การจดจำรูปร่างวัตถุ
OBJECT RECOGNITION

ชาญชัย พิสิทธิวิทยานนท์
CHANCHAI PISITTIVITHAYANON

วิทยานิพนธ์สำหรับปริญญาวิศวกรรมศาสตร์มหาบัณฑิต
สาขาวิศวกรรมไฟฟ้า
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง
ปีการศึกษา 2531

การจดจำรูปร่างวัตถุ
OBJECT RECOGNITION

ชาญชัย พิสิทธิวิทยานนท์
CHANCHAI PISITTIVITHAYANON

อาจารย์ที่ปรึกษา
ผศ.ดร. ฟูศักดิ์ ชีวสุวิทย์
ADVISOR
FUSAK CHEEVASUVIT D.Eng. (ENST)

วิทยานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง
ปีการศึกษา 2531

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

แบบฟอร์มการให้คะแนนการสอบวิทยานิพนธ์

สำหรับนักศึกษาระดับมหาบัณฑิต

ชื่อนักศึกษา นายชาญชัย พิสิทธิวิทยานนท์ เลขประจำตัว 28.0004

ชื่อเรื่องวิทยานิพนธ์ การจดจำรูปร่างวัตถุ (Object Recognition)

ชื่ออาจารย์ผู้ควบคุมการสอบ	ลายมือชื่อ	ผลการสอบ
ผศ.ดร.ฟูศักดิ์ ธีวสุวิทย์		ดีเยี่ยม
รศ.กิตติ ตีรเศรษฐ		ดีเยี่ยม
รศ.ดร.ชม กิ้มปาน		ผ่าน
ดร.บุญวัฒน์ อัตชู		ดี

วันเดือนปี ที่สอบ 8 กรกฎาคม 2531 เวลา 10.00 น. สถานที่ ห้อง A-305

(นายทิฆัมพร รักไทยอดม)

คณบดีบัณฑิตวิทยาลัย

กิตติกรรมประกาศ

(ACKNOWLEDGMENT)

ขอกราบขอบพระคุณ คุณพ่อสมบูรณ์ พิสิทธิวิทยานนท์ คุณแม่มุกดา พิสิทธิวิทยานนท์ และ พี่ๆ ผู้ให้ความอุปการะในทุกๆด้าน

ขอกราบขอบพระคุณท่านอาจารย์ ผศ.ดร.ฟูศักดิ์ ชีวสุวิทย์ และ รศ.กิตติ ตีรเศรษฐ เป็นอย่างสูง ผู้ซึ่งประสาทวิชาความรู้ ตลอดจนให้คำปรึกษาแนะนำแนวทาง และวิธีการแก้ไข จนกระทั่งงานวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี รวมทั้งขอกราบขอบพระคุณท่านอาจารย์ รศ.มนัส สังวรศิลป์ คุณ สุรพันธ์ เอื้อไพบูลย์ เป็นอย่างสูง ที่กรุณาให้ความช่วยเหลือทางด้านข้อมูลภาพถ่าย ซึ่งนับได้ว่าเป็นสิ่งสำคัญอย่างยิ่ง ในการทำวิทยานิพนธ์ฉบับนี้

และขอบพระคุณ อาจารย์(ภาควิชา วิศวกรรมการวัดคุมฯ) เพื่อนๆ และ น้องๆ ที่ให้ความช่วยเหลือในสิ่งต่างๆ และ กำลังใจ

สารบัญ

หัวข้อวิทยานิพนธ์ การจดจำรูปร่างวัตถุ
นักศึกษา นาย ชาญชัย พิสิทธิวิทยานนท์
อาจารย์ที่ปรึกษา ผศ.ดร.ฐศักดิ์ ชีวสุวิทย์
ระดับการศึกษา วิศวกรรมศาสตรมหาบัณฑิตทางวิศวกรรมไฟฟ้า
ปีการศึกษา พ.ศ. 2531

บทคัดย่อ

การจดจำรูปร่างวัตถุจากภาพสองระดับซึ่งเป็นภาพ 2 มิตินั้น โดยปกติจะใช้วิธีจดจำรูปร่างวัตถุแบบทุกๆจุดภาพ ซึ่งทำให้สิ้นเปลืองหน่วยความจำมาก อีกทั้งการคำนวณหาสหสัมพันธ์เพื่อตรวจสอบชนิดของวัตถุ จะต้องใช้เวลาในการคำนวณนาน วิทยานิพนธ์ฉบับนี้ขอเสนอเทคนิคในการตรวจจับหาขอบของวัตถุจากภาพ 2 ระดับ โดยมีจุดประสงค์หลักเพื่อลดเวลาที่ใช้ในการตรวจสอบชนิดของวัตถุ และลดหน่วยความจำที่ใช้ และสามารถทำงานได้บนเครื่องไมโครคอมพิวเตอร์ IBM PC ขอบของวัตถุที่ได้จะถูกนำมาเข้ารหัสของ Freeman ก่อนที่จะทำการตรวจสอบชนิดของวัตถุนั้น โดยขอบของวัตถุจะถูกหมุนให้ได้แนวแกนหลักให้ตรงกับแนวแกนหลักของวัตถุอ้างอิง จากนั้นทำการหาสหสัมพันธ์รหัสลูกโซ่แบบมิติเดียวของขอบวัตถุใดๆกับวัตถุอ้างอิง ซึ่งจะทำให้สามารถลดทั้งเวลาในการคำนวณและลดขนาดของหน่วยความจำลงได้ และยังได้พัฒนาวิธีการจดจำให้ได้รหัสลูกโซ่ที่สั้นลงไปอีก โดยทำการตรวจจับหาจุดสำคัญซึ่งเป็นจุดเปลี่ยนแนวของเส้นตรงต่างๆที่ประกอบขึ้นเป็นขอบของวัตถุ ทำให้ได้รหัสลูกโซ่ของมุมกับระยะทาง รหัสลูกโซ่นี้จะถูกนำมาใช้ในการตรวจสอบชนิดของวัตถุด้วยวิธีการทำสหสัมพันธ์ ซึ่งสามารถลดเวลาในการคำนวณสหสัมพันธ์เพื่อการตรวจสอบชนิดของวัตถุ และลดหน่วยความจำลงได้เป็นอย่างมาก รหัสลูกโซ่นี้มีคุณสมบัติเด่นคือ สามารถใช้ในการตรวจสอบชนิดของวัตถุได้โดยไม่ต้องทำการหมุนแนวแกนหลักของวัตถุ ในตอนท้ายของวิทยานิพนธ์นี้ ได้แสดงตัวอย่างของการทดสอบหาวัตถุที่บางส่วนถูกบดบังด้วยวัตถุชนิดอื่น จากผลการทดสอบสามารถแยกแยะชนิดของวัตถุได้อย่างเป็นที่น่าพอใจ

Thesis Title Object Recognition
Name Chanchai PISITTIVITHAYANON
Thesis Advisor Fusak CHEEVASUVIT D.Eng.(ENST)
Level of Study MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING
Academic Year 1988

Abstract

The most important problem for robotic is how to identify an object. The classical technique for solving this problem uses the correlation method in order to obtain a maximum correlation coefficient. For the binary image, the correlation can be accomplished by superimpose any object to the reference object. A 2-dimensional correlation takes a very long computation times and consumes too much memory. The principle research in this thesis is to reduce the computation time and the memory size in the object identification process, and it must be implemented on a microcomputer IBM PC

In this thesis, the edge detection technique is developed and the Freeman code is used for the detected edge to form a chain code. Before identifying an object, we first rotate its principle axis to the principle axis of the reference object. The object is identified by using 1-dimentional correlation of the chain code of any object with the chain code of the reference object. By this way, the problem of computation time and memory size can be solved. We developed, furthermore, a technique of recognizing the break points on the edge around the object. So,

the chain code of angle and distance has been formed by these break points. We found that the quantity of data information is reduced, in consequence the memory size can be saved and the computation time will be greatly reduced. This kind of chain code gives a great advantage because an object can be identified without rotating its principle axis. The identification of an object with occlude parts is shown as an example in the last section of this thesis.

III, 5

บทที่ 1
บทนำ
(Introduction)

การพัฒนาทางด้านหุ่นยนต์นั้นควรที่จะพัฒนาไปทุกส่วนของตัวหุ่น ไม่เฉพาะแต่แขน ขา หรือเสียงพูด เราควรจะพัฒนาทางด้านตาเพื่อการมองเห็นของหุ่นยนต์ด้วย สำหรับในอดีตนั้นการประมวลผลเกี่ยวกับภาพต้องใช้หน่วยความจำมาก ทำให้ต้องทำอยู่บนเครื่องคอมพิวเตอร์ขนาดมินิขึ้นไป จึงเป็นเหตุให้การพัฒนาทางด้านนี้ของประเทศเรามีโอกาสน้อยมากและไปได้ช้า แต่ปัจจุบันเครื่องคอมพิวเตอร์ขนาดไมโครมีหน่วยความจำมากพอที่จะใช้ในการประมวลผลของภาพ จึงทำให้เกิดการพัฒนาในส่วนของการมองของหุ่นยนต์ ดังเสนอในวิทยานิพนธ์ฉบับนี้

สำหรับการจดจำวัตถุต่างๆหลังการมองเห็นของมนุษย์คือ จดจำรูปพรรณ สี ความมันของผิวและอื่นๆ ซึ่งถ้าหากเราพิจารณาให้ลึกซึ้งแล้ว ข้อมูลตัวสำคัญที่จะเป็นประโยชน์มาก และเราน่าทำการจดจำเป็นอันดับแรกคือ สัณฐานรูปพรรณ ในการจดจำรูปพรรณของวัตถุในตัวมนุษย์นั้นจะใช้วิธีการจดจำแบบทั้งภาพ(ทุกๆจุดภาพของวัตถุ) ซึ่งถ้าหากเราให้คอมพิวเตอร์จดจำรูปพรรณของวัตถุในลักษณะแบบทั้งภาพนี้ ก็จะสิ้นเปลืองหน่วยความจำมาก วิธีการดังกล่าวนี้เป็นวิธีที่ไม่เหมาะสม แนวทางหนึ่งที่ช่วยลดปัญหาส่วนนี้ลงได้คือ การหาขอบของภาพวัตถุ แล้วนำขอบของภาพวัตถุที่ได้มาเข้ารหัสให้เป็นรูปแบบใดรูปแบบหนึ่งซึ่งในวิทยานิพนธ์ฉบับนี้ ขอบภาพวัตถุจะถูกนำมาเข้ารหัสของ Freeman อย่างไรก็ดีรหัสของ Freeman ก็ยังไม่ใช่วิถีทางแก้ไขที่ดีที่สุด ต่อมาจึงได้พัฒนาให้ขอบของวัตถุอยู่ในรูปรหัสความยาวและมุม ในการตรวจสอบชนิดของวัตถุด้วยรหัสของความยาวและมุมนี้จะช่วยลดเวลาในการเปรียบเทียบหาชนิดของวัตถุได้เป็นอย่างมาก อีกทั้งยังช่วยลดหน่วยความจำของเครื่องไมโครคอมพิวเตอร์ลงด้วย ผลการทดลองและรายละเอียดของวิธีการต่างๆเหล่านี้ได้แสดงไว้ในบทถัดๆไปของวิทยานิพนธ์ฉบับนี้แล้ว

สำหรับอุปกรณ์ที่ใช้ในวิทยานิพนธ์ฉบับนี้จะประกอบด้วย กล้องถ่ายภาพ ดิจิไตเซอร์ และเครื่องไมโครคอมพิวเตอร์ IBM PC/XT อย่างละชุด อุปกรณ์ดังกล่าวนี้เป็นอุปกรณ์เบื้องต้นที่จำเป็นสำหรับงานการจดจำรูปแบบของวัตถุ

ในบทที่ 2 จะกล่าวถึงการรับข้อมูลภาพจากกล้องถ่ายภาพ และนำข้อมูลภาพมาตัดให้เป็นภาพสองระดับ หรือที่เข้าใจกันใน ภาพขาวดำ จากนั้นจึงนำภาพสองระดับมาทำการหาขอบเพื่อเข้ารหัสลูกโซ่ของ Freeman สำหรับใช้ในบทต่อๆไป

ในบทที่ 3 เป็นการหมุนวัตถุที่กำลังตรวจสอบให้มีแนวแกนหลักของวัตถุอยู่ในแนวแกนเดียวกันกับแนวแกนหลักของวัตถุอ้างอิงก่อน จากนั้นจึงนำขอบของวัตถุที่หมุนแล้วนี้มาเข้ารหัสลูกโซ่ของ Freeman และทำสหสัมพันธ์ (Correlation) เพื่อการตรวจสอบ

ในบทที่ 4 จะกล่าวถึงวิธีการนำเอารหัสลูกโซ่ของ Freeman แบบ 8-ทิศทาง มาหาจุดเปลี่ยนแนว (Break point) และนำจุดเปลี่ยนแนวที่หาได้นี้มาทำให้อยู่ในรูปรหัสลูกโซ่ของ มุมและระยะทาง จากนั้นนำรหัสลูกโซ่ใหม่นี้มาทำสหสัมพันธ์ ซึ่งได้แสดงผลไว้ท้ายของบทนี้

ในบทที่ 5 เป็นการตรวจหาภาพวัตถุที่ถูกบดบังโดยใช้เทคนิคของบทที่ 4 มาปรับให้เหมาะสมกับกรณีดังกล่าว และใช้เรขาคณิตวิเคราะห์ (Geometric Correction) เพื่อหาตำแหน่งของวัตถุที่ถูกบดบังไป

ในบทที่ 6 เป็นบทสรุปรวมของผลงานวิจัยสำหรับทุกๆบท

ภาคผนวก ก. เป็นรายละเอียดของโปรแกรม

ขอบเขตของวิทยานิพนธ์

เนื่องจากการถ่ายภาพของวัตถุชนิดเดียวกัน จะให้ขนาดของวัตถุไม่เท่ากันถ้าระยะทางจากกล้องถึงวัตถุเปลี่ยนไป ดังนั้นในวิทยานิพนธ์นี้จึงได้กำหนดขอบเขตของการทำงานเอาไว้ กล่าวคือการจำแนกชนิดของวัตถุนั้นสามารถทำได้ก็ต่อเมื่อระยะทางของกล้องกับวัตถุทุกชิ้นอยู่ห่างเท่าๆกัน

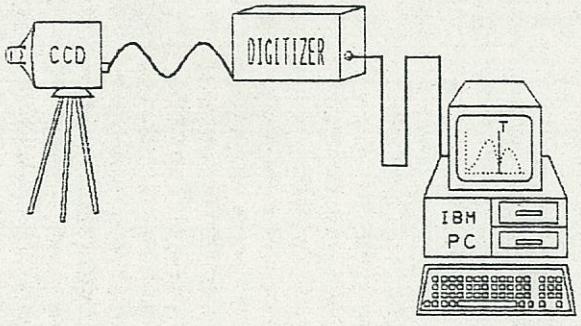
บทที่ 2

การหาขอบของภาพวัตถุ

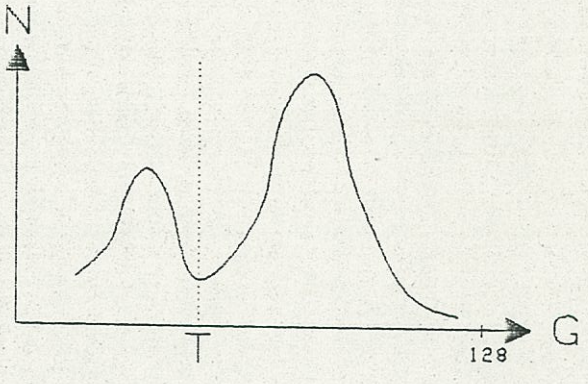
ในการจดจำวัตถุนั้น สิ่งที่จะให้ข้อมูลเกี่ยวกับรูปพรรณของวัตถุได้มากที่สุด โดยไม่จำเป็นที่จะต้องจดจำทั้งภาพคือ ส่วนขอบของวัตถุ ซึ่งในบทนี้จะได้กล่าวถึงขบวนการเริ่มต้นจากรับข้อมูลจากกล้องถ่าย จนกระทั่งถึงการนำข้อมูลของขอบวัตถุมาเก็บเข้าหน่วยจำ ของเครื่องคอมพิวเตอร์

2.1 การตัดค่าความเข้มของข้อมูลภาพให้เป็นภาพ 2 ระดับ (Binary Image)

ภาพวัตถุที่ถ่ายได้จากกล้องนั้นจะมีข้อมูลเป็นค่าระดับความเข้มของสีเทา ซึ่งอยู่ในรูปของอนาล็อก ข้อมูลภาพวัตถุที่จุดนี้จะนำมาผ่านเครื่องดิจิไตเซอร์เพื่อการแปลงขนาดสัญญาณความเข้ม (อนาล็อก) ให้เป็นสัญญาณดิจิตอล สำหรับป้อนเข้าสู่เครื่องไมโครคอมพิวเตอร์เพื่อการประมวลผล ดังรูปที่ 2.1 การประมวลผลในขั้นตอนแรกจะทำการเปลี่ยนระดับความเข้มสีเทาของข้อมูลภาพจาก 128 ระดับ(ดิจิไตเซอร์ที่ใช้ในงานวิทยานิพนธ์) ให้เป็นระดับความเข้มสีเทาเพียง 2 ระดับ เท่านั้น โดยนำภาพข้อมูลที่ได้จากดิจิไตเซอร์มาสร้างกราฟฮิสโตแกรม กราฟดังกล่าวนี้เป็นการแสดงความสัมพันธ์ระหว่างค่าความเข้มสีเทา(G) กับ จำนวนจุดภาพ(N) เพื่อใช้ในการดูกลุ่มข้อมูล ซึ่งโดยปกติแล้วกลุ่มข้อมูลความเข้มของวัตถุ กับ กลุ่มข้อมูลความเข้มของฉาก จะมีค่าที่แตกต่างกันดังแสดงในรูปที่ 2.2 ฮิสโตแกรมในรูปนี้สามารถสร้างเป็นภาพสองระดับ (Binary image) ได้ โดยการตั้งค่าเธร์ดโฮลด์ (Threshold) ไว้ที่ระดับความเข้มสีเทา T ค่าเธร์ดโฮลด์ T นี้จะแบ่งค่าความเข้มสีเทาของฉาก กับ วัตถุ ออกจากกันเป็น 2 ระดับ โดยจะให้ฉากมีระดับสีเทาเป็น 0 และระดับสีเทาของวัตถุเป็น 1 ในการตัดแบ่งภาพให้เป็นสองระดับนี้ทำได้ทั้งแบบด้วยมือ (manual) หรือแบบอัตโนมัติดังวิธีการของ [1]



รูปที่ 2.1 แสดงขั้นตอนการรับข้อมูลภาพจากกล้องเข้า
เครื่องคอมพิวเตอร์



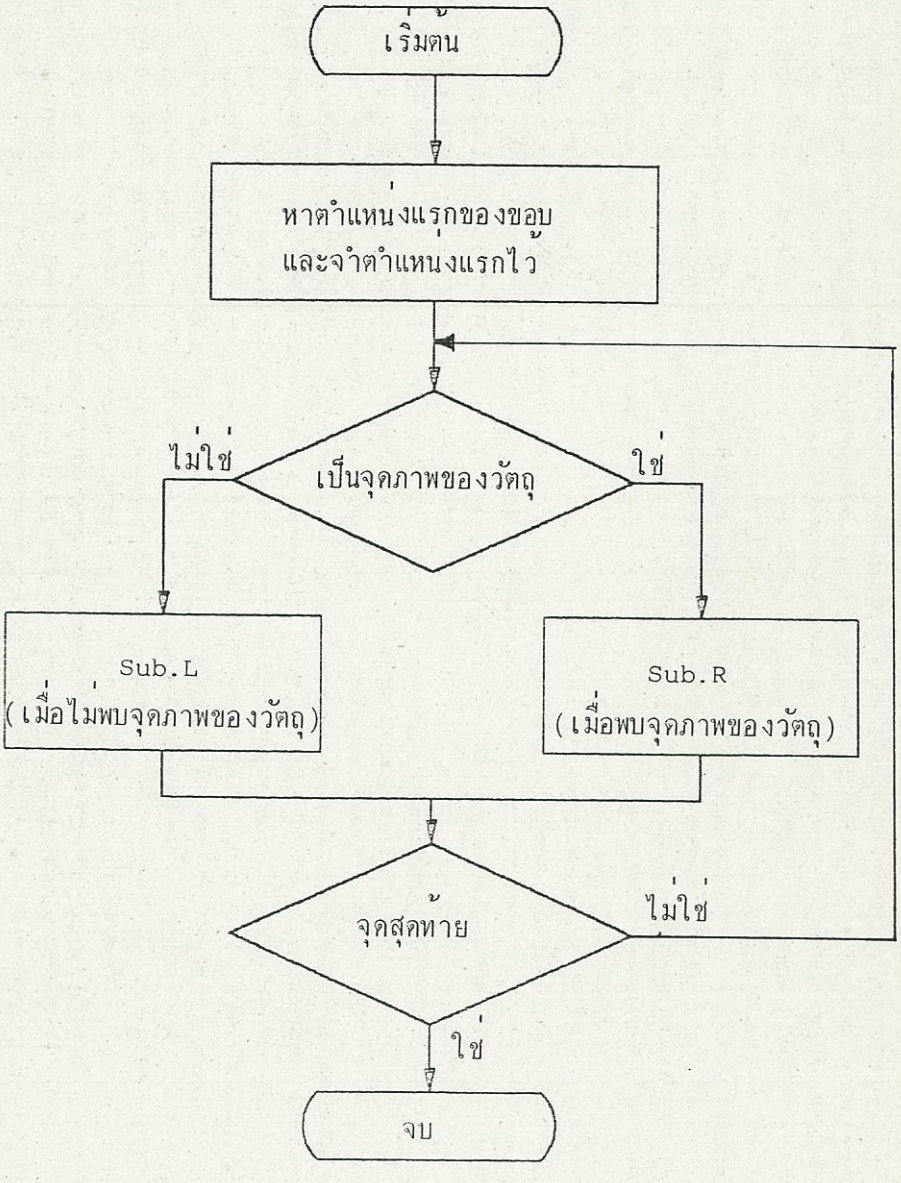
รูปที่ 2.2 กราฟฮิสโตแกรม แสดงความสัมพันธ์ ระหว่าง
ค่าความเข้มสีเทา(G) กับ จำนวนจุดภาพ(N)

2.2 การติดตามขอบของวัตถุ (Contour following)

การติดตามขอบของวัตถุจากข้อมูลภาพความเข้ม 2 ระดับนั้นมีอยู่หลายวิธี แต่จะขอยกมากล่าวเพียง 2 วิธี ดังต่อไปนี้

2.2.1 วิธีการตรวจจับขอบวัตถุของหุ่นยนต์

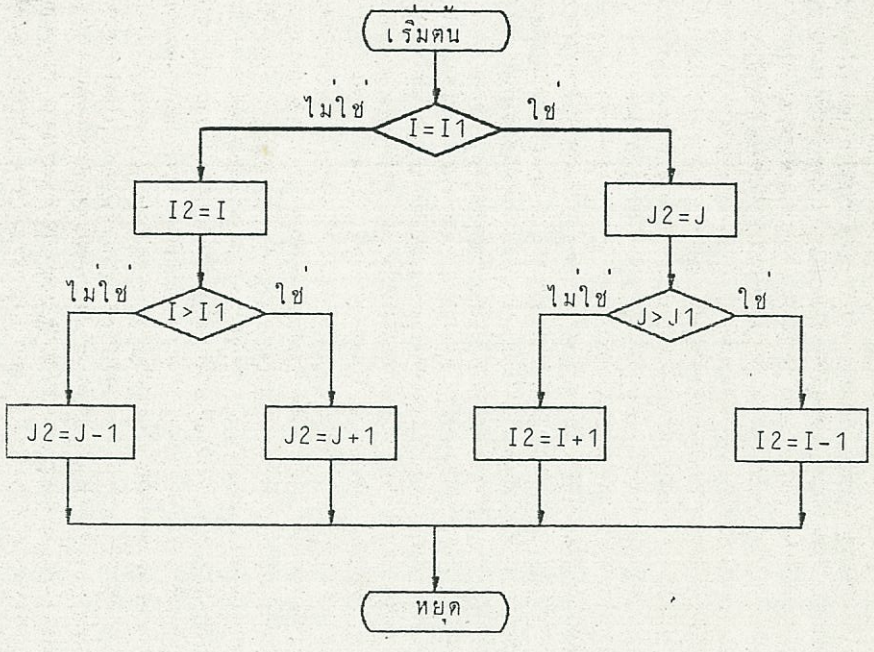
จากเอกสารอ้างอิง[2] นั้นการตรวจหาจุดขอบของวัตถุทำได้โดยอาศัยกฎเกณฑ์ พอสรุปขั้นตอนการทำได้ตาม ผังที่2.1 ข้างล่างนี้



ผังที่ 2.1 แสดงขั้นตอนการตรวจจับขอบวัตถุ

จากผังการหาขอบข้างต้นนี้สามารถอธิบายรายละเอียดได้ดังนี้

เราจะเริ่มตรวจจับขอบวัตถุทางด้านขวาตอนบนของภาพ กล่าวคือ เราจะ scan จากขวาไปซ้าย (ตัวอย่างนี้เป็นเพียงการตรวจจับขอบเส้นนอกทางด้านขวาของวัตถุเท่านั้น ปกติแล้ววัตถุจะมีรูปทรงอะไรก็ได้ โดยที่ขอบของวัตถุอาจจะวนครบรอบก็สามารถใช้วิธีการนี้ตรวจจับได้) การ scan นี้ถ้าไปเจอจุดที่เป็นจุดภาพของวัตถุให้เรียกใช้ Subroutine RIGHT แต่ถ้าเป็นจุดของ Back ground ก็ให้เรียก Subroutine LEFT (ในทางตรงข้ามถ้า scan จากซ้ายไปขวา เมื่อเจอจุดที่เป็นจุดภาพ ของวัตถุให้เรียก Subroutine LEFT และถ้าเจอ Back ground ก็ให้เรียก Subroutine RIGHT) Subroutine RIGHT และ LEFT แสดงตามผังที่ 2.2 และ ผังที่ 2.3 ตามลำดับ

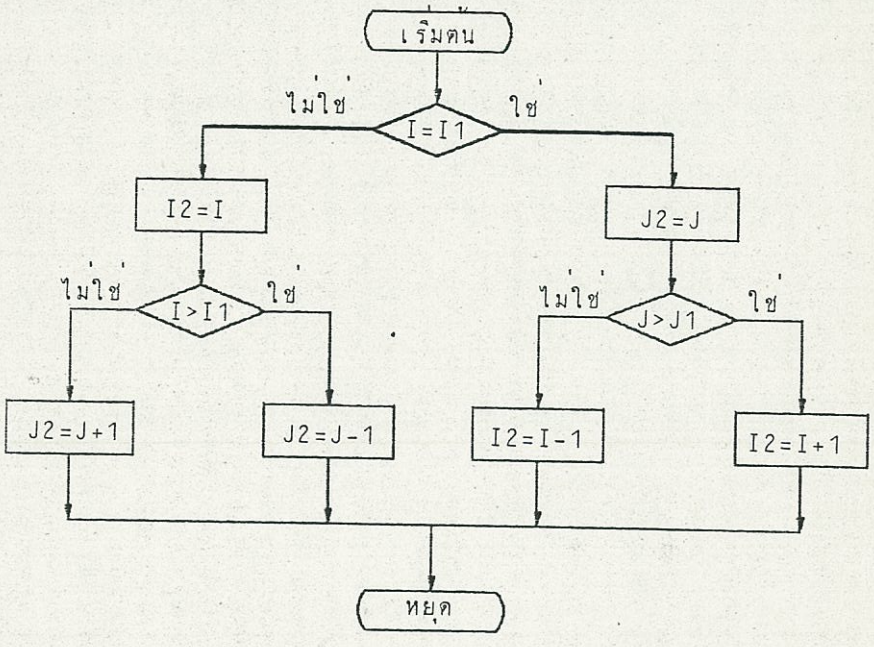


ผังที่ 2.2 แสดง Subroutine RIGHT (I,J,I1,J1,I2,J2)

โดยที่ (I1,J1) เป็น coordinate ของจุดภาพที่ผ่านมา

(I,J) เป็น coordinate ของจุดภาพปัจจุบัน

(I2,J2) เป็น coordinate ของจุดภาพอนาคต



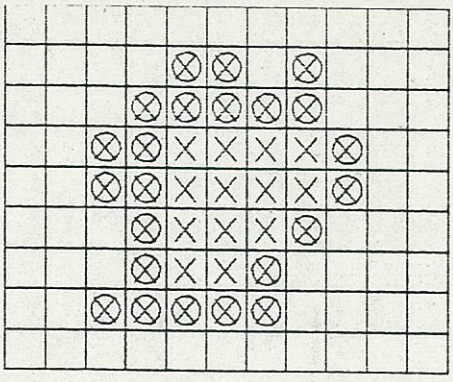
ผังที่ 2.3 แสดง Subroutine LEFT (I,J,I1,J1,I2,J2)

โดยที่ (I1,J1) เป็น coordinate ของจุดภาพที่ผ่านมา

(I,J) เป็น coordinate ของจุดภาพปัจจุบัน

(I2,J2) เป็น coordinate ของจุดภาพอนาคต

จากหลักการที่กล่าวในข้างต้นนี้ได้ทำการทดสอบหาขอบ กับ ข้อมูลภาพความเข้ม 2 ระดับจะได้ผลดังที่แสดงในรูป 2.3



รูปที่ 2.3 แสดงขอบที่หาได้จากวิธีการตรวจจับขอบวัตถุจากเอกสารอ้างอิง [2]

2.2.2 วิธีการตรวจหาขอบของวัตถุด้วยการใช้ตารางหน้าต่าง

จากวิธีการตรวจจับขอบวัตถุของหุ่นยนต์ ยังได้ขอบของภาพที่หนาเกินความจำเป็น ดังส่วนที่เป็นวงกลมในรูปที่ 2.3 ความหนาของขอบวัตถุนี้สามารถแก้ไขได้ด้วย การใช้ตารางหน้าต่าง[3] (Window) ซึ่งจะให้ขอบของวัตถุบางที่สุด และเมื่อนำไปเข้ารหัสของ Freeman แล้วจะช่วยลดเวลาในการทำสหสัมพันธ์ (Correlation) และสามารถลดหน่วยความจำสำหรับใช้เก็บข้อมูลของรหัสลูกโซ่ (Chain code) ของขอบวัตถุ

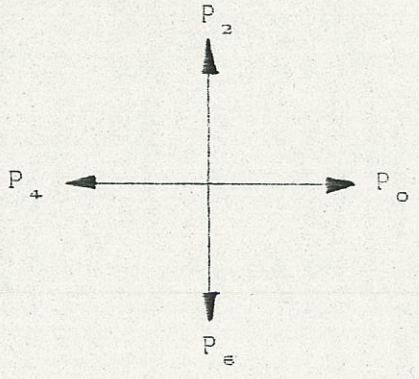
2.2.2.1 ตารางหน้าต่าง (Window) คือ ตารางขนาด 3x3 ดังรูปที่ 2.4 ซึ่งเราจะวางตารางหน้าต่างนี้ทาบลงบนจุดภาพที่เป็นจุดขอบของวัตถุ เพื่อหาจุดขอบของวัตถุจุดถัดไป โดยให้จุด P_x อยู่ที่ตำแหน่งของจุดขอบของวัตถุ

P_3	P_2	P_1
P_4	P_x	P_0
P_5	P_6	P_7

รูปที่ 2.4 แสดงตำแหน่งของตารางหน้าต่าง

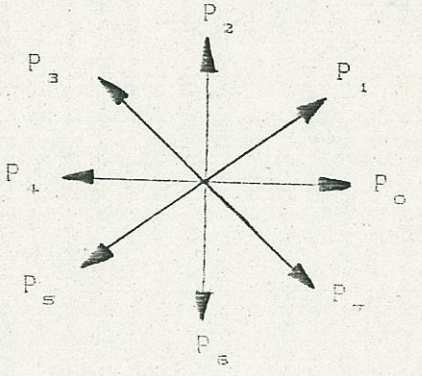
2.2.2.2 Connectivity คือ ความต่อเนื่องของจุดภาพแต่ละจุดบนระนาบ X-Y ซึ่งพิจารณาได้ 2 แบบ คือ

แบบที่ 1 4-Connectivity เราจะพิจารณาในรูปทิศทางได้ 4 ทิศทาง ซึ่งดูจากตารางหน้าต่าง(รูปที่ 2.4) กล่าวคือ 4-Connectivity ของ P_x จะเป็น P_0 , P_2 , P_4 หรือ P_6 ดังแสดงในรูปที่ 2.5



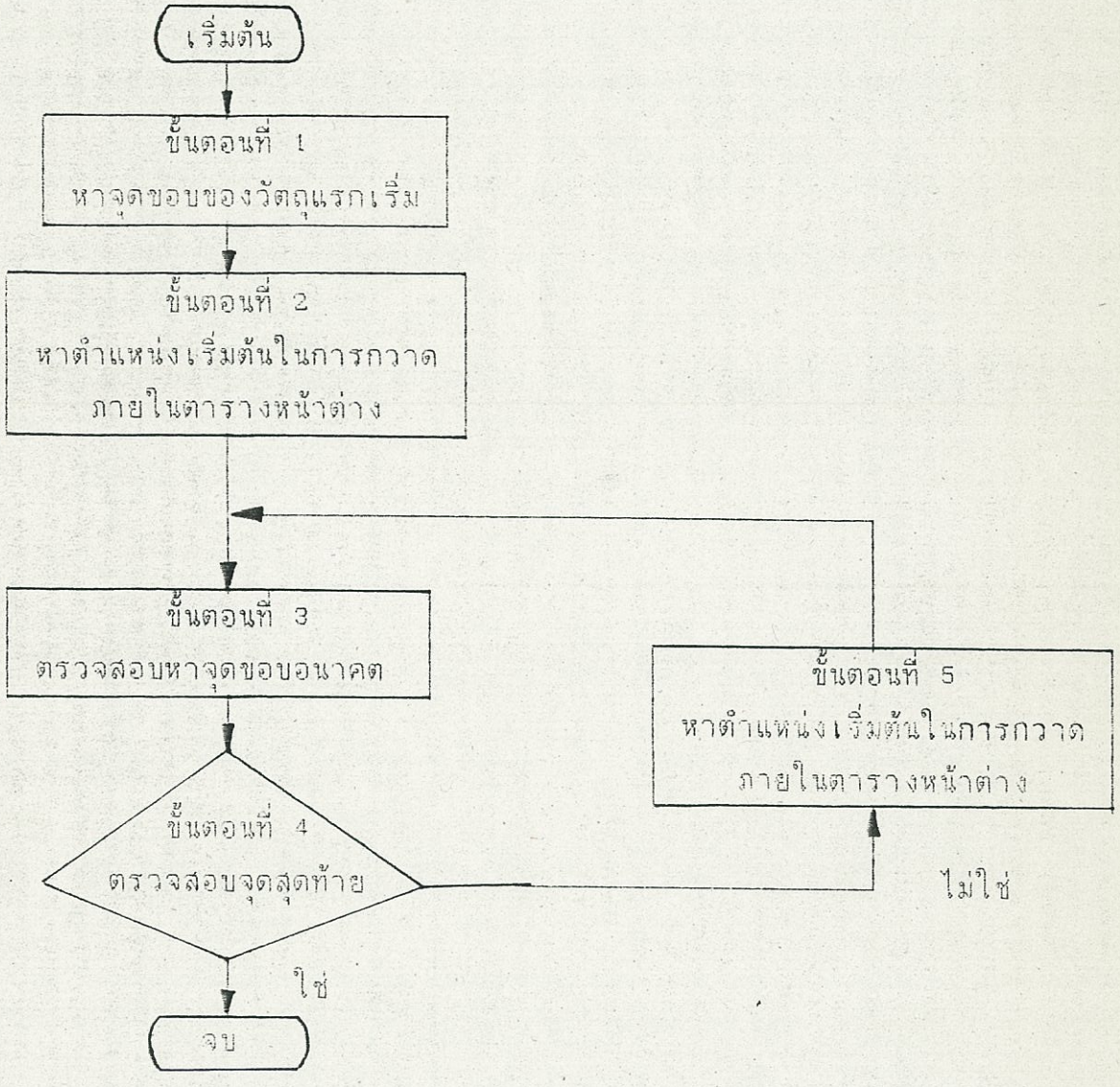
รูปที่ 2.5 P_x จะเปลี่ยนทิศทางจากจุดศูนย์กลางไปยัง P_0 , P_2 , P_4 หรือ P_6 ตำแหน่งใดตำแหน่งหนึ่ง

แบบที่ 2 8-Connectivity เราจะพิจารณาในรูปทิศทางได้ 8 ทิศทาง ซึ่งดูจากตารางหน้าต่าง(รูปที่ 2.4)กล่าวคือ 8-Connectivity ของ P_x จะเป็น P_0 , P_1 , P_2 , P_3 , P_4 , P_5 , P_6 หรือ P_7 ดังแสดงในรูปที่ 2.6



รูปที่ 2.6 P_x จะเปลี่ยนทิศทางจากจุดศูนย์กลางไปยัง P_0 , P_1 , ... , P_6 หรือ P_7 ตำแหน่งใดตำแหน่งหนึ่ง

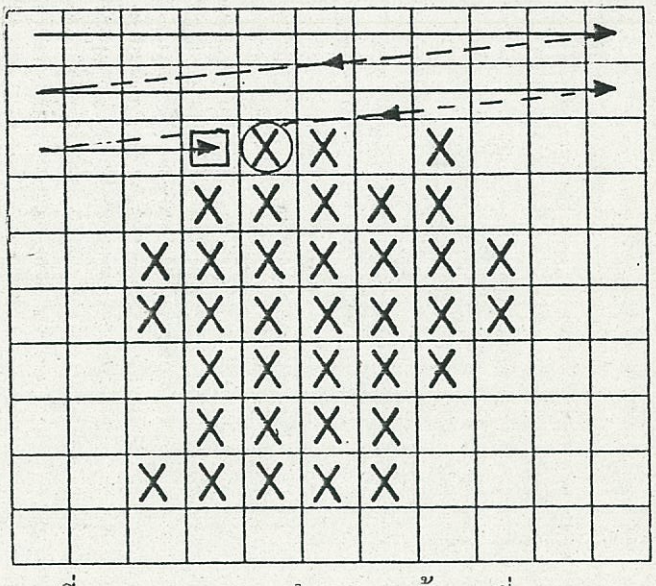
วิธีการตรวจสอบหาขอบของวัตถุโดยใช้ตารางหน้าต่างสามารถอธิบายและเขียนเป็นขั้นตอนตาม ผังที่ 2.4



ผังที่ 2.4 แสดงขั้นตอนการหาขอบของวัตถุด้วยใช้ตารางต่าง

ขั้นตอนที่ 1 จะต้องหาจุดขอบของภาพวัตถุที่ต้องการจะพิจารณาเสียก่อน ซึ่งในวิทยานิพนธ์ฉบับนี้ได้ใช้วิธีการหาโดย กวาดจุดภาพจากซ้ายไปขวาและบนลงล่าง เมื่อได้จุดขอบของภาพแล้วจะต้องเก็บตำแหน่งของจุดภาพนี้ไว้ เพื่อตรวจสอบว่าเป็นจุดขอบสุดท้าย

ขั้นตอนที่ 2 หาตำแหน่งเริ่มต้นในการกวาดจุดภาพภายในตารางหน้าต่างที่เหมาะสม ซึ่งจำเป็นจะต้องนำเอาทิศทางการกวาดหาจุดขอบของภาพวัตถุในขั้นตอนที่ 1 มาใช้ในการพิจารณาหาตำแหน่งเริ่มต้นนี้ด้วยกล่าวคือ จะใช้จุดภาพก่อนจุดขอบของวัตถุแรก (ในขั้นตอนที่ 1) 1 ตำแหน่งดังรูปที่ 2.7 เป็นจุดเริ่มของการกวาดภายในตารางหน้าต่าง ซึ่งในรูปที่ 2.7 ตำแหน่งเริ่มต้นในการกวาดจุดภาพภายในตารางหน้าต่างที่เหมาะสมนั้น คือ ตำแหน่ง P_4 อันเป็นรูป □ ในรูปที่ 2.7 โดยที่ ⊗ เป็นตำแหน่งของ P_x



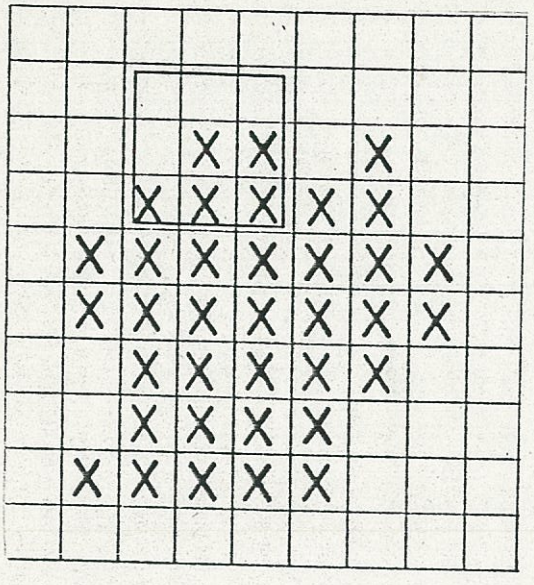
รูปที่ 2.7 แสดงการทำงานของขั้นตอนที่ 1 และ 2
โดยที่ ⊗ หมายถึง จุดขอบของวัตถุเริ่มต้น
□ หมายถึง จุดภาพก่อนจุดขอบของวัตถุเริ่มต้น

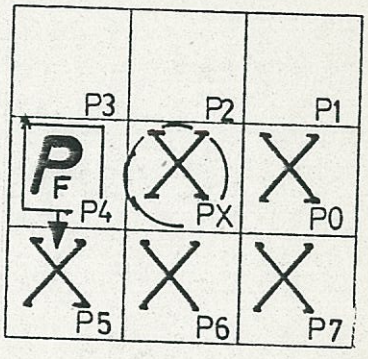
ตารางที่ 2.1 เป็นตารางการหาตำแหน่งเริ่มต้น ในการกวาดจุดภาพภายในตารางหน้าต่างที่เหมาะสม โดยพิจารณาจากการหาจุดขอบจุดแรกเริ่มของขั้นตอนที่ 1

ตารางที่ 2.1 กำหนดตำแหน่งเริ่มต้นตรวจสอบรอบตารางหน้าต่าง (ขั้นตอนที่ 2)

ทิศทางการกวาดหาจุดขอบจุดแรกเริ่ม	ตำแหน่งเริ่มต้นในการกวาดจุดภาพภายในตารางหน้าต่างที่เหมาะสม
ซ้าย —> ขวา ; บน <—> ล่าง	P_4
ล่าง —> บน ; ซ้าย <—> ขวา	P_6
บน —> ล่าง ; ซ้าย <—> ขวา	P_2
ขวา —> ซ้าย ; บน <—> ล่าง	P_0
อื่น ๆ	P_F

ขั้นตอนที่ 3 ตรวจหาจุดขอบ(Contour)อนาคต ($P_0 - P_7$) โดยเริ่มตรวจสอบจากตำแหน่ง P_F ที่หาได้จากขั้นตอนที่ 2 หรือ 5 ทำการกวาดรอบตารางหน้าต่างไปทีละหนึ่งตำแหน่งในทิศทางทวนเข็มหรือตามเข็มเพียงทิศทางใดทิศทางหนึ่งเท่านั้น และจุดแรกที่ตรวจพบว่าเป็นจุดของภาพของวัตถุ ให้ถือว่าจุดนั้นเป็นจุดขอบอนาคต เราจะใช้จุดขอบอนาคตจุดนี้เป็นจุดขอบของวัตถุ ที่ใช้หาจุดขอบอนาคตถัดไป ซึ่งจาก รูปที่ 2.8 จะเห็นว่าทิศทางการตรวจสอบจะเริ่มที่ P_4 , P_5 , P_6 , P_7 ,P_0 , P_1 , P_2 และ P_3 ตามลำดับในทิศทางทวนเข็มและได้ P_5 เป็นจุดแรกที่ตรวจพบว่า เป็นจุดของภาพวัตถุก่อน ดังนั้น P_5 จึงเป็นจุดขอบอนาคต





$P_F = P_4$ = ตำแหน่งเริ่มต้นของการตรวจสอบ

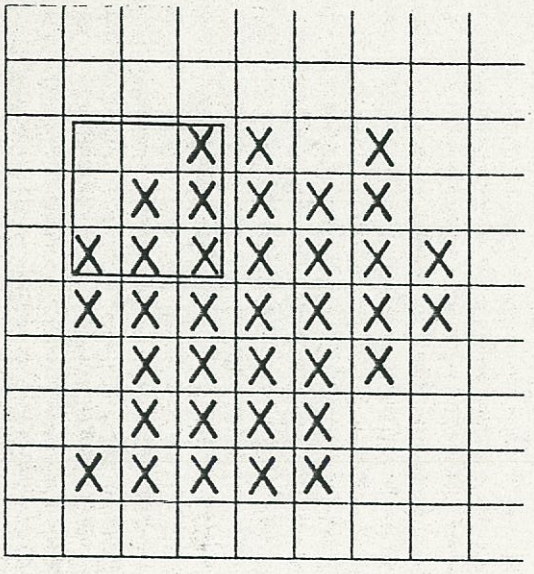
รูปที่ 2.8 แสดงตำแหน่งเริ่มต้นการกวาดรอบตารางหน้าต่าง ซึ่งหาได้จากการทำงานในขั้นตอนที่ 2 หรือ 5

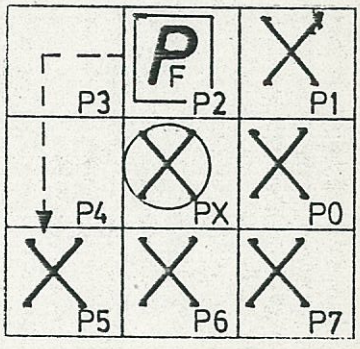
ขั้นตอนที่ 4 ตรวจสอบตำแหน่งของจุดขอบอนาคต ที่ได้จากขั้นตอนที่ 3 นั้นเป็นตำแหน่งสุดท้ายหรือไม่ ถ้าไม่ใช่ก็ให้ไปทำงานขั้นตอนที่ 5 ถ้าหากเป็นตำแหน่งสุดท้ายก็ให้หยุดการทำงาน

ขั้นตอนที่ 5 หาตำแหน่งเริ่มต้นในการกวาดรอบตารางหน้าต่าง ซึ่งการหาตำแหน่งเริ่มต้นในขั้นตอนนี้จะต่างจากในขั้นตอนที่ 2 โดยขั้นตอนนี้จะใช้จุดขอบวัตถุอดีต หรือตำแหน่งของจุดขอบวัตถุปัจจุบัน ในขั้นตอนที่ 3 มาพิจารณาหาตำแหน่งเริ่มต้นในการกวาดรอบตารางหน้าต่าง

$$P_F = P_{\text{Contour อดีต} + 1}$$

ดังตัวอย่างในรูปที่ 2.9





$P_F = P_z$ = ตำแหน่งเริ่มต้นของการตรวจสอบ

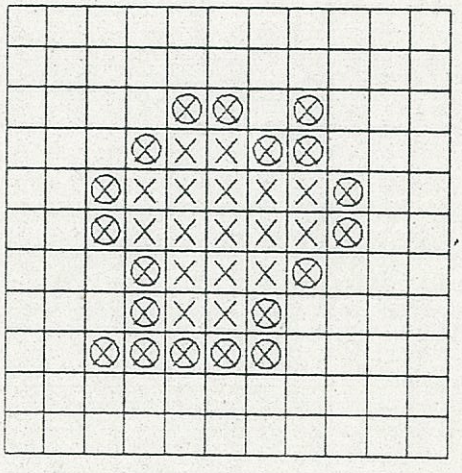
รูปที่ 2.9 แสดงตำแหน่งเริ่มต้นที่หาได้จากสูตร $P_F = P_{\text{Contour อดีต + 1}}$ หรือ เปิดจากตารางที่ 2.2

