การเลือกหยิบวัตถุของแขนกล

โดยการตรวจสอบจากรูปร่างของวัตถุ

OBJECT IDENTIFICATION FOR ROBOT ARM

โดย

นาย ศุภสิทธิ์ ศิริทองถาวร รหัส 33100383

น.ส. อรทัย หนูเล็ก รหัส 33100500

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษา

ตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมการวัดคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง

ปีการศึกษา 2536

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

สาขา วิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การเลือกหยิบวัตถุของแขนกลโดยการตรวจสอบจากรูปร่างของวัตถุ

OBJECT IDENTIFICATION FOR ROBOT ARM

ผู้จัดทำ

1. นาย ศุภสิทธิ์ ศิริทองถาวร รหัสประจำตัว 33100383
2. น.ส. อรทัย หนูเล็ก รหัสประจำตัว 33100500

..... อาจารย์ที่ปรึกษา

(ร.ศ.ดร. ฟูศักดิ์ ชีวสุวิทย์)

สารบัญ

การเลือกหยิบวัตถุของแขนกลโดยการตรวจสอบจากรูปร่างของวัตถุ

OBJECT IDENTIFICATION FOR ROBOT ARM

โดย นาย ศุภสิทธิ์ ศิริทองถาวร รหัสประจำตัว 33100383
น.ส. อรทัย หนูเล็ก รหัสประจำตัว 33100500

อาจารย์ที่ปรึกษา รศ.ดร.ฟูศักดิ์ ชีวสุวิทย์

บทคัดย่อ

ในปัจจุบัน ลักษณะการทำงานของแขนกล จะดำเนินไปตามลำดับขั้นของโปรแกรมที่ได้กำหนดตำแหน่งและทิศทางการเคลื่อนที่ไว้แน่นอนแล้ว ซึ่งทำให้แขนกลไม่สามารถตัดสินใจกำหนดทิศทางการเคลื่อนที่ได้ด้วยตนเอง ดังนั้นปริญญานิพนธ์ฉบับนี้ จึงได้เสนอการนำเทคนิคการเปรียบเทียบแกนหลักมาใช้ร่วมกับแขนกล เพื่อทำการแยกแยะชนิดของวัตถุที่มีรูปร่างทางเรขาคณิตแตกต่างกัน และสามารถเลือกหยิบวัตถุตามชนิดที่ต้องการได้ด้วยตนเองซึ่งการใช้เทคนิคดังกล่าว จะทำให้แขนกลสามารถแยกแยะชนิดของวัตถุได้อย่างถูกต้อง ไม่ว่าวัตถุจะหมุนไปในทิศทางใดก็ตาม นับได้ว่าเป็นแนวทางในการพัฒนากระบวนการคิดของแขนกล ให้มีประสิทธิภาพสูงขึ้น

ABSTRACT

In present, the movement of robot arm depends on the steps of program which the position and direction have already been assigned.The robot arm can not assign the movement by itself. So in ihis thesis,the technique of principal axis comparison is applied for robot arm in order to identify the shape of object. Then the robot arm can assign its movement to pick up the required object by itself. By this way, although the object is rotated in different direction, the robot arm can identify the shape of object correctly. So it is an idea to develop ability of thinking for robot arm in order to work more efficiently.

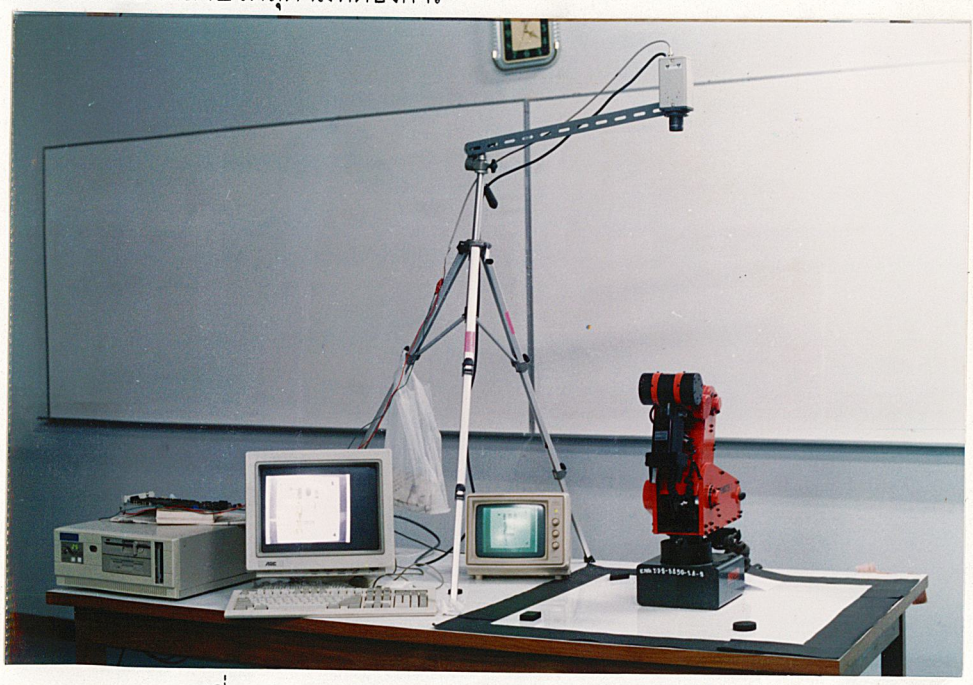
บทที่ 1

บทนำ

ในปัจจุบัน หุ่นยนต์ได้เข้ามามีบทบาท ในวงการอุตสาหกรรมของประเทศ มากยิ่งขึ้น แต่ความสามารถในการทำงาน ยังมีความยืดหยุ่นไม่มากนัก กล่าวคือ ในสถานการณ์ที่เงื่อนไขเปลี่ยนแปลงไป หุ่นยนต์ยังไม่สามารถที่จะตัดสินใจเลือกกระทำการใดๆได้ด้วยตนเองดังนั้น จึงได้มีการพัฒนาโดยการนำคอมพิวเตอร์ มาทำหน้าที่เป็นหน่วยประมวลผล เพื่อช่วยในการตัดสินใจของหุ่นยนต์ ตัวอย่างเช่น การนำหลักของการรู้จำเสียงมาใช้ในการบังคับหุ่นยนต์ให้ทำงานตามคำสั่งที่เป็นเสียงพูด และการนำหลักของการจดจำรูปแบบ มาใช้ในการตัดสินใจเลือกหยิบอุปกรณ์อิเลคทรอนิคส์ นำไปวางบนแผงวงจรในตำแหน่งที่ต้องการ

สำหรับในปริญญานิพนธ์ฉบับนี้ ได้ทำการพัฒนาให้หุ่นยนต์สามารถมองเห็น และเลือกหยิบวัตถุที่มีรูปร่างตามคำสั่งได้ โดยจำลองส่วนประกอบของการทำงานออกเป็น 3 ส่วนคือ

1. แขน ใช้ robot arm รุ่น RM-501 ในการหยิบจับวัตถุ
2. ตา ใช้ กล้องโทรทัศน์วงจรปิด ประกอบกับ การ์ด digitizer ในการบันทึกภาพ
3. สมอง ใช้ คอมพิวเตอร์รุ่น 386SX ในการประมวลผลสัญญาณภาพ เพื่อใช้ในการตัดสินใจให้แขนกลจับวัตถุตามที่ต้องการ



รูปที่ 1.1 แผนภาพแสดงส่วนประกอบของการทำงาน

การที่จะให้แขนกลสามารถกระทำการดังกล่าวได้นั้น ต้องอาศัยหลักการต่างๆ ดังรายละเอียดที่จะกล่าวถึงในบทต่อไปนี้

บทที่ 2 ได้กล่าวถึง การเตรียมข้อมูลภาพสำหรับการประมวลผล นับตั้งแต่การรับภาพ, การขจัดสัญญาณรบกวนออกจากภาพ จนถึงการตรวจหาขอบของภาพวัตถุ ซึ่งเป็นข้อมูลที่สำคัญในการบอกขอบเขตของวัตถุแต่ละชิ้น

บทที่ 3 กล่าวถึง การกำจัดส่วนเกินของขอบภาพวัตถุ ซึ่งไม่จำเป็นในการนำมาใช้พิจารณาเพื่อแบ่งแยกขอบเขตของวัตถุ โดยทำการลดความหนาของขอบภาพลงให้เหลือเพียง 1 จุดภาพ

บทที่ 4 กล่าวถึง การนำขอบของภาพวัตถุที่ลดความหนาลงแล้ว มาเข้ารหัสลูกโซ่แบบ 8 ทิศทางเพื่อนำมาใช้ในการพิจารณาแบ่งแยกเนื้อของวัตถุแต่ละชิ้น

บทที่ 5 กล่าวถึง การแยกแยะรูปร่างของวัตถุ โดยใช้หลักการของการเปรียบเทียบแกนหลักโดยอาศัยรหัสลูกโซ่ที่ได้จากบทที่ 4 เป็นตัวบอกขอบเขตของวัตถุแต่ละชิ้น

บทที่ 6 กล่าวถึง การควบคุมทิศทางการเคลื่อนไหวของแขนกล

และในบทที่ 7 จะเป็นการ สรุปผลของงานทั้งหมดที่ได้จากการทดลอง

บทที่ 2

การตรวจหาขอบของภาพวัตถุ

2.1 ข้อมูลภาพ

ในการจดจำรูปร่างของวัตถุนั้น จะต้องเริ่มต้นมาจากการรับภาพของวัตถุเข้ามาเสียก่อน โดยอาศัยอุปกรณ์สำหรับรับภาพ ซึ่งได้แก่กล้องต่างๆ เพื่อแปลงความสว่างในแต่ละตำแหน่งของภาพนั้นให้อยู่ในรูปของสัญญาณทางไฟฟ้าแบบอนาล็อก จากนั้นจะต้องมีอุปกรณ์สำหรับแปลงสัญญาณภาพดังกล่าวให้เป็นสัญญาณที่คอมพิวเตอร์สามารถรับรู้ได้ คือ สัญญาณแบบดิจิตอล ซึ่งโดยทั่วไปจะใช้การ์ด Digitizer ซึ่งจะประกอบด้วยวงจร Analog to Digital Converter (ADC) และมีการจัดเรียงสัญญาณดิจิตอล ซึ่งแทนค่าความสว่างของภาพในแต่ละจุดให้เกิดเป็นภาพ ซึ่งเรียกว่า Digital image

Digital image นี้จะเป็นข้อมูลของภาพวัตถุที่คอมพิวเตอร์สามารถนำไปใช้ในการจดจำรูปร่างวัตถุต่อไป แต่โดยส่วนใหญ่แล้ว ข้อมูลภาพที่รับเข้ามานี้ยังไม่เหมาะสมที่จะนำไปประมวลผลได้ทันที คืออาจจะมี noise รบกวนอยู่ ถ้านำข้อมูลดิบไปประมวลผลทันที อาจจะทำให้เกิดการตีความที่ผิดพลาดได้ ดังนั้นจึงจำเป็นจะต้องผ่านกระบวนการ Preprocessing เสียก่อน โดยกระบวนการ Preprocessing ที่ได้นำมาใช้ในปริญญานิพนธ์ฉบับนี้ คือ การ Smoothing ซึ่งจะได้กล่าวถึงในหัวข้อต่อไป

2.2 Smoothing

Smoothing เป็นกระบวนการในการปรับปรุงคุณภาพของข้อมูลภาพ โดยจะเป็นการลด noise และความผิดพลาดอื่นๆซึ่งอาจจะเกิดขึ้นกับข้อมูลภาพ อันอาจจะเป็นผลมาจากการ Sampling, Quantization, การส่งถ่ายข้อมูล หรือ การรบกวนจากสิ่งแวดล้อมขณะทำการรับภาพ ซึ่งกระบวนการในการ Smoothing นี้ สามารถกระทำได้หลายวิธีด้วยกัน แต่ปริญญานิพนธ์ฉบับนี้ได้เลือกมาทดสอบเพียง 2 วิธีเท่านั้น คือ

2.2.1.Neighborhood Averaging

วิธีการนี้เป็นเทคนิคแบบ spatial domain อย่างง่ายสำหรับการ Smoothing โดยกำหนดให้ค่าความสว่างของภาพ ณ จุดภาพ (x,y) ของข้อมูลภาพ ถูกแทนด้วยฟังก์ชัน $f(x,y)$ และเมื่อ

ผ่านกระบวนการ Smoothing แล้วจะได้ภาพใหม่ซึ่งมีฟังก์ชั่น $g(x,y)$ แทนข้อมูลภาพที่ขจัด noise ออกไปแล้ว ซึ่งค่าความเข้มของแต่ละจุดภาพ (x,y) ของภาพใหม่นี้ ได้มาจากการหาค่าเฉลี่ยของค่าความเข้มของจุดภาพซึ่งอยู่ในบริเวณใกล้เคียงกับจุด(x,y)) สามารถเขียนความสัมพันธ์ได้ดังนี้

$$g(x,y) = \frac{1}{P} \sum_{(n,m)\varepsilon S} f(n,m)$$

สำหรับพิกัด x และ y ทุกๆค่าของข้อมูลภาพ

โดยที่ S เป็นเซทของตำแหน่งของจุดภาพ ที่อยู่ในบริเวณใกล้เคียงกับจุด (x,y) รวมทั้งจุด(x,y)นั้นเองด้วย

และ P คือจำนวนจุดภาพทั้งหมดที่อยู่ในเซท S เช่น ถ้าให้บริเวณใกล้เคียงเป็นหน้าต่างขนาด 3x3 ก็จะได้ P เท่ากับ 9 จุด

2.2.2 Median Filtering

เนื่องจากการ Smoothing ด้วยวิธี Neighborhood Averaging นั้นมีข้อเสียอยู่อย่างหนึ่งคือ จะทำให้ภาพบริเวณขอบวัตถุ และข้อมูลที่สำคัญอื่นๆ มัวลงไป บ่อยครั้งเราสามารถลดความมัวนี้ได้โดยใช้วิธีการ Median Filtering ซึ่งค่าความเข้มของแต่ละจุดภาพ(x,y)ของภาพใหม่ จะได้มาจากการหาค่ามัธยฐานของค่าความเข้มของจุดภาพที่อยู่ในบริเวณใกล้เคียงกับจุด (x,y) แทนที่จะใช้ค่าเฉลี่ยตามวิธีการ Neighborhood Averaging เมื่อทำการปรับปรุงคุณภาพของภาพด้วยกระบวนการ Smoothing แล้ว เนื่องจากภาพที่จะรับเข้ามา เป็นภาพของวัตถุหลายชิ้น ดังนั้น การแยกแยะรูปร่างของวัตถุแต่ละชิ้น จำเป็นจะต้องรู้ขอบเขตของวัตถุแต่ละชิ้นนั้นเสียก่อน จึงต้องทำการตรวจหาขอบของภาพวัตถุด้วยกระบวนการ Edge Detection ดังจะได้กล่าวถึงหลักการในหัวข้อต่อไป

2.3 Edge Detection

Edge Detection เป็นกระบวนการที่มีบทบาทสำคัญในการมองเห็นของหุ่นยนต์ ซึ่งเป็นขั้นตอนของการเตรียมข้อมูลภาพเริ่มต้นสำหรับอัลกอริทึมในการตรวจหาวัตถุ หลักการเบื้องต้นของ Edge Detection คือ ในบริเวณที่เป็นขอบใดขอบหนึ่งของภาพวัตถุ ย่อมจะมีอัตราการเปลี่ยนแปลงความเข้มของจุดภาพจากด้านหนึ่งไปยังอีกด้านหนึ่ง มากกว่าบริเวณที่เป็นเนื้อวัตถุ ดังนั้นในการที่จะตรวจหาว่าจุดภาพใดเป็นส่วนของขอบของภาพวัตถุ จะต้องมีการหาอัตราการเปลี่ยนแปลงนั้น โดยสามารถหาได้ในรูปของ gradient ดังนี้

กำหนดให้ gradient ของภาพ $f(x,y)$ ที่ตำแหน่งจุดภาพ(x,y) เป็นเวคเตอร์ 2 มิติ

คือ
$$\mathbf{G}\left[f(x,y)\right] = \begin{bmatrix} Gx \\ Gy \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (2.1)$$

จากหลักการวิเคราะห์ทางเวคเตอร์ ทำให้สรุปได้ว่า เวคเตอร์ $\mathbf{G}$ จะชี้ไปในทิศทางที่มีอัตราการเปลี่ยนแปลงของ f มากที่สุด ณ ตำแหน่ง (x,y) แต่ในกระบวนการ Edge Detection เราสนใจเฉพาะ gradient ซึ่งเป็นขนาดของเวคเตอร์ $\mathbf{G}$ นี้เท่านั้น

$$\begin{aligned} G[f(x,y)] &= \left[Gx^2 + Gy^2\right]^{1/2} \\ &= \left[\left[\frac{\partial f}{\partial x}\right]^2 + \left[\frac{\partial f}{\partial y}\right]^2\right]^{1/2} \end{aligned} \quad (2.2)$$

แต่ในทางปฏิบัติ เราสามารถประมาณค่าของ gradient ได้ด้วยค่าสัมบูรณ์

$$G[f(x,y)] = |Gx| + |Gy| \quad (2.3)$$

ทำให้ง่ายต่อการคำนวณสำหรับฮาร์ดแวร์ที่มีขีดจำกัดด้านการคำนวณ สำหรับการหาค่า Gx และ Gy นั้นสามารถทำได้หลายวิธี วิธีหนึ่งที่ใช้ คือ หาค่าความแตกต่างของจุด(x,y) กับจุดถัดไป ดังนี้

$$Gx = \frac{\partial f}{\partial x} = f(x,y) - f(x-1,y)$$

และ
$$Gy = \frac{\partial f}{\partial y} = f(x,y) - f(x,y-1) \quad 2.4)$$

มีอีกวิธีหนึ่งที่ซับซ้อนกว่าวิธีแรกเล็กน้อย คือ คำนวณจากจุดภาพที่อยู่ภายในกรอบหน้าต่างขนาด 3x3 ที่มีจุด (x,y) เป็นตำแหน่งศูนย์กลาง ดังนี้

$$\begin{aligned} Gx = \frac{\partial f}{\partial x} &= \left[f(x+1,y-1) + 2f(x+1,y) + f(x+1,y+1)\right] \\ &\quad -\left[f(x-1,y-1) + 2f(x-1,y) + f(x-1,y+1)\right] \\ &= (g+2h+i) - (a+2b+c) \end{aligned}$$

และ
$$\begin{aligned} Gy = \frac{\partial f}{\partial y} &= \left[f(x-1,y+1) + 2f(x,y+1) + f(x+1,y+1)\right] \\ &\quad -\left[f(x-1,y-1) + 2f(x,y-1) + f(x+1,y-1)\right] \\ &= (c+2e+i) - (a+2d+g) \end{aligned} \quad 2.5)$$

ซึ่งค่า a ถึง i ตามสมการทั้งสองนี้ จะแทนค่าของจุดภาพซึ่งอยู่ล้อมรอบจุดภาพ(x,y) ซึ่งสามารถแสดงเป็นรูปอย่างง่าย ดังนี้

a	b	c
d	(x,y)	e
g	h	i

จะเห็นได้ว่า จุดภาพที่อยู่ใกล้กับจุด(x,y)มากกว่า จะมีน้ำหนักเป็น 2 เท่า สำหรับการหา Gx และ Gy ด้วยวิธีนี้จะมีข้อดีกว่าวิธีแรก คือ จะมีการเฉลี่ยมากกว่า ซึ่งมีผลดีทำให้ลดความไวต่อ noise ลงไป ขนาดของกรอบหน้าต่างอาจใช้มากกว่า 3x3 ก็ได้ แต่ขนาด 3x3 จะเป็นที่นิยมมากที่สุดสำหรับการมองของคอมพิวเตอร์ เนื่องจากใช้เวลาคำนวณไม่มาก และต้องการขีดความสามารถของฮาร์ดแวร์ไม่สูงนัก

