การสืบค้นแบบอักษรโดยใช้ตัวอย่าง

CONTENT-BASED FONT RETRIEVAL

นายนันทวรรธน์ สมยา รหัส 51362022

นายศิริศักดิ์ ทำสวน รหัส 51363098

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2554

ใบรับรองปริญญานิพนธ์

หัวข้อโครงงาน	**การสืบค้นแบบอักษรโดยใช้ตัวอย่าง**
ผู้ดำเนินโครงงาน	**นายนันทวรรธน์ สมยา รหัส 51362022**
	นายศิริศักดิ์ ทำสวน รหัส 51363098
อาจารย์ที่ปรึกษา	อาจารย์รัฐภูมิ วรานุสาสน์
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2554

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

.....ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. พนมขวัญ ริยะมงคล)

.....กรรมการ
(ดร. สุรเดช จิตประไพกุลศาล)

.....กรรมการ
(อาจารย์ภาณุพงศ์ สอนคม)

.....กรรมการ
(อาจารย์รัฐภูมิ วรานุสาสน์)

หัวข้อโครงงาน	การสืบค้นแบบอักษรโดยใช้ตัวอย่าง
ผู้ดำเนินโครงงาน	นายนันทวรรธน์ สมขา รหัส 51362022
	นายศิริศักดิ์ ทำสวน รหัส 51363098
อาจารย์ที่ปรึกษา	อาจารย์รัฐภูมิ วรานุสาสน์
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2554

.....

บทคัดย่อ

โครงงานนี้เป็นการศึกษาและพัฒนาโปรแกรมสืบค้นแบบอักษรโดยใช้ตัวอย่าง เพื่ออำนวยความสะดวกแก่ผู้ใช้ที่ต้องการทราบชื่อของแบบอักษรในภาพ เนื่องจากเอกสารที่ใช้อยู่ ในปัจจุบันใช้แบบอักษรที่ แตกต่างกัน ซึ่งเป็นการ ยากที่จะ ทราบ ชื่อของแบบอักษร ที่สนใจ คณะผู้จัดทำได้พัฒนาโปรแกรมสืบค้นแบบอักษรโดยตัวอย่าง ซึ่งเป็นเครื่องมือที่ช่วย ค้นหาชื่อแบบอักษร จากภาพตัวอักษร โปรแกรมจะตัดตัวอักษรออกจากภาพโดย การทำโปรเจคชันทั้งในแนวตั้งและแนวนอน เมื่อได้ตัวอักษรออก มาจะ ทำการแบ่งภาพ อักษร ออกเป็นบล็อก โดยที่แต่ละบล็อกจะให้รหัสตามปริมาณของสีดำ จากนั้นจะคำนวณค่าความเหมือนจากรหัสสีที่ได้และแสดงผลลัพธ์ออกมาเป็นร้อยละ ผลลัพธ์ที่ได้สามารถบอกชื่อแบบอักษรโดยเรียงลำดับความใกล้เคียง จากมากไปน้อย ซึ่งมีความถูกต้องเป็นที่น่าพอใจ

Project Title	Content-Based Font Retrieval		
Name	Mr. Nantawat	Somya	ID. 51362022
	Mr. Sirisak	Tamsuan	ID. 51363098
Project advisor	Mr. Rattapoom Waranusast		
Major	Computer Engineering		
Department	Electrical and Computer Engineering		
Academic year	2011		

Abstract

This project is a development of Content-Based Font Retrieval to facilitate users in searching for font names in an image containing characters. Nowadays, documents contain enormous amount of fonts, which is difficult for a user to know a font name from an image of characters. We developed a system to search and retrieve a font name by example in an image. The program receives an image containing characters as an input, then extracts each character using both vertical and horizontal projections. The extracted characters are then divided into smaller blocks. Each block is assigned with a code according to its ratio of black to white pixels. Similarity of the character and the stored templates are then computed based on differences of codes on the blocks. The results are shown ranked from highest to lowest similarity. Experimental results showed that the algorithm gave satisfactory results.

กิตติกรรมประกาศ

โครงงานนี้สำเร็จลุล่วงไปด้วยความกรุณาเป็นอย่างยิ่งจากอาจารย์รัฐภูมิ วรานุสาสน์ ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงาน และ ให้ความกรุณาในการให้ความช่วยเหลือในการให้คำปรึกษาและตรวจทานปริญญานิพนธ์มาโดยตลอด คณะผู้ดำเนินงานโครงงานจึงขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบคุณคณาจารย์ทุกท่านในภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ ที่ถ่ายทอดความรู้ให้กับคณะผู้ดำเนินงานจนคณะผู้ดำเนินงานสามารถจัดทำโครงงานได้สำเร็จ

ขอขอบคุณคุณพ่อคุณแม่ที่ให้กำลังใจเสมอมา

ขอขอบคุณเพื่อนๆภาควิชาวิศวกรรมคอมพิวเตอร์ที่ให้ความช่วยเหลือและคำปรึกษาในเรื่องต่างๆเป็นอย่างดี

รวมถึงขอขอบคุณทุกท่านที่ไม่ได้กล่าวถึง ณ ที่นี้ ที่ให้ความช่วยเหลือและเป็นกำลังใจในการจัดทำโครงงานนี้ส่งผลให้สำเร็จลุล่วงไปด้วยดี

นายนันทวรรธน์ สมยา

นายศิริศักดิ์ ทำสวน

สารบัญ

หน้า

สารบัญ(ต่อ)

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

สารบัญรูป(ต่อ)

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ปัจจุบันงานเอกสารส่วนใหญ่ทำโดยคอมพิวเตอร์ เพราะมีความสะดวก รวดเร็ว สามารถเพิ่ม ลบ แก้ไขได้ง่าย นอกจากนี้ยังสามารถกำหนดขนาด หรือลักษณะของตัวอักษรได้ เช่น ตัวหนา ตัวเอียง ขีดเส้นใต้ และที่สำคัญยังมีแบบอักษร (Font) ให้เลือกใช้ได้หลายแบบ ซึ่งในปัจจุบันแบบอักษรมีมากมายหลายร้อยแบบให้เลือกตามความต้องการของผู้ใช้ แต่บาง ครั้ง ในชีวิตประจำวัน เราอาจจะไปพบแบบอักษรที่น่าสนใจ หรือแบบอักษรที่ถูกใจจากในเว็บหรือรูปภาพต่างๆแล้ว ต้องการทราบว่า ชื่อของแบบอักษรนี้มีชื่อว่าอะไรเพื่อที่จะได้หาแบบอักษรที่ต้องการมาใช้กับเอกสารของตนได้ แต่การที่จะหาแบบอักษร หรือเปรียบเทียบแบบ อักษรด้วยมือจะใช้เวลานานและทำได้ ไม่สะดวก

ผู้จัดทำจึงได้นำแนวความคิดนี้มาแก้ปัญหาโดย จัดทำโปรแกรมเพื่อค้นหาแบบอักษร แบบอัตโนมัติจากรูปสแกนหรือรูปภาพ ตัวอักษร ตามตัวอย่างได้ เพื่อลดเวลาในการค้นหาและอำนวยความสะดวกมากขึ้น จากเดิมที่ต้องใช้คน ในการไล่แบบอักษรที่อยู่บนเครื่อง ซึ่งใช้เวลานาน และบางครั้งก็ไม่ตรงตามที่ต้องการ

เมื่อเราทราบแบบอักษรที่ต้องการแล้วจะสามารถ ไปดาวน์โหลดฟอนต์ (.ttf) มาติดตั้ง ในเครื่องเพื่อนำไปใช้ในงานเอกสารต่อไปได้

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษาอัลกอริทึม (Algorithm) ของการประมวลผลภาพดิจิทัล

1.2.2 เพื่อออกแบบโปรแกรม ค้นหาแบบอักษรจาก ภาพตัวอย่าง

1.2.3 เพื่อให้ได้โปรแกรมที่สามารถบอก ชื่อแบบอักษรได้ ถูกต้องตาม ภาพตัวอย่าง

1.3 ขอบข่ายของงาน

1.3.1 ตัวอักษรที่นำมาเป็นตัวอย่างจะต้องเป็นตัวอักษรสีดำและมีพื้นหลังที่เป็นสีขาวเท่านั้น

1.3.2 ตัวอักษรที่นำมาเป็นตัวอย่างต้องเป็นตัวอักษรภ าษาอังกฤษและตัวเลขเท่านั้นไม่รว มแบบอักษรที่เป็นภาษาไทย

1.3.3 โปรแกรมสามารถวิเคราะห์ได้เฉพาะตัวอักษรบรรทัดเดียวเท่านั้น

1.3.4 ภาพที่นำมาวิเคราะห์ต้องเป็นภาพจากการ แคปเจอร์จากคอมพิวเตอร์ หรือภาพที่มาจา กการสแกนเท่านั้น ไม่รวมภาพที่นำมาจากภาพถ่าย

1.4 ขั้นตอนการดำเนินงาน

1.4.1 ศึกษาค้นคว้าข้อมูลเกี่ยวกับ การประมวลผลภาพ (Image Processing)

1.4.2 เขียนโปรแกรมตัดแยกตัวอักษรออกจากกัน

1.4.3 เขียนโปรแกรมเปรียบเทียบความแตกต่างระหว่างภาพที่นำมาเป็นตัวอย่างกับภาพในฐานข้อมูล

1.4.4 ทดสอบการทำงานของระบบ

1.4.5 แก้ไขข้อผิดพลาดและเก็บรายละเอียดส่วนต่างๆของโครงงาน

1.4.6 สรุปผลการทดลองและจัดทำรูปเล่มโครงงาน

1.5 กิจกรรมการดำเนินการ

ตารางที่ 1.1 แสดงขั้นตอนการดำเนินงาน

กิจกรรม	ปี 2554							ปี 2555				
	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.
1.รวบรวมข้อมูลศึกษาค้นคว้า	←	→										
2. ศึกษาวิธีการประมวลผลภาพ และ การแยกตัวอักษรออกจากพื้นหลัง		←	—	—	→							
3. ศึกษาการเปรียบเปรียบความแตกต่างของแต่ละตัวอักษร				←	—	—	→					

4. สร้างฐานข้อมูลเพื่อเก็บไว้ใช้เปรียบเทียบรูปแบบของตัวอักษร						←	—	—	→			
5.ทดสอบการใช้งานของโปรแกรม									←	—	—	→
6. จัดทำรูปเล่มรายงานและสรุปผล		←	—	—	—	—	—	—	—	—	—	→

1.6 ผลที่คาดว่าจะได้รับ

1.6.1 สามารถทำงานได้ถูกต้อง อำนวยความสะดวกแก่ผู้ใช้ได้

1.6.2 สามารถสร้างโปรแกรมที่สามารถบอกแบบอักษรได้ ถูกต้องตามตัวอย่าง

1.6.3 สามารถนำความรู้เกี่ยวกับการวิเคราะห์รูปภาพไปประยุกต์ใช้กับงานอื่นได้

1.6.4 ได้พัฒนากระบวนการคิดวิเคราะห์อัลกอริทึมที่เหมาะสมกับปัญหา

1.7 งบประมาณ

1.7.1 ค่าถ่ายเอกสาร และการเข้าเล่ม	1000 บาท
1.7.2 หนังสือคู่มือเกี่ยวกับการใช้ศึกษาค้นคว้า	500 บาท
1.7.3 ค่าพิมพ์เอกสาร พร้อม ส่วนที่แก้ไข	**500 บาท**
รวมทั้งสิ้น	**2000 บาท**

บทที่ 2
หลักการและทฤษฎีเบื้องต้น

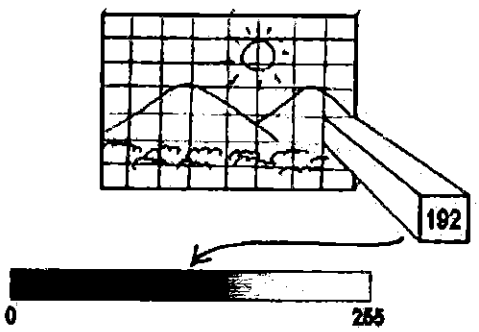
2.1 ความรู้เบื้องต้นเกี่ยวกับการประมวลผลภาพ

2.1.1 ลักษณะและความหมายของพิกเซล [1]

ในงาน กราฟิกที่ใช้ในงานคอมพิวเตอร์พิกเซล (Pixel) ถือเป็นหน่วยย่อยที่เล็กที่สุดของรูปภาพ เป็นจุดเล็กๆ ที่รวมกันทำให้เกิดภาพขึ้น ภาพหนึ่ง ภาพ จะประกอบด้วย พิกเซล หรือจุดมากมาย ซึ่งแต่ละภาพที่สร้างขึ้นจะมีความหนาแน่นของจุด เหนือพิกเซล เหล่านี้แตกต่างกันไป ซึ่งความละเอียดของภาพมีหน่วยเป็นพีพีไอ (Pixel Per Inch) คือ จำนวนจุดต่อนิ้ว พิกเซล มีความสำคัญต่อการสร้างภาพของคอมพิวเตอร์มาก เพราะองค์ประกอบของกราฟิก เช่น จุด เส้น แบบลายและสีของภาพนั้นประกอบขึ้นจากพิกเซล เมื่อเราขยายภาพจะเห็นเป็นภาพจุด โดยปกติแล้ว ภาพที่มีความละเอียดสูงหรือคุณภาพดีควรจะมีค่าความละเอียด 300 X 300 พีพีไอ ขึ้นไป ยิ่งค่าพีพีไอ สูงขึ้นเท่าไร ภาพก็จะมีความละเอียดคมชัดขึ้นมาก ขึ้นเท่านั้น ขณะเดียวกันจุดหรือพิกเซล แต่ละจุดก็จะแสดงคุณสมบัติทางสีให้กับ ภาพด้วย โดยแต่ละจุดจะเป็นตัวสร้างสีประกอบกันเป็นภาพรวม ซึ่งอาจมีขนาดความเข้มและสีแตกต่างกันได้ ทำให้เกิดเป็นภาพที่มีสีสันต่างๆ การแสดงผลของอุปกรณ์แสดงผล (Output Devices) ไม่ว่าจะเป็นเครื่องพิมพ์แบบดอทเมทริกซ์ (Dot-matrix) หรือแบบเลเซอร์ (Laser) รวมทั้งจอภาพ จะเป็นการแสดงผลแบบ แรสเตอร์ (Raster Devices) นั่นคือ อาศัยการรวมกันของพิกเซลออกมาเป็นรูป

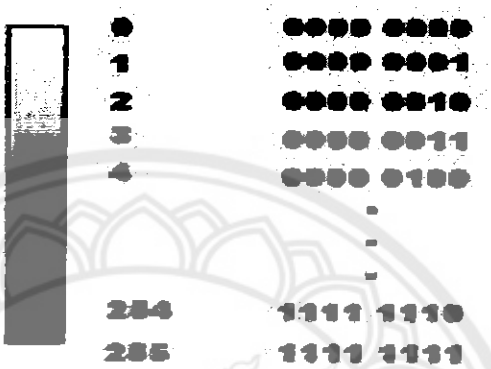
2.1.2 การประมาณค่าความเข้มของแสง (Image Quantization) [2]

เมื่อเราได้ภาพจากการสุ่มตัวอย่าง (Sampling) มาแล้ว แต่ละจุดในภาพจะ ถูกแทนด้วยสี ภาพในโทนสีระดับเทา (Grayscale) จะประกอบไปด้วยสีดำ และ ไล่เฉดสีจางลงไปจนถึงสีขาวดัง รูปที่ 2.1



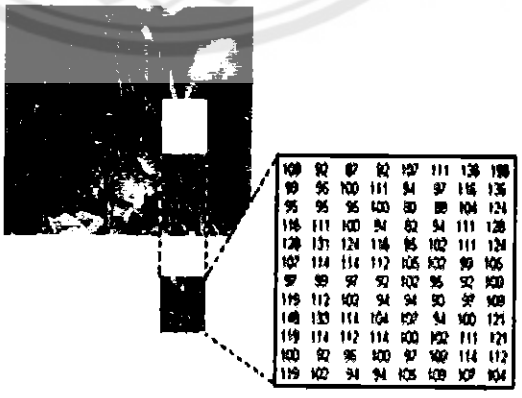
รูปที่ 2.1 การสุ่มตัวอย่างภาพสีเป็นภาพระดับเทา

สีดำ จะแทนด้วยค่าตัวเลข 255 สีขาวจะแทนด้วยค่าตัวเลขคือ 0 รวมทั้งสิ้น 256 ระดับสี (0-255) หรือ 2 กำลัง 8 โดยที่ 8 ก็คือ จำนวนบิต (bit) ในหน่วยความจำที่ใช้ในการเก็บค่านี้หนึ่งค่า เพราะฉะนั้น สีดำ จะถูกแทนด้วยรหัสในเลขฐานสองคือ 00000000 และสีขาวก็จะถูกแทนด้วยรหัส 11111111และสีที่อยู่ตรงกลางระหว่างสีดำกับสีขาวก็จะไล่ไปตามลำดับการนับของบิตในเลขฐานสองดังรูป



