การวิเคราะห์แยกตัวอักษรบนหน้าเอกสาร

(CHARCTER SEGMENTATION ON DOCUMENT)

โดย

นายกมนต์ ลาภสอาดพร

นายสุวิทย์ วงศ์คุ้มสิน

อาจารย์ที่ปรึกษา

รองศาสตราจารย์ ดร.ชม กิ้มปาน

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

ปีการศึกษา 2538

การวิเคราะห์แยกตัวอักษรบนหน้าเอกสาร

(CHARCTER SEGMENTATION ON DOCUMENT)

โดย

นายกมนต์ ลาภสอาดพร

นายสุวิทย์ วงศ์คุ้มสิน

อาจารย์ที่ปรึกษา

รองศาสตราจารย์ ดร.ชม กิ้มปาน

ปริญญานิพนธ์ปีการศึกษา 2538

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การวิเคราะห์แยกตัวอักษรบนหน้าเอกสาร

(CHARCTER SEGMENTATION ON DOCUMENT)

ผู้จัดทำ

1. นายคมนต์ ลาภสอาดพร 36013141
2. นายสุวิทย์ วงศ์คุ้มสิน 36013183

---------- อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร. ชม กิ้มปาน)

การวิเคราะห์แยกตัวอักษรบนหน้าเอกสาร

(CHARCTER SEGMENTATION ON DOCUMENT)

นายกมนต์ ลาภสอาดพร
นายสุวิทย์ วงศ์คุ้มสิน

รองศาสตราจารย์ ดร. ชม กิ้มปาน
อาจารที่ปรึกษา
ปีการศึกษา 2538

บทคัดย่อ

ปริญญานิพนธ์นี้ได้เสนอถึง การวิเคราะห์แยกตัวอักษรบนหน้าเอกสาร โดยใช้ไมโครคอมพิวเตอร์ ซึ่งเป็นการให้เครื่องไมโครคอมพิวเตอร์สามารถที่จะจำแนกว่าเป็นรูปภาพหรือตัวอักษร และทำการแยกตัวอักษรทีละตัวออกจากประโยค โดยการป้อนข้อมูล (หน้าเอกสาร) ผ่านเครื่องตรวจกวาดภาพ (Image Scanner) แล้วทำการส่งข้อมูลที่ได้นั้นส่งให้แก่เครื่องไมโครคอมพิวเตอร์ เพื่อทำกระบวนการแยกตัวอักษรและรูปภาพออกจากหน้าเอกสาร จากการศึกษาและทำการทดลอง ผลที่ได้มีการแยกรูปภาพและตัวอักษรได้ถูกต้อง

Abstract

This project proposes " Character segmentation on document for Microcomputers ". The algorithm consists of the several routines witch is finally resulting successfully characters segmentation. Characters and picture written on the papers, are extracted by scanner and become input for segmentation. The identification is devided picture and characters. The experiment had been conducted.

สารบัญ

สารบัญรูปประกอบ

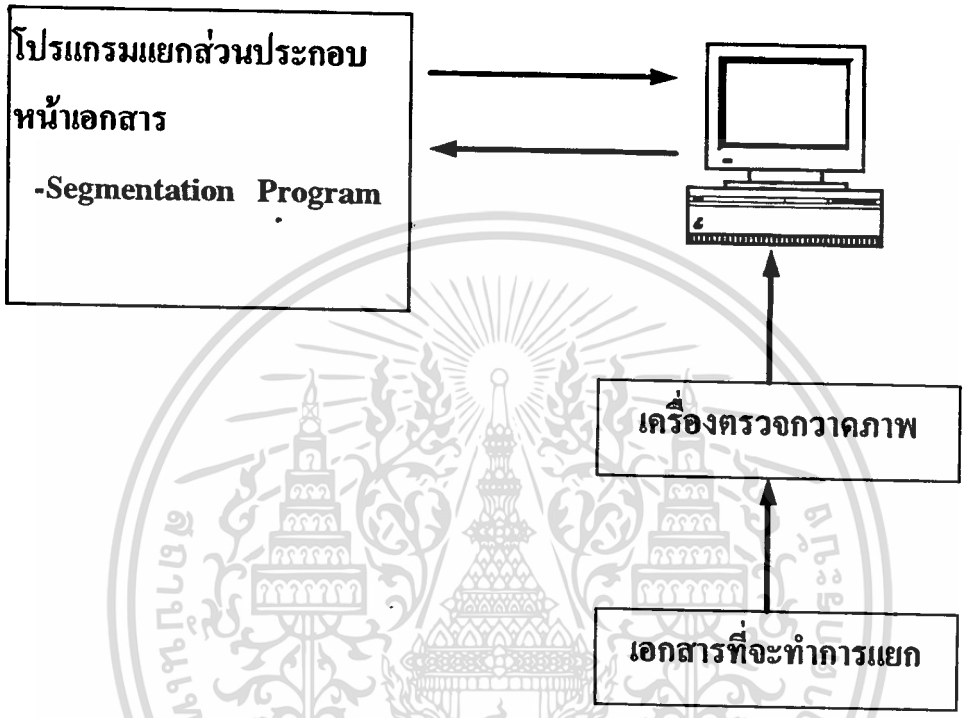
สารบัญรูปประกอบ (ต่อ)

สารบัญตารางประกอบ

บทที่ 1
บทนำ

ในปัจจุบันนี้ความก้าวหน้าทางด้านเทคโนโลยีต่างๆ ก้าวหน้าไปอย่างรวดเร็วมาก ซึ่งเทคโนโลยีทางด้านคอมพิวเตอร์ก็ได้มีการเจริญเติบโตมากขึ้นด้วย ซึ่งทุกวันนี้คอมพิวเตอร์ได้เข้ามามีบทบาทที่สำคัญในหลายๆ เรื่อง งานหลายๆ อย่างก็ได้มีการนำคอมพิวเตอร์เข้ามาช่วยในการทำงาน เช่น งานที่มีการกระทำซ้ำๆกับหลายๆครั้งหรืองานที่มีความซับซ้อน ก็สามารถใช้คอมพิวเตอร์เข้ามาช่วยงานเหล่านี้ได้ เพื่อให้การทำงานมีความรวดเร็วและมีประสิทธิภาพสูงขึ้น และได้มีการพัฒนาให้คอมพิวเตอร์สามารถที่จะเรียนรู้เรื่องต่างๆได้มากมาย โดยพยายามที่จะทำให้เครื่องไมโครคอมพิวเตอร์สามารถเลียนแบบลักษณะการทำงานของมนุษย์ให้มากที่สุด ในส่วนของปริญญานิพนธ์ฉบับนี้ก็เป็นอีกลักษณะหนึ่งที่จะพยายามให้การทำงานของเครื่องไมโครคอมพิวเตอร์มีการทำงานเลียนแบบการทำงานของมนุษย์ โดยให้คอมพิวเตอร์มีความสามารถที่จะทำการแยกตัวอักษรและรูปภาพได้ ซึ่งความสามารถแยกสิ่งต่างๆ ก็เป็นคุณสมบัติประการหนึ่งของมนุษย์ และลักษณะเดียวกันกับระบบสิ่งมีชีวิตอื่นๆ รูปแบบจะเป็นลักษณะประจำของแต่ละวัตถุ ซึ่งเราจะต้องทำการแยกสิ่งต่างๆ ทุกๆ อย่างในชีวิตประจำวัน การแยกแยะของคนเราเป็นระบบความรู้ที่มีการเปลี่ยนแปลงตามเงื่อนไงที่ทันสมัยมาก ทั้งนี้ก็เนื่องจากการแยกสิ่งต่างๆ ของมนุษย์เป็นขบวนการที่ซับซ้อนมาก แต่อย่างไรก็ตามเราก็พยายามจะหาวิธีการต่างๆ เพื่อที่จะสอนให้คอมพิวเตอร์เรียนรู้กระบวนการดังกล่าว ซึ่งสามารถที่จะสรุปกระบวนการต่างๆได้ว่าเป็นการแยกตัวอักษรจากข้อความหรือเป็นการแยกรูปภาพ โดยเริ่มตั้งแต่มีตัวอักษรหรือข้อความและรูปภาพที่อยู่บนหน้ากระดาษเข้ามา และเริ่มรับรู้ส่วนต่างๆบนหน้าเอกสารด้วยการมองเห็นด้วยการใช้สายตา เพื่อเก็บลักษณะโครงสร้างของตัวอักษรหรือรูปภาพเหล่านั้น แล้วจึงทำการแยกว่าเป็นตัวอักษรหรือเป็นรูปภาพ ซึ่งการเรียนรู้ลักษณะดังกล่าวของมนุษย์ สามารถที่จะนำมาประยุกต์โดยใช้อุปกรณ์ทางด้านอิเลกทรอนิค เพื่อเป็นการแบ่งเบาภาระการทำงานของมนุษย์ลง ในรูปแบบของการทำงานแทนได้ โดยในส่วนของการมองเห็นด้วยตาเพื่อทำการเก็บลักษณะโครงสร้างของตัวอักษรหรือรูปภาพต่างๆ นั้น สามารถที่จะแทนได้ด้วยเครื่องตรวจกวาดภาพ (Image Scanner) ซึ่งมีความสามารถในการเก็บสัญญาณแสงที่ได้จากการสะท้อนกลับของการเปล่งแสงไปยังอักษรหรือภาพต่างๆ เหล่านั้น และแปลงสัญญาณแสงที่ได้ให้เป็นสัญญาณไฟฟ้าที่จะทำการเก็บส่วนประกอบต่างๆ ไว้ และในส่วนของการแยกตัวอักษรหรือรูปภาพก็จะใช้ส่วนของการประมวลผลรวมกับโปรแกรม ซึ่งเปรียบเสมือนกับเป็นส่วนสมองของมนุษย์ จากวิธีการและขั้นตอนใน

การทำงาน สามารถที่จะเขียนแสดงการทำแทนการมองเห็นและการแยกส่วนต่างๆ ของมนุษย์ในรูปแบบของกระบวนการทางเครื่องมืออิเลกทรอนิค ดังรูปที่ 1.1



รูปที่ 1.1 ภาพประกอบการแยกส่วนประกอบบนหน้าเอกสาร

ในส่วนของการทำงานวิจัยฉบับนี้ จะเป็นการวิเคราะห์แยกส่วนประกอบบนบหน้าเอกสาร ซึ่งจะเป็นแนวทางในการพัฒนาอีกรูปแบบหนึ่งของคอมพิวเตอร์ ที่จะให้มีการเลียนแบบการทำงานของมนุษย์ และสามารถนำไปประยุกต์ในการใช้งาน โดยเป็นส่วนหนึ่งของการป้อนข้อมูลอัตโนมัติ (Automatic data entry) ซึ่งในปริญญานิพนธ์ฉบับนี้จะทำการวิเคราะห์แยกตัวอักษรทั้งที่เป็นภาษาไทย ภาษาอังกฤษ และที่เป็นรูปภาพ ซึ่งจะมีรายละเอียดของเนื้อหาในบทต่อๆ ไป

ลักษณะและความสำคัญของปัญหา

เนื่องจากว่าในระบบจดจำตัวอักษรนั้น จำเป็นต้องมีการวิเคราะห์ส่วนประกอบของหน้าเอกสารโดยการแยกภาพออกจากหน้าเอกสาร และยังต้องการให้แยกตัวอักษรออกจาก

ประโยคที่ละตัว เพื่อที่จะทำการวิเคราะห์และจดจำตัวอักษร ปัญหาที่เกิดขึ้นคือ ในการวิเคราะห์และจดจำตัวอักษรนั้น ข้อมูลที่ต้องการยังไม่ได้อยู่ในรูปแบบที่ต้องการ นั่นคือภาพและตัวอักษรที่ยังไม่ได้แยกออกจากกันนั่นเอง

ส่วนประกอบของหน้าเอกสาร จึงได้กลายมาเป็นอุปสรรคหนึ่งของระบบจดจำตัวอักษรตัวอย่างเช่น ในกรณีของหน้าเอกสารหนึ่งหน้ากระดาษนั้น อาจจะไม่ได้มีตัวอักษรล้วนๆ พิมพ์อยู่อย่างเดียว แต่อาจจะประกอบไปด้วยข้อความ รูปภาพประกอบข้อความ และภาพสัญลักษณ์ต่างๆ ที่พิมพ์อยู่รวมกัน ซึ่งถ้าไม่ได้มีการแยกแยะส่วนประกอบต่างๆ ของหน้าเอกสารให้ถูกต้องแล้ว อาจจะทำให้การเรียนรู้และจดจำตัวอักษรมีความผิดพลาด หรือทำไม่ได้เลย เช่นในกรณีที่ข้อมูลที่ส่งผ่านให้กับระบบการเรียนรู้และจดจำตัวอักษร กลายเป็นภาพไม่ใช่ตัวอักษรจะเพราะฉะนั้นขบวนการวิเคราะห์และระบุส่วนประกอบของหน้าเอกสาร จึงเป็นขั้นตอนที่สำคัญเพื่อการเตรียมข้อมูลให้กับระบบการเรียนรู้และจดจำตัวอักษรทำงานเป็นไปอยางถูกต้อง

ส่วนมากในระบบการเรียนรู้และจดจำตัวอักษรนั้นยังไม่มีโปรแกรมสำเร็จรูปใดที่ทำการวิเคราะห์ส่วนประกอบของหน้าเอกสาร ถ้าต้องการที่จะใช้ก็จะต้องทำการเขียนโปรแกรมในส่วนนี้ขึ้นมาเอง ซึ่งทำไหเกิดความยุ่งยาก และล่าช้าในการที่จะเขียนระบบการเรียนรู้และจดจำตัวอักษร ถ้ามีโปรแกรมที่ทำการวิเคราะห์หน้าเอกสาร จะทำให้ระบบการเรียนรู้และจดจำตัวอักษรมีความสะดวกมากยิ่งขึ้น ที่ไม่ต้องมาทำการจำแนกส่วนประกอบหน้าเอกสารเอง

ในโครงงานนี้ได้ทำการวิเคราะห์ส่วนประกอบของหน้าเอกสารขึ้นมาให้แก่ระบบจดจำตัวอักษร โดยจะทำการแยกภาพออกจากหน้าเอกสาร และแยกตัวอัดษรออกจากประโยคในหน้าเอกสารออกมาทีละตัว โดยตัวอักษรที่แยกออกมาได้นั้นจะนำมาทำการเรียงข้อมูลให้ถูกต้องเพื่อที่พร้อมที่จะส่งไหกับระบบจดจำตัวอักษรต่อไป

บทที่ 2
วัตถุประสงค์ ทฤษฎีและหลักการที่นำมาใช้

2.1 บทนำ

การวิเคราะห์แยกตัวอักษรบนหน้าเอกสาร เป็นขบวนการทำงานหนึ่งในระบบการรู้จำตัวอักษร (Character recognition) ซึ่งส่วนประกอบต่างๆ บนหน้าเอกสารได้กลายเป็นอุปสรรคหนึ่งของระบบการรู้จำตัวอักษร ดังเช่น ในเอกสารหนึ่งหน้ากระดาษ โดยทั่วไปแล้วจะเป็นกระดาษขนาด A4 ซึ่งอาจจะประกอบไปด้วยข้อความที่เป็นตัวอักษร ซึ่งถ้าพิจารณาลักษณะโครงร่าง (Layout) ของหน้าเอกสาร จะพบว่าหน้าเอกสารมีลักษณะเป็นเส้นทึบเรียงกันอยู่ในแนวนอนและลักษณะเส้นทึบเป็นกลุ่ม ซึ่งเส้นทึบเหล่านั้นก็คือภาพของตัวอักษรที่พิมพ์เรียงกันเป็นแถวเป็นบรรทัดนั่นเอง ซึ่งภาพของตัวอักษรที่เป็นเส้นทึบๆ นั้นจะมีช่องว่างทางแนวนอนก็คือ ช่องว่างระหว่างบรรทัด และช่องว่างทางแนวตั้งก็คือ ช่องว่างระหว่างคอลัมน์ เป็นสิ่งที่ใช้แยกข้อความแต่ละบรรทัดและแต่ละข้อความออกจากกัน เมื่อนำเอาลักษณะโครงร่างของหน้าเอกสารดังกล่าวนี้มาพิจารณา ก็สามารถใช้ช่องว่างในแนวนอนและแนวตั้งนี้มาแบ่งแยกบริเวณของข้อความที่พิมพ์อยู่บนหน้าเอกสารออกมาได้ และส่งกลุ่มของข้อความที่ได้นี้ไปเข้าขบวนการแยกภาพตัวอักษรออกทีละตัวต่อไป

สำหรับกรณีของปัญหาที่ซับซ้อนมากกว่านี้คือ บนหน้าเอกสารจะไม่ได้ประกอบไปด้วยข้อความของตัวอักษรเพียงอย่างเดียว แต่จะประกอบไปด้วยข้อความและรูปภาพซึ่งอาจจะเป็นภาพสัญญลักณ์ หรือภาพโลโกพิมพ์รวมอยู่ด้วย ซึ่งในกรณีนี้ถ้าไม่มีการแยกส่วนประกอบต่างๆ บนหน้าเอกสารว่าเป็นรูปภาพหรือภาพตัวอักษรแล้วจะทำให้กระบวนการรู้จำตัวอักษรเกิดความผิดพลาดมากยิ่งขึ้น เพราะฉนั้นจะต้องทำการแยกรูปภาพที่อยู่บนหน้าเอกสารออกไป ก่อนที่จะทำการส่งเข้าขบวนการรู้จำตัวอักษรในขั้นต่อไป

2.2 วัตถุประสงค์

จุดมุ่งหมายของปริญญานิพนธ์นี้ก็คือ ต้องการจัดแยกตัวอักษรเพื่อที่จะนำเอาตัวอักษรที่ได้นี้เข้าสู่ขบวนการรู้จำตัวอักษรต่อไป ซึ่งจะเป็นการพัฒนาให้คอมพิวเตอร์มีการทำงานที่มีความชาญฉลาดมากยิ่งขึ้น การป้อนข้อมูลจากหน้าเอกสารเข้าเครื่องคอมพิวเตอร์ก็จะทำได้

สะดวกขึ้น จากกระบวนการแยกตัวอักษรที่จัดทำขึ้นนี้ สามารถที่จะนำไปประยุกต์ใช้ให้เป็นประโยชน์กับงานอื่นๆ ได้

2.3 ทฤษฎีที่นำมาใช้

2.3.1 ทฤษฎีเซลลูล่าออโตมาตา (Cellular Automata)

เทคนิคที่นำมาประยุกต์ใช้วิเคราะห์และระบุส่วนประกอบของหน้าเอกสารเป็นเทคนิคใหม่โดยอาศัยทฤษฎีเซลลูล่าออโตมาตา (Cellular Automata) ซึ่งเป็นทฤษฎีที่ว่าด้วยการแก้ปัญหาชนิดหนึ่งอาศัยหลักการที่กล่าวว่า การแก้ปัญหาที่ไม่ว่าจะเป็นเรื่องใดก็ตาม จะยึดหลักการแบ่งปัญหาออกเป็นส่วนย่อยๆ แล้วค่อยจัดการกับส่วนย่อยๆ เหล่านั้นก่อน เมื่อจัดการกับองค์ประกอบย่อยๆ เหล่านั้นได้ก็จะทำให้สามารถจัดการกับปัญหาใหญ่ทั้งหมดได้

ส่วนย่อยๆ ที่ถูกแบ่งแยกออกจะเรียกอีกอย่างว่า เซล (Cell) การจัดการกับแต่ละเซลแล้วจะสามารถจัดการกับระบบรวมหรือปัญหาใหญ่ได้ทั้งหมดนั้น มาจากความจริงที่ว่าปัญหาใหญ่ที่สามารถแบ่งแยกย่อยออกไปได้นั้น จะเกี่ยวข้องกับปัญหาย่อยหรือเซลที่ถูกแยกออกมา และเมื่อไล่แก้ปัญหาย่อย ก็เหมือนกับแก้ปัญหาใหญ่ทีละเล็กละน้อยจนในที่สุดเมื่อแก้ปัญหาย่อยได้หมดก็เป็นการแก้ปัญหาใหญ่ได้ทั้งหมดด้วย

การแก้ปัญหาส่วนย่อยหรือเซล นอกจากจะใช้สถานะหรือรายละเอียดที่ได้จากภายในเซลเองแล้วก็ยังสามารถใช้สถานะของเซลอื่นๆ ที่อยู่รอบๆ เซลนั้นมาใช้ช่วยในการแก้ปัญหาได้ด้วย ซึ่งสุดท้ายจะนำไปสู่การแก้ปัญหาทั้งหมด

การแก้ปัญหาของทฤษฎีเซลลูล่าออโตมาตายึดหลักใหญ่ๆ อยู่ 2 ข้อคือ

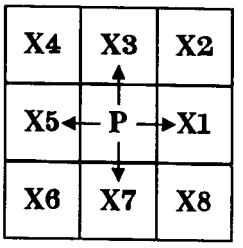
1. แยกสิ่งที่สนใจและต้องการแก้ไขหรือหาคำตอบ ออกเป็นส่วนย่อยๆ หรือเซล
2. แก้ไขปัญหาส่วนย่อยๆทีละส่วนให้หมด โดยอาศัยข้อมูลจากภายในส่วนย่อยและข้อมูลจากส่วนย่อยที่อยู่รอบๆ นั้นด้วย

2.3.2 ความสัมพันธ์ระหว่างจุดภาพข้างเคียง (Neighbourhood pixel)

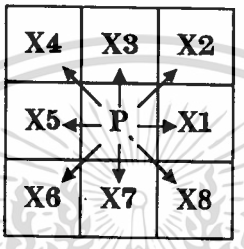
จุดภาพข้างเคียงบนรูปแบบสองระดับใดๆ จะใช้ตารางแบบ 3X3 เป็นแบบในการพิจารณา ซึ่งจะแบ่งได้เป็น 2 แบบคือ

1. ภาพข้างเคียงแบบ 4 จุดภาพ (4-Neighbourhood pixel)
2. ภาพข้างเคียงแบบ 8 จุดภาพ (8-Neighbourhood pixel)

กำหนดให้ P เป็นจุดภาพใดๆ บนรูปแบบสองระดับ จุดภาพข้างเคียงของจุด P ทั้ง 2 แบบแสดงไว้ในรูปที่ 2.1 และรูปที่ 2.2 ตามลำดับ



รูปที่ 2.1 จุดข้างเคียง 4 จุด



รูปที่ 2.2 จุดข้างเคียง 8 จุด

2.3.3 การติดตามขอบของภาพ (Contour Following)

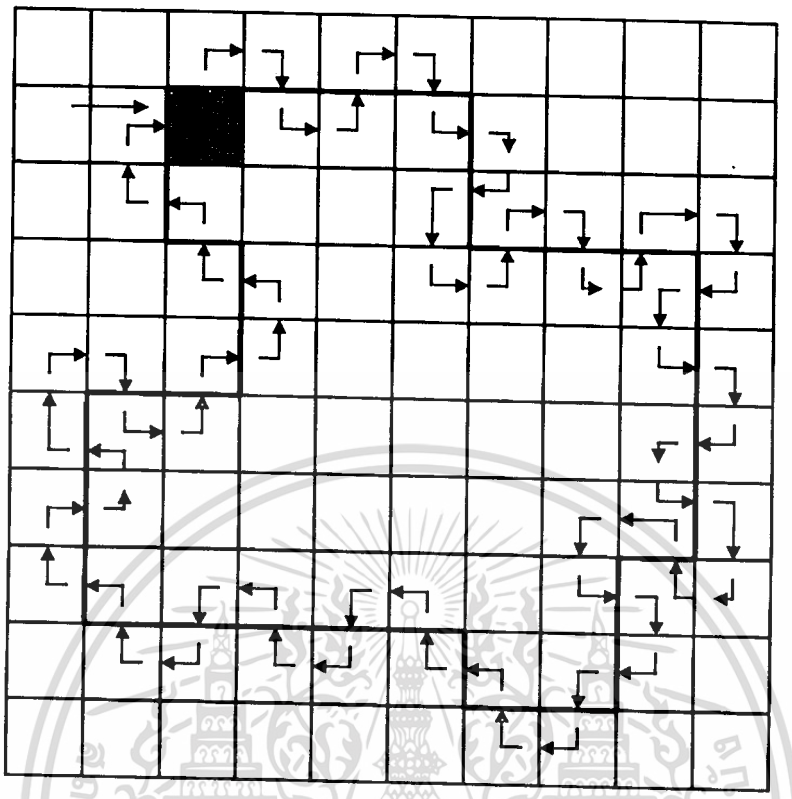
เทคนิคการติดตามรอยของขอบภาพ ถูกนำมาใช้ในการแยกและคัดลอกส่วนของรูปภาพใดๆ ที่อยู่บนรูปใหญ่ ข้อมูลภาพที่จะนำมาประมวลผลด้วยเทคนิคนี้จะต้องอยู่ในรูปของข้อมูลไบนารี่ นั่นคือจุดภาพจะแสดงด้วยตัวเลข 0 กับ 1 เท่านั้นโดยที่จุดภาพที่มีค่าเป็น 1 แทนจุดดำหรือจุดที่เป็นส่วนของรูปภาพ และจุดภาพที่มีค่าเป็น 0 แทนจุดขาวหรือจุดที่เป็นช่องว่างบนกระดาษหรือพื้นเบื้องหลัง

การทำงานของเทคนิคการติดตามรอยขอบของภาพ เป็นการเดินไต่ไปตามขอบระหว่างส่วนที่เป็นรูปภาพ (Image) กับส่วนที่เป็นพื้นเบื้องหลัง (Background) โดยจะตรวจกวาดไปทุกๆ จุดภาพ (Pixel) โดยจะเริ่มจากจุดมุมซ้ายบนของข้อมูลภาพ ตรวจกวาดไปในทิศทางจากซ้ายไปขวา และเลื่อนลงจากบนลงล่าง เมื่อตรวจกวาดมาพบจุดภาพใดๆ ที่มีค่าของจุดภาพเป็น 1 ก็จะเปลี่ยนลักษณะการเคลื่อนที่ไปยังจุดถัดไปเสียใหม่ โดยมีเงื่อนไขของการเคลื่อนที่ดังต่อไปนี้

1. ถ้าจุดที่อยู่ปัจจุบันเป็นจุดของภาพหรือมีค่าของจุดเป็น 1 ให้เลี้ยวซ้าย แล้วก้าวเดินตรงไปข้างหน้าไปยังจุดถัดไป

2. ถ้าจุดที่อยู่ปัจจุบันเป็นเป็นพื้นเบื้องหลังหรือมีค่าของจุดเป็น 0 ให้เลี้ยวขวา แล้วก้าวเดินตรงไปข้างหน้าไปยังจุดถัดไป

3. การเคลื่อนที่จะสิ้นสุดลง เมื่อจุดที่อยู่ปัจจุบันเป็นจุดเดียวกันกับจุดเริ่มต้น



รูปที่ 2.3 แสดงลักษณะการติดตามรอยขอบภาพ

2.4 การระบุส่วนประกอบหน้าเอกสาร

2.4.1 การกำหนดขอบเขต (Blocking zone)

การกำหนดขอบเขตจะใช้หลักการของความสัมพันธ์ระหว่างจุดภาพข้างเคียงมาใช้ ซึ่งการทำงานในขั้นตอนนี้จะเริ่มด้วยการจัดเตรียมข้อมูลภาพหน้าเอกสาร โดยการแปลงเป็นข้อมูลเป็นแบบไบนารี่ นั่นคือแต่ละจุดภาพจะถูกแทนด้วยตัวเลขฐานสองเท่านั้น คือตัวเลข 0 กับ 1 โดยที่จุดภาพที่มีค่าเป็น 0 จะแทนด้วยพื้นสีขาวหรือช่องว่างที่ไม่ใช่ตัวอักษร และจุดภาพที่มีค่าเป็น 1 จะแทนส่วนที่เป็นลายเส้นของตัวอักษรหรือลายเส้นของรูปภาพ โดยการกำหนดขอบเขตนี้จะต้องเตรียมข้อมูลที่เป็นไบนารี่ไว้ 2 ชุดเพื่อใช้ในการประกอบการวิเคราะห์

ข้อมูลชุดที่ 1 จุดภาพแต่ละจุดจะถูกกำหนดสถานะของจุดภาพ โดยพิจารณาจุดภาพที่ล้อมรอบจุดที่กำลังพิจารณา นับจำนวนจุดภาพรอบๆ ที่มีค่าเป็น 1 และพิจารณาลักษณะของจุดภาพรอบๆ ที่ปรากฏ ซึ่งจะถูกนำมาใช้เป็นเงื่อนไขในการกำหนดสถานะของจุด โดยจะมีเงื่อนไขอยู่ด้วยกัน 4 เงื่อนไขดังต่อไปนี้

1. ถ้าจุดภาพที่กำลังพิจารณาอยู่มีค่าเป็น 1 และจำนวนจุดภาพที่อยู่รอบๆ ข้างที่มีค่าเป็น 1 มีจำนวนน้อยกว่าจำนวนที่กำหนดไว้แล้ว จุดภาพที่กำลังพิจารณาอยู่จะถูกกำหนดให้เป็น จุดตาย เนื่องจากมีจำนวนจุดภาพรอบข้างที่เป็น 1 น้อยเกินไป

2. ถ้าจุดภาพที่กำลังพิจารณาอยู่มีค่าเป็น 1 และจำนวนจุดภาพที่อยู่รอบๆ ข้างที่มีค่าเป็น 1 มีจำนวนมากกว่าหรือเท่ากับจำนวนที่กำหนดไว้แล้ว จุดภาพที่กำลังพิจารณาอยู่จะถูกกำหนดให้เป็น จุดเป็น เนื่องจากมีจำนวนจุดภาพรอบข้างที่เป็น 1 ตามที่กำหนด

3. ถ้าจุดภาพที่กำลังพิจารณาอยู่มีค่าเป็น 0 และจำนวนจุดภาพที่อยู่รอบๆ ข้างที่มีค่าเป็น 1 มีจำนวนมากกว่าหรือเท่ากับจำนวนที่กำหนดไว้แล้ว จุดภาพที่กำลังพิจารณาอยู่จะถูกกำหนดให้เป็น จุดเกิด เนื่องจากมีจำนวนจุดภาพรอบข้างที่เป็น 1 ตามที่กำหนด

4. ถ้าจุดภาพที่กำลังพิจารณาอยู่มีค่าเป็น 0 และจำนวนจุดภาพที่อยู่รอบๆ ข้างที่มีค่าเป็น 1 มีจำนวนน้อยกว่าจำนวนที่กำหนดไว้แล้ว จุดภาพที่กำลังพิจารณาอยู่จะยังคงสถานะเดิมต่อไปและค่าของจุดภาพยังคงมีค่าเหมือนเดิมไม่เปลี่ยนแปลง

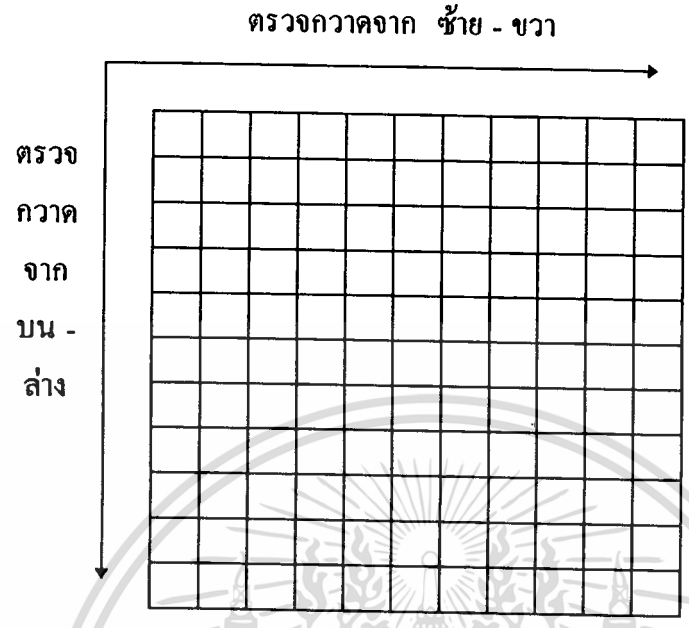
จากการพิจารณาจุดภาพตามที่กล่าวมาแล้ว และเมื่อทราบสถานะของจุดภาพจุดหนึ่งแล้ว จะทำการเปลี่ยนค่าของจุดภาพให้มีค่าที่สอดคล้องกับสถานะของจุดภาพนั้นๆ แต่การเปลี่ยนแปลงค่าของจุดภาพจะกระทำกับจุดภาพในข้อมูลชุดที่สอง ณ.ตำแหน่งพิกัดของจุดภาพเดียวกันกับพิกัดของจุดภาพในข้อมูลชุดที่ 1 ที่กระทำเช่นนี้ ก็เนื่องจากว่าจุดภาพถูกเปลี่ยนค่าตามสถานะใหม่แล้วจะได้ไม่ส่งผลกระทบต่อการพิารณากำหนดสถานะของจุดภาพจุดต่อไปในข้อมูลภาพต้นฉบับ สำหรับเงื่อนไขในการเปลี่ยนค่าของจุดภาพใหม่จะถูกกำหนดโดยเงื่อนไข 3 ลักษณะดังนี้

1. จุดตาย ซึ่งเดิมมีค่าเป็น 1 จะถูกเปลี่ยนให้มีค่าใหม่เป็น 0

2. จุดเป็น ซึ่งเดิมมีค่าเป็น 1 ยังคงให้มีค่าเป็น 1 ต่อไป

3. จุดเกิด ซึ่งเดิมมีค่าเป็น 0 จะถูกเปลี่ยนให้มีค่าใหม่เป็น 1

จากการพิจารณากำหนดสถานะของจุดภาพและเปลี่ยนค่าใหม่ให้กับจุดภาพจะกระทำกับทุกๆ จุดภาพในข้อมูลภาพ เริ่มต้นจากจุดภาพที่มุมบนซ้ายมือ และตรวจกวาดไปตามแถวในทิศทางจากซ้ายไปขวา และเลื่อนแถวลงมาในทิศทางจากบนลงล่าง

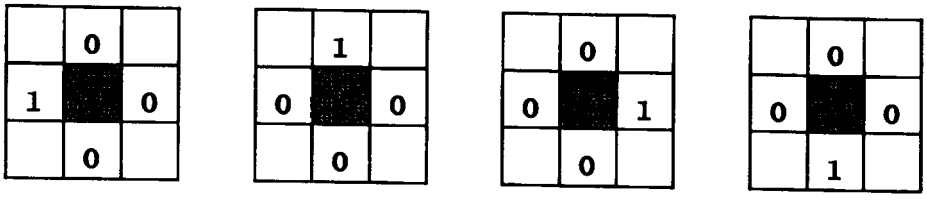


รูปที่ 2.4 ลักษณะการตรวจกวาดของการพิจารณาจุดภาพ

การประยุกต์ใช้งานทฤษฎีเซลลูล่าออโตมาตากับการกำหนดขอบเขตของส่วนประกอบของหน้าเอกสาร มีการปรับแต่งเงื่อนไขของการพิจารณาสถานะของจุดภาพบางประการให้เหมาะสมกับวัตถุประสงค์ของงานที่ทำ เนื่องจากจุดมุ่งหมายของการกำหนดขอบเขตของส่วนประกอบของหน้าเอกสารคือขอบเขตที่เป็นลักษณะสี่เหลี่ยมที่เป็นข้อความหรือขอบเขตสี่เหลี่ยมที่เป็นรูปภาพ ดังนั้นการพิจารณาจุดภาพที่ล้อมรอบจุดภาพที่กำลังพิจารณาอยู่ จะไม่ทำการพิจารณาจุดภาพในแนวเส้นทะแยงมุมทั้งสองเส้น จะพิจารณาเฉพาะจุดรอบข้างที่อยู่ทางด้านข้างซ้าย ด้านข้างขวา ด้านบน และด้านล่าง รวมทั้งหมด 4 จุดเท่านั้น ซึ่งจะช่วยในการค้นหาขอบเขตในการลักษณะสี่เหลี่ยมจะทำได้ง่ายขึ้น และยังช่วยลดการคำนวณได้อีกด้วย ซึ่งเป็นผลทำให้การทำงานมีความรวดเร็วขึ้น

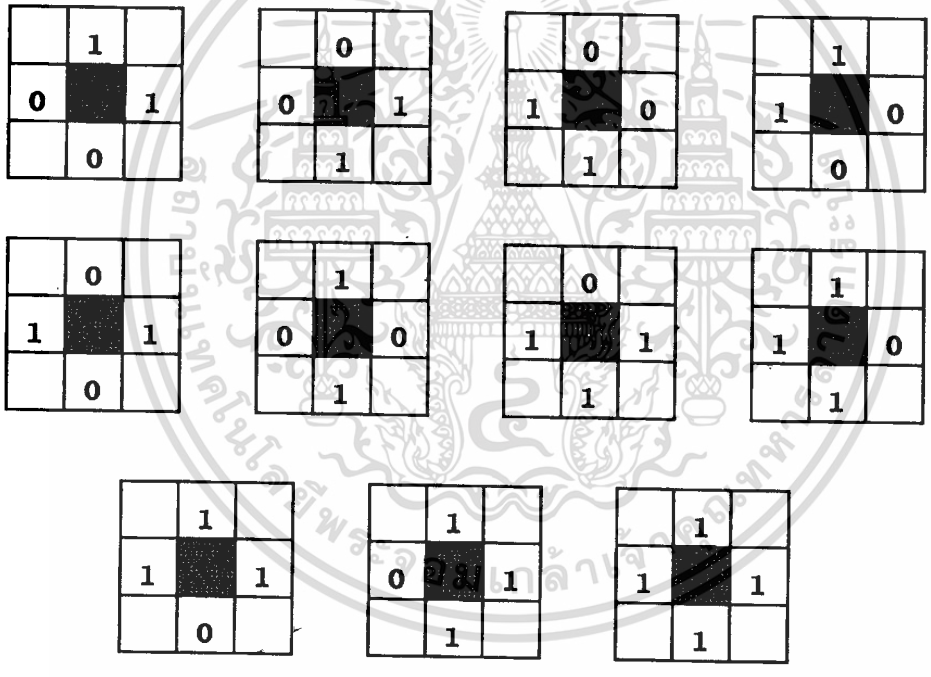
เงื่อนไขของการพิจารณาสถานะของจุดภาพจะทำการกำหนดเงื่อนไขดังต่อไปนี้