ตารางที่ 2.2 เป็นตารางหาตำแหน่งเริ่มต้นการกวาดจุดภาพภายในตารางหน้าต่างที่เหมาะสมโดยพิจารณาจากจุดขอบปัจจุบัน ในขั้นตอนที่ 3 และตารางนี้ใช้ได้เฉพาะ การกวาดภายในตารางหน้าต่าง ที่มีทิศทางทวนเข็มเท่านั้น

ตารางที่ 2.2 กำหนดตำแหน่งเริ่มต้นตรวจสอบรอบตารางหน้าต่าง (ขั้นตอนที่ 5)

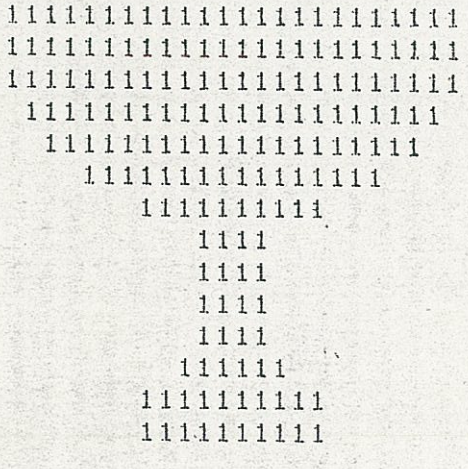
จุดขอบอนาคต ที่ได้ในขั้นตอนที่ 3	ค่าของตำแหน่งเริ่มต้นการกวาดจุดภาพภายในตารางหน้าต่างสำหรับครั้งต่อไป
P_0	P_5
P_1	P_6
P_2	P_7
P_3	P_0
P_4	P_1
P_5	P_2
P_6	P_3
P_7	P_4

จากการทดสอบเทคนิคที่คิดขึ้นมานี้กับ ภาพวัตถุที่ได้จำลองขึ้นมาดังรูปที่ 2.3 จะได้ผลดังแสดงในรูปที่ 2.10 จะพบว่าการใช้เทคนิค Window นี้สามารถติดตามขอบของวัตถุได้บางที่สุด และจะอยู่ในรูปของ 8-Connectivity

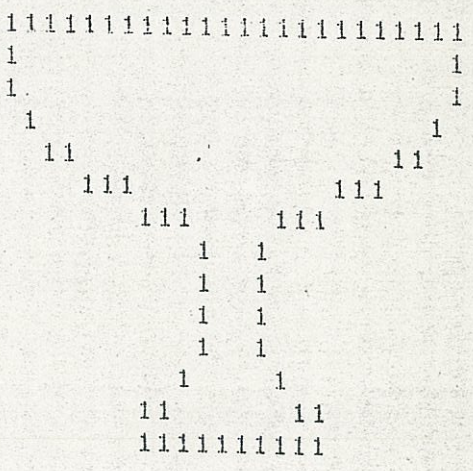


รูปที่ 2.10 แสดงขอบที่หาได้ด้วยวิธีการใช้ Window

การใช้เทคนิค Window นี้ยังสามารถนำเอาจุดขอบอนาคต มาเข้ารหัสลูกโซ่ได้ทันที จึงเป็นการลดขบวนการการเข้ารหัสลูกโซ่ ดังผลที่แสดงในรูปที่ 2.11 ก,ข,ค



รูปที่ 2.11 ก. แสดงภาพที่จำลองขึ้น



รูปที่ 2.11 ข. แสดงผลการหาขอบของวัตถุ โดยใช้ตารางหน้าต่าง

Format Data Edge

X , Y : Chain Code

6, 3:6	6, 4:6	6, 5:7	7, 6:7	8, 7:0	9, 7:7	10, 8:0	11, 8:0
12, 8:7	13, 9:0	14, 9:0	15, 9:7	16,10:6	16,11:6	16,12:6	16,13:5
15,14:5	14,15:4	13,15:6	13,16:0	14,16:0	15,16:0	16,16:0	17,16:0
18,16:0	19,16:0	20,16:0	21,16:0	22,16:2	22,15:4	21,15:3	20,14:3
19,13:2	19,12:2	19,11:2	19,10:1	20, 9:0	21, 9:0	22, 9:1	23, 8:0
24, 8:0	25, 8:1	26, 7:0	27, 7:1	28, 6:1	29, 5:2	29, 4:2	29, 3:4
28, 3:4	27, 3:4	26, 3:4	25, 3:4	24, 3:4	23, 3:4	22, 3:4	21, 3:4
20, 3:4	19, 3:4	18, 3:4	17, 3:4	16, 3:4	15, 3:4	14, 3:4	13, 3:4
12, 3:4	11, 3:4	10, 3:4	9, 3:4	8, 3:4	7, 3:4	6, 3:6	

รูปที่ 2.11 ค. แสดงตำแหน่ง X , Y และ ทิศทางของจุดขอบภาพถัดไป (หรือรหัสลูกโซ่ โดยอาศัยข้อมูลทิศทางจากรูป 2.6)

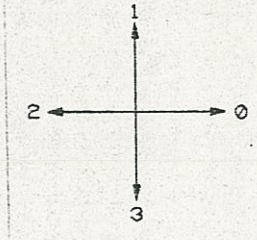
จากผลลัพธ์ของการหาขอบวัตถุโดยใช้เทคนิคตารางหน้าต่าง (หัวข้อ 2.2.2) จะให้รหัสข้อมูลของขอบน้อยกว่าวิธีการตรวจจับขอบวัตถุของหุ่นยนต์ (หัวข้อ 2.2.1) ดังนั้นงานวิทยานิพนธ์ฉบับนี้จึงได้เลือกใช้เทคนิคตารางหน้าต่างในการหาขอบวัตถุ เพื่อใช้กับเทคนิคในบทถัดไป

บทที่ 3

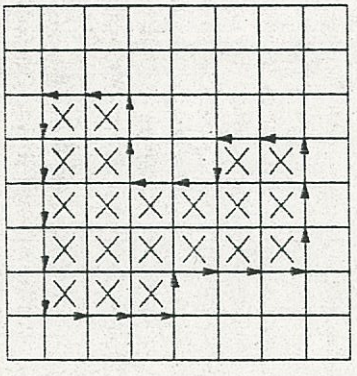
สหสัมพันธ์ด้วยรหัสลูกโซ่สำหรับการตรวจสอบวัตถุ

(Chain code correlation for object identification)

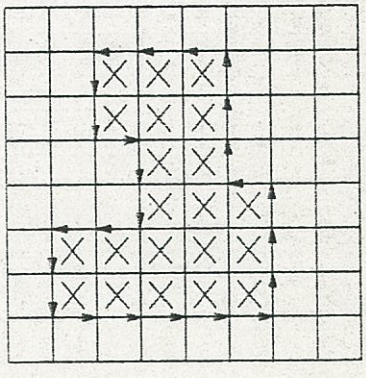
วิธีหนึ่งจากหลาย ๆ วิธีในการตรวจสอบภาพวัตถุทั้งสองเป็นวัตถุเดียวกันหรือไม่นั้น ทำได้โดยการตรวจสอบจากรหัสลูกโซ่ของภาพวัตถุทั้งสอง สำหรับรหัสลูกโซ่ที่นิยมใช้กันมีอยู่สองชนิดคือแบบ 4-ทิศทาง (4 connectivity) และ 8-ทิศทาง (8-connectivity) (สำหรับในบทนี้ได้ใช้วิธีการหาขอบภาพจากบทที่ 2 หัวข้อ 2.2.2 ซึ่งจะได้รหัสลูกโซ่เป็นแบบ 8-ทิศทาง) รหัสลูกโซ่นี้ได้จากการติดตามรอยขอบของวัตถุ ดังรูป 3.1 ซึ่งเป็นแบบ 4 ทิศทาง ส่วนในรูปที่ 3.2 นั้นรหัสที่ได้จะเป็นแบบ 8 ทิศทาง



รูปที่ 3.1 ก แสดงทิศทางของรหัสแบบ 4-ทิศทาง



รูปที่ 3.1 ข แสดงการเข้ารหัสที่ขอบวัตถุ

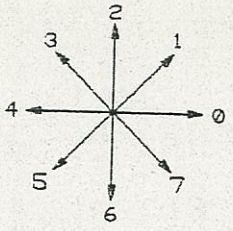


รูปที่ 3.1 ค แสดงการเข้ารหัสที่ขอบวัตถุ

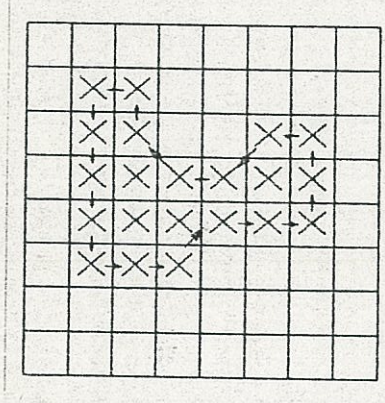
รหัสลูกโซ่ของรูป 3.1 ข 33333000100011122322112 2

รหัสลูกโซ่ของรูป 3.1 ค 33033223300000111211122 2

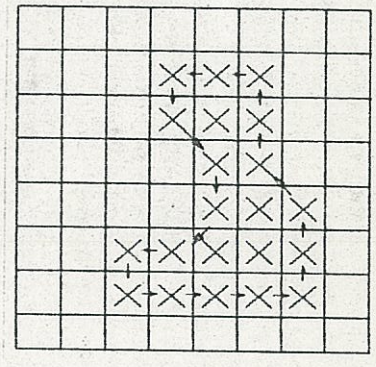
รูปที่ 3.1 ก ข ค รหัสลูกโซ่แบบ 4-ทิศทาง



รูปที่ 3.2 ก แสดงทิศทางของรหัสแบบ 8-ทิศทาง



รูปที่ 3.2 ข แสดงการเข้ารหัสที่ขอบวัตถุ



รูปที่ 3.2 ค แสดงการเข้ารหัสที่ขอบวัตถุ

รหัสลูกโซ่ของรูปที่ 3.2 ข 66660010022454324

รหัสลูกโซ่ของรูปที่ 3.2 ค 67654600002232244

รูปที่ 3.2 ก ข ค รหัสลูกโซ่แบบ 8-ทิศทาง

ซึ่งจากรูปที่ 3.1 ข และ 3.1 ค ภาพของวัตถุนั้นเป็นภาพวัตถุเดียวกัน แต่มีแนวแกนของวัตถุที่ต่างกันอยู่ 90 องศา ดังนั้นรหัสลูกโซ่ต่างๆ ของภาพวัตถุจึงมีค่าต่างกันอยู่ 1 ซึ่งเราสามารถปรับค่าชดเชยให้กับรหัสลูกโซ่ของภาพวัตถุ รูป 3.1 ก ได้ โดยบวกค่า 1 (90°) ให้กับรหัสลูกโซ่ทุกๆตัว

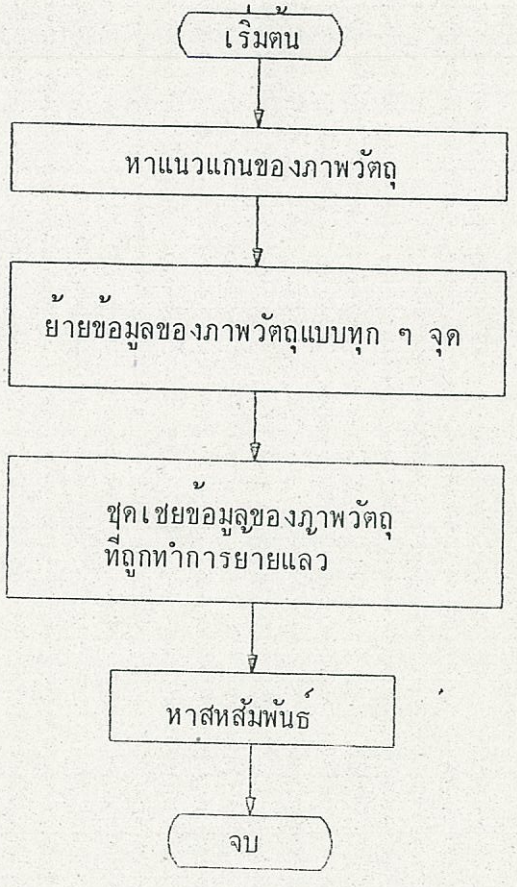
รหัสลูกโซ่เดิม + ค่าชดเชย = รหัสลูกโซ่ใหม่ แบบ Modulo-4

รหัสลูกโซ่เดิม	333330001000111223221122
ค่าชดเชย	11111.....111
รหัสลูกโซ่ใหม่	444441112111222334332233
รหัสลูกโซ่ใหม่แบบ Modulo 4	000001112111222330332233

ซึ่งจะได้รหัสเดียวกับรูป 3.1 ค

จากขบวนการที่กล่าวข้างต้นนี้เราจะได้รหัสลูกโซ่ใหม่ ซึ่งมีแนวแกนของภาพวัตถุเปลี่ยนไปจากเดิม 90 องศา ส่วนในกรณีของรหัสลูกโซ่แบบ 8-ทิศทาง นั้นก็สามารถใช้วิธีดังกล่าวได้ในทำนองเดียวกันกับแบบ 4-ทิศทาง

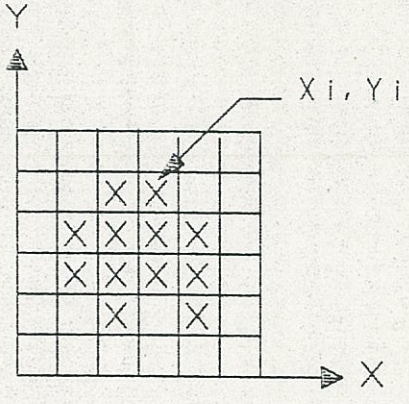
แต่ความเป็นจริงแล้วปัญหาที่เกิดขึ้นคือ แนวแกนของภาพวัตถุเปลี่ยนไปจากเดิมนั้นจะเป็นมุมใดๆก็ได้ เช่นเปลี่ยนไป 30 องศา ซึ่งทำให้ไม่สามารถที่จะบวกค่าชดเชยให้กับรหัสได้ทั้งนี้เนื่องจากมุมต่ำสุดที่ใช้ชดเชยคือ 45 องศา ที่ได้จากรหัสแบบ 8-ทิศทาง ทางหนึ่งที่จะแก้ไขปัญหานี้คือ จะต้องทำการหมุนวัตถุให้มีแนวแกนของภาพวัตถุอยู่ในแนวที่ต้องการ ซึ่งขั้นตอนและรายละเอียดการแก้ปัญหาจุดนี้จะเป็นไปตามผังที่ 3.1 ข้างล่างนี้



ผังที่ 3.1 ขั้นตอนการแก้ปัญหากรณีที่แนวแกนของภาพวัตถุอ้างอิง กับวัตถุที่กำลังตรวจสอบ มีแนวแกนที่แตกต่างกัน

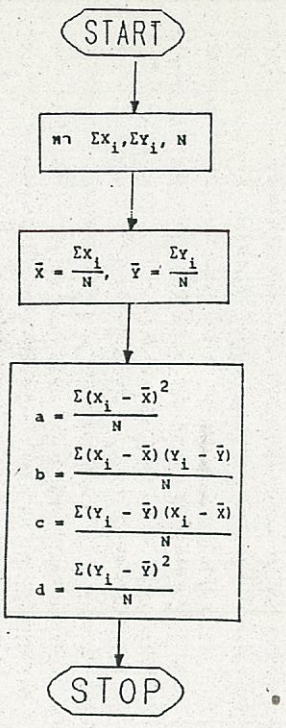
3.1 การหาแนวแกนหลักของภาพวัตถุ (principle axis)

ข้อมูลของภาพวัตถุที่นำมาประมวลผลเพื่อหาแนวแกนหลักของวัตถุนั้น จะเป็นดังรูป 3.3 ซึ่งอยู่บนระนาบ x, y การคำนวณหาแนวแกนหลักของภาพนั้นทำได้โดยการใช้เทคนิคการวิเคราะห์ข้อมูลทางสถิติเข้าช่วย เทคนิคดังกล่าวนี้เรียกกันว่า การวิเคราะห์องค์ประกอบหลัก (priciple component analysis) ซึ่งผู้สนใจสามารถหารายละเอียดเพิ่มเติมได้จากเอกสารอ้างอิง [4]



รูปที่ 3.3 แสดงตำแหน่ง Xi, Yi ของข้อมูลภาพที่จะนำมาประมวลผล

วิธีการวิเคราะห์องค์ประกอบหลัก ทำได้โดยการคำนวณหาอีลิเมนต์ในเมตริกซ์ของสมการ (3.1) ตามผังที่ 3.2 ข้างล่างนี้



ผังที่ 3.2 ขั้นตอนการคำนวณหาอีลิเมตริกซ์ของสมการ (3.1)

การวิเคราะห์องค์ประกอบหลักนี้จะให้เมตริกซ์ดังสมการ (3.1)

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad \ldots(3.1)$$

จากเมตริกซ์ที่ได้นำมาหาค่า Eigenvalue (λ) ซึ่งจะได้

$$\begin{bmatrix} a - \lambda & b \\ c & d - \lambda \end{bmatrix}$$

$$\det = (a - \lambda)(d - \lambda) - (bc) \qquad \ldots(3.2)$$

ทำการแก้สมการ (3.1) เพื่อหาค่า Eigenvalue (λ)

โดยที่ λ คือ ค่า Eigenvalue

นำค่า Eigenvalue ที่มีค่ามากที่สุดมาทำการหา Eigenvector

$$\begin{bmatrix} a - \lambda_{max} & b \\ c & d - \lambda_{max} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad ..(3.3)$$

โดยที่

λ_{max} = ค่า Eigenvalue สูงสุด

$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix}$ = ค่า Eigenvector [5] ที่เกิดจากค่า λ_{max}

นำค่า Eigenvector มาหามุมซึ่งจะเป็นแนวแกนหลักของข้อมูลภาพที่บิดไปเมื่อเทียบกับแกนของแนวนอน

$$\theta_E = \tan^{-1} (V_2/V_1) \quad ...(3.4)$$

3.2 การหมุนข้อมูลของภาพวัตถุ (Rotate)

สำหรับการหมุนข้อมูลภาพ [6] ทำได้โดยใช้สมการ (3.5)

$$\begin{bmatrix} \cos\theta_R & \sin\theta_R \\ -\sin\theta_R & \cos\theta_R \end{bmatrix} \begin{bmatrix} X_1 \\ Y_1 \end{bmatrix} = \begin{bmatrix} X'_1 \\ Y'_1 \end{bmatrix} \quad ...(3.5)$$

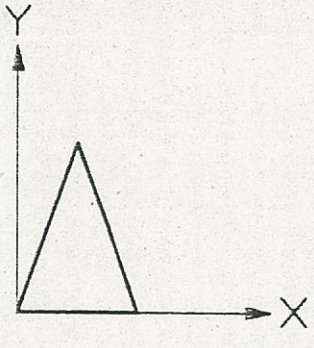
เมื่อ θ_R = มุมที่ต้องการจะให้ข้อมูลภาพถูกหมุน

X_i , Y_i = ตำแหน่งเดิมข้อมูลภาพ

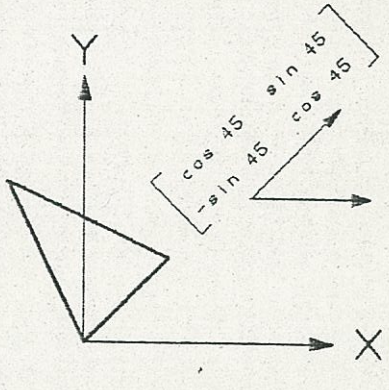
X'_i , Y'_i = ตำแหน่งใหม่ของข้อมูลภาพ

การหมุนข้อมูลภาพด้วยสมการ (3.5) นั้นจะเป็นการหมุนไปรอบจุด Coordinate (0,0) ดังรูป 3.4 แต่การหมุนข้อมูลภาพนั้นจำเป็นจะต้องหมุนอยู่ที่จุดศูนย์กลางของวัตถุ ซึ่งจุดศูนย์กลางของวัตถุนี้คือตำแหน่ง $\bar{X}, \bar{Y}$ ที่หาได้จากหัวข้อ 3.1 ดังนั้นจะต้องเลื่อนข้อมูลภาพมาให้จุดศูนย์กลางนี้ของวัตถุมาอยู่ตรง Coordinate (0,0) ด้วยสมการ (3.6)

$$\begin{bmatrix} X_i \\ Y_i \end{bmatrix} - \begin{bmatrix} \bar{X} \\ \bar{Y} \end{bmatrix} = \begin{bmatrix} X'_i \\ Y'_i \end{bmatrix} \quad \ldots(3.6)$$

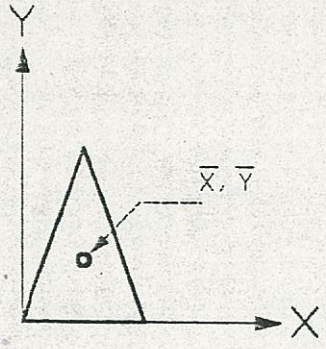


รูป 3.4 ก ข้อมูลภาพปกติ

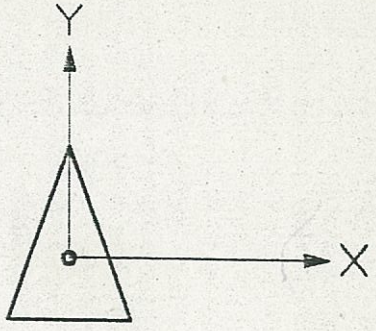


รูป 3.4 ข ข้อมูลภาพที่ถูก Rotate ด้วยสมการ (3.4) มุม 45°

จากสมการ(3.6) จะเป็นการเลื่อนจุดศูนย์ถ่วงของข้อมูลภาพมาอยู่ตำแหน่ง Coordinate (0,0) ดังรูป 3.5



รูป 3.5 ก ข้อมูลภาพปกติ



รูป 3.5 ข ข้อมูลภาพถูกเลื่อนด้วย สมการ 3.6

และในกรณีที่ต้องการจะเลื่อนข้อมูลภาพให้กลับไปที่บริเวณเดิมหลังจากการหมุนข้อมูลภาพแล้วสามารถทำได้โดยใช้สมการ (3.7)

$$\begin{bmatrix} X_i \\ Y_i \end{bmatrix} + \begin{bmatrix} \bar{X} \\ \bar{Y} \end{bmatrix} = \begin{bmatrix} X'_i \\ Y'_i \end{bmatrix} \quad \ldots(3.7)$$

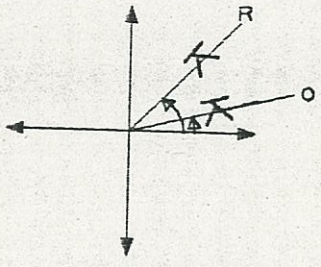
<u>ข้อสังเกต</u> สมการ (3.4) เมื่อ มีค่าเป็นบวกจุดภาพต่างๆ บนระนาบ X,Y จะถูกหมุนไปในทิศทางทวนเข็มนาฬิกาดังรูป 3.4 แต่กรณีที่ มีค่าเป็นลบจุดภาพต่าง ๆ บนระนาบ X,Y จะถูกหมุนไปในทิศทางตามเข็มนาฬิกา

3.2.1 การหามุมหมุนภาพวัตถุอัตโนมัติอย่างง่าย

จากข้างต้นในหัวข้อ 3.1 เป็นการนำข้อมูลภาพมาหาค่า Eigenvalve และ Eigenvector ซึ่งทำให้ทราบค่ามุมของแนวแกนหลัก (θ_{E1}) มุมของแนวแกนรอง (θ_{E2}) และค่าความชัน (S) ด้วยสมการ (3.4) ซึ่งมีพารามิเตอร์อยู่ 2 ตัวคือมุมของแนวแกนหลัก (θ_{E1}) และค่าความชัน (S) ที่สามารถจะนำมาหามุมหมุนภาพได้อย่างง่ายด้วยการ

พิจารณากรณีต่าง ๆ ได้ 4 กรณีคือ

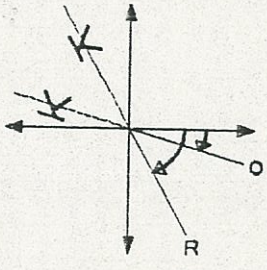
กรณีที่ 1 ภาพวัตถุอ้างอิง (R) มีค่า S เป็นบวก
ภาพวัตถุที่พิจารณา (O) มีค่า S เป็นบวก



$$\theta_R = \hat{R} - \hat{O}$$

รูป 3.6 ก ช่วยประกอบการพิจารณากรณีที่ 1

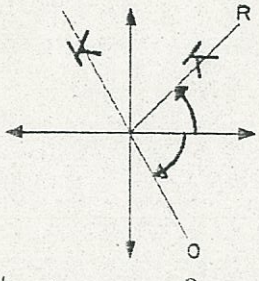
กรณีที่ 2 ภาพวัตถุอ้างอิง (R) มีค่า S เป็นลบ
ภาพวัตถุที่พิจารณา (O) มีค่า S เป็นลบ



$$\theta_R = -(|\hat{R}| - |\hat{O}|)$$

รูป 3.6 ข ช่วยประกอบการพิจารณากรณีที่ 2

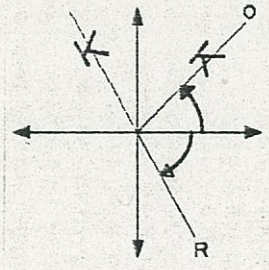
กรณีที่ 3 ภาพวัตถุอ้างอิง (R) มีค่า S เป็นบวก
ภาพวัตถุที่พิจารณา (O) มีค่า S เป็นลบ



$$\theta_R = (\hat{R} - (180+\hat{O}))$$

รูป 3.6 ค ช่วยประกอบการพิจารณากรณีที่ 3

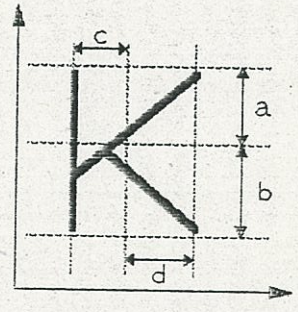
กรณีที่ 4 ภาพวัตถุอ้างอิง (R) มีค่า S เป็นลบ
ภาพวัตถุที่พิจารณา (O) มีค่า S เป็นบวก



$$\theta_R = ((180 + \hat{R}) - \hat{O})$$

รูป 3.6 ง ช่วยประกอบการพิจารณากรณีที่ 4

วิธีการหามุมหมุนภาพวัตถุอัตโนมัติอย่างง่ายที่ได้กล่าวไปข้างต้นนั้นยังมีข้อจำกัดอยู่คือ ภาพวัตถุจะอยู่ทำมุมกันไม่เกิน 180° ซึ่งข้อจำกัดนี้สามารถแก้ได้โดยการหาพารามิเตอร์เพิ่มเติมเพื่อการพิจารณา เช่น หาค่าระยะขอบเขตของแนวแกนหลักและแนวแกนรองดังแสดงรูป 3.7 (เป็นวิธีหนึ่งที่จะหาพารามิเตอร์เพิ่มเติมเพื่อประกอบการพิจารณาหามุมหมุน)



รูป 3.7 แสดงวิธีการหาพารามิเตอร์เพิ่มเติมเพื่อใช้ในการพิจารณาหามุมหมุนแบบอัตโนมัติ

3.2.2 การชดเชยข้อมูลภาพที่ขาดหายไปหลังการหมุนข้อมูลภาพ

จะพบว่าผลลัพธ์ที่ได้หลังจากการหมุนข้อมูลแล้วจะเกิดจุดภาพที่ขาดหายไป แบบเว้าแหว่ง หรือโหว ซึ่งเกิดจากความเป็น Discrete ของจุดภาพและของข้อมูล ดังนั้นจึงจำเป็นต้องทำการชดเชยข้อมูลของภาพที่ขาดหายไปหลังจากการหมุนข้อมูลภาพ ซึ่งจะใช้สมการที่ (3.8) ซึ่งเป็นสมการทางลอจิก [7] เข้ามาช่วยแก้ไข

$$B = p + b.g\ (d+c) + d.e\ (b+g) \qquad \ldots(3.8)$$

โดยที่

สัญญลักษณ์ + คือ OR

และสัญญลักษณ์ . คือ AND

a, b, c, d, e, f, g, h เป็นข้อมูลข้างเคียงของ p ดังรูปที่ 3.8

a	b	c
d	p	e
f	g	h

รูปที่ 3.8 p เป็นจุดข้อมูลที่กำลังถูกพิจารณา

โดยกำหนดให้จุดที่มีข้อมูลภาพเป็นจริงหรือ "1" และจุดที่ไม่ใช้ข้อมูลของภาพเป็นเท็จหรือ "0" ซึ่งจะใช้สมการ 3.8 นี้กระทำทั้งภาพ

3.2.3 ผลการทดสอบกับข้อมูลภาพจริง (แบบทุกจุด)

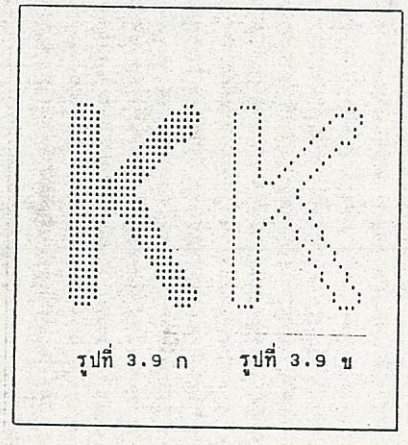
จากหลักการและวิธีการที่กล่าวมาข้างต้นนี้ได้นำมาทดสอบกับข้อมูลภาพขนาด 64x64 จุดภาพ โดยมีลำดับขั้นการทดสอบดังนี้

- นำข้อมูลภาพอ้างอิงแสดงในรูป 3.9 ก มาหามุม θ_{ER} และหาขอบเขตภาพวัตถุอ้างอิงสำหรับการเข้ารหัสลูกโซ่แสดงในรูป 3.9 ข
- นำข้อมูลภาพที่พิจารณาแสดงในรูป 3.10 ก และ 3.11 ก มาหามุม θ_{EO} หาขอบของวัตถุที่พิจารณาสำหรับการเข้ารหัสลูกโซ่ แสดงในรูป 3.10 ข และ 3.11 ข
- หามุมที่ต้องการหมุน θ_R จากค่า θ_{ER} และ θ_{EO}
- ทำการหมุนข้อมูลภาพไป θ_R องศา แสดงในรูป 3.12 ก และ 3.13 ก

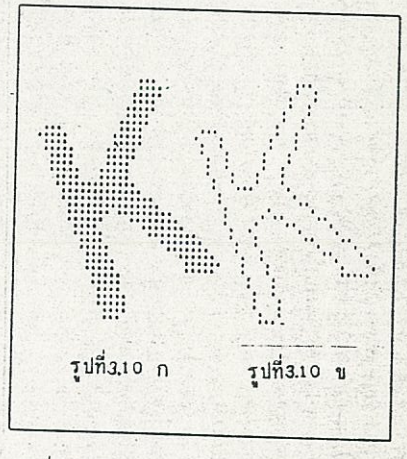
- ชดเชยข้อมูลภาพวัตถุที่ขาดหายไป เนื่องจากการหมุนแสดงในรูป 3.12 ข และ 3.13 ข

- หาขอบเขตของภาพวัตถุหลังจากการหมุนสำหรับเข้ารหัสลูกโซ่แสดงในรูป 3.12 ค และ 3.13 ค

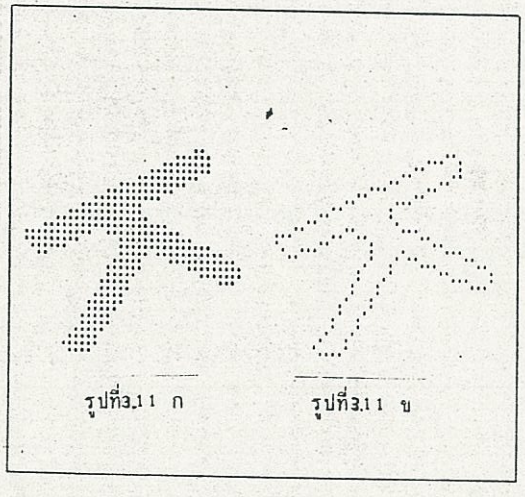
- หาสหสัมพันธ์ (correlation) ข้อมูลภาพ 3.9 ข กับข้อมูลภาพ 3.10 ข , 3.11 ข , 3.12 ค , 3.13 ค



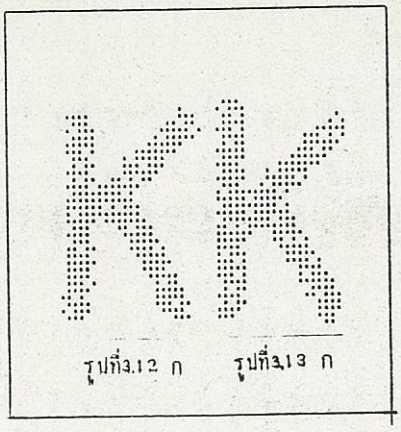
รูปที่ 3.9 ก แสดงภาพวัตถุ K
รูปที่ 3.9 ข แสดงขอบของภาพวัตถุ K ซึ่งใช้สำหรับหารหัสลูกโซ่



รูปที่ 3.10 ก แสดงภาพวัตถุ K ซึ่งมีแนวแกนต่างจากรูปที่ 3.9 ก
รูปที่ 3.10 ข แสดงขอบของภาพวัตถุ K ซึ่งใช้สำหรับหารหัสลูกโซ่

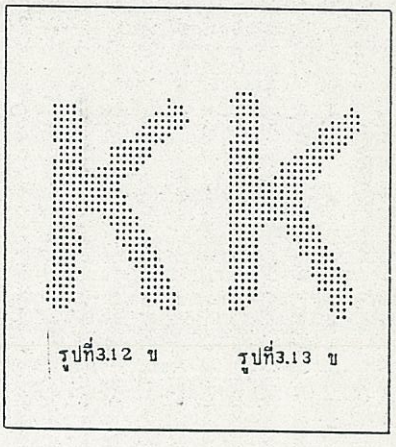


รูปที่ 3.11 ก แสดงภาพวัตถุ K ซึ่งมีแนวแกนต่างจากรูปที่ 3.9 ก
รูปที่ 3.11 ข แสดงขอบของภาพวัตถุ K ซึ่งใช้สำหรับหารหัสลูกโซ่

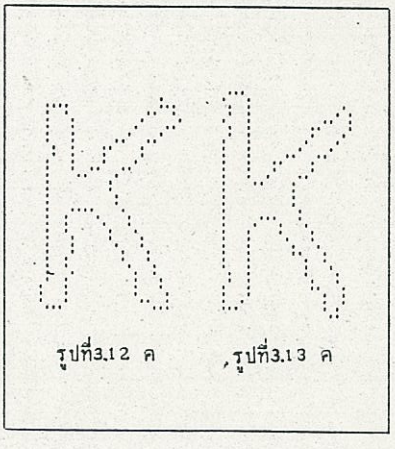


รูปที่ 3.12 ก แสดงผลที่ได้จากการย้าย
แนวแกนของรูปที่ 3.10 ก

รูปที่ 3.13 ก แสดงผลที่ได้จากการย้าย
แนวแกนของรูปที่ 3.11 ก



รูปที่ 3.12 ข , 3.13 ข แสดงผลการชดเชยข้อมูลที่ขาดหายในระหว่าง
การย้ายข้อมูลของ รูปที่ 3.12 ก , 3.13 ก



รูปที่ 3.12 ค , 3.13 ค แสดงขอบของภาพวัตถุ รูปที่ 3.12 ข , 3.13 ข
ซึ่งใช้สำหรับหารหัสลูกโซ่

ขั้นตอนการทำ Correlation

เราจะหาค่ารหัสลูกโซ่ของข้อมูลภาพ รูป 3.9 ข (ใช้เป็นตัวมาตรฐาน) และรหัสลูกโซ่ของข้อมูลภาพที่กำลังตรวจสอบ รูป 3.12 ค และ รูป 3.13 ค มาเปรียบเทียบกัน

รหัสลูกโซ่ของรูป 3.9 ข ได้ดังนี้

5666666666666666666666666667001222222222211177767767767767001233233
233233233211111110121234545555555555543222222222344

รหัสลูกโซ่ของรูป 3.10 ข ได้ดังนี้

6565656565665654432322323445767667676766767676676676676001223223232323
311170707707707760000133334333433433322121221212123444

รหัสลูกโซ่ของรูป 3.11 ข ได้ดังนี้

5445454545445454545454545446770100101010707665565655656565557000121121
121211211707007070070012343434434343443321001010101022344

รหัสลูกโซ่ของรูป 3.12 ค ได้ดังนี้

6545555454665454322222223244466666765666666666666666666566700122213212
232221000777677677666707002232323232332332321101121011013243

รหัสลูกโซ่ของรูป 3.13 ค ได้ดังนี้

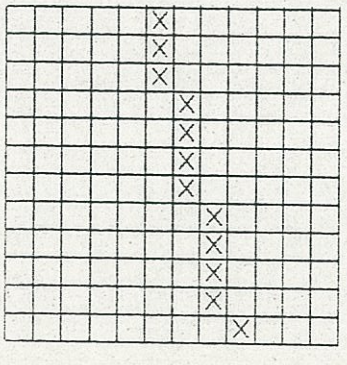
5765765666666666666666666765666001122222222211007677607667607667021223
2323323323323222101210111122355454655545465543222222222222344

3.3 การหมุนข้อมูลภาพวัตถุแบบบางจุด

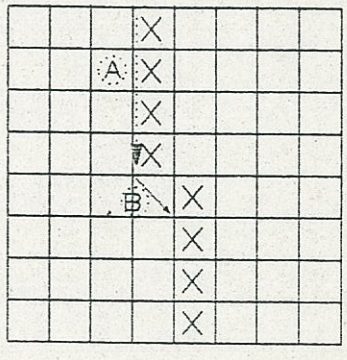
จากขั้นตอนการย้ายข้อมูลภาพวัตถุแบบทุกๆจุดนั้น ยังมีข้อผิดพลาดอยู่บ้าง ซึ่งเกิดจากความเป็น Discrete ของข้อมูลภาพและตัวจอภาพที่เป็นสาเหตุหลัก ที่ทำให้จุดข้อมูลภาพหลังการหมุนหายไป และข้อสำคัญอีกประการหนึ่งคือเวลาที่ใช้ในการหมุนข้อมูลภาพแบบทุกจุดจะกินเวลานาน ดังนั้นจึงได้ทำการพัฒนา แนวทางการแก้ไขในส่วนนี้โดยจะเลือกหมุนเฉพาะจุดแรกที่มีการเริ่มเปลี่ยนแนวของเส้นตรง (Break point) จากนั้นจึงทำการลากเส้นระหว่างจุด Break point โดยใช้วิธีการทางกราฟฟิกเข้าช่วย

3.3.1 การหาจุดเปลี่ยนแนวของเส้นอย่างง่าย (Break point)

หลักการหาจุดเปลี่ยนแนวของเส้นนี้ได้จากการศึกษารูปแบบของเส้นตรง ที่มีค่าความชันต่างๆ กันจะพบว่าเส้นตรงนั้นประกอบด้วยกลุ่มของจุดต่างๆที่เรียงต่อกันดังรูปที่ 3.14 และรูปที่ 3.15 ซึ่งกลุ่มจุดเหล่านี้เมื่อเข้ารหัสทิศทางแล้วจะมีแนวทิศทางแบ่งออกได้มากที่สุดเพียง 2 แนวเท่านั้น ซึ่งจะมีทิศทางที่ไปในทิศทางเดียวกันส่วนมากแนวหนึ่ง ซึ่งจะขอเรียกแนวทิศทางนี้ว่า **แนวทิศทางหลัก A** และอีกแนวหนึ่งจะมีแนวทิศทางเปลี่ยนไปจากแนวทิศทางหลักไม่เกิน 45 องศา (หรือหนึ่งรหัสลูกโซ่ในกรณี 8-connectivity) ซึ่งแนวทิศทางที่เปลี่ยนไปจากแนวทิศทางหลักนี้จะขอเรียกว่า **แนวทิศทางรอง B** สำหรับแนวทิศทางรองนี้โดยปกติแล้วจะขึ้นอยู่ระหว่างแนวทิศทางหลักเป็นระยะๆ



รูป 3.14 แสดงกลุ่มจุดของเส้นตรง



รูป 3.15 แสดงแนวทิศทางหลัก A และแนวทิศทางรอง B

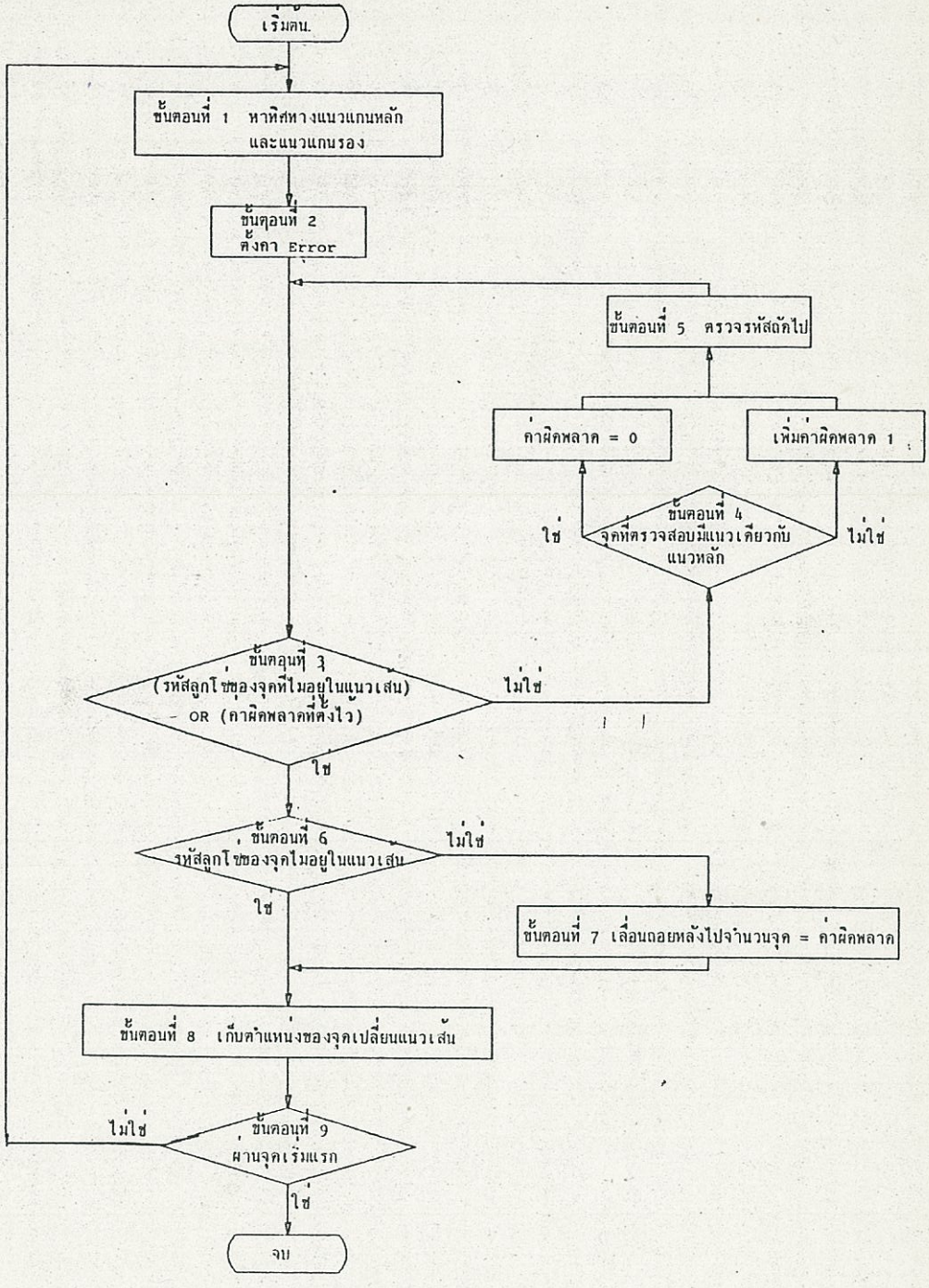
จากคุณลักษณะเด่นดังกล่าวนี้จึงพอสรุปและวางเป็นกฎเกณฑ์เพื่อหาจุดเปลี่ยนแนวของเส้นตรงได้ดังนี้

ให้ถือจุดแรกที่พบว่ามีการเปลี่ยนแนวเส้นหลักหรือแนวเส้นรองไปจากเดิม มากกว่าหรือเท่ากับ 90 องศา (2 รหัสลูกโซ่ในกรณี 8-conetivity) เป็นจุดที่เริ่มเปลี่ยนแนวของเส้นและจะต้องมีการตั้งค่าผิดพลาดไว้สำหรับจำนวนของรหัสลูกโซ่ในทิศทางรอง โดยที่ค่าผิดพลาดนี้จะขึ้นอยู่กับจำนวนกลุ่มจุดของแนวทิศทางหลัก ผลจากการทดลองทำให้ได้ค่าผิดผลาดที่ยังยอมรับได้ดังแสดงไว้ในตารางที่ 3.1

ตารางที่ 3.1 กำหนดจำนวนรหัสลูกโซ่ในทิศทางรองที่ยังยอมรับได้

จำนวนจุดของแนวเส้นหลัก	ค่าผิดพลาดที่ยังยอมรับได้ (หน่วยเป็นจุด)
1 ⟶ 2	3
3 ⟶ 4	2
5 ขึ้นไป	1

จากข้อกำหนดดังกล่าวสามารถเขียนเป็นผังการตัดสินใจเพื่อหาจุดเปลี่ยนแนวเส้นได้ตามผังที่ 3.3 ดังนี้



ผังที่ 3.3 แสดงขั้นตอนการตัดสินใจเพื่อหาจุดเปลี่ยนแนวเส้น

การทำงานในผังของการตัดสินใจสำหรับหาจุดเปลี่ยนแนวเส้นคือ

ขั้นตอนที่ 1 นำรหัสลูกโซ่(แนวของจุด) มาพิจารณาหาจำนวนกลุ่มจุดที่เป็นแนวทิศทางหลัก ดังตัวอย่างที่แสดงในรูป 3.14 และ 3.15

ขั้นตอนที่ 2 ตั้งค่าผิดพลาดที่ยังยอมรับได้ โดยใช้ตาราง 3.1 ประกอบ

ขั้นตอนที่ 3 ตรวจสอบทิศทางของจุด(รหัสลูกโซ่) มีแนวทิศทางต่างออกไปจากแนวทิศทางหลักหรือแนวทิศทางรองมากกว่าหรือเท่ากับ 90 องศา หรือค่าผิดพลาดมีค่ามากกว่าที่ตั้งไว้ (ในขั้นตอนที่ 2) หรือไม่

ขั้นตอนที่ 4 ทิศทางของจุดที่พิจารณาในขั้นตอนที่ 3 อยู่ในแนวทิศทางหลักหรือไม่ ถ้าใช่ก็ให้ตัวนับค่าผิดพลาดเป็น 0 ถ้าไม่ใช่ก็ให้ตัวนับค่าผิดพลาดเพิ่มขึ้น 1

