Intelligent Image Database System

Juthamas Kroswas

Raweewan Wasanakitsomboon

Bachelor of Engineering Program in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

2012

Thesis – Academic Year 2012

B.Eng. in Software Engineering

International College

King Mongkut's Institute of Technology Ladkrabang

Title : Intelligent Image Database System

Authors :

1. Ms. Juthamas Kroswas Student ID : 52090003
2. Ms. Raweewan Wasanakitsomboon Student ID : 52090020

Approved for submission

.....

(Dr. Ukrit Watchareeruetai)

Advisor

INTERNATIONAL COLLEGE

Date 17/9/2013

Intelligent Image Database System

Ms. Juthamas Kroswas 52090003
Ms. Raweewan Wasanakitsomboon 52090020
Dr. Ukrit Watchareeruetai Advisor
Academic Year 2012

ABSTRACT

Content-based image retrieval (CBIR) is a technique to search for image by using the content of image. The content-based of images refers to colors, shapes, textures, or any other information from the image data rather than keywords, tags, or descriptions implicated with the images.

An intelligent image database system is a computer-based system that can search for similar images in an image database by using a query image. Moreover, the system would be able to manage images imported by a user to collect in a database and can semi-automatically classify them into their own categories.

This project concentrates on two important tasks: 1) search for similar images in database by using a query image and 2) management of imported images by user to correct in a database according to date of imported and can semi-automatically classifying them into their own categories. In this project, we use an existing image processing technique, i.e., color histogram, this technique is calculate the color of image then return the histogram value of image. The value of the color histogram is used to compare a query image with each image in the database to obtain the best matching. Classifying category with imported images in order to response desire image correctly also uses the color histogram technique and adopts the basis of k-nearest neighbor (k-NN) and then find the majority of category name in k-NN. We made an experiment to evauate the accurracy of searching image. The result indicates that the developped system can retrieve images with accuracy of 85%.

Acknowledgements

We would like to express our deep gratitude and appreciation to Dr. Ukrit Watchareeruetai our research supervisors for his valuable and constructive suggestions, for guidance, enthusiastic encouragement and useful critiques of this research work. We would also like to thank for his assistance in keeping our progress on schedule. His willingness to give his time so generously has been very much appreciated.

And would like to appreciate and deep thank you for advice given by others professors of International College has been a great help in various useful information that helped us achieve good and correct according our goals.

Also thanks staffs of International College for supporting places to work and some equipment that is one of achievement part of our research.

Finally, we wish to thank our parents for their support and encouragement throughout our study.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Statement of problem and motivation

Nowadays, photographing using digital cameras and smart phone grow rapidly. Of course, the number of images not only personal collections but online also is never static. No one declined to search the desire images for reduce time instead of look around from the beginning. If your images collections contain large volume of data, finding desire images problem will occur.

Many program for searching images, both online and offline applications are made available so much for wide variety of task that useful which user can easy to perform a various tasks depend on what user really need. These applications are often created and developed by software developers who want to provide users with a better and easier way to search desire images precisely. Developers not only build software from existing software and fixed some of their problems for better operation but also try to improve from the good existing software for the best overall operation.

Resources of these applications are images from local or online database that also created by some software by unofficial third-party experts and even more some software's resources can be found in the form of user-generated content by themselves. For example, users can import their own images into public database or can access those images by only themselves in local database to search on these large resources of data both online and offline. Not so much software that use image is a query to express the content within image directly to search for desire image(s). Also having humans manually enter keywords for images in a large database can be inefficient and may not capture every keyword that describes the image. Many well-known systems usually use human manually enter keywords or sometimes description which associated with the desire image(s). That's it what system produce is not very similar with query input as user really need to be the result(s).

We applied searching image by keyword which is not accuracy much as content of image by used image is a query and retrieve the content directly within image out to

process similarity. They hope to retrieve the desired images come out for least of garbage data, so, first thing we have to understand is query.

A picture is worth a thousand words. Imagine the difficulty of cataloging a database of up to 100,000 images by providing a written description of each one. Not only would the cataloger have to describe the color, shape and texture of every element within each picture and its relationship to every other element; but also anyone searching for an image would have to guess exactly what words had been used in the description. With computers capable of displaying millions of colors and an effectively infinite number of shapes and textures, the task would be unenviable, if not impossible.

Due to currently intelligent image database system became very popular and interested in image processing. Spreading of photographing digital images need to manage depend on what operation user want including features of each image of those software to easier to achieve their goals. We need fundamental method for solving the large-sized of images that is CBIR.

Content-based image retrieval (CBIR) is technique that we chose to manage the searching process by content of image means analyze the actual contents rather than person manually enter keywords or some descriptions that is associated with image. The term 'content' in this context might refer to colors, shapes, textures, or any other information that can be derived from the image itself but for this software we chose color feature for matching images similarity.

Many CBIR systems have been developed but the problem to retrieve images on the basis of the pixel content remains largely unsolved. Various dimensions for example color, texture, shape and another used for compare similarity of two images but we selected one of them to process similarity of each image is measuring distance focused on color similarity that is obtained by calculate a color histogram which identifies the pixels' proportion in image by holding specific values.

Operates on a simple principle is the best way to query a database of images is to "show it" an image similar to the one you're seeking and to ask for all the images that match the sample in one or more features. For example, a photo researcher might select a shade of red and sketch a freehand outline of a spiky flower. In an instant, the spiky red flowers in the database that best match the drawing will pop up on the screen along with any other object that resembles a spiky red flower. The researcher can choose how

many images appear. Most will select no more than the 10 or 20 small "thumbnail" images that fill a single screen.

1.2 Objectives

In this project, we aim to do two main features: 1) searching and 2) categorization. For searching, we using CBIR is an important technique adopt for solving image retrieval in large databases. And applied color feature by used computing color histogram and compare between two images to obtain the similarity as results. In categorization, we can categorize images in users' database divided by what user want to declare category name. This categorize system used semi-automatic to train the system how to classify those images after user teach some experiments until system can learn it by themselves and helps user to assign category automatically. Also use histogram technique applied with basis of k-Nearest Neighbor (k-NN) method to classify those images along with semi-automatic to manipulate images.

We find out related work done that existing and approximate to our program and compared between strong and weak points and noticed that excepted searching and a little bit features such as import, show thumbnail, require internet (online system), almost all of them not have categorization, we aim search by using query image and categorization is mainly so, it better if we develop this software which not the same exist software.

1.3 Scope of project

We summarized the scope of our software and explain what does software do and limitations of the software. The basis of the software is search desire image which contained in personal database in user's computer. User has to input image is a query and result in sequence order from the most similar image to the least one. Also user can import images to database and automatically assign name of folder for imported images sort by date imported. All imported images can specify category by manually until system can recognize and assign each image into folder that be fitting with all images within that folder it called semi-automatic system. Moreover user can view imported images in folder 'all image' this folder contain all imported image that first time it is exist and view by imported date as well. User allows adding or deleting category as user want.

This software is appropriate for local system that mean in user's computer directory. Database means how we manage the images and process all data that existing

within user's computer to carry out. And it is offline software system to use for searching and categorization without any Internet connection.

1.4 Structure of thesis

The rest of this thesis is organized as follows:

Chapter 2 describes the comparison of existing software, related work and problem description.

Chapter 3 describes the technical background; explain searching management feature and tools used in this software.

Chapter 4 describes the software requirement specification including functional and non-functional requirements and software analysis and design descriptions.

Chapter 5 describes the structure of software system that represent by using class diagram and package diagram.

Chapter 6 describes the result of the experiment. And describes the effectiveness and the deficiencies of the software and evaluate the experimentation result.

Chapter 7 concludes of the thesis and future works.

Chapter 2

Problem description and related work

2.1 Existing image retrieval software

We discuss representative work done in content-based image retrieval in the recently. The two fundamental necessities for image retrieval are (1) Searching for a particular image item by using query image; and (2) Categorization.

And this table is showed the comparison between those of all that famous and successful for image retrieval. All software that show in table you may be known, like a Google images, we met the missing point of them are the image result will be retrieve by using color more than shape and texture so, the result will not similar with query at all, in other way it will show the same image with query image and they cannot specify category so, sometimes, the result will not efficient.

We show our software at the end blue column in Table 2.1, all feature that our software has is same as iPhoto and ISSBP but if you noticed all of them not have categorization, we aim search by using query image and categorization is mainly so, it better if we develop this software which not the same exist software.

Table 2.1 Comparison of features provided by existing software and by this project.

Feature \ Software	Google images[1]	iPhoto[2]	VIRaL[3]	Bigimbaz[4]	ISSBP or imense[5]	Intelligent Image Database System
Query is an image	yes (allow text annotations)	yes	yes	yes (allow text annotations)	yes	yes
External image query	yes(anyone personal data)	only within machine	yes(anyone personal data)	yes(anyone personal data)	only within machine	only within machine
Display summarize of total result	yes	yes	yes	yes	yes	yes
Internet required (online)	required	no (offline)	required	required	no (offline)	no (offline)
Importing	not allow	yes	not allow	not allow	yes	yes
Thumbnail	Show & Can select view of thumbnail	yes	yes	yes	yes	yes
Salient object detection (color,size,style,..)	yes	no	no	no	no	no
Specify category	no	no	no	no	no	yes

2.2 Analysis techniques and experiments

2.2.1 Perception of the hash algorithm

There is a technique related to this project called "perception of the hash algorithm" [4]. In this technique, two different images are generate and get fingerprint, then compare fingerprint of those image and count the number of different bit is called "Hamming distance value".

Implement this algorithm

- **The first step to reduce the size**: Will shrink the pictures to 8x8 size, a total of 64 pixels. The role of this step is to remove the details of the picture, leaving only the structure, shading and other basic information to get rid of different sizes, the ratio of the picture difference. At first, we are provided the original number of pixels in the desire image. For getting new size of pixels, we divide the original pixels by new desire size of pixels to reduce size of image. For example, if the original pixels size of image divided into 16 by 16 grid and need to reduce into 8 by 8 grid, and then divide 16 by 8, the result is equal to 2. Finally, combine 2 pixels of row and also 2 pixels of column into one pixel that shown in Figure 2.1 and Figure 2.2.

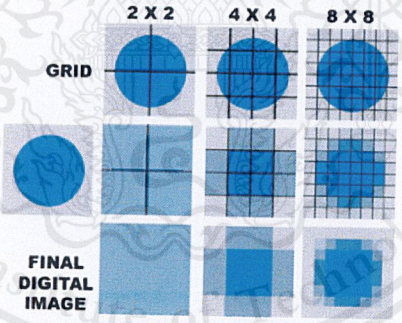


Figure 2.1 Reduce size of image from 16x16 grid to 8x8 grid

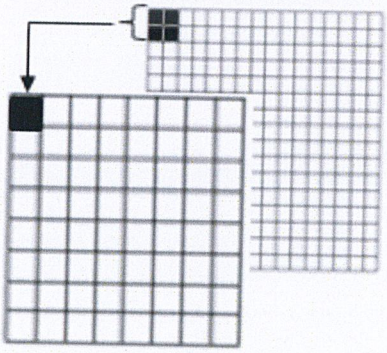


Figure 2.2 Remove the details of image by reduce size of pixel

Actually, the shape of image begins to appear in the 8x8 pixels. For more resolution (or pixels) will produce more details in the image. Many pixels are better for quality of an image.

- **The second step**, to simplify colors: Will be reduced after the picture into 256 gray. In other words, all the pixels of a total of only 256 colors. Allows 256 different intensities i.e., gray shades, each grid (or each pixel) contain the averaged color in gray scale, which gives you at least 0 – 255 range shown in Figure 2.3.

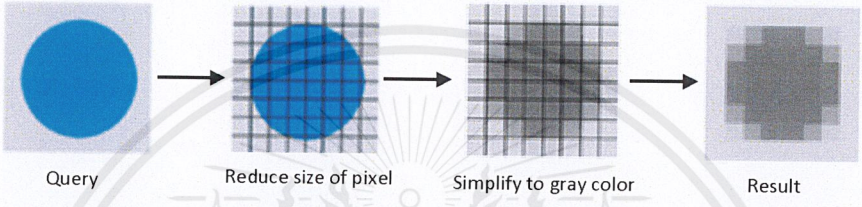


Figure 2.3 Simplify to gray color procedure

- **The third step** is to calculate the average: Calculate the average of all 256 gray levels.
- **The fourth** step is relatively gray pixel: Each pixel gray scale, and the average for comparison. Greater than or equal to the average, denoted by 1; less than the average, denoted by 0.
- **The fifth step** is to calculate the hash value: Compare the results of the previous step, together, constitute a 256-bit integer, which is the fingerprint of this picture, see figure 2.4.

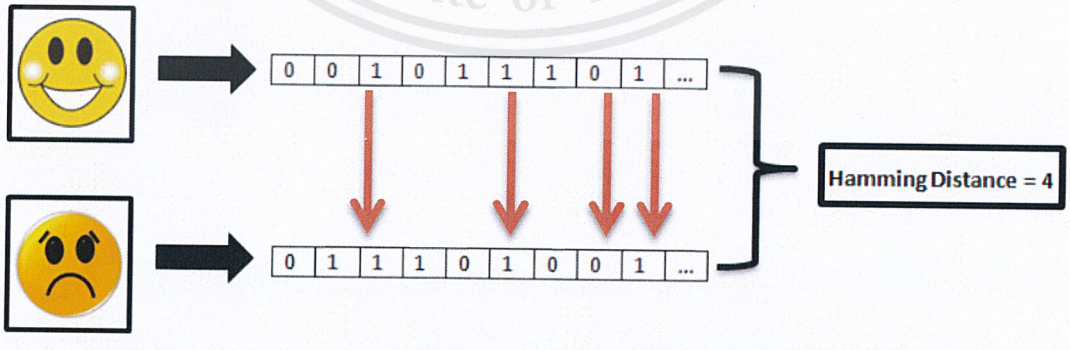


Figure 2.4 Example for comparison between two different images for finding hamming distance value

This example it contains 4 different bits of fingerprint so, hamming distance equal to 4. If hamming distance value is small that means these two images is more similar.

After you get a fingerprint, you can compare the different pictures, look at 64 the number of bits is not the same. In theory, this is equivalent to computing the Hamming distance (Hamming distance). If the same data bit is not more than 5, it shows two pictures are very similar; if greater than 10, it shows that this is two different pictures.

Chapter 3

Technical background

3.1 Searching management

3.1.1 CBIR technique

Many CBIR systems have been developed, but the problem to retrieve images on the basis of the pixel content remains largely unsolved.

Content comparison using image distance measures

The most regular method to compare two images in content-based image retrieval is using an image distance measure. An image distance measure compares the similarity of two images in various dimensions such as color, texture, shape, etc. For example a distance of 0 signifies an exact match with the query image, with respect to the dimensions that were considered as one may intuitively gather, a value greater than 0 indicates various degrees of similarities between the images. Search results then can be sorted based on their distance to the queried image. A long list of distance measures can be found in.

3.1.2 Operation of CBIR

The processes of CBIR (Figure 3.1) explain the each process below.

- Extract Features from Images
- Let the user do Query
    - Query by Sketch
    - Query by Keywords
    - Query by Example (image)
        - Pick example images then ask the system to retrieve "similar" images.
- Refine the result by Relevance Feedback
    - User gives a feedback to the query results
    - System recalculates feature weights

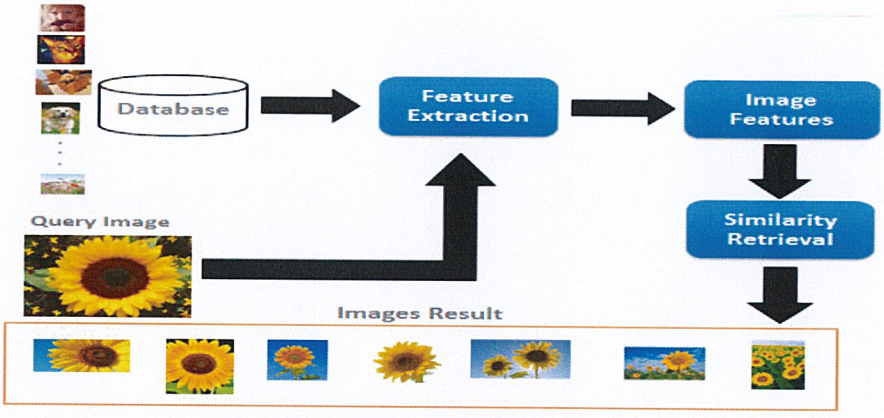


Figure 3. 1 CBIR procedure

3.1.3 Color Histogram

This was just a simple example how the histogram works and why it is useful. The histogram can keep count not only of color intensities but also whatever image features which we want to measure for example gradients, directions and so on.

