|  |
| --- |
| \\WIN-OCJ0T7LG9V7\Users\Bodomite\Desktop\Capture.PNG |
| Report  Automated Image Recognition System |
| |  |  |  | | --- | --- | --- | | IIS – G08 | 3/21/15 | CSC3030 – Intelligent information Systems | |

Contents

[Introduction 2](#_Toc414755364)

[Team Members 2](#_Toc414755365)

[The System 2](#_Toc414755366)

[The Methodology 2](#_Toc414755367)

[System “Correctness” 3](#_Toc414755368)

[The Default System 5](#_Toc414755369)

[Pre-Processing 5](#_Toc414755370)

[Segmentation 5](#_Toc414755371)

[Feature Extraction 5](#_Toc414755372)

[Classifier 5](#_Toc414755373)

[Post-Processing 6](#_Toc414755374)

[Brightness Enhancement 6](#_Toc414755375)

[Contrast Enhancement 7](#_Toc414755376)

[Automated Linear Stretch 7](#_Toc414755377)

[Histogram Equalisation 7](#_Toc414755378)

[Power Law 8](#_Toc414755379)

[Summary 10](#_Toc414755380)

[Noise Reduction 11](#_Toc414755381)

[Low Pass Filter (LPF) 11](#_Toc414755382)

[Median Filter 12](#_Toc414755383)

[Summary 12](#_Toc414755384)

[Segmentation and Post Processing 13](#_Toc414755385)

[Feature Extraction 14](#_Toc414755386)

[Classification 15](#_Toc414755387)

[Appendix 16](#_Toc414755388)

[System Instructions 16](#_Toc414755389)

# Introduction

This is a report on the Automated Image Recognition System (AIRSystem) developed by Group 8 for CSC3030 – Intelligent Information Systems.

## Team Members

* 40057686 - Glen Brown
* 14375036 – Raymond McMillan
* 40098106 – Minh Trung Tran
* 40075949 – Minh Tuan Huynh

## The System

The system is essentially a pipeline that takes in image sets, transforms the images into data sets, and then trains / tests against a classifier to determine if the images are in a class or not. In this system, the two classes that images can be in is either “Glaucoma” or “Healthy”. The system contains 4 main components:

1. Pre-processing.
2. Segmentation and Post-Processing.
3. Feature Extraction
4. Classification.

Each member of the team will cover a section of the system, in the same order as listed above (in addition to Glen Brown writing this introduction).

## The Methodology

In order to determine the best set of techniques to use at each stage of the system, we will use a set methodology. We are going to maximise the system “correctness” at each individual stage, so when complete, the entire system should be as correct as possible given the techniques we’ve implemented.

To measure the correctness of the system, we are going to run a full training / testing cycle for each technique / parameter value and see which gives the best results.

### System “Correctness”

To judge the system correctness, we need a metric that accurately represents the quality of the system. We decided that the best way was to measure how well the classifier did as that takes the entire system into account (and it’s the entire system that we are ultimately trying to tweak for maximum “correctness”). Within that though, there are still a couple of choices: Accuracy, and what we are calling “True Accuracy”.

#### Accuracy

Accuracy is defined as follows:

There is however a problem with this metric:

Consider a classifier that simply returns that an image is in the class “Glaucoma” 100% of the time.

Now, consider that when classifying a set of images, that 95 out of 100 images have glaucoma. The accuracy would then be the following:

Given the result of 95%, you would assume that the system was very accurate, despite it just returning true each time! As a further example, let’s assume that only 3 out of 100 images have glaucoma:

The point is that Accuracy when defined like this is very much dependent on the ratio of Glaucoma to Healthy images in the testing set. This is true in our assignment as 13 / 16 images are Glaucoma, with only 3 being healthy. Because of the uneven ratio, our terrible system that returns true each time would have an accuracy as follows:

Because of this, we decided to use a different metric when maximising the “correctness” of the system, which we call “True Accuracy”.

