การรู้จำตัวอักษรภาษาไทยโดยการใช้โครงข่ายประสาทเทียม

Optical Thai Character Recognition

จัดทำโดย

นาย จิตเกษม ปินทะยา รหัสประจำตัว 46010109

นาย อัครินทร์ ล้วนจำเริญ รหัสประจำตัว 46010947

อาจารย์ที่ปรึกษา

ผศ.เกียรติกูล เจียรนัยธนะกิจ

รายงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาวิชา 01072129 โครงงานคอมพิวเตอร์

หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

ปริญญานิพนธ์ปีการศึกษา 2549

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การรู้จำตัวอักษรภาษาไทยโดยการใช้โครงข่ายประสาทเทียม

Optical Thai Character Recognition

ผู้จัดทำ 1. นายจิตเกษม ปินทะยา รหัส 46010109

2. นายอัครินทร์ ล้วนจำเริญ รหัส 46010947

อาจารย์ที่ปรึกษา

(ผศ.เกียรติกูล เจียรนัยธนะกิจ)

การรู้จำตัวอักษรภาษาไทยโดยการใช้โครงข่ายประสาทเทียม

นาย จิตเกษม ปินทะยา 46010109
นาย อัครินทร์ ล้วนจำเริญ 46010947
ผศ.เกียรติกูล เจียรนัยธนะกิจ อาจารย์ที่ปรึกษา
ปีการศึกษา 2006

บทคัดย่อ

ในปัจจุบันนี้ เทคโนโลยีทางด้านการประมวลผลภาพ ได้เข้ามามีบทบาทกับชีวิตประจำวันของเรา มาก ขึ้น ดังจะเห็นได้จากการที่มีการพัฒนาแอพลิเคชั่นที่เกี่ยวข้องกับการประมวลผลภาพมากมายขึ้น ตัวอย่างเช่น การพัฒนาระบบตรวจจับรถยนต์โดยใช้กล้องวิดีโอ , การทำ motion detection , การใช้ image processing เพื่อคัดแยกวัตถุในโรงงานอุตสาหกรรม เป็นต้น ซึ่งโครงงานที่ผู้จัดทำรับผิดชอบอยู่นี้ ซึ่งเกี่ยวกับ OCR (Optical Character recognition) เองก็เป็นหนึ่งในเทคโนโลยีทางด้านการประมวลผลภาพเช่นเดียวกัน ซึ่งมีการพัฒนาขึ้นมาอย่างต่อเนื่องเรื่อย ๆ OCR นั้นเกี่ยวข้องกับการทำให้ภาพข้อความที่ถูกสแกนเข้ามาจากเครื่องสแกนสู่คอมพิวเตอร์ และทำให้ภาพกลายเป็นข้อความอักษรให้ได้ ซึ่งในการทำ OCR โดยทั่วไปนั้นก็จะแตกต่างกันไปตามลักษณะของภาษาที่ OCR ตัวนั้น ๆ สามารถรองรับได้ เพราะภาษาแต่ละภาษานั้นมีรูปแบบที่ไม่เหมือนกัน สิ่งนี้ก็มีส่วนทำให้ OCR มีความยากหรือว่าง่าย แตกต่างกันไปด้วย OCR นั้นนอกจากจะมีความเกี่ยวข้องกับเทคโนโลยีของการประมวลผลภาพแล้วยังต้องมีส่วนอื่น ๆ เข้ามาเกี่ยวข้องด้วย ซึ่งส่วนอื่น ๆ ที่ว่านี้ก็ได้แก่ขั้นตอนของการแยกแยะ รูปภาพตัวอักษร (Classification) การ classify นั้นในโครงงานที่ ผู้จัดทำได้ทำอยู่นี้ จะอาศัยทฤษฎีของโครงข่ายประสาทเทียม (Neural network) เพื่อเอามาใช้ในการ classify รูปของตัวอักษรแต่ละตัว ว่าจะเป็นตัวอักษรตัวอะไรโดยได้เลือกใช้ model แบบ back-propagation ดังนั้นการรวมระหว่างเทคโนโลยีภาพเข้ากับเทคโนโลยีของการ classify (Neural network) นั้น ก็สามารถที่จะทำให้กระบวนการทำ OCR นั้นสำเร็จลุล่วงไปตามเป้าหมายที่วางไว้ได้

Optical Thai Character Recognition

Mr.Jitkasem Pintaya 46010109

Mr.Arkarin Lounjumlern 46010947

Asst.Prof.Kietkul Jiaranaithanakit

Educational Year 2006

Abstract

At the present, processing technology has played a major role in our everyday life. The outstanding aspect of this technology is the improvement of processing applications such as the innovation of car searching by using VDO camera, motion detection, image processing when separated materials in Factories etc.

Importantly, the project that I am responsible is involved with OCR project which is one of the processing technologies as well. Furthermore, OCR project that has continually been developed has a task to transfer the image which has already been scanned by the scanner to a computer. Then, transform the image file into alphabets. The process of making OCR is different depending on the languages whose each OCR can support because each language has a unique pattern. Because of the pattern, it causes the multi-level of making OCR as well. Additionally, OCR is not only the part of image processing technology but also involved with another parts. For example, image and alphabetic classification. In this project, classification relies on the neural network theory in order to classify each alphabet by using back-propagation model. As a result, the mergence between image technology and classified technology (neural network) will make the OCR project complete in an expected duration.

กิตติกรรมประกาศ

โครงงานนี้สำเร็จลุล่วงไปด้วยดีเพราะได้รับความกรุณาจากหลายฝ่าย และขอกราบขอบพระคุณในความอนุเคราะห์ของ ผศ. เกียรติกูล เจียรนัยธนะกิจ ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงานในครั้งนี้เป็นอย่างสูงที่ได้ให้การสนับสนุนช่วยเหลือแก้ปัญหาด้านต่างๆ พร้อมให้ความรู้และคำแนะนำที่ดีสำหรับการทำโครงงานนี้ ตลอดจนช่วยเหลือในการแก้ไขข้อผิดพลาดของปริญญานิพนธ์ด้วย เป็นผลให้ตัวโครงงานและรูปเล่มรายงานในครั้งนี้สำเร็จสมบูรณ์ไปได้ด้วยดี

และขอขอบคุณพี่ๆนักวิจัยจาก NECTEC ฝ่าย RDI-3 Image technology ที่ช่วยเอื้อเฟื้อความรู้ คำแนะนำที่ดีๆ ที่ช่วยให้การทำโครงงานในครั้งนี้เป็นรูปเป็นร่างและสำเร็จลุล่วงไปได้ด้วยดี รวมทั้งอนุเคราะห์ฐานข้อมูลตัวอักษรภาษาไทยทั้งหมดที่ใช้ในโครงงานนี้ด้วย เพื่อใช้ในการ train ทั้งหมดเพื่อให้โครงงานสามารถรู้จำตัวอักษรได้

สุดท้ายนี้ ขอกราบขอบพระคุณอาจารย์ทุกท่านในภาควิชาที่ได้ประสิทธ์ประสาทวิชาความรู้ต่างๆ มา ณ ที่นี้ด้วย

ผู้จัดทำโครงงาน

สารบัญ

หน้า

สารบัญรูป

หน้า

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง

บทที่ 3 การออกแบบและพัฒนา

บทที่ 1
บทนำ

1.1 ความเป็นมาของปัญหา

เนื่องจากว่าในปัจจุบันนี้มนุษย์เราต้องการความสะดวกสะบายมากขึ้นในหลาย ๆ ด้าน จึงได้มีการพัฒนาเทคโนโลยีต่าง ๆ ขึ้นมาเพื่อจะอำนวยความสะดวกเหล่านั้นในส่วนของงานเอกสารก็เช่นกัน ก็ได้มีการคิดค้น OCR ขึ้นมา เพื่อเป็นระบบที่เอาไว้สำหรับรับรูป scan จากหนังสือหรือจากข้อความต่าง ๆ เพื่อให้ได้ output ออกมาเป็นข้อความที่อยู่ในคอมพิวเตอร์ (Text file) ซึ่งโครงงานนี้ก็มีจุดประสงค์เพื่อทำให้ได้ตามที่ว่ามานี้เช่นกัน

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อเป็นการศึกษา ทฤษฎีทางด้านการประมวลผลภาพต่าง ๆ ในโครงงาน OCR

1.2.2 เพื่อศึกษาเรียนรู้ ทฤษฎีทางด้าน neural network ในการที่ จะเอามาประยุกต์ใช้กับOCR เพื่อทำการ classify รูปภาพได้

1.2.3 ทำการพัฒนาโครงงานนี้เพื่อที่จะให้สามารถที่จะนำไปใช้งานจริงได้โดยพยายามทำให้มีข้อผิดพลาดที่น้อยที่สุดด้วย

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1.3.1 ได้รับความรู้ ความเข้าใจเกี่ยวกับกระบวนการประมวลผลภาพ อาทิเช่น การทำภาพให้เป็น grayscale , การ threshold ภาพ , การกำจัด noise , การทำ segmentation เป็นต้น

1.3.2 ได้รับความรู้ ความเข้าใจเกี่ยวกับทฤษฎีของโครงข่ายประสาทเทียม (Neural network) ทั้งในขั้นพื้นฐานไปจนถึงขั้นที่สามารถนำมาใช้ในการ classify ข้อมูลของรูปภาพได้ รวมไปถึงได้เรียนรู้เกี่ยวกับ model ต่าง ๆ ทาง neural network ที่สามารถนำมาประยุกต์ใช้ในโครงงานนี้ได้

1.3.3 สามารถสร้างโปรแกรมที่ทำการ แยกตัวอักษรแต่ละตัวออกมาจากภาพได้ , หาจำนวนของบรรทัดทั้งหมดที่อยู่ในภาพได้ รวมไปถึงแยกแยะระดับต่าง ๆ ของตัวอักษรหรือ สระ และ วรรณยุกต์ที่อยู่ในแต่ละบรรทัดได้ด้วย และที่สำคัญคือจะต้องทำการ recognize รูปภาพตัวอักษรแต่ละรูปได้ว่าเป็นตัวอักษรอะไรได้

1.3.4 สามารถนำโปรแกรมนี้ไปลองใช้งานจริง ซึ่งถ้าใช้ได้จริงก็จะถือว่าเป็นประโยชน์ในการที่จะไม่ต้องจ่ายเงินเพื่อค่าของ software สำหรับ OCR อื่น ๆ

1.4 ขอบเขตของโครงงาน

สำหรับในโครงงานนี้ได้วางขอบเขตสำหรับแต่ละเทอมไว้ดังนี้

<u>เทอม 1</u>

- พัฒนาส่วนของการรับภาพจากผู้ใช้ รวมไปถึงส่วนของการแสดงภาพให้ผู้ใช้เห็น

- พัฒนาในส่วนของระบบประมวลผลภาพต่าง ๆ อาทิเช่น การทำให้ภาพเป็นแบบ grayscale, การทำ adaptive threshold, การกำจัด noise ต่าง ๆ การทำ segmentation ฯลฯ

- ทำการจัดหา training data (ซึ่ง training data ทั้งหมดที่ใช้ในโครงงานนี้เป็นเพียง training data บางส่วนของ training data ทั้งหมดที่ NECTEC ใช้) รวมทั้งการปรับปรุง และแก้ไข training data ทั้งหมดให้มีความเหมาะสมที่จะใช้ในโครงงานนี้

- พัฒนาอัลกอริธึมในส่วนของการหาระดับต่าง ๆ ของข้อความ เนื่องจากว่าภาษาไทยนั้นมีข้อแตกต่างจากภาษาอังกฤษตรงที่ว่าในข้อความหนึ่ง ๆ นั้น มีได้อยู่ 4 ระดับ (ระดับสระ อุ อู, ระดับที่วางพยัญชนะ, ระดับสระ อิ อี, ระดับ วรรณยุกต์) และการที่เรารู้ว่า ตัวอะไรอยู่ในระดับไหนของแต่ละบรรทัดนั้นก็มีความจำเป็นอย่างยิ่ง เพราะจะใช้เมื่อตอนที่เราจะเอาตัวอักษรแต่ละตัวมาประกอบกันเป็นข้อความ

- พัฒนาส่วนของ neural network สำหรับการ train ซึ่งในเทอมนี้ผู้จัดทำได้พยายามพัฒนาส่วนของ neural network ให้สามารถทำงานได้ก่อน และทำการ train ได้เช่นกัน แต่ว่ายังไม่ดีนัก (model ที่ใช้สำหรับการ train ในตอนนี้คือ back-propagation)

<u>เทอม 2</u>

- พัฒนาเพิ่มเติมในส่วนของการติดต่อกับผู้ใช้ (graphical user interface) ให้ผู้ใช้สามารถใช้งานโปรแกรมได้ง่ายขึ้น

- จัดการพัฒนาในส่วนของ neural network ให้ดีขึ้น โดยเฉพาะในส่วนของการ train นั้น ซึ่งในเทอมนี้จะเปลี่ยนการ train ใหม่ โดยอาศัยการ train แบบที่ใช้ข้อมูลที่เป็น unknown มาใช้เพื่อวิเคราะห์ว่าการ train ในช่วงนั้น ๆ ให้ผลลัพธ์ที่ดีมากน้อยเพียงใด โดยการเอาข้อมูลที่เป็น unknown มาลองทำการ recognize ดู แล้วก็ตรวจดูว่า output ที่ได้จาก neural network นั้นผิดพลาดไปจากข้อมูลที่เป็น unknown มากน้อยเพียงใด วิธีการแบบนี้อาศัย training set (ข้อมูลที่เอาไว้สำหรับ train จริง ๆ) กับ validation set (set ของข้อมูลที่เอาไว้สำหรับตรวจสอบการ train)

- ทำ graphic display ในส่วนของการ train ของ neural network เพื่อแสดงให้เห็นถึง graph ของการ train ในขณะนั้น ๆ

- อนุญาติให้ผู้ใช้ทำการแก้ไขข้อมูลต่าง ๆ ของภาพ หรือ ของตัวอักษรแต่ละตัวก่อนที่จะเอาข้อมูลของตัวอักษรแต่ละตัวไปทำการ recognize จริง เช่น การอนุญาติให้ผู้ใช้สามารถแก้ไขขอบเขตของ blob ตัวอักษรแต่ละตัวได้ ในกรณีที่ผู้ใช้เห็นว่า blob ของตัวอักษรใด ๆ สองตัวนั้นมีการทับกัน ติดกัน ก็

อนุญาติให้ผู้ใช้ ทำการแยก blob สองตัวนั้นออกมาจากกันได้

1.5 ส่วนประกอบของรายงาน

เนื้อหาของรายงานชิ้นนี้จะประกอบไปด้วยทั้งสิ้น 4 บท โดยในแต่ละบทจะมีเนื้อหาอย่างคร่าว ๆ ดังต่อไปนี้

<u>บทที่ 2</u>

จะพูดถึง ทฤษฎีต่าง ๆ ที่เกี่ยวข้องกับโครงงานนี้ ซึ่งในบทนี้จะประกอบไปด้วย 2 ส่วนหลัก ๆ ซึ่งได้แก่ ทฤษฎีทางด้านการประมวลผลภาพ และอีกส่วนก็จะเกี่ยวกับ ทฤษฎีทางด้าน neural network

<u>บทที่ 3</u>

จะเป็นส่วนของการออกแบบโครงงานนี้ ในบทนี้จะพูดถึงขั้นตอนต่าง ๆ รวมถึงการทำงานของโปรแกรม เริ่มตั้งแต่การโหลดภาพ การประมวลผลภาพต่าง ๆ , ขั้นตอนของการหาจำนวนของบรรทัดในภาพที่เรารับเข้ามา, ขั้นตอนของการหาระดับย่อย ๆ ในแต่ละบรรทัด (ซึ่งก็จะมีระดับย่อย ๆ ในแต่ละบรรทัดไม่เกิน 4 ระดับ), การแปลงข้อมูล ภาพเพื่อให้อยู่ในรูปแบบที่เหมาะสม เพื่อที่จะเอาไปเข้าสู่กระบวนการทางการ train โดยใช้ neural network หรือการ recognize โดยใช้ neural network

<u>บทที่ 4</u>

จะพูดถึงการสรุปการทำงานของโปรแกรมที่ได้พัฒนามาจนถึงขณะนี้ ในบทนี้จะพูดถึงผลการทดลองต่าง ๆ เมื่อ input ภาพข้อความให้กับโปรแกรมรวมถึงผลการทดลองที่ได้ทำการ recognize ภาพอย่างคร่าว ๆ ด้วย

<u>บทที่ 5</u>

จะเป็นบทที่สรุปผลการทำงานทั้งหมด ข้อจำกัดของโครงงานทั้งหมด รวมทั้งการวิจารณ์ผลการทดลอง

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

ในบทนี้จะว่าด้วยเรื่องของทฤษฎีต่าง ๆ ที่เกี่ยวข้องกับโครงงานนี้ ซึ่งแบ่งออกได้เป็น 2 ส่วน คือ ทฤษฎีทางด้านการประมวลผลภาพ และ ทฤษฎีทางด้านของ Neural network ซึ่งมีเนื้อหาดังต่อไปนี้

2.1 ทฤษฎีทางด้านการประมวลผลภาพ

หลังจากที่เรา scan รูปเข้ามาแล้ว ขั้นตอนที่เราจะทำต่อไปก็คือการ ทำภาพให้อยู่ในรูปของ grayscale เพื่อที่จะเอารูปนั้นมาทำการ threshold อีกทีหนึ่ง สำหรับขั้นตอนของการทำภาพจาก rgb ให้เป็น grayscale นั้นสามารถทำได้ดังขั้นตอนต่อไปนี้

เนื่องจากว่า model สีแบบ rgb นั้นประกอบด้วย 3 channel คือ r ขนาด 8 bit , g ขนาด 8 bit และ b ขนาด 8 bit เช่นกัน ดังนั้นจึงเก็บโดยใช้ขนาด 24 bit หรือ 3 byte ต่อ 1 pixel เมื่อทำการ convert มาเป็น grayscale จะใช้ เพียงแค่ 1 byte ในการเก็บแต่ละ pixel

$$Y = 0.3*R + 0.59*G + 0.11*B$$

R คือ component ของสีแดง

G คือ component ของสีเขียว

B คือ component ของสีน้ำเงิน

Y คือค่าของ grayscale

หลังจากที่เราเอาภาพมาทำเป็น grayscale ขั้นตอนต่อไปที่เราทำก็คือการเอาภาพที่เป็น grayscale มาทำการ threshold เพื่อทำภาพให้อยู่ในรูปของ binary หรือว่าภาพที่มีสองสีนั้นเอง(ได้แก่สีขาวกับสีดำ) ที่เราต้องเอาภาพมาทำการ binarization ก็เพื่อที่จะเอาภาพนั้นมาทำการหาบริเวณของตัวอักษร สำหรับการ threshold นั้นมีสอง คือ

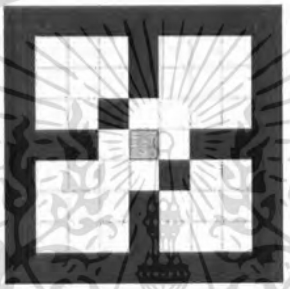
1. การ threshold แบบธรรมดา

การ threshold แบบนี้เรียกได้อีกอย่างหนึ่งว่า global threshold เนื่องจากว่าเราจะเลือกค่าที่จะเอามาทำการ threshold กับภาพเพียงแค่ค่าเดียวทั้งนั้น และ apply กับทุก ๆ ค่าpixel ในภาพเลย

2. การ threshold แบบ adaptive

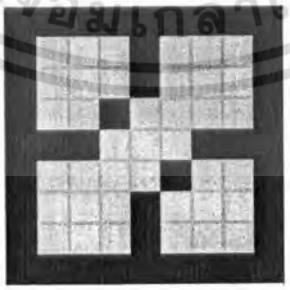
วิธีการก็ตรงตามชื่อของมันคือ การ threshold แบบนี้จะเลือกค่าที่จะเอามาทำการ threshold ได้หลาย ๆ ค่า โดยเราก็จะ apply ค่าที่เลือกได้นี้กับบริเวณใดบริเวณหนึ่งที่อยู่ในภาพเท่านั้นไม่ใช้กับทุก ๆ pixel ในภาพหรอก ส่วนใหญ่บริเวณที่เราเลือก apply นั้นมักจะเป็นบริเวณที่เป็นขนาดสี่เหลี่ยมที่ได้มีการกำหนดไว้จากผู้เขียนโปรแกรมเอง

เมื่อได้ภาพที่เป็นแบบ binary แล้ว(ซึ่งก็คือภาพที่เกิดจากกระบวนการ threshold) ขั้นตอนต่อไปที่เราจะทำก็คือเราจะเอาภาพแบบ binary ข้างต้นมาหา blob (blob คือกลุ่มของ pixel ที่อยู่ในภาพที่เราสนใจที่จะเอามาทำอะไรซักอย่างหนึ่ง ในที่นี้ซึ่งเป็นการทำ OCR blob ก็จะหมายถึงตัวอักษรที่เป็นภาพ ซึ่งจะแสดงเป็นสีขาว ส่วนตัวที่เราไม่สนใจ หรือว่าไม่ใช่ตัวอักษรนั้นก็จะถูกแสดงด้วยสีดำ) ในการหา blob นั้นเราจะต้องทำการแยก blob ทั้งหมดออกมาจากภาพให้ได้ ซึ่งวิธีการที่เขาใช้กันก็มีอยู่ได้หลายวิธี แต่วิธีที่จะนำเสนอในที่นี้เรียกว่าการ flood fill ซึ่ง flood fill นั้นหมายถึงการเติมให้เต็ม แต่เราสามารถใช้ flood fill เพื่อที่จะบ่งบอกถึงขอบเขตของ blob รวมทั้งที่ตั้งของ blob นั้น ๆ ได้ด้วยการ flood fill สามารถ แสดงให้เห็นได้ดังนี้



รูปที่ 2.1 ภาพแสดงตำแหน่งเริ่มต้นของการ flood fill

รูปที่ 2.2 การทำ flood fill แบบ 4 ทิศทาง



รูปที่ 2.3 การทำ flood fill แบบ 8 ทิศทาง

ในที่นี้ได้เลือกใช้วิธีการ flood fill แบบ 8 ทิศทางเนื่องจากว่าจะให้ผลลัพธ์ที่ดีกว่าและแสดงถึงการเชื่อมได้ดีกว่า แบบ 4 ทิศทาง นอกจากนี้การ implement อัลกอริธึมของการ flood fill นั้นได้ใช้ stack มาช่วยด้วยเพราะเห็นว่าการเรียกใช้ อัลกอริธึม flood fill แบบ recursive นั้นอาจจะทำให้ stack เต็มได้

การ flood fill ใน project นี้ ได้ใช้การ flood fill จากสีขาวซึ่งเป็นสีของ blob ที่เราสนใจ เราจะ flood บริเวณที่เป็นสีขาวให้เป็นสีดำ เนื่องจากจะเป็นการตัด blob ที่เราได้ทำการ flood fill เสร็จแล้วทิ้งไปจะได้มาต้องมา flood fill ซ้ำหลายรอบ

