DEVELOPMENT OF A FAST LOGO RECOGNITION SYSTEM USING GPU

LEE KEAN LIN

A project report submitted in partial fulfilment of the requirements for the award of the degree of Bachelor (Hons.) of Mechatronics Engineering

Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

May 2011

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature : _____

Name : _____

ID No. : _____

Date : _____

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"Development of a Fast Logo Recognition System Using GPU"** was prepared by **LEE KEAN LIN** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of (Hons.) Mechatronics Engineering at University Tunku Abdul Rahman.

Approved by,

Signature : _____

Supervisor : Prof. Dr. Tay Hong Haur

Date : _____

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Prof. Dr. Tay Yong Haur for his invaluable advice, guidance and his enormous patience throughout the development of the research.

DEVELOPMENT OF A FAST LOGO RECOGNITION USING GPU

ABSTRACT

Recently, the pattern recognition becomes more important and well developed due to the technology development. Many techniques and methods are proposed to improve and enhance the efficiency and accuracy in this field. For the logo recognition, the database becomes larger and the recognition system needed to be efficiency, accuracy and fast. Therefore, in this final year project, we used SURF (Speed up Robust Feature) descriptor to extract the feature points of the logo and used the vocabulary tree concept as our image retrieval method to search over 240 images in our logo database. The image retrieval system contains bag of words model and cosine law equation to determine the similarity of the images. GPU will be added to accelerate the speed in matching.

TABLE OF CONTENTS

DECLARATION **ii**
APPROVAL FOR SUBMISSION **iii**
ACKNOWLEDGEMENTS **v**
ABSTRACT **vi**
TABLE OF CONTENTS **vii**
LIST OF TABLES **xi**
LIST OF FIGURES **xiii**
LIST OF SYMBOLS / ABBREVIATIONS **xvi**
LIST OF APPENDICES **xvii**

CHAPTER

1 INTRODUCTION **1**
1.1 Background 1
1.2 Aims and Objectives 2

2 LITERATURE REVIEW **3**
2.1 Application 3
2.2 Feature Extraction 4
2.2.1 SIFT 5
2.2.2 SURF 10
2.2.3 Comment 14
2.3 Indexing and Matching 15
2.3.1 Artificial Neural Network 15
2.3.2 SVM 17

2.3.3 ANN/KNN KD Tree 19
2.3.4 Vocabulary Tree 21
2.3.5 Comment 23

3 SYSTEM DESCRIPTION 24
3.1 Prototyping Process 24
3.2 Conceptual Design 26
3.2.1 Training Stage 27
3.2.2 Application Stage 28
3.3 Graphic User Interface (GUI) 28

4 TRAINING STAGE 34
4.1 Training Stage Overview 34
4.2 Feature Extraction 36
4.2.1 Image Name Recordation 37
4.2.2 Building of Feature Space 39
4.3 Building of Vocabulary Tree 40
4.3.1 Information Loading 42
4.3.2 Feature Space Loading 42
4.3.3 Matrix Construction 43
4.3.4 Hierarchical K-means Clustering 44
4.4 Feature Extraction & KNN KD Tree Search 45
4.4.1 Load Vocabulary Tree 47
4.4.2 Image Name Recordation 48
4.4.3 Display Current Position Image Name 48
4.4.4 Resize Image 48
4.4.5 SURF Feature Description 49
4.4.6 Data Construction 49
4.4.7 KNN KD Tree Search 49
4.5 TF Computational 50
4.5.1 Load Data 52
4.5.2 Visual Word Initialization 52
4.5.3 Matrix Construction 53

	4.5.4	TF Computational	53
4.6	IDF Computational		55
	4.6.1	Load Data	56
	4.6.2	IDF Computation	56
4.7	TF-IDF Computational		57
	4.7.1	Load Data	59
	4.7.2	Matrix Construction	59
	4.7.3	TF-IDF Score	61
	4.7.4	Recordation	63
	4.7.5	Calculation	64
5	**Application Stage**		**66**
5.1	Application stage overview		66
5.2	Initialization		68
5.3	Feature Extraction		70
5.4	KNN KD Tree Search		71
5.5	TF-IDF Weighting		71
5.6	Cosine Law Matching		74
5.7	GPU Acceleration		78
6	**EXPERIMENTAL RESULTS AND ANALYSIS**		**79**
6.1	Experiment Setup		79
	6.1.1	First Experiment Setup	79
	6.1.2	Second Experiment Setup	82
6.2	Result		82
6.3	Effect of the Change of KNN and Visual Word		83
6.4	Performance Measurement		85
	6.4.1	Accuracy	87
	6.4.2	Detection Rate	87
	6.4.3	False Alarm Rate	88
	6.4.4	False Negative Rate	88
	6.4.5	Sensitivity or True Positive Rate	89
	6.4.6	Specificity or True Negative Rate	89

6.4.7 Precision or Positive Predictive Value (PPV) 90
6.4.8 Negative Predictive Value (NPV) 90
6.4.9 False Discovery Rate (FDR) 91
6.4.10 Receiver Operation Characteristic (ROC) 91
6.4.11 F1 score 92
6.4.12 Threshold Value Selection and Analysis 92
6.5 Speed Analysis 93
6.6 Error Analysis 94

7 CONCLUSION 95
7.1 Comment 95
7.2 Summary 95
7.3 Conclusion 96
7.4 Future Work 96

REFERENCES 97

APPENDICES 99

LIST OF TABLES

TABLE	TITLE	PAGE
	Table 4.1: File for read and write	41
	Table 4.2: File for read and write	46
	Table 4.3: File for read and write	51
	Table 4.4: File for read and write	55
	Table 4.5: File for read and write	58
	Table 5.1: Input Data File in "data" directory file	67
	Table 5.2: Switch case	77
	Table 6.1: The Average similarity value between tested images and its corresponding database image	84
	Table 6.2: The Average similarity value between tested images and all database image	84
	Table 6.3: Distinguish rate (The value in table 6.1 divide by the value in table 6.2)	84
	Table 6.4: The Effect of Changes of Threshold Value	86
	Table 6.5: The Accuracy for Changes in Threshold Value	87
	Table 6.6: The Detection Rate for Changes in Threshold Value	87
	Table 6.7: The False Alarm Rate for Changes in Threshold Value	88
	Table 6.8: The False Negative Rate for Changes in Threshold Value	88
	Table 6.9: The Sensitivity for Changes in Threshold Value	89

Table 6.10: The Specificity for Changes in Threshold Value 89

Table 6.11: The Precision for Changes in Threshold Value 90

Table 6.12: The NPV for Changes in Threshold Value 90

Table 6.13: The FDR for Changes in Threshold Value 91

Table 6.14: The F1 Score for Changes in Threshold Value 92

Table 6.15: The Performance Measurement for Each Threshold Value 92

Table 6.16: The Searching Speed for Each Model for 240 database image 93

LIST OF FIGURES

FIGURE	TITLE	PAGE
Figure 2.1:	Diagram showing the blurred images at different scales, and the computation of the difference-of-Gaussian images(Lowe 2004)	6
Figure 2.2:	The operation of non maximum suppression(Lowe 2004)	7
Figure 2.3:	Histogram of the gradient magnitude and orientation around the key point (Lowe, 2004)	8
Figure 2.4:	SIFT feature descriptor and 64 dimension vector (Lowe 2004)	9
Figure 2.5:	SIFT descriptor (Lowe, 2004)	9
Figure 2.6:	Lattice points and Haar filter	11
Figure 2.7:	Orientation Assignment(Paul, 2009)	12
Figure 2.8:	Sliding window and its subregion (Paul, 2009)	12
Figure 2.9:	Haar filter for subregion(Paul, 2009)	13
Figure 2.10:	The whole responses in the sliding window(Paul, 2009)	14
Figure 2.11:	SURF descriptor (internet)	14
Figure 2.12:	Nueral Network	16
Figure 2.13:	Operation of the separate line	18
Figure 2.14:	Scattering way in VC dimension	18
Figure 2.15:	KD tree	20

Figure 2.16: Hierarchical k-mean cluster with branch factor 3 (Nister, 2006) 21

Figure 3.1: Prototyping Process 24

Figure 3.2: System Overview 26

Figure 3.3: Graphic User Interface for logo recognition system 29

Figure 3.4: The loaded image was shown in Portion 5 29

Figure 3.5: Result in GUI 30

Figure 3.6: History interface 31

Figure 3.7: Sort operation 31

Figure 3.8: Saving process 32

Figure 3.9: Result in test file 32

Figure 3.10: Clear operation 33

Figure 4.1: Training Stage Overview 34

Figure 4.2: Feature Extraction Section 36

Figure 4.3: Data file generated at Feature Extraction section 37

Figure 4.4: Flow chart of Image Name Recordation 37

Figure 4.5: Flow chart for Building of Feature Space 39

Figure 4.6: Process in Building of Vocabulary Tree 40

Figure 4.7: Flow chart of Information Loading 42

Figure 4.8: Flow chart of Feature Space Loading 43

Figure 4.9: Flow chart of Matrix Construction for Feature Space 44

Figure 4.10: Process Flow of Feature Extraction & KNN KD Tree Search 45

Figure 4.11: Process Flow of Feature Extraction & KNN KD Tree Search 50

Figure 4.12: Flow Chart for TF computation 54

Figure 4.13: Process flow for IDF Computational 55

Figure 4.14: Flow Chart for IDF computation 56

Figure 4.15: Process flow of TF-IDF computational 57

Figure 4.16: Flow Chart for TF matrix loading 59

Figure 4.17: Flow Chart for IDF computation and loading 60

Figure 4.18: Flow Chart for TF-IDF computation 61

Figure 4.19: Flow Chart for TF-IDF data writing 63

Figure 4.20: Flow Chart for calculation 64

Figure 5.1: Application Stage Overview 66

Figure 5.2: Overview of initialization 68

Figure 5.3: Process flow of feature extraction 70

Figure 5.4: Process flow of TF-IDF weighting 71

Figure 5.5: Process flow of cosine law matching 74

Figure 5.6: Two vectors 74

Figure 5.7: Cosine law computation 75

Figure 5.8: Cosine law computation II 76

Figure 6.1: Tested logo with illumination change 80

Figure 6.2: Tested logo with rotation change 80

Figure 6.3: Tested logo with scale change 81

Figure 6.4: The logo images that not include in database 82

Figure 6.5: The logo images that not edited from database image 82

Figure 6.6: Table of Confusion Matrix 86

Figure 6.7: ROC 91

LIST OF SYMBOLS / ABBREVIATIONS

c_p	specific heat capacity, J/(kg·K)
h	height, m
K_d	discharge coefficient
M	mass flow rate, kg/s
P	pressure, kPa
P_b	back pressure, kPa
R	mass flow rate ratio
T	temperature, K
v	specific volume, m^3
α	homogeneous void fraction
η	pressure ratio
ρ	density, kg/m^3
ω	compressible flow parameter
ID	inner diameter, m
MAP	maximum allowable pressure, kPa
MAWP	maximum allowable working pressure, kPa
OD	outer diameter, m
RV	relief valve

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
APPENDIX A: Result 1		99
APPENDIX B: Result 2		100
APPENDIX C: Programming Code		101

CHAPTER 1

INTRODUCTION

1.1 Background

As the technology of the human being is fast developed, pattern recognition becomes a major issue to overcome the huge information in database. In the field of the pattern recognition, logo recognition becomes famous and important in the human life.

At first, human can implement and know the information of each logo as well. In that background, the logo database was very few only compare with today. Beside these, the manpower was enough to do categorization based on different logo in this field. However, the needs in logo recognition field is lack due to the rapidly change in environment and technology. The raise of the manpower cost and the impact of the huge information because of the internet and global earth effect provided an environment for logo recognition.

As the technology improved, logo recognition is well developed to satisfy the need of the environment. Nowadays, logo recognition was very useful in many areas such as industrial area, commercial area and residential area. Besides that, more methods and techniques have been proposed to perfect the application and function of the logo recognition. It solves the problem that rapidly increases of the logo database and provide an automatic and efficiency system to recognize the logo.

In our system, it can be divided into two stages which are training stage and application stages. In training stage, it includes building of vocabulary tree, KNN KD tree searching algorithm, weighting and normalization. The training stage generated the database type for the application stage. The application stage includes KNN KD tree searching, weighting and normalization and matching to database. Moreover, a graphical user interface (GUI) was designed for the application stages. Afterwards, the matching algorithms will be accelerated by using GPU.

1.2 Aims and Objectives

The objective of this title is developing a system that can recognize the logo in the 80 classes for database and give the result to the user. One class logo has three logo images. In the logo recognition system, the accuracy rate and speed rate always an issue. So, we designed a logo recognize system that have to be robust and withstand variety transformation like translation, rotation and scaling. Beside these, the system has to be more efficiency and fast. A visualize interface is needed to be user friendly and easy to control. Our system is designed for detect one logo in the image. The logo is the main portion in the image with the less influence of the background.

CHAPTER 2

LITERATURE REVIEW

2.1 Application

Nowadays, the logo recognition has been developed well and use in many field. As we mention before at the introduction, it used in residential, commercial, industrial purpose.

For the residential purpose, it used to help the people to get the information of the logo. The Google Goggles provided us to use the image that captured by hand phone send to its server and give the information about that logo. It lets the life becomes convenient when impact by the huge information.

For industrial purpose, the logo recognition helps to upgrade the warehouse system. By using a camera, when the container or box was put in the warehouse, the logo recognition system will recognize the logo on the box and put it in the right position automatically. It increases the efficiency of the work. Besides that, it also applied in the production line that checks the logo on the product.

For the commercial purpose, the document logo classification system and the development of the E business very need the help of logo recognition to categorize the document. Recently, internet becomes popular in everywhere. With the growing of the E business, the documents related need to be categorize for convenient.

2.2 Feature Extraction

Feature extraction and detection for logo recognition is always a very challenging task. In the review of the previous methods, we know that many algorithms and techniques have been proposed during these many years.

There are two types feature extraction descriptors which are global descriptor and local descriptor. For the global descriptor, it can detect and discriminate the wanted features of the different logos from background. The methods of line segment and edge detector, such as Harris operator, hough transform and sobel operator are belong to global descriptor. However, they are insufficient in this field. For example, Harris edge detector cannot be used individually for the feature extraction because it cannot withstand the change of scaling. From here, we can conclude that good feature extraction methods have to invariant to rotation, scaling, and translation and affine transformation. Beside these, it has to give complete and pertinent information to do recognition. Among these methods that mention before, the geometric invariant methods that proposed by Doerman 1993 also cannot fulfil these condition. The method he used is use complex mathematic methods to know the invariants and do recognition. These invariants have their own limits. It cannot be used in large database because of the calculation problems. For Hu's moment invariant, it just used in the global feature extraction because it cannot discriminate well in the local feature. It is a good global feature extraction method to detect the things that no complicated in the feature.

Now, the Fourier descriptor also developed in the field of shape descriptor. It provides a concept that transforms the image in Fourier transform and invariant to transformation. However, its limitation is also same with the moment invariant method. The Log polar transform methods also provide a log polar space that invariant to rotation. But these methods have to combine with the other to ensure its capability. For example, Fourier Merlin transform is the combination of the Fourier transform and log polar transform. The retina coding is a good idea for the artificial neural network purpose. However, the coding method is act as additional methods compare with the direct input vector of the neural network. The efficiency is low and not suitable used at here.

Local descriptor focuses on the one logo features only. For our logo recognition system, we need not to use global descriptor as our feature extraction method because the system requirement is detected one logo with less of the influence of the background. Therefore, the local descriptor is used for our main choice. In local descriptor, the SIFT and SURF descriptor are very famous descriptors and powerful. These two descriptors will discussed at below section.

2.2.1 SIFT

David G. Lowe (1999) proposed a new method for object recognition. It can extract the pertinent information and key point of the logo to do recognition. The key point of an image is some kinds of invariants that its properties are remains unchanged under variety transformation like rotation, illumination, translation, scaling. This method is Scale Invariant Feature Transform (SIFT), which is a scale pyramid approach. It detects the interest point in the image that can be invariants to variety transformation. This method can be said as new milestone to the pattern recognition. SIFT has four major stages which are scale-space extrema detection, keypoint localization, orientation assignment and keypoint descriptor.

At the first stage, scale space extrema is known as the interest points or key points are detected by using difference of Gaussian image. The input image is successive smoothed by applying a Gaussian Kernel to the input image with certain increment of σ.

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{2.1}$$

Where

G(x, y, σ) = Gaussian kernel

σ = variance

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.2}$$

L(x, y, σ) is the convolution of the Gaussian kernel and input image.

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$$
$$= L(x, y, k\sigma) - L(x, y, \sigma) \quad (2.3)$$

The base of the first octave is the original image, the upper layer of it is applied by σ=1.2 Gaussian filter. The following layer is applied by kσ Gaussian filter which the k value depend on how many layer being constructed in an octave. The image will become smoother and more blur from the bottom to top. Therefore, the two successive smoothed images are subtracted together to obtain the difference of Gaussian (DoG).. In this case, we can obtain the DoG between each level in the pyramid. Beside these, the scale of the image is changed to construct another pyramid octave by using same concept. Note that the next octave base layer is the layer that applied 1.2σ Gaussian filter and resampled into half size in the first octave. Therefore, the DoG images that calculated and constructed into pyramid form as shown as figure 4.1. It is known as scale space function. This DoG method is very efficiency and approximately approaches Laplace of Gaussian.

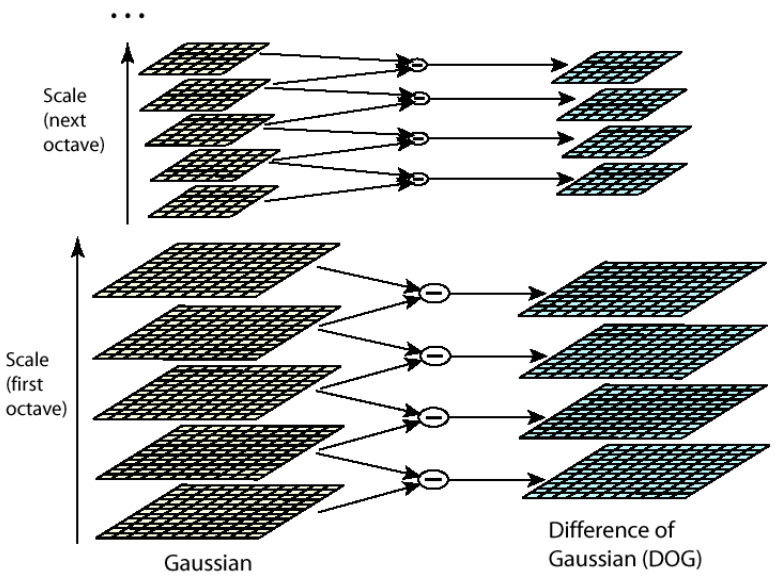

Figure 2.1: Diagram showing the blurred images at different scales, and the computation of the difference-of-Gaussian images(Lowe 2004)

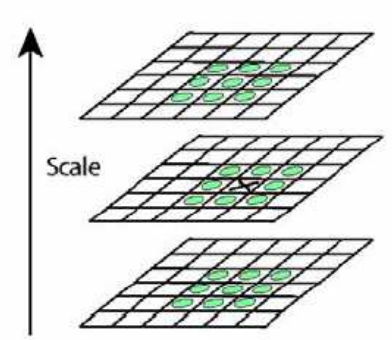

Figure 2.2: The operation of non maximum suppression(Lowe 2004)

After that, maxima and minima of this scale-space function are determined by using the NMS, nonmaximum suppression method. This method is used to compare the pixel intensity in 3x3x3 neighborhood, which is 9 pixel intensity of its upper layer, 9 pixel intensity of its lower layer and 8 pixel intensity of its own layer around it. The extrema can found out in the comparison of them. Thus, the scale space extrema is found out and become a candidate point of SIFT key which is scale invariant.

The second stage is key point elimination. There are two conditions to be a SIFT key from candidate key point which are the keypoints with low contrast are removed and responses along edges are eliminated. These can be achieved by using Taylor series and hessian matrix.

The third stage is orientation assignment for the SIFT key. For each image sample, L (x, y), at this scale, the gradient magnitude, m (x, y) and orientation, θ (x, y) are computed by using the pixel differences.

$$m(x, y) = \sqrt{(L(x+1, y)-L(x-1, y))^2 + (L(x, y+1)-L(x, y-1))^2} \quad (2.4)$$

where

m(x, y) = gradient magnitude

$$\theta(x, y) = \tan^{-1}((L(x, y+1)-L(x, y-1))/(L(x+1, y)-L(x-1, y))) \quad (2.5)$$

where θ (x, y) = orientation

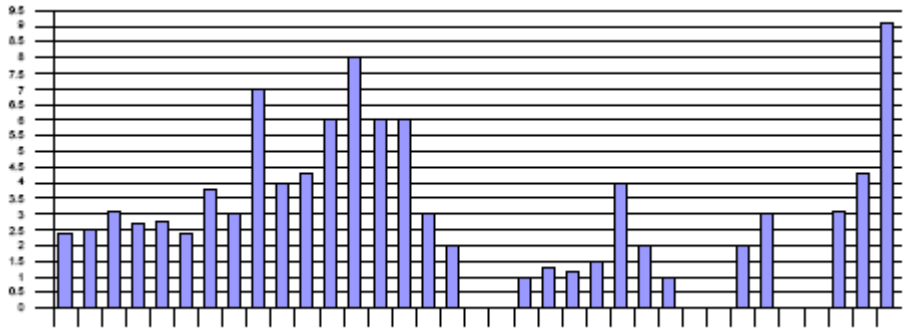

Figure 2.3: Histogram of the gradient magnitude and orientation around the key point (Lowe, 2004)

In the Figure, it show that the histogram of the gradient magnitude and its orientation in the neighborhood of the keypoint (using the Gaussian image at the closest scale to the keypoint's scale). In the figure, it consists of 36 bins and each bin hold 10 degree. It covers 360 degree around the key point. The contribution of each neighboring pixel is weighted by the gradient magnitude and a Gaussian window with σ is 1:5 times the scale of the keypoint. Peaks in the histogram correspond to dominant orientations. A separate keypoint is created for the direction corresponding to the histogram maximum, and any other direction within 80% of the maximum value. All the properties of the keypoint are measured relative to the keypoint orientation, this provides invariance to rotation.