ขั้นตอนที่ 5 พิจารณาจุดถัดไป

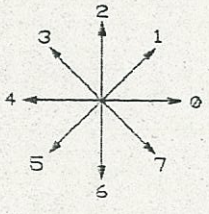
ขั้นตอนที่ 6 พิจารณาเงื่อนไขใดเป็นสาเหตุทำให้การทำงานหลุดออกจากลูปของขั้นตอนที่3 ถ้าเกิดจากเงื่อนไขรหัสลูกโซ่ของจุดไม่อยู่ในแนวทิศทางหลัก และแนวทิศรอง ก็ให้ทำในขั้นตอนที่ 8 ได้เลย

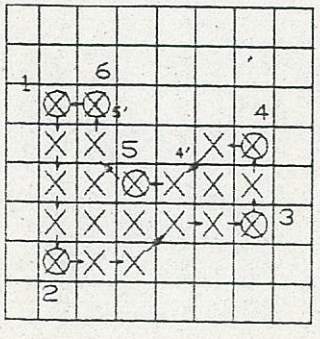
ขั้นตอนที่ 7 เนื่องจากตรวจพบในขั้นตอนที่ 6 ว่า ตัวนับค่าผิดพลาดมีค่ามากกว่าที่ตั้งไว้ เพราะฉะนั้นจำเป็นจะต้องถอยหลังกลับไปตามจำนวนค่าผิดพลาดเพื่อกลับไปเริ่มพิจารณาหาจุดเปลี่ยนแนวเส้นถัดไป

ขั้นตอนที่ 8 เป็นขั้นตอนที่พบจุดเปลี่ยนแนวเส้น ดังนั้นจะทำการเก็บตำแหน่งและข้อมูลของจุดเปลี่ยนแนวเส้น

ขั้นตอนที่ 9 เป็นการตรวจสอบถึงจุดต่างๆ ที่พิจารณาผ่านมานั้นได้ผ่านจุดเริ่มแรกแล้วหรือยัง ถ้ายังก็ให้กลับไปเริ่มต้นใหม่ถ้าผ่านก็ให้หยุดการทำงานได้

ตัวอย่างของขั้นตอนการหาจุดแรกของการเปลี่ยนแนวของเส้นอย่างง่ายมีดังนี้



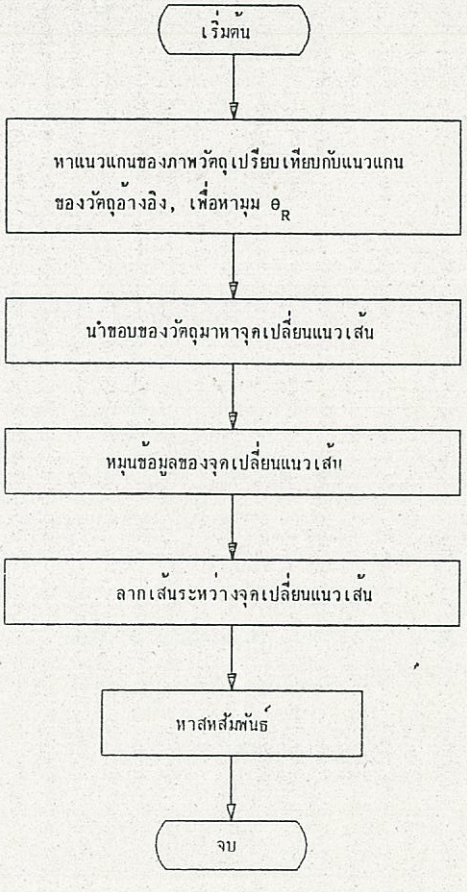


รูปที่ 3.16 แสดงการหาจุดเปลี่ยนแนวเส้น

จากรูปที่ 3.16 นั้นได้แสดงการหาจุดเปลี่ยนที่แนวเส้น โดยที่ 6 ⟶ 1 , 1 ⟶ 2, 2 ⟶ 3 และ 3 ⟶ 4 เป็นจุดที่มีการเปลี่ยนแนวทิศทางรหัสไปจากเดิม มากกว่าหรือเท่ากับ 90 องศา และส่วนของ 4 ⟶ 5 ที่จุด 5 ได้เปลี่ยนแนวทิศทางรหัสไปจาก จุด 4' (แนวทิศทางรอง) ไปจากแนวทิศทางเดิมมากกว่า 90 องศา ส่วนในช่วงของ 5 ⟶ 6 ก็ทำนองเดียวกับช่วงของ 4 ⟶ 5

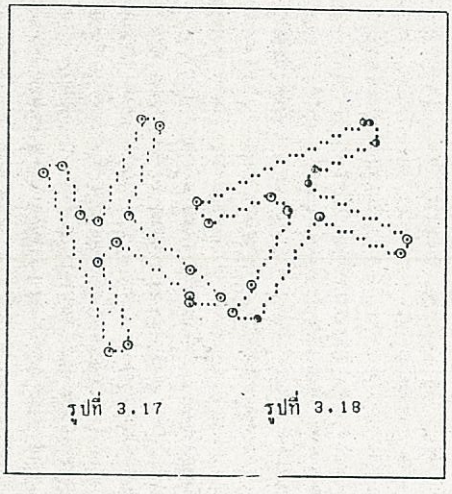
3.3.2 แสดงผลการทดสอบกับข้อมูลภาพจริง (แบบบางจุด)

จากกฎเกณฑ์ที่ได้กล่าวในข้างต้นนี้ สามารถนำมารวบรวมเป็นขบวนการเดียวกันเพื่อใช้ในการหมุนข้อมูลของภาพวัตถุเพียงบางจุด โดยมีลำดับขั้นตอนการทำงานดังผังที่ 3.4 ข้างล่างนี้

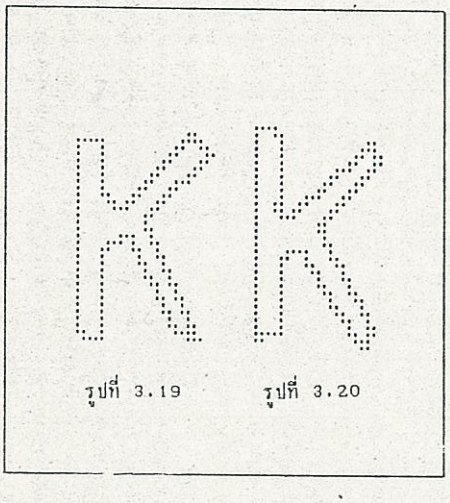


ผังที่ 3.4 ขั้นตอนการตรวจสอบภาพวัตถุกรณีที่แนวแกนของภาพวัตถุอ้างอิง กับวัตถุที่กำลังตรวจสอบ มีแนวแกนที่แตกต่างกัน (พัฒนามาจาก ผังที่ 3.1)

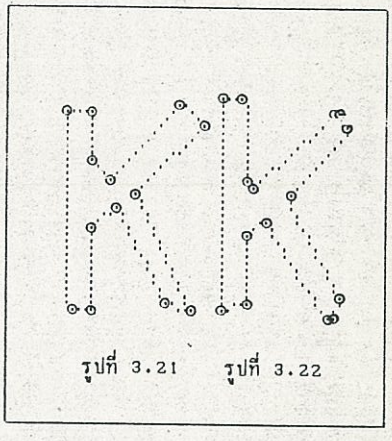
จากการใช้ผังการทำงานข้างต้นนั้นจะให้ผลของการหาจุดเปลี่ยนแนวดังรูป 3.17 และรูป 3.18 จุดเปลี่ยนแนวเส้นในภาพแสดงด้วยจุดที่ถูกวงกลมล้อมรอบไว้ ส่วนรูป 3.19 และ 3.20 แสดงผลหลังจากการย้ายข้อมูลบางจุดของรูป 3.17 และ 3.18 ตามลำดับ จากนั้นนำขอบของวัตถุรูป 3.19 และ 3.20 มาหาจุดเปลี่ยนแนวอีกครั้ง ดังแสดงในรูป 3.21 และ 3.22 เพื่อใช้สำหรับการหาสหสัมพันธ์เป็นส่วน ๆ ส่วนรูป 3.23 เป็นรูปวัตถุอ้างอิง

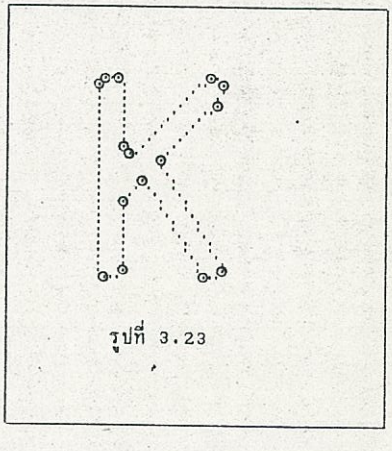


รูปที่ 3.17,3.18 แสดงจุดเปลี่ยนแนวเส้น(จุดที่วงกลมล้อมรอบ) ของรูปที่ 3.10 , 3.11 ตามลำดับ



รูปที่ 3.19,3.20 แสดงผลหลังจากการย้ายข้อมูลบางจุดของรูปที่ 3.17,3.18 พร้อมกับลากเส้นระหว่างจุด





รูปที่ 3.21,3.22,3.23 แสดงขอบเขตวัตถุรูปที่ 3.17,3.18 และ 3.19 พร้อมกับแสดงจุดเปลี่ยนแนว (ล้อมรอบด้วยวงกลม)

ในขั้นตอนการหาสหสัมพันธ์นี้เราตัดข้อมูลของรหัสลูกโซ่ออกเป็นส่วนๆ โดยใช้จุดเปลี่ยนแนวเป็นจุดแยกข้อมูลของรหัสลูกโซ่แล้วจึงทำการหาสหสัมพันธ์ โดยที่จุดเปลี่ยนแนวต่างๆ ได้แสดงในรูปที่ 3.21, 3.22 และ 3.23

รหัสลูกโซ่ของรูป 3.21

ส่วนที่ 1	55555555555
ส่วนที่ 2	333
ส่วนที่ 3	2222222
ส่วนที่ 4	4444
ส่วนที่ 5	66666666666666666666666666667
ส่วนที่ 6	000
ส่วนที่ 7	222222222222
ส่วนที่ 8	1101
ส่วนที่ 9	76767677676767
ส่วนที่ 10	0700
ส่วนที่ 11	32323232323232323
ส่วนที่ 12	11111101111
ส่วนที่ 13	3334

รหัสลูกโซ่ของรูป 3.22

ส่วนที่ 1	66666666666666666666666666666666
ส่วนที่ 2	0100
ส่วนที่ 3	2222222222
ส่วนที่ 4	110
ส่วนที่ 5	76776776777677
ส่วนที่ 6	0
ส่วนที่ 7	212
ส่วนที่ 8	3232332323332322
ส่วนที่ 9	1111111112
ส่วนที่ 10	32
ส่วนที่ 11	4
ส่วนที่ 12	5555455554555
ส่วนที่ 13	3
ส่วนที่ 14	222222222223
ส่วนที่ 15	444

รหัสลูกโซ่ของรูป 3.23

ส่วนที่ 1	5
ส่วนที่ 2	666666666666666666666666667
ส่วนที่ 3	001
ส่วนที่ 4	2222222222
ส่วนที่ 5	111
ส่วนที่ 6	77767767767767
ส่วนที่ 7	001
ส่วนที่ 8	2332332332332332
ส่วนที่ 9	111111101
ส่วนที่ 10	212
ส่วนที่ 11	34
ส่วนที่ 12	5455555555554
ส่วนที่ 13	3
ส่วนที่ 14	2222222223
ส่วนที่ 15	44

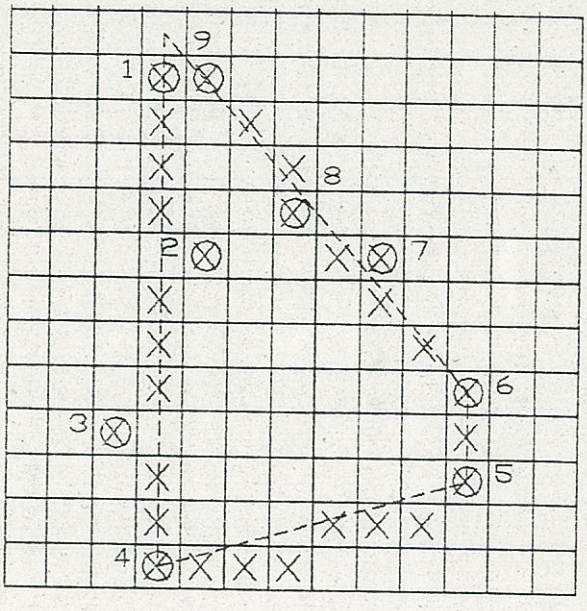
บทที่ 4
การจดจำรูปร่างวัตถุในรูปมุมและระยะทาง
(Angle and Distance information of Object)

จากบทก่อน ๆ ที่ได้กล่าวมานั้น การเก็บข้อมูลของภาพวัตถุจะเก็บอยู่ในลักษณะรหัสลูกโซ่ของ Freeman แบบ 8-ทิศทาง [8] ซึ่งรหัสลูกโซ่ดังกล่าวของข้อมูลภาพแต่ละภาพยังมีขนาดใหญ่อยู่ และใช้เวลามากเมื่อนำรหัสข้อมูลภาพไปทำสหสัมพันธ์ ดังนั้นในบทนี้จึงได้พัฒนาในส่วนของรหัสข้อมูลภาพวัตถุ โดยจะจัดข้อมูลภาพวัตถุซึ่งเดิมทีเป็นรหัสลูกโซ่ของ Freeman แบบ 8-ทิศทาง ให้อยู่ในรูปของระยะทางและมุม ข้อมูลของมุมและระยะทางเหล่านี้จะถูกนำมาจัดให้อยู่เรียงตามกันไป ซึ่งจะใช้เทคนิคการหาจุดเปลี่ยนแนวเส้นอย่างง่ายในบทที่ 3 มาพัฒนาให้ดีขึ้น

เนื่องจากวัตถุใดๆนั้น ขอบของวัตถุสามารถประมาณได้ด้วยเส้นตรงที่ต่อเรียงกันจนครบรอบวัตถุ เส้นตรงสองเส้นที่ต่อกันจะก่อให้เกิดเป็นมุมขึ้นมา ฉะนั้นไม่ว่าวัตถุจะหมุนอยู่ในทิศใดก็ตาม โดยทางทฤษฎีแล้วมุมกับระยะทางของเส้นตรงต่างๆ ที่ประกอบขึ้นเป็นขอบของวัตถุนั้นจะไม่มีการเปลี่ยนแปลง ดังนั้นการตรวจสอบชนิดของวัตถุจึงทำได้โดยการเปรียบเทียบหรือหาสหสัมพันธ์ระหว่างรหัสลูกโซ่มุม-ระยะทางของวัตถุใด ๆ กับรหัสลูกโซ่มุม-ระยะทางของวัตถุอ้างอิงในหน่วยความจำ จะเห็นได้ชัดว่าข้อมูลของมุม-ระยะทางหนึ่งข้อมูลนั้น สามารถใช้แทนข้อมูลของจุดภาพบนขอบวัตถุที่เป็นเส้นตรงได้เป็นจำนวนมาก จึงทำให้ขนาดของรหัสลูกโซ่มุม-ระยะทางสั้นกว่ารหัสลูกโซ่ใน [9] เป็นอย่างมาก ซึ่งมีผลทำให้ลดเวลาที่ใช้ในการตรวจสอบชนิดของวัตถุ และลดขนาดของหน่วยความจำลงได้เป็นอย่างมาก

ในทางปฏิบัตินั้นการหาจุดเปลี่ยนแนวเส้นนั้นยังจะมีปัญหาอยู่เนื่องจาก noise ของข้อมูลภาพ และจากการตัดค่าความเข้มระดับสีเทา (ตัด Threshold) ที่เป็นต้นเหตุทำให้ภาพ 2 ระดับเกิดจุดเว้าและแหว่งขึ้นตามขอบของภาพวัตถุ ดังนั้นจึงทำให้เกิดจุดเปลี่ยนแนวเส้นขึ้นในส่วนที่เป็นแนวเส้นตรงเดียวกัน ดังรูปที่ 4.1 จุดเว้าและแหว่งคือ ตำแหน่งหมายเลข 2 และ 3 ถ้าหากไม่พิจารณาให้ดีก็จะถือเอาจุดเหล่านี้เป็นจุดเปลี่ยนแนวไปด้วย แต่โดยความเป็นจริงแล้วจะมีเส้นตรงเพียงเส้นเดียวที่ลากจากจุดที่ 1 ไปยังจุดที่ 4

ดังนั้นก่อนที่จะตรวจหาจุดเปลี่ยนแนวนั้นควรจะทำการปรับแต่งข้อมูลของจุดเว้าและแหว่งให้ราบเรียบเสียก่อน ซึ่งหลักการในการปรับแต่งจะได้กล่าวในหัวข้อถัดไป



รูปที่ 4.1 แสดงขอบของข้อมูลภาพมีการเว้าและแหว่ง และเกิดจุดเปลี่ยนแนวเส้นที่ 2, 3, 7 และ 8 ชั้น เมื่อหาจุดเปลี่ยนแนวเส้นโดยใช้วิธีในหัวข้อ 3.3.1

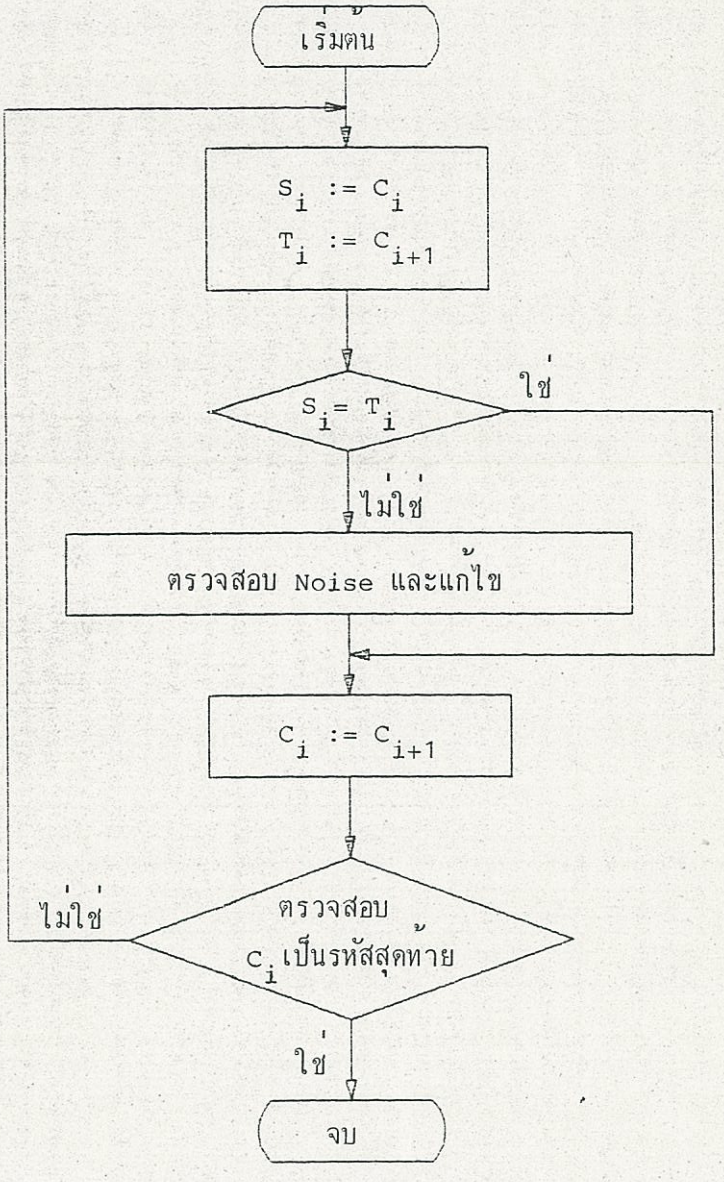
4.1 วิธีการปรับรหัสลูกโซ่ให้ราบเรียบ

จากการทดลองถ่ายข้อมูลภาพ (10 ภาพ) แล้วนำมาแปลงค่าความเข้มสีเทาให้เป็นภาพความเข้ม 2 ระดับ เพื่อดู noise ที่เกิดขึ้นตามขอบของข้อมูลภาพ ในที่สุดสามารถสรุปได้เป็นหลักการง่าย ๆ 2 ข้อคือ

ก. จุดภาพที่เป็น noise จะอยู่ติดต่อกันไม่เกิน 3 จุดภาพ

ข. ค่าทิศทางของรหัส (noise) จะมีทิศทางแกว่งไปมาแบบสมมาตร

หลักการดังกล่าวนี้จะถูกนำมาสร้างเป็นเงื่อนไข สำหรับการปรับรหัสลูกโซ่ให้ราบเรียบ โดยมีลำดับขั้นตอนการปรับรหัสลูกโซ่ตามผังที่ 4.1 ต่อไปนี้



เมื่อ C_i , S_i คือค่ารหัสของข้อมูลตัวที่ i

C_{i+1} , T_i คือค่ารหัสของข้อมูลตัวถัดไปคือตัวที่ i + 1

ผังที่ 4.1 ขั้นตอนทำการปรับรหัสลูกโซ่ให้ราบเรียบ

จากผังข้างบนจะใช้สำหรับปรับข้อมูลรหัสลูกโซ่ทุกๆจุด ซึ่งในระหว่างขบวนการตรวจสอบนั้นถ้าพบว่ารหัสลูกโซ่มี noise ปะปนอยู่จะทำการแก้ไขทันที แล้วสร้างรหัสให้ใหม่ ซึ่งรายละเอียดจะได้อธิบายดังต่อไปนี้

ขั้นตอนที่ 1 นำรหัสลูกโซ่ (C_i) และรหัสลูกโซ่ตัวถัดไป (C_{i+1}) มาตรวจสอบโดยพักข้อมูลไว้ที่ S_i และ T_i ตามลำดับ สาเหตุที่ต้องพักข้อมูลไว้ เนื่องจากรหัสลูกโซ่ (C_i , C_{i+1}) อาจถูกทำการแก้ไขเนื่องจากเป็นรหัสของ noise

ขั้นตอนที่ 2 นำข้อมูลรหัสลูกโซ่ที่พักไว้ใน S_i และ T_i มาตรวจสอบว่าเป็นรหัสลูกโซ่ที่มีทิศทางเดียวกันหรือไม่ ถ้าหากใช่รหัสลูกโซ่ที่มีทิศทางเดียวกันจะไม่มีการพิจารณา มิเช่นนั้นจะพิจารณา ว่ารหัสเป็นรหัสของ noise หรือไม่

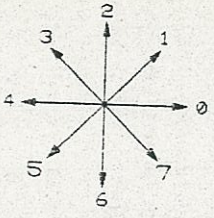
ขั้นตอนที่ 3 เป็นการตรวจสอบและพิจารณารหัส noise ซึ่งจะใช้เงื่อนไข 2 ประการหลักที่ได้กล่าวไว้ในข้างต้นมาช่วยตัดสินใจว่ารหัสลูกโซ่ที่พิจารณาอยู่เป็น noise ของข้อมูลภาพหรือไม่ ถ้าหากตรวจสอบพบว่าเป็น noise (คือมีการแกว่งไปมาของรหัสอยู่บริเวณขอบของข้อมูล) แล้วก็จะทำการแก้ไขข้อมูลของรหัสและขอบภาพวัตถุเสียใหม่ สำหรับรายละเอียดในขั้นตอนนี้จะขอยกไปกล่าวในหัวข้อถัดไป (4.1.1)

ขั้นตอนที่ 4 พิจารณารหัสลูกโซ่ตัวถัดไป

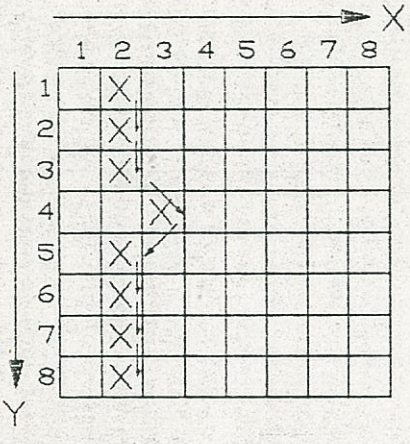
ขั้นตอนที่ 5 เป็นการตรวจสอบว่ากระทำการปรับรหัสลูกโซ่จนครบทั้งภาพแล้วหรือไม่ ถ้ายังไม่ครบจะกลับไปตรวจสอบต่อไปอีก

4.1.1 วิธีการตรวจสอบ noise ที่ปะปนอยู่ตามขอบภาพวัตถุจากรหัสลูกโซ่

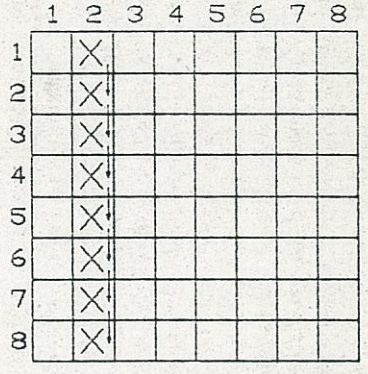
จากเงื่อนไข 2 ประการดังที่ได้กล่าวไว้ข้างต้น เมื่อนำมาตรวจสอบกับขอบของวัตถุในรูปที่ 4.2 โดยใช้รหัสทิศทางลูกโซ่ดังแสดงในรูปที่ 4.2 ก จากขอบวัตถุในรูปที่ 4.2 ข นั้น จะพบว่ารหัสลูกโซ่ของขอบวัตถุ ในรูปที่ 4.2 ข จากตำแหน่งที่ (2,2) ถึง (2,6) คือ 66756 ซึ่งรหัสลูกโซ่ นี้มีการแกว่งไปจากแนวทิศหมายเลข 6 ของ Freeman code



รูปที่ 4.2 ก แสดงทิศทางรหัสของ Freeman



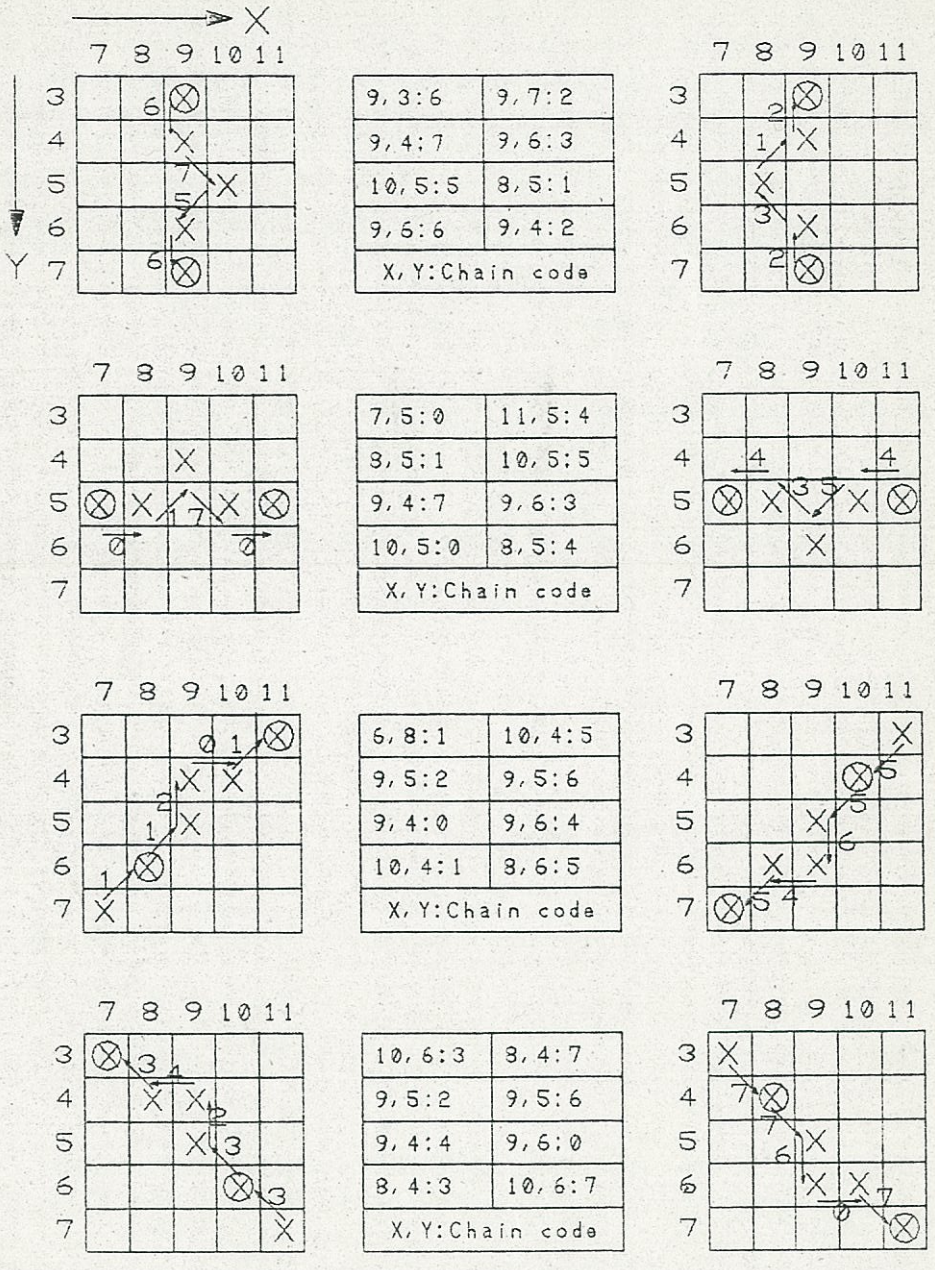
รูปที่ 4.2 ข ขอบของวัตถุที่มี noise



รูปที่ 4.2 ค ขอบวัตถุหลังการปรับให้ราบเรียบ

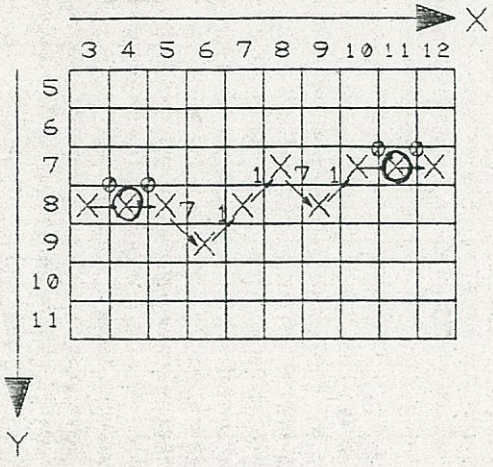
อย่างสมมาตรกล่าวคือ รหัสจะแกว่งไปทางขวาและซ้ายของแนวทิศหมายเลข 6 หลังจากนั้นจะกลับมายังแนวทิศหมายเลข 6 อีกครั้งนั้นก็หมายความว่า ได้พบสัญญาณรบกวนขึ้นที่ขอบของวัตถุแล้ว(ในที่นี้คือตำแหน่ง(3,4) เป็นจุดของสัญญาณรบกวน) ซึ่งจะต้องทำการปรับแต่งขอบของวัตถุบริเวณนี้ใหม่ โดยใช้ Bresenham ออการิทึ่ม [10] มาช่วยในการปรับปรุงขอบของข้อมูลภาพ และในที่สุดจะได้ขอบวัตถุใหม่ที่เป็นแนวเส้นตรงจากตำแหน่ง (2,1) ไปยัง ตำแหน่ง (2,6) ดังรูปที่ 4.2 ค

และสำหรับ noise ที่เกิดขึ้นตามขอบของภาพวัตถุซึ่งมีลักษณะที่แตกต่างกันออกไปดังตัวอย่างในรูปที่ 4.3 และ รูปที่ 4.4 ก็จะยังคงสามารถตรวจสอบหา noise ได้ โดยใช้การตรวจสอบการแกว่งไปมาจากแนวทิศทางหลักดังวิธีข้างต้นนี้

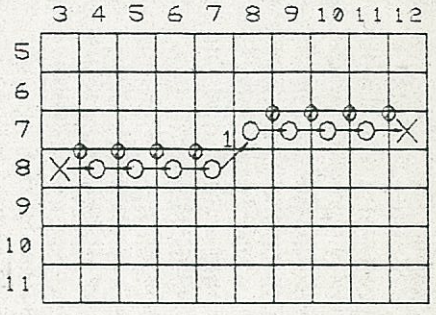


<u>หมายเหตุ</u> (X) เป็นตำแหน่งหัวท้ายของแนวเส้น ซึ่งจะใช้เป็นจุดอ้างอิงสำหรับการลากเส้นขอบของภาพวัตถุ เมื่อพบว่ากลุ่มของขอบข้อมูลภาพมี noise ปะปนอยู่

รูปที่ 4.3 แสดงรูปลักษณ์และรหัสลูกโซ่ของขอบวัตถุที่มี noise ปะปนอยู่



รูปที่ 4.4 ก แสดงขอบของวัตถุที่มี noise ปะปนอยู่



รูปที่ 4.4 ข แสดงขอบของวัตถุหลังจากทำการตรวจสอบ และแก้ไขรหัสลูกโซ่ให้ราบเรียบ

หมายเหตุ (X) เป็นตำแหน่งหัวท้ายของแนวเส้น ซึ่งจะใช้เป็นจุดอ้างอิงสำหรับการลากเส้นขอบของภาพวัตถุ เมื่อพบว่ากลุ่มของขอบข้อมูลภาพมี noise ปะปนอยู่

รูปที่ 4.4 แสดงรูปลักษณ์และรหัสลูกโซ่ของขอบวัตถุที่มี noise ปะปนอยู่ และหลักจากแก้ไขรหัสลูกโซ่ให้ราบเรียบขึ้น

จากรูปที่ 4.3 และ รูปที่ 4.4 จะสังเกตุได้ชัดสำหรับทิศทางของรหัสลูกโซ่จะมีการแกว่งไปมาจากทิศทางเดิม $\pm 45^{\circ}$ (หรือ ± 1 รหัส) และในที่สุดสามารถสรุปทิศทางการแกว่งที่ใช้ในพิจารณหาจุด noise ได้ดังตารางที่ 4.1

ตารางที่ 4.1 สรุปทิศทางการแกว่งไปมาแบบสมมาตรจากทิศทางหลักของรหัสลูกโซ่

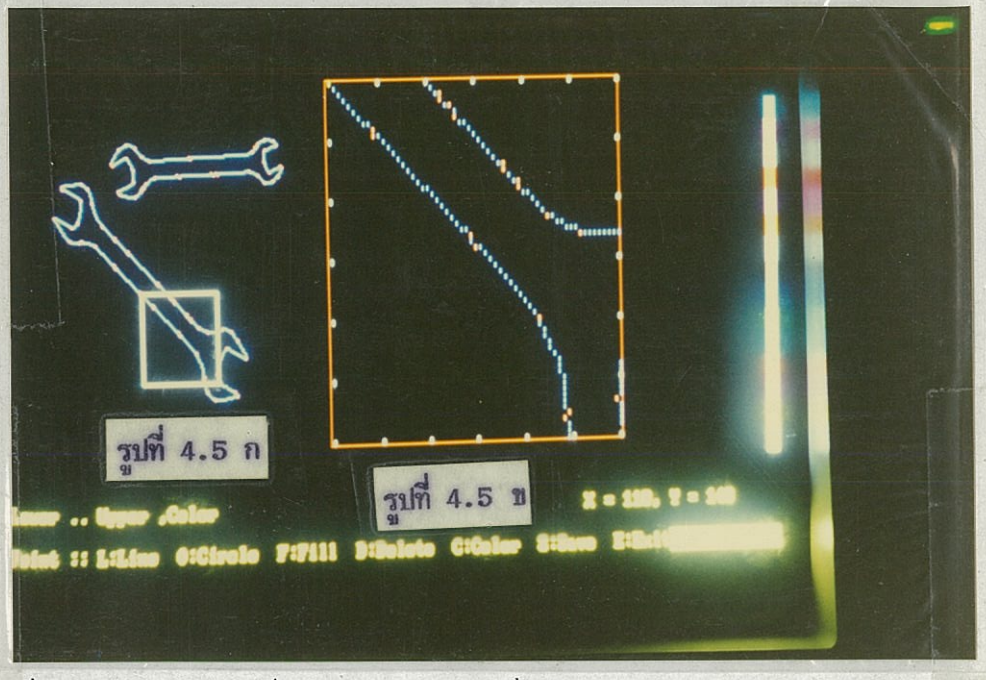
ทิศทางหลักของรหัสลูกโซ่	ทิศทางการแกว่งของรหัสที่จะนำมาพิจารณาว่าเป็นจุด noise	
	ในทิศทางทวนเข็ม	ในทิศทางตามเข็ม
0	1	7
1	2	0
2	3	1
3	4	2
4	5	3
5	6	4
6	7	5
7	0	6

ในที่สุดเมื่อปรับแต่งข้อมูลของเส้นตรงให้ราบเรียบ ขบวนการต่อไปก็จะทำการตรวจหาจุดเปลี่ยนแนว (break point) ของหัวข้อที่ 3.3.1

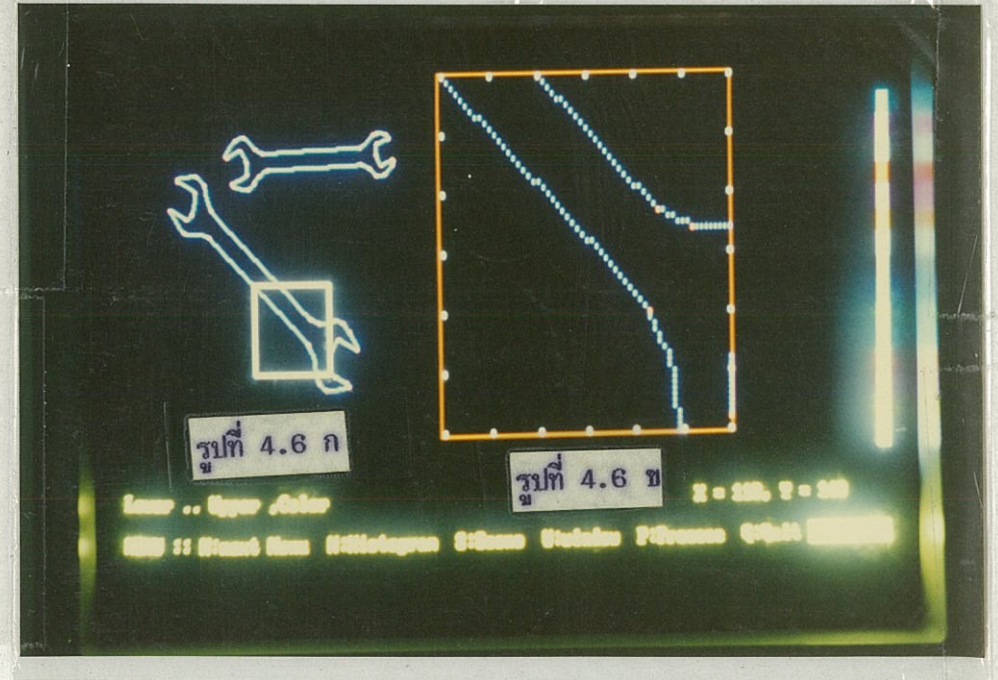
4.1.2 การทดสอบกับข้อมูลภาพที่ได้จากกล้อง

จากการทดสอบกับภาพขอบวัตถุ ของประแจปากตายในรูปที่ 4.5 ก ซึ่งเมื่อขยายให้ภาพใหญ่ขึ้นจะเห็นดังรูปที่ 4.5 ข โดยที่การตรวจสอบหาจุดเปลี่ยนแนวในรูปที่ 4.5 นี้ยังไม่ได้ทำการปรับแต่งข้อมูลของขอบวัตถุให้ราบเรียบ เมื่อทำการตรวจหาจุดเปลี่ยนแนวเส้นก็จะได้ตำแหน่งของจุดเปลี่ยนแนวเส้นดังได้ทำจุดสีแดงเป็นที่สังเกตุในรูปที่ 4.5 ข ซึ่งจะพบว่ายังมีจุดเปลี่ยนแนวเส้นบางจุดที่หาได้นั้น อยู่ในแนวเส้นตรงเดียวกันจึงทำให้รหัสลูกโซ่แบบมุม-ระยะทางมีขนาดยาว แต่หลังจากได้ใช้กระบวนการปรับแต่งข้อมูลของขอบวัตถุให้ราบเรียบกับรูปที่ 4.5 แล้วจึงทำการหาจุดเปลี่ยนแนวเส้นตามหัวข้อ 3.3.1 ก็จะได้จุดเปลี่ยนแนวเส้นดังแสดงในรูปที่4.6 ก ซึ่งเมื่อขยายให้ภาพใหญ่ขึ้นและทำวงกลมไว้สำหรับ

เป็นจุดสังเกตุ จะเห็นจุดเปลี่ยนแนวเส้นต่างๆ ดังในรูปที่ 4.6 ข นั้นคือรูปที่ 4.6 ข จะได้จุดเปลี่ยนแนวเส้นที่ถูกต้องกว่า และยังให้ขนาดของรหัสลูกโซ่แบบมุม-ระยะทางที่สั้นกว่าเมื่อเทียบกับรหัสลูกโซ่แบบมุม-ระยะทางของรูปที่ 4.5 ข



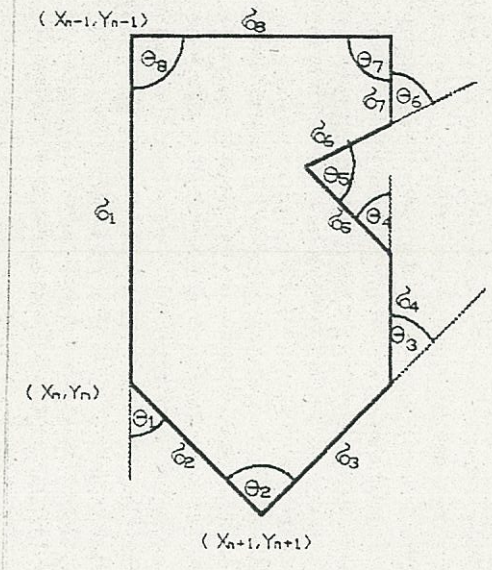
รูปที่ 4.5 แสดงจุดเปลี่ยนแนวของขอบวัตถุที่หาได้ก่อนการปรับขอบวัตถุให้ราบเรียบ



รูปที่ 4.6 แสดงจุดเปลี่ยนแนวของขอบวัตถุที่หาได้หลังจากการปรับขอบวัตถุให้ราบเรียบ

4.2 วิธีการแปลงข้อมูลภาพให้อยู่ในรูปของมุมและระยะทาง

หลังจากกระบวนการดีเทคจุดเปลี่ยนแนวของเส้นตรงบนขอบวัตถุในหัวข้อที่ผ่านมา ขั้นต่อไปนี้เป็นการคำนวณหาระยะทางระหว่างจุดเปลี่ยนแนวที่อยู่ประชิดกันพร้อมทั้งคำนวณหามุมที่เกิดจากจุดเปลี่ยนแนว 3 จุด ที่อยู่ถัดกันไป ตัวอย่างเช่นในรูปที่ 4.7 เป็นรูปหลายเหลี่ยม



รูปที่ 4.7 วัตถุรูปหลายเหลี่ยม

วัตถุในรูปที่ 4.7 นั้นจะประกอบด้วยเส้นตรง 8 เส้น และมุม 8 มุม เส้นตรงแต่ละเส้นนั้นสามารถคำนวณหาขนาดความยาวหรือระยะทางระหว่างจุดเปลี่ยนแนว 2 จุดที่อยู่ถัดกันไป จากการใช้สูตรในสมการที่ (4.1) ตัวอย่างเช่นการคำนวณหาระยะทางของ σ_1 ซึ่งอยู่ระหว่างจุดเปลี่ยนแนว (X_{n-1}, Y_{n-1}) กับ (X_n, Y_n) โดย

$$\sigma_1 = \{ (X_{n-1} - X_n)^2 + (Y_{n-1} - Y_n)^2 \}^2 \qquad \text{.....(4.1)}$$

ส่วนมุมที่อยู่ระหว่างเส้นตรง 2 เส้นซึ่งเกิดจากจุดเปลี่ยนแนว 3 จุด สามารถคำนวณได้ โดยใช้ทฤษฎีของ cos ดังสมการที่ (4.2) ตัวอย่างเช่นการคำนวณค่ามุม $\theta 1$

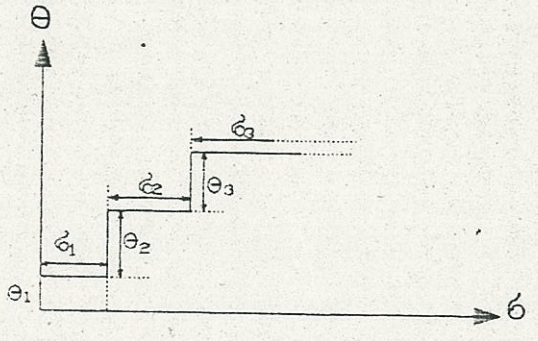
$$\theta_1 = \cos^{-1} \{ (A^2+C^2-B^2) / (2AC)) \} \quad \ldots\ldots\ldots (4.2)$$

เมื่อ
$$A^2 = (X_{n+1} - X_{n-1})^2 + (Y_{n+1} - Y_{n-1})^2$$
$$B^2 = (X_n - X_{n-1})^2 + (Y_n - Y_{n-1})^2$$
$$C^2 = (X_{n+1} - X_n)^2 + (Y_{n+1} - Y_n)^2$$

โดยที่ (X_{n-1}, Y_{n-1}), (X_n, Y_n) และ (X_{n+1}, Y_{n+1}) เป็น ตำแหน่ง Co-ordinate ของจุดเปลี่ยนแนว 3 จุดที่ถัดกันไป

ในกรณีที่ θ หาได้จาก สมการ(4.2) มีค่าเกิน 90° ค่า θ จะถูกลบด้วย 180° เสอม

ข้อมูลของมุมและระยะทางเหล่านี้จะถูกนำมาจัดเรียงสลับกันไป โดยมุมที่อยู่ถัดๆ ไป จะมีค่าแบบสะสมไปเรื่อยๆ ตัวอย่างของรหัสลูกโซ่แบบมุม-ระยะทางของรูปที่ 4.7 จะเป็น $\sigma_1, \theta_1, \sigma_2, (\theta_1+\theta_2), \sigma_3, (\theta_1+\theta_2+\theta_3), \ldots\ldots, \sigma_8, (\theta_1+\theta_2+\ldots+\theta_8)$ รหัสลูกโซ่นี้จะเป็นแบบ shift around เพื่อใช้ในการทำสหสัมพันธ์มิติเดียว รหัสลูกโซ่แบบมุม-ระยะทางนี้สามารถนำมาพล็อตเป็นกราฟได้ดังรูปที่ 4.8



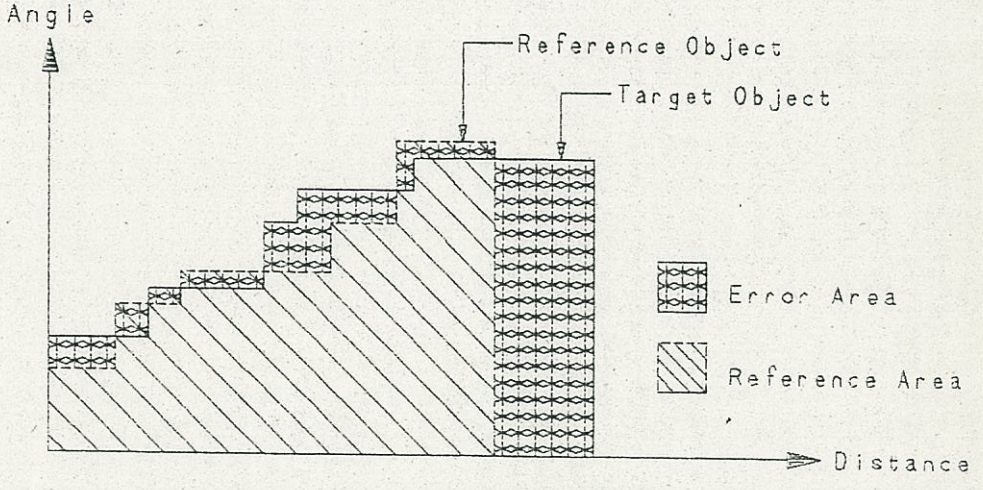
เมื่อ σ = ระยะทาง θ = มุม

รูปที่ 4.8 กราฟของรหัสลูกโซ่แบบมุม-ระยะทาง

4.3 การหาสหสัมพันธ์มิติเดียว (หรือการเปรียบเทียบข้อมูลของวัตถุ)