1. ถ้ากำหนดให้จุด X เป็นจุดใดๆ ที่มีค่าเป็น 1 แล้ว จุด X จะมีสถานะเป็น จุดตาย ก็ต่อเมื่อมีจุดรอบข้างที่มีค่าเป็น 1 น้อยกว่า 2 จุด ดังแสดงในรูปที่ 2.5



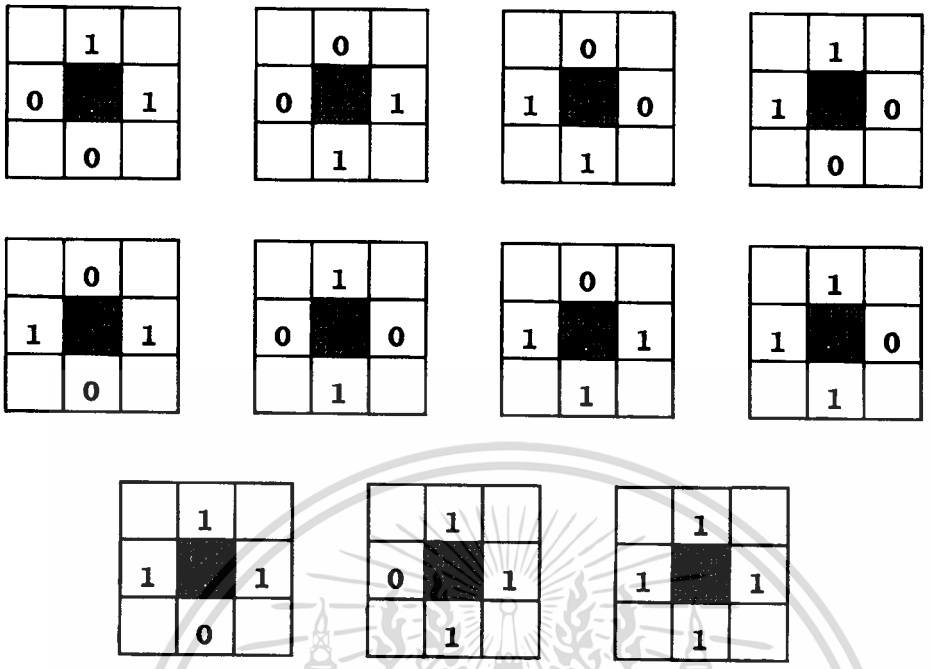
รูปที่ 2.5 แสดงลักษณะของจุดตาย

2. ถ้ากำหนดให้จุด X เป็นจุดใดๆ ที่มีค่าเป็น 1 แล้ว จุด X จะมีสถานะเป็น **จุดเป็น** ก็ต่อเมื่อมีจุดรอบข้างที่มีค่าเป็น 1 มากกว่าหรือเท่ากับ 2 จุด ดังแสดงในรูปที่ 2.6



รูปที่ 2.6 แสดงลักษณะของจุดเป็น

3. ถ้ากำหนดให้จุด Y เป็นจุดใดๆ ที่มีค่าเป็น 0 แล้ว จุด Y จะมีสถานะเป็น **จุดเกิด** ก็ต่อเมื่อมีจุดรอบข้างที่มีค่าเป็น 1 มากกว่าหรือเท่ากับ 2 จุด ดังแสดงในรูปที่ 2.7



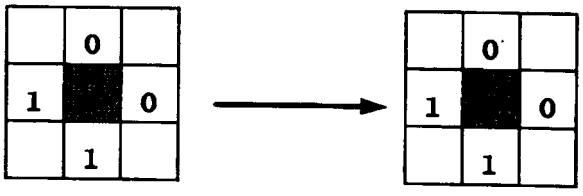
รูปที่ 2.7 แสดงลักษณะของจุดเกิด

สำหรับเงื่อนไขของการเปลี่ยนค่าใหม่ของจุดภาพตามสถานะของจุดภาพที่หาได้นั้น ยังคงใช้เงื่อนไขเดิม ซึ่งมีรายละเอียดดังต่อไปนี้

1. จุดตาย จุดภาพที่มีสถานะเป็นจุดตาย ซึ่งค่าของจุดภาพนั้นมีค่าเป็น 1 จะถูกเปลี่ยนให้มีค่าเป็น 0 ดังแสดงในรูปที่ 2.8

รูปที่ 2.8 แสดงลักษณะของจุดตายที่มีค่าเป็น 1 และถูกเปลี่ยนเป็น 0

2. จุดเป็น จุดภาพที่มีสถานะเป็นจุดเป็น ซึ่งค่าของจุดภาพนั้นมีค่าเป็น 1 จะยังคงมีค่าเป็น 1 ต่อไปไม่เปลี่ยนแปลง ดังแสดงในรูปที่ 2.9



รูปที่ 2.9 ตัวอย่างของจุดเป็นที่มีค่าเป็น 1 ยังคงมีค่าเป็น 1 ต่อไป

3. **จุดเกิด** จุดภาพที่มีสถานะเป็นจุดเกิด ซึ่งค่าของจุดภาพนั้นมีค่าเป็น 0 จะถูกเปลี่ยนให้มีค่าเป็น 1 ดังแสดงในรูปที่ 2.10

รูปที่ 2.10 ตัวอย่างของจุดเกิดที่มีค่าเป็น 0 ถูกเปลี่ยนให้มีค่าเป็น 1

2.4.2 ค่าความหยาบของจุดภาพ (Coarseness value)

จากการประยุกต์ใช้ทฤษฎีเซลลูล่าออโตมาตา ในการกำหนดขอบเขตของส่วนประกอบของหน้าเอกสาร จะสามารถกำหนดได้เพียงโครงร่างของขอบเขตของส่วนประกอบหน้าเอกสารเท่านั้น ขอบเขตต่างๆ ของส่วนประกอบหน้าเอกสารยังคงอยู่กระจัดกระจายบนหน้ากระดาษซึ่งยังไม่เหมาะที่จะแยกคัดลอกออกมา จึงจำเป็นที่จะต้องมีขั้นตอนการทำงานอีกขั้นตอนหนึ่งที่จะจับกลุ่มของขอบเขตต่างๆ ที่กระจัดกระจายกันอยู่รวมเข้าด้วยกันให้ถูกต้องและเหมาะสม ซึ่งการกระทำหนึ่งที่นำมาใช้ก็คือ ค่าความหยาบของจุดภาพ

การใช้ค่าความหยาบของจุดภาพ ก็คือการใช้ค่าคงที่ใดๆ ที่ใช้กำหนดระยะห่างของจุดภาพที่เป็นจุดภาพในขอบเขตต่างๆ ที่กระจัดกระจายอยู่ ตัวอย่างเช่น ถ้าจุดภาพของ 2 ขอบเขตใดๆ อยู่ในช่วงระยะห่างของค่าความหยาบของจุดภาพที่กำหนด ก็จะทำการรวม 2 ขอบเขตนั้นเข้าเป็นขอบเขตหรือพื้นที่เดียวกัน ในการกำหนดค่าความหยาบของจุดภาพ ถ้ากำหนดให้ค่าความหยาบของจุดภาพมีค่าน้อยเกินไป จะทำให้ไม่สามารถจับรวมกลุ่มของขอบเขตที่กระจัดกระ

กันอยู่ได้ และถ้ากำหนดให้ค่าความหยาบของจุดภาพมีค่ามากเกินไปก็จะทำให้ขอบเขตที่กระจัดกระจายอยู่รวมกลุ่มกันเป็นขอบเขตใหญ่เพียงขอบเขตเดียว ซึ่งจะไม่สามารถแยกส่วนประกอบหน้าเอกสารได้ ดังนั้นการกำหนดขอบเขตของส่วนประกอบของหน้าเอกสารที่ถูกต้องและเหมาะสม จึงจำเป็นจะต้องกำหนดค่าความหยาบของจุดภาพที่ใช้ให้เหมาะสมด้วยเช่นกัน

ค่าความหยาบของจุดภาพที่เหมาะสม ที่จะสามารถกำหนดขอบเขตของส่วนประกอบหน้าเอกสารได้อย่างถูกต้อง จะไปสอดคล้องกับตัวแปรอีกตัวหนึ่งนั่นก็คือ ค่าความละเอียดของจุดภาพของข้อมูลภาพ (Image resolution) ซึ่งค่าความละเอียดของจุดภาพเป็นค่าคงที่ ที่ถูกกำหนดไว้ตั้งแต่ขั้นตอนแรกของการตรวจกวาดภาพหน้าเอกสารด้วยเครื่องตรวจกวาดภาพ ค่าความหยาบของจุดภาพที่สอดคล้องกับค่าความละเอียดของจุดภาพ ซึ่งจะต้องทำการทดสอบหาค่าที่เหมาะสมเพื่อที่จะทำการแยกส่วนประกอบต่างๆ ได้ถูกต้อง

2.4.3 การแยกและคัดลอกขอบเขต (Extrating rectangular)

ในการคัดลอกขอบเขตที่เป็นส่วนประกอบของหน้าเอกสารออกจากภาพบนหน้าเอกสาร ซึ่งจะได้นำเอาเทคนิคการติดตามรอยขอบของภาพมาใช้ในการแยก และการคัดลอกขอบเขตออกมา การทำงานจะเริ่มต้นจากการตรวจกวาดจุดภาพจากจุดมุมบนด้านซ้ายไปในทิศทางจากด้านซ้ายไปทางด้านขวา เมื่อตรวจกวาดจนครบแถวแล้วก็จะทำการเลื่อนลงมาข้างล่างจนหมดทั้งหน้าเอกสาร จากการตรวจกวาดภาพเมื่อพบจุดภาพที่เป็นจุดใดจุดหนึ่งของขอบเขต ซึ่งค่าของจุดภาพที่เป็น 1 ก็จะทำการเคลื่อนที่ติดตามรอยขอบของภาพไปเลื่อยๆ ตำแหน่งหรือพิกัดของจุดภาพที่เป็นขอบเขตของภาพจะถูกทำการบันทึกเก็บไว้ และเมื่อเคลื่อนที่ไปจนครบรอบนั่นก็คือเดินตามขอบภาพจนมาถึงจุดเริ่มต้น ก็นจะนำพิกัดของจุดที่เป็นขอบของภาพทั้งหมดมาทำการคำนวณหาจุดพิกัดที่มีค่าน้อยที่สุดและพิกัดที่มีค่ามากที่สุด ทั้งในแนวนอนและแนวตั้ง ซึ่งจะนำค่าดังกล่าวมากำหนดเป็นมุมทั้งสี่ของรูปภาพตัวอักษรซึ่งจะได้เป็นรูปสี่เหลี่ยมล้อมรอบตัวอักษร ซึ่งแต่ละจุดมุมทั้งสี่ของกรอบภาพจะกระทำได้ดังนี้

1. จุดพิกัดมุมบนด้านซ้ายก็คือ พิกัดของจุดขอบที่มีค่าน้อยที่สุดในแนวนอนหรือแนวแกน X กับพิกัดของจุดขอบที่มีค่าน้อยที่สุดในแนวตั้งหรือแนวแกน Y นั่นคือพิกัดของ (X_{min}, Y_{min})

2. จุดพิกัดมุมบนด้านขวาก็คือ พิกัดของจุดขอบที่มีค่ามากที่สุดในแนวนอนหรือแนวแกน X กับพิกัดของจุดขอบที่มีค่าน้อยที่สุดในแนวตั้งหรือแนวแกน Y นั่นคือพิกัดของ (X_{max}, Y_{min})

3. จุดพิกัดมุมล่างด้านซ้ายก็คือ พิกัดของจุดขอบที่มีค่าน้อยที่สุดในแนวนอนหรือแนวแกน X กับพิกัดของจุดขอบที่มีค่ามากที่สุดในแนวตั้งหรือแนวแกน Y นั่นคือพิกัดของ (X_{min},Y_{max})

4. จุดพิกัดมุมล่างด้านขวาก็คือ พิกัดของจุดขอบที่มีค่ามากที่สุดในแนวนอนหรือแนวแกน X กับพิกัดของจุดขอบที่มีค่ามากที่สุดในแนวตั้งหรือแนวแกน Y นั่นคือพิกัดของ (X_{max},Y_{max})

เมื่อมีการคำนวนหาค่าของจุดมุมทั้งสี่ของกรอบภาพได้แล้วก็สามารถที่จะกำหนดขอบเขตของส่วนประกอบของหน้าเอกสาร ที่นำมาเข้าขบวนการนั้นได้ในลักษณะของพื้นที่รูปสี่เหลี่ยมซึ่งจุดภาพทั้งหมดที่อยู่ภายในบริเวณรูปสี่เหลี่ยมจะถูกคัดลอกออกไปจากหน้าเอกสาร และทำการแยกไปเก็บไว้ต่างหากเพื่อที่จะนำไปประมวลผลขั้นต่อไป ซึ่งจุดภาพในบริเวณที่ถูกคัดลอกออกไปแล้วจะถูกเปลี่ยนค่าของจุดภาพภายในทั้งหมดให้เป็นจุดขาวหรือมีค่าเป็น 0 จากนั้นก็จะทำการย้อนกลับไปยังจุดเริ่มต้นที่พบขอบเขตก่อนหน้าที่จะทำการคัดลอกจุดภาพออกไป และก็จะเริ่มทำการตรวจกวาดหาจุดภาพที่เป็นขอบเขตอื่นๆ ต่อไป เพื่อที่จะทำการกำหนดขอบเขตและคัดลอกขอบเขตนั้นๆ ออกมาซึ่งจะมีการทำงานวนไปเรื่อยๆ จนสามารถแยกส่วนประกอบหน้าเอกสารออกได้ทั้งหมด

บทที่ 3
กระบวนการแยกตัวอักษร

กระบวนการแยกตัวอักษร เป็นกระบวนการที่สำคัญอย่างหนึ่งของกระบวนการรู้จำตัวอักษร ซึ่งจะเป็นการแยกตัวอักษรต่างๆ ออกจากข้อความหรือประโยคให้ออกมาทีละตัวเพื่อที่จะนำตัวอักษรที่ได้จากขบวนการนี้เข้าสู่ขบวนการรู้จำตัวอักษรต่อไป ซึ่งในกระบวนการรู้จำตัวอักษรจะสามารถประมวลผลได้ครั้งละหนึ่งตัวอักษรเท่านั้น จากข้อมูลที่ใช้งานจริงจะมีลักษณะเป็นหน้าเอกสาร ซึ่งจะประกอบไปด้วยตัวอักษรหรือข้อความที่เป็นแถวเป็นบรรทัด ดังนั้นจึงต้องทำการแยกอักษรเหล่านั้นออกมาให้มีข้อมูลอักษรเป็นตัวๆ

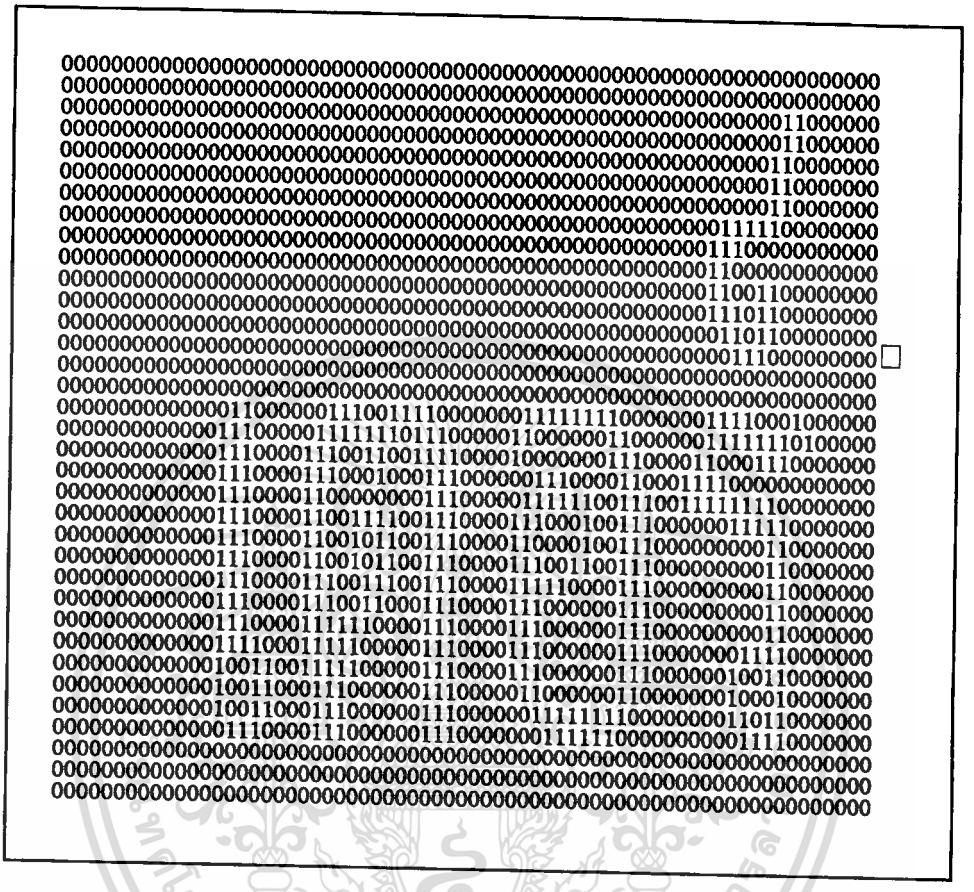
จากการศึกษาลักษณะการเรียงตัวกันของตัวอักษรภาษาไทย เราสามารถที่จะสรุปได้ดังนี้ ตัวอักษรภาษาไทยที่จัดเรียงกันเป็นข้อความนั้นสามารถแบ่งออกได้เป็น 4 ระดับคือ

1. ระดับที่ 1 เป็นระดับบนสุด ประกอบไปด้วย วรรณยุกต์ และตัวการันต์
2. ระดับที่ 2 เป็นระดับบนกลาง ประกอบไปด้วย สระระดับบน
3. ระดับที่ 3 เป็นระดับกลาง ประกอบไปด้วย ตัวพยัญชนะ และสระระดับกลาง
4. ระดับที่ 4 เป็นระดับล่าง ประกอบไปด้วย สระระดับล่าง

ซึ่งลักษณะดังกล่าวของระบบอักษรภาษาไทย มีความแตกต่างกับภาษาอังกฤษเป็นอย่างมาก การแยกตัวอักษรออกจากรูปประโยคทีละตัวอักษรนั้น ได้มีการทำการศึกษาและทำการแยกตัวอักษรได้หลายวิธีการด้วยกัน เช่น การใช้วิธีการติดตามขอบเขตของภาพ หรือวิธีการทางฮิสโตแกรม ดังที่ได้กล่าวมาแล้ว ซึ่งในปริญญานิพนธ์ฉบับนี้ได้เลือกวิธีการติดตามขอบเขตของภาพมาใช้ทำการวิจัย ซึ่งวิธีการนี้จะไม่เกิดปัญหาในรูปแบบของการตรวจควาดภาพที่มีการวางเอกสารไม่ตรงหรือมีการวางที่เอียง

3.1 การแปลงข้อมูล TIFF file เป็น binary

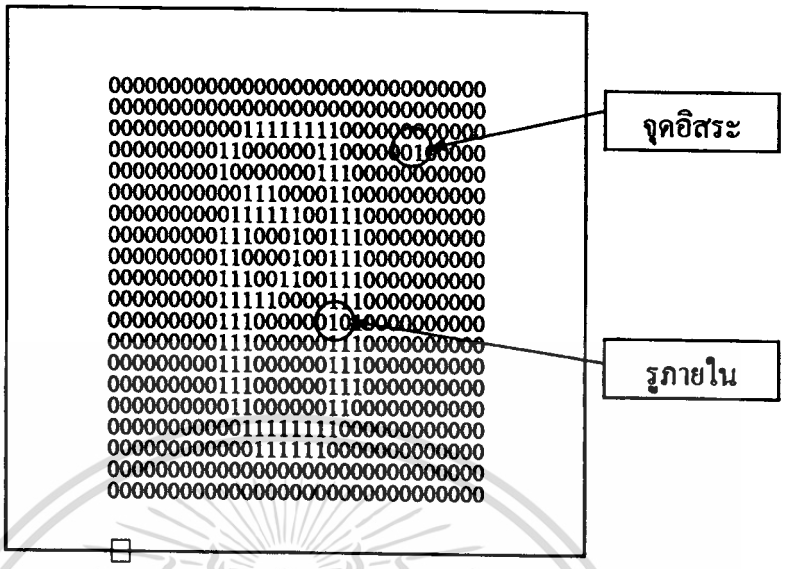
เมื่อรับข้อมูลภาพจากเครื่องตรวจกวาดภาพจะต้องทำการแปลงข้อมูลจากแฟ้มข้อมูลที่ได้ในรูปของ TIFF file ให้อยู่ในรูปแบบที่ใช้ในการประมวลผลได้ โดยพิจารณาตามโครงสร้างแฟ้มข้อมูล ซึ่งการแปลงข้อมูลภาพให้เป็นแบบข้อมูล binary โดยจะทำการแปลงจุดภาพให้มีค่าตัวเลขสองระดับคือค่า 0 และค่า 1 ซึ่งจุดภาพที่เป็นจุดดำหรือเนื้อตัวอักษรจะแทนด้วยค่า 1 และจุดภาพที่เป็นจุดขาวหรือพื้นของภาพจะแทนด้วยค่า 0 ดังแสดงในรูปที่ 3.1



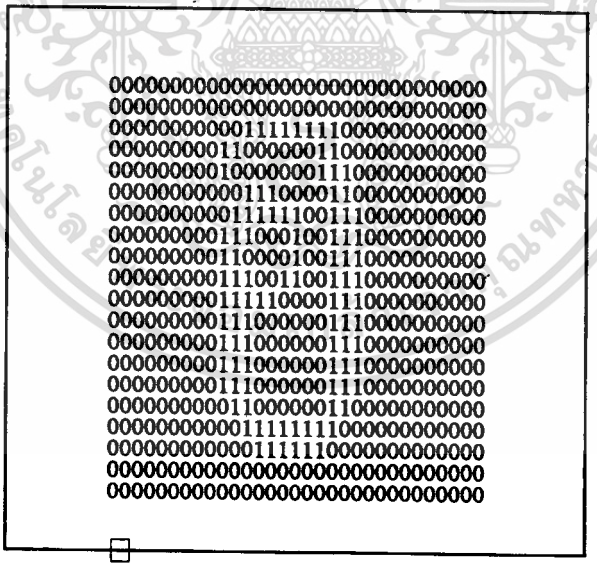
รูปที่ 3.1 แสดงข้อมูลที่ได้จากการแปลง TIFF file เป็น binary

3.2 การกำจัดสัญญาณรบกวน

จากการแปลงข้อมูลภาพแล้วข้อมูลที่ได้อาจจะมีสัญญาณรบกวนที่ไม่ใช่ข้อมูลของหน้าเอกสาร เพราะฉะนั้นจะต้องทำการกำจัดสัญญาณรบกวนที่มีเพิ่มเข้ามา หรือส่วนที่ขาดหายไปให้มีความสมบูรณ์ รูปแบบของสัญญาณรบกวนที่พบคือจะมีลักษณะเป็น จุดอิสระ (Isolate point) และแบบที่เป็นรูภายใน (Isolate Hole) การกำจัดสัญญาณรบกวนทั้งสองลักษณะนี้จะใช้ทฤษฎีของจุดเป็น, จุดตาย และจุดเกิด มาช่วยในการวิเคราะห์โดยจะทำการตรวจกวาดไปบนข้อมูลเอกสารทั้งหมด



รูปที่ 3.2 แสดงรูปที่เกิดสัญญาณรบกวนจากจุดอิสระและรูภายใน



รูปที่ 3.3 แสดงการกำจัดสัญญาณรบกวนแล้ว

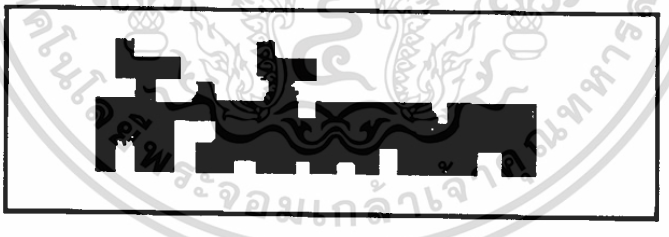
3.3 การกำหนดส่วนประกอบออกเป็นแต่ละส่วน

ในหนึ่งหน้าเอกสารนั้น เราจะต้องทำการกำหนดส่วนประกอบต่าง ๆ ในหน้าเอกสารออกเป็นแต่ละส่วน เพื่อที่จะนำส่วนประกอบต่างๆ เหล่านี้ ไปทำการกำหนดประเภทว่าส่วนประกอบเหล่านั้นเป็นภาพตัวอักษรหรือรูปภาพ การกำหนดส่วนประกอบในหน้าเอกสารออกเป็นส่วนเดี่ยวๆ จากภาพหน้าเอกสาร วิธีการที่นำมาใช้คือ

1. การเพิ่มค่าความหยาบของจุดภาพ

ณ.ที่จุดภาพของ 2 เซตใดๆ อยู่ในช่วงระยะห่างของค่าความหยาบของจุดภาพที่กำหนดไว้ก็จะทำการรวม 2 ขอบเขตนั้นเข้าเป็นขอบเขตหรือพื้นที่เดียวกัน ดังตัวอย่างรูปที่ 3.5

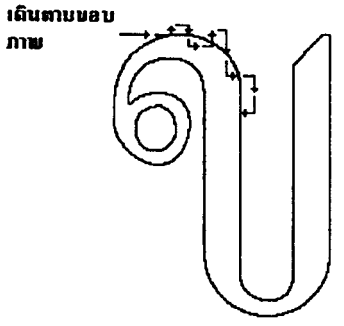
รูปที่ 3.4 แสดงภาพตัวอักษรที่ยังไม่ได้เพิ่มความหยาบของภาพ



รูปที่ 3.5 แสดงภาพตัวอักษรที่ภานการเพิ่มค่าศวามหยาบของภาพแล้ว

2. การเดินตามขอบของภาพ

เป็นการเดินไต่ไปตามขอบของภาพระหว่างส่วนที่เป็นรูปภาพ (image) กับส่วนที่เป็นพื้นเบื้องหลัง (Background) โดยจะตรวจกวาดไปทุกๆ จุดภาพ (Pixel) โดยจะเริ่มจากจุดมุมซ้ายบนของข้อมูลภาพ(เป็นจุดภาพจุดแรกที่ตรวจกวาดเจอ) แล้วตรวจกวาดไปในทิศทางจากซ้ายไปขวา และเลื่อนลงจากบนลงล่าง เมื่อตรวจกวาดมาพบจุดภาพใดๆ ที่มีค่าของจุดภาพเป็น 1 ก็จะเปลี่ยนลักษณะการเคลื่อนที่ไปยังจุดภาพจุดถัดไปเสียใหม่ จนกว่าจะมาพบจุดเริ่มต้นที่ตรวจกวาดเจอ ซึ่งอาจจะกล่าวใหม่ได้ว่าเป็นการไต่ไปตามขอบนอกของรูปภาพหรือตัวอักการนั่นเอง



รูปที่ 3.6 แสดงการเดินตามขอบภาพเพื่อทำการแยกตัวอักษร

3.4 การแยกส่วนประกอบออกจากหน้าเอกสาร

การแยกส่วนประกอบในหน้าเอกสารออกจากภาพหน้าเอกสาร ได้นำเทคนิคการติดตามรอยขอบของภาพมาใช้ในการแยกขอบเขตออกจากหน้าเอกสาร การทำงานเริ่มตรวจกวาดจุดภาพ เมื่อตรวจกวาดมาพบจุดภาพที่เป็นจุดใดจุดหนึ่งของขอบเขต ซึ่งมีค่าของจุดภาพเป็น 1 ก็จะเคลื่อนติดตามรอยขอบของเขตไป โดยอาศัยเทคนิคการติดตามรอยขอบของภาพในขณะที่เคลื่อนติดตามรอยขอบของภาพไป ตำแหน่งหรือพิกัดของจุด (corrdibate) ที่เป็นขอบของภาพจะถูกบันทึกเก็บไว้ด้วย และเมื่อเคลื่อนที่ไปจนครบรอบนั่นคือกลับมาอยู่ที่จุดเริ่มต้นแล้ว ก็จะนำพิกัดของจุดที่เป็นขอบภาพทั้งหมดมาคำนวณหาจุดพิกัดที่มีค่ามากที่สุดและน้อยที่สุดทั้งในแนวนอนและแนวตั้งมากำหนดเป็นจุดมุมทั้งสี่ของขอบเขตภาพ แต่ละจุดมุมของขอบเขตภาพรูปสี่เหลี่ยมมีดังนี้

1. พิกัดของจุดมุมบนด้านซ้ายคือ พิกัดของจุดขอบที่มีค่าน้อยที่สุดในแนวตั้ง
2. พิกัดของจุดมุมบนด้านขวาคือ พิกัดของจุดขอบที่มีค่าน้อยที่สุดในแนวตั้ง
3. พิกัดของจุดมุมล่างด้านซ้ายคือ พิกัดของจุดขอบที่มีค่ามากที่สุดในแนวตั้ง
4. พิกัดของจุดมุมล่างด้านขวาคือ พิกัดของจุดขอบที่มีค่ามากที่สุดในแนวตั้ง

โดยกำหนดให้จุดมุมบนด้านซ้ายของข้อมูลภาพทั้งหมดเป็นจุดเริ่มต้น (Origin point) มีพิกัดของจุดเป็น (0,0) แนวแกน X คือแกนอ้างอิงในแนวนอน และแนวแกน Y คือแกนอ้างอิงในแนวตั้ง

เมื่อได้จุดมุมทั้ง 4 ของรูปสี่เหลี่ยมแล้ว ก็จะสามารถกำหนดขอบเขตของส่วนประกอบของหน้าเอกสารนั้นๆ ได้ จุดภาพทั้งหมดที่อยู่ภายในขอบเขตนั้นจะถูกคัดลอกออกจากภาพหน้าเอกสาร แล้วนำมาแยกเก็บไว้ต่างหากเพื่อการประมวลในขั้นตอนต่อไป จุดภาพในบริเวณขอบเขตที่ถูกคัดลอกจุดภาพออกไปแล้ว ก็จะถูกเปลี่ยนค่าของจุดภาพภายในบริเวณนั้นๆ

ให้มีค่าเป็น 0 ให้หมด จากนั้นก็จะย้อนกลับไปยังจุดเริ่มต้นที่พบขอบเขตก่อนหน้านี้ที่เพิ่งจะคัดลอกจุดภาพออกไป เริ่มตรวจกวาดหาจุดภาพที่เป็นขอบเขตอื่นต่อไป เพื่อที่จะกำหนดขอบเขตและคัดลอกขอบเขตนั้นๆ ออกมา การทำงานจะทำงานวนไปในลักษณะเช่นนี้ไปเรื่อยๆ จนสามารถกำหนดและแยกคัดลอกขอบเขตที่เป็นส่วนประกอบของหน้าเอกสารออกมาได้ทั้งหมด

3.5 การคำนวณหาจุดอ้างอิงในการแบ่งแยกชนิดของภาพและตัวอักษร

ส่วนประกอบในหน้าเอกสารแต่ละตัวที่อยู่ในขั้นตอนการแบ่งแยกชนิดของภาพและตัวอักษรออกจากกันจะมีการบันทึกขนาดและความสูงของส่วนประกอบต่างๆ นั้นด้วย และการบันทึกพิกัดของส่วนประกอบในหน้าเอกสาร เพื่อที่จะให้รู้ว่าอยู่ในตำแหน่งของแถวใด หรืออยู่ในส่วนใดในหน้าเอกสารส่วนนั้น

ในบรรทัดแรก (ส่วนแรก) ของหน้าเอกสาร (อาจจะเป็นรูปภาพก็ได้) จะบันทึกขนาดความสูงของส่วนประกอบหน้าเอกสารส่วนนั้นเอาไว้ โดยมีการกำหนดตัวแปรที่ใช้ในการคำนวณดังนี้

ให้ H(i) แทนขนาดความสูงของส่วนประกอบในหน้าเอกสาร

i=1,2,3,...,n

H(i+1) = ขนาดความสูงของส่วนประกอบในหน้าเอกสารถัดไป

H_{min} = ตำแหน่งถ้าที่มีความสูงที่มีน้อยที่สุดในหน้าเอกสาร

H_{max} = ตำแหน่งค่าความสูงที่มีค่ามากที่สุดในหน้าเอกสาร

ขั้นตอนการคำนวณหาค่าความสูงของส่วนประกอบมีดังนี้

1. บันทึกตำแหน่งของพิกัดค่าความสูงของส่วนประกอบบรรทัดแรกที่เจอ และถ้ากำหนดให้เป็นพิกัดค่าความสูงที่มีค่ามากที่สุดและน้อยที่สุด $H_{min}=H_{max}=H_{(1)}$

2. ตรวจกวาดไปยังส่วนประกอบบรรทัดต่อไป และบันทึกตำแหน่งค่าความสูงไว้ในตัวแปรที่สมมุติขึ้น แล้วทำตามเงื่อนไขต่อไปนี้

2.1 ถ้าความสูงของบรรทัดมีค่าน้อยกว่า H_{min} ให้บันทึกพิกัดค่าความสูง H_{min} ใหม่ด้วยค่าตำแหน่งพิกัดความสูงของส่วนประกอบปัจจุบัน

2.2 ถ้าความสูงของบรรทัดมีค่ามากกว่า H_{min}แล้วย้อนกลับไปพิจารณาตามเงื่อนไขในข้อที่ 2 โดยวนเช่นนี้ไปเรื่อยๆ จนกระทั่งถึงส่วนประกอบสุดท้ายของหน้าเอกสารก็จะได้ค่าของตำแหน่งพิกัดความสูงมากที่สุดและน้อยที่สุด ซึ่งจะนำมาคำนวณหาค่าอ้างอิงต่อไป

ขั้นตอนการหาจุดอ้างอิงมีดังนี้

1. นำเอาตำแหน่งพิกัดความสูงสูงสุดลบด้วย ตำแหน่งพิกัดความสูงต่ำสุดและนำมาบันทึกไว้ และนำมาคำนวณต่อไปดังนี้

$$x = (H_{max} - H_{min})/5$$

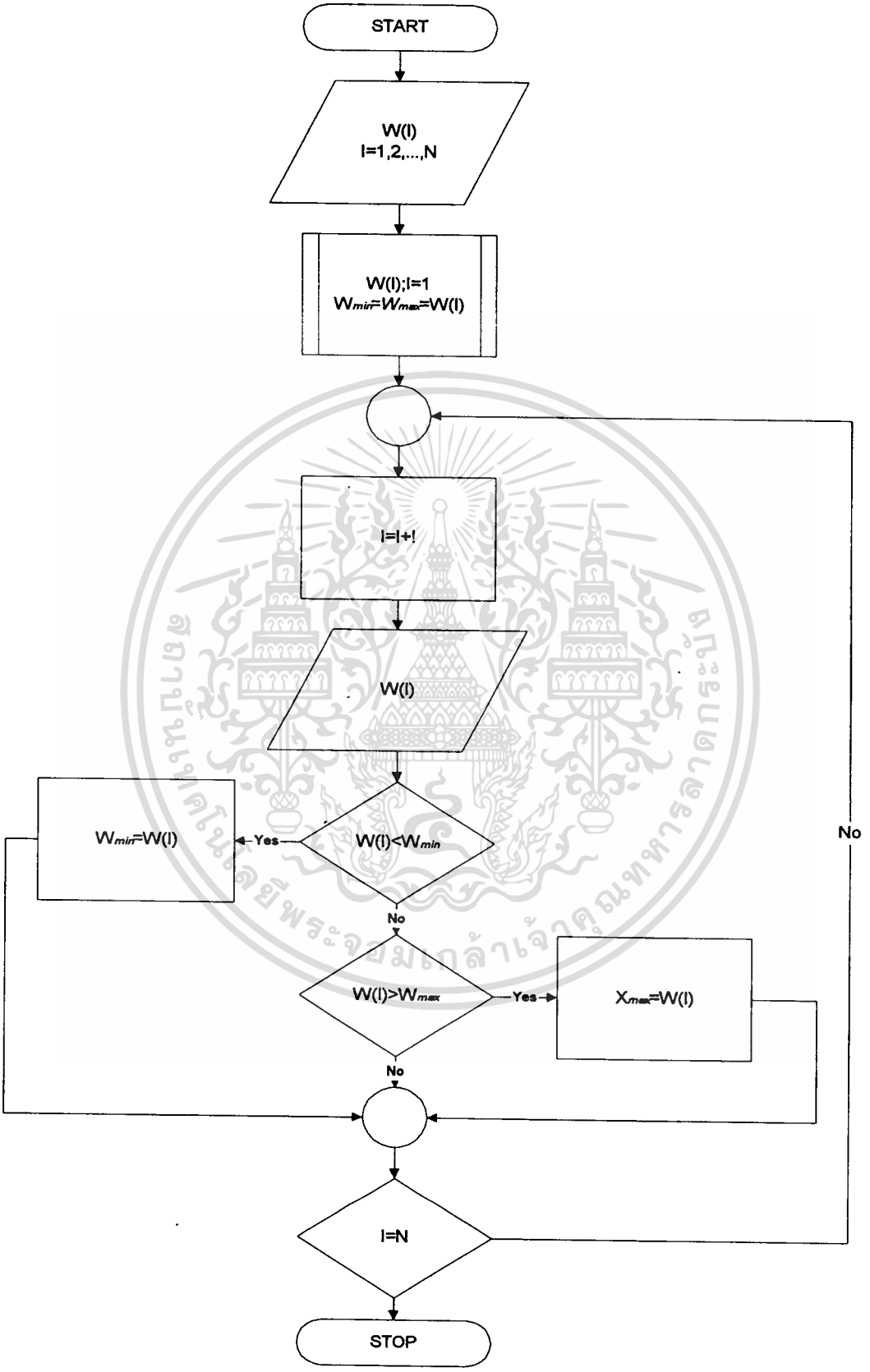
$$\text{level } 1 = H_{min} + x$$

$$\text{level } 2 = \text{level } 1 + x$$

$$\text{level } 3 = \text{level } 2 + x$$

$$\text{level } 4 = \text{level } 3 + x$$

$$\text{level } 5 = \text{level } 4 + x$$



รูปที่ 3.7 แสดงผังการทำงานของการหาความกว้างและความสูงของส่วนประกอบ

2. กำหนดตัวแปร Score ขึ้นมา 5 ชุด

Score(i) ;I = 1,2,3,4,5

3. คำนวณหาค่าว่าค่าตำแหน่งพิกัดความสูงในระดับใดมากที่สุด และให้ค่าตำแหน่งพิกัดนั้นเป็นพิกัดอ้างอิง

