

**PATTERN RECOGNITION USING K-NEAREST NEIGHBORS (KNN)  
TECHNIQUE**

By

MOHD FARISFAIZ BIN CHE LAH

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Programme  
in Partial Fulfillment of the Requirements  
for the Degree  
Bachelor of Engineering (Hons)  
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas  
Bandar Seri Iskandar  
31750 Tronoh  
Perak Darul Ridzuan

© Copyright 2010  
by  
Mohd Farisfaiz Bin Che Lah, 2010

# **CERTIFICATION OF APPROVAL**

## **Pattern Recognition Using K-Nearest Neighbors (KNN) Technique**

by

Mohd Farisfaiz Bin Che Lah

A project dissertation submitted to the  
Electrical & Electronics Engineering Programme  
Universiti Teknologi PETRONAS  
in partial fulfillment of the requirement for the  
BACHELOR OF ENGINEERING (Hons)  
(ELECTRICAL & ELECTRONICS ENGINEERING)

Approved by,

---

(Dr Brahim Belhaouari Samir)  
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

June 2010

## **CERTIFICATION OF ORIGINALITY**

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

---

MOHD FARISFAIZ BIN CHE LAH

## **ABSTRACT**

The aim of this project is to develop a better image processing algorithm using K-Nearest Neighbors (KNN) technique. This project will apply supervised learning, thus require the author to gather training data and testing data consist of written alphabets. The algorithm will be built in the Matlab. Initially, the author needs to introduce the simpler version of feature extraction to replace the present extractor that more complex and time consuming. The basic idea of this type of feature extraction is to apply various shapes of curls on an image and the intersection of these two produce a stream vector. This step will be applied for entire images. For classification phase, there are two type of classifier that will be used which are Traditional K-Nearest Neighbors(KNN) and Clustering K-Nearest Neighbors(C-KNN).The result of each classifier will be analyze to get the optimal result or accuracy. Next, sample data need to be added in order to measure the robustness of algorithm. For the future, it is recommended that another version of input applied to the algorithm such as signature and thumbprint. If possible, the author also needs to build the hardware or application module that able to perform an accurate recognition and user-friendly.

## **ACKNOWLEDGEMENT**

First of all, I would like to express my greatest praise to Allah S.W.T after successfully finish my final year prject. With my sicere appreciation, I would like to give my utmost gratitude to my supervisor, Dr Brahim Belhaouari Samir for his detailed and constructive comments along with continuous assistance, support and motivation from beginning until the end of the project

My most appreciation also goes to him especially for the entire efforts and contribution of his ideas also guidance in Matlab programming. His expertise and wide knowledge have been great value for me.

Above all, I thank my family for always standing there besides me and continuously supportive throughout my life.

Last but not least, my regards and blessing also goes to all lecturers and colleagues who had supported me in any aspect while completing this project.

## TABLE OF CONTENTS

<b>CERTIFICATION OF APPROVAL</b>	. . . . .	<b>ii</b>
<b>CERTIFICATION OF ORIGINALITY</b>	. . . . .	<b>iii</b>
<b>ABSTRACT</b>	. . . . .	<b>iv</b>
<b>ACKNOWLEDGEMENT</b>	. . . . .	<b>v</b>
<b>LIST OF FIGURES</b>	. . . . .	<b>viii</b>
<b>LIST OF TABLES</b>	. . . . .	<b>ix</b>
<b>CHAPTER 1:</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Background of Study	1
	1.2 Problem statement	2
	1.3 Objectives	2
	1.4 Scope of Study	3
<b>CHAPTER 2:</b>	<b>LITERATURE REVIEW</b>	<b>4</b>
	2.1 Artificial Intelligence	4
	2.2 Character Recognition	4
	2.3 Feature Extraction	5
	2.3 K-Nearest Neighbors (KNN) method.	6
	2.4 K-Means Clustering	7
<b>CHAPTER 3:</b>	<b>METHODOLOGY</b>	<b>9</b>
	3.1 Procedure Identification	9
	3.1 Detailed Procedure	10
	3.2 Tools	17

<b>CHAPTER 4:</b>	<b>RESULTS AND DISCUSSION</b>	<b>18</b>
4.1	Simulation of Algorithm	18
4.2	Accuracy of Recognition	23
4.3	Analysis on Accuracy	24
4.4	Graphical User Interface (GUI)	30
4.4	Problem Encountered & Future Planning	31
<b>CHAPTER 5:</b>	<b>CONCLUSSION AND RECOMMENDATIONS</b>	<b>32</b>
5.1	Conclusion	33
5.2	Recommendation	34
<b>REFERENCES</b>		<b>35</b>
<b>APPENDICES</b>		<b>37</b>
	Appendix a: Gantt chart for fyp 1	38
	Appendix b: Gantt chart for fyp 2	38
	Appendix c: source code in matlab for logarithmic spiral equation.	39
	Appendix d: additional code for Fermat spiral.	40
	Appendix e: additional code for unidirectional galaxy	41
	Appendix f: additional code for bidirectional galaxy	42
	Appendix g: Source code in matlab for classification	43
	Appendix h: Source code for cluster KNN	55

## LIST OF FIGURES

Figure 1: Spiral Equation to Interpret Image . . . . .	5
Figure 2: Classification Using KNN Technique of Three Different Features	7
Figure 3: Classification Using Improved KNN/Cluster KNN . . . . .	8
Figure 4: Progress Flow for the Whole Project . . . . .	9
Figure 5: Training Samples . . . . .	10
Figure 6: Image Enhancement Using Morphological Operation . . . . .	11
Figure 7: Algorithm to Extract the Features and convert the Data into Vector Form	12
Figure 8: Algorithm To Classify The Image Using KNN Method . . . . .	14
Figure 9: K-Means Clustering Algorithm . . . . .	16
Figure 10: Feature Extraction Using Logarithmic Spiral before Parameter Tuning	19
Figure 11: Feature Extraction Using Logarithmic Spiral after Parameter Tuning	19
Figure 12: Comparison between Feature Extraction Using Fermat/Logarithmic	20
Figure 13: Initial Curl for Galaxy . . . . .	21
Figure 14: Angled-Shifted Curl for Galaxy . . . . .	21
Figure 15: Feature Extraction of Images Using Galaxy Equation . . . . .	22
Figure 16: Comparison between Unidirectional Galaxy with Bidirectional Galaxy	23
Figure 17: Graph of Average Accuracy For Each Methods . . . . .	28
Figure 18: Analysis on Accuracy When Varying K Value . . . . .	29
Figure 19: Graphical User Interface . . . . .	30
Figure 20: Proposed Module for Signature Recognition System . . . . .	31



## LIST OF TABLES

Table 1: List of Hardware and Software Used in This Project	17
Table 2: Result of Accuracy after Parameter Tuning for Logarithmic Spiral	24
Table 3: Result of Accuracy after Parameter Tuning for Multiple Spiral	25
Table 4: Result of Accuracy after Parameter Tuning for unidirectional Galaxy	
Equation	26
Table 5: Result of Accuracy after Parameter Tuning for bidirectional Galaxy	
Equation	27
Table 6: Average Accuracy Produced By Each Method for KNN and C-KNN	28
Table 7: Effect on Accuracy While Varying K Values	29

# CHAPTER 1

## INTRODUCTION

### 1.1 Background of Study

Pattern recognition is the process of interpreting raw data and taking an action based on the category of the pattern [4]. Description scheme is usually based on the availability of a set of patterns that already have been classified. This patterns called training set and this method can be categorize as supervised learning. Meanwhile, in unsupervised learning, system is not given an exact labeling of the patterns instead it establishes the classes based on statistical regularities of the pattern. [4]Regression and classification methods based on similarity of the input have been widely used in applications which involve large set of high-dimensional data.[1] One of the most preferable methods in cluster analysis is k-means clustering. The algorithm will use K-nearest neighbors (KNN) technique to classify each training samples into their respective cluster.

## **1.2 Problem Statement**

Many researchers have found that the KNN algorithm accomplishes very good performance in their experiments on different data sets. The traditional KNN text classification algorithm has three limitations which are calculation complexity, dependence on training set and no weight difference between samples. [9]. Besides, the current system of recognition, normally use Neural Network and Wavelet Transform to extract the feature from image. The author found that these approaches are more complex and time consuming. Hence, the author will attempt new approach in extracting the features which more simple and resulting in non-lengthy vectors to reduce the execution time. In addition, the existing systems also consume high cost to be implemented and too complicated to be handled by the user. Therefore, the author will try to come out with a system with a reasonable cost and less complexity

## **1.3 Objectives**

The objectives of this project are:

- To implement K-Nearest Neighbors(KNN) algorithm in classifying sample data into their respective cluster
- To introduce the new approach in extracting features from image
- To come out with an accurate recognition system which using K-Nearest Neighbors(KNN) methods

## **1.4 Scope of Study**

During early phase of this project, literature research has to be made in order to gather the information about the current classifier system and their weaknesses. Initially, the author will use alphabets as training samples. A lot of samples are needed in order to increase the efficiency of the recognition. Then, the images need to be scanned and an equation needs to be applied on them to extract the feature into vector form. Various approaches will be introduced to find out the best method for feature extraction and for the time being the simplest way is to use spiral curve. Later, KNN algorithm will be applied to the database in order to classify each image into their respective cluster. At the end of first phase of this project, author is expected to come out with an algorithm that able to classify the images accurately.

For the phase two (second semester), the author will continue to apply different set of samples to the algorithm and at the same time work on the accuracy of recognition. For instance, extended alphabets input will be attempted to the algorithm. Apart from that, more advance algorithm will be introduced in order to enhance the accuracy of recognition. One of them is using the Cluster K-Nearest Neighbors analysis. If possible, the author also needs to come out with a workable application that applies K-Nearest Neighbors (KNN) technique in its recognition process.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 Artificial Intelligence**

Artificial intelligence is the intelligence of machines and the branch of computer science which aims to create it .AI is defined as the study and design of intelligent agents where an intelligent agent is a system that perceives its environment and takes actions which maximize its chances of success. There are a few algorithms can be used for various kinds of AI purpose such as normal based models, normal mixture models, Bayesian estimates, histogram method, k-nearest-neighbors method, expansion by basic functions, kernel methods and many more.

#### **2.2 Character Recognition**

Committing words to paper in handwriting is a uniquely human act, performed daily by million people. Handwriting recognizer can be divided into two categories which are on-line and off-line. On-line recognizers run and receive the data as the user writes. Off-line recognizers run after the data have been collected and the image of handwriting for analysis is given to recognizer as bitmap. Thus, the speed of recognizer is not dependent on the writing speed of the user, but the speed dictated by the specifications of the system in words or character per second.[10].Classical methods in pattern recognition suffice for recognition of visual characters due to:[11]

- a) The 'same' characters differs in size, shapes and style for each person
- b) Visual characters are subject to spoilage due to noise

### 2.3 Feature Extraction

In order to get the binary value of image, an equation is required for extracting its feature. There are a few methods available such as Archimedean spiral, Logarithmic spiral and Fermat spiral. The logarithmic spiral can be distinguished from the spiral by the fact that the distances between the turnings of a logarithmic spiral increase in geometric progression, while in an Archimedean spiral these distances are constant [5]. In Archimedean, the path will be rotate away from fixed point with constant angular velocity. Equivalently, in polar coordinates  $(r, \theta)$  it can be described by the equation

$$r = a + b\theta \quad (2.1)$$

with real  $a$  and  $b$ . Changing the parameter  $a$  will turn the spiral, while  $b$  controls the distance between successive turnings. Conversely, for Logarithmic spiral, distances between the turnings keep increasing since its angular velocity keep changing. In parametric form, the curve can be defined as

$$x \begin{cases} x(t) = r \cos(t) = ae^{bt} \cos(t) \\ y(t) = r \sin(t) = ae^{bt} \sin(t) \end{cases} \quad (2.2)$$

Other equation that can be used is Fermat spiral whereby two spirals with different direction is applied at the same time. The equations are

$$r^2 = a^2\theta \quad (2.3)$$

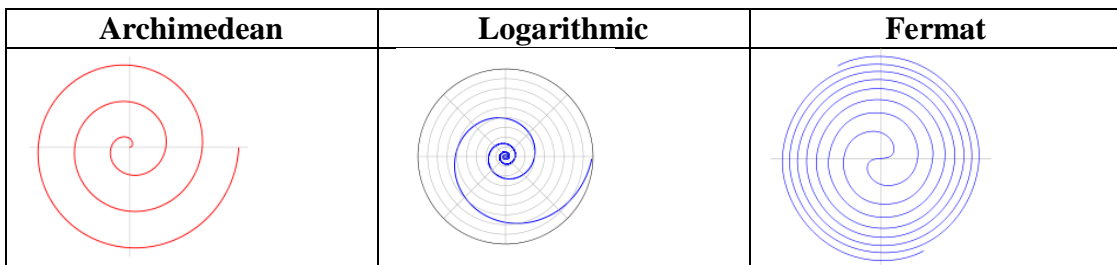


Figure 1: Spiral Equation to Interpret Image

### 2.3 K-Nearest Neighbors(KNN) Method

In application, usually we have a description of a texture sample and we want to find which element of database matches that sample. Hence, classification will be needed to associate the appropriate class label (type of texture) with the test sample by using the measurements that describe it. One way to make the association is by finding the member of the class with measurement which differs by the latest amount from the test sample's measurement. [8]

For this project, the author will only focus on using K-Nearest Neighbors (KNN) as a method to do pattern recognition. Nearest neighbor is a supervised learning algorithm where the results are classified based on majority of K-nearest neighbor category. The purpose of this algorithm is to classify a new object based on attributes and training samples. KNN technique only requires:

- ❖ An integer k
- ❖ A set of labeled examples (training data)
- ❖ A metric to measure “closeness”

For example, in the figure 2, there are three types of features available which are  $\omega_1, \omega_2, \omega_3$ . We need to identify the class for object  $X_u$ . The classification is using majority vote among the classification of the K objects. [1] Usually Euclidean distance is used as the distance metric yet this is only applicable to continuous variables [3].

$$D = \sqrt{(X_i - Y_i)^2 + (X_j - Y_j)^2} \quad (2.4)$$

In this case the number of neighbors obtained are 5. Out of them, four are belong to  $\omega_1$ , Therefore,  $X_u$  is belong to  $\omega_1$  cluster.

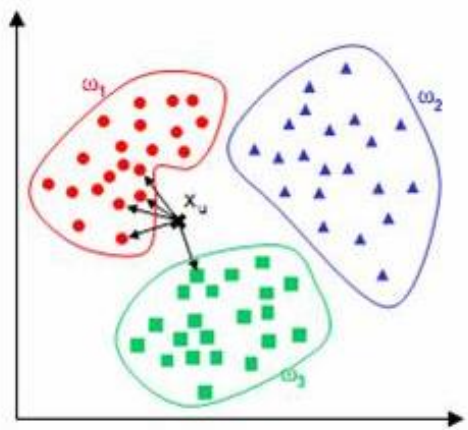


Figure 2: Classification Using KNN Technique of Three Different Features

The best choice of value  $k$  depends upon the data. Normally, larger values of  $k$  reduce the effect of noise on the classification. [2] Training sample is considered as nearest neighbors if the distance of this training sample to the query instance is less than or equal to the  $K$ -th smallest distance. In other words, the distance of all training samples will be sorted to the query instance and determine the  $K$ -th minimum distance.[4]

## 2.4 K-Means Clustering

The traditional approach of KNN has three limitations that lead to less accurate recognition:[9]

- a) High calculation complexity-less number of samples make classifier not optimal while huge number of samples consume more time
- b) Dependency on training set-depend on training data excessively. It needs recalculation even if slight changes in training samples
- c) No weight difference between samples-all training samples are treated equally without no difference between small number of data with huge number of data



Therefore, wide varieties of methods have been proposed to deal with this problem. One of them is by using K-means clustering approach. K-means clustering algorithm first was introduced by J. MacQueen (1967) and then by J. A. Hartigan and M.A. Wong around 1975 [1]. Generally, K-means clustering is an algorithm to classify or to group objects based on attributes or features into k number of class. The clustering is evaluated by determining the features that have some similarities between each other and represent it by central point.

