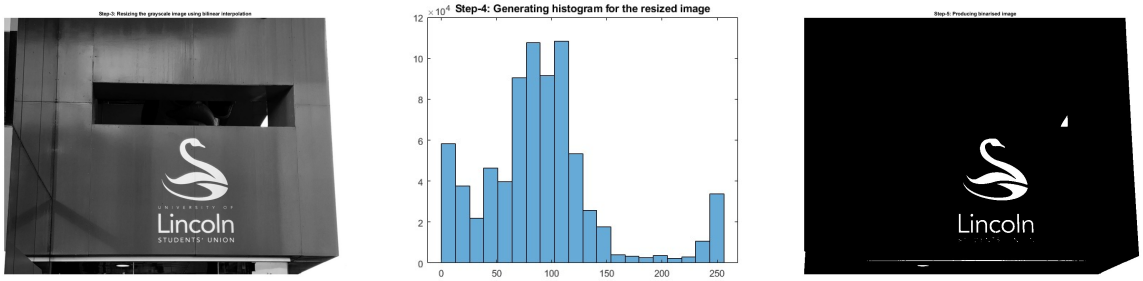
CMP3108M/9055M Image Processing

Jacob Morgan — 2523460
25234606@students.lincoln.ac.uk

Contents

1 Pre Processing



(a) Resizing the grayscale image (b) Generating histogram (c) Binarised image

Figure 1: Prepossessing Task 1-4

1.1 Loading input image

Here I am simply loading the first image from the image folder using the `imread` function.

1.2 Conversion of input image to grey-scale image

I used the `rgb2gray` to convert the image to grey scale making it easier to differentiate between colours and segment out the swan later on. "Reversing the intensity levels of an image in this manner produces the equivalent of a photographic negative. This type of processing is particularly suited for enhancing white or gray detail embedded in dark regions of an image, especially when the black areas are dominant in size" (Gonzalez & Woods 2017)

1.3 Resizing the grayscale image using bilinear interpolation

Interpolation fits a 2-D polynomial to the known points surrounding the unknown point and uses its value at the unknown point (Peters 2021). This helps reduce contrast and helps smooth out images, whilst also making the images blurry. Utilising bilinear interpolation would make the most sense in this context since I have to detect a curved segment of the image, the segment itself needs to be as high quality and smooth as possible.

1.4 Generating histogram for the resized image

To decide the binarise threshold we use a histogram to get the amount of pixels at each brightness level. The large spike at the end of figure 1b is where I would expect the swan to be, since it is majority white pixels. I decided the threshold should be 210 since that is where the peak starts the rise.

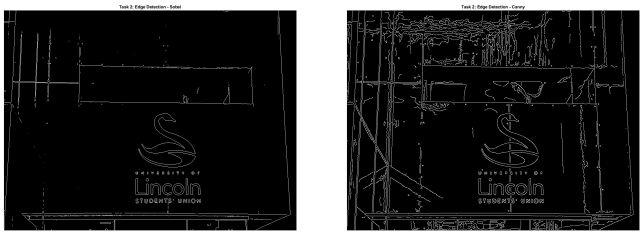
1.5 Producing binarised image

To get rid of unnecessary details and colours I used the `imbinerize` function in matlab to binerise the image. This produces figure 1c which is very close to the result I want, I will just now need

to remove some anomaly's such as the white block on the right side of the image and the `Lincoln` text.

2 Edge Detection

Edges correspond to large changes in intensity of neighbours pixels in at least one direction. Edges are one of the most important features associated with images. Here I used the sobel and canny techniques to work out the edges of the swan. I used matlabs built in function `edge` to apply the filters onto the binarised image to get the results seen in figure 2.



(a) Edge Detection - Sobel (b) Edge Detection - Canny

Figure 2

3 Simple Swan Segmentation

From figure 3 you can tell the swan is segmented perfectly from the rest of the image. I did this by using a filter to get rid of every group of pixels below 2000 pixels and above 7500 and filters out any remaining noise. The filter function I used is called `bwareafilt`. Another way of doing this would be connect component analysis which involves analysing the binarised image and storing the three largest values in an array. Then extracting the swan using the array.