ดังนั้น จึงสามารถคำนวณหาค่า Gx โดยใช้หน้ากากตามรูป 2.1 และคำนวณหาค่าของ Gy โดยใช้หน้ากากตามรูป 2.2 ได้ ซึ่งหน้ากากทั้ง 2 แบบนี้เรียกกันโดยทั่วไปว่า Sobel Operater เมื่อหาค่าของ Gx และ Gy โดยใช้ Sobel Operater ทั้งสองแล้วก็นำมารวมกันโดยอาจจะใช้สมการที่ 2.2 หรือประมาณโดยใช้สมการ 2.3 ก็ได้ เมื่อย้าย Sobel Operater ไปจนทั่วทุกจุดของภาพ $f(x,y)$ ก็จะได้ gradient ของทุกจุดของภาพนั้นออกมา

-1	-2	-1
0	0	0
1	2	1

รูปที่ 2.1 แสดงหน้ากากสำหรับการคำนวณหาค่า Gx

-1	0	1
-2	0	2
-1	0	1

รูปที่ 2.2 แสดงหน้ากากสำหรับการคำนวณหาค่า Gy

หลังจากหาค่า gradient ของแต่ละจุดภาพได้แล้ว ก็มีหลายวิธีที่จะนำค่า gradient เหล่านั้น มาสร้างให้เกิดภาพ $g(x,y)$ ที่จะนำไปใช้งาน วิธีที่ง่ายที่สุด คือ ใช้ค่า gradient ของภาพ f ที่จุด(x,y)ใดๆ มาเป็นค่าของภาพ g ที่จุด(x,y)นั้นๆ จะได้ว่า

$$g(x,y) = G[f(x,y)] \qquad (2.6)$$

หรือมีอีกวิธีการหนึ่งที่จะให้ได้ภาพที่จะนำไปใช้งาน คือ สร้างเป็น binary image โดยมีความสัมพันธ์ ดังนี้คือ

$$g(x,y) = \begin{cases} 1 & ;G[f(x,y)] > T \\ 0 & ;G[f(x,y)] \le T \end{cases} \quad (2.7)$$

เมื่อ T เป็นค่า threshold ที่ไม่เป็นลบ จะเห็นได้ว่า เฉพาะจุดที่มี gradient สูงกว่า Threshold เท่านั้นที่จะมีความสำคัญ กล่าวคือจุดภาพที่เป็นขอบของวัตถุเท่านั้นที่จะมี $g(x,y)$ เป็น 1 ส่วนบริเวณเนื้อวัตถุจะมีค่าเป็น 0 ทำให้สามารถตรวจหาขอบของภาพวัตถุได้

นอกจากนี้แล้ว ค่า threshold ที่ใช้ในการเปรียบเทียบกับค่า gradient นั้นอาจจะไม่ใช่ค่าคงที่ก็ได้ ตามที่เสนอในบทความที่[2] ซึ่งได้เสนอการคำนวณหาค่า threshold ของจุด(x,y)ใดๆ ดังนี้ ให้

$$T = \frac{1}{8}\sum_{i=1}^{8} Gi + t \quad 2.8)$$

โดยที่ T คือ ค่า threshold ที่ปรับค่าได้

Gi คือ ค่า gradient ของจุดภาพ 8 จุดที่ล้อมรอบจุด (x,y)

t คือ ค่าคงที่ที่นำมายกระดับของ threshold ซึ่งขึ้นอยู่กับลักษณะการจัดแสง

จะเห็นได้ว่า ค่า threshold จะเปลี่ยนไปตามค่า gradient ของจุดที่ล้อมรอบจุดที่พิจารณา เมื่อนำค่า gradient ของจุดที่พิจารณามาเปรียบเทียบกับค่า threshold ที่ได้โดยใช้ความสัมพันธ์ตามสมการที่ 2.6 แล้ว ก็จะได้ภาพซึ่งเป็น binary image เหมาะสำหรับนำไปผ่านกระบวนการจดจำวัตถุต่อไป

บทที่ 3

การลดความหนาของขอบภาพ (Thinning)

Thinning เป็นกระบวนการในการแปลงภาพ binary ให้เป็นโครงสร้างของเส้นที่มีความหนาเพียง 1 จุดภาพ การนำ thinning ไปใช้งานมีมากมาย เช่น การตรวจสอบ printed circuit boards, การนับเส้นใยแอสเบสตอสบนตัวกรองอากาศ, การวิเคราะห์รูปร่างของโครโมโซม, การแยกแยะลายนิ้วมือ, การจดจำตัวอักษรเป็นต้น

คุณสมบัติที่ดีของ Thinning ในการประมวลผลภาพและการจดจำรูปแบบ คือ

1. ลดเนื้อที่ในการจัดเก็บข้อมูล
2. ลดเวลาในการส่งถ่ายข้อมูล
3. ทำให้ง่ายต่อการแยกลักษณะเด่นของรูปร่าง จากภาพที่ถูก digitize มาแล้ว

ปัญหาสำคัญที่ต้องคำนึงถึงมี 3 ข้อ คือ

1. การคงสภาพของความต่อเนื่อง
2. การคงอยู่ของ end-points
3. ความแน่นอนของจุดที่จะถูกกำจัดอย่างสมมาตร

<u>หลักการทำงานของ Thinning</u>

Thinning จะแปลงรูปแบบเดิมของภาพ binary ให้ลดลงมาเป็นรูปร่างของโครงสร้างที่มีความหนา 1 จุดภาพ Thinning จะเสร็จสมบูรณ์ได้ต้องผ่านกระบวนการ 4 ขั้นตอน ซึ่งจะกระทำซ้ำไปซ้ำมาหลายๆ ครั้ง จนกระทั่งภาพนั้นไม่สามารถถูกลดความหนาได้อีกต่อไป กระบวนการทั้ง4 ขั้นตอนเรียงตามลำดับได้ดังนี้

(1) Fill

(2) Delete

3) Label

4) Strip

3.1 คำจำกัดความ

ตารางหน้าต่างขนาด3x3 : A,B,C,D,E,F,G,H และ X ประกอบเข้าด้วยกันเป็นตารางขนาด 3x3 ของจุดภาพที่ X

A	B	C
D	X	E
F	G	H

4-neighbour : จุด B,D,E และ G ที่ล้อมรอบจุด X เรียกว่า 4-neighbour ของจุดภาพที่ X

	B	
D	X	E
	G	

8-neighbour : จุด A,B,C,D,E,F,G และ H ที่ล้อมรอบจุด X เรียกว่า 8-neighbour ของจุดภาพที่ X

A	B	C
D	X	E
F	G	H

Cross-point : จุดภาพที่ X จะเป็น cross-point ก็ต่อเมื่อมีค่าเป็น 1 และใน 8-neighbour มีค่าที่แน่นอนดังแสดงในรูป

0	1	0
1	X	1
0	1	0

End-point : จุดภาพที่ X จะเป็น end-point ก็ต่อเมื่อมีค่าเป็น 1 และใน 8-neighbour มีจุดภาพเพียงจุดเดียวเท่านั้นที่มีค่าเป็น 1

0	0	1
0	X	0
0	0	0

Break-point : จุดภาพที่ X จะเป็น break-point ก็ต่อเมื่อมีค่าเป็น 1 และตารางหน้าต่างขนาด 3x3 ตรงกับแบบใดแบบหนึ่งในจำนวน 6 แบบ ดังต่อไปนี้

—	—	—
0	X	0
—	—	—

|	0	|
|	X	|
|	0	|

—	—	—
0	X	|
1	0	|

1	0	|
0	X	|
—	—	—

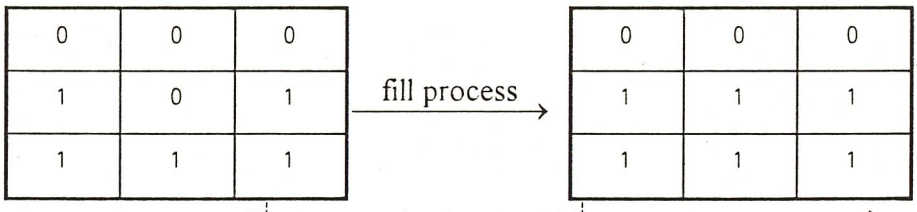
|	0	1
|	X	0
—	—	—

—	—	—
|	X	0
|	0	1

หมายเหตุ : เส้นประที่ลากผ่านตำแหน่งต่างๆ แสดงว่าตำแหน่งเหล่านี้จะประกอบด้วยจุดภาพที่มีค่าเป็น 1 เป็นจำนวน 1 จุด หรือมากกว่า 1 จุดก็ได้

3.2 กระบวนการ Fill

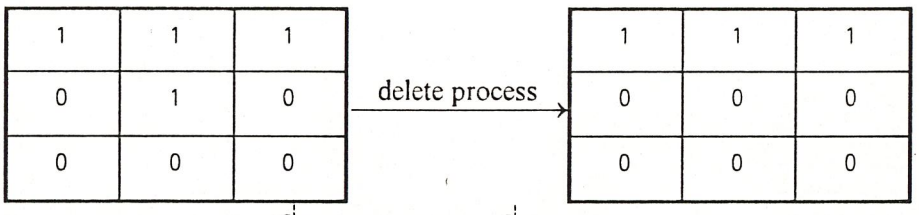
มีจุดประสงค์เพื่อทำให้ภาพ binary มีความราบเรียบ ซึ่งจะทำการ Fill ช่องว่างที่อยู่ในภาพ โดยการเปลี่ยนค่าของจุดภาพที่ X จาก 0 (สีขาว)เป็น 1 (สีดำ) ก็ต่อเมื่อใน 4-neighbour มีจุดภาพที่มีค่าเป็น 1 มากกว่า 2 จุด หรือ อีก 4 จุดที่เหลือมีค่าเป็น 1 มากกว่า 2 จุด ดังแสดงในรูปที่ 3.1



รูปที่ 3.1 แสดงตัวอย่างจุดภาพที่ถูก Fill

3.3 กระบวนการ Delete

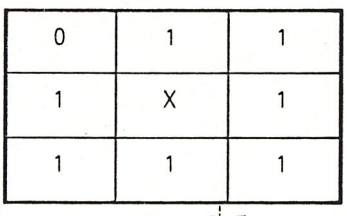
เป็นการทำให้ภาพ binary มีความราบเรียบเช่นเดียวกัน ซึ่งจะทำการกำจัดจุดภาพที่มีค่าเป็น 1 และเป็นจุดที่แยกตัวอยู่โดดเดี่ยว โดยจุดภาพที่ X จะถูกเปลี่ยนจาก 1 เป็น 0 ก็ต่อเมื่อฟังก์ชันบูลีน (A+B+D)(E+G+H)+(B+C+E)(D+F+G) เป็นเท็จ และ X ไม่ได้เป็นทั้ง break-point และ end-point ดังแสดงในรูปที่ 3.2



รูปที่ 3.2 แสดงจุดภาพที่ถูก delete

3.4 กระบวนการ Label

จะทำการ Label จุดที่เป็นขอบของภาพ (contour point) ทั้งหมด ซึ่งจุดขอบเหล่านั้นแบ่งออกได้เป็น 2 แบบ คือ แบบที่ 1 เป็น corner-point และแบบที่ 2 เป็น trace-point จุดภาพที่ X ซึ่งมีค่าเป็น 1 จะเป็น corner-point ก็ต่อเมื่อใน 8-neighbour มีจุดภาพที่มีค่าเป็น 1 อยู่ 7 จุด และต้องมีจำนวน 4 จุดที่อยู่ใน 4-neighbour ดังแสดงในรูปที่3.3



รูปที่ 3.3 แสดงจุดภาพที่เป็น corner-point

จุดภาพที่ X ซึ่งมีค่าเป็น 1 จะเป็น trace-point ก็ต่อเมื่อใน 4-neighbour มีจุดภาพที่มีค่าเป็น 1 ไม่เกิน 3 จุด ดังแสดงในรูปที่3.4

0	0	1
1	X	1
1	1	1

รูปที่ 3.4 แสดงจุดภาพที่เป็น trace-point

ซึ่งจุดขอบของภาพจะถูก Label จากซ้ายไปขวา และบนลงล่าง ตามลำดับ

3.5 กระบวนการ Strip

จะทำการกำจัดจุดที่เป็นจุดขอบของภาพทั้ง 2 แบบคือ trace-point และ end-point โดยทั้ง trace-point และ end-point จะถูก trace ทีละจุดและจะถูก Delete ทันที ถ้าจุดนั้นไม่ใช่ทั้ง end-point และ break-point

ในการ trace แต่ละรอบ ตารางหน้าต่างขนาด 3x3 จะถูกเลื่อนจากซ้ายไปขวาและจากบนลงล่างของภาพ binary เพื่อค้นหาตำแหน่งซึ่งจุด X เป็น trace-point จุดแรก ต่อจากนั้น 8-neighbourของ X ก็จะถูกตรวจสอบไปตามลำดับของการสแกนเริ่มต้นที่ถูกกำหนดไว้ เพื่อค้นหา trace-point ถัดไป ลำดับของการสแกนเริ่มต้นขึ้นอยู่กับชนิดของ trace ซึ่งมีอยู่ 4 ชนิด ได้แก่

(1) จุดขอบด้านนอก ทิศทางตามเข็มนาฬิกา

(2) จุดขอบด้านนอก ทิศทางทวนเข็มนาฬิกา

(3) จุดขอบด้านใน ทิศทางตามเข็มนาฬิกา

(4) จุดขอบด้านใน ทิศทางทวนเข็มนาฬิกา

การทำซ้ำของกระบวนการ strip ในรอบที่เป็นจำนวนคี่ ถ้าลำดับของการสแกนมีทิศทางตามเข็มนาฬิกาจะถูกสมมุติให้เป็น trace ตามจุดขอบด้านนอก และถ้าลำดับของการสแกนมีทิศทางทวนเข็มนาฬิกาก็จะเป็น trace ตามจุดขอบด้านใน

ส่วนการทำซ้ำในรอบที่เป็นจำนวนคู่ ถ้าลำดับของการสแกนมีทิศทางทวนเข็มนาฬิกาจะถูกสมมุติให้เป็น trace ตามจุดขอบด้านนอก และถ้าลำดับของการสแกนมีทิศทางตามเข็มนาฬิกาจะเป็น trace ตามจุดขอบด้านใน

จุดประสงค์ของการ trace และ strip จุดขอบด้านในและด้านนอกสลับกันไป ก็เพื่อทำให้เกิดความสมดุลทั้งจุดขอบด้านในและด้านนอกของภาพ binary ที่มีวงรอบปิด การทำงานที่สลับไปมาระหว่างทิศทางตามเข็มนาฬิกากับทิศทางทวนเข็มนาฬิกา จะช่วยในการ strip จุดขอบของภาพได้อย่างสมมาตร ดังนั้นโครงสร้างที่ได้จะไม่เอนเอียงไปด้านใดด้านหนึ่ง

จุดขอบเริ่มแรกที่ถูก trace จะเป็นจุดขอบด้านนอกเสมอ อย่างไรก็ตามเมื่อพบจุดที่ X เป็นtrace-point เริ่มแรกแล้ว ก็ยังไม่รู้ว่าจุดขอบถัดไปจะเป็นด้านนอกหรือด้านใน ซึ่งในการ Strip นี้ trace ตามจุดขอบด้านนอก จะถูกสมมุติขึ้นเป็นอันดับแรก ดังแสดงในตารางที่ 1 ซึ่งแสดงลำดับของการสแกนเริ่มต้นสำหรับ trace แต่ละชนิด ในขณะเดียวกัน trace-point ที่ X ถัดไปจะถูกกำหนดขึ้น แล้ว 8-neighbour ก็จะถูกสแกนไปตามลำดับ ตำแหน่งของ X ที่จะเป็น trace-point ถัดไปจะสัมพันธ์กับตารางหน้าต่าง 3x3 ที่ผ่านมาแล้ว ในการกำหนดลำดับของการสแกนถัดไป ดังแสดงในตารางที่ 2

Type of trace	Start acanning sequence
Outer contour, clockwise	E H G F D A B C
Outer contour, anti-clockwise	D F G H E C B A
Inner contour , clockwise	G F D A B C E H
Inner contour , anti-clockwise	G H E C B A D F

ตารางที่ 1 แสดงลำดับของการสแกนครั้งแรก

Position of X in previous window	Outer contour		Inner contour	
	Clockwise	Anti-clockwise	Clockwise	Anti-clockwise
A	G F D A B C E H	E C B A D F G H	Same as outer	Same as outer
B	F D A B C E H G	H E C B A D F G	contour	contour
C	D A B C E H G F	G H E C B A D F	anti-clockwise	clockwise
D	H G F D A B C E	C B A D F G H E		
E	A B C E H G F D	F G H E C B A D		
F	E H G F D A B C	B A D F G H E C		
G	C E H D F D A B	A D F G H E C B		
H	B C E H G F D A	D F G H E C B A		

ตารางที่ 2 แสดงลำดับของการสแกนครั้งถัดไป

ยกตัวอย่างเช่น ถ้าตำแหน่งของ X ที่จะเป็น trace-point ถัดไปอยู่ในตำแหน่ง H ของตารางหน้าต่างที่ผ่านมา และเป็น trace ตามจุดขอบด้านนอก ทิศทางตามเข็มนาฬิกา ลำดับ

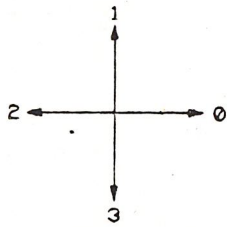
การสแกนของ 8-neighbour จะเป็น B,C,E,H,G,F,D,A ถ้าในกระบวนการสแกนพบจุดภาพใดๆ ที่มีค่าเป็น 1 และไม่ได้ถูก label เป็น trace-point แล้วลำดับในการ trace ของจุดขอบด้านในจะถูกใช้แทน ลำดับขั้นของการสแกนและการ Strip จะถูกกระทำซ้ำๆ เรื่อยไปจนกระทั่งจุดขอบถูก tace กลับไปยังตำแหน่งเริ่มต้น กระบวนการทั้งหมดจะถูกทำซ้ำไปเรื่อยๆ จนกระทั่งไม่มี trace-point และ corner-point เหลืออยู่เลย

ภาพที่ผ่านกระบวนการ Thinning ทั้ง 4 ขั้นตอนแล้ว จะถูกตรวจสอบเพื่อพิจารณาว่ายังมีจุดใดที่ไม่ใช่ break-point,end-point หรือ cross-point เหลืออยู่ กระบวนการ Fill, Delete, Label และ Strip ก็จะถูกกระทำซ้ำเรื่อยไปจนกระทั่งการ Thinning สิ้นสุดลง

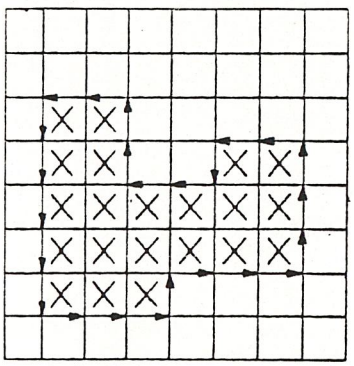
บทที่ 4

การทำสหสัมพันธ์ด้วยรหัสลูกโซ่สำหรับการกำหนดขอบเขตของวัตถุ

ในการตรวจสอบชนิดของวัตถุว่าเป็นวัตถุชนิดเดียวกันหรือไม่นั้น จะต้องกำหนดขอบเขตของวัตถุเพื่อนำมาใช้ในการพิจารณาแบ่งแยกเนื้อของวัตถุแต่ละชิ้นเสียก่อน สามารถหาได้จากการตรวจสอบจากรหัสลูกโซ่ของภาพวัตถุ โดยจะนำข้อมูลของจุดภาพที่เป็นขอบของภาพวัตถุ ซึ่งผ่านกระบวนการ Thinning มาแล้ว มาจัดให้อยู่ในรูปของรหัสลูกโซ่ สำหรับรหัสลูกโซ่ที่นิยมใช้กันมีอยู่ 2 ชนิดคือ แบบ 4-ทิศทาง(4-connectivity) และแบบ 8-ทิศทาง(8-connectivity)