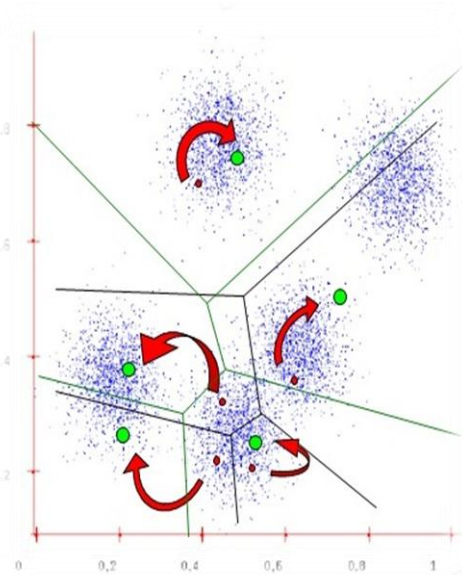


Figure 3: Classification Using improved KNN/Cluster KNN

In the example in figure 3, we divided objects into 5 different groups. Each group will consist of a random number of data which is relatively close to each other. Each subclass will be represented by one data or more depending on the number of subclasses. Then, a classification algorithm (nearest neighbors) will be used to classify the representative data. By applying this method, we can reduce the classification time and classification will be more efficient in overlapping areas since it takes more consideration to samples of training data that are less frequent.

# CHAPTER 3

## METHODOLOGY

### 3.1 Procedure Identification

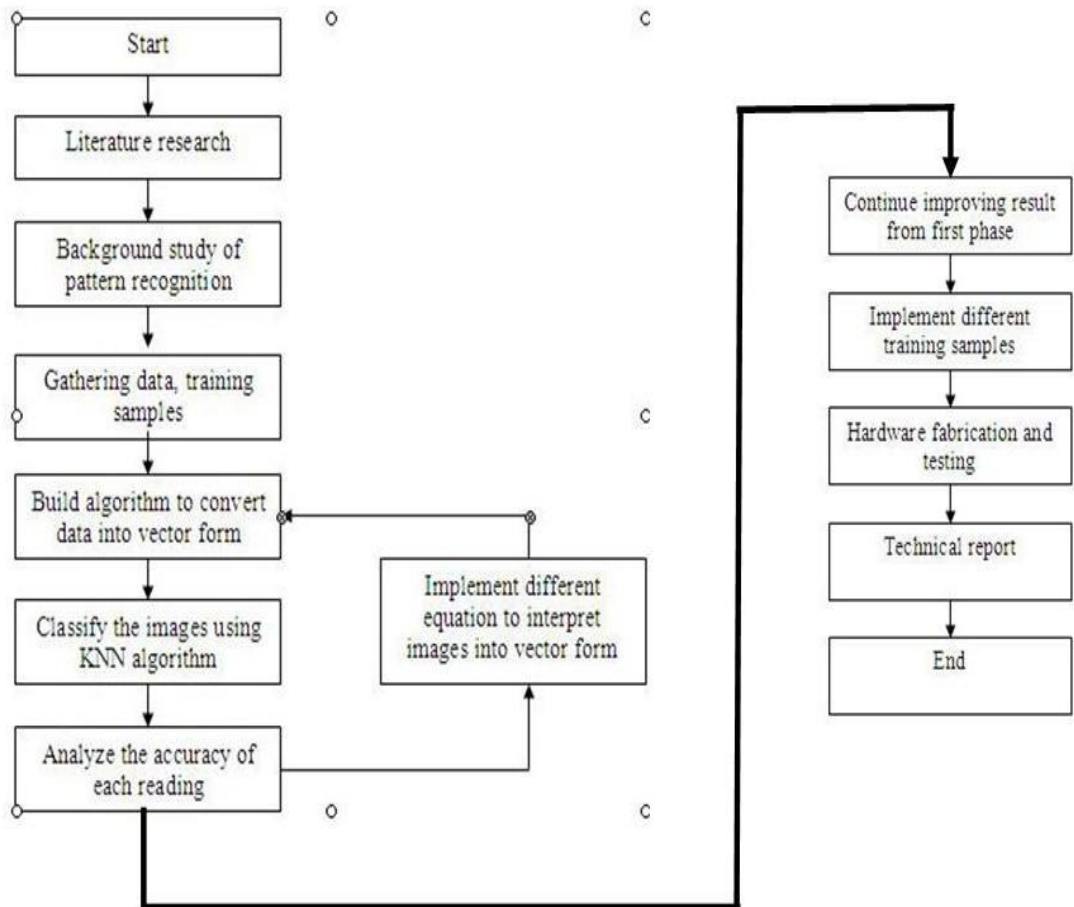


Figure 4: Progress Flow for the Whole Project

### 3.2 Detailed of The Procedure

Throughout this project, there are some procedures that will be followed. This is to ensure that the project can be accomplished within the given timeframe.

#### *3.2.1 Data Research and Gathering*

The samples that are going to be applied as the input in the algorithm will be gathered from various sources. For the initial stage of this project, the input will be the written alphabets from different people.. The number of training samples of each alphabet will be increased from if necessary to observe the robustness of the algorithm. The author also needs to seek the existing recognition system which related to this project in order to set the bench-mark in term of the accuracy. For the time being, the highest accuracy obtained is 99.21% to 99.95% by using Neural-Network method.[7]

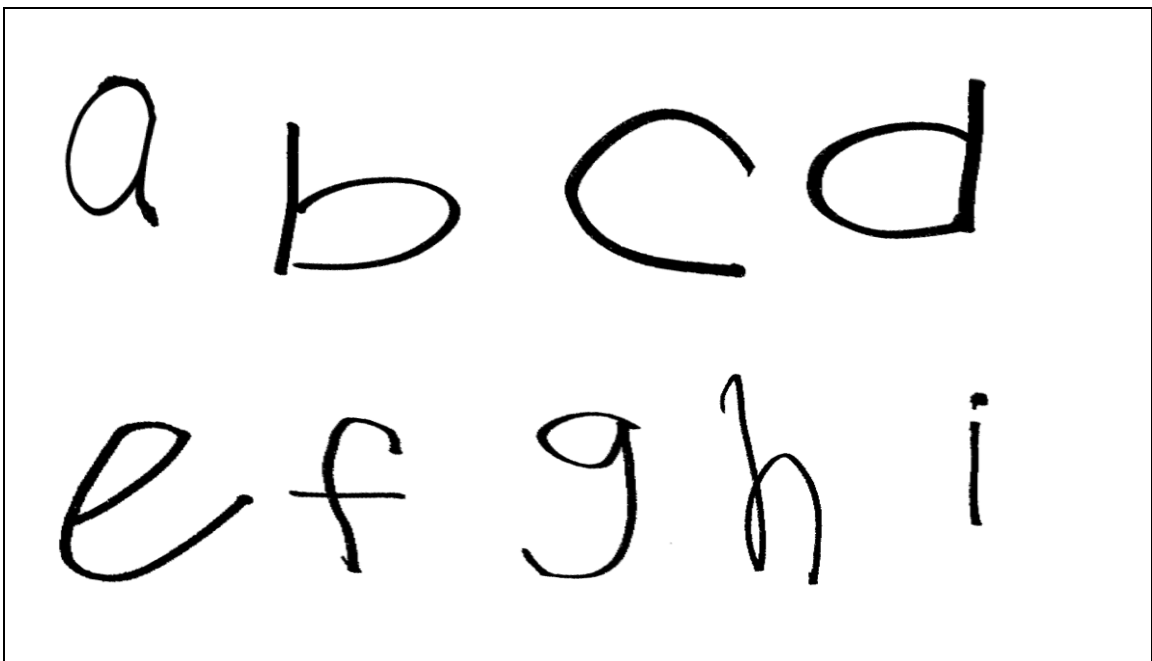


Figure 5: training samples

### 3.2.2. Image Enhancement

The raw image that been taken from various person may contain certain noise and low quality. Therefore, these images must be enhanced first before being used in the algorithm. One way to improve the quality of image is by using morphological approach. Specific morphological operation will be applied if we use this feature. Its operation can be any of these values: 'bothat', 'bridge', 'erode', 'dilate' etc. After implement several test using all the operation's value, the author find out that 'erode' has given he best result on enhancing the image. This operation (erode) will perform dilation using the structuring elements ones (3).

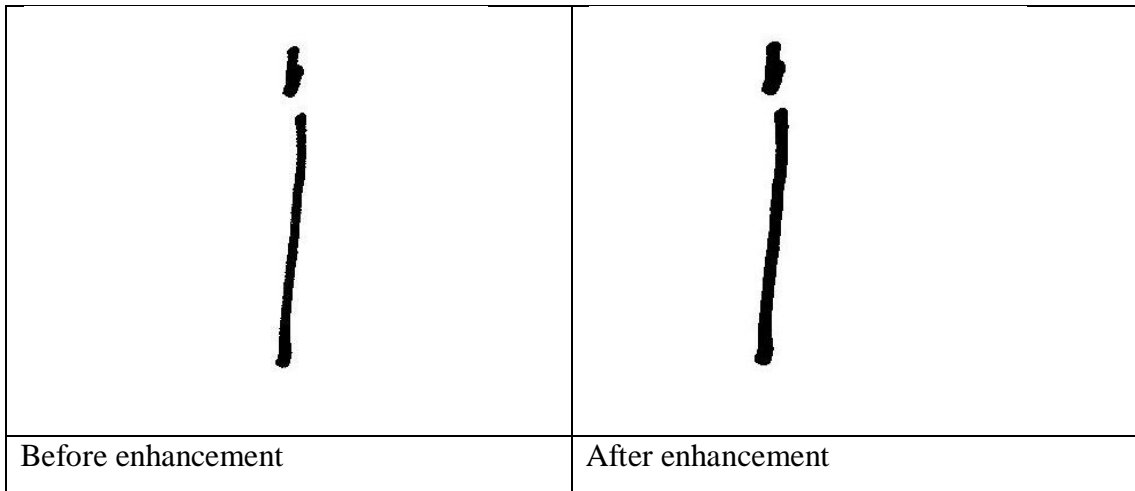


Figure 6: Image Enhancement Using Morphological Operation

From figure 6, it shows that there is some improvement occur on the image whereby the colour depth is getting thicker hence producing clear image. These enhanced images later will be used as the input in algorithm.

### 3.2.3 Create the Algorithm in Matlab

There are two types of algorithm that are going to be created. The first algorithm will extract feature from training samples and saved in vector form.(Refer Appendix C)

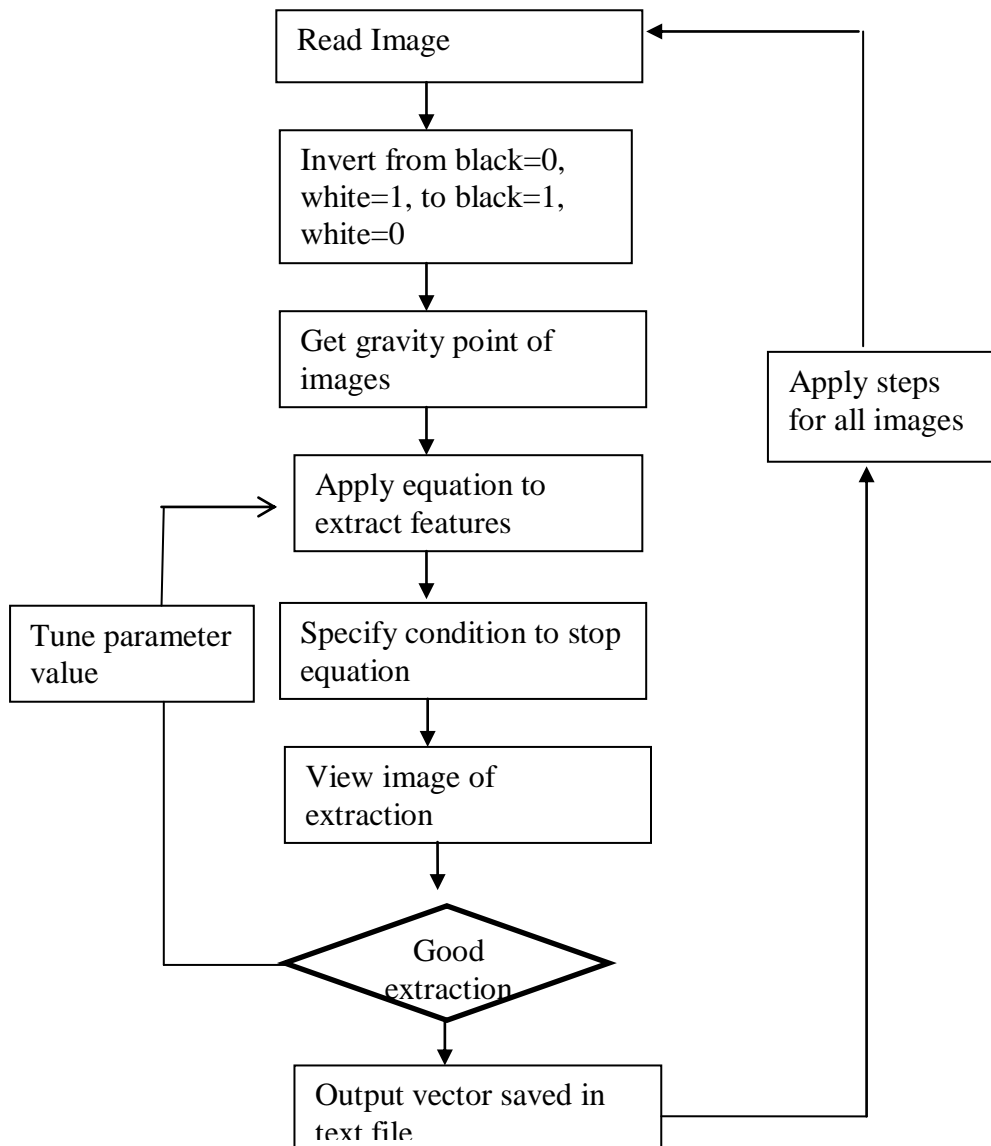


Figure 7: Algorithm to Extract the Features and Convert the Data into Vector Form

The input images need to be saved in monochrome format and their dimension must be standardized. After that, the image will be read as the input in the matlab by using imread command. Result of this command will give output of the image in 1 and 0 formats. In normal condition zero will represent black while white will represent 1. conversely, normal condition should black to be 1 and white to be zero. Hence, this value will be subtracted with 1 and put in absolute form.. The next step is to find the gravity point of the image. The equation to find gravity point is

$$Gravity\ point(O_G) = \frac{\sum_i \sum_j X_{i,j}}{n.m} \quad (3.1)$$

This gravity point will be the starting point for feature extraction. One of the methods that can be used for feature extraction is spiral equation. The formula are:

$$\begin{cases} X = X_g + V_0 \Delta t \cos(\omega \Delta t) \\ Y = Y_g + V_0 \Delta t \sin(\omega \Delta t) \end{cases} \quad (3.2)$$

The equation will produce an error if no specific condition made to terminate the operation. The equation will break if the curl exceeds the outermost radius of the image or the innermost radius of the image (negative value) .The intersection between image and equation will produce a stream of vector. This vector will be written in text file. This file we will use later in classification phase. All the images will be gone through the same step. At the end of each step, vector need to be saved with different name.

