หุ่นยนต์ติดตามวัตถุโดยการประมวลผลภาพ

OBJECT TRACKING ROBOT BY IMAGE PROCESSING

โดย

นาย จักร์พงศ์ ชุ่มใจ

นาย สุจินต์ ศรีดา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

หุ่นยนต์ติดตามวัตถุโดยการประมวลผลภาพ

OBJECT TRACKING ROBOT BY IMAGE PROCESSING

โดย

นาย จักร์บดี ชุ่มใจ รหัส 45015270

นาย สุจินต์ ศรีดา รหัส 45015301

อาจารย์ที่ปรึกษา

ดร. สุรพันธ์ เอื้อไพบูลย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

รายงานเรื่อง หุ่นยนต์ติดตามวัตถุโดยการประมวลผลภาพ

OBJECT TRACKING ROBOT BY IMAGE PROCESSING

จัดทำโดย นาย จักร์บดี ชุ่มใจ รหัส 45015270

นาย สุจินต์ ศรีดา รหัส 45015301

อาจารย์ที่ปรึกษา ดร. สุรพันธ์ เอื้อไพบูลย์

รายงานฉบับนี้ได้ผ่านการตรวจสอบจากอาจารย์ที่ปรึกษาแล้ว

ลงชื่อ ..... อาจารย์ที่ปรึกษา

(ดร. สุรพันธ์ เอื้อไพบูลย์)

วันที่ 28 / มี.ค. / 48

หุ่นยนต์ติดตามวัตถุโดยการประมวลผลภาพ

จักร์บดี ชุ่มใจ
สุจินต์ ศรีดา
ดร.สุรพันธ์ เอื้อไพบูลย์ (อาจารย์ที่ปรึกษา)
ปีการศึกษา 2547

บทคัดย่อ

ในปริญญานิพนธ์ฉบับนี้เป็นการศึกษาเกี่ยวกับการประมวลผลรูปภาพเชิงเลขโดยการใช้คอมพิวเตอร์ประมวลผลและควบคุมให้ตัวหุ่นยนต์สามารถติดตามวัตถุที่ถูกกำหนดเป็นเป้าหมายภายในสนามจำลองได้เองอย่างอัตโนมัติ

โดยมีจุดมุ่งหมายเพื่อให้สามารถสร้างและออกแบบระบบประมวลผลภาพเชิงเลขโดยการเขียนโปรแกรมขึ้นเองเพื่อสร้างฟังก์ชั่นประมวลผลที่เหมาะสมให้สามารถนำไปควบคุมอุปกรณ์ต่างๆให้ทำงานตามต้องการ และยังเป็นตัวเสริมความรู้พื้นฐานในการใช้งานขั้นสูงในอนาคตต่อไป

OBJECT TRACKING ROBOT BY IMAGE PROCESSING

JAKBODEE CHUMJAI

SUJIN SRIDA

DR.SURAPHAN AIRPIBOON (ADVISER)

Abstract

This thesis is present to study about digital image processing by using a computer for process and control a robot to tracking an object everywhere it's placed or moved in the simulation field automatically.

The purpose of this report is design and creation the computer programs about digital image processing system for control any devices we wanted and addition basic knowledge for advancement in the future.

สารบัญ

หน้า

สารบัญรูปภาพ

บทที่ 1
บทนำ

เครื่องจักกลระบบอัตโนมัติภายในโรงงานอุตสาหกรรมในปัจจุบันนี้ นับวันก็ยิ่งต้องมีความต้องการความสามารถในการทำงานที่ต้องใช้ความละเอียดและแม่นยำที่สูงมายิ่งขึ้นเรื่อยๆ ยิ่งงานในด้านการประกอบวงจรอิเล็กทรอนิกส์ที่นับวันก็ยิ่งมีการพัฒณาอุปกรณ์ให้มีขนาดที่เล็กลงเรื่อยๆแล้วก็ยิ่งต้องการเครื่องจักกลที่มีความแม่นยำในการวางตำแหน่งของตัวอุปกรณ์สูงมากขึ้นด้วย ซึ่งแนวโน้มของอุปกรณ์ที่จะถูกใช้งานในด้านวงจรอิเล็กทรอนิกส์ในปัจจุบันและอนาคต จะเป็นอุปกรณ์จำพวก เซอเฟส เมาท์ ดีไวซ์ (Surface Mount Devices ,SMD) ที่ถูกออกแบบและสร้างให้มีขนาดเล็กลงเรื่อยๆจนยากที่จะนำมาประกอบด้วยตนเองอย่างเช่นในอดีต ดังนั้นเครื่องจักที่จะถูกใช้งานเพื่อทำการประกอบวงจรที่มีความละเอียดสูงมากนี้จึงต้องอาศัยระบบควบคุมที่มีความน่าเชื่อถือสูง ซึ่งระบบที่ว่านี้ก็คงหนีไม่พ้นการประมวลผลด้วยข้อมูลภาพถ่ายหรือ (Digital Image Processing) ซึ่งเป็นการใช้ภาพถ่ายจากกล้องถ่ายภาพหรือกล้อง วีดิโอ มาทำการประมวลผลแทนการใช้การตรวจวัดแบบอื่นที่ใช้ตัวเซ็นเซอร์มาทำการวัดโดยการสัมผัส ซึ่งจะมีข้อเสียตรงที่การสร้างเซ็นเซอร์มาใช้งานจะมีความยุ่งยากในการสร้าง การออกแบบ ไปจนถึงการติดตั้ง ดังนั้นแนวคิดในการใช้ (Digital Image Processing)จึงเป็นข้อได้เปรียบตรงที่ มีความยืดหยุ่นสูงต่อการปรับการทำงานในรูปแบบต่างๆ และราคาถูกกว่าเมื่อเทียบกับวิธีอื่น .

ซึ่งในรายงานฉบับนี้ ได้ทำการจำแนกรายละเอียดของงานที่ได้ทำไปแล้วออกเป็นบทต่างๆ ดังที่ได้แสดงไว้ข้างล่างนี้แต่ยังเป็นเพียงส่วนหนึ่งของโครงงานทั้งหมดที่ได้กำหนดขอบเขตไว้ในตอนแรกเท่านั้น

บทที่ 1 บทนำ จะแสดงรายละเอียดอย่างคร่าวๆเกี่ยวกับขอบเขตโครงงาน แนวคิดในการศึกษาเกี่ยวกับการประมวลผลภาพ (Image Processing) และการประยุคการใช้งานให้เข้ากันได้กับรูปแบบของงานที่จะทำ

บทที่ 2 ทฤษฎี โดยจะกล่าวถึงทฤษฎีที่เกี่ยวข้องที่เกี่ยวข้องของการประมวลผลภาพ (Image Processing)ภายในโครงงาน ,ไมโครคอนโทรเลอร์ เอ็มซีเอ็ส-51(Microcontroller MCS-51) ,พอร์ตขนาน(Parallel Port) ที่ใช้ในการควบคุมหุ่นยนต์และหลักการพื้นฐานต่างๆ ที่นำมาใช้ในการเขียนโปรแกรม

บทที่ 3 การออกแบบ โดยจะกล่าวถึงแนวคิดและวิธีการทำงานของโปรแกรมที่ใช้ในโครงงาน เช่น ส่วนของโปรแกรมประมวลผลข้อมูลภาพ, โปรแกรมควบคุมไมโครคอนโทรเลอร์, ส่วนรีโหมดคอนโทรเพื่อการควบคุมไร้สาย

บทที่ 4 การทดลอง และผลการทดลอง โดยจะกล่าวถึงการทดสอบ และผลการทดสอบของโครงงานเกี่ยวกับความคืบหน้าของงาน ว่าได้นำเดินงานไปตามขอบเขตที่ได้ระบุไว้มากน้อยเพียงใด และกล่าวถึงผลการทดสอบของเนื้องานที่ได้ทำไปแล้ว ว่ามีแนวโน้มว่าสามารถทำได้ตามที่ระบุไว้จริงหรือไม่

บทที่ 5 สรุป และวิจารณ์ จะกล่าวถึงปัญหาที่พบในการทำงานและแสดงแนวความคิดในการแก้ปัญหาที่ได้เกิดขึ้นว่าจะทำอย่างไร การพัฒนาโปรแกรมให้มีความสามารถในการทำงานที่เร็วขึ้นนั้นทำได้อย่างไร.

1.1 รายละเอียดพอสังเขปของโครงงาน

ในโครงงานชุดนี้จะใช้กล้อง วีดิโอ ส่งภาพเคลื่อนไหวเพื่อส่งต่อให้กับเครื่องคอมพิวเตอร์ประมวลผลว่าจะบังคับให้หุ่นยนต์ให้สามารวิ่งเข้าหาวัตถุที่กำหนดไว้โดยที่จุดหมายปลายทางนั้นจะเป็นวัตถุที่ไม่ได้หยุดอยู่กับที่ โดยการเขียนโปรแกรมของโครงงานนี้ได้ใช้โปรแกรม บอแลนด์ ซี++ บิวล์เดอร์ 6.0 (Borland C++ Builder 6.0)ในการประมวลผลทางด้านภาพ (Digital Image Processing)และควบคุมหุ่นยนต์โดยติดต่อผ่ายพอร์ตขนานของเครื่องคอมพิวเตอร์ ซึ่งเป็นโปรแกรมได้มีคอมโพเนนท์(Components)หรือเครื่องมือให้สามารถใช้งานได้สะดวกเป็นอย่างดี

1.2 วัตถุประสงค์ของโครงงาน

ในการศึกษาและทดลองในโครงงานชุดนี้ก็เพื่อศึกษาการประมวลผลภาพจากคอมพิวเตอร์ซึ่งจะทำให้สามารถเข้าใจแนวคิดการออกแบบโปรแกรมเพื่อเข้าใจว่าคอมพิวเตอร์นั้นทำการตรวจจับข้อมูลภาพรู้จักภาพและประมวลผลมันได้อย่างไรและนำผลที่ได้ไปทำควบคุมการเคลื่อนที่ของอุปกรณ์อื่นๆให้ทำงานตามที่กำหนดไว้ซึ่งในโครงงานนี้คือการควบคุมให้หุ่นยนต์ติดตามวัตถุได้เองโดยอัตโนมัติโดยใช้คอมพิวเตอร์เป็นตัวประมวลผลสัญญาณที่รับเข้ามาจากกล้อง วีดิโอ