ในการหาสหสัมพันธ์มิติเดียวทำได้โดยการเปรียบเทียบกราฟรหัสลูกโซ่ของวัตถุใด ๆ กับกราฟรหัสลูกโซ่อ้างอิง จากนั้นก็คำนวณหาเปอร์เซนต์ของพื้นที่ error ระหว่างกราฟทั้งสองเทียบกันพื้นที่ของกราฟรหัสลูกโซ่ที่สั้นกว่าแล้วใช้ค่าเปอร์เซนต์ของพื้นที่ error นี้มาเป็นตัวตัดสินใจว่าวัตถุที่ตรวจสอบจะเป็นชนิดเดียวกับวัตถุอ้างอิงหรือไม่ ในทางปฏิบัตินั้นเปอร์เซนต์ของพื้นที่ error ของวัตถุชนิดเดียวกันไม่ควรจะเกิน 10 เปอร์เซนต์

สำหรับตัวอย่างของการหาสหสัมพันธ์ สมมุติว่าต้องการตรวจสอบข้อมูลภาพ 2 ภาพว่าเป็นภาพวัตถุชนิดเดียวกันหรือไม่ จะสามารถทำได้โดยการนำข้อมูลภาพวัตถุมาผ่านขบวนการประมวลผลดังที่ได้กล่าวมาตั้งแต่ต้นจนถึงขณะนี้ ซึ่งจะได้ข้อมูลภาพอยู่ในรูปของมุมและระยะทาง จากนั้นนำข้อมูลของมุมและระยะทางมาเขียนเป็นกราฟและนำกราฟทั้งสองมาซ้อนทับกันเพื่อการเปรียบเทียบ (Maching) วิธีการเปรียบเทียบนี้บางครั้งเรียกว่าการหาสหสัมพันธ์มิติเดียว แล้วคำนวณหาพื้นที่ของกราฟที่มีการเลื่อมล้ำกันโดยเปรียบเทียบกับพื้นที่ทั้งหมดของกราฟอ้างอิง เพื่อหาเปอร์เซอร์พื้นที่ error ดังรูปที่ 4.9



% Error = Error Area * 100 / Reference Area

Error Area = พ.ท. เลื่อมล้ำ , Distance = ระยะทาง

Reference Area = พ.ท. อ้างอิง , Angle = มุม

รูปที่ 4.9 แสดงการเปรียบเทียบข้อมูลภาพวัตถุ 2 ชนิด

4.4 ผลลัพธ์ในการตรวจสอบชนิดวัตถุ

จากวิธีการต่าง ๆ ที่กล่าวมาจะถูกนำมาประยุกต์สำหรับการตรวจสอบชนิดของวัตถุ โดยมีขั้นตอนต่าง ๆ ดังนี้

ขั้นตอนที่ 1 ถ่ายภาพข้อมูล แล้วทำการแปลงให้เป็นภาพสีเทา 2 ระดับ ภาพวัตถุที่ถ่าย คือประแจปากตายตัวเล็กดังแสดงในรูปที่ 4.10 ก ซึ่งกำลังจะถูกตรวจสอบ ส่วนรูปที่ 4.10 ข และ รูปที่ 4.10 ค เป็นภาพประแจอ้างอิงที่มีขนาดต่างกัน

ขั้นตอนที่ 2 ตรวจจับหาขอบของวัตถุรูปที่ 4.10 ก รูปที่ 4.10 ข และรูปที่ 4.10 ค ซึ่งแต่ละรูปจะมีขอบของวัตถุดังแสดงในรูปที่ 4.11 ก รูปที่ 4.11 ข และ รูปที่ 4.11 ค ตามลำดับ

ขั้นตอนที่ 3 ทำการปรับแต่งข้อมูลขอบของวัตถุให้ราบเรียบ ก็จะได้รูปขอบวัตถุใหม่เป็นรูปที่ 4.12 ก รูปที่ 4.12 ข และรูปที่ 4.12 ค ซึ่งสอดคล้องกับรูปที่ 4.11 ก รูปที่ 4.11 ข และรูปที่ 4.11 ค

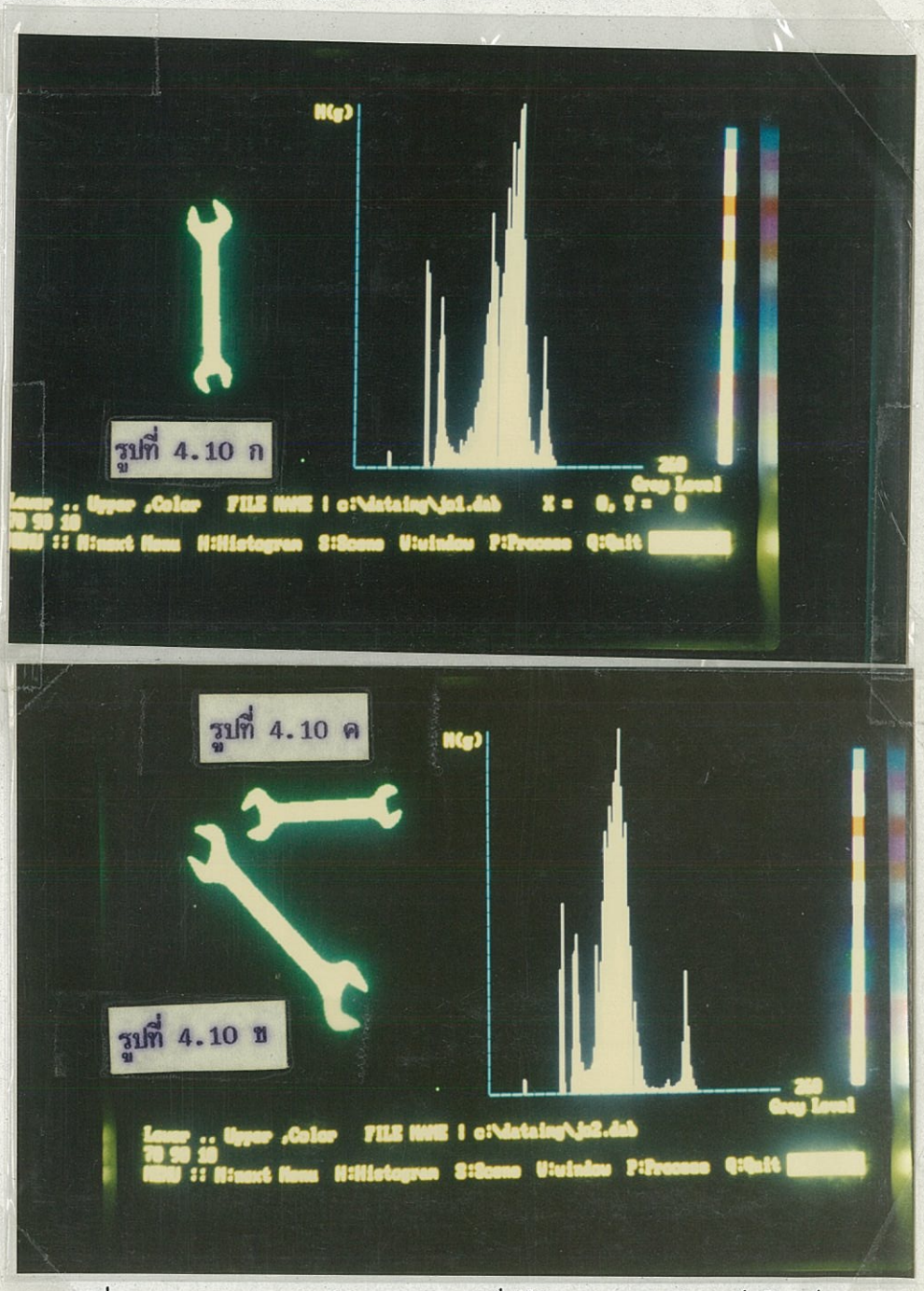
ขั้นตอนที่ 4 ตรวจสอบหาจุดเปลี่ยนแนว เพื่อนำมาสร้างกราฟของรหัสลูกโซ่แบบมุม-ระยะทาง

ขั้นตอนที่ 5 ทำการหาสหสัมพันธ์มิติเดียว ระหว่างกราฟของวัตถุของรูปที่ 4.12 ก ซึ่งเป็นกราฟของวัตถุที่กำลังตรวจสอบ กับกราฟวัตถุรูปที่ 4.12 ข และรูปที่ 4.12 ค ซึ่งเป็นกราฟของวัตถุอ้างอิง เพื่อตรวจสอบว่าจะเป็นวัตถุชนิดใด ซึ่งในรูปที่ 4.13 ก เป็นการเปรียบเทียบระหว่างรหัสลูกโซ่ของรูปที่ 4.12 ก กับ รูปที่ 4.12 ข ส่วนกราฟ รูปที่ 4.13 ข เป็นการเปรียบเทียบระหว่างรหัสลูกโซ่ของรูปที่ 4.12 ก กับ 4.12 ค ในที่สุดจะได้ตารางของสหสัมพันธ์มิติเดียวที่ให้เปอร์เซนต์ของพื้นที่ error ต่ำสุด ดังตารางที่ 4.2

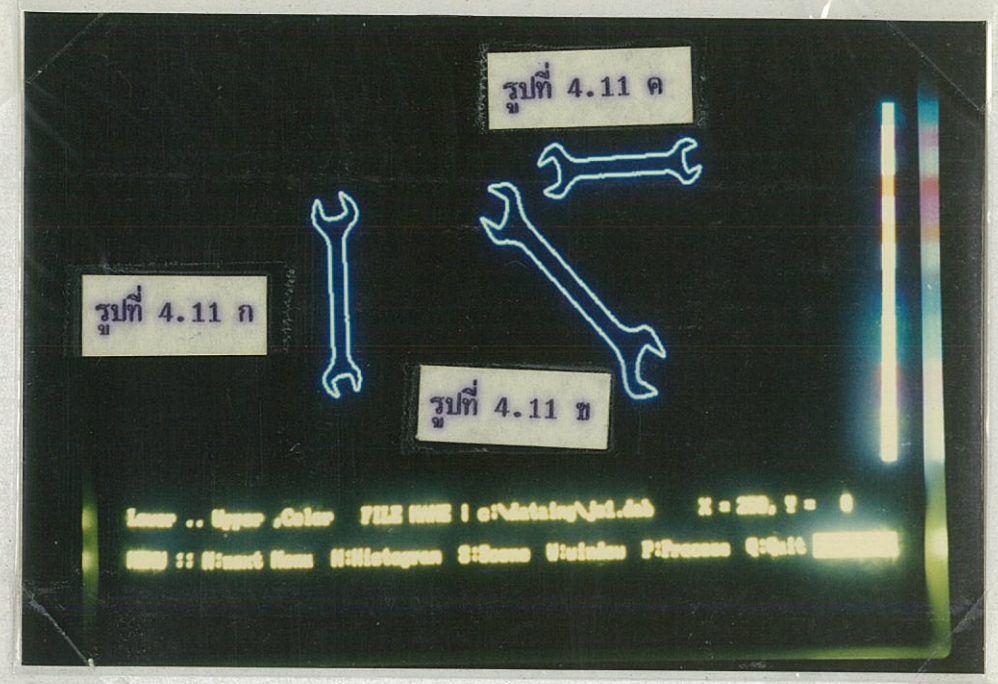
ตารางที่ 4.2 แสดง % พื้นที่ error จากการเปรียบเทียบเพื่อตรวจสอบชนิดของวัตถุ

วัตถุใด ๆ	วัตถุอ้างอิง	% ของพื้นที่ error
รูปที่ 4.10 ก	รูปที่ 4.10 ข	76.82
รูปที่ 4.10 ก	รูปที่ 4.10 ค	7.39

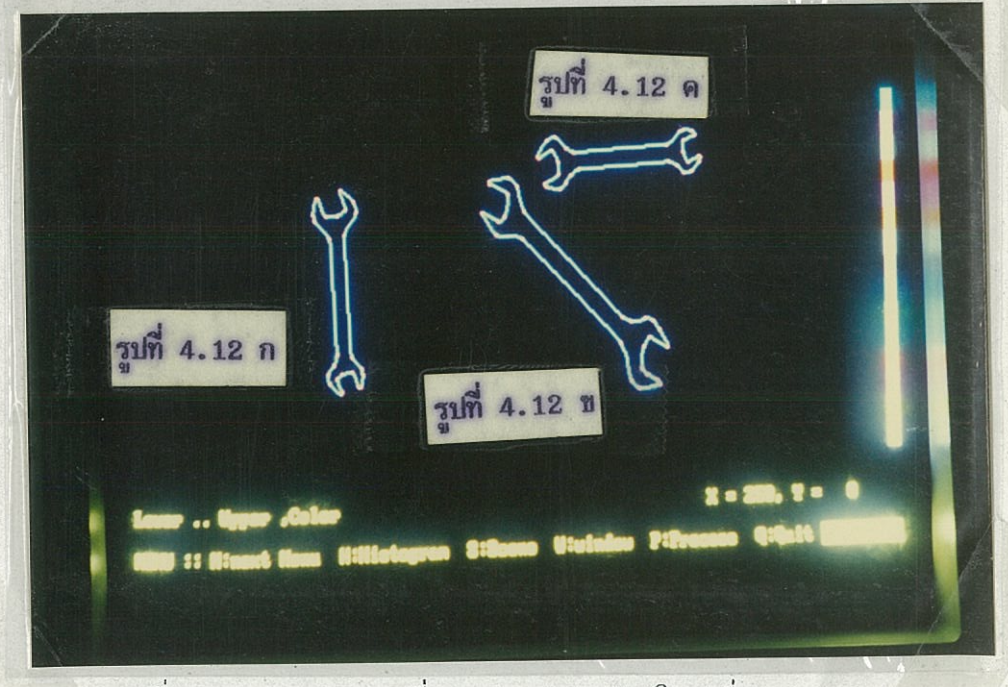
ดังนั้นจึงสรุปได้ว่าวัตถุของรูปที่ 4.10 ก จะเป็นชนิดเดียวกับวัตถุของรูปที่ 4.10 ค ทั้งนี้เพราะเปอร์เซนต์ของพื้นที่ error ต่ำกว่า 10 % ที่ตั้งไว้



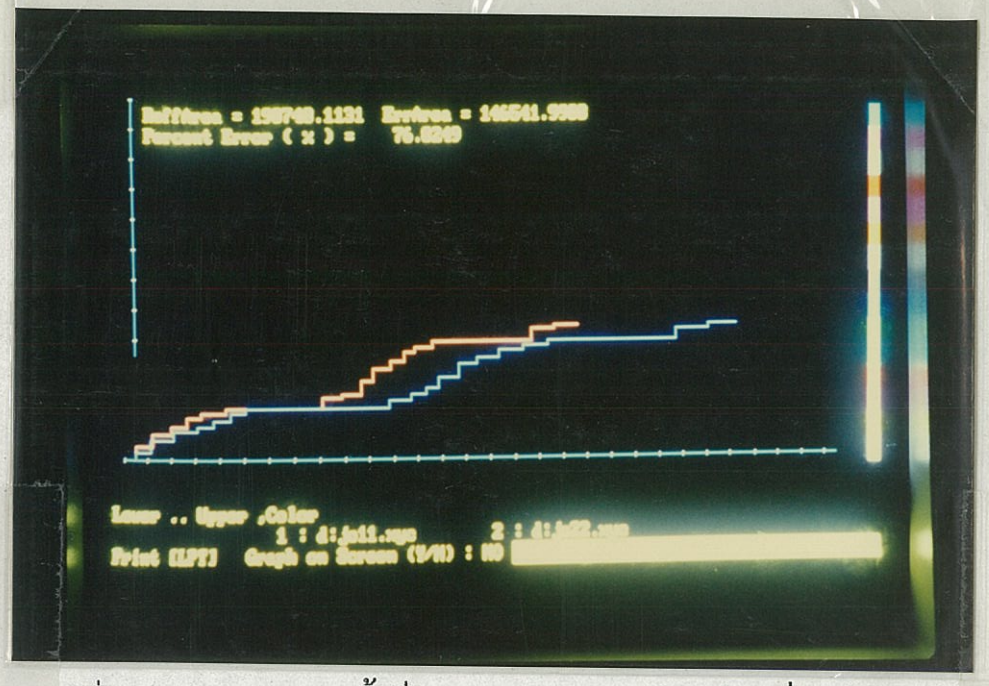
รูปที่ 4.10 แสดงภาพสีเทาสองระดับซึ่งได้จากการตัดค่าเธร์ดโฮลด์ในฮิตโตแกรมทางซ้ายมือของรูป



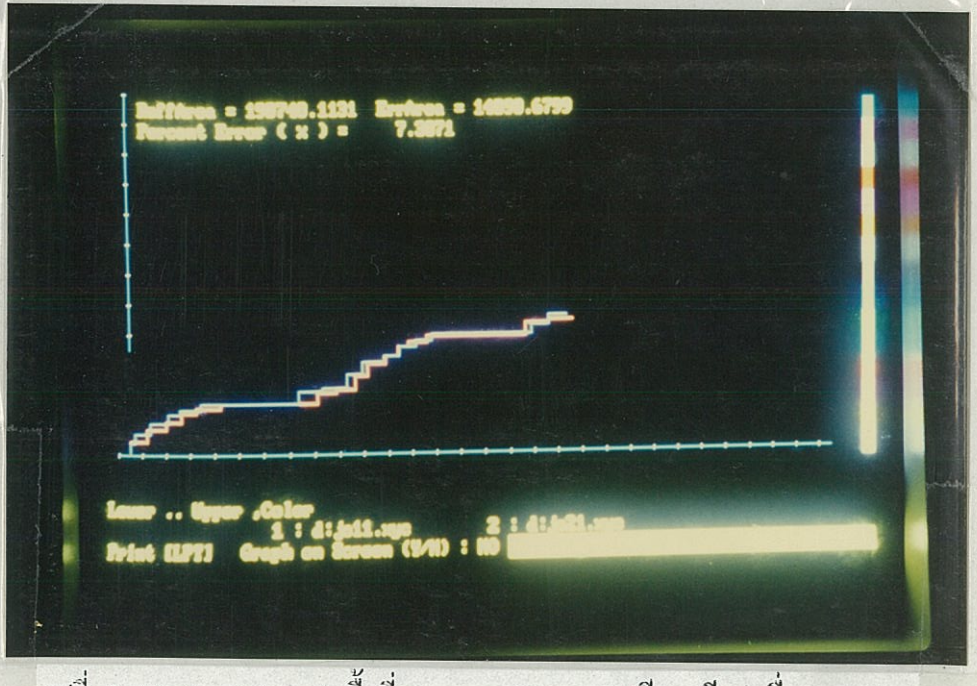
รูปที่ 4.11 แสดงขอบของวัตถุในรูปที่ 4.10



รูปที่ 4.12 แสดงจุดเปลี่ยนแนวขอบของวัตถุในรูปที่ 4.11



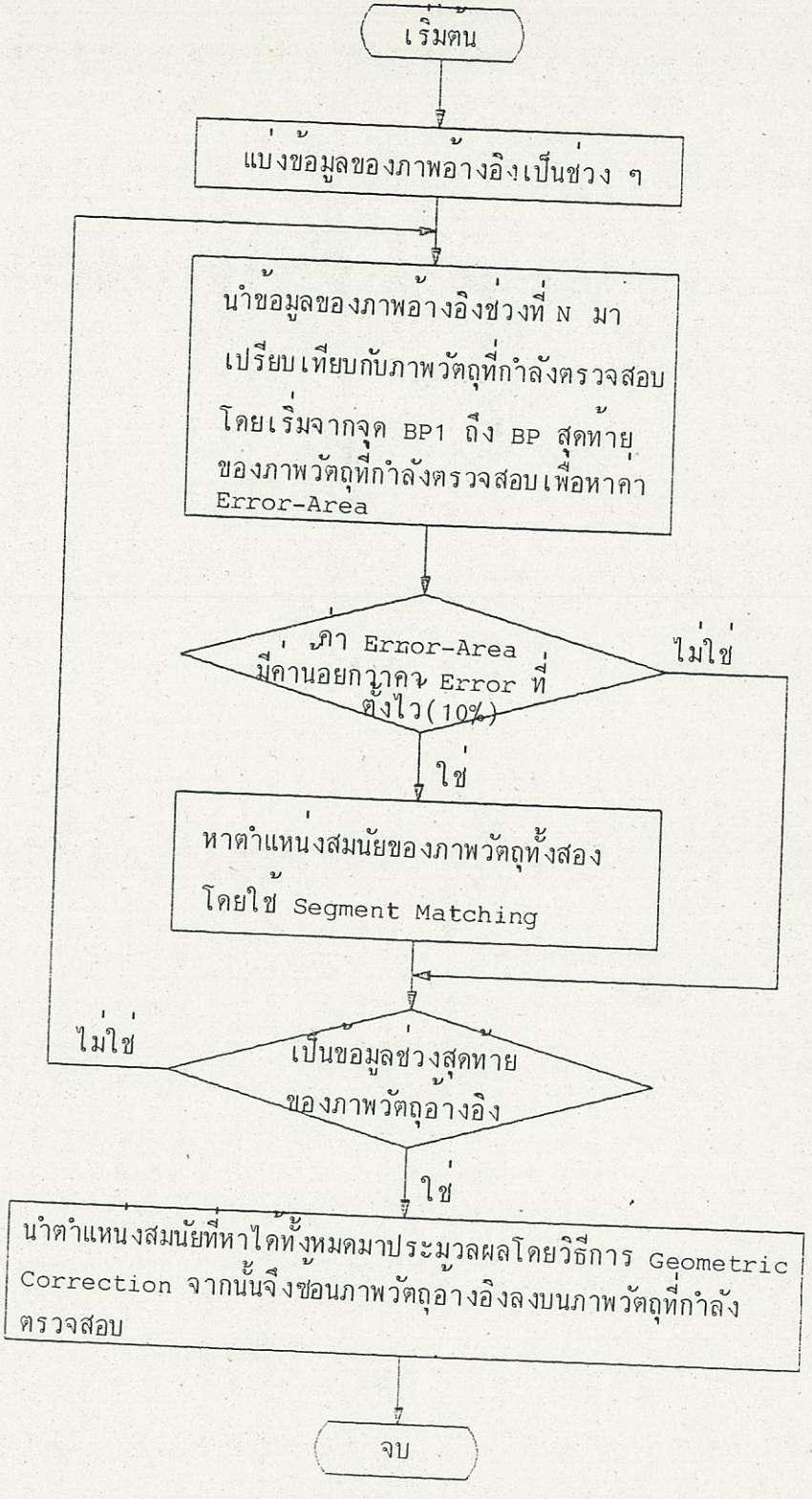
รูปที่ 4.13 ก แสดง % พื้นที่ error จากการเปรียบเทียบเพื่อตรวจสอบชนิดของวัตถุในรูปที่ 4.10 ก กับ รูปที่ 4.10 ข



รูปที่ 4.13 ข แสดง % พื้นที่ error จากการเปรียบเทียบเพื่อตรวจสอบชนิดของวัตถุในรูปที่ 4.10 ก กับ รูปที่ 4.10 ค

บทที่ 5
การหาวัตถุที่มีบางส่วนถูกบดบัง
(Occluded Objects Correlation)

สำหรับกรณีของภาพวัตถุที่กำลังจะตรวจสอบนั้น เป็นภาพวัตถุที่มีการซ้อนทับกันของวัตถุมากกว่า 1 ชิ้น และต่างชนิดกันเราสามารถใช้เทคนิคของบทที่ 4 มาปรับปรุงให้ใช้กับการตรวจสอบภาพวัตถุที่มีการซ้อนทับกัน หรือถูกบดบังได้ โดยเราจะแบ่งฐานข้อมูลของภาพวัตถุอ้างอิงออกเป็นช่วงๆ แล้วนำข้อมูลแต่ละช่วงมาทำการเปรียบเทียบ (Matching) เพื่อหาค่าพื้นที่เลื่อมล้ำต่ำสุดในแต่ละช่วงมาเปรียบเทียบกับค่าที่ตั้งไว้ ถ้าหากค่าพื้นที่เลื่อมล้ำต่ำกว่าค่าที่ตั้งไว้ก็จะนำช่วงนั้น ๆ ของข้อมูลภาพวัตถุอ้างอิง กับภาพวัตถุที่กำลังตรวจสอบมาค้นหาตำแหน่ง (coordinate) ที่สมนัยกันระหว่างภาพวัตถุทั้งสอง ตำแหน่งที่สมนัยกันนี้จะถูกนำมาประมวลผลโดยวิธีการ Geometric Correction [11] เพื่อใช้ในการวางภาพวัตถุอ้างอิงลงบนภาพวัตถุที่กำลังตรวจสอบ สำหรับลักษณะการเปรียบเทียบเพื่อจะหาพื้นที่เลื่อมล้ำนี้ จะใช้วิธีการของหัวข้อ 4.3 มาปรับให้เหมาะสมกับการเปรียบเทียบแบบช่วง ๆ (Intersection Matching) และที่กล่าวมาทั้งหมดในข้างต้นนี้สามารถเขียนเป็นผังแสดงลำดับขั้นการดำเนินการของบทนี้ได้ตามผังที่ 5.1 ดังนี้



ผังที่ 5.1 ขั้นตอนการหาวัตถุที่มีบางส่วนถูกบดบัง

5.1 การตรวจสอบหาตำแหน่งสมนัยระหว่างภาพวัตถุอ้างอิงกับภาพวัตถุที่กำลังตรวจสอบ

จากการหาจุดเปลี่ยนแนว (BP) ในภาพวัตถุ นั้นเราจะได้ตำแหน่ง Coordinate ของจุด BP ทั้งหมดบนภาพวัตถุ และนำจุด BP ที่ได้มาแปลงให้อยู่ในรูปของมุมและระยะทาง โดยข้อมูลดังกล่าวนำมาจัดเป็นตารางที่ 5.1 (Array ของ Record) ดังนี้

ตารางที่ 5.1 แสดงข้อมูลของภาพวัตถุ

จุด BP	Coordinate		Code Object	
	X	Y	ระยะทาง	มุมสะสม
1	30	5	15.57	45
2	19	16	26	135
3	19	42	22	225
4	41	42	5	288.43
5	41	37	11.18	360
6	31	32	14.14	405
7	41	22	6	450
8	41	16	15.57	540

สำหรับการตัดข้อมูลออกเป็นช่วง ๆ จะคำนวณจากจำนวนจุด BP โดยแต่ละช่วงจะมีจุด BP อยู่ประมาณ 30 % ของจำนวนจุด BP ทั้งหมด จากตารางข้างบนนี้เราสามารถแบ่งเป็นช่วง ๆ โดยใช้สมการ (5.1)

จำนวนจุด BP = จำนวนจุด BP ทั้งหมด x 30 % .....(5.1)

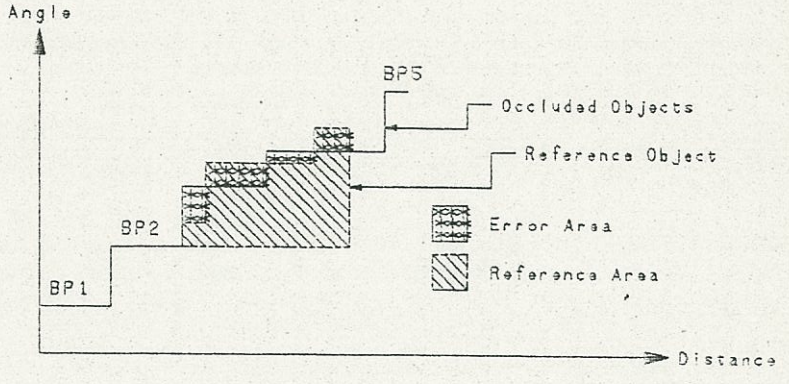
จากตารางข้อมูลของภาพวัตถุอ้างอิงจะมีจำนวนจุด BP อยู่ในช่วงหนึ่ง ๆ เป็นแทนค่า = 8 x 30 / 100 = 2.4

≈ 2 จุด BP ต่อหนึ่งช่วง

ภาพวัตถุอ้างอิงจึงถูกแบ่งออกเป็น 4 ช่วง โดยที่ช่วงหนึ่ง ๆ จะมีจุด BP อยู่ 2 จุด ซึ่งแต่ละช่วงจะถูกนำไปเปรียบเทียบ (Intersection Matching) แบบเป็นช่วง ๆ กับวัตถุที่กำลังตรวจสอบจนหมดรอบรูปวัตถุ

5.1.1 การเปรียบเทียบข้อมูลแบบช่วง ๆ (Intersection Matching)

การเปรียบเทียบหาพื้นที่เลื่อมล้ำจะยังคงใช้วิธีการของหัวข้อ 4.3 แต่ในการหาค่า Error Area จะถูกดำเนินการหาในลักษณะใหม่ เพื่อให้เหมาะสมกับการเปรียบเทียบเป็นช่วง ๆ ซึ่งวิธีดำเนินการมีดังนี้คือ กราฟของวัตถุอ้างอิงจะถูกนำไปเปรียบเทียบกับกราฟของวัตถุที่กำลังตรวจสอบ(ดังแสดงรูป 5.1) โดยเริ่มจากจุด BP ที่ 1 เรื่อยไปจนกระทั่งถึงจุด BP สุดท้ายของวัตถุที่กำลังตรวจสอบ ซึ่งในช่วงหนึ่งๆ ของภาพวัตถุอ้างอิงที่ทำการเปรียบเทียบกับภาพวัตถุที่กำลังตรวจสอบนั้นจะได้ค่า Error Area ที่ต่ำที่สุด และถ้าหากค่า Error Area ที่หาได้นี้ยังมีค่าต่ำกว่า % Area Error ที่ตั้งไว้ (น้อยกว่า 10 %) อีก นั้นก็หมายความว่า ขอบของวัตถุอ้างอิง กับขอบของวัตถุที่กำลังตรวจสอบนั้น มีโอกาสสูงที่จะเป็นขอบของวัตถุเดียวกัน แต่ยังไม่สามารถยืนยันได้ จำเป็นจะต้องมีการตรวจสอบอีกครั้งโดยพิจารณาเป็นส่วนๆ ซึ่งในที่นี้หมายถึงการพิจารณาแบบจุดต่อจุดหรือ Segment Matching อีกครั้ง



% Error = Error Area * 100 / Reference Area

Error Area = พ.ท. เลื่อมล้ำ , Distance = ระยะทาง

Reference Area = พ.ท. อ้างอิง , Angle = มุม

รูปที่ 5.1 แสดงการคำนวณหาพื้นที่เลื่อมล้ำ (Error Area) ในกรณีภาพวัตถุที่กำลังตรวจสอบถูกบดบัง

5.1.2 การเปรียบเทียบข้อมูลแบบจุดต่อจุด (Segment Matching)

สาเหตุที่ต้องนำเอาวิธีการเปรียบเทียบข้อมูลแบบจุดต่อจุดมีอยู่ 2 ประการคือ

1. ต้องการเพิ่มความมั่นใจว่าช่วงของขอบวัตถุอ้างอิงที่ใช้ในการเปรียบเทียบกับช่วงของขอบวัตถุที่กำลังตรวจสอบเป็นขอบของวัตถุชนิดเดียวกัน

2. ต้องการหาตำแหน่งสมนัยของภาพวัตถุทั้งสอง

การเปรียบเทียบนี้จะถูกดำเนินการก็ต่อเมื่อได้ผ่านขั้นตอนการเปรียบเทียบข้อมูลแบบทีละช่วง (Intersection Matching) เสียก่อน และพบว่ามี Error Area ต่ำกว่าที่กำหนดไว้ การเปรียบเทียบข้อมูลแบบจุดต่อจุด (Segment Matching) จะมีลักษณะการดำเนินการดังนี้คือ เปรียบเทียบระยะทางและมุมระหว่างภาพวัตถุอ้างอิง กับ ภาพวัตถุที่กำลังตรวจสอบ โดยหาค่าแตกต่างของระยะทาง (% Error Length) ซึ่งหาได้จากสมการ (5.2)

$$\% \text{ Error Length} = |\sigma_R - \sigma_O| \times 100 / \sigma_R \quad \% \qquad \ldots\ldots(5.2)$$

โดยที่ σ_R : คือระยะทางของวัตถุอ้างอิง

σ_O : คือระยะทางของวัตถุที่กำลังตรวจสอบ

และค่าแตกต่างของมุม (% Error Angle) ซึ่งหาได้จากสมการ (5.3)

$$\% \text{ Error Angle} = |\theta_R - \theta_O| \times 100 / \theta_R \quad \% \qquad \ldots\ldots(5.3)$$

โดยที่ θ_R : คือมุมของวัตถุอ้างอิง

θ_O : คือมุมของวัตถุที่กำลังตรวจสอบ

ตัวอย่าง การเปรียบเทียบข้อมูลแบบจุดต่อจุดของวัตถุอ้างอิง กับ วัตถุที่กำลังตรวจสอบพิจารณาได้จากตารางของวัตถุอ้างอิง กับวัตถุที่กำลังตรวจสอบดังต่อไปนี้

วัตถุอ้างอิง

จุด BP	ตำแหน่ง BP		รหัสลูกโซ่	
	X	Y	σ	θ
9	19	16	26	90
10	19	42	22	180
11	41	42	5	243.43

วัตถุที่กำลังตรวจสอบ

จุด BP	ตำแหน่ง BP		รหัสลูกโซ่	
	U	V	σ	θ
20	22	37	25.32	88.78
21	47	41	22.36	177.78
22	51	19	5.1	241.21

จากการเปรียบเทียบข้อมูลทีละช่วง (intersection Matching) พบว่าจุด BP ที่ 9, 10 และ 11 ของภาพวัตถุอ้างอิง กับจุด BP ที่ 20, 21 และ 22 ของภาพวัตถุที่กำลังตรวจสอบมี % Error Area น้อยกว่า 10% ดังนั้นเราจะนำจุด BP เหล่านี้ของวัตถุทั้งสองมาเปรียบเทียบกันแบบจุดต่อจุด (Segment Matching) โดยใช้ระยะทางและมุมตรงจุด BP ที่ 9, 10 และ 11 ในตารางของภาพวัตถุอ้างอิงกับจุด BP ที่ 20, 21 และ 22 ในตารางของภาพกำลังตรวจสอบมาจัดเป็นคู่ๆ ตามลำดับแล้วนำข้อมูลทีละคู่มาคำนวณหาค่าความแตกต่างของความยาวและมุมระหว่างข้อมูลทั้งสอง ถ้าหากค่าความแตกต่างของความยาวและมุมมีค่าน้อยกว่าที่ตั้งไว้ ก็จะเก็บตำแหน่ง Coordinate (X,Y) ของภาพวัตถุอ้างอิงกับตำแหน่ง Coordinate (U,V) ของภาพวัตถุที่กำลังตรวจสอบไว้เพื่อนำไปประมวลผลทาง Geometric Correction ต่อไป

ตัวอย่างการหาค่าความแตกต่างของระยะทางและมุม

ที่จุด BP_{Reff-9} กับ BP_{obj-20}

% Error Length = |26 - 25.32| x 100 / 26

= 2.62 %

% Error Angle = |90 - 88.78| x 100 / 90

= 1.36 %

ค่า % Error Length และ % Error Angle มีค่าน้อยกว่า 10 %

เพราะฉะนั้น ตำแหน่งที่สมนัยกัน ของภาพวัตถุทั้งสองคือ

19, 16 <—> 22, 37 .....(*)

ที่จุด $BP_{Reff-10}$ กับ BP_{obj-21}

% Error Length = |22 - 22.36| x 100 / 22

= 1.64 %

% Error Angle = |180 - 177.78| x 100 / 180

= 1.23 %

เพราะฉะนั้น ตำแหน่งที่สมนัยกับของภาพวัตถุทั้งสองคือ

19, 42 <—> 47, 41 .....(*)

ในทำนองเดียวกันที่จุด $BP_{Reff-11}$ กับ BP_{obj-22}

% Error Length = |5 - 5.1| x 100 / 5

= 2 %

% Error Angle = |243.43 - 241.21| x 100 / 243.43

= 0.91 %

เพราะฉะนั้น ตำแหน่งที่สมนัยกันของภาพวัตถุทั้งสองคือ

41, 42 <—> 51, 19 .....(*)

5.2 การซ้อนภาพวัตถุอ้างอิงลงบนภาพวัตถุที่กำลังตรวจสอบโดยใช้ Geometric Correction

เราสามารถดูภาพวัตถุที่ถูกบดบังว่ามีรูปร่างเป็นอย่างไรได้โดยการนำตำแหน่งสมนัยของภาพวัตถุอ้างอิงกับภาพวัตถุที่กำลังตรวจสอบ (ซึ่งหาได้จากหัวข้อ 5.1) มาประมวลผลโดยใช้วิธี Geometric Correction ซึ่งหลังจากการประมวลผลด้วยวิธีนี้จะทำให้สามารถเห็นรูปร่างทั้งหมดของภาพวัตถุที่ถูกบดบังได้ โดยการซ้อนภาพวัตถุอ้างอิงลงบนตำแหน่งของภาพวัตถุที่ถูกบดบัง

5.2.1 วิธีการ Geometric Correction

วิธีการ Geometric Correction นี้จะเป็นการแปลงตำแหน่ง Coordinate บนระนาบ X, Y ให้ไปอยู่ตำแหน่ง Coordinate บนระนาบ U, V หรือกล่าวได้อีกนัยหนึ่งเพื่อให้เข้าใจง่ายขึ้นคือ การนำภาพสองภาพที่เหมือนกันแต่มีตำแหน่งและขนาดที่ต่างกันมาซ้อนทับให้พอดีกัน เพื่อให้มองดูเป็นภาพเดียวกันนั้นเอง

โดยหลักการ Geometric Correction แล้วจะเป็นการแปลงระบบ Coordinate 2 ระบบ ระหว่างระบบ Coordinate ของภาพข้อมูลที่ 1 กับระบบ Coordinate ของภาพข้อมูลที่ 2 โดยระบบ Coordinate สามารถเขียนให้มีความเกี่ยวพันกันได้ดังสมการที่ (5.4) และสมการ (5.5)

$$X = a_o + a_1U + a_2V \qquad \ldots\ldots (5.4)$$

$$Y = b_o + b_1U + b_2V \qquad \ldots\ldots (5.5)$$

ซึ่งเราสามารถหาค่าสัมประสิทธิ์ a_o, a_1, a_2 ได้โดยใช้ least square Method จากสมการ (5.4) เขียนใหม่ได้เป็นสมการ (5.6)

$$X_i = a_o + a_1U_i + a_2V_i \qquad \ldots\ldots (5.6)$$

โดยมี Normal Equation เป็นสมการ (5.7)

$$\begin{bmatrix} n & \Sigma U_i & \Sigma V_i \\ \Sigma U_i & \Sigma U_i^2 & \Sigma U_iV_i \\ \Sigma V_i & \Sigma U_iV_i & \Sigma V_i^2 \end{bmatrix} \begin{bmatrix} a_o \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \Sigma X_i \\ \Sigma U_iX_i \\ \Sigma V_iX_i \end{bmatrix} \qquad \ldots (5.7)$$

เอาค่า n หารตลอดทั้งสองข้างและเขียนใหม่เป็นสมการ (5.8)

$$\begin{bmatrix} 1 & \bar{U} & \bar{V} \\ \bar{U} & \overline{U^2} & \overline{UV} \\ \bar{V} & \overline{UV} & \overline{V^2} \end{bmatrix} \begin{bmatrix} a_o \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \bar{X} \\ \overline{UX} \\ \overline{VX} \end{bmatrix} \quad \ldots\ldots (5.8)$$

จากสมการ (5.8) หาค่าสัมประสิทธิ์ a_o, a_1, a_2 ได้จากสมการ (5.9)

$$\begin{bmatrix} a_o \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1 & \bar{U} & \bar{V} \\ \bar{U} & \overline{U^2} & \overline{UV} \\ \bar{V} & \overline{UV} & \overline{V^2} \end{bmatrix}^{-1} \begin{bmatrix} \bar{X} \\ \overline{UX} \\ \overline{VX} \end{bmatrix} \quad \ldots\ldots (5.9)$$

สำหรับ Inverse Matrix สามารถหาได้จากสมการ (5.10)

$$\begin{bmatrix} 1 & p & q \\ p & r & s \\ q & s & t \end{bmatrix}^{-1} = 1 / |S| \begin{bmatrix} rt-s^2 & qs-pt & ps-qr \\ qs-pt & t-q^2 & pq-s \\ ps-qr & pq-s & r-p^2 \end{bmatrix} \quad \ldots\ldots (5.10)$$

โดยที่ $|S| = (rt-s^2) + p(qs-pt) + q(ps-qr)$

ซึ่งจากที่กล่าวมาในข้างต้นนี้ในทำนองเดียวกันเราสามารถหาค่าสัมประสิทธิ์ b_o, b_1 และ b_2 ได้

จากค่าสัมประสิทธิ์ทั้งหมดที่หาได้คือ a_o, a_1, a_2, b_o, b_1 และ b_2 นี้เราสามารถแปลง Coordinate (X,Y) ของภาพวัตถุอ้างอิงให้อยู่บน Coordinate (U,V) ของภาพวัตถุที่ถูกบดบังได้โดยใช้สมการ (5.12)

จากสมการ (5.4) และ (5.5) เขียนใหม่เป็น (5.11)

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} a_o \\ b_o \end{bmatrix} + \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix} \quad \ldots\ldots (5.11)$$

ย้ายข้างสมการ (5.11) จะได้

$$\begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix}^{-1} \begin{bmatrix} X-a_o \\ Y-b_o \end{bmatrix}$$

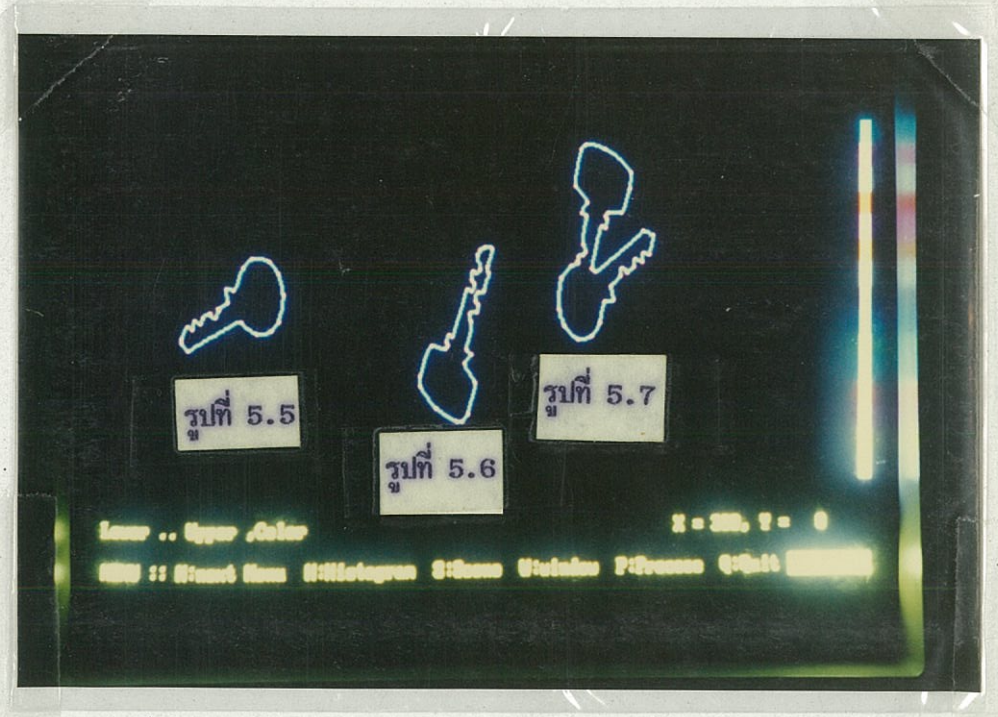
$$= 1 / |S| \begin{bmatrix} b_2 & -a_2 \\ -b_1 & a_1 \end{bmatrix} \begin{bmatrix} X-a_o \\ Y-b_o \end{bmatrix} \quad \ldots\ldots (5.12)$$

โดยที่ $|S| = a_1 . b_2 - a_2 . b_1$

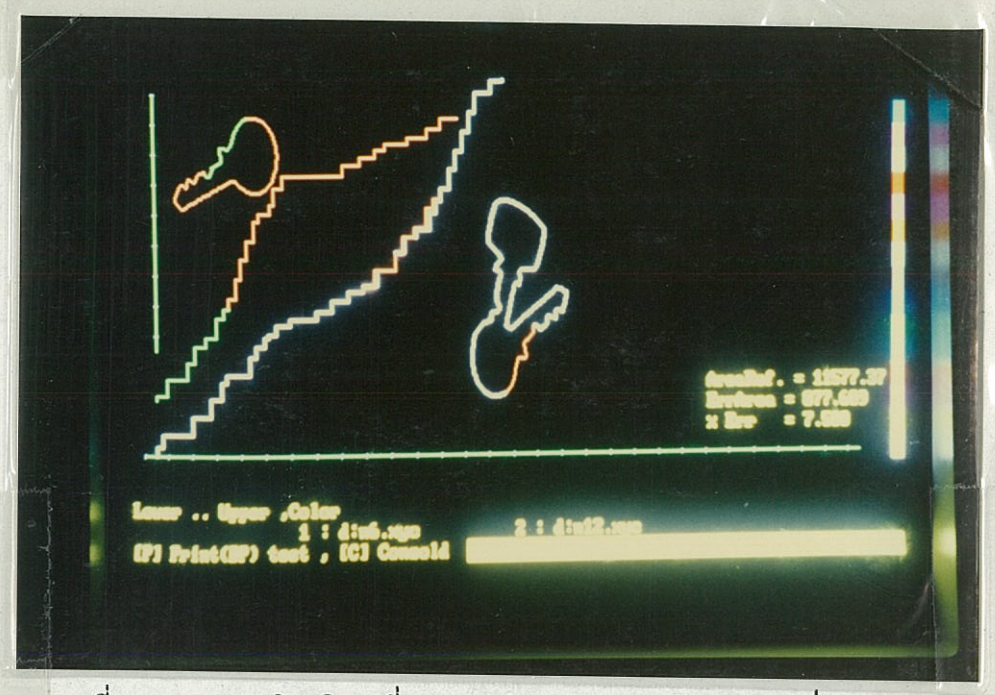
5.3 ผลการทดสอบกับข้อมูลภาพ

จากข้อมูลภาพซึ่งเป็นลูกกุญแจดังแสดงในรูปที่ 5.2 และ 5.3 นั้น เป็นข้อมูลภาพที่เก็บอยู่ในหน่วยความจำของไมโครคอมพิวเตอร์ ส่วนข้อมูลภาพในรูปที่ 5.4 นั้น เป็นข้อมูลภาพของลูกกุญแจสองดอกที่วางซ้อนทับกันอยู่ ซึ่งข้อมูลภายในรูปที่ 5.4 นี้ จะถูกทำการเปรียบเทียบกับลูกกุญแจต่าง ๆ ที่เก็บอยู่ในฐานข้อมูลหน่วยความจำของไมโครคอมพิวเตอร์ สำหรับรูปที่ 5.5 รูปที่ 5.6 และ รูปที่ 5.7 นั้นแสดงขอบของลูกกุญแจ และจุดเปลี่ยนแนวเส้น และในรูปที่ 5.8 และ รูปที่ 5.9 แสดงให้เห็นถึงลักษณะของการเปรียบเทียบแบบช่วง ๆ (Intersection Matching) ระหว่างข้อมูลขอบของวัตถุอ้างอิง กับข้อมูลขอบของวัตถุที่กำลังตรวจสอบ ซึ่งรูปที่ 5.8 นั้น แสดงถึงบริเวณขอบที่สมนัยกันของวัตถุในรูป 5.2 กับ วัตถุในรูป 5.4 ส่วนรูปที่ 5.9 ได้แสดงถึงบริเวณขอบที่สมนัยกันของวัตถุในรูป 5.3 กับ วัตถุในรูป 5.4 และหลังจากทำการหาตำแหน่งสมนัยของภาพวัตถุอ้างอิงกับวัตถุที่กำลังตรวจสอบเสร็จแล้ว จะนำเอาตำแหน่งสมนัยของวัตถุทั้งสองมาประมวลผลด้วยวิธีการ Geometric Correction ก็จะทำให้สามารถเห็นรูปร่างของวัตถุที่ถูกบดบังได้ สำหรับรูปที่ 5.10 และรูปที่ 5.11 นั้นเป็นผลลัพธ์ที่ได้หลังจากการดำเนินการตามวิธีการ Geometric Correction