The second algorithm is to classify the alphabets into their respective cluster. Generally, the program will compute the distance of each vector/image and sort them into their cluster (Refer Appendix G).

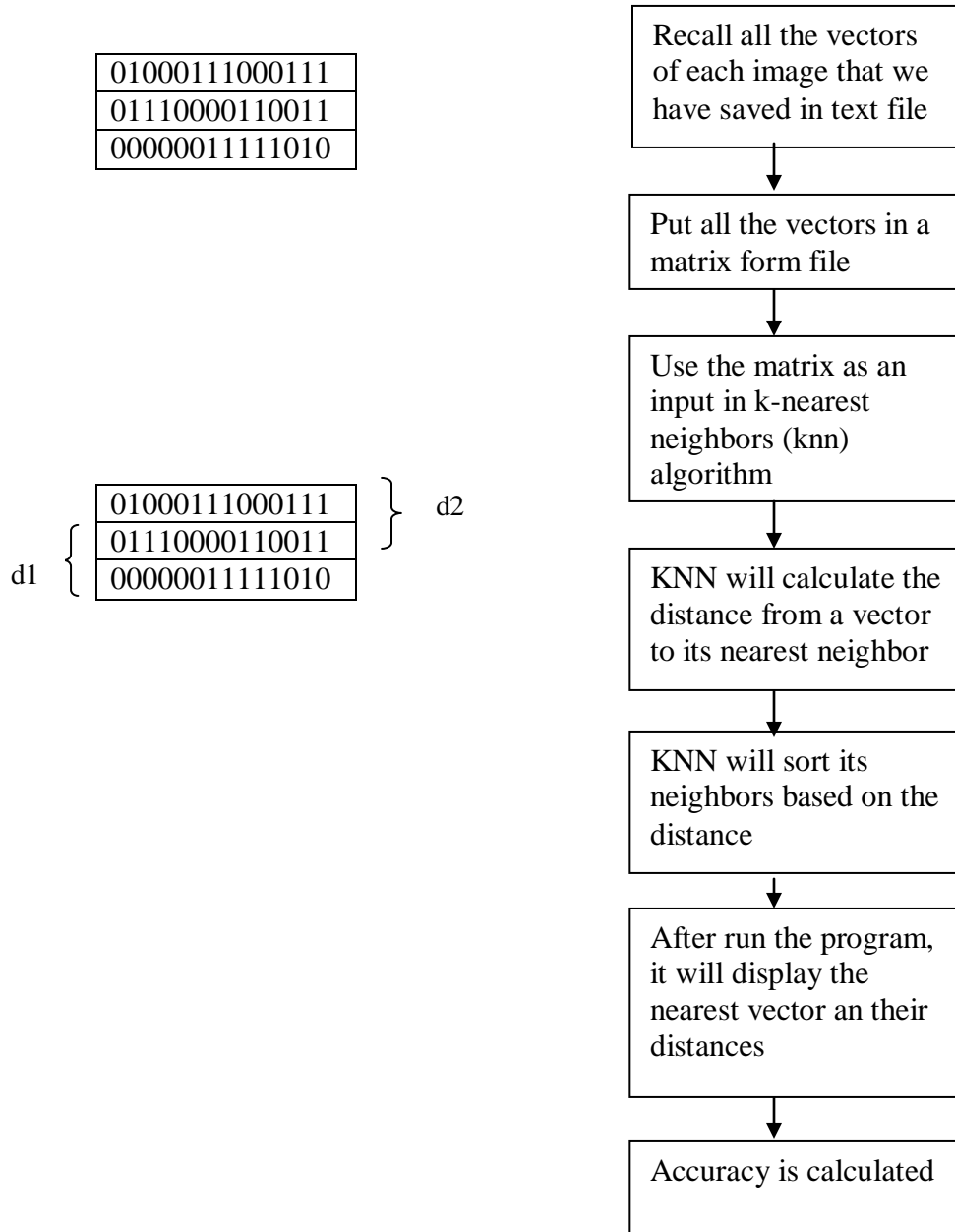


Figure 8: Algorithm to Classify the Image Using KNN Method

In this program, classification will involve the usage of KNN method. First, all the vectors from previous algorithm will be recalled and assign a unique name for each of it.. After that, all the vectors will be arrange together to produce one matrix .This matrix later will be as an input in KNN algorithm. In KNN algorithm, the Euclidean distance of a vector to another will be calculated. Then, the neighbor for each vector will be sorted based on its nearest distance. The user will specify the value of K which is the values of neighbor that they expect to get. After the program is run, it will display the neighbors and distance for each vector. From that, we can calculate the accuracy by determining whether the neighbors are within the expected range or not.

In order to improve the accuracy, parameter values will be tuned, thus the images can be extracted better. Various equations will also being attempted to analyze the best feature extraction method. The equations are linear equation, spiral equation and galaxy equation

#### *3.2.4 Cluster K-Nearest neighbors(C-KNN)*

Generally, his method aims to classify or group objects based on attributes or features into k number of group. The grouping is done by minimizing the sum of squares of distances between data and corresponding cluster centroid. In order to reduce the time taken for classification, training data must be clustered into subclasses whereby each subclass will be represented by one data. Each subclass contains a random number of data which relatively close to each other. Then, the representative data will be classified using nearest neighbors approach.



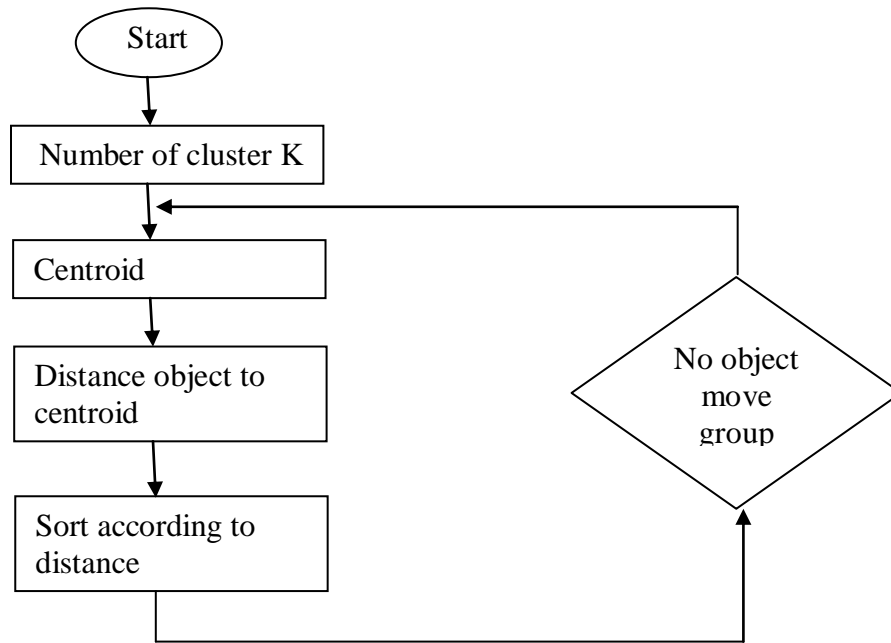


Figure 9: K-means clustering algorithm

The flow of how k-means clustering is being applied is as follow:

1. Determine the value of k(cluster number)
2. Put initial partition that classifies the data into k clusters. The training samples may be assigned randomly or systematically by taking the first k training sample as single element clusters while the remaining samples assigned to cluster with nearest neighbors. After each assignment, the centroid will be recalculated.
3. Take each sample in sequence and compute its distance from the centroid of each cluster. If the samples is not within the cluster of closest centroid,the samples will switch to right cluster. The centroid will be computed once again.
4. Step 3 will be repeated for a few times until convergence is achieved

### 3.2.5 Apply Different Set Of Inputs To Algorithm

In order to increase robustness of classifier, training samples will be added. Besides, a different set of input will also be attempted. By doing this, compatibility of the algorithm with various inputs will be known. The inputs can be signature, object image, human voice and others.

### 3.2.6 Hardware/Application Fabrication And Testing

In the final phase of this project, author supposes to come out with hardware or application that related to real life application. It can be biometric devices to detect fingerprint, iris or palm. The aim to come out with a patentable hardware is to introduce low cost system without neglecting its accuracy.

## 3.3 Tools

Table 1: List of Hardware and Software Used In This Project

<b>Software</b>	<b>Hardware</b>
<b>Matlab/programming software</b> ❖ The programming tool is for algorithm programming purposes.	<b>Scanner/capture device</b> ❖ These devices will be used to get the inputs form the user for recognition purposes

## CHAPTER 4

### RESULTS AND DISCUSSION

#### 4.1 Simulation of algorithm

From the initial phase, the highest accuracy achieved is 82.29% by using dual spiral extraction. The total samples used are written alphabets of a,b,c and d. The number of k is assigned as 3 means sample is accurate if its majority (2/3) of its neighbors are within the desired range. For the next stage, the author has increased the alphabets from a until i. Each alphabet consist of 15 training samples which resulting to total number of sample to 135 images. By adding the training samples, the robustness of algorithm against data size can be analyzed. The feature space of all images will be extracted using various approaches to find out the best solution to produce vector data.

##### *4.1.1 Logarithmic Spiral equation*

The equation involves in this method are:

$$\begin{cases} X = X_g + V_0 \Delta t \cos(\omega \Delta t). \\ Y = Y_g + V_0 \Delta t \sin(\omega \Delta t). \end{cases} \quad V_0 \leq \omega \quad (4.1)$$

$X_g$  and  $Y_g$  are the starting point of the equation or the gravity point of the images. The manipulated variables involve here are  $V_0, \omega$  and  $\Delta$ . An efficient variables should use small  $V_0$  and high  $\omega$  and  $\Delta$ . The initial parameter value is  $V_0 = 7, \Delta = 0.01, \omega = 0.1$

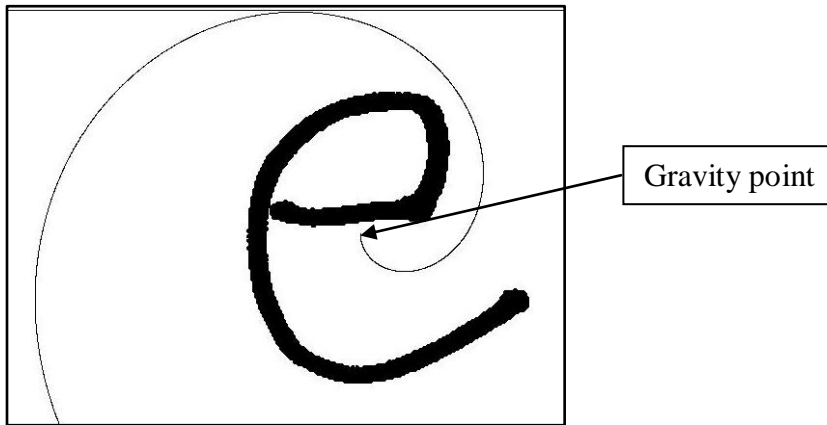


Figure 10: Feature Extraction Using Logarithmic Spiral before Parameter Tuning

From figure 10, we can see that the curl of spiral not good enough since the expansion of curl is too big. Hence, it might cause miss extraction which later will reduce the accuracy of recognition. Therefore, the next step is to tune the parameter in order to improve the formation of spiral

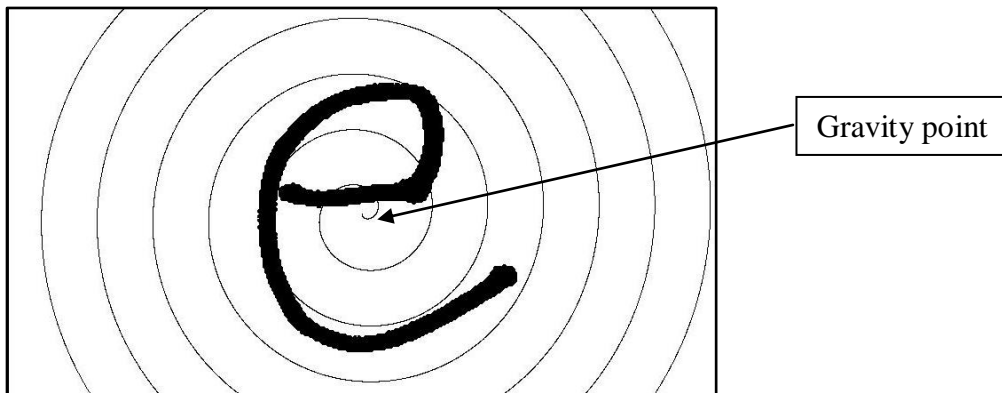


Figure 11: Feature Extraction Using Logarithmic Spiral after Parameter Tuning

The consideration made while tune the parameter is  $\omega$  must bigger than  $V_0$ . In this case, the bigger value of  $\omega$  will increase the angular velocity thus increase number of curls. Meanwhile smaller number of  $V_0$  will reduce the speed so compress the distance between each curls.

#### 4.1.2 Fermat Spiral equation

Instead of using single equation, the author attempts to use two equations with different direction. The equation is called Fermat spiral. The second equations implemented are:

$$\begin{cases} X = X_g + V_0 \Delta t \cos(\omega \Delta t) \\ Y = Y_g + V_0 \Delta t \sin(\omega \Delta t) \\ X = X_g + V_0 \Delta t \cos(\omega \Delta t) \\ Y = Y_g - V_0 \Delta t \sin(\omega \Delta t) \end{cases} \quad V_0 \leq \omega \quad (4.2)$$

Then vector of each equation will be compiled together in an array (Vect\_f=[vect\_f<sub>1</sub> vect\_f<sub>2</sub>]).The parameter values used are still the same. The accuracy obtained is slightly less than single spiral.

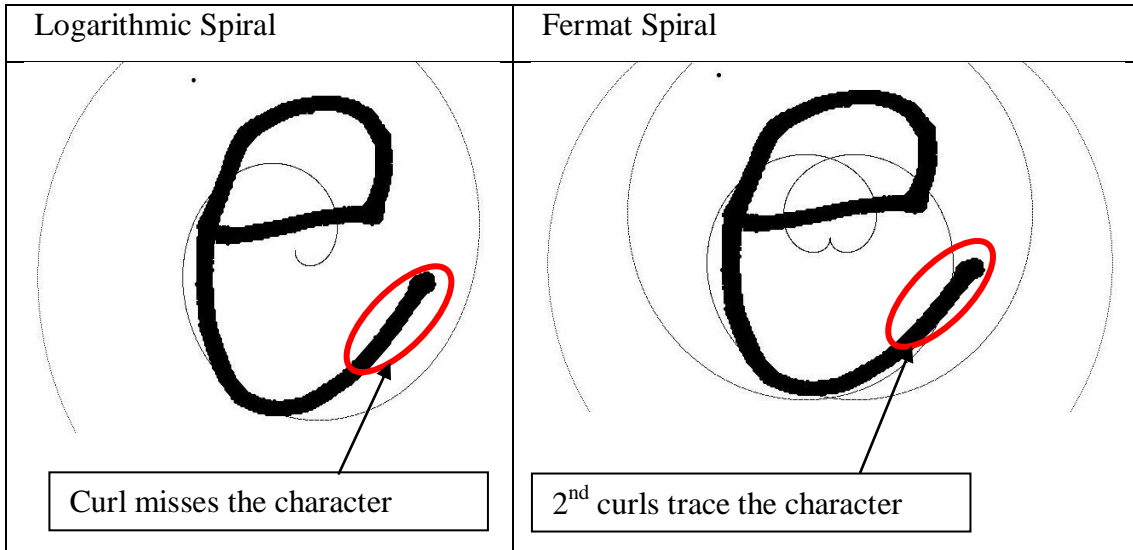


Figure 12: Comparison between Feature extraction using Fermat or logarithmic spiral

From figure 12, it is proven that the implementation of spiral in both direction will avoid miss-tracing of character. To make the intersection between image and spiral, parameter values must be tuned.