Last stage is about the keypoint descriptor. Each key is shown as a square, with a line from the center to one side of the square indicating orientation. The size of square is 8x8s and expended from the interest point. To the main direction of the axis can be set up in the coordinates of each feature point, SIFT feature point selection in a size and scale the corresponding square area into 16 blocks, eight along the direction of each piece of statistical proportion, so the formation of the 128 feature points dimensional feature vector, the normalized image intensity change is completed; and also the formation of 128-dimensional vector, normalized a complete contrast of the change and intensity change.

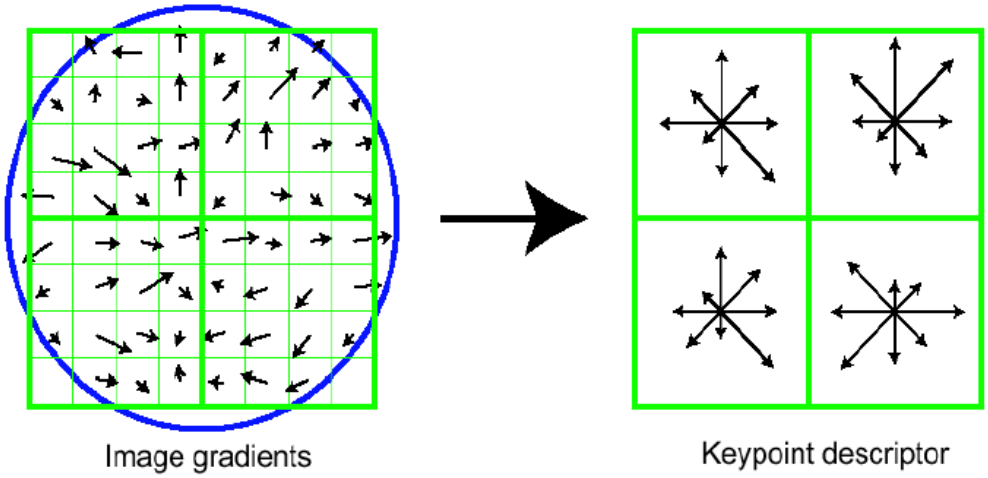

Figure 2.4:SIFT feature descriptor and 64 dimension vector (Lowe 2004)

Figure 2.5:SIFT descriptor (Lowe, 2004)

PCA SIFT (Yank Ke and R. Sukthankar, 2004) and GLOH are the variant of the SIFT have also been used for image recognition. PCA SIFT is the SIFT method that applied Principal Component Analysis. PCA is a standard technique for dimensionality reduction. As we know that SIFT technique has four major stages, the different of PCA SIFT and SIFT is the fourth stage. In the fourth stage of PCA SIFT, the input vector is created by cantenating the horizontal and vertical gradient maps for the 41x41 patch centered at the keypoint which has 2x39x39=3042 elements. Therefore, the vectors are applied with the PCA to convert the high dimension

samples into low dimensional (20 dimension) feature space. It required less storage and speed up the matching. However, the information of the feature does not be affected so much. Another method GLOH, is an extension of the SIFT descriptor designed to increase its robustness and distinctiveness. Its keypoints are detected by using Harris operator and its histogram is computed for 17 location and 16 orientation bins in a log polar location grid. The log polar form has a fovea of high resolution and decrease when away from the center. It uses radial distance and angle to be variable in the polar coordinate system. PCA is used to reduce the dimension to 128. By using log polar form, GLOH has rotational translation scale (RTS) invariant features and robust to rotation.

2.2.2 SURF

Speed up Robust Feature (SURF) is a descriptor that proposed by Herbert Bay 2007. This method is more robust and faster than SIFT according to SIFT. By using partially concept of SIFT, which is scale pyramid, it can perform well in logo recognition. It has three steps in SURF. First, the image was converted to grayscale and computed as integral image type. The concept of integral image is proposed by Viola and Jones 2001, it allow for efficient and fast computation of box –type convolution filters. The integral image is the sum of the block pixel intensity from left to right. The second step is applying a 9x9 filter box to the integral image. This will approximate approach the effect of σ = 1.2 Gaussian Filter.

$$H(X,\sigma)=\begin{bmatrix} L_{xx}(X,\sigma) & L_{xy}(X,\sigma) \\ L_{xy}(X,\sigma) & L_{yy}(X,\sigma) \end{bmatrix} \tag{2.6}$$

$$\begin{cases} L_{xx}(X,\sigma)=\dfrac{\partial^2}{\partial x^2}g(\sigma) \\ L_{yy}(X,\sigma)=\dfrac{\partial^2}{\partial y^2}g(\sigma) \\ L_{xy}(X,\sigma)=\dfrac{\partial^2}{\partial x\partial y}g(\sigma) \end{cases} \tag{2.7}$$

$$\det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \tag{2.8}$$

Next, the SURF use determinant of Hessian matrix to represent the DOG to find the extrema in local feature. The same step will be repeated by changing the different size filter box which is 15x15, 21x21, and 27x27 sizes. This will construct a four level image pyramid with different scale. The base of the pyramid is the image that applied by 9x9 filter box. The scale of the image will increase when the level of pyramid is increased. Therefore, the NMS, nonmaximum suppression method is used in 3x3x3 neighborhood. This step is same as the SIFT. The only different is using the NMS to find the maximum determinant of the Hessian matrix in 3x3x3 neighborhood. This is known as the interest point. The last step is assigning the orientation and direction to make them invariant to rotation, also the illumination.

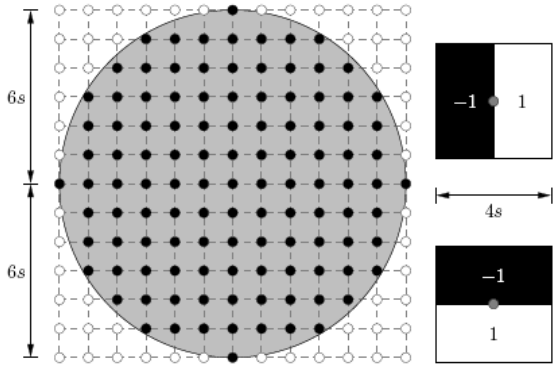

Figure 2.6: Lattice points and Haar filter

The Haar wavelet responses of size 4s in x and y direction are calculated within a circular neighborhood 6s around the interest point, where s is represented as scale that the interest point detected. The dominant orientation is the calculation of the sum of all responses within a sliding orientation window of size π/3.

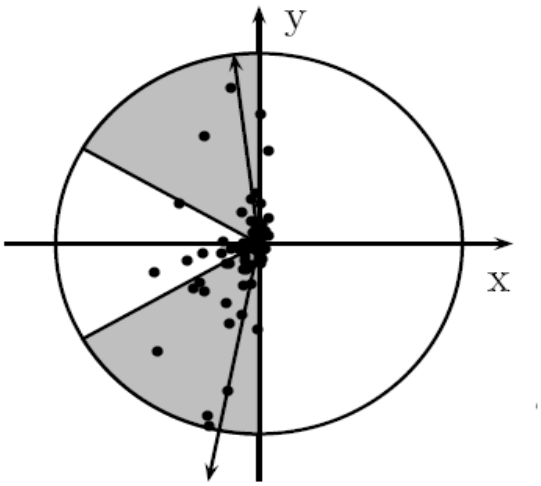

Figure 2.7: Orientation Assignment(Paul, 2009)

The sliding window will be rotate 5 degree a step and select the largest vector in the circular neighborhood. The direction of the largest vector is defined as its main direction. Therefore, construct a square region 20x20s as the interest point is act as its origin point. Thus, the direction of the largest vector is shown by a line from center to the side of the square. Its direction follows the direction of the largest vector and the side of the square is perpendicular to this line.

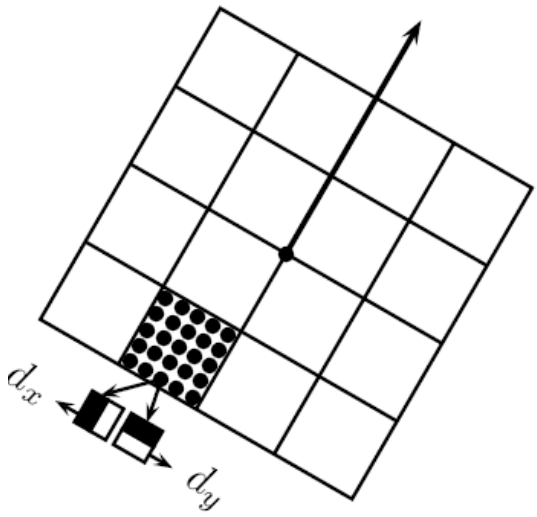

Figure 2.8: Sliding window and its subregion (Paul, 2009)

The size of the square that we mention before is 20x20s size but depend on which scale that the interest point detected. Then, the square region is divided into 4x4 subregions. This preserves important spatial information.

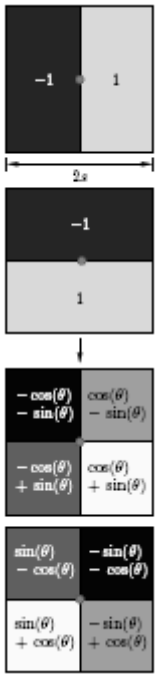

Figure 2.9: Haar filter for subregion(Paul, 2009)

For every subregion, Haar wavelet of size 2s are calculated for 5x5 regularly distributed sample point and weighted by a Gaussian centred at the sampled point with standard deviation 3.3*s* to increase the robustness towards geometric deformations and localization error. The *dx* and *dy* are used to denote the Gaussian weighted Haar wavelet responses in *x* and *y* directions and aligned to the primary orientation.

$$V = (\sum d_x, \sum d_y, \sum | d_x |, \sum | d_y |) \tag{2.9}$$

For each of the 4 × 4 sub-regions, *dx*, |*dx*|, *dy*, |*dy*| are used to denote the sums of the *x* and *y* responses of all the 5×5 sampled points. And for all the 4×4 sub-regions, each subregion contributes four values to the descriptor vector leading to an overall vector of length 4x4x4= 64. The wavelet responses are invariant to the

contrast and illumination change. Therefore, the third step is found out the rotation, illumination invariant for the SURF descriptor.

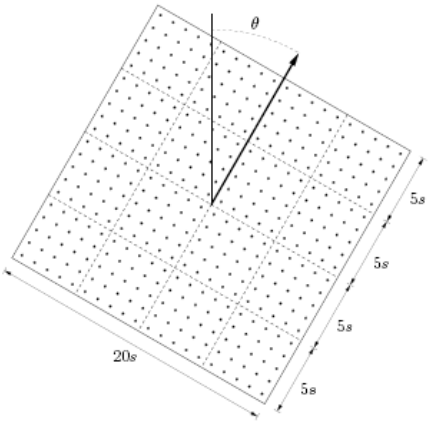

Figure 2.10: The whole responses in the sliding window(Paul, 2009)

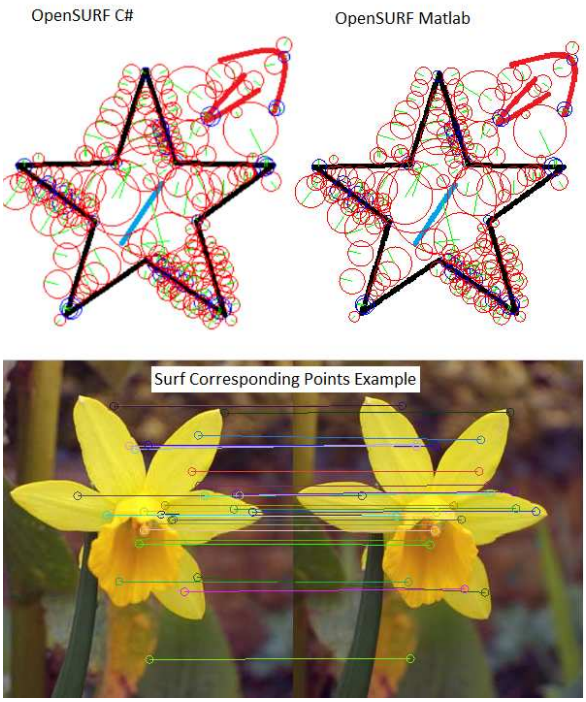

Figure 2.11: SURF descriptor (internet)

2.2.3 Comment

After review the different methods of feature extraction, SIFT is a great milestone for the pattern recognition field. It is a feature extraction technique that nearly perfect compare to the others. It uses the DoG to achieve the effect of LoG. It is very fast

and simple. Actually, the first stage of SIFT can use the Harris operator and Sobel operator such as edge detector to replace the DoG. But, they are slower compare with the DoG. The accuracy rate and speed are the advantages of it. However, the feature that extracted is in high dimensional way. So, how to improve and lower down the dimension is an issue. Therefore, PCA SIFT and GLOH were proposed to improve the SIFT. Although they use PCA to lower down the dimension to increase the speed, the PCA SIFT decreases its discriminate ability. GLOH is very accuracy but it consumed much time in computational.

By using the scale pyramid concept and partially concept of SIFT, SURF was proposed to have more robustness to transformations. The only weakness of SURF is the invariant of transformation is less efficient compare to SIFT. However, its ability is good for the many cases. That's why we use SURF to do the feature extraction.

There are factors that need to be careful in feature extraction. The quality of the image, the size of the image and the noise will affect the functionality of the feature extraction. If the quality of the image is not good and size of the image is too small, the feature that extracted and accuracy might be influence.

2.3 Indexing and Matching

The growing of the information in the database, indexing is more important to save the time in matching. Here, there are the four main methods, artificial neural network, SVM, KNN KD Tree search and vocabulary tree in this part.

2.3.1 Artificial Neural Network

Artificial neural network is represented as computational model that inspired by the structure and function of the human brain neural network. In this field, GRNN or BPNN usually act as main role in the artificial neural network for the pattern recognition. GRNN is known as Generalize Regressive Neural Network provided by

Specht 1991. It consists of the concept of the back propagation and radial basis function. The generalization and regression properties make the network can learn by itself. Beside these, it is a very useful tool to perform prediction and comparison.

As we know that an artificial neural network contains three layers, which are input layer, hidden layer and output layer. First, the input nodes in the input layer receive the weights of the image (pixel intensity of the image or the feature that extracted). It consists of two hidden layers. The first hidden layer in the GRNN contains the radial units where the second hidden layer contains units that help to estimate the weighted average. This is a specialized procedure. Each output has a special unit assigned in this layer that forms the weighted sum for the corresponding output. The weighted sum must be divided through by the sum of the weighting factors to get the weighted average from the weighted sum and calculated by a single special unit. The output layer then performs the actual divisions (using special division units). Hence, the second hidden layer always has exactly one more unit than the output layer. In regression problems, typically only a single output is estimated, and so the second hidden layer usually has two units.

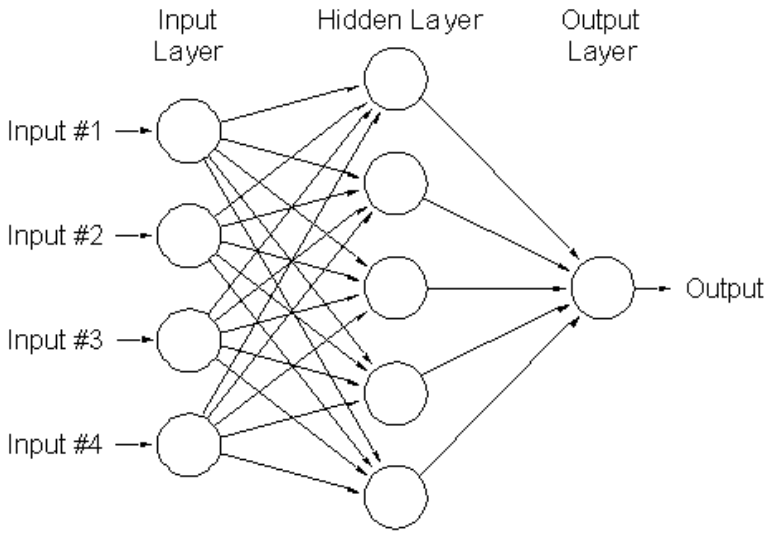

Figure 2.12: Nueral Network

The GRNN copies the training cases into the network to be used to estimate the response on tested points. This is its regressive properties. The output is

estimated using a weighted average of the outputs of the training cases, where the weighting is related to the distance of the point from the point being estimated (so that points nearby contribute most heavily to the estimate). Hence, the tested data set was input to the GRNN and the compare with the training data set that had been trained before. Therefore, through the function that calculate the error between them and use it to train again until obtain a optimize result. The error is calculated based on the Euclidean distance between tested and training data. The training set is treated as optimal set and regressive function and the GRNN train the tested weights to approach the training set. A here, the iteration, learning rate and error bound have to defined. Iteration is how much the loops need to be trained in the network. Error bound is the error between tested and training weights. After obtain the error, use it to change the input weights to close the optimal solution. The change is defined as learning rate, which means the magnitude of the change. The smaller the learning rate, the increase the accuracy rate because it tune the changes of the weights precisely.

The GRNN can be modified by assigning radial units that represent clusters rather than each individual training case. This reduces the size of the network and increases execution speed. Centers can be assigned using any appropriate algorithm.

2.3.2 SVM

SVM is known as Support Vector Machine which is one of the supervised methods that used for recognition. It is very famous classifier that developed from the statistic method and learning itself but different with the artificial neural network. It helps to analyze the data, classified pattern and process regression of the data. Recently, SVM becomes a major method for pattern recognition. It solves the problems that faced by content based image Retrieval (CBIR), which is the full expression of the low level features and the link between low level content feature and high level semantic content based image retrieval. Nowadays, the database that used for recognition is increase to very large, it is impossible to categorize manually. SVM is developed and can categorize the input source in the database by learning itself and its regression properties. This concept can be used in classification of the logo.

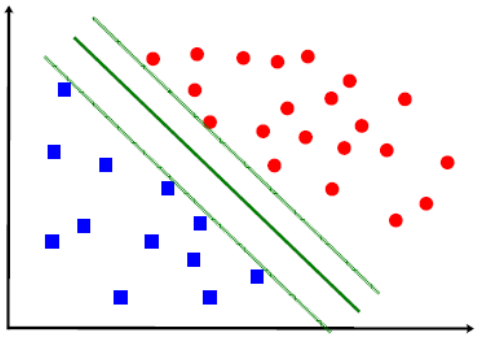

Figure 2.13: Operation of the separate line

The concept of SVM is use a line or plane to divide the different point in the high dimension space. First, the points of the different sample sets are mapped into a high level space. This high level space can be infinite depend on the input sources. Therefore, a hyperplane is found in this high dimension space to separate these points belong to its own set. The hyperplane might not unique. The main purpose of hyperplane is used for classification of the different classes from the input samples by using its regression properties. The hyperplanes are founded based on the maximum width of the margin of two different sets. The width of the margin is depending on the maximum distance that the points of two different sets. If the width of the margin is large, that means that the classification is good for the accuracy implementation for the future. The maximum margin also approaches the structural minimum risk. The larger of the margin, the lower the risk faced. It increases the manipulation of the generalization of the SVM.

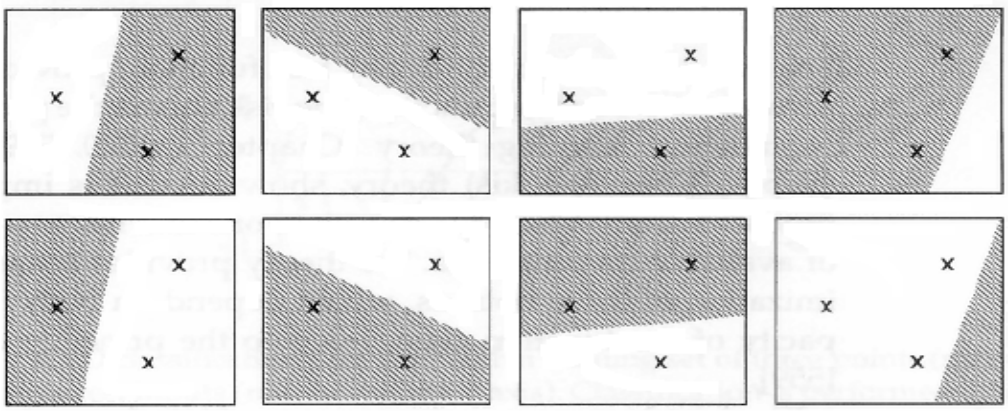

Figure 2.14: Scattering way in VC dimension

The other concept in SVM is the VC dimensions that proposed by Vapnik and Chervonenkis 1968. VC dimension is stand for Vapnik-Chervonenkis dimension, it measures the capacity of a hypothesis space. The capacity of a classification model is a measure of complexity and expressive power and of a set of functions by assessing how wiggly its member can be. It collects all the probability of the functions that can shatter the points. It uses 2^n to show the how many way the points that can be shattered which n is the number of the point in this dimension. It shows the learning capability of the machine. The VC dimension is used for the minimization structural risk because it can estimate the error on future data based by using the training error and VC dimension. Through these concepts, the lower the VC dimension after learning, the structural risk can be minimized.

Thus, SVM can solve the problem of the classification of the high dimensions data sets by its own regressive and learning properties. It is a concept to apply this method in logo recognition after using SURF or SIFT because the dimension of feature extracted by them is very high. Once the features input to the SVM, it will classified it according the classes based on the similarity. SVM is very important in the artificial intelligent field and pattern classification.