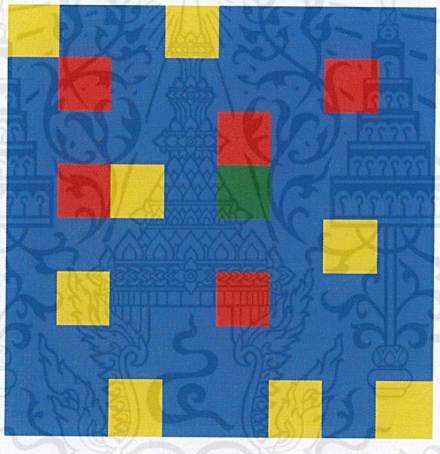


Figure 3.2 Image from camera

Assume that the image (Figure 3.2) is the image from the camera. The small piece of square is pixels that gather to the image. Separate the color and count amount of individual color as follow:

- Blue 50 pixels
- Yellow 8 pixels
- Red 5 pixels
- Green 1 pixel

Then arrange the information individual color to graph (Figure 3.3):

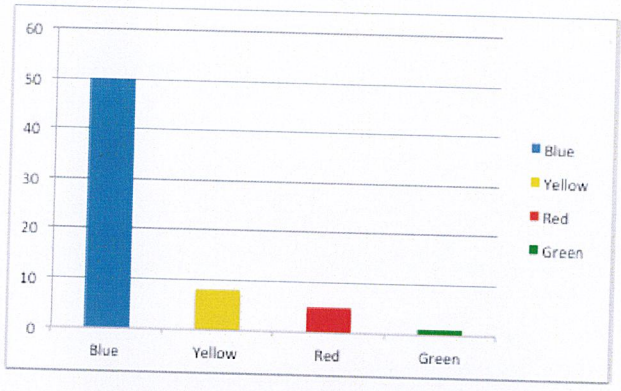


Figure 3. 3 Graph show amount of individual color pixel in Figure 3.2

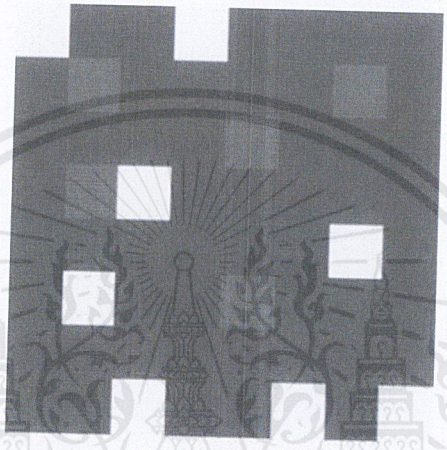


Figure 3. 4 Convert color image to grayscale

For easier understanding, adjust the image to gray scale (Figure 3.4). If construct the graph horizontal by brighten scale from 0 is the most dark to 255 is the most brighten (Figure 3.5). Arrange from left to right. Take each pixel to evaluate the brighten scale and then organize the same brighten value, lay down the same brighten value in the same place that change to bars graph in vertical.

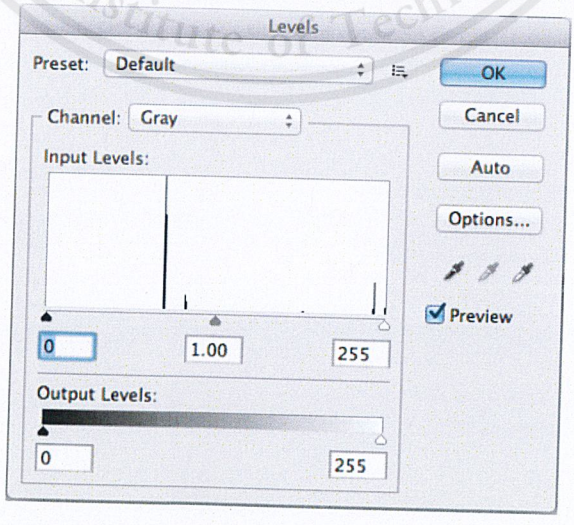


Figure 3.5 Level of grayscale in range 0 – 255

For more illustration, the example from original image (Figure 3.6) then adjust to gray scale color and mark from gray color to level color in each area shown in Figure 3.7.

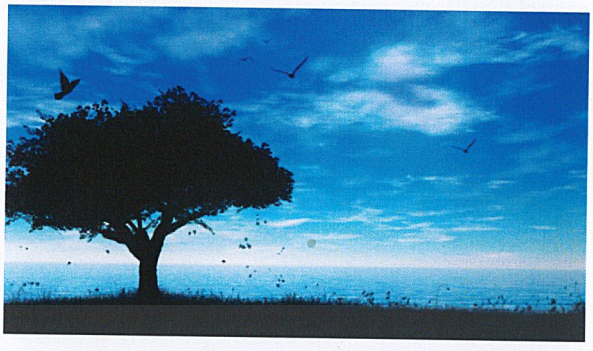


Figure 3.6 Example image for more illustration

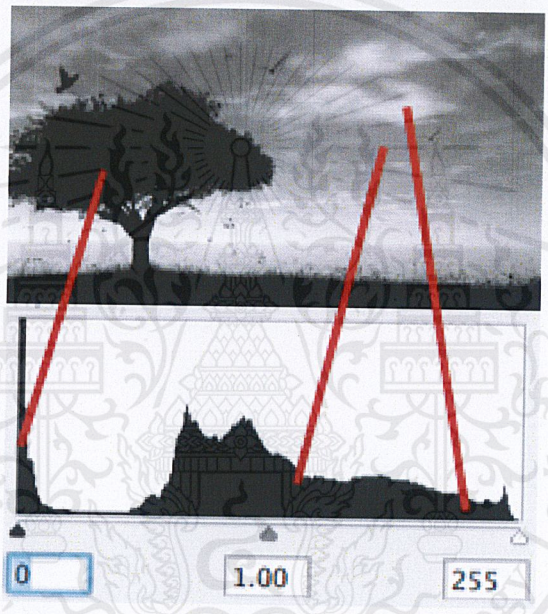


Figure 3.7 Level of grayscale from example image

3.1.2 HSV Color Model[1]

The HSV stands for the Hue, Saturation, and Value based on the artists (Tint, Shade, and Tone). The coordinate system describe in a hexacone in Figure 3.8. And Figure 3.9 view of the HSV color model. The Value represents intensity of a color, which is decoupled from the color information in the represented image. The hue and saturation components are intimately related to the way human eye perceives color resulting in image processing algorithms with physiological basis.

As hue varies from 0 to 1.0, the corresponding colors vary from red, through yellow, green, cyan, blue, and magenta, back to red, so that there are actually red values

[1] http://scien.stanford.edu/pages/labsite/2002/psych221/projects/02/sojeong/

both at 0 and 1.0. As saturation varies from 0 to 1.0, the corresponding colors (hues) vary from unsaturated (shades of gray) to fully saturated (no white component). As value, or brightness, varies from 0 to 1.0, the corresponding colors become increasingly brighter.

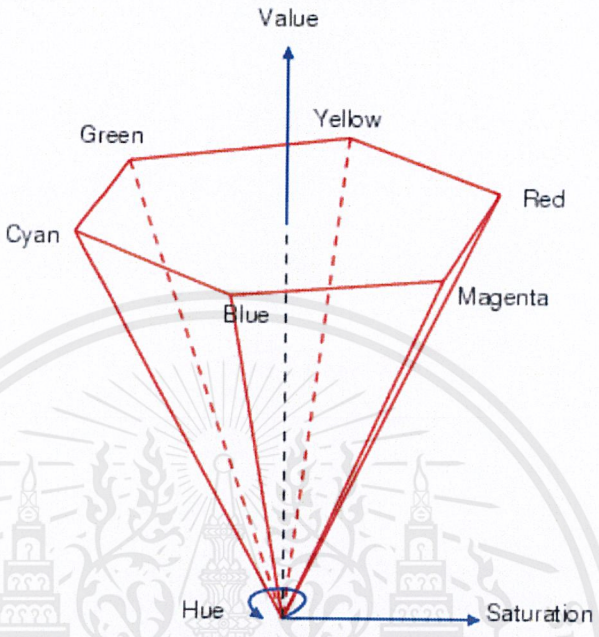


Figure 3. 8 HSV coordinates system

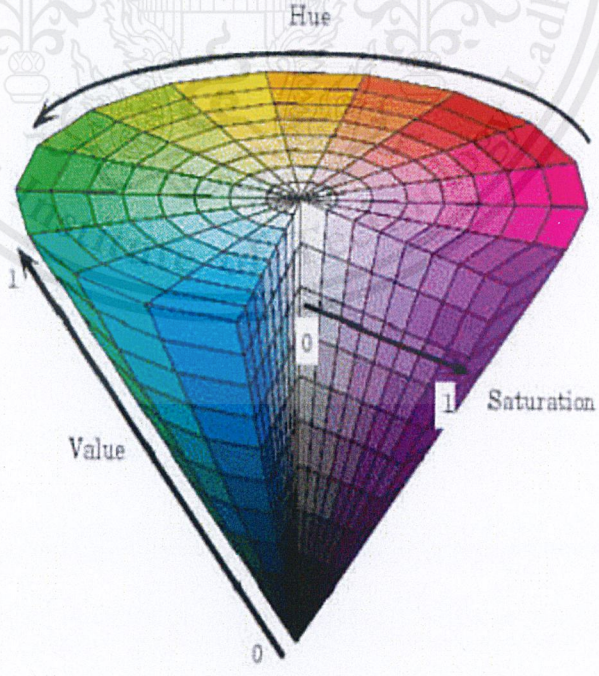


Figure 3. 9 HSV color model

3.2 Image categorization

The program categorize automatic image collection are trained by using set of images that are classify by hand. Actually, in CBIR searching method make use of classifier trained on image search results. In other words, in image search, object categorization is one component of the system. We applied searching system with categorization that first trains the system to be recognizing the specific visual information contained in images of each folder. Then system will know by themselves which folder for the new image will place, called semi-automatic system.

The principle to obtain the nearest object among many images suppose placed such this graph (Figure 3.10) closely with k-nearest neighbor. Categorization is done by applying basis of k-Nearest Neighbor algorithm. The k-NN (k-nearest neighbor) algorithm is a method for classifying objects based on closest training examples in the feature space. We chose color histogram for computing similarity of two images and also applied it to be used with classification by comparing histogram of each image and find the nearest value in each category compared with query image and assign the query image to that category as semi-automatic technique.

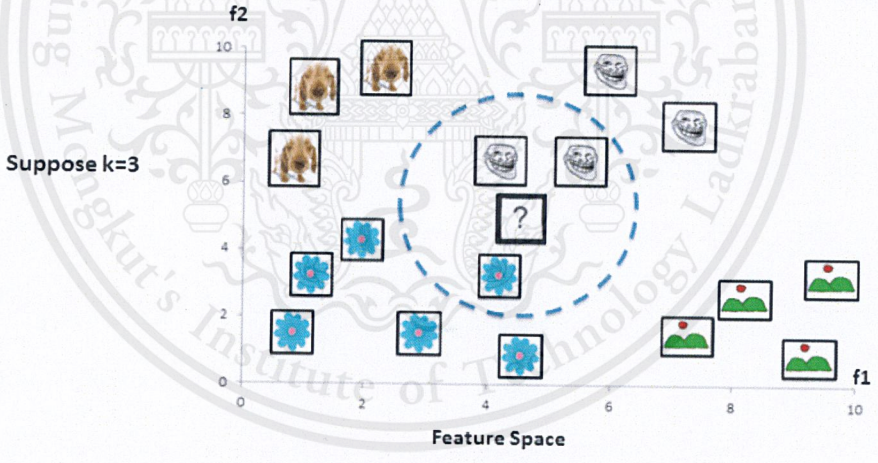


Figure 3. 10 Process of k-nearest neighbor method

Figure 3.10 is shows how k-NN algorithm works f1 and f2. Axis f1 and f2 is features space. Group of dog represent the animal category, troll face represent person category, blue flower represent the flower category and mountain represent place category. The question mark image is query image. We supposed k=3, where k is the number of nearest neighbor which near the query image. Two of troll face, which is object of person category is nearest, and followed by one of object of flower category. So, we can summarized that query image is object of person category.

3.3 Development tools

Software development tool is a program or application that software developers use to create, debug, maintain, or otherwise support other programs and applications[2]. A participation of many application or program that use to develop this software some of them we have to associate with each other to achieve the goal of this software project. Each software we have chosen it pretty much suitability and easy to apply for getting the solution. However, some of them we have never learnt and do not know how to use it at all, but we cannot denied those software because we need them and it is main of our software as well so, we need to study first and try to build the software under the functions of software tools in the right way.

We need the following tools to complete the project:

Table 3.1 Tools we need to develop this software

Tools	Detail
Microsoft Visual Studio 2010[3]	We develop this software by using Microsoft visual studio 2010, C++ language to implement and associated with QT creator to build GUI.
OpenCV(Open Source Computer Vision) [2]	Library of programming functions for real time computer vision. We need OpenCV to help us implement some function of image processing to achieve the software searching.
QTcreator[4]	Qt Creator is a cross-platform C++ integrated development environment which is part of the Qt SDK. It includes a visual debugger and an integrated GUI layout and forms designer.

2 Read more: http://en.wikipedia.org/wiki/Programming_tool

3 Read more: http://en.wikipedia.org/wiki/Microsoft_Visual_Studio

Microsoft Office Project[4]	Microsoft Project is a project management software program which is designed to assist a project manager in developing a plan, assigning resources to tasks, tracking progress, managing the budget, and analyzing workloads.
Visual Paradigm	*Visual Paradigm* is a visual modeling tool for all kinds of UML diagrams

Moreover, some necessary software program or application are involved and it was like assistants of us and behind the scenes to achieve the goal and we trusting to use it to back-up our files is Dropbox[5] application, the big cloud to keep all works and documents there to share and verify our work easily. Another software, it is important one to makes the report and all documents that is Microsoft Office Word 2010 and to build and make for good presentation we chose Microsoft Office Powerpoint 2010.

4 Read more: http://en.wikipedia.org/wiki/Microsoft_Project#cite_note-1

5 Dropbox is a free service that lets you bring your photos, docs, and videos anywhere and share them easily.

Chapter 4

Software overview

After we determined software specifications we have to plan the process for solving and planning problem of a software solution. For software analysis aim what the system will and it is captured in the functional requirements. And for software design, creates a detailed design that acts as the blue-print that will construct the code to create the system. The tool we use is UML (Unified Modeling Language)[6].

4.1 Software requirement specification

This software is an intelligent image database program for searching and managing images imported by users. This software can collects images by import images to the program for divided into category and performs searching by using a query image. Also can view image that user imported to the program ordered by imported date.

4.1.1 User requirement specification

4.1.1.1 Functional requirement

1. Import & Display
    - Import: Users can browse images in user's computer and can import selected images to the program.
    - Display: Users can view imported images, which are kept in separated folders depending on the date of importing.
2. Smart search
    - Search by image: Users can search for similar images by using a query image.

---

6 Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of object-oriented software engineering. The Unified Modeling Language includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems. The Unified Modeling Language was developed by Grady Booch, Ivar Jacobson and Jim Rumbaugh at Rational Software in the 1990s.[6]

- Sorting: The result of searching will be sorted based on their similarity.
- Quantity: Users can see the total number of similar images.
- Search in a category: Users can specify the category(s) to be searched.

3. Categorization
   - Category: Assign category to each imported image.
   - Semi-automatic: Categorization is a semi-automatic process. Users can teach the program to recognize the image categories.
   - Add category: Users can add a new category.

4.1.1.2 Non-functional requirement

1. Graphic user interface (GUI): The program is easy to understand and easy to use.
2. Performance: Can quickly respond for show the result of similar images.
3. Database: This software is local program that have a database to keep and search the images.
4. Accuracy: Program can calculate total of result and compare the image correctly.

