การจดจำรูปแบบตัวพิมพ์โน้ต

RECOGNITION OF PRINTED MUSIC NOTE

โดย

1. นาย นิรัติ บุญชูมณี
2. นาย ภิญโญ เหล่ามงคลชัย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตร์บัณฑิต
สาขาวิศวกรรมการวัดคุมทางอุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2537

ปีการศึกษา 2537

การจดจำรูปแบบตัวพิมพ์โน้ตดนตรี

Recognition of Printed Music Note

นาย นิรัติ บุญชูมณี

นาย ภิญโญ เหล่ามงคลชัย

อาจารย์ที่ปรึกษา

อาจารย์ เกษตร์ ศิริสันติสัมฤทธิ์

ปริญญานิพนธ์ปีการศึกษา 2537

ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การจดจำรูปแบบตัวพิมพ์โน้ตดนตรี (Recognition of Printed Music Note)

ผู้จัดทำ

1. นาย นิรัติ บุญชุมณี

2. นาย ภิญโญ เหล่ามงคลชัย

..... อาจารย์ที่ปรึกษา

(เกษตร์ ศิริสันติสัมฤทธิ์)

การจดจำรูปแบบตัวพิมพ์โน้ตดนตรี

นาย นิรัติ บุญชูมณี
นาย ภิญโญ เหล่ามงคลชัย

อ.เกษตร์ ศิริสันติสัมฤทธิ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2537

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ ได้รายงานถึงวิธีการ การจดจำรูปแบบตัวพิมพ์โน้ตดนตรีโดยวิธีการหาลักษณะเด่นของตัวโน้ต ได้แก่จำนวนและตำแหน่งของจุดตัด,จุดแยก และจุดปลายของตัวโน้ตแต่ละตัว การจดจำรูปแบบตัวพิมพ์โน้ตดนตรีที่ได้นี้สามารถนำไปประยุกต์ใช้งานทางด้านดนตรีได้ โดยคอมพิวเตอร์จะรับภาพตัวพิมพ์โน้ตดนตรีจากสแกนเนอร์ แล้วใช้วิธีการประมวลผลภาพและการจดจำรูปแบบมาวิเคราะห์ภาพตัวพิมพ์โน้ตดนตรีและแปลงให้เป็นรหัสมิดี้ เพื่อนำไปควบคุมเครื่องดนตรีประเภทมิดี้ สำหรับในโครงงานฉบับนี้จะกล่าวถึง หลักการวิเคราะห์และการจดจำตัวพิมพ์โน้ตเป็นส่วนใหญ่ ส่วนหลักการของระบบมิดี้ ได้กล่าวไว้เพียงคร่าวๆเท่านั้น

การจดจำรูปแบบตัวพิมพ์โน้ตดนตรี เป็นการพิจารณาถึงโครงสร้างของตัวโน้ต ด้วยการแยกตัวโน้ตออกจากบรรทัด 5 เส้น จากนั้นจึงทำการแยกตัวโน้ตแต่ละตัวออกจากกันโดยใช้วิธีตีกรอบรอบตัวโน้ต ข้อมูลภาพของตัวโน้ตจะถูกทำให้เหลือแต่โครงร่าง โดยใช้วิธีการทำให้บางเพื่อนำโครงร่างของตัวโน้ตที่ได้ไปหาลักษณะเด่น แล้วตีความหมายให้ทราบระดับเสียงและความยาวเสียงของตัวโน้ต จึงนำไปสร้างเป็นฐานข้อมูล เพื่อแปลงเป็นรหัสมิดี้ สำหรับส่งไปควบคุมเครื่องดนตรีประเภทมิดี้ต่อไป

Recognition of Printed Music Note

Nirat Boonchoomanee
Pinyo Lhowmongkolchai
Kaset Sirisantisumrit Advisor
1994

Abstract

This thesis describes the process of Recognition of printed music note by find the characteristic of music note such as number and position of cross point , branch point and end point of music note. The Recognition of printed music note can be application in music by computer receive picture of note from scanner. Then use image processing and pattern recognition to analysis and convert to MIDI code for control MIDI instrument. Almost subject in this thesis that mention to analysis principle and printed music note and a few mention of midi system.

Recognition of printed music note is the method to analysis the structure of music note by extraction the staff line. Then separate each of music note by framing in order to find it's meaning. The image's data has to be reduced by Thinning method and used that to find out characteristic and encode. When we know pitch and length of music note. These would be used to make database for convert to MIDI code in order to control MIDI instrument.

สารบัญ

เรื่อง หน้า

สารบัญ (ต่อ)

เรื่อง หน้า

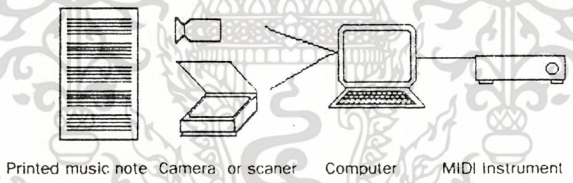
บทที่ 1
บทนำ

1.1 กล่าวนำ

ศิลปะที่ถ่ายทอดจินตนาการของมนุษย์ออกมาเป็นเสียงดนตรีที่ไพเราะ คือ เครื่องดนตรีนั่นเอง ซึ่งมนุษย์ได้พยายามค้นคว้า หาเครื่องดนตรีชนิดใหม่ๆ โดยใช้เทคโนโลยีที่ทันสมัย อันได้แก่ การนำเอาเทคโนโลยีทางด้านอิเล็กทรอนิกส์มาประยุกต์ใช้กับเครื่องดนตรี เช่น กีตาร์ไฟฟ้า, กลองไฟฟ้า,KEYBOARD และ ฯลฯ

เทคโนโลยีที่ได้รับความสนใจอย่างมากอีกชนิดหนึ่ง คือ MIDI (Musical Instrument Digital Interface) ซึ่งเป็นมาตราฐาน การสื่อสารข้อมูลแบบอนุกรมเพื่อใช้สำหรับให้เครื่องดนตรีอิเล็กทรอนิกส์ ที่มาจากบริษัทผู้ผลิตหลายบริษัท สามารถติดต่อสื่อสารกันได้ และเพื่อให้เครื่องคอมพิวเตอร์สามารถติดต่อสื่อสารกับเครื่องดนตรีอิเล็กทรอนิกส์เหล่านั้นได้อีกด้วย ทำให้การใช้งานมีความสะดวกในการประพันธ์เพลง หรือการใช้งานในห้องบันทึกเสียงต่างๆ เพราะผู้ใช้สามารถใช้โปรแกรมประเภทซีเควนเซอร์ (sequencer) ทำการป้อนตัวโน้ตดนตรีหรือรหัสควบคุมให้กับเครื่องเล่นดนตรี และสามารถนำบทเพลงมาแก้ไขดัดแปลงได้โดยง่าย

ในโครงงานนี้ได้นำเสนอถึงวิธีการใช้เครื่องดนตรีประเภท MIDI ให้มีความสะดวกมากขึ้น ด้วยวิธีการใช้วิธีการประมวลผลภาพ และการจดจำรูปแบบเข้ามาใช้วิเคราะห์ ภาพตัวพิมพ์โน้ตดนตรี และตีความหมายให้เป็นรหัส MIDI แทนการป้อนตัวโน้ตดนตรีจากเครื่องคอมพิวเตอร์ ดังรูปที่ 1.1



รูปที่ 1.1 การใช้งานเครื่องดนตรีมิดี้ร่วมกับการประมวลผลภาพ

1.2 ระบบมิดี้

1.2.1 มิดี้คืออะไร

มิดี้เป็นมาตรฐานการสื่อสารข้อมูล แบบอนุกรม ใช้สื่อสารระหว่างเครื่องดนตรีอิเล็กทรอนิกส์กับเครื่องคอมพิวเตอร์ โดยข้อมูลที่ส่งไปนั้นเป็นข้อมูลแบบดิจิตอล เครื่องดนตรีเหล่านี้ ได้แก่ ซินธิไซเซอร์ (เครื่องดนตรีประเภทลิ่มคีย์ที่เรามักเรียกว่า คีย์บอร์ด) เครื่องเป่าอิเล็กทรอนิกส์ (เช่น แซกโซโฟนมิดี้,แซกโซโฟนไฟฟ้า) กีตาร์ มิดี้ดรัมแมชีน (หรือที่นักดนตรีชอบเรียกว่า ริทึมบอกซ์ ทำหน้าที่เล่นเป็นจังหวะกลอง) โทนเจ็น (แหล่งกำเนิดเสียงที่รับข้อมูลทางมิดี้เท่านั้น) และอุปกรณ์อื่นที่มีการบ่งบอกไว้ว่าสามารถสื่อสารข้อมูลด้วยระบบมิดี้ได้

เราจะเรียกอุปกรณ์หรือเครื่องดนตรีที่สามารถสื่อสารทางมิดี้ได้ว่า midi device หรือเรียกแบบ ไทย ก็คือ อุปกรณ์มิดี้ โดยการใช้มิดี้เป็นตัวกลางในการสื่อสารกับอุปกรณ์มิดี้เหล่านี้ ก็จะทำให้เรามีสตูดิโอชั้นเยี่ยม

อยู่ในบ้านได้อย่างสบาย ทำให้การทำเพลงไม่ว่าจะทำเล่นๆหรือทำเพื่องานโฆษณาไปจนถึงการบันทึกลงเทป เพื่อทำเป็นการค้าเป็นเรื่องที่ไม่ยากเย็นนัก และที่สำคัญคือราคาที่ถูกมากเมื่อเทียบกับสตูดิโอที่ไม่ใช้ระบบมิดี้

1.2.2 ระบบมิดี้

ระบบมิดี้จะประกอบด้วยส่วนอินพุท คือ คีย์บอร์ด ซึ่งเป็นส่วนที่เราจะเล่นดนตรีลงไป โดยขณะที่เราเล่นโน้ตต่างๆนั้นข้อมูลของโน้ตจะถูกแปลงเป็นข้อมูลของมิดี้ที่แสดงคีย์ที่ถูกกด ส่งไปที่ midi in ของส่วน master ซึ่งปกติจะเป็นเครื่องซีเควนเซอร์หรือไมโครคอมพิวเตอร์ ทำหน้าที่ควบคุมหน่วยกำเนิดเสียง (sound module หรือ tone gen) แต่ละชิ้นให้เล่นเป็นโน้ตต่างๆกันเพื่อประสานเสียงกันเป็นเสียงดนตรี โดยแต่ละชิ้นก็จะแทนเสียงเครื่องดนตรีแต่ละประเภท

ข้อมูลที่ตัว master ส่งไปนั้นก็จะได้จากการบันทึก (record) ข้อมูลที่นักดนตรีเล่นผ่านเข้ามาทาง midi in เก็บไว้ในหน่วยความจำก่อน คล้ายๆกับการอัดเทปทั่วไปแต่ข้อมูลที่บันทึกเป็นข้อมูลมิดี้แทน ไม่ใช่เป็นเสียงเพลง ดังนั้นการแก้ไขข้อมูลดังกล่าวในกรณีที่นักดนตรีเล่นพลาด ย่อมทำได้ง่ายกว่าเพราะเป็นข้อมูลแบบดิจิตอล

เราจะเห็นว่าโดยการใช้ระบบมิดี้มาช่วยในการทำงานทางดนตรี เราไม่จำเป็นต้องมีนักดนตรีหลายคน เพื่อเล่นดนตรีในแต่ละชิ้นเราสามารถใช้ master กับนักดนตรีคนเดียวก็สามารถทำเพลงได้อย่างสบาย โดยให้ตัว master เล่นเครื่องดนตรีชิ้นอื่นแทนนักดนตรีคน อื่น ซึ่งความจริงก็คือเราบันทึกข้อมูลของแต่ละชิ้นเก็บไว้ใน master ก่อนนั้นเอง

1.2.3 ประเภทของอุปกรณ์มิดี้

อุปกรณ์มิดี้ทุกชิ้นจะมีสิ่งหนึ่งที่เหมือนกันคือ มีไมโครโปรเซสเซอร์เป็นหัวใจของการทำงาน แต่เราก็ยังแบ่งอุปกรณ์มิดี้ได้เป็น 2 พวกใหญ่ๆคือ

1. midi instrument พวกนี้มีหน้าตาเหมือนเครื่องดนตรีโดยทั่วไป แต่ติดตั้งระบบมิดี้เข้าไปด้วย เช่น ซินธิไซเซอร์ กีตาร์มิดี้ แซกโซโฟนไฟฟ้า กลองไฟฟ้า ปกติแล้วในตัวมันเองจะมีหน่วยกำเนิดเสียงอยู่ด้วย หรือพูดง่ายๆคือมันสามารถให้เสียงดนตรีออกมาได้ ถ้ามันไม่สามารถให้เสียงดนตรีออกมาได้เราจะเรียกมันว่า คอนโทรลเลอร์ เพราะตัวมันเองมีหน้าตาเป็นเครื่องดนตรีแต่ไม่สามารถให้เสียงดนตรีได้ ต้องส่งข้อมูลมิดี้ไปควบคุมหน่วยกำเนิดเสียงตัวอื่นให้ออกเสียงดนตรีแทน

2.non-midi instrument พวกนี้หน้าตาของมันจะไม่มีความเป็นเครื่องดนตรีเลย แต่จะมีปุ่มฟังชั่นเต็มไปหมดแล้วอาจจะมีจอแสดงผลเล็กๆไปจนถึงจอ LCD ขนาดใหญ่ที่เราสามารถใช้พูลดาวน์เมนูได้ อุปกรณ์พวกนี้ได้แก่ ซีเควนเซอร์ ดรัมแมชีน เป็นต้น หน้าที่โดยรวมของมันคือการประมวลผลเกี่ยวกับข้อมูลทางมิดี้ทั้งหมด และการควบคุมอุปกรณ์มิดี้ที่มีหน่วยกำเนิดเสียงให้เล่นเป็นเพลงออกมา จะว่าไปแล้วอุปกรณ์เหล่านี้ก็เปรียบเสมือน conductor ที่คอยดูแลให้นักดนตรีแต่ละคนเล่นดนตรีนั่นเอง

1.2.4 ภายในตัวมิดี้

การสื่อสารข้อมูลของมิดี้เป็นการสื่อสารแบบอะซิงโครนัส ด้วยความเร็ว 31250 บิทต่อวินาที ประกอบด้วย start bit และ stop bit อย่างละ 1 บิท และมีขนาดของข้อมูล 8 บิท รวมเป็น 10 บิท สายที่ใช้เป็นสายตีเกลียว (twisted-pair) มีชิลด์ (shield) พันรอบ ขนาดของความยาวสายไม่ควรเกิน 1.5 เมตร ใช้ขั้วต่อ (connector) แบบ 5 pin DIN ตัวผู้ มิดี้จะใช้เพียง pin 4,5 เป็นสายสัญญาณ pin ที่ 2 จะต่ออยู่กับกราวด์ (GND)

สำหรับพอร์ตของมิดี้ที่ติดกับอุปกรณ์มิดี้ก็จะเป็น DIN ตัวเมีย สามารถแบ่งพอร์ตได้ 3 ประเภทคือ MIDI IN,MIDI OUT และ MIDI THRU โดย MIDI IN จะเป็นพอร์ตที่รับข้อมูลจากอุปกรณ์ภายนอก MIDI OUT จะเป็นพอร์ตที่ส่งข้อมูลออกในขณะที่มีการเล่น ส่วน MIDI THRU จะเป็นพอร์ตที่ส่งข้อมูลออกจากตัวอุปกรณ์แต่เป็นข้อมูลที่เหมือนกับได้รับจาก MIDI IN,MIDI THRU จึงเปรียบเสมือนปลั๊ก 3 ตา ที่ไว้สำหรับส่งผ่านข้อมูลจาก master ไปยังอุปกรณ์ตัวอื่นนั่นเอง

เราจะเห็นว่าข้อมูลมิดี้ที่ส่งจาก master เข้าสู่อุปกรณ์ตัวแรกแล้ว ข้อมูลชุดเดิมก็จะส่งต่อไปยังอุปกรณ์ตัวต่อไปที่มีข้อมูลเหมือนกันทุกประการ แต่อุปกรณ์แต่ละตัวจะตอบสนองเฉพาะข้อมูลบางชุดเท่านั้น โดยข้อมูลที่ส่งไปแต่ละชุดนั้น (package) จะประกอบด้วยจำนวนไบท์ที่แตกต่างกัน (ชุดข้อมูลที่ส่งไปนี้จะมีไบท์แสดงสถานะหรือในภาษาคอมพิวเตอร์เรียกว่า heading) เป็นตัวแสดงว่า ชุดข้อมูลดังกล่าวเป็นของอุปกรณ์ตัวใด โดยในระบบมิดี้ ตัว master จะควบคุมอุปกรณ์มิดี้ได้ 16 ตัว อุปกรณ์แต่ละตัวก็จะถูกกำหนดให้ตอบสนองเฉพาะแชนแนล (channel) ใดแชนแนลหนึ่งใน 16 แชนแนล เราสามารถเพิ่มอุปกรณ์มิดี้ในระบบให้มากกว่า 16 ตัว ได้ โดยการกำหนดให้อุปกรณ์บางตัวตอบสนองในแชนแนลที่เหมือนกัน แต่อุปกรณ์คู่ดังกล่าวก็จะทำงานตามข้อมูลที่ได้รับจากมิดี้เหมือนกันทุกประการ เช่น ทั้งคู่จะเล่นโน้ต โด เร มี พร้อมๆกัน

และในทางตรงกันข้ามเราสามารถลดจำนวนอุปกรณ์มิดี้ในระบบให้น้อยกว่า 16 ตัวได้ โดยใช้อุปกรณ์มิดี้ที่สามารถประพฤติตัวคล้ายกับเป็นอุปกรณ์หลายๆตัวได้ (เช่น ซินธิไซเซอร์) ภายในจะมีหน่วยกำเนิดเสียงหลายๆตัว (module) แล้วผู้ใช้ก็จะกำหนดให้แต่ละโมดูลตอบสนองในแต่ละแชนแนลเท่านั้น ลักษณะความสามารถอันนี้เราเรียกว่า Multitimbral

1.2.5 รูปแบบของการส่งข้อมูลมิดี้

การส่งข้อมูลในระบบมิดี้จะเป็นการส่งในลักษณะ "ชุดข้อมูล" (package) โดยในไบท์แรกของแต่ละชุดข้อมูล (heading) ซึ่งในมิดี้เราเรียกว่า ไบท์แสดงสถานะจะมี MSB เป็น 1 (1xxxxxxxb) และข้อมูลที่ตามไบท์แสดงสถานะซึ่งก็คือไบท์ข้อมูลจะมี MSB เป็น 0 ดังนั้นเราจะสรุปได้ว่า MSB ที่ส่งในระบบมิดี้จะเป็นตัวที่ชี้ว่าข้อมูลตัวนั้นไบท์แสดงสถานะหรือไบท์ข้อมูลทำให้สามารถใช้เพียง 7 บิทที่เหลือมาแสดงค่าความหมายต่างๆทางดนตรีได้เท่านั้น

ไบท์แสดงสถานะเป็นตัวที่แสดงว่าไบท์ข้อมูลที่ตามมามีความหมายถึงอะไรในทางดนตรีเราสามารถแบ่งประเภทของไบท์แสดงสถานะได้ 2 ประเภทคือ ประเภทข้อมูลแสดงแชนแนล (channel massage) และประเภทข้อมูลแสดงระบบ (system massage)

1.2.6 ข้อมูลแสดงแชนแนล

ไบท์แสดงสถานะที่เป็นประเภทข้อมูลแสดงแชนแนลจะส่งตรงไปให้กับแชนแนลใดแชนแนลหนึ่งในระบบเท่านั้น โดยที่ไบท์แสดงสถานะประเภทนี้จะบ่งบอกถึงแชนแนลที่ส่งไปในบิทที่ 0-3 ถ้าอุปกรณ์มิดี้ที่เป็นสลาฟ (slave) ตัวใดถูกเซ็ตให้มีแชนแนลตรงกับแชนแนลที่ส่งไปนี้สลาฟตัวนั้นจึงจะตอบสนองต่อไบท์ข้อมูลที่ตามมา ข้อมูลแสดงแชนแนลจะแบ่งออกเป็นข้อมูลเสียง (voice massage) กับข้อมูลโหมด (mode massage)

ข้อมูลเสียง (Channel voice massage)

1. Note On,Note Off จะเกี่ยวข้องถึงการเล่นโน้ตต่างๆบนอุปกรณ์มิดี้
2. Control Change ข้อมูลชุดนี้จะเป็นการควบคุมเกี่ยวกับตัวควบคุมเสียงบางประเภท เช่น ความดัง (volume)
3. Pich Bend Change เกี่ยวข้องกับการเอื้อนเสียงของแชนแนล
4. Program Change จะใช้ในการเปลี่ยนเสียงของแชนแนลนั้นๆ
5. Polyphanic Key ข้อมูลชุดนี้จะส่งเมื่อมีการขยี้คีย์หลังจากที่เราเล่นโน้ตไปแล้ว
6. Channel Pressure ข้อมูลนี้จะเกี่ยวข้องกับการขยี้บนคีย์เช่นกันแต่จะมีผลต่อเสียงทั้งแชนแนลไม่เฉพาะเจาะจงเฉพาะคีย์ในข้อ 5

ข้อมูลโหมด (Channel mode massage)

เป็นการควบคุมโหมดในการทำงานของแต่ละแชนแนลในระบบโดยจะมีโหมดการทำงานดังนี้

1. Poly/Mono คือการเซ็ตให้มีความสามารถในการเปล่งเสียงโน้ตแบบ Mono คือ 1 โน้ตเท่านั้น หรือเป็นแบบ Poly ที่สามารถเปล่งเสียงได้หลายๆโน้ตพร้อมกัน
2. Omni On/Off เป็นการบังคับให้แชนแนลนั้นตอบสนองเฉพาะข้อมูลที่ตรงกับแชนแนลของตน หรือตอบสนองในทุกๆแชนแนล ถ้า On ตอบสนองในทุกแชนแนลโดยปกติ ถ้านำอุปกรณ์มิดี้มาต่อกันเป็นระบบแล้วโหมดที่ใช้งานคือ Poly:Omni OFF

1.2.7 ข้อมูลระบบ (System massage)

ข้อมูลประเภทนี้จะส่งไปโดยไม่เจาะจงแชนแนลใดแชนแนลหนึ่งในระบบทุกๆแชนแนลจะตอบรับข้อมูลเหล่านี้ แบ่งออกได้เป็น

1. Common Massage ข้อมูลชุดนี้จะเกี่ยวข้องกับการเลือกเพลง การเลือกจังหวะ การเลือกตำแหน่งของเพลงที่จะเล่น และการบังคับให้อุปกรณ์ทางอนาล็อกปรับแต่งระบบเสียงของตัวเอง (tune request)
2. System Real Time จะเกี่ยวข้องโดยตรงกับการสั่งเริ่มเล่นเพลง (start) การหยุด (stop) และข้อมูลที่สำคัญคือ system clock ที่เปรียบเสมือน clock ของเพลงที่ทำให้เพลงเดินเร็วหรือช้า ถ้าความถี่ในการส่ง system clock มาก เพลงก็จะเดินเร็วตามขึ้นมาด้วย

บทที่ 2
ทฤษฎีและหลักการของระบบการประมวลผลภาพ

2.1 ประวัติของระบบการประมวลผลภาพดิจิตอล

ความสนใจเกี่ยวกับการประมวลผลข้อมูลภาพ ได้มีการประยุกต์ใช้งานครั้งแรกเมื่อมีการส่งภาพดิจิตอลของหนังสือพิมพ์ ผ่านเคเบิลใต้น้ำ (Submarin cable) ผ่านมหาสมุทรแอตแลนติกระหว่างลอนดอนกับนิวยอร์ก ในปี คศ. 1920 ทำให้ลดระยะเวลาในการส่งจากนานเป็นสัปดาห์ให้เหลือแค่ 3 ชั่วโมง โดยปลายทางจะรับภาพโดยพิมพ์ออกทางเครื่องโทรพิมพ์ (Telegraph printer) แต่คุณภาพของภาพที่ส่งไปยังไม่ดีพอ ในส่วนการปรับปรุงจะต้องเลือกลักษณะการพิมพ์และระบบส่งภาพให้มีความสัมพันธ์กับระดับความสว่าง จนกระทั่งในปี คศ. 1921 ก็ได้มีการพัฒนาในส่วนนี้จนสำเร็จ หลังจากนั้นก็พัฒนาโดยการเพิ่มระดับความสว่างให้มากขึ้นจนถึง 15 ระดับ ในปีคศ. 1929 อีก 35 ปีต่อมาคือในปี 1964 ได้มีการนำคอมพิวเตอร์มาปรับปรุงภาพดิจิตอล โดยได้มีการวิจัยที่ห้องทดลอง Jet Propulsion Laboratory (Pasadena California) ภาพของดวงจันทร์ได้ถูกส่งกลับมาจากยานอวกาศ Ranger 7 โดยภาพถูกประมวลโดยคอมพิวเตอร์ จากปีคศ.1964 เป็นต้นมาสาขาทางด้านการประมวลผลข้อมูลภาพได้ถูกนำไปใช้งานทางด้านต่างๆมากมายเช่น ด้านการแพทย์ ด้านการอุตสาหกรรม หรือเทคโนโลยีดาวเทียมเ ป็นต้น

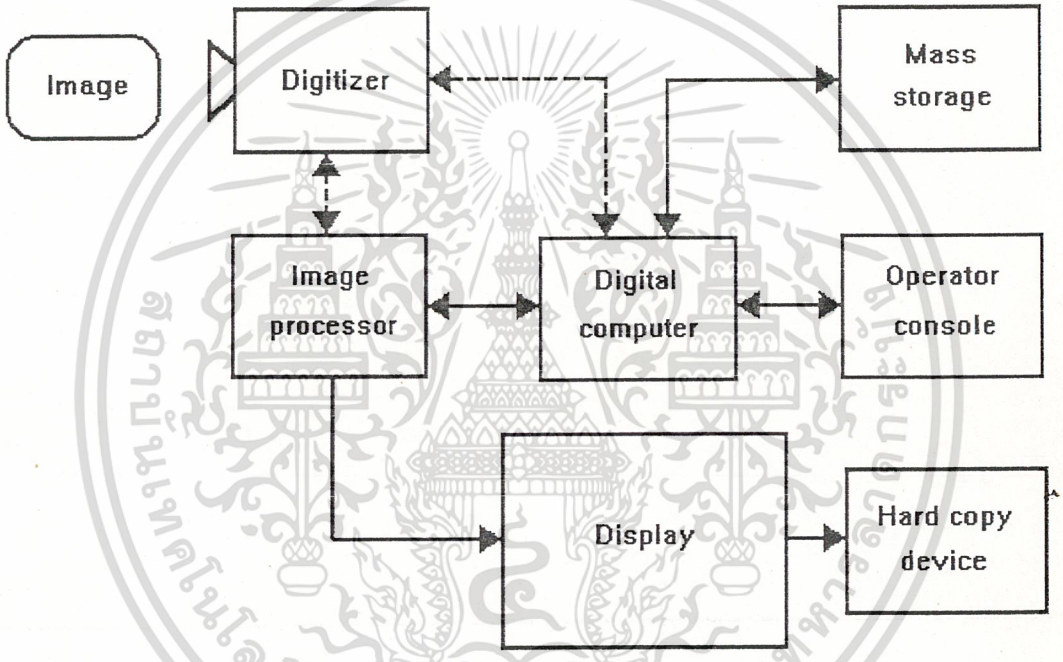
2.2 การนำเสนอข้อมูลภาพดิจิตอล (Digital image representation)

คำว่า อิมเมจ (image) หมายถึง ความเข้มของแสงซึ่งแสดงได้ด้วยฟังชั่นความเข้มของแสงในระนาบ 2 มิติ f(x,y) โดย x และ y เป็นโคออดิเนทที่เกิดขึ้นที่ภาพจริง ณ.จุดต่างๆและค่าของฟังชั่น f ณ จุด (x,y) ใดๆ จะเป็นสัดส่วนโดยตรงกับความสว่างหรือระดับเทา (gray level) ของภาพที่จุดนั้นๆ ซึ่งแสดงได้ดังรูปที่ 2.1

รูปที่ 2.1 ข้อมูลภาพแบบดิจิตอลแสดงถึงแนวแกน x และแกน y และฟังชั่นแสดงความเข้มของแสง ณ. จุด (x,y) ใดๆ

ในการพิจารณาข้อมูลภาพแบบดิจิตอลจะแทนด้วยเมตริกซ์หนึ่งซึ่งมีแถวและหลักที่มีลักษณะเป็นเอกลักษณ์ ก็จะได้ค่าของระดับเทา ณ. จุดต่างๆ โคออดิเนทหรือจุดต่างๆที่เกิดขึ้นในเมตริกซ์เรียกว่า พิกเซล (pixel) หรือจุดย่อยของภาพ (picture element) โดยทั่วไปขนาดของข้อมูลภาพสามารถเปลี่ยนแปลงได้แล้วแต่การใช้งาน โดยมากจะเลือกเป็นสี่เหลี่ยมจัตุรัส และจะแบ่งระดับเทาด้วยตัวเลขจำนวนเต็มยกกำลังสอง เช่นพื้นที่ 512x512 และมีระดับเทา 128 ระดับ

2.3 องค์ประกอบของระบบประมวลผลภาพดิจิตอล (Element of a digital image processing system)



รูปที่ 2.2 องค์ประกอบของระบบประมวลผลข้อมูลภาพ

ส่วนประกอบพื้นฐานของระบบประมวลผลข้อมูลภาพ โดยทั่วไปแสดงได้ดังรูป การทำงานในแต่ละบล็อกอธิบายได้ดังนี้

2.3.1 ตัวประมวลผลภาพดิจิตอล (Image processor)

ตัวประมวลผลภาพดิจิตอลเป็นหัวใจของระบบประมวลผลภาพ ตัวประมวลผลภาพประกอบด้วย ชุดโมดูลของฮาร์ดแวร์ ซึ่งมีหน้าที่การทำงาน 4 อย่างคือ การได้มาซึ่งภาพดิจิตอล,เก็บข้อมูล,ประมวลผลระดับล่าง และ แสดงผล โมดูลของภาพดิจิตอลจะมีสัญญาณโทรทัศน์เป็นอินพุทและแปลงสัญญาณอินพุทให้เป็นสัญญาณดิจิตอล ตัวประมวลผลภาพที่ทันสมัยสามารถทำภาพให้เป็นดิจิตอลภายในเวลา 1 เฟรม (1/30 วินาที) ด้วยเหตุนี้จึงเรียกตัวที่นำภาพออกมาแสดงว่าเฟรมแกรบเบอร์ (frame grabber)

โมดูลเก็บข้อมูลภาพ หรือ เฟรมบัฟเฟอร์ (frame buffer) เป็นหน่วยความจำที่สามารถเก็บภาพดิจิตอลโดยทั่วไป โมดูลเก็บภาพจะรวมอยู่ในตัวประมวลผลภาพ คุณสมบัติที่เด่นชัดของตัวเก็บข้อมูลภาพคือสารบัญของหน่วยความจำสามารถโหลดหรือผ่านที่อัตราความเร็วของโทรทัศน์ (30 ภาพต่อวินาที) และกลับกัน หน่วยความจำสามารถทำแอดเดรสที่อัตราความเร็วของโทรทัศน์ด้วยโมดูลแสดงผล โดยเอาท์พุทจะออกมาที่จอมอนิเตอร์ตำแหน่งของหน่วยความจำสามารถขยายหรือเลื่อนในแนวตั้งและแนวนอนได้โมดูล ประมวลผลภาพทำหน้าที่ในระดับล่างเช่นเดียวกับการกระทำเชิงเลขและลอจิก (Arithmetic-Logic Operation) โมดูลนี้จึงมักถูกเรียกว่าหน่วยกระทำเชิงเลขและลอจิก (Arithmetic-Logic Unit ; ALU) ส่วนนี้เป็นฮาร์ดแวร์ที่ออกแบบเป็นพิเศษโดยให้อัตราความเร็วในการประมวลผลภาพเป็นแบบขนาน

2.3.2 ดิจิไตเซอร์ (Digitizer)

ดิจิไตเซอร์เปลี่ยนสัญญาณข้อมูลภาพให้เป็นข้อมูลเชิงเลขเพื่อเป็นอินพุทให้กับดิจิตอลคอมพิวเตอร์ (digital computer) จำพวกของอุปกรณ์ที่ใช้กันได้แก่ ไมโครเดนซิโตมิเตอร์ (microdensitometer),สแกนเนอร์ (scanner),กล้องวีดิคอน (vidicon camera) และโฟโต้เซนซิทีฟโซลิดสเตทอาร์เรย์ (photo sensitive solid-state arrays) อุปกรณ์ 2 ชนิดแรกจะต้องมีภาพหรือฟิล์มโปร่งแสงมาป้อนก่อนทำการดิจิไตซ์ ส่วนอุปกรณ์ชนิดอื่นสามารถบันทึกเก็บไว้เป็นข้อมูลภาพได้

2.3.3 ดิจิตอลคอมพิวเตอร์ (Digital computer)

จากที่กล่าวมาแล้วเกี่ยวกับตัวประมวลผลข้อมูลภาพ (image processor) แต่ระดับการประมวลผลของตัวประมวลผลยังต่ำดังนั้นโดยทั่วไปเราจะพบว่า ตัวประมวลผลภาพจะต่อเชื่อม (interface) กับคอมพิวเตอร์ ระบบคอมพิวเตอร์ที่ใช้สำหรับการประมวลผลภาพ ในย่านตั้งแต่ไมโครโปรเซสเซอร์จนถึงระบบคอมพิวเตอร์ขนาดใหญ่ สามารถที่จะคำนวณฟังชั่นที่ซับซ้อนบนพื้นที่ของภาพใหญ่ๆได้ ในระบบการประมวลผลภาพขนาดเล็กจะใช้มินิคอมพิวเตอร์หรือไมโครคอมพิวเตอร์ก็เพียงพอ กรณีข้อมูลภาพมีขนาดใหญ่จะต้องใช้คอมพิวเตอร์ระดับเมนเฟรม ในการใช้งานการประมวลผลภาพขณะทำการประมวลจะต้องใช้หน่วยความจำมาก หน่วยความจำหลักอาจไม่เพียงพอจึงต้องมีอุปกรณ์ต่อพ่วงขณะทำงาน

2.3.4 อุปกรณ์เก็บข้อมูล (Storage device)

ข้อมูลภาพขนาด 512x512 พิกเซล แต่ละพิกเซลมีการควอนไตซ์ 8 บิท จะต้องใช้หน่วยความจำ 0.25 เมกะไบท์ ตัวเก็บข้อมูลหลักๆมีอยู่ 3 ชนิดคือ แผ่นแม่เหล็ก (magnetic disk),เทปแม่เหล็ก (magnetic tape)และแผ่นแสง (optical disk) แผ่นแม่เหล็กมีความจุ 700 เมกะไบท์หรือมากกว่า เทปแม่เหล็กแบบความหนาแน่นสูง (6400 ไบท์ต่อนิ้ว) สามารถเก็บข้อมูลภาพโดยประมาณ 4 ฟุต แผ่นแสงทำงานโดยอาศัยเลเซอร์ในการอ่านหรือเขียนสามารถจุข้อมูลได้ 4 กิกะไบท์หรือประมาณ 16000 ข้อมูลภาพต่อแผ่น

2.3.5 อุปกรณ์แสดงผล (Display device)

อุปกรณ์ทันสมัยที่ใช้ในการแสดงผลการประมวลผลภาพคือ จอมอนิเตอร์แบบโมโนโครมและจอสี มอนิเตอร์จะขับเอาท์พุทซึ่งได้มาจาก ส่วนแสดงผลข้อมูลภายในตัวประมวลผลภาพ สัญญาณดังกล่าว สามารถ นำไปเก็บบันทึกไว้ได้ในอุปกรณ์บันทึกสัญญาณเช่น สไลด์,ภาพถ่ายและฟิล์ม อุปกรณ์แสดงผลอื่นๆ เช่น จอแบบหลอดคาโถด (cathode ray tube) และอุปกรณ์เครื่องพิมพ์ (printing device)

2.4 หลักการเบื้องต้นของการประมวลผลภาพ

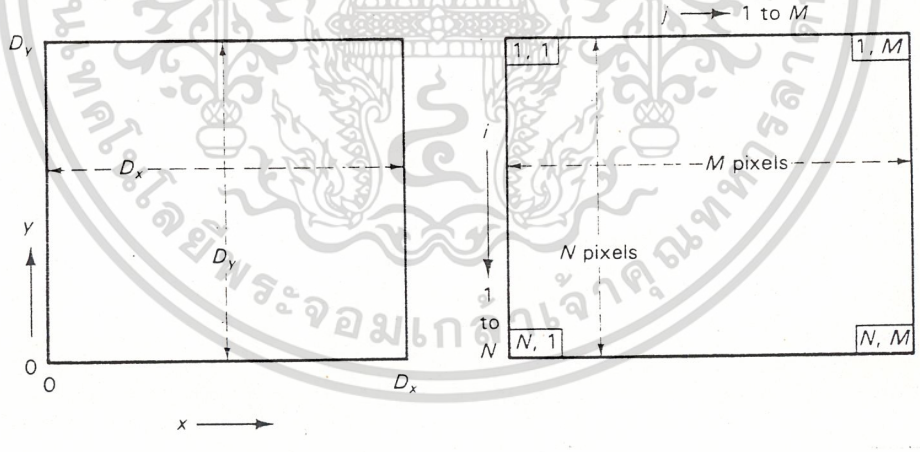
ในหัวข้อแรกจะแสดงถึงความหมายของคำต่างๆที่ใช้ในการประมวลผลภาพ

2.4.1 พิกเซล (pixel)

การแสดงข้อมูลภาพดิจิตอลสามารถอธิบายได้ด้วยเมตริกซ์ NxM และให้จุดต่างๆที่อยู่ในเมตริกซ์เป็นจุด x,y ใดๆ ซึ่งเป็นส่วนประกอบของภาพ ในแต่ละจุด x,y ใดๆ เรียกว่า พิกเซลหรือจุดภาพ และในแต่ละพิกเซลจะแสดงให้เราเห็นได้ด้วยฟังชั่นของความเข้มของแสง (องค์ประกอบของ p(i,j)) เมื่อเราเปรียบเทียบระหว่างภาพและพิกเซลเมตริกซ์ (pixel metrix) ดังรูปที่ 2.3 จะเห็นว่าจุดกำเนิดของภาพจะอยู่ที่มุมล่างซ้าย แต่จุดกำเนิดของพิกเซลจะอยู่ที่มุมบนซ้าย ซึ่งจะเป็นลักษณะการประมวลผลภาพในกราฟฟิกของคอมพิวเตอร์

กล่าวคือ i = x เมื่อ 1 <= i <= N

j = (M - y)1 <= j <= M



a) ลักษณะของรูปภาพ b) ลักษณะของพิกเซลเมตริกซ์

รูปที่ 2.3 ความสัมพันธ์ของภาพโดยทั่วไปกับพิกเซลเมตริกซ์

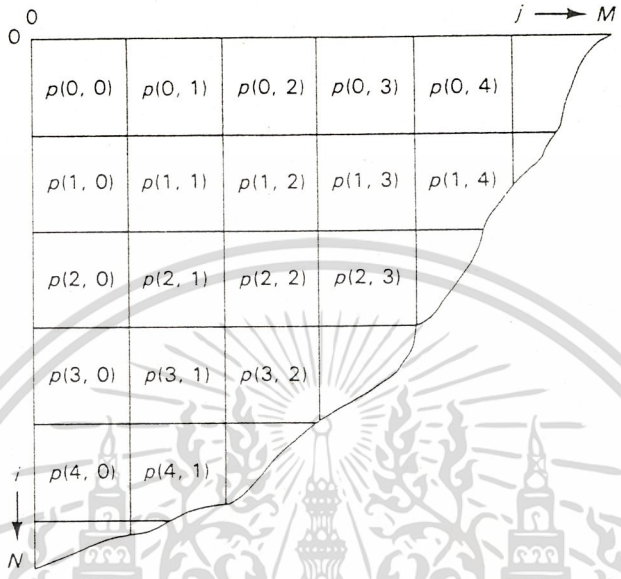
เมื่อ x = Dx/N

y = Dy/M

N = จำนวนสูงสุดของพิกเซลในแนวตั้ง

M = จำนวนสูงสุดของพิกเซลในแนวนอน

เมื่อเราให้จุดต่างๆบนเมตริกซ์เป็น p(i,j) ใดๆ

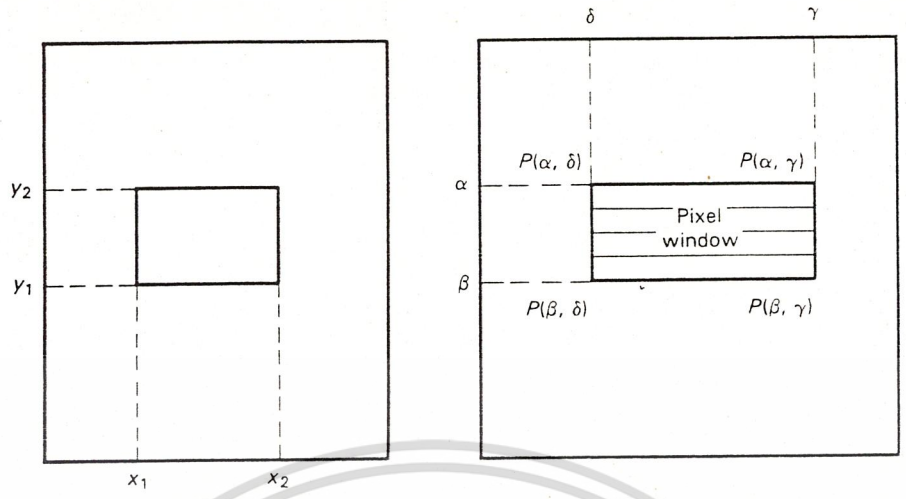


รูปที่ 2.4 ดัชนีของพิกเซลในเมตริกซ์

ค่าของพิกเซลหรือฟังชั่น p(i,j) ณ.จุดใดๆ จะแสดงได้ด้วยค่าของความเข้มของแสงซึ่งอาจแบ่งได้หลายระดับ ถ้ามีแค่ 2 ระดับก็จะเป็นแค่ 0 กับ 1

2.4.2 หน้าต่าง (windows)

เป็นพื้นที่ส่วนย่อยของภาพหรือเรียกว่า หน้าต่าง และสามารถกำหนดได้ด้วยมุม 4 มุมแสดงด้วยค่าของพิกเซล P(β ,δ),P(β,γ),P(α ,δ),P(α ,γ) แสดงได้ดังรูป 2.5

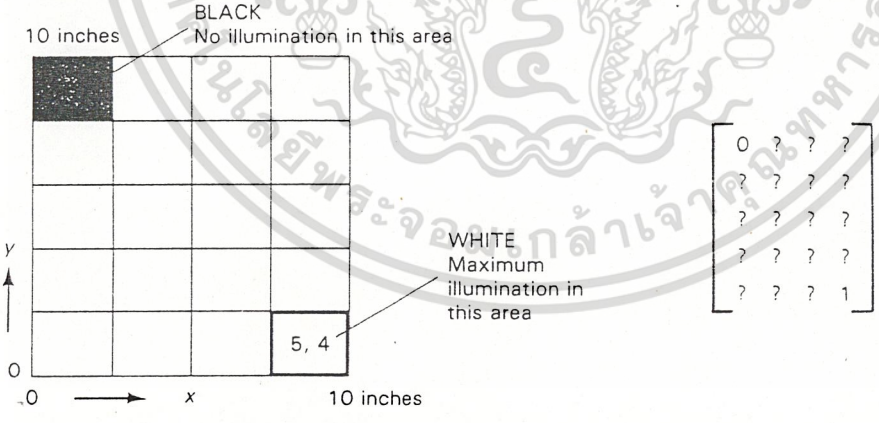


a) หน้าต่างของภาพ b) หน้าต่างของพิกเซล

รูปที่ 2.5 แสดงหน้าต่างของภาพและหน้าต่างของพิกเซล

2.4.3 ตำแหน่งของพิกเซล (pixel position)

ตำแหน่งของจุดภาพหรือพิกเซลทุกจุดจะต้องอยู่ภายในพื้นที่ NXM เมื่อเราพิจารณาจากรูปที่ 2.6 (a) จะเห็นว่าไม่มีแสงณ.บริเวณมุมบนซ้าย และบริเวณที่สว่างที่สุดอยู่ที่มุมล่างขวาของภาพซึ่งมีขนาด 10x10 นิ้ว พื้นที่ที่ไม่มีแสงแสดงได้ด้วยศูนย์ ส่วนพื้นที่ ที่สว่างที่สุดแสดงได้ด้วยหนึ่ง และภาพที่เห็นได้คือ 5x5 เมตริกซ์ (5 แถว,5 คอลัมน์) แต่ละส่วนของภาพจะกว้าง 2.5 นิ้ว บริเวณมุมบนซ้ายจะเป็นศูนย์ และพื้นที่ 2.5x2 นิ้วจะเป็นหนึ่ง ซึ่งนำมาเขียนเป็นเมตริกซ์ได้ดังรูปที่ 2.6 (b)



a) ภาพ b) แปลงเป็นพิกเซลเมตริกซ์

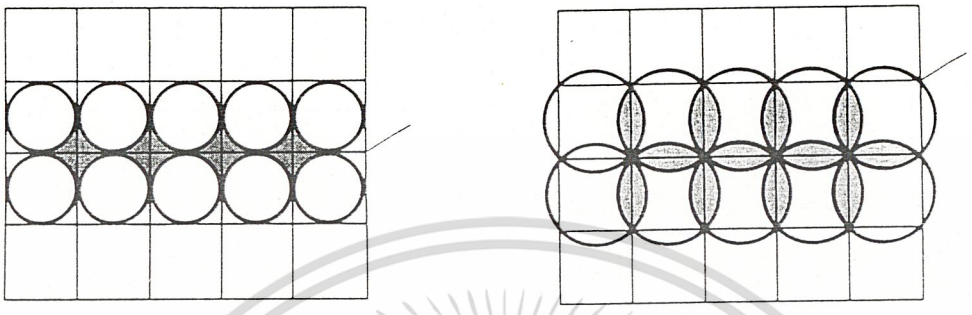
รูปที่ 2.6 การแปลงภาพให้เป็นพิกเซลเมตริกซ์

จากเมตริกซ์ในรูปที่ 2. 6(b) ถ้ามีระดับเทา (gray scale) เป็น 16 ระดับค่าของพิกเซล ณ.บริเวณที่สว่างที่สุดจะมีค่าเป็น 15

ลักษณะการแสดงผลในแต่ละพิกเซลหรือจุดภาพจะขึ้นอยู่กับเซ็นเซอร์ซึ่งมีอยู่ 2 ลักษณะ

- แสดงผลเฉพาะพิกเซล
- แสดงผลเหลื่อมกัน

ซึ่งแสดงดังรูปที่ 2.7



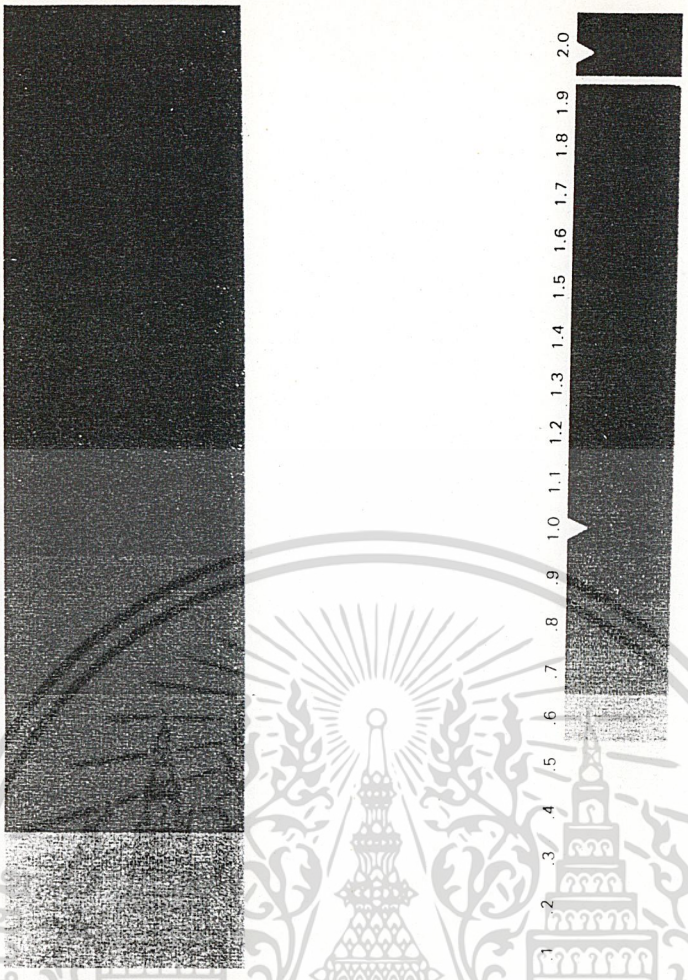
a) แสดงผลเฉพาะพิกเซล b) แสดงผลเหลื่อมกัน

รูปที่ 2.7 การแสดงผลของจุดภาพ

2.4.4 **ระดับเทา** (gray scale)

ระดับเทา เป็นค่าที่บอกถึงความสว่างของจุดภาพ เราสามารถเพิ่มความสว่างหรือระดับเทาให้กับจุดภาพได้หลายระดับ โดยการเพิ่มจำนวนของบิทในการนำเสนอค่าของพิกเซล ตัวอย่างเช่น ต้องการระดับความสว่าง 4 ระดับ ก็ต้องใช้บิทข้อมูลจำนวน 2 บิท, 16 ระดับต้องใช้ 4 บิทและ 256 ระดับใช้ 8 บิท จำนวนของระดับความสว่างหาได้จาก 2 ยกกำลังด้วยจำนวนบิท

ระดับเทา		ค่าย่านของระดับเทา
2^1	2 ค่า	0 และ 1
2^3	8 ค่า	0 ถึง 7
2^4	16 ค่า	0 ถึง 15
2^8	256 ค่า	0 ถึง 255



รูปที่ 2.8 ตัวอย่างของระดับเทา

2.4.5 **ฮิสโตแกรม (histograms)**

ฮิสโตแกรมเป็นกราฟแท่งที่บอกถึงความถี่ของแต่ละความเข้มของแสง (gray scale) ของภาพ จากรูปในแนวแกน x เป็นค่าของระดับเทา และแกน y เป็นจำนวนของจุดภาพที่มีระดับเทาต่างๆ

เราสามารถสร้างฮิสโตแกรมได้โดย

1) ทำภาพให้เป็นระดับที่แตกต่างกัน

2) นับจำนวนจุดภาพที่มีระดับเทาเดียวกัน ทุกระดับเทา

3) พล็อตความถี่ของจุดภาพที่แต่ละระดับเทา

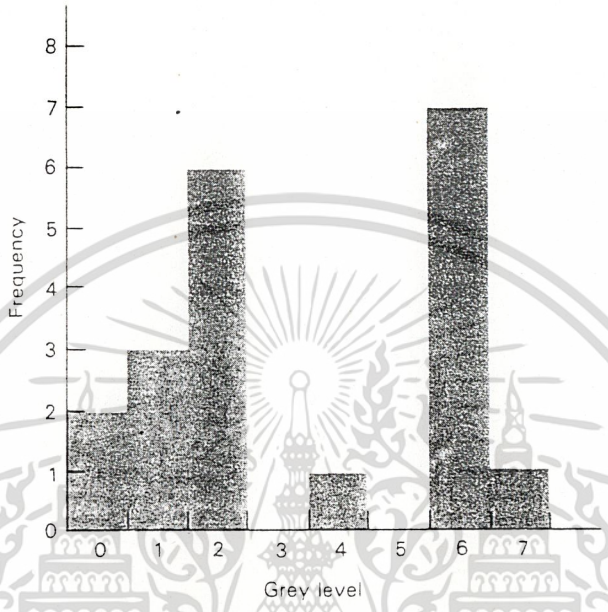
โดยเราสามารถหาความน่าจะเป็นของจุดภาพ ณ.ระดับเทาหนึ่งๆได้ตามสมการ

