บทที่ 4

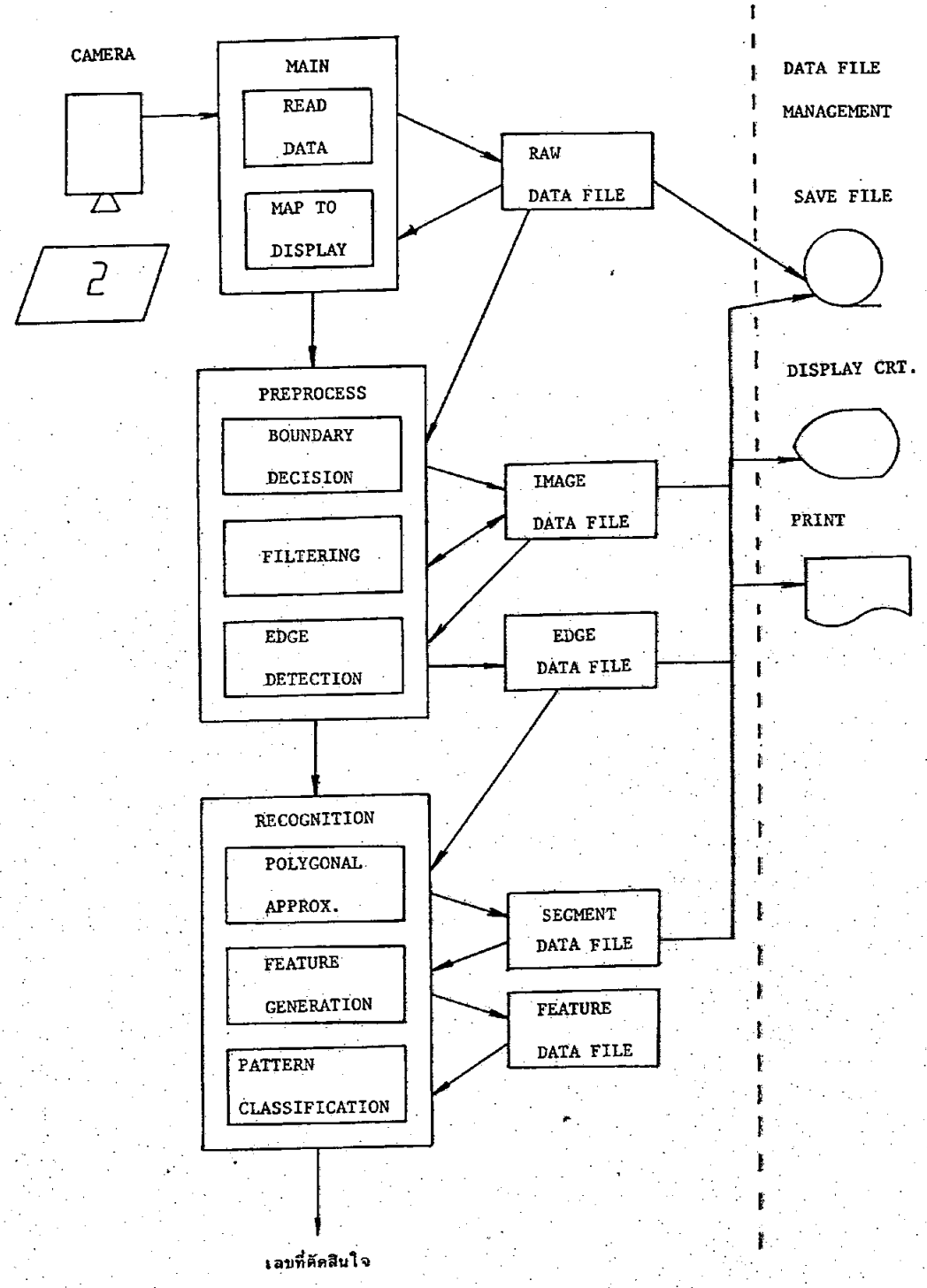
การออกแบบโปรแกรมระบบจดจำตัวเลขอารบิค

4.1 ลักษณะข้อมูลอินพุท

การออกแบบในส่วนโปรแกรมจดจำตัวเลขอารบิคมีความต้องการให้สามารถจำแนกทั้งตัวเลขอารบิคจากเครื่องพิมพ์ดีด และตัวเลขอารบิคเขียนด้วยมือ ดังนั้น ลักษณะของข้อมูลอินพุท (ตัวเลขอารบิค) จะมีได้ 2 แบบ คือ

1. ตัวเลขอารบิคจากเครื่องพิมพ์ดีด พิมพ์ลงกระดาษธรรมดา เนื่องจากตัวเลขจากเครื่องพิมพ์มีขนาดเส้นตัวพิมพ์เล็ก ดังนั้นการนำไปอ่านโดยเครื่อง OCR ที่สร้างขึ้น จะต้องนำตัวเลขพิมพ์นั้นไปขยายด้วยเครื่องถ่ายเอกสารให้มีขนาดเส้นตัวเลขใดขึ้น เพื่อป้องกันข้อผิดพลาดในการจำแนก เนื่องจากปัญหาการอ่านตัวเลขขนาดเล็กไป ทำให้เกิดเส้นบาง, เส้นแหว่งหรือขาด โดยได้เลือกขยายตัวพิมพ์ให้มีอัตราการขยายต่อพื้นที่ เพิ่มขึ้น 1.8 เท่าจากของเดิม

2. ตัวเลขอารบิคตัวเขียน จะทำการเขียนตัวเลขลงในกรอบสี่เหลี่ยม โดยในกรอบ 1 กรอบเขียนตัวเลขเพียงตัวเดียว สาเหตุที่ต้องเขียนลงในกรอบเพื่อต้องการกำหนดขอบเขตขนาดตัวอักษรใหญ่สุดที่สามารถอ่านเข้าไปเก็บในหน่วยความจำได้ ทั้งนี้เนื่องจากในการประมวลผลที่จะที่จะได้กล่าวต่อไป ข้อมูลบิทภาพจะถูกเก็บในลักษณะตัวแปรจำนวนเต็ม (integer) 2 มิติ ซึ่งในหน่วยความจำเครื่องไมโครคอมพิวเตอร์ APPLE II มีได้แค่ 64 KB ซึ่งเราต้องจัดสรรเป็น 2 ส่วน สำหรับข้อมูลภาพและโปรแกรมการทำงาน ดังนั้น ขนาดสูงสุดของตัวเลขจึงต้องมีขนาดจำกัด โดยในการออกแบบได้จองที่ไว้ให้ตัวเลขอารบิคมีขนาดได้สูงสุด 100 x 100 จุดภาพ



รูปที่ 4.1 แสดงโครงสร้างของระบบโปรแกรมจดจำตัวเลขอารบิค ที่เขียนขึ้น

4.2 โครงสร้างของระบบโปรแกรมจดจำตัวเลขอารบิค

โปรแกรมจดจำตัวเลขอารบิคส่วนใหญ่จะเขียนด้วยภาษาเบสิกโดยมีส่วนโปรแกรมย่อยภาษาแอสเซมบลีเพียงเล็กน้อย โครงสร้างของระบบโปรแกรมแสดงอยู่ในรูปที่ 4.1 จากในรูปส่วน DATA FILE MANAGMENT จะหมายถึงความสามารถในการจัดการกับไฟล์ข้อมูล เช่น สามารถเก็บลงดิสค์ แสดงผลออกทางจอภาพ พิมพ์ออกทางเครื่องพิมพ์ โปรแกรมจดจำตัวเลขอาริบิคสามารถแบ่งออกเป็น 3 ส่วนคือ

1. โปรแกรมหลัก (main) ประกอบด้วยโมดการทำงานดังนี้
   - READ DATA ควบคุมการอ่านข้อมูลบิทภาพจากกล้องวิดิคอนเข้ามาเก็บเป็นไฟล์ในหน่วยความจำชื่อ RAW DATA
   - MAP TO DISPLAY นำข้อมูลไฟล์ RAW DATA มาจัดเรียงลำดับแอดเดรสใหม่เพื่อแสดงผลออกทางจอภาพในโมดกราฟฟิก
2. โปรแกรมจัดการล่วงหน้า (preprocess) ประกอบด้วยโมดการทำงานดังนี้
   - BOUDARY DECISION ทำการแยกตัวเลขออกจากกรอบ (กรณีใช้กับตัวเลขลายมือเขียน) โดยอาศัย ข้อมูลไฟล์ IMAGE DATA ซึ่งเป็นผลจากการเปลี่ยนรูปฟอร์มการเก็บข้อมูลจากไฟล์บิทภาพ RAW DATA ไปอยู่ในลักษณะตัวแปรร่วม (COMMON VARIABLE)
   - FILTERING จากข้อมูลไฟล์ IMAGE DATA ทำการกำจัดสิ่งแปลกปลอมผลที่ได้เก็บลงในไฟล์เดิม
   - EDGE DETECTION หาขอบตัวเลขเก็บในไฟล์ EDGE DATA
3. โปรแกรมจดจำลักษณะ (recognition) ประกอบด้วยโมดการทำงานดังนี้
   - POLYGONAL APPROXIMATION ทำการเกลาข้อมูลไฟล์ EDGE DATA ให้ได้เป็นรูปหลายเหลี่ยมเก็บในไฟล์ SEGMENT DATA
   - FEATURE GENERATION ทำการกำเนิดข้อมูลพีเชอะจากไฟล์ SEGMENT DATA เก็บเป็นข้อมูลไฟล์ใหม่ชื่อ FEATURE DATA