#### True Accuracy

To understand what we’re calling true accuracy, we need to lay out a table on what true / false positives / negatives are:

|  |  |  |
| --- | --- | --- |
| Class determined by Classifier | Actual Class | Name |
| Positive | Positive | True-Positive |
| Positive | Negative | False-Positive |
| Negative | Positive | False-Negative |
| Negative | Negative | True-Negative |

True Accuracy is defined as follows.

What this does is get the rate at which positive images where actually determined to be positive, and the rate at which negative images where actually determined to be negative, and averages them.

To show why this is better, using our useless system where everything returns true on the test-set (where 13/16 of the images are Glaucoma)

Whilst the accuracy was 81%, the True Accuracy is now down to 50%. Using the same classifier but on a test set where only 3 / 100 images are Glaucoma:

As you can see, the True Accuracy is still 50%.

In summary, while the “Accuracy” was heavily dependent on the ratio of Glaucoma to Healthy images in the test set, the “True Accuracy” value is now independent of this ratio, giving a much clearer view of how “correct” the system itself is. It is for this reason that we will use the true accuracy throughout when maximising performance of the AIRSystem.

## The Default System

Since the way we’ll be maximising the true accuracy of the system is to maximise it at each section, we need a default system so that we can still obtain values of true accuracy. The makeup of this system doesn’t matter, provided it returns actionable results (i.e. not totally random so we can compare with other systems).

## Pre-Processing

* No Brightness Enhancement.
* No Contrast Enhancement.
* No Noise Reduction.

## Segmentation

* No Post-Processing

## Feature Extraction

The features we will be extracting are:

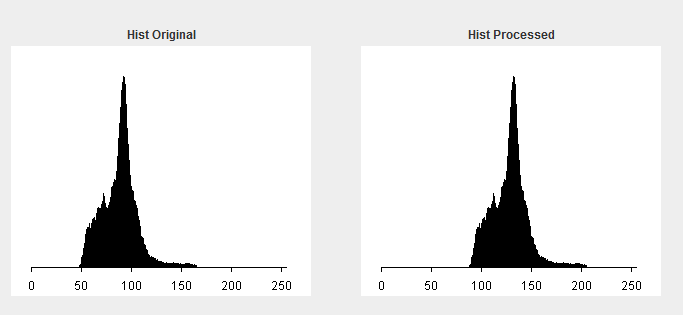
## Classifier

* -nearest neighbour.

# Post-Processing

## Brightness Enhancement

We implemented Automated Brightness Enhancement (ABH): the point of which is to normalise the image so that the mean grey-value in the image is equal to 127 (255 / 2). The results look like the following:



|  |  |  |
| --- | --- | --- |
| Technique | Accuracy (%) | True Accuracy (%) |
| ABH Disabled | 63 | 64 |
| ABH Enabled | 68 | 55 |

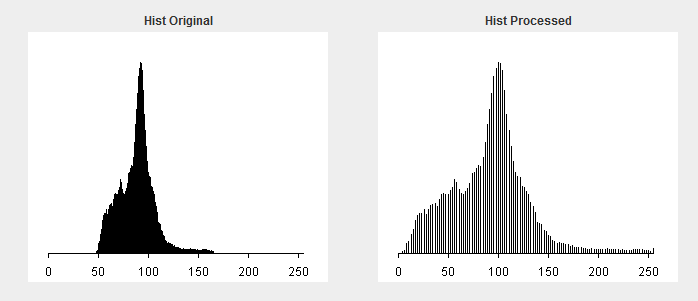
The system does not benefit from the ABH being enabled (as the true accuracy decreases): we will therefore leave it disabled. The reason it does harm to the system is likely due to it destroying information in images that use more of the dynamic range than the example above, which would result in a lot of grey levels being clipped at 0 and 255.

The reason the accuracy increases while the true accuracy falls is that the ABH is skewing the classifier to return positive (that the image is Glaucoma) across the board, therefore increasing the number of both True-Positives (improving accuracy due to the class-ratio imbalance) and False-Positives (which decreases the number of True-Negatives, decreasing true accuracy).

