การรู้จำตัวอักษรภาษาไทย

OPTICAL THAI CHARACTER RECOGNITION

โดย

นางสาว ฐิติรัตน์ วิยะรัตน์

นาย วศิน รังษีกาญจน์ส่อง

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

การรู้จำตัวอักษรภาษาไทย

OPTICAL THAI CHARACTER RECOGNITION

โดย

นางสาว ฐิติรัตน์ วิยะรัตน์

นาย วศิน รังษีกาญจน์ส่อง

อาจารย์ที่ปรึกษา

อาจารย์ สมเกียรติ วังศิริพิทักษ์

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชา วิศวกรรมศาสตร์คอมพิวเตอร์

คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การรู้จำตัวอักษรภาษาไทย

(OPTICAL THAI CHARACTER RECOGNITION)

ผู้จัดทำ

1. นางสาว ฐิติรัตน์	วิยะรัตน์	39014149
2. นายวศิน	รังษีกาญจน์ส่อง	39014466

อาจารย์ที่ปรึกษา

(อาจารย์ สมเกียรติ วังศิริพิทักษ์)

การรู้จำตัวอักษรภาษาไทย

(OPTICAL THAI CHARACTER RECOGNITION)

โดย นางสาว ฐิติรัตน์ วิยะรัตน์ 39014149
นายวศิน รังษีกาญจน์ส่อง 39014466
อ.สมเกียรติ วังศิริพิทักษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้มีวัตถุประสงค์ในการเสนอการรู้จำตัวอักษรตัวพิมพ์ภาษาไทย รวมถึงขั้นตอนเบื้องต้นก่อนการรู้จำ ระบบการรู้จำนี้ออกแบบขึ้นสำหรับใช้กับรูปภาพบิทแมบที่ได้จากเครื่องแสกนเนอร์ ขั้นตอนเบื้องต้นก่อนการรู้จำจะประกอบด้วย การแปลงข้อมูลภาพจากข้อมูลภาพที่ได้จากเครื่องแสกนเนอร์ซึ่งใช้ระบบสี RGB โดยรหัสสีแต่ละตัวแทนด้วยเลขฐานสิบหก 2 บิท ให้เป็นภาพขาวดำ การตั้งค่ากั้นหน้า กั้นหลัง ของหน้าเอกสาร การหาเส้นบรรทัดของตัวอักษร การตัดคอลัมน์ และการหาขนาดของตัวอักษรโดยอัตโนมัติ สำหรับการรู้จำเราใช้วิธีการนำตัวอักษรต้นแบบมาเปรียบเทียบกับตัวอักษรที่เป็นข้อมูลภาพ เสมือนว่าจะนำภาพมาซ้อนทับกัน โดยเทียบจากจุดเริ่มต้นของตัวอักษรและสระแต่ละตัว โดยเริ่มพิจารณาจากตัวอักษร สระล่าง สระบน ตามลำดับ เพื่อวิเคราะห์ว่าอินพุทที่นำมาทำการรู้จำนั้นคืออักษรตัวใด

Abstract

This thesis presents an automatic optical character recognition system for hand-printed Thai characters, including its pre-processes. The system is designed for bitmap images obtained from optical scaner. The pre-processes consist of binarization of the input image, setting left and right margin, forming baseline and column ,and finding font size automatically. For recognition method, we compare the character's image derived from the pre-process with the model characters in the knowledge base, considering base character, lower, and upper orderly, to identify which character the image represents.

สารบัญ

สารบัญ(ต่อ)

สารบัญรูปประกอบ

สารบัญรูปประกอบ(ต่อ)

สารบัญรูปประกอบ(ต่อ)

สารบัญตารางประกอบ

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เมื่อทำการเปรียบเทียบระหว่างส่วนที่รับข้อมูลของเครื่องคอมพิวเตอร์และของมนุษย์ ส่วนรับรู้ข้อมูลของมนุษย์จะมีความสามารถสูงกว่าคอมพิวเตอร์มากไม่ว่าจะเป็นการรับรู้การเห็นภาพ การได้ยินการสัมผัส การได้กลิ่น หรือการรู้รส ดังนั้น จึงได้มีผู้พยายามประดิษฐ์ คิดค้น และพัฒนาส่วนรับรู้ของเครื่องคอมพิวเตอร์ให้ทัดเทียมกับมนุษย์ ไม่ว่าจะเป็นการพัฒนาให้เครื่องคอมพิวเตอร์สามารถรู้จำข้อมูลทางภาพ (Pattern Recognition) หรือทางเสียง (Voice Recognition) เลียนแบบความสามารถของมนุษย์

การรู้จำตัวอักษร(Character Recognition) เป็นสาขาหนึ่งของการรู้จำภาพ การนำเทคนิคการรู้จำตัวอักษรมาประยุกต์ใช้กับเครื่องคอมพิวเตอร์ เป็นการพัฒนาให้เครื่องคอมพิวเตอร์สามารถเข้าใจภาษาเขียนที่มนุษย์ใช้ติดต่อสื่อสารกันได้ ซึ่งจะมีผลทำให้สามารถนำเครื่องคอมพิวเตอร์มาใช้งานในการแยกแยะตัวอักษรจากภาษาเขียนแทนมนุษย์ได้ ซึ่งการประมวลผลด้วยเครื่องคอมพิวเตอร์มีความสะดวกรวดเร็ว ถูกต้อง แม่นยำและประหยัดค่าใช้จ่ายมากกว่า เทคนิคการรู้จำตัวอักษรด้วยเครื่องคอมพิวเตอร์นี้ยังสามารถนำไปใช้กับงานวิจัยอื่นๆได้ เช่น การสร้างอุปกรณ์สำหรับช่วยคนพิการตาบอด การตรวจสอบความถูกต้องเอกสารทางธุรกิจ เป็นต้น

1.2 วัตถุประสงค์ของการศึกษา

เพื่อศึกษาถึงกรรมวิธีของการรู้จำตัวอักษรภาษาไทยในขณะทำการอินพุทข้อมูลโดยผ่านอุปกรณ์นำข้อมูลเข้าแบบพิกเซล ได้อย่างถูกต้อง แม่นยำ และรวดเร็ว นอกจากนี้กรรมวิธีที่ใช้จะต้องไม่ขึ้นกับขนาดของตัวอักษร ตำแหน่งเริ่มต้นของตัวอักษร

จุดมุ่งหมายของปริญญานิพนธ์นี้คือ ต้องการจัดแยกตัวอักษรเพื่อที่จะนำเอาตัวอักษรที่ได้นี้เข้าสู่กระบวนการรู้จำตัวอักษรต่อไป ซึ่งเป็นการพัฒนาให้เครื่องคอมพิวเตอร์มีการทำงานที่มีความเฉลียวฉลาดมากขึ้น การป้อนข้อมูลจากหน้าเอกสารเข้าเครื่องคอมพิวเตอร์ก็จะได้สะดวกมากยิ่งขึ้น

1.3 ทฤษฎีหรือแนวคิดที่นำมาใช้

กระบวนการการรู้จำตัวอักษรโดยใช้วิธีขั้นพื้นฐาน ประกอบด้วยวิธีการหา Base line และ การหาแนวของคอลัมน์ เพื่อใช้หาจุดเริ่มต้น(Origin) ของตัวอักษรซึ่งจะเป็นจุดอ้างอิงในการเปรียบเทียบรูปภาพตัวอักษรกับตัวอักษรต้นแบบ (Pattern) ต่อไปโดยในการเพิ่มประสิทธิภาพของการรู้จำตัวอักษรจะต้องมีสัญญาณรบกวนเป็นจำนวนน้อยมาก

1.4 ขอบเขตของโครงงาน

โครงงานนี้มุ่งพัฒนาทางด้านการตรวจรู้อักษรตัวพิมพ์ภาษาไทย โดยแบ่งออกเป็น 2 ส่วน คือ

1. ส่วนที่แปลงตัวอักษรที่ได้จากการสแกน(scan) เข้ามาด้วยเครื่องสแกนเนอร์(Scanner) ซึ่งจะพบว่าตัวอักษรต่างๆจะอยู่รวมกันเป็นประโยค ดังนั้นจะต้องทำการแยกให้อยู่ในรูปตัวอักษรตัวเดี่ยวๆแต่ละรูปก่อนจึงจะสามารถนำไปประมวลผลในขั้นต่อไปได้

2. ทำการเปรียบเทียบรูปภาพของตัวอักษรกับตัวอักษรต้นแบบที่โหลดมาเก็บไว้ในหน่วยความจำ เพื่อตรวจดูว่าข้อมูลรูปภาพตัวอักษรส่วนนี้ ตรงกับตัวอักษรใด

1.5 วิธีที่ใช้ในการดำเนินการ

ในโครงงานนี้ได้ทำการทดสอบโปรแกรมด้วยการเขียนภาษา Delphi ให้ทำงานบนไมโครคอมพิวเตอร์ ระบบปฏิบัติการวินโดวส์ 9x (Windows 9x) เริ่มต้นด้วยการเก็บข้อมูลโดยสแกนตัวอักษรจากเอกสารแล้วนำเปรียบเทียบกับตัวอักษรต้นแบบ (Pattern) ที่มีอยู่ในหน่วยความจำ โปรแกรมที่ทำการทดสอบนี้ได้ดีทำงานบนเครื่องที่ใช้ CPU Pentium ความเร็วอย่างต่ำ 200 MHz หน่วยความจำอย่างน้อย 32 MB

1.6 ผลงานวิจัยที่ผ่านมา

ที่ผ่านมาได้มีการวิจัยทางด้านการรู้จำตัวอักษรภาษาไทยหลายวิธี ซึ่งมีดังนี้

1. ผลงานวิจัยเรื่อง " A Recognition Method Of Handprinted Thai Character by local Feature" โดยใช้หลักการในการวิเคราะห์ฟีเจอร์พื้นฐาน(local feature) ของตัวอักษรภาษาไทย โดยพิจารณาจากทิศทางที่เปลี่ยนไปของการลากเส้นขึ้นมาของตัวอักษร เพื่อวิเคราะห์หาส่วนเว้า(Concavity) และส่วนโค้งนูน(Convexity) บนตัวอักษร

2. งานวิจัยเรื่อง "การรู้จำตัวอักษรลายมือเขียนไทย" (An online Recognition Method for Hand Written Thai Characters) โดยนายยุบรรพจน์ พรมคุณและ นายวรายุทย์ เหมวิจิตร คณะวิศวกรรมคอมพิวเตอร์ สถาบันพระจอมเกล้าเจ้าคุณทหารลาดกระบังปี 2542 เป็นการพิจารณาแบบ online โดยใช้หลักการแทนทิศทางการลากเส้นตัวอักษรด้วยรหัสลูกโซ่ (chain code) แล้วเปรียบเทียบกับ rules ใน knowledge base ความถูกต้อง : 79.92%

3. งานวิจัยเรื่อง "การจดจำรูปแบบตัวอักษรคัดลายมือภาษาไทยโดยวิธีแยกลักษณะเด่น" ซึ่งการพิจารณาถึงรูปแบบโครงสร้างของตัวอักษรทั้งลายเส้นและพื้นเบื้องหลังของลายเส้นตัวอักษรควบคู่กันไป ขั้นตอนแบ่งออกเป็น 3 ขั้นตอนใหญ่ๆคือ ขั้นตอนแรกเป็นการเปลี่ยนลักษณะลายเส้นและลักษณะลายพื้นเบื้องหลังลายเส้นเป็นรหัสเบื้องต้น(Initial feature extraction) ขั้นตอนที่สองการลดทอนรหัสเบื้องต้นที่ซ้ำซ้อน(Unification) ขั้นตอนที่สามเป็นการรวมรหัสลายเส้นและลายพื้นเข้าด้วยกันเพื่อให้เป็นคุณสมบัติของตัวอักษร(Concentration) ผลของการรู้จำตัวอักษรแต่ละตัวต้องใช้เวลาประมาณ 3 นาที ซึ่งใช้เวลานานเกินไป จึงไม่สามารถนำมาใช้งานจริงได้

4. ผลงานวิจัยเรื่อง "การจดจำตัวอักษรลายมือเขียนภาษาไทยโดยการพิจารณาหัวตัวอักษร" (Recognition of Hand Written Thai Character Considering the Head of Character) โดย สุรพันธุ์ เอื้อไพบูลย์ คณะวิศวกรรมไฟฟ้า สถาบันพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปี 2531 ใช้วิธีการเลือกพิจารณาส่วนหัวของตัวอักษรภาษาไทยที่มีลักษณะเป็นวงกลมเล็กๆ ขั้นตอนการทำงานแบ่งออกเป็น 4 ขั้นตอนคือ ขั้นตอนที่หนึ่งเป็นการเตรียมภาพตัวอักษรแต่ละตัวไว้ 2 ชุด โดยชุดหนึ่งเป็นภาพปกติ และอีกชุดหนึ่งจะเป็นภาพตัวอักษรที่ผ่านการทำบางมาแล้ว ขั้นตอนที่สอง จะทำการค้นหาจากส่วนหัวของตัวอักษรปกติ และคำนวณหาคุณสมบัติทางโทโพโลยี่จากตัวอักษรที่ถูกทำให้บาง ในขั้นตอนที่สามเป็นการแบ่งตัวอักษรออกเป็นกลุ่มโดยพิจารณาส่วนหัวของตัวอักษร และในขั้นตอนสุดท้าย เป็น

การวิเคราะห์อักษรด้วยเทคนิคพิเศษ 4 เทคนิค ได้แก่ 1) Subhead Region 2) Feature Code 3) Head Style และ 4) Width per height ratio การทำงานของระบบใช้กล้องเก็บภาพ CCD เป็นอุปกรณ์ป้อนข้อมูล(Input device) ซึ่งการใช้งานคงยุ่งยากพอสมควร และต้องพึ่งผู้มีความเชี่ยวชาญเป็นผู้ปรับแต่งความคมชัดในการถ่ายภาพ และในส่วนอัลกอลิทึมที่ใช้ เป็นการพิจารณาเฉพาะส่วนหัวของตัวอักษรซึ่งค่อนข้างจะเป็นการพิจารณาเฉพาะเจาะจงมากเกินไป หากข้อมูลภาพเป็นภาพตัวอักษรที่เขียนแบบไม่มีหัว หรือลายเส้นของหัวขาดไม่สมบูรณ์หรือไม่เป็นวงปิด หรือหัวมีลักษณะทึบตัน ผลของการรู้จำตัวอักษรย่อมผิดพลาดแน่นอน แต่อย่างไรก็ตาม หากทุกอย่างเป็นไปตามเงื่อนไขที่ผู้ทำการวิจัยกำหนดไว้ ในงานวิจัยนี้สามารถรู้จำตัวอักษรลายมือเขียนให้ผลของการรู้จำตัวอักษรลายมือเขียนที่ดีพอสมควรความถูกต้อง : 98.82% เวลาโดยเฉลี่ยที่ใช้ในการรู้จำตัวอักษร 1 ตัว : 2.55 วินาที (บนเครื่อง IBM PC/XT)

5. ผลการวิจัยเรื่อง “การออกแบบพจนานุกรมสำหรับการเรียนรู้ตัวอักษรคัดลายมือไทย-อังกฤษ อัตโนมัติบนเครื่องคอมพิวเตอร์” เป็นการพัฒนาปรับปรุงจากผลงานวิจัยอื่น มีการเพิ่มเติมส่วนของการรู้จำตัวอักษรโดยค้นหาคุณสมบัติเด่นของตัวอักษรจากพจนานุกรมของการรู้จำที่จัดเก็บโดยพิจารณาจากความถี่ของการใช้งานของตัวอักษรแต่ละตัว ทำให้การค้นหาทำได้อย่างรวดเร็ว นอกจากนั้นตัวระบบเองยังสามารถที่จะรู้จำตัวอักษรที่ไม่มีข้อมูลในพจนานุกรมได้โดยอัตโนมัติโดยอาจให้ผู้ใช้ช่วยตัดสินใจบ้าง ส่วนของพจนานุกรมสามารถเพิ่มเติมตัวเองได้โดยอัตโนมัติเช่นกัน ประสิทธิภาพการรู้จำถูกปรับปรุงให้ดีขึ้นมาก โดยเฉพาะอย่างยิ่งในเรื่องของเวลาที่ใช้ในการรู้จำ แต่ในการเขียนลายมือคัดเพื่อเป็นข้อมูลในการวิเคราะห์ยังจำกัดขนาดตัวอักษร และจำนวนบรรทัดจึงไม่เหมาะที่จะนำไปใช้งานจริง

6. ผลงานวิจัยเรื่อง “การรู้จำตัวอักษรภาษาไทยโดยใช้วิธีการค้นหาลักษณะโครงสร้างของลายเส้น” ได้นำเสนอวิธีการค้นหาลักษณะโครงสร้างลายเส้นซึ่งเป็นลักษณะเด่นของตัวอักษร (Topological feature extraction) การพิจารณากระทำกับโครงสร้างของลายเส้นตัวอักษรที่ถูกแบ่งเป็น 8 ส่วนในแนวตั้ง แนวนอน และในแนวเส้นทะแยงมุมทั้งสอง ในขั้นตอนต่อไปทำการแยกพิจารณาในแต่ละส่วน(Octant)หาคุณสมบติทางโทโพโลยี่ของลายเส้นของตัวอักษรที่ถูกแบ่งทั้ง 8 ส่วน เนื่องจากการพิจารณากระทำกับโครงสร้างของลายเส้นของตัวอักษรอย่างละเอียดจึงสามารถเก็บคุณลักษณะเด่นของตัวอักษรได้ครบถ้วน ส่งผลให้ความถูกต้องของการรู้จำมีสูงเกินกว่า 95% แต่การแบ่งพื้นที่พิจารณาออกเป็นส่วนๆที่มีขนาดพื้นที่เล็กเกินไป จะทำให้การพิจารณาโครงสร้างของลายเส้นในแต่ละส่วนอาจจะไม่ใช่ลายเส้นที่แท้จริงในกรณีที่ตัวอักษรที่มีความหนา แต่แนวทางแก้ไขสามารถทำได้โดยเพิ่มเติมส่วนของการทำตัวอักษรให้บาง ซึ่งจะได้โครงร่างของลายเส้นที่แท้จริง แล้วจึงนำไปพิจารณาก็จะสามารถแก้ปัญหาในส่วนนี้ได้ และคาดว่าจะได้ความถูกต้องแน่นอนแม้ว่ารูปแบบ(font)ของตัวอักษรจะแตกต่างกัน

7. ผลงานวิจัยเรื่อง “ วิธีการรู้จำลายมือเขียนภาษาไทยโดยใช้กรอบมาตราฐาน ” (A Method of the Thai Character Recognition by Using Standard Frame)โดย กฤษฎา ลิมปานนท์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าธนบุรี ปี 2535 เป็นวิธีการรู้จำลายมือเขียนภาษาไทยโดยใช้กรอบมาตราฐาน ซึ่งจะมีกรอบมาตราฐานต่างกันไปแต่ละพยัญชนะหรือสระ อาศัยคุณสมบัติพิเศษของแต่ละตัวพยัญชนะหรือสระมาเป็นตัวอ้างอิงในการสร้างกรอบมาตราฐานแต่ละตัว ซึ่งจะมีชุดรหัส 2 ชุดที่กำหนดให้เป็นกรอบมาตราฐาน ชุดรหัสนี้จะแทนส่วนต่างๆของพยัญชนะหรือสระ เมื่อมีข้อ

มูลภาพของลายมือเขียนภาษาไทยป้อนเข้ามา ก็จะถูกนำมาเปรียบเทียบกับกรอบมาตราฐานที่ถูกสร้างไว้แล้วในคลังข้อมูล (data bank) ทำให้สามารถตัดสินได้ว่าเป็นพยัญชนะหรือสระตัวใดได้
ความถูกต้อง : 80.62%

8. ผลงานวิจัยเรื่อง "การรู้จำตัวลายมือเขียนตัวเลขไทยด้วยนีโอคอกนิตรอน" เป็นวิธีแบบ offline โดยนำจุดเด่นของตัวเลขภาษาไทยบางส่วนมาทำการวิเคราะห์และเรียนรู้ด้วยโครงข่ายประสาทเทียม ในโมเดลนีโอคอกนิตรอน

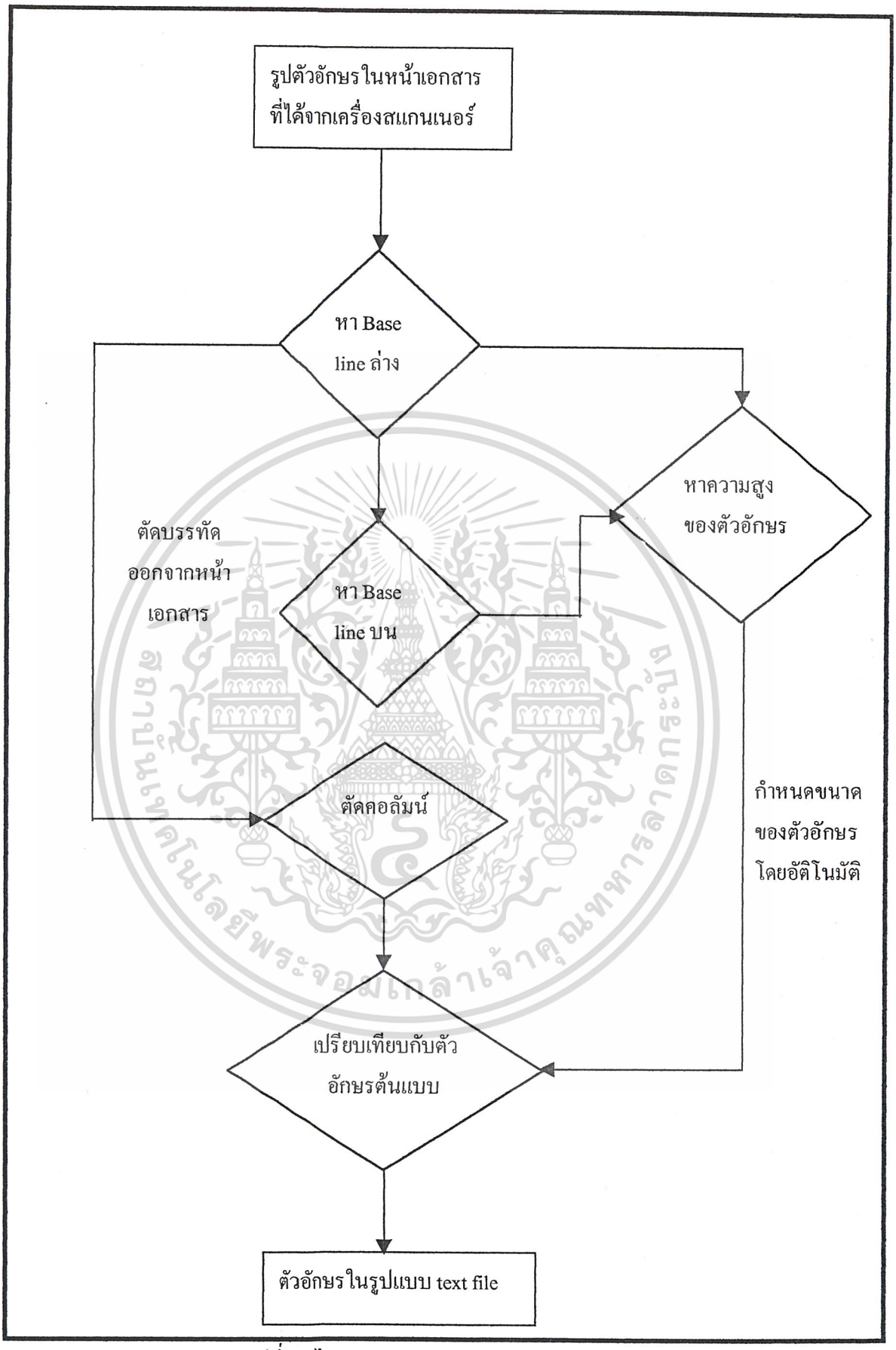
บทที่ 2
ขอบเขตและขั้นตอนการทำงานของระบบ

2.1 ขอบเขตของโครงงานโดยละเอียด

ระบบการรู้จำตัวอักษรภาษาไทย(Thai OCR) ที่พัฒนาขึ้นจะสามารถรับอินพุทขนาดของตัวอักษรได้ตั้งแต่ตัวอักษรขนาด 14 ถึง 24 เนื่องจากถ้ามีตัวอักษรมีขนาดเล็กกว่า 14 ตัวอักษรจะติดกันมากเกินไป ทำให้การประมวลผลของระบบผิดพลาดและถ้ามีขนาดตัวอักษรมากกว่า 24 หน่วยความจำ(RAM) จะไม่พอและทำให้การประมวลผลล่าช้า ชนิดของตัวอักษรที่ระบบเราใช้มีทั้งหมด 10 ชนิด คือ AngsanaUPC, Angsana New, Cordia UPC, Cordia New,Browallia UPC,Browalia New,Dillenia UPC, Jusmine UPC,Lily UPC,KodchiangUPC ซึ่งเป็นชนิดของตัวอักษรที่นิยมใช้ในปัจจุบัน แล้วสแกนโดยเครื่องสแกนเนอร์ที่มีความละเอียดสูง สแกนที่ความละเอียด 75 dpi แฟ้มข้อมูลเป็น Bmp file ในโหมดการทำงาน RGB เนื่องจากทำให้ได้อินพุทที่ได้มีขนาดใกล้เคียงกับตัวอักษรจริงในหน้าเอกสาร จากนั้นเข้าสู่กระบวนการแยกตัวอักษรเพื่อแยกตัวอักษรออกมาเป็นตัวเดี่ยวๆ จึงสามารถนำไปเปรียบเทียบกับตัวอักษรต้นแบบ (pattern) ในหน่วยความจำได้

2.2 ภาพรวมของระบบ

ระบบที่ทำการพัฒนาขึ้นเพื่อใช้ในการรู้จำตัวอักษรภาษาไทยมีหลักการคร่าวๆคือ รับอินพุทที่เป็นรูปภาพของตัวอักษรที่ได้จากการสแกนผ่านเครื่องสแกนเนอร์เข้ามาแล้วสแกนหาBase line ล่างเพื่อใช้ในการตัดบรรทัดออกจากหน้าเอกสาร และหา Base line บนเพื่อใช้หาความสูงของตัวอักษร จากนั้นตัดตัวอักษรออกมาเป็นตัวเดี่ยวๆ เพื่อเตรียมเข้าสู่กระบวนการเปรียบเทียบตัวอักษรอินพุทกับตัวอักษรต้นแบบ (pattern) ออกมาได้เป็นตัวอักษรในรูป text file เขียนแสดงเป็นไดอะแกรมได้ดังต่อไปนี้



รูปที่ 2.1 ไดอะแกรมแสดงการทำงานของระบบ

โดยมีขั้นตอนการทำงานของระบบดังต่อไปนี้

1. โหลดรูปที่สแกนเข้ามา
2. กำหนดค่าชนิดของตัวอักษรเพื่อใช้เป็นตัวอักษรต้นแบบ(pattern) ในการเปรียบเทียบ
3. กำหนดค่าอัตโนมัติให้ระบบ ขนาดตัวอักษร,ความสูงของตัวอักษร,ความสูงของบรรทัด
4. โหลดตัวอักษรต้นแบบ(Pattern) มาเก็บไว้ในหน่วยความจำ
5. สแกนหา Baseline ล่างเพื่อหาเส้นบรรทัด
6. สแกนหา Baseline บนเพื่อหาความสูงของตัวอักษร
7. สแกนหาคอลัมน์(Column)ทีละบรรทัดจะได้ขอบเขตของตัวอักษร(อาจจะมีตัวเดียวหรือหลายตัวปนกัน)
8. เปรียบเทียบขอบเขตของตัวอักษรที่ได้กับตัวอักษรต้นแบบที่เก็บไว้ในหน่วยความจำ
   - 8.1 เปรียบเทียบตัวอักษรระดับกลาง
   - 8.2 เปรียบเทียบตัวอักษรระดับล่าง
   - 8.3 เปรียบเทียบตัวอักษรระดับบนชั้นที่สอง
   - 8.4 เปรียบเทียบตัวอักษรระดับบนชั้นที่หนึ่ง
   - 8.5 ตัดขอบเขตของตัวอักษรที่ทำการเปรียบเทียบแล้วออกไป
   - 8.6 ตรวจสอบขอบเขตที่เหลือว่าเป็นตัวอักษรมากกว่า 1 ตัวหรือไม่
   - 8.7 ถ้ามีตัวอักษรมากกว่า 1 ตัวจะต้องกลับไปทำข้อ 8.1
   - 8.8 เลื่อนคอลัมน์(Column)จนหมดบรรทัด
   - 8.9 เลื่อนบรรทัดจนหมดบรรทัด
9. แสดงผลตัวอักษรออกมาเป็น text file

บทที่ 3

กระบวนการแยกตัวอักษร

กระบวนการแยกตัวอักษร เป็นกระบวนการที่สำคัญอย่างหนึ่งของกระบวนการรู้จำตัวอักษร ซึ่งการแยกตัวอักษรต่างๆออกจากข้อความหรือประโยคให้ออกมาทีละตัวเพื่อที่จะนำตัวอักษรที่ได้จากกระบวนการนี้เข้าสู่กระบวนการรู้จำตัวอักษรต่อไป ซึ่งในกระบวนการรู้จำตัวอักษรจะสามารถประมวลผลได้ครั้งละหนึ่งตัวอักษรเท่านั้น จากข้อมูลที่ใช้จริงจะมีลักษณะเป็นหน้าเอกสาร ซึ่งจะประกอบด้วยตัวอักษรหรือข้อความที่เป็นแถวเป็นบรรทัด ดังนั้นจึงต้องทำการแยกตัวอักษรเหล่านั้นออกมาเป็นข้อมูลตัวอักษรเป็นตัวๆ

ซึ่งในการแยกตัวอักษรออกจากประโยคทีละตัวอักษรนั้น ได้มีการทำการศึกษาและทำการแยกตัวอักษรไว้หลายวิธีด้วยกันเช่น การใช้วิธีการตามขอบภาพ หรือวิธีทางฮิสโตรแกรม ซึ่งวิธีตามขอบภาพและฮิสโตรแกรมนั้นจะสามารถใช้ได้เพียงคร่าวๆเท่านั้นว่าบรรทัดอยู่ในช่วงไหน แต่จะหาเส้นบรรทัดไม่ได้ ทำให้จะมีปัญหาตอนทำการเปรียบเทียบตัวอักษรที่ได้จากการสแกนกับรูปแบบตัวอักษรนั้นๆ เนื่องจากจะหาจุดอ้างอิงในการเปรียบเทียบไม่ได้ ซึ่งในปริญญานิพนธ์ฉบับนี้ได้เลือกใช้วิธีการหาเส้นบรรทัดมาใช้เพราะจะสามารถหาจุดอ้างอิงได้

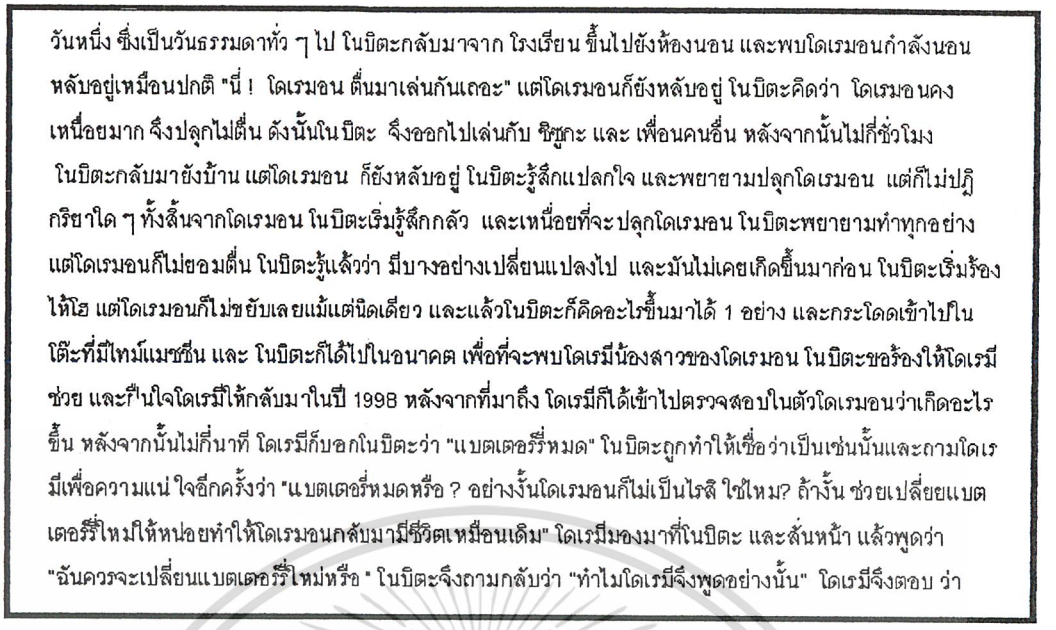
3.1 การอ่านข้อมูลที่ถูกสแกนโดยเครื่องสแกนเนอร์

ภาพที่ได้จากเครื่องสแกนเนอร์ที่ทำการสแกนออกมาจะเก็บไว้อยู่ในรูปของแฟ้มข้อมูลแบบ PSD (Photoshop format) ซึ่งเป็นการนำข้อมูลจากเครื่องสแกนเนอร์มาสร้างเป็นรูปใหม่ ซึ่งลักษณะแฟ้มข้อมูลแบบดังกล่าวจะสนับสนุนโหมดการทำงานของภาพทุกโหมดการทำงาน

3.1.1 การแปลงข้อมูลจาก PSD (Photoshop format) เป็น bmp file

เมื่อรับข้อมูลภาพจากเครื่องสแกนเนอร์จะต้องทำการดัดแปลงข้อมูลจากแฟ้มข้อมูลแบบ PSD (Photoshop format) ที่ได้ให้อยู่ในรูปแบบของ BMP file ในโหมดการทำงานของ RGB ที่สามารถนำไปประมวลผลในขั้นตอนต่อไปได้

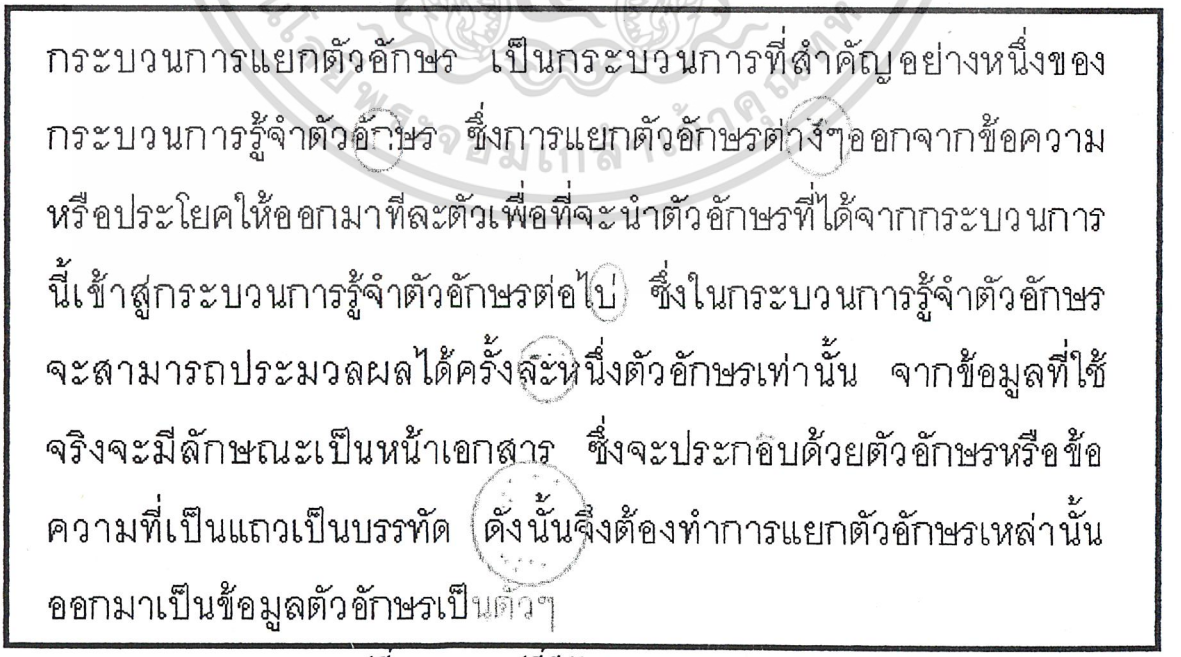
ในข้อมูลภาพที่อ่านได้จากสแกนเนอร์ (scanner) เราจะกำหนดสีของตัวอักษรและสีพื้นหลังของหน้าเอกสาร โดยกำหนดค่าสีของตัวอักษรให้เป็นสีดำ 1 และสีของพื้นหลังเป็นสีขาว 0 แล้วนำไปเก็บไว้ในบล็อคข้อมูลที่ได้จากการสแกนเพื่อใช้เปรียบเทียบกับตัวอักษรต้นแบบ(Pattern) ต่อไป


วันหนึ่ง ซึ่งเป็นวันธรรมดาทั่ว ๆ ไป โนบิตะกลับมาจาก โรงเรียน ขึ้นไปยังห้องนอน และพบโดเรมอนกำลังนอนหลับอยู่เหมือนปกติ "นี่ ! โดเรมอน ตื่นมาเล่นกันเถอะ" แต่โดเรมอนก็ยังหลับอยู่ โนบิตะคิดว่า โดเรมอนคงเหนื่อยมาก จึงปลุกไม่ตื่น ดังนั้นโนบิตะ จึงออกไปเล่นกับ ชิซูกะ และ เพื่อนคนอื่น หลังจากนั้นไม่กี่ชั่วโมง โนบิตะกลับมายังบ้าน แต่โดเรมอน ก็ยังหลับอยู่ โนบิตะรู้สึกแปลกใจ และพยายามปลุกโดเรมอน แต่ก็ไม่ปฏิกริยาใด ๆ ทั้งสิ้นจากโดเรมอน โนบิตะเริ่มรู้สึกกลัว และเหนื่อยที่จะปลุกโดเรมอน โนบิตะพยายามทำทุกอย่าง แต่โดเรมอนก็ไม่ยอมตื่น โนบิตะรู้แล้วว่า มีบางอย่างเปลี่ยนแปลงไป และมันไม่เคยเกิดขึ้นมาก่อน โนบิตะเริ่มร้องไห้โฮ แต่โดเรมอนก็ไม่ขยับเลยแม้แต่นิดเดียว และแล้วโนบิตะก็คิดอะไรขึ้นมาได้ 1 อย่าง และกระโดดเข้าไปในโต๊ะที่มีไทม์แมชชีน และ โนบิตะก็ได้ไปในอนาคต เพื่อที่จะพบโดเรมีน้องสาวของโดเรมอน โนบิตะขอร้องให้โดเรมีช่วย และื่นใจโดเรมีให้กลับมาในปี 1998 หลังจากที่มาถึง โดเรมีก็ได้เข้าไปตรวจสอบในตัวโดเรมอนว่าเกิดอะไรขึ้น หลังจากนั้นไม่กี่นาที โดเรมีก็บอกโนบิตะว่า "แบตเตอรี่หมด" โนบิตะถูกทำให้เชื่อว่าเป็นเช่นนั้นและถามโดเรมีเพื่อความแน่ ใจอีกครั้งว่า "แบตเตอรี่หมดหรือ ? อย่างงั้นโดเรมอนก็ไม่เป็นไรสิ ใช่ไหม? ถ้างั้น ช่วยเปลี่ยยแบตเตอรี่ใหม่ให้หน่อยทำให้โดเรมอนกลับมามีชีวิตเหมือนเดิม" โดเรมีมองมาที่โนบิตะ และลั่นหน้า แล้วพูดว่า "ฉันควรจะเปลี่ยนแบตเตอรี่ใหม่หรือ " โนบิตะจึงถามกลับว่า "ทำไมโดเรมีจึงพูดอย่างนั้น" โดเรมีจึงตอบ ว่า

รูปที่ 3.1 แสดงรูปแฟ้มข้อมูลแบบ BMP ในโหมด RGB

3.2 การกำจัดสัญญาณรบกวน

เนื่องจากเวลาที่ทำการสแกนภาพของแฟ้มข้อมูลตัวอักษรเข้ามานั้น อาจมีสัญญาณรบกวนในรูปแบบต่างๆ เช่น ตัวอักษรขาด หรือมีจุดสีดำ เมื่อเวลานำไปเปรียบเทียบกับตัวอักษรต้นแบบ(Pattern) โปรแกรมจะทำการตัดสัญญาณรบกวนที่เป็นจุดออกไปโดยอัตโนมัติเพราะสัญญาณรบกวนนั้นไม่มีอยู่จริงในตัวอักษรต้นแบบ(Pattern) นอกจากกรณีที่สัญญาณรบกวนนั้นจะเป็นจุดและอยู่ในแนวบรรทัดเดียวกับตัวอักษรระดับกลางเท่านั้น และเนื่องจากในขั้นตอนของการตัดตัวอักษรนั้นจะสามารถกำจัดสัญญาณรบกวนที่มีขนาดเล็กกว่าขนาดของตัวอักษรได้ซึ่งโอกาสจะมีสัญญาณรบกวนที่มีขนาดใหญ่กว่าตัวอักษรน้อยมาก

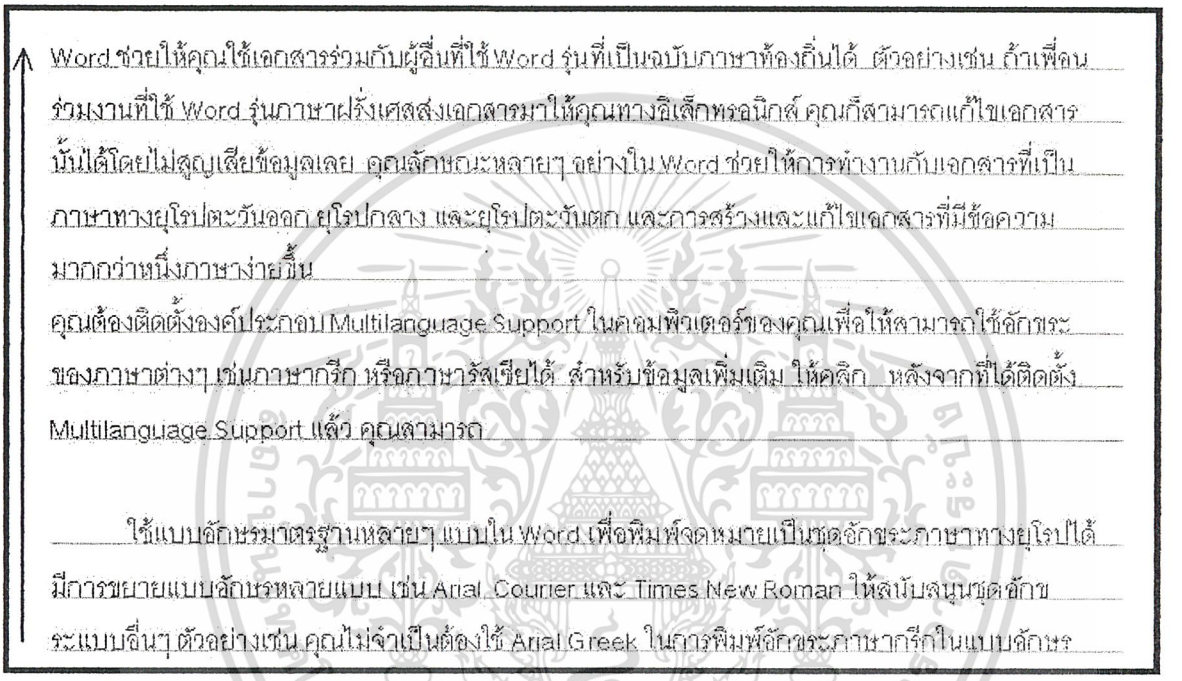

กระบวนการแยกตัวอักษร เป็นกระบวนการที่สำคัญอย่างหนึ่งของกระบวนการรู้จำตัวอักษร ซึ่งการแยกตัวอักษรต่างๆออกจากข้อความหรือประโยคให้ออกมาทีละตัวเพื่อที่จะนำตัวอักษรที่ได้จากกระบวนการนี้เข้าสู่กระบวนการรู้จำตัวอักษรต่อไป ซึ่งในกระบวนการรู้จำตัวอักษรจะสามารถประมวลผลได้ครั้งละหนึ่งตัวอักษรเท่านั้น จากข้อมูลที่ใช้จริงจะมีลักษณะเป็นหน้าเอกสาร ซึ่งจะประกอบด้วยตัวอักษรหรือข้อความที่เป็นแถวเป็นบรรทัด ดังนั้นจึงต้องทำการแยกตัวอักษรเหล่านั้นออกมาเป็นข้อมูลตัวอักษรเป็นตัวๆ

รูปที่ 3.2 แสดงรูปที่มีสัญญาณรบกวน

3.3 การหา Base line

3.3.1 การหา Base line ของตัวอักษรต้นแบบ (Pattern character)

ขั้นตอนการหา Base line ของตัวอักษรต้นแบบ เริ่มจากการสแกนจากด้านล่างขึ้นสู่ด้านบนเนื่องจากตัวอักษรต้นแบบ(Pattern)ไม่มีสัญญาณรบกวน(noise) จนกระทั่งพบสีดำของเนื้อตัวอักษร จะสามารถคำนวณหา Base line ของแต่ละบรรทัดได้จากความสูงเฉลี่ยของตัวอักษรระดับกลางและความสูงเฉลี่ยของบรรทัดเมื่อเทียบกับขนาดของตัวอักษร


Word ช่วยให้คุณใช้เอกสารร่วมกับผู้อื่นที่ใช้ Word รุ่นที่เป็นฉบับภาษาท้องถิ่นได้ ตัวอย่างเช่น ถ้าเพื่อนร่วมงานที่ใช้ Word รุ่นภาษาฝรั่งเศสส่งเอกสารมาให้คุณทางอิเล็กทรอนิกส์ คุณก็สามารถแก้ไขเอกสารนั้นได้โดยไม่สูญเสียข้อมูลเลย คุณลักษณะหลายๆ อย่างใน Word ช่วยให้การทำงานกับเอกสารที่เป็นภาษาทางยุโรปตะวันออก ยุโรปกลาง และยุโรปตะวันตก และการสร้างและแก้ไขเอกสารที่มีข้อความมากกว่าหนึ่งภาษาง่ายขึ้น

คุณต้องติดตั้งองค์ประกอบ Multilanguage Support ในคอมพิวเตอร์ของคุณเพื่อให้สามารถใช้อักขระของภาษาต่างๆ เช่นภาษากรีก หรือภาษารัสเซียได้ สำหรับข้อมูลเพิ่มเติม ให้คลิก หลังจากที่ได้ติดตั้ง Multilanguage Support แล้ว คุณสามารถ

ใช้แบบอักษรมาตรฐานหลายๆ แบบใน Word เพื่อพิมพ์จดหมายเป็นชุดอักขระภาษาทางยุโรปได้ มีการขยายแบบอักษรหลายแบบ เช่น Arial, Courier และ Times New Roman ให้สนับสนุนชุดอักขระแบบอื่นๆ ตัวอย่างเช่น คุณไม่จำเป็นต้องใช้ Arial Greek ในการพิมพ์อักขระภาษากรีกในแบบอักษร

รูปที่ 3.3 แสดงรูปการหา Base line ของตัวอักษร

เมื่อหา Base line ของตัวอักษรต้นแบบได้แล้ว จะทำการนำตัวอักษรต้นแบบทีละตัวไปเก็บเอาไว้พร้อมกับคำนวณหาค่ากั้น(Margin) ของตัวอักษรนั้นด้วยซึ่งจะได้จากค่าความกว้างและความสูงของตัวอักษรแต่ละตัว เพราะการหาค่ากั้น(Margin)นั้น จะใช้เป็นจุดอ้างอิงในการเปรียบเทียบต่อไป