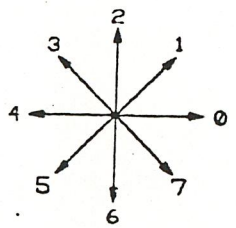


รูปที่ 4.1 แสดงทิศทางของรหัสลูกโซ่แบบ 4-ทิศทาง

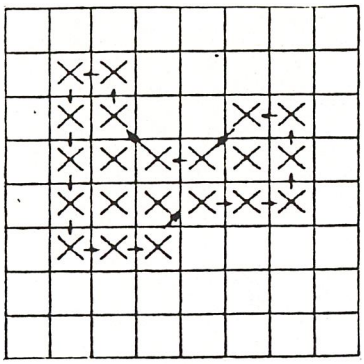


รูปที่ 4.2 แสดงการเข้ารหัสที่ขอบของภาพวัตถุ

รหัสลูกโซ่ของรูปที่ 4.2 จะเป็น 33333000100011122322112 2



รูปที่ 4.3 แสดงทิศทางของรหัสลูกโซ่แบบ 8-ทิศทาง

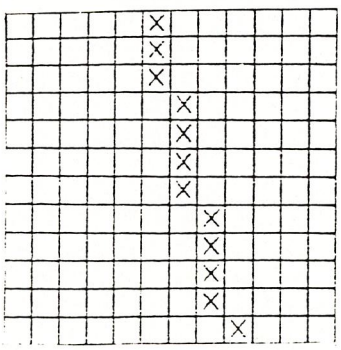


รูปที่ 4.4 แสดงการเข้ารหัสที่ขอบของภาพวัตถุ

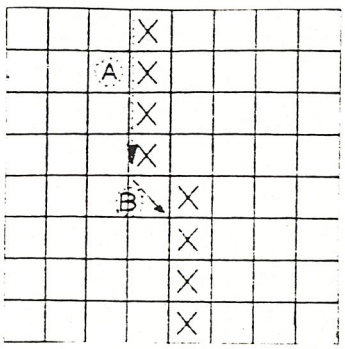
รหัสลูกโซ่ของรูปที่ 4.4 จะเป็น 66660010022454324

4.1 การหาตำแหน่งจุดเปลี่ยนแนวของเส้นตรง (Break point)

เส้นตรงจะประกอบขึ้นจากกลุ่มของจุดต่างๆ ที่เรียงต่อกัน ซึ่งเมื่อนำกลุ่มของจุดเหล่านี้มาเข้ารหัสลูกโซ่แล้ว จะมีแนวทิศทางแบ่งออกเป็น 2 แนวคือ ส่วนมากแล้วจุดจะมีทิศทางไปในทิศเดียวกัน เรียกว่า แนวทิศทางหลัก และจะมีอีกแนวทิศทางหนึ่งที่เปลี่ยนไปจากแนวทิศทางหลักไม่เกิน 45 องศา (หรือ 1 รหัสลูกโซ่) เรียกว่า แนวทิศทางรอง โดยปกติแล้วแนวทิศทางรองจะคั่นอยู่ระหว่างแนวทิศทางหลักเป็นระยะๆ



รูปที่ 4.5 แสดงกลุ่มของจุดที่เรียงต่อกันเป็นเส้นตรง



รูปที่ 4.6 แสดงแนวทิศทางหลัก:A และแนวทิศทางรอง:B

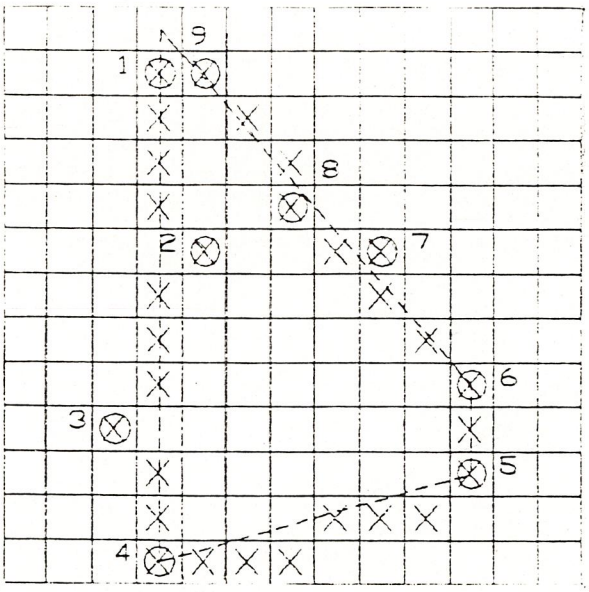
ซึ่งหลักในการหาจุดเปลี่ยนแนวของเส้นตรงทำได้โดย เมื่อพบจุดแรกที่มีการเปลี่ยนแนวทิศทางหลักหรือแนวทิศทางรองไปจากเดิมมากกว่าหรือเท่ากับ 90 องศา ให้ถือว่าจุดนั้นเป็นจุดที่เริ่มเปลี่ยนแนวของเส้น และ ต้องมีการตั้งค่าผิดพลาดสำหรับจำนวนของรหัสลูกโซ่ในแนวทิศทาง

รอง โดยค่าผิดพลาดที่กำหนดขึ้นนี้ จะขึ้นอยู่กับจำนวนกลุ่มจุดของแนวทิศทางหลัก จากการทดลองค่าผิดพลาดที่ยอมรับได้มีค่าดังแสดงในตารางที่ 1

จำนวนจุดของแนวเส้นหลัก	ค่าผิดพลาดที่ยอมรับได้ (จุด)
1-4	3
5 ขึ้นไป	2

ตารางที่ 1 แสดงค่าผิดพลาดที่ยอมรับได้จากการทดลอง

ในทางปฏิบัติการหาจุดเปลี่ยนแนวของเส้น มักจะพบปัญหาในเรื่องของจุดเว้าและแหว่งที่เกิดขึ้นตามขอบของภาพวัตถุอันเนื่องมาจากสัญญาณรบกวน (noise)ของข้อมูลภาพ และจากการตัดค่า threshold ทำให้เกิดจุดเปลี่ยนแนวเส้นขึ้นในส่วนที่เป็นเส้นตรงเดียวกัน ดังแสดงในรูปที่ 4.7



รูปที่ 4.7 แสดงขอบของภาพวัตถุที่มีการเว้าและแหว่ง
และแสดงจุดเปลี่ยนแนวเส้นที่เกิดขึ้นที่ 2,3,7 และ 8

จุดเว้าและแหว่งคือ จุดที่ 2 และ3 ถ้าพิจารณาไม่ดีแล้วก็จะถือเป็นจุดเปลี่ยนแนวไปด้วย ดังนั้นก่อนที่จะตรวจหาจุดเปลี่ยนแนวเส้น ควรทำการปรับแต่งขอบของภาพวัตถุให้ราบเรียบเสียก่อน

4.2 วิธีการปรับรหัสลูกโซ่ให้ราบเรียบ

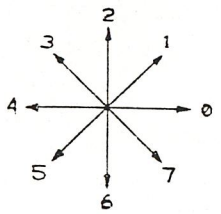
เงื่อนไขที่กำหนดขึ้นสำหรับการปรับรหัสลูกโซ่ให้ราบเรียบมี 2 ข้อคือ

1. จุดภาพที่เป็น noise จะอยู่ติดต่อกันไม่เกิน 3 จุดภาพ (ใช้เป็นตัวกำหนดขนาดความยาวของจำนวนจุดภาพ ที่ทำการตรวจสอบหาจุดภาพที่เป็น noise)

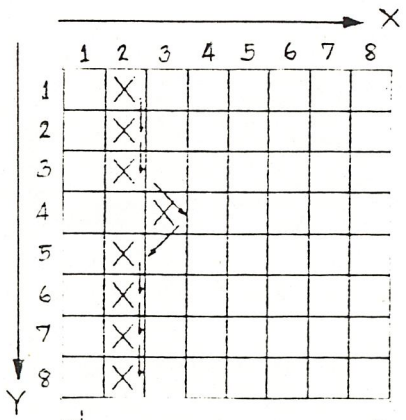
2. ค่าทิศทางรหัสลูกโซ่ของจุดภาพที่เป็น noise จะมีทิศทางแกว่งไปมาแบบสมมาตร (ใช้เป็นตัวบ่งบอกว่ามี noise เกิดขึ้นที่ขอบของข้อมูลภาพ)

<u>วิธีการตรวจสอบ noise ที่อยู่ตามขอบของภาพวัตถุจากรหัสลูกโซ่</u>

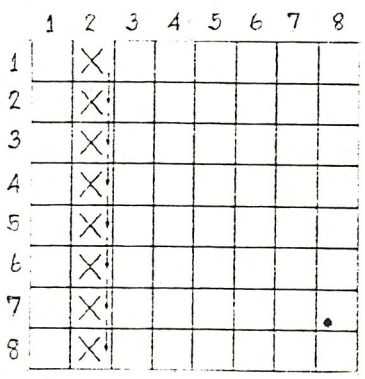
จากเงื่อนไขที่กำหนดไว้ 2 ข้อ จะนำมาใช้ในการตรวจสอบขอบของภาพวัตถุดังแสดงในรูปที่ 4.8 โดยใช้รหัสลูกโซ่แบบ 8-ทิศทาง ดังแสดงในรูปที่ 4.8ก และจากรูปที่ 4.8ข จะพบว่ารหัสลูกโซ่ของขอบภาพจากตำแหน่งที่ (2,2)ถึง(2,6) มีการแกว่งไปจากแนวทิศทางที่ 6 ของรหัสลูกโซ่อย่างสมมาตร กล่าวคือ แกว่งไปทางซ้ายและทางขวาของแนวทิศที่ 6 แล้วจึงกลับมายังแนวทิศที่ 6 เช่นเดิม แสดงว่าเกิด noise ที่ขอบของภาพวัตถุ ซึ่งในที่นี้คือจุดภาพตำแหน่งที่ (3,4) ดังนั้นต้องทำการปรับแต่งขอบของภาพวัตถุบริเวณนี้ใหม่ จนได้ขอบของภาพวัตถุใหม่ที่เป็นแนวเส้นตรงจากตำแหน่ง (2,1)ไปยังตำแหน่ง(2,6)ดังแสดงในรูปที่ 4.8ค



รูปที่ 4.8ก แสดงทิศทางของรหัสลูกโซ่แบบ 8 ทิศทาง



รูปที่ 4.8ข แสดงขอบของภาพวัตถุที่มี noise



รูปที่ 4.8ค แสดงขอบของภาพวัตถุที่ปรับให้ราบเรียบ

จากรูป จะสังเกตพบว่าทิศทางของรหัสลูกโซ่จะมีการแกว่งไปมาจากทิศทางเดิม ±45 องศา (หรือ ±1 รหัส) และสามารถสรุปทิศทางการแกว่งที่ใช้พิจารณาหาจุดภาพที่เป็น noise ได้ดังแสดงในตารางที่ 2

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

ตารางที่ 2 สรุปทิศทางการแกว่งไปมาแบบสมมาตรจากทิศทางหลัก

เมื่อทำการปรับแต่งขอบของภาพวัตถุให้ราบเรียบแล้ว ก็สามารถทำการตรวจหาจุดเปลี่ยนแนว (break point) ตามหัวข้อที่ 4.1 ได้ต่อไป

4.3 วิธีการกำจัดจุดหักมุมที่เป็นส่วนเกิน

หลังจากที่ทำการหาจุดเปลี่ยนแนวของเส้นตรง ซึ่งเป็นส่วนหนึ่งของขอบภาพวัตถุดังที่กล่าวมาแล้วในหัวข้อ 4.1 ต่อไปจะคำนวณหาค่ามุมที่เกิดจากจุดเปลี่ยนแนว 3 จุดที่อยู่ถัดกันไป เพื่อที่จะหาว่าจุดเปลี่ยนแนวจุดใดควรจะเป็นจุดมุมทั้งสี่ของระนาบสี่เหลี่ยม โดยทำการคำนวณหาระยะทางระหว่างจุดเปลี่ยนแนว 2 จุดที่อยู่ถัดกันไป จากการใช้สูตรตามสมการที่ 4.1

ตัวอย่างเช่นการคำนวณหาระยะทางของ A, Bและ C โดยที่ (x_n,y_n), (x_{n+1},y_{n+1}) และ (x_{n+2},y_{n+2}) เป็นตำแหน่งโคออร์ดิเนตของจุดเปลี่ยนแนว 3 จุดที่อยู่ถัดกัน ซึ่งหาได้จากสมการที่ 4.1 ดังนี้

$$A = \left[(x_{n+1} - x_n)^2 + (y_{n+1} - y_n)^2\right]^{1/2}$$
$$B = \left[(x_{n+2} - x_{n+1})^2 + (y_{n+2} - y_{n+1})^2\right]^{1/2}$$
$$C = \left[(x_{n+2} - x_n)^2 + (y_{n+2} - y_n)^2\right]^{1/2} \qquad (4.1)$$

จากนั้นสามารถหามุมที่อยู่ระหว่างเส้นตรง 2 เส้น ซึ่งเกิดจากจุดเปลี่ยนแนว 3 จุดได้จากการคำนวณโดยใช้กฎของ cos ดังสมการที่ 4.2

$$\theta = \cos^{-1}\left\{\frac{A^2 + B^2 - C^2}{2 \times A \times B}\right\} \qquad (4.2)$$

ถ้าค่า ที่หาได้มีค่ามากกว่า 140° จะถือว่าจุดเปลี่ยนแนวจุดนั้น ไม่ใช่จุดที่จะเป็นจุดมุมของระนาบสี่เหลี่ยม

4.4 การรวมจุดหักมุมที่อยู่ใกล้กันให้เป็นจุดเดียว

เมื่อกำจัดจุดหักมุมที่เป็นส่วนเกินออกไปแล้ว จำนวนจุดที่เหลืออยู่จะลดลง และมีบางจุดที่มีตำแหน่งใกล้เคียงกันจนสามารถที่จะยุบรวมกันให้เป็นจุดเดียวได้ โดยทำการหาค่าเฉลี่ยของระยะระหว่างจุดหักมุม 2 จุดนั้น ซึ่งค่าที่คำนวณได้จะเป็นค่าที่แสดงถึงตำแหน่งใหม่ของจุดหักมุมและจุดนั้นจะถือเป็นจุดมุมของระนาบสี่เหลี่ยม

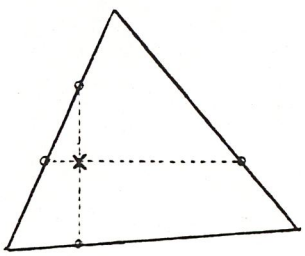
บทที่ 5

การแยกแยะรูปร่างของวัตถุ

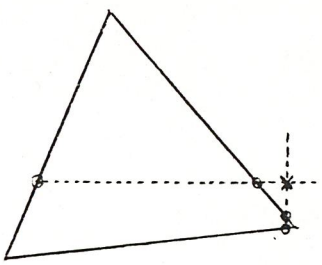
เนื่องจากข้อมูลภาพประกอบไปด้วยวัตถุหลายชิ้นด้วยกัน ดังนั้นก่อนที่เราจะสามารถทราบได้ว่าวัตถุแต่ละชิ้นมีรูปร่างอย่างไร เราจะต้องรู้เสียก่อนว่ามีจุดใดบ้างที่เป็นเนื้อของวัตถุแต่ละชิ้นนั้น โดยอาศัยรหัสลูกโซ่ของขอบภาพของวัตถุชิ้นดังกล่าวช่วยในการพิจารณา ดังจะได้กล่าวถึงในหัวข้อถัดไป

5.1 การพิจารณาแยกส่วนที่เป็นเนื้อของวัตถุแต่ละชิ้น

เนื่องจากรหัสลูกโซ่ของขอบภาพของวัตถุชิ้นหนึ่งๆ เป็นข้อมูลที่สำคัญที่สามารถบอกให้ทราบถึงขอบเขตของวัตถุชิ้นนั้นได้ ดังนั้นจุดภาพที่จะพิจารณาได้ว่าเป็นเนื้อของวัตถุชิ้นใดจะต้องอยู่บนหรือถูกล้อมรอบด้วยรหัสลูกโซ่ของขอบภาพของวัตถุชิ้นนั้น กล่าวคือ จะต้องทำการพิจารณาลูกโซ่ตั้งแต่ต้นไปจนครบวงรอบว่า จุดทุกจุดบนลูกโซ่นั้นได้ล้อมรอบจุดที่พิจารณา ทั้งด้านบน, ด้านล่าง,ด้านซ้าย และด้านขวา ครบทุกด้านหรือไม่ ดังรูป



รูปที่ 5.1 แสดงจุดภาพที่เป็นเนื้อของวัตถุ



รูปที่ 5.2 แสดงจุดภาพที่ไม่ใช่เนื้อของวัตถุ

จากรูปจะพบว่า จุดภาพในรูปที่ 5.1 ถูกปิดล้อมครบทุกด้าน จึงสรุปได้ว่าเป็นเนื้อของวัตถุส่วนจุดภาพในรูปที่ 5.2 ไม่ได้ถูกปิดล้อมทางด้านบนและด้านขวา จึงสรุปได้ว่าไม่ใช่เนื้อของวัตถุ

แต่เนื่องจาก ถ้าต้องพิจารณารหัสลูกโซ่ว่าล้อมรอบจุดภาพทุกจุดหรือไม่ จะทำให้เสียเวลาเพราะมีบางจุดภาพที่เราสามารถบอกได้ทันทีว่า ไม่ใช่เนื้อของวัตถุ คือ จุดที่อยู่สูงกว่าขอบบนสุดของวัตถุ, จุดที่อยู่ต่ำกว่าขอบล่างของวัตถุ, จุดที่อยู่ทางซ้ายของขอบซ้ายสุดของวัตถุ

และจุดที่อยู่ทางขวาของขอบขวาสุดของวัตถุ ซึ่งเป็นไปไม่ได้ที่จุดเหล่านี้จะถูกปิดล้อมด้วยรหัสลูกโซ่ของวัตถุชิ้นนั้น ดังนั้น ถ้าเราไม่ต้องเสียเวลามาพิจารณาจุดภาพเหล่านี้ จะทำให้ประหยัดเวลาในการคำนวณมากยิ่งขึ้น

เมื่อเราสามารถกำหนดได้แล้วว่า มีจุดภาพใดบ้างที่เป็นส่วนของเนื้อของวัตถุชิ้นที่เรากำลังพิจารณาอยู่ เราก็สามารถที่จะนำจุดภาพเหล่านั้นไปคำนวณ โดยใช้วิธีการเปรียบเทียบแกนหลักของวัตถุ เพื่อให้สามารถแยกแยะรูปร่างของวัตถุได้ต่อไป ซึ่งจะได้อธิบายไว้ในหัวข้อถัดไป

5.2 การเปรียบเทียบแกนหลัก

การเปรียบเทียบแกนหลัก (Principal Axis Comparison) เป็นการนำหลักการวิเคราะห์องค์ประกอบหลัก(Principal Component Analysis) มาใช้ประโยชน์เพื่อแยกความแตกต่างของวัตถุแต่ละชนิด ซึ่งทฤษฎีการวิเคราะห์องค์ประกอบหลักนั้น ได้ถูกเขียนขึ้นในปีค.ศ.1923 โดย Hotelling และ ในปี ค.ศ.1964 โดย Searle และได้ถูกนำไปประยุกต์ใช้งานในด้านต่างๆ เป็นเวลานานหลายปีมาแล้ว