- PATTERN CLASSIFICATION ทำการตัดสินใจจากข้อมูล FEATURE DATA ว่าตัวเลขที่อ่านเป็นตัวเลขอะไร

รายละเอียดของโปรแกรมหลักได้เคยกล่าวไว้แล้วในหัวข้อที่ 3.4.4 ส่วนโปรแกรมที่เหลือจะได้กล่าวในหัวข้อต่อไป ตัวอย่างข้อมูลที่พิมพ์ออกในโมดกราฟิก ในแต่ละขบวนการประมวลผลแสดงในรูปที่ 4.2

READ DATA

BOUNDARY D.

FILTERING

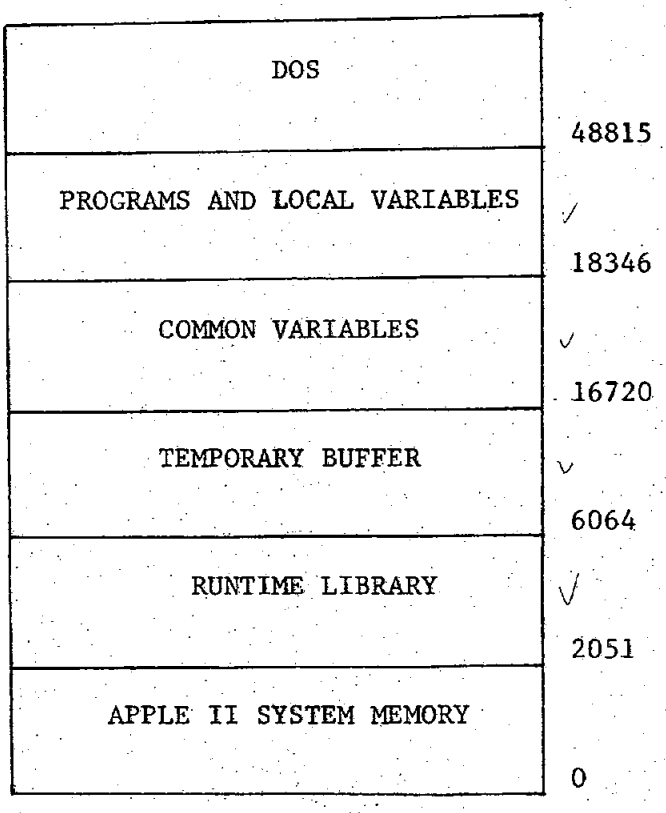
EDGE D.

POLYGONAL APPROX.

รูปที่ 4.2 แสดงตัวอย่างข้อมูลที่พิมพ์ในแต่ละขบวนการโปรเซส

4.3 MEMORY MAP ของระบบโปรแกรม

จากรูปที่ 4.1 จะเห็นได้ว่าในระบบจดจำตัวเลขอารบิคมีการประมวลผลหลายขั้นตอน และมีการเกี่ยวข้องกับไฟล์ข้อมูลมาก ดังนั้นการที่จะสามารถเขียนโปรแกรมการทำงานทั้งหมดลงในเครื่องไมโครคอมพิวเตอร์ APPLE II ซึ่งหน่วยความจำไม่เกิน 64 KB ได้จะต้องมีการแบ่งโปรแกรมเป็นส่วน ๆ แล้วเรียกมาทำงาน ทีละขั้นตอนเป็นแบบ CHAIN PROGRAM ดังนั้น จากข้อบังคับตามที่กล่าวมา และความต้องการให้มีเวลาในการประมวลผลน้อย เราจะทำการแปลภาษาเบสิกให้เป็นภาษาเครื่องโดยใช้ตัวแปลชื่อ TASC[36] ซึ่งจะทำให้สามารถมีการทำงานแบบ over lay ทั้งยังมี speed ในการ run สูงขึ้นด้วย แผนผังการจองที่ของระบบโปรแกรมซึ่งเรียกว่า MEMORY MAP แสดงในรูปที่ 4.3 ดังนี้



รูปที่ 4.3 แสดง MEMORY MAP ของระบบโปรแกรมจดจำตัวเลขอารบิคที่ออกแบบ

1) APPLE II SYSTEM MEMORY (0-2050) เป็นส่วนที่ไมโครคอมพิวเตอร์ APPLE II ใช้ในการเก็บค่าคงที่ของระบบ[37]

2) RUNTIME LIBRARY (2051-6063) เป็นส่วนที่รวบรวม machine language routine ต่าง ๆ ที่ โปรแกรมจะต้องใช้เป็นส่วนหนึ่งของ TASC Compiler

3) TEMPORARY BUFFER (6064-16719) ใช้เป็นที่เก็บไฟล์ชั่วคราว เช่น RAW DATA ทั้งยังเป็นที่เก็บโปรแกรมย่อยต่าง ๆ ที่เขียนด้วยภาษาแอสเซมบลี 6502 และยังใช้เป็น graphic plot area

4) COMMON VARIABLES (16720-18345) เป็นส่วนที่ใช้ในการเก็บข้อมูลและตัวแปรต่าง ๆ ดังนี้

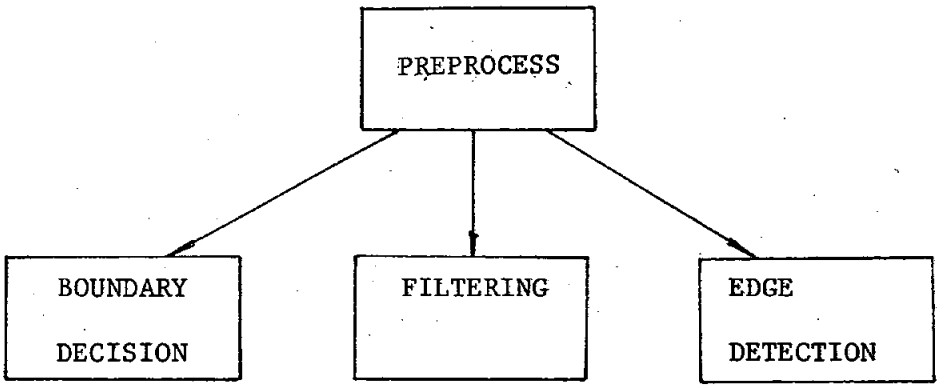
   - IMAGE DATA FILE
   - EDGE DATA FILE
   - SEGMENT DATA FILE
   - FEATURE DATA FILE

5) PROGRAMS AND LOCAL VARIABLES (18346-48815) เป็นส่วนที่เก็บ machine code ของระบบโปรแกรม เนื่องจากเนื้อที่ส่วนนี้ไม่มากพอที่จะเก็บโปรแกรมได้ทั้งหมด จึงต้องแบ่งออกเป็น 3 ส่วน ดังที่กล่าวในหัวข้อ 4.2 การทำงานจะทำเพียงที่ละส่วนในเนื้อที่ส่วนนี้

6) DOS (48816-65536) เป็นส่วนที่เก็บโปรแกรมควบคุมการทำงานของ APPLE II

4.4 โปรแกรมจัดการล่วงหน้า

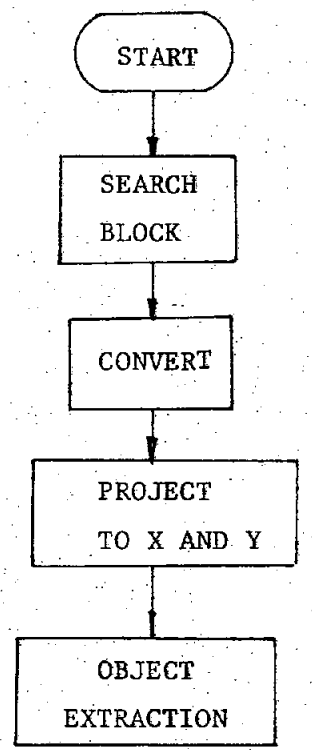
ในส่วนนี้มีการทำงาน 3 โมดดังแสดงในรูปที่ 4.4