1.3 ประโยชน์ที่คาดว่าจะได้รับ

สิ่งที่คาดว่าจะได้รับหลังการการศึกษาและทดลองสร้างโครงงานชุดนี้คือสามารถที่จะประยุคและพัฒนาการใช้คอมพิวเตอร์ในงานควบคุมอุปกรณ์อื่นๆได้กว้างขวางยิ่งขึ้นโดยเฉพาะในงานด้านการประมวลผลด้วยข้อมูลภาพโดยใช้คอมพิวเตอร์ซึ่งจะเป็นที่สนใจและถูกใช้งานในการควบคุมมากยิ่งขึ้นในชีวิตประจำวันและในอนาคต

บทที่ 2

ทฤษฎี

2.1 ดิจิตอล อิมเมจ โปรเซสชิ่ง (Digital Image Processing)

การประมวลผลภาพด้วยคอมพิวเตอร์นั้น โดยทั่วไปแล้วสามารถทำการแบ่งแยกออกเป็น 2 ประเภทใหญ่ๆด้วยกันคือ การประมวลผลภาพในระดับต่ำ (Low-level Image Processing) และการประมวลผลภาพในระดับสูง (High-level Image Processing)ซึ่งสามารถอธิบายความแตกต่างของการประมวลผลทั้งสองระดับได้ดังนี้

การประมวลผลภาพในระดับต่ำนั้นเป็นการประมวลผลเชิงตัวเลขเกือบทั้งหมดเพื่อหาตัวแปรต่างๆ มาอธิบาย ข้อมูลภาพ และมีจุดประสงค์ที่จะนำตัวแปรเหล่านี้ไปใช้ในการประมวลผลภาพในระดับสูงต่อไป การประมวลผลภาพในระดับต่ำโดยทั่วไปจะประกอบไปด้วยการกำจัดสัญญาณรบกวน , การทำให้ภาพคมชัดขึ้น, การหาขอบเขตของภาพ, การทำเซ็กเม็นท์ภาพหรือการแบ่งแยกวัตถุภายในภาพ, การสร้างภาพไบนารี เป็นต้น

ในขณะที่ การประมวลผลภาพในระดับสูงคือการทำให้คอมพิวเตอร์รู้จักและเข้าใจภาพได้ เช่น การจดจำรูปแบบของตัวเลขตัวอักษร, แยกแยะวัตถุที่มีรูปร่างแตกต่างไปจากเดิมที่ถูกกำหนดไว้, การจดจำใบหน้าคน เป็นต้น

ซึ่งในโครงงานนี้ก็จัดได้ว่า เป็นการประมวลผลสัญญาณภาพแบบระดับต่ำ เพราะลักษณะของการประมวลผลที่จะทำการแยกแยะระหว่างพื้นฉากหลังกับวัตถุเท่านั้น และตรวจสอบเฉพาะวัตถุที่เป็นหุ่นยนต์และวัตถุที่ถูกสมมุติให้เป็นจุดหมายปลายทางเพื่อให้สามารถทำการแบ่งแยกความหมายของภาพที่รับเข้ามาได้ ดังนั้นในส่วนทฤษฎีนั้นจะเสนอส่วนที่เรียกว่าการแบ่งส่วนภาพหรือ (Image Segmentation)มาเป็นส่วนแรกดังนี้

2.2 การแบ่งส่วนภาพ

การแบ่งส่วน (Segmentation) นั้นเป็นการกำหนดขอบเขตของวัตถุภายในภาพบริเวณรอบจุดใดๆที่แตกต่างกันให้สามารถเห็นความแตกต่างชัดขึ้นเพื่อนำไปประมวลผลขั้นสูงต่อไปได้ง่ายขึ้น ซึ่งสิ่งนี้เป็นส่วนที่จำเป็นต่อการวิเคราะห์ภาพว่าวัตถุในภาพนั้น มีขนาดใหญ่เท่าใด, ส่วนไหนเป็นพื้นหลัง, แล้ววัตถุทั้งหมดนั้นมีอยู่เท่าไหร่ ดังนั้นการแบ่งส่วน(Segmentation)นั้นเป็นพื้นฐานที่จำเป็นสำหรับการชี้และการอธิบายวัตถุที่อยู่ในรูปภาพ ให้สามารถถูกประมวลผลโดยโปรแกรมได้ดีและสะดวกยิ่งขึ้น ซึ่งการแบ่งส่วน (Segmentation)นี้สามารถทำได้ง่ายๆโดยการนำข้อมูลภาพที่รับเข้ามานำไปเปรียบเทียบกับค่าคงที่ค่าหนึ่งซึ่งเรียกว่า ค่าเทรชโฮลด์ (Threshold)และภาพที่ได้หลังจากการเปรียบเทียบจะถูกเรียกว่า ภาพไบนารี่(Binary Image)

2.2.1 **การทำเทรชโฮลด์ (Thresholding technique)**

การทำเทรชโฮลด์ถือว่าเป็นเทคนิคที่สำคัญในการประมวลผลภาพในส่วนของการทำเซกเมนต์ภาพ ซึ่งจุดประสงค์ของการทำเซกเมนต์ภาพ คือการแยกองค์ประกอบของภาพไปเป็นส่วนประกอบย่อยๆ ที่มีความสัมพันธ์กันทางกายภาพของภาพนั้น และส่วนประกอบที่ถูกแยกออกมานั้นอาจถูกนำไปประมวลผลภาพในส่วนอื่นได้ต่อไป ซึ่งการทำเซกเมนต์ภาพจะมีหลักการทำงานในแนวเดียวกันกับสายตาของคน คือสามารถแยกลักษณะเด่นออกมาจากภาพที่มองเห็นได้และเทคนิคการทำเทรชโฮลด์ ซึ่งถือว่าเป็นเทคนิคในการแยกองค์ประกอบของภาพที่ง่ายเทคนิคหนึ่งมีหลักการว่า จุดภาพที่มีคุณสมบัติอยู่ในช่วงใดๆจะถูกจัดเป็นกลุ่มได้โดยที่ระดับความเข้ม นั้นสามารถที่จะแบ่งแยกกลุ่มของจุดภาพออกเป็น 2 กลุ่มได้อย่างชัดเจน คือกลุ่มของวัตถุ (object) ซึ่งจะมีระดับความเข้มของภาพ g(x,y) ค่อนข้างต่ำ (มืด) กับกลุ่มของส่วนที่เป็นพื้นหลัง (background) ที่จะมีระดับความเข้มของภาพ g(x,y) ค่อนข้างสูง (สว่าง) ดังเช่นภาพที่ 2.1 ซึ่งแสดงฮิสโตแกรมของระดับความเข้มของภาพที่ถูกแบ่งออกเป็น 256 ระดับ(ในกรณีของภาพขนาดละเอียดที่ 8 บิต) จะเห็นได้ว่าการที่จะแยกกลุ่มข้อมูลออกเป็น 2 กลุ่มอย่างชัดเจนย่อมสามารถทำได้โดยการเลือกค่าเทรชโฮลด์ที่ มีค่าความเข้มอยู่ระหว่างกลุ่มทั้งสอง บนฮิสโตแกรมระดับความเข้มของภาพแล้วทำการจรวจสอบแต่ละจุดภาพว่าถ้ามีค่า g(x,y) น้อยกว่าค่าเทรชโฮลด์ ถือว่าเป็นจุดภาพของวัตถุที่แสดงได้ ด้วยจุดดำแต่หากว่าจุด g(x,y) นั้นมีค่ามากกว่าหรือเท่ากับค่าเทรชโฮลด์ก็ถือว่าเป็นจุดภาพใน ส่วนพื้นหลังที่แสดงได้ด้วยจุดขาว ดังนั้นข้อมูลภาพ gthr(x,y) ที่ผ่านการทำเทรชโฮลด์สามารถนิยามได้ดังนี้

$$\left.\begin{array}{lll} \text{IF} & g(x,y) \geq T & gthr(x,y)\ \text{Object} = 255 \\ \text{Else} & & gthr(x,y)\ \text{Background} = 0 \end{array}\right\} \quad \ldots\ldots 2.1$$

ซึ่งในเงื่อนไขข้างต้นนี้เราอนุมานว่า สนใจเฉพาะวัตถุที่สว่างกว่าพื้นหลัง หากว่าเป็นการหาวัตถุที่วางอยู่ในพื้นหลังที่สว่างกว่าวัตถุก็จะแสดงได้ว่า

$$\left.\begin{array}{lll} \text{IF} & g(x,y) \leq T & gthr(x,y)\ \text{Object} = 255 \\ \text{Else} & & gthr(x,y)\ \text{Background} = 0 \end{array}\right\} \quad \ldots\ldots 2.2$$

โดยที่

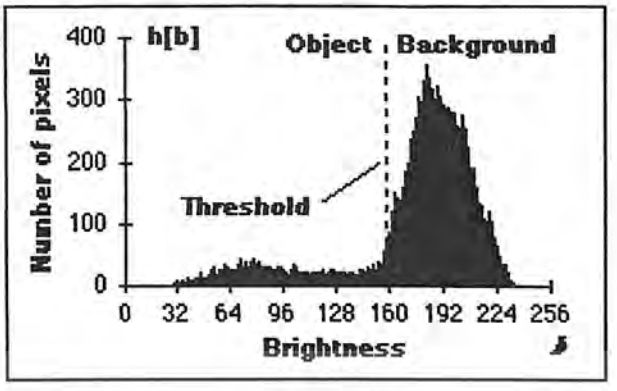
gthr(x,y) คือ ข้อมูลภาพผลลัพธ์เป็นไบนารี

g(x,y) คือ ข้อมูลภาพอินพุทที่มีระดับความเข้ม 0 ถึง L ระดับ

T คือ ค่าเทรชโฮลด์ เป็นค่าคงที่ที่มีค่าระหว่าง 0 ถึง L

0 คือ จุดดำ (ส่วนที่เป็นวัตถุ)

255 คือ จุดขาว (ส่วนที่เป็นพื้นหลัง)