การวิเคราะห์องค์ประกอบหลัก เป็นเทคนิคทางสถิติที่อยู่บนพื้นฐานของ Variance และ Covariance ของเซ็ทข้อมูล (Variance หรือ ความแปรปรวน เป็นการวัดการแตกกระจายที่อยู่ภายในหนึ่งตัวแปร ส่วน Covariance เป็นการวัดการแตกกระจายในระหว่างสองตัวแปร) โดยที่การวิเคราะห์องค์ประกอบหลัก จะเป็นการแปลงแบบเชิงเส้นของความแปรปรวนของข้อมูลที่มีตัวแปร m ตัว(m องค์ประกอบ) ให้เป็นเซ็ทข้อมูลที่มีตัวแปร n ตัว(n องค์ประกอบ) โดยจุดประสงค์ของทฤษฎีในการแปลง ก็คือ ต้องการได้คืนมาซึ่งความแปรปรวนทั้งหมด เพื่อรักษาความแปรปรวนทุกๆอย่างที่ต้องการเอาไว้

หลังจากการแปลงแล้ว องค์ประกอบแรกจะมีความแปรปรวนสูงสุดจากความแปรปรวนทั้งหมด องค์ประกอบที่ 2 จะมีความแปรปรวนสูงสุดจากความแปรปรวนที่เหลือ และไล่ลงไปเรื่อยสำหรับองค์ประกอบถัดไป ซึ่งในองค์ประกอบหลังๆ จะมีความแปรปรวนน้อยจนอาจถือได้ว่าไม่มีค่า และตัดทิ้งไปได้ ถ้าหากมีการตัดองค์ประกอบหลังๆดังกล่าวทิ้งไป ก็จะเป็นการลดขนาดของมิติของภาพ ทำให้จำนวนของความแปรปรวนหลังการแปลงมีค่าน้อยกว่า จำนวนความแปรปรวนของข้อมูลเดิม

เมื่อนำหลักการวิเคราะห์องค์ประกอบหลักดังกล่าว มาประยุกต์ใช้กับการแยกความแตกต่างของภาพวัตถุ ก็คือ พยายามที่จะลดจำนวนความแปรปรวนของจุดภาพแต่ละจุดของวัตถุ ให้เหลือเพียง 2 องค์ประกอบ คือ ความแปรปรวนในแนวแกนหลัก และแนวแกนรอง โดยมีเงื่อนไขอยู่ว่า แนวแกนหลัก และแนวแกนรองจะต้องตั้งฉากซึ่งกันและกัน ความแปรปรวนในแนวแกนหลักนี้เรียกว่า Maximum Eigenvalue (λmax) และความแปรปรวนในแนวแกนรองเรียกว่า Minimum Eigenvalue (λmin) ซึ่ง eigenvalue ทั้งสองค่านี้ สามารถใช้เป็นลักษณะเฉพาะของวัตถุ แทนตัววัตถุใดๆได้เป็นอย่างดี

ในการหาค่า eigenvalue ทั้งสอง จำเป็นจะต้องหาค่า Covariance เสียก่อน ซึ่งค่า Covariance นี้แสดงได้ในลักษณะของเมตริกซ์ 2x2 เรียกว่า Covariane matrix ดังนี้

$$C = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{5.1}$$

โดยค่า a,b,c และ d คำนวณหาได้จากความแปรปรวนระหว่าง 2 ตัวแปร คือ พิกัด xและ y ของจุดใดๆที่เป็นตัวภาพวัตถุ ดังนี้

$$a = \frac{\sum (x_i - \bar{x})^2}{N} \tag{5.2}$$

$$b = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{N} \tag{5.3}$$

$$c = \frac{\sum (y_i - \bar{y})(x_i - \bar{x})}{N} \tag{5.4}$$

$$d = \frac{\sum (y_i - \bar{y})^2}{N} \tag{5.5}$$

โดยที่ x_i, y_i คือ ค่าพิกัด x และ y ณ จุดใดๆที่เป็นตัวภาพของวัตถุ

$\bar{x}, \bar{y}$ คือ ค่าเฉลี่ยของพิกัด x และ y ตามลำดับ

N คือ จำนวนจุดภาพที่เป็นตัวภาพวัตถุ

จากนั้น จึงคำนวณหาค่า eigenvalue โดยใช้สมการ

$$\det(C - \lambda I) = 0 \tag{5.6}$$

โดยที่ λ คือ ค่า eigenvalue

I คือ Identity matrix

จากสมการที่ 5.1และ 5.6 เขียนใหม่ได้เป็น

$$\begin{vmatrix} a-\lambda & b \\ c & d-\lambda \end{vmatrix} = 0 \qquad (5.7)$$

ทำการแก้สมการเพื่อหาค่า λ ได้ดังนี้

$$\begin{aligned} (a-\lambda)(d-\lambda)-bc &= 0 \\ ad-a\lambda-d\lambda-\lambda^2-bc &= 0 \\ \lambda^2-(a+d)\lambda+(ad-bc) &= 0 \qquad (5.8) \end{aligned}$$

จะได้ λ ออกมา 2 ค่า คือ

$$\lambda = \frac{(a+d)+\sqrt{(a+d)^2-4(ad-bc)}}{2} \qquad (5.9)$$

ซึ่งก็คือ ค่า Maximum Eigenvalue และ

$$\lambda = \frac{(a+d)-\sqrt{(a+d)^2-4(ad-bc)}}{2} \qquad (5.10)$$

ซึ่งก็คือ ค่า Minimum Eigenvalue

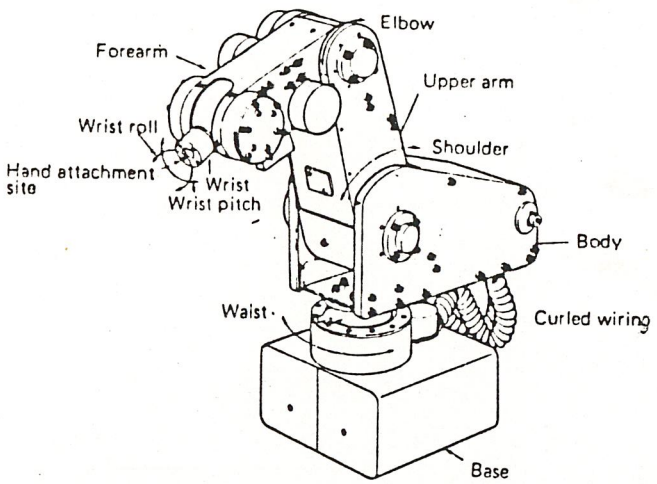
เมื่อได้ค่า eigenvalue ทั้งสองแล้ว จะต้องนำไปเปรียบเทียบกับค่า eigenvalue ของวัตถุมาตรฐานที่เก็บค่าอยู่ในหน่วยความจำ เพื่อตรวจสอบว่าวัตถุที่เราหาค่า eigenvalue ออกมาได้นั้นเป็นวัตถุชนิดเดียวกับวัตถุมาตรฐานหรือไม่ ซึ่งวิธีการในการเปรียบเทียบ และผลของการทดลองจะได้กล่าวถึงในบทที่ 7

บทที่ 6

การควบคุมทิศทางการเคลื่อนไหวของแขนกล

6.1 ส่วนประกอบของแขนกล รุ่น RM-501

1) Robot Unit (RM-501) ประกอบด้วยส่วนต่างๆ ดังรูปที่ 6.1



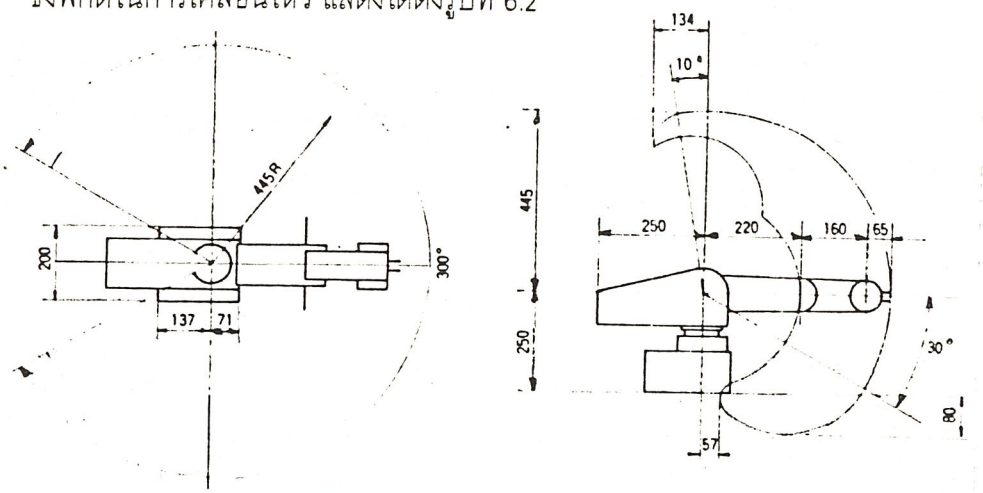
รูปที่ 6.1 ลักษณะภายนอกของ Robot Unit

Robot Unit มีคุณสมบัติดังที่แสดงไว้ในตารางที่ 1

โครงสร้าง		มี 5 degree of freedom โดยที่ข้อต่อต่างๆเป็นแบบตามแนวตั้ง
พิกัดในการเคลื่อนไหวของข้อต่อต่างๆ	Waist rotation	300°
	Shoulder rotation	130°
	Elbow rotation	90°
	Wrist rotation	± 90°
	Wrist roll	± 180°
น้ำหนักสูงสุดที่หยิบได้		1.2 kg (รวมน้ำหนักของ Hand ด้วย)
ความเร็วสูงสุด		400 mm/sec
ค่าความคลาดเคลื่อนของตำแหน่ง		± 0.5 mm

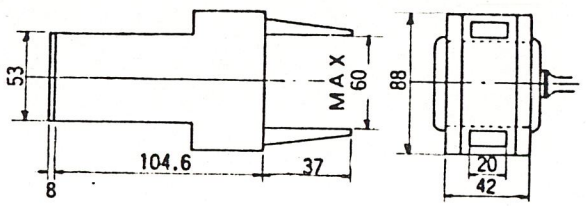
ตารางที่ 1 คุณสมบัติของ Robot Unit

ซึ่งพิกัดในการเคลื่อนไหว แสดงได้ดังรูปที่ 6.2



รูปที่ 6.2 พิกัดในการเคลื่อนไหวของข้อต่อต่าง ๆ

2) Standard Hand ใช้มอเตอร์ชนิด DC servomotor มีพิกัดของแรงดันเท่ากับ 24 V DC และมีพิกัดกำลังเท่ากับ 15 W มีน้ำหนัก 700 กรัม และมีแรงในการบีบวัตถุสูงสุด 4 kg มีลักษณะภายนอก ดังแสดงไว้ในรูปที่ 6.3



รูปที่ 6.3 ลักษณะภายนอกของ Standard Hand

3) Drive Unit (DU) คุณสมบัติที่สำคัญมีดังนี้

- ระบบการควบคุม เป็นแบบ PTP (Point to Point)
- สามารถตั้งระดับความเร็วได้ตั้งแต่ 0-9 คือ มีความเร็วตั้งแต่ 40-400 mm/sec (เมื่อเปิดเครื่อง ระดับความเร็วจะถูกตั้งไว้ที่ระดับ 4 คือ 200 mm/sec)
- สามารถติดต่อกับระบบภายนอกได้ 2 ระบบคือ แบบขนาน ผ่านทาง centro base และ แบบอนุกรม ผ่านทาง RS-232C
- มีสัญญาณ I/O ภายนอก เป็น 8 บิต ทั้ง Input และ Output

- สามารถรับคำสั่งจาก personal computer ได้โดยใช้โปรแกรมภาษาประยุกต์ เช่น BASIC,ASSEMBLY และ C
- Connection Cable ใช้สายต่อแบบ parallel ที่แปลงจากระบบ Centronic เป็นระบบ 25 เข็ม เหมือนกับ printer port ของ personal computer

6.2 รูปแบบคำสั่งที่ใช้

จากคุณสมบัติต่าง ๆ ของแขนกลรุ่น RM-501 ดังกล่าว ได้ทำการทดลองควบคุมการเคลื่อนไหวของแขนกล โดยใช้โปรแกรมประยุกต์ภาษา C เพื่อส่งคำสั่งต่าง ๆ ไปยัง Drive Unit ผ่านทาง printer port โดยใช้คำสั่งพื้นฐาน ดังนี้

- คำสั่ง Nest เพื่อกำหนดตำแหน่งเริ่มต้นทาง mechanics ซึ่งเป็นตำแหน่งที่ส่วน Body ของแขนกลหมุนไปทางขวาจนสุด ,ส่วน Upper Arm ยกขึ้นจนสุด ,ส่วน Fore Arm และ Hand พับลงจนสุด

มีรูปแบบของคำสั่งคือ " NT "

- คำสั่ง Home เพื่อตั้งให้ตำแหน่งปัจจุบันเป็นตำแหน่งอ้างอิง

มีรูปแบบของคำสั่งคือ " HO "

- คำสั่ง Origin เพื่อควบคุมให้แขนกลเคลื่อนที่กลับไปอยู่ในตำแหน่งอ้างอิง

มีรูปแบบของคำสั่งคือ " OG "

- คำสั่ง Move I เพื่อควบคุมให้แต่ละข้อต่อหมุนไป เป็นจำนวน step เท่ากับที่ระบุโดยพารามิเตอร์ a_1 ถึง a_5

มีรูปแบบของคำสั่งคือ " MI a_1,a_2,a_3,a_4,a_5,a_6"

โดยที่ a_1 เป็นจำนวน step ในการหมุนของ Waist ซึ่ง 1 step = 0.025°

a_2 เป็นจำนวน step ในการหมุนของ Shoulder ซึ่ง 1 step = 0.025°

a_3 เป็นจำนวน step ในการหมุนของ Elbow ซึ่ง 1 step = 0.025°

a_4,a_5 เป็นจำนวน step ที่ต้องใช้ประกอบกันในการหมุนของ Wrist ซึ่ง 1 step = 0.075°

a_6 เป็น 0 ตลอดเวลา

ซึ่งพารามิเตอร์ a_1 ถึง a_5 สามารถมีค่าได้ทั้งบวกและลบ โดยที่เครื่องหมายของ a_1, a_2, a_3 บอกถึงทิศทางการหมุนของข้อต่อ Body, Shoulder และ Elbow ตามลำดับ ดังตารางที่ 2

พารามิเตอร์	+	--
a_1	ตามเข็มนาฬิกา	ทวนเข็มนาฬิกา
a_2	หมุนขึ้น	หมุนลง
a_3	หมุนขึ้น	หมุนลง

ตารางที่ 2 พารามิเตอร์และทิศทางการเคลื่อนที่ของแต่ละข้อต่อ

และเครื่องหมายของ a_4 และ a_5 ใช้ประกอบกันในการกำหนดทิศทางการหมุนของ Wrist ดังตารางที่ 3

a_4	a_5	การเคลื่อนที่ของ Wrist
+	+	ตามเข็มนาฬิกา
--	--	ทวนเข็มนาฬิกา
+	--	หมุนขึ้น
--	+	หมุนลง

ตารางที่ 3 พารามิเตอร์และทิศทางการเคลื่อนที่ของ Wrist

- คำสั่ง Grip Closed เพื่อควบคุมให้ Hand หนีบวัตถุ

มีรูปแบบของคำสั่งคือ " GC "

- คำสั่ง Grip Opened เพื่อควบคุมให้ Hand กางออก

มีรูปแบบของคำสั่งคือ " GO "

- คำสั่ง Reset เพื่อทำการ reset เมื่อเกิด error ขึ้น

มีรูปแบบของคำสั่งคือ " RS "

จากข้อมูลเบื้องต้นดังกล่าว เนื่องจาก RM-501 นี้ มีการควบคุมแบบ Point to Point จึงได้กำหนดให้มีตำแหน่งอ้างอิง เพื่อเป็นจุดเริ่มต้นในการเคลื่อนที่ไปยังตำแหน่งเป้าหมาย เพื่อให้ง่ายต่อการคำนวณหาค่าของมุมที่ข้อต่อต่าง ๆ จะต้องหมุนไป ซึ่งในการคำนวณนี้ต้องทราบถึงระยะห่างของข้อต่อต่าง ๆ ได้แก่

Shoulder อยู่สูงจากพื้น	25	mm
Elbow อยู่ห่างจาก Shoulder	22	mm
Wrist อยู่ห่างจาก Elbow	16	mm
ปลายของ Hand อยู่ห่างจาก Wrist	21.46	mm

กำหนดให้ h เป็นระยะห่างระหว่างปลายของ Hand กับพื้น

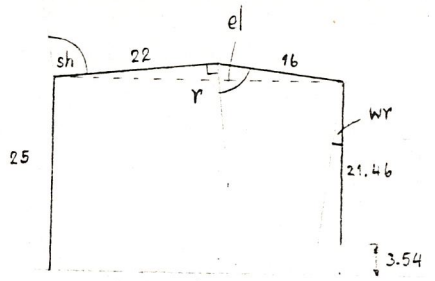
r เป็นรัศมีที่ปลายของ Hand อยู่ห่างจากจุดหมุนของ Body

l เป็นระยะห่างระหว่าง Shoulder กับปลายของ Hand

6.3 วิธีการคำนวณหาค่าของมุมสำหรับข้อต่อต่าง ๆ

เนื่องจาก RM-501 นี้ มี 5 degrees of freedom จึงทำให้มีวิธีการคำนวณหาค่าของมุมต่างๆ ได้หลายลักษณะ แต่เพื่อให้ง่ายต่อการคำนวณ จึงได้กำหนดค่าของมุมใดมุมหนึ่งให้เป็นมุมฉาก หรือ 180° ซึ่งมีผลทำให้สามารถหาค่าของมุมต่าง ๆ ได้ 5 ลักษณะ คือ

1. กำหนดให้ Hand ตั้งฉากกับพื้น ดังรูป



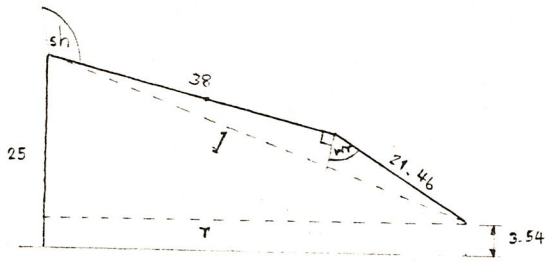
$$sh = 100 - \cos^{-1}\left\{\frac{22^2 + l^2 - 16^2}{2 \times 22 \times l}\right\} - \tan^{-1}\left\{\frac{21.46 + h - 25}{r}\right\}$$

$$el = \cos^{-1}\left\{\frac{22^2 + 16^2 - l^2}{2 \times 22 \times 16}\right\} - 90$$

$$wr = \cos^{-1}\left\{\frac{16^2 + l^2 - 22^2}{2 \times 16 \times l}\right\} + \tan^{-1}\left\{\frac{r}{21.46 + h - 25}\right\} - 90$$

การคำนวณในลักษณะนี้ ใช้ได้กับขนาดของ r ที่อยู่ระหว่าง 27.203 ถึง 38 cm

2. กำหนดให้ Fore Arm ขนานกับ Upper Arm (el = 90°) ดังรูป



$$l = \sqrt{r^2 + (25-h)^2}$$

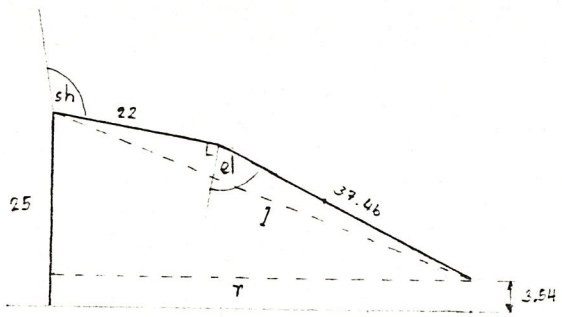
$$sh = 190 - \cos^{-1}\left\{\frac{38^2 + l^2 - 21.46^2}{2 \times 38 \times l}\right\} - \tan^{-1}\left\{\frac{r}{25-h}\right\}$$

$$el = 90$$

$$wr = \cos^{-1}\left\{\frac{38^2 + 21.46^2 - l^2}{2 \times 38 \times 21.46}\right\} - 90$$