รูปที่ 4.4 แสดงส่วนประกอบของโปรแกรมจัดการล่วงหน้า

4.4.1 โปรแกรมการหาขอบเขต

จะทำการค้นหากรอบของตัวเลขเสร็จแล้วจะทำการแยกเอาแต่ตัวเลขเพียงอย่างเดียว โดยการกำจัดกรอบทิ้ง ซึ่งบีโฟลว์ชาร์ท แสดงการทำงานในรูปที่ 4.5 ซึ่งจะอธิบายการทำงานแต่ละขั้นได้คือ



รูปที่ 4.5 แสดงโฟลว์ชาร์ทโปรแกรมการหาขอบเขต

ก. SEARCH BLOCK จะทำการสแกนตรวจหากรอบใส่ตัวอักษร โดยการสแกนข้อมูลภาพจากบนลงล่างทีละแถว ถ้าพบว่ามีข้อมูลที่ตำแหน่งใดเป็น 1 แสดงว่าพบขอบบนของกรอบแล้วการหาขอบด้านล่าง ก็ทำในลักษณะเดียวกันเปลี่ยนแต่การสแกนเริ่มจากตำแหน่งล่างสุดของข้อมูลภาพขึ้นไปทีละแถว ส่วนการหาขอบด้านซ้ายและด้านขวา จะทำการสแกนข้อมูลจากซ้ายสุดไปทางขวาและขวาสุดไปซ้าย โดยการสแกนกระทำทีละคอลัมน์ข้อมูลภาพ ในการทำงานขั้นตอนนี้ โปรแกรมจะถูกเขียนด้วยภาษาแอสเซมบลี โดยมีภาษาเบสิก เป็นตัวเรียกมาใช้เป็นโปรแกรมย่อยอีกทีหนึ่ง ในขั้นตอนนี้เมื่อการทำงานเสร็จเราจะรู้ตำแหน่งแถวของขอบด้านบนและด้านล่าง รู้ตำแหน่งคอลัมน์ของขอบกรอบด้านซ้ายและด้านขวา

ข. CONVERT จะทำการเปลี่ยนข้อมูลบิทภาพของกรอบที่บรรจุตัวเลขไปเก็บในลักษณะตัวแปร 2 มิติ (two dimensions) ทั้งนี้ก็เพื่อการประมวลผลต่อ ๆไปมีการอ้างอิงข้อมูลได้ง่ายขึ้น ในขั้นตอนนี้โปรแกรมจะถูกเขียนด้วยภาษาเบสิกร่วมกับภาษาแอสเซมบลี

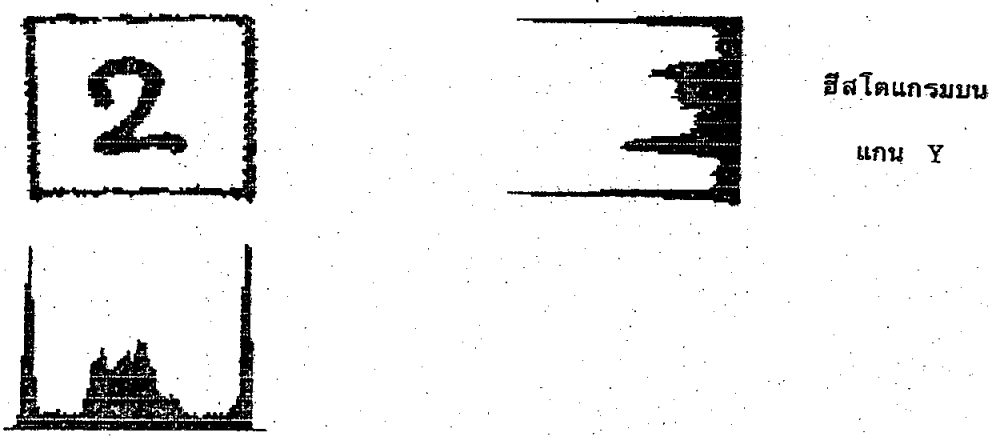
ค. PROJECT TO X AND Y ในขั้นตอนนี้ตัวแปรเก็บข้อมูลภาพ 2 มิติ จะถูกนับจำนวนจุดที่มีค่าเป็น 1 ในแต่ละคอลัมน์ซึ่งจะถูกสร้างเป็นรูปฮีสโตแกรมแกน X และจะถูกนับค่า 1 ในแต่ละแถวสร้างเป็นรูปฮีสโตแกรมแกน Y ดังแสดงในรูปที่ 4.6

ง. OBJECT EXTRACTION ในขั้นตอนนี้เป็นการแยกตัวเลขออกจากกรอบ โดยการหาจุดบนฮีสโตแกรม แกน X และ Y ซึ่งมีความชันเป็นศูนย์ที่ไม่อยู่ตรงตำแหน่งกรอบ โดยใช้วิธีการตั้งจุดอ้างอิงซึ่งจะเรียก X-REF สำหรับฮีสโตแกรมแกน X และ Y-REF สำหรับฮีสโตแกรมแกน Y ถ้าจุดบนฮีสโตแกรมซึ่งมีค่าต่ำกว่า X-REF(Y-REF) และมีความชันเป็นศูนย์ เราจะเลือกจุดนั้นเป็นจุดแบ่งแยกกรอบออกจากตัวเลข ดังแสดงจุดแบ่งสำหรับฮีสโตแกรมแกน X ในรูปที่ 4.7

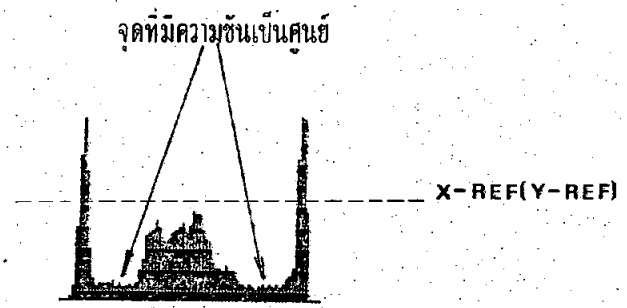
การเลือกค่า X-REF และ Y-REF จะได้จากการทดสอบตัวเลขที่เขียนในกรอบซึ่งเราสุ่มขึ้นมา พบว่า บริเวณถัดจากกรอบเข้ามาตรงตำแหน่งรอบจุดซึ่งมีความชันเป็นศูนย์มีค่าต่ำกว่า 12 ทั้งแกน X และแกน Y ส่วนบริเวณกรอบตัวเลขมีค่ามากกว่า 30 ขึ้นไป ดังนั้นค่า X-REF และ Y-REF จึงสามารถเลือกในช่วง 12 ถึง 30 สำหรับโปรแกรมที่เขียนขึ้นจะเลือกค่า X-REF และ Y-REF เท่ากับ 25

จุดอ่อนของวิธีการในขั้นตอนนี้ก็คือ ถ้าตัวเลขถูกเขียนชิดติดกรอบหรือเขียนทับกรอบการแยกกรอบจะเกิดข้อผิดพลาดได้ ทั้งนี้เนื่องจากจุดความชันเป็นศูนย์อาจเกิดในตำแหน่งเนื้อตัวอักษร ดังแสดงให้เห็นในรูปที่ 4.8 ดังนั้นทางแก้ไขควรเขียนตัวเลขอยู่ภายในกรอบ

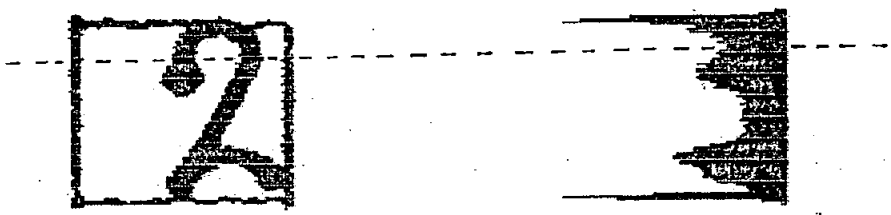
โปรแกรมการหาขอบเขตถ้าใช้กับตัวพิมพ์ดังที่กล่าวในหัวข้อ 4.4.1 ขั้นตอน ค. และ ง. ไม่ต้องทำ เนื่องจากตัวพิมพ์ไม่มีกรอบล้อมรอบ