สำหรับ code คร่าว ๆ ของ algorithm การ flood fill นั้นแสดงได้ดังนี้

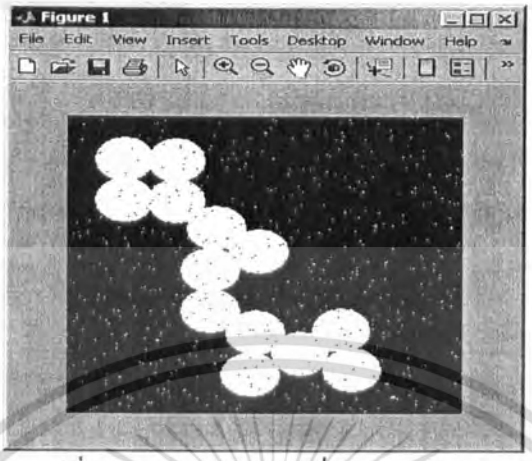
```
void floodFill8Stack(int x, int y, int newColor, int oldColor)
{
  if(newColor == oldColor) return; //avoid infinite loop
  emptyStack();

  if(!push(x, y)) return;
  while(pop(x, y))
  {
    screenBuffer[x][y] = newColor;
    if(x + 1 < w && screenBuffer[x + 1][y] == oldColor)
    {
      if(!push(x + 1, y)) return;
    }
    if(x - 1 >= 0 && screenBuffer[x - 1][y] == oldColor)
    {
      if(!push(x - 1, y)) return;
    }
    if(y + 1 < h && screenBuffer[x][y + 1] == oldColor)
    {
      if(!push(x, y + 1)) return;
    }
    if(y - 1 >= 0 && screenBuffer[x][y - 1] == oldColor)
    {
```

```
        if(!push(x, y - 1)) return;
      }
      if(x + 1 < w && y + 1 < h && screenBuffer[x + 1][y + 1] == oldColor)
      {
        if(!push(x + 1, y + 1)) return;
      }
      if(x + 1 < w && y - 1 >= 0 && screenBuffer[x + 1][y - 1] == oldColor)
      {
        if(!push(x + 1, y - 1)) return;
      }
      if(x - 1 >= 0 && y + 1 < h && screenBuffer[x - 1][y + 1] == oldColor)
      {
        if(!push(x - 1, y + 1)) return;
      }
      if(x - 1 >= 0 && y - 1 >= 0 && screenBuffer[x - 1][y - 1] == oldColor)
      {
        if(!push(x - 1, y - 1)) return;
      }
    }
}
```

Noise reduction ใน OCR

สำหรับภาพที่เราได้รับเข้ามานั้นบางทีแล้วอาจจะมี noise ปะปนเข้ามากับภาพก็ได้ เนื่องจากว่าเราต้องรับภาพจากเครื่อง scan เพราะฉะนั้น noise ชนิดหนึ่งที่อาจจะเกิดขึ้นได้ก็คือ salt and pepper noise ซึ่ง noise ชนิดนี้ ถ้าหากว่าเราไม่ทำการกำจัดมันออกไปก่อนที่จะเอารูปนั้นไปเข้าสู่กระบวนการทางด้าน recognition อื่น ๆ ก็จะทำให้กระบวนการทางด้าน image (recognition) เกิดปัญหาขึ้นได้ อย่างเช่นทำให้เราได้สิ่งที่เราไม่ต้องการแทนที่เราจะได้รูปของตัวอักษรจริง ๆ เป็น ต้น salt and pepper noise เมื่อเกิดขึ้นกับรูปใด ๆ แล้ว ก็จะทำให้รูปนั้นมีลักษณะคล้าย ๆ กับรูปดังแสดงข้างล่างนี้



รูปที่ 2.4 ภาพแบบไบนารีที่มี noise ปนอยู่

รูปที่ 2.5 ภาพแบบไบนารีที่ผ่านการกำจัด noise แล้ว

จากรูปจะเห็นว่า salt and pepper noise นั้นทำให้เกิดจุดบริเวณเล็ก ๆ โดด ๆ โดยจะกระจายทั่ว ๆ ไปในบริเวณของรูปนั้น ๆ ถ้าเราต้องการที่จะกำจัด salt and pepper noise เราก็จะต้องลบจุดสีดำที่เรียกกันว่า pepper ทิ้งไป และนอกจากนี้ salt and pepper noise ยังทำให้เกิดหลุมในรูปได้เช่นกัน ซึ่งเราเรียกหลุมนี้กัน ว่า salt

หนึ่งในวิธีการที่ใช้ในการกำจัด salt and pepper noise แบบนี้ก็คือ การใช้ median filtering แต่ว่าสำหรับใน project ที่ทำอยู่ในขณะนี้ จะใช้วิธีการที่คล้าย ๆ กันก็คือ จะทำการกำหนด window ขนาด 3 X 3 ซึ่งเราจะเอา window นี้ไปวางไว้ตรงจุดที่เราสนใจโดยให้จุดตรงกลางของ window ทับกันกับจุดที่เราสนใจ จากนั้นเราก็จะนับจำนวนของจุดสีขาวทั้งหมดที่อยู่กายใน window นี้ แล้วก็ทำการเปรียบเทียบดูว่าถ้า จำนวนของจุดสีขาวที่นับได้นั้นมีค่าน้อยกว่า 2 ก็จะกำหนดจุดที่เราสนใจให้

กลายเป็นสีดำ แต่ถ้าหากว่าจุดสีขาวที่นับได้นั้นมีค่ามากกว่า 5 ก็จะกำหนดจุดที่เราสนใจให้กลายเป็นสีขาว ถ้านอกเหนือจากนั้นก็จะกำหนดให้จุดที่เราสนใจ ไม่มีการเปลี่ยนแปลงคือเป็นสีอะไรมาก่อนก็เป็นสีนั้น ๆ อยู่อย่างเดิม

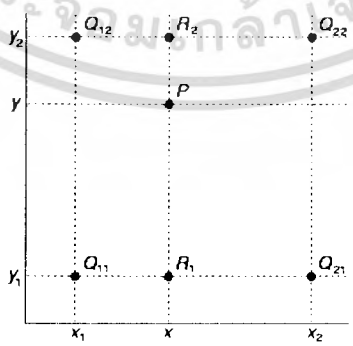
การ Interpolation

เนื่องจากว่าเมื่อเราสามารถที่จะแยก blob ของตัวอักษรออกมาได้หมดแล้ว แต่ว่า blob ที่แยกได้นั้นอาจจะมีขนาดไม่เท่ากันกับขนาดของ blob ซึ่งใช้ในการ train ที่เอาไว้ใช้ในการ recognition ดังนั้นเพื่อเป็นการแก้ปัญหาสำหรับภาพที่รับเข้ามา ที่มีขนาดของตัวอักษรต่าง ๆ กันได้ เราเลยจำเป็นที่จะต้องทำการ scale blob ให้อยู่ในขนาดที่เป็น มาตรฐาน ขนาดใดขนาดหนึ่งซะก่อน ถ้าหากว่า blob ใด ๆ มีขนาดที่เล็กกว่าขนาดมาตรฐาน ก็จำเป็นจะต้องขยายขนาดให้ใหญ่ขึ้น และถ้าหากว่า blob มีขนาดที่ใหญ่กว่าขนาดมาตรฐาน ก็จำเป็นจะต้องลดขนาดให้เล็กลงด้วย กระบวนการย่อหรือขยายนี้เราเรียกว่าเป็น การ interpolation (หรือเรียกว่าการ scaling ก็ได้เช่นกัน) เทคนิคที่ใช้ในการ interpolate ในปัจจุบันมีอยู่หลายเทคนิคมากมาย อาทิเช่น การใช้วิธี interpolate แบบ nearest neighbor , bilinear interpolation , bicubic interpolation เป็นต้น แต่ในโครงงานนี้ วิธีที่ผู้จัดทำได้ใช้ คือได้เลือกใช้วิธีการของ bilinear interpolation ซึ่งถือว่าเป็นวิธีที่มีความซับซ้อนไม่มากนัก และก็ง่ายต่อการ implement ด้วย

ต่อไปจะพูดถึงขั้นตอนวิธีในการทำ bilinear interpolation ซึ่งมีขั้นตอนดังต่อไปนี้

ในทางคณิตศาสตร์ bilinear interpolation คือส่วนเพิ่มเติมของ linear interpolation เพื่อที่จะเอาไว้สำหรับ interpolate ฟังก์ชันใด ๆ ที่มี 2 ตัวแปร ไอเดียสำคัญคือ จะทำ linear interpolation ก่อนในทิศทางหนึ่ง และหลังจากนั้นก็จะทำการ linear interpolation ในทิศทางอื่น ๆ ต่อไป

สมมุติว่าเราต้องการที่จะหาค่าของฟังก์ชัน f ที่เราไม่เคยรู้ ซึ่งอยู่ที่จุด P = (X,Y) เราจะสมมุติว่า เรารู้ค่าของ f ที่จุด 4 จุด ซึ่งได้แก่ จุด Q11 = (x1,y1) , จุด Q12 = (x1,y2) , จุด Q21 = (x2,y1) และจุด Q22 = (x2,y2) ดังรูปที่อยู่ด้านล่างนี้



รูปที่ 2.6

จุดสีแดงสี่จุดแสดงจุดของข้อมูลที่เรารู้ค่า

จุดสีเขียวคือจุดที่เราต้องการที่จะ interpolate

อันดับแรกเราจะทำ linear interpolation ในทิศทางตามแกน x ซึ่งก็จะทำให้ได้สมการออกมาดังรูปข้างล่างนี้

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \quad \text{where} \quad R_1 = (x, y_1), \tag{2.1}$$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \quad \text{where} \quad R_2 = (x, y_2). \tag{2.2}$$

เราจะทำต่อไปโดยการ interpolate ในทิศทางของแกน y ด้วยเช่นกัน ทำให้ได้

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2). \tag{2.3}$$

ในที่สุดเราจะได้ค่าของ f(x,y) ที่ต้องการโดยประมาณเป็นดังนี้

$$f(x, y) \approx \frac{f(Q_{11})}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y_2 - y) + \frac{f(Q_{21})}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y_2 - y) \tag{2.4}$$

$$+ \frac{f(Q_{12})}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y - y_1) + \frac{f(Q_{22})}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y - y_1). \tag{2.5}$$

ถ้าเราเลือกระบบ coordinate ที่ซึ่งจุด 4 จุดของ f นั้นถูกรู้ว่าเป็น (0,0), (0,1), (1,0), และ (1,1) ดังนั้นสูตรในการ interpolation ก็จะทำให้อยู่ในรูปอย่างง่ายได้เป็น

$$f(x, y) \approx f(0, 0)\,(1 - x)(1 - y) + f(1, 0)\,x(1 - y) + f(0, 1)\,(1 - x)y + f(1, 1)xy. \tag{2.6}$$

หรือถ้าจะเขียนให้อยู่ในรูปของการกระทำทาง matrix ก็จะเขียนได้เป็น

$$f(x, y) \approx \begin{bmatrix} 1 - x & x \end{bmatrix} \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1 - y \\ y \end{bmatrix} \tag{2.7}$$

ในทางตรงกันข้ามกับสิ่งที่เคยกล่าวมา การ interpolate นั้นจะไม่เป็น linear เพราะมันอยู่ในรูปด้านล่างนี้แทน

$$(a_1x + a_2)(a_3y + a_4), \tag{2.8}$$

ดังนั้นมันจะเป็นผลคูณของ linear function 2 อัน ในทางกลับกันเราสามารถที่จะเขียนการ interpolate ให้อยู่ในรูปดังข้างล่างนี้ได้

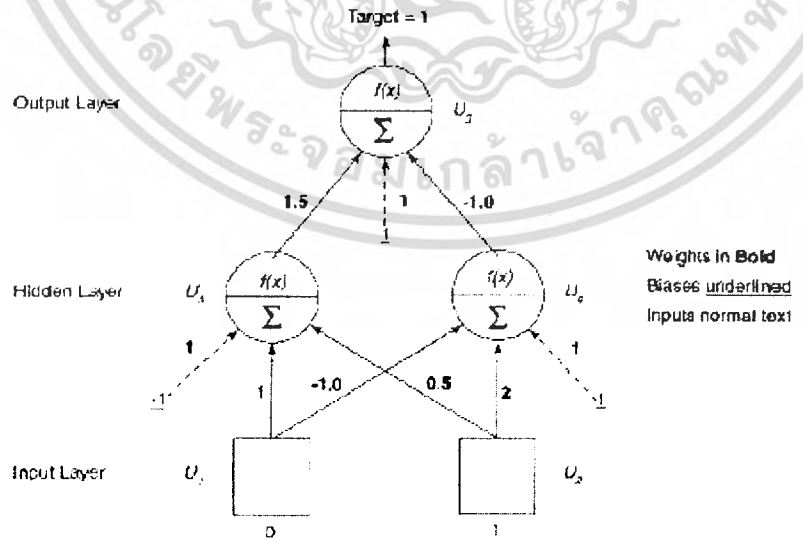
$$b_1 + b_2x + b_3y + b_4xy. \tag{2.9}$$

ในสองกรณีข้างต้น จำนวนของค่าคงที่ (ซึ่งมี 4 อัน) สัมพันธ์กับจำนวนของจุดข้อมูล เมื่อ f ถูกกำหนดค่าให้

ผลลัพธ์ของ bilinear interpolation จะเป็นอิสระจากลำดับของการ interpolate ถ้าหากว่าเราทำการ linear interpolation ในทิศทางของแกน y ก่อนอันดับแรก จากนั้นทำในทิศทางของแกน x ค่าประมาณของผลลัพธ์ที่ได้ก็จะมีค่าที่เท่ากัน

2.2 ทฤษฎีทางด้าน Neural network

สำหรับในส่วนของการ recognition สำหรับตัว ocr นั้น จากการที่ได้ไปศึกษา paper ต่างๆมานั้น ส่วนใหญ่แล้วก็จะใช้วิธีการที่เรียกกันว่า Back-propagation หรือการเรียนรู้แบบโครง่ข่ายประสาทเทียมแบบแพร่กระจายย้อนกลับ สำหรับตัว neural network ที่ข้าพเจ้าได้สร้างมานั้นจะเป็น neural network แบบที่มีหลายๆชั้นหรือที่เรียกกันว่า Multiple-Layer Perceptrons (MLP) โดยปกติแล้วตัว MLP เองก็จะประกอบไปด้วย layer ที่สำคัญ 3 layer คือ input layer , hidden layer , output layer ส่วนใหญ่เท่าที่พบมานั้น MLP ก็จะมีอยู่ 3 layer ก็เพียงพอที่จะใช้แก้ปัญหาได้เป็นจำนวนมาก แต่บางที MLP อาจจะมีถึง 4 layer ก็เป็นไปได้เช่นกัน รูปข้างล่างนี้แสดงถึง multiple layer network



รูปที่ 2.7 โครงสร้างอย่างง่ายของ MLP

สำหรับวิธีการของ back-propagation จะเป็นดังนี้คือ

อัลกอริธึมจะเริ่มต้นโดยการกำหนด weight ซึ่งเป็นเส้นเชื่อมระหว่าง node ต่างๆที่อยู่ในแต่ละ layer เข้าด้วยกัน โดยเราจะกำหนดค่าให้กับแต่ละ weight แบบ random ด้วย หลังจากที่เรากำหนดค่าแบบ random ให้กับ weight แล้วกระบวนการที่จะกล่าวต่อไปนี้จะมีการทำซ้ำจนกระทั่งค่าของ mean-squared error (MSE) ของ output layer มีค่าที่น้อยเพียงพอ

ในการที่จะใช้ neural network แบบ backpropagation นั้น เราจะต้องทำให้มันเกิดการเรียนรู้ก่อน โดยเราจะต้องทำการ train neural network เพื่อให้มันสามารถที่จะใช้ในการ recognition อะไรก็ตามที่เหมือนหรือคล้ายๆกับ ตัวอย่างที่เอาไป train ได้ ดังนั้นสิ่งที่เราจะต้องมีสำหรับ neural network แบบนี้ก็คือ training data

0.ทำการกำหนด weight ให้กับเส้นเชื่อมที่เชื่อม node ต่าง ๆ อย่าง random ก่อน

1.ใส่ data ตัวอย่างที่เราต้องการจะ train ให้กับ input layer และ output ที่ถูกต้องของ data ตัวอย่างให้กับ output layer

2.คำนวณการแพร่กระจายไปข้างหน้าของ data ตัวอย่างผ่าน network (คำนวณค่าผลรวมของ weight ของเส้นเชื่อมใน network, Si, และค่าที่ได้จากฟังก์ชัน activation, ui, สำหรับทุก ๆ เซลล์)

3.เริ่มต้นที่ output ให้มองย้อนผ่านทาง output และ เซลล์ที่อยู่ตรงกลาง คำนวณค่าของ errorตามสมการข้างล่างนี้ด้วย

$$\delta_o = (C_i - u_5)u_5(1-u_5) \qquad \text{For the output cell}$$

$$\delta_i = \left(\sum_{m:m>i} w_{m,j}\delta_o\right)u_i(1-u_i) \qquad \text{For all hidden cells} \tag{2.10}$$

(สังเกตว่า mจะหมายถึงเซลล์ทั้งหมดที่ถูกเชื่อมไปกับ hidden node w คือค่า weight ที่กำหนดให้ และ u คือ ค่าที่ได้จาก activation function)

4.ในท้ายที่สุด weight ที่อยู่ภายใน network จะถูก update ใหม่ตามสมการข้างล่างนี้

$$w^*_{i,j} = w_{i,j} + \rho\delta_o u_i \qquad \text{For weights connecting hidden to output}$$

$$w^*_{i,j} = w_{i,j} + \rho\delta_i u_i \qquad \text{For weights connecting hidden to output} \tag{2.11}$$

เมื่อ p หมายถึง learning rate (หรือขนาดของการก้าว) ค่าเล็ก ๆนี้ จะเป็นตัวจำกัดการเปลี่ยน แปลงที่อาจจะเกิดขึ้นระหว่างแต่ละ step ค่าของ p สามารถถูกจูนเพื่อจะใช้ในการตัดสินว่าอัลกอริธึมแบบ back-

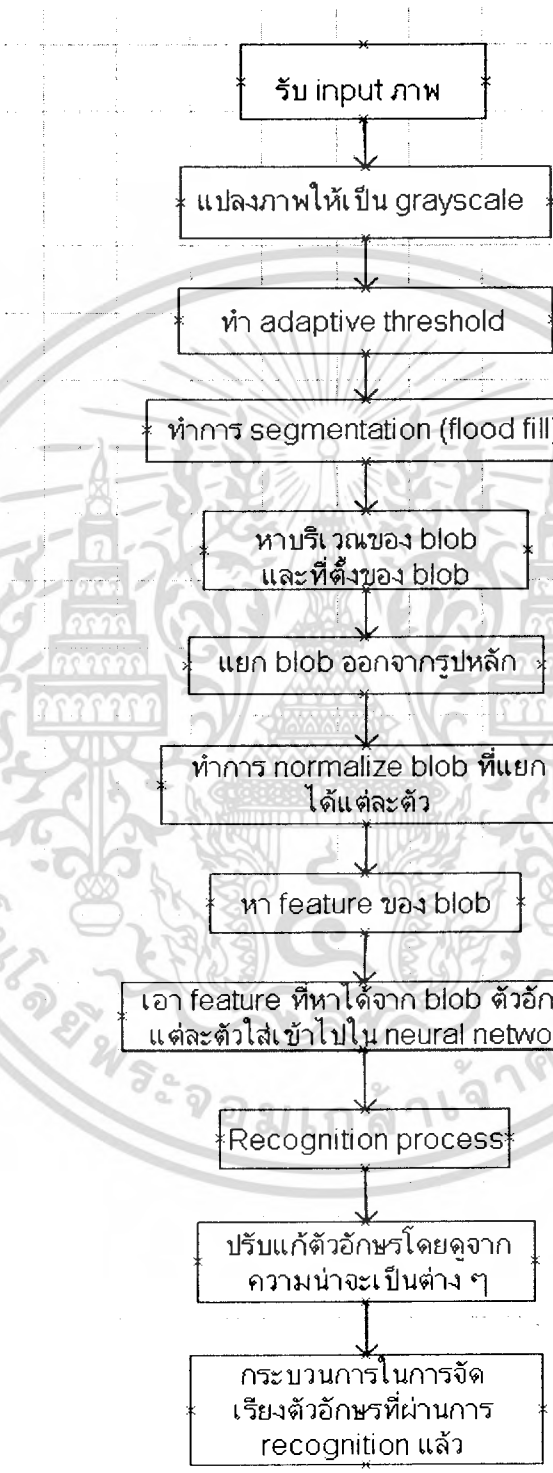
propagation จะเข้าถึงจุดที่ต้องการที่เป็นคำตอบได้เร็วเพียงใด เราควรที่จะกำหนดค่าเริ่มต้นให้กับ p ให้เป็นค่าน้อย ๆ(0.1) เพื่อเป็นการทดสอบจากนั้นก็ค่อย ๆ เพิ่มค่าขึ้น

สำหรับการหาค่าของ learning rate นั้นมี trick เล็กน้อย คือ ถ้าหากว่าเรา กำหนดค่าของ learning rate ที่มากเกินไปก็จะทำให้กระบวนการของ neural network นั้นรวดเร็ว แต่ว่าค่าที่มากเกินไปนี้เองอาจจะทำให้เกิดการออสซิเลตได้เนื่องจากเมื่อใกล้ เข้าสู่ solution มันอาจจะเลยไปได้จึงต้องย้อนกลับไปมาหลายรอบ แต่ถ้าหากว่าเรากำหนดค่าของ learning rate ที่น้อยเกินไปก็อาจทำให้กระ บวนการของ neural network นั้นช้าเกินไปกว่าที่เราจะเข้าหาคำตอบได้ ดังนั้นทางที่ดีควรจะมีการทำ neural network ที่สามารถที่จะปรับค่าของ learning rate ในขณะที่กำลัง train อยู่ได้

บทที่ 3

การออกแบบและพัฒนา

Block Diagram



Blob คือ บริเวณอิสระต่าง ๆ ในภาพที่เราสนใจ และสังเกตได้เพราะมันแยกกันอยู่ชัดเจน

ในบทนี้จะพูดถึงการออกแบบโครงงานซึ่งประกอบไปด้วยหลาย ๆ ส่วน ดังนี้

3.1 กระบวนการในการรับ input ภาพ

ใน project นี้ สิ่งสำคัญที่จำเป็นต้องมีก็คือ ส่วนของการรับภาพ ในการรับภาพของ project นี้เราได้อาศัย library เข้ามาช่วย เพื่อให้การจัดการเกี่ยวกับการ load ภาพ และ save ภาพ ทำได้สะดวกมากยิ่งขึ้น และ library ที่เราได้เลือกมาก็คือ opencv opencv นั้นเป็น library เกี่ยวกับ image processing ที่ open source และยังไม่ต้องเสียเงินอีกด้วย จุดเด่นของ opencv คือความง่ายในการใช้งาน ซึ่งทำให้เราสามารถเขียน code เพื่อทำการรับภาพ (load) หรือการ save ภาพได้เพียงแค่การเรียกใช้ function เพียง function เดียวเท่านั้น นอกจากนี้เรายังใช้ opencv ในการ show ภาพด้วย ต่อไปจะกล่าวถึงรูปแบบของ function ต่างๆใน opencv ที่ได้นำมาใช้ใน project นี้

การ load ภาพ

การ load ภาพทำได้โดยการเรียกใช้ function cvLoadImage ซึ่งมี prototype เป็นดังนี้

IplImage* cvLoadImage(const char* filename,int iscolor = 1);

Filename --> ชื่อของ file ที่จะถูก load

Iscolor --> รูปแบบสีจำเพาะของรูปภาพที่ถูก load ถ้ามากกว่า 0 รูปภาพที่ load จะโดนจัดให้เป็นรูปภาพ 3-channel ถ้าเท่ากับ 0 รูปภาพที่ load จะโดนจัดให้เป็น grayscale ถ้าน้อยกว่า 0 รูปภาพที่ load จะถูก load มาเป็น (ด้วยตัวเลขของ channel ขึ้นอยู่กับ file) ฟังก์ชัน cvLoadImage load รูปภาพจาก file จำเพาะ และ ส่งค่า pointer มาที่รูปภาพที่ load ในขณะนี้รูปแบบรูปภาพตามนี้ได้ถูกสนับสนุน

Windows bitmaps	- BMP, DIB;
JPEG files	- JPEG, JPG, JPE;
Portable Network Graphics	- PNG;
Portable image format	- PBM, PGM, PPM;
Sun rasters	- SR, RAS;
TIFF files	- TIFF, TIF.