การคำนวณในลักษณะนี้ ใช้ได้กับขนาดของ r ตั้งแต่ 38 ถึง 55.45 cm

3.กำหนดให้ Hand ขนานกับ Fore Arm (wr = 90°) ดังรูป



$$l = \sqrt{r^2 + (25-h)^2}$$

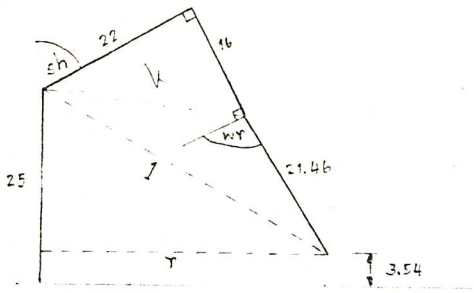
$$sh = 190 - \cos^{-1}\left\{\frac{22^2 + l^2 - 37.46^2}{2 \times 22 \times l}\right\} - \tan^{-1}\left\{\frac{r}{25-h}\right\}$$

$$el = \cos^{-1}\left\{\frac{22^2 + 37.46^2 - l^2}{2 \times 22 \times 37.46}\right\} - 90$$

$$wr = 90$$

การคำนวณในลักษณะนี้ ใช้ได้กับขนาดของ r ที่อยู่ระหว่าง 37.77 ถึง 55.45 cm

4.กำหนดให้ Fore Arm ตั้งฉากกับ Upper Arm (el = 0°)ดังรูป



$$l = \sqrt{r^2 + (25-h)^2}$$

$$k = \sqrt{22^2 + 16^2}$$

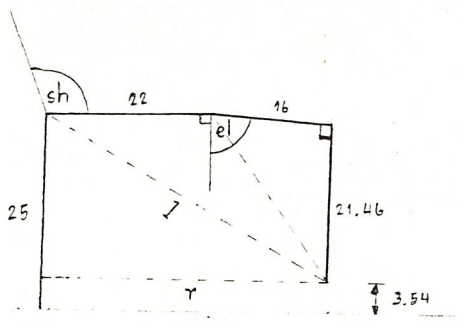
$$sh = 190 - \tan^{-1}\left\{\frac{16}{22}\right\} - \cos^{-1}\left\{\frac{k^2 + l^2 - 21.46^2}{2 \times k \times l}\right\} - \tan^{-1}\left\{\frac{r}{25-h}\right\}$$

$$el = 0$$

$$wr = \cos^{-1}\left\{\frac{k^2 + 21.46^2 - l^2}{2 \times k \times 21.46}\right\} + \tan^{-1}\left\{\frac{22}{16}\right\} - 90$$

การคำนวณในลักษณะนี้ ใช้ได้กับขนาดของ r ที่ไม่เกิน 37.77 cm

5.กำหนดให้ Hand ตั้งฉากกับ Fore Arm (wr = 0°) ดังรูป



$$l = \sqrt{r^2 + (25-h)^2}$$

$$k = \sqrt{16^2 + 21.46^2}$$

$$sh = 190 - \cos^{-1}\left\{\frac{22^2 + l^2 - k^2}{2 \times 22 \times l}\right\} - \tan^{-1}\left\{\frac{r}{25-h}\right\}$$

$$el = \cos^{-1}\left\{\frac{22^2 + k^2 - l^2}{2 \times 22 \times k}\right\} + \tan^{-1}\left\{\frac{21.46}{16}\right\} - 90$$

$$wr = 0$$

การคำนวณในลักษณะนี้ ใช้ได้กับขนาดของ r ตั้งแต่ 0 ถึง 38 cm

จากวิธีการคำนวณทั้ง 5 ลักษณะดังกล่าวนี้ เนื่องจากแขนกลที่ใช้ มอเตอร์ที่ควบคุมการหมุนของ Wrist ไม่สามารถใช้งานได้ จึงจำเป็นต้องเลือกใช้วิธีการคำนวณตามข้อที่ 5 คือกำหนดให้ส่วน Hand ตั้งฉากกับ Fore Arm เพื่อหลีกเลี่ยงการหมุนของ Wrist และให้มีช่วงการทำงานที่กว้างที่สุด

6.4 การแปลงระยะของจุดภาพให้เป็นระยะทางจริง

เนื่องจาก ในการกำหนดตำแหน่งของวัตถุเป้าหมายจากข้อมูลภาพ จะได้ระยะทางของวัตถุออกมาในรูปของระยะห่างระหว่างจุดภาพ ซึ่งจำเป็นจะต้องมีการแปลงระยะของจุดภาพดังกล่าว ให้ออกมาในรูปของระยะทางที่สามารถวัดได้จริง เช่น เซนติเมตร หรือ มิลลิเมตร กล่าวคือจะต้องมีค่า Factor สำหรับคูณกับค่าระยะของจุดภาพ ให้ได้เป็นค่าของระยะทางจริง ซึ่งค่า Factor นี้ขึ้นอยู่กับความสูงของกล้องจากระนาบ เมื่อระยะความสูงของกล้องเปลี่ยนไป ขนาดภาพของระนาบ ก็จะเปลี่ยนไปด้วย ดังนั้น เราจะสามารถหาค่าของ Factor ได้โดยการนำขนาดภาพของระนาบที่คำนวณได้ และ ระยะความสูงของกล้อง ไปเทียบอัตราส่วนกับค่า Factor มาตรฐาน ก็จะสามารถหาค่า Factor ที่ต้องการได้

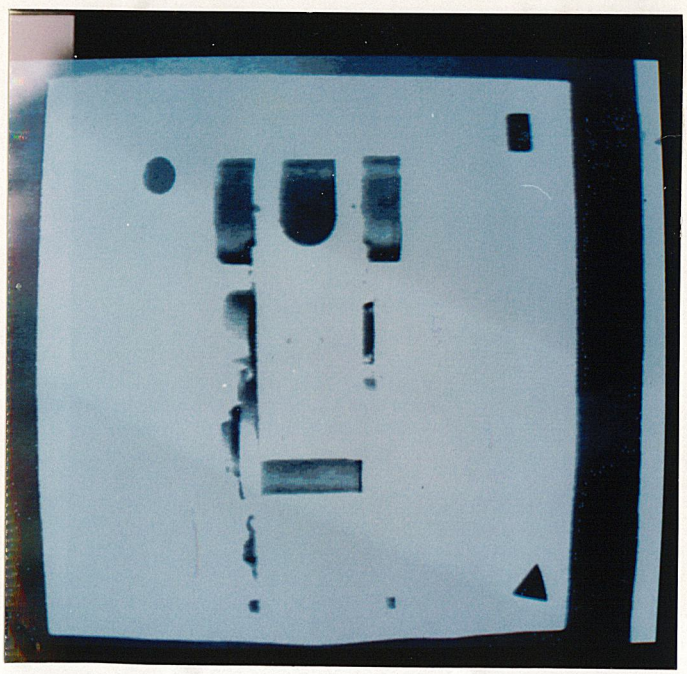
บทที่ 7

ผลการทดลองและข้อเสนอแนะ

7.1 ผลการทดลองและสรุป

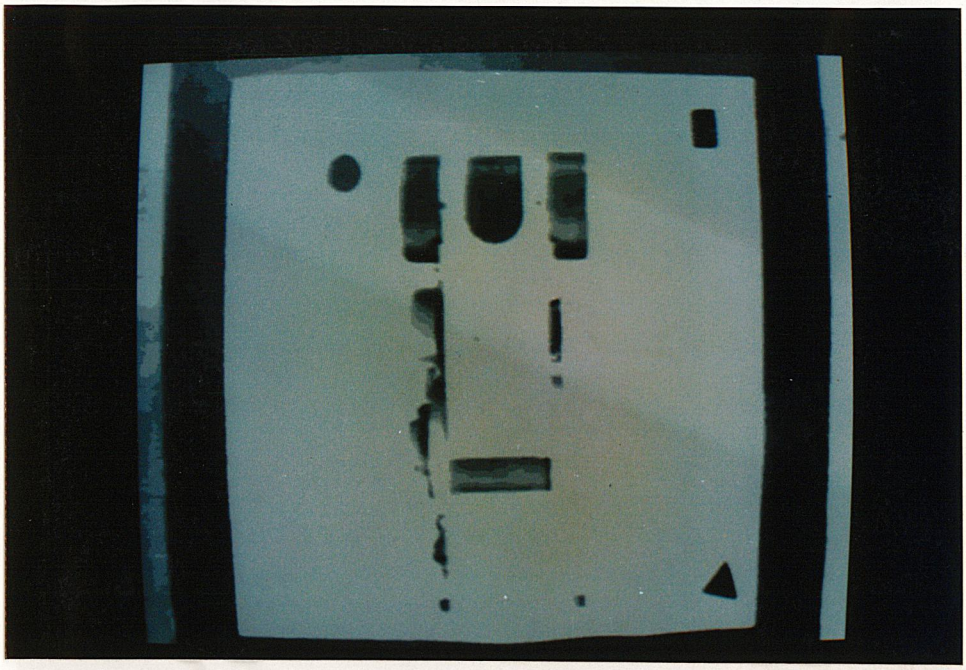
จากการทดลองได้ทำการรับภาพของ แขนกล และ วัตถุต่างๆ ที่ถูกวางอยู่บนระนาบสี่เหลี่ยมเข้ามาโดยใช้โทรทัศน์วงจรปิด และทำการแปลงสัญญาณที่รับเข้ามาให้เป็น digital image ที่มีขนาด 512x512 จุดภาพ มีระดับความเข้ม 256 ระดับ โดยใช้การ์ด digitizer ซึ่งเป็นผลงานวิจัยระดับปริญญาโทของนาย บัณฑิต สุมนวัฒนเดช ภาควิชา เทคโนโลยีวัดคุมทางอุตสาหกรรม ตามเอกสารอ้างอิง[6]

ซึ่งจากการทดลองเก็บภาพ ได้ข้อมูลภาพที่ถูกรบกวนโดยแสงสว่างรอบข้าง และ ผลของ noise ที่รบกวนการ์ด digitizer ขณะทำการ sampling ข้อมูล แสดงไว้ในรูปที่ 7.1

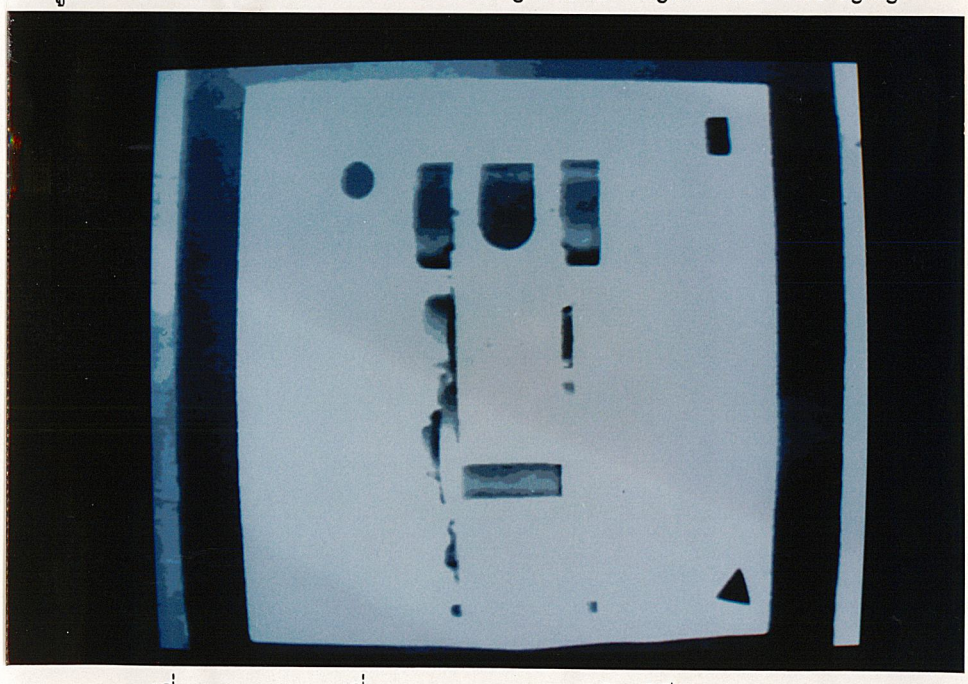


รูปที่ 7.1 แสดงถึงข้อมูลภาพที่ยังไม่ผ่านการ Smoothing

จากนั้น ได้นำข้อมูลภาพที่ได้ มาผ่านกระบวนการ Smoothing โดยเปรียบเทียบระหว่างวิธี Neighborhood Averaging และ Median Filtering ซึ่งได้ผลการทดลองดังรูป



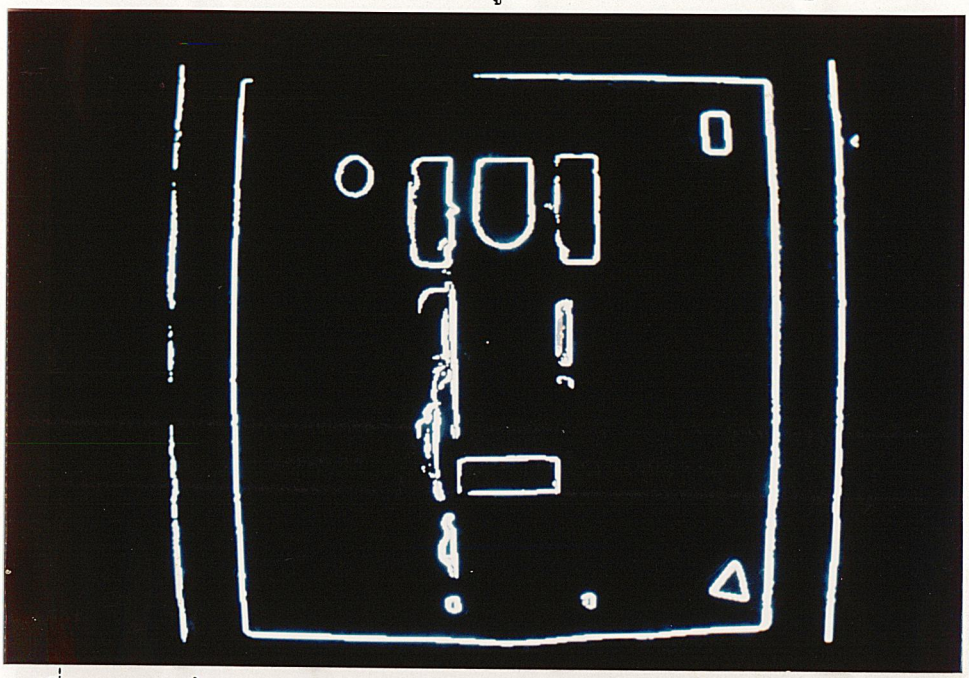
รูปที่ 7.2 แสดงภาพที่ผ่านการ Smoothing โดยวิธี Neighborhood Averaging



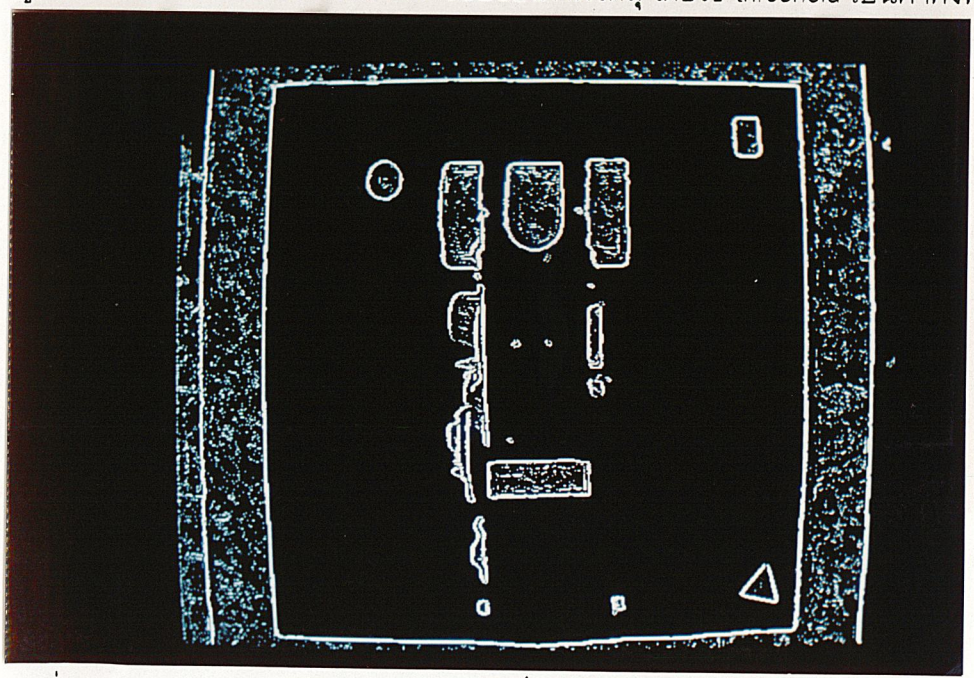
รูปที่ 7.3 แสดงภาพที่ผ่านการ Smoothing โดยวิธี Median Filtering

จากรูปจะเห็นได้ว่า วิธี Median Filtering สามารถขจัด noise ออกไปได้ดีกว่าวิธี Neighborhood Averaging โดยไม่ทำให้ข้อมูลที่สำคัญ คือ จุดภาพบริเวณขอบของภาพวัตถุมัวลงไป ดังนั้น จึงเลือกใช้วิธี Median Filtering เป็นวิธีหลักที่ใช้สำหรับการ Smoothing

จากนั้น ได้ทำการตรวจหาขอบของภาพวัตถุ โดยคำนวณหาค่า Gx และ Gy ด้วย Sobel Operater ตามรูปที่ 2.1 และ 2.2 แล้วนำค่า Gx และ Gy นั้นไปหาค่า gradient ตามสมการที่ 2.3 จากนั้นจึงสร้างเป็น binary image โดยเปรียบเทียบระหว่าง วิธีที่ใช้ threshold เป็นค่าคงที่ กับวิธีที่ใช้ threshold ที่ปรับค่าได้ ซึ่งได้ผลการทดลองดังรูปที่ 7.4 และ 7.5 ตามลำดับ



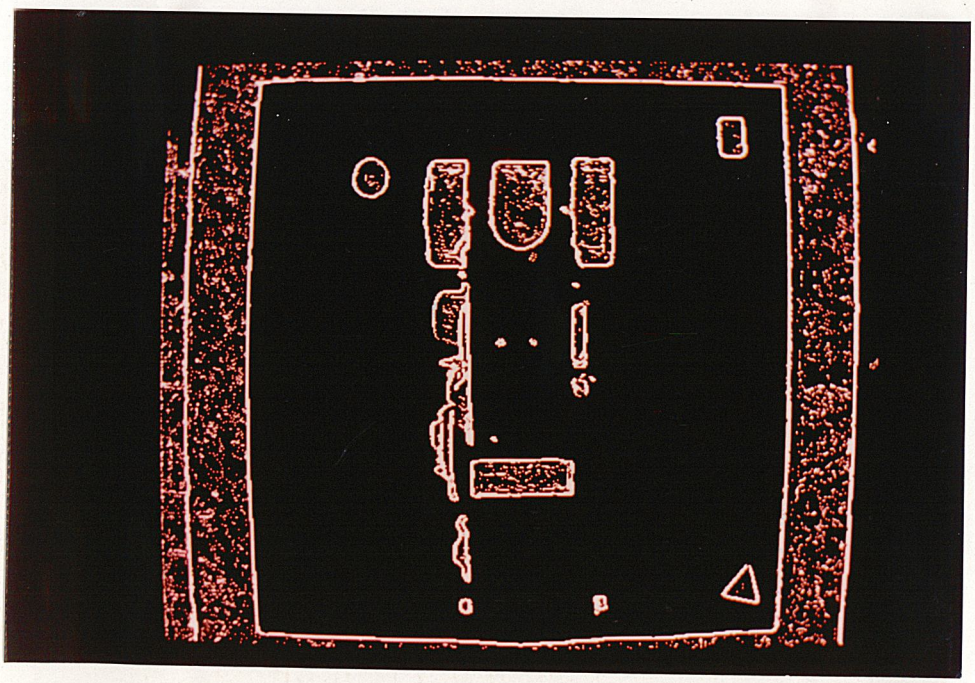
รูปที่ 7.4 แสดงถึงผลของการตรวจหาขอบของภาพวัตถุ โดยใช้ threshold เป็นค่าคงที่