รูปที่ 2.2 การแทนระดับสีด้วยเลขฐาน 2

ถ้าภาพเป็นแบบโทนสีเทาแต่ละจุดภาพก็จะถูกแทนที่ด้วยตัวเลขที่บอกถึงค่าสีตั้งแต่ 0-255 ดังรูปที่ 2.3 แต่ละจุดจะถูกแทนที่ด้วยตัวเลข ซึ่งตัวเลขเหล่านี้ก็อยู่ระหว่าง 0-255 คือตั้งแต่ 0,1,2,3,4,... 255 เป็นโทนสีเทา แต่ถ้าเป็นภาพขาวดำจะมีอยู่ด้วยกันแค่ 2 สีคือ สีดำ แทนด้วยเลข 0 กับสีขาวแทนด้วยเลข 1 เพราะฉะนั้นถ้าเป็นภาพขาวดำหนึ่งจุดภาพจะใช้พื้นที่เก็บข้อมูลเพียง 1 บิต เท่านั้น แต่ถ้าเป็นภาพในโทนสีเทานั้น ใน 1 จุดภาพจะใช้พื้นที่ในการ เก็บข้อมูล 8 บิต เพราะว่าค่าระดับสีเมื่อแปลงเป็นเลขฐานสองแล้วจะได้ 8 บิต



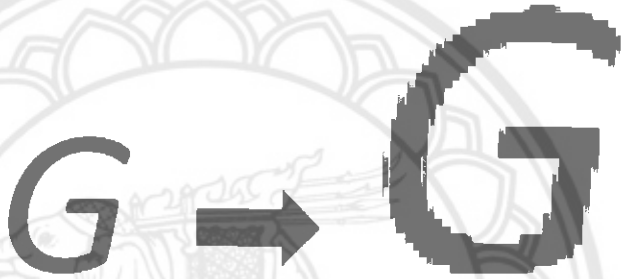
รูปที่ 2.3 การแทนแต่ละจุดของภาพด้วยเลขที่บอกถึงค่าสีตั้งแต่ 0-255

2.2 พื้นฐานเกี่ยวกับการประมวลผลภาพ

2.2.1 ชนิดของรูปภาพ โดยทั่วไปแบ่งออกเป็น 2 ชนิดคือ ภาพแบบบิตแมป (Bitmap) และภาพแบบเวกเตอร์ (Vector)

2.2.1.1 ภาพบิตแมป [3]

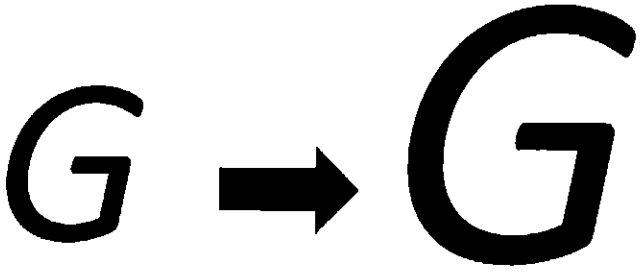
ภาพบิตแมป (Bitmap) เป็นภาพที่ประกอบด้วยจุดสีต่างๆที่มีความเข้มและสีแตกต่างกัน มีจำนวนคงที่ตายตัว มีลักษณะเป็นสี่เหลี่ยมเล็กๆเป็นช่องๆเหมือนตารางรวมๆกันทำให้เกิดภาพ มีข้อดีคือเหมาะกับภาพที่ต้องการระบายสี สร้างสี กำหนดสีที่ต้องการความละเอียดได้ง่าย ข้อเสีย คือ การขยายขนาดของภาพจะทำให้จุดโตขึ้นและเห็นเป็นรอยหยักๆ ถ้าเป็นเส้นโค้งก็จะยิ่งเห็นได้ชัด ไฟล์ภาพพวกนี้มีหลายรูปแบบ อาทิ เช่น BMP, TIF, JPG, PCT ฯลฯ



รูปที่ 2.4 แสดงการขยายของภาพบิตแมป

2.2.1.2 ภาพเวกเตอร์

ภาพเวกเตอร์ (Vector) เป็นภาพที่สร้างด้วยส่วนประกอบของเส้นลักษณะต่างๆ เป็นภาพที่เกิดจากการกำหนดพิกัดและการคำนวณค่าบนระนาบสองมิติ รวมทั้งมุมและระยะทาง ตามทฤษฎีเวกเตอร์ในทางคณิตศาสตร์ ในการก่อให้เกิดเป็นเส้น หรือรูปภาพ ข้อดีคือ สามารถย่อขยายได้ โดยคุณภาพไม่เปลี่ยนแปลง ข้อเสียคือภาพไม่เหมือนภาพจริงเป็นได้เพียงภาพวาดหรือใกล้เคียง ภาพถ่ายเท่านั้น ข้อมูลภาพพวกนี้ได้แก่ไฟล์สกุล svg, ps, eps ฯลฯ

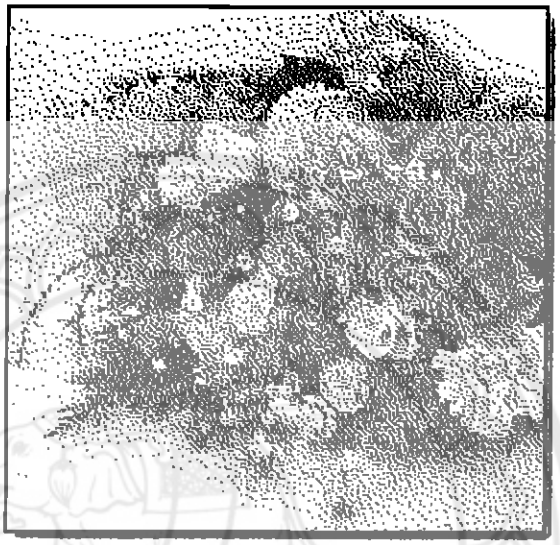


รูปที่ 2.5 แสดงการขยายของภาพเวกเตอร์

2.2.2 โหมดสีต่างๆ [5]

2.2.2.1 ภาพบิตแมป

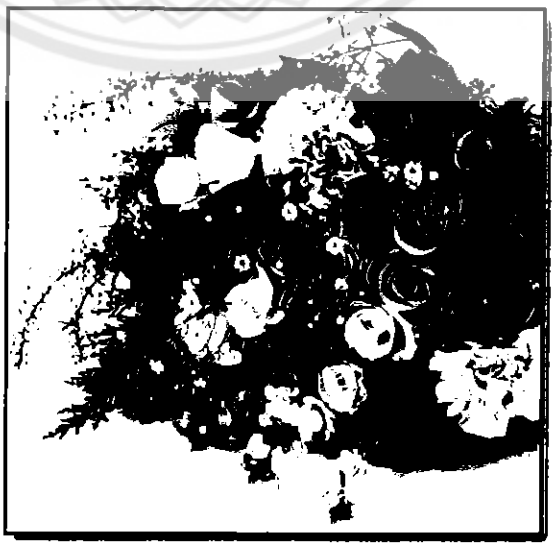
ภาพบิตแมป (Bitmap) เป็นโหมดสีที่มีการเก็บข้อมูลสีเพียง 1 บิต ต่อพิกเซล ซึ่งจะทำให้รูปในโหมดนี้มีเพียง สีขาวหรือดำเท่านั้น และไม่สามารถไล่เฉดสีได้ทำให้ภาพหยาบมากและไม่สามารถตกแต่งใด ๆ ได้ ข้อดีของโหมดภาพแบบนี้คือ ภาพที่ได้จากโหมดนี้จะมีขนาดที่เล็กมาก สามารถใช้สร้างภาพลายเส้น หรือโลโกที่ไม่ต้องการสีสันได้



รูปที่ 2.6 แสดงภาพโหมดสีแบบบิตแมป

2.2.2.2 ภาพระดับเทา

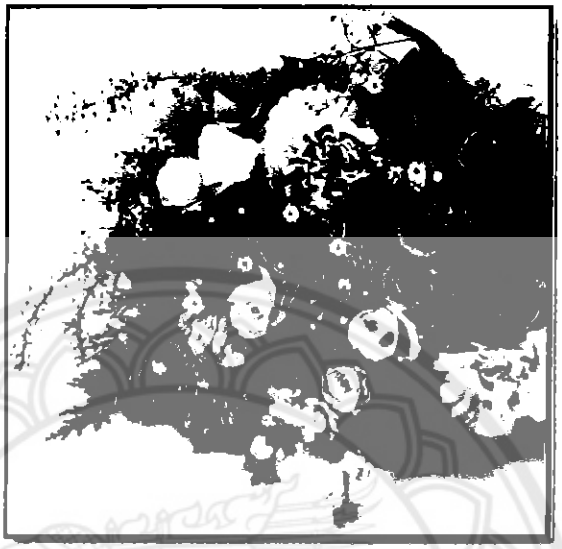
ภาพระดับเทา (Grayscale) เป็นโหมดสำหรับภาพขาวดำ สามารถไล่เฉดสีได้ทำให้ภาพมีความคมชัดกว่าภาพบิตแมปมาก สามารถใช้กับเครื่องมือใน โปรแกรมโฟโต้ชอป (Photoshop) ได้แทบทุกตัว



รูปที่ 2.7 แสดงภาพโหมดสีระดับเทา

2.2.2.3 ภาพ 2 โทนสี

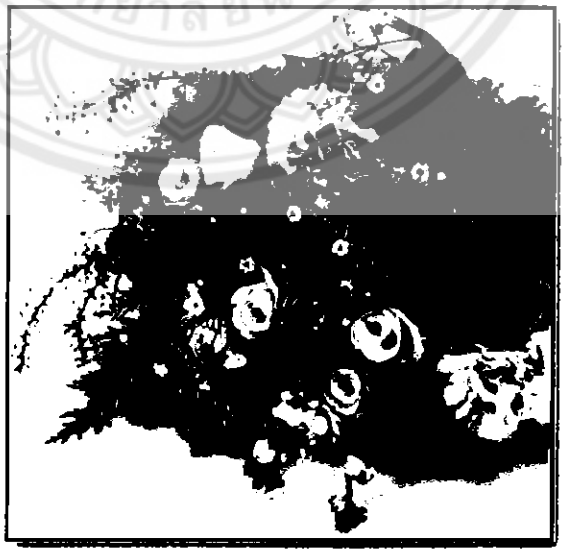
ภาพ 2 โทนสี (Duotone) เป็นโหมดสีที่สามารถปรับความคมชัดและเฉดสีของภาพระดับเทาให้มีความหลากหลายมากขึ้น สามารถใช้สีอื่น ๆ เข้ามาเสริมในสีดำ ทำให้ภาพมีความน่าสนใจขึ้น



รูปที่ 2.8 แสดงภาพโหมดสีแบบ 2 โทนสี

2.2.2.4 ภาพสีอินเด็กซ์

ภาพสี อินเด็กซ์ (Indexed Color) เป็นโหมดสีที่จำกัดไว้เพียง 256 สี โดยโปรแกรมโฟโต้ชอปจะปรับสีให้ใกล้เคียงกับสีที่กำหนดไว้ทั้ง 256 สีที่ถูกกำหนดไว้แล้ว ซึ่งจะทำให้ขนาดของภาพไม่ใหญ่มาก และยังคงคุณสมบัติของภาพไว้อย่างครบถ้วน



รูปที่ 2.9 แสดงภาพโหมดสีแบบภาพสีอินเด็กซ์

2.2.2.5 ภาพสี อาร์จีบี

ภาพสี อาร์จีบี (RGB Color) โหมดสีที่ถอดคุณสมบัติของภาพแบบอาร์จีบี มาสร้างเห็นโหมดภาพ โดยมีสี แดง เขียว และน้ำเงิน โดยแต่ละสีจะไล่ได้ 256 ระดับ โดยใช้หลักการการรวมแสงสี ซึ่งสามารถสร้างสีได้สูงสุด 16.7 ล้านสี หลักการแสดงสีของ จอคอมพิวเตอร์นั้นจะแสดงเป็นอาร์จีบี อยู่แล้ว ฉะนั้น ไม่ว่าจะเลือกโหมดการทำงานใดก็ตาม การแสงดผลบนจอภาพก็ จะใช้เป็นอาร์จีบี อยู่เช่นเดิม

รูปที่ 2.10 แสดงภาพโหมดสีแบบอาร์จีบี

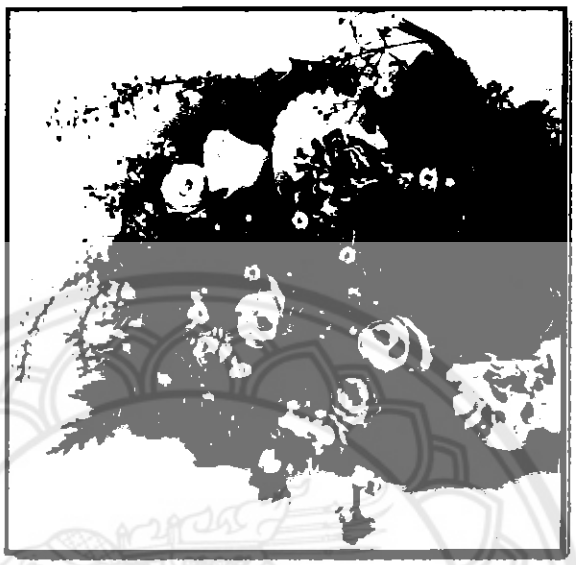
2.2.2.6 ภาพสี ซีเอ็มวายเค

ภาพสี ซีเอ็มวายเค (CMYK) คือโหมดมาตรฐานสำหรับเครื่องพิมพ์ โดยแบ่งสีเป็น 4 สี หลักได้แก่ ฟ้า ชมพูม่วง เหลือง และดำ โดยในแต่ละสีจะมีค่า 8 บิตซึ่งทำให้ในแต่ละพิกเซล จะเก็บค่าถึง 32 บิตในโหมดนี้โฟโต้ชอปจัดเตรียมสำหรับภาพที่ใช้ในการพิมพ์ โดยแก้ไขจุดบกพร่องของโหมดสีอาร์จีบีที่เครื่องพิมพ์ไม่สามารถพิมพ์สีบางสีออกไปได้

รูปที่ 2.11 แสดงภาพโหมดสีแบบซีเอ็มวายเค

2.2.2.7 แล็บคัลเลอร์

แล็บคัลเลอร์ (Lab Color) เป็นโหมดสีที่มีความเหมือน จริงที่สุด สำหรับงานพิมพ์ที่เป็นโฟโต้ ซีดี (Photo CD) หรือภาพที่ต้องการใช้งานระหว่าง ระบบคอมพิวเตอร์ที่แตกต่างกัน เช่น วินโดว์ (Windows) ไปยังระบบแมค (Mac)



รูปที่ 2.12 แสดงภาพโหมดสีแบบแล็บคัลเลอร์

2.2.2.8 โหมดสีแบบ 8 บิตต่อ 1 พิกเซล

โหมดสีแบบ 8 บิตต่อ 1 พิกเซล (Multichannel) การเก็บสีจำนวน 8 บิตต่อพิกเซล ทำให้มีความจำกัดของจำนวนสี ซึ่งใจใช้กับการพิมพ์แบบพิเศษที่ไม่ต้องการความละเอียด และสีที่ซับซ้อนมากนัก

รูปที่ 2.13 แสดงภาพโหมดสีแบบ 8 บิตต่อ 1 พิกเซล

2.3 การหาค่าขีดแบ่ง

การหาค่าขีดแบ่งหรือการทำเทรชโฮลด์ (Threshold) เป็นเทคนิคการประมวลผลภาพอย่างง่ายเพื่อที่จะแบ่งแยกส่วนพื้นหน้า (Foreground) หรือวัตถุออกจากพื้นหลัง (Background) โดยใช้ค่าระดับความเข้มคงที่ค่าหนึ่งเป็นตัวกำหนดในการแยกแยะ เพื่อให้ภาพผลลัพธ์ที่ได้เป็นภาพ (Binary) ที่มีค่าระดับความเข้มเทา เพียง 2 ระดับเท่านั้นคือขาวและดำ แต่เนื่องจากความสว่างของแสงภายในรูปมีไม่เท่ากันการกำหนดเทรชโชลด์ เพียงค่าเดียวจึงทำให้รูปที่ได้เกิดความผิดพลาดได้ ทางเลือกที่ดีกว่าคือการเปลี่ยนมาใช้วิธีการของโอซึ (Otsu's Thresholding Method) ซึ่งเป็นอัลกอริทึมที่ฉลาดกว่าเพื่อปรับแสงของฉาพให้มีความเหมาะสมที่สุดต่อการนำไปใช้ต่อ

วิธีการของโอซึ เป็นการคำนวณหาค่าขีดแบ่งจากการกระจายตัวของความถี่ที่ทับซ้อนกัน (Intraclass variance) น้อยที่สุด ซึ่งวิธีนี้จะต้องทำการแบ่งกราฟแสดงความถี่ออกเป็น 2 ส่วนเพื่อคำนวณหาความแปรปรวน