P(b) ณ.จุด (x,y) ในภาพ = ค่าของ b / จำนวนของพิกเซลทั้งหมดในภาพ

เช่น ที่ระดับเทา 6 ค่าของฮิสโตแกรมเท่ากับ 7 จะได้

$$p(6) = \frac{7}{20}$$

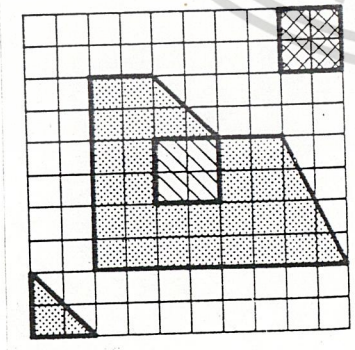
$$= 0.35$$



รูปที่ 2.9 ฮิสโตแกรมที่มี 8 ระดับเทา

รูปร่างของฮิสโตแกรมจะขึ้นอยู่กับคุณสมบัติของภาพ ประโยชน์ของฮิสโตแกรมคือใช้สำหรับปรับค่าเทรสโฮล (threshold) เพื่อจะแปลงระดับเทาของภาพให้เป็นภาพ 2 ระดับหรือใช้สำหรับปรับแต่งส่วนของสเปกตรัมระดับเทา

ตัวอย่างของภาพที่มีระดับเทาต่างกันเมื่อนำมาสร้างเป็นฮิสโตแกรม

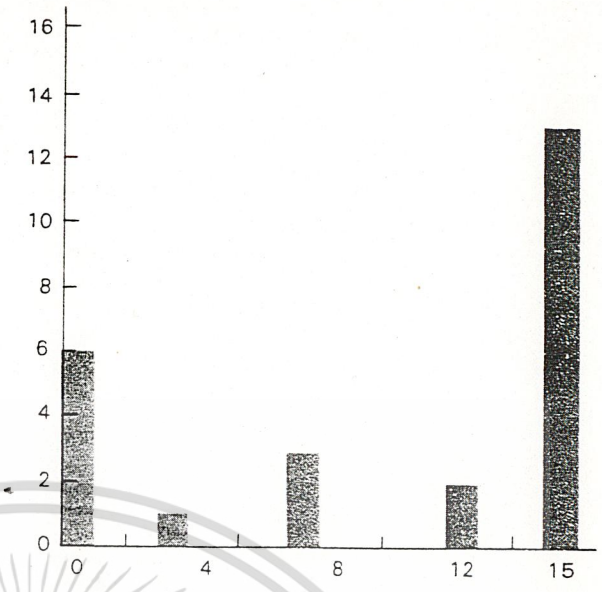


a) ขั้นที่ 1

$$\begin{bmatrix} 15 & 15 & 15 & 15 & 12 \\ 15 & 0 & 7 & 15 & 15 \\ 15 & 0 & 7 & 0 & 12 \\ 15 & 0 & 0 & 0 & 3 \\ 7 & 15 & 15 & 15 & 15 \end{bmatrix}$$

b) ขั้นที่ 2

Value	Number of Pixels
0	6
3	1
7	3
12	2
15	13
Total	25



c) ขั้นที่ 3 d) ขั้นที่ 4

รูปที่ 2.10 การสร้างฮิสโตแกรมจากภาพ

วิธีการ

1) หาจำนวนทั้งหมดของพิกเซลในเมตริกซ์ M x N

จะเห็นว่า M = 10 N = 10

พิกเซลทั้งหมด = 10 x 10 = 100

2) สร้างพื้นที่ของภาพแทนด้วยเมตริกซ์ จากตัวอย่างจะได้ เมตริกซ์ 5x5 จำนวนของพิกเซลทั้งหมดที่แทนในเมตริกซ์จะลดลงเหลือ 25 พิกเซล

3) ทำตารางความสัมพันธ์ระหว่างค่าระดับเทาและจำนวนของพิกเซล

4) สร้างฮิสโตแกรมเป็นกราฟแท่งโดยให้ระดับเทาเพิ่มทีละ 1 ระดับในแนวแกน x และเนื่องจากค่าระดับเทาสูงสุดคือ 15 จึงมีระดับเทาทั้งหมด 16 ระดับในแนวแกน y

2.5 การจำลองภาพ

จากคำจำกัดความของ อิมเมจ (image) หมายถึง ฟังชั่นความเข้มของแสงในระนาบ 2 มิติ คือ f(x,y) เมื่อค่าของ f ที่อยู่บนโคออดิเนท (x,y) คือความสว่างบนภาพ ณ.จุดนั้นๆ แสงจะอยู่ในรูปของพลังงาน ค่าฟังชั่น f(x,y) จึงไม่เป็นศูนย์และมีค่าๆหนึ่งนั่นคือ

$$0 < f(x,y) < \infty \quad .....(2.1)$$

เราเห็นภาพของวัตถุได้โดยการอาศัยการสะท้อนของแสงที่ไปกระทบวัตถุ ซึ่งสามารถอธิบายฟังชั่น f(x,y) ได้ด้วยส่วนประกอบ 2 ส่วน ส่วนประกอบอันที่ 1 คือ แสงที่ส่องไปยังวัตถุ (illumination component) และ

ส่วนประกอบที่ 2 คือ แสงที่สะท้อนกลับมาจากวัตถุ (reflectance component) และสามารถเขียนเป็นฟังชั่นได้ คือ i(x,y) และ r(x,y) ตามลำดับ ฟังชั่น f(x,y) เกิดจากการคูณกันระหว่าง i(x,y) และ r(x,y)

$$f(x,y) = i(x,y)r(x,y) \quad .....(2.2)$$

เมื่อ

$$0 < i(x,y) < \infty \quad .....(2.3)$$

$$0 < r(x,y) < 1 \quad .....(2.4)$$

สมการที่ 2.4 เป็นฟังชั่นการสะท้อนกลับของแสง ณ.จุด (x,y) มีค่าระหว่าง 0 (คือวัตถุดูดกลืนแสงทั้งหมด ไม่สะท้อนกลับเลย) กับ 1 (คือวัตถุสะท้อนแสงกลับทั้งหมด ไม่มีการดูดกลืนแสง) ส่วนฟังชั่น i(x,y) หาได้จากแหล่งกำเนิดแสง ส่วนฟังชั่น r(x,y) หาได้จากคุณสมบัติการดูดแสงของวัตถุ ในวันที่มีแสงแดดจ้า ดวงอาทิตย์จะให้ความสว่าง 9000 แรงเทียนส่องมายังผิวโลก ถ้าวันใดมีเมฆมากความสว่างจะลดลงเหลือ 1000 แรงเทียนโดยประมาณ ค่าของ r(x,y) จะมีค่า 0.01 สำหรับกำมะหยี่,0.65 สำหรับสเตนเลส,09 สำหรับเงินและ 0.93 สำหรับหิมะ

ความเข้มของแสงแบบโมโนโครม เรียกว่า ระดับเทา (gray level ; l) ของภาพณ.จุดนั้นๆ จะได้

$$L_{min} \leq l \leq L_{max}$$

โดย

$$L_{min} = i_{min}r_{min}$$

$$L_{max} = i_{max}r_{max}$$

โดยทั่วไป L_{min} จะมีค่าประมาณ 0.005 และ L_{max} มีค่าประมาณ 100 สำหรับใช้กับการประมวลผลภาพ

ช่วง [L_{min},L_{max}] เรียกว่า ระดับเทา เราอาจเขียนช่วงใหม่ให้เป็น [0,L] , เมื่อ l = 0 นั่นคือสีดำ และ l =L นั่นคือสีขาวตามสเกลของระดับเทา

2.6 พื้นฐานเกี่ยวกับความสัมพันธ์ระหว่างจุดภาพ

ในส่วนนี้จะแสดงให้เห็นถึงความสัมพันธ์ระหว่างจุดภาพ ในระบบการประมวลผลภาพดิจิตอล ฟังชั่นของข้อมูลภาพแสดงได้ด้วยฟังชั่น f(x,y) และจะแทนด้วยตัวอักษรตัวเล็กคือ p และ q ส่วนเซตย่อยของ จุดภาพของฟังชั่น f(x,y) จะแทนด้วยตัว S

2.6.1 สิ่งรอบข้างพิกเซล (neighbors of a pixel)

พิกเซล p ณ.โคออดิเนท (x,y) จะโดนล้อมรอบด้วยพิกเซลอื่นทั้งแถวนอนและแถวตั้ง นั่นคือ

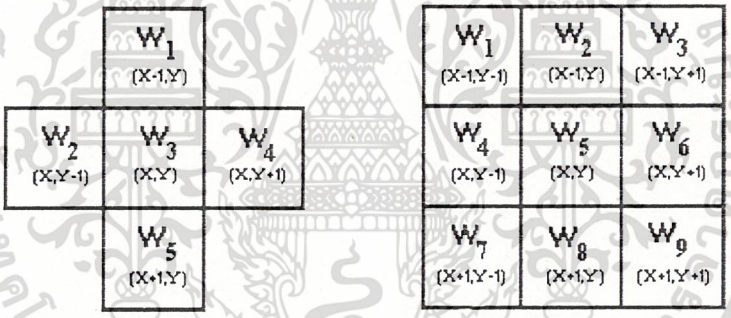
(x+1,y) , (x-1,y) , (x,y+1) , (x,y-1)

คือเซตของพิกเซล เรียกว่า การล้อมรอบทั้ง 4 ด้านของพิกเซล p (4-neighbors of p) ซึ่งแสดงได้โดย $N_4(p)$ นั่นคือจุดทั้ง 4 จะอยู่ห่างจากจุด (x,y) เป็นระยะ 1 หน่วย ในแนวทั้ง 4 จุด โดยจุดทั้ง 4 มีโคออดิเนทดังนี้

(x+1,y+1) , (x+1,y-1) , (x-1,y+1) , (x-1,y-1)

ในกรณีเราคิดจุดล้อมรอบ

ในแนวทแยงด้วยจะเห็นว่าจุดภาพหรือพิกเซลจะโดนล้อมรอบด้วยจุดภาพอื่นๆ 8 พิกเซลด้วยกัน จึงเรียกว่า 8-neighbors ของ p หรือ $N_8(p)$ ซึ่งแสดงได้ดังรูปที่ 2.11



a) 4-neighbors b) 8-neighbors

รูปที่ 2.11 แสดงจุดภาพที่โดนจุดภาพที่เราสนใจล้อมรอบ

2.6.2 ความต่อเนื่องของจุดภาพ (connectivity)

ความต่อเนื่องระหว่างพิกเซลเป็นหลักการสำคัญที่จะใช้หาขอบเขตของวัตถุและส่วนประกอบพื้นที่ของภาพ พิกเซล 2 พิกเซลต่อถึงกันได้แสดงว่าพิกเซลนั้นๆจะต้องอยู่ติดกัน ให้ V เป็นเซตของระดับเทาถ้าแต่ละความต่อเนื่องของพิกเซลมีความเข้ม 59,60,61 จะได้ V = { 59,60,61 } เราสามารถพิจารณาความต่อเนื่องได้ 3 วิธี

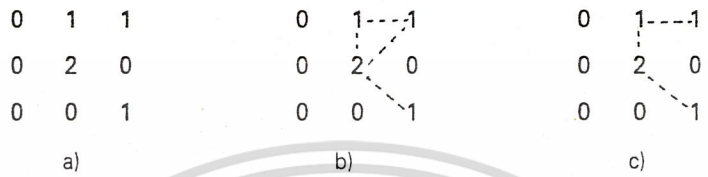
a) ความต่อเนื่อง 4 (4 - connectivity) พิกเซล p และ q ที่มีค่าจากเซต V ต่อถึงกัน 4 ตัว ถ้า q อยู่ใน $N_4(p)$

b) ความต่อเนื่อง 8 (8 - connectivity) พิกเซล p และ q ที่มีค่าจากเซต V ต่อถึงกัน 8 ตัว ถ้า q อยู่ใน $N_8(p)$

c) ความต่อเนื่องแบบผสม (mixed connectivity) พิกเซล p และ q ที่มีค่าจากเซต V จะต่อกันแบบผสม ถ้า

(i) q อยู่ใน $N_4(p)$ หรือ

(ii) q อยู่ใน $N_D(p)$ และไม่มีสมาชิกในเซต $N_4(p) \cap N_4(q)$



รูปที่ 2.12 แสดงลักษณะความต่อเนื่องของพิกเซล

a) การเรียงกันของพิกเซล

b) พิกเซลหมายเลข 2 มีสมาชิกล้อมรอบ 8 ตัว

c) พิกเซลแบบล้อมรอบผสมของพิกเซลชนิดเดียวกัน

2.6.3 **การวัดระยะระหว่างพิกเซล** (distance measure)

ให้พิกเซล p,q และ z อยู่ที่โคออดิเนท (x,y) , (s,t) และ (u,v) ตามลำดับ เรียก D ว่าฟังชั่นระยะ (distance function) หรือ เมตริกซ์ ถ้า

a) $D(p,q) \geq 0$ ($D(p,q) = 0$ ถ้า $p = q$),

b) $D(p,q) = D(q,p)$,

c) $D(p,z) \leq D(p,q) + D(q,z)$

ระยะยูคลิด (Euclidean) ระหว่าง p และ q คือ

$$D_e(p,q) = [\,(x-s)^2 + (y-t)^2\,]^{1/2}$$

สำหรับระยะที่วัดได้ของพิกเซลจะมีระยะน้อยกว่าหรือเท่ากับค่าของ r จาก จุด (x,y) หรือจุดที่อยู่บนดิสก์ของรัศมี r เมื่อให้จุด (x,y) เป็นจุดศูนย์กลาง

ระยะ D4 ระหว่าง p และ q ถูกกำหนดโดย

$D4(p,q) = |x-s| + |y-t|$

ในกรณีนี้ระยะห่าง D_4 เป็นระยะ (x,y) ซึ่งเป็นระยะที่น้อยกว่าหรือเท่ากับค่าของ r จากจุดศูนย์กลางของพิกเซลรูปเพชร ณ.จุด (x,y) จากตัวอย่าง พิกเซล D4 มีระยะน้อยกว่าเท่ากับ 2 จาก (x,y) หรือจุดศูนย์กลาง

```
        2
    2   1   2
2   1   0   1   2
    2   1   2
        2
```

จะเห็นว่าพิกเซลล้อมรอบ 4 ตัวของพิกเซลที่มีค่า 0 จะเป็น 1 หรือ $D_4 = 1$

สำหรับระยะ D_8 ระหว่างจุด p และ q ถูกกำหนดโดย

$D_8(p,q) = \max(x-s , y-t)$

ในกรณีนี้พิกเซลที่มีระยะ D_8 จากจุด (x,y) จะมีค่าน้อยกว่าหรือเท่ากับค่าของ r จากจุดศูนย์กลางของพิกเซลรูปสี่เหลี่ยมณ.จุด (x,y) จากตัวอย่างพิกเซล D_8 มีระยะน้อยกว่าเท่ากับ 2 จากจุด (x,y) หรือจุดศูนย์กลาง

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

ซึ่งจะเห็นว่าพิกเซลล้อมรอบ 8 ตัวของพิกเซลที่มีค่า 0 จะเป็น 1 หรือ $D_8 = 1$

2.7 <u>การสุ่มแบบต่อเนื่อง</u> (uniform sampling)

ในการทำการประมวลผลภาพฟังชั่นของข้อมูล f(x,y) จะต้องนำมาทำให้เป็นดิจิตอลทั้งสเปเทียลและแอมปลิจูด การทำ digitization ในโคออดิเนทสเปเทียลของจุด (x,y) จะหมายถึงการสุ่มข้อมูลภาพ (image sampling) ส่วนขนาดของการ digitization จะหมายถึงความสว่างของระดับเทา (gray - level quantization) สมมติให้ฟังชั่นของภาพต่อเนื่องจัดเรียงอยู่ในรูปของเมตริกซ์ NxN

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ & & & \\ & & & \\ f(N-1,0) & f(N-1,1) & \cdots & f(N-1,N-1) \end{bmatrix} \quad \text{.....(2.5)}$$

จะเห็นว่าด้านขวาของสมการของสมการเป็นภาพดิจิตอล (digital image) ในแต่ละจุดหรืออิลิเมนท์จะเรียกว่า พิกเซลหรือ เพล จากขบวนการ digitization เราจะต้องตัดสินใจเลือกขนาดของ N และจำนวนของระดับเทาว่าจะแยกได้กี่ระดับในแต่ละพิกเซล ได้ใช้สมการ

$$N = 2^n \quad \text{.....(2.6)}$$

และ

$$G = 2^m \quad \text{.....(2.7)}$$

เมื่อ G คือจำนวนของระดับเทาโดยแยกได้ตั้งแต่ 0 ถึง L จากสมการที่ 2.6 ปละ 2.7 เราสามารถคำนวณจำนวนบิทที่ใช้ในการเก็บภาพที่ผ่านการ digitization แล้ว

$$b = N \times N \times m \quad .....(2.8)$$

เมื่อ b คือจำนวนบิทที่ใช้ในการเก็บภาพที่ผ่านการdigitization

N\m	1	2	3	4	5	6	7	8
32	1024	2048	3072	4096	5120	6144	7168	8192
64	4096	8192	12288	16384	20480	24576	28672	32768
128	16384	32768	49152	65536	81920	98304	124688	131072
256	65536	131072	196608	262144	327680	393216	458752	524088
512	262144	524288	786432	1048576	1310720	1572864	1835008	2097152

ตารางที่ 2.1 จำนวนบิทที่ต้องใช้เมื่อมีการเปลี่ยนแปลงขนาดของ N และ m

N\m	1	2	3	4	5	6	7	8
32	128	256	512	512	1024	1024	1024	1024
64	512	1024	2048	2048	4096	4096	4096	4096
128	2048	4096	8192	8192	16384	16384	16384	16384
256	8192	16384	32768	32768	65536	65536	65536	65536
512	32768	65536	131072	131072	262144	262144	262144	262144

ตารางที่ 2.2 จำนวนของไบท์ (8 บิท) ที่ต้องใช้เมื่อมีการเปลี่ยนแปลงขนาดของ N และ m

2.8 <u>การแปลงภาพ</u> (image transform)

2.8.1 **ฟูเรียร์ทรานสฟอร์มเบื้องต้น** (concept of the fourier transform)

ให้ f(x) เป็นฟังชั่นต่อเนื่องของตัวแปรจริง x ฟูเรียร์ทรานสฟอร์มของฟังชั่น f(x) แสดงได้โดย $\Im\{f(x)$ จะถูกนิยามโดยสมการ

$$\Im\{f(x)\} = F(u) = \int_{-\infty}^{\infty} f(x)\exp[-j2\pi ux]\,dx \quad .....(2.9)$$

เมื่อ $j = -1$

ฟังชั่นที่ทำฟูเรียร์ทรานสฟอร์มไปแล้ว สามารถแปลงกลับมาเป็นฟังชั่นเดิมได้ โดยใช้อินเวอร์สฟูเรียร์ทรานสฟอร์ม (inverse fourier transform) โดย

$$\Im^{-1}\{F(u)\} = f(x)$$

$$= \int_{-\infty}^{\infty} F(u)\exp[-j2\pi ux]\,dx \quad .....(2.10)$$

จากสมการที่ 2.9 และ 2.10 จะเรียกว่า คู่สมการฟูเรียร์ทรานสฟอร์ม (fourier transform pair) ซึ่งจะทำการทรานสฟอร์มได้ถ้าฟังชั่น f(x) เป็นฟังชั่นแบบต่อเนื่องและสามารถทำการอินติเกรทได้ และ F(u) สามารถอินติเกรทได้ด้วยสภาวะเช่นนี้โดยมากจะเกิดในทางปฏิบัติ เมื่อเราให้ f(x) เป็นจริงเราจะได้ฟูเรียร์ทรานสฟอร์มเป็นจำนวนเชิงซ้อน (complex) นั่นคือ

$$F(u) = R(u) + jI(u) \quad .....(2.11)$$

เมื่อ R(u) และ I(u) เป็นส่วนจริง (real component) และส่วนจินตภาพ (imaginary component) ของ F(u) เราสามารถเปลี่ยนสมการที่ 2.11 ให้อยู่ในฟอร์มของเอ็กซ์โปเนนเชี่ยล (exponential form) ได้

$$F(u) = |F(u)|\, e^{j\phi(u)} \quad .....(2.12)$$

เมื่อ

$$|F(u)| = \left[R^2(u) + I^2(u) \right]^{1/2} \quad .....(2.13)$$

และ

$$\phi(u) = \tan^{-1}\left[\frac{I(u)}{R(u)}\right] \quad .....(2.14)$$

แมกนิจูด (magnitude) ของฟังชั่น f(x) เรียกว่า ฟูเรียร์ สเปกตรัม (fourier spectrum) ของ f(x) และ $\phi(u)$ คือ มุมเฟส (phase angle)

สเปกตรัมกำลัง

$$P(u) = F(u)^2$$
$$= R^2(u) + I^2(u) \quad .....(2.15)$$

P(u) คือสเปกตรัมกำลัง (power spectrum) ของฟังชั่น f(x)

ตัวแปร u ที่ใช้ในฟูเรียร์ทรานสฟอร์มเรียกว่าตัวแปรความถี่ (frequency variable) เมื่อใช้สูตรของออยเลอร์โนเทอมของเอ็กซ์โปเนเชียล จะเปลี่ยนเป็น

$$\exp(-j2\pi ux) = \cos 2\pi ux - j\sin 2\pi ux \quad .....(2.16)$$

ถ้าเราอินติเกรทสมการที่ 2.9 จะได้ว่า F(u) ประกอบด้วยผลบวกอนันต์ของ sin และ cos และค่าแต่ละค่าของ u จะเป็นความถี่ของการตอบสนองของคู่ sin-cosin

รูปที่ 2.13 แสดงการทำฟูเรียร์ทรานสฟอร์มของฟังชั่น 1 มิติ f(x)

ฟูเรียร์ทรานสฟอร์มสามารถขยายไปยังฟังชั่น f(x,y) ซึ่งเป็นฟังชั่น 2 มิติได้ ถ้า f(x,y) เป็นฟังชั่นแบบต่อเนื่องและสามารถอินติเกรทได้ และ F(u) สามารถอินติเกรทได้ซึ่งจะได้ฟูเรียร์ทรานสฟอร์มของ f(x,y) คือ

$$\Im\{f(x,y)\} = F(u,v)$$
$$= \int_{-\infty}^{\infty}\int f(x,y)\exp[-j2\pi(ux+vy)]dxdy \quad .....(2.17)$$

และอินเวอร์สฟูเรียร์ทรานสฟอร์มคือ

$$\Im^{-1}\{f(u,v)\} = F(x,y)$$
$$= \int_{-\infty}^{\infty}\int f(u,v)\exp[j2\pi(ux+vy)]dudv \quad .....(2.18)$$

เมื่อ u และ v เป็นตัวแปรคงที่

เช่นเดียวกันกับฟูเรียร์ทรานสฟอร์ม 1 มิติ เราจะได้

ฟูเรียร์ สเปกตรัม

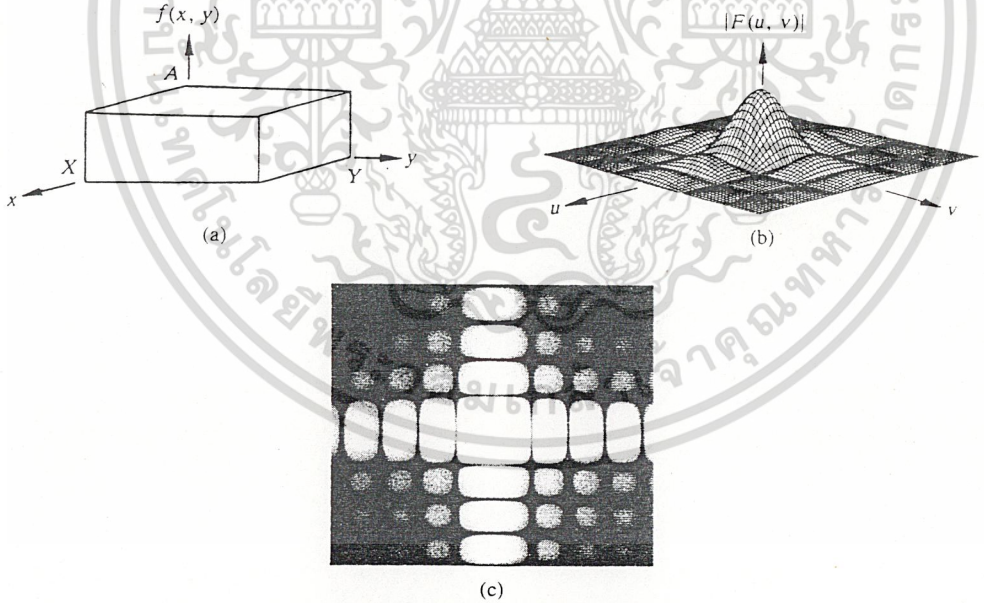
$$F(u,v) = \left[R^2(u,v) + I^2(u,v)\right]^{1/2} \quad .....(2.19)$$

มุมเฟส

$$\Phi(u,v) = \tan^{-1}\left[\frac{I(u,v)}{R(u,v)}\right] \quad .....(2.20)$$

และ สเปกตรัมกำลัง

$$P(u,v) = F)u,v)^2 = R^2(u,v) + I^2(u,v) \quad .....(2.21)$$



a) ฟังชั่น f(x,y)

b) ฟูเรียร์ สเปกตรัม

c) สเปกตรัมที่แสดงอยู่ในรูปของฟังชั่นความเข้มของแสง

รูปที่ 2.14 แสดงการทำฟูเรียร์ทรานสฟอร์มของฟังชั่น 2 มิติ

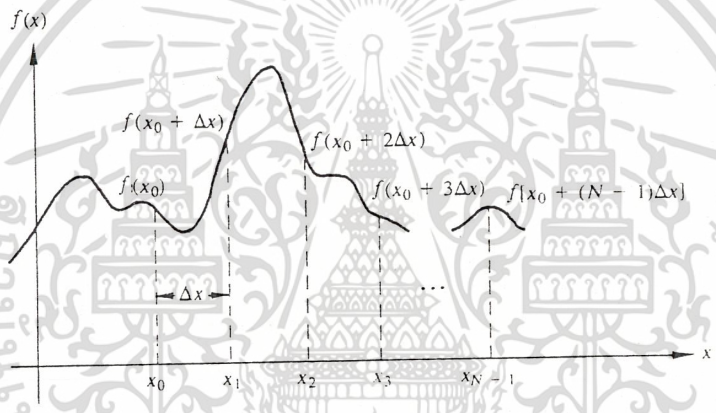
2.8.2 ฟูเรียร์ทรานสฟอร์มแบบไม่ต่อเนื่อง (discrete fourier transform)

สมมติให้ฟังชั่น f(x,y) เป็นฟังชั่นต่อเนื่องที่ถูกสุ่มออกเป็นลำดับ { $f(x_0)$,$f(x_0+\Delta x)$, $f(x_0+2\Delta x)$,....$f(x_0+[N-1]\Delta x)$ } โดย N เป็นจำนวนตัวอย่างในการสุ่มและสามารถเขียนฟังชั่นได้

$$f(x) = f(x_0 + x\Delta x) \quad .....(2.22)$$

เมื่อ x คือ 0,1,2,...,N-1 และลำดับ { f(0),f(1),f(2),...,f(N-1) } จะได้กำหนดช่องกว้างที่เท่ากันแต่นำมาแบ่งออกจากฟังชั่นต่อเนื่องคือ

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\exp[-j2\pi ux/N] \quad .....(2.23)$$



รูปที่ 2.15 แสดงฟังชั่นต่อเนื่องที่ถูกสุ่มในช่วงกว้างเท่าๆกัน

เมื่อ u = 0,1,2,...,N-1 และ

$$f(x) = \sum_{u=0}^{N-1} F(u)\exp[j2\pi ux/N] \quad .....(2.24)$$

เมื่อ x = 0,1,2,...,N-1

ค่าของ u = 0,1,2,...,N-1 ในสมการที่ 2.24 จะสมนัยกับค่าที่สุ่มมา ณ.จุด 0,Δu,2Δu,...,(N-1)Δu ซึ่ง u และ x จะสัมพันธ์กัน

$$\Delta u = \frac{1}{N\Delta x} \quad .....(2.25)$$

ในกรณีเป็นฟังชั่น 2 ตัวแปร สมการคู่ของฟูเรียร์ทรานสฟอร์มแบบไม่ต่อเนื่องจะกำหนดได้โดย

$$F(u,v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp[-j2\pi(ux/M + vy/N)] \quad .....(2.26)$$

เมื่อ u = 0,1,2,...,M-1 , v = 0,1,2,...,N-1 และ

$$F(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} f(u,v)\exp[j2\pi(ux/M + vy/N)] \quad .....(2.27)$$

เมื่อ x = 0,1,2,...,M-1 , y = 0,1,2,...,N-1

การสุ่มของฟังชั่นในตอนนี้จะเป็นแบบ 2 มิติ โดยจะถูกแบ่งเป็น Δx และ Δy ตามแนวแกน x และแกน y ส่วนฟังชั่น 1 มิติ ฟังชั่นไม่ต่อเนื่อง f(x,y) จะถูกสุ่มออกเป็นฟังชั่น f(x0,xΔx,yΔy) สำหรับ x = 0,1,2,...,M-1 และ y = 0,1,2,...,N-1 เราจะได้ความสัมพันธ์ของพิกัดความถี่ (frequency domain) และพิกัดภาพ (spatial domain) ดังนี้

$$\Delta u = \frac{1}{M\Delta x} \quad .....(2.28)$$

และ

$$\Delta v = \frac{1}{N\Delta x} \quad .....(2.29)$$

เมื่อภาพโดนสุ่มเป็นรูปสี่เหลี่ยมจัตุรัสเราจะได้ M = N

$$F(u,v) = \frac{1}{N}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\exp[-j2\pi(ux + vy)/N] \quad .....(2.30)$$

เมื่อ u,v =0,1,2,...,N-1

$$F(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} f(u,v)\exp[j2\pi(ux + vy)/N] \quad .....(2.31)$$

เมื่อ x,y = 0,1,2,...,N-1

สเปกตรัมฟูเรียร์,มุมเฟสและพลังงานของสเปกตรัมของฟังชั่น 1 และ 2 มิติแบบไม่ต่อเนื่อง หาได้จากสมการที่ 2.13 ถึง 2.15 และสมการที่ 2.19 ถึง 2.21 ตามลำดับเพียงแต่ตัวแปรอิสระจะโดนสุ่ม

2.8.3 คุณสมบัติของฟูเรียร์ทรานสฟอร์ม 2 มิติ

2.8.3.1 **การถ่ายค่าและสเกล** (distribution and scaling)

จากนิยามของการทรานสฟอร์มแบบต่อเนื่องและการทรานสฟอร์มแบบไม่ต่อเนื่อง

$$\Im\{f_1(x,y) + f_2(x,y)\} = \Im\{f_1(x,y)\} + \{f_2(x,y)\} \quad .....(2.32)$$

และโดยทั่วไป

$$\Im\{f_1(x,y) \cdot f_2(x,y)\} \neq \Im\{f_1(x,y)\} \cdot \{f_2(x,y)\} \quad .....(2.33)$$

ฟูเรียร์ทรานสฟอร์มของผลบวกของฟังชั่น จะเท่ากับผลบวกของฟูเรียร์ทรานสฟอร์มของแต่ละฟังชั่น ส่วนปริมาณสเกลาของ a และ b

$$af(x,y) \Leftrightarrow aF(u,v) \quad .....(2.34)$$

$$f(ax,by) \Leftrightarrow \frac{1}{ab} F(u/a, v/b) \quad .....(2.35)$$

2.8.3.2 **ค่าเฉลี่ย** (average value)

การหาค่าเฉลี่ยของฟังชั่นแบบไม่ต่อเนื่อง 2 มิติ จะหาได้โดย

$$\bar{f}(x,y) = \frac{1}{N^2} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \quad .....(2.36)$$

แทนค่าของ u = v = 0 ในสมการที่ 2.30 จะได้

$$F(0,0) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \quad .....(2.37)$$

ดังนั้นเราจะเห็น $\bar{f}(x,y)$ สัมพันธ์กับฟูเรียร์ทรานสฟอร์มของ f(x,y) ดังสมการ

$$\bar{f}(x,y) = \frac{1}{N} F(0,0) \quad .....(2.38)$$

2.8.3.3 **ลาปลาซ** (laplacian)

ลาปลาซของฟังชั่น 2 ตัวแปร f(x,y) กำหนดโดย

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad .....(2.39)$$

และจะได้ฟูเรียร์ทรานสฟอร์ม 2 มิติ

$$\Im\{\nabla^2 f(x,y)\} \Leftrightarrow -(2\pi)^2(u^2 + v^2)F(u,v) \quad .....(2.40)$$

การแปลงลาปลาซจะใช้มากเกี่ยวกับเรื่องการแปลงภาพ

2.8.3.4 **การคอนโวลูชั่น** (convolution)

การคอนโวลูชั่นของฟังชั่น 2 ฟังชั่น f(x) และ g(x) คือ f(x)∗g(x) จะถูกกำหนดโดย

$$f(x)*g(x) = \int_{-\infty}^{\infty} f(\alpha)g(x-\alpha)d\alpha \quad .....(2.41)$$

เมื่อ α เป็นตัวแปรหุ่นของการอินติเกรท จากตัวอย่างจะทำการคอนโวลูชั่นระหว่างฟังชั่น f(x) และ g(x) ดังรูปที่ 2.16 (a) และ 2.16 (b) ตามลำดับก่อนที่จะทำการอินติเกรทจะต้องทำฟังชั่นให้อยู่ในรูปของ g(x-α) ก่อน ตามขั้นตอนในรูปที่ 2.16 (c) และ 2.16 (d) จากนั้นคูณฟังชั่น g(-α) ด้วย g(x-α) แล้วอินติกรัล ผลคูณจาก ∞ ถึง - ∞

ผลคูณของ f(α) และ g(x-α) คือส่วนที่แรเงาในรูปที่ 2.16 (e) เมื่อให้ $0 \le x \le$ เมื่อผลคูณเป็นศูนย์สำหรับค่า α ที่อยู่นอกช่วง [0,x] เราจะพบว่า $f(x)*g(x) = x/2$ สำหรับในรูปที่ 2.16 (f) ค่า x จะอยู่ในช่วง [1,2] ในกรณีนี้จะได้ $f(x)*g(x) = (1-x/2)$ ดังนั้นแสดงว่า f(α)g(x-α) จะเป็นศูนย์สำหรับค่า x ที่อยู่นอกช่วง [0,2] จะได้

$$f(x)*g(x) = \begin{cases} x/2 & 0 \le x \le 1 \\ 1-x/2 & 1 \le x \le 2 \\ 0 & \text{otherwise} \end{cases}$$

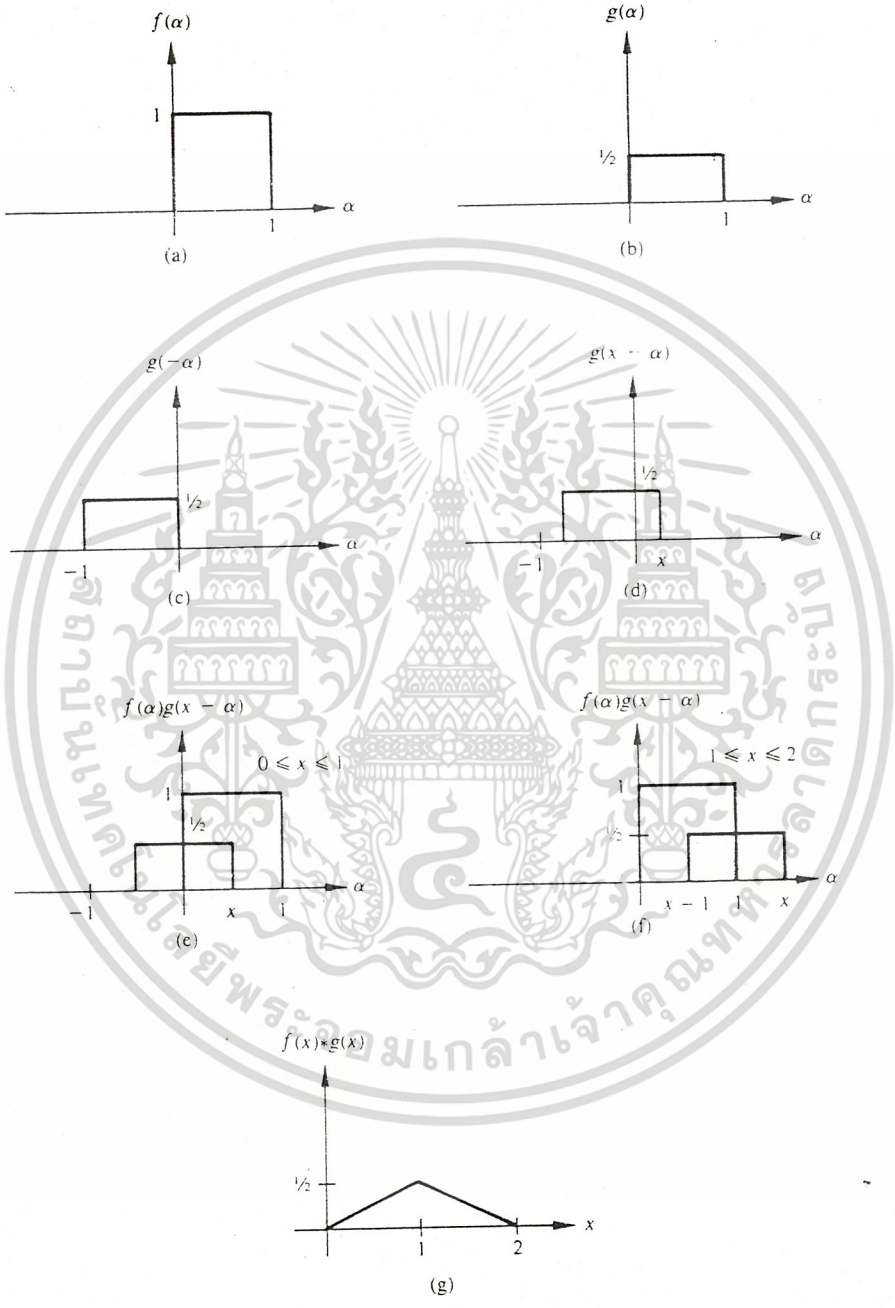
ผลที่ได้จะอยู่ในรูปที่ 2.16 (g)

การคอนโวลูชั่นของฟังชั่น f(x) ที่เป็นฟังชั่นอิมพัลซ์ (impulse function) $\delta(x-x_0)$ จะนิยามโดยความสัมพันธ์

$$\int_{-\infty}^{\infty} f(x)\delta(x-x_0)dx = f(x_0) \quad .....(2.42)$$

ฟังชั่น $\delta(x-x_0)$ จะมีพื้นที่เท่ากับ 1

$$\int_{-\infty}^{\infty} \delta(x-x_0)dx = \int_{x_0^-}^{x_0^+} \delta(x-x_0)dx = 1 \quad .....(2.43)$$



รูปที่ 2.16 กราฟแสดงการทำคอนโวลูชั่น ส่วนที่แรเงาแสดงถึงพื้นที่ที่ผลคูณไม่เท่ากับหนึ่ง

แบบ 1 มิติคือ

$$f(x)*g(x) \Leftrightarrow F(u)G(u) \quad .....(2.44)$$

และ

$$f(x)g(x) \Leftrightarrow F(u)*G(u) \quad .....(2.45)$$

เมื่อ F(u) คือ ฟูเรียร์ทรานสฟอร์มของ f(x)

G(u) คือ ฟูเรียร์ทรานสฟอร์มของ g(x)

ถ้าฟังชั่นเป็นแบบ 2 มิติจะได้กฎของการทำคอนโวลูชั่นดังนี้

$$f(x,y)*g(x,y) \Leftrightarrow F(u,v)G(u,v) \quad .....(2.46)$$

และ

$$f(x,y)g(x,y) \Leftrightarrow F(u,v)*G(u,v) \quad .....(2.47)$$

2.9 <u>การทำภาพให้ชัดเจนขึ้น</u> (image enhancement)

การทำให้ชัดเจนขึ้นมีวัตถุประสงค์ก็เพื่อต้องการให้ภาพที่จะนำมาประมวลผล มีผลลัพธ์ที่เหมาะสมกับงานที่จะนำไปประยุกต์ และการใช้เทคนิคใดเทคนิคหนึ่งเพื่อทำให้ภาพคมชัดขึ้นก็จะเหมาะกับงานหนึ่งๆ เท่านั้น การพิจารณาจะแบ่งเป็น 2 ลักษณะคือ วิธีในโดเมนเชิงความถี่ (frequency-domain method) และวิธีในโดเมนสเปเทียล (spatial-domain method) เทคนิคอันแรกจะใช้เทคนิคของการแปลงฟูเรียร์ (fourier transform) ของภาพ ส่วนเทคนิคอย่างที่สองจะใช้การอ้างถึงระนาบภาพของตัวมันเองและการปฏิบัติการโดยตรงกับพิกเซลของภาพ

2.9.1 **วิธีในโดเมนสเปเทียล** (spatial-domain method)

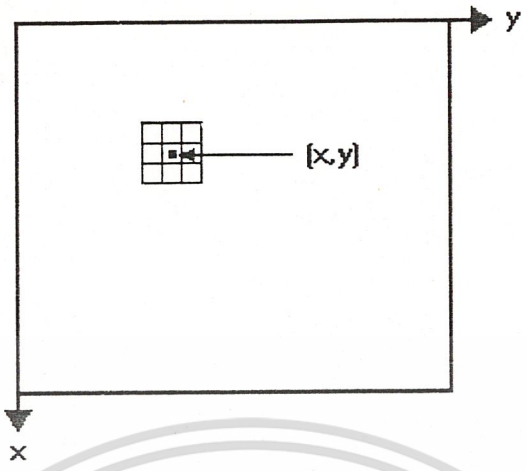
คำว่า โดเมนสเปเทียล หมายถึงการรวมกันของพิกเซลของภาพและวิธีในโดเมนสเปเทียลก็คือการปฏิบัติการโดยตรงกับพิกเซล ฟังชั่นของการประมวลผลข้อมูลภาพในโดเมนสเปเทียลจะเขียนได้คือ

$$g(x,y) = T[f(x,y)] \quad .....(2.48)$$

เมื่อ f(x,y) คือฟังชั่นของภาพอินพุท

g(x,y) คือภาพที่ผ่านการประมวลผลแล้ว

T คือตัวปฏิบัติการที่กระทำกับฟังชั่น f(x,y) ซึ่งจะปฏิบัติ การในย่านขอบเขตของ (x,y)

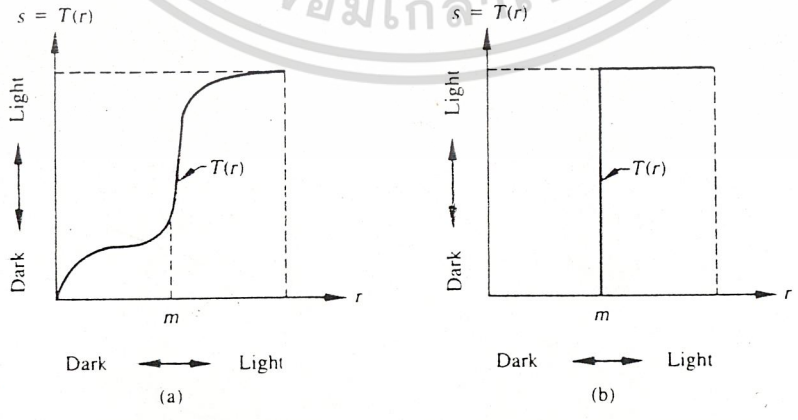


รูปที่ 2.17 ภาพย่อยขนาด 3x3 พิกเซลที่มีจุด (x,y) เป็นจุดศูนย์กลาง

ศูนย์กลางอยู่ที่ (x,y) ซึ่งแสดงดังรูปที่ 2.17 จุดศูนย์กลางอยู่ที่ของภาพย่อยจะเลื่อนจากพิกเซลหนึ่งไปยังพิกเซลหนึ่ง หลักในการกำหนดขอบเขตของ (x,y) โดยพิจารณาภาพย่อยที่เป็นสี่เหลี่ยมจัตุรัสหรือสี่เหลี่ยมผืนผ้าที่มีเซลหนึ่ง โดยเริ่มพิกเซลแรกสุดที่มุมบนซ้ายแล้วกระทำกับจุด (x,y) แต่ละจุดซึ่งจะได้ค่าของ g ณ.ตำแหน่ง (x,y) นั้นๆ ส่วนรูปฟอร์มของ T เมื่อภาพมีขนาด 1x1 ในกรณีนี้ค่า g จะขึ้นอยู่กับค่าของฟังชั่น f ที่จุด (x,y) และ T จะกลายเป็นฟังชั่นของระดับเทา (gray level) นั่นคือ

$$s = T(r) \quad .....(2.49)$$

เมื่อ r และ s เป็นตัวแปรที่ใช้แทนระดับเทาของ f(x,y) และ g(x,y) ที่จุด (x,y) T(r) แสดงดังรูป 2.18 (a) ผลของการแปลงจะทำให้ระดับความสว่างของภาพสูงกว่าแบ่งออกเป็น 2 ส่วน คือส่วนที่สว่าง (light) และส่วนที่มืด (dark) โดยถูกแบ่งที่จุด m ส่วนที่ค่า r อยู่เหนือจุด m จะสว่าง และส่วนที่ค่า r ต่ำกว่าจุด m จะมืด เทคนิคอันนี้จะเรียกว่า การแบ่งความแตกต่างของแสง (contrast stretching) ซึ่งจะเห็นดังรูป 2.18 (b) โดย T(r) จะให้ภาพออกมาเป็น 2 ระดับ (binary image)



รูปที่ 2.18 ฟังชั่นการแปลงระดับเทาสำหรับทำให้ภาพเป็น 2 ระดับ

จากรูป 2.19 ถ้าเราให้ w1,w2,...,w9 เป็นสัมประสิทธิ์ของมาส์ก (mask) และพิจารณาย่านรอบๆจุด (x,y) 8 จุด

W_1 (X-1,Y-1)	W_2 (X-1,Y)	W_3 (X-1,Y+1)
W_4 (X,Y-1)	W_5 (X,Y)	W_6 (X,Y+1)
W_7 (X+1,Y-1)	W_8 (X+1,Y)	W_9 (X+1,Y+1)

รูปที่ 2.19 รูปทั่วไปของมาส์ก 3x3 ที่จะไปกระทำกับพิกเซลของภาพตามจุดต่างๆจะทำการปฏิบัติการได้

$$T[f(x,y)] = w1f(x-1,y-1) + w2f(x-1,y) + w3f(x-1,y+1) + w4f(x,y-1) + w5f(x,y) + w6f(x,y+1) + w7f(x+1,y-1) + w8f(x+1,y) + w9f(x+1,y+1) \quad .....(2.50)$$

ซึ่งถ้าเป็นมาส์กขนาดใหญ่กว่า รูปแบบของฟังชั่นก็จะมีลักษณะคล้ายกับมาส์กแบบ 3x3 สมการที่ 2.50 จะเปลี่ยนสัมประสิทธิ์ฟังชั่นของมาส์ก

2.9.2 วิธีในโดเมนเชิงความถี่ (frequency-domain method)

ทฤษฎีการคอนโวลูชั่นเป็นทฤษฎีที่นำมาใช้กับวิธีในโดเมนเชิงความถี่ ให้ g(x,y) คือฟังชั่นที่ทำการคอนโวลูชั่นโดยฟังชั่นของ f(x,y) และตัวปฏิบัติการที่ไม่เปลี่ยนตำแหน่ง (position-invarent operator) นั่นคือ

$$g(x,y) = h(x,y) * f(x,y) \quad .....(2.51)$$

จากทฤษฎีของการคอนโวลูชั่นจะได้โดเมนเชิงความถี่ในรูปของความสัมพันธ์

$$G(u,v) = H(u,v)f(u,v) \quad .....(2.52)$$

เมื่อ G,H และ F คือฟูเรียร์ทรานสฟอร์มของ g,h และ f ตามลำดับ ตัวแปลง (transform) บางครั้งเรียกว่าทรานสเฟอร์ฟังชั่น (transfer function) ของการประมวลผล ดังนั้น

$$g(x,y) = \mathfrak{I}^{-1}[H(u,v)F(u,v)] \quad .....(2.53)$$

2.9.3 การทำให้ภาพมีระดับที่ราบเรียบ (image smooting)

2.9.3.1 การหาค่าเฉลี่ยรอบย่าน (neighborhood average)

การเฉลี่ยค่ารอบย่านเป็นเทคนิคโดยตรงที่ใช้ในโดเมนสเปเทียล เพื่อทำให้สัญญาณข้อมูลภาพเกิดความราบเรียบ ให้ภาพ f(x,y) มีขนาด NxN และ g(x,y) คือผลของการกระทำให้ภาพราบเรียบที่ระดับเทาของพิกเซล (x,y) ซึ่งจะได้ความสัมพันธ์

$$g(x,y) = \frac{1}{M}\sum_{(n,m)\in s} f(n,m) \quad .....(2.54)$$

เมื่อ x,y = 0,1,...,N-1

s คือ เซตของโคออดิเนทของจุดที่อยู่รอบๆจุด (x,y) รวมทั้งจุด (x,y) ด้วย

M คือ ผลรวมของจำนวนของจุดในรอบๆย่านนั้น

2.9.3.2 ฟิลเตอร์แบบมัธยฐาน (median filtering)

ภาพที่เกิดความเบลอร์ที่ขอบภาพและมีสัญญาณรบกวนอาจแก้ไขได้โดยการตัดค่าเทรสโฮล (threshold) ซึ่งโดยทั่วไปจะใช้วิธีการลองผิดลองถูก (trial and error) แต่ถ้าเราใช้ฟิลเตอร์แบบมัธยฐานแทนระดับเทาของแต่ละพิกเซลที่อยู่รอบๆย่านแทนการเฉลี่ย จะทำให้ภาพมีประสิทธิภาพดียิ่งขึ้น การหาค่ามัธยฐาน (median;M) คือเซตของค่าที่มีค่าในเซ็ตครึ่งหนึ่งมากกว่า M และอีกครึ่งหนึ่งน้อยกว่า M สามารถทำได้โดยเรียงข้อมูลจากน้อยไปหามาก แล้วนำค่าที่อยู่ตรงกลางมาใช้แทนค่า M สมมติให้ย่านขนาด 3x3 มีข้อมูล (10,20,20,20,15,20,20,25,100)จะสามารถเรียงข้อมูลจากน้อยไปหามากได้เป็น (10,15,20,20,20,20,20 ,25,100) จะได้ค่ามัธยฐานคือ 20 นั่นเอง

2.9.3.3 ฟิลเตอร์แบบความถี่ต่ำ (lowpass filtering)

ขอบภาพและส่วนที่คมของภาพนั้นในระดับเทาของภาพจะมีส่วนของสัญญาณความถี่สูงของการแปลงฟูเรียร์อยู่ ซึ่งจะทำให้ภาพเบลอร์ ดังนั้นจึงต้องกำจัดช่วงความถี่สูงที่ไม่ต้องการออกไป จากโดเมนเชิงความถี่

$$G(u,v) = H(u,v)F(u,v) \quad .....(2.55)$$

เมื่อ F(u,v) คือภาพที่ต้องการทำให้เรียบโดยการแปลงฟูเรียร์ ปัญหาอยู่ที่การเลือกฟังชั่น H(u,v) ให้ผลที่ได้ออกมาเป็น G(u,v) โดยการลดทอนส่วนประกอบของความถี่สูงของ F(u,v) ออกไปการแปลงอินเวอร์สฟูเรียร์ของ G(u,v) จะได้ g(x,y) ที่มีความราบเรียบขึ้นทั้งนี้ก็เพราะส่วนประกอบของความถี่สูงจะโดนกรองออกไปและข้อมูลที่ผ่านไปได้จะเป็นช่วงของสัญญาณความถี่ต่ำ ซึ่งความถี่ต่ำที่ผ่านไปได้จะไม่โดนลดทอนลง วิธีนี้จึงเรียกว่าการกรองความถี่ต่ำ ฟังชั่น H(u,v) หมายถึงฟังชั่นการส่งถ่ายค่าของการกรอง (filter transfer function)