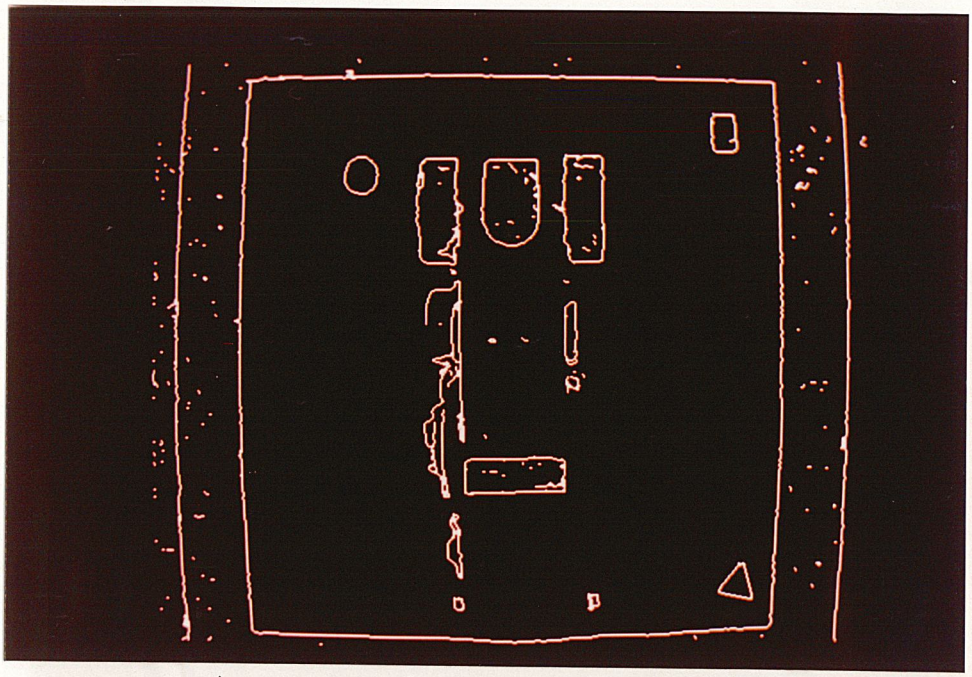
รูปที่ 7.5 แสดงถึงผลของการตรวจหาขอบของภาพวัตถุ โดยใช้ threshold ที่ปรับค่าได้

จากรูปจะเห็นได้ว่า การใช้ threshold เป็นค่าคงที่ จะทำให้ขอบของภาพที่ได้ค่อนข้างหนา และไม่ต่อเนื่อง ซึ่งจะเป็นอุปสรรคสำคัญในการจดจำรูปร่างวัตถุ เมื่อเปรียบกับการใช้ค่า threshold ที่ปรับค่าได้ ซึ่งทำให้ได้ขอบของภาพที่บางกว่าและต่อเนื่องกว่า แม้จะมีส่วนอื่นที่ไม่ใช่ขอบภาพเกินเข้ามาด้วย แต่ส่วนเกินเหล่านี้เราสามารถที่จะขจัดออกไปได้ในภายหลัง ดังนั้น จึงได้เลือกวิธีการ Edge Detection โดยใช้ threshold ที่ปรับค่าได้ เป็นวิธีหลักในการตรวจหาขอบของภาพวัตถุ ซึ่งจะใช้สำหรับทดลองในขั้นต่อไป

จากข้อมูลภาพที่ผ่านการตรวจหาขอบของภาพวัตถุมาแล้ว ดังรูปที่ 7.5 จะนำมาลดความหนาของขอบภาพลง (Thinning) โดยผ่านกระบวนการทั้ง 4 ขั้นตอน คือ fill, delete, labelและ strip ซึ่งได้แสดงผลเพื่อเปรียบเทียบภาพก่อนทำการ Thinning และหลังผ่านกระบวนการ Thinning ครั้งแรก ดังรูปที่ 7.6 และ 7.7 ตามลำดับ

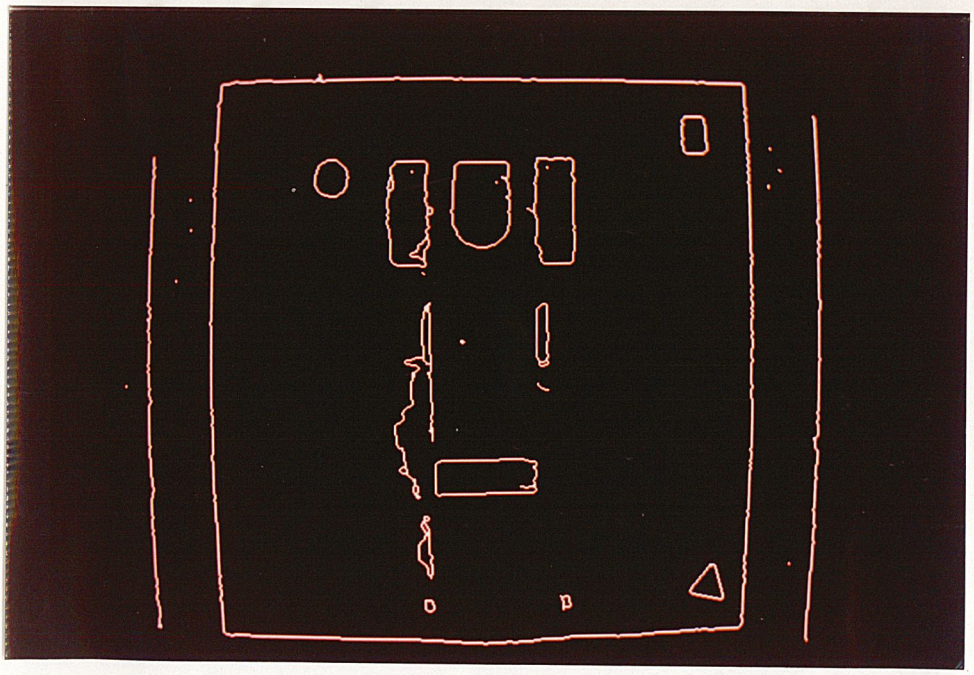


รูปที่ 7.6 แสดงภาพก่อนทำการ Thinning



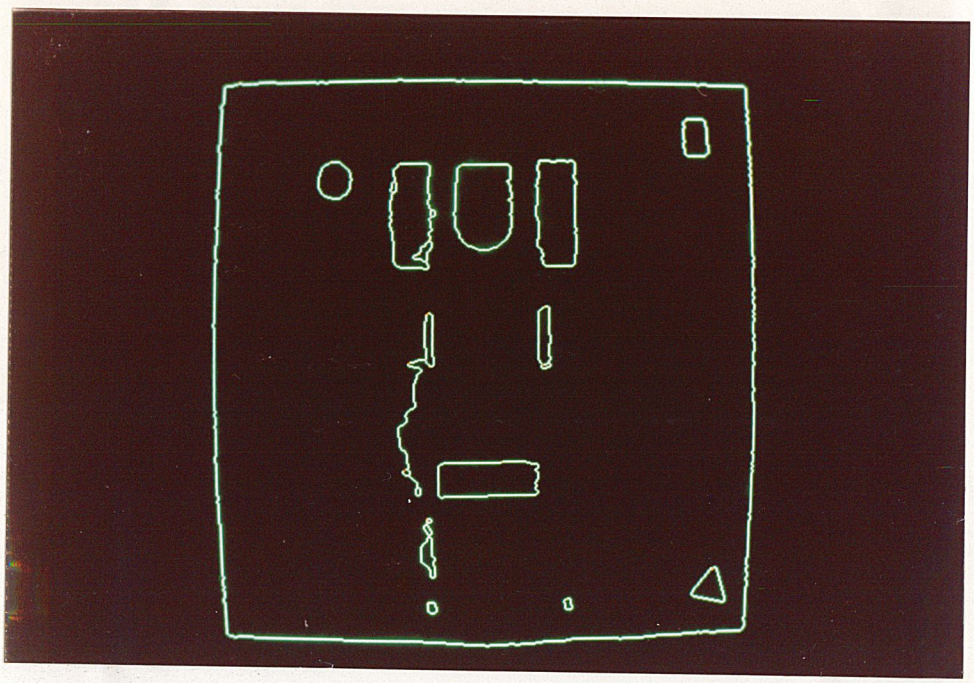
รูปที่ 7.7 แสดงภาพที่ผ่านกระบวนการ Thinning ครั้งที่ 1

แต่ขอบของภาพวัตถุในรูปที่ 7.7 ยังมีความหนามากกว่า 1 จุดภาพอยู่ จึงต้องผ่านกระบวนการ Thinning อีกครั้ง ได้ภาพที่มีความหนาของขอบภาพวัตถุลดลง ดังแสดงในรูปที่ 7.8



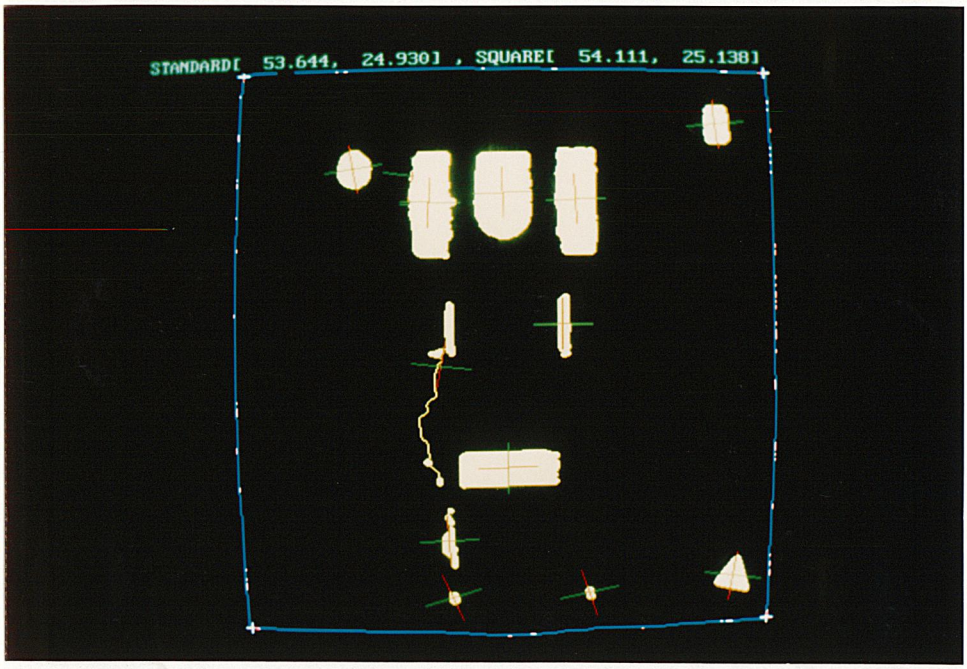
รูปที่ 7.8 แสดงภาพที่ผ่านกระบวนการ Thinning ครั้งที่ 2

ซึ่งภาพที่ผ่านกระบวนการ Thinning ครั้งที่ 2 ก็ยังมีขอบภาพวัตถุที่หนาอยู่ จึงต้องทำการ Thinning ซ้ำไปเรื่อยๆจนกระทั่งได้ขอบภาพที่มีความหนาเพียง 1 จุดภาพ ดังภาพที่ผ่านกระบวนการ Thinning ครั้งสุดท้าย ซึ่งแสดงไว้ในรูปที่ 7.9



รูปที่ 7.9 แสดงภาพที่ผ่านกระบวนการ Thinning จนเสร็จสมบูรณ์

จากภาพที่มีขอบภาพหนาเพียง 1 จุดภาพนี้ ได้นำข้อมูลของขอบภาพของวัตถุแต่ละชิ้น และ แขนกล รวมทั้งระนาบสี่เหลี่ยม มาเข้ารหัสลูกโซ่แบบ 8 ทิศทาง ซึ่งรหัสลูกโซ่ของระนาบสี่เหลี่ยมที่ได้ จะต้องนำไปหามุมทั้งสี่ โดยผ่าน 4 ขั้นตอน คือ การปรับรหัสลูกโซ่ให้ราบเรียบ, การหาจุดหักมุม, การกำจัดจุดหักมุมที่เป็นส่วนเกิน และ การรวมจุดหักมุมที่อยู่ใกล้กันให้เป็นจุดเดียว เพื่อให้ได้มุมของระนาบเพียง 4 มุมที่ต้องการเท่านั้น ส่วนรหัสลูกโซ่ของวัตถุแต่ละชิ้น และ แขนกล ซึ่งถูกวางอยู่บนระนาบสี่เหลี่ยม จะนำไปใช้กำหนดของเขตของวัตถุแต่ละชิ้นสำหรับการเปรียบเทียบแกนหลัก เพื่อให้สามารถแยกแยะรูปร่างของแขนกล และวัตถุแต่ละชิ้นได้ ดังแสดงในรูปที่ 7.10



รูปที่ 7.10 แสดงผลของการหาจุดมุมของระนาบ และ การหาแกนหลักของวัตถุ

ซึ่งจากรูปที่ 7.10 จุดมุมทั้งสี่ของระนาบสี่เหลี่ยม ได้แสดงเป็นกากบาทอยู่ที่มุมทั้งสี่ของระนาบ ซึ่งจุดมุมทั้งสี่นี้ จะนำไปหาขนาดของระนาบ เพื่อกำหนดอัตราส่วนการแปลงระยะจุดภาพให้เป็นระยะทางจริง ส่วนค่า eigenvalue ของวัตถุและแขนกลที่ได้จะนำไปเปรียบเทียบกับค่า eigenvalue มาตรฐาน ตามตารางดังต่อไปนี้

ชนิดของวัตถุ	Maximum Eigenvalue	Minimum Eigenvalue
วัตถุรูปวงกลม	41.185	37.814
วัตถุรูปสามเหลี่ยม	32.044	29.776
วัตถุรูปสี่เหลี่ยม	53.644	24.930
ส่วนหัวของแขนกล	246.990	124.905
ส่วนท้ายของแขนกล	407.846	48.127

โดยที่จากการทดลอง จะต้องมีการรับค่าว่า ต้องการให้แขนกลหยิบวัตถุรูปร่างอะไร จากนั้น คอมพิวเตอร์จะทำการคำนวณเพื่อหาค่า eigenvalue ของภาพวัตถุทั้งหมดที่อยู่ในระนาบ เพื่อที่จะเปรียบเทียบให้รู้ว่า ภาพวัตถุใดเป็นวัตถุที่ต้องการ และภาพใดเป็นส่วนหัวและส่วนท้ายของแขนกล เพื่อที่จะสามารถกำหนดจุดหมุนและทิศทางของแขนกลได้ จากนั้นจึงจะสามารถหาตำแหน่ง

งและทิศทางของวัตถุที่ต้องการอ้างอิงกับแขนกลได้ เพื่อทำให้สามารถระบุคำสั่งให้แขนกลเคลื่อนไหวไปหยิบวัตถุได้อย่างถูกต้อง

ผลจากการทดลอง สังเกตพบว่า ภาพที่รับเข้ามาจากกล้อง เมื่อทำการแปลงเป็น digital image แล้ว จะมีการขยายขนาดของภาพเพิ่มขึ้นตามแนวดิ่ง ทำให้ภาพของระนาบ ซึ่งเป็นรูปสี่เหลี่ยมจัตุรัส เพี้ยนไปเป็นรูปสี่เหลี่ยมผืนผ้า ที่มีอัตราส่วนของขนาดของภาพตามแนวนอน กับแนวดิ่ง เท่ากับ 0.82 ดังนั้น จึงจำเป็นจะต้องนำค่า 0.82 นี้มาคูณกับค่าระยะของภาพตามแนวดิ่ง เพื่อชดเชยความเพี้ยนที่เกิดขึ้น

นอกจากนี้ ในการเปรียบเทียบแกนหลักเพื่อแยกแยะรูปร่างของวัตถุนั้น เนื่องจากค่า eigenvalue เป็นค่าที่แสดงถึงความแปรปรวนตามแนวแกนของวัตถุ ซึ่งคำนวณมาจากค่ากำลังสองของระยะจากจุดภาพทุกๆจุดที่เป็นเนื้อของวัตถุไปยังจุด mean ของวัตถุนั้น จึงทำให้ค่า eigenvalue มีมิติเป็นค่ากำลังสองของระยะ ซึ่งในการเปรียบเทียบแกนหลัก ควรจะเปรียบเทียบกันด้วยค่าของระยะ มากกว่า จะเปรียบเทียบกันด้วย ค่ากำลังสองของระยะ ดังนั้น ในการทดลอง จึงได้ทำการเปรียบเทียบแกนหลัก โดยเปรียบเทียบด้วยค่ารากที่สองของ eigenvalue แทนการเปรียบเทียบด้วยค่า eigenvalue ที่คำนวณได้ และได้ทำการเผื่อค่าความผิดพลาดไว้ 5%

7.2 ข้อเสนอแนะ

เนื่องจากในการทดลอง ได้ทำการประมวลผลในลักษณะของซอฟท์แวร์ ซึ่งจะมีข้อจำกัดในเรื่องความเร็วในการคำนวณ ดังนั้นจึงควรจะได้มีการนำอัลกอลิทึมที่ได้จากโครงงานนี้ไปพัฒนาในเชิงฮาร์ดแวร์ ซึ่งจะทำให้แขนกลสามารถตอบสนองได้รวดเร็วขึ้น และควรศึกษาและประยุกต์ใช้เทคนิคที่จะใช้แยกแยะวัตถุในกรณีที่เกิดการซ้อนหรือทับกัน เพื่อให้สามารถแยกแยะรูปร่างของวัตถุได้มีประสิทธิภาพเพิ่มขึ้น นอกจากนี้ เนื่องจากการจดจำรูปร่างของวัตถุในโครงงานนี้ เป็นเพียงการมองในเชิง 2 มิติ ดังนั้นจึงควรจะได้มีการพัฒนาให้หุ่นยนต์สามารถมองและจดจำภาพในลักษณะ 3 มิติได้ต่อไป

กิตติกรรมประกาศ

(ACKNOWLEGEMENT)

ขอกราบขอบพระคุณท่านอาจารย์ที่ปรึกษา รองศาสตราจารย์ ดร. ฟูศักดิ์ ชีวสุวิทย์ เป็นอย่างสูง ที่ท่านได้ประสาทวิชาความรู้ และให้คำปรึกษาเสนอแนะแนวทางในการทดลองโครงงานและวิธีแก้ไขปัญหาต่างๆ ตลอดจนช่วยอำนวยความสะดวกในเรื่องของอุปกรณ์ อันเป็นผลทำให้โครงงานและปริญญานิพนธ์ฉบับนี้ สำเร็จลุล่วงไปได้ด้วยดี