## Contrast Enhancement

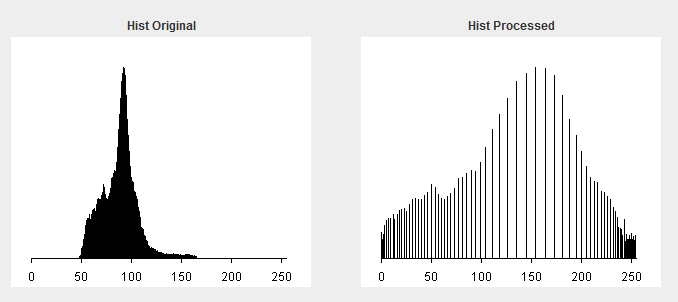
We have 3 different types of contrast enhancement: Automated Linear Stretch (ALS), Histogram Equalisation and the Power Law. These are covered in the lecture slides so we will not cover them here, except to say that the Automated Linear Stretch can detect good start / end grey levels to stretch out an image to use the full dynamic range. We will only pick one of these techniques, if any.

### Automated Linear Stretch



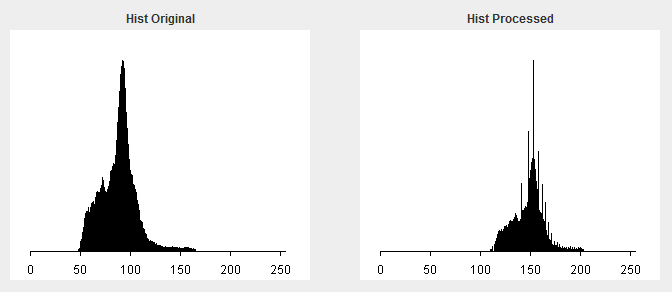
|  |  |  |
| --- | --- | --- |
| Technique | Accuracy (%) | True Accuracy (%) |
| ALS Disabled | 63 | 64 |
| ALS Enabled | 56 | 47 |

### Histogram Equalisation



|  |  |  |
| --- | --- | --- |
| Technique | Accuracy (%) | True Accuracy (%) |
| Histogram Equalisation Disabled | 63 | 64 |
| Histogram Equalisation Enabled | 44 | 27 |

### Power Law

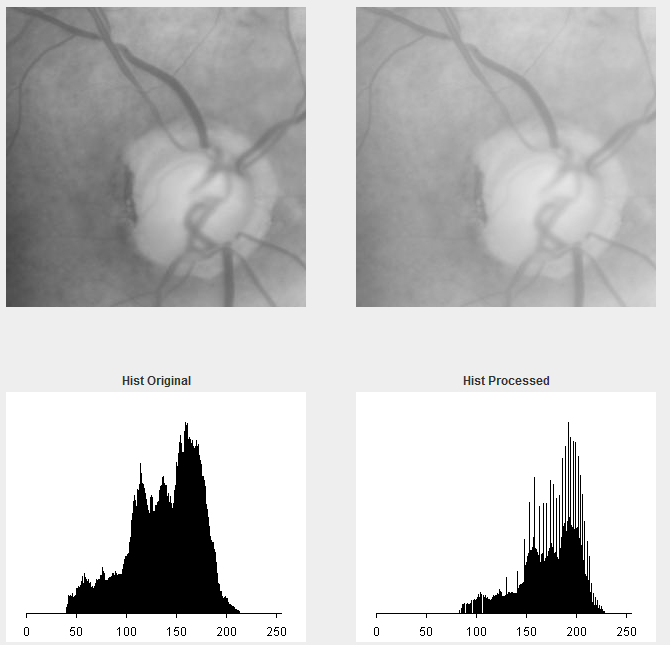


This example is where

We have no automated way to find the value for (gamma) per image. Therefore, we will test every value of gamma from 0 to 2.0 in 0.1 increments and see if any of the results give a higher true accuracy than when it’s not enabled at all.