### 4.1.3 Galaxy Equation

In galaxy, it still uses the spiral equations which are:

$$\begin{cases} X = X_g + V_0 \Delta t \cos(\omega \Delta t + \theta). \\ Y = Y_g + V_0 \Delta t \sin(\omega \Delta t + \theta). \end{cases} \quad V_0 \gg \omega \quad (4.3)$$

$X_g$  and  $Y_g$  are the starting point of the equation or the gravity point of the images. The manipulated variables involve here are  $V_0, \omega$  and  $\Delta$ . If we give the value of  $V_0 \gg \omega$ , the equation will produce a curve. To make it become a galaxy, curve need to be shifted by certain angle. Let say if we want to have ten curves in the galaxy, the angle shift need to be made by increment of  $\pi/5$  until it reach  $9\pi/5$ .

The vector produce by interpretation of each curves will be combined in a vector ( $\text{Vect}_f = [\text{vect}_{f_1} \dots \text{vect}_{f_{10}}]$ ) (Refer Appendix E). In order to find the best formation of galaxy that should be mapped to images, several simulation must be done to find out right number of curl for the galaxy.

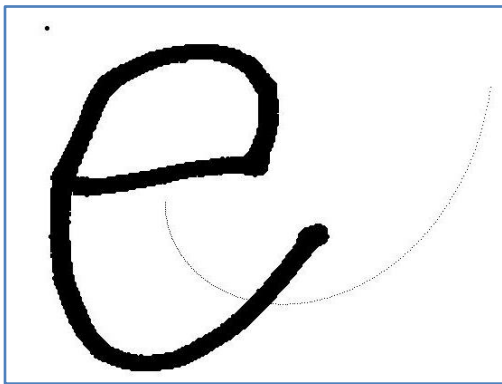


Figure 13: Initial Curl For Galaxy

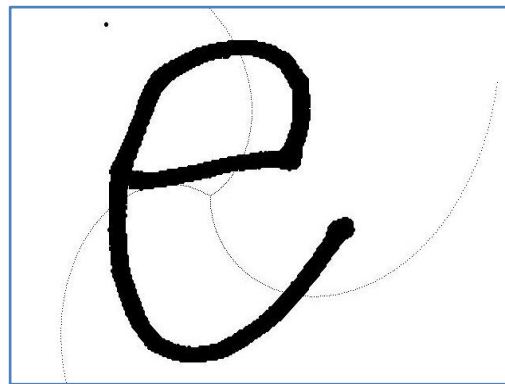


Figure 14: Angled -Shifted Curl For

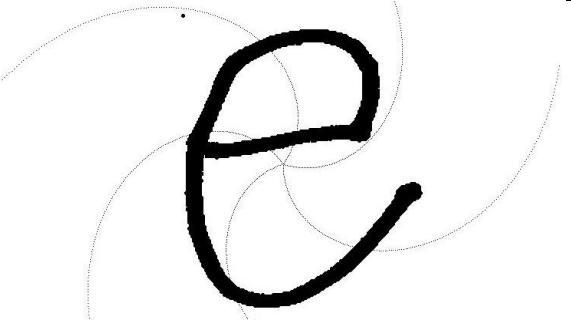
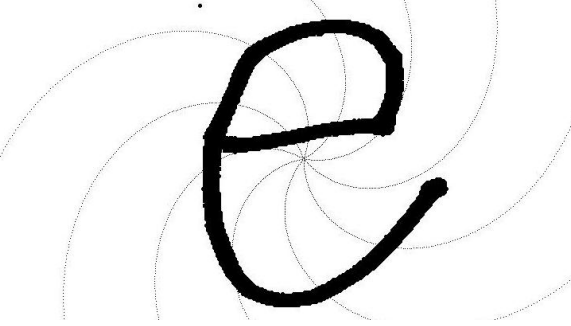
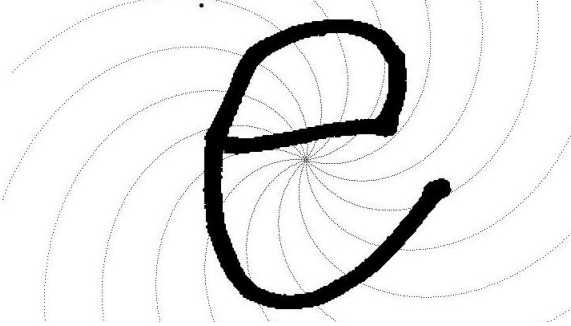
	<p>Number of curls:5  Parameter:<math>V_0=500</math>; <math>\Delta= 0.01</math>; <math>\omega = 1</math>  Accuracy:20.63%</p>
	<p>Number of curls:9  Parameter:<math>V_0=500</math>; <math>\Delta= 0.01</math>; <math>\omega = 1</math>  Accuracy:28.57%</p>
	<p>Number of curls:15  Parameter:<math>V_0=500</math>; <math>\Delta= 0.01</math>; <math>\omega = 1</math>  Accuracy:25.40%</p>

Figure 15:Feature Extraction Of Images Using Galaxy Equation

Based on the result in figure 15, we can see that the accuracy will reduce if the number of curls is too many. Hence, it is preferable that the number of curls is less so that the error during classification can be reduced.

#### 4.1.4 Dual Galaxy Equation

Instead of using unidirectional galaxy, it is better to use bidirectional galaxy means applied galaxy in forward and backward direction. It is because in one directional galaxy, there is high possibility that the some character miss interpreted by the curl.

The equation of dual galaxy is as follow:

$$\begin{cases} X = X_g + V_0 \Delta t \cos(\omega \Delta t + \theta) \\ Y = Y_g + V_0 \Delta t \sin(\omega \Delta t + \theta) \end{cases} \quad \begin{cases} X = X_g + V_0 \Delta t \cos(\omega \Delta t + \theta) \\ Y = Y_g - V_0 \Delta t \sin(\omega \Delta t + \theta) \end{cases} \quad V_0 \gg \omega \quad (4.4)$$

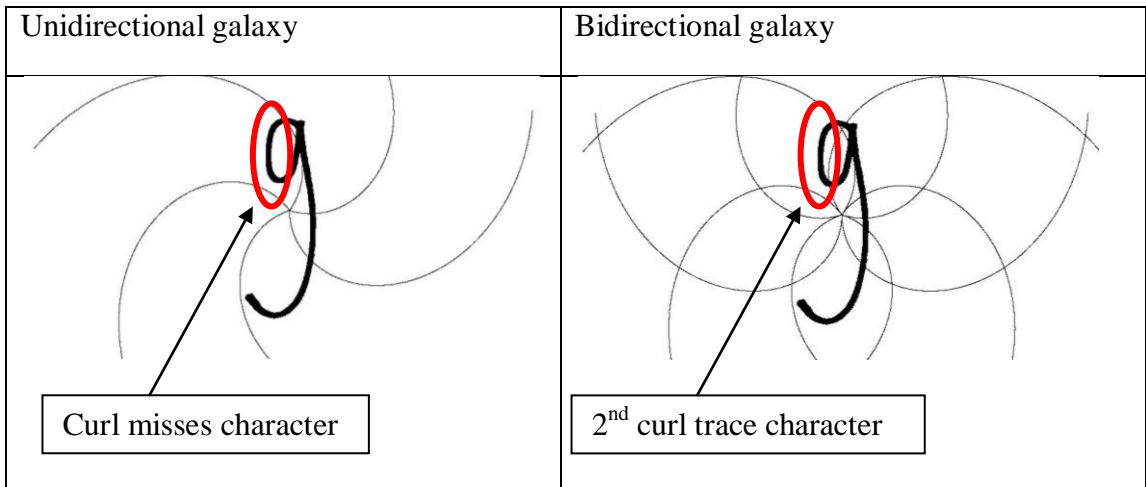


Figure 16: Comparison Between Unidirectional Galaxy With Bidirectional Galaxy

#### 4.2 Accuracy of Recognition

In the initial stage, only 60 samples used consist of 15 each alphabets (a,b,c,d).The highest accuracy achieved previously was 89.29%.For the second phase ,the training samples is added until 135 images. All the methods will be attempted in order to find the most efficient technique to extract the features. Initially, the value of k defined as 3.Thus, if 2 out of 3 neighbors within the expected neighbor's range, recognition is considered successful.



### 4.3 Analysis On Accuracy

From 15 samples of each class, 8 of the images were used for training and other 7 will be used as testing data. In classification phase, two type of classifier will be used which are Euclidean KNN and cluster KNN .For each attempt, parameter value will be tuned to improve the accuracy. Below is the result of analysis done for each method:

Table 2: Result of Accuracy after Parameter Tuning For Logarithmic Spiral

method	parameter			Accuracy (%)		Length
	V0	del	$\omega$	KNN	C-KNN	
Logarithmic spiral	1	0.4	4	77.78	86.7	580
	0.1	0.4	5	76.19	91.1	5469
	2	0.4	4	79.37	93.3	307
	0.1	0.4	7	77.78	91.1	5455
	0.1	0.4	10	73.02	95.6	5487
	9	0.02	3.5	73.1	93.3	1253
	9	0.02	2.5	74.6	93.3	1253
	9	0.05	2.5	74.6	91.1	501
	9	0.01	2.5	73.02	88.9	2505
	9	0.03	2.5	76.2	91.1	835
	9	0.02	2.1	61.9	88.9	1266
	9	0.02	2.7	65.08	88.9	1274
	7	0.02	2.6	76.2	93	1573
	7	0.02	2.5	80.9	91	1627
	9.5	0.02	2.5	69.8	88.9	1190
	9.5	0.02	3.5	76.2	93.3	1167
	9.5	0.03	3.9	76	91.1	778
	9.5	0.03	4.5	73	93.3	790
	9	0.001	2.5	53.97	99	25043
	Highest Accuracy				80.9	99

Table 3: Result of Accuracy after Parameter Tuning For Multiple Spiral

method	parameter			Accuracy (%)		Length
	V0	del	$\omega$	KNN	C-KNN	
Multiple spiral	0.1	0.4	5	69.84	88.9	10947
	2	0.4	4	76.19	88.9	614
	0.1	0.4	7	69.84	88.9	10917
	0.1	0.4	10	73.02	95.6	10983
	9	0.02	3.5	66.67	91.1	2505
	9	0.02	2.5	69.84	91.1	2505
	9	0.05	2.5	66.7	88.9	1001
	9	0.01	2.5	69.8	91.1	5009
	9	0.03	2.5	69.8	91.1	1669
	9	0.02	2.1	65.05	91.1	2531
	9	0.02	2.7	61.9	93.3	2547
	7	0.02	2.6	68.25	93.3	3197
	7	0.02	2.5	68.25	91.1	3253
	9.5	0.02	2.5	66.7	88.9	2379
	9.5	0.02	3.5	65.1	91.1	2333
	9.5	0.03	3.9	66.7	93.3	1555
	9.5	0.03	4.5	65.1	93.3	1579
	Highest Accuracy				76.19	95.6

For single and double spiral approach, it shown that it cluster KNN method has significantly improve accuracy of recognition. Generally the accuracy for each execution is above 80% with minimal elapsed time. It also concluded that there will be better if the value of  $\Delta$  is smaller which mean sampling occur more rapidly during feature extraction.

Table 4: Result Of Accuracy After Parameter Tuning For Unidirectional Galaxy

method	parameter			Accuracy (%)		Length
	V0	del	$\omega$	KNN	C-KNN	
Galaxy (33 curls)	200	0.05	2	30.16	60	1096
	80	0.05	0.2	25.4	56	2740
	80	0.5	0.2	14.29	58	292
Galaxy (20 curls)	200	0.05	2	22.2	64	674
	80	0.05	0.2	23.8	62	1689
	80	0.5	0.2	9.5	44	182
Galaxy (10 curls)	200	0.05	2	42.9	75.6	352
	80	0.05	0.2	36.5	60	894
	80	0.5	0.2	15.9	49	97
Galaxy (5 curls)	200	0.05	2	38.1	69	193
	80	0.05	0.2	34.9	64	508
	80	0.5	0.2	15.8	42	57
	200	0.01	0.2	28.6	62	958
	200	0.5	1	30.2	62	26
Highest Accuracy				42.9	75.6	

From table 4, it shows that the lesser number of curls, the higher the accuracy. Furthermore, as the value of  $\Delta$  become smaller, accuracy also increase. However, in general galaxy approach still not give a satisfactory result in term of accuracy compare to spiral method.

Hence, in order to improve the accuracy, simulation using bidirectional galaxy has been attempted.

Table 5: Result Of Accuracy After Parameter Tuning For Bidirectional Galaxy

method	parameter			Accuracy (%)		Length
	V0	del	$\omega$	KNN	C-KNN	
Galaxy (33 curls)	200	0.05	2	25.39	53.3	2203
	80	0.05	0.2	22.2	46.7	5549
	80	0.5	0.2	7.93	42.2	591
Galaxy (20 curls)	200	0.05	2	25.39	51.1	1358
	80	0.05	0.2	30.16	51.1	3429
	80	0.5	0.2	26.48	33.3	368
Galaxy (10 curls)	200	0.05	2	34.92	60	713
	80	0.05	0.2	31.75	53.3	1835
	80	0.5	0.2	22.22	35.6	196
Galaxy (5 curls)	200	0.05	2	41.27	55.6	391
	80	0.05	0.2	36.5	53.3	1062
	80	0.5	0.2	30.16	44.4	114
	200	0.01	0.2	47.62	57.8	1877
	200	0.5	1	33.3	46.7	51
Highest Accuracy				47.62	60	

From table 5, we can conclude that the accuracy slightly increase by using bidirectional approach. It means that some characters that not traced in unidirectional can be overcome if galaxy applied in both directions. However, classification using clusters analysis still unable to give high accuracy as high as spiral method in each trial.

After attempt all the methods, the author has came out wit the best solution for feature extraction. It can be proved by performing an analysis on average accuracy generated by every method.