รูป 2.1 แสดงฮิสโตแกรมของระดับความเข้มของภาพขนาด 256 ระดับ

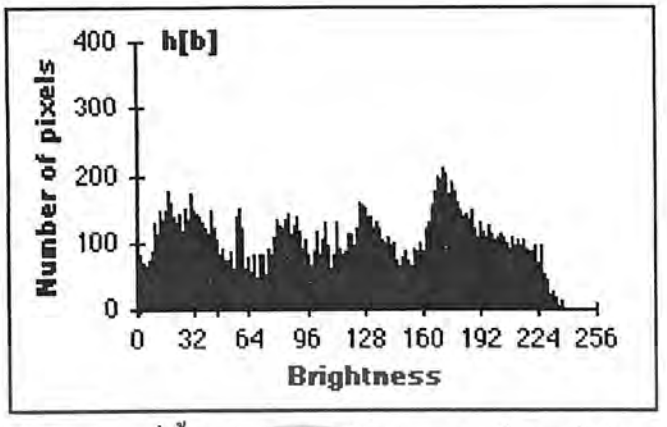
เมื่อได้ทำการศึกษาเกี่ยวกับการทำเทรชโฮลด์ไปแล้วภาพที่ผ่านการทำเทรชโฮลด์เราสามารถเรียกอีกอย่างหนึ่งว่าภาพไบนารี่ (Binary Image) ก็ย่อมเกิดคำถามที่จะตามมาว่าแล้วจะใช้ค่าเทรชโฮลด์เท่าไหร่ดีละ? ซึ่งการกำหนดค่าของการทำเทรชโฮลด์นี้สามารทำได้อยู่ 2 รูปแบบคือการกำหนดค่าเทรชโฮลด์แบบคงที่ตลอดทั้งภาพ(Fixed Threshold)และอีกวิธีคือเลือกค่าเทรชโฮลด์จากระดับฮีสแกรม (Histogram-derived thresholds)

2.2.1.1**เทรชโฮลด์แบบคงที่ตลอดทั้งภาพ(Fixed Threshold)**

ซึ่งเป็นทางเลือกหนึ่งที่ถูกใช้ทำเทรชโฮลด์โดยไม่ได้ขึ้นกับข้อมูลภาพที่ได้นำมาประมวลผลโดยการทำเช่นนี้ได้ก็ต่อเมื่อเราทราบแล้วว่าวัตถุที่มีสีเข้มมากๆ ได้ถูกวางไว้บนพื้นหลังที่เป็นสีขาวหรือสีอ่อนกว่าวัตถุโดยที่พื้นหลังจะต้องมีความสว่างที่เท่ากันตลอด ดังนั้นแล้วเราสามารถเลือกค่าเทรชโฮลด์ เป็น128 จากระดับความเข้มของภาพขนาด 0 ถึง 255 ได้ซึ่งจะทำให้ผลลัพธ์ที่ได้ถูกต้องมากที่สุด แต่วิธีการนี้จะมีข้อจำกัดตรงที่หากว่าความสว่างของพื้นหลังมีค่าไม่คงที่แล้วจะทำให้เกิดการผิดพลาดในการตรวจจับวัตถุได้

2.2.1.2 **เทรชโฮลด์จากระดับฮีสแกรม (Histogram-derived thresholds)**

ในกรณีที่ข้อมูลภาพมีความไม่สม่ำเสมอเกิดขึ้นในส่วนของวัตถุ หรือส่วนของพื้นหลังหรือในทั้งสองส่วนซึ่งภาพในลักษณะเช่นนี้ ฮีสโตแกรมระดับความเข้มของภาพที่เกิดขึ้นอาจมีลักษณะดังรูปที่ 2.2 ดังนั้นในการกำหนดหาค่าเทรชโฮลด์เพื่อจะนำมาใช้งานจึงสามารถใช้หลักการของการหาค่าเฉลี่ย(Mean)ระหว่างระดับความสว่างที่ความเข้มภาพสูงสุดกับระดับความมืดที่ความเข้มภาพสูงสุดเช่นกันโดยสามารถแสดงได้ดังสมการดังต่อไปนี้



รูป 2.2 กราฟฮีสโตแกรมที่พื้นหลังกับวัตถุมีระดับความเข้มภาพไม่แตกต่างกันมาก

$$T = \frac{h[g(x,y)](\text{Brightness}) + h[g(x,y)](\text{Darkness})}{2}$$

โดยที่

h[g(x,y)](Brightness) คือ ความสว่างของจุดที่มีจำนวนของ พิเซลส์ มากที่สุด

h[g(x,y)](Darkness) คือ ความมืดของจุดที่มีจำนวนของ พิเซลส์ มากที่สุด

ซึ่งเมื่อทำการเมื่อคำนวณเทรชโฮลด์ได้แล้ว ก็สามารถทำการเซกเมนต์ภาพได้โดยนำค่าเทรชโฮลด์ ที่ได้มาแทนค่าในสมการที่ 2.1 หรือ 2.2 ได้ทันที

2.3 การแยกส่วนสี(Color Segmentation)

ในโครงงานชุดนี้ได้ทำการเลือกใช้สีที่จะนำมาประมวณผลในโปรแกรมอยู่ด้วยกันทั้งหมดอยู่ 3 สีด้วยกันซึ่งทั้งหมดจะเป็นแม่สีได้แก่ สีแดง สีน้ำเงิน(แทนสีของตัวหุ่นยนต์) และสีเขียว(แทนสีของวัตถุปลายทาง) ซึ่งเหตุผลที่เลือกใช้แม่สีดังกล่าวก็เพราะ ว่าเราจะสามารถออกแบบส่วนโปรแกรมจะใช้ในการกรองสีได้ง่ายกว่าการใช้สีอื่นๆที่ไม่ใช้แม่สี รูปที่2.5แสดงฮีสโตรแกรม ของภาพแม่สีทั้งสามที่รับเข้ามาจากกล้อง[1] ซึ่งได้ถูกแปลงให้เป็นภาพเกรย์(Gray Image) ขนาดความละเอียดที่ 8 บิต ก่อนการคำนวณหาระดับฮีสโตรแกรม เหตุที่ต้องแปลงภาพให้เป็นภาพเกรย์ก่อนก็เพราะว่าเพียงต้องการแสดงให้เห็นถึงความแตกต่างของ กลุ่มของจุดสี (Color Pixels) ของแม่สีทั้งสามซึ่งจะเห็นว่าเมื่อนำภาพสีที่มีวัตถุแม่สีทั้งสามมาแปลงเป็นภาพเกรย์ แล้ว จะได้ระดับของอีสโตรแกรมที่มีความแตกต่างดังนี้คือ สีน้ำเงินที่ถูกแปลงให้กลายเป็นระดับสีเทา(Gray Color)แล้วจำมีระดับความสว่างที่บริเวณ 0 < ระดับความสว่าง > 64

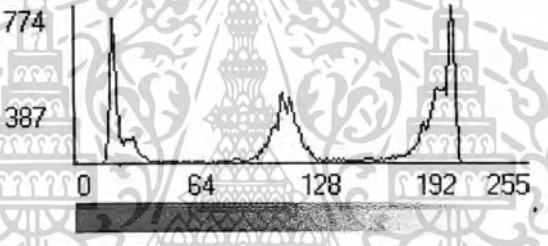
---

[1] ภาพที่รับมาจากกล้องวีดิโอจะถูกตั้งความลึกของสีไว้ที่ 24 บิต คือ Red 8 Bits + Green 8 Bits + Blue 8 Bits

เพราะมีความสว่างน้อยกว่าแม่สีอื่น สีแดงจะอยู่บริเวณ 64 < ระดับความสว่าง > 128 และสีเขียว 128 < ระดับความสว่าง > 255 จะเห็นว่าแม่สีต่างๆนั้นจะกระจายออกเป็นกลุ่มๆอย่างเห็นได้ชัดเจนแสดงดังรูปต่อไปนี้

รูป 2.3 ภาพสีจากกล้องขนาด24บิต

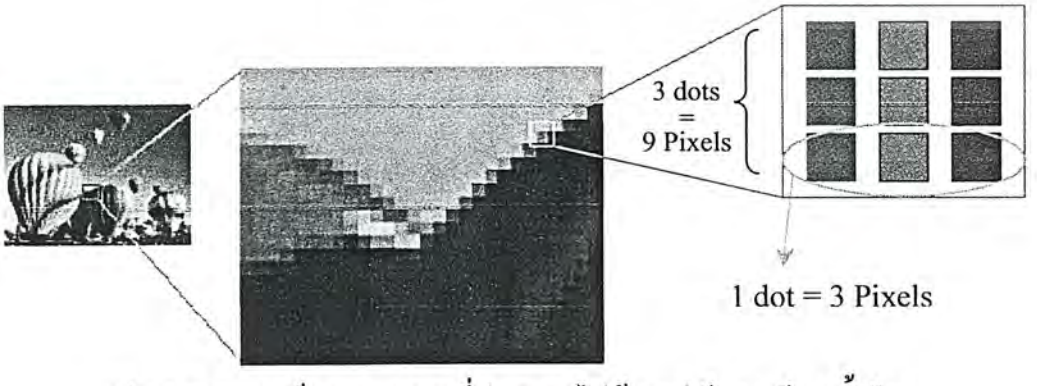
รูป 2.4 ภาพสีที่ถูกแปลงเป็นสีเทาขนาด8บิตแล้ว



รูป 2.5 แสดงระดับฮีสโตรแกรมของรูป 2.4

2.3.1 **การเข้าถึงระดับพิเซลในภาพ(Pixel Accessing)**

ภาพหนึ่งภาพหรือหนึ่งรูปนั้นจะประกอบไปด้วยจุดสีต่างๆมากมายที่มีความแตกต่างกันของระดับสีเต็มไปหมด ซึ่งสีต่างๆที่เรามองเห็นล้วนเกิดจากการผสมกันระหว่างแม่สีทั้งสามคือ แดง เขียว และน้ำเงิน เมื่อพิจารณาเข้าไปถึงจุดเล็กสุดของภาพจะประกอบไปด้วยจุดสี 3 จุดซึ่งถูกเรียกว่าพิเซล (Pixels) จะประกอบไปด้วยแม่สีทั้งสามที่ได้กล่าวไปแล้วซึ่งได้แสดงดังรูปที่2.6