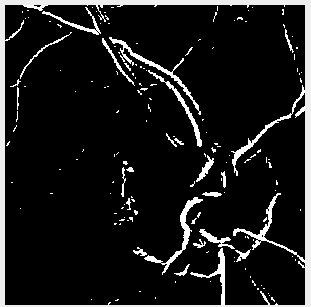
|  |  |  |
| --- | --- | --- |
| Technique | Accuracy (%) | True Accuracy (%) |
| Power Law Disabled | 63 | 64 |
| The following all refer to the Power Law being enabled with “Technique” being the value | | |
| 0.0 | 81 | 50 |
| 0.1 | 63 | 64 |
| 0.2 | 38 | 36 |
| 0.3 | 63 | 51 |
| 0.4 | 50 | 31 |
| 0.5 | 69 | 55 |
| 0.6 | 88 | 92 |
| 0.7 | 81 | 76 |
| 0.8 | 63 | 51 |
| 0.9 | 63 | 64 |
| 1.1 | 63 | 64 |
| 1.1 | 56 | 60 |
| 1.2 | 56 | 60 |
| 1.3 | 56 | 60 |
| 1.4 | 56 | 60 |
| 1.5 | 44 | 40 |
| 1.6 | 63 | 64 |
| 1.7 | 63 | 64 |
| 1.8 | 50 | 44 |
| 1.9 | 56 | 60 |
| 2.0 | 38 | 36 |

Highlighted in green above shows that when, the accuracy is 88% and true accuracy is 92%. Here’s an image with the power law applied when:



As you can see, the original image is on the left with the processed image on the right, and their respective histograms underneath. It seems that the power law has made the dynamic range of the image worse rather than better!

Having a look at the segmented binary image may explain why this has improved the true accuracy to >90%:



Judging from this image, reducing the contrast at the higher end (which is what does) is allowing the Segmenter (which is still set at its defaults of Edge Extraction with and no post-processing) to detect the veins (and the ring of the optic nerve) on the images better.

This is due to the background of the image becoming more uniform due to the reduction in contrast at the “white end”, while not affecting the veins that much as they are middle-grey / darker.

The reason that values of produce worse results is that the veins start being affected at that point as the veins themselves are not that dark compared to the full dynamic range available: this make me think that a system which darkens the image brightness without destroying information, and then applying a the Power Law with may achieve even better results.

### Summary

By far the best technique, and in fact the only technique to improve both the accuracy and true accuracy, was the Power Law with, and more specifically.

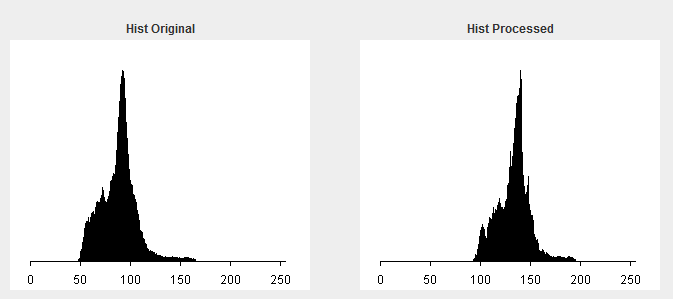
From the evidence above, success lies in successfully extracting information about the veins, which the Power Law helps with by making the whiter background more uniform so the veins “stand out” more. This hypothesis also matches perfectly with the fact that the other two methods, which greatly increased the dynamic range, did terribly.

From now on, contrast enhancement using the Power Law with will be enabled.

## Noise Reduction

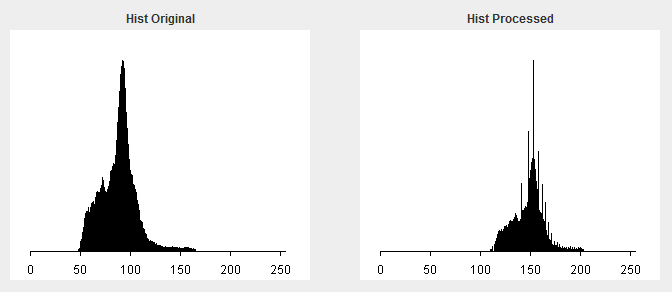
We have two types of noise reduction: Low Pass Filter and Median Filter. Just looking at the images, it seems the images do not contain either CCD or Salt and Pepper noise. The proof however will be seeing if the system gets more accurate when each is enabled.

