CMP3108M Image Processing
Assignment 1

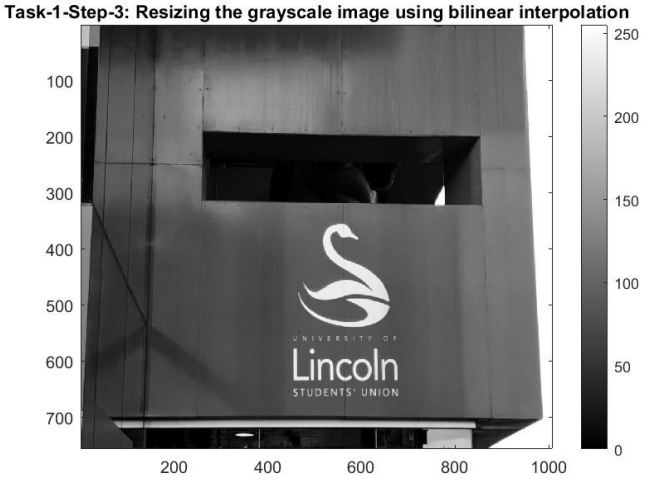
Marcus Valerio, 25150223

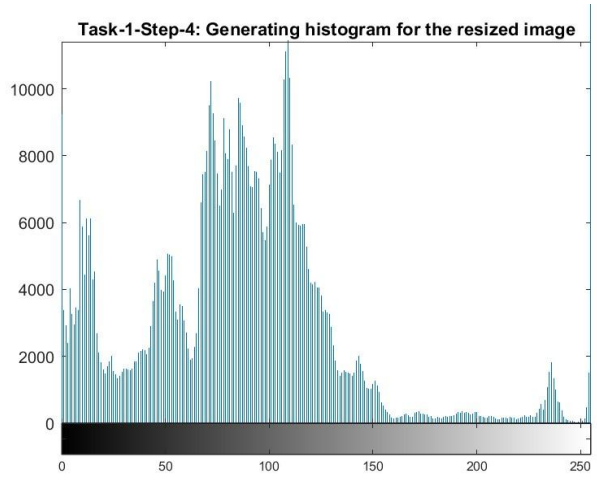
University of Lincoln, School of Computer Science

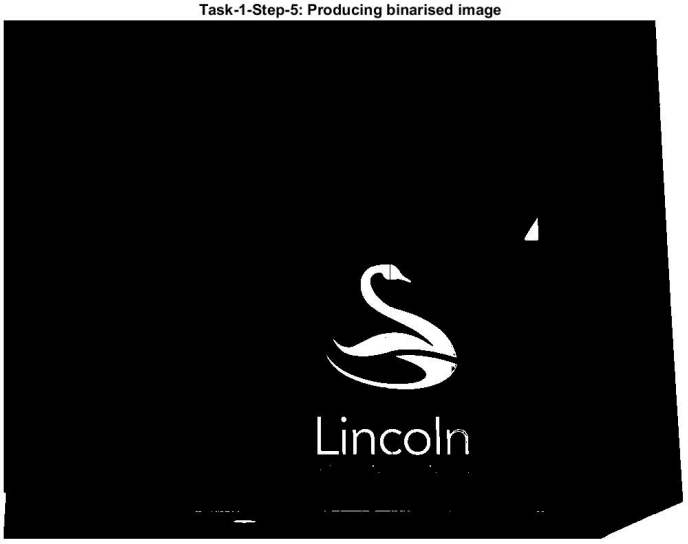
25150223@students.lincoln.ac.uk

1 Task 1 - Pre-processing

Based on the histogram created after resizing the greyscale image, I chose to binarize the image at the threshold value of 0.8671. This was calculated by choosing 222 as my value on the histogram and dividing it by 256, the total number of different grey levels in the image. My reasoning for the value of 222 was that it was after the large peak in the histogram, therefore only leaving fewer pixels with higher values and reducing noise, whilst also maintaining the outline of the swan for the future functionality of morphological operations. If I had used Otsu's thresholding method or automatic thresholding, the threshold value would have been too low, allowing more pixels with lower grey level values to be in the binary image, creating unnecessary noise.