3.3.2 การหา Base line ของข้อมูลที่สแกนเข้ามา

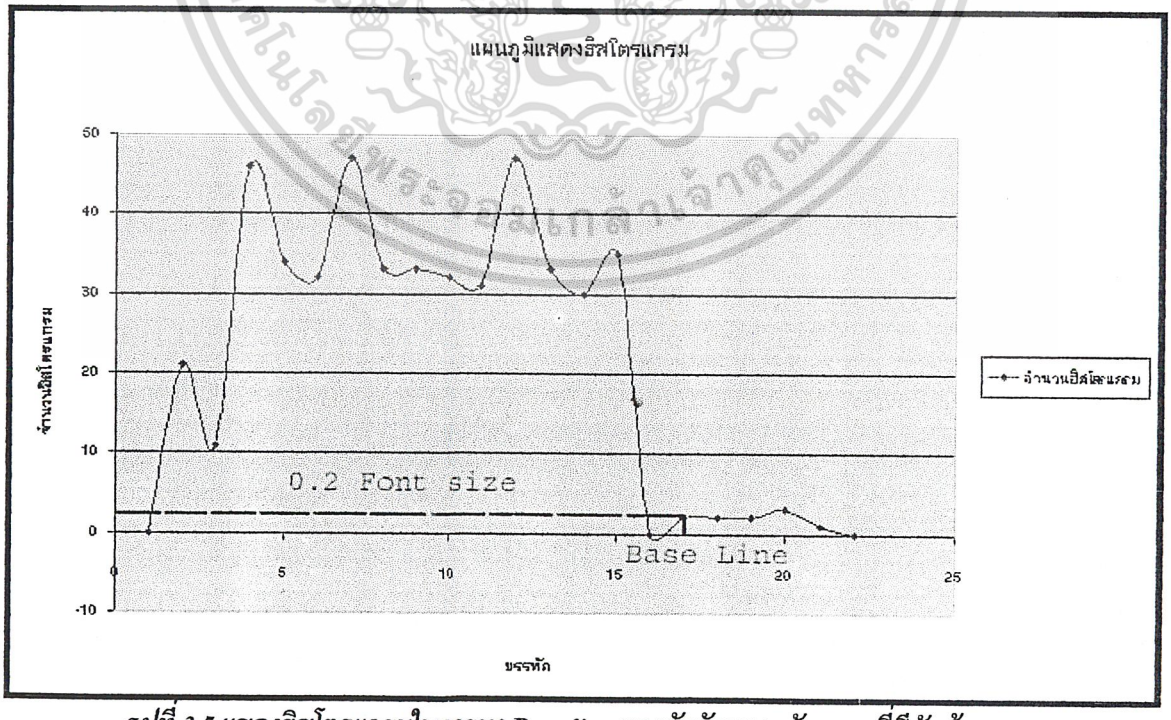
ขั้นตอนการหา Base line จากข้อมูลที่สแกนเข้ามา จะทำการสแกนจากด้านบนลงสู่ด้านล่างเริ่มที่ตำแหน่ง (0,0) ด้านบนซ้ายของหน้าเอกสาร ตำแหน่งที่จะสแกนจะสามารถเปลี่ยนแปลงตามจำนวนคอลัมน์และแถวภายในหน้าเอกสารนั้นๆทำให้ระบบทำงานได้เร็วยิ่งขึ้น จากนั้นจะทำการสแกนโดยจะเริ่มสแกนจากด้านบนลงด้านล่าง ตามที่ค่าที่กำหนดเอาไว้จนกว่าจะเจอเนื้อของตัวอักษรหรือจุดสีดำจุดแรกแล้วจะทำการเปรียบเทียบค่าของจุดสี ณ.ตำแหน่งปัจจุบันกับค่าจุดสีของตำแหน่งที่แล้ว ตามปกติค่าของจุดสีที่จะเป็น Base line จะมีค่าเข้าใกล้ศูนย์ แต่ในบางกรณีการหา Base line ของตัวอักษรเช่น ฎ ฏ ฐ จะมีค่าจุดสีไม่เข้าใกล้ศูนย์ การหา Base line ของตัวอักษรดังกล่าวจะนำค่าของจุดสีของแถวที่แล้วคูณกับค่าคงที่ค่าหนึ่ง(0.2) โดยที่ค่าดังกล่าวได้มาจากค่า

ในเชิงสถิติ แล้วนำผลลัพธ์ที่ได้ไปเปรียบเทียบกับค่าจุดสี ณ.ตำแหน่งแถวปัจจุบัน ซึ่งถ้าค่าจุดสี ณ.ตำแหน่งแถวปัจจุบันมีค่าน้อยกว่าจะต้องพิจารณาเงื่อนไขดังต่อไปนี้

1. ณ.ตำแหน่งแถวนั้น – ตำแหน่งของ Baseline ที่หาได้ในบรรทัดที่แล้วมีค่ามากกว่าหรือเท่ากับความสูงของบรรทัด(LineHigh)
2. และ ณ.ตำแหน่งแถวนั้น – ตำแหน่งที่เป็นสีขาวที่พบครั้งสุดท้ายมีค่ามากกว่าหรือเท่ากับความสูงของตัวอักษร

ถ้าเป็นไปตามเงื่อนไขตามที่กล่าวมา แสดงว่าจุดดังกล่าวคือตำแหน่งของ Base line ในบรรทัดนั้น

รูปที่ 3.4 การหา Baseline ของตัวอักษรระดับกลางที่มีตัวห้อย



รูปที่ 3.5 แสดงฮิสโตรแกรมในการหา Base line ของตัวอักษรระดับกลางที่มีตัวห้อย

ห้อง 507 เทพชัยแมนชั่น เลขที่ 81 ม. 6 ต. นาดี อ.เมือง
จ.สมุทรสาคร 74000
ป.ณ. สมุทรสาคร

สั่งซื้อCDทั้ง11แผ่น2500บาท กรุณาส่งมาที่ พลชาติ ตู้ป.ณ. 207
มหาวิทยาลัยขอนแก่น อ.เมือง จ.ขอนแก่น 40002 แบบพกง.
กรุณาแพคอย่างแน่นหนาด้วยและถ้าเป็นไปได้ช่วยระบุต้นทางหลอกๆ
ว่าจากร้านขายยาอะไรก็ได้
เพราะไปรษณีย์ที่นี่ชอบแกะและขโมย ขอเบอร์ติดต่อกลับด้วยครับ
เบอร์ที่นี่

รูปที่ 3.6 แสดงรูปภาพเพื่อใช้หาฮิสโตรแกรม

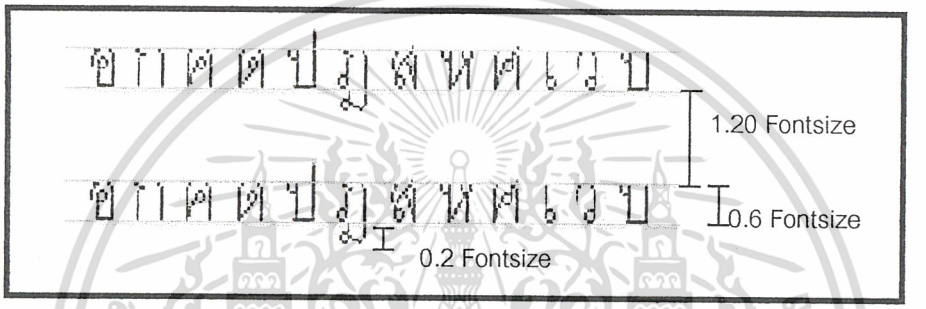
รูปที่ 3.7 รูปแสดงฮิสโตรแกรมรูปภาพตัวอักษรในรูปที่ 3.6

จากรูปที่ 3.6 จะใช้ฮิสโตรแกรมเพื่อพิจารณาสมการที่เหมาะสมในการหา Base line ของแต่ละบรรทัด โดยนำค่าฮิสโตรแกรมที่ได้ไปสร้างเป็นกราฟให้ละเอียดขึ้น โดยจะกล่าวอยู่ในส่วนของภาคผนวกต่อไป

เมื่อเราสามารถหาตำแหน่งของ Base line ในบรรทัดแรกได้แล้วการหา Base line ในบรรทัดถัดไปสามารถหาได้จากการกระโดดข้ามไปสแกนบรรทัดแถวถัดไปโดยมีเงื่อนไขดังต่อไปนี้

1. บรรทัดแถวถัดไปจะมีค่าความสูงระหว่างบรรทัดไม่เกิน ตำแหน่ง Base line บรรทัดแรก + (ความสูงของตัวอักษร*0.2)
2. Base line ของบรรทัดแถวไปมีค่าความสูงไม่เกิน ตำแหน่ง Base line บรรทัดแรก + (ความสูงของตัวอักษร*0.6)
3. บรรทัดที่อยู่ถัดลงไปอีกจะมีความสูงไม่เกิน ตำแหน่ง Base line แรก + (ความสูงของตัวอักษร*0.8)

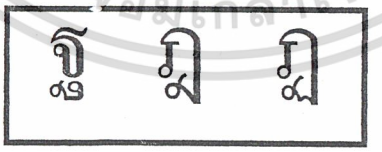
แต่ในกรณีที่ค่าของจุดสี ณ.ตำแหน่งแถวปัจจุบันมีค่ามากกว่า ค่าของจุดสี ณ.ตำแหน่งแถวที่แล้ว*1.2 แสดงว่าบรรทัดที่กำลังหา Base line อยู่นั้นมีตัวอักษรที่มีส่วนบนอยู่เช่น ส,ฮ,พ,ป,ฝ



รูปที่ 3.8 แสดงค่าประมาณความสูงของตัวอักษร,ความกว้างระหว่างบรรทัด และตัวห้อย

3.4 การสแกนคอลัมน์ทั้งหมดในหนึ่งบรรทัด

เริ่มต้นจะต้องกำหนดขอบเขตในการสแกนหาคอลัมน์ในหนึ่งบรรทัดเสียก่อน เมื่อได้ระยะบรรทัดนั้นแล้ว จะสามารถกำหนดให้สแกนหาคอลัมน์ต่อไป วิธีการหาขอบเขตนี้จะช่วยให้โปรแกรมทำงานได้เร็วยิ่งขึ้น การกำหนดขอบเขตจะต้องพิจารณาเผื่อในกรณีที่มีตัวอักษรที่มีส่วนล่างอยู่ในบรรทัดนั้นด้วย ดังนั้นจำเป็นต้องกำหนดขอบเขตการสแกนต่ำกว่าระยะบรรทัดจริง เช่น ฐ ฎ ฏ โดยเฉพาะตัว ฐ ที่มีส่วนล่างล้ำมาทางด้านหน้าของตัวอักษร จะทำให้เวลาที่เราสแกนหาคอลัมน์จริงๆนั้นอาจจะมีปัญหาได้



รูปที่ 3.9 แสดงตัวอักษร ฎ ฏ ฐ

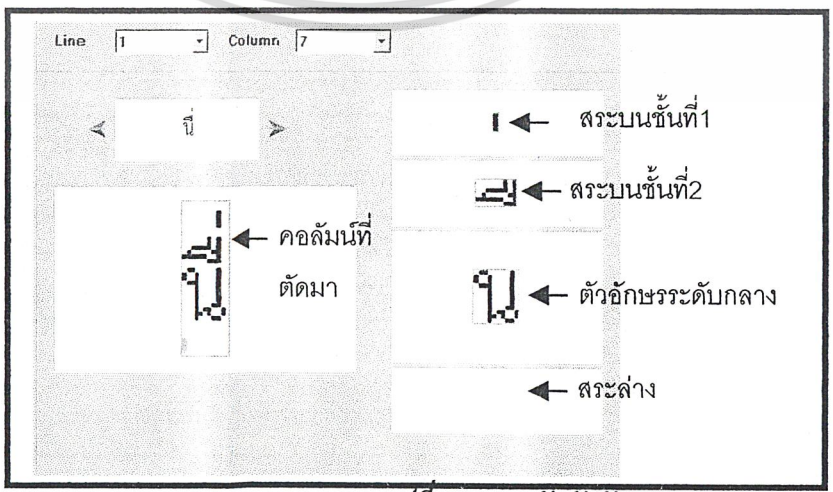
เมื่อได้ขอบเขตในการสแกนหาคอลัมน์แล้ว จะทำการสแกนจากทางด้านซ้ายไปขวา จนกระทั่งเจอจุดสีดำของตัวอักษรแล้วทำการเปรียบเทียบจุดระหว่างตำแหน่งปัจจุบัน กับตำแหน่งที่แล้ว ถ้าตำแหน่งที่แล้วเป็นจุดที่เป็นสีขาวแสดงว่าตำแหน่งปัจจุบันนี้เป็นตำแหน่งของคอลัมน์ของตัวอักษรนั้นๆ

รูปที่ 3.10 แสดงการหาคอลัมน์

3.5 การตัดตัวอักษร

การตัดตัวอักษรจะตัดทีละหนึ่งคอลัมน์ตามที่เราเลือกได้ แต่จะไม่แสดงการเว้นวรรคของข้อความ และจะทำการตรวจสอบหาสระล่างและสระบนตามลำดับเนื่องจากจะแสดงการตัดทีละหนึ่งคอลัมน์ซึ่งประกอบด้วย ตัวอักษร สระบน และสระล่าง เท่านั้น โดยจะแบ่งการแสดงผลออกเป็น 5 ส่วนคือ

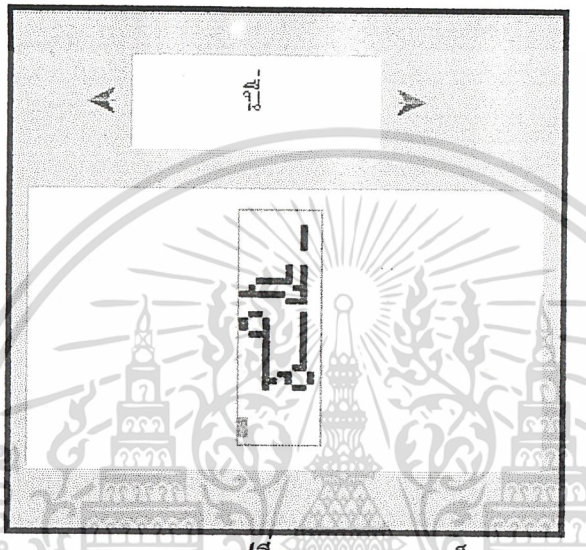
1. ส่วนแรกแสดงบล็อคที่ตัดมาจากคอลัมน์ที่สแกนได้
2. ส่วนที่สองจะแสดงตัวอักษรระดับกลาง
3. ส่วนที่สามจะแสดงสระล่างของตัวอักษร
4. ส่วนที่สี่แสดงสระบนชั้นที่สอง
5. ส่วนที่ห้าแสดงสระบนชั้นที่หนึ่ง



รูปที่ 3.11 การตัดตัวอักษร

การแสดงบล็อกที่ตัดมาจากคอลัมน์ที่สแกนได้

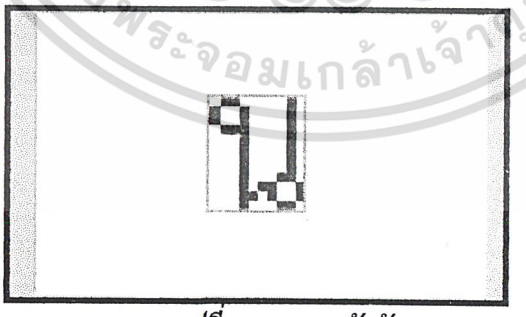
จะต้องกำหนดขอบเขตของการสแกนจากบล็อคข้อมูลต้นฉบับเหมือนกับการสแกนหาคอลัมน์ โดยจะแสดงเฉพาะขอบเขตที่ต้องการเท่านั้น ซึ่งจะต้องเผื่อในกรณีที่มีตัวอักษรหรือสระเยื้องขึ้นมาข้างหน้า เช่น “โ”, “ไ” , “ใ” และสระเยื้องด้านหลัง เช่น ไม้หันอากาศ การันต์ จะต้องหาระยะเยื้องดังกล่าวโดยจะกล่าวรายละเอียดในบทถัดไป ซึ่งระยะเยื้องหน้าและหลังดังกล่าวใช้ในการแยกส่วนตัวอักษรและกระบวนการรู้จำตัวอักษรต่อไป



รูปที่ 3.12 แสดงการบล็อค

การแสดงตัวอักษรตรงกลาง

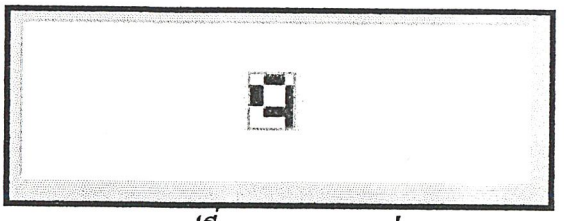
ในกรณีตัวอักษรระดับกลางจะมีปัญหาที่ตัวอักษรที่ล้ำลงข้างล่างและข้างบน เช่น ป,ฎ,ฐ เป็นต้น โดยตัวอักษรระดับกลางจะมีส่วนล้ำขึ้นไปด้านบนไม่เกิน 80 เปอร์เซ็นต์ของความสูงของตัวอักษร และด้านล่างไม่เกิน 50 เปอร์เซ็นต์ของตัวอักษรดังรูปที่ 3.1



รูปที่ 3.13 แสดงตัวอักษรกลาง

การแสดงสระล่างของตัวอักษร

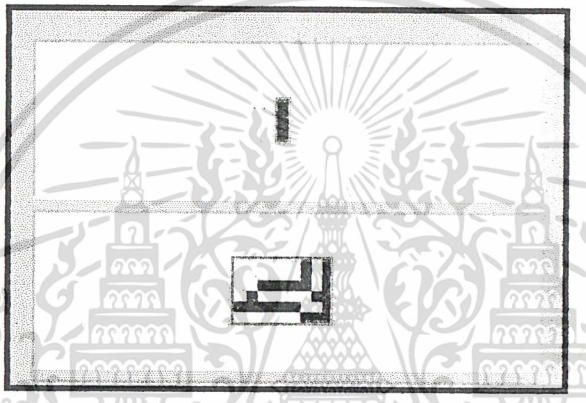
จะแสดงสระล่าง ดังต่อไปนี้



รูปที่ 3.14 แสดงสระล่าง

การแสดงสระบนชั้นที่หนึ่งและชั้นที่สอง

จะแสดงสระบนชั้นหนึ่งและชั้นที่สองดังต่อไปนี้

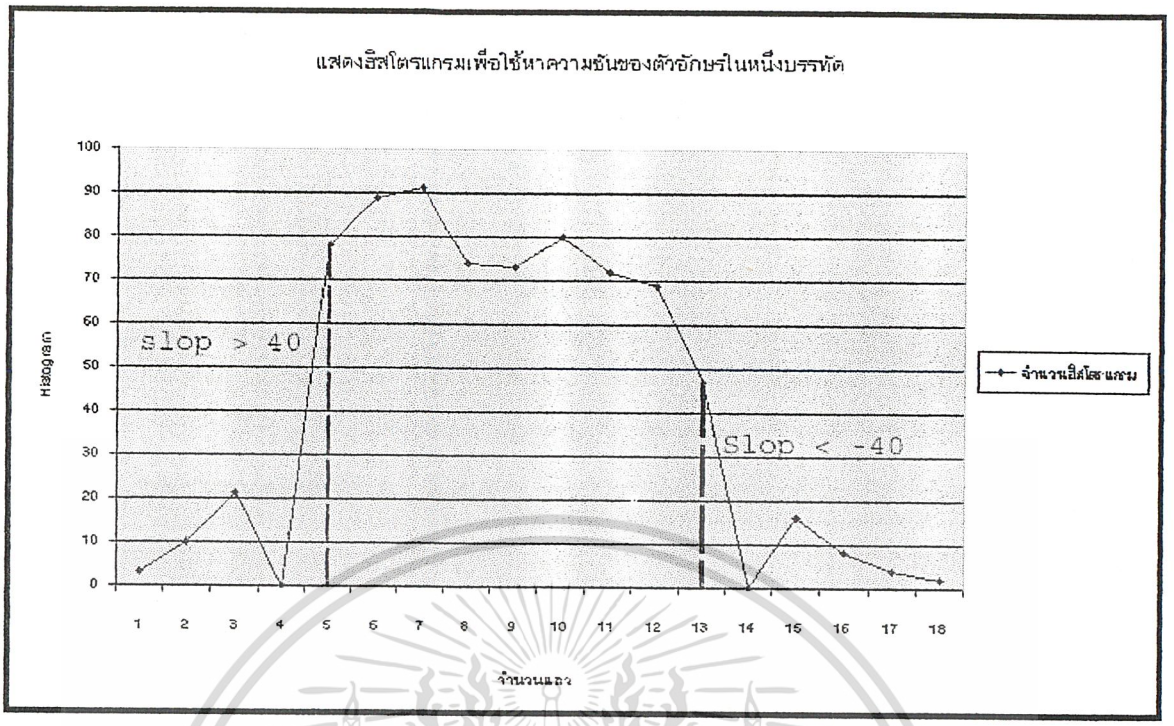


รูปที่ 3.15 แสดงสระบนชั้นที่สอง

3.6 การคำนวณหาความสูงของตัวอักษรโดยอัตโนมัติ

ขั้นตอนการคำนวณหาความสูงของตัวอักษรโดยอัตโนมัติ

เริ่มจากสแกนหาสีดำของเนื้อตัวอักษรของแต่ละบรรทัดแล้วเก็บค่าดังกล่าวไว้ จากนั้นนำค่าที่เก็บไว้ในตอนแรกมาหาความชัน ซึ่งจะได้ค่าความชันในแต่ละแถวออกมา เก็บค่าความชันดังกล่าวไว้ พิจารณาค่าความชันดังกล่าวมากกว่า 40 (ค่าดังกล่าวได้มาจากค่าเฉลี่ยความชันของข้อมูลที่รวบรวบไว้) แสดงว่าตำแหน่งนั้นเป็นตำแหน่งของบรรทัดด้านบน แต่ถ้าค่าความชันดังกล่าวน้อยกว่า –40 แสดงว่าเป็นบรรทัดของสระบนหรือเป็นบรรทัดล่างเมื่อได้ตำแหน่งของบรรทัดด้านบนกับบรรทัดด้านล่างที่ได้จากการหา Base line แล้วจะสามารถคำนวณหาค่าความสูงของตัวอักษรในบรรทัดนั้นๆได้สามารถคำนวณหาความสูงของบรรทัดได้ซึ่งจะมีความสูงเท่ากับความสูงของตัวอักษรนั้นเอง



รูปที่ 3.16 แสดงค่าความชันเพื่อใช้หาบรรทัดบน

รูปที่ 3.17 แสดงรูปของความสูงของตัวอักษร

เมื่อได้ค่าความสูงของตัวอักษรมาแล้ว นำไปเปรียบเทียบเพื่อหาขนาดของตัวอักษรต่อไป โดยจะไล่เปรียบเทียบกับตัวอักษรเมื่อได้ขนาดของตัวอักษรแล้วจะนำค่าที่ได้ไปกำหนดค่าให้กับโปรแกรมโดยอัตโนมัติ

บทที่ 4

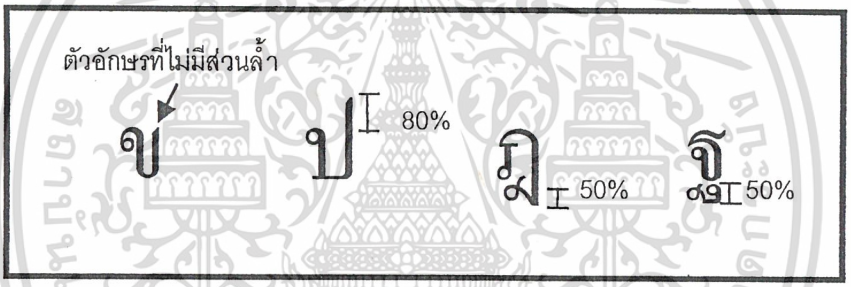
กระบวนการรู้จำตัวอักษร

เมื่อผ่านกระบวนการแยกอักษรออกมาเป็นตัวอักษรเดี่ยวๆได้แล้ว จะเข้าสู่กระบวนการการพิจารณาเปรียบเทียบค่าที่ได้จากการสแกนข้อมูลกับค่าที่เก็บไว้ในตัวอักษรต้นแบบเพื่อดูว่าข้อมูลดังกล่าวควรจะเป็นตัวอักษรอะไร

4.1 ขั้นตอนการเปรียบเทียบตัวอักษร

4.1.1 แนวคิดในการพิจารณาตัวอักษร

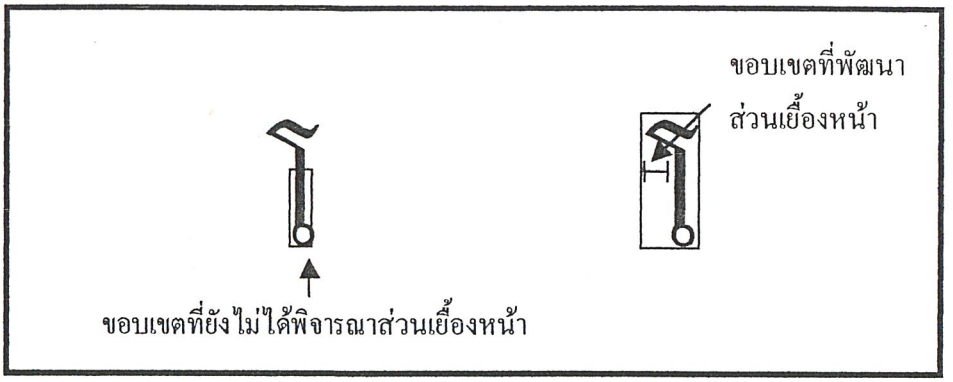
เริ่มต้นจากการสแกนหาขอบบนของบล็อคตัวอักษร และสแกนหาขอบซ้ายของบล็อคตัวอักษร จากนั้นทำการตรวจสอบตัวอักษรที่อยู่กลางบรรทัด ตัวอักษรจะมีส่วนล้ำขึ้นไปด้านบนได้ไม่เกิน 80 เปอร์เซ็นต์ของความสูงตัวอักษร และสามารถล้ำไปด้านล่างได้ไม่เกิน 50 เปอร์เซ็นต์ของความสูงตัวอักษรดังรูปที่ 4.1



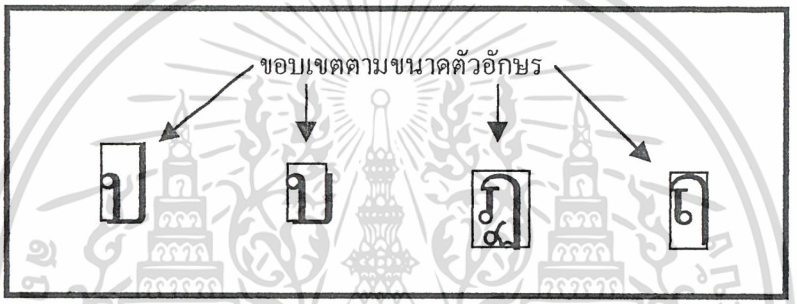
รูปที่ 4.1 แสดงรูปของตัวอักษร ฎ ป ฐ

4.1.2 กระบวนการเปรียบเทียบขั้นที่หนึ่ง

หลังจากพิจารณาตัวอักษรแล้วทำการกำหนดขอบเขตอีกครั้งตามขนาดของตัวอักษรจริงๆ ดังตัวอย่างในรูปที่ 4.3 เพราะอาจจะมีกรณีที่ตัวอักษรจะเยื้องไปทางด้านหน้า เยื้องไปทางด้านล่าง ล้ำขึ้นด้านบนและด้านล่างแล้วทำการ ตรวจสอบตัวอักษรที่อยู่กลางบรรทัดแบบที่ไม่มีส่วนล้ำเช่น ก,ข,ค,ม เป็นต้น และแบบพิเศษชนิดที่หนึ่งที่มีส่วนเยื้องขึ้นมาทางด้านหน้า ได้แก่ “โ”, “ไ”, “ใ” “เ” ซึ่งถ้าไม่พิจารณาขอบเขตของตัวอักษรที่เยื้องขึ้นมาจะทำให้ได้ขอบเขตดังในรูปที่ 4.2 ซึ่งเมื่อทำการเปรียบเทียบแล้วจะได้เป็น “เ” ทั้งหมด และแบบพิเศษชนิดที่สองที่มีส่วนขึ้นด้านบนหรือด้านล่าง ได้แก่ “พ”, “ฟ”, “ฝ”, “ฬ”, “บ”, “ป” ดังที่แสดงในรูปที่ 4.1 โดยทำการเปรียบเทียบตัวอักษรที่อยู่กลางบรรทัดกับตัวอักษรต้นแบบให้หมดซึ่งโดยทั่วไปแล้วจะสามารถเปรียบเทียบได้เกือบทั้งหมด แต่เนื่องจากตัวอักษรบางตัวมีลักษณะคล้ายกันมากทำให้ไม่สามารถแยกตัวอักษรเหล่านั้นได้ จึงจำเป็นต้องมีกระบวนการในการแยกความแตกต่างของตัวอักษรที่คล้ายกันออกมาดังตัวอย่างในการแยกความแตกต่างระหว่าง พ,ฟ,ป,บ ในรูปที่ 4.4 และรูปที่ 4.5 และจะกล่าวถึงรายละเอียดต่อไป



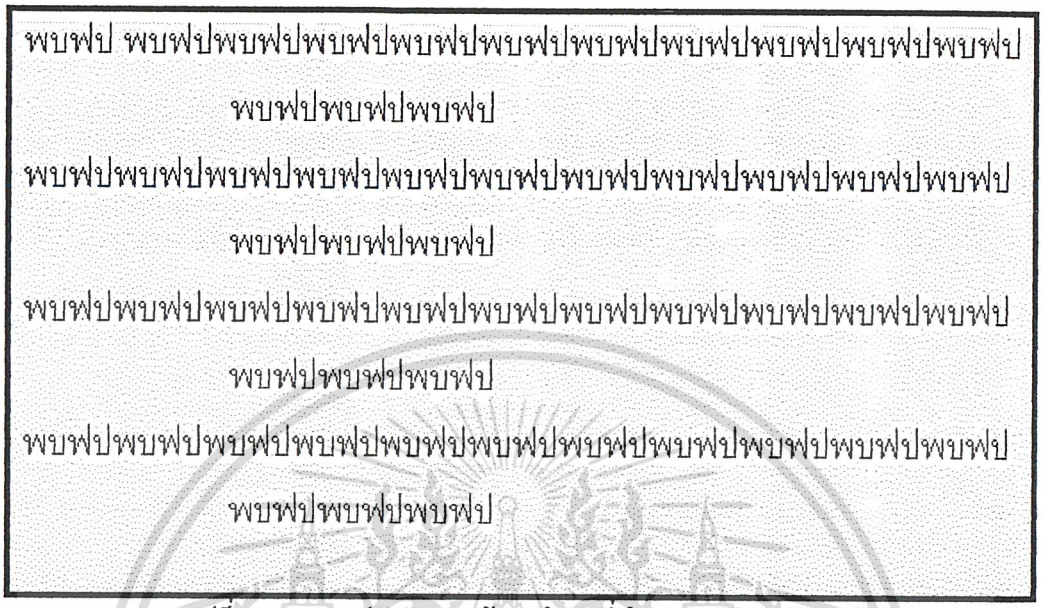
รูปที่ 4.2 แสดงขอบเขตของสระ “โ”



รูปที่ 4.3 แสดงรูปขอบเขตตามขนาดของตัวอักษร

พบฟป พบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟป
พบฟปพบฟปพบฟป
พบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟป
พบฟปพบฟปพบฟป
พบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟป
พบฟปพบฟปพบฟป
พบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟป
พบฟปพบฟปพบฟป

รูปที่ 4.4 แสดงรูปของตัวอักษรที่มีลักษณะคล้ายคลึงกัน



รูปที่ 4.5 แสดงรูปตัวอักษรคล้ายคลึงกันที่ผ่านการเปรียบเทียบแล้ว

4.1.3 การแยกความแตกต่างระหว่างอักษรที่คล้ายคลึงกัน

เนื่องจากมีตัวอักษรหลายตัวที่มีลักษณะคล้ายคลึง ถ้าใช้วิธีการทำการเปรียบเทียบขั้นที่หนึ่งเพียงอย่างเดียวจะไม่สามารถแยกตัวอักษรที่มีลักษณะคล้ายกันได้ ดังกรณีต่อไปนี้

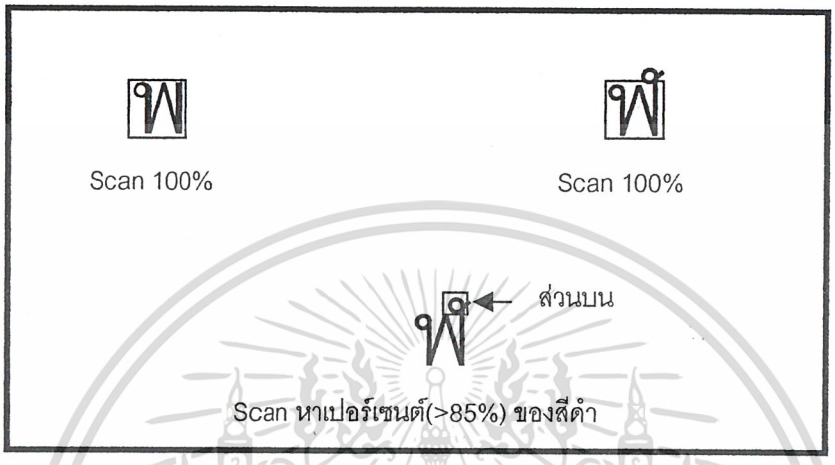
4.1.3.1 กรณี "โ", "ไ", "ใ" "เ"

ถ้าใช้กระบวนการขั้นที่หนึ่งอย่างเดียวจะไม่สามารถแยกความแตกต่างระหว่าง "เ" , "โ","ใ", "ไ" เนื่องจากโดยกระบวนการขั้นที่หนึ่งจะเปรียบเทียบออกมาเป็น "เ" ทั้งหมดดังที่กล่าวไว้ในหัวข้อที่ 4.1.1 เนื่องจากการสแกนหาคอลัมน์นั้นถ้าเป็น "โ", "ไ", "ใ" จะเจอเป็น "เ" ดังนั้นจึงต้องหาขอบด้านซ้ายของ "โ", "ไ", "ใ" เพื่อนำมาอ้างอิงเปรียบเทียบกับ "เ" ว่ามีระยะที่จะต้องเลื่อนไปทางด้านซ้ายเท่าไหร่แล้วเก็บค่าดังกล่าวไว้ เมื่อได้ระยะที่ถอยไปทางด้านซ้ายแล้วจะต้องการโหลดข้อมูลมาเก็บไว้ใหม่อีกครั้งเนื่องจากข้อมูลเก่าจะเปรียบเทียบได้เฉพาะ "เ" เท่านั้น แล้วเมื่อโหลดข้อมูลใหม่มาแล้วจะนำกลับไปเปรียบเทียบอีกครั้งหนึ่งซึ่งถ้าตัวอักษรนั้นคือ "เ" ก็จะได้ผลลัพธ์ออกมาเป็น "เ" เหมือนเดิม แต่ถ้าเป็นตัวอักษร "โ", "ไ", "ใ" ก็จะได้ผลลัพธ์ออกมาเป็น "โ", "ไ", "ใ"

4.1.3.2 กรณี "พ" กับ "ฬ"

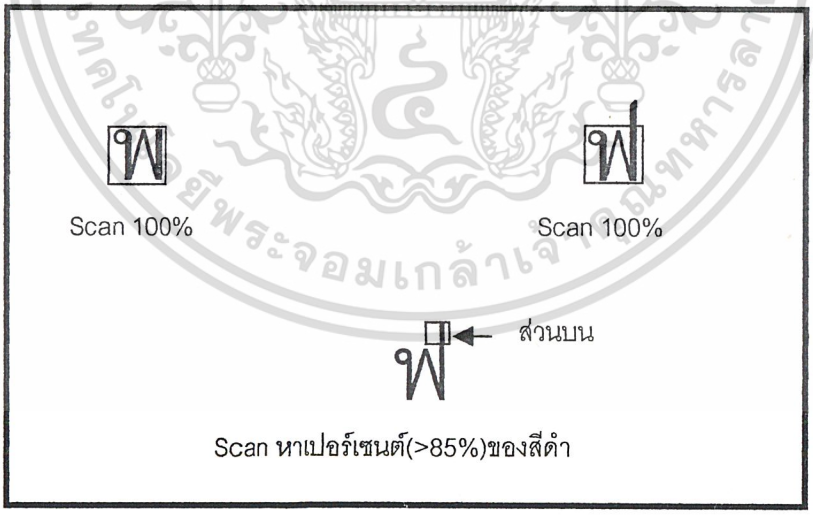
ถ้าใช้กระบวนการขั้นตอนที่หนึ่งอย่างเดียวจะไม่สามารถแยกความแตกต่างระหว่าง "พ" กับ "ฬ" ได้โดยกระบวนการขั้นที่หนึ่งจะเปรียบเทียบออกมาเป็น "พ" ทั้งหมด ดังนั้นจะต้องแบ่งการเปรียบเทียบออกเป็น 2 ส่วนคือส่วนที่หนึ่งเป็นตัวอักษรส่วนกลาง(Middle character) ซึ่งจะทำการสแกนทั้งส่วนที่เป็นเนื้อสีดำของตัวอักษรและส่วนที่เป็นพื้นที่ว่าง เช่นกรณีนี้ "ฬ" จะได้เป็น "พ" แล้วนำไปเปรียบเทียบเปอร์เซ็นต์ที่ตั้งเอาไว้ซึ่งในกรณีนี้จะได้ออกมาเป็น 100 เปอร์เซ็นต์ ส่วนที่สอง เป็นส่วนต่อขึ้นไปข้างบน(Upper character) ซึ่ง

ในส่วนนี้จะทำการสแกนเฉพาะส่วนที่เป็นเนื้อสีดำของตัวอักษรเท่านั้นซึ่งถ้าพบว่ามีส่วนที่เป็นเนื้อสีดำมากกว่าเปอร์เซ็นต์ที่กำหนดไว้ (ระบบกำหนดไว้ที่มากกว่า 85%ตามข้อมูลที่ได้รวบรวมมา) ได้ผลลัพธ์ออกมาเป็นตัว “ฬ”



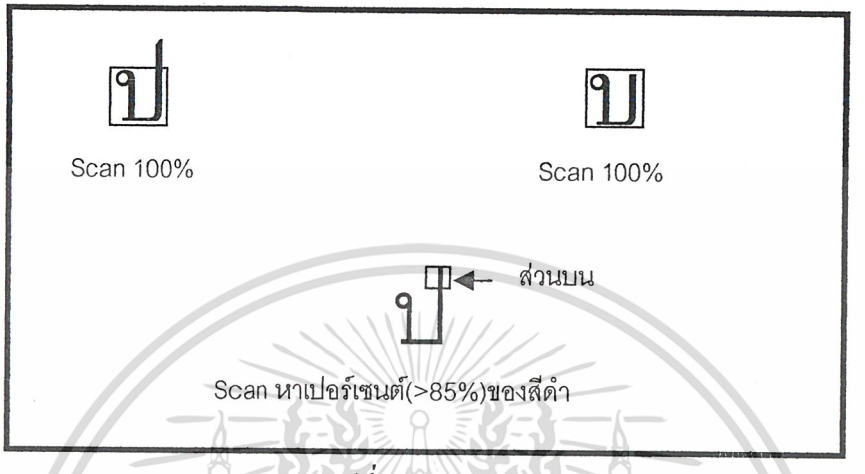
รูปที่ 4.6 แสดงรูป พ,ฬ

4.1.3.3 กรณี “พ” กับ “ฟ” จะมีหลักเกณฑ์การเปรียบเทียบคล้ายกับกรณี “พ” กับ “ฬ” เช่นกัน



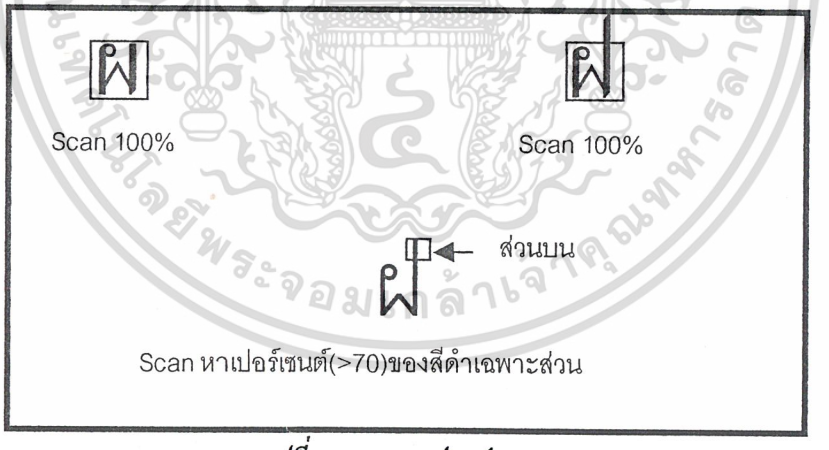
รูปที่ 4.7 แสดงรูป พ,ฟ

4.1.3.4 กรณี “บ” กับ “ป” จะมีหลักเกณฑ์การเปรียบเทียบคล้ายกับกรณี “พ” กับ “ฟ” เช่นกัน



รูปที่ 4.8 แสดงรูป บ,ป

4.1.3.5 กรณี “ผ” กับ “ฝ” จะมีหลักเกณฑ์การเปรียบเทียบคล้ายกับกรณี “พ” กับ “ฟ” เช่นกัน



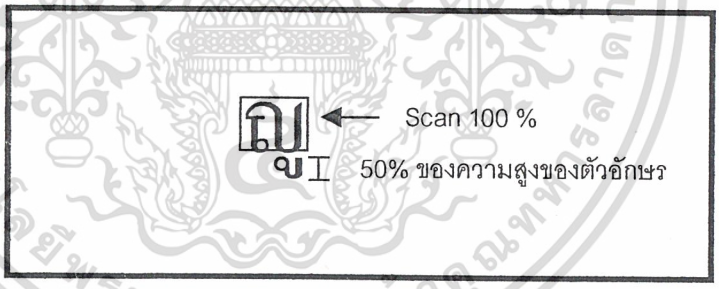
รูปที่ 4.9 แสดงรูป ผ,ฝ

4.1.4 การเปรียบเทียบตัวอักษรด้านล่างหรือสระล่าง

จากการสังเกตจะพบว่าตัวพยัญชนะในภาษาไทยเท่านั้นที่จะมีสระล่าง และตัวพยัญชนะที่ไม่มีสระล่างแน่นอน ได้แก่ “ฐ”, “ฎ”, “ฏ”, “ฤ”, “ฦ” จากนั้นจะทำการสแกนหาขอบบนของสระล่างโดยยึดเส้น Base line ที่หาได้ในตอนแรก โดยตั้งต้นที่ตำแหน่ง 1.8*ของความสูงของตัวอักษร ซึ่งพิจารณาว่าใน 1 คอลัมน์ประกอบด้วย ตัวอักษรระดับกลาง สระล่างและสระบน โดยที่สระบนจะมีความสูงประมาณ 80 เปอร์เซ็นต์ของความสูงของตัวอักษร และตัวอักษรระดับกลางบรรทัดมีความสูง

ประมาณ 100 เปอร์เซ็นต์ของความสูงของตัวอักษร ส่วนสระล่างจะกำหนดมีความสูงประมาณ 50 เปอร์เซ็นต์ของความสูงของตัวอักษรดังรูปที่4.10 ซึ่งค่าประมาณเหล่านี้ได้มาจากข้อมูลตัวอักษรที่รวบรวมไว้จากนั้นจะหาขอบบน และขอบซ้ายที่เริ่มต้นมีเนื้อสีดำของตัวอักษร จากนั้นจะทำการเปรียบเทียบว่าเป็นสระ “อุ” หรือ สระ “อู” และทำการตรวจสอบอีกครั้งว่าเป็นตัว “ญ” ที่ตามด้วยสระ “อุ” หรือไม่ ซึ่งคำดังกล่าวไม่มีในพจนานุกรม จะเปรียบเทียบออกมาเป็น “ญ” เท่านั้นแต่ในกรณีที่ เป็นตัว “ญ” ตามด้วยสระ “อู” เช่นคำว่า กตัญญู จะสามารถเปรียบเทียบได้ออกมาเป็น “ญ” และ สระ “อู” ตามลำดับดังรูปที่ 4.11

รูปที่ 4.10 แสดงส่วนสูงโดยประมาณของส่วนต่างๆของพยัญชนะ



รูปที่ 4.11 แสดงรูป ญ

4.1.5 การเปรียบเทียบตัวอักษรด้านบนหรือสระบน

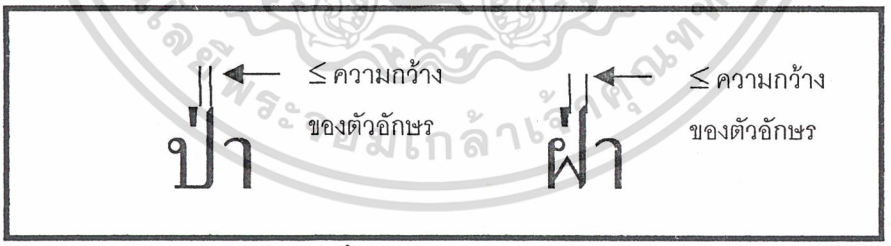
การเปรียบเทียบตัวอักษรด้านบนหรือสระด้านบนจะต้องทำการ โหลดข้อมูลเข้ามาใหม่ เนื่องจากสระบนนั้นโดยส่วนมากจะเหลื่อมไปทางด้านหน้าเล็กน้อย เช่น สระ “อิ” สระ “อี” เป็นต้น จะเหลื่อมตัวพยัญชนะไปเล็กน้อย ประมาณไม่เกิน 10 เปอร์เซ็นของความสูงของตัวอักษร ซึ่งค่าดังกล่าวได้มาจากข้อมูลตัวอักษรที่รวบรวมไว้

รูปที่ 4.12 แสดงสระ อิ,อี ให้เห็นชัดเจน

และมีสระบนบางตัวที่จะเหลื่อมไปทางด้านหลังเช่น ไม้หันอากาศ ไม้โท ซึ่งจะเหลื่อมไปทางด้านหลังเล็กน้อย ประมาณไม่เกิน 20 เปอร์เซ็นต์ของความสูงของตัวอักษรดังแสดงไว้ในรูปที่ 4.10 ซึ่งค่าดังกล่าวได้มาจากข้อมูลตัวอักษรที่รวบรวมไว้

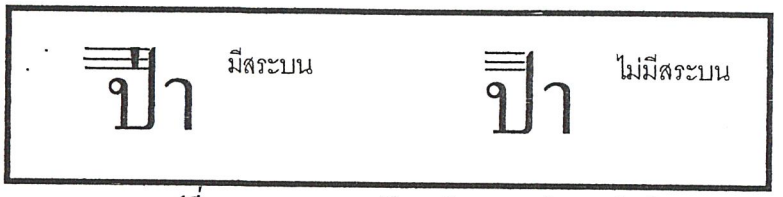
รูปที่ 4.13 แสดงสระ ไม้หันอากาศ ไม้โท อย่างชัดเจน

จากนั้นจะต้องทำการตรวจสอบก่อนว่าตัวอักษร “ป”, “ฝ”, “ฟ”, “พ” มีสระบนหรือไม่ โดยทำการสแกนจากด้านขวาไปซ้ายจนสุดความกว้างของตัวอักษร เพราะถ้าตัวอักษรเหล่านี้มีสระบน สระบนของตัวพยัญชนะเหล่านี้จะมีลักษณะดังรูปที่ 4.14 คือจะอยู่ระหว่างตัวอักษร แต่ถ้าสแกนแล้วไม่พบเนื้อสีดำแสดงว่าตัวอักษรดังกล่าวไม่มีสระบน



รูปที่ 4.14 แสดง คำว่า ป่า ฝ่า

จากรูปที่ 4.14 จะเห็นว่าคำว่า “ฝ่า” หรือ “ป่า” ตำแหน่งของไม้เอกจะไม่เหมือนกับคำว่า “ด่า” หรือ “ว่า” คำว่า “ฝ่า” ไม้เอกจะอยู่ระหว่างตัว “ฝ” ไม่ได้อยู่กึ่งกลางระหว่าง “ฝ” กับสระอา ดังนั้นจะทำให้เสมือนไม่เห็นไม้เอกดังกล่าวในเวลาที่เปรียบเทียบกับตัวอักษรต้นแบบ ดังนั้นถ้าเป็นอักษร “ป”, “ฝ”, “ฟ”, “พ” สแกนจากด้านขวาไปซ้ายจนสิ้นสุดความกว้างของตัวอักษร แล้วไม่พบเนื้อสีดำเลยแสดงว่าไม่มีสระนั้นอยู่จริง หลังจากทำการตรวจสอบแล้ว จะทำการเปรียบเทียบกับตัวอักษรต้นแบบต่อไป

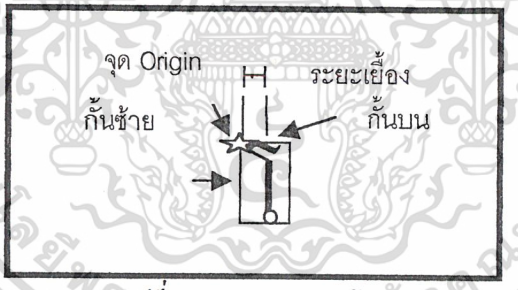


รูปที่ 4.15 แสดงการเปรียบเทียบระหว่าง คำว่า ป้า กับ ปา

ในขั้นตอนการเปรียบเทียบของไม้หันอากาศ ซึ่งจะใช้หลักการเดียวกับการเปรียบเทียบของไม้เอก เนื่องจากไม้หันอากาศจะเหลื่อมตัวพยัญชนะไปทางด้านหลังประมาณ 20 เปอร์เซ็นต์ของความสูงของตัวอักษร ดังที่แสดงไว้ในรูปที่ 4.13 และนำมาหาขอบบนและขอบด้านซ้าย เพื่อใช้เปรียบเทียบกับตัวอักษรต้นแบบต่อไปหลังจากพิจารณาตัวไม้หันอากาศแล้ว จะพิจารณาสระ “อิ”, “อี”, “อึ”, “อื” ตามลำดับ โดยวิธีการเดียวกันกับวิธีข้างต้น เพราะสระดังกล่าวจะมีส่วนเหลื่อมตัวอักษรไปทางด้านหลังได้เช่นเดียวกับกรณีของตัวไม้หันอากาศ