## Low Pass Filter (LPF)



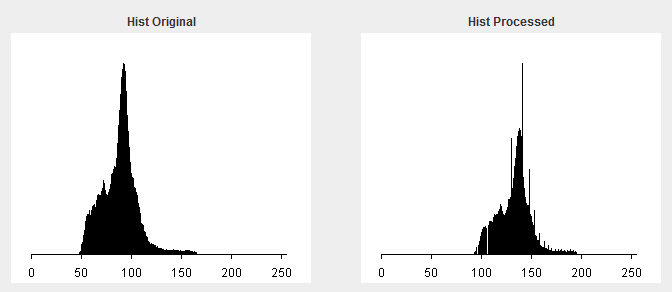
|  |  |  |
| --- | --- | --- |
| Technique | Accuracy (%) | True Accuracy (%) |
| Low Pass Filter Disabled | 88 | 92 |
| Low Pass Filter Enabled | 75 | 71 |

To show why this is reducing the accuracy by so much, here’s what the histogram looks like when just the contrast enhancement is applied without the LPF:



As you can see, the LPF appears to be “undoing” the contrast enhancement to an extent. It’ s “mistaking” the low contrast smudgy background for noise upon which it then creates more distinct light/dark patches, which increases its dynamic range.

## Median Filter



|  |  |  |
| --- | --- | --- |
| Technique | Accuracy (%) | True Accuracy (%) |
| Median Filter Disabled | 88 | 92 |
| Median Filter Enabled | 81 | 75 |

Once again, similar to the LPF, the Median Filter is undoing some of the work of the contrast enhancement. It is doing it to a lesser extent however, and so has a slightly higher accuracy than the LPF.

## Summary

Both of the noise reduction filters only hurt accuracy due to them undoing the work of the contrast enhancement to varying degrees, and there is not enough noise in the image whatsoever to justify either of them being enabled.

Interestingly, I decided to test applying the Noise Reduction first, before the contrast enhancement to see if the results would be better (since it wouldn’t be “undoing” the contrast enhancement) and got the exact same results as when applying the noise reduction afterwards.

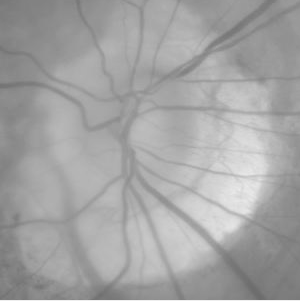
This shows that the noise reduction is wiping out subtle structures / information in the images which the Segmenter can take advantage of once the contrast enhancement has been applied.

Noise Reduction will not be enabled from this point onward.

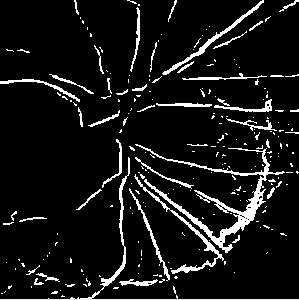
# Segmentation and Post Processing

Segmentation

Our segmentation process consists of edge extraction followed by automatic thresholding. We initially take the pre-processed image and perform edge extraction using the Sobel Mask.



As we can see edge extraction is key to extracting information about the veins and boundaries of the optic disc. We then perform automatic thresholding on the gradient magnitude image to give us our binary segmented image.