การ save ภาพ

การ save ภาพทำได้โดยการเรียกใช้ function cvSaveImage ซึ่งมี prototype เป็นดังนี้

Int cvSaveImage(const char* filename,const CuArr* image);

Filename --> ชื่อของ file

Image --> รูปภาพที่ถูก save

ฟังก์ชัน cvSaveImage save รูปภาพไว้ที่ file จำเพาะ รูปแบบรูปภาพที่ถูกเลือกจะขึ้นอยู่กับ filename ที่ขยายออก มีเพียงรูปภาพ 8 bit single-channel หรือ 3-channel(ด้วย 'BGR' channel) ที่สามารถถูก save โดยใช้ฟังก์ชันนี้ ถ้ารูปแบบ,ความซับซ้อน หรือ ระดับ channel แตกต่างกัน ใช้ cvCvtScale และ cvCvtColor เปลี่ยนมันก่อนการ save หรือใช้ cvSave ที่มีโดยทั่วไป save รูปภาพในรูปแบบของ XML หรือ YAML

หลังจากที่เรา load ภาพหรือ save ภาพได้แล้ว ก็จำเป็นจะต้องมีการแสดงภาพด้วย ซึ่ง function ใน opencv ที่เกี่ยวข้องกับการแสดงภาพนั้นมีดังนี้

Int cvNamedWindow(const char* name,int flags);

function นี้เอาไว้ใช้สำหรับตั้งชื่อให้กับ window ที่จะเอามาแสดงภาพ และกำหนด property บาง อย่างให้กับ window โดยที่

Name ---> ชื่อของ Window ที่ซึ่งจะถูกใช้บ่งบอก window และจะปรากฏอยู่ที่ส่วนบรรยายใต้ window

Flags ---> Flags ของ Window ในขณะนี้มีเพียงตัวเดียวที่ยังสนับสนุน flag คือ CV_WINDOW_AUTOSIZE ถ้ามัน set ขนาดของ window จะปรับเปลี่ยนขนาดตัวมันเองโดยอัตโน มัติเพื่อให้เหมาะสมกับการแสดงรูปภาพในขณะที่ user ไม่สามารถปรับเปลี่ยนขนาดเองได้ ฟังก์ชัน cvNamedWindow สร้าง window ที่ซึ่งสามารถใช้ placeholder สำหรับ images และ trackbars window ที่สร้างอ้างอิงโดยชื่อของมัน

ถ้า window มีชื่อของมันเองอยู่แล้ว ฟังก์ชันนี้จะไม่ทำอะไร

Void cvShowImage(const char* name,const CvArr* image);

function นี้เอาไว้สำหรับแสดง window ที่มีรูปภาพออกมาให้ user เห็น โดยที่

Name ---> ชื่อของ window

Image ---> รูปภาพที่จะทำการแสดง

ฟังก์ชัน cvShowImage แสดงรูปภาพใน window ที่จำเพาะ ถ้า window นั้นถูกสร้างขึ้นมาโดย CV_WINDOW_AUTOSIZE flag แล้ว รูปภาพจะถูกแสดงด้วยขนาดดั้งเดิม ถ้าไม่เช่นนั้น รูปภาพจะถูกสเกลให้เหมาะสมกับ window

เมื่อเราสามารถ load รูปภาพได้แล้ว ซึ่งรูปภาพที่เรา load เข้ามาจะต้องอยู่ในรูปแบบของ rgb 24 บิต (ประกอบด้วย r = 8 บิต, g = 8 บิต, b = 8 บิต)และเป็นภาพที่มี format เป็น bmp ด้วยขนาดของรูปจะมีขนาดเท่าไหร่ก็ได้ หลังจากที่เรารับภาพเข้ามาได้แล้ว เราก็จะทำการ casting รูปที่เข้ามาโดยเก็บรูปภาพโดยใช้เป็น pointer ของ char แทน เนื่องจากการเก็บข้อมูลของรูปภาพนั้น เมื่ออยู่ในหน่วยความจำมักจะเก็บเรียงกันไปเรื่อยเป็นแถวยาวซึ่งจะไม่ใช่เป็นอาเรย์ขนาด 2 มิติแน่นอนและเนื่องจากรูปภาพนั้นเก็บทีละ 8 บิตสำหรับ r, g, b ตามลำดับ ดังนั้นจึงมีความเหมาะสมแล้ว ที่เราจะ casting ไปเป็นอาเรย์ของ char (pointer ที่ใช้เป็นของ char) เพราะว่าข้อมูลที่เป็น char ก็มีขนาด 8 บิตด้วยเช่นกัน

เมื่อเรา casting รูปภาพให้อยู่ในรูปของอาเรย์ของ char ได้แล้วสิ่งที่เราต้องทำต่อไปคือเราต้องเก็บข้อมูลของภาพให้มองเป็น pixel เนื่องจากว่าเมื่อเรามองข้อมูลภาพให้เป็น pixel นั้นจะทำให้เราสามารถ process

ภาพได้ง่ายกว่าการมองภาพให้เห็นทีละ byte ทีละ byte ไปและเนื่องจากว่าภาพขนาด 24 บิตนั้นเอง ดังนั้น สิ่งต่อไปที่เราทำก็คือทำการ group อาเรย์ของ char ข้างต้นเพื่อให้มองเป็น pixel ให้ได้โดย group ทีละ 3 byte (3 byte = 24 bit) การ group ที่ว่านี้ก็เหมือนกับการ casting โดยการ casting แบบนี้เราจะ cast โดยการใช้ struct แทนโดยใน struct นี้ประกอบไปด้วย component ของ blue green และ red ตามลำดับ เมื่อทำการ cast มาแล้วข้อมูลที่เป็นอาเรย์ของ char ก็จะถูก match เข้าโดยตรงและถูกต้องกับแต่ละ component ได้ ในส่วนของ struct นี้เราตั้งชื่อมันว่า F_PIXEL โดยที่รูปร่างหน้าตาเป็นดังต่อไปนี้

```
typedef struct
{
        BYTE   blue;
        BYTE   green;
        BYTE   red;
} F_PIXEL;
```

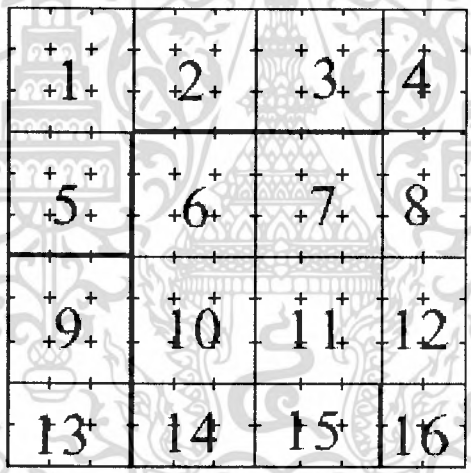
ลองสังเกตดูดี ๆ จากโครงสร้างที่อยู่ข้างบนนี้ จะพบว่าเราจะเอา blue มาก่อน แล้วตามด้วย green กับ red ตามลำดับ ที่เราต้องเอา blue มาก่อนก็เนื่องจากว่า ข้อมูลภาพที่เก็บอยู่ในหน่วยความจำนั้น มันจะเรียงเป็นดังนี้คือ bgrbgrbgr และจะเรียงแบบนี้ไปเรื่อย ๆ ดังนั้นเพื่อให้ component ของ blue , green และ red มันmatch กันได้อย่างถูกต้องกับโครงสร้างที่เราสร้างขึ้นมา เราจึงจำเป็นจะต้องเรียงเฉกเช่นเดียวกันมาข้างต้นด้วย

3.2 การ threshold ใน project นี้

สำหรับใน project นี้ เราจะใช้การ threshold แบบที่เป็น adaptive เนื่องจากว่า การทำ adaptive threshold นั้นจะให้ผลของการ threshold ที่ดีกว่า global threshold เพราะภาพที่ได้จาก scanner นั้นเมื่อทำเป็น grayscale แล้วเราจะพบว่าในหลาย ๆ บริเวณของภาพ อาจจะมีความแตกต่างของค่า grayscale ระหว่าง foreground กับ background ไม่เท่ากันก็ได้ ดังนั้นทางที่ดีจึงควรใช้การ threshold แบบ adaptive

เราจะใช้ adaptive threshold โดยการอนุญาตให้ผู้ใช้สามารถที่จะทำการกำหนดขนาดของ window เองได้ ทั้งนี้เพื่อให้ผู้ใช้ได้เห็นความแตกต่างของการกำหนดขนาดของ window ที่มีขนาดต่าง ๆ กัน ในที่นี้ขนาดของ window ที่ผู้ใช้สามารถ กำหนดได้นั้นจะอนุญาติให้กำหนดได้เฉพาะเลขคี่เท่านั้น เป็น window ที่เป็นสี่เหลี่ยมจตุรัสขนาด n x n (โดยที่ n เป็นเลขคี่) ที่ต้องทำเช่นนี้ก็เพื่อว่าเราจะสามารถหา pixel ที่เป็นจุดกึ่งกลางได้ง่ายกว่าการกำหนด window ให้เป็นแบบอื่น เมื่อเรากำหนด window ที่จะใช้ได้แล้ว ขั้นตอนต่อไปที่ทำก็คือ เอา window ที่ว่านี้ไปวางทาบในภาพ โดยให้ window นี้อยู่ในภาพไม่เกินออกมานอกขอบของภาพด้วย เมื่อเราเอา window นี้ไปวาง ไว้ในบริเวณใดบริเวณหนึ่งของภาพแล้ว (ดูรูปข้างล่างประกอบด้วย) เราก็จะเอาค่า graylevel ของทุก ๆ pixel ที่อยู่ภายในขอบเขตของ window มาทำการบวกรวมกัน แล้วก็หารด้วยจำนวนของช่องทั้งหมดของ window ก็จะได้ค่าเฉลี่ยของค่า graylevel ทั้งหมดที่อยู่

ภายในบริเวณที่ window นั้นไปวางทาบ และเราก็จะเก็บค่าเฉลี่ยที่ได้นี้เพื่อเอาไว้สำหรับเป็นค่า threshold สำหรับ pixelทั้งหมดที่อยู่ภายใน บริเวณนั้นด้วย เมื่อเราทำการหาค่าของ threshold ของบริเวณนี้เสร็จแล้ว ต่อไปเราก็จะทำการเลื่อน window ที่ว่านี้ไปในบริเวณถัดไปโดยบริเวณถัดไปที่เราจะเลื่อน window ไปนั้น จะต้องเป็นบริเวณที่ยังไม่เคยถูกกำหนดค่า threshold มาก่อน พูดง่าย ๆ ก็คือเลื่อน window ไปโดยไม่ให้ window เดิมกับ window ใหม่ทับกันได้ จากนั้นเราก็จะเลื่อน window ไปเรื่อย ๆ เพื่อคำนวณค่าเฉลี่ยของ graylevel (threshold) จากซ้ายไปขวาและจากบนลงล่าง ไปจนกระทั่ง pixel ทุก ๆ pixel ในภาพได้ถูกกำหนดค่าที่จะเอาไว้ threshold เรียบร้อยแล้ว(ในที่นี้เก็บค่า threshold ไว้เป็นอาเรย์ขนาด 2 มิติ ที่มีขนาดเท่ากับ ความกว้าง x ความยาวของภาพที่รับเข้ามา) ต่อไปที่เราทำก็คือเราจะ apply ค่าของ threshold ที่อยู่ในอาเรย์นี้ให้กับค่าของข้อมูลภาพแต่ละ pixel ไปเลย ถ้าหากว่าค่า graylevel ของ pixel นั้น ๆ มีขนาดที่มากกว่าค่า threshold ที่ตรงกับ pixel ของมันก็จะถูกกำหนดให้ pixel นั้นเป็นสีขาว หากค่าของ graylevel ของ pixel นั้น ๆ มีขนาดที่น้อยกว่าค่า threshold สำหรับ pixel นั้น ก็จะถูกกำหนดให้ pixel นั้นเป็นสีดำ เมื่อเรา apply ค่า threshold กับทุก ๆ pixel ในภาพเราก็จะได้ภาพที่เป็นแบบ binary (ภาพที่มี 2 สี)



รูปที่ 3.1 ลำดับของ window ที่จะถูกใช้สำหรับ adaptive threshold

3.3 การ segmentation หรือการ floodfill

ในส่วนของการ flood fill นอกจากว่าเราจะต้องระมัดระวังในเรื่องของ stack โดยหลีกเลี่ยงการเรียกใช้ฟังก์ชันแบบ recursive แล้ว ยังมีอีกสิ่งหนึ่งที่เราต้องระวังนั่นก็คือ ขอบของภาพ เมื่อมีจุด สีขาวซึ่งเป็นจุดที่เราสนใจไปอยู่ในบริเวณขอบของภาพเลย ก็จะเกิดปัญหาคือ จุดสีขาวที่อยู่บน ขอบภาพนั้นจะไม่สามารถหาเพื่อนบ้านรอบทิศทั้ง 8 ด้านของมันได้ครบ ถ้าหากว่าเราไม่มีการป้องกัน ที่ดีก็อาจจะทำให้เกิดปัญหาของการอ้างตำแหน่งใน memory ที่ผิดพลาดได้ (เพราะอาจจะเกิดการอ้างอิง ไปในบริเวณของจุดที่ไม่มีอยู่ในภาพจริงก็ได้) เพื่อที่จะแก้ปัญหานี้ สิ่งที่เราทำก็คือ เราก็จะทำการกำหนด

ขอบของภาพ ให้เป็นบริเวณที่จะทำการ flood fill ไม่ได้ เราทำเช่นนี้โดยการ fill ขอบของภาพ ทุกๆ ด้าน 4 ด้านให้เป็นสีดำ หรือสีของ background ก็จะสามารถที่จะแก้ปัญหาในลักษณะนี้ได้

โครงงานนี้จะใช้ stack เข้ามาช่วยในการ flood fill ด้วย โดยเราจะมอง stack เหมือนกับเป็นอาเรย์ ขนาด 1 มิติ และเราก็จะสร้าง image map ของภาพขึ้นมาด้วย image map ในที่นี้ เอาไว้สำหรับ เป็นตัว mark เพื่อบ่งบอกถึงสถานะของจุด ว่า ได้ผ่านการ flood fill มาแล้วหรือยัง ถ้าจุดนั้นไม่ ถูกทำการ flood fill เลยก็จะกำหนดให้เป็น -1 ก่อน และหากว่าจุดนั้นผ่านการ flood fill มาแล้วเราก็จะ กำหนดให้มีค่าเป็น 0 เพื่อที่จะได้ไม่ต้องมา flood fill ซ้ำที่จุดเดิมอีกรอบ สำหรับ psuedo code ของการ flood fill โดยใช้ stack นั้นให้ดูได้จากบทที่ 2 ซึ่งได้กล่าวถึงการ flood fill มาบ้างแล้ว

เมื่อเราทำการ flood fill ไปนั้น ขณะที่เราทำการ flood fill เราก็จะทำการหาขอบเขตของ blob ไปด้วยในระหว่างที่ทำการ flood fill โดยขอบเขตที่ว่านี้ก็จะได้แก่ xmin , ymin และ xmax , ymax (นั่นคือดูว่าใน blob นั้น จุดที่อยู่ในแกน x และ แกน y ที่น้อยที่สุดมีค่าเป็นเท่าไร และจุดที่อยู่ในแกน x และแกน y ที่มากที่สุดมีค่าเป็นเท่าไร) เนื่องจากว่ามีจุดอยู่ทั้งสิ้น 2 จุด ทำให้เราเลยสามารถกำหนดบริเวณขอบ เขตของ blob ได้เป็นสี่เหลี่ยม บริเวณที่ตั้งและขอบเขตของ blob แต่ละ blob ที่หาได้นี้ถือว่ามีความสำคัญ เพราะว่าเราจะใช้ค่าต่าง ๆ เหล่านี้ เพื่อใช้ในการ extract blob (หรือการแยก blob ออกมาจากภาพอีกที) ที่เราต้องทำการหาตำแหน่งขอบเขตของ blob ก่อนแล้วจึงค่อยมาทำการแยก blob ออกทีหลัง ที่เป็นเช่นนี้ เมื่อเราทำการหาขอบเขตของ blob ไปพร้อมกับการแยก blob อาจจะเกิดข้อผิดพลาดขึ้นได้ เช่นลองดูตัวอย่างข้างล่างนี้

รูปที่ 3.2 ตัวอักษรที่มีการ intersect กัน

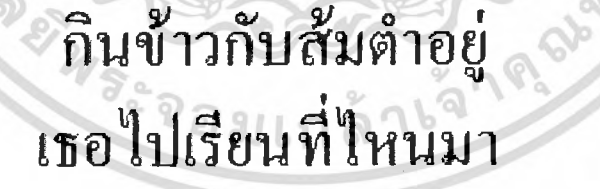
จากรูปข้างบนเมื่อเราหาขอบเขตของ blob ทั้งสองตัว (สระ อิ กับ ฟ) เราจะพบว่าบริเวณของ blob สองตัวนี้มีการ intersect กัน ซึ่งถ้าหากว่าเป็นเรามอง เวลาที่ทำการแยก blob เราก็จะเอา สระ อิ ออกมาก่อน แล้วค่อยตามด้วย ฟ แต่ในทางคอมพิวเตอร์นั้นไม่ใช่ เพราะเมื่อเราทำการ process ภาพ เพื่อหา blob นั้นมันก็จะเจอ blob ของตัว ฟ ก่อนเนื่องจากว่า ฟ นั้นมันมีขอบบนที่อยู่สูงกว่า สระ อิ นั่น เอง ดังนั้น ฟ จึงเป็น blob ตัวแรกเลย เมื่อเราหาขอบบนของ ฟ ได้แล้ว และทำการ extract blob ของ ตัว ฟ ออกมาจากรูปภาพ เราก็จะได้บางส่วนของ สระ อิ ติดมากับ blob ของ ฟ ด้วย ซึ่งก็เป็นสิ่ง ที่เรา ไม่ต้องการเลย (บริเวณของ ฟ นั้นไป intersect กับบริเวณของสระ อิ) ดังนั้นเราจึงต้องเขียนอัลกอ ริธึม เพื่อจะต้องจัดลำดับก่อนหลังของ blob ที่จะถูก extract ออกมาจากภาพหลักอีกที เพื่อไม่ให้เกิดปัญหาดังเช่นข้างต้นขึ้นมา

3.4 กระบวนการทางด้านการหาจำนวนของบรรทัดและระดับของตัวอักขระ

ก่อนอื่นจะพูดถึง วิธีการที่จะหาระดับหลักให้ได้ก่อน ซึ่ง ระดับหลักนี้จำเป็นในการที่จะสามารถเอาไว้ตัดสินได้ว่าข้อความที่เรารับเข้ามาประมวลผลนั้น มีอยู่ทั้งสิ้นกี่บรรทัด ระดับหลักในที่นี้มีความหมายถึงระดับที่วางพยัญชนะ ซึ่งตามปกติแล้ว ข้อความที่เรารับเข้ามาจะต้องประกอบ ด้วยพยัญชนะก่อน ถึงจะ มีสระ หรือว่า วรรณยุกต์ตามมาได้ สำหรับวิธีการที่จะหาระดับหลัก ซึ่งเป็นระดับที่วางพยัญชนะนั้น สามารถทำได้ดังนี้คือ

สืบเนื่องมาจากการทำอิสโตแกรมในแต่ละแถวของรูปนั้น ทำให้เราสามารถที่ จะตัดสินหาช่วงที่มีความกว้างตามแกนตั้ง(y) ที่มากที่สุดได้ ซึ่งตำแหน่งล่างสุด ของช่วงที่กว้างที่สุดนี้ จะถือว่าเป็นระดับหลัก เนื่องมาจากว่า ช่วงที่กว้างที่สุดนั้น เป็นช่วงที่เอาไว้สำหรับวางพยัญชนะ ซึ่งโดยปกติแล้ว ข้อความในบรรทัดใด ๆ ก็จะต้องประกอบด้วยพยัญชนะอยู่แล้ว และพยัญชนะโดยส่วนใหญ่ก็จะมีความสูงมากกว่า สระ และ วรรณยุกต์ ยกเว้นแต่ สระบางตัวอย่างเช่น สระ ไ ใ เป็นต้น ซึ่งจะมีความสูง เท่ากับพยัญชนะเลย ซึ่งได้แก้ ตัว ป ฟ

นอกจากการหาระดับหลักจะใช้การเปรียบเทียบ ความกว้างในแต่ละชั้นแล้ว ยังเอาระยะห่างของแต่ละชั้นมาทำการคิดด้วย คือว่าระยะห่างระหว่างระดับต่าง ๆ ที่อยู่ในบรรทัดเดียวกันนั้น จะมีค่าที่ต่อยห่างระดับที่ห่างกัน และอยู่คนละบรรทัด การทำเช่นนี้ได้ เราจะต้องกำหนดค่า threshold ขึ้นมาไว้ค่าหนึ่งก่อน ซึ่งค่า threshold นี้จะเอาไว้เพื่อเป็นตัวบ่งบอกได้ว่า ระดับใด ๆ ก็ตามที่ได้มาจากการทำ histogram นั้นควรจะอยู่ในบรรทัดเดียวกันหรือไม่การหาค่า threshold นั้นเราสามารถทำได้ โดยดูจากตัวอย่างดังนี้คือ ถ้าหากว่าข้อความที่รับเข้ามาหลายบรรทัดเป็นดังนี้



รูปที่ 3.3 ตัวอย่างของข้อความที่มีหลายบรรทัด

การกำหนดค่า threshold จะใช้ระยะห่างระหว่างสระ อิ กับ ตัว ก เป็นสำคัญและเราจะหาดูว่า ถ้าหากระหว่างระดับไหนที่มีความห่างมากกว่า threshold นี้ละก็ เราก็จะถือว่า ระดับสองระดับนั้นอยู่คนละบรรทัดกัน นอกจากนี้ถ้าเราสังเกตดี ๆ เราจะพบอีก ข้อจำกัดหนึ่ง ที่เป็นตัวช่วยในการตัดสินว่า ระดับไหนจะอยู่คนละบรรทัด นั่นคือ การดูจากลำดับความกว้างของระดับใด ๆ จากตัวอย่างข้างต้นเราจะพบว่าถ้ามองตามความกว้างของระดับใด ๆ ระดับแรกสุดที่เจอเลยคือ ระดับของ สระ อิ ไม้โท ไม้หันอากาศ เป็น ต้น ระดับถัดมาก็