2.9.3.4 ฟิลเตอร์แบบอุดมคติ (ideal filter)

ฟิลเตอร์ 2 มิติ แบบอุดมคติที่จะมีการส่งถ่ายฟังชั่น โดยมีความสัมพันธ์

$$H(u,v) = \begin{cases} 1 & D(u,v) \le D_0 \\ 0 & D(u,v) > D_0 \end{cases} \quad .....(2.56)$$

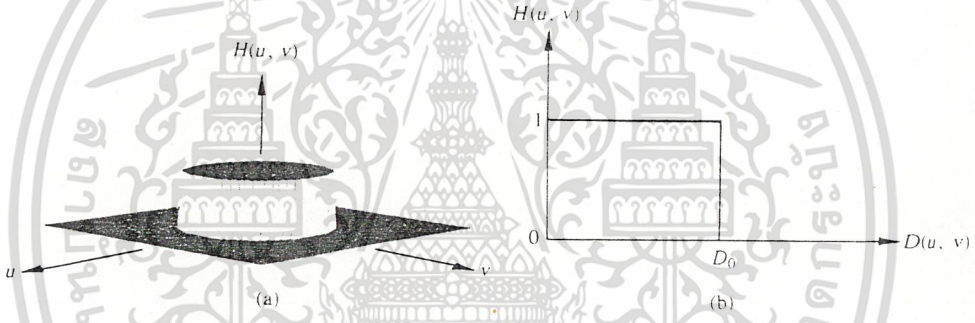
เมื่อ D_0 คือค่าจำเพาะที่ไม่เป็นลบ

D(u,v) คือระยะทางจากจุด (u,v) ถึงจุดกำเนิดในระนาบความถี่

นั่นคือ

$$D(u,v) = (u^2 + v^2)^{1/2} \quad .....(2.57)$$

สำหรับฟังชั่น 3 มิติ จะสามารถพล็อตค่าของ H(u,v) เทียบกับ u และ v แสดงดังรูป 2.20 (a) ตามอุดมคติความถี่ที่โดนกรองจะอยู่ในรัศมี D_0 ของวงกลมซึ่งความถี่ในช่วงนี้จะผ่านไปได้โดยไม่โดนลดทอน ส่วนความถี่ที่อยู่นอกช่วงรัศมีของวงกลมจะโดนลดทอนหรือโดนตัดออก



(a) ฟังชั่น 3 มิติที่นำมากรองความถี่

(b) รูปหน้าตัด

รูปที่ 2.20 ฟิลเตอร์แบบอุดมคติกรองความถี่ต่ำ

2.9.4 การทำให้ภาพมีความคม (image sharpening)

2.9.4.1 การทำภาพให้มีความคมโดยใช้ดิฟเฟอเรนเชียล (sharpening by differentiation)

โดยทั่วไปการใช้วิธีดิฟเฟอเรนเชียลในการประมวลผลภาพจะใช้การทำเกรเดียนท์ (gradient) ให้ภาพ คือฟังชั่น f(x,y) เกรเดียนท์ของฟังชั่น f ที่โคออดิเนท (x,y) จะนิยามโดยเวคเตอร์

$$G[f(x,y)] = \begin{bmatrix} \partial f \\ \partial x \\ \partial f \\ \partial y \end{bmatrix} \quad .....(2.58)$$

คุณสมบัติของการเกรเดียนท์คือ

1. เวคเตอร์ $\mathbf{G}[f(x,y)]$ จะเป็นทิศทางของอัตราการเพิ่มสูงสุดของฟังชั่น f(x,y)
2. ขนาดของเวคเตอร์ $\mathbf{G}[f(x,y)]$ จะถูกกำหนดโดย G[f(x,y)] และจะได้

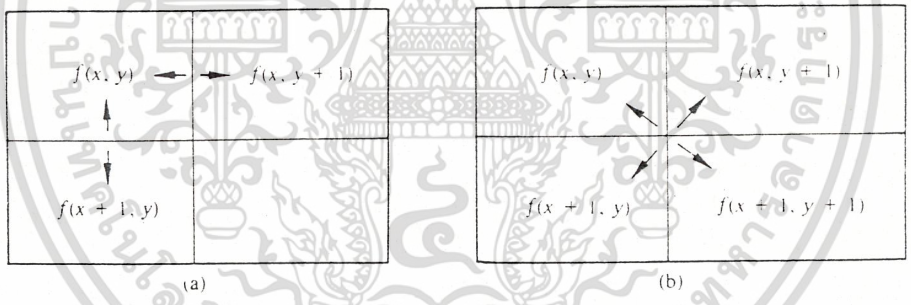
$$\mathbf{G}[f(x,y)] = \text{mag}[\mathbf{G}] = [(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2]^{1/2} \quad .....(2.59)$$

เท่ากับอัตราการเพิ่มสูงสุดของฟังชั่น f(x,y) ต่อระยะทาง 1 หน่วยในทิศทางของ G ในการประมวลผลภาพอนุพันธ์ของสมการ 2.59 จะประมาณโดย

$$\mathbf{G}[f(x,y)] \cong \{[f(x,y)-f(x+1,y)]^2 + [f(x,y)-f(x,y+1)]^2\}^{1/2} \quad .....(2.60)$$

ผลที่ได้จะเป็นค่าสัมบูรณ์ นั่นคือ

$$\mathbf{G}[f(x,y)] \cong f(x,y)-f(x+1,y + f(x,y)-f(x,y+1) \quad .....(2.61)$$



รูปที่ 2.21 แสดงการทำเกรเดียนท์ของฟังชั่น 2 มิติ

ความสัมพันธ์ระหว่างพิกเซลของสมการ 2.60 และ 2.61 แสดงดังรูปที่ 2.21 (a) ซึ่งให้ภาพมีขนาด NxN เราจะไม่สามารถทำเกรเดียนท์ที่แถวสุดท้าย (x=N) หรือคอลัมน์สุดท้าย (y=N) การใช้งานอื่นๆโดยประมาณบางครั้งเรียกว่า Roberts gradient ใช้การดิฟเฟอเรนเชียลทแยง (cross differential) แสดงดังรูป 2.21 (b) ซึ่งจะได้ความสัมพันธ์

$$\mathbf{G}[f(x,y)] \cong \{[f(x,y)-f(x+1,y+1)]^2 + [f(x+1,y)-f(x,y+1)]^2\}^{1/2} \quad .....(2.62)$$

หรือใช้ค่าสัมบูรณ์

$$G[f(x,y)] \cong f(x,y)-f(x+1,y+1 + f(x+1,y)-f(x,y+1) \quad .....(2.63)$$

2.9.4.2 ฟิลเตอร์แบบกรองความถี่สูง (highpass filtering)

จากการทำการกรองแบบความถี่ต่ำอาจทำให้ภาพเบลอร์ (blurred) ได้อันเนื่องมาจากเราลดทอนส่วนประกอบของความถี่สูงของการแปลงฟูเรียร์ซึ่งจะทำให้ขอบภาพเกิดการเปลี่ยนแปลงของระดับเทา การทำให้ภาพมีความคมในโดเมนเชิงความถี่โดยวิธีการกรองแบบความถี่สูงจะทำการลดทอนส่วนประกอบของสัญญาณความถี่ต่ำ โดยปราศจากการกระทบกระเทือนต่อสัญญาณความถี่สูง ในการแปลงฟูเรียร์เราจะพิจารณาส่วนที่ต่ำกว่าความถี่สูง ก่อนจะทำการพิจารณาเราจะทำการเลื่อนเฟสศูนย์ (zero-phase shift) เพื่อให้เกิดการสมมาตรและทำให้สมบูรณ์ได้โดยหน้าตัดจะขยายฟังชั่นของระยะทางจากจุดกำเนิดของการแปลงฟูเรียร์

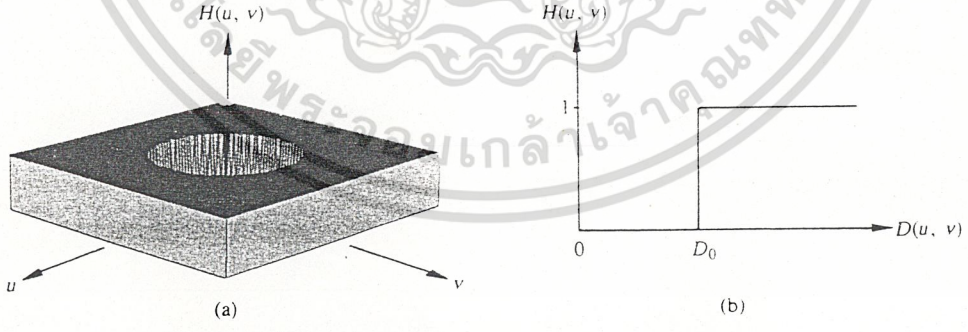
2.9.4.3 ฟิลเตอร์แบบอุดมคติ (ideal filter)

ฟังชั่น 2 มิติของการกรองแบบอุดมคติจะมีการส่งถ่ายค่าตามความสัมพันธ์

$$H(u,v) = \begin{cases} 0 & D(u,v) \le D_0 \\ 1 & D(u,v) > D_0 \end{cases} \quad .....(2.64)$$

เมื่อ D_0 คือระยะที่ตัดโดยวัดจากจุดกำเนิดของระนาบความถี่

โดยแสดงดังรูปที่ 2.22 การกรองแบบความถี่สูงจะมีคุณสมบัติที่ตรงกันข้ามกับการกรองแบบความถี่ต่ำโดยความถี่ที่อยู่นอกรัศมีของวงกลม D_0 จะผ่านไปได้โดยไม่ลดทอน ส่วนความถี่ที่อยู่ในช่วงของวงกลมจะผ่านไปไม่ได้

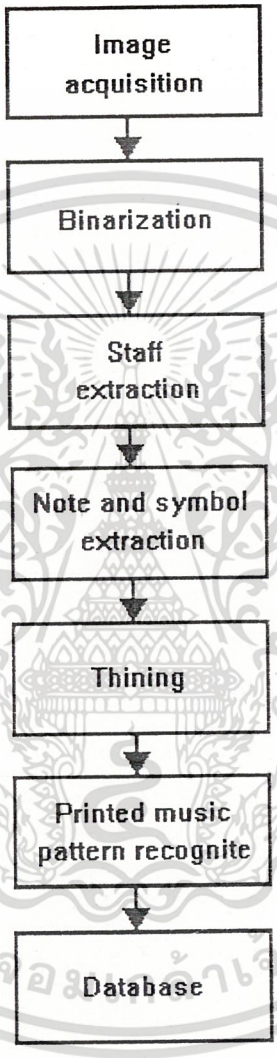


(a) ฟังชั่น 3 มิติที่นำมากรองความถี่สูง (b) รูปหน้าตัด

รูปที่ 2.22 ฟิลเตอร์แบบอุดมคติกรองความถี่สูง

บทที่ 3
การออกแบบและอัลกอริทึมในการจดจำรูปแบบ

ลำดับขั้นการทำโครงงานเป็นไปตามลำดับดังนี้



รูปที่ 3.1 ขั้นตอนการทำการจดจำรูปแบบตัวพิมพ์โน้ต

3.1 ขั้นตอนการทำการจดจำรูปแบบตัวพิมพ์โน้ต

ขั้นตอนที่ 1 การเก็บภาพ

การเก็บภาพจะเป็นการเก็บข้อมูลของภาพตัวพิมพ์โน้ตเข้ามา โดยใช้สแกนเนอร์ หรือกล้อง CCD เข้ามาเก็บไว้ในแผ่นดิสก์ หรือ Frame memory ก็ได้

ขั้นตอนที่ 2 การปรับข้อมูลภาพให้มี 2 ระดับ ข้อมูลภาพที่อยู่ใน Frame memory หรือในแผ่นดิสก์จะถูกถ่ายลงในหน่วยความจำ จากนั้นจะถูกทำการแปลงให้มีค่าเพียง 2 ระดับโดยทำการตัดค่า threshold ที่เหมาะสมจะทำให้ภาพที่ได้มีความชัดเจนและง่ายต่อการวิเคราะห์

ขั้นตอนที่ 3 การแยกบรรทัด 5 เส้น

ภาพตัวพิมพ์โน้ตจะมีบรรทัด 5 เส้นเพื่อเป็นตัวบอกระดับเสียงของตัวโน้ต ในขั้นตอนนี้ เส้นบรรทัดทั้ง 5 เส้นจะถูกกำจัดออกโดยใช้หลักการการลบเส้นที่มีความยาวเป็นเส้นตรงที่ต่อเนื่องกันในแนวนอนและยาวมากที่สุด

ขั้นตอนที่ 4 การดึงสัญลักษณ์และตัวโน้ตดนตรี

การดึงสัญลักษณ์และตัวโน้ตดนตรีในภาพออกมาเป็นส่วนๆ เพื่อที่จะนำภาพที่ดึงออกมาไปหาคุณสมบัติและวิเคราะห์ต่อไป การดึงสัญลักษณ์และตัวโน้ตดนตรีในภาพออกมาทำได้โดยการใช้วิธีติดตามขอบของวัตถุในภาพ ด้วยตารางหน้าต่าง เพื่อตีกรอบภาพสัญลักษณ์และตัวโน้ตดนตรีของแต่ละตัว พร้อมทั้งทำการกำหนดหมายเลขให้กับส่วนของภาพแต่ละภาพที่ดึงออกมาได้ เพื่อนำไปใช้ในขั้นตอนการรู้จำต่อไป

ขั้นตอนที่ 5 การทำให้บาง

การทำให้บางเป็นการลดขนาดของข้อมูลภาพที่ได้จากขั้นตอนที่ 4 ลงเพื่อให้เหลือแต่โครงของภาพโดยใช้วิธีการทำ Thinning

ขั้นตอนที่ 6 การรู้จำตัวสัญลักษณ์และตัวโน้ตดนตรี

ขั้นตอนนี้จะทำการสแกนข้อมูลของภาพแต่ละภาพที่ได้มาจากขั้นตอนที่ 5 เพื่อหาลักษณะเด่นที่มีอยู่ในภาพทำการจัดแบ่งและตีความหมายของภาพ

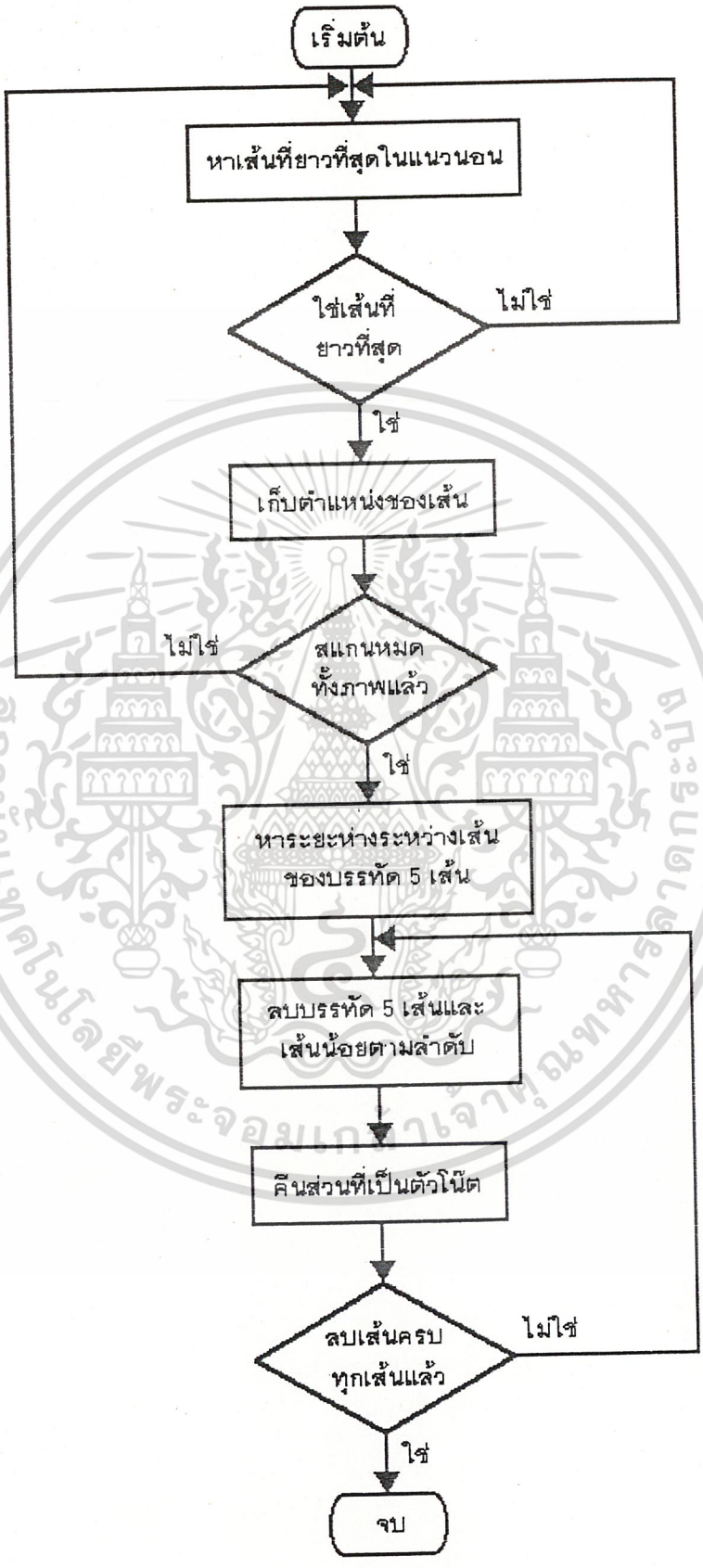
ขั้นตอนที่ 7 สร้างฐานข้อมูล

นำเอาผลการรู้จำในขั้นตอนที่ 6 มาเก็บไว้ในรูปของฐานข้อมูล เพื่อนำข้อมูลที่ได้ไปเปลี่ยนให้เป็นรหัสควบคุมของมิดี้คาร์ด

ในการเก็บภาพเข้ามาจะใช้สแกนเนอร์ สแกนรูปตัวโน๊ตเข้ามา จากนั้นก็ทำการตัด เทรสโฮล (threshold) เพื่อทำให้ภาพมีความชัดเจนและมีแค่ 2 ระดับเท่านั้นแล้วจึงนำไปทำในขั้นตอนต่อไป

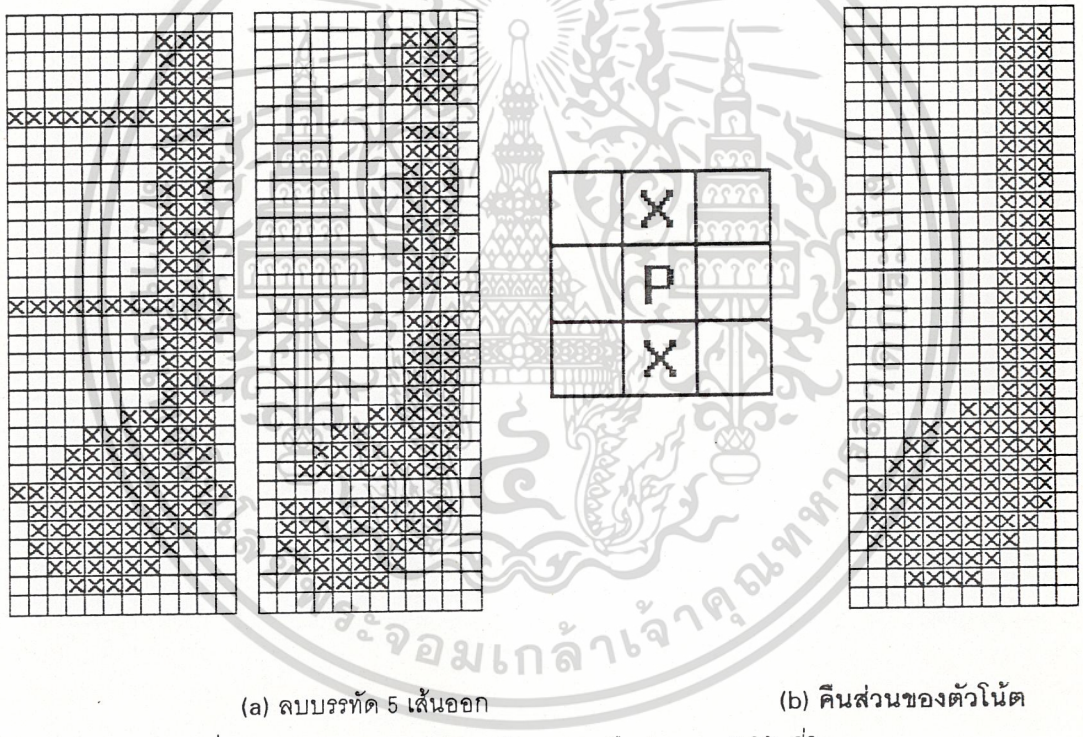
3.2 <u>การลบบรรทัด 5 เส้น</u> (staff extraction)

เมื่อเราได้ภาพ 2 ระดับของตัวโน้ตแล้ว ขั้นตอนต่อมาคือการแยกตัวโน้ตออกจากบรรทัด 5 เส้นโดยการลบบรรทัด 5 เส้นซึ่งมีวิธีการทำตามชาร์ตดังนี้



รูปที่ 3.2 ชาร์ตแสดงการลบบรรทัด 5 เส้น

จากชาร์ตจะเห็นว่าเริ่มด้วยจะต้องสแกนหาเส้นที่ยาวที่สุดในแนวนอน เมื่อเจอแล้วก็เก็บตำแหน่งเอาไว้ จากนั้นก็สแกนให้ทั่วทั้งภาพ ในการเขียนตัวพิมพ์โน้ตดนตรีตัวโน้ตบางแผ่นจะมีเส้นน้อยอยู่ด้วยซึ่งเราจะต้องทำการลบเส้นน้อยออกไปด้วย โดยมากระยะห่างระหว่างเส้นบรรทัด 5 เส้น กับระยะห่างระหว่างเส้นน้อยทั้งบนและล่างจะมีระยะห่างเท่ากัน จึงใช้หลักการหาระยะห่างระหว่างเส้นของบรรทัด 5 เส้นก่อนแล้วบวกเพิ่มไป 2 ครั้งจากเส้นบรรทัด 5 เส้นเส้นล่างสุด และลบออก 2 ครั้งจากเส้นบรรทัด 5 เส้นเส้นบนสุด จากนั้นลบบรรทัด 5 เส้นและเส้นน้อยทั้งบนและล่างด้านละ 2 เส้น เมื่อเราลบเส้นออกแล้วจะเห็นว่าทำให้ส่วนของตัวโน้ตโดนตัดออกไปด้วยเราจึงต้องทำการคืนส่วนของตัวโน้ต โดยนำตำแหน่งของเส้นที่ลบที่เก็บไว้มาสแกนว่าพิกเซลที่อยู่บนและล่างเป็น 1 หรือพิกเซลใดพิกเซลหนึ่งเป็น 0 หรือไม่ถ้าเป็น 1 ก็ให้คืนส่วนของตัวโน้ตหรือคืนส่วนที่ลบไปในครั้งแรกแต่ถ้าพิกเซลที่อยู่บนและล่างเป็น 0 แสดงว่าส่วนนั้นไม่ใช่ส่วนของตัวโน้ตก็ไม่ต้องคืนพิกเซลที่ลบในครั้งแรก



(a) ลบบรรทัด 5 เส้นออก (b) คืนส่วนของตัวโน้ต

รูปที่ 3.3 แสดงการลบบรรทัด 5 เส้นและการคืนส่วนของตัวโน้ตที่โดนลบ

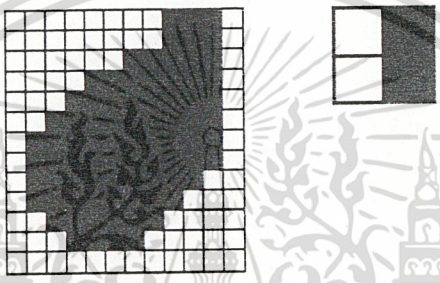
3.3 การวิเคราะห์หาขอบภาพ (Edge detection)

การวิเคราะห์หาขอบภาพเป็นงานวิเคราะห์เริ่มแรกของการจดจำภาพ เนื่องจากขอบของภาพเป็นข้อมูลสำคัญที่ทำให้สามารถทำนายขอบเขตของวัตถุ เส้นขอบเขตของเงาและอื่นๆ ซึ่งจากผลอันนี้จะนำไปสู่การวิเคราะห์โครงร่างของพื้นผิว (surface contour),โครงสร้างทางฟิสิกส์ (texture) ระยะทางและองค์ประกอบภายในของวัตถุในภาพ ซึ่งการวิเคราะห์เหล่านี้ล้วนเริ่มต้นจากการวิเคราะห์หาขอบของภาพทั้งสิ้น

3.3.1 นิยามของขอบภาพทางดิจิตอลและคณิตศาสตร์ที่ใช้ในการวิเคราะห์หาของขอบภาพ

ขอบของภาพทางดิจิตอล (edge in digital image) มีหลายลักษณะ เช่น ขอบภาพแบบหลังคา (roof adge) หรือขอบภาพที่มีการเปลี่ยนแปลงแบบทันทีทันใด (step adge) ดังนั้นจึงให้คำจำกัดความของ "ขอบภาพ" ว่า เป็นบริเวณภายในภาพที่ปรากฏการเปลี่ยนแปลงระดับความเข้มของแสงอย่างชัดเจนและทันทีทันใด

คณิตศาสตร์ที่ใช้วิเคราะห์หาขอบภาพนั้นทำได้ 2 ทางคือ frequency domain และทาง spatial domain ซึ่งการวิเคราะห์หาขอบภาพในที่นี้จะใช้วิธีทาง spatial domain ซึ่งเป็นวิธีการหาความสัมพันธ์ของข้อมูลภาพที่จุดๆหนึ่งกับข้อมูลภาพรอบจุดๆนั้น เรียกวิธีนี้ว่า neighborhood operators โดยมีขั้นตอนการวิเคราะห์ 2 ขั้นตอนคือ windowing และ masking หรือ template matching แสดงดังในรูปที่ 3.4



รูปที่ 3.4 ตัวอย่างการวิเคราะห์ด้วยวิธี neighborhood operators

การวิเคราะห์ภาพด้วยวิธีนี้โดยกำหนดตาราง (windows) W(p,q) ให้มีความสัมพันธ์กับจุดมุ่งหมายในการวิเคราะห์ เช่นในรูป 3.4 เป็นตัวอย่างการวิเคราะห์ในแนวตั้ง ดังนั้นการกำหนดตารางเป็น 2 ส่วนในแนวตั้ง จากนั้นทำการเลื่อนตารางไปตาม จุดต่างๆในภาพ เพื่อหาความสัมพันธ์ เรียกว่า cross-correlation; $\rho(i,j)$ ตามสมการที่ 3.1

$$\rho(i,j) = \sum_{p=-m}^{m}\sum_{q=-n}^{n} W(p,q)I(i+p,j+q) \quad .....(3.1)$$

$$m \le i \le M-m \ , \ n \le j \le N-n$$

โดยที่ I(i,j) แทนข้อมูลที่มีขนาด MxN และ $m \le i \le M \ ; \ n \le j \le N$

W(p,q) แทนตารางขนาด (2m+1) x (2n+1) เมื่อ m << M และ n << N

โดยมากจะกำหนดให้ขนาด M เท่า N และ m เท่ากับ n ซึ่งค่า cross-correlation นี้ เมื่อเกิดการ match กันขึ้นระหว่างข้อมูลของภาพกับตาราง จะทำให้เกิดค่าความสัมพันธ์ขึ้น อาจเป็นค่าสูงสุดหรือต่ำสุดหรือเป็นศูนย์ขึ้นอยู่กับตารางที่ตั้งขึ้นตามจุดมุ่งหมาย และค่าความสัมพันธ์นี้จะเกิดขึ้นบริเวณการเปลี่ยนแปลงของข้อมูลภาพหรือบริเวณขอบของภาพ

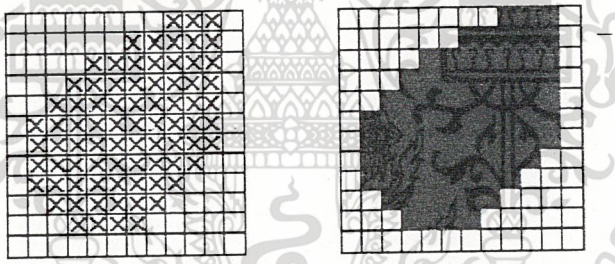
เทคนิคการหาค่า cross-correlation กับภาพนั้นคล้ายกับการฟิลเตอร์ภาพ ดังนั้นจึงเรียกวิธีนี้ว่า match filter และถ้ามองสมการที่ 3.1 ให้ดีก็เหมือนกับวิธี convolution เพียงแต่ค่าในตารางหมุนไป 180 องศา เท่านั้น ตามสมการ 3.2

$$g(i,j) = \sum_{p=-m}^{m} \sum_{q=-n}^{n} W(i-p, j-q) I(p,q) \quad .....(3.2)$$

ดังนั้นการวิเคราะห์หาขอบภาพด้วยวิธี neighborhood operator จึงต้อง convolution ข้อมูลภาพกับ operator ใดๆแทนด้วยสัญลักษณ์ W+I

3.3.2 เทคนิคการวิเคราะห์หาขอบภาพด้วยวิธีวิเคราะห์ความต่อเนื่อง

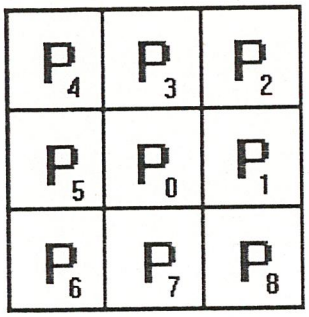
การวิเคราะห์หาขอบภาพทาง spatial domain มีทฤษฎีและเทคนิคมากมายขึ้นอยู่กับการประยุกต์ใช้ความรู้ทางคณิตศาสตร์ คอมพิวเตอร์ และอื่นๆแต่ในที่นี้จะใช้วิธี การวิเคราะห์ความต่อเนื่อง (connectivity analysis) ซึ่งหมายถึง การวิเคราะห์ข้อมูลภาพที่มีระดับเทาเพียง 2 ระดับคือระดับเทาเป็น 0 หรือ 1 ซึ่งเรียกข้อมูลภาพในลักษณะนี้ว่า ภาพสองระดับ (binary picture) ดังรูปที่ 3.5



(a) ข้อมูลภาพที่มีระดับเทาเป็น 0 กับ 1 (ช่องว่างไว้เป็น 1) (b) แสดงภาพ 2 ระดับ

รูปที่ 3.5 ภาพสองระดับ (binary picture)

เพื่อหาความสัมพันธ์ของจุดในภาพที่อยู่ข้างเคียง (neighbouring pixels) โดยอาศัยคุณสมบัติทางเรขาคณิต (topological properties) แทนด้วยสัญลักษณ์ $N_C(4)$ และ $N_C(8)$ โดยกำหนดรูปแบบความสัมพันธ์ของจุดเป็นตารางขนาด 3x3 ดังรูป 3.6



รูปที่ 3.6 รูปแบบความสัมพันธ์ของจุดเป็นตาราง 3x3

โดยทั่วไปกำหนดให้วัตถุหรือสิ่งที่สนใจ (object or pattern) มีค่าระดับเทาเป็น 1 ส่วนพื้นหรือสิ่งที่ไม่สนใจมีค่าระดับเทาเป็น 0 ดังนั้นจากรูป 3.6 จุด P_0 เป็นจุดใดๆภายในเนื้อวัตถุ ซึ่งนำมาวิเคราะห์หาความสัมพันธ์กับจุดรอบๆจุด P_0 นั้น อันประกอบด้วยจุด $P_1, P_2, P_3, P_4, P_5, P_6, P_7$ และ P_8 มีค่าระดับเทาเป็น 0 ดังนั้นค่าตัวเลขต่อเนื่อง $N_C(4)$ และ $N_C(8)$ สามารถคำนวณได้จากสมการ 3.3 และ 3.4

$$N_C(4) = \sum_{k \in S_1} (P_k - P_k \cdot P_{k+1} \cdot P_{k+2}) \quad .....(3.3)$$

$$N_C(8) = \sum_{k \in S_1} (P_k - P_k \cdot P_{k+1} \cdot P_{k+2}) \quad .....(3.4)$$

เมื่อ $S_1 = [1,3,5,7]$

$k \leq 8$

ถ้า $k \geq 9$ แล้ว $k = k-8$

$+$ หมายถึง เครื่องหมายคูณ $P_k = 1-P_k$

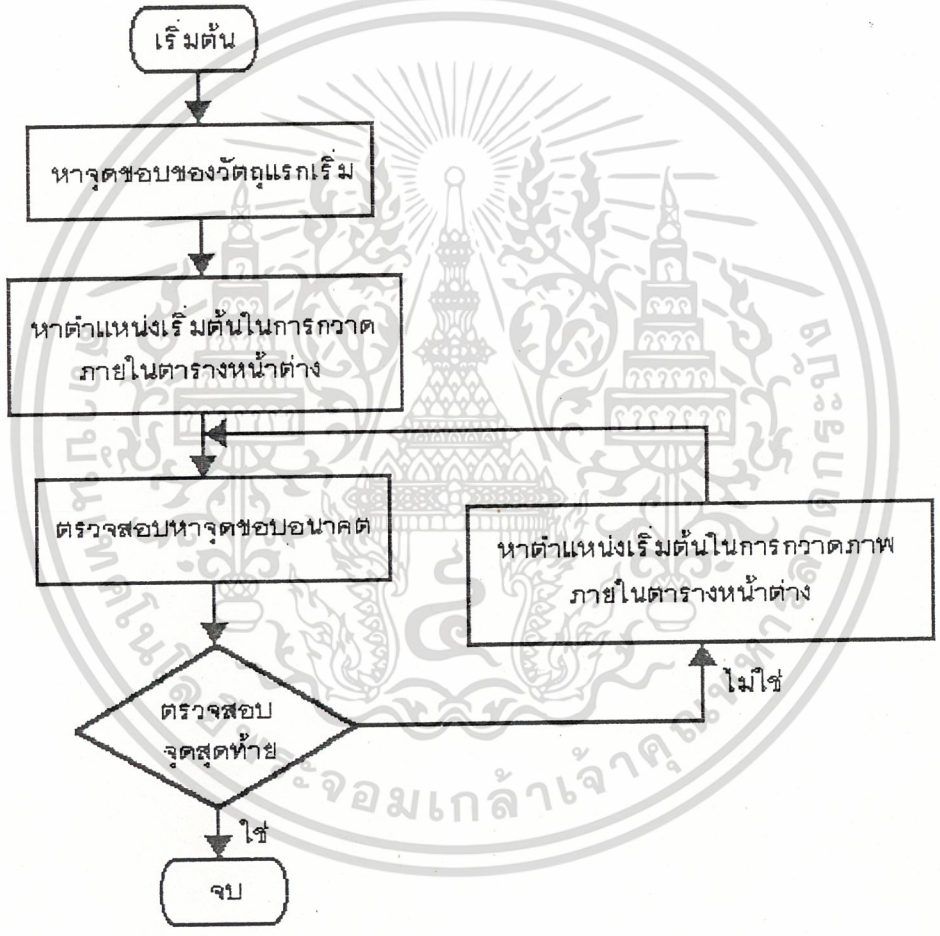
ค่าตัวเลขต่อเนื่อง $N_C(4)$ และ $N_C(8)$ นี้เป็นตัวกำหนดคุณสมบัติของจุด P_0 กับจุดรอบๆจุดนี้ ซึ่งความสัมพันธ์ต่างๆและค่าของ $N_C(4)$ และ $N_C(8)$ ตามตารางที่ 3.1

ค่าของ NC(4) และ NC(8)	0	1	2	3	4
ลักษณะของจุด	จุดอิสระภายใน	จุดปลาย	จุดต่อเนื่อง	จุดแยก	จุดตัด

ตารางที่ 3.1 คุณสมบัติของจุดกึ่งกลาง P_0 ของตารางขนาด 3x3 กับจุดรอบข้างๆ

เนื่องจากขอบของภาพในกรณีนี้ คือจุดภายในวัตถุที่มีค่าระดับเทาเป็น 1 และจุดปลายของวัตถุก่อนที่ระดับเทาจะเปลี่ยนเป็น 0 ดังนั้นจากตารางที่ 3.1 ทำให้สามารถหาขอบของวัตถุได้โดยการตรวจสอบค่า $N_C(4)$ และ $N_C(8)$ ว่ามีค่าเท่ากับ 1 หรือไม่ ถ้าค่าทั้งสองเป็น 1 แสดงว่าจุดนั้นมีความสัมพันธ์เป็นจุดปลาย หรือเป็นส่วนหนึ่งของเส้นขอบภาพนั่นเอง

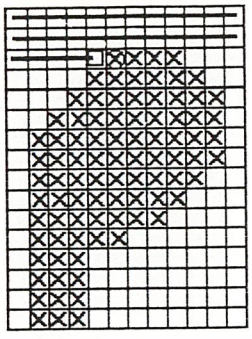
วิธีการตรวจหาขอบภาพด้วยการใช้ตารางหน้าต่าง
สามารถเขียนเป็นขั้นตอนตามชาร์ตดังนี้



รูปที่ 3.7 ชาร์ตแสดงขั้นตอนการตรวจหาขอบภาพ

ขั้นตอนที่ 1 จะต้องหาจุดขอบของภาพวัตถุที่ต้องการจะพิจารณาเสียก่อน โดยกวาดจุดภาพจากซ้ายไปขวาและบนลงล่าง

ขั้นตอนที่ 2 หาตำแหน่งเริ่มต้นในการกวาดจุดภาพ ภายในตารางหน้าต่างที่เหมาะสม ซึ่งจำเป็นที่จะต้อง



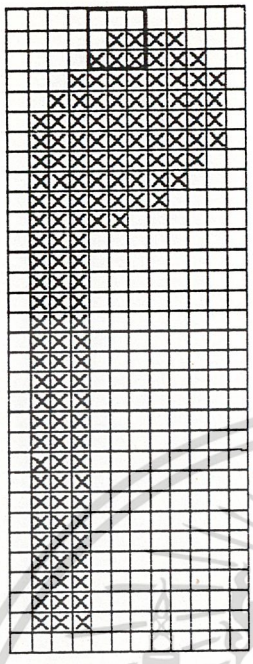
○ หมายถึงจุดขอบของวัตถุเริ่มต้น

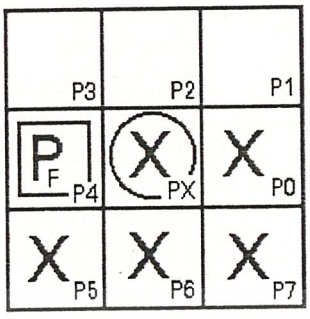
□ หมายถึงจุดภาพก่อนจุดขอบของวัตถุแรกเริ่ม

รูปที่ 3.8 แสดงการทำงานในขั้นตอนที่ 1 และ 2

นำเอาทิศทางการกวาดจุดภาพในขั้นตอนที่ 1 มาช่วยในการพิจารณาหาตำแหน่งเริ่มต้นนี้ด้วย กล่าวคือ เราจะใช้จุดภาพก่อนจุดขอบของวัตถุแรก (ในขั้นตอนที่ 1) หนึ่งตำแหน่งดังรูป 3.8 เป็นจุดเริ่มของการกวาดภายในตารางหน้าต่างซึ่งในรูป 3.8 ตำแหน่งเริ่มต้นในการกวาดจุดภาพภายในตารางหน้าต่างที่เหมาะสม นั้นคือ ตำแหน่ง P_4 อันเป็น รูป สี่เหลี่ยม ในรูปที่ 3.8 โดยที่ วงกลม เป็นตำแหน่งของ Px

ขั้นตอนที่ 3 ตรวจหาจุดขอบ (contour) อนาคต (P_0 - P_7) โดยเริ่มตรวจสอบจากตำแหน่ง P_F ที่หาได้จากขั้นตอนที่ 2 หรือ 5 ทำการกวาดรอบตารางหน้าต่างไปทีละหนึ่งตำแหน่งในทิศทางทวนเข็มหรือตามเข็มเพียงทิศทางใดทิศทางหนึ่งเท่านั้น และจุดแรกที่ตรวจพบว่าเป็นจุดขอบภาพของวัตถุให้ถือว่าจุดนั้นเป็นจุดขอบอนาคต เราจะใช้จุดขอบอนาคตจุดนี้เป็นจุดขอบของวัตถุที่ใช้หาจุดขอบอนาคตถัดไป ซึ่งจากรูปที่ 3.9 จะเห็นว่าทิศทางการตรวจสอบจะเริ่มที่ P_4,P_5,P_6,P_7,P_0,P_1,P_2 และ P_3 ในทิศทางทวนเข็ม และได้ P_5 เป็นจุดแรกที่ตรวจพบว่าเป็นจุดของภาพวัตถุก่อน ดังนั้น P_5 จึงเป็นจุดขอบอนาคต





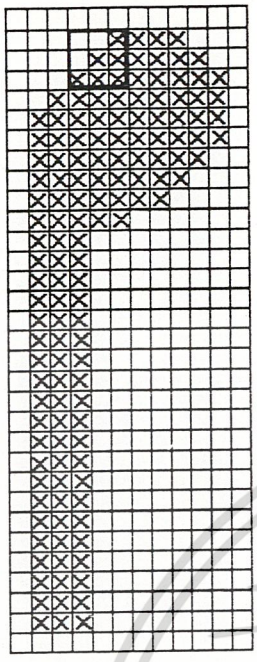
$P_F = P_4$ = ตำแหน่งเริ่มต้นของการตรวจสอบ

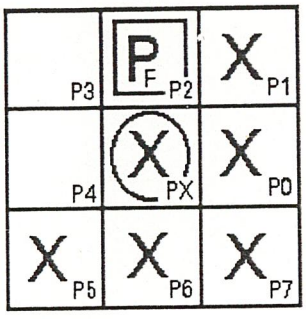
รูปที่ 3.9 แสดงตำแหน่งเริ่มต้น การกวาดรอบตารางหน้าต่างซึ่งหาได้จากการทำงานในขั้นตอนที่ 2 หรือ ขั้นตอนที่ 5

ขั้นตอนที่ 4 ตรวจสอบตำแหน่งของจุดขอบอนาคต ที่ได้จากขั้นตอนที่ 3 นั้นเป็นตำแหน่งสุดท้ายหรือไม่ถ้าไม่ใช่ก็ให้ไปทำในขั้นตอนที่ 5 ถ้าหากเป็นตำแหน่งสุดท้ายก็ให้หยุดการทำงาน

ขั้นตอนที่ 5 หาตำแหน่งเริ่มต้นในการกวาดรอบตารางหน้าต่าง ซึ่งการหาตำแหน่งเริ่มต้นในขั้นตอนนี้จะต่างไปจากขั้นตอนที่ 2 โดยขั้นตอนนี้จะใช้จุดขอบวัตถุอดีตหรือตำแหน่งของจุดขอบวัตถุปัจจุบันในขั้นตอนที่ 3 มาพิจารณาหาตำแหน่งเริ่มต้นในการกวาดรอบตารางหน้าต่าง

$P_{Final} = P_{Contour\ อดีต + 1}$ ดังรูปที่ 3.10





$P_F = P_2$ = ตำแหน่งเริ่มต้นของการตรวจสอบ

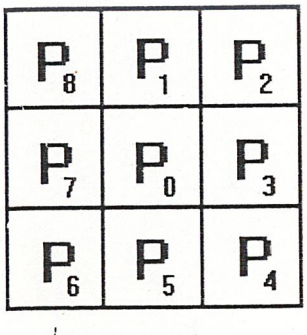
รูปที่ 3.10 แสดงตำแหน่งเริ่มต้นที่หาได้จากสูตร $P_{Final} = P_{Contour\ อดีต} + 1$

3.4 <u>ทฤษฎีโครงกระดูก</u> (skeleton)

วิธีการหาโครงร่างมีหลายวิธี และแต่ละวิธีก็ใช้งานแตกต่างกัน ซึ่งเราจะต้องเอาวิธีใดวิธีหนึ่งมาใช้กับโครงงานนี้ ในที่นี้เราใช้วิธีของ Zhang และ Suen ซึ่งได้พัฒนาขึ้นในปี ค.ศ. 1984 ใช้สำหรับ thining binary regions หมายถึงภาพที่มีลักษณะบาง และข้อมูลที่เก็บจะเป็นลักษณะ binary คือ จุดที่เป็นภาพจะมีค่าเป็น 1 และจุดที่เป็นฉากหลังจะมีค่าเป็น 0 ซึ่งวิธีการของ skeleton นี้จะทำให้เราได้เส้นโครงร่างที่มีความหนา 1 พิกเซลอยู่ที่กึ่งกลางภาพ

วิธีการมีอยู่ 2 ขั้นตอน ซึ่งจะใช้กับ contour point ที่อยู่ภายในขอบเขตของภาพโดย contour point เป็นจุดที่มีค่าเป็น 1 และมีจุดรอบๆอย่างน้อย 1 จุดที่มีค่าเป็น 0

ในการตรวจเช็คจุดภาพทำโดยนำเอาเมตริกซ์ขนาด 3x3 ไปวางครอบจุดที่ต้องการเช็คแล้วใช้จุดที่อยู่รอบๆตัวมันเป็นตัวตรวจสอบ จากรูป 3.11 เป็นเมตริกซ์ขนาด 3x3 ที่นำไปครอบจุดที่เราต้องการตรวจสอบ



รูปที่ 3.11 เมตริกซ์ขนาด 3x3

ขั้นตอนการตรวจสอบจุดของภาพ

ขั้นตอนที่ 1

เมื่อเราใช้เมตริกซ์ ขนาด 3x3 ไปครอบจุดที่ต้องการตรวจสอบและจุดนั้นจะถูกลบเมื่อเงื่อนไขต่อไปนี้เป็นจริง

$$\left.\begin{array}{l}(a).\ 2 \le N(P_0) \le 6 \\ (b).\ S(P_0) = 1 \\ (c).\ P1*P3*P5 = 0 \\ (d).\ P3*P5*P7 = 0\end{array}\right\} \quad .....(3.5)$$

โดยที่ $N(P_0)$ คือจำนวนจุดรอบจุด P_0 ซึ่งมีค่าเป็น 1 เราจึงได้

$$N(P_0) = P_1 + P_2 + \ldots + P_7 + P_8 \quad .....(3.6)$$

และ $S(P_0)$ คือจำนวนของการเปลี่ยนแปลงจาก 0 ไปเป็น 1 ในลำดับของ P1,P2,...,P8 ตัวอย่างในรูปที่ 3.12 ได้ $N(P_0) = 4$ และ $S(P_0) = 3$

0	0	1
1	P_0	0
1	0	1

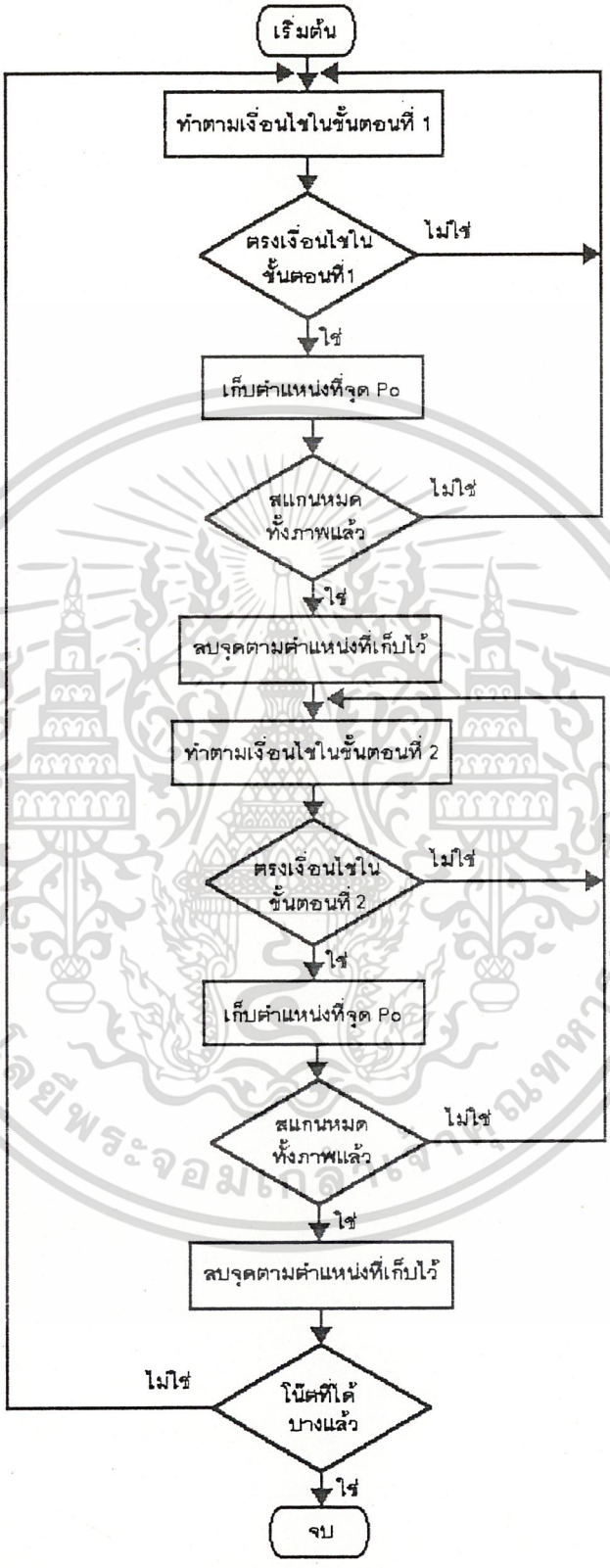
รูปที่ 3.12 แสดงการตรวจสอบเงื่อนไข

จากสมการที่ 3.5 จะได้ $N(P_0) = 4$ และ $S(P_0) = 3$

ขั้นตอนที่ 2

สำหรับขั้นตอนนี้ จะใช้เงื่อนไข (a) และ (b) ในการตรวจสอบจะเหมือนขั้นตอนแรก แต่เงื่อนไข (c) และ (d) จะถูกเปลี่ยนไป

$$\left.\begin{array}{l}(a).\ 2 \le N(P_0) \le 6 \\ (b).\ S(P_0) = 1 \\ (c').\ P1*P3*P7 = 0 \\ (d').\ P1*P5*P7 = 0\end{array}\right\} \quad .....(3.7)$$