4. หาตำแหน่งพิกัดอ้างอิงมาคำนวณกับค่าตำแหน่งพิกัดสูงสุด เพื่อตรวจสอบว่าค่าตำแหน่งพิกัดของความสูงที่สูงสุดเป็นภาพตัวอักษรหรือรูปภาพ ดังนี้

if(base_line*2>H_{max})

5. นำค่าตำแหน่งพิกัดอ้างอิงมาคำนวณกับทุกส่วนประกอบในหน้าเอกสาร เพื่อตรวจสอบว่าส่วนประกอบนั้นเป็นรูปภาพหรือภาพตัวอักษร

5.1 ถ้าตำแหน่งพิกัดอ้างอิงมากกว่า หรือเท่ากับค่าตำแหน่งพิกัดปัจจุบัน ให้ส่วนประกอบนั้นเป็นภาพตัวอักษร

5.2 ถ้าไม่ ก็ให้ส่วนประกอบนั้นเป็นรูปภาพ

ดังนี้

```
if(base_line >= H(i))
        Zone_type(i) = text
else    Zone_type(i) = picture
```

โดยในหลักการที่นำเสนอและได้นำมาใช้นี้ได้นำมาจากหลักการที่ว่า โดยทั่วไปแล้วรูปภาพกับภาพตัวอักษรนั้น รูปภาพนั้นจะมีความสูงมากกว่าภาพตัวอักษรมาก ซึ่งจะเห็นได้จากรูปที่ 3.8

ซึ่งจากรูปเราจะเห็นได้ว่า รูปภาพนั้นจะมีความสูงเมื่อเทียบกับความสูงของตัวอักษรนั้นจะมีค่ามากกว่า 2 เท่าขึ้นไป โดยในกรณีที่ประโยคในบรรทัดนั้นไม่มีวรรณยุกต์ เช่น คำว่า "วารสาร" นั้น จะมีความสูงในระดับหนึ่ง คำว่า "วิศวกรรมศาสตร์" จะมีความสูงในระดับหนึ่ง และคำว่า "เทคโนลาดกระบัง" จะมีความสูงในระดับหนึ่ง แต่ในระดับความสูงของแต่ละประโยค ของประโยคในภาษาไทย ที่มีวรรณยุกต์ และไม่มีวรรณยุกต์ต่างๆ จะมีความสูงมากกว่ากันไม่เกิน 2 เท่าตัว

ในขั้นตอนการคำนวณเมื่อเราได้ความสูงบรรทัดอ้างอิงแล้ว (เป็นจำนวนของส่วนประกอบในหน้าเอกสาร ที่อยู่ในระดับความสูงนั้นมากที่สุด) เราจะมาเทียบกับความสูงของส่วนประกอบในหน้าเอกสาร ที่มีความสูงมากที่สุดว่ามีความสูงมากกว่าเป็น 2 เท่าของความสูงบรรทัดอ้างอิงหรือไม่ ถ้ามากกว่าจะทำการเปลี่ยนค่าของจุดบรรทัดอ้างอิงใหม่ ให้เป็นค่าของความสูง

ของส่วนประกอบที่มีค่ามากที่สุด ซึ่งหมายความว่าในหน้าเอกสารนั้นไม่มีรูปภาพเลย และถ้าน้อยกว่าหรือเท่ากันจะให้คงค่าเดิมเอาไว้ ซึ่งหมายความว่าในหน้าเอกสารนั้นจะมีรูปภาพอยู่ด้วย

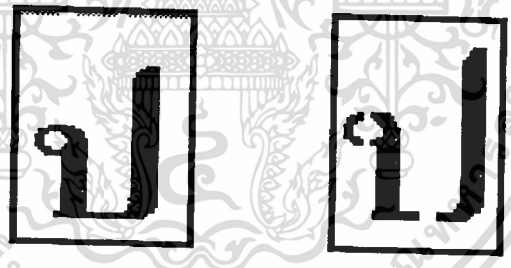
ดังนั้นในขั้นตอนการหาแบ่งแยกชนิดว่าเป็นส่วนประกอบหน้าเอกสารส่วนนั้นเป็นรูปภาพหรือภาพตัวอักษรนั้น จะเป็นการตรวจสอบว่าความสูงของจุดความสูงอ้างอิงกับความสูงส่วนประกอบในหน้าเอกสารที่นำมาตรวจสอบ ถ้ามากกว่าให้ส่วนประกอบนั้นเป็นรูปภาพ แต่ถ้ามีค่าน้อยกว่าหรือเท่ากัน ให้ส่วนประกอบนั้นเป็นประโยคของตัวอักษร

รูปที่ 3.8 แสดงภาพของความสูงและความสูงของส่วนประกอบบนหน้าเอกสาร

3.6 การแยกตัวอักษรออกจากประโยค

เมื่อได้ทำการแยกภาพออกจากหน้าเอกสารแล้ว ขั้นตอนต่อไปก็คือ การแยกตัวอักษรแต่ละตัวออกจากประโยค โดยอาศัยหลักการเดินตามขอบภาพเช่นเดียวกับการแยกภาพออกจากตัวอักษรแต่ในการเพิ่มค่าความหยาบของจุดภาพนั้น จะใช้ค่าที่น้อยกว่ามากและเมื่อเราเพิ่มค่าความหยาบของจุดภาพแล้ว ในการเดินตามขอบภาพเพื่อต้องการแยกตัวอักษรนั้นในครั้งแรก ตัวอักษรที่ได้อาจจะมีวรรณยุกต์ติดมาด้วย ทำให้ต้องมีการแยกวรรณยุกต์ออกจากตัวอักษรด้วย เพื่อความถูกต้อง โดยในการแยกตัวอักษรครั้งที่ 2 นี้ การเพิ่มค่าความหยาบของจุดภาพนั้น จะเพิ่มในแนวนอนเท่านั้น จะไม่เพิ่มค่าความหยาบในแนวตั้ง เพราะว่าการเพิ่มในแนวตั้งจะทำให้มีวรรณยุกต์ติดมาด้วย

เหตุผลที่ต้องมีการเพิ่มค่าความหยาบในขั้นนี้ เพราะว่าถ้าไม่เพิ่มค่าความหยาบแล้วการเดินตามขอบภาพนั้นจะได้ภาพที่ไม่สมบูรณ์ จะมีการขาดหายไปในบางส่วนของภาพตัวอักษร ดังรูปที่ 3.9



รูปที่ 3.9 แสดงภาพของตัวอักษรที่มีการขาดหาย

จากรูปจะเห็นได้ว่าตัวอักษร “ป” นั้นขาดหัวและขาดกลางตัว เราจึงต้องเพิ่มค่าความหยาบของจุดภาพ เพื่อให้ภาพมีความต่อเนื่องมากขึ้น

รูปที่ 3.10 แสดงภาพประโยคที่ยังไม่ผ่านการแยกตัวอักษร

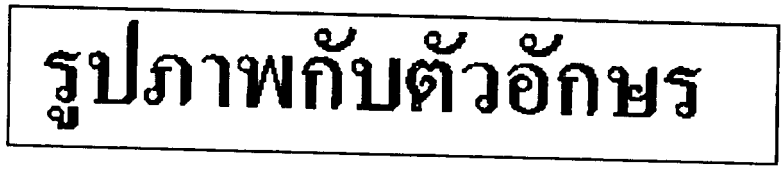
รูปที่ 3.11 แสดงการตรวจกวาดเพื่อหาตัวอักษร

รูปที่ 3.12 แสดงภาพประโยคที่ผ่านการแยกตัวอักษรครั้งแรก
โดยใช้หลักการเดินตามขอบภาพและลบตัวอักษรตัวนั้นออกจากภาพ

3.7 การจัดเรียงรูปประโยคหลังการวิเคราะห์

ตัวอักษรแต่ละตัวที่ถูกแยกออกจากภาพประโยคแล้ว ก่อนที่จะทำการส่งผ่านไปทำการวิเคราะห์และขบวนการจดจำตัวอักษร จะถูกบันทึกค่าตำแหน่งพิกัด และคำนวณหาขนาดความสูงของภาพและความสูงของตัวอักษร ซึ่งก็จะนำไปคำนวณหาค่าตำแหน่งของตัวอักษร (x_{con}, y_{con}) ในภาพประโยคได้ เมื่อภาพตัวอักษรถูกส่งไปวิเคราะห์และได้ผลลัพธ์ออกมาแล้วว่าเป็นตัวอักษรอะไรรหัส ASCII ของภาพตัวอักษรนั้นจะถูกนำมาจัดเรียงลำดับก่อนการจัดบันทึกลงไฟล์ โดยจะจัดเรียงตัวอักษรตามค่า y_{con} ของตัวอักษรแต่ละตัวจากค่าน้อยไปมาก เพื่อจัดเรียงตัวอักษรให้อยู่ในบรรทัดที่ถูกต้อง และจัดเรียงตามค่า x_{con} จากค่าน้อยไปหามากเพื่อจัดเรียงตัวอักษรให้อยู่ในตำแหน่งที่ประกอบขึ้นเป็นคำที่ถูกต้อง ตัวอย่างของการจัดเรียงตัวอักษรมีขั้นตอนการจัดเรียงดังนี้

ขั้นตอนที่ 1. ตัวอย่างข้อความเดิมเมื่อยังไม่ถูกแยกตัวอักษรออกจากประโยค

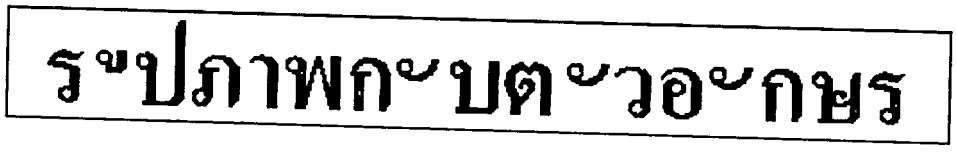


รูปที่ 3.13 แสดงข้อความเมื่อยังไม่ถูกแยกออกจากประโยค

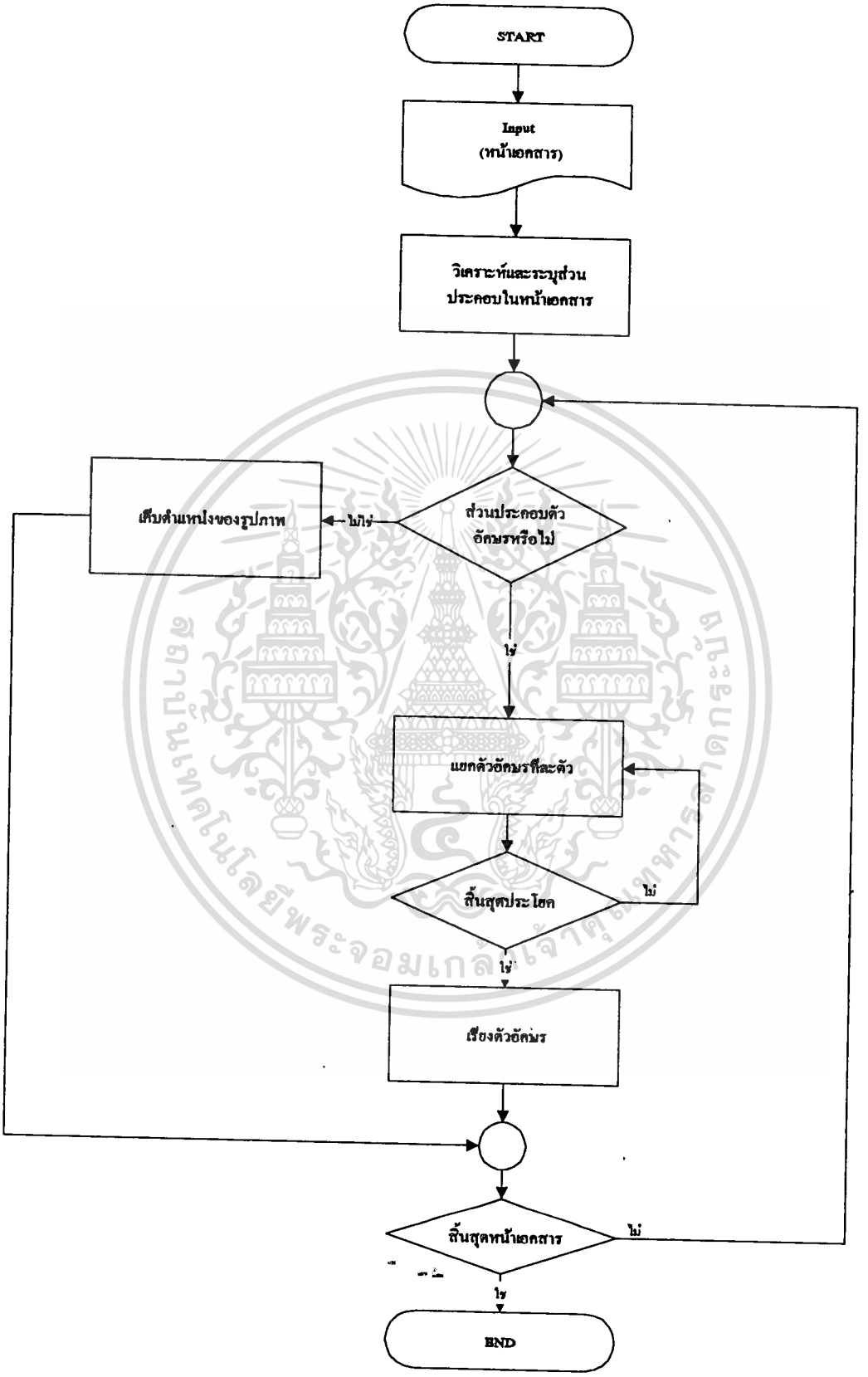
ขั้นตอนที่ 2. เมื่อผ่านขั้นตอนการแยกตัวอักษรออก จะแยกตัวอักษรในแต่ละบรรทัดออกมาโดยมีลำดับดังนี้

รูปที่ 3.14 แสดงข้อความที่ผ่านขั้นตอนการแยกตัวอักษรออกจากประโยค

ขั้นตอนที่ 3. ในข้อความแม้ว่าตัวอักษร "ป" จะถูกแยกออกมาก่อนก็ตาม เมื่อจัดเรียงตัวอักษรตามค่า $x_{position}$ จะได้ตัวอักษร "ร" เป็นตัวอักษรตัวแรกแล้วตามด้วยตัวสะระ "อู" เรียงต่อไปเรื่อย ๆ จนหมดบรรทัด โดยเรียงตามค่า $x_{position}$ ก็จะสามารถเรียงตัวได้ เมื่อจัดเรียงตัวอักษรทั้งหมดแล้ว จะได้ตัวอักษรในแฟ้มข้อมูลอักขระเรียงเป็นประโยคข้อความ



รูปที่ 3.15 แสดงข้อความที่ผ่านการเรียงตัวอักษร



รูปที่ 3.16 แสดงแผงผังการทำงานของระบบวิเคราะห์และแยกตัวอักษร

บทที่ 4
การทำงานและการทดลอง

4.1 การทำงานของโปรแกรมวิเคราะห์หน้าเอกสาร

การทำงานของโปรแกรมในขั้นแรก เริ่มจากการแปลงแฟ้มข้อมูลหน้าเอกสารให้เป็นข้อมูลที่พร้อมที่จะใช้ในการคำนวน จากนั้นในขั้นตอนต่อไปจะเป็นวิเคราะห์ส่วนประกอบในหน้าเอกสาร โดยในขั้นตอนนี้จะเป็นการระบุส่วนประกอบในหน้าเอกสาร ส่วนประกอบในหน้าเอกสารที่เป็นรูปภาพจะถูกแยกออกไป ส่วนที่เป็นประโยคตัวอักษรจะถูกนำมาแยกที่ละตัว หลังจากนั้นจะนำตัวอักษรที่แยกนำมาเรียงกันใหม่ ดังนั้นผลลัพธ์ที่ได้จะเป็นแฟ้มข้อมูลที่ประกอบไปด้วยตำแหน่งของตัวอักษรที่แยกออกมา

4.1.1 อุปกรณ์ที่ต้องนำมาใช้ในการทำงานของโปรแกรม

1. เครื่องไมโครคอมพิวเตอร์ไอบีเอ็ม หรือเครื่องคอมแพทิเบิล
2. หน่วยความจำ RAM(Ramdom Access Memory) อย่างน้อย 4 Mbytes ขึ้นไป
3. จอแสดงผลเป็น VGA เป็นอย่างต่ำ
4. ฮาร์ดดิส 1 ตัว พร้อมเนื้อที่ว่าง 6 Mbytes ขึ้นไป
5. ระบบปฏิบัติการ MS-DOS เป็นอย่างต่ำ
6. เครื่องตรวจควาดภาพ
7. ไมโครซอร์ฟเมาส์ หรือเมาส์ที่คอมแพทิเบิล

4.1.2 การติดตั้งโปรแกรมวิเคราะห์หน้าเอกสาร

1. ใส่แผ่นติดตั้งไว้ที่ดิสก์ไดรฟ์
2. สร้างไดเร็คทอรีย่อยไว้ที่ไดรฟ์ C: เช่น MD C:\analist
3. copy File ในแผ่นติดตั้ง ไปที่ไดรฟ์ C: และไดเร็คทอรีที่สร้างไว้ เช่น

   copy a:*.* c:\analist <Enter>
4. รอจนกระทั่งการกอปปี้เสร็จสมบูรณ์

4.2 ขั้นตอนการใช้งานของโปรแกรม

การทำงานของโปรแกรมจะเป็นการทำงานผ่านเมนู โดยการใช้เมาส์และคีย์บอร์ดเลือกขั้นตอนต่างๆ โดยจะมีขั้นตอนดังต่อไปนี้

ขั้นที่ 1. การเรียกการใช้งานของโปรแกรม จะเป็นการเรียกชื่อโปรแกรมที่คอสพรอมท์ เช่น C:\analist\segment <Enter>

โปรแกรมจะเริ่มการทำงาน โดยมีเมนูในการสั่งงาน

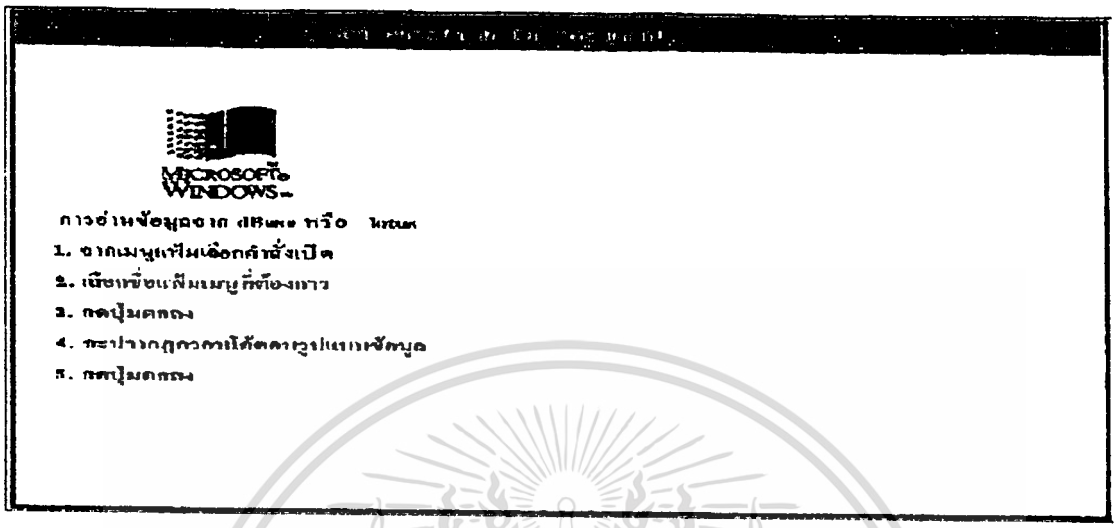
รูปที่ 4.1 แสดงการทำงานของโปรแกรมเมื่อเริ่มต้นทำงาน

ขั้นที่ 2. การเปิดภาพหน้าเอกสาร

โดยการเลือกที่เมนู FILE ตามด้วยฟังชั่น Open จากนั้นพิมพ์ชื่อแฟ้มภาพหน้าเอกสาร โดยโปรแกรมจะแปลงข้อมูลหน้าเอกสารให้พร้อมที่จะใช้ในการคำนวณ หลังจากพิมพ์ชื่อแฟ้มหน้าเอกสารแล้วจะแสดงหน้าเอกสารให้เห็น

```
Enter File name(*.tif): test.tif
```

รูปที่ 4.2 แสดงการพิมพ์ชื่อแฟ้มข้อมูลเพื่อทำการเปิดแฟ้มข้อมูล



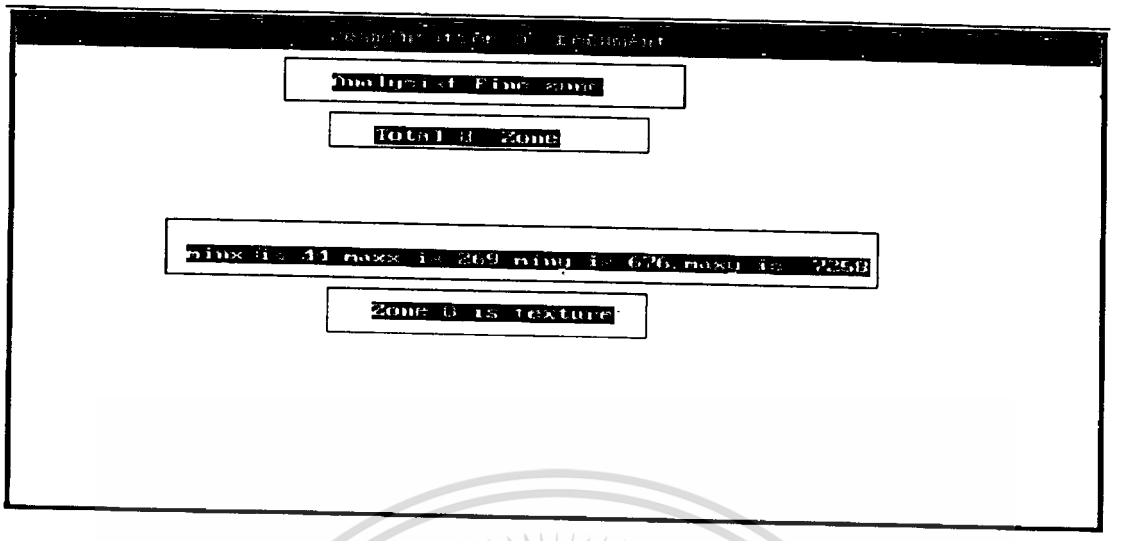
รูปที่ 4.3 แสดงภาพหน้าเอกสารหลังจากการเปิดแฟ้มข้อมูล

ขั้นที่ 3. การระบุหน้าเอกสาร

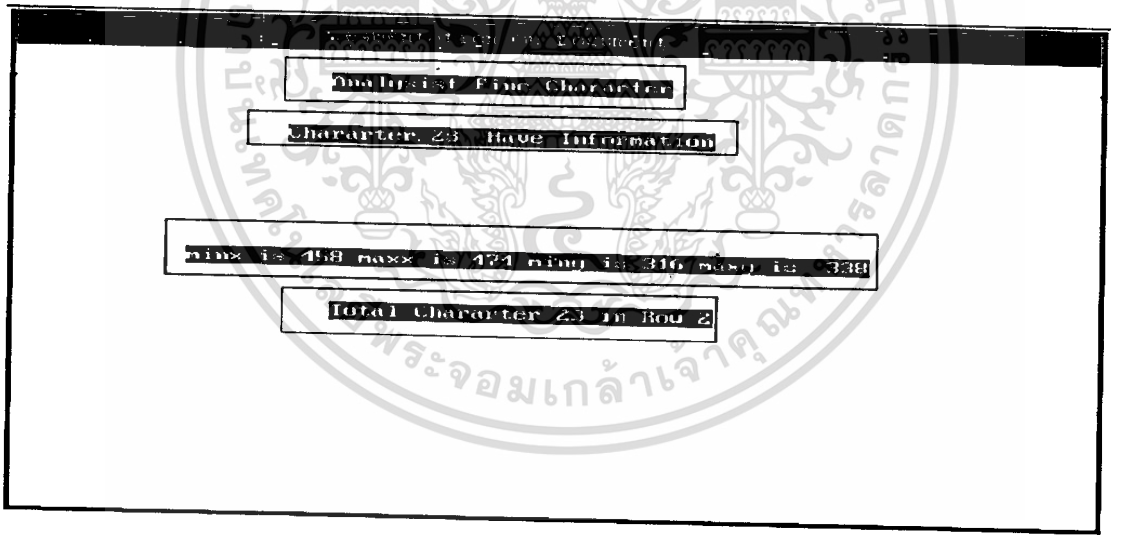
ในขั้นตอนที่ 3. นี้ จะทำงานได้จะต้องผ่านขั้นตอนการเปิดภาพหน้าเอกสารในขั้นตอนที่ 2. มาก่อน การทำงานในขั้นตอนที่ 3. ทำได้โดยเลือกเมนู Start โดยโปรแกรมจะทำการวิเคราะห์และระบุส่วนประกอบในหน้าเอกสาร และกำหนดขอบเขตของส่วนประกอบต่างๆ หลังจากนั้นจะเป็นการระบุส่วนประกอบของหน้าเอกสาร และทำการแยกตัวอักษรออกทีละตัว โดยทุกๆขั้นตอนที่กล่าวมา โปรแกรมจะทำงานเองอัตโนมัติ

กระบวนการต่างๆที่อยู่ในขั้นตอนการระบุหน้าเอกสาร

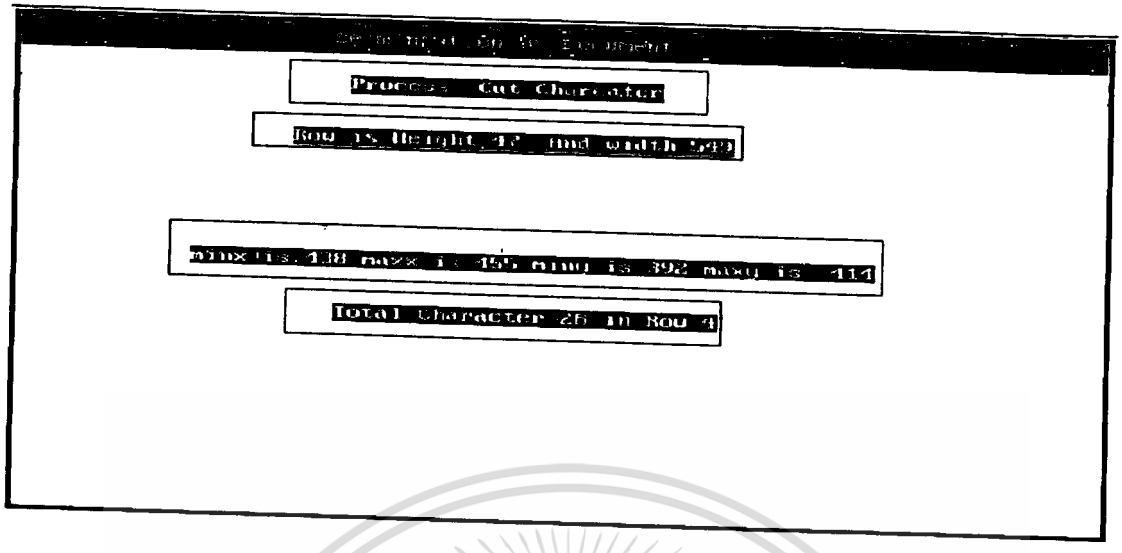
- การระบุส่วนประกอบของหน้าเอกสาร (แสดงในรูปที่ 4.4)
- การแยกตัวอักษรทีละตัว (แสดงในรูปที่ 4.5)
- การแยกตัวพยัญชนะ,สระ,วรรณยุกต์ และตัวการันต์ออกจากกัน (แสดงในรูปที่ 4.6)
- การเรียงตัวอักษรและเก็บตำแหน่งของตัวอักษร (แสดงในรูปที่ 4.7)



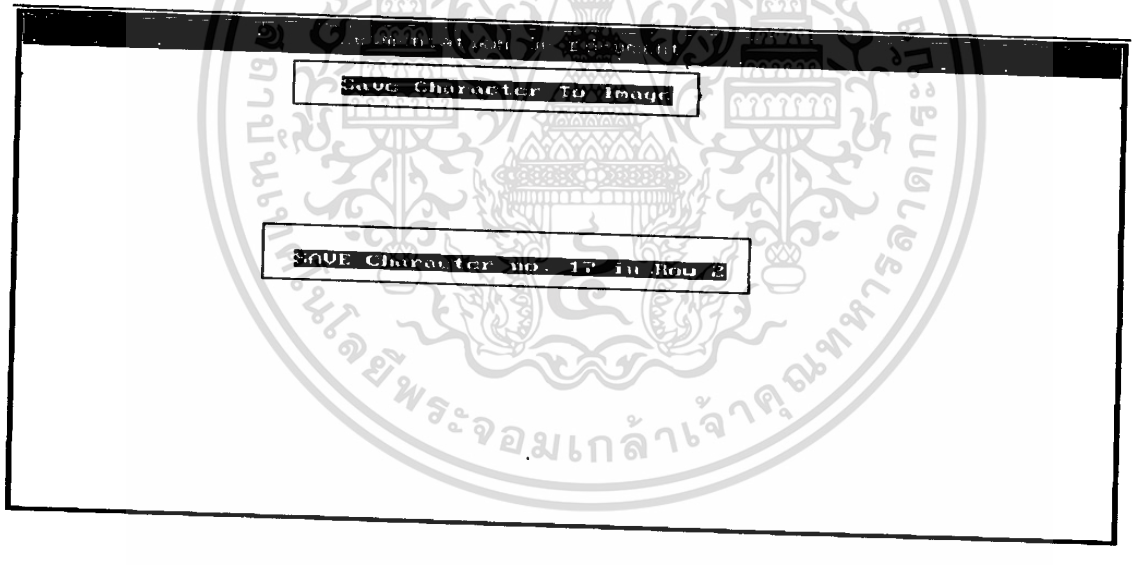
รูปที่ 4.4 แสดงรูป ในขั้นตอนการทำงานในการระบุส่วนประกอบในหน้าเอกสาร



รูปที่ 4.5 แสดงรูป ในขั้นตอนการแยกตัวอักษรออกทีละตัว



รูปที่ 4.6 แสดงรูป ในขั้นตอนการแยกตัวพยัญชนะและสระออกจากตัวอักษร



รูปที่ 4.7 แสดงขั้นตอนการเรียงตัวอักษรและเก็บตำแหน่งของตัวอักษร

ขั้นที่ 3. การแสดงผล

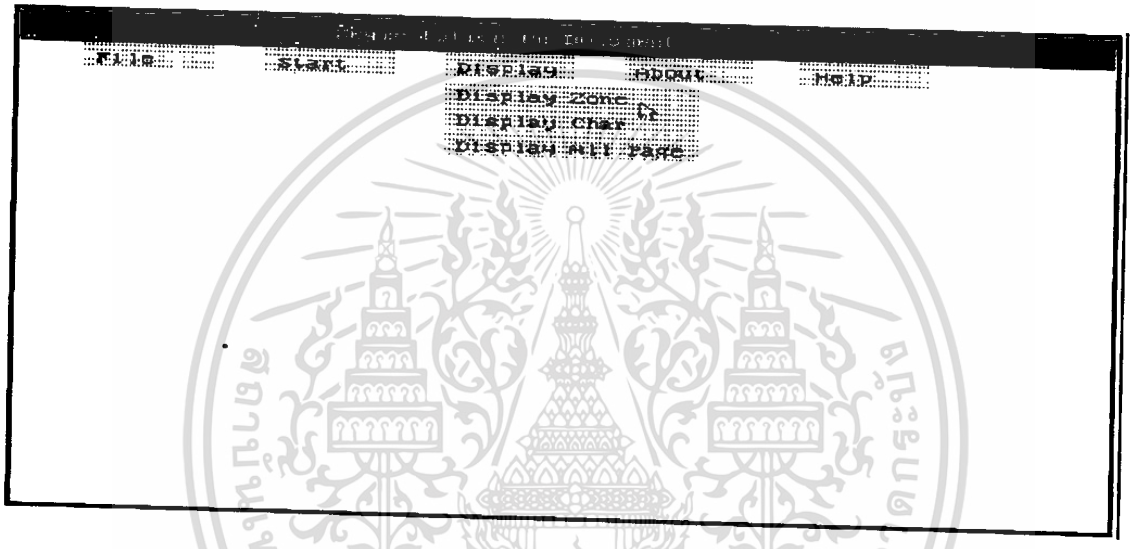
เลือกเมนู Display ซึ่งจะมีฟังชั่นให้เลือก 3. ฟังชั่น คือ

1. Display Zone โดยจะเป็นการแสดงผลการระบุส่วนประกอบในหน้าเอกสาร ซึ่งสามารถเลือกดูผลการระบุส่วนประกอบในแต่ละส่วนของหน้าเอกสารได้ ดังแสดงในรูปที่ 4.9

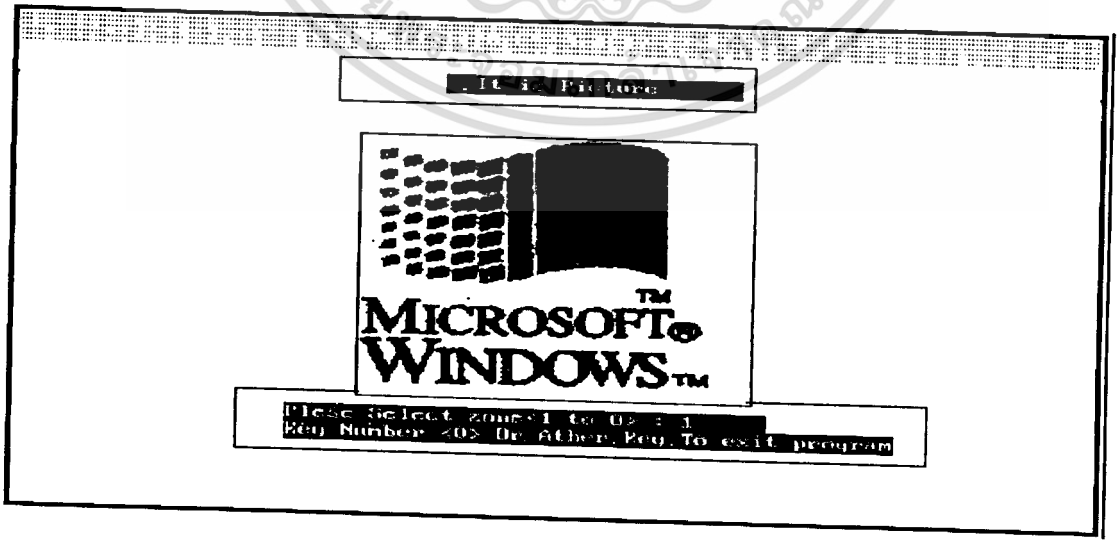
และ 4.10 โดยในรูปที่ 4.9 จะแสดงผลการระบุส่วนประกอบในหน้าเอกสาร ในกรณีที่เป็นรูปภาพ ส่วนในรูปที่ 4.10 จะแสดงในกรณีที่เป็นกลุ่มตัวอักษร

2. Display Char โดยเป็นการแสดงผลการแยกตัวอักษรออกทีละตัวออกจากประโยค

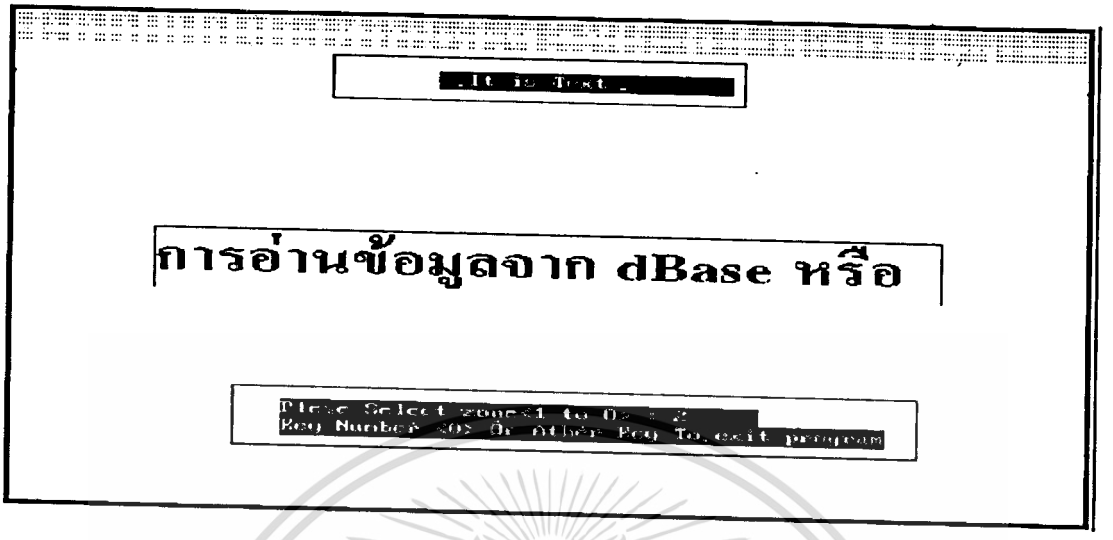
3. Display All Page โดยเป็นการแสดงผลการเรียงตัวอักษร ของแต่ละส่วนประกอบ