สมการการทำเทรชโฮลด์ด้วยวิธี โอซึ สามารถเขียนได้คือ

กำหนดให้น้ำหนักของผลรวมของความแปรปรวนของระดับสีทั้ง 2 ส่วน

$$\sigma_{\omega}^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t)$$

เมื่อ ω_1 , ω_2 คือความน่าจะเป็นของระดับสีส่วนที่ 1 และ 2 ซึ่งแบ่งโดยค่าขีดแบ่ง (t) และ σ_1^2, σ_2^2 คือค่าความแปรปรวนของส่วนที่ 1 และ 2 ตามลำดับ

วิธีการของโอซึ จะหาค่าต่ำสุดของการกระจายตัวของความถี่ที่ซ้อนทับกัน โดยสามารถคำนวณได้จากสมการด้านล่าง

$$\sigma_b^2(t) = \sigma^2 - \sigma_{\omega}^2(t) = \omega_1(t)\omega_2(t)\,[\mu_1(t) - \mu_2(t)]^2$$

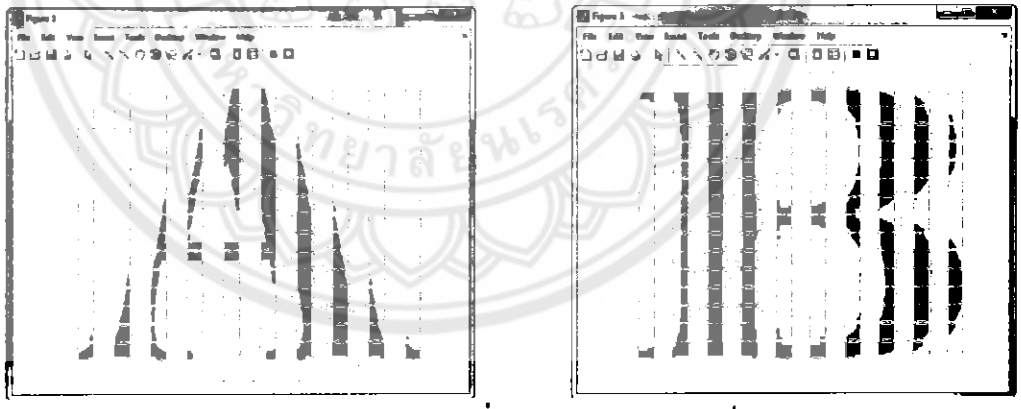
มีขั้นตอนการทำงานดังนี้

1. คำนวณค่าความน่าจะเป็นของแต่ละระดับสีในกราฟแสดงค่าความถี่
2. กำหนดค่าเริ่มต้นของ $\omega_i(0)$ และ $\mu_i(0)$
3. คำนวณค่าขีดแบ่ง i ตั้ง แต่ 1 จนถึงระดับสีสูงสุด
   - 3.1 ปรับค่า ω_i , μ_i
   - 3.2 คำนวณค่า $\sigma_b^2(t)$
4. หาค่า i ที่ทำให้ $\sigma_b^2(t)$ มีค่าสูงที่สุด

รูปที่ 2.14 ก) ภาพต้นฉบับ ข) ภาพหลังจากการแยกด้วยค่าขีดแบ่งที่คำนวณได้จากวิธีการของโอซึ

2.4 การแบ่งภาพออกเป็นส่วนๆ [7]

ภาพหนึ่งภาพเราสามารถตัดออกมาเป็นส่วนๆได้ ซึ่งในที่นี้เราจะเรียกส่วนที่ตัดออกมาว่าบล็อก(Block) [5] เราแบ่งภาพออกเป็นส่วนๆ เพื่อที่จะได้นำแต่ละส่วนไปวิเคราะห์ เปรียบเทียบ เช่น ดังรูปด้านล่างในแต่ละบล็อกจะประกอบไปด้วยบล็อกที่เป็นสีขาวทั้งหมด บล็อกที่เป็นสีดำทั้งหมด และ บล็อกที่ประกอบไปด้วยสีขาวและสีดำปนกัน ซึ่งในแต่ละบล็อกเราจะทำการเก็บค่าของสีดำ โดยที่ถ้าพิกเซลนั้นมีสีดำ 1 เราก็จะให้ค่าแก่พิกเซลนั้นเป็น 1 แต่ถ้าพิกเซลนั้นเป็นสี ขาวเราก็จะให้ค่าพิกเซลนั้นเป็น 0 จากนั้นก็จะรวมค่าคะแนนในแต่ละบล็อกนำไปวิเคราะห์ในขั้นตอนต่อไป



รูป 2.15 แสดงการเปลี่ยนแปลงของแต่ละบล็อก

หลักของการแบ่งบล็อกนั้น ก็ขึ้นอยู่กับความต้องการของผู้เขียนโปรแกรมว่าต้องการกี่บล็อก ขนาดเล็ก ใหญ่เท่าไหร่ จากตัวอย่างในรูป 2.15 แบ่งภาพออกเป็น 15*10 = 150 บล็อก

2.5 การจำแนกตัวอักษรออกจากบรรทัดข้อความ [8]

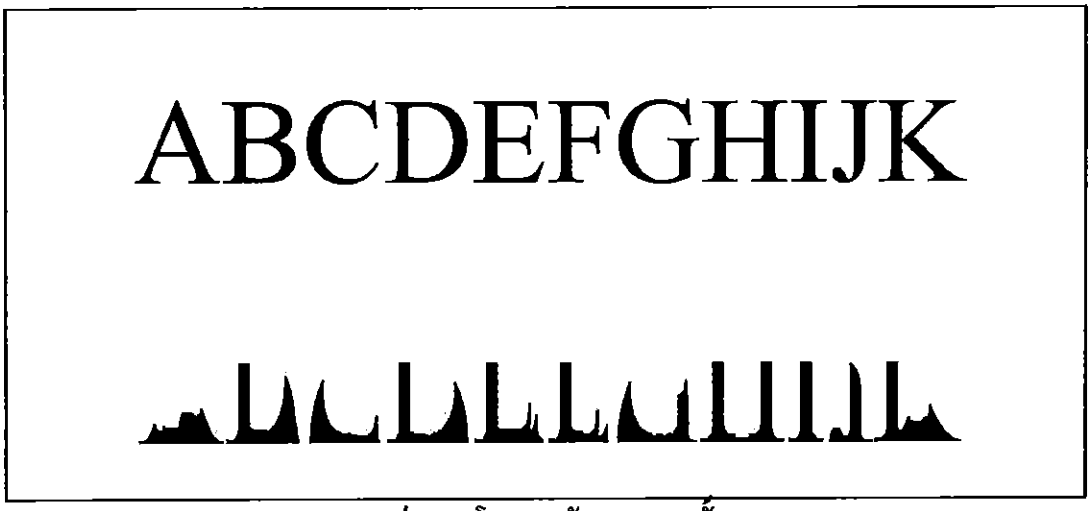
โพรเจกชัน (Projection) บนเส้นตรงสามารถหาได้โดยแบ่งเส้นตรงออกเป็นช่องๆ และหาจำนวนของจุดภาพที่มีค่าเท่ากับ 1 ซึ่งอยู่ในแนวตั้งฉากกับเส้นตรงในแต่ละช่องนั้น การโปรเจกชันเป็นการแสดงข้อมูลของภาพที่มีประโยชน์อย่างยิ่ง อย่างไรก็ตาม โพรเจกชัน มีขอเสียอันเนื่องมากจากเป็นข้อมูลที่ไม่เป็น เอกลักษณ์ของภาพมากกว่าหนึ่งภาพอาจได้ผลลัพธ์ในการ โพรเจกชันเหมือนกัน การทำโพรเจกชันในแนวนอนและแนวตั้งหาได้อย่างง่ายๆ จากการนับจำนวนจุดภาพที่มีค่าเท่ากับ 1 ในทิศทางแนวตั้งของช่องแต่ละช่องในเส้นตรงที่แบ่งไว้ตามลำดับ ดังแสดงไว้ในรูป 2.16 จุดภาพที่มีสีดำมีค่าเท่ากับ 1 และจุดภาพขาวที่มีค่าเท่ากับ 0

รูป 2.16 ตัวอย่างเช่นการใช้การ โพรเจกชันเพื่อใช้แบ่งตัวอักษรแต่ละตัวออกจากกัน

การจำแนกบรรทัดข้อความ และจำแนกตัวอักษรออกจากบรรทัดข้อความ

ในขั้นตอนนี้เป็นขั้นตอนของการตัดตัวอักษรลายมือเขียนภาษาอังกฤษ จากเอกสารภาพเชิงดิจิทัล เมื่อได้ภาพที่มีความชัดเจนจากการกำจัดสัญญาณรบกวนแล้ว จะนำรูปภาพเข้าสู่ขั้นตอนการจำแนกบรรทัดข้อความและจำแนกตัวอักษรออกจากบรรทัดข้อความ โดยใช้สามารถใช้ วิธีการโพรเจกชันเป็นวิธีการหาค่าสมดุลของ จุดดำที่ประกอบกันเป็นตัวอีกษร ซึ่งวิธีการ โพรเจกชัน นั้นสามารถแบ่งออกเป็นสองวิธีคือ

1. วิธีการโพรเจกชันตามแนวตั้ง เป็นวิธีการที่ สามารถแยกตัวอักษรออกแต่ละตัวออกจากกัน ได้โดยคำนวณจากจุดสีดำหรือจุดพิกเซลที่มีค่าเป็น 1 ตามแนวแกน Y ซึ่งก็หมายถึงส่วนที่เป็นตัวอักษรตามแนวตั้งนั่นเอง ผลจะได้ออกมาในรูปของกราฟ ฮิสโทแกรม (Histogram) ช่วงของพิกเซลที่ไม่มีตัวอักษรนั้นจะเกิดช่องว่างขึ้น จากความแตกต่างนี้จึงสามารถนำมาใช้วิเคราะห์หาจุดเริ่มต้นและจุดสิ้นสุดของตัวอักษรจากด้านซ้ายไปด้านขวาได้
2. วิธีการโพรเจกชันตามแนวนอนเป็นวิธีการที่สามารถแยกตัวอักษรจากจุดบนสุดไปถึงจุดล่างสุดได้โดยวิธีการคำนวณก็เหมือนกันกับการ โพรเจกชันตามแนวตั้ง เพียงแค่เปลี่ยนจากกาคำนวณหาค่าจุดสีดำตามแนวแกน Y ไปเป็นการคำนวณหาค่าจุดสีดำตามแนวแกน X ผลที่ได้ออกมาก็จะอยู่ในรูปของฮิสโตแกรม เช่นกัน และจะใช้ความแตกต่างของช่องว่างที่เกิดขึ้นจากช่วงที่ไม่มีตัวอักษรในการวิเคราะห์หาจุดเริ่มต้นและจุดสิ้นสุดของตัวอักษรแต่ละตัวได้



รูป 2.17 โพรเจกชันตามแนวตั้ง

รูป 2.18 โพรเจกชันตามแนวนอน

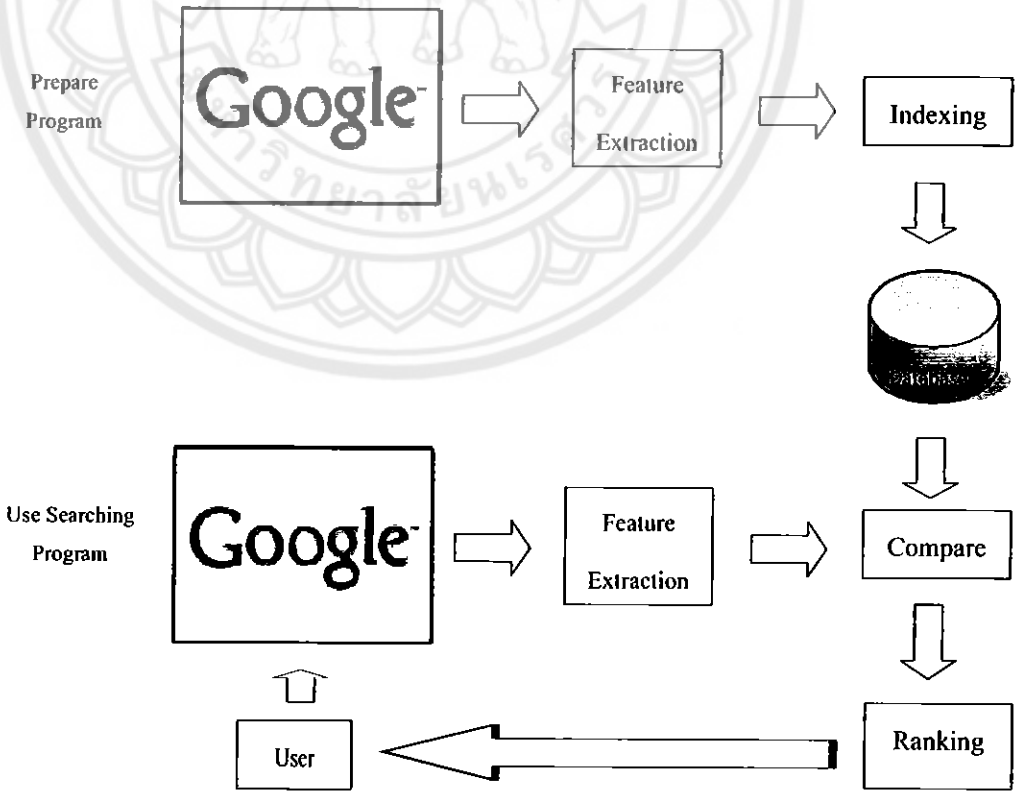
บทที่ 3

วิธีการดำเนินโครงการ

ขั้นตอนและวิธีการดำเนินงานนั้น มีขั้นตอนหลักๆดังนี้ คือ ตัดภาพตัวอักษรออกจากกัน การแยกภาพตัวอักษรออกเป็นส่วนๆ การระบุโค้ดจากจำนวนของพิกเซลสีดำในแต่ละส่วน การเปรียบเทียบหาผลต่างของโค้ดจากรูปที่นำมาเป็นตัวอย่างกับรู ปในฐานข้อมูล ทำการแสดงชื่อแบบอักษรที่มีผลต่างน้อยที่สุด ซึ่งแสดงว่ารูปนั้นใกล้เคียงกับรูปที่นำมาเป็นตัวอย่างมากที่สุด

3.1 ภาพรวมของระบบ

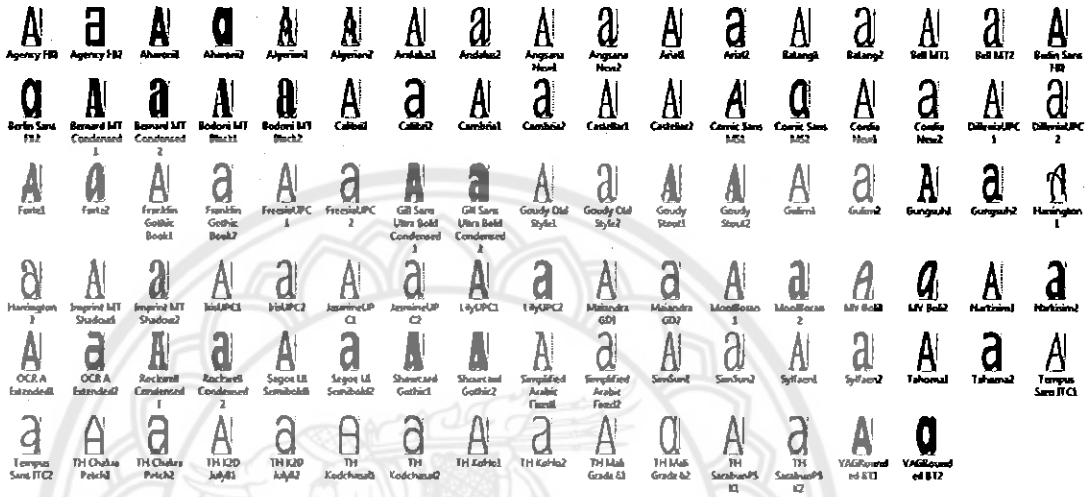
ในการทำงานของระบบสืบค้นตัวอักษร โดยใช้ตัวอย่าง จะต้องใช้หลักการวิเคราะห์ภาพ เพื่อนำไปแยกความแตกต่างระหว่างตัวอักษรและตัวเลขแต่ละตัวได้ว่า เป็นแบบอักษรลักษณะใด โดยจะแบ่งการทำงานเป็นการทำในส่วนของการเก็บลักษณะของตัวอักษรกับ การทำงานในส่วนของผู้ใช้ คือ การใช้โปรแกรม ค้นหาลักษณะตัวอักษร ซึ่งสามารถนำมาเขียนเป็นแผนภาพได้ดังนี้