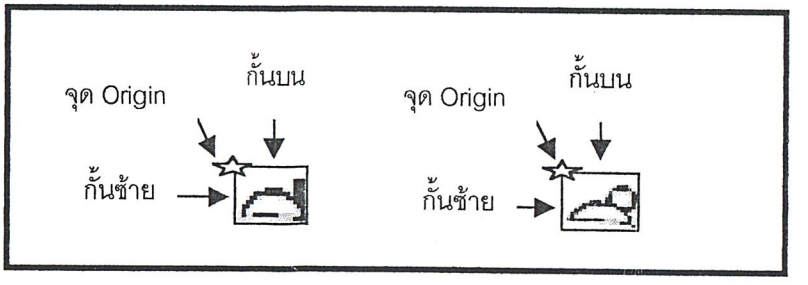
4.1.6 ขั้นตอนการหาจุดเริ่มต้น (Origin) ของตัวอักษร

การหาจุดเริ่มต้น (Origin) ของตัวอักษร เป็นการหาจุดอ้างอิงเพื่อทำให้เราสามารถทำการเปรียบเทียบกับตัวอักษรที่สแกนเข้ามากับตัวอักษรต้นแบบได้ ในกรณีปกติจุดเริ่มต้นขอบตัวอักษร(Origin) ทั่วๆไป เช่น “ก”, “อ” จะอยู่ที่ตำแหน่งที่คอลัมน์ตัดกับเส้น Baseline-FontHigh (ความสูงของตัวอักษร) ซึ่งจะอยู่ตรงมุมบนด้านซ้ายของตัวอักษร ยกเว้นสระ “โ”, “ไ”, “ใ” ที่จะมีส่วนที่เยื้องไปทางด้านหน้า ซึ่งเราต้องพิจารณาหาระยะที่เยื้องอีกครั้ง จึงจะได้จุดเริ่มต้น(Origin) จริงๆดังแสดงไว้ในรูปที่ 4.16



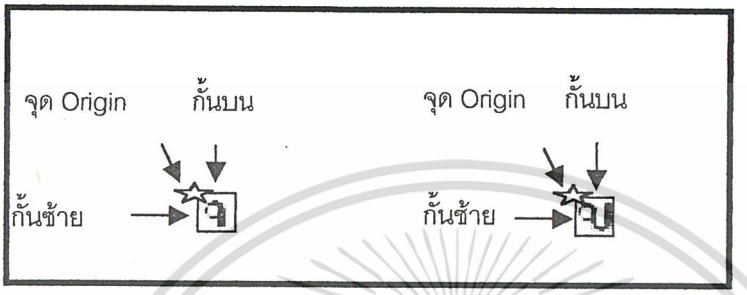
รูปที่ 4.16 แสดง สระ โอ

การหาจุดเริ่มต้น(Origin) ของสระบน จะต้องทำต้องทำการสแกนเพื่อหาขอบซ้ายและขอบบนก่อน เพื่อใช้กำหนดเป็นจุดเริ่มต้น(Origin) ดังนั้นสระบนจะต้องทำการหาจุดเริ่มต้นเป็นตัวๆไปดังรูปที่ 4.17 ซึ่งจะแตกต่างกับการหาจุดเริ่มต้น(Origin)ของตัวอักษร เมื่อเราได้ค่าคอลัมน์มาแล้วจะได้ค่าขอบซ้ายเลย ยกเว้นในกรณีสระ “โ”, “ไ”, “ใ” ที่กล่าวไว้ข้างต้น



รูปที่ 4.17 แสดง สระบน

การหาจุดเริ่มต้น(Origin)ของสระล่างจะมีขั้นตอนเหมือนกับการหาจุดเริ่มต้น(Origin)ของสระบน โดยจะทำสแกนเข้ามาใหม่เพื่อทำการหาขอบซ้ายและขอบบน เพื่อกำหนดเป็นจุดเริ่มต้น(Origin) ต่อไปดังรูปที่ 4.18



รูปที่ 4.18 แสดง สระล่าง

4.1.7 การส่งข้อมูลเพื่อไปทำการแปล

จะทำการส่งข้อมูลไปแปลโดยจะหาขอบเขตของบล็อคข้อมูลแล้วส่งไปแปล โดยจะโหลดข้อมูลเข้ามาทีละบล็อค โดยขนาดของบล็อค ด้านกว้างจะมีความยาวไม่เกิน 1.8 เท่าของความสูงของตัวอักษรและด้านยาวจะมีความยาวไม่เกิน 0.5 เท่าของความสูงของตัวอักษร ถ้าความกว้างของคอลัมน์ ณ ตำแหน่งปัจจุบันเล็กกว่าความกว้างของคอลัมน์ที่ผ่านมา ให้ลดขนาดของบล็อคลง แต่ถ้า ณ ตำแหน่งของคอลัมน์ดังกล่าวไม่มีตัวอักษรอยู่เลยแสดงว่าเป็นตำแหน่งของเว้นวรรคข้อความหรือย่อหน้า จะต้องทำการคำนวณหาระยะโดยเทียบมาจากค่าประมาณของสเปสบาร์รวมกับช่องว่างระหว่างตัวอักษรดังกล่าว ซึ่งใช้สมการดังต่อไปนี้ในการคำนวณ

```
Onespace = FontHigh*0.67
```

เพื่อทำการกระโดดข้ามไปเปรียบเทียบยังส่วนที่มีตัวอักษรต่อไป หลังจากนั้นจะทำการเปรียบเทียบตัวอักษรที่โหลดไว้ใน บล็อคกับตัวอักษรต้นแบบที่โปรแกรมได้คำนวณหาไว้แล้ว

กรณีที่เป็นสระ “แอ” จะต้องทำการตรวจสอบตัวถัดไปอีก 1 ตัวเนื่องจากจะทำการเปรียบเทียบได้เป็นสระ “เอ” สองตัวติดกัน ถ้าดูโดยผิวเผินดูเหมือนว่าโปรแกรมไม่ได้ผิดพลาด แต่ถือว่าผิดพลาดทางอัลกอลิทึม ส่วนในกรณีที่เป็นสระ “อำ” กับสระ “อา” ในตอนแรกจะเปรียบเทียบออกมาเป็นสระ “อา” ทุกครั้งก่อน หลังจากนั้นจะทำการตรวจสอบหานคลิขิต ถ้ามีนคลิขิตจะเปลี่ยนสระ “อา” ดังกล่าว ให้มีค่าเป็น สระ “อำ”

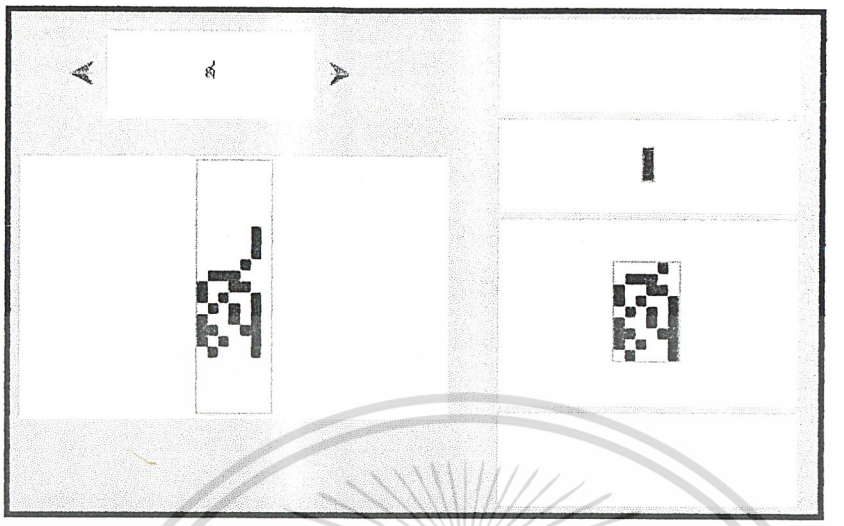
4.1.8 การเปรียบเทียบตัวอักษรที่ติดกัน

ในกรณีมีตัวอักษรที่ตัวติดกันดังเช่นตัวอย่างรูปที่ 4.19 จะไม่สามารถแยกออกจากกันด้วยการสแกนคอลัมน์ได้ดังแสดงไว้ในรูปที่ 4.21 ดังนั้นจะต้องทำการเปรียบเทียบตัวอักษรตัวแรกกับตัวอักษรต้นแบบเสียก่อน เมื่อเปรียบเทียบเสร็จแล้วจึงหาความกว้างของตัวอักษรตัวแรกที่ทำการเปรียบเทียบได้แล้ว จากนั้นจะตัด

ส่วนนี้ออกไปแล้วทำการเปรียบเทียบตัวอักษรที่อยู่ถัดไป ทำวิธีการนี้ไปเรื่อยๆจนกระทั่งไม่พบตัวอักษรที่ติดกันอีก

รูปที่ 4.19 แสดงตัวอักษรที่ติดกัน

รูปที่ 4.20 แสดงตัวอักษรที่ติดกันที่ผ่านการเปรียบเทียบแล้ว



รูปที่ 4.21 แสดงรูปของตัวอักษรติดกันจริง

หลังจากนั้นจะทำการเปรียบเทียบกับตัวอักษรต้นแบบอีกครั้ง จากนั้นจะทำการคำนวณหาขอบซ้ายของตัวอักษรตัวถัดไป อาศัยหลักการที่ว่าขนาดของตัวอักษรระดับกลางที่อยู่ถัดไปนั้นจะต้องมีขนาดมากกว่าหรือเท่ากับ 25 เปอร์เซ็นต์ของความสูงของตัวอักษรซึ่งค่านี้ได้จากข้อมูลของตัวอักษรที่รวบรวมไว้ เมื่อได้ขอบซ้ายของตัวอักษรทุกๆตัวแล้วจะสามารถนำไปใช้ในการเปรียบเทียบกับตัวอักษรต้นแบบได้หมดหน้าเอกสารจากนั้นจะได้ค่าอินพุทออกมาว่ารูปตัวอักษรดังกล่าวเป็นตัวอักษรใดดังรูปที่ 4. 20

บทที่ 5

การทำงานและทดลอง

5.1 การทำงานของโปรแกรมรู้จำตัวอักษรภาษาไทย

การทำงานในขั้นตอนแรกของโปรแกรม เริ่มจากการรับข้อมูลในหน้าเอกสาร หาขนาดของตัวอักษรในหน้าเอกสาร ความสูงของตัวอักษร ความสูงของบรรทัดโดยอัตโนมัติ แล้วจะโหลดค่าชนิดของตัวอักษรต้นแบบมาเก็บไว้ในหน่วยความจำ ทำการสแกนหา Base line จะได้ว่ามีเส้นบรรทัดอยู่ที่ไหนบ้าง ทำการสแกนหาคอลัมน์ทีละบรรทัด ก็จะได้ขอบเขตของตัวอักษรซึ่งอาจจะมีตัวเดียวหรือหลายตัวปนกันอยู่ จากนั้นจะทำการเปรียบระหว่างขอบเขตของตัวอักษรที่ได้กับตัวอักษรต้นแบบที่เก็บไว้ในหน่วยความจำ ดังนั้นผลลัพธ์ที่ได้จะเป็นแฟ้มข้อมูล txt file

5.1.1 อุปกรณ์ที่ต้องนำมาใช้ในการทำงานของโปรแกรม

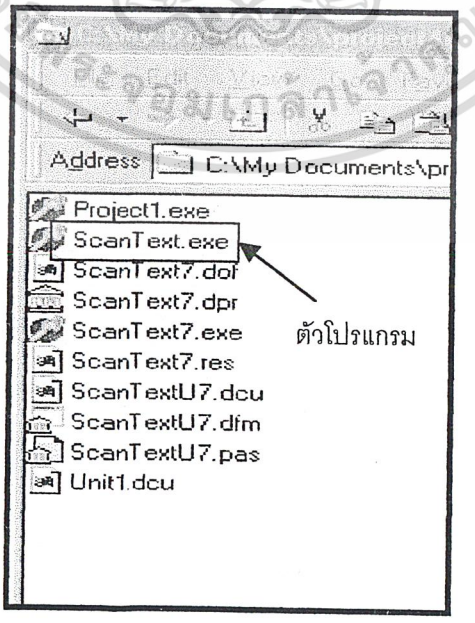
1. เครื่องไมโครคอมพิวเตอร์ CPU Pentium ความเร็ว 200 MHz หน่วยความจำ 32 MB
2. จอภาพแสดงผล
3. ฮาร์ดดิส 1 ตัว มีเนื้อที่ว่าง อย่างน้อย 2 Mbytes ขึ้นไป
4. เครื่องตรวจกวาดภาพ (Scanner) ยี่ห้อ Umax รุ่น PowerLook 3000 optical resolution 3048X3048 dpi

5.2 ขั้นตอนการใช้งานของโปรแกรม

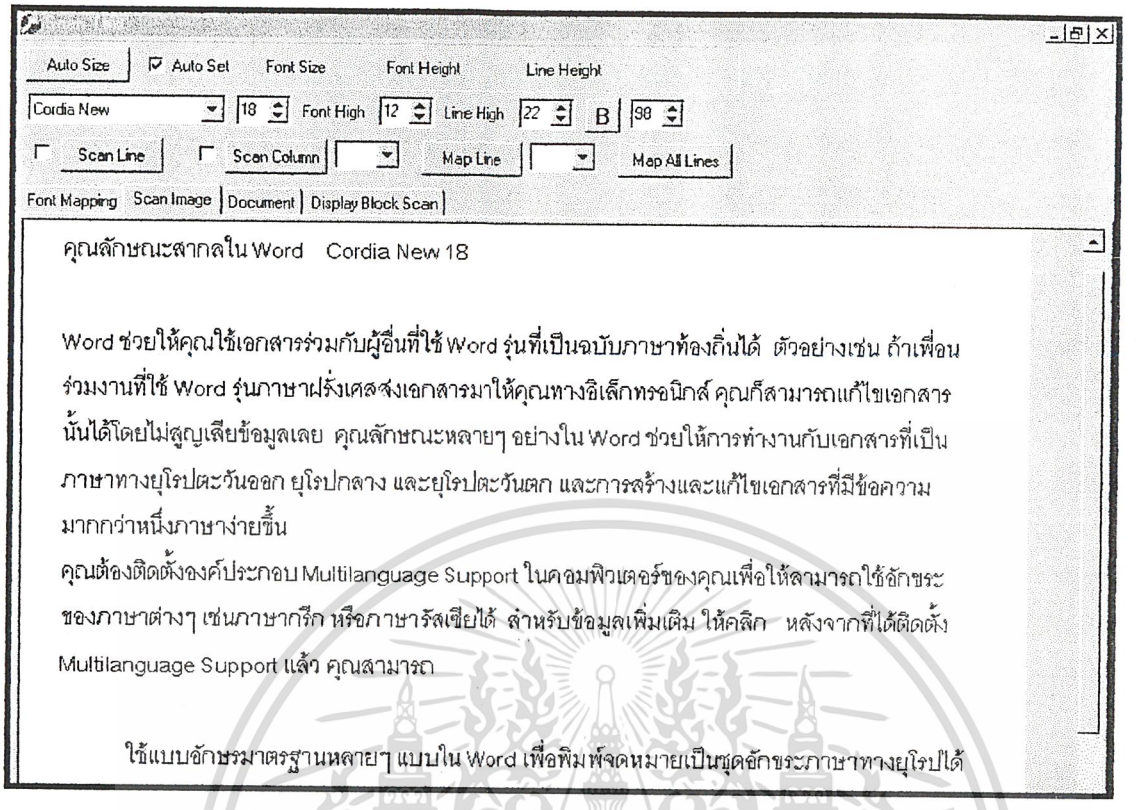
การทำงานของโปรแกรมจะเป็นการทำงานผ่านปุ่มการทำงานต่างๆ โดยจะมีขั้นตอนดังต่อไปนี้

ขั้นที่ 1 การเรียกใช้การทำงานของโปรแกรม

คลิ๊กที่ตัวโปรแกรมดังรูปที่ 5.1 แล้วโปรแกรมจะเริ่มทำงาน โดยมีปุ่มการทำงานต่างๆ เป็นตัวควบคุมการทำงาน



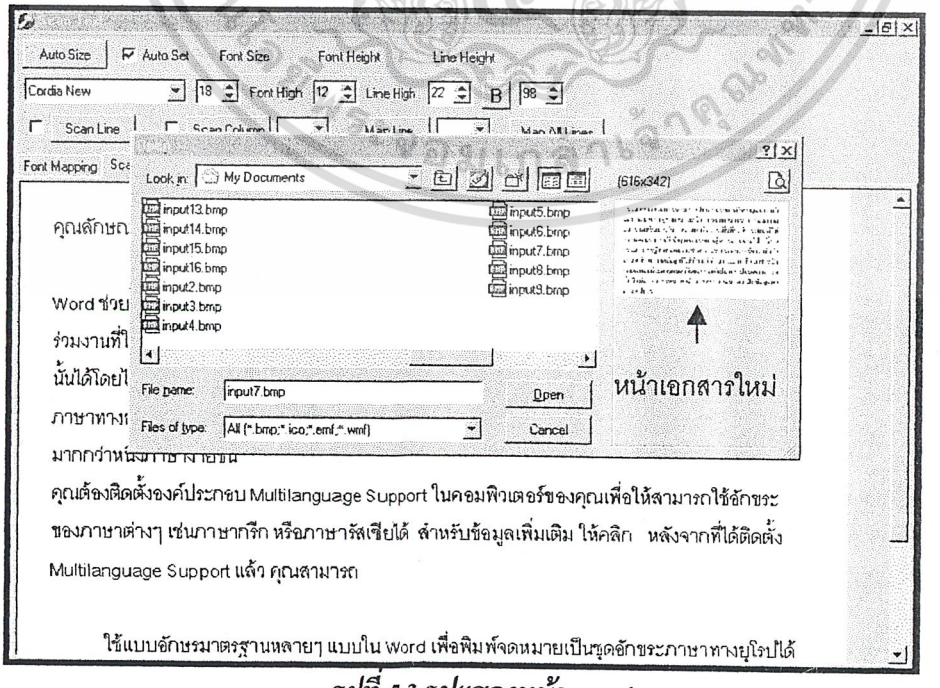
รูปที่ 5.1 แสดงไอคอนของ program



รูปที่ 5.2 แสดงการทำงานของโปรแกรมเมื่อเริ่มต้นทำงาน

ขั้นที่ 2 การเปิดภาพหน้าเอกสาร

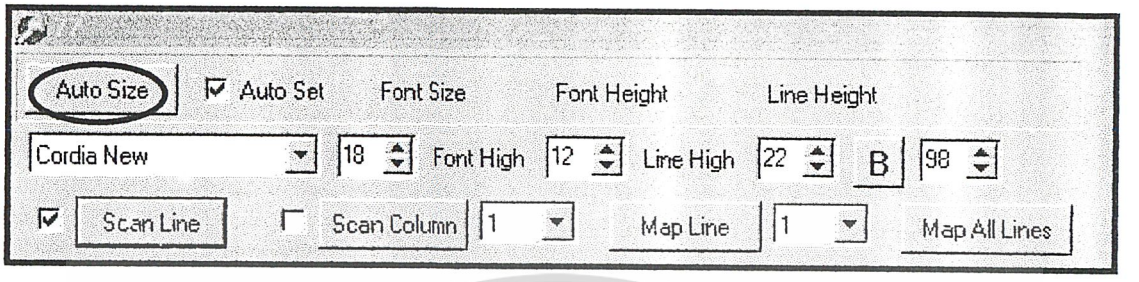
โดยการดับเบิ้ลคลิ๊กที่รูปภาพของหน้าเอกสารในส่วนของหน้า Scan Image เพื่อทำการเปิดรูปภาพหน้าเอกสารใหม่ได้ โดยรูปของหน้าเอกสารจะเป็น Bmp file โหมดการทำงานเป็น RGB



รูปที่ 5.3 รูปแสดงหน้าscan image

ขั้นที่ 3 การกำหนดขนาดและความสูงของตัวอักษรโดยอัตโนมัติ

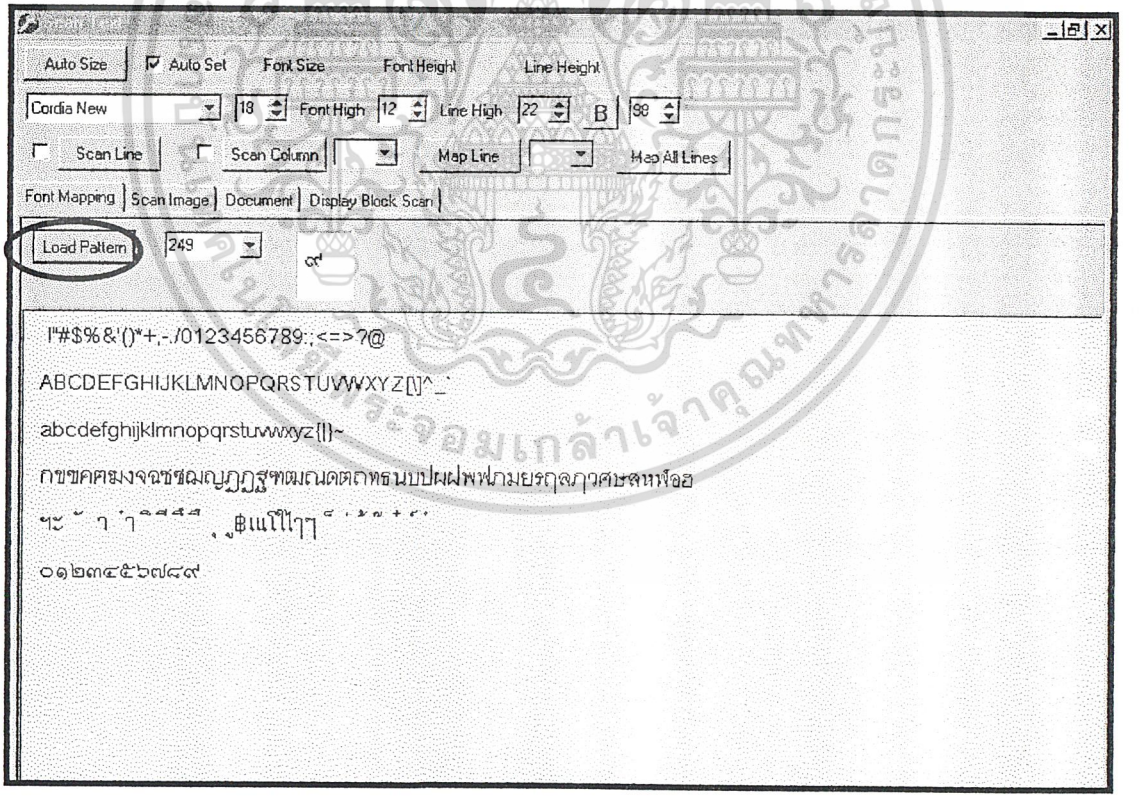
ในขั้นตอนนี้ จะทำงานได้โดยการกดปุ่ม Auto Size โปรแกรมจะทำการหาขนาดและความสูงของตัวอักษรในรูปภาพของหน้าเอกสาร โดยอัตโนมัติ เพื่อใช้ในการ โหลดตัวอักษรต้นแบบ



รูปที่ 5.4 รูปแสดงส่วน Auto size ของโปรแกรม

ขั้นที่ 4 การโหลดตัวอักษรต้นแบบมาไว้ในหน่วยความจำ

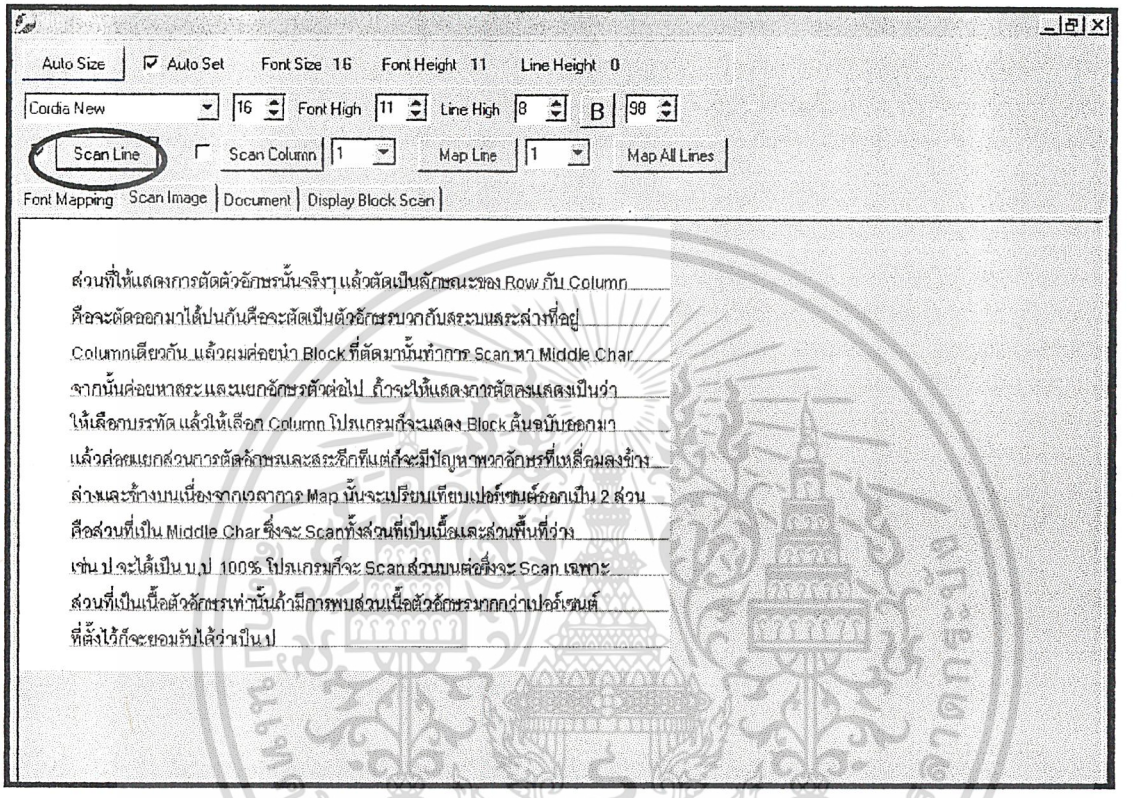
จะทำการเลือกชนิดของตัวอักษรและขนาดของตัวอักษร แล้วโหลดตัวอักษรต้นแบบมาไว้ในหน่วยความจำดังรูปที่5.5 จะทำงานได้โดยการกดปุ่ม Load Pattern ในหน้า Font Mapping



รูปที่ 5.5 แสดงการโหลดตัวอักษรต้นแบบในหน้า font Mapping

ขั้นที่ 5 การหา Base line

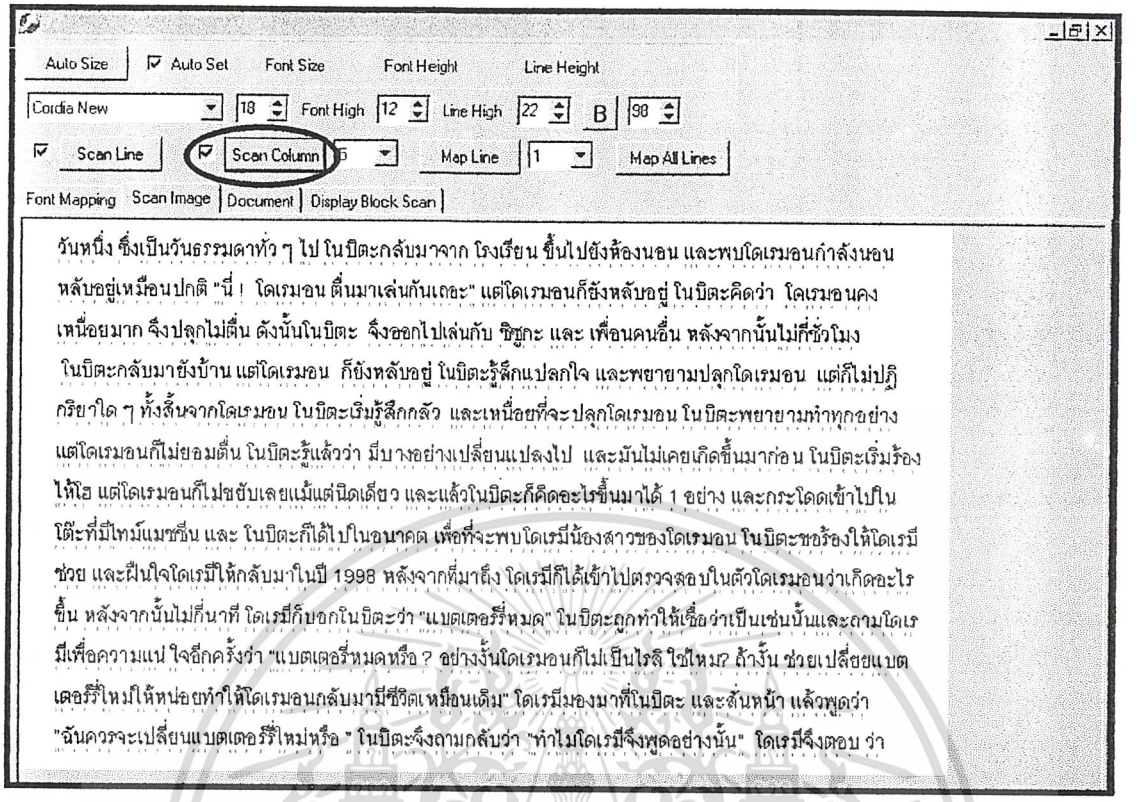
ในขั้นตอนที่ 5 นี้จะทำงานได้จะต้องผ่านขั้นตอนการทำงานการเปิดภาพหน้าเอกสารในขั้นตอนที่ 2 มาก่อน โดยโปรแกรมจะทำการวิเคราะห์หา Base line ของตัวอักษรในหน้าเอกสาร ทำได้โดยเลือกกดปุ่มการทำงาน Scan line ถ้าต้องการแสดงเส้นของ Base line ในภาพหน้าเอกสาร สามารถ check block ด้านหน้าของปุ่ม Scan lineดังแสดงในรูปที่5.6



รูปที่ 5.6 แสดงหน้า scanline ในหน้า scan image

ขั้นตอนที่ 6 การหาคอลัมน์ (Column)

ในขั้นตอนที่ 6 นี้จะทำงานได้จะต้องผ่านขั้นตอนการทำงานการเปิดภาพหน้าเอกสารในขั้นตอนที่ 2 มาก่อนเช่นกัน โดยโปรแกรมจะทำการวิเคราะห์เพื่อหาคอลัมน์ (column) ของตัวอักษรในหน้าเอกสาร ทำได้โดยเลือกกดปุ่มการทำงาน Scan column ถ้าต้องการให้แสดงพิกัดของคอลัมน์ (column) ในหน้าเอกสาร สามารถ check box ด้านหน้าของปุ่ม Scan column ดังแสดงในรูปที่ 5.7



รูปที่ 5.7 แสดงหน้า Scan column

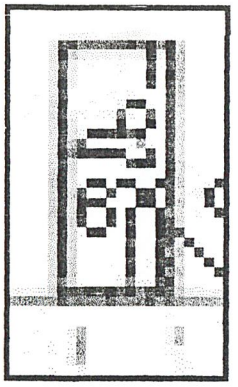
ขั้นตอนที่ 7 การเปรียบเทียบตัวอักษรในภาพหน้าเอกสารกับตัวอักษรต้นแบบ

ในขั้นตอนที่ 7 จะเป็นการทำการเปรียบเทียบตัวอักษรในภาพหน้าเอกสารกับตัวอักษรต้นแบบ (Pattern) โดยสามารถเลือกทำการเปรียบเทียบทีละบรรทัดโดย check box หน้าปุ่ม Map Line และเลือกบรรทัดที่ต้องการ หรือกดปุ่ม Map All Lines เพื่อทำการเปรียบเทียบทั้งหมดของหน้าเอกสารได้

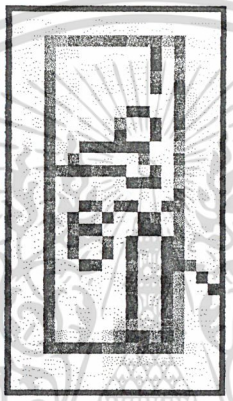
กระบวนการต่างๆที่อยู่ในขั้นตอนการเปรียบเทียบตัวอักษรในภาพหน้าเอกสารกับตัวอักษรต้นแบบที่เก็บไว้ในหน่วยความจำ

- เปรียบเทียบตัวอักษรระดับกลาง
- เปรียบเทียบตัวอักษรระดับล่าง ได้แก่ สระล่าง ทั้งหมด
- เปรียบเทียบตัวอักษรระดับบนชั้นที่ 2 และเปรียบเทียบตัวอักษรระดับบนชั้นที่ 1 ดังรูปที่ 5.10
- ทำการตัดขอบเขตตัวอักษรที่ผ่านการเปรียบเทียบแล้วออกไป
- พิจารณาคอลัมน์ที่อยู่ถัดไปจนหมดบรรทัดนั้น
- พิจารณาบรรทัดถัดไปจนกระทั่งหมดหน้าเอกสาร

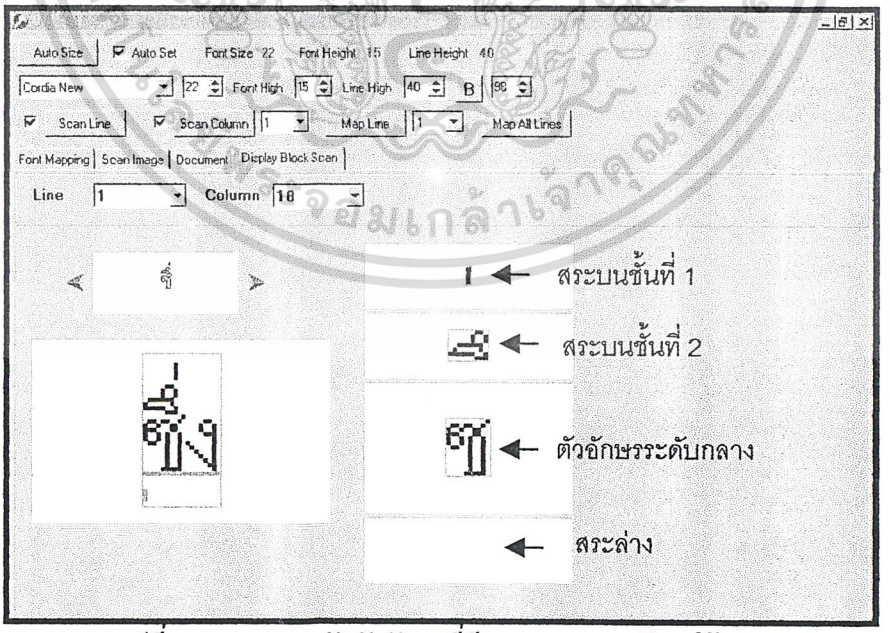
ในรูปที่ 5.8แสดงหน้าเอกสารที่มีสระสองชั้นอยู่ และรูปที่ 5.9 แสดงสระสองชั้นที่ผ่านการเปรียบเทียบแล้ว



รูปที่ 5.8 แสดงส่วนของรูปภาพตัวอักษรที่มีสระบนสองระดับ



รูปที่ 5.9 แสดงผลลัพธ์ของรูปภาพที่มีสระบนสองระดับ

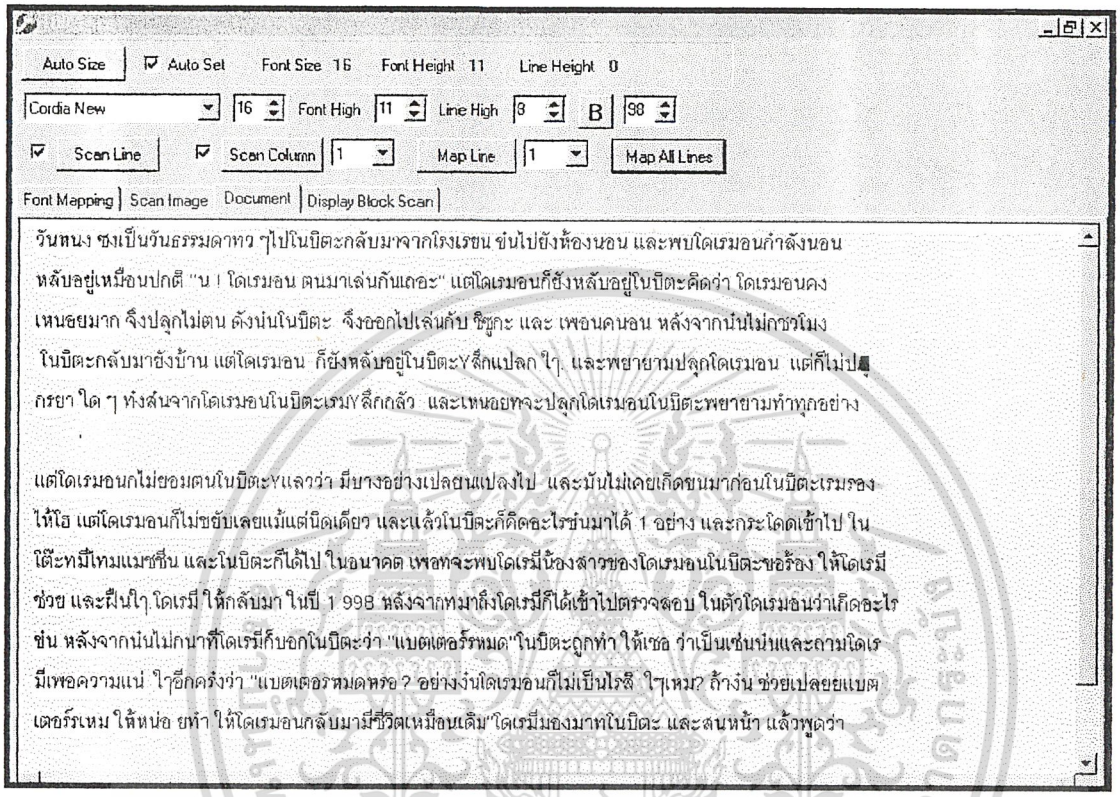


รูปที่ 5.10 แสดงการตัดตัวอักษรที่มีสระบนสองระดับอยู่ด้วย

ขั้นตอนที่ 8 การแสดงผล

การแสดงผลจะมีให้เลือกในหน้าต่าง ดังต่อไปนี้

1. ภายในหน้าต่างของ document จะแสดงภาพตัวอักษรในหน้าเอกสารที่ผ่านการเปรียบเทียบแล้ว และแสดงผลออกมาเป็นแฟ้มข้อมูลประเภท txt file ดังรูปที่ 5.11



รูปที่ 5.11 แสดงผลลัพธ์หน้า document

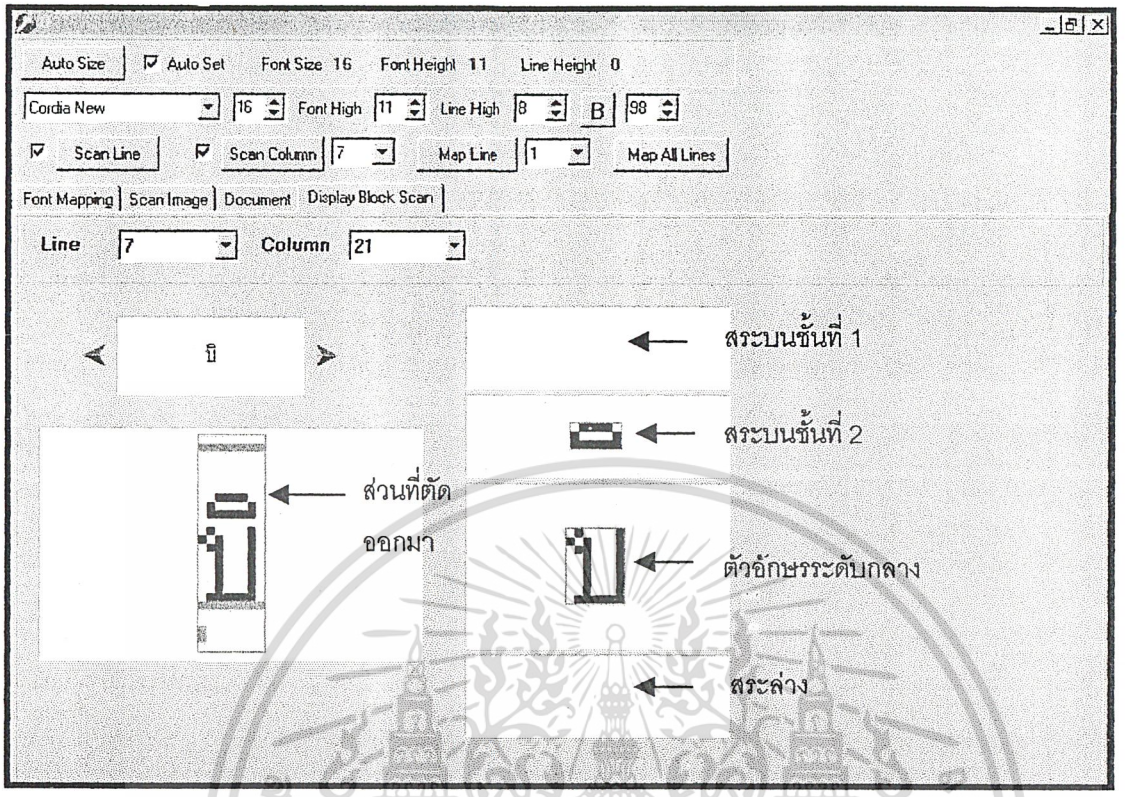
2. หน้าต่าง Display block Scan แสดงผลออกมาเป็นตัวอักษรที่ตัดออกมาแล้ว จะแสดงผลทีละตัว ในแนวของคอลัมน์นั้นเอง ดังนั้นในบางกรณีจะมีตัวอักษรที่ติดกันด้วย แบ่งออกเป็นกรณีดังได้ต่อไปนี้

รูปที่ 5.12 แสดงตัวอักษรที่ไม่มีส่วนติดกันและมีสระบนเพียงชั้นเดียวและแยกออกมาเป็นตัวๆได้

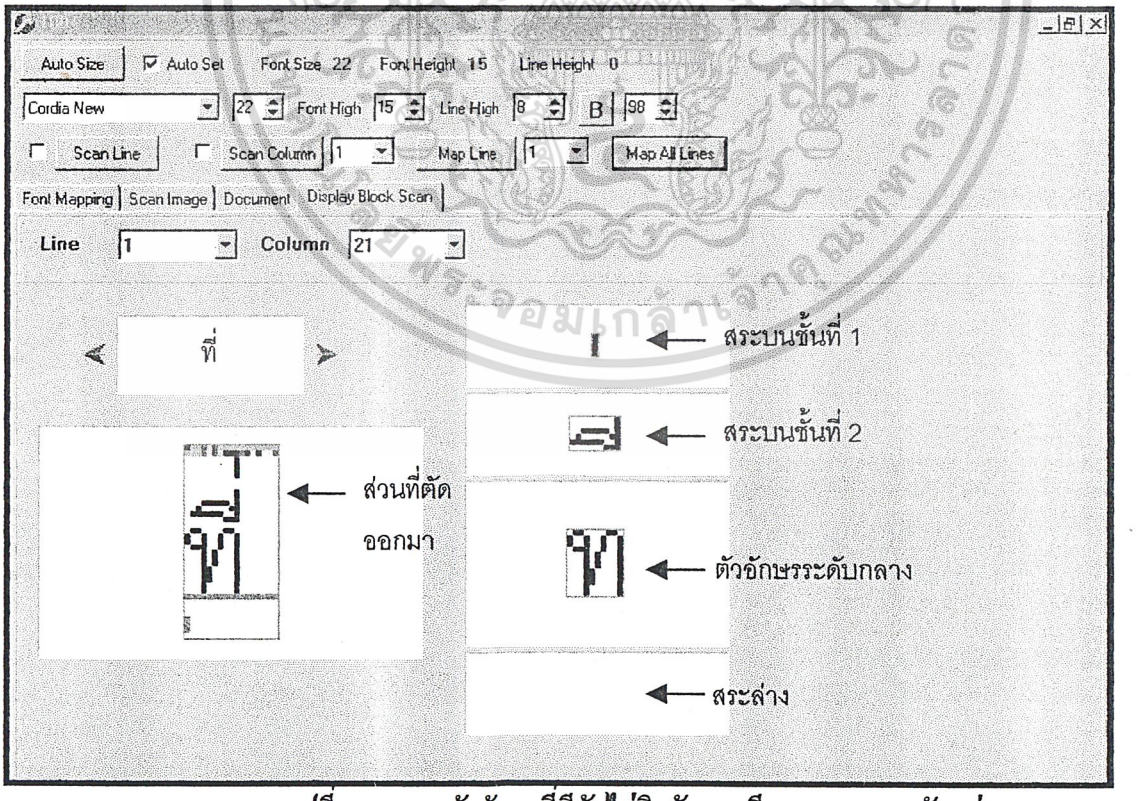
รูปที่ 5.13 แสดงตัวอักษรที่ไม่มีส่วนติดกันและมีสระบนสองชั้นและแยกออกมาเป็นตัวๆได้

รูปที่ 5.14 แสดงตัวอักษรที่มีส่วนติดกันและมีสระบนเพียงชั้นเดียวและสามารถแยกออกมาเป็นตัวๆได้

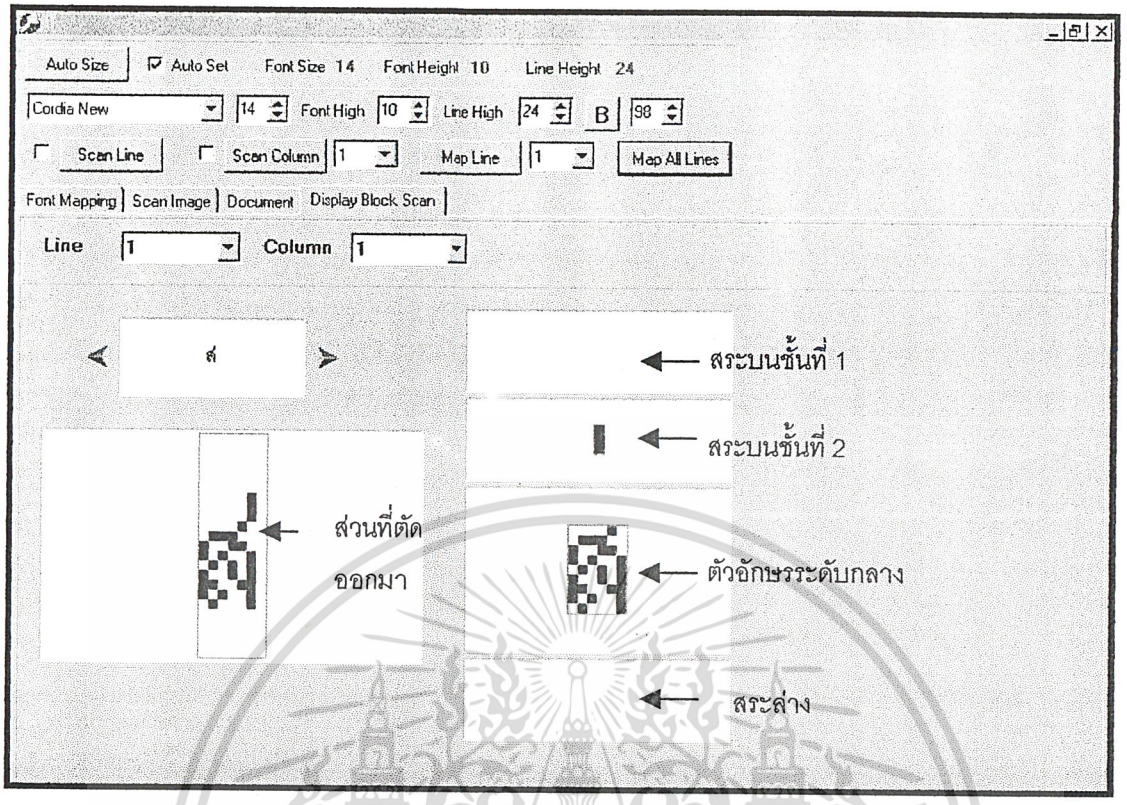
รูปที่ 5.15 แสดงตัวอักษรที่มีตัวอักษรระดับกลางติดกัน ในกรณีนี้จะไม่สามารถแยกออกมาเป็นตัวๆได้