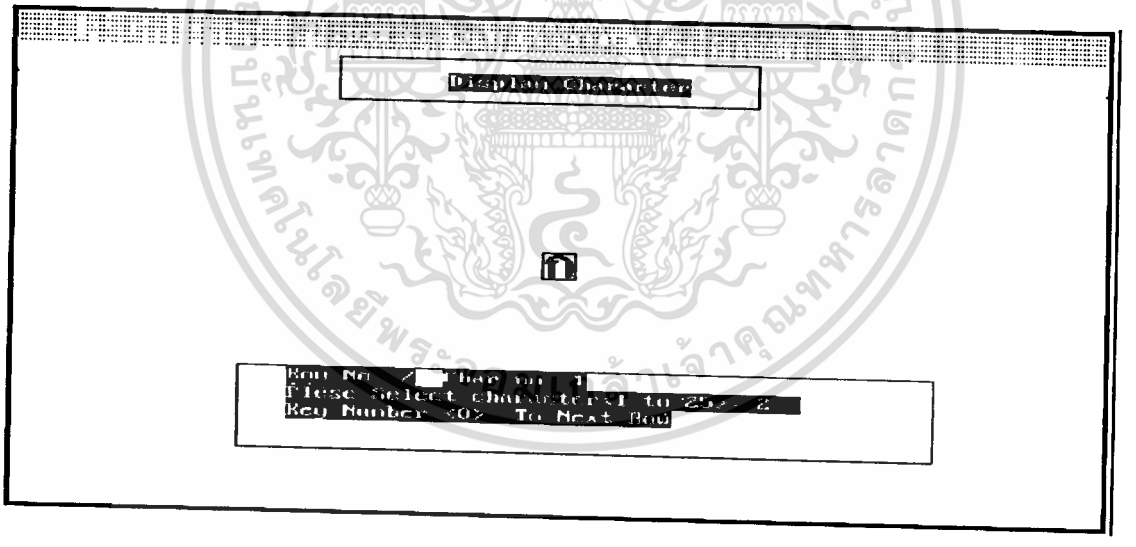
รูปที่ 4.8 แสดงภาพเมนู Display เพื่อดูผลลัพธ์ของการทำงานของโปรแกรม



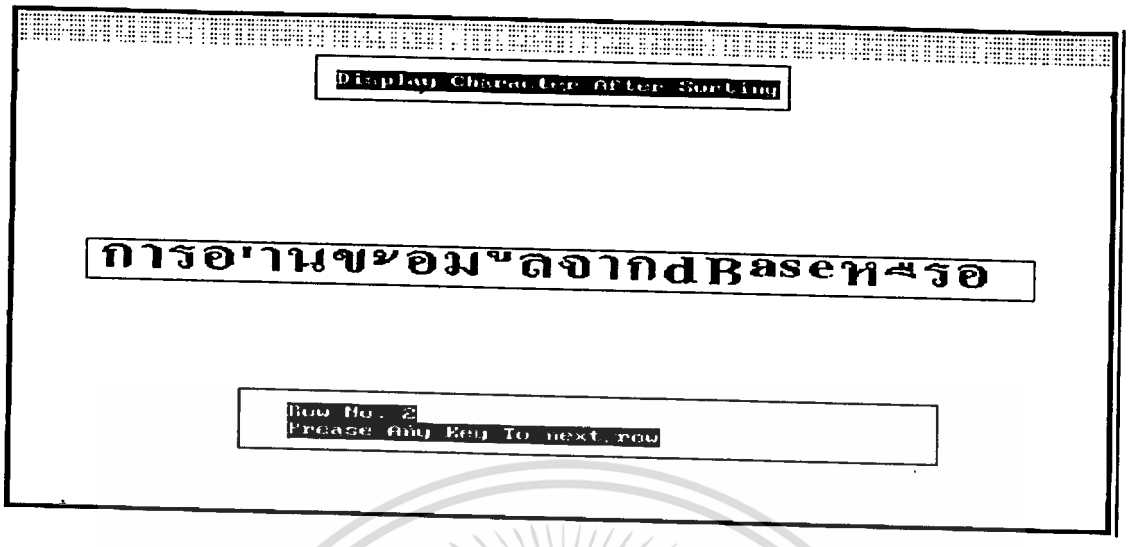
รูปที่ 4.9 แสดงผลการระบุส่วนประกอบในหน้าเอกสาร ในกรณีที่เป็นรูปภาพ



รูปที่ 4.10 แสดงผลการระบุส่วนประกอบในหน้าเอกสาร ในกรณีที่เป็นกลุ่มตัวอักษร



รูปที่ 4.11 แสดงผลการแยกตัวอักษรออกทีละตัว



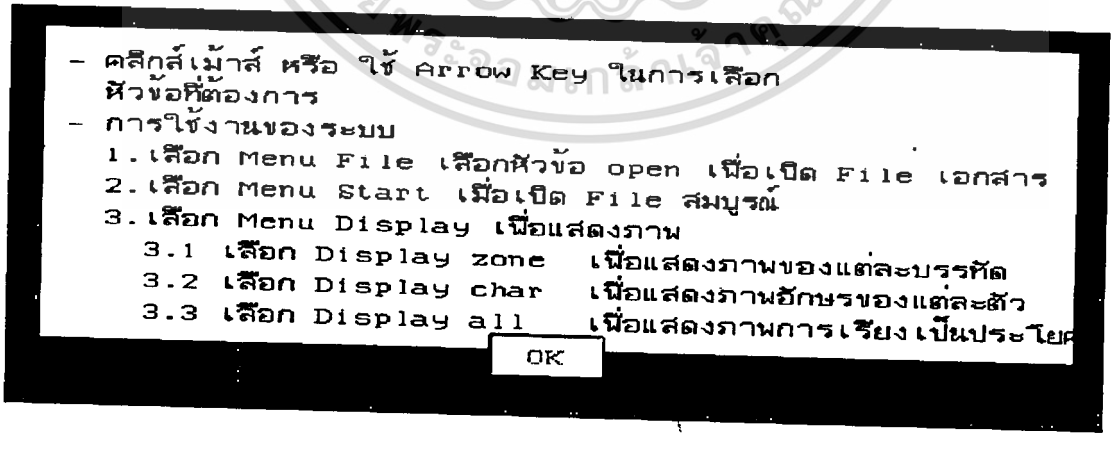
รูปที่ 4.12 แสดงรูปตัวอักษรที่ทำการเรียงแล้ว

ขั้นที่ 4. การออกจากการทำงานของโปรแกรม

โดยการเลือกเมนู File เลือกฟังชั่น Exit โปรแกรมจะสิ้นสุดการทำงาน และออกจากโปรแกรมไปที่ดอสพรอมท์

การใช้ Help

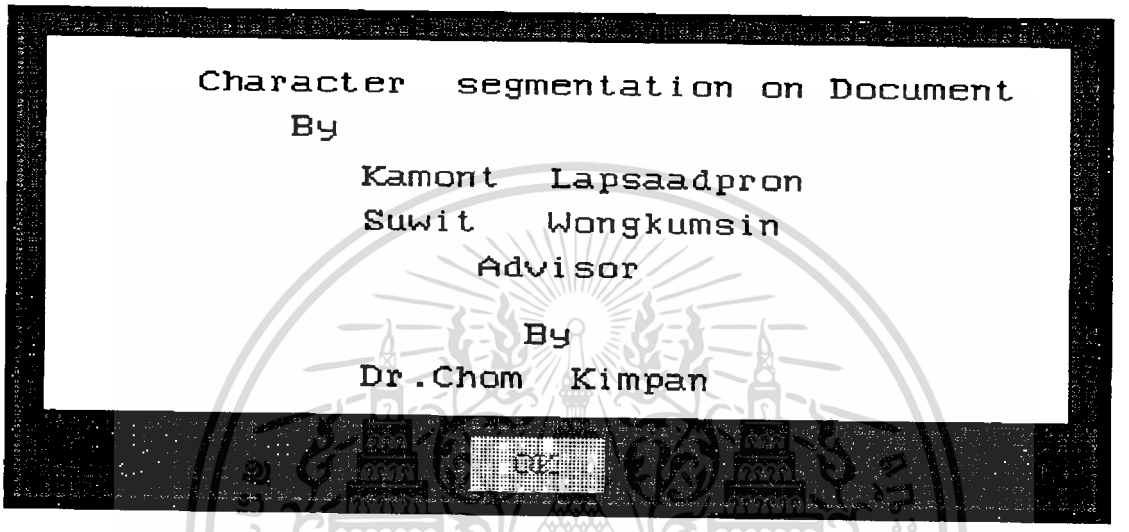
ใช้คีย์บอร์ดหรือเมาส์เลือกไปที่เมนู Help จะปรากฏหน้าต่างที่อธิบายการใช้งานของโปรแกรม



รูปที่ 4.13 แสดงหน้าต่างที่ช่วยสอนการใช้งานของโปรแกรม

การใช้ About

ใช้คีย์บอร์ดหรือเมาส์เลือกไปที่เมนู About จะปรากฏหน้าต่างที่บอกถึงผู้จัดทำโปรแกรม



รูปที่ 4.14 แสดงเมนู About

4.3 การทดสอบการทำงานของโปรแกรม

4.3.1 การทดสอบการระบุหน้าเอกสาร

โดยเป็นการทดสอบหน้าเอกสารที่มีรูปภาพกับประโยคตัวอักษรที่อยู่รวมกัน และเป็นการทดสอบหน้าเอกสารที่ประโยคตัวอักษรในขนาดของตัวอักษรที่แตกต่างกันและตัวอักษรต่างประเภท ซึ่งจะได้ผลการทดสอบดังตาราง 4.1-3

ชื่อแบบตัวอักษร	ขนาดตัวอักษร	ผลการทดสอบ
AngsanaUPC	18	ถูกต้อง
CordiaUPC	18	ถูกต้อง

ตารางที่ 4.1 แสดงผลการทดสอบโดยใช้ประเภทของตัวอักษรต่างชนิดกัน

ชื่อแบบตัวอักษร	ขนาดตัวอักษร	ผลการทดสอบ
AngsanaUPC	20	ถูกต้อง
CordiaUPC	24	ถูกต้อง

ตารางที่ 4.2 แสดงผลการทดสอบโดยใช้ประเภทและขนาดของตัวอักษรต่างชนิดกัน

ประเภทส่วนประกอบ	ขนาดของส่วนประกอบ	ผลการทดสอบ
รูปภาพ	-	ถูกต้อง
ประโยคตัวอักษร	20	ถูกต้อง
ประโยคตัวอักษร	16	ถูกต้อง

ตารางที่ 4.3 แสดงผลการทดสอบโดยใช้ประเภทและขนาดของตัวอักษรต่างชนิดกัน และมีรูปภาพบนหน้าเอกสารด้วย

4.3.2 การทดสอบการแยกตัวอักษร

ในการทดสอบ จะทดสอบการแยกตัวอักษรที่มีระดับแตกต่างกันไป ดังนี้

1. ระดับบนสุด ประกอบด้วย วรรณยุกต์ และตัวการันต์
2. ระดับกลาง ประกอบด้วย พยัญชนะ และสระระดับบนและกลาง
3. ระดับล่างสุด ประกอบด้วย สระระดับล่าง

ผลการทดสอบสามารถแยกตัวอักษรได้ถูกต้องเกือบ 100% ยกเว้นตัวอักษรที่ติดกัน จะไม่สามารถแยกออกจากกันได้

4.3.3 การทดสอบการเรียงตัวอักษร

เป็นการทดสอบการเรียงตัวอักษรที่มีระดับแตกต่างกันโดยการทดสอบจะทดสอบตัวอักษรที่มี สระ วรรณยุกต์ และตัวการันต์ต่างๆ

ผลการทดสอบ การเรียงตัวอักษรสามารถเรียงได้อย่างถูกต้อง

บทที่ 5
บทสรุป

5.1 สรุปผลการทำงาน

5.1.1 หน้าเอกสารที่ใช้ในการวิเคราะห์

การจัดเก็บหน้าเอกสารให้อยู่ในรูปแบบที่เครื่องคอมพิวเตอร์สามารถนำไปประมวลผลได้นั้นทำโดยการผ่านเครื่องตรวจกวาดภาพ จากนั้นข้อมูลหน้าเอกสารจะถูกแปลงให้เป็นข้อมูลแบบไบนารี ซึ่งหนึ่งจุดภาพในหน้าเอกสารจะถูกเปลี่ยนให้เป็นหนึ่งไบท์

จากการที่หน้าเอกสารขนาด A4 นั้นมีขนาดประมาณ 8"*11" เมื่อทำการตรวจกวาดภาพ โดยใช้เครื่องตรวจกวาดภาพความละเอียดของภาพประมาณ 300 จุดต่อนิ้ว จะทำให้ขนาดความกว้างของภาพเท่ากับ 2,400 จุด ความสูงเท่ากับ 3,300 จุด ทำให้หน้าเอกสารจะประกอบด้วยจุดภาพประมาณ 6,920,000 จุด ทำให้ต้องมีการลดความละเอียดของจุดภาพลงมา เพราะการที่ข้อมูลภาพมีขนาดใหญ่จะทำให้ผลการประมวลผลนั้นใช้เวลานานขึ้น

5.1.2 การวิเคราะห์หน้าเอกสาร

ในการวิเคราะห์หน้าเอกสาร จะใช้ทฤษฎีหลักๆ 2 ทฤษฎี คือ ทฤษฎีเซลลูล่าออโตมาต้า และทฤษฎีการเดินตามขอบของภาพ ในทฤษฎีเซลลูล่าออโตมาต้านั้นจะใช้ระบุส่วนประกอบหน้าเอกสารออกเป็นส่วนๆ ส่วนทฤษฎีการเดินตามขอบภาพ จะใช้ในการแยกส่วนประกอบของหน้าเอกสารแต่ละส่วนออกมา ในส่วนของการระบุประเภทของหน้าเอกสาร จะใช้หลักการเปรียบเทียบความสูงของรูปภาพกับภาพตัวอักษร

ในการประมวลผลหน้าเอกสารจะทำการประมวลผลที่ละจุดภาพ จึงทำให้การประมวลผลช้าและยังต้องมีการประมวลผลซ้ำซ้อนอีก ในบางขบวนการยังทำให้การประมวลผลจะช้าลงไปอีกมาก ซึ่งมีวิธีการแก้ปัญหาคือ การเพิ่มประสิทธิภาพของเครื่องไมโครคอมพิวเตอร์ให้สูงขึ้น เช่น การใช้หน่วยประมวลกลางให้มีความเร็วสูงขึ้น

5.2 ข้อจำกัดในการใช้งาน

1. หน้าเอกสารที่ใช้ต้องเป็นภาพขาว-ดำ

ในภาพหน้าเอกสารที่จะนำมาวิเคราะห์นั้นจะต้องตรวจกวาดมาในรูปแบบของภาพขาว-ดำ สาเหตุที่จะต้องจำกัดเป็นภาพขาวและดำ หรือการเก็บหน้าเอกสารโดยผ่านเครื่องตรวจกวาดภาพให้อยู่ในรูปภาพขาว-ดำ เพราะว่าถ้าเป็นภาพสีจะทำให้การวิเคราะห์หรือการ

ประมวลผลจะใช้เวลามากกว่าภาพขาว-ดำ เนื่องจากในการเปลี่ยนแฟ้มข้อมูลหน้าเอกสารที่เป็นภาพสีมาเป็นภาพขาว-ดำ จะทำให้ขนาดของข้อมูลมีขนาดเล็กลงไปมากซึ่งจะทำให้เวลาในการประมวลผลลดลงไปมากเช่นกัน

2. จำนวนส่วนประกอบในภาพมีน้อย

ในกรณีภาพหน้าเอกสารมีเพียงบรรทัดเดียวหรือไม่กี่บรรทัดอาจจะทำให้การระบุประเภทของหน้าเอกสารมีความผิดพลาดได้ เนื่องจากจะทำให้มีข้อมูลภาพที่ใช้ในทำการวิเคราะห์และประมวลผลไม่เพียงพอ ซึ่งจะทำให้การวิเคราะห์อาจเกิดความผิดพลาดได้

3. ภาพหน้าเอกสารมีสัญญานรบกวนมาก

ภาพที่มีสัญญาณรบกวนมาก จะทำให้การระบุประเภทของตัวอักษร และการแยกตัวอักษรผิดได้ เนื่องจากโปรแกรมอาจทำการระบุว่าสัญญาณรบกวนที่มีอยู่นั้นเป็นภาพของตัวอักษรได้

4. ภาพหน้าเอกสารที่ไม่สมบูรณ์

ในกรณีที่ภาพหน้าเอกสารที่ไม่สมบูรณ์ จะทำให้การวิเคราะห์มีความผิดพลาดเช่นเดียวกัน ซึ่งจะทำให้การแยกตัวอักษรไม่สมบูรณ์ ลักษณะของภาพเอกสารที่ไม่สมบูรณ์ก็คือตัวอักษรมีความไม่ต่อเนื่องหรือการขาดหายไปในบางส่วนของตัวอักษร

5.3 แนวทางในการพัฒนาในอนาคต

ในระบบวิเคราะห์และระบุส่วนประกอบหน้าเอกสารนี้ถึงแม้ว่าจะมีความสามารถในการวิเคราะห์หน้าเอกสารได้ถูกต้อง แต่ก็ยังอยู่ในระดับหนึ่งเท่านั้นในการใช้งาน เพื่อเป็นการเพิ่มขีดความสามารถในการใช้งานของระบบวิเคราะห์หน้าเอกสาร จึงควรที่จะมีการพัฒนาระบบดังนี้

1. การสามารถวิเคราะห์หน้าเอกสารที่เป็นภาพสีได้ ซึ่งจะทำให้การใช้งานมีประสิทธิภาพเพิ่มมากขึ้น

2. การเพิ่มประสิทธิภาพในส่วนของการกำจัดสัญญาณรบกวน ที่ปนมากับหน้าเอกสารจะทำให้การวิเคราะห์และการระบุส่วนประกอบของภาพมีความถูกต้องมากยิ่งขึ้น

3. สามารถที่จะวิเคราะห์หน้าเอกสารที่เป็นตารางได้ จะทำให้ระบบการวิเคราะห์และระบุส่วนประกอบในหน้าเอกสารมีความสมบูรณ์ครอบคลุมหน้าเอกสารได้ทุกประเภท ซึ่งมีผลทำให้ระบบการเรียนรู้และจดจำตัวอักษร มีความหลากหลายในการใช้งาน

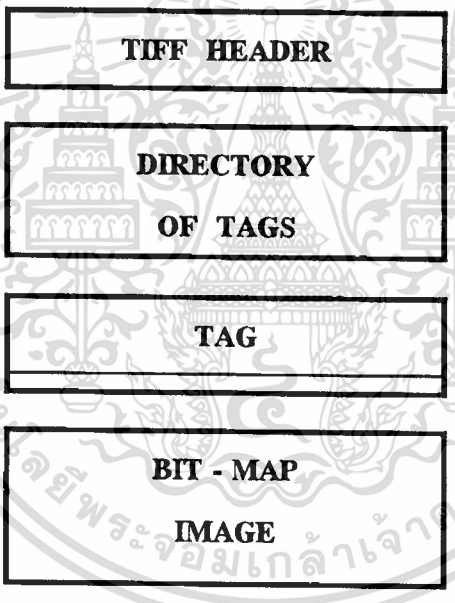
4. การประมวลผลให้เร็วขึ้นโดยการปรับปรุงอัลกอริทึมที่ใช้ให้มีการประมวลผลจุดภาพที่ซ้ำกันให้ลดน้อยลง

จากปริญญานิพนธ์นี้อาจจะมีบางส่วนที่ระบบการการวิเคราะห์และระบุส่วนประกอบในหน้าเอกสารยังขาดอยู่ จึงหวังว่าผู้วิจัยท่านอื่นที่มีความสนใจในเรื่องนี้ จะได้แนวทางในการพัฒนาระบบให้มีความสมบูรณ์มากขึ้น ซึ่งจะทำให้สามารถนำไปใช้งานได้อย่างมีประสิทธิภาพมากที่สุด

ภาคผนวก ก.

ข้อมูลภาพกราฟิกรูปแบบ TIFF จากเครื่องตรวจกวาดภาพ

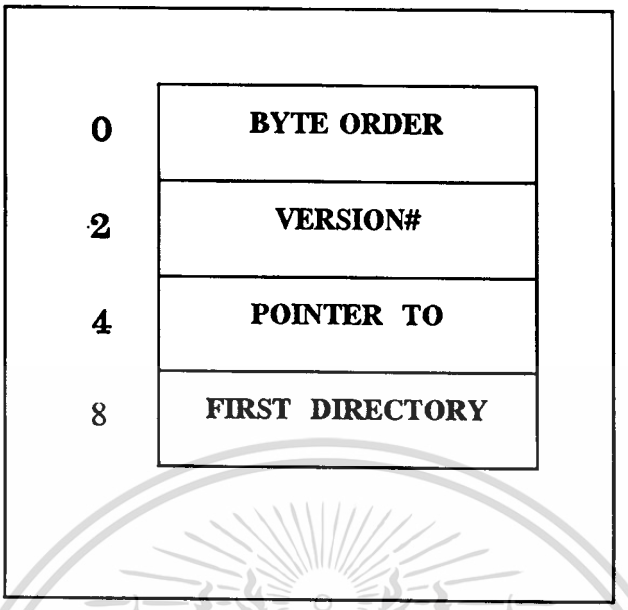
ข้อมูลภาพที่ได้จากเครื่องตรวจกวาดภาพ (Image Scanner) ของ Hewlett Packard (Scanjet Scanner) ซึ่งมีลักษณะข้อมูลที่เก็บในแฟ้มข้อมูลมีลักษณะที่แตกต่างจากเครื่องตรวจกวาดภาพชนิดอื่น โดยมีลักษณะการเก็บข้อมูลในแฟ้มข้อมูลในรูปของ TIFF file ซึ่งสามารถอธิบายโครงสร้างทั่วๆ ไปของ TIFF file เพื่อให้เห็นภาพแฟ้มข้อมูลโดยง่าย โดยสามารถแบ่งโครงสร้างออกได้เป็น 4 ส่วนด้วยกัน ดังแสดงในรูปที่ 1



รูปที่ 1 แสดงรายละเอียดโครงสร้างของ TIFF File

1. Details of header section

เป็นส่วนแรกที่ปรากฏอยู่ใน TIFF file และเป็นโครงสร้างที่สำคัญ โดยในส่วนนี้จะมีความยาว 8 ไบท์ ซึ่งจะบอกรายละเอียดของข้อมูลเบื้องต้น ในลักษณะของลำดับสองไบท์ โดยแสดงรายละเอียดของรูปแบบการเก็บข้อมูล (BYTE ORDER) ตัวเลขแสดงรุ่นของข้อมูล TIFF Version (VERSION#) และข้อมูลแสดงตำแหน่งแรกของการแสดงรายละเอียด ของข้อมูลเข้า (POINTERTO) ดังแสดงในรูปที่ 2



รูปที่ 2 แสดงรายละเอียด TIFF Header

Tif file มี headea ดังนี้

```
typedef struct{
        unsigned int numbertype ;
        unsigned int version ;
        unsigned long offset ;
        }TIFFHEAD;
```

โดยถ้าเราอ่าน 8 byte แรก เราจะได้ numbertype คือ 4949H หรือ 4D4DH ถ้าได้ 4949H จะเป็น file ที่ใช้ของ Intel number ถ้าเป็น 4D4DH จะเป็น motorola number

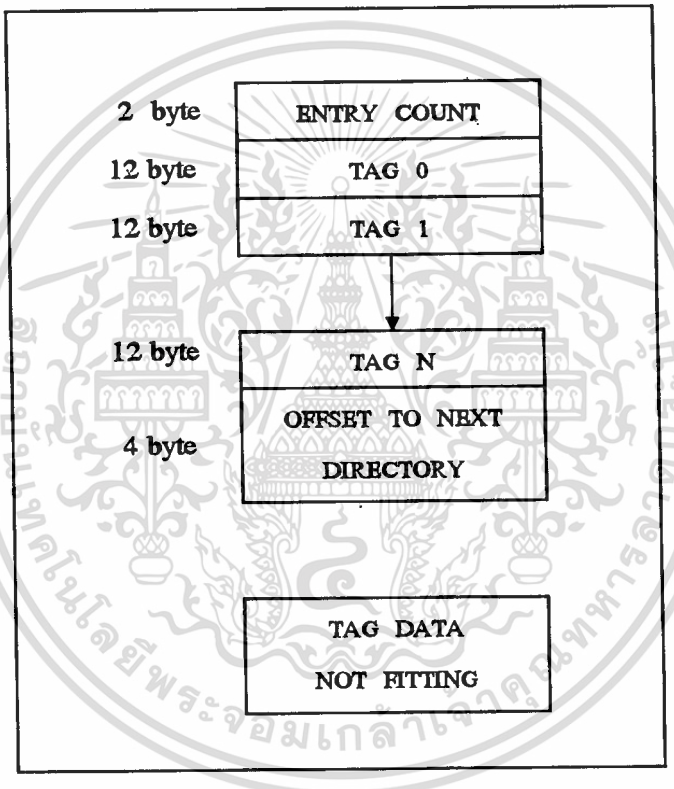
โดยที่ Version จะเป็น 42H เสมอ

Offset จะเป็นระยะห่างจากจุด start ของ file ไปยัง image file directory

2. Details of the Directory

เป็นส่วนของข้อมูลที่แสดงหัวรายละเอียดของข้อมูล สามารถแบ่งออกได้เป็น 3 ส่วนด้วยกันคือ ส่วนที่หนึ่งบอกด้วยสองไบท์แรก เป็นส่วน Directory บอกรายละเอียดเกี่ยวกับจำนวนของ Tag (รายละเอียดหลักของข้อมูลภาพ) ส่วนที่สองเป็นส่วนที่แสดงรายละเอียดของข้อมูลในแต่ละ Tag ซึ่งประกอบด้วยข้อมูลแสดงรายละเอียดของแต่ละ Tag จำนวน 12 byte เป็นส่วนของการ

บอกรายละเอียดข้อมูลของภาพ เช่น ขนาดความกว้างของภาพ ความยาวของภาพ เทคนิคการเก็บภาพ ลักษณะการเก็บข้อมูลภาพ เป็นต้น ตามจำนวนของ Tag ซึ่งในแต่ละ Tag จะบอกรายละเอียดของภาพหนึ่งอย่างตามรูปแบบการเก็บของข้อมูลภาพ และส่วนที่สามเป็นส่วนท้ายที่แสดงค่า offset ซึ่งจะเป็นตัวชี้ต่อไปยังส่วนของรายละเอียดอื่นๆ ซึ่งถ้าไม่มีส่วนของรายละเอียดถัดไปก็จะแสดงค่าเป็นศูนย์ ดังรูปที่ 3



รูปที่ 3 แสดงรายละเอียดของ Directory Section

image directory จะเริ่มต้นด้วย integer มี define จำนวนของ tags ใน directory แต่ละ tag จะมีความยาว 12 byte

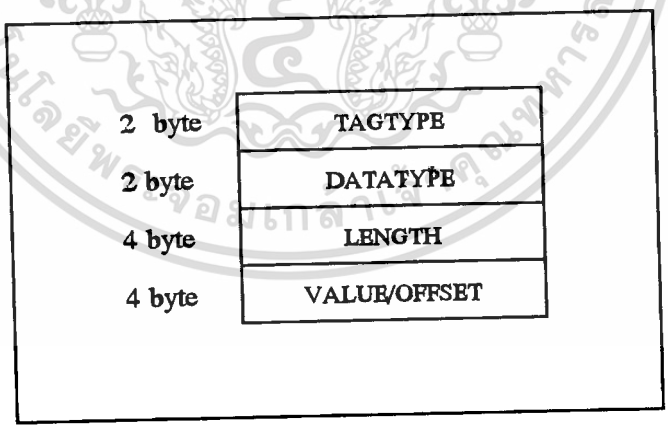
```
typedef struct{
        unsigned int tag ;
        unsigned int type ;
        unsigned long length ;
```

```
    unsigned long offset ;
}TIFFTAG ;
```

ถ้า Tag number คือ 256 tag นี้จะเรียกว่า ImageWidth และ define width ใน PIXED ของ image ที่จะ unpacked

3. Organization of each Tag Entry

องค์ประกอบโครงสร้างของแฟ้มข้อมูลในส่วนของแต่ละ Tag จะประกอบด้วยข้อมูลยาว 12 byte สองไบท์แรกแสดง Tag ตำแหน่งอ้างอิงของข้อมูลหลัก (TAGTYPE) ส่วนสองไบท์ถัดไป (ไบท์ 3 และไบท์ 4) แสดงถึงชนิดของข่าวสาร (DATATYPE) เช่น ถ้าสองไบท์นั้นมีค่าเป็น 282 (11AH) จะบอกรายละเอียดของความละเอียดในการสแกนตามแกน X ของข้อมูลภาพ ส่วนอีก 4 ไบท์ถัดไป (ไบท์ 5,6,7 และ 8) จะแสดงค่าความยาวของข้อมูล (LENGTH) ส่วนอีก 4 ไบท์ถัดไป (ไบท์ 9,10,11 และ 12) จะแสดงค่าของข้อมูล (VALUE/OFFSET) เช่น ถ้าในส่วนของ DATATYPE มีค่าเป็น 100H และส่วนของ VALUE มีค่าเป็น 0AH ก็จะได้ขนาดความกว้างของข้อมูลภาพมีขนาด 10 จุดภาพ เป็นต้น ดังแสดงในรูปที่ 4



รูปที่ 4 แสดงรายละเอียดข้อมูลแต่ละ Tag

Type field จะแสดงลักษณะประเภทของ number ที่ใช้ในการแสดงลักษณะของ Intermation ใน tag ปัจจุบัน ประเภทของ data ที่ define สำหรับ tag มี 5 ประเภทคือ

1. define data เป็น byte
2. define string ของ ASCII charecters
3. define data แบบ unsigned integer
4. define data แบบ unsigned long integer
5. define data ที่ rational number ของ two long integer

lenght field จะ define ความยาวของ data ใน current tag แต่อย่างไรก็ตามไม่ได้ define ความยาว lenght เป็น byte แต่จะ define มันเป็น Unit ของ specified by tag ดังนั้น lenght value ของ tag ที่ specified 1 integer จะเป็น 1 สำหรับ 1 integer, length value สำหรับ 500 byte string จะเป็น 1 สำหรับ 1 string

offset value ของ tag จะ define information ของ tag ถ้า length element specified data ที่จะใส่เข้าไปใน 4 byte ขนาดที่แท้จริงของ offset element data จะถูกเก็บไว้ใน offset element อีกทางเลือกหนึ่ง มันจะเก็บไว้ในบางส่วนของ TIFF file โดยได้ตำแหน่งจาก offset field ที่มีความสัมพันธ์ไปสู่จุดเริ่มต้นของ fiele

NewSubfileType จะกำหนดคุณสมบัติของ Image

ImageWidth จะกำหนดความกว้างของรูปภาพใน pitels

ImageDepth จะกำหนดความลึกของรูปภาพใน pitels

BitsPesSample จะกำหนดจำนวนของ bit ต่อ pitels

strip offsets specified

Compression tag

1. no Compression
2. CCITT group 3 Huffman encoding
3. CCITT group 3 Fax compression
4. CCITT group 4 Fax compression
5. LZW string table compression

32773 - macintosh PackBits run - length encoding

4. Bit - Map Image

เป็นส่วนเฉพาะข้อมูลที่มีการจัดเก็บตามลักษณะที่ได้กำหนดไว้ในรูปของข้อมูลแบบ binary ตามในส่วนของ Directory Section ที่ได้กล่าวมาแล้วซึ่งจากโครงสร้างที่ได้กล่าวแล้วนั้น จะทำให้เข้าใจโครงสร้างลักษณะของข้อมูลต่างๆ การจัดเก็บตลอดจนถึงการอ่านข้อมูลออกมาใช้งานอย่างถูกต้อง

ภาคผนวก ข.

รูปแบบงานวิจัยการเรียนรู้และจดจำตัวอักษรที่ผ่านมา

จากการวิจัยเรื่องการรู้จำตัวอักษรที่ผ่านมา วิธีการตัดสินใจสำหรับการรู้จำตัวอักษรส่วนใหญ่เป็นวิธีทางด้านสถิติ และวิธีการที่มีการใช้มากแบ่งออกได้เป็นสามกลุ่ม คือ การเทียบรูปแบบ (Pattern maching) หรือการเปิดตาราง (Table lookup) การเทียบความเหมือน (Corelation) และการตัดสินด้วยสมการเส้นตรง (Linearly equation decision)

แบบแรกเป็นการเทียบรูปแบบหรือการเปิดตาราง โดยนำต้นแบบตัวอักษรมาทำการวิเคราะห์หาคุณสมบัติต่างๆ แล้วนำไปสร้างเป็นตารางเก็บไว้ และเมื่อมีอินพุทป้อนเข้ามาจะนำคุณสมบัติของตัวอักษรที่เป็นอินพุทตัวนั้นมาทำการเปรียบเทียบกันในตาราง ถ้าตัวอักษรนั้นมีคุณสมบัติตรงกับอักษรตัวใดในตาราง ก็จะให้สรุปว่าเป็นอักษรตัวนั้น การออกแบบตารางที่ซับซ้อนขึ้นจะพิจารณาว่าอักษรหนึ่งตัวอาจมีต้นแบบหลายแบบซึ่งจะมีคุณสมบัติต่างกันออกไป จึงต้องทำการสร้างเป็นกลุ่มของข้อมูลต้นแบบตัวหนึ่งขึ้นมา และถ้านำเอาคุณสมบัติของอักษรที่ไม่รู้จักมาเทียบตรงกับตัวใดตัวหนึ่งในกลุ่มก็จะสรุปว่าเป็นอักษรตัวนั้น วิธีในการเทียบตัวอักษรนี้มีข้อจำกัดที่สำคัญ 2 ข้อคือ ลักษณะหรือคุณสมบัติที่ใช้จะต้องมีความคงที่ และผลลัพธ์การทำงานที่ได้ไม่มีค่าที่บ่งถึงระดับความเหมือนระหว่างต้นแบบกับอินพุท

แบบที่สองคือการเทียบความเหมือน เป็นการนำเนื้อของตัวอักษรทั้งตัวมาเทียบ (โดยการนำมาลบกัน หรือค่าเปรียบเทียบในรูปของสมการต่างๆ) กับต้นแบบจำนวนมากเพื่อหาว่า ต้นแบบใดมีความแตกต่างจากอินพุทน้อยที่สุด (โดยการเรียงลำดับหาค่าสูงสุด ต่ำสุด) โดยจัดกลุ่มของตัวอย่างให้อยู่ในกลุ่มของตัวต้นแบบที่มีความแตกต่างน้อยที่สุด วิธีนี้มีปัญหาในการคำนวณถ้าเนื้ออักษรใหญ่จะมีจำนวนตัวอย่างจำนวนมากจะใช้เวลาคำนวนมาก แต่จะได้ผลลัพธ์ที่บอกให้รู้ด้วยว่าค่าที่บ่งระดับความเหมือนระหว่างอินพุทกับต้นแบบมีเพียงใด

แบบที่สามเป็นการตัดสินใจโดยการนำเอาคุณสมบัติ 2 อย่างมาสร้างเป็นกราฟของรูปแบบ แบบ 2 มิติ นำกลุ่มทั้งหมดมาพล็อตลงในนี้ แล้วสร้างสมการตัดสินใจแบบเส้นตรงขึ้นมาจำนวนหนึ่ง เพื่อหาทางแบ่งกลุ่มของตัวอย่างให้ออกเป็นกลุ่มเล็กๆ หลายกลุ่ม ในระดับการแบ่งกลุ่มพื้นฐานจะมีเพียง 2 กลุ่มที่พิจารณาได้ว่าใช้สมการเส้นตรงเส้นเดียวสามารถแบ่งกลุ่มรูปแบบออกเป็น 2 กลุ่ม แล้ววิเคราะห์หาพารามิเตอร์ของสมการเส้นตรงนั้น เมื่อมีอินพุทที่ไม่รู้กลุ่มป้อนเข้ามา จะนำเอาคุณสมบัติของอินพุทนั้นมาดูเพื่อว่าจะจัดในกลุ่มใด โดยใช้เส้นตรงที่กำหนดเป็นแนวทาง หรือใช้การคำนวนโดยการแทนค่าตำแหน่งของอินพุทลงในสมการของเส้นตรงที่ใช้ใน

การตัดสินใจ เพื่อหาว่าพิกัดของอินพุทอยู่บนหรือใต้เส้นตรงนั้น ทำให้เลือกกลุ่มได้ ปัญหาของวิธีการนี้คือ รูปแบบในธรรมชาติส่วนมากมีการจัดกลุ่มที่ไม่สามารถใช้สมการเส้นตรงตัดสินได้

ภาคผนวก ค.