2.3.3 ANN/KNN KD Tree

In the nearest neighbour method, there are two popular methods introduced here which are KNN KD tree method and ANN KD tree method. First, KD tree is a data structure for storing a finite set of points from k- dimensional space. The elements stored in the KD tree are high dimensional vectors. At the first level of the tree, the data is split into two halves by a hyper plane orthogonal to a chosen dimension at the threshold value. The split is made in the median according to the greater variance in the data set. It determines the half of the data the query vector belongs. Thus, each of the halves of the data split recursively in the same way to create a fully balanced binary tree. By using this concept, the data point can be found quickly and easily in k dimension space. Based on the KD structure, the ANN, Approximate nearest

neighbour and KNN, k nearest neighbour are used to search the data points by given query data points.

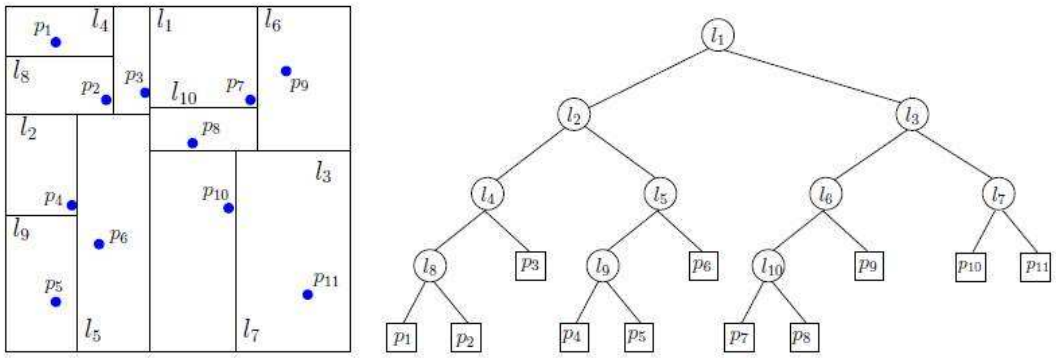

Figure 2.15: KD tree

First, the concept of the KNN is search the k closest data point by given a query point. For example, if k is 1, it will search the data point that closest to the query point. When the k is 2, it searches the nearest data point and the second nearest data point that close to query point. For the nearest neighbour ratio matching method that introduced by Lowe 2004, he used the 2NN search to find the nearest point and second nearest point to the query point in the KD tree which was constructed by 30000 feature point from 100 images. After that, the ratio of the distance of the nearest point and second nearest point was obtained. If the ratio was less than 0.65, the feature point (query point) that extracted from sample image can be said matched with the feature point that extracted from the training image in database. Beside these, Lowe also used the best bin first method to accelerate the speed in indexing. He found the 200 nearest neighbour feature point that close to matched point and treated them as candidate point to perform the matching algorithm. Therefore, nearest neighbour ratio method can be performed between the query point and these candidate points because they had high probability to be matched. This method will not influence the accuracy rate so much and will speed up the process.

Beside the nearest neighbour method, ANN method is similar to the KNN method. The difference is the way it searching. ANN will predicted and search the data point for the query point within the distance. That’s mean if the distance of data

point and the query point is within the distance range, the data point can be said close to the query point. It would care about this data point is the nearest one to the query point. It saves a lot of memory and time. Therefore, it is faster than the KNN method. Liangfu Xia, Feihu Qi, Qianhao Zhou, 2008 applied the ANN KD tree method and the nearest neighbour ratio method and combined together to achieve their target. This combination is more efficiency and faster than the KNN method the mentioned before.

2.3.4 Vocabulary Tree

The concept of the vocabulary tree is first introduced by Nister and Stewenius, 2006. They designed a CD-cover recognition system by using this concept to be their image retrieval system. According to nister, 2006, this system is very fast and able to query the database with 50000 images in 25ms only. This vocabulary tree concept contains four parts which are building the vocabulary tree, nearest neighbour search, scoring and normalized difference.

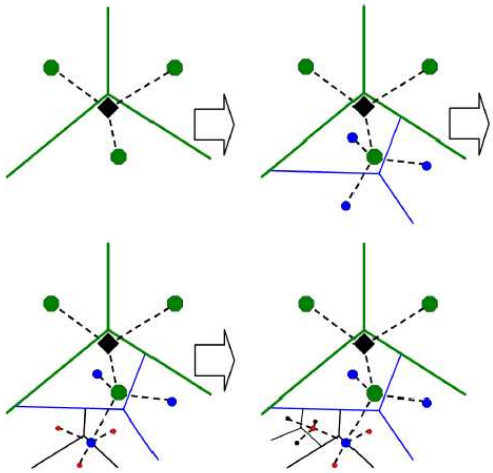

Figure 2.16: Hierarchical k-mean cluster with branch factor 3 (Nister, 2006)

First, they used the hierarchical k means clustering method to cluster the all the feature points that extracted from the database images in KD tree. The

hierarchical k-means cluster method will explore the centroid it found to three centroids when branch factor is 3. This kind of method is the more advance method in clustering method. After that, the centroids was obtained and treated as word. These centroids point in KD tree was known as vocabulary tree. Then, the feature points from two images were applied in nearest neighbour search and if the feature point was closest to the one of the centroids, this feature point can be treated as this centriod. Therefore, these centroids which were close to the feature points were the content of words for the two images.

After nearest neighbour search, weighting or scoring method is processed. Nister used idf concept to do weighting.

$$q_i = n_i w_i \tag{2.10}$$

$$d_i = m_i w_i \tag{2.11}$$

$$w_i = \ln \frac{N}{N_i}, \tag{2.12}$$

q_i is query vector and d_i is database vector while n_i and m_i are indices vector that found in nearest neighbour search. N is number of image in database and N_i is the number of the same centroid found in all database image content. Therefore, the weight, w_i was obtained ant treated as constant value. So, by using the equation 2.10 and 2.11, the q_i and d_i were obtained and used for checking their difference.

$$s(q,d) = \| \frac{q}{\| q \|} - \frac{d}{\| d \|} \| . \tag{2.13}$$

Therefore, the normalized difference method computed the similarity between two images. The value that close to zero mean's that the two images is the same. This method is very efficiency in large database.

2.3.5 Comment

Among these four methods, they have their own advantage points and weak points. For the artificial neural network, it is complicated to be constructed. A lot of parameters have to tune to optimal solution. Although its capability is good but the computational time is slow because of the computational. For SVM, a machine learning method is very famous in recently year. It is a very powerful technique in classification. It developed from the statistic theory and provided in good result. However, it is hard to construct the SVM system. Some of the theory inside hasn't been developed well. For KNN/ANN KD Tree search, ANN is faster than KNN and provides a good result in the common condition. Although they gave a good performance and very fast and efficiency, they gave a poor result relatively when the k dimension is increased. The last method, vocabulary tree is suit for larger database system. It faster and more efficiency than the KNN/ANN KD tree method.

CHAPTER 3

SYSTEM DESCRIPTION

3.1 Prototyping Process

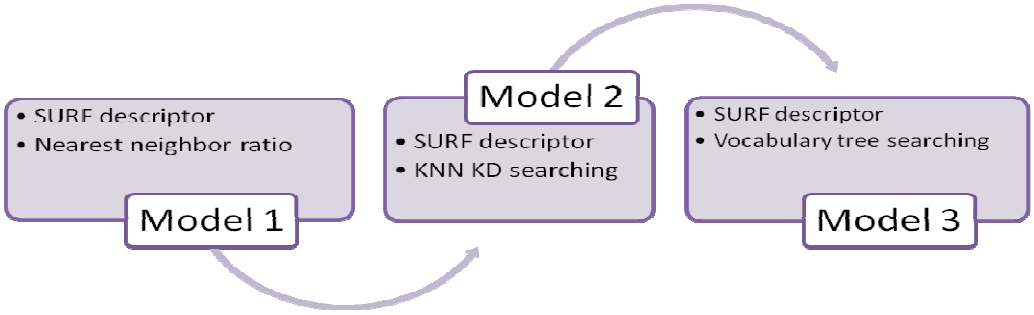

Figure 3.1: Prototyping Process

In our logo recognition system, we designed and upgraded our system from model 1 to mode 3. The SURF descriptor was used as feature extraction method through every model. The image retrieval system is the main changes for every model. The speed of indexing and matching, efficiency and accuracy rate were improved once the model has been upgraded.

At first, in our model 1, the image retrieval system includes two main sections which are breadth first search and nearest neighbour ratio approach. The feature points for each image in database were saved in DAT file. It searched the DAT file that contains the feature points for the image in database. There has a lots

of DAT files to store the each image feature points. The way it searched and opened the file is breath first search. It opened the first image in the each class first. Note that each class have three images. The system opened the DAT file and used the nearest neighbour ratio to find whether this image was matched with query image or not.

In our second model, we used KNN KD Tree search as our image retrieval system. The feature points of all images in database were put into KD tree and the order of these points was set and can be specified to the image it belongs to. A query image is then performs the KNN search in the KD tree to find their nearest neighbour points. Furthermore, the nearest neighbour points found was record according from the nearest point to far point. Therefore, the scoring system was applied. The closest point was scoring the higher mark, the second closest point gained the mark that lower than the first and so on. After that, the highest mark gained for the image in database can be said the image is similar to the query image.

In our final designed system, we used vocabulary tree concept as our main concept in image retrieval system. The detail will be discussed in 3.2 Conceptual Design.

3.2 Conceptual Design

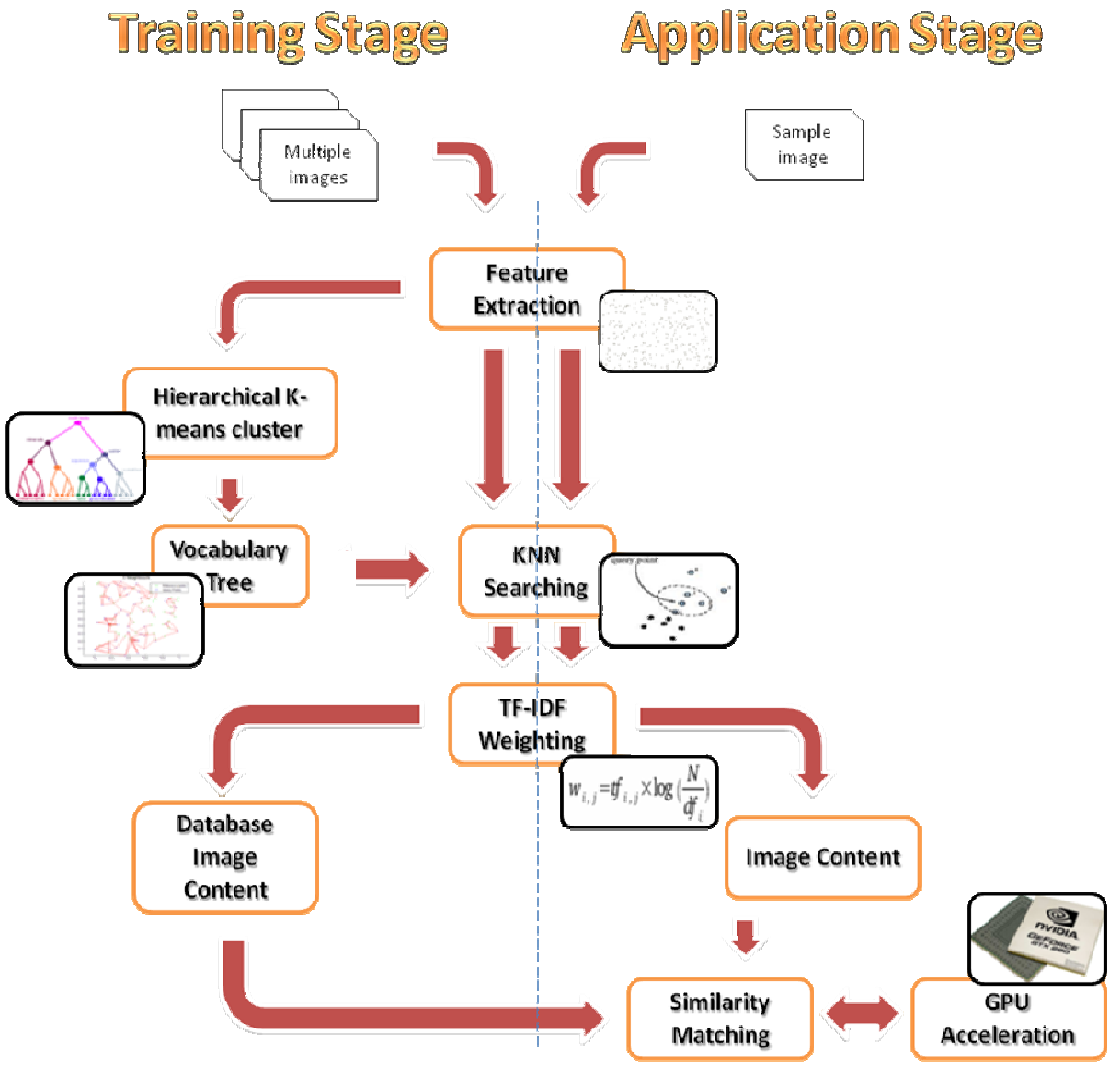

Figure 3.2: System Overview

The Figure 3.2 shows that the system overview of our logo recognition system design. In our system, it can be divided into two stages which are training stage and application stage. In our system, training stage programming and application programming are separated. The training stage generates vocabulary tree and database images content for the application stage. In training stage, there is no graphic user interface and has to start the Visual studio C++ to activate the "Training Stage" program. This can be done in offline mode. However, the application stage which is online mode is provided graphic user interface to make the user convenient

and user friendly. In our design, we used OPENSURF SURF descriptor as our main method in feature extraction. Beside this, we also used the OPENCV 2.1 in our project. They are open source program and freely to use. The description will brief at the below.

3.2.1 Training Stage

For the training stage, it provided the parameter tuning for the building of vocabulary tree and the database images content. From the Figure 3.2, the images that treated as database images were input to the system. We provided 80 classes of logos which 240 logo images. Note that one class of logo consists of three logo images. These logo images were stored in a specified directory file, and then system will automatically run the feature extraction process for these images. The image format must be JPEG format or else the system cannot read them.

We used SURF descriptor from OPENSURF to extract the feature point. After feature extraction, all feature points was found and saved in DAT file. The system read this DAT file and perform hierarchical k-means cluster to find out the centroids of each cluster. These centroids were treated as visual words and save in the vocabulary tree.

After that, same procedure was repeated. Each image feature points were extracted again and performed KNN Search for each image feature points. The vocabulary tree can be said like dictionary. These unknown feature points of the query image will find the nearest neighbour centroids that we have generated before, and use them to represent these unknown feature points. For example, an unknown word was put into dictionary and dictionary found the word for this unknown word. Therefore, the word found in dictionary can be represented the unknown word. This is how it works in the KNN searching. So, the content of each image is made up of these visual words (centroids) found.

TF-IDF (Term frequency and inverse term frequency) is a weighting method that used for weighting the visual words that the image has. This process emphasizes the feature of the image. For example, a document contains many “diamond” words and “the” words. The “diamond” word is rare to see in others document. However, “the” is very common in all the documents. Therefore, we can say that the “diamond” is the feature of this document and ignore the “the” word. It calculates the visual word that frequently occurred and rarely occur in others image. So, TF-IDF weighting method show how is important of the visual word in the image as its feature. The larger the TF-IDF value, the more important the visual word for the image. Afterwards, save these TF-IDF values for each image into the database.

3.2.2 Application Stage

In application stage, it is online mode with graphic user interface support. It will start to process when the matching button in GUI is pressed. When this button is pressed, the image that loaded will go to the feature extraction part as shown in figure 3.2. The process is same as training part. The TF-IDF is calculated and uses the cosine law equation to determine the similarity between the images in database and query image. This portion is accelerated with GPU, Nvidia Geforce GT 9400. So that, the time for image retrieval become fast. After determine the image, it will send the image name to the GUI.

3.3 Graphic User Interface (GUI)

Graphic User Interface (GUI) was designed for the application stage by using C++ programming. This provides the user can easily use this logo recognition system to recognize the unknown logo. Then, give a result to the user. It is user friendly.

In our GUI design, it has two tab, main tab and history tab to change the interface.

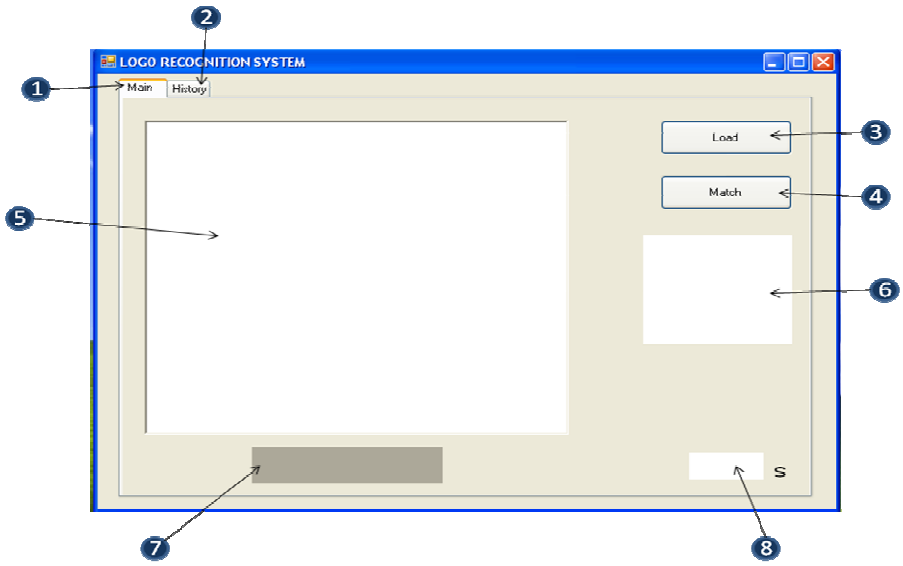

Figure 3.3: Graphic User Interface for logo recognition system

In main tab interface, from figure 3.3, there has eight portions to execute the system and show the information. These eight portions were labelled in number. Portion 1 is the main tab while portion 2 is the history tab. When user starts the program, it shows the main tab interface first. History tab interface is record the history of the results.

From portion 3, it provided “Load” button the let the user load the unknown logo image and shows the image in portion 5.

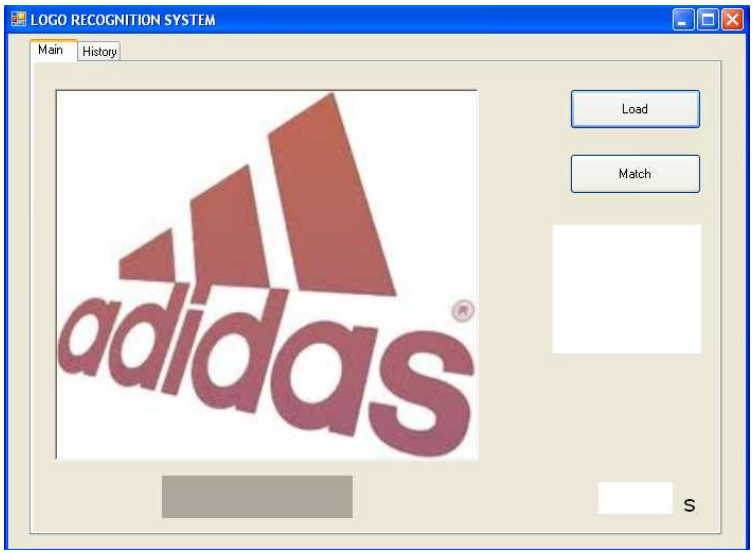

Figure 3.4: The loaded image was shown in Portion 5

After that, in portion 4, user presses the “Match” button to execute the program in application stage.

Figure 3.5: Result in GUI

The result is the shown in portion 6, 7 and 8. Portion 6 indicates the logo image found in database while portion 7 shows the name of the logo classified. Beside these, portion 8 shows the time consumed in the matching. After the logo is classified, it will show the message box to ask the user whether the result is right or not. Then the system will record the related information in history. The user must answer it or the user cannot execute the next step.

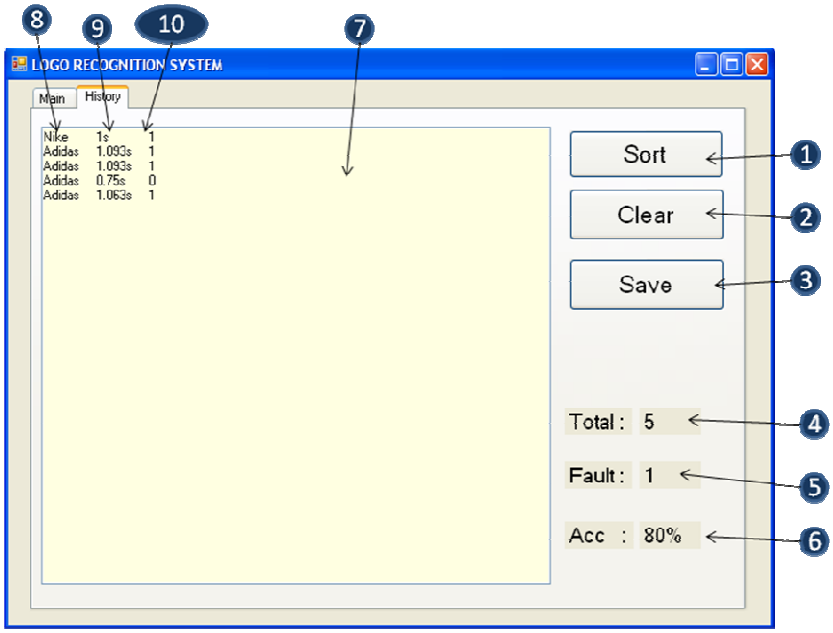

Figure 3.6: History interface

Figure 3.6 shows the History tab interface and it consists of ten portions at here. Portion 7 shows the information that recorded in main tab interface. Portion 8 is the first column and denote as name of the logo found, portion 9 which is the second column indicate the time elapse for matching while portion 10 shows the correctness of the result that answer by the user. 1 is success to classified, vice verse. Portion 4, “Total” is total number of the operation, portion 5, “Fault” is the number of the logo incorrectly matching and “Acc” in portion 6 indicates the percentage of correct image matching over total operation (logo classification).