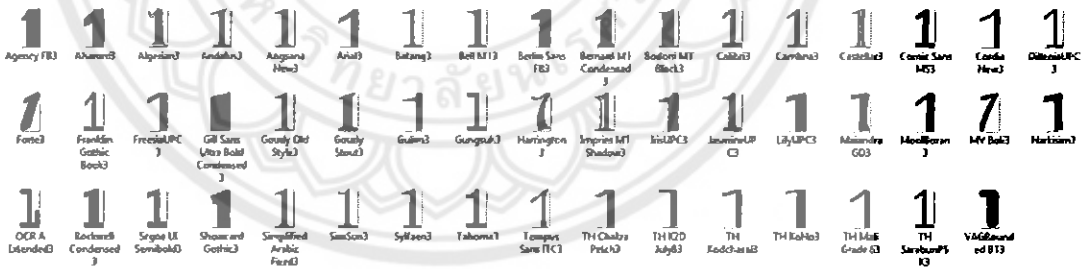
รูปที่ 3.1 การทำงานของโปรแกรม

3.2 ส่วนของการเก็บแบบอักษร

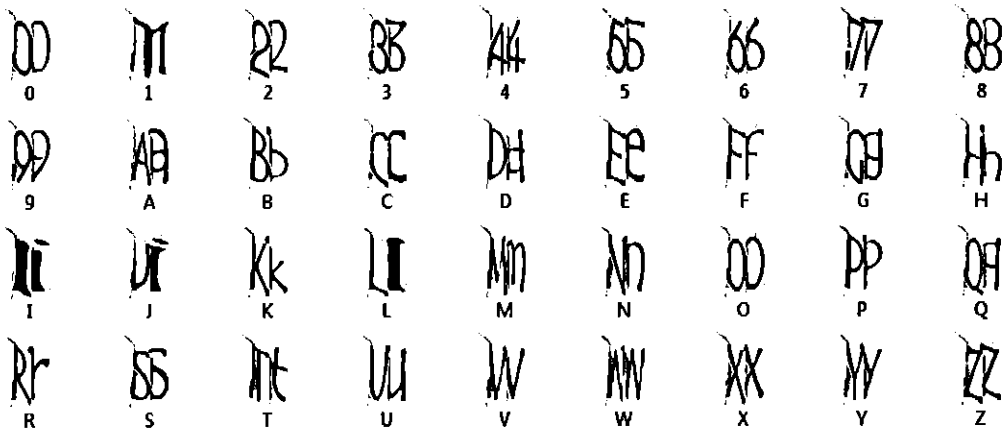
เริ่มต้นการทำงานจากการเก็บรูปแบบแบบอักษรเพื่อใช้เป็นตัวอย่างโดยผู้จัดทำโครงงานได้ใช้แบบอักษรในชื่อและลักษณะที่ต่างกันเพื่อเป็นตัวอย่าง 50 แบบ โดยแบ่งการเก็บตัวอักษรแต่ละตัวแยกออกเป็นโฟลเดอร์ ดังรูปที่ 3.2



รูปที่ 3.2 การเก็บแบบตัวอักษรในโฟลเดอร์



รูปที่ 3.3 การเก็บแบบตัวเลขในโฟลเดอร์



รูปที่ 3.4 การเก็บแบบอักษรและตัวเลขในโฟลเดอร์

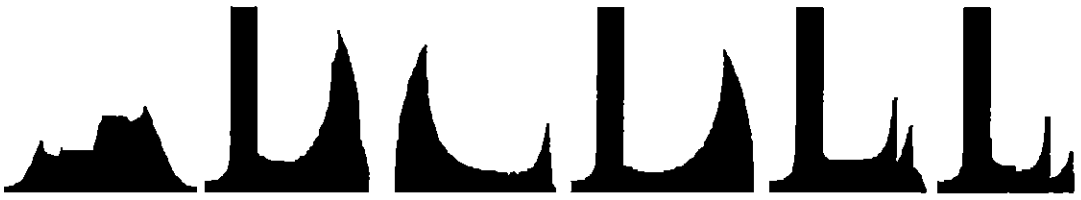
3.3 การฉายแสงในแนวตั้งและแนวนอน

ในขั้นตอนนี้จะ นำภาพตัวอักษรที่ ทราบชื่อมา ทำการฉายแสงในแนวตั้งก่อนแล้วตัดตัวอักษรออกจากกัน จากนั้นทำการนำตัวอักษรแต่ละตัวมาฉายแสงในแนวนอนเพื่อตัดพื้นที่ด้านบนและด้านล่างของตัวอักษรที่เป็นส่วนเกินออก

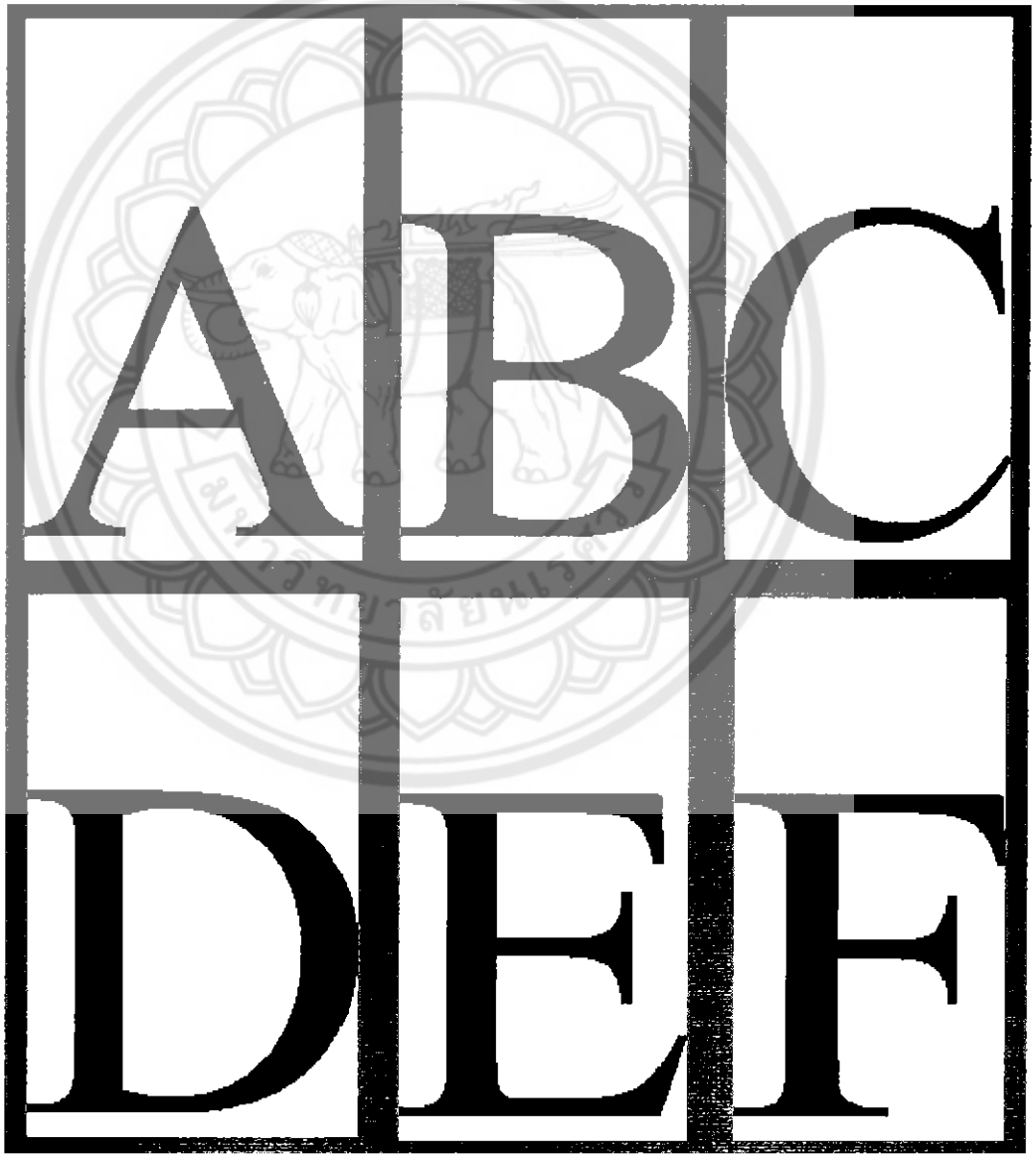
รูปที่ 3.5 รูปต้นฉบับที่จะนำมาทำการโปรเจคชั่น

3.3.1 การฉายแสงในแนวตั้ง

เราจะเริ่มกระบวนการตัดตัวอักษรออกจากกันด้วยการทำการฉายแสงในแนวตั้ง (Vertical Projection Profile) จะทำให้เราได้ฮิสโทแกรมดังรูปที่ 3.5 ซึ่งฮิสโทแกรมนี้จะทำให้เราทราบจุดเริ่มต้นและจุดสิ้นสุดในแนวแกน x ของแต่ละตัวอักษรได้ เราจึงทำการตัด (Crop) ตัวอักษร ได้ดังรูปที่ 3.6



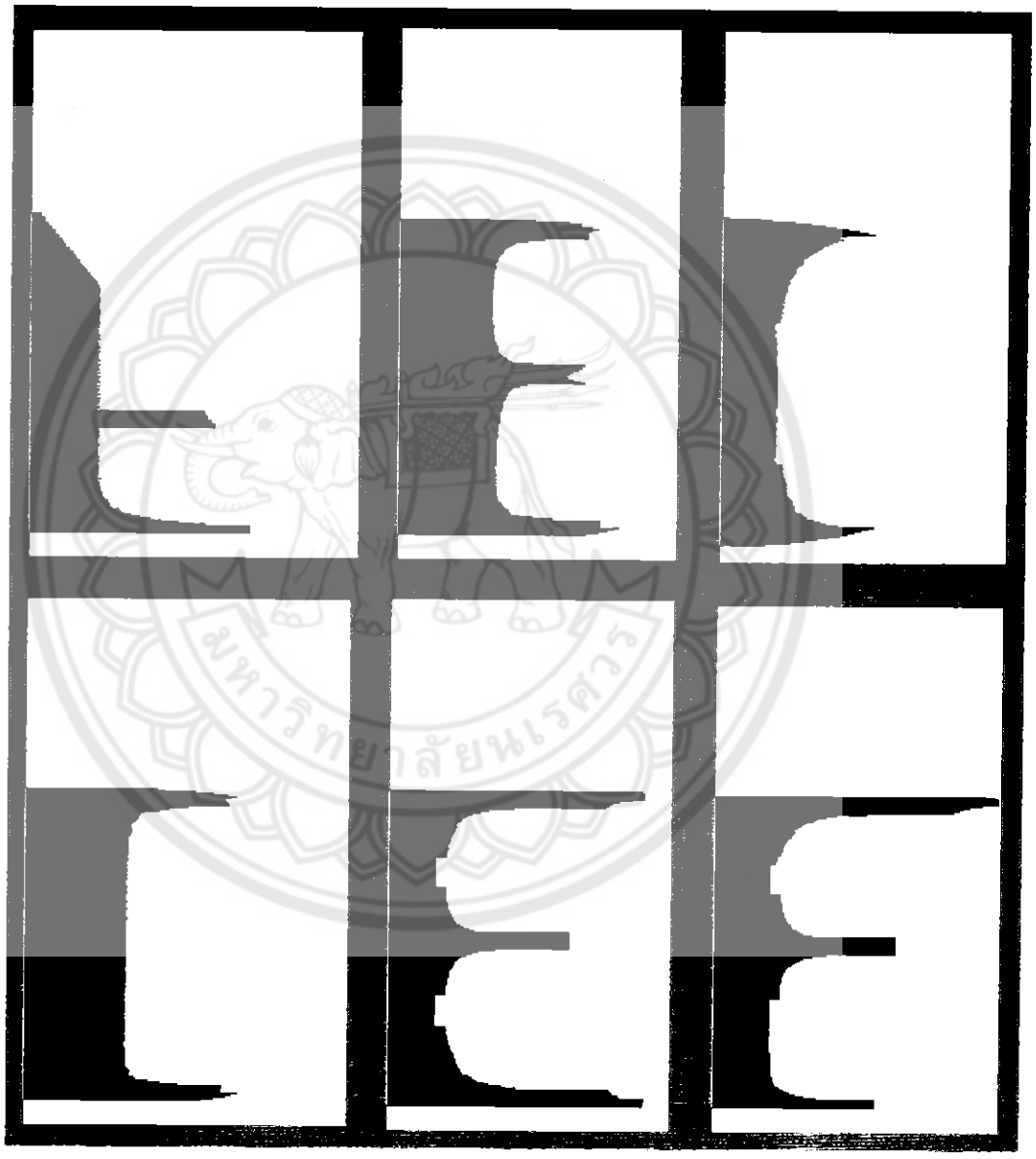
รูปที่ 3.6 ฮิสโทแกรมที่ได้จากการ ฉายแสงในแนวตั้ง



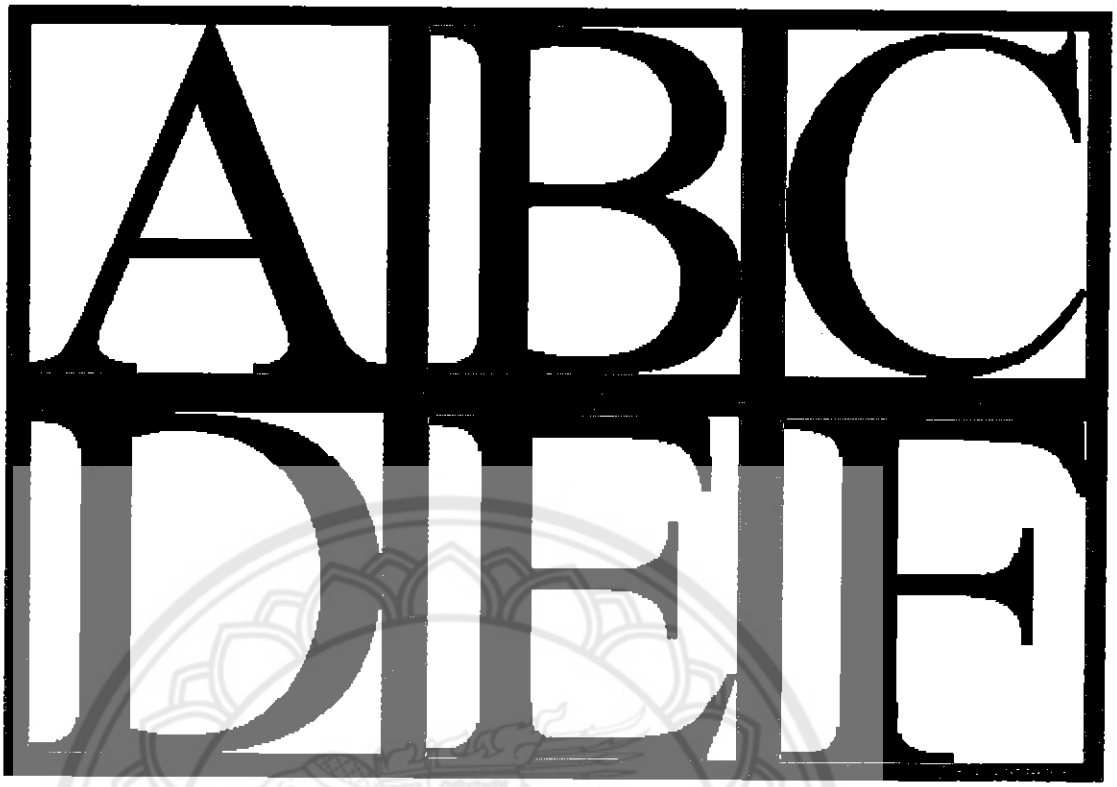
รูปที่ 3.7 ภาพที่ตัดได้จากการฉายแสงในแนวตั้ง

3.3.2 การฉายแสงในแนวนอน

ในการฉายแสงในแนวนอนเราจะนำรูปที่ได้จากการตัดในขั้นตอนการฉายแสงในแนวตั้งมาทำการฉายแสงในแนวนอน(Horizontal Projection Profile) ทีละรูป (เนื่องจากตัวอักษรแต่ละตัวมีความสูงไม่เท่ากัน) จะทำให้เราได้ฮิสโทแกรมของแต่ละตัวอักษรดังรูปที่ 3.7 ซึ่งทำให้เราทราบจุดสูงสุดและจุดต่ำสุดของแต่ละตัวอักษร ซึ่งทำให้เราตัดตัวอักษรได้ดังรูปที่ 3.8