คือระดับของพยัญชนะ ในที่นี้ก็เริ่มตั้งแต่ ก , น เป็นต้น ซึ่งมีค่าของความกว้างมากกว่า ระดับแรกแน่นอนอยู่แล้ว ระดับที่ 3 คือระดับของสระ อู ซึ่งพบว่ามีความกว้างที่น้อยกว่าระดับที่ 2 ดังนั้นเราจึงสามารถตั้งข้อกำหนดได้ว่า หากความกว้างของระดับลดลงจากเดิม ลงเหลืออยู่ประมาณ ครึ่งหนึ่งของความกว้างครั้งก่อน และมีระยะห่างจากระดับครั้งก่อนอยู่ในค่าที่เหมาะสม แสดงว่าระดับนี้ เป็น ระดับล่างสุดของบรรทัดนี้ แต่วิธีการนี้ก็มีข้อจำกัด เช่นกัน อย่างเช่นลองดูตัวอย่างต่อไปนี้

กกกกกกกกกกก

กกกกกกกกกกก

กกกกกกกกกกก

รูปที่ 3.4 รูปตัวอย่างที่ในแต่ละบรรทัดนั้นมีความสูงของบรรทัดเท่ากัน

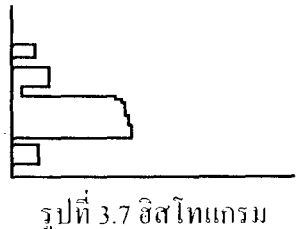
จากรูปข้างบนเราจะพบว่า ความกว้างของแต่ละระดับนั้นมีค่าที่เท่ากันและระยะห่างของแต่ละระ ดับก็มีค่าที่เท่ากันด้วยเช่นกัน ดังนั้นในกรณีนี้ค่า threshold ก็จะมีค่าที่เท่ากับ ระยะห่างของแต่ละระดับหักลบออกด้วยค่า ๆ หนึ่ง ที่น้อย ๆ เพื่อที่จะทำให้ใช้ค่านี้เพื่อเป็นค่า threshold ได้อย่างมีความถูกต้องเหมาะสม ต่อไปจะกล่าวถึงการพยายามที่จะหาระดับต่าง ๆ ในแต่ละบรรทัด

1. ก่อนอื่นเราจะทำการหา histogram สำหรับแต่ละแถวก่อน สำหรับกระบวนการ row histogram นั้นเราจะสามารถทำได้โดยการ ไล่จากแถวที่ 1 ไปจนถึงแถวสุดท้าย ซึ่งก็จะมีค่าเท่ากับความสูงของภาพ ในแต่ละแถวของภาพนั้น ให้เราทำการ scan เพื่อหาจุดของ pixel ที่เป็นสีขาว แล้วนับจำนวนจุดสีขาวนั้นที่อยู่ในแต่ละแถว จากนั้นก็เก็บในอาเรย์ไว้ กระบวนการนี้จะสามารถทำให้เราสามารถที่จะตัดสินว่าในรูปของเรานั้นน่าจะประกอบไปด้วยข้อความประมาณกี่บรรทัดได้

ปูมีกี่ขา

รูปที่ 3.5 รูปตั้งต้น

รูปที่ 3.6 รูปตั้งต้นที่ผ่านการ threshold



รูปที่ 3.7 ฮิสโทแกรม

2. เมื่อเราเสร็จสิ้นกระบวนการของ row histogram แล้ว สิ่งที่เราทำต่อไปก็คือ การหาความกว้างของช่วงต่าง ๆ ที่ปรากฏอยู่ใน histogram เช่น จากรูปตัวอย่างที่ปรากฏอยู่ข้างบนเราก็จะได้ row histogram ดังรูปด้านขวา นอกจากว่าเราจะหาความกว้างของช่วงที่ปรากฏใน histogram แล้ว เรายังหาระดับที่เป็นฐานของช่วงต่าง ๆ ด้วย (base line) เราจำเป็นที่จะต้องหา base line ในแต่ละช่วงของ histogram เพื่อที่จะเอาไว้สำหรับอ้างอิงถึงระดับต่าง ๆ ในแต่ละบรรทัด ทั้งนี้ก็เนื่องมาจากว่า ในภาษาไทยนั้น ในแต่ละบรรทัดนอกจากจะประกอบไปด้วยพยัญชนะแล้ว ยังประกอบไปด้วยสระ และ วรรณยุกต์ด้วย อย่างเช่นคำว่า "ปูมีกี่ขา " จากข้อความนี้ เราเลยสามารถที่จะจัดระดับในแต่ละบรรทัดของภาษาไทยได้เป็น 4 ระดับ ดังรูปด้านล่างนี้ กำหนดให้ สระ อู อยู่ในระดับล่างสุดหรือระดับที่ 1 ป อยู่ในระดับที่ 2 สระ อี อยู่ในระดับที่ 3 ไม้เอก อยู่ในระดับที่ 4 และเราทำการเก็บความกว้าง และระดับฐานของแต่ละช่วงไว้ใน อาเรย์ด้วย

รูปที่ 3.8 เส้นตรงในแนวนอนแสดงระดับต่าง ๆ ในบรรทัด

3. แต่เนื่องจากกระบวนการข้างต้น ยังไม่เพียงพอที่จะหาระดับได้อย่างถูกต้อง ในแต่ละบรรทัด เนื่องจากเมื่อเราสังเกตให้ดีในรูปข้างบนเราจะพบว่าช่วงของ histogram ช่วงที่ 2 นั้น ซึ่งมี ป เป็นตัวที่กำหนดความกว้างของช่วงนั้น แต่ว่ามีสระ อี ซึ่งในความเป็นจริงจะต้องถูกกำหนดให้เป็นอีกระดับหนึ่งด้วย เพราะว่ามันเป็นสระ แต่เนื่องจากว่า ความสูงของตัว ป นั้นไปทับกับระดับของสระ อี ด้วย ซึ่งทำให้ได้ผลลัพธ์ที่ไม่ถูกนัก ดังนั้นทำให้เราต้องหา mechanism บางอย่างที่จะสามารถทำให้บอกได้ว่า ป กับ สระ อี นั้นมันอยู่คนละระดับกัน ถึงแม้ว่าทั้งสองตัวนี้จะอยู่ในช่วงของ histogram เดียวกัน

3.5 ปัญหาอันเนื่องมาจากกระบวนการ segmentation (flood fill)

ปัญหาที่เห็นได้ชัดคือ ปัญหาการ overlapped กันของ blob ตัวอย่าเช่น ข้อความที่ว่า "ปีนี้ปีไก่" จากการที่เราสังเกตข้อความนี้ เราจะพบว่ามีการซ้อนทับกันของ blob เกิดขึ้น อย่างเช่นคำว่า 'ปี' เราพบว่าบริเวณ blob ของ สระอี กับ blob ของ ป นั้นมันทับกัน ดังนั้นเราจะต้องมี วิธีการ ในการแยก blob ของ สระอี กับ ป ออกจากกัน ซึ่งในที่นี้เราจะต้องแยก สระอี ออกมาก่อน เพราะว่ามันเป็นตัวที่ไปทับ ป จากนั้นก็จะสามารถแยก blob ป ออกมาได้ นอกจาก blob อาจจะซ้อนทับกันแบบโดยตรงๆซึ่งสามารถสังเกตได้อย่างชัดเจนแล้ว blob ยังสามารถซ้อนทับกันโดยทางอ้อมก็ได้ ในกรณีนี้จะหมายถึงว่า ระดับของ blob ตัวนั้นๆไปทับกับระดับของ blob ตัวอื่นๆ ทำให้ blob สองตัวนั้นมันมีระดับที่เหมือนกัน ทั้งๆที่เมื่อพิจารณาจากความเป็นจริง blob สองตัวนี้จะต้องมีระดับที่แตกต่างกัน ตัวอย่างเช่น จากข้อความที่ว่า "ฟางอยู่ในทุ่งนา" จากข้อความนี้เราจะสังเกตได้เลยว่า blob แต่ละ blob ไม่มีการซ้อนทับกันโดยตรง แต่มีการซ้อนทับกันโดนอ้อม อย่างเช่น

ตัว ฟ กับ ไม้เอก จะถูกจัดให้อยู่ในระดับเดียวกันเพราะขอบเขตของ blob ของทั้งสองตัวนั้นซ้อนทับกันแกน y ดังนั้นเราจึงต้องมี วิธีการ เพื่อที่จะบอกให้ได้ว่า ความจริงแล้ว ฟ กับ ไม้เอก อยู่ในระดับที่ต่างกัน

ต่อไปจะกล่าวถึง วิธีการ ที่ใช้จัดการกับ blob ที่มีการซ้อนทับกันเกิดขึ้นไม่ว่าจะโดยทางตรงหรือทางอ้อมก็ตาม

<u>การซ้อนทับกันโดยทางตรงจะสามารถจัดการได้ดังนี้</u>

รูปที่ 3.9 ตัวอย่างของการทับกันโดยตรง

ในการที่จะตัดสินว่า blob ใดๆมีการ intersect เกิดขึ้นหรือไม่นั้นเราจะใช้วิธีการทำโดยตรวจสอบดูว่าขอบเขตบริเวณที่ blob ตัวนั้นๆอยู่ มีการไปทับกับขอบเขตบริเวณของ blob ตัวอื่นๆหรือไม่ วิธีการหนึ่งที่ทำได้โดยการสร้าง array ขนาด 2 มิติขึ้นมา จากนั้นก็เติม array 2 มิตินั้นด้วยหมายเลข blob ของตัวอักษร ถ้าหากว่า บริเวณในภาพนั้นไม่มี blob ตั้งอยู่ ให้กำหนดค่าเป็น -1 ไว้ แต่ถ้าหากว่าเป็นบริเวณที่มี blob อยู่ ให้กำหนดค่าใน array 2 มิตินั้นให้เป็นหมายเลขของ blob ด้วย

เราจะพบว่าในขอบเขต blob ของตัวอักษร ป นั้น นอกจากจะมีเลข blob ของตัว ป แล้ว ยังมีเลข blob ของตัว สระอิ อยู่ภายในบริเวณนั้นด้วย ทำให้บ่งชี้ได้ว่ามีการ intersect กันเกิดขึ้นแน่นอน และเราก็จะทำการเก็บข้อมูลไว้ด้วยว่า blob ตัวไหนมีการ intersect กับ blob ตัวไหนอีกด้วย

ต่อไป เราก็จะเอา blob ตัวที่มันไปทับกับตัวอื่น ๆ มาดูว่ามันไปทับกับตัวไหน สังเกตได้ว่าการทับกันของ blob ข้างต้นนั้น ในความเป็นจริงแล้วจะต้องทำให้เกิดระดับใหม่ขึ้นมาด้วย เนื่องจากว่าตัวสระ อิ กับตัว ป นั้น มันควรที่จะอยู่คนละระดับกัน เราสามารถที่จะหาระดับใหม่ได้ โดยการเปรียบเทียบขนาดของ blob 2 อันใด ๆ ที่มัน intersect กันเราจะพบว่า blob ที่มีขนาดที่น้อยกว่านั้น จะทำให้เกิดระดับใหม่ขึ้นมา นอกจากจะใช้ในการตัดสินระดับได้แล้ว การหาขนาดของ blob ที่ intersect กัน 2 อัน ก็สามารถที่จะช่วยในการที่จะแยก blob ของตัว ป กับ สระ อิ มาได้โดยมีลำดับที่ถูกต้อง ลองคิดดูว่าคำว่า "ปี" หากว่าเราเอาตัว ป ออกมาก่อนก็จะทำให้ blob ที่แยกออกมามีบางส่วนของสระ อิ ติดมาด้วยซึ่งทำให้เกิดความผิดพลาดได้ ซึ่งการแยกที่ถูกต้องนั้นจะต้องเอาสระ อิ ออกมาก่อนถึงค่อยเอา ป ออกมา หรือหมายความได้อีกนัยหนึ่งก็คือ เราจะแยกเอา blob ที่มีพื้นที่น้อยกว่าออกมาก่อน blob ที่มีพื้นที่มากกว่า แต่การทำแบบนี้ก็ไม่แน่ว่าจะถูกต้องเสมอไปหรอก

การซ้อนทับกันโดยทางอ้อม จะสามารถที่จะจัดการแก้ไขได้ดังนี้

เมื่อเกิดปัญหาการซ้อนทับกันโดยทางอ้อม ดังที่เคยกล่าวไว้แล้วใน paragraph ก่อน ๆ เพื่อที่จะแก้ปัญหานี้ ผู้พัฒนาได้ใช้วิธีการดังนี้ คือ

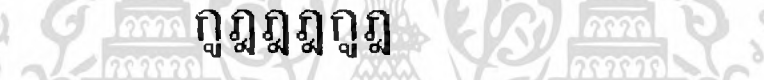
1. จะต้องหาระดับหลักของแต่ละบรรทัดให้ได้ก่อน ระดับหลักก็คือระดับที่เอาไว้สำหรับวางพยัญชนะ

2. เมื่อทำการหาระดับหลักได้แล้ว ก็จะต้องดูว่า ในระดับหลัก หรือ ระดับที่เอาไว้สำหรับวางพยัญชนะนี้ มีความกว้างเป็นเท่าไรด้วย เพื่อที่จะเอาไว้กำหนดขอบเขตบนของมันได้

3. เมื่อเราทำการหาความกว้างของระดับหลักได้แล้ว เราก็จะดูว่ามี blob ตัวไหนที่ระดับล่างของมันอยู่ระหว่างขอบล่างของระดับหลัก แต่ไม่เกินขอบบนสุดของระดับหลัก และ blob ตัวนี้จะต้องมีระดับล่างสุดไม่เท่ากับระดับล่างสุดของระดับหลักด้วย

4. ถ้าหากว่ามี blob ดังที่กล่าวถึงในข้อที่ 3 นั้น เราก็จะเอาระดับล่างของ blob ตัวนั้นมาจัดให้เป็นระดับใหม่ได้อีกระดับหนึ่ง

การ intersect กันโดยทางอ้อมอีกรูปแบบหนึ่งที่เจอ จะมีลักษณะดังรูปด้านล่างนี้



รูปที่ 3.10 การซ้อนทับกันโดยทางอ้อมกับระดับ 0

เมื่อเราสังเกตจากรูป เราจะพบว่า ฎ นั้นมีการ intersect กันโดยทางอ้อมกับสระ อู สำหรับ การแก้ปัญหานี้ เราจะไม่สามาระแก้โดยใช้วิธีการเดียวกับที่กล่าวมาข้างต้นได้ซึ่งวิธีการข้างต้นนั้นเอาไว้ สำหรับแก้ปัญหาเมื่อ blob ตัวที่มัน intersect กันโดยทางอ้อมนั้น อยู่สูงกว่าขอบล่างสุดของระดับหลักแค่ นั้น ดังนั้นเราจำเป็นต้องหาวิธีการอื่น

ถ้าหากว่าเราใช้ algorithm ดังที่ได้กล่าวมาข้างต้นกับภาพที่อยู่ด้านบนนั้น จะทำให้เราไม่สามารถที่จะแยกระหว่างระดับหลัก กับระดับของสระ อู ได้ เพราะว่ามันทับกัน (ซึ่งในความเป็นจริงระดับหลักจะถูกกำหนดให้เป็นระดับที่ 1 และ สระ อู นั้นจะถูกกำหนดให้เป็นระดับที่ 0)

วิธีการแก้เราจะใช้ วิธีการหา blob ของตัวที่มีลักษณะเป็นสระ อุ หรือ สระ อู กล่าวคือ เราจะใช้ความสูงของ blob เข้ามาเกี่ยวข้องด้วย ถ้าหากว่า blob ตัวนั้น มีขอบล่าง อยู่บนขอบล่างของระดับหลัก และก็มีความสูงของ blob น้อยกว่า ครึ่งหนึ่งของความกว้างของระดับหลักละก็ แสดงว่าจะต้องมีสระประเภท สระ อุ สระ อู อยู่ด้วย ถ้าหากว่า ยังไม่มีระดับที่ 0 ก็ให้เรากำหนดให้มีระดับที่ 0 เพิ่มเข้ามาด้วย

นอกจากนี้ปัญหาอีกปัญหาหนึ่ง ที่เป็นที่น่าปวดหัวสำหรับเราเลยก็คือ ปัญหาของสระ อะ นั่น เอง เนื่องจากว่าสระ อะ นั้น ความเป็นจริงควรที่จะถูกจัดให้อยู่ในระดับเดียวกันกับพยัญชนะ แต่จากลักษณะ

รวมทั้งตำแหน่งที่มันอยู่จึงทำให้หลายครั้งมีความผิดพลาดเกิดขึ้นจากกระบวนการ segmentation โดยเฉพาะอย่างยิ่งมีส่วนทำให้ ตำแหน่งของระดับต่าง ๆ ผิดเพี้ยนไปลอง สังเกตดูจากตัวอย่าง

ครูกินบะหมี่อยู่

รูปที่ 3.11 บรรทัดที่มีสระอะปนอยู่ด้วย ทำให้ได้ระดับทั้งหมดเพิ่มเป็น 5 ระดับ

จากข้อความข้างต้น เราจะสามารถหาระดับได้ถึง 5 ระดับ ซึ่งมันไม่ถูกต้องสำหรับข้อความภาษาไทย ที่จะมีอย่างมากได้แค่ 4 ระดับเท่านั้น ดังนั้นเราจำเป็นต้องมีวิธีการในการที่จะกำจัดสระ อะ ออกไปเพื่อไม่ให้มันถูกจัดไว้ว่าเป็นอีกระดับหนึ่งในบรรทัด หรือทำให้รู้ว่าสระ อะนั้นอยู่ในระดับเดียวกันกับระดับของพยัญชนะเป็นต้น ซึ่งในวิธีการที่คิดขึ้นไว้สำหรับจัดการกับสระ อะ นี้ เป็นดังนี้

เนื่องจากว่าสระ อะ นั้นประกอบไปด้วย blob จำนวน 2 อัน และ blob ตัวที่อยู่ด้านล่างนั้น มันก็จะติดกับระดับหลักในแต่ละบรรทัดนั้น ๆ ด้วย เมื่อเราจะทำการ check สระ อะ เราจำเป็นที่จะต้องทราบ ความกว้างของบรรทัดนั้น ๆ ก่อน จากนั้น เราก็จะทำการสุ่ม blob ของตัวอักษร ที่อยู่ในบรรทัดนั้น ๆ ขึ้นมา โดยที่ blob ที่สุ่มขึ้นมาจะต้องมีขอบล่าง เป็นค่าเดียวกับขอบล่างของระดับหลักด้วย จากนั้นเราก็จะทำการหาอัตราส่วนของความกว้างของระดับหลัก ต่อ ความกว้างของตัวอักษรหนึ่งตัวในระดับหลัก การทำเช่นนี้มีข้อดีอีกอย่างคือว่า จะสามารถทำให้รู้ได้ว่า ในบรรทัดนั้น ๆ มีตัวอักษรที่พิเศษประกอบอยู่ด้วยหรือไม่ (อักษรพิเศษคือพยัญชนะที่สูงผิดปกติอย่างเช่น 'ป' 'ฟ' 'ฝ' เป็นต้น) อัตราส่วนของความสูงของตัวอักษรที่พิเศษ ต่อความกว้างของมัน จะมีค่าที่มากกว่า อัตราส่วนนี้ของตัวอักษรแบบธรรมดา (ตัวอักษรแบบธรรมดาอย่างเช่น 'ก' 'ข' 'บ' 'อ' เป็นต้น) เพราะเราสังเกตได้ว่าการมีสระ อะ นั้น เนื่องจากถ้ามองกันดี ๆ แล้วจะพบว่ามันก็เป็นปัญหาของการซ้อนทับกันโดยทางอ้อมอีก แต่คราวนี้สระ อะ ไม่ควรจะถูกจัดให้เป็น ระดับใหม่ในบรรทัดนั้น ๆ การซ้อนทับกันโดยทางอ้อมนั้นส่วนใหญ่เกิดขึ้นกับตัวอักษรแบบพิเศษ เพราะว่าตัวอักษรแบบธรรมดานั้นเมื่อเกิดการซ้อนทับกันโดยทาง อ้อมก็ไม่เกิดปัญหาอะไรขึ้นมาเลย ลองสังเกตดูจากรูปตัวอย่างด้านล่าง

กระบี่มังกร

รูปที่ 3.12 รูปนี้สระอะจะไม่มีผลกับกระบวนการหาระดับ

ถ้าเป็นเช่นรูปข้างบน ก็ไม่จำเป็นจะต้องเอาไปเข้าอัลกอริธึม เพื่อหาการซ้อนทับกันโดยทางอ้อม เพราะว่าในกรณีนี้ สระ อะ มันก็จะอยู่ในระดับหลักของบรรทัดถูกต้องอยู่แล้ว แต่ถ้าหากว่าเป็นดังรูปข้างล่างนี้

ฝืนทำความสะอาด

รูปที่ 3.13 บรรทัดที่มีทั้งสระอะ และตัวอักษรพิเศษ

เมื่อมีตัวอักษรพิเศษปรากฏขึ้นมา ในที่นี้ได้แก่ ฝ ปรากฏขึ้นในบรรทัดใด ๆ ควรที่เราจะเอาไปเข้าอัลกอริธึมเพื่อหาการซ้อนทับกันโดยทางอ้อมด้วย เพราะว่าอาจจะมีสระ อิ อี เป็นต้น ถูกบังโดยตัว ฝ อยู่ได้ ถ้าหากว่าเรา หาระดับใหม่จากรูปข้างบน โดยไม่ได้กำหนดข้อจำกัดอะไรเลย เราก็จะได้ระดับใหม่เพิ่มขึ้นมาอีก 2 ระดับคือ ระดับของสระ อะ ตัวบน กับระดับของสระ อี แต่สระ อะ ไม่ควรที่จะเป็นระดับใหม่ แต่จากการสังเกตดู เราจะพบว่าขอบเขตล่างของ blob สระ อะ ตัวบนนั้นอยู่ต่ำกว่า ขอบเขตล่างของ blob สระ อี และ ขอบเขตล่างของ blob สระ อะ นั้น ก็จะอยู่สูงจากระดับหลักในบรรทัดนั้น ๆ น้อยกว่า 6/10 ของความกว้างของระดับหลัก ในขณะที่ ตัว สระ อี นั้น ขอบเขตล่างของมัน อยู่สูงจากระดับหลักมากกว่า 6/10 ของความกว้างของระดับหลัก โดยการใช้ข้อจำกัดดังที่ได้กล่าวมานี้นั้น จะสามารถทำให้เราสามารถที่จะกำจัดสระ อะ ไม่ให้มันไปเป็นอีกระดับหนึ่งได้