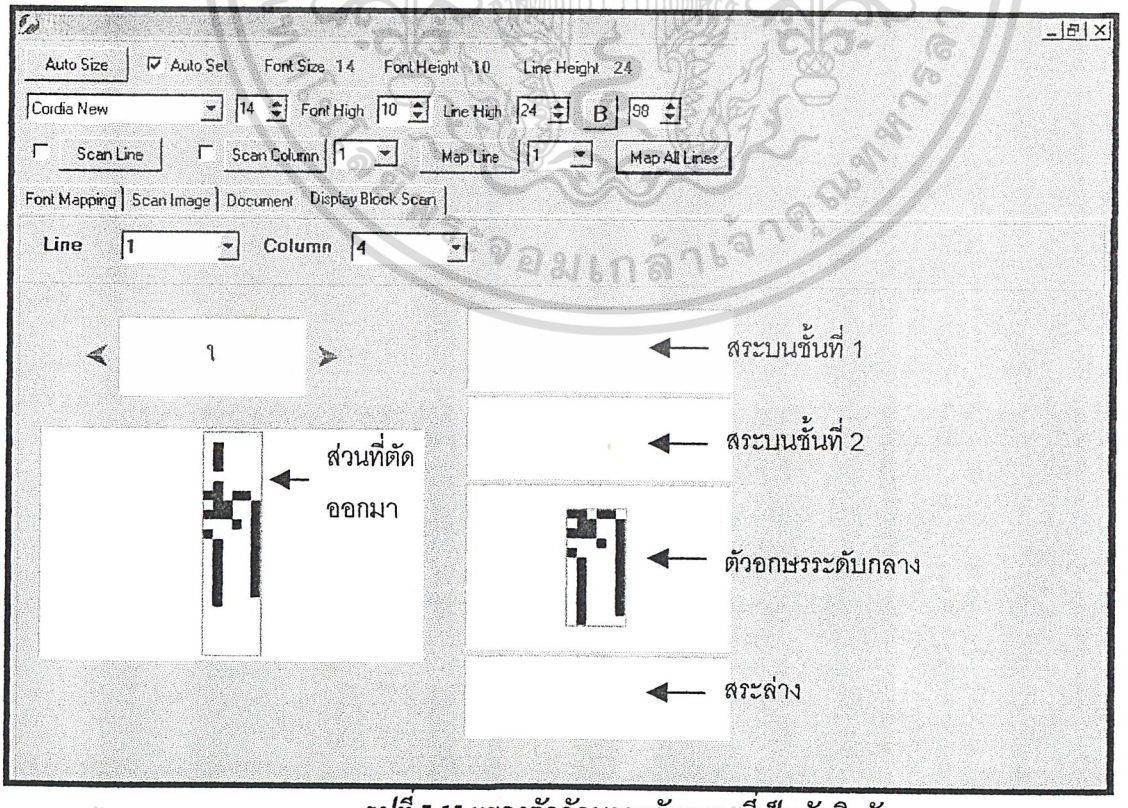
รูปที่ 5.12 แสดงตัวอักษรที่ไม่ได้ติดกันและมีสระบน 1 ชั้น



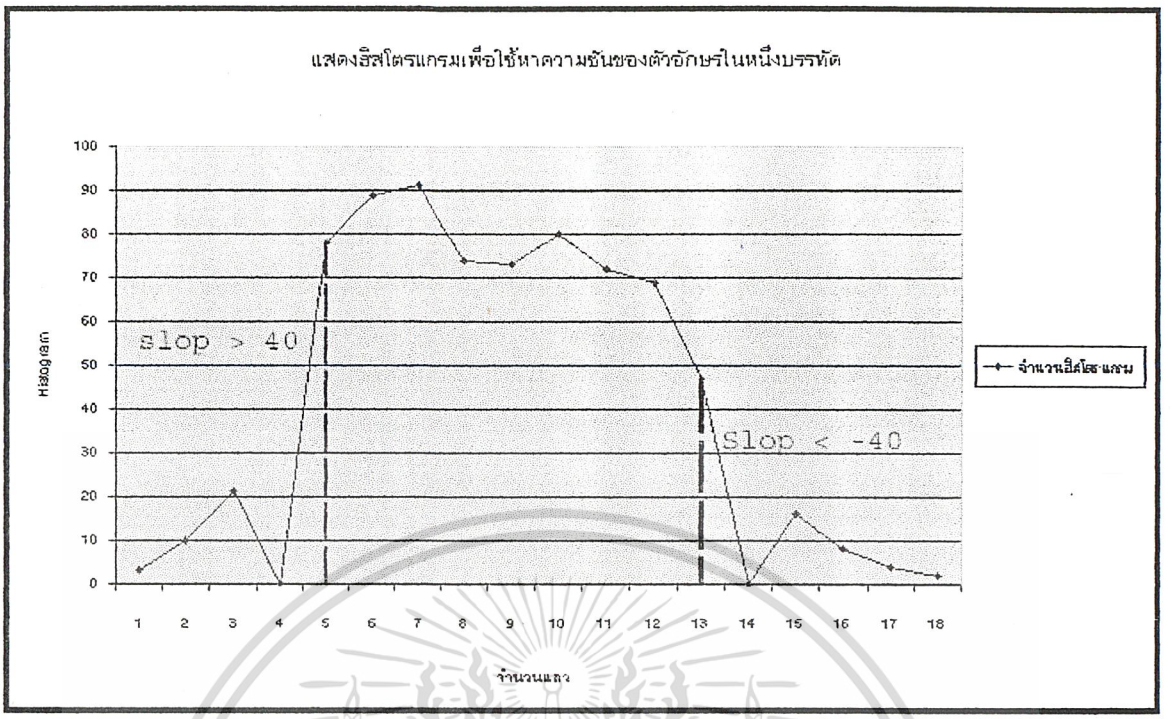
รูปที่ 5.13 แสดงตัวอักษรที่มีตัวไม่ติดกันและมีสระบนสองระดับอยู่



รูปที่ 5.14 แสดงตัวอักษรที่มีสระบนระดับเดียวและตัวติดกัน



รูปที่ 5.15 แสดงตัวอักษรระดับกลางที่เป็นตัวติดกัน



รูปที่ 3.16 แสดงค่าความชันเพื่อใช้หาบรรทัดบน

รูปที่ 3.17 แสดงรูปของความสูงของตัวอักษร

เมื่อได้ค่าความสูงของตัวอักษรมาแล้ว นำไปเปรียบเทียบเพื่อหาขนาดของตัวอักษรต่อไป โดยจะไล่เปรียบเทียบกับตัวอักษรเมื่อได้ขนาดของตัวอักษรแล้วจะนำค่าที่ได้ไปกำหนดค่าให้กับโปรแกรมโดยอัตโนมัติ

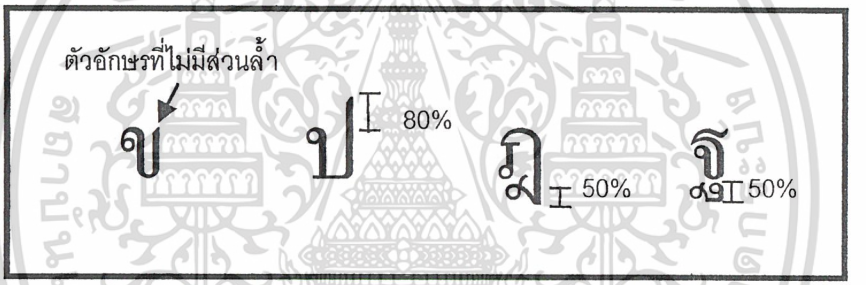
บทที่ 4
กระบวนการรู้จำตัวอักษร

เมื่อผ่านกระบวนการแยกอักษรออกมาเป็นตัวอักษรเดี่ยวๆได้แล้ว จะเข้าสู่กระบวนการการพิจารณาเปรียบเทียบค่าที่ได้จากการสแกนข้อมูลกับค่าที่เก็บไว้ในตัวอักษรต้นแบบเพื่อดูว่าข้อมูลดังกล่าวควรจะเป็นตัวอักษรอะไร

4.1 ขั้นตอนการเปรียบเทียบตัวอักษร

4.1.1 แนวคิดในการพิจารณาตัวอักษร

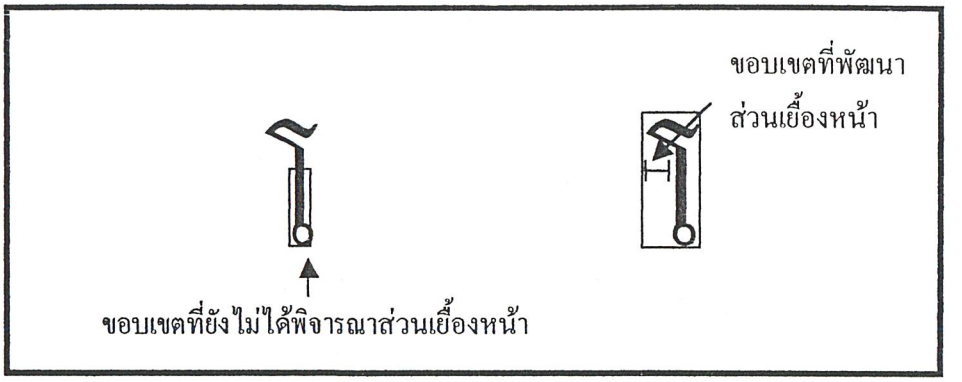
เริ่มต้นจากการสแกนหาขอบบนของบล็อคตัวอักษร และสแกนหาขอบซ้ายของบล็อคตัวอักษร จากนั้นทำการตรวจสอบตัวอักษรที่อยู่กลางบรรทัด ตัวอักษรจะมีส่วนล้ำขึ้นไปด้านบนได้ไม่เกิน 80 เปอร์เซ็นต์ของความสูงตัวอักษร และสามารถล้ำไปด้านล่างได้ไม่เกิน 50 เปอร์เซ็นต์ของความสูงตัวอักษรดังรูปที่ 4.1



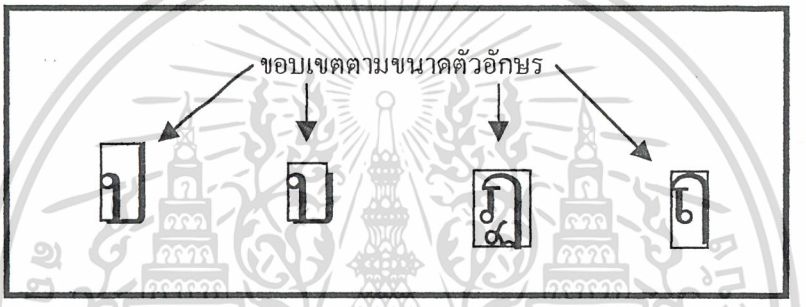
รูปที่ 4.1 แสดงรูปของตัวอักษร ฎ ป ฐ

4.1.2 กระบวนการเปรียบเทียบขั้นที่หนึ่ง

หลังจากพิจารณาตัวอักษรแล้วทำการกำหนดขอบเขตอีกครั้งตามขนาดของตัวอักษรจริงๆ ดังตัวอย่างในรูปที่ 4.3 เพราะอาจจะมีกรณีที่ตัวอักษรจะเยื้องไปทางด้านหน้า เยื้องไปทางด้านล่าง ล้ำขึ้นด้านบนและด้านล่างแล้วทำการ ตรวจสอบตัวอักษรที่อยู่กลางบรรทัดแบบที่ไม่มีส่วนล้ำเช่น ก,ข,ค,ม เป็นต้น และแบบพิเศษชนิดที่หนึ่งที่มีส่วนเยื้องขึ้นมาทางด้านหน้า ได้แก่ “โ”, “ไ”, “ใ” “เ” ซึ่งถ้าไม่พิจารณาขอบเขตของตัวอักษรที่เยื้องขึ้นมาจะทำให้ได้ขอบเขตดังในรูปที่ 4.2 ซึ่งเมื่อทำการเปรียบเทียบแล้วจะได้เป็น “เ” ทั้งหมด และแบบพิเศษชนิดที่สองที่มีส่วนขึ้นด้านบนหรือด้านล่าง ได้แก่ “พ”, “ฟ”, “ฝ”, “ฬ”, “บ”, “ป” ดังที่แสดงในรูปที่ 4.1 โดยทำการเปรียบเทียบตัวอักษรที่อยู่กลางบรรทัดกับตัวอักษรต้นแบบให้หมดซึ่งโดยทั่วไปแล้วจะสามารถเปรียบเทียบได้เกือบทั้งหมด แต่เนื่องจากตัวอักษรบางตัวมีลักษณะคล้ายกันมากทำให้ไม่สามารถแยกตัวอักษรเหล่านั้นได้ จึงจำเป็นต้องมีกระบวนการในการแยกความแตกต่างของตัวอักษรที่คล้ายกันออกมาดังตัวอย่างในการแยกความแตกต่างระหว่าง พ,ฟ,ป,บ ในรูปที่ 4.4 และรูปที่ 4.5 และจะกล่าวถึงรายละเอียดต่อไป



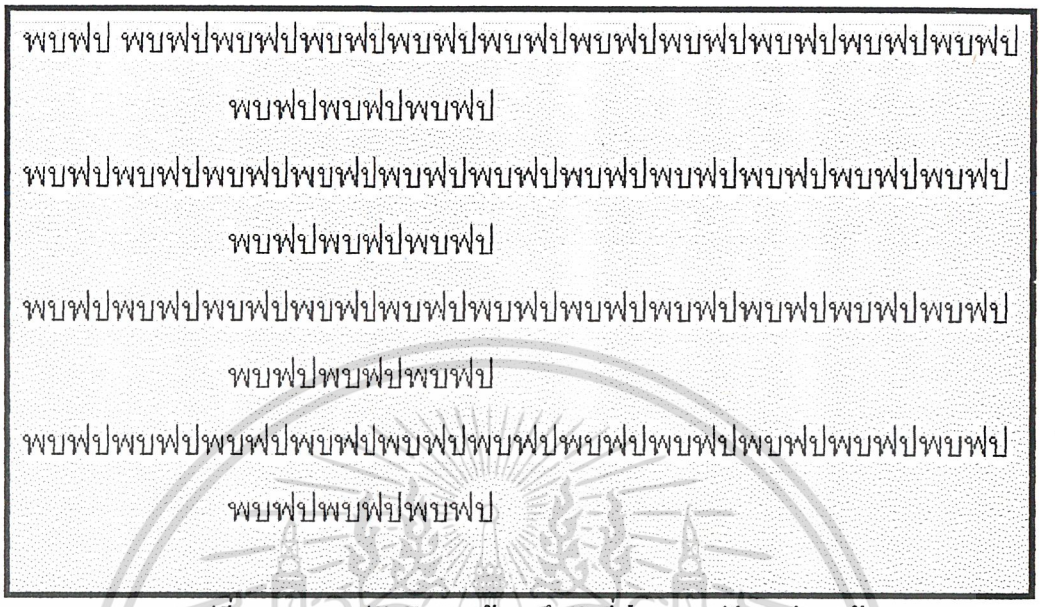
รูปที่ 4.2 แสดงขอบเขตของสระ “โ”



รูปที่ 4.3 แสดงรูปขอบเขตตามขนาดของตัวอักษร

พบฟป พบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟป
พบฟปพบฟปพบฟป
พบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟป
พบฟปพบฟปพบฟป
พบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟป
พบฟปพบฟปพบฟป
พบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟปพบฟป
พบฟปพบฟปพบฟป

รูปที่ 4.4 แสดงรูปของตัวอักษรที่มีลักษณะคล้ายคลึงกัน



รูปที่ 4.5 แสดงรูปตัวอักษรคล้ายคลึงกันที่ผ่านการเปรียบเทียบแล้ว

4.1.3 การแยกความแตกต่างระหว่างอักษรที่คล้ายคลึงกัน

เนื่องจากมีตัวอักษรหลายตัวที่มีลักษณะคล้ายคลึง ถ้าใช้วิธีการทำการเปรียบเทียบขั้นที่หนึ่งเพียงอย่างเดียวจะไม่สามารถแยกตัวอักษรที่มีลักษณะคล้ายกันได้ ดังกรณีต่อไปนี้

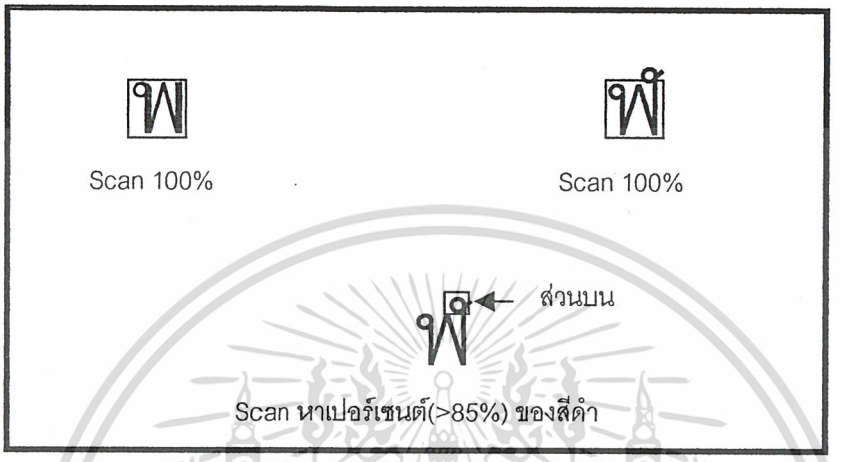
4.1.3.1 กรณี “โ”, “ไ”, “ใ” “เ”

ถ้าใช้กระบวนการขั้นที่หนึ่งอย่างเดียวจะไม่สามารถแยกความแตกต่างระหว่าง “เ” , “โ”,“ใ”, “ไ” เนื่องจากโดยกระบวนการขั้นที่หนึ่งจะเปรียบเทียบออกมาเป็น “เ” ทั้งหมดดังที่กล่าวไว้ในหัวข้อที่ 4.1.1 เนื่องจากการสแกนหาคอลัมน์นั้นถ้าเป็น “โ”, “ไ”, “ใ” จะเจอเป็น “เ” ดังนั้นจึงต้องหาขอบด้านซ้ายของ “โ”, “ไ”, “ใ” เพื่อนำมาอ้างอิงเปรียบเทียบกับ “เ” ว่ามีระยะที่จะต้องเลื่อนไปทางด้านซ้ายเท่าไหร่แล้วเก็บค่าดังกล่าวไว้ เมื่อได้ระยะที่ถอยไปทางด้านซ้ายแล้วจะต้องการโหลดข้อมูลมาเก็บไว้ใหม่อีกครั้งเนื่องจากข้อมูลเก่าจะเปรียบเทียบได้เฉพาะ “เ” เท่านั้น แล้วเมื่อโหลดข้อมูลใหม่มาแล้วจะนำกลับไปเปรียบเทียบอีกครั้งหนึ่งซึ่งถ้าตัวอักษรนั้นคือ “เ” ก็จะได้ผลลัพธ์ออกมาเป็น “เ” เหมือนเดิม แต่ถ้าเป็นตัวอักษร “โ”, “ไ”, “ใ” ก็จะได้ผลลัพธ์ออกมาเป็น “โ”, “ไ”, “ใ”

4.1.3.2 กรณี “พ” กับ “ฬ”

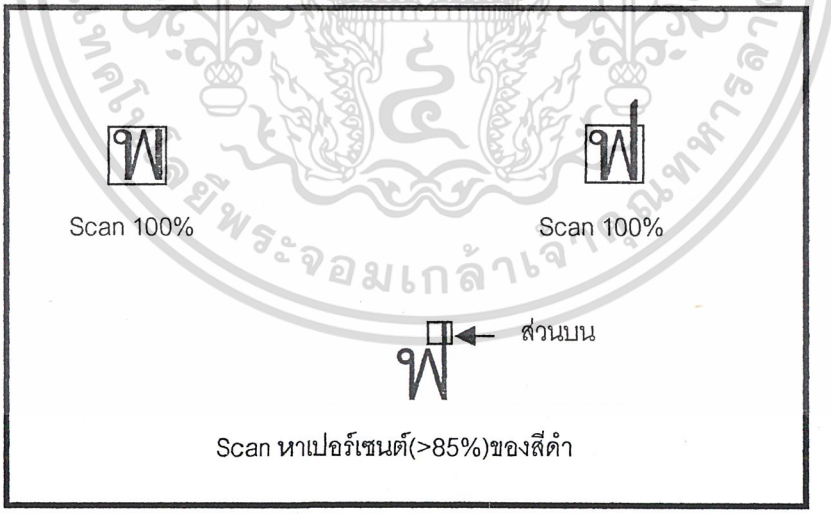
ถ้าใช้กระบวนการขั้นตอนที่หนึ่งอย่างเดียวจะไม่สามารถแยกความแตกต่างระหว่าง “พ” กับ “ฬ” ได้โดยกระบวนการขั้นที่หนึ่งจะเปรียบเทียบออกมาเป็น “พ” ทั้งหมด ดังนั้นจะต้องแบ่งการเปรียบเทียบออกเป็น 2 ส่วนคือส่วนที่หนึ่งเป็นตัวอักษรส่วนกลาง(Middle character) ซึ่งจะทำการสแกนทั้งส่วนที่เป็นเนื้อสีดำของตัวอักษรและส่วนที่เป็นพื้นที่ว่าง เช่นกรณีนี้ “ฬ” จะได้เป็น “พ” แล้วนำไปเปรียบเทียบเปอร์เซ็นต์ที่ตั้งเอาไว้ ซึ่งในกรณีนี้จะได้ออกมาเป็น 100 เปอร์เซ็นต์ ส่วนที่สอง เป็นส่วนต่อขึ้นไปข้างบน(Upper character) ซึ่ง

ในส่วนนี้จะทำการสแกนเฉพาะส่วนที่เป็นเนื้อสีดำของตัวอักษรเท่านั้นซึ่งถ้าพบว่ามีส่วนที่เป็นเนื้อสีดำมากกว่าเปอร์เซ็นต์ที่กำหนดไว้ (ระบบกำหนดไว้ที่มากกว่า 85%ตามข้อมูลที่ได้รวบรวมมา) ได้ผลลัพธ์ออกมาเป็นตัว “ฬ”



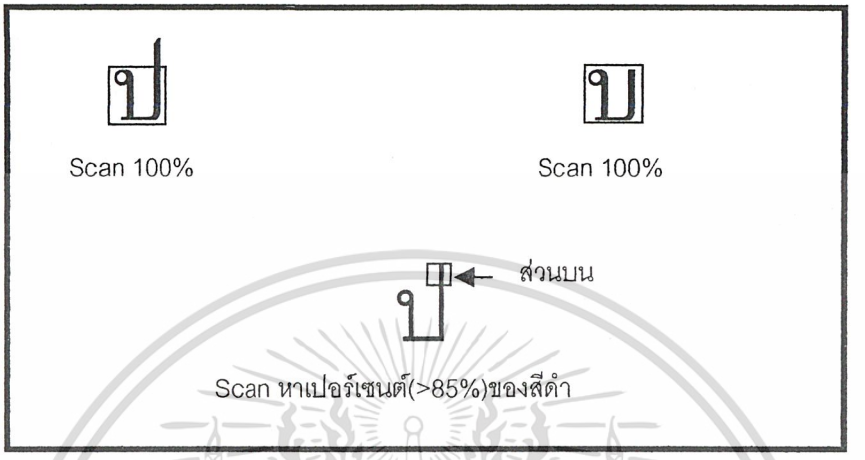
รูปที่ 4.6 แสดงรูป พ,ฬ

4.1.3.3 กรณี “พ” กับ “ฟ” จะมีหลักเกณฑ์การเปรียบเทียบคล้ายกับกรณี “พ” กับ “ฬ” เช่นกัน



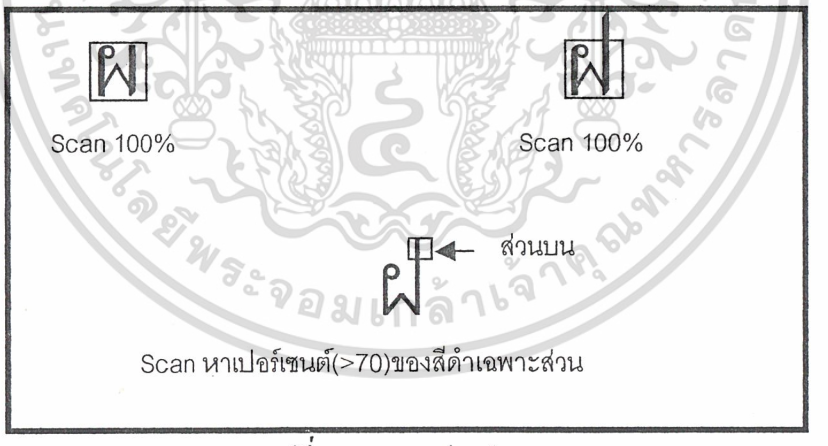
รูปที่ 4.7 แสดงรูป พ,ฟ

4.1.3.4 กรณี “บ” กับ “ป” จะมีหลักเกณฑ์การเปรียบเทียบคล้ายกับกรณี “พ” กับ “ฟ” เช่นกัน



รูปที่ 4.8 แสดงรูป บ,ป

4.1.3.5 กรณี “ผ” กับ “ฝ” จะมีหลักเกณฑ์การเปรียบเทียบคล้ายกับกรณี “พ” กับ “ฟ” เช่นกัน



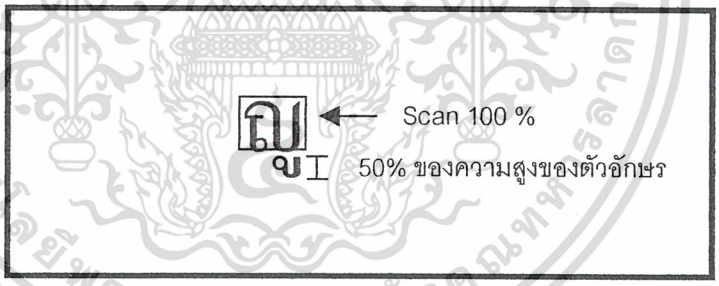
รูปที่ 4.9 แสดงรูป ผ,ฝ

4.1.4 การเปรียบเทียบตัวอักษรด้านล่างหรือสระล่าง

จากการสังเกตจะพบว่าตัวพยัญชนะในภาษาไทยเท่านั้นที่จะมีสระล่าง และตัวพยัญชนะที่ไม่มีสระล่างแน่นอน ได้แก่ “ฐ”, “ฎ”, “ฏ”, “ฤ”, “ฦ” จากนั้นจะทำการสแกนหาขอบบนของสระล่างโดยยึดเส้น Base line ที่หาได้ในตอนแรก โดยตั้งต้นที่ตำแหน่ง 1.8*ของความสูงของตัวอักษร ซึ่งพิจารณาว่าใน 1 คอลัมน์ประกอบด้วย ตัวอักษรระดับกลาง สระล่างและสระบน โดยที่สระบนจะมีความสูงประมาณ 80 เปอร์เซ็นต์ของความสูงของตัวอักษร และตัวอักษรระดับกลางบรรทัดมีความสูง

ประมาณ 100 เปอร์เซ็นต์ของความสูงของตัวอักษร ส่วนสระล่างจะกำหนดมีความสูงประมาณ 50 เปอร์เซ็นต์ของความสูงของตัวอักษรดังรูปที่4.10 ซึ่งค่าประมาณเหล่านี้ได้มาจากข้อมูลตัวอักษรที่รวบรวมไว้จากนั้นจะหาขอบบน และขอบซ้ายที่เริ่มต้นมีเนื้อสีดำของตัวอักษร จากนั้นจะทำการเปรียบเทียบว่าเป็นสระ “อุ” หรือ สระ “อู” และทำการตรวจสอบอีกครั้งว่าเป็นตัว “ญ” ที่ตามด้วยสระ “อุ” หรือไม่ ซึ่งคำดังกล่าวไม่มีในพจนานุกรม จะเปรียบเทียบออกมาเป็น “ญ” เท่านั้นแต่ในกรณีที่ เป็นตัว “ญ” ตามด้วยสระ “อู” เช่นคำว่า กตัญญู จะสามารถเปรียบเทียบได้ออกมาเป็น “ญ” และ สระ “อู” ตามลำดับดังรูปที่ 4.11

รูปที่ 4.10 แสดงส่วนสูงโดยประมาณของส่วนต่างๆของพยัญชนะ



รูปที่ 4.11 แสดงรูป ญ

4.1.5 การเปรียบเทียบตัวอักษรด้านบนหรือสระบน

การเปรียบเทียบตัวอักษรด้านบนหรือสระด้านบนจะต้องทำการโหลดข้อมูลเข้ามาใหม่ เนื่องจากสระบนนั้นโดยส่วนมากจะเหลื่อมไปทางด้านหน้าเล็กน้อย เช่น สระ “อิ” สระ “อี” เป็นต้น จะเหลื่อมตัวพยัญชนะไปเล็กน้อย ประมาณไม่เกิน 10 เปอร์เซ็นของความสูงของตัวอักษร ซึ่งค่าดังกล่าวได้มาจากข้อมูลตัวอักษรที่รวบรวมไว้

รูปที่ 4.12 แสดงสระ อิ,อี ให้เห็นชัดเจน

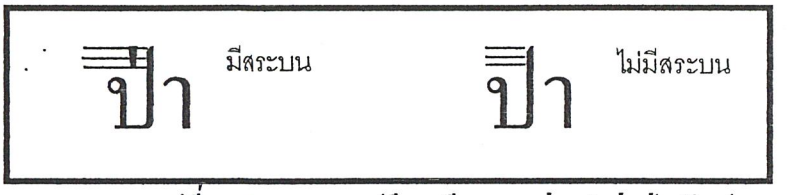
และมีสระบนบางตัวที่จะเหลื่อมไปทางด้านหลังเช่น ไม้หันอากาศ ไม้โท ซึ่งจะเหลื่อมไปทางด้านหลังเล็กน้อย ประมาณไม่เกิน 20 เปอร์เซ็นต์ของความสูงของตัวอักษรดังแสดงไว้ในรูปที่ 4.10 ซึ่งค่าดังกล่าวได้มาจากข้อมูลตัวอักษรที่รวบรวมไว้

รูปที่ 4.13 แสดงสระ ไม้หันอากาศ ไม้โท อย่างชัดเจน

จากนั้นจะต้องทำการตรวจสอบก่อนว่าตัวอักษร “ป”, “ฝ”, “ฟ”, “พ” มีสระบนหรือไม่ โดยทำการสแกนจากด้านขวาไปซ้ายจนสุดความกว้างของตัวอักษร เพราะถ้าตัวอักษรเหล่านี้มีสระบน สระบนของตัวพยัญชนะเหล่านี้จะมีลักษณะดังรูปที่ 4.14 คือจะอยู่ระหว่างตัวอักษร แต่ถ้าสแกนแล้วไม่พบเนื้อสีดำแสดงว่าตัวอักษรดังกล่าวไม่มีสระบน

รูปที่ 4.14 แสดง คำว่า ป่า ฝ่า

จากรูปที่ 4.14 จะเห็นว่าคำว่า “ฝ่า” หรือ “ป่า” ตำแหน่งของไม้เอกจะไม่เหมือนกับคำว่า “ด่า” หรือ “ว่า” คำว่า “ฝ่า” ไม้เอกจะอยู่ระหว่างตัว “ฝ” ไม่ได้อยู่กึ่งกลางระหว่าง “ฝ” กับสระอา ดังนั้นจะทำให้เสมือนไม่เห็นไม้เอกดังกล่าวในเวลาที่เปรียบเทียบกับตัวอักษรต้นแบบ ดังนั้นถ้าเป็นอักษร “ป”, “ฝ”, “ฟ”, “พ” สแกนจากด้านขวาไปซ้ายจนสิ้นสุดความกว้างของตัวอักษร แล้วไม่พบเนื้อสีดำเลยแสดงว่าไม่มีสระนั้นอยู่จริง หลังจากทำการตรวจสอบแล้ว จะทำการเปรียบเทียบกับตัวอักษรต้นแบบต่อไป

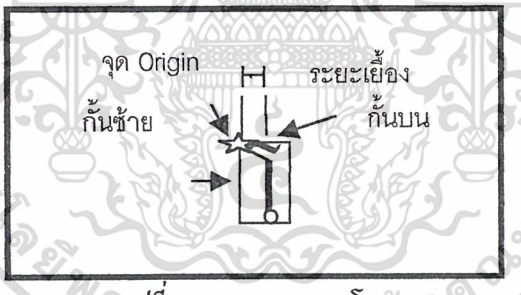


รูปที่ 4.15 แสดงการเปรียบเทียบระหว่าง คำว่า ป่า กับ ปา

ในขั้นตอนการเปรียบเทียบของไม้หันอากาศ ซึ่งจะใช้หลักการเดียวกับการเปรียบเทียบของไม้เอก เนื่องจากไม้หันอากาศจะเหลื่อมต่ำพยัญชนะไปทางด้านหลังประมาณ 20 เปอร์เซ็นต์ของความสูงของตัวอักษร ดังที่แสดงไว้ในรูปที่ 4.13 และนำมาหาขอบบนและขอบด้านซ้าย เพื่อใช้เปรียบเทียบกับตัวอักษรต้นแบบต่อไปหลังจากพิจารณาตัวไม้หันอากาศแล้ว จะพิจารณาสระ “อิ”, “อี”, “อึ”, “อื” ตามลำดับ โดยวิธีการเดียวกันกับวิธีข้างต้น เพราะสระดังกล่าวจะมีส่วนเหลื่อมตัวอักษรไปทางด้านหลังได้เช่นเดียวกับกรณีของตัวไม้หันอากาศ

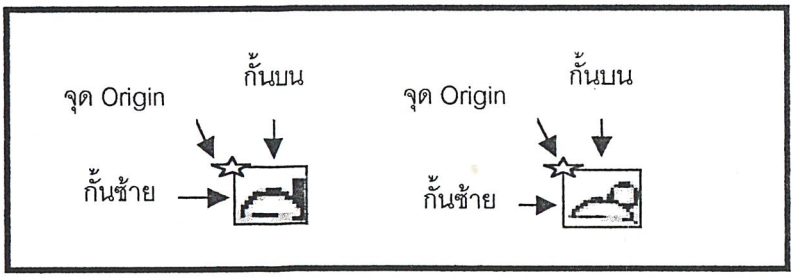
4.1.6 ขั้นตอนการหาจุดเริ่มต้น (Origin) ของตัวอักษร

การหาจุดเริ่มต้น (Origin) ของตัวอักษร เป็นการหาจุดอ้างอิงเพื่อทำให้เราสามารถทำการเปรียบเทียบกับตัวอักษรที่สแกนเข้ามากับตัวอักษรต้นแบบได้ ในกรณีปกติจุดเริ่มต้นขอบตัวอักษร(Origin) ทั่วๆไป เช่น “ก”, “อ” จะอยู่ที่ตำแหน่งที่คอลัมน์ตัดกับเส้น Baseline-FontHigh (ความสูงของตัวอักษร) ซึ่งจะอยู่ตรงมุมบนด้านซ้ายของตัวอักษร ยกเว้นสระ “โ”, “ไ”, “ใ” ที่จะมีส่วนที่เยื้องไปทางด้านหน้า ซึ่งเราต้องพิจารณาหาระยะที่เยื้องอีกครั้ง จึงจะได้จุดเริ่มต้น(Origin) จริงๆดังแสดงไว้ในรูปที่ 4.16



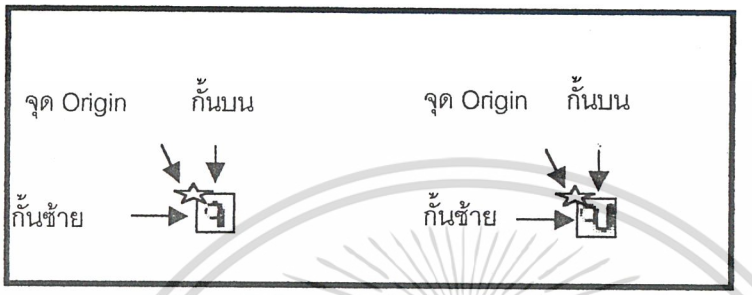
รูปที่ 4.16 แสดง สระ โ

การหาจุดเริ่มต้น(Origin) ของสระบน จะต้องทำต้องทำการสแกนเพื่อหาขอบซ้ายและขอบบนก่อน เพื่อใช้กำหนดเป็นจุดเริ่มต้น(Origin) ดังนั้นสระบนจะต้องทำการหาจุดเริ่มต้นเป็นตัวๆไปดังรูปที่ 4.17 ซึ่งจะแตกต่างกับการหาจุดเริ่มต้น(Origin)ของตัวอักษร เมื่อเราได้ค่าคอลัมน์มาแล้วจะได้ค่าขอบซ้ายเลย ยกเว้นในกรณีสระ “โ”, “ไ”, “ใ” ที่กล่าวไว้ข้างต้น



รูปที่ 4.17 แสดง สระบน

การหาจุดเริ่มต้น(Origin)ของสระล่างจะมีขั้นตอนเหมือนกับการหาจุดเริ่มต้น(Origin)ของสระบน โดยจะทำสแกนเข้ามาใหม่เพื่อทำการหาขอบซ้ายและขอบบน เพื่อกำหนดเป็นจุดเริ่มต้น(Origin) ต่อไปดังรูปที่ 4.18



รูปที่ 4.18 แสดง สระล่าง

4.1.7 การส่งข้อมูลเพื่อไปทำการแปล

จะทำการส่งข้อมูลไปแปลโดยจะหาขอบเขตของบล็อคข้อมูลแล้วส่งไปแปล โดยจะโหลดข้อมูลเข้ามาทีละบล็อค โดยขนาดของบล็อค ด้านกว้างจะมีความยาวไม่เกิน 1.8 เท่าของความสูงของตัวอักษรและด้านยาวจะมีความยาวไม่เกิน 0.5 เท่าของความสูงของตัวอักษร ถ้าความกว้างของคอลัมน์ ณ ตำแหน่งปัจจุบันเล็กกว่าความกว้างของคอลัมน์ที่ผ่านมา ให้ลดขนาดของบล็อคลง แต่ถ้า ณ ตำแหน่งของคอลัมน์ดังกล่าวไม่มีตัวอักษรอยู่เลยแสดงว่าเป็นตำแหน่งของเว้นวรรคข้อความหรือย่อหน้า จะต้องทำการคำนวณหาระยะโดยเทียบมาจากค่าประมาณของสเปสบาร์รวมกับช่องว่างระหว่างตัวอักษรดังกล่าว ซึ่งใช้สมการดังต่อไปนี้ในการคำนวณ

```
Onespace = FontHigh*0.67
```

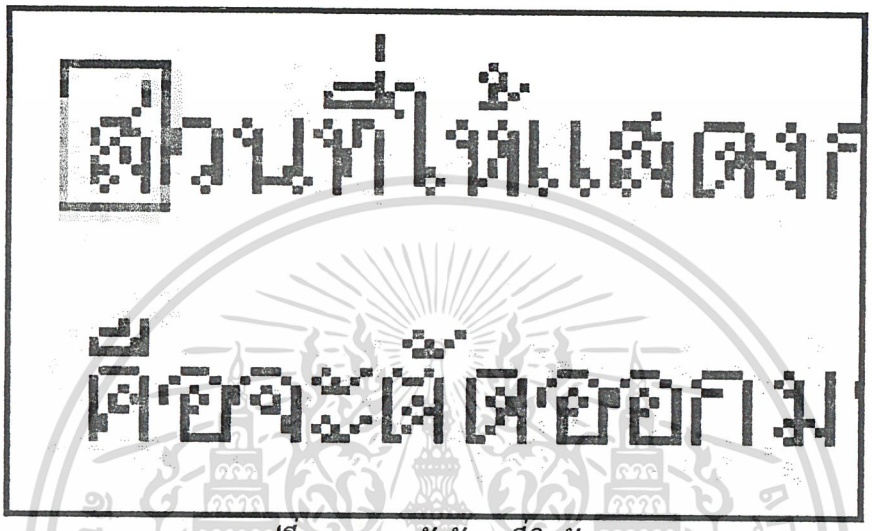
เพื่อทำการกระโดดข้ามไปเปรียบเทียบยังส่วนที่มีตัวอักษรต่อไป หลังจากนั้นจะทำการเปรียบเทียบตัวอักษรที่โหลดไว้ใน บล็อคกับตัวอักษรต้นแบบที่โปรแกรมได้คำนวณหาไว้แล้ว

กรณีที่เป็นสระ “แอ” จะต้องทำการตรวจสอบตัวถัดไปอีก 1 ตัวเนื่องจากจะทำการเปรียบเทียบได้เป็นสระ“เอ” สองตัวติดกัน ถ้าดูโดยผิวเผินดูเหมือนว่าโปรแกรมไม่ได้ผิดพลาด แต่ถือว่าผิดพลาดทางอัลกอลิทึม ส่วนในกรณีที่เป็นสระ “อำ” กับสระ “อา” ในตอนแรกจะเปรียบเทียบออกมาเป็นสระ “อา” ทุกครั้งก่อน หลังจากนั้นจะทำการตรวจสอบหานคลิขิต ถ้ามีนคลิขิตจะเปลี่ยนสระ “อา” ดังกล่าว ให้มีค่าเป็น สระ “อำ”

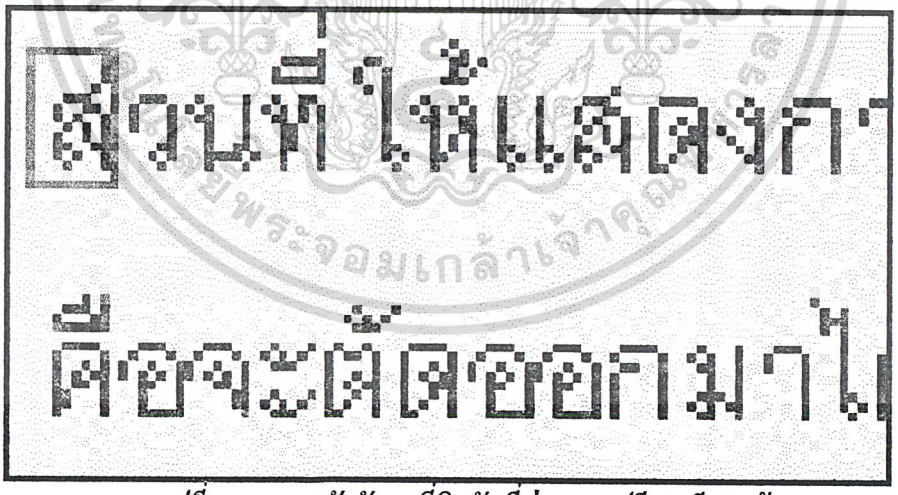
4.1.8 การเปรียบเทียบตัวอักษรที่ติดกัน

ในกรณีมีตัวอักษรที่ตัวติดกันดังเช่นตัวอย่างรูปที่ 4.19 จะไม่สามารถแยกออกจากกันด้วยการสแกนคอลัมน์ได้ดังแสดงไว้ในรูปที่ 4.21 ดังนั้นจะต้องทำการเปรียบเทียบตัวอักษรตัวแรกกับตัวอักษรต้นแบบเสียก่อน เมื่อเปรียบเทียบเสร็จแล้วจึงหาความกว้างของตัวอักษรตัวแรกที่ทำการเปรียบเทียบได้แล้ว จากนั้นจะตัด

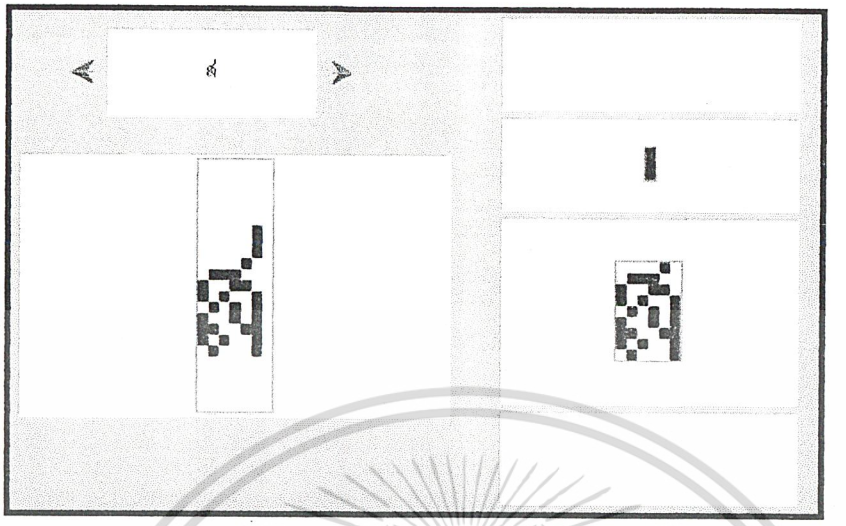
ส่วนนี้ออกไปแล้วทำการเปรียบเทียบตัวอักษรที่อยู่ถัดไป ทำวิธีการนี้ไปเรื่อยๆจนกระทั่งไม่พบตัวอักษรที่ติดกันอีก



รูปที่ 4.19 แสดงตัวอักษรที่ติดกัน



รูปที่ 4.20 แสดงตัวอักษรที่ติดกันที่ผ่านการเปรียบเทียบแล้ว



รูปที่ 4.21 แสดงรูปของตัวอักษรติดกันจริง

หลังจากนั้นจะทำการเปรียบเทียบกับตัวอักษรต้นแบบอีกครั้ง จากนั้นจะทำการคำนวณหาขอบซ้ายของตัวอักษรตัวถัดไป อาศัยหลักการที่ว่าขนาดของตัวอักษรระดับกลางที่อยู่ถัดไปนั้นจะต้องมีขนาดมากกว่าหรือเท่ากับ 25 เปอร์เซ็นต์ของความสูงของตัวอักษรซึ่งค่านี้ได้จากข้อมูลของตัวอักษรที่รวบรวมไว้ เมื่อได้ขอบซ้ายของตัวอักษรทุกๆตัวแล้วจะสามารถนำไปใช้ในการเปรียบเทียบกับตัวอักษรต้นแบบได้หมดหน้าเอกสาร จากนั้นจะได้ค่าอินพุทออกมาว่ารูปตัวอักษรดังกล่าวเป็นตัวอักษรใดดังรูปที่ 4. 20

บทที่ 5

การทำงานและทดลอง

5.1 การทำงานของโปรแกรมรู้จำตัวอักษรภาษาไทย

การทำงานในขั้นตอนแรกของโปรแกรม เริ่มจากการรับข้อมูลในหน้าเอกสาร หาขนาดของตัวอักษรในหน้าเอกสาร ความสูงของตัวอักษร ความสูงของบรรทัดโดยอัตโนมัติ แล้วจะโหลดค่าชนิดของตัวอักษรต้นแบบมาเก็บไว้ในหน่วยความจำ ทำการสแกนหา Base line จะได้ว่ามีเส้นบรรทัดอยู่ที่ไหนบ้าง ทำการสแกนหาคอลัมน์ทีละบรรทัด ก็จะได้ขอบเขตของตัวอักษรซึ่งอาจจะมีตัวเดียวหรือหลายตัวปนกันอยู่ จากนั้นจะทำการเปรียบระหว่างขอบเขตของตัวอักษรที่ได้กับตัวอักษรต้นแบบที่เก็บไว้ในหน่วยความจำ ดังนั้นผลลัพธ์ที่ได้จะเป็นแฟ้มข้อมูล txt file

5.1.1 อุปกรณ์ที่ต้องนำมาใช้ในการทำงานของโปรแกรม

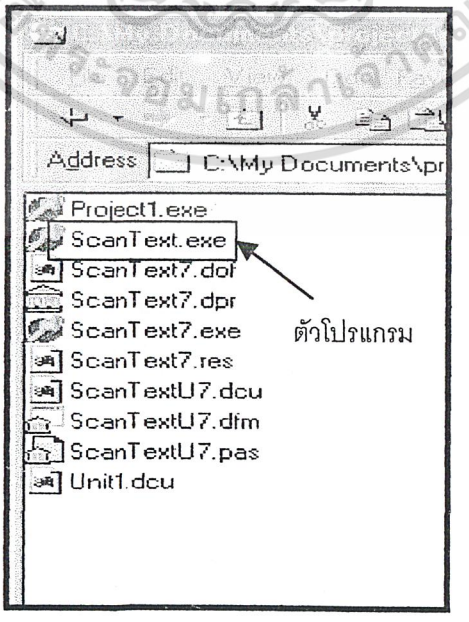
1. เครื่องไมโครคอมพิวเตอร์ CPU Pentium ความเร็ว 200 MHz หน่วยความจำ 32 MB
2. จอภาพแสดงผล
3. ฮาร์ดดิส 1 ตัว มีเนื้อที่ว่าง อย่างน้อย 2 Mbytes ขึ้นไป
4. เครื่องตรวจกวาดภาพ (Scanner) ยี่ห้อ Umax รุ่น PowerLook 3000 optical resolution 3048X3048 dpi