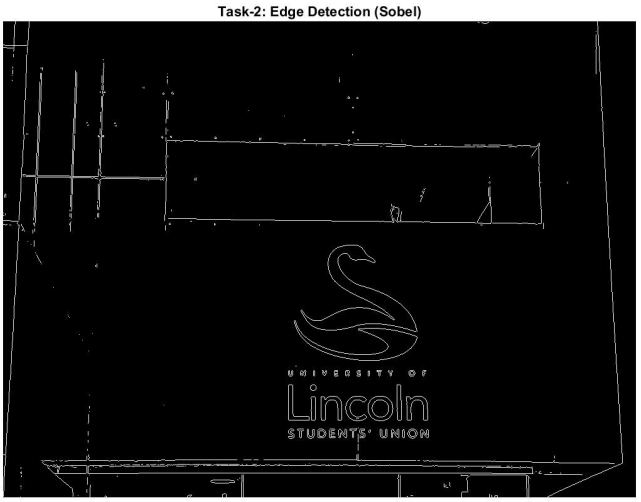


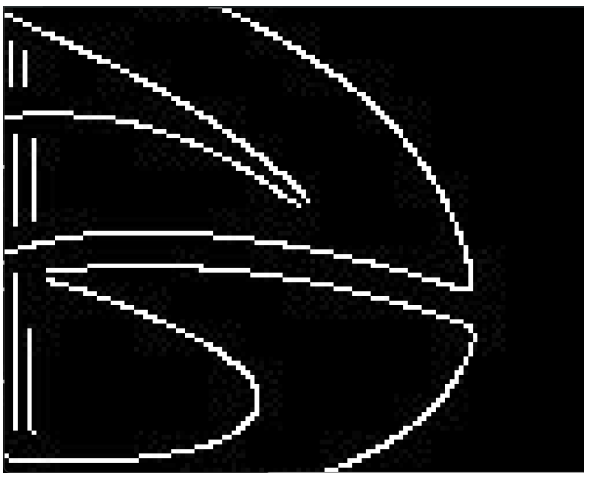


2 Task 2 - Edge Detection

Using the Canny operator compared to the Sobel operator creates a smoother image, with marginally thicker lines and a more filled out swan edge. This is due to the use of Non-maxima suppression and thresholding in the Canny edge detection technique. However, this leaves the image produced by using Canny's technique with unwanted edges in the whole image, whereas Sobel's edge-detected image is left with random noise. "Sobel is more sensitive to noise. This operator can give a smooth image since this operator has some smoothing effect of

the random noise in the image. But it has removed some important edge due to this factor. Canny operator not just can keep important edge, but this operator brings all the image contains" [1].





3 Task 3 - Simple Swan Segmentation

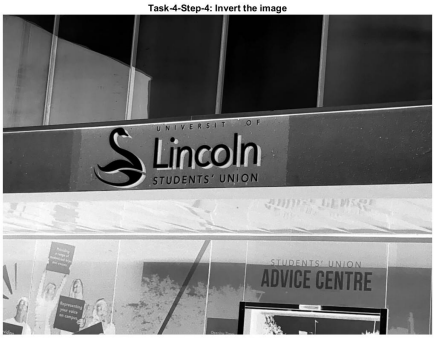
For this task, both the binary image produced in task 1 and the Sobel edge image created from task 2 were used to segment the swan in each picture; the process was almost identical also. The first step was to dilate the image to allow small gaps on the edges to be filled in both images; this created a thick, smooth border for the Sobel edge image and a complete edge for the binarised image. Next, the images were filled to eliminate any holes that were present in the binary image and to fill the Sobel edge image completely. These images were then eroded to reduce any noise in the image and to get the swan back to its original size. The 'imclose()' function to close the images was not available to use as it only offered one structuring element to be used in the operation, whereas in the dilating and eroding processes outlined above, two perpendicular line structuring elements were used as other shapes would not compliment the asymmetrical shape of the swan logo. The Sobel image was eroded again to refine the swan shape further and remove more noise that would complicate further steps.

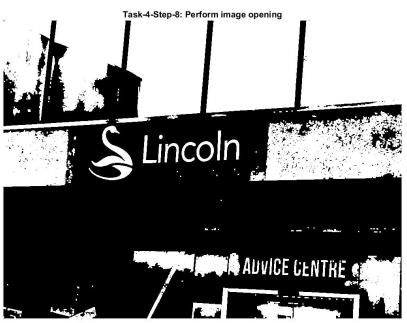
Both images had all detected objects on the border removed through the use of the 'imclearborder()' function [2] to aid in noise reduction and allow the pieces of the swan logo to be the two largest in the image. The final stage was to obtain the swan logo using connected component analysis. The function 'bwconncomp()' [3] was implemented to loop through all the components found in the images in ascending order, converting the values of all components except the two largest to zero, making them black. This left only the white swan in the foreground in both images.