เมื่อเราสามารถที่จะทำการหาระดับต่าง ๆ ที่อยู่ในแต่ละบรรทัดได้แล้ว รวมไปถึงจำนวนของบรรทัดทั้งหมดที่อยู่ในรูปภาพที่เรารับเข้ามาเพื่อจะแปลงเป็นตัวอักษรแล้วละก็ ขั้นตอนต่อไปที่เราทำก็คือการแยกเอา blob ของตัวอักษร ทั้งหมดที่อยู่ในรูปภาพออกมาให้ได้ เพื่อที่จะเอา blob ที่ว่านี้มาดูลักษณะ และเอา blob ไปเข้าสู่กระบวนการ recognition เพื่อที่จะสามารถบ่งบอกได้ว่า blob ตัวนั้น ๆ จะหมายถึงพยัญชนะตัวใดในภาษาไทย สำหรับกระบวนการแยก blob แต่ละตัวออกมาจากภาพนี้ เราจะไม่ใช้กระบวนการของ row histogram เลย แต่เราจะอาศัยวิธีการที่เรียกกันว่า flood fill เพื่อที่จะมาหาบริเวณของเขตของ blob แทน ส่วนในเรื่องของขั้นตอนวิธีการของการ flood fill นั้น ได้กล่าวมาก่อนแล้วในช่วงต้น ๆ ของรายงานชิ้นนี้ การ flood fill สำหรับการหา blob นี้เราจะใช้การ flood fill แบบ 8 ทิศทาง เพื่อที่จะเข้าถึงจุดทุกจุดที่เชื่อมกันของ blob ได้ และเราจะ flood fill เฉพาะภาพที่ผ่านการ threshold มาแล้วเท่านั้น แต่ก็มีสิ่งที่ต้องระวังเช่นกันสำหรับการ flood fill กล่าวคือ เมื่อเราทำการ threshold ภาพที่รับเข้ามา ให้กลายเป็นภาพแบบ binary แล้ว การทำ threshold นั้นจะต้องระมัดระวัง เพื่อให้เมื่อทำการ threshold แล้ว ต้องทำให้ไม่มี blob ของตัวอักษรใด ๆ ติดกันเลย เพราะถ้าหากว่า มีตัวอักษรที่ติดกันอยู่ เมื่อเราทำการ flood fill ก็จะได้ภาพที่มีตัวอักษรอยู่ 2 ตัว ทำให้ไม่สามารถที่จะเอาไปทำการ recognition ได้ เพราะเราทำการ recognition โดยใช้ blob ของตัวอักษรเพียง 1 ตัวเท่านั้น ในการแยก blob นั้นเราก็จะใช้การ flood fill ให้กับทุก ๆ blob ที่อยู่ในภาพจน

ครบทุก blob เมื่อ แยก blob ตัวไหนออกมาได้แล้วก็จะเติม blob ที่ทำการแยกเสร็จแล้ว จากสีขาว ให้กลายไปเป็นสีดำ เพื่อที่จะเป็นการ mark ว่าไม่ต้องมา flood fill blob ตัวนี้อีกแล้ว

เมื่อเราทำการแยก blob ออกมาแล้ว เนื่องจากว่า blob ที่แยกออกมานั้น อาจจะมีขนาดที่ต่างๆกันได้ และมีขนาดที่ต่างจากข้อมูลที่เราไป train เพื่อเอาไว้สำหรับการ recognition ดังนั้นเราจึงจำเป็นที่จะต้องหา feature ที่มีลักษณะเด่นเพื่อเราไว้สำหรับเป็นตัวแทนของ blob ได้ สิ่งแรกที่เราทำก็คือ จะทำการ scale blob ของตัวอักษรให้มีขนาดที่เป็นมาตรฐาน ขนาดหนึ่งๆก่อน ซึ่งไม่ว่า blob ที่ได้จากระบวนการ flood fill จะมีขนดเป็นเท่าใดก็ตามก็จะถูก scale ให้อยู่ในขนาดมาตรฐานเท่ากันหมดเลย ในที่นี้ทางผู้จัดทำได้ใช้การ scale blob ใดๆให้มีขนาดเป็น 32x32 pixel ส่วนในเรื่องของการ scale นั้นได้เลือกใช้ algorithm ในการ scale แบบ bilinear หรือ bilinear interpolation ซึ่งก็ได้เคยกล่าวไว้แล้วในหัวข้อก่อนหน้านี้แต่ จากการ scaleนั้นเราจะพบว่า เมื่อเรา scale จากภาพ (blob) ที่มีแค่2สีคือสีขาวกับสีดำเท่านั้นให้ได้ขนาดที่ ใหญ่ขึ้นไปเป็น 32x32 pixel นั้นเราจะพบว่าเมื่อทำการ scale blob เสร็จแล้วกลับไม่ได้ภาพที่เป็น 2 สีแต่ กลายเป็นภาพแบบ grayscale แทนและก็มีจำนวนสีในแต่ละ blob ไม่คงที่แตกต่างกันไป ซึ่งลักษณะ เช่นนี้ก็เป็นข้อดีอีกอย่างหนึ่งเพราะว่า เมื่อข้อมูลของ blob มีความแตกต่างหันมากๆนั้นก็จะทำให้ กระบวนการแยกนั้นสามารถที่จะมีความถูกต้องมากกว่าเมื่อข้อมูลของ blob มีความแตกต่างกันน้อยๆ (พูด ได้ง่ายๆก็คือว่ามี feature ที่มากกว่า) เมื่อเราทำการ scale blob ไปเป็นขนาด 32x32 pixel เสร็จเรียบร้อย แล้ว เนื่องจากว่ามีข้อมูลเกี่ยวกับ blob มากเกินไป เพราะถ้าเราจะหาข้อมูลนี้ไปใช้ เราจะต้องเอา memory สำหรับแต่ละ blob เป็นขนาดถึง 1024 ซึ่งถ้ามี blobจำนวนมากที่จะต้องเอาไป train หรือเอาไป recognition นั้น ก็จะทำให้ใช้เวลาในการPROCESS นานพอสมควร ตัวนั้นเพื่อความเหมาะสม หลังจากที่ เราทำการ scale blob ไปเป็น 32x32 แล้วเราก็จะทำการ group ให้ได้เป็นขนาด 8x8 โดยการ group นั้น จะทำดังนี้ สังเกตจากรูป

รูปที่ 3.14 สี่เหลี่ยมสีแดงคือ feature 1 อันที่เราหาได้จาก blob ขนาด 32 x 32

เมื่อเราสร้าง window จัตุรัสขนาด 4x4 แล้วอา window นี้ไปใส่ไว้ใน blob ที่มีขนาดเป็น 32x32 เรา จะสามารถวาง window ขนาด 4x4 นี้ได้เต็มบริเวณเอง blob นั้นทันที และขนาดของ window ที่วางนี้ก็ จะมีค่าที่เท่ากับ 8x8 หรือ 64 สำหรับกระบวนการหา feature นั้นเราก็จะทำตามนี้นั้นเอง จากนั้นเราก็จะ เอาค่าที่อยู่ใน blob ขนาด 32x32 ที่อยู่ในแต่ละ window ซึ่งมีขนาดเท่ากับ 16 ช่องมาทำการบวกรวมกัน

ให้หมดครบทุกๆช่องในแต่ละ window จากนั้นก็จับหารด้วย 16x255 ซึ่งค่า (16 x 255) นี้ก็จะหมายถึงว่า window นั้น ๆ มีสีขาวหมดเลยเพราะว่าค่า 255 ในทาง grayscale นั้นก็หมายถึงสีขาว และก็ 16 ที่เอามาคูณกับ 255 ก็คือว่าเป็นจำนวน ช่องทั้งหมดที่อยู่ใน window สุดท้ายก็จะเอาค่าผลบวกทั้งหมดที่ได้ตอนต้น มาทำการหารด้วยค่า 16 x 255 เพื่อเป็นการทำให้ค่าของ window ขนาด 16 x 16 นั้นมีค่าที่เป็น ทศนิยม และมีค่าอยู่ในช่วง 0.0 – 1.0 ซึ่งจากการที่มีจำนวนของ window ทั้งหมด 8 x 8 ก็จะทำให้เราได้ ค่าที่เป็นทศนิยมตั้งแต่ 0.0 – 1.0 จำนวนทั้งสิ้น 64 ค่า และเราก็จะใช้ค่านี้เพื่อเป็นตัวแทนของ blob ตัวอักษรแต่ละตัว ต่อไปจากนี้เราก็จะใช้ค่าทั้ง 64 ค่าที่หาได้นี้ รวมกับอัตราส่วนของความสูงของ blob ต่อความกว้างของ blob ซึ่งก็จะได้ค่าออกมาทั้งสิ้น 65 ค่า เพื่อเอาไว้สำหรับ recognition ว่า blob ตัวนั้น ๆ เป็นตัวอักษรอะไร โดยเราจะเอาค่าทั้ง 65 ค่านี้ใส่ไว้เป็น input ของ neural network เพื่อให้ neural network ทำการวิเคราะห์ว่า input ทั้งหมดที่ใส่เข้าไปนั้นจะให้ผลลัพธ์ออกมาเป็นตัวอักษรตัวอะไร ต่อไปเราจะพูดถึงกระบวนการทางด้าน neural network ซึ่งเป็นส่วนที่สำคัญสำหรับเอาไว้ recognition ตัวอักษร

3.6 ส่วนของ Neural Network ในโปรเจ็ค OCR เป็นดังต่อไปนี้

- ส่วนของการสร้าง network
- ส่วนของการ train data
- ส่วนของการ recognition

3.6.1 การสร้าง Neural Network

โครงสร้างของ neural network แบบ multilayer-layer ที่ใช้กันโดยทั่วไปและสามารถเอาไว้แก้ ปัญหาได้เป็นจำนวนมากนั้น จะประกอบไปด้วย layer ทั้งสิ้นอยู่ 3 Layerได้แก่ input layer, hidden layer และก็ output layer ใน project นี้ก็เช่นกัน neural network ที่ใช้ใน project นี้จะมีทั้งสิ้น 3 layer เช่นกันและการเชื่อมต่อในแต่ละ layer ก็จะเป็นแบบ fully-connected คือว่าทุกๆ node ในแต่ละ layer นั้นจะเชื่อมต่อถึงกันหมดเลยในการสร้าง neural network นั้นนอกจากว่าเราจะกำหนดจำนวนของ layer ที่จะใช้แล้วยังต้องกำหนดจำนวน node ที่อยู่ในแต่ละ layer ด้วยซึ่งเราจะกำหนดจำนวน node ในแต่ละ layer เป็นดังต่อไปนี้คือ

- **input layer** จะมี node อยู่ทั้งสิ้นจำนวน 65 node ซึ่งก็จะหมายถึงว่าเราจะมี input ทั้งสิ้น65 ตัวซึ่ง input ของ neural network ทั้งหมดก็คือ ตัวเลขทศนิยมที่แทนBLOBตัวอักษรแต่ละตัวทั้ง65 ค่า

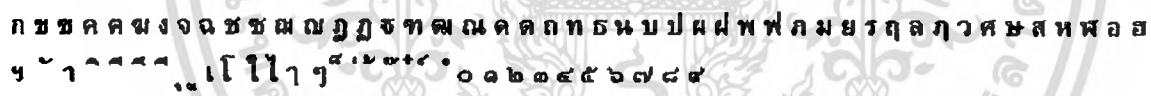
- **hidden layer** ใน hidden layer นั้นไม่มีสูตรที่แน่นอนตายตัวที่จะกำหนดจำนวนของ node ที่จะอยู่ใน layer นี้แต่ว่าโดยส่วนใหญ่แล้วมักจะกำหนดจำนวนของ hidden node ให้มีค่าอยู่ระหว่างจำ นวน node ของ input layer และจำนวน node ของ input layer และจำนวน node ของ output layer เนื่องจากว่าเรากำหนดจำนวนของ hidden node เป็นอะไรก็ได้ในproject นี้จึงกำหนดให้มีค่าเป็น65เท่ากับจำนวนของ input node พอดีเลย

- **output layer** จำนวน node ใน output layer นี้จะมีค่าเท่ากับจำนวนของตัวอักษรทั้งหมด ที่เราสามารถจะแทนได้โดยส่วนของ input layer (เท่ากับจำนวนตัวอักษรทั้งหมดที่เราสามารถที่จะ recogni- tion ออกมาได้) ใน project นี้เราจะกำหนดให้มีค่าเท่ากับ 28 เนื่องจากว่าตัวอักษรทั้งหมดที่สามารถที่จะ recognition ได้นั้นมีทั้งสิ้นอยู่ 28 ตัว

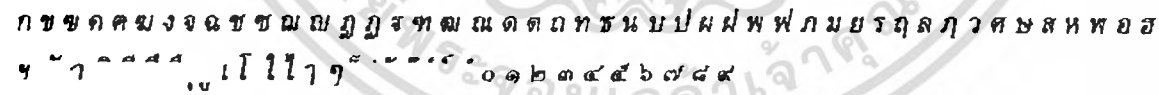
เมื่อสร้างโครงสร้างของ neural network มาแล้ว เราก็จะกำหนดค่าให้กับเส้นเชื่อมที่เชื่อมอยู่ระหว่าง node ต่าง ๆ ที่อยู่ใน neural network โดยค่าที่กำหนดให้กับเส้นเชื่อมนี้จะเป็นค่าที่ได้จากการ random เพื่อที่เราต้องการให้ neural network นี้มันทำการ train และปรับน้ำหนักของเส้นเชื่อมระหว่าง node ต่าง ๆ ให้มีความถูกต้องเหมาะสมเพื่อเอาไว้ใช้ในการ recognition ได้ด้วยตัวของมันเอง

3.6.2 กระบวนการในการ train

ในขั้นตอนของการ train neural network นั้น อันดับแรก ก็จะต้องมี training data ซึ่ง training data นี้ทางผู้จัดทำก็ได้ใช้ training data ที่มาจาก nectec เนื่องจากว่าเป็น training data สำหรับ OCR โดยเฉพาะ และเพื่อที่จะเป็นการประหยัดเวลาทำให้ไม่ต้องไปสร้าง training data ขึ้นมาใหม่เอง อีกทั้ง training data สำหรับ OCR นั้น ก็มีจำนวนมากด้วย training data ที่ผู้จัดทำได้ไปเอามาจาก nectec นั้น ไม่ได้ไปเอามาทั้งหมด เพียงแต่คัดสรรมาเฉพาะบางส่วนที่จำเป็นจะต้องใช้เพื่อเป็นการศึกษาเกี่ยวกับ OCR เท่านั้น training data ที่ว่ามีลักษณะดังรูปข้างล่างนี้



รูปที่ 3.15 training data สำหรับตัวอักษรปกติ



รูปที่ 3.16 training data สำหรับตัวอักษร ตัวเอียง

training data ที่ใช้ก็จะอยู่ในรูปของไฟล์รูปภาพที่มีนามสกุลเป็น bmp ขนาด 24 บิต ใน training data จะประกอบไปด้วย พยัญชนะ สระ และ วรรณยุกต์ ในภาษาไทยที่ใช้งานกันโดยทั่วไปในปัจจุบัน ซึ่งก็จะมีจำนวนทั้งสิ้นถึง 78 ตัวด้วยกัน รูปภาพของ font ที่ใช้ในการ train ในครั้งนี้ประกอบไปด้วยรูปภาพ font จำนวนทั้งสิ้น 7 font ได้แก่

- AngsanaUPC
- BrowalliaUPC
- CordiaUPC

- DilleniaUPC
- EucrosiaUPC
- FreesiaUPC
- IrisUPC

เนื่องจากว่า project นี้เกี่ยวกับ OCR ดังนั้นภาพที่รับเข้ามาเพื่อจะทำการ recognition นั้นก็จะมาจาก Scanner (โดยการ scan เอาข้อความตัวอักษรเข้าไป ก็จะได้ไฟล์ภาพออกมา) ดังนั้น training data ที่เราใช้ทั้งหมดนี้จึงเป็น training data ที่เป็นภาพที่มาจากการ scan จริง ๆ ด้วยเช่นกันโดยไฟล์ภาพ training data ที่ได้มาทั้งหมดนี้ ถูก scan ที่ความละเอียด 300 dpiและใช้ชนิดของการ scan เป็นแบบ sharpen เพื่อให้มีความคมชัดมากขึ้น และภาพทุกภาพที่ใช้ล้วนแต่เป็นภาพที่ผ่านการ scan มาเพียงครั้งเดียวเท่านั้นด้วย ดังนั้นใน project นี้จึง recommend ภาพที่ผู้ใช้จะเอามาทำการ recognition ว่าควรจะเป็นดังที่กล่าวมาข้างต้นด้วย (สิ่งที่สำคัญก็คือ ความละเอียดของภาพนั้นไม่ควรที่จะต่ำกว่า 300 dpi ด้วย) นอกจากนี้ขนาดของ font ที่ใช้ใน training data นี้มีขนาดอยู่ตั้งแต่ 14-22 และ training data ที่ใช้ในครั้งนี้ก็จะมีทั้ง font ปกติ และ font ที่เป็นตัวเอียงด้วย

เมื่อมี training data แล้วขั้นตอนต่อไปคือการเอา training data ทั้งหมดมา train โดยใช้ neural network สำหรับการ train นี้ก็จะใช้ model ที่เรียกว่า back-propagation ในส่วนของการ train นั้นจะมีสิ่งที่แตกต่างจากส่วนของการ recognition คือว่า ในส่วนของการ train จะมีการกำหนด input และ กำหนด output ที่ถูกต้องแน่นอนให้กับ neural network จากนั้นการ train ก็จะเหมือนกับการปรับ weight ที่เชื่อมอยู่ระหว่าง node ทุก ๆ node ในแต่ละ layer ที่อยู่ใน neural network ให้มีค่าที่เหมาะสม (ตอนเริ่มต้นนั้นเรากำหนด weight โดยการ random ขึ้นมาก่อน) เพื่อที่จะทำให้ neural network นี้สามารถที่จะ recognition ได้อย่างถูกต้อง ในส่วนของการปรับ weight นั้นจะอาศัยวิธีการทาง neural network ที่เรียกกันว่า feed-forward ก่อนเพื่อที่จะคำนวณค่าของ error ของ weight จากนั้นเมื่อรู้ค่า error แล้วก็จะทำการ back-propagation หรือการแพร่กลับ เพื่อที่จะปรับ weight ให้มีค่า error ที่น้อยลง และในการ train ก็จะมีการทำซ้ำกระบวนการเช่นนี้ไปเรื่อย ๆ (คือทำ feed-forward ก่อนแล้วตามด้วย back-propagation) จนกระทั่งเราได้ค่า error ที่มีค่าน้อย ๆ ที่เป็นที่น่าพอใจ และมั่นใจพอว่า neural network ที่ผ่านการ train นี้สามารถที่จะ recognition ข้อมูลที่มันยังไม่รู้ได้แล้ว และสิ่งสุดท้ายที่เราทำก็คือ เราจะทำการ save weight เก็บไว้เพื่อเอาไว้ใช้งานในคราวต่อไป เพื่อเราจะได้ไม่ต้องมาทำการ train ใหม่อีกครั้ง เนื่องจากกระบวนการ train นี้เป็นกระบวนการที่ต้องอาศัยเวลาอย่างมาก ส่วน pseudo code ของการ train จะเป็นไปดังข้างล่างนี้

1. แปลงจาก training data ซึ่งเป็นรูปภาพ ให้อยู่ในรูปของ input vector ที่จำเป็นสำหรับตัวอักษรที่จะเอาไป train แต่ละตัว เพื่อเอาไปใส่เป็น input ของ neural network

2. กำหนด output ที่ถูกต้อง ซึ่ง match กับ feature ของตัวอักษรใด ๆ ที่ใส่เป็น input ของ neural network

3. ทำกระบวนการ feed-forward เพื่อที่จะคำนวณค่าของแต่ละ node ในทุก ๆ layer ในขณะนั้นว่าให้ผล

อะไรออกมา กระบวนการนี้จำเป็นเพื่อที่จะเอาค่าที่ได้จาก output node ของ neural network ไปเปรียบเทียบกับ output ที่เราอยากให้เป็นดังในขั้นตอนที่ 2 แล้วจะสามารถคำนวณค่า error ออกมาได้

4. คำนวณค่า error ออกมา จากนั้นก็เอาค่า error นี้บวกทบกับค่า error ที่ได้จากการ train ของแต่ละตัวอักษรในครั้งก่อน ๆ

5. ทำกระบวนการ back-propagation เพื่อเป็นการปรับ weight ที่เชื่อมในแต่ละ node ให้มันดีขึ้นเพื่อที่จะคำนวณค่าของ error ได้น้อยลงในครั้งต่อ ๆ ไป

6.ทำกระบวนการที่ 1 - 5 ใหม่แต่กับตัวอักษรตัวต่อไปที่อยู่ใน training data

7. เมื่อทำการ train จนครบทุก ๆ ตัวอักษรทั้งหมดของ training data แล้วให้หา error รวมโดยการเอา error ที่ได้จากการบวกทบไปเรื่อย ๆ ในขั้นตอนที่ 4 มาหารด้วยจำนวนรวมของตัวอักษรทั้งหมดที่อยู่ ใน training data (รวมไปถึงตัวที่มันซ้ำกันแต่อยู่คนละไฟล์ด้วย)

8. ทำการ check ค่า error ที่ได้ในข้อ 7 กับค่าที่เราต้องการ หากพบว่า error มันต่ำกว่าค่าที่เรากำ หนดไว้ก็ให้หยุดการ train ถือเป็นอันเสร็จสิ้นกระบวนการ train แต่ถ้าไม่ ก็ต้องกลับไปทำกระบวน การข้างต้นซ้ำอีกจนกว่าจะต่ำกว่าที่เรากำหนดไว้

3.6.3 การ recognition

เมื่อเรามีการ training data ไปแล้ว ส่วนต่อไปที่เราจะกล่าวถึงก็คือ การ recognition การ recognition คือการที่เราใส่ input ที่เป็น unknown เข้าไปแล้วให้ neural network ทำการวิเคราะห์ว่าสิ่งที่เราใส่เข้าไปนั้นจะเป็นตัวอักษรอะไร เมื่อผู้ใช้ต้องการใช้งานโปรแกรมเพื่อ recognition ภาพ สิ่งที่เราทำก็คือ เราก็จะสร้าง neural network ขึ้นมาชุดหนึ่ง ซึ่งเป็นชุดที่มีลักษณะเหมือนกับตอนที่เราทำการ train data เมื่อสร้าง neural network เสร็จแล้วแทนที่เราจะทำการ train ใหม่ เราก็จะโหลด weight ที่ save เก็บไว้นั้นขึ้นมาแล้วเอาไปใส่ใน neural network แทน ในส่วนของการ recognition นั้นเราก็จะเอาข้อมูลที่ผู้ใช้ป้อนเข้ามา (ในที่นี้ก็ได้แก่ไฟล์ภาพจากเครื่อง scanner) มาทำการแปลงให้อยู่ในรูปแบบที่เหมาะ สมสำหรับที่จะเอาเข้าสู่กระบวนการ recognition (ซึ่งกระบวนการแปลงต่าง ๆ นั้นได้กล่าวถึงมาก่อนหน้านี้แล้ว) หลังจากที่เอาข้อมูลมาแปลงเสร็จจนได้เป็นข้อมูลของแต่ละตัวอักษรไป เราก็จะเอาข้อมูลของแต่ละตัวอักษรนั้น (ในที่นี้ก็คือ feature ขนาด 65 อัน) ใส่เข้าไปเป็น input ของ neural network เพื่อที่จะให้ neural network นั้นทำการวิเคราะห์ออกมาว่าข้อมูลที่เราใส่เข้าไปจะเป็นตัวอักษรตัวอะไร ซึ่งเราจะใช้กระบวนการที่เรียกว่า feed-forward เพื่อหาค่าของ output node ที่มีค่าที่มากที่สุด และเราก็จะเอา output node ที่มากที่สุดนั้น ไปตีความหมายเพื่อให้ได้ตัวอักษรจริง ๆ ที่สอดคล้องกับ input (การทำเช่นนี้ เรียกได้อีกอย่างว่า winner-take-all เพราะว่าจะเอา output ที่มากที่สุดเป็นคำตอบ) และเราก็จะต้องทำกระบวนการนี้ซ้ำ ๆ กับทุก ๆ รูปตัวอักษรที่อยู่ในภาพ จนครบทุกตัวด้วย