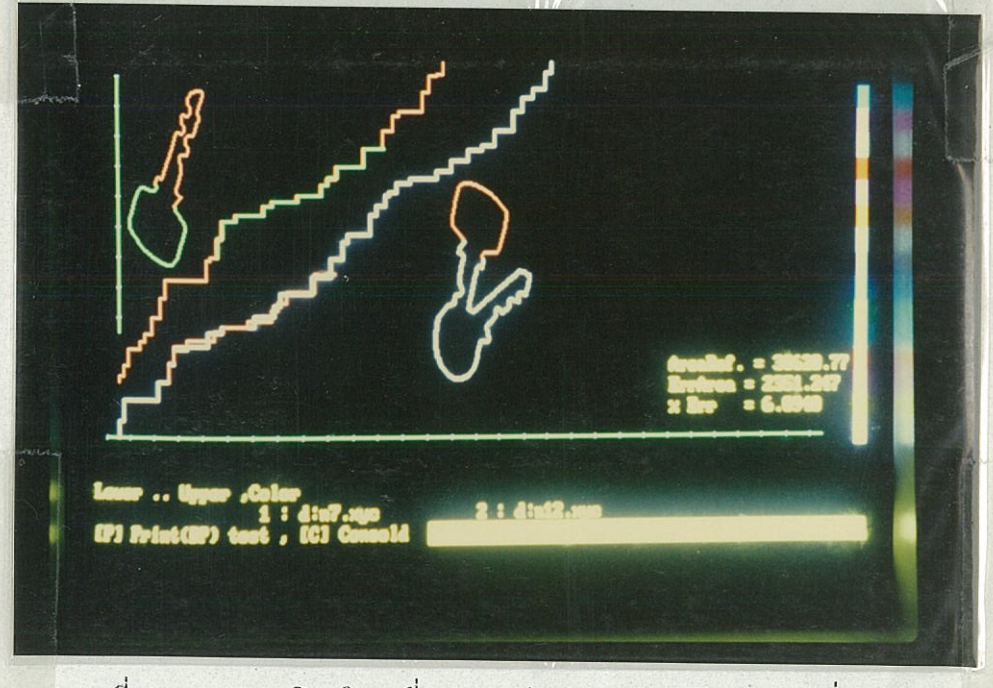
รูปที่ 5.2 5.3 5.4 แสดงภาพสีเทาสองระดับของลูกกุญแจ



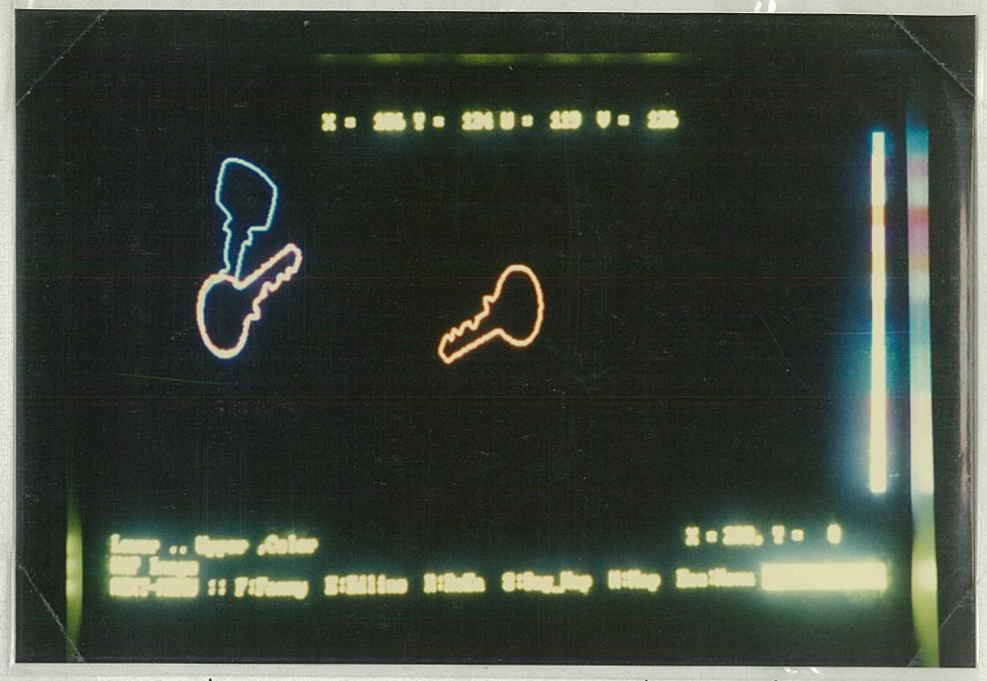
รูปที่ 5.5 5.6 5.7 แสดงขอบและจุดเปลี่ยนแนวเส้นของลูกกุญแจ



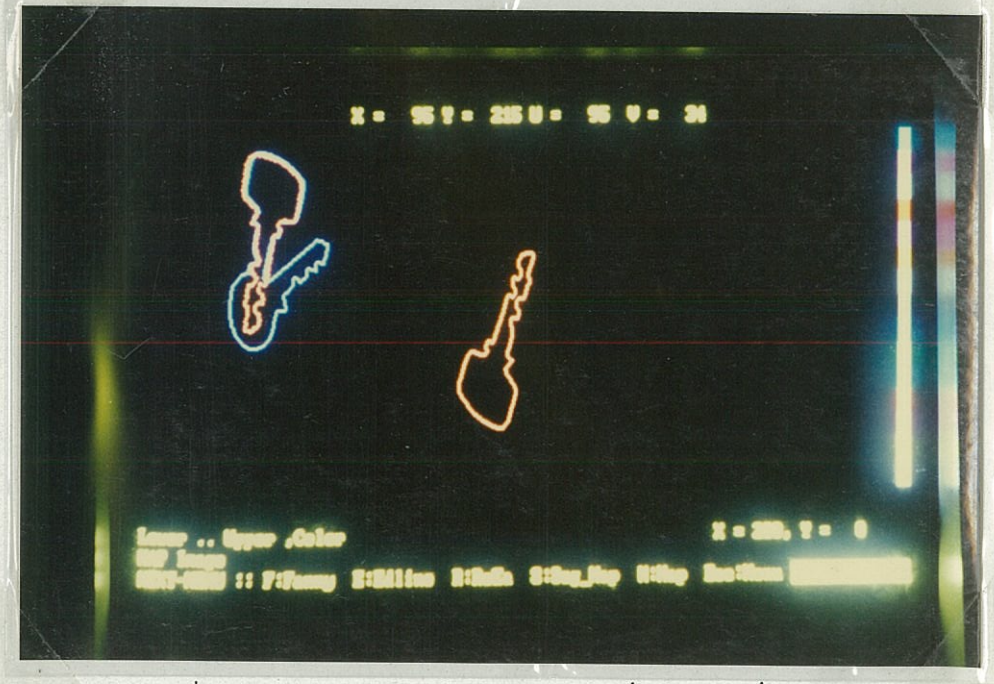
รูปที่ 5.8 แสดงถึงบริเวณที่ตรวจพบว่าขอบของวัตถุอ้างอิง(รูปที่ 5.2) มีความสมนัยกันกับขอบของวัตถุที่กำลังตรวจสอบ(รูปที่ 5.4)



รูปที่ 5.9 แสดงถึงบริเวณที่ตรวจพบว่าขอบของวัตถุอ้างอิง(รูปที่ 5.3) มีความสมนัยกันกับขอบของวัตถุที่กำลังตรวจสอบ(รูปที่ 5.4)



รูปที่ 5.10 แสดงผลการตรวจสอบวัตถุที่ถูกบดบังเมื่อใช้
วัตถุในรูป 5.2 เป็นวัตถุอ้างอิง



รูปที่ 5.11 แสดงผลการตรวจสอบวัตถุที่ถูกบดบังเมื่อใช้
วัตถุในรูป 5.3 เป็นวัตถุอ้างอิง

Program :
Programmer :

```
{ ****************************************************************
  *                                                              *
  *             PROCEDURE  MODULE_ROTATE_OBJECT                  *
  * /*  Begin  Eigenvalue & Eigenvector  then Rotate_Object  */ *
  *                                                              *
  *              CREATE  ( MM DD YY ) 11 10 87                   *
  *                                                              *
  *                        VERSION 2.0                           *
  *                                                              *
  *              BY  CHANCHAI PISITTIVITHAYANON                  *
  *                                                              *
  ****************************************************************}


Overlay procedure Module_Rotate_Object ;
type
  EleMxLyF = (A11,A12,A21,A22) ;  { Element matrix Formular }
  FMatrixT = array[EleMxLyF] of real;
  MatrixTT = array[1..2] of real;
var
  RotaPnT1   : DataPnT;
  RotaScene  : DataPnT;
  FMatrix    : FMatrixT;
  Compensate : MatrixTT;
  Element1,Element2,Element3,Element4 : real;
  Eigenvalue1,Eigenvalue2             : real;
  Eigenvector1,Eigenvector2           : MatrixTT;
  HTop : integer;

procedure Find_Element(RotaPnT : DataPnT);  {Find Element Matrix}
var
  tem : byte;
  NumPix,SumX,SumY,SumXX,SumYY : real;
  RoXX,RoYY,StardX,StardY      : real;
begin
  NumPix := 0;
  SumX   := 0;
  SumY   := 0;
  SumXX  := 0;     { Sum(X^2) }
  SumYY  := 0;     { Sum(Y^2) }
```

```
  Element1 := 0;   { Element Matrix Aii }
  Element2 := 0;   { Element Matrix Aij }
  Element3 := 0;   { Element Matrix Aji }
  Element4 := 0;   { Element Matrix Ajj }
  for Line_Y := 1 to Y_Max do
    for Line_X := 1 to X_Max do
    begin
      tem := RotaPnT^.DataImage[Line_X]^.ArrayP[Line_Y];
      if tem = G8H then
      begin
        NumPix := NumPix + 1;
        SumX   := SumX+Line_X;
        SumY   := SumY+Line_Y;
        SumXX  := SumXX+Sqr(Line_X);
        SumYY  := SumYY+Sqr(Line_Y);
      end;
    end;
  Mean_X := SumX / NumPix ;
  Mean_Y := SumY / NumPix ;
  RoXX := Sqr(SumXX)/NumPix - Sqr(Mean_X) ;
  RoYY := Sqr(SumYY)/NumPix - Sqr(Mean_Y) ;
  StardX := Sqrt(RoXX) * Sqrt(NumPix);
  StardY := Sqrt(RoYY) * Sqrt(NumPix);
  for Line_Y := 1 to Y_Max do
    for Line_X := 1 to X_Max do
    begin
      tem := RotaPnT^.DataImage[Line_X]^.ArrayP[Line_Y];
      if tem = G8H then
      begin
        Element1 := Element1 + Sqr(Line_X-Mean_X);
        Element2 := Element2 + (Line_X-Mean_X) * (Line_Y-Mean_Y);
        Element4 := Element4 + Sqr(Line_Y-Mean_Y);
      end;
    end;
  Element1 := Element1 / NumPix;
  Element2 := Element2 / NumPix;
  Element3 := Element2 ;
  Element4 := Element4 / NumPix;
end; { Find_Element }

procedure Find_EigenValue;
var
  TemX,TemY,TemZ : real;
  A,B,C,D        : real;
begin
  A := Element1;
  B := Element2;
  C := Element3;
  D := Element4;
```

```
  TemX := Sqr(D+A);
  TemY := 4*(A*D - B*C);
  if TemX >= TemY then
  begin
    Eigenvalue1 := ( (D+A) + Sqrt(TemX-TemY) )/2;
    Eigenvalue2 := ( (D+A) - Sqrt(TemX-TemY) )/2;
  end
  else
  begin
    gotoXY(1,23); ClrEol;
    Sound(2000); Delay(300); NoSound;
    write('Compute occupy imaginary ');
  end;
end; { Find_Eigenvalue }

procedure Find_Eigenvector(Lamda : real ; var Eigenvector : MatrixTT);
var
  plusflage1 , plusflage2 : boolean;
  A,B,C,D                 : real;
begin
  A := Element1 - Lamda ;
  B := Element2 ;
  C := Element3 ;
  D := Element4 - Lamda ;
  if (A = 0) AND (B = 0) then
     begin
       A := C ;
       B := D ;
     end;
  if (A = 0) AND (B = 0) then
     begin
       Eigenvector[1] := 1;
       Eigenvector[2] := 0;
     end
  else
     begin
       Eigenvector[1] := 1;
       if B = 0 then
          begin
            Eigenvector[1] := 0;
            Eigenvector[2] := 1;
          end
       else
          begin
            Eigenvector[2] := (A / B) * (-1) ;
          end;
     end;
end; { Find_Eigenvector }
```

```
procedure Edit_Axis(x,y,slop:real );
var
  x1,y1,x2,y2 : integer;
  temX,temY,m,c : real ;
begin
  x1 := round(x); y1 := round(y);
  EGAPlot(x1,y1,0,7);
  m := slop;
  c := y - (m*x);
  x1 := 1;
  y1 := trunc(m*x1 + c);
  if (y1 < 1) OR (y1 > Y_Max) then
    begin
      y1 := 1;
      x1 := trunc( (y1 - c ) / m );
      if (x1 < 1) OR (x1 > X_Max) then
        begin
          x1 := X_Max;
          y1 := trunc( m*x1 + c );
          if (y1 < 1) OR (y1 > Y_Max) then
            begin
              y1 := Y_Max;
              x1 := trunc( (y1 - c ) / m );
            end;
        end;
    end;
  y2 := Y_Max;
  x2 := trunc( (y2 - c ) / m );
  if (x2 < 1) OR (x2 > X_Max) then
    begin
      x2 := X_Max;
      y2 := trunc( m*x2 + c );
      if (y2 < 1) OR (y2 > Y_Max) then
        begin
          y2 := 1;
          x2 := trunc( (y2 - c ) / m );
          if (x2 < 1) OR (x2 > X_Max) then
            begin
              x2 := 1;
              y2 := trunc( m*x2 + c );
            end;
        end;
    end;
  EGADraw (X1;Y1,X2,Y2,0,14);
end; { Edit_Axis }

procedure Eigen_Process(RotaPnT : DataPnT) ;
var
```

```
  slopEigen1 : real;
  slopEigen2 : real;
procedure Yes_No_Edit_Axis;
begin { Yes_No_Edit_Axis }
  repeat
    gotoXY(1,23); ClrEol;
    write('Display Axis on picture (Y/N) ');
    read (Kbd,ch);
  until UpCase(ch) IN ['Y','N'];
  if UpCase(ch) = 'Y' then
    begin
      if Eigenvalue1 ) Eigenvalue2 then
        Edit_Axis(Mean_X,Mean_Y,SlopEigen1)
      else
        Edit_Axis(Mean_X,Mean_Y,SlopEigen2);
    end;
end;  { Yes_No_Edit_Axis }
begin { Eigen_Process }
  gotoXY(1,23); ClrEol;
  textcolor(4); write('Compute ! ');
  textbackground(0);   write('Please wait... ');
  Find_Element(RotaPnT) ;
  Find_Eigenvalue;
  Find_Eigenvector(Eigenvalue1,Eigenvector1);
  if Eigenvector1[1] ( 0.00000001 then
    SlopEigen1 := 0.00000001
  else
    SlopEigen1 := Eigenvector1[2] / Eigenvector1[1] ;
  Find_Eigenvector(Eigenvalue2,Eigenvector2);
  if Eigenvector2[1] ( 0.00000001 then
    SlopEigen2 := 0.00000001
  else
    SlopEigen2 := Eigenvector2[2] / Eigenvector2[1] ;
  Yes_No_Edit_Axis;
  sound(2000); delay(500); nosound;
end;  { Eigen_Process }


{ ******************************************
        END;   Eigenvalue & Eigenvector
        BEGIN; Rotate_Object
               Version 1.1A
  ****************************************** }


procedure Initial_Rota;
begin
  Wait_Message;
  New(RotaPnT1) ;
```

```
  New(RotaScene);
  RotaPnT1^.NextDa  := nil  ;
  RotaScene^.NextDa := nil ;
  for Line_X := 1 to X_Max do
  begin
    New(RotaPnT1^.DataImage[Line_X]);
    New(RotaScene^.DataImage[Line_X]);
    RotaPnT1^.DataImage[Line_X]^.NextAr  := nil ;
    RotaScene^.DataImage[Line_X]^.NextAr := nil ;
    for Line_Y := 1 to Y_Max do
    begin
      RotaPnT1^.DataImage[Line_X]^.ArrayP[Line_Y]  := GBL;
      RotaScene^.DataImage[Line_X]^.ArrayP[Line_Y] := GBL;
    end;
  end;
end; { Initial_Rota }

procedure Take_Formular(zeta:real);
begin
  FMatrix[A11] := cos(zeta);
  FMatrix[A12] := sin(zeta);
  FMatrix[A21] := -sin(zeta);
  FMatrix[A22] := FMatrix[A11];
end;  { Formular_Matrix }

procedure Product_Matrix(var  Result : MatrixTT) ;
var
  temResult  : MatrixTT;
begin
  temResult[1] := FMatrix[A11] * Result[1] + FMatrix[A21] * Result[2] ;
  temResult[2] := FMatrix[A12] * Result[1] + FMatrix[A22] * Result[2] ;
  Result[1]    := temResult[1] ;
  Result[2]    := temResult[2] ;
end; { Product_Matrix }

procedure Sub_Matrix (var dataXY : MatrixTT);
begin { Sub_Matrix }
  dataXY[1] := dataXY[1] - Mean_X ;
  dataXY[2] := dataXY[2] - Mean_Y ;
end; { Sub_Matrix }

procedure Add_Matrix (var dataXY : MatrixTT) ;
begin { Add_Matrix }
  dataXY[1] := dataXY[1] + Mean_X ;
  dataXY[2] := dataXY[2] + Mean_Y ;
end;  { Add_Matrix }

procedure Find_Angle (var Alpha : real );
var
```

```
  tem : real ;
begin { Find_Angle }
  if Eigenvalue1 >= Eigenvalue2 then
    begin
      if Eigenvector1[1] = 0 then
        tem := Pi/2
      else
        tem   := ArcTan(Eigenvector1[2]/Eigenvector1[1]);
    end
  else        { Eigenvalue2 > Eigenvalue1 }
    begin
      if Eigenvector2[1] = 0 then
        tem := pi/2
      else
        tem   := ArcTan(Eigenvector2[2]/Eigenvector2[1]);
    end;
  Alpha := tem ;
end; { Find_Angle }

function Find_Angle_Rota (Alpha1,Alpha2 : real ) : real ;
var
  tem1,tem2,tem : real;
  P_Falg1,P_Falg2 : boolean ;
begin { Find_Angle_Rota }
  if Alpha1 < 0 then
    P_Falg1 := false
  else
    P_Falg1 := true ;
  if Alpha2 < 0 then
    P_Falg2 := false
  else
    P_Falg2 := true ;
  if P_Falg1 AND P_Falg2 then
    tem := Alpha1 - Alpha2
  else
      if NOT(P_Falg1) AND NOT(P_Falg2) then
        tem := -( ABS(Alpha1) - ABS(Alpha2) )
      else
        if P_Falg1 AND NOT(P_Falg2) then
          tem := Alpha1 - (pi + Alpha2)
        else
           if NOT(P_Falg1) AND P_Falg2 then
             tem := ( (pi + Alpha1) - Alpha2 ) ;
    Find_Angle_Rota := tem ;
  gotoXY(1,23); ClrEol;
  write('Rotate Angle = ',( (tem)*180/pi ):3:2 ); read(kbd,ch);
end; { Find_Angle_Rota }

procedure Rota_Data( RotaPnt : DataPnT ; Alpha : real);
```

```
var
  tem             : byte;
  ResultX,ResultY : integer;
  TemMa           : MatrixTT;
begin
  Take_Formular(Alpha);
  for Line_Y := 1 to Y_Max do
  begin
    for Line_X := 1 to X_Max do
    begin
      tem := RotaPnT^.DataImage[Line_X]^.ArrayP[Line_Y];
      if tem = G8H then
        begin
          TemMa[1] := Line_X;   TemMa[2] := Line_Y;
          Sub_Matrix(TemMa);
          Product_Matrix(TemMa);
          Add_Matrix(TemMa);
          ResultX := round(TemMa[1]);
          ResultY := round(TemMa[2]);
          if RotaScene^.DataImage[ResultX]^.ArrayP[ResultY] = G8L then
             RotaScene^.DataImage[ResultX]^.ArrayP[ResultY] := tem
          else
            begin
              ResultX := round(TemMa[1]);
              ResultY := round(TemMa[2]);
              RotaScene^.DataImage[ResultX]^.ArrayP[ResultY] := tem;
            end; { if }
        end; { if }
    end; { for }
  end; { for }
end; { Rota_Data }

procedure DisplayResult;
var
  tem : byte;
begin
  for Line_Y := 1 to Y_Max do
    for Line_X := 1 to X_Max do
    begin
      tem := RotaScene^.DataImage[Line_X]^.ArrayP[Line_Y] ;
      if tem = G8H then
        begin
          EGAPlot(Line_X+PosiI,Line_Y+PosiJ,0,12);
        end;
    end;
end; { DisplayResult }

procedure Pre_Eigen_Rota_Process(var Alpha : real );
var
```

```
  Alpha1,Alpha2 : real ;
begin { Pre_Eigen_Rota }
  gotoXY(1,22); write('Refference Scene ');
  Read_Data_File(RotaPnT1);
  Eigen_Process(RotaPnT1);
  Find_Angle(Alpha1);
  gotoXY(1,22); write('Taget Scene      ');
  Read_Data_File(RotaPnT1);
  gotoXY(1,22); write('                 ');
  Eigen_Process(RotaPnT1);
  Find_Angle(Alpha2);
  Alpha := Find_Angle_Rota(Alpha1,Alpha2);
end; { Pre_Eigen_Rota }


begin  { Module_Rotate_Object }
  Mark(HTop);
  gotoXY(1,23); ClrEol;
  write('Compute ! ');
  Initial_Rota;
  Pre_Eigen_Rota_Process(Zeta);
  Rota_Data(RotaPnT1,Zeta);
  DisplayResult;
  sound(2000); delay(500); nosound;
  Yes_No_SavePicture(Rotascene);
  Release(HTop);
end; { Module_Rotate_Object }

{ ************************************************

            End  Rotate_Object

  ************************************************ }
```

Program :
Programmer :

```
{ ****************************************************************
  *                                                              *
  *             PROCEDURE  EDGE FOLLING BY WINDO                 *
  *                                                              *
  *               CREATE  ( MM DD YY ) 6 12 87                   *
  *                                                              *
  *                       VERSION 1.2C                           *
  *                                                              *
  *             BY    CHANCHAI PISITTIVITHAYANON                 *
  *                                                              *
  ****************************************************************}


Overlay procedure Contour_Following( BvisionV : DataPnT );
Label
      End_Procedure;
Type
      CodeT     = 0..8 ;
      DataRecP  = ^DataRecPT ;
      DataRecPT = Record
                     DataI1 : byte;
                     DataJ1 : byte;
                     DataC1 : CodeT;
                     NextPn : DataRecP;
                  End;

Var
     ch        : char ;
     StartPI,StartPJ : integer ;
     NextEI,NextEJ   : byte ;
     TemX,TemY       : byte ;
     ChainCode : CodeT ;
     NumE      : integer ;
     ResEdge   : DataRecP;
     FirResEdge ,NewResEdge, LastResEdge  : DataRecP;
     ClockWise : boolean;
     Error     : boolean;

procedure Initial_Edge_Detection;
begin  { Initial_Edge_Detection }
```

```
  New(Result8);
  Result8^.NextDa := nil ;
  for Line_X := 1 to X_Max do
  begin
    New(Result8^.DataImage[Line_X]);
    Result8^.DataImage[Line_X]^.NextAr := nil ;
    for Line_Y := 1 to Y_Max do
    begin
      Result8^.DataImage[Line_X]^.ArrayP[Line_Y] := GBL;
    end;
  end;
end;  { Initial_Edge_Detection }

Procedure SaveDataE ;
Var
     Created  : boolean;
     Filename : string[50];
     DataFile : DataT;
     DataRec  : DataRecT;
     ch       : char;

  procedure  WriteDataRec ;
  begin { WriteDataRec }
    NewResEdge := FirResEdge;
    while NOT (NewResEdge = nil) do
    begin
      DataRec.DataI1 := NewResEdge^.DataI1;
      DataRec.DataJ1 := NewResEdge^.DataJ1;
      DataRec.DataC1 := NewResEdge^.DataC1;
      write(DataFile,DataRec);
      NewResEdge  := NewResEdge^.NextPn;
    end; { while }
  end;  { WriteDataRec }

Begin  { SaveDataE }
  gotoXY(1,23); ClrEol;
  write ('Input file name for SAVE dataedge ...');
  read (Filename);
  if NOT(Filename = '') then
    begin
      Assign(DataFile,Filename);
      {$I-} Reset(DataFile); {$I+}
      Created := IOresult = 0;
      if NOT Created then
        begin
          rewrite(DataFile);
          WriteDataRec;
          close(DataFile);
        end
```

```
      else  { #1 }
        begin
          gotoXY(1,23); ClrEol;
          sound(2000); delay(500); nosound;
          write(Filename,' Exist !    ENTER : OverWrite  ,  Esc : Quit ');
          repeat
            read(kbd,ch);
          until (ch = #$0D) OR (ch = #27);
          if ch = #$0D then
            begin
              rewrite(DataFile);
              WriteDataRec;
              close(DataFile);
            end;
        end; { #1 }
    end; { if }
  sound(2000); delay(500); nosound;
end;  { SaveDataE }


Procedure DisplayDataLPT ;
var a : integer;
begin
  writeln (Lst,'                              Format Data Edge');
  writeln(Lst);writeln(Lst);
  writeln (Lst,'                             X , Y : Chain Code');
  writeln(Lst);writeln(Lst);writeln(Lst);writeln(Lst);
  NewResEdge := FirResEdge;
  a := 1;
  while NOT (NewResEdge = nil) do
  begin
    write(Lst,NewResEdge^.DataI1:3,',',NewResEdge^.DataJ1:3,':'
             ,NewResEdge^.DataC1:1,'  ');
    NewResEdge := NewResEdge^.NextPn;
    if (a mod 6) = 0 then
      writeln(Lst);
    a := a + 1 ;
  end;
    writeln(Lst); writeln(Lst);
end; { DisplayDataLPT }

Procedure DisplayDataCON ;
var a : integer;
begin
  gotoXY(1,23); ClrEol;
  NewResEdge := FirResEdge ;
  a := 1 ;
  while NOT (NewResEdge = nil) do
  begin
```

```
    write (NewResEdge^.DataI1:3,',',NewResEdge^.DataJ1:3,':',
           NewResEdge^.DataC1,'  ');
    NewResEdge := NewResEdge^.NextPn ;
    if (a mod 7) = 0 then
      begin
        read(kbd,ch);
        gotoXY(1,23);  ClrEol;
      end;
      a := a + 1 ;
   end;
   read(kbd,ch);
end; { DisplayDataCON }

Procedure Pre_Prompt ;
begin
  gotoXY(1,23); ClrEOL ;
  write ('   Press  any key  to  Continue   ');
  read (Kbd,ch) ;
end;  { Pre_Prompt }


procedure Find_FPosi (var xx,yy : integer);
var
  TemXX,TemYY : integer;
  TemColor : byte;
  ch : char;
begin { Find_FPosi }
  gotoXY(1,23); ClrEol;
  write('Press Arrow key for Move Point :: Enter to Continue ');
  while NOT(ch = Enter) do
  begin
    gotoXY(60,21); write('X = ',XX:3,', Y = ',YY:3);
    TemColor := EGAGetdot(xx,yy,0);
    TemXX := xx ;
    TemYY := yy ;
    EGAPlot (xx,yy,0,15);
    read(kbd,ch);
    cursor(xx,yy,ch);
    EGAPlot (TemXX,TemYY,0,TemColor);
  end; { while }
  Repeat
    xx := xx + 1;
  until ((GBH = BvisionV^.DataImage[xx]^.ArrayP[yy])
        OR ((X_Max-1) < xx));
  while ((GBL = BvisionV^.DataImage[xx]^.ArrayP[yy])
        AND ((Y_Max-1) > yy)) do
  begin
    yy := yy + 1;
    xx := 1;
```

```
    Repeat
      xx := xx + 1;
    until ((G8H = BvisionV^.DataImage[xx]^.ArrayP[yy])
          OR ((X_Max-1) < xx));
  end;
  if (Y_Max-1) <= yy then
    begin
      gotoXY(1,23); ClrEol;
      write('Error ... Contain scene EMPTY ');
      sound(2000); delay(500); nosound;
      delay(500);
      Error := true;
    end;
  sound(2000); delay(500); nosound;
end; { Find_FPosi }

procedure Keep_Edge (PosI,PosJ : byte ; CCode : CodeT );
var
  F_DataRec ,N_DataRec ,L_DataRec : DataRecP ;
begin
    New (NewResEdge);
    NewResEdge^.DataI1 := PosI;
    NewResEdge^.DataJ1 := PosJ;
    NewResEdge^.DataC1 := CCode;
    if FirResEdge = nil then
      FirResEdge := NewResEdge
    else
      LastResEdge^.NextPn := NewResEdge;
    LastResEdge := NewResEdge;
    LastResEdge^.NextPn := nil ;
end; { Keep_Edge }

procedure YesNo_ClockWise;
begin
  gotoXY(1,23); ClrEol;
  write('Set CLOCK WSIE  ( Y / N ), YES ');
  read(Kbd,ch);
  if Upcase(ch) = 'N' then
    ClockWise := false
  else
    ClockWise := true ;
end; { YesNo_ClockWise }

Procedure Next_edge (Var PosI , PosJ : byte ; Var CCode : codeT);
Type
  GrayL  = G8L..G8H;
  windoT = array [CodeT] of GrayL;
Var
  direction : CodeT; { code }
```

```
  windo : windoT;
  ProtecLoop :  byte;
Begin
      { 0.................I }
      { : +----+----+----+    }
      { : ¦ W3 ¦ W2 ¦ W1 ¦    }
      { : +----+----+----+    }
      { : ¦ W4 ¦    ¦ W0 ¦    }
      { : +----+----+----+    }
      { : ¦ W5 ¦ W6 ¦ W7 ¦    }
      { : +----+----+----+    }
      { J                     }

  Windo[0] := BvisionV^.DataImage[PosI+1]^.ArrayP[PosJ  ];
  Windo[1] := BvisionV^.DataImage[PosI+1]^.ArrayP[PosJ-1];
  Windo[2] := BvisionV^.DataImage[PosI  ]^.ArrayP[PosJ-1];
  Windo[3] := BvisionV^.DataImage[PosI-1]^.ArrayP[PosJ-1];
  Windo[4] := BvisionV^.DataImage[PosI-1]^.ArrayP[PosJ  ];
  Windo[5] := BvisionV^.DataImage[PosI-1]^.ArrayP[PosJ+1];
  Windo[6] := BvisionV^.DataImage[PosI  ]^.ArrayP[PosJ+1];
  Windo[7] := BvisionV^.DataImage[PosI+1]^.ArrayP[PosJ+1];

  direction := ccode;
  if ClockWise then
    begin
      case  direction  of
        0 : direction := 7 {5};
        1 : direction  := 7 {6};
        2 : direction  := 1 {7};
        3 : direction  := 1 {0};
        4 : direction  := 3 {1};
        5 : direction  := 3 {2};
        6 : direction  := 5 {3};
        7 : direction  := 5 {4};
      end; { case }
      ProtecLoop := 0;
      while (windo [direction] <> GBH) OR (ProtecLoop < 8) do
      begin
        ProtecLoop := ProtecLoop + 1;
        direction := direction + 1;
        if direction > 7 then
          direction := 0;
      end; { while }
    end
  else {  NOT(ClockWise).  }
    begin
      case  direction  of
        0 : direction := 1 ;
        1 : direction := 3 ;
```

```
        2 : direction := 3 ;
        3 : direction := 5 ;
        4 : direction := 5 ;
        5 : direction := 7 ;
        6 : direction := 7 ;
        7 : direction := 1 ;
       end; { case }
       ProtecLoop := 0;
       while (windo [direction] <> GBH) OR (ProtecLoop < 8) do
       begin
         ProtecLoop := ProtecLoop + 1;
         direction := direction - 1;
         if direction = 255 then
           direction := 7;
       end; { while }
     end;
   ccode := direction;
   Case  direction  OF   {* transform to standas window *}
     0  : begin PosI := PosI + 1 ; PosJ := PosJ     ; end;
     1  : begin PosI := PosI + 1 ; PosJ := PosJ - 1 ; end;
     2  : begin PosI := PosI     ; PosJ := PosJ - 1 ; end;
     3  : begin PosI := PosI - 1 ; PosJ := PosJ - 1 ; end;
     4  : begin PosI := PosI - 1 ; PosJ := PosJ     ; end;
     5  : begin PosI := PosI - 1 ; PosJ := PosJ + 1 ; end;
     6  : begin PosI := PosI     ; PosJ := PosJ + 1 ; end;
     7  : begin PosI := PosI + 1 ; PosJ := PosJ + 1 ; end;
   end; { CASE }
end;  { Next_edge }

Procedure DisplayRes;
Var
     ii  : integer;
     jj  : integer;
     tem : byte;
begin
     for jj := 1 to  Y_Max do
     begin
         for ii := 1 to X_Max do
         begin
             tem := ResultB^.DataImage[ii]^.ArrayP[jj];
             if tem = GBH then
                 EGAPlot(ii+PosiI,jj+PosiJ,0,1);
         end;
     end;
end;  { DisplayRes }


Begin  { Contour_Following }
  Mark(HeapTop);
```

```
  Error := false;
  Initial_Edge_Detection;
  YesNo_ClockWise;
  FirResEdge := nil;
  StartPI := 1 ; StartPJ := 1;   {* set value of initial *}
  Find_FPosi (StartPI,StartPJ);  {* find first point of binary_vision *}
  if Error then GOTO End_Procedure;
  ChainCode := 7 ;     {* set initial for Loopscan in
                          the PROCEDURE NEXT_EDGE (use only 7) *}
  NextEI := StartPI; NextEJ := StartPJ;
  Next_Edge(NextEI,NextEJ,ChainCode);
  Keep_Edge(StartPI,StartPJ,ChainCode);{* keep data numver &
                                       position into memory *}
  Result8^.DataImage[StartPI]^.ArrayP[StartPJ] :=
  BvisionV^.DataImage[StartPI]^.ArrayP[StartPJ] ;
  while NOT( (StartPI = NextEI) AND (StartPJ = NextEJ) ) do
  begin
     TemX := NextEI ; TemY := NextEJ ;
     Next_edge(NextEI,NextEJ,ChainCode);  {* find edge *}
     Keep_Edge(TemX,TemY,ChainCode); {* keep data number&position
                                      into memory           *}
     Result8^.DataImage[TemX]^.ArrayP[TemY] :=
     BvisionV^.DataImage[TemX]^.ArrayP[TemY] ;
  end;
  DisplayRes;
  repeat
       gotoXY(1,23); ClrEol;
       write('Selet..Alphabet for Display Chain_code    C : Console  P : LPT  Q : Quit ');
       read(Kbd,ch);
       if UpCase(ch) = 'P' then
           DisplayDataLPT;
       if UpCase(ch) = 'C' then
           DisplayDataCON;
  until UpCase(ch) in ['P','C','Q'];
  repeat
       gotoXY(1,23); ClrEol;
       write ('Save Dataedge ( Y/N ) '); read(Kbd,ch) ;
       if Upcase(ch) = 'Y' Then
          begin
           SavedataE ;
          end;
  until  UpCase(ch) in ['Y','N'];
  Yes_No_SavePicture(Result8);
  End_Procedure :
  Release(HeapTop);
End; { Contour_Following }
```

Program :
Programmer :

```
{ ****************************************************************
  *            PROCEDURE  SMOOTHING BINARY IMAGES                *
  *            /*   BINARY IMAGES IS PASSED    */                *
  *            /*       THE PROCEDURE          */                *
  *                                                              *
  *              CREATE  ( MM DD YY ) 8 28 87                    *
  *                                                              *
  *                       VERSION 1.0                            *
  *                                                              *
  *            BY    CHANCHAI PISITTIVITHAYANON                  *
  *                                                              *
  ****************************************************************}

Overlay procedure  Smoothing_Binary_Images(SceneBi : DataPnT );

Type
     LabelT = (A,B,C,D,E,F,G,H,P);
     GrayL  = G8L..G8H;
     windoT = array [LabelT] of boolean;
Var
     TemSceneBi : DataPnT;
     TemSmooth  : DataPnT;
     Windo  : windoT;
     result : boolean;

procedure Initial_Smoothing;
begin
  New(TemSceneBi);
  TemSceneBi^.NextDa := nil ;
  New(TemSmooth);
  TemSmooth^.NextDa := nil ;
  for Line_X := 0 to X_Max+1 do
  begin
    New(TemSceneBi^.DataImage[Line_X]);
    TemSceneBi^.DataImage[Line_X]^.NextAr := nil ;
    New(TemSmooth^.DataImage[Line_X]);
    TemSmooth^.DataImage[Line_X]^.NextAr := nil ;
  end;

  for Line_X := 0 to X_Max+1 do
```

```
  begin
    TemSceneBi^.DataImage[Line_X]^.ArrayP[  1   ] := GBL ;
    TemSceneBi^.DataImage[Line_X]^.ArrayP[Y_Max ] := GBL ;
    TemSceneBi^.DataImage[  1   ]^.ArrayP[Line_X] := GBL ;
    TemSceneBi^.DataImage[X_Max ]^.ArrayP[Line_X] := GBL ;
  end;
  for Line_Y := 1 to Y_Max do
    for Line_X := 1 to X_Max do
    begin
      TemSceneBi^.DataImage[Line_X]^.ArrayP[Line_Y] :=
      SceneBi^.DataImage[Line_X]^.ArrayP[Line_Y];
      TemSmooth^.DataImage[Line_X]^.ArrayP[Line_Y]  :=
      TemSceneBi^.DataImage[Line_X]^.ArrayP[Line_Y];
    end;
end; { Initial_Smoothing }

procedure TransfromTrueFalse(PosI,PosJ:integer);

      { O.................I }
      { : +---+---+---+    }
      { : ¦ A ¦ B ¦ C ¦    }
      { : +---+---+---+    }
      { : ¦ D ¦ P ¦ E ¦    }
      { : +---+---+---+    }
      { : ¦ F ¦ G ¦ H ¦    }
      { : +---+---+---+    }
      { J                  }

begin { TransfromTrueFalse }

    if TemSmooth^.DataImage[PosI-1]^.ArrayP[PosJ-1] = GBH then
       Windo[A] := true
    else
       Windo[A] := false;
    if TemSmooth^.DataImage[PosI  ]^.ArrayP[PosJ-1] = GBH then
       Windo[B] := true
    else
       Windo[B] := false;
    if TemSmooth^.DataImage[PosI+1]^.ArrayP[PosJ-1] = GBH then
       Windo[C] := true
    else
       Windo[C] := false;
    if TemSmooth^.DataImage[PosI-1]^.ArrayP[PosJ  ] = GBH then
       Windo[D] := true
    else
       Windo[D] := false;
    if TemSmooth^.DataImage[PosI+1]^.ArrayP[PosJ  ] = GBH then
       Windo[E] := true
    else
```

```
      Windo[E] := false;
    if TemSmooth^.DataImage[PosI-1]^.ArrayP[PosJ+1] = G8H then
      Windo[F] := true
    else
      Windo[F] := false;
    if TemSmooth^.DataImage[PosI  ]^.ArrayP[PosJ+1] = G8H then
      Windo[G] := true
    else
      Windo[G] := false;
    if TemSmooth^.DataImage[PosI+1]^.ArrayP[PosJ+1] = G8H then
      Windo[H] := true
    else
      Windo[H] := false;
    if TemSmooth^.DataImage[PosI  ]^.ArrayP[PosJ  ] = G8H then
      Windo[P] := true
    else
      Windo[P] := false;
end;  { TransfromTrueFalse }

procedure Algorithum_Step1;
{ Step_1 .. fills in small (one pixel) holes in otherwise dark areas }
begin
  for Line_Y := 1 to Y_Max do
    for Line_X := 1 to X_Max do
    begin
      TransfromTrueFalse(Line_X,Line_Y);
      result := Windo[P]
                OR ( Windo[B] AND Windo[G] AND ( Windo[D] OR Windo[E] ) )
                OR ( Windo[D] AND Windo[E] AND ( Windo[B] OR Windo[G] ) );
      if result then
        TemSceneBi^.DataImage[Line_X]^.ArrayP[Line_Y] := G8H
      else
        TemSceneBi^.DataImage[Line_X]^.ArrayP[Line_Y] := G8L;
    end;
end; { Algorithum_Step1 }

procedure Algorithum_Step2;
{ Step_2 .. fills in small notches in straightedge segments  }
begin
  for Line_Y := 1 to Y_Max do
    for Line_X := 1 to X_Max do
    begin
      TransfromTrueFalse(Line_X,Line_Y);
      result := Windo[P] AND
                ( ( Windo[A] OR Windo[B] OR Windo[D] ) AND
                  ( Windo[E] OR Windo[G] OR Windo[H] ) OR
                  ( Windo[B] OR Windo[C] OR Windo[E] ) AND
                  ( Windo[D] OR Windo[F] OR Windo[G] ) );
      if result then
```

```
        TemSceneBi^.DataImage[Line_X]^.ArrayP[Line_Y] := GBH
      else
        TemSceneBi^.DataImage[Line_X]^.ArrayP[Line_Y] := GBL;
    end;
end; { Algorithum_Step2 }

procedure Algorithum_Step3;
{ Step_3 .. eliminates isolated 1's }
begin
  for Line_Y := 1 to Y_Max do
    for Line_X := 1 to X_Max do
    begin
      TransfromTrueFalse(Line_X,Line_Y);
      result := NOT(Windo[P]) AND
                ( Windo[D] AND Windo[F] AND Windo[G] ) AND
                NOT( Windo[A] OR Windo[B] OR Windo[C] OR Windo[E] OR Windo[H] )
                OR Windo[P] ;
      if result then
        TemSceneBi^.DataImage[Line_X]^.ArrayP[Line_Y] := GBH
      else
        TemSceneBi^.DataImage[Line_X]^.ArrayP[Line_Y] := GBL;
    end;
end; { Algorithum_Step3 }

procedure Algorithum_Step4;
{ Step_4 .. eliminates small bumps along straightedge segments }
begin
  for Line_Y := 1 to Y_Max do
    for Line_X := 1 to X_Max do
    begin
      TransfromTrueFalse(Line_X,Line_Y);
      result := NOT(Windo[P]) AND
                ( Windo[A] AND Windo[B] AND Windo[D] ) AND
                NOT( Windo[C] OR Windo[E] OR Windo[F] OR Windo[G] OR Windo[H] )
                OR Windo[P] ;
     if result then
        TemSceneBi^.DataImage[Line_X]^.ArrayP[Line_Y] := GBH
      else
        TemSceneBi^.DataImage[Line_X]^.ArrayP[Line_Y] := GBL;
    end;
end; { Algorithum_Step4 }

procedure Algorithum_Step5;
{ Step_5 .. replaces missing corner points }
begin
  for Line_Y := 1 to Y_Max do
    for Line_X := 1 to X_Max do
    begin
      TransfromTrueFalse(Line_X,Line_Y);
```

```
      result := NOT(Windo[P]) AND
                ( Windo[E] AND Windo[G] AND Windo[H] ) AND
                NOT( Windo[A] OR Windo[B] OR Windo[C] OR Windo[D] OR Windo[F] )
                OR Windo[P] ;
      if result then
         TemSceneBi^.DataImage[Line_X]^.ArrayP[Line_Y] := GBH
      else
         TemSceneBi^.DataImage[Line_X]^.ArrayP[Line_Y] := GBL;
    end;
end; { Algorithum_Step5 }

procedure Algorithum_Step6;
{ Step_6 ..  }
begin
  for Line_Y := 1 to Y_Max do
    for Line_X := 1 to X_Max do
    begin
      TransfromTrueFalse(Line_X,Line_Y);
      result := NOT(Windo[P]) AND
                ( Windo[B] AND Windo[C] AND Windo[E] ) AND
                NOT( Windo[A] OR Windo[D] OR Windo[F] OR Windo[G] OR Windo[H] )
                OR Windo[P] ;
      if result then
         TemSceneBi^.DataImage[Line_X]^.ArrayP[Line_Y] := GBH
      else
         TemSceneBi^.DataImage[Line_X]^.ArrayP[Line_Y] := GBL;
    end;
end; { Algorithum_Step6 }

procedure Display_Smooth;
begin
  gotoXY(60,21); write('X = ',PosiI:3,', Y = ',PosiJ:3);
  for Line_Y := 1 to Y_Max do
    for Line_X := 1 to X_Max do
      if TemSceneBi^.DataImage[Line_X]^.ArrayP[Line_Y] = GBH then
         EGAPlot(Line_X+PosiI,Line_Y+PosiJ,0,15);
end; { Display_Smooth }

procedure SwapResult;
begin
  for Line_Y := 1 to Y_Max do
    for Line_X := 1 to X_Max do
    begin
      TemSmooth^.DataImage[Line_X]^.ArrayP[Line_Y] :=
      TemSceneBi^.DataImage[Line_X]^.ArrayP[Line_Y];
    end;
end; { SwapResult }

procedure Yes_No_Step1;
```

```
begin
  gotoXY(1,23); ClrEol;
  write('Smooth1 : Delete noise of picture  (Y/N) ');
  read(kbd,ch);
  if NOT (UpCase(ch) = 'N') then
  begin
    gotoXY(1,23); ClrEol; write('Compute ! Please wait ... ');
    Algorithum_Step1;
    Display_Smooth;
    SwapResult;
  end;
end; { Yes_No_Step1 }

procedure Yes_No_Step2;
begin
  gotoXY(1,23); ClrEol;
  write('Smooth2 : Fill notches of picture  (Y/N) ');
  read(kbd,ch);
  if NOT (UpCase(ch) = 'N') then
  begin
    gotoXY(1,23); ClrEol; write('Compute ! Please wait ... ');
    Algorithum_Step2;
    Display_Smooth;
    SwapResult;
  end;
end; { Yes_No_Step2 }

procedure Yes_No_Step3;
begin
  gotoXY(1,23); ClrEol;
  write('Smooth3 : Fill corner up-left of picture  (Y/N) ');
  read(kbd,ch);
  if NOT (UpCase(ch) = 'N') then
  begin
    gotoXY(1,23); ClrEol; write('Compute ! Please wait ... ');
    Algorithum_Step3;
    Display_Smooth;
    SwapResult;
  end;
end; { Yes_No_Step3 }

procedure Yes_No_Step4;
begin
  gotoXY(1,23); ClrEol;
  write('Smooth4 : Fill corner up-right of picture  (Y/N) ');
  read(kbd,ch);
  if NOT (UpCase(ch) = 'N') then
  begin
    gotoXY(1,23); ClrEol; write('Compute ! Please wait ... ');
```

```
    Algorithum_Step4;
    Display_Smooth;
    SwapResult;
  end;
end; { Yes_No_Step4 }

procedure Yes_No_Step5;
begin
  gotoXY(1,23); ClrEol;
  write('Smooth5 : Fill corner down-left of picture  (Y/N) ');
  read(kbd,ch);
  if NOT (UpCase(ch) = 'N') then
  begin
    gotoXY(1,23); ClrEol; write('Compute ! Please wait ... ');
    Algorithum_Step5;
    Display_Smooth;
    SwapResult;
  end;
end; { Yes_No_Step5 }

procedure Yes_No_Step6;
begin
  gotoXY(1,23); ClrEol;
  write('Smooth6 : Fill corner down-right of picture  (Y/N) ');
  read(kbd,ch);
  if NOT (UpCase(ch) = 'N') then
  begin
    gotoXY(1,23); ClrEol; write('Compute ! Please wait ... ');
    Algorithum_Step6;
    Display_Smooth;
  end;
end; { Yes_No_Step6 }

begin { Smoothing_Binary_Images }
  mark(HeapTop);
  Initial_Smoothing;
  Yes_No_Step1;
  Yes_No_Step2;
  Yes_No_Step3;
  Yes_No_Step4;
  Yes_No_Step5;
  Yes_No_Step6;
  sound(2000); delay(500); nosound;
  Yes_No_SavePicture(TemSceneBi);
  release(HeapTop);
end;  { Smoothing_Binary_Images }
```