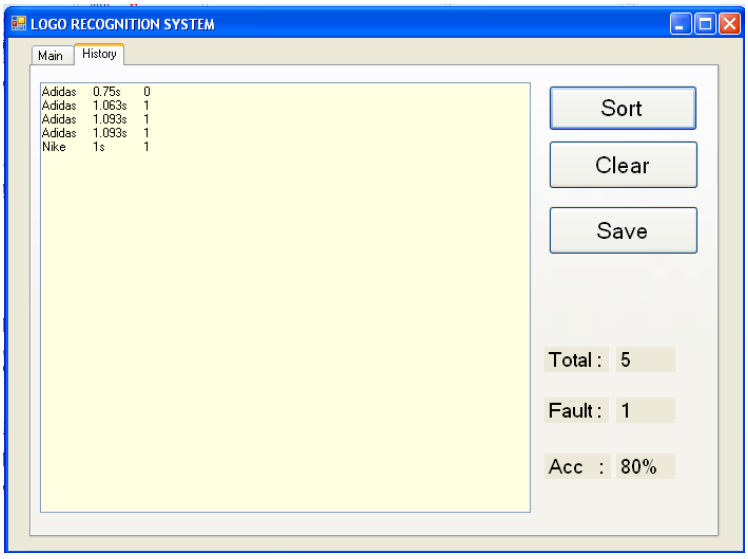

Figure 3.7: Sort operation

In Figure 3.7, when the “Sort” button in portion 1 is pressed, it will sort the arrangement according to their name. After that, the portion 3, “save” button is used for save the history so that the user can know its performance and what the program do in previous.

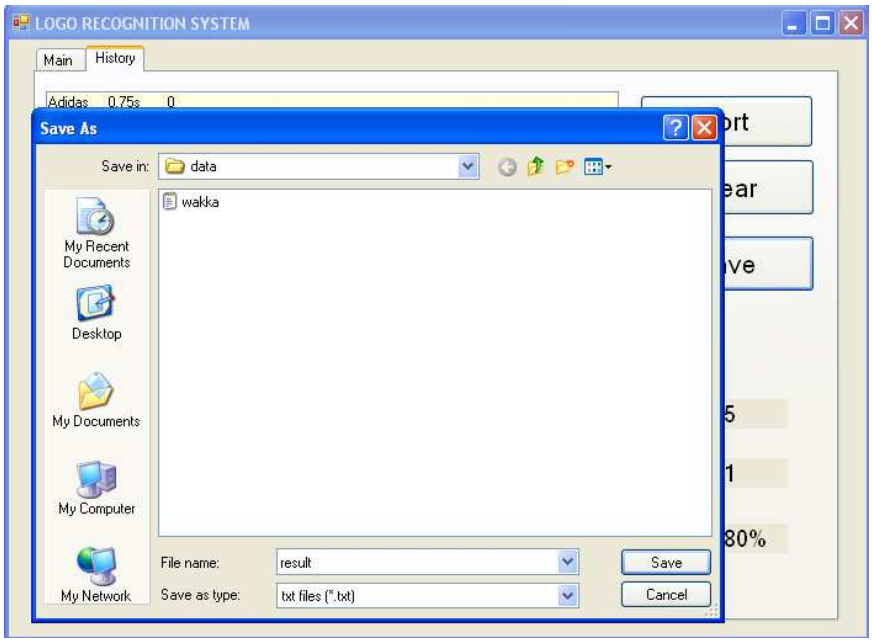

Figure 3.8: Saving process

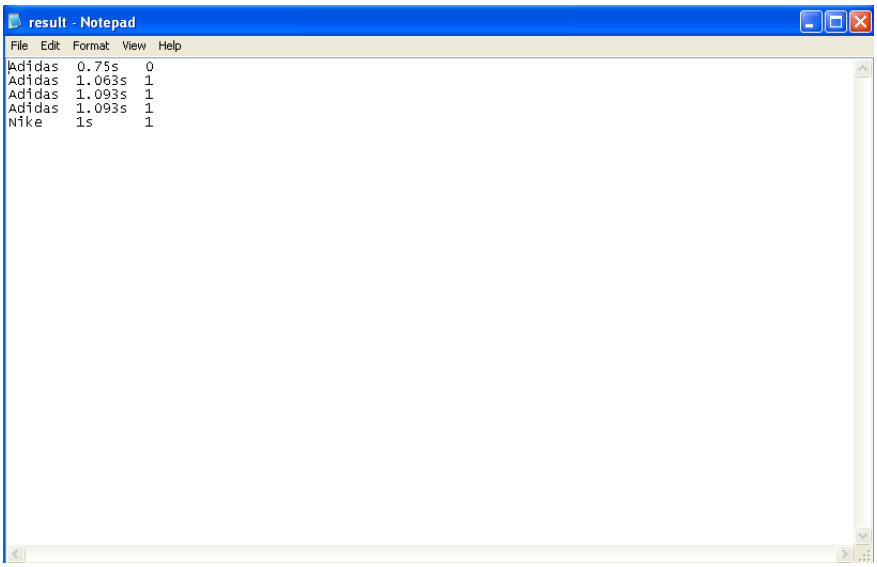

Figure 3.9: Result in test file

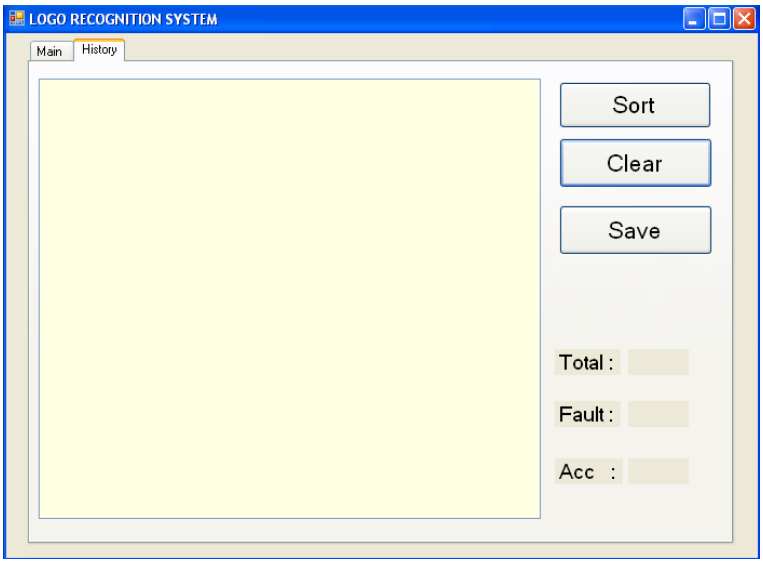

Figure 3.10: Clear operation

The figure 3.10 shows the clear operation when the “Clear” button is pressed. This operation lets the user to start over new their task.

CHAPTER 4

TRAINING STAGE

4.1 Training Stage Overview

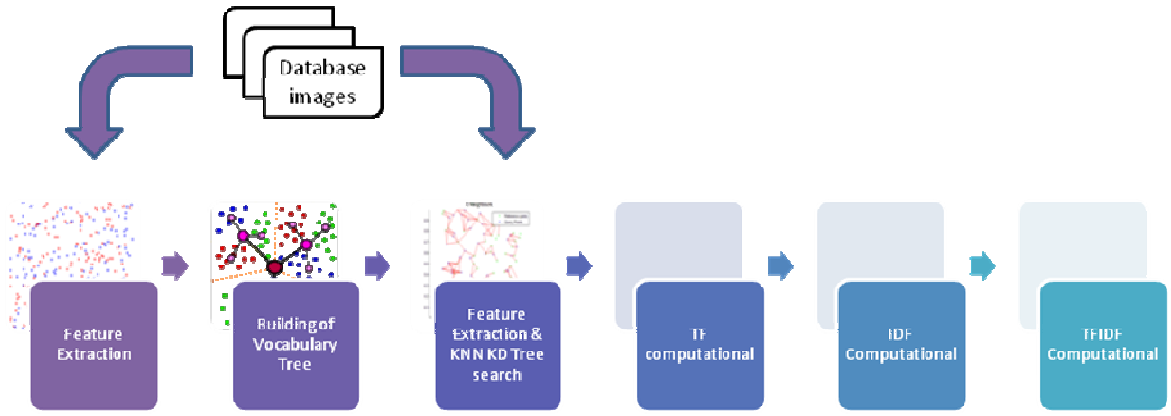

Figure 4.1: Training Stage Overview

In training stage, it has six sections in the whole “Training” program which are feature extraction, building of vocabulary tree, feature extraction & KNN KD tree search, TF computational, IDF computational and TFIDF Multiplication. This programming is running in offline mode. It used for generating the data for the application stage like vocabulary tree and TF-IDF score. The users can tune the

parameters and generate the vocabulary tree and database depends on their requirement.

First, the training stage was designed into six sections and each section uses the data that generated in previous section to process and generate the data that used for next section. For example, by inputting the multiple images which contain three logo images each class, the feature extraction section will extract the feature points of the images and generates the feature space and related data. Therefore, the building of vocabulary tree section used these data to generate the vocabulary tree. Then, the same images set was put in the Feature extraction & KNN KD Tree Search to generate the indices and related data for the next section. So, the process must start from first section to last section to generate the data needed for the application stage.

Note that each section has to be run one by one manually by changes the “PROCEDURE” number from one to six. Each number represented the corresponding section. To make more convenient for the user, the images that used for database has to be put in together and be arranged according to their name. The first section will automatically search and record the specified directory file that the JPEG images stored. Therefore, the user can be easily built the database system easily. The reason to divide into six parts is because it is convenient to let the user to check the data file that generated.

4.2 Feature Extraction

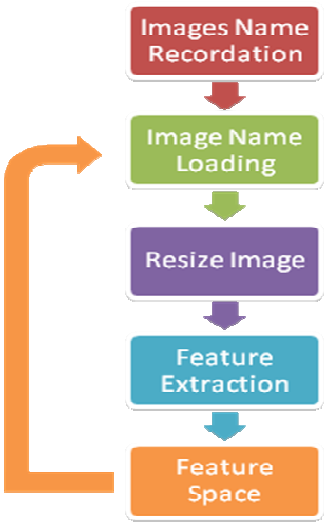

Figure 4.2: Feature Extraction Section

In Feature Extraction section, it consists of five parts program at here which are image name recordation, image name loading, resize image, feature extraction and feature space. Image name recordation was designed to search the JPEG images and record their name for the loading purpose. The others four parts can be grouped into “Building of feature space”. The “Image Name Loading” loads the first image by given the name that stored in the first of the vector container in the “Image Name Recordation” part. Afterward, the image was resized to 520 x 400 pixels size. Moreover, the “Feature Extraction” part will extract the SURF feature of the image and store in the feature space. Noted the SURF descriptor we used is done by OPENSURF. This part will be processed as loop until the last image that stored at the last position of the vector container.

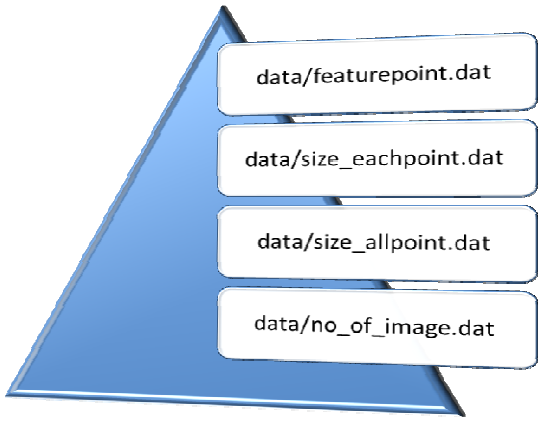

Figure 4.3: Data file generated at Feature Extraction section

From Figure 4.3, there are four dat file was generated after this section. They were stored in data directory file. The “featurepoint.dat” is the feature space that includes all SURF feature points of all input images. “size_eachpoint.dat” indicates the number of SURF feature points found for each image in order to their position in the vector container. “size_allpoint.dat” and “no_of_image.dat” show the information about the number of all feature points that stored in feature space and number of input images.

4.2.1 Image Name Recordation

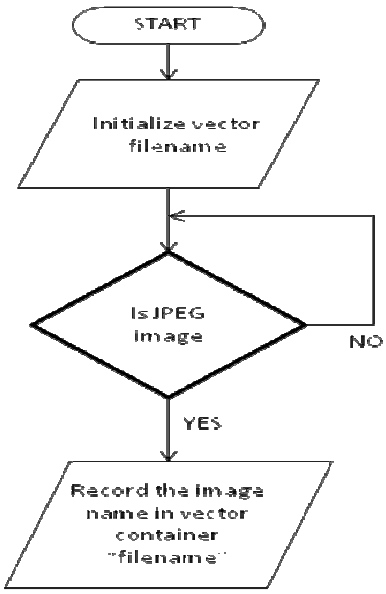

Figure 4.4: Flow chart of Image Name Recordation

The function "`put_nameofimages_indir(filename)`" is responsible to "Image Name Recordation" part. It searched the JPEG images in the specified directory file and stored into the vector "filename".

Pseudo code for the function `put_nameofimages_indir(filename)`:

Vector:*filename*
Find the specified directory file
Search the JPEG image
If(.JPG)
pushback to filename
else
continue seach until end

4.2.2 Building of Feature Space

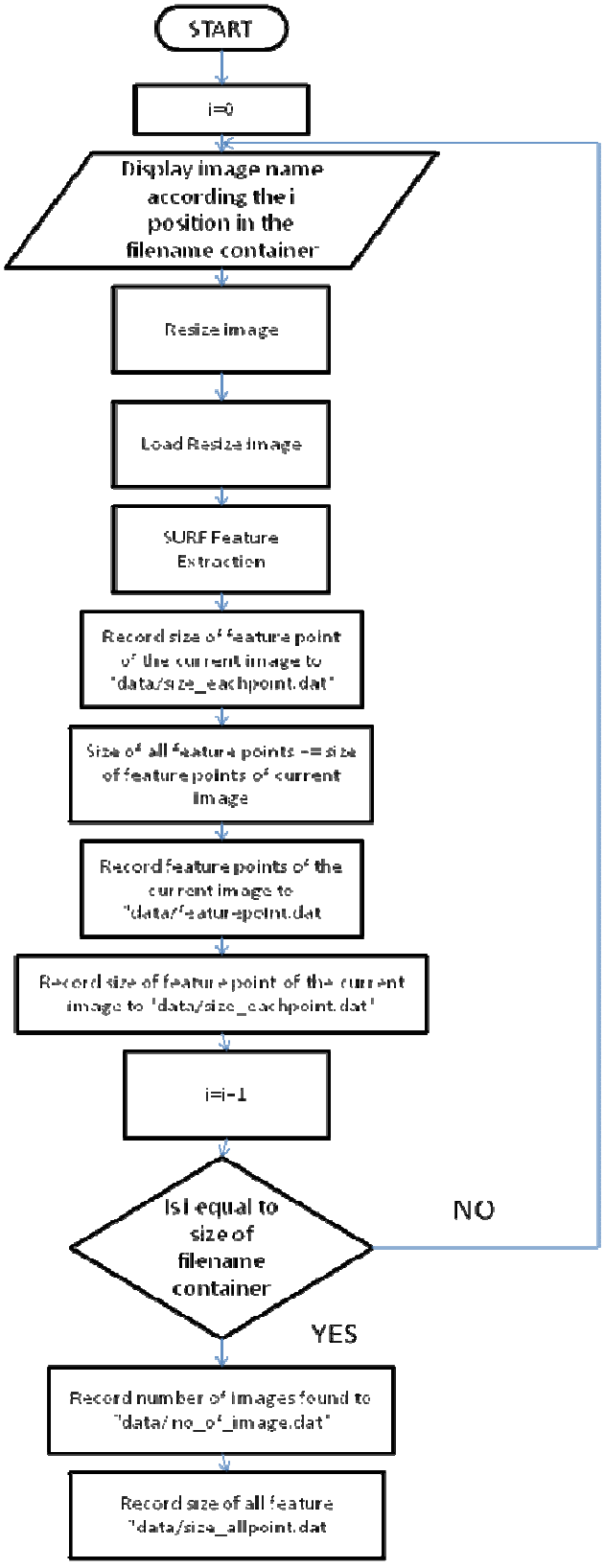

Figure 4.5: Flow chart for Building of Feature Space

The Figure 4.5 has shown the process flow of Building of Feature Space in detail. This part loads the image according to the name that stored in “filename” and standardizes the size of the input image for feature extraction. After that, it will store the SURF feature points to the feature space (featurepoint.dat).

Pseudo code for the Building of Feature Space

```
Initialize i, no_all_point =0;
For i from 0 to the filename size
        Display image name for i position in the filename;
        Call resize_image (parameters: pointer to input image, pointer to resized image)
                The image was resized to 520x400;
                Return (pointer to resized image);
        Load resize image;
        Call surfDetDes(parameters: pointer to resize image,pointer to ipts)
                Extract feature point of resize image;
        no_all_point=no_all_point+size of feature point of each image
        Write the size of feature point of each image to "data/size_allpoint.dat"
        For k from 0 to size of feature point of each image
                For j from 0 to 64
                        Write the feature point from ipts to "data/featurepoint.dat"
                EndFor
        EndFor
EndFor
```

4.3 Building of Vocabulary Tree

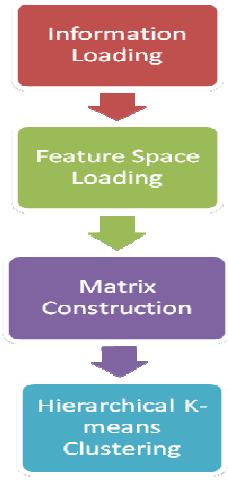

Figure 4.6: Process in Building of Vocabulary Tree

As we can see in Figure 4.6, Building of Vocabulary Tree consists of four processes which are Information Loading, Feature Space Loading, Matrix Construction and Hierarchical K-means Clustering. This section read the data that generated by previous section to build up the vocabulary tree.

First, it reads the feature space and uses the Hierarchical k-means clustering to find the centroids of each cluster. We used Hierarchical k-means clustering as our clustering method because it is more stable and advance compare to other clustering method. The centroids found are treated as a set of visual words as mentioned at section 3.2.1. For hierarchical k-means cluster, unlike the k-means cluster which find the centroids of each cluster that we set, the hierarchical k-means cluster will explore the centroids of cluster if the two centroids of clusters are too close by given cb index. Beside this, the way it explores and finds the centroids is in hierarchical way. First, the centroids are random generated. Then it will explore its centroids by given branch factor. Supposed the branch factor is 5, then the centroids will explore to maximum 5 centroids of clusters. However, it is not necessary the centroid has to explore to 5 centroids, it can explore to 4, 3, even choose not to explore the centroid. Thus, it is more accurate and reliable than the k-means cluster.

Table 4.1: File for read and write

Data file used for read	Data file generated

`data/featurepoint.dat`
`data/size_allpoint.dat`
`data/no_of_image.dat`
	`data/vocabularytree.dat`
	`data/numberofcluster.dat`

The data file that used for read was discussed at the section 4.2. The vocabulary tree was stored in “vocabularytree.dat” and the number of cluster or centriod (visual word) was stored in “numberofcluster.dat”.

4.3.1 Information Loading

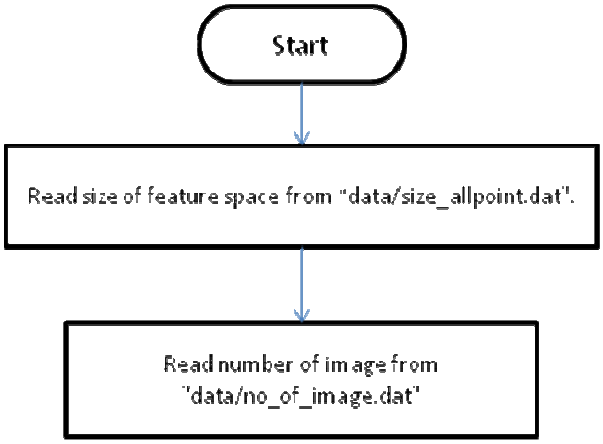

Figure 4.7: Flow chart of Information Loading

The information loading loads the number of image and number of all feature points only.

Pseudo code for `load_information_no_point(number_image,number_allpoint);`

```
load_information_no_point  (parameters:pointer to number of image,pointer to number of all feature points)
Read the number of image from data/size_allpoint.dat
Read the number of all feature points from data/no_of_image.dat
```

4.3.2 Feature Space Loading

The Feature Space Loading loads the feature points in the feature space to IpVec ipts.

Pseudo code for `load_data(number_allpoint,"data/featurepoint.dat",ipts);`

```
Load_data (parameters: number of all feature points,  "data/featurepoint.dat",  ipts)
For i from 0 to number of all feature points
        For j from 0 to 64
                Read the value from "data/featurepoint.dat" to  ipts[i][j]
        EndFor
EndFor
```

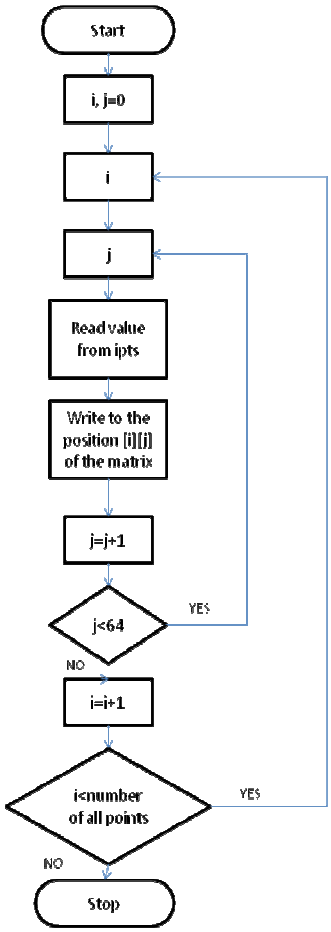

Figure 4.8: Flow chart of Feature Space Loading

4.3.3 Matrix Construction

The matrix construction is used to construct opencv format type matrix to process the function. This function was designed to change the IpVec format to cvMat format.