3.7 Training set and validation set

เมื่อเรากำหนดข้อมูลใน Training data ทั้งหมดที่จะเอาไว้สำหรับเป็น training set และ validation set แล้ว ขั้นตอนต่อไปก็คือขั้นตอนของการ train ซึ่งในการ train ข้อมูลโดยใช้ training set และ validation set นั้นจะมีความแตกต่างจากขั้นตอนการ train โดยทั่วๆไป ดังนี้คือ

- ในขั้นตอนของการ train นั้น จะใช้ข้อมูลจากส่วนของ training set ซึ่งมีอยู่ 50% ของข้อมูลทั้งหมดนั้น เราจะเอาข้อมูลใน training set มา train ก่อน โดยจะมีการกำหนดจำนวน epochs ที่ต้องการที่แน่นอนไว้ด้วย เช่น 500 รอบ,1000 รอบ,1500 รอบ เป็นต้น

- ขั้นตอนต่อไป หลังจากที่เราทำการ train ข้อมูลไปจนครบจำนวนของ epochs ที่เรากำหนดไว้ตอนต้นแล้ว

ต่อไปเราก็จะเอา neural network ที่ผ่านการ train ไปทำการทดสอบกับ validation set การทดสอบกับ validation set ในที่นี้ก็คือการตรวจสอบความถูกต้องของ โครงข่าย neural network ที่เราได้ train มาแล้ว โดยเราจะเอาข้อมูลของแต่ละตัวอักษรใน validation set มาใส่ไว้เป็น input ของ neural network แล้วเราก็จะกำหนด output ที่ถูกต้องซึ่ง match กับข้อมูล input ที่เราได้ใส่เป็น input จากนั้นเราก็จะใช้กระบวนการ feed-forward ของ neural network กับข้อมูล input นั้น แล้วดูว่ามันให้ผลลัพธ์ออกมาเป็นตัวอะไร ซึ่งถ้ามันเหมือนกับข้อมูลของ output ที่ซึ่งเราได้กำหนดไว้ตั้งแต่ตอนแรกนั้น เราก็จะถือว่ามัน match กัน ถือว่าถูกต้อง

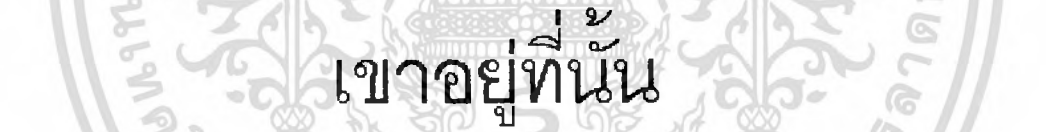
แต่ถ้าหากว่าให้ output ออกมาแล้วไม่ match กับข้อมูล output ที่ซึ่งเราได้กำหนดไว้ตั้งแต่ตอนต้น ก็ถือว่ามันไม่ match กัน และเราก็จะทำแบบนี้กับข้อมูลตัวอักษรทั้งหมดที่อยู่ใน validation set แล้ววัดเป็นเปอร์เซ็นต์ของความถูกต้องออกมาโดยดูว่า neural network มันให้ผลลัพธ์ออกมาถูกต้องกี่เปอร์เซ็นต์ จากนั้นก็เอาเปอร์เซ็นต์ความถูกต้องได้นี้มาเปรียบเทียบกับค่าที่เราได้กำหนดไว้ค่าหนึ่ง ซึ่งถ้าเปอร์เซ็นต์ความถูกต้องที่วัดได้มีค่าที่มากกว่าหรือเท่ากับที่เราได้กำหนดไว้นี้ เราก็จะหยุดการ train เพียงแค่นั้น และถือว่าข้อมูลมีความถูกต้องในระดับหนึ่งแล้ว และจะหยุดกระบวนการ trainไว้เพียงเท่านี้ด้วย แต่ถ้าหากว่าเปอร์เซ็นต์ความถูกต้องที่วัดได้มีค่าที่น้อยกว่าค่าที่กำหนดไว้ เราก็จะต้องกลับไปทำการ train ต่อ เพราะถือว่าข้อมูลยังมีความถูกต้องไม่เพียงพอ เราก็จะกลับไป train เท่ากับจำนวน epochs ที่เราได้กำหนดไว้ตั้งแต่ตอนต้น เมื่อ train เสร็จก็เอาไปทดสอบกับ validation set ต่อ และก็จะทำแบบนี้ไปเรื่อยๆจนกว่าเราจะได้เปอร์เซ็นต์ความถูกต้องที่มากกว่าค่าที่เราได้กำหนดไว้จึงจะหยุด

ปัญหาหนึ่งที่อาจจะพบได้ในการ train แบบนี้คือ เมื่อเราทำการ train ไปเรื่อยๆ กลับจะพบว่ามีการแกว่งของค่าเปอร์เซ็นต์ความถูกต้องที่วัดได้จาก validation set นั่นคือ เราสังเกตได้ว่าเมื่อเราทำการ train ไปเรื่อยๆ แล้วเอาไปทดสอบกับ validation set เปอร์เซ็นต์ความถูกต้องที่ได้ก็จะมีค่าที่เพิ่มมากขึ้นเรื่อยๆ แต่บางครั้งเราพบว่า เปอร์เซ็นต์ความถูกต้องของครั้งใหม่นั้น มีค่าที่น้อยกว่าเปอร์เซ็นต์ความถูกต้องของครั้งเก่า เมื่อเป็นเช่นนี้ก็แสดงได้เลยว่า neural network เริ่ม overfit แล้ว เราก็จะทำการแก้ไขโดยการ backtrack กลับไปที่ state ก่อนหน้าก่อนที่เปอร์เซ็นต์ขอองความถูกต้องจะลดลง ซึ่งคือ state ที่ให้ค่าของเปอร์เซ็นต์ความถูกต้อง

ออกมาสูงที่สุด จากนั้นก็โหลดค่าของ weight เดิมที่เราได้ทำการ backup เก็บไว้ที่ state นั้นกลับเข้ามาใส่ให้แก่ neural network และก็ลดจำนวนของ epochs ลงให้กลับไปมีค่าเท่ากับค่าของ epochs ตอนที่อยู่ที่ state ก่อนหน้าด้วย เมื่อเรา backtrack กลับไปก่อนหน้าแล้วสิ่งที่เราทำต่อไปก็คือ ลดค่าของ learning rate ลง ทั้งนี้เพื่อให้อัตราของการเรียนรู้นั้นน้อยลง หรือว่าช้าลงด้วย ซึ่งจะช่วยให้ไม่เกิดการ oscillate ขึ้นอีก นอกจากนี้หลังจากที่เราลดค่าของ learning rate ลงเสร็จแล้ว เราก็อาจจะลดค่าของ epochs ลงได้ด้วยเช่นกัน ทั้งนี้ เราลดค่าของ epochs ลงก็เพื่อให้เราสามารถนำ neural network นั้นไปเข้าสู่กระบวนการตรวจสอบโดยใช้ validation set ได้บ่อยขึ้น

3.8 Algorithm ในการเรียงลำดับตัวอักษร

หลังจากที่เราหาระดับต่างๆในแต่ละบรรทัดได้แล้ว (ในที่นี้ก็คือระดับตั้งแต่ระดับที่ 0 ซึ่งก็คือระดับของสระอุ สระอู เป็นต้น ไปจนถึงระดับบนสุด ซึ่งก็คือระดับที่ 3) รวมทั้ง blob ตัวอักษรแต่ละตัวที่เราสามารถแยกออกมาจากรูปภาพได้ ก็ผ่านขั้นตอนของการ Recognition จนได้มาเป็นตัวอักษรจริงๆซึ่ง match กับ blob ตัวนั้นๆแล้วทุกๆตัว ขั้นตอนต่อไปก็คือขั้นตอนของการเรียงตัวอักษรต่างๆที่ recognize ออกมาได้นั้นให้เป็นข้อความที่ถูกต้อง เหมือนกับไฟล์รูปภาพก่อนกระบวนการ process ต่างๆ ซึ่งการเรียงตัวอักษรต่างๆที่ recognize ออกมาได้นั้นให้เป็นข้อความที่ถูกต้องนั้น เราจะอาศัยการดูจากตำแหน่งของ blob แต่ละตัวเป็นหลัก ทั้งนี้ก็เพราะว่าลำดับของ blob แต่ละตัวที่ออกมาได้นั้น มักจะเป็นลำดับของการพิมพ์ที่ไม่ถูกต้อง ลองดูจากรูปข้างล่างนี้



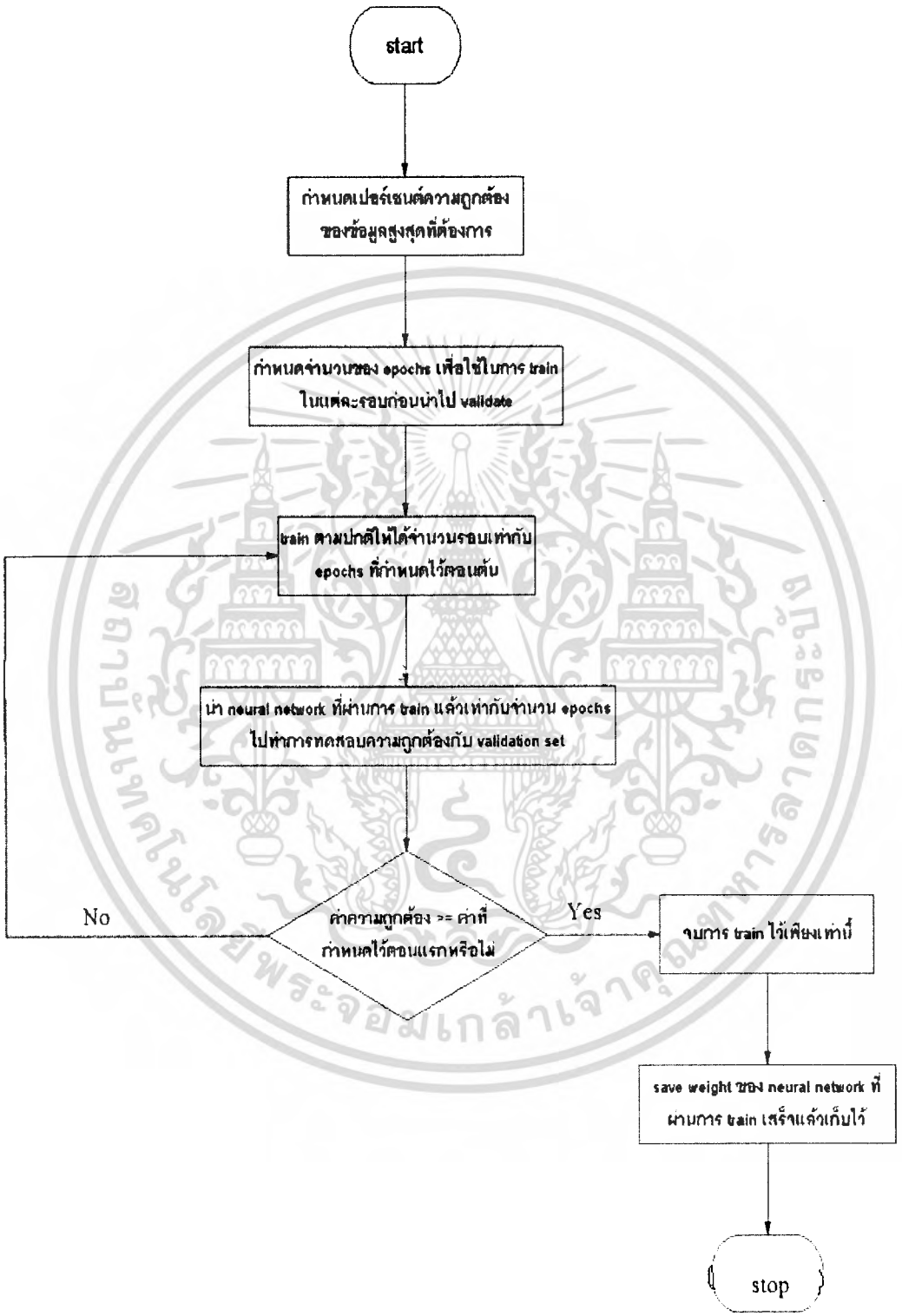
รูปที่ 3.17 ตัวอย่างของข้อความที่ต้องผ่านการเรียงลำดับตัวอักษรใหม่

จากรูปข้างบนนี้เมื่อเราสามารถแยก blob ตัวอักษรแต่ละตัวออกมาได้ ลำดับของ blob ที่สามารถแยกออกมาได้จะเป็นดังนี้คือ อันดับ 1 ได้แก่ ไม้เอกที่อยู่ในระดับที่ 3 อันดับ 2 ได้แก่ ไม้โทที่อยู่ในระดับที่ 3 อันดับ 3 ได้แก่ ไม้เอก ที่อยู่ในระดับที่ 2 อันดับ 4 ได้แก่ สระอี ที่อยู่บนตัว ท เป็นต้น ที่เป็นเช่นนี้ก็เนื่องจากว่า เราได้ทำการ process ภาพเพื่อหา blob จากบนลงมาสู่ล่าง และก็จากซ้ายไปสู่ขวาตามลำดับ จึงได้เจอ sequence ของ blob เป็นดังที่กล่าวมา เมื่อเป็นเช่นนี้จึงทำให้เราต้องทำการเรียงลำดับของ blob ใหม่ เพราะว่าตอนที่เราทำการพิมพ์ข้อความจริงๆนั้นเราไม่สามารถพิมพ์ไม้เอกมาก่อนแล้วตามด้วยไม้โทดัง sequence ข้างบนได้ เราจะต้องพิมพ์ blob ที่อยู่ในระดับที่ 1 ก่อน ซึ่งในที่นี้ก็ได้แก่ สระเอ (เ) คือตัวแรกที่เราต้องพิมพ์ลงไปใน file ผลลัพธ์ จากนั้นก็ตามด้วย ข,ว ตามลำดับ พูดง่ายๆก็คือ พิมพ์ตัวอักษรในระดับที่ 1

เรียงตามลำดับจากซ้ายไปขวา และถ้าหากว่าตัวอักษรทีเราพิมพ์ซึ่งอยู่ในระดับที่ 1 นั้นมีวรรณยุกต์หรือสระอื่นๆประกอบอยู่ข้างบนหรือข้างล่างของตัวมัน

เราก็จะพิมพ์สระหรือวรรณยุกต์ตัวนั้นๆต่อจากพยัญชนะ โดยมีข้อแม้ว่าเมื่อพิมพ์พยัญชนะเสร็จแล้วจะทำการพิมพ์สระหรือวรรณยุกต์ ถ้าหากว่าข้างใต้พยัญชนะมี สระอุ หรือ สระอู อยู่ และมีวรรณยุกต์ที่อยู่บนพยัญชนะอยู่ด้วย เราจะต้องพิมพ์ระดับที่ 0 ก่อน ซึ่งในที่นี้ก็ได้แก่ ว่า เราจะต้องพิมพ์ สระอุ หรือ สระอู ก่อน แล้วจึงตามด้วยวรรณยุกต์ที่อยู่บนพยัญชนะตัวนั้นๆตามมา แต่ถ้าหากมรสระหรือวรรณยุกต์ที่อยู่ในลำดับที่ติดกัน เช่น มีสระอยู่ในระดับที่2 และมีวรรณยุกต์อยู่ในระดับที่ 3 ถ้าเป็นเช่นนี้เราก็จะทำการพิมพ์สระที่อยู่ในระดับที่ 2 แล้วตามด้วยวรรณยุกต์ที่อยู่ในระดับที่ 3 ตามลำดับ แต่ถ้าหากว่า พยัญชนะนั้นๆมีสระหรือวรรณยุกต์กำกับเพียงตัวเดียว เราก็ทำการพิมพ์ตัวนั้นต่อจากพยัญชนะได้เลยโดยไม่มีข้อแม้อย่างใดเลย

Flowchart ของอัลกอริธึมแบบ Training Set and Validation Set ดังนี้

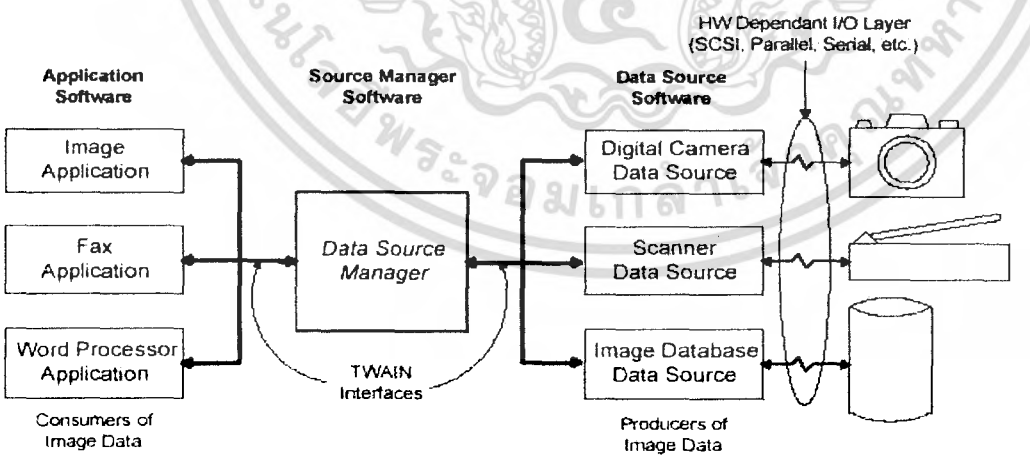


3.9 การ Interfacing กับ scanner

ในโปรเจคนี้อนุญาตให้มีการรับภาพมาจาก scanner ได้ ซึ่งผู้ใช้สามารถรับภาพจาก scanner ได้จากภายในตัวโปรแกรมได้เลย ซึ่งจะมีปุ่มสำหรับการเลือกชนิดของ scanner ที่ติดต่ออยู่กับคอมพิวเตอร์ และก็จะมีปุ่ม scan ซึ่งเอาไว้สำหรับ scan เพื่อให้ได้ภาพออกมา สำหรับในโปรแกรมที่พัฒนาขึ้นมานั้น เมื่อ scan ภาพเข้ามาจาก scanner แล้ว ภาพจะมีความละเอียด 300 dpi ซึ่งเป็นความละเอียดที่ค่อนข้างจะสูง แต่มีข้อดีคือจะทำให้โปรแกรมสามารถ process ภาพได้ดีมากยิ่งขึ้น รวมทั้งมี noise ที่ไม่มากด้วย สำหรับการเขียนโปรแกรมเพื่อติดต่อกับ scanner นั้น เราจะอาศัย tool ช่วยที่มีชื่อว่า twain ร่วมกับ library ของ EZTWAIN ทั้งนี้เนื่องจากว่าตัว twain นั้นมีโครงสร้างที่ซับซ้อนและยาก และต้องใช้ระยะเวลานานกว่าจะศึกษาทำความเข้าใจถึงฟังก์ชันการคำนวณของมันทั้งหมด จึงเลือกใช้ EZTWAIN ซึ่งเป็น library สำหรับไปติดต่อกับ twain อีกที ทั้งนี้เพราะ EZTWAIN นั้น ได้ encapsulate การติดต่อกับ twain ในส่วนที่ค่อนข้างจะยุ่งยากไว้หมดแล้ว เหลือเพียงให้เราเขียน code เพื่อ interface กับตัวของ EZTWAIN เอง ซึ่งมีความง่ายกว่าการเขียน code เพื่อ interface โดยตรงกับ twain มากๆ ต่อไปจะพูดถึง twain และโครงสร้างของ twain

Twain คือเป็นการริเริ่มจัดทำมาตราฐานการติดต่อสื่อสาร (โปรโตคอล) ที่ใช้กับอุปกรณ์ประดิษฐ์เกี่ยวกับภาพ, รูป (Imaging Devices เกิดขึ้นจากกลุ่มอุตสาหกรรมผู้ผลิตอุปกรณ์รวมตัวกันโดยไม่หวังผลประโยชน์ ซึ่งต้องการมาตราฐานซอฟท์แวร์ในการติดต่อสื่อสารและเครื่องมือในการเขียนโปรแกรมเพื่อติดต่อสื่อสารกับอุปกรณ์ (API: Applications Programming Interface) ซึ่งตั้งข้อกำหนดในการใช้งานไว้ 3 ส่วนประกอบหลักของ Twain คือ

1. Application Software
2. Source Manager Software
3. Data Source Software



รูปที่ 3.18 รูปแสดงไดอะแกรมการติดต่อระหว่างคอมพิวเตอร์กับอุปกรณ์ที่ใช้ TWAIN

ต่อไปจะกล่าวถึง EZWAIN และฟังก์ชันต่างๆที่จำเป็นต้องเรียกใช้เพื่อให้สมารถรับภาพเข้ามาจาก scanner ได้ฟังก์ชันต่าง ๆ ของการรับภาพจาก Scanner โดย Eztwain มีหน้าที่ดังนี้

Twain_SelectImageSource

ฟังก์ชันนี้มีไว้สำหรับเลือก Twain device (ในที่นี้ก็ได้แก่เครื่อง scan) ที่ต่อเชื่อมอยู่กับเครื่องคอมพิวเตอร์ของเรา ในกรณีที่มี device เพียงอันเดียวที่เชื่อมอยู่ ก็ไม่จำเป็นต้องเรียกฟังก์ชันนี้ก็ได้ ซึ่งฟังก์ชันนี้จะทำงานก็ต่อเมื่อมี device มากกว่า 1 ตัว ให้เราเลือก

Twain_SetHideUI

ฟังก์ชันนี้จะทำหน้าที่ซ่อนหน้าต่างของ User interface ไม่ให้ผู้ใช้เห็น

Twain_OpenDefaultSource

ฟังก์ชันนี้เอาไว้สำหรับตรวจสอบดูว่า Dialog ของ source select นั้นเคยเปิดมาก่อนแล้วหรือยัง ถ้าหากว่าเคยเปิดมาก่อนแล้ว ก็จะไม่ทำอะไรเลย และก็จะ return ค่าออกมาเป็น true และจะ return ค่าเป็น false ถ้าหากว่า source manager นั้นไม่ถูก load และไม่ถูกเปิด

Twain_SetCurrentUnits

ฟังก์ชันนี้ใช้สำหรับทำการเซตหน่วยการวัดในปัจจุบันสำหรับ Source หน่วยของการวัดนี้จะถูกประกาศไว้ในไฟล์ Twain.h ในที่นี้เรากำหนดให้หน่วยของการวัดอยู่ในหน่วยของ นิ้ว

Twain_SetCurrentPixelType

ฟังก์ชันนี้ทำการเซตรูปแบบปัจจุบันของจุด เพื่อที่จะรับเข้ามาจาก Scanner ให้ได้ไปเป็นตามรูปแบบนั้น

Twain_SetBitDepth

ฟังก์ชันนี้จะทำการเซตความลึกของบิต ที่เอาไว้สำหรับแสดงค่าสีของ Pixel ใด ๆ ในที่นี้เซตเป็น 24 บิต หมายความได้ว่า 1 pixel นั้นใช้ 24 บิตในการเก็บ แบ่งเป็น Red 8 บิต Green 8 บิต และ Blue 8 บิต

Twain_SetCurrentResolution

ฟังก์ชันนี้ทำการเซตความละเอียดของภาพ ที่เราต้องการจะรับมาจาก Scanner ความละเอียด จะอยู่ในรูปของจำนวนจุดต่อหน่วยปัจจุบัน ซึ่งหน่วยปัจจุบันนั้น เราได้เซตไว้โดยฟังก์ชันTwain_SetCurrentUnits เรียบร้อยแล้ว