รูปที่ 4.6 แสดงการโปรเจคสร้างฮีสโตแกรมบนแกน X และ Y



รูปที่ 4.7 แสดงจุดที่มีความชันเป็นศูนย์ที่เลือก



รูปที่ 4.8 แสดงความผิดพลาดที่เกิดเนื่องจากเขียนตัวเลขชิดกรอบ

4.4.2 โปรแกรมกำจัดสัญญาณรบกวน (filtering)

ทำการสร้างหน้าต่าง ขนาด 3 x 3 วิ่งไปทุกตำแหน่งบนข้อมูลภาพตามวิธีที่ได้กล่าวมาแล้วในหัวข้อที่ 2.4.2 สาเหตุที่เลือกหน้าต่าง ขนาด 3 x 3 ก็เพราะจากการตรวจสอบข้อมูลตัวเลขลายมือที่เขียนบนกระดาษพบว่า บนกระดาษมีรอยเปื้อนเป็นจุดซึ่งเมื่อผ่านการอ่านเข้ามาเป็นบิทภาพมีขนาดไม่เกิน 2 x 2 จุด ดังนั้นการเลือกหน้าต่างจึงต้องให้มีขนาดใหญ่กว่าขนาดจุดเปื้อนที่ต้องการกำจัด แต่ถ้าเลือกขนาดโตเกินไปการทำงานของโปรแกรมจะช้าลงเนื่องจากมีข้อมูลคำนวณเพิ่มขึ้น ดังนั้นจึงจะเลือกขนาดหน้าต่าง 3 x 3 ดังแสดงตำแหน่งอ้างอิงในรูปที่ 4.9

ถ้าให้ S เป็นข้อมูลภาพก่อนการกำจัดสัญญาณรบกวน

จะได้ว่า ผลรวมภายในหน้าต่าง (Sum)

$$Sum = A + B + C + D + E + F + G + H + I$$

เราจะสามารถสร้างข้อมูลภาพชุดใหม่ S' ซึ่งมีตำแหน่งสมนัยกับ S โดยมีเงื่อนไขว่า

ค่าไลจิกที่ตำแหน่ง E' บน S' = 1 ถ้า Sum $>$ threshold (θ)

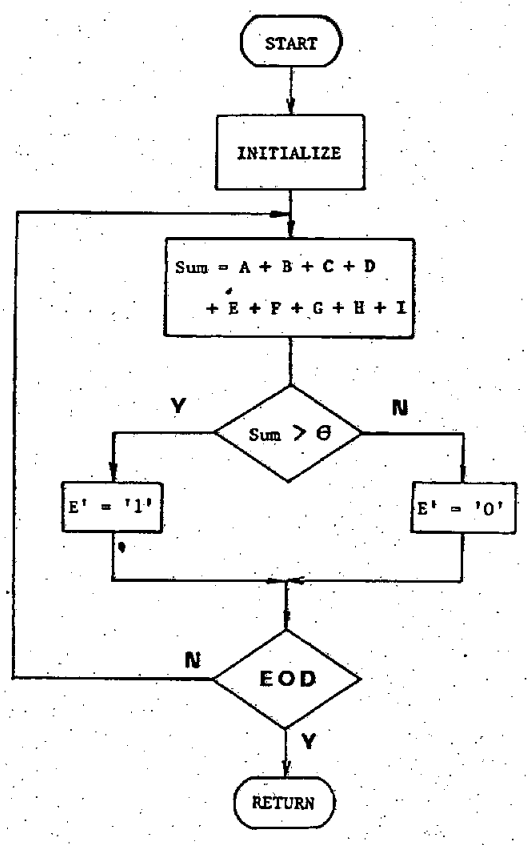
= 0 กรณีอื่น ๆ

การเลือกค่าเทรชโฮลขึ้นอยู่กับขนาดของสัญญาณรบกวนที่ต้องการกำจัด และขนาดความกว้างของเส้นตัวเลข เนื่องจากการกำจัดสัญญาณรบกวนใช้หน้าต่างขนาด 3 x 3 ดังนั้นค่าเทรชโฮลสามารถเลือกตั้งแต่ 1 - 9 แต่จากการทดสอบพบว่าควรเลือกค่าใน

ช่วง 4 - 6 ซึ่งจะสามารถลดจุดเบื้อนต่ำได้ดี และข้อมูลภาพไม่สูญเสียมากนัก โฟลว์ชาร์ทการทำงานแสดงในรูปที่ 4.10 และตัวอย่างผลที่ได้จากขบวนการนี้แสดงในรูปที่ 4.11 โดยเทรดโฮลเท่ากับ 5

A	B	C
D	E	F
G	H	I

รูปที่ 4.9 แสดงตำแหน่งอ้างอิงหน้าต่าง 3 X 3



รูปที่ 4.10 แสดงโฟลว์ชาร์ทโปรแกรมกำจัดสัญญาณรบกวน

2

รูปที่ 4.11 แสดงตัวอย่างผลการกำจัดสัญญาณรบกวนจากตัวเลขในรูปที่ 4.6

4.4.3 โปรแกรมหาขอบตัวอักษร

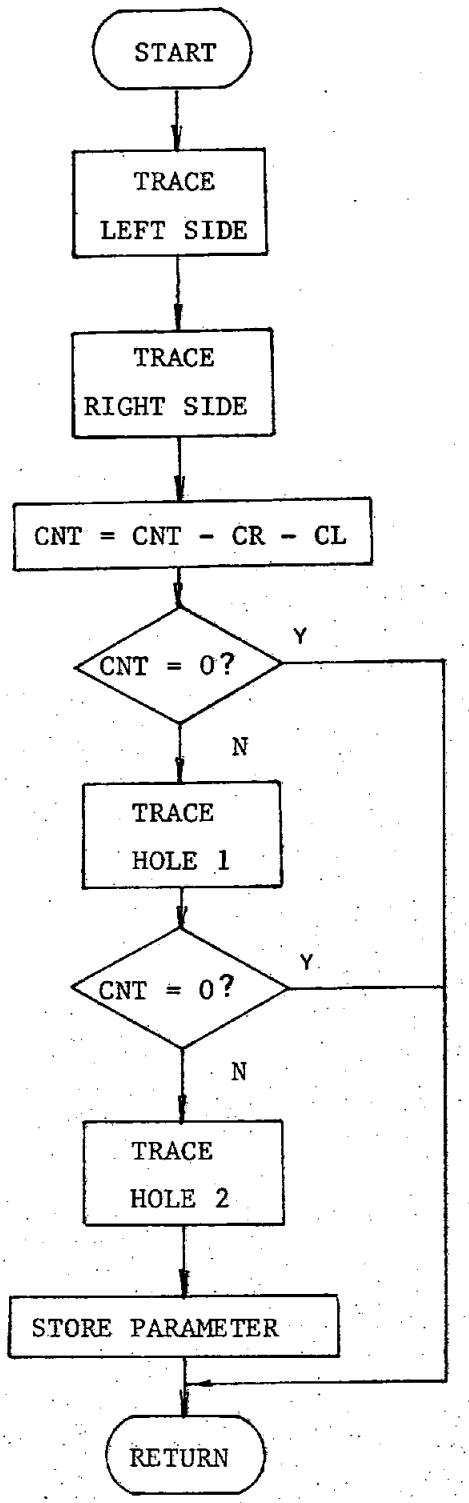
อาศัยหลักการที่ได้กล่าวแล้วในหัวข้อ 2.4.2 ซึ่งแบ่งการทำงานในโปรแกรมได้ 2 ขั้นตอน ดังนี้คือ

ก. REPRESENT EDGE จะดัดแปลงวิธีค่าเกรเดียนต์โดยตำแหน่ง BEP จะอยู่ภายในเนื้อตัวเลขที่มีการเปลี่ยนค่า '0' เป็น '1' หรือ '1' เป็น '0' จุด BEP จะถูกเปลี่ยนจาก '1' เป็นโลจิก '4' ซึ่งการเปลี่ยนก็เพื่อข้อมูล '4' แสดงจุดขอบ จำนวนจุดขอบจะเก็บในตัวแปรชื่อ CNT