Pseudo code for `construct_cvmatrix(feature,ipts);`

```
construct_cvmatrix (parameters: pointer to output matrix, ipts)
For i from rows of output matrix
        For j from columns of output matrix
                Read the data in ipts[i][j]
                Write to the output matrix[i][j]
        EndFor
EndFor
```

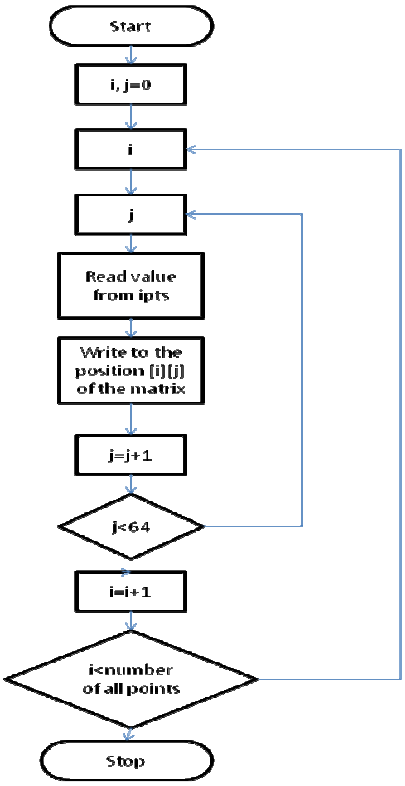

Figure 4.9: Flow chart of Matrix Construction for Feature Space

4.3.4 Hierarchical K-means Clustering

The hierarchical k-means clustering used to cluster the feature space to find out the centroids of the cluster. These centroids denoted to its corresponding visual words. The visual word data were saved in the vocabulary tree.

Pseudo code for hierarchical k-means clustering

```
Parameters: number of leaf and branch factor
Construct kmeans index by input parameters
Construct hierarchical k-means index (parameters: feature space matrix, kmeans index)
Perform hierarchical k-means clustering
(Parameters: k-means index, feature space matrix, pointer to vocabulary tree matrix)
Return (vocabulary tree matrix)
```

4.4 Feature Extraction & KNN KD Tree Search

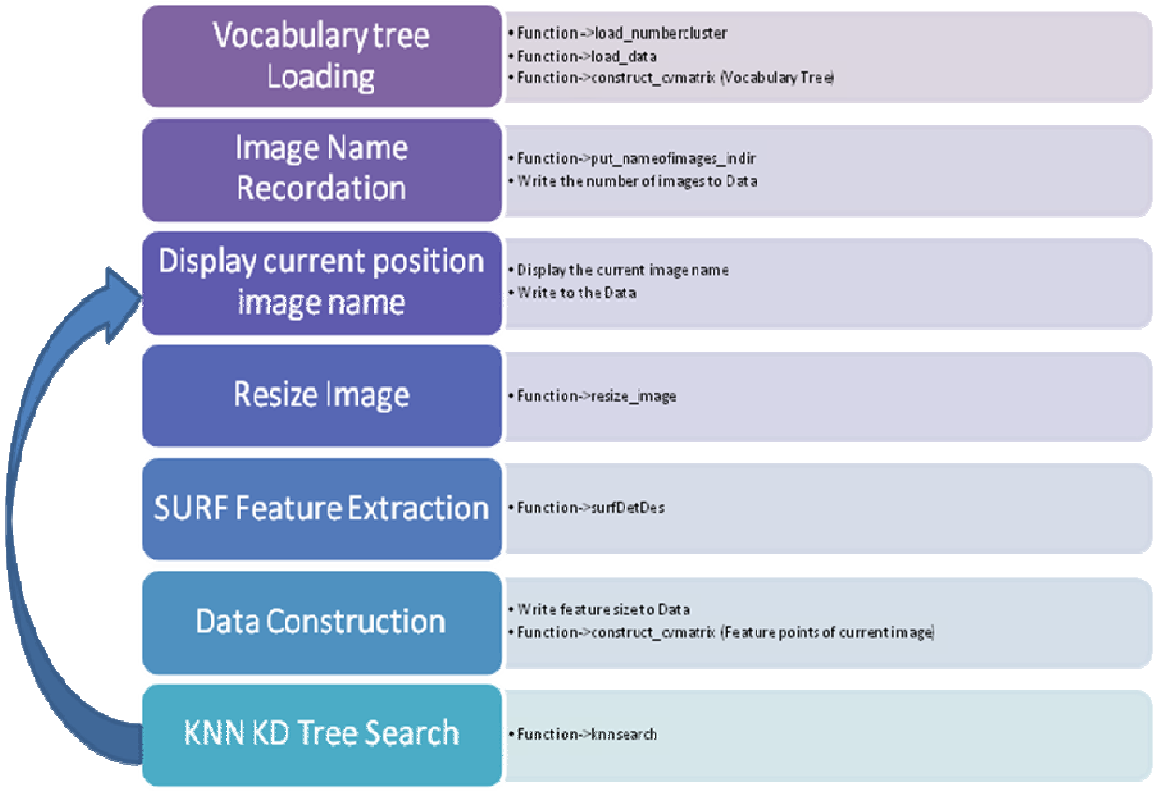

Figure 4.10: Process Flow of Feature Extraction & KNN KD Tree Search

In this section, it determined the visual words that the image had and treated them as the content of an image. By using KNN KD Tree Search algorithms, the feature points of the input image will find the nearest neighbour visual words in vocabulary tree. The K of the KNN is defined that the number the nearest neighbour wanted and the vocabulary tree is in KD tree form. It is because the KD tree form will accelerate the speed of searching. Therefore, the nearest neighbour visual words are found and become to be the content of the image. For example, if the feature point of the image find three nearest neighbour visual words around it according the distance between them when KNN is 3, these three visual words are replaced the feature point and include in the content of the image. So, the content of the image is not using the feature point to represent them but using the visual words. Therefore, we can know the content of the image consists of two visual word 1, one visual word 2, zero visual word 3and etc. This concept is very important in our image retrieval system design. It standardizes the content of the image by using the visual words in vocabulary tree so that we can easily know the similarity of two images.

After KNN algorithms, it will give an indices matrix that contains which centroids are nearest to the query feature points. It depends on the number of knn. If the KNN is 3, it will show the first three nearest centroids (visual words). This matrix records the visual words that near to the query points according the order of the query points.

Table 4.2: File for read and write

Data file used for read	Data file generated

`data/vocabularytree.dat`
`data/numberofcluster.dat`
	`data/number_image.dat`
	`data/no_point_each_image.dat`
	`data/name.dat`
	`data/knnindices.dat`

The number of cluster, or can be said number of visual word is read from `data/vocabularytree.dat`. After that, load the data in `data/vocabularytree.dat` . `data/number_image.dat`,`data/no_point_each_image.dat` and `data/name.dat` save the number of image for database, size of the feature points for each image and their name.

Pseudo code for Feature extraction & KNN KD tree searching

```
Initialize number of knn=3, number of cluster, vector : filename , IpVec: ipts and ipts1
Call load_numbercluster (parameters: pointer to number of cluster)
        Return ( pointer to number of cluster)
Call Load_data(parameters:number of cluster,"data/vocabularytree.dat",pointer to ipts)
        Return (pointer to ipts)
Call construct _cvmatrix (parameters: pointer to vocabulary tree, ipts)
        Return (pointer to vocabulary tree)
Change vocabulary tree format from cvmat to cv::Mat
Call  put_nameofimages_indir (parameters: pointer to filename)
        Return (pointer to filename)
Save number of image into data/number_image.dat
For  i from 0 to number of image
        Display i image
        Save the image name in data/name.dat
        Call resize_image(parameters: input image, pointer to resized image)
                Resize to 520x400
                Return (pointer to resized image)
        Call surfDetDes (parameters: resized image, pointer to ipts1)
                Return (pointer to ipts1)
        Save the size of feature point to data/no_point_each_image.dat
        Call construct _cvmatrix (parameters: pointer to query matrix, ipts1)
        Change query matrix format from cvmat to cv::Mat
        Call knnsearch (parameters: query matrix, vocabulary tree matrix, number of knn)
                Return ( indices matrix)
EndFor
```

4.4.1 Load Vocabulary Tree

It consists of three functions which are load_numbercluster, load_data, construct_cvmatrix for vocabulary tree. This part loads the data needed to undergo this program.

Pseudo code for function `load_numbercluster(no_cluster);`

load_numbercluster (parameters: *pointer to number of cluster*)
    Read Number of cluster from `data/numberofcluster.dat`
    Return (*pointer to number of cluster*)

Pseudo code for function
`load_data(no_cluster,"data/vocabularytree.dat",ipts)`

Load_data (parameters: *number of cluster*, *"data/vocabularytree.dat",* *ipts*)
    For i from 0 to *number of cluster*
        For j from 0 to 64
            Read the value from `"data/vocabularytree.dat" to` *ipts[i][j]*
        EndFor
    EndFor

Pseudo code for function `construct_cvmatrix(vocabtree,ipts);`

* Please refer to 4.3.3

4.4.2 Image Name Recordation

Please refer to 4.2.1

4.4.3 Display Current Position Image Name

Please refer to 4.2.2

4.4.4 Resize Image

Please refer to 4.2.2

4.4.5 SURF Feature Description

Please refer to 4.2.2

4.4.6 Data Construction

Pseudo code for function `construct_cvmatrix(datafeature,ipts1);`

```
construct_cvmatrix (parameters: pointer to datafeature matrix, ipts1)
For i from 0 to rows of output matrix
        For j from 0 to columns of output matrix
                Read the data in ipts1[i][j]
                Write to the datafeature matrix[i][j]
        EndFor
EndFor
```

4.4.7 KNN KD Tree Search

Pseudo code for function `knnsearch(query,vocabulary_tree,no_knn);`

```
knnsearch (parameters: query matrix, vocabulary matrix, number of knn)
Setup Kd tree index parameter
Setup index parameter, search parameter
Construct resulting_indices matrix
Construct resulting_distance matrix
Perform knn search
Call save_indice(parameters: "data/knnindices.dat",resulting_indices)
        For i from 0 to rows of query matrix
                For j from 0 to number of knn
                        Read the value in resulting_indices[i][j]
                        Write to "data/knnindices.dat"
                EndFor
        End For
```

4.5 TF Computational

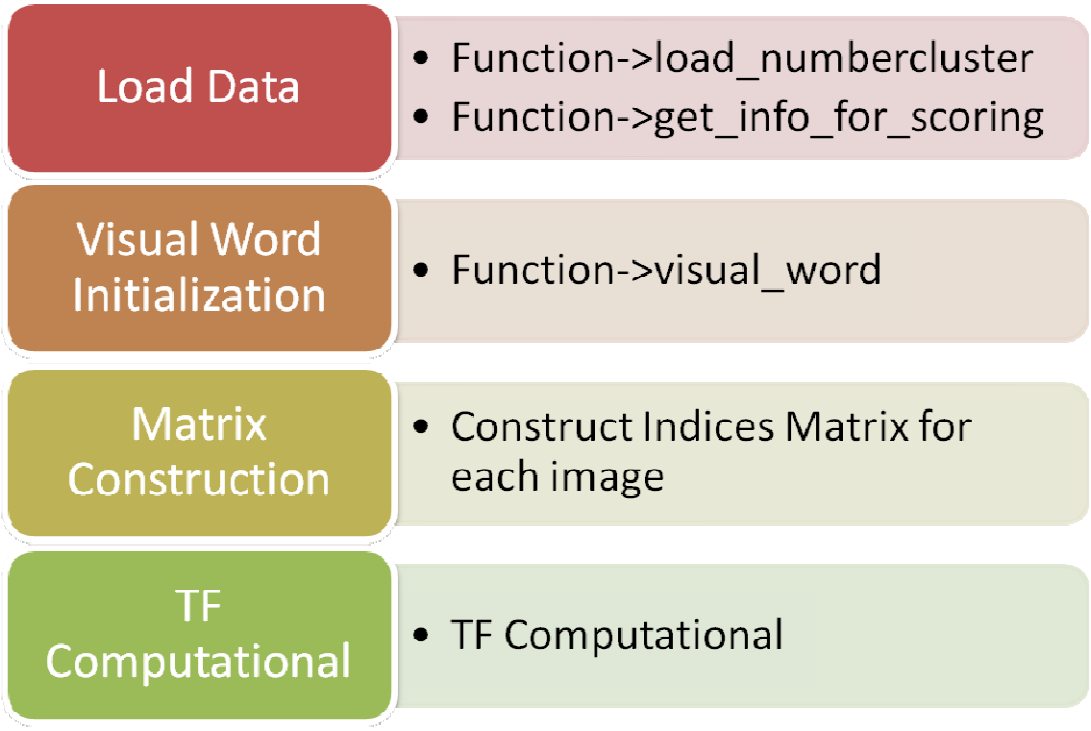

Figure 4.11: Process Flow of Feature Extraction & KNN KD Tree Search

From figure 4.11, we can know the process flow for this section. The TF-IDF concept was mentioned before at section 3.2.1. In TF computation, the formula we used is shown below.

$$TF = \frac{frequency\ of\ the\ this\ visual\ word\ found\ in\ this\ image}{number\ of\ feature\ point\ found\ in\ this\ image} \quad (4.1)$$

By using this formula, we can calculate the TF score by given knnindices file. Thus, TF will count the frequency of the specified visual word found in this image and divide by number of feature points found in this image.

Table 4.3: File for read and write

Data file used for read	Data file generated
data/knnindices.dat data/numberofcluster.dat data/number_image.dat data/no_point_each_image.dat	data/tfscore.dat

`data/tfscore.dat` store the TF score that found for each image.

Pseudo code for TF computational

initialize *number of knn, number of visual words, number of images*

vector: *indice, visual word, size of feature point of each image*

Call load_numbercluster(*pointer to number of visual word*)

Return (*pointer to number of visual word*)

Call get_info_for_scoring(parameters*: pointer to number of image& size of feature point of each image*)

Return (*pointer to number of image& size of feature point of each image*)

Call visual_word(parameters: *pointer to visualword & numberof visual word*)

Return (parameters: *pointer to visualword & numberof visual word*)

Perform TF Computation

Save into `data/tfscore.dat`

4.5.1 Load Data

Pseudo code for function `load_numbercluster(no_visualword);`

Please refer to section 4.4.1

Pseudo code for function `get_info_for_scoring(no_image,sizeof_each_img);`

`get_info_for_scoring` (parameters: *pointer to number of image & sizeof_each _img(vector)*)
Read the number of image from `data/number_image.dat`
For i from 0 to number of image
    Read the value from `data/no_point_each_image.dat`
    Pushback value to the *sizeof_each_img*
EndFor

4.5.2 Visual Word Initialization

Pseudo code for function `visual_word(visualword,no_visualword);`

`visual_word` (parameters: *pointer to visualword(vector) & number of visualword*)
For i from 0 to *number of visual word*
    vword=0;
    Pushback *vword* to the *visual word*
EndFor

4.5.3 Matrix Construction

Pseudo code for construction of indices matrix

```
Vector: sizeof_each _img  , indice
For k from 0 to size of sizeof _each_img
        Initialize d
        For i from 0 to size of sizeof _each_img[k]
                For j from 0 to number of knn
                        Read the d from data/knnindices.dat
                        Put d into indice_data[i][j]
                EndFor
        EndFor
        Push the indice_data matrix into indice[k]
EndFor
```

4.5.4 TF Computational

Pseudo code for TF Computational

```
Vector: indice , sizeof_each _img
Initialize tf
For k from 0 to size of indice
        For i from 0 to size of indice[k]
                For j from 0 to number of knn
                        For m from 0 to number of visual word
                                tf= value[i][j] in matrix for  indice[k]
                                If tf equal to m
                                Then visualword[m] increased by 1
                        EndFor
                EndFor
        EndFor
        For m from 0 to number of visual word
                Tfdata=visualword[m]/ sizeof_each _img[k]
        EndFor
EndFor
```

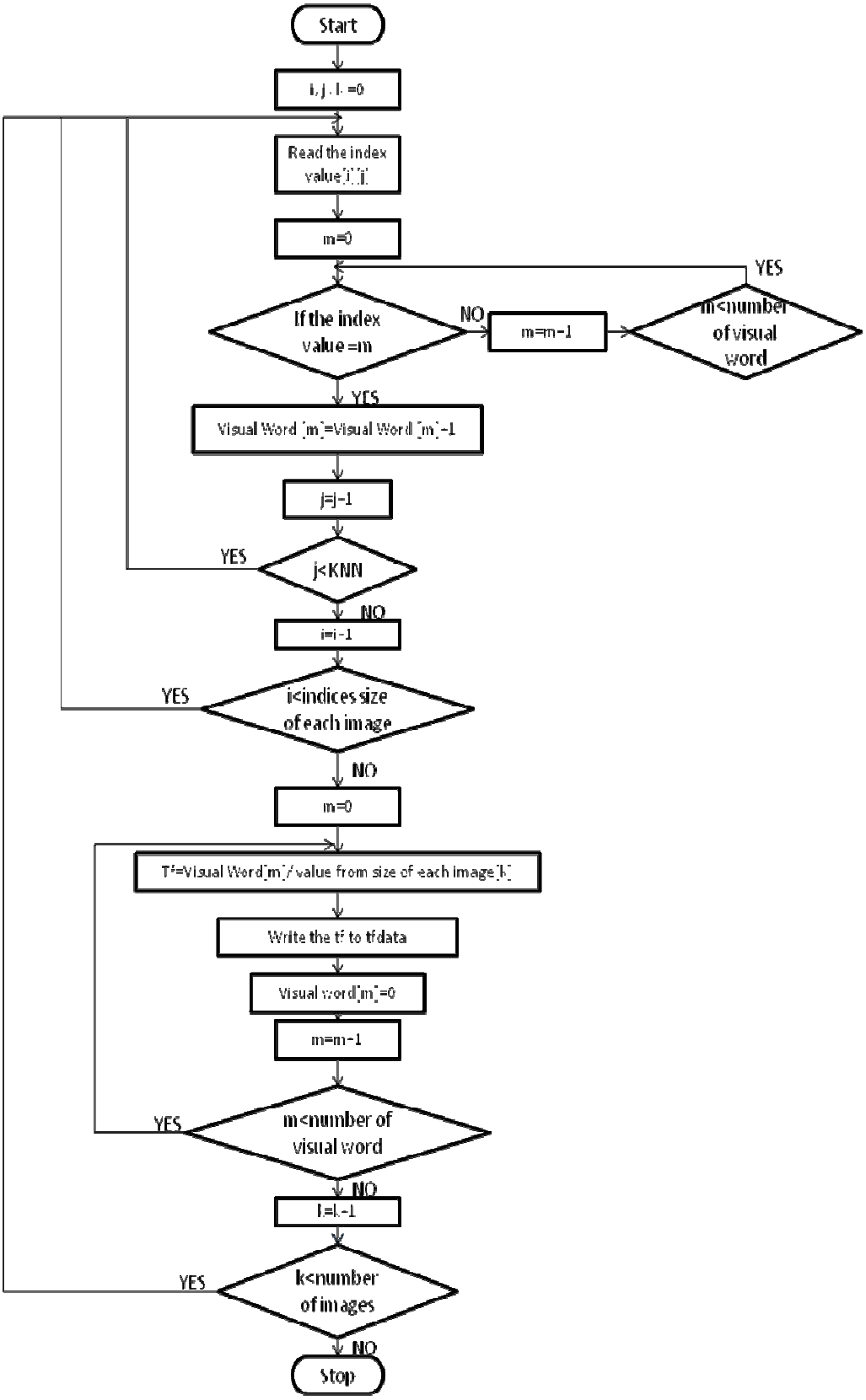

Figure 4.12: Flow Chart for TF computation

4.6 IDF Computational

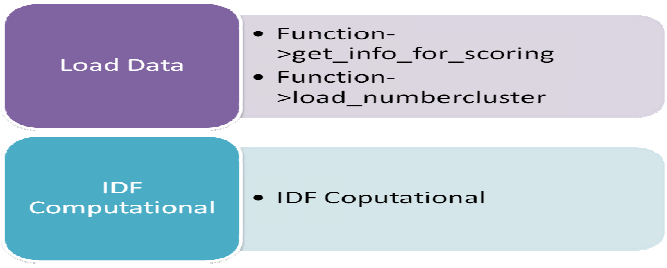

Figure 4.13: Process flow for IDF Computational

From figure 4.13, it includes two parts only which are Load Data and IDF Computational. Beside these, the TF data was used to compute IDF value. As we know, the idf values acts as constant for all after computed. We used the equation that show below to compute IDF value for each visual word. For example, “diamond” is occurred only in few document only, but the “the” is common occur in most document. IDF computation make “diamond” word has larger idf value and “the” is lower value. Thus we can know that if the “diamond” word occur in others document, it can be said that it has large probability that these two documents is very similar. Therefore, same concept applied in our system for image recognition.

$$IDF = log \frac{number\ of\ images\ in\ database}{number\ of\ images\ include\ this\ visual\ word} \tag{4.2}$$

Table 4.4: File for read and write

Data file used for read	Data file generated

`data/tfscore.dat`
`data/numberofcluster.dat`
`data/number_image.dat`
`data/no_point_each_image.dat`
	`data/idfscore.dat`

After this process, idf value for each visual word is recorded in `data/idfscore.dat`.