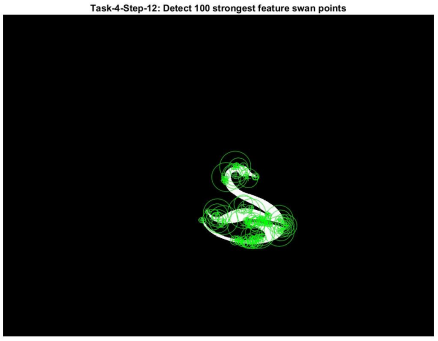
4 Task 4 - Swan Recognition

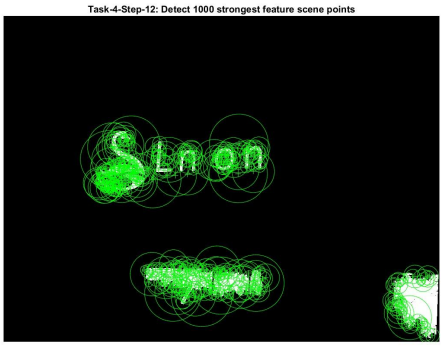
Unfortunately, it was not possible to get this task functioning properly, however, the steps that were taken to attempt this task will be mentioned.

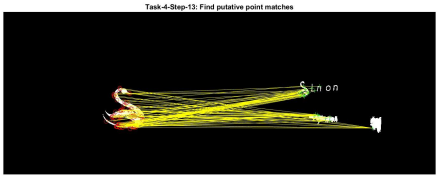
The first three steps were similar to task one, where the scene image was read into the script, converted into greyscale and reduced in size by half by bilinear interpolation. The next few steps work together to enhance the greyscale image and boost the contrast; this was done by creating an inverted version of the greyscale image, reducing its haze, and inverting it back to produce an enhanced version of the original greyscale image [4]. This image was then binarised using Otsu's threshold with a fudge factor of 0.191 applied to allow the soft threshold to work for most of the other 15 images. As a few images had swans connecting to other objects, image opening was used to create the gap and leave the swan separate, and the border was cleared to take away the largest object that was present in most of the images. Similar to task three, connected component analysis was applied to eliminate the smallest objects (unnecessary noise) to leave the ten largest objects, which would include the swan. To use as a guide to detecting the swan, the ground truth image "IMG_01_GT.JPG" was read into the Matlab script. Feature points were then detected on both the true swan image and the scene image to highlight the strongest features detected in both images using the function 'extractFeatures()' [5], which would be used to detect similarities. These points in both images were aligned, although there were anomalies detected that needed to be cleared by using the 'estgeotform2d' function [5]. However, matching the points was impossible as it could not find enough strong feature points between the two swans to confidently make a connection.











References

[1] - Beeran Kutty, S., Saaidin, S., Megat Yunus, P.N.A. and Abu Hassan, S. (2014). *Evaluation of canny and sobel operator for logo edge detection*. [online] IEEE Xplore. doi:10.1109/ISTMET.2014.6936497.

[2] - uk.mathworks.com. (n.d.). *Suppress light structures connected to image border - MATLAB imclearborder - MathWorks United Kingdom*. [online] Available at: https://uk.mathworks.com/help/images/ref/imclearborder.html [Accessed 20 Dec. 2022].

[3] - uk.mathworks.com. (n.d.). *Find and count connected components in binary image - MATLAB bwconncomp - MathWorks United Kingdom*. [online] Available at: https://uk.mathworks.com/help/images/ref/bwconncomp.html [Accessed 20 Dec. 2022].

[4] - uk.mathworks.com. (n.d.). *Low-Light Image Enhancement - MATLAB & Simulink Example - MathWorks United Kingdom*. [online] Available at: https://uk.mathworks.com/help/images/low-light-image-enhancement.html [Accessed 04 Jan. 2023].

[5] - uk.mathworks.com. (n.d.). *Object Detection in a Cluttered Scene Using Point Feature Matching - MATLAB & Simulink - MathWorks United Kingdom*. [online] Available at: https://uk.mathworks.com/help/vision/ug/object-detection-in-a-cluttered-scene-using-point-feature-matching.html [Accessed 04 Jan. 2023].