รูป 2.6 แสดงจุดเล็กสุดของภาพที่ประกอบไปด้วยแม่ สีแดง เขียว น้ำเงิน

ซึ่งที่สีสันต่างกันความความสว่างของแต่ละพิเซลก็จะไม่เหมือนกัน เช่น จุดภาพจุดหนึ่งได้ถูกกำหนดให้แสดงสีแดงซึ่งเป็นแม่สี (ไม่มีสีอื่นเลยนอกจาสีแดง) ดังนั้นจึงขอสมมุติให้พิเซลทั้งสามเสมือนว่าเป็นหลอดไฟสามสีคือ หลอดไฟสีแดง หลอดไฟสีเขียว และหลอดไฟสีน้ำเงินตามลำดับ เพื่อให้ง่ายต่อความเข้าใจ ในการทำงานของหลอดไฟทั้งสาม ก็จะมีเพียงหลอดสีแดงเท่านั้นที่ติดสว่าง (ความสว่างถูกกำหนดโดยระดับฮีสโตรแกรม 0 - 255 ระดับ หรือ 8บิต) ส่วนสีที่เหลือก็จะดับอยู่แบบนั้น และถ้าจุดภาพหรือ Dot ดังกล่าวถูกกำหนดให้แสดงขาว หลอดไฟทั้งสามก็จะต้องติดสว่างเหมือนกันหมด และความสว่างของสีของทั้งสามหลอดก็จะต้องเท่ากัน (ความสว่างเท่ากับ255 เพราะถ้าต่ำกว่านี้จะเป็นสีเทา) เป็นต้น

ดังนั้นในการเข้าถึงข้อมูลภาพในระดับพิเซลก็เพื่อทำการอ่านค่าความสว่างของสีต่างๆ ว่ามีขนาดเท่าไหร่บ้างเพื่อนำมาเปรียบเทียบกับค่าคงที่ที่จะทำการแยกสี(Color Segmentation)ซึ่งการเปรียบเทียบค่านี้ก็คือการทำเทรชโฮลด์สีนั้นเอง ซึ่งจะมีหลักอยู่ว่าจุดของภาพดังกล่าวมีระดับความสว่างของสีต่างๆอยู่ในช่วงของแม่สีใดบ้าง ถ้าสมมุติว่าภาพจากกล้องที่รับเข้ามามีความสว่างอยู่ในย่านของสะน้ำเงินพอดี (เช่นความสว่างของพิเซล สีน้ำเงิน-สีแดง-สีเขียว = 150-90-65)ซึ่งภาพรวมของจุดสีดังกล่าวคือสีน้ำเงินเพียงแต่ว่ายังไม่ใช้สีน้ำเงินที่แท้จริงซึ่งมีค่าเท่ากับ (255-0-0) ดังนั้นจึงต้องทำการปรับความสว่างของพิเซลในจุดดังกล่าว(Pixels in Dot)โดยวิธีการทำเทรชโฮลด์ดังที่ได้กล่าวเอาไว้แล้ว

หลังจากที่เราได้ทำการแยกสีหรือ (Colors Segmentation) ไปแล้วลำดับต่อไปก็จะเป็นการตรวจจับสีหรือตรวจหาชนิดของวัตถุ(ตัวหุ่นยนต์และจุดหมายปลายทาง)ว่าอยู่ตรงส่วนใดของภาพซึ่งเรียกวิธีการนี้ว่า(Object Tracking)

2.4 การติดตามวัตถุ(Object Tracking)

ระบบที่ใช้การประมวลผลรูปภาพ(Image Processing)เพื่อการคัดแยกหรือตรวจหาวัตถุที่ต้องการจากรูปภาพนั้นโดยส่วนมากแล้วจะต้องมีวิธีการใดวิธีการหนึ่งที่จะใช้ติดตามหรือแยกแยะวัตถุที่ต้องการออกจากวัตถุอื่นๆที่ปะปนมากับในรูปออกให้ได้ ซึ่งก็ได้มีผู้คิดค้นวิธีการอยู่หลายแบบด้วยกัน และในรายงานเล่มนี้ก็ได้ค้นคว้าหาวิธีจากแหล่งข้อมูลขนาดใหญ่ ซึ่งก็คืออินเตอร์เน็ท (Internet) ซึ่งก็ได้พบวิธีที่น่าจะเป็นไปได้สำหรับการเขียนโปรแกรมมี 2 แบบดังนี้ วิธีแรก การแม็ชชิ่งกับแม่แบบ(Template Matching with Distance Transforms) หรือ (Model Comparing Method) ซึ่งวิธีการนี้เป็นวิธีที่มีความยืดหยุ่นค่อนข้างสูง ใช้ได้กับรูปภาพที่มีสภาพแสงไม่คงที่ แต่ก็มีความซับซ้อนในการออกแบบและเขียนโปรแกรมอย่างมากและใช้เวลาในการทำงานสูง ซึ่งจะกล่าวให้เห็นถึงหลักการทำของวิธีแรกอย่างคร่าวๆดังนี้

2.4.1 การแทรกกิ้งโดยใช้การเทียบกับต้นแบบ(Model Comparing Method)

ในวิธีการนี้เราจะต้องมีสิ่งที่สำคัญอันดับแรกก็คือ ต้นแบบ(Model) ที่จะใช้เปรียบเทียบซึ่งอาจได้จากรูปถ่ายหน้าคนที่ต้องการทดสอบแล้วนำภาพที่ได้มาทำการตัดทอนส่วนที่ไม่จำเป็นในรูปทิ้งไป (เก็บเอาแต่ลักษณะเด่นของหน้าคนในรูป) หลังจากที่ได้โมเดลมาแล้วก็นำมาเก็บไว้เพื่อรอการเรียกใช้งาน ขั้นต่อไปพิจารณารูปภาพที่จะนำมาทำการแทรกกิ้งใบหน้าคน โดยจะเตรียมจากจากการแยกส่วนภาพ ซึ่งในที่นี้คือการหาขอบภาพ (เพื่อให้ง่ายต่อการเปรียบเทียบ) หรือที่รู้จักกันในชื่อ Edge Detection เมื่อได้ทำการเตรียมรูปภาพที่จะนำไปเปรียบเทียบกับตัวโมเดลที่เก็บไว้แล้วนั้น ขั้นต่อไปก็เป็นส่วนของการเปรียบเทียบระหว่างโมเดลกับรูปภาพ ซึ่งจะมีวิธีการเพื่อใช้หาวิธีการเปรียบเทียบอีกขั้นหนึ่งนั้นก็คือ ดิสแทนซ์ ทรานส์ฟอร์ม(Distance Transform) และ หนึ่งในวิธีการนั้นคือวิธีของ Hausdorff Transform แต่จะไม่ขอกล่าวถึงรายละเอียดในรายงานเล่มนี้เนื่องจาเกินขอบเขต แต่ก็ได้รวบรวมเอกสารที่เกี่ยวข้องไว้ในซีดี-รอมเรียบร้อยแล้ว[2] และหลังจาที่ได้ทำการเปรียบเทียบระหว่างรูปภาพทั้งสองแล้ว ผลที่ได้แสดงดังรูป 2.7

ในหลักการตลอดหัวข้อที่ 2.4.1 ที่ได้กล่าวมานี้ นอกจากจะสามารถทำการแทรกกิ้งใบหน้าคนได้แล้วถ้าหากว่านำไปทำการจัดการกับโมเดลให้เป็นฐานข้อมูลที่เหมาะสมแล้วเราจะสามารถจดจำใบหน้าหรือรู้จักว่าคนที่กำลังอยู่ในภาพนั้นเป็นคนๆเดียวกันกับที่เก็บไว้ในฐานข้อมูลหรือไม่ได้อีกด้วย เรียกว่ารู้จักวัตถุ (Object Recognition) นั้นเอง.

รูป 2.7 แสดงการแทรกกิ้งโดยใช้การเปรียบเทียบกับโมเดล[3]

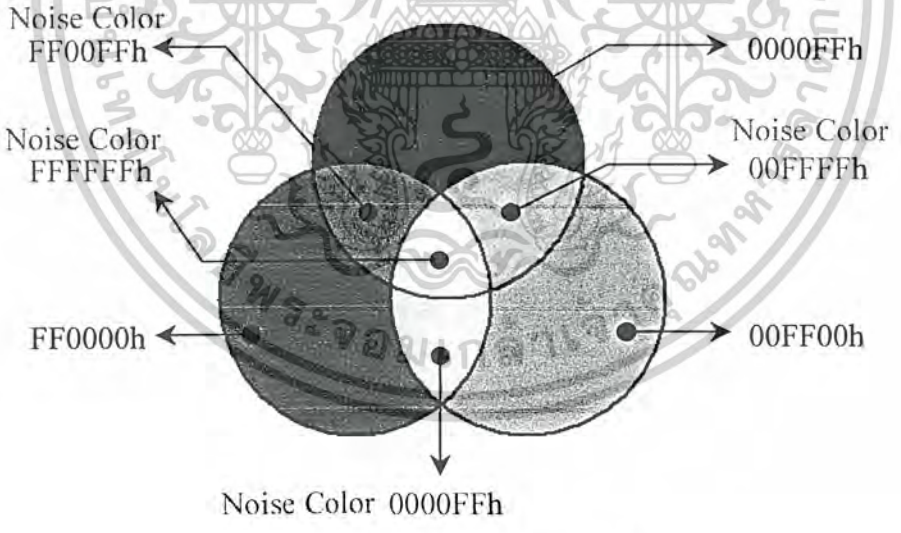
---

2 สามารถติดต่อได้ที่ ธุรการภาควิชา อิเล็กทรอนิกส์ ชั้น 3 ตึก บี

3 อ้างอิงจาก http://www.bioid.com

2.4.2 การแทรกกิ้งโดยการคัดแยกสีหรือกรองสี(Color Filtrations)