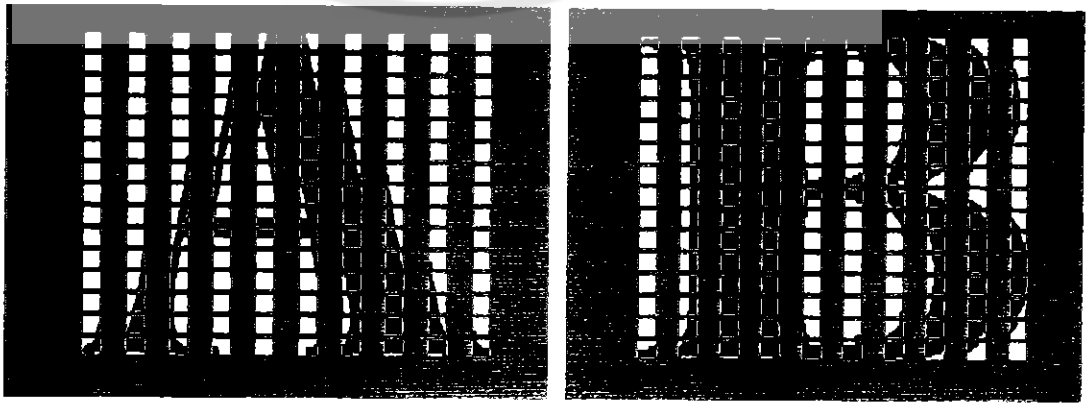
รูปที่ 3.8 ฮิสโทแกรมที่ได้จากการฉายแสงในแนวนอน



รูปที่ 3.9 ภาพที่ตัดได้จากการฉายแสงในแนวนอน

3.4 การแบ่งภาพออกเป็นบล็อก

เมื่อเราตัดตัวอักษรได้ดังรูปที่ 3.9 จะนำรูปมาทำการเปลี่ยนขนาดให้มีขนาด 225x120 พิกเซล จากนั้นนำมาทำการแบ่งภาพเป็นบล็อก (Block) บล็อกละ 15x15 พิกเซล จะได้จำนวน 15x8 บล็อก ดังรูปที่ 3.10 เพื่อแยกกันประมวลผลทีละบล็อก



รูปที่ 3.10 การแบ่งภาพตัวอักษรออกเป็นบล็อก

โดยแต่ละบล็อกจะทำการกำหนดเป็นโค้ด ดังรูปที่ 3.11 โดยนับจำนวนพิกเซลที่มีสีดำ โดยจะกำหนดโค้ดจากจำนวนพิกเซลสีดำ ดังตารางที่ 3.1

ตารางที่ 3.1 ตารางแสดงการกำหนดโค้ดจากจำนวนพิกเซล

จำนวนพิกเซลสีดำ(เปอร์เซ็นต์)	โค้ด
0.00-20.00	1
20.01-40.00	2
41.00-60.00	3
61.00-80.00	4
81.00-100.00	5

ก.รูปอินพุตที่นำมาสืบค้น ข.รูปที่เก็บไว้ในฐานข้อมูล

รูปที่ 3.11 การกำหนดโค้ดในแต่ละบล็อก

จากนั้นจะนำโค้ดแต่ละบล็อกไปเทียบกับโค้ดของบล็อกในตำแหน่งเดียวกันของรูปในฐานข้อมูลเพื่อหาผลต่าง ดังรูปที่ 3.12

0	0	1	3	2	0	0	0
0	0	1	2	2	0	0	0
0	0	1	0	1	1	0	0
0	0	1	0	1	2	0	0
0	0	2	0	1	2	0	0
0	0	3	1	1	2	0	0
0	1	3	1	2	2	0	0
0	1	3	0	2	1	1	0
0	2	1	1	3	1	2	0
0	2	2	1	1	0	2	0
0	3	4	4	3	0	3	0
1	3	2	2	2	2	2	1
2	2	0	0	0	2	1	1
1	0	0	0	0	2	0	1
1	0	1	0	2	4	0	1

รูปที่ 3.12 รูปแสดงการหาผลต่างในแต่ละบล็อก

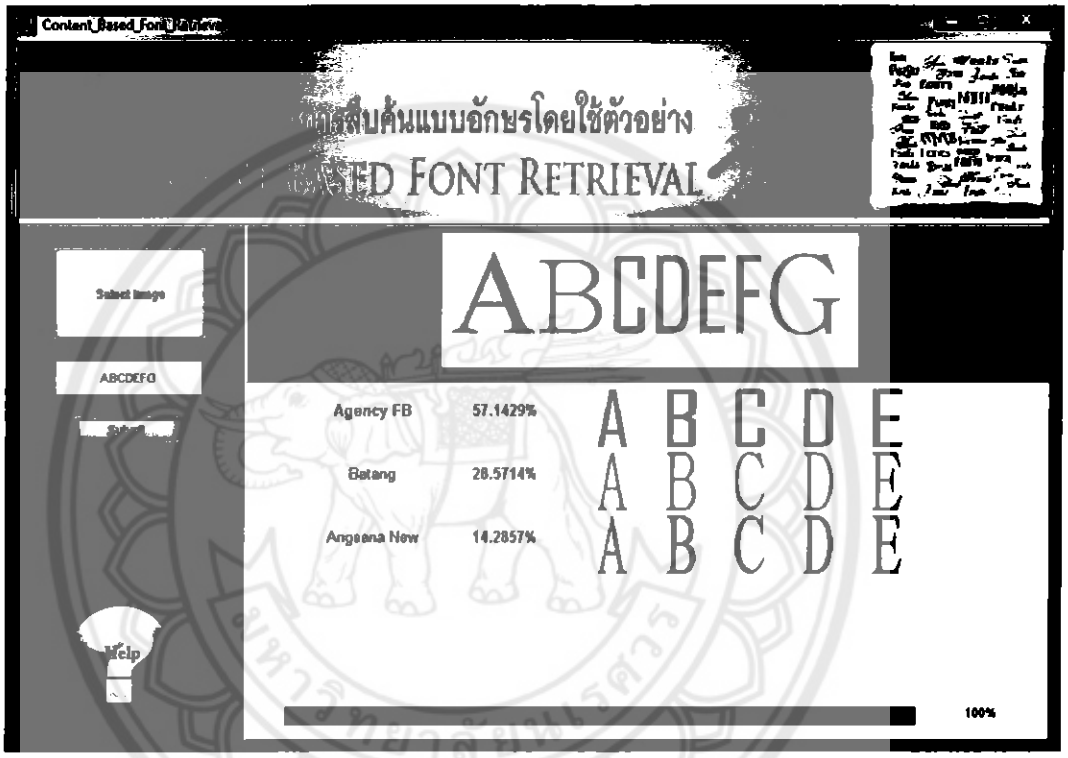
จากนั้นทำการหาผลรวมของผลต่างจากทุกบล็อกในรูป โดย จากรูปที่ 3.12 สามารถเขียนสมการผลรวมได้เป็น

$$\sum |f(x,y) - g(x,y)|$$

ซึ่งจากรูปที่ 3.12 สามารถหาผลรวมได้เท่ากับ 118 จากนั้นนำรูปอินพุตนี้ไปเทียบกับรูปอื่นๆ ในฐานข้อมูลเพื่อหาผลรวมของผลต่าง จากนั้นทำการเปรียบเทียบว่ารูปใดในฐานข้อมูลทำให้ผลรวมของผลต่างมีค่าน้อยที่สุด ซึ่งแสดงว่ารูปนั้นมีความแตกต่างกับรูปที่นำมาสืบค้นน้อยที่สุด จึงสามารถระบุได้ว่าแบบอักษรนั้นคือแบบอักษรที่เรานำมาสืบค้น

3.5 การสรุปผลและการคำนวณ

จากผลรวมของผลต่างในแต่ละบล็อกของแต่แต่ละรูป เราจะเก็บชื่อ แบบอักษร ของรูปที่มีผลรวมของผลต่างน้อยที่สุด ซึ่งจะหมายความว่ารูปนั้นเหมือนรูปที่เรานำมาเป็นตัวอย่างมากที่สุด และเราจะนับจำนวนของชื่อ แบบอักษร ไว้ หากชื่อ แบบอักษร ใดมีจำนวนซ้ำกันมากที่สุดก็จะมีเปอร์เซ็นต์ที่จะเป็นแบบอักษรนั้นมากที่สุดดังรูปที่ 3.12



รูปที่ 3.13 รูปแสดงผลการทำงานของโปรแกรม

จากรูปที่ 3.12 จะพบว่ารูปที่เรานำมาเป็นตัวอย่างมีตัวอักษรทั้งหมด 7 ตัว และมีจำนวน 4 ตัวที่มีความคล้ายคลึง แบบอักษร Agency FB มากที่สุด มี 2 ตัวที่คล้ายคลึง แบบอักษร Batang มากที่สุด และมี 1 ตัวที่คล้ายคลึง แบบอักษร Angsana New มากที่สุด ดังนั้นเปอร์เซ็นต์ของ แบบอักษร Agency FB, Batang และ Angsana New จึงคำนวณได้ตามลำดับดังนี้

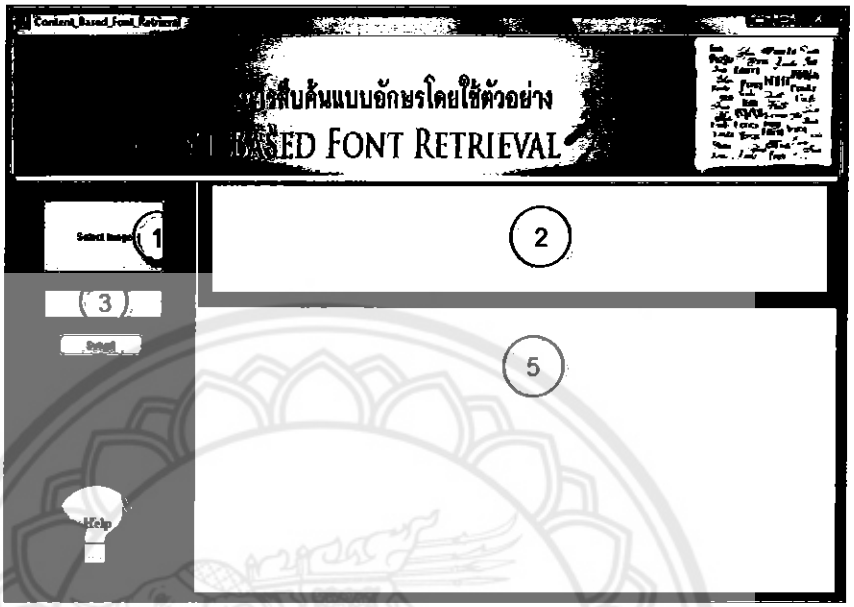
Agency FB = (4/7)*100 = 57.1429%

Batang = (2/7)*100 = 28.5714%

Angsana New = (1/7)*100 = 14.2857%

3.6 การออกแบบหน้าต่างโปรแกรมส่วนติดต่อกับผู้ใช้

หน้าต่างโปรแกรมส่วนติดต่อกับผู้ใช้ (Graphic User Interface) แบ่งออกเป็น 6 ส่วน ดังนี้



รูปที่ 3.14 รูปแสดงหน้าต่างโปรแกรมส่วนติดต่อกับผู้ใช้

1. ปุ่มกดเพื่อเลือกรูปภาพที่ผู้ใช้ต้องการ

เมื่อผู้ใช้ต้องการจะเริ่มการค้นหาแบบอักษร จะกดปุ่มนี้เพื่อเลือกรูปภาพที่ต้องการ โดยเมื่อกดปุ่มนี้จะมีหน้าต่างขึ้นมาให้ผู้ใช้เลือกรูปภาพที่ต้องการ

2. หน้าต่างแสดงรูปภาพที่ผู้ใช้เลือกมา

เมื่อผู้ใช้เลือกรูปภาพที่ต้องการมาแล้ว รูปภาพนั้นจะแสดงขึ้นมาที่หน้าต่างนี้ ทำให้ผู้ใช้สามารถตรวจสอบได้ว่าเลือกรูปภาพมาถูกหรือไม่

3. ช่องสำหรับพิมพ์ตัวอักษรตามรูปภาพที่เลือกมา

เมื่อเลือกรูปภาพมาแล้ว ขั้นตอนต่อไปคือ ผู้ใช้จะต้องพิมพ์ตัวอักษรตามรูปภาพที่เลือกมาโดยสามารถดูได้จากหน้าต่างแสดงรูปภาพ

4. ปุ่มกดเพื่อยืนยันตัวอักษรที่ผู้ใช้พิมพ์ และสั่งเริ่มการทำงานของโปรแกรม

เมื่อผู้ใช้พิมพ์ตัวอักษรครบถ้วน และแน่ ใจว่าถูกต้องแล้ว ก็จะกดปุ่มนี้เพื่อเริ่มการทำงานของโปรแกรม

5. หน้าต่างแสดงผลการทำงานของโปรแกรม

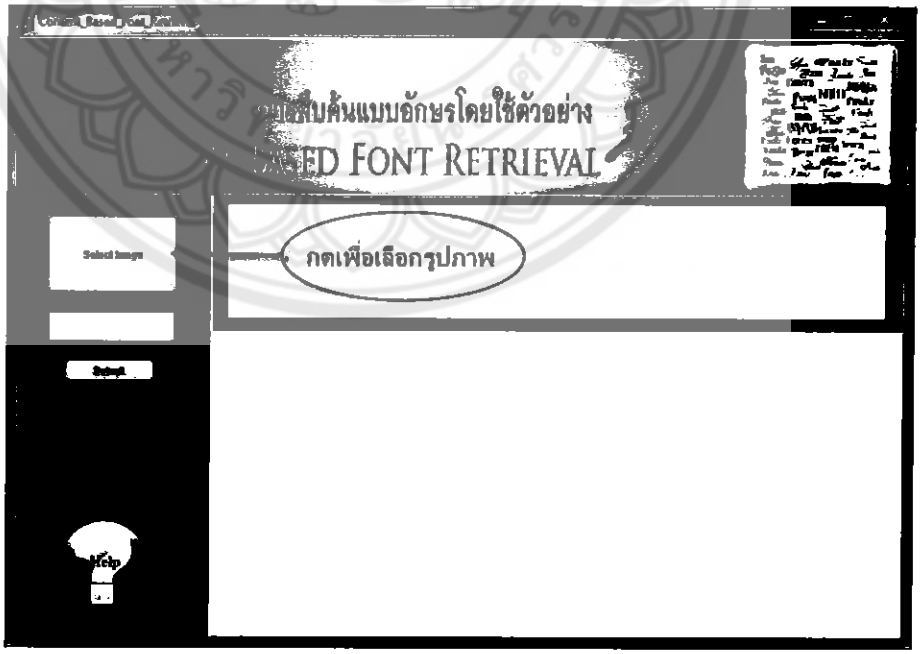
เมื่อโปรแกรมทำงานเสร็จจะแสดงผลการทำงานที่หน้าต่างนี้ โดยจะแสดงชื่อของแบบอักษรที่ใกล้เคียงกับรูปที่นำมาค้นหามากที่สุด ไล่ลำดับลงไปเรื่อย และแสดงรูปแบบอักษรตัวอย่างของชื่อแบบอักษรนั้น

6. ปุ่มกดเพื่อแสดงหน้าต่างแสดงวิธีการใช้งานโปรแกรม

ปุ่มกดนี้เมื่อกดแล้วจะแสดงหน้าต่างแสดงวิธีการใช้งานโปรแกรม โดยจะบอกอย่างละเอียดว่าขั้นตอนการใช้งานโปรแกรมมีอะไรบ้าง