5.2 ขั้นตอนการใช้งานของโปรแกรม

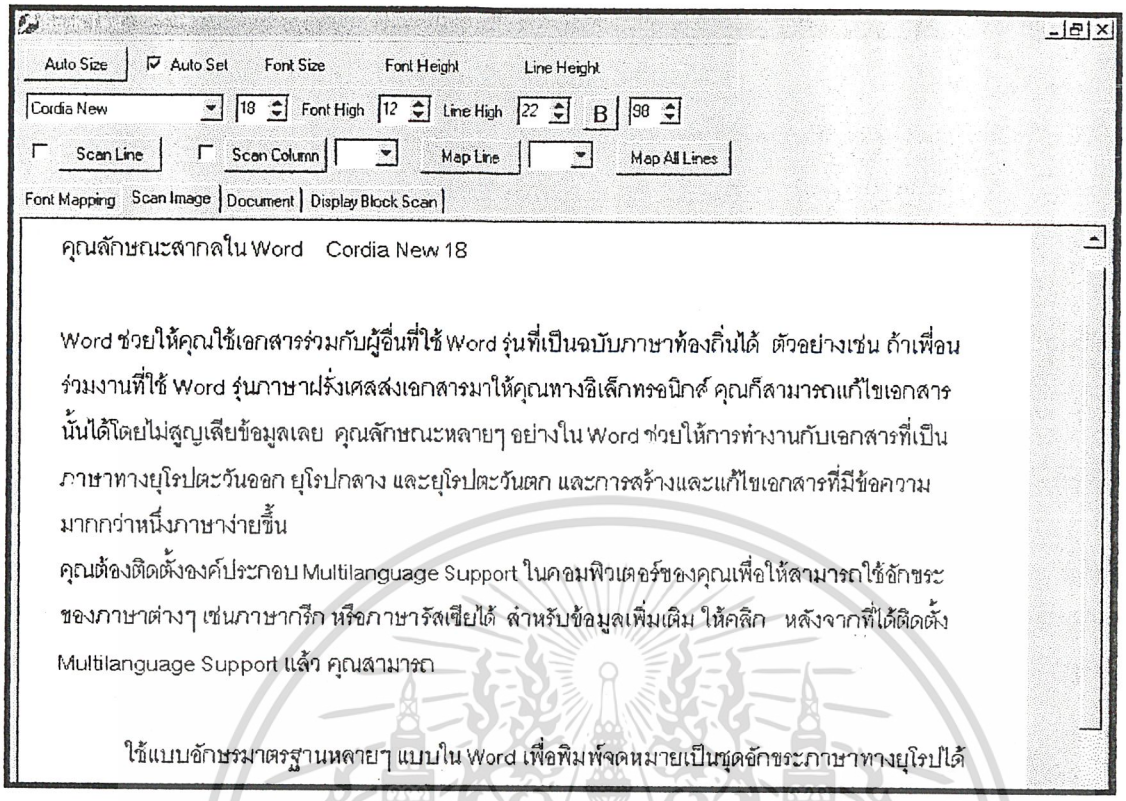
การทำงานของโปรแกรมจะเป็นการทำงานผ่านปุ่มการทำงานต่างๆ โดยจะมีขั้นตอนดังต่อไปนี้

ขั้นที่ 1 การเรียกใช้การทำงานของโปรแกรม

คลิ๊กที่ตัวโปรแกรมดังรูปที่ 5.1 แล้วโปรแกรมจะเริ่มทำงาน โดยมีปุ่มการทำงานต่างๆ เป็นตัวควบคุมการทำงาน



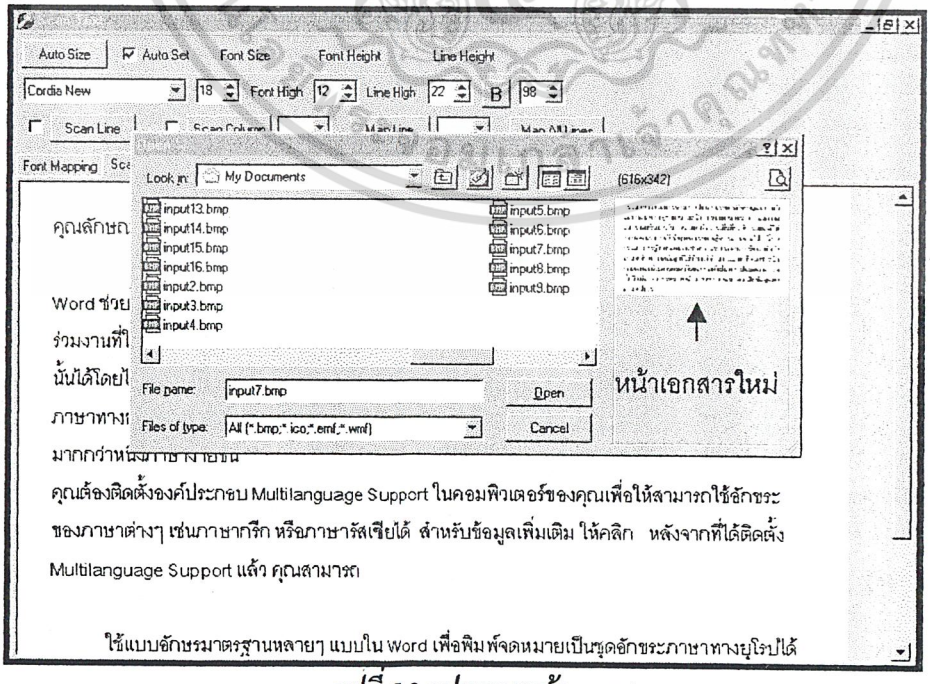
รูปที่ 5.1 แสดงไอคอนของ program



รูปที่ 5.2 แสดงการทำงานของโปรแกรมเมื่อเริ่มต้นทำงาน

ขั้นที่ 2 การเปิดภาพหน้าเอกสาร

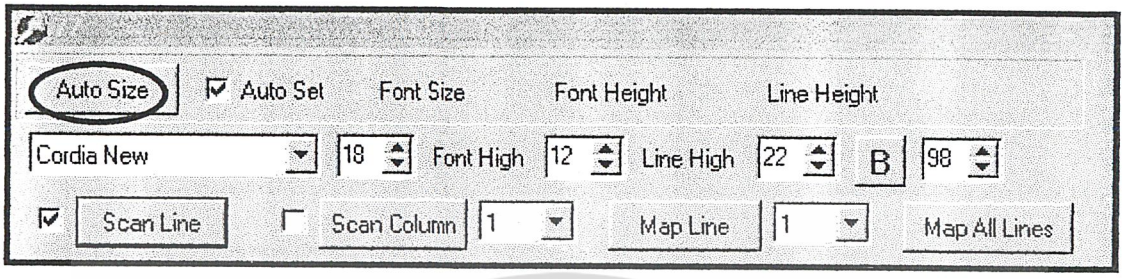
โดยการดับเบิ้ลคลิ๊กที่รูปภาพของหน้าเอกสารในส่วนของหน้า Scan Image เพื่อทำการเปิดรูปภาพหน้าเอกสารใหม่ได้ โดยรูปของหน้าเอกสารจะเป็น Bmp file โหมดการทำงานเป็น RGB



รูปที่ 5.3 รูปแสดงหน้าscan image

ขั้นที่ 3 การกำหนดขนาดและความสูงของตัวอักษรโดยอัตโนมัติ

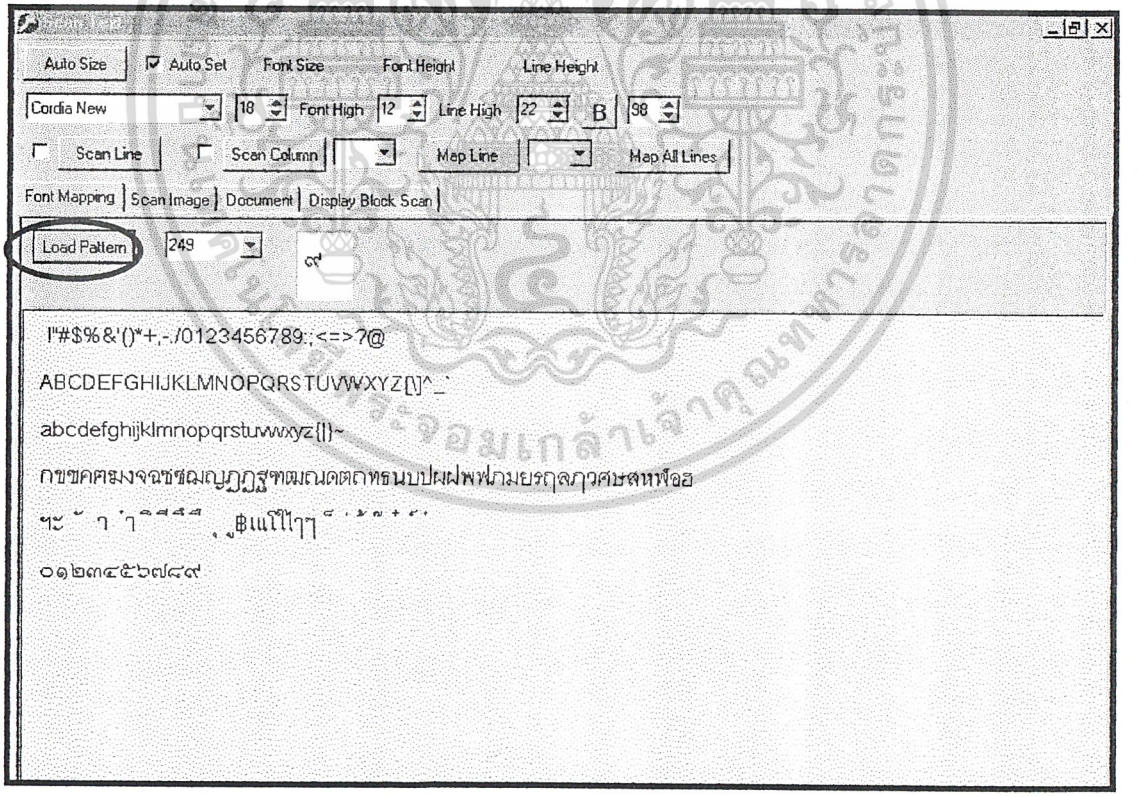
ในขั้นตอนนี้ จะทำงานได้โดยการกดปุ่ม Auto Size โปรแกรมจะทำการหาขนาดและความสูงของตัวอักษรในรูปภาพของหน้าเอกสาร โดยอัตโนมัติ เพื่อใช้ในการโหลดตัวอักษรต้นแบบ



รูปที่ 5.4 รูปแสดงส่วน Auto size ของโปรแกรม

ขั้นที่ 4 การโหลดตัวอักษรต้นแบบมาไว้ในหน่วยความจำ

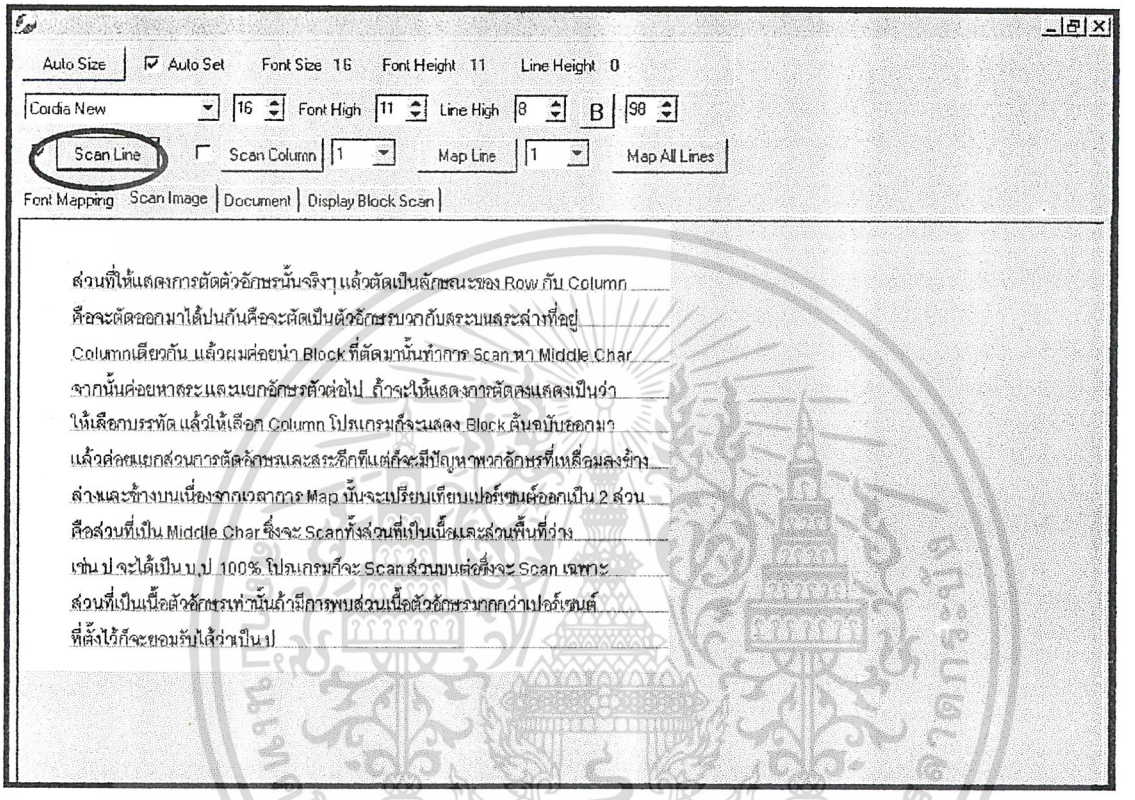
จะทำการเลือกชนิดของตัวอักษรและขนาดของตัวอักษร แล้วโหลดตัวอักษรต้นแบบมาไว้ในหน่วยความจำดังรูปที่5.5 จะทำงานได้โดยการกดปุ่ม Load Pattern ในหน้า Font Mapping



รูปที่ 5.5 แสดงการโหลดตัวอักษรต้นแบบในหน้า font Mapping

ขั้นที่ 5 การหา Base line

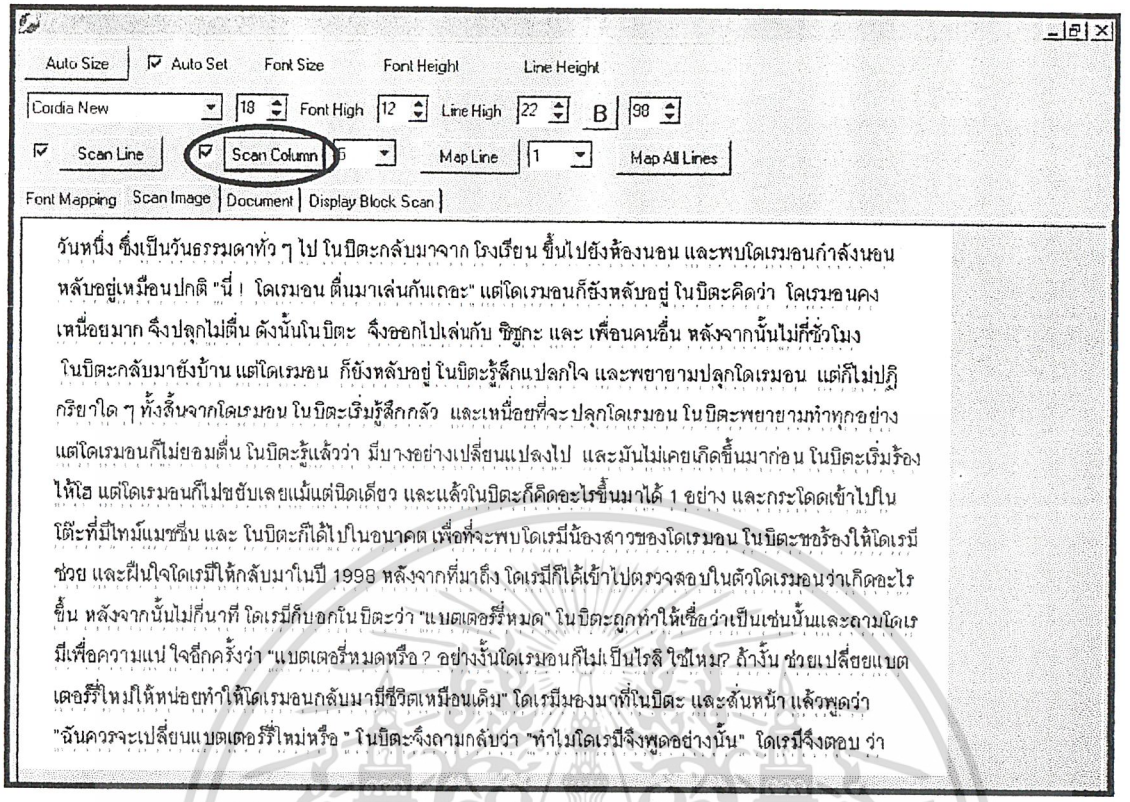
ในขั้นตอนที่ 5 นี้จะทำงานได้จะต้องผ่านขั้นตอนการทำงานการเปิดภาพหน้าเอกสารในขั้นตอนที่ 2 มาก่อน โดยโปรแกรมจะทำการวิเคราะห์หา Base line ของตัวอักษรในหน้าเอกสาร ทำได้โดยเลือกกดปุ่มการทำงาน Scan line ถ้าต้องการแสดงเส้นของ Base line ในภาพหน้าเอกสาร สามารถ check block ด้านหน้าของปุ่ม Scan lineดังแสดงในรูปที่5.6



รูปที่ 5.6 แสดงหน้า scanline ในหน้า scan image

ขั้นตอนที่ 6 การหาคอลัมน์ (Column)

ในขั้นตอนที่ 6 นี้จะทำงานได้จะต้องผ่านขั้นตอนการทำงานการเปิดภาพหน้าเอกสารในขั้นตอนที่ 2 มาก่อนเช่นกัน โดยโปรแกรมจะทำการวิเคราะห์เพื่อหาคอลัมน์ (column) ของตัวอักษรในหน้าเอกสาร ทำได้โดยเลือกกดปุ่มการทำงาน Scan column ถ้าต้องการให้แสดงพิกัดของคอลัมน์ (column) ในหน้าเอกสาร สามารถ check box ด้านหน้าของปุ่ม Scan column ดังแสดงในรูปที่ 5.7



รูปที่ 5.7 แสดงหน้า Scan column

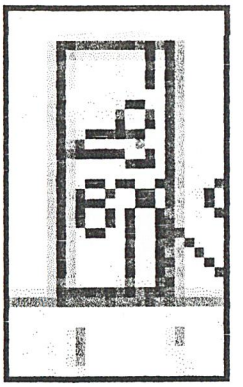
ขั้นตอนที่ 7 การเปรียบเทียบตัวอักษรในภาพหน้าเอกสารกับตัวอักษรต้นแบบ

ในขั้นตอนที่ 7 จะเป็นการทำการเปรียบเทียบตัวอักษรในภาพหน้าเอกสารกับตัวอักษรต้นแบบ (Pattern) โดยสามารถเลือกทำการเปรียบเทียบทีละบรรทัดโดย check box หน้าปุ่ม Map Line และเลือกบรรทัดที่ต้องการ หรือกดปุ่มMap All Lines เพื่อทำการเปรียบเทียบทั้งหมดของหน้าเอกสารได้

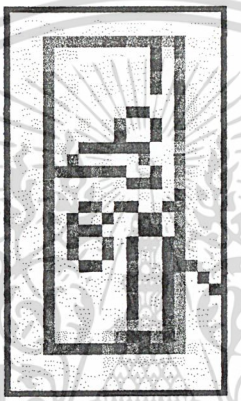
กระบวนการต่างๆที่อยู่ในขั้นตอนการเปรียบเทียบตัวอักษรในภาพหน้าเอกสารกับตัวอักษรต้นแบบที่เก็บไว้ในหน่วยความจำ

- เปรียบเทียบตัวอักษรระดับกลาง
- เปรียบเทียบตัวอักษรระดับล่าง ได้แก่ สระล่าง ทั้งหมด
- เปรียบเทียบตัวอักษรระดับบนชั้นที่ 2 และเปรียบเทียบตัวอักษรระดับบนชั้นที่ 1 ดังรูปที่ 5.10
- ทำการตัดขอบเขตตัวอักษรที่ผ่านการเปรียบเทียบแล้วออกไป
- พิจารณาคอลัมน์ที่อยู่ถัดไปจนหมดบรรทัดนั้น
- พิจารณาบรรทัดถัดไปจนกระทั่งหมดหน้าเอกสาร

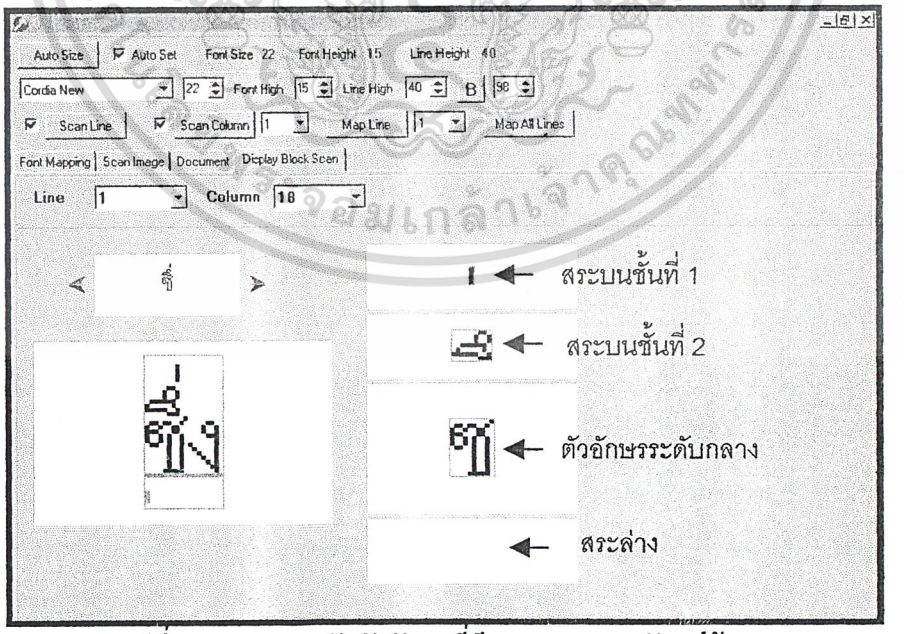
ในรูปที่ 5.8แสดงหน้าเอกสารที่มีสระสองชั้นอยู่ และรูปที่ 5.9 แสดงสระสองชั้นที่ผ่านการเปรียบเทียบแล้ว



รูปที่ 5.8 แสดงส่วนของรูปภาพตัวอักษรที่มีสระบนสองระดับ



รูปที่ 5.9 แสดงผลลัพธ์ของรูปภาพที่มีสระบนสองระดับ

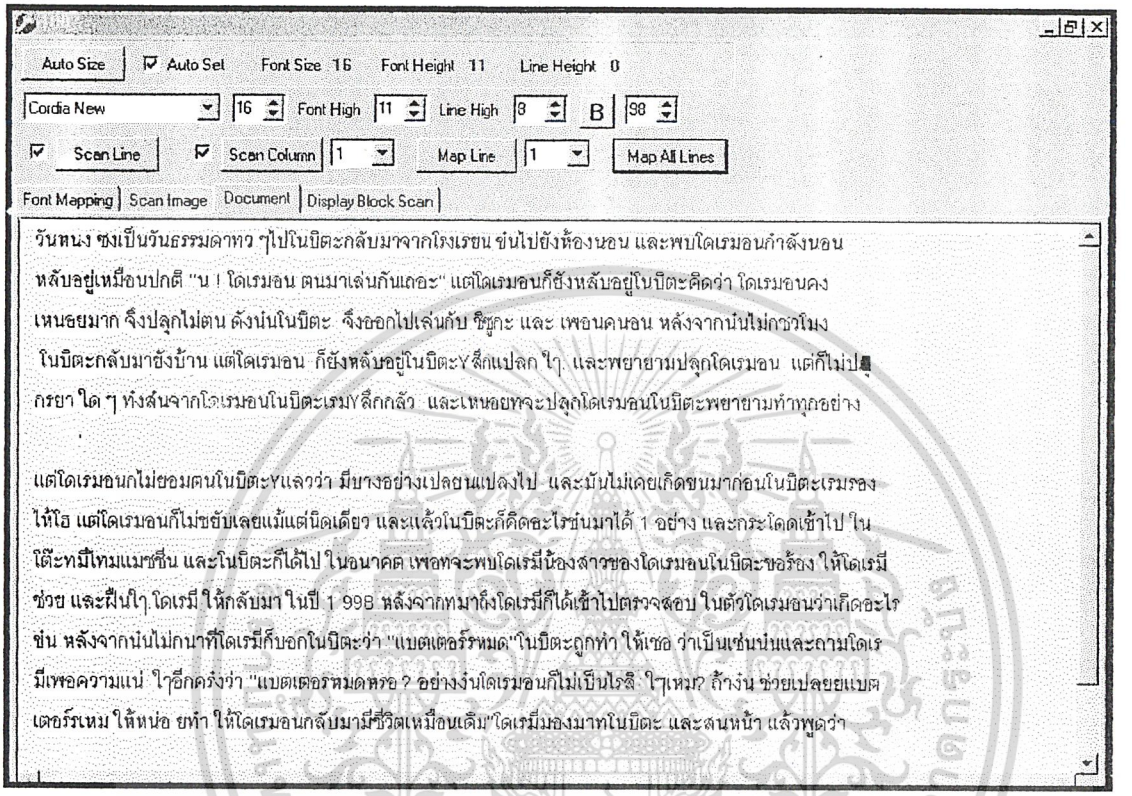


รูปที่ 5.10 แสดงการตัดตัวอักษรที่มีสระบนสองระดับอยู่ด้วย

ขั้นตอนที่ 8 การแสดงผล

การแสดงผลจะมีให้เลือกในหน้าต่าง ดังต่อไปนี้

1. ภายในหน้าต่างของ document จะแสดงภาพตัวอักษรในหน้าเอกสารที่ผ่านการเปรียบเทียบแล้ว และแสดงผลออกมาเป็นแฟ้มข้อมูลประเภท txt file ดังรูปที่ 5.11



รูปที่ 5.11 แสดงผลลัพธ์หน้า document

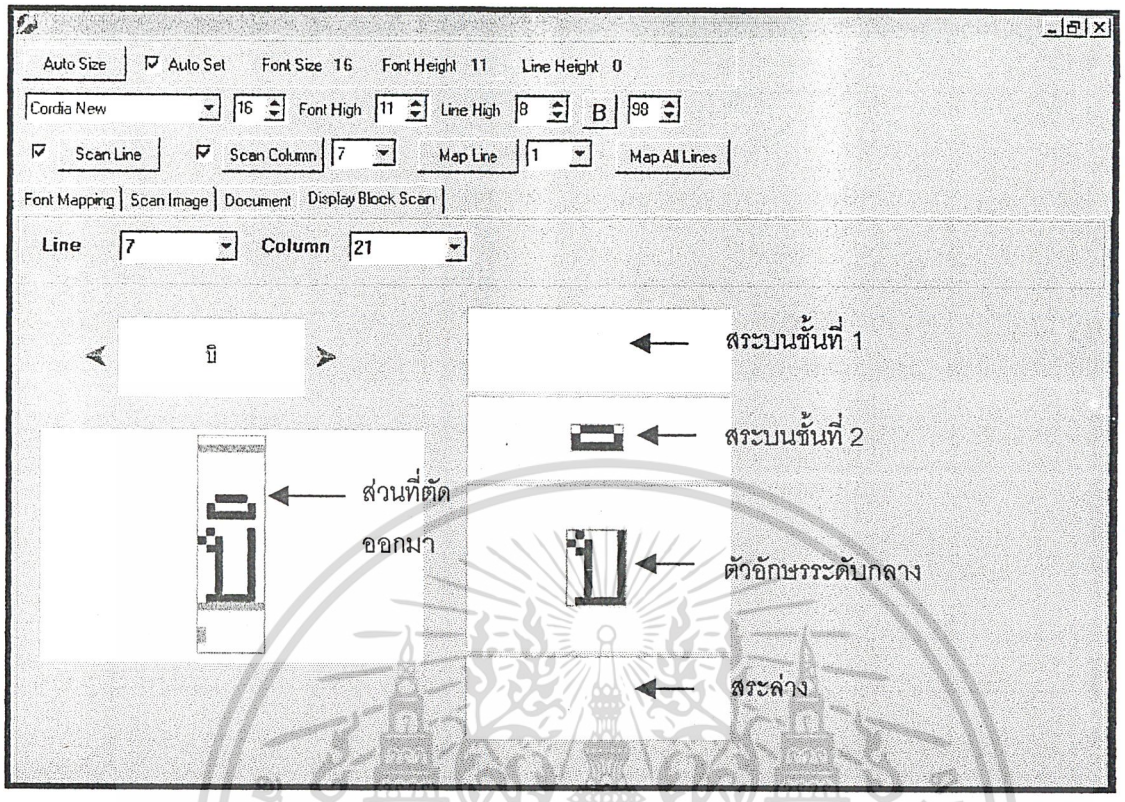
2. หน้าต่าง Display block Scan แสดงผลออกมาเป็นตัวอักษรที่ตัดออกมาแล้ว จะแสดงผลทีละตัว ในแนวของคอลัมน์นั้นเอง ดังนั้นในบางกรณีจะมีตัวอักษรที่ติดกันด้วย แบ่งออกเป็นกรณีดังได้ต่อไปนี้

รูปที่ 5.12 แสดงตัวอักษรที่ไม่มีส่วนติดกันและมีสระบนเพียงชั้นเดียวและแยกออกมาเป็นตัวๆได้

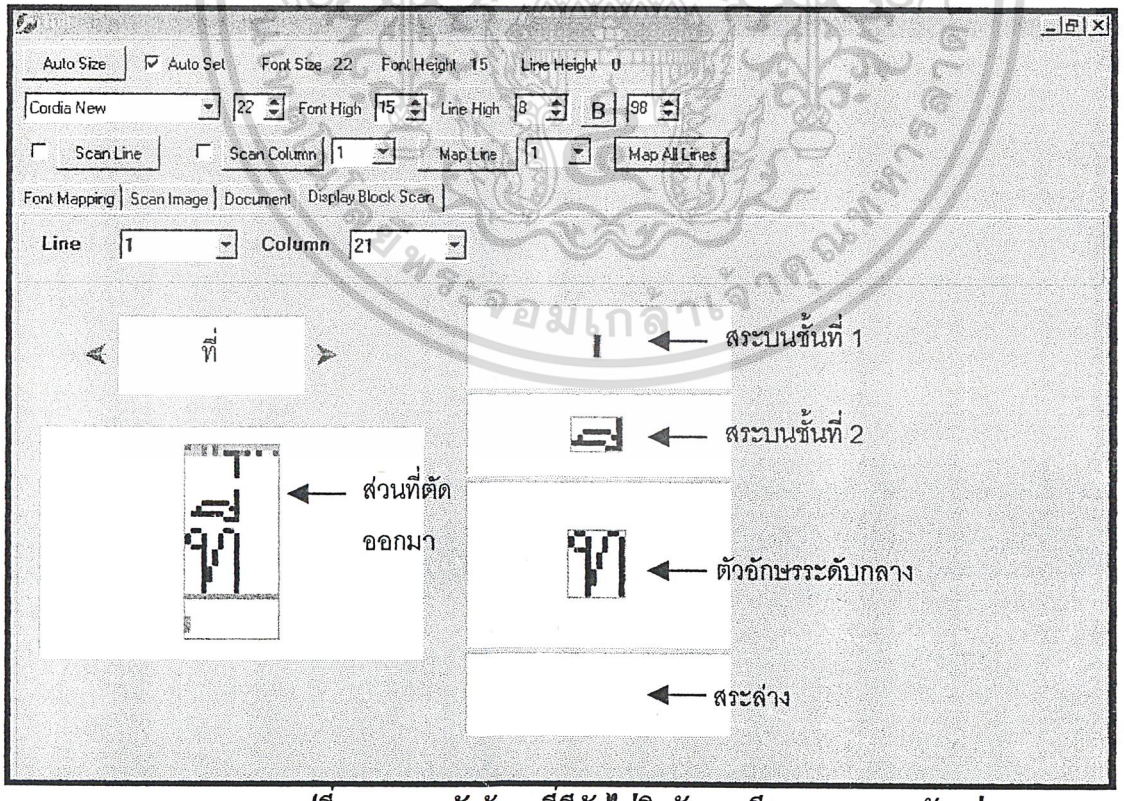
รูปที่ 5.13 แสดงตัวอักษรที่ไม่มีส่วนติดกันและมีสระบนสองชั้นและแยกออกมาเป็นตัวๆได้

รูปที่ 5.14 แสดงตัวอักษรที่มีส่วนติดกันและมีสระบนเพียงชั้นเดียวและสามารถแยกออกมาเป็นตัวๆได้

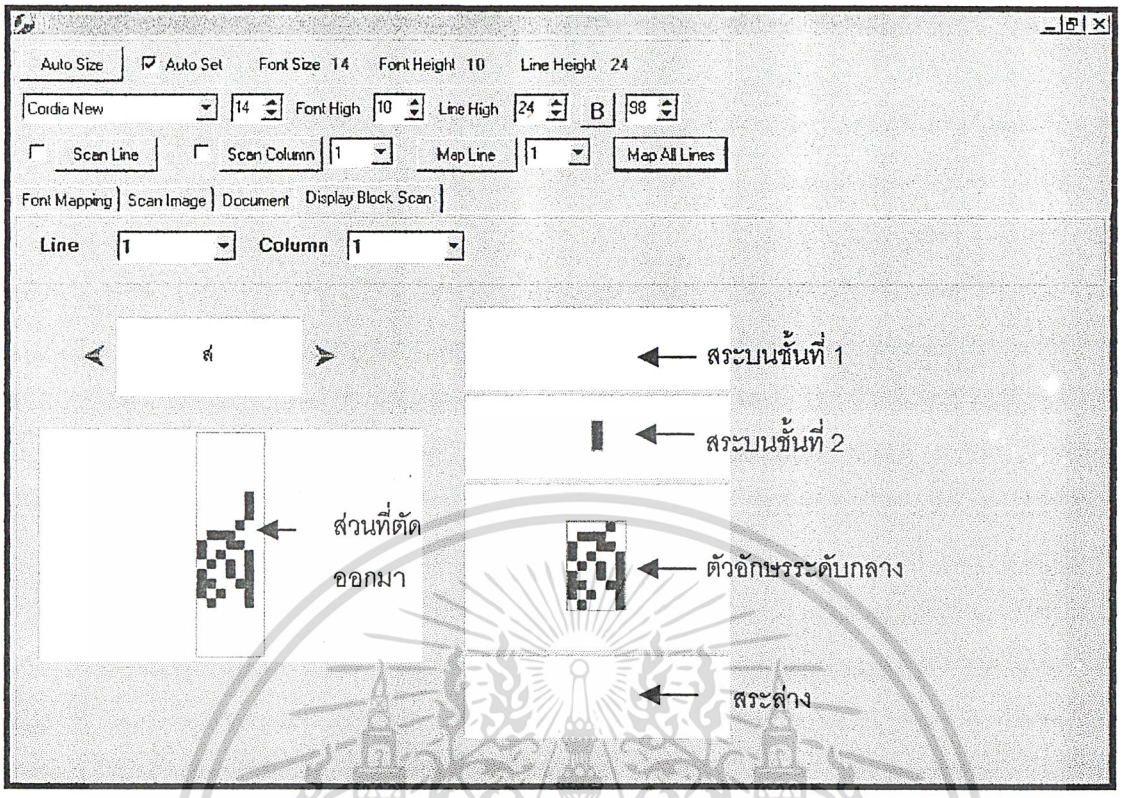
รูปที่ 5.15 แสดงตัวอักษรที่มีตัวอักษรระดับกลางติดกัน ในกรณีนี้จะไม่สามารถแยกออกมาเป็นตัวๆได้



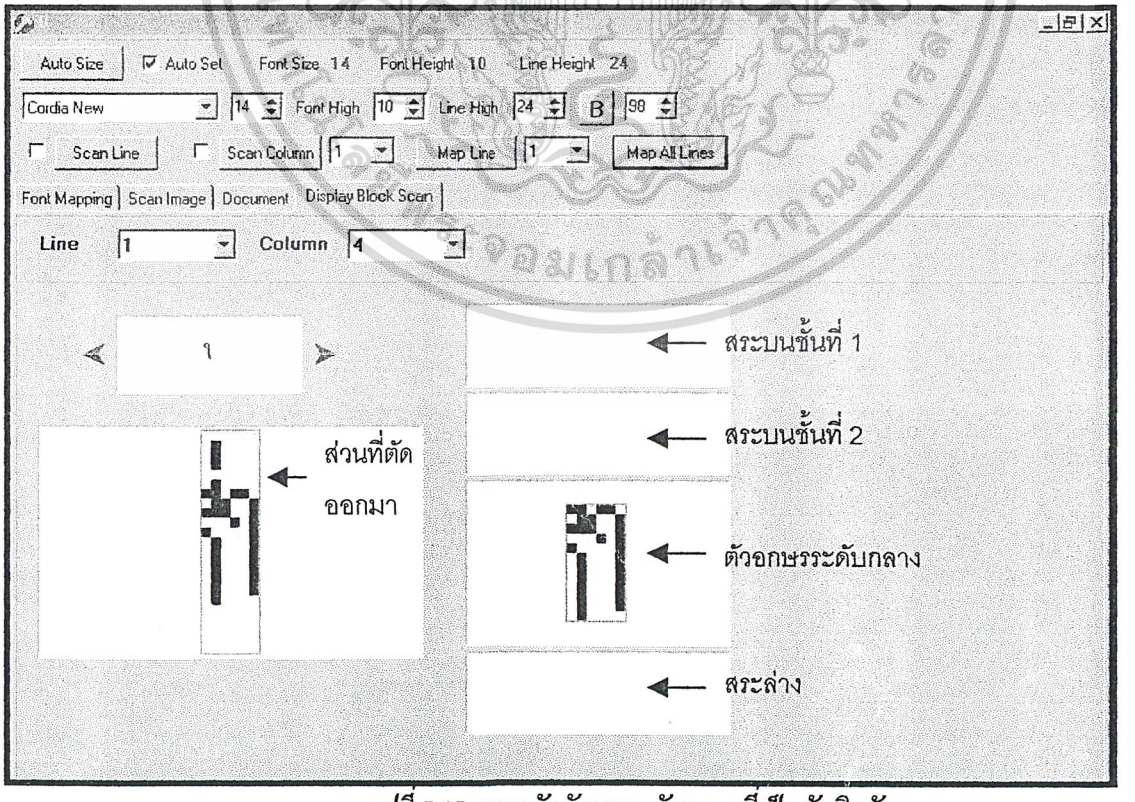
รูปที่ 5.12 แสดงตัวอักษรที่ไม่ได้ติดกันและมีสระบน 1 ชั้น



รูปที่ 5.13 แสดงตัวอักษรที่มีตัวไม่ติดกันและมีสระบนสองระดับอยู่



รูปที่ 5.14 แสดงตัวอักษรที่มีสระบนระดับเดียวและตัวติดกัน



รูปที่ 5.15 แสดงตัวอักษรระดับกลางที่เป็นตัวติดกัน

5.3 การทดสอบการทำงานของโปรแกรม

5.3.1 การทดสอบการตัดตัวอักษร

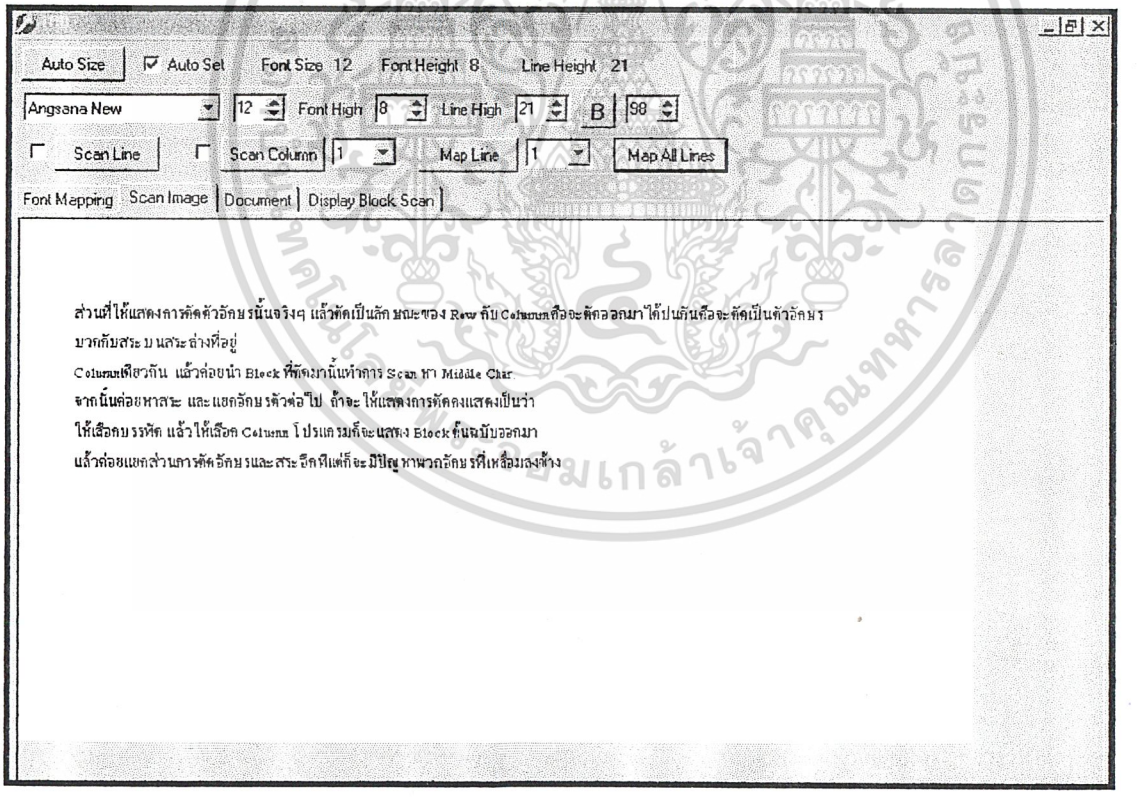
ในการทดสอบ จะทดสอบแยกตัวอักษรที่มีระดับแตกต่างกันไป ดังนี้

1. ระดับบนสุด จะแบ่งเป็นสระชั้นที่ 1 และชั้นที่ 2 ซึ่งประกอบด้วย วรรณยุกต์ และตัวการันต์
2. ระดับกลางประกอบด้วย พยัญชนะ และสระระดับกลาง
3. ระดับล่างสุด ประกอบด้วย สระระดับล่าง

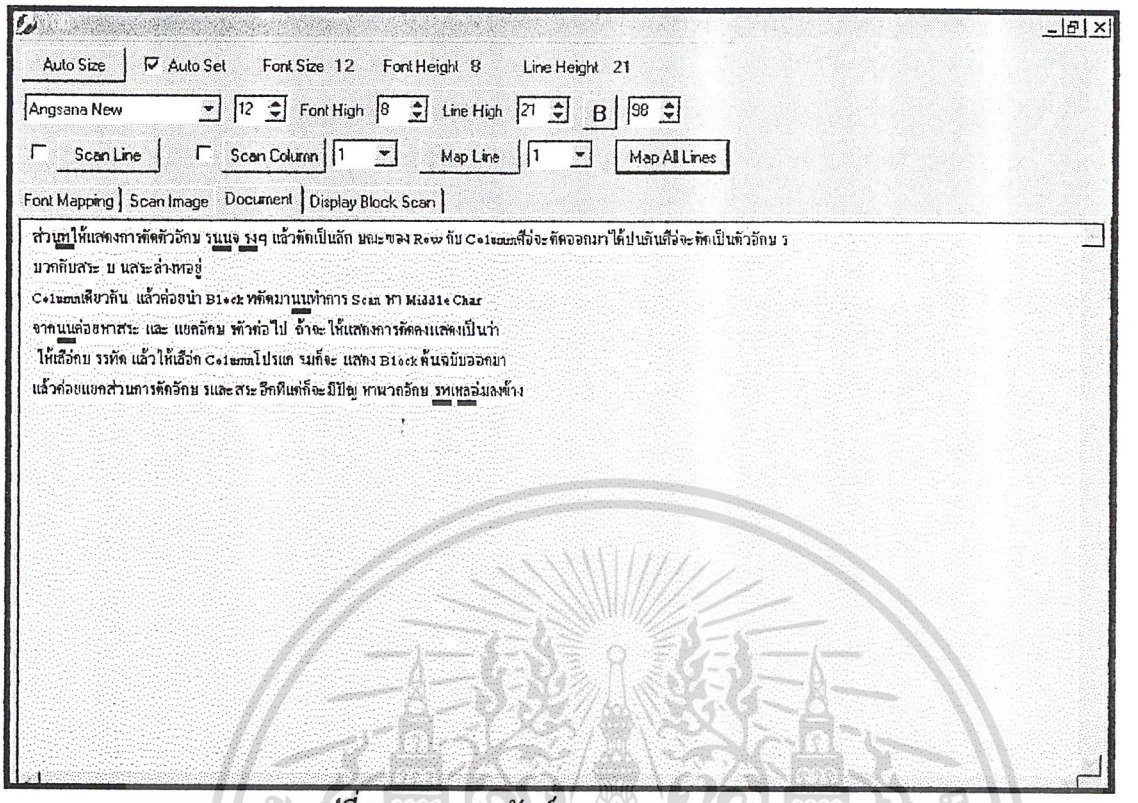
ผลการทดสอบ โดยทำการทดสอบกับเอกสาร 4 หน้า ตัวอักษรทั้งหมดประมาณ 6500 ตัว สามารถแยกตัวอักษรได้ถูกต้องทั้งหมด 5850 ตัว คิดเป็นเปอร์เซ็นต์ประมาณ 90 เปอร์เซ็นต์ในกรณีนี้รวมทั้งตัวอักษรที่ติดกันด้วย เพราะในโปรแกรมจะสามารถใช้วิธีการเปรียบเทียบทีละตัวแล้วตัดตัวอักษรที่ติดกันออกมาได้

5.3.2 การทดสอบรู้จำตัวอักษร

ในการทดสอบจะทำการเปรียบเทียบภาพตัวอักษรในหน้าเอกสารกับตัวอักษรต้นแบบ (Pattern) โดยจะทำการทดสอบตัวอักษรประเภทต่างๆ ซึ่งมีผลการทดสอบดังต่อไปนี้