4.1.2 System requirement specification

4.1.2.1 Functional requirement

1. Management
   - This program has system for managing images into database by separate folders depending on the date of importing and categorization based on type of image by using semi-automatic process.
   - This program can create more categories.
2. Image search
   - This program can search from a query image and can compare with all images in category(s) that specified by user.
   - If a query image matches with any image so, program will be show all result and program will sort by the most similarity.
   - Otherwise, will show text message "no result".
   - This program can calculate the total number of similar images.
3. Modification record

- When user imported desire images into database of this program, it will be record date created for makes folder of those images.
- If user would like to change the name of folder, the system will be update and change to date modified.
- Image's data in database always verify and up to date.

4.1.2.2 Non-functional requirement

1. Artificial intelligence: This program has to training from user for first and can learn by itself later on.
2. Correctly: This program can match images and manage images into category correctly.
3. Database: Managing to keep and retrieve images.
4. Performance: This program reduces time for searching even though it's contains a lot of data.

4.2 Software analysis and design

4.2.1 Use case diagram

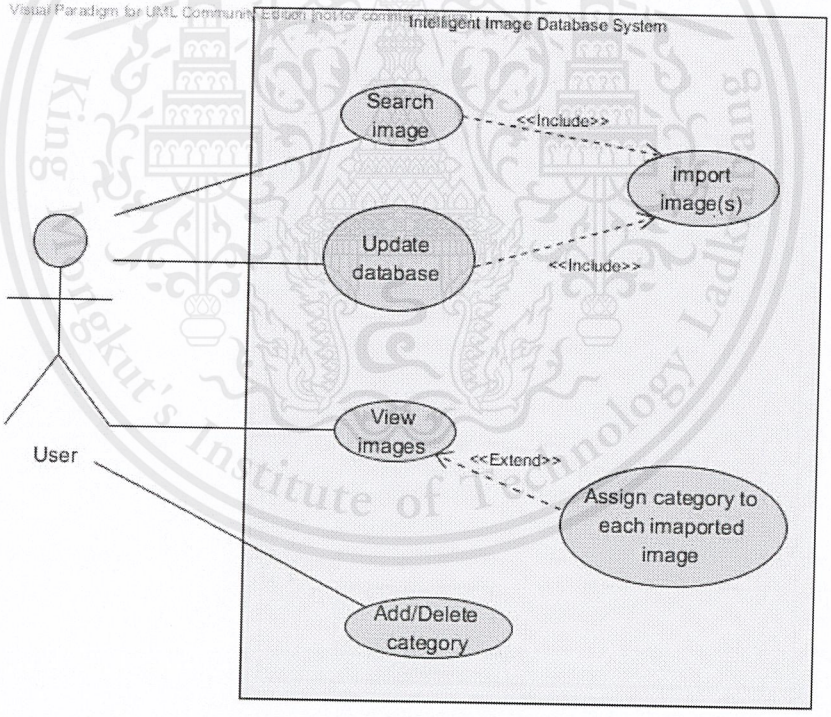


Figure 4. 1 Use case diagram

As show in Figure 4.1 it is overall of the software that show what users' required for this software and each function that interact with user and association between each use case within the system named *Intelligent Image Database System* The use case extract from user requirements. This use case diagram has six use case and one actor (User). The detail of each use case will be described as following.

Use case diagram description

Table 4. 1 Search image Use case diagram description

Use case	Search image
Actor	User
Purpose	For search similar image in the system.
Pre-condition	User wants to search similar image.
Post-condition	The system shows results of the similar images.
Process	1. On tab "Search". 2. Click browse button to select query image, and click open. 3. Chose the category, as user wants. 4. Click Search button. 5. The image has been process and show results of similar images.

Table 4. 2 Update database Use case diagram description

Use case	Update database (Import image to store in the system)
Actor	User
Purpose	For import image of user to store in the system.
Pre-condition	User wants to update database by import image(s) in to system.
Post-condition	All image(s) of user imported is stored in the system
Process	1. On tab "Browse". 2. Click import button to select image as user want to import. 3. Selected image(s) to import. 4. Click "Import All" or "Import Selected" button. 5. Import image complete.

Table 4. 3 Import image(s) Use case diagram description

Use case	Import image(s)
Actor	User
Purpose	If user wants to search image or update database, user must be import image first.
Pre-condition	User wants to search image or update database.
Post-condition	Imported.
Process	1. Want to search or update database. 2. Select image(s). 3. Imported.

Table 4. 4 View image Use case diagram description

Use case	View images
Actor	User
Purpose	For view image in the system that user imported.
Pre-condition	User wants to view images in the system that user imported.
Post-condition	Finish view images.
Process	1. On tab "Category". 2. Chose the folder to view. 3. View images. 4. Finish view images.

Table 4. 5 Assign category to each imported image Use case diagram description

Use case	Assign category to each imported image.
Actor	User
Purpose	For assign category to each imported image.
Pre-condition	User wants to assign category to each imported image.
Post-condition	Finish assign category to each imported image.
Process	1. On tab "Category". 2. Click folder "All images". 3. Assign category to each image. 4. Finish assign category to each image.

Table 4. 6 Add/Delete category Use case diagram description

Use case	Add/Delete category
Actor	User
Purpose	For add or delete category to assign image that user imported.
Pre-condition	User wants to add more categories or wants to delete existing category.
Post-condition	Add more categories or delete category done.
Process	1. On tab "Category". 2. Click "Add or Delete Category" button. 3. If click Add, fill in name of category and click ok. 4. If click delete, category will be delete from system and click ok. 5. Finishes add or delete category.

4.2.2 Activity diagram

Activity diagram describe the workflow of the software system that start from a starting point to the end point detailing the many decision paths that exist in the progression of each event contained in the activity step by step.

Search process shown in Figure 4.2 starts from open program until the process of search end.

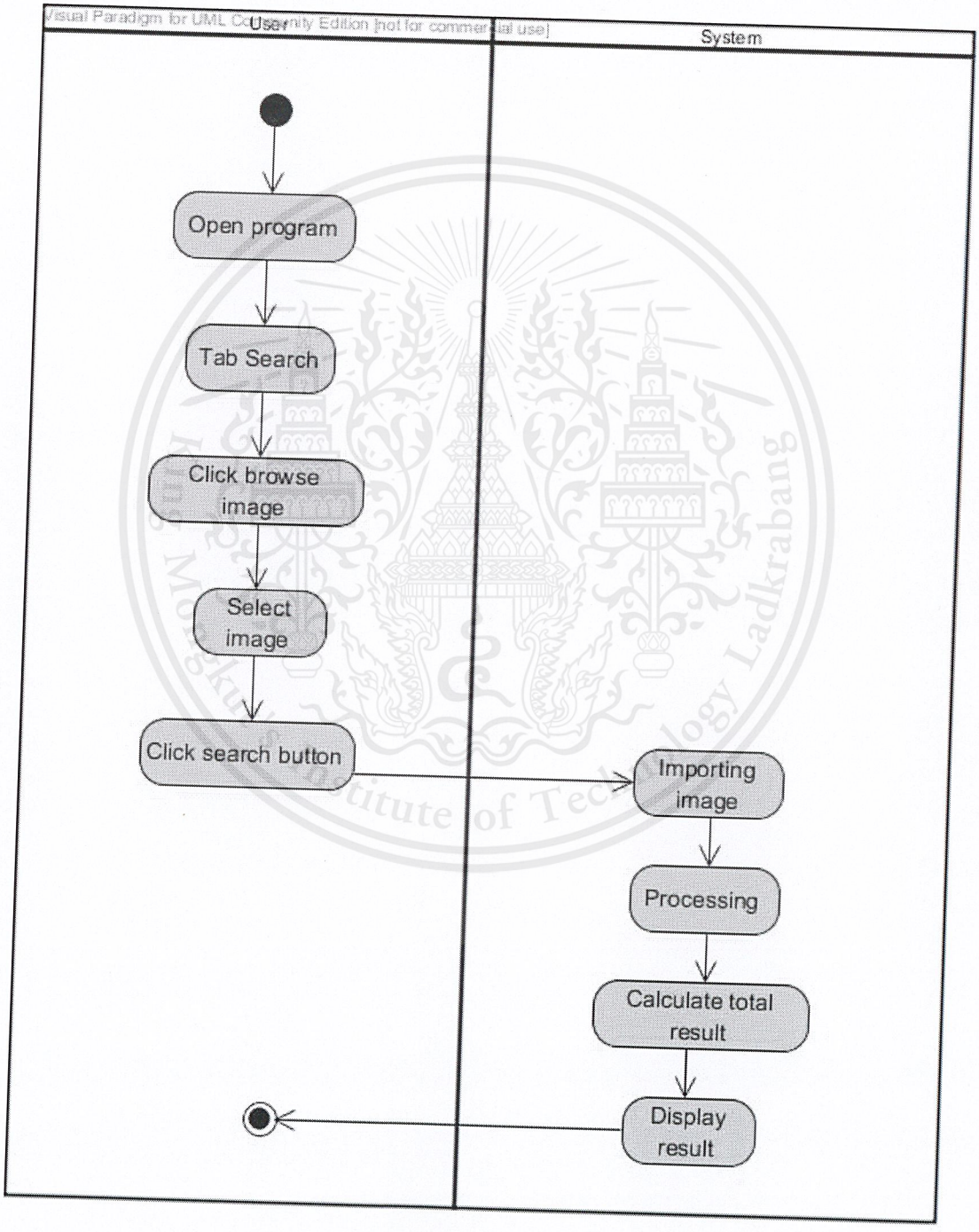


Figure 4. 2 Show procedure of searching in activity diagram

Imported image to the program and view all images show in Figure 4.3. Starting from open program to the end of process.

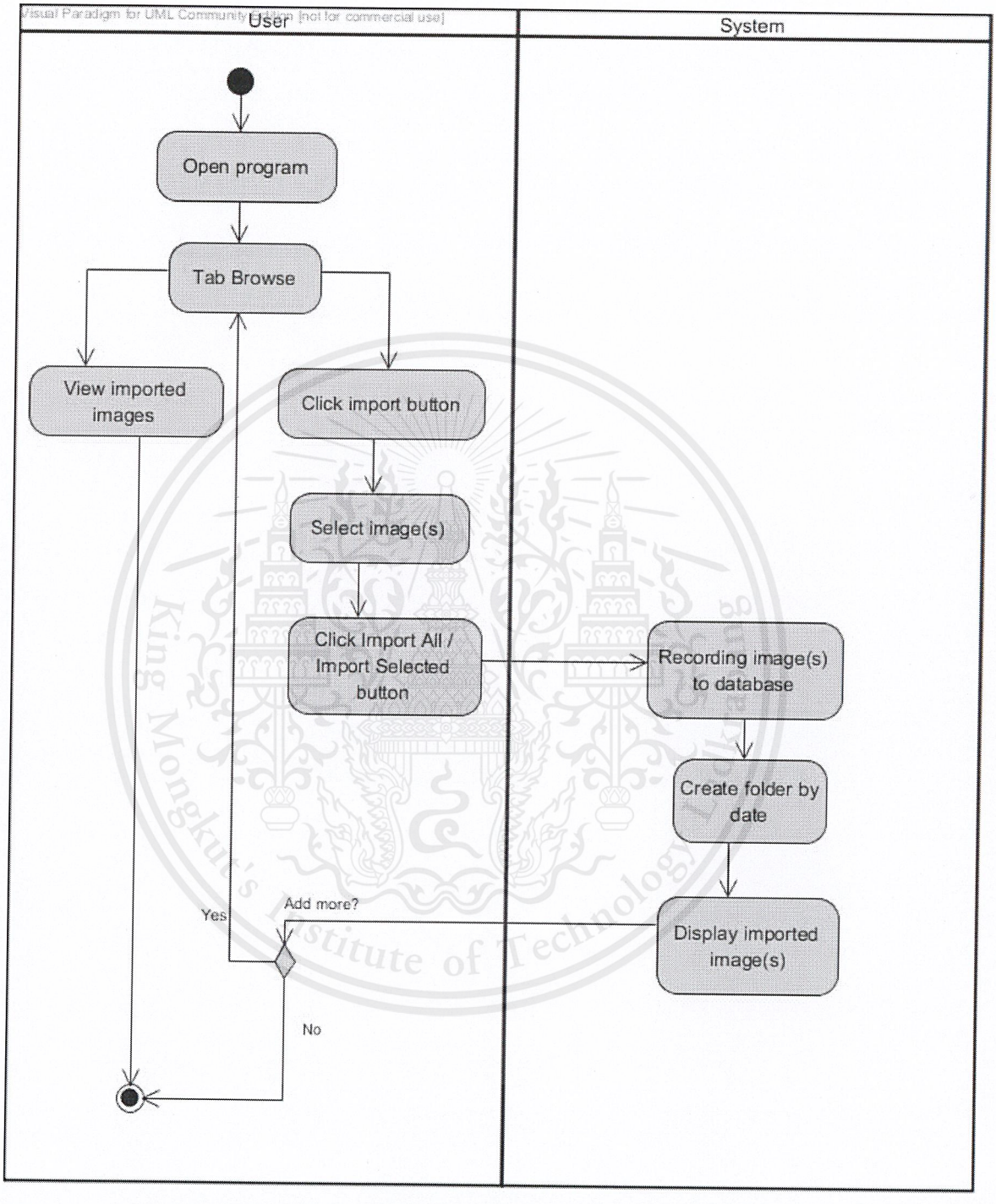


Figure 4. 3 Show procedure of importing in activity diagram

Categorization process shown in Figure 4.4 starts from open program until user assigns the image then end the program.

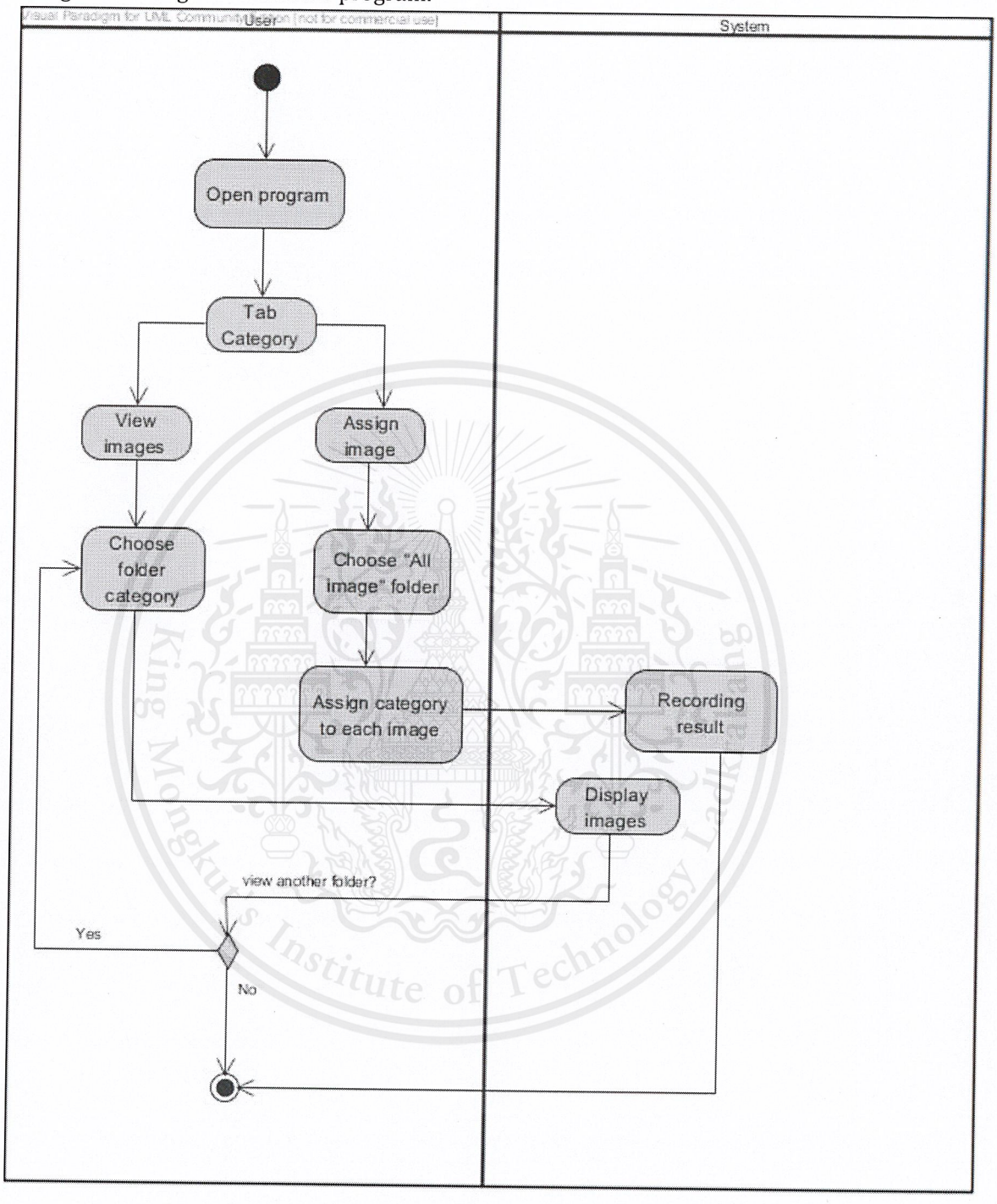


Figure 4. 4 Show procedure of categorization in activity diagram

Chapter 5

Software design

After we finished planned the requirements to get the software product, then we should design the software system by applying the requirements thoroughly to construct the structure diagram that are class diagram to describes the structure of software system present by system's classes, their attributes, methods and the relationships among the classes, also package diagram represents dependencies between the packages.