Table 6: Average Accuracy Produced By Each Method for KNN and C-KNN

methods	Average accuracy	
	KNN	C-KNN
Logarithmic spiral	73.09	91.73
Fermat Spiral	68.12	91.23
Unidirectional galaxy	26.3	59.11
Bidirectional galaxy	29.66	48.9

### Average Accuracy For Each Method

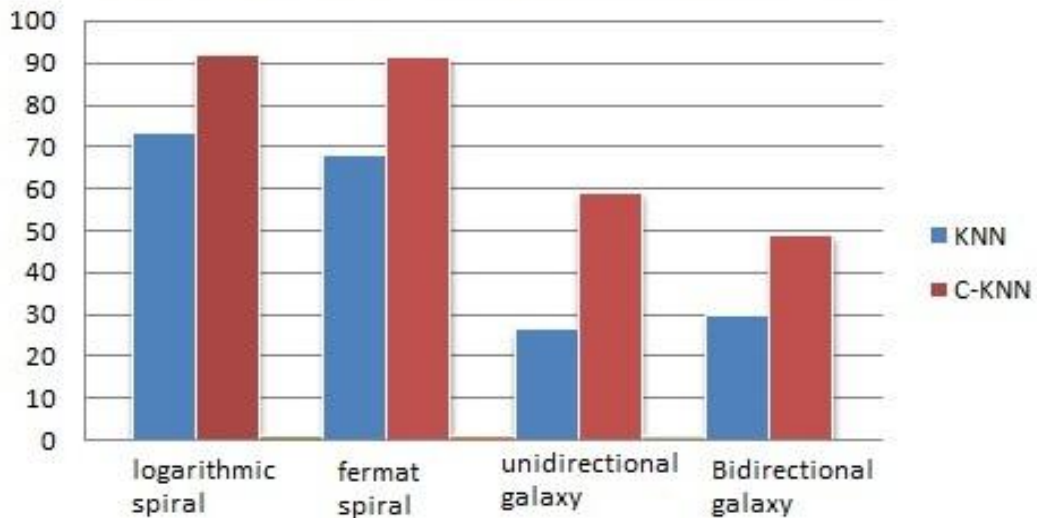


Figure 17: Graph of Average Accuracy For Each Methods

From the graph, it is concluded that only Logarithmic spiral and Fermat spiral reach the average accuracy of 90% by using C-KNN classifier. Hence, either Logarithmic or Fermat spiral is the most optimal approach for feature extraction.

Table 7: Effect On Accuracy While Varying K Values

K	Accuracy			
	Single spiral	Multiple spiral	unidirectional galaxy	Bidirectional galaxy
1	82	84	52	54
3	73	73	43	47.6
5	73	64	38	38.1
7	60	57	32	30.2
9	54	52	25	23.8

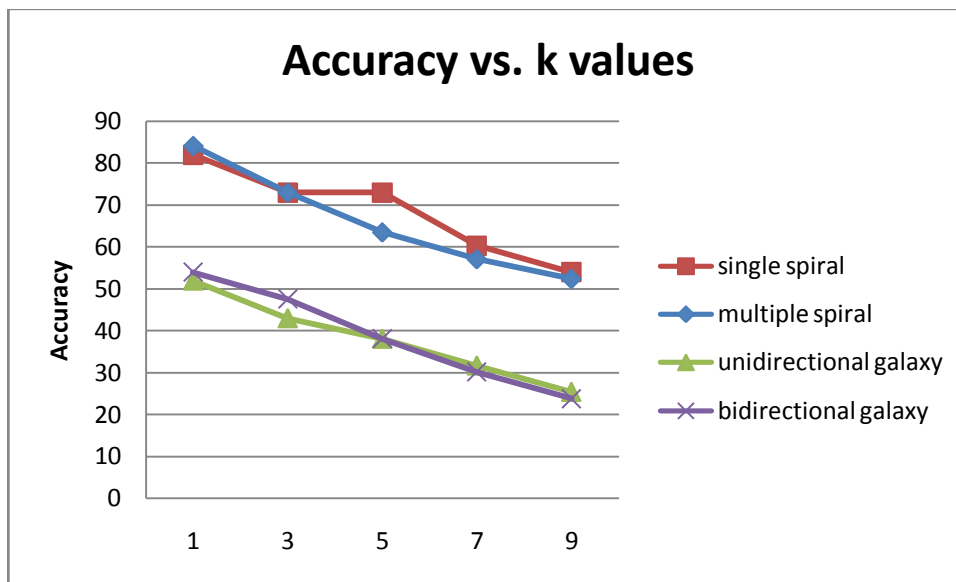


Figure 18: Analysis on Accuracy When Varying K Value

For each approach, the optimal parameter values used while varying the value of k. According to figure 17, it shows that the accuracy is inversely proportional with k values. As we increase the k values, the accuracy decreases. Thus, it is recommended that the value of k assigned is small to get the optimal result.

#### 4.4 Graphical User Interface(GUI) Application

A user-friendly Graphic User Interface (GUI) is built to integrate the classification module with database of written alphabets. By having this application, it will assist the user to apply character recognition system to their real application. Through the GUI the user can insert an image of alphabet. This new image can be classified to its respective cluster by referring to the database. The user may choose to use any approach such as single spiral, multiple spiral, unidirectional galaxy or bidirectional galaxy for feature extraction. Lastly, the class those image belong to will appear.

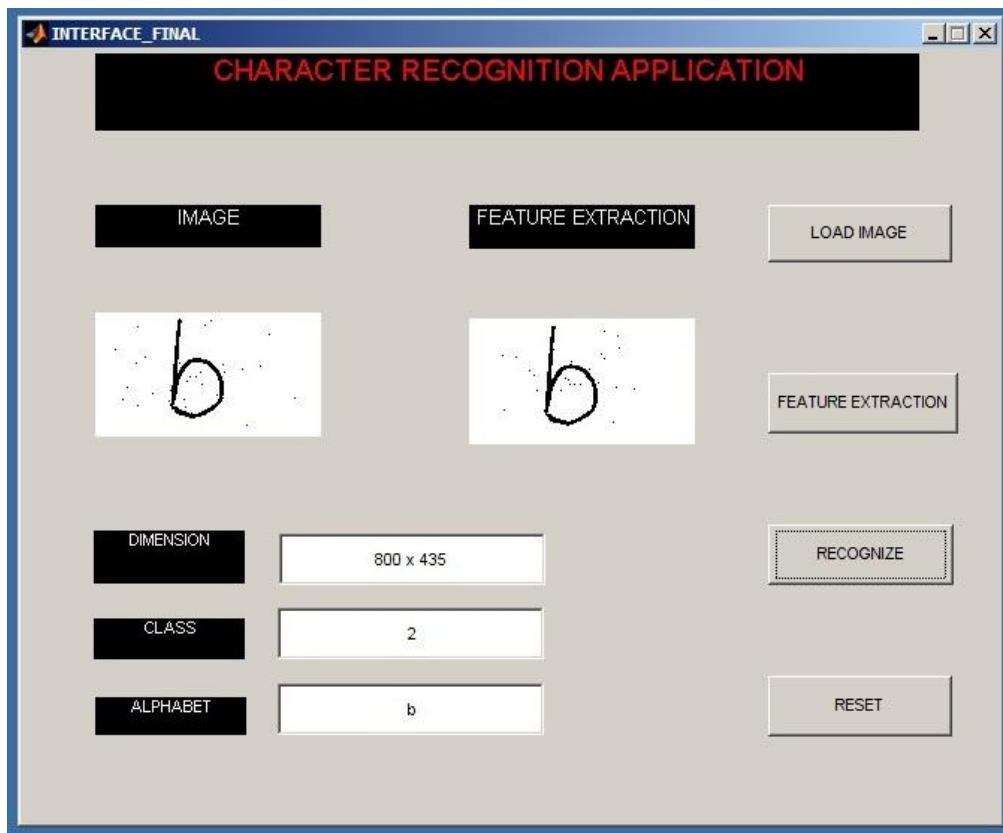


Figure 19: Graphical User Interface

## 4.5 Problem Encountered & Future Planning

The accuracy achieved so far is still low compare to author's target which is 99.21% to 99.95% by using Neural-Network method.[7].Besides, result also inconsistent for every execution and still searching for the solemnly factor that affect accuracy. For instance, if the period of sampling  $\Delta$  is too small it will cause low accuracy recognition. Thus, spiral with very short sampling not necessarily gives the good result. In addition, the size of vector still big and will consume more time for classification.

For the future work, it is recommended to attempt another input data to the algorithm. From that, we can analyze the degree of robustness for the classifier. For instance, the algorithm can be applied for signature recognition to record the attendance. The user will insert their signature into system. After that, the system will register the user into attendance record. The application will also preview the details of the user.The overview of the application can be seen below:

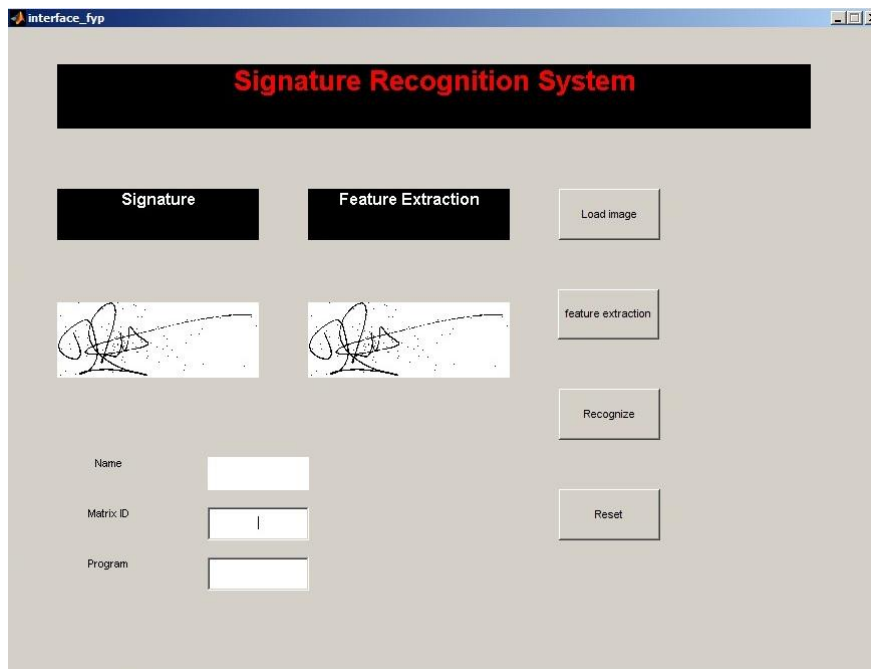


Figure 20: Proposed module for Signature Recognition System



## **CHAPTER 5**

### **CONCLUSION AND RECOMMENDATIONS**

#### **5.1 Conclusion**

Artificial Intelligent plays an important role in today's life. There are various approaches on pattern recognition. However-Nearest Neighbors (KNN) method is more preferable since it is simple and has faster execution time. The crucial aspect in pattern recognition is its accuracy since this system is widely used for safety purposes. The other idea of this project is to reduce the data size by shorten vector form representation of the image using simple feature extraction approach. If the vector size is too big, it will take longer time to being classified. Moreover, it also tends to produce error in classification by misplace the right class that objects belong to. After implementing all four extraction methods, the most efficient result is using single/logarithmic spiral. The Highest accuracy achieved throughout this project is 99.05 %.The user interface window has been developed in order to ease the user to utilize the algorithm.

## 5.2 Recommendations

Since the accuracy produced by certain approach is still unsatisfactory, thus the author need to keep on doing the improvisation on the algorithm in order to increase the plausibility of the recognition. The easiest way to improve the recognition is by tuning the parameter value ( $V_0$ ,  $\omega$ ,  $\Delta$ ) for feature extraction. Assignment of the value can vary depending on the method used to get the features. Other than that, there are few suggested methods or additional algorithm that can be implemented to increase the accuracy.

### 5.2.1 Variance-modified Methods

This method used to enhance the accuracy of recognition. We can use Gaussian approximation variables for each execution of algorithm to obtained the highest accuracy

$$P\left(\frac{error}{x}\right) = \sum_{i=1}^2 P(x \in class_i), P\left(\frac{error}{x} \in class_i\right) \quad (5.1)$$

### 5.2.2 Genetic K-Nearest Neighbors Method

Genetic algorithm (GA) is combined with K-nearest neighbors algorithm called G-KNN to overcome the limitation of traditional KNN. By using GA, k-number of samples are chosen for each iteration and the classification accuracy is calculated as fitness. The highest accuracy is recorded each time thus avoid the calculation of similarities between each sample.[9]

## REFERENCES

- [1] KardiTeknomo,*K-Nearest Neighbor Tutorial* [journal]-Japan, Graduate School of Information Sciences, 2006
  
- [2] Yang Song,JianHuangDingZhou,HongyuanZha, Lee Giles  
*IKNN:Informative K-Nearest Neighbor Pattern Classification*[Journal]-university Park,PAUSA.
  
- [3] David Bremner, Erik Demaine, Jeff Erickson, John Iacono, Stefan Langerman, Pat Morin, and Godfried Toussaint, *Output-sensitive algorithms for computing nearest-neighbor decision boundaries, Discrete and Computational Geometry*[book]-New Jersey,John Wiley and Sons Inc,2005,Vol. 33
  
- [4] Richard O. Duda, Peter E. Hart, David G. Stork *Pattern classification* [book]-New York,Wiley,2001,2nd edition
  
- [5] Beyer, W. H.*CRC Standard Mathematical Tables*,.Boca Raton, FL: CRC Press, p. 225, 1987, 28th edition
  
- [6] Li Baoli, Yu Shiwen, and Lu Qin,"*An Improved k-Nearest Neighbor Algorithm for Text Categorization*"[journal]-Institute of Computational Linguistics Department of Computer Science and Technology PekingUniversity, Beijing, P.R. China

- [7] Steffen Wachenfeld, Sebastian Terlunen, XiaoyiJiang, "Robust Recognition of 1-D Barcodes Using Camera Phones"[journal] Computer Vision and Pattern Recognition Group, Department of Computer Science, University of Munster, Germany
- [8] Mark Nixon, Alberto Aguado, *Feature Extraction & Image Processing*[book]-Oxford, Newness, p.301 2002, 1<sup>st</sup> Edition
- [9] N. Suguna, Dr K. Thanushkodi, "Improved k-Nearest neighbors Classification Using Genetic Algorithm"[journal]-Computer Science and Engineering, Akshaya College of Engineering and Technology, Coimbatore, TamilNadu, India
- [10] Avi Drissman, Dr. Sethi, "Handwriting Recognition Systems: An Overview"[journal]-CSC 496, India
- [11] Shashank Araokar, *Visual Character Recognition Using Artificial Neural Network* [journal]-Electronics and telecommunication, University of Mumbai, India
- [12] L.Harmon. "automatic recognition of print and script.Proc."IEE,60:1165-1176, 1972.
- [13] Dongxiao N., Zhihong G., Xing M. 2007, "Research on Neural Networks Based on Culture Particle Swarm Optimization and Its Application in Power Load Forecasting" Natural Computation, ICNC 2007. Third International Conference, vol.1, pp. 2700274.
- [14] D. M. Titterington, A. F. M. Smith, and U. E. Makov, *Statistical Analysis of Finite Mixture Distributions*. John Wiley, New York, 1985

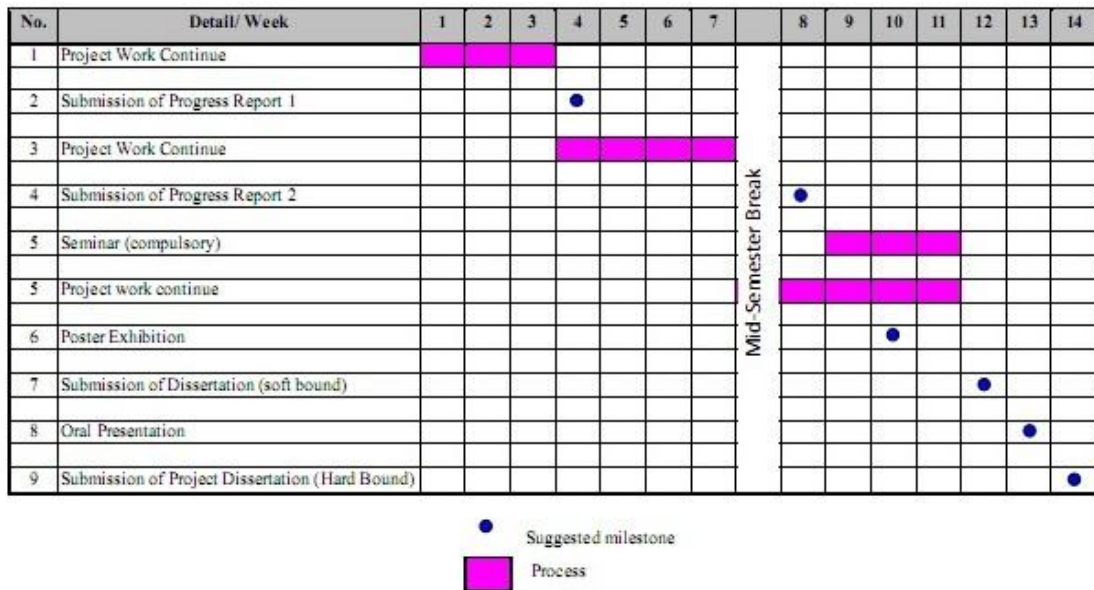
- [15] X. Jia and J. A. richards, “*Fast k-NN classification Using the Cluster-Space Approach*”, IEEE Trans., Geosci. Remote Sens., vol. 2, no. 2, April 2005.