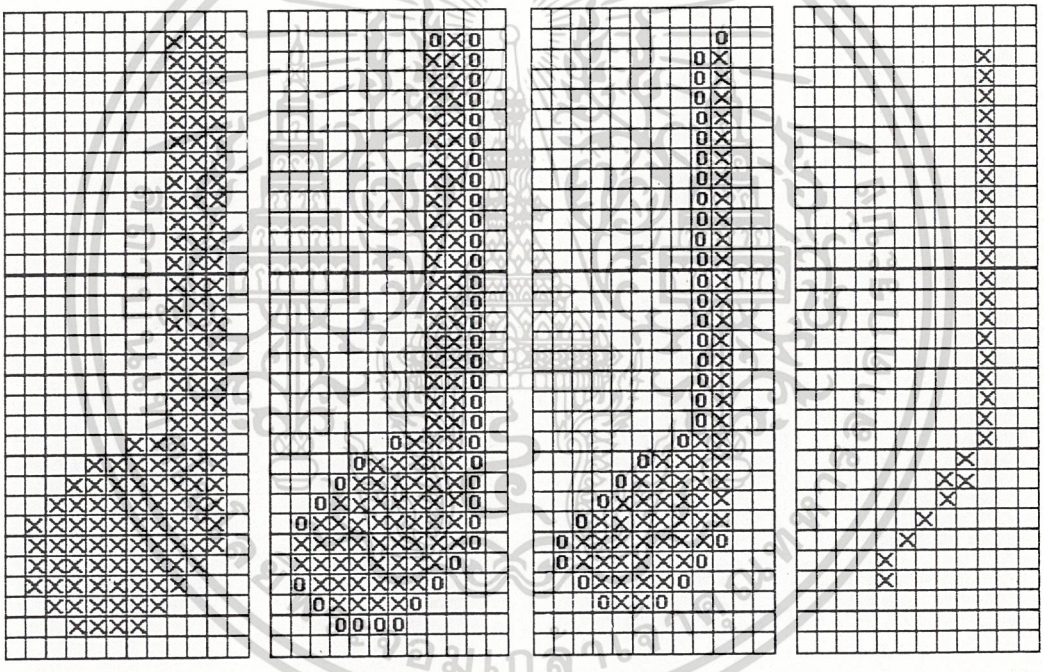


รูปที่ 3.13 แสดงชาร์ตการหาโครงร่างของตัวโน้ต

วิธีการ

จากขั้นตอนที่หนึ่ง เราจะตรวจสอบทุกจุดในภาพ เพื่อหาจุดที่มีเงื่อนไขเป็นจริงตาม (a) ถึง (d) ถ้าพบจุดที่ตรงตามเงื่อนไข ก็ทำสัญลักษณ์ (mark) ที่จุดนั้นก่อน รอจนกว่าตรวจสอบจุดทุกจุดในภาพครบแล้ว จึงทำการลบจุดที่ได้ทำสัญลักษณ์ไว้ทั้งหมด จากนั้นทำการตรวจสอบใหม่ทั้งภาพตามเงื่อนไขในขั้นตอนที่สองและลบจุดที่ทำสัญลักษณ์ไว้ทั้งหมดเช่นกัน ทำซ้ำในขั้นตอนแรกใหม่จนกว่าจะไม่มีจุดที่ตรงตามเงื่อนไขใดๆ เลย ขั้นสุดท้ายเราจะได้โครงร่างของภาพมีลักษณะเป็นเส้นขนาด 1 พิกเซลที่อยู่กึ่งกลางภาพ

จากรูป 3.14 รูป (a) แสดงการประมวลผลในขั้นตอนที่ 1 จะเห็นว่าจุดที่ถูกลบเป็นจุดที่อยู่ขอบด้านขวา,ด้านล่างและจุดของมุมบนด้านซ้าย รูป (b) เป็นผลที่ได้จากการประมวลผลขั้นที่ 2 จะเห็นว่าจุดที่ถูกลบจะเป็นจุดที่ขอบด้านซ้าย,ด้านบนและมุมล่างด้านซ้าย รูป (c) คือโครงร่างของภาพที่ได้ เมื่อทำการทำซ้ำในขั้นตอนที่ 1 และ 2 จนไม่มีจุดใดที่ถูกต้องตามเงื่อนไข



(a) (b) (c)

รูปที่ 3.14 (a) ผลจากขั้นตอนที่ 1ในการประมวลผลรอบแรก

(b) ผลจากขั้นตอนที่ 2 ในการประมวลผลรอบแรก

(c) ผลจากการประมวลผลรอบสุดท้าย

3.5 การหาลักษณะเด่นของตัวโน้ต

ก่อนที่จะหาลักษณะเด่นของตัวโน้ตได้นั้น ตัวโน้ตจะต้องผ่านขั้นตอนการทำ skeleton มาแล้วทั้งนี้ก็เพื่อให้เหลือเฉพาะโครงร่างของตัวโน้ตเท่านั้น โดยโครงร่างของตัวโน้ตแต่ละตัวจะมีลักษณะเด่นที่ไม่เหมือนกัน ซึ่งลักษณะเด่นของตัวโน้ตคือ จุดตัด,จุดแยกและจุดปลาย วิธีการหาลักษณะเด่นทำได้โดยใช้หน้าต่าง 3x3

ครอบจุดที่ต้องการหาทั้งโครงร่างของตัวโน้ต ว่าจุดที่อยู่กลางกรอบหน้าต่างมีลักษณะเด่นเป็นอย่างไร ซึ่งหน้าต่างที่จะใช้เป็นเครื่องมือในการหาลักษณะเด่นของตัวโน้ตแสดงดังรูปที่ 3.15

X_4	X_3	X_2
X_5	X_0	X_1
X_6	X_7	X_8

รูปที่ 3.15 แสดงหน้าต่างขนาด 3x3

ในการหาลักษณะเด่น จะใช้จุด X_0 เป็นจุดที่จะบอกว่ามีลักษณะเด่นเป็นเช่นไร โดยพิจารณาจุดที่ล้อมรอบจุด X_0 ทั้ง 8 ว่ามีเงื่อนไขเป็นอย่างไร ในการหาจุดแยก,จุดตัดและจุดปลายทำได้ดังนี้

จุดตัดเป็นจุดที่มีส่วนประกอบรอบๆมีการเปลี่ยนปลงจากค่า 1 ไปเป็น 0 อยู่ 4 ช่วง

จุดแยกเป็นจุดที่มีส่วนประกอบรอบๆมีการเปลี่ยนปลงจากค่า 1 ไปเป็น 0 อยู่ 3 ช่วง

จุดปลายเป็นจุดที่มีส่วนประกอบรอบๆมีการเปลี่ยนปลงจากค่า 1 ไปเป็น 0 อยู่ 1 ช่วง

ในที่นี้เราจะกำหนดฟังก์ชั่นขึ้น เพื่อจำแนกกลุ่มของจุดภาพตามการเปลี่ยนแปลงจากค่า 1 ไปเป็น 0 ที่กล่าวข้างต้นดังนี้

$$N_s = \sum_{k\in\{1,2,\ldots,8\}} X_k - X_{k-1} \quad .....(3.8)$$

เมื่อ Ns คือจำนวนครั้งที่มีการเปลี่ยนแปลงของส่วนประกอบรอบๆจากค่า 1 ไปเป็น 0 หรือจากค่า 0 ไปเป็น 1

เนื่องจาก Ns นี้ยังไม่สามารถจำแนกได้ครบถ้วน โดยเฉพาะในกรณีของจุดปลายกับจุดต่อ ดังนั้นเราจึงต้องกำหนดฟังก์ชั่นมาประกอบเพิ่มเติมซึ่งก็ได้กำหนดดังนี้

$$N_r = \sum_{k\in\{2,4,6,8\}} (X_k + X_{k+1} + X_{k+2}) \quad .....(3.9)$$

เมื่อ Nr คือผลรวมของค่าที่เป็น 0 ในแต่ละด้านของส่วนประกอบทั้งสี่ $X = (1-x)$ สมการทั้งสองสมการสามารถหาจุดตัด จุดแยก จุดปลายและจุดต่อเนื่องของ X_0 ได้โดยการประกอบกันดังแสดงในตารางที่ 3.2

ฟังก์ชั่น	คุณสมบัติของ X_0				
ลักษณะเด่น	จุดตัด	จุดแยก	จุดปลาย	จุดต่อ	
Ns	8	6	2	2	4
Nr	4,8	5-9	7,9,10,11	1-6,8	2-10

ตารางที่ 3.2 แสดงค่าจำแนกของค่าคุณสมบัติของ X_0

เมื่อนำตารางที่ 3.2 มาเขียนการทำงานในลักษณะการตัดสินใจจะได้ดังนี้

1) Ns =, 6 ——> X_0 คือ จุดแยก (branch) Nr = 5 ถึง 9

2) Ns = 8 ——> X_0 คือ จุดตัด (cross) Nr = 4 หรือ 8

3) Ns = 2 และ (Nr = 7 หรือ Nr = 9 ถึง 11) ——> X_0 คือ จุดปลาย (end)

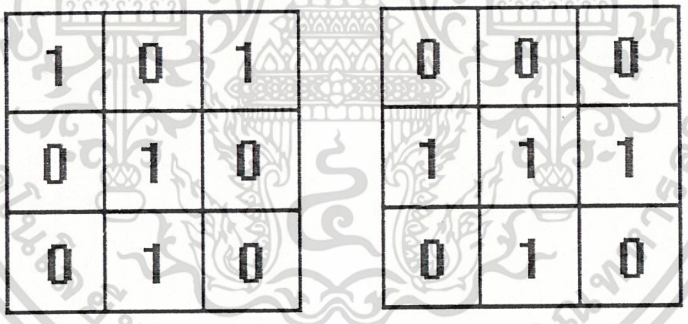
4) Ns = 4 และ (Nr = 1 ถึง 6 หรือ Nr = 8) ——> X_0 คือ จุดต่อ (connect)

5) Ns = 4 และ (Nr = 2 ถึง 10) ——> X_0 คือ จุดต่อ (connect)

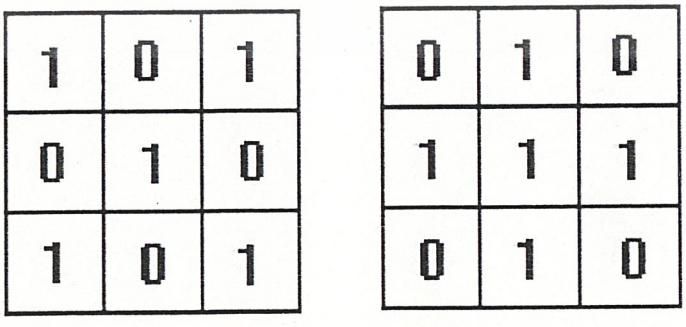
สำหรับโครงงานฉบับนี้พิจารณาเฉพาะค่าของ Ns จึงมีลักษณะในการตัดสินใจดังนี้

คือ Ns = 8 คือจุดตัด,Ns = 6 คือจุดแยก และ Ns = 2 คือจุดปลาย

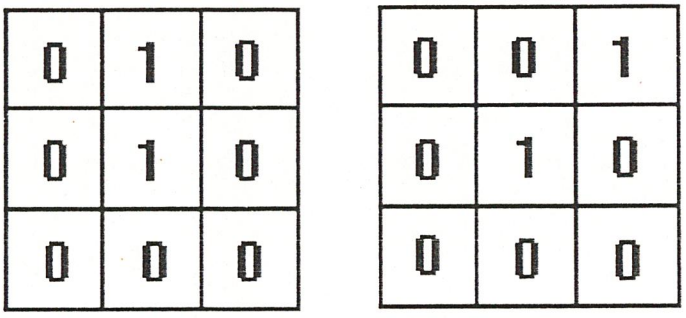
ตัวอย่างของจุดแยก จุดตัด และจุดปลายแสดงดังรูปที่ 3.16 , 3.17 และ 3.18 ตามลำดับ



รูปที่ 3.16 ตัวอย่างของจุดแยก



รูปที่ 3.17 ตัวอย่างของจุดตัด



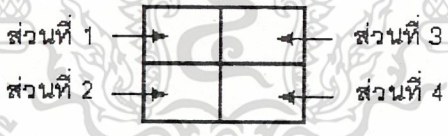
รูปที่ 3.18 ตัวอย่างของจุดปลาย

เมื่อพบว่าจุดใหนเป็นจุดแยก,จุดตัด และจุดปลายก็ทำดังต่อไปนี้

1. ถ้าเป็นจุดตัดให้ใส่หมายเลข 8 ที่จุดนั้น
2. ถ้าเป็นจุดแยกให้ใส่หมายเลข 6 ที่จุดนั้น
3. ถ้าเป็นจุดปลายให้ใส่หมายเลข 2 ที่จุดนั้น

3.6 การแบ่งกลุ่มตัวโน้ต (Note classification)

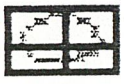
จากการหาลักษณะเด่นของตัวโน้ต ทำให้สามารถหาจำนวนและตำแหน่งของจุดตัด,จุดแยก และจุดปลายของตัวโน้ตแต่ละตัว เพื่อที่จะนำลักษณะเด่นของตัวโน้ตแต่ละตัวมาทำขั้นตอนการจดจำรูปแบบ โดยตัวโน้ตแต่ละตัวจะถูกแบ่งออกเป็น 4 ส่วน (Zone) ดังรูปที่ 3.19



รูปที่ 3.19 การแบ่งส่วนของตัวโน้ตออกเป็น 4 ส่วน

เมื่อเราแบ่งตัวโน้ตออกเป็น 4 ส่วน ทำให้เราสามารถทราบจำนวนของจุดตัด,จุดแยก และจุดปลายที่อยู่ในแต่ละส่วนของตัวโน้ตแต่ละตัวได้ก็จะสามารถแบ่งกลุ่มของตัวโน้ตตามลักษณะเด่นได้ 12 กลุ่ม (Group) ดังนี้

กลุ่มที่ 1 (Group 1) คือกลุ่มที่ไม่มีจุดตัด ,จุดแยกและจุดปลายในส่วนต่างๆทั้ง 4 ส่วนเลย ได้แก่ โน้ตตัวกลม



รูปที่ 3.20 ลักษณะเด่นของตัวโน้ตในกลุ่มที่ 1

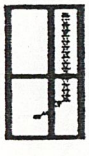
กลุ่มที่ 2 (Group 2) คือกลุ่มที่มีจุดปลายอยู่ในส่วนที่ 3 หนึ่งจุด และมีจุดแยกในส่วนที่ 4 หนึ่งจุด ได้แก่ โน้ตตัวขาว

รูปที่ 3.21 ลักษณะเด่นของตัวโน้ตในกลุ่มที่ 2

กลุ่มที่ 3 (Group 3) คือกลุ่มที่มีจุดแยกในส่วนที่ 1 หนึ่งจุด และมีจุดปลายในส่วนที่ 2 หนึ่งจุด ได้แก่ โน้ตตัวขาวคว่ำ

รูปที่ 3.22 ลักษณะเด่นของตัวโน้ตในกลุ่มที่ 3

กลุ่มที่ 4 (Group 4) คือกลุ่มที่มีจุดปลายในส่วนที่ 2 หนึ่งจุด และมีจุดปลายในส่วนที่ 3 หนึ่งจุด ได้แก่ โน้ตตัวดำ



รูปที่ 3.23 ลักษณะเด่นของตัวโน้ตในกลุ่มที่ 4

กลุ่มที่ 5 (Group 5) คือกลุ่มที่มีจุดปลายในส่วนที่ 2 หนึ่งหรือสองจุด และมีจุดปลายในส่วนที่ 4 หนึ่งหรือสองจุด เช่น โน้ตตัวเขบ็ตสองตัวติดกัน

รูปที่ 3.24 ลักษณะเด่นของตัวโน้ตในกลุ่มที่ 5

กลุ่มที่ 6 (Group 6) คือกลุ่มที่มีจุดปลายในส่วนที่ 1 หนึ่งหรือสองจุด และมีจุดปลายในส่วนที่ 3 หนึ่งหรือสองจุด เช่นโน้ตตัวเขบ็ตหงายสองตัวติดกัน

รูปที่ 3.25 ลักษณะเด่นของตัวโน้ตในกลุ่มที่ 6

กลุ่มที่ 7 (Group 7) คือ กลุ่มที่มีจุดปลายในส่วนที่1 สองจุด และมีจุดปลายในส่วนที่ 2 หนึ่งจุด ได้แก่ ตัวเขบ็ตหนึ่งชั้นคว่ำ ตัวเขบ็ตสองชั้นคว่ำ และตัวหยุดตัวเขบ็ตสองชั้น

รูปที่ 3.26 ลักษณะเด่นของตัวโน้ตในกลุ่มที่ 7

กลุ่มที่ 8 (Group 8) คือกลุ่มที่มีจุดปลายในส่วนที่1 หนึ่งจุด และมีจุดแยกในส่วนที่ 1 หนึ่งจุด ได้แก่ ตัวชาร์ป ตัวแฟลต และตัวเนเจอรัล

รูปที่ 3.27 ลักษณะเด่นของตัวโน้ตในกลุ่มที่ 8

กลุ่มที่ 9 (Group 9) คือกลุ่มที่มีจุดปลายในส่วนที่1 หนึ่งจุด และมีจุดปลายในส่วนที่ 2 หนึ่งจุด ได้แก่ ตัวหยุดเขบ็ตหนึ่งชั้น,ตัวหยุดตัวดำ และเครื่องหมาย บอกจังหวะ เลข 3

รูปที่ 3.28 ลักษณะเด่นของตัวโน้ตในกลุ่มที่ 9

กลุ่มที่ 10 (Group 10) คือกลุ่มที่มีจุดปลายในส่วนที่1 หนึ่งจุด และมีจุดปลายในส่วนที่ 3 หนึ่งจุด ได้แก่เครื่องหมายบอกจังหวะ เลข 2

รูปที่ 3.29 ลักษณะเด่นของตัวโน้ตในกลุ่มที่ 10

กลุ่มที่ 11 (Group 11) คือกลุ่มที่มีจุดปลายในส่วนที่ 4 หนึ่งจุด และมีจุดแยกในส่วนที่ 4 หนึ่งจุด ได้แก่เครื่องหมายบอกจังหวะ เลข 4

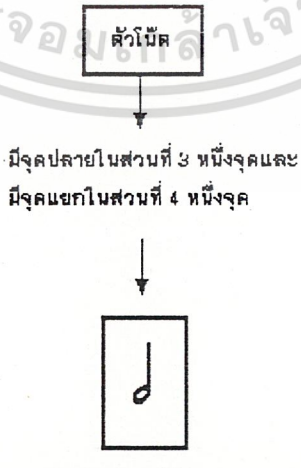
รูปที่ 3.30 ลักษณะเด่นของตัวโน้ตในกลุ่มที่ 11

กลุ่มที่ 12 (Group 12) คือกลุ่มที่มีจุดแยกในส่วนที่ 1 สองจุด,มีจุดปลายในสวนที่ 2 สองจุด และจุดตัดในส่วนที่ 4 หนึ่งจุด ได้แก่กุญแจประจำหลักซอล

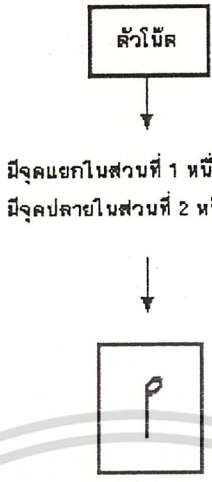
รูปที่ 3.31 ลักษณะเด่นของตัวโน้ตในกลุ่มที่ 12

จะสังเกตเห็นได้ว่าตัวโน้ตในกลุ่มที่ 5,6,7,8 และ 9 สามารถแบ่งเป็นกลุ่มย่อย (Subgroup) ได้อีก ซึ่งการแบ่งกลุ่มของตัวโน้ตสามารถแยกกลุ่มได้โดยการทำโครงร่างแบบต้นไม้ (Tree structure)

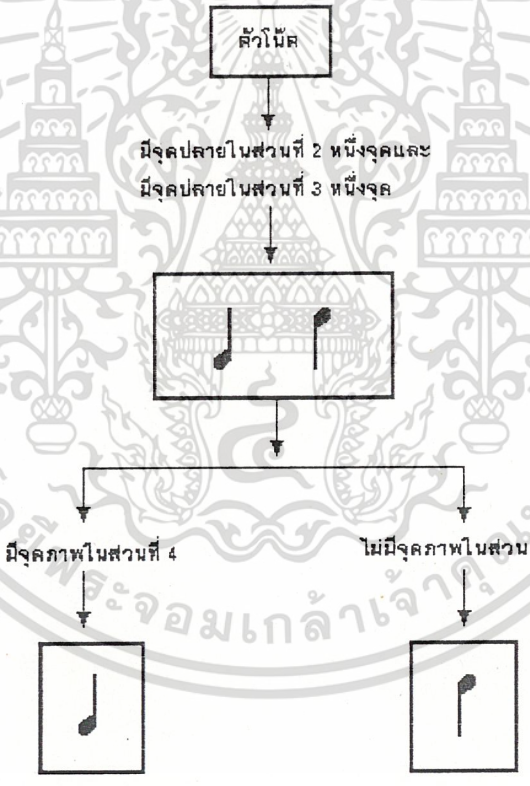
รูปที่ 3.32 โครงร่างแบบต้นไม้ของตัวโน้ตในกลุ่มที่ 1



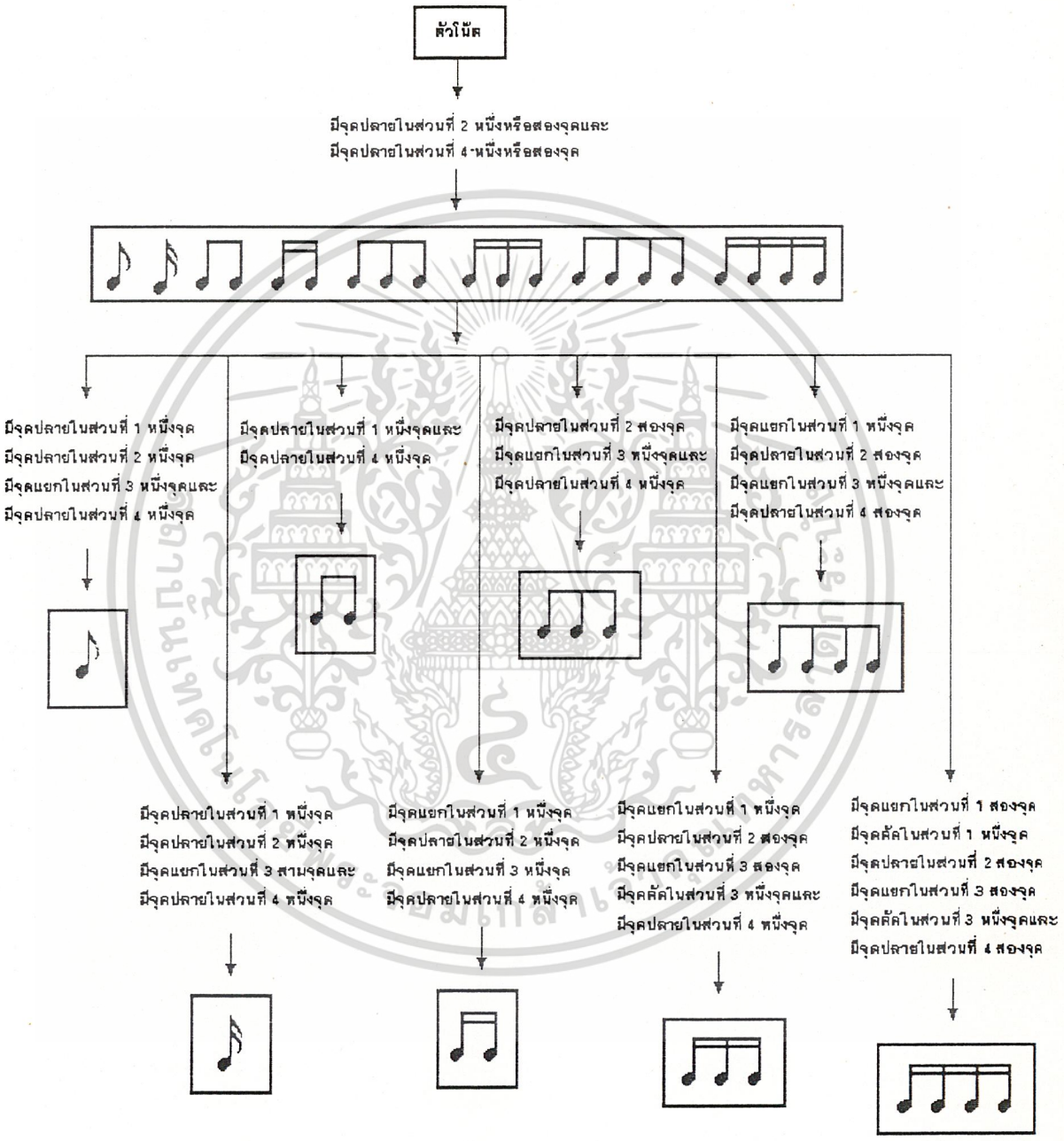
รูปที่ 3.33 โครงร่างแบบต้นไม้ของตัวโน้ตในกลุ่มที่ 2



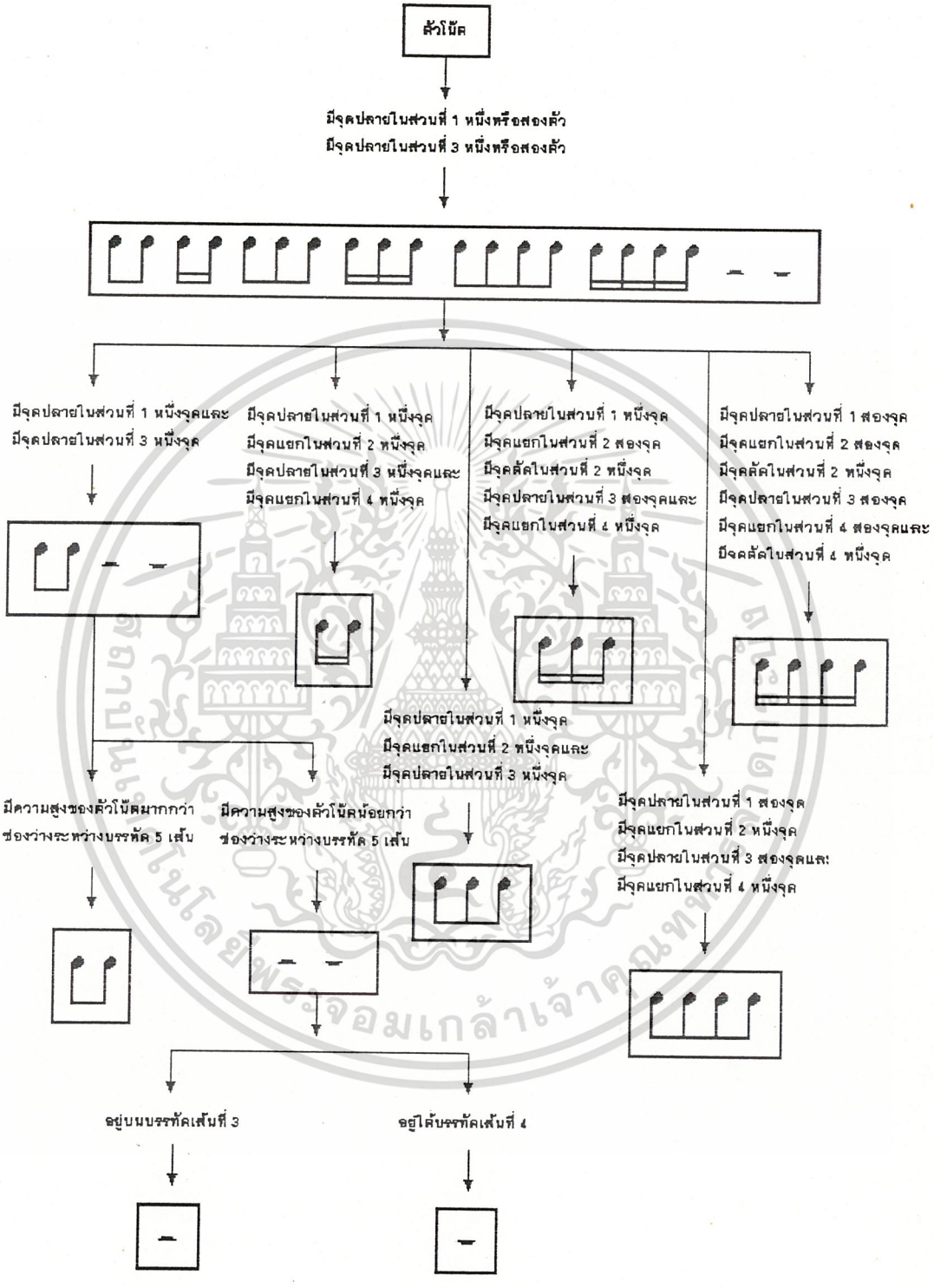
รูปที่ 3.34 โครงร่างแบบต้นไม้ของตัวโน้ตในกลุ่มที่ 3



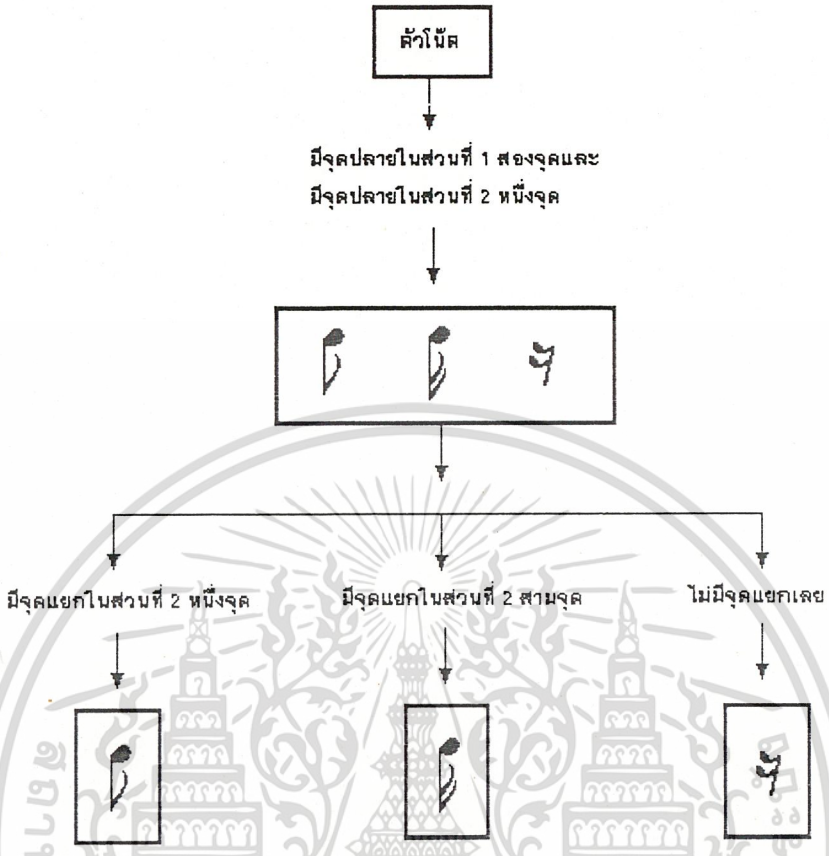
รูปที่ 3.35 โครงร่างแบบต้นไม้ของตัวโน้ตในกลุ่มที่ 4



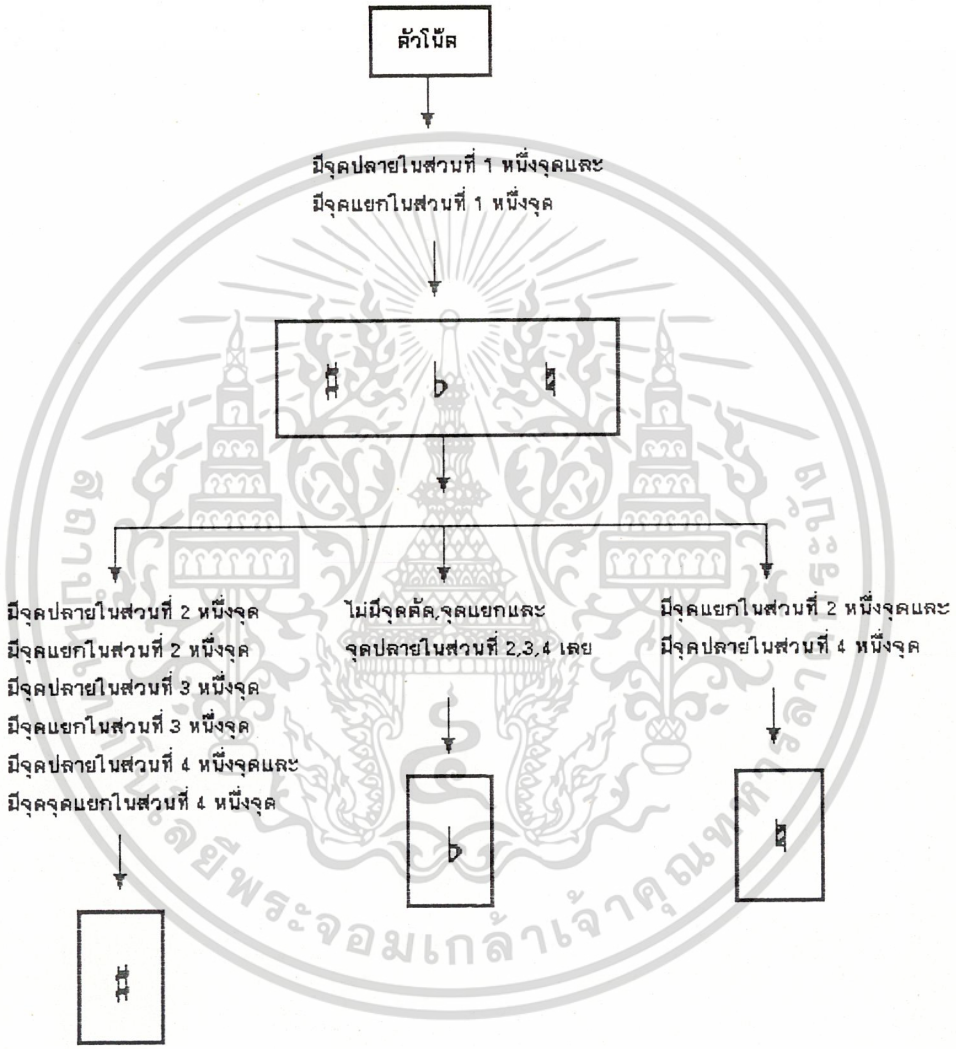
รูปที่ 3.36 โครงร่างแบบต้นไม้ของตัวโน้ตในกลุ่มที่ 5



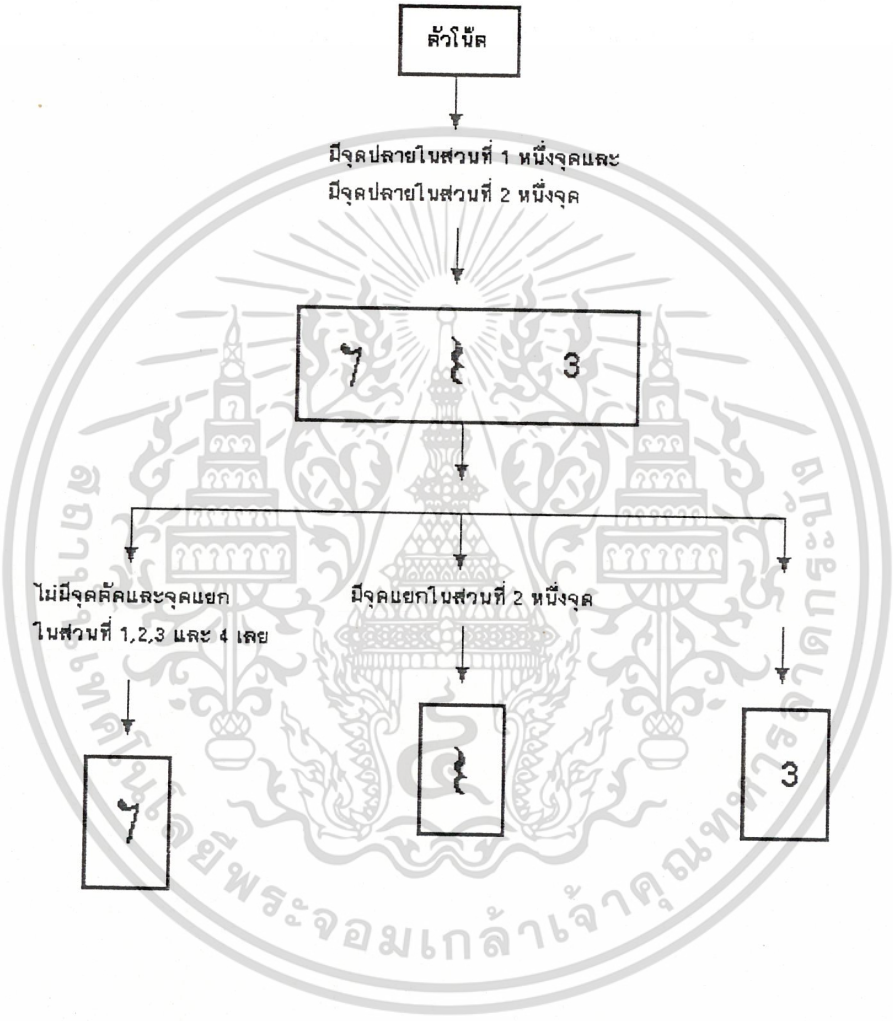
รูปที่ 3.37 โครงร่างแบบต้นไม้ของตัวโน้ตในกลุ่มที่ 6



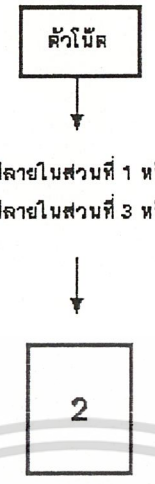
รูปที่ 3.38 โครงร่างแบบต้นไม้ของตัวโน้ตในกลุ่มที่ 7



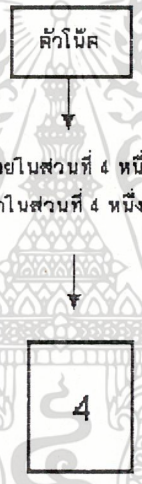
รูปที่ 3.39 โครงร่างแบบต้นไม้ของตัวโน้ตในกลุ่มที่ 8



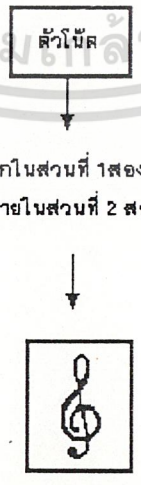
รูปที่ 3.40 โครงร่างแบบต้นไม้ของตัวโน้ตในกลุ่มที่ 9



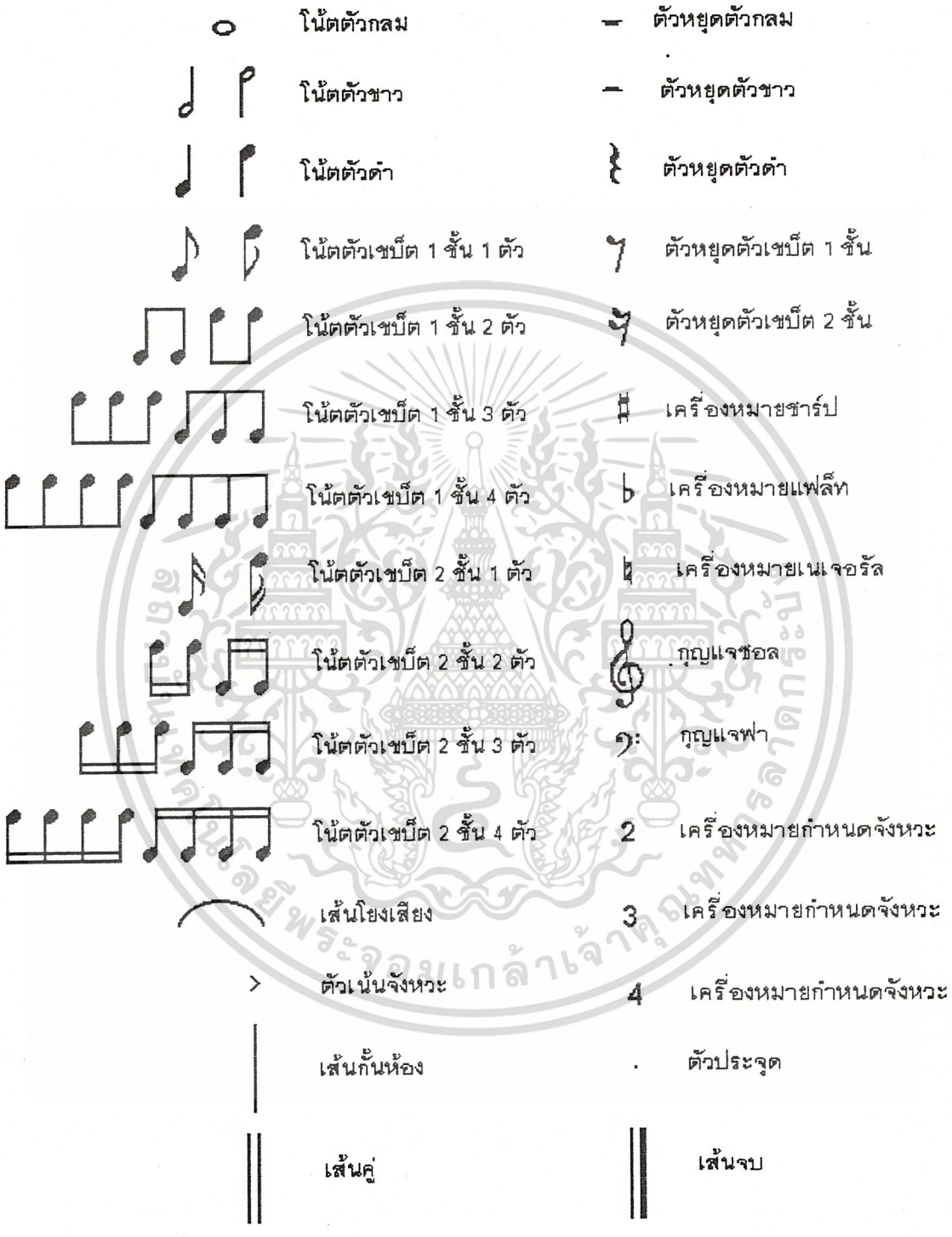
รูปที่ 3.41 โครงร่างแบบต้นไม้ของตัวโน้ตในกลุ่มที่ 10



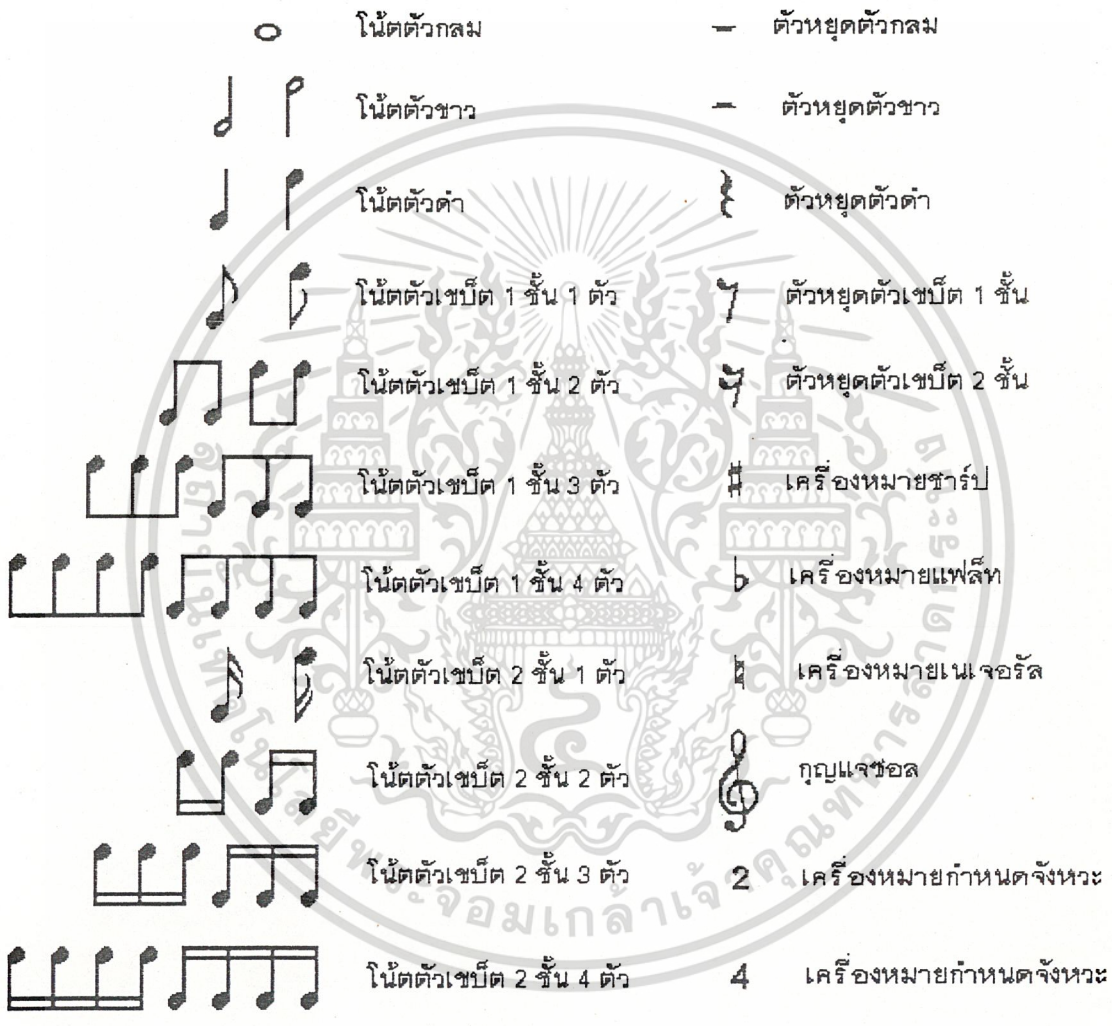
รูปที่ 3.42 โครงร่างแบบต้นไม้ของตัวโน้ตในกลุ่มที่ 11



รูปที่ 3.43 โครงร่างแบบต้นไม้ของตัวโน้ตในกลุ่มที่ 12



รูปที่ 3.44 ภาพตัวพิมพ์โน้ตและเครื่องหมายทางดนตรีสากล

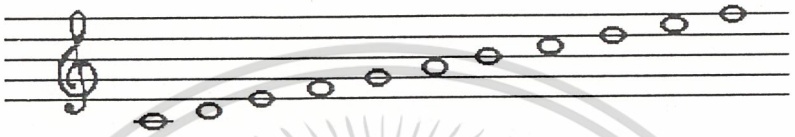


รูปที่ 3.45 ภาพตัวพิมพ์โน้ตและเครื่องหมายทางดนตรีที่ทำการวิเคราะห์ได้

3.7 <u>การหาตำแหน่งของตัวโน้ตบนบรรทัด 5 เส้น</u>

เมื่อเราทราบว่าตัวโน้ตแต่ละตัวเป็นโน้ตชนิดใด ก็จะทำให้รู้ความสั้น - ยาวของเสียง แต่เรายังไม่ทราบ2ว่าเสียงนั้นมีระดับเสียงเป็นอย่างไร ระดับเสียงของตัวโน้ต หมายถึงความสูง - ต่ำของระดับเสียง ซึ่งมีชื่อเรียกเรียงตามลำดับจากเสียงต่ำไปเสียงสูง รวม 7 ชื่อดังนี้

ชื่อระดับเสียง	โด	เร	มี	ฟา	ซอล	ลา	ที
ชื่อตัวอักษร	C	D	E	F	G	A	B



รูปที่ 3.44 แสดงระดับเสียงของตัวโน้ตบนบรรทัด 5 เส้น

ระดับเสียงของตัวโน้ตแต่ละตัวสามารถหาได้จากตำแหน่งของตัวโน้ตบนบรรทัด 5 เส้น จากขั้นตอนการแยกตัวโน้ตออกจากบรรทัด 5 เส้น เราได้เก็บตำแหน่งของบรรทัด 5 เส้นเอาไว้ และจากขั้นตอนการตีกรอบแยกตัวโน้ตแต่ละตัวออกจากกันเราได้เก็บตำแหน่งของตัวโน้ตแต่ละตัวเอาไว้เช่นกัน เมื่อนำเอาตำแหน่งของตัวโน้ตไปเปรียบกับบรรทัด 5 เส้น ก็จะทำให้ทราบระดับเสียงของตัวโน้ตแต่ละตัว

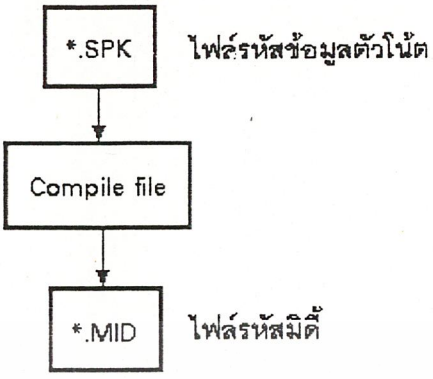
ในการเอาตำแหน่งของตัวโน้ตไปเปรียบเทียบกับตำแหน่งของบรรทัด 5 เส้นสามารถแบ่งได้ 2 ประเภท

1. ถ้าเป็นตัวโน้ตหงายจะเอาตำแหน่งต่ำสุดของตัวโน้ต ไปเปรียบเทียบกับตำแหน่งของบรรทัด 5 เส้น

2. ถ้าเป็นตัวโน้ตคว่ำจะเอาตำแหน่งสูงสุดของตัวโน้ต ไปเปรียบเทียบกับตำแหน่งของบรรทัด 5 เส้น

3.8 <u>การสร้างไฟล์รหัส .MID</u>

จากขั้นตอนการประมวลผลภาพและการจดจำรูปแบบ ทำให้เราทราบจังหวะและระดับเสียงของตัวโน้ตแต่ละตัว เพื่อนำไปเข้ารหัส สร้างเป็นไฟล์รหัสข้อมูลของตัวโน้ต (ไฟล์นามสกุล .SPK) ซึ่งสามารถเล่นเสียงเพลงออกทาง PC Speaker ได้เพื่อตรวจสอบว่าเสียงที่ได้จากการวิเคราะห์ถูกต้องหรือไม่ การติด ต่อกับอุปกรณ์ประเภทมิดี้ต้องใช้ไฟล์มาตรฐานของมิดี้ เช่น (ไฟล์นามสกุล .MID) ดังนั้นเพื่อที่จะใช้เชื่อมต่อกับอุปกรณ์ประเภทมิดี้ได้นั้น เราจะต้องทำการแปลงไฟล์ .SPK ให้เป็นไฟล์ .MID