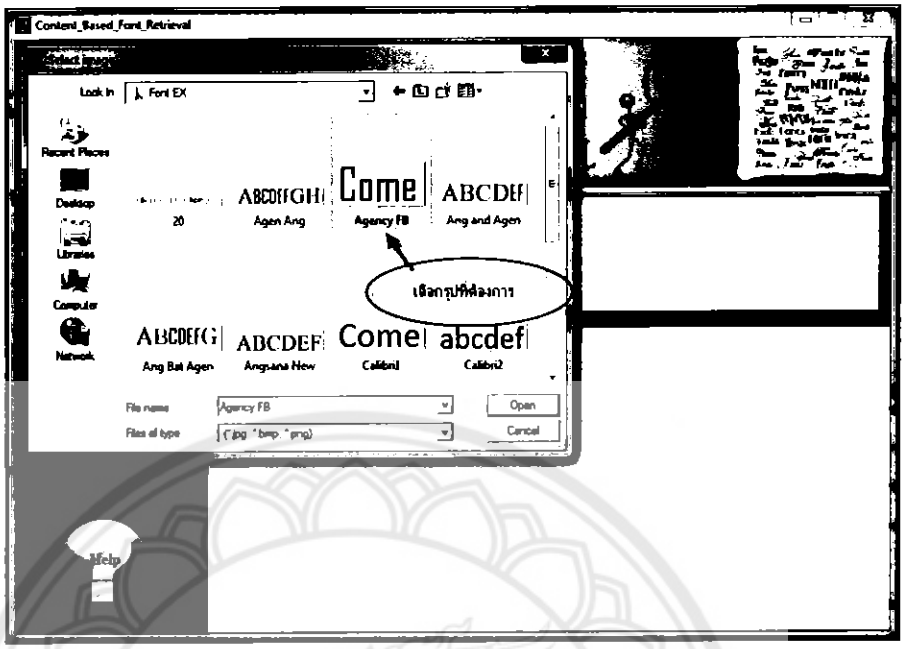
3.6.1 ลำดับขั้นตอนการทำงานของโปรแกรม

1.เริ่มต้นการใช้งานโดยผู้ใช้เลือกที่ปุ่ม Select Image



รูปที่ 3.15 รูปแสดงปุ่มเลือกรูปภาพ

2. เลือกรูปที่เราต้องการค้นหาจากรูป เมื่อเลือกแล้วรูปจะได้รูปที่เลือกมาแสดงผล

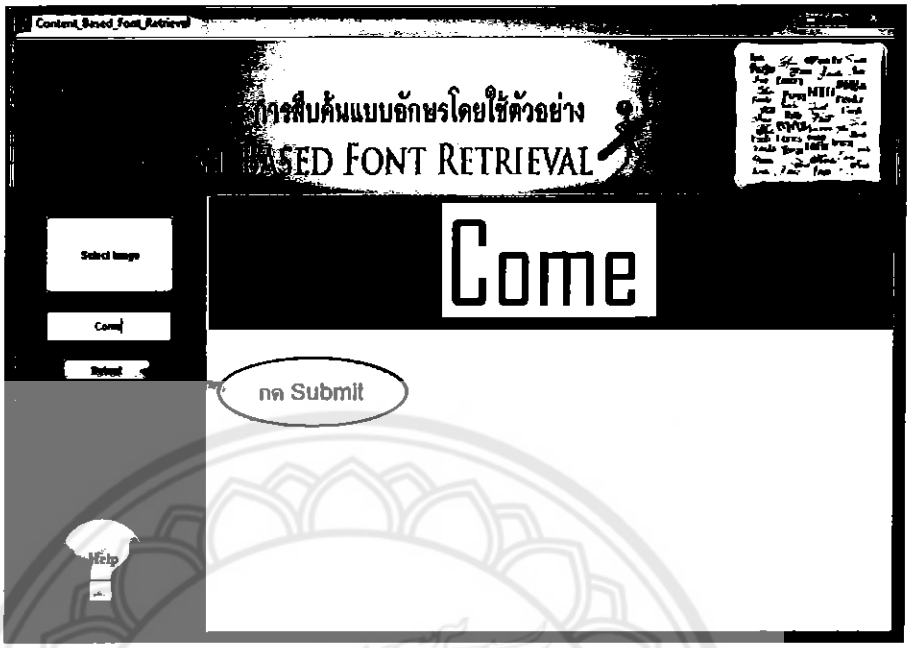


รูปที่ 3.16 รูปแสดงการเลือกรูปภาพตัวอักษรที่ต้องการ

3. ผู้ใช้พิมพ์อักษรตามรูปที่เลือกเพื่อยืนยันความถูกต้องของคำที่ค้นหา

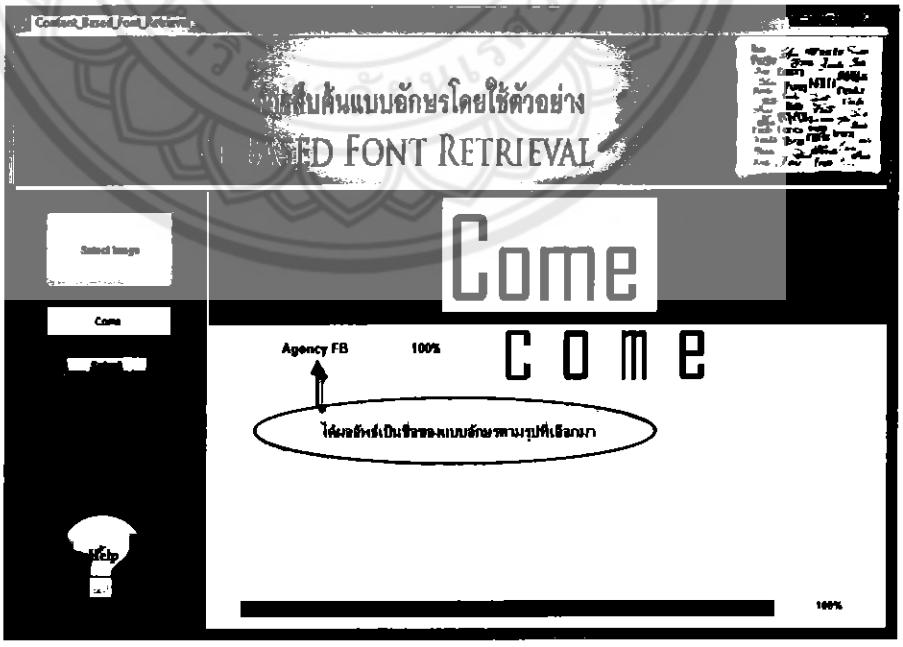
รูปที่ 3.17 รูปแสดงช่องสำหรับพิมพ์ตัวอักษรตามรูปภาพที่ เลือกมา

4. ผู้ใช้กดปุ่ม Submit เพื่อทำการประมวลผล แล้วรอสักครู่เพื่อรอผลลัพธ์



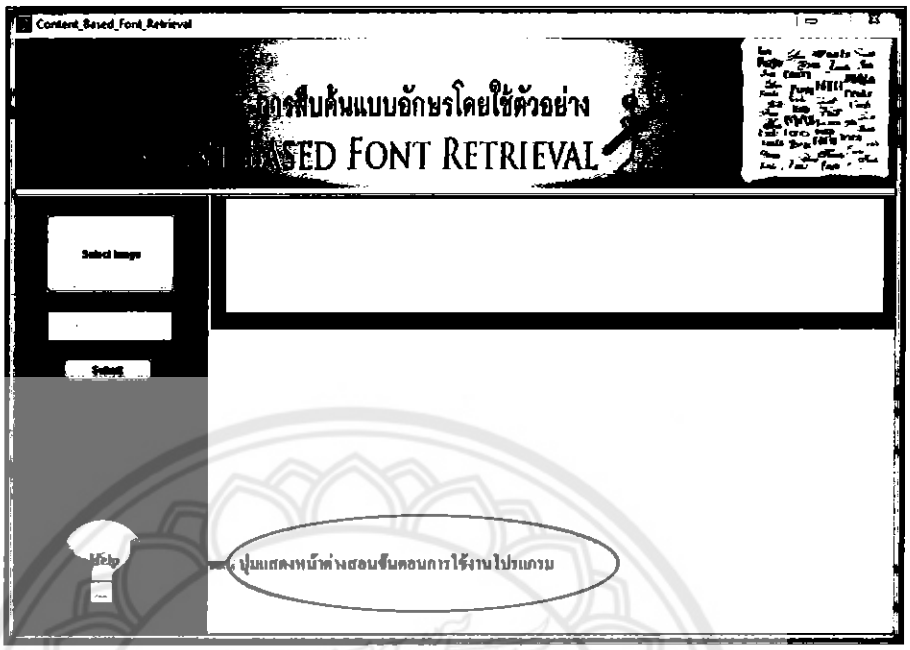
รูปที่ 3.18 รูปแสดงปุ่ม Submit เพื่อเริ่มรันโปรแกรม

5. จะได้ผลลัพธ์เป็นชื่อแบบอักษร ตามรูปที่เลือกมา



รูปที่ 3.19 ผลที่ได้จากการทำงานของโปรแกรม

6. โปรแกรมจะมีปุ่มช่วยเหลือสำหรับบอกวิธีการ และขั้นตอนการใช้งานโปรแกรม



รูปที่ 3.20 รูปแสดงปุ่มช่วยเหลือในการสอนใช้งานโปรแกรม

บทที่ 4
ผลการดำเนินการ

ในบทนี้จะกล่าวถึงผลการดำเนินงานการพัฒนาโปรแกรมสืบค้นแบบอักษร โดยใช้ตัวอย่าง

ภาพแบบอักษรที่ใช้ในการทดสอบเป็นภาพแบบอักษรที่มาจากการแคปเจอร์จากคอมพิวเตอร์ ซึ่งจะทำการทดลองทั้งหมด 3 แบบ คือ แบบอักษรตัวพิมพ์ใหญ่ แบบอักษรตัวพิมพ์เล็ก และแบบอักษรตัวเลข

4.1 การทดสอบการทำงานโดยใช้แบบอักษรตัวพิมพ์ใหญ่

ในการทดสอบการทำงานของโปรแกรมโดยใช้แบบอักษรตัวพิมพ์ใหญ่นี้ จะทำการทดลองโดยแคปเจอร์คำว่า THE QUICK BROWN FOX JUMP OVER THE LAZY DOG จากคอมพิวเตอร์ ซึ่งคำนี้จะมีการใช้ตัวอักษรครบทั้ง 26 ตัว โดยเปลี่ยนแบบอักษรทั้งหมด 50 แบบตามที่ได้เก็บไว้ในฐานข้อมูล โดยมีผลการทดสอบดังตารางที่ 4.1

ตารางที่ 4.1 ตารางแสดงความถูกต้องของผลการทดลองต่อการรับภาพของแบบอักษรตัวพิมพ์ใหญ่แบบต่างๆ

แบบอักษร	ความถูกต้อง
Agency FB	100.0000%
Aharoni	94.1176%
Algerian	94.1176%
Andalus	100.0000%
Angsana New	97.0588%
Arial	94.1176%
Batang	94.1176%
Bell MT	82.3529%
Berlin Sans FB	82.3529%
Bernard MT Condensed	100.0000%

Bodoni MT Black	100.0000%
Calibri	94.1176%
Cambria	82.3529%
Castellar	100.0000%
Comic Sans MS	100.0000%
Cordia New	100.0000%
DilleniaUPC	100.0000%
Forte	100.0000%
Franklin Gothic Book	82.3529%
FreesiaUPC	88.2353%
Gill Sans Ultra Bold Condensed	94.1176%
Goudy Old Style	88.2353%
Goudy Stout	100.0000%
Gulim	88.2353%
Gungsuh	88.2353%
Harrington	100.0000%
Imprint MT Shadow	94.1176%
IrisUPC	94.1176%
JasmineUPC	100.0000%
LilyUPC	88.2353%
Maiandra GD	100.0000%
MoolBoran	94.1176%
MV Boli	100.0000%
Narkisim	100.0000%
OCR A Extended	94.1176%
Rockwell Condensed	100.0000%
Segoe UI Semibold	100.0000%
Showcard Gothic	100.0000%
Simplified Arabic Fixed	100.0000%
SimSun	100.0000%
Sylfaen	82.3529%

Tahoma	88.2353%
Tempus Sans ITC	94.1176%
TH Chakra Petch	88.2353%
TH K2D July8	82.3529%
TH Kodchasal	94.1176%
TH KoHo	88.2353%
TH Mali Grade 6	94.1176%
TH SarabunPSK	88.2353%
VAGRounded BT	100.0000%

จากผลการทดสอบ การนำภาพ แบบอักษรตัวพิมพ์ใหญ่ ที่แคปเจอร์จากคอมพิวเตอร์มาทดสอบโปรแกรม ผลปรากฏว่าโปรแกรมสามารถ บอกชื่อแบบอักษรทุกแบบได้ถูกต้องมากกว่า 80% ซึ่งถือว่าอยู่ในระดับที่ยอมรับได้

4.2 การทดสอบการทำงานโดยใช้แบบอักษรตัวพิมพ์เล็ก

ในการทดสอบการทำงานของโปรแกรมโดยใช้แบบอักษรตัวพิมพ์เล็กนี้ จะทำการทดลองโดยแคปเจอร์คำว่า the quick brown fox jump over the lazy dog จากคอมพิวเตอร์ ซึ่งคำนี้จะมีการใช้ตัวอักษรครบทั้ง 26 ตัว โดยเปลี่ยนแบบอักษรทั้งหมด 46 แบบ ตามที่ได้เก็บไว้ในฐานข้อมูล โดยมีผลการทดสอบดังตารางที่ 4.2

ตารางที่ 4.2 ตารางแสดงความถูกต้องของผลการทดลองต่อการรับภาพของแบบอักษรตัวพิมพ์เล็กแบบต่างๆ

แบบอักษร	ความถูกต้อง
Agency FB	94.1176%
Aharoni	100.0000%
Algerian	-
Andalus	88.2353%
Angsana New	88.2353%
Arial	94.1176%

Batang	100.0000%
Bell MT	88.2353%
Berlin Sans FB	100.0000%
Bernard MT Condensed	100.0000%
Bodoni MT Black	100.0000%
Calibri	88.2353%
Cambria	88.2353%
Castellar	-
Comic Sans MS	94.1176%
Cordia New	88.2353%
DilleniaUPC	94.1176%
Forte	100.0000%
Franklin Gothic Book	88.2353%
FreesiaUPC	94.1176%
Gill Sans Ultra Bold Condensed	100.0000%
Goudy Old Style	82.3529%
Gill Sans Ultra Bold Condensed	100.0000%
Goudy Stout	-
Gulim	100.0000%
Gungsuh	100.0000%
Harrington	100.0000%
Imprint MT Shadow	94.1176%
IrisUPC	88.2353%
JasmineUPC	88.2353%
LilyUPC	100.0000%
Maiandra GD	88.2353%
MoolBoran	100.0000%
MV Boli	100.0000%
Narkisim	88.2353%
OCR A Extended	100.0000%
Rockwell Condensed	88.2353%

Segoe UI Semibold	100.0000%
Showcard Gothic	-
Simplified Arabic Fixed	100.0000%
SimSun	94.1176%
Sylfaen	82.3529%
Tahoma	82.3529%
Tempus Sans ITC	88.2353%
TH Chakra Petch	88.2353%
TH K2D July8	88.2353%
TH Kodchasal	94.1176%
TH KoHo	100.0000%
TH Mali Grade 6	88.2353%
TH SarabunPSK	82.3529%
VAGRounded BT	100.0000%

จากผลการทดสอบการนำภาพแบบอักษรตัวพิมพ์เล็กที่แคปเจอร์จากคอมพิวเตอร์มาทดสอบโปรแกรม ผลปรากฏว่าโปรแกรมสามารถบอกชื่อแบบอักษรทุกแบบได้ถูกต้องมากกว่า 80% ซึ่งถือว่าอยู่ในระดับที่ยอมรับได้

4.3 การทดสอบการทำงานโดยใช้แบบอักษรตัวเลข

ในการทดสอบการทำงานของโปรแกรมโดยใช้แบบอักษรตัวเลขนี้ จะทำการทดลองโดยแคปเจอร์คำว่า 1234567890 จากคอมพิวเตอร์ โดยเปลี่ยนแบบอักษรทั้งหมด 50 แบบ ตามที่ได้เก็บไว้ในฐานข้อมูล โดยมีผลการทดสอบดังตารางที่ 4.3

ตารางที่ 4.3 ตารางแสดงความถูกต้องของผลการทดลองต่อการรับภาพของแบบอักษร ตัวเลขแบบต่างๆ

Agency FB	100.0000%
Aharoni	100.0000%
Algerian	100.0000%
Andalus	100.0000%

Angsana New	100.0000%
Arial	100.0000%
Batang	100.0000%
Bell MT	100.0000%
Berlin Sans FB	100.0000%
Bernard MT Condensed	100.0000%
Bodoni MT Black	100.0000%
Calibri	100.0000%
Cambria	100.0000%
Castellar	100.0000%
Comic Sans MS	100.0000%
Cordia New	100.0000%
DilleniaUPC	100.0000%
Forte	100.0000%
Franklin Gothic Book	100.0000%
FreesiaUPC	100.0000%
Gill Sans Ultra Bold Condensed	100.0000%
Goudy Old Style	100.0000%
Goudy Stout	100.0000%
Gulim	100.0000%
Gungsuh	100.0000%
Castellar	100.0000%
Imprint MT Shadow	100.0000%
IrisUPC	100.0000%
JasmineUPC	100.0000%
LilyUPC	100.0000%
Maiandra GD	100.0000%
MoolBoran	100.0000%
Narkisim	100.0000%
OCR A Extended	100.0000%
Rockwell Condensed	100.0000%

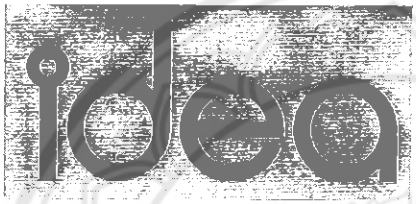
Segoe UI Semibold	100.0000%
Showcard Gothic	100.0000%
Simplified Arabic Fixed	100.0000%
SimSun	100.0000%
Sylfaen	100.0000%
Tahoma	100.0000%
Tempus Sans ITC	100.0000%
TH Chakra Petch	90.0000%
TH K2D July8	100.0000%
TH Kodchasal	100.0000%
TH KoHo	100.0000%
TH Mali Grade 6	100.0000%
TH SarabunPSK	100.0000%
VAGRounded BT	100.0000%

จากผลการทดสอบ การนำภาพแบบอักษรตัวเลขที่แคปเจอร์จากคอมพิวเตอร์มาทดสอบโปรแกรม ผลปรากฏว่าโปรแกรมสามารถบอกชื่อแบบอักษรทุกแบบได้ถูกต้องมากกว่า 90% ซึ่งถือว่าอยู่ในระดับที่ยอมรับได้