ทฤษฎีฮิสโตแกรม

ขบวนการจำกัดขอบเขตบนหน้าเอกสารเพื่อทำการแยกตัวอักษรบนหน้าเอกสารอีกวิธีหนึ่งก็คือการใช้ฮิสโตแกรมในการหาขอบเขตของกลุ่มตัวอักษร โดยใช้คุณสมบัติของช่องว่างระหว่างบรรทัดและช่องว่างระหว่างตัวอักษร มาใช้ในการวิเคราะห์โดยมีหลักการดังนี้

1. หาฮิสโตแกรมของภาพในแนวนอน

2. ในระหว่างที่ทำการหาฮิสโตแกมในแนวนอน ถ้ามีค่าฮิสโตแกรมที่จุดใดที่มีการเปลี่ยนจากค่า 0 ไปเป็นค่า 1 ก็ให้สันนิษฐานว่าจุดนั้นเป็นจุดเริ่มต้นของเส้นบรรทัดบน

3. ถ้ามีค่าฮิสโตแกรมที่จุดใดที่มีการเปลี่ยนจากค่า 1 ไปเป็นค่า 0 ก็ให้สันนิษฐานว่าจุดนั้นเป็นจุดของเส้นบรรทัดล่าง

4. ถ้าเส้นระหว่างบรรทัดมีค่าที่ห่างพอ ก็จะทำการตัดบรรทัดนั้นออกไปเก็บอีกที่หนึ่ง

5. ทำการแยกส่วนต่างๆ ออกมาทีละบรรทัด จนหมดทั้งหน้าเอกสาร

จากเทคโนโลยีการพิวเตอร์ว่าควรจะ

รูปที่ 2.4 การตัดบรรทัดที่มีการเปลี่ยนค่าศูนย์

6. นำแถวที่ได้เป็นบรรทัด แล้วนำมาทำการตัดอักษรออกทีละตัว (character segmentation) โดยอาศัยฟังก์ชันฮิสโตแกรมของเซลในแนวตั้ง ในกรณีของตัวอักษรที่มีการกำหนดขนาดคงที่ปัญหาที่เกิดจะไม่ยุ่งยากนัก แต่กรณีการพิมพ์ตัวอักษรที่มีการปรับช่องว่างแบบไม่เท่ากันจะเกิดปัญหาที่มากยิ่งขึ้น

พวเตอรวาควรจะ

รูปที่ 2.5 การตัดตัวอักษรออกจากบรรทัด

ภาคผนวก ง.

โปรแกรมการทำงาน

MAIN_SEG.C

```
#inolude <stdio.h>
#inolude <oonio.h>
#inolude <graphios.h>
#inolude "mouse16.h"
#inolude "fex_e.h"

extern int m_init (void);
extern InitThai(void);
extern int Choice_head(int x,int y,int oount,ohar *sub[]);
extern int open1();
extern void putimag1();
extern int getzone();
extern int segment();
extern int out_row();
extern int dspzone();
extern int save();
extern void sound1();
extern void soundburst();

void UserInput();
void head_menu();
void help_menu(void);
void about_show();
int Choice_Menu(int x,int y,int oount,ohar *sub[]);
int ohoiceopen(int seleot);
int ohoiceopen1(int seleot);
void initial_menu();
void error_oocer(void);
void p1_getzone();
void p2_segment();
void p3_out_row();
void p4_save();

ohar *fileopen[]= {
          " Open ",
          " Exit ",
          };
ohar *dsp[]= {
          " Display Zone     ",
          " Display Char     ",
          " Display All Page ",
          };

ohar *ohoice[]= {
    " File    ",
    " Start   ",
    " Display ",
```

```
    " About   ",
    " Help    ",
    };

POINT p;
int o_char,color_red,color_brown;
int check,check_wall,color_mt,color_mb;
int color_wall,color_sel,color_bg;
int char_M,color_box;

main()
{
  int dr=DETECT,mo;

  initgraph(&dr,&mo,"");
  m_init();
  InitThai();
  check = 0;
  o_char = 1; char_M = 14;
  check_wall=1;                          /* Have Wallpaper */
  color_mt=LIGHTGREEN;
  color_mb=LIGHTRED;
  color_wall=13;
  color_box=0;
  color_sel=11;
  color_bg=2;
  color_red=6;
  color_brown=3;
  m_on();
  head_menu();
  UserInput();
}

void UserInput()
{
  int select,select2;
  int check_module;

/*  window(20,50,1000,350);*/
  setcolor(BROWN);
  settextstyle(1,0,3);
/*  outtextxy(70,65,"*** CONTENT ***");
  settextstyle(1,0,1);*/
  do {
      select = Choice_head(50,45,5,choice);
    switch(select) {
        case 0 :                                 /* Unit 1 */
            select2=Choice_Menu(50,70,2,fileopen);
          /*    clear_page(1);*/
          /*    RestoreWin();*/
            choiceopen(select2);
            break;
        case 1 :                                 /* Unit 2 */
            /**********get_zone*******/
            m_off();
```

```
                p1_getzone();
                p2_segment();
                p3_out_row();
                p4_save();
                m_on();
                break;
        case 2 :
                select2=Choice_Menu(250,70,3,dsp);
                clear_page(1);
                choiceopen1(select2);
                break;
        case 3 :
                about_show();
                break;
        case 4 :                                /* Unit 5 */
                help_menu();                                    /* Exit */
                break;

        default:
                break;
    }
  }while(check==0);
  RestoreWin();
  if(check==9) {
      closegraph();
      exit(0);
     }
  else
    return;
}

int Choice_Menu(int x,int y,int count,char *sub[])
{
  BUTTON exit,cancel,help,str[10];
  int t,i,j,top,old,st_top;
  int arow_choice,click;
  int positionx;
  top=st_top=y;
  arow_choice = click = old = 0;

  m_off();
  RestoreWin();
/*  MakeButton(40,310," Exit ",&exit);
  DrawButton(&exit);
  MakeButton(130,310,"Cancel",&cancel);
  DrawButton(&cancel);
  MakeButton(220,310," Help ",&help);
  DrawButton(&help);*/
  for(i=0,j=70;i<count;i++,j+=23) {
  outthaibg(x,j,color_wall,BLUE,sub[i]);
  MakeButton(x,j,"                    ",&str[i]);
  }
/*  outthaibg(50,,BLUE,WHITE,sub[0]);*/
  m_on();
/*  positionx=70;*/
```

```
  do {
     t=0;
/*     y=70;*/
     if(kbhit()) {
          key.i = bioskey(0);
          if(key.ch[1]==45) {                    /* ALT_X */
            arow_choice=8;
            t=1;
          }
          else if(key.ch[1]==72) {               /* AR_UP */
            m_off();
            top = y+(arow_choice*23);
            outthaibg(x,top,color_wall,BLUE,sub[arow_choice]);
            arow_choice--;
            if(arow_choice<0)
               arow_choice=count-1;
            top = y+(arow_choice*23);
            outthaibg(x,top,BLUE,WHITE,sub[arow_choice]);
            m_on();
          }
          else if(key.ch[1]==80) {               /* AR_DOWN */
            m_off();
            top = y+(arow_choice*23);
            outthaibg(x,top,color_wall,BLUE,sub[arow_choice]);
            arow_choice++;
            if(arow_choice>count-1)
               arow_choice=0;
            top = y+(arow_choice*23);
            outthaibg(x,top,BLUE,WHITE,sub[arow_choice]);
            m_on();
          }
          else if(key.ch[0]==13)                 /* ENTER */
            t=1;
          else if(key.ch[0]==27) {               /* ESC  */
            arow_choice=9;
            t=1;
          }
          else
            t=0;
     }
     if(Mousehit(&p)) {
     y=70;
          for(i=0;i<count;i++) {
            if(PointInRect(&p,&str[i])) {
               arow_choice=i;
               top=y+(arow_choice*23);
               m_off();
               outthaibg(x,top,BLUE,WHITE,sub[arow_choice]);
               m_on();
            }
          }
   /*     if(PointInRect(&p,&exit)) {            /* Exit */
     /*     TrackButton(&exit);
            arow_choice = 8;
            t=1;
```

```
        }
        if(PointInRect(&p,&cancel)) {          /* Canole */
    /*    TrackButton(&cancel);
          arow_choice =9;
          t=1;
        }
        if(PointInRect(&p,&help)) {            /* Help */
        /*    TrackButton(&help);
           help_menu();
           m_limit(20,50,300,350);
        } */
        if(old!=arow_choice) {
          m_off();
          outthaibg(x,st_top,color_wall,BLUE,sub[old]);
          old=arow_choice;
          st_top=top;
          click=0;
          m_on();
        }
        while(Mousehit(&p));
        if(PointInRect(&p,&str[arow_choice])) {
            click++;
            if(click>=2)
            t=1;
        }
    }
   }while(t==0);
   return(arow_choice);
   RestoreWin2();
}

void head_menu()
{
  m_off();
  setfillstyle(SOLID_FILL,WHITE);
  bar(LEFT,TOP,RIGHT,BOTTOM);
  setfillstyle(SOLID_FILL,BLUE);
  bar(LEFT+6,TOP+6,RIGHT-6,TOP+32);
  setfillstyle(SOLID_FILL,color_box);
  bar(LEFT+2,TOP+2,RIGHT-2,TOP+4);
  bar(LEFT+2,BOTTOM-4,RIGHT-2,BOTTOM-2);
  bar(LEFT+2,TOP+2,LEFT+4,BOTTOM-2);
  bar(RIGHT-4,TOP+2,RIGHT-2,BOTTOM-2);

  outthai(LEFT+180,TOP+7,WHITE,"Segmentation On Doument");
/*  MakeButton(LEFT+6,TOP+5," File ",&file);
  DrawButton(&file);
  MakeButton(RIGHT-73,TOP+5," About ",&about);
  DrawButton(&about);*/
  m_on();
  return;
}

void help_menu(void)
{
```

```
  BUTTON ab_ok;
  int t;
  char ch;

  m_off();
  win_2(100,150,550,400,30);
  MakeButton(300,350," OK ",&ab_ok);
  DrawButton(&ab_ok);
  outthai(120,170,char_M,"- คลิกส์เม้าส์ หรือ ใช้ Arrow Key ในการเลือก");
  outthai(120,190,char_M,"  หัวข้อที่ต้องการ");
  outthai(120,210,char_M,"- การใช้งานของระบบ");
  outthai(120,230,char_M,"  1.เลือก Menu File เลือกหัวข้อ open เพื่อเปิด File เอกสาร");
  outthai(120,250,char_M,"  2.เลือก Menu Start เมื่อเปิด File สมบูรณ์");
  outthai(120,270,char_M,"  3.เลือก Menu Display เพื่อแสดงภาพ");
  outthai(120,290,char_M,"    3.1 เลือก Display zone  เพื่อแสดงภาพของแต่ละบรรทัด");
  outthai(120,310,char_M,"    3.2 เลือก Display char  เพื่อแสดงภาพอักษรของแต่ละตัว");
  outthai(120,330,char_M,"    3.3 เลือก Display all   เพื่อแสดงภาพการเรียงเป็นประโยค");
  m_on();
  do {
    t=1;
    if(kbhit()) {
          ch = getch();
          if(ch==13 || ch==27)
            t=0;
          else
            t=1;
    }
    else if(MouseHit(&p)) {
          if(PointInRect(&p,&ab_ok)) {
              TrackButton(&ab_ok);
              t=0;
          }
          else
              t=1;
    }
  } while(t!=0);
  RestoreWin2();
  return;
}

void about_show()
{
  BUTTON ab_ok;
  int t;
  char ch;
  m_off();
  win_2(150,100,520,320,30);
  MakeButton(310,287," OK ",&ab_ok);
  DrawButton(&ab_ok);
  outthai(180,118,15,"    Character  segmentation on Document");
  outthai(180,138,11,"                                  By");
  outthai(180,160,11,"           Kamont  Lapsaadpron");
  outthai(180,180,11,"           Suwit  Wongkumsin");
  outthai(180,200,14,"              Advisor       ");
  outthai(180,230,14,"                By          ");
```

```
  outthai(180,250,14,"          Dr.Chom  Kimpan          ");
  m_on();
  do {
    t=1;
    if(kbhit()) {
          oh = getch();
          if(oh==13 || oh==27)                    /* ENTER or ESC */
            t=0;
          else
            t=1;
    }
    else if(Mousehit(&p)) {
          if(PointInRect(&p,&ab_ok)) {            /* Cancle */
              TrackButton(&ab_ok);
              t=0;
          }
          else
              t=1;
    }
  } while(t!=0);
  RestoreWin2();
  return;
}

int choiceopen(int select)
 {
          int check_module;
/*        select = Choice_Menu(20,45,2,*sub);*/
    do{
    switch(select) {
            case 0 :
                      m_off();
                      check_module=open1();
                      sound1();
                      if(check_module==0){
                      clear_page(1);
                      soundburst();
                      outthai(180,200,1,"มีความผิดพลาดเกิดขึ้น จะทำการออกจากโปรแกรมการทำงาน");
                      check=9;
                      getch();
                      }

                      clear_page(1);
                      if(check_module!=0) putimag1();
                      clear_page(1);
                      m_on();
                      m_on();
                      RestoreWin();
                      select=9;
                    break;
            case 1 :
                    select=9;
                    check=9;
                      break;
            default:
```

```
                    break;
    }
  }while(select!=9);
 }
int choiceopen1(int select)
 {
 int gdriver=DETECT,gmode;
 int check_module;
/*        select = Choice_Menu(20,45,2,*sub);*/
    do{
    switch(select) {
          case 0 :
                    m_off();
                    check_module=dspzone();
                    sound1();
                    if(check_module==0){
                    clear_page(1);
                    soundburst();
                    outthai(180,200,1,"มีความผิดพลาดเกิดขึ้น จะทำการออกจากโปรแกรมการทำงาน");
                    check=9;
                    getch();
                    }
                    m_on();
                    m_on();
                    clear_page(1);
                    RestoreWin();
                    select=9;

                    break;
          case 1 :
              m_off();
                    check_module=dspchr();
                    sound1();
                    if(check_module==0){
                    clear_page(1);
                    soundburst();
                    outthai(180,200,1,"มีความผิดพลาดเกิดขึ้น จะทำการออกจากโปรแกรมการทำงาน");
                    check=9;
                    getch();
                    }
                    m_on();
                    m_on();
                    clear_page(1);
                    RestoreWin();
                  select=9;
                    break;
          case 2 :
                    m_off();
                    check_module=dspall();
                    sound1();
                    if(check_module==0){
                    clear_page(1);
                    soundburst();
                    outthai(180,200,1,"มีความผิดพลาดเกิดขึ้น จะทำการออกจากโปรแกรมการทำงาน");
                    check=9;
```

```
                        getch();
                        }
                        m_on();
                        m_on();
                        clear_page(1);
                        RestoreWin();
                    select=9;
                    break;
            default:
                        break;
    }
  }while(select!=9);
 }

void error_occer()
{
                        clear_page(1);
                        soundburst();
                        outthai(180,200,1,"มีความผิดพลาดเกิดขึ้น จะทำการออกจากโปรแกรมการทำงาน");
                        check=9;
                        getch();
}

void p1_getzone()
{
int check_module;
/***********get_zone*******/
                clear_page(1);
                check_module=get_zone();
                sound1();
                if(check_module==0){
                                error_occer();
                                }
                delay(1500);
}
void p2_segment()
{
int check_module;
                /***********segment*******/
                clear_page(1);
                check_module=segment();
                sound1();
                if(check_module==0){
                                error_occer();
                                }
                delay(1500);
                clear_page(1);
}

void p3_out_row()
{
int check_module;
                    clear_page(1);
                check_module=out_row();
                sound1();
```

```
                if(check_module==0){
                        error_occer();
                        }
                delay(1500);
                clear_page(1);
                RestoreWin();
}

void p4_save()
{
int check_module;
                    clear_page(1);
                check_module=save();
                sound1();
                if(check_module==0){
                        error_occer();
                        }
                delay(1500);
                clear_page(1);
                RestoreWin();
}

void initial_menu()
{
  clear_page(1);

}
```

OPENTIF.C

```
#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <math.h>
#include <alloc.h>
#include <string.h>
#include <conio.h>
#include <graphics.h>
#include "read.h"
/*  #include "wtwg.h" */

#define farmemcpy(dest,src,num) \
        movedata(FP_SEG(src),FP_OFF(src),FP_SEG(dest),FP_OFF(dest),num)

    char *hlist;

  /* extern prototype */
  extern void beep(void);
/*  extern int open1(void);
  extern  putimag1(void);*/
  /* local variables */
   char masktable[8]={ 0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01 };
   unsigned int numbertype='II';
   FILEINFO fi;
/*   char path[40];*/
 /*  char pathmsg[40];*/
 /*  int priority;*/
   int wide,len;
  char *data,*temp;
  int gdriver=DETECT,gmode;
   FILE *fp;
   FILE *outfile;

  /* local prototype */
  void readtif();
  void strmfe(char *new,char *old,char *ext);
  int readheap(char *p,unsigned int n);
  unsigned long fgetlong(FILE *fp);
  unsigned int fgetword(FILE *fp);
  int unpacktiff(FILE *fp,FILEINFO *fi);
  int readline(char *p,FILE *fp,int bytes,FILEINFO *fi);
  void convertline(char *dest,char *source,FILEINFO *fi,FILE *tp);
  void setdefaults(FILEINFO *fi);
  void decodetag(FILEINFO *fi,FILE *fp);
  int open1(void);
  void get_row(FILE *tp,int wide);
  void save_f(void);
  int putimag1(void);

   FILE *tp;
/* Read TIFF routine start here */

  /* unpack a TIFF file */
   int unpacktiff(FILE *fp,FILEINFO *fi)
   {
```

```
        char *p,*pr;
        unsigned long l;
        int i,j,n;
        int run=0;

/*      wopen(wxabsmax-19,wyabsmax-1,19,1,112,1,112,1); */
/*      indicate=w0;                                    */
/*      wputs(" Please wait..."); */

        /*get the number types */
        numbertype=fgetword(fp);

        /* see fi it's real */
        if(numbertype=='II' || numbertype=='MM') {

                /* trash the verdion */
                fgetword(fp);

                /* seek through the image directories */
                while((l=fgetlong(fp)) != 0L) {

                        fseek(fp,l,SEEK_SET);

                        /* set default values */
                        setdefaults(fi);

                        /*decode all the tags */
                        n=fgetword(fp);
                        for(i=0;i<n;++i) decodetag(fi,fp);

                        /* save the file position */
                        l=ftell(fp);

                        /* figure the dits - limit only monochrome pictures */
                        fi->bits=fi->samples*fi->bitspersample;

                        /* figure the bytes */
                        if(fi->bits <= 8)
                                fi->bytes=pixels2bytes(fi->width)*fi->bits;
                        else fi->bytes=fi->width/8;
                        wide=fi->width;
                        len=fi->depth;
                        if(fi->width != 0 &&
                           fi->depth != 0 &&
                           fi->offset != 0L) {
                                if((p=malloc(fi->bytes))==NULL){
                                        clear_page(1);
                                        setcolor(RED);
                                        rectangle(170,120,450,170);
                                        gotoxy(24,10);
                                        outthai(180,135,1,"Error allocate pr in unpacktif()");
                                /*      printf("Error allocate pr in unpacktif()\n");*/
                                        return(0);
                                        }
                                if((pr=malloc(fi->width))==NULL){
```

```
                        clear_page(1);
                        setcolor(RED);
                        rectangle(170,120,450,170);
                        gotoxy(24,10);
                        outthai(180,135,1,"Error allocate pr in unpacktif()");
                        /*printf("Error allocate pr in unpacktif()\n");*/
                        return(0);
                        }
/*      (fi->setup)(fi);  */
/*      openfile(tp); */
                if ((tp=fopen("tiftext.txt","wb"))==NULL){
                clear_page(1);
                setcolor(RED);
                rectangle(170,120,450,170);
                gotoxy(24,10);
                outthai(180,135,1,"Can't open file name: tiftext.txt");
                /*printf(outthai(180,135,1,);"Can't open file name: tiftext.txt\n");*/
                return(0);
                }
                fseek(tp,0L,SEEK_SET);
                if(fi->count==1L) {
                        fseek(fp,fi->offset,SEEK_SET);
                        for(i=0;i<fi->depth;++i) {
                                if(readline(p,fp,fi->bytes,fi) !=fi->bytes) {
                                        pr=p=NULL;
                                        free(pr);
                                        free(p);

                                        return(BAD_READ);
                                }

                                convertline(pr,p,fi,tp);
                        }
                }
                else {
                        for(l=0;l<fi->count;++l) {
                                fseek(fp,fi->offset+(l*sizeof(long)),SEEK_SET);
                                fseek(fp,fgetlong(fp),SEEK_SET);
                                for(j=0;j<(int)fi->rowsperstrip;++j) {
                                        if((int)((l*fi->rowsperstrip)+j) >= fi->
depth) break;

                                        if(readline(p,fp,fi->bytes,fi) != fi->bytes) {
                                                pr=p=NULL;
                                                free(pr);
                                                free(p);

                                        /*      wheap_freeall();  */
                                                return(BAD_READ);
                                        }
                                        convertline(pr,p,fi,tp);
                                }
                        }
                }
                pr=p=NULL;
                free(pr);
```

```
                                free(p);

                        /*      wclear();  */
                        /*      wputs(" <Esc> to exit. "); */
                        /*      wbury();  */

                                fclose(tp);
                                return(GOOD_READ);
                                fseek(fp,1,SEEK_SET);
                        }
                }
                return(GOOD_READ);
        } else return(BAD_FILE);
}

void convertline(char *dest,char *source,FILEINFO *fi,FILE *tp)
{
        int i;
        char ch;
        switch(fi->bits) {
                case 8:
                        memcpy(dest,source,fi->width);
                        break;
                case 4:
                        for(i=0;i<fi->width;) {
                                dest[i++]=(*source>>4) & 0x0f;
                                dest[i++]=*source & 0x0f;
                                ++source;
                        }
                        break;
                case 1:
                        for(i=0;i<fi->width;++i) {
                                if(source[i>>3]&masktable[i&0x0007])
                                {
                                ch=0x00;
                                dest[i]=0x00;
                                }
                                else
                                 {
                                 ch=0x01;
                                 dest[i]=0xff;
                                 }
                                fwrite(&ch,sizeof(char),1,tp);
                        }
                        break;
        }
}

void setdefaults(FILEINFO *fi)
{
        int i;

        fi->width=0;
        fi->depth=0;
        fi->bits=0;
```

```
        fi->threshhold=1;
        fi->samples=1;
        fi->compression=1;
        fi->offset=0L;
}

void decodetag(FILEINFO *fi,FILE *fp)
{
        long length,offset,pos;
        int tag,type,i;

        tag=fgetword(fp);
        type=fgetword(fp);

        if(type == TIFFlong) {
                length=fgetlong(fp);
                offset=fgetlong(fp);
        }
        else {
                length=(unsigned long)fgetword(fp);
                fgetword(fp);
                offset=(unsigned long)fgetword(fp);
                fgetword(fp);
        }

        switch(tag) {
                case SubfileType:
                        break;
                case ImageWidth:
                        fi->width=(unsigned int)offset;
                        break;
                case ImageLength:
                        fi->depth=(unsigned int)offset;
                        break;
                case RowsPerStrip:
                        if(type==TIFFlong) fi->rowsperstrip=offset;
                        else fi->rowsperstrip=offset & 0xffffL;
                        break;
                case StripOffsets:
                        if(type==TIFFlong) fi->offset=offset;
                        else fi->offset=offset & 0xffffL;
                        fi->count=(int)length;
                        break;
                case StripByteCounts:
                        if(type==TIFFlong) fi->bytecount=offset;
                        else fi->bytecount=offset & 0xffffL;
                        break;
                case SamplesPerPixel:
                        fi->samples=(int)offset;
                        break;
                case BitsPerSample:
                        if(length > 1L) {
                                pos=ftell(fp);
                                fseek(fp,offset,SEEK_SET);
                                fi->bitspersample=fgetword(fp);
```

```
                    fseek(fp,pos,SEEK_SET);
                }else fi->bitspersample=(int)offset;
                break;
        case Threshholding:
                fi->threshhold=(int)offset;
                break;
        case Compression:
                fi->compression=(int)offset;
                break;

        case Group3Option:
                break;
        case Group4Option:
                break;
        case FillOrder:
                break;
/*      case Threshholding:
                break; */
        case CellWidth:
                break;
        case CellLength:
                break;
        case MinSampleValue:
                break;
        case MaxSampleValue:
                break;
        case PhotometricInterp:
                break;
        case GrayResponseUnit:
                break;
        case GrayResponseCurve:
                break;
        case ColorResponseUnit:
                break;
        case ColorResponseCurves:
                break;
        case XResolution:
                break;
        case YResolution:
                break;
        case ResolutionUnit:
                break;
        case Orientation:
                break;
        case DocumentName:
                break;
        case PageName:
                break;
        case XPosition:
                break;
        case YPosition:
                break;
        case PageNumber:
                break;
        case ImageDescription:
```

```
                break;
        case Make:
                break;
        case Model:
                break;
        case FreeOffsets:
                break;
        case FreeByteCounts:
                break;
                case ColorMap:
                pos=ftell(fp);
                fseek(fp,offset,SEEK_SET);
                for(i=0;i<(i<<fi->bitspersample);++i){
                        if(i>=256) break;
                        fi->palette[i*RGB_SIZE+RGB_RED]=
                                fgetword(fp)>>8;
                        }
                        for(i=0;i<(i<<fi->bitspersample);++i){
                if(i>=256) break;
                                fi->palette[i*RGB_SIZE+RGB_GREEN]=
                                fgetword(fp)>>8;
                        }
                        for(i=0;i<(i<<fi->bitspersample);++i){
                                if(i>=256) break;
                                fi->palette[i*RGB_SIZE+RGB_BLUE]=
                                fgetword(fp)>>8;
                        }
                        fseek(fp,pos,SEEK_SET);
                        break;
                }
        }

unsigned int fgetword(FILE *fp)

{
  if(numbertype=='II') return((fgetc(fp) & 0xff) + ((fgetc(fp) & 0xff)<<8));
  else  return(((fgetc(fp) & 0xff) <<8) + (fgetc(fp)& 0xff));
}

/**********/
unsigned long fgetlong(FILE *fp)

{
        if(numbertype=='II')
                return((unsigned long)(fgetc(fp) & 0xff)+
                    ((unsigned long)(fgetc(fp) & 0xff)<<8)+
                    ((unsigned long)(fgetc(fp) & 0xff)<<16)+
                    ((unsigned long)(fgetc(fp) & 0xff)<<24));
        else
                return((unsigned long)(fgetc(fp) & 0xff)<<24)+
                    ((unsigned long)(fgetc(fp) & 0xff)<<16)+
                    ((unsigned long)(fgetc(fp) & 0xff)<<8)+
                    ((unsigned long)(fgetc(fp) & 0xff));
}
```

```
/************/
 /* data file name with specifio extension */
 void  strmfe(char *new,char *old,char *ext)
/*************/
{
        while(*old != 0 && *old != '.') *new++,*old++;
        *new++='.';
        while(*ext) *new++=*ext++;
        *new=0;
 }

int readline(p,fp,bytes,fi)
        char *p;
        FILE *fp;
        int bytes;
        FILEINFO *fi;
{
        int c,i,n=0;

        if(fi->compression==1) return(fread(p,1,bytes,fp));
        else if(fi->compression==0x005) {
                do {
                        c=fgetc(fp) & 0xff;
                        if(c & 0x80) {
                                if(c!=0x80) {
                                        i= ((~c) & 0xff)+2;
                                        c=fgetc(fp);
                                        while(i--)  p[n++] =c;
                                }
                        }
                        else {
                                i=(c & 0xff)+1;
                                while(i--) p[n++]= fgetc(fp);
                        }
                }while(n<bytes);
                return(n);
        }
        else return(0);
}

int  open1(void)
{
        char path[40];
        char *fname[10];
        int gdriver=DETECT,gmode;

        if (1)   {
                setcolor(BLACK);
                rectangle(170,120,450,170);
                gotoxy(24,10);
                outthai(180,135,1,"Enter file name (*.tif):  ");
                /*printf("Enter file name (*.tif):  ");*/
                gotoxy(49,10);
                scanf("%s",fname);
                strmfe(path,fname[1],"TIF");
```

```
                strupr(path);

                if ((fp =fopen(fname,"rb")) !=NULL){
                        unpaoktiff(fp,&fi);
                /*      fp=NULL;*/
                        folose(fp);
                        save_f();
                /*      outfile=NULL;*/
                        folose(outfile);
                /*      if(putimag1()==0){
                                return(0);
                                }*/
                        }
                else{
                        olear_page(1);
                        setoolor(RED);
                        reotangle(170,120,450,170);
                        gotoxy(24,10);
                        outthai(180,135,1,"Error File Not found");
                        /*printf("Error %s not fount.",path);*/
                        getoh();
                        return(0);
                    }/*end else*/
        }
/*      putimag1();*/
/*      setoolor(YELLOW);
        reotangle(170,170,450,210);
        gotoxy(24,20);
        printf("OpenFile 'O.k'   ");
        gotoxy(24,22);
        printf("\nprease any key");
        getoh(); */
        return(1);
}
int putimag1(void)
{
  ohar a;
  long i,j,pos_x,pos_y,oolor;
  int x_o,y_o;    /*  move to oenter  */
  float soale_x,soale_y,ratio;
  int maxx,maxy; /* max soreen x and max soreen y; */

        if((tp=fopen("tiftext.txt","r"))==NULL) {
                olear_page(1);
                setoolor(RED);
                reotangle(170,120,450,170);
                gotoxy(24,10);
                outthai(180,135,1,"Can't open file");
                /*printf("Can't open file\n");*/
                return(0);
                }
        fseek(tp,0L,0);

        if((data=(ohar *)malloo(wide))==NULL) {
                olear_page(1);
```

```
                setcolor(RED);
                rectangle(170,120,450,170);
                gotoxy(24,10);
                outthai(180,135,1,"Not Enouge memory in Show image");
                getch();
                return(0);
                }
/*      data= (char *)malloc(wide);*/
        temp=data;

        maxx=getmaxx();
        maxy=getmaxy();
        x_o=(maxx/2)-(wide/2);
        y_o=(maxy/2)-(len/2);

         maxx-=100; /*position of image in case big size*/
         maxy-=100;
        if ((len>maxy) ||(wide>maxx)) {
                scale_x=maxx / (float)wide;
                scale_y=maxy / (float)len;
                ratio =(scale_x < scale_y) ?  scale_x:scale_y;
        /*      x_o=((scale_x*wide))-(maxx/2);
                y_o=((scale_y*len))-(maxy/2);*/
         for(j=0;j<len;j++) {
            pos_y=(float)j * ratio +0.5;
            get_row(tp,wide);
                for(i=0;i<wide;i++) {
                        pos_x=(float)i * ratio +0.5;
                        if(*data==0x01) color=BLACK;
                        if(*data==0x00) color=WHITE;
                        putpixel(pos_x+20,pos_y+80,color);
                        data++;
                        }
                        if(j == len-2) break;
                }
            }
            else {
                 for (j=0;j<len;j++){
                        get_row(tp,wide);
                        for(i=0;i<wide;i++) {
                        if(*data==0x01) color=BLACK;
                        if(*data==0x00) color=WHITE;
                        putpixel(i+x_o,j+y_o,color);
                        data++;
                        }
                }
         }
         a=getch();

         fclose(tp);
         data=NULL;
         free(data);
         return(1);
         }
```