4.6.1 Load Data

Please refer to 4.5.1

4.6.2 IDF Computation

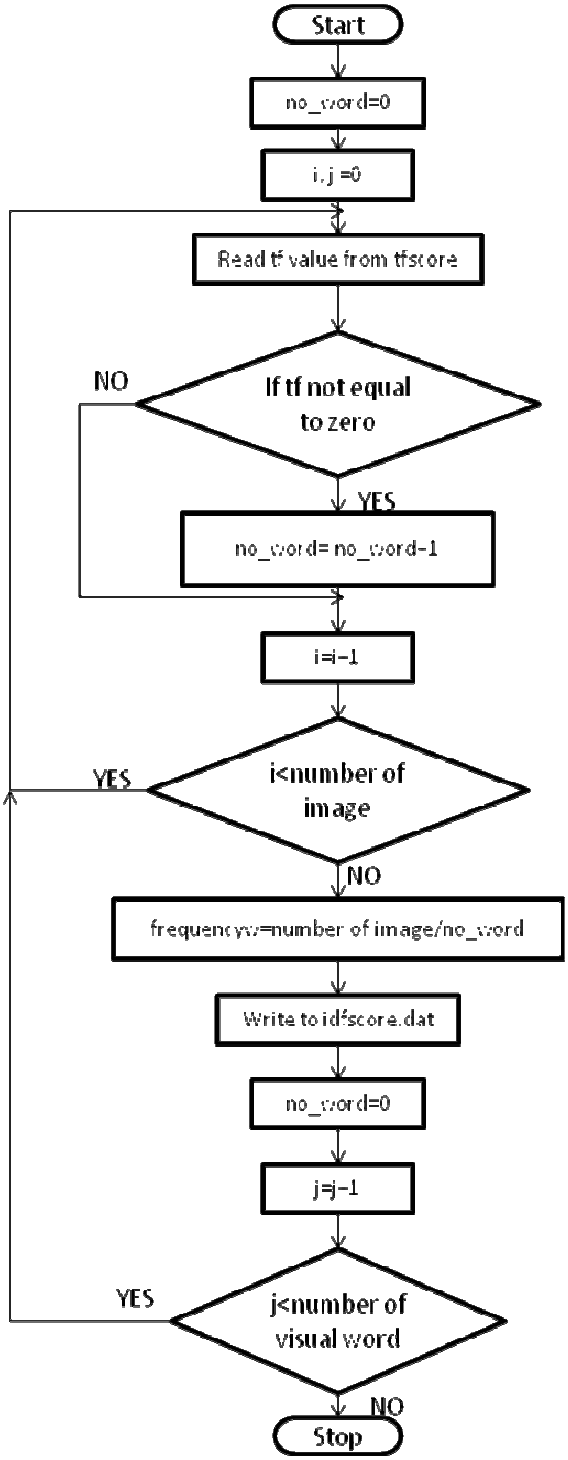

Figure 4.14: Flow Chart for IDF computation

Pseudo code for IDF Computational

```
Vector: indice , sizeof_each _img
Initialize idf,no_word=0
For j from 0 to number of visual word
        For i from 0 to number of image
                Read d from data/tfscore.dat
                If d not equal to 0
                Then no_word was increased by 1
        EndFor
        idf=number of image/no_word
        write idf to data/idfscore.dat
        no_word=0
EndFor
```

4.7 TF-IDF Computational

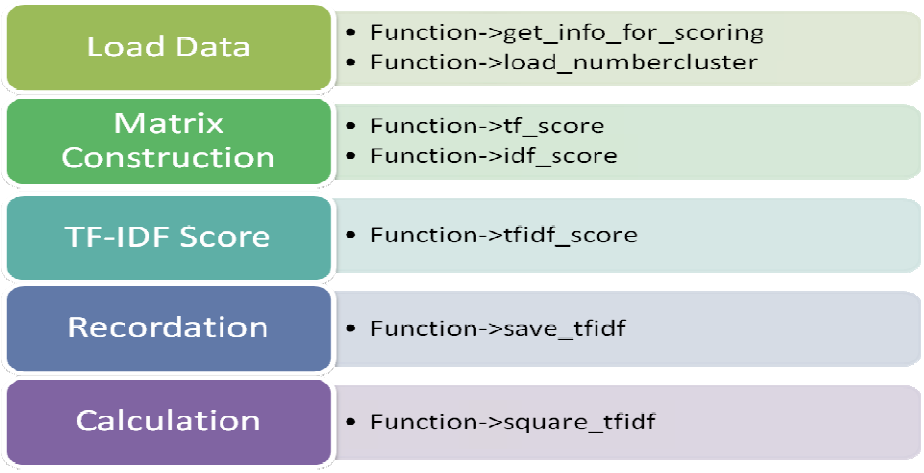

Figure 4.15: Process flow of TF-IDF computational

From figure 4.15, TF-IDF computational has five process, load data, matrix construction, TF-IDF score, recordation and calculation. Therefore, tf-idf values were stored in `data/tfidf_database.dat` for each image as our database type. Beside that the data used for cosine law equation was done in calculation. This

calculation calculated the square root of square of sum of the tf-idf values for each image and stored in `data/rootsquare_tfidf.dat.`

$$\sqrt[2]{(TF-IDF)_i{}^2+(TF-IDF)_{i+1}{}^2+(TF-IDF)_{i+2}{}^2+\cdots+(TF-IDF)_n{}^2} \quad (4.3)$$

Where i is 0 and n is number of the images in database.

Table 4.5: File for read and write

Data file used for read	Data file generated
`data/tfscore.dat`	`data/idfscore1.dat`
`data/numberofcluster.dat`	`data/tfidf_database.dat`
`data/number_image.dat`	`data/rootsquare_tfidf.dat`
`data/no_point_each_image.dat`	
`data/idfscore.dat`	

Pseudo code for TF-IDF Computational

Vector: *sizeof_each _img*

Cvmat Matrix:*tfscore,idfscore,tfidfscore*

Initialize *number of visual word,number of image*

Call load_numbercluster(parameters: *pointer to number of visual word*)

Return (*pointer to number of visual word*)

Call get_info_for_scoring(parameters*: pointer to number of image& size of feature point of each image*)

Return (*pointer to number of image& size of feature point of each image*)

Call tf_score(parameters: *pointer to tfscore, number of image, number of visual word*)

Return (*pointer to tfscore*)

Call idf_score(parameters: *pointer to idfscore, number of visual word*)

Return (*pointer to idfscore*)

Change tfscore from cvmat to cv::Mat format

Change idfscore from cvmat to cv::Mat format

Call tfidf_score(parameters: pointer *to tfidfscore,number of image, number of visual word,tfscore,idfscore*)

Return (*tfidfscore*)

Change tfidfscore from cvmat to cv::Mat format

Call save_tfidf (parameters: *tfidfscore, number of image, number of visual word*)

Write to `data/tfidf_database.dat`

Call square_tfidf(parameters*: tfidfscor, number of image, number of visual word*)

Write to `data/rootsquare_tfidf.dat`

4.7.1 Load Data

Please refer to 4.5.1

4.7.2 Matrix Construction

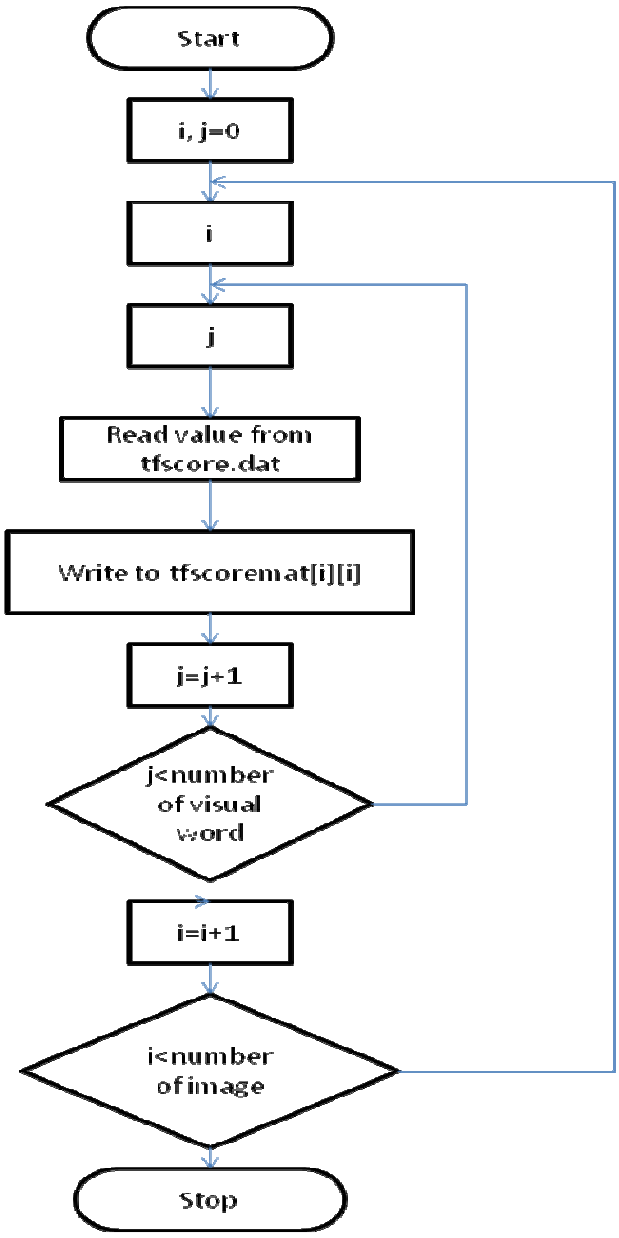

Figure 4.16: Flow Chart for TF matrix loading

Pseudo code for function

`tf_score(tfscoremat,number_of_image,no_visualword);`

`tf_score` (parameters: *pointer to tfscore, number of image, number of visual word*)

For i from 0 to *number of image*

    For j from 0 to *number of visual word*

        Read the tf data from `data/tfscore.dat`

        Write the tf data to the *tfscore matrix[i][j]*

    EndFor

EndFor

Return (pointer to *tfscore*)

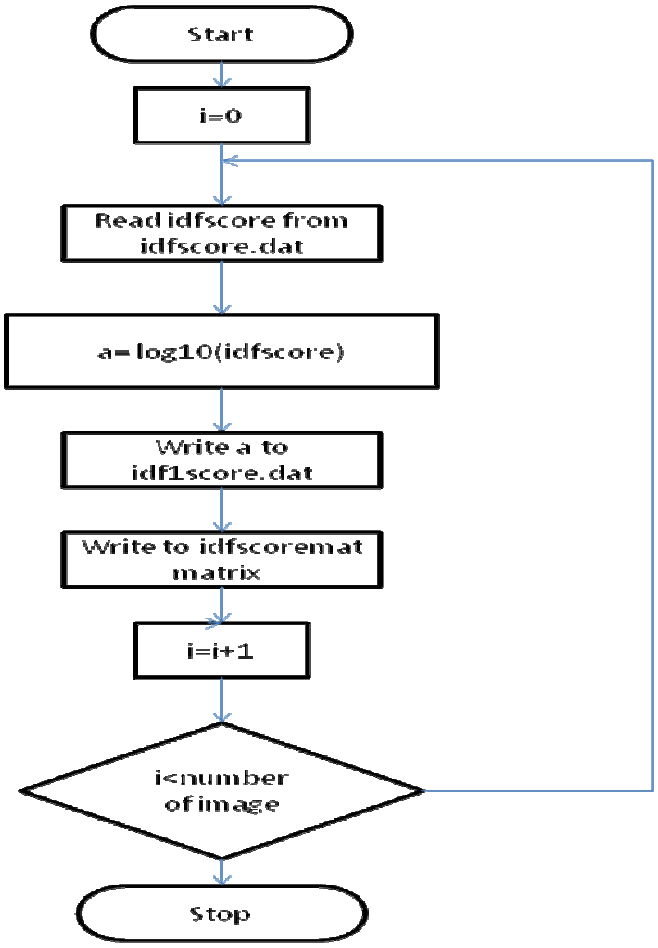

Figure 4.17: Flow Chart for IDF computation and loading

Pseudo code for function `idf_score(idfscoremat,no_visualword);`

`idf_score` (parameters: pointer to *idfscore*, *number of visual word*)

For i from 0 to *number of visual word*

Read the idf data from `data/idfscore.dat`

idf=log10 (idf)

Write the idf data to the `data/idfscore1.dat`

Write the idf data to the *idfscore matrix[i][j]*

EndFor

Return (pointer to *idfscore*)

4.7.3 TF-IDF Score

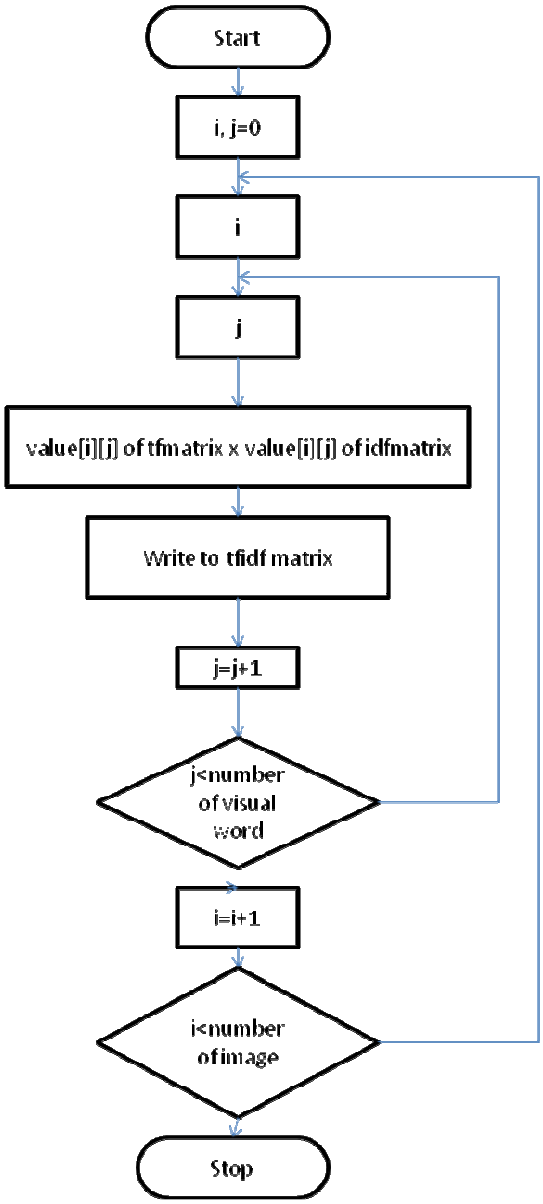

Figure 4.18: Flow Chart for TF-IDF computation

Pseudo code for function
`tfidf_score(tfidfscoremat,number_of_image,no_visualword,tfmat,idfmat);`

`tfidf_score`(parameters: *pointer to tfidfscore,number of image, number of visual word,tfscore,idfscore*)

For i from 0 to *number of image*

    For j from 0 to *number of visual word*

        Read the tf data from *tfscore matrix[i][j]*

        Read the idf data from *idfscore matrix[0][j]*

        Tfidf=tf x idf

        Write the tfidf data to the *tfidfscore matrix[i][j]*

    EndFor

EndFor

Return (pointer to *tfidfscore*)

4.7.4 Recordation

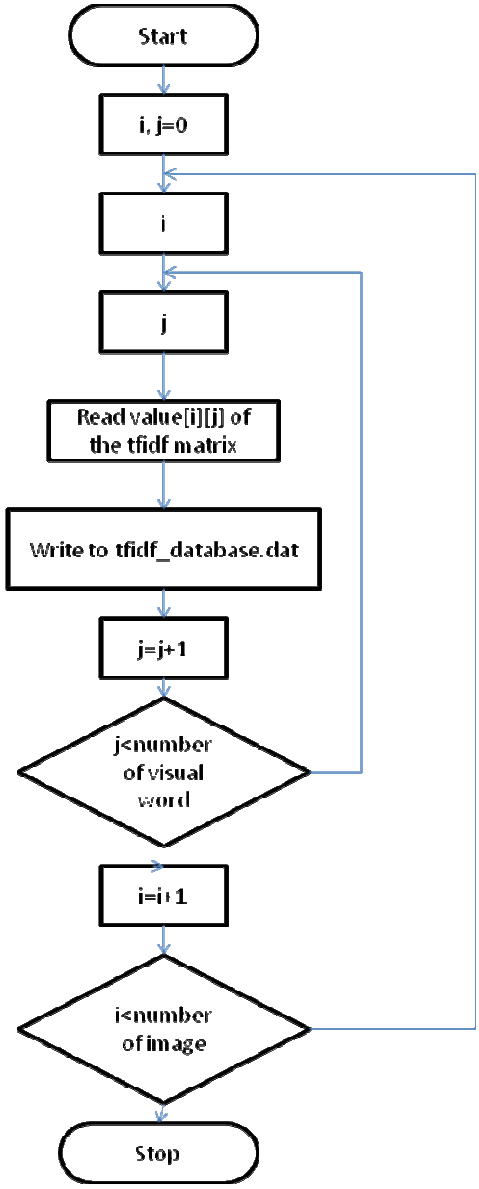

Figure 4.19: Flow Chart for TF-IDF data writing

Pseudo code for function
`save_tfidf(tfidfmat,number_of_image,no_visualword);`

`save_tfidf` (parameters: tfidfscore, number of image, number of visual word)

For i from 0 to number of image

    For j from 0 to number of visual word

        Read the tfidf data from tfidfscore matrix[i][j]

        Write the tfidf data to the `data/tfidf_database.dat`

    EndFor

EndFor

4.7.5 Calculation

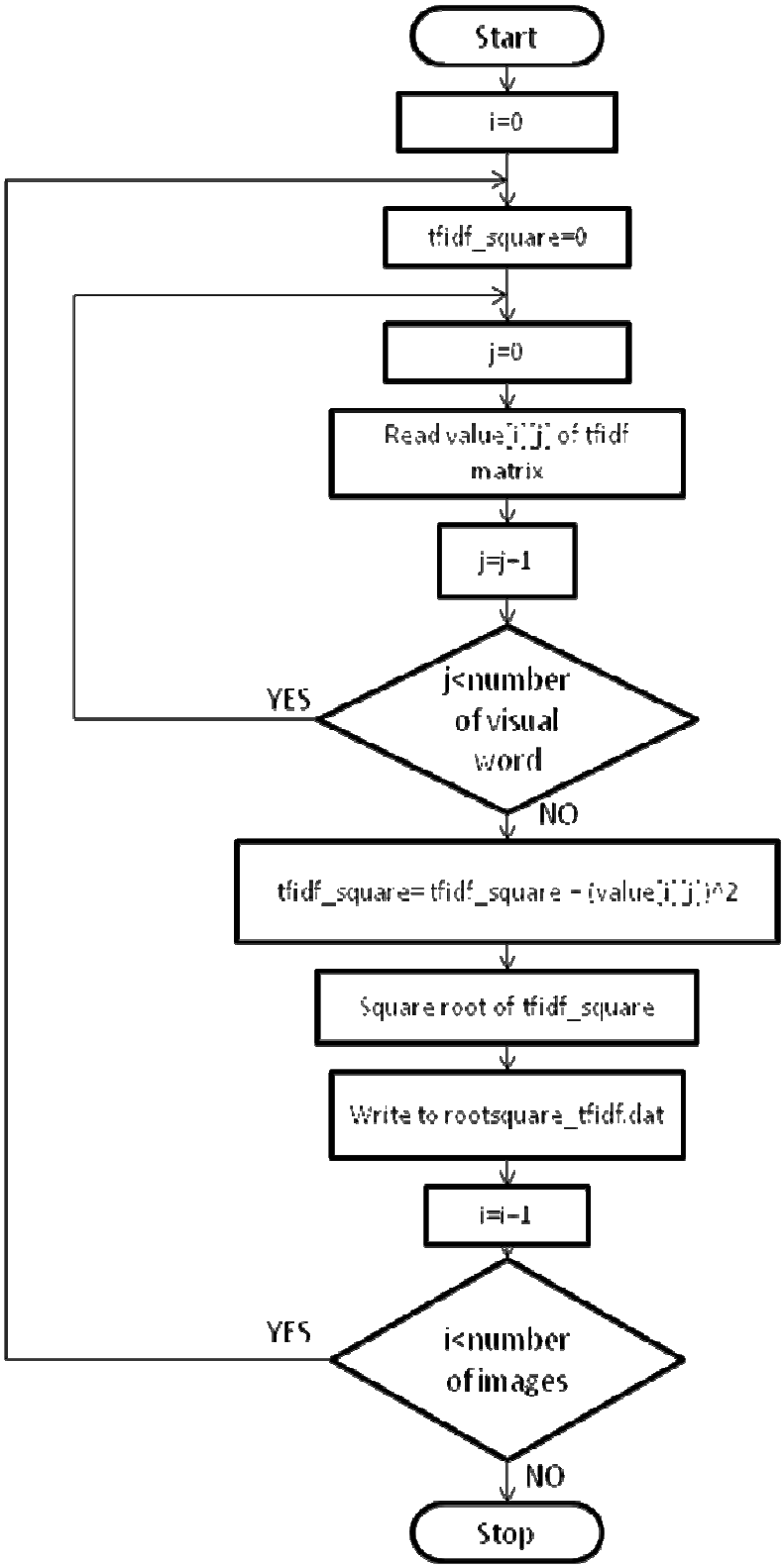

Figure 4.20: Flow Chart for calculation

Pseudo code for function
`square_tfidf(tfidfmat,number_of_image,no_visualword);`

`square_tfidf` (parameters: tfidfscore, number of image, number of visual word)

For i from 0 to number of image

    Sum of Square of tfidf=0

    For j from 0 to number of visual word

        Read the tfidf data from tfidfscore matrix[i][j]

        Sum of Square of tfidf = Sum of Square of tfidf + (tfidf)^2

    EndFor

    Square root of sum of square of tfidf

    Write this value to the `data/rootsquare_tfidf.dat`

EndFor

CHAPTER 5

Application Stage

5.1 Application stage overview

Figure 5.1: Application Stage Overview

The overview of the application stage was shown in figure 5.1. This stage was provided with graphic user interface that mentioned before. Furthermore, this stage also can be say our image retrieval system for indexing and matching in online mode. First, the initialization was done once the GUI was started. Afterwards, the feature extraction, KNN KD tree search, TF-IDF weighting and cosine law matching was activated when the “Match” button in GUI was pressed. The logo image that loaded in GUI will pass the image to this feature extraction part, so it can process all the parts and give a result, logo name. The first four parts is same as training stage that we mentioned before at section 4. The only difference is the training stage

processes lots of images for the needed in database and application stage only do for one query image only. Beside this, the matching part, we used cosine law equation to implement whether the query image and images from database is match or not. Moreover, this part was accelerated by using GPU, Nvidia Gefore GT 9400. The computational of cosine law will be passed to GPU and compute it in parallel. Thus, the elapse time is reduced.