Program :
Programmer :

```
{ ***************************************************************
  *                                                             *
  *             PROCEDURE  BREAK POINT DETECTION                *
  *                                                             *
  *               CREATE  ( MM DD YY ) 10 9 87                  *
  *                                                             *
  *                       VERSION 3.1 A                         *
  *                                                             *
  *             BY    CHANCHAI PISITTIVITHAYANON                *
  *                                                             *
  ***************************************************************}


Overlay procedure CodeBox;
Type
  DataRecP  = ^DataRecPT;
  DataRecPT  = Record
                 DataI  : byte;
                 DataJ  : byte;
                 DataC  : CodeT;
                 NextPnF : DataRecP;
                 NextPn8 : DataRecP;
               end;
Var
  NumE    : integer;
  FirDataRec, LastDataRec ,DumyDataRec : DataRecP ;
  TemDumyDataRec : DataRecP;
  ResultEnFi : File8T;
  FileVar    : DataT;
  FileName   : String50;
  ResultC  : DataPnT;
  ChainC   : CodeT;
  DataRec : DataRecT;
  Result_I : byte;
  Result_J : byte;
  Result_C : CodeT;
  LoadDataActive : boolean;
  LoopOk  : boolean;
  MaxDrop,MinDrop : byte;
  HepTop : ^integer;
```

```
procedure InitialCodeBox;
begin  { InitialCodeBox }
  if X_Max = 66 then
     MaxDrop := 4
  else
     MaxDrop := 7;
  MinDrop := 3 ;
  New (ResultC);
  ResultC^.NextDa := nil ;
  for Line_X := 1 to X_Max do
  begin
    New (ResultC^.DataImage[Line_X]) ;
    ResultC^.DataImage[Line_X]^.NextAr := nil ;
    for Line_Y := 1 to Y_Max do
    begin
      ResultC^.DataImage[Line_X]^.ArrayP[Line_Y] := GBL ;
    end;
  end;
end; { InitialCodeBox }

procedure ReadData;
var
  TemDataRec : DataRecP;
begin  { ReadData }
  FirDataRec := nil ;
  while NOT EOF(FileVar) do
  begin
    read(FileVar,DataRec);
    New (DumyDataRec);
    DumyDataRec^.DataI := DataRec.DataI1;
    DumyDataRec^.DataJ := DataRec.DataJ1;
    DumyDataRec^.DataC := DataRec.DataC1;
    if FirDataRec = nil then
      begin
        FirDataRec := DumyDataRec;
        TemDataRec := DumyDataRec;
      end
    else
      begin
        DumyDataRec^.NextPnB := TemDataRec ;
        LastDataRec^.NextPnF := DumyDataRec;
        TemDataRec           := DumyDataRec;
      end;
    LastDataRec := DumyDataRec;
  end; { while }
  FirDataRec^.NextPnB  := nil ;
  LastDataRec^.NextPnF := FirDataRec ;
end; { ReadData }
```

```
procedure LoadData;
var
  Created    : boolean;
  OK         : char;
begin  { LoadData }
  repeat
    gotoXY(1,23); ClrEol;
    write('Load File name about dataedge ... ');
    readln(FileName);
    if FileName = '' then
      begin
        Created := true;
        LoadDataActive := false;
      end
    else  { NOT(FileName = '') }
      begin
        Assign(FileVar,FileName);
        {$I-}  reset(FileVar);  {$I+}
        Created := IOresult = 0 ;
        if Created then
          begin
            LoadDataActive := true;
            ReadData;
            close(FileVar);
          end
        else  { not create }
          begin
            gotoXY(1,23); ClrEol;
            write(FileName,' File NOT Found  ,  Esc : Exit ');
            sound(2000); delay(500); nosound;
            read(Kbd,OK);
            if OK = #27 then
              begin
                Created := true;
                LoadDataActive := false ;
              end;
          end;
      end; { else not (FileName = '') }
  until Created ;
  sound(2000); delay(500); nosound;
end; { LoadData }


Procedure MovePointer (var SeekDataRec : DataRecP
                     ;  PnDirection : integer ) ;
var
  Count_Seek : integer ;
```

```
begin { MovePointer }
  if PnDirection > 0 then
     for Count_Seek := 1 to PnDirection do
       SeekDataRec := SeekDataRec^.NextPnF ;
  if PnDirection < 0 then
     for Count_Seek := -1 downto PnDirection do
       SeekDataRec := SeekDataRec^.NextPnB ;
end; { MovePointer }


procedure DeletePnt ( DetPn : DataRecP );
var
  tem1,tem2 : DataRecP;
begin { DeletePnt }
  tem1 := DetPn^.NextPnB;
  tem2 := DetPn^.NextPnF;
  tem1^.NextPnF := tem2;
  tem2^.NextPnB := tem1;
  dispose(DetPn);
end; { DeletePnt }

procedure InsertPnt(DumyI : DataRecP ; DataR : DataRecT);
var
  Tem,Ntem : DataRecP;
begin { InsertPnt }
  Tem := DumyI^.NextPnB;
  New(Ntem);
  Ntem^.DataI := DataR.DataI1 ;
  Ntem^.DataJ := DataR.DataJ1 ;
  Ntem^.DataC := DataR.DataC1 ;
  Tem^.NextPnF := Ntem;
  Ntem^.NextPnF := DumyI;
  DumyI^.NextPnB := Ntem;
  NTem^.NextPnB :=  Tem;
end; { InsertPnt }


procedure EvenType(N:byte ; var H,L : byte);
begin { EvenType }
  if N = 0 then
    L := 7
  else
    L := N - 1 ;
  H := N + 1 ;
end; { EvenType }

procedure OddType(N:byte ; var H,L : byte);
begin { OddType }
  if N = 7 then
```

```
    H := 0
  else
    H := N + 1 ;
  L := N - 1 ;
end; { OddType }

procedure T_C_P(C : CodeT ; var x,y : integer);
begin { T_C_P }
  Case  C  OF   {* transform to standas window *}
    0  : begin x := x + 1 ; y := y     ; end;
    1  : begin x := x + 1 ; y := y - 1 ; end;
    2  : begin x := x     ; y := y - 1 ; end;
    3  : begin x := x - 1 ; y := y - 1 ; end;
    4  : begin x := x - 1 ; y := y     ; end;
    5  : begin x := x - 1 ; y := y + 1 ; end;
    6  : begin x := x     ; y := y + 1 ; end;
    7  : begin x := x + 1 ; y := y + 1 ; end;
  end; { CASE }
end; { T_C_P }

function FindCode (x0,y0,x1,y1 : integer) : CodeT;
begin { FindCode }
  if (x1)x0) AND (y1=y0) then FindCode := 0
  else
    if (x1)x0) AND (y1(y0) then FindCode := 1
    else
      if (x1=x0) AND (y1(y0) then FindCode := 2
      else
        if (x1(x0) AND (y1(y0) then FindCode := 3
        else
          if (x1(x0) AND (y1=y0) then FindCode := 4
          else
            if (x1(x0) AND (y1)y0) then FindCode := 5
            else
              if (x1=x0) AND (y1)y0) then FindCode := 6
               else
                 if (x1)x0) AND (y1)y0) then FindCode := 7 ;
end; { FindCode }


procedure TuningLine(Head,Tail : DataRecP);
var
  X0,Y0,X1,Y1 : integer;
  ErrorTerm,HalfX,HalfY : integer;
  x,y,DeltaX,DeltaY,Xstep,Ystep : integer;
  Dumy : DataRecP;
  DataR : DataRecT;
begin { TuningLine }
  Dumy := Head^.NextPnF;
```

```
  repeat
    DeletePnt(Dumy);
    Dumy := Head^.NextPnF;
  until ( (Dumy^.DataI = Tail^.DataI) AND (Dumy^.DataJ = Tail^.DataJ) ) ;
  Dumy := Head;
  XO := Head^.DataI ; YO := Head^.DataJ ;
  X1 := Tail^.DataI ; Y1 := Tail^.DataJ ;
  x := XO; y := YO;
  Xstep := 1; Ystep := 1;
  if XO ) X1 then Xstep := -1;
  if YO ) Y1 then Ystep := -1;
  DeltaX := abs(X1-XO);
  DeltaY := abs(Y1-YO);
  ErrorTerm := 0;
  if NOT ((xO=x1) AND (yO=y1)) then
     if DeltaX ) DeltaY then
       begin
         HalfX := trunc(DeltaX/2);
         repeat
           x := x + Xstep;
           ErrorTerm := ErrorTerm + DeltaY;
           if NOT(ErrorTerm (= HalfX) then
             begin
               ErrorTerm := ErrorTerm - DeltaX;
               y := y + Ystep;
             end;
           DataR.DataI1 := x; DataR.DataJ1 := y;
           DataR.DataC1 := 8;
           Dumy^.DataC  := FindCode(Dumy^.DataI,Dumy^.DataJ,x,y);
           InsertPnt(Tail,DataR);
           Dumy := Dumy^.NextPnF;
         until (x=x1) and (y = y1);
       end
     else
       begin
         HalfY := trunc(DeltaY/2);
         repeat
           y := y + Ystep;
           ErrorTerm := ErrorTerm + Deltax;
           if Not(ErrorTerm (= HalfY) then
             begin
               ErrorTerm := ErrorTerm - DeltaY;
               x := x + Xstep;
             end;
           DataR.DataI1 := x; DataR.DataJ1 := y;
           DataR.DataC1 := 8;
           Dumy^.DataC := FindCode(Dumy^.DataI,Dumy^.DataJ,x,y);
           InsertPnt(Tail,DataR);
           Dumy := Dumy^.NextPnF;
```

```
        until (x = x1) and (y = y1);
      end;
  DeletePnt(Dumy);
end; { TuningLine }

function NearStart(Head,Tail : DataRecP) : Boolean ;
begin { NearStart }
  NearStart := false ;
  while NOT (Head = Tail) do
  begin
    if Head^.NextPnB = nil then
      NearStart := true ;
    Head := Head^.NextPnF;
  end;
end; { NearStart }

procedure RePointStart(Head : DataRecP) ;
var
  NewFirst : DataRecP ;
  Tem : byte;
begin { RePointStart }
  FirDataRec := Head ;
  Tem := FirDataRec^.DataJ ;
  while FirDataRec^.DataJ <= Tem do
  begin
    Tem := FirDataRec^.DataJ;
    FirDataRec := FirDataRec^.NextPnF;
  end;
  FirDataRec := FirDataRec^.NextPnB;
  LastDataRec := FirDataRec^.NextPnB ;
  FirDataRec^.NextPnB := nil;
end; { RePointStart }


procedure ConsiderNoise (Head:DataRecP) ;
Const
  Noi = 2;
{  MaxDrop = 7;  see Initial Code Box
   MinDrop = 3; }
var
  Norm,Hi,Low : byte;
  Tail : DataRecP;
  Drop,Noise : integer;
  GateHi,GateLow : boolean;
begin { ConsiderNoise }
  Tail := Head ;
  Norm := Head^.DataC;
  Drop := 1; Noise := 0 ; GateHi := true; GateLow := true;
  if Odd(Head^.DataC) then
```

```
    OddType(Norm,Hi,Low)
  else
    EvenType(Norm,Hi,Low);
  repeat
    Tail := Tail^.NextPnF;
    if Tail^.DataC = Hi then
      begin
        Noise := Noise + 1 ;
        if GateHi then GateHi := false ;
      end
    else
      if Tail^.DataC = Low then
        begin
          Noise := Noise - 1;
          if GateLow then GateLow := false ;
        end;
    Drop := Drop + 1;
  until (Noi < {=} ABS(Noise)) OR
        ( (Drop > MinDrop)AND(ABS(Noise) <= Noi)AND(Tail^.DataC = Norm) )
         OR (Drop >= MaxDrop) OR (NOT (Tail^.DataC IN [Norm,Hi,Low]) ) ;
  if ( (ABS(Noise) < Noi) AND (Tail^.DataC = Norm) AND
       NOT(GateHi) AND NOT(GateLow)  ) then
     begin
       if NearStart(Head,Tail) then
         begin
           TuningLine(Head,Tail);
           RePointStart(Head);
         end
       else
         TuningLine(Head,Tail);
     end;   .
end; { ConsiderNoise }

procedure ReduceNoise;
var
  Normal : byte;
  X_DataRec : DataRecP;
begin { ReduceNoise }
  DumyDataRec := FirDataRec;
  repeat
    Normal := DumyDataRec^.DataC;
    X_DataRec := DumyDataRec^.NextPnF;
    if NOT (Normal = X_DataRec^.DataC) then
      ConsiderNoise(DumyDataRec);
    DumyDataRec := DumyDataRec^.NextPnF;
  until DumyDataRec = FirDataRec ;
end; { ReduceNoise }

procedure Put_Conner(Dumy : DataRecP );
```

```
var
  TemDumy : DataRecP;
  Ndata : DataRecT;
  I,J : integer;
  CC  : CodeT;
begin { Put_Conner }
  TemDumy := Dumy^.NextPnF ;
  TemDumy^.DataC := Dumy^.DataC ;
  I := TemDumy^.DataI ;
  J := TemDumy^.DataJ ;
  CC := Dumy^.DataC ;
  T_C_P(CC,I,J);
  TemDumy := TemDumy^.NextPnF;
  Ndata.DataC1 := TemDumy^.DataC;
  Ndata.DataI1 := I ;
  Ndata.DataJ1 := J ;
  InsertPnt(TemDumy,Ndata);
end; { Put_Conner }


procedure Save_New_DE ;
var
  Filevar_Text  : Text ;
  pos : byte ;
  a   : integer ;
  chr : char ;
  Str_X,Str_Y : string[5] ;
begin { Save_New_DE }
  DumyDataRec := FirDataRec ;
  repeat
    EGAPlot((DumyDataRec^.DataI+PosiI),(DumyDataRec^.DataJ+PosiJ),0,3);
    DumyDataRec := DumyDataRec^.NextPnF;
  until DumyDataRec = FirDataRec ;
  pos := 0 ;
  repeat
    pos := pos + 1 ;
    chr := copy(filename,pos,1);
  until (chr = '.') OR (chr = ' ');
  if chr = '.' then
  begin
    delete(filename,pos,3);
    filename := concat(filename,'.DE1');
  end;
  gotoXY(1,23); ClrEol ;
  write('SAVE New Data edge ',Filename  ,' [ Y / N ]  Esc : Key Filename ');
  read(kbd,ch);
  if Upcase(ch) = 'E' then
    begin
      gotoXY(1,23); ClrEol ; write('Enter Filename '); read(Filename);
```

```
    end;
  if (UpCase(ch) = 'Y') OR ( (Upcase(ch) = 'E') AND NOT(Filename = '') ) then
    begin
      assign(Filevar_Text,Filename);
      rewrite(Filevar_Text);
      DumyDataRec := FirDataRec ;
      repeat
        Str(DumyDataRec^.DataI,Str_X) ;
        a := length(Str_X);
        for Pos := a to 5 do
          Str_X := concat(Str_X,' ');
        Str(DumyDataRec^.DataJ,Str_Y) ;
        writeln(Filevar_Text,concat(Str_X,Str_Y));
        DumyDataRec := DumyDataRec^.NextPnF ;
      until DumyDataRec = FirDataRec ;
      close(Filevar_Text);
    end;
  sound(3000); delay(100); nosound;
end; { Save_New_DE }

procedure Find_NHL(Nom : CodeT ; var Hi,Lo : CodeT);
begin { Find_NHL }
  if ( Nom = 0) OR ( Nom = 7) then
    begin
      Case Nom of                                    {              }
           0 : begin  Lo := 7 ; Hi := 1 ;  end;      { Protect 0 & 7 }
           7 : begin  Lo := 6 ; Hi := 0 ;  end;      {              }
          end;                                       {              }
    end
  else
    begin
      Hi := Nom + 1 ;
      Lo := Nom - 1 ;
    end;
end; { Find_NHL }

procedure FindTuning_NHL (Dumy:DataRecP ; var Nom,HL:CodeT ;
                                          var Leng:byte);
var
  CountNom , CountHL : byte;
  TemHL : CodeT;
begin { FindTuning_NHL }
  CountNom := 1; CountHL := 0;
  while Dumy^.DataC = HL do
  begin
    CountHL := CountHL + 1 ;
    MovePointer(Dumy,1);
  end;
  if CountHL ) CountNom then   { probability ; CountHL )= CountNom }
```

```
    begin
      Leng := CountHL;
      TemHL := Nom;
      Nom   := HL;
      HL    := TemHL;
    end
  else
      Leng := 1;
end; { FindTuning_NHL }

procedure SimulateIn ( var SimDataRec : DataRecP ) ;
Const
  initial = 8 ;  { The 8 can change form 0..8 , But not effect  }
  SetErrCC = 3 ;  { SetErrCC is Seting Error Curve Constan }
  SizePointing = 1 ;
Var
  Norm,high,low,HiLo : CodeT;
  SetLen,errSet : byte;

  function SetErr (leng : byte) : byte;
  Const
    MaxLen = 5;
  begin  { SetErr }
    if leng >= MaxLen then       {                }
      SetErr := 2                {  Offset error  }
    else                         {                }
      SetErr := SetErrCC;        {                }
  end;  { SetErr }

  procedure HelpMe (errS : byte; Nor,HL :CodeT ) ;
  var
    errCout : byte;
    TemC : CodeT;
    Count : integer;
  begin  { HelpMe }
    errCout := 1 ;
    repeat
      MovePointer(SimDataRec,SizePointing);
      if SimDataRec = FirDataRec then LoopOK := true ;
      TemC := SimDataRec^.DataC ;
      if TemC = Nor then
        errCout := 0 ;
      if TemC = HL then
        errCout := errCout + 1 ;
    until ((TemC <> Nor) AND (TemC <> HL)) OR (errCout >= errS) OR (LoopOk) ;
    if (errCout >= errS) AND NOT(LoopOk) then
       begin
         Count := -1 * (errCout-1) ;
         MovePointer(SimDataRec,Count);
```

```
      end;
  end; { HelpMe }

begin  { SimulateLn }
  LoopOk := false ;
  Norm := SimDataRec^.DataC ;
  Find_NHL(Norm,High,Low);
  SetLen := {1} 0 ;
  while Norm = SimDataRec^.DataC do
  begin
    SetLen := SetLen + 1 ;
    MovePointer(SimDataRec,SizePointing);
    if SimDataRec = FirDataRec then
      LoopOk := true ;
  end;
  HiLo := SimDataRec^.DataC ;
  if ( HiLo = High ) OR (HiLo = Low ) AND NOT(LoopOK) then
    begin
      if HiLo = High then
        begin
          if SetLen = 1 then
            FindTuning_NHL(SimDataRec,Norm,High,SetLen);
          errSet := SetErr(SetLen);
          HelpMe (errSet,Norm,High);
        end
      else { HiLo = Low }
        begin
          if SetLen = 1 then
            FindTuning_NHL(SimDataRec,Norm,Low,SetLen);
          errSet := SetErr(SetLen);
          HelpMe (errSet,Norm,Low );
        end;
    end;
end;  { SimulateLn }

procedure WriteDataRec;
begin { WriteDataRec }
  rewrite(FileVar);
  DumyDataRec := FirDataRec ;
  repeat
    DataRec.DataI1 := DumyDataRec^.DataI;
    DataRec.DataJ1 := DumyDataRec^.DataJ;
    DataRec.DataC1 := DumyDataRec^.DataC;
    write(FileVar,DataRec);
    DumyDataRec := DumyDataRec^.NextPnF;
  until DumyDataRec = FirDataRec ;
  close(FileVar);
end; { WriteDataRec }
```

```
procedure SaveResultDataRec;
var
  Created : boolean;
  ch      : char;
begin { SaveResultDataRec }
  repeat
    gotoXY(1,23); ClrEol;
    write('Enter Filename for SAVE Encode data ( X,Y:C ) ... ');
    read(FileName);
    if FileName = '' then
      begin
        Created := true;
      end
    else
      begin
        assign(FileVar,FileName);
        {$I-} reset(FileVar); {$I+}
        Created := IOresult = 0 ;
        if NOT Created then
          begin
            WriteDataRec;
            Created := true ;
          end { NOT Create }
        else
          begin
            gotoXY (1,23); ClrEOL;
            write(FileName,' Exist ¦ Enter : OverWRITE  ,  Esc : Quit ');
            sound(2000); delay(500); nosound;
            read(kbd,ch);
            if NOT( (ch = Enter) OR (ch = Esc) ) then  Created := false ;
            if ch = Esc  then  Created := true  ;
            if ch = #$00 then
              begin
                rewrite(FileVar);
                WriteDataRec;
                Created := true ;
              end;
          end;
      end; { else  FileName = '' }
  until Created ;
  sound(2000); delay(500); nosound;
end; { SaveResultDataRec }

procedure DisplayResultPicture;
begin { DisplayResultPicture }
  gotoXY(1,23); ClrEol;
  write('Display Encode Result (Y/N) ? Yes ');
  read(Kbd,ch);
  if Upcase(ch) = 'Y' then
```

```
    begin
      Wait_Message;
      for Line_Y := 1 to Y_Max do
        for Line_X := 1 to X_Max do
          if (ResultC^.DataImage[Line_X]^.ArrayP[Line_Y]) = GBH then
              EGAPlot(Line_X+PosiI,Line_Y+PosiJ,0,12);
    end;
end; { DisplayResultPicture }


procedure SaveResultPicture;
begin  { SaveResultPicture }
  gotoXY(1,23); ClrEol;
  write('Enter FILENAME for SAVE Encode picture ');
  read(FileName);
  if NOT(FileName = '') then
    begin
      OpenSaveFile(ResultEnFi,FileName);
      for Line_Y := 1 to Y_Max do
        for Line_X := 1 to X_Max do
        begin
          write (ResultEnFi,ResultC^.DataImage[Line_X]^.ArrayP[Line_Y]);
        end;
      close(ResultEnFi);
    end;
end; { SaveResultPicture }

procedure ReMovePoint(var First,Secon : DataRecP);
const
  Direction = -1;
var
  Tem : DataRecP;
begin { ReMovePoint }
  if NOT(First = Secon) then  { protect the same Heap }
  begin
    Tem := Secon ;
    First^.NextPnF := Tem ;
    MovePointer(Tem,Direction);
  end;
end; { ReMovePoint }


procedure Full_SimulateLn;
begin { Full_SimulateLn }
  DumyDataRec := FirDataRec ;
  TemDumyDataRec := FirDataRec;
  repeat
    SimulateLn(DumyDataRec);
    if LoopOk then
```

```
      TemDumyDataRec^.NextPnF := FirDataRec
    else
      ReMovePoint(TemDumyDataRec,DumyDataRec);
    TemDumyDataRec := DumyDataRec ;
    Result_I := DumyDataRec^.DataI;
    Result_J := DumyDataRec^.DataJ;
    ResultC^.DataImage[Result_I]^.ArrayP[Result_J] := GBH ;
  until LoopOk ;
end; { Full_SimulateLn }

begin  { CodeBox }
  Mark(HepTop);
  InitialCodeBox;
  LoadData;
  if LoadDataActive then
    begin
      ReduceNoise ;
      Save_New_DE ;
      Full_SimulateLn;
      DisplayResultPicture ;
      SaveResultDataRec ;
      SaveResultPicture;
    end;
  sound(2000); delay(500); nosound;
  release(HepTop);
end; { CodeBox }
```

Program :
Programmer :

```
{ ***************************************************************
  *                                                             *
  *          PROCEDURE  DRAW LINE BETWEEN BREAK POINT           *
  *                                                             *
  *               CREATE  ( MM DD YY ) 24 12 87                 *
  *                                                             *
  *                       VERSION 1.0                           *
  *                                                             *
  *             BY  CHANCHAI PISITTIVITHAYANON                  *
  *                                                             *
  ***************************************************************}


Overlay procedure Create_Line ;
type
  DataRecP   = ^DataRecPT;
  DataRecPT  = Record
                 DataI  : real;
                 DataJ  : real;
                 DataC  : CodeT;
                 NextPnF : DataRecP;
                 NextPnB : DataRecP;
               end;
  FileRecT = FILE OF DataRecT;
  EleMxLyF = (A11,A12,A21,A22) ;  { Element matrix Formular }
  FMatrixT = array[EleMxLyF] of real;
  MatrixTT = array[1..2] of real;
var
  FMatrix  : FMatrixT;
  FileName : string50;
  FileVar  : FileRecT;
  DataRec  : DataRecT;
  FirDataRec , LastDataRec , DumyDataRec : DataRecP;
  LoadDataRecActive : boolean;

procedure ReadDataRec;
var
  TemDataRec : DataRecP;
begin { ReadDataRec }
  FirDataRec := nil;
```

```
  while NOT EOF(FileVar) do
  begin
    read(FileVar,DataRec);
    New (DumyDataRec);
    DumyDataRec^.DataI := DataRec.DataI1;
    DumyDataRec^.DataJ := DataRec.DataJ1;
    DumyDataRec^.DataC := DataRec.DataC1;
    if FirDataRec = nil then
      begin
        FirDataRec := DumyDataRec;
        TemDataRec := DumyDataRec;
      end
    else
      begin
        DumyDataRec^.NextPnB := TemDataRec ;
        LastDataRec^.NextPnF := DumyDataRec;
        TemDataRec           := DumyDataRec;
      end;
    LastDataRec := DumyDataRec;
  end;
  FirDataRec^.NextPnB  := nil ;
  LastDataRec^.DataC   := 8 ; { Protect run time error. (1..8 Not effect) }
  LastDataRec^.NextPnF := FirDataRec ;
end; { ReadDataRec }.


procedure LoadDataRec;
var
  Created : boolean;
  ch      : char;
begin  { LoadDataRec }
  repeat
    gotoXY(1,23); ClrEol;
    write('Load File name about ENCODE dataedge ... ');
    readln(FileName);
    if FileName = '' then
      begin
        Created := true ;
        LoadDataRecActive := false ;
      end
    else { NOT (FileName = '') }
      begin
        Assign(FileVar,FileName);
        {$I-} reset(FileVar); {$I+}
        Created := IOresult = 0 ;
        if Created then
          begin
            LoadDataRecActive := true ;
            ReadDataRec;
```

```
            close(FileVar);
          end
        else { not Created }
          begin
            gotoXY(1,23); ClrEol;
            write(FileName,' File NOT Found  ,  Esc : Exit ');
            sound(2000); delay(500); nosound;
            read(Kbd,ch);
            if ch = #27 then
              begin
                Created := true;
                LoadDataRecActive := false ;
              end;
          end;
      end; { NOT (FileName = '') }
  until Created ;
end; { LoadDataRec }


procedure Take_Formular(zeta:real);
begin
  FMatrix[A11] := cos(zeta);
  FMatrix[A12] := sin(zeta);
  FMatrix[A21] := -sin(zeta);
  FMatrix[A22] := FMatrix[A11];
end;  { Formular_Matrix }

procedure Product_Matrix(var  Result : MatrixTT) ;
var
  temResult  : MatrixTT;
begin
  temResult[1] := FMatrix[A11] * Result[1] + FMatrix[A21] * Result[2] ;
  temResult[2] := FMatrix[A12] * Result[1] + FMatrix[A22] * Result[2] ;
  Result[1]    := temResult[1] ;
  Result[2]    := temResult[2] ;
end; { Product_Matrix }

procedure Sub_Matrix (var dataXY : MatrixTT);
begin { Sub_Matrix }
  dataXY[1] := dataXY[1] - Mean_X ;
  dataXY[2] := dataXY[2] - Mean_Y ;
end; { Sub_Matrix }

procedure Add_Matrix (var dataXY : MatrixTT) ;
begin { Add_Matrix }
  dataXY[1] := dataXY[1] + Mean_X ;
  dataXY[2] := dataXY[2] + Mean_Y ;
end;  { Add_Matrix }
```

```
procedure Rota_DataRecP (Alpha : real);
var
  Matrix : MatrixTT;
begin { Rota_DataRecP }
  Take_Formular(Alpha);
  DumyDataRec := FirDataRec ;
  repeat
    Matrix[1] := DumyDataRec^.DataI ;
    Matrix[2] := DumyDataRec^.DataJ ;
    Sub_Matrix(Matrix);
    Product_Matrix(Matrix);
    Add_Matrix(Matrix);
    DumyDataRec^.DataI := Matrix[1];
    DumyDataRec^.DataJ := Matrix[2];
    DumyDataRec := DumyDataRec^.NextPnF ;
  until ( (DumyDataRec^.DataI = FirDataRec^.DataI) AND
          (DumyDataRec^.DataJ = FirDataRec^.DataJ) ) ;
end; { Rota_DataRecP }

procedure DrawLineBetweenTowPoint ( First , Secon : DataRecP) ;
var
  X1,X2,Y1,Y2 : integer;
begin  { DrawLineBetweenTowPoint }
  X1 := round(First^.DataI + PosiI);
  Y1 := round(First^.DataJ + PosiJ);
  X2 := round(Secon^.DataI + PosiI);
  Y2 := round(Secon^.DataJ + PosiJ);
  EGADraw(X1,Y1,X2,Y2,0,14);
end;  { DrawLineBetweenTowPoint }

procedure EditLine;
var
  TemDataRec : DataRecP ;
begin  { EditLine }
  DumyDataRec := FirDataRec ;
  TemDataRec  := DumyDataRec ;
  repeat
    DumyDataRec := DumyDataRec^.NextPnF ;
    DrawLineBetweenTowPoint (TemDataRec,DumyDataRec);
    TemDataRec := DumyDataRec ;
    read(kbd,ch);
  until ( (FirDataRec^.DataI = TemDataRec^.DataI) AND
          (FirDataRec^.DataJ = TemDataRec^.DataJ) );
end; { EditLine }

begin  { Create_Line }
  Mark(HeapTop);
  LoadDataRec;
```

```
  if LoadDataRecActive then
  begin
    gotoXY(1,23); ClrEol;
    write('Rotate_Angle ','[ ',Zeta*180*pi:10:3,'Deg. ] : ');
    read(Zeta);
    Zeta := Zeta*pi/180 ;
    if NOT (Zeta = 0) then
       Rota_DataRecP(Zeta);
    EditLine;
  end;
  Release(HeapTop);
  sound(2000); delay(500); nosound;
end;  { Create_Line }
```

Program :
Programmer :

```
{ ****************************************************************
  *                                                              *
  *          PROCEDURE  OBJECTS COMPARISON IN L-A CURVE          *
  *                                                              *
  *               CREATE  ( MM DD YY ) 1 31 88                   *
  *                                                              *
  *                       VERSION 2.10                           *
  *                                                              *
  *             BY    CHANCHAI PISITTIVITHAYANON                 *
  *                                                              *
  ****************************************************************}


Overlay procedure Compare;
const
  ZeroX = 20;
  ZeroY = 250;
type
  DataRecP  = ^DataRecPT;
  DataRecPT  = Record
                 DataI  : byte;
                 DataJ  : byte;
                 DataC  : CodeT;
                 NextPnF : DataRecP;
                 NextPnB : DataRecP;
               end;
  DataRozaP = ^DataRoza;
  DataRoza  = Record
                RoSeg : real;
                ZaSeg : real;
                NextPnF : DataRozaP;
                NextPnB : DataRozaP;
              end;

var
  FileVar : DataT;
  FileName : String[50];
  HeapTop : ^integer;
  FirDataRec,DumyDataRec,LastDataRec : DataRecP;
  DataRec : DataRecT;
```

```
  LoadDataAcTive : boolean;
  FirRoza1,FirRoza2,DumyRoza,LastRoza : DataRozaP;
  RozaFu : DataRozaP;


procedure ReadData;
var
  TemDataRec : DataRecP;
begin  { ReadData }
  FirDataRec := nil ;
  while NOT EOF(FileVar) do
  begin
    read(FileVar,DataRec);
    New (DumyDataRec);
    DumyDataRec^.DataI := DataRec.DataI1;
    DumyDataRec^.DataJ := DataRec.DataJ1;
    DumyDataRec^.DataC := DataRec.DataC1;
    if FirDataRec = nil then
      begin
        FirDataRec := DumyDataRec;
        TemDataRec := DumyDataRec;
      end
    else
      begin
        DumyDataRec^.NextPnB := TemDataRec ;
        LastDataRec^.NextPnF := DumyDataRec;
        TemDataRec           := DumyDataRec;
      end;
        LastDataRec := DumyDataRec;
  end; { while }
  FirDataRec^.NextPnB  := nil ;
  LastDataRec^.NextPnF := FirDataRec ;
end; { ReadData }


procedure LoadData;
var
  Created    : boolean;
  OK         : char;
begin  { LoadData }
  repeat
    gotoXY(1,23); ClrEol;
    write('Load File name about X,Y : C for ROZA  ... ');
    readln(FileName);
    if FileName = '' then
      begin
        Created := true;
        LoadDataActive := false;
      end
```

```
    else  { NOT(FileName = '') }
      begin
        Assign(FileVar,FileName);
        {$I-}  reset(FileVar);  {$I+}
        Created := IOresult = 0 ;
        if Created then
          begin
            LoadDataActive := true;
            ReadData;
            close(FileVar);
          end
        else  { not create }
          begin
            gotoXY(1,23); ClrEol;
            write(FileName,' File NOT Found  ,  Esc : Exit ');
            sound(2000); delay(500); nosound;
            read(Kbd,OK);
            if OK = #27 then
              begin
                Created := true;
                LoadDataActive := false ;
              end;
          end;
      end; { else not (FileName = '') }
  until Created ;
  sound(2000); delay(500); nosound;
end; { LoadData }


procedure Calculate_Roza(Dumy : DataRecP ; MinLeng : byte ;
                         var RoS,ZaS : real ; var Jump : boolean );
                        { Ros in Lenght , ZaS in Degrees }
var
  x,y,x1,y1,x2,y2,x3,y3 : real;
  AA,BB,CC  : real;
begin { Calculate_Roza }
  Jump := false ;
  x1 := Dumy^.DataI ; y1 := Dumy^.DataJ ;
  Dumy := Dumy^.NextPnF;
  x2 := Dumy^.DataI ; y2 := Dumy^.DataJ ;
  Dumy := Dumy^.NextPnF;
  x3 := Dumy^.DataI ; y3 := Dumy^.DataJ ;
  if (sqrt(sqr(x2-x1)+sqr(y2-y1)) < MinLeng ) {OR (ZaS < 10 )} then
    begin     { first side = 3 pixel }
      x2 := x3 ;  y2 := y3 ;
      Dumy := Dumy^.NextPnF ;
      x3 := Dumy^.DataI ; y3 := Dumy^.DataJ ;
      Jump := true ;
    end;
```

```
  if (sqrt(sqr(x3-x2)+sqr(y3-y2)) < MinLeng ) {OR (ZaS < 10 )} then
    begin  {  second side = 3 pixel }
      Dumy := Dumy^.NextPnF ;
      x3 := Dumy^.DataI ; y3 := Dumy^.DataJ ;
    end;
  AA := sqr(x3-x2)+sqr(y3-y2) ;
  BB := sqr(x1-x3)+sqr(y1-y3) ;
  CC := sqr(x2-x1)+sqr(y2-y1) ;
  ZaS := ArcCos( (AA + CC - BB) / (2*sqrt(AA*CC)) );
  if ZaS > 90 then  ZaS := 180 - ZaS ;
  RoS := sqrt(CC) ;
end; { Calculate_Roza }

function Round_of_Object(RecDumy : DataRecP) : byte ;
var   { calculate percent of leng on object , 2.63 % }
  Dumy : DataRecP ;
  x1,y1,x2,y2,Leng : real ;
begin { Round_Of_Object }
  Leng := 0 ;
  Dumy := RecDumy ;
  repeat
    x1 := Dumy^.DataI ;
    y1 := Dumy^.DataJ ;
    Dumy := Dumy^.NextPnF ;
    x2 := Dumy^.DataI ;
    y2 := Dumy^.DataJ ;
    Leng := Leng + Sqrt(sqr(x2-x1)+sqr(y2-y1) ) ;
  until Dumy = RecDumy ;
  if Leng > 250 then
    Round_Of_Object := 10  { for size 256 * 256 }
  else
    Round_Of_Object := 3 ;  { for size less than 64 * 64 }
end; { Round_Of_Object }


procedure Transform_To_Roza(var FirRecDumy : DataRecP ;
                            var FirRoDumy : DataRozaP);
var
  TemRec : DataRecP;
  TemRo,LastRo,DumyRo : DataRozaP ;
  RoS,ZaS : real;
  Jump : boolean;
  MinLeng : byte;
begin { Transform_To_Roza }
  FirRoDumy := nil;
  TemRec := FirRecDumy;
  MinLeng := Round_Of_Object (FirRecDumy);
  repeat
    Jump := false;
```

```
    Calculate_Roza(TemRec,MinLeng,RoS,ZaS,Jump);
    New (DumyRo);
    DumyRo^.RoSeg := RoS ;
    DumyRo^.ZaSeg := ZaS ;
    if FirRoDumy = nil then
      begin
        FirRoDumy := DumyRo ;
        TemRo     := DumyRo ;
      end
    else
      begin
        DumyRo^.NextPnB := TemRo ;
        LastRo^.NextPnF := DumyRo ;
        TemRo           := DumyRo ;
      end;
    LastRo := DumyRo ;
    if Jump then
      begin
        TemRec := TemRec^.NextPnF;
      end;
    TemRec := TemRec^.NextPnF;
  until TemRec = FirRecDumy ;
  FirRoDumy^.NextPnB := LastRo ;
  LastRo^.NextPnF  := FirRoDumy ;
end; { Transform_To_Roza }

procedure ShowResult(QumyRo : DataRozaP ; Color :integer);
var
  TemFir      : DataRozaP;
  X,TemX,TemY : integer;
  TemZa       : integer;
  RoS,ZaS,TemRoS : integer;
begin { ShowResult }
  TemFir := DumyRo ;
  TemX := ZeroX ;
  TemY := ZeroY ;
  TemZa := 0;
  repeat
    RoS := round(DumyRo^.RoSeg);
    TemZa := TemZa + round(DumyRo^.ZaSeg/10) ;
    ZaS :=  ZeroY - TemZa ;
    EGADraw(TemX,ZaS,TemX,TemY,0,Color);
    TemY := ZaS;
    for x := TemX to RoS+TemX do
      EGAPlot(x,ZaS,0,Color);
    TemX := TemX + RoS ;
    DumyRo := DumyRo^.NextPnF;
  until TemFir = DumyRo;
end; { ShowResult }
```

```
procedure InitialDataFu(Transive : DataRozaP ;
                        var Receive : DataRozaP );
var
  TemRo,LastRo,DumyRo : DataRozaP ;
  FirTr : DataRozaP ;
  TemZeta : real ;
begin { InitialDataFu }
  FirTr := Transive;
  Receive := nil ;
  TemZeta := 0;
  repeat
    new(DumyRo);
    DumyRo^.RoSeg := Transive^.RoSeg ;
    TemZeta := TemZeta + Transive^.ZaSeg ;
    DumyRo^.ZaSeg := TemZeta ;
    if Receive = nil then
      begin
        Receive := DumyRo ;
        TemRo   := DumyRo ;
      end
    else
      begin
        DumyRo^.NextPn8 := TemRo;
        LastRo^.NextPnF := DumyRo;
        TemRo           := DumyRo;
      end;
    LastRo := DumyRo;
    Transive := Transive^.NextPnF;
  until Transive = FirTr;
  Receive^.NextPn8 := LastRo;
  LastRo^.NextPnF := Receive ;
end; { InitialDataFu }

procedure EnterDataFu(Transive : DataRozaP ; var Receive : DataRozaP );
var
  FirTr,DumyFu : DataRozaP;
  TemZeta : real;
begin { EnterDataFu }
  FirTr := Transive ;
  DumyFu := Receive ;
  TemZeta := 0 ;
  repeat
    DumyFu^.RoSeg := Transive^.RoSeg ;
    TemZeta := TemZeta + Transive^.ZaSeg ;
    DumyFu^.ZaSeg := TemZeta ;
    Transive := Transive^.NextPnF;
    DumyFu := DumyFu^.NextPnF;
```

```
  until Transive = FirTr ;
end; { EnterDataFu }

function Convolution(Reffer,Convo : DataRozaP) : real ;
var
  FirReffer , FirConvo : DataRozaP;
  TemRoSeg , TemRoSegA , TemRoSegB : real ;
  DeltaZe , SumErr : real;
  Passed1 , Passed2 : boolean ;
begin { Convolution }
  FirReffer := Reffer ;
  FirConvo  := Convo  ;
  SumErr := 0 ;
  TemRoSegA := Reffer^.RoSeg ;
  TemRoSegB := Convo^.RoSeg ;
  Passed1 := false ;
  Passed2 := false ;
  repeat
    DeltaZe := ABS(Reffer^.ZaSeg - Convo^.ZaSeg);
    TemRoSeg := TemRoSegA - TemRoSegB ;
    if TemRoSeg >= 0 then
      begin { Reffer^.RoSeg > Convo^.RoSeg }
        SumErr := SumErr + (DeltaZe * TemRoSegB);
        TemRoSegA := TemRoSegA - TemRoSegB ;
        Convo := Convo^.NextPnF ;
        TemRoSegB := Convo^.RoSeg ;
        Passed1 := true ;
      end
    else
      begin { Reffer^.RoSeg < Convo^.RoSeg }
        SumErr := SumErr + (DeltaZe * TemRoSegA);
        TemRoSegB := TemRoSegB - TemRoSegA ;
        Reffer := Reffer^.NextPnF ;
        TemRoSegA := Reffer^.RoSeg ;
        Passed2 := true ;
      end;
  until Passed1 AND Passed2 AND
        ( (FirReffer = Reffer) OR (FirConvo = Convo) ) ;
  if FirReffer = Reffer then { other mean TemRoSeg >= 0 }
    begin
      repeat
        SumErr := SumErr + (TemRoSegB * Convo^.ZaSeg) ;
        Convo := Convo^.NextPnF ;
        TemRoSegB := Convo^.RoSeg ;
      until Convo = FirConvo ;
    end
  else  { FirConvo = Convo }
    if FirConvo = Convo then
      begin
```

```
        repeat
          SumErr := SumErr + (TemRoSegA * Reffer^.ZaSeg) ;
          Reffer := Reffer^.NextPnF ;
          TemRoSegA := Reffer^.RoSeg ;
        until Reffer = FirReffer ;
      end;
  Convolution := SumErr ;
end; { Convolution }

function TotalArea(Dumy : DataRozaP) : real ;
var
  TemFir : DataRozaP ;
  SumArea : real ;
begin { TotalArea }
  SumArea := 0 ;
  TemFir := Dumy ;
  repeat
    SumArea := SumArea + (Dumy^.RoSeg * Dumy^.ZaSeg) ;
    Dumy := Dumy^.NextPnF ;
  until TemFir = Dumy ;
  TotalArea := SumArea ;
end; { TotalArea }

procedure Main_Roza;
var
  ReffArea,ErrArea : real;
  DumyFu1 : DataRozaP;
  DumyFu2 : DataRozaP;
  DumyRoza1 , DumyRoza2 : DataRozaP;

begin { Main_Roza }
  Transform_To_Roza(FirDataRec,FirRoza1);
  InitialDataFu(FirRoza1,DumyFu1);
  ShowResult(FirRoza1,12);
  ReffArea := TotalArea(DumyFu1);
  Plotaxis(ZeroX,ZeroY,1);
  gotoXY(1,22); write('Taget Senece ');
  LoadData;
  gotoXY(1,22); write('                ');
  if LoadDataActive then
  begin
    gotoXY(40,22); write('2 : ',FileName);
    Transform_To_Roza(FirDataRec,FirRoza2);
    InitialDataFu(FirRoza2,DumyFu2);
    ShowResult (FirRoza2,9);
    DumyRoza1 := FirRoza1 ;
    DumyRoza2 := FirRoza2 ;
    repeat
      ErrArea := Convolution(DumyFu1,DumyFu2);
```

```
      gotoXY(5,3);
      write('Percent Error ( % ) = ',(ErrArea * 100 / ReffArea):10:4 );
      gotoXY(5,2);
      write('ReffArea = ',ReffArea:10:4,'  ErrArea = ',ErrArea:10:4);
      repeat
        gotoXY(1,23); ClrEol;
        write('[4] Reff. Object Shift :: [6] Taget Object Shift :: [Q]uit ');
        read(kbd,ch);
      until ch IN ['4','6','Q','q'] ;
      PlotAxis(ZeroX,ZeroY,1);
      ShowResult(DumyRoza2,0);
      ShowResult(DumyRoza1,12);
      if ch = '4' then
        begin
          ShowResult(DumyRoza1,0);
          DumyRoza1 := DumyRoza1^.NextPnF;
          EnterDataFu (DumyRoza1,DumyFu1);
          ReffArea := TotalArea(DumyFu1);
          ShowResult (DumyRoza1,12);
        end
      else
        if ch = '6' then
          begin
            ShowResult (DumyRoza2,0);
            DumyRoza2 := DumyRoza2^.NextPnF;
            EnterDataFu (DumyRoza2,DumyFu2);
            ShowResult (DumyRoza2,9);
          end;
    until Upcase(ch) = 'Q' ;
    gotoXY(1,23); ClrEol;
    write('Print [LPT]   Graph on Screen (Y/N) : NO '); read(kbd,ch);
      if Upcase(ch) = 'Y' then
      begin
        ShowResult(DumyRoza2,9);
        EGAPrintScreen;
      end;
    PlotAxis(ZeroX,ZeroY,0);
    ShowResult(DumyRoza1,0);
    gotoXY(5,3);
    write('                                                  ');
    gotoXY(5,2);
    write('                                                  ');
  end; { if LoadData }
end; { Main_Roza }

begin { Compare }
  mark(HeapTop);
  gotoXY(1,22); write('Reff. Senece');
  LoadData;
```

```
  gotoXY(1,22); write('            ');
  gotoXY(18,22); write('1 : ',FileName);
  if LoadDataActive then
     Main_Roza ;
  gotoXY(18,22); write('                                          ');
  Release(HeapTop);
end; { Compare }
```