We have implemented a class SegmentedTesterAlpha which will test every value of α in the range -2.0 to 2.0 in increments of 0.1 to see if any of the values will produce a higher true accuracy than our default system alpha value of 1. On running this tester the results were as follows:

|  |  |  |
| --- | --- | --- |
| Technique | Accuracy (%) | True Accuracy (%) |
| The following all refer to the Automatic thresholding with “Technique” being the value α | | |
| -2.0 | 81 | 50 |
| -1.9 | 81 | 50 |
| -1.8 | 81 | 50 |
| -1.7 | 81 | 50 |
| -1.6 | 81 | 50 |
| -1.5 | 81 | 50 |
| -1.4 | 81 | 50 |
| -1.3 | 81 | 50 |
| -1.2 | 81 | 50 |
| -1.1 | 81 | 50 |
| -1.0 | 81 | 50 |
| -0.9 | 81 | 50 |
| -0.8 | 81 | 50 |
| -0.7 | 81 | 50 |
| -0.6 | 81 | 50 |
| -0.5 | 81 | 50 |
| -0.4 | 81 | 50 |
| -0.3 | 50 | 31 |
| -0.2 | 56 | 60 |
| -0.1 | 44 | 40 |
| 0.0 | 50 | 56 |
| 0.1 | 56 | 60 |
| 0.2 | 63 | 64 |
| 0.3 | 56 | 60 |
| 0.4 | 69 | 68 |
| 0.5 | 75 | 72 |
| 0.6 | 75 | 85 |
| 0.7 | 75 | 59 |
| 0.8 | 63 | 38 |
| 0.9 | 69 | 55 |
| 1.0 | 88 | 92 |
| 1.1 | 88 | 79 |
| 1.2 | 81 | 76 |
| 1.3 | 75 | 72 |
| 1.4 | 75 | 72 |
| 1.5 | 75 | 72 |
| 1.6 | 63 | 64 |
| 1.7 | 63 | 64 |
| 1.8 | 63 | 64 |
| 1.9 | 69 | 81 |
| 2.0 | 50 | 56 |

As we can see our default system value where α = 1 actually produces accuracy of 88% and a true accuracy of 92%. This therefore shall be the value we pass into our Segmenter from now on.

For proof that the Sobel Mask is best for performing edge extraction I can compare the results from using the **Prewitt** masks for edge extraction

|  |  |  |
| --- | --- | --- |
| Technique | Accuracy (%) | True Accuracy (%) |
| The following all refer to the Automatic thresholding with “Technique” being the value α | | |
| -2.0 | 81 | 50 |
| -1.9 | 81 | 50 |
| -1.8 | 81 | 50 |
| -1.7 | 81 | 50 |
| -1.6 | 81 | 50 |
| -1.5 | 81 | 50 |
| -1.4 | 81 | 50 |
| -1.3 | 81 | 50 |
| -1.2 | 81 | 50 |
| -1.1 | 81 | 50 |
| -1.0 | 81 | 50 |
| -0.9 | 81 | 50 |
| -0.8 | 81 | 50 |
| -0.7 | 81 | 50 |
| -0.6 | 81 | 50 |
| -0.5 | 81 | 50 |
| -0.4 | 81 | 50 |
| -0.3 | 81 | 50 |
| -0.2 | 50 | 56 |
| -0.1 | 38 | 49 |
| 0.0 | 44 | 53 |
| 0.1 | 50 | 44 |
| 0.2 | 63 | 64 |
| 0.3 | 50 | 44 |
| 0.4 | 63 | 51 |
| 0.5 | 81 | 76 |
| 0.6 | 69 | 55 |
| 0.7 | 75 | 72 |
| 0.8 | 75 | 85 |
| 0.9 | 69 | 68 |
| 1.0 | 81 | 88 |
| 1.1 | 81 | 88 |
| 1.2 | 69 | 68 |
| 1.3 | 56 | 60 |
| 1.4 | 75 | 72 |
| 1.5 | 81 | 88 |
| 1.6 | 81 | 88 |
| 1.7 | 81 | 88 |
| 1.8 | 69 | 68 |
| 1.9 | 69 | 68 |

Interestingly we can see that using the Prewitt masks for edge extraction as part of our segmentation is more effective for certain values of alpha versus using the Sobel masks with the same value of alpha. Maximum true accuracy using the Prewitt masks occurs when α = 1 as with using the Sobel mask, however true accuracy is slightly reduced (88% v 92%) We can therefore conclude the Sobel masks are the best for performing the edge extraction in our segmentation process.