SEGMENT.C

```
#include <stdio.h>
#include <stdlib.h>
#include <alloc.h>
#include <conio.h>
#include <mem.h>
#include <dos.h>
#include <string.h>
#include <graphics.h>
/*#include "read_s.h"*/
#include "zone.h"
#include "chr.h"

/*extern int segment(void);*/
extern long f_pos;
extern FILE *tp;
extern int lines_per_buffer;
extern SAMPLEINFO    sample[5];

ZONE zone_list[130];
int num_sample_s;
ZONE  zone_char[130];
int zone_count;
int char_count;
int img_width;
unsigned char   *data_s,*temp_s;
unsigned char far *ptr;
int x1,y1,x2,y2;
 /*extern int num_sample_s;*/
int zonewidth,zoneheight;
int combine_box;
int char_wide=0;
int type;

int getrow(FILE *tp,int wide);
int read_zone(int n,FILE *tp);
int extract_char(unsigned char far  *image,int dx,int dy,int x,int y,ZONE *zone_ptr);
void block_char(unsigned char  far  *image,int dx,int dy);
void sequence_char(unsigned char far  *image,int dx,int dy);
void sort_char(int dx,int dy);
void combine_sample_char(void);
void overlap_sample_char(ZONE *array,int *array_size);
void record_char(int n);
void overlap_char(ZONE *zone_array,int *array_size);
int segment(void);
void write_info(FILE *fzonechr);
void read_info(FILE *fzonelis);

FILE *txt;
/*WINDOW *char_win,*recog_win;*/
unsigned int index;
CHAR_INFO st[100];
char *msg ="\n\
          \n Zone image too big to fit memmory.\n\
          \n Save this zone and continue later.\n\
          \n\n";
```

```
int read_zone(int n,FILE *tp)
{       unsigned char *p;
          int a,k,i,j;

          int bytes_per_line,lines_per_page;
          int lines_per_buffer;
          unsigned long mem_left;
          int remain_lines,remain_block;

          x1=zone_list[n].x1;
          x2=zone_list[n].x2;
          y1=zone_list[n].y1;
          y2=zone_list[n].y2;
          zonewidth=(x2-x1)+1;
          zoneheight=(y2-y1)+1;
          bytes_per_line=zonewidth;
          lines_per_page=zoneheight;
          /*mem_left=coreleft();*/
            mem_left=0xffoo;
            /**********************************/
            /*        Test Sample             */
            /**********************************/
          if(mem_left < (long)zonewidth * (long)zoneheight)  {
                    lines_per_buffer=0xffoo/bytes_per_line;
                    num_sample_s=(float)lines_per_page/(float)lines_per_buffer+0.5;
                    remain_lines=lines_per_page-((num_sample_s-1)*lines_per_buffer);
                    remain_block=lines_per_buffer*num_sample_s;
          if((ptr=(unsigned char*)farmalloo(0xffoo))==NULL)
                    {
                    rectangle(170,120,450,170);
                    gotoxy(24,10);
                    printf("Error malloo memmory ptr in sample ");
                    getch();
                    return(0);
                    } /*end malloo ptr*/
                    memset(ptr,0x00,0xffoo);
          if((data_s=(unsigned char*)malloo(img_width))==NULL){
                    rectangle(170,120,450,170);
                    gotoxy(24,10);
                    printf("Error molloo data");
                    getch();
                    return(0);
                    } /*end malloo data*/
             temp_s=data_s;
                    fseek(tp,0,SEEK_SET);
                    fseek(tp,(long)img_width*y1,SEEK_SET);
          for(i=0;i<num_sample_s;i++) {
                    combine_box=i*lines_per_buffer;
                    if(i==num_sample_s-1){
                    for(a=0;a<remain_lines;a++) {
                              f_pos=ftell(tp);
                              fseek(tp,f_pos,SEEK_SET);
                              getrow(tp,img_width);
                              for(j=0;j<zonewidth;j++) {
```

```
                                ptr[a*zonewidth+j]=data_s[j+x1];
                                        } /*end loop j*/
                                }       /*end loop a*/

                } /*end if i==numsample-1*/
                else {
                        for(k=0;k<lines_per_buffer;k++){
                                f_pos=ftell(tp);
                                fseek(tp,f_pos,SEEK_SET);
                                getrow(tp,img_width);
                                for(j=0;j<zonewidth;j++) {
                                        ptr[k*zonewidth+j]=data_s[j+x1];
                                                } /*end loop j*/
                                }       /*end loop a*/
                } /*end else if(i!=num_sample_s-1) */
                block_char(ptr,zonewidth,lines_per_buffer);
                sequence_char(ptr,zonewidth,lines_per_buffer);
                record_char(i);
        }/*end for i*/
        free(data_s);
    /*  ptr=NULL;*/
    /*ptr=(unsigned char*)realloc((unsigned char *)ptr,1);*/
        farfree(ptr);
        /*ptr=NULL;*/
/*      fclose(tp);*/
        combine_sample_char();
        overlap_sample_char(zone_char,&char_count);
        sort_char(zonewidth,zoneheight);
        return(1);
     } /*end if image must sample*/

     /*********************************/
     /*          No sample        */
     /*********************************/
        else { /* no sample*/
                if((data_s=(unsigned char*)malloc(img_width))==NULL) {
                        rectangle(170,120,450,170);
                        gotoxy(24,10);
                        printf("Error allocate dsta_s in read_zone()\n");
                        getch();
                        return(0);
                        }/*end malloc data*/
                if((ptr=(unsigned char*)farmalloc(0xffoo))==NULL)
                {
                        rectangle(170,120,450,170);
                        gotoxy(24,10);
                        printf("Error allocate ptr in No sample()\n");
                        getch();
                        return(0);
                } /*end malloc ptr*/

        temp_s=data_s;

                fseek(tp,(long)img_width*y1,SEEK_SET);
                for(i=0;i<zoneheight;i++) {
```

```
                        getrow(tp,img_width);
                    for(j=0;j<zonewidth;j++) {
                        ptr[i*zonewidth+j]=data_s[j+x1];
                    }/* end loop j*/
                }
                block_char(ptr,zonewidth,zoneheight);
                sequence_char(ptr,zonewidth,zoneheight);
                sort_char(zonewidth,zoneheight);
                free(data_s);
                /*ptr=NULL;*/
                farfree(ptr);
                /*ptr=NULL;*/
                return(1);
        }
}


int getrow(FILE *tp,int wide)
/*FILE *tp;
int wide; */
{
int k;
        data_s=temp_s;
        for(k=0;k<wide;k++)
                {
                fread(data_s,sizeof(char),1,tp);
                data_s++;
                }
        data_s=temp_s;
        return(0);
}

void block_char(unsigned char far *image,int dx,int dy)
{
        int i,j,x,y,soore;
        char  *line,*pixel;

        for(y=1,line=image+dx ; y < dy-1 ; y++, line+=dx) {
            /*  indicator(" Block zones ",y);*/
/* walk through buffer scanning for neighbors */
                for (x=1, pixel=line+1; x < dx-1 ; x++, pixel++) {
                        soore = 0;
                        if(*(pixel-1) & 0x01) /*ALIVE          /* left */
                                soore++;
                        if(*(pixel+1) & 0x01)/*ALIVE)          /* right */
                                soore++;
                        if(*(pixel-dx) & 0x01)/*ALIVE)         /* up */
                                soore++;
                        if(*(pixel+dx) & 0x01)/*ALIVE          /* down */
                                soore++;
                        if((*pixel == 0x00) && (soore >= 2))/*LIFE_SCORE))/*0x00 is WHITE */
                                *pixel = 0xf0;
                        else if((*pixel == 0x01) && (soore <= 0))/*DEATH_SCORE))*/
                                *pixel = 0x0f;
                }
```

```
        }

                        for(pixel = image ; pixel < image + dy * dx; pixel++) {
                /* birth and bury */
                if(*pixel == 0xf0)/*PREGNANT)*/
                        *pixel = 0x01;
                else if (*pixel == 0x0f) /*SICK*/
                        *pixel = 0x00;
                }
}
void sequence_char(unsigned char far *image,int dx,int dy)
{
        int x,y,x_pos,ypos;/**position of x in image***/
        unsigned char far  *tr;
        ypos=0;
        for(y=ypos,char_count=0 ; y < dy && char_count < 130 ; y+=1) {/*MAX_ZONE and
MIN_Y_ZONE*/
                /* extract zones form block images in y order */
tr = image + y * dx;
                x_pos=0;
                for (x =x_pos ; x < dx ; x += 1)/* MIN_X_ZONE*/
                        if (*(tr+x)) { /* found point */
                                if (char_count > 130)/*MAX_ZONE*/
                                        break;
                                while (x > 0 && *(tr+x-1))      /* back up to left side */
                                        x--;
                                x_pos=x;
                                if(extract_char(image,dx,dy,x,y,zone_char+char_count)){
                                        char_count++;   /* get zone */

                                        }
*                               }
                        /*ypos=y;*/
        }
        /* remove overlaping zones */
        overlap_char(zone_char,&char_count);
        return;
}
void overlap_char(ZONE *zone_array,int *array_size)
{
        int i,j,ax1,ay1,ax2,ay2,bx1,by1,bx2,by2;

        for (i = 0 ; i < *array_size-1 ; i++)
        {       /* compare each zone against the rest */
             /*   indicator(" Overlap zones",j);*/
                ax1 = (zone_array+i)->x1;
                ay1 = (zone_array+i)->y1;
                ax2 = (zone_array+i)->x2;
                ay2 = (zone_array+i)->y2;
                for (j = i+1 ; j < *array_size ; j++)
                {
                        bx1 = (zone_array+j)->x1;
                        by1 = (zone_array+j)->y1;
                        bx2 = (zone_array+j)->x2;
                        by2 = (zone_array+j)->y2;
```

```
                        if ((bx1>=ax2) || (ax1>=bx2) || (by1>=ay2) || (ay1>=by2))
                                continue;       /* no overlap */
                        bx1 = min(bx1,ax1);     /* combine zones */
                        by1 = min(by1,ay1);
                        bx2 = max(bx2,ax2);
                        by2 = max(by2,ay2);
                        (zone_array+i)->x1 = bx1;
                        (zone_array+i)->y1 = by1;
                        (zone_array+i)->x2 = bx2;
                        (zone_array+i)->y2 = by2;
                        (*array_size)--;        /* new zone count */
                        memmove((char *) (zone_array+j),(char *) (zone_array+j+1),
                                sizeof(ZONE)*(*array_size-j));  /* shift array to remove zone */
                        i = -1;         /* start all over */
                        break;
                }
        }
        return;
}

void sort_char(int dx,int dy)
{
        int x,y,i,j,index,fudge_x,fudge_y;/*z=0;*/
        ZONE temp;

for (i = 0 ; i < char_count-1 ;i++) {
/*indicator(" Sorting zones ",z+i);*/
/* sort on x1 */
for (j = i+1 ; j < char_count ; j++) {
if (zone_char[j].x1 < zone_char[i].x1) {
temp = zone_char[i];
zone_char[i] = zone_char[j];
                        zone_char[j] = temp;
                }
        }
        }
        for (i = 0 ; i < char_count-1 ; i++) {
                /* order zones left to right, up to down */
           /*   indicator(" Sorting zones ",z+i);*/
                x = zone_char[i].x2;
                y = zone_char[i].y1;
                fudge_x = zone_char[i].x1+dx/20;        /* 5% slippage on alignment */
                for (j = i+1, index = -1 ; j < char_count ; j++) {
                        /* find any zones above and within x extsnt of zone_char[i] */
                        if (zone_char[j].x1 > x)
                                break;
                        if ((zone_char[j].y1 < y) && (zone_char[j].x1 <= fudge_x)) {
                                x = zone_char[j].x2;
                                y = zone_char[j].y1;
                                index = j;
                        }
                }
                if (index != -1) {
                        temp = zone_char[i];
                        zone_char[i] = zone_char[index];
```

```
                        zone_char[index] = temp;
                }
        }
}
int extract_char(unsigned char far *image,int dx,int dy,int x,int y,ZONE *zone_ptr)
{
        typedef enum {GOING_UP,GOING_RIGHT,GOING_DOWN,GOING_LEFT } HEADING;
        int ix,iy,minx,miny,maxx,maxy;  /* perimeter variables & min/max values */
        HEADING dir;                    /* current direction */
        unsigned char far  *previous,*next,*here;     /* buffer pointers */

        minx = maxx = ix = x;           /* preset min/max x/y and perimeter vars */
        miny = maxy = iy = y;
        dir = GOING_UP;                 /* starting direction */
        do      /* walk perimeter, recording min/max of rectangular region */
        {
                if (ix < minx)          /* update min/max */
                        minx = ix;
                if (ix > maxx)
                        maxx = ix;
                if (iy < miny)
                        miny = iy;
                if (iy > maxy)
                        maxy = iy;
                here = image + iy * dx + ix;    /* where are we? */
                next = here + dx;
                previous = here - dx;
                switch (dir)            /* base on current direction. */
                {
                        case GOING_UP:
                        if (ix > 0 && *(here-1))
                        {
                                ix--;
                                dir = GOING_LEFT;
                                break;
                        }
                        if (iy > 0 && *previous)
                        {
                                iy--;
                                break;
                        }
                        if (ix < dx-1 && *(here+1))
                        {
                                ix++;
                                dir = GOING_RIGHT;
                                break;
                        }
                        if (iy < dy-1 && *next)
                        {
                                iy++;
                                dir = GOING_DOWN;
                                break;
                        }
                        break;
                        case GOING_RIGHT:
```

```
if (iy > 0 && *previous)
{
        iy--;
        dir = GOING_UP;
        break;
}
if (ix < dx-1 && *(here+1))
{
        ix++;
        break;
}
if (iy < dy-1 && *next)
{
        iy++;
        dir = GOING_DOWN;
        break;
}
if (ix > 0 && *(here-1))
{
        ix--;
        dir = GOING_LEFT;
        break;
}
break;
case GOING_DOWN:
if (ix < dx-1 && *(here+1))
{
        ix++;
        dir = GOING_RIGHT;
        break;
}
if (iy < dy-1 && *next)
{
        iy++;
        break;
}
if (ix > 0 && *(here-1))
{
        ix--;
        dir = GOING_LEFT;
        break;
}
if (iy > 0 && *previous)
{
        iy--;
        dir = GOING_UP;
        break;
}
break;
case GOING_LEFT:
if (iy < dy-1 && *next)
{
        iy++;
        dir = GOING_DOWN;
        break;
```

```
                        }
                        if (ix > 0 && *(here-1))
                        {
                                ix--;
                                break;
                        }
                        if (iy > 0 && *previous)
                        {
                                iy--;
                                dir = GOING_UP;
                                break;
                        }
                        if (ix < dx-1 && *(here+1))
                        {
                                ix++;
                                dir = GOING_RIGHT;
                                break;
                        }
                        break;
                }
        } while (ix != x || iy != y);   /* until we return to the start */
        for (iy=miny, here=image+miny*dx+minx; iy<=maxy+3; iy++,here+=dx)
                memset(here,0x00,maxx-minx+3); /* white out the region */
        if ((maxx-minx+1 < 1) || (maxy-miny+1 < 1)) /*MIN_X_ZONE and MIN_Y_ZONE/*
                return 0;               /* big enough? */
        if (minx > 0)                   /* expand dimensions by one pixel */
                minx--;
        if (maxx < dx-1)
                maxx++;
        if (miny > 0)
                miny--;
        if (maxy < dy-1)
                maxy++;
        zone_ptr->x1 = minx+x1;                 /* save zone */
        zone_ptr->y1 = miny+y1+combine_box;
        zone_ptr->x2 = maxx+x1;
        zone_ptr->y2 = maxy+y1+combine_box;
        zone_ptr->wide=(maxx-minx)+1;
        return 1;
}

void overlap_sample_char(ZONE *zone_array,int *array_size)
{
        int i,j,ax1,ay1,ax2,ay2,bx1,by1,bx2,by2;
        for (i = 0; i < *array_size-1; i++)
        {  /* compare each zone against the rest */
                ax1 = (zone_array+i)->x1;
                ay1 = (zone_array+i)->y1;
                ax2 = (zone_array+i)->x2;
                ay2 = (zone_array+i)->y2;
                /* expand zone down (y2) by one pixel */
                if(ay2 < zoneheight-1)
                        ay2++;
                for (j = i+1 ; j < *array_size ; j++)
                {
```

```
                        bx1 = (zone_array+j)->x1;
                        by1 = (zone_array+j)->y1;
                        bx2 = (zone_array+j)->x2;
                        by2 = (zone_array+j)->y2;
                        /* expand zone up (y1) by one pixel */
                        if(by1 > 0)
                                by1--;
                        if (((bx1-2)>=(ax2+2)) || (ax1>=bx2) || ((by1-1)>=(ay2+1)) || (ay1>=by2))
                                continue;       /* no overlap */
                        bx1 = min(bx1,ax1);     /* combine zones */
                        by1 = min(by1,ay1);
                        bx2 = max(bx2,ax2);
                        by2 = max(by2,ay2);
                        (zone_array+i)->x1 = bx1;
                        (zone_array+i)->y1 = by1;
                        (zone_array+i)->x2 = bx2;
                        (zone_array+i)->y2 = by2;
                        (*array_size)--;        /* new zone count */
                        memmove((char *) (zone_array+j),(char *) (zone_array+j+1),
                                sizeof(ZONE)*(*array_size-j));  /* shift array to remove zone */
                        i = -1;         /* start all over */
                        break;
                }
        }
        return;
}


void record_char(int n)
{
        int i;
        sample[n].zones = char_count;
        sample[n].line_no = n * lines_per_buffer;
        for(i=0; i<char_count; i++) {
                sample[n].zone_info[i].x1=zone_char[i].x1;
                sample[n].zone_info[i].x2=zone_char[i].x2;
                sample[n].zone_info[i].y1=zone_char[i].y1;
                sample[n].zone_info[i].y2=zone_char[i].y2;
        }
        return;
}

 void combine_sample_char(void)
{
        int i,j,k;

        for(i=1; i<num_samples; i++) {
                for(j=0; j<sample[i].zones; j++) {
                        sample[i].zone_info[j].y1 += sample[i].line_no;
                        sample[i].zone_info[j].y2 += sample[i].line_no;
                }
        }
        for(k=0,char_count=0; k<num_samples; k++) {
                char_count+=sample[k].zones;
        }
```

```
        i=0;
        do {
                for(j=0; j<num_sample_s; j++) {
                        for(k=0; k<sample[j].zones; k++) {
                                zone_char[i].x1=sample[j].zone_info[k].x1;
                                zone_char[i].x2=sample[j].zone_info[k].x2;
                                zone_char[i].y1=sample[j].zone_info[k].y1;
                                zone_char[i].y2=sample[j].zone_info[k].y2;
                                i++;
                        }
                }
        } while(i!=char_count);
        return;
}

int  segment(void)
{
int a,n;
long fpos;
int zone_depth=0;
int check_module;
FILE *fzonelis,*fzonechr;

                rectangle(165,50,390,90);
                gotoxy(25,5);
                printf("Analysist Fine Character");
        if((fzonelis=fopen("zonelist.txt","rb"))==NULL) {
                rectangle(170,120,450,170);
                gotoxy(24,10);
                printf("Error Open File zonelist.txt\n");
                getch();
                return(0);
                }
                read_info(fzonelis);
                folose(fzonelis);
        /*      fzonelis=NULL;*/
   /*   printf("%d\t",zone_count);
        printf("%d\n",img_width); */
/*for(a=0;a<zone_count;a++)
      {

        printf("minx is %d maxx is %d miny is %d maxy is  %d\n",zone_list[a].x1,zone_list[a].
x2,zone_list[a].y1,zone_list[a].y2);
        zone_depth+=(zone_list[a].y2-zone_list[a].y1)+1;

        }
        printf("zone_depth is %d\n",zone_depth);*/
        if((fzonechr=fopen("zonechar.txt","wb"))==NULL) {
                rectangle(170,120,450,170);
                gotoxy(24,10);
                printf("Error Open File zonechr.txt\n");
                getch();
                return(0);
                }
                /*setvbuf(fzonechr,NULL,_IOFBF,0);*/
```

```
                fwrite(&zone_count,sizeof(int),1,fzoneohr);
                fwrite(&zone_depth,sizeof(int),1,fzoneohr);

        /*if((tp=fopen("tiftext.txt","r"))==NULL) {
                printf("Error Open File tiftext.txt\n");
                return(0);
                } */
        for(n=0;n<zone_count;n++)
                {
                if(zone_list[n].type==2)
                {
                        char_count=1;

                        type=2;
                        write_info(fzoneohr);
                continue;
                }
                /*if((ptr=malloc(1))==NULL)
                        {
                        rectangle(170,120,450,170);
                        gotoxy(24,10);
                        printf("Error malloc memmory ptrin sample ");
                        getch();
                        return(0);
                        }
                        free(ptr);*/
                if((tp=fopen("tiftext.txt","r"))==NULL) {
                        rectangle(170,120,450,170);
                        gotoxy(24,10);
                        printf("Error Open File tiftext.txt\n");
                        getch();
                        return(0);
                        }
                /*setvbuf(tp,NULL,_IONBF,0);*/
                check_module=read_zone(n,tp);
                if(check_module==0){
                        rectangle(170,120,450,170);
                        gotoxy(24,10);
                        printf("Error in Process.txt\n");
                        getch();
                        return(0);
                }
                /*data_s=NULL;*/
                /*free(data_s);*/
                /*data_s=NULL;*/
                fclose(tp);
                /*tp=NULL;*/
                for(a=0;a<char_count;a++)
                        char_wide+= (5+zone_char[a].wide);/* 5 is blank in char*/

                type=1;/*this is a text*/
                write_info(fzoneohr);
                char_wide=0;
                setcolor(GREEN);
        for(a=0;a<char_count;a++){
```

```
                    reotangle(145,100,420,130);
                    gotoxy(22,8);
                    printf("Chararter %d  ",a+1);
                    gotoxy(36,8);
                    printf("Have Information");
                    /*show blook */
                    reotangle(100,200,500,250);
                    delay(100);
                    gotoxy(15,15);
          printf("minx is %d maxx is %d miny is %d maxy is  %d\n",zone_ohar[a].x1,zone_ohar[a].
x2,zone_ohar[a].y1,zone_ohar[a].y2);
          }
          /**/

/*        printf("\nzone have wide %d\n",ohar_wide);*/
          /*  */

          reotangle(165,260,410,300);
          gotoxy(25,18);
          printf("Total Chararter %d in Row %d",ohar_oount,n+1);
            }
          /*folose(tp);*/
          folose(fzoneohr);
          /*fzoneohr=NULL;*/
          return(1);

 }

void write_info(FILE *fzoneohr)
{
            fwrite(&ohar_oount,sizeof(int),1,fzoneohr);
            fwrite(&img_width,sizeof(int),1,fzoneohr);
            fwrite(&ohar_wide,sizeof(int),1,fzoneohr);
            fwrite(&zoneheight,sizeof(int),1,fzoneohr);
            fwrite(zone_ohar,sizeof(ZONE),ohar_oount,fzoneohr);
            fwrite(&type,sizeof(int),1,fzoneohr);
}

void read_info(FILE *fzonelis)
{
          fread(&zone_oount,sizeof(int),1,fzonelis);
          fread(&img_width,sizeof(int),1,fzonelis);
          fread(zone_list,sizeof(ZONE),zone_oount,fzonelis);
          /*fread(zone_type,sizeof(int),zone_oount,fzonelis);*/
}
```

GETZONE.C

```
#inolude <stdio.h>
#inolude <stdlib.h>
#inolude <string.h>
#inolude <ermo.h>
#inolude <dos.h>
#inolude <math.h>
#inolude <alloo.h>
#inolude <graphios.h>
#inolude "zone.h"
#inolude "read_g.h"
/*#inolude "wtwg.h"*/

/* extern prototypes & variables */
int img_width_g,img_length;

/*extern  get_zone();*/

/* looal variables */
unsigned ohar far *image=NULL;
ohar *lines=NULL;
SAMPLEINFO sample[53];          /* zone extent, zone_oount, line offset */
ZONE zone_list_g[100];/*MAX_ZONES      /* this ould be dynamioally allooated */
int zone_oount_g;               /* index of zones */
int lines_per_buffer;/* number of lines in a buffer */
int num_sample;                 /* number of sample of image */
FILE *tp_g;
long f_pos;
ohar *data_g=NULL,*temp_g;
int base_line;

/* looal prototypes */
int zone(int ooarseness,int order,FILE *tp_g);
void blook_zones(unsigned ohar far *image,int dx,int dy,int ooarseness);
void sequenoe_zones(unsigned ohar far *image,int dx,int dy);
int extraot_zone(unsigned ohar far *image,int dx,int dy,int x,int y,ZONE *zone_ptr);
void overlap_zones(ZONE *zone_array,int *array_size);
void sort_zones(int dx,int dy,int order);
ohar *get_page(int dx,int dy,FILE *tp_g);
void reoord_zone(int n);
void oombine_samples(void);
void overlap_samples(ZONE *array,int *array_size);
int get_zone();

int zone(int ooarseness,int order,FILE *tp_g)
{
        int lines_per_page;          /* number of lines of image */
        int bytes_per_line;
        int remain_line;             /* remain line after sampling */
        unsigned ohar far *ptr;
        int dx,dy;/*offset;*/
        unsigned long image_size;
        unsigned int i,j;
        FILE *out_fp;

        if(ooarseness < 0 || ooarseness >100) {
```

```
                /*test coarseness parameter*/
            /*  errno = EINVAL;*/
                return(0);
}
        bytes_per_line = dx =img_width_g;
        lines_per_page = dy =img_length;
        image_size=(long)img_width_g * (long)img_length;
        if(image_size > (long)3500000) {
        /*      beep();
        /*      wprompto(NULL," Image too big to detext  block zone.\n",NULL);*/
                rectangle(170,120,450,170);
                gotoxy(24,10);
                printf(" Image too big to detext  block zone");
                return(0);
        }
        else {
        /*      wreopen(indicate);
        /*      indicate=w0;*/
    /*          wgoto(0,0);*/
   /*           wputs(" Please wait...");*/
                if(image_size >     BUFFER_SIZE) {
                        /* sampling image along bertical */
                        lines_per_buffer=0xffoo / bytes_per_line; /*0xff is MAX_ALLOC*/
                        num_sample=(float)lines_per_page / (float)lines_per_buffer + 0.5;
                        remain_line=lines_per_page - ((num_sample-1) * lines_per_buffer);
                        if((image=(unsigned char*)farmalloc(0xffoo))==NULL)     {
                          rectangle(170,120,450,170);
                          gotoxy(24,10);
                          printf("error allocate image\n");
                          getch(0);
                          return(0);
                          }
                        memset(image,0x00,0xffoo);
                        if((lines=(char *)malloc(bytes_per_line))==NULL){
                                rectangle(170,120,450,170);
                                gotoxy(24,10);
                                printf("Error allocate p in get_page()\n");
                                getch();
                                return(0);
                                }
                        ptr=image;
                        temp_g=lines;
                        for(i=0;i<num_sample;i++) {
                        memset(image,0x00,0xffoo);
                             /*  offset=lines_per_buffer * i;*/
                                if(i==num_sample-1) {
                                        for(j=0;j<remain_line;j++,image+=bytes_per_line) {
                                                /*readheap(line,offset+j);*/
                                                f_pos=ftell(tp_g);
                                                fseek(tp_g,f_pos,SEEK_SET);
                                                get_row_g(tp_g,img_width_g);/*get row in
getzone*/
                                                memcpy(image,lines,bytes_per_line);
                                        }
                }
```

```
                        else {
                                for(j=0;j<lines_per_buffer;j++,image+=bytes_per_line) {
                                        /*readheap(lines,offset+j);*/
                                        f_pos=ftell(tp_g);
                                        fseek(tp_g,f_pos,SEEK_SET);
                                        get_row_g(tp_g,img_width_g);
                                        memcpy(image,lines,bytes_per_line);
                                }
                        }
                        image=ptr;
                        block_zones(image,bytes_per_line,lines_per_buffer,coarseness);
                        sequence_zones(image,bytes_per_line,lines_per_buffer);
                        record_zone(i);
                }
                free(lines);
                farfree(image);
                fclose(tp_g);
                tp_g=NULL;
                /* combine zones in each sample */
                combine_samples();

                /* find orerlaped zones of each sample & combine them */
                overlap_samples(zone_list_g,&zone_count_g);
                sort_zones(bytes_per_line,lines_per_page,order);
        }
        else {
                /* image can be stored in buffer, no need sampling */
                if((image=get_page(dx,dy,tp_g))==NULL){
                        rectangle(170,120,450,170);
                        gotoxy(24,10);
                        printf("Error allocate p in get_page()\n");
                        getch();
                        return(0);
                        }
                block_zones(image,dx,dy,coarseness);
                sequence_zones(image,dx,dy);
                sort_zones(dx,dy,order);
        }
    /*  wclear();
        wgoto(0,0);
        wsetattr(48);
        wputs(" <Esc> to exit. ");
        wabandon();  */
        farfree(image);
        return(1);
    }
}

/*void indicator(char title[15],int i)
{
    /*  wgoto(0,0);
        wprintf("%s %3d",title,i);*/
  /*    return;
} /*
```

```
/*************** LOCAL FUNCTIONS ********************/

char *get_page(int dx,int dy,FILE *tp_g)
{
        int i;
        unsigned char far *img_ptr;

        if((image=(unsigned char *)farmalloc(0xffoo))==NULL) {
                        rectangle(170,120,450,170);
                        gotoxy(24,10);
                        printf("Error allocate p in get_page()\n");
                        getch();
                        return(0);
                        }
        if((lines=(char *)malloc(dx))==NULL) {
                rectangle(170,120,450,170);
                gotoxy(24,10);
                printf("Error allocate p in get_page()\n");
                getch();
                return(0);
                }
        temp_g=lines;
        for(i=0,img_ptr=image; i<dy; i++,img_ptr+=dx) {
             /*    readheap(p,i);*/
                        f_pos=ftell(tp_g);
                        fseek(tp_g,f_pos,SEEK_SET);
                        get_row_g(tp_g,img_width_g);
                    memcpy(img_ptr,lines,dx);
}
        free(lines);
        fclose(tp_g);
        /*tp_g=NULL;*/
        return image;
}

void block_zones(unsigned char far *image,int dx,int dy,int coarseness)
{
        int i,j,x,y,score;
        unsigned char far *line,*pixel;

        for(y=1,line=image+dx ; y < dy-1 ; y++, line+=dx) {
             /*   indicator(" Block zones ",y);*/
/* walk through buffer scanning for neighbors */
                    for (x=1, pixel=line+1; x < dx-1 ; x++, pixel++) {
                        score = 0;
                        if(*(pixel-1) & 0x01) /*ALIVE          /* left */
                                score++;
                        if(*(pixel+1) & 0x01)/*ALIVE)          /* right */
                                score++;
                        if(*(pixel-dx) & 0x01)/*ALIVE)         /* up */
                                score++;
                        if(*(pixel+dx) & 0x01)/*ALIVE          /* down */
                                score++;
                        if((*pixel == 0x00) && (score >= 2))/*LIFE_SCORE))/*0x00 is WHITE */
                                *pixel = 0xf0;
```

```
                        else if((*pixel == 0x01) && (score <= 0))/*DEATH_SCORE))*/
                                *pixel = 0x0f;
                }
        }
        for(pixel = image ; pixel < image + dy * dx; pixel++) {
                /* birth and bury */
                if(*pixel == 0xf0)/*PREGNANT)*/
                        *pixel = 0x01;
                else if (*pixel == 0x0f) /*SICK*/
                        *pixel = 0x00;
}
        if(coarseness <= 0)             /* no need to coalesce */
                return;
        /* coalesce regions, based on coarseness, i.e. coarseness == 2
         * will close all horizontal/vertical gape of 2 pixels */
        for(y=0, line=image+1 ; y < dy ; y++, line+=dx) {
                /* coalesce horizontally */
                for(x=1, pixel=line ; x < dx-coarseness ; x++,pixel++) {
                        if((*pixel == 0x00) && (*(pixel-1) == 0x01)) {
                                for(i =1 ; i <= coarseness ; i++) {
                                        if(*(pixel+i)) {
                                        for(i=0 ; i < coarseness ; i++, pixel++)
                                                        *pixel = 0xff;/*BLACK*/
                                                pixel--;
                                                x+=coarseness-1;
                                                break;
                                        }
                                }
                        }
                }
        }
        /*coarseness=((coarseness*2)/6)*2;*/
        for(x=0, line=image+dx ; x < dx ; x++, line++) {
                /* coalesce vertically */
for(y=1, pixel=line ; y < dy-(int)(coarseness/2) ; y++, pixel+=dx) {
                        if((*pixel == 0x00) &&(*(pixel-dx) == 0x01)) {
                        for (i=1, j=dx; i<= coarseness/2 ; i++, j+=dx) {
                                if (*(pixel+j)) {
                                        for (i = 0 ;i < coarseness/2 ; i++, pixel+=dx)
                                                *pixel=0xff;
                                                pixel -=dx;
                                                y += (coarseness/2-1);
                                                break;
                                        }
                                }
                        }
                }
 }
        return;
 }

 void sequence_zones(unsigned char far *image,int dx,int dy)
 {
        int x,y;
        unsigned char far *ptr;
```

```
        for(y=0,zone_count_g=0 ; y < dy && zone_count_g < 200 ; y+=1) {/*MAX_ZONE and
MIN_Y_ZONE*/
                /* extract zones form block images in y order */
ptr = image + y * dx;
                for (x =0 ; x < dx ; x += 1)/* MIN_X_ZONE*/
                        if (*(ptr+x)) { /* found point */
                                if (zone_count_g >= 80)/*MAX_ZONE*/
                                        break;
                                while (x > 0 && *(ptr+x-1))     /* back up to left side */
                                        x--;
                                if(extract_zone(image,dx,dy,x,y,zone_list_g+zone_count_g)){
                                        zone_count_g++;   /* get zone */
                                }
                        }

        }
        /* remove overlaping zones */
        overlap_zones(zone_list_g,&zone_count_g);
        return;
}

void sort_zones(int dx,int dy,int order)
{
        int x,y,i,j,index,fudge_x,fudge_y,z=0;
        ZONE temp;

        if (order == 1){/*COLUMN_MAJOR) {*/
for (i = 0 ; i < zone_count_g-1 ;i++) {
/*indicator(" Sorting zones ",z+i);*/
/* sort on x1 */
for (j = i+1 ; j < zone_count_g ; j++) {
if (zone_list_g[j].x1 < zone_list_g[i].x1) {
temp = zone_list_g[i];
zone_list_g[i] = zone_list_g[j];
                                zone_list_g[j] = temp;
                        }
                }
        }
        for (i = 0 ; i < zone_count_g-1 ; i++) {
                /* order zones left to right, up to down */
            /*  indicator(" Sorting zones ",z+i);*/
                x = zone_list_g[i].x2;
                y = zone_list_g[i].y1;
                fudge_x = zone_list_g[i].x1+dx/20;        /* 5% slippage on alignment */
                for (j = i+1, index = -1 ; j < zone_count_g ; j++) {
                        /* find any zones above and within x extsnt of zone_list_g[i] */
                        if (zone_list_g[j].x1 > x)
                                break;
                        if ((zone_list_g[j].y1 < y) && (zone_list_g[j].x1 <= fudge_x)) {
                                x = zone_list_g[j].x2;
                                y = zone_list_g[j].y1;
                                index = j;
                        }
                }
```

```
                if (index != -1) {
                        temp = zone_list_g[i];
                        zone_list_g[i] = zone_list_g[index];
                        zone_list_g[index] = temp;
                }
        }
}
else { /* ROW_MAJOR */
        /* already sorted in y1 order */
        for (i = 0 ; i < zone_count_g-1 ; i++) {
            /*  indicator(" Sorting zones ",z+i);*/
                /* order zones up to down, left to right */
                y = zone_list_g[i].y2;
                x = zone_list_g[i].x1;
                fudge_y = zone_list_g[i].y1+dy/20;      /* 5% slippage on alignment */
                for (j = i+1, index = -1 ; j < zone_count_g ; j++) {
                        /* find any zones left of and within y extent of zone_list_g[i] */
                        if (zone_list_g[j].y1 > y)
                                break;
                        if (zone_list_g[j].x1 < x && zone_list_g[j].y1 <= fudge_y) {
                                y = zone_list_g[j].y2;
                                x = zone_list_g[j].x1;
                                index = j;
                        }
                }
                if (index != -1) {
                        temp = zone_list_g[i];
                        zone_list_g[i] = zone_list_g[index];
                        zone_list_g[index] = temp;
                }
        }
}
return;
}

 int extract_zone(unsigned  char far *image,int dx,int dy,int x,int y,ZONE *zone_ptr)
{
        typedef enum {GOING_UP,GOING_RIGHT,GOING_DOWN,GOING_LEFT } HEADING;
        int ix,iy,minx,miny,maxx,maxy;  /* perimeter variables & min/max values */
        HEADING dir;                    /* current direction */
        unsigned char far *previous,*next,*here;     /* buffer pointers */

        minx = maxx = ix = x;           /* preset min/max x/y and perimeter vars */
        miny = maxy = iy = y;
        dir = GOING_UP;                 /* starting direction */
        do      /* walk perimeter, recording min/max of rectangular region */
        {
                if (ix < minx)          /* update min/max */
                        minx = ix;
                if (ix > maxx)
                        maxx = ix;
                if (iy < miny)
                        miny = iy;
                if (iy > maxy)
                        maxy = iy;
```

```
here = image + iy * dx + ix;    /* where are we? */
next = here + dx;
previous = here - dx;
switch (dir)            /* base on current direction. */
{
        case GOING_UP:
        if (ix > 0 && *(here-1))
        {
                ix--;
                dir = GOING_LEFT;
                break;
        }
        if (iy > 0 && *previous)
        {
                iy--;
                break;
        }
        if (ix < dx-1 && *(here+1))
        {
                ix++;
                dir = GOING_RIGHT;
                break;
        }
        if (iy < dy-1 && *next)
        {
                iy++;
                dir = GOING_DOWN;
                break;
        }
        break;
        case GOING_RIGHT:
        if (iy > 0 && *previous)
        {
                iy--;
                dir = GOING_UP;
                break;
        }
        if (ix < dx-1 && *(here+1))
        {
                ix++;
                break;
        }
        if (iy < dy-1 && *next)
        {
                iy++;
                dir = GOING_DOWN;
                break;
        }
        if (ix > 0 && *(here-1))
        {
                ix--;
                dir = GOING_LEFT;
                break;
        }
        break;
```

```
                case GOING_DOWN:
                if (ix < dx-1 && *(here+1))
                {
                        ix++;
                        dir = GOING_RIGHT;
                        break;
                }
                if (iy < dy-1 && *next)
                {
                        iy++;
                        break;
                }
                if (ix > 0 && *(here-1))
                {
                        ix--;
                        dir = GOING_LEFT;
                        break;
                }
                if (iy > 0 && *previous)
                {
                        iy--;
                        dir = GOING_UP;
                        break;
                }
                break;
                case GOING_LEFT:
                if (iy < dy-1 && *next)
                {
                        iy++;
                        dir = GOING_DOWN;
                        break;
                }
                if (ix > 0 && *(here-1))
                {
                        ix--;
                        break;
                }
                if (iy > 0 && *previous)
                {
                        iy--;
                        dir = GOING_UP;
                        break;
                }
                if (ix < dx-1 && *(here+1))
                {
                        ix++;
                        dir = GOING_RIGHT;
                        break;
                }
                break;
        }
} while (ix != x || iy != y);   /* until we return to the start */
for (iy=miny, here=image+miny*dx+minx; iy<=maxy; iy++,here+=dx)
        memset(here,0x00,maxx-minx+1); /* white out the region */
if ((maxx-minx+1 < 1) || (maxy-miny+1 < 1)) /*MIN_X_ZONE and MIN_Y_ZONE/*
```

```
                return 0;               /* big enough? */
        if (minx > 0)                   /* expand dimensions by one pixel */
                minx--;
        if (maxx < dx-1)
                maxx++;
        if (miny > 0)
                miny--;
        if (maxy < dy-1)
                maxy++;
        zone_ptr->x1 = minx;            /* save zone */
        zone_ptr->y1 = miny;
        zone_ptr->x2 = maxx;
        zone_ptr->y2 = maxy;
        return 1;
}

void overlap_zones(ZONE *zone_array,int *array_size)
{
        int i,j,ax1,ay1,ax2,ay2,bx1,by1,bx2,by2;

        for (i = 0 ; i < *array_size-1 ; i++)
        {       /* compare each zone against the rest */
            /*  indicator(" Overlap zones",j);*/
                ax1 = (zone_array+i)->x1;
                ay1 = (zone_array+i)->y1;
                ax2 = (zone_array+i)->x2;
                ay2 = (zone_array+i)->y2;
                for (j = i+1 ; j < *array_size ; j++)
                {
                        bx1 = (zone_array+j)->x1;
                        by1 = (zone_array+j)->y1;
                        bx2 = (zone_array+j)->x2;
                        by2 = (zone_array+j)->y2;
                        if ((bx1>=ax2) || (ax1>=bx2) || (by1>=ay2) || (ay1>=by2))
                                continue;       /* no overlap */
                        bx1 = min(bx1,ax1);     /* combine zones */
                        by1 = min(by1,ay1);
                        bx2 = max(bx2,ax2);
                        by2 = max(by2,ay2);
                        (zone_array+i)->x1 = bx1;
                        (zone_array+i)->y1 = by1;
                        (zone_array+i)->x2 = bx2;
                        (zone_array+i)->y2 = by2;
                        (*array_size)--;        /* new zone count */
                        memmove((char *) (zone_array+j),(char *) (zone_array+j+1),
                                sizeof(ZONE)*(*array_size-j));  /* shift array to remove zone */
                        i = -1;         /* start all over */
                        break;
                }
        }
        return;
}
 void overlap_samples(ZONE *zone_array,int *array_size)
{
        int i,j,ax1,ay1,ax2,ay2,bx1,by1,bx2,by2;
```

```
        for (i = 0; i < *array_size-1; i++)
        {  /* compare each zone against the rest */
                ax1 = (zone_array+i)->x1;
                ay1 = (zone_array+i)->y1;
                ax2 = (zone_array+i)->x2;
                ay2 = (zone_array+i)->y2;
                /* expand zone down (y2) by one pixel */
                if(ay2 < img_length-1)
                        ay2++;
                for (j = i+1 ; j < *array_size ; j++)
                {
                        bx1 = (zone_array+j)->x1;
                        by1 = (zone_array+j)->y1;
                        bx2 = (zone_array+j)->x2;
                        by2 = (zone_array+j)->y2;
                        /* expand zone up (y1) by one pixel */
                        if(by1 > 0)
                                by1--;
                        if (((bx1-2)>=(ax2+2)) || (ax1>=bx2) || ((by1-2)>=(ay2+2)) || (ay1>=by2))
                                continue;       /* no overlap */
                        bx1 = min(bx1,ax1);     /* combine zones */
                        by1 = min(by1,ay1);
                        bx2 = max(bx2,ax2);
                        by2 = max(by2,ay2);
                        (zone_array+i)->x1 = bx1;
                        (zone_array+i)->y1 = by1;
                        (zone_array+i)->x2 = bx2;
                        (zone_array+i)->y2 = by2;
                        (*array_size)--;        /* new zone count */
                        memmove((char *) (zone_array+j),(char *) (zone_array+j+1),
                                sizeof(ZONE)*(*array_size-j));  /* shift array to remove zone */
                        i = -1;         /* start all over */
                        break;
                }
        }
        return;
}

 void record_zone(int n)
{
        int i;
        sample[n].zones = zone_count_g;
        sample[n].line_no = n * lines_per_buffer;
        for(i=0; i<zone_count_g; i++) {
                sample[n].zone_info[i].x1=zone_list_g[i].x1;
                sample[n].zone_info[i].x2=zone_list_g[i].x2;
                sample[n].zone_info[i].y1=zone_list_g[i].y1;
                sample[n].zone_info[i].y2=zone_list_g[i].y2;
        }
        return;
}

 void combine_samples(void)
{
        int i,j,k;
```

```
        for(i=1; i<num_sample; i++) {
                for(j=0; j<sample[i].zones; j++) {
                        sample[i].zone_info[j].y1 += sample[i].line_no;
                        sample[i].zone_info[j].y2 += sample[i].line_no;
                }
        }
        for(k=0,zone_oount_g=0; k<num_sample; k++) {
                zone_oount_g+=sample[k].zones;
        }
        i=0;
        do {
                for(j=0; j<num_sample; j++) {
                        for(k=0; k<sample[j].zones; k++) {
                                zone_list_g[i].x1=sample[j].zone_info[k].x1;
                                zone_list_g[i].x2=sample[j].zone_info[k].x2;
                                zone_list_g[i].y1=sample[j].zone_info[k].y1;
                                zone_list_g[i].y2=sample[j].zone_info[k].y2;
                                i++;
                        }
                }
        } while(i!=zone_oount_g);
        return;
}

get_row_g(tp_g,wide)
FILE *tp_g;
int wide;
{
int k;
        lines=temp_g;
        for(k=0;k<wide;k++)
                {
                fread(lines,sizeof(ohar),1,tp_g);
            /*  *lines=geto(tp_g);*/
                lines++;
                }
                lines=temp_g;
}

int get_zone()
{
int a;
int oheok_z;
FILE *tp_g,*fzonelis,*insize;
                reotangle(165,50,390,90);
                gotoxy(25,5);
                printf("Analysist Fine zone");

        if((insize=fopen("sizetif.txt","r"))==NULL) {
                reotangle(170,120,450,170);
                gotoxy(24,10);
                printf("Error Open File tiftext.txt\n");
                getoh();
                return(0);
```

```
            }
                fread(&img_length,sizeof(int),1,insize);
                fread(&img_width_g,sizeof(int),1,insize);
                folose(insize);
                insize=NULL;
        if((tp_g=fopen("tiftext.txt","r"))==NULL) {
                reotangle(170,120,450,170);
                gotoxy(24,10);
                printf("Error Open File tiftext.txt\n");
                getch();
                return(0);
                }
        oheok_z=zone(27,2,tp_g);
        if(oheok_z==0)
        {
        reotangle(170,120,450,170);
        gotoxy(24,10);
        printf("Error Occer");
        getch();
        return(0);
        }
      oheok_row();
      for(a=0;a<zone_count_g;a++)
                {
                if(base_line>=zone_list_g[a].y2-zone_list_g[a].y1)
                        zone_list_g[a].type=1;/*1 is text*/
                else   zone_list_g[a].type=2;/*2 is piture*/
                }
                reotangle(190,100,370,130);
                gotoxy(28,8);
                printf("Total %d  Zone",zone_count_g);
        if((fzonelis=fopen("zonelist.txt","wb"))==NULL) {
                reotangle(170,120,450,170);
                gotoxy(24,10);
                printf("Error Open File zonelist.txt\n");
                getch();
                return(0);
                }
        fwrite(&zone_count_g,sizeof(int),1,fzonelis);
        fwrite(&img_width_g,sizeof(int),1,fzonelis);
        fwrite(zone_list_g,sizeof(ZONE),zone_count_g,fzonelis);
        folose(fzonelis);
        /*fzonelis=NULL;*/

for(a=0;a<zone_count_g;a++)
        {
        setoolor(GREEN);
        reotangle(100,200,500,250);
        delay(700);
        gotoxy(15,15);
        printf("minx is %d maxx is %d miny is %d maxy is  %d",zone_list_g[a].x1,zone_list_g[a].
x2,zone_list_g[a].y1,zone_list_g[a].y2);
        if(zone_list_g[a].type== 1){
                setoolor(GREEN);
                reotangle(190,260,370,300);
```

```
                gotoxy(28,18);
                printf("Zone %d is text\n",a+1);
                }
        else{
        setcolor(GREEN);
                rectangle(190,260,370,300);
                gotoxy(28,18);
                printf("Zone %d is picture\n",a+1);
                }
        }
        return(1);
}

check_row()
{
int a;
int min,max,line;
int level1,level2,level3,level4,level5;
int score1,score2,score3,score4,score5;
int row_high[24];
int high_score;
int check_line;
/*int base_line;*/
        min=max=zone_list_g[0].y2-zone_list_g[0].y1;
        score1=score2=score3=score4=score5=0;
        for(a=0;a<zone_count_g;a++)
                {
                row_high[a]=zone_list_g[a].y2-zone_list_g[a].y1+1;
                if(min>row_high[a])
                        min=row_high[a];
                if(max<row_high[a])
                        max=row_high[a];
                }
        line=1+((max-min+1)/5);
        level1=min+line;
        level2=level1+line;
        level3=level2+line;
        level4=level3+line;
        level5=level4+line;
        for(a=0;a<zone_count_g;a++)
                {
                row_high[a]=zone_list_g[a].y2-zone_list_g[a].y1;
                if(row_high[a]<=level1)
                        score1++;
                if(row_high[a]>level1 && row_high[a]<=level2)
                        score2++;
                if(row_high[a]>level2 && row_high[a]<=level3)
                        score3++;
                if(row_high[a]>level3 && row_high[a]<=level4)
                        score4++;
                if(row_high[a]>level4 && row_high[a]<=level5)
                        score5++;
                }
        high_score=score1;
```

```
        base_line=level1;/* set initial value*/

        if(high_soore<=soore2)
                {
                base_line=level2;
                high_soore=soore2;
                }
        if(high_soore<=soore3)
                {
                base_line=level3;
                high_soore=soore3;
                }
        if(high_soore<=soore4)
                {
                base_line=level4;
                high_soore=soore4;
                }
        if(high_soore<=soore5)
                {
                base_line=level5;
                high_soore=soore5;
                }

        /*if base line is pioture*/
        if(base_line >=level2 )
                base_line=level2;

        /*if level4 and level5 is text*/
        if(soore4>=1) {
                if(level1*3>level4 && base_line*2>level4)
                base_line=level4;
                check_line=1;
                }
        if(soore5>=1){
                if(level1*2>level5 && base_line*2>level5)
                base_line=level5;
                check_line=1;
                }

        base_line+=7;
        /*      setcolor(RED);
                rectangle(170,120,450,170);
                gotoxy(24,15);
                printf("Row text is high %d\n",base_line);*/
}
```