## **APPENDICES**

## APPENDIX A GANTT CHART FOR FYP 1



## APPENDIX B GANTT CHART FOR FYP 2



## APPENDIX C

### SOURCE CODE IN MATLAB FOR LOGARITHMIC SPIRAL EQUATION

```
clearall;

x=imread('A1.bmp');%A1-A7,B1-B8,C1-C5,D1-D8,E1-E5
x=abs(x-1);
[n_xm_x]=size(x);
%%% next step is finding the gravity point
adx=0;
for i=1:n_x
for j=1:m_x
adx=adx+i*x(i,j);
end
end
g_x=adx/sum(sum(x~=0));
adx=0;

for i=1:n_x
for j=1:m_x
adx=adx+j*x(i,j);
end
end
g_y=adx/sum(sum(x~=0));
gravity=[n_xm_x; g_xg_y]

V0=1; del=0.4;w=4;
N=round( max(g_x/(V0*del),g_y/(V0*del)) + min((n_x-g_x)/(V0*del),(m_x-
g_y)/(V0*del)) )/2;

vect_f(1)=x(round(g_x+(V0*del*0*cos(w*del*0))),round(g_y+(V0*del*0*sin(w*del*0))))
;
for T=1:N
if round(g_x+(V0*del*T*cos(w*del*T)))<=0
break;
end

if round(g_y+(V0*del*T*sin(w*del*T)))<=0
break;
end
if round(g_x+(V0*del*T*cos(w*del*T)))>n_x
break;
end

if round(g_y+(V0*del*T*sin(w*del*T)))>m_x
break;
end
vect_f=[vect_f
x(round(g_x+(V0*del*T*cos(w*del*T))),round(g_y+(V0*del*T*sin(w*del*T))))];
end

[mx posit]=sort(vect_f);
siz_L=size(vect_f, 2);
vect_f(posit(siz_L)+1:siz_L)=[];
fid = fopen('vect_A1.txt','wt');
fwrite(fid,vect_f,'float');
fclose(fid);
size(vect_f)
```



## APPENDIX D

### ADDITIONAL CODE FOR FERMAT SPIRAL EQUATION

```
V0=1; del=0.4;w=4;
N=round( max(g_x/(V0*del),g_y/(V0*del)) + min((n_x-g_x)/(V0*del),(m_x-
g_y)/(V0*del)) )/2;

%%%% apply spiral equation in reverse rotation

vect_fr(1)=x(round(g_x+(V0*del*0*cos(w*del*0)),round(g_y-(V0*del*0*sin(w*del*0))));
for T=1:N
if round(g_x+(V0*del*T*cos(w*del*T))<=0
break;
end

if round(g_y-(V0*del*T*sin(w*del*T))<=0
break;
end
if round(g_x+(V0*del*T*cos(w*del*T))>n_x
break;
end

if round(g_y-(V0*del*T*sin(w*del*T))>m_x
break;
end
vect_fr=[vect_fr x(round(g_x+(V0*del*T*cos(w*del*T))),round(g_y-
(V0*del*T*sin(w*del*T))))];
end

%%%% apply spiral equation in forward rotation

vect_ff(1)=x(round(g_x+(V0*del*0*cos(w*del*0)),round(g_y+(V0*del*0*sin(w*del*0))));
for T=1:N
if round(g_x+(V0*del*T*cos(w*del*T))<=0
break;
end

if round(g_y+(V0*del*T*sin(w*del*T))<=0
break;
end
if round(g_x+(V0*del*T*cos(w*del*T))>n_x
break;
end

if round(g_y+(V0*del*T*sin(w*del*T))>m_x
break;
end
vect_ff=[vect_ff
x(round(g_x+(V0*del*T*cos(w*del*T))),round(g_y+(V0*del*T*sin(w*del*T))))];
end

%%%% arrange the vector in an array vect_f=[vect_ffvect_fr];

vect_f=[vect_ffvect_fr];
[mx posit]=sort(vect_f);
siz_L=size(vect_f, 2);

vect_f(posit(siz_L)+1:siz_L)=[];
fid = fopen('vect_A1.txt','wt');
fwrite(fid,vect_f,'float');
fclose(fid);
size(vect_f)
```

## APPENDIX E

### ADDITIONAL CODE FOR UNIDIRECTIONAL GALAXY

```

c=bwmorph(x,'erode');
c=abs(c-1);
[n_x m_x]=size(c);
%% next step is find the gravity point
adx=0;
for i=1:n_x
    for j=1:m_x
        adx=adx+i*c(i,j);
    end
end
g_x=adx/sum(sum(c~=0));
adx=0;

for i=1:n_x
    for j=1:m_x
        adx=adx+j*c(i,j);
    end
end
g_y=adx/sum(sum(c~=0));
gravity=[n_x m_x; g_x g_y]
parameter();

Graph_function=zeros(size(c));
final_pt=round( max((n_x-g_x)/(V0*del),(m_x-g_y)/(V0*del)) );
vect_ff=0;
for j_a=0:n_g
    for T=0:1:final_pt

        if
            (round(g_x+(V0*del*T*cos((w*del*T+(2*pi*j_a)/n_g))))>0)&(round(g_y+(V0*del*T*sin((w*del*T+(2*pi*j_a)/n_g))))>0)&(r
            ound(g_x+(V0*del*T*cos((w*del*T+(2*pi*j_a)/n_g))))<=n_x)&(round(g_y+(V0*del*T*sin((w*del*T+(2*pi*j_a)/n_g))))<=m
            _x)

            Graph_function(round(g_x+(V0*del*T*cos(w*del*T+(2*pi*j_a)/n_g))),round(g_y+(V0*del*T*sin(w*del*T+(2*pi*j_a)/n_g)))
            )=1;
            vect_ff=[vect_ff
            c(round(g_x+(V0*del*T*cos(w*del*T+(2*pi*j_a)/n_g))),round(g_y+(V0*del*T*sin(w*del*T+(2*pi*j_a)/n_g))))];
        end

    end
end

vect_f=[vect_ff];
[mx posit]=sort(vect_f);
siz_L=size(vect_f, 2);

vect_f(posit(siz_L)+1:siz_L)=[];

```

## APPENDIX F

### ADDITIONAL CODE FOR BIDIRECTIONAL GALAXY

```

c=bwmorph(x,'erode');
c=abs(c-1);
[n_x m_x]=size(c);

adx=0;
for i=1:n_x
    for j=1:m_x
        adx=adx+i*c(i,j);
    end
end
g_x=adx/sum(sum(c~=0));
adx=0;

for i=1:n_x
    for j=1:m_x
        adx=adx+j*c(i,j);
    end
end
g_y=adx/sum(sum(c~=0));
gravity=[n_x m_x; g_x g_y]
parameter();

Graph_function=zeros(size(c));
final_pt=round( max((n_x-g_x)/(V0*del),(m_x-g_y)/(V0*del) ) );
vect_ff=0;
for j_a=0:n_g
    for T=0:1:final_pt

        if
            (round(g_x+(V0*del*T*cos((w*del*T+(2*pi*j_a)/n_g))))>0)&(round(g_y+(V0*del*T*sin((w*del*T+(2*pi*j_a)/n_g))))>0)&(r
            ound(g_x+(V0*del*T*cos((w*del*T+(2*pi*j_a)/n_g))))<=n_x)&(round(g_y+(V0*del*T*sin((w*del*T+(2*pi*j_a)/n_g))))<=m
            _x)

            Graph_function(round(g_x+(V0*del*T*cos(w*del*T+(2*pi*j_a)/n_g))),round(g_y+(V0*del*T*sin(w*del*T+(2*pi*j_a)/n_g)))
            )=1;
            vect_ff=[vect_ff
            c(round(g_x+(V0*del*T*cos(w*del*T+(2*pi*j_a)/n_g))),round(g_y+(V0*del*T*sin(w*del*T+(2*pi*j_a)/n_g))))];
            end

        end
    end

    for j_a=0:n_g
        for T=0:1:final_pt

            if (round(g_x+(V0*del*T*cos((w*del*T+(2*pi*j_a)/n_g))))>0)&(round(g_y-
            (V0*del*T*sin((w*del*T+(2*pi*j_a)/n_g))))>0)&(round(g_x+(V0*del*T*cos((w*del*T+(2*pi*j_a)/n_g))))<=n_x)&(round(g_
            y-(V0*del*T*sin((w*del*T+(2*pi*j_a)/n_g))))<=m_x)

                Graph_function(round(g_x+(V0*del*T*cos(w*del*T+(2*pi*j_a)/n_g))),round(g_y-
                (V0*del*T*sin(w*del*T+(2*pi*j_a)/n_g))))=1;
                vect_ff=[vect_ff x(round(g_x+(V0*del*T*cos(w*del*T+(2*pi*j_a)/n_g))),round(g_y-
                (V0*del*T*sin(w*del*T+(2*pi*j_a)/n_g))))];
            end

        end
    end

    vect_f=[vect_ff];