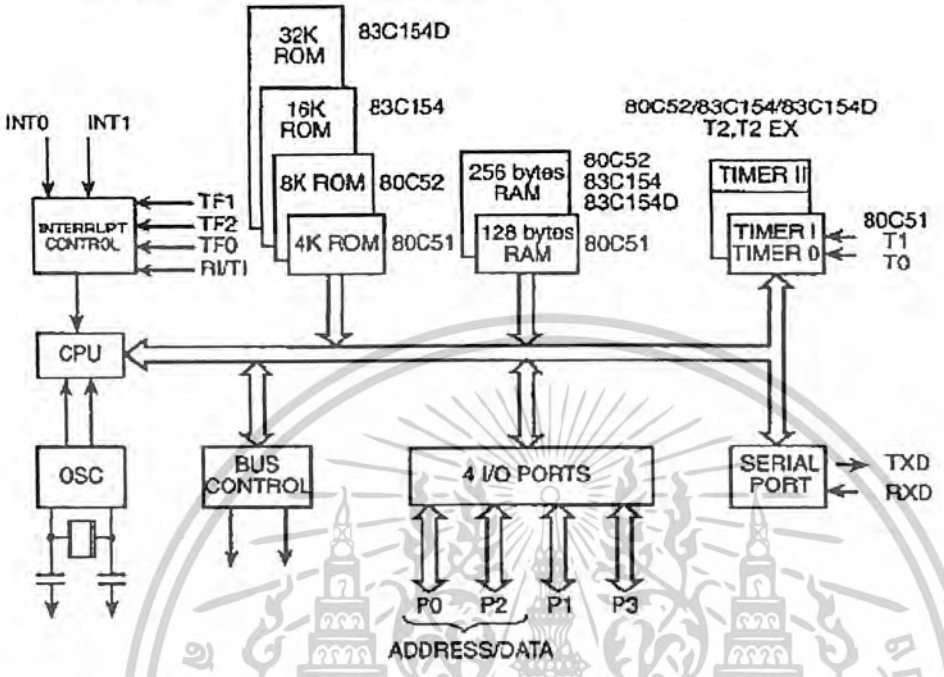
เป็นวิธีการที่ถูกในมาใช้ในโครงงานชุดนี้ เหตุผลก็เพราะสีที่ใช้ในโครงงานมีเพียงแม่สี 3 สีคือ สีแดง สีเขียว และสีน้ำเงิน เท่านั้นจำทำให้มีความรวดเร็วในการประมวลผลเนื่องจากการแทรกกิ้งแบบนี้จะใช้หลักการกรองเอาสีอื่นที่ไม่ต้องการทิ้งไป ให้เหลือไว้แต่สีของวัตถุที่ต้องการเท่านั้น แต่การกรองสีนั้นหากมีสัญญาณรบกวนที่เป็นสีเดียวกับวัตถุที่ต้องการหาในภาพ ก็จะเกิดการผิดพลาดในการแทรกกิ้งได้ ดังนั้นจะต้องคำนึงถึงขนาดของสิ่งที่จะแทรกกิ้งด้วยเพื่อตัดปัญหาของสัญญาณรบกวนออกไป จากนั้นจึงหาว่ากลุ่มของสีที่อยู่ในวัตถุที่มีขนาดเท่ากับที่ต้องการหรือไม่ ถ้าใช้ก็จะทำการหาขอบเขตหรือรัศมีของวัตถุดังกล่าว เพื่อหาตำแหน่งศูนย์กลางของวัตถุออกมาเราก็จะได้พิกัดที่จะนำไปใช้งานได้แล้ว รูป 2.8 แสดงการผสมสีกันของแม่สีทั้งสามซึ่งกำหนดให้แม่สีดังกล่าวมีระดับความสว่างของแต่ละสีเท่ากับ 255-255-255 = R-G-B (คือระดับที่เป็นไปได้ของเลขฐานสองขนาด 8 บิต * 3 สี) หรือเขียนแทนด้วยสัญญาลักษณ์เลขฐานสิบหกจะได้เป็น FFFFFFh ให้จะเห็นว่าบริเวณที่มีการซ้อนทับกันหระหว่างแม่สีจะเกิดสีใหม่ขึ้นมาซึ่งในการแทรกกิ้งโดยการกรองสีจะถือว่าสีนั้นคือสัญญาณรบกวนต้องกำจัดออกไปและจะกล่าวถึงวิธีการอีกครั้งในบทที่3



รูป 2.8 แสดงการผสมสีกันของแม่สีทั้งสาม

2.5 ไมโครคอนโทรลเลอร์และสเต็ปเปอร์มอเตอร์

2.5.1 โครงสร้างของไมโครคอนโทรลเลอร์



รูป 2.9 โครงสร้างของไมโครคอนโทรลเลอร์

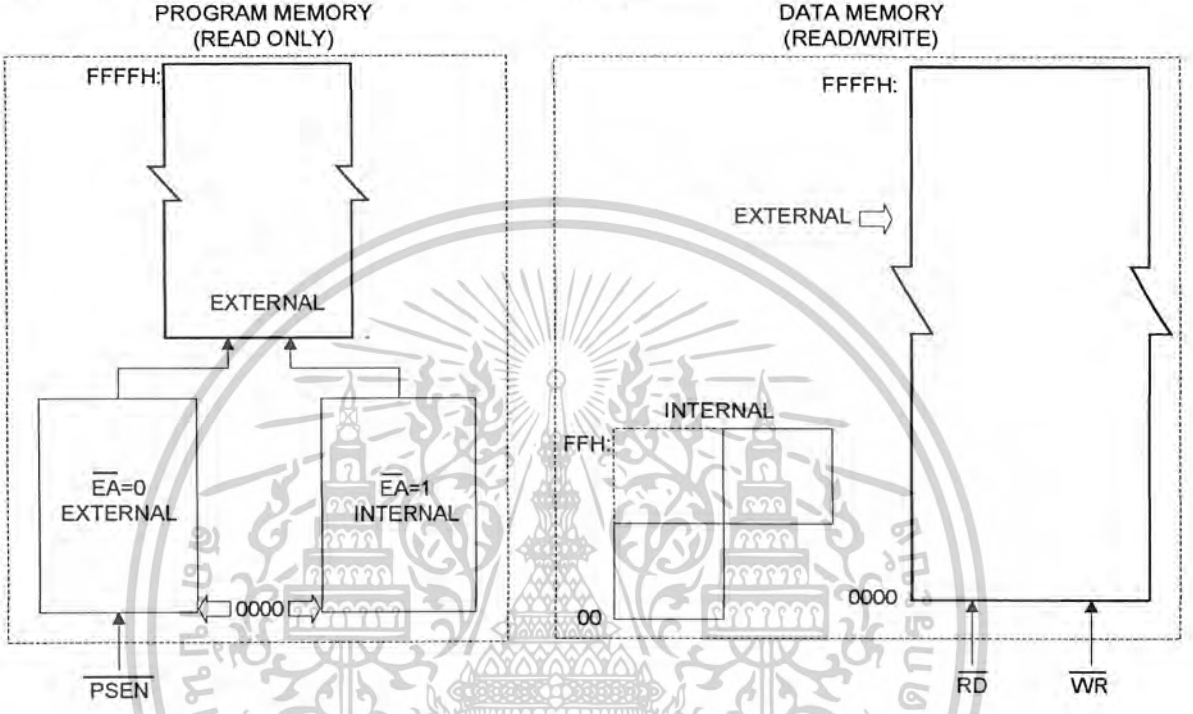
คุณสมบัติเบื้องต้นของ Microcontroller ในตระกูล MCS-51 จะมีดังนี้

- มี Core CPU ที่เป็น 8-Bit และชุดคำสั่งที่เหมาะในงานควบคุม และสามารถประมวลผลทาง Logic กับข้อมูลในระดับ BIT ได้
- มีหน่วยความจำโปรแกรม 4K ภายใน และรองรับการใช้งานของหน่วยความจำ โปรแกรมได้ถึง 64K
- มีหน่วยความจำ ข้อมูล (RAM) 128 Bytes ภายใน และรองรับการใช้งานของหน่วยความจำ ข้อมูล ได้ ถึง 64K
- มี Port ที่เป็นได้ทั้ง I / O ทั้งหมด 4 port และสามารถใช้งานได้ในระดับ BIT
- มีส่วน Timer / Counter ขนาด 16 Bit สองชุด สำหรับใช้ในการจับเวลา หรือนับจำนวน
- มี Full duplex UART สำหรับใช้ รับ/ส่ง ข้อมูลแบบอนุกรม รับ Interrupt ได้จาก 6 แหล่งกำเนิด โดย มี 5 ตำแหน่งของ ISR และการ Interrupt โดยสามารถจัดระดับความสำคัญได้ 2 ระดับมีตัวกำเนิดความถี่ Clock ภายใน

2.5.2 โครงสร้างของหน่วยความจำใน MCS-51

2.5.2 .1การแยกใช้หน่วยความจำโปรแกรมและข้อมูลภายใน

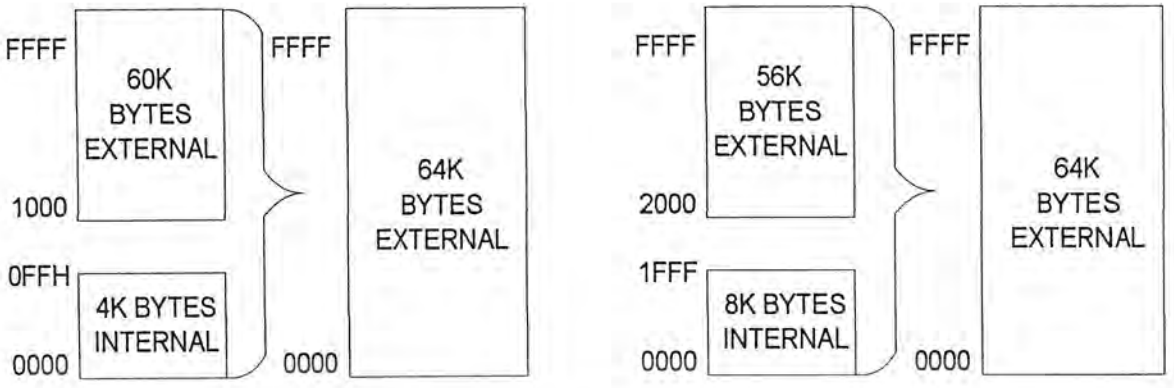
สำหรับ Microcontroller MCS-51 นั้นถูกออกแบบมาให้ มีหน่วยความจำสำหรับเก็บโปรแกรม (Op-code) และหน่วยความจำสำหรับเก็บข้อมูลที่แยกออกจากกันดังรูป



รูป 2.10 โครงสร้างของหน่วยความจำใน MCS-51

การออกแบบของ Data memory ที่แยกออกมานี้ จะทำให้สามารถเรียกใช้งานได้โดยใช้ Address เพียง 8 Bit เท่านั้น ซึ่งจะทำได้อย่างรวดเร็วใน CPU ที่เป็น 8 Bit แต่การใช้ Address เพียง 8 Bit นี้ จะทำให้อ้างถึงตำแหน่งของหน่วยความจำได้เพียง 256 ตำแหน่งเท่านั้น (00h – FFh) ซึ่งก็เพียงพอสำหรับการอ้างถึงตำแหน่งของ Internal Data Memory อย่างไรก็ดีการอ้างถึงตำแหน่ง Data memory โดยใช้ Address แบบ 16 Bit สำหรับ External data memory ก็สามารถทำได้ โดยใช้ DPTR: Data pointer (Data memory address register)