Appendix

```
% MATLAB script for Assessment 1
% Task 1: Preprocessing --------------------------
clear; close all; clc;
% Step-1: Load input image
I = imread('IMG_01.jpg');
figure;
imshow(I);
```

```
title('Task-1-Step-1: Load input image');
% Step-2: Conversion of input image to grey-scale image
Igray = rgb2gray(I);
figure;
imshow(Igray);
title('Task-1-Step-2: Conversion of input image to greyscale');
% Step-3: Resizing the grayscale image using bilinear interpolation
IgrayHalf = imresize(Igray,0.5,'bilinear');
figure;
imagesc(IgrayHalf);
colorbar;
colormap gray;
title('Task-1-Step-3: Resizing the grayscale image using bilinear interpolation');
% Step-4: Generating histogram for the resized image
figure;
imhist(IgrayHalf)
title('Task-1-Step-4: Generating histogram for the resized image');
% Step-5: Producing binarised image
Ibinary = imbinarize(IgrayHalf,0.8671); % 222/256 to calculate threshold
figure;
imshow(Ibinary);
title('Task-1-Step-5: Producing binarised image')
%-----------------------------------------------------------------
% Task 2: Edge Detection ----------------------
% -------------------- SOBEL --------------------
edgeS = edge(IgrayHalf,'sobel');
figure;
imshow(edgeS);
title('Task-2: Edge Detection (Sobel)')
% -------------------- CANNY --------------------
edgeC = edge(IgrayHalf,'canny');
figure;
imshow(edgeC);
title('Task-2: Edge Detection (Canny)');
%-----------------------------------------------------------------
% Task-3: Simple Segmentation ----------------------
% ------------------ Binary Image ------------------
% Step-1: Dilate Image
se90 = strel('line',3,90);
se0 = strel('line',3,0);
BDilate = imdilate(Ibinary,[se90,se0]);
figure;
imshow(BDilate)
title('Task-3-Step-1: Dilate Image (Binary Image)')
% Step-2: Fill Image
BFill = imfill(BDilate,'holes');
figure;
imshow(BFill)
title('Task-3-Step-2: Fill Image (Binary Image)')
% Step-3: Erode Image
BErode = imerode(BFill,[se90,se0]);
figure;
imshow(BErode)
title('Task-3-Step-3: Erode Image (Binary Image)')
% Step-4: Remove Connected Objects On Border
Bnobord = imclearborder(BErode,4);
figure;
imshow(Bnobord);
title('Task-3-Step-4: Remove Connected Objects On Border (Binary Image)')
% Step-5: Finalizing the Swan
```

```
BCC = bwconncomp(Bnobord);
while BCC.NumObjects > 2
  numPixels = cellfun(@numel,BCC.PixelIdxList);
  [smallest,idx] = min(numPixels);
  Bnobord(BCC.PixelIdxList{idx}) = 0;
  BCC = bwconncomp(Bnobord);
end
figure;
imshow(Bnobord)
title('Task-3-Step-5: Finalizing the Swan (Binary Image)')
% ----------------- Sobel Edge ------------------
% Step-1: Dilate Image
se90 = strel('line',3,90);
se0 = strel('line',3,0);
SDilate = imdilate(edgeS,[se90,se0]);
figure;
imshow(SDilate)
title('Task-3-Step-1: Dilate Image (Sobel Edge)')
% Step-2: Fill Image
SFill = imfill(SDilate,'holes');
figure;
imshow(SFill)
title('Task-3-Step-2: Fill Image (Sobel Edge)')
% Step-3: Erode Image
SErode = imerode(SFill,[se90,se0]);
SErode = imerode(SErode,[se90,se0]);
figure;
imshow(SErode)
title('Task-3-Step-3: Erode Image (Sobel Edge)')
% Step-4: Remove Connected Objects On Border
Snobord = imclearborder(SErode,4);
figure;
imshow(Snobord);
title('Task-3-Step-4: Remove Connected Objects On Border (Sobel Edge)')
% Step-5: Finalizing the Swan
SCC = bwconncomp(Snobord);
while SCC.NumObjects > 2
  numPixels = cellfun(@numel,SCC.PixelIdxList);
  [smallest,idx] = min(numPixels);
  Snobord(SCC.PixelIdxList{idx}) = 0;
  SCC = bwconncomp(Snobord);
end
figure;
imshow(Snobord)
title('Task-3-Step-5: Finalizing the Swan (Sobel Edge)')

% MATLAB script for Assessment Item-1