4.4 การทดสอบการทำงานของโปรแกรมโดยใช้แบบอักษรจากการสแกน

การทดสอบการทำงานของโปรแกรมโดยใช้แบบอักษรจากการสแกนนี้ ผู้จัดทำได้ทำการสแกนภาพจากหนังสือหรือเอกสารต่างๆ แล้วแคปเจอร์เฉพาะส่วนที่เป็นภาพแบบอักษรเพื่อนำมาทำการทดสอบซึ่งได้ผลดังตารางที่ 4.4

ตารางที่ 4.4 ตารางแสดงความถูกต้องของผลการทดลองต่อการรับภาพของแบบอักษร จากการสแกน

รูปที่	รูปภาพ	ผลการทำงานของโปรแกรม	
1		Cordia New Tahoma	75.0000% 25.0000%
2		Simplified Arabic Fixed TH K2D July8	75.0000% 25.0000%
3		Gulim Cordia New TH Kodchasal	72.7273% 9.0909% 9.0909%
4		Angsana New Rockwell Condensed MoolBoran	76.4706% 17.6471% 5.8824%
5		Gill Sans Ultra Bold Condensed Showcard Gothic	73.3333% 26.6667%
6		TH K2D July8 Cordia New	83.3333% 16.6667%
7		Angsana New Cambria	90.9091% 9.0909%

8		Agency FB	100.0000%
9		Lily UPC	100.0000%
10		VAGRounded BT Calibri	71.4286% 28.5714%
11		Gill Sans Ultra Bold Condensed SimSun Algerian Agency FB Harrington	22.2222% 11.1111% 11.1111% 11.1111% 11.1111%
12		Gill Sans Ultra Bold Condensed Rockwell Condensed Aharoni	60.0000% 20.00000% 20.0000%
13		Agency FB TH Chakra Petch Gill Sans Ultra Bold Condensed Comic Sans MS	57.1429% 14.2857% 14.2857% 14.2857%
14		Angsana New Rockwell Condensed Bodoni MT Black	77.7778% 11.1111% 11.1111%
15		Agency FB Tahoma LilyUPC	60.0000% 20.0000% 20.0000%

16		Segoe UI Semibold	100.0000%
17		Franklin Gothic Book FreesiaUPC Arial	60.0000% 20.0000% 20.0000%
18		Aharoni LilyUPC	85.7143% 14.2857%
19		LilyUPC MoolBoran Arial	77.7777% 11.1111% 11.1111%
20		Gill Sans Ultra Bold Condensed SimSun Algerian Agency FB	66.6666% 11.1111% 11.1111% 11.1111%

จากผลการทดสอบปรากฏว่าโปรแกรมสามารถบอกชื่อแบบอักษรที่ถูกต้องได้มากกว่า 60% ซึ่งภาพบางภาพสามารถบอกผลได้ถูกต้อง 100% ซึ่งจะต้องเป็นภาพที่พื้นหลังเป็นสีขาวและตัวอักษรสีดำ ตามที่ได้กำหนดไว้ และมีแบบอักษรนั้นในฐานข้อมูล ซึ่งบางภาพ ได้ผลลัพธ์ออกมาไม่ถูกต้องนั้นเพราะพื้นหลังของภาพไม่เป็นสีขาว จึงทำให้โปรแกรมตัดตัวอักษรออกมาได้ไม่สมบูรณ์ ผลที่ออกมาจึงผิดพลาด และบางรูปเป็นแบบอัก ษรที่ไม่มีอยู่ในฐานข้อมูล โปรแกรมจึงแสดงแบบอักษรที่ใกล้เคียงที่สุดออกมา ซึ่งผลที่ออกมาจึงมีข้อผิดพลาด

4.5 การทดสอบขนาดของบล็อก

การทดสอบขนาดของบล็อกนี้คือการทดสอบว่าการแบ่งรูปภาพออกเป็นบล็อกขนาดเท่าใดจะทำให้โปรแกรมให้ผลการทำงานถูกต้องมากที่สุด โดยทดสอบจากภาพที่แคปเจอร์มาจากคอมพิวเตอร์ ซึ่งจะให้ความสำคัญต่อความถูกต้องมากที่สุด เรื่องความเร็วของการทำงานของโปรแกรมจะมีความสำคัญรองลงมา

ตารางที่ 4.5 แสดงขนาดบล็อกและความถูกต้องของข้อมูล

ขนาดบล็อก(พิกเซล)	ความถูกต้อง(เปอร์เซ็นต์)	เวลา(วินาที)
5	100.0	52.205
10	100.0	22.044
15	100.0	15.559
20	85.714	14.451
25	85.714	13.943
30	85.714	13.701
35	71.429	13.518
40	57.143	12.428
45	57.143	12.120
50	42.857	13.214
55	28.571	12.557
60	28.571	12.653
65	14.289	12.742

จากผลการทดสอบปรากฏว่ายิ่งบล็อกมีขนาดเล็กลงก็จะทำให้โปรแกรมแสดงผลลัพธ์ที่ถูกต้องมากขึ้น แต่หากบล็อกยิ่งมีขนาดใหญ่ขึ้นก็จะทำให้โปรแกรมทำงานเร็วยิ่งขึ้น จากตารางที่ 4.4 ปรากฏว่า บล็อกขนาด 5x5 10x10 และ 15x15 ให้ผลลัพธ์ที่มีความถูกต้องเท่ากัน แต่เนื่องจากบล็อกขนาด 15x15 ทำให้โปรแกรมทำงานได้เร็วกว่า ผู้จัดทำจึงเลือกใช้บล็อกขนาด 15x15 ในโปรแกรม

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุปผลการดำเนินโครงการ

โครงงานนี้พัฒนาขึ้นเพื่อสร้างโปรแกรมค้นหาแบบอักษรโดยใช้ตัวอย่าง เพื่อลดระยะเวลาในการค้นหาแบบเดิมคือต้องค้นหาและทดลองเปรียบเทียบไปทีละแบบอักษร ถ้าไม่ใช่แบบอักษรที่ต้องการ ต้องทำการเลือกใหม่แบบนี้ไปจนกว่าจะได้แบบอักษรที่ต้องการ

ผู้จัดทำโครงงานจึงสร้างโปรแกรมนี้ขึ้น โดยมีเป้าหมายเพื่อลดระยะเวลาในการค้นหาแบบอักษรจากแบบเดิมมาใช้เป็นการค้นหาจากภาพ โดยโปรแกรมจะเริ่มจากการรับภาพที่ต้องการค้นหามาทำการโพรเจกชันในแนวตั้งและแนวนอน เพื่อแยกตัวอักษรออกเป็นตัว จากนั้นจึงทำการแบ่งภาพออกเป็นบล็อกเพื่อทำการโค้ดบล็อกให้คะแนนความสำคัญของแต่ละบล็อก แล้วนำคะแนนมารวมกันแล้วหาเป็นเปอร์เซ็นต์จากรูปต้นฉบับ

ผลลัพธ์ที่ได้ชื่อของแบบอักษรที่ถูกต้องจะแสดงออกมาใน อันดับที่ 1 ถึงอันดับที่ 2 เสมอ ซึ่งอยู่ในระดับที่ยอมรับได้ที่จะไปค้นหาแบบอักษรที่ต้องการ

5.2 ปัญหาในการดำเนินงานและแนวทางในการแก้ไขปัญหา

ตารางที่ 5.1 ปัญหาในการดำเนินงานและแนวทางในการแก้ไข

ปัญหาในการดำเนินงาน	แนวทางการแก้ไขปัญหา
1. ขนาดของ บล็อกที่ไม่เหมาะสมจะทำให้ผลของโปรแกรมผิดพลาด เนื่องจากขนาดของบล็อกจะมีผลต่อความถูกต้อง และความเร็วของโปรแกรมเป็นอย่างมาก	แก้ไขโดย ทดลองเปลี่ยนขนาดของบล็อกไปเรื่อยๆ เพื่อหาขนาดที่เหมาะสมที่สุด ซึ่งเราจะให้ความสำคัญต่อความถูกต้องมากที่สุด ส่วนเรื่องความเร็วจะเป็นเรื่องลองลงมา โดยผลสรุปคือ ขนาด 15x15 พิกเซล จะให้ความถูกต้องมาก
2. การเก็บแบบอักษรแยกเป็นโฟลเดอร์ทำให้เกิดปัญหาไม่สามารถตั้งชื่อซ้ำกันได้ คือ ในคอมพิวเตอร์ การตั้งชื่อโฟลเดอร์ว่า A กับ a จะถือว่าเป็นชื่อเดียวกัน	แก้ไขปัญหาโดยทำการเก็บแบบอักษรตัวเดียวกันทั้งตัวพิมพ์ใหญ่และตัวพิมพ์เล็กไว้ในโฟลเดอร์เดียวกัน

5.3 ข้อเสนอแนะในการดำเนินโครงงาน

1. ลักษณะแบบอักษรที่เก็บไว้เพื่อใช้เปรียบเทียบยังมีน้อยเกินไปถ้า ทำการค้นหาแบบอักษรโดยที่ไม่มีต้นแบบของแบบอักษรที่เก็บอยู่การแสดงผลที่ออกมา จะมีความผิดพลาดได้ ทางแก้ไขก็คือต้องมีการเก็บตัวต้นแบบของแบบอักษรที่ออกมาใหม่อยู่ตลอดเพื่อผลลัพธ์ที่ถูกต้องและครอบคลุมการใช้งาน

2. ภาพที่นำมาใช้ในโปรแกรมนี้ได้ตอนนี้ยังต้องเป็นภาพที่ มีตัวอักษรเป็นสีดำและพื้นหลังสีขาวเท่านั้น สิ่งที่ควรทำต่อไปคือรับภาพตัวอักษรที่เป็นภาพสีมาประมวลผลได้ จนถึงสามารถประมวลผลภาพที่มีพื้นหลังของตัวอักษรไม่ใช่สีขาวได้

5.4 ความรู้ที่จำเป็นต่อโครงงานนี้

ผู้ที่จะนำไปพัฒนาต่อควรมีความรู้เบื้องต้นเกี่ยวกับการเขียนโปรแกรม MATLAB เพื่อที่จะสามารถศึกษาเรียนรู้ นำไปต่อยอดได้อย่างรวดเร็ว และมี ความรู้พื้นฐานเกี่ยวกับ Image Processing ในการวิเคราะห์ แยกแยะ และประมวลผลภาพได้

เอกสารอ้างอิง

[1] Dearp. (25 สิงหาคม 2549). ลักษณะและความหมายของ Pixel. สืบค้นเมื่อ 4 กรกฎาคม 2554, http://m2you.blogspot.com/2006/08/pixel.html

[2] ไม่ปรากฏชื่อผู้แต่ง. "Computer vision" [Online].Available: http://staff.cs.psu.ac.th/wiphada/Teaching%20Coureses/sem%202-2547/344-471%20AI-ES/com_vision.ppt

[3] ยุทธนา แสงสุวรรณ. (14 พฤศจิกายน 2552). กราฟิกที่ใช้ในงานคอมพิวเตอร์. สืบค้นเมื่อ 2 พฤศจิกายน 2554, http://images.lekyutthana.multiply.multiplycontent.com/journal

[4] เกียรติพงษ์ กตติสุพัฒน์. (2553). รู้จักกับความละเอียดของภาพ. สืบค้นเมื่อ 20 มกราคม 2555, http://www.108award.com/index.php?lay=show&ac=article&Id=53896525&Ntype=1

[5] edu-mine. (2552). โหมดสีต่างๆ. สืบค้นเมื่อ 20 มการาคม 2555, http://www.edu-mine.com/photoshop/lesson5_colormode.html

[6] ไม่ปรากฏชื่อผู้แต่ง. (17 เมษายน 2555). Otsu's method. สืบค้นเมื่อ 25 เมษายน 2555, http://en.wikipedia.org/wiki/Otsu's_method#cite_note-Otsu-1

[7] นิรุธ อำนวยศิลป์. (2554). การประมวลผลภาพและวีดีโอ. กรุงเทพฯ: บริษัทด่านสุทธาการพิมพ์ จำกัด

[8] ณัฐธิดา ลีสม, โอฬาริก สุรินต๊ะ (2554). "การตัดตัวอักษรลายมือเขียนภาษาไทยออกจากเอกสารภาพเชิงดิจิทัล". สืบค้นเมื่อ 20 เมษายน 2555, http://www.wbi.msu.ac.th/file/595/doc_6_paper.doc

ภาคผนวก ก

การเขียนโปรแกรมส่วนติดต่อกับผู้ใช้

ในโปรแกรมส่วนติดต่อผู้ใช้ (Graphic User Interface) นี้จะทำหน้าที่รับภาพที่ผู้ใช้ต้องการ และรับค่าตัวอักษรที่ผู้ใช้พิมพ์ ส่งไปให้โปรแกรมส่วนประมวลผล จากนั้นโปรแกรมส่วนประมวลผลจะส่งผลลัพธ์มาแสดงที่โปรแกรมส่วนติดต่อผู้ใช้