Program :
Programmer :

```
{ ****************************************************************
  *                                                              *
  *          PROCEDURE  INTERSETTION-COMPARE OBJECTS             *
  *                /* Accumulate angle */                        *
  *                                                              *
  *               CREATE  ( MM DD YY ) 3 14 88                   *
  *                                                              *
  *                        VERSION 2.4                           *
  *                                                              *
  *             BY    CHANCHAI PISITTIVITHAYANON                 *
  *                                                              *
  ****************************************************************}


Overlay procedure Segment_Convolution;
const
  ZeroX = 20 ;
  ZeroY = 250 ;
  OffsetPic_X1 = -30 ;
  OffsetPic_Y1 = -80 ;
  OffsetPic_X2 = 200 ;
  OffsetPic_Y2 = 50 ;
  SetErr       = 10 ; { 10 % }

type
  DataRecP  = ^DataRecPT;
  DataRecPT  = Record
                 DataI  : byte;
                 DataJ  : byte;
                 DataC  :  CodeT;
                 NextPnF : DataRecP;
                 NextPnB : DataRecP;
               end;
  DataRozaP = ^DataRoza;
  DataRoza  = Record
                RoSeg : real;
                ZaSeg : real;
                NumSet : byte;
                NextPnF : DataRozaP;
                NextPnB : DataRozaP;
```

```
              end;
  DataMapP  = ^DataMapT;
  DataMapT  = Record
                MapX : byte;
                MapY : byte;
                MapU : byte;
                MapV : byte;
                NextPnF : DataMapP;
                NextPnB : DataMapP;
              end;

var
  FirDataRec1,FirDataRec2,DumyDataRec,LastDataRec  : DataRecP;
  DataPic1,DataPic2 : DataRecP ;
  FirRoza1,FirRoza2,DumyRoza,LastRoza : DataRozaP;
  RefDataFu,ObjDataFu : DataRozaP ;
  DataRec : DataRecT;
  MapUV1,MapUV2 : DataRozaP ;
  FirDataMap,DumyDataMap,LastDataMap : DataMapP;
  FileVar : DataT;
  FileName : string[40];
  LoadDataActive : boolean;
  FileVar_Text : text ;
  FileName_Text : string[40] ;
  Status_ON : boolean ;


procedure ReadData(var FirDataRec : DataRecP);
var
  num : byte ;
  TemDataRec : DataRecP;
begin  { ReadData }
  num := 1 ;
  FirDataRec := nil ;
  while NOT EOF(FileVar) do
  begin
    read(FileVar,DataRec);
    New (DumyDataRec);
    DumyDataRec^.DataI := DataRec.DataI1;
    DumyDataRec^.DataJ := DataRec.DataJ1;
    DumyDataRec^.DataC := num ;
    if FirDataRec = nil then
      begin
        FirDataRec := DumyDataRec;
        TemDataRec := DumyDataRec;
      end
    else
      begin
        DumyDataRec^.NextPnB := TemDataRec ; .
```

```
        LastDataRec^.NextPnF := DumyDataRec;
        TemDataRec^          := DumyDataRec;
      end;
    LastDataRec := DumyDataRec;
    num := num + 1 ;
  end; { while }
  FirDataRec^.NextPn8  := nil ;
  LastDataRec^.NextPnF := FirDataRec ;
end; { ReadData }


procedure LoadData(var FirDataRec : DataRecP);
var
  Created    : boolean;
  OK         : char;
begin  { LoadData }
  repeat
    gotoXY(1,23); ClrEol;
    write('Load File name about X,Y : C for ROZA  ... ');
    readln(FileName);
    if FileName = '' then
      begin
        Created := true;
        LoadDataActive := false;
      end
    else  { NOT(FileName = '') }
      begin
        Assign(FileVar,FileName);
        {$I-}  reset(FileVar);  {$I+}
        Created := IOresult = 0 ;
        if Created then
          begin
            LoadDataActive := true;
            ReadData(FirDataRec);
            close(FileVar);
          end
        else  { not create }
          begin
            gotoXY(1,23); ClrEol;
            write(FileName,' File NOT Found  ,  Esc : Exit ');
            sound(2000); delay(500); nosound;
            read(Kbd,OK);
            if OK = #27 then
              begin
                Created := true;
                LoadDataActive := false ;
              end;
          end;
      end; { else not (FileName = '') }
```

```
  until Created ;
  sound(2000); delay(500); nosound;
end; { LoadData }

procedure ReadDataEdge(var Pic : DataRecP);
var
  TemDataRec : DataRecP;
  StrData : string[5] ;
  Data,ErrCode : integer ;
begin  { ReadDataEDge }
  Pic := nil ;
  while NOT EOF(FileVar_Text) do
  begin
    New (DumyDataRec);
    read(FileVar_Text,StrData) ;
    val(StrData,Data,ErrCode); DumyDataRec^.DataI := round(Data);
    readln(FileVar_Text,StrData) ;
    val(StrData,Data,ErrCode); DumyDataRec^.DataJ := round(Data);
    if Pic = nil then
      begin
        Pic := DumyDataRec;
        TemDataRec := DumyDataRec;
      end
    else
      begin
        DumyDataRec^.NextPnB := TemDataRec ;
        LastDataRec^.NextPnF := DumyDataRec;
        TemDataRec           := DumyDataRec;
      end;
    LastDataRec := DumyDataRec;
  end; { while }
  Pic^.NextPnB  := nil ;
  LastDataRec^.NextPnF := Pic ;
end; { ReadDataEdge }

procedure LoadEdge_Pic(var Pic : DataRecP);
var
  Created    : boolean;
  OK         : char;
begin  { LoadEdge_Pic }
  repeat
    gotoXY(1,23); ClrEol;
    write('Load New Data edge [ Filename.DE1 ] for Display  ... ');
    readln(Filename_Text);
    if (Filename_Text = '') then
      begin
        Created := true;
        LoadDataActive := false;
      end
```

```
    else  { NOT(Filename_Text = '') }
      begin
        Assign(FileVar_Text,Filename_Text);
        {$I-}  reset(FileVar_Text);  {$I+}
        Created := IOresult = 0 ;
        if Created then
          begin
            LoadDataActive := true;
            ReadDataEdge(Pic);
            close(FileVar_Text);
          end
        else  { not create }
          begin
            gotoXY(1,23); ClrEol;
            write(Filename_Text,' File NOT Found  ,  Esc : Exit ');
            sound(2000); delay(500); nosound;
            read(Kbd,OK);
            if OK = #27 then
              begin
                Created := true;
                LoadDataActive := false ;
              end;
          end;
      end; { else not (Filename_Text = '') }
  until Created ;
  sound(2000); delay(500); nosound;
end; { LoadEdge_Pic }

procedure Keep_XY_UV(XY1,XY2,UV1,UV2 : byte);
var
  TemDataMap : DataMapP;
begin  { Keep_XY_UV }
    New (DumyDataMap);
    DumyDataMap^.MapX := XY1 ;
    DumyDataMap^.MapY := XY2 ;
    DumyDataMap^.MapU := UV1 ;
    DumyDataMap^.MapV := UV2 ;
    if FirDataMap = nil then
      begin
        FirDataMap := DumyDataMap;
        TemDataMap := DumyDataMap;
      end
    else
      begin
        DumyDataMap^.NextPnB := TemDataMap ;
        LastDataMap^.NextPnF := DumyDataMap;
        TemDataMap           := DumyDataMap;
      end;
    LastDataMap := DumyDataMap;
```

```
end; { Keep_XY_UV }


procedure DisplayPic ( Pic : DataRecP ; Segment1,Segment2 : DataRozaP ;
             FirDataRec : DataRecP ; OffsetX,OffsetY,color : integer );
var
  DumyPic : DataRecP ;
  DumyRec : DataRecP ;
  x1,y1,x2,y2 : byte ;
begin { DisplayPic }
  DumyRec := FirDataRec ;
  while NOT (DumyRec^.DataC = Segment1^.NumSet) do
    DumyRec := DumyRec^.NextPnF ;
  x1 := DumyRec^.DataI ; y1 := DumyRec^.DataJ ;
  while NOT (DumyRec^.DataC = Segment2^.NumSet) do
    DumyRec := DumyRec^.NextPnF ;
  x2 := DumyRec^.DataI ; y2 := DumyRec^.DataJ ;
  DumyPic := Pic ;
  while NOT( (DumyPic^.DataI = x1) AND (DumyPic^.DataJ = y1) ) do
  begin
    DumyPic := DumyPic^.NextPnF ;
  end;
  repeat
    EGAPlot(DumyPic^.DataI+OffsetX,DumyPic^.DataJ+OffsetY,0,color);
    DumyPic := DumyPic^.NextPnF ;
  until (DumyPic^.DataI = x2) AND (DumyPic^.DataJ = y2);
end; { DisplayPic }

procedure ShowResultSegment(DumyRo1,DumyRo2 : DataRozaP ;
                            OffsetX,OffsetY : real ; Color :integer);
const
  ScalX = 1 ;
  ScalY = 0.15 ;
var
  TemX,TemY : integer;
  RoS,ZaS,TemRoS : integer;
  ZeroXPlusOffset,ZeroYPlusOffset : integer ;
begin { ShowResult }
  ZeroXPlusOffset := ZeroX + round(OffsetX*ScalX);
  ZeroYPlusOffset := ZeroY - round(OffsetY*ScalY);
  TemX := ZeroXPlusOffset ;
  TemY := ZeroYPlusOffset ;
  repeat
    RoS := round(DumyRo1^.RoSeg*ScalX);
    ZaS := ZeroYPlusOffset - round(DumyRo1^.ZaSeg*ScalY) ;
    EGADraw(TemX,ZaS,TemX,TemY,0,Color);
    TemY := ZaS;
    EGADraw(TemX,ZaS,RoS+TemX,ZaS,0,Color);
    TemX := TemX + RoS ;
```

```
    DumyRo1 := DumyRo1^.NextPnF;
  until DumyRo2 = DumyRo1;
end; { ShowResultSegment }


procedure Calculate_Roza(Dumy : DataRecP ; Minleng : byte ;
                         var RoS,ZaS : real ; var Jump : boolean );
                        { Ros in Lenght , ZaS in Degrees }
var
  x,y,x1,y1,x2,y2,x3,y3 : real;
  AA,BB,CC  : real;
begin { Calculate_Roza }
  Jump := false ;
  x1 := Dumy^.DataI ; y1 := Dumy^.DataJ ;
  Dumy := Dumy^.NextPnF;
  x2 := Dumy^.DataI ; y2 := Dumy^.DataJ ;
  Dumy := Dumy^.NextPnF;
  x3 := Dumy^.DataI ; y3 := Dumy^.DataJ ;
  if (sqrt(sqr(x2-x1)+sqr(y2-y1)) < MinLeng ) {OR (ZaS < 10 )} then
    begin     { First side = 3 pixel }
      x2 := x3 ;  y2 := y3 ;
      Dumy := Dumy^.NextPnF ;
      x3 := Dumy^.DataI ; y3 := Dumy^.DataJ ;
      Jump := true ;
    end;
  if (sqrt(sqr(x3-x2)+sqr(y3-y2)) < MinLeng ) {OR (ZaS < 10 )} then
    begin    {  second side = 3 pixel }
      Dumy := Dumy^.NextPnF ;
      x3 := Dumy^.DataI ; y3 := Dumy^.DataJ ;
    end;
  AA := sqr(x3-x2)+sqr(y3-y2) ;
  BB := sqr(x1-x3)+sqr(y1-y3) ;
  CC := sqr(x2-x1)+sqr(y2-y1) ;
  ZaS := ArcCos( (AA + CC - BB) / (2*sqrt(AA*CC)) );
  if ZaS > 90 then  ZaS := 180 - ZaS ;
  RoS := sqrt(CC) ;
end; { Calculate_Roza }

function Round_of_Object(RecDumy : DataRecP) : byte ;
var   { calculate percent of leng on object , 2.5 % }
  Dumy : DataRecP ;
  x1,y1,x2,y2,Leng : real ;
begin { Round_Of_Object }
  Leng := 0 ;
  Dumy := RecDumy ;
  repeat
    x1 := Dumy^.DataI ;
    y1 := Dumy^.DataJ ;
    Dumy := Dumy^.NextPnF ;
```

```
    x2 := Dumy^.DataI ;
    y2 := Dumy^.DataJ ;
    Leng := Leng + Sqrt(sqr(x2-x1)+sqr(y2-y1)) ;
  until Dumy = RecDumy ;
  if Leng > 250 then
    Round_Of_Object := 3  { for size 256*256 }
  else
    Round_Of_Object := 3 ; { for size less than 64 * 64 }
end; { Round_Of_Object }

procedure Transform_To_Roza(var FirRecDumy : DataRecP ;
                            var FirRoDumy : DataRozaP);
var
  TemRec : DataRecP;
  TemRo,LastRo,DumyRo : DataRozaP ;
  RoS,ZaS : real;
  Jump : boolean;
  MinLeng : byte ;
begin { Transform_To_Roza }
  FirRoDumy := nil;
  TemRec := FirRecDumy;
  MinLeng := Round_Of_Object(FirRecDumy);
  repeat
    Jump := false;
    Calculate_Roza(TemRec,MinLeng,RoS,ZaS,Jump);
    New (DumyRo);
    DumyRo^.RoSeg := RoS ;
    DumyRo^.ZaSeg := ZaS ;
    DumyRo^.NumSet := TemRec^.DataC ;
    if FirRoDumy = nil then
      begin
        FirRoDumy := DumyRo ;
        TemRo     := DumyRo ;
      end
    else
      begin
        DumyRo^.NextPnB := TemRo ;
        LastRo^.NextPnF := DumyRo ;
        TemRo           := DumyRo ;
      end;
    LastRo := DumyRo ;
    if Jump then
      begin
        TemRec := TemRec^.NextPnF;
      end;
    TemRec := TemRec^.NextPnF;
  until TemRec = FirRecDumy ;
  FirRoDumy^.NextPnB := LastRo ;
  LastRo^.NextPnF  := FirRoDumy ;
```

```
end; { Transform_To_Roza }

procedure NewReserveFu(Transive : DataRozaP ;
                       var Receive : DataRozaP );
var
  TemRo,LastRo,DumyRo : DataRozaP ;
  FirTr : DataRozaP ;
  TemZeta : real ;
begin { NewReserveFu }
  FirTr := Transive;
  Receive := nil ;
  TemZeta := 0;
  repeat
    new(DumyRo);
    DumyRo^.RoSeg := Transive^.RoSeg ;
    DumyRo^.ZaSeg := Transive^.ZaSeg ;
    DumyRo^.NumSet := Transive^.NumSet ;
    if Receive = nil then
      begin
        Receive := DumyRo ;
        TemRo   := DumyRo ;
      end
    else
      begin
        DumyRo^.NextPnB := TemRo;
        LastRo^.NextPnF := DumyRo;
        TemRo           := DumyRo;
      end;
    LastRo := DumyRo;
    Transive := Transive^.NextPnF;
  until Transive = FirTr;
  Receive^.NextPnB := LastRo;
  LastRo^.NextPnF := Receive ;
end; { NewReserveFu }

procedure ResetDataFu(Roza1,Roza2 : DataRozaP);
var
  Dumy1,Dumy2 : DataRozaP;
begin { ResetDataFu }
  Dumy1 := Roza1 ;
  Dumy2 := Roza2 ;
  repeat
    Dumy2^.RoSeg := Dumy1^.RoSeg ;
    Dumy2^.ZaSeg := Dumy1^.ZaSeg ;
    Dumy2^.NumSet := Dumy1^.NumSet ;
    Dumy2 := Dumy2^.NextPnF ;
    Dumy1 := Dumy1^.NextPnF ;
  until (Dumy1 = Roza1) AND (Dumy2 = Roza2) ;
end; { ResetDataFu }
```

```
procedure EnterDataFu(Roza1,Roza2 : DataRozaP);
var
  SumZaSeg : real ;
begin { EnterDataFu }
  SumZaSeg := 0 ;
  repeat
    SumZaSeg := SumZaSeg + Roza1^.ZaSeg ;
    Roza1^.ZaSeg := SumZaSeg ;
    Roza1 := Roza1^.NextPnF ;
  until Roza1 = Roza2 ;
end; { EnterDataFu }

procedure Matching_Check (Map_XY1,Map_XY2,Map_UV1,Map_UV2 : DataRozaP);
var
  DumyXY,DumyUV : DataRozaP ;
  OK1,OK2 : boolean ;
begin { Matching_Check }
  OK1 := false ;
  OK2 := false ;
  DumyXY := Map_XY1;
  DumyUV := Map_UV1;
  gotoXY(1,23); ClrEol;
  repeat
    gotoXY(1,23);
    write('Table_XY = ',DumyXY^.NumSet:4,' Table_UV = ',DumyUV^.NumSet:4);
    DumyXY := DumyXY^.NextPnF ;
    DumyUV := DumyUV^.NextPnF ;
    if DumyXY = Map_XY2 then OK1 := true ;
    if DumyUV = Map_UV2 then OK2 := true ;
    read(kbd,ch);
  until OK1 AND OK2 ;
end;  { Matching_Check }

procedure MatchingPosiXY_UV (Map_XY1,Map_XY2,Map_UV1,Map_UV2 : DataRozaP ;
                             Run_OK : boolean);
const
  Leng_Err = 10 ; { 10 % }
  Angl_Err = 15 ; { 15 % }
var
  DumyXY,DumyUV : DataRozaP ;
  DumyRecXY,DumyRecUV : DataRecP ;
  Leng,Angl     : real ;
  Table         : byte ;
begin { MatchingPosiXY_UV }
  if Run_OK then
  begin
    Matching_Check(Map_XY1,Map_XY2,Map_UV1,Map_UV2);
    DumyXY := Map_XY1 ;
```

```
    DumyUV := Map_UV1 ;
    repeat
      Leng := ABS(DumyXY^.RoSeg - DumyUV^.RoSeg)*100/DumyXY^.RoSeg ;
      Angl := ABS(DumyXY^.ZaSeg - DumyUV^.ZaSeg)*100/DumyXY^.ZaSeg ;
      if (Leng < Leng_Err) AND (Angl < Angl_Err) then
        begin
          Table := DumyXY^.NumSet;
          DumyRecXY := FirDataRec1 ;
          repeat
            DumyRecXY := DumyRecXY^.NextPnF ;
          until DumyRecXY^.DataC = Table ;
          Table := DumyUV^.NumSet ;
          DumyRecUV := FirDataRec2 ;
          repeat
            DumyRecUV := DumyRecUV^.NextPnF;
          until DumyRecUV^.DataC = Table;
          Keep_XY_UV (DumyRecXY^.DataI,DumyRecXY^.DataJ,
                      DumyRecUV^.DataI,DumyRecUV^.DataJ);
          DumyRecXY := DumyRecXY^.NextPnF ;
          DumyRecUV := DumyRecUV^.NextPnF ;
          Keep_XY_UV (DumyRecXY^.DataI,DumyRecXY^.DataJ,
                      DumyRecUV^.DataI,DumyRecUV^.DataJ);
        end;
      DumyXY := DumyXY^.NextPnF ;
      DumyUV := DumyUV^.NextPnF ;
    until (DumyXY = Map_XY2) OR (DumyUV = Map_UV2) ;
  end ; { if Run_OK }
end; { MatchingPosiXY_UV }

procedure ConvolutionSegment(Reff1,Reff2,ConV1 : DataRozaP);
var
  DumyRef,DumyRoza,DumyConV,DumyCo : DataRozaP ;
  TemShow : DataRozaP ;
  TemRoSeg,TemRoSegA,TemRoSegB : real ;
  DeltaZe,OffsetR,OffsetZ : real ;
  AreaRef,ErrArea,P_Err,MinErr : real ;
  Near_Endy : boolean ;

        procedure Sub_1 ;
        begin  { Sub_1 }
          ErrArea := 0 ;
          DumyConV := DumyCo ;
          DumyRef := Reff1 ;
          TemRoSegA := DumyRef^.RoSeg ;
          TemRoSegB := DumyConV^.RoSeg ;
        end; { Sub_1 }

        procedure Sub_2 ;
        begin  { Sub_2 }
```

```
          DisplayPic(DataPic2,DumyCo,DumyConV,FirDataRec2,
                     OffsetPic_X2,OffsetPic_Y2,12);
          gotoXY(50,16) ; write('ErrArea = ',ErrArea:4:2) ;
          P_Err := 100*ErrArea/AreaRef ;
          gotoXY(50,17) ; write('% Err   = ',P_Err:4:2);
          if P_Err < SetErr then
            begin
              sound(3000); delay(200); nosound;
              read(kbd,ch);
              if Upcase(ch) = 'P' then
                EGAPrintScreen;
            end ;
          if P_Err < MinErr then
            begin
              Status_ON := true ;
              MapUV1 := DumyCo ;
              MapUV2 := DumyConV ;
              MinErr := P_Err ;
            end;
          ShowResultSegment(Reff1,Reff2,OffSetR,OffSetZ,0);
          OffSetR := OffSetR + round(DumyCo^.RoSeg) ; {the round for display}
          OffSetZ := DumyCo^.ZaSeg ;
        end; { Sub_2 }

begin { ConvolutionSegment }
  EnterDataFu(Reff1,Reff2);
  AreaRef := 0 ;
  DumyRef := Reff1;
  repeat
    AreaRef := AreaRef + (DumyRef^.RoSeg * DumyRef^.ZaSeg) ;
    DumyRef := DumyRef^.NextPnF ;
  until DumyRef = Reff2 ;
  gotoXY(50,15); write('AreaRef. = ',AreaRef:3:2);
  MinErr := SetErr ;
  EnterDataFu(ConV1,ConV1);
  Near_Endy := false ;
  Status_ON := false ;
  DumyCo := ConV1 ;
  OffSetR := 0 ;
  OffSetZ := 0 ;
  repeat
    Sub_1 ;
    DisplayPic(DataPic2,ConV1,ConV1,FirDataRec2,OffsetPic_X2,OffsetPic_Y2,3);
    ShowResultSegment(ConV1,ConV1,0,0,3);
    ShowResultSegment(Reff1,Reff2,OffsetR,OffsetZ,12);
    repeat
      DeltaZe := ABS(DumyRef^.ZaSeg + OffSetZ - DumyConV^.ZaSeg);
      TemRoSeg := TemRoSegA - TemRoSegB ;
      if TemRoSeg >= 0 then
```

```
        begin { DumyRef^.RoSeg ) DumyConV^.RoSeg }
          ErrArea := ErrArea + (DeltaZe * TemRoSegB);
          TemRoSegA := TemRoSegA - TemRoSegB ;
          DumyConV := DumyConV^.NextPnF ;
          if DumyConV = ConV1 then Near_Endy := true ;
          TemRoSegB := DumyConV^.RoSeg ;
        end
      else
        begin { DumyRef^.RoSeg ( DumyConV^.RoSeg }
          ErrArea := ErrArea + (DeltaZe * TemRoSegA);
          TemRoSegB := TemRoSegB - TemRoSegA ;
          DumyRef := DumyRef^.NextPnF ;
          TemRoSegA := DumyRef^.RoSeg ;
        end;
    until  (DumyRef = Reff2) OR Near_Endy ;
    Sub_2 ;
    DumyCo := DumyCo^.NextPnF ;
  until Near_Endy ;
  DisplayPic(DataPic2,ConV1,ConV1,FirDataRec2,OffsetPic_X2,OffsetPic_Y2,0);
  ShowResultSegment(ConV1,ConV1,0,0,0);
     {  ***  NEAR_ENDY *** }
  ResetDataFu(FirRoza2,ConV1);
  DumyCo := DumyCo^.NextPnB ;
  EnterDataFu(DumyCo,DumyCo);
  TemShow := DumyCo ;
  Near_Endy := false ;
  OffsetR := 0 ;
  OffsetZ := 0 ;
  repeat
    Sub_1 ;
    DisplayPic(DataPic2,TemShow,TemShow,FirDataRec2,OffsetPic_X2,OffsetPic_Y2,3);
    ShowResultSegment(TemShow,TemShow,0,0,3);
    ShowResultSegment(Reff1,Reff2,OffsetR,OffsetZ,12);
    repeat
      DeltaZe := ABS(DumyRef^.ZaSeg + OffSetZ - DumyConV^.ZaSeg);
      TemRoSeg := TemRoSegA - TemRoSegB ;
      if TemRoSeg )= 0 then
        begin { DumyRef^.RoSeg ) DumyConV^.RoSeg }
          ErrArea := ErrArea + (DeltaZe * TemRoSegB);
          TemRoSegA := TemRoSegA - TemRoSegB ;
          DumyConV := DumyConV^.NextPnF ;
          TemRoSegB := DumyConV^.RoSeg ;
        end
      else
        begin { DumyRef^.RoSeg ( DumyConV^.RoSeg }
          ErrArea := ErrArea + (DeltaZe * TemRoSegA);
          TemRoSegB := TemRoSegB - TemRoSegA ;
          DumyRef := DumyRef^.NextPnF ;
          TemRoSegA := DumyRef^.RoSeg ;
```

```
        end;
    until  (DumyRef = Reff2) ;
    Sub_2 ;
    if DumyCo = ConV1 then Near_Endy := true ;
    DumyCo := DumyCo^.NextPnF ;
  until Near_Endy ;
  DisplayPic(DataPic2,TemShow,TemShow,FirDataRec2,OffsetPic_X2,OffsetPic_Y2,0);
  ShowResultSegment(TemShow,TemShow,0,0,0);
end; { ConvolutionSegment }

procedure INC(var Roza:DataRozaP; n:byte; Reff_Roza:DataRozaP;
              var Ok : boolean);
begin { INC }
  repeat
    Roza := Roza^.NextPnF ;
    if Roza = Reff_Roza then
      Ok := true ;
    n := n - 1 ;
  until n = 0 ;
end; { INC }


function Cal_30 ( Roza : DataRozaP ) : byte ;
const
  Percent_Segment = 30 ; { 30 % }
var
  Dumy : DataRozaP ;
  count : byte ;
begin  { Cal_30 }
  count := 0 ;
  Dumy := Roza ;
  repeat
    Dumy := Dumy^.NextPnF ;
    count := count + 1 ;
  until Dumy = Roza ;
  Cal_30 := round(count*Percent_Segment/100) ;
end;  { Cal_30 }

procedure OffsetXY(Roza : DataRozaP ; num : byte ; var X , Y : real);
begin { OffsetXY }
  repeat
    X := X + Roza^.RoSeg ;
    Y := Y + Roza^.ZaSeg ;
    num := num - 1 ;
    Roza := Roza^.NextPnF ;
  until num = 0 ;
end; { OffsetXY }
```

```
procedure SaveDataMap;
Var
     Created  : boolean;
     Filename : string[50];
     DataFile : text;
     ch       : char;

  procedure  WriteDataMap ;
  const
    NumStr = 4 ;
  var
    DumyMap : DataMapP ;
    DataStr : string[5];
    DataXYUV : string[20];
    a : byte ;

  begin { WriteDataMap }
    DumyMap := FirDataMap;
    repeat
      DataXYUV := '' ;
      str(DumyMap^.MapX,DataStr);
      DataXYUV := concat(DataXYUV,DataStr);
      for a := length(DataStr) to NumStr do
        DataXYUV := concat(DataXYUV,' ');
      str(DumyMap^.MapY,DataStr);
      DataXYUV := concat(DataXYUV,DataStr);
      for a := length(DataStr) to NumStr do
        DataXYUV := concat(DataXYUV,' ');
      str(DumyMap^.MapU,DataStr);
      DataXYUV := concat(DataXYUV,DataStr);
      for a := length(DataStr) to NumStr do
        DataXYUV := concat(DataXYUV,' ');
      str(DumyMap^.MapV,DataStr);
      DataXYUV := concat(DataXYUV,DataStr);
      writeln(DataFile,DataXYUV);
      DumyMap := DumyMap^.NextPnF ;
    until DumyMap = FirDataMap ;
  end;  { WriteDataMap }


Begin  { SaveDataMap }
  gotoXY(1,23); ClrEol;
  write ('Input file name for SAVE datamap [ .MAP ] ...');
  read (Filename);
  if NOT(Filename = '') then
    begin
      Assign(DataFile,Filename);
      {$I-} Reset(DataFile); {$I+}
```

```
      Created := IOresult = 0;
      if NOT Created then
        begin
          rewrite(DataFile);
          WriteDataMap;
          close(DataFile);
        end
      else  { #1 }
        begin
          gotoXY(1,23); ClrEol;
          sound(2000); delay(500); nosound;
          write(Filename,' Exist !    ENTER : OverWrite  ,  Esc : Quit ');
          repeat
            read(kbd,ch);
          until (ch = #$00) OR (ch = #27);
          if ch = #$0D then
            begin
              rewrite(DataFile);
              WriteDataMap;
              close(DataFile);
            end;
        end; { #1 }
    end; { if }
  sound(2000); delay(500); nosound;
end;  { SaveDataMap }


procedure Settion_Convolution(Reffer,Convo : DataRozaP ; NumINC : byte );
var
  Ref1,Ref2,Obj1 : DataRozaP ;
  DumyR,DumyC    : DataRozaP ;
  OffsetDispX,OffsetDispY : real ;
  Endy : boolean ;
begin { Settion_Convolution }
  Endy := false ;
  DumyR := Reffer ;
  DumyC := Convo ;
  OffsetDispX := 0 ;
  OffsetDispY := 0 ;
  FirDataMap := nil ;
  repeat
    DisplayPic(DataPic1,Reffer,Reffer,FirDataRec1,OffsetPic_X1,OffsetPic_Y1,12);
    EnterDataFu(Reffer,Reffer);
    ShowResultSegment(Reffer,Reffer,0,250,12);
    ResetDataFu(FirRoza1,Reffer);
    Ref1 := DumyR ;
    INC(DumyR,NumINC,Reffer,Endy);
    Ref2 := DumyR ;
    DisplayPic(DataPic1,Ref1,Ref2,FirDataRec1,OffsetPic_X1,OffsetPic_Y1,10);
```

```
    EnterDataFu(Ref1,Ref2);
    ShowResultSegment(Ref1,Ref2,0+OffsetDispX,250+OffsetDispY,10);
    ResetDataFu(FirRoza1,Reffer);
    ConvolutionSegment(Ref1,Ref2,DumyC);
    DisplayPic(DataPic1,Ref1,Ref2,FirDataRec1,OffsetPic_X1,OffsetPic_Y1,0);
    ShowResultSegment(Ref1,Ref2,0+OffsetDispX,250+OffsetDispY,0);
    ResetDataFu(FirRoza1,Reffer);
    ResetDataFu(FirRoza2,DumyC);
    MatchingPosiXY_UV(Ref1,Ref2,MapUV1,MapUV2,status_ON) ;
    OffsetXY(Ref1,NumINC,OffsetDispX,OffsetDispY) ;
  until Endy ;
  FirDataMap^.NextPnB  := nil ;
  LastDataMap^.NextPnF := FirDataMap ;
  DisplayPic(DataPic1,Reffer,Reffer,FirDataRec1,OffsetPic_X1,OffsetPic_Y1,0);
  EnterDataFu(Reffer,Reffer);
  ShowResultSegment(Reffer,Reffer,0,250,0);
  SaveDataMap;
end; { Settion_Convolution }


procedure Main_Settion_Convo;
var
  NumSegment : byte ;
begin { Main_Settion_Convo }
  Transform_To_Roza(FirDataRec1,FirRoza1);
  LoadEdge_Pic(DataPic1);
  gotoXY(1,20); write('Obj. Senece');
  LoadData(FirDataRec2) ;
  gotoXY(1,20); write('            ');
  if LoadDataActive then
  begin
    gotoXY(40,22); write('2 : ',FileName);
    Transform_To_Roza(FirDataRec2,FirRoza2);
    LoadEdge_Pic(DataPic2);
    Plotaxis(ZeroX,ZeroY,10);
    NewReserveFu(FirRoza1,RefDataFu);
    NewReserveFu(FirRoza2,ObjDataFu);
    NumSegMent := Cal_30(FirRoza1);
    Settion_Convolution(RefDataFu,ObjDataFu,NumSegment);
    Plotaxis(ZeroX,ZeroY,0);
  end; { if LoadDataActive }
end; { Main_Settion_Convo }


begin { Segment_Convolution }
  mark(HeapTop);
  gotoXY(1,20); sound(300); delay(100); nosound;
  write('Reff. Senece');
```

```
  LoadData(FirDataRec1);
  gotoXY(1,20); write('            ');
  gotoXY(18,22); write('1 : ',FileName);
  if LoadDataActive then
    Main_Settion_Convo;
  gotoXY(18,22); write('                                        ');
  release(HeapTop);
end; { Segment_Convolution }
```

File : MAP2.INC
Current Date : Tuesday June 28, 1988
Current Time : 2:53 AM

Program :
Programmer :

```
{ ****************************************************************
  *                                                              *
  *              PROCEDURE  GEOMETRIC CORECTION                  *
  *                                                              *
  *               CREATE  ( MM DD YY ) 4 10 88                   *
  *                                                              *
  *                        VERSION 1.0                           *
  *                                                              *
  *             BY    CHANCHAI PISITTIVITHAYANON                 *
  *                                                              *
  ****************************************************************}


Overlay procedure map ;
type
  MapP = ^MapT ;
  MapT = record
           dataX : integer ;
           dataY : integer ;
           dataU : integer ;
           dataV : integer ;
           NextPn8 : MapP ;
           NextPnF : MapP ;
         end;
  DataRecT = record
               DataI1 : byte ;
               DataJ1 : byte ;
               DataC1 : byte ;
             end;
var
  FirDataRec,LastDataRec : MapP ;
  FirMapxy : MapP ;
  DataMap : MapT ;
  CoeffiA0,CoeffiA1,CoeffiA2,CoeffiB0,CoeffiB1,CoeffiB2  : real ;
  filevar  : text ;
  filename : string[40] ;
  ch : char ;
  HTop : ^integer;
  DataRec : DataRecT;
  FilevarT : File of DataRecT;
```

```
  FilenameT : string[40] ;

procedure ReadDataMap(Var FirMap : MapP);
{ DATA FROM   :   XXXXXYYYYYUUUUUVVVVV }
var
  stg : string[5];
  tem : integer;
  LastMap,DumyMap,TemMap : MapP;
begin  { ReadDataMap }
  FirMap := nil ;
  while NOT EOF(FileVar) do
  begin
    read(FileVar,stg);
    val(stg,DataMap.DataX,tem);
    read(FileVar,stg);
    val(stg,DataMap.DataY,tem);
    read(FileVar,stg);
    val(stg,DataMap.DataU,tem);
    read(FileVar,stg);
    val(stg,DataMap.DataV,tem);
    readln(FileVar,stg);
    New (DumyMap);
    DumyMap^.DataX := DataMap.DataX;
    DumyMap^.DataY := DataMap.DataY;
    DumyMap^.DataU := DataMap.DataU;
    DumyMap^.DataV := DataMap.DataV;
    if FirMap = nil then
      begin
        FirMap := DumyMap;
        TemMap := DumyMap;
      end
    else
      begin
        DumyMap^.NextPnB := TemMap ;
        LastMap^.NextPnF := DumyMap;
        TemMap           := DumyMap;
      end;
    LastMap := DumyMap;
  end; { while }
  FirMap^.NextPnB  := nil ;
  LastMap^.NextPnF := FirMap ;
end; { ReadDataMap }


procedure EnterDataMap(var FirMap : MapP) ;
begin { EnterDataMap }
  repeat
    gotoXY(1,23); ClrEol;
    write('Enter Data File Name { .Map } '); read(filename);
```

```
    assign(filevar,filename);
    {$I-} reset(filevar); {$I+}
  until (IOresult = 0) OR (Filename = '') ;
  if NOT(filename = '') AND (IOresult = 0) then
     ReadDataMap(FirMap);
end; { EnterDataMap }

procedure Pre_Calculate (FirMap : MapP);
var
  Dumy : MapP ;
  sui,svi,sxi,syi,suui,svvi,suvi,suxi,svxi,suyi,svyi : real ;
  uu,vv,xx,yy : integer ;
  abs_SS , n : real ;
begin { Pre_Calculate }
  Dumy := FirMap ;
  n := 0 ;
  sui := 0 ;
  svi := 0 ;
  sxi := 0 ;
  syi := 0 ;
  suui := 0 ;
  svvi := 0 ;
  suvi := 0 ;
  suxi := 0 ;
  svxi := 0 ;
  suyi := 0 ;
  svyi := 0 ;
  abs_SS := 0 ;
  repeat
    n := n + 1 ;
    xx := Dumy^.DataX ;
    yy := Dumy^.DataY ;
    uu := Dumy^.DataU ;
    vv := Dumy^.DataV ;
    sui := sui + uu ;
    svi := svi + vv ;
    sxi := sxi + xx ;
    syi := syi + yy ;
    suui := suui + Sqr(uu) ;
    svvi := svvi + Sqr(vv) ;
    suvi := suvi + ( uu * vv ) ;
    suxi := suxi + ( uu * xx ) ;
    svxi := svxi + ( vv * xx ) ;
    suyi := suyi + ( uu * yy ) ;
    svyi := svyi + ( vv * yy ) ;
    Dumy := Dumy^.NextPnF ;
  until Dumy = FirMap ;
  sui := sui / n ;
  svi := svi / n ;
```

```
  sxi := sxi / n ;
  syi := syi / n ;
  suui := suui / n ;
  svvi := svvi / n ;
  suvi := suvi / n ;
  suxi := suxi / n ;
  svxi := svxi / n ;
  suyi := suyi / n ;
  svyi := svyi / n ;
  abs_SS := ( ( suui * svvi - sqr(suvi) ) +
           ( sui * ( svi * suvi - sui * svvi ) ) +
           ( svi * ( sui * suvi - svi * suui ) ) );
  CoeffiA0 := ( ( suui * svvi - sqr(suvi) ) * sxi +
              ( svi * suvi - sui * svvi ) * suxi +
              ( sui * suvi - svi * suui ) * svxi ) / abs_SS ;
  CoeffiA1 := ( ( svi * suvi - sui * svvi ) * sxi +
              ( svvi - sqr(svi) ) * suxi +
              ( sui * svi - suvi ) * svxi ) / abs_SS ;
  CoeffiA2 := ( ( sui * suvi - svi * suui ) * sxi +
              ( sui * svi  - suvi ) * suxi +
              ( suui - sqr(sui) ) * svxi ) / abs_SS ;
  CoeffiB0 := ( ( suui * svvi - sqr(suvi) ) * syi +
              ( svi * suvi - sui * svvi ) * suyi +
              ( sui * suvi - svi * suui ) * svyi ) / abs_SS ;
  CoeffiB1 := ( ( svi * suvi - sui * svvi ) * syi +
              ( svvi - sqr(svi) ) * suyi +
              ( sui * svi - suvi ) * svyi ) / abs_SS ;
  CoeffiB2 := ( ( sui * suvi - svi * suui ) * syi +
              ( sui * svi  - suvi ) * suyi +
              ( suui - sqr(sui) ) * svyi ) / abs_SS ;
end; { Pre_Calculate }


procedure ReadData;
var
  TemDataRec : MapP ;
  DumyDataRec : MapP ;
begin  { ReadData }
  FirDataRec := nil ;
  while NOT EOF(FileVarT) do
  begin
    read(FileVarT,DataRec);
    New (DumyDataRec);
    DumyDataRec^.DataX := DataRec.DataI1 ;
    DumyDataRec^.DataY := DataRec.DataJ1 ;
    DumyDataRec^.DataU := DataRec.DataC1 ;
    if FirDataRec = nil then
      begin
        FirDataRec := DumyDataRec;
```

```
        TemDataRec := DumyDataRec;
      end
    else
      begin
        DumyDataRec^.NextPnB := TemDataRec ;
        LastDataRec^.NextPnF := DumyDataRec;
        TemDataRec           := DumyDataRec;
      end;
    LastDataRec := DumyDataRec;
  end; { while }
  FirDataRec^.NextPnB  := nil ;
  LastDataRec^.NextPnF := FirDataRec ;
end; { ReadData }


procedure LoadData;
var
  Created    : boolean;
  OK         : char;
  LoadDataActive : boolean ;
begin  { LoadData }
  repeat
    gotoXY(1,23); ClrEol;
    write('Load File name about dataedge ... ');
    readln(FileNameT);
    if FileNameT = '' then
      begin
        Created := true;
        LoadDataActive := false;
      end
    else  { NOT(FileName = '') }
      begin
        Assign(FileVarT,FileNameT);
        {$I-}  reset(FileVarT);  {$I+}
        Created := IOresult = 0 ;
        if Created then
          begin
            LoadDataActive := true;
            ReadData;
            close(FileVarT);
          end
        else  { not create }
          begin
            gotoXY(1,23); ClrEol;
            write(FileName,' File NOT Found  ,  Esc : Exit ');
            sound(2000); delay(500); nosound;
            read(Kbd,OK);
            if OK = #27 then
              begin
```

```
              Created := true;
              LoadDataActive := false ;
            end;
        end;
    end; { else not (FileName = '') }
  until Created ;
  sound(2000); delay(500); nosound;
end; { LoadData }


procedure Map_Calculate ( FirMap : MapP );
var
  Dumy : MapP ;
  abs_SS : real ;
  TemU , TemV : real ;
  xx , yy : integer ;
begin
  Dumy := FirMap ;
  abs_SS := CoeffiA1 * CoeffiB2 - CoeffiA2 * CoeffiB1 ;
  repeat
    xx := Dumy^.DataX ;
    yy := Dumy^.DataY ;
    TemU := ( CoeffiB2*(xx-CoeffiA0) - CoeffiA2*(yy-CoeffiB0) ) / abs_SS ;
    TemV := ( CoeffiA1*(yy-CoeffiB0) - CoeffiB1*(xx-CoeffiA0) ) / abs_SS ;
    Dumy^.DataX := round(TemU) ;
    Dumy^.DataY := round(TemV) ;
    EGAPlot(Dumy^.DataX{+PosiI},Dumy^.DataY{+PosiJ},0,12);
    Dumy := Dumy^.NextPnF ;
  until Dumy = FirMap ;
end ; { Map_Calculate }


begin { Map }
  Mark(HTop);
  gotoXY(1,22); write('MAP Image ');
  EnterDataMap(FirMapXY);
  Pre_Calculate(FirMapXY);
  LoadData ;
  Map_Calculate(FirDataRec);
  read(kbd,ch);
  release(HTop);
end; { Map }
```

บทที่ 6
สรุปผลงานวิจัย

การหาขอบของภาพวัตถุในภาพสองระดับโดยใช้เทคนิคตารางหน้าต่างจะได้ขอบของภาพวัตถุบางที่สุด และมีลักษณะความต่อเนื่องของขอบเป็นแบบ 8-ทิศทาง และเทคนิคนี้จะง่ายต่อการนำไปเข้ารหัสของ Freeman ที่ทำเป็นรหัสลูกโซ่ สำหรับใช้ในการทำ Object Identification ด้วยวิธีสหสัมพันธ์หนึ่งมิติ (1 - dimensional correlation) ซึ่งทำให้การตรวจสอบวัตถุใช้เวลาน้อยลงมาก และจุดขอบของวัตถุ (Contour) ที่ได้มานั้นจะมีการตัดจุดบางจุดที่ไม่สำคัญออก ทำให้รหัสลูกโซ่ ของ Freeman สั้นลงและมีผลต่อการเพิ่มความเร็วให้กับการทำสหสัมพันธ์โดยตรง เมื่อเปรียบเทียบกับวิธีใน [2]

การตรวจสอบชนิดของวัตถุโดย การหาแนวแกนของภาพวัตถุแล้วทำการย้ายข้อมูลของภาพวัตถุแบบทุกจุดภาพนั้น จะสิ้นเปลืองเวลาที่ใช้กับการคำนวณสูง และผลจากการย้ายข้อมูลภาพจะเกิดจุดที่ขาดหายไปแบบเว้าแหว่งหรือโหว่ ซึ่งเกิดจากความเป็น Discrete ของจอภาพและของข้อมูลภาพเอง การชดเชยจุดข้อมูลที่หายไปนั้น จำเป็นต้องทำกับข้อมูลของภาพทั้งภาพ ซึ่งใช้เวลามาก และรหัสลูกโซ่ที่ได้มาจากการย้ายข้อมูลของภาพทั้งภาพนั้นยากต่อการทำสหสัมพันธ์ เนื่องจากข้อมูลของรหัสลูกโซ่ที่ได้มีจำนวนรหัสที่ผิดไปจากเดิมในการลดความผิดพลาดอันเนื่องจากสาเหตุดังกล่าวนี้ จึงได้ทำการพัฒนาวิธีการย้ายข้อมูลเฉพาะจุดที่สำคัญบางจุดเท่านั้น (จุดเปลี่ยนแนวเส้น) และใช้เทคนิคทางกราฟฟิกมาช่วยในการลากเส้นระหว่างจุดสำคัญบางจุดนี้ ทำให้สามารถเพิ่มความเร็วของการดำเนินการหมุนข้อมูลภาพและ ช่วยลดเวลาในการทำสหสัมพันธ์ของภาพวัตถุ ทำให้การตรวจสอบวัตถุทำได้อย่างมีประสิทธิภาพ

อย่างไรก็ตามการตรวจสอบชนิดของวัตถุ ด้วยการย้ายแนวแกนหลักของวัตถุ จะยังคงต้องใช้เวลาพอสมควร ซึ่งในที่สุดก็ได้พัฒนาวิธีใหม่ โดยใช้ข้อมูลของมุมและระยะทางมาช่วยในการตรวจสอบชนิดของวัตถุโดยการทำสหสัมพันธ์มิติเดียวของรหัสลูกโซ่แบบมุม-ระยะทาง ของวัตถุใดๆ กับวัตถุอ้างอิง รหัสลูกโซ่ดังกล่าวจะมีขนาดสั้นทำให้สามารถลดเวลาในการคำนวณเพื่อใช้ตรวจสอบชนิดของวัตถุ ทั้งยังทำให้ขนาดของหน่วยความจำที่ต้องใช้ก็จะลดลงไปด้วย เนื่องจากรหัสลูกโซ่แบบมุม-ระยะทางนี้จะเป็นแบบ shift around ปัญหาเกี่ยวกับการ rotation ของวัตถุจึงถูกกำจัดไปโดยปริยาย ส่วนการทำ accumulate ค่ามุมในรหัสลูกโซ่นี้เพื่อจะช่วยให้สามารถแยกแยะชนิดของวัตถุได้ง่าย ทั้งนี้เพราะถ้าวัตถุต่างชนิดกัน เมื่อนำกราฟรหัสลูกโซ่มาเปรียบเทียบกันแล้ว จะมองเห็นพื้นที่ error ได้

อย่างเด่นชัด

จากการนำเอาการทำสหสัมพันธ์มิติเดียวของรหัสลูกโซ่แบบมุม-ระยะทางไปประยุกต์ใช้ในงานการตรวจสอบวัตถุที่มีบางส่วนถูกบดบังนั้น จะเห็นผลลัพธ์การตรวจสอบทำได้อย่างมีประสิทธิภาพ โดยที่การทำสหสัมพันธ์มิติเดียวนั้นได้ใช้ความยาวของรหัสลูกโซ่เพียง 30 % ก็สามารถทำการตรวจสอบวัตถุได้อย่างถูกต้อง

การวิจัยของวิทยานิพนธ์ฉบับนี้ ได้เขียนโปรแกรมในการทำงานเป็นภาษา PASCAL บนเครื่องไมโครคอมพิวเตอร์ IBM PC/XT สำหรับการประยุกต์ในโรงงานอุตสาหกรรมที่มีหุ่นยนต์เป็นตัวทำงานนั้น สามารถใช้เครื่องไมโครคอมพิวเตอร์ขนาดเล็กได้ โดยเขียนโปรแกรมในการตัดสินใจต่าง ๆ ให้อยู่ในรูปของภาษาเครื่อง (Machine language) ก็จะทำให้การตรวจสอบวัตถุโดยหุ่นยนต์เป็นไปได้อย่างรวดเร็ว

ข้อเสนอแนะ

ดังได้กล่าวมาแล้วในบทนำว่าวิทยานิพนธ์ฉบับนี้เสนอเทคนิคการจดจำ และตรวจสอบชนิดของวัตถุ โดยมีข้อจำกัดคือระยะของวัตถุแต่ละชิ้นที่จะนำมาตรวจสอบจะต้องว่างห่างจากกล้องถ่ายภาพเท่าๆกัน ดังนั้นถ้าหากผู้ใดสนใจที่จะทำการวิจัยต่อจากวิทยานิพนธ์ฉบับนี้ ก็สามารถทำได้ด้วยการหาเทคนิคในการจำแนกชนิดของวัตถุ โดยยอมให้ระยะทางจากวัตถุต่างๆห่างจากกล้องถ่ายภาพเท่าใดก็ได้ อีกประการหนึ่งที่น่าจะทำการวิจัยต่อก็คือการหาพารามิเตอร์ใดๆมาประมาณส่วนโค้งของวัตถุแทนเส้นตรง ทั้งนี้เพราะส่วนโค้งหนึ่งๆต้องใช้เส้นตรงมาประมาณหลายเส้นด้วยกัน แต่ถ้าหากสามารถใช้พารามิเตอร์ส่วนโค้งเพียง 2 ตัว คือรัศมีของส่วนโค้ง กับ ความยาวของส่วนโค้ง (arc) แล้วจะได้รหัสลูกโซ่ที่สั้นขึ้น ทำให้สามารถลดเวลาในการคำนวณลงได้อีก

หนังสืออ้างอิง

(REFERENCES)

[1] วชิระ ยองใย, ฟูศักดิ์ ชีวสุวิทย์, " เธร็ชโฮลด์ที่ปรับค่าได้อัตโนมัติแบบง่ายสำหรับงานการตรวจหาขอบวัตถุ (A simple auto-adaptive threshold in edge detection) ", การประชุมทางวิชาการวิศวกรรมไฟฟ้า สถาบันอุดมศึกษาแห่งประเทศไทย ครั้งที่ 9, มหาวิทยาลัยขอนแก่น, หน้า 1-13-1 - 1-13-8, เล่มที่ 1, ธันวาคม 2529.