ส่วนของ Program memory จะเป็นหน่วยความจำที่อ่านได้เพียงอย่างเดียว และสามารถมีได้ทั้งหมด 64K ตำแหน่ง

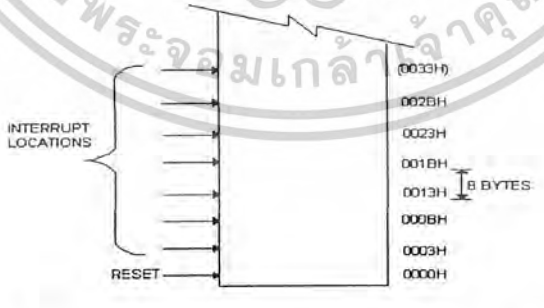


รูป 2.11 โปรแกรมเมมโมรี่ของ MCS 51

สำหรับ MCS-51 ในแบบที่มี Program memory ภายใน ก็จะมีพื้นที่ในการเก็บโปรแกรมภายใน IC เอง 4K, 8K, 16K หรือ 32K Address (ตามเบอร์ของ IC ที่ใช้) สำหรับ Address ที่มากกว่านี้ ก็จะเป็น Program memory ภายนอก ส่วน MCS-51 ที่ไม่มี Internal program memory ส่วนของ Program memory ทั้งหมดจะอยู่ภายนอก

สำหรับการอ่าน External program memory นั้น MCS-51 จะใช้ขาสัญญาณ PSEN (Program Store Enable)

สำหรับ Data memory ซึ่งสามารถที่จะอ่านหรือเขียนข้อมูลลงไปได้ ก็จะมี Address ที่แยกออกจาก Program memory และมีตำแหน่งของ External data memory ได้ทั้งหมด 64K Address ในการติดต่อกับ External data memory นั้น MCS-51 จะใช้ขาสัญญาณ RD และ WR



รูป 2.12 การต่อ External program memory และ External data memory ร่วมกัน

ในกรณีที่ต้องการ ส่วนของ External program memory และ External data memory ร่วมกันนั้น จะทำได้โดยการนำสัญญาณ PSEN และ RD มารวมกัน โดยใช้ AND gate ก็จะทำให้ได้สัญญาณที่เป็นการอ่าน External Program/Data memory ดังรูปที่ 2.4

รูปแสดงส่วนของ Program memory ในตำแหน่งเริ่มต้น ซึ่งเมื่อ CPU เริ่มการทำงานหลังจากการ Reset ก็จะเริ่มการทำงานตามคำสั่งที่ Address 0000h และสำหรับ Address ที่แสดงต่อมานั้น จะตำแหน่งที่อยู่ของ ISR: Interrupt Service Routine โดยการทำงานของ Interrupt ใน MCS-51 เมื่อเกิดสัญญาณ Interrupt เข้ามา มันก็จะกระโดดการทำงานมายังโปรแกรมในตำแหน่งที่กำหนดนี้ นั่นเองตัวอย่างเช่น สำหรับ Interrupt 0 จากภายนอก เมื่อ MCS-51 ได้รับสัญญาณ Interrupt นี้ มันก็จะกระโดดการทำงานมายังโปรแกรมใน Address 0003h และสำหรับ Interrupt ที่เกิดจาก Timer 0 ก็จะกระโดดการทำงานมายังโปรแกรมใน Address 000Bh และ Interrupt 0 จากภายนอก ก็จะกระโดดการทำงานมายังโปรแกรมใน Address 0013h . ในแต่ละ Address กำหนดให้สำหรับ ISR นั้นจะมีพื้นในการเก็บโปรแกรมได้ 8 Address ซึ่งถ้า ISR ที่ต้องการ เป็นโปรแกรมที่สั้นๆ ก็จะสามารถใส่เข้าไปได้ แต่ถ้าเป็น ISR ที่ยาวมากแล้วก็จะทำได้ โดยการใช้คำสั่ง Jump ไปยังโปรแกรมที่ต้องการอีกที

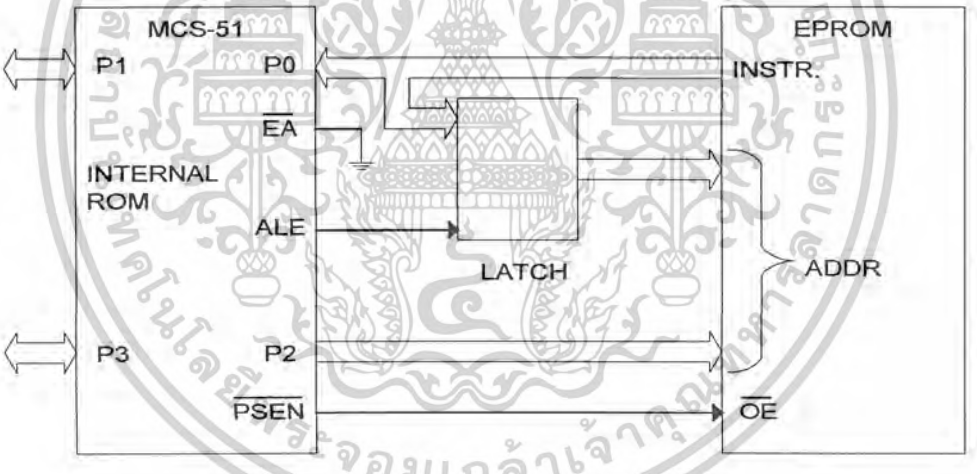
2.5.2.2 หน่วยความจำโปรแกรมภายนอก

สำหรับ MCS-51 ที่มี Internal program memory นั้น ผู้ใช้สามารถที่จะเลือกได้ว่า จะใช้งานของ Internal program memory นั้น หรือไม่ โดยการต่อของขาสัญญาณEA : External access เข้ากับ VCC หรือ GNDเช่นถ้า MCS-51 มี Internal program memory 4K (0000h-0FFFh) แล้วต่อขาสัญญาณ EA นี้เข้ากับ VCC การ Fetch คำสั่งที่ Address น้อยกว่า 0FFFh ก็จะได้จาก Internal program memory และถ้าเป็น Address ตั้งแต่ 1000h ก็ จะเป็นการอ่านจาก External program memory นั่นเอง แต่ถ้าต่อขา EA เข้ากับ GND การ Fetch คำสั่งทั้งหมดจะกระทำกับ External program memory

สำหรับ MCS-51 ที่ไม่มี Internal program memory แล้วขา EA จะต้องต่อ GND เสมอ

สัญญาณ PSEN ซึ่งเป็นสัญญาณที่ใช้ในการ Fetch คำสั่งจาก External program memory นั้น จะไม่ทำงานเมื่อเป็นการ Fetch คำสั่งจาก Internal program memory

การต่อ External program memory นั้นจะทำได้ดังรูปที่ 2.13



รูป 2.13 การต่อ External program memory

MCS-51 จะใช้ขาสัญญาณ ของ I/O port 16 bit (Port 0 และ Port 2) มาทำหน้าที่เป็น Bus ของระบบ โดยจะใช้งานของ Port 0 ทำหน้าที่เป็น Address และ Data bus สลับกัน คือส่งค่าของ Low byte ของ Program counter (PCL) ออกมาที่ Port 0 นี้ หลังจากส่งค่าของ PCL ออกมาแล้วมันจะเข้าสู่สภาวะ Float เพื่อรอรับคำสั่งที่จะอ่านได้จาก External program memory ในระหว่างที่ ค่าของ PCL ออกมาที่ P0 นี้ สัญญาณ ALE: Address Latch Enable จะไปทำให้ค่าของ PCL ถูกเก็บเข้าที่ LATCH และในเวลานั้น ค่าของ PCH ก็ถูกส่งออกมาที่ Port 2 เพื่อสร้างเป็น Address ขนาด 16 bit แล้ว สัญญาณ PSEN ก็จะเป็นตัวอ่านข้อมูลจาก Memory ที่ต้องการ

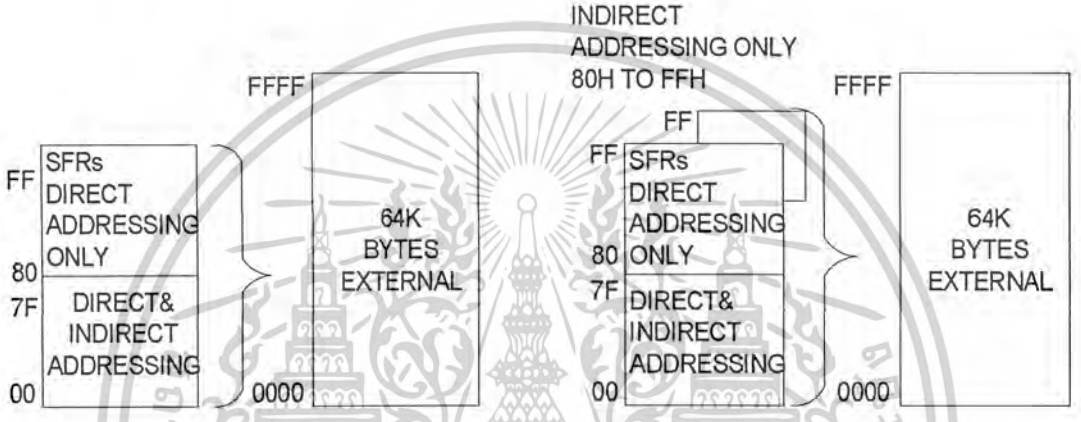
Data memory

Data memory ของ MSC-51 นั้นจะแบ่งออกเป็น Internal data memory และ External data memory โดยการใช้งานของ Data memory ทั้งสองส่วนนี้จะมี Address ที่แยกจากกันด้วย

MCS-51 with 128 bytes Internal Data memory (ซ้าย)

MCS-51 with 256 bytes Internal Data memory (ขวา)