Figure 3: Swan segmented from the rest of the image

4 Swan Recognition

The first technique I used to create the program was create a target object, I utilised the segmented swan I got from `task1to3`. I used the matlab function `detectSURFFeatures` to detect the points on the segmented swan. These points are used to compare to other points I get from a binarised version of the photo. I then extract the features using `extractfeatures` for both the image and the swan. This leaves me with features for both the swan and the image in which I can use the `matchFeatures` function to get both points on the image where a swan is detected. Transforming these two points onto each other and warping the image with the points gotten from a `estimateGeometricTransform` function leaves me with the final output. This final output will be used to compare against the ground truth image later on.

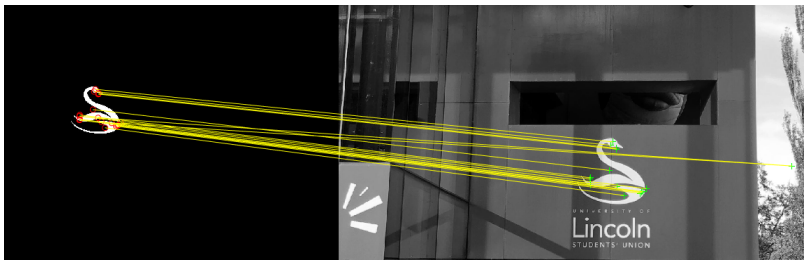


Figure 4: Points of interest mapped onto image

Figure 4 shows a visualisation of what I'm doing to the segmented swan. Cross referencing it with a binarised image. One setback here is binarising the images. I found that 0.7 works as the threshold for around 60% of the images but it isn't perfect, to improve I would implement `Otsu's method` , which is used to perform automatic image thresholding (Morse 2000). Matlab has a function for this called `graythresh` (*graythresh method* 2023), but when used in my code the function always predicts the threshold to be far too low. A way to combat this would be to use a multiplier , but this wouldn't be a permanent solution.

5 Performance Evaluation

A dice function or `Sørensen Dice coefficient` is primarily used to find the similarity between two images or samples. Represented as the following equation

$$DSC = \frac{2|X \bigcap Y|}{|X| + |Y|}$$

Where $|X|$ and $|Y|$ are the number of elements of each set. I use this to test if my predicted image is similar to the ground truth image supplied. I used the built in function `dice` in matlab to compare the two binarised images and get a metric. The results can be seen in table 1.

1 : 0.9262	2 : 0.5320	3 : 0.9048	4 : 0.9277	5 : 0.8309	6 : 0.5451
7 : 0.8098	8 : 0.0220	9 : 0.7817	10 : 0.7462	11 : 0.8728	12 : 0.9103
13 : 0.0142	14 : 0	15 : 0.9076	16 : 0.9076		

Table 1: Table of dice scores [Image number : dice score]

To further analyse these results I can get the mean of the dice scores. This tells us the efficiency of the recognition function. I can get the standard deviation as well to tells me how spread out my predictions. These can be seen in 2.

Mean	0.6181
Standard Deviation	0.3830

Table 2: Mean and Standard Deviation for dice scores

My overall efficiency is more than half of the images. Which isnt the best and could be improved upon. My standard deviation for my recognition is quite high, meaning my dice score could range from 0 to 0.9 for any of my images.

This could be improved upon by improving how we deal with threshold of binarised images. As stated earlier in Section 4 , the use of `Otsu's method` could help get the threshold automatically , fitting a threshold to each image and not using an inefficient hard threshold. Overall I can't say that my accuracy of my method is very reliable.