CUT_ROW.C

```
#include <stdio.h>
#include <stdlib.h>
#include <alloc.h>
#include <conio.h>
#include <mem.h>
#include <dos.h>
#include <string.h>
#include <graphics.h>
/*#include "read_c.h"*/
#include "zone.h"
#include "chr.h"

/*extern  void out_row(void);*/
/*#include "wtwg.h"*/
 ZONE zone_char_c[80]; /**character is out***/
/*SAMPLEINFO      sample[20];*/
ZONE char_out_first[100];
int zone_count_c;
int count_c;
int char_count_c;
int img_width_c;
FILE *tp_c;
char   *data_c,*temp_c;
int x1_c,y1_c,x2_c,y2_c;
int lines_per_buffer_c;
int num_sample_c;
int zonewidth_c,zoneheight_c;
long fpos_c;
int combine_box_c;
int char_wide_c=0;
int type_c;

int getrow_c(FILE *tp_c,int wide);
int read_char_c(int n,FILE *tp_c);
int extract_char_c(char  *image,int dx,int dy,int x,int y,ZONE *zone_ptr);
void block_char_c(char   *image,int dx,int dy);
void sequence_char_c(char  *image,int dx,int dy);
void sort_char_c(int dx,int dy);
void overlap_char_c(ZONE *zone_array,int *array_size);
int out_row(void);
void read_info_c(FILE *fzonelis);
void write_info_c(FILE *fzonechr);
/*FILE *txt;*/
/*WINDOW *char_win,*recog_win;*/
unsigned int index_c;
CHAR_INFO st_c[100];
char *msg_c ="\n\
            \n Zone image too big to fit memmory.\n\
            \n Save this zone and continue later.\n\
            \n\n";

int read_char_c(int n,FILE *tp_c)
{
        int a,k,i,j;
        char  *ptr=NULL;
```

```
        int remain_lines,remain_block;

        x1_o=ohar_out_first[n].x1;
        x2_o=ohar_out_first[n].x2;
        y1_o=ohar_out_first[n].y1;
        y2_o=ohar_out_first[n].y2;
        zonewidth_o=(x2_o-x1_o)+1;
        zoneheight_o=(y2_o-y1_o)+1;
        /*mem_left=ooreleft();*/

          if((ptr=(ohar *)malloo(zonewidth_o * zoneheight_o))==NULL)
                  {
                    rectangle(170,120,450,170);
                    gotoxy(24,10);
                    printf("Error allocate image in read_ohar_o()\n");
                    getch();
                    return(0);
                  }  /*end malloo ptr*/
          if((data_o=(ohar *)malloo(img_width_o))==NULL) {
                    rectangle(170,120,450,170);
                    gotoxy(24,10);
                    printf("Error allocate p in display_zones()\n");
                    getch();
                    return(0);
                  }/*end malloo data_o*/
          temp_o=data_o;

          fseek(tp_o,(long)img_width_o*y1_o,SEEK_SET);
          for(i=0;i<zoneheight_o;i++) {
                    getrow_o(tp_o,img_width_o);
                    for(j=0;j<zonewidth_o;j++) {
                                ptr[i*zonewidth_o+j]=data_o[j+x1_o];
                              }/* end loop j*/
                    }
          block_ohar_o(ptr,zonewidth_o,zoneheight_o);
          sequence_ohar_o(ptr,zonewidth_o,zoneheight_o);
          sort_ohar_o(zonewidth_o,zoneheight_o);

          free(data_o);
          /*data_o=NULL;  */  /***put NULL Value***/
         /* ptr=NULL;*/
          free(ptr);
         /* ptr=NULL;*/
          return(1);
}

int getrow_o(FILE *tp_o,int wide)
/*FILE *tp_o;
int wide;*/
{
int k;
        data_o=temp_o;
        for(k=0;k<wide;k++)
                {
                fread(data_o,sizeof(ohar),1,tp_o);
```

```
                data_o++;
                }
        data_o=temp_o;
        return(0);
}

void block_char_o(char  *image,int dx,int dy)
{
        int i,j,x,y,soore;
        char  *line,*pixel;
        int coarseness;

        for(y=1,line=image+dx ; y < dy-1 ; y++, line+=dx) {
            /*   indicator(" Block zones ",y);*/
/* walk through buffer scanning for neighbors */
                for (x=1, pixel=line+1; x < dx-1 ; x++, pixel++) {
                        soore = 0;
                        if(*(pixel-1) & 0x01) /*ALIVE         /* left */
                                soore++;
                        if(*(pixel+1) & 0x01)/*ALIVE)         /* right */
                                soore++;
                        if(*(pixel-dx) & 0x01)/*ALIVE)        /* up */
                                soore++;
                        if((*pixel == 0x00) && (soore >= 2))/*LIFE_SCORE))/*0x00 is WHITE */
                                *pixel = 0xf0;
                        else if((*pixel == 0x01) && (soore <= 0))/*DEATH_SCORE))*/
                                *pixel = 0x0f;
                }
        }

                        for(pixel = image ; pixel < image + dy * dx; pixel++) {
                /* birth and bury */
                if(*pixel == 0xf0)/*PREGNANT)*/
                        *pixel = 0x01;
                else if (*pixel == 0x0f) /*SICK*/
                        *pixel = 0x00;
}
                  /*coarseness=(int) (dx*0.25);*/
                coarseness=3;
                if(coarseness <= 0)             /* no need to coalesce */
                return;
        /* coalesce regions, based on coarseness, i.e. coarseness == 2
         * will close all horizontal/vertical gaps of 2 pixels */
        for(y=0, line=image+1 ; y < dy ; y++, line+=dx) {
                /* coalesce horizontally */
                for(x=1, pixel=line ; x < dx-coarseness ; x++,pixel++) {
                        if((*pixel == 0x00) && (*(pixel-1) == 0x01)) {
                                for(i =1 ; i <= coarseness ; i++) {
                                        if(*(pixel+i)) {
                                        for(i=0 ; i < coarseness ; i++, pixel++)
                                                        *pixel = 0xff;/*BLACK*/
                                                pixel--;
                                                x+=coarseness-1;
                                                break;
                                        }
```

```
                                }
                        }
                }
        }

        return;
}

void sequence_char_o(char  *image,int dx,int dy)
{
        int x,y;
        char  *tr;

        for(y=0; y < dy && char_count_o < 200 ; y+=1) {/*MAX_ZONE and MIN_Y_ZONE*/
                /* extract zones form block images in y order */
tr = image + y * dx;
                for (x =0 ; x < dx ; x += 1)/* MIN_X_ZONE*/
                        if (*(tr+x)) { /* found point */
                                if (char_count_o >= 100)/*MAX_ZONE*/
                                        break;
                                while (x > 0 && *(tr+x-1))      /* back up to left side */
                                        x--;
                                if(extract_char_o(image,dx,dy,x,y,zone_char_o+char_count_o)){
                                        char_count_o++;   /* get zone */

                                }
                        }
        }
        /* remove overlaping zones */
        overlap_char_o(zone_char_o,&char_count_o);
        return;
}
void overlap_char_o(ZONE *zone_array,int *array_size)
{
        int i,j,ax1,ay1,ax2,ay2,bx1,by1,bx2,by2;

        for (i = 0 ; i < *array_size-1 ; i++)
        {       /* compare each zone against the rest */
             /*   indicator(" Overlap zones",j);*/
                ax1 = (zone_array+i)->x1;
                ay1 = (zone_array+i)->y1;
                ax2 = (zone_array+i)->x2;
                ay2 = (zone_array+i)->y2;
                for (j = i+1 ; j < *array_size ; j++)
                {
                        bx1 = (zone_array+j)->x1;
                        by1 = (zone_array+j)->y1;
                        bx2 = (zone_array+j)->x2;
                        by2 = (zone_array+j)->y2;
                        if ((bx1>=ax2) || (ax1>=bx2) || (by1>=ay2) || (ay1>=by2))
                                continue;       /* no overlap */
                        bx1 = min(bx1,ax1);     /* combine zones */
                        by1 = min(by1,ay1);
                        bx2 = max(bx2,ax2);
                        by2 = max(by2,ay2);
```

```
                        (zone_array+i)->x1 = bx1;
                        (zone_array+i)->y1 = by1;
                        (zone_array+i)->x2 = bx2;
                        (zone_array+i)->y2 = by2;
                        (*array_size)--;        /* new zone count */
                        memmove((char *) (zone_array+j),(char *) (zone_array+j+1),
                                sizeof(ZONE)*(*array_size-j));  /* shift array to remove zone */
                        i = -1;         /* start all over */
                        break;
                }
        }
        return;
}

void sort_char_c(int dx,int dy)
{
        int x,y,i,j,index_c,fudge_x,fudge_y;
        ZONE temp_c;

for (i = 0 ; i < char_count_c-1 ;i++) {
/*indicator(" Sorting zones ",z+i);*/
/* sort on x1 */
for (j = i+1 ; j < char_count_c ; j++) {
if (zone_char_c[j].x1 < zone_char_c[i].x1) {
temp_c = zone_char_c[i];
zone_char_c[i] = zone_char_c[j];
                        zone_char_c[j] = temp_c;
                }
        }
        for (i = 0 ; i < char_count_c-1 ; i++) {
                /* order zones left to right, up to down */
                x = zone_char_c[i].x2;
                y = zone_char_c[i].y1;
                fudge_x = zone_char_c[i].x1+dx/20;      /* 5% slippage on alignment */
                for (j = i+1, index_c = -1 ; j < char_count_c ; j++) {
                        /* find any zones above and within x extsnt of zone_char_c_c[i] */
                        if (zone_char_c[j].x1 > x)
                                break;
                        if ((zone_char_c[j].y1 < y) && (zone_char_c[j].x1 <= fudge_x)) {
                                x = zone_char_c[j].x2;
                                y = zone_char_c[j].y1;
                                index_c = j;
                        }
                }
                if (index_c != -1) {
                        temp_c = zone_char_c[i];
                        zone_char_c[i] = zone_char_c[index_c];
                        zone_char_c[index_c] = temp_c;
                }
        }
}
 int extract_char_c(char  *image,int dx,int dy,int x,int y,ZONE *zone_ptr)
{
        typedef enum {GOING_UP,GOING_RIGHT,GOING_DOWN,GOING_LEFT } HEADING;
```

```
int ix,iy,minx,miny,maxx,maxy;  /* perimeter variables & min/max values */
HEADING dir;                    /* current direction */
char  *previous,*next,*here;    /* buffer pointers */

minx = maxx = ix = x;           /* preset min/max x/y and perimeter vars */
miny = maxy = iy = y;
dir = GOING_UP;                 /* starting direction */
do      /* walk perimeter, recording min/max of rectangular region */
{
        if (ix < minx)          /* update min/max */
                minx = ix;
        if (ix > maxx)
                maxx = ix;
        if (iy < miny)
                miny = iy;
        if (iy > maxy)
                maxy = iy;
        here = image + iy * dx + ix;    /* where are we? */
        next = here + dx;
        previous = here - dx;
        switch (dir)            /* base on current direction. */
        {
                case GOING_UP:
                if (ix > 0 && *(here-1))
                {
                        ix--;
                        dir = GOING_LEFT;
                        break;
                }
                if (iy > 0 && *previous)
                {
                        iy--;
                        break;
                }
                if (ix < dx-1 && *(here+1))
                {
                        ix++;
                        dir = GOING_RIGHT;
                        break;
                }
                if (iy < dy-1 && *next)
                {
                        iy++;
                        dir = GOING_DOWN;
                        break;
                }
                break;
                case GOING_RIGHT:
                if (iy > 0 && *previous)
                {
                        iy--;
                        dir = GOING_UP;
                        break;
                }
                if (ix < dx-1 && *(here+1))
```

```
{
        ix++;
        break;
}
if (iy < dy-1 && *next)
{
        iy++;
        dir = GOING_DOWN;
        break;
}
if (ix > 0 && *(here-1))
{
        ix--;
        dir = GOING_LEFT;
        break;
}
break;
oase GOING_DOWN:
if (ix < dx-1 && *(here+1))
{
        ix++;
        dir = GOING_RIGHT;
        break;
}
if (iy < dy-1 && *next)
{
        iy++;
        break;
}
if (ix > 0 && *(here-1))
{
        ix--;
        dir = GOING_LEFT;
        break;
}
if (iy > 0 && *previous)
{
        iy--;
        dir = GOING_UP;
        break;
}
break;
oase GOING_LEFT:
if (iy < dy-1 && *next)
{
        iy++;
        dir = GOING_DOWN;
        break;
}
if (ix > 0 && *(here-1))
{
        ix--;
        break;
}
if (iy > 0 && *previous)
```

```
                        {
                                iy--;
                                dir = GOING_UP;
                                break;
                        }
                        if (ix < dx-1 && *(here+1))
                        {
                                ix++;
                                dir = GOING_RIGHT;
                                break;
                        }
                        break;
                }
        } while (ix != x || iy != y);   /* until we return to the start */
        for (iy=miny, here=image+miny*dx+minx; iy<=maxy+3; iy++,here+=dx)
                memset(here,0x00,maxx-minx+3); /* white out the region */
        if ((maxx-minx+1 < 3) || (maxy-miny+1 < 3)) /*MIN_X_ZONE and MIN_Y_ZONE*/
                return 0;               /* big enough? */
        if (minx > 0)                   /* expand dimensions by one pixel */
                minx--;
        if (maxx < dx-1)
                maxx++;
        if (miny > 0)
                miny--;
        if (maxy < dy-1)
                maxy++;
        zone_ptr->x1 = minx+x1_o;           /* save zone */
        zone_ptr->y1 = miny+y1_o+combine_box_o;
        zone_ptr->x2 = maxx+x1_o;
        zone_ptr->y2 = maxy+y1_o+combine_box_o;
        zone_ptr->wide=(maxx-minx)+1;
        return 1;
}

int out_row(void)
{
int a,b,n;
long fpos;
int zone_depth=0;
int temp_o_o;
int temp_h;
int check_module;
FILE *fzonelis,*fzonechr;
        setcolor(RED);
        rectangle(165,50,400,90);
        gotoxy(26,5);
        printf("Process  Cut Character");
        if((fzonelis=fopen("zonechar.txt","rb"))==NULL) {
                rectangle(170,120,450,170);
                gotoxy(24,10);
                printf("Error Open File zonelist.txt\n");
                getch();
                return(0);
            }
                fread(&zone_count_o,sizeof(int),1,fzonelis);
```

```
                fread(&zone_depth,sizeof(int),1,fzonelis);
                read_info_o(fzonelis);
        if((fzoneohr=fopen("zoneohr1.txt","wb"))==NULL) {
                rectangle(170,120,450,170);
                gotoxy(24,10);
                printf("Error Open File zoneohr.txt\n");
                getch();
                return(0);
            }
                fwrite(&zone_count_o,sizeof(int),1,fzoneohr);
                fwrite(&zone_depth,sizeof(int),1,fzoneohr);

    setcolor(GREEN);
    for(n=0;n<zone_count_o;n++)
        {
        if((tp_o=fopen("tiftext.txt","r"))==NULL) {
                rectangle(170,120,450,170);
                gotoxy(24,10);
                printf("Error Open File tiftext.txt\n");
                getch();
                return(0);
                }
        if(type_o==2)
        {
                char_count_o=1;
                type_o=2;
                write_info_o(fzoneohr);
                read_info_o(fzonelis);
                fclose(tp_o);
        continue;
        }
        else type_o=1;
        for(b=0;b<count_o;b++)
                {
                check_module=read_char_o(b,tp_o);
                if(check_module==0).
                        {
                        rectangle(170,120,450,170);
                        gotoxy(24,10);
                        printf("Error in Cut Row of Character");
                        getch();
                        return(0);
                        }
                data_o=NULL;
                }
        fclose(tp_o);
     /* tp_o=NULL;*/
        for(a=0;a<char_count_o;a++)
                char_wide_o+= (5+zone_char_o[a].wide);/* 5 is blank in char*/

        type_o=1;/*this is a text*/
        zoneheight_o=zone_char_o[0].y2-zone_char_o[0].y1; /*define zoneheight_o*/
        for(a=1;a<char_count_o;a++) {
                if(zoneheight_o < (zone_char_o[a].y2-zone_char_o[a].y1))
                        zoneheight_o=zone_char_o[a].y2-zone_char_o[a].y1;
```

```
            }
                zoneheight_o+=10;/*Border box*/
        rectangle(145,100,420,130);
        gotoxy(22,8);
        printf("Row is Height %d  And width %d",zoneheight_o,char_wide_o);
        write_info_o(fzonechr);
        read_info_o(fzonelis);
        char_wide_o=0;
        for(a=0;a<char_count_o;a++){
                rectangle(100,200,500,250);
                delay(100);
                gotoxy(15,15);
        printf("minx is %d maxx is %d miny is %d maxy is  %d\n",zone_char_o[a].x1,zone_char_o[a].
x2,zone_char_o[a].y1,zone_char_o[a].y2);
        }
        rectangle(165,260,410,300);
        gotoxy(25,18);
        printf("Total Character %d in Row %d",char_count_o,n+1);
        char_count_o=0;
        zoneheight_o=zone_char_o[0].y2-zone_char_o[0].y1; /*define zoneheight_o*/
        char_count_o=0;
          }
        fclose(fzonelis);
        fclose(fzonechr);
    /*  fzonelis=fzonechr=NULL;*/
        return(1);
}

void write_info_o(FILE *fzonechr)
{
          fwrite(&char_count_o,sizeof(int),1,fzonechr);
          fwrite(&img_width_o,sizeof(int),1,fzonechr);
          fwrite(&char_wide_o,sizeof(int),1,fzonechr);
          fwrite(&zoneheight_o,sizeof(int),1,fzonechr);
          fwrite(zone_char_o,sizeof(ZONE),char_count_o,fzonechr);
          fwrite(&type_o,sizeof(int),1,fzonechr);
}

void read_info_o(FILE *fzonelis)
{
int temp_o_o,temp_h;
          fread(&count_o,sizeof(int),1,fzonelis);
          fread(&img_width_o,sizeof(int),1,fzonelis);
          fread(&temp_o_o,sizeof(int),1,fzonelis);
          fread(&temp_h,sizeof(int),1,fzonelis);
          fread(char_out_first,sizeof(ZONE),count_o,fzonelis);
          fread(&type_o,sizeof(int),1,fzonelis);
}
```

SAVE.C

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <stdlib.h>
#include <string.h>
#include <alloc.h>
#include "zone.h"

 /*#include "wtwg.h" */

 /* extern variable & prototype*/
/* extern save(void);*/

  ZONE zone_list_f[120];
  int zone_count_f;
  int zone_depth_f;
  int char_count_f;
  int wide_char_f;
  int last_height_f=0;/*last height character*/
  int char_wide_f;
  int img_width_f;
  int zoneheight_f;
  int zonewidth_f;
  int type_f;
  FILE *tp_f,*fsave_f;

  int a;

/* extern WINDOW *zone_win; */

 int get_row_f(FILE *tp_f,int wide);

 /*local prototype */
 int savezone_f(FILE *tp_f,FILE *fzonelis);
 int savechar_f(int n,FILE *tp_f);
 int save(void);
 FILE *tp_f;

  char *data_f,*temp_f;

 int savezone_f(FILE *tp_f,FILE *fzonelis)
 {
          int i,n,key,ch,number,count=0;
          int z;
          char a,*set_zone;
          char buff[30],**choices;
          int check_module;

          gotoxy(1,1);
 /*       printf("Detected Zone Image ");
          sprintf(buff,"\n Detected %d Zone(s).",choices);

          puts(buff);*/
          if(char_wide_f==0) char_wide_f=1;
          if((set_zone=malloc(char_wide_f))==NULL){
```

```
        printf("Error allocate p in display_zones()\n");
        getch();
        return(0);
    }
    /*set block */
    memset(set_zone,0x00,char_wide_f);
    for(i=0;i<zoneheight_f;i++)
            fwrite(set_zone,sizeof(char),char_wide_f,fsave_f);
    set_zone=NULL;
    free(set_zone);
    if((data_f=malloc(img_width_f))==NULL) {
        printf("Error allocate data in display_zones()\n");
        getch();
        return(0);
    }
    temp_f=data_f;

            n=char_count_f;
            if(n< 0 || n>char_count_f)
                    return(0);
            else {
                for(z=1;z<2;z++) {

                    for(count=0;count<char_count_f;count++){
                        check_module=savechar_f(count,tp_f);
                        if(check_module==0){
                            printf("Error");
                            getch();
                            return(0);
                            }
                        }
                    /*fclose(tp_f);
                    fclose(fsave_f);
                    if((tp_f=fopen("tiftext.txt","r"))==NULL) {
                        printf("Error Open File tiftext.txt\n");
                        return(0);
                        }

                    if((fsave_f=fopen("tifchar.txt","a"))==NULL){
                            printf("Error Open File tiftext.txt\n");
                            return(0);
                            } */
    /*                      last_height_f+=zoneheight_f*.85;*/
                    wide_char_f=0;
                    /*read_info(fzonelis);*/
                    }
                    number=n;
            /*      printf("please any key");
                    a=getch();*/
                }

      /*free(choices);*/
    data_f=NULL;
        free(data_f);
        return(1);
```

```
}
int saveohar_f(int n,FILE *tp_f)
{       register int i,j,k;
        int x1,x2,y1,y2;
        ohar *p,buff[20],oh;
        int zonelength,maxx,maxy;/*int zonewidth_f*/
        float sx,sy,faot;
        int o,xp,yp,np;
        ohar black,white;

        rectangle(150,200,420,250);
        gotoxy(22,15);
        sprintf(buff,"SAVE Character no. %2d in Row %d\n ",n,a+1);
        puts(buff);

        x1=zone_list_f[n].x1;
        x2=zone_list_f[n].x2;
        y1=zone_list_f[n].y1;
        y2=zone_list_f[n].y2;
        zonewidth_f=x2-x1+1;
        zonelength=y2-y1+1;
        maxx=getmaxx();
        maxy=getmaxy();
        if(zonewidth_f > maxx ||zonelength > maxy) {
                sx=maxx / (float)zonewidth_f;
                sy=maxy / (float)zonelength;
                faot=(sx<sy) ? sx:sy;
                np=(faot<1)  ? 1:faot+0.9-(float)1;

                fseek(tp_f,(long)img_width_f*y1,SEEK_SET);
                for(j=0;j<zonelength;j++) {
                                get_row_f(tp_f,img_width_f);
                        yp=(float)j*faot+0.5;
                        for(i=0;i<zonewidth_f;i++) {
                                o=(data_f[x1+i] ==0x01) ? 0x00 :0xff;
                                xp=(float)i*faot+0.5;
                                putpixel(xp,yp,o);
                        }
                }
        }
        else {
        fseek(tp_f,(long)img_width_f*y1,SEEK_SET);
        for(j=0;j<zonelength;j++) {
                get_row_f(tp_f,img_width_f);
          fseek(fsave_f,wide_ohar_f+(ohar_wide_f*j)+(last_height_f*ohar_wide_f),SEEK_SET);
        for(i=0;i<zonewidth_f;i++) {
        if(data_f[x1+i] == 0x01) fwrite(0xff,sizeof(ohar),1,fsave_f);
        else fwrite(0x00,sizeof(ohar),1,fsave_f);/*black*/
                                o=(data_f[x1+i] ==0x01) ? 0x00 :0xff;
                                /*putpixel(i,j,o);*/
                        }
                }
                wide_ohar_f+=zonewidth_f+4;
        }
        return(1);
```

```
 }

get_row_f(FILE *tp_f,int wide)
/*FILE *tp_f;
int wide;*/
{
int k;
                fread(data_f,sizeof(char),wide,tp_f);
}

int save(void)
{
int fname;
char tifchar[10];
FILE *fzonelis;

        /*clrscr();*/
                rectangle(165,50,390,90);
         gotoxy(25,5);
         printf("Save Character To Image");

        if((fzonelis=fopen("zonechr1.txt","rb"))==NULL) {
                printf("Error Open File zonelist.txt\n");
                getch();
                return(0);
                }
        fread(&zone_count_f,sizeof(int),1,fzonelis);
        fread(&zone_depth_f,sizeof(int),1,fzonelis);
        for(a=0;a<zone_count_f;a++)
                {
                /*if(zone_list_f[a].type==2) continue;*/
                if((tp_f=fopen("tiftext.txt","r"))==NULL) {
                        printf("Error Open File tiftext.txt\n");
                        getch();
                        return(0);
                        }
        fname=a+1;
        itoa(fname,tifchar,10);
        if((fsave_f=fopen(tifchar,"w"))==NULL){
                printf("Error Open File tiftext.txt\n");
                getch();
                return(0);
                }
                read_info_f(fzonelis);
                if(type_f==2) {
                                folose(tp_f);
                                continue;
                                }
                wide_char_f=3;
                last_height_f=3;
                savezone_f(tp_f,fzonelis);
                folose(fsave_f);
                folose(tp_f);
        /*      fsave_f=tp_f=NULL;*/
                }
```

```
    /*  folose(fzonelis);*/
        fzonelis=NULL;
        /*folose(fsave_f);
        folose(tp_f);*/
        return(1);
}
read_info_f(FILE *fzonelis)
{
                fread(&ohar_ount_f,sizeof(int),1,fzonelis);
                fread(&img_width_f,sizeof(int),1,fzonelis);
                fread(&ohar_wide_f,sizeof(int),1,fzonelis);
                fread(&zoneheight_f,sizeof(int),1,fzonelis);
                fread(zone_list_f,sizeof(ZONE),ohar_ount_f,fzonelis);
                fread(&type_f,sizeof(int),1,fzonelis);
}
```

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <stdlib.h>
#include <string.h>
#include <alloc.h>
#include "zone.h"

extern  ZONE zone_list[60];
extern  int zone_count;
extern  int img_width;
extern FILE *fp;
extern  char *data,*temp;

/*  ZONE zone_list[60];
  int zone_count;
  int img_width;
 FILE *fp;
 char *data,*temp;*/

int getrow_d(FILE *fp,int wide);

int showzone(FILE *fp);
int dispzone(int n,FILE *fp);
int dspzone();

int showzone(FILE *fp)
   {
        int i,n,key,ch,number;
        char a;
        char buff[30];

        for(i=0;i<zone_count;i++) {

                }

        setcolor(RED);
        rectangle(195,50,430,90);
        gotoxy(33,5);
        printf("Display Zones");
        if((data=malloc(img_width))==NULL) {
          rectangle(170,120,450,170);
          gotoxy(24,10);
          printf("Error allocate data in display_zones()\n");
          return(0);
        }
        temp=data;
        do {
                setcolor(GREEN);
                rectangle(140,355,530,410);
                gotoxy(22,25);
                printf("Key Number <0> Or Ather Key To exit program");
                gotoxy(22,24);
                printf("Plese Select zone<1 to %d> :      ",zone_count);
                gotoxy(50,24);
```

```
                scanf("%d",&n);
                n=n-1;
                if(n< 0 || n>zone_count-1)
                        return(1);

                else {
                        dispzone(n,tp);
                }
        }while(key!=ESCAPE);
        free(data);
        return(1);
}
int dispzone(int n,FILE *tp)
{       register int i,j,k;
        int x1,x2,y1,y2;
        char *p,buff[20],ch,a;
        int zonewidth,zonelength,maxx,maxy;
        float sx,sy,fact;
        int c,xp,yp,np;
        int center_x,center_y;

        x1=zone_list[n].x1;
        x2=zone_list[n].x2;
        y1=zone_list[n].y1;
        y2=zone_list[n].y2;
        zonewidth=x2-x1+1;
        zonelength=y2-y1+1;
        maxx=getmaxx();
        maxy=getmaxy();
        if(zonewidth > maxx ||zonelength > maxy) {
                maxx-=100;/***set image in Border on Soreen****/
                maxy-=100;
                sx=maxx / (float)zonewidth;
                sy=maxy / (float)zonelength;
                fact=(sx<sy) ? sx:sy;
                fseek(tp,(long)img_width*y1,SEEK_SET);
                center_x=(maxx/2)-((zonewidth*fact)/2);
                center_y=(maxy/2)-((zonelength*fact)/2);
                for(j=0;j<zonelength;j++) {
                                getrow_c(tp,img_width);
                        yp=(float)j*fact+0.5;
                        for(i=0;i<zonewidth;i++) {
                                c=(data[x1+i] ==0x01) ? 0x00 :0xff;
                                xp=(float)i*fact+0.5;
                                putpixel(xp+20,yp+200,c);
                        }
                }
            gotoxy(33,5);
                        if(zone_list[n].type==1)
                                printf("  It is Text        ");
                        if(zone_list[n].type==2)
                                printf("  It is Picture     ");
                /**********Clear Soreen*******/
                getch();
                for(j=0;j<zonelength;j++) {
```

```
                        yp=(float)j*faot+0.5;
                        for(i=0;i<zonewidth;i++) {
                        xp=(float)i*faot+0.5;
                                putpixel(xp+20,yp+200,0xff);
                        }
                }
        }
        else {

                center_x=(maxx/2)-(zonewidth/2);
                center_y=(maxy/2)-(zonelength/2);
                /**********Box Screen***********/
                setcolor(GREEN);
                rectangle(center_x-2,center_y-2,center_x+zonewidth+2,center_y+zonelength+2);
                /*******************************/
                fseek(tp,(long)img_width*y1,SEEK_SET);
                for(j=0;j<zonelength;j++) {
                        getrow_d(tp,img_width);
                        for(i=0;i<zonewidth;i++) {
                                if(data[x1+i] == 0x01) putpixel(i+center_x,j+center_y,0x00);
                                else putpixel(i+center_x,j+center_y,0xff);/*black*/
                        }
                }
                gotoxy(33,5);
                        if(zone_list[n].type==1)
                                printf("  It is Text        ");
                        if(zone_list[n].type==2)
                                printf("  It is Picture     ");
                a=getch();
                /******Clear Screen*******************/
                for(j=0;j<zonelength+8;j++) {
                        for(i=0;i<zonewidth+8;i++) {
                                putpixel(i+center_x-4,j+center_y-4,0xff);/*black*/
                        }
        }

                /***************************/
        }
        return(1);
}

getrow_d(tp,wide)
FILE *tp;
int wide;
{
int k;
        data=temp;
        for(k=0;k<wide;k++)
                {
                fread(data,sizeof(char),1,tp);
                data++;
                }
        data=temp;
}
```