5.1 Class diagram

Class diagram could contain relationships between classes and relationships between classes' instances. Relationships between classes are represented by association, aggregation, dependency or composition.

According to Figure 4.5, in the diagram there are four boundary class diagrams, two control diagrams and tree entity diagrams. Four boundary class diagram represent the user interface that main window is consist of search, browse and category tab. Two control class diagrams represent the algorithm of searching and categorization, categorization dependency on searching because of searching need to use data of images in categorization. Tree entity class diagrams represent the data in user's computer and database in system compose of image.

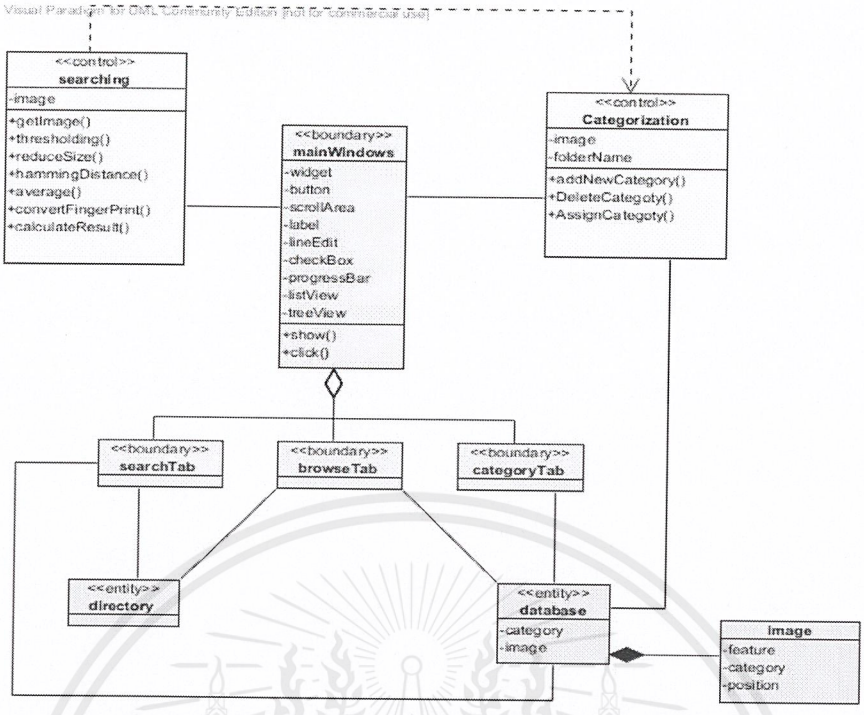


Figure 5.1 Relationship between each class and specified attributes and operations

Class diagram description

Table 5.1mainWindow Class diagram description

Class Name	mainWindow <<boundary>>
Attributes	-widget -button -scrollArea -label -lineEdit -checkBox -progressBar -listView -treeView
Operations	+show() +click()
Consist	Search tab, browse tab and category tab.
Descriptions	This class is user interface to describe component of the program.

Table 5.2 searching Class diagram description

Class Name	searching <<control>>
Attributes	-image
Operations	+getImage() +thresholding() +reduceSize() +hammingDisance() +average() +convetFingerPrint() +calculateResult()
Descriptions	This class is the process of searching in the system.

Table 5.3 Categorization Class diagram description

Class Name	Categorization <<control>>
Attributes	-image -folderName
Operations	+addNewCategory() +deleteCategory() +assignCategory()
Descriptions	This class shows the part of categorization can do such as add and delete category.

Table 5.4 Directory Class diagram description

Class Name	Directory <<entity>>
Attributes	-
Operations	-
Descriptions	This class is directory of user computer that connect with tab browse when user want to import image from user's computer to system.

Table 5.5 Database Class diagram description

Class Name	Database <<entity>>
Attributes	-category -image
Operations	-
Descriptions	This class is the database of the system to store image and determine category of image. Database consist of image class is the main important of this program must use.

Table 5.6 Image Class diagram description

Class Name	Image <<entity>>
Attributes	-feature -category -position
Operations	-
Descriptions	This is important class of this system to search, in each image must have identification such as feature and position. Position mean that the position of graph of K-NN algorithm to keep information of image for easy to search.

5.2 Package diagram

"A package diagram in the Unified Modeling Language depicts the dependencies between the packages that make up a mode and can use packages containing use cases to illustrate the functionality of a software system." we designed Package diagram for larger-scale systems of software structure for indicated dependencies between major elements of a system and easier to analyze subsystem and represent how it communicate within each package. Constructing diagram of packages and dependencies helps us to keep an application's dependencies under control.

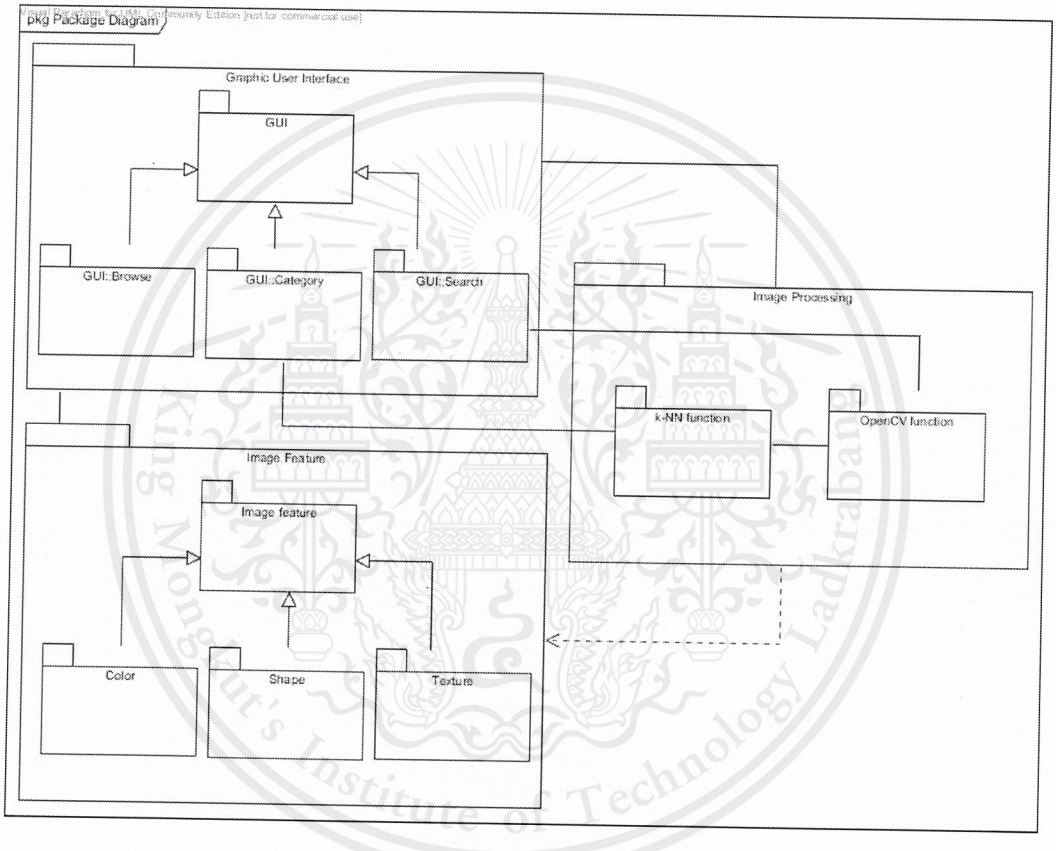


Figure 5.2 Package diagram

5.2.1 Sub system Graphic User Interface

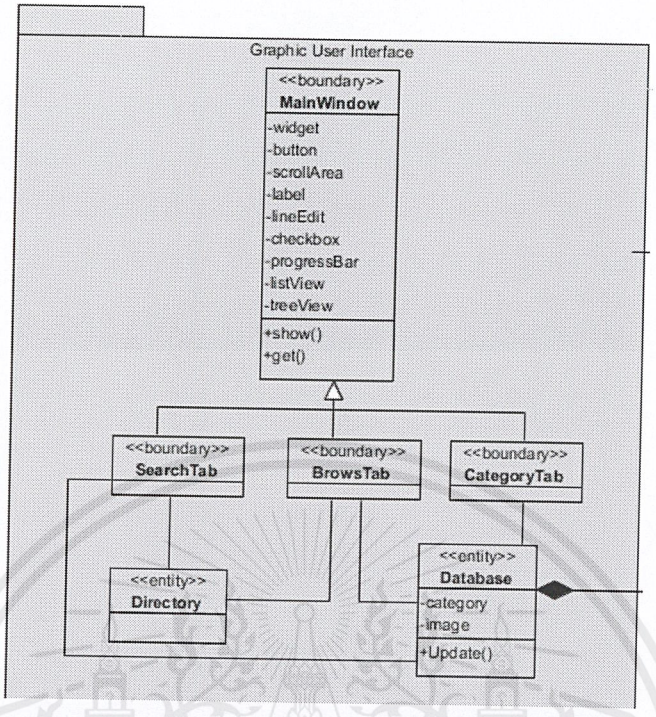


Figure 5.3 Sub package of Graphic User Interface

5.2.2 Sub system Image feature

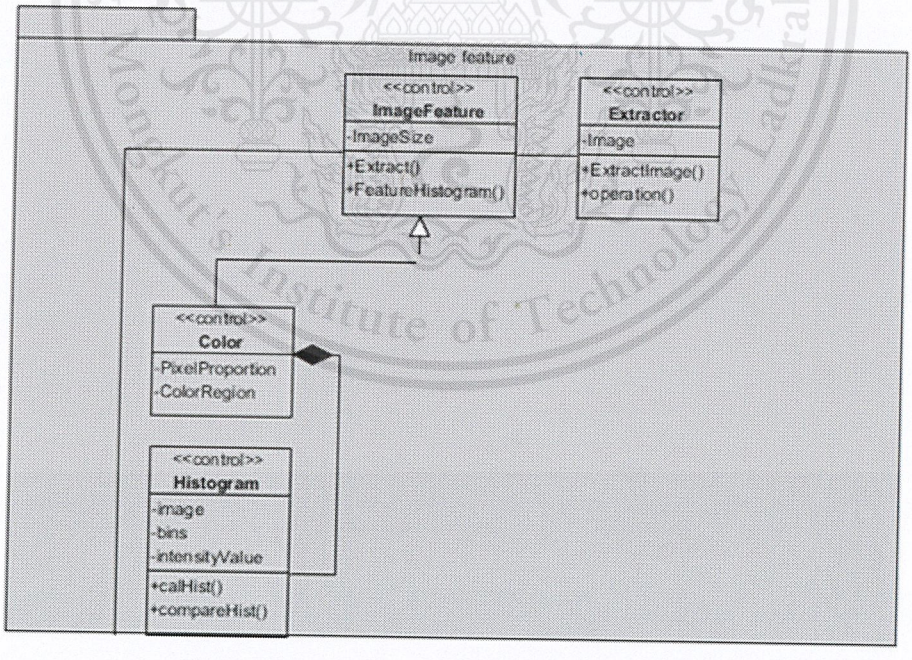


Figure 5.4 Sub package of Image feature

5.2.3 Sub system Image processing

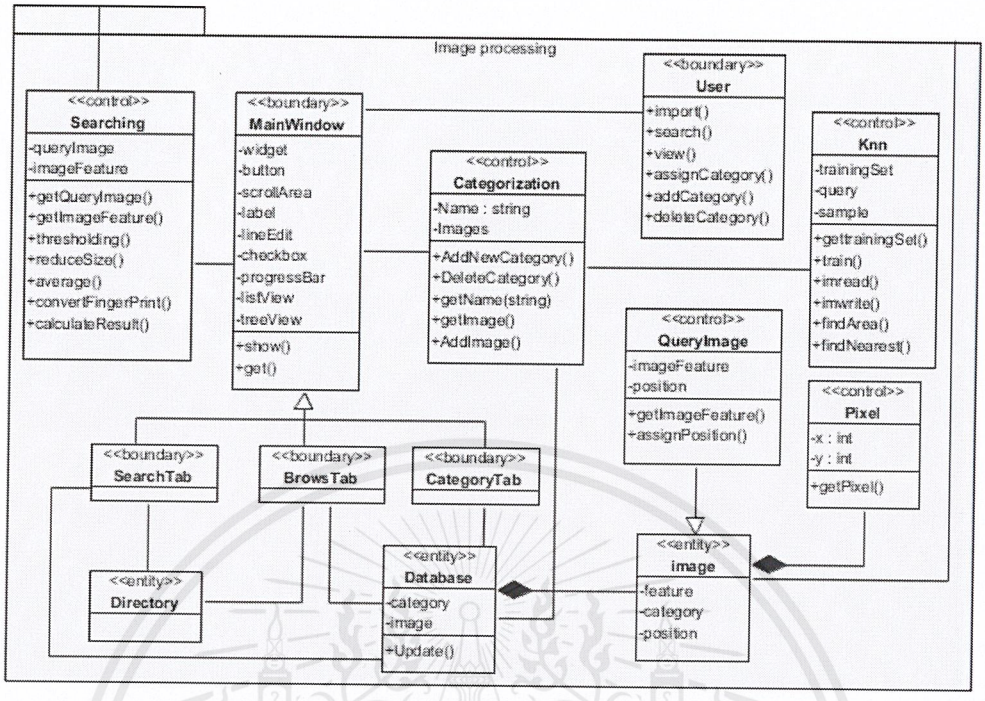


Figure 5.5 Sub package of Image processing

Chapter 6

Evaluations and discussions

6.1 Graphic user interface

6.1.1 Smart search

Search by image: Users can search for similar images by using a query image.And can specify the category(s) to be searched. Open program the users will see the program look like Figure 6. 1.

Smart search, if the users want to search the similar image by using image, users click browse button to browse the image in user's computer as you can see in Figure 6.2, select image and upload. Users can specify the category to be search. And click search button.