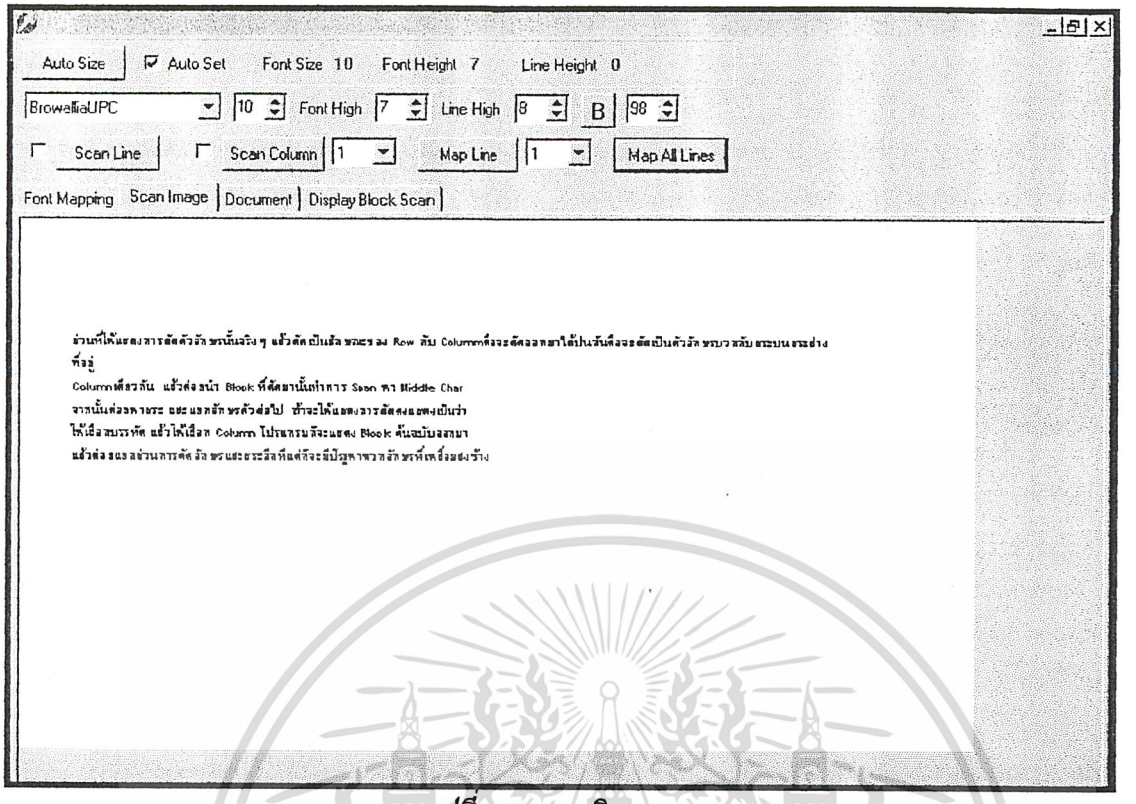


รูปที่ 5.16 แสดงอินพุท Angsana New ขนาด 12

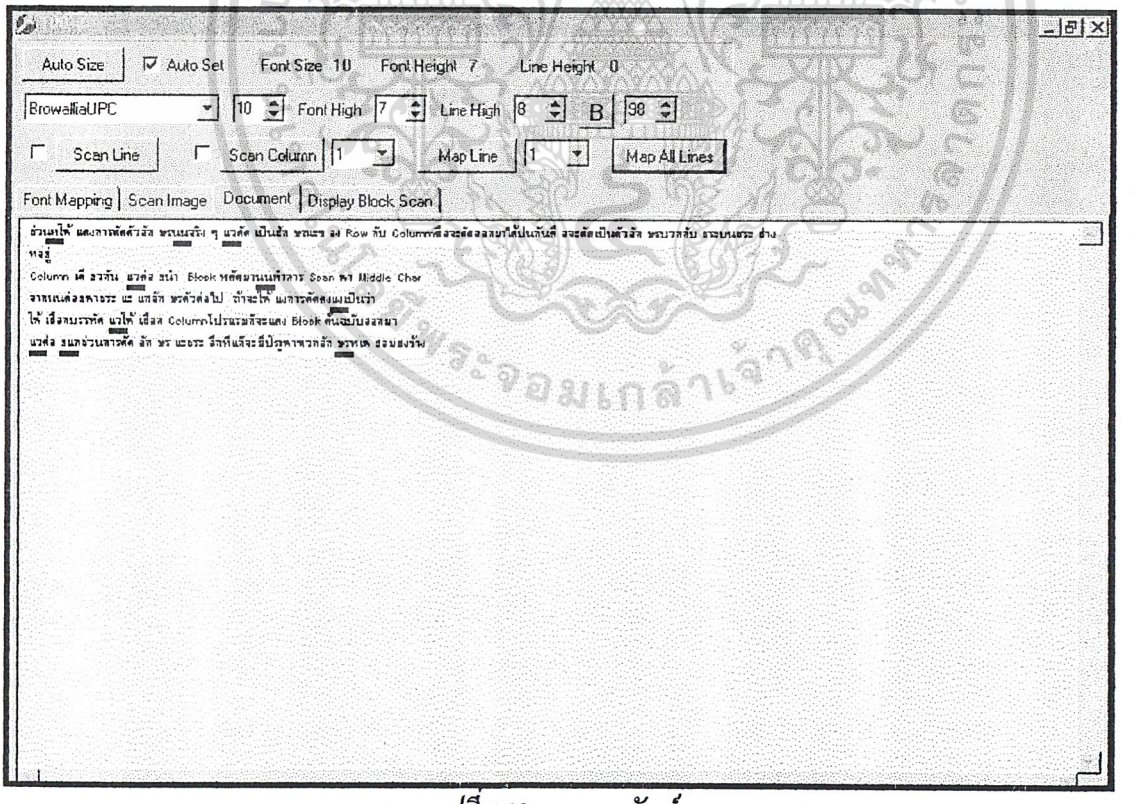


รูปที่ 5.17 แสดงผลลัพธ์ของ Angsana New ขนาด 12

จากรูปที่ 5.16 และ 5.17 อินพุท Angsana New ขนาด 12 มีตัวอักษรอินพุททั้งหมด 512 ตัว คิดเป็นเปอร์เซ็นต์ความถูกต้องประมาณ 99%

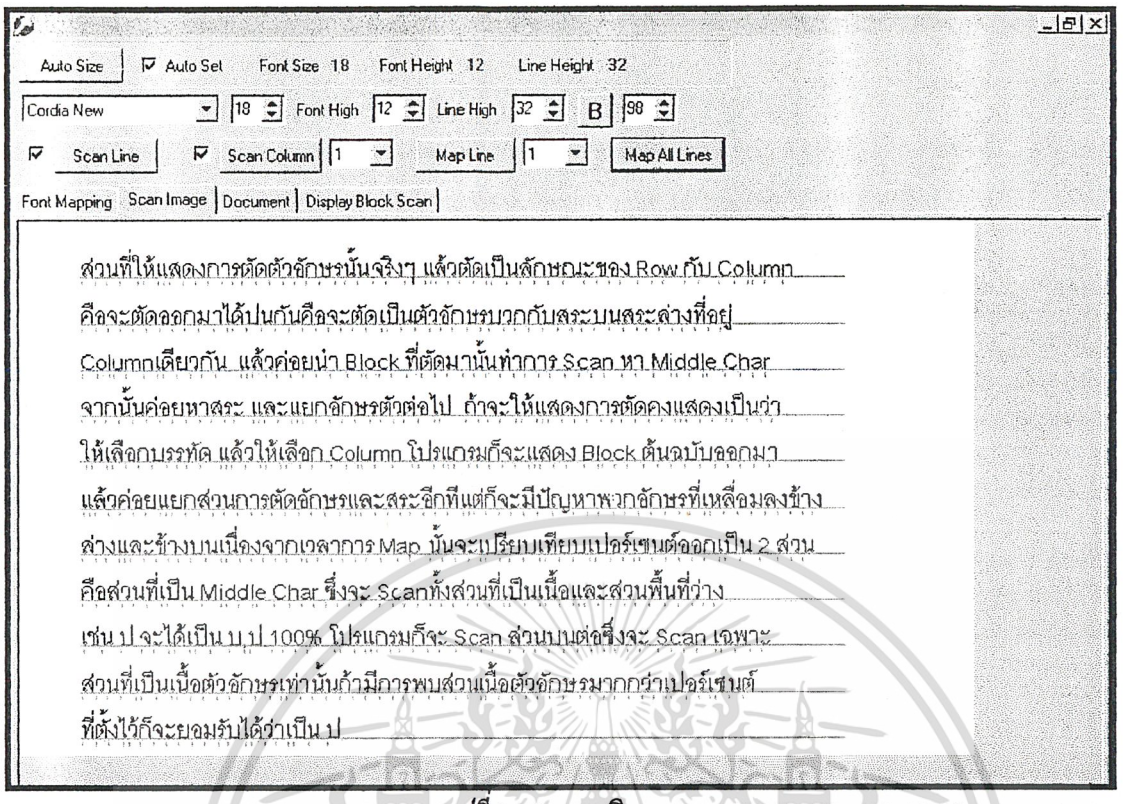


รูปที่ 5.18 แสดงอินพุทของ BrowalliaUPC ขนาด 10

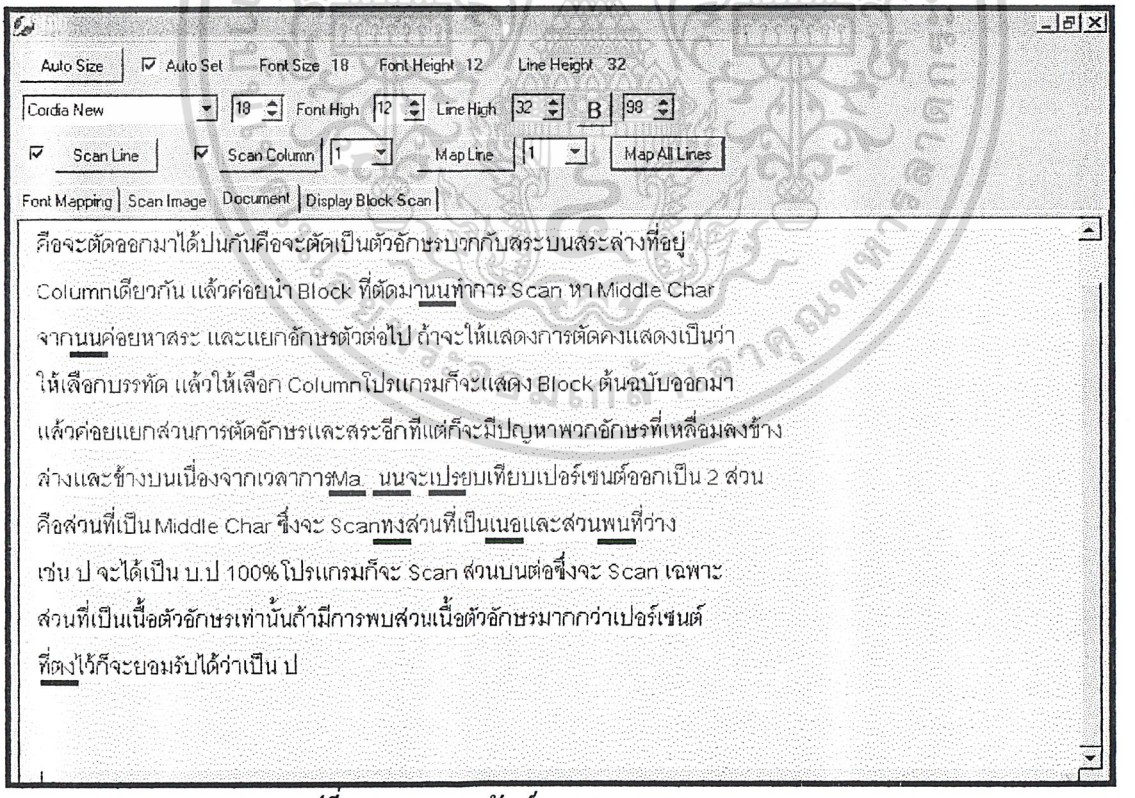


รูปที่ 5.19 แสดงผลลัพธ์ของ BrowalliaUPC ขนาด 10

จากรูปที่ 5.18 และ 5.19 อินพุท Browallia ขนาด 10 มีตัวอักษรอินพุททั้งหมด 512 ตัว คิดเป็นเปอร์เซ็นต์ความถูกต้องประมาณ 98%

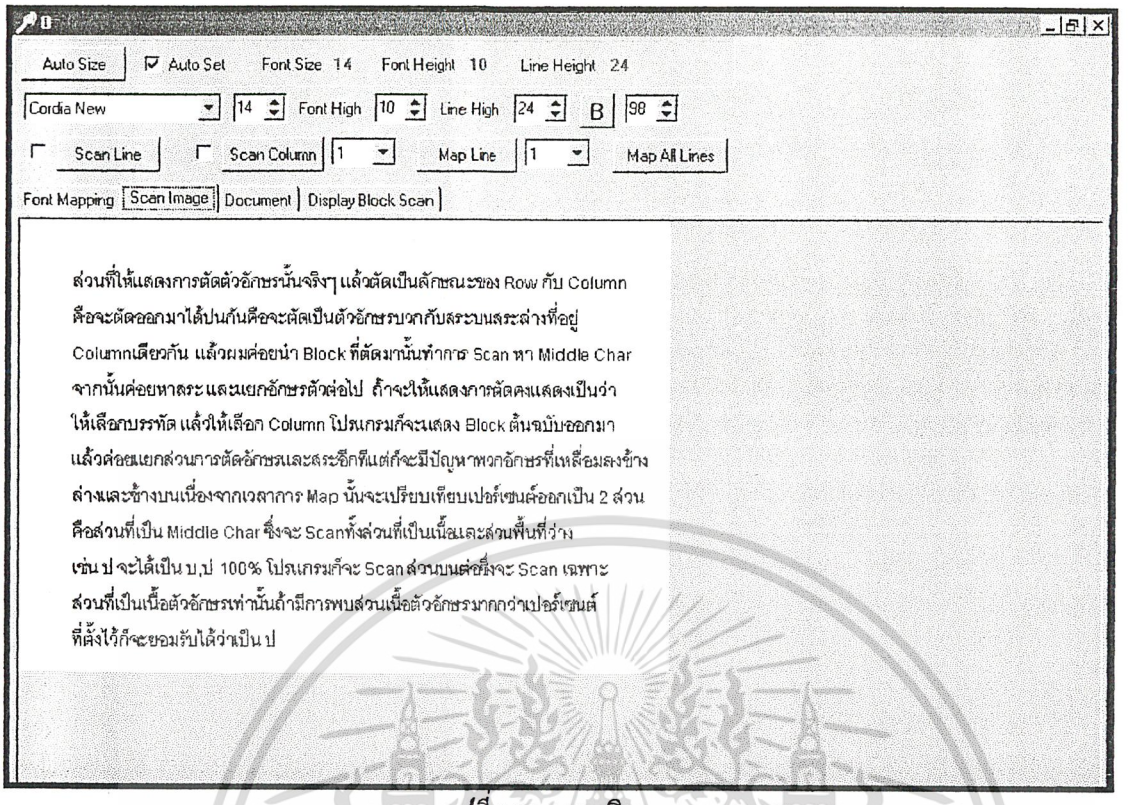


รูปที่ 5.20 แสดงอินพุทของ Cordia New ขนาด 18

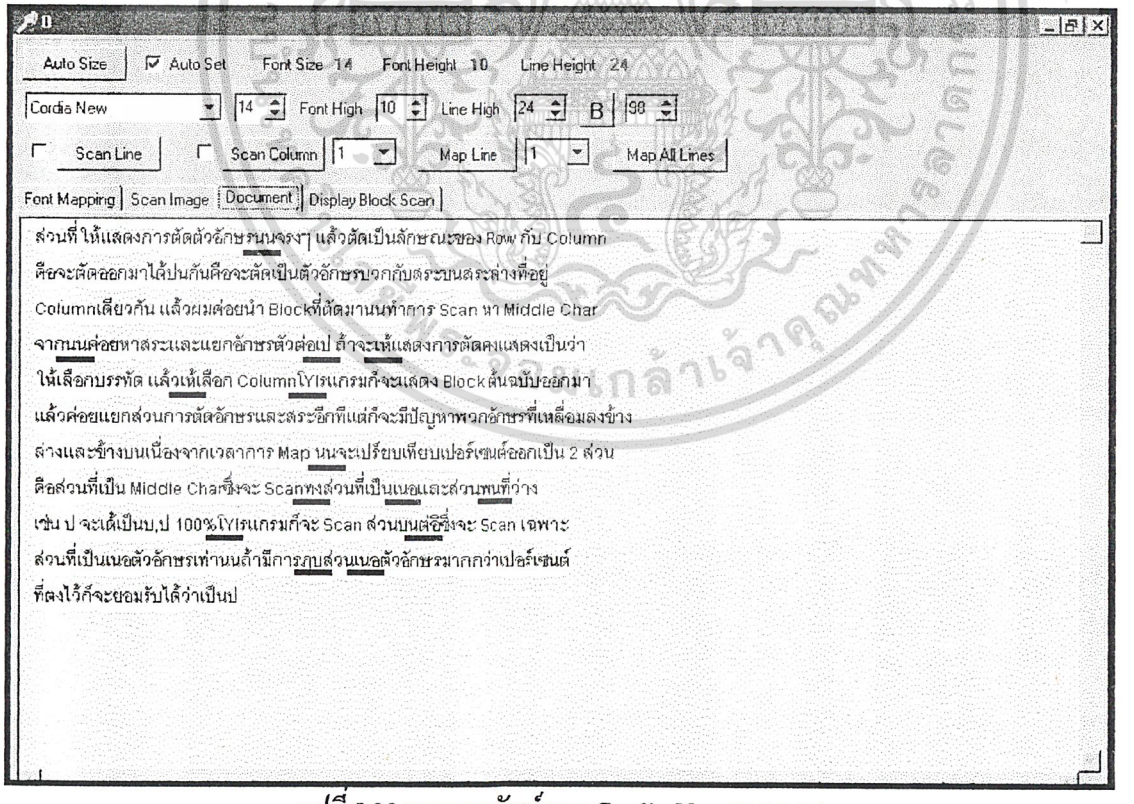


รูปที่ 5.21 แสดงผลลัพธ์ของ Cordia New ขนาด 18

จากรูปที่ 5.20 และ 5.21 อินพุท Cordia New ขนาด 18 มีตัวอักษรอินพุททั้งหมด 512 ตัว คิดเป็นเปอร์เซ็นต์ความถูกต้องประมาณ 98%



รูปที่ 5.22 แสดงอินพุทของ Cordia New ขนาด 14



รูปที่ 5.23 แสดงผลลัพธ์ของ Cordia New ขนาด 14

จากรูปที่ 5.22 และ 5.23 อินพุท Cordia New ขนาด 14 มีตัวอักษรอินพุททั้งหมด 512 ตัว คิดเป็นเปอร์เซ็นต์ความถูกต้องประมาณ 96%

บทที่ 6
บทสรุป

6.1 สรุปผลการทำงาน

จากการทำงานทำให้สามารถแยกตัวอักษรออกจากประโยคได้ และนำมาผ่านกระบวการหา Base line และคอลัมน์ (Column) ได้โดยใช้หลักการคำนวณจากความสูงของภาพตัวอักษร และหาจุดอ้างอิงเพื่อนำภาพตัวอักษรไปเปรียบเทียบกับตัวอักษรต้นแบบ (Pattern)

สำหรับลักษณะของโครงงานก็ได้สรุปมาเป็นข้อๆดังต่อไปนี้

1. ในกรณีที่ตัวอักษรไม่ได้ติดกันหรือตัดกัน สามารถตัดตัวอักษรออกมาเป็นตัวๆได้ โดยใช้กระบวนการในการหา Base line และคอลัมน์ (Column)
2. สำหรับตัวอักษรที่อยู่ติดกันหรือมีส่วนที่ตัดกันไม่สามารถแยกออกจากกันได้ในทุกๆกรณี ใช้กระบวนการในการไล่เปรียบเทียบตัวอักษรตัวแรกแล้วเมื่อทำการเปรียบเทียบเสร็จแล้ว จะตัดส่วนนี้ออก แล้วทำการตรวจสอบรูปตัวอักษรตัวถัดไป
3. ภาพของตัวอักษรแต่ละตัวที่ถูกตัดออกจากประโยค จะถูกพิจารณาเปรียบเทียบระหว่างรูปภาพตัวอักษรกับตัวอักษรต้นแบบ (Pattern) โดยจะหาจุดเริ่มต้น(Origin) เพื่อใช้เป็นจุดอ้างอิง ในการเปรียบเทียบ

6.2 ปัญหาและข้อจำกัด

1. เครื่องสแกนเนอร์ควรจะมีความละเอียดสูง โดยสามารถสแกนได้ที่ความละเอียดตั้งแต่ประมาณ 3048*3048 ขึ้นไป สำหรับสแกนเนอร์ที่ใช้ในการทำโครงงานครั้งนี้เป็นเครื่องสแกนเนอร์ที่สามารถทำการสแกนได้ครั้งละ 1 หน้ากระดาษ และยังมีการปรับความคมชัดและความสว่างได้สะดวก ทำให้เราสามารถเลือกปรับความคมชัดได้ตามความต้องการ
2. ลักษณะของแฟ้มข้อมูลที่ได้จากเครื่องสแกนเนอร์จะเป็นแฟ้มข้อมูลในรูปแบบ PSD จะต้องแปลงแฟ้มข้อมูลลักษณะดังกล่าวให้เป็นข้อมูลในรูปแบบที่สามารถนำไปใช้งานกับโปรแกรมได้เสียก่อน คือ แปลงให้เป็นแฟ้มข้อมูลแบบ BMP โหมดการทำงาน RGB
3. ภาพประโยคที่นำมาใช้ในการทดสอบจะต้องมีความคมชัดของตัวอักษรสูง เช่น พิมพ์ด้วยเครื่องเลเซอร์พรินเตอร์ สแกนด้วยสแกนเนอร์ที่ปรับค่าความละเอียดสูงเพื่อให้ภาพตัวอักษรมีความคมชัด เป็นต้น และกระดาษที่ใช้ควรจะมีความละเอียดสูง ไม่หยาบ ทั้งนี้เพราะป้องกันมิให้เกิดจุดรบกวนขึ้นเวลาที่นำภาพประโยคไปทำการสแกน
4. ตัวอักษรในหน้าเอกสารจะต้องเป็นตัวอักษร ขนาดและชนิดเดียวกันตลอดทั้งหน้าเอกสาร และขณะสแกนหน้าเอกสารจะต้องไม่เอียง

6.3 ปัญหาในการพัฒนาโครงงาน

ในระหว่างการพัฒนาโครงงาน เกิดปัญหาต่างๆ มากมาย พอสรุปเป็นปัญหาหลักๆได้ ดังนี้

1. เนื่องจากทฤษฎีที่ใช้ในการทำโครงงานในตอนแรก มีความยาก ทำให้เข้าใจได้ยาก
2. การค้นคว้า หรือหาจุดอ้างอิง เพื่อนำมาใช้ในการพัฒนาโปรแกรมค่อนข้างหายาก จำเป็นต้องคิดพัฒนาโปรแกรมขึ้นมาเอง

3. จะต้องทำการศึกษาการเขียนโปรแกรมให้สามารถทำงานได้เร็ว เพราะมีข้อมูลที่ต้องทำการวิเคราะห์อยู่เป็นจำนวนมาก

4. ปัญหาที่เกิดจากการผิดพลาดในการเขียนโปรแกรม ทำให้ต้องเสียเวลาตรวจสอบบ่อยครั้ง

6.4 แนวทางการแก้ปัญหา

1.ทำการวิเคราะห์ความต้องการของระบบโดยการช่วยกันคิดเอง ว่าระบบที่ได้น่าจะเป็นอย่างไร และมีการทำงานอย่างไรบ้าง

2.ทำการศึกษาจากตัวอย่าง หรือหาเอกสารที่ง่ายต่อความเข้าใจมาช่วย

3.ต้องมีการตรวจสอบความผิดพลาดอยู่เสมอเพื่อให้ไม่เกิดความซับซ้อนเมื่อตรวจพบข้อผิดพลาด จะทำให้การแก้ไขทำได้ง่ายขึ้น

4.เพิ่มขั้นตอนการทำ pre-process ให้มากขึ้นเพื่อลดข้อจำกัดที่ต้องใช้สแกนเนอร์ความละเอียดสูง และสามารถทำงานได้แม้อินพุทจะมีสัญญาณรบกวน

5.เพิ่มขั้นตอนการปรับตำแหน่งหน้าเอกสาร (Orientation) เข้าไปเพื่อขจัดข้อจำกัดที่หน้าเอกสารจะต้องตั้งตรงเสมอขณะสแกน

6.5 บทวิจารณ์ผล

สำหรับการพัฒนาโครงงานนี้ ซึ่งได้กล่าวในวัตถุประสงค์ว่า เพื่อสร้างระบบที่อำนวยความสะดวกและให้ประโยชน์ในส่วนของอินพุตทำให้ไม่จำเป็นต้องเสียเวลาในการพิมพ์งาน ซึ่งโปรแกรมประยุกต์ดังกล่าวจะประสบผลสำเร็จและให้ประโยชน์สูงสุดได้ จำเป็นต้องขึ้นอยู่กับปัจจัยในหลายๆด้าน ไม่ว่าจะเป็น ประสิทธิภาพของโปรแกรมประยุกต์เอง เทคโนโลยีที่นำมาสนับสนุนซึ่งจะมีทฤษฎีใหม่ๆเสมอ และงานวิจัยที่พัฒนาเรื่องการรู้จำตัวอักษร เป็นต้น ซึ่งถ้าปัจจัยต่างๆ เหล่านี้สนับสนุนต่อระบบงานในแง่บวก ทางผู้พัฒนาก็คาดว่า ระบบนี้ต้องได้รับการสนับสนุนให้มีประสิทธิภาพมากขึ้นเป็นแน่แท้

6.6 ข้อเสนอแนะ

การพิจารณาตัวอักษรโดยให้โปรแกรมทำการคำนวณและปรับเปลี่ยนค่าเอง โปรแกรมจะสามารถคำนวณหาค่าต่างๆและปรับเปลี่ยนให้เหมาะสมกับตัวอักษรแต่ละแบบและชนิดได้เอง จะทำให้โปรแกรมมีความยืดหยุ่นมากยิ่งขึ้น

6.7 แนวทางการพัฒนาในอนาคต

การพัฒนาโปรแกรมในอนาคต ควรจะทำให้โปรแกรมสามารถเรียนรู้ และตัดสินใจได้เองว่ารูปตัวอักษรนั้นควรจะเป็นตัวอักษรอะไร โดยมีหลักการดังต่อไปนี้

แนวคิดการรู้จำตัวอักษร

พิจารณาพยัญชนะและสระไทย จะพบว่ามีความต่อเนื่อง ประกอบด้วย เส้นตรง เส้นโค้ง วงกลม ที่เป็นหัวและวงกลมภายใน ดังนั้นลายเส้นที่ได้จากการเขียนหรือการพิมพ์จึงมีโอกาสเปลี่ยนแปลงมากได้ แต่ลักษณะเฉพาะตัวและลายเส้นของพยัญชนะและสระใดๆ ต่างก็มีคุณสมบัติที่แตกต่างกันแต่ละตัว จากเหตุผลดังกล่าวสามารถอาศัยคุณสมบัติดังกล่าวนี้รู้จำพยัญชนะหรือสระได้ โดยจะแทนคุณสมบัติต่างๆด้วยรหัส และจะมีชุดรหัสตัวอย่างเปรียบเสมือนเป็นกรอบ เพื่อใช้เปรียบเทียบกับชุดรหัสจากอินพุท

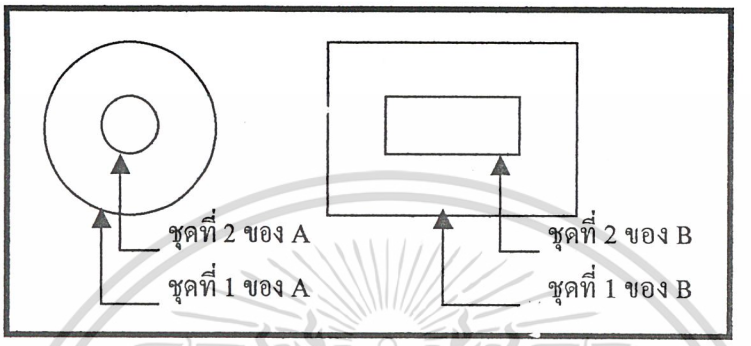
ชุดที่ 1 เป็นชุดรหัสที่ครอบคลุมตัวพิมพ์ทั้งหมด ที่อาจเกิดขึ้นได้

ชุดที่ 2 เป็นชุดรหัสบังคับที่พยัญชนะหรือสระต้องมี

การตัดสินใจเลือกว่าเป็นพยัญชนะหรือสระใดนั้น จะเกิดขึ้นเมื่อรหัสของอินพุทอยู่ระหว่างชุดรหัสชุดที่ 1 กับชุดรหัสชุดที่ 2

เช่น ให้ อักษร A มีชุดรหัสเป็นวงกลม,อักษรตัว B มีชุดรหัสเป็นสี่เหลี่ยมผืนผ้า

ชุดรหัสตัวอย่างของ A และ B จะแสดงได้ดังรูป



รูปที่ 6.1 แสดงรหัส 2 ชุด

การจดจำตัวอักษร

รูปแบบของรหัส

รูปที่ 6.2 แสดงรหัส

โดยที่รหัสตัวแรก รหัสภายใน รหัสตัวสุดท้าย มีดังตารางที่ 1 ซึ่งแต่ละตัวจะมีความหมายดังตารางที่ 2

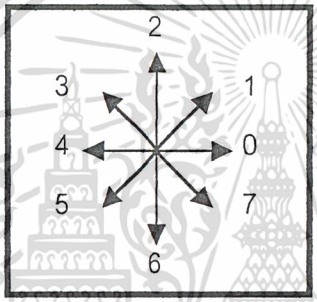
รหัสตัวแรก	รหัสภายใน	รหัสตัวสุดท้าย
Q1		
Q2		P (point – กรณีไม่มีหัว)
Q3		CC (หัวตามเข็ม)
Q4		CW (หัวทวนเข็ม)
NQ		

ตารางที่ 1 แสดงรหัสที่แทนรหัสตัวแรก รหัสภายใน และรหัสตัวสุดท้าย

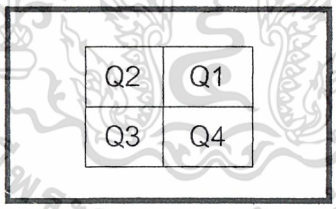
ลักษณะ	รหัส
ตำแหน่งจุดปลาย	Q1,Q2,Q3,Q4 และ NQ (ไม่มีจุดปลาย)
ส่วนของเส้นตรงและเส้นโค้ง	0,1,2,…,6 และ 7
ส่วนทีเป็นเส้นแยก	S(separate line)
ส่วนที่เป็นวงกลมภายใน	C(circle)
ส่วนเริ่มต้นที่เป็นปลาย	P(point)
ส่วนเริ่มต้นที่เป็นหัว	CC(หัวทวนเข็มนาฬิกา)
	CW(หัวตามเข็มนาฬิกา)

ตารางที่ 2 แสดงความหมายของอักษร

ซึ่งรหัสที่แทนส่วนของเส้นตรงและเส้นโค้ง เป็นรหัสที่แทนทิศทางของลายเส้นด้วยมีทิศทางดังรูป



รูปที่ 6.3 แสดงทิศทางรหัสของ 0,1,2,…,6 และ 7



รูปที่ 6.4 แสดงรหัสที่แทนแต่ละควอแดรนท์

จากปริญญานิพนธ์นี้อาจจะมีบางส่วนที่เกี่ยวข้องกับการรู้จำตัวอักษรภาษาไทยยังขาดหายไปอยู่ จึงหวังว่าผู้วิจัยท่านอื่นที่มีความสนใจในเรื่องนี้ จะได้แนวทางในการพัฒนาให้มีความสมบูรณ์มากยิ่งขึ้น ซึ่งจะทำให้สามารถนำไปใช้งานได้อย่างมีประสิทธิภาพมากยิ่งขึ้น

ภาคผนวก ก
ทฤษฎีฮิสโตแกรม

กระบวนการจำกัดขอบเขตบนหน้าเอกสารเพื่อทำการแยกตัวอักษรบนหน้าเอกสารอีกวิธีหนึ่งคือการใช้ฮิสโตแกรมในการหาขอบเขตของกลุ่มตัวอักษร โดยใช้คุณสมบัติของช่องว่างระหว่างบรรทัดและช่องว่างระหว่างตัวอักษร มาใช้ในการวิเคราะห์โดยมีหลักการดังต่อไปนี้

1. หาฮิสโตแกรมของภาพในแนวนอน
2. ในระหว่างที่ทำการหาฮิสโตแกรมในแนวนอน ถ้ามีค่าฮิสโตแกรมที่จุดใดที่มีการเปลี่ยนจากค่า 0 เป็นค่า 1 ให้สันนิษฐานว่าจุดนั้นเป็นจุดเริ่มต้นของเส้นบรรทัดบน
3. ถ้ามีค่าฮิสโตแกรมที่จุดใดที่มีค่าการเปลี่ยนจากค่า 1 ไปเป็นค่า 0 ก็ให้สันนิษฐานว่าจุดนั้นเป็นจุดของเส้นบรรทัดบรรทัดล่าง
4. ถ้าเส้นระหว่างบรรทัดมีค่าที่ห่างพอ ก็จะทำการตัดบรรทัดนั้นออกไปเก็บไว้อีกที่หนึ่ง
5. ทำการแยกส่วนต่างๆออกมาทีละบรรทัด จนหมดหน้าเอกสาร

รูปที่ 1 แสดงการตัดบรรทัด

6. นำแถวที่ได้เป็นบรรทัด แล้วนำมาทำการตัดตัวอักษรออกทีละตัว(Character segmentation) โดยอาศัยฟังก์ชันฮิสโตแกรมของเซลในแนวตั้ง ในกรณีของตัวอักษรที่มีการกำหนดขนาดคงที่ปัญหาที่เกิดจะไม่ยุ่งยากมากนัก แต่ในกรณีการพิมพ์ตัวอักษรที่มีการปรับช่องว่างแบบไม่เท่ากันจะเกิดปัญหาเพิ่มมากยิ่งขึ้น

รูปที่ 2 แสดงการตัดตัวอักษรออกจากบรรทัด

แต่เนื่องจากการหาฮิสโตแกรมจะทำให้ได้แนวของบรรทัดหรือตัวอักษรแบบคร่าวๆเท่านั้น แต่ไม่สามารถหาจุดเริ่มต้น (Origin) เพื่อเป็นจุดอ้างอิงในการเปรียบเทียบตัวอักษรต่อไปดังนั้นใช้ฮิสโตรแกรมมาเพื่อหาค่าคงที่ต่างๆ ในสมการต่อไป เพื่อคำนวณหาค่ากั้นซ้าย กั้นบน และคำนวณหาเส้นบรรทัด และเส้นบรรทัดล่าง(Base line) โดยนำค่าฮิสโตรแกรมต่างๆมาทำเป็นกราฟเพื่อให้เข้าใจมากขึ้น

ภาคผนวก ข

โปรแกรมและเครื่องมือต่างๆ ของ Delphi4

Delphi คืออะไร

Delphi เป็นเครื่องมือสำหรับการพัฒนาแอปพลิเคชั่นบน Windows ภาษาพื้นฐานที่ใช้ คือภาษา Pascal (Object Pascal) ในการเขียนโปรแกรม นอกจากนี้ Delphi ยังสนับสนุนการพัฒนาโปรแกรมแบบ Visual Programming ที่ผู้พัฒนาสามารถเลือกคอมโพเนนต์ที่ต้องการมาวางบนฟอร์ม แล้วกำหนดคุณสมบัติบางอย่างของคอมโพเนนต์เหล่านั้น รวมทั้งอาจมีการเขียนโปรแกรมควบคุมการทำงานของคอมโพเนนต์เหล่านั้นด้วย

ความต้องการของระบบสำหรับ Delphi 4

สำหรับเครื่องที่จะใช้งาน Delphi 4 ได้ จำเป็นจะต้องมีส่วนประกอบดังนี้เป็นอย่างน้อย

- Windows 95/98 หรือ Windows NT
- CPU 80486 หรือ Pentium หรือสูงกว่า
- หน่วยความจำ 8 MB(16 MB สำหรับ Windows 95/98 และ 32 MB สำหรับWindows NT)
- จอภาพ VGA หรือจอภาพที่มีรายละเอียดสูงกว่า
- เครื่องอ่าน CD-ROM
- Mouse
- เนื้อที่บนฮาร์ดดิสก์ที่ต้องการขึ้นอยู่กับรูปแบบการติดตั้ง และเวอร์ชั่นของ Delphi

โปรแกรมและเครื่องมือต่างๆของ Delphi 4

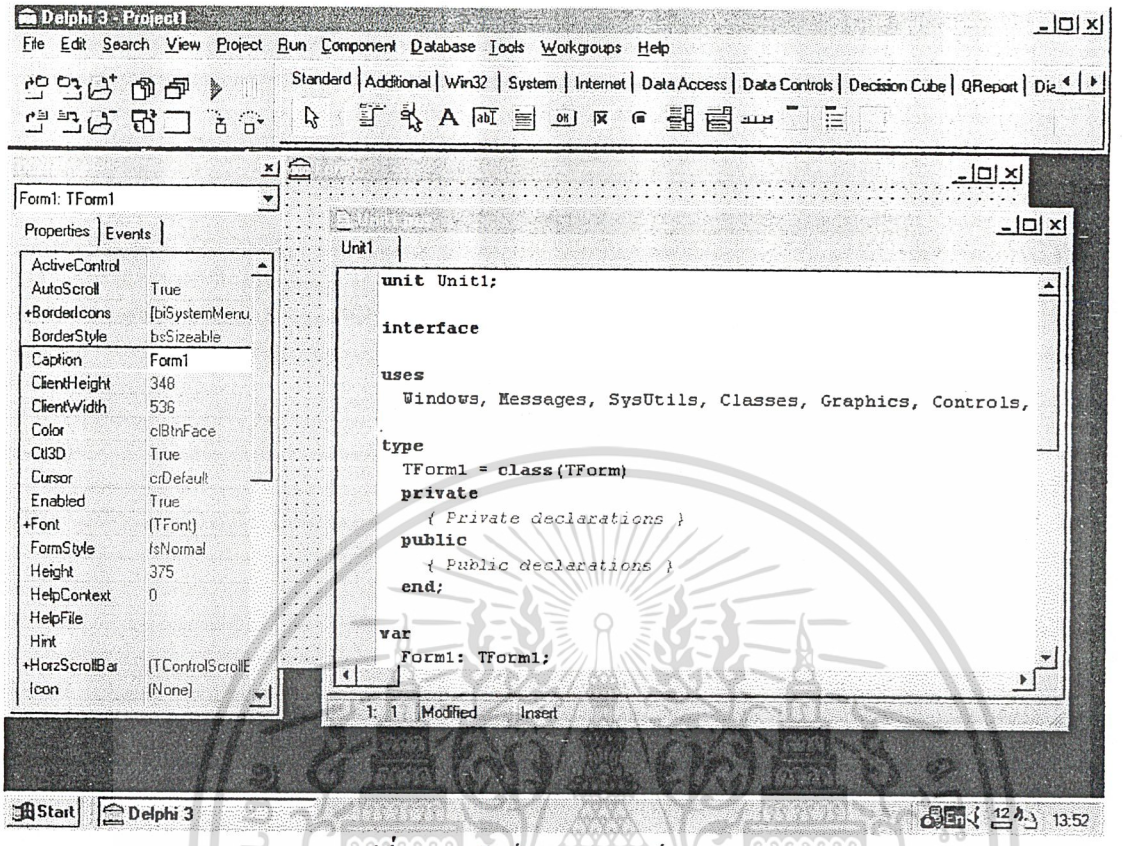
วินโดว์หลัก(MainWindow)

ประกอบไปด้วย

ไตเติ้ลบาร์(Title bar)

เมนูหลัก(Main Menu Bar)

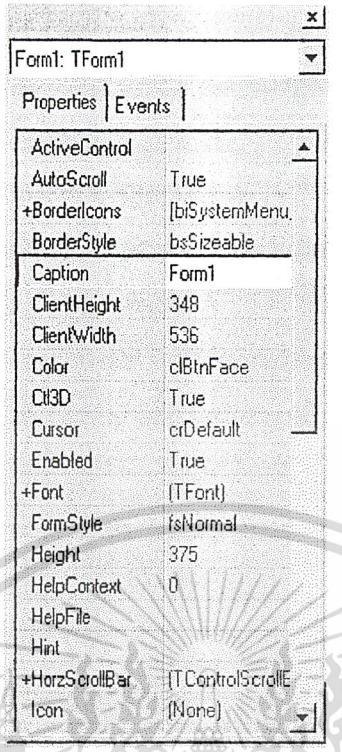
สปีดบาร์(Speed bar)



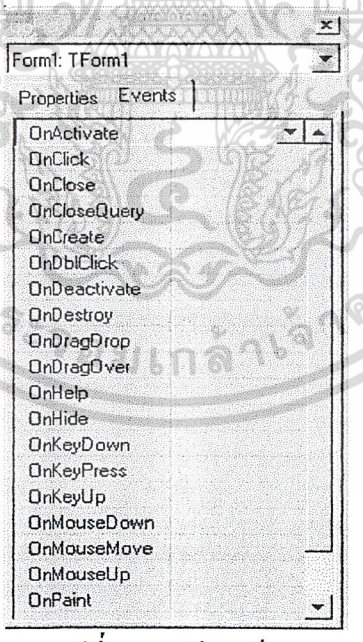
รูปที่ 1 ออบเจ็กต์อินสเปกเตอร์(Object Inspector)

ออบเจ็กต์อินสเปกเตอร์ถูกใช้กำหนดคุณสมบัติ(Properties) และเหตุการณ์ต่างๆ(Events) ที่เกิดขึ้นกับแต่ละคอมโพเนนต์ในแอพพลิเคชันของเรา

ในออบเจ็กต์อินสเปกเตอร์แบ่งออกเป็น 2 เพจ คือ พรอมเพอร์ตี้เพจ และอีเวนต์เพจ ซึ่งสามารถคลิกที่แท็บเพจเพื่อสลับไปมาระหว่าง 2 เพจนี้ได้ ดังรูปที่ 2 และ 3



รูปที่ 2 แสดงพรอพเพอตี้

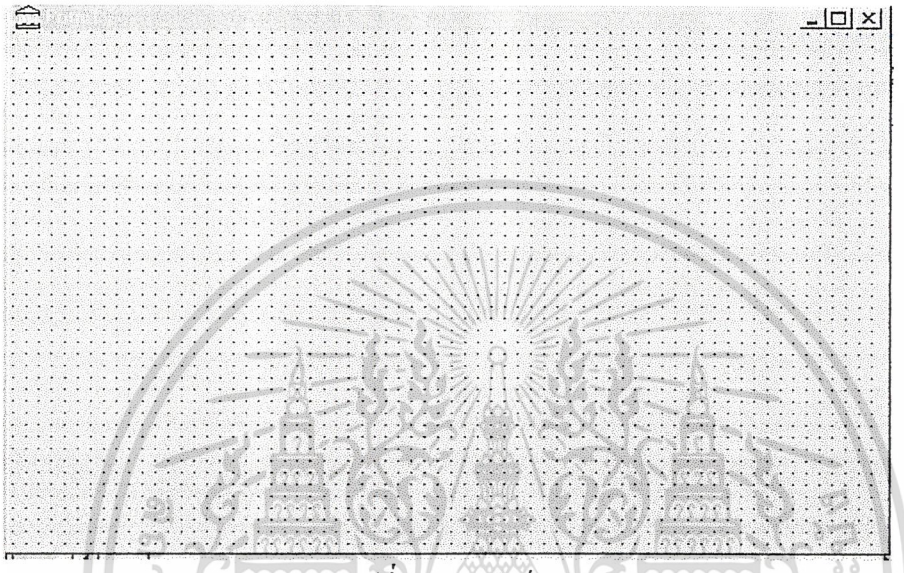


รูปที่ 3 แสดงอีเวนต์

ฟอร์ม(Form)

ฟอร์มเป็นพื้นที่สำหรับสร้างเป็นส่วนที่ใช้ติดต่อกับผู้ใช้(user interface)ของแอพพลิเคชัน ซึ่งจะมีการนำเอาคอมโพเนนต์ต่างๆมาจัดวางไว้ภายใน

ฟอร์มถือเป็นคอมโพเนนต์ตัวหนึ่งเช่นกัน ซึ่งมีชื่อว่า TForm โดยคอมโพเนนต์ Tform นี้เป็นที่รวมของคอมโพเนนต์อื่นๆหรือเรียกว่าพาเรนต์ (parent) ของคอมโพเนนต์ที่อยู่บนฟอร์ม โดยที่โปรเจ็กต์หนึ่งๆสามารถมีฟอร์มได้หลายฟอร์ม



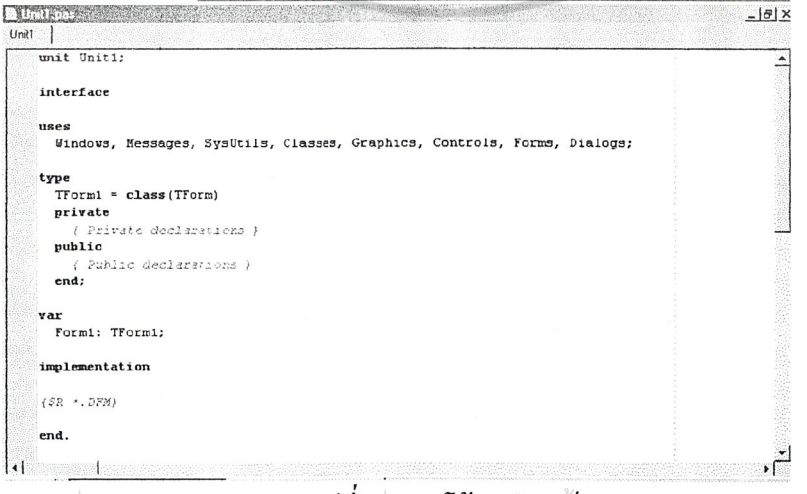
รูปที่ 4 แสดงฟอร์ม

โค้ดเอดิเตอร์(Code Editor)

สำหรับโค้ดเอดิเตอร์ประกอบด้วยส่วนสำคัญคือ

เอดิเตอร์(Editor)

เป็นส่วนที่ใช้สำหรับเขียนโค้ดของโปรแกรม เมื่อเริ่มโปรเจ็กต์ใหม่ทุกครั้ง เดลไฟจะสร้างโค้ดของยูนิต(ไพล์.pas) ขึ้นมาในโค้ดเอดิเตอร์ เมื่อมีการเพิ่มฟอร์มใหม่ขึ้นออกมาในโปรเจ็กต์ ก็จะมีการเพิ่มหน้าใหม่ของโค้ดซึ่งเป็นยูนิตใหม่อีกยูนิตหนึ่งด้วย เราสามารถสลับไปมาระหว่างแต่ละยูนิตได้โดยใช้แท็บ ดังรูปที่ 5



รูปที่ 5 แสดงโค้ดเอดิเตอร์

โปรเจ็กต์เมเนเจอร์(Project Manager)

โปรเจ็กต์เมเนเจอร์ประกอบด้วย รายชื่อของไฟล์ยูนิตและฟอร์มที่ประกอบขึ้นเป็นแอพพลิเคชัน นอกจากนี้ยังแสดงชื่อไดเร็กทอรี(Directory) ที่เก็บไฟล์เหล่านั้นด้วย

ออบเจ็กต์บราวเซอร์(Object Browser)

ออบเจ็กต์บราวเซอร์จะให้ข้อมูลเกี่ยวกับความสัมพันธ์ของออบเจ็กต์,ยูนิต, และ Global Symbols ที่โปรแกรมของเราใช้อยู่ เราเรียกใช้ออบเจ็กต์บราวเซอร์ได้ 2 วิธี คือ

- เลือกเมนูคำสั่ง View -> Browser
- คลิกขวาในหน้าต่างโค้ดเอดิเตอร์ แล้วเลือกคำสั่ง Browse Symbol at Cursor

อย่างไรก็ตามเราต้องทำการคอมไพล์โปรเจ็กต์เสียก่อนถึงจะใช้บราวเซอร์นี้ได้ถ้ามีการปิดโปรเจ็คต์แล้วเปิดขึ้นมาใหม่ก็จะต้องทำการคอมไพล์อีกครั้งหนึ่งจึงจะใช้บราวเซอร์นี้ได้เช่นกัน

เขียนโปรแกรมแบบ object oriented โดยใช้เดลไฟ

การเขียนโปรแกรมแบบ Object Oriented(OOP)

การเขียนโปรแกรมด้วยเดลไฟเป็นการเขียนโปรแกรมแบบอ๊อบเจ็กต์โอเรียนเต็ด(Object Oriented Programming : OOP) โดยที่เดลไฟจะมองทุกอย่างในโปรแกรมเป็นออบเจ็กต์ซึ่งมีลักษณะเฉพาะ และมีความสามารถภายในตัวออบเจ็กต์นั้น

ในเดลไฟนั้น คอมโพเนนต์ก็ถือว่าเป็นออบเจ็กต์ โดยในคอมโพเนนต์จะมีลักษณะเฉพาะของคอมโพเนนต์ นั้นก็คือ พรอพเพอร์ตี้(Property) และมีความสามารถของคอมโพเนนต์ ซึ่งก็คือ เมธอด (Method) ซึ่งทั้งสองสิ่งเป็นช่องทางที่เราจะติดต่อหรือเรียกใช้คอมโพเนนต์นั้นได้

การเขียนโปรแกรมแบบออบเจ็กต์มีข้อดีที่สำคัญมากคือสามารถนำกลับมาใช้ได้อีก(Re-use) ทำให้ลดเวลาในการสร้างแอพพลิเคชันต่างๆลงเป็นอย่างมาก

รู้จักกับClass

Class คือการรวมเอาข้อมูลที่ใช้ลักษณะของออบเจ็กต์ และความสามารถของออบเจ็กต์ที่จะมีอยู่ ถ้าจะพูดให้เข้าใจง่ายกว่านั้น Class คือ ต้นแบบของออบเจ็กต์นั้นเอง

Class หนึ่งๆสามารถสร้างออบเจ็กต์ให้ได้หลายๆตัว โดยที่แต่ละตัวมีโครงสร้างข้อมูลภายในเหมือนกัน

รู้จักกับ Data field

ใน class จะมีข้อมูลที่แสดงลักษณะเฉพาะของ class ซึ่งเราเรียกว่า Data Field และแต่ละ field ก็สามารถกำหนดระดับการเข้าถึง (Level Access) ได้

รู้จักกับ Method

นอกจากเราจะใช้ Data Field ในการบ่งบอกความแตกต่างของแต่ละ Class แล้ว เราสามารถบอกความแตกต่างของแต่ละ Class ได้ด้วยความสามารถ หรือที่เรียกว่า Method ของ Class

รู้จักกับ Constructor และ Destructor

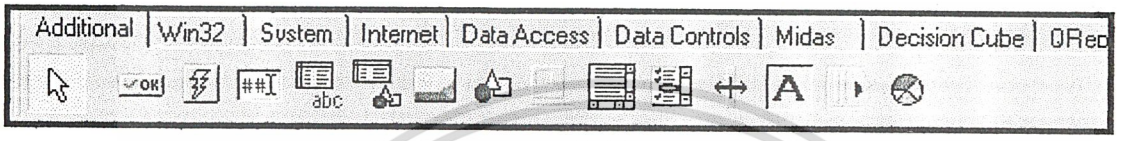
Constructor เป็นเมธอดพิเศษ ซึ่งจะถูกเรียกใช้ทุกครั้งที่มีการสร้างอินสแตนซ์ของ Class ขึ้นมา โดยเมธอดนี้จะทำการจองทรัพยากรสำหรับอินสแตนซ์ และกำหนดค่าเริ่มต้น

สำหรับ Destructor นั้นจะทำหน้าที่ตรงกันข้ามกับ Constructor คือคืนทรัพยากรที่อินสแตนซ์ของ Class ใช้ไปเมื่อต้องการเลิกใช้งานอินสแตนซ์ของ Class นั้นๆ

การใช้งานคอมโพเนนต์ชั้นสูง

ไดอะล็อกบ๊อกซ์ในเดลไฟ

เดลไฟได้เตรียมคอมโพเนนต์สำหรับสร้างไดอะล็อกบ็อกซ์พื้นฐานที่ใช้งานในแอพพลิเคชัน เช่น ไดอะล็อกบ๊อกซ์ในการเปิดไฟล์ เป็นต้น ซึ่งคอมโพเนนต์เหล่านี้จะอยู่ในเพจ Dialogs ของคอมโพเนนต์พาเลต์ ดังรูปที่ 6



รูปที่ 6 กวมโพเนนต์เกี่ยวกับ image

การใช้งานเดลไฟในด้านกราฟฟิก

เราสามารถนำรูปภาพกราฟฟิกมาใช้ในแอพพลิเคชันของเราได้หลายวิธี ตั้งแต่การโหลดรูปภาพที่มีอยู่แล้วมาไว้ในแอพพลิเคชันในขณะออกแบบ การสร้างรูปแบบขึ้นมาใหม่โดยใช้ graphical control ในขณะออกแบบ หรือแม้กระทั่งการวาดภาพขึ้นมาในขณะโปรแกรมกำลังรันอยู่

คอมโพเนนต์ Image

ใช้สำหรับแสดงรูปภาพลงบนฟอร์ม หรือแม้กระทั่งโหลดรูปภาพมาเป็น background ของฟอร์ม โดยเราสามารถเลือกรูปภาพจากอ๊อบเจ็กต์อินสเปกเตอร์ในขณะออกแบบ หรือโหลดภาพขึ้นแสดงในขณะรันไทม์ได้

คลาส TCanvas

เราใช้ TCanvas เป็นพื้นที่ในการวาดภาพลงบนออบเจ็กต์ใดๆ โดย Tcanvasมีพรอพเพอร์ตี้,อีเวนต์และเมธอดที่ช่วยเราในการสร้างสรรค์รูปภาพสวยๆ โดยการ

- สามารถกำหนดสีของพื้นหลัง รูปแบบและสีของลายเส้น และกำหนด font ที่จะใช้ได้
- กำหนดรูปแบบและสีของภาพที่จะให้ปรากฏ
- เขียนตัวอักษรต่างๆลงบนภาพ
- โหลดรูปภาพกราฟฟิกลงบนออบเจ็กต์
- เปลี่ยนรูปภาพในขณะรันไทม์ได้

พรอพเพอร์ตี้สำคัญของ TCanvas

Brush	ใช้สำหรับกำหนดรูปแบบและสีของภาพหรือรูปทรงต่างๆ
CilpRect	กำหนดขอบเขตสำหรับวาดภาพบน canvas เช่นในอีเวนต์ OnPaint เราสามารถกำหนดขอบเขตที่ให้สามารถวาดรูปลงบนฟอร์มได้

CopyMode	กำหนดว่าจะให้รูปภาพที่ copy มาจาก canvas ปัจจุบันนั้นปรากฏในรูปแบบใด
Font	กำหนดรูปแบบตัวอักษรที่จะใช้เขียนบน canvas
Handle	ใช้เมื่อมีการเรียกใช้ Window API โดยตรง
Pen	กำหนดรูปแบบและสีของลายเส้นบน canvas
PenPos	พิกัด x,y ของพรอพเพอร์ตี้ Pen
Pixels	กำหนดสีของจุดแต่ละจุดบน ClipRect

เมธอดสำคัญของ Tcanvas

Arc	วาดโค้งลงบน canvas โดยใช้ pen
BruchCopy	แสดงรูปภาพโดยมีพื้นหลังแบบโปร่งใส
CopyRect	copy รูปแบบหรือบางส่วนของรูปแบบลงบน canvas
Draw	copy รูปภาพจากหน่วยความจำลงบน canvas
Ellipse	วาดรูปวงรีลงบน canvas โดยใช้ pen และสีของวงรีนี้จะเป็นสีเดียวกับที่ตั้งไว้ในพรอพเตอร์ตี้ brush
FloodFill	ใส่สีของ brush ลงบน canvas
LineTo	วาดเส้นจากตำแหน่งปัจจจุบันไปยังตำแหน่งที่กำหนดในพารามิเตอร์ x และ y
MoveTo	กำหนดตำแหน่งปัจจุบันที่จะใช้วาดภาพ
Pie	วาดรูป Pie ลงบน canvas
Polygon	วาดรูปหลายเหลี่ยมลงบน canvas โดยใช้อาร์เรย์ของจุด และสีของรูปหลายเหลี่ยมนี้ จะเป็นสีเดียวกับสีที่ตั้งไว้ในพรอพเพอร์ตี้ Brush
Polyline	ใช้พรอพเพอร์ตี้ pen วาดเส้นหลายๆเส้นลงบน canvas โดยใช้อาร์เรย์ของจุด
Rectangle	วาดรูปสี่เหลี่ยมลงบน canvas โดยเส้นขอบและสีของเหลี่ยมนี้จะเป็นไปตามพรอมเพอร์ตี้ pen และ brush ตามลำดับ
RoundRect	วาดสี่เหลี่ยมที่มีแต่ละมุมเป็นส่วนโค้ง
StretchDraw	copy รูปภาพจากหน่วยความจำลงบน canvas
TextHeight	แสดงความสูงของตัวอักษรในพารามิเตอร์
TextOut	เขียนตัวอักษรบนพื้นที่ของ canvas
TextRect	เขียนตัวอักษรบนพื้นที่ที่กำหนดโดยพรอพเพอร์ตี้ ClipRect

การสร้างแอพพลิเคชันด้านกราฟฟิกที่เขียนขึ้นจึงมักจะใช้พรอพเพอร์ตี้และเมธอดนี้เป็นส่วนใหญ่ในการเขียนแอพพลิเคชันเกี่ยวกับกราฟิก

บรรณานุกรม

J.R. Parker , **" ALGORITHMS FOR IMAGE PROCESSING AND COMPUTER VISION"** , WILEY COMPUTER PUBLISHING, 1997.