```

## APPENDIX G

### SOURCE CODE IN MATLAB FOR CLASSIFICATION

```
clear all;
fid = fopen('vect_A1.txt','rt')
vect_A1=fread(fid,'float')
fclose(fid)

fid = fopen('vect_A2.txt','rt')
vect_A2=fread(fid,'float')
fclose(fid)
fid = fopen('vect_A3.txt','rt')
vect_A3=fread(fid,'float')
fclose(fid)
fid = fopen('vect_A4.txt','rt')
vect_A4=fread(fid,'float')
fclose(fid)
fid = fopen('vect_A5.txt','rt')
vect_A5=fread(fid,'float')
fclose(fid)
fid = fopen('vect_A6.txt','rt')
vect_A6=fread(fid,'float')
fclose(fid)
fid = fopen('vect_A7.txt','rt')
vect_A7=fread(fid,'float')
fclose(fid)
fid = fopen('vect_A8.txt','rt')
vect_A8=fread(fid,'float')
fclose(fid)
fid = fopen('vect_A9.txt','rt')
vect_A9=fread(fid,'float')
fclose(fid)
fid = fopen('vect_A10.txt','rt')
vect_A10=fread(fid,'float')
fclose(fid)
fid = fopen('vect_A11.txt','rt')
vect_A11=fread(fid,'float')
fclose(fid)
fid = fopen('vect_A12.txt','rt')
vect_A12=fread(fid,'float')
fclose(fid)
fid = fopen('vect_A13.txt','rt')
vect_A13=fread(fid,'float')
fclose(fid)
fid = fopen('vect_A14.txt','rt')
vect_A14=fread(fid,'float')
fclose(fid)
fid = fopen('vect_A15.txt','rt')
vect_A15=fread(fid,'float')
fclose(fid)
fid = fopen('vect_B1.txt','rt')
vect_B1=fread(fid,'float')
fclose(fid)
fid = fopen('vect_B2.txt','rt')
vect_B2=fread(fid,'float')
fclose(fid)
fid = fopen('vect_B3.txt','rt')
vect_B3=fread(fid,'float')
fclose(fid)
fid = fopen('vect_B4.txt','rt')
```

```
vect_B4=fread(fid,'float')
fclose(fid)
fid = fopen('vect_B5.txt','rt')
vect_B5=fread(fid,'float')
fclose(fid)
fid = fopen('vect_B6.txt','rt')
vect_B6=fread(fid,'float')
fclose(fid)
fid = fopen('vect_B7.txt','rt')
vect_B7=fread(fid,'float')
fclose(fid)
fid = fopen('vect_B8.txt','rt')
vect_B8=fread(fid,'float')
fclose(fid)
fid = fopen('vect_B9.txt','rt')
vect_B9=fread(fid,'float')
fclose(fid)
fid = fopen('vect_B10.txt','rt')
vect_B10=fread(fid,'float')
fclose(fid)
fid = fopen('vect_B11.txt','rt')
vect_B11=fread(fid,'float')
fclose(fid)
fid = fopen('vect_B12.txt','rt')
vect_B12=fread(fid,'float')
fclose(fid)
fid = fopen('vect_B13.txt','rt')
vect_B13=fread(fid,'float')
fclose(fid)
fid = fopen('vect_B14.txt','rt')
vect_B14=fread(fid,'float')
fclose(fid)
fid = fopen('vect_B15.txt','rt')
vect_B15=fread(fid,'float')
fclose(fid)
fid = fopen('vect_C1.txt','rt')
vect_C1=fread(fid,'float')
fclose(fid)
fid = fopen('vect_C2.txt','rt')
vect_C2=fread(fid,'float')
fclose(fid)
fid = fopen('vect_C3.txt','rt')
vect_C3=fread(fid,'float')
fclose(fid)
fid = fopen('vect_C4.txt','rt')
vect_C4=fread(fid,'float')
fclose(fid)
fid = fopen('vect_C5.txt','rt')
vect_C5=fread(fid,'float')
fclose(fid)
fid = fopen('vect_C6.txt','rt')
vect_C6=fread(fid,'float')
fclose(fid)
fid = fopen('vect_C7.txt','rt')
vect_C7=fread(fid,'float')
fclose(fid)
fid = fopen('vect_C8.txt','rt')
vect_C8=fread(fid,'float')
fclose(fid)
fid = fopen('vect_C9.txt','rt')
vect_C9=fread(fid,'float')
fclose(fid)
fid = fopen('vect_C10.txt','rt')
vect_C10=fread(fid,'float')
fclose(fid)
fid = fopen('vect_C11.txt','rt')
vect_C11=fread(fid,'float')
```

```
fclose(fid)
fid = fopen('vect_C12.txt','rt')
vect_C12=fread(fid,'float')
fclose(fid)
fid = fopen('vect_C13.txt','rt')
vect_C13=fread(fid,'float')
fclose(fid)
fid = fopen('vect_C14.txt','rt')
vect_C14=fread(fid,'float')
fclose(fid)
fid = fopen('vect_C15.txt','rt')
vect_C15=fread(fid,'float')
fclose(fid)
fid = fopen('vect_D1.txt','rt')
vect_D1=fread(fid,'float')
fclose(fid)
fid = fopen('vect_D2.txt','rt')
vect_D2=fread(fid,'float')
fclose(fid)
fid = fopen('vect_D3.txt','rt')
vect_D3=fread(fid,'float')
fclose(fid)
fid = fopen('vect_D4.txt','rt')
vect_D4=fread(fid,'float')
fclose(fid)
fid = fopen('vect_D5.txt','rt')
vect_D5=fread(fid,'float')
fclose(fid)
fid = fopen('vect_D6.txt','rt')
vect_D6=fread(fid,'float')
fclose(fid)
fid = fopen('vect_D7.txt','rt')
vect_D7=fread(fid,'float')
fclose(fid)
fid = fopen('vect_D8.txt','rt')
vect_D8=fread(fid,'float')
fclose(fid)
fid = fopen('vect_D9.txt','rt')
vect_D9=fread(fid,'float')
fclose(fid)
fid = fopen('vect_D10.txt','rt')
vect_D10=fread(fid,'float')
fclose(fid)
fid = fopen('vect_D11.txt','rt')
vect_D11=fread(fid,'float')
fclose(fid)
fid = fopen('vect_D12.txt','rt')
vect_D12=fread(fid,'float')
fclose(fid)
fid = fopen('vect_D13.txt','rt')
vect_D13=fread(fid,'float')
fclose(fid)
fid = fopen('vect_D14.txt','rt')
vect_D14=fread(fid,'float')
fclose(fid)
fid = fopen('vect_D15.txt','rt')
vect_D15=fread(fid,'float')
fclose(fid)
fid = fopen('vect_E1.txt','rt')
vect_E1=fread(fid,'float')
fclose(fid)
fid = fopen('vect_E1.txt','rt')
vect_E1=fread(fid,'float')
fclose(fid)
fid = fopen('vect_E2.txt','rt')
vect_E2=fread(fid,'float')
fclose(fid)
```

```
fid = fopen('vect_E3.txt','rt')
vect_E3=fread(fid,'float')
fclose(fid)
fid = fopen('vect_E4.txt','rt')
vect_E4=fread(fid,'float')
fclose(fid)
fid = fopen('vect_E5.txt','rt')
vect_E5=fread(fid,'float')
fclose(fid)
fid = fopen('vect_E6.txt','rt')
vect_E6=fread(fid,'float')
fclose(fid)
fid = fopen('vect_E7.txt','rt')
vect_E7=fread(fid,'float')
fclose(fid)
fid = fopen('vect_E8.txt','rt')
vect_E8=fread(fid,'float')
fclose(fid)
fid = fopen('vect_E9.txt','rt')
vect_E9=fread(fid,'float')
fclose(fid)
fid = fopen('vect_E10.txt','rt')
vect_E10=fread(fid,'float')
fclose(fid)
fid = fopen('vect_E11.txt','rt')
vect_E11=fread(fid,'float')
fclose(fid)
fid = fopen('vect_E12.txt','rt')
vect_E12=fread(fid,'float')
fclose(fid)
fid = fopen('vect_E13.txt','rt')
vect_E13=fread(fid,'float')
fclose(fid)
fid = fopen('vect_E14.txt','rt')
vect_E14=fread(fid,'float')
fclose(fid)
fid = fopen('vect_E15.txt','rt')
vect_E15=fread(fid,'float')
fclose(fid)
fid = fopen('vect_F1.txt','rt')
vect_F1=fread(fid,'float')
fclose(fid)
fid = fopen('vect_F2.txt','rt')
vect_F2=fread(fid,'float')
fclose(fid)
fid = fopen('vect_F3.txt','rt')
vect_F3=fread(fid,'float')
fclose(fid)
fid = fopen('vect_F4.txt','rt')
vect_F4=fread(fid,'float')
fclose(fid)
fid = fopen('vect_F5.txt','rt')
vect_F5=fread(fid,'float')
fclose(fid)
fid = fopen('vect_F6.txt','rt')
vect_F6=fread(fid,'float')
fclose(fid)
fid = fopen('vect_F7.txt','rt')
vect_F7=fread(fid,'float')
fclose(fid)
fid = fopen('vect_F8.txt','rt')
vect_F8=fread(fid,'float')
fclose(fid)
fid = fopen('vect_F9.txt','rt')
vect_F9=fread(fid,'float')
fclose(fid)
fid = fopen('vect_F10.txt','rt')
```

```
vect_F10=fread(fid,'float')
fclose(fid)
fid = fopen('vect_F11.txt','rt')
vect_F11=fread(fid,'float')
fclose(fid)
fid = fopen('vect_F12.txt','rt')
vect_F12=fread(fid,'float')
fclose(fid)
fid = fopen('vect_F13.txt','rt')
vect_F13=fread(fid,'float')
fclose(fid)
fid = fopen('vect_F14.txt','rt')
vect_F14=fread(fid,'float')
fclose(fid)
fid = fopen('vect_F15.txt','rt')
vect_F15=fread(fid,'float')
fclose(fid)
fid = fopen('vect_G1.txt','rt')
vect_G1=fread(fid,'float')
fclose(fid)
fid = fopen('vect_G2.txt','rt')
vect_G2=fread(fid,'float')
fclose(fid)
fid = fopen('vect_G3.txt','rt')
vect_G3=fread(fid,'float')
fclose(fid)
fid = fopen('vect_G4.txt','rt')
vect_G4=fread(fid,'float')
fclose(fid)
fid = fopen('vect_G5.txt','rt')
vect_G5=fread(fid,'float')
fclose(fid)
fid = fopen('vect_G6.txt','rt')
vect_G6=fread(fid,'float')
fclose(fid)
fid = fopen('vect_G7.txt','rt')
vect_G7=fread(fid,'float')
fclose(fid)
fid = fopen('vect_G8.txt','rt')
vect_G8=fread(fid,'float')
fclose(fid)
fid = fopen('vect_G9.txt','rt')
vect_G9=fread(fid,'float')
fclose(fid)
fid = fopen('vect_G10.txt','rt')
vect_G10=fread(fid,'float')
fclose(fid)
fid = fopen('vect_G11.txt','rt')
vect_G11=fread(fid,'float')
fclose(fid)
fid = fopen('vect_G12.txt','rt')
vect_G12=fread(fid,'float')
fclose(fid)
fid = fopen('vect_G13.txt','rt')
vect_G13=fread(fid,'float')
fclose(fid)
fid = fopen('vect_G14.txt','rt')
vect_G14=fread(fid,'float')
fclose(fid)
fid = fopen('vect_G15.txt','rt')
vect_G15=fread(fid,'float')
fclose(fid)
fid = fopen('vect_H1.txt','rt')
vect_H1=fread(fid,'float')
fclose(fid)
fid = fopen('vect_H2.txt','rt')
vect_H2=fread(fid,'float')
```



```
fclose(fid)
fid = fopen('vect_H3.txt','rt')
vect_H3=fread(fid,'float')
fclose(fid)
fid = fopen('vect_H4.txt','rt')
vect_H4=fread(fid,'float')
fclose(fid)
fid = fopen('vect_H5.txt','rt')
vect_H5=fread(fid,'float')
fclose(fid)
fid = fopen('vect_H6.txt','rt')
vect_H6=fread(fid,'float')
fclose(fid)
fid = fopen('vect_H7.txt','rt')
vect_H7=fread(fid,'float')
fclose(fid)
fid = fopen('vect_H8.txt','rt')
vect_H8=fread(fid,'float')
fclose(fid)
fid = fopen('vect_H9.txt','rt')
vect_H9=fread(fid,'float')
fclose(fid)
fid = fopen('vect_H10.txt','rt')
vect_H10=fread(fid,'float')
fclose(fid)
fid = fopen('vect_H11.txt','rt')
vect_H11=fread(fid,'float')
fclose(fid)
fid = fopen('vect_H12.txt','rt')
vect_H12=fread(fid,'float')
fclose(fid)
fid = fopen('vect_H13.txt','rt')
vect_H13=fread(fid,'float')
fclose(fid)
fid = fopen('vect_H14.txt','rt')
vect_H14=fread(fid,'float')
fclose(fid)
fid = fopen('vect_H15.txt','rt')
vect_H15=fread(fid,'float')
fclose(fid)
fid = fopen('vect_I1.txt','rt')
vect_I1=fread(fid,'float')
fclose(fid)
fid = fopen('vect_I2.txt','rt')
vect_I2=fread(fid,'float')
fclose(fid)
fid = fopen('vect_I3.txt','rt')
vect_I3=fread(fid,'float')
fclose(fid)
fid = fopen('vect_I4.txt','rt')
vect_I4=fread(fid,'float')
fclose(fid)
fid = fopen('vect_I5.txt','rt')
vect_I5=fread(fid,'float')
fclose(fid)
fid = fopen('vect_I6.txt','rt')
vect_I6=fread(fid,'float')
fclose(fid)
fid = fopen('vect_I7.txt','rt')
vect_I7=fread(fid,'float')
fclose(fid)
fid = fopen('vect_I8.txt','rt')
vect_I8=fread(fid,'float')
fclose(fid)
fid = fopen('vect_I9.txt','rt')
vect_I9=fread(fid,'float')
fclose(fid)
```

```

fid = fopen('vect_I10.txt','rt')
vect_I10=fread(fid,'float')
fclose(fid)
fid = fopen('vect_I11.txt','rt')
vect_I11=fread(fid,'float')
fclose(fid)
fid = fopen('vect_I12.txt','rt')
vect_I12=fread(fid,'float')
fclose(fid)
fid = fopen('vect_I13.txt','rt')
vect_I13=fread(fid,'float')
fclose(fid)
fid = fopen('vect_I14.txt','rt')
vect_I14=fread(fid,'float')
fclose(fid)
fid = fopen('vect_I15.txt','rt');
vect_I15=fread(fid,'float');
fclose(fid)

```

```

n=max([size(vect_A1,1),size(vect_A2,1), size(vect_A3,1),size(vect_A4,1), size(vect_A5,1)
,size(vect_A6,1),size(vect_A7,1),size(vect_A8,1), size(vect_A9,1),size(vect_A10,1),
size(vect_A11,1),size(vect_A12,1),size(vect_A13,1),size(vect_A14,1),size(vect_A15,1),size(vect_B1,1),size(vect_B2,1),
size(vect_B3,1),size(vect_B4,1), size(vect_B5,1) ,size(vect_B6,1),size(vect_B7,1),size(vect_B8,1),
size(vect_B9,1),size(vect_B10,1),
size(vect_B11,1),size(vect_B12,1),size(vect_B13,1),size(vect_B14,1),size(vect_B15,1),size(vect_C1,1),size(vect_C2,1),
size(vect_C3,1),size(vect_C4,1), size(vect_C5,1) ,size(vect_C6,1),size(vect_C7,1),size(vect_C8,1),
size(vect_C9,1),size(vect_C10,1),
size(vect_C11,1),size(vect_C12,1),size(vect_C13,1),size(vect_C14,1),size(vect_C15,1),size(vect_D1,1),size(vect_D2,1),
size(vect_D3,1),size(vect_D4,1), size(vect_D5,1) ,size(vect_D6,1),size(vect_D7,1),size(vect_D8,1),
size(vect_D9,1),size(vect_D10,1),
size(vect_D11,1),size(vect_D12,1),size(vect_D13,1),size(vect_D14,1),size(vect_D15,1),size(vect_E1,1),size(vect_E2,1),size(vect_E3,1),size(vect_E4,1),size(vect_E5,1),size(vect_E6,1),size(vect_E7,1),size(vect_E8,1),size(vect_E9,1),size(vect_E10,1),size(vect_E11,1),size(vect_E12,1),size(vect_E13,1),size(vect_E14,1),size(vect_E15,1),size(vect_E15,1),size(vect_E15,1),size(vect_E15,1),size(vect_E15,1),size(vect_F1,1),size(vect_F2,1),size(vect_F3,1),size(vect_F4,1),size(vect_F5,1),size(vect_F6,1),size(vect_F7,1),size(vect_F8,1),size(vect_F9,1),size(vect_F10,1),size(vect_F11,1),size(vect_F12,1),size(vect_F13,1),size(vect_F14,1),size(vect_F15,1),size(vect_G1,1),size(vect_G2,1),size(vect_G3,1),size(vect_G4,1),size(vect_G5,1),size(vect_G6,1),size(vect_G7,1),size(vect_G8,1),size(vect_G9,1),size(vect_G10,1),size(vect_G11,1),size(vect_G12,1),size(vect_G13,1),size(vect_G14,1),size(vect_G15,1),size(vect_H1,1),size(vect_H2,1),size(vect_H3,1),size(vect_H4,1),size(vect_H5,1),size(vect_H6,1),size(vect_H7,1),size(vect_H7,1),size(vect_H8,1),size(vect_H9,1),size(vect_H10,1),size(vect_H11,1),size(vect_H12,1),size(vect_H13,1),size(vect_H14,1),size(vect_H15,1),size(vect_I1,1),size(vect_I2,1),size(vect_I3,1),size(vect_I4,1),size(vect_I5,1),size(vect_I6,1),size(vect_I7,1),size(vect_I8,1),size(vect_I9,1),size(vect_I10,1),size(vect_I11,1),size(vect_I12,1),size(vect_I13,1),size(vect_I14,1),size(vect_I15,1)]);

```

```

v1=zeros(n,1);
v2=zeros(n,1);
v3=zeros(n,1);
v4=zeros(n,1);
v5=zeros(n,1);
v6=zeros(n,1);
v7=zeros(n,1);
v8=zeros(n,1);
v9=zeros(n,1);
v10=zeros(n,1);
v11=zeros(n,1);
v12=zeros(n,1);
v13=zeros(n,1);
v14=zeros(n,1);
v15=zeros(n,1);
v16=zeros(n,1);
v17=zeros(n,1);
v18=zeros(n,1);
v19=zeros(n,1);
v20=zeros(n,1);
v21=zeros(n,1);
v22=zeros(n,1);
v23=zeros(n,1);