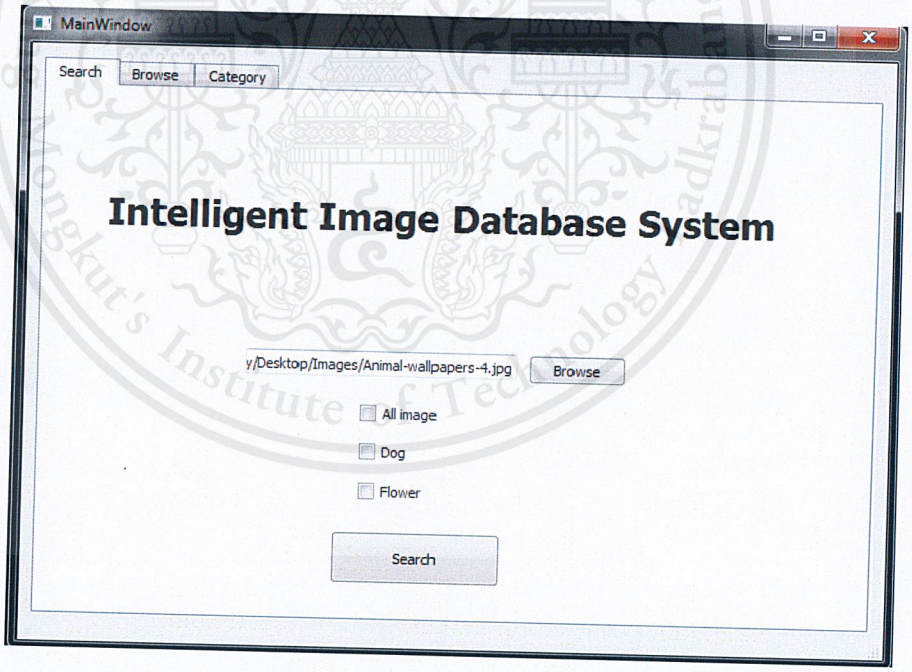


Figure 6. 1 Home screen

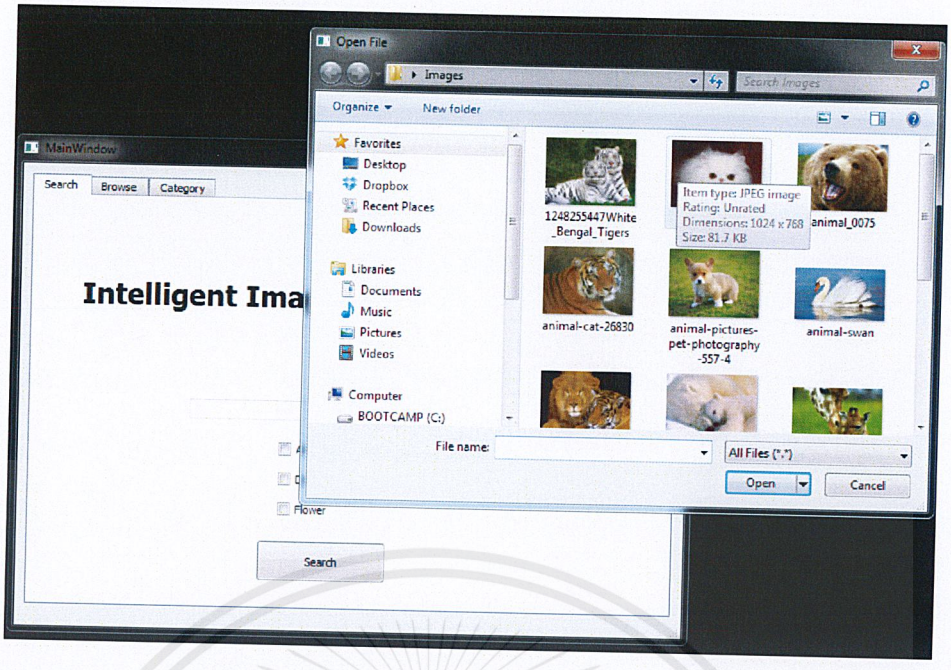


Figure 6. 2 Browse directory in user's computer

6.1.2 Result ordering

Sorting: The result of searching will be sorted based on their similarity show in Figure 6.3.

Quantity: Users can see the total number of similar images. The result will look like this and the users can view full image by click each result image.

Figure 6. 3 Searching results

6.1.3 Import and display function

Import: Users can browse images in user's computer and can import selected images to the program.

Display: Users can view imported images, which are kept in separated folders depending on the date of importing.

Browse and Import, when the users click browse tab the program show the history of import image separated by date of import(Figure 6.4). Import image to the program by click import button, window browse will show for select images after selected images then click import button(Figure 6.5).

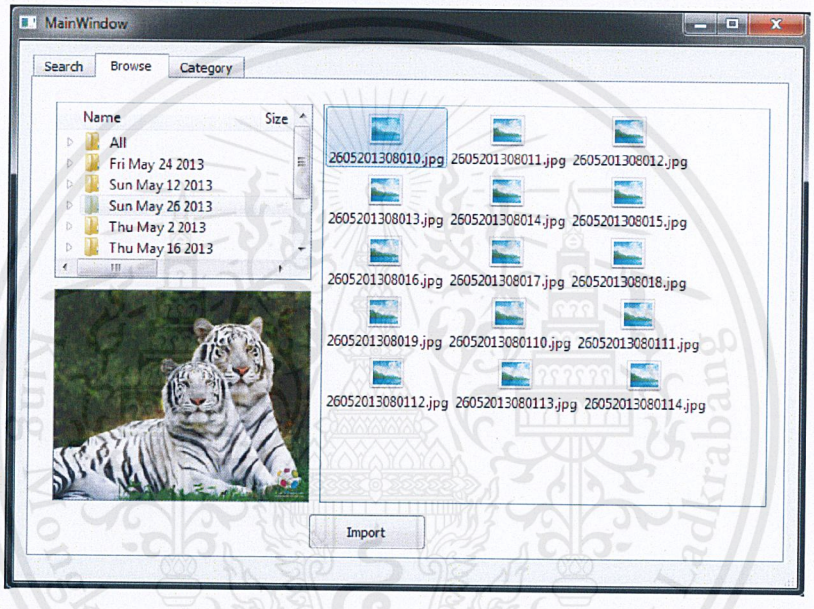


Figure 6. 4 View imported images seperated by date importing

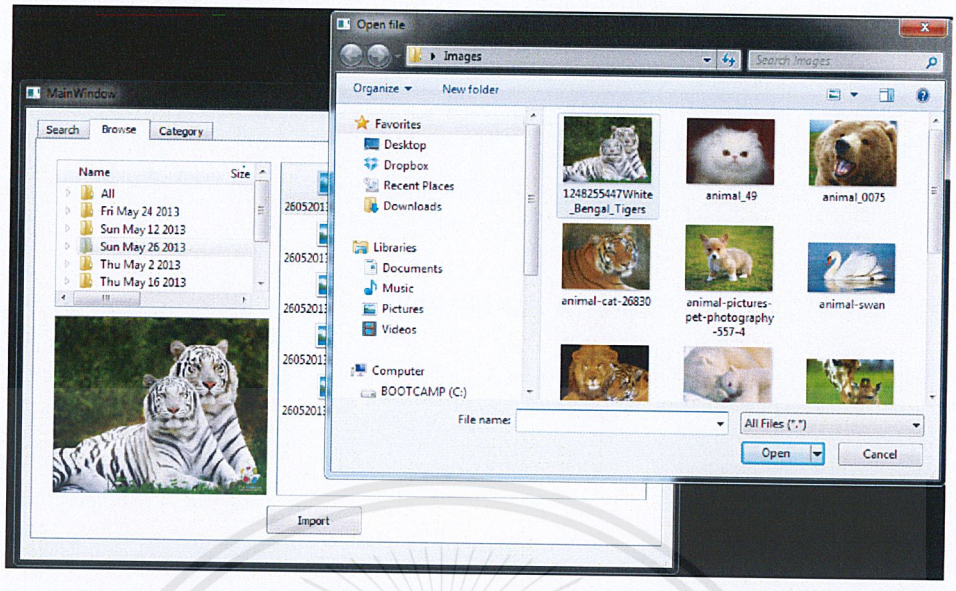


Figure 6. 5 Import images into program's database

6.1.4 Categorization

Semi-automatic: Categorization is a semi-automatic process. Users can teach the program to recognize the image categories

Category: Assign category to each imported image after the users imported the image all image will record in all images folder. Users can teach the program to recognize the image categories.

After user assign the category of image then image will classify in each folder (Figure 6.6). And the correct symbol represent when the users teach the program by categorization the images in a period of time. Then the program can recognize the image categories (Figure 6.7).

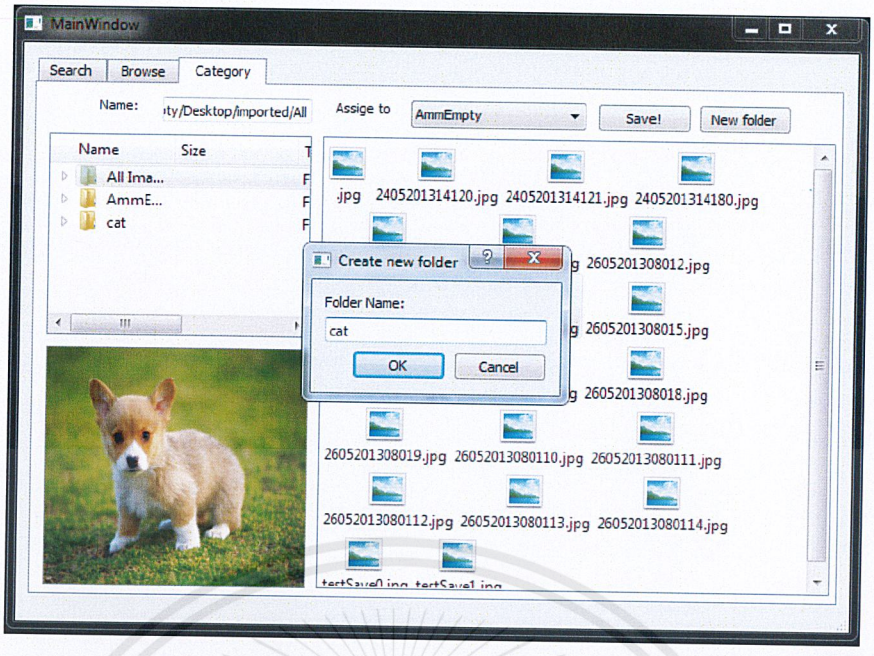


Figure 6. 6 Show all folder(s) generated by user(imported images)

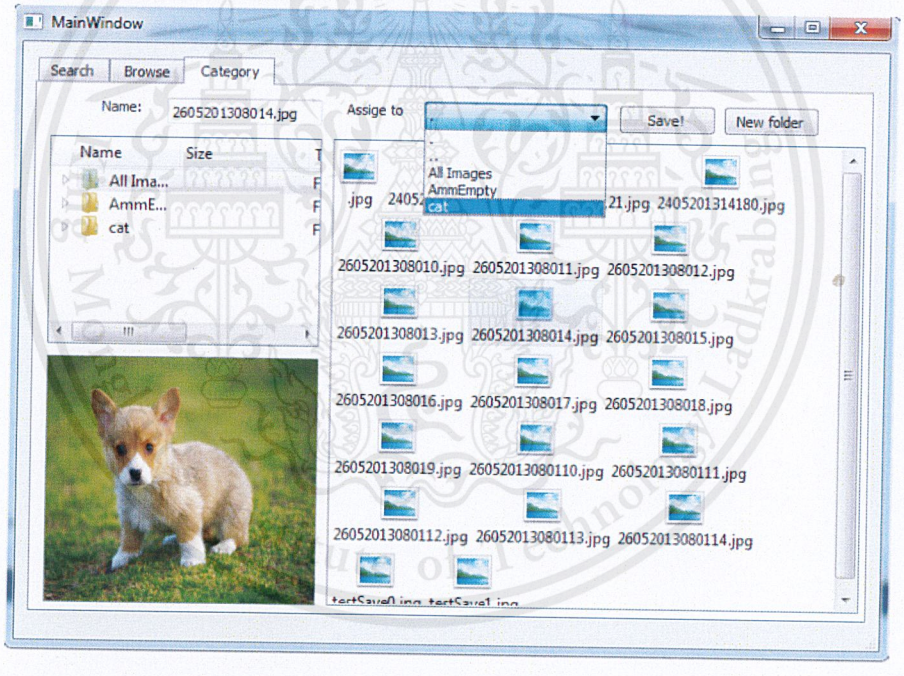


Figure 6. 7 Assign category(s) for each image

6.2 Experimentations

The experiment is to be tested and estimated the percentage of efficiency and discussed each experiment that we tested how optimal they are.

An experiment has been done to compare the accuracy of the two methods: perception of hash algorithm and color histogram comparison method. We use 20 images from 5 different groups. Note that we divide the images into groups based on their color. The images used in the experiment are shown in Figs. 6.8 – 6.12. In the

experiment, each image will be used as a query image. Then searching will be performed. If the top-four result includes an image from the group of the query image, we will say that the search is successful. The number of successful search of the two methods will be counted and compared. The result of perception of hash algorithm is 12 of 20 that are 60%. And the result of color histogram comparison is 17 of 20 that are 85%. The percentage of the result shown color histogram comparison is better than perception of hash algorithm.

Figure 6. 8 Images for experiment in group 1.

Figure 6. 9 Images for experiment in group 2.

Figure 6. 10 Images for experiment in group 3.

Figure 6. 11 Images for experiment in group 4.

Figure 6. 12 Images for experiment in group 5.

Chapter 7

Conclusions

Intelligent image database system is the system for searching and management images by using CBIR techniques is fundamental which searching by using visual information contained in the image itself to obtain more accuracy than texting some word that associated with those images. Another feature in this program is categorization images that contained in local computer.

We applied color feature for this program for computing distance measures based on color similarity which is achieved by computing a color histogram for each image. For checking images using the colors that they contain is one of the most widely used techniques because it does not depend on image size or orientation but in other hands, they also are weak point for measurement similar image.

7.1 Conclusion and future work

We have presented the design, implementation, of Intelligent Image Database System as local, it can be improve database of this software to stay on network like a Drop box and apply it as online searching. Providing to collect the personal images by login in to system on web site that user can search, import and view image of user as the local system. And can share to another people to see image by folder as user want.

Challenging to track the place where are users taken photo and share to social network and improve the performance and add more feature(s) to easier, efficiency, precisely and faster to save the time to search desire images and support more platform such as on Android, iOS, windows mobile and so on.

Bibliography

[1] [OpenCV1] Gary Bradski and Adrian Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library (1st Edition), O'Reilly Media, 2008

[2] [OpenCV2] Robert Lganière, OpenCV 2 Computer Vision Application Programming Cookbook, Packt Publishing, 2011

[3] [ImageProcessing] Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing (3rd Edition), Prentice Hall, 2007.

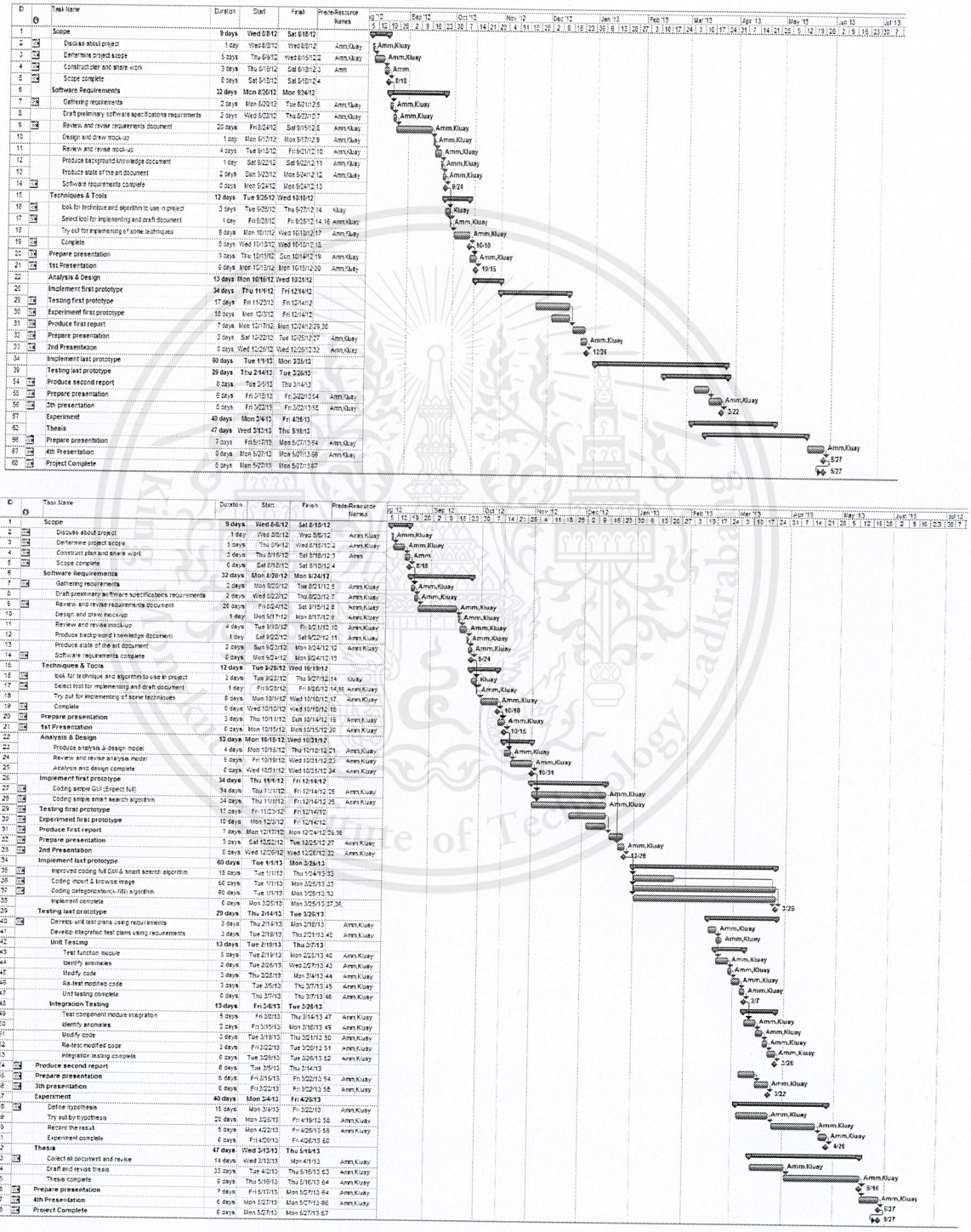
[4] [Qt] Jasmin Blanchette and Mark Summerfield, C++ GUI Programming with Qt 4 (2nd Edition), Prentice Hall, 2008