ขอกราบขอบพระคุณท่านอาจารย์ ธนิตย์ ตรีสุวรรณวัฒน์ ที่กรุณาช่วยค้นหาคู่มือของแขนกลรุ่น RM-501 และ ท่านอาจารย์ ประภาษ อุคคกิมาพันธุ์ ที่กรุณาให้คำปรึกษาปัญหาทางด้าน mechanic อย่างดียิ่ง

ขอขอบพระคุณ คุณบัณฑิต สุมนวัฒนเดช ที่กรุณาให้ความช่วยเหลือ และแก้ไขปัญหาต่างๆที่เกิดขึ้นกับ การ์ด digitizer ซึ่งนับเป็นส่วนที่สำคัญอย่างยิ่งของโครงงานนี้ และต้องขอขอบพระคุณ คุณสักรียา ชิตวงค์ และ คุณอาโมทย์ สมบูรณ์แก้ว ที่คอยให้ข้อมูล ตลอดจนคำแนะนำต่างๆที่เป็นประโยชน์อย่างยิ่ง ซึ่งทำให้การทดลองของโครงงานนี้ ผ่านไปได้ด้วยดี

ท้ายที่สุดนี้ ขอขอบคุณเพื่อนๆทุกคนที่เป็นห่วงเป็นใย และให้กำลังใจต่อผู้ทำโครงงานนี้ตลอดมา

เอกสารอ้างอิง

(REFERENCES)

[1] K.S. Fu, R.C.Gonzalez, C.S.G. Lee, "ROBOTICS:Control, Sensing and Intelligence", McGraw-Hill Book Company, Singapore, 1987

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

[3] Yat Keung Chu, Ching Y. Suen, "An alternate smoothing and stripping algorithm for thinning digital binary patterns", Signal Processing, Vol.11, No.3, October 1986, pp.207-222

[4] ชาญชัย พิสิทธิวิทยานนท์, "การจดจำรูปร่างวัตถุ (Object recognition)", วิทยานิพนธ์ปริญญาวิศวกรรมศาสตร์มหาบัณฑิต สาขาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง, ปีการศึกษา 2531

[5] อาโมทย์ สมบูรณ์แก้ว, วิทยา ทิพย์สุวรรณพร, ฟูศักดิ์ ชีวสุวิทย์, "การตรวจสอบหาชนิดวัตถุของหุ่นยนต์โดยอาศัยการเปรียบเทียบแกนหลัก (Object identification of robot by comparing its principle axises)",การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 14,มหาวิทยาลัยสงขลานครินทร์,หน้า 4-73 ถึง 4-76,พฤศจิกายน 2534

[6] บัณฑิต สุมนวัฒนเดช, คเชนทร์ แจ่มกมล,"แผงวงจรควบคุมการเก็บภาพขนาด 512x512 จุดภาพ โดยใช้ตัวแปลงสัญญาณอนาล็อกเป็นดิจิตอล 2 ชุดร่วมกันทำงาน (512x512 pixels image interface card by using 2 A/D converters)",การประชุมใหญ่ทางวิชาการประจำปี 2536 วิศวกรรมสถานแห่งประเทศไทย ในพระบรมราชูปถัมภ์, หน้า 630 ถึง 641, พฤศจิกายน 2536