6 Appendix

6.1 Task1to3.m

```
% MATLAB script for Assessment 1
% Task 1: Preprocessing ---------------------------
clear; close all; clc;

% Step-1: Load input image
I = imread('Images/IMG_01.JPG');
figure;
imshow(I);
title("Step-1: Load input image");

% Step-2: Conversion of input image to grey-scale image
Igray = rgb2gray(I);
figure;
imshow(Igray)
size(Igray)
title("Step-2: Conversion of input image to grey-scale image")

% Step-3: Resizing the grayscale image using bilinear interpolation
Igraybi = imresize(Igray,0.5,"bilinear");
figure;
imshow(Igraybi)
size(Igraybi)
title("Step-3: Resizing the grayscale image using bilinear interpolation")

% Step-4: Generating histogram for the resized image
figure;
histogram(Igraybi,20)
title("Step-4: Generating histogram for the resized image")

% Step-5: Producing binarised image
threshold = double(210/255);

binarisedImage = imbinarize(Igraybi,threshold);
figure;
imshow(binarisedImage)
title("Step-5: Producing binarised image")
%------------------------------------------------------------
% Task 2: Edge Detection ----------------------
edgeDetectionSobel = edge(Igraybi,'sobel');
figure;
imshow(edgeDetectionSobel)
title("Task 2: Edge Detection - Sobel")
edgeDetectionCanny = edge(Igraybi,'canny');
```

```
figure;
imshow(edgeDetectionCanny)
title("Task 2: Edge Detection - Canny")

%-------------------------------------------------------------
% Task-3: Simple Segmentation ----------------------
binerisedImagecleaner = bwareafilt(binarisedImage,[2000,7500]); %removes pixels <2000 - 7500>
figure;
imshow(binerisedImagecleaner)
title("Task 3: Simple Segmentation")
imwrite(binerisedImagecleaner,"Images/binswan.jpg")
```

6.2 Task4to5.m

```
% MATLAB script for Assessment Item-1
clear; close all; clc;
GT = imread("Images/binswan.jpg");
GT = imcrop(GT,[440.5 340.5 249 223]);

dice_scores = [];

for i = 01:16
  if i < 10
    i = sprintf("%02d",i);
  end

  try
    groundtruth = imread("Assignment_GT/IMG_" + i + "_GT.JPG");

    actual = imread("Images/IMG_" + i + ".JPG");

    Ir = swan_detection(actual, GT);
    figure;
    imshow(Ir);
    title("Swan " +i)
    Ir = imbinarize(Ir,0.9);
    groundtruth = imbinarize(groundtruth,0.9);
    dice_score = dice(Ir, groundtruth);
    dice_scores = [dice_scores dice_score];
  catch ME
    disp("Error Threshold too low")
  end
end

meandice = mean(dice_scores)
```

```
stddice  = std(dice_scores)
dice_scores


function Ir = swan_detection(actual,GT)

  actualGray = rgb2gray(actual);

  swanPoints = detectSURFFeatures(GT);

  meanContrast = mean(mean(actualGray));
  img = imadjust(actualGray,[meanContrast/255,1]);
  thresh = graythresh(img);

  binarisedImage = imbinarize(actualGray,0.7);

  actualPoints = detectSURFFeatures(binarisedImage);

  [sceneFeatures,scenePoints] = extractFeatures(binarisedImage,actualPoints);


  [swanFeatures,swanPoints] = extractFeatures(GT,swanPoints);
  swanPairs = matchFeatures(swanFeatures,sceneFeatures);
  matchedSwanPoints = swanPoints(swanPairs(:,1),:);
  matchedScenePoints = scenePoints(swanPairs(:,2),:);

  [tform, inlierSwanPoints, inlierScenePoints] = estimateGeometricTransform(matchedSwanPoints, matchedScenePoints, 'affine

  % Transform obj image onto input image  % On the obj image variable to get output
  outputView = imref2d(size(binarisedImage));
  Ir = imwarp(GT,tform,"OutputView",outputView);

end
```

References

Gonzalez, R. & Woods, R. (2017), *Digital Image Processing, Global Edition*, Pearson Education, Limited, Harlow, UNITED KINGDOM.
URL: *http://ebookcentral.proquest.com/lib/ulinc/detail.action?docID=5573669*

graythresh method (2023).
URL: *https://uk.mathworks.com/help/images/ref/graythresh.html?searchHighlight=graythreshs*$_t id = srchtitle_g raythresh_1$

Morse, B. S. (2000), 'Lecture 4: Thresholding'.
URL: *https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL*$_C OPIES/MORSE/threshold.pdf$

Peters, A. (2021), 'Lectures on Image Processing : Alan Peters : Free Download, Borrow, and Streaming : Internet Archive'.
URL: *https://ia902707.us.archive.org/23/items/Lectures*$_o n_I mage_P rocessing/EECE_4 353_1 3_R esampling.pdf$