In table 5.1, the DAT files that generated from training stage in offline mode was read by application stage for further operation.

Table 5.1: Input Data File in "data" directory file

File name	Description
featurepoint.dat	Feature space
idfscore.dat	idf score without log operation
idfscore1.dat	Idf score
knnindices.dat	Indices for visual words found in each database image
knnindicestem.dat	Indices for visual words found in query image
no_of_image.dat	Number of images in database
no_point_each_image.dat	Size of feature points for each image in database
number_image.dat	Number of images in database
Numberofcluster.dat	Number of visual words
rootsquare_tfidf.dat	Square root of sum of square of tfidf for each image in database
size_allpoint.dat	Size of feature space
size_eachpoint.dat	Size of feature points for each image in database
tfidf_database.dat	TF-IDF score for each image of database
tfscore.dat	TF score for each image of database
Vocabularytree.dat	Vocabulary tree (dictionary)

5.2 Initialization

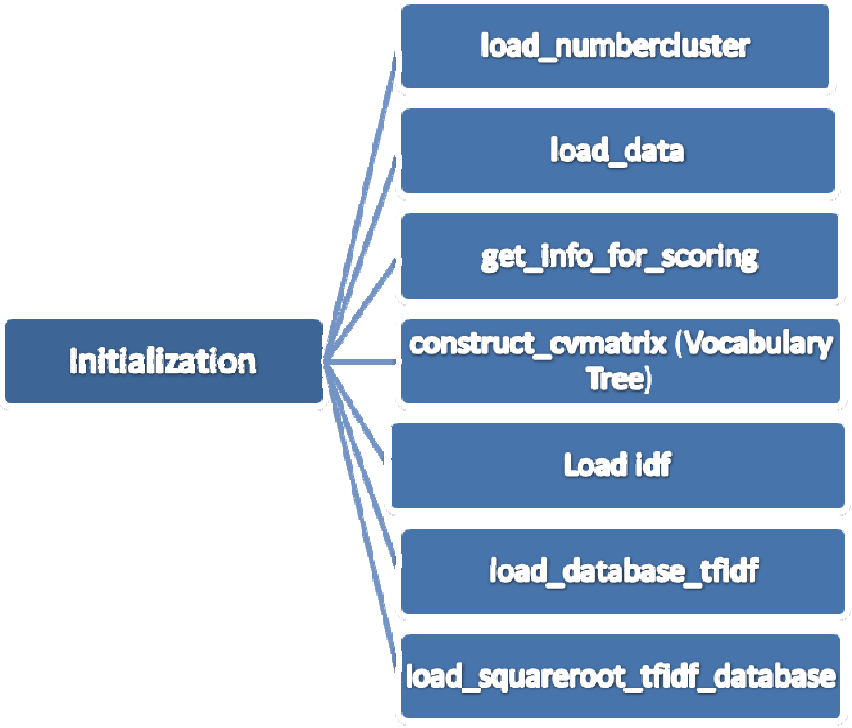

Figure 5.2: Overview of initialization

In this section, all the data needed from the "data" file was read and passed into the application stage. The operation for load_numbercluster, load_data, get_info_for scoring, construct matrix (vocabulary tree loading) was mentioned at training stage. Thus we focus on load idf, load_database_tfidf and load_squareroot_tfidf_database. The pseudo codes of them were shown at below.

Pseudo code for function `load_idf(idfmatrix,no_cluster);`

```
Cvmat:idfscore
load_idf (parameters: pointer to idfscore, number of visual word)
For i from 0 to number of visual word
        Read the idf data from data/idfscore1.dat
        Write the idf data to the idfscore matrix[0][i]
EndFor
Return (pointer to idfscore)
```

Pseudo code for function

`load_database_tfidf(number_of_image,no_cluster,database_tfidf);`

```
Cvmat:tfidfscore
load_database_tfidf (parameters: number of image, number of visual word, pointer to tfidfscore)
For i from 0 to number of image
        For j from 0 to number of visual word
                Read the tfidf data from data/tfidf_database.dat
                Write the tfidf data to the tfidfscore matrix[i][j]
        EndFor
EndFor
Return (pointer to tfidfscore)
```

Pseudo code for function

`load_squareroot_tfidf_database(sum_sr_database,number_of_image);`

```
Vector:sum_sr_database
load_squareroot_tfidf_database (parameters: pointer to sum_sr_database, number of image)
For i from 0 to number of image
        Read the value data from data/rootsquare_tfidf.dat
        Write the value data to the sum_sr_database[i]
EndFor
Return (pointer to sum_sr_database)
```

5.3 Feature Extraction

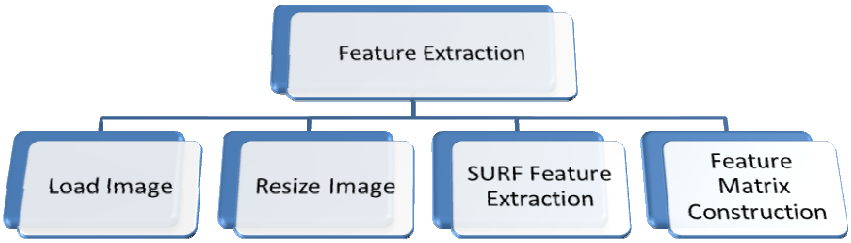

Figure 5.3: Process flow of feature extraction

The feature extraction part is same as the feature extraction part from training stage at chapter 4. The difference is this part is process one image only from GUI. The image was loaded from GUI and passed this section. Therefore, this section can be referring to the 4.2 section.

5.4 KNN KD Tree Search

The process is same with section 4.3. It performs the KNN KD Tree search and save the indices (visual words) found in data/knnindicestem.dat.

Pseudo code for function `knnsearch(query,vocabulary_tree,no_knn);`

```
knnsearch (parameters: query matrix, vocabulary matrix, number of knn)
Setup Kd tree index parameter
Setup index parameter, search parameter
Construct resulting_indices matrix
Construct resulting_distance matrix
Perform knn search
Call save_indice(parameters:"data/knnindicestem.dat",resulting_indices)
        For i from 0 to rows of query matrix
                For j from 0 to number of knn
                        Read the value in resulting_indices[i][j]
                        Write to "data/knnindicestem.dat"
                EndFor
        End For
```

5.5 TF-IDF Weighting

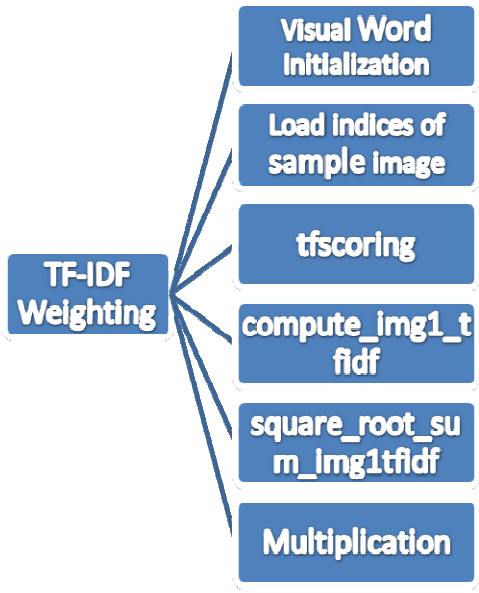

Figure 5.4: Process flow of TF-IDF weighting

This part has same concept with the TF-IDF part in chapter 4. However, the data needed for cosine law calculation was executed in this section. The multiplication of square root of sum of square of tfidf for query image and Square root of sum of square of tfidf for database image was computed at here and represented as vector format. The pseudo code was shown below that different with chapter 4.

Pseudo code for function `load_indice( img1_indice,no_cluster);`

```
cvmat:img1_indice
load_indice (parameters:pointer to img1_indice, number of visual word)
For i from 0 to size of SURF poins for sample image
        For j from 0 to number of visual word
                Read the index data from data/knnindicestem.dat
                Write the index data to the img1_indice matrix[i][j]
        EndFor
EndFor
Return (pointer to img1_indice)
```

Pseudo code for

`tfscoring(img1_indices,tfmatrix,img1_visualword,no_cluster,num_knn,ipts1);`

```
Vector: ipts1,img1_visualword
Cvmat:img1_indices, tfmatrix
tfscoring (parameters: img1_indice,pointer to tfmatrix,img1_visualword,number of  visual word, number of knn,ipts1)
For j from 0 to number of knn
        For m from 0 to number of visual word
                f= value [0][j] in matrix for  img1_indices
                If f equal to m
                Then img1_visualword[m] increased by 1
        EndFor
EndFor
For j from 0 to number of visual word
        Tfdata= img1_visualword [j]/ size of ipts1
        Write the tfdata into tfmatrix[0][j]
EndFor
Return (pointer to tfmatrix)
```

Pseudo code for function
`compute_img1_tfidf(img1_tf,img1_idf,img1tfidf,no_cluster);`

Cvmat:*img1_tf, img1_idf, img1tfidf*

compute_img1_tfidf (parameters: *img1_tf,img1_idf* pointer to *img1tfidf, number of visual word*)

For i from 0 to *number of image*

    For j from 0 to *number of visual word*

        Read the tf data from *img1_tf* matrix [0][j]

        Read the idf data from *img1_idf* matrix [0][j]

        Tfidf=tf x idf

        Write the tfidf data to the *img1_tfidf* matrix[i][j]

    EndFor

EndFor

Return (pointer to *img1_tfidf*)

Pseudo code for function
`square_root_sum_img1tfidf(img1tfidfmat,no_cluster,sum_sr_img1);`

Cvmat:*img1tfidf*

Vector:*sum_sr_img1*

square_root_sum_img1tfidf (parameters: *img1tfidf, number of visual word, sum_sr_img1*)

Sum of Square of tfidf=0

For j from 0 to *number of visual word*

    Read the tfidf data from *img1tfidf* matrix[i][j]

    Sum of Square of tfidf = Sum of Square of tfidf + (tfidf)^2

EndFor

sum_sr_img1=Square root of sum of square of tfidf

Return (*sum_sr_img1*)

Pseudo code for function multiplication

Vector :*multiplenum , sum_sr_img1, sum_sr_database*

For i from 0 to *number of image*

    Value= the data from *sum_sr_img1* x the data from *sum_sr_database [i]*

    Write the value to *multiplenum[i]*

EndFor

5.6 Cosine Law Matching

Figure 5.5: Process flow of cosine law matching

This section is the crucial part for the matching. As we know the cosine law equation is find out the angle θ between two vectors. The formula 5.1 is cosine law equation and shown at below.

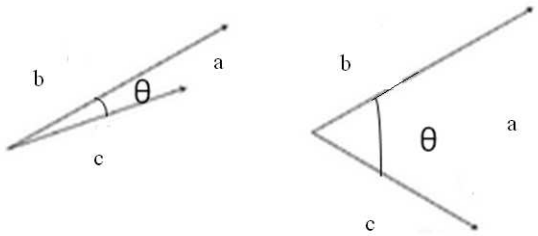

Figure 5.6: Two vectors

$$\cos A = \frac{b^2 + c^2 - a^2}{2bc} \tag{5.1}$$

By applied the equation 5.1, we will get the cosA value, if the cosA is 1, that means two lines are overlapped with each others.

$$\cos A = \frac{< b, c >}{|b||c|} \tag{5.2}$$

$$\frac{tfidf_i \times tfidf_j + tfidf_{i+1} \times tfidf_{j+1} + tfidf_{i+2} \times tfidf_{j+2} + \cdots + tfidf_{ni} \times tfidf_{nj}}{\sqrt{tfidf_i{}^2 + tfidf_{i+1}{}^2 + tfidf_{i+2}{}^2 + \cdots tfidf_{ni}{}^2} \times \sqrt{tfidf_j{}^2 + tfidf_{j+1}{}^2 + tfidf_{j+2}{}^2 + \cdots + tfidf_{nj}{}^2}} \tag{5.3}$$

Where i is zero from query image, j is zero from database image, n is number of visual word

From equation 5.2, this formula is written in vector form. Thus we can conclude that equation 5.3 is equal to equation 5.2. So, the similarity between the images is found. When the value close to 1, which means θ is zero, the two vectors are overlapped. Therefore, the TF-IDF value from database images and query image can be treated as vectors in higher dimension. By using this concept, after applied the equation 5.3, the more the value approach to 1, the more similarity the two images have.

For determination, it is done by using switch case in c++. The variable noi was obtained and it indicated the position of the image in database. Besides that, we know there are there images in each class. Therefore, the noi divided by 3 in integer form pass to the switch case to select the image according to value of noi after division.

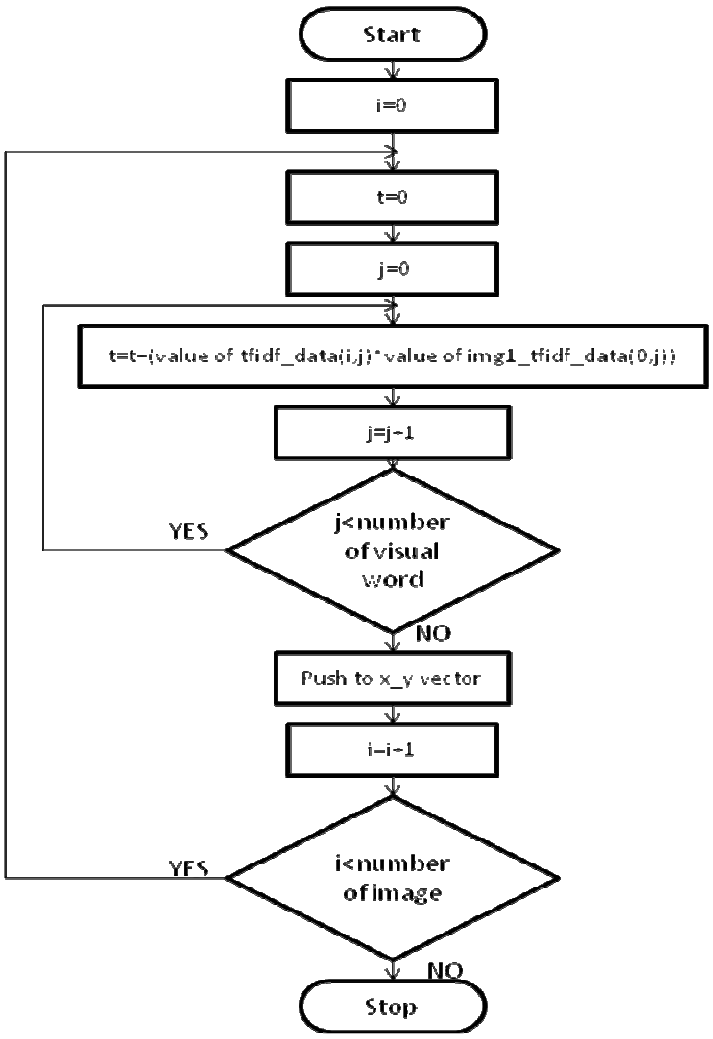

Figure 5.7: Cosine law computation

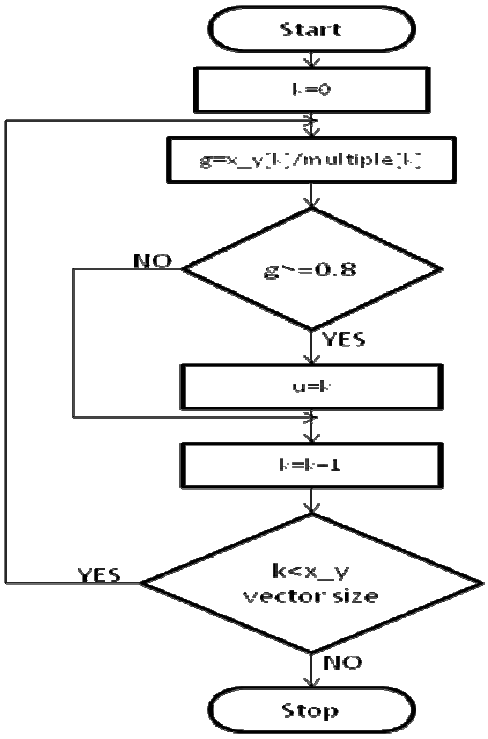

Figure 5.8: Cosine law computation II

Pseudo code for function `cosinelaw(xy,number_of_image,no_cluster,tf_idf_database,img1tfidfmat,multiplenum,similaritylist,noi)`

Cvmat: *img1tfidfmat, tf_idf_database*

Vector:*multiplenum*

Parameters:*number of image,number of visual word,*pointer to *noi*

For i from 0 to *number of image*

    Initialize t=0

    For j from 0 to *number of visual word*

        Value= the data from i*mg1tfidfmat* [0][j]x *tf_idf_database* [i][j]

        t=t+value

    EndFor

    Write t to the *xy[i]*

EndFor

For k from 0 to *size of xy*

    g= *xy[k]/multiple[k]*

    If g<0.8

    Then *noi*=k

EndFor

Return (pointer to noi)

Pseudo code for matching

```
Parameters:noi
Initialize z
noi=noi/3 (integer format)
switch (noi)
```

Table 5.2: Switch case

```
case 0:z="100plus";break;
case 1:z="acer";break;
case 2:z="adidas";break;
case 3:z="airasia";break;
case 4:z="airjordan";break;
case 5:z="amd";break;
case 6:z="apple";break;
case 7:z="asus";break;
case 8:z="ati";break;
case 9:z="avast";break;
case 10:z="avg";break;
case 11:z="blizzard";break;
case 12:z="bmw";break;
case 13:z="burgerking";break;
case 14:z="caltex";break;
case 15:z="capcom";break;
case 16:z="carrefour";break;
case 17:z="celcom";break;
case 18:z="cocacola";break;
case 19:z="compaq";break;
case 20:z="dell";break;
case 21:z="digi";break;
case 22:z="domino";break;
case 23:z="ea";break;
case 24:z="esso";break;
case 25:z="facebook";break;
case 26:z="firefox";break;
case 27:z="ford";break;
case 28:z="google";break;
case 29:z="honda";break;
case 30:z="hp";break;
case 31:z="hyundai";break;
case 32:z="ibm";break;
case 33:z="intel";break;
case 34:z="johnwiley";break;
case 35:z="jusco";break;
case 36:z="kappa";break;
case 37:z="kfc";break;
case 38:z="lee";break;
case 39:z="levis";break;
```

```
case 40:z="LG";break;
case 41:z="logitech";break;
case 42:z="macfee";break;
case 43:z="marrybrown";break;
case 44:z="mazda";break;
case 45:z="mcdonald";break;
case 46:z="mcgrawhill";break;
case 47:z="Mercedes";break;
case 48:z="mitsubishi";break;
case 49:z="motorola";break;
case 50:z="munchy";break;
case 51:z="nike";break;
case 52:z="nintento";break;
case 53:z="nissan";break;
case 54:z="nokia";break;
case 55:z="nvidia";break;
case 56:z="pearson";break;
case 57:z="pepsi";break;
case 58:z="perodua";break;
case 59:z="petronas";break;
case 60:z="pizza";break;
case 61:z="polo";break;
case 62:z="pringles";break;
case 63:z="proton";break;
case 64:z="publicbank";break;
case 65:z="puma";break;
case 66:z="reebok";break;
case 67:z="samsung";break;
case 68:z="sega";break;
case 69:z="shell";break;
case 70:z="sony";break;
case 71:z="sonyericsson";break;
case 72:z="spritzer";break;
case 73:z="squaresoft";break;
case 74:z="toshiba";break;
case 75:z="toyota";break;
case 76:z="utar";break;
case 77:z="vaio";break;
case 78:z="windows";break;
case 79:z="youtube";break;
default: cout<<"find nothing";
```

5.7 GPU Acceleration

In GPU acceleration part, we used CUDA toolkit 2.3 and NVidia GPU Computing SDK to setup for programming by using GT 9400. We used it to accelerate the speed of computation in cosine law part and return the noi to the switch case. Therefore, we transfer the TF-IDF data of database images and query image from CPU addresses to GPU addresses. It executes the multiplication process of the TF-IDF data of database images and query image in parallel and bring out the value to the CPU to do division.

CHAPTER 6

EXPERIMENTAL RESULTS AND ANALYSIS

6.1 Experiment Setup

As we mentioned in previous chapter, our logo recognition system is divided into two mode, offline mode and online mode. In offline mode (training stage), we used 30 classes, 170 logo images that change in rotation, illumination, and scale as our tested logo images. Then, we set the different parameters which are number of visual words and number of KNN to find out the similarity between the tested logos and database logos. This is our first experiment. The second experiment is use the different threshold value of the implementation of the logo and find out their performance.

6.1.1 First Experiment Setup

First, as the previous section mentioned, 30 classes, 162 logo images have been used for tested images. This experiment showed out the result about the similarity change between tested logos and database logos in rotation, illumination, scale, number of KNN and number of visual words. Therefore, we can know the effect and changes in similarity by changing the number of visual words and number of KNN. Therefore, the optimize selection of the number of visual words and number of KNN were obtained.

First, 10 classes' logo images with 5 logo images in each class are changing in illumination. From the figure 6.1, the first logo images is increase its brightness by 50, second one is 25, third one is the original logo, fourth one is decreased its brightness by 25, the last one is decreased by 50. The 10 classes' logo images are along this kind of setup as our tested images.

Figure 6.1: Tested logo with illumination change

Secondly, another 9 class's logo images with 7 logo images in each class are changing in rotation from 0 degree to 30 degree. From the figure 6.2, the first logo is the original logo, after that, the logo is rotated by 5 degree, 10 degree, 15 degree, 20 degree, 25 degree and 30 degree. The 10 classes' logo images are along this kind of setup as our tested images.