[5] Neal, Krawetz, "Perception of the hash algorithm implementation" http://www.hackerfactor.com/blog/?/archives/432-Looks-Like-It.html. Last access on May 28th, 2012 at 6:11 am „m

Appendix A

How we planned to achieve our project

Gantt chart full version



ID	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	Scope	9 days	Wed 8/8/12	Sat 8/18/12		
2	Discuss about project	1 day	Wed 8/8/12	Wed 8/8/12		Amm,Kluay
3	Determine project scope	5 days	Thu 8/9/12	Wed 8/15/12	2	Amm,Kluay
4	Construct plan and share work	3 days	Thu 8/16/12	Sat 8/18/12	3	Amm
5	Scope complete	0 days	Sat 8/18/12	Sat 8/18/12	4	
6	Software Requirements	32 days	Mon 8/20/12	Mon 9/24/12		
7	Gathering requirements	2 days	Mon 8/20/12	Tue 8/21/12	5	Amm,Kluay
8	Draft preliminary software specifications requirements	2 days	Wed 8/22/12	Thu 8/23/12	7	Amm,Kluay
9	Review and revise requirements document	20 days	Fri 8/24/12	Sat 9/15/12	8	Amm,Kluay
10	Design and draw mock-up	1 day	Mon 9/17/12	Mon 9/17/12	9	Amm,Kluay
11	Review and revise mock-up	4 days	Tue 9/18/12	Fri 9/21/12	10	Amm,Kluay
12	Produce background knowledge document	1 day	Sat 9/22/12	Sat 9/22/12	11	Amm,Kluay
13	Produce state of the art document	2 days	Sun 9/23/12	Mon 9/24/12	12	Amm,Kluay
14	Software requirements complete	0 days	Mon 9/24/12	Mon 9/24/12	13	
15	Techniques & Tools	12 days	Tue 9/25/12	Wed 10/10/12		
16	look for technique and algorithm to use in project	3 days	Tue 9/25/12	Thu 9/27/12	14	Kluay
17	Select tool for implementing and draft document	1 day	Fri 9/28/12	Fri 9/28/12	14,16	Amm,Kluay
18	Try out for implementing of some techniques	8 days	Mon 10/1/12	Wed 10/10/12	17	Amm,Kluay
19	Complete	0 days	Wed 10/10/12	Wed 10/10/12	18	
20	Prepare presentation	3 days	Thu 10/11/12	Sun 10/14/12	19	Amm,Kluay
21	1st Presentation	0 days	Mon 10/15/12	Mon 10/15/12	20	Amm,Kluay
22	Analysis & Design	13 days	Mon 10/15/12	Wed 10/31/12		
26	Implement first prototype	34 days	Thu 11/1/12	Fri 12/14/12		
27	Testing first prototype	17 days	Fri 11/23/12	Fri 12/14/12		
30	Experiment first prototype	15 days	Mon 12/3/12	Fri 12/14/12		
31	Produce first report	7 days	Mon 12/17/12	Mon 12/24/12	29,30	
32	Prepare presentation	3 days	Sat 12/22/12	Tue 12/25/12	27	Amm,Kluay
33	2nd Presentation	0 days	Wed 12/26/12	Wed 12/26/12	32	Amm,Kluay
34	Implement last prototype	90 days	Tue 1/1/13	Mon 3/25/13		
38	Testing last prototype	29 days	Thu 2/14/13	Tue 3/26/13		
54	Produce second report	6 days	Tue 3/5/13	Thu 3/14/13		
55	Prepare presentation	6 days	Fri 3/15/13	Fri 3/22/13	54	Amm,Kluay
56	3th presentation	0 days	Fri 3/22/13	Fri 3/22/13	55	Amm,Kluay
57	Experiment	40 days	Mon 3/4/13	Fri 4/26/13		
62	Thesis	47 days	Wed 3/13/13	Thu 5/16/13		
66	Prepare presentation	7 days	Fri 5/17/13	Mon 5/27/13	64	Amm,Kluay
67	4th Presentation	0 days	Mon 5/27/13	Mon 5/27/13	66	Amm,Kluay
68	Project Complete	0 days	Mon 5/27/13	Mon 5/27/13	67	

Figure A.1 planned to complete the project describe by Gantt chart

Appendix B

Source code

The perception of hash algorithm

```
#include <iostream>
#include<conio.h>
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <vector>

using namespace std;

float blockAverage(cv::Mat in,int x,int y,int blockSizeRow,int
blockSizeCol){
	float sum=0;
	for(int i=0; i<x+blockSizeRow; i++)
		for(int j=0; j<y+blockSizeCol; j++)
			sum += in.at<uchar>(i,j);
		sum /= (blockSizeRow+blockSizeCol);
	return sum;
}

void reduceTo8x8(cv::Mat in, cv::Mat out){
	int blockSizeRow = in.rows/out.rows;
	int blockSizeCol = in.cols/out.cols;
	for(int r=0, x=0; r<out.rows; r++,x += blockSizeRow)
		for(int c=0, y=0; c<out.cols; c++, y += blockSizeCol)
			out.at<uchar>(r,c) =
(uchar)blockAverage(in,x,y,blockSizeRow,blockSizeCol);
}

void quantizeTo64(cv::Mat in,cv::Mat out){
	for (int i=0;i<in.rows;i++)
		for(int j=0;j<in.cols;j++)
			out.at<uchar>(i,j)= in.at<uchar>(i,j)/4;
}
float findGlobalMean(cv::Mat in){
	float sum =0;
	for (int i=0;i<in.rows;i++)
		for(int j=0;j<in.cols;j++)
			sum += in.at<uchar>(i,j);
	sum = sum/(in.rows*in.cols);
	return sum;
}
void threSholding(cv::Mat in, cv::Mat out, const float th){
	for (int i=0;i<in.rows;i++)
		for(int j=0;j<in.cols;j++)
			if(in.at<uchar>(i,j)>= th)
				out.at<uchar>(i,j) = 1;
			else
				out.at<uchar>(i,j) = 0;
}
void convertToFingerPrint(cv::Mat in, vector<int>& fp ){
	for (int i=0;i<in.rows;i++)
		for(int j=0;j<in.cols;j++) {
			fp.push_back( in.at<uchar>(i,j) );
		}
}
```

```
void displayFingerprint( vector<int> fp ) {
	cout << endl;
	cout << "size of fp = " << fp.size() << endl;
	for(int i=0;i<fp.size();i++)
		cout << fp[i] ;
	cout << endl;
}

int calHammingDist( vector<int> fp1, vector<int> fp2 ){
	int dist = 0;
	for(int i=0;i<fp1.size();i++)
		if( fp1[i] != fp2[i] )
			dist++;
	return dist;
}

vector<int> imageToFingerprint(cv::Mat in) {
	//step 1: reduce to 8 by 8
	cv::Mat _8by8Image1( 8, 8, in.type() );
	cv::Mat _8by8Image2 = _8by8Image1.clone( );
	reduceTo8x8( in, _8by8Image1 );

	//step2: quantize the color to 64 different levels
	quantizeTo64( _8by8Image1, _8by8Image2);
	/*cv::Mat out = _8by8Image2.clone();
	cv::imwrite("out1.png", out);*/

	//step 3: find global mean of intensity
	float th = findGlobalMean(_8by8Image2);
	threSholding(_8by8Image2, _8by8Image1, th );
	//step 4: convert thresholded image to fingerprint
	vector<int> fp;
	convertToFingerPrint( _8by8Image1,fp);

	return fp;
}

int main(){
	//first image
	cv::Mat image1 = cv:: imread("flowe4.jpg", 0);
	vector<int> fp1 = imageToFingerprint( image1 );

	//second image
	cv::Mat image2 = cv:: imread("car3.jpg", 0);
	vector<int> fp2 = imageToFingerprint( image2 );

	//compare two fingerprints
	cout << "The distance between two images is " << calHammingDist( fp1,
fp2 ) << endl;

	getch();
	return 1;

}
```

GUI with searching and categorization

Main.cpp

```
#include <QtGui/QApplication>
#include "mainwindow.h"
#include <stdio.h>
#include <conio.h>
#include <iostream>
#include <fstream>
#include <string>
#include <windows.h>
#include <direct.h>
#include <queue>
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"

using namespace std;
using namespace cv;

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();

    return a.exec();
}
```