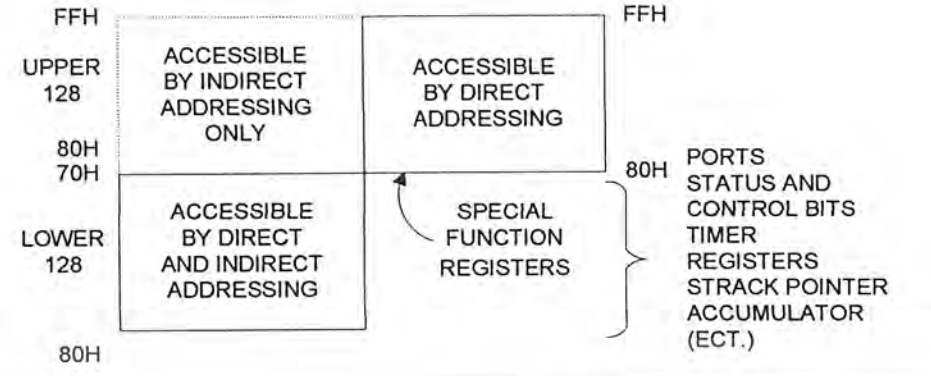
รายละเอียดดูดังรูปที่ 2.14



รูป 2.14 Data memory ของ MSC-51

2.5.2.3 หน่วยความจำข้อมูลภายใน

ผังการจัดแบ่งพื้นที่ของ Internal data memory จะเป็นดังรูป ซึ่งมันจะถูกแบ่งออกเป็น สามส่วนด้วยกัน คือ Lower 128, Upper 128 และ SFRและจากการที่ Internal data memory นั้นมีเพียง 256 ตำแหน่งเท่านั้น ทำให้การอ้าง Address สามารถทำได้ โดยใช้เพียง 8 Bit และจากการที่ Internal data memory นั้นมีเพียง 256 ตำแหน่งเท่านั้น ทำให้การอ้าง Address สามารถทำได้โดยใช้เพียง 8 Bit



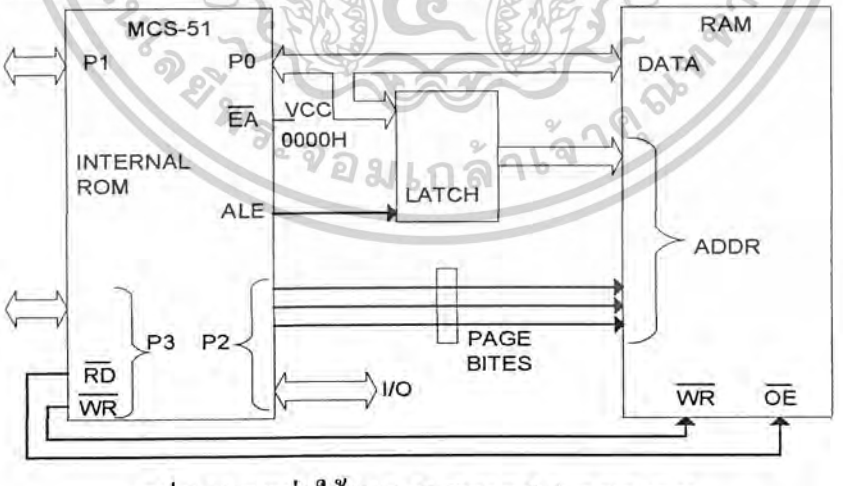
รูป 2.15 Internal Data Memory

แต่ด้วยเทคนิคของการอ้างถึงตำแหน่งข้อมูล (Addressing mode) ทำให้ได้ตำแหน่งของ Internal data memory ทั้งหมด 384 bytes

โดยสำหรับข้อมูลตั้งแต่ Address 80h – FFh ถ้าอ้างถึงข้อมูลที่ Address นั้นๆ ด้วยวิธีของ Direct addressing ก็จะได้ข้อมูลที่มาจากคนละส่วน กับการอ้างถึงข้อมูลที่ Address เดียวกันนั้น ด้วยวิธีของ Indirect addressing จากรูปจะเห็นได้ว่า Memory ในส่วนของ Upper 128 ก็จะมี Address เดียวกับ Memory ในส่วนของ SFR แต่จะใช้วิธีการเข้าถึงข้อมูลที่แตกต่างกันนั่นเอง

2.5.2.4 หน่วยความจำข้อมูลภายนอก

การต่อใช้งานของ External data memory จะทำได้ดังตัวอย่าง ซึ่งเป็นการต่อใช้งานของ RAM ขนาด 2K โดยให้ MCS-51 นี้ ทำงานจาก Internal program memory



รูป 2.16 การต่อใช้งานของ External data memory

อย่างไรก็ตามจังหวะเวลาของการอ่านข้อมูลจาก External data memory นั้นจะใช้เวลาที่มากกว่าการอ่านข้อมูลจาก Program memory

2.5.3 **ชุดคำสั่งของ MCS -51**

คำสั่งของ 8051 จะสามารถแบ่งออกเป็นชุดๆตามประเภทการทำงานได้ดังนี้

Arithmetic Instructions

คำสั่งที่เป็นการทำงานทาง Math ที่มี จะแสดงดังตาราง และแสดง Addressing mode ที่สามารถนำมาใช้เป็น <byte> Operand ได้สำหรับแต่ละคำสั่ง

MNEMONIC	OPERATION	ADDRESSING MODES				EXECUTION TIME (µs)
		Dir	Ind	Reg	Imm	
ADD A, <byte>	A = A + <byte>	X	X	X	X	
ADDC A, <byte>	A = A + <byte> + C	X	X	X	X	1
SUBB A, <byte>	A = A – <byte> – C	X	X	X	X	1
INC A	A = A + 1	Accumulator only				1
INC <byte>	<byte> = <byte> + 1	X	X	X		1
INC DPTR	DPTR = DPTR + 1	Data Pointer only				2
DEC A	A = A – 1	Accumulator only				1
DEC <byte>	<byte> = <byte> – 1	X	X	X		1
MUL AB	B:A = B × A	ACC and B only				4
DIV AB	A = Int [A/B] B = Mod [A/B]	ACC and B only				4
DA A	Decimal Adjust	Accumulator only				1

Table 2 : A list of the MHS C51 Arithmetic Instructions.

รูป 2.17 Arithmetic Instructions

เช่น ADD A, <byte> ก็จะใช้ได้คือ

ADD	A, 7FH	(direct	addressing)
ADD	A, @R0	(indirect	addressing)
ADD	A, R7	(register	addressing)
ADD	A, #127	(immediate constant)	

สำหรับค่าเวลาที่ใช้ในการทำงานที่แสดงนั้น สมมุติให้ CPU ทำงานที่ Clock 12 MHz

2.5.3.1 **คำสั่งทางลอจิก**

MNEMONIC	OPERATION	ADDRESSING MODES				EXECUTION TIME (μs)
		Dir	Ind	Reg	Imm	
ANL A, <byte>	A = A AND <byte>	X	X	X	X	1
ANL <byte>, A	<byte> = <byte> AND A	X				1
ANL <byte>, # data	<byte> = <byte> AND # data	X				2
ORL A, <byte>	A = A OR <byte>	X	X	X	X	1
ORL <byte>, A	<byte> = <byte> OR A	X				1
ORL <byte>, # data	<byte> = <byte> OR # data	X				2
XRL A, <byte>	A = A XOR <byte>	X	X	X	X	1
XRL <byte>, A	<byte> = <byte> XOR A	X				1
XRL <byte>, # data	<byte> = <byte> XOR # data	X				2
CLR A	A = 00H	Accumulator only				1
CLP A	A = NOT A	Accumulator only				1
RL A	Rotate ACC Left 1 bit	Accumulator only				1
RLC A	Rotate Left through Carry	Accumulator only				1
RR A	Rotate ACC Right 1 bit	Accumulator only				1
RRC A	Rotate Right through Carry	Accumulator only				1
SWAP A	Swap Nibbles in A	Accumulator only				1

Table 3 : A list of the MHS C51 Logical Instructions.

รูป 2.18 Logical Instructions

2.5.3.2 คำสั่งเคลื่อนย้ายข้อมูลภายใน

MNEMONIC	OPERATION	ADDRESSING MODES				EXECUTION TIME (μs)
		Dir	Ind	Reg	Imm	
MOV A, <src>	A = <src>	X	X	X	X	1
MOV <dest>, A	<dest> = A	X	X	X		1
MOV <dest>, <src>	<dest> = <src>	X	X	X	X	2
MOV DPTR, # data 16	DPTR = 16-bit immediate constant				X	2
PUSH <src>	INC SP : MOV"@SP", <scr>	X				2
POP <dest>	MOV <dest>, "@SP" : DEC SP	X				2
XCH A, <byte>	ACC and <byte> Exchange Data	X	X	X		1
XCHD A, @Ri	ACC and @ Ri exchange low nibbles		X			1

Table 4 : A list of the MHS C51 Data Transfer Instructions that Access Internal Data Memory Space.

รูป 2.19 Data Transfers ของ Internal RAM

2.5.3.3 คำสั่งเคลื่อนย้ายข้อมูลภายนอก

ADDRESS WIDTH	MNEMONIC	OPERATION	EXECUTION TIME (μs)
8 bits	MOVX A, @ Ri	Read external RAM @ Ri	2
8 bits	MOVX @ Ri, A	Write external RAM @ Ri	2
16 bits	MOVX A, @ DPTR	Read external RAM @ DPTR	2
16 bits	MOVX @ DPTR, A	Write external RAM @ DPTR	2

Table 5 : A list of the MHS C51 Data Transfer Instructions that Access External Data Memory Space.

รูป 2.20 Data Transfers ของ External RAM

2.5.3.4 คำสั่งการเปิดตาราง

MNEMONIC	OPERATION	EXECUTION TIME (µs)
MOVC A, @A + DPTR	Read Pgm Memory at (A + DPTR)	2
MOVC A, @A + PC	Read Pgm Memory at (A + PC)	2

Table 6 : The MHS C51 Lookup Table Read Instructions.

รูป2.21 Lookup table

2.5.3.5 คำสั่งทางบูลีน

MNEMONIC	OPERATION	EXECUTION TIME (µs)
ANL C,bit	C = C AND bit	2
ANL C,/bit	C = C AND (NOT bit)	2
ORL C,bit	C = C OR bit	2
ORL C,/bit	C = C OR (NOT bit)	2
MOV C,bit	C = bit	1
MOV bit,C	bit = C	2
CLR C	C = 0	1
CLR bit	bit = 0	1
SETB C	C = 1	1
SETB bit	bit = 1	1
CPL C	C = NOT C	1
CPL bit	bit = NOT bit	1
JC rel	Jump if C = 1	2
JNC rel	Jump if C = 0	2
JB bit,rel	Jump if bit = 1	2
JNB bit,rel	Jump if bit = 0	2
JBC bit,rel	Jump if bit = 1 ; CLR bit	2

Table 7 : A list of the MHS C51 Boolean Instructions.