[2] ฟูศักดิ์ ชีวสุวิทย์ " วิธีการตรวจจับขอบวัตถุของหุ่นยนต์ (Edge detection method for robot) ", การประชุมวิชาการ สถิติประยุกต์ ครั้งที่ 5 , คณะสถิติประยุกต์ สถาบันบัณฑิตพัฒนบริหารศาสตร์, หน้า C03-1 - C03-10, 23-24 พฤษภาคม 2528.

[3] ชาญชัย พิสิทธิวิทยานนท์, ฟูศักดิ์ ชีวสุวิทย์, กิตติ ตีรเศรษฐ, " การติดตามขอบวัตถุด้วยการใช้ตารางหน้าต่าง (Contour Following by Window method)", การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 10 คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, หน้า 2-148 - 2-157 เล่มที่ 2, 24-25 พฤศจิกายน 2530.

[4] สุชาติ ประสิทธิรัฐสินธุ์ , ลัดดาวัลย์ รอดมณี "เทคนิคการวิเคราะห์ตัวแปรหลายตัว สำหรับการวิจัยทางสังคมศาสตร์ ",พิมพ์ที่ ห้างหุ้นส่วนจำกัดภาพพิมพ์ 187/25 บางยี่ขัน กรุงเทพฯ, สิงหาคม 2527.

[5] จเร สุรวัฒน์ปัญญา, " NUMERICAL COMPUTATION ด้วยโปรแกรมภาษาเบสิค ", พิมพ์ครั้งที่ 1 จำนวน 200 เล่ม, กันยายน 2528.

[6] CHAN S.PARK, " Interactive Microcomputer Graphics ", Addision - Wesley Publising Company, 1985.

[7] K.S.Fu , R.C.Gonzalez , C.S.G.Lee "Robotics:Control, Sensing,Vision,and Intellingence ", McGraw-Hill Book Company , Singapore , 1987.

[8] R.O. DUDA and P.E. HART , " Pattern Classificate and Scene Analysis ", John Wiley & Sons, New York , 1973.

[9] ชาญชัย พิสิทธิวิทยานนท์, ธนิตย์ ตีรสุวรรณวัตน์, ฟูศักดิ์ ชีวสุวิทย์, " สหสัมพันธ์ด้วยรหัสลูกโซ่สำหรับการตรวจสอบวัตถุ (Chain code correlation for object identification) ", การประชุมวิชาการ สถิติประยุกต์ ครั้งที่ 7 , คณะสถิติประยุกต์ สถาบันบัณฑิตพัฒนบริหารศาสตร์, หน้า 378 - 390, 19-20 พฤษภาคม 2531.

[10] William M.Newman and Robert F.Sproull, " Principles Interactive Computer Graphics ", McGRAW-Hill Book Company, Japan, 1984.

[11] John A.Richards, " Remote Sensing Digital Image Analysis : An Introduction ", Springer-Verlag Berlin Heidelberg New York, 1986.

[12] C.T. HELMERS , " Robotics Age : In the beginning ", Hayden Book Company , INC., Hasbrouck Heights , New Jersey , 1983.

[13] J.C. SIMON and R.M. HARALICK , " Digital Image Processing ", D. Reidel Publishing Company , 1981.

ภาคผนวก 1

```
Text File List Processing Program V4.00C Page : 1
File : COLAT.PAS
Current Date : Tuesday June 28, 1988
Current Time : 2:08 AM

Program :
Programmer :

{ ****************************************************************
  *                                                              *
  *            PROGRAM  OBJECTS RECOGNITION                      *
  *                                                              *
  *            CREATE  ( MM DD YY ) 8 10 87                      *
  *                                                              *
  *                  VERSION 1.5 A                               *
  *                                                              *
  *          BY    CHANCHAI PISITTIVITHAYANON                    *
  *                                                              *
  ****************************************************************}

PROGRAM OBJECTS_RECOGNITION ;

CONST
  Esc   =  #27;
  Enter = #$0D;
  GBH   =  49;
  GBL   =  48;
  X_Max  =  256 ;
  Y_Max  =  256 ;
  Gray_Max = 255 ;
  SizePix  = 4;
  Tone    = 0;  { Set Page(0) }
  Scal = 80;

TYPE
  ArrayPnT    =  ^ArrayPoint;
  ArrayPoint  =  record
                   ArrayP : array [1..Y_Max] of byte;
                   NextAr : ArrayPnT;
                 end;
  DataPnT     =  ^DataPoint;
  DataPoint   =  record
                   DataImage : array [1..X_Max] of ArrayPnT;
                   NextDa : DataPnT;
                 end;
  FileT       =  Text;
  FileBT      =  file of byte;
  NewFileT    =  file of DataPnT;
```

```
  FormatL     =  array [0..255] of byte;
  String50    =  String[50];
  CodeT       =  0..8;
  DataRecT    = Record
                  DataI1 : byte;
                  DataJ1 : byte;
                  DataC1 : CodeT;
                end;
  DataT       = FILE OF DataRecT;

VAR
  FileVar     :  FileT ;
  FileName    :  String50;
  DataDig     :  DataPnT;
  BinPict     :  DataPnT;
  ResultB     :  DataPnT;
  Mean_X,Mean_Y,Zeta :  Real;
  Border      :  byte;
  Line_X      :  integer;
  Line_Y      :  integer;
  PosiI,PosiJ :  integer;
  arrayTem    :  array [0..Gray_Max] of integer;
  ch          :  char;
  HeapTop     :  Integer;

{$I EGASET.INC }
{$I START1.INC }
{$I START2.INC }
{$I EIGEN.INC  }
{$I DETEDGE.INC}
{$I SMOOTH.INC }
{$I ENCODER.INC}
{$I EDLINE.INC }
{$I ROZA2.INC  }
{$I SETTION.INC}
{$I MAP2.INC   }

procedure Sub_Selet_Scene;
begin
  gotoXY(1,23); ClrEol;
  write('SCENE :: T:Threshold  N:New scene  C:Create  P:Print LPT  Esc:Menu ');
  read(kbd,ch);
  while NOT (UpCase(ch) = Esc) do
  begin
    case UpCase(ch) of
     'N' :  begin
              Read_Data_File(DataDig);
              Display_Scene;
            end;
```

```
      'C' :  CreatePictureOnScreen(PosiI,PosiJ);
      'T' :  Display_Scene;
      'P' :  EGAPrintScreen;
      end; { case }
     gotoXY(1,23); ClrEol;
     write('SCENE :: T:Threshold  N:New scene  C:Create  P:Print LPT  Esc:Menu ');
     read(kbd,ch);
   end; { while }
end; { Sub_Selet_Scene }

procedure Sub_Selet_Histogram;
begin { Sub_Selet_Histogram }
  gotoXY(1,23); ClrEol;
  write('Histogram :: S:Show  D:Delete  G:Graph  Esc:Menu ');
  read(kbd,ch);
  while NOT (UpCase(ch) = Esc) do
  begin
    case UpCase(ch) of
      'S' : DisplayHis;
      'D' : begin
              DeleteHis;
              DeleteScaleHis;
            end;
      'G' : Command_Graph;
    end; { case }
    gotoXY(1,23); ClrEol;
    write('Histogram :: S:Show  D:Delete  G:Graph  Esc:Menu ');
    read(kbd,ch);
  end; { while }
end; { Sub_Selet_Histogram }

procedure Sub_Selet_Window;
begin
  gotoXY(1,23); ClrEol;
  write('WINDOW :: M:Move  Z:Zoom  C:Cancel zoom  D:Delete  S:Size  Esc:Menu');
  read(kbd,ch);
  while NOT (UpCase(ch) = Esc) do
  begin
    case UpCase(ch) of
      'M' : MoveWindow(PosiI,PosiJ);
      'Z' : begin
              ExpressWindow(PosiI,PosiJ,true);
              CreatePictureOnScreen(PosiI,PosiJ);
            end;
      'C' : ExpressWindow(PosiI,PosiJ,false);
      'D' : DeleteDataInWindow(PosiI,PosiJ);
      'S' : SetBorderSize(Border);
    end;
    gotoXY(1,23); ClrEol;
```

```
    write('WINDOW :: M:Move  Z:Zoom  C:Cancel zoom  D:Delete  S:Size  Esc:Menu ');
    read(kbd,ch);
  end;
end; { Sub_Selet_Window }

procedure Sub_Selet_Process;
begin
  gotoXY(1,23); ClrEol;
  write('PROCESS :: R:Rotation  S:Smoothing  C:Contour  E:Encode  Esc:Menu ');
  read(kbd,ch);
  while NOT (UpCase(ch) = Esc) do
  begin
    case UpCase(ch) of
     'R' : Module_Rotate_Object;
     'S' : Smoothing_Binary_Images(DataDig);
     'C' : Contour_Following(DataDig);
     'E' : CodeBox;
    end; { case }
    gotoXY(1,23); ClrEol;
    write('PROCESS :: R:Rotation  S:Smoothing  C:Contour  E:Encode  Esc:Menu ');
    read(kbd,ch);
  end; { while }
end; { Sub_Selet_Process }

procedure Next_Menu;
begin
  gotoXY(1,23); ClrEol;
  write('NEXT-MENU :: E:Edline  R:RoZa  S:Seg_Map  M:Map  Esc:Menu ');
  read(kbd,ch);
  while NOT (UpCase(ch) = Esc) do
  begin
    case UpCase(ch) of
     'R' : Compare;
     'S' : Segment_Convolution;
     'E' : Create_Line;
     'M' : Map;
    end; { case }
    gotoXY(1,23); ClrEol;
    write('NEXT-MENU :: E:Edline  R:RoZa  S:Seg_Map  M:Map  Esc:Menu ');
    read(kbd,ch);
  end; { while }
end; { Sub_Selet_Histogram }


procedure Selet_Modu;
begin
  PosiI := 0; PosiJ := 0;
  gotoXY(1,23); ClrEol;
  write('MENU :: M:next Menu  H:Histogram  S:Scene  W:window  P:Process  Q:Quit ');
```

```
  read(kbd,ch);
  while NOT (UpCase(ch) = 'Q') do
  begin
    case UpCase(ch) of
     'M' : Next_Menu;
     'H' : Sub_Selet_Histogram;
     'S' : Sub_Selet_Scene;
     'W' : Sub_Selet_Window;
     'P' : Sub_Selet_Process;
    end; { case }
    gotoXY(1,23); ClrEOL;
    write('MENU :: M:next Menu  H:Histogram  S:Scene  W:window  P:Process  Q:Quit ');
    read(kbd,ch);
  end; { while }
end; { Selet_Modu }


begin { Main Program }
  Mark (HeapTop);
  Start;
  Selet_Modu;
  TextMode;
  release (HeapTop);
end.
```

File : EGASET.INC
Current Date : Tuesday June 28, 1988
Current Time : 2:22 AM

Program :
Programmer :

```
{*****************************************************************
 *                                                               *
 *                  Module : EGA Setting                         *
 *               Version 1.20C : July 2 1987                     *
 *              Produced by : INS.Research Team                  *
 *             Programmer :  Phakorn Hutasangkas.                *
 *                        :  P.Chanchai                          *
 *                                                               *
 *****************************************************************}

{ Direction to use :
  Procedure --) EGAGraphColorMode;        (-- Switch to EGA Mode
                EGAPalette (PaletteNumber);      (-- Set Palette
                EGAPlot (PageNumber,x,y,ColorNumber);   (-- Plot
                             at Page, and x,y with ColorNumber}

type
  _RegType = record
               case integer of
                 1 : (AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags : integer);
                 2 : (AL,AH,BL,BH,CL,CH,DL,DH          : byte);
               end;

var
  _Regs    : _RegType;

procedure EGAGraphColorMode;
begin { EGAGraphColorMode }
  _Regs.AX := $0010;          { Hi-Res 640 X 350 }
  Intr ($10,_Regs);
end { EGAGraphColorMode };

procedure EGAPalette (PaletteNumber,ColorValue : byte);
begin { EGAPalette }
  _Regs.AH := $0B;                          { Set Color Palette }
  _Regs.BH := PaletteNumber and $0F;   { Palette Color (0..127) }
  _Regs.BL := ColorValue;                        { Color Value }
  Intr ($10,_Regs);
end { EGAPalette };
```

```
procedure EGAPlot (x,y,Page,Color : integer);
begin { EGAPlot }
  _Regs.BH := Page;                          { Page number  }
  _Regs.CX := x;                             { Column number }
  _Regs.DX := y;                             { Row number    }
  _Regs.AL := Color;                         { Color number  }
  _Regs.AH := $0C;                           { Write Dot     }
  Intr ($10,_Regs);
end { EGAPlot };

procedure EGADraw (x0,y0,x1,y1,Page,Color : integer);
var
  ErrorTerm,HalfX,HalfY : integer;
  x,y,DeltaX,DeltaY,Xstep,Ystep : integer;
begin { EGADraw }
  x := X0; y := Y0;
  Xstep := 1; Ystep := 1;
  if X0 > X1 then Xstep := -1;
  if Y0 > Y1 then Ystep := -1;
  DeltaX := abs(X1-X0);
  DeltaY := abs(Y1-Y0);
  ErrorTerm := 0;
  if NOT ((x0=x1) AND (y0=y1)) then
     if DeltaX > DeltaY then
       begin
         HalfX := trunc(DeltaX/2);
         repeat
           EGAPlot(x,y,Page,Color);
           x := x + Xstep;
           ErrorTerm := ErrorTerm + DeltaY;
           if NOT(ErrorTerm <= HalfX) then
             begin
               ErrorTerm := ErrorTerm - DeltaX;
               y := y + Ystep;
             end;
         until (x=x1) and (y = y1);
       end
     else
       begin
         HalfY := trunc(DeltaY/2);
         repeat
           EGAPlot(x,y,Page,Color);
           y := y + Ystep;
           ErrorTerm := ErrorTerm + Deltax;
           if Not(ErrorTerm <= HalfY) then
             begin
               ErrorTerm := ErrorTerm - DeltaY;
               x := x + Xstep;
             end;
```

```
        until (x = x1) and (y = y1);
      end;
  EGAPlot(x1,y1,Page,Color);
end; { EGADraw }

procedure EGADrawBorder(x1,y1,x2,y2,page,color:integer);
begin
  EGADraw(x1,y1,x1,y2,page,color);
  EGADraw(x1,y1,x2,y1,page,color);
  EGADraw(x2,y2,x2,y1,page,color);
  EGADraw(x2,y2,x1,y2,page,color);
end; { EGADrawBorder }

function  EGAGetdot ( X,Y,Page : integer ) : byte;
begin
  _Regs.BH := Page;
  _Regs.DX := Y;
  _Regs.CX := X;
  _Regs.AH := $0D;
  Intr($10,_Regs);
  EGAGetdot := _Regs.AL;
end; { EGAGetdot }

procedure EGACircle (x,y,Radius,Page,Color : integer);
const n = 14;
type Circ = array [1..n] of integer;
const xx : Circ = (0,121,239,355,465,568,663,749,823,885,935,971,993,1000);
var xk1,xk2,yk1,yk2,xp1,yp1,xp2,yp2 : integer;
    xfact,yfact : real;
    i : integer;

begin { EGACircle }
  xfact := abs(Radius * 0.001);
  yfact := xfact * 0.8;       { Circle Ratio 4:5 }
  if xfact ) 0.0 then
  begin
    xk1 := trunc(xx[1] * xfact + 0.5);
    yk1 := trunc(xx[n] * yfact + 0.5);
    for i := 2 to n do
    begin
      xk2 := trunc(xx[i] * xfact + 0.5);
      yk2 := trunc(xx[n-i+1] * yfact + 0.5);
      xp1 := x - xk1;
      yp1 := y + yk1;
      xp2 := x - xk2;
      yp2 := y + yk2;
      EGADraw(xp1,yp1,xp2,yp2,Page,Color);
      xp1 := x + xk1;
      xp2 := x + xk2;
```

```
      EGADraw(xp1,yp1,xp2,yp2,Page,Color);
      yp1 := y - yk1;
      yp2 := y - yk2;
      EGADraw(xp1,yp1+1,xp2,yp2+1,Page,Color);
      xp1 := x - xk1;
      xp2 := x - xk2;
      EGADraw(xp1,yp1+1,xp2,yp2+1,Page,Color);
      xk1 := xk2;
      yk1 := yk2;
    end;
  end
  else EGAPlot(x,y,Page,Color);
end { EGACircle };

procedure EGAElispe (x,y,Radius,Page,Color : integer;
                     Ratio : real);
const n = 14;
type Circ = array [1..n] of integer;
const xx : Circ = (0,121,239,355,465,568,663,749,823,885,935,971,993,1000);
var xk1,xk2,yk1,yk2,xp1,yp1,xp2,yp2 : integer;
    xfact,yfact : real;
    i : integer;

begin { EGAElispe }
  xfact := abs(Radius * 0.001);
  yfact := xfact * Ratio;
  if xfact ) 0.0 then
  begin
    xk1 := trunc(xx[1] * xfact + 0.5);
    yk1 := trunc(xx[n] * yfact + 0.5);
    for i := 2 to n do
    begin
      xk2 := trunc(xx[i] * xfact + 0.5);
      yk2 := trunc(xx[n-i+1] * yfact + 0.5);
      xp1 := x - xk1;
      yp1 := y + yk1;
      xp2 := x - xk2;
      yp2 := y + yk2;
      EGADraw(xp1,yp1,xp2,yp2,Page,Color);
      xp1 := x + xk1;
      xp2 := x + xk2;
      EGADraw(xp1,yp1,xp2,yp2,Page,Color);
      yp1 := y - yk1;
      yp2 := y - yk2;
      EGADraw(xp1,yp1+1,xp2,yp2+1,Page,Color);
      xp1 := x - xk1;
      xp2 := x - xk2;
      EGADraw(xp1,yp1+1,xp2,yp2+1,Page,Color);
      xk1 := xk2;
```

```
      yk1 := yk2;
    end;
  end
  else EGAPlot(x,y,Page,Color);
end { EGAElispe };


procedure EGAToneColor ( x1,y1,x2,y2,page : integer );
var
  rangeX,rangeY : integer;
  chanal,tem,color : integer;
begin
  rangeX := abs(x1-x2);
  rangeY := abs(y1-y2);
  if rangeX < rangeY then
    begin
      chanal := trunc(rangeY/16);
      for color := 0 to 15 do
      begin
        for tem := y1 to y1+chanal do
        begin
          EGADraw(x1,tem,x2,tem,page,color);
        end;
        y1 := y1 + chanal ;
      end;
    end
  else     { rangeY < rangeX }
    begin
      chanal := trunc(rangeX/16);
      for color := 0 to 15 do
      begin
        for tem := x1 to x1+chanal do
        begin
          EGADraw(tem,y1,tem,y2,page,color);
        end;
        x1 := x1 + chanal;
      end;
    end;
end; { EGAToneColor }

procedure EGAPrintScreen;
const
  Esc = #27;
  NumData = 1920;
var
  x,y : integer;
  Data : array [1..3] of byte;
  Line,SubLine : integer;
```

```
begin { PrintScreen }
  Write (Lst,Esc,'3',Chr(24));
  y := 0;
  for Line := 0 to (350 div 8) do
  begin
    Writeln (Lst);
    Write (Lst,Esc,'*',#3,Chr(NumData mod 256),Chr(NumData div 256));
    for x := 0 to 639 do
    begin
      Data[1] := $00;
      Data[2] := $00;
      Data[3] := $00;
      for SubLine := 0 to 7 do
      begin
        Data[1] := Data[1] shl 1;
        Data[2] := Data[2] shl 1;
        Data[3] := Data[3] shl 1;
        case EGAGetdot (x,y + SubLine,0) of
          1..5 : begin
                   Data[1] := Data[1] or $00;
                   Data[2] := Data[2] or $01;
                   Data[3] := Data[3] or $00;
                 end;
          6..10: begin
                   Data[1] := Data[1] or $01;
                   Data[2] := Data[2] or $00;
                   Data[3] := Data[3] or $01;
                 end;
         11..15: begin
                   Data[1] := Data[1] or $01;
                   Data[2] := Data[2] or $01;
                   Data[3] := Data[3] or $01;
                 end;
        end;
      end;
      Write (Lst,Chr(Data[1]));
      Write (Lst,Chr(Data[2]));
      Write (Lst,Chr(Data[3]));
    end;
    y := y + 8;
  end;
  Write (Lst,Esc,'2');
end { PrintScreen };

procedure PlotAxis(ZeroX,ZeroY,Color:integer);
var x,y,TemColor : integer;
begin { PlotAxis }
  if Color = 0 then
    TemColor := 0
```

```
  else
    TemColor := 3;
  EgaDraw (ZeroX,10,ZeroX,180,0,color); { Y-Axis }
  EgaDraw (10,ZeroY,600,ZeroY,0,color); { X-Axis }
  x := 10;
  while x < 600 do
  begin
    EgaDraw (x,ZeroY-1,x,ZeroY+1,0,TemColor);
    x := x + 20;
  end;
  y := 10;
  while y < 180 do
  begin
    EgaDraw (ZeroX-1,y,ZeroX+1,y,0,TemColor);
    y := y + 20;
  end;
end { PlotAxis };
```

File : START1.INC
Current Date : Tuesday June 28, 1988
Current Time : 2:23 AM

Program :
Programmer :

```
{*****************************************************************
 *                                                               *
 *            SEVERAL PROCEDURE FOR PROGRAM COLAT.PAS            *
 *                                                               *
 *                          PART ONE                             *
 *                                                               *
 *                        VERSION 1.3 A                          *
 *                                                               *
 *              BY    CHANCHAI PISITTIVITHAYANON                 *
 *                                                               *
 *****************************************************************}

procedure Clear_Var;
var a : integer;
begin  { Clear_Var }
  for a := 0 to Gray_Max do
    arrayTem[a] := 0;
end;  { Clear_Var }

procedure NewPointer;
begin
  PosiI := 0; PosiJ := 0 ;
  New (DataDig);
  DataDig^.NextDa := nil;
  New (BinPict);
  BinPict^.NextDa := nil;
  for Line_X := 1 to Y_Max do
  begin
    New(DataDig^.DataImage[Line_X]);
    DataDig^.DataImage[Line_X]^.NextAr := nil ;
    New(BinPict^.DataImage[Line_X]);
    BinPict^.DataImage[Line_X]^.NextAr := nil ;
    for Line_Y := 1 to X_Max do
    begin
      DataDig^.DataImage[Line_X]^.ArrayP[Line_Y]  := 0 ;
      BinPict^.DataImage[Line_X]^.ArrayP[Line_Y]  := 0 ;
    end;
  end;
end;  { NewPointer }
```

```
procedure Wait_Message;
begin
  gotoXY(1,23); ClrEol;
  write('Please Wait');
end;

procedure Initialize_Start;
begin
  Border := 60;
  EGAGraphColorMode;
  NewPointer;
  EGAPalette(tone,0);  { Set palette(0) }
end;  { Initialize }

procedure SaveNewFile(var BiFileVar : FileBT ;
                      var BiPicture : DataPnT);
begin { SaveNewFile }
  Wait_Message;
  for Line_Y := 1 to Y_Max do
    for Line_X := 1 to X_Max do
      write (BiFileVar,BiPicture^.DataImage[Line_X]^.ArrayP[Line_Y]);
  close(BiFileVar);
end; { SaveNewFile }

procedure OpenSaveFile(var BiFileVar:FileBT;var BiFileName:String50 );
var
  CreateFile : boolean;
begin { OpenSaveFile }
  Assign (BiFileVar,BiFileName);
  {$I-} Reset(BifileVar) {$I+};
  CreateFile := IOresult = 0 ;
  if NOT CreateFile then
    rewrite(BiFileVar)
  else
    begin
      gotoXY (1,23); ClrEOL;
      write(BiFileName,' Exist : Reture to OverWRITE  : Esc to Quit ');
      repeat
        read(kbd,ch);
      until (ch = #$0D) OR (ch = #27);
      if ch = #$0D then
        rewrite(BiFileVar);
    end;
end;   { OpenSaveFile }

procedure CreateFileBinaryPicture(BiPicture : DataPnT);
var
  BiFileVar  : FileBT;
  BiFileName : String50;
```

```
begin  { CreateFileBinaryPicture }
  gotoXY (1,23); ClrEOL ;
  write (' Enter Filename',#39,'s Binary for SAVE ...');
  read (BiFileName);
  if NOT(BiFileName = '') then
  begin
    OpenSaveFile(BiFileVar,BiFileName);
    SaveNewFile(BiFileVar,BiPicture);
  end;
  Sound(2000); delay(500); nosound;
end;   { CreateFileBinaryPicture }

procedure Yes_No_SavePicture(Picture:DataPnT);
var
  ch : char;
begin
  gotoXY(1,23); ClrEol;
  write('Do You want save result of picture (Y/N) ');
  repeat
    read(Kbd,ch);
  until UpCase(ch) IN ['Y','N'];
  if UpCase(ch) = 'Y' then
    begin
      gotoXY(1,23); ClrEol;
      write('Result of picture being saved ... ');
      CreateFileBinaryPicture(Picture);
    end;
end; { Yes_No_SavePicture }

procedure Yes_No_SavePictureOnScreen;
var
  ch : char;
  BiFileVar  : FileBT;
  BiFileName : String50;
  tem1,tem0 : byte;
begin
  tem1 := 68H; tem0 := 68L;
  gotoXY(1,23); ClrEol;
  write('SAVE picture on screen { X = ',PosiI:3,' Y = ',PosiJ:3,' } (Y/N) ');
  repeat
    ch := UpCase(ch);
    read(Kbd,ch);
  until UpCase(ch) IN ['Y','N'];
  if UpCase(ch) = 'Y' then
    begin
      gotoXY (1,23); ClrEOL ;
      write (' Enter Filename',#39,'s Binary for SAVE ...');
      read (BiFileName);
      OpenSaveFile(BiFileVar,BiFileName);
```

```
      gotoXY(1,23); ClrEol;
      write('Result of picture being saved ... ');
      for Line_Y := 1 to Y_Max do
      begin
        for Line_X := 1 to X_Max do
          begin
            if EGAGetdot(Line_X+PosiI,Line_Y+PosiJ,0) > 0 then
               write(BiFileVar,tem1)
            else
               write(BiFileVar,tem0);
          end;
      end;
      Close(BiFileVar);
    end; { if }
  sound(2000); delay(500); nosound;
end; { Yes_No_SavePictureOnScreen }

procedure  Read_Data_File(var DataPic : DataPnT) ;
                          {  i------------i            }
var                       {  i /¦\    /¦\ i   Format    }
  tem : byte;             {  i  ¦ ---->¦  i     OF      }
  Created : boolean;      {  i  ¦      ¦  i     in      }
  Ok : char;              {  i------------i   ReadData  }
begin { Read_Data_File }
  Created := False;
  repeat
    gotoXY(1,23); ClrEOL ;
    write ('Enter filename ( Data from Digitizer ) ');
    readln ( FileName );
    Assign ( FileVar,FileName );
    {$I-} reset (FileVar) {$I+} ;
    Created := IOresult = 0 ;
    if Created then
      begin
        Wait_Message;
        Clear_Var;
        for Line_Y := 1 to Y_Max do
        begin
          sound (2000); delay(3); nosound;
          for Line_X := 1 to X_Max do
          begin
            read ( FileVar,ch);
            DataPic^.DataImage[Line_X]^.ArrayP[Line_Y] := ord(ch);
            tem := DataPic^.DataImage[Line_X]^.ArrayP[Line_Y];
            arrayTem[tem] := arrayTem[tem]+1;
          end;
        end;
        close(FileVar);
        gotoXY(25,21); write('FILE NAME ¦ ',FileName,'        ');
```

```
      end
    else
      begin
        gotoXY(1,23); ClrEol;
        Sound(2000); Delay(500); noSound;
        write('File ',FileName,' NOT Found...  Esc : Exit   Anykey to again ');
        read(kbd,Ok);
      end;
  until (Created) OR (Ok = #27);
end;  { Read_Data_File }

Function Max_Frequency : integer;
var
  a : byte;
  tem : integer;
begin { Max_Frequency }
  tem := 0;
  for a := 1 to Gray_Max do
  begin
    if tem < ArrayTem[a] then tem := ArrayTem[a];
  end;
  Max_Frequency := tem;
end;  { Max_Frequency }

procedure Getline(x1,y1,x2,y2,page:integer; var dotline : FormatL);
var
  x,y : integer;
  tem : integer;
begin  { Getline }
  if y1 = y2 then
    for x := x1 to x2 do
    begin
      tem := x - x1 ;
      dotline[tem] := EGAGetdot(x,y1,page);
    end
  else
    if x1 = x2 then
      for y := y1 to y2 do
      begin
        tem := y - y1 ;
        dotline[tem] := EGAGetdot(x1,y,page);
      end;
end;  { Getline }


procedure Putline(x1,y1,x2,y2,page:integer; dotline : FormatL);
var
  x,y      : integer;
```

```
  tem      : integer;
begin
  if y1 = y2 then
    for x := x1 to x2 do
    begin
      tem := x - x1 ;
      EGAPlot(x,y1,page,dotline[tem]);
    end
  else
    if x1 = x2 then
      for y := y1 to y2 do
      begin
        tem := y - y1 ;
        EGAPlot(x1,y,page,dotline[tem]);
      end;
end;  { Putline }

procedure Put_Pixel(X1,Y1,color:integer);
const
  Blank_L = 1;
var
  x  : integer;
begin  { Put_Pixel }
  for x := x1+Blank_L+1 to x1+SizePix-Blank_L  do
    if Odd(x) then  { tecnic up speed }
      begin
         EGADraw( x,Y1+Blank_L,x,Y1+SizePix-Blank_L,0,0); { Clr_Color }
         EGADraw( x,Y1+Blank_L,x,Y1+SizePix-Blank_L,0,color);
      end;
end; { Put_Pixel }

Function Get_Pixel(X1,Y1:integer):byte; { NOW NO USE }
const
  Blank_L = 1;
begin { Get_Pixel }
  if Odd(X1) then
    Get_Pixel := EGAGetdot(X1,Y1+Blank_L,0)
  else
    Get_Pixel := EGAGetdot(X1+1,Y1+Blank_L,0);
end; { Get_Pixel }

procedure SetBorderSize(var size : byte );
begin { SetBorderSize }
  gotoXY(1,23); ClrEol;
  write('Window size ');
  textcolor(red + blink); write(size);
  textcolor(yellow); write(' Enter new size ');
  read(size);
  sound(2000); delay(500); nosound;
```

```
end; { SetBorderSize }

procedure ExpressWindow(DataIn_X,DataIn_Y:integer;FillColor:boolean);
const
  Disp_X = 260;
  Disp_Y = 0;
var
  PosX,PosY,color : integer;
begin
  if FillColor then
    begin
      EGADrawBorder(Disp_X,Disp_Y,Disp_X+((Border+1)*SizePix),
                                  Disp_Y+((Border+1)*SizePix),0,4);
      for PosX := 0 to Border do
      begin
        for PosY := 0 to Border do
        begin
          color := EGAGetdot( DataIn_X+PosX , DataIn_Y+PosY , 0);
          Put_Pixel(Disp_X+(PosX*SizePix),Disp_Y+(PosY*SizePix),color);
        end;
        Sound(2000); Delay(5); noSound;
      end;
      Sound(2000); Delay(500); noSound;
    end
  else  { Delete ExpressWindow }
    begin
      EGADrawBorder(Disp_X,Disp_Y,Disp_X+((Border+1)*SizePix),
                                  Disp_Y+((Border+1)*SizePix),0,0);
      for PosX := 0 to Border do
      begin
        for PosY := 0 to Border do
        begin
          Put_Pixel(Disp_X+(PosX*SizePix),Disp_Y+(PosY*SizePix),0);
        end;
        Sound(2000); Delay(5); noSound;
      end;
      Sound(2000); Delay(500); noSound;
    end;
end; { ExpressWindow }

procedure Cursor(var XX,YY : integer ; direction : char );
var
  temX,temY : integer;
begin { Cursor }
  temX := XX ; temY := YY ; { protect cursor out of screen }
  case direction of
         #49 :  begin
                  XX := XX - 1 ;   YY := YY + 1 ;
                end;                          .          { End   }
```

```
          #50 :  begin YY := YY + 1 ; end;              { DOWN  }
          #51 :  begin
                   XX := XX + 1 ;   YY := YY + 1 ;
                 end;                                   { PgDn  }
          #52 :  begin XX := XX - 1 ; end;              { LIFT  }
          #54 :  begin XX := XX + 1 ; end;              { RIGHT }
          #55 :  begin
                   XX := XX - 1 ;   YY := YY - 1 ;
                 end;                                   { Home  }
          #56 :  begin YY := YY - 1 ; end;              {  UP   }
          #57 :  begin
                   XX := XX + 1 ;   YY := YY - 1 ;
                 end;                                   { PgUp  }
          #27 :  begin
                   read (kbd,direction);
                   case direction of
                     #71 : begin
                             XX := XX - 10 ; YY := YY - 10 ;
                           end;                         { Home  + 10 }
                     #72 : begin
                             YY := YY - 10 ;
                           end;                         {  UP   + 10 }
                     #73 : begin
                             XX := XX + 10 ; YY := YY - 10 ;
                           end;                         { PgUp  + 10 }
                     #75 : begin
                             XX := XX - 10 ;
                           end;                         { LIFT  + 10 }
                     #77 : begin
                             XX := XX + 10 ;
                           end;                         { RIGHT + 10 }
                     #79 : begin
                             XX := XX - 10 ; YY := YY + 10 ;
                           end;                         { End   + 10 }
                     #80 : begin
                             YY := YY + 10 ;
                           end;                         { DOWN  + 10 }
                     #81 : begin
                             XX := XX + 10 ; YY := YY + 10 ;
                           end;                         { PgDn  + 10 }
                   end; { Case }
                 end;
        end;  { Case }
  if XX < 0 then
    begin
      XX := temX ;
    end;
  if YY < 0 then
    begin
```

```
      YY := temY ;
    end;
  sound (2000); delay (10); nosound ;
end;  { Cursor }


procedure MoveWindow(var XX,YY : integer);
var
  Stop : char;
  TemColorH1,TemColorH2,TemColorV1,TemColorV2 : FormatL;
  Tem_X,Tem_Y : integer;
begin { MoveWindow }
  Stop := 'a' ;                              { Protect 'Stop' }
  gotoXY(1,23); ClrEOL;
  write('Arrow key for Move window  ::  Enter for Continue ');
  while Stop <> #$0D do
  begin
    gotoXY(60,21); write('X = ',PosiI:3,', Y = ',PosiJ:3);
    Getline(XX,YY,XX+Border,YY,0,TemColorH1);
    Getline(XX,YY+Border,XX+Border,YY+Border,0,TemColorH2);
    Getline(XX,YY,XX,YY+Border,0,TemColorV1);
    Getline(XX+Border,YY,XX+Border,YY+Border,0,TemColorV2);
    Tem_X := XX;
    Tem_Y := YY;
    EGADrawBorder(XX,YY,XX+Border,YY+Border,0,4);
    read(kbd,Stop);
    Cursor(XX,YY,Stop);
    Putline(Tem_X,Tem_Y,Tem_X+Border,Tem_Y,0,TemColorH1);
    Putline(Tem_X,Tem_Y+Border,Tem_X+Border,Tem_Y+Border,0,TemColorH2);
    Putline(Tem_X,Tem_Y,Tem_X,Tem_Y+Border,0,TemColorV1);
    Putline(Tem_X+Border,Tem_Y,Tem_X+Border,Tem_Y+Border,0,TemColorV2);
  end;  { while }
  Sound(2000); Delay(500); noSound;
end;  { MoveWindow }

procedure  CreateScaleHis;
var
  a : integer;
begin { CreateScaleHis }
  EGADraw (300,260,560,260,0,1);                    { Hor }
  EGADraw (300,260,300,0,0,1);                      { Ver }
  gotoXY (34,1); write ('N(g)');
  gotoXY (73,19); write ('260');       { Axis Gral level = 260 }
  gotoXY (70,20); write ('Gray Level');
  a := 300 ;
  repeat
    EGAPlot (a,260,0,0);
    a := a + 10;                                 { Scale Hor }
  until (a > 560);
```

```
  a := 260 ;
  repeat
    EGAPlot (300,a,0,0);                              { Scale Ver }
    a := a - 10;
  until (a < 0);
end; { CreateScaleHis }

procedure DeleteScaleHis;
begin { DeleteScaleHis }
  EGADraw (300,260,560,260,0,0);                        { Hor }
  EGADraw (300,260,300,0,0,0);                          { Ver }
  gotoXY (34,1);
  TextColor(0); write ('N(g)'); TextColor(14);
  gotoXY (73,19);                      { Axis Gral level = 260 }
  TextColor(0); write ('260');  TextColor(14);
  gotoXY (70,20);
  TextColor(0); write ('Gray Level'); TextColor(14);
end; { DeleteScaleHis }


procedure  DisplayHis;
var
  tem : integer;
begin { DisplayHis }
  if NOT (Max_Frequency = 0)  then { Protect Division by zero attempted }
    begin
      tem := Max_Frequency ;
      for Line_X := 1 to Gray_Max do
      begin
        EGADraw(Line_X+300,260,
            .    Line_X+300,260-trunc(260*(ArrayTem[Line_X]/tem)),
                 0,3);
      end;
      EGAToneColor(630,0,639,260,0);
      CreateScaleHis;
      Sound(2000); Delay(500); noSound;
    end; { if }
end;  { DisplayHis }

procedure DeleteDataInWindow(DataIn_X,DataIn_Y : integer);
var
  i : integer;
begin { DeleteDataInWindow }
  for i := 0 to Border do
  begin
    EGADraw(DataIn_X+i,DataIn_Y,DataIn_X+i,DataIn_Y+Border,0,0);
  end;
  sound(2000); delay(500); nosound;
end; { DeleteDataInWindow }
```

```
procedure DeleteHis;
var
  tem : integer;
begin { DeleteHis }
  tem := Max_Frequency ;
  for Line_X := 1 to Gray_Max do
  begin
    EGADraw(Line_X+300,260,
            Line_X+300,260-trunc(260*(ArrayTem[Line_X]/tem)),
            0,0);
  end;
  DeleteScaleHis;
  Sound(2000); Delay(500); noSound;
end; { DeleteHis }


procedure  Display_Scene;          {------------}  {           }
var                                { --------)  }  {  Format   }
  tem  : integer;  .               {     |      }  {    of     }
  temP : integer;                  {    \|/     }  { Displayer }
  lowe , upp ,col : integer;       { --------)  }  {           }
begin { Display_Scene }            {------------}  {           }
  gotoXY (1,21); write ('Lower .. Upper ,Color ');
  gotoXY (1,23); ClrEOL;
  write ('Key numberic (Gray Level)');
  gotoXY (1,22); read ( lowe,upp,col);
  for Line_Y := 1 to Y_Max do
    for Line_X := 1 to X_Max do
    begin
      tem := Datadig^.DataImage[Line_X]^.ArrayP[Line_Y];
      if (tem <= upp) AND (tem >= lowe) then
         EGAPlot ( Line_X+PosiI,Line_Y+PosiJ,tone,col );
    end;
  sound (2000);  delay (500); nosound ;
end;  { Display_Scene }


procedure Start;
begin { Start }
  Initialize_Start;
  Read_Data_File(DataDig);
  DisplayHis;
  Display_Scene;
end; { Start }
```

Program :
Programmer :

```
{*****************************************************************
 *                                                               *
 *            SEVERAL PROCEDURE FOR PROGRAM COLAT.PAS            *
 *                                                               *
 *                          PART TWO                             *
 *                                                               *
 *                        VERSION 1.3 A                          *
 *                                                               *
 *              BY    CHANCHAI PISITTIVITHAYANON                 *
 *                                                               *
 *****************************************************************}


procedure CreatePictureOnScreen(XX,YY : integer);
const
  Disp_X = 260;
  Disp_Y = 0;
var
  Stop : char;
  TemColor , Color , TemX , TemY : integer;
  x1,y1,x2,y2 : integer;
  OffSetW_X , OffSetW_Y : integer;
  NumPoint : byte;
  ch : char;
begin  { CreatePictureOnScreen }
  Stop := 'a' ; { Protect 'Stop' }
  gotoXY(1,23); ClrEOL;
  Color := 14;
  NumPoint := 0;
  OffSetW_X := XX ;
  OffSetW_Y := YY ;
  write('Point :: L:Line  O:Circle  F:Fill  D:Delete  C:Color  S:Save  E:Exit');
  while NOT (UpCase(Stop) = 'E') do
  begin
    gotoXY(60,21); write('X = ',XX:3,', Y = ',YY:3);
    TemColor := EGAGetdot(XX,YY,0);
    TemX := XX ;
    TemY := YY ;
    EGAPlot(XX,YY,0,Color);
    Put_Pixel( Disp_X+((XX-OffSetW_X)*SizePix) ,
```

```
            Disp_Y+((YY-OffSetW_Y)*SizePix) , Color );
    read(kbd,Stop);
    if UpCase(Stop) IN ['L','O','F','D','S','C'] then
       case UpCase(Stop) of
        'L' :  begin
                 case NumPoint of
                   0 : begin
                         EGAPlot(TemX,TemY,0,12);
                         x1 := TemX ; y1 := TemY ;
                         NumPoint := 1;
                         gotoXY(30,22); write('POINT ',NumPoint:2);
                       end;
                   1 : begin
                         EGAPlot(TemX,TemY,0,12);
                         x2 := TemX ; y2 := TemY ;
                         NumPoint := 0;
                         EGADraw(x1,y1,x2,y2,0,15);
                         gotoXY(30,22); write('        ');
                       end;
                 end; { case NumPoint }
               end;
        'O' :  begin
                 case NumPoint of
                   0 : begin
                         EGAPlot(TemX,TemY,0,12);
                         x1 := TemX ; y1 := TemY;
                         NumPoint := 1 ;
                         gotoXY(30,22); write('POINT ',NumPoint:2);
                       end;
                   1 : begin
                         EGAPlot(TemX,TemY,0,12);
                         x2 := TemX ; y2 := TemY;
                         NumPoint := 0;
                         x2 := round( Sqrt(Sqr(abs(x1-x2))+Sqr(abs(y1-y2))) );
                         EGACircle(x1,y1,x2,0,color);
                         EGAPlot(x1,y1,0,0);
                         gotoXY(30,22); write('        ');
                       end;
                 end; { case NumPoint }
               end;
        'F' :  begin
                 EGAPlot(TemX,TemY,0,Color);
                 Put_Pixel( Disp_X+((TemX-OffSetW_X)*SizePix) ,
                            Disp_Y+((TemY-OffSetW_Y)*SizePix) , Color );
                 sound (2000); delay (10); nosound ;
               end;
        'D' :  begin
                 EGAPlot(TemX,TemY,0,0);
                 Put_Pixel( Disp_X+((TemX-OffSetW_X)*SizePix) ,
```

```
                        Disp_Y+((TemY-OffSetW_Y)*SizePix) , 0 );
              sound (2000); delay (10); nosound ;
            end;
       'S' :  begin
              Yes_No_SavePictureOnScreen;
              gotoXY(1,23); ClrEol;
              write('Point :: L:Line  O:Circle  F:Fill  D:Delete  C:Color  S:Save  E:Exit');
              EGAPlot(TemX,TemY,0,TemColor);
              Put_Pixel( Disp_X+((TemX-OffSetW_X)*SizePix) ,
                        Disp_Y+((TemY-OffSetW_Y)*SizePix) , TemColor );
              end;
       'C' :  begin
              gotoXY(1,23); ClrEOL;
              write('Current color = ',color);
              gotoXY(whereX,23); write('   Charge to '); read(color);
              gotoXY(1,23); ClrEOL;
              write('Point :: L:Line  O:Circle  F:Fill  D:Delete  C:Color  S:Save  E:Exit');
              EGAPlot(TemX,TemY,0,TemColor);
              Put_Pixel( Disp_X+((TemX-OffSetW_X)*SizePix) ,
                        Disp_Y+((TemY-OffSetW_Y)*SizePix) , TemColor );
            end;
       end { case }
    else
       begin
         Cursor(XX,YY,Stop);
         EGAPlot(TemX,TemY,0,TemColor);
         Put_Pixel( Disp_X+((TemX-OffSetW_X)*SizePix) ,
                    Disp_Y+((TemY-OffSetW_Y)*SizePix) , TemColor );
       end;
  end;  { while }
  sound (2000); delay(500); nosound;
end; { CreatePictureOnScreen }

procedure Create_BinaryPicture;
begin { Create_BinaryPicture }
  for Line_Y := 1 to Y_Max do
    for Line_X := 1 to X_Max do
    begin
      if EGAGetdot(Line_X,Line_Y,0) > 0 then
        BinPict^.DataImage[Line_X]^.ArrayP[Line_Y] := G8L
      else
        BinPict^.DataImage[Line_X]^.ArrayP[Line_Y] := G8H;
    end;
end;  { create_BinaryPicture }

function ATan (x,y : real) : real ;
const
  pi180 = 57.2957795;
var
```

```
  a : real;
begin { ATan }
  if x = 0 then
    if y = 0 then ATan := 0
    else ATan := 90
  else { x <> 0 }
    if y = 0 then ATan := 0
    else { x and y <> 0 }
      begin
        a := arctan(abs(y/x)) * pi180 ;
        if x > 0 then
          if y > 0 then ATan := a { x,y > 0 }
          else Atan := -a { x > 0 , y < 0 }
        else { x < 0 }
          if y > 0 then ATan := 180 - a { x < 0 , y > 0 }
          else atan := 180 + a { x,y < 0 }
      end; { else }
end; { ATan }


function ArcCos(x:real) : real;
{ arccos in degrees }
{ function ATan is required }
begin { ArcCos }
  if x = 0 then arccos := 90
  else
    if x = 1 then arccos := 0
    else
      if x = -1 then arccos := 180
      else arccos := ATan( x/sqrt(1-sqr(x)),1 );
end; { ArcCos }
```