ภาคผนวก

```
#include <stdio.h>
#include <graphics.h>
#include <dos.h>
#include <conio.h>
#include <alloc.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>

#define B_W   0
#define COLOR 1
#define port 0x300
#define SIZE 512
#define PI 3.141592
#define POINT 20
#define DEGREE(x) 180*x/PI
#define CosLaw(a,b,c) acos((a*a+b*b-c*c)/(2*a*b))

int  load();
void Graphic();
int  Digitize();
void Input_image();
void smooth();
void edge();
void threshold();
unsigned char pix();
void putpix();
int  thin();
void position();
void cursor();
void save();
void display();
void fill();
void delete();
void label();
int  strip();
void trace();
void nextscan();
int  scan();
int  chkend();
```

```
int  chkbreak();
int  boolean();
int  chain();
void reduce();
int  sym();
void dispchain();
char encode();
void breakpoint();
int  neighbor();
void freechain();
void over_angle();
void combine();
void eigen();
int  is_fig();
void convert();
void initRobot();
void resetRobot();
void arm();

typedef struct node *ptype;
struct node { int x,y;
      char c;
      ptype  nxt; } ;
ptype head;
struct { int x,y; } plane[POINT],front,rear,center,obj;
char fig;
unsigned char huge image[SIZE][SIZE];
float H= 114;
float wide;
unsigned char WANT;

void main()
{ int i,j;
  unsigned char m,n,error=0;

  do
  {
   initRobot();

   if (Digitize())
```

```
   { Graphic(COLOR);
     for (n=0; thin(n)==1; n++);
     thin();

     for(n=0; n<POINT; n++)
     {  plane[n].x = -1;
plane[n].y = -1;
     }
     gotoxy(1,1);
     for (j=1, fig=0; j<SIZE-1 && error==0; j++)
       for (i=1, head=NULL;i<SIZE-1 && fig<40 && error==0;i++)
 if (pix(i,j)==5)
 { if (fig==0)
   { if (chain(i,j,-1)==1)
     { reduce();
       dispchain();
       breakpoint();
       over_angle();
       combine();
       for (m=0,n=0; m<4; n++)
      if (plane[n].x>0 && plane[n].y>0)
      { plane[m] = plane[n];
        cursor(plane[m].x,plane[m].y);
        m++;
      }
       for (n=0,wide=1.0; n<4; n++)
       wide *= pow( pow((plane[n+1].x)-(plane[n].x),2.0)
          + pow((plane[n+1].y)*0.82-(plane[n].y)*0.82 ,2.0) ,0.125);
    } else
       { error = 1;
       gotoxy(25,15);
       printf("!! CHAIN ERROR !!");
       break;
       }
    fig++;
  } else
    if ( chain(i,j,-1)==1)
    { eigen();
      fig++;
    }
```

```
    freechain(head);
  }

      if (error==0)
      { convert(&front.x,&front.y,54.3);
        convert(&rear.x ,&rear.y ,50.0);
        convert(&obj.x  ,&obj.y  ,1);
        center.x = rear.x + (int)(0.55*((front.x)-(rear.x)));
        center.y = rear.y + (int)(0.55*((front.y)-(rear.y)));
        cursor(center.x,center.y);
        setcolor(1);
        line(center.x,center.y,front.x,front.y);
        setcolor(8);
        line(center.x,center.y,obj.x,obj.y);
        arm();
      }
      position(&i,&j);
      closegraph();
    }
    resetRobot();
    clrscr();
    printf("\nPress any key to continue,ESC. to end program");
  }
  while (getch()!=27);
  clrscr();
  return;
}

void Graphic(int mode)
{ int gdriver = VGA, gmode = VGAHI, errorcode;
  struct palettetype pal;
  char i;

  initgraph(&gdriver, &gmode, "\\cpp\\bgi");
  errorcode = graphresult();
  if (errorcode != grOk)
  { printf("Graphics error: %s\n",grapherrormsg(errorcode));
    printf("Press any key to halt");
    getch();
    exit(1);
```

```
  }
  if (mode == B_W)
  { getpalette(&pal);
    for (i=0; i<16; i++)
      setrgbpalette(pal.colors[i], i*4, i*4, i*4);
  }
  return;
}

int Digitize()
{ char ans;

  clrscr();
  outportb(port,0x00);
  printf("\nPress any key to get PICTURE");
  getch();
  outportb(port,0x02);
  printf("\nSave Y/N ?");
  switch(getche())
  {  case('y'):
     case('Y'):ans=1; break;
     default  :ans=0; break;
  }

  if( ans==1 )
  {  Graphic(B_W);
     Input_image();
     smooth();
     edge();
     threshold();
     if ( (WANT=getch()) ==27) ans = 0;
     else WANT -= 0x30;
     closegraph();
  }
  return(ans);
}

int load()
{  FILE *fp;
   int i,j;
```

```
    unsigned char *ptr;
    fpos_t pos;

    if ( (fp = fopen("\\test.img","r")) == NULL )
      return(0);
    rewind(fp);
    for (j=0, pos=0L; j<512; j++, pos+=512)
    { fseek(fp,pos,SEEK_SET);
      fgetc(fp);
      ptr = fp->buffer;
      for (i=0; i<512; i++)
      { putpixel(i,j,ptr[i]/16);
        putpix(i,j,ptr[i]);
      }
    }
    fclose(fp);
    return(1);
}

void save()
{ FILE *fp2;
  int i,j;

  if ( (fp2 = fopen("\\proj\\test.img","wt")) != NULL )
  {  rewind(fp2);
     for (j=0; j<SIZE; j++)
       for (i=0; i<SIZE; i++)
 fputc((pix(i,j)>1)?255:0,fp2);
     fclose(fp2);
  }
  return;
}

void Input_image()
{  unsigned char far *eptr;
   unsigned char far *optr;
   int n;
   unsigned char i,c;

   eptr = MK_FP(0xd000,0000);
```

```
    optr = MK_FP(0xe000,0000);
    for(i=0; i<4; i++)
    {  outportb(port,0x07);
       for (c=0; c<128; c+=2)
        for (n=0; n<512; )
        {
 image[i*128+c][n] = *(eptr+(i*16384)+c*128+n/2);
 putpixel(n,(i*128+c),image[i*128+c][n]/16);
 n++;
 image[i*128+c][n] = *(optr+(i*16384)+c*128+n/2);
 putpixel(n,(i*128+c),image[i*128+c][n]/16);
 n++;
        }
       outportb(port,0x03);
       for (c=0; c<128; c+=2)
        for (n=0; n<512; )
        {
 image[i*128+c+1][n] = *(eptr+(i*16384)+c*128+n/2);
 putpixel(n,i*128+c+1,image[i*128+c+1][n]/16);
 n++;
 image[i*128+c+1][n] = *(optr+(i*16384)+c*128+n/2);
 putpixel(n,i*128+c+1,image[i*128+c+1][n]/16);
 n++;
        }
    }
    outportb(port,0x02);
    outportb(port,0x00);
    return;
}
unsigned char pix(int i, int j)
{
   return(image[j][i]);
}

void putpix(int i,int j,unsigned char ch)
{
  image[j][i] = ch;
  return;
}
```

```
void threshold()
{  unsigned char imgtem[SIZE];
   int i,j;
   int avg;
   unsigned char next;

   for (j=0; j<SIZE; j++)
     for (i=0, next=pix(0,j); i<SIZE; i++ )
     {  if (i!=0 && j!=0 && i!=SIZE-1 && j!=SIZE-1)
{ avg = ( imgtem[i-1]  + imgtem[i]  + imgtem[i+1]
      + next                        + pix(i+1,j)
      + pix(i-1,j+1) + pix(i,j+1) + pix(i+1,j+1) );

  if (pix(i,j)*8-avg > 8) avg = 15;
  else  avg = 0;
}
else  avg = 0;
putpixel(i,j,avg);
imgtem[i] = next;
next = pix(i,j);
putpix(i,j,avg);
     }
   return;
}

void smooth()
{  unsigned char imgtem[SIZE];
   int i,j;
   unsigned char next,avg;
   unsigned char sort[9],m,n;

   for (j=0; j<SIZE; j++)
   { for (i=0, next=pix(0,j); i<SIZE; i++ )
     {  if (i!=0 && j!=0 && i!=SIZE-1 && j!=SIZE-1)
{ sort[0] = imgtem[i-1];
  sort[1] = imgtem[i];
  sort[2] = imgtem[i+1];
  sort[3] = next;
  sort[4] = pix(i,j);
  sort[5] = pix(i+1,j);
```

```
   sort[6] = pix(i-1,j+1);
   sort[7] = pix(i,j+1);
   sort[8] = pix(i+1,j+1);
   for (m=0; m<5; m++)
    for (n=m+1; n<9; n++)
     if (sort[n] < sort[m])
     { avg = sort[m];
       sort[m] = sort[n];
       sort[n] = avg;
     }
   avg = sort[4];
 }
 else  avg = pix(i,j);
 putpixel(i,j,avg/16);
 imgtem[i] = next;
 next = pix(i,j);
 putpix(i,j,avg);
      }
      putpixel(520,j,15);
    }
    return;
 }

 void edge()
 {  unsigned char imgtem[SIZE],next;
    int i,j,GX,GY,GXY;

    for (j=0; j<SIZE; j++)
    { for (i=0, next=pix(0,j); i<SIZE; i++ )
      {  if (i!=0 && j!=0 && i!=SIZE-1 && j!=SIZE-1)
 {  GX = -imgtem[i-1]  + imgtem[i+1]
       -2*next        + 2*pix(i+1,j)
       -pix(i-1,j+1) + pix(i+1,j+1) ;
    GY = -imgtem[i-1] - 2*imgtem[i]  - imgtem[i+1]
       + pix(i-1,j+1) + 2*pix(i,j+1) + pix(i+1,j+1) ;
    GXY = (abs(GX) + abs(GY))/8 ;
 }
 else  GXY = 0;
 /*putpixel(i,j, GXY/16*3 );*/
 imgtem[i] = next;
```

```
next = pix(i,j);
putpix(i,j,GXY);
     }
     putpixel(520,j,0);
   }
   return;
}

void position(int *pi,int *pj)
{  unsigned char si[3],sj[3],c;
   int i,j;

   i = SIZE/2;
   j = SIZE/2;
   outtextxy(550,450,"I :");
   outtextxy(550,470,"J :");
   setcolor(10);
   sprintf(si,"%d",i);
   sprintf(sj,"%d",j);
   outtextxy(600,450,si);
   outtextxy(600,470,sj);
   cursor(i,j);
   while ( (c=getch()) != 13)
   {  cursor(i,j);
      setcolor(BLACK);
      outtextxy(600,450,si);
      outtextxy(600,470,sj);
      if (c==0)
      { c = getch();
if (c==77 && i<SIZE-1) i++;
if (c==75 && i>0)   i--;
if (c==72 && j>0)   j--;
if (c==80 && j<SIZE-1) j++;
      }
      setcolor(10);
      sprintf(si,"%d",i);
      sprintf(sj,"%d",j);
      outtextxy(600,450,si);
      outtextxy(600,470,sj);
      cursor(i,j);
```

```
    }
    *pi = i;
    *pj = j;
    return;
}

void cursor(int i,int j)
{ char n;
  unsigned char hor[9],ver[9];

  for (n=-4; n<5; n++)
  {  hor[n+4] = getpixel(i+n,j);
     putpixel(i+n,j,15-hor[n+4] );
     ver[n+4] = getpixel(i,j+n);
     putpixel(i,j+n,15-ver[n+4] );
  }
  return;
}

void display()
{ int i,j;
    for (j=0; j<SIZE; j++)
      for (i=0; i<SIZE; i++)
putpixel(i,j,pix(i,j)*3);
}

int thin(unsigned char iter)
{
   fill(iter);
   delete();
   label();
   return( strip(iter) );
}

void fill(t)
{ unsigned char imgtem[SIZE];
  int i,j;
  unsigned char next,m,n;

  for (j=0; j<SIZE; j++)
```

```
  { for (i=0, next=pix(0,j); i<SIZE; i++ )
    { m = 0;
      n = 0;
      if (i>0 && j>0 && i<SIZE-1 && j<SIZE-1)
if ( pix(i,j)<=1 )
{  m += (imgtem[i-1] <=1) ?0:1;
   n += (imgtem[i]   <=1) ?0:1;
   m += (imgtem[i+1] <=1) ?0:1;
   n += (next        <=1) ?0:1;
   n += (pix(i+1,j)  <=1) ?0:1;
   m += (pix(i-1,j+1)<=1) ?0:1;
   n += (pix(i,j+1)  <=1) ?0:1;
   m += (pix(i+1,j+1)<=1) ?0:1;
}
      imgtem[i-1] = next;
      next = pix(i,j);
      if (n>2 || (m>2 && t==0)  || next>1)
putpix(i,j,4);
      else putpix(i,j,0);
      if (pix(i,j)>0) putpixel(i,j,pix(i,j)*3);
    }
    imgtem[SIZE-1] = pix(SIZE-1,j);
  }
  return;
}

void delete()
{ int i,j,a,b;
  char m,n;

  for (j=0; j<SIZE; j++)
    for (i=0; i<SIZE; i++)
    { a = i;
      b = j;
      m = -1;
      n = 0;
      while ( a>=0 && b>=0 && a<SIZE && b<SIZE )
       if ( pix(a,b)>1 && boolean(a,b)==0 )
       {
 scan(a,b,&m,&n,-1);
```

```
  putpix(a,b,0);
  putpixel(a,b,0);
  a += m;
  b += n;
        } else  break;
     }
   return;
 }

 void label()
 { unsigned char imgtem[SIZE];
   int i,j;
   unsigned char next,tem,c,t;

   for (j=0; j<SIZE; j++)
   { for (i=0, next=1; i<SIZE; i++ )
     { c = 0;
       t = 0;
       tem = pix(i,j);
       if (i>0 && j>0 && i<SIZE-1 && j<SIZE-1)
       { if (pix(i,j)>0)
{
  t += (imgtem[i] >0) ?1:0;
  t += (next      >0) ?1:0;
  t += (pix(i+1,j)>0) ?1:0;
  t += (pix(i,j+1)>0) ?1:0;

  if ( imgtem[i] && next && pix(i+1,j) && pix(i,j+1) )
  { c += (imgtem[i-1] >0) ?1:0;
    c += (imgtem[i+1] >0) ?1:0;
    c += (pix(i-1,j+1)>0) ?1:0;
    c += (pix(i+1,j+1)>0) ?1:0;
  }
}
if (c==3)
  putpix(i,j,3);
if (t<4 && pix(i,j)>0)
  putpix(i,j,2);
      }
      if (tem>0) putpixel(i,j,pix(i,j)*3);
```

```
      if (i>0) imgtem[i-1] = next;
      next = tem;
    }
    imgtem[SIZE-1] = pix(SIZE-1,j);
  }
  return;
}

int strip(unsigned char iter)
{ int i,j;
  unsigned char tem;
  long remain=0;

  for (j=0; j<SIZE; j++)
  { for (i=0; i<SIZE; i++)
    { tem = pix(i,j);
      if ( tem==2 || tem==3)    /* meet trace- or corner- point */
      { if (iter%2 == 1)        /* odd iteration */
  trace(i,j,+1);          /* out clock */
else                      /* even iteration */
  trace(i,j,-1);          /* out counterclock */
putpixel(i,j,pix(i,j)*2);
      }
      if ( tem==1 )
putpixel(i,j,0);
      if ( tem==4 )
remain ++;
    }
  }
  if (remain>2) return(1);
  else return(0);
}

void trace(int i,int j,char dir)
{ int a,b;
  char m,n;

  a = i;
  b = j;
  m = dir;
```

```
    n = 0;
    if ( scan(a,b,&m,&n,dir)==8 )
    { putpixel(a,b,0);
      putpix(a,b,0);
      return;
    }
    if ( pix(a+m,b+n)!=2 && pix(a+m,b+n)!=3)
    { dir = -dir;
      m = 0;
      n = 1;
      scan(a,b,&m,&n,dir);
    }
    do
    { if (pix(a+m,b+n)!=2 && pix(a+m,b+n)!=3)
        break;
      a += m;
      b += n;
      if (a<0 || b<0 || a>=SIZE || b>=SIZE)  break;
      if ( (chkend(a,b)==0) && (chkbreak(a,b)==0) )
      {  putpix(a,b,1);
         putpixel(a,b,0);
      }
      else { putpix(a,b,5);
     putpixel(a,b,pix(a,b)*2); }
      nextscan(&m,&n,dir);
      scan(a,b,&m,&n,dir);
    } while( (a+m)!=i || (b+n)!=j);
   return;
}

void nextscan(char *x,char *y,char clock)
{  switch( (*x)*(*y)*clock )
    { case  0 : if (*x==0)
          { *x = clock*(*y);
          *y = -(*y);
          }
          else
          { *y = -clock*(*x);
          *x = -(*x);
          }
```

```
      break;
    case  1 : *y = -clock*(*x);
      *x = 0;
      break;
    case -1 : *x = clock*(*y);
      *y = 0;
      break;
  }
  return;
}

int scan(int a,int b,char *x,char *y,char clock)
{ int t;

  for (t=0; t<8; t++)
  { if ( pix(a+(*x),b+(*y)) >1 )
      return(t);
    switch ((*x)*(*y)*clock)
    { case  0 : if (*x==0) *x = clock*((*x)-(*y));
      else       *y = clock*((*x)-(*y));
      break;
      case  1 : *x = 0;    break;
      case -1 : *y = 0;    break;
    }
  }
  return(t);
}

int chkend(int i,int j)
{ unsigned char n=0;

  if ( pix(i,j) >0 )
  {  n += (pix(i-1,j-1)>1) ?1:0;
     n += (pix(i,j-1)  >1) ?1:0;
     n += (pix(i+1,j-1)>1) ?1:0;
     n += (pix(i-1,j)  >1) ?1:0;
     n += (pix(i+1,j)  >1) ?1:0;
     n += (pix(i-1,j+1)>1) ?1:0;
     n += (pix(i,j+1)  >1) ?1:0;
     n += (pix(i+1,j+1)>1) ?1:0;
```

```
    if (n==1) return(1);
  }
  return(0);
}

int chkbreak(int i,int j)
{
  if (pix(i,j)<=1)  return(0);

  if ( pix(i-1,j)<=1 && pix(i+1,j)<=1 )
  {  if ( ( (pix(i-1,j-1)>1) || (pix(i,j-1)>1) || (pix(i+1,j-1)>1) ) &&
  ( (pix(i-1,j+1)>1) || (pix(i,j+1)>1) || (pix(i+1,j+1)>1) ) )
       return(1);
  }
  if ( pix(i,j-1)<=1 && pix(i,j+1)<=1 )
  {  if ( ( (pix(i-1,j-1)>1) || (pix(i-1,j)>1) || (pix(i-1,j+1)>1) ) &&
  ( (pix(i+1,j-1)>1) || (pix(i+1,j)>1) || (pix(i+1,j+1)>1) ) )
       return(1);
  }
  if ( pix(i-1,j)<=1 && pix(i,j+1)<=1 && pix(i-1,j+1)>1 )
  {  if ( (pix(i-1,j-1)>1) || (pix(i,j-1)  >1) || (pix(i+1,j-1)>1) ||
  (pix(i+1,j)  >1) || (pix(i+1,j+1)>1) )
       return(1);
  }
  if ( pix(i-1,j)<=1 && pix(i-1,j-1)>1 && pix(i,j-1)<=1 )
  {  if ( (pix(i+1,j-1) >1)|| (pix(i+1,j)>1) ||
  (pix(i-1,j+1)>1) || (pix(i,j+1)>1) || (pix(i+1,j+1)>1) )
       return(1);
  }
  if ( pix(i,j-1)<=1 && pix(i+1,j)<=1 && pix(i+1,j-1)>1 )
  {  if ( (pix(i-1,j-1)>1) || (pix(i-1,j)>1) ||
  (pix(i-1,j+1)>1) || (pix(i,j+1)>1) || (pix(i+1,j+1)>1) )
       return(1);
  }
  if ( pix(i,j+1)<=1 && pix(i+1,j)<=1 && pix(i+1,j+1)>1 )
  {  if ( (pix(i-1,j-1) >1) || (pix(i,j-1)  >1) || (pix(i+1,j-1)>1) ||
  (pix(i-1,j)   >1) || (pix(i-1,j+1)>1) )
       return(1);
  }
  return(0);
```

```
}

int boolean(int i,int j)
{  char t1,t2;
   t1  = (pix(i-1,j-1)>1) | (pix(i,j-1)  >1) | (pix(i-1,j)  >1);
   t1 &= (pix(i+1,j)  >1) | (pix(i,j+1)  >1) | (pix(i+1,j+1)>1);
   t2  = (pix(i,j-1)  >1) | (pix(i+1,j-1)>1) | (pix(i+1,j)  >1);
   t2 &= (pix(i-1,j)  >1) | (pix(i-1,j+1)>1) | (pix(i,j+1)  >1);
   return(t1|t2);
}

int chain(int i,int j,char dir)
{ int a,b;
  char m,n;
  ptype ptr;

  a = i;
  b = j;
  m = dir;
  n = 0;

  if ( scan(a,b,&m,&n,dir) != 8 ) /* meet trace */
  { if ( (head = (ptype)malloc(7)) == NULL )  return(0);
    ptr = head;
    do
    { if ( (ptr->nxt = (ptype)malloc(7)) == NULL )
break;
      ptr->x = a;
      ptr->y = b;
      ptr->c = encode(m,n);
      putpix(a,b,3);
      putpixel(a,b,12);
      a += m;
      b += n;
ptr = ptr->nxt;
      nextscan(&m,&n,dir);
      if ( scan(a,b,&m,&n,dir) ==7)  break;
    } while ( (a+m)!=i || (b+n)!=j );

    ptr->nxt = head;
```

```
      ptr->x = a;
      ptr->y = b;
      ptr->c = encode(m,n);
      putpix(a,b,3);
      putpixel(a,b,12);
      if ((a+m)==i && (b+n)==j)
        return(1);
    }
    return(0);
}

void reduce()
{ unsigned char m,n;
  int diff[9],tem;
  ptype p,q,r;

  p = head;
  do
  { q = p->nxt;
    if (p->c != q->c)
    { for (n=0; n<9; n++)
      { diff[n] = (p->c) - (q->c);
if (p->c==0 && q->c==7)  diff[n] += 8;
if (p->c==7 && q->c==0)  diff[n] -= 8;
q = q->nxt;
      }
      q = p->nxt;
      for (n=1,tem=0; n<=9; n++)
      { q = q->nxt;
tem += diff[n-1];
if ( tem==0 && sym(diff,n)==1)   break;
      }
      if (n <= 9)
      { r = p->nxt;
r->c = p->c;
for (m=2; m<=n; m++)
{ r = r->nxt;
  r->x = p->x + m*((q->x)-(p->x))/(n+1);
  r->y = p->y + m*((q->y)-(p->y))/(n+1);
  r->c = p->c;
```

```
}
      }
    }
    p = p->nxt;
  } while (p != head);
  return;
}

int sym(int *A,unsigned char n)
{ unsigned char t;

  for (t=1; t<=n/2; t++)
    if ( (A[t-1]+A[n-t]) != 0)  return(0);
  return(1);
}

void  dispchain()
{ ptype ptr;

  ptr = head;
  do
  {  putpixel(ptr->x,ptr->y,3);
     ptr = ptr->nxt;
  } while (ptr != head);
  return;
  }

char encode(char m,char n)
{ switch(m)
  { case -1 : switch(n)
      { case -1 : return(3);
      case  0 : return(4);
      case  1 : return(5);
      }
    case  0 : switch(n)
      { case -1 : return(2);
      case  1 : return(6);
      }
    case  1 : switch(n)
      { case -1 : return(1);
```

```
      case  0 : return(0);
      case  1 : return(7);
      }
  }
  return(0);
}

void breakpoint()
{ int count,len,tem;
  unsigned char seterr,error,bp=0;
  char major,minor;
  ptype p,q;

   p = head;
   do
   { major = p->c;
     p = p->nxt;
     for (count=1; p->c==major && p!=head; count++)
p = p->nxt;
     if ( (neighbor(p->c,major)==1) && (p!=head) )
     { minor = p->c;
       for (len=0; p->c==minor && p!=head; len++)
 p = p->nxt;
       if (len > count)
       { tem   = count;
 count = len;
 len   = tem;
 tem   = major;
 major = minor;
 minor = tem;
       }
       if (count>=5)
 seterr = 2;
       else seterr = 3;
       error = 0;
       while (p!=head)
       { if (p->c == major)
   error = 0;
 else if (p->c == minor)
     { if (error==0)  q = p;
```

```
        error++;
        if (error>seterr)
        {  p = q;  break; }
        }
        else break;
  p = p->nxt;
         }
       }
       if (p!=head)
       { plane[bp].x = p->x;
         plane[bp].y = p->y;
         bp++;
       }
     } while (p != head);
     return;
}

int neighbor(char c1,char c0)
{ char tem;

  tem = abs(c1-c0);
  if ( ((c0==0) && (c1==7)) || ((c0==7) && (c1==0)) )
    return(8-tem);
  return(tem);
}

void freechain(ptype f)
{ ptype q;

  do
  { q = f->nxt;
    free(f);
    f = q;
  } while (f!=head && f!=NULL);
  return;
}

void over_angle()
{ unsigned char cnt,n;
  int px,py,px1,py1;
```

```
float A,B,C,theta;

for (n=0,cnt=0; n<POINT; n++)
{ if (plane[n].x!=-1)
    cnt++;
}
px1 = plane[1].x;
py1 = plane[1].y;
px  = plane[0].x;
py  = plane[0].y;
n = 0;
for (theta=0; n<cnt-1; n++,theta=0)
{ A = pow( (plane[n+1].x)      -      px      ,2.0 )
    + pow( (plane[n+1].y)      -      py      ,2.0 );
  B = pow( (plane[(n+2)%cnt].x)-(plane[n+1].x),2.0 )
    + pow( (plane[(n+2)%cnt].y)-(plane[n+1].y),2.0 );
  C = pow( (plane[(n+2)%cnt].x)-      px      ,2.0 )
    + pow( (plane[(n+2)%cnt].y)-      py      ,2.0 );

  theta = acos( (A+B-C) / (2*sqrt(A*B)) )*180/PI;
  px = plane[n+1].x;
  py = plane[n+1].y;
  if (theta >= 140)
  { plane[n+1].x = -1;
    plane[n+1].y = -1;
  }
}
A = pow( (plane[0].x) -      px     ,2.0 )
  + pow( (plane[0].y) -      py     ,2.0 );
B = pow(      px1     - (plane[0].x) ,2.0 )
  + pow(      py1     - (plane[0].y) ,2.0 );
C = pow(      px1     -      px     ,2.0 )
  + pow(      py1     -      py     ,2.0 );
theta = acos( (A+B-C) / (2*sqrt(A*B)) )*180/PI;
if (theta >= 140)
{ plane[0].x = -1;
  plane[0].y = -1;
}
return;
}
```

```
void combine()
{ unsigned char m,n,l;

    for (m=0; m<POINT; m++)
    { for (; plane[m].x==-1 && m<POINT; m++);
      if (m==POINT)  break;
      for (n=m+1; plane[n].x==-1; )
n = (n+1)%POINT;
      l = abs(plane[n].x - plane[m].x);
      if ( abs(plane[n].y-plane[m].y) > l )
l = abs(plane[n].y - plane[m].y);
      if ( l <= 6)
      { plane[m].x = (plane[n].x + plane[m].x)/2;
plane[m].y = (plane[n].y + plane[m].y)/2;
plane[n].x = -1;
plane[n].y = -1;
      }
    }
  return;
}

void eigen()
{
  struct{ float max,min;
  char name[10]; } Evalue[5] = { {246.990, 124.905, "FRONT"    } ,
       {407.846,  48.127, "REAR"     } ,
       { 41.185,  37.814, "CIRCLE"   } ,
       { 32.044,  29.776, "TRIANGLE" } ,
       { 53.644,  24.930, "RECTANGLE"} };
  float   cat,dog;
  double  a,b,c,d;
  unsigned int     N;
  float   meanx , meany;
  float   vxmax,vymax,vxmin,vymin;
  unsigned long  Exi,Eyi;
  float   eigenmax, eigenmin;
  int i,j;
  int xmin,ymin,xmax,ymax;
  ptype r;
```

```
  r = head;
  xmin = r->x;
  xmax = xmin;
  ymin = r->y;
  ymax = ymin;
  do
  { if (r->x < xmin)
      xmin = r->x;
    else if (r->x > xmax)
   xmax = r->x;
    if (r->y < ymin)
      ymin = r->y;
    else if (r->y > ymax)
   ymax = r->y;
    r = r->nxt;
  } while (r!=head);

  N = Exi = Eyi = 0;
  a = b = c = d = 0.0;

  for (j=ymin; j<=ymax; j++)
    for (i=xmin; i<=xmax; i++)
    { if (is_fig(i,j)==1)
      { N++;
Exi += i;
Eyi += j;
putpixel(i,j,3);
      }
    }
  meanx = Exi/(float)N;
  meany = Eyi/(float)N;

  for(j=ymin; j<=ymax; j++)
   for(i=xmin; i<=xmax; i++)
    {
      if(is_fig(i,j)==1)
      {
a  += (i-meanx)*(i-meanx);
b  += (i-meanx)*(j-meany)*0.82;
```

```
d  += (j-meany)*(j-meany)*0.82*0.82;
putpixel(i,j,14);
      }
    }
    a = a/N;
    b = b/N;
    c = b;
    d = d/N;

    cat  = (d+a)*(d+a);
    dog  =  4 * ((a*d)-(b*c));
    if(cat >= dog)
    {
eigenmax = ( (d + a) + (sqrt(cat - dog)) ) / 2;
eigenmin = ( (d + a) - (sqrt(cat - dog)) ) / 2;
    }

    if (a-eigenmax+c != 0)
    { cat   = -(b+d-eigenmax)/(a-eigenmax+c);
      vymax = 1/sqrt(1+cat*cat);
      vxmax = cat * vymax;
    } else
      { vxmax = 1;
if (b+d-eigenmax == 0)
  vymax = 1;
else vymax = 0;
      }
    setcolor(RED);
    line(meanx-20*vxmax,meany-20*vymax,meanx+20*vxmax,meany+20*vymax);

    if (a-eigenmin+c != 0)
    { dog   = -(b+d-eigenmin)/(a-eigenmin+c);
      vymin = 1/sqrt(1+dog*dog);
      vxmin = dog * vymin;
    } else
      { vxmin = 1;
if (b+d-eigenmin == 0)
  vymin = -1;
else vymin = 0;
      }
```

```
    setcolor(GREEN);
    line(meanx-20*vxmin,meany-20*vymin,meanx+20*vxmin,meany+20*vymin);

/*    printf("fig %d |%7.2f %7.2f |",fig,a,b);
    printf("      emax =%8.3f [%6.3f,%6.3f]",eigenmax,vxmax,vymax);
    if (vxmax != 0)
      printf("      theta = %6.2f\n",atan(vymax/vxmax)*180/PI);
    else printf("      theta =  90.00\n");
    printf( "       |%7.2f %7.2f |",c,d);
    printf("      emin =%8.3f [%6.3f,%6.3f]",eigenmin,vxmin,vymin);
    if (vxmin != 0)
      printf("      theta = %6.2f\n",atan(vymin/vxmin)*180/PI);
    else printf("      theta =  90.00\n");*/

    if ( fabs( sqrt(eigenmin)-sqrt(Evalue[0].min) ) < 0.05*sqrt(Evalue[0].min) )
    { front.x = meanx;
      front.y = meany;
    }
    if ( fabs( sqrt(eigenmax)-sqrt(Evalue[1].max) ) < 0.05*sqrt(Evalue[1].max) )
    { rear.x = meanx;
      rear.y = meany;
    }
    if ( ( fabs(sqrt(eigenmin)-sqrt(Evalue[WANT].min)) < 0.05*sqrt(Evalue[WANT].m
      && ( fabs(sqrt(eigenmax)-sqrt(Evalue[WANT].max)) < 0.05*sqrt(Evalue[WANT].m
    { obj.x = meanx;
      obj.y = meany;
      printf("STD[%8.3f,%8.3f] , OBJ[%8.3f,%8.3f]\n",Evalue[WANT].max,Evalue[WANT
    }
    return;
}

int is_fig(int i,int j)
{ ptype r,s;
  char mx=0,px=0,my=0,py=0;

  r = head;
  do
  { if (r->x==i && r->y==j)
      return(1);
    if (r->x==i && r->y!=j)
```

```
    { my |= (r->y<j)?1:0;
      py |= (py==0 && r->y>j)?1:0;
    }
    if (r->y==j && r->x!=i)
    { mx |= (r->x<i)?1:0;
      px |= (r->x>i)?1:0;
    }
    r = r->nxt;
  } while (r!=head);
  if (mx==1 && px==1 && my==1 && py==1)
    return(1);
  else return(0);
}

void convert(x,y,h)
int *x,*y;
float h;
{
   *x = (int)((*x-256)*(H-h)/H);
   *y = (int)((*y-256)*(H-h)/H*0.82);
   *x += 256;
   *y += 256;
   return;
}

void initRobot()
{  printf("\nWAIT FOR A MINUTE.");

   /* Set initial position of Robot Arm */
   fprintf(stdprn,"%s\n","NT");
   delay(100);
   fprintf(stdprn,"%s\n","MI -6000,0,0,0,0,0");
   delay(1000);
   fprintf(stdprn,"%s\n","GC");
   delay(100);
   fprintf(stdprn,"%s\n","HO");
   delay(100);
   return;
}
```

```
void resetRobot()
{  fprintf(stdprn,"%s\n","RS");
   delay(500);
   return;
}

      /* Function to move arm to pixel (i,j) */

void arm()
{  float A,B,C,bd,sh,el,wr,r,l,k;
   unsigned char cmd[40],
   bd_str[6], sh_str[6], el_str[6], wr_str[6];
   float x, y, x1, h=2.0,H=27;
   float Factor;

   /* Calculate radius and angle */
   Factor = 70.0/wide;
   printf(" wide = %7.3f,Factor = %7.3f\n",wide,Factor);
   A = pow( front.x - center.x ,2.0 )
     + pow( front.y - center.y ,2.0 );
   B = pow(   obj.x - center.x ,2.0 )
     + pow(   obj.y - center.y ,2.0 );
   C = pow( front.x -    obj.x ,2.0 )
     + pow( front.y -    obj.y ,2.0 );

   bd = acos( (A+B-C) / (2*sqrt(A*B)) );

   r = sqrt(B)*Factor;
   x = r*sin(bd);
   y = r*cos(bd);
   if (front.y == center.y)
   { if ( (front.x - center.x)*(obj.y - center.y)>0 )
     {  x = -x;
        bd = -bd;
     }
   }
   else
   { x1 = center.x + (obj.y-center.y)*(front.x-center.x)/(front.y-center.y);
     if ( (front.y - center.y)*(obj.x - x1) >0 )
     {  x = -x;
```

```
        bd = -bd;
      }
    }
    printf(" r = %6.2f,theta = %6.2f",r,bd);
    if (x>-11 && x<11 && y>-15 && y<8)
       printf("Sorry, my hand may attack my base.\n");
    else
    if ( r<=sqrt(38*38-(H-21.46-h)*(H-21.46-h)) )
    { bd = DEGREE(bd);
      if (bd>=-150 && bd<=150 )
      {  l = sqrt( (H-h)*(H-h) + r*r );
k = sqrt(21.46*21.46+16*16);
        sh = (DEGREE(CosLaw(l,22,k)) - DEGREE(atan((H-h)/r))) - 100;
        el = DEGREE(CosLaw(22,k,l)) + DEGREE(atan(21.46/16)) -90;
        wr = 0;

/* pick object at calculated radius and angle */
        fprintf(stdprn,"%s\n","RS");
        delay(500);
        fprintf(stdprn,"%s\n","GO");
        delay(100);
        sprintf(cmd,"MI %d,%d,%d,0,0,0\n",(int)bd*40,(int)sh*40,(int)el*40);
        fprintf(stdprn,"%s\n",cmd);
        delay(1000);
        fprintf(stdprn,"%s\n","OG");
        delay(100);
        fprintf(stdprn,"%s\n","GO");
        delay(100);
        fprintf(stdprn,"%s\n","RS");
        delay(500);
      }else { printf("Sorry, I can't turn my arm to pick it.\n"); }
    }else { printf("Sorry, it's too far to pick.\n"); }

}
```