รูปที่ 3.46 การแปลงไฟล์รหัสข้อมูลตัวโน้ตเป็นไฟล์รหัสมิดี้

การสร้างไฟล์รหัสข้อมูล

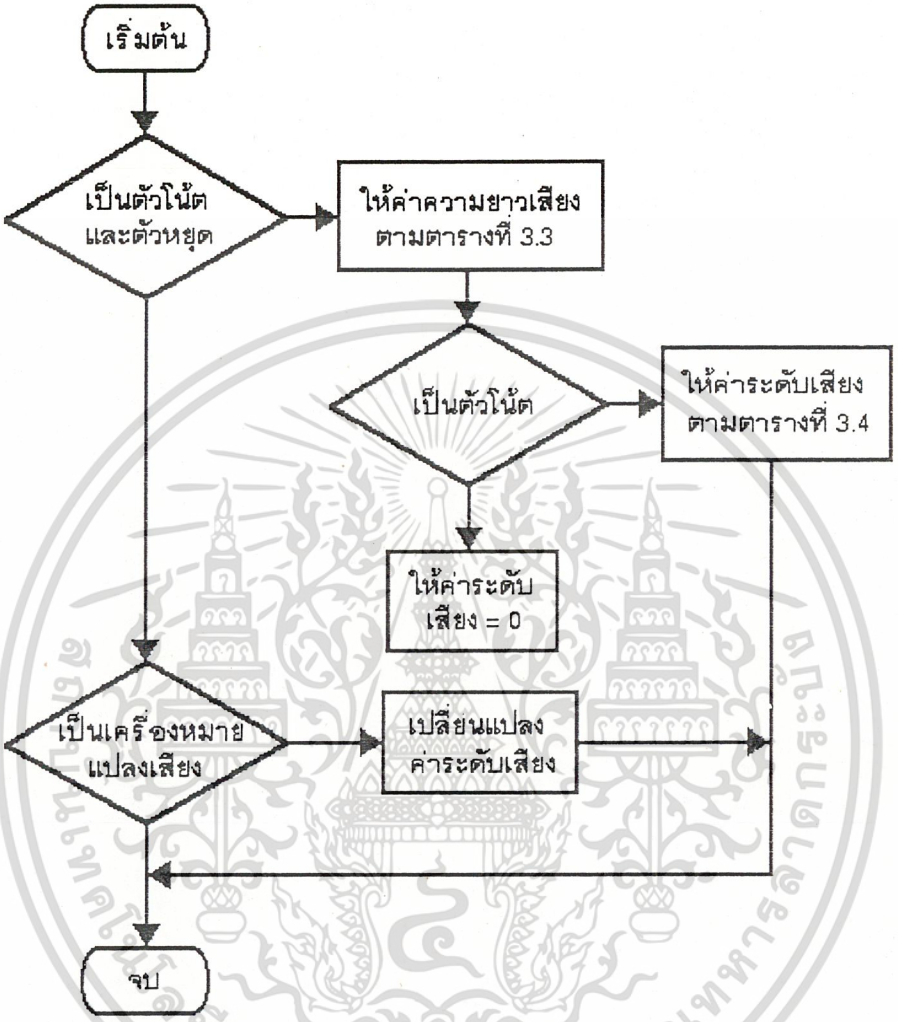
จากขั้นตอนการประมวลผลภาพและการจดจำรูปแบบ ทำให้เราทราบระดับเสียงและความยาวเสียงของตัวโน้ตแต่ละตัว ซึ่งสามารถนำไปสร้างเป็นฐานข้อมูลได้ โดยสร้างเป็นไฟล์รหัสข้อมูลของตัวโน้ต (ไฟล์ที่มีส่วนขยายเป็น .SPK) โดยจะเก็บเป็นเลขฐานสิบหก ซึ่งตัวโน้ตแต่ละตัวจะใช้เนื้อที่ในการเก็บ 4 ไบท์ 2 ไบท์

ตัวโน้ต	ตัวหยุด	ค่าความยาวของเสียง
		32
		16
		8
		4
		2

ตารางที่ 3.3 แสดงค่าความยาวเสียงของตัวโน้ตและตัวหยุด

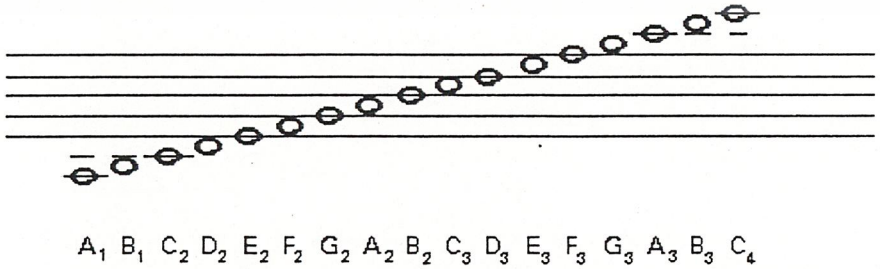
แรกจะบอกค่าระดับของเสียง ส่วน 2 ไบท์หลังจะบอกค่าความยาวของเสียง ตัวอย่างเช่น 19 00 04 00 1B 00 04 00 2 ไบท์แรกคือ 19 00 เมื่อแปลง 19 เป็นเลขฐานสิบจะเท่ากับ 25 เมื่อไปเปรียบเทียบกับตารางที่ 3.4 ได้ระดับเสียงเท่ากับ C_2 2 ไบท์หลังคือ 04 00 เมื่อแปลงเป็นเลขฐานสิบจะเท่ากับ 4 เมื่อไปเปรียบ

เทียบกับตารางที่ 3.3 ได้ตัวโน้ตตัวเขบ็ต 1 ชั้น เพราะฉะนั้นเราจะรู้ว่าโน้ตตัวแรกคือ ตัวเขบ็ต 1 ชั้น มีระดับเสียง C_2



รูปที่ 3.47 ชาร์ตการสร้างไฟล์รหัสข้อมูลตัวโน้ต

ในตัวหยุดนั้นระดับของเสียงจะเท่ากับ 0 ส่วนตัวโน้ตนั้นระดับเสียงจะขึ้นอยู่กับตำแหน่งหัวของตัวโน้ตที่ตกบนบรรทัด 5 เส้นดังนี้



A_1 B_1 C_2 D_2 E_2 F_2 G_2 A_2 B_2 C_3 D_3 E_3 F_3 G_3 A_3 B_3 C_4

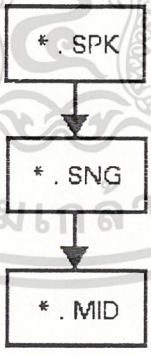
รูปที่ 3.48 แสดงระดับของเสียง

การเก็บระดับของเสียงในไฟล์รหัสข้อมูลจะเก็บเป็นค่าของตัวเลขดังนี้

ระดับของเสียง	ค่าระดับของเสียง	ระดับของเสียง	ค่าระดับของเสียง
A_1	22	C_3	37
B_1	24	D_3	39
C_2	25	E_3	41
D_2	27	F_3	42
E_2	29	G_3	44
F_2	30	A_3	46
G_2	32	B_3	48
A_2	34	C_4	49
B_2	36		

ตารางที่ 3.4 แสดงค่าระดับของเสียงของตัวโน้ตบนบรรทัด 5 เส้น

เมื่อเราได้ไฟล์รหัสข้อมูลตัวโน้ตที่เก็บระดับเสียงและความยาวเสียงของตัวโน้ตแต่ละตัวแล้ว เมื่อต้องการติดต่อกับอุปกรณ์ประเภทมิดี้ เราต้องทำการแปลงไฟล์รหัสข้อมูลตัวโน้ตให้เป็นไฟล์รหัสมิดี้ โดยมีขั้ตอนดังนี้



ไฟล์ .SNG จะเป็นการเตรียมไฟล์รหัสข้อมูลตัวโน้ตก่อนที่จะแปลงเป็นไฟล์รหัส .MID โดยไฟล์ .SNG จะประกอบด้วย header file,massage และ end of file ในตัว header file จะเป็นตัวบอกข้อมูลเกี่ยวกับโน้ตเพลง เช่น จังหวะ,ความยาวของห้องเสียง,ความดังของเสียง,ชนิดของเครื่องดนตรี เป็นต้น massage จะเป็นตัวบอกว่าจะเล่นตัวโน้ตอะไร,เป็นเวลานานเท่าไร,ให้เล่นเสียงของตัวโน้ตออกทาง channel ไหน และ end of file จะเป็นตัวบอกการจบเพลง

ไฟล์รหัส .MID จะมีลักษณะเป็นส่วนๆโดยแต่ละส่วนจะประกอบไปด้วยตัวอักษร 4 ตัว และตามด้วยข้อมูลอีก 32 บิท ซึ่งสามารถแบ่งออกเป็น 2 ประเภท คือ Header Chunks และ Track Chunks ไฟล์รหัสมิดี้จะเริ่มต้นด้วย HEADER chunks และตามด้วย Track Chunks มีลักษณะดังนี้

" Mthd " < length of header data > < header data >

" Mtrk " < length of track data > < track data >

" Mtrk " < length of track data > < track data >

Header Chunks เป็นตัวบอกว่ามี Track Chunks ทั้งหมดกี่ Track ซึ่งมีลักษณะดังนี้

< Header Chunk > = < Chunk type > < length > < format > < ntrks >

โดยที่

- < Chunk type > มีตัวอักษร 4 ตัวคือ " Mthd "
- < length > มีขนาด 32 บิท ใช้บอกความยาวของข้อมูลใน Header Chunk ที่ตามหลัง Chunk type
- < format > เป็นตัวบอก made ของไฟล์รหัสมิดี้
- < ntrks > เป็นตัวบอกจำนวนของ Track Chunk

Track Chunk เป็นตัวเก็บข้อมูลของตัวโน้ตที่จะเก็บ ซึ่งมีจำนวนสูงสุดได้ถึง 16 แชนแนล มีลักษณะดังนี้

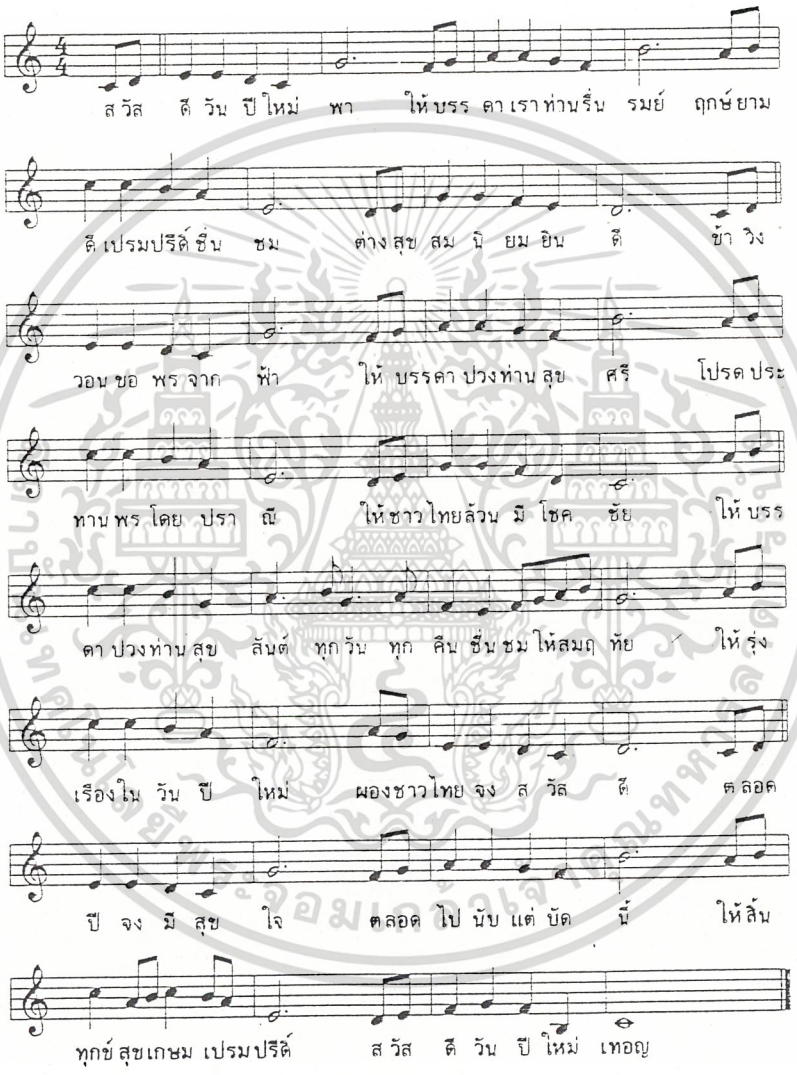
< Track Chunk > = < Chunk type > < length > < Mtrk event > < Mtrk event > .....

โดยที่

- < Chunk type > มีตัวอักษร 4 ตัว คือ " Mtrk "
- < length > มีขนาด 32 บิท ใช้บอกความยาวของข้อมูลใน Track Chunk ที่ตามหลัง Chunk type
- < Mtrk event > จะเป็นตัวบอกรายละเอียดเกี่ยวกับ Track event แต่ละ track

เพลง พรปีใหม่

เพลงพระราชนิพนธ์



รูปที่ 3.49 **โน้ตเพลง** Happy New Year

ตัวอย่างการสร้างไฟล์จากเพลง Happy New Year

จากโน้ตดนตรีในรูปที่ 3.49 เป็นโน้ตดนตรีของเพลง Happy New Year ซึ่งมีกุญแจซอลเป็นกุญแจประจำหลัก มีอัตราจังหวะเป็น 4 : 4 ซึ่งสามารถสร้างเป็นไฟล์รหัสข้อมูลได้ดังนี้

19	00	04	00	1B	00	04	00	-	1D	00	08	00	1D	00	08	00
1B	00	08	00	19	00	08	00	-	20	00	18	00	1E	00	04	00
20	00	04	00	22	00	08	00	-	22	00	08	00	20	00	08	00
1E	00	08	00	24	00	18	00	-	22	00	04	00	24	00	04	00
25	00	08	00	25	00	08	00	-	24	00	08	00	22	00	08	00
1D	00	18	00	1B	00	04	00	-	1D	00	04	00	20	00	08	00
20	00	08	00	1E	00	08	00	-	1D	00	08	00	1D	00	18	00
19	00	04	00	1B	00	04	00	-	1D	00	08	00	1D	00	08	00
1B	00	08	00	19	00	08	00	-	20	00	18	00	1E	00	04	00
20	00	04	00	22	00	08	00	-	22	00	08	00	20	00	08	00
16	00	08	00	24	00	18	00	-	22	00	04	00	24	00	04	00
25	00	08	00	25	00	08	00	-	24	00	08	00	22	00	08	00
1D	00	08	00	1B	00	04	00	-	1D	00	04	00	20	00	08	00
20	00	08	00	1E	00	08	00	-	1B	00	08	00	19	00	18	00
22	00	04	00	24	00	04	00	-	25	00	08	00	25	00	08	00
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..
..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..	..

ตัวอย่างการตีความหมายไฟล์รหัสข้อมูล

19 00 04 00 หมายถึง โน้ตเขบ็ต 1 ชั้น มีระดับเสียง C_2

1B 00 04 00 หมายถึง โน้ตเขบ็ต 1 ชั้น มีระดับเสียง D_2

10 00 08 00 หมายถึง โน้ตตัวดำ มีระดับเสียง E_2

การเปลี่ยนเสียงของเครื่องดนตรีชนิดต่างๆเป็นไปตามตารางที่ 3.5

PROG#	TONE	Pt 1 #
1/00H	AcouPiano 1	4
2/01H	AcouPiano 2	2
3/02H	AcouPiano 3	1
4/03H	ElecPiano 1	3
5/04H	ElecPiano 2	2
6/05H	ElecPiano 3	2
7/06H	ElecPiano 4	1
8/07H	Honkytonk	3
9/08H	Elec Org 1	3
10/09H	Elec Org 2	3
11/0AH	Elec Org 3	2
12/0BH	Elec Org 4	2
13/0CH	Pipe Org 1	3
14/0DH	Pipe Org 2	3
15/0EH	Pipe Org 3	2
16/0FH	Accordion	2
17/10H	Harpsi 1	4
18/11H	Harpsi 2	3
19/12H	Harpsi 3	1
20/13H	Clavi 1	3
21/14H	Clavi 2	2
22/15H	Clavi 3	1
23/16H	Celesta 1	4
24/17H	Celesta 2	2
25/18H	SynBrass 1	2
26/19H	SynBrass 2	3
27/1AH	SynBrass 3	2
28/1BH	SynBrass 4	2
29/1CH	Syn Bass 1	2
30/1DH	Syn Bass 2	2
31/1EH	Syn Bass 3	2
32/1FH	Syn Bass 4	1

PROG #	TONE	Pt 1 #
33/20H	Fantasy	3
34/21H	Harmo Pan	3
35/22H	Chorale	3
36/23H	Glasses	2
37/24H	Soundtrack	4
38/25H	Atmosphere	4
39/26H	Warm Bell	4
40/27H	Funny Vox	1
41/28H	Echo Bell	3
42/29H	Ice Rain	3
43/2AH	Oboe 2001	2
44/2BH	Echo Pan	2
45/2CH	DoctorSolo	2
46/2DH	Schooldaze	2
47/2EH	Bellsinger	1
48/2FH	SquareWave	2
49/30H	Str Sect 1	4
50/31H	Str Sect 2	3
51/32H	Str Sect 3	2
52/33H	Pizzicato	3
53/34H	Violin 1	3
54/35H	Violin 2	2
55/36H	Cello 1	3
56/37H	Cello 2	2
57/38H	Contrabass	2
58/39H	Harp 1	3
59/3AH	Harp 2	2
60/3BH	Guitar 1	2
61/3CH	Guitar 2	2
62/3DH	Elec Gtr 1	4
63/3EH	Elec Gtr 2	3
64/3FH	Sitar	4

PROG #	TONE	Pt 1 #
65/40H	AcouBass 1	2
66/41H	AcouBass 2	1
67/42H	ElecBass 1	2
68/43H	ElecBass 2	1
69/44H	SlapBass 1	3
70/45H	SlapBass 2	2
71/46H	Fretless 1	4
72/47H	Fretless 2	2
73/48H	Flute 1	4
74/49H	Flute 2	2
75/4AH	Piccolo 1	3
76/4BH	Piccolo 2	2
77/4CH	Recorder	2
78/4DH	Pan Pipes	3
79/4EH	Sax 1	4
80/4FH	Sax 2	3
81/50H	Sax 3	2
82/51H	Sax 4	1
83/52H	Clarinet 1	3
84/53H	Clarinet 2	2
85/54H	Oboe	2
86/55H	Engl Horn	2
87/56H	Bassoon	2
88/57H	Harmonica	2
89/58H	Trumpet 1	3
90/59H	Trumpet 2	2
91/5AH	Trombone 1	3
92/5BH	Trombone 2	2
93/5CH	Fr Horn 1	3
94/5DH	Fr Horn 2	2
95/5EH	Tuba	2
96/5FH	Brs Sect 1	4

PROG #	TONE	Pt 1 #
97/60H	Brs Sect 2	3
98/61H	Vibe 1	3
99/62H	Vibe 2	2
100/63H	Syn Mallet	1
101/64H	Windbell	3
102/65H	Glock	2
103/66H	Tube Bell	4
104/67H	Xylophone	1
105/68H	Marimba	3
106/69H	Koto	2
107/6AH	Sho	4
108/6BH	Shakuhachi	4
109/6CH	Whistle 1	2
110/6DH	Whistle 2	1
111/6EH	Bottleblow	4
112/6FH	Breathpipe	3
113/70H	Timpani	2
114/71H	MelodicTom	1
115/72H	Deep Snare	2
116/73H	ElecPerc 1	2
117/74H	ElecPerc 2	2
118/75H	Taiko	3
119/76H	Taiko Rim	1
120/77H	Cymbal	2
121/78H	Castanets	2
122/79H	Triangle	2
123/7AH	Orche Hit	4
124/7BH	Telephone	1
125/7CH	Bird Tweet	4
126/7DH	OneNoteJam	4
127/7EH	WaterBells	3
128/7FH	JungleTune	4

PROG # : MIDI Program Change Number (decimal indication / hexadecimal indication).
Ptl # : The number of partials used for a sound.

ตารางที่ 3.5 การสังเคราะห์เสียงของ LA SOUND MODULE CM-32

บทที่ 4
การทดลองและผลที่ได้

4.1 ส่วนที่ติดต่อกับผู้ใช้

โปรแกรมในส่วนติดต่อกับผู้ใช้ ได้เขียนเป็นโหมดกราฟฟิกเก็บไว้ในไฟล์ชื่อ THESIS8.BMP มีปุ่มใช้งานทั้งหมด 6 ปุ่มซึ่งแต่ละปุ่มมีหน้าที่การทำงานดังนี้

1. ปุ่ม Open จะแสดงรายชื่อไฟล์ที่มีส่วนขยายเป็น .BMP ทั้งหมดที่มีอยู่ใน Directory ปัจจุบัน จากนั้นเราจึงใส่ชื่อของไฟล์และส่วนขยายที่เราต้องการเปิด โปรแกรมจะแสดงผลภาพที่เราจะเปิด
2. ปุ่ม Step จะเป็นขั้นตอนการประมวลผลภาพและการจดจำตัวพิมพ์โน้ต โดยจะแสดงการทำงานของโปรแกรมให้เห็นทีละขั้นตอน ตั้งแต่การแยกตัวโน้ตออกจากบรรทัด 5 เส้น การตีกรอบรอบตัวโน้ตแต่ละตัวเพื่อแยกให้ตัวโน้ตเป็นอิสระต่อกัน การทำให้บางเพื่อหาโครงร่างของตัวโน้ต การหาลักษณะเด่นของตัวโน้ตแต่ละตัวโดยวิธีหาจุดตัด จุดแยก จุดปลาย ในแต่ละส่วนของตัวโน้ต และเก็บเสียงของตัวโน้ตที่ได้จากการวิเคราะห์
3. ปุ่ม Save จะเป็นการเก็บผลที่ได้จากขั้นตอนการประมวลผลภาพและการจดจำรูปแบบตัวพิมพ์โน้ตไว้เป็นฐานข้อมูล โดยโปรแกรมจะให้ใส่ชื่อไฟล์ที่ต้องการเก็บโดยมีส่วนขยายเป็น .SPK ถ้าชื่อของนี้มีอยู่แล้วก็จะเป็นการนำเอาผลที่ได้ไปเขียนต่อท้ายไฟล์เก่า แต่ถ้ายังไม่เคยมีชื่อไฟล์นั้นอยู่เลย จะเป็นการเปิดไฟล์ใหม่เพื่อเก็บผลที่ได้
4. ปุ่ม Convert จะเป็นปุ่มที่ใช้แปลงไฟล์รหัสตัวโน้ตที่มีส่วนขยายเป็น .SPK ให้เป็นไฟล์รหัสมิดี้ที่มีส่วนขยายเป็น .MID โดยโปรแกรมจะให้ใส่ชื่อไฟล์ที่มีส่วนขยายเป็น .SPK ที่จะต้องการ Convert จากนั้นจะให้ใส่หมายเลข Program ของเครื่องดนตรีที่ต้องการ (ดูจากตารางการสังเคราะห์เสียง) ซึ่งเมื่อทำการ Convert แล้วจะได้ไฟล์ที่มีส่วนขยายเป็น .SNG และ .MID
5. ปุ่ม Play จะเป็นการเล่นเสียงเพลงตามไฟล์รหัสตัวโน้ตที่มีส่วนขยายเป็น .SPK ออกทาง PC Speaker โดยโปรแกรมจะแสดงชื่อไฟล์ที่มีส่วนขยายเป็น .SPK ทั้งหมดที่มีอยู่ใน Directory ปัจจุบัน จากนั้นจะให้ใส่ชื่อไฟล์และส่วนขยายของไฟล์ที่เราต้องการเล่นออกทาง PC Speaker แล้วเล่นเพลงออกทาง PC Speaker
6. ปุ่ม Exit ใช้สำหรับออกจากโปรแกรม

จากโปรแกรมข้างต้นจะทำให้เราได้ไฟล์รหัสมิดี้ที่มีส่วนขยายเป็น .MID ซึ่งสามารถเล่นเสียงเพลงได้ โดยคณะผู้จัดทำได้เขียนโปรแกรมสำหรับเล่นไฟล์รหัสมิดี้ผ่าน Sound Blaster ซึ่งในขั้นแรกจะต้อง Load Driver ของ Sound Blaster ก่อน โดยแยกไฟล์ SBMIDI และ SBSIM จากนั้นก็แยกโปรแกรม Play MIDI ตามด้วยชื่อไฟล์รหัสมิดี้ที่ต้องการเล่น เช่น Playmidi Happy.mid โปรแกรมก็จะเล่นไฟล์รหัสมิดี้ออกมาเป็นเสียงเพลง

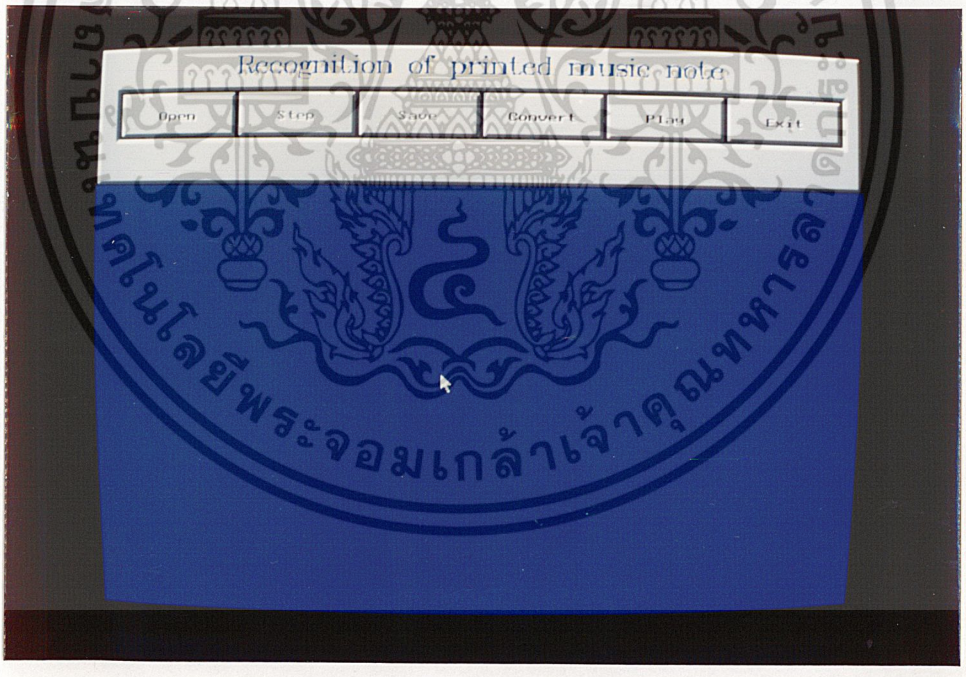
เนื่องจากคณะผู้จัดทำขาดประสบการณ์ในด้านการเขียนโปรแกรมการแสดงภาพ .BMP ออกทางจอภาพ ทำให้การแสดงผลภาพของตัวโน้ต ไม่สามารถแสดงภาพทั้งหมดที่เก็บจากสแกนเนอร์ได้ จึงต้องเก็บภาพของตัวโน้ตเป็นส่วนๆ ตัวอย่างเช่น ในเพลง Happy New Year จะต้องมีการเก็บภาพออกเป็น 8 ส่วน เมื่อจะทำการวิเคราห์ตัวโน้ตจะมีขั้นตอนดังนี้

1. เรียกไฟล์ THESIS8.BMP
2. แสดงภาพตัวโน้ตโดยกดปุ่ม Open แล้วใส่ชื่อไฟล์ HAPPY1.BMP

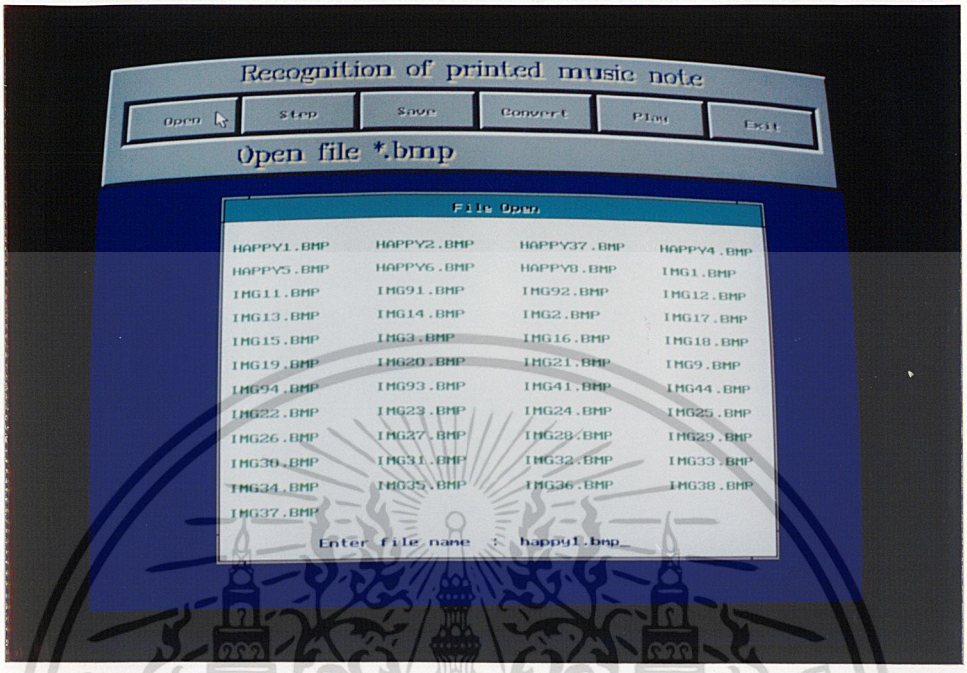
3. ทำการวิเคราะห์ตัวโน้ตโดยกดปุ่ม Step จนได้ผลการวิเคราะห์
4. เก็บผลการวิเคราะห์ตัวโน้ตโดยกดปุ่ม Save แล้วใส่ชื่อไฟล์ที่ต้องการ Save เช่น HAPPY.SPK
5. แสดงภาพในส่วนที่ 2,3,4,5,6,7 และ 8 โดยใส่ชื่อไฟล์ HAPPY2.BMP , HAPPY37.BMP , HAPPY4.BMP , HAPPY5.BMP , HAPPY6.BMP , HAPPY37.BMP และ HAPPY8.BMP ตามลำดับ
6. ทำการวิเคราะห์ตัวโน้ตในแต่ละส่วนและเก็บผลการทดลองที่ได้ในไฟล์ HAPPY.SPK
7. เมื่อได้ไฟล์รหัสข้อมูลตัวโน้ตครบแล้ว ถ้าต้องการแปลงเป็นไฟล์รหัสมิดี้ให้กดปุ่ม Convert ใส่ชื่อไฟล์รหัสข้อมูลตัวโน้ตที่จะทำการแปลง (HAPPY.SPK) ก็จะได้ไฟล์รหัสมิดี้ (HAPPY.MID)
8. ใส่หมายเลขโปรแกรมของเครื่องคนตรีตามตารางที่ 3.5

4.2 ผลการทดลองการวิเคราะห์ตัวโน้ต

การประมวลผลและการวิเคราะห์ตัวโน้ตในขั้นตอนการหาลักษณะเด่นของตัวโน้ต โดยหาจำนวนและตำแหน่งของจุดแยก,จุดตัด และจุดปลาย ในโครงงานนี้สามารถวิเคราะห์ตัวโน้ตและสัญลักษณ์ทางดนตรีรวม 32 ตัว ซึ่งโน้ตเพลงที่วิเคราห์ได้ไม่ซับซ้อนมากนัก แสดงดังรูป

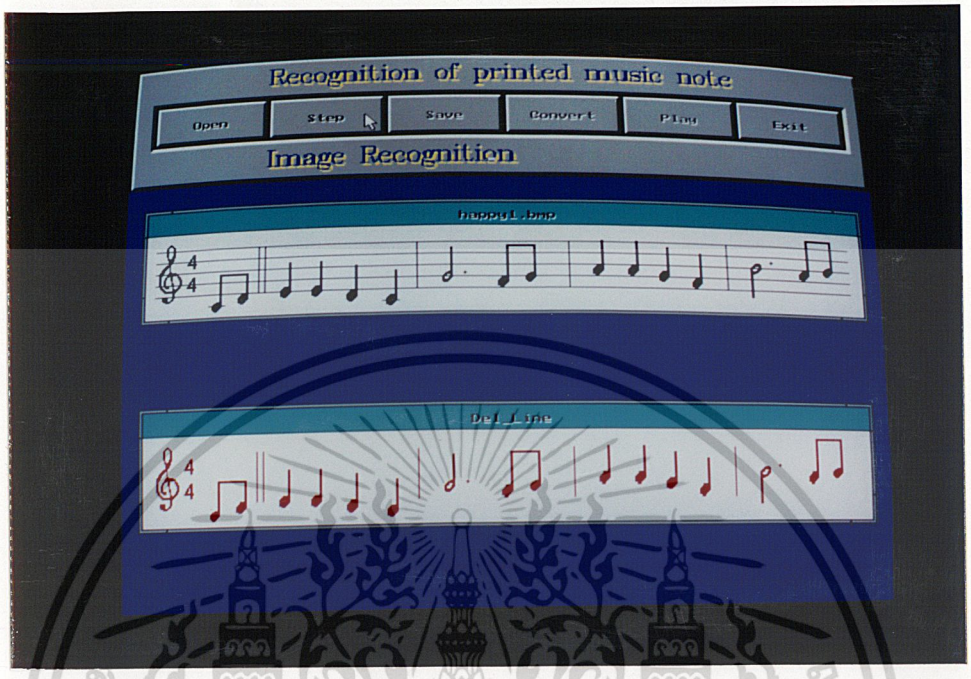


รูปที่ 4.1 ภาพแสดงส่วนติดต่อกับผู้ใช้



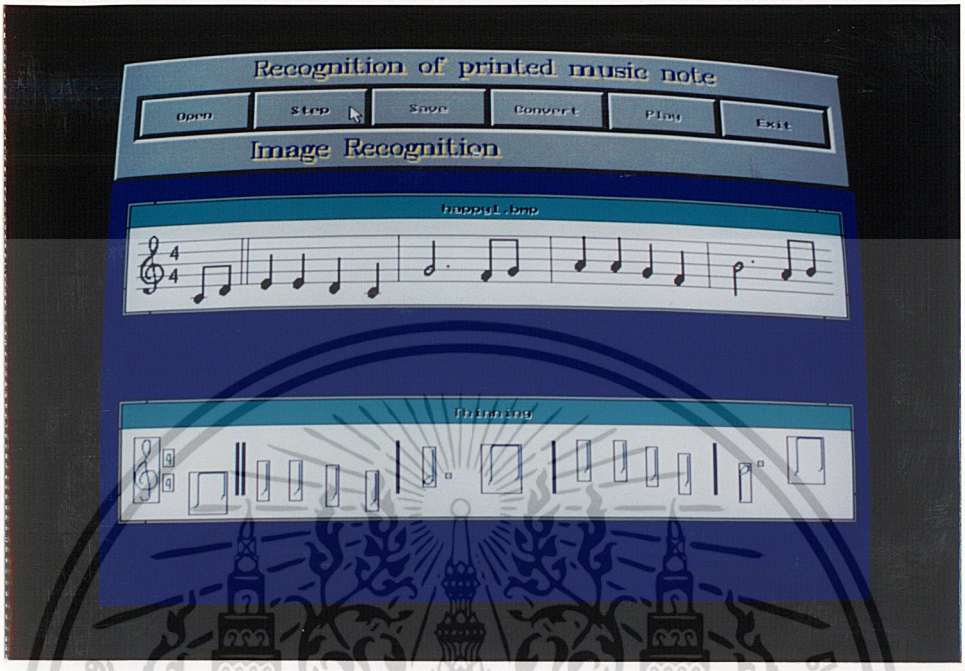
รูปที่ 4.2 ภาพแสดงการใส่ชื่อไฟล์ภาพตัวโน้ตที่จะทำการวิเคราะห์

รูปที่ 4.3 ภาพแสดงตัวโน้ตที่จะนำมาวิเคราะห์



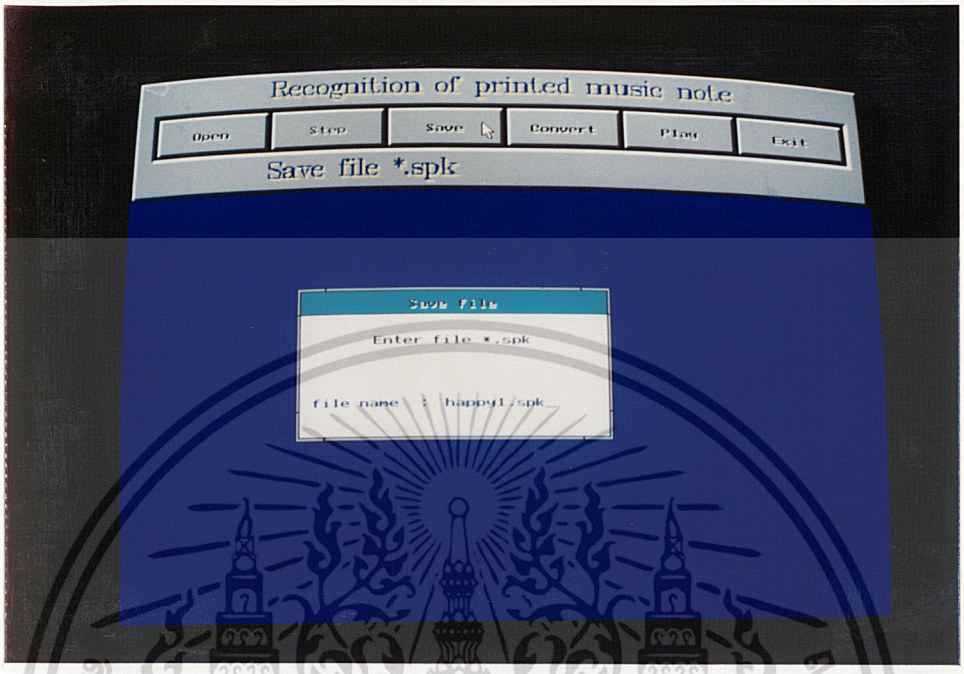
รูปที่ 4.4 ภาพแสดงการแยกตัวโน้ตออกจากบรรทัด 5 เส้น

รูปที่ 4.5 ภาพแสดงการตีกรอบรอบตัวโน้ต

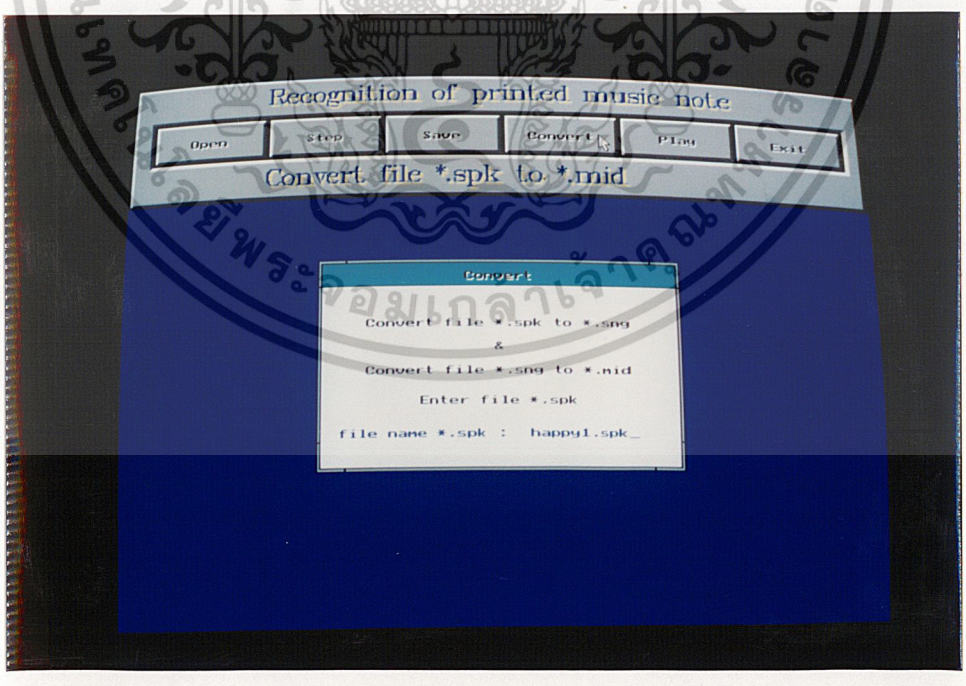


รูปที่ 4.6 ภาพแสดงการทำตัวโน้ตให้บาง

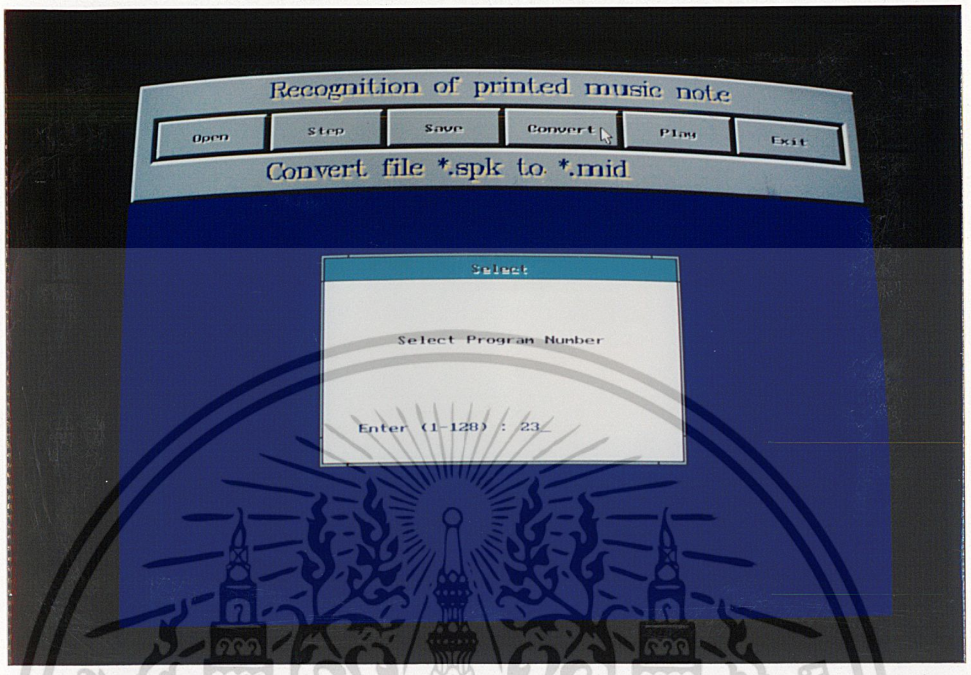
รูปที่ 4.7 ภาพแสดงการหาลักษณะเด่นของตัวโน้ต



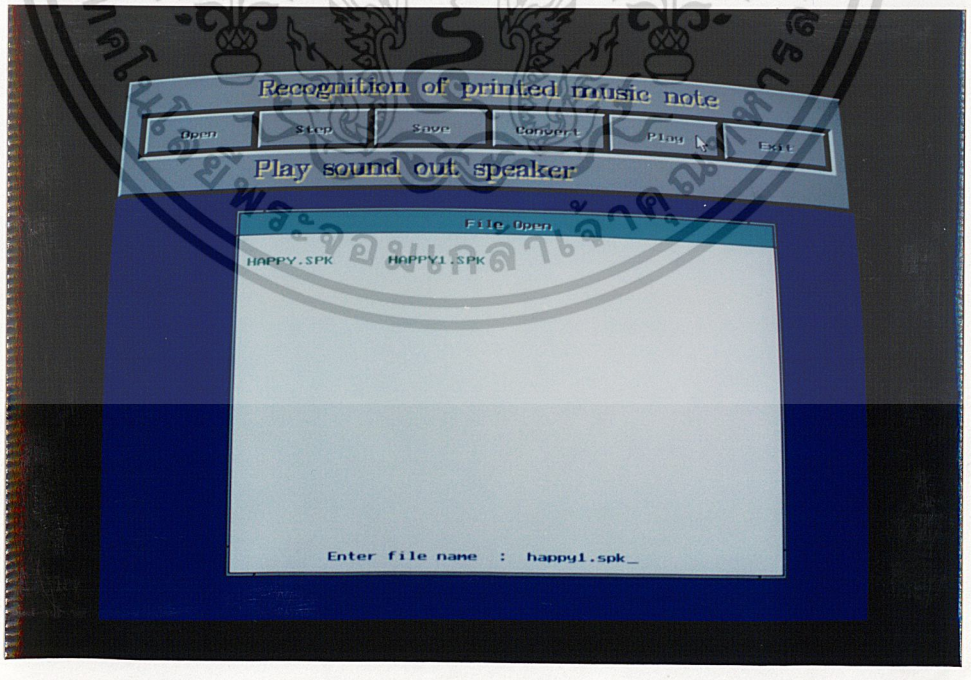
รูปที่ 4.8 ภาพแสดงการการเก็บผลการวิเคราะห์โดย Save ไว้ในไฟล์ชื่อ *.SPK



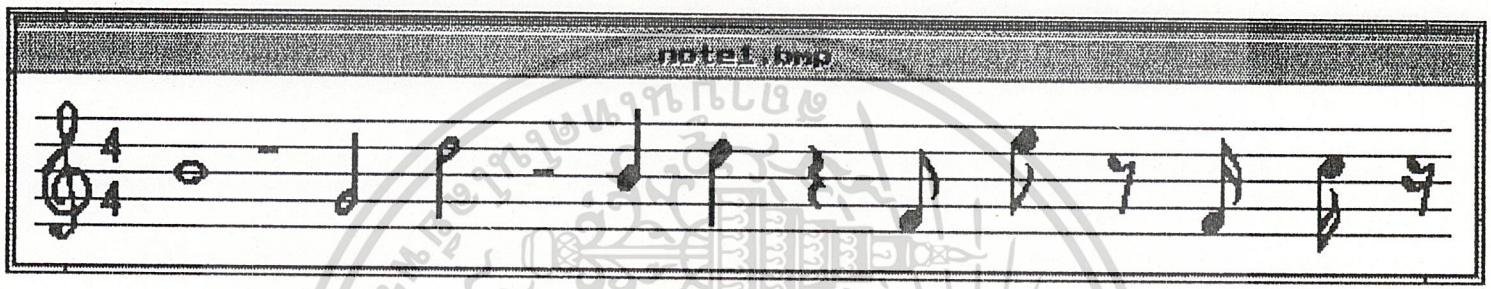
ภาพที่ 4.9 ภาพแสดงการแปลงไฟล์ให้เป็นไฟล์รหัสมิดี้โดยการกดปุ่ม Convert เพื่อแปลงไฟล์ *.SPK ให้เป็น *.MID



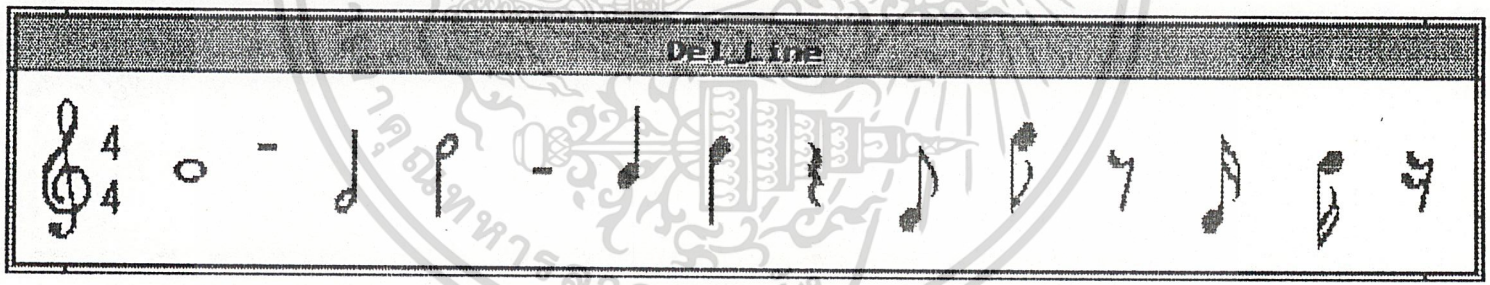
รูปที่ 4.10 ภาพแสดงการใส่หมายเลขโปรแกรมของเครื่องดนตรีตามตารางที่ 3.5



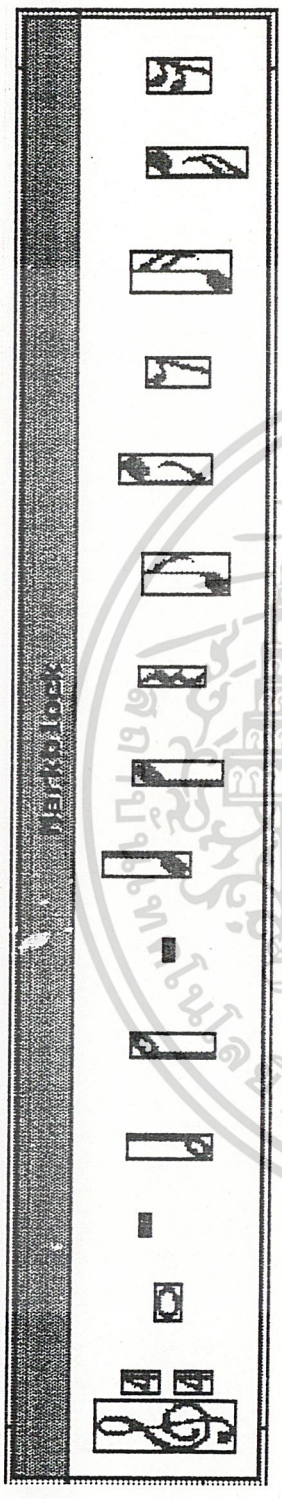
รูปที่ 4.11 ภาพแสดงการใส่ชื่อไฟล์ *.SPK ที่ต้องการเล่นเป็นเสียงเพลง



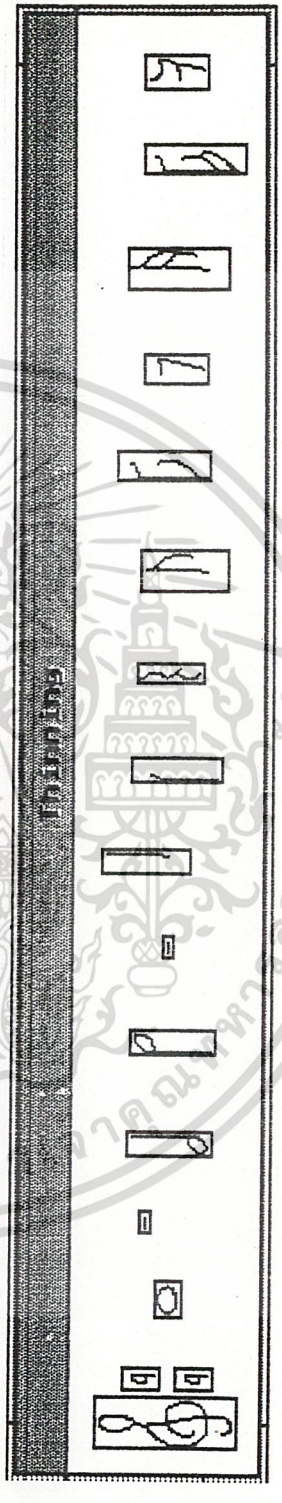
รูปที่ 4.12 ภาพแสดงตัวโน้ตที่จะนำมาวิเคราะห์



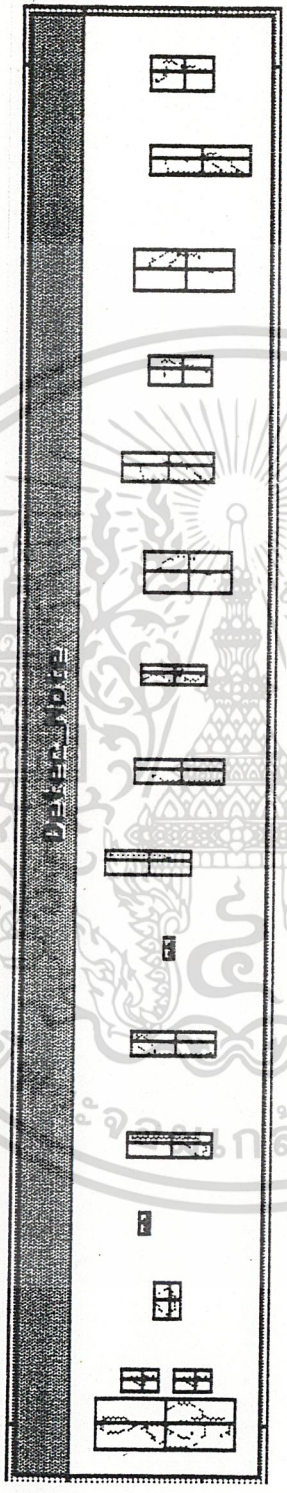
รูปที่ 4.13 ภาพแสดงการแยกตัวโน้ตออกจากบรรทัด 5 เส้น