ชาญชัย ดีอ่วม, **" การรู้จำตัวอักษรคัดลายมือภาษาไทยของคอมพิวเตอร์โดยวิเคราะห์โครงสร้างแบบทัน ทีทันใด"**,วิทยานิพนธ์คณะวิทยาการคอมพิวเตอร์และเทคโนโลยีสาร สนเทศ,2542.

กฤษฎา ลิมปานันท์ และ ดร.โกสินทร์ จำนงไทย, **"วิธีการรู้จำลายมือเขียนภาษาไทยโดยใช้กรอบมาตราฐาน"**,การประชุมวิชาการวิศวกรรมไฟฟ้า ครั้งที่ 20 ,2540

Daniel B Hentchel, **"Creation of a Neural Network to Assist in Deciphering Degraded Anceint Hebraw Texts"** , http://www.cis.rit.edu/~dbh6913/Thesis/contents.html

Rafael C. Gonzalez & Richard E. Woods, "Digital Image Processing",Addison Wesley,1993.

ชม กิ้มปาน และ สมศักดิ์ วลัยรัชต์, **" การรู้จำตัวอักษรลายมือเขียนภาษาไทยของไมโครคอมพิวเตอร์ โดยการพิจารณาลักษณะลายเส้นขณะที่ลากไป"**, การประชุมวิชาการทางวิศวกรรมไฟฟ้าครั้งที่ 13 ,คณะวิศวกรรมศาตร์ มหาวิทยาลัยเชียงใหม่, 8-9 พฤศจิกายน,2533

มนัส สังวรศิลป์ , พิชัย ดูศิริวานิชกร , สุรพันธ์ เอื้อไพบูลย์ และ เชวง ตันติยาภรณ์ , " การจดจำตัวอักษรลายเขียนภาษาไทยโดยการพิจารณาหัวของตัวอักษร" , การประชุมวิชาการวิศวกรรมไฟฟ้า ครั้งที่ 11 , 2531

Rafael C. Gonzalez and Richard E. Woods, **"Digital image processing"**, Addison-Wesly Publishing Company , 1992.

ภาคผนวก ค

โปรแกรมการทำงาน

```
unit ScanTextU7;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, ComCtrls, Spin, ExtDlgs, Buttons;

type
  // กำหนดโครงสร้างข้อมูลที่เป็น Margin
  TMargin = record
    Left : longint;
    Top : longint;
    Right : longint;
    Bottom : longint;
  end;

  // กำหนดขนาดของ Pattern
  TPattern = array[0..39,0..47] of longint;

  // กำหนดขนาดของ Buffer ของ Block ตัวอักษรที่จะทำการ Map
  TLoadData = array[0..80,0..60] of longint;

  // กำหนดโครงสร้างของ Pattern
  TPatternData = record
    PMargin : TMargin;
    PData : TPattern;
    PRight : longint;
  end;
```

```
TForm1 = class(TForm)
  Panel1: TPanel;
  FontName: TComboBox;
  FontSizeSp: TSpinEdit;
  MapLineBtn: TButton;
  ScanLineBtn: TButton;
  PageControl1: TPageControl;
  TabSheet1: TTabSheet;
  ScrollBox2: TScrollBox;
  TabSheet2: TTabSheet;
  ScrollBox1: TScrollBox;
  ScanImage: TImage;
  TabSheet3: TTabSheet;
  ResultText: TMemo;
  OpenPictureDialog1: TOpenPictureDialog;
  LineHighSp: TSpinEdit;
  Label1: TLabel;
  Label2: TLabel;
  FontHighSp: TSpinEdit;
  Panel2: TPanel;
  DisplayFont: TMemo;
  LoadPattern: TButton;
  PttCharC: TComboBox;
  Panel3: TPanel;
  PttImage: TImage;
  ScanColumnBtn: TButton;
  ColumnScanLine: TComboBox;
  ScanTextLine: TComboBox;
  MapAllLineBtn: TButton;
  FontBold: TSpeedButton;
  ListBox1: TListBox;
  DisplayRow: TCheckBox;
  DisplayCol: TCheckBox;
  Recognize: TSpinEdit;
```

TabSheet4: TTabSheet;

Panel4: TPanel;

Panel5: TPanel;

Panel6: TPanel;

ImageBlock: TImage;

Panel7: TPanel;

ImageUpper1: TImage;

Panel8: TPanel;

ImageUpper2: TImage;

Panel9: TPanel;

ImageMidle: TImage;

Panel10: TPanel;

ImageLower: TImage;

Label3: TLabel;

DisplayTextLine: TComboBox;

Label4: TLabel;

DisplayTextCol: TComboBox;

TextBlock: TPanel;

PrevColBtn: TSpeedButton;

NextColBtn: TSpeedButton;

AutoSizeBtn: TButton;

HistoryHit: TListBox;

HistDifVal: TListBox;

HistFontHeight: TListBox;

FontHeightL: TLabel;

Label5: TLabel;

FontSizeL: TLabel;

Label7: TLabel;

LineHeightL: TLabel;

Label8: TLabel;

HistLineHeight: TListBox;

Bevel1: TBevel;

AutoSet: TCheckBox;

procedure SetDisplayFont;

```
    procedure FontNameChange(Sender: TObject);
    procedure FontSizeSpChange(Sender: TObject);
    procedure LoadPatternClick(Sender: TObject);
    procedure PttCharCChange(Sender: TObject);
    procedure FontHighSpChange(Sender: TObject);
    procedure LineHighSpChange(Sender: TObject);
    procedure FontBoldClick(Sender: TObject);
    procedure RecognizeChange(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure ScanImageDblClick(Sender: TObject);
    procedure ScanLineBtnClick(Sender: TObject);
    procedure ScanColumnBtnClick(Sender: TObject);
    procedure MapLineBtnClick(Sender: TObject);
    procedure MapAllLineBtnClick(Sender: TObject);
    procedure DisplayTextLineChange(Sender: TObject);
    procedure DisplayTextColChange(Sender: TObject);
    procedure PrevColBtnClick(Sender: TObject);
    procedure NextColBtnClick(Sender: TObject);
    procedure AutoSizeBtnClick(Sender: TObject);
  private
    { Private declarations }
    procedure GetFontNames;
    procedure ScanColumn(LineNo : longint);
    procedure FindPttBaseLine;
    procedure LoadPttData(AsciiNo : longint);
    procedure LoadScanData(LineNo,SCol,ECol,SRow,ERow : longint);
    function  AsciiMapRight(LineNo,SCol,ECol,SRow,ERow : longint): string;
    function  ScanText(LineNo, StrCol, EndCol : longint): string;
    procedure DisplayBlockScan(BlockCol : longint);
  public
    // กำหนดค่าให้กับ Pattern
    PttFontName : string;
    PttFontSize : longint;
    PttFontColor : TColor;
```

```
PttBgColor : TColor;

Margin : TMargin;
FontHigh : longint;
LineHigh : longint;
LineCount : longint;
ColCount : longint;
ShiftPrev : longint;
EndLineCol : longint;
PRecognize : longint;

// เก็บค่าต่างๆ ไว้ใช้ในการตัดตัวอักษร
MemShiftPrev : longint;
MemMidleTop : longint;
MemReg : boolean;
MemLower : TRect;
MemUpper1 : TRect;
MemUpper2 : TRect;

// เก็บแถว
BaseLineArray : array[0..50] of longint;  // Line number of A4 must less then 50 line
// เก็บ Column ใน 1 แถว
ColLineArray : array[0..200] of longint;

// เก็บค่าของ Pattern
PttBaseL : longint;
PttChar : TPatternData;
PttArray : array[32..255] of TPatternData;

ScanData : TLoadData;
BlankData : TLoadData;

ScanTextRun : boolean;
```

```
// Exp:
//   0
//   1      Midle Char
//   :
//   45  Baseline
//   46      Lower Char
//   :
//   55  DownLine
//   56      Upper Char
//   :
//   80  UpperChar


   { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

// กำหนด Margin
function SetMargin(MLeft,MTop,MRight,MBottom : longint): TMargin;
begin
  with Result do
  begin
    Left := MLeft;
    Top := MTop;
    Right := MRight;
    Bottom := MBottom;
  end;
end;
```

```
// Add ชื่อ Font ลงใน Combobox
function EnumFontsProc(var LogFont: TLogFont; var TextMetric: TTextMetric;
 FontType: longint; Data: Pointer): longint; stdcall;
begin
 TStrings(Data).Add(LogFont.lfFaceName);
 Result := 1;
end;

// โหลด Font ของ Windows
procedure TForm1.GetFontNames;
var DC: HDC;
begin
 DC := GetDC(0);
 EnumFonts(DC, nil, @EnumFontsProc, Pointer(FontName.Items));
 ReleaseDC(0, DC);
 FontName.Sorted := True;
end;

// กำหนดค่าให้กับระบบเมื่อผู้ใช้เลือก Font
procedure TForm1.SetDisplayFont;
begin
 FontName.ItemIndex := FontName.Items.IndexOf(PttFontName);
 FontSizeSp.Text := IntToStr(PttFontSize);
 DisplayFont.Font.Name := FontName.Items[FontName.ItemIndex];
 DisplayFont.Font.Size := PttFontSize;

 if FontBold.Down then
  DisplayFont.Font.Style := [fsBold]
 else
  DisplayFont.Font.Style := [];

 PttImage.Canvas.Font.Name := DisplayFont.Font.Name;
 PttImage.Canvas.Font.Size := DisplayFont.Font.Size;
```

```
  PttImage.Canvas.Font.Charset := DisplayFont.Font.Charset;
  PttImage.Canvas.Font.Style := DisplayFont.Font.Style;

  ResultText.Font.Name := DisplayFont.Font.Name;
  ResultText.Font.Size := DisplayFont.Font.Size;
  ResultText.Font.Charset := DisplayFont.Font.Charset;
  ResultText.Font.Style := DisplayFont.Font.Style;

  TextBlock.Font.Name := DisplayFont.Font.Name;
  TextBlock.Font.Size := DisplayFont.Font.Size;
  TextBlock.Font.Charset := DisplayFont.Font.Charset;
  TextBlock.Font.Style := DisplayFont.Font.Style;
end;
// แสดงผลเมื่อมีการเปลี่ยน Font ใหม่
procedure TForm1.FontNameChange(Sender: TObject);
begin
  PttFontName := FontName.Items[FontName.ItemIndex];
  SetDisplayFont;
end;

procedure TForm1.FontSizeSpChange(Sender: TObject);
begin
  PttFontSize := StrToInt(FontSizeSp.Text);
  FontHighSp.Value := Round(PttFontSize*0.65);
  //  High of middle charecter estimate 65% of Font Size
  // ความสูงของตัวอักษรตรงกลาง เช่น 'A' จำมีค่าประมาณ 65% ของขนาด Font
  FontHigh := FontHighSp.Value;

  LineHighSp.Value := Round(PttFontSize*1.2);
  // Distance betaween line must more than 20% of Font Size
  // ความสูงของบรรทัดมีค่าไม่ต่ำกว่า 20% ของขนาด Font
  LineHigh := LineHighSp.Value;
  SetDisplayFont;
end;
```

```
procedure TForm1.FontHighSpChange(Sender: TObject);
begin
  FontHigh := FontHighSp.Value;
end;

procedure TForm1.LineHighSpChange(Sender: TObject);
begin
  LineHigh := LineHighSp.Value;
end;

procedure TForm1.FontBoldClick(Sender: TObject);
begin
  FontBold.AllowAllUp := not(FontBold.AllowAllUp);
  FontBold.Down := not(FontBold.Down);

  SetDisplayFont;
end;

procedure TForm1.RecognizeChange(Sender: TObject);
begin
  PRecognize := Recognize.Value;
end;

procedure TForm1.FormCreate(Sender: TObject);
var i,j : longint;
    CharStr : string;

  procedure AddChar;
  begin
    if ((i >= 209) and (i <= 217)) or ((i >= 231) and (i <= 237)) then
      CharStr := CharStr + ' ' + Chr(i)
    else CharStr := CharStr + Chr(i);
```

```
    if i in [64,96,126,160,206,237] then
    begin
      DisplayFont.Lines.Add(CharStr);
      CharStr := '';
    end;
  end;

begin
  // แสดงหมายเลขตัวอักษรลงใน Memo
  CharStr := '';
  for i := 32 to 126 do AddChar;

  for i := 161 to 217 do AddChar;

  for i := 223 to 237 do AddChar;

  for i := 240 to 249 do AddChar;

  DisplayFont.Lines.Add(CharStr);
  GetFontNames;
  PttFontName := DisplayFont.Font.Name;
  PttFontSize := -MulDiv(DisplayFont.Font.Height, 72, Screen.PixelsPerInch);
  PttFontColor := clBlack;
  PttBgColor := clWhite;
  RecognizeChange(nil);
  SetDisplayFont;

  // ลบค่าใน Block Pattern
  for i := 0 to 80 do
    for j := 0 to 60 do
      BlankData[i,j] := 0;
end;

// โหลดรูปเข้ามาใหม่
```

```
procedure TForm1.ScanImageDblClick(Sender: TObject);
begin
 if OpenPictureDialog1.Execute then
  if OpenPictureDialog1.FileName <> '' then
   ScanImage.Picture.LoadFromFile(OpenPictureDialog1.FileName);
end;

// โหลด Pattern ที่จะใช้เป็นต้นแบบในการ Map
procedure TForm1.LoadPatternClick(Sender: TObject);
var i : longint;
begin
 Screen.Cursor := crHourGlass;
 PttCharC.Clear;

 // โหลดค่า Ascii เก็บไว้เพื่ออ้างอิงในการโหลด Pattern
 for i := 32 to 126 do
  PttCharC.Items.Add(IntToStr(i));

 for i := 161 to 217 do
  PttCharC.Items.Add(IntToStr(i));

 for i := 223 to 237 do
  PttCharC.Items.Add(IntToStr(i));

 for i := 240 to 249 do
  PttCharC.Items.Add(IntToStr(i));

 // หาว่า Base Line ของ Pattern อยู่ตำแหน่งไหน
 FindPttBaseLine;

 // ทำการโหลด Pattern ทั้งหมดมาเก็บไว้ใน Array
 for i := 32 to 126 do
 begin
  LoadPttData(i);
```

```
    PttArray[i] := PttChar;
  end;

  // กำหนดให้ Ascii 32 (Spacebar) มีค่า Margin เท่ากับ Ascii 98 (b)
  PttArray[32].PMargin := PttArray[98].PMargin;  // 98=b

  // ทำการโหลด Pattern ที่เหลือทั้งหมดมาเก็บไว้ใน Array ต่อไป
  for i := 161 to 217 do
  begin
    LoadPttData(i);
    PttArray[i] := PttChar;
  end;

  for i := 223 to 237 do
  begin
    LoadPttData(i);
    PttArray[i] := PttChar;
  end;

  for i := 240 to 249 do
  begin
    LoadPttData(i);
    PttArray[i] := PttChar;
  end;
  Screen.Cursor := crDefault;
end;

// แสดงลักษณะของ Pattern เมื่อผู้ใช้เลือกค่า Ascii
procedure TForm1.PttCharCChange(Sender: TObject);
var Chr : byte;
begin
  if PttCharC.ItemIndex < 0 then Exit;
  Chr := StrToInt(PttCharC.Text);
  PttImage.Canvas.Lock;
```

```
  try
   with PttImage.Canvas do
   begin
    Brush.Color := clWhite;
    FillRect(PttImage.BoundsRect);
    Font.Color := clBlack;

    if ((Chr >= 209) and (Chr <= 217)) or ((Chr >= 231) and (Chr <= 237)) then
     TextOut(5,3,'  ' + Char(Chr) + ' ')
    else
     TextOut(5,3,Char(Chr));
   end;
  finally
   PttImage.Canvas.Unlock;
  end;
end;

// process ในการคำนวณหา Base Line ของ Pattern
// โดย scan จากด้านล่างขึ้นด้านบนจนพบเนื้อตัวอักษร
procedure TForm1.FindPttBaseLine;
var Row,Col : longint;
begin
 PttCharC.ItemIndex := PttCharC.Items.IndexOf(IntToStr(65));  // 65=A
 if PttCharC.ItemIndex > 0 then
 begin
  PttCharCChange(nil);
  for Row := PttImage.Height - 1 downto 0 do
   for Col := 0 to PttImage.Width - 1 do
    if PttImage.Canvas.Pixels[Col,Row] <> PttBgColor then
    begin
     PttBaseL := Row + 1;
     exit;
    end;
 end;
```

```
end;

// Process ในการโหลด Pattern ของตัวอักษร 1 ตัวมาเก็บไว้
// พร้อมทั้งคำนวณหา Margin ของตัวอักษรนั้นด้วย
procedure TForm1.LoadPttData(AsciiNo : longint);
var Row,Col : longint;
   FirstHit : boolean;
begin
 PttCharC.ItemIndex := PttCharC.Items.IndexOf(IntToStr(AsciiNo));
 if PttCharC.ItemIndex > -1 then
 begin
  PttCharCChange(nil);
  PttChar.PMargin := SetMargin(PttImage.Width - 1,PttImage.Height - 1,0,0);
  FirstHit := True;
  for Row := 0 to PttImage.Height - 1 do
   for Col := 0 to PttImage.Width - 1 do
    if PttImage.Canvas.Pixels[Col,Row] <> PttBgColor then
    begin
     PttChar.PData[Col,Row] := 1;

     if FirstHit then
     begin
      PttChar.PMargin.Top := Row;
      FirstHit := False;
     end;

     if Col < PttChar.PMargin.Left then PttChar.PMargin.Left := Col;
     if Col > PttChar.PMargin.Right then PttChar.PMargin.Right := Col;
     if Row > PttChar.PMargin.Bottom then PttChar.PMargin.Bottom := Row;
    end
    else PttChar.PData[Col,Row] := 0;
 end;
end;
```

```
// Process ในการ scan ข้อมูลจากรูปเข้ามาเก็บใน Block ScanData
procedure TForm1.LoadScanData(LineNo,SCol,ECol,SRow,ERow : longint);
var Row,Col : longint;
begin
 ScanData := BlankData;
 for Row := SRow to ERow do
  for Col := SCol to ECol do
   if ScanImage.Canvas.Pixels[Col,Row] = PttFontColor then
   begin
//    ScanImage.Canvas.Pixels[Col,Row] := clBlue; // Mark
    ScanData[Col - SCol, Row - SRow] := 1;
   end
   else
    ScanData[Col - SCol, Row - SRow] := 0;
end;

// process ในการ scan หา Base Line ทั้งหมดจากรูป
procedure TForm1.ScanLineBtnClick(Sender: TObject);
var Row,Col,ECol : longint;
  FirstHit : boolean;
  HitT : longint;
  LastHitT,LastBaseL,LastNoHit : longint;
  MLastHitT,MLastBaseL,MLastNoHit : longint;
  RowCheck,RowCheckC : longint;
  NextRowC1,NextRowC2 : longint;
begin
 Screen.Cursor := crHourGlass;
 ColumnScanLine.Clear;
 ScanTextLine.Clear;
 DisplayTextLine.Clear;
 DisplayTextCol.Clear;

 with ScanImage.Picture.Bitmap.Canvas do
 begin
```

```
FirstHit := True;
Margin := SetMargin(ScanImage.Width,ScanImage.Height,0,0);

LineCount := 0;
LastHitT := 0;
LastBaseL := 0;
LastNoHit := 0;
RowCheck := -1;

MLastHitT := -1;
MLastBaseL := -1;
MLastNoHit := -1;

// การกำหนดค่าในการกระโดดข้ามไป scan line ถัดไป
RowCheckC := Round(FontHigh*0.2); // Goto Midle Char
NextRowC1 := Round(FontHigh*0.6); // Goto Base Line
NextRowC2 := Round(FontHigh*0.8); // Goto NextLine

Row := 0;
ECol := ScanImage.Width - 1;
repeat
 HitT := 0;
 for Col := 0 to ECol do
  if Pixels[Col,Row] = PttFontColor then
  begin
   HitT := HitT + 1;

   // หา Margin ของรูป
   if FirstHit then
   begin
    Margin.Top := Row;
    FirstHit := False;
   end;
```

```
    if Col < Margin.Left then Margin.Left := Col;
    if Col > Margin.Right then Margin.Right := Col;
    if Row > Margin.Bottom then Margin.Bottom := Row;
   end;

 Application.ProcessMessages;

 if HitT <= LastHitT*0.2 then
 begin  // Base Line is near HitT decread near 0
      // HitT near 0 except some charecters [gypq estimate <= 20% of Base Line]

  if (Row - LastBaseL >= LineHigh) and
    (Row - LastNoHit  >= FontHigh) then
  begin
   MLastHitT := LastHitT;
   MLastBaseL := LastBaseL;
   MLastNoHit := LastNoHit;
   LastBaseL := Row;
   LastNoHit := Row;

   BaseLineArray[LineCount] := Row;
   LineCount := LineCount + 1;

   RowCheck := Row + RowCheckC;
   Row := RowCheck - 1;
  end;
 end;

 if (RowCheck = Row) then
 begin
  if (HitT >= MLastHitT*1.2) then
  begin  // if Upper char then HitT must more than mid char 20% of MLastHitT
      // And then reset data
```

```
    LastBaseL := MLastBaseL;
    LastNoHit := MLastNoHit;
    BaseLineArray[LineCount] := 0;
    LineCount := LineCount - 1;
    Row := Row + NextRowC1 - 1;
   end
   else Row := Row + NextRowC2 - 1;
  end;

  if HitT = 0 then LastNoHit := Row;

  LastHitT := HitT;
  Row := Row + 1;
 until Row > ScanImage.Height - 1;

 Margin.Right := Margin.Right + PttFontSize;
 Margin.Bottom := Margin.Bottom + PttFontSize;
 // Add pixel not scan

 for Row := 0 to LineCount - 1 do
  ColumnScanLine.Items.Add(IntToStr(Row + 1));

 ScanTextLine.Items.Assign(ColumnScanLine.Items);
 DisplayTextLine.Items.Assign(ColumnScanLine.Items);

 if ColumnScanLine.Items.Count > 0 then
 begin
  ColumnScanLine.ItemIndex := 0;
  ScanTextLine.ItemIndex := 0;
  DisplayTextLine.ItemIndex := 0;
  DisplayTextLineChange(nil);
 end;

 if DisplayRow.Checked then
```

```
    begin
      Pen.Color := clRed;  // Display Line
      for Row := 0 to LineCount - 1 do
      begin
        MoveTo(Margin.Left,BaseLineArray[Row]);
        LineTo(Margin.Right,BaseLineArray[Row]);
      end;
    end;
  end;
  Screen.Cursor := crDefault;
end;

// process ในการ scan column ทั้งหมดใน 1 line
procedure TForm1.ScanColumn(LineNo : longint);
var Row,Col : longint;
    SRow,ERow : longint;
    HitT : longint;
    LastHitT : longint;
begin
 with ScanImage.Picture.Bitmap.Canvas do
 begin
   ColCount := 0;
   LastHitT := 0;
   EndLineCol := 0;

   for Col := Margin.Left to Margin.Right do
   begin
     SRow := BaseLineArray[LineNo] - FontHigh + 1;   // 40% Font High
     ERow := BaseLineArray[LineNo] + Round(FontHigh*0.4); // ฐ
     HitT := 0;
     for Row := SRow to ERow do
       if Pixels[Col,Row] = PttFontColor then
         HitT := HitT + 1;
```

```
TForm1 = class(TForm)
  Panel1: TPanel;
  FontName: TComboBox;
  FontSizeSp: TSpinEdit;
  MapLineBtn: TButton;
  ScanLineBtn: TButton;
  PageControl1: TPageControl;
  TabSheet1: TTabSheet;
  ScrollBox2: TScrollBox;
  TabSheet2: TTabSheet;
  ScrollBox1: TScrollBox;
  ScanImage: TImage;
  TabSheet3: TTabSheet;
  ResultText: TMemo;
  OpenPictureDialog1: TOpenPictureDialog;
  LineHighSp: TSpinEdit;
  Label1: TLabel;
  Label2: TLabel;
  FontHighSp: TSpinEdit;
  Panel2: TPanel;
  DisplayFont: TMemo;
  LoadPattern: TButton;
  PttCharC: TComboBox;
  Panel3: TPanel;
  PttImage: TImage;
  ScanColumnBtn: TButton;
  ColumnScanLine: TComboBox;
  ScanTextLine: TComboBox;
  MapAllLineBtn: TButton;
  FontBold: TSpeedButton;
  ListBox1: TListBox;
  DisplayRow: TCheckBox;
  DisplayCol: TCheckBox;
  Recognize: TSpinEdit;
```

TabSheet4: TTabSheet;

Panel4: TPanel;

Panel5: TPanel;

Panel6: TPanel;

ImageBlock: TImage;

Panel7: TPanel;

ImageUpper1: TImage;

Panel8: TPanel;

ImageUpper2: TImage;

Panel9: TPanel;

ImageMidle: TImage;

Panel10: TPanel;

ImageLower: TImage;

Label3: TLabel;

DisplayTextLine: TComboBox;

Label4: TLabel;

DisplayTextCol: TComboBox;

TextBlock: TPanel;

PrevColBtn: TSpeedButton;

NextColBtn: TSpeedButton;

AutoSizeBtn: TButton;

HistoryHit: TListBox;

HistDifVal: TListBox;

HistFontHeight: TListBox;

FontHeightL: TLabel;

Label5: TLabel;

FontSizeL: TLabel;

Label7: TLabel;

LineHeightL: TLabel;

Label8: TLabel;

HistLineHeight: TListBox;

Bevel1: TBevel;

AutoSet: TCheckBox;