DSPZONE.C

```
int dspzone()
{
int a;
FILE *fzonelis;
int check_module;

        if((fzonelis=fopen("zonelist.txt","rb"))==NULL) {
                rectangle(170,120,450,170);
                gotoxy(24,10);
                printf("Error Open File zonelist.txt\n");
                return(0);
                }
        fread(&zone_count,sizeof(int),1,fzonelis);
        fread(&img_width,sizeof(int),1,fzonelis);
        fread(zone_list,sizeof(ZONE),zone_count,fzonelis);
        fclose(fzonelis);

        if((tp=fopen("tiftext.txt","r"))==NULL) {
                rectangle(170,120,450,170);
                gotoxy(24,10);
                printf("Error Open File tiftext.txt\n");
                return(0);
                }
        check_module=showzone(tp);
        if(check_module==0){
                return(0);
                }
        fclose(tp);
        return(1);
}
```

```
#inolude <stdio.h>
#inolude <oonio.h>
#inolude <graphios.h>
#inolude <stdlib.h>
#inolude <string.h>
#inolude <alloo.h>
#inolude "zone.h"

extern int img_width;

extern FILE *tp;
extern ohar *data,*temp;
extern ZONE zone_list[65];
extern int ohar_oount;
extern int type;

int row_oount;

int getrow_ohr(FILE *tp,int wide);
int showohar(FILE *tp);
int dispohar(int n,FILE *tp);
int dspohr();


int showohar(FILE *tp)
    {
        int i,n,key,oh,number;
        ohar a;
        ohar buff[30];

        if((data=malloo(img_width))==NULL) {
            reotangle(170,120,450,170);
            gotoxy(24,10);
            printf("Error allooate p in display_zones()\n");
            getoh();
            return(0);
        }
        temp=data;
        do {
                gotoxy(22,23);
                printf("Plese Seleot oharaoter<1 to %d>:    ",ohar_oount);
                gotoxy(55,23);
                n=0;
                soanf("%d",&n);
                n=n-1;
                if(n< 0 || n>ohar_oount){}
                        else  {
                                dispohar(n,tp);
                                /*a=getoh();*/
                                }
        }while(n>=0);
        free(data);
        data=NULL;
        return(1);
}
```

```
int dispohar(int n,FILE *tp)
 {      register int i,j,k;
        int x1,x2,y1,y2;
        ohar *p,buff[20],oh;
        int zonewidth,zonelength,maxx,maxy;
        float sx,sy,faot;
        int o,xp,yp,np;
        int center_x,center_y;

        gotoxy(33,22);
        printf("Char no. %d\n ",n);

        x1=zone_list[n].x1;
        x2=zone_list[n].x2;
        y1=zone_list[n].y1;
        y2=zone_list[n].y2;
        zonewidth=x2-x1+1;
        zonelength=y2-y1+1;
        maxx=getmaxx();
        maxy=getmaxy();
        center_x=(maxx/2) -( zonewidth/2);
        center_y=(maxy/2) -( zonelength/2);
        if(zonewidth > maxx ||zonelength > maxy) {
                sx=maxx / (float)zonewidth;
                sy=maxy / (float)zonelength;
                faot=(sx<sy) ? sx:sy;
                np=(faot<1)  ? 1:faot+0.9-(float)1;

                olrsor();
                fseek(tp,(long)img_width*y1,SEEK_SET);
                for(j=0;j<zonelength;j++) {
                                getrow_ohr(tp,img_width);
                        yp=(float)j*faot+0.5;
                        for(i=0;i<zonewidth;i++) {
                                o=(data[x1+i] ==0x01) ? 0x00 :0xff;
                                xp=(float)i*faot+0.5;
                                putpixel(xp,yp,o);
                        }
                }
        }
        else {
            fseek(tp,(long)img_width*y1,SEEK_SET);
            setoolor(GREEN);
            reotangle(center_x-1,center_y-1,center_x+zonewidth,center_y+zonelength);
            for(j=0;j<zonelength;j++) {
                        getrow_ohr(tp,img_width);
                    for(i=0;i<zonewidth;i++) {
                            if(data[x1+i] == 0x01) putpixel(i+center_x,j+center_y,0x00);
                            else putpixel(i+center_x,j+center_y,0xff);/*blaok*/
                    }
                }/*end loop j*/
                /*olear Soreen*/
                getoh();
                for(j=0;j<zonelength+8;j++) {
                        for(i=0;i<zonewidth+8;i++) {
```

```
                                        putpixel(i+center_x-4,j+center_y-4,0xff);/*black*/
                                }
                        }/*end loop j*/

                }
                return(1);
 }

getrow_chr(tp,wide)
FILE *tp;
int wide;
{
int k;
                fread(data,sizeof(char),wide,tp);
}


int dspchr()
{
int a,b;
int char_wide;
int zoneheight;
int zone_depth;
FILE *fzonelis;
int check_module;

                setcolor(RED);
        rectangle(195,50,430,90);
        gotoxy(33,5);
        printf("Display Chararter");

        if((fzonelis=fopen("zonechr1.txt","rb"))==NULL) {
                rectangle(170,120,450,170);
                gotoxy(24,10);
                printf("Error Open File zonelist.txt\n");
                getch();
                return(0);
                }
        fread(&row_count,sizeof(int),1,fzonelis);
        fread(&zone_depth,sizeof(int),1,fzonelis);

        if((tp=fopen("tiftext.txt","r"))==NULL) {
                rectangle(170,120,450,170);
                gotoxy(24,10);
                printf("Error Open File tiftext.txt\n");
                getch();
                return(0);
                }
        for(b=0;b<row_count;b++)
                {
                fread(&char_count,sizeof(int),1,fzonelis);
                fread(&img_width,sizeof(int),1,fzonelis);
                fread(&char_wide,sizeof(int),1,fzonelis);
                fread(&zoneheight,sizeof(int),1,fzonelis);
                fread(zone_list,sizeof(ZONE),char_count,fzonelis);
```

```
                fread(&type,sizeof(int),1,fzonelis);
                /*********if picture to back for loop **********/

                if(type==2) continue;
                /***********************************************/

                setcolor(GREEN);
                rectangle(140,335,530,410);
                gotoxy(22,24);
                printf("Key Number <0>  To Next Row");
                gotoxy(22,22);
                printf("Row No. %d\n",b+1);
                check_module=showchar(tp);
                if(check_module==0){
                        return(0);
                        }
                }
        fclose(fzonelis);
        fclose(tp);
        return(1);
}
```

DSPALL.C

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <stdlib.h>
#include <string.h>
#include <alloc.h>
#include "zone.h"

 /* extern variable & prototype */
extern int zone_count;
extern int type;
extern char *data,*temp;
extern FILE *tp;
extern int char_count;
extern int img_width;

int zone_depth;
int char_width;
int zone_height;
ZONE zonelis[65];

int getrow_a(FILE *tp,int char_width);
int showall(FILE *tp);
void open_zone_char(FILE *fzonechr);
/*int dspzone();*/
int dspall();


int showall(FILE *tp)
 {
        int i,j,n,key,ch,number;
        int maxx,maxy,xp,yp,sx,sy;
        float fact;
        if(char_width==0) return(1);
        if((data=(char *)malloc(char_width))==NULL) {
            printf("Error allocate data in display_zones()\n");
            return(0);
        }
        temp=data;
        maxx=getmaxx();
        maxy=getmaxy();
        xp=(maxx/2)-(char_width/2);
        yp=(maxy/2)-(zone_height/2);

        if(char_width > maxx ||zone_height > maxy) {
        /*      sx=char_width/(float)maxx;
                sy=zone_height/(float)maxy;
                fact=(sx<sy) ? sx:sy;       */
                fact=0.75;
                xp=(maxx/2)-((char_width)/2)*fact;
                yp=(maxy/2)-((zone_height)/2)*fact;
                for(j=0;j<zone_height;j++) {
                                getrow_a(tp,char_width);
                        yp=(float)j*fact+0.5;
                        for(i=0;i<char_width;i++) {
```

```
                                xp=(float)i*faot+0.5;
                                        if(data[i] == 0x01) putpixel(xp+50,yp+220,0x00);/*black**/
                                else putpixel(xp+50,yp+220,0xff);/*white*/
                        }
                }
                /*****************************/
                getch();
                for(j=0;j<zone_height;j++) {
                        yp=(float)j*faot+0.5;
                        for(i=0;i<char_width;i++) {
                                xp=(float)i*faot+0.5;
                                putpixel(xp+50,yp+220,0xff);/*White**/
                        }
                }
                /*****************************/
        }
        else {

        setcolor(GREEN);
        rectangle(xp-1,yp-1,xp+char_width,yp+zone_height);
                for(j=0;j<zone_height;j++) {
                                getrow_a(tp,char_width);
                        for(i=0;i<char_width;i++) {
                                if(data[i] == 0x01) putpixel(i+xp,j+yp,0x00);/*black**/
                                else putpixel(i+xp,j+yp,0xff);/*white*/
                        }/*** end for ***/
                }/******end j ***/
        /**********clear Screen*********/
        getch();
        for(j=0;j<zone_height+8;j++) {
                        for(i=0;i<char_width+8;i++) {
                                putpixel(i+xp-4,j+yp-4,0xff); /****White*****/
                        }
                }
        /********end clear Screen***************/
}
        return(1);
}

getrow_a(tp,char_width)
FILE *tp;
int char_width;
{
int k;
        data=temp;
        for(k=0;k<char_width;k++)
                {
                fread(data,sizeof(char),1,tp);
                data++;
                }
        data=temp;
}

int dspall()
{
```

```
int a;
int j,k;
int fpos1,fpos2;
int check_module;
char fsave[10];
FILE *fzonelis,*fzonechr;
        setcolor(RED);
        rectangle(180,50,445,90);
        gotoxy(25,5);
        printf("Display Character After Sorting");
        if((fzonechr=fopen("zonechr1.txt","rb"))==NULL) {
                printf("Error open File zonechar.txt\n");
                return(0);
                }
        fread(&zone_count,sizeof(int),1,fzonechr);
        fread(&zone_depth,sizeof(int),1,fzonechr);
    for(j=0;j<zone_count;j++)
        {
                open_zone_char(fzonechr);
                if(type==2) continue;
                k=j+1;
                itoa(k,fsave,10);
                if((fp=fopen(fsave,"r"))==NULL) {
                        printf("Error Open File %d\n",fsave);
                        return(0);
                        }

                setcolor(GREEN);
                rectangle(140,355,530,410);
                gotoxy(22,25);
                printf("Prease Any Key To next row");
                gotoxy(22,24);
                printf("Row No. %d\n",j+1);
                check_module=showall(fp);
                if(check_module==0){
                        return(0);
                        }
                fclose(fp);
                getch();
                }
                return(1);

}
void open_zone_char(FILE *fzonechr)
{

        fread(&char_count,sizeof(int),1,fzonechr);
        fread(&img_width,sizeof(int),1,fzonechr);
        fread(&char_width,sizeof(int),1,fzonechr);
        fread(&zone_height,sizeof(int),1,fzonechr);
        fread(zonelis,sizeof(ZONE),char_count,fzonechr);
        fread(&type,sizeof(int),1,fzonechr);
}
```

```
#include <dos.h>
#include <stdio.h>
#include <alloc.h>
#include <string.h>
#include <ctype.h>
#include <conio.h>
#include <io.h>
#include <stdlib.h>
#include <fcntl.h>
#include <graphics.h>
#include "mouse16.h"
#include "fex_e.h"

#define FONTDEPTH  19

/*
union REGS inreg,outreg;
typedef struct { unsigned *image,xkey,ykey;} g_cursor;

struct  {
            char ff_reserved[21];
            char ff_attrib;
            int ff_ftime;
            int ff_fdate;
            long ff_fsize;
            char ff_name;
        } ffblk;

static unsigned arrow_image[32] =
  { 0xFE3F,0xFC7F,0xF87F,0xF0FF,
    0xE0FF,0xC000,0x8000,0x0000,
    0x8000,0xC000,0xE0FF,0xF0FF,
    0xF87F,0xFC7F,0xFF3F,0xFFFF,
    0x0080,0x0100,0x0300,0x0600,
    0x0E00,0x1C00,0x3FFF,0x7FFF,
    0x3FFF,0x1C00,0x0E00,0x0600,
    0x0300,0x0100,0x0080,0x0000
    };

static g_cursor ARROW = {NULL,1,7};

static unsigned check_image[32] =
  { 0xFFF0,0xFFE0,0xFFC1,0xFF83,
    0xFF07,0x060F,0x001F,0x803F,
    0xC07F,0xE0FF,0xF1FF,0xFBFF,
    0xFFFF,0xFFFF,0xFFFF,0xFFFF,
    0x0003,0x0006,0x000C,0x0018,
    0x0030,0x0060,0x70C0,0x3980,
    0x1F00,0x0E00,0x0400,0x0000,
    0x0000,0x0000,0x0000,0x0000
    };

static g_cursor CHECK = { NULL,5,10 };

static unsigned glove_image[32] =
```

```
  { 0xF3FF,0xE1FF,0xE1FF,0xE1FF,
    0xE1FF,0xE049,0xE000,0x8000,
    0x0000,0x0000,0x0000,0x0000,
    0x0000,0x0000,0x0000,0x8001,
    0x0000,0x0C00,0x0C00,0x0C00,
    0x0C00,0x0C00,0x0DB6,0x8DB6,
    0x6DB6,0x6FFE,0x6FFE,0x7FFE,
    0x7FFE,0x7FFE,0x7FFE,0x0000
    };

static g_oursor GLOVE = { NULL,4,0};

static unsigned rat_image[32] =
  { 0xFE7F,0xFF7F,0xF00F,0xE007,
    0xE007,0xE007,0xE007,0xE007,
    0xE007,0xE007,0xE007,0xE007,
    0xE007,0xE00F,0xF81F,0xFE7F,
    0x0000,0x0000,0x0000,0x0E70,
    0x0E70,0x0E70,0x0E70,0x0000,
    0x0FF0,0x0FF0,0x0FF0,0x0FF0,
    0x0FF0,0x07E0,0x0180,0x0000
    };

static g_oursor RAT = { NULL,7,0};
  */

int m_init (void);
int Mousehit(POINT *p);
int PointInReot(POINT *p,RECT *r);
void m_on(void);
void m_off(void);
void m_reset(void);
void m_move(int row,int ool);
void TraokButton(BUTTON *button);
void DrawButton(BUTTON *button);
void m_info(int *left,int *right,int *row,int *ool);
void m_limit(int left,int top,int right,int buttom);
void Fill3DBox(int left,int top,int right,int bottom);
void FillReot(int left,int top,int right,int bottom,int oolour);
void DrawLine(int left,int top,int right,int bottom,int oolour);
void MakeButton(int left,int top,ohar *string,BUTTON *button);

int m_init(void)
{
  union REGS regs;
  regs.x.ax=0;
  int86(0x33,&regs,&regs);
  return regs.x.ax;
}

void m_on(void)
{
  union REGS regs;
  regs.x.ax = 1;
  int86(0x33,&regs,&regs);
```

```
  return;
}

void m_off(void)
{
  union REGS regs;
  regs.x.ax = 2;
  int86(0x33,&regs,&regs);
  return;
}

void m_info (int *left,int *right,int *row,int *col)
{
  union REGS regs;
  regs.x.ax = 3;
  int86(0x33,&regs,&regs);
  *right = 0;
  *left  = 0;
  switch(regs.x.bx) {
     case 1 :
               *left  = 1;
               break;
     case 2 :
               *right = 1;
               break;
     case 3 :
               *right = 1;
               *left  = 1;
               break;
 }
 *row = regs.x.dx ;
 *col = regs.x.cx ;
 return;
}

void m_move (int row,int col)
{
  union REGS regs;
  regs.x.dx = row;
  regs.x.cx = col;
  regs.x.ax = 4;
  int86(0x33,&regs,&regs);
  return;
}

void m_limit (int left,int top,int right,int buttom)
{
  union REGS regs;
  regs.x.dx = left;
  regs.x.cx = right-10;
  regs.x.ax = 7;
  int86(0x33,&regs,&regs);
  regs.x.dx = buttom-10;
  regs.x.cx = top;
  regs.x.ax = 8;
```

```
  int86(0x33,&regs,&regs);
  return;
}

void m_reset (void)
{
  union REGS r;
  r.x.ax=0x21;
  int86(0x33,&r,&r);
  return;
}

int Mousehit(POINT *p)
{
  int left,right,row,col;
  m_info(&left,&right,&row,&col);
  p->x=col;
  p->y=row;
  return (left);
}

void FillRect(int left,int top,int right,int bottom,int colour)
{
  m_off();
  setfillstyle(SOLID_FILL,colour);
  bar(left,top,right,bottom);
  m_on();
  return;
}

void MakeButton(int left,int top,char *string,BUTTON *button)
{
  button->frame.left = left & 0xfff8;
  button->frame.right=button->frame.left+(strlen(string)<<3)+16;
  button->frame.top=top;
  button->frame.bottom=button->frame.top+FONTDEPTH+8;
  button->text=string;
  return;
}

void DrawButton(BUTTON *button)
{
  FillRect(button->frame.left+1,button->frame.top+1,
           button->frame.right-1,button->frame.bottom-1,WHITE);
  DrawLine(button->frame.left+1,button->frame.top,
           button->frame.right-1,button->frame.top,BLACK);
  DrawLine(button->frame.left,button->frame.top+1,
           button->frame.left,button->frame.bottom-1,BLACK);
  DrawLine(button->frame.right,button->frame.top+1,
           button->frame.right,button->frame.bottom-1,BLACK);
  DrawLine(button->frame.left+1,button->frame.bottom,
           button->frame.right-1,button->frame.bottom,BLACK);
  Fill3DBox(button->frame.left+1,button->frame.top+1,
           button->frame.right-1,button->frame.bottom-1);
  m_off();
```

```
  outthaibg(button->frame.left+8,button->frame.top+2,
              oolor_mt,oolor_mb,button->text);
  m_on();
  return;
}

void TraokButton(BUTTON *button)
{
  FillReot(button->frame.left+1,button->frame.top+1,
            button->frame.right-1,button->frame.bottom-1,oolor_mt);
  m_off();
  outthaibg(button->frame.left+8,button->frame.top+2,
            oolor_mt,oolor_mb,button->text);
  m_on();
  while(Mousehit(&p));
     DrawButton(button);
  return;
}

void Fill3DBox(int left,int top,int right,int bottom)
{
  FillReot(left,top,right,bottom,oolor_mt);
  DrawLine(left,top,right-1,top,WHITE);
  DrawLine(left,top+1,left,bottom-1,WHITE);
  DrawLine(left+3,bottom,right,bottom,LIGHTGRAY);
  DrawLine(right,top+2,right,bottom-1,LIGHTGRAY);
  return;
}

int PointInReot(POINT *p,RECT *r)
{
  if(p->x > r->left && p->x < r->right && p->y > r->top  && p->y < r->bottom)
     return(1);
  else
     return(0);
}

void DrawLine(int left,int top,int right,int bottom,int oolour)
{
  m_off();
  setoolor(oolour);
  line(left,top,right,bottom);
  m_on();
  return;
}
```

```
#include <stdio.h>
#include <alloc.h>
#include <graphics.h>
#include "mouse16.h"
#include "const.h"
#include "fex_e.h"

void clear_page(int n);
void RestoreWin();
void RestoreWin2();

void win_1(int left,int top,int right,int buttom,int t);
void win_2(int left,int top,int right,int buttom,int t);
int Choice_head(int x,int y,int count,char *sub[]);

extern void head_menu();

void far *buf,*buf2;
int left_e,top_e,left_2,top_2;
BUTTON file,about;

void win_1(int left,int top,int right,int buttom,int t)
{
  m_off();
  left_e=left;
  top_e=top;
  buf=farmalloc(imagesize(left-1,top-1,right+5,buttom+5));
  getimage(left-1,top-1,right+5,buttom+5,buf);
  setfillstyle(SOLID_FILL,BLACK);
  bar(left+5,top+5,right+5,buttom+5);
  bar(left-1,top-1,right+1,buttom+1);
  setfillstyle(SOLID_FILL,color_wall);
  bar(left,top,right,buttom);
  setcolor(BLACK);
  line(left+10,top+10,right-10,top+10);
  line(left+10,top+10,left+10,buttom-(10+t));
  setcolor(WHITE);
  line(left+10,buttom-(10+t),right-10,buttom-(10+t));
  line(right-10,top+10,right-10,buttom-(10+t));
  m_on();
  m_limit(left,top,right+10,buttom+10);
  return;
}

void win_2(int left,int top,int right,int buttom,int t)
{
  m_off();
  left_2=left;
  top_2=top;
  buf2=farmalloc(imagesize(left-1,top-1,right+5,buttom+5));
  getimage(left-1,top-1,right+5,buttom+5,buf2);

  setfillstyle(SOLID_FILL,0);
  bar(left+5,top+5,right+5,buttom+5);
```

```
  setfillstyle(SOLID_FILL,3);
  bar(left,top,right,buttom);

  setfillstyle(SOLID_FILL,7);
  bar(left+10,top+10,right-10,buttom-(10+t));

  setcolor(BLACK);
  line(left+10,top+10,right-10,top+10);
  line(left+10,top+10,left+10,buttom-(10+t));
  rectangle(left,top,right,buttom);

  setcolor(WHITE);
  line(left+10,buttom-(10+t),right-10,buttom-(10+t));
  line(right-10,top+10,right-10,buttom-(10+t));
  m_on();
  m_limit(left,top,right+10,buttom+10);
  return;
}

void RestoreWin()
{
/*  m_limit(LEFT,TOP,RIGHT+5,BOTTOM+5);*/
  m_off();
  putimage(left_e-1,top_e-1,buf,0);
  farfree(buf);
  m_on();
  return;
}

void RestoreWin2()
{
  m_limit(LEFT,TOP,RIGHT+5,BOTTOM+5);
  m_off();
  putimage(left_2-1,top_2-1,buf2,0);
  farfree(buf2);
  m_on();
  return;
}

int Choice_head(int x,int y,int count,char *sub[])
{
  BUTTON str[6];
  int t,i,j,left,old,st_left;
  int arow_choice,click;

  arow_choice=old=click=0;
  st_left=left=x;

        m_off();

/*  MakeButton(555,15," about ",&about);
  DrawButton(&about);
  MakeButton(16,15," file ",&file);
  DrawButton(&file);*/
```

```
for(i=0,j=x;i<count;i++,j+=100) {
  outthaibg(j,y,color_wall,1,sub[i]);
  MakeButton(j,y,"      ",&str[i]);
}
line(15,70,625,70);
outthaibg(x,y,6,1,sub[arow_choice]);
m_on();
do {
  t=0;
  if(kbhit()) {
      key.i = bioskey(0);
      if(key.ch[0]==13)                    /* ENTER */
        t=1;
      else if(key.ch[0]==27) {             /* ESC */
        arow_choice = 9;
        t=1;
      }
      else if(key.ch[1]==75) {             /* AR_LEFT */
        m_off();
        outthaibg(left,y,color_wall,1,sub[arow_choice]);
        arow_choice--;
        if(arow_choice<0)
          arow_choice = count-1;
        left=x+(arow_choice*100);
        outthaibg(left,y,6,1,sub[arow_choice]);
        m_on();
      }
      else if(key.ch[1]==77) {             /* AR_RIGHT */
        m_off();
        outthaibg(left,y,color_wall,1,sub[arow_choice]);
        arow_choice++;
        if(arow_choice>=count)
          arow_choice = 0;
        left=x+(arow_choice*100);
        outthaibg(left,y,6,1,sub[arow_choice]);
        m_on();
      }
      else
        t=0;
  }
  if(Mousehit(&p)) {
      for(i=0;i<count;i++) {
        if(PointInRect(&p,&str[i])) {
          m_off();
          outthaibg(left,y,6,1,sub[arow_choice]);
          arow_choice=i;
          left=x+(arow_choice*100);
          outthaibg(left,y,6,1,sub[arow_choice]);
          m_on();
        }
      }
/*    if(PointInRect(&p,&about)){          /* About */
 /*     TrackButton(&about);
      about_u();
      }
```

```
        else if(PointInRect(&p,&file)) {        /* Return To Menu */
  /*      TrackButton(&file);
          arow_choice = 9;
          t=1;
        } */
         if(old!=arow_choice) {
          m_off();
          outthaibg(left,y,6,1,sub[arow_choice]);
          outthaibg(st_left,y,color_wall,1,sub[old]);
          old=arow_choice;
          st_left=left;
          click = 0;
          m_on();
        }
        while(Mousehit(&p));
        if(PointInRect(&p,&str[arow_choice])) {
          click++;
          if(click>=2)
            t=1;
        }
    }
  }while(t==0);
  return(arow_choice);
}



void clear_page(int n)
{
  m_off();
  setfillstyle(SOLID_FILL,WHITE);
  if(n==1) {
    bar(16,45,624,464);
    setcolor(BLACK);
  }
  else {
    bar(16,78,624,435);
    setcolor(BLACK);
    line(15,436,625,436);
  }
  m_on();
  return;
}
```

```
#include <graphics.h>
#include <dos.h>
#include <stdio.h>
#include "v2var.h"

union k{
     char ch[2];
     int i;
     }key;

unsigned char font[256][20];

void outchar(int xx,int yy,unsigned char ch,int Color);
void outcharul(int x,int y,unsigned char ch,int Color);
void outthai(int x,int y,int Color,char *txt);
void outthaibg(int x, int y,int ColorB,int ColorT,unsigned char  *txt);
char InitThai(void);

void outchar(int xx,int yy,unsigned char ch,int Color)
{
 int x,y,color;
 unsigned t = 0x80;
 for(y=0;y<20;y++)
 for(x=0,t=0x80;x<8;x++,t=t>>1)
    if(font[ch][y] & t)
      putpixel(xx+x,yy+y,Color);
}

void outcharul(int x,int y,unsigned char ch,int Color)
{
 int c = getcolor();
 setcolor(Color);
 line(x,y+TEXTHIGHT-5,x+TEXTWIDTH,y+TEXTHIGHT-5);
 outchar(x,y,ch,Color);
 setcolor(c);
}

void outthai(int x,int y,int Color,char *txt)
{
 int i=0;
 unsigned char ch;

 do {
    if (txt[i] == '') {
          i++;
          outcharul(x,y+1,txt[i],Color);
    }
 if (txt[i] != ' ') outchar(x,y+1,txt[i],Color);
 i++;
 ch = txt[i];
 if (ch == ' ') x+=8;
 else
   if (ch == 216 || ch == 217 || ch == 215
      || ch == 212 || ch == 213 || ch == 214
      || ch == 231 || ch == 233 || ch == 209
```

```
      || ch == 234 || ch == 235 || ch == 232
      || ch == 236) ;
    else  x+=8;
 }while(txt[i]!=0);
}

char InitThai(void)
{
 FILE *fp;
 if ((fp = fopen("NORMAL.FON","rb"))!=NULL) {
    fread(font,256,20,fp);
    fclose(fp);
    return (THAIINST);
 } else
   return (THAIUNINST);
}

void outthaibg(int x, int y,int ColorB,int ColorT,unsigned char  *txt)
{
 int i=0,x2=x;
 unsigned char ch = txt[0];
 for (i = 0;txt[i]!=0;i++) {
   ch = txt[i];
   if (ch == ' ') x2 +=8;
   else
    if (  ch == 216 || ch == 217 || ch == 215
          || ch == 212 || ch == 213 || ch == 214
          || ch == 231 || ch == 233 || ch == 209
          || ch == 234 || ch == 235 || ch == 232
          || ch == 236 || ch == '~') ;
      else  x2+=8;
 }
 setfillstyle(SOLID_FILL,ColorB);
 bar(x,y,x2,y+TEXTHIGHT);
 outthai(x,y,ColorT,txt);
}
```

```
#define MAX_MALLOC 0xffee /*maximum malloc  memmory size*/
#define BUFFER_SIZE 0xfe00

#define ESCAPE 0x1b
#define BACKSPACE 0x08
#define ENTER  0x0d
/*blocking */
#define WHITE 0x00      /*white color*/
#define BLACK 0x01      /*black color*/
#define PREGNANT 0xfo   /*ready to be black*/
#define SICK 0x0f       /*ready to be write */
#define ALIVE 0x01
#define LIFE_SCORE 2    /*neighbors for birth */
#define DEATH_SCORE 0   /*neighbors for death   */

/*sequencing*/
/*#define MIN_X_ZONE 1  /*minimumn allowable x extent for a zone*/
/*#define MIN_Y_ZONE 1    /*minimumn allowable y extent for a zone  */
#define MAX_ZONES 100   /*maximumn number of zone*/
#define COLUMN_MAJOR 1  /*order zone with columns dominate*/
#define ROW_MAJOR 2     /*order zone with row dominate*/

typedef struct {int x1,x2,y1,y2,wide,type; } ZONE;/*type 1 is picture*/
                                                  /*type 2 is text*/
typedef enum {GOING_UP,GOING_RIGHT,GOING_DOWN,GOING_LEFT } HEADING;
typedef struct {short x,y,width,height; } DISPLAY_BOX;
typedef struct {
                ZONE zone_info[100];/*MAX_ZONES*/
                int zones;
                int line_no;
                }SAMPLEINFO;
```

SOUND.C

```
#include <dos.h>
void sound1(void);
void sound2(void);
void soundburst(void);

void sound1()
{
   int a;

   for(a=0;a<10;a++)
          {
            sound(a*200);
            delay(100);
          }
        nosound();
}

void sound2()
{
   int a;

   for(a=20;a>0;a--)
          {
            sound(a*200);
            delay(5);
          }
        nosound();
}

void soundburst()
{
   int a;

   for(a=30;a>0;a--)
          {
            sound(a*200);
            delay(7);
          }
        nosound();
}
```

READ.H

```
#define GOOD_READ      0
#define GOOD_WRITE     0
#define BAD_FILE       1
#define BAD_WRITE      1
#define  BAD_READ      2
#define MEMORY_ERROR          3
#define WRONG_BITS     4

#define SCREENWIDE     640     //mode11,2 color CGA screen dimention
#define SCREENDEEP     480
#define STEP           64      //size of a step when panning

#define RGB_RED                0
#define RGB_GREEN      1
#define RGB_BLUE       2
#define RGB_SIZE       3

#define HOME           0x4700
#define CURSOR_UP      0x4800
#define CURSOR_LEFT    0x4b00
#define CURSOR_RIGHT   0x4d00
#define END            0x4f00
#define CURSOR_DOWN    0x5000

/*  */
#define TIFFbyte  1
#define TIFFasoii  2
#define TIFFshort  3
#define TIFFlong  4
#define TIFFrational      5

/*  */
#define NewSubFile          254
#define SubfileType         255
#define ImageWidth          256
#define ImageLength         257
#define RowsPerStrip        278
#define StripOffsets        273
#define StripByteCounts     279
#define SamplesPerPixel 277
#define BitsPerSample   258
#define Compression         259
#define PlanarConfiguration 284
#define Group3Option        292
#define Group4Option        293
#define FillOrder  266
#define  Threshholding      263
#define CellWidth 264
#define CellLength          265
#define MinSampleValue      280
#define MaxSampleValue      281
#define PhotometricInterp   262
#define GrayResponseUnit    290
#define GrayResponseCurve 291
#define ColorResponseUnit 300
```

READ.H

```
#define ColorResponseCurves        301
#define XResolution        282
#define YResolution        283
#define  ResolutionUnit    296

#define Orientation        274
#define DooumentName       269
#define PageName 285
#define XPosition  286
#define YPosition  287
#define PageNumber         297
#define ImageDesoription 270
#define Make               271
#define Model              272
#define FreeOffsets        288
#define FreeByteCounts     289
#define ColorMap 320
#define Artist             315
#define DateTime  306
#define HostComputer       316
#define Software   305

#define pixels2bytes(n)    ((n+7)/8)

typedef struct {
                    unsigned int width,depth,bytes,bits;
                    unsigned int flags;
                    unsigned int rowsperstrip;
                    unsigned int ocunt;
                    unsigned int samples;
                    unsigned int threshhold;
                    unsigned int ocmpression;
                    unsigned int bitspersample;
                    unsigned long bytecount;
                    unsigned long offset;
                    ohar palette[768];
                    int (*setup)();
                    int (*olosedown)();
          } FILEINFO;

ohar *buffer= NULL;
```

MSC.H

```
#ifndef __SMALL__
        #ifndef __LARGE__
                #ifndef __MEDIUM__
                        #ifndef __COMPACT__
                                #ifndef __HUGE__
                                        #ifndef __TINY__
                                                #define __MEDIUM__ 1
                                        #endif
                                #endif
                        #endif
                #endif
        #endif
#endif          /* end memory model indicator */


                #define PSEUDOREGS union REGS regs; struct SREGS sregs;
                #define _AX     regs.x.ax
                #define _AL     regs.h.al
                #define _AH     regs.h.ah
                #define _BX     regs.x.bx
                #define _BL     regs.h.bl
                #define _BH     regs.h.bh
                #define _CX     regs.x.cx
                #define _CL     regs.h.cl
                #define _CH     regs.h.ch
                #define _DX     regs.x.dx
                #define _DL     regs.h.dl
                #define _DH     regs.h.dh

                #define _ES     sregs.es
                #define _CS     sregs.cs
                #define _SS     sregs.ss
                #define _DS     sregs.ds

                /* Microsoft C can't access the flags.
                 */
                #undef _FLAGS

                #define INTERRUPT(intno) int86((intno), &regs, &regs)

                #define INTERRUPTX(intno) int86x ( (intno), &regs, &regs, &sregs )

/* interrupt handlers
 */
#define enable()            _enable()
#define disable()           _disable()
#define setvect(x,y)        _dos_setvect((x),(y))
#define getvect(x)          _dos_getvect((x))

/* In Line outport()
 */
#define outport(x,y)  outp((x),(y))
#define inport(x)     inp ((x))

/* Memory allocation.
```

```
 * uses different names from TurboC
 */
        #define farmalloc(nb)    halloc(nb, 1)
        #define farfree(x)               hfree(x)

/* make a far ptr given a seg & offset
 */
#define MK_FP(s,o) (void far *)(((unsigned long)(s)<<16)|((unsigned)(o)))

/* simple graphics calls
 */
#define line(a,b,c,d)           \
                        _moveto((a),(b)), _lineto((c),(d))
#define lineto(a,b)  _lineto ((a), (b))
#define setviewport(a,b,c,d,x)  \
        _setviewport((a),(b),(c),(d)), \
        if (x) {_setcliprgn ((a),(b),(c),(d)) }
#define putpixel(a,b,c)  _setcolor((c)), _setpixel ((a),(b))
#define getpixel(a,b)    _getpixel ((a),(b))
#define setcolor(x)      _setcolor((x))
```

SFEX.H

```
#define LEFT            10
#define  RIGHT          630
#define  TOP            10
#define  BOTTOM         470

extern POINT p;
extern BUTTON file,about;
extern int o_char,oheok,oheok_wall,oolor_mt,oolor_mb;
extern int oolor_wall,oolor_sel,oolor_bg;
extern int oolor_red,oolor_brown;
extern int char_M;
extern union z{
     char ch[2];
     int i;
     }key;

extern void m_on(void);
extern void m_off(void);
extern void clear_page(int n);
extern void func_logo();
extern void m_move (int row,int col);
extern void m_limit (int left,int top,int right,int buttom);
extern void about_u();
extern void UserInput(void);
extern int  Mousehit(POINT *p);
extern void DrawButton(BUTTON *button);
extern void TrackButton(BUTTON *button);
extern int  PointInRect(POINT *p,RECT *r);
extern void MakeButton(int left,int top,char *string,BUTTON *button);
extern void outthai(int x,int y,int Color,char *txt);
extern void outthaibg(int x, int y,int ColorB,int ColorT,unsigned char  *txt);

extern void win_1(int left,int top,int right,int buttom,int t);
extern void win_2(int left,int top,int right,int buttom,int t);

extern void RestoreWin();
extern void RestoreWin2();

extern void win_a(int x,int y,int l,int h,int s);
extern void win_r(int x,int y,int l,int h,int s,int i,int d);
extern void win_v(int x,int y,int s,int l,int h,int i,int d);
extern void win_i(int x,int y,int s,int l,int h,int i,int d);
extern void win_b(int x,int y,int l);
extern void win_l(int x,int y,int l,int h,int d);
extern void win_o(int x,int y,int l,int h,int s,int i,int d);
extern void win_k(int x,int y,int s);
extern void win_g(int x,int y);
```

กิตติกรรมประกาศ

ขอขอบพระคุณ รองศาสตราจารย์ ดร.ชม คิ้มปาน ที่ให้คำแนะนำและเป็นที่ปรึกษาในการแก้ปัญหา และให้ความอนุเคราะห์เครื่องไมโครคอมพิวเตอร์ เครื่องตรวจควาดภาพที่ใช้ในการทำปริญญานิพนธ์นี้ สิ่งหนึ่งที่จะไม่ได้ก็คือ ขอขอบคุณบิดา มารดา ที่ให้การสนับสนุนในการศึกษามาโดยตลอด และขอขอบคุณอาจารย์ทุกท่าน และเพื่อนๆ ทุกคนที่ให้ความช่วยเหลือและเป็นกำลังใจ ในการทำปริญญานิพนธ์จนสำเร็จลุล่วงด้วยดี

หนังสืออ้างอิง

1. ประสาน ตังติสานนท์, "การจดจำรูปแบบตัวอักษรคัดลายมือไทย โดยวิธีแยกลักษณะเด่น", วิทยานิพนธ์มหาบัณฑิต คณะวิศวกรรมสาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2529
2. สุรพันธ์ เอื้อไพบูลย์, "การจดจำรูปแบบตัวอักษรคัดลายมือไทย โดยการพิจารณาหัวของตัวอักษร", วิทยานิพนธ์มหาบัณฑิต คณะวิศวกรรมสาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2531
3. ชาย เกษมอมรกุล, "การออกแบบพจนานุกรมสำหรับการเรียนรู้อักษรลายมือไทย - อังกฤษอัตโนมัติบนเครื่องไมโครคอมพิวเตอร์", วิทยานิพนธ์มหาบัณฑิต คณะวิศวกรรมสาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2532
4. สุรสิทธิ์ ราตรี, "การรู้จำตัวอักษรตัวพิมพ์ภาษาไทย โดยวิธีค้นหาลักษณะโครงสร้างลายเส้น", วิทยานิพนธ์มหาบัณฑิต คณะวิศวกรรมสาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2532
5. สมศักดิ์ วลัยรัชต์, "การวิเคราะห์และระบุส่วนประกอบของหน้าเอกสารและการรู้จำตัวอักษร", วิทยานิพนธ์มหาบัณฑิต คณะวิศวกรรมสาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2537
6. วัชระ ฉัตรวิริยะ, "การแยกแยะตัวอักษรภาษาไทย โดยใช้ฟังก์ชั่นฮิสโตแกรมและหลักการตั้งสมมุติฐานและทดสอบ", วิทยานิพนธ์มหาบัณฑิต คณะวิศวกรรมสาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2537
7. ศิวัฒน์ ศิวะบวร, "การประยุกต์ใช้งานภาษา C", ซีเอ็ด, 608 หน้า, 2536
8. มัณฑนา ปราการสมุทร, "การเขียนชุดคำสั่งภาษา C", คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 314 หน้า, 2534
9. ธันวา ศรีประโมง, "การเขียนโปรแกรมภาษา C สำหรับวิศวกรรม", มหาวิทยาลัยเทคโนโลยีมหานคร , 740 หน้า, 2537
10. Steve Rimmer, "Supercharged Bitmaped Graphics", McGraw-Hill, 646 p. 1992