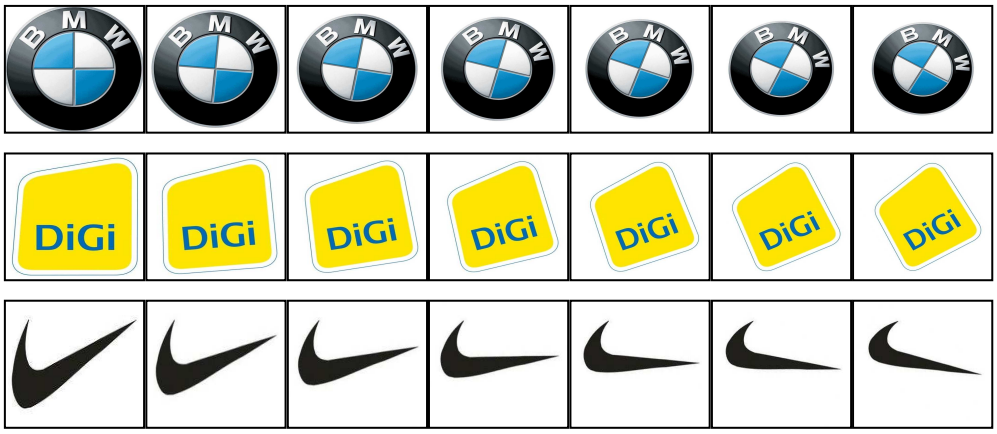

Figure 6.2: Tested logo with rotation change

Figure 6.3: Tested logo with scale change

Thirdly, another 10 class's logo images with 5 logo images in each class are changing in scale from 25% of the original size to 200% of the original size. From the figure 6.3, the first logo is 25% of the original logo, second is 50% of the original logo, third one is the original logo, fourth and fifth are 150% and 200% of the original logo. The 10 classes' logo images are along this kind of setup as our tested images.

Then, these 30 classes, 162 tested images were setup. Therefore, we used these tested images to find out the changes of the similarity by changing the number of visual words and number of KNN. In addition, we also can find out the similarity changes between the original image in database and the images that changed in rotation, illumination and scale. Therefore, we set the number of visual words to 100, 150 and 200. This parameter was changed in training stage program at part 2, building of vocabulary tree, "number_of_leaf" parameter. Noticed that we have to check out the centroids found in vocabulary tree and ignore the centroid point is zero. It is because, in that case, the node is choosing not to expand.

Afterwards, the KNN value was changed in part 3 and 4, featureextraction_knntreesearch and tf_computational, no_knn parameter to 3, 5 and 7 at the case when number of visual words is 100, 150 and 200. Therefore, after the parameters were changed, the data that generated were the database of the application stage. The result was generated after the application stage by input the tested images.

6.1.2 Second Experiment Setup

The objective of this experiment is finding out of the performance of the system by changing the threshold value. The threshold value was set up to 0.86, 0.88, 0.90 and 0.92. Therefore, the performance and result were obtained. This setting was done in application stage only. The parameter of threshold value is in the tfidf.cpp, cosinelaw function. In this experiment, we choose KNN=5, number of Visual words =150 as our database.

Here, the 16 logo images that not include in database, the 47 logo images that not edited from database image, the 240 logo images in database and the 162 tested images from experiment one were combined together and become 465 tested images of this experiment. After the result, the threshold value will be determined according to the performance after result.

Figure 6.4: The logo images that not include in database

Figure 6.5: The logo images that not edited from database image

6.2 Result

Because of the huge data in result, the result for experiment 1 and 2 will be shown in appendix A and B.

6.3 Effect of the Change of KNN and Visual Word

In our image retrieval system, there are three parameters can be tuned and affect all the performance of the system. The first one is number of cluster (centroids) from the Hierarchical K-means Clustering function. The Hierarchical K-means Cluster was provided in Opencv. Therefore, this function was used to find out the number of centroids to construct a vocabulary tree. There are three parameters at here, however, in most of the time, the other two parameters is fixed. These parameters are branch factor which set the maximum number of the centroids can be expanded from one centroid while the other parameter, cb index define the distance between two centroids whether it needed to expand or not if too close. Therefore, we just need to change the number of cluster to define our visual words according to the centroids of the cluster.

Beside this parameter, number of k in KNN also acts a crucial parameter in over system. We use the KNN Search function that provided in Opencv to search the nearest neighbour points. These two parameters need to be tuned when the database and the requirement are changed. Otherwise, it will affect the accuracy and the speed of the system.

From the result 1 for experiment 1, we noticed that when KNN increase, the similarity between tested image and corresponding database image was increase. It is because when the KNN increase, it is more tolerance and high matching probability to match the similar image. However, the increase of KNN will make the irrelevant image match with the database image. This wrong match probability will also be increase.

After that the changes of the number of visual words also increase the distinguisher rate. It decreases the probability of the wrong match. However, the ability of matching with the similar image or the image that change in rotation, illumination and scale will be decreased.

Therefore, the overall similarity value between tested images and its corresponding database image that show in result 1 were obtained. Beside this result,

the list of the similarity value between tested image and 240 database images also were obtained in result 0. Note that result 1 is come from result 0 and result 0 data was too huge that cannot show in this report. So, we extracted the tested images and its corresponding images matched from result 0 to result 1. From these two results, we find out the average value for the similarity value between tested images and its corresponding database image and the similarity value between tested image and 240 database images in different parameters (KNN and Visual word) setup.

Table 6.1: The Average similarity value between tested images and its corresponding database image

KNN value	Visual word=100	Visual word=150	Visual word=200
3	0.9029	0.8804	0.8640
5	0.9232	0.9081	0.8850
7	0.9411	0.92106	0.8948

Table 6.2: The Average similarity value between tested images and all database image

KNN value	Visual word=100	Visual word=150	Visual word=200
3	0.4425	0.37	0.32
5	0.51	0.449	0.38
7	0.56	0.50	0.43

Table 6.3: Distinguish rate (The value in table 6.1 divide by the value in table 6.2)

KNN value	Visual word=100	Visual word=150	Visual word=200
3	2	2.38	2.7
5	1.84	2	2.38
7	1.67	1.84	2

The average match value was shown in Table 6.1 while the average the similarity value between all database images and each tested image was shown in Table 6.2. In Table 6.1, the increase of KNN will increase the similarity value; the increase of visual word will decrease the similarity value. However, the value for visual word =100, KNN=5 was close to the value for visual word =150, KNN=7, so the others value. Beside this, the table 6.2 also give same result. In conclusion, we can know that the if the visual word increase, the increase of KNN value will make this set same with the previous set that not increase in visual word and KNN. It is because the increase of visual word will increase the distinguish rate, the increase the KNN will decrease the distinguish rate and they will balance together and remains same. The distinguish rate is the fraction of these two table data. When the rate is higher, that means the similarity of other irrelevant images and the corresponding image to one tested image is differ so much and can be discriminated easily and avoids miss matching with irrelevant images. From these three table, we chose the set that average in result which is visual word = 150 and KNN =5 as our database setup parameter.

6.4 Performance Measurement

As the technology for the image classification and retrieval system increase, more and more evaluation and measurement method for the system performance was proposed and introduced nowadays. In addition, these measurement methods need a collection of the tested image to test the performance of the system. After the measurement is obtained, the evaluation of system can be concluded.

There are some fundamental elements for the evaluation are true positive, true negative, false positive and false negative. The evaluation uses these elements to calculate the performance of the system. These elements are obtained in measurement. By using this evaluation method, we can know the effect of changing in the threshold value. Therefore, from the evaluation, we also can de

First, we need to know the concept of these elements. This concept is come from the confusion matrix.

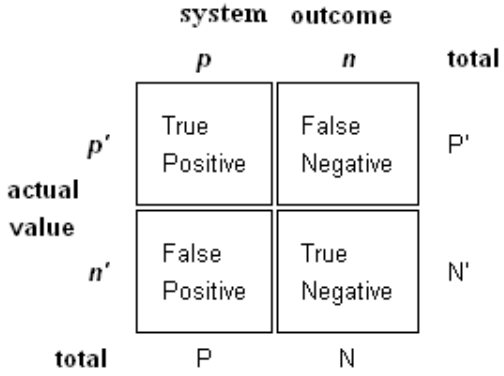

Figure 6.6: Table of Confusion Matrix

As the figure 6.1 shown, true positive (TP) or hit means that the image that the system found is really matching in actual word. False negative (FN) or miss is the system found the image but the database did not contain this image actually. False Positive (FP) or False alarm is the image that the system found is not matching in actual. True negative (TN) or correct rejection is the system cannot find the image that it should not be found in system actually. P is the total of TP and FP while N is the total of FN and TN. Therefore, by measuring this value for these elements, we can know the evaluation of the system.

From the result of the experiment 2, the TP, FP, FN and TN value were obtained. So, we used these elements to measure the performance of the system.

Table 6.4: The Effect of Changes of Threshold Value

Threshold Value	TP	FP	FN	TN	P	N
0.88	310	140	5	10	315	150
0.90	333	116	0	16	333	132
0.92	337	112	1	15	338	127

6.4.1 Accuracy

Accuracy is the measurement that how well the system can recognize correctly.

$$ACC = (TP + TN) / (P + N) \quad (6.1)$$

By using the equation 5.1, we obtained the accuracy for each threshold value set.

Table 6.5: The Accuracy for Changes in Threshold Value

Threshold value	Accuracy
0.88	0.68
0.90	0.751
0.92	0.757

6.4.2 Detection Rate

Detection Rate is the measurement of the performance that how the system can detect and distinguish the database logo images and non database logo images.

$$Detection\ Rate = TP/ (TP+FN) \quad (6.2)$$

Table 6.6: The Detection Rate for Changes in Threshold Value

Threshold value	Detection Rate
0.88	0.98
0.90	1
0.92	0.997

6.4.3 False Alarm Rate

False alarm rate, as known as false positive rate or fall-out, it shows the probability of the system classified a wrong matching with other image in database.

$$FAR = FP / FP+TN \quad (6.3)$$

Table 6.7: The False Alarm Rate for Changes in Threshold Value

Threshold value	False Alarm Rate
0.88	0.93
0.90	0.878
0.92	0.881

6.4.4 False Negative Rate

False negative rate shows the probability of the system match an image that not occurs in database.

$$FAR = FN / TP+FN \quad (6.4)$$

Table 6.8: The False Negative Rate for Changes in Threshold Value

Threshold value	False Negative Rate
0.88	0.016
0.90	0
0.92	0.003

6.4.5 Sensitivity or True Positive Rate

Sensitivity also known as hit rate or recall, is successful rate to classify the different images in database.

$$Sensitivity = TP / (TP + FN) \quad (6.5)$$

Table 6.9: The Sensitivity for Changes in Threshold Value

Threshold value	Sensitivity
0.88	0.984
0.90	1
0.92	0.997

6.4.6 Specificity or True Negative Rate

Specificity determines the successful rate that the system can match the images in database correctly. The higher the specificity, the higher the probability that matches wrongly with the other images in database.

$$SPC = TN / (FP + TN) \quad (6.6)$$

Table 6.10: The Specificity for Changes in Threshold Value

Threshold value	Specificity
0.88	0.066
0.90	0.1212
0.92	0.118

6.4.7 Precision or Positive Predictive Value (PPV)

Precision is the fraction of the image retrieved in database that is correctly matched.

$$PPV = TP / (TP + FP) \quad (6.7)$$

Table 6.11: The Precision for Changes in Threshold Value

Threshold value	Precision
0.88	0.688
0.90	0.74
0.92	0.75

6.4.8 Negative Predictive Value (NPV)

Negative Predictive value show that how the system can implement correctly that the non database image is not in database.

$$NPV = TN / (TN + FN) \quad (6.8)$$

Table 6.12: The NPV for Changes in Threshold Value

Threshold value	NPV
0.88	0.67
0.90	1
0.92	0.938

6.4.9 False Discovery Rate (FDR)

False discovery rate shows that the probability that the system may match wrongly.

$$FDR = FP / (FP + TP) \quad (6.9)$$

Table 6.13: The FDR for Changes in Threshold Value

Threshold value	FDR
0.88	0.31
0.90	0.258
0.92	0.249

6.4.10 Receiver Operation Characteristic (ROC)

ROC is a graphical plot of sensitivity which is the graph of true positive rate versus false positive rate when the threshold value change. It provided us to select the optimize threshold value.

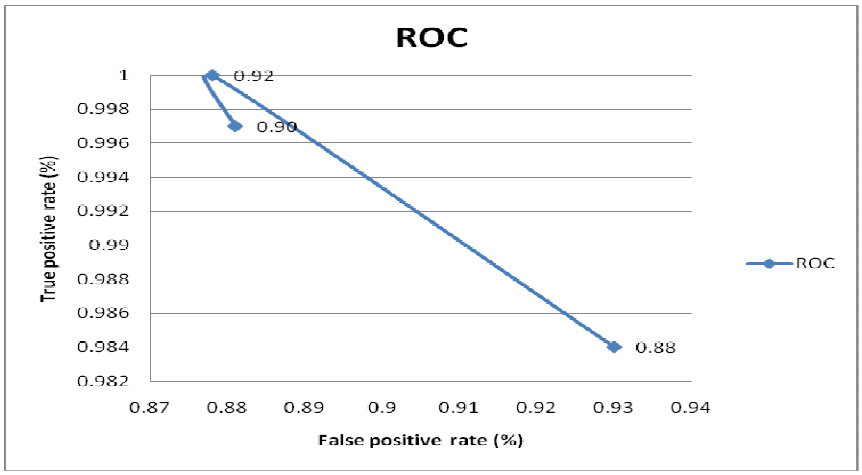

Figure 6.7: ROC

The coordinate (0, 1) is defined as perfect classification. Thus, the coordinate that close to this (0,1) can be said that close to perfect classification. At here, threshold value 0.90 is close to perfect classification.

6.4.11 F1 score

F1 score is a measure of test accuracy, it consider both precision and accuracy and is weighted average of them. F1 reaches its best value in 1 and worst in 0.

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{6.10}$$

Table 6.14: The F1 Score for Changes in Threshold Value

Threshold value	F1 score
0.88	0.677
0.90	0.74
0.92	0.758

6.4.12 Threshold Value Selection and Analysis

Table 6.15: The Performance Measurement for Each Threshold Value

Evaluation	0.88	0.90	0.92
Accuracy	0.68	0.75	0.757
Detection Rate	0.98	1	0.997
FAR	0.93	0.878	0.881
FNR	0.016	0	0.003
Sensitivity (Recall)	0.984	1	0.997
Specificity	0.066	0.1212	0.198
Precision	0.688	0.74	0.75
NPV	0.67	1	0.938
FDR	0.31	0.28	0.249
F1 Score	0.677	0.74	0.738

From the Table 6.15, the increase the threshold value, the good effect measure such as accuracy rate, detection rate, recall, specificity, precision, NPV and F1 score also increase. When the value closer to 1, the better the performance. Moreover, the bad affect measure such as FAR, FNR and FDR was decrease as well. The smaller these value, the better the performance.

Therefore, the value of 0.88 was not concerned due to the poor evaluation. We added up the value for good effect measurement and obtained the sum of these values which are 5.35 and 5.38 for 0.90 and 0.92. The full mark is 7. They were so close and hard to select. However, from the ROC graph, 0.9 is better than 0.92. Thus, we chose the 0.90 as our threshold value.

6.5 Speed Analysis

From model 1 to model 3 that mentioned before in section 3, prototyping process, the searching speed was increase tremendously. For table 6.12, in same condition that the database contains 240 logo images, the searching speed when search all images for each model were recorded down. This result was obtained by using intel dual core processor 1.73GHz and 1.5G RAM without GPU.

Table 6.16: The Searching Speed for Each Model for 240 database image

Model	Searching Speed
Model 1	36s
Model 2	6s
Model 3	0.25s

The model 3 which is the final model that used the concept of vocabulary tree gives a remarkable result in image retrieval system. The model 1 used the nearest neighbour ratio which high cost in computational increase the elapse time in searching. For the model 2, KNN will slow down the speed as the capacity of it increase. In this case,

the whole feature points for 240 images are about 37000. That's why the long time elapsed.

However, the searching speed for model 3 is very fast because of the simple computational. Notice that this result is record from non GPU system result. By using GPU to accelerate, the time elapsed will down to micro seconds for 240 images.

6.6 Error Analysis

From the result 1 and result 2, the failure to match correctly occurs because of some reason. First, is the problem of the SURF descriptor, although it is strong descriptor, it has its limit too. As we noticed in result 1, the change in scale and illumination give a pretty good result but poor in rotation. When the image is rotated up to 10 or 15 degree, the classification rate is slow down and easily miss match with others image in database. Secondly, the database image should very similar with the tested image and less background interrupt. It is because the influence of the background will be extracted with its feature points and take counts in the system. This will result in the similarity value will small between them and database image. Beside these, the contour of the logo must be obvious or the feature point of it might be not extracted of give wrong information of the surround of the interest point. Furthermore, the white background and black logo and black background and white logo which are same logo and contour, their similarity is close to 0 for the SURF descriptor. Therefore, the database images have to be increase more with different environment to increase the ability of the system. It is because in the result 2, the tested logo in different environment has less similarity with its corresponding database image and hard to classified it.

CHAPTER 7

CONCLUSION

7.1 Comment

In this report, the analysis and discussion of the result were mentioned in section 6. In overall, our system is performed well and flexible. As the evaluation give a 5.35 points with full mark is 7, this can be said our system is good and pass the evaluation. Beside this, the searching speed also very fast and can be used for video tracking system. This system not only recognizes the logo image only, the human face or other images also can recognize well. Therefore, this system can be said very flexible. Beside these, it is user friendly. Easy to setup and tune the system. The user can build up the database as they like and requirements. Moreover, GUI provided the user more easily uses our system to execute the program.

7.2 Summary

In our logo recognition system, we used 80 classes logo with 240 logo images each class as our database. We used the concept of vocabulary tree in our retrieval system. In our system, the visual word number was set to 150 and KNN value was set to 5 after analysis and selection. This system is well performed in illumination change and scale change, also the logo image with less influence of the background. For rotation and affine change, SURF gave a relatively poor result to the previous

changes. Furthermore, GPU was added to accelerate the similarity function. The speed of the search all the database image is 270ms. The evaluation of the system is 5.33 points over 7 points which show the system is performed nicely.

7.3 Conclusion

In conclusion, our system strong in recognize the logo images that changes in illumination, scale and similar to database image but weak in rotation and affine relatively. The evaluation of the system is 5.33 points over 7 points which show the system is performed nicely. Our system cannot recognize the logo image that inverted colour but same logo contour image. The speed is tremendously fast and response fast also. Therefore, our system is achieved our target we set and well developed in retrieval system and speed acceleration.

7.4 Future Work

For the future work, speed and accuracy are the main issue for the whole system. In speed section, in our system, the feature extraction costs about 3 to 4 seconds. It is too long for the whole system. This section can used CUDASURF to accelerate. Beside this, the KNN search part also cost about 27ms to build up the indices. This part also can be accelerated by using GPU.

For the accuracy part, SURF descriptor has its own limit. Although SURF is very fast, the invariant against rotation and affine is not good with other changes like illumination. SIFT is believed that can give a good performance against this changes.

The storage of the data that generated in training stage (Prework) can be changed by store the data in line by line. It is because the DAT file has limit in width. As the database increase up to 400 images, the accuracy will start to decrease if the data is not stored line by line.

REFERENCES

David Doermann, Ehud Rivlin, and Isaac Weis, “Logo Recognition using geometric Invariants,” in Document Analysis and Recognition, pp. 894-897, 1993.

Andre Folkers and Haran Samet, “Content-Base Image Retrieval using Fourier Descriptors on a Logo Database,” 16th International Conference on Pattern Recognition, vol 3, p. 30521, ICPR'02.

A. Soffer and H. Samet, “Using Negative Shape Features For Logo Similarity Matching,” pp. 571-573, ICPR’98.

J. Neuman, H. Samet, and A. Soffer, “Integration of local and Global Shape Analysis for Logo classification,” Pattern recognition letters 23:1212, pp. 1449-1457, 2002.

J.R. Eakins, J.M. Broadman, and M.E. Grahman, “Similarity Retrieval of Trademark Images,” ACM Multimedia ’98, pp. 53-63.

J.L. Crowly, “Brand Identification Using Gaussian Derivative Histograms,” ICVS2003, LNCS 2626, 2003.

D. Lowe, “Object Recognition from Local Scale Invariant Keypoints,” Proceedings of the International Conference on Computer Vision, ICCV, pp. 1150-1157, 1999.

D. Lowe, “Local Feature View Clustering for 3-D Object Recognition,” Proceedings of the Conference on Computer Vision and Pattern Recognition, CVPR, 2001.

D. Lowe, “Distinctive Image Features from Scale Invariant Keypoints,” International Journal of Computer Vision, 2004.
K. Mikolajczyk and C. Schmid, “A performance Evaluation of Local Descriptors,” IEEE transactions on pattern analysis and machine intelligence, vol. 27, no. 10, October 2005.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, “SURF: Speeded Up Robust Features,” 9th European Conference on Computer Vision, Austria, 2006.

George Wolberg and Siavsh Zokai, “Robust Image Registration Using Log Polar Transform,” Proceedings of IEEE, International Conference on Image Processing, Sept. 2000.

Hu M. K., “Visual Pattern Recognition by Moment Invariants,” IEEE Transactions on Information Theory, vol. IT-8, pp. 179-187, 1962.

K. Mikolajczyk, Bastian Liebe, and B. Schiele, “Local Features for Object Class Recognition,” 10th IEEE International Conference on Computer Vision, ICCV2005, pp. 1792-1799, Beijing, China,17-20 Oct. 2005.

Guangyu Zhu and David Doermann, “Automatic Logo Detection,” in Proceedings of ICDAR'2007, pp. 864-868.

Luo Juan and Oubong Gwun, “ A Comparison of SIFT, PCA SIFT and SURF”, International Journal of Image Processing (IJIP) Vllume (3), Issue (4)

D. Nistér and H. Stewénius, Scalable Recognition with a Vocabulary Tree, *accepted for oral presentation at CVPR 2006*.

APPENDICES

APPENDIX A: Result 1

APPENDIX B: Result 2

APPENDIX C: Programming Code

APPENDIX D: Logo Recognition System Program

APPENDIX E: Database Logo