procedure SetDisplayFont;

```
  procedure FontNameChange(Sender: TObject);
  procedure FontSizeSpChange(Sender: TObject);
  procedure LoadPatternClick(Sender: TObject);
  procedure PttCharCChange(Sender: TObject);
  procedure FontHighSpChange(Sender: TObject);
  procedure LineHighSpChange(Sender: TObject);
  procedure FontBoldClick(Sender: TObject);
  procedure RecognizeChange(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure ScanImageDblClick(Sender: TObject);
  procedure ScanLineBtnClick(Sender: TObject);
  procedure ScanColumnBtnClick(Sender: TObject);
  procedure MapLineBtnClick(Sender: TObject);
  procedure MapAllLineBtnClick(Sender: TObject);
  procedure DisplayTextLineChange(Sender: TObject);
  procedure DisplayTextColChange(Sender: TObject);
  procedure PrevColBtnClick(Sender: TObject);
  procedure NextColBtnClick(Sender: TObject);
  procedure AutoSizeBtnClick(Sender: TObject);
private
  { Private declarations }
  procedure GetFontNames;
  procedure ScanColumn(LineNo : longint);
  procedure FindPttBaseLine;
  procedure LoadPttData(AsciiNo : longint);
  procedure LoadScanData(LineNo,SCol,ECol,SRow,ERow : longint);
  function  AsciiMapRight(LineNo,SCol,ECol,SRow,ERow : longint): string;
  function  ScanText(LineNo, StrCol, EndCol : longint): string;
  procedure DisplayBlockScan(BlockCol : longint);
public
  // กำหนดค่าให้กับ Pattern
  PttFontName : string;
  PttFontSize : longint;
  PttFontColor : TColor;
```

```
PttBgColor : TColor;

Margin : TMargin;
FontHigh : longint;
LineHigh : longint;
LineCount : longint;
ColCount : longint;
ShiftPrev : longint;
EndLineCol : longint;
PRecognize : longint;

// เก็บค่าต่างๆ ไว้ใช้ในการตัดตัวอักษร
MemShiftPrev : longint;
MemMidleTop : longint;
MemReg : boolean;
MemLower : TRect;
MemUpper1 : TRect;
MemUpper2 : TRect;

// เก็บแถว
BaseLineArray : array[0..50] of longint;  // Line number of A4 must less then 50 line
// เก็บ Column ใน 1 แถว
ColLineArray : array[0..200] of longint;

// เก็บค่าของ Pattern
PttBaseL : longint;
PttChar : TPatternData;
PttArray : array[32..255] of TPatternData;

ScanData : TLoadData;
BlankData : TLoadData;

ScanTextRun : boolean;
```

```
// Exp:
//   0
//   1      Midle Char
//   :
//   45  Baseline
//   46      Lower Char
//   :
//   55  DownLine
//   56      Upper Char
//   :
//   80  UpperChar


    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

// กำหนด Margin
function SetMargin(MLeft,MTop,MRight,MBottom : longint): TMargin;
begin
  with Result do
  begin
    Left := MLeft;
    Top := MTop;
    Right := MRight;
    Bottom := MBottom;
  end;
end;
```

```
// Add ชื่อ Font ลงใน Combobox
function EnumFontsProc(var LogFont: TLogFont; var TextMetric: TTextMetric;
 FontType: longint; Data: Pointer): longint; stdcall;
begin
 TStrings(Data).Add(LogFont.lfFaceName);
 Result := 1;
end;

// โหลด Font ของ Windows
procedure TForm1.GetFontNames;
var DC: HDC;
begin
 DC := GetDC(0);
 EnumFonts(DC, nil, @EnumFontsProc, Pointer(FontName.Items));
 ReleaseDC(0, DC);
 FontName.Sorted := True;
end;

// กำหนดค่าให้กับระบบเมื่อผู้ใช้เลือก Font
procedure TForm1.SetDisplayFont;
begin
 FontName.ItemIndex := FontName.Items.IndexOf(PttFontName);
 FontSizeSp.Text := IntToStr(PttFontSize);
 DisplayFont.Font.Name := FontName.Items[FontName.ItemIndex];
 DisplayFont.Font.Size := PttFontSize;

 if FontBold.Down then
  DisplayFont.Font.Style := [fsBold]
 else
  DisplayFont.Font.Style := [];

 PttImage.Canvas.Font.Name := DisplayFont.Font.Name;
 PttImage.Canvas.Font.Size := DisplayFont.Font.Size;
```

```
  PttImage.Canvas.Font.Charset := DisplayFont.Font.Charset;
  PttImage.Canvas.Font.Style := DisplayFont.Font.Style;

  ResultText.Font.Name := DisplayFont.Font.Name;
  ResultText.Font.Size := DisplayFont.Font.Size;
  ResultText.Font.Charset := DisplayFont.Font.Charset;
  ResultText.Font.Style := DisplayFont.Font.Style;

  TextBlock.Font.Name := DisplayFont.Font.Name;
  TextBlock.Font.Size := DisplayFont.Font.Size;
  TextBlock.Font.Charset := DisplayFont.Font.Charset;
  TextBlock.Font.Style := DisplayFont.Font.Style;
end;
// แสดงผลเมื่อมีการเปลี่ยน Font ใหม่
procedure TForm1.FontNameChange(Sender: TObject);
begin
  PttFontName := FontName.Items[FontName.ItemIndex];
  SetDisplayFont;
end;

procedure TForm1.FontSizeSpChange(Sender: TObject);
begin
  PttFontSize := StrToInt(FontSizeSp.Text);
  FontHighSp.Value := Round(PttFontSize*0.65);
  //  High of middle charecter estimate 65% of Font Size
  // ความสูงของตัวอักษรตรงกลาง เช่น 'A' จำมีค่าประมาณ 65% ของขนาด Font
  FontHigh := FontHighSp.Value;

  LineHighSp.Value := Round(PttFontSize*1.2);
  // Distance betaween line must more than 20% of Font Size
  // ความสูงของบรรทัดมีค่าไม่ต่ำกว่า 20% ของขนาด Font
  LineHigh := LineHighSp.Value;
  SetDisplayFont;
end;
```

```
procedure TForm1.FontHighSpChange(Sender: TObject);
begin
  FontHigh := FontHighSp.Value;
end;

procedure TForm1.LineHighSpChange(Sender: TObject);
begin
  LineHigh := LineHighSp.Value;
end;

procedure TForm1.FontBoldClick(Sender: TObject);
begin
  FontBold.AllowAllUp := not(FontBold.AllowAllUp);
  FontBold.Down := not(FontBold.Down);

  SetDisplayFont;
end;

procedure TForm1.RecognizeChange(Sender: TObject);
begin
  PRecognize := Recognize.Value;
end;

procedure TForm1.FormCreate(Sender: TObject);
var i,j : longint;
    CharStr : string;

  procedure AddChar;
  begin
    if ((i >= 209) and (i <= 217)) or ((i >= 231) and (i <= 237)) then
      CharStr := CharStr + '  ' + Chr(i)
    else CharStr := CharStr + Chr(i);
```

```
    if i in [64,96,126,160,206,237] then
    begin
      DisplayFont.Lines.Add(CharStr);
      CharStr := '';
    end;
  end;

begin
  // แสดงหมายเลขตัวอักษรลงใน Memo
  CharStr := '';
  for i := 32 to 126 do AddChar;

  for i := 161 to 217 do AddChar;

  for i := 223 to 237 do AddChar;

  for i := 240 to 249 do AddChar;

  DisplayFont.Lines.Add(CharStr);
  GetFontNames;
  PttFontName := DisplayFont.Font.Name;
  PttFontSize := -MulDiv(DisplayFont.Font.Height, 72, Screen.PixelsPerInch);
  PttFontColor := clBlack;
  PttBgColor := clWhite;
  RecognizeChange(nil);
  SetDisplayFont;

  // ลบค่าใน Block Pattern
  for i := 0 to 80 do
    for j := 0 to 60 do
      BlankData[i,j] := 0;
end;

// โหลดรูปเข้ามาใหม่
```

```
procedure TForm1.ScanImageDblClick(Sender: TObject);
begin
 if OpenPictureDialog1.Execute then
  if OpenPictureDialog1.FileName <> '' then
   ScanImage.Picture.LoadFromFile(OpenPictureDialog1.FileName);
end;

// โหลด Pattern ที่จะใช้เป็นต้นแบบในการ Map
procedure TForm1.LoadPatternClick(Sender: TObject);
var i : longint;
begin
 Screen.Cursor := crHourGlass;
 PttCharC.Clear;

 // โหลดค่า Ascii เก็บไว้เพื่ออ้างอิงในการโหลด Pattern
 for i := 32 to 126 do
  PttCharC.Items.Add(IntToStr(i));

 for i := 161 to 217 do
  PttCharC.Items.Add(IntToStr(i));

 for i := 223 to 237 do
  PttCharC.Items.Add(IntToStr(i));

 for i := 240 to 249 do
  PttCharC.Items.Add(IntToStr(i));

 // หาว่า Base Line ของ Pattern อยู่ตำแหน่งไหน
 FindPttBaseLine;

 // ทำการโหลด Pattern ทั้งหมดมาเก็บไว้ใน Array
 for i := 32 to 126 do
 begin
  LoadPttData(i);
```

```
  PttArray[i] := PttChar;
 end;

 // กำหนดให้ Ascii 32 (Spacebar) มีค่า Margin เท่ากับ Ascii 98 (b)
 PttArray[32].PMargin := PttArray[98].PMargin;  // 98=b

 // ทำการโหลด Pattern ที่เหลือทั้งหมดมาเก็บไว้ใน Array ต่อไป
 for i := 161 to 217 do
 begin
  LoadPttData(i);
  PttArray[i] := PttChar;
 end;

 for i := 223 to 237 do
 begin
  LoadPttData(i);
  PttArray[i] := PttChar;
 end;

 for i := 240 to 249 do
 begin
  LoadPttData(i);
  PttArray[i] := PttChar;
 end;
 Screen.Cursor := crDefault;
end;

// แสดงลักษณะของ Pattern เมื่อผู้ใช้เลือกค่า Ascii
procedure TForm1.PttCharCChange(Sender: TObject);
var Chr : byte;
begin
 if PttCharC.ItemIndex < 0 then Exit;
 Chr := StrToInt(PttCharC.Text);
 PttImage.Canvas.Lock;
```

```
  try
   with PttImage.Canvas do
   begin
    Brush.Color := clWhite;
    FillRect(PttImage.BoundsRect);
    Font.Color := clBlack;

    if ((Chr >= 209) and (Chr <= 217)) or ((Chr >= 231) and (Chr <= 237)) then
     TextOut(5,3,'  ' + Char(Chr) + ' ')
    else
     TextOut(5,3,Char(Chr));
   end;
  finally
   PttImage.Canvas.Unlock;
  end;
end;

// process ในการคำนวณหา Base Line ของ Pattern
// โดย scan จากด้านล่างขึ้นด้านบนจนพบเนื้อตัวอักษร
procedure TForm1.FindPttBaseLine;
var Row,Col : longint;
begin
 PttCharC.ItemIndex := PttCharC.Items.IndexOf(IntToStr(65));  // 65=A
 if PttCharC.ItemIndex > 0 then
 begin
  PttCharCChange(nil);
  for Row := PttImage.Height - 1 downto 0 do
   for Col := 0 to PttImage.Width - 1 do
    if PttImage.Canvas.Pixels[Col,Row] <> PttBgColor then
    begin
     PttBaseL := Row + 1;
     exit;
    end;
  end;
```

```
end;

// Process ในการโหลด Pattern ของตัวอักษร 1 ตัวมาเก็บไว้
// พร้อมทั้งคำนวณหา Margin ของตัวอักษรนั้นด้วย
procedure TForm1.LoadPttData(AsciiNo : longint);
var Row,Col : longint;
   FirstHit : boolean;
begin
 PttCharC.ItemIndex := PttCharC.Items.IndexOf(IntToStr(AsciiNo));
 if PttCharC.ItemIndex > -1 then
 begin
  PttCharCChange(nil);
  PttChar.PMargin := SetMargin(PttImage.Width - 1,PttImage.Height - 1,0,0);
  FirstHit := True;
  for Row := 0 to PttImage.Height - 1 do
   for Col := 0 to PttImage.Width - 1 do
    if PttImage.Canvas.Pixels[Col,Row] <> PttBgColor then
    begin
     PttChar.PData[Col,Row] := 1;

     if FirstHit then
     begin
      PttChar.PMargin.Top := Row;
      FirstHit := False;
     end;

     if Col < PttChar.PMargin.Left then PttChar.PMargin.Left := Col;
     if Col > PttChar.PMargin.Right then PttChar.PMargin.Right := Col;
     if Row > PttChar.PMargin.Bottom then PttChar.PMargin.Bottom := Row;
    end
    else PttChar.PData[Col,Row] := 0;
 end;
end;
```

```
// Process ในการ scan ข้อมูลจากรูปเข้ามาเก็บใน Block ScanData
procedure TForm1.LoadScanData(LineNo,SCol,ECol,SRow,ERow : longint);
var Row,Col : longint;
begin
 ScanData := BlankData;
 for Row := SRow to ERow do
  for Col := SCol to ECol do
   if ScanImage.Canvas.Pixels[Col,Row] = PttFontColor then
   begin
//     ScanImage.Canvas.Pixels[Col,Row] := clBlue; // Mark
     ScanData[Col - SCol, Row - SRow] := 1;
   end
   else
     ScanData[Col - SCol, Row - SRow] := 0;
end;

// process ในการ scan หา Base Line ทั้งหมดจากรูป
procedure TForm1.ScanLineBtnClick(Sender: TObject);
var Row,Col,ECol : longint;
  FirstHit : boolean;
  HitT : longint;
  LastHitT,LastBaseL,LastNoHit : longint;
  MLastHitT,MLastBaseL,MLastNoHit : longint;
  RowCheck,RowCheckC : longint;
  NextRowC1,NextRowC2 : longint;
begin
 Screen.Cursor := crHourGlass;
 ColumnScanLine.Clear;
 ScanTextLine.Clear;
 DisplayTextLine.Clear;
 DisplayTextCol.Clear;

 with ScanImage.Picture.Bitmap.Canvas do
 begin
```

```
FirstHit := True;
Margin := SetMargin(ScanImage.Width,ScanImage.Height,0,0);

LineCount := 0;
LastHitT := 0;
LastBaseL := 0;
LastNoHit := 0;
RowCheck := -1;

MLastHitT := -1;
MLastBaseL := -1;
MLastNoHit := -1;

// การกำหนดค่าในการกระโดดข้ามไป scan line ถัดไป
RowCheckC := Round(FontHigh*0.2); // Goto Midle Char
NextRowC1 := Round(FontHigh*0.6); // Goto Base Line
NextRowC2 := Round(FontHigh*0.8); // Goto NextLine

Row := 0;
ECol := ScanImage.Width - 1;
repeat
 HitT := 0;
 for Col := 0 to ECol do
  if Pixels[Col,Row] = PttFontColor then
  begin
   HitT := HitT + 1;

   // หา Margin ของรูป
   if FirstHit then
   begin
    Margin.Top := Row;
    FirstHit := False;
   end;
```

```
    if Col < Margin.Left then Margin.Left := Col;
    if Col > Margin.Right then Margin.Right := Col;
    if Row > Margin.Bottom then Margin.Bottom := Row;
   end;

Application.ProcessMessages;

if HitT <= LastHitT*0.2 then
begin  // Base Line is near HitT decread near 0
      // HitT near 0 except some charecters [gypq estimate <= 20% of Base Line]

  if (Row - LastBaseL >= LineHigh) and
    (Row - LastNoHit  >= FontHigh) then
  begin
    MLastHitT := LastHitT;
    MLastBaseL := LastBaseL;
    MLastNoHit := LastNoHit;
    LastBaseL := Row;
    LastNoHit := Row;

    BaseLineArray[LineCount] := Row;
    LineCount := LineCount + 1;

    RowCheck := Row + RowCheckC;
    Row := RowCheck - 1;
  end;
end;

if (RowCheck = Row) then
begin
  if (HitT >= MLastHitT*1.2) then
  begin  // if Upper char then HitT must more than mid char 20% of MLastHitT
       // And then reset data
```

```
    LastBaseL := MLastBaseL;
    LastNoHit := MLastNoHit;
    BaseLineArray[LineCount] := 0;
    LineCount := LineCount - 1;
    Row := Row + NextRowC1 - 1;
   end
   else Row := Row + NextRowC2 - 1;
  end;

  if HitT = 0 then LastNoHit := Row;

  LastHitT := HitT;
  Row := Row + 1;
 until Row > ScanImage.Height - 1;

 Margin.Right := Margin.Right + PttFontSize;
 Margin.Bottom := Margin.Bottom + PttFontSize;
 // Add pixel not scan

 for Row := 0 to LineCount - 1 do
  ColumnScanLine.Items.Add(IntToStr(Row + 1));

 ScanTextLine.Items.Assign(ColumnScanLine.Items);
 DisplayTextLine.Items.Assign(ColumnScanLine.Items);

 if ColumnScanLine.Items.Count > 0 then
 begin
  ColumnScanLine.ItemIndex := 0;
  ScanTextLine.ItemIndex := 0;
  DisplayTextLine.ItemIndex := 0;
  DisplayTextLineChange(nil);
 end;

 if DisplayRow.Checked then
```

```
    begin
     Pen.Color := clRed;  // Display Line
     for Row := 0 to LineCount - 1 do
     begin
       MoveTo(Margin.Left,BaseLineArray[Row]);
       LineTo(Margin.Right,BaseLineArray[Row]);
     end;
    end;
  end;
  Screen.Cursor := crDefault;
end;

// process ในการ scan column ทั้งหมดใน 1 line
procedure TForm1.ScanColumn(LineNo : longint);
var Row,Col : longint;
    SRow,ERow : longint;
    HitT : longint;
    LastHitT : longint;
begin
  with ScanImage.Picture.Bitmap.Canvas do
  begin
    ColCount := 0;
    LastHitT := 0;
    EndLineCol := 0;

    for Col := Margin.Left to Margin.Right do
    begin
      SRow := BaseLineArray[LineNo] - FontHigh + 1;   // 40% Font High
      ERow := BaseLineArray[LineNo] + Round(FontHigh*0.4); // ฐ
      HitT := 0;
      for Row := SRow to ERow do
        if Pixels[Col,Row] = PttFontColor then
          HitT := HitT + 1;
```

```
    Application.ProcessMessages;

    if (HitT > 0) and (LastHitT = 0) then
    begin
     ColLineArray[ColCount] := Col;
     ColCount := ColCount + 1;
    end;

    if HitT > 0 then EndLineCol := Col;

    LastHitT := HitT;
   end;

   if DisplayCol.Checked then
   begin
    Pen.Color := clRed;  // Display Column
    for Col := 0 to ColCount - 1 do
    begin
     MoveTo(ColLineArray[Col],BaseLineArray[LineNo] + Round(FontHigh*0.4));
     LineTo(ColLineArray[Col],BaseLineArray[LineNo] + Round(FontHigh*0.2));
    end;
   end;
  end;
end;

procedure TForm1.ScanColumnBtnClick(Sender: TObject);
begin
 if ColumnScanLine.ItemIndex < 0 then Exit;
 ScanColumn(StrToInt(ColumnScanLine.Text) - 1);
end;

// process ในการเปรียบเทียบค่าที่โหลดเข้ามาเก็บไว้ใน ScanData กับ ค่า
// ที่เก็บไว้ใน Pattern เพื่อจะดูว่าค่าที่ได้ควรจะเป็นตัวอะไร
function TForm1.AsciiMapRight(LineNo,SCol,ECol,SRow,ERow : longint): string;
```

```
var vSRow,vERow : longint;
    vSCol,vECol : longint;
    AllPos : longint;
    i,SamePos : longint;
    Ascii : longint;
    MapMaxRight : longint;
    MapSamePos : longint;
    LowerAscii : longint;
    UpperAscii1 : longint;
    UpperAscii2 : longint;
    STop,SLeft : longint;
    MemUpRow,UpRow : longint;

    // scan หาขอบบนของ Block ตัวอักษร
    procedure ScanTop;
    var Row,Col : longint;
    begin
     STop := -1;
     for Row := vSRow to vERow do
     begin
       for Col := vSCol to vECol do
        if ScanData[Col,Row] = 1 then
        begin
          STop := Row - vSRow;
          break;
        end;
       if STop > -1 then break;
     end;
    end;

    // scan หาขอบซ้ายของ Block ตัวอักษร
    procedure ScanLeft;
    var Row,Col : longint;
    begin
```

```
  SLeft := -1;
  for Col := vSCol to vECol do
  begin
   for Row := vSRow to vERow do
    if ScanData[Col,Row] = 1 then
    begin
     SLeft := Col - vSCol;
     break;
    end;
   if SLeft > -1 then break;
  end;
end;

// ตรวจสอบตัวอักษรที่อยู่กลางบรรทัด
procedure MapMidleAscii;
var Row,Col : longint;
   vDifT : longint;
begin
 vSRow := PttBaseL - Round(FontHigh*1.8); // ล้ำไปด้านบนไม่เกิน 80% ของความสูง
 vERow := PttBaseL + Round(FontHigh*0.5); // ล้ำไปด้านล่างไม่เกิน 50% ของความสูง
 vDifT := 0;

 // กำหนดขอบเขตอีกครั้งตามขนาดของตัวอักษร
 if PttArray[Ascii].PMargin.Top > vSRow then
 begin
  vDifT := PttArray[Ascii].PMargin.Top - vSRow;
  vSRow := PttArray[Ascii].PMargin.Top;
 end;

 if PttArray[Ascii].PMargin.Bottom < vERow then
  vERow := PttArray[Ascii].PMargin.Bottom;

 vSCol := PttArray[Ascii].PMargin.Left;
 vECol := PttArray[Ascii].PMargin.Right;
```

```
  AllPos := 0;
  SamePos := 0;
  for Row := vSRow to vERow do
   for Col := vSCol to vECol do
   begin
     if PttArray[Ascii].PData[Col,Row] = ScanData[Col - vSCol,Row - vSRow + vDifT] then
       SamePos := SamePos + 1;

     AllPos := AllPos + 1;
   end;

  if AllPos > 0 then
  begin  // ตรวจสอบ percent ความถูกต้อง
   PttArray[Ascii].PRight := Round(SamePos*100/AllPos);
   if (PttArray[Ascii].PRight > MapMaxRight) or
     ((PttArray[Ascii].PRight = MapMaxRight) and (SamePos > MapSamePos)) then
   begin
     MapMaxRight := PttArray[Ascii].PRight;
     MapSamePos := SamePos;
     Result := Chr(Ascii);
   end;
  end;
 end;

// ตรวจสอบตัวอักษรที่อยู่กลางบรรทัด แบบพิเศษ 1
procedure MapMidleAscii1;
var Row,Col : longint;
begin
  vSCol := PttArray[Ascii].PMargin.Left;
  vECol := PttArray[Ascii].PMargin.Right;

  AllPos := 0;
  SamePos := 0;
```

```
  for Row := vSRow to vERow do
   for Col := vSCol to vECol do
   begin
    if PttArray[Ascii].PData[Col,Row] = ScanData[Col - vSCol,Row - vSRow] then
     SamePos := SamePos + 1;

    AllPos := AllPos + 1;
   end;

  if AllPos > 0 then
  begin  // ตรวจสอบ percent ความถูกต้อง
   PttArray[Ascii].PRight := Round(SamePos*100/AllPos);
   if (PttArray[Ascii].PRight > MapMaxRight) or
    ((PttArray[Ascii].PRight > PRecognize) and (SamePos > MapSamePos)) then
   begin
    MapMaxRight := PttArray[Ascii].PRight;
    MapSamePos := SamePos;
    Result := Chr(Ascii);
    MemShiftPrev := ShiftPrev;
    MemMidleTop := vSRow;
   end;
  end;
 end;

// ตรวจสอบตัวอักษรที่อยู่กลางบรรทัด แบบพิเศษ 2
procedure MapMidleAscii2;
var Row,Col : longint;
  vDifT : longint;
begin
 vSRow := PttBaseL - Round(FontHigh*1.8);
 vERow := PttBaseL + Round(FontHigh*0.5);
 vDifT := 0;

 if PttArray[Ascii].PMargin.Top > vSRow then
```

```
begin
  vDifT := PttArray[Ascii].PMargin.Top - vSRow;
  vSRow := PttArray[Ascii].PMargin.Top;
end;

if PttArray[Ascii].PMargin.Bottom < vERow then
  vERow := PttArray[Ascii].PMargin.Bottom;

vSCol := PttArray[Ascii].PMargin.Left;
vECol := PttArray[Ascii].PMargin.Right;

AllPos := 0;
SamePos := 0;
for Row := vSRow to vERow do
  for Col := vSCol to vECol do
  begin
    if PttArray[Ascii].PData[Col,Row] = 1 then
    begin
      if PttArray[Ascii].PData[Col,Row] = ScanData[Col - vSCol,Row - vSRow + vDifT] then
        SamePos := SamePos + 1;

      AllPos := AllPos + 1;
    end;
  end;

if AllPos > 0 then
begin // ตรวจสอบ percent ความถูกต้อง
  PttArray[Ascii].PRight := Round(SamePos*100/AllPos);
  if PttArray[Ascii].PRight > PRecognize then
  begin
    MapMaxRight := PttArray[Ascii].PRight;
    MapSamePos := SamePos;
    Result := Chr(Ascii);
    MemMidleTop := vSRow;
```

```
    end;
  end;
end;

// ตรวจสอบตัวอักษรที่อยู่ล่างบรรทัด
procedure MapLowerAscii;
var Row,Col : longint;
begin
  vSRow := PttArray[Ascii].PMargin.Top;
  vERow := PttArray[Ascii].PMargin.Bottom;
  vSCol := PttArray[Ascii].PMargin.Left;
  vECol := PttArray[Ascii].PMargin.Right;

  AllPos := 0;
  SamePos := 0;
  for Row := vSRow to vERow do
    for Col := vSCol to vECol do
    begin
      if PttArray[Ascii].PData[Col,Row] = 1 then
      begin
        if PttArray[Ascii].PData[Col,Row] = ScanData[Col - vSCol + SLeft,Row - vSRow + STop] then
          SamePos := SamePos + 1;

        AllPos := AllPos + 1;
      end;
    end;

  if AllPos > 0 then
  begin // ตรวจสอบ percent ความถูกต้อง
    PttArray[Ascii].PRight := Round(SamePos*100/AllPos);
    if (PttArray[Ascii].PRight > MapMaxRight) or
      ((PttArray[Ascii].PRight = MapMaxRight) and (SamePos > MapSamePos)) then
    begin
      MapMaxRight := PttArray[Ascii].PRight;
```

```
    MapSamePos := SamePos;
    LowerAscii := Ascii;

    if MemReg then
    begin
     MemLower.Left := SLeft;
     MemLower.Top := STop;
     MemLower.Right := vECol - vSCol + 1;
     MemLower.Bottom := vERow - vSRow + 1;
    end;
   end;
  end;
end;

// ตรวจสอบตัวอักษรที่อยู่บนบรรทัด แบบพิเศษ 1 (บนสุด)
procedure MapUpperAscii1;
var Row,Col : longint;
begin
  vSRow := PttArray[Ascii].PMargin.Top;
  vERow := PttArray[Ascii].PMargin.Bottom;
  vSCol := PttArray[Ascii].PMargin.Left;
  vECol := PttArray[Ascii].PMargin.Right;

  AllPos := 0;
  SamePos := 0;
  for Row := vSRow to vERow do
   for Col := vSCol to vECol do
   begin
    if PttArray[Ascii].PData[Col,Row] = 1 then
    begin
     if PttArray[Ascii].PData[Col,Row] = ScanData[Col - vSCol + SLeft,Row - vSRow + STop] then
      SamePos := SamePos + 1;

     AllPos := AllPos + 1;
```

```
      end;
     end;

    if AllPos > 0 then
    begin // ตรวจสอบ percent ความถูกต้อง
     PttArray[Ascii].PRight := Round(SamePos*100/AllPos);
     if PttArray[Ascii].PRight > 70 then
     if (PttArray[Ascii].PRight > MapMaxRight) or
       ((PttArray[Ascii].PRight = MapMaxRight) and (SamePos > MapSamePos)) then
     begin
      MapMaxRight := PttArray[Ascii].PRight;
      MapSamePos := SamePos;
      UpperAscii1 := Ascii;
      MemUpRow := UpRow;

      if MemReg then
      begin
       MemUpper1.Left := SLeft;
       MemUpper1.Top := STop;
       MemUpper1.Right := vECol - vSCol + 1;
       MemUpper1.Bottom := vERow - vSRow + 1;
      end;
     end;
    end;
   end;

  // ตรวจสอบตัวอักษรที่อยู่บนบรรทัด แบบพิเศษ /
  procedure MapUpperAscii2;
  var Row,Col : longint;
  begin
   vSRow := PttArray[Ascii].PMargin.Top;
   vERow := PttArray[Ascii].PMargin.Bottom;
   vSCol := PttArray[Ascii].PMargin.Left;
   vECol := PttArray[Ascii].PMargin.Right;
```

```
AllPos := 0;
SamePos := 0;
for Row := vSRow to vERow do
 for Col := vSCol to vECol do
 begin
  if PttArray[Ascii].PData[Col,Row] = 1 then
  begin
   if PttArray[Ascii].PData[Col,Row] = ScanData[Col - vSCol + SLeft,Row - vSRow + STop] then
    SamePos := SamePos + 1;

   AllPos := AllPos + 1;
  end;
 end;

if AllPos > 0 then
begin // ตรวจสอบ percent ความถูกต้อง
 PttArray[Ascii].PRight := Round(SamePos*100/AllPos);
 if PttArray[Ascii].PRight > 70 then
 if (PttArray[Ascii].PRight > MapMaxRight) or
   ((PttArray[Ascii].PRight = MapMaxRight) and (SamePos > MapSamePos)) then
 begin
  MapMaxRight := PttArray[Ascii].PRight;
  MapSamePos := SamePos;
  UpperAscii2 := Ascii;

  if MemReg then
  begin
   MemUpper2.Left := SLeft;
   MemUpper2.Top := STop;
   MemUpper2.Right := vECol - vSCol + 1;
   MemUpper2.Bottom := vERow - vSRow + 1;
  end;
 end;
```

```
    end;
  end;

  // วน Loop เพื่อตรวจสอบตัวอักษรที่อยู่บนบรรทัด
  procedure LoopMapUpperAscii;
  begin
    vSRow := MemUpRow + 1;
    vERow := Round(FontHigh*0.8);
    vSCol := 0;
    vECol := PttArray[Ord(Result[1])].PMargin.Right - PttArray[Ord(Result[1])].PMargin.Left +
          Round(FontHigh*0.3);

    ScanTop;
    if STop > -1 then
    begin
      STop := STop + vSRow;

      ScanLeft;

      if SLeft > -1 then
        if Result[1] in ['ป','ฝ','ฟ','ฬ'] then
          if SLeft >= PttArray[Ord(Result[1])].PMargin.Right - PttArray[Ord(Result[1])].PMargin.Left +
1 then
            SLeft := -1;

      if SLeft > -1 then
        MapUpperAscii2;
    end;
  end;

begin
  Result := '';
  MapMaxRight := 0;
  MapSamePos := 0;
```

```
  ShiftPrev := 0;

(*
   Ascii 32..126
// !"#$%&'()*+,-./0123456789:;<=>?@'#$D'ABCDEFGHIJKLMNOPQRSTUVWXYZ
//[\]^_`'#$D'abcdefghijklmnopqrstuvwxyz{|}~
   Ascii 161..217
//กขฃคฅฆงจฉชซฌญฎฏฐฑฒณดตถทธนบปผฝพฟภมยรฤลฦวศษสหฬอฮฯะัาำ
   Ascii 223..237
//฿แโใไๅๆ็่้๊๋์
   Ascii 240..249
//๐๑๒๓๔๕๖๗๘๙ *)

  LoadScanData(LineNo,SCol,ECol,SRow,ERow); // โหบด Block ข้อมูลรูปภาพมาเก็บไว้เพื่อ Map

  for Ascii := 32 to 126 do MapMidleAscii;  // Map ตัวอักษรที่ละชุด
  for Ascii := 161 to 211 do MapMidleAscii;
  for Ascii := 223 to 230 do MapMidleAscii;
  for Ascii := 240 to 249 do MapMidleAscii;

  if MapMaxRight > PRecognize then
  begin
   if Result = 'ๆ' then  // 224 = ๆ
   begin
    for Ascii := 226 to 228 do  // ๅ ็ ่
    begin
     // หาระยะขอบซ้ายของสระโดยหาจากครึ่งหนึ่งของความสูง (ๅ ็ ่)
     for i := PttArray[Ascii].PMargin.Left to PttArray[Ascii].PMargin.Right do
      if PttArray[Ascii].PData[i,PttArray[Ascii].PMargin.Bottom - Round(FontHigh*0.5)] = 1 then //
50% Font high estimate left 'ๆ'
       break;

     // ได้ระยะขอบด้านซ้ายที่จะต้องเลื่อน
     ShiftPrev := i - PttArray[Ascii].PMargin.Left;
```

```
    // ขยายขอบเขตของตัวอักษรที่จะโหลด
    vSCol := SCol - ShiftPrev;
    vECol := vSCol + PttArray[Ascii].PMargin.Right - PttArray[Ascii].PMargin.Left;
    vSRow := SRow + Round(FontHigh*1.8) - PttBaseL + PttArray[Ascii].PMargin.Top;
    vERow := vSRow + PttArray[Ascii].PMargin.Bottom - PttArray[Ascii].PMargin.Top;

    // โหบด Block ข้อมูลรูปภาพมาเก็บไว้เพื่อ Map อีกครั้ง
    LoadScanData(LineNo,vSCol,vECol,vSRow,vERow);

    vSRow := PttArray[Ascii].PMargin.Top;
    vERow := PttArray[Ascii].PMargin.Bottom;

    // Map สระ
   MapMidleAscii1; // ต้องใช้เงื่อนไขในการตรวจสอบแตกต่างจากเดิม MapMidleAscii
  end;

  if Result = 'ไ' then ShiftPrev := 0; // ถ้าไม่ใช่สระ (ไ ใ ไ) ให้กำหนดระยะขอบเป็นศูนย์เหมือนเดิม
end;

// Map ตัวอักษรที่ใกล้เคียงกัน
if Result = 'พ' then
begin
  Ascii := Ord('ฟ');
 MapMidleAscii2; // ต้องใช้เงื่อนไขในการตรวจสอบแตกต่างจากเดิม MapMidleAscii
end;

// Map ตัวอักษรที่ใกล้เคียงกัน
if (Result = 'ป') or (Result = 'ผ') or (Result = 'พ') then
begin
  Ascii := Ord(Result[1]) + 1;
 MapMidleAscii2; // ต้องใช้เงื่อนไขในการตรวจสอบแตกต่างจากเดิม MapMidleAscii
end;
```

```
// Map ตัวอักษรที่ใกล้เคียงกัน
if (Result >= 'ก') and (Result <= 'ฮ') then
begin
 //  Check Lower Ascii
 if not(Result[1] in ['ฎ','ฏ','ฐ','ฤ','ฦ']) then
 begin
  LowerAscii := -1;

  MapMaxRight := 0;
  MapSamePos := 0;

  vSRow := Round(FontHigh*1.8);
  vERow := vSRow + Round(FontHigh*0.5);
  vSCol := 0;
  vECol := PttArray[Ord(Result[1])].PMargin.Right - PttArray[Ord(Result[1])].PMargin.Left;

  ScanTop;
  if STop > -1 then
  begin
   STop := STop + vSRow;

   ScanLeft;

   if SLeft > -1 then
    for Ascii := 216 to 217 do  //ฺ ฺ
     MapLowerAscii;
  end;

  if LowerAscii > -1 then
   if not((LowerAscii = 216) and (Result = 'ญ')) then
    Result := Result + Chr(LowerAscii);
end;

// ขยายขอบเขตของตัวอักษรที่จะโหลด เนื่องจากเป็นสระบน
```

```
    // ระยะขอบซ้ายมีไม่เกิน 10% ของความสูงตัวอักษร เช่น สระอิ
    // ระยะขอบขวามีไม่เกิน 20% ของความสูงตัวอักษร  เช่น ไม้หันอากาศ
    // ขนาดไม่เกิน 80% ของความสูงตัวอักษร
    LoadScanData(LineNo,SCol - Round(FontHigh*0.1),ECol + Round(FontHigh*0.2),SRow,SRow +
Round(FontHigh*0.8));
    //  Check Upper 1
    UpperAscii1 := -1;

    MapMaxRight := 0;
    MapSamePos := 0;

    MemUpRow := -1;
    for Ascii := 232 to 235 do  //'
    begin
     vSRow := 0;
     vERow := 0;

     if PttArray[Ascii].PMargin.Bottom - PttArray[Ascii].PMargin.Top > vERow then
      vERow := PttArray[Ascii].PMargin.Bottom - PttArray[Ascii].PMargin.Top;

     vSCol := 0;
     // ความกว้างของตัวอักษร = ขอบขวา - ขอบซ้าย + (0.1 + 0.2 =) 0.3 ของความสูง เนื่องจากเริ่มจาก
vSCol = 0
     vECol := PttArray[Ord(Result[1])].PMargin.Right - PttArray[Ord(Result[1])].PMargin.Left +
          Round(FontHigh*0.3);

     ScanTop;
     if STop > -1 then
     begin
      STop := STop + vSRow;

      vERow := vERow + STop;
      UpRow := vERow;
      ScanLeft;
```

```
      if SLeft > -1 then
        if Result[1] in ['ป','ฝ','ฟ','พ'] then
          if SLeft >= PttArray[Ord(Result[1])].PMargin.Right - PttArray[Ord(Result[1])].PMargin.Left +
1 then
            SLeft := -1;

      if SLeft > -1 then
        MapUpperAscii1;
    end;
  end;

  // Check Upper 2
  UpperAscii2 := -1;

  MapMaxRight := 0;
  MapSamePos := 0;

  Ascii := 209; //ั
  LoopMapUpperAscii;

  for Ascii := 212 to 215 do //ิ
    LoopMapUpperAscii;

  for Ascii := 231 to 237 do //็
    LoopMapUpperAscii;

  if UpperAscii2 > -1 then
    Result := Result + Chr(UpperAscii2);

  if UpperAscii1 > -1 then
    Result := Result + Chr(UpperAscii1);
  end;
end;
```

```
  Caption := IntToStr(MapMaxRight);
end;

// ทำการส่งข้อมูลไปแปลโดยจะหาขอบเขตของ Block แล้วส่งไปแปล
// ใน Procedure AsciiMapRight
function  TForm1.ScanText(LineNo, StrCol, EndCol : longint): string;
var Col : longint;
    SCol,ECol : longint;
    SRow,ERow : longint;
    Ascii : string;
    AsciiNo : longint;
    SpaceBar : longint;
    OneSpace : real;
    SkipCol : boolean;
    SpAscii : string;
    ColLimit : real;
    LenStr : longint;
begin
  if ScanTextRun then Exit;
  ScanTextRun := True;
  Screen.Cursor := crHourGlass;

  Result := '';
  ScanColumn(LineNo);
  if EndCol = 0 then EndCol := ColCount - 1;

  OneSpace := FontHigh*0.67; // SpaceBar + Space Betaween Charector
  SkipCol := False;
  SpAscii := '';
  MemReg := True;
  for Col := StrCol to EndCol do
  begin
    if not(SkipCol) then
```

```
begin
  SCol := ColLineArray[Col];
  ECol := ColLineArray[Col] + 80; // ขนาดของ Block ที่กำหนดไว้ในการโหลด

  SRow := BaseLineArray[LineNo] - Round(FontHigh*1.8); // ด้านบนไม่เกิน 1.8 เท่าของความสูงของ
ตัวอักษร
  ERow := BaseLineArray[LineNo] + Round(FontHigh*0.5); // ด้านล่างไม่เกิน 0.5 เท่าของความสูง
ของตัวอักษร

  if Col < ColCount - 1 then // เปรียบเทียบว่ามีค่ามากกว่า Column ถัดไปหรือไม่ ถ้าใช้ก็ลดขนาดของ
Block ลง
  begin
    if ECol > ColLineArray[Col + 1] - 1 then
      ECol := ColLineArray[Col + 1] - 1;
  end
  else ECol := ColLineArray[Col] + FontHigh;  // FontWidt <= FontHigh

  if Col = 0 then // คำนวณหาระยะย่อหน้าของข้อความ
    Result := StringOfChar(' ',Round((ColLineArray[Col] - Margin.Left) / OneSpace));

  // Map ตัวอักษรที่โหลดไว้ใน Block กับ Pattern ตัวอักษรที่เลือกไว้
  Ascii := AsciiMapRight(LineNo,SCol,ECol,SRow,ERow);

  AsciiNo := Ord(Ascii[1]);

  // ถ้าเป็นสระแอก็ให้เลื่อนการตรวจสอบไปอีก 1 ตัวเนื่องจากจะ Map ได้เป็นสระเอสองตัวติดกัน
  if Ascii = 'แ' then SkipCol := True;

  // กรณีที่เป็นการเว้นวรรคของตัวอักษร
  if (SpAscii <> '') and (ShiftPrev > 0) then
    SpAscii := Copy(SpAscii,1,Length(SpAscii) - 1);

  LenStr := Length(Result);
  if LenStr > 0 then
```

```
    if Ascii[1] in ['โ','ใ','ไ'] then // เป็นสระที่เยื้องไปด้านซ้าย
    begin
      if Result[LenStr] in ['เ','แ','โ','ใ','ไ','ำ'] then // จึงต้องตัดออก
        Result := Copy(Result,1,LenStr - 1);
    end
    else if Result[LenStr] = 'ำ' then // ตรวจสอบสระ อำ
    begin
      Result := Copy(Result,1,LenStr - 1);
      if Ascii[1] = 'า' then Ascii := 'ำ';
    end;
  Result := Result + SpAscii + Ascii;

  SpAscii := '';

  if Col < ColCount - 1 then // คำนวนการเว้นวรรคข้อความ
    SpaceBar := ColLineArray[Col + 1] - ColLineArray[Col] -
          (PttArray[AsciiNo].PMargin.Right - PttArray[AsciiNo].PMargin.Left + 1)
  else
    SpaceBar := Round(OneSpace);

  if Round(SpaceBar / OneSpace) > 0 then // คำนวณหาตัวอักษรตัวต่อไปที่ติดกันไม่สามารถแยกออก
จากการ scan column ได้
  begin // ต้องแยกออกโดยการ Map อักษรแล้วหาความกว้างตัวอักษรที่ Map ได้แล้ว Cut ส่วนที่เหลือ
จากนั้นก็วนลูป Map ต่อไปจนหมด Block
    SCol := ColLineArray[Col] + PttArray[AsciiNo].PMargin.Right - PttArray[AsciiNo].PMargin.Left
+ 1 - ShiftPrev;
    ECol := SCol + 80;

    repeat
      if Col < ColCount - 1 then
      begin
        if ECol > ColLineArray[Col + 1] - 1 then
          ECol := ColLineArray[Col + 1] - 1;
      end
```

```
      else ECol := ColLineArray[Col] + FontHigh;  // FontWidt <= FontHigh

      // Map ตัวอักษรที่โหลดไว้ใน Block กับ Pattern ตัวอักษรที่เลือกไว้อีกครั้ง
      Ascii := AsciiMapRight(LineNo,SCol,ECol,SRow,ERow);

      AsciiNo := Ord(Ascii[1]);

      // ถ้าเป็นสระแอก็ให้เลื่อนการตรวจสอบไปอีก 1 ตัวเนื่องจากจะ Map ได้เป็นสระเอสองตัวติดกัน
      if Ascii[1] = 'แ' then SkipCol := True;

      // ตัวอักษรเว้นวรรค
      if Ascii[1] <> ' ' then
        Result := Result + Ascii
      else
        SpAscii := SpAscii + ' ';

      // คำนวณหาขอบซ้ายของตัวอักษรตัวต่อไป
      SCol := SCol + PttArray[AsciiNo].PMargin.Right - PttArray[AsciiNo].PMargin.Left + 1 -
ShiftPrev;
      ECol := SCol + 80;

      if Col < ColCount - 1 then
       ColLimit := ColLineArray[Col + 1] - 0.25*FontHigh // ขนาดของตัวอักษรตัวถัดไปนั้นจะต้องมี
ขนาด
      else                                  // มากกว่าหรือเท่ากับ 25% ของความสูงของตัวอักษร
        ColLimit := EndLineCol;
     until SCol >= ColLimit;
    end;
   end
   else SkipCol := False;
   MemReg := False;
  end;
  ScanTextRun := False;
  Screen.Cursor := crDefault;
```

```
 end;

 // ทำการ Map ตัวอักษรที่ละบรรทัดตามที่เลือก
 procedure TForm1.MapLineBtnClick(Sender: TObject);
 begin
  if ScanTextLine.ItemIndex < 0 then Exit;
  ResultText.Clear;
  ResultText.Lines.Add(ScanText(StrToInt(ScanTextLine.Text) - 1,0,0));
 end;

 // ทำการ Map ตัวอักษรทั้งหมดทุกบรรทัด
 procedure TForm1.MapAllLineBtnClick(Sender: TObject);
 var LineNo : integer;
 begin
  ResultText.Clear;
  for LineNo := 0 to LineCount - 1 do
   ResultText.Lines.Add(ScanText(LineNo,0,0));
 end;

 // แสดงการตัดตัวอักษร
 procedure TForm1.DisplayBlockScan(BlockCol : longint);
 var LineNo : longint;
    Col,TextLen : longint;
    SCol,ECol : longint;
    SRow,ERow : longint;
    DisplayText : string;
    AsciiNo : longint;
    DestRect : TRect;
    MulX,MulY : longint;
    HaveSpecialC : boolean;
 begin
  Screen.Cursor := crHourGlass;
  DisplayText := '';
  LineNo := StrToInt(ColumnScanLine.Text) - 1;
```

```
MemShiftPrev := 0;
MemMidleTop := 0;

// ทำการ Map ทีละ 1 column ตามที่เลือก
DisplayText := ScanText(LineNo,BlockCol,BlockCol + 1);

// ทำการตัดการเว้นวรรคของข้อความด้านหน้าออก
while DisplayText[1] = ' ' do
  if DisplayText[1] = ' ' then
    DisplayText := Copy(DisplayText,2,Length(DisplayText));

HaveSpecialC := False;
TextLen := Length(DisplayText);
for Col := 2 to TextLen do // ตรวจสอบสระบนสระล่างเนื่องจากจะแสดงการตัดทีละ 1 column เท่านั้น
จึงมีตัวอักษรสระบนและสระล่างเท่านั้น
  if not(DisplayText[Col] in [ ]) then
  begin
    if DisplayText[Col] = 'ำ' then
      HaveSpecialC := True;

    DisplayText := Copy(DisplayText,1,Col - 1);

    break;
  end;

AsciiNo := Ord(DisplayText[1]); // นำตัวอักษรที่ Map ได้ไปแสดงการตัด
// โดยจะแบ่งการแสดงผลออกเป็น 5 ส่วน
// ส่วนแรกจะแสดง Block ที่ตัดมาจาก column ที่ scan ได้
// ส่วนที่สองจะแสดงตัวอักษรตรงกลาง
// ส่วนที่สามจะแสดงสระล่างของตัวอักษร
// ส่วนที่สี่จะแสดงสระบนชั้นที่ 2
// ส่วนที่ห้าจะแสดงสระบนชั้นที่ 1
// ดังข้างล่าง
//
```

```
// ลำดับการเรียง
//
// สระบนชั้นที่ 1
// สระบนชั้นที่ 2
// ตัวอักษร
// สระล่าง
//

//========================================================== Draw Image Block

SCol := ColLineArray[BlockCol];
ECol := ColLineArray[BlockCol] + 80;

SRow := BaseLineArray[LineNo] - Round(FontHigh*1.8);
ERow := BaseLineArray[LineNo] + Round(FontHigh*0.5);

if BlockCol < ColCount - 1 then
begin
  if ECol > ColLineArray[BlockCol + 1] - 1 then
    ECol := ColLineArray[BlockCol + 1] - 1;
end
else ECol := ColLineArray[BlockCol] + FontHigh;  // FontWidt < FontHigh

if DisplayText[1] in ['ิ','ี','ึ','ื'] then
  SCol := SCol - MemShiftPrev;

if Length(DisplayText) = 1 then
  ECol := SCol + PttArray[AsciiNo].PMargin.Right - PttArray[AsciiNo].PMargin.Left + 1;

MulX := ImageBlock.Width Div (ECol - SCol);
MulY := ImageBlock.Height Div (ERow - SRow);

with DestRect do
begin
```

```
  Left := 0;
  Top := 0;

  if MulX > MulY then MulX := MulY;

  Right := MulX*(ECol - SCol);
  Bottom := MulX*(ERow - SRow);

  Left := (ImageBlock.Width - Right) Div 2;
  Right := Left + Right;
  Top := (ImageBlock.Height - Bottom) Div 2;
  Bottom := Top + Bottom;
end;

with ImageBlock.Canvas do
begin
  Pen.Color := clRed;
  Brush.Color := clWhite;
  FillRect(ImageBlock.BoundsRect);
  CopyRect(DestRect,ScanImage.Canvas,Rect(SCol,SRow,ECol,ERow));
  Brush.Color := clRed;
  FrameRect(Rect(DestRect.Left - 1, DestRect.Top - 1, DestRect.Right + 1, DestRect.Bottom + 1));
end;

//========================================================== Draw Midle Char

  SRow := BaseLineArray[LineNo] - PttBaseL + PttArray[AsciiNo].PMargin.Top;
  ERow := SRow + PttArray[AsciiNo].PMargin.Bottom - PttArray[AsciiNo].PMargin.Top + 1;

  ECol := SCol + PttArray[Ord(DisplayText[1])].PMargin.Right - PttArray[Ord(DisplayText
[1])].PMargin.Left + 1;

  with DestRect do
  begin
```

```
  Left := 0;
  Top := 0;

  Right := MulX*(ECol - SCol);
  Bottom := MulX*(ERow - SRow);

  Left := (ImageMidle.Width - Right) Div 2;
  Right := Left + Right;
  Top := (ImageMidle.Height - Bottom) Div 2;
  Bottom := Top + Bottom;
 end;

 with ImageMidle.Canvas do
 begin
  Pen.Color := clRed;
  Brush.Color := clWhite;
  FillRect(ImageMidle.BoundsRect);
  CopyRect(DestRect,ScanImage.Canvas,Rect(SCol,SRow,ECol,ERow));
  Brush.Color := clRed;
  FrameRect(Rect(DestRect.Left - 1, DestRect.Top - 1, DestRect.Right + 1, DestRect.Bottom + 1));
 end;

//========================================================== Draw Lower

  SCol := ColLineArray[BlockCol];
  SRow := BaseLineArray[LineNo] - Round(FontHigh*1.8);

  SCol := SCol + MemLower.Left;
  ECol := SCol + MemLower.Right;
  SRow := SRow + MemLower.Top;
  ERow := SRow + MemLower.Bottom;

  with DestRect do
  begin
```

```
  Left := 0;
  Top := 0;

  Right := MulX*(ECol - SCol);
  Bottom := MulX*(ERow - SRow);

  Left := (ImageLower.Width - Right) Div 2;
  Right := Left + Right;
  Top := (ImageLower.Height - Bottom) Div 2;
  Bottom := Top + Bottom;
 end;

 with ImageLower.Canvas do
 begin
  Pen.Color := clRed;
  Brush.Color := clWhite;
  FillRect(ImageLower.BoundsRect);
  if DisplayText[2] in ['','',''] then
  begin
   CopyRect(DestRect,ScanImage.Canvas,Rect(SCol,SRow,ECol,ERow));
   Brush.Color := clRed;
   FrameRect(Rect(DestRect.Left - 1, DestRect.Top - 1, DestRect.Right + 1, DestRect.Bottom + 1));
  end;
 end;

//=========================================================== Draw Upper 2

  SCol := ColLineArray[BlockCol];
  SRow := BaseLineArray[LineNo] - Round(FontHigh*1.8);

  SCol := SCol - Round(FontHigh*0.1) + MemUpper2.Left;// - ShiftPrev;
  ECol := SCol + MemUpper2.Right;
  SRow := SRow + MemUpper2.Top;
  ERow := SRow + MemUpper2.Bottom;
```

```
 with DestRect do
 begin
  Left := 0;
  Top := 0;

  Right := MulX*(ECol - SCol);
  Bottom := MulX*(ERow - SRow);

  Left := (ImageUpper2.Width - Right) Div 2;
  Right := Left + Right;
  Top := (ImageUpper2.Height - Bottom) Div 2;
  Bottom := Top + Bottom;
 end;

 with ImageUpper2.Canvas do
 begin
  Pen.Color := clRed;
  Brush.Color := clWhite;
  FillRect(ImageUpper2.BoundsRect);
  if (Length(DisplayText) > 2) or ((Length(DisplayText) > 1) and not(DisplayText[2] in ['ั','ิ','ี','ึ'])) or
    HaveSpecialC then
  begin
   CopyRect(DestRect,ScanImage.Canvas,Rect(SCol,SRow,ECol,ERow));
   Brush.Color := clRed;
   FrameRect(Rect(DestRect.Left - 1, DestRect.Top - 1, DestRect.Right + 1, DestRect.Bottom + 1));
  end;
 end;

//=========================================================== Draw Upper 1

 SCol := ColLineArray[BlockCol];
 SRow := BaseLineArray[LineNo] - Round(FontHigh*1.8);
```

```
  SCol := SCol - Round(FontHigh*0.1) + MemUpper1.Left;
  ECol := SCol + MemUpper1.Right;
  SRow := SRow + MemUpper1.Top;
  ERow := SRow + MemUpper1.Bottom;

  with DestRect do
  begin
   Left := 0;
   Top := 0;

   Right := MulX*(ECol - SCol);
   Bottom := MulX*(ERow - SRow);

   Left := (ImageUpper1.Width - Right) Div 2;
   Right := Left + Right;
   Top := (ImageUpper1.Height - Bottom) Div 2;
   Bottom := Top + Bottom;
  end;

  with ImageUpper1.Canvas do
  begin
   Pen.Color := clRed;
   Brush.Color := clWhite;
   FillRect(ImageUpper1.BoundsRect);
   if (Length(DisplayText) > 3) or ((Length(DisplayText) > 2) and not(DisplayText[2] in ['ุ','ู'])) then
   begin
    CopyRect(DestRect,ScanImage.Canvas,Rect(SCol,SRow,ECol,ERow));
    Brush.Color := clRed;
    FrameRect(Rect(DestRect.Left - 1, DestRect.Top - 1, DestRect.Right + 1, DestRect.Bottom + 1));
   end;
  end;

//=================================================
```

```
  TextBlock.Caption := DisplayText;
  Screen.Cursor := crDefault;
end;

// แสดงการตัดตัวอักษรเมื่อมีการเปลี่ยนบรรทัด
procedure TForm1.DisplayTextLineChange(Sender: TObject);
var Col : longint;
begin
  DisplayTextCol.Clear;
  if DisplayTextLine.ItemIndex < 0 then Exit;
  ColumnScanLine.ItemIndex := DisplayTextLine.ItemIndex;
  ScanColumn(StrToInt(DisplayTextLine.Text) - 1);

  for Col := 0 to ColCount - 1 do
    DisplayTextCol.Items.Add(IntToStr(Col + 1));

  if DisplayTextCol.Items.Count > 0 then
    DisplayTextCol.ItemIndex := 0;

  if DisplayTextCol.ItemIndex >= 0 then
    DisplayBlockScan(StrToInt(DisplayTextCol.Text) - 1);
end;

// แสดงการตัดตัวอักษรเมื่อมีการเปลี่ยน column
procedure TForm1.DisplayTextColChange(Sender: TObject);
begin
  if DisplayTextCol.ItemIndex >= 0 then
    DisplayBlockScan(StrToInt(DisplayTextCol.Text) - 1);
end;

// แสดงการตัดตัวอักษรเมื่อมีการเปลี่ยน column ก่อนหน้านี้
procedure TForm1.PrevColBtnClick(Sender: TObject);
begin
  if DisplayTextCol.ItemIndex > 0 then
```

```
 begin
   DisplayTextCol.ItemIndex := DisplayTextCol.ItemIndex - 1;
   DisplayTextColChange(nil);
 end;
end;

// แสดงการตัดตัวอักษรเมื่อมีการเปลี่ยน column ถัดไป
procedure TForm1.NextColBtnClick(Sender: TObject);
begin
 if DisplayTextCol.ItemIndex < DisplayTextCol.Items.Count - 1 then
 begin
   DisplayTextCol.ItemIndex := DisplayTextCol.ItemIndex + 1;
   DisplayTextColChange(nil);
 end;
end;

// คำนวณหาความสูงของตัวอักษรโดยอัตโนมัติ
procedure TForm1.AutoSizeBtnClick(Sender: TObject);
var i,HitT : longint;
   Col,ECol : longint;
   Row : longint;
   NextVal,PrevVal : longint;
   DiffVal : longint;
   SkipNext : boolean;
   FontHeight : integer;
   Count,StaticCount : integer;
   LastHeight : integer;
   LineHeight : integer;
   BaseL,TopL : longint;

 function  FindFontSize: integer; // คำนวณความสูงของตัวอักษร
 var Found : boolean;
    R,C : longint;
 begin
```

```
  BaseL := 0;
  TopL := 0;
  Found := False;
  for R := PttImage.Height - 1 downto 0 do // หาขอบล่าง
  begin
   for C := 0 to PttImage.Width - 1 do
    if PttImage.Canvas.Pixels[C,R] <> PttBgColor then
    begin
     BaseL := R + 1;
     Found := True;
     break;
    end;
   if Found then break;
  end;

  Found := False;
  for R := 0 to PttImage.Height - 1 do // หาขอบบน
  begin
   for C := 0 to PttImage.Width - 1 do
    if PttImage.Canvas.Pixels[C,R] <> PttBgColor then
    begin
     TopL := R + 1;
     Found := True;
     break;
    end;
   if Found then break;
  end;
  Result := BaseL - TopL + 1; // หาความสูง
 end;

begin
 Screen.Cursor := crHourGlass;
 HistoryHit.Clear;
 HistDifVal.Clear;
```

```
HistFontHeight.Clear;
HistLineHeight.Clear;

ECol := ScanImage.Width - 1;
Row := 0;
with ScanImage.Picture.Bitmap.Canvas do
begin
 // scan หาการ Hit ของแต่ละบรรทัดแล้วเก็บค่าไว้ใน List box
 repeat
  HitT := 0;
  for Col := 0 to ECol do
   if Pixels[Col,Row] = PttFontColor then
    HitT := HitT + 1;

  Application.ProcessMessages;

  HistoryHit.Items.Add(IntToStr(HitT));

  Row := Row + 1;
 until Row > ScanImage.Height - 1;

 SkipNext := False;
 FontHeight := 1;
 LineHeight := 1;
 for Row := 1 to HistoryHit.Items.Count - 2 do
 begin
  if not(SkipNext) then
  begin
   NextVal := StrToInt(HistoryHit.Items[Row + 1]);
   PrevVal := StrToInt(HistoryHit.Items[Row - 1]);
   DiffVal := (NextVal - PrevVal) Div 2;  // ทำการหาค่าความชันของแต่ละแถวที่ scan ได้จากขั้นตอน
แรก
   HistDifVal.Items.Add(IntToStr(DiffVal)); // เก็บค่าความชันลง List box
```

```
      if DiffVal > 40 then // ถ้าความชันมากกว่า 40 แสดงว่าเป็นบรรทัดด้านบน
      begin
{       Pen.Color := clBlue;  // Display Line
       MoveTo(Left,Row);
       LineTo(Width,Row); }
       SkipNext := True;
      HistLineHeight.Items.Add(IntToStr(LineHeight)); // เก็บค่าความสูงของบรรทัดในบรรทัดนั้นลง
List box
       FontHeight := 1;
       LineHeight := 1;
      end
      else if DiffVal < -40 then // ถ้าความชันน้อยกว่า -40 แสดงว่าเป็นบรรทัดด้านล่าง
      begin
{       Pen.Color := clRed;  // Display Line
       MoveTo(Left,Row);
       LineTo(Width,Row); }
       SkipNext := True;
      HistFontHeight.Items.Add(IntToStr(FontHeight)); // เก็บค่าความสูงของ Font ในบรรทัดนั้นลง
List box
      end;
      FontHeight := FontHeight + 1;
      LineHeight := LineHeight + 1;
    end
    else SkipNext := False;
  end;

  FontHeight := 0;
  LastHeight := 0;
  Count := 0;
  StaticCount := 0;
  for Row := 0 to HistFontHeight.Items.Count - 1 do // คำนวณหาความสูงของ Font จาก List box ที่เก็บ
ไว้
  begin
    if LastHeight <> StrToInt(HistFontHeight.Items[Row]) then
```

```
    begin
      Count := 0;
      LastHeight := StrToInt(HistFontHeight.Items[Row]);
    end
    else Count := Count + 1;

    if StaticCount < Count then
    begin
      StaticCount := Count;
      FontHeight := StrToInt(HistFontHeight.Items[Row]);
    end;
  end;

  LineHeight := 0;
  LastHeight := 0;
  Count := 0;
  StaticCount := 0;
  for Row := 0 to HistLineHeight.Items.Count - 1 do // คำนวณหาความสูงของบรรทัดจาก List box ที่
เก็บไว้
  begin
    if LastHeight <> StrToInt(HistLineHeight.Items[Row]) then
    begin
      Count := 0;
      LastHeight := StrToInt(HistLineHeight.Items[Row]);
    end
    else Count := Count + 1;

    if StaticCount < Count then
    begin
      StaticCount := Count;
      LineHeight := StrToInt(HistLineHeight.Items[Row]);
    end;
  end;
end;
```

```
 PttImage.Canvas.Font := DisplayFont.Font;
 for i := 10 to 24 do // ไล่คำนวณหาขนาดของตัวอักษรจากความสูงของตัวอักษรที่ได้มา
 begin
  PttImage.Canvas.Brush.Color := clWhite;
  PttImage.Canvas.FillRect(Rect(0,0,PttImage.Width,PttImage.Height));
  PttImage.Canvas.Font.Size:= i;
  PttImage.Canvas.TextOut(5,5,'ก');

  if FindFontSize = FontHeight then
  begin
   FontSizeL.Caption := IntToStr(i);

   // Find English Font Height
   PttImage.Canvas.Brush.Color := clWhite;
   PttImage.Canvas.FillRect(Rect(0,0,PttImage.Width,PttImage.Height));
   PttImage.Canvas.Font.Size:= i;
   PttImage.Canvas.TextOut(5,5,'A');
   FontHeightL.Caption := IntToStr(FindFontSize + 1);

   LineHeightL.Caption := IntToStr(LineHeight);

   if AutoSet.Checked then // กำหนดค่าที่ได้ไว้สำหรับโหลด Pattern
   begin
    FontSizeSp.Value := StrToInt(FontSizeL.Caption);
    FontSizeSpChange(nil);
    FontHighSp.Value := StrToInt(FontHeightL.Caption);
    LineHighSp.Value := StrToInt(LineHeightL.Caption);
   end;
   break;
  end;
 end;

 Screen.Cursor := crDefault;
```

```
end;

end.
```

บรรณานุกรม

J.R. Parker , " **ALGORITHMS FOR IMAGE PROCESSING AND COMPUTER VISION**" , WILEY COMPUTER PUBLISHING, 1997.

ชาญชัย ดีอ่วม, " **การรู้จำตัวอักษรคัดลายมือภาษาไทยของคอมพิวเตอร์โดยวิเคราะห์โครงสร้างแบบทัน ทีทันใด**",วิทยานิพนธ์คณะวิทยาการคอมพิวเตอร์และเทคโนโลยีสาร สนเทศ,2542.

กฤษฎา ลิมปานันท์ และ ดร.โกสินทร์ จำนงไทย, **"วิธีการรู้จำลายมือเขียนภาษาไทยโดยใช้กรอบมาตราฐาน"**,การประชุมวิชาการวิศวกรรมไฟฟ้า ครั้งที่ 20 ,2540

Daniel B Hentchel, **"Creation of a Neural Network to Assist in Deciphering Degraded Anceint Hebraw Texts"** , http://www.cis.rit.edu/~dbh6913/Thesis/contents.html

Rafael C. Gonzalez & Richard E. Woods, "Digital Image Processing",Addison Wesley,1993.

ชม กิ้มปาน และ สมศักดิ์ วลัยรัชต์, " **การรู้จำตัวอักษรลายมือเขียนภาษาไทยของไมโครคอมพิวเตอร์ โดยการพิจารณาลักษณะลายเส้นขณะที่ลากไป**", การประชุมวิชาการทางวิศวกรรมไฟฟ้าครั้งที่ 13 ,คณะวิศวกรรมศาตร์ มหาวิทยาลัยเชียงใหม่, 8-9 พฤศจิกายน,2533

มนัส สังวรศิลป์ , พิชัย ดุศิริวานิชกร , สุรพันธ์ เอื้อไพบูลย์ และ เชวง ตันติยาภรณ์ , " **การจดจำตัวอักษรลายเขียนภาษาไทยโดยการพิจารณาหัวของตัวอักษร**" , การประชุมวิชาการวิศวกรรมไฟฟ้าครั้งที่ 11 , 2531

Rafael C. Gonzalez and Richard E. Woods, **"Digital image processing"**, Addison-Wesly Publishing Company , 1992.