ข. CONTOUR TRACING โดยการกวาดรอบตัวเลขเพื่อเก็บข้อมูลตำแหน่งขอบเป็นลำดับ เพื่อใช้เป็นข้อมูลในการประมวลขั้นต่อ ๆ ไป ไฟลว์ชาร์ทการทำงานแสดงอยู่ในรูปที่ 4.12 เริ่มต้นแบ่งจุดขอบเป็น 2 ส่วน คือ จุดขอบด้านซ้ายและด้านขวาตัวเลข โดยมีจุดสูงสุดกับจุดต่ำสุดของขอบตัวเลขเป็นตัวแบ่งด้านซ้ายและขวา จากนั้นจะทำการกวาดคอนทัวร์จุดขอบด้านซ้ายและด้านขวา จำนวนจุดขอบด้านซ้ายและด้านขวาจะถูกเก็บในตัวแปรชื่อ CR และ CL หลังจากนั้นจะทำการหักค่า CR และ CL ออกจากตัวแปร CNT ถ้าผลที่ได้ค่า CNT มากกว่าศูนย์ แสดงว่าตัวเลขมีรู จะทำการกวาดจุดขอบของรู หาค่า h1 และ h2 ซึ่งใช้เป็นพารามิเตอร์ในขบวนการประมวลผลต่อไปจำนวนจุดรอบรูที่หนึ่งจะถูกหักออกจาก CNT ถ้าผลลัพธ์ ยังไม่เป็นศูนย์แสดงว่าตัวเลขมีรูที่สอง จะทำการกวาดจุดขอบรูแบบเดียวกัน

เพื่อหาค่าพารามิเตอร์ของรูที่สอง ค่าพารามิเตอร์ของรู และตำแหน่งพิกัด (coordinate) ของขอบด้านซ้ายกับด้านขวาจะถูกเก็บในหน่วยความจำ ซึ่งได้จองไว้แล้ว การประมวลผลต่อไปจะใช้ค่าเหล่านี้เป็นข้อมูล แทนข้อมูลบิทภาพที่เก็บในตัวแปร 2 มิติ

จุดอ่อนของโปรแกรมนี้ คือ ในกรณีจุดขอบไม่เป็นเส้นปิด[12] (closed curve) การทำงานของโปรแกรมจะไม่ถูกต้อง กรณีดังกล่าวแสดงตัวอย่างในรูปที่ 4.13

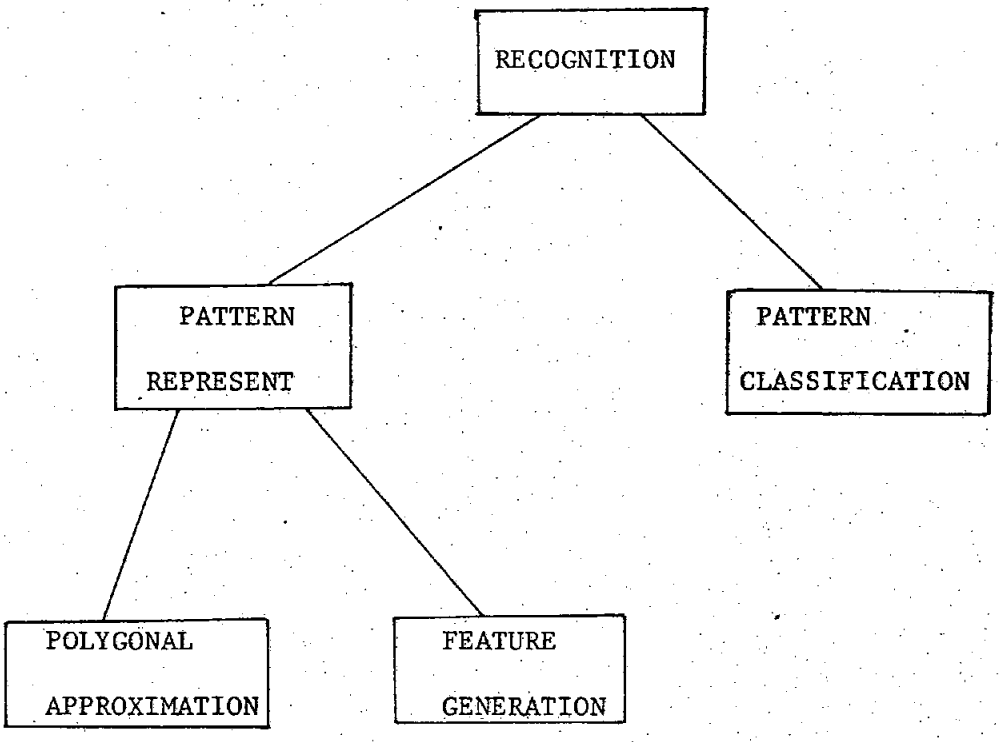


รูปที่ 4.12 แสดงโฟลว์ชาร์ทในขบวนการ กวาดคอนทัวร์

รูปที่ 4.13 แสดงเงื่อนไขกรณีจุดขอบไม่เป็นเส้นปิด

4.5 โปรแกรมจดจำลักษณะ (recognition)

อาศัยหลักการที่ได้กล่าวมาแล้วในหัวข้อ 2.4.3 มีโครงสร้างการทำงานดังแสดงในรูปที่ 4.14 โปรแกรมในส่วนนี้จะเขียนด้วยภาษาเบสิกทั้งหมด



รูปที่ 4.14 แสดงโครงสร้างการทำงานโปรแกรมจดจำลักษณะ

4.5.1 โปรแกรมนำเสนอรูปแบบ (pattern representation)

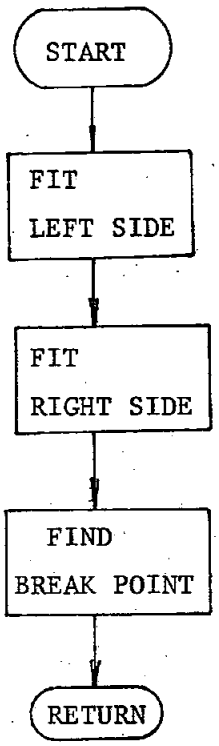
เป้าหมายคือ การนำข้อมูลจุดขอบตัวเลขมาสร้างตัวแปรลักษณะสำคัญโดยแบ่งการทำงานได้เป็น 2 ขั้นตอน ดังแสดงในรูปที่ 4.14

ก. POLYGONAL APPROXIMATION จะใช้วิธี split and merge ดังที่กล่าวมาแล้วในหัวข้อ 2.4.3 ประมาณจุดขอบตัวเลขให้เป็นรูปหลายเหลี่ยม โฟลว์ชาร์ทการทำงานแสดงในรูปที่ 4.15 โดยการเกลาจุดขอบที่ละด้านโดยเริ่มทางด้านซ้ายก่อนและตามด้วยทางขวา ทั้งนี้ก็เพื่อแยกส่วนผลการเกลาออกเป็นด้านซ้ายและด้านขวา หลังจากนั้น จะทำการหาจุด break point ระหว่างเซกเมนต์ เพื่อต่อแต่ละเซกเมนต์ให้ได้เป็นรูปหลายเหลี่ยม ผลจากการหาจุด break point จะทำการคำนวณหามุมที่จุดหักมุมต่าง ๆ ในขั้นตอนมีตัวแปรที่จะพิจารณา อยู่คือ

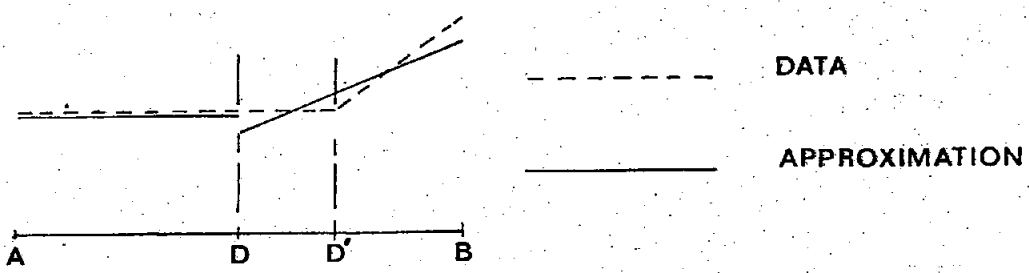
ML & MR - จำนวนเซกเมนต์เริ่มต้นทางซ้ายและขวา

Emax - ค่าความผิดพลาดสูงสุดที่จะยอมรับได้ในการเกลาเส้นโค้ง

- ค่า ML(MR) มีผลต่อค่าผิดพลาดรวม เนื่องจากประสิทธิภาพของขบวนการ split ยังมีข้อบกพร่องในเงื่อนไขการแบ่งเซกเมนต์ ตัวอย่างเช่น ผลการแบ่งครึ่งอาจไม่ได้จุดหักมุมที่ดีที่สุด ตัวอย่างแสดงใน รูปที่ 4.16 จุด D ซึ่งเป็นจุดแบ่งครึ่ง เซกเมนต์ AB ไม่ได้เป็นจุดหักมุมในการประมาณที่ดีที่สุด จุดที่ดีที่สุดน่าจะเป็นจุด D' ดังนั้นถ้าเลือกค่า ML(MR) น้อยเกินไป ทำให้ในโปรแกรม polygonal approximation จะเกิดขบวนการ split หลายครั้ง ทำให้ผลการประมาณเป็นรูปหลายเหลี่ยมไม่ดีพอ แต่ถ้าเลือกค่า ML(MR) มากเกินความจำเป็น ขบวนการประมวลผลจะเสียเวลามาก เนื่องจากจะเกิดขบวนการ merge หลายครั้งในการต่อเซกเมนต์ที่เล็ก ๆ ให้เป็นเซกเมนต์ใหญ่ ดังนั้นการเลือก ML(MR) ต้องให้อยู่ในช่วงกำลังเหมาะซึ่งค่าที่ได้จะหามาจากการทดลองโดยเลือก ML = 8 , MR = 8



รูปที่ 4.15 แสดงโฟลว์ชาร์ทการทำงานในโปรแกรม polygonal approximation



รูปที่ 4.16 แสดงจุดอ่อนของขบวนการ split

– ค่า Emax มีผลต่อการลดความแปรปรวนบริเวณขอบตัวเลข โดยถ้าเลือกค่า Emax สูงเกินไป ผลการประมาณ จะเกิดการสูญเสียรูปลักษณะเดิมได้ แต่ถ้ากำหนด Emax ต่ำเกินไป ผลความแปรปรวนจะทำให้ฟังก์ชันในการโปรเซสตัดสินใจซับซ้อนขึ้น จากการทดลองจะเลือกใช้ค่า Emax = 20 ตัวอย่างผลการเปลี่ยนค่า Emax แสดงในรูปที่ 4.17

ก ข ค

รูปที่ 4.17 แสดงผลของกระบวนการ polygonal approximation โดย
ก) Emax = 5 ข) Emax = 20 ค) Emax = 40

ข. FEATURE Generation เริ่มต้นจะทำการกวาดขอบตัวเลขในทิศทางทวนเข็มนาฬิกา เริ่มจากจุดสูงสุดของตัวเลขวนไปทางด้านซ้าย จนพบจุดต่ำสุดกวาดขอบด้านขวาขึ้นไปพบจุดสูงสุดในขณะที่กวาดนั้นจุดหักมุมต่าง ๆ จะถูกตรวจสอบเพื่อหามุมเว้า (concave angle) หรือส่วนโค้งเว้า (concave arc) โดยการตรวจสอบว่า ถ้ามุมที่จุดหักมุมใดมีขนาดน้อยกว่า 180 องศา ถือว่าเป็น มุมเว้า ส่วนมุมที่มากกว่าหรือเท่ากับ 180 องศา ถือเป็นมุมนูน (convex angle) ถ้ามีมุมเว้าตั้งแต่ 2 มุมขึ้นไปอยู่ชิดกันถือว่าเป็นส่วนเว้า ในการพิจารณาตำแหน่งมุมเว้าหรือส่วนเว้าจะถูกอธิบายเป็นโซน ซึ่งการแบ่งโซนแสดงในรูปที่ 4.18 โดย BCDEFGHJK เป็นขอบด้านซ้ายและ MNPSQRTUV เป็นขอบด้านขวา ลักษณะสำคัญที่สร้างขึ้นเพื่อใช้ในการจำแนกตัวเลข ตัวเลขอารบิคแบ่งเป็น 2 ชุด

1. ลักษณะสำคัญบอกจำนวนหรือตำแหน่ง

U1 จำนวนรู

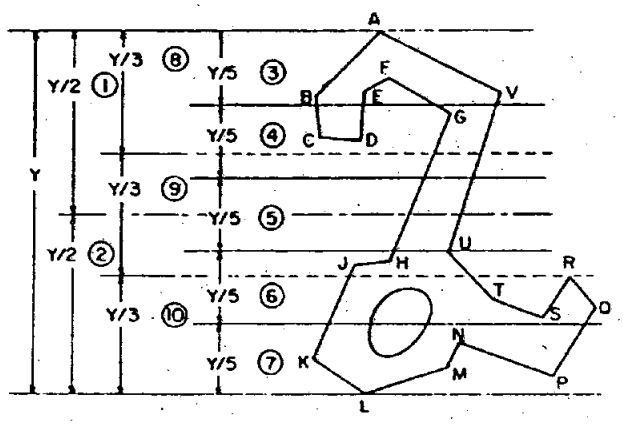
U2 ตัวแปรบรรยายตำแหน่งของรู (กรณีมีรูเดียว)

U2 = 1 ถ้า (h1 + h2)/2 อยู่ในโซน 6 หรือ 7 และ hI อยู่ในโซน 5,6 หรือ 7 (รูล่าง)

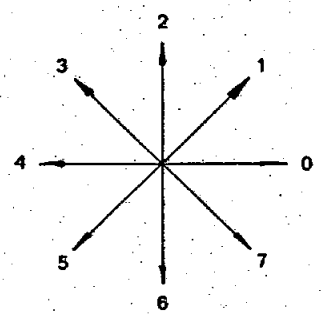
U2 = 2 ถ้า (h1 + h2)/2 อยู่ในโซน 1 และ h2 อยู่ในโซน อยู่ในโซน 3,4,5 (รูบน)

U2 = 0 กรณีอื่น

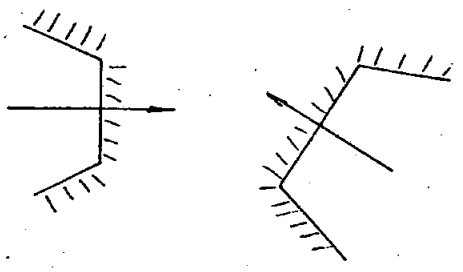
U3 จำนวนเซกเมนต์ที่เป็นเส้นตั้งฉาก (80 < มุมเซกเมนต์ < 100) และมีความยาวมากกว่า y/3



รูปที่ 4.18 แสดงการแบ่งพื้นที่ตัวเลขเป็นโซน



รูปที่ 4.19 แสดงการอธิบายทิศทางของเซกเมนต์หรือเส้นแบ่งครึ่งมุม



รูปที่ 4.20 แสดงนิยาม เส้นแบ่งครึ่งส่วน เว้า

U4 จำนวนมุม เว้าและส่วน เว้าทางด้านซ้าย

U5 จำนวนมุม เว้าและส่วน เว้าทางด้านขวา

U6,U7,U8 ตำแหน่งของส่วน เว้าหรือมุม เว้า อันดับแรก อันดับสอง อันดับสาม ตามลำดับ

U9 จำนวนมุมในส่วน เว้าอันดับแรกทางด้านซ้าย

U10 จำนวนมุมในส่วน เว้าอันดับสุดท้ายทางด้านขวา

U11,U12 ตำแหน่ง h1,h2 ของรูบน

U13,U14 ตำแหน่ง h1,h2 ของรูล่าง (ใช้กรณี U1 = 2)

2. ลักษณะสำคัญบอกทิศทาง ใช้อธิบายทิศทางของ เซก เมนต์ หรือทิศทาง เส้นแบ่งครึ่งส่วน เว้า ซึ่งทิศทางนั้นแสดงในรูปที่ 4.19 ซึ่งจะอธิบายในลักษณะคล้าย chain code นิยาม ของ เส้นแบ่งครึ่งส่วน เว้าแสดงในรูปที่ 4.20 ลักษณะสำคัญที่สร้างขึ้นมีดังนี้

U15 ทิศทาง เส้นแบ่งครึ่งส่วน เว้าแรกทางด้านซ้าย

U16 ทิศทาง เส้นแบ่งครึ่งส่วน เว้าสุดท้ายด้านขวา

U17 จำนวนส่วน เว้าบนขอบตัว เลขทั้ง 2 ด้าน

U18 ทิศทาง เซก เมนต์ที่พุ่งออกจากส่วน เว้าหรือมุม เว้าอันดับแรก (กรณีเป็นส่วน เว้าจะ เป็นมุมสุดท้าย) ด้านซ้าย

U19 ทิศทาง เซก เมนต์ที่พุ่ง เข้าส่วน เว้าหรือมุม เว้าอันดับสุดท้ายด้านขวา

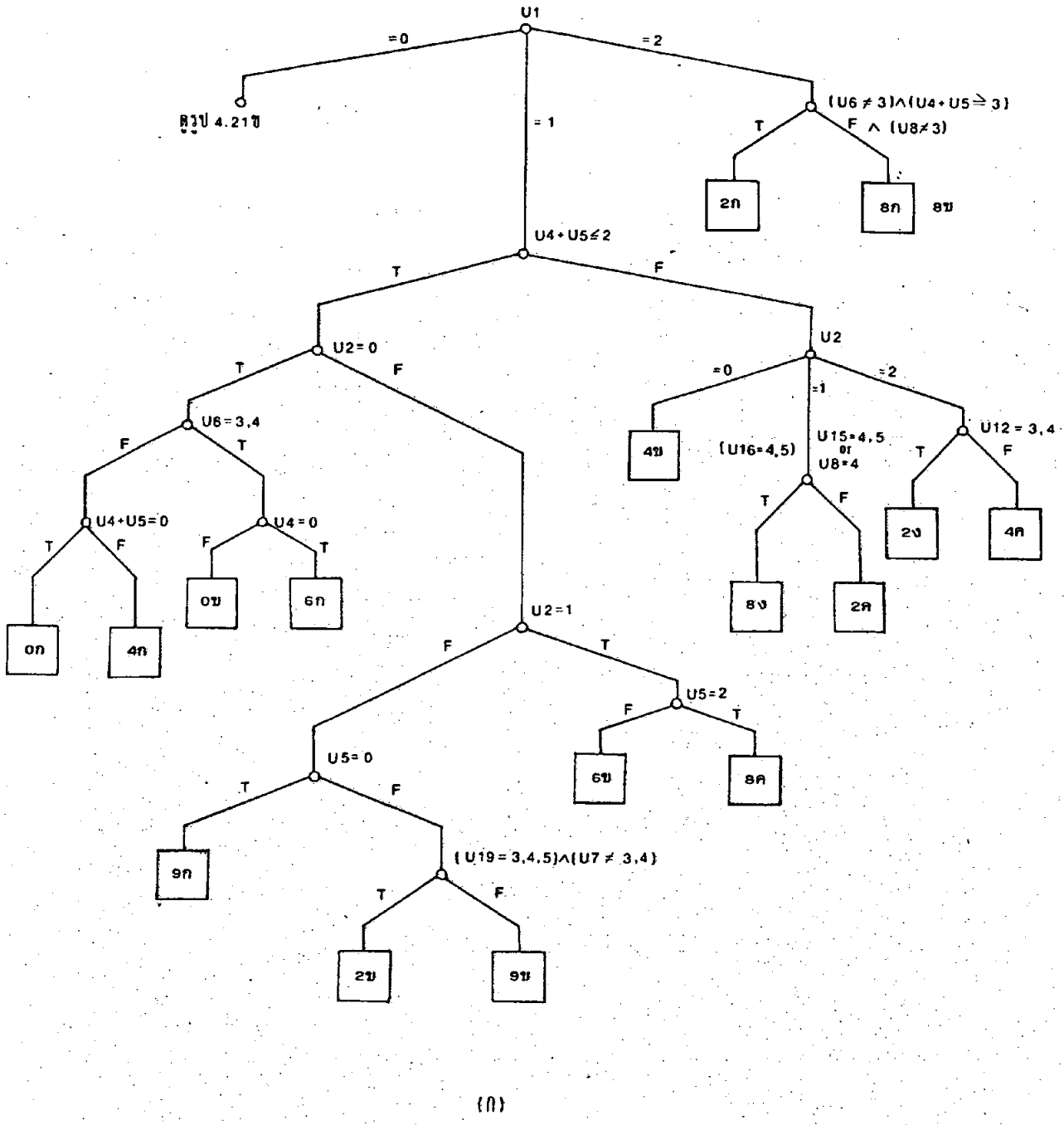
U20 ทิศทาง เซก เมนต์พุ่ง เข้ามุม เว้าบนสุดด้านซ้าย

ตัวอย่างค่าลักษณะสำคัญที่ได้จากรูปที่ 4.18 มีค่าดังต่อไปนี้

U1 = 1 , U2 = 1 , U3 = 0 , U4 = 1 , U5 = 2 , U6 = 4 , U7 = 7 , U8 = 6 , U9 = 4 , U10 = 3 , U11 = 6 , U12 = 7 , U15 = 1 , U16 = 5 , U17 = 2 , U18 ได้แก่เวกเตอร์ HJ , U19 ได้แก่เวกเตอร์ RS , U20 ได้แก่เวกเตอร์ DE

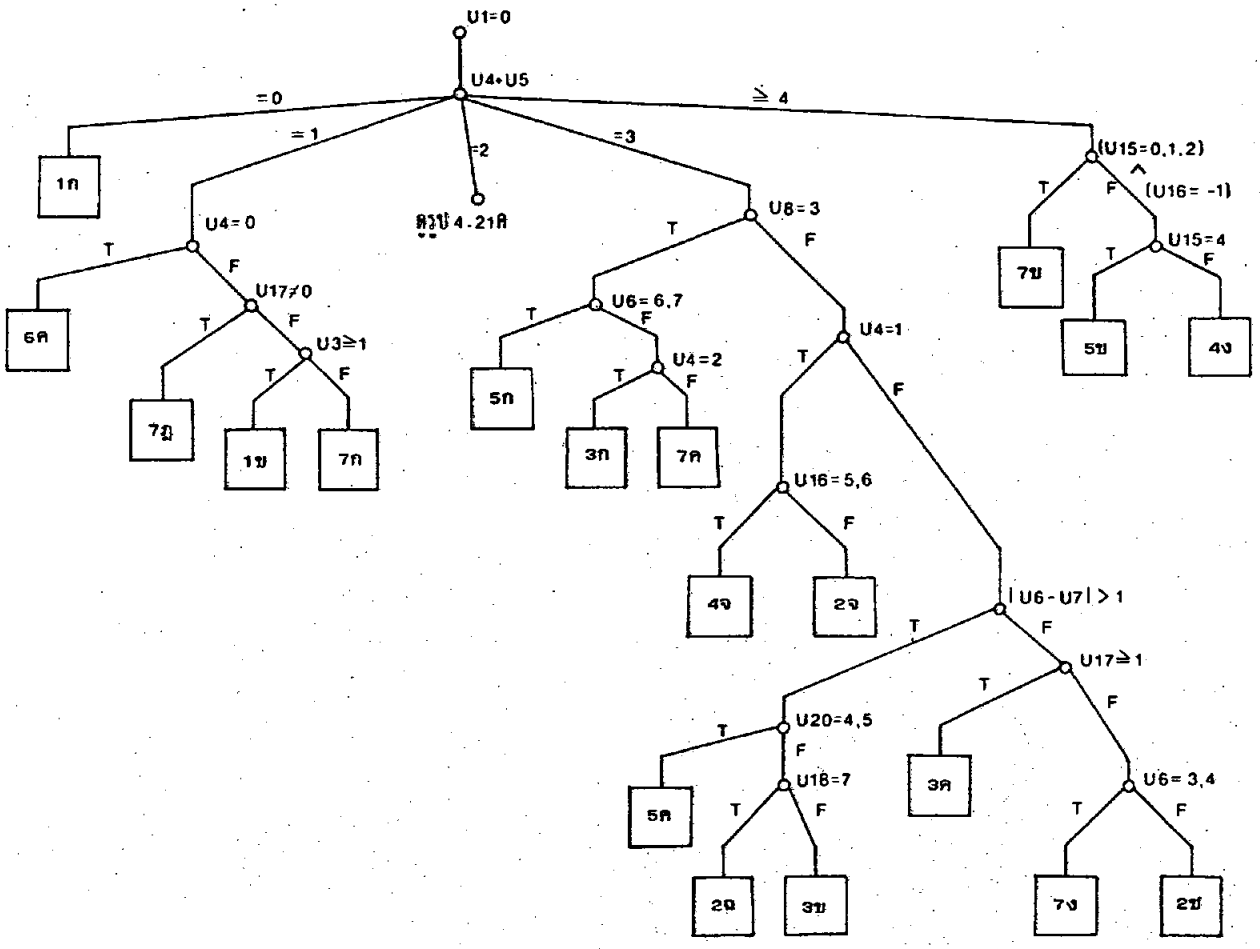
4.5.2 โปรแกรมจำแนกรูปแบบ (pattern classification)

จะทำการสร้าง classification tree เพื่อใช้จำแนกตัวเลข โดยการสร้าง tree นี้จะอาศัยข้อมูลที่รู้แล้วเกี่ยวกับตัวเลขและข้อมูลที่ได้ จากการสุ่มตัวอย่างมาประกอบกัน ดังนั้น tree ที่ได้อาจจะยังไม่ดีนัก การสร้าง tree จะดีขึ้น ถ้าเราสามารถสุ่มตัวอย่างได้มากขึ้น ในการจำแนกตัวเลขจะพิจารณาตัวเลขแต่ละตัวออกเป็น กลุ่มแต่ละปลายของโนด จะเป็นรูปแบบตัวเลขในแต่ละ class จากการทดสอบกับตัวเลขอารบิคตัวพิมพ์ และจากตัวเลขอารบิคลายมือเขียน (รายละเอียดขั้นตอนการทดสอบจะกล่าวในบทต่อไป) จะออกแบบ classification tree ได้ดังแสดงในรูปที่ 4.21 และสมาชิกใน class ของตัวเลขแสดงไว้แล้วในรูปที่ 4.22

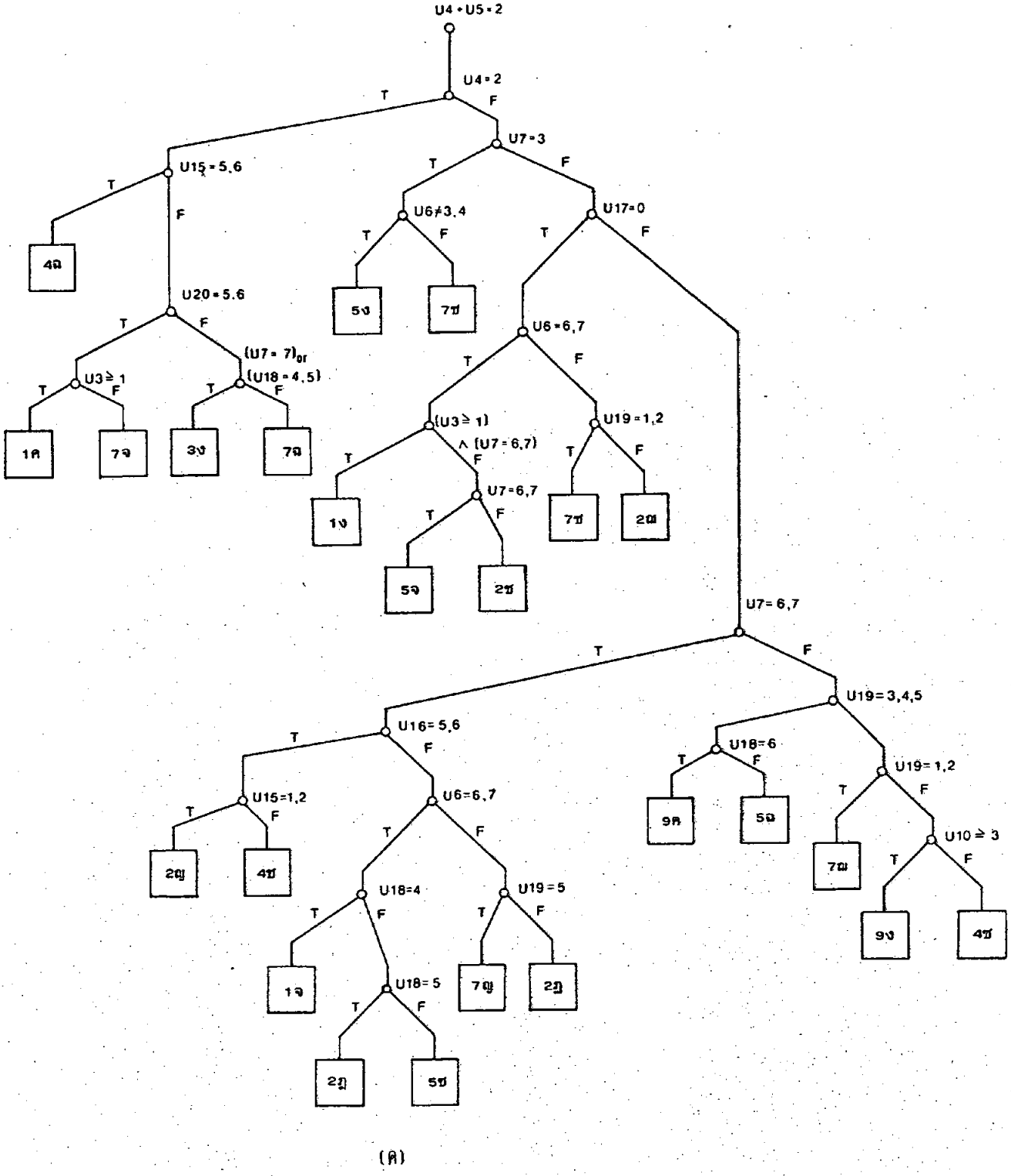


(ก)

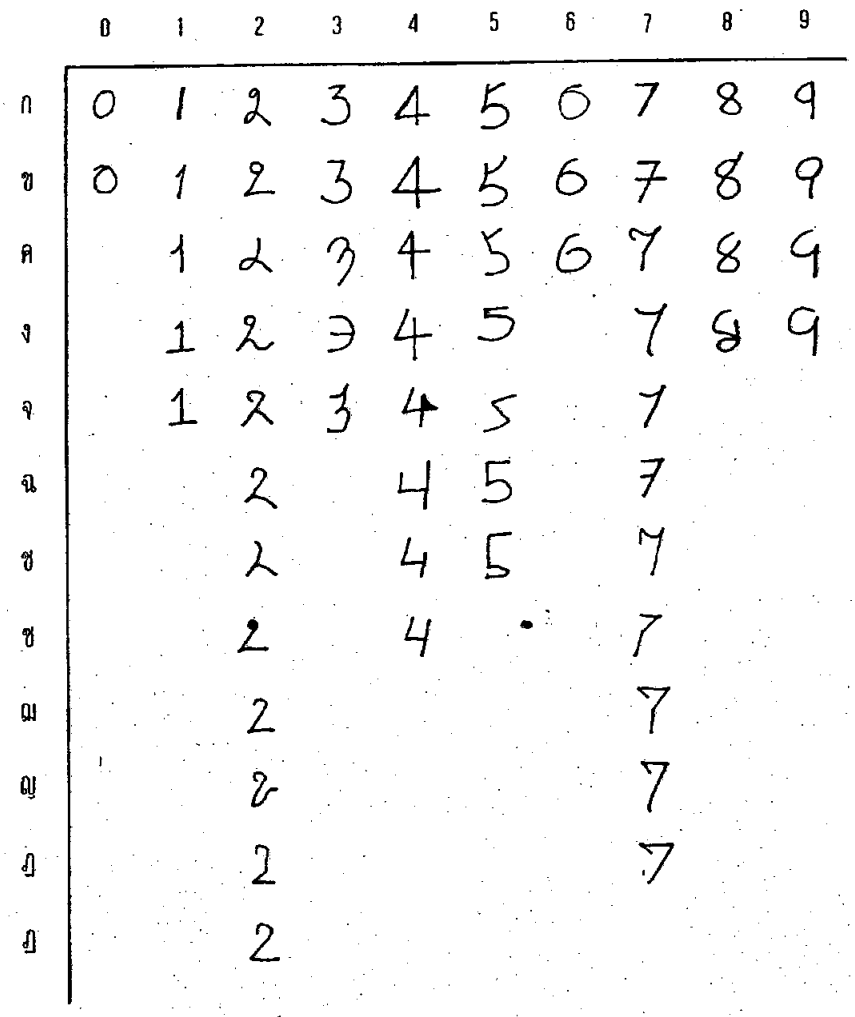
รูปที่ 4.21 แสดง classification tree ที่สร้างขึ้น


U1=0
U4+U5
=0
=1
=2
=3
≧ 4
1ก
U4=0
ดูรูป 4.21ค
U8=3
(U15=0,1,2)
^
(U16= -1)
T
F
6ค
U17≠0
U3≧1
7ฎ
1ข
7ก
U6=6,7
U4=1
5ก
U4=2
3ก
7ค
U16=5,6
4จ
2จ
|U6-U7|>1
U17≧1
U20=4,5
U18=7
5ค
2ฎ
3ข
3ค
U6=3,4
7ง
2ข
7ข
5ข
4ง

(ข)


U4 + U5 = 2
U4 = 2
U15 = 5,6
U7 = 3
U6 ≠ 3,4
U17 = 0
U20 = 5,6
U6 = 6,7
(U7 = 7)or (U18 = 4,5)
U3 ≥ 1
(U3 ≥ 1) ∧ (U7 = 6,7)
U19 = 1,2
U7 = 6,7
U7 = 6,7
U19 = 3,4,5
U16 = 5,6
U18 = 6
U19 = 1,2
U15 = 1,2
U6 = 6,7
U10 ≥ 3
U18 = 4
U19 = 5
U18 = 5

(ก)



รูปที่ 4.22 แสดงสมาชิกใน class ของตัวเลข 0 ถึง 9