```

```
v24=zeros(n,1);
v25=zeros(n,1);
v26=zeros(n,1);
v27=zeros(n,1);
v28=zeros(n,1);
v29=zeros(n,1);
v30=zeros(n,1);
v31=zeros(n,1);
v32=zeros(n,1);
v33=zeros(n,1);
v34=zeros(n,1);
v35=zeros(n,1);
v36=zeros(n,1);
v37=zeros(n,1);
v38=zeros(n,1);
v39=zeros(n,1);
v40=zeros(n,1);
v41=zeros(n,1);
v42=zeros(n,1);
v43=zeros(n,1);
v44=zeros(n,1);
v45=zeros(n,1);
v46=zeros(n,1);
v47=zeros(n,1);
v48=zeros(n,1);
v49=zeros(n,1);
v50=zeros(n,1);
v51=zeros(n,1);
v52=zeros(n,1);
v53=zeros(n,1);
v54=zeros(n,1);
v55=zeros(n,1);
v56=zeros(n,1);
v57=zeros(n,1);
v58=zeros(n,1);
v59=zeros(n,1);
v60=zeros(n,1);
v61=zeros(n,1);
v62=zeros(n,1);
v63=zeros(n,1);
v64=zeros(n,1);
v65=zeros(n,1);
v66=zeros(n,1);
v67=zeros(n,1);
v68=zeros(n,1);
v69=zeros(n,1);
v70=zeros(n,1);
v71=zeros(n,1);
v72=zeros(n,1);
v73=zeros(n,1);
v74=zeros(n,1);
v75=zeros(n,1);
v76=zeros(n,1);
v77=zeros(n,1);
v78=zeros(n,1);
v79=zeros(n,1);
v80=zeros(n,1);
v81=zeros(n,1);
v82=zeros(n,1);
v83=zeros(n,1);
v84=zeros(n,1);
v85=zeros(n,1);
v86=zeros(n,1);
v87=zeros(n,1);
v88=zeros(n,1);
v89=zeros(n,1);
v90=zeros(n,1);
```

```

v91=zeros(n,1);
v92=zeros(n,1);
v93=zeros(n,1);
v94=zeros(n,1);
v95=zeros(n,1);
v96=zeros(n,1);
v97=zeros(n,1);
v98=zeros(n,1);
v99=zeros(n,1);
v100=zeros(n,1);
v101=zeros(n,1);
v102=zeros(n,1);
v103=zeros(n,1);
v104=zeros(n,1);
v105=zeros(n,1);
v106=zeros(n,1);
v107=zeros(n,1);
v108=zeros(n,1);
v109=zeros(n,1);
v110=zeros(n,1);
v111=zeros(n,1);
v112=zeros(n,1);
v113=zeros(n,1);
v114=zeros(n,1);
v115=zeros(n,1);
v116=zeros(n,1);
v117=zeros(n,1);
v118=zeros(n,1);
v119=zeros(n,1);
v120=zeros(n,1);
v121=zeros(n,1);
v122=zeros(n,1);
v123=zeros(n,1);
v124=zeros(n,1);
v125=zeros(n,1);
v126=zeros(n,1);
v127=zeros(n,1);
v128=zeros(n,1);
v129=zeros(n,1);
v130=zeros(n,1);
v131=zeros(n,1);
v132=zeros(n,1);
v133=zeros(n,1);
v134=zeros(n,1);
v135=zeros(n,1);

v1(1:size(vect_A1,1))=vect_A1;
v2(1:size(vect_A2,1))=vect_A2;
v3(1:size(vect_A3,1))=vect_A3;
v4(1:size(vect_A4,1))=vect_A4;
v5(1:size(vect_A5,1))=vect_A5;
v6(1:size(vect_A6,1))=vect_A6;
v7(1:size(vect_A7,1))=vect_A7;
v8(1:size(vect_A8,1))=vect_A8;
v9(1:size(vect_A9,1))=vect_A9;
v10(1:size(vect_A10,1))=vect_A10;
v11(1:size(vect_A11,1))=vect_A11;
v12(1:size(vect_A12,1))=vect_A12;
v13(1:size(vect_A13,1))=vect_A13;
v14(1:size(vect_A14,1))=vect_A14;
v15(1:size(vect_A15,1))=vect_A15;
v16(1:size(vect_B1,1))=vect_B1;
v17(1:size(vect_B2,1))=vect_B2;
v18(1:size(vect_B3,1))=vect_B3;
v19(1:size(vect_B4,1))=vect_B4;
v20(1:size(vect_B5,1))=vect_B5;

```

```
v21(1:size(vect_B6,1))=vect_B6;
v22(1:size(vect_B7,1))=vect_B7;
v23(1:size(vect_B8,1))=vect_B8;
v24(1:size(vect_B9,1))=vect_B9;
v25(1:size(vect_B10,1))=vect_B10;
v26(1:size(vect_B11,1))=vect_B11;
v27(1:size(vect_B12,1))=vect_B12;
v28(1:size(vect_B13,1))=vect_B13;
v29(1:size(vect_B14,1))=vect_B14;
v30(1:size(vect_B15,1))=vect_B15;
v31(1:size(vect_C1,1))=vect_C1;
v32(1:size(vect_C2,1))=vect_C2;
v33(1:size(vect_C3,1))=vect_C3;
v34(1:size(vect_C4,1))=vect_C4;
v35(1:size(vect_C5,1))=vect_C5;
v36(1:size(vect_C6,1))=vect_C6;
v37(1:size(vect_C7,1))=vect_C7;
v38(1:size(vect_C8,1))=vect_C8;
v39(1:size(vect_C9,1))=vect_C9;
v40(1:size(vect_C10,1))=vect_C10;
v41(1:size(vect_C11,1))=vect_C11;
v42(1:size(vect_C12,1))=vect_C12;
v43(1:size(vect_C13,1))=vect_C13;
v44(1:size(vect_C14,1))=vect_C14;
v45(1:size(vect_C15,1))=vect_C15;
v46(1:size(vect_D1,1))=vect_D1;
v47(1:size(vect_D2,1))=vect_D2;
v48(1:size(vect_D3,1))=vect_D3;
v49(1:size(vect_D4,1))=vect_D4;
v50(1:size(vect_D5,1))=vect_D5;
v51(1:size(vect_D6,1))=vect_D6;
v52(1:size(vect_D7,1))=vect_D7;
v53(1:size(vect_D8,1))=vect_D8;
v54(1:size(vect_D9,1))=vect_D9;
v55(1:size(vect_D10,1))=vect_D10;
v56(1:size(vect_D11,1))=vect_D11;
v57(1:size(vect_D12,1))=vect_D12;
v58(1:size(vect_D13,1))=vect_D13;
v59(1:size(vect_D14,1))=vect_D14;
v60(1:size(vect_D15,1))=vect_D15;
v61(1:size(vect_E1,1))=vect_E1;
v62(1:size(vect_E2,1))=vect_E2;
v63(1:size(vect_E3,1))=vect_E3;
v64(1:size(vect_E4,1))=vect_E4;
v65(1:size(vect_E5,1))=vect_E5;
v66(1:size(vect_E6,1))=vect_E6;
v67(1:size(vect_E7,1))=vect_E7;
v68(1:size(vect_E8,1))=vect_E8;
v69(1:size(vect_E9,1))=vect_E9;
v70(1:size(vect_E10,1))=vect_E10;
v71(1:size(vect_E11,1))=vect_E11;
v72(1:size(vect_E12,1))=vect_E12;
v73(1:size(vect_E13,1))=vect_E13;
v74(1:size(vect_E14,1))=vect_E14;
v75(1:size(vect_E15,1))=vect_E15;
v76(1:size(vect_F1,1))=vect_F1;
v77(1:size(vect_F2,1))=vect_F2;
v78(1:size(vect_F3,1))=vect_F3;
v79(1:size(vect_F4,1))=vect_F4;
v80(1:size(vect_F5,1))=vect_F5;
v81(1:size(vect_F6,1))=vect_F6;
v82(1:size(vect_F7,1))=vect_F7;
v83(1:size(vect_F8,1))=vect_F8;
v84(1:size(vect_F9,1))=vect_F9;
v85(1:size(vect_F10,1))=vect_F10;
v86(1:size(vect_F11,1))=vect_F11;
v87(1:size(vect_F12,1))=vect_F12;
```

```

v88(1:size(vect_F13,1))=vect_F13;
v89(1:size(vect_F14,1))=vect_F14;
v90(1:size(vect_F15,1))=vect_F15;
v91(1:size(vect_G1,1))=vect_G1;
v92(1:size(vect_G2,1))=vect_G2;
v93(1:size(vect_G3,1))=vect_G3;
v94(1:size(vect_G4,1))=vect_G4;
v95(1:size(vect_G5,1))=vect_G5;
v96(1:size(vect_G6,1))=vect_G6;
v97(1:size(vect_G7,1))=vect_G7;
v98(1:size(vect_G8,1))=vect_G8;
v99(1:size(vect_G9,1))=vect_G9;
v100(1:size(vect_G10,1))=vect_G10;
v101(1:size(vect_G11,1))=vect_G11;
v102(1:size(vect_G12,1))=vect_G12;
v103(1:size(vect_G13,1))=vect_G13;
v104(1:size(vect_G14,1))=vect_G14;
v105(1:size(vect_G15,1))=vect_G15;
v106(1:size(vect_H1,1))=vect_H1;
v107(1:size(vect_H2,1))=vect_H2;
v108(1:size(vect_H3,1))=vect_H3;
v109(1:size(vect_H4,1))=vect_H4;
v110(1:size(vect_H5,1))=vect_H5;
v111(1:size(vect_H6,1))=vect_H6;
v112(1:size(vect_H7,1))=vect_H7;
v113(1:size(vect_H8,1))=vect_H8;
v114(1:size(vect_H9,1))=vect_H9;
v115(1:size(vect_H10,1))=vect_H10;
v116(1:size(vect_H11,1))=vect_H11;
v117(1:size(vect_H12,1))=vect_H12;
v118(1:size(vect_H13,1))=vect_H13;
v119(1:size(vect_H14,1))=vect_H14;
v120(1:size(vect_H15,1))=vect_H15;
v121(1:size(vect_I1,1))=vect_I1;
v122(1:size(vect_I2,1))=vect_I2;
v123(1:size(vect_I3,1))=vect_I3;
v124(1:size(vect_I4,1))=vect_I4;
v125(1:size(vect_I5,1))=vect_I5;
v126(1:size(vect_I6,1))=vect_I6;
v127(1:size(vect_I7,1))=vect_I7;
v128(1:size(vect_I8,1))=vect_I8;
v129(1:size(vect_I9,1))=vect_I9;
v130(1:size(vect_I10,1))=vect_I10;
v131(1:size(vect_I11,1))=vect_I11;
v132(1:size(vect_I12,1))=vect_I12;
v133(1:size(vect_I13,1))=vect_I13;
v134(1:size(vect_I14,1))=vect_I14;
v135(1:size(vect_I15,1))=vect_I15;

x=[v1';v2';v3';v4';v5';v6';v7';v8';v9';v10';v11';v12';v13';v14';v15';v16';v17';v18';v19';v20';v21';v22';v23';v24';v25';v26';v27';v28'
;v29';v30';v31';v32';v33';v34';v35';v36';v37';v38';v39';v40';v41';v42';v43';v44';v45';v46';v47';v48';v49';v50';v51';v52';v53';v54'
;v55';v56';v57';v58';v59';v60';v61';v62';v63';v64';v65';v66';v67';v68';v69';v70';v71';v72';v73';v74';v75';v76';v77';v78';v79';v80';v
81';v82';v83';v84';v85';v86';v87';v88';v89';v90';v91';v92';v93';v94';v95';v96';v97';v98';v99';v100';v101';v102';v103';v104';v105'
;v106';v107';v108';v109';v110';v111';v112';v113';v114';v115';v116';v117';v118';v119';v120';v121';v122';v123';v124';v125';v126'
;v127';v128';v129';v130';v131';v132';v133';v134';v135'];]
fid = fopen('x.txt','wt')
dlmwrite('x.txt',x,'delimiter','\t','precision','%3.1d');
fclose(fid)
[posit_NN dist_NN]=kNearestNeighbors(x,3)

N=15;
k=3;
accur=0;
for i=1:9
    for j=((N+1)/2+1+(i-1)*N):i*N
        cls_ass=posit_NN(j,1:k);
    end
end

```

```
lk=0;
for n=1:k
    if 1+(i-1)*N<=cls_ass(n)&cls_ass(n)<=i*N
        lk=lk+1;

        end
    end

    if lk>=(k+1)/2
        accur=accur+1;

        end
    end
a=accur
end
accuracy=(accur/63)*100
```

## APPENDIX H SOURCE CODE FOR CLUSTER KNN

```

NNN_NEW()

class_number=9;
class_size=[10 10 10 10 10 10 10 10];
class_size_t=[15 15 15 15 15 15 15 15];

k=1;
control=0;
size_subclass=zeros(class_number,max(class_size));           %%%%%%%%%111 class_size become max class_size

fid=fopen('x.txt','r');
hole_data=fscanf(fid,'%17f');

dim=size(hole_data,1)/(15*9);

hole_data=reshape(hole_data, dim,sum(class_size_t));
hole_data=hole_data';
size(hole_data);

data_t=hole_data(1:class_size(1),:);
for tr=1:(class_number-1)
    data_t=[data_t;hole_data(1+sum(class_size_t(1:tr)):class_size(tr+1)+sum(class_size_t(1:tr
end

data_control=hole_data(class_size(1)+1:class_size_t(1),:);
for tr=1:(class_number-1)

data_control=[data_control;hole_data(1+sum(class_size_t(1:tr))+class_size(tr+1):class_size_t(tr+1)+sum(class_size_t(1:tr),:);
end

fclose(fid);

dim=size(data_t,2);

incream=1;
check_part=1000;
while(check_part>0 & incream<9) % check that the coefficient is never negative
    var1=sum(sum((data_t-ones(sum(class_size),1)*mean(data_t)).^2)); %%%%%%%%%111 class_size have to be change
    var2=0.5*var1;

for cn=1:class_number
    control=0;
    x=data_t( sum(class_size(1:cn-1))+1:sum(class_size(1:cn)),:);
    var1=sum(sum((x-ones(class_size(cn),1)*mean(x)).^2));
    var_class_i(cn)=var1;
    var1=2*var1;

for k=1:(class_size(cn)-1)
    [y,b,c]=kMeansCluster(x,k,1);

```



```

size_subclass(cn,:)=zeros(1,max(class_size));
for n=1:k
    [Rf,Cf,Vo]=find(b==n);
    size_subclass(cn,n)=sum(b==n);
    contro_var(n)=sum(sum((x(find(b==n,:))-ones(sum(Vo),1)*c(n,:).^2));
end
var2=var1;
var1=sum(contro_var(1:k));
var_list(k)=var1;
variation_var_list(cn,k)=abs(((var1-var2)/max(var2,var1)));
if abs((var1-var2)/max(var2,var1))<coefficient(cn)
    control=control+1;

end
if abs((var1-var2)/max(var2,var1))>= coefficient(cn) % choose the threshold
    control=0;

end

if (control >= 2 | var1==0) % better to take control>=2
    cn;
    k;
    break;
end
end
subclass_number(cn)=k;

var_class_f(cn)=var1;
if cn==1
    diction_class=[y,cn*ones(class_size(cn),1)];
    repre_class=c;

else
    diction_class=[diction_class;y,cn*ones(class_size(cn),1)];
    repre_class=[repre_class;c];

end

end

[class_assig,
occur,erreur_position,time_spend_NN,trust_coefficient,trust_coefficient2]=classify_control_data(repre_class,subclass_number,d
ata_t,class_size,size_subclass,1);
class_aff=ones(1,class_size(1));
for p=2:class_number
    class_aff=[class_aff, p*ones(1,class_size(p))];
end
[r c v]=find(class_assig~=class_aff);
if sum(v)~=0
    erreur_location=unique(class_aff(c));
    coefficient(erreur_location)=coefficient_reduction*coefficient(erreur_location);
end

check_part=sum(v);

incream=incream+1;
end

```

```

accuracy=0.5;
class_number=9;
accu=0;
h1=1;
Max_Accu=0;
for co=1:-0.5:0
    co
    for cor=1:-0.5:0
        h1=1;
        coefficient=co*ones(class_number,1);
        coefficient_reduction=cor;
        while(accuracy<0.99 & h1<4)

            NNN_New;

            [class_assign, accuracy ,erreur_position, time_spend_NN, trust_coefficient,trust_coefficient2]=
            classify_control_data(repre_class,subclass_number,data_control,class_size_t-class_size,size_subclass,1);

            if accuracy>=Max_Accu
                Max_Accu=accuracy
                subclass_number
                save test repre_class subclass_number size_subclass ;
                co_max=co
                cor_max=cor
            end

            h1=h1+1;
            accu=accu+accuracy;

        end

    end
end
end

```