รูป 2.22 Boolean Instructions

2.5.3.6 คำสั่งการกระโดด

MNEMONIC	OPERATION	EXECUTION TIME (µs)
JMP addr	Jump to addr	2
JMP @A + DPTR	Jump to A + DPTR	2
CALL addr	Call subroutine at addr	2
RET	Return from subroutine	2
RETI	Return from interrupt	2
NOP	No operation	1

Table 8 : Unconditional Jumps in MHS C51.

รูป 2.23 Jump Instructions

MNEMONIC	OPERATION	ADDRESSING MODES				EXECUTION TIME (μs)
		DIR	IND	REG	IMM	
JZ rel	Jump if A = 0	Accumulator only				2
JNZ rel	Jump if A ≠ 0	Accumulator only				2
DJNZ <byte>,rel	Decrement and jump if not zero	X		X		2
CJNZ A,<byte>,rel	Jump if A = <byte>	X			X	2
CJNE <byte>,#data,rel	Jump if <byte> ≠ #data		X	X		2

Table 9 : Conditional Jumps in MHS C51 Devices.

รูป 2.24 ตารางสถานการณ์กระโดด

2.5.4TIMER และ COUNTER

จากส่วนประกอบภายในของ MSC-51 ซึ่งมี Timer/Counters ให้นั้น สำหรับ 80C51 จะมี 16 bit Timer/Counter register ให้ 2 ชุด (คือ Timer 0 และ Timer 1) และสำหรับ 80C52, 83C154 และ 83C154D จะมีเพิ่มขึ้นอีก 1 ตัว (คือ Timer 2) ซึ่งทั้งหมดนี้สามารถที่จะใช้งานเป็น Timer หรือ Even counter ก็ได้

เมื่อใช้งานเป็น Timer ค่าใน Register จะเพิ่มขึ้นเองทุกๆ Machine cycle จากที่ 1 Machine cycle ใช้สัญญาณ Clock 12 ลูก ดังนั้น ค่าของ Timer จะนับขึ้นด้วยความเร็ว 1/12 ของความถี่ Clock ที่ใช้

เมื่อใช้งานเป็น Counter ค่าใน Register จะมีค่าเพิ่มขึ้นทุกครั้งที่มีการเปลี่ยนแปลงสถานะ 0 – 1 ของสัญญาณจากภายนอกที่ขา Input ที่สัมพันธ์กัน (T0, T1 และ T2) ในการทำงานของ Counter นี้ สัญญาณจากภายนอก จะถูกตรวจสอบในช่วงเวลาของ S5P2 ของทุกๆ Machine cycle เมื่อมันตรวจพบสถานะที่เป็น 1 ใน Cycle ใดๆ แล้วตรวจพบสถานะที่เป็น 0 ใน Cycle ถัดมา ค่าของ Counter ก็จะเพิ่มขึ้น โดยค่าของ Register จะมีการเปลี่ยนแปลงในช่วงเวลาของ S3P1 ของ Cycle ต่อไป จากที่การทำงานของ Counter ที่ตรวจสอบสัญญาณ ใน 2 Cycle ก็จะทำให้ได้ความถี่สูงสุดที่จะนับได้คือ 1/24 ของสัญญาณ Clock ที่ใช้

Timer 0 and Timer 1

Timer/Counter 0 และ Timer/Counter 1 ที่อยู่ใน MCS-51 จะสามารถกำหนดรูปแบบการใช้งานได้จาก Bit C/~T ของ Register TMOD ใน SFR และ Mode การทำงานจะเลือกได้จาก Bit M0, M1 ซึ่งเลือกได้ทั้งหมด 4 Mode ด้วยกัน การทำงานของ Timer/Counter ใน Mode 0, 1 และ 2 ของ Timer 0 และ Timer 1 จะเหมือนกัน ส่วนการทำงานใน Mode 3 จะแตกต่างกัน โดยสรุปได้ดังนี้ล

2.5.4.1 **รีจิสเตอร์(Register TMOD)**

GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
TIMER 1				TIMER 0			

GATE	When TRx (in TCON) is set and GATE = 1, TIMER/COUNTERx will run only while INTx pin is high (hardware control). When GATE = 0, TIMER/COUNTERx will run only while TRx = 1 (software control).
C/$\overline{T}$	Timer or Counter selector. Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from Tx input pin).
M1	Mode selector bit (NOTE 1).
M0	Mode selector bit (NOTE 1).

NOTE 1 :

M1	M0	OPERATING MODE	
0	0	0	13-bit Timer
0	1	1	16-bit Timer/Counter
1	0	2	8-bit Auto-Reload Timer/Counter
1	1	3	(Timer 0) TL0 is an 8-bit Timer/Counter controlled by the standard Timer 0 control bits, TH0 is an 8-bit Timer and is controlled by Timer 1 control bits.
1	1	3	(Timer 1) Timer/Counter 1 stopped.

2.5.4.2 **บิทควบคุม ไทม์เมอร์ (GATE BIT)**

เป็น Bit ที่ใช้ควบคุมให้ Timer ทำงานหรือไม่ ถ้า Bit นี้ของ Timer x ถูกตั้งเป็น 1 (Hardware controlled) จะทำให้ Timer ทำงานก็ต่อเมื่อสัญญาณที่ขา INTx เป็น 1 และ Bit TRx ใน Register TCON เป็น 1 ด้วย 0 (Software controlled) จะทำให้ Timer ทำงานเมื่อ Bit TRx ใน Register TCON เป็น

C/~T Bit

สำหรับเลือกการทำงานของ Timer/Counter ว่าจะใช้เป็น Timer หรือ Counter ถ้า Bit นี้เป็น 1 (Counter) เลือกการทำงานเป็น Counter ซึ่งก็จะทำการนับสัญญาณที่เข้ามาที่ ขา Tx 0 (Timer) เลือกการทำงานเป็น Timer ซึ่งก็จะทำการนับสัญญาณ Clock/1

2.6 สเต็ปเปอร์มอเตอร์

สเต็ปเปอร์มอเตอร์ได้รับการพัฒนามาอย่างต่อเนื่อง จนในปัจจุบันสเต็ปเปอร์มอเตอร์ที่นิยมใช้อย่างแพร่หลายมากที่สุดคือสเต็ปเปอร์มอเตอร์แบบยูนิโพล่าร์ (UNI-POLAR STEPPER MORTOR) มีลักษณะการพันขดลวดของมอเตอร์ดังรูปที่ 2.25

แบบ 5 สาย 4 เฟส แบบ 6 สาย 4 เฟส

รูป 2.25 โครงสร้างอย่างง่ายของสเต็ปเปอร์มอเตอร์แบบยูนิโพล่าร์

สเต็ปเปอร์มอเตอร์แบบนี้มีการพันขดลวดสองขดในแต่ละขั้วแม่เหล็กของสเตเตอร์แต่ละขดแบ่งเป็นสองเฟสรวมมอเตอร์ทั้งตัวจะมี 4 เฟสคือ เฟส 1 , 2 ,3 , 4 มีการต่อสายออกมาจากขดลวดแต่ละขดเพื่อจ่ายไฟเลี้ยง ทำให้สเต็ปเปอร์มอเตอร์แบบนี้มีทั้งแบบ 5 สาย และ 6 สาย ถ้าเป็นแบบ 5 สายจะเป็นการนำขดลวดไฟเลี้ยงของขดทั้งสองมาต่อรวมกันเป็นสายเดียว

2.6.1 การกระตุ้นและควบคุมการหมุนของสเต็ปเปอร์มอเตอร์

การกระตุ้นและควบคุมการหมุนของมอเตอร์ให้เคลื่อนที่ไปแต่ละสเต็ปทำได้โดยจ่ายกำลังไฟฟ้าไปยังขดลวดแต่ละขดบนสเตเตอร์ ซึ่งต้องป้อนแบบซีเควนเชียลในรูปแบบที่ถูกต้องด้วย สามารถแบ่งได้เป็น 3 รูปแบบคือ แบบฟูลสเต็ปหนึ่งเฟส , แบบฟูลสเต็ป 2 เฟส และแบบฮาร์ฟสเต็ป (HALF STEP)

แบบฟูลสเต็ปหนึ่งเฟส (full step) เป็นการกระตุ้นที่มีรูปแบบง่ายที่สุด โดยทำการกระตุ้นขดลวดทีละขดในเวลาหนึ่งใล่เรียงถัดกันไป เช่นเริ่มต้นที่ขด 1 , 2 , 3 , 4 แล้ววนกลับมาที่ 1 ใหม่ วนไปเรื่อยๆ หรือเริ่มที่ขดหนึ่งแล้ววนมาที่ขด 4 ,3 , 2 แล้ววนกลับมายังขดที่1 อีกครั้ง ซึ่งทำให้ทิศทางของการหมุนสวนกันการกระตุ้นรูปแบบนี้จึงมีขดลวดเพียงขดเดียว ในเวลาหนึ่งที่ถูกกระตุ้นเท่านั้นวงจรกระตุ้นจึงมีราคาถูกและง่าย ขั้นตอนการทำงานแสดงดังตารางที่ 1

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	-	ทำงาน	-	-
3	-	-	ทำงาน	-
4	-	-	-	ทำงาน

ตาราง 1 แสดงลำดับการทำงานของขดลวดในแต่ละเฟสของมอเตอร์เมื่อได้รับการกระตุ้นแบบฟูลสเต็ปหนึ่งเฟส

แบบฟูลสเต็ปสองเฟสเป็นการกระตุ้นซึ่งคล้ายกับแบบหนึ่งเฟส แต่การกระตุ้นแบบนี้จะทำการกระตุ้นโดยจ่ายกำลังไฟฟ้าไปที่ขดลวดสองขด ที่อยู่ใกล้กันในเวลาเดียวกันและเรียงถัดกันไปเช่นเดียวกับแบบหนึ่งเฟสดังตัวอย่าง ขดลวดชุดแรกที่ถูกกระตุ้นจะเป็นขดที่ 1 และ 2 ตามด้วยการกระตุ้นขดที่ 2 และ 3 ต่อไปเป็นขดที่ 3 และ 4 จากนั้นจึงเป็นขดที่ 4 และ 1 และกลับมาที่ขด 1 , 2 วