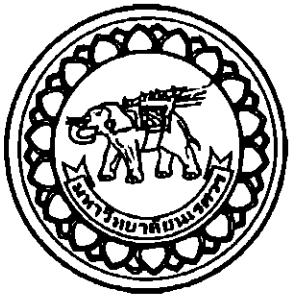


การแปลความหมายตัวอักษรช่องสัญญาณความถี่ผ่านหน้าจอแอลซีดี

RF Chanel OCR from LCD Monitor

นายภีรภาคย์ วรรณา รหัส 52371405

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร
ปีการศึกษา 2555

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อโครงงาน	**การแปลความหมายตัวอักษรช่องสัญญาณความถี่จากหน้าจอแอลซีดี**
ผู้ดำเนินโครงงาน	**นายภีรภาคย์ วรรณา** **รหัส** 52371405
ที่ปรึกษาโครงงาน	อาจารย์เศรษฐา ตั้งค้าวานิช
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2555

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้ปริญญานิพนธ์**ฉบับนี้เป็นส่วนหนึ่ง**ของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

.....ที่ปรึกษา**โครงงาน**

(อาจารย์เศรษฐา ตั้งค้าวานิช)

.....กรรมการ

(ผศ.ดร.พนมขวัญ ริยะมงคล)

.....กรรมการ

(ผศ.ดร.สุรเชษฐ์ กานต์ประชา)

.....กรรมการ

(ดร.พนัส นัถฤทธิ์)

หัวข้อโครงงาน	**การแปลความหมายตัวอักษรช่องสัญญาณความถี่ผ่านหน้าจอแอลซีดี**
ผู้ดำเนินโครงงาน	**นายภีรภาคย์ วรรณา รหัส 52371405**
อาจารย์ที่ปรึกษา	**อาจารย์เศรษฐา ตั้งค้าวานิช**
สาขาวิชา	**วิศวกรรมคอมพิวเตอร์**
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2555

.....

บทคัดย่อ

โครงงานนี้นำเสนอวิธีการประยุกต์การใช้หลักการทางด้านการประมวลผลทางภาพ (Image Processing) มาใช้ในการทำกระบวนการรู้จำอักขระทางภาพ (Optical Character Recognition) ที่หน้าจอมอนิเตอร์ที่ให้ความสนใจภายใต้ค่าความสว่างของแสงที่แตกต่างกันออกไป โดยทั่วไปการที่จะวัดค่าสัญญาณบางอย่างจากเครื่องมืออุปกรณ์ทางอิเล็คทรอนิคนั้น นิยมใช้สายสัญญาณทำหน้าที่เป็นตัวกลางระหว่างอุปกรณ์อิเล็คทรอนิคกับเครื่องมือวัดสัญญาณเพื่อช่วยในการวัดค่า โครงงานนี้เป็นการวัดค่าแบบไม่ใช้สายสัญญาณ โดยจะใช้กล้องเว็บแคมจับภาพหน้าจอมอนิเตอร์แล้วนำมาผ่านกระบวนการประมวลผลขั้นต้น(Pre Processing) จากนั้นจะนำภาพที่ได้เข้าสู่ขั้นตอนของการการรู้จำเพื่อแปลความหมายของตัวอักษรในภาพออกมาเป็นอักขระในเครื่องคอมพิวเตอร์ที่สามารถที่จะทำการแก้ไขหรือเปลี่ยนแปลงได้ ทั้งนี้ผลของการทดลองที่ได้ควรจะมีความถูกต้องของการแปลความหมายและแนวโน้มของค่าที่ใช้ในการทดลองไปในทิศทางเดียวกันและควรที่จะมีความสัมพันธ์กันทางเชิงเส้นอีกด้วย จึงจะถือว่าเป็นการทดลองที่ดี ที่จะนำไปใช้ในการเปรียบเทียบความแตกต่างหว่างสองรูปแบบสีในขั้นตอนสุดท้าย

Project Title	RF Chanel OCR from LCD Monitor
Name	Mr. Piirapaak One-naa ID. 52371405
Project Advisor	Mr.Settha Tangkawanit
Major	Computer Engineering.
Department	Electrical and Computer Engineering.
Academic year	2012

Abstract

Nowaday, when we want to measurement some signal value from electronic devices, mostly we usually use an electric wire to be an intermediate media between electronic devices and the devices we use measurement. This methods may cause damage to those stuff. So, this project will present the digital image processing theorem to show the value from measurement instead of electric wire. By using webcam monitoring the monitor screen within the area we interest. Then, photograph and take it to preprocessing process. After that, we take these pictures to the Optical Character Recognition (OCR) process to translate the meaning and record it to the text file in computer. However, the result from experiment supposed to accurate and also relate to linear relation if we have both of its properties this experiment be accomplishment. Eventually, we can compare the efficacy between RGB model color and HSV model color to find out that which one is better under intensity condition.

กิตติกรรมประกาศ

โครงงานวิศวกรรมฉบับนี้ สำเร็จลุล่วงไปได้ด้วยดีเนื่องจากความอนุเคราะห์ของอาจารย์ที่ปรึกษาโครงงานคือ อาจารย์เศรษฐา ตั้งค้าวานิช รวมถึงอาจารย์ทุกท่านที่กรุณาสละเวลาอันมีค่าคอยชี้แนะแนวทางการทำงานและคอยให้คำปรึกษา ให้ความช่วยเหลือ

ขอขอบพระคุณบรรดาพี่ เพื่อนและน้องๆ ทุกคนที่คอยช่วยเหลือทั้งแรงกาย ความคิดและคำชี้แนะ ที่ซึ่งทำให้โครงงานนี้ประสบความสำเร็จลุล่วงไปได้ด้วยดี

และท้ายที่สุดขอขอบพระคุณบิดา และมารดาของผู้จัดทำ ที่เป็นผู้อยู่เบื้องหลังของทุกความสำเร็จของข้าพเจ้า คอยให้การสนับสนุนในทุกๆด้านตลอดมาโดยเฉพาะอย่างยิ่งในด้านการศึกษาที่ถือว่าเป็นสิ่งสำคัญที่สุดสิ่งหนึ่งในชีวิตของผู้จัดทำ ผู้จัดทำขออ้อมรำลึกในพระคุณและขอกราบขอบพระคุณมา ณ ที่นี้

นายภีรภาคย์ วรรณา

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญตาราง

ตารางที่ หน้า

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

โดยทั่วไปเมื่อต้องการวัดค่าสัญญาณจากอุปกรณ์ทางอิเล็คทรอนิค จะต้องมีการต่อสายสัญญาณ บางครั้งอาจจะต้องทำการเปิดตัวอุปกรณ์นั้นแล้วต่อสายสัญญาณที่ใช้ในการวัดเข้าไปข้างในเพื่อให้ได้มาซึ่งค่าสัญญาณต่างๆที่ต้องการ การทำในลักษณะนี้อาจจะส่งผลกระทบต่อตัวอุปกรณ์อาจก่อให้เกิดความเสียหายได้ โครงงานนี้จึงพัฒนาโปรแกรมที่สามารถดึงข้อมูลค่าสัญญาณ โดยที่ไม่มีการใช้สายสัญญาณหรือไม่มีการงัดแงะอุปกรณ์ทางอิเล็คทรอนิคแต่จะใช้เพียงการอ่านค่าจากภาพถ่ายที่ได้จากกล้องเว็บแคมเท่านั้น

โครงงานนี้จึงนำทฤษฎีและหลักการของการประมวลผลทางภาพ (Image Processing) เข้ามาช่วย โดยใช้หลักของการรู้จำอักขระ (Optical Character Recognition) มาใช้ในพัฒนากระบวนการรู้จำอักขระนั้น เป็นหลักการที่ได้รับความนิยมในด้านของการใช้แปลความหมายจากสื่อสิ่งพิมพ์ต่างๆ เช่น หน้ากระดาษหนังสือ ป้ายโฆษณา แผ่นป้ายทะเบียน หรือหน้าจอมอนิเตอร์ต่างๆ ที่ให้ความสนใจ ออกมาเป็นข้อมูลตัวอักษรที่สามารถทำการแก้ไขปรับปรุงได้ แล้วบันทึกโดยอัตโนมัติลงในคอมพิวเตอร์ได้ตลอดเวลา

ดังนั้น ผู้จัดจึงได้ทำการศึกษาหลักการประมวลผลภาพต่างๆเพื่อนำมาประยุกต์ใช้ในการสร้างโปรแกรมแปลความหมายจากรูปภาพผ่านทางกล้องเว็บแคมเพื่ออำนวยความสะดวกและเป็นอีกหนึ่งทางเลือกให้กับผู้ใช้ในการช่วยไม่ให้เป็นการเสียเวลาในการเฝ้ามองและจดบันทึกค่าเองด้วยมือ

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อพัฒนาโปรแกรมแปลความหมายช่องสัญญาณความถี่จากหน้าจอแอลซีดี

1.2.2 เพื่อหลีกเลี่ยงความเสียหายของอุปกรณ์ในกรณีที่มีการงัดแงะเกิดขึ้น

1.2.3 เพื่อเป็นทางเลือกให้กับผู้ใช้ที่ไม่ต้องการเสียเวลานั่งบันทึกค่าด้วยตัวเอง

1.3 ขอบเขตของโครงงาน

1.3.1 ตัวอักษรที่ใช้ทดสอบบนหน้าจอแอลซีดี เครื่องรับส่งสัญญาณวิทยุ จำนวน 2 ตัวและตัวเลขจำนวน 10 ตัวประกอบด้วย

- ตัวอักษรภาษาอังกฤษตัวพิมพ์ใหญ่จำนวน 2 ตัว ได้แก่ 'C' , 'H'
- ตัวเลขจำนวน 10 ตัว ได้ก่ '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'

1.3.2 ขนาดภาพ 640 x 480, ตัวอักษรมีลักษณะค่อนข้างเป็นสีดำ, ไม่เอียง

1.3.3 ลักษณะรูปแบบของตัวอักษรมีลักษณะแบบตัวแสดงผล 7 ส่วน (Seven segment)

1.3.4 การประมวลผลภาพทำในรูปภาพที่มีค่าความสว่างของแสงที่แตกต่างกันออกไป

1.3.4 ระยะของกล้องกับหน้าจอมอนิเตอร์ได้โฟกัสที่ระยะห่างประมาณ 15 – 20 เซนติเมตร

1.4 แผนการดำเนินงาน

ตารางที่ 1.1 ตารางขั้นตอนและแผนการดำเนินโครงงาน

กิจกรรม	ปี 2556		
	มี.ค.	เม.ย.	พ.ค.
1.ศึกษาค้นคว้าข้อมูลในการทำโครงงาน			
2.ศึกษาโครงสร้างภาษาแมทแลบ(MATLAB)			
3.ศึกษาข้อมูลของการรู้จำอักขระทางภาพและข้อมูลต่างๆเกี่ยวกับการประมวลผลภาพ			
4.ศึกษาการเชื่อมโยงโปรแกรมกับกล้องเว็บแคม			
5.ทำการออกแบบโปรแกรมและกำหนดส่วนประกอบ			
6.เขียนโปรแกรม			
7.ทดสอบการใช้งานและแก้ไขข้อบกพร่อง			
8.สรุปผลการทำโครงงานและจัดทำรายงาน			

1.5 ผลที่คาดว่าจะได้รับ

1.5.1 ได้โปรแกรมที่สามารถแปลความหมายช่องสัญญาณความถี่จากหน้าจอแอลซีดีได้

1.5.2 ประหยัดเวลาในการนั่งเฝ้าหน้าจอเพื่อทำการบันทึกค่า โปรแกรมจะทำการบันทึกเอง

1.6 งบประมาณของโครงงาน

1.6.1 ค่าอุปกรณ์ในการดำเนินโครงงาน	500 บาท
1.6.4 ค่าเข้าเล่มโครงงาน	500 บาท
รวมเป็นเงินทั้งสิ้น	1,000 บาท

หมายเหตุ ขออนุมัติถัวเฉลี่ยทุกรายการ

บทที่ 2
หลักการและทฤษฎีที่เกี่ยวข้อง

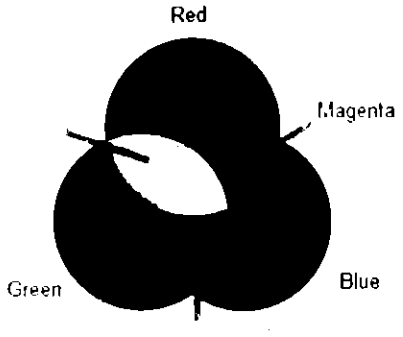
การแปลความหมายจากหน้าจอมอนิเตอร์ ก่อนที่จะได้ภาพที่สมบูรณ์พร้อมนำไปสู่กระบวนการการรู้จำอักขระนั้น รูปภาพจะผ่านกระบวนการในขั้นตอนต่างๆเพื่อจะได้ภาพที่สมบูรณ์เหมาะสมกับงานที่จะทำที่สุด โดยหลักการและทฤษฎีเหล่านั้น จะถูกกล่าวไว้ทั้งหมดในบทนี้

2.1 การประมวลผลรูปภาพดิจิตอล (Digital Image Processing)

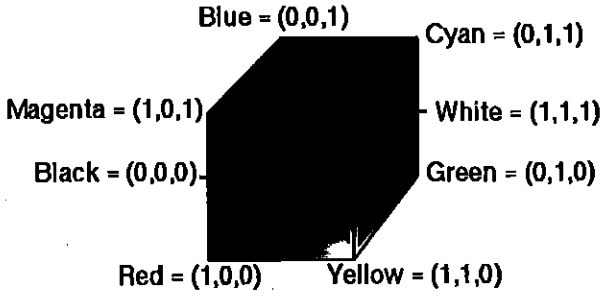
การประมวลผลรูปภาพดิจิตอล (Digital Image Processing)[1] คือ กระบวนการในการใช้คอมพิวเตอร์เข้ามาช่วยในการเข้าถึงรูปภาพในระดับพิกเซล เพื่อทำการแก้ไข ปรับปรุงหรือเพิ่มประสิทธิภาพของรูปภาพใดๆ ที่มีการเก็บข้อมูลแต่ละรูปอยู่ในรูปแบบของเมตริกซ์ ทำการแก้ไขปรับปรุงให้ได้รูปภาพออกมาตามจุดมุ่งหมาย

2.1.1 ระบบแบบจำลองสีแบบ RGB (RGB Color Model)

ประกอบไปด้วยสีพื้นฐาน 3 สี ได้แก่ แดง (Red) , เขียว (Green) และน้ำเงิน (Blue) เป็นสามสีพื้นฐานในการสร้างสีอื่นๆต่อไป ในแต่ละระนาบสีทั้ง 3 สีนี้จะประกอบไปด้วยค่าของสีและค่าความสว่าง ระดับความเข้มของสีในแต่ละพารามิเตอร์อยู่ในช่วงของตัวเลขระหว่าง 0 – 255 ระดับ สำหรับสีใดๆจะถูกอธิบายไว้ด้วยระดับสเกลของตัวเลข ตั้งแต่ 0 – 255 ตัวอย่างเช่น สีแดง จะมีค่า (255,0,0) สีเหลือง จะมีค่า (255,255,0) เป็นต้น ซึ่งถ้าค่าระดับเลเวลของ RGB เปลี่ยนไป จะทำให้ผลลัพธ์ของสีที่ได้เปลี่ยนไปด้วย (รูปที่ 2)



รูปที่ 2.1 แสดงระบบสีแบบจำลองสีแบบ RGB



รูปที่ 2.2 แสดงรูปสีเมื่อระดับสีของ RGB เปลี่ยนแปลง

ที่มา: http://prosjekt.ffi.no/unik-4660/lectures04/chapters/Introduction.html

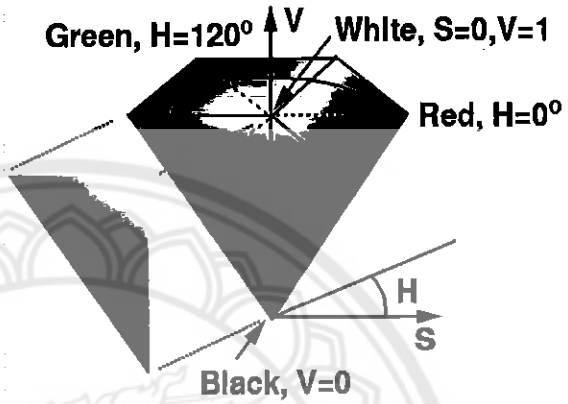
ภาพสี (Color Image)

ค่าของรูปภาพสี ประกอบไปด้วยค่าระดับสีจำนวนบิต 8 บิต ทำให้รูปภาพหนึ่งรูปภาพจะมีระดับของความเข้มแสงที่ 2^8 บิต หรือมีค่าตั้งแต่ 0 – 255 ระดับต่อหนึ่งเมตริกซ์ โดยจะประกอบไปด้วยเมตริกซ์ 3 ระนาบซ้อนทับกันอยู่ ได้แก่ แผ่นสีแดง สีเขียว และแผ่นสีน้ำเงิน หรือภาพสีหนึ่งภาพจะมีจำนวนบิตทั้งหมด 24 บิต หรือมีระดับของสีรวมกันทั้งหมด 2^{24} สี ดังรูปที่ 2.1

รูปที่ 2.3 ภาพสี และค่าในแต่ละพิกเซล

2.1.2 ระบบแบบจำลองสีแบบ HSV (HSV Color Model)

อีกระบบของสีที่ได้พูดถึงได้แก่ ระบบสีแบบ HSV เป็นการพิจารณาสีโดยใช้ค่าสีบริสุทธิ์ (Hue), ค่าความบริสุทธิ์ของสีเมื่อผสมกับแสงสีขาว (Saturation) และค่าความสว่างของแสง (Value)



รูปที่ 2.4 แสดงแบบจำลองแบบสีของ HSV

ที่มา: http://prosjekt.ffi.no/unik4660/lectures04/chapters/Introduction.html

ซึ่งค่าสีบริสุทธิ์ คือค่าของสีหลัก (แดง เขียว น้ำเงิน) มีค่าอยู่ระหว่าง 0–255 ที่มุม 0องศา ค่าสีบริสุทธิ์จะมีสีแดง ที่มุม 120 องศา มีสีเขียว ที่มุม 240 องศาจะมีสีน้ำเงิน และจะเปลี่ยนสีตามสเป็คตรัมของแสง เรื่อยๆจนครบรอบ

ค่าสีบริสุทธิ์ (Hue) คำนวณจาก RGB ได้จาก

$$Red(h) = Red - \min(Red, Green, Blue) \quad (2.1)$$

$$Green(h) = Green - \min(Red, Green, Blue) \quad (2.2)$$

$$Blue(h) = Blue - \min(Red, Green, Blue) \quad (2.3)$$

จากรูปที่ 2.4 สีบริสุทธิ์จะมีค่าหนึ่งที่เป็น 0 อย่างน้อยหนึ่งค่า แต่ถ้าค่าทั้งสองเป็น 0 ค่าสีบริสุทธิ์ จะเป็นมุมของสีและถ้าทั้งสามค่ามีค่าเท่ากับ 0 สีของค่าสีบริสุทธิ์จะเป็นสีขาว

ค่าความบริสุทธิ์ของสีเมื่อผสมกับแสงสีขาว (Saturation) คือ ระยะห่างจากสีขาว ถ้ามีค่าเท่ากับ 0 สีที่ได้จะไม่มีค่าสีบริสุทธิ์ผสมอยู่ด้วย สีที่ได้จึงเป็นสีขาวล้วน แต่ถ้าค่าที่ได้เป็น 255 จะได้เป็นสีดำ ค่าความบริสุทธิ์ของสีเมื่อผสมกับแสงสีขาว คำนวณได้จาก

$$S = \frac{\max(red,green,blue) - \min(red,green,blue)}{\max(red,green,blue)} \qquad (2.4)$$

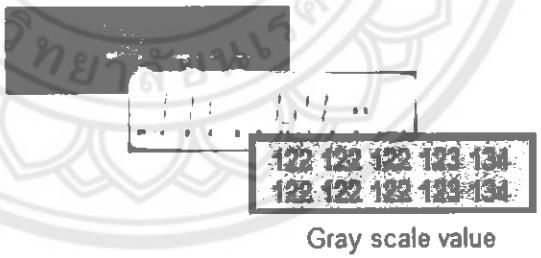
ค่าความสว่างของแสง (Value) หรือ Brightness คือ ความสว่างของสี มีค่าตั้งแต่ 0 – 100 โดยภาพจะสว่างมากขึ้นเรื่อยๆ เมื่อค่าความสว่าง มีค่าเพิ่มขึ้นเรื่อยๆ

ซึ่งค่าค่าความสว่างของแสง คำนวณได้จาก

$$V = \max(red, green, blue) \qquad (2.5)$$

2.1.3 ภาพระดับเทา (Gray Image)

ภาพระดับเทาคือการนำค่าระดับความเข้มของสีจากระบบสี RGB ทั้ง 3 แผ่นมารวมกันแล้วนำมาหาค่าเฉลี่ย ภาพที่ได้จะมีลักษณะเป็นสีเทามองด้วยสายตามนุษย์จะเห็นสีของภาพในระดับเทา 2 สีคือสีขาวกับสีดำ โดยส่วนใหญ่ค่าระดับเทามีค่าความเข้มของสีทั้งหมด 8 บิต หรือ 2^8 ค่า หรือ 256 ระดับ ดังรูปที่ 2.5 (ข) ทั้งนี้ค่าระดับความเข้มแสงอาจเปลี่ยนแปลงได้ตามที่ผู้ใช้กำหนด



รูปที่ 2.5(ก) ภาพระดับเทา และค่าในแต่ละพิกเซล

0 255

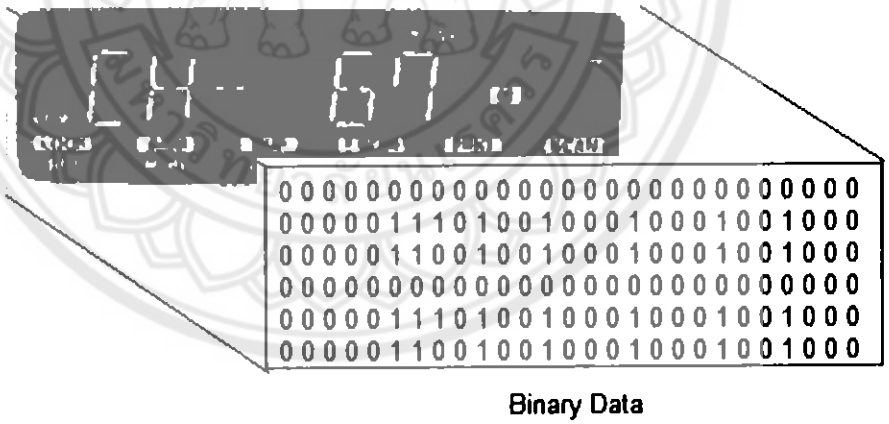
(ข)

รูปที่ 2.5(ข) ค่าระดับความเข้มของสี 8 บิต แสดงการไล่ระดับเฉดสีขาว

ที่มา: http://www.colorsimulator.com/gradent_gray_scale_test_pattern.htm

2.1.4 ภาพขาวดำ (Black & White Image)

ภาพขาวดำ คือ ภาพที่มีระดับความเข้มของสี 1 บิต หรือ 2^1 ระดับ กล่าวคือ ใน 1 พิกเซลจะมีค่าเพียง 0 หรือ 1 หรือสีดำหรือไม่ก็สีขาวเท่านั้น โดยกำหนดค่าพิกเซลที่มีค่าเท่ากับ 1 จะมีสีขาว และ พิกเซลที่มีค่าเท่ากับ 0 จะมีสีดำ ดังรูปที่ 2.6



รูปที่ 2.6 ภาพแบบไบนารี และค่าในแต่ละพิกเซล

2.1.5 รูปแบบการเก็บภาพให้อยู่ในระบบข้อมูลดิจิตอล

รูปแบบของข้อมูลภาพจะมีการกำหนดตำแหน่งเป็นลักษณะในรูปแบบของเมตริกซ์โดยที่ จำนวนหลัก(Column) และแถว (Row) จะแทนจำนวนจุดภาพของภาพ เช่น ภาพขนาด 640*480 จะหมายถึงแฟ้มข้อมูลของภาพดังกล่าวมีขนาด 640 หลักและ 480 แถว

$$\text{Image} = \begin{bmatrix} f(1,1) & f(1,2) & f(1,3) & \dots & f(1,480) \\ f(2,1) & f(2,2) & f(2,3) & \dots & f(2,480) \\ f(3,1) & f(3,2) & f(3,3) & \dots & f(3,480) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ f(640,1) & f(640,2) & f(640,3) & \dots & f(640,480) \end{bmatrix}$$

รูปที่ 2.7 เมตริกซ์ของรูปภาพขนาด 640*480

จากรูปที่ 2.7 แสดงข้อมูลภาพที่ประกอบไปด้วยจำนวนสมาชิกของเมตริกซ์ทั้งหมด i * j ตัว โดยที่ตำแหน่งในเมตริกซ์ f(i,j) คือค่าความสว่างของแสงของภาพที่จุดนั้น

2.2 การแปลงภาพสี (Image conversion)

การแปลงภาพสี (Image conversion)[3] เป็นการแปลงภาพที่บันทึกไว้ **ซึ่งปกติแล้วรูปภาพที่** บันทึกไว้จะเป็นรูปภาพสี RGB ซึ่งสามารถนำรูปภาพสีนี้ไปทำการแปลงสู่ระบบสีอื่นๆ **เพื่อง่ายต่อ** การทำกระบวรการประมวลทางภาพ โดยในโครงงานนี้มีทำการแปลงรูปภา**พสีเป็นภาพระดับเทา** และแปลงรูปภาพสีเป็นภาพแบบ HSV

2.2.1 การแปลงภาพสีเป็นภาพระดับเทา

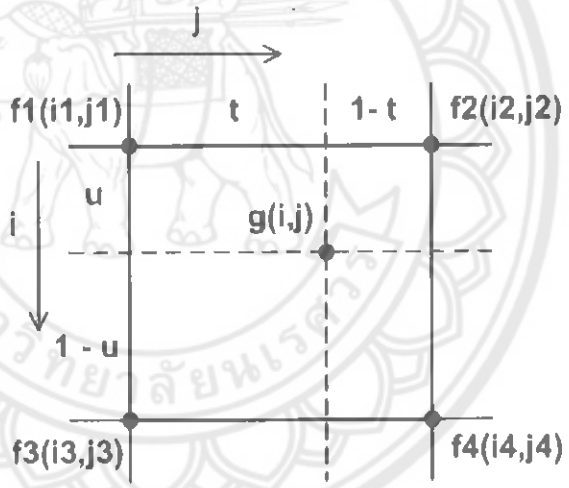
ภาพที่มนุษย์มองเห็นโดยทั่วไป จัดเป็นภาพสี ซึ่งคุณสมบัติภาพสีนี้จะมี 3 ระนาบ ทำให้ การวิเคราะห์ภาพมีความซับซ้อน และยากต่อการพิจารณา การลดทอนความเป็นสีสามารถทำได้ โดยการแปลงให้เป็นภาพระดับเทา ขั้นตอนดังกล่าวนั้นมีหลายวิธี เช่น เลือกระนาบใดระนาบหนึ่ง จากภาพสี หรือ นำทั้ง 3 ระนาบมาทำการหาค่าเฉลี่ยเพื่อให้เหลือเพียงระนาบเดียว เป็นต้น ซึ่งภาพ ระดับเทาที่ได้จะซับซ้อนน้อยกว่าภาพสีเมื่อนำมาวิเคราะห์วัตถุต่างๆ ภายในภาพ

2.2.1 การแปลงภาพสีเป็นภาพ HSV

เป็นการประยุกต์การใช้งานรูปภาพสีอีกแบบ จากคุณสมบัติที่จำกัดประการหนึ่งของภาพสีที่ค่าระดับความเข้มของสีกับค่าของความสว่างของแสงจะอยู่รวมกัน ไม่สามารถทำการแยกค่าทั้งสองนี้ออกจากกันได้ ในกรณีเมื่อต้องการหาค่าความสว่างของแสงในภาพสี จึงไม่สามารถหาค่านี้ได้ ด้วยคุณสมบัติของภาพ HSV ที่มีมิติของสี 3 ระนาบได้แก่ Hue, Saturation และ Value เมื่อแปลงภาพสีมาเป็นภาพ HSV แล้ว จะสามารถหาค่าความสว่างของแสงได้

2.3 การปรับขนาดภาพ (Image Resize)

ในโครงงานนี้ มีการใช้ขั้นตอนการปรับขนาดของภาพ (Image resize)[4] ด้วยการใช้เทคนิคไบริเนียอินเทอโพเรชั่น (Bilinear Interpolation) ซึ่งเป็นการกำหนดค่าพิกเซลในตำแหน่งใหม่ โดยใช้ค่าพิกเซลหรือค่าความเข้มแสง (intensity) เฉลี่ยของ 4 พิกเซลที่อยู่รอบซึ่งสามารถอธิบายดังรูปที่ 2.8



รูปที่ 2.8 แสดงวิธีการทำไบริเนียอินเทอโพเรชั่น (Bilinear Interpolation)

ที่มา: http://www.ecpe.nu.ac.th/panomkhawn/imagepro/pdf/ch02-part2.pdf

จากรูปที่ 2.8 จะสามารถสร้างสมการได้ดังสมการที่ 2.6

$$g(i, j) = (1-t)*(1-u)*f1(i1,j1) + t*(1-u)*f2(i2,j2) + (1-t)*u*f3(i3,j3) + t* u*f4(i4,j4) \quad (2.6)$$

2.4 การแยกภาพ (Image Segmentation)

การแยกภาพ (Image Segmentation)[5] คือ แยกวัตถุหรือตัวอักษรออกจากฉากหลังหลัง มักจะทำหลังจากปรับปรุงคุณภาพของภาพให้ดีขึ้นแล้ว ซึ่งมีความสำคัญมาก เพราะการวิเคราะห์จะไม่สามารถกระทำกับภาพทั้งภาพเพื่อให้ได้ผลลัพธ์ที่ดีได้ เพราะเนื่องจากจะช้า และจำนวนข้อมูลในการวิเคราะห์มาก แต่จะวิเคราะห์ภาพเป็นส่วนๆ ไป เช่น การหาวัตถุที่ควรจะเป็นตัวอักษร ว่าอยู่ตำแหน่งใดภายในภาพ เป็นต้น ซึ่งการแยกภาพนี้เป็นขั้นตอนสำคัญของการวิเคราะห์ภาพ

2.4.1 ประโยชน์ของการแยกภาพ

2.4.1.1 ลดข้อมูลที่ไม่จำเป็นในรูปภาพลง เนื่องจากการทำการแยกภาพ เป็นการแยกแยะระหว่างส่วนที่สนใจ เช่น ตัวอักษร กับส่วนที่ไม่ต้องการนั่นคือ ฉากหลัง เมื่อตัดข้อมูลในส่วนที่ไม่ต้องการออกไป จำนวนข้อมูลที่จำเป็นในการวิเคราะห์จะลดลงอย่างมาก และยังทำให้การประมวลผลไวขึ้นอีกด้วย

2.4.1.2 จัดระเบียบข้อมูลในรูปภาพให้เป็นกลุ่มได้ดีขึ้น ข้อมูลภาพที่ผ่านการแบ่งแยกแล้ว จะมีโครงสร้างที่ชัดเจนขึ้นและนำไปใช้งานได้สะดวกขึ้น

2.4.1.3 แสดงข้อมูลในรูปแบบที่เข้าใจง่ายยิ่งขึ้น ซึ่งข้อมูลภาพที่ผ่านการแบ่งแยกแล้วจะทำให้สามารถแยกแยะความแตกต่างและความเหมือนของรูปภาพได้

การทำการแยกภาพ เพื่อแยกข้อมูลภาพในส่วนที่ต้องการออกมาได้ วิธีการที่ใช้สำหรับการแยกภาพ ในโครงงานนี้ คือ การพิจารณาความสว่างของภาพสำหรับภาพระดับเทาด้วยการสร้างภาพแบบไบนารีด้วยการทำเทรชโฮลด์ (Threshold) โดยอธิบายได้ดังหัวก่อนหน้าแล้ว

2.5 การประมวลผลภาพกับรูปร่างและโครงสร้างของภาพ (Morphological Image Processing)

การประมวลผลภาพกับรูปร่างและโครงสร้างของภาพ (Morphological Image Processing)[5] คือ การประมวลผลภาพโดยการเปลี่ยนแปลงลักษณะรูปร่างหรือโครงสร้างของภาพ โอเปอเรชั่นพื้นฐานโดยทั่วไปมักจะกระทำกับภาพแบบไบนารี แต่ที่กระทำกับภาพระดับเทานั้นก็มี ส่วนในโครงงานนี้จะกล่าวถึงแค่ส่วนที่กระทำกับภาพแบบไบนารีเท่านั้น มีหลายโอเปอเรชั่น ได้แก่

- การขยายภาพ (Dilation) คือ การขยายภาพโดยมีสัดส่วนเท่ากันทั่วทั้งภาพ
- การย่อภาพ (Erosion) คือ การย่อภาพโดยมีสัดส่วนเท่ากันทั่วทั้งภาพ
- การโครชซิ่ง (Closing คือ การขยายภาพ (Dilation) แล้วตามด้วยการทำย่อภาพ (Erosion)

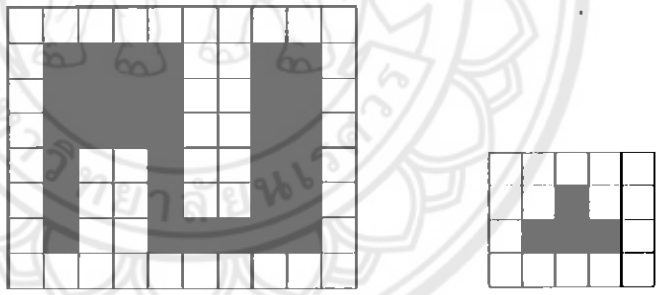
- การโอเพนนิ่ง (Opening) คือ การทำการย่อภาพ (Erosion) แล้วตามด้วยการขยายภาพ (Dilation)

ในโครงงานนี้จะกล่าวถึง เฉพาะการทำโคลสซิ่ง (Closing) เท่านั้น

2.5.1 การย่อภาพ (Erosion)

การย่อภาพเป็นลักษณะของการลบข้อมูลภาพบริเวณขอบของภาพ การย่อภาพสามารถทำได้มีลักษณะคล้ายกับการขยายภาพโดยการสร้างรูปภาพย่อย (Structuring Element) ขึ้นแล้วนำรูปภาพย่อย (Structuring Element) ไปสแกนตามข้อมูลภาพ

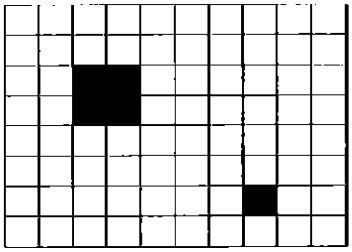
สำหรับทุกตำแหน่งที่เลื่อนรูปภาพย่อย (Structuring Element) ไปบนภาพก็จะมีการเปรียบเทียบกับข้อมูลภาพ ถ้าข้อมูลภาพมีค่าเหมือนกับรูปภาพย่อย (Structuring Element) จะทำการกำหนดค่าข้อมูลภาพในตำแหน่งที่ตรงกับจุดเริ่มต้น (Origin) ของรูปภาพย่อย (Structuring Element) ถูกกำหนดให้มีค่าเท่ากับ 1



(ก) (ข)

รูปที่ 2.9 (ก) แสดงรูปภาพเริ่มต้น (Original image)
(ข) แสดงรูปภาพย่อย (Structuring Element)

ซึ่งผลที่ได้จะมีเพียง 5 ตำแหน่งเท่านั้นที่มีค่าเหมือนกับรูปแบบ (template)

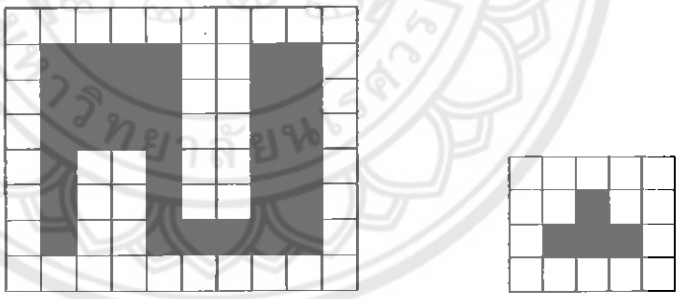


รูปที่ 2.10 แสดงผลลัพธ์ที่ได้จากการย่อภาพ (Erosion)

ผลที่ได้ตามรูปที่ 2.13 ข้อมูลภาพที่ผ่านการทำโอเปอเรชั่นกับรูปภาพย่อย (Structuring Element) แล้วพบว่ารูปภาพย่อย (Structuring Element) จะเป็นตัวกำหนดขนาดของผลลัพธ์ที่ได้

2.5.2 การขยายภาพ (Dilation)

การขยายภาพจะทำได้โดยกำหนดรูปแบบ (template) และนำรูปภาพย่อย (Structuring Element) นี้สแกนไปบนข้อมูลภาพตามลำดับตลอดทั้งภาพซึ่งในขณะที่จุดเริ่มต้น (Origin) ของรูปภาพย่อย (Structuring Element) ตรงกับตำแหน่งข้อมูลภาพที่พิกเซลมีค่าเท่ากับ 1 นั้นก็จะทำการยูเนียนรูปภาพย่อย (Structuring Element) นี้เข้ากับข้อมูลภาพดังตัวอย่าง

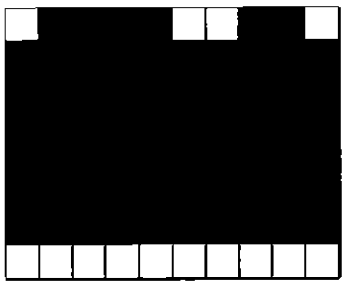


(ก) (ข)

รูปที่ 2.11 (ก) แสดงรูปภาพเริ่มต้น (Original image)

(ข) แสดงรูปภาพย่อย (Structuring Element)

ซึ่งผลลัพธ์ที่ได้จะเป็นไปตามรูปที่ 2.12

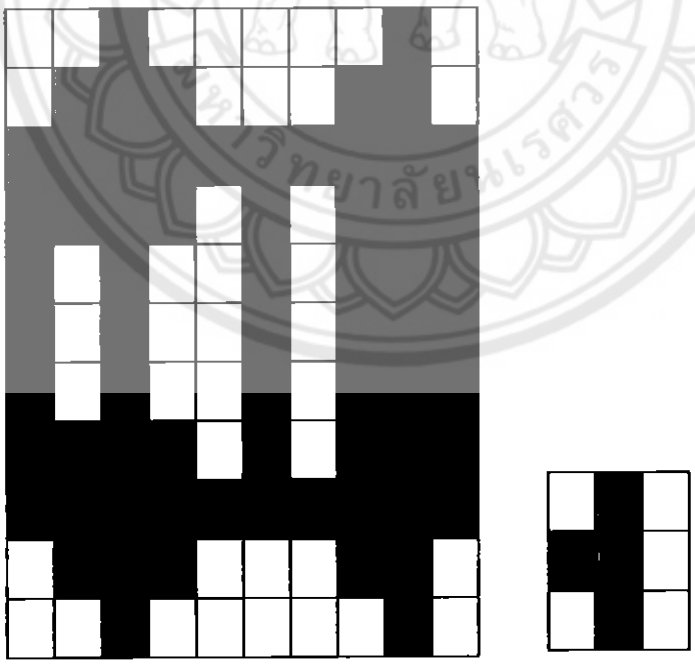


รูปที่ 2.12 แสดงผลลัพธ์ที่ได้จากการขยายภาพ (Dilation)

2.5.3 การทำโอเปอเรชั่นโคลสซิ่ง (Closing)

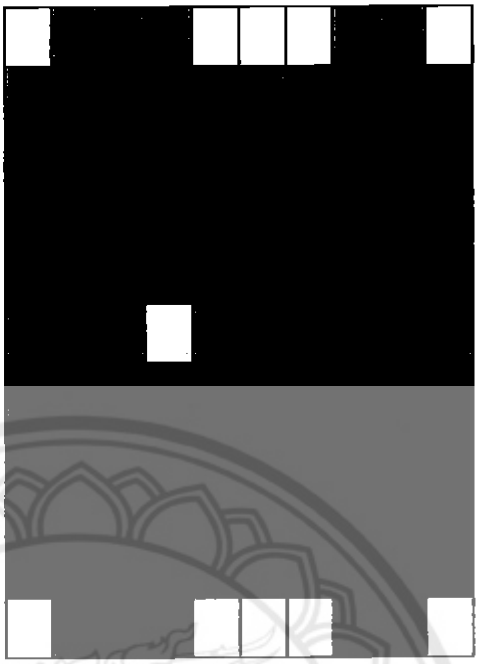
การทำโอเปอเรชั่นโคลสซิ่ง A• B = (A ⊕B) ΘB คือการนำรูปภาพมาทำกระบวนการขยายภาพ (Dilation) ก่อน จากนั้นจึงทำกระบวนการย่อภาพ (Erosion) เป็นขั้นตอนต่อไป โดยใช้รูปภาพโครงสร้างย่อย (Structuring Element) รูปเดียวกันทำตลอดทั้งกระบวนการ

ตัวอย่างขั้นการการทำกระบวนการ Closing

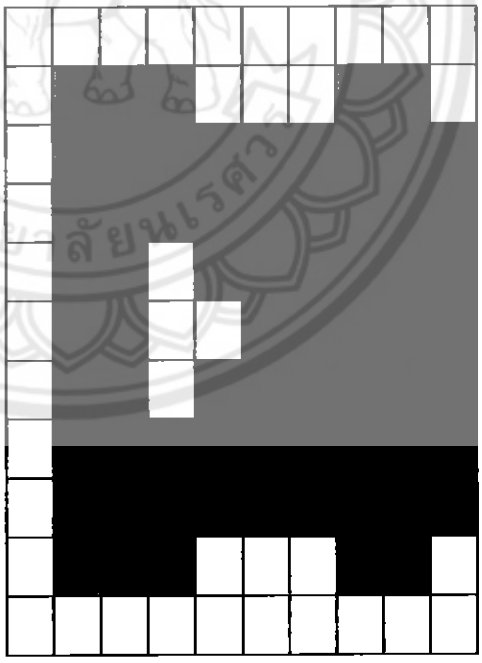


(ก) (ข)

รูปที่ 2.13 (ก) แสดงรูปภาพเริ่มต้น (Original image) (ข) แสดงรูปภาพย่อย (Structuring Element)



รูปที่ 2.14 แสดงผลลัพธ์ที่ได้จากการทำการย่อภาพ (Dilation)



รูปที่ 2.15 แสดงผลลัพธ์ของการทำโอเปอเรชั่นโอเพนนิ่ง (Closing)

2.6 การรู้จำอักขระ (Optical Character Recognition)

การรู้จำอักขระทางภาพ (Optical character recognition) เป็นการแปลความหมายจากหน้าจอมอนิเตอร์ หน้าหนังสือ ป้ายต่างๆ หรือตัวอักษรสัญลักษณ์ต่างๆที่สามารถมองเห็นได้ด้วยตา ถอดความออกมาเป็นไฟล์ในคอมพิวเตอร์ที่สามารถแก้ไขได้ในขั้นตอนสุดท้าย การจับภาพนอกจากสายตาแล้วอาจจะใช้เครื่องมืออื่นๆได้อีกเช่น กล้องดิจิตอล กล้องเว็บแคมหรือเครื่องสแกนเนอร์

โครงสร้างทั่วไปของกระบวนการการรู้จำอักขระ แบ่งออกได้เป็น 3 ขั้นตอน คือ

รูปที่ 2.16 รูปโครงสร้างของกระบวนการการรู้จำอักขระ

2.6.1 กระบวนการประมวลผลขั้นต้น (Pre Processing)

ขั้นตอนการทำงานของโปรแกรมการรู้จำอักขระทางภาพโอซี (OCR) นั้น ก่อนที่โปรแกรมจะประมวลผลออกมาได้ว่ารูปภาพที่รับเข้ามานั้นประกอบด้วยตัวอักขระอะไรบ้าง จำเป็นต้องนำไปผ่านกระบวนการขั้นตอนที่สำคัญหลายขั้นตอน ขั้นตอนหลายๆขั้นตอนดังกล่าวจะเรียกโดยรวมว่า กระบวนการประมวลผลขั้นต้น (Preprocessing) เป็นขั้นตอนในการปรับแต่งและจัดเตรียมข้อมูลให้เหมาะสมกับขั้นตอนการรู้จำต่อไป ขั้นตอนนี้มีความสำคัญต่อประสิทธิภาพโดยรวมของกระบวนการรู้จำอักระด้วยภาพของระบบ เพราะหากมีความผิดพลาดเกิดขึ้นในส่วนนี้ก็จะส่งผลกระทบไปยังส่วนถัดไปของระบบด้วย ขั้นตอนการประมวลผลเบื้องต้นในโปรแกรมที่สำคัญ ได้แก่

2.6.1.1 การกรองข้อมูลแทรกซ้อน (Noise Filtering)

การกรองข้อมูลแทรกซ้อนมีจุดประสงค์เพื่อลดทอนส่วนของรูปภาพที่เป็นสิ่งแปลกปลอมอันไม่พึงประสงค์ออกไป โดยข้อมูลแทรกซ้อนที่เกิดขึ้นส่วนใหญ่มักจะมาจากคุณภาพของเอกสารต้นฉบับที่นำมาทำการอ่าน ซึ่งเป็นต้นเหตุสำคัญที่ทำให้ความถูกต้องของโปรแกรมลดลง จึงจำเป็นที่จะต้องจัดการกับส่วนเกินเหล่านี้ออกไปให้ได้มากที่สุดเท่าที่จะเป็นไปได้ แต่ยังไม่มีวิธีการใดที่

รับรองได้ว่าสามารถจัดการกับข้อมูลแทรกซ้อนได้ โดยสมบูรณ์ ดังนั้นส่วนการรู้จำของโปรแกรมก็จะต้องมีความทนทานต่อการแทรกซ้อนเหล่านี้ได้พอสมควร

2.6.1.2 การปรับแต่งข้อมูล (Normalization)

การปรับแต่งข้อมูลเป็นการปรับภาพตัวอักษรให้อยู่ในรูปแบบที่ระบบต้องการเพื่อนำไปใช้ในขั้นต่อไป ตัวอย่างการปรับแต่งข้อมูลในโปรแกรมการรู้จำทั่วๆ ไป อาทิเช่น การปรับขนาดรูปตัวอักษร, การปรับตัวอักษรที่เอียงให้ตรง, การแปลงรูปสีหรือเกรย์สเกลให้เป็นขาวดำ หรือในทางกลับกัน การแปลงรูปขาวดำให้เป็นสีหรือเกรย์สเกล เป็นต้น

2.6.1.3 การตัดตัวอักษรที่สนใจ (Cropping)

การตัดตัวอักษรที่สนใจเป็นการตัดแยกเอาเฉพาะรูปตัวอักษรที่ต้องการออกมาจากภาพเพื่อส่งให้ขั้นตอนการรู้จำในการระบุว่ารูปตัวอักษรนั้นเป็นตัวอักษรอะไร หลักการพอสังเขปที่ใช้สำหรับการตัดรูปตัวอักษรโดยทั่วไปจะใช้พื้นที่สีขาว (สีพื้น) รอบรูปเป็นตัวกำหนดขอบเขตในการตัด ในขั้นตอนนี้มักจะประสบปัญหาที่ส่งผลกระทบต่ออัตราความถูกต้องของระบบโดยรวมอยู่สองปัญหา ปัญหาแรกคือปัญหาตัวติด เกิดจากรูปของตัวอักษรตั้งแต่สองตัวขึ้นไปมีส่วนที่เชื่อมติดกัน ทำให้ไม่สามารถแยกตัวอักษรออกจากกันโดยใช้พื้นที่สีขาวรอบๆ ได้ จำเป็นต้องหาอัลกอริธึมพิเศษมาช่วยในการแยกตัวอักษรออกจากกัน ส่วนปัญหาที่สองในทางตรงกันข้าม เป็นปัญหาตัวขาดที่รูปตัวอักษรหนึ่งๆ ถูกแยกออกเป็นส่วนๆ ทำให้เวลาตัดตัวอักษรจากตัวเดียวจะได้เป็นสองตัว ซึ่งก็ต้องหาวิธีการเฉพาะสำหรับมาจัดการอีกเช่นกัน

2.6.2 การรู้จำ (Recognition)

กระบวนการขั้นตอนนี้ถือว่าเป็นขั้นตอนสำคัญของกระบวนการรู้จำอักขระด้วยภาพมากที่สุด เพราะขั้นตอนนี้เป็นส่วนขั้นตอนที่จะบอกได้ว่าตัวอักษรที่อยู่ในรูปภาพที่รับเข้ามานั้นเป็นตัวอักษรอะไรบ้าง เช่นเดียวกับกระบวนการขั้นตอนอื่นๆ ในขั้นตอนนี้มีหลายวิธีการที่จะให้ได้มาซึ่งการแปลความหมายตัวอักษรจากรูปภาพที่รับเข้ามาได้อย่างถูกต้องแม่นยำที่สุด โดยที่ปัจจุบันนี้มีวิธีการที่ใช้ในการช่วยแปลความหมายจากรูปภาพที่เป็นที่นิยมกันอย่างแพร่หลายแบ่งออกเป็น 4 เทคนิคใหญ่ๆที่สำคัญได้แก่

- วิธีทางการเข้าคู่รูปแบบ (Template Matching)

- วิธีทางสถิติ (Statistical Approach)
- วิธีการวิเคราะห์ทางโครงสร้าง (Structural Analysis)
- วิธีทางโครงข่ายประสาทเทียม (Neural Network)

โดยที่ในการทดลองครั้งนี้ได้เลือกใช้วิธีการแรก หรือวิธีการเข้าคู่รูปแบบ (Template Matching) มาใช้ในการทดลอง โดยจะอธิบายรายละเอียดคร่าวๆของวิธีการที่จะใช้ดังนี้

2.6.3 การเข้าคู่รูปแบบ (Template Matching)

เป็นกระบวนการที่เริ่มใช้กันในยุคแรกๆของการทำกระบวนการ OCR หลักการโดยทั่วไป คือ จะต้องมีรูปแบบ (template) ที่สร้างขึ้นมาสำหรับอ่านตัวอักษร โดยมีการกำหนดตำแหน่งสำคัญที่สามารถใช้แยกแยะความแตกต่างระหว่างตัวอักษรแต่ละตัว เวลาทำงานก็ให้นำรูปภาพที่ต้องการอ่านไปทาบบนแบบเพื่อวัดความคล้ายคลึงกันของภาพกับตัวแบบ จากนั้นก็ระบุว่าเป็นรหัสตัวอักษรอะไร โดยใช้ค่าผ่านระดับหรือวิธีการบางอย่างในการตัดสิน วิธีการนี้จะค่อนข้างอ่อนไหวต่อข้อมูลแทรกซ้อน ขนาด และการเอียงของตัวอักษร จึงจำเป็นต้องมีขั้นตอนการปรับแต่งข้อมูลที่ดี นอกจากนั้นขั้นตอนการเปรียบเทียบก็ไม่ใช่ว่าสามารถเทียบกันแบบจุดต่อจุดได้ เพราะในทางปฏิบัติตัวอักษรที่ส่งเข้าสามารถมีความแปรปรวนได้หลายรูปแบบ ดังนั้นวิธีการเทียบก็ต้องมีประสิทธิภาพเพียงพอที่จะรองรับกับปัญหาดังกล่าวได้

2.6.4 กระบวนการการประมวลผลขั้นท้าย (Postprocessing)

หลังจากเสร็จสิ้นกระบวนการการรู้จำอักขระแล้ว ตัวอักษรที่ถูกตัดแยกออกมาเป็นตัวอักษรเดี่ยวที่ถูกส่งไปประมวลผลจนได้ผลลัพธ์ออกมาเป็นตัวอักขระ ไม่มีการยืนยันว่าการตีความที่ได้ออกมาจากขั้นตอนกระบวนการรู้จำนั้นจะแปลความได้ถูกต้องสมบูรณ์ เพราะไม่มีโปรแกรมการแปลความหมายใดที่จะแปลความได้ถูกต้องแม่นยำ ดังนั้นเพื่อเพิ่มประสิทธิภาพความแม่นยำให้แก่โปรแกรมจึงได้มีการเสริมส่วนการตรวจสอบและแก้ไขข้อความเข้ามาโดยในการทดลองนี้หลังจากผ่านกระบวนการเข้าคู่รูปแบบแล้ว อาจจะเกิดความผิดพลาดของโปรแกมที่อาจจะแปลความหมายของตัวอักษรได้มากกว่า 1 ตัว ออกมาทำให้โปรแกรมเกิดความผิดพลาดได้ ทำให้หลังจากการการแปลความหมายเสร็จจะทำการตรวจสอบว่าผลที่ได้ออกมาตรงกับผลก่อนที่จะทำการเข้าคู่รูปแบบหรือไม่ ซึ่งถ้าหากไม่ใช่จะต้องมีกระบวนการบางอย่างมาช่วยในการปรับปรุงให้โปรแกรมมีความสมบูรณ์และแปลความหมายได้อย่างถูกต้องและแม่นยำ

การทดลองการเลือกค่าเทรสโฮลด์โดยใช้ภาพจากกระบวนการแยกรูปภาพ

การทดลองนี้เป็นการทดลองการหาค่าเทรสโฮลด์ที่เหมาะสมเพื่อนำค่าเทรสโฮลด์ที่ได้นั้นไปทำต่อในกระบวนการของการทำขั้นตอนการแยกรูปภาพ เพื่อให้ภาพออกมามีความสมบูรณ์ของรูปภาพมากที่สุด โดยที่ในขั้นตอนแรกจะสร้างรูปภาพต้นแบบ (Template) ของรูปภาพที่ได้หลังกระบวนการแยกรูปภาพขึ้นมาก่อนหนึ่งภาพ โดยที่จะเลือกจากรูปภาพที่มีความสว่างของภาพกลางๆ เนื่องจากถ้าเลือกภาพจากค่าความสว่างน้อยๆ ตัวอักษรที่ได้จะเกิดความแตกต่างกันมากเมื่อนำไปเปรียบเทียบกับภาพที่ได้จากภาพที่มีความสว่างมากๆนั่นเอง หรือจะเลือกภาพโดยวิธีการหาค่าเฉลี่ยความสว่างของภาพ กล่าวเป็น

$$\text{ความสว่างของภาพต้นแบบ} = \sum_{i=1}^{N} \frac{template(i)}{N} \qquad (2.7)$$

เมื่อทราบหลักการและทฤษฎีต่างๆที่ใช้ในการการพัฒนาโปรแกรมแล้ว เนื้อหาบทต่อไปจะกล่าวถึงระบบแบบแผนการทำงานของโครงงานชิ้นโดยแสดงให้เห็นถึงขั้นตอนการทำงานตั้งแต่กระบวนการเริ่มต้นจนสิ้นสุดกระบวนการออกมาเป็นภาพที่สมบูรณ์

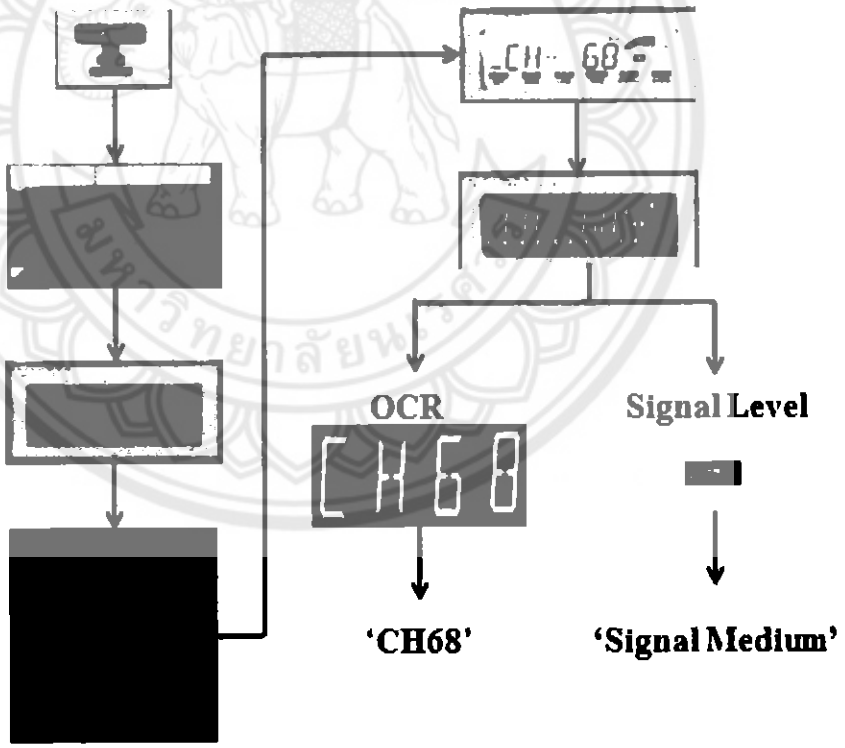
บทที่ 3

ขั้นตอนการดำเนินงาน

ในบทนี้จะกล่าวถึงขั้นตอนการดำเนินงานทั้งหมดของกระบวนการการแปลความหมายจากกล้องเว็บแคมโดยจะเริ่มตั้งแต่ขั้นตอนการรับภาพเข้ามาจนกระทั่งเสร็จสิ้นกระบวนการการประมวลผลขั้นต้น เพื่อพร้อมเข้าสู่กระบวนการแยกภาพและการเข้าคู่รูปแบบต่อไป

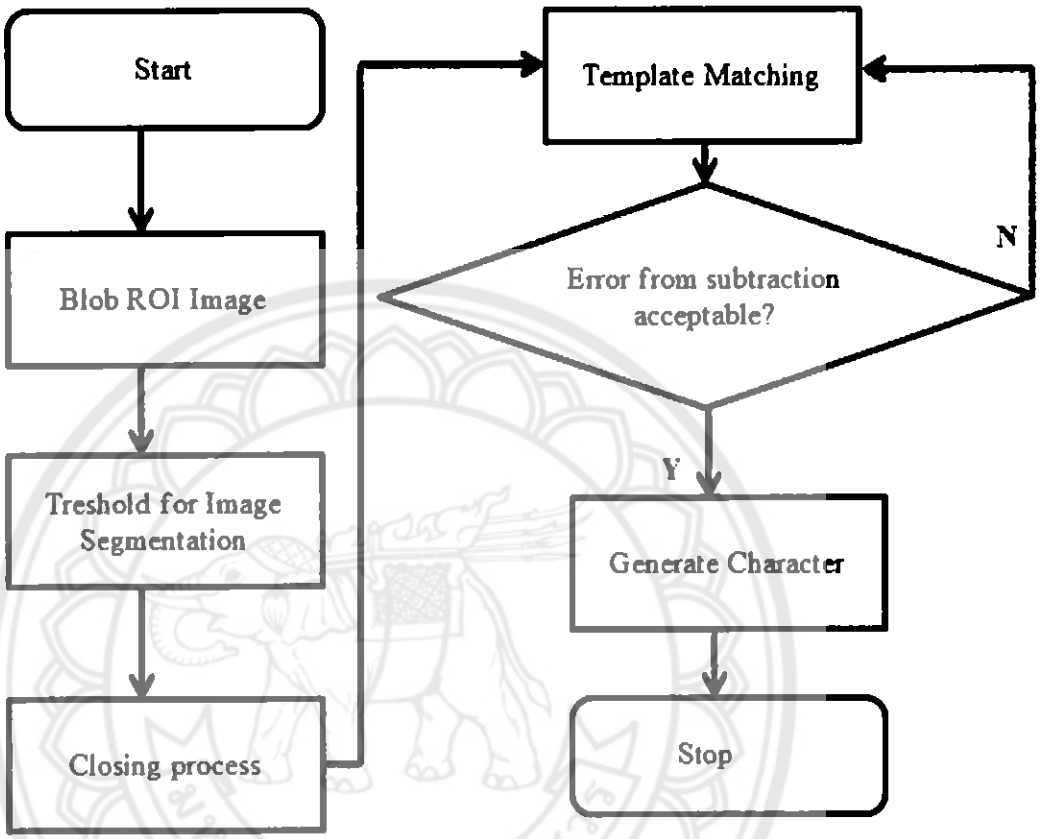
3.1 โครงสร้างของโปรแกรมแปลความหมายหน้าจอมอนิเตอร์ผ่านกล้องเว็บแคม

โปรแกรมแปลความหมายหน้าจอมอนิเตอร์ผ่านกล้องเว็บแคมประกอบไปด้วยหน้าจอของเครื่องรับสัญญาณที่สนใจ ซึ่งถูกเฝ้ามองและบันทึกภาพด้วยกล้องเว็บแคมตลอดเวลา จากนั้นนำภาพที่บันทึกได้เข้าสู่กระบวนการประมวลผลทางภาพ จนถึงขั้นตอนสุดท้ายคือการแปลความหมายจากภาพออกมาเป็นข้อมูลตัวอักษรที่สามารถแก้ไขเปลี่ยนแปลงได้



รูปที่ 3.1 ภาพโดยรวมของขั้นตอนการดำเนินโครงการ (Big Picture)

ในส่วนของโปรแกรมแปลความหมายหน้าจอมอนิเตอร์ผ่านกล้องเว็บแคม มีขั้นตอนการทำงานทั้งหมดดังรูป



รูปที่ 3.2 แสดงขั้นตอนการทำงานของโปรแกรม

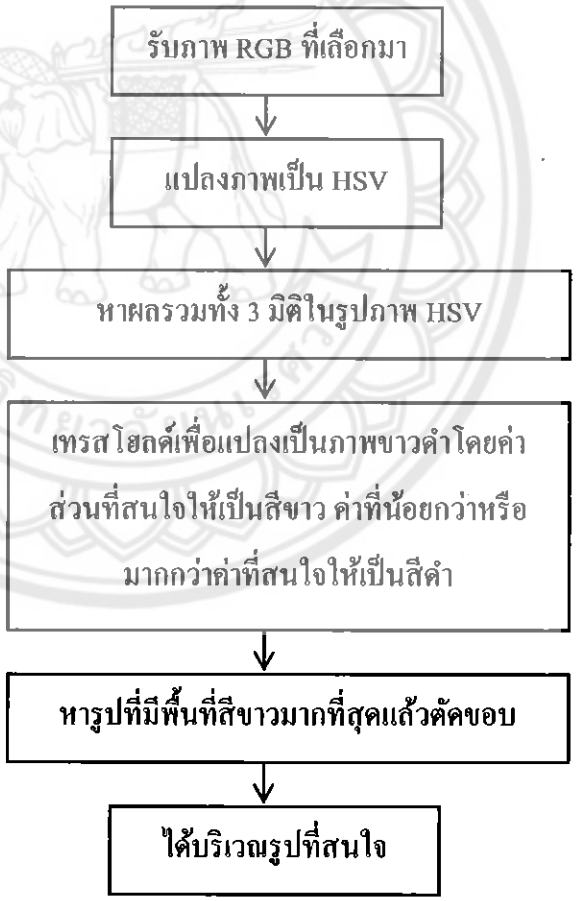
ขั้นตอนแรกจะเป็นการจับภาพของหน้าจอมอนิเตอร์ผ่านกล้องเว็บแคม โดยภาพถ่ายที่ได้จะอยู่ในรูปของเมตริกซ์ เมตริกซ์หนึ่งที่จะถูกนำไปเป็นภาพอินพุตในระบบนี้ ทำการเลือกตำแหน่งหน้าจอของมอนิเตอร์ซึ่งเป็นส่วนที่ให้ความสนใจเท่านั้น หลังจากขั้นตอนการหาพื้นที่สนใจได้แล้ว ขั้นตอนต่อไปคือการแปลงภาพให้เป็นระดับเทา ในกรณีของภาพในระบบสี RGB และแปลงภาพให้เป็นภาพแบบ HSV ของภาพในระบบสี HSV จากนั้นก็เข้าสู่กระบวนการการปรับแต่งและเพิ่มประสิทธิภาพของภาพเพื่อให้ภาพมีความสมบูรณ์ก่อนที่จะเข้าสู่กระบวนการแยกคำและนำอักษรที่ได้เข้าสู่กระบวนการการรู้จำอักขระ(Optical Character Recognition) เพื่อแปลความหมายจากภาพเป็นขั้นตอนสุดท้าย

3.2 ขั้นตอนการนำภาพเข้าสู่โปรแกรม

3.2.1 ภาพส่วนที่ให้ความสนใจ (Region of Interest Image)

เป็นขั้นตอนแรกของการประมวลผลรูปภาพ ภาพที่ได้จะเป็นภาพใหญ่ประกอบไปด้วยส่วนที่สนใจ (Region of Interest Image) และสิ่งแวดล้อมรอบๆภาพซึ่งในกรณีหลังจะไม่ให้ความสนใจ จึงต้องทำการนำภาพเฉพาะส่วนที่สนใจออกมาให้ได้เท่านั้นเพื่อนำไปเข้าสู่กระบวนการประมวลผลขั้นต้น (Preprocessing) ในลำดับต่อไป

การหาส่วนของภาพที่ให้ความสนใจนั้น สามารถหาได้จากคำสั่ง roicolor โดยทำให้บริเวณภาพที่สนใจเป็นสีขาว และบริเวณอื่นๆเป็นสีดำ จากนั้นใช้กระบวนการของการหาค่าพื้นที่ของสีขาวที่มากที่สุดในรูปภาพแล้วทำการตัดขอบรอบๆส่วนนั้น จนได้เฉพาะส่วนของรูปภาพที่สนใจออกมา ดังรูปที่ 3.3



รูปที่ 3.3 ภาพส่วนที่ให้ความสนใจ (Region of Interest Image)

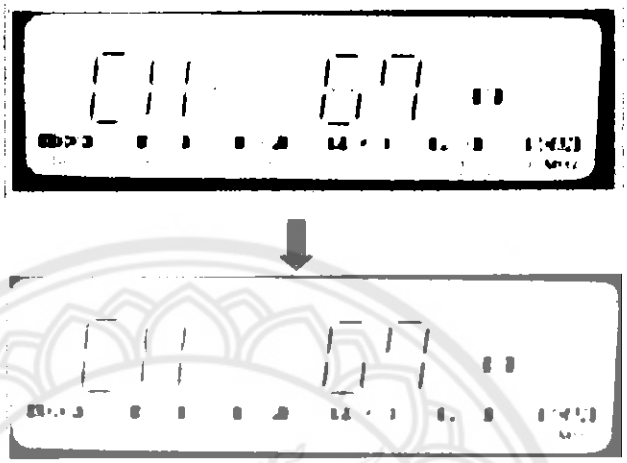
3.3 ขั้นตอนการประมวลผลขั้นต้น (Preprocessing)

ในส่วนของขั้นตอนการประมวลผลขั้นต้นโปรแกรมแปลความหมายหน้าจอมอนิเตอร์ผ่านกล้องเว็บแคม มีขั้นตอนการทำงานทั้งหมด ดังรูปที่ 3.4

รูปที่ 3.4 ขั้นตอนการทำงานของการประมวลผลขั้นต้น (Preprocessing)

3.3.1 ภาพระดับเทา (Gray Scale)

แปลงภาพจากภาพต้นฉบับที่เป็นแบบ RGB ให้เป็นภาพระดับเทา เพื่อเป็นการหาค่าระดับเทาเฉลี่ยของรูปนั้นๆและเพื่อเป็นการง่ายต่อการทำเทรสโฮลด์รูปภาพดังรูปที่ 3.5

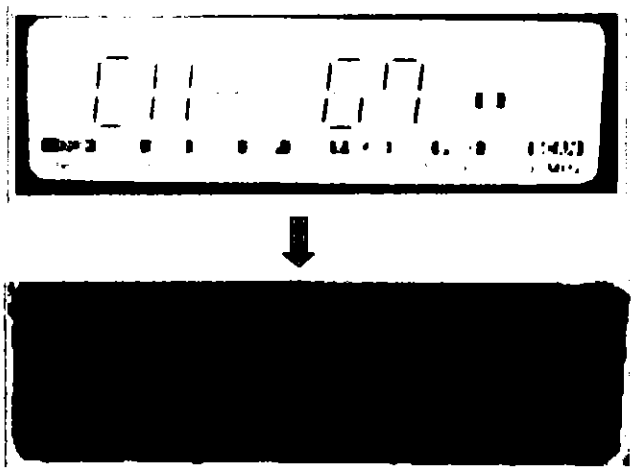


รูปที่ 3.5 แปลงภาพสีแบบ RGB เป็นภาพแบบระดับเทา

เมื่อทำการแปลงภาพจากภาพต้นฉบับให้เป็นภาพระดับเทาแล้ว ขั้นตอนต่อไปคือการใช้ เทรสโฮลด์เข้ามาช่วยในการทำให้รูปภาพระดับเทากลายเป็นรูปภาพขาวดำ ที่มีค่าระดับของสีเพียง 2 ค่า คือ 0 กับ 1 เท่านั้น

3.3.2 ภาพในระบบสี HSV (HSV image)

แปลงภาพจากภาพต้นฉบับที่เป็นแบบ RGB ให้เป็นภาพในระบบสี HSV เพื่อเป็นการหาค่าระดับค่าความสว่างของแสงเฉลี่ยของรูปนั้นๆและเพื่อเป็นการง่ายต่อการทำเทรสโฮลด์รูปภาพดังรูปที่ 3.6



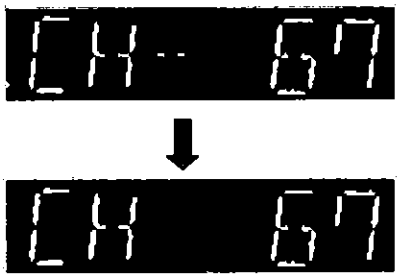
รูปที่ 3.6 แปลงภาพสีแบบ RGB เป็นภาพระบบสี HSV

ขั้นตอนการทำหมือนขั้นตอนการทำในภาพระดับเทา แต่แตกต่างกันที่ในหัวข้อนี้เป็นการแปลงจากภาพต้นฉบับมาเป็นภาพในระบบสี HSV ในแผ่นของ V ที่เป็นแผ่นค่าความสว่างของรูปภาพ เมื่อทำการแปลงภาพจากภาพต้นฉบับให้เป็นภาพในระบบสี HSV แล้ว ขั้นตอนต่อไปคือการใช้ เทรสโฮลด์เข้ามาช่วยในการทำให้รูปภาพระดับเทากลายเป็นรูปภาพขาวดำเช่นเดียวกัน

3.3.3 การกำจัดสัญญาณรบกวนและการทำกระบวนการโคลสซิ่ง (Closing Operation)

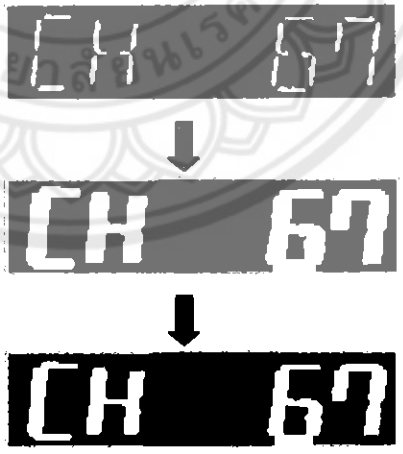
ในการประมวลผลทางภาพ สัญญาณรบกวน (noise) เป็นสิ่งที่ไม่สามารถหลีกเลี่ยงได้ ไม่ว่าจะเป็นจากขั้นตอนการแปลงภาพไปเป็นรูปภาพแบบอื่น หรือจากขั้นตอนของการทำเทรสโฮลด์ ซึ่งถ้าไม่กำจัดในส่วนนี้อาจจะส่งผลกระทบต่อการแยกคำเพื่อนำไปประมวลผลในขั้นตอนต่อไปได้

หลังจากที่ได้ภาพที่ผ่านกระบวนการทำเทรสโฮลด์แล้ว ภาพที่ได้จะนำมาทำการกลับสี (invert) ของรูปภาพก่อน จากนั้นจะเป็นขั้นตอนของการกำจัดสัญญาณรบกวนแบบหยาบก่อน โดยกำหนดว่าวภาพใดที่มีขนาดระดับพิกเซลไม่เกิน 20 หน่วย จะถูกลบออกไปจากภาพ ดังภาพที่แสดงไว้ ดังรูปที่ 3.7



รูปที่ 3.7 แสดงการกำจัดสัญญาณรบกวนรูปภาพ

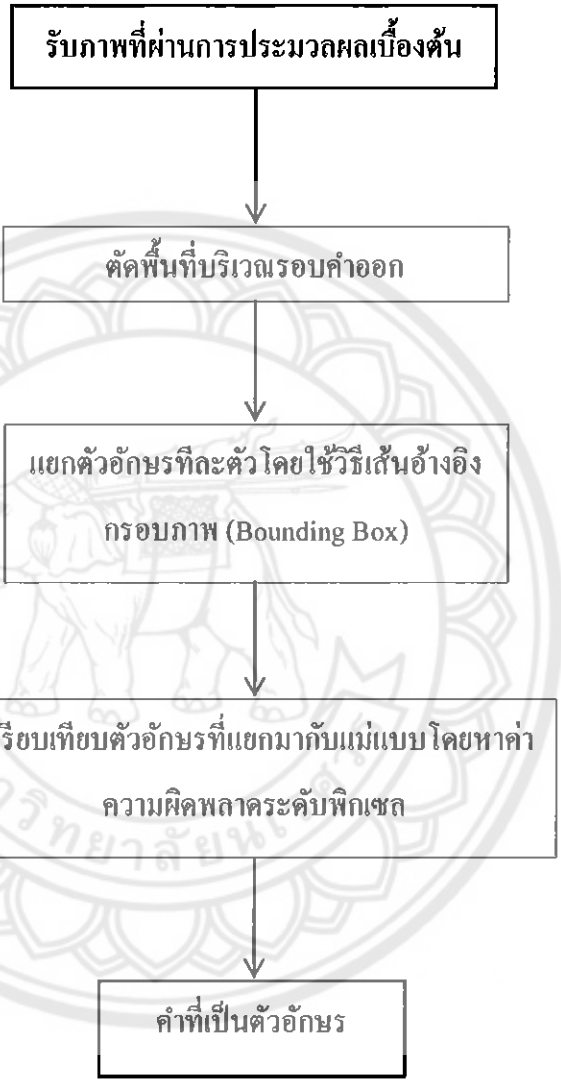
หลังจากขั้นตอนการกำจัดสัญญาณรบกวนแล้ว จะเข้าสู่กระบวนการโคลสซิ่ง เนื่องจากรูปภาพที่ได้ลักษณะตัวอักษรเป็นรูปแบบของตัวแสดงผล 7 ส่วน (Seven segment) รูปภาพที่ได้จึงมีลักษณะแยกออกจากกัน ไม่มีความต่อเนื่องของเส้น จึงใช้กระบวนการโคลสซิ่งเข้ามาช่วยปรับปรุงรูปภาพให้มีลักษณะของรูปแบบตัวอักษรที่ต่อเนื่องโดยไม่แยกออกจากกัน โดยขั้นตอนแรกคือการขยายรูปภาพของตัวอักษรแบบตัวแสดงผล 7 ส่วน ให้ส่วนที่ขาดในแต่ละตัวอักษรติดกันก่อน แต่ลักษณะตัวอักษรที่ได้หลังจากนี้จะมีลักษณะอวบอ้วนจึงเข้าสู่กระบวนการย่อขนาดตัวอักษรลงในขั้นตอนต่อมา จนได้ลักษณะของตัวอักษรที่น่าพอใจหรือมีความคล้ายคลึงกับตัวอักษรต้นแบบ (Template) มากที่สุด ดังแสดงไว้ในรูปตัวอย่างที่ 3.8



รูปที่ 3.8 รูปที่ผ่านกระบวนการโคลสซิ่ง

3.4 การรู้จำอักขระทางภาพ (Optical Character Recognition (OCR))

ในส่วนของขั้นตอนการรู้จำอักขระทางภาพของโปรแกรมแปลแปลความหมายจากรูปภาพ มีขั้นตอนการทำงานทั้งหมด ดังรูปที่ 3.9

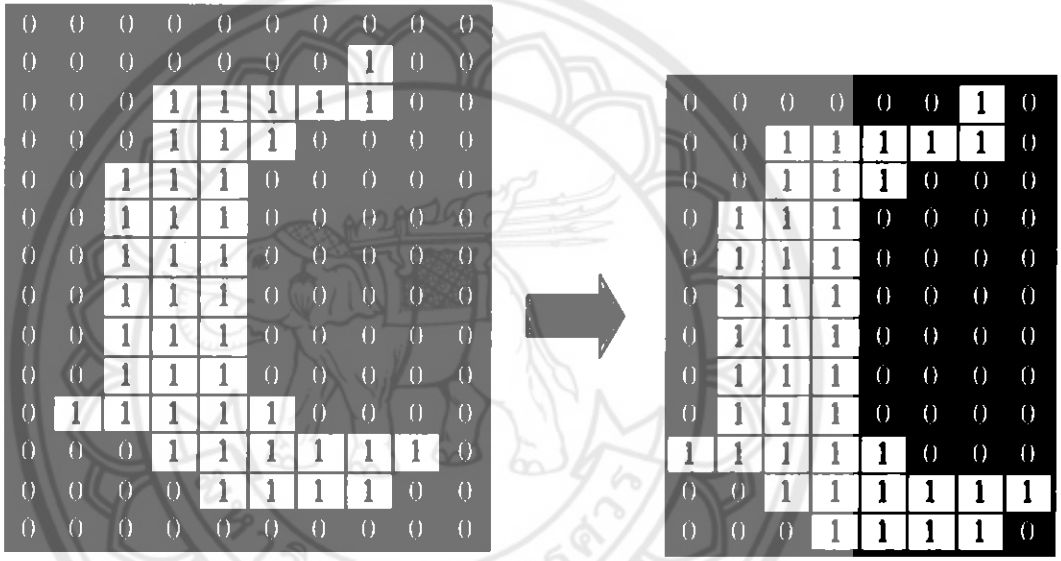


รูปที่ 3.9 รูปแสดงขั้นตอนการรู้จำอักขระทางภาพ (OCR)

3.4.1 การตัดพื้นที่บริเวณรอบคำ (Crop Image)

ขั้นตอนของการตัดตัวอักษรออกเพื่อแบ่งตัวอักษรให้แยกออกจากกันอย่างสิ้นเชิง ซึ่งวิธีการในการตัดนั้นมีหลายทฤษฎีและอยู่ที่ความถนัดในการนำมาใช้ของแต่ละคน เพื่อที่สุดท้ายแล้วทำให้ได้ตัวอักษรที่พร้อมจะนำไปประมวลผลในขั้นตอนต่อไป

โดยวิธีการตัดในการทดลองนี้ จะใช้วิธีการที่พิจารณาจากตำแหน่งพิกเซลสีขาวที่กว้างที่สุดของขอบบนและขอบล่าง และของด้านซ้ายสุดและขวาสุด ที่เป็นพื้นที่ของตัวอักษรตัวนั้นๆ ดังแสดงไว้ ดังรูปที่ 3.10



รูปที่ 3.10 แสดงการตัดพื้นที่ให้เหลือเฉพาะพื้นที่ตัวอักษรที่ต้องการ

3.4.2 การตัดตัวอักษรโดยวิธีเส้นอ้างอิงกรอบภาพ (Bounding Box)

หลังจากได้รูปภาพที่ผ่านขั้นตอนการกำจัดสัญญาณรบกวนแล้ว จะได้รูปภาพที่พร้อมเข้าสู่กระบวนการตัดตัวอักษรเพื่อนำตัวอักษรที่ตัดได้แต่ละตัวเข้าสู่กระบวนการการรู้จำอักขระต่อไป โดยวิธีการตัดตัวอักษร จะใช้วิธีการตัวจากเส้นอ้างอิงกรอบภาพ ซึ่งทำให้ภาพตัวอักษรที่ได้ออกมามีความเล็กใหญ่ของรูปภาพที่แตกต่างกันออกไป ทำให้ต้องนำภาพตัวอักษรที่ตัดได้ไปทำให้มีขนาดเท่ากับขนาดของตัวอักษรต้นแบบเสียก่อน โดยในการทดลองนี้ได้กำหนดภาพตัวอักษรต้นแบบให้มีขนาด 42 x 24 พิกเซล โดยรูปภาพที่พร้อมเข้าสู่กระบวนการตัดแสดงไว้ ดังรูปที่ 3.11

รูปที่ 3.11 แสดงรูปภาพที่พร้อมสู่กระบวนการตัดรูป

โดยในโปรแกรมจะตัดตัวอักษรตัวแรกที่ถูกแยกออกมา แล้วไล่ตัดไปจนครบทุกตัวอักษร แล้วทำการปรับขนาดเพื่อให้มีขนาดเท่ากับรูปภาพต้นแบบ เพื่อนำไปเปรียบเทียบกับรูปตัวอักษรต้นแบบ ในการที่จะระบุว่าตัวอักษรในภาพคือตัวอะไร ดังตารางที่ 3.1

รอบที่	ตัวอักษรแรก	ตัวอักษรที่เหลือ	ปรับขนาด [42x24]
1			
2			
3			
4		-	

ตารางที่ 3.1 ตารางแสดงรอบการแยกตัวอักษรโดยวิธีเส้นอ้างอิงกรอบภาพ

3.4.3 การระบุตัวอักษรโดยวิธีทางการเข้าคู่รูปแบบ (Template Matching)

เป็นกระบวนการที่เริ่มใช้กันในยุคแรกๆของการทำกระบวนการกระบวนการรู้จำอักขระหลักการโดยทั่วไปคือ จะต้องมีภาพต้นแบบ ที่สร้างขึ้นมาสำหรับอ่านตัวอักษร โดยมีการกำหนดตำแหน่งสำคัญที่สามารถใช้แยกแยะความแตกต่างระหว่างตัวอักษรแต่ละตัว กระบวนการทำงานคือ นำรูปภาพที่ต้องการเปรียบเทียบไปทาบบนรูปภาพต้นแบบเพื่อวัดความคล้ายเสมือน จากนั้นก็ระบุว่าเป็นรหัสตัวอักษรอะไร โดยใช้ค่าผ่านระดับหรือวิธีการบางอย่างในการตัดสิน วิธีการนี้จะค่อนข้างอ่อนไหวต่อข้อมูลแทรกซ้อน ขนาด และการเอียงของตัวอักษร จึงจำเป็นต้องมีขั้นตอนการปรับแต่งข้อมูลที่ดี นอกจากนั้นขั้นตอนการเปรียบเทียบก็ไม่ใช่ว่าสามารถเทียบกันแบบจุดต่อจุดได้ เพราะในทางปฏิบัติตัวอักษรที่ส่งเข้าสามารถมีความแปรปรวนได้หลายรูปแบบ ดังนั้นวิธีการเทียบก็ต้องมีประสิทธิภาพเพียงพอที่จะรองรับกับปัญหาดังกล่าวได้

3.4.4 หาค่าความผิดพลาดระดับพิกเซลในรูปเพื่อทำการจับคู่ตัวอักษร

การทดลองการหาค่าความผิดพลาดระดับพิกเซลในกรณีที่ตัวอักษรที่อ่านได้นำมาเปรียบเทียบกับรูปต้นแบบที่เป็นตัวอักษรเดียวกัน

ภาพตัวอักษรที่เป็นต้นแบบของตัวอักษร 'C' ขนาด 42 x 24 พิกเซล ผลรวมของพิกเซลสีขาวมีค่า 351 จาก 1008 พิกเซล นำมาลบกับภาพตัวอักษรที่ตัดมาได้จากกระบวนการการประมวลผลขั้นต้น ซึ่งเป็นตัวอักษร 'C' เช่นเดียวกัน โดยผลรวมของพิกเซลสีขาวในรูปนี้มีค่าเท่ากับ 358 พิกเซล

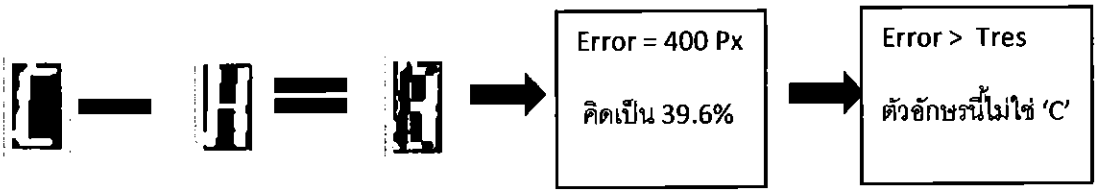
(a) Template (b) 'C' crop (c) output

รูปที่ 3.12 แสดงตัวอย่างการเข้าคู่รูปแบบกรณีตัวอักษรเดียวกัน

จากการนำภาพทั้งสองมาลบกัน ค่าความผิดพลาดระดับพิกเซลจากตัวอย่างนี้จะอยู่ที่ 97 จุด ซึ่งเมื่อนำมาผ่านเทรสโฮลด์ค่าค่าหนึ่งที่มีหน้าที่ในการตัดสินใจว่า รูปภาพที่ได้จากกระบวนการ การประมวลผลขั้นต้น รูปนั้นจะเป็นรูปเดียวกับรูปภาพต้นแบบหรือไม่ โดยจะดูจากค่าเทรสโฮลด์ค่าหนึ่งที่ได้ตั้งไว้ ในการทดลองนี้ได้กำหนดเทรสโฮลด์ค่านี้ไว้ว่า ถ้าค่าความผิดพลาดระดับพิกเซลหลังกระบวนการการทำการลบกันมีค่าน้อยกว่า 15% พิกเซลให้ถือว่า รูปภาพที่ได้จากกระบวนการการประมวลผลขั้นต้นเป็นตัวอักษรเดียวกันกับรูปภาพต้นแบบ ที่กำลังเปรียบเทียบอยู่

การทดลองการหาค่าความผิดพลาดระดับพิกเซลในกรณีที่ตัวอักษรที่อ่านได้นำมาเปรียบเทียบกับรูปภาพต้นแบบที่เป็นตัวอักษรต่างชนิดกัน

ขั้นตอนนี้เป็นการนำตัวอักษรที่ได้จากกระบวนการการประมวลผลขั้นต้น ซึ่งเป็นตัวอักษรอื่นที่ไม่ใช่ตัว 'C' ในตัวอย่างนี้นำตัวอักษร 'H' มาทำการทดลอง



รูปที่ 3.13 แสดงการเข้าคู่รูปแบบกรณีคนละตัวอักษร

จากการนำภาพทั้งสองมาลบกัน ค่าความผิดพลาดระดับพิกเซลจากตัวอย่างนี้จะอยู่ที่ 400 จุด หรือคิดเป็นร้อยละของความผิดพลาดจะได้เท่ากับ 39.6% ซึ่งค่าที่ได้เมื่อนำไปเปรียบเทียบกับค่าเทรสโฮลด์ที่ได้กำหนดไว้แล้วจะได้ว่า ค่าความผิดพลาดระดับพิกเซลมีค่ามากกว่าเทรสโฮลด์ ทำให้ได้ผลสรุปออกมาได้ว่า รูปภาพที่ได้จากกระบวนการการประมวลผลขั้นต้นเป็นคนละตัวอักษรกับรูปภาพต้นแบบที่กำลังเปรียบเทียบอยู่

หลังจากจบบทนี้ จะเห็นถึงระบบการทำงานทั้งหมดของโปรแกรม เริ่มตั้งแต่การรับภาพ นำภาพเข้าสู่กระบวนการการประมวลผลขั้นต้นจนได้ภาพที่พร้อมจะเข้าสู่กระบวนการการปมวลผลทางภาพดิจิตอล บทต่อไปจะเป็นการทดลองการเลือกค่าเทรสโฮลด์ที่เหมาะสมกับรูปภาพที่มีตัวแปรที่สำคัญอยู่ 2 อย่าง คือค่าความสว่างของแสงและขนาดของรูปภาพ เพื่อให้ได้รูปภาพที่ออกมาสมบูรณ์มากที่สุด

บทที่ 4

ผลการทดลอง

หลังจากศึกษาทฤษฎีและหลักการที่ใช้ในบทที่ 2 และกระบวนการขั้นตอนการทำงานของทั้งระบบในบทที่ 3 แล้ว บทนี้จะนำผลการทดลองที่ได้ โดยแสดงวิธีการได้มาของผลการทดลองว่าแต่ละผลการทดลองมีขั้นตอนอย่างไรและผลการทดลองที่ได้ออกมาในรูปแบบไหน

4.1 การทดลองหาค่าเทรสโฮลด์ที่เหมาะสมของรูปภาพในขั้นตอนการประมวลผลขั้นต้น

ทำการทดลองด้วยวิธีการลองผิดลองถูก (Trial and error method) โดยเริ่มไล่หาค่าเทรสโฮลด์จากค่าที่น้อยที่สุดที่สามารถทำการเทรสโฮลด์ออกมาได้ โดยค่าเทรสโฮลด์ที่เริ่มแปลงรูปภาพ HSV มาเป็นรูปภาพแบบขาวดำได้ มีค่าน้อยสุดอยู่ที่ 0.55 ซึ่งก็เป็นเพียงค่าเริ่มต้น ที่ยังไม่ใช่ค่าที่ทำให้รูปภาพออกมาสมบูรณ์มากนัก ทำจนถึงค่าเทรสโฮลด์สุดท้ายที่มีค่าเข้าใกล้ 1.0 หลังสิ้นสุดกระบวนการ จะพิจารณาค่าความผิดพลาดระดับพิกเซลว่าที่ค่าเทรสโฮลด์ใดมีค่าความผิดพลาดน้อยสุด เป็นการเสร็จสิ้นกระบวนการของ 1 ค่าระดับความสว่างของรูปภาพ โดยต้องทำซ้ำจนครบค่าความสว่างของรูปภาพที่กำหนดขึ้นมาทุกค่าความสว่างแสง

4.1.1 ตัวอย่างการทดลองกรณีที่ 1

การทดลองนี้ เลือกใช้ค่าเทรสโฮลด์ที่ 0.95 นำมาลบกับภาพที่ได้จากหลังจากเสร็จสิ้นในขั้นตอนการประมวลผลขั้นต้น ภาพที่ได้จะถูกนำมาปรับขนาดให้มีขนาดเท่ากับรูปภาพต้นแบบก่อนการนำไปลบกัน ดังรูปตัวอย่างต่อไปนี้

รูปที่ 4.1 (ก) รูปภาพต้นแบบ

รูปที่ 4.1 (ข) รูปภาพที่ได้จากกระบวนการประมวลผลขั้นต้นด้วยค่าเทรสโฮลด์ 0.95 จากรูปต้นฉบับที่มีค่าความสว่างของแสงเฉลี่ยนอยู่ในช่วง [0.41-0.50]

รูปที่ 4.1 (ค) รูปผลลัพธ์จากรูปภาพต้นแบบลบรูปภาพที่จากกระบวนการประมวลผลขั้นต้น

จากตัวอย่างข้างต้นเป็นการแสดงวิธีการหาค่าเทรสโฮลด์ที่เหมาะสม โดยนำภาพที่ได้จากการทำเทรสโฮลด์ค่าหนึ่งมาลบกับรูปภาพต้นแบบ จากนั้นนำภาพผลลัพธ์ที่ได้จากการลบกันมาหาค่าพิกเซลสีขาวรวมบนแผ่นรูปภาพที่เป็นผลลัพธ์แล้วไปทำเปรียบเทียบเป็นอัตราส่วนร้อยละออกมา ซึ่งค่าที่ได้ก็คือค่าความผิดพลาดระดับพิกเซลนั้นเอง จากตัวอย่างข้างต้นสามารถคำนวนค่าความผิดพลาดระดับพิกเซลจากการลบกันได้เท่ากับ 320 พิกเซล คิดเป็นร้อยละของความผิดพลาดได้เท่ากับ 4.3%

4.1.2 ตัวอย่างการทดลองกรณีที่ 2

ในกรณีการทำเทรสโฮลด์ด้วยค่าเทรสโฮลด์ที่ 0.99 แล้วเกิดการขาดความสมบูรณ์ของรูปภาพอย่างชัดเจน จะถือว่าเป็นค่าเทรสโฮลด์ที่ไม่เหมาะสมในการนำมาใช้ ดังรูปตัวอย่างต่อไปนี้

รูปที่ 4.2 (ก) รูปภาพต้นแบบ

รูปที่ 4.2 (ข) รูปภาพที่ได้จากกระบวนการประมวลผลขั้นต้นด้วยค่าเทรสโฮลด์ 0.99

จากรูปต้นฉบับที่มีค่าความสว่างของแสงเฉลี่ยนอยู่ในช่วง [0.41-0.50]

รูปที่ 4.2 (ค) รูปผลลัพธ์จากรูปภาพต้นแบบลบรูปภาพที่จากกระบวนการประมวลผลขั้นต้น

ผลลัพธ์ที่ได้ในกรณีนี้รูปภาพมีความผิดเพี้ยนไปจากเดิมเป็นอย่างมาก จึงถือว่าค่าเทรสโฮลด์ในช่วงนี้ไม่เหมาะในการนำมาใช้ เพราะมีค่าความผิดพลาดระดับพิกเซลจากการลบกันได้เท่ากับ 2,954 พิกเซล คิดเป็นร้อยละของความผิดพลาดได้มากถึง 39.6% ซึ่งค่าที่ได้ของการทดลองทั้งหมดได้แสดงไว้ที่ตารางข้างล่าง ดังต่อไปนี้

4.2 ผลการทดลองหาค่าเทรสโฮลด์ที่เหมาะสมของรูปภาพ

ตารางที่ 4.1 แสดงผลการทดลองการเลือกค่าเทรสโฮลด์ด้วยระบบสีแบบ HSV

$\overline{V}$ / Tresh	[0.11 - 0.20]		[0.21-0.30]		[0.31-0.40]		[0.41-0.50]		[0.51 -0.70]	
	Δ Pixel	%error	Δ Pixel	%error	Δ Pixel	%error	Δ Pixel	%error	Δ Pixel	%error
0.55	1997	26.8%	1997	26.8%	1997	26.8%	1997	26.8%	1997	26.8%
0.60	1729	23.1%	1663	22.3%	1786	23.9%	1711	22.9%	1724	23.1%
0.65	1522	20.4%	1527	20.5%	1537	20.6%	1850	24.8%	1435	19.2%
0.70	1437	19.3%	1504	20.2%	1440	19.3%	881	11.8%	796	10.7%
0.75	1156	15.5%	1197	16.0%	1091	14.6%	530	7.1%	530	7.1%

0.80	889	11.9%	860	11.5%	901	12.1%	451	6.0%	493	6.6%
0.85	412	5.5%	857	11.5%	615	8.2%	320	4.3%	533	7.1%
0.90	412	5.5%	584	7.8%	524	7.0%	320	4.3%	320	4.3%
0.95	367	4.9%	240	3.2%	101	1.4%	320	4.3%	320	4.3%
0.99	496	6.6%	470	6.3%	730	9.8%	2957	39.6%	2954	39.6%

จากตารางข้างต้น ผลต่างระหว่างพิกเซล (Δ Pixel) คือการนำภาพที่เสร็จสิ้นจากกระบวนการประมวลผลขั้นต้นแล้วนำไปลบกับรูปภาพต้นแบบ ผลที่ได้คือความผิดพลาดระดับพิกเซลของสองภาพที่นำมาลบกัน จากนั้นก็คิดเป็นร้อยละของความผิดพลาดเป็นลำดับต่อมา ซึ่งผลการทดลองที่ได้ออกมา ค่าเทรสโฮลด์ที่ 0.95 จะได้ภาพที่มีความผิดพลาดน้อยที่สุด โดยมีความเป็นเชิงเส้นแบบเส้นตรง ที่ค่าความสว่างแสงเฉลี่ยในช่วง

[0.11 - 0.20] Error ของรูปภาพจะอยู่ที่ 4.9%

[0.21 - 0.30] Error ของรูปภาพจะอยู่ที่ 3.2%

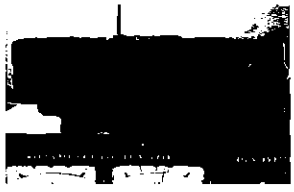
[0.31 - 0.40] Error ของรูปภาพจะอยู่ที่ 1.4%

[0.41 - 0.50] Error ของรูปภาพจะอยู่ที่ 4.3%

[0.51 - 0.70] Error ของรูปภาพจะอยู่ที่ 4.3%

4.3 ผลการทดลองของรูปที่ได้หลังการหาค่าเทรสโฮลด์ที่เหมาะสมแบบ HSV

ตารางที่ 4.2 แสดงผลลัพธ์ที่ได้หลังจากหาค่าเทรสโฮลด์ที่เหมาะสมแล้วในระบบสี HSV

Picture	V mean	ROI Original	Segmentation
	0.122		
	0.286		
	0.361		
	0.574		
	0.640		

จากตารางข้างต้นคือข้อมูลรูปภาพ ที่จะใช้ในการทดลอง V mean คือค่าเฉลี่ยของแสงของรูปภาพในระบบสีแบบ HSV ในมิติของแผ่น V ที่จะใช้เป็นเกณฑ์ในการเลือกเทรสโฮลด์ สำหรับทำการหาพื้ที่สนใจ (ROI) ของรูปภาพ ในคอลัมน์ที่ 3 เป็นรูปภาพต้นฉบับหลังจากที่หาพื้นที่สนใจออกมาได้แล้ว จนได้ออกมาเป็นภาพสุดท้ายที่เสร็จสิ้นในกระบวนการประมวลผลขั้นต้น ซึ่งเป็นขั้นตอนสุดท้ายก่อนที่จะนำไปตัดแล้วเข้าสู่กระบวนการการเข้าคู่รูปแบบ ต่อไป

หลังจากที่ทำการทดลองในระบบสีแบบ HSV เสร็จแล้ว ต่อไปจะเป็นการทดลองในส่วนของระบบสีแบบ RGB ซึ่งการทดลองทุกอย่างจะเหมือนในขั้นตอนของในการทดลองในระบบสีแบบ HSV ทุกประการ ผลการทดลองที่ได้สรุปออกมาเป็นตารางดังนี้

ตารางที่ 4.3 แสดงการทดลองการเลือกค่าเทรสโฮลด์ด้วยระบบสีแบบ RGB

$\overline{V}$ / Tresh	[21 - 60]		[61 - 80]		[81 - 100]		[101 - 120]		[121 - 140]	
	Δ Pixel	%error	Δ Pixel	%error	Δ Pixel	%error	Δ Pixel	%error	Δ Pixel	%error
0.35	1997	26.8%	1997	26.8%	1997	26.8%	1997	26.8%	1997	26.8%
0.40	1997	26.8%	1997	26.8%	1997	26.8%	1732	23.2%	1994	26.7%
0.45	1997	26.8%	1997	26.8%	1997	26.8%	1416	19.0%	1630	21.8%
0.50	1997	26.8%	1997	26.8%	1997	26.8%	637	8.6%	882	11.8%
0.55	1997	26.8%	1997	26.8%	1997	26.8%	465	6.2%	285	3.8%
0.60	1997	26.8%	1997	26.8%	1997	26.8%	228	3.1%	233	3.1%
0.65	1571	21.1%	1997	26.8%	1997	26.8%	355	4.8%	512	6.9%
0.70	1575	21.1%	1659	22.2%	1781	23.9%	501	6.7%	754	10.1%
0.75	643	8.6%	1082	14.6%	1052	14.1%	926	12.4%	1402	18.8%
0.80	430	5.8%	643	8.6%	608	8.1%	2959	39.7%	5093	68.3%

0.85	346	4.6%	396	5.3%	340	4.6%	5465	73.2	5093	68.3%
0.90	223	3.0%	136	1.8%	65	0.9%	5465	73.2	5093	68.3%
0.95	1117	15.0%	809	10.8%	684	9.2%	5465	73.2	5093	68.3%
0.99	5465	73.2%	5456	73.1%	5349	71.7%	5465	73.2	5093	68.3%

จากตารางข้างต้น ผลต่างระหว่างพิกเซล (Δ Pixel) คือการนำภาพที่เสร็จสิ้นจากกระบวนการประมวลผลขั้นต้นนำไปลบกับรูปภาพต้นแบบ ผลที่ได้คือความผิดพลาดระดับพิกเซลของสองภาพที่นำมาลบกัน จากนั้นก็คิดเป็นร้อยละของความผิดพลาดเป็นลำดับต่อมา ซึ่งผลการทดลองที่ได้ออกมา ค่าเทรสโฮลด์ที่ได้ออกมาจะมีค่าที่ทำให้รูปภาพออกมามีความสมบูรณ์มากที่สุดอยู่ 2 ค่า คือ ที่ 0.90 และ 0.60 ในระดับความสว่างของรูปภาพแบบระดับเทาที่แตกต่างกันออกไป

ที่ค่าความสว่างของภาพในช่วง [21 - 60] Error ของภาพที่น้อยที่สุดมีค่า 3.0% ที่ค่าเทรสโฮลด์ 0.9

ที่ค่าความสว่างของภาพในช่วง [61 - 80] Error ของภาพที่น้อยที่สุดมีค่า 1.8% ที่ค่าเทรสโฮลด์ 0.9

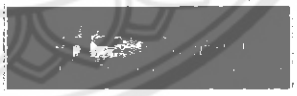
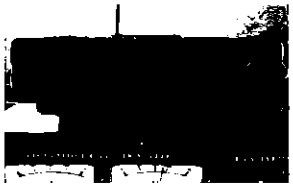
ที่ค่าความสว่างของภาพในช่วง [81 - 100] Error ของภาพที่น้อยที่สุดมีค่า 0.9% ที่ค่าเทรสโฮลด์ 0.9

ที่ค่าความสว่างของภาพในช่วง [101 - 120] Error ของภาพที่น้อยที่สุดมีค่า 3.1% ที่ค่าเทรสโฮลด์ 0.6

ที่ค่าความสว่างของภาพในช่วง [121 - 140] Error ของภาพที่น้อยที่สุดมีค่า 3.1% ที่ค่าเทรสโฮลด์ 0.6

4.4 ผลการทดลองของรูปที่ได้หลังการหาค่าเทรสโฮลด์ที่เหมาะสมแบบ RGB

ตารางที่ 4.4 แสดงผลลัพธ์ที่ได้หลังจากหาค่าเทรสโฮลด์ที่เหมาะสมแล้วในระบบสี RGB

Picture	G mean	ROI Original	Segmentation
	24.00		
	60.05		
	76.48		
	119.87		
	132.08		

จากตารางข้างต้นคือข้อมูลรูปภาพที่จะใช้ในการทดลอง G mean คือค่าเฉลี่ยระดับของเกรย์สเกลในรูปภาพชนิด RGB ที่จะใช้เป็นเกณฑ์ในการเลือกเทรสโฮลด์ ในคอลัมน์ที่ 3 เป็นรูปภาพต้นฉบับหลังจากที่หาพื้นที่ที่ให้ความสนใจออกมาได้แล้ว จนได้ออกมาเป็นภาพสุดท้ายในขั้นตอนการประมวลผลขั้นต้น ซึ่งเป็นขั้นตอนสุดท้ายก่อนที่จะนำไปตัดแล้วเข้าสู่กระบวนการการเข้าคู่รูปแบบ เหมือนกับในตารางของ HSV เช่นกัน

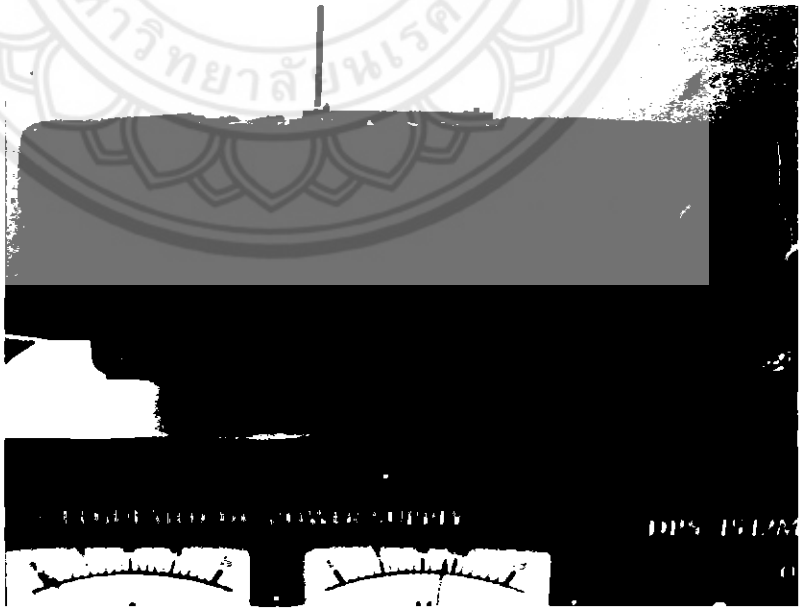
การทดลองต่อมาหลังจากทำการเลือกค่าเทรสโฮลด์ได้ค่าที่เหมาะสมแล้ว ขั้นตอนต่อไปจะเข้าสู่กระบวนการการเข้าคู่รูปแบบเพื่อแปลความหมายของรูปภาพที่ได้มาจากการทำกระบวนการก่อนหน้า โดยทำการตัดรูปภาพแยกออกจากกันทั้งหมด แล้วทำการไล่เปรียบเทียบด้วยกระบวนการเข้าคู่รูปแบบทีละรูปย่อยที่ตัดออกมาได้แล้วทำการแปลความหมายจนครบทุกรูปภาพ ซึ่งวิธีการเข้าคู่รูปแบบได้อธิบายไว้แล้วในบทก่อนหน้า

อีกหนึ่งเงื่อนไขที่ใช้ในการทดลองคือ การเข้าคู่รูปแบบโดยการลดขนาดของรูปภาพที่เป็นข้อมูลนำเข้า โดยขนาดที่ใช่ในการทดลองมี 4 ระดับ ได้แก่

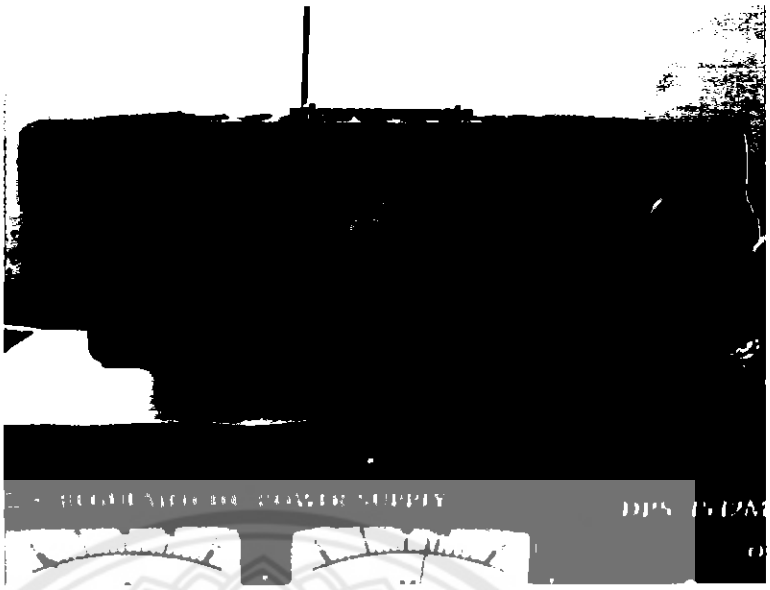
- ขนาดความละเอียดของหน้าจอภาพที่ระดับ 640 x 480
- ขนาดความละเอียดของหน้าจอภาพที่ระดับ 400 x 300
- ขนาดความละเอียดของหน้าจอภาพที่ระดับ 384 x 288
- ขนาดความละเอียดของหน้าจอภาพที่ระดับ 320 x 240

โดยขนาดของรูปภาพเท่าขนาดจริงที่ใช้ในการทดลอง แสดงไว้ดังรูปตัวอย่างด้านล่างต่อไปนี้

รูปที่ 4.3(ก) ขนาดความละเอียดของหน้าจอภาพที่ระดับ 640 x 480



รูปที่ 4.3(ข) ขนาดความละเอียดของหน้าจอภาพที่ระดับ 400 x 300



รูปที่ 4.3(ค) ขนาดความละเอียดของหน้าจอภาพที่ระดับ 384 x 288

รูปที่ 4.3(ง) ขนาดความละเอียดของหน้าจอภาพที่ระดับ 320 x 240

โดยทำการทดลองจากรูปภาพที่มีขนาดที่แตกต่างกันออกไปทั้งหมด 4 ขนาดและทำการทดลองภายใต้เงื่อนไขระดับความสว่างของแสงที่แตกต่างกันออกไปอีก 5 ระดับ บันทึกค่าที่ได้ทั้งจากระบบสีแบบ HSV และแบบ RGB ทั้งหมดลงในตาราง ดังที่ได้แสดงไว้ตามตารางข้างล่าง ต่อไปนี้

4.5 ผลการทดลองการหาค่าเกณฑ์ของการแปลความหมายในแต่ละตัวอักษร

ในกระบวนการการเข้าคู่รูปแบบขั้นตอนในการที่จะระบุว่าตัวอักษรที่กำลังดำเนินการอยู่นั้น โปรแกรมจะแปลความหมายได้ออกมาเป็นตัวอะไร ใช้หลักการการเปรียบเทียบเข้ามาช่วยในการตัดสินใจของโปรแกรม ทดลองโดยนำรูปภาพต้นแบบมาทำการลบกัน จากนั้นหาร้อยละของความผิดพลาดระดับพิกเซล เพื่อที่จะใช้เป็นเกณฑ์ในการตัดสินใจของโปรแกรมว่าจะแปลความหมายออกมาเป็นตัวอะไร โดยผลการทดลองบันทึกไว้ ดังตารางข้างล่าง ต่อไปนี้

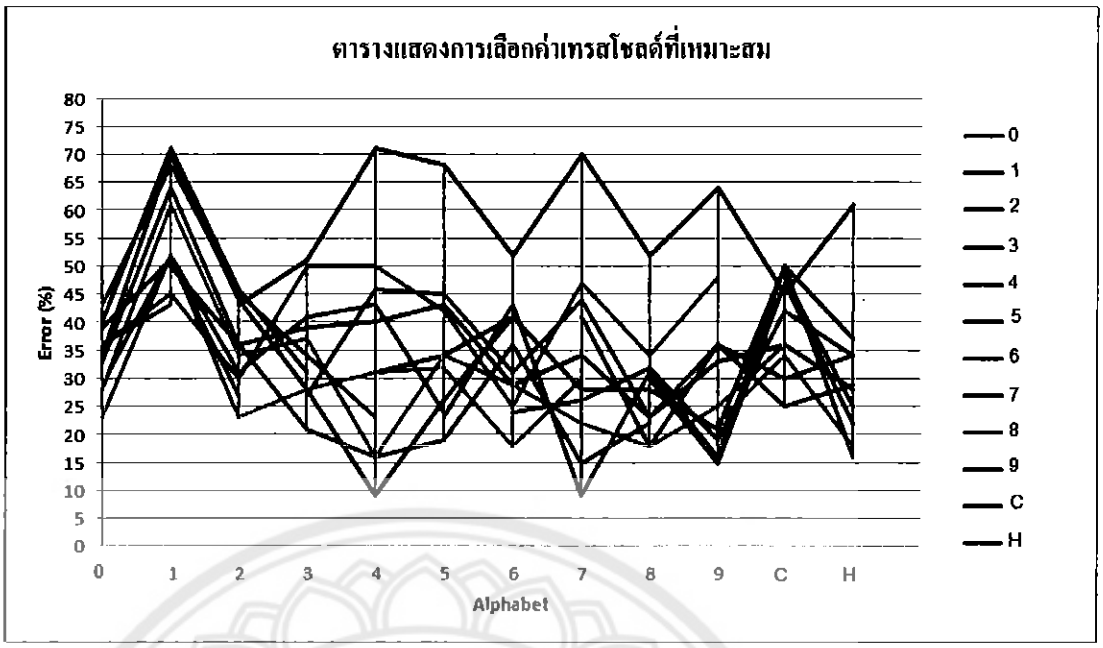
ตารางที่ 4.5 แสดงการเปรียบเทียบหาค่าความผิดพลาดระดับพิกเซลโดยวิธีการแยกวัตถุออกจากฉากหลัง (Background subtraction)ในอุดมคติ

	'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'G'	'H'
'0'	0	508	370	396	409	435	299	343	234	333	363	288
'1'	508	0	434	512	713	685	529	703	526	641	457	616
'2'	370	434	0	270	463	457	311	447	238	363	307	346
'3'	396	512	270	0	315	349	411	287	286	213	505	378
'4'	409	713	463	315	0	236	434	92	317	162	506	169
'5'	435	685	457	349	236	0	244	260	329	192	426	345
'6'	299	529	311	411	434	244	0	416	189	362	252	291
'7'	343	703	447	287	92	260	416	0	301	154	472	223
'8'	234	526	238	286	317	329	189	301	0	223	341	182
'G'	333	641	363	213	162	192	362	154	223	0	484	257
'H'	363	457	307	505	506	426	252	472	341	484	0	367
	288	616	346	378	169	345	291	223	182	257	367	0

ตารางที่ 6 แสดงการเปรียบเทียบหาร้อยละของค่าความผิดพลาดระดับพิกเซลโดยวิธีการแยกวัตถุออกจากฉากหลัง (Background subtraction)ในอุดมคติ

	'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'	'C'	'H'
'0'	0%	50%	36%	39%	40%	43%	29%	34%	23%	33%	36%	28%
'1'	50%	0%	43%	51%	71%	68%	52%	70%	52%	64%	45%	61%
'2'	36%	43%	0%	27%	46%	45%	31%	44%	23%	36%	30%	34%
'3'	39%	51%	27%	0%	31%	34%	41%	28%	28%	21%	50%	37%
'4'	40%	71%	46%	31%	0%	23%	43%	9%	31%	16%	50%	16%
'5'	43%	68%	45%	34%	23%	0%	24%	26%	32%	19%	42%	34%
'6'	29%	52%	31%	41%	43%	24%	0%	41%	18%	36%	25%	29%
'7'	34%	70%	44%	28%	9%	26%	41%	0%	30%	15%	47%	22%
'8'	23%	52%	23%	28%	31%	32%	18%	30%	0%	22%	34%	18%
'9'	33%	64%	36%	21%	16%	19%	36%	15%	22%	0%	48%	25%
'C'	36%	45%	30%	50%	50%	42%	25%	47%	34%	48%	0%	36%
'H'	28%	61%	34%	37%	16%	34%	29%	22%	18%	25%	36%	0%

ผลจากตารางที่ได้ เมื่อนำมาเปรียบเทียบเป็นกราฟเพื่อใช้เป็นเกณฑ์ในการตัดว่าควรจะใช้ค่าเทรสโฮลด์ที่เท่าไรจึงจะมีความเหมาะสมมากที่สุด ดังกราฟที่แสดงไว้ ดังต่อไปนี้



รูปที่ 4.4 แสดงการหาค่าที่เหมาะสมในการทำเทรสโฮลด์ของการแปลความตัวอักษร

จากตารางค่าที่น้อยที่สุดของการลบกันระหว่างตัวอักษรที่แตกต่างกันออกไปนั้น อยู่ที่ 9% โดยเป็นค่าของการลบกันระหว่างเลข 4 กับเลข 7 แต่เป็นเพียงค่าส่วนน้อยเท่านั้นจึงไม่เหมาะสมในการนำค่านี้ไปเป็นเงื่อนไขในการคำเทรสโฮลด์ จึงดูค่าน้อยสุดในลำดับต่อไป โดยจะเห็นว่าจะไม่มีค่าไหนเลยที่น้อยกว่า 15% ลงมา ทำให้ได้ข้อสรุปที่ว่าค่า 15% เป็นค่าที่เหมาะมสในการมาเทรสโฮลด์ในการแปลความหมายของตัวอักษร

4.5 ผลการทดลองการแปลความหมายจากรูปภาพของโปรแกรม

ตารางที่ 4.7 แสดงการแปลความหมายตัวอักษร ที่ค่าความสว่างเฉลี่ย 0.122 ของ HSV และค่าระดับเทาเฉลี่ยที่ 24.0 ของ RGB

Picture	$\overline{V}$	HSV				$\overline{G}$	RGB				∑ Error (%)	
		Interpreted & Error (%)					Interpreted & Error (%)				HSV	RGB
640x480	0.122	'C'	'H'	'6'	'7'	24.0	'C'	'H'	'6'	'7'		

		10.8	6.1	8.0	5.3		8.0	8.3	5.3	7.7	30.2	29.3
400x300	0.122	'C'	'H'	'6'	'7'	24.0	'C'	'H'	'6'	'7'		
		11.7	11.9	12.9	7.5		11.4	9.8	14.0	9.6	44.0	44.8
384x288	0.122	'C'	'H'	'6'	'7'	24.0	'C'	'H'	'6'	'7'		
		12.7	8.7	10.7	10.9		11.4	9.8	9.3	9.6	43.0	40.1
320x240	0.122	'C'	'H'	'-'	'7'	24.0	'C'	'H'	'6'	'7'		
		13.1	9.5	15.4	14.3		10.5	9.6	10.4	8.7	52.3	39.2

ผลการทดลองการเปรียบเทียบค่าโมเด็ลสีระหว่าง RGB และ HSV ภายใต้ค่าความสว่างเฉลี่ยของแสงที่ 0.122 ในระบบสี HSV และที่ค่าระดับเทาเฉลี่ยที่ 24.0 ผลที่ได้ออกมาโดยดูจากความผิดพลาดระดับพิกเซลรวมที่ค่ายิ่งน้อยยิ่งดี สรุปได้ว่าระบบสีแบบ RGB มีค่าความผิดพลาดระดับพิกเซลรวมที่น้อยกว่าระบบสีแบบ HSV และระบบสี RGB ให้ผลการทดลองที่ดีกว่าระบบสีแบบ HSV ในเกือบทุกๆกรณีของขนาดความละเอียดของหน้าจอ มีแค่บางกรณีที่ระบบสี RGB จะด้อยกว่าแบบของ HSV แต่ค่าความแตกต่างของผลรวมของความผิดพลาดระดับพิกเซลในกรณีที่ HSV ทำได้ดีกว่า RGB นั้นก็แทบจะไม่แตกต่างหรือแตกต่างกันน้อยมาก และในกรณีที่ภาพมีความละเอียดของหน้าจอเล็กๆ HSV จะเกิดการแปลความหมายที่ผิดพลาดไปซึ่งต่างจากระบบสีของ RGB ที่ยังคงแปลงความหมายได้ถูกต้องในทุกๆขนาดของภาพ

ตารางที่ 4.8 แสดงการแปลความหมายตัวอักษร ที่ค่าความสว่างเฉลี่ย 0.286 ของ HSV และค่าระดับเทาเฉลี่ยที่ 60.0 ของ RGB

Picture	$\overline{V}$	HSV	$\overline{G}$	RGB	∑ Error (%)	
		Interpreted & Error (%)		Interpreted & Error (%)	HSV	RGB

640x480	0.286	'C'	'H'	'6'	'7'	60.0	'C'	'H'	'6'	'7'		
		6.2	6.4	10.1	7.7		7.1	6.2	6.2	7.5	30.4	26.9
400x300	0.286	'C'	'H'	'6'	'7'	60.0	'C'	'H'	'6'	'7'		
		9.8	12.0	11.8	8.3		14.5	10.3	9.5	11.2	41.9	45.5
384x288	0.286	' - '	'H'	' - '	'7'	60.0	'C'	'H'	'6'	'7'		
		15.0	10.3	15.8	12.6		14.5	10.3	9.6	11.2	53.7	45.6
320x240	0.286	' - '	'H '	'6'	' - '	60.0	'C'	'H'	'6'	'7'		
		20.4	8.4	13.0	16.7		11.6	8.2	9.8	12.1	58.5	41.7

ผลการทดลองการเปรียบเทียบค่าโมเด็ลสีระหว่าง RGB และ HSV ภายใต้ค่าความสว่างเฉลี่ยของแสงที่ 0.286 ในระบบสี HSV และที่ค่าระดับเทาเฉลี่ยที่ 60.0 ผลที่ได้ออกมาโดยดูจากความผิดพลาดระดับพิกเซลรวมที่ค่ายิ่งน้อยยิ่งดี สรุปได้ว่าระบบสีแบบ RGB มีค่าความผิดพลาดระดับพิกเซลรวมที่น้อยกว่าระบบสีแบบ HSV และระบบสี RGB ให้ผลการทดลองที่ดีกว่าระบบสีแบบ HSV ในเกือบทุกๆกรณีของขนาดความละเอียดของหน้าจอ มีบางกรณีที่ระบบสี RGB จะด้อยกว่าแบบของ HSV แต่ค่าความแตกต่างของผลรวมของความผิดพลาดระดับพิกเซลในกรณีที่ HSV ทำได้ดีกว่า RGB นั้นแทบจะไม่แตกต่างหรือแตกต่างกันน้อยมาก สุดท้ายในกรณีที่ภาพมีความละเอียดของหน้าจอเล็กๆระบบสีแบบ HSV จะมีการแปลความหมายที่ผิดพลาดออกไป ซึ่งต่างจากระบบสีของ RGB ที่ยังคงแปลงความหมายได้ถูกต้องในทุกๆขนาดของภาพ

ตารางที่ 4.9 แสดงการแปลความหมายตัวอักษร ที่ค่าความสว่างเฉลี่ย 0.361 ของ HSV และค่าระดับเทาเฉลี่ยที่ 76.4 ของ RGB

Picture	$\overline{V}$	HSV	$\overline{G}$	RGB	∑ Error (%)	
		Interpreted & Error (%)		Interpreted & Error (%)	HSV	RGB

640x480	0.361	'C'	'H'	'6'	'7'	76.4	'C'	'H'	'6'	'7'		
		6.2	6.0	8.5	5.2		6.4	7.1	6.4	4.9	25.9	24.8
400x300	0.361	'C'	'H'	'6'	'7'	76.4	'C'	'H'	'6'	'7'		
		8.9	13.2	11.2	7.1		11.5	13.0	11.0	8.8	40.4	44.3
384x288	0.361	'C'	'H'	' - '	'7'	76.4	'C'	'H'	'6'	'7'		
		12.6	10.0	16.4	11.3		12.5	10.9	10.1	10.9	50.3	44.4
320x240	0.361	'C'	'H '	'6'	' - '	76.4	'C'	'H'	'6'	'7'		
		14.0	8.8	14.8	17.4		11.0	9.5	10.7	12.4	55.0	43.6

ผลการทดลองการเปรียบเทียบค่าโมเด็ลสีระหว่าง RGB และ HSV ภายใต้ค่าความสว่างเฉลี่ยของแสงที่ 0.361 ในระบบสี HSV และที่ค่าระดับเทาเฉลี่ยที่ 76.4 ผลที่ได้ออกมาโดยดูจากความผิดพลาดระดับพิกเซลรวมที่ค่ายิ่งน้อยยิ่งดี สรุปได้ว่าระบบสีแบบ RGB มีค่าความผิดพลาดระดับพิกเซลรวมที่น้อยกว่าระบบสีแบบ HSV และระบบสี RGB ให้ผลการทดลองที่ดีกว่าระบบสีแบบ HSV ในเกือบทุกๆกรณีของขนาดความละเอียดของหน้าจอ มีบางกรณีที่ระบบสี RGB จะด้อยกว่าแบบของ HSV แต่ค่าความแตกต่างของผลรวมของความผิดพลาดระดับพิกเซลในกรณีที่ HSV ทำได้ดีกว่า RGB นั้นแทบจะไม่แตกต่างหรือแตกต่างกันน้อยมาก สุดท้ายในกรณีที่ภาพมีความละเอียดของหน้าจอเล็กๆระบบสีแบบ HSV จะมีการแปลความหมายที่ผิดพลาดออกไป ซึ่งต่างจากระบบสีของ RGB ที่ยังคงแปลงความหมายได้ถูกต้องในทุกๆขนาดของภาพ

ตารางที่ 4.10 แสดงการแปลความหมายตัวอักษร ที่ค่าความสว่างเฉลี่ย 0.574 ของ HSV และค่าระดับเทาเฉลี่ยที่ 119.8 ของ RGB

Picture	$\overline{V}$	HSV	$\overline{G}$	RGB	∑ Error (%)	
		Interpreted & Error (%)		Interpreted & Error (%)	HSV	RGB

640x480	0.574	'C'	'H'	'6'	'7'	119.8	'C'	'H'	'6'	'7'	
		9.6	11.2	9.1	11.8		7.0	5.5	8.8	5.3	41.7	26.6
400x300	0.574	'C'	'H'	'6'	'7'	119.8	'C'	'H'	'6'	'7'		
		8.4	12.1	9.1	7.9		7.7	9.9	8.5	11.0	37.5	37.1
384x288	0.574	'C'	'H'	'6'	'7'	119.8	'C'	'H'	'4'	'7'		
		14.3	11.7	7.0	11.4		12.1	13.5	16.7	11.7	44.4	54.0
320x240	0.574	'C'	'H '	'6'	' - '	119.8	'C'	'H'	'6'	'7'		
		14.3	8.9	12.1	21.0		11.5	13.0	11.9	8.8	56.3	45.2

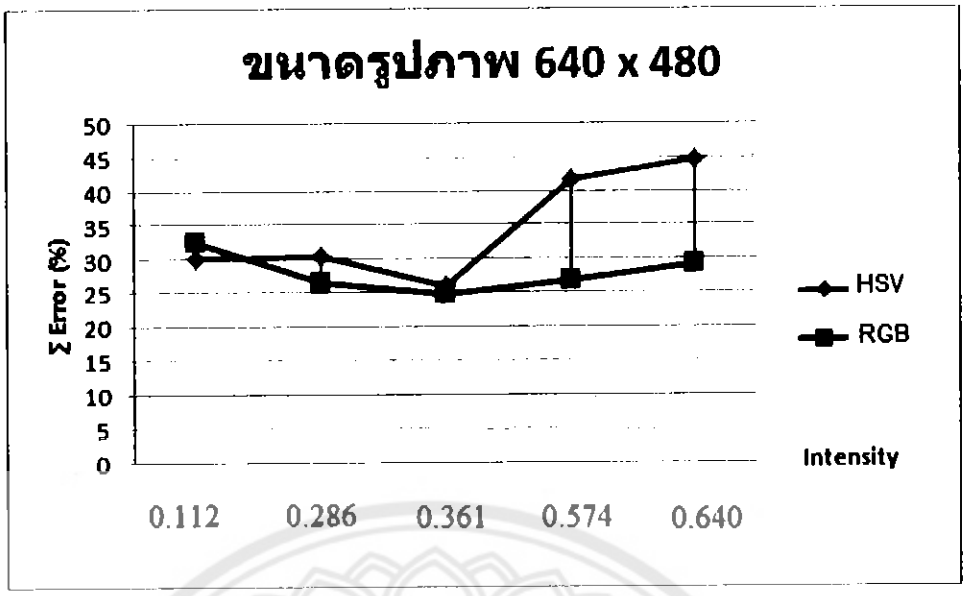
ผลการทดลองการเปรียบเทียบค่าโมเด็ลสีระหว่าง RGB และ HSV ภายใต้ค่าความสว่างเฉลี่ยของแสงที่ 0.574 ในระบบสี HSV และที่ค่าระดับเทาเฉลี่ยที่ 119.8 ผลที่ได้ออกมาโดยดูจากความผิดพลาดระดับพิกเซลรวมที่ค่ายิ่งน้อยยิ่งดี สรุปได้ว่าระบบสีแบบ RGB มีค่าความผิดพลาดระดับพิกเซลรวมที่น้อยกว่าระบบสีแบบ HSV และระบบสี RGB ให้ผลการทดลองที่ดีกว่าระบบสีแบบ HSV ในเกือบทุกๆกรณีของขนาดความละเอียดของหน้าจอ มีบางกรณีที่ระบบสี RGB จะด้อยกว่าแบบของ HSV แต่ค่าความแตกต่างของผลรวมของความผิดพลาดระดับพิกเซลในกรณีที่ HSV ทำได้ดีกว่า RGB นั้นแทบจะไม่แตกต่างหรือแตกต่างกันน้อยมาก สุดท้ายในกรณีที่ภาพมีความละเอียดของหน้าจอเล็กๆระบบสีแบบ HSV จะมีการแปลความหมายที่ผิดพลาดออกไป ซึ่งต่างจากระบบสีของ RGB ที่ยังคงแปลงความหมายได้ถูกต้องในทุกๆขนาดของภาพ

ตารางที่ 4.11 แสดงการแปลความหมายตัวอักษร ที่ค่าความสว่างเฉลี่ย 0.640 ของ HSV และค่าระดับเทาเฉลี่ยที่ 132.0 ของ RGB

Picture	$\overline{V}$	HSV	$\overline{G}$	RGB	∑ Error (%)	
		Interpreted & Error (%)		Interpreted & Error (%)	HSV	RGB

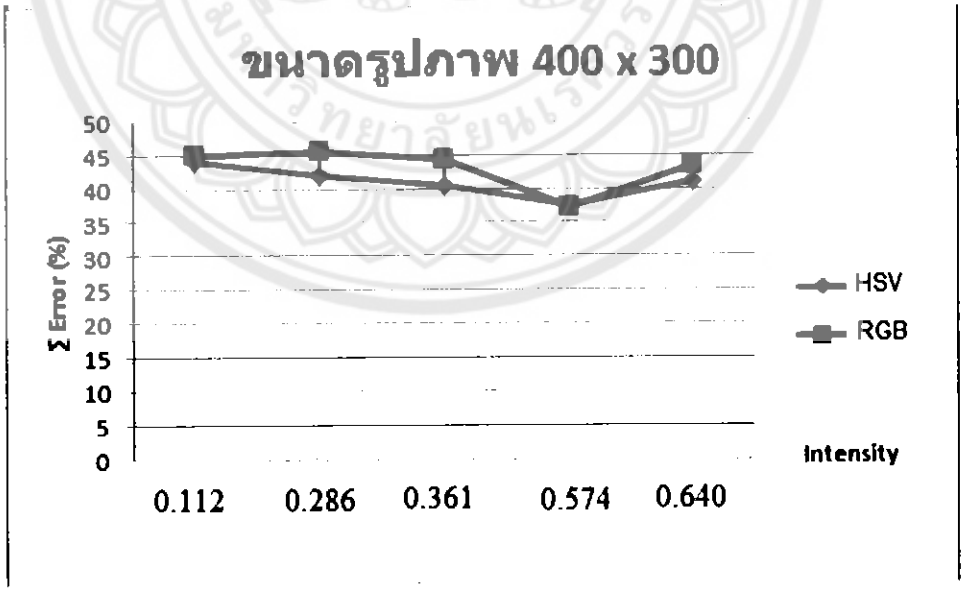
640x480	0.640	'C'	'H'	'6'	'7'	132.0	'C'	'H'	'6'	'7'		
		11.6	10.4	13.2	9.4		8.5	6.4	6.6	11.0	44.6	32.5
400x300	0.640	'C'	'H'	'6'	'7'	132.0	'C'	'H'	'6'	'7'		
		8.6	14.0	7.9	10.7		12.5	10.3	7.1	13.5	41.2	43.4
384x288	0.640	' - '	'H'	'6'	'7'	132.0	'C'	'H'	'6'	'7'		
		18.8	12.3	7.9	13.0		12.2	9.8	9.5	14.0	52.0	45.5
320x240	0.640	'C'	'H'	'6'	' - '	132.0	'C'	'H'	'6'	'7'		
		14.5	8.4	13.8	19.6		12.1	12.0	12.3	11.8	56.3	48.2

ผลการทดลองการเปรียบเทียบค่าโมเด็ลสีระหว่าง RGB และ HSV ภายใต้ค่าความสว่างเฉลี่ยของแสงที่ 0.640 ในระบบสี HSV และที่ค่าระดับเทาเฉลี่ยที่ 132.0 ผลที่ได้ออกมาโดยดูจากความผิดพลาดระดับพิกเซลรวมที่ค่ายิ่งน้อยยิ่งดี สรุปได้ว่าระบบสีแบบ RGB มีค่าความผิดพลาดระดับพิกเซลรวมที่น้อยกว่าระบบสีแบบ HSV และระบบสี RGB ให้ผลการทดลองที่ดีกว่าระบบสีแบบ HSV ในเกือบทุกๆกรณีของขนาดความละเอียดของหน้าจอ มีบางกรณีที่ระบบสี RGB จะด้อยกว่าแบบของ HSV แต่ค่าความแตกต่างของผลรวมของความผิดพลาดระดับพิกเซลในกรณีที่ HSV ทำได้ดีกว่า RGB นั้นแทบจะไม่แตกต่างหรือแตกต่างกันน้อยมาก สุดท้ายในกรณีที่ภาพมีความละเอียดของหน้าจอเล็กๆระบบสีแบบ HSV จะมีการแปลความหมายที่ผิดพลาดออกไป ซึ่งต่างจากระบบสีของ RGB ที่ยังคงแปลงความหมายได้ถูกต้องในทุกๆขนาดของภาพ โดยสามารถสรุปผลการทดลองที่ได้เป็นกราฟดังรูปต่อไปนี้



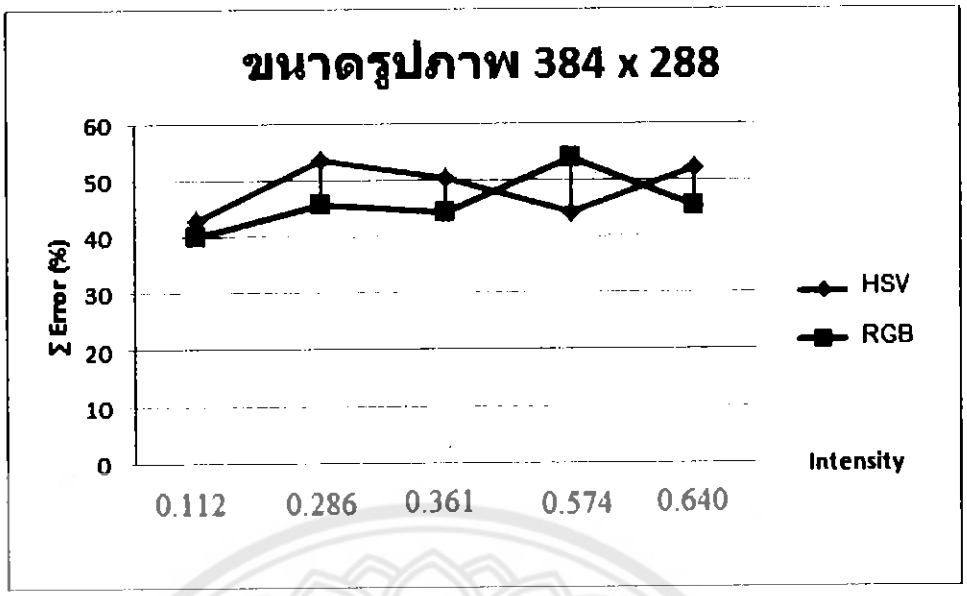
รูปที่ 4.5 แสดงผลรวมร้อยละของความผิดพลาดของตัวอักษรที่ขนาดรูปภาพ 640 x 480

จากผลการทดลองที่รูปภาพขนาด 640 x 480 ทั้งสองระบบสีสามารถแปลความหมายออกมาได้อย่างถูกต้องและถ้ามองในด้านร้อยละของความผิดพลาดระดับพิกเซลระบบสีแบบ RGB ให้ผลการทดลองออกมาที่ดีกว่า โดยมีร้อยละของความผิดพลาดที่น้อยกว่าระบบสีแบบ HSV อยู่มาก



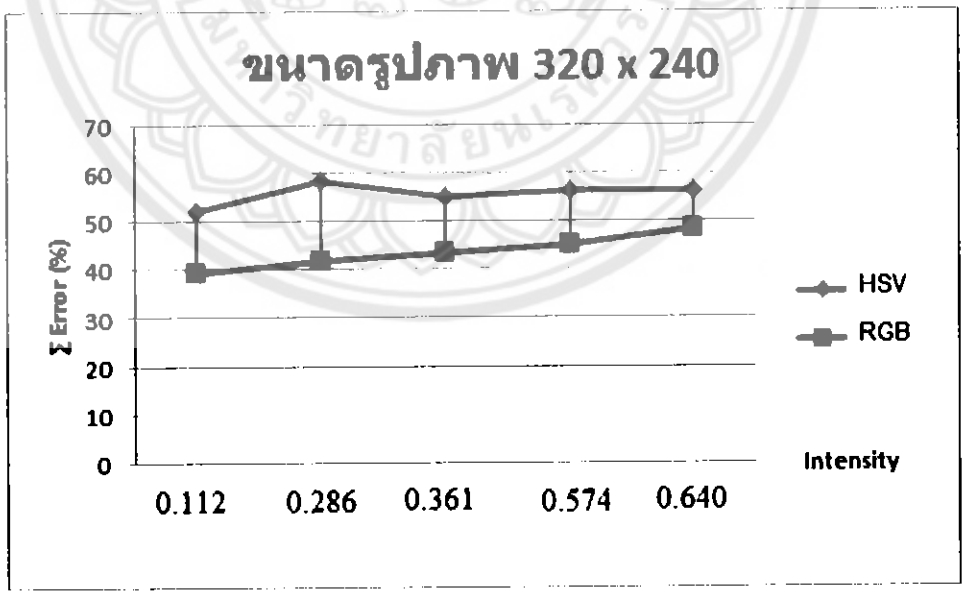
รูปที่ 4.6 แสดงผลรวมร้อยละของความผิดพลาดของตัวอักษรที่ขนาดรูปภาพ 400 x 300

ทั้งสองระบบสียังแปลความหมายออกมาได้อย่างถูกต้องและแต่ในด้านร้อยละของความผิดพลาดระดับพิกเซลระบบสีแบบ HSV ในระดับนี้ทำได้ดีกว่าแบบ RGB เล็กน้อย



รูปที่ 4.7 แสดงผลรวมร้อยละของความผิดพลาดของตัวอักษรที่ขนาดรูปภาพ 384 x 288

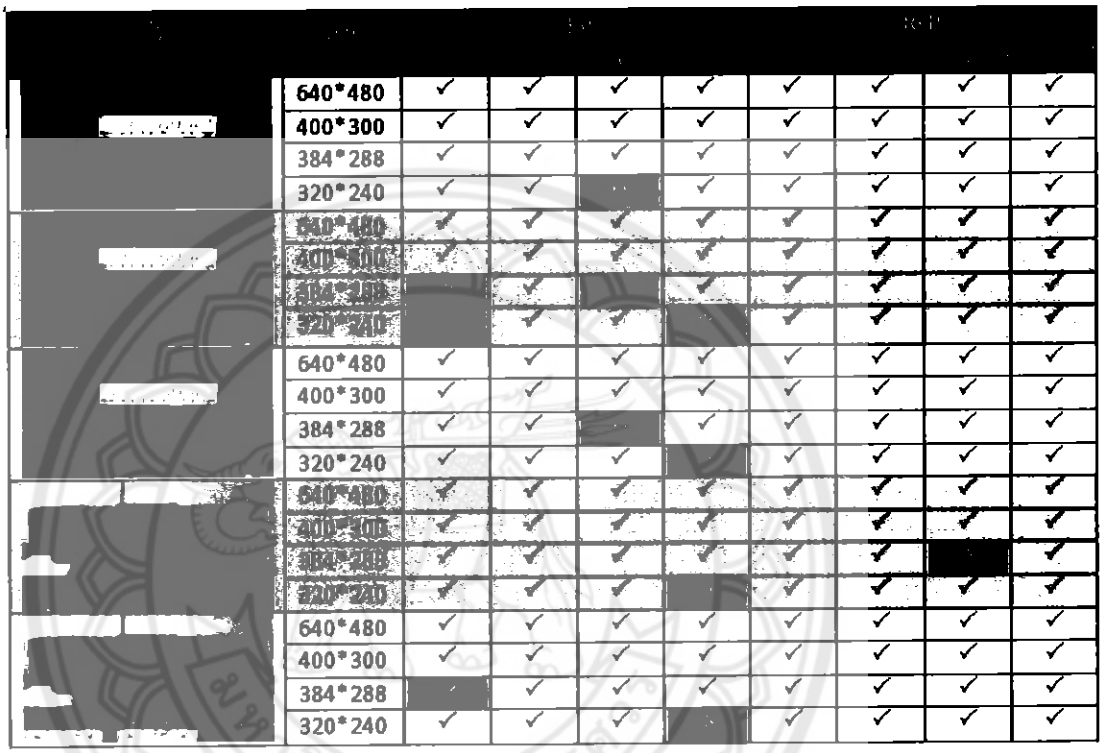
ระบบสีแบบ HSV เริ่มแปลความหมายผิดพลาดโดยที่ระบบสีแบบ RGB ยังคงแปลได้ถูกต้องอยู่ และถ้ามองในด้านร้อยละของความผิดพลาดระดับพิกเซลระบบสีแบบ RGB ให้ผลการทดลองออกมาที่ดีกว่า โดยยังคงมีร้อยละของความผิดพลาดที่น้อยกว่าระบบสีแบบ HSV



รูปที่ 4.8 แสดงผลรวมร้อยละของความผิดพลาดของตัวอักษรที่ขนาดรูปภาพ 320 x 240

ระบบสีแบบ HSV ยังแปลความหมายผิดพลาด โดยที่ระบบสีแบบ RGB ยังคงแปลได้ถูกต้องอยู่ และและร้อยละของความผิดพลาดระดับพิกเซลระบบสีแบบ RGB มีร้อยละของความผิดพลาดที่น้อยกว่าระบบสีแบบ HSV อยู่มาก

ตารางที่ 4.12 ตารางสรุปผลความถูกต้องในการแปลความหมายจากกระบวนการรู้จำอักขระ

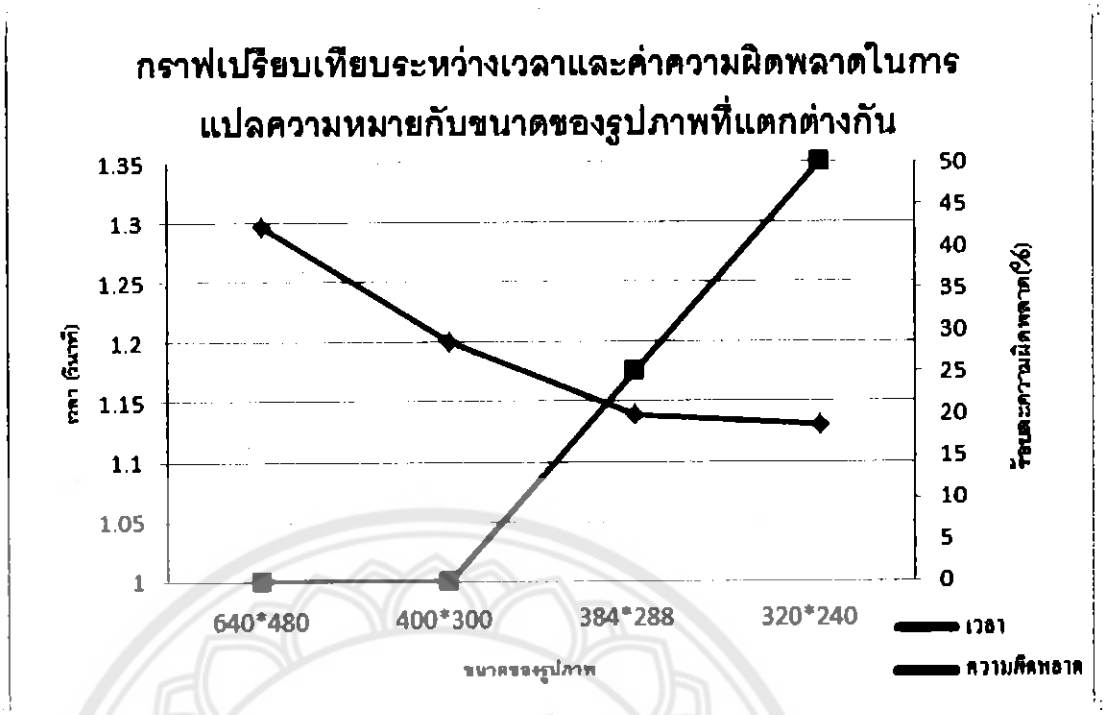


	640*480	✓	✓	✓	✓	✓	✓	✓	✓
	400*300	✓	✓	✓	✓	✓	✓	✓	✓
	384*288	✓	✓	✓	✓	✓	✓	✓	✓
	320*240	✓	✓		✓	✓	✓	✓	✓
	640*480	✓	✓	✓	✓	✓	✓	✓	✓
	400*300	✓	✓	✓	✓	✓	✓	✓	✓
	384*288		✓		✓	✓	✓	✓	✓
	320*240		✓	✓		✓	✓	✓	✓
	640*480	✓	✓	✓	✓	✓	✓	✓	✓
	400*300	✓	✓	✓	✓	✓	✓	✓	✓
	384*288	✓	✓		✓	✓	✓	✓	✓
	320*240	✓	✓	✓		✓	✓	✓	✓
	640*480	✓	✓	✓	✓	✓	✓	✓	✓
	400*300	✓	✓	✓	✓	✓	✓	✓	✓
	384*288	✓	✓	✓	✓	✓	✓		✓
	320*240	✓	✓	✓		✓	✓	✓	✓
	640*480	✓	✓	✓	✓	✓	✓	✓	✓
	400*300	✓	✓	✓	✓	✓	✓	✓	✓
	384*288		✓	✓	✓	✓	✓	✓	✓
	320*240	✓	✓	✓		✓	✓	✓	✓

ตารางที่ 4.13 ตารางแสดงเวลาที่ใช้ต่อขนาดความละเอียดของภาพ

ขนาดรูปภาพ	ความถูกต้อง *	เวลาที่ใช้ (วินาที)
640*480	✓	1.297483
400*300	✓	1.200304
384*288	✓	1.139001
320*240	✓	1.130045

*ความถูกต้องของการแปลความหมายจากรูปภาพจากตารางข้างต้น ค่าเทรสโฮลด์บางค่าจะต้องเปลี่ยนค่าไปตามขนาดของรูปภาพที่เปลี่ยนไปด้วยเพื่อให้ได้ความถูกต้องในการแปลความหมายมากที่สุด



รูปที่ 4.9 กราฟแสดงความสัมพันธ์ระหว่างเวลาและค่าความผิดพลาดในการแปลความหมายกับขนาดของความละเอียดของรูปภาพที่แตกต่างกันออกไป

จากรูปตัวอย่างข้างต้น ยกตัวอย่างผลการทดลองที่ได้จากรูปภาพที่มีความเข้มแสงเฉลี่ยที่ 0.286 ซึ่งภาพจะมีลักษณะค่อนข้างมืด ผลการแปลความหมายตัวอักษร 'C', 'H', '6' และ '7' มองในด้านเวลาที่ใช้ที่กระบวนการเดียวกันขนาดของภาพที่ใช้จะแปรผันตรงกับเวลา คือยิ่งรูปภาพที่ใช้ขนาดใหญ่เวลาที่ใช้ทั้งหมดก็จะมากขึ้นด้วยแต่ก็ยังสรุปไม่ได้ว่าภาพใหญ่หรือภาพเล็กดีกว่ากัน เพราะถ้ามองในอีกด้าน คือในด้านความถูกต้องของกาแปรความหมายซึ่งขนาดของรูปภาพจะแปรผันแบบผกผันกับความผิดพลาด กล่าวคือยิ่งภาพมีขนาดใหญ่การแปลความหมายของคำที่ได้จะมีค่าเป็นศูนย์หรือไม่มีความผิดพลาดเกิดขึ้นเลยก็ว่าได้ ต่างจากขนาดของรูปภาพขนาดเล็กที่การแปลความหมายจะเริ่มผิดพลาดมากขึ้นเรื่อยๆตามขนาดของภาพที่เล็กลง

บทที่ 5

บทสรุป

5.1 วิเคราะห์และสรุปผลการทดลอง

เนื่องจากโครงงานนี้ ใช้หลักการของการประมวลผลภาพดิจิตอลเป็นหลัก ทำให้การใช้งานโปรแกรมค่อนข้างมีข้อจำกัด และปัจจัยที่ส่งผลกระทบต่อการประมวลผลภาพมีอยู่หลากอย่างเช่น สภาพแวดล้อมภายนอก คุณภาพของเอกสารที่ต้องการแปล คุณภาพของกล้องเว็บแคม และการจับภาพ เป็นต้น เมื่อปัจจัยเหล่านี้ส่งผลกระทบจะทำให้การระบุตัวอักษรมีความถูกต้องน้อยลง จะส่งผลต่อกระบวนการรู้จำอักขระจะผิดพลาดไปด้วย

ในการทดลองจะทำการเปรียบเทียบประสิทธิภาพระหว่าง 2 ระบบสี โดยอาศัยหน้าจอมอนิเตอร์มาเป็นตัวชี้วัด โดยทำการทดลองภายใต้ค่าความสว่างของแสงที่แตกต่างกันออกไป เพื่อทดสอบว่าหลังจากเสร็จสิ้นกระบวนการแล้ว ระบบสีไหนมีความผิดพลาดในการแปลความน้อยที่สุด โดยผลที่ได้ออกมานั้นสามารถสรุปได้ว่า ระบบสีแบบ RGB ให้ผลการทดลองที่ถูกต้องมากว่าระบบสีแบบ HSV แม้กระทั้งการทดสอบในภาพที่มีขนาดที่เล็กลง ระบบสี RGB ยังคงแปลความหมายออกมาได้อย่างถูกต้องต่างจากระบบสีแบบ HSV ที่ขนาดของภาพมีขนาดเล็กลงการแปลความหมายจากภาพนั้นก็จะเริ่มผิดเพี้ยนไป ปัจจัยที่ส่งผลกระทบมากที่สุดในการประมวลผลภาพ คือ ความคมชัดของตัวอักษร ซึ่งสาเหตุส่วนใหญ่ที่ทำให้การประมวลผลภาพมีความผิดพลาดเกิดจากความละเอียดของกล้องเว็บแคมน้อย ไม่สามารถปรับโฟกัสได้ ทำให้ตัวอักษรที่ทำการจับภาพไม่สามารถระบุได้ว่าตัวอักษรนั้นคือตัวใด และอีกหนึ่งปัจจัยที่ส่งผลกระทบคือ มุมเอียงต่างๆ ซึ่งสาเหตุอาจเกิดจากขั้นตอนการรับภาพผู้ใช้อาจจับเอกสารไม่ตรง ทำให้เกิดมุมเอียงซึ่งจะส่งผลต่อการระบุตัวอักษร ปัจจัยสุดท้ายที่ส่งผลกระทบคือ ตัวอักษรที่มีความคล้ายคลึงกันของตัวอักษร โดยที่ตัวอักษรอักษรที่มักเกิดข้อผิดพลาดคือตัวเลข 4, 7 และ 9 ซึ่งตัวเลขมีลักษณะใกล้เคียงกัน แต่ปัญหานี้สามารถแก้ได้จากภายในโปรแกรมของเรา

5.2 ปัญหาและแนวทางแก้ไข

จากการทดลองเพื่อหาปัจจัยที่ส่งผลกระทบต่อการทำงานของโปรแกรม ทำให้ทราบถึงสาเหตุต่างๆ ที่ทำให้เกิดข้อผิดพลาดขึ้น โดยส่งผลให้ไม่สามารถทำการแปลความหมายจากหน้าจอมอนิเตอร์ได้อย่างสมบูรณ์ ซึ่งพบปัญหาอุปสรรคในการใช้งานและแนวทางแก้ไขปัญหาดังนี้

ปัญหาและอุปสรรค	แนวทางการแก้ไข
1. ระยะห่างของการวางกล้อง	1. ระบุระยะห่างการวางกล้องให้ชัดเจน
2.ขนาดความละเอียดของภาพยังไม่ยืดหยุ่น	2. ปรับปรุงโปรแกรมให้ลองรับขนาดของภาพได้หลายๆขนาด
3.การทำการเทรสโฮลด์รูปภาพแล้วตัวอักษรที่ได้ออกมายังไม่มีความสมบูรณ์ 100%	3. เพิ่มการประมวลผลขั้นต้นและขั้นตอนการรู้จำให้มีประสิทธิภาพมากยิ่งขึ้น
4. กรณีถ่ายภาพมุมเอียง	4. เพิ่มการปรับภาพมุมเอียงกลับเป็นปกติ

ดังนั้นในทางปฏิบัติถ้าสามารถขจัดปัญหาและอุปสรรคดังกล่าวได้ก็จะสามารถเพิ่มประสิทธิภาพการทำงานของโปรแกรมได้อย่างสมบูรณ์มากยิ่งขึ้น

5.3 แนวทางการพัฒนาในอนาคต

1. พัฒนาโปรแกรมให้สามารถแปลความหมายในบริเวณส่วนอื่นของหน้าจอมอนิเตอร์ได้

2.ปรับปรุงให้สามารถใช้ได้จากทุกระยะห่างของการตั้งกล้อง

3. ปรับปรุงให้สามารถแปลความหมายจากภาพที่รับเข้ามาแบบมุมเอียงได้

4.ปรับปรุงให้สามารถแปลความหมายได้ถูกต้องมากยิ่งขึ้นในการประมวลผลแบบเวลาจริง

เอกสารอ้างอิง

[1] Gonzalez, R.C. and Richard E. Woods. (2001). Digital Image Processing. New Jersey : Prentice-Hall. Inc.

[2] Baxes, Gregory A. (1994). Digital Image Processing. Canada : John Wiley & Sons. Inc.

[3] "Digital Image Fundamental." [online].
Available:http://www.ecpe.nu.ac.th/panomkhawn/imagepro/pdf/ch09.pdf. 2554.

[4] "Image Segmentation." [online].
Available:Http://fivedots.coe.psu.ac.th/~montri/Teaching/image/segment.DOC. 2554.

[5] "Morphological Image Processing." [online].
Available:Http://www.ecpe.nu.ac.th/panomkhawn/imagepro/pdf/ch09.pdf. 2554.

[6] "Thresholding." [online]. Available:Http://www.wbi.msu.ac.th/file/648/doc_45.ppt. 2554.

[7] "Morphological Image Processing." [online].
Available:Http://fivedots.coe.psu.ac.th/~montri/Teaching/image/morph.DOC. 2554.

[8] Settha Tangkawanit and Surachet Kanprachar. (2007). Performance of RGB and HSV Color Systems in Object Detection Applications under Different Illumination Intensities. In the International MultiConference (ICIE 2007) March 21-23, 2007, Hong Kong

ภาคผนวก ก.

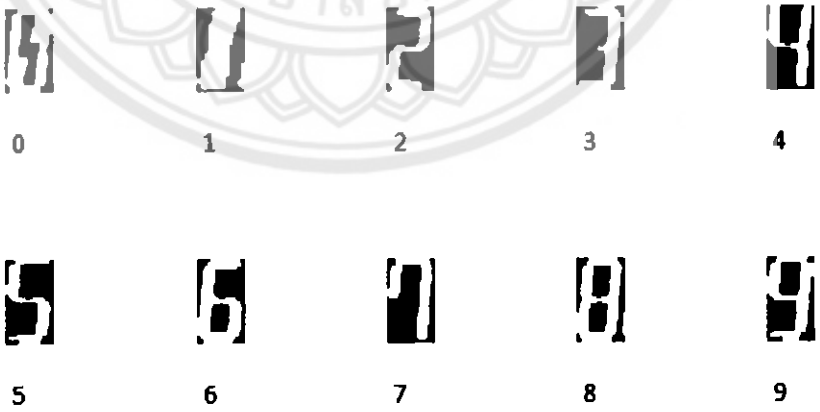
ภาพรูปแบบ (Templates) ในโปรแกรม

การแสดงภาพรูปแบบที่เก็บไว้ภายในโปรแกรม

1. ภาพรูปตัวอักษรใหญ่ตัว C และตัว H

รูปที่ ก-1 แสดงตัวอักษรรูปแบบ C และ H

2. ภาพรูปแบบตัวเลข ตั้งแต่เลข 0 ไปจนถึง 9



รูปที่ ก-2 แสดงตัวเลขรูปแบบ 0 ไปจนถึง 9