Twain_AcquireNative

ฟังก์ชันนี้จะทำการรับรูปภาพเข้ามาจาก Data source ที่ถูกเลือกอยู่ในปัจจุบันจำนวนเพียงแค่ 1 รูปภาพเท่านั้น ฟังก์ชันนี้จะ return ค่าส่งกลับเป็น handle DIB ของรูปภาพที่ถูกรับเข้ามา

Twain_WriteNativeToFilename

ฟังก์ชันนี้จะทำหน้าที่เขียนไฟล์ภาพซึ่งเราได้รับมาจาก Scanner โดยภาพที่เรารับเข้ามาจาก scanner นั้นจะถูกเก็บไว้เป็น handle DIB ที่ได้มาจากฟังก์ชัน Twain_AcquireNative แล้วจากนั้นเราก็จะต้องส่ง handle DIB ตัวนี้ไปเป็นพารามิเตอร์ของฟังก์ชันนี้ด้วย เพื่อให้มันสร้างไฟล์ภาพที่เราต้องการออกมาให้

บทที่ 4
ผลการทดลอง

สำหรับผลการทดลองในโครงงานนี้เราได้เลือกฟอนต์ที่จะมาทดสอบจำนวนทั้งสิ้น 7 ฟอนต์ด้วยกันซึ่งได้แก่

- AngsanaUPC ขนาด 16 pt และ 24 pt
- BrowalliaUPC ขนาด 16 pt และ 24 pt
- CordiaUPC ขนาด 16 pt และ 24 pt
- DilleniaUPC ขนาด 16 pt และ 24 pt
- EucrosiaUPC ขนาด 16 pt และ 24 pt
- FreesiaUPC ขนาด 16 pt และ 24 pt
- IrisUPC ขนาด 16 pt และ 24 pt

ภาพที่เราได้สแกนเข้ามานั้น เราทำการสแกนที่ความละเอียด 300 dpi โดยที่สแกนเป็นแบบ สีขาว-ดำ สำหรับทุก ๆ ภาพ บนกระดาษขนาด A4 ที่มีข้อความที่เราต้องการทดสอบอยู่ โดยข้อความที่เราใช้ในการทดสอบครั้งนี้คือข้อความดังข้างล่างนี้

"เชลซียังไม่ยอมยกธงแม้รู้ข่าว
แมนฯยูไนเต็ดชิงเก็บชัยช่วงต้น
หัวค่ำด้วยการบุกมาเอาชนะ
ปอร์ทสมัธถึงถิ่น"

เราทำการพิมพ์ข้อความข้างต้นด้วยฟอนต์ต่างกัน ที่แต่ละฟอนต์ มีขนาด 16 pt และ 24 pt ซึ่งมี 7 ฟอนต์ดังที่ได้กล่าวมาแล้ว ก็จะทำให้เราได้รูปที่ผ่านการสแกนมาทั้งสิ้น 14 รูปดังข้างล่างนี้

เชลซียังไม่ยอมยกธงแม้รู้ข่าว
แมนฯยูไนเต็ดชิงเก็บชัยช่วงต้น
หัวค่ำด้วยการบุกมาเอาชนะ
ปอร์ทสมัธถึงถิ่น
AngsanaUPC ขนาด 16 pt

เชลซียังไม่ยอมยกธงแม้รู้ข่าว
แมนฯยูไนเต็ดชิงเก็บชัยช่วงต้น
หัวค่ำด้วยการบุกมาเอาชนะ
ปอร์ทสมัธถึงถิ่น

AngsanaUPC ขนาด 24 pt

เชลซียังไม่ยอมยกธงแม้รู้ข่าว
แมนฯยูไนเต็ดชิงเก็บชัยช่วงต้น
หัวค่ำด้วยการบุกมาเอาชนะ
ปอร์ทสมัธถึงถิ่น

BrowalliaUPC ขนาด 16 pt

เชลซียังไม่ยอมยกธงแม้รู้ข่าว
แมนฯยูไนเต็ดชิงเก็บชัยช่วงต้น
หัวค่ำด้วยการบุกมาเอาชนะ
ปอร์ทสมัธถึงถิ่น

BrowalliaUPC ขนาด 24 pt

เชลซียังไม่ยอมยกธงแม้รู้ข่าว
แมนฯยูไนเต็ดชิงเก็บชัยช่วงต้น
หัวค่ำด้วยการบุกมาเอาชนะ
ปอร์ทสมัธถึงถิ่น

CordiaUPC ขนาด 16 pt

เชลซียังไม่ยอมยกธงแม้รู้ข่าว
แมนฯยูไนเต็ดชิงเก็บชัยช่วงต้น
หัวค่ำด้วยการบุกมาเอาชนะ
ปอร์ทสมัธถึงถิ่น

CordiaUPC ขนาด 24 pt

เชลซียังไม่ยอมยกธงแม้รู้ข่าว
แมนฯยูไนเต็ดชิงเก็บชัยช่วงต้น
หัวค่ำด้วยการบุกมาเอาชนะ
ปอร์ทสมัธถึงถิ่น

DilleniaUPC ขนาด 16 pt

เชลซียังไม่ยอมยกธงแม้รู้ข่าว
แมนฯยูไนเต็ดชิงเก็บชัยช่วงต้น
หัวค่ำด้วยการบุกมาเอาชนะ
ปอร์ทสมัธถึงถิ่น

DilleniaUPC ขนาด 24 pt

เชลซียังไม่ยอมยกธงแม้รู้ข่าว
แมนฯยูไนเต็ดชิงเก็บชัยช่วงต้น
หัวค่ำด้วยการบุกมาเอาชนะ
ปอร์ทสมัธถึงถิ่น

EucrosiaUPC ขนาด 16 pt

เชลซียังไม่ยอมยกธงแม้รู้ข่าว
แมนฯยูไนเต็ดชิงเก็บชัยช่วงต้น
หัวค่ำด้วยการบุกมาเอาชนะ
ปอร์ทสมัธถึงถิ่น

EucrosiaUPC ขนาด 24 pt

เชลซียังไม่ยอมยกธงแม้รู้ข่าว
แมนฯยูไนเต็ดชิงเก็บชัยช่วงต้น
หัวค่ำด้วยการบุกมาเอาชนะ
ปอร์ทสมัธถึงถิ่น

FreesiaUPC ขนาด 16 pt

เชลซียังไม่ยอมยกธงแม้รู้ข่าว
แมนฯยูไนเต็ดชิงเก็บชัยช่วงต้น
หัวค่ำด้วยการบุกมาเอาชนะ
ปอร์ทสมัธถึงถิ่น

FreesiaUPC ขนาด 24 pt

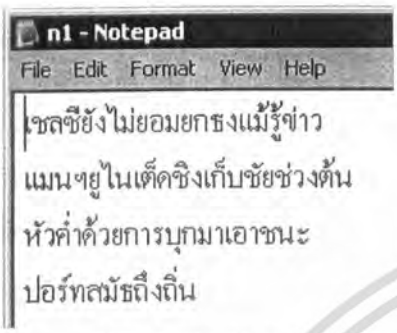
เชลซียังไม่ยอมยกธงแม้รู้ข่าว
แมนฯยูไนเต็ดชิงเก็บชัยช่วงต้น
หัวค่ำด้วยการบุกมาเอาชนะ
ปอร์ทสมัธถึงถิ่น

IrisUPC ขนาด 16 pt

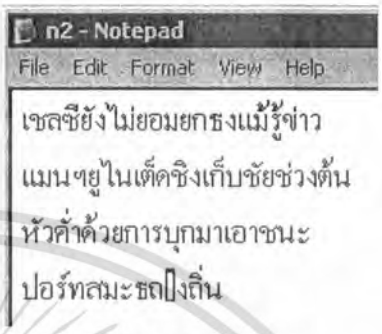
เชลซียังไม่ยอมยกธงแม้รู้ข่าว
แมนฯยูไนเต็ดชิงเก็บชัยช่วงต้น
หัวค่ำด้วยการบุกมาเอาชนะ
ปอร์ทสมัธถึงถิ่น

IrisUPC ขนาด 24 pt

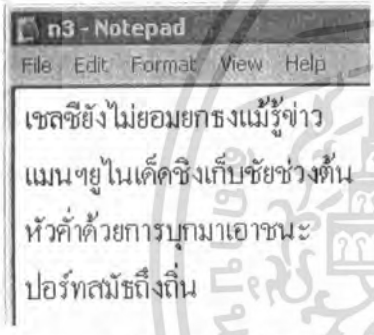
ต่อไปเป็นผลที่ได้จากการ recognize ของโครงงานนี้ ในที่นี้ทุกภาพที่รับเข้ามาก่อนที่จะถูก recognize นั้น จะถูกทำการ threshold ให้ได้เป็นภาพแบบไบนารีก่อน โดยที่ได้ทดลองทำการกำหนดค่า threshold ให้มีค่าเป็น 151 สำหรับทุก ๆ ภาพด้วย



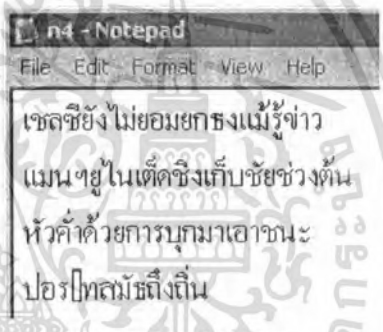
AngsanaUPC ขนาด 16 pt



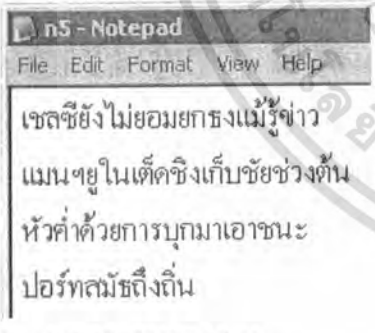
AngsanaUPC ขนาด 24 pt



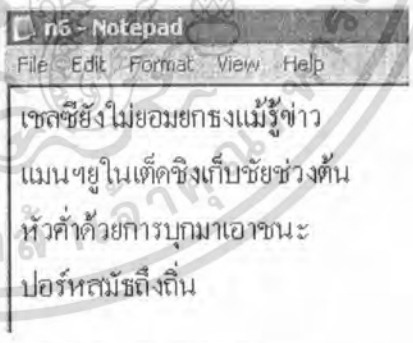
BrowalliaUPC ขนาด 16 pt



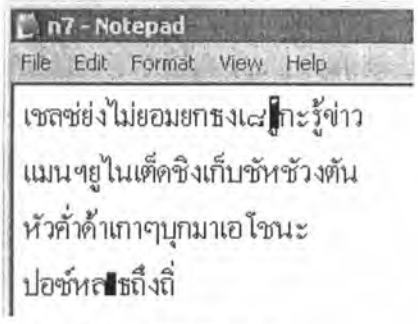
BrowalliaUPC ขนาด 24 pt



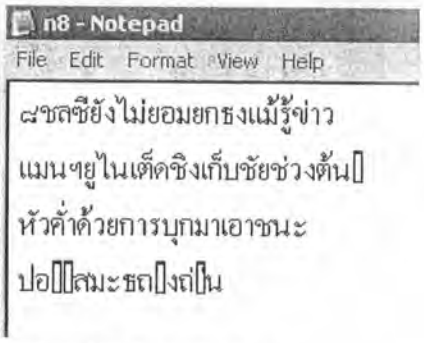
CordiaUPC ขนาด 16 pt



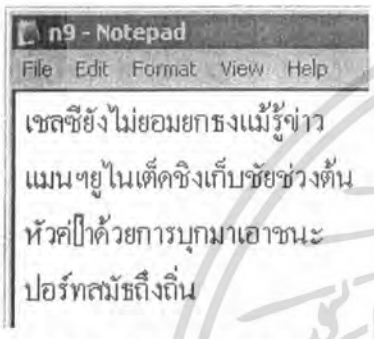
CordiaUPC ขนาด 24 pt



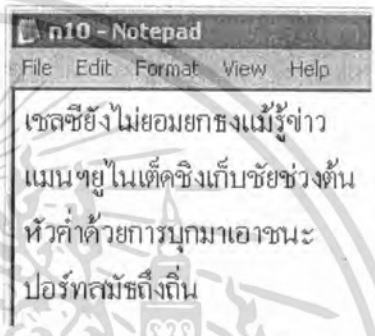
DilleniaUPC ขนาด 16 pt



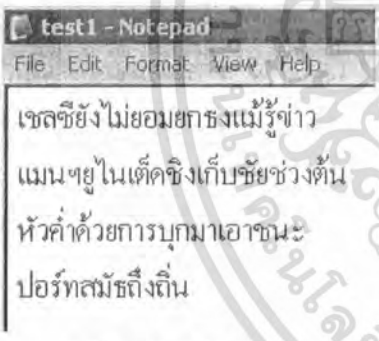
DilleniaUPC ขนาด 16 pt



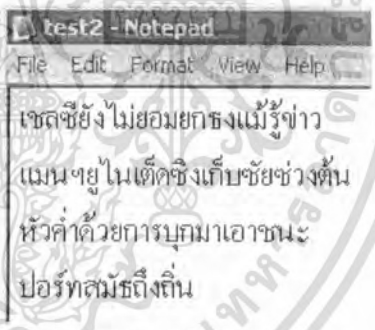
EucrosiaUPC ขนาด 16 pt



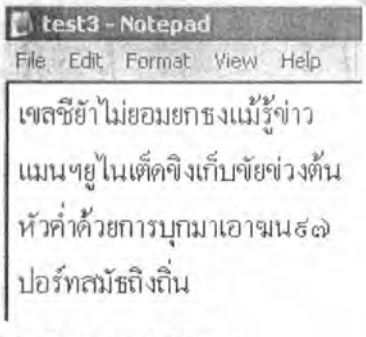
EucrosiaUPC ขนาด 24 pt



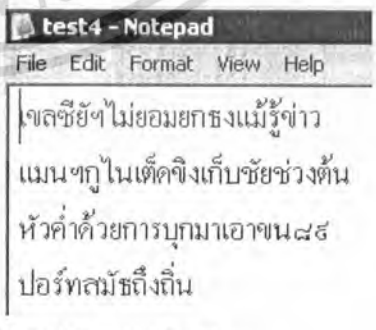
FreesiaUPC ขนาด 16 pt



FreesiaUPC ขนาด 24 pt



IrisUPC ขนาด 16 pt



IrisUPC ขนาด 24 pt

ตารางแสดงผลการทำงานและการแยกแยะตัวอักษรของโปรแกรม

ชื่อฟอนต์	ขนาด	จำนวนทั้งหมด (ตัว)	จำนวนที่แยกแยะได้ถูก (ตัว)	คิดเป็นเปอร์เซนต์ของทั้งหมด
angsanaUPC	16	100	100	100
angsanaUPC	24	100	97	97
browalliaUPC	16	100	98	98
browalliaUPC	24	100	98	98
cordiaUPC	16	100	99	99
cordiaUPC	24	100	96	96
dilleniaUPC	16	100	82	82
dilleniaUPC	24	100	90	90
eucrosiaUPC	16	100	99	99
eucrosiaUPC	24	100	100	100
freesiaUPC	16	100	100	100
freesiaUPC	24	100	97	97
DilleniaUPC	16	100	94	94
DilleniaUPC	24	100	95	95

รวมเปอร์เซนต์ความถูกต้องเฉลี่ย = 96.071 %

บทที่ 5
สรุปผลการทดลอง

ข้อจำกัดของ Project มีดังต่อไปนี้

1. ไม่สามารถที่จะรับภาพของข้อความ ที่ข้อความแต่ละบรรทัดไม่อยู่ในแนวระดับได้ ซึ่งใน project นี้กำหนดให้ตัวอักษรทุกตัวจะต้องอยู่ในระดับเดียวกันเท่านั้น
2. ตัวอักษรทุกตัวควรจะทับกันให้น้อยที่สุดเท่าที่จะทำได้ ทั้งนี้ก็เพื่อให้การ recognition นั้นทำได้อย่างมีประสิทธิภาพมากที่สุดเท่าที่จะทำได้นั่นเอง เช่น อัลกอริธึมจะไม่สามารถแยกแยะตัว ปั ซึ่งไม้หันอากาศมันทับกับตัว ป อยู่นั่นเอง ในกรณีการทับกันเช่นนี้อาจจะให้ผลการ recognize ที่ผิดพลาดได้
3. สามารถทำงานได้ดีกับรูปภาพที่ผ่านการ scan ที่ความละเอียดตั้งแต่ 300 dpi ขึ้นไป
4. ขนาดของฟอนต์ที่จะสามารถ recognize ได้อย่างมีประสิทธิภาพควรจะมีขนาดตั้งแต่ 16 point ขึ้นไป
5. บางทีตัว การันต์ หรือว่า ไม้ไต่คู้ ก็ชอบมีปัญหา ตรงที่มันอาจที่จะไปทับกับระดับอื่น ๆ ได้ด้วยกันนั่นเอง เพราะว่าหางของมันยาว
6. ผู้ใช้ต้องกำหนดค่า threshold เอง ทั้งนี้เพราะว่าการอนุญาติให้ผู้ใช้กำหนดค่า threshold เอง จะทำให้มีความผิดพลาดในการกำหนดค่า threshold น้อยกว่าทำให้เป็นระบบอัติโนมัติ
7. ไม่ควรนำภาพที่ฟอนต์ต่างกัน หรือว่าฟอนต์เดียวกันแต่มีขนาดต่างกัน มาทำการ recognize โดยโปรแกรมนี้ เพราะจะทำให้ได้ผลที่ผิดเพี้ยนไปจากภาพที่ input เข้ามา
8. ควรจะ scan ภาพในโหมด black & white มากกว่าที่จะเป็นแบบโหมด color หรือ grayscale

สรุป

จากผลการทดลองเราจะพบว่า มีเปอร์เซ็นต์ความถูกต้องอยู่ที่ประมาณ 90 กว่าเปอร์เซ็นต์ และโครงงานสามารถแยกบรรทัดได้อย่างถูกต้อง ทั้งนี้ปัจจัยหลายอย่างที่มีผลต่อความถูกต้องของข้อมูลได้แก่

- ขนาดความละเอียดของภาพ ยิ่งภาพมีความละเอียดมาก ก็จะทำให้โปรแกรมสามารถแยกแยะตัวอักษรได้ดียิ่งขึ้น

- ระยะห่างในแต่ละระดับที่ตัวอักษรแต่ละตัวในภาพวางอยู่ ก็มีผล เนื่องจากถ้าระดับของอักษรเหล่านี้ห่างจากระดับอื่น (คือมีช่องว่างขั้นกลาง) อย่างชัดเจน ก็จะทำให้กระบวนการหาตัวอักษร และกระบวนการจัดเรียงตัวอักษรในขั้นสุดท้ายทำได้อย่างมีประสิทธิภาพ

- การกำหนดค่า threshold ของภาพก็มีผล เนื่องจากว่าในโครงงานนี้จะอาศัยภาพที่ผ่านการ threshold แล้วไปเข้าสู่กระบวนการ recognize อื่นๆต่อไป และในโครงงานนี้ก็อนุญาตให้ผู้ใช้กำหนดค่า threshold เองด้วย เนื่องจากการทำ adaptive threshold นั้น ได้ลองทำมาแล้ว แต่ให้ผลลัพธ์ที่ไม่ค่อยจะแน่นอนเท่าไหร่ นอกจากนี้ การกำหนด threshold เอง ยังเป็นการทำให้เห็นภาพ binary ที่ได้ว่าเหมาะสมหรือดีพอที่จะเอาไปเข้าสู่กระบวนการอื่นๆหรือไม่

- ค่าของความห่าง ที่บอกว่า ตัวอักษรที่อยู่คนละระดับกันนั้น อยู่คนละบรรทัดกันด้วย ก็มีผลเช่นกัน และมีผลมากด้วย ในโครงงานนี้สามารถคำนวณค่าของความห่างระหว่างบรรทัดได้เองอย่างอัตโนมัติ แต่บางครั้ง (นานๆครั้ง) ก็อาจจะคำนวณค่านี้ผิดพลาดได้ ซึ่งขึ้นกับปัจจัยหลายอย่าง เช่น ระยะห่างในแต่ละระดับของข้อความในหลายๆบรรทัดอาจมีค่าที่แตกต่างกันมากเกินไป ในโครงงานนี้จึงได้เพิ่มให้ผู้ใช้สามารถกำหนดระยะห่างที่จะเอาไว้แยกบรรทัดได้ เช่นเดียวกันกับการกำหนดค่า threshold ได้นั่นเอง

- การที่ภาพที่นำมาวิเคราะห์ ประกอบไปด้วยตัวอักษรหลายขนาด และ font ต่างกัน ก็มีผลต่อความถูกต้องของข้อมูลเช่นกัน

โดยรวมแล้วก็อยู่ในขั้นที่น่าพอใจ เพราะถึงแม้ว่าจะมีข้อผิดพลาดหรือเกิด error ที่ไม่สามารถคาดการณ์ได้เช่นการเกิด error เมื่อข้อความในบรรทัดใด ๆ มีการเอียงเพียงเล็กน้อย แต่ว่าถ้าภาพที่รับเข้ามาอยู่ในแนวระดับเดียวกันหมดทุกตัวอักษรในบรรทัด ก็จะให้ผลลัพธ์ออกมาที่ดีมาก นอกจากนี้ระบบ neural network นั้นสามารถ train ใหม่ให้มีความถูกต้องมากยิ่งขึ้นด้วยเช่นกัน ซึ่งก็จะทำให้ได้ผลลัพธ์ของข้อมูลที่มีความถูกต้องมากขึ้นตามไปด้วย

วิจารณ์ผลการทดลอง

โครงงานนี้ถือเป็นการศึกษา และทำให้เข้าใจกระบวนการของ OCR (optical character recognition) ได้ดียิ่งขึ้น ทำให้ได้เรียนรู้สิ่งใหม่ๆ ยกตัวอย่างเช่น ได้เรียนรู้เกี่ยวกับ วิธีการแยกระดับของข้อความภาษาไทย รวมไปถึงวิธีการตรวจสอบภาพข้อความใด ๆว่ามีกี่บรรทัด ได้เรียนรู้เกี่ยวกับการ train neural network โดยใช้วิธี training set และ validation set เป็นต้น นอกจากนี้ถึงแม้ว่าจะยังไม่สามารถนำไปใช้งานได้จริงเพราะยังมีข้อผิดพลาดบางอย่างอยู่ เช่น เมื่อข้อความในบรรทัดใด ๆ มีการเอียงเพียงเล็กน้อย ก็ทำให้โปรแกรมแฮงค์ได้ แต่ทางผู้จัดทำก็ยังเชื่อว่า วิธีการต่าง ๆ ที่ได้พัฒนาลงไปในโครงงานนี้ล้วนแต่เป็นทักษะ และเป็นประโยชน์ในการพัฒนาต่อยอด เพื่อที่จะทำให้โครงงานนี้สามารถใช้งานได้ดียิ่งขึ้นในอนาคต

เอกสารอ้างอิง

Thai OCR : A Neural Network Application

Chularat Tanprasert and Thaweesak Koanantakool
National Electronics and Computer Technology Center (NECTEC)
National Science and Technology Development Agency
Ministry of Science, Technology and Environment
Bangkok, THAILAND 10400