% Task 4: Robust Swan Recognition ----------------
clear; close all; clc;
% Step-1: Read image
I = imread("IMG_02.JPG");
figure, imshow(I)
title('Task-4-Step-1: Read Image')
% Step-2: Produce Greyscale Image
grayscale = rgb2gray(I);
figure, imshow(grayscale)
title('Task-4-Step-2: Produce Greyscale Image');
% Step-3: Resizing the grayscale image using bilinear interpolation
grayscaleHalf = imresize(grayscale,0.5,'bilinear');
figure;
```

```
imagesc(grayscaleHalf);
colorbar;
colormap gray;
title('Task-4-Step-3: Resizing the grayscale image using bilinear interpolation');
% Step-4: Invert the image
Inv = imcomplement(grayscaleHalf);
figure, imshow(Inv)
title('Task-4-Step-4: Invert the image')
% Step-5: Reduce the haze
InvRed = imreducehaze(Inv, 0.95, 'method', 'approxdcp');
figure, imshow(InvRed)
title('Task-4-Step-5: Reduce the haze')
% Step-6: Invert to obtain enhanced image
enhanced = imcomplement(InvRed);
figure, imshow(enhanced)
title('Task-4-Step-6: Invert to obtain enhanced image')
% Step-7: Binarise the image
T = graythresh(enhanced); % Use Otsu's thresholding
BW = imbinarize(enhanced,T+0.191); % Refine the threshold
figure, imshow(BW)
title('Task-4-Step-7: Binarise the image')
% Step-8: Perform image opening
se90 = strel('line',3,90);
se0 = strel('line',3,0);
erode = imerode(BW, [se90, se0]);
dilate = imdilate(erode, [se90, se0]);
figure, imshow(dilate)
title('Task-4-Step-8: Perform image opening')
% Step-9: Clear the border
BWclear = imclearborder(dilate);
figure, imshow(BWclear)
title('Task-4-Step-9: Clear the border')
% Step-10: Remove the smallest components, leave ten largest
CC = bwconncomp(BWclear);
while CC.NumObjects > 10
  numPixels = cellfun(@numel,CC.PixelIdxList);
  [smallest,idx] = min(numPixels);
  BWclear(CC.PixelIdxList{idx}) = 0;
  CC = bwconncomp(BWclear);
end
figure, imshow(BWclear)
title('Task-4-Step-10: Remove the smallest components, leave ten largest')
% Step-11: Read Swan image
swan = imread("IMG_01_GT.JPG");
figure, imshow(swan)
title('Task-4-Step-11: Read Swan image')
% Step-12: Detect feature points
swanPoints = detectSURFFeatures(swan);
scenePoints = detectSURFFeatures(BWclear);
figure, imshow(swan)
title('Task-4-Step-12: Detect 100 strongest feature swan points')
hold on;
plot(selectStrongest(swanPoints, 100));
figure, imshow(BWclear)
title('Task-4-Step-12: Detect 1000 strongest feature scene points')
hold on;
plot(selectStrongest(scenePoints, 1000));
% Step-13: Find putative point matches
[swanFeatures, swanpoints] = extractFeatures(swan, swanPoints); % Extract feature descriptors
[sceneFeatures, scenePoints] = extractFeatures(BWclear, scenePoints); % Extract feature descriptors
```

```
swanPairs = matchFeatures(swanFeatures, sceneFeatures);
matchedSwanPoints = swanPoints(swanPairs(:, 1), :);
matchedScenePoints = scenePoints(swanPairs(:, 2), :);
figure;
showMatchedFeatures(swan, BWclear, matchedSwanPoints, matchedScenePoints, 'montage');
title('Task-4-Step-13: Find putative point matches')
% Step-14: Locate the swan in the scene
[tform, inlierIdx] = estgeotform2d(matchedSwanPoints, matchedScenePoints, 'affine');
inlierSwanPoints   = matchedSwanPoints(inlierIdx, :);
inlierScenePoints = matchedScenePoints(inlierIdx, :);
figure;
showMatchedFeatures(swan, BWclear, inlierSwanPoints, inlierScenePoints, 'montage');
title('Task-4-Step-14: Locate the swan in the scene')
% Task 5: Performance Evaluation ----------------
% Step 1: Load ground truth data
% GT = imread("IMG_01_GT.png");
% To visualise the ground truth image, you can
% use the following code.
% figure, imshow(GT)
```