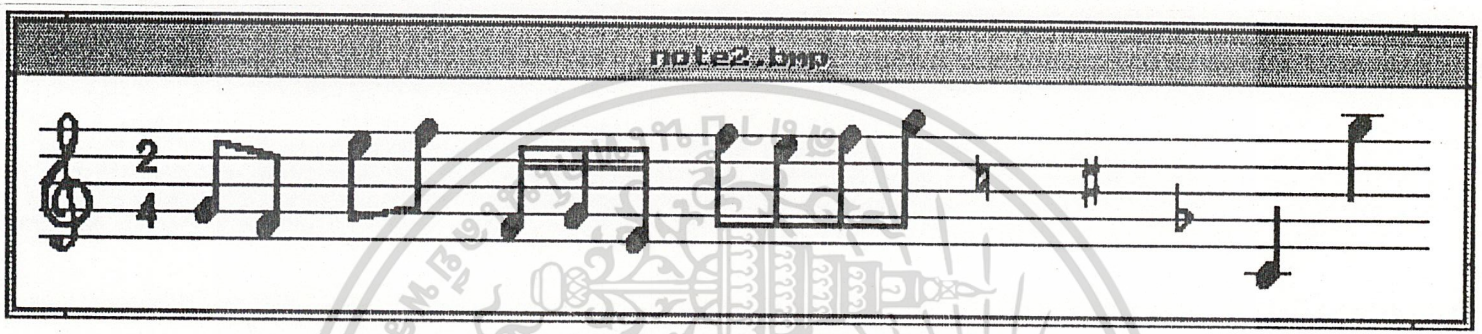
ภาพที่ 4.14 ภาพแสดงการตีกรอบปรอบตัวโน้ต



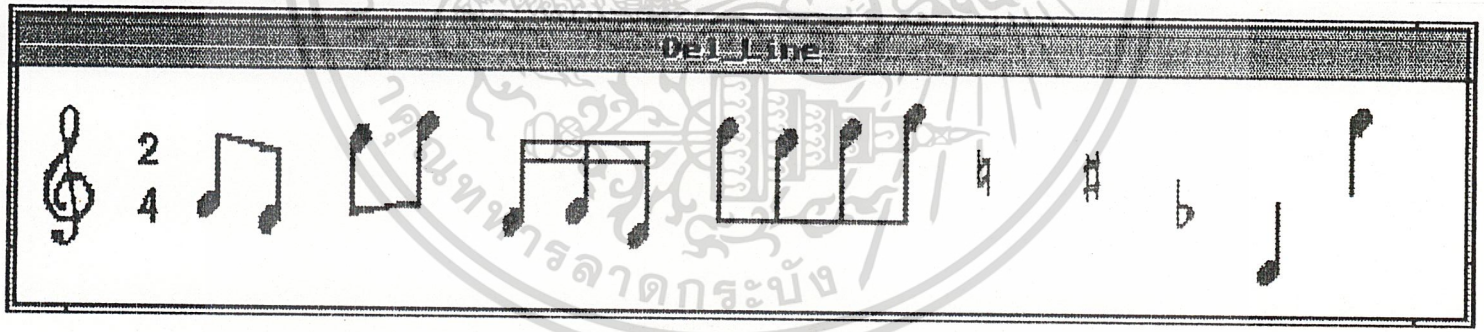
ภาพที่ 4.15 ภาพแสดงการทำตัวโน้ตให้บาง



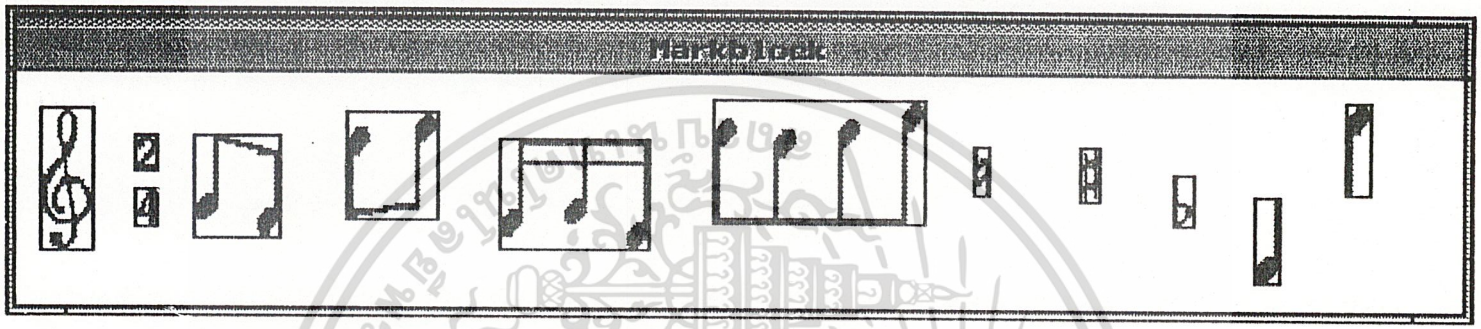
รูปที่ 4.16 ภาพแสดงการนำลักษณะเด่นของตัวโป๊ะ



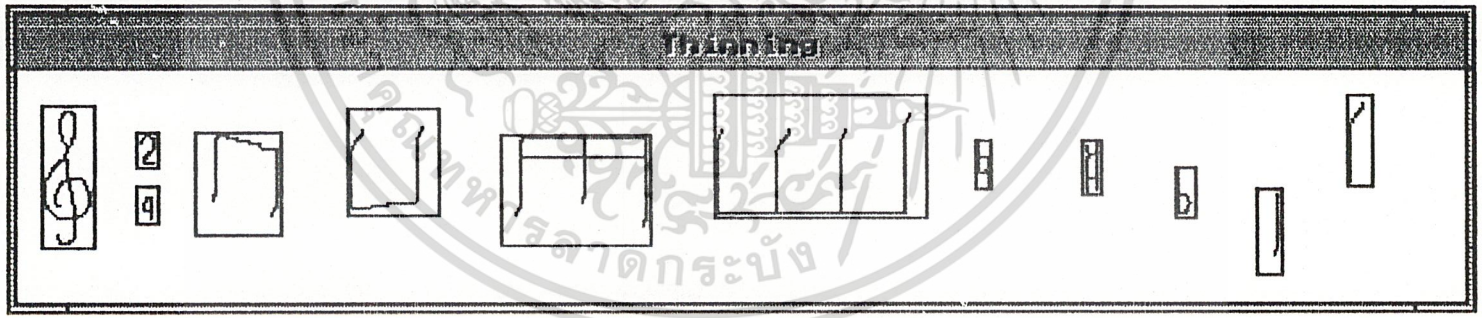
รูปที่ 4.17 ภาพแสดงตัวโน้ตที่จะนำมาวิเคราะห์



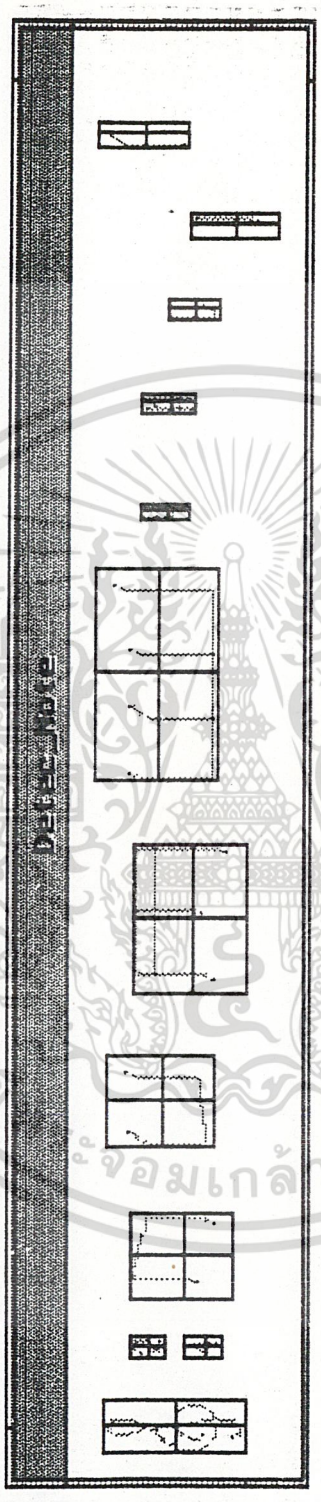
รูปที่ 4.18 ภาพแสดงการแยกตัวโน้ตออกจากบรรทัด 5 เส้น



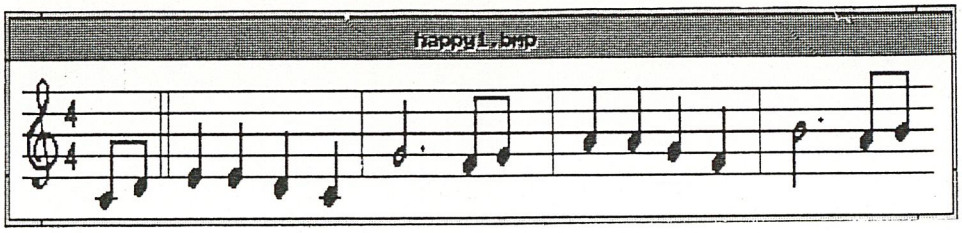
รูปที่ 4.19 ภาพแสดงการตีกรอบรอบตัวโน้ต

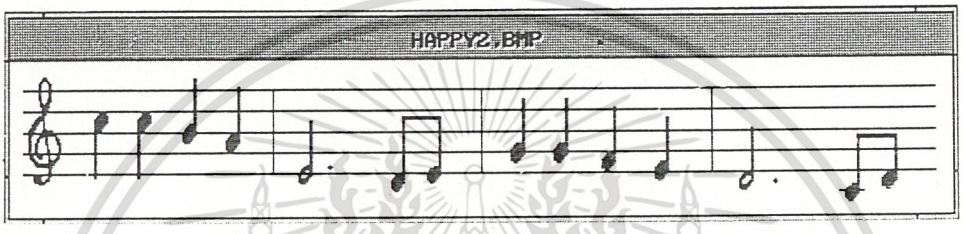


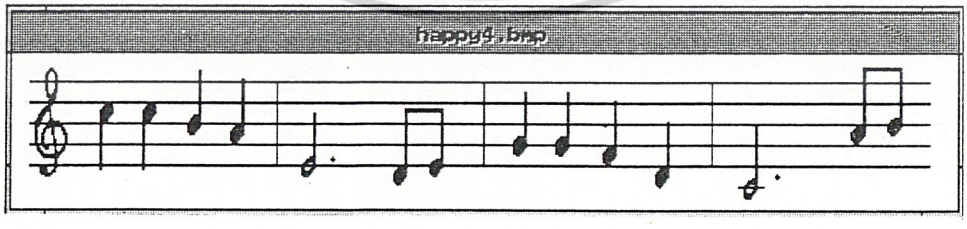
รูปที่ 4.20 ภาพแสดงการทำตัวโน้ตให้บาง



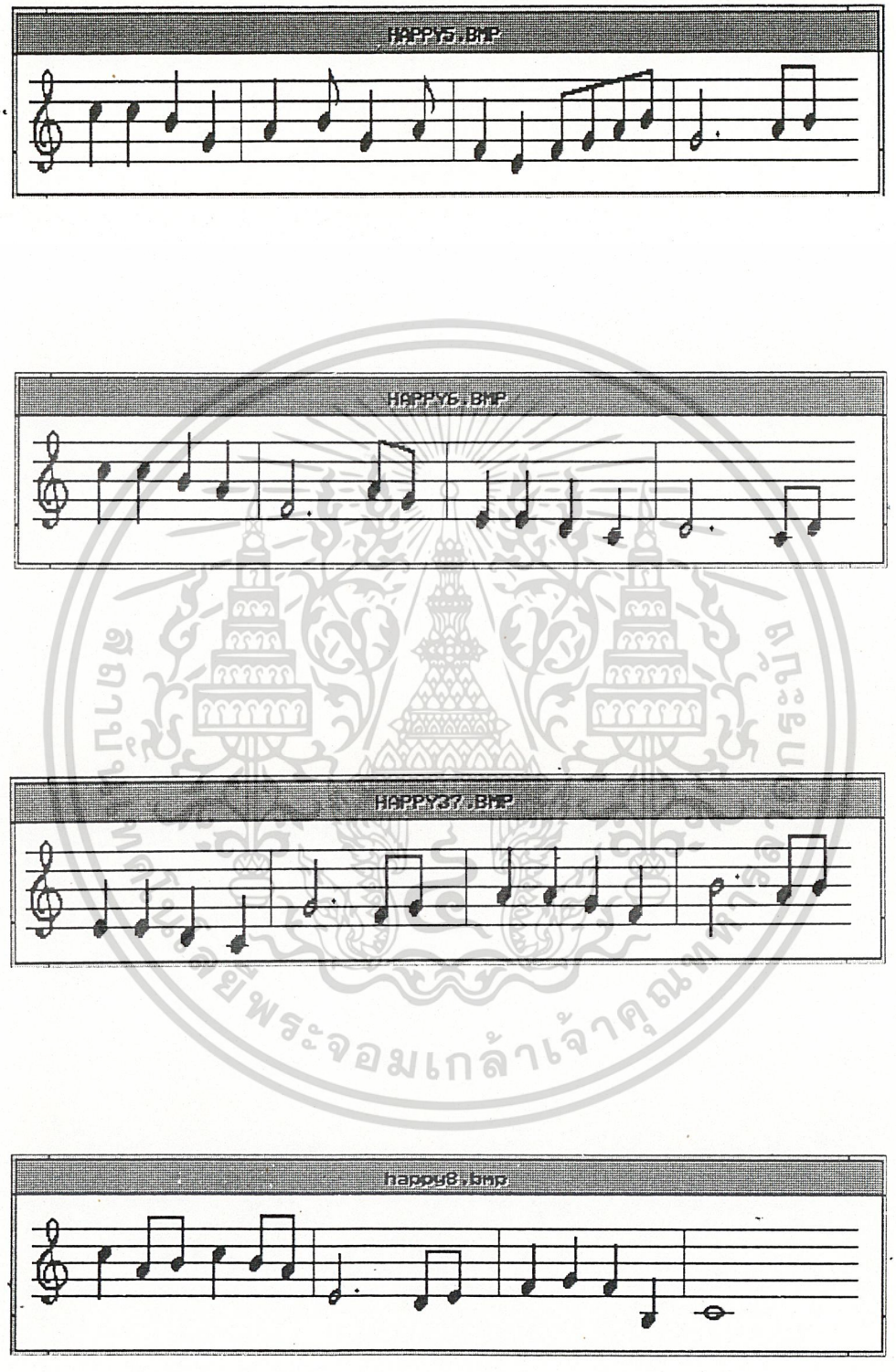
รูปที่ 4.21 ภาพแสดงการหักล้างระนาบด้านหน้าของตัวโมเดล



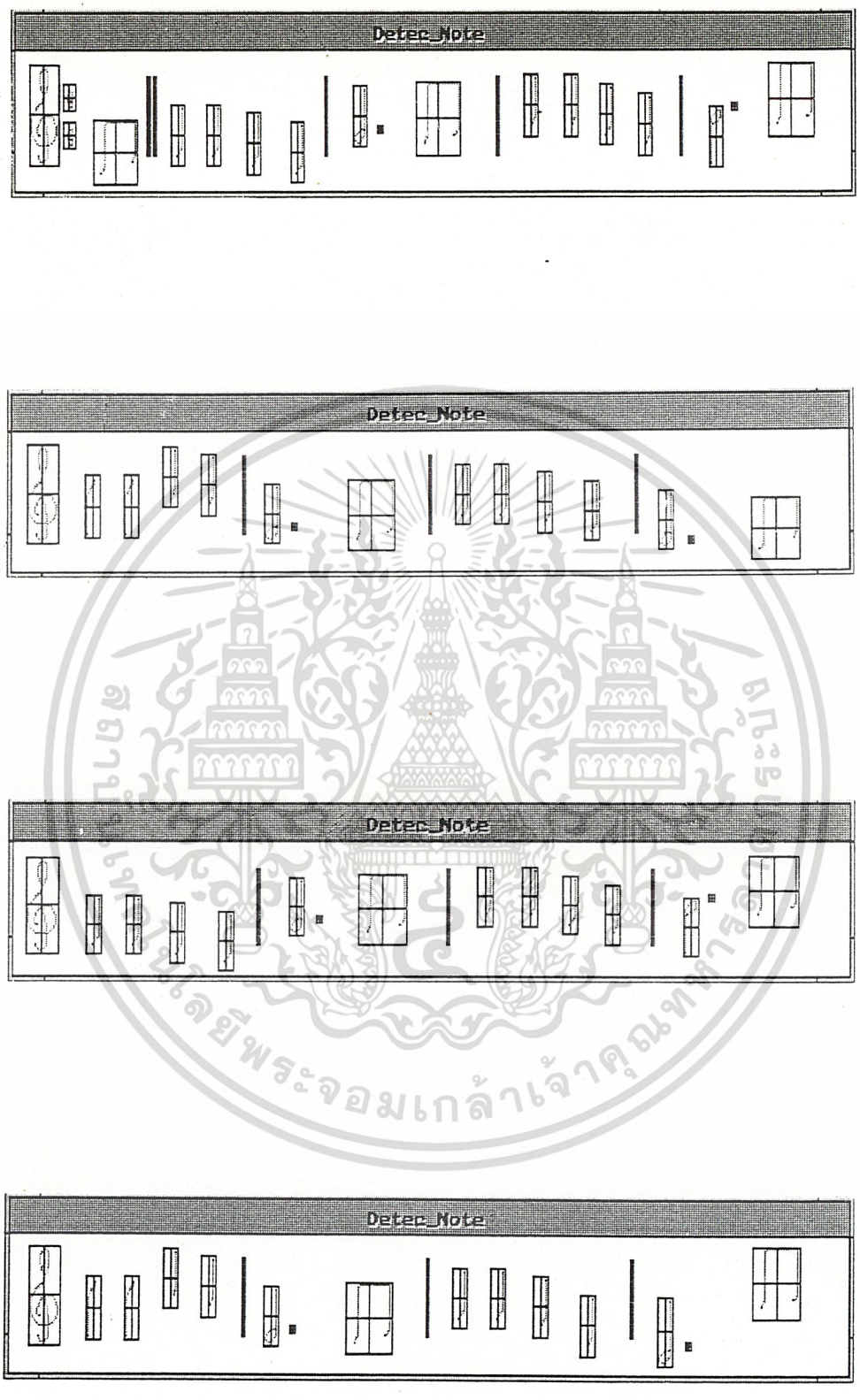




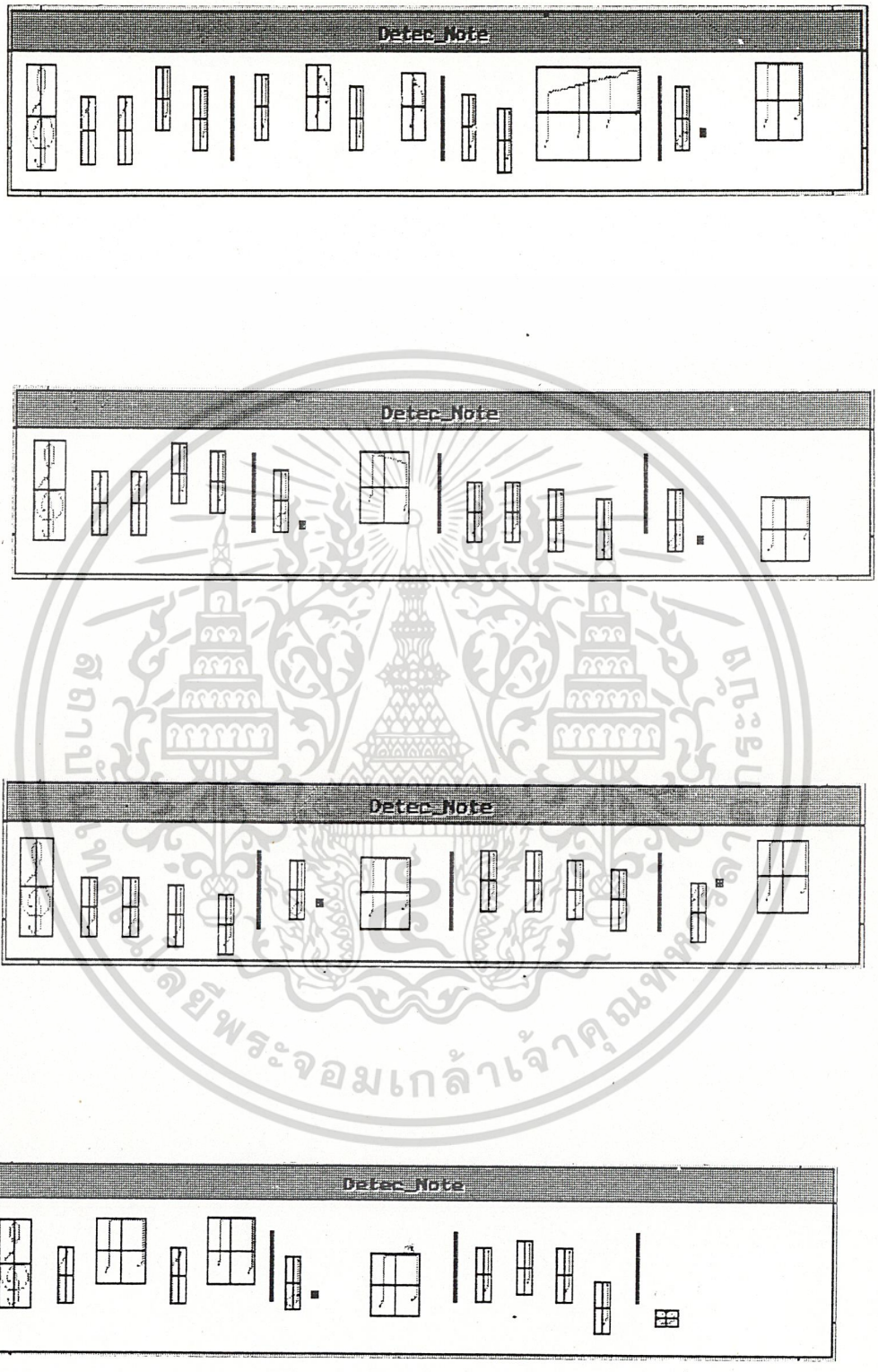
รูปที่ 4.22-1 ภาพแสดงตัวโน้ตเพลง Happy New Year ก่อนทำการวิเคราะห์



รูปที่ 4.22-2 ภาพแสดงตัวโน้ตเพลง Happy New Year ก่อนทำการวิเคราะห์



รูปที่ 4.23-1 ภาพแสดงตัวโน้ตเพลง Happy New Year หลังทำการวิเคราะห์



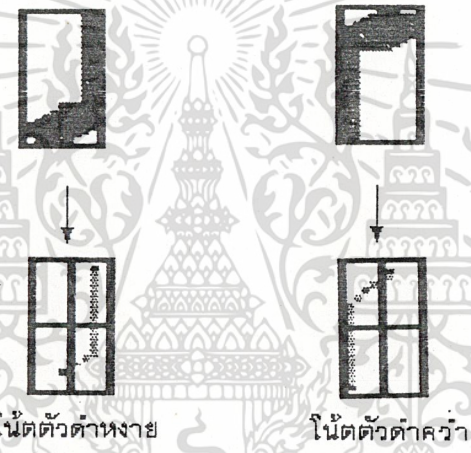
รูปที่ 4.23-2 ภาพแสดงตัวโน้ตเพลง Happy New Year หลังทำการวิเคราะห์

บทที่ 5
สรุปและวิจารณ์ผลการทดลอง

5.1 ปัญหาที่พบและการแก้ไข

จากการทดลองวิเคราะห์ตัวโน้ตในช่วงแรกๆ โน้ตที่วิเคราะห์ได้ยังไม่ถูกต้อง กล่าวคือมีโน้ตบางตัวที่ เมื่อวิเคราะห์ออกมาแล้ว ได้จังหวะและระดับเสียงผิดไป เช่นโน้ตตัวดำและตัวเขบ็ตแบบกลุ่ม

ปัญหาที่เกิดกับโน้ตตัวดำได้แก่ โน้ตตัวดำคว่ำและโน้ตตัวดำหงาย เมื่อหาลักษณะเด่นของตัวโน้ตโดยวิธีพิจารณาจำนวนและตำแหน่งของจุดตัด,จุดแยก และจุดปลายนั้น ผลที่ได้ออกมาเหมือนกัน คือจะมีจุดปลายในส่วนที่ 2,3 จำนวนเท่ากัน ดังรูป 5.1



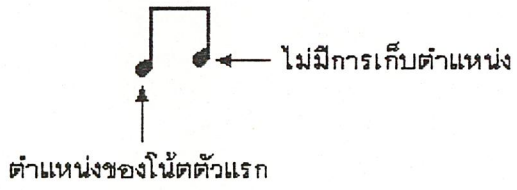
รูปที่ 5.1 ปัญหาที่เกิดกับโน้ตตัวดำ

จากการวิเคราะห์ คอมพิวเตอร์จะมองเห็นว่าเป็นตัวโน้ตตัวเดียวกัน ดังนั้นเราจึงต้องตั้งเงื่อนไขเพิ่มขึ้นอีก โดยพิจารณาในส่วนที่ 4 ว่ามีจุดของภาพอยู่ในส่วนที่ 4 หรือไม่ ถ้ามีจุดภาพอยู่ในส่วนที่ 4 ก็แสดงว่าโน้ตตัวนั้นเป็นโน้ตตัวดำหงาย ถ้าไม่มีแสดงว่าเป็นโน้ตตัวดำคว่ำ

ปัญหาที่เกิดกับโน้ตตัวเขบ็ตแบบกลุ่มได้แก่ เมื่อมีตัวเขบ็ตตั้งแต่ 2 ตัวขึ้นไปติดกันจะทำให้ระดับเสียงของตัวโน้ตผิดไป

รูปที่ 5.2 โน้ตตัวเขบ็ตที่ได้จากการวิเคราะห์ที่ผิดพลาด

เหตุที่เป็นเช่นนั้นเพราะในขั้นตอนการตีกรอบจะไม่ได้เก็บตำแหน่งของตัวโน้ตตัวที่ 2 ไว้

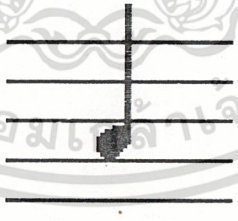


รูปที่ 5.3 ปัญหาที่เกิดกับโน้ตตัวเขบ็ต

ดังนั้นเราจึงต้องหาตำแหน่งของตัวโน้ต เองโดยการประมาณค่าจากจุดปลายของตัวโน้ต ที่ผ่านกระบวนการหาลักษณะเด่นมาแล้ว เปรียบกับบรรทัด 5 เส้น

รูปที่ 5.4 ตำแหน่งบนบรรทัด 5 เส้น

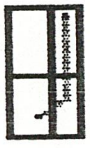
ปัญหาในขั้นตอนการแยกบรรทัด 5 เส้นออกจากตัวโน้ต ทำให้รูปร่างของตัวโน้ตผิดไปเล็กน้อยดังรูป



รูปที่ 5.5 ตัวโน้ตก่อนที่จะแยกออกจากบรรทัด 5 เส้น

รูปที่ 5.6 ตัวโน้ตที่แยกออกจากบรรทัด 5 เส้นแล้ว

เมื่อนำตัวโน้ตที่แยกออกจากบรรทัด 5 เส้นแล้วไปหาโครงร่างของตัวโน้ตโดยวิธีการทำให้บาง จะทำให้ได้โครงร่างผิดไป ซึ่งจะมีผลต่อการจดจำรูปแบบ



รูปที่ 5.7 โครงร่างของตัวโน้ตที่ต้องการ

รูปที่ 5.8 โครงร่างที่ผิดไปอันเนื่องมาจากขั้นตอนการลบบรรทัด 5 เส้น

ปัญหาอีกอย่างหนึ่งก็คือ การนำภาพตัวพิมพ์โน้ตมาทำการประมวลผลภาพ นั้นยังไม่สามารถทำได้กับตัวโน้ตหรือสัญลักษณ์ทางดนตรีที่มีอยู่ได้ทั้งหมด โครงงานฉบับนี้จึงทำเฉพาะตัวโน้ตที่สำคัญและใช้กันบ่อยๆ เท่านั้น สำหรับการที่จะทำการประมวลผลและจดจำรูปแบบตัวพิมพ์โน้ตและสัญลักษณ์ได้หมดทุกตัวนั้น จะต้องมีการพัฒนาต่อไป

โปรแกรมในส่วนที่ติดต่อกับผู้ใช้ จะเป็นการแสดงให้เห็นถึงการทำงานในแต่ละขั้นตอน โดยจะแสดงขั้นตอนทีละ step ซึ่งในการนำไปใช้งานจริงๆ ไม่จำเป็นต้องแสดงให้เห็นการทำงานของทุกขั้นตอนก็ได้

5.2 <u>แนวทางการพัฒนาต่อ</u>

1) พัฒนาโปรแกรมส่วนติดต่อกับผู้ใช้ให้สามารถใช้ได้สะดวกยิ่งขึ้น เช่น สามารถประมวลผลภาพและจดจำรูปแบบตัวพิมพ์โน้ตได้ทีเดียวทั้งหมดทั้งแผ่นภาพตัวโน้ต โดยไม่ต้องมาตัดแบ่งเป็นส่วนๆ

2) ขั้นตอนการแยกตัวโน้ตออกจากบรรทัด 5 เส้น ถ้ามีการปรับปรุงให้สามารถแยกตัวโน้ตออกจากบรรทัด 5 เส้นได้โดยที่รูปร่างของตัวโน้ตไม่มีส่วนเกินเพิ่มขึ้นมา จะทำให้การประมวลผลภาพและการจดจำรูปแบบตัวพิมพ์โน้ตมีประสิทธิภาพยิ่งขึ้น

3) การแบ่งกลุ่มของตัวโน้ตยังไม่ดีเท่าที่ควร การแบ่งกลุ่มตัวโน้ตที่ดี จะช่วยลดเวลาในการทำงานของโปรแกรมและทำให้ทำงานได้ถูกต้องแม่นยำยิ่งขึ้น

4) ตัวโน้ตที่สามารถวิเคราะห์ได้ เป็นเพียงบางส่วนเท่านั้น จึงควรมีการพัฒนาต่อ เช่น ให้โปรแกรมสามารถวิเคราะห์ตัวโน้ตที่เป็นคอร์ด และตัวโยงเสียง