ภาคผนวก

Library ต่างๆที่ใช้ช่วยใน project นี้

- Opencv

เป็น library ที่พัฒนาขึ้นโดยบริษัท Intel ซึ่งพัฒนาขึ้นโดยเหล่านักวิจัยของ Intel เอง เป็น library ที่ open source และมีความสามารถช่วยในการพัฒนาระบบ image processing ได้เป็นอย่างดี โดยมีจุดประสงค์หลักคือ ระบบคอมพิวเตอร์ vision แบบ real time ตัวอย่างของ application ที่สามารถใช้ opencv ช่วยได้แก่

- Image arithmetic and logic operations
- Image creation and access (same image header used for both libraries)
- Image filtering
- Linear image transformation
- Image morphology
- Color space conversion
- Image histogram and thresholding
- Geometric transformation (zoom-decimate, rotate, mirror, shear, warp, perspective transform. affine transform)
- Image moments

สำหรับข้อมูลเพิ่มเติมสามารถดูได้ที่

http://www.intel.com/technology/computing/opencv/overview.htm

หรือว่าสามารถ ดาวน์โหลดตัว library ได้ที่

http://sourceforge.net/projects/opencvlibrary/

- Allegro

เป็น library สำหรับการสร้างเกม แต่ที่เอามาใช้ใน project นี้ก็เพื่อจะเอาไว้แสดง graphic ตอนที่เราทำการ train neural network ทั้งนี้เราใช้ Allegro ในการแสดงค่าของจำนวนรอบที่ผ่านการ train แล้ว และเอาไว้แสดงเปอร์เซนความถูกต้องของข้อมูลที่ผ่านการ train โดยวิธี training set และ validation set ด้วย โดยความถูกต้องจะแสดงออกมาอยู่ระหว่าง 0.0 – 1.0 ยิ่งใกล้ 1.0 ยิ่งมีความถูกต้องมากขึ้น

หากต้องการทราบข้อมูลเพิ่มเติม สามารถเปิดดูได้ที่ website

http://www.talula.demon.co.uk/allegro/

และสามารถที่จะทำการ download ได้ที่

http://sourceforge.net/project/showfiles.php?group_id=5665&package_id=168871

- Snaperhelper

เป็น library ที่ช่วยในการ capture ภาพบน screen โดยที่สามารถที่จะทำการ capture เป็นแบบ region หรือ capture เป็น window ได้ ซึ่ง library ตัวนี้ไม่ได้เป็น open source เวลาใช้งานนั้นจะแถม file .lib กับ file .dll มาให้สองตัว เราใช้ file .lib เอาไว้เวลาทำการ compile และ link ใน visual studio 2005 และ .dll นั้นมีไว้เรียกใช้ โดยต้องอยู่ใน directory เดียวกับ file execute ของ project ของเราด้วย

สามารถดูเนื้อหาเพิ่มเติมเกี่ยวกับ library ตัวนี้ และ load มาใช้งานได้ที่

http://www.codeproject.com/tools/screen_snaper.asp

- Eztwain

เช่น library ที่เราใช้ติดต่อกับ scanner ทั้งนี้ตัว library ตัวนี้จะทำการไปเรียกตัว library ของ twain อีกทีนึง ทั้งนี้เนื่องจากการเขียนโปรแกรมเพื่อติดต่อกับตัว twain โดยตรงนั้นค่อนข้างจะยุ่งยาก [illegible] eztwain มาเป็นส่วนที่ห่อหุ้ม twain ไว้ ทั้งนี้เพื่อให้การเขียนโปรแกรมติดต่อกับ scanner นั้นทำได้ง่าย และมีความรวดเร็ว เพื่อเป็นการประหยัดเวลาในการพัฒนา project นี้อีกด้วย

สำหรับข้อมูลเพิ่มเติมสามารถดูได้ที่

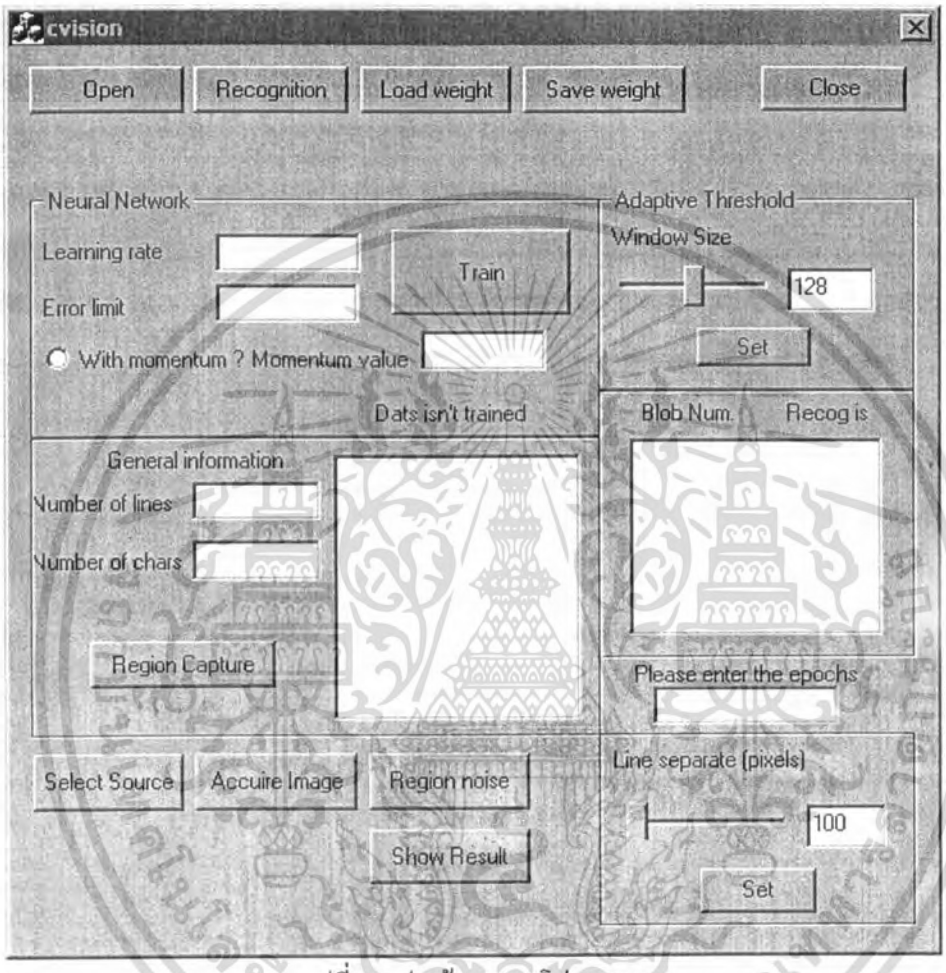
http://www.dosadi.com/eztwain1.htm

หากต้องการ download library ตัวนี้มาใช้โดยที่เป็นเวอร์ชันที่ไม่ต้องเสียเงินด้วย สามารถ download ได้ที่

http://www.dosadi.com/download.htm แล้วก็เลือกตัวที่เป็น freeware ด้วย

คู่มือการใช้งานโปรแกรม

เมื่อเปิดโปรแกรมเข้ามาครั้งแรกจะปรากฏดังภาพด้านล่าง



รูปที่ ๑ รูปหน้าตาของโปรแกรม

<u>- สำหรับแปลงภาพให้เป็นตัวอักษร</u>

- กดที่ปุ่ม Load weight เพื่อเป็นการโหลดน้ำหนักของเส้นเชื่อมเข้ามาในส่วนของ neural network เพื่อให้สามารถใช้ neural network เพื่อการแยะแยะตัวอักษรที่ถูกต้องได้

- รับภาพเข้ามาในโปรแกรม โดยสามารถทำได้สองวิธีคือ

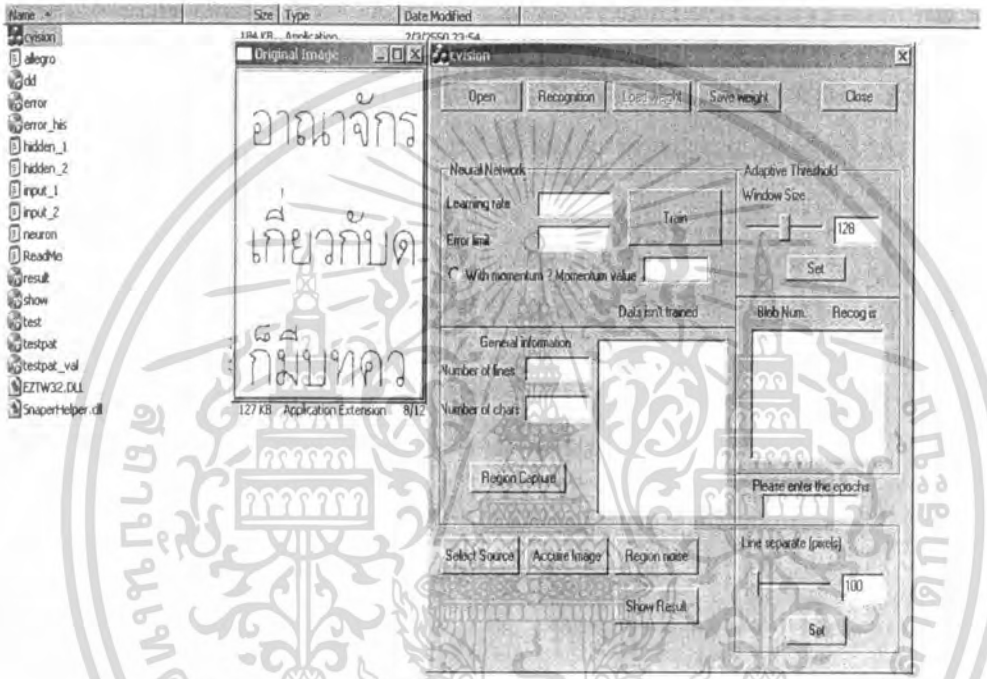
วิธีแรก

1. รับภาพจาก Scanner โดยสามารถที่จะเลือกชนิดของ Scanner ที่ติดต่ออยู่กับเครื่องคอมพิวเตอร์ได้ โดยกดที่ปุ่ม Select Source

2. เมื่อเลือกชนิด Scanner เสร็จแล้วให้กดที่ปุ่ม Acquire Image เพื่อ ให้ Scanner ทำการ scan ภาพเข้ามาในโปรแกรมของเรา ก็จะปรากฏภาพที่ผ่านการ scan แล้วให้เราได้เห็นด้วย

วิธีที่สอง

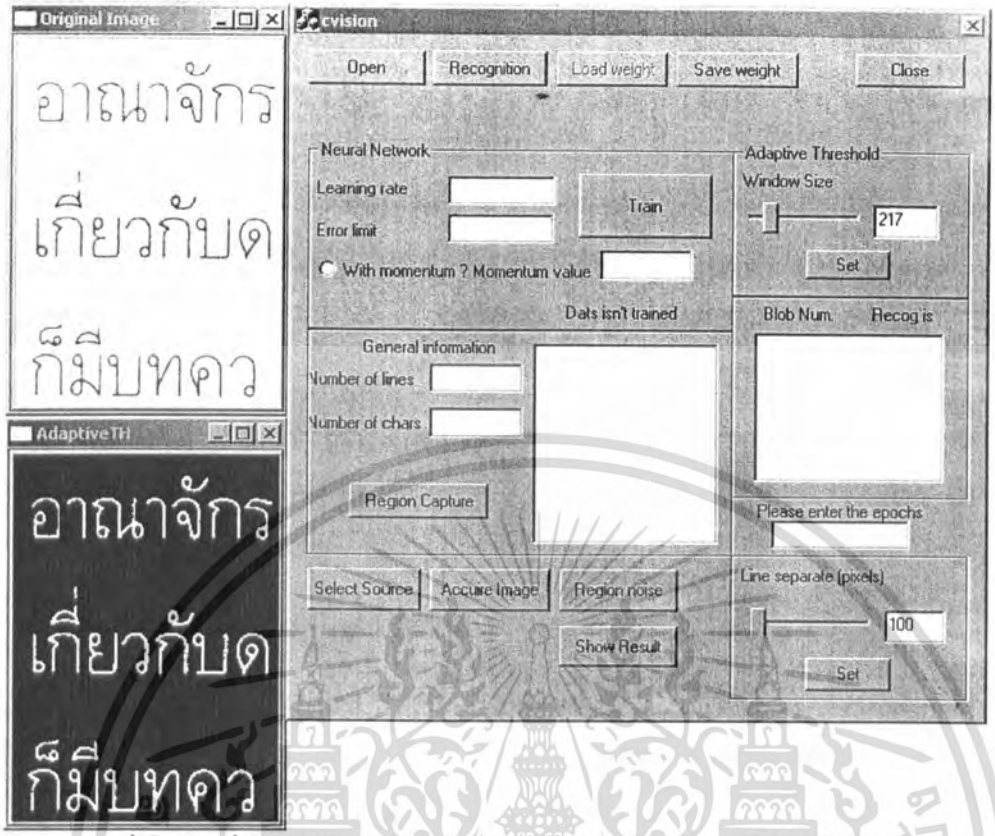
ให้กดที่ปุ่ม Open ก็จะปรากฏ Dialog สำหรับเลือกรูปที่จะเอาเข้ามา process ในโปรแกรม ของเรา ในที่นี้ให้เลือกไฟล์รูปที่เป็นแบบ Bitmap (*.bmp) เท่านั้น จากนั้นก็จะปรากฏภาพที่เราได้เลือกมาแล้วให้เราได้เห็นด้วย ดังข้างล่าง



รูปที่ ๒ รูปหลังจากที่ทำการเปิดไฟล์ภาพที่จะให้โปรแกรมวิเคราะห์

เราสามารถเลือกเฉพาะบริเวณภายในรูปภาพเพื่อเอามาประมวลผลได้ โดยการกดที่ปุ่ม Region Capture แล้วก็จากนั้นก็ไปกดเลือกบริเวณที่ต้องการจะเอามาประมวลผลที่อยู่ภายในขอบเขตของ หน้าต่าง Original Image

เมื่อเราเลือกรูปภาพเข้ามาแล้ว จากนั้นเราก็ต้องทำการเซตค่าของ Threshold เพื่อทำให้ได้ภาพที่เป็นแบบไบนารี เพื่อที่จะเอาไปใช้ในกระบวนการอื่น ๆ ต่อไป ให้เราทำการเลื่อนตัว slider เพื่อเป็นตัวกำหนดค่าของการ threshold ในที่นี้โปรแกรมจะอนุญาติให้กำหนดค่า threshold ได้เฉพาะที่เป็นเลขคี่เท่านั้น เมื่อเลือกค่าได้แล้วให้กดที่ปุ่ม set โปรแกรมก็จะแสดงภาพแบบไบนารี ที่ได้มาจากกระบวนการ threshold ภาพตั้งต้นด้วยค่าที่กำหนดโดยผู้ใช้เอง ให้เราดูภาพที่ได้จากการ threshold โดยที่ให้ตัวอักษรที่แสดงด้วยสีขาวนั้นเห็นได้เต็มตัวของมัน จะทำให้การแปลงเป็นตัวอักษรมีความถูกต้องมากขึ้น

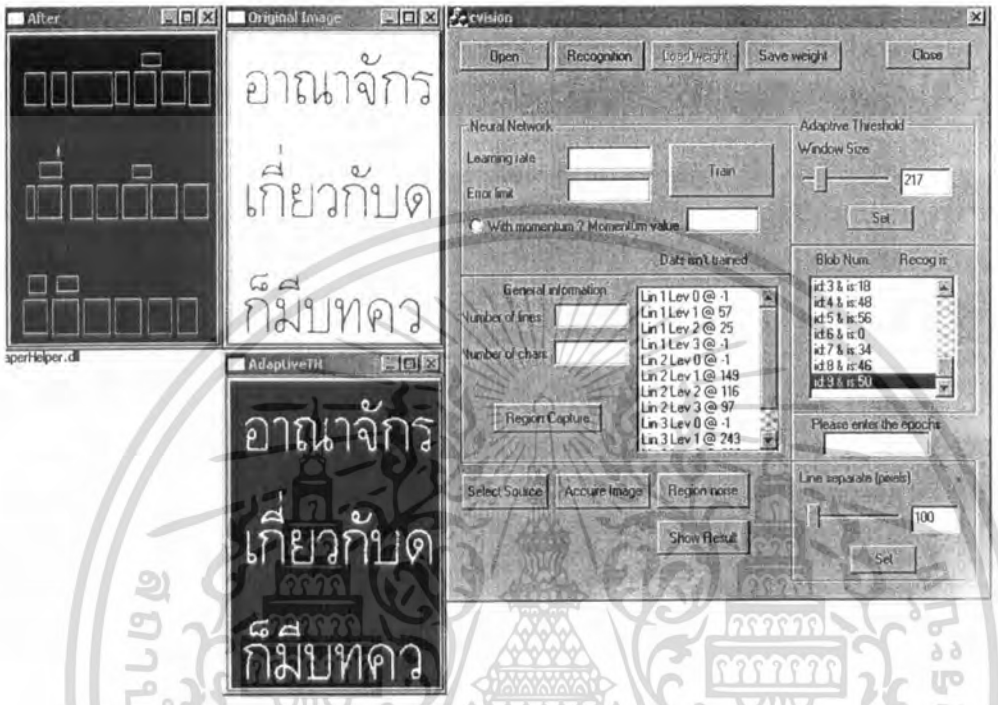


รูปที่ ๓ ภาพที่ได้หลังจากทำการกดที่ปุ่ม Set ก็จะได้ภาพแบบไบนารีออกมา

รูปที่ ๔ ตัวอย่างรูปแบบไบนารีที่ควรจะทำการกำหนดค่า Threshold ใหม่ เพราะตัวอักษรขาด

รูปที่ ๕ ตัวอย่างรูปแบบไบนารีที่กำหนดค่าของ Threshold ไว้ดีแล้ว

หลังจากที่เราทำการ Threshold จนได้รูปที่เป็นไบนารีออกมาแล้ว ให้เราทำการกดที่ปุ่ม Recognition เพื่อให้โปรแกรมทำการวิเคราะห์รูปที่เป็นแบบไบนารีออกมาเป็นข้อความที่สอดคล้องกับภาพ (เป็น text file) ซึ่งโปรแกรมนี้จะทำการแสดงบริเวณของตัวอักษรที่โปรแกรมสามารถแยกออกมาได้ ดังรูปด้านล่างนี้



รูปที่ ๖ บริเวณของตัวอักษรต่าง ๆ ที่โปรแกรมแยกแยะออกมาได้แสดงได้ด้วยสี่เหลี่ยมสีเขียว

หลังจากที่เรากดปุ่ม Recognition เสร็จแล้วและขึ้นหน้าต่าง After มาแล้วขั้นตอนต่อไปให้เราดูผลลัพธ์ที่โปรแกรมนี้ให้ออกมาโดยกดที่ปุ่ม Show Result โปรแกรมก็จะไปทำการเปิด ไฟล์ที่ได้เขียนเป็นข้อความที่สอดคล้องกับภาพไว้ โดยใช้โปรแกรม Notepad ซึ่งก็แสดงให้เห็นดังรูปด้านล่างนี้

รูปที่ ๗ ไฟล์ result.txt ซึ่งเป็นไฟล์ผลลัพธ์สุดท้ายที่ได้จากการทำงานของโปรแกม

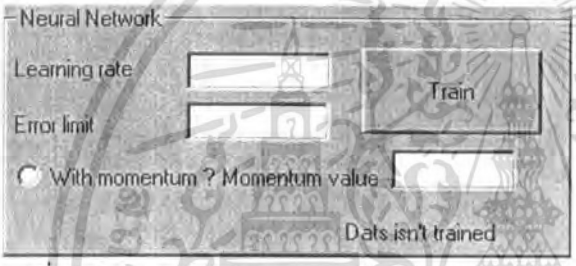
ปุ่มอื่น ๆ ที่มีไว้ช่วยการทำงานของโปรแกรม

ปุ่ม Region Noise เป็นปุ่มที่เอาไว้สำหรับการลบบริเวณที่เราไม่ต้องการที่อยู่บนภาพที่เป็นแบบไบนารี ซึ่งบางทีเราก็อาจจะมีความจำเป็นที่ต้องลบบริเวณบางบริเวณที่ไม่ต้องการที่อยู่บนภาพไบนารี ทั้งนี้ก็เพื่อทำให้กระบวนการ Recognition นั้นทำได้อย่างมีประสิทธิภาพ และไม่ทำให้โปรแกรมเกิด error ด้วย

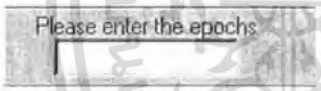
ในช่อง Line separate (pixels) ช่องนี้จะมีไว้กำหนดระยะห่างเป็น pixels ของแต่ละบรรทัดที่อยู่ในไฟล์ภาพตั้งต้นที่เรารับเข้ามา ในกรณีที่โปรแกรมให้ผลลัพธ์การทำงานที่ผิดพลาด เช่น ให้ผลลัพธ์ออกมาเป็นจำนวนบรรทัดที่ไม่ถูกต้อง หรือว่า ให้ผลลัพธ์เป็นไฟล์ข้อความออกมาแต่ไม่สามารถที่ให้ตัวอักษรที่ตรงกับภาพได้เกิน 50 เปอร์เซนต์

<u>- สำหรับการ Train ข้อมูล</u>

ช่องที่เกี่ยวข้องกับการ Train ข้อมูลโดยใช้ neural network มีดังนี้



รูปที่ ๘ ส่วนของ neural network ในโปรแกรม



รูปที่ ๙ ช่องที่ให้กำหนดจำนวน epochs

<u>การ Train ข้อมูลใหม่นั้นให้เราทำดังนี้</u>

ในโครงงานนี้จะใช้การ train แบบที่อาศัย training set และ validation set และค่าที่เราต้องทำการกำหนดให้กับส่วนของการ train โดย neural network มีดังนี้

Learning rate -> เอาไว้สำหรับกำหนดค่าของ learning rate ใส่ค่าที่เป็นทศนิยมได้อย่างเช่น 0.2 เป็นต้น

Error limit -> เอาไว้สำหรับกำหนดเปอร์เซนต์ความถูกต้องของการ train ข้อมูล เปอร์เซนต์ความถูกต้องนั้นจะมีค่าอยู่ระหว่าง 0.0 – 1.0 เราควรกำหนดค่าของเปอร์เซนต์ความถูกต้องให้อยู่ที่ประมาณ 0.9 ขึ้นไป ทั้งนี้ถ้าหากว่าเรากำหนดค่าของเปอร์เซนต์ความถูกต้องมากเกินไปก็อาจทำให้การ train นั้นใช้เวลานานตามไปด้วย แต่ถ้าหากกำหนดน้อยเกินไปก็ทำให้ข้อมูลที่จะใช้ในการ recognize นั้นไม่ค่อยมีความถูกต้องเท่าที่ควร ตัวอย่างการกำหนดค่าที่เหมาะสมก็อย่างเช่น 0.94, 0.95 เป็นต้น

เราสามารถที่จะกำหนดให้การ train นั้นใช้ momentum หรือไม่ใช้ก็ได้ ซึ่งถ้าต้องการที่จะ train โดยไม่ใช้ momentum เราก็ไม่ต้องทำการเลือกตรงปุ่ม radio button ที่เขียนว่า With momentum ? แต่ว่าถ้าหากต้องการ train โดยใช้ momentum นั้นก็ให้เราทำการเลือกตรง radio button ด้วยแล้วก็กำหนดค่า momentum ตรงช่องว่างที่อยู่ข้างกับ label ที่เขียนว่า Momentum value โดยทั่วไปนั้นค่าของ momentum ที่เหมาะสมก็จะอยู่ที่ประมาณ 0.9 ขึ้นไป