To devise a method that works more widely we need to return to edge detection methods applied to the enhanced (not binarised) image.

6 Performance Evaluation

Appendix

Task1to4.m

```
clear; close all;

save_report_images = true;

% Task 1: Pre-processing ------------------------
% Step-1: Load input image
I = imread('images/IMG_01.jpg');
%figure, imshow(I), title('Input Image');

% Step-2: Covert image to grayscale
I_gray = rgb2gray(I);
%figure, imshow(I_gray), title('Grayscaled Image');

% Step-3: Rescale image
I_scaled = imresize(I_gray, 0.5, 'bilinear');
figure, imshow(I_scaled), title('Scaled Image');

% Step-4: Produce histogram before enhancing
H_scaled = figure, imhist(I_scaled), title('Pre-enhancement Histogram');

% Step-5: Enhance image before binarisation
I_adjusted = imadjust(I_scaled);
figure, imshow(I_adjusted), title('Enhanced Image');

% Step-6: Histogram after enhancement
H_adjusted = figure, imhist(I_adjusted), title('Post-enhancement Histogram');

% Step-7: Image Binarisation
cutoffX = 500;
left_half =  I_adjusted(:,1:cutoffX);
right_half = I_adjusted(:,  cutoffX+1:end);
left_half  = im2bw(left_half,  70 /255);
right_half = im2bw(right_half, 130/255);
%figure, imshow(left_half);
%figure, imshow(right_half);
I_naively_binarised = im2bw(I_adjusted, 90/255);
%figure, imshow(I_naively_binarised), title('Binarised Image without Adaptive Thresholding')
I_binarised = [left_half right_half];
figure, imshow(I_binarised), title('Binarised Image with Adaptive Thresholding');
```

```
% Task 2: Edge detection ------------------------
[~, threshold] = edge(I_adjusted, 'prewitt');
threshold = threshold;
E_prewitt = edge(I_adjusted, 'prewitt', threshold);
%figure, imshow(E_prewitt), title('Prewit Edge Detectiont');

[~, threshold] = edge(I_adjusted, 'sobel');
E_sobel = edge(I_adjusted, 'sobel', threshold);
%figure, imshow(E_sobel), title('Sobel Edge Detection');

[~, threshold] = edge(I_adjusted, 'canny');
threshold = threshold * 6;
E_canny = edge(I_adjusted, 'canny', threshold);
figure, imshow(E_canny), title('Canny Edge Detection');

% laplacian
% reference mask somehow? It's in lecture 7
mask = [ 0 -1  0
    -1  4 -1
     0 -1  0];
mask = mask / 50;
E_laplacian = conv2(I_adjusted, mask);
%figure, imshow(E_laplacian), title('Laplacian Edge Detection');

% Task 3: Simple segmentation --------------------
close all;
mask = zeros(size(I_adjusted));
mask(1:end-1,1:end-1) = 1;
S = activecontour(I_adjusted, mask);
figure, imshow(S), title('Segmented Image');

% Task 4: Object Recognition --------------------

% Saving images for report --------------------

if save_report_images
saveas(H_scaled, '../report/images/scaled_histogram.png');
saveas(H_adjusted, '../report/images/enhanced_histogram.png');
imwrite(I_scaled, '../report/images/scaled.png');
imwrite(I_adjusted, '../report/images/enhanced.png');
imwrite(I_binarised, '../report/images/binarised.png');
imwrite(I_naively_binarised, '../report/images/naively_binarised.png');
imwrite(E_prewitt, '../report/images/prewitt.png');
imwrite(E_sobel, '../report/images/sobel.png');
imwrite(E_canny, '../report/images/canny.png');
imwrite(E_laplacian, '../report/images/laplacian.png');
```

```
end
```

Task5to6.m

```
pkg load image
% Task 5: Robust method --------------------------

% Task 6: Performance evaluation -----------------
% Step 1: Load ground truth data
GT = imread("ground-truth/IMG_01_GT.png");

% To visualise the ground truth image, you can
% use the following code.
L_GT = label2rgb(GT, 'prism','k','shuffle');
figure, imshow(L_GT)
```