Post-processing

We have implemented the post – processing techniques opening and closing from our methods for erosion and dilation. On testing we tried each of the possible post processing combinations and seen the following results.

|  |  |  |
| --- | --- | --- |
| Technique | Accuracy (%) | True Accuracy (%) |
| Post processing Disabled | 88 | 92 |
| Post processing Enabled Closing then Opening | 38 | 49 |
| Post processing Enabled Opening then Closing | 25 | 41 |
| Post processing Enabled Closing only | 69 | 68 |
| Post processing enabled Opening only | 25 | 41 |

We can see that the true accuracy drops quite drastically when any of the post processing techniques are applied.

# 

# 

# If we look at the post processed image of the Post processing(using closing only which produced the best true accuracy) we can see that some of the vascular detail has lost in comparison to the segemented image. Post processing should therefore not be enabled from now on.

# Feature Extraction

The idea behind obtaining useful features out of the segmented image is to extract the feature vectors using moments. Playing around with moments is the basis for the calculation of relevant features needed in analyzing the object.

In this program, the following features of the object will be taken into account: Area, Parameter, Compactness and Position of Centroid.

But first, we must make use of the moment formula to build a *moment* method, which will in turn be used to calculate the relevant feature vectors.

*private double moment(BufferedImage image, int k, int l)*

*{*

*// Get the raster of the image.*

*Raster r = image.getRaster();*

*// Get the moment.*

*double m = 0f;*

*for(int i = 0; i < r.getHeight(); i++)*

*for(int j = 0; j < r.getWidth(); j++)*

*m += Math.pow(i, k) \* Math.pow(j, l) \* r.getSample(j, i, 0) / 255;*

*return m;*

*}*

Area

In order to get the area vector, we just need to calculate *M00* using the *moment* method.

Perimeter

The relevant object’s perimeter is calculated by first eroding the thing, calculate the new object’s area, then calculate the difference between that and the original object’s area (Perimeter = Original – Erode).

Compactness

The whole point of getting area and perimeter is so we can use these values to calculate what is really needed: compactness; as it is the much more useful shape description for an Automated Image Recognisation System. Compactness is calculated by squaring the perimeter, then dividing it by the area.

Centroid Position

We get the X and Y coordinates of the centroid by calculating *M01* and *M10.*

# Classification

In our system, we have three functions that used to recognise the training images. These are SVM Function, Linear Discriminant Function, and Nearest Neighbour Function. When we put those Features above into account, here are the results of each function:

## SVM Function:

* Accuracy: 56.25%
* True Negative rate: 0.33
* True Positive rate: 0.62
* True Accuracy: 47.44%

## Linear Discriminant Function:

* Accuracy: 37.50%
* True Negative rate: 1.00
* True Positive rate: 0.23
* True Accuracy: 61.54%

## Nearest Neighbour Function:

* K = 1:
  + Accuracy: 62.50%
  + True Negative rate: 0.00
  + True Positive rate: 0.77
  + True Accuracy: 38.46%
* **K = 3:**
  + **Accuracy: 87.50%**
  + **True Negative rate: 1.00**
  + **True Positive rate: 0.85**
  + **True Accuracy: 92.31%**
* K =5:
  + Accuracy: 56.25%
  + True Negative rate: 0.67
  + True Positive rate: 0.54
  + True Accuracy: 60.26%

As the results, the Nearest Neighbour Function that using k equals to 3 is returned the highest accuracy rate. It’s because it can recognise the curve of the training images’ features, compare it to the SVM and the Linear Discriminant Function, those are just a straight line between the images.

## Summary:

For this data set of the images, the Nearest Neighbour Function is the best method that used to classify the object, however, for different sets; we can use different methods so that it can classify the object more accurately.

# Appendix

## System Instructions

To run the system against all training / test images in the Datasets folder, perform the following instructions:

1. Clone the repo off GitHub from <https://github.com/Bodomite/IIS-1415-G08.git>
2. Import the entire project from the “trunk” folder into eclipse.
3. Open up the IIS\_G08.java class.
4. Click “Run” and watch output in the console.