ภาคผนวก ก
โปรแกรม
Recognition of Printed Music Note

```
/*****************************************************************************/
/*                                                                           */
/*  PROGRAM FOR : RECOGNITION OF PRINTED MUSIC NOTE                          */
/*                                                                           */
/*       COMPILER : BORLAND/TURBO C/C++                                      */
/*                                                                           */
/*****************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <alloc.h>
#include <malloc.h>
#include <graphics.h>
#include <dos.h>
#include <mem.h>
#include <conio.h>
#include <ctype.h>
#include <io.h>
#include <bios.h>
#include <standard.h>
#include <screenf.h>
#include <mpu401.h>
#include <filefunc.h>
#include <string.h>
#include <fcntl.h>
#include <drvrfunc.h>
#include <play.c>
#include <mouse.c>
#include <drvrfunc.c>



#define offset1         2
#define offset2         3
#define NTRACK          8
#define NUM_EVENTS      999     /* must be even number */
#define HIGH_NOTE       72
#define LOW_NOTE        60
#define METER           4
#define TICKS_PER_BEAT  120
#define NTRACK          8
#define TITLE_WIDE      51
#define TRACK_NAME_WIDE 9
#define NBYTES          30000   /* default track data buffer */
#define META            0xFF    /* meta event codes */
#define TEXTEVENT       01
#define SEQNAME         03
#define INSNAME         04
#define CHANPREF        0x20
#define ENDTRACK        0x2F
```

```
#define SETTEMPO        0x51
#define TIMESIG         0X58
#define NOTE_ON         0x90
#define  MIDI_DRIVER    0x10


void convert(unsigned char *buffer);
void display(unsigned char *buffer,int strow,int stcol,int notecolor,int background);
void inverse_pic(unsigned char *pic,unsigned int size);
void getdata(FILE *f1,unsigned int *data,unsigned int offset);
void thinning(unsigned char *buffer);
void clip_bounary(unsigned char *buffer);
void set_neighbor(unsigned char *buffer,unsigned int row,unsigned int col,
     unsigned char neighbor[]);
void set_neighbor1(unsigned char *buffer,unsigned int row,unsigned int col,
     unsigned char neighbor[]);
void set_neighbor2(unsigned char *buffer,unsigned int row,unsigned int col,
     unsigned char neighbor[]);
void del_fing(unsigned char *buffer);
void co_or(unsigned int point,unsigned int *r,unsigned int *c);
void checkline(unsigned char *buffer,int position[15][4]);
void delette(int position[15][4],unsigned char *buffer);
void delline1(unsigned char *buff,unsigned int row,unsigned int stcol,unsigned int
endcol);
void checkedge(unsigned char *buffer,unsigned char *address[101][2],
     unsigned char *numpice);
void sort(unsigned int  *buffer[101][2],unsigned int n);
void markblock(unsigned char *buffer,unsigned int min_row,unsigned int max_row,
     unsigned int min_col,unsigned int max_col);
void drawblock(unsigned int min_row,unsigned int max_row,unsigned int min_col,
     unsigned int max_col);
void addr_to_coor(unsigned char *address,unsigned int *row,unsigned int *col);
void detecNote(unsigned char *address[101][2],unsigned char numpic);
void detecNote1(unsigned char *address[101][2],unsigned char numpic,int
position[15][4],
     int Note,unsigned char Line,int _row[3]);
void detecNote2(unsigned char *address[101][2],unsigned char numpic);
void detecNote3(unsigned char *address[101][2],unsigned char numpic);
void adjacent(unsigned char ne[]);
void savespk(void);
void convert_mid(void);
void playspk(void);
void guide();
void box(int startx,int starty,int endx,int endy,char mas[30]);
void box1(int startx,int starty,int endx,int endy,char mas[30]);
void massage(int startx,int starty,int endx,int endy);
void menu(void);
void botton_ok(void);
void border(int startx,int starty,int endx,int endy,char mas[30]);
void printfxy(int startx,int starty,int endx,int endy,char mas[30]);
```

```
void shadow(int startx,int starty,char mas[30]);
void convert_spk_sng(char spkname[13],char sngname[13],char midname[13]);
void write_event(FILE *stream, int nbytes, int b0, int b1, int b2, int b3);
void putc_to_file(char *addr, int size, FILE *stream);
void puti_to_file(int *addr, int size, FILE *stream);
void putl_to_file(long *addr, int size, FILE *stream);
void convert_note(int a,int b);
void convert_sng_mid(char sngname[13],char midname[13]);
void write_buf(char *sp, char *ep, FILE *outfile);  /* function prototypes */
void put_to_file(void *addr, int size, FILE *stream);
void put_len(long n, FILE *stream);
void write_mthd(int ntrack, FILE *outfile);
void get_from_file(void *addr, int size, FILE *stream);
unsigned char *addr(unsigned int row,unsigned int col);
unsigned char step1(unsigned char *buffer);
unsigned char step2(unsigned char *buffer);
unsigned char mark1(unsigned char *buffer,unsigned int row,unsigned int col);
unsigned char mark2(unsigned char *buffer,unsigned int row,unsigned int col);
unsigned char num_trans(unsigned char neighbor[]);
unsigned char ppp_chk1(unsigned char neighbor[]);
unsigned char ppp_chk2(unsigned char neighbor[]);
int opengraph(void);
int  CrossNum(unsigned char ne[]);
int stringin(int startx,int starty,char *s);
char *copy_var_len(long n, char *cp);

FILE *open_file(char *filename, char *status);


unsigned char  *picdataref;
unsigned int   maxrowf,maxcolf;
unsigned int   maxrow,maxcol;
unsigned int   XX,YY,YYY;
unsigned char  Line;
        int   g1,g2,g3,g4,g5,g6;
        int   position[15][4];
        int   Note,_row[3],_col[3];
        int   s[999],d[999];
        int   numnote;
        int   GraphDriver;
        int   GraphMode;
        int   maxx, maxy;
        int   note,lenght;
        struct MOUSETYPE m;
```

```
/*********************************************************************/
/*                                                                   */
/*                          MAIN FUNCTION                            */
/*                                                                   */
/*********************************************************************/

void  main()

{

  FILE    *fp;
  register int    i,r,c,y;
  unsigned char   pic;
  unsigned int    staddr,datasizef,datasize,filesize,n;
  unsigned char   *picdata,notes[10],*address[101][2],numpice;
  unsigned int    r1,c1;
           int    graph,Note,step=0;
           int    botton,loop1,loop2=1;

          char   filename[13]="";
  clrscr();
  graph = opengraph();
  if(!graph)
   {
     printf(" Can't open graph ");
     getch();
     exit(1);
   }
  menu();
  maxx = getmaxx();
  maxy = getmaxy();
  if(!mouse_installed())
   {
    closegraph();
    printf("Error resetting mouse\n");
    getch();
    exit(1);
   }
  setregion(0,0,635,475);
  g1=1;g2=1;g3=1;g4=1;g5=1;g6=1;
  do{m = getmouse();
  guide();
  if(m.status == MOUSE_LEFT)
   {
    botton = 0;

    if((m.x>20)&&(m.x<120)&&(m.y>40)&&(m.y<80))
         botton = 1;    /*OPEN BOTTON*/
    else if((m.x>120)&&(m.x<220)&&(m.y>40)&&(m.y<80))
```

```
        botton = 2;    /*STEP BOTTON*/
    else if((m.x>220)&&(m.x<320)&&(m.y>40)&&(m.y<80))
        botton = 3;    /*SAVE BOTTON*/
    else if((m.x>320)&&(m.x<420)&&(m.y>40)&&(m.y<80))
        botton = 4;    /*PLAY BOTTON*/
    else if((m.x>420)&&(m.x<520)&&(m.y>40)&&(m.y<80))
        botton = 5;    /*EXIT BOTTON*/
    else if((m.x>520)&&(m.x<620)&&(m.y>40)&&(m.y<80))
        botton = 6;    /*EXIT BOTTON*/

  switch (botton)
    {
     case 1 : mouse_hide();
                box1(20,40,120,80,"Open");
                mouse_display();
                do{
                m = getmouse();
                }while((m.status == MOUSE_LEFT)&&((m.x>20)
                  &&(m.x<120)&&(m.y>40)&&(m.y<80)));
                mouse_hide();
                box(20,40,120,80,"Open");
                mouse_display();
                if((m.x>20)&&(m.x<120)&&(m.y>40)&&(m.y<80))
                 {
                loop1 = 1;
                fclose(fp);
                dir("*.bmp",filename);
                //cleardevice();
                mouse_hide();
                setfillstyle(SOLID_FILL,BLUE);
                bar(0,120,639,479);
                mouse_display();
                setregion(0,0,635,475);
                if((fp = fopen(filename,"rb"))== NULL)
                 {
                  mouse_hide();
                  setfillstyle(SOLID_FILL,BLUE);
                  bar(0,120,639,479);
                  border(200,230,450,330,"Massege");
                  setfillstyle(SOLID_FILL,WHITE);
                  bar(200,230,450,330);
                  setcolor(BLACK);
                  printfxy(200,240,450,260,"Can't open file        ");
                  outtextxy(360,248,filename);
                  botton_ok();
                  setcolor(DARKGRAY);
                  outtextxy(154,58,"Step");
                  setcolor(BLACK);
                  shadow(54,58,"Open");
                  shadow(342,58,"Convert");
                  shadow(454,58,"Play");
```

```
   shadow(554,58,"Exit");
   mouse_display();
   loop1 = 0;
   break;
  }
mouse_hide();
setcolor(BLACK);
shadow(54,58,"Open");
shadow(154,58,"Step");
shadow(554,58,"Exit");
getdata (fp,&filesize,2);
getdata(fp,&staddr,10);
getdata(fp,&maxcolf,18);
getdata(fp,&maxrowf,22);

datasizef =  filesize - staddr;

if (staddr == 118)
  datasize = datasizef*2;
else
  datasize = datasizef;
maxcol =  (datasize/maxrowf)-1;
maxrow =  maxrowf-1;
if((picdata  = (unsigned char *)malloc(datasize))==NULL)
  {
   free(picdata);
   closegraph();
   printf(" _ram error ");
   getch();
   exit(1);
  }
picdataref = picdata;
fseek(fp,staddr,SEEK_SET);
/* read data */
for(n=1;n<=datasizef;n++)
  {
   pic =  getc(fp);
   if(staddr == 118)
     {
      *picdata = pic >> 4;  /* high byte */
      picdata++;
      *picdata = pic & 0x0f; /* rowbyte */
      picdata++;
     }
   else /* staddr == 1078 */
     {
      *picdata = pic;
      picdata++;
     }
  }
YY =  200-(maxrow/2);
```

```
                YYY =  YY+maxrow+100;
                XX  =  320-(maxcol/2);
                picdata = picdataref;
                inverse_pic(picdata,datasize);
                convert(picdata);
                border(XX,YY,maxcol+XX,maxrow+YY,filename);
                display(picdata,YY,XX,BLACK,WHITE);
                mouse_display();
                loop1 = 1;
                loop2 = 2;
                step = 0;
                break;
                }
                else
                 {
                 loop2 = 1;
                 break;
                 }
case 2 : if(loop1 == 1)
                 {
                  mouse_hide();
                  box1(120,40,220,80,"Step");
                  mouse_display();
                  do{
                  m = getmouse();
                  }while((m.status == MOUSE_LEFT)&&((m.x>120)
                    &&(m.x<220)&&(m.y>40)&&(m.y<80)));
                  mouse_hide();
                  box(120,40,220,80,"Step");
                  mouse_display();
                  if((m.x>120)&&(m.x<220)&&(m.y>40)&&(m.y<80))
                   {
                  if(step == 0)
                   {
                    mouse_hide();
                    clip_bounary(picdata);
                    checkline(picdata,position);
                    border(XX,YYY,(XX+maxcol),(YYY+maxrow),"Del_Line");
                    display(picdata,YYY,XX,RED,WHITE);
                    mouse_display();
                   }
                  else if(step == 1)
                   {
                    mouse_hide();
                    border(XX,YYY,(XX+maxcol),(YYY+maxrow),"Markblock");
                    checkedge(picdata,address,&numpice);
                    mouse_display();
                   }
                  else if(step == 2)
                   {
                    mouse_hide();
```

```
    thinning(picdata);
    border(XX,YYY,(XX+maxcol),(YYY+maxrow),"Thinning");
    display(picdata,YYY,XX,BLUE,WHITE);
    detecNote2(address,numpice);
    mouse_display();
   }
  else if(step == 3)
   {
    mouse_hide();
    detecNote(address,numpice);
    border(XX,YYY,(XX+maxcol),(YYY+maxrow),"Detec_Note");
    display(picdata,YYY,XX,YELLOW,WHITE);
    detecNote2(address,numpice);
    mouse_display();
   }
  else if(step == 4)
   {
    mouse_hide();
    detecNote3(address,numpice);
    mouse_display();
   }
  else if(step == 5)
   {
    numnote = 0;
    s[0] = 0;
    d[0] = 0;
    mouse_hide();
    detecNote1(address,numpice,position,Note,Line,_row);
    mouse_display();
   }
  loop2 = 2;
  step++;
  if(step == 6)
   {
    mouse_hide();
    setcolor(BLACK);
    shadow(254,58,"Save");
    shadow(342,58,"Convert");
    shadow(454,58,"Play");
    setcolor(DARKGRAY);
    outtextxy(154,58,"Step");
    mouse_display();
    step = 0;
    loop1 = 2;
    loop2 = 1;
    free(picdata);
    fclose(fp);
   }
  break;
 }
else break;
```

```
                }
               else break;
case 3 : if(loop1 == 2)
                {
                 mouse_hide();
                 box1(220,40,320,80,"Save");
                 mouse_display();
                 do{
                 m = getmouse();
                  }while((m.status==MOUSE_LEFT)&&((m.x>220)
                    &&(m.x<320)&&(m.y>40)&&(m.y<80)));
                 mouse_hide();
                 box(220,40,320,80,"Save");
                 mouse_display();
                 if((m.x>220)&&(m.x<320)&&(m.y>40)&&(m.y<80))
                   {
                    savespk();
                    break;
                   }
                 else break;
                }
               else break;
case 4 : if(loop2 == 1)
                {
                 mouse_hide();
                 box1(320,40,420,80,"Convert");
                 mouse_display();
                 do{
                 m = getmouse();
                 }while((m.status == MOUSE_LEFT)&&((m.x>320)
                   &&(m.x<420)&&(m.y>40)&&(m.y<80)));
                 mouse_hide();
                 box(320,40,420,80,"Convert");
                 mouse_display();
                 if((m.x>320)&&(m.x<420)&&(m.y>40)&&(m.y<80))
                   {
                    convert_mid();
                    loop1=1;
                    break;
                   }
                }
                 else break;

case 5 : if(loop2 == 1)
                {
                 mouse_hide();
                 box1(420,40,520,80,"Play");
                 mouse_display();
                 do{
                 m = getmouse();
                 }while((m.status == MOUSE_LEFT)&&((m.x>420)
```

```
                    &&(m.x<520)&&(m.y>40)&&(m.y<80)));
                   mouse_hide();
                   box(420,40,520,80,"Play");
                   mouse_display();
                   if((m.x>420)&&(m.x<520)&&(m.y>40)&&(m.y<80))
                    {
                   mouse_hide();
                   setcolor(DARKGRAY);
                   outtextxy(154,58,"Step");
                   outtextxy(254,58,"Save");
                   setcolor(BLACK);
                   shadow(54,58,"Open");
                   shadow(342,58,"Convert");
                   shadow(454,58,"Play");
                   shadow(554,58,"Exit");
                   mouse_display();
                   playspk();
                   loop1 = 3;
                   break;
                  }
                   else break;
                    }
                 else break;
      case 6 : mouse_hide();
                 box1(520,40,620,80,"Exit");
                 mouse_display();
                 do{
                 m = getmouse();
                 }while((m.status == MOUSE_LEFT)&&((m.x>520)
                  &&(m.x<620)&&(m.y>40)&&(m.y<80)));
                 mouse_hide();
                 box(520,40,620,80,"Exit");
                 mouse_display();
                 if((m.x>520)&&(m.x<620)&&(m.y>40)&&(m.y<80))
                  {
                 free(picdata);
                 closegraph();
                 exit(0);
                  }
                 else break;
    }
   }
  }while(m.status != 8);
  free(picdata);
  closegraph();
  exit(1);
}
```

```
/*************************************************************************/
/*                                                                       */
/*                          SUB FUNCTION                                 */
/*                                                                       */
/*************************************************************************/


int opengraph(void)

{
  int graphdriver = DETECT,graphmode=1,ErrorCode;
  char *driverpath = "c:/language/borlandc/bgi";
  initgraph(&graphdriver,&graphmode,driverpath);
  ErrorCode = graphresult();
  if(ErrorCode != grOk)
   {
    closegraph();
    return 0;
   }
  return 1;
}


void inverse_pic(pic,size)
  unsigned char *pic;
  unsigned int size;

{
  int row;
  unsigned char *buffer,*buffref;
  unsigned int maxC;
  buffer = (unsigned char *)malloc(size);
  if(buffer == NULL)
   {
    printf(" ram error ");
    getch();
    exit(1);
   }
  buffref = buffer;
  maxC = maxcol+1;
  for(row = maxrow;row >= 0;row--)
    {
     pic    = addr(row,0);
     memmove(buffer,pic,maxC);
     buffer += (maxC);
    }
  buffer = buffref;
  for(row = 0;row <= maxrow;row++)
    {
```

```
      pic  = addr(row,0);
      memmove(pic,buffer,maxC);
      buffer += (maxC);
     }
   free(buffer);
}


void convert(buffer)
  unsigned char *buffer;

{
  unsigned char check;
  int row,col,y;
  for(row=0,y=maxrow;row<=maxrow;row++,y--)
    for(col=0;col<=maxcol;col++)
         {
          if(col < maxcolf)
            {
             check = (*buffer&0X0F);
             if(check == 0)
               *buffer = 1;
             else if(check==0xF)
               *buffer = 0;
             else
               *buffer = 1;
            }
           else {*buffer = 0; }
            buffer++;
         }
}


void getdata(f1,data,offset)
  FILE *f1;
  unsigned int *data,offset;

{
  int i ;
  *data = 0;
  fseek(f1,offset,SEEK_SET);
  for(i=1;i<=2;i++)
    {
     if(i==2)
         *data |= (getc(f1)<< 8);
     else
         *data =  getc(f1);
    }
}
```

```
unsigned char *addr(row,col)
  unsigned int row,col;

{
  unsigned char *address;
  address = (col+row*(maxcol+1)+picdataref);
  return(address);
}

/*****************************   MENU    *********************************/

void menu()

{
  box(0,0,639,120,"");
  massage(17,37,624,83);
  box(20,40,120,80,"Open");
  box(120,40,220,80,"Step");
  box(220,40,320,80,"Save");
  box(320,40,420,80,"Convert");
  box(420,40,520,80,"Play");
  box(520,40,620,80,"Exit");
  setfillstyle(SOLID_FILL,BLUE);
  bar(0,120,639,479);
  settextstyle(1,0,3);
  setcolor(YELLOW);
  outtextxy(122,4,"Recognition of printed music note");
  setcolor(BLUE);
  outtextxy(120,2,"Recognition of printed music note");
  settextstyle(0,0,0);
  setcolor(DARKGRAY);
  outtextxy(254,58,"Save");
  outtextxy(154,58,"Step");
}

void box(int startx,int starty,int endx,int endy,char mas[30])

{
  mouse_hide();
  setcolor(BLACK);
  rectangle(startx,starty,endx,endy);
  rectangle(startx+1,starty+1,endx-1,endy-1);
  setfillstyle(SOLID_FILL,WHITE);
  bar(startx+2,starty+2,endx-2,endy-2);
  setfillstyle(SOLID_FILL,LIGHTGRAY);
  bar(startx+4,starty+4,endx-5,endy-5);
  setcolor(DARKGRAY);
  line(endx-2,starty+2,endx-2,endy-2);
```

```
  line(endx-3,starty+3,endx-3,endy-3);
  line(endx-4,starty+4,endx-4,endy-4);
  line(endx-2,endy-2,startx+2,endy-2);
  line(endx-3,endy-3,startx+3,endy-3);
  line(endx-4,endy-4,startx+4,endy-4);
  setcolor(BLACK);
  printfxy(startx,starty,endx,endy,mas);
  mouse_display();
}

void box1(int startx,int starty,int endx,int endy,char mas[30])

{
  mouse_hide();
  setcolor(BLACK);
  line(startx+2,starty+2,startx+2,endy-2);
  line(startx+2,starty+2,endx-2,starty+2);
  setfillstyle(SOLID_FILL,LIGHTGRAY);
  bar(startx+3,starty+3,endx-3,endy-3);
  setcolor(BLACK);
  printfxy(startx+1,starty+1,endx,endy,mas);
  mouse_display();
}

void massage(int startx,int starty,int endx,int endy)

{
  setfillstyle(SOLID_FILL,YELLOW);
  bar(startx,starty,endx,endy);
  setcolor(DARKGRAY);
  line(startx,starty,startx,endy);
  line(startx+1,starty+1,startx+1,endy-1);
  line(startx+2,starty+2,startx+2,endy-2);
  line(startx,starty,endx,starty);
  line(startx+1,starty+1,endx-1,starty+1);
  line(startx+2,starty+2,endx-2,starty+2);
  setcolor(WHITE);
  line(endx,starty,endx,endy);
  line(endx-1,starty+1,endx-1,endy-1);
  line(endx-2,starty+2,endx-2,endy-2);
  line(startx,endy,endx,endy);
  line(startx+1,endy-1,endx-1,endy-1);
  line(startx+2,endy-2,endx-2,endy-2);

}

void border(int startx,int starty,int endx,int endy,char mas[30])
```

```
{
  setcolor(LIGHTGRAY);
  line(startx-2,starty-22,startx-2,endy+2);
  line(startx-3,starty-23,startx-3,endy+3);
  line(startx-2,starty-22,endx+2,starty-22);
  line(startx-3,starty-23,endx+3,starty-23);
  line(endx+2,starty-22,endx+2,endy+2);
  line(endx+3,starty-23,endx+3,endy+3);
  line(startx-2,endy+2,endx+2,endy+2);
  line(startx-3,endy+3,endx+3,endy+3);
  setcolor(BLACK);
  line(startx-1,starty-21,startx-1,endy+1);
  line(startx-4,starty-24,startx-4,endy+4);
  line(startx-4,starty-1,endx+4,starty-1);
  line(startx-1,starty-21,endx+1,starty-21);
  line(startx-4,starty-24,endx+4,starty-24);
  line(endx+1,starty-21,endx+1,endy+1);
  line(endx+4,starty-24,endx+4,endy+4);
  line(startx-1,endy+1,endx+1,endy+1);
  line(startx-4,endy+4,endx+4,endy+4);
  line(startx-4,endy-20,startx-1,endy-20);
  line(endx+1,endy-20,endx+4,endy-20);
  line(startx+20,starty-24,startx+20,starty-21);
  line(startx+20,endy+1,startx+20,endy+4);
  line(endx-20,starty-24,endx-20,starty-21);
  line(endx-20,endy+1,endx-20,endy+5);
  setfillstyle(SOLID_FILL,CYAN);
  bar(startx,starty-20,endx,starty-2);
  setcolor(WHITE);
  printfxy(startx,starty-20,endx,starty-2,mas);
}

void printfxy(int startx,int starty,int endx,int endy,char mas[30])
{
  char *spt;
  int  font,width,high,printfx,printfy;
  font = 0;
  spt  = mas;
  while(*spt)
   {
    font++;
    spt++;
   }
  width  =  endx-startx+1;
  high   =  endy-starty+1;
  font   =  font*8;
  printfx =  startx+(width/2)-(font/2);
  printfy =  starty+(high/2)-2;
  shadow(printfx,printfy,mas);
}
```

```
void shadow(int startx,int starty,char mas[30])

{
 setcolor(WHITE);
 outtextxy(startx+1,starty+1,mas);
 setcolor(BLACK);
 outtextxy(startx,starty,mas);
}

dir(char type[6],char filename[13])

{
  struct ffblk ffblk;
  char _filename[44][13];
  char temp;
  int done,i;
  int nfile = 0;
  int lpx = 1;
  int lpy = 1;
  mouse_hide();
  setcolor(DARKGRAY);
  outtextxy(54,58,"Open");
  outtextxy(154,58,"Step");
  outtextxy(254,58,"Save");
  outtextxy(342,58,"Convert");
  outtextxy(454,58,"Play");
  outtextxy(554,58,"Exit");
  setfillstyle(SOLID_FILL,BLUE);
  bar(0,120,639,479);
  mouse_display();
  done = findfirst(type,&ffblk,0);
  while (done)
    {
     mouse_hide();
     setfillstyle(SOLID_FILL,BLUE);
     bar(0,120,639,479);
     border(200,230,450,330,"Massege");
     setfillstyle(SOLID_FILL,WHITE);
     bar(200,230,450,330);
     setcolor(BLACK);
     printfxy(200,240,450,260,"File not found      ");
     outtextxy(370,248,type);
     outtextxy(360,248,filename);
     botton_ok();
     setcolor(DARKGRAY);
     outtextxy(54,58,"Open");
     outtextxy(154,58,"Step");
     outtextxy(254,58,"Save");
     outtextxy(342,58,"Convert");
```

```
      outtextxy(454,58,"Play");
      outtextxy(554,58,"Exit");
      mouse_display();
      break;
     }
   while (!done)
     {
      sprintf(_filename[nfile],"%s",ffblk.ff_name);
      nfile++;
      done = findnext(&ffblk);
     }
    if(nfile>0)
     {
      mouse_hide();
      border(110,160,560,440,"File Open");
      setfillstyle(SOLID_FILL,WHITE);
      bar(110,160,560,440);
      for(i=0;i<nfile;i++)
           {
            setcolor(GREEN);
            outtextxy((118*lpx),160+(lpy*20),_filename[i]);
            lpx++;
            if(lpx == 5)
             {
              lpy++;
              lpx = 1;
             }
           }
      setcolor(BLUE);
      outtextxy(190,425,"Enter file name  : ");
      filename[0]=0;
      mouse_display();
      stringin(350,425,filename);
     }
  return 0;
}


int stringin(int startx,int starty,char *s)
/* string()  :Read character from keybord */

{
  int xx,c;
  unsigned char oldchar[1],ch,chex;
  setcolor(BLUE);
  sprintf(oldchar,"%s",s);
  for(xx=0;oldchar[xx]!=0;xx++);
    for(c=xx+1;c<20;c++)
         s[c] = 0;
  do{
    setfillstyle(SOLID_FILL,WHITE);
```

```
    bar(startx-10,starty-10,startx+120,starty+10);
    outtextxy(startx,starty,s);                    /*Put string to screen*/
    putch(0x20);                                   /*Put space follow string*/
    outtextxy(startx+(xx*8),starty,"_");           /*Move cursor to edit location*/
    ch = getch();                                  /*Read first character*/
    if(ch<0x20)                                    /*Coltrol code*/
     {
        switch(ch)
          {
           case 0x00 : getch();                    /*Read scan code*/
                       break;
           case 0x1b : sprintf(s,"%s",oldchar);/*restore oldstring*/
                       return 0;                   /*cancle editing*/
           case 0x07 : for(c=xx;c<11;c++)          /*del character<g^>*/
                       s[c] = s[c+1];              /*Move character leftside*/
                       s[11] = 0;                  /*Clear most right character*/
                       break;
           case 0x08 : if(xx>0)                    /*del character<del>*/
                       {
                        xx--;                      /*Like case 0x07*/
                        for(c=xx;c<11;c++)
                          s[c] = s[c+1];
                        s[11] = 0;
                       }
     }
    }
    else
     {
        if(xx<12)
          {
           for(c=10;c>=xx;c--)                     /*Insert character*/
             s[c+1] = s[c];
           s[xx] = ch;
           xx++;                                   /*Move cursor location*/
          }
     }
    }while(ch!=0x0d);
  return 0;
 }

void botton_ok(void)
{
  box(300,280,350,310,"Ok");
  mouse_display();
  do
   {
    m = getmouse();
    if(m.status==MOUSE_LEFT)
     {
        if((m.x>300)&&(m.x<350)&&(m.y>280)&&(m.y<310))
          {
```

```
                mouse_hide();
                box1(300,280,350,310,"Ok");
                mouse_display();
                do{
                  m = getmouse();
                 }while((m.status == MOUSE_LEFT)&&((m.x>300)
                   &&(m.x<350)&&(m.y>280)&&(m.y<310)));
                 mouse_hide();
                 box(300,280,350,310,"OK");
                 mouse_display();
                 if((m.x>300)&&(m.x<350)&&(m.y>280)&&(m.y<310))
                     {
                      mouse_hide();
                      setfillstyle(SOLID_FILL,BLUE);
                      bar(0,120,639,479);
                      mouse_display();
                      m.status=9;
                      setcolor(BLACK);
                      shadow(54,58,"Open");
                      shadow(342,58,"Convert");
                      shadow(454,58,"Play");
                      shadow(554,58,"Exit");
                      break;
                     }
              }
        }
    }while(m.status!=9);
}

void guide()

 {
  if(((m.x>20)&&(m.x<120)&&(m.y>40)&&(m.y<80))&&g1==1)
    {
     setfillstyle(SOLID_FILL,LIGHTGRAY);
     bar(5,85,634,115);
     settextstyle(1,0,3);
     setcolor(YELLOW);
     outtextxy(122,84,"Open file *.bmp");
     setcolor(BLUE);
     outtextxy(120,82,"Open file *.bmp");
     settextstyle(0,0,0);
     g1=2;g2=1;g3=1;g4=1;g5=1;g6=1;
    }
   else if(((m.x>120)&&(m.x<220)&&(m.y>40)&&(m.y<80))&&g2==1)
    {
     setfillstyle(SOLID_FILL,LIGHTGRAY);
     bar(5,85,634,115);
     settextstyle(1,0,3);
     setcolor(YELLOW);
     outtextxy(122,84,"Image Recognition");
```

```
    setcolor(BLUE);
    outtextxy(120,82,"Image Recognition");
    settextstyle(0,0,0);
    g2=2;g1=1;g3=1;g4=1;g5=1;g6=1;
   }
  else if(((m.x>220)&&(m.x<320)&&(m.y>40)&&(m.y<80))&&g3==1)
   {
    setfillstyle(SOLID_FILL,LIGHTGRAY);
    bar(5,85,634,115);
    settextstyle(1,0,3);
    setcolor(YELLOW);
    outtextxy(122,84,"Save file *.spk");
    setcolor(BLUE);
    outtextxy(120,82,"Save file *.spk");
    settextstyle(0,0,0);
    g3=2;g1=1;g2=1;g4=1;g5=1;g6=1;
   }
  else if(((m.x>320)&&(m.x<420)&&(m.y>40)&&(m.y<80))&&g4==1)
   {
    setfillstyle(SOLID_FILL,LIGHTGRAY);
    bar(5,85,634,115);
    settextstyle(1,0,3);
    setcolor(YELLOW);
    outtextxy(122,84,"Convert file *.spk to *.mid");
    setcolor(BLUE);
    outtextxy(120,82,"Convert file *.spk to *.mid");
    settextstyle(0,0,0);
    g4=2;g1=1;g2=1;g3=1;g5=1;g6=1;
   }
  else if(((m.x>420)&&(m.x<520)&&(m.y>40)&&(m.y<80))&&g5==1)
   {
    setfillstyle(SOLID_FILL,LIGHTGRAY);
    bar(5,85,634,115);
    settextstyle(1,0,3);
    setcolor(YELLOW);
    outtextxy(122,84,"Play sound out speaker");
    setcolor(BLUE);
    outtextxy(120,82,"Play sound out speaker");
    settextstyle(0,0,0);
    g5=2;g1=1;g2=1;g3=1;g4=1;g6=1;
   }
  else if(((m.x>520)&&(m.x<620)&&(m.y>40)&&(m.y<80))&&g6==1)
   {
    setfillstyle(SOLID_FILL,LIGHTGRAY);
    bar(5,85,634,115);
    settextstyle(1,0,3);
    setcolor(YELLOW);
    outtextxy(122,84,"Exit program");
    setcolor(BLUE);
    outtextxy(120,82,"Exit program");
    settextstyle(0,0,0);
```

```
    g6=2;g1=1;g2=1;g3=1;g4=1;g5=1;
   }
}

/*************************** CLIP_BOUNARY ***************************/

void clip_bounary(buffer)
  unsigned char *buffer;

{
 unsigned int row,col,x;
 x = maxrow-1;
 for(col=0;col<=maxcol;col++)
   {
    buffer = addr(0,col);
    *buffer=0;
    buffer = addr(1,col);
    *buffer=0;
    buffer = addr(maxrow,col);
    *buffer=0;
    buffer = addr(x,col);
    *buffer=0;
   }
 x = maxcol-1;
 for(row=0;row<=maxrow;row++)
   {
    buffer = addr(row,0);
    *buffer=0;
    buffer = addr(row,1);
    *buffer=0;
    buffer = addr(row,maxcol);
    *buffer=0;
    buffer = addr(row,x);
    *buffer=0;
   }
}

/*************************** CHECKLINE ***************************/

void checkline(buffer,position)
  unsigned char *buffer;
  int position[15][4];

{
  unsigned  int row,col;
  unsigned int  count,cen,sum,longline;
  unsigned char ne[9];
  count   = 0;
```

```
  longline = 0;
  for(row =1 ;row < maxrow;row++)
    for(col=1;col < maxcol;col++)
          {
           buffer = addr(row,col);
           set_neighbor(buffer,row,col,ne);
           if(ne[0]&ne[3] == 1)
             count++;
           else if(count > 0 && count > longline)
             longline = count;
           else
             count = 0;
          }
  count = 0;
  cen   = 0;
  for(row =1 ;row < maxrow;row++)
    for(col=1;col < maxcol;col++)
          {
           buffer = addr(row,col);
           set_neighbor(buffer,row,col,ne);
           if(ne[0]&ne[3] == 1)
          {
  if(count == 0)
    position[cen][0] = col;
  count++;
}
  else if(count > 0)
   {
    sum =  longline - count;
    if(sum <= 10 || sum == 0)
         {
          position[cen][1] = col;
          position[cen][2] = row;
          position[cen][3] = count;
          cen ++;
          count = 0;
         }
    else count = 0;
   }
    else count = 0;
   }
  delette(position,buffer);
}


void delette(position,buffer)
  int position[15][4];
  unsigned char *buffer;

{
  int cen   = 0;
```

```
  int row,stcol,endcol,count;
  int leger[4];
  leger[0] = position[0][2]-(position[1][2]-position[0][2]);
  leger[1] = position[0][2]-(2*(position[1][2]-position[0][2]));
  leger[2] = position[4][2]+(position[1][2]-position[0][2]);
  leger[3] = position[4][2]+(2*(position[1][2]-position[0][2]));
  for(count=0;count <= 4;count++,cen++)
    {
     row    = position[cen][2];
     stcol  = position[cen][0];
     endcol = position[cen][1];
     delline1(buffer,row,stcol,endcol);
    }
  row = leger[0];
  delline1(buffer,row,stcol,endcol);
  row = leger[1];
  delline1(buffer,row,stcol,endcol);
  row = leger[2];
  delline1(buffer,row,stcol,endcol);
  row = leger[3];
  delline1(buffer,row,stcol,endcol);
}


void delline1(buff,row,stcol,endcol)
  unsigned char *buff;
  unsigned int stcol,endcol,row;

{
  unsigned char ne[9];
  unsigned int  col,scol,srow;
// scol =  20;
// srow =  30;
  for( col = stcol ; col <= endcol ;col++ )
    {
     buff = addr(row,col);
     set_neighbor(buff,row,col,ne);
     if((ne[1]|ne[5]) == 0 )
         {
          *buff = 0;
//        putpixel((scol+col),(srow+row),BLACK);
         }
    }
}


/*************************       DISPLAY NOTE      **************************/


void display(buffer,strow,stcol,notecolor,background)
  unsigned char *buffer;
```

```
  int strow,stcol;
  int notecolor,background;

{
  int  row,col,y;
  unsigned char pic;
  for(row = 0;row  <= maxrow;row++)
    for(col=0;col <= maxcol;col++)
         {
          pic = (*buffer&0xf);
          if(pic ==1 )
            putpixel((col+stcol),(row+strow),notecolor);
          else if(pic == 0)
            putpixel((col+stcol),(row+strow),background);
          else if(pic == 2)
            putpixel((col+stcol),(row+strow),BLUE);
          else if(pic == 6)
            putpixel((col+stcol),(row+strow),BLACK);
          else if(pic == 8)
            putpixel((col+stcol),(row+strow),GREEN);
          else
            putpixel((col+stcol),(row+strow),WHITE);
          buffer ++;

         }
}

/*************************     CHECKEDGE     *************************/


void checkedge(buffer,address,numpice)
  unsigned char *buffer,*address[101][2],*numpice;

{
  unsigned int  max_row,min_row,min_col,max_col;
  unsigned int  erow,ecol,row,col,i,p;
  unsigned char *staddress,*lastpoint;
  unsigned char ne[9],select,ready;
  *numpice = 0;
  erow = maxrow-1;
  ecol = maxcol-1;
  do
   {
    ready = 0;
    for(i = 0;i<=9;i++)
         ne[i] = 0;
    for(col=1;col<=ecol;col++)
         for(row = 1;row <= erow;row++)
           {
            buffer = addr(row,col);
```

```
set_neighbor(buffer,row,col,ne);
i = 1;
min_col = 0;
max_col = 0;
min_row = 0;
max_row = 0;
while((ne[i] == 0)&&(i<9))
 i++;
if(i<9)
 {
  ready = 1;
  min_col = col;
  min_row = row;
  i = i-1;
  lastpoint = buffer;
  co_or(i,&row,&col);
  buffer = addr(row,col);
  staddress = buffer;
  do
     {
     if (min_row > row) min_row = row;
     if (max_row < row) max_row = row;
     if (max_col < col) max_col = col;
     set_neighbor(buffer,row,col,ne);
     i = 1;
     do
      {
      if(ne[i]==0)
       {
        i++ ;
       select = 0;
       }
      else                    /* ne[i]=1 */
       {
        if(ne[i-1] ==0)
         {
          if(i != 1)
              {
               co_or((i-1),&row,&col);
               if(lastpoint != addr(row,col))
                {
                 lastpoint = buffer;
                 buffer = addr(row,col);
                 select = 1;
                }
               else
                {
                 if(i < 5)
                   p = i+4;
                 else
                   p = p-4;
```

```
                co_or(p,&row,&col);
                i++;
                select = 0;
               }
             }
          else if(ne[ε]==0)          /* i ==1  */
             {
              co_or(8,&row,&col);
              if(lastpoint != addr(row,col))
               {
                lastpoint = buffer;
                buffer = addr(row,col);
                select = 1;
               }
              else
               {
                co_or(4,&row,&col);
                select = 0;
                i++;
               }
             }
           else
             {
              i++;
              select = 0;
             }
           }
          else
           {
            i++;
            select = 0;
           }
         }
       }while(select==0);
      }while (buffer != staddress);
min_col = min_col;
max_col = max_col;
min_row = min_row;
max_row = max_row;
if(max_col < ecol)
    col = max_col+1;
else col = ecol;
    row = 0;
(*numpice)++;
address[*numpice][0] = addr(min_row,min_col);
address[*numpice][1] = addr(max_row,max_col);
markblock(buffer,min_row,max_row,min_col,max_col);
drawblock(min_row,max_row,min_col,max_col);

} /* end if */
```

```
            } /* end for */

    }while(ready==1);

    sort(address,*numpice);
    for(col=1;col<=ecol;col++)
      for(row = 1;row <= erow;row++)
          {
           buffer = addr(row,col);
           if(*buffer == 2)
             *buffer = 1;
          }
}

void sort(buffer,n)
  unsigned int  *buffer[101][2],n;

{

  unsigned int  check,*buff[2],num1,num2,row1,col1,row2,col2;
  int i,j,k;
  num1 = n;
  num2 = num1-1;
  j = num2;
  do
   {
    check = 0;
    for(i=1;i<=j;i++)
        {
         addr_to_coor(buffer[i][0],&row1,&col1);
         addr_to_coor(buffer[i+1][0],&row2,&col2);
         if(col1 > col2)
           {
            for(k=0;k<=1;k++)
              {
                 buff[k]       =  buffer[i][k];
                 buffer[i][k]  =  buffer[i+1][k];
                 buffer[i+1][k] =  buff[k] ;
              }
            check = 1;
           }
        }
    j--;
   }while(check==1);
}


void markblock(buffer,min_row,max_row,min_col,max_col)
  unsigned char *buffer;
  unsigned int min_row,max_row,min_col,max_col;
```

```
{
  unsigned int row,col;
  for(row=min_row;row<=max_row;row++)
    for(col=min_col;col<=max_col;col++)
         {
          buffer = addr(row,col);
          if(*buffer==1)
            *buffer = 2;
         }
}


void drawblock(min_row,max_row,min_col,max_col)
  unsigned int min_row,max_row,min_col,max_col;

{
  unsigned int row,col;
  int color;
  color = BLACK;
  for(col = min_col;col<=max_col;col++)
    putpixel((col+XX),(min_row+YYY),color);
  for(col = min_col;col<=max_col;col++)
    putpixel((col+XX),(max_row+YYY),color);
  for(row = min_row;row<=max_row;row++)
    putpixel((min_col+XX),(row+YYY),color);
  for(row = min_row;row<=max_row;row++)
    putpixel((max_col+XX),(row+YYY),color);
}


void addr_to_coor(address,row,col)
  unsigned char *address;
  unsigned int *row,*col;

{
  unsigned int  offset,stoffset;
  stoffset =  FP_OFF(picdataref);
  offset   =  FP_OFF(address)-stoffset;
  *col     =  (offset%(maxcol+1));
  *row     =  (offset-*col)/(maxcol+1);
}


/*****************************    THINING    *****************************/


void thinning( buffer )
  unsigned char *buffer;
{
  unsigned char flag1,flag2;
  do
```

```
   {
    flag1=step1(buffer);
    flag2=step2(buffer);
   }while(flag1||flag2);
}


unsigned char step1(buffer)
  unsigned char *buffer;

{
  unsigned char flag1;
  unsigned int  row,col;
  flag1=0;
  for(row=1;row<maxrow;row++)
     for(col=1;col<maxcol;col++)
          {
           flag1|=mark1(buffer,row,col);
          }
  if(flag1==1)
    {
     del_fing(buffer);
    }
  return flag1;
}


unsigned char step2(buffer)
  unsigned char *buffer;

{
  unsigned char flag2;
  unsigned int  row,col;
  flag2=0;
  for(row=1;row<maxrow;row++)
     for(col=1;col<maxcol;col++)
          {
           flag2|=mark2(buffer,row,col);
          }
  if(flag2==1)
    {
     del_fing(buffer);
    }
  return flag2;
}


unsigned char mark1(buffer,row,col)
  unsigned char *buffer;
  unsigned int  row,col;
{
```

```
  unsigned char neighbor[9];
  unsigned char pass;
  buffer = addr(row,col);
  if(*buffer==0)
    return 0;
  set_neighbor(buffer,row,col,neighbor);
  pass=num_trans(neighbor);
  if(pass==0)
    return 0;
  pass=ppp_chk1(neighbor);
  if(pass==0)
    return 0;
  *buffer = 0x3;
    return 1;
}


unsigned char mark2(buffer,row,col)
  unsigned char *buffer;
  unsigned int  row,col;
{
  unsigned char neighbor[9];
  unsigned char pass;
  buffer = addr(row,col);
  if((*buffer)==0)
    return 0;
  set_neighbor(buffer,row,col,neighbor);
  pass=num_trans(neighbor);
  if(pass==0)
    return 0;
  pass=ppp_chk2(neighbor);
  if(pass==0)
    return 0;
  *buffer =0x3;
  return 1;
}


void set_neighbor(buffer,row,col,neighbor)
  unsigned char *buffer;
  unsigned char neighbor[];
  unsigned int  row,col;

{
  unsigned char i;
  buffer = addr(row,col);
  neighbor[0]=*buffer;
  buffer = addr(row-1,col);
  neighbor[1]=*buffer;
  buffer = addr(row-1,col+1);
  neighbor[2]=*buffer;
```

```
 buffer = addr(row,col+1);
 neighbor[3]=*buffer;
 buffer = addr(row+1,col+1);
 neighbor[4]=*buffer;
 buffer = addr(row+1,col);
 neighbor[5]=*buffer;
 buffer = addr(row+1,col-1);
 neighbor[6]=*buffer;
 buffer = addr(row,col-1);
 neighbor[7]=*buffer;
 buffer = addr(row-1,col-1);
 neighbor[8]=*buffer;
 for(i=0;i<9;i++)
   neighbor[i]&=0x1;
}


unsigned char num_trans(neighbor)
  unsigned char neighbor[9];

{
 unsigned char np,sp;
 unsigned char i;
 np=sp=0;
 for(i=1;i< 8;i++)
   {
    if(neighbor[i]==1)
        np++;
    if((neighbor[i]==0)&&(neighbor[i+1]==1))
        sp++;
   }
 if(neighbor[8]==1)
   np++;
 if((neighbor[8]==0)&&(neighbor[1]==1))
   sp++;
 if((np>=2)&&(np<=6)&&( sp==1))
   return 1;
 return 0;
}


unsigned char ppp_chk1(neighbor)
  unsigned char neighbor[9];

{
 if(((neighbor[3]&neighbor[5])&(neighbor[1]|neighbor[7])) == 0 )
   return 1;
 else
   return 0;
}
```

```
unsigned char ppp_chk2(neighbor)
  unsigned char neighbor[9];

{
  if(((neighbor[1]&neighbor[7])&(neighbor[3]|neighbor[5]))==0)
    return 1;
  else
    return 0;
}


void del_fing(buffer)
  unsigned char *buffer;
{
  unsigned int row,col;
  for(row=0;row<=maxrow;row++)
     for(col=0;col<=maxcol;col++)
          {
           if(*buffer>=0x2)
           {
            *buffer=0;
           }
           buffer++;
          }
}


void co_or(point,r,c)
  unsigned int point,*r,*c;

{
  switch (point)
   {
    case 0 : *r= *r;
             *c= *c;
             break;
    case 1 : *r =  *r-1;
             *c =  *c;
             break;
    case 2 : *r =  *r-1;
             *c =  *c+1;
             break;
    case 3 : *r =  *r;
             *c =  *c+1;
             break;
    case 4 : *r =  *r+1;
             *c =  *c+1;
             break;
    case 5 : *r =  *r+1;
             *c =  *c;
```

```
            break;
    case 6 : *r =  *r+1;
            *c =  *c-1;
            break;
    case 7 : *r =  *r;
            *c =  *c-1;
            break;
    case 8 : *r =  *r-1;
            *c =  *c-1;
            break;
   }
}


int CrossNum(ne)
  unsigned char ne[];

{
  int i,n=0;
  for(i=1;i<=7;i++)
    {
     if(ne[i]>0)
         ne[i]=1;
     if(ne[i+1]>0)
         ne[i+1]=1;
     n=n+abs(ne[i+1]-ne[i]);
    }
  if(ne[8]>0)
    ne[8]=1;
  if(ne[1]>0)
    ne[1]=1;
  n=n+abs(ne[1]-ne[8]);
  return n;
}

/*****************************   DETECNOTE    *************************/


void detecNote(address,numpic)
  unsigned char *address[101][2],numpic;

{
  unsigned int row,col,strow,stcol,endrow,endcol;
  unsigned char ne[9],*buffer;
  int  n,i;
  for(i=1;i<=numpic;i++)
    {
     addr_to_coor(address[i][0],&strow,&stcol);
     addr_to_coor(address[i][1],&endrow,&endcol);
     for(row=strow+1;row <= endrow;row++)
```

```
        for(col=stcol+1;col <= endcol;col++)
          {
           /* closegraph();*/
           set_neighbor1(buffer,row,col,ne);
           /*  buffer=addr(row,col);
           adjacent(ne);
           printf("row=%d,col=%d,buffer=%d\n",row,col,*buffer);
           getch();*/
           if(ne[0]==1)
               {
                n=CrossNum(ne);
                buffer=addr(row,col);
                /*adjacent(ne);
                printf("row=%d,col=,%d,buffer=%d\n",row,col,*buffer);
                getch();*/
                if(n==2)
                 *buffer=2;
                /* set_neighbor1(buffer,row,col,ne);
                buffer=addr(row,col);
                adjacent(ne);
                printf("row=%d,col=%d,buffer=%d\n",row,col,*buffer);
                getch();*/
                if(n==6)
                  *buffer = 6;
                if(n==8)
                  *buffer = 8;
               }
           }
    }
    setcolor(BLACK);
    line(stcol+XX,strow+YYY,endcol+XX,strow+YYY);
    line(endcol+XX,strow+YYY,endcol+XX,endrow+YYY);
    line(stcol+XX,endrow+YYY,endcol+XX,endrow+YYY);
    line(stcol+XX,strow+YYY,stcol+XX,endrow+YYY);
}


void detecNote2(address,numpic)
  unsigned char *address[101][2],numpic;

{
  unsigned int strow,stcol,endrow,endcol;
  int  i;
  for(i=1;i<=numpic;i++)
    {
     addr_to_coor(address[i][0],&strow,&stcol);
     addr_to_coor(address[i][1],&endrow,&endcol);
     setcolor(BLACK);
     line(stcol+XX,strow+YYY,endcol+XX,strow+YYY);
     line(endcol+XX,strow+YYY,endcol+XX,endrow+YYY);
     line(stcol+XX,endrow+YYY,endcol+XX,endrow+YYY);
```

```
      line(stcol+XX,strow+YYY,stcol+XX,endrow+YYY);
     }
}


void detecNote3(address,numpic)
  unsigned char *address[101][2],numpic;

{
  unsigned int strow,stcol,endrow,endcol,midrow,midcol;
  int  i;
  for(i=1;i<=numpic;i++)
     {
      addr_to_coor(address[i][0],&strow,&stcol);
      addr_to_coor(address[i][1],&endrow,&endcol);
      midrow = (((endrow - strow)/2)+strow);
      midcol = (((endcol - stcol)/2)+stcol);
      setcolor(BLACK);
      line(stcol+XX,strow+YYY,endcol+XX,strow+YYY);
      line(endcol+XX,strow+YYY,endcol+XX,endrow+YYY);
      line(stcol+XX,endrow+YYY,endcol+XX,endrow+YYY);
      line(stcol+XX,strow+YYY,stcol+XX,endrow+YYY);
      line(midcol+XX,strow+YYY,midcol+XX,endrow+YYY);
      line(stcol+XX,midrow+YYY,endcol+XX,midrow+YYY);

     }
}


void adjacent(unsigned char ne[])

{
  printf("%d%d%d\n",ne[8],ne[1],ne[2]);
  printf("%d%d%d\n",ne[7],ne[0],ne[3]);
  printf("%d%d%d\n",ne[6],ne[5],ne[4]);
}


void set_neighbor1(buffer,row,col,neighbor)
  unsigned char *buffer;
  unsigned char neighbor[];
  unsigned int  row,col;

{
  unsigned char i;
  buffer = addr(row,col);
  neighbor[0]=*buffer;
  buffer = addr(row-1,col);
  neighbor[1]=*buffer;
  buffer = addr(row-1,col+1);
  neighbor[2]=*buffer;
```

```
  buffer = addr(row,col+1);
  neighbor[3]=*buffer;
  buffer = addr(row+1,col+1);
  neighbor[4]=*buffer;
  buffer = addr(row+1,col);
  neighbor[5]=*buffer;
  buffer = addr(row+1,col-1);
  neighbor[6]=*buffer;
  buffer = addr(row,col-1);
  neighbor[7]=*buffer;
  buffer = addr(row-1,col-1);
  neighbor[8]=*buffer;
}


void detecNote1(address,numpic,position,Note,Line,_row)
  unsigned char *address[101][2],numpic,Line;
  int Note,_row[3];
  int position[15][4];

{
  unsigned int  row,col,strow,stcol,midrow,midcol,endrow,endcol;
  unsigned char ne[1],*buffer;
  int  n,i,o,p,count,condition,high,ACC,black,width;
  int  con[4][3] = {{0,0,0},{0,0,0},{0,0,0},{0,0,0}};
  width = position[2][2]-position[1][2];
  for(i=1;i<=numpic;i++)
    {
     addr_to_coor(address[i][0],&strow,&stcol);
     addr_to_coor(address[i][1],&endrow,&endcol);
     midrow = (((endrow - strow)/2)+strow);
     midcol = (((endcol - stcol)/2)+stcol);
     black = 0;

     for(o=0;o<=3;o++)
          for(p=0;p<=2;p++)
            {
             con[o][p] = 0;
            }
     for(row=strow+1;row<=midrow;row++)
          for(col=stcol+1;col<=midcol;col++)
            {
             set_neighbor2(buffer,row,col,ne);
             if(ne[1]==2||ne[1]==6||ne[1]==8)
                {
                 if(ne[1]==2 && con[0][0]==0)
                   con[0][0] = 2;
                 if(ne[1]==2 && con[0][0]>0)
                   con[0][0] = con[0][0]+10;
                 if(ne[1]==6 && con[0][1]==0)
                   con[0][1] = 6;
```

```
            if(ne[1]==6 && con[0][1]>0)
              con[0][1] = con[0][1]+10;
            if(ne[1]==8 && con[0][2]==0)
              con[0][2] = 8;
            if(ne[1]==8 && con[0][2]>0)
              con[0][2] = con[0][2]+10;
           }
        }
for(row=midrow+1;row<=endrow;row++)
        for(col=stcol+1;col<=midcol;col++)
       {
        set_neighbor2(buffer,row,col,ne);
        if(ne[1]==2||ne[1]==6||ne[1]==8)
         {
            if(ne[1]==2 && con[1][0]==0)
              con[1][0] = 2;
            if(ne[1]==2 && con[1][0]>0)
              con[1][0] = con[1][0]+10;
            if(ne[1]==6 && con[1][1]==0)
              con[1][1] = 6;
            if(ne[1]==6 && con[1][1]>0)
              con[1][1] = con[1][1]+10;
            if(ne[1]==8 && con[1][2]==0)
              con[1][2] = 8;
            if(ne[1]==8 && con[1][2]>0)
              con[1][2] = con[1][2]+10;
         }
        }
for(row=strow+1;row<=midrow;row++)
     for(col=midcol+1;col<=endcol;col++)
      {
       set_neighbor2(buffer,row,col,ne);
       if(ne[1]==2||ne[1]==6||ne[1]==8)
           {
            if(ne[1]==2 && con[2][0]==0)
              con[2][0] = 2;
            if(ne[1]==2 && con[2][0]>0)
              con[2][0] = con[2][0]+10;
            if(ne[1]==6 && con[2][1]==0)
              con[2][1] = 6;
            if(ne[1]==6 && con[2][1]>0)
              con[2][1] = con[2][1]+10;
            if(ne[1]==8 && con[2][2]==0)
              con[2][2] = 8;
            if(ne[1]==8 && con[2][2]>0)
              con[2][2] = con[2][2]+10;
           }
       }
for(row=midrow+1;row<=endrow;row++)
     for(col=midcol+1;col<=endcol;col++)
      {
```

```
                set_neighbor2(buffer,row,col,ne);
                if(ne[1]==2||ne[1]==6||ne[1]==8)
                    {
                     if(ne[1]==2 && con[3][0]==0)
                       con[3][0] = 2;
                     if(ne[1]==2 && con[3][0]>0)
                       con[3][0] = con[3][0]+10;
                     if(ne[1]==6 && con[3][1]==0)
                       con[3][1] = 6;
                     if(ne[1]==6 && con[3][1]>0)
                       con[3][1] = con[3][1]+10;
                     if(ne[1]==8 && con[3][2]==0)
                       con[3][2] = 8;
                     if(ne[1]==8 && con[3][2]>0)
                       con[3][2] = con[3][2]+10;
                    }
                 else if(ne[1]==1)
                     black = 1;
               }
        condition = 0;

if(con[0][0]==0&&con[0][1]==0&&con[0][2]==0&&con[1][0]==0&&con[1][1]==0&&con
[1][2]==0&&con[2][0]==0&&con[2][1]==0&&con[2][2]==0&&con[3][0]==0&&con[3][1]
==0&&con[3][2]==0)
            condition = 1;
        else if(con[0][0]==0&&con[2][0]==12&&con[3][1]==16)
            condition = 2;
        else if(con[0][1]==16&&con[1][0]==12&&con[2][1]==0)
            condition = 3;
        else
if((con[0][0]==0&&con[1][0]==12&&con[2][0]==12&&con[2][1]==0)||(con[0][0]==0&&c
on[2][0]==12&&con[2][1]==0&&con[3][0]==12))
            condition = 4;
        else
if(con[1][1]==0&&(con[1][0]==12||con[1][0]==22)&&(con[3][0]==12||con[3][0]==22))
            condition = 5;
        else
if(con[0][1]==0&&(con[0][0]==12||con[0][0]==22)&&(con[2][0]==12||con[2][0]==22))
            condition = 6;
        else if(con[0][0]==22&&con[1][0]==12)
            condition = 7;
        else if(con[0][0]==12&&con[0][1]==16)
            condition = 8;
        else if(con[0][0]==12&&con[1][0]==12&&con[2][1]==0)
            condition = 9;
        else if(con[0][0]==12&&con[1][0]==0&&con[1][1]==0&&con[3][0]==12)
            condition = 10;
        else if(con[0][0]==0&&con[3][0]==12&&con[3][1]==16)
            condition = 11;
        else if(con[0][1]==26&&con[1][0]==22&&con[3][2]==18)
            condition = 12;
```

```
switch(condition)
    {
     case 1 : if((endrow-strow)<5)
                d[numnote-1] = d[numnote-1] + (d[numnote-1]/2);
                else if((endrow-strow)>5)
                 {
                  Note = 32; /* WHOLE NOTE */
                  detecLine(endrow,position,&Line);
                  Makedata(Line,Note,ACC);
                  ACC = 0;
                 }
                break;
     case 2 : Note = 16; /* HALF NOTE */
                detecLine(endrow,position,&Line);
                Makedata(Line,Note,ACC);
                ACC = 0;
                break;
     case 3 : Note = 16; /* HALF NOTE */
                detecLine(strow+width+2,position,&Line);
                Makedata(Line,Note,ACC);
                ACC = 0;
                break;
     case 4 : if(con[0][0]==0&&con[1][0]==12&&con[2][0]==12)
                 {
                  if(black==1)
                     {
                      Note = 8; /* QUARTER NOTE */
                      detecLine(endrow,position,&Line);
                      Makedata(Line,Note,ACC);
                      ACC = 0;
                     }
                   else if(black==0)
                     {
                      Note = 8; /* QUARTER NOTE */
                      detecLine(strow+width+2,position,&Line);
                      Makedata(Line,Note,ACC);
                      ACC = 0;
                     }
                 }
                else if(con[0][0]==0&&con[2][0]==12&&con[3][0]==12)
                 {
                       Note = 8; /* QUARTER NOTE */
                       detecLine(endrow,position,&Line);
                       Makedata(Line,Note,ACC);
                       ACC = 0;
                 }
                break;
     case 5 : if(con[0][0]==0&&con[2][0]==12&&con[2][1]==16) /* EIGHT NOTE
*/
                 {
                  Note = 4;
```

```
 detecLine(endrow,position,&Line);
 Makedata(Line,Note,ACC);
 ACC = 0;
}
else if(con[0][0]==00&&con[2][0]==12&&con[2][1]==36)
 /* SIXTEEN NOTE */
 {
  Note = 2;
  detecLine(endrow,position,&Line);
  Makedata(Line,Note,ACC);
  ACC = 0;
 }
else if(con[2][1]==00) /* TWO-EIGHT NOTE */
 {
  Note = 4;
  checkNote(address,i,2,_row);
  for(count=0;count<=1;count++)
       {
        detecLine1(_row[count],position,&Line);
        Makedata(Line,Note,ACC);
        //printf("Notenumber %d = %d,%c\n",i,Note,Line);
       }
 }
 else if(con[0][1]==16&&con[2][1]==16&&con[3][0]==12)
  /*TWO-SIXTEEN NOTE */
  {
   Note = 2;
   checkNote(address,i,2,_row);
   for(count=0;count<=1;count++)
       {
        detecLine1(_row[count],position,&Line);
        Makedata(Line,Note,ACC);
        //printf("Notenumber %d = %d,%c\n",i,Note,Line);
       }
  }
 else if(con[2][0]==0&&con[2][1]==16&&con[3][0]==12)
  /* TREE-EIGHT NOTE */
  {
   Note = 4;
   checkNote(address,i,3,_row);
   for(count=0;count<=2;count++)
       {
        detecLine1(_row[count],position,&Line);
        Makedata(Line,Note,ACC);
        //printf("Notenumber %d = %d,%c\n",i,Note,Line);
       }
  }
 else if(con[0][1]==16&&con[2][1]==26&&con[2][2]==18)
  /* TREE-SIXTEEN NOTE */
  {
   Note = 2;
```

```
                checkNote(address,i,3,_row);
                for(count=0;count<=2;count++)
                    {
                    detecLine1(_row[count],position,&Line);
                     Makedata(Line,Note,ACC);
                     //printf("Notenumber %d = %d,%c\n",i,Note,Line);
                    }
               }
              else if(con[0][1]==16&&con[2][1]==16&&con[3][0]==22)
               /* FOUR-EIGHT NOTE */
               {
                Note = 4;
                checkNote(address,i,4,_row);
                for(count=0;count<=3;count++)
                    {
                     detecLine1(_row[count],position,&Line);
                     Makedata(Line,Note,ACC);
                     //printf("Notenumber %d = %d,%c\n",i,Note,Line);
                    }
               }
              else
if(con[0][1]==26&&con[0][2]==18&&con[2][1]==26&&con[2][2]==18)
               /* FOUR-SIXTEEN NOTE */
               {
                Note = 2;
                checkNote(address,i,4,_row);
                for(count=0;count<=3;count++)
                    {
                     detecLine1(_row[ccunt],position,&Line);
                     Makedata(Line,Note,ACC);
                     //printf("Notenumber %d = %d,%c\n",i,Note,Line);
                    }
               }
              break;
        case 6 : if(con[1][1]==00)
               {
                high = endrow-strow;
                if(high<(position[1][2]-position[0][2]))
                    {
                     if(endrow==position[2][2]+1) /* HALF NOTE REST */
                      {
                       Note = 16;
                       Line = 'N';
                       Makedata(Line,Note,ACC);
                      }
                     else if(strow==position[1][2]-1) /* WHOLE NOTE REST */
                      {
                       Note = 32;
                       Line = 'N';
                       Makedata(Line,Note,ACC);
                      }
```

```
                    }
                else if(high>(position[1][2]-position[0][2])) /* TWO-EIGHT NOTE
*/
                    {
                     Note = 4;
                     checkNote(address,i,2,_row);
                     for(count=0;count<=1;count++)
                       {
                        detecLine1(_row[count],position,&Line);
                        Makedata(Line,Note,ACC);
                        //printf("Notenumber %d = %d,%c\n",i,Note,Line);
                       }
                    }
                }
               else if(con[1][1]==16&&con[2][0]==12&&con[3][1]==16)
                /* TWO-SIXTEEN NOTE */
               {
                Note = 2;
                checkNote(address,i,2,_row);
                for(count=0;count<=1;count++)
                   {
                    detecLine1(_row[count],position,&Line);
                    Makedata(Line,Note,ACC);
                    //printf("Notenumber %d = %d,%c\n",i,Note,Line);
                   }
               }
              else
if((con[0][0]==12&&con[1][1]==16)||(con[0][0]==22&&con[1][1]==16))
                /* TREE-EIGHT NOTE */
               {
                Note = 4;
                checkNote(address,i,3,_row);
                for(count=0;count<=2;count++)
                   {
                    detecLine1(_row[count],position,&Line);
                    Makedata(Line,Note,ACC);
                    //printf("Notenumber %d = %d,%c\n",i,Note,Line);
                   }
               }
              else if(con[1][1]==26&&con[1][2]==18&&con[3][1]==16)
                /* TREE-SIXTEEN NOTE */
               {
                Note = 2;
                checkNote(address,i,3,_row);
                for(count=0;count<=2;count++)
                   {
                    detecLine1(_row[count],position,&Line);
                    Makedata(Line,Note,ACC);
                    //printf("Notenumber %d = %d,%c\n",i,Note,Line);
                   }
               }
```

```
                else if(con[0][0]==22&&con[1][1]==16&&con[3][1]==16)
                  /* FOUR-EIGHT NOTE */
                  {
                   Note = 4;
                   checkNote(address,i,4,_row);
                   for(count=0;count<=3;count++)
                        {
                         detecLine1(_row[count],position,&Line);
                         Makedata(Line,Note,ACC);
                         //printf("Notenumber %d = %d,%c\n",i,Note,Line);
                        }
                  }
                 else
if(con[1][1]==26&&con[1][2]==18&&con[3][1]==26&&con[3][2]==18)
                   /* FOUR-SIXTEEN NOTE */
                   {
                    Note = 2;
                    checkNote(address,i,4,_row);
                    for(count=0;count<=3;count++)
                        {
                         detecLine1(_row[count],position,&Line);
                         Makedata(Line,Note,ACC);
                         //printf("Notenumber %d = %d,%c\n",i,Note,Line);
                        }
                   }
                  break;
           case 7 : if(con[1][1]==16)  /* EIGHT NOTE */
                   {
                    Note = 4;
                    detecLine(strow+width+2,position,&Line);
                    Makedata(Line,Note,ACC);
                   }
                  else if(con[1][1]==36) /* SIXTEEN NOTE */
                   {
                    Note = 2;
                    detecLine(strow+width+2,position,&Line);
                    Makedata(Line,Note,ACC);
                   }
                  else if(con[2][1]==16) /* SIXTEEN NOTE REST */
                   {
                    Note = 2;
                    Line = 'N';
                    Makedata(Line,Note,ACC);
                   }
                  break;
           case 8 : if(con[1][0]==0&&con[1][1]==16&&con[3][0]==12)  /* NATURAL */
                     ACC = 1;
                    else if(con[1][0]==12&&con[1][1]==16&&con[2][0]==12
                          &&con[2][1]==16&&con[3][0]==12&&con[3][1]==16) /*
 SHARP */
                     ACC = 2;
```

```
                else if(con[1][1]==0) /* FLAT */
                  ACC = 3;
                break;
          case 9 : if(con[1][1]==0) /* EIGHT NOTE REST */
                  {
                   checkNote(address,i,2,_row);
                   if(_col[0]==_col[1])
                       {
                        break;
                       }
                   else
                       {
                        Note = 4;
                        Line = 'N';
                        Makedata(Line,Note,ACC);
                       }
                  }
                  else if(con[1][1]==16&&con[3][0]==12) /* QUARTER NOTE REST
*/
                  {
                   Note = 8;
                   Line = 'N';
                   Makedata(Line,Note,ACC);
                  }
                 break;
          case 10 : break;
          case 11 : break;
          case 12 : break;
          default : break;
         }
    }

  plays(s,d,numnote);
}


void set_neighbor2(buffer,row,col,neighbor)
  unsigned char *buffer;
  unsigned char neighbor[];
  unsigned int row,col;

{
  buffer = addr(row,col);
  neighbor[1] = *buffer;
}


detecLine(row,position,Line)
  unsigned int  row;
  unsigned char *Line;
  int position[15][4];
```

```
{
  int leger[4];
  leger[0] = position[0][2]-(position[1][2]-position[0][2]);
  leger[1] = position[0][2]-(2*(position[1][2]-position[0][2]));
  leger[2] = position[4][2]+(position[1][2]-position[0][2]);
  leger[3] = position[4][2]+(2*(position[1][2]-position[0][2]));
  row = row - 1;
  if(row > leger[1] && row < leger[0])
    *Line = 'Z';
  else if(row == leger[0])
    *Line = 'B';
  else if(row > leger[0] && row < position[0][2])
    *Line = 'A';
  else if(row == position[0][2])
    *Line = 'G';
  else if(row > position[0][2] && row < position[1][2])
    *Line = 'F';
  else if(row == position[1][2])
    *Line = 'E';
  else if(row > position[1][2] && row < position[2][2])
    *Line = 'D';
  else if(row == position[2][2])
    *Line = 'C';
  else if(row > position[2][2] && row < position[3][2])
    *Line = 'b';
  else if(row == position[3][2])
    *Line = 'a';
  else if(row > position[3][2] && row < position[4][2])
    *Line = 'g';
  else if(row == position[4][2])
    *Line = 'f';
  else if(row > position[4][2] && row < leger[2])
    *Line = 'e';
  else if(row == leger[2])
    *Line = 'd';
  else if(row > leger[2] && row < leger[3])
    *Line = 'c';
  else if(row == leger[3])
    *Line = 'y';
  else if(row > leger[3])
    *Line = 'x';
  return 0;
}


detecLine1(endrow,position,Line)
  unsigned int  endrow;
  unsigned char *Line;
  int position[15][4];
{
  int leger[4];
```

```
 leger[0] = position[0][2]-(position[1][2]-position[0][2]);
 leger[1] = position[0][2]-(2*(position[1][2]-position[0][2]));
 leger[2] = position[4][2]+(position[1][2]-position[0][2]);
 leger[3] = position[4][2]+(2*(position[1][2]-position[0][2]));
 endrow = endrow - 1;
 if(endrow >= leger[1]-2 && endrow <= leger[1]+2)
  *Line = 'Z';
 else if(endrow > leger[1]+2 && endrow < leger[0]-2)
  *Line = 'B';
 else if(endrow >= leger[0]-2 && endrow <= leger[0]+2)
  *Line = 'A';
 else if(endrow > leger[0]+2 && endrow < position[0][2]-2)
  *Line = 'G';
 else if(endrow >= position[0][2]-2 && endrow <= position[0][2]+2)
  *Line = 'F';
 else if(endrow > position[0][2]+2 && endrow < position[1][2]-2)
  *Line = 'E';
 else if(endrow >= position[1][2]-2 && endrow <= position[1][2]+2)
  *Line = 'D';
 else if(endrow > position[1][2]+2 && endrow < position[2][2]-2)
  *Line = 'C';
 else if(endrow >= position[2][2]-2 && endrow <= position[2][2]+2)
  *Line = 'b';
 else if(endrow > position[2][2]+2 && endrow < position[3][2]-2)
  *Line = 'a';
 else if(endrow >= position[3][2]-2 && endrow <= position[3][2]+2)
  *Line = 'g';
 else if(endrow > position[3][2]+2 && endrow < position[4][2]-2)
  *Line = 'f';
 else if(endrow >= position[4][2]-2 && endrow <= position[4][2]+2)
  *Line = 'e';
 else if(endrow > position[4][2]+2 && endrow < leger[2]-2)
  *Line = 'd';
 else if(endrow >= leger[2]-2 && endrow <= leger[2]+2)
  *Line = 'c';
 else if(endrow > leger[2]+2 && endrow < leger[3]-2)
  *Line = 'y';
 else if(endrow >= leger[3]-2 && endrow <= leger[3]+2)
  *Line = 'x';
 return 0;
}


checkNote(address,numNote,k,_row)
  unsigned char *address[101][2];
  unsigned int numNote,k;
  int _row[3];
{
  unsigned int  row,col,strow,endrow,stcol,endcol;
  unsigned char ne[1],*buffer;
  int c,i;
```

```
  int count = 0;
  addr_to_coor(address[numNote][0],&strow,&stcol);
  addr_to_coor(address[numNote][1],&endrow,&endcol);
  c = (endcol-stcol)/k;
  for(i=0;i<=(k-1);i++)
    for(row=strow+1;row<=endrow;row++)
        for(col=stcol+(i*c)+1;col<=stcol+(i*c)+c;col++)
          {
           set_neighbor2(buffer,row,col,ne);
           buffer=addr(row,col);
           if(ne[1]==2)
            {
              _row[count] = row;
              count++;
            }
          }
  return 0;
}


Makedata(unsigned char Line,int Note,int ACC)

{
  if(Line == 'Z')
    s[numnote] = _C4;
  else if(Line == 'B')
    s[numnote] = _B3;
  else if(Line == 'A')
    s[numnote] = _A3;
  else if(Line == 'G')
    s[numnote] = _G3;
  else if(Line == 'F')
    s[numnote] = _F3;
  else if(Line == 'E')
    s[numnote] = _E3;
  else if(Line == 'D')
    s[numnote] = _D3;
  else if(Line == 'C')
    s[numnote] = _C3;
  else if(Line == 'b')
    s[numnote] = _B2;
  else if(Line == 'a')
    s[numnote] = _A2;
  else if(Line == 'g')
    s[numnote] = _G2;
  else if(Line == 'f')
    s[numnote] = _F2;
  else if(Line == 'e')
    s[numnote] = _E2;
  else if(Line == 'd')
    s[numnote] = _D2;
```

```
else if(Line == 'c')
  s[numnote] = _C2;
else if(Line == 'y')
  s[numnote] = _B1;
else if(Line == 'x')
  s[numnote] = _A1;
else if(Line == 'N')
  s[numnote] = _null;
if(Note == 32)
  d[numnote] = 32;   /*WHOLE NOTE*/
else if(Note == 16)
  d[numnote] = 16;   /*HALF NOTE*/
else if(Note == 8)
  d[numnote] = 8;    /*QUARTER NOTE*/
else if(Note == 4)
  d[numnote] = 4;    /*EIGHT NOTE*/
else if(Note == 2)
  d[numnote] = 2;    /*SIXTEEN NOTE*/
if(ACC == 2)
  s[numnote] = s[numnote]+1;
else if(ACC == 3)
  s[numnote] = s[numnote]-1;
numnote++;
return 0;
}

/****************************  SAVENOTE  ****************************/

void savespk(void)

{
  FILE *stream;
  int i,l,n;
  char spkname[13]="";
  int botton;
  mouse_hide();
  setfillstyle(SOLID_FILL,BLUE);
  bar(0,120,639,479);
  border(150,230,400,330,"Save file");
  setfillstyle(SOLID_FILL,WHITE);
  bar(150,230,400,330);
  setcolor(BLACK);
  printfxy(150,240,400,260,"Enter file *.spk");
  setcolor(BLUE);
  outtextxy(160,300,"file name  : ");
  mouse_display();
  stringin(270,300,spkname);
  border(150,230,400,330,"Confirm");
  setfillstyle(SOLID_FILL,WHITE);
```

```
bar(150,230,400,330);
setcolor(BLACK);
printfxy(150,240,400,260,"Are you sure ?");
box(200,280,250,310,"Yes");
box(300,280,350,310,"No");
do
 {
  m = getmouse();
  if(m.status==MOUSE_LEFT)
   {
   if((m.x>200)&&(m.x<250)&&(m.y>280)&&(m.y<310))
       botton = 1;
   else if((m.x>300)&&(m.x<350)&&(m.y>280)&&(m.y<310))
       botton = 2;
   else continue;
   switch(botton)
       {
        case 1 :  mouse_hide();
                  box1(200,280,250,310,"Yes");
                  mouse_display();
                  do{
                  m = getmouse();
                  }while((m.status == MOUSE_LEFT)&&((m.x>200)
                    &&(m.x<250)&&(m.y>280)&&(m.y<310)));
                  mouse_hide();
                  box(200,280,250,310,"Yes");
                  mouse_display();
                  if((m.x>200)&&(m.x<250)&&(m.y>280)&&(m.y<310))
                   {
                    if((stream = fopen(spkname,"a+b")) == NULL)
                       {
                        printf("Error in open file\n");
                        exit(1);
                       }
                    for(i=0;i<=numnote-1;i++)
                       {
                        l = s[i];
                        n = d[i];
                        fwrite(&l,2,1,stream);
                        fwrite(&n,2,1,stream);
                       }
                    fclose(stream);
                    mouse_hide();
                    setfillstyle(SOLID_FILL,BLUE);
                    bar(0,120,639,479);
                    border(200,230,450,330,"Massege");
                    setfillstyle(SOLID_FILL,WHITE);
                    bar(200,230,450,330);
                    setcolor(BLACK);
                    printfxy(200,240,450,260,"Save file finish");
                    botton_ok();
```

```
                shadow(254,58,"Save");
                break;
               }
        case 2 :  mouse_hide();
                box1(300,280,350,310,"No");
                mouse_display();
                do{
                m = getmouse();
                }while((m.status == MOUSE_LEFT)&&((m.x>300)
                  &&(m.x<350)&&(m.y>280)&&(m.y<310)));
                box(300,280,350,310,"No");
                if((m.x>300)&&(m.x<350)&&(m.y>280)&&(m.y<310))
                  {
                   mouse_hide();
                   setfillstyle(SOLID_FILL,BLUE);
                   bar(0,120,639,479);
                   mouse_display();
                   m.status=9;
                   setcolor(BLACK);
                   shadow(54,58,"Open");
                   shadow(254,58,"Save");
                   shadow(342,58,"Convert");
                   shadow(454,58,"Play");
                   shadow(554,58,"Exit");
                   break;
                  }

        }
    }

   }while(m.status!=9);


}

/*****************************      CONVERT      *****************************/

void convert_mid(void)
 {
  int i=0;
  char convertname[13]="";
  char spkname[13]="", sngname[13]="", midname[13]="";
  char sng[]=".sng", mid[]=".mid";


  mouse_hide();
  setfillstyle(SOLID_FILL,BLUE);
  bar(0,120,639,479);
  border(170,200,460,350,"Convert");
  setfillstyle(SOLID_FILL,WHITE);
```

```
  bar(170,200,460,350);
  setcolor(BLACK);
  printfxy(170,220,460,240,"Convert file *.spk to *.sng");
  printfxy(170,240,460,260,"&");
  printfxy(170,260,460,280,"Convert file *.sng to *.mid");
  setcolor(RED);
  printfxy(170,285,460,305,"Enter file *.spk");
  setcolor(BLUE);
  outtextxy(185,320,"file name *.spk : ");
  mouse_display();
  stringin(340,320,spkname);
  for(i=0;i<=12;i++)
   {
    if(spkname[i]!='.')
         convertname[i] = spkname[i];
   }
  strcpy(sngname,convertname);
  strcat(sngname,sng);
  strcpy(midname,convertname);
  strcat(midname,mid);
  convert_spk_sng(spkname,sngname,midname);
}

void convert_spk_sng(char spkname[13],char sngname[13],char midname[13])
{
   int metrate, meter, pitchbend, exclusive, channel, i, midivol,
        event_count, active, divisor, time, nevent,inst;
   long numevents;
   char event_data[5],val[3];
   char *song_title = "Note from Recognition of printed music note.";
   char *track1_name = "TestTrak";
   char *empty_name = "<    >";
   int l,n,m,_numnote,a[999],b[999];
   int ii = 0;
   FILE *stream;
   FILE *_stream;

   do
    {
     mouse_hide();
     setfillstyle(SOLID_FILL,BLUE);
     bar(0,120,639,479);
     border(170,200,460,350,"Select");
     setfillstyle(SOLID_FILL,WHITE);
     bar(170,200,460,350);
     setcolor(BLACK);
     printfxy(170,240,460,260,"Select Program Number");
     setcolor(BLUE);
     outtextxy(200,320,"Enter (1-128) : ");
     mouse_display();
     val[0]=0;
```

```
 stringin(330,320,val);
 inst = atoi(val);
}while(inst>128||inst<1);
metrate = 100;
meter = METER;
pitchbend = FALSE;
exclusive = FALSE;
channel = 0;
midivol = 100;
numevents = NUM_EVENTS;
active = FALSE;
if((_stream = fopen(spkname,"rb")) == NULL)
 {
  mouse_hide();
  setfillstyle(SOLID_FILL,BLUE);
  bar(0,120,639,479);
  border(200,230,450,330,"Massege");
  setfillstyle(SOLID_FILL,WHITE);
  bar(200,230,450,330);
  setcolor(BLACK);
  printfxy(200,240,450,260,"Can' open file      ");
  outtextxy(365,248,spkname);
  setcolor(DARKGRAY);
  outtextxy(54,58,"Open");
  outtextxy(154,58,"Step");
  outtextxy(254,58,"Save");
  outtextxy(342,58,"Convert");
  outtextxy(454,58,"Play");
  outtextxy(554,58,"Exit");
  botton_ok();
  mouse_display();
 }
else
 {
  while(fread(&l,2,1,_stream) == 1)
      {
       if(ferror(_stream))
         {
          printf("Error in file\n");
          exit(1);
         }
       fread(&n,2,1,_stream);
       a[ii] = l;
       b[ii] = n;
       ii++;
      }
  fclose(_stream);
 stream = fopen(sngname, "wb");
 if (stream == NULL)
  {
   mouse_hide();
```

```
  setfillstyle(SOLID_FILL,BLUE);
  bar(0,120,639,479);
  border(200,230,450,330,"Massege");
  setfillstyle(SOLID_FILL,WHITE);
  bar(200,230,450,330);
  setcolor(BLACK);
  printfxy(200,240,450,260,"Can' open file      ");
  outtextxy(365,248,sngname);
  botton_ok();
  mouse_display();
}
else{
putc_to_file(song_title, 51, stream);        /* store common data */
puti_to_file(&metrate, 1, stream);
puti_to_file(&meter, 1, stream);
puti_to_file(&pitchbend, 1, stream);
puti_to_file(&exclusive, 1, stream);

putc_to_file(track1_name, 9, stream);        /* store track 1 header */
puti_to_file(&channel, 1, stream);
putl_to_file(&numevents, 1, stream);
puti_to_file(&active, 1, stream);
puti_to_file(&midivol, 1, stream);

numevents = 1;
for (i = 1; i < NTRACK; i++){        /* store other track's headers */
      putc_to_file(empty_name, 9, stream);
      puti_to_file(&channel, 1, stream);
      putl_to_file(&numevents, 1, stream);
      puti_to_file(&active, 1, stream);
      puti_to_file(&midivol, 1, stream);
}

time = 0;

/* write track 1 midi data */

write_event(stream, 2, 0, MES_END, 0, 0);  /* starting mes_end event */
write_event(stream, 3, 0, 192, inst, 0);
event_count = 1;
ii = 0;
convert_note(a[ii],b[ii]);
nevent = 1;
while (event_count < NUM_EVENTS - 2){
      while (time < meter * TICKS_PER_BEAT)   /* add mes_end if needed */
        {
         lenght = lenght-16;
         if(lenght==(-8))
          {
           if(nevent==1)
              {
```

```
                write_event(stream, 4, 0, NOTE_ON + channel, note, 64);
                event_count += 1;
                nevent = 2;
               }
            if(nevent==2)
               {
                write_event(stream, 4, TICKS_PER_BEAT, NOTE_ON +
channel,note,0);
                event_count += 1;
                time += TICKS_PER_BEAT;
               }
            else if(nevent==10)
               {
                if(time==0)
                  {
                   write_event(stream, 4, TICKS_PER_BEAT, NOTE_ON +
channel,note,0);
                   event_count += 1;
                   time += TICKS_PER_BEAT;
                  }
                 else
                  {
                   time = 0;
                   write_event(stream, 1, 248, 0, 0, 0);
                   write_event(stream, 4, 0, NOTE_ON + channel,note,0);
                   event_count += 2;
                   time += (TICKS_PER_BEAT*2);
                  }
               }
             ii++;
             convert_note(a[ii],b[ii]);
             nevent = 1;
            }
           else if(lenght>0)
           {
            if(nevent==1)
               {
                write_event(stream, 4, 0, NOTE_ON + channel, note, 64);
                event_count += 1;
                nevent = 2;
               }
             write_event(stream, 1, 248, 0, 0, 0);
             event_count += 1;
             time += (TICKS_PER_BEAT*2);
           }
           else if(lenght==0)
           {
            if(nevent==1)
               {
                write_event(stream, 4, 0, NOTE_ON + channel, note, 64);
                write_event(stream, 1, 248, 0, 0, 0);
```

```
                time += (TICKS_PER_BEAT*2);
                event_count += 2;
                nevent = 2;
               }
             else if(nevent==2)
                {
                write_event(stream, 1, 248, 0, 0, 0);
                write_event(stream, 4, 0, NOTE_ON + channel,note,0);
                event_count += 1;
                time += (TICKS_PER_BEAT*2);
                ii++;
                convert_note(a[ii],b[ii]);
                nevent = 1;
               }
            }
           else if(lenght==(-12))
            {
             write_event(stream, 4, 0, NOTE_ON + channel, note, 64);
             write_event(stream, 4, (TICKS_PER_BEAT/2), NOTE_ON +
channel,note,0);
             event_count += 2;
             time += (TICKS_PER_BEAT/2);
             ii++;
             convert_note(a[ii],b[ii]);
            }
           else if(lenght==(-14))
            {
             write_event(stream, 4, 0, NOTE_ON + channel, note, 64);
             write_event(stream, 4, (TICKS_PER_BEAT/4), NOTE_ON +
channel,note,0);
             event_count += 2;
             time += (TICKS_PER_BEAT/4);
             ii++;
             convert_note(a[ii],b[ii]);
            }
           else if(lenght==(-16))
            {
             if(nevent==2)
                  write_event(stream, 4, 0, NOTE_ON + channel,note,0);
              else if(nevent==10)
                  {
                   if(time==0)
                    write_event(stream, 4, 0, NOTE_ON + channel,note,0);
                   else
                    {
                     time = 0;
                     write_event(stream, 4, TICKS_PER_BEAT, NOTE_ON +
channel,note,0);
                     time += TICKS_PER_BEAT;
                    }
                  }
```

```
            event_count += 1;
            ii++;
            convert_note(a[ii],b[ii]);
            nevent = 1;
           }
          }
           write_event(stream, 2, 0, MES_END, 0, 0);
           time -= meter * TICKS_PER_BEAT;
           if(nevent==2)
            nevent = 10;
           event_count += 1;


    }
    write_event(stream, 2, 0, ALL_END, 0, 0); /* write track terminator */

    for (i = 1; i < NTRACK; i++)    /* rest of tracks have only one event */
          write_event(stream, 2, 0, MES_END, 0, 0);

    fclose(stream);
    convert_sng_mid(sngname,midname);
   }
  }
}


/* Writes the five data bytes in an event with one call to putc_to_file() */
void
write_event(stream, nbytes, b0, b1, b2, b3)
FILE *stream;
int nbytes, b0, b1, b2, b3;
{
    char data[5];

    data[0] = nbytes;
    data[1] = b0;
    data[2] = b1;
    data[3] = b2;
    data[4] = b3;
    putc_to_file(data, 5, stream);
}


void
putc_to_file(addr, size, stream)        /* put near char data to stream */
char *addr;
int size;
FILE *stream;
{
    int i;
```

```
    for(i = 0; i < size; i++){
        fputc(*addr++, stream);
    }
}


void
puti_to_file(addr, size, stream)        /* put near int data to stream */
int *addr;
int size;
FILE *stream;
{
    int i, end;
    char *caddr;

    caddr = (char *)addr;
    end = size * sizeof(int);

    for(i = 0; i < end; i++){
        fputc(*caddr++, stream);
    }
}


void
putl_to_file(addr, size, stream)        /* put near long int data to stream */
long *addr;
int size;
FILE *stream;
{
    int i, end;
    char *caddr;

    caddr = (char *)addr;
    end = size * sizeof(long);

    for(i = 0; i < end; i++){
        fputc(*caddr++, stream);
    }
}

void convert_note(int a,int b)
{
 if(a==_A1)
   note=57;
 if(a==_B1)
   note=59;
 if(a==_C2)
   note=60;
 if(a==_D2)
```

```
  note=62;
if(a==_E2)
  note=64;
if(a==_F2)
  note=65;
if(a==_G2)
  note=67;
if(a==_A2)
  note=69;
if(a==_B2)
  note=71;
if(a==_C3)
  note=72;
if(a==_D3)
  note=74;
if(a==_E3)
  note=76;
if(a==_F3)
  note=77;
if(a==_G3)
  note=79;
if(a==_A3)
  note=81;
if(a==_B3)
  note=83;
if(a==_C4)
  note=84;
 lenght = b;
}

void convert_sng_mid(char sngname[13],char midname[13])
{
    char title[TITLE_WIDE], trackname[NTRACK][TRACK_NAME_WIDE];
    unsigned char *buf, *cp, *cp1, b[4], runstatus;
    int metrate, meter, ntrack, trk, midichan[NTRACK], i, n, at_end;
    long eventcount[NTRACK], event, ticks, msec_qnote;
    FILE *infile, *outfile;


    infile = open_file(sngname, "rb");      /* open the files specified on */
    outfile = open_file(midname, "wb");     /* the command line. */

    /* All of the data is first written to a memory buffer called buf. */
    /* When conversion is complete, the buffer is written to disk. */
    /* This allows the length of the buffer to be know prior to writing. */

    buf = (char *)malloc(NBYTES);
    if (buf == NULL){
        fputs("\nCould not allocate memory for track data.", stdout);
        exit(0);
    }
```

```
get_from_file(title, TITLE_WIDE, infile);   /* read infile header data */
get_from_file(&metrate, sizeof(int), infile);
get_from_file(&meter, sizeof(int), infile);
get_from_file(&n, sizeof(int), infile);     /* ignore pitchbend flag */
get_from_file(&n, sizeof(int), infile);     /* ignore exclusive flag */

for (i = 0; i < NTRACK; i++){
      get_from_file(trackname[i], TRACK_NAME_WIDE, infile);
      get_from_file(&midichan[i], sizeof(int), infile);
      get_from_file(&eventcount[i], sizeof(long), infile);
      get_from_file(&n, sizeof(int), infile); /* ignore play status */
      get_from_file(&n, sizeof(int), infile); /* ignore midi volume */
}

ntrack = 0;
for (i = 0; i < NTRACK; i++){   /* find number of tracks with data */
      if (eventcount[i] > 1)
         ntrack++;
}
write_mthd(ntrack + 1, outfile);    /* put header chunck to outfile */

cp = buf;         /* cp points to start of allocated memory area */

*cp++ = 0;        /* time sig., tempo and title track added first */
*cp++ = META;     /* note that data is written to buffer. */
*cp++ = TIMESIG;
*cp++ = 4;
*cp++ = (char)meter;
*cp++ = 2;        /* MT always uses quarter note for beat, etc. */
*cp++ = 24;
*cp++ = 8;

*cp++ = 0;
*cp++ = META;     /* tempo, most significant bytes first */
*cp++ = SETTEMPO;
msec_qnote = 60000000/metrate;     /* a long data type (4 bytes) */
cp1 = (char *)&msec_qnote;         /* cp1 points to long's memory area */
*cp++ = 3;                    /* 3 is fixed value for this META */
*cp++ = *(cp1 + 2);           /* write value in correct order */
*cp++ = *(cp1 + 1);           /* opposite to the 80x86 convention */
*cp++ = *cp1;

*cp++ = 0;                    /* song name as meta text event */
*cp++ = META;
*cp++ = TEXTEVENT;
*cp++ = TITLE_WIDE;
strcpy(cp, title);            /* title copied in one shot */
cp += TITLE_WIDE;             /* update pointer */

*cp++ = 0;                    /* end of title meta event */
```

```
*cp++ = META;
*cp++ = ENDTRACK;
*cp++ = 0;

put_to_file("MTrk", 4, outfile);    /* write first track chunk */
put_len((long)(cp - buf), outfile); /* write computed buffer length */
write_buf(buf, cp, outfile);        /* write whole buffer at once */

for (trk = 0; trk < NTRACK; trk++){
      if (eventcount[trk] > 1){
         cp = buf;          /* cp points back to start of memory buffer */
         *cp++ = 0;          /* track name as meta instrument name event */
         *cp++ = META;
         *cp++ = INSNAME;
         *cp++ = TRACK_NAME_WIDE;
         strcpy(cp, trackname[trk]);
         cp += TRACK_NAME_WIDE;

         *cp++ = 0;         /* track channel as MIDI channel prefix */
         *cp++ = META;
         *cp++ = CHANPREF;
         *cp++ = 1;
         *cp++ = midichan[trk];

         at_end = ticks = event = 0;
         runstatus = 0;
         while (event++ < eventcount[trk] && !at_end){
            fgetc(infile);
            for (i = 0; i < 4; i++){
               b[i] = (char) fgetc(infile);
            }
            if (b[1] == ALL_END)
               at_end = 1;
            else if (b[0] == TIME_OUT)
               ticks += 240;
            else{               /* convert event data to files format */
               ticks += b[0];
               if (b[1] >= 0x80 && b[1] <= 0xEF){  /* if MIDI channel */
                     cp = copy_var_len(ticks, cp);   /* voice message */
                     if (b[1] == runstatus)      /* check running status */
                        ;
                     else
                        *cp++ = b[1];
                     *cp++ = b[2];
                     if (b[1] < 0xC0 || b[1] >= 0xE0)
                        *cp++ = b[3];          /* if four byte message */
                     ticks = 0;
                     runstatus = b[1];
               }
            }
            if (cp - buf + 3 >= NBYTES){
```

```
                    fputs("\nTrack shortened, out of buffer space.", stdout);
                    break;
                }
            }
            *cp++ = 0;
            *cp++ = META;
            *cp++ = ENDTRACK;
            *cp++ = 0;

            put_to_file("MTrk", 4, outfile);    /* write track chunk */
            put_len((long)(cp - buf), outfile); /* write data length */
            write_buf(buf, cp, outfile);        /* write all data at once */
        }
    } /* for (trk... */
    fclose(infile);                 /* close files and exit to dos */
    fclose(outfile);
    mouse_hide();
    setfillstyle(SOLID_FILL,BLUE);
    bar(0,120,639,479);
    border(200,230,450,330,"Convert");
    setfillstyle(SOLID_FILL,WHITE);
    bar(200,230,450,330);
    setcolor(BLACK);
    printfxy(200,240,450,260,"Convert complete");
    mouse_display();
    botton_ok();
}


void
write_buf(sp, ep, outfile)
char *sp, *ep;
FILE *outfile;
{
    while (sp != ep)
        fputc(*sp++, outfile);
}


char
*copy_var_len(n, cp)
long n;
char *cp;
{
    register long buffer;

    buffer = n & 0x7F;
    while ((n >>= 7) > 0){
        buffer <<= 8;
        buffer |= 0x80;
        buffer += (n & 0x7F);
```

```
    }

    while (buffer & 0x80){
        *cp++ = (char) buffer;
        buffer >>= 8;
    }
    *cp++ = (char) buffer;
    return(cp);
}


void
put_len(n, stream)      /* chunk lengths are always 4 bytes long */
long n;
FILE *stream;
{
    char *cp;

    cp = (char *)&n;
    fputc(*(cp + 3), stream);
    fputc(*(cp + 2), stream);
    fputc(*(cp + 1), stream);
    fputc(*cp, stream);
}


void
write_mthd(ntrack, outfile)     /* write header chunk to output file */
int ntrack;
FILE *outfile;
{
    int i, n;

    put_to_file("MThd", 4, outfile);        /* MThd = chunk type */
    n = 0;
    for (i = 0; i < 3; i++)
        put_to_file(&n, 1, outfile);
    n = 6;
    put_to_file(&n, 1, outfile);            /* 00 00 00 06 = lenght*/
    n = 0;
    put_to_file(&n, 1, outfile);
    n = 1;
    put_to_file(&n, 1, outfile);            /* 00 01 = format */
    n = 0;
    put_to_file(&n, 1, outfile);
    put_to_file(&ntrack, 1, outfile);       /* 00 0n = number of tracks */
    put_to_file(&n, 1, outfile);
    n = 120;
    put_to_file(&n, 1, outfile);            /* 00 78 = 120 ticks/Q note */
```

```
}


void
get_from_file(addr, size, stream)   /* get data from stream, put into near */
void *addr;                         /* memory */
int size;
FILE *stream;
{
   int i;
   char *addr2;

   addr2 = (char *)addr;
   for(i = 0; i < size; i++){
         *addr2++ = fgetc(stream);
   }
}


void
put_to_file(addr, size, stream)     /* put near data to stream */
void *addr;
int size;
FILE *stream;
{
   int i;
   char *addr2;

   addr2 = (char *)addr;
   for(i = 0; i < size; i++){
      fputc(*addr2++, stream);
   }
}


FILE
*open_file(filename, status)
char *filename, *status;
{
   FILE *file;

   file = fopen(filename, status);
   if (file == NULL)
    {
      mouse_hide();
      setfillstyle(SOLID_FILL,BLUE);
      bar(0,120,639,479);
      border(200,230,450,330,"Massege");
      setfillstyle(SOLID_FILL,WHITE);
      bar(200,230,450,330);
```

```
      setcolor(BLACK);
      printfxy(200,240,450,260,"Can' open file      ");
      outtextxy(365,248,filename);
      botton_ok();
      mouse_display();
    }
   return(file);
}

/***************************      PLAYNOTE      ****************************/

void playspk(void)

{
  FILE *stream;
  char spkname[13]="";
  int l,n,m,_numnote,a[999],b[999];
  int i = 0;
  dir("*.spk",spkname);

  if((stream = fopen(spkname,"r+b")) == NULL)
    {
     mouse_hide();
     setfillstyle(SOLID_FILL,BLUE);
     bar(0,120,639,479);
     border(200,230,450,330,"Massege");
     setfillstyle(SOLID_FILL,WHITE);
     bar(200,230,450,330);
     setcolor(BLACK);
     printfxy(200,240,450,260,"Can't open file      ");
     outtextxy(360,248,spkname);
     botton_ok();
     setcolor(DARKGRAY);
  shadow(54,58,"Open");
  shadow(342,58,"Convert");
  shadow(454,58,"Play");
  shadow(554,58,"Exit");
     mouse_display();
    }
  else
    {
     while(fread(&l,2,1,stream) == 1)
         {
          if(ferror(stream))
            {
             mouse_hide();
             setfillstyle(SOLID_FILL,BLUE);
             bar(0,120,639,479);
             border(200,230,450,330,"Massege");
```

```
            setfillstyle(SOLID_FILL,WHITE);
            bar(200,230,450,330);
            setcolor(BLACK);
            printfxy(200,240,450,260,"Error to open file      ");
            outtextxy(360,248,spkname);
            botton_ok();
            mouse_display();
            break;
           }
          fread(&n,2,1,stream);
          a[i] = l;
          b[i] = n;
          i++;
         }
      mouse_hide();
      setfillstyle(SOLID_FILL,BLUE);
      bar(0,120,639,479);
      setcolor(BLACK);
      shadow(54,58,"Open");
      shadow(454,58,"Play");
      shadow(554,58,"Exit");
      mouse_display();
      plays(a,b,i);
      fclose(stream);
    }
}
```