Source Code ของโปรแกรม

```
function varargout = Content_Based_Font_Retrieval(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Content_Based_Font_Retrieval_OpeningFcn, ...
                   'gui_OutputFcn',  @Content_Based_Font_Retrieval_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end


function Content_Based_Font_Retrieval_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
help = imread('help-icon.jpg'); % show picture in help bottom
set(handles.help,'CData',help);
```

```
head = imread('header.jpg'); % show picture in header
imshow(head,'parent',handles.header);
set(handles.text,'enable','off') %  set enable off for edit text
function varargout = Content_Based_Font_Retrieval_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function Select_Callback(hObject, eventdata, handles)
global image
try
currentPath = pwd;  % current path
[imageName,pathName] = uigetfile( ...
        { '*.jpg;*.bmp;*.png'}, ...
        'Select image');
cd(pathName);

image = imread(char(imageName));
imshow(image,'parent',handles.original); % show picture from user select
cd(currentPath);
catch
        disp('no select file')
end
set(handles.text,'enable','on'); % set enable on for edit text

%----------clear all name font, percent and example picture----------
fontSet = [handles.font1 handles.font2 handles.font3 handles.font4 handles.font5];
percentSet = [handles.percen1 handles.percen2 handles.percen3 handles.percen4
handles.percen5];
set(fontSet,'String',{''});
set(percentSet,'String',{''});
whiteImg = imread('BG.png');
imshow(whiteImg,'parent',handles.axes4);
imshow(whiteImg,'parent',handles.axes5);
imshow(whiteImg,'parent',handles.axes6);
```

```
imshow(whiteImg,'parent',handles.axes7);
imshow(whiteImg,'parent',handles.axes8);
imshow(whiteImg,'parent',handles.axes25);
imshow(whiteImg,'parent',handles.axes10);
imshow(whiteImg,'parent',handles.axes11);
imshow(whiteImg,'parent',handles.axes12);
imshow(whiteImg,'parent',handles.axes13);
imshow(whiteImg,'parent',handles.axes14);
imshow(whiteImg,'parent',handles.axes15);
imshow(whiteImg,'parent',handles.axes16);
imshow(whiteImg,'parent',handles.axes17);
imshow(whiteImg,'parent',handles.axes18);
imshow(whiteImg,'parent',handles.axes19);
imshow(whiteImg,'parent',handles.axes20);
imshow(whiteImg,'parent',handles.axes21);
imshow(whiteImg,'parent',handles.axes22);
imshow(whiteImg,'parent',handles.axes23);
imshow(whiteImg,'parent',handles.axes24);
imshow(whiteImg,'parent',handles.axes25);
imshow(whiteImg,'parent',handles.axes26);
imshow(whiteImg,'parent',handles.axes27);
imshow(whiteImg,'parent',handles.axes28);
imshow(whiteImg,'parent',handles.axes29);

%------------------------------Type input in space rectanger----------------------------------
function text_Callback(hObject, eventdata, handles)
global word
word = get(handles.text,'String')%input of system is 'word',then have type is string
%length(word)
% --- Executes during object creation, after setting all properties.
function text_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
```

```
        set(hObject,'BackgroundColor','white');
end
% -------------------- when you click submit button ----------------------------------
function submit_Callback(hObject, eventdata, handles)
global word image num
currentPath1 = pwd; % current path
tic
num = 1;
% -------- find all font on image input --------------------
[arrFullName arrCutName] = projectionGUI(handles,image); %call function projection
% ----------------------------------------------------------


% ------- find most dupplicate font name in array 'arrAllFont' --------
% input -> arrAllFont(tpye is array)
%   Ex.    ['angsana' 'bell' 'thSarabun' 'angsana']
% output -> table 2 column
%   Ex.    -------------------------
%         | font name | most number |
%         | angsana   |     2     |
%         | bell      |     1     |
%         | thSarabun |     1     |
x=1;
y=1;
arrFont = {};
%arrScore = 0;
for i = 1:length(arrCutName)
        count = 1;
        pointer = arrCutName(i);
        if sum(strcmp(pointer,arrFont)) == 0 % if no match in arrFont
                arrFont(x) = pointer;
                x = x+1;
                for j = i+1:length(arrCutName)
```

```
                    if strcmp(pointer,arrCutName(j)) == 1    % if a(i) == a(j)
                    count = count+1;
                    end
                end
                arrScore(y) = count;
                y=y+1;
            end
        end

% ----- sort by max to min ----------------------------
[a index] = sort(arrScore,'descend');
% ------ calculate percen --------------------------
count = 1;
keepFont = {'','','','',''};
arrPerCen = 0;
ff = true(225,120);
for xx=1:length(index)
keepFont(count) = arrFont(index(xx));
        percen = (arrScore(index(xx))/sum(arrScore))*100;
        arrPerCen(count) = percen;
        count =count+1;
end


% ---------- show name and image (1) --------------------
showFont = char(keepFont(1));
showFontTxt = showFont;
set(handles.font1,'String',showFontTxt);
set(handles.percen1,'String',strcat(num2str(arrPerCen(1)),'%'));
%       ------------------
cd(strcat([currentPath1 '\character\' word(1)]));
tempName = char(arrFullName(1));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
```

```
ff = imread(setName);
imshow(ff,'parent',handles.axes4);
%       ------------------
cd(strcat([currentPath1 '\character\' word(2)]));
tempName = char(arrFullName(2));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes5);
%       ------------------
cd(strcat([currentPath1 '\character\' word(3)]));
tempName = char(arrFullName(3));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes6);
%       ------------------
cd(strcat([currentPath1 '\character\' word(4)]));
tempName = char(arrFullName(4));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes7);
%       ------------------
cd(strcat([currentPath1 '\character\' word(5)]));
tempName = char(arrFullName(5));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes8);
% ---------- show name and image (2) ---------------------
showFont = char(keepFont(2));
showFontTxt = showFont;
set(handles.font2,'String',showFontTxt);
set(handles.percen2,'String',strcat(num2str(arrPerCen(2)),'%'));
%       ------------------
```

```
cd(strcat([currentPath1 '\character\' word(1)]));
tempName = char(arrFullName(1));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes10);
%       ------------------
cd(strcat([currentPath1 '\character\' word(2)]));
tempName = char(arrFullName(2));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes11);
%       ------------------
cd(strcat([currentPath1 '\character\' word(3)]));
tempName = char(arrFullName(3));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes12);
%       ------------------
cd(strcat([currentPath1 '\character\' word(4)]));
tempName = char(arrFullName(4));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes13);
%       ------------------
cd(strcat([currentPath1 '\character\' word(5)]));
tempName = char(arrFullName(5));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes14);
% ---------- show name and image (3) --------------------
showFont = char(keepFont(3));
showFontTxt = showFont;
```

```
set(handles.font3,'String',showFontTxt);
set(handles.percen3,'String',strcat(num2str(arrPerCen(3)),'%'));
%       --------------------
cd(strcat([currentPath1 '\character\' word(1)]));
tempName = char(arrFullName(1));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes15);
%       --------------------
cd(strcat([currentPath1 '\character\' word(2)]));
tempName = char(arrFullName(2));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes16);
%       --------------------
cd(strcat([currentPath1 '\character\' word(3)]));
tempName = char(arrFullName(3));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes17);
%       --------------------
cd(strcat([currentPath1 '\character\' word(4)]));
tempName = char(arrFullName(4));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes18);
%       --------------------
cd(strcat([currentPath1 '\character\' word(5)]));
tempName = char(arrFullName(5));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes19);
```

```
% ---------- show name and image (4) ----------
showFont = char(keepFont(4));
showFontTxt = showFont;
set(handles.font4,'String',showFontTxt);
set(handles.percen4,'String',strcat(num2str(arrPerCen(4)),'%'));
%       ----------
cd(strcat([currentPath1 '\character\' word(1)]));
tempName = char(arrFullName(1));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes20);
%       ----------
cd(strcat([currentPath1 '\character\' word(2)]));
tempName = char(arrFullName(2));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes21);
%       ----------
cd(strcat([currentPath1 '\character\' word(3)]));
tempName = char(arrFullName(3));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes22);
%       ----------
cd(strcat([currentPath1 '\character\' word(4)]));
tempName = char(arrFullName(4));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes23);
%       ----------
cd(strcat([currentPath1 '\character\' word(5)]));
tempName = char(arrFullName(5));
```

```
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes24);
% ---------- show name and image (5) ---------------------
showFont = char(keepFont(5));
showFontTxt = showFont;
set(handles.font5,'String',showFontTxt);
set(handles.percen5,'String',strcat(num2str(arrPerCen(5)),'%'));
%       -------------------
cd(strcat([currentPath1 '\character\' word(1)]));
tempName = char(arrFullName(1));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes25);
%       -------------------
cd(strcat([currentPath1 '\character\' word(2)]));
tempName = char(arrFullName(2));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes26);
%       -------------------
cd(strcat([currentPath1 '\character\' word(3)]));
tempName = char(arrFullName(3));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes27);
%       -------------------
cd(strcat([currentPath1 '\character\' word(4)]));
tempName = char(arrFullName(4));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes28);
```

```
%          -------------------
cd(strcat([currentPath1 '\character\' word(5)]));
tempName = char(arrFullName(5));
setName = strcat([showFontTxt tempName(length(tempName)-4) '.jpg']);
ff = imread(setName);
imshow(ff,'parent',handles.axes29);

function help_Callback(hObject, eventdata, handles)
set(gcf,'pointer','watch');
web ([cd '\help\help.htm'], '-browser'); % kick help bottom for show help in browser
set(gcf,'pointer','arrow');
```

ภาคผนวก ข

การเขียนโปรแกรมส่วนการประมวลผล

ในโปรแกรมส่วนการประมวลผลนี้จะทำหน้าที่รับภาพที่ผู้ใช้ต้องการ และตัวอักษรที่ผู้ใช้พิมพ์จากโปรแกรมส่วนติดต่อผู้ใช้ มาทำการประมวลผลหาแบบอักษรที่ตรงกับภาพที่ผู้ใช้เลือกมามากที่สุดแล้วส่งผลที่ได้ กลับไปให้โปรแกรมส่วนติดต่อผู้ใช้ เพื่อนำไปแสดงผลต่อไป

Source Code ของโปรแกรม

Function Projection

```
function [arrFullName arrCutName] = projection(handles,pic)
global word  num
z = 0;
level = graythresh(pic);
pic = im2bw(pic,level);
[M N]=size(pic);
```

ฟังก์ชัน โพรเจกชัน เริ่มจากการรับภาพจากแฟ้มข้อมูลเข้ามาและ ปรับปรุงภาพให้เหมาะสมต่อการนำไปประมวลผลต่อไป โดยแปลงภาพเป็นภาพเกรย์สเกล จากนั้นแปลงภาพเกรย์สเกลไปเป็นภาพขาว-ดำ

```
%% ---------- vertical projection (STEP 1) ----------
picTempVer = ones(M,N);
for i=N:-1:1
        count = M;
        for j=M:-1:1
                if (pic(j,i) == 0)
                picTempVer(count,i) = 0;
```

```
            count = count-1;
            end
        end
end
```

ขั้นตอนที่ 1 เริ่มจากการโพรเจกชันในแนวตั้งจะได้กราฟฮิสโทแกรมในแนวตั้ง

```
%% --------- cut of vertical (STEP 1.1)
count = 0;
for col=1:N-1
    if (picTempVer(M,col) == 1)&&(picTempVer(M,col+1) == 0)
        count = count+1;
        keep(count)=col+1;
    end
    if (picTempVer(M,col) == 0)&&(picTempVer(M,col+1) == 1)
        count = count+1;
        keep(count)=col;
    end
end
```

ขั้นตอนต่อมา ทำการหาจุดเริ่มต้นและจุดสิ้นสุดในแนวแกน x ของตัวอักษรจากกราฟฮิสโทแกรม

```
%% ---------------------- end of STEP 1 ----------------------------

tt = 1;
for i = 1:2:length(keep)-1
    x = imcrop(pic,[keep(i) 0 keep(i+1)-keep(i) M]);
```

จากจุดเริ่มต้นและจุดสิ้นสุดที่ได้ ทำการตัดขอบที่เกินจากตัวอักษรทั้งด้านซ้ายและด้านขวาออก

```
% ---------- holizontal projection (STEP 2) ----------
[M N]=size(x);
picTempHol = ones(M,N);
for ii=1:M
        count = 1;
        for jj=1:N
                if (x(ii,jj) == 0)
                        picTempHol(ii,count) = 0;
                        count = count+1;
                end
        end
end
```

ขั้นตอนที่ 2 ทำการโพรเจกชันในแนวนอนจะได้กราฟฮิสโทแกรมในแนวนอน

```
%% ---------- cut holizontal (STEP 2.1) ----------
[M N]=size(picTempHol);
%up to botton
count1 = 1;
for col = 1:N
        for row = 1:M
                if picTempHol(row,col) == 0
                        keepRowUp(count1) = row;
                        count1 = count1+1;
                end
        end
end
%botton to up
```

```
count2 = 1;
for col = 1:N
        for row = M:-1:1
                if picTempHol(row,col) == 0
                        keepRowBot(count2) = row;
                        count2 = count2+1;

                end
        end
end
heigth = keepRowBot(1)-keepRowUp(1);
xx = imcrop(x,[0 keepRowUp(1) N heigth]);
crop = imresize(xx,[225 120]);
```

ขั้นตอนต่อมาคือ ทำการตัดจากกราฟฮิสโทแกรมโดยตัดแยกขอบล่างสุดและบนสุด ที่เกินจากตัวอักษรออกจากนั้นก็ทำการปรับขนาดตัวอักษรออกให้มีขนาด 225x120 พิกเซล

```
% ------------ end of STEP 2 ----------------------------------------

codeTableImg = block(crop);
z = z+1;
str = word(z); %'A','B'...
name = compare(handles,str,codeTableImg); % name = a name of font
value = num/length(word);
update_waitbar(handles,value)
num=num+1;

arrFullName(tt) = {name};
arrCutName(tt) = {name(1:end-5)};
tt = tt+1;
end
```

จะทำการตั้งชื่อของภาพที่ได้ออกมาใหม่โดยจับคู่กับตัวอักษรที่ผู้ใช้พิมพ์ โดยภาพตัวอักษรตัวแรกที่ตัดได้จะจับคู่กับตัวอักษรตัวแรกที่ผู้ใช้พิมพ์เข้าไป และตัวอักษรที่ตัดได้เป็นลำดับถัดมา ก็จะจับคู่กับตัวอักษรที่ผู้ใช้พิมพ์เข้าไปเป็นตัวถัดมา จะจับคู่แบบนี้จนครบ เพื่อยืนยันว่าภาพตัวอักษรที่ตัดออกมาแต่ละตัวคือตัวอะไร เพื่อที่จะนำไปใช้ในขั้นตอนต่อไป

Function block

ฟังก์ชันบล็อก เป็นการแบ่งภาพตัวอักษรที่ ได้จาก การทำ โพรเจกชันเป็น บล็อกๆแล้วทำการให้คะแนนแต่ละบล็อกเพื่อนำไปเปรียบเทียบกับรูปต้นฉ บับต่อไป โดยขั้นตอนการทำงานเริ่มจากแปลงรูปโดยการปรับขนาดให้มีขนาด 225x120

```
function codeTable = block(img)
img = imresize(img,[225 120]);
% ----- create code table ------------
[M N D] = size(img);
sizeBlock = 15;
numberOfRow = M/sizeBlock; %Ini -> 20
numberOfCol = N/sizeBlock; %Ini -> 10
codeTable = zeros(numberOfRow,numberOfCol); % Ini -> 0 al
```

ทำการกำหนดขนาดของบล็อก โดยโปรแกรมกำหนดให้มีขนาด 15x15 พิกเซล

```
allPixInOneBlk = sizeBlock*sizeBlock;
y = 1;
z = 0;
for i = 1:floor(M/sizeBlock)
        x = 1;
        for j = 1:floor(N/sizeBlock)
                code = 0;
                white = 0;
                black = 0;
```

```
                bock = imcrop(img,[x y sizeBlock-1 sizeBlock-1]);
                for k = 1:sizeBlock-1
                for l = 1:sizeBlock-1
                        if bock(k,l) == 0
                        black = black+1;
                end
        end
end
```

ทำการแบ่งภาพออกเป็นบล็อกขนาด 15x15 พิกเซล โดยนำด้านกว้างและด้านยาวมาหารด้วยขนาดของบล็อก

```
%---- calcilate percen in one block -----
    % sol -> (black pixel/all pixel)*100
      per = (black/allPixInOneBlk)*100;

    %---- convert percen to 5 code ----------
    % sol ->  0-20  code 1
    %       21-40  code 2
    %       41-60  code 3
    %       61-80  code 4
    %       81-100 code 5
    if (per >= 0) && (per <= 20)
      code = 1;
    elseif (per > 20) && (per <= 40)
      code = 2;
    elseif (per > 40) && (per <= 60)
      code = 3;
    elseif (per > 60) && (per <= 80)
      code = 4;
    elseif (per > 80) && (per <= 100)
      code = 5;
```

```
    end
codeTable(i,j) = code;
x = x+sizeBlock;
end
```

ทำการนำจำนวนพิกเซลสีดำที่อยู่ในแต่ละบล็อกแล้วให้ค่าของตัวเลขเป็นโค้ดแทนแต่ละบล็อกโดย

ถ้ามีจำนวนพิกเซลสีดำ 0 ถึง 20 เปอร์เซ็นต์ จะให้ค่าเท่ากับ 1

ถ้ามีจำนวนพิกเซลสีดำมากกว่า 20 ถึง 40 เปอร์เซ็นต์ จะให้ค่าเท่ากับ 2

ถ้ามีจำนวนพิกเซลสีดำมากกว่า 40 ถึง 60 เปอร์เซ็นต์ จะให้ค่าเท่ากับ 3

ถ้ามีจำนวนพิกเซลสีดำมากกว่า 60 ถึง 80 เปอร์เซ็นต์ จะให้ค่าเท่ากับ 4

ถ้ามีจำนวนพิกเซลสีดำมากกว่า 80 ถึง 100 เปอร์เซ็นต์ จะให้ค่าเท่ากับ 5

Function Compare

```
function font = compare(handles,arrWord,codeTableImg)
% ----- cd folder 'Large' ------
min = 10000000;
currentFolder = pwd;
% -------- check 'arrWord' is character? ----------
%  - Use function 'ischar' in MATLAB Program
%  - sol -> returns logical 1 (true) if A is a character array and logical 0 (false) otherwise.
tf = ischar(arrWord);
% ------- set path (for fine folder character) ---------
newPath = strcat([currentFolder '\character\' arrWord]);
%--- into new path for load all file in folder ----------
cd(newPath)
file = dir('*.jpg'); % load all file dot 'jpg' in folder of new path
for i=1:length(file)
        cd(newPath)
        file = dir('*.jpg');
        fontName = file(i).name;
```

```
        data = imread(fontName);
        level = graythresh(data);
data = im2bw(data,level);
cd(currentFolder)

codeTableFolder = block(data); % call block function
[mCodeTableImg nCodeTableImg] = size(codeTableImg);
[mCodeTableFolder nCodeTableFolder] = size(codeTableFolder);
```

จากที่ภาพตัวอักษรได้จับคู่กับตัวอักษรที่ผู้ใช้พิมพ์เข้ามาไว้แต่แรกแล้ว จึงนำตัวอักษรนั้นไปเซต path ที่อยู่ไฟล์ เพื่อให้สามารถนำภาพตัวอักษร นั้นไปเทียบกับตัวอักษรตัวเดียวกันได้อย่างถูกต้อง

```
% ----- fine differanc of 2 codeTable ---------
if (mCodeTableImg == mCodeTableFolder) && (nCodeTableImg == nCodeTableImg)
        valueTemp = 0;
        valueSum = 0;
        for ic = 1 : mCodeTableImg
                for jc = 1 : nCodeTableImg
                        valueTemp = abs(codeTableImg(ic,jc)-codeTableFolder(ic,jc));
                        valueSum = valueTemp + valueSum;
                end
        end
end
  % ---------------------- END ---------------------------
```

ทำการหาค่าผลต่างระหว่างบล็อก 2 บล็อกโดยนำค่าบล็อกลบกันแบบ บล็อกต่อบล็อก แล้วใส่เครื่องหมายแอปโซลูท แล้วทำการรวมค่า ผลต่าง ของทุกบล็อกด้วยคำสั่ง sum จากนั้นเก็บค่าไว้ ทำการเปรียบเทียบแบบนี้ไปทุกๆตัวอักษรในแฟ้มข้อมูลนั้นจนครบ

```
% ---------- find minimum differanc value ----------
if valueSum < min
        min = valueSum;
        font = fontName;
end
```

ทำการเปรียบเทียบ เลือกแบบอักษรที่มีผลต่าง รวมน้อยที่สุด ซึ่งแสดงว่าแบบอักษรนั้นมีลักษณะใกล้เคียงกับรูปที่ผู้ใช้เลือกมามากที่สุด จากนั้นส่งค่ากลับไปให้โปรแกรมส่วนติดต่อผู้ใช้งานเพื่อแสดงผลที่โปรแกรมสามารถคำนวณได้