Mainwindow.cpp

```
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "qfiledialog.h"
#include "qpixmap.h"
#include "qdialog.h"
#include "qtabwidget.h"
#include "qlabel.h"
#include <QtCore>
#include <QtGui>
#include "qmessagebox.h"
#include "QMessageBox"
#include "qprogressbar.h"
#include <vector>
#include <string>
#include "qvector.h"
#include "qfileinfo.h"
#include "qdir.h"
#include "qstring.h"
#include "qstringlist.h"
#include "qvector.h"
#include "searchImg.h"
#include "qcombobox.h"
```

```
using namespace std;
using namespace cv;
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->widgetSearch->hide();
    //ui->widgetBrowse->hide();
      ////////////////////////////////Browse Tab//////////////////////////
    //create directory model
      QString queryName;
    QString Path = "C:/Users/AmmEmpty/Desktop/imported/";
    dirmodel = new QFileSystemModel(this);
    dirmodel->setFilter(QDir::NoDotAndDotDot | QDir::AllDirs);
    dirmodel->setRootPath(Path);
    ui->tb_treeView->setModel(dirmodel);
      QModelIndex idx = dirmodel-
>index("C:/Users/AmmEmpty/Desktop/imported/");
      ui->tb_treeView->setRootIndex(idx);
    //create file model
    filemodel = new QFileSystemModel(this);
    filemodel->setFilter(QDir::NoDotAndDotDot | QDir::Files);
    filemodel->setRootPath(Path);
    //set list view
      ui->tb_listView->setViewMode(QListView::IconMode);
    ui->tb_listView->setIconSize(QSize(200,200));
    ui->tb_listView->setModel(filemodel);
      ui->tb_listView->setRootIndex(idx);
    imageLabelBrowse = new QLabel;
    imageLabelBrowse->setBackgroundRole(QPalette::Base);
```

```
    imageLabelBrowse->setSizePolicy(QSizePolicy::Ignored,
QSizePolicy::Ignored);

    imageLabelBrowse->setScaledContents(true);

      ////////////////////////////////////////////////////////////////////

      ////////////////////Category////////////////////////////////

      ////create directory model

    QString PathCatTab = "C:/Users/AmmEmpty/Desktop/imported/All/";

    dirmodelCat = new QFileSystemModel(this);

    dirmodelCat->setFilter(QDir::NoDotAndDotDot | QDir::AllDirs);

    dirmodelCat->setRootPath(PathCatTab);

    ui->tc_treeView->setModel(dirmodelCat);

      QModelIndex idx2 = dirmodelCat-
>index("C:/Users/AmmEmpty/Desktop/imported/All/");

      ui->tc_treeView->setRootIndex(idx2);

    //create file model

    filemodelCat = new QFileSystemModel(this);

    filemodelCat->setFilter(QDir::NoDotAndDotDot | QDir::Files);

    filemodelCat->setRootPath(PathCatTab);

    //set list view

      ui->tc_listView->setViewMode(QListView::IconMode);

    ui->tc_listView->setIconSize(QSize(200,200));

    ui->tc_listView->setModel(filemodelCat);

      ui->tc_listView->setRootIndex(idx2);

imageLabelCat = new QLabel;

    imageLabelCat->setBackgroundRole(QPalette::Base);

    imageLabelCat->setSizePolicy(QSizePolicy::Ignored,
QSizePolicy::Ignored);

    imageLabelCat->setScaledContents(true);


      ////////////////////////////////////////////////////////////

      //create imageLabel

    imageLabel = new QLabel;

    imageLabel->setBackgroundRole(QPalette::Base);

    imageLabel->setSizePolicy(QSizePolicy::Ignored, QSizePolicy::Ignored);
```

```
imageLabel->setScaledContents(true);
imageResult = new QLabel;
imageResult->setBackgroundRole(QPalette::Base);
imageResult->setSizePolicy(QSizePolicy::Ignored, QSizePolicy::Ignored);
imageResult->setScaledContents(true);
imageResult2 = new QLabel;
imageResult2->setBackgroundRole(QPalette::Base);
imageResult2->setSizePolicy(QSizePolicy::Ignored, QSizePolicy::Ignored);
imageResult2->setScaledContents(true);
imageResult3 = new QLabel;
imageResult3->setBackgroundRole(QPalette::Base);
imageResult3->setSizePolicy(QSizePolicy::Ignored, QSizePolicy::Ignored);
imageResult3->setScaledContents(true);
imageResult4 = new QLabel;
imageResult4->setBackgroundRole(QPalette::Base);
imageResult4->setSizePolicy(QSizePolicy::Ignored, QSizePolicy::Ignored);
imageResult4->setScaledContents(true);
imageResult5 = new QLabel;
imageResult5->setBackgroundRole(QPalette::Base);
imageResult5->setSizePolicy(QSizePolicy::Ignored, QSizePolicy::Ignored);
imageResult5->setScaledContents(true);
//create scrollArea
showDialog = new QDialog(this);
scrollArea = new QScrollArea(showDialog);
   QString imgImport;
   QString nameImgToChange;
   QString folderNameToChange;
   QString pathDirOld;
   //ComboBox
   QDir dirList = "C:/Users/AmmEmpty/Desktop/imported/All/";
```

```
        dirList.setFilter(QDir::Dirs);
        QStringList folderList = dirList.entryList();
        ui->tc_comboBox->addItems(folderList);

        //Images List in directory
        QDir chosenDir2 = "C:/Users/AmmEmpty/Desktop/imported/All/All
Images/";

        imgSearchPath = "C:/Users/AmmEmpty/Desktop/imported/All/All Images/";
        histSearchPath = "C:/Users/AmmEmpty/Desktop/imported/Histogram/";
        QStringList fileFilter;
        fileFilter.append("*.jpg");
        fileFilter.append("*.png");
        fileFilter.append("*.bmp");
        QFileInfoList infoList2 = chosenDir2.entryInfoList();
        int count = infoList2.count();
        for (int j = 0; j < count ; j++){
            QFileInfo fileInfoInList = infoList2.at(j);
            QString fileNameInList = fileInfoInList.fileName();
            if( j >= 2 )
                imgList.push_back(fileNameInList.toStdString());
        }
        ////////////importDB/////////////
        importDB( imgList, imgSearchPath, histList, histSearchPath );
}
MainWindow::~MainWindow()
{
    delete dirmodel;
    delete filemodel;
    delete imageLabel;
    delete scrollArea;
    delete ui;
}
```

```
void MainWindow::open()
{
        QString path = "C:/Users/AmmEmpty/Desktop/Images/";
    queryName = QFileDialog::getOpenFileName(this, tr("Open File"), path);
    if (!queryName.isEmpty()) {
        QImage image(queryName);
        if (image.isNull()) {
            QMessageBox::information(this, tr("Image Viewer"), tr("Cannot
load %1.").arg(queryName));
            return;
        }
        imageLabel->setPixmap(QPixmap::fromImage(image));
    }
    else
        return;
      ui->ts_lineEdit->setText(queryName);
}
void MainWindow::on_ts_browseButton_clicked()
{
    open();
}
void MainWindow::on_tb_treeView_clicked(const QModelIndex &index)
{
    QString Path = dirmodel->fileInfo(index).absoluteFilePath();
    ui->tb_listView->setRootIndex(filemodel->setRootPath(Path));
}
void MainWindow::on_tb_listView_clicked(const QModelIndex &index)
{
      QString imgImport = filemodel->fileInfo(index).absoluteFilePath();
      QString nameImg;
      QImage image(imgImport);
      imageLabelBrowse->setPixmap(QPixmap::fromImage(image));
```

```
        ui->tb_imagescrollArea->setWidget(imageLabelBrowse);
}

void MainWindow::on_ts_searchButton_clicked()
{
    ui->widgetSearch->setStyleSheet("background-color:white;");
    ui->widgetSearch->show();
    ui->scrollResult->setGeometry(100,20,150,150);
    ui->scrollResult->setWidget(imageLabel);
      Mat src_base;
      src_base = imread(queryName.toStdString());
      const int maxResult = 6;
      SearchResult result = searchImg2(src_base, imgSearchPath, imgList,
maxResult);
      //ui->ts_lineEdit->setText( result.getCompareImg(0).c_str() );
      string imgName = imgSearchPath + result.getCompareImg(0);
      QImage image( imgName.c_str() );
   imageResult->setPixmap(QPixmap::fromImage(image));
   ui->scrollArea->setWidget(imageResult);
     string imgName2 = imgSearchPath + result.getCompareImg(1);
     QImage image2( imgName2.c_str() );
   imageResult2->setPixmap(QPixmap::fromImage(image2));
   ui->scrollArea_2->setWidget(imageResult2);
   string imgName3 = imgSearchPath + result.getCompareImg(2);
     QImage image3( imgName3.c_str() );
   imageResult3->setPixmap(QPixmap::fromImage(image3));
   ui->scrollArea_3->setWidget(imageResult3);
   string imgName4 = imgSearchPath + result.getCompareImg(3);
     QImage image4( imgName4.c_str() );
   imageResult4->setPixmap(QPixmap::fromImage(image4));
   ui->scrollArea_4->setWidget(imageResult4);
   string imgName5 = imgSearchPath + result.getCompareImg(4)
     QImage image5( imgName5.c_str() );
  imageResult5->setPixmap(QPixmap::fromImage(image5));
```

```
        ui->scrollArea_5->setWidget(imageResult5);
    }

    void MainWindow::on_backResult_clicked()
    {

        ui->widgetSearch->hide();
    }

    string getName(int n)
    {

                stringstream Photo;

                Photo << n;

                return Photo.str();
    }

    void MainWindow::on_importButton_clicked()
    {

        QString path = "C:/Users/AmmEmpty/Desktop/Images/";

        QStringList imgListImport = QFileDialog::getOpenFileNames(this,
tr("Open file"), path);

        //Date

        QDate date = QDate::currentDate();

        QString dateString = date.toString();

        //Time

        QString time = QDateTime::currentDateTime().toString("ddMMyyyyhhmm");

        //Create folder by date current

        QString folder = "C:/Users/AmmEmpty/Desktop/imported/"+dateString;

        QDir().mkdir(folder);

        for(int i = 0 ; i < imgListImport.size() ; i++){

                    //Save img

                    Mat image1 = imread(imgListImport[i].toStdString(), 1);

        imwrite("C:/Users/AmmEmpty/Desktop/imported/"+dateString.toStdString()
+"/"+time.toStdString()+getName(i)+".jpg",image1);

                    imwrite("C:/Users/AmmEmpty/Desktop/imported/All/All
Images/"+time.toStdString()+getName(i)+".jpg",image1);
```

```
        }
    }
    void MainWindow::on_tc_treeView_clicked(const QModelIndex &index)
    {
        QString path = dirmodelCat->fileInfo(index).absolutePath();
        QString pathDirOld = dirmodelCat->fileInfo(index).absoluteFilePath();
        ui->tc_listView->setRootIndex(filemodelCat->setRootPath(pathDirOld));
        ui->tc_lineEdit->setText(path);
    }
    ///////Category///////
    void MainWindow::on_tc_listView_clicked(const QModelIndex &index)
    {
        QString imgImport = filemodelCat->fileInfo(index).absoluteFilePath();
        QString nameImgToChange = filemodelCat->fileInfo(index).fileName();
        QImage image(imgImport);
        imageLabelCat->setPixmap(QPixmap::fromImage(image));
        ui->tc_scrollArea_6->setWidget(imageLabelCat);
        ui->tc_lineEdit->setText(nameImgToChange);
        //double similarity = compareHist2(histQ, histDB);
        QDir dirListAuto = "C:/Users/AmmEmpty/Desktop/imported/All/";
        dirListAuto.setFilter(QDir::Dirs);
        Mat src_base;
        src_base = imread("C:/Users/AmmEmpty/Desktop/imported/All/All
Images/"+nameImgToChange.toStdString());
        QStringList folderListAuto = dirListAuto.entryList();
        vector<double> histValue;
        for(int i = 3 ; i < folderListAuto.count() ; i++){
            QDir pathEachFolder = folderListAuto.at(i);
            QString pathEach = folderListAuto.at(i);
            QString fullPath =
"C:/Users/AmmEmpty/Desktop/imported/All/"+pathEachFolder.dirName()+"/";
            QDir pathImgInFolder = fullPath;
            ui->ts_lineEdit->setText(nameImgToChange);
```

```
                QStringList filter;
                filter.append("*.jpg");
                filter.append("*.png");
                filter.append("*.bmp");
                const int maxResult = 1;
                QFileInfoList imgInfoList = pathImgInFolder.entryInfoList();
                for(int j = 3 ; j < imgInfoList.count() ; j++){
                      QFileInfo fileImgList = imgInfoList.at(j);
                      QString fileImgName =
"C:/Users/AmmEmpty/Desktop/imported/All/All Images/"+fileImgList.fileName();
                      //ui->ts_lineEdit->setText(fileImgName);
                      SearchResult category =  searchCat(src_base,
fileImgName.toStdString(), pathEach.toStdString(), maxResult);
                      catList.push_back(category.getImgPath(0).c_str());
                      //catList.push_back(category.getImgPath(j-3).c_str());
                      //ui->ts_lineEdit->setText(catList[0]);
                }
          }
          ui->tc_comboBox->addItem(catList[0]);
          //ui->tc_comboBox->insertItem(0, catList[0]);
          //for(int i = 0 ; i < folderList.count() ; i++){
          //     QDir pathFolder = "C:/Users/AmmEmpty/Desktop/imported/All/All
Images/"+folderList.at(i)+"/";
          //     QStringList fileFilterPath;
          //     fileFilterPath.append("*.jpg");
          //     fileFilterPath.append("*.png");
          //     fileFilterPath.append("*.bmp");
          //     vector<string> listEachFolder;
          //     QFileInfoList infoListImgCat = pathFolder.entryInfoList();
          //     int count = infoListImgCat.count();
          //     for (int j = 0; j < count ; j++){
          //          QFileInfo fileInfoListCat = infoListImgCat.at(j);
          //          fileNameInListImgCat = fileInfoListCat.fileName();
```

```
            //
            listEachFolder.push_back(fileNameInListImgCat.toStdString());
            //      }
            //      /*if(fileNameInListImgCat.append()){
            //      }*/
            //}
            //ui->tc_comboBox->addItems(folderList);
            //ui->ts_lineEdit->setText( result.getCompareImg(1).c_str() );
    }
    void MainWindow::on_tc_newFolderButton_clicked(
    {
        bool ok;
        QString text = QInputDialog::getText(this, tr("Create new folder"),
    QLineEdit::Normal                    tr("Folder Name:"),
                                         QDir::home().dirName(), &ok);
        QString folder = "C:/Users/AmmEmpty/Desktop/imported/All/"+text;
        if (ok && !text.isEmpty())
         QDir().mkdir(folder);
    }
    void MainWindow::on_tc_comboBox_activated(const QString &arg1)
    {
        folderNameToChange = ui->tc_comboBox->currentText();
    }
    void MainWindow::on_tc_saveButton_ok_clicked()
    {
        QString fol = ui->tc_comboBox->currentText();
        QString test = ui->tc_lineEdit->displayText();
        string testtest = "C:/Users/AmmEmpty/Desktop/imported/All/All
Images/"+test.toStdString();
        Mat image1 = imread(testtest, 1);
        imwrite("C:/Users/AmmEmpty/Desktop/imported/All/"+fol.toStdString()+"/
"+test.toStdString(),image1);
}
```

SearhImg.cpp

```
#include "searchImg.h"

//-------------------------

//Class SearchResult

////-------------------------

//void SearchResult::addResult(CompareResult a_result){

//     if(resultVector.size() < maxResult){

//               resultVector.push_back(a_result);

//               sort( resultVector.begin(), resultVector.end(),
std::greater<CompareResult>() );

//     }

//     else {

//               if( resultVector[0] < a_result ) {

//                    resultVector[0] = a_result;

//                    sort( resultVector.begin(), resultVector.end(),
std::greater<CompareResult>() );

//               }

//     }

//}

//-------------------------

void SearchResult::addResult(CompareResult a_result){

      if(resultVector.size() < maxResult){

            resultVector.push_back(a_result);

            sort( resultVector.begin(), resultVector.end(),
std::greater<CompareResult>() );

      }

      else {

            if( resultVector.size() == maxResult )

                  resultVector.push_back(a_result);

            else

                  resultVector[maxResult] = a_result;

            sort( resultVector.begin(), resultVector.end(),
std::greater<CompareResult>() );
```

```
        }
    }
    void SearchResult::addSimilarity(CatResult result){

        if(catResultVector.size() < maxResult){
            catResultVector.push_back(result);
            sort( catResultVector.begin(), catResultVector.end(),
std::greater<CatResult>() );
        }
    }
    //-------------------------
    void SearchResult::display()
        for( int i = 0; i < resultVector.size(); ++i )
            resultVector[i].display();
        cout << endl;
}
//-------------------------
cv::Mat calculateHist( const cv::Mat& Q ) {
    cv::Mat hist;
    //MatND hist;
    cv::Mat hsv( Q.rows, Q.cols, CV_32FC3 );
    cvtColor( Q, hsv, CV_BGR2HSV );
    int h_bins = 30; int s_bins = 32;
    int histSize[] = { h_bins, s_bins };
    // hue varies from 0 to 256, saturation from 0 to 180
    float h_ranges[] = { 0, 180 };
    float s_ranges[] = { 0, 256 };
    const float* ranges[] = { h_ranges, s_ranges };
    int channels[] = { 0, 1 };
    calcHist( &hsv, 1, channels, Mat(), hist, 2, histSize, ranges, true,
false );
    normalize( hist, hist, 0, 1, NORM_MINMAX, -1, Mat() );
    return hist;
```

```
        }

        //-------------------------

        cv::Mat readHist( const string& filename ) {

              cv::Mat hist = imread(filename);

              return hist;

        }

        //-------------------------

        double compareHist( const cv::Mat& histQ, const cv::Mat& histDB ) {

              int compare_method = INTERSECTION_METHOD;

              double similarity = compareHist( histQ, histDB, compare_method );

              return similarity;

      }

      //-------------------------

      SearchResult searchImg2( const cv::Mat& Q, const string& imgPath, const
      vector<string>& imgList, int maxResult ){

            ofstream strm;

            strm.open("output.txt");

            cv::MatND histQ = calculateHist2(Q);

            SearchResult result(maxResult);  //??????

            for(int i = 0; i < imgList.size(); i++){

                  cv::Mat imgDB = imread( imgPath + imgList[i] );

                  cv::MatND histDB = calculateHist2(imgDB);

                  double similarity = compareHist2(histQ, histDB);

                  CompareResult c( similarity, imgList[i] );

                  strm << imgList[i] << " " << similarity << endl;

                  result.addResult(c);

            }

            for(int i = 0; i < 6; i++){

                  strm << result.getCompareImg(i) << " " <<
      result.getSimilarity(i) << endl;

            }

            return result;

}
```

```
//-------------------------


SearchResult searchCat( const cv::Mat& Q, const string imgName, const string
path, int maxResult ){

      cv::MatND histQ = calculateHist2(Q);

      SearchResult result(maxResult);

      cv::Mat imgDB = imread(imgName);

      cv::MatND histDB = calculateHist2(imgDB);

      double similarity = compareHist2(histQ, histDB);

      CatResult c(similarity, imgName, path);

      result.addSimilarity(c);

      return result;

}

//-------------------------

cv::MatND calculateHist2( const cv::Mat& Q ) {

      MatND hist;

      cv::Mat hsv;

      cvtColor( Q, hsv, CV_BGR2HSV );

      int h_bins = 30; int s_bins = 32;

      int histSize[] = { h_bins, s_bins };

      // hue varies from 0 to 256, saturation from 0 to 180

      float h_ranges[] = { 0, 180 };

      float s_ranges[] = { 0, 256 };

      const float* ranges[] = { h_ranges, s_ranges };

      int channels[] = { 0, 1 };

      calcHist( &hsv, 1, channels, Mat(), hist, 2, histSize, ranges, true,
false );

      normalize( hist, hist, 0, 1, NORM_MINMAX, -1, Mat() );

      return hist;

}

//-------------------------

double compareHist2( const cv::MatND& histQ, const cv::MatND& histDB ) {

      int compare_method = INTERSECTION_METHOD;
```

```
        double similarity = compareHist( histQ, histDB, compare_method );
        return similarity;
    }
    //-------------------------
    void importDB( const vector<string>& imgList, vector<string>& histList ){
        for(int i = 0; i < imgList.size(); i++){
            cv::Mat img = imread( imgList[i] );
            cv::Mat hist = calculateHist( img );
            string histName = imgList[i] + "_hist.bmp";
            imwrite( histName, hist );
            histList.push_back(histName);
        }
    }
    //-------------------------
    void importDB( const vector<string>& imgList, const string& imgPath,
    vector<string>& histList, const string& histPath ){
        for(int i = 0; i < imgList.size(); i++){
            cv::Mat img = imread( imgPath + imgList[i] );
            cv::Mat hist = calculateHist( img );
            string histName = imgList[i] + "_hist.bmp";
            imwrite( histPath + histName, hist );
            histList.push_back(histName);
        }
}
```

img_processing.cpp

```
#include "img_processing.h"

double compareImgs( string filename1, string filename2, int compare_method )
{
    Mat src_base, hsv_base;
    Mat src_test1, hsv_test1;
    src_base = imread( filename1, 1 ); //image query
    src_test1 = imread( filename2, 1 ); //image database
    // Convert to HSV
```

```
        cvtColor( src_base, hsv_base, CV_BGR2HSV );

        cvtColor( src_test1, hsv_test1, CV_BGR2HSV );

        // Using 30 bins for hue and 32 for saturation

        int h_bins = 50; int s_bins = 60;

        int histSize[] = { h_bins, s_bins };

        // hue varies from 0 to 256, saturation from 0 to 180

        float h_ranges[] = { 0, 256 };

        float s_ranges[] = { 0, 180 };

        const float* ranges[] = { h_ranges, s_ranges };

        // Use the o-th and 1-st channels

        int channels[] = { 0, 1 };

        // Histograms

        MatND hist_base;

        MatND hist_test1;

        // Calculate the histograms for the HSV images

        calcHist( &hsv_base, 1, channels, Mat(), hist_base, 2, histSize,
ranges, true, false );

        normalize( hist_base, hist_base, 0, 1, NORM_MINMAX, -1, Mat() );

        calcHist( &hsv_test1, 1, channels, Mat(), hist_test1, 2, histSize,
ranges, true, false );

        normalize( hist_test1, hist_test1, 0, 1, NORM_MINMAX, -1, Mat() );

        double base_test1 = compareHist( hist_base, hist_test1, compare_method
);

        //printf( " Method [%d] Perfect, Base-Test(1), Base-Test(2) : %f, %f,
%f \n", i, base_base, base_test1, base_test2 );

        return base_test1;

}
```

Mainwindows.h

```
#ifndef MAINWINDOW_H

#define MAINWINDOW_H

#include <QMainWindow>

#include <QtCore>
```

```
#include <QtGui>
#include <string>
#include "img_processing.h"
#include "searchImg.h"
using namespace std;
namespace Ui {
class MainWindow;
}
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
private slots:
    void open();
    void on_ts_browseButton_clicked();
    void on_tb_treeView_clicked(const QModelIndex &index);
    void on_tb_listView_clicked(const QModelIndex &index);
    void on_ts_searchButton_clicked();
    void on_backResult_clicked();
    void on_importButton_clicked();
    void on_tc_treeView_clicked(const QModelIndex &index);
    void on_tc_listView_clicked(const QModelIndex &index);
    void on_tc_saveButton_ok_clicked();
    void on_tc_newFolderButton_clicked();
    void on_tc_comboBox_activated(const QString &arg1);
private:
    Ui::MainWindow *ui;
    QFileSystemModel *filemodel;
    QFileSystemModel *dirmodel;
    QFileSystemModel *fileCat;
```

```
    QPixmap *imageBrowse;
    QDialog *showDialog;
    QLabel *imageLabel;
    QLabel *imageLabelBrowse;
    QScrollArea *scrollArea;
    int i;
      QFileSystemModel *filemodelCat;
      QFileSystemModel *dirmodelCat;
      QLabel *imageLabelCat;
   QGraphicsPixmapItem *item;
   QLabel *imageResult;
   QLabel *imageResult2;
   QLabel *imageResult3;
   QLabel *imageResult4;
   QLabel *imageResult5;
     QString queryName;
     QString imgImport;
     QString nameImg;
     QString nameImgToChange;
     QString folderNameToChange;
     QString pathDirOld;
     //img list
     QFileInfoList infoList2;
     QStringList fileFilter2;
     QDir chosenDir2;
     QFileInfo fileInfo2;
     QString imageData2;
     vector<string> imgList, histList;
     string imgSearchPath, histSearchPath;
     QString qStrResult;
     QStringList ListEeachFolder;
     QStringList folderList;
```

```
        QString fileNameInListImgCat;

        QVector<QString> catList;

};

#endif // MAINWINDOW_H
```

SearchImg.h

```
#ifndef SEARCH_IMG_HPP

#define SEARCH_IMG_HPP

#include "opencv2/highgui/highgui.hpp"

#include "opencv2/imgproc/imgproc.hpp"

#include <iostream>

#include <fstream>

#include <queue>

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

using namespace cv;

//--------------------------

const int CORRELATION                = 0;

const int CHI_SQUARE_METHOD          = 1

const int INTERSECTION_METHOD    = 2;

const int BHATTACHARAYA_METHOD   = 3;

//--------------------------

class CompareResult{

      double similarity;

      string compareImg;

public:

      CompareResult( double s, string comImg ) : similarity(s),
compareImg(comImg) {}

      bool operator< (const CompareResult& another) const { return
(similarity < another.similarity);  }

      bool operator> (const CompareResult& another) const { return
(similarity > another.similarity);  }
```

```
        void display() { cout << "Image: " << compareImg << ", Similarity: "
    << similarity << endl; }

        double getSimilarity() { return similarity; }

        string getCompareImg() { return compareImg; }
    };

    //-------------------------

    class CatResult{

        double similarity;

        string imgName;

        string path;

    public:

        CatResult( double s, string imgName , string path) : similarity(s),
    imgName(imgName), path(path) {}

        bool operator< (const CatResult& another) const { return (similarity <
    another.similarity); }


        bool operator> (const CatResult& another) const { return (similarity >
    another.similarity); }

        double getSimilarity() { return similarity; }

        string getImgName() { return imgName; }

        string getImgPath() { return path;}
    };

    //-----------------------

    class SearchResult{

        vector<CompareResult> resultVector;

        vector<CatResult> catResultVector;

        //priority_queue<CompareResult> resultVector;

        int maxResult;

public:

        SearchResult( int m ) : maxResult(m) {}

        void addResult(CompareResult a_result);

        void addSimilarity(CatResult result);

        void display();

        void addSimilarity(string similarity);
```

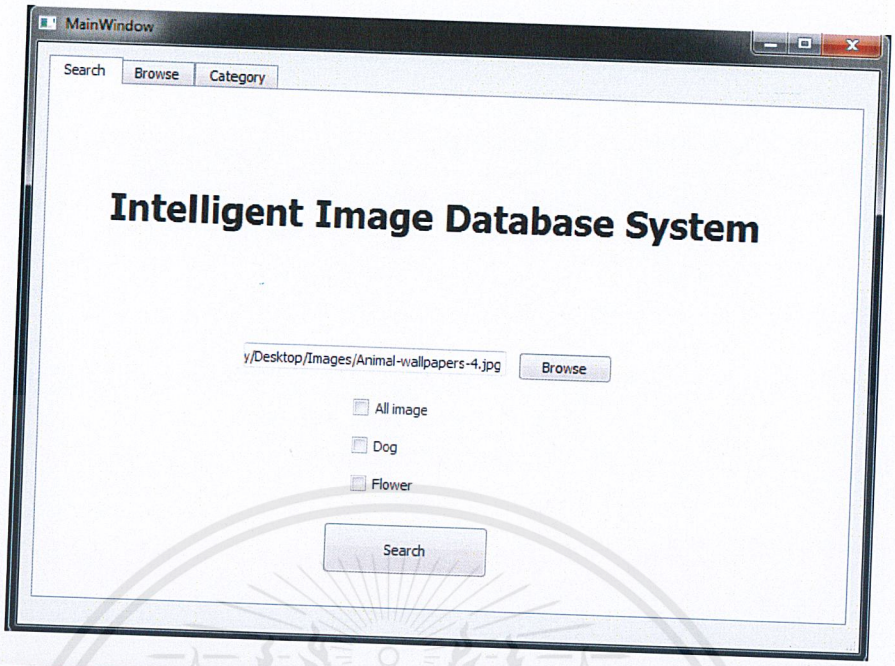
```
        double getSimilarity( int index ) { return
resultVector[index].getSimilarity(); }

        string getCompareImg( int index ) { return
resultVector[index].getCompareImg(); }

        string getImgPath( int index ) { return
catResultVector[index].getImgPath();}

};

//-------------------------

//-------------------------

SearchResult searchImg( const cv::Mat& Q, const vector<string>& histList,
int maxResult);

        cv::Mat calculateHist( const cv::Mat& Q );

        cv::Mat readHist( const string& filename );

        double  compareHist( const cv::Mat& histQ, const cv::Mat& histDB );
//-------------------------


SearchResult searchImg2( const cv::Mat& Q, const string& imgPath, const
vector<string>& imgList, int maxResult);

        cv::MatND calculateHist2( const cv::Mat& Q );

        double  compareHist2( const cv::MatND& histQ, const cv::MatND& histDB
);

//-------------------------

SearchResult searchCat( const cv::Mat& Q, const string imgName,const string
path, int maxResult );

        cv::MatND calculateHist2( const cv::Mat& Q );

        double  compareHist2( const cv::MatND& histQ, const cv::MatND& histDB
);

//-------------------------

void importDB( const vector<string>& imgList, vector<string>& histList );

void importDB( const vector<string>& imgList, const string& imgPath,
vector<string>& histList, const string& histPath );

//-------------------------

//int findHistogram();

int findCategory(Mat src_base, string imgDB, int k);

int findCategoryHist(MatND hist, string imgDB, int k);

//-------------------------
```

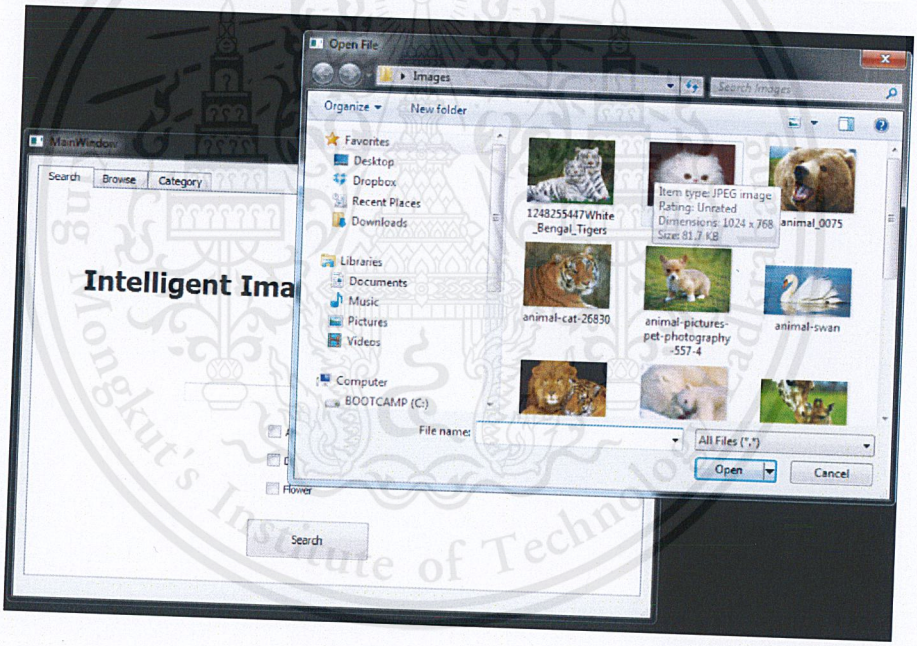
```
#endif
```

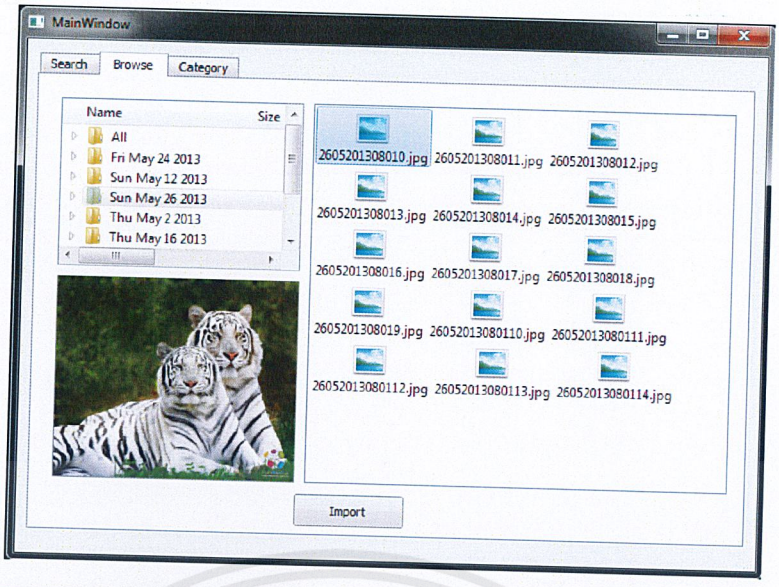
img_processing.h

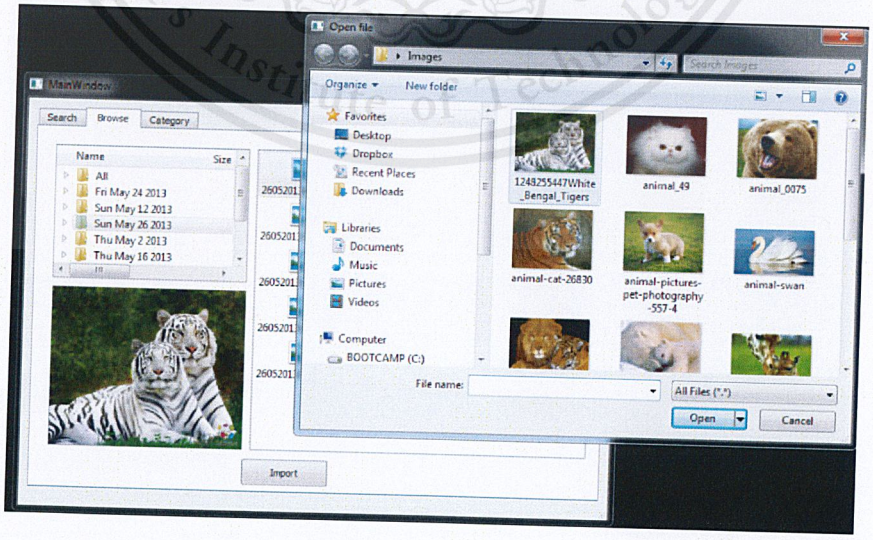
```
#ifndef IMG_PROC_H

#define IMG_PROC_H

#include <iostream>

#include <conio.h>

#include <string>

#include "opencv2/highgui/highgui.hpp"

#include "opencv2/imgproc/imgproc.hpp"

using namespace std;

using namespace cv;

double compareImgs( string filename1, string filename2, int compare_method
);

#endif
```

Simple GUI









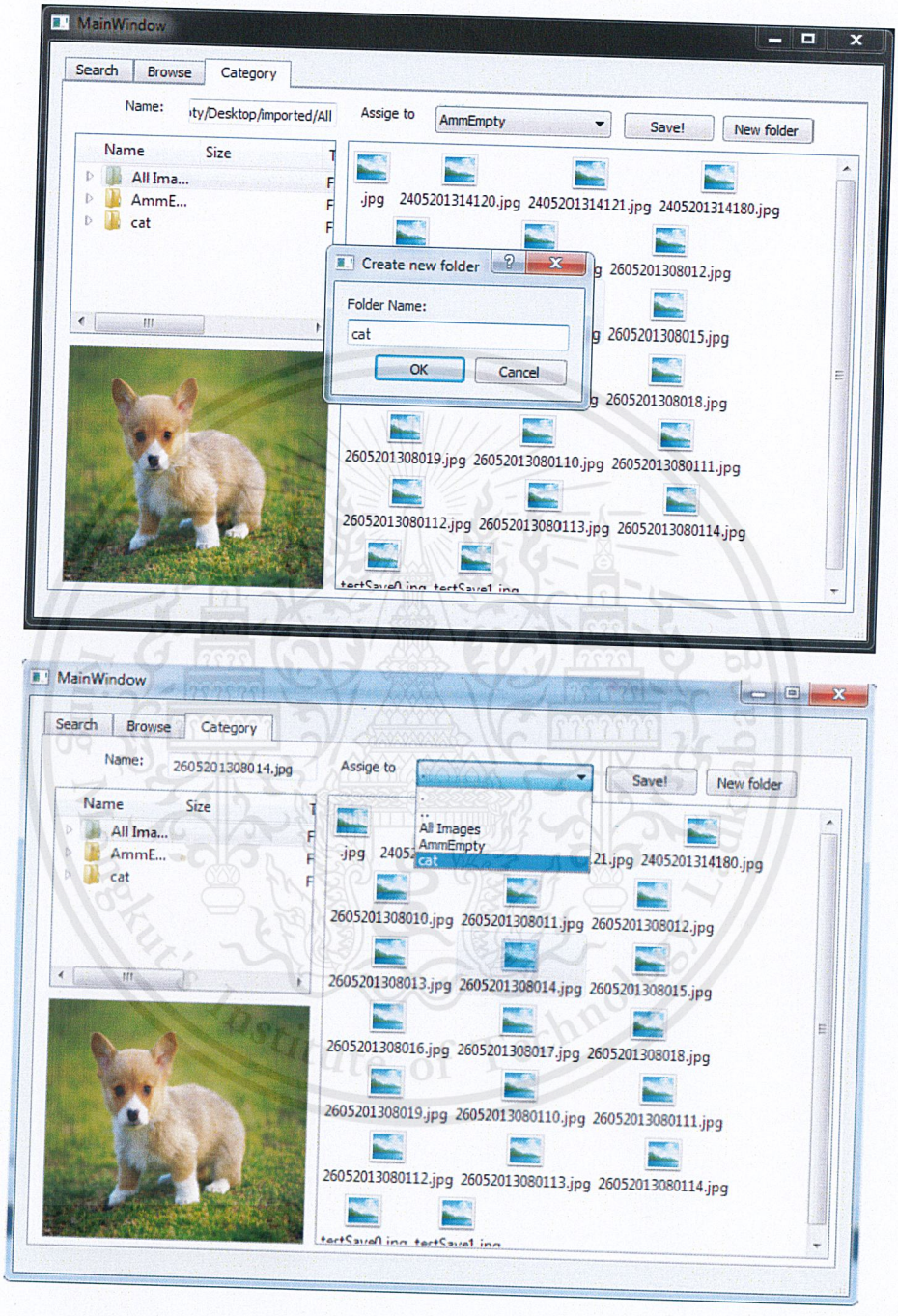


Figure B.1 Simple Graphic User Interface for each screen