ภาคผนวก ข

The MIDI 1.0 Specification

The MIDI 1.0 Specification*

Introduction

MIDI is the acronym for Musical Instrument Digital Interface.

MIDI enables synthesizers, sequencers, home computers, rhythm machines, etc. to be interconnected through a standard interface.

Each MIDI-equipped instrument usually contains a receiver and a transmitter. Some instruments may contain only a receiver or transmitter. The receiver receives messages in MIDI format and executes MIDI commands. It consists of an opto-isolator, Universal Asynchronous Receiver/Transmitter (UART), and other hardware needed to perform the intended functions. The transmitter originates messages in MIDI format, and transmits them by way of a UART and line driver.

The MIDI standard hardware and data format are defined in this specification.

Conventions

Status and Data bytes given in Tables I through VI are given in binary.

Numbers followed by an "H" are in hexadecimal.

All other numbers are in decimal.

---

*MIDI Manufacturers Association, 5316 West 57th Street, Los Angeles, CA, 90056

Hardware

The interface operates at 31.25 (± 1%) Kbaud, asynchronous, with a start bit, 8 data bits (D0 to D7), and a stop bit. This makes a total of 10 bits for a period of 320 microseconds per serial byte.

Circuit: 5 mA current loop type. Logical 0 is current ON. One output shall drive one and only one input. The receiver shall be opto-isolated and require less than 5 mA to turn on. Sharp PC-900 and HP 6N138 opto-isolators have been found acceptable. Other high-speed opto-isolators may be satisfactory. Rise and fall times should be less than 2 microseconds.

Connectors: DIN 5 pin (180 degree) female panel mount receptacle. An example is the SWITCHCRAFT 57 GB5F. The connectors shall be labelled "MIDI IN" and "MIDI OUT." Note that pins 1 and 3 are not used, and should be left unconnected in the receiver and transmitter.

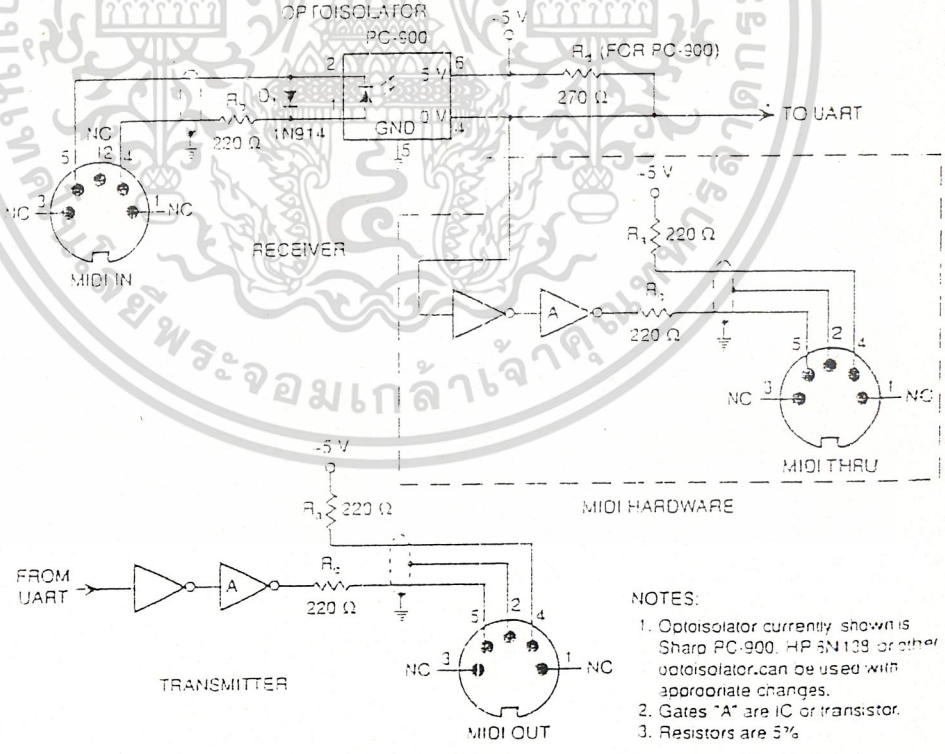


Fig. A-1. Standard hardware configuration for MIDI in, out, and thru ports.

Cables shall have a maximum length of fifty feet (15 meters), and shall be terminated on each end by a corresponding 5-pin DIN male plug, such as the SWITCHCRAFT 05GM5M. The cable shall be shielded twisted pair, with the shield connected to pin 2 at both ends.

A "MIDI THRU" output may be provided if needed, which provides a direct copy of data coming in MIDI IN. For very long chain lengths (more than three instruments), higher-speed opto-isolators must be used to avoid additive rise/fall time errors which affect pulse width duty cycle.

Data Format

All MIDI communication is achieved through multi-byte "messages" consisting of one Status byte followed by one or two Data bytes, except Real-Time and Exclusive messages (see below).

Message Types

Messages are divided into two main categories: Channel and System.

Channel

Channel messages contain a four-bit number in the Status byte which address the message specifically to one of sixteen channels. These messages are thereby intended for any units in a system whose channel number matches the channel number encoded into the Status byte.

There are two types of Channel messages: Voice and Mode.

Voice–To control the instrument's voices. Voice messages are sent over the Voice Channels.

Mode–To define the instrument's response to Voice messages. Mode messages are sent over the instrument's Basic Channel.

System

System messages are not encoded with channel numbers.

There are three types of System messages: Common, Real-Time, and Exclusive.

Common–Common messages are intended for all units in a system.

Real-Time–Real-Time messages are intended for all units in a system.

They contain Status bytes only—no Data bytes. Real-Time messages may be sent at any time—even between bytes of a message which has a different status. In such cases the Real-Time message is either ignored or acted upon, after which the receiving process resumes under the previous status.

Exclusive–Exclusive messages can contain any number of Data bytes, and are terminated by an End of Exclusive (EOX) or any other Status byte. These messages include a Manufacturer's Identification (ID) code. If the receiver does not recognize the ID code, it should ignore the ensuing data.

So that other users can fully access MIDI instruments, manufacturers should publish the format of data following their ID code. Only the manufacturer can update the format following their ID.

Data Types

Status Bytes

Status bytes are eight-bit binary numbers in which the Most Significant Bit (MSB) is set (binary 1). Status bytes serve to identify the message type; that is, the purpose of the Data bytes which follow the Status byte.

Except for Real-Time messages, new Status bytes will always command the receiver to adopt their status, even if the new Status is received before the last message was completed.

Running Status—For Voice and Mode messages only, when a Status byte is received and processed, the receiver will remain in that status until a different Status byte is received. Therefore, if the same Status byte would be repeated, it may (optionally) be omitted so that only the correct number of Data bytes need be sent. Under Running Status, then, a complete message need only consist of specified Data bytes sent in the specified order.

The Running Status feature is especially useful for communicating long strings of Note On/Off messages, where "Note On with Velocity of 0" is used for Note Off. (A separate Note Off Status byte is also available.)

Running Status will be stopped when any other Status byte intervenes, except that Real-Time messages will only interrupt the Running Status temporarily.

Unimplemented Status—Any status bytes received for functions which the receiver has not implemented should be ignored, and subsequent data bytes ignored.

Undefined Status—Undefined Status bytes must not be used. Care should be taken to prevent illegal messages from being sent during power-up or power-down. If undefined Status bytes are received, they should be ignored, as should subsequent Data bytes.

Data Bytes

Following the Status byte, there are (except for Real-Time messages) one or two Data bytes which carry the content of the message. Data bytes are eight-bit binary numbers in which the MSB is reset (binary 0). The number and range of Data bytes which must follow each Status byte are specified in the tables which follow. For each Status byte the correct number of Data bytes must always be sent. Inside the receiver, action on the message should wait until all Data bytes required under the current status are received. Receivers should ignore Data bytes which have not been properly preceded by a valid Status byte (with the exception of "Running Status," above).

Channel Modes

Synthesizers contain sound generation elements called voices. Voice assignment is the algorithmic process of routing Note On, Off data from the keyboard to the voices so that the musical notes are correctly played with accurate timing.

When MIDI is implemented, the relationship between the sixteen available MIDI channels and the synthesizer's voice assignment must be defined. Several Mode messages are available for this purpose (see Table III). They are Omni (On/Off), Poly, and Mono. Poly and Mono are mutually exclusive, i.e., Poly Select disables Mono, and vice versa. Omni, when on, enables the receiver to receive Voice messages in all Voice Channels without discrimination. When Omni is off, the receiver will accept Voice messages from only the selected Voice Channel(s). Mono, when on, restricts the assignment of Voices to just one voice per Voice Channel (Monophonic.) When Mono is off (=Poly On), any number of voices may be allocated by the Receiver's normal voice assignment algorithm (Polyphonic)

For a receiver assigned to Basic Channel "N," the four possible modes arising from the two Mode messages are:

Mode	Omni		
1	On	Poly	Voice messages are received from all Voice Channels and assigned to voices polyphonically.
2	On	Mono	Voice messages are received from all Voice Channels, and control only one voice, monophonically.
3	Off	Poly	Voice messages are received in Voice Channel N only, and are assigned to voices polyphonically.
4	Off	Mono	Voice messages are received in Voice Channels N thru N+M−1, and assigned monophonically to voices 1 thru M, respectively. The number of voices M is specified by the third byte of the Mono Mode Message.

Four modes are applied to transmitters (also assigned to Basic Channel N). Transmitters with no channel selection capability will normally transmit on Basic Channel 1 (N=0).

Mode	Omni		
1	On	Poly	All voice messages are transmitted in Channel N.
2	On	Mono	Voice messages for one voice are sent in Channel N.
3	Off	Poly	Voice messages for all voices are sent in Channel N
4	Off	Mono	Voice messages for voices 1 thru M are transmitted in Voice Channels N thru N+M−1, respectively. (Single voice per channel.)

A MIDI receiver or transmitter can operate under one and only one mode at a time. Usually the receiver and transmitter will be in the same mode. If a mode cannot be honored by the receiver, it may ignore the message (and any subsequent data bytes), or it may switch to an alternate mode (usually Mode 1, Omni On/Poly).

Mode messages will be recognized by a receiver only when sent in the Basic Channel to which the receiver has been assigned, regardless of the current mode. Voice messages may be received in the Basic Channel and in other channels (which are all called Voice Channels), which are related specifically to the Basic Channel by the rules above, depending on which mode has been selected.

A MIDI receiver may be assigned to one or more Basic Channels by default or by user control. For example, an eight-voice synthesizer might be assigned to Basic Channel 1 on power-up. The user could then switch the instrument to be configured as two four-voice synthesizers, each assigned to its own Basic Channel. Separate Mode messages would then be sent to each four-voice synthesizer, just as if they were physically separate instruments.

Power-Up Default Conditions

On power-up all instruments should default to Mode # 1. Except for Note On, Off Status, all Voice messages should be disabled. Spurious or undefined transmissions must be suppressed.

Table I. Summary of Status Bytes

Status D7— D0	# of Data Bytes	Description
		Channel Voice Messages
1000nnnn	2	Note Off event
1001nnnn	2	Note On event (velocity=0: Note Off)
1010nnnn	2	Polyphonic key pressure/after touch
1011nnnn	2	Control change
1100nnnn	1	Program change
1101nnnn	1	Channel pressure/after touch
1110nnnn	2	Pitch bend change
		Channel Mode Messages
1011nnnn	2	Selects Channel Mode
		System Messages
11110000	* * * * *	System Exclusive
11110sss	0 to 2	System Common
11111ttt	0	System Real Time

Notes:

nnnn: N-1, where N = Channel #,
i.e., 0000 is Channel 1.
0001 is Channel 2.
.
.
.
1111 is Channel 16.

*****: 0iiiiiii, data, ..., EOX

iiiiiii: Identification

sss: 1 to 7

ttt: 0 to 7

Table II. Channel Voice Messages

Status	Data Bytes	Description
1000nnnn	0kkkkkkk	Note Off (see notes 1-4)
	0vvvvvvv	vvvvvvv: note off velocity
1001nnnn	0kkkkkkk	Note On (see notes 1-4)
	0vvvvvvv	vvvvvvv - 0: velocity
		vvvvvvv = 0: note off
1010nnnn	0kkkkkkk	Polyphonic Key Pressure (After-Touch)
	0vvvvvvv	vvvvvvv: pressure value
1011nnnn	0ccccccc	Control Change
	0vvvvvvv	ccccccc: control # (0-121)
		(see notes 5-8)
		vvvvvvv: control value
		ccccccc = 122 thru 127: Reserved.
		(See Table III)
1100nnnn	0ppppppp	Program Change
		ppppppp: program number (0-127)
1101nnnn	0vvvvvvv	Channel Pressure (After-Touch)
		vvvvvvv: pressure value
1110nnnn	0vvvvvvv	Pitch Bend Change LSB (see note 10)
	0vvvvvvv	Pitch Bend Change MSB

Notes:

1. nnnn: Voice Channel # (1-16, coded as defined in Table I notes)

2. kkkkkkk: note # (0 - 127)
   kkkkkkk = 60: Middle C of keyboard

0	12	24	36	48	60	72	84	96	108	120	127
ac	c	c	c	c	c	c	c				
|——— piano range ———|											

3. vvvvvvv: key velocity
   A logarithmic scale would be advisable.

0	1	64	127
off	ppp pp	p mp	mf f fff	ffff

   vvvvvvv = 64: in case of no velocity sensors
   vvvvvvv = 0: Note Off, with velocity = 64

4. Any Note On message sent should be balanced by sending a Note Off message for that note in that channel at some later time.

5. ccccccc: control number

ccccccc	Description
0	Continuous Controller 0 MSB
1	Continuous Controller 1 MSB (MODULATION BENDER)
2	Continuous Controller 2 MSB
3	Continuous Controller 3 MSB
4-31	Continuous Controllers 4-31 MSB
32	Continuous Controller 0 LSB
33	Continuous Controller 1 LSB (MODULATION BENDER)
34	Continuous Controller 2 LSB
35	Continuous Controller 3 LSB
36-63	Continuous Controllers 4-31 LSB
64-95	Switches (On/Off)
96-121	Undefined
122-127	Reserved for Channel Mode messages (see Table III).

6. All controllers are specifically defined by agreement of the MIDI Manufacturers Association (MMA) and the Japan MIDI Standards Committee (JMSC). Manufacturers can request through the MMA or JMSC that logical controllers be assigned to physical ones as necessary. The controller allocation table must be provided in the user's operation manual.

7. Continuous controllers are divided into Most Significant and Least Significant Bytes. If only seven bits of resolution are needed for any particular controllers, only the MSB is sent. It is not necessary to send the LSB. If more resolution is needed, then both are sent, first the MSB, then the LSB. If only the LSB has changed in value, the LSB may be sent without re-sending the MSB.

8. vvvvvvv: control value (MSB)
(for controllers)

0 ——————————————————— 127
min max

(for switches)

0 - - - - - - - - - - - 127
off on

Numbers 1 through 126, inclusive, are ignored.

9. Any messages (e.g., Note On), which are sent successively under the same status, can be sent without a Status byte until a different Status byte is needed.

10. Sensitivity of the pitch bender is selected in the receiver. Center position value (no pitch change) is 2000H, which would be transmitted EnH-00H-40H.

Table III. Channel Mode Messages

Status	Data Bytes	Description
1011nnnn	0ccccccc 0vvvvvvv	Mode Messages ccccccc = 122: Local Control vvvvvvv = 0, Local Control Off vvvvvvv = 127, Local Control On ccccccc = 123: All Notes Off vvvvvvv = 0 ccccccc = 124: Omni Mode Off (All Notes Off) vvvvvvv = 0 ccccccc = 125: Omni Mode On (All Notes Off) vvvvvvv = 0 ccccccc = 126: Mono Mode On (Poly Mode Off) (All Notes Off) vvvvvvv = M, where M is the number of channels. vvvvvvv = 0, the number of channels equals the number of voices in the receiver. ccccccc = 127: Poly Mode On (Mono Mode Off) vvvvvvv = 0 (All Notes Off)

Notes:

1. nnnn: Basic Channel # (1-16, coded as defined in Table I)

2. Messages 123 thru 127 function as All Notes Off messages. They will turn off all voices controlled by the assigned Basic Channel. Except for message 123, All Notes Off, they should not be sent periodically, but only for a specific purpose. In no case should they be used in lieu of Note Off commands to turn off notes which have been previously turned on. Therefore any All Notes Off command (123-127) may be ignored by receiver with no possibility of notes staying on, since any Note On command must have a corresponding specific Nore Off command.

3. Control Change #122, Local Control, is optionally used to interrupt the internal control path between the keyboard, for example, and the sound-generating circuitry. If 0 (Local Off message) is received, the path is disconnected: the keyboard data goes only to MIDI and the sound-generating circuitry is controlled only by incoming MIDI data. If a 7FH (Local On message) is received, normal operation is restored.

4. The third byte of "Mono" specifies the number of channels in which Monophonic Voice messages are to be sent. This number, "M", is a number between 1 and 16. The channel(s) being used, then, will be the current

Basic Channel (=N) thru N+M-1 up to a maximum of 16. If M=0, this is a special case directing the receiver to assign all its voices, one per channel, from the Basic Channel N through 16.

Table IV. System Common Messages

Status	Data Bytes	Description
11110001		Undefined
11110010		Song Position Pointer
	0lllllll	lllllll: (Least significant)
	0hhhhhhh	hhhhhhh: (Most significant)
11110011	0sssssss	Song Select
		sssssss: Song #
11110100		Undefined
11110101		Undefined
11110110	none	Tune Request
11110111	none	EOX: "End of System Exclusive" flag

1. Song Position Pointer: Is an internal register which holds the number of MIDI beats (1 beat = 6 MIDI clocks) since the start of the song. Normally it is set to 0 when the START switch is pressed, which starts sequence playback. It then increments with every sixth MIDI clock receipt, until STOP is pressed. If CONTINUE is pressed, it continues to increment. It can be arbitrarily preset (to a resolution of 1 beat) by the SONG POSITION POINTER message.

2. Song Select: Specifies which song or sequence is to be played upon receipt of a Start (Real-Time) message.

3. Tune Request: Used with analog synthesizers to request them to tune their oscillators.

4. EOX: Used as a flag to indicate the end of a System Exclusive transmission (see Table VI).

Table V. System Real Time Messages

Status	Data Bytes	Description
11111000		Timing Clock
11111001		Undefined
11111010		Start
11111011		Continue
11111100		Stop
11111101		Undefined
11111110		Active Sensing
11111111		System Reset

Notes:

1. The System Real Time messages are for synchronizing all of the system in real time.

2. The System Real Time messages can be sent at any time. Any messages which consist of two or more bytes may be split to insert Real Time messages.

3. Timing clock (F8H)
   The system is synchronized with this clock, which is sent at a rate of 24 clocks/quarter note.

4. Start (from the beginning of song) (FAH)
   This byte is immediately sent when the PLAY switch on the master (e.g., sequencer or rhythm unit) is pressed.

5. Continue (FBH)
   This is sent when the CONTINUE switch is hit. A sequence will continue at the time of the next clock.

6. Stop (FCH)
   This byte is immediately sent when the STOP switch is hit. It will stop the sequence.

7. Active Sensing (FEH)
   Use of this message is optional, for either receivers or transmitters. This is a "dummy" Status byte that is sent every 300 ms (max), whenever there is no other activity on MIDI. The receiver will operate normally if it never receives FEH. Otherwise, if FEH is ever received, the receiver will expect to receive FEH or a transmission of any type every 300 ms (max). If a period of 300 ms passes with no activity, the receiver will turn off the voices and return to normal operation.

8. System Reset (FFH)
   This message initializes all of the system to the condition of just having turned on power. The system Reset message should be used sparingly, preferably under manual command only. In particular, it should not be sent automatically on power up.

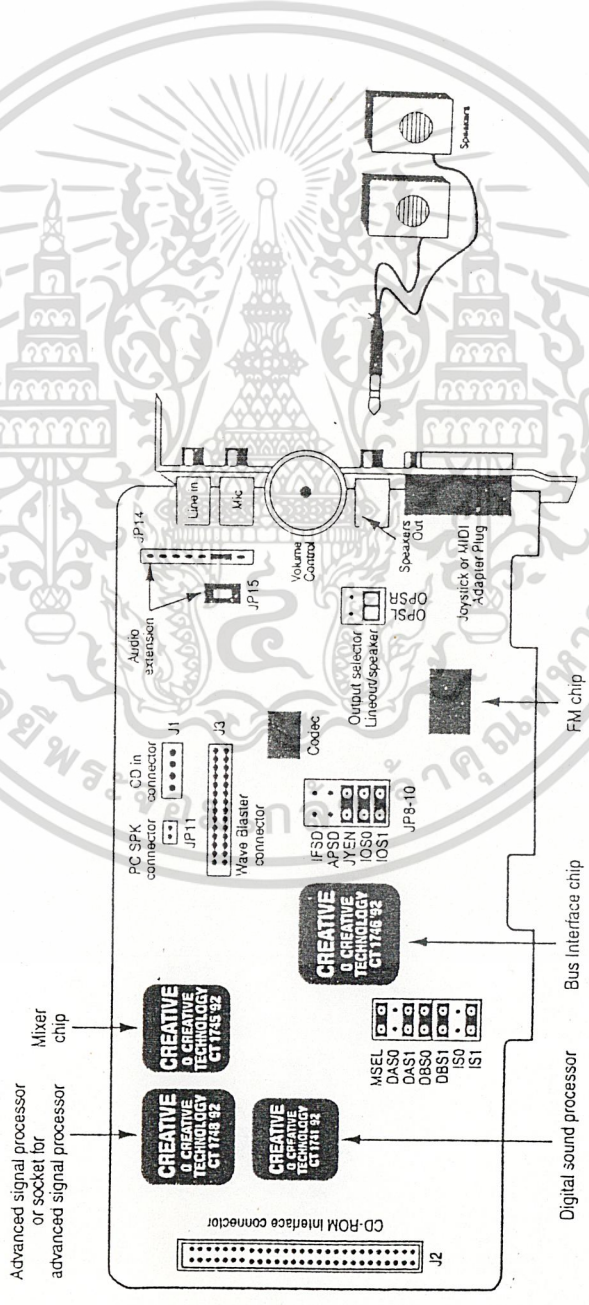
Table VI. System Exclusive Messages

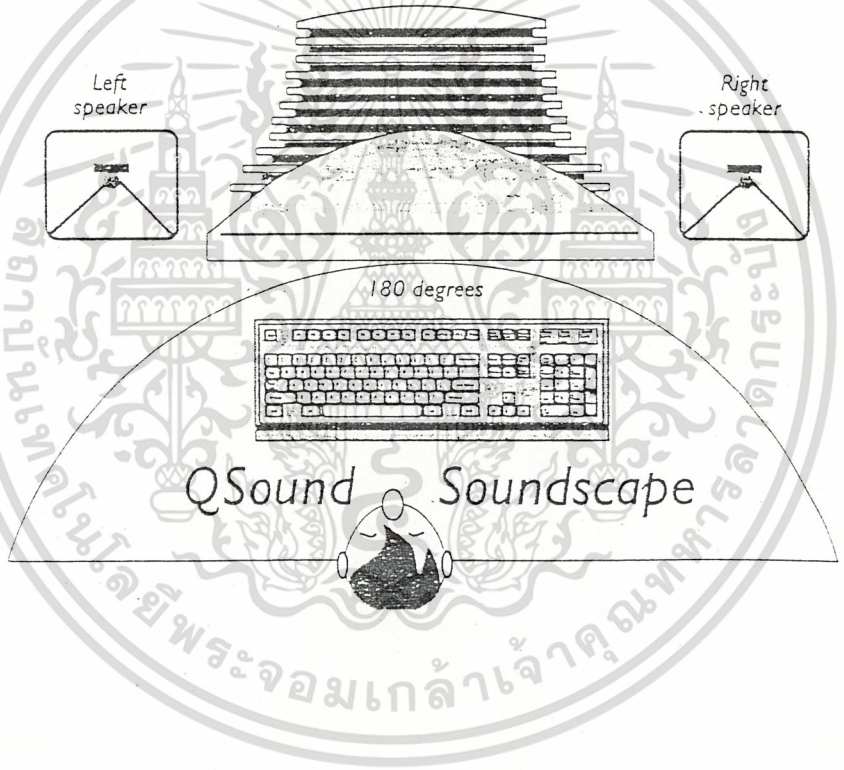
Status	Data Bytes	Description
11110000	0iiiiiii	Bulk dump etc. iiiiiii: identification
	(0*******)	Any number of bytes may be sent here, for any purpose, as long as they all have a zero in the most significant bit.
	(0*******)	
	11110111	EOX: "End of System Exclusive"

Notes:

1. iiiiiii: identification ID (0-127)
2. All bytes between the System Exclusive Status byte and EOX or the next Status byte must have zeroes in the MSB.
3. The ID number can be obtained from the MMA or JMSC.
4. In no case should other Status or Data bytes (except Real-Time) be interleaved with System Exclusive, regardless of whether or not the ID code

ภาคผนวก ค
ฮาร์ดแวร์และอุปกรณ์ต่อร่วม





กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลงได้ด้วยดีเนื่องจาก ได้รับคำแนะนำจาก อ. เกษตร์ ศิริสันติสัมฤทธิ์ ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงาน และ คุณ นพดล ฟูติตระกูล , คุณ มัฆวาน จันทร์กอฮอ ซึ่งเป็นรุ่นพี่ ที่กรุณา ให้ข้อมูล,อุปกรณ์ที่ใช้ทำโครงงาน ตลอดจนคำแนะนำที่ดี รวมทั้งขอขอบคุณเพื่อนๆที่คอยให้กำลังใจมาโดยตลอด

เอกสารอ้างอิง

1. สุรพันธุ์ เอื้อไพบูลย์, "การจดจำตัวอักษรลายมือเขียนภาษาไทยโดยการพิจารณาหัวของตัวอักษร",วิทยานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์มหาบัณฑิต,คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง,47-58 หน้า ,2531
2. Hymn,"ระบบมิดี้",วารสารคอมพิวเตอร์รีวิว,ปีที่ 9,ฉบับที่ 86,2534,หน้า 187-192
3. มนตรี พจนารถลาวัณย์, "การเขียนโปรแกรมคอมพิวเตอร์ด้วยเทอร์โบซี",2535
4. ธันวา ศรีประมง, "การเขียนโปรแกรมภาษาซี สำหรับวิศวกรรม",2537
5. ทรงชัย วีระทวีมาศ,เกรียงไกร ไชวเจริญสุข และ ครรชิต ไมตรี, " การปรับปรุงวิธีการหาค่าคุณสมบัติทางโทโปโลยีของภาพดิจิตอลทั่วไป",บทความในการประชุมทางวิชาการ วิศวกรรมไฟฟ้า 9 สถาบัน ครั้งที่ 13
6. ฟูศักดิ์ ชีวสุวิทย์, "วิธีการตรวจจับขอบวัตถุของหุ่นยนต์" ,การประชุมทางวิชาการ สถิติประยุกต์ครั้งที่ 5 C03-1 - C03-10 หน้า
7. Kimpan.C,"Printed Thai Characters Recognition",Dissertation for the degree of Doctor of Engineering,King Mongkut's Institue of Technology Chaokhun Taharn Ladkrabang,303-327 p.,1986
8. Rafael C.Gonzalaz and Paul Wintz,"Digital Image Processing",1-179 and 398-402 p.,1987
9. Louis J.Galbiati,Jr,"Machine Vision and Digital Image Processing Fandamentals",54-68 p.,1990
10. Herbert Schildt,"Terbo C/C++ : The Complete Reference,Second Edition",1992
11. Peter M.Ridge/David M.Golden/Ivan Luk and Scott Sindorf,"Sound Blaster : The Official Book,Second Edition",441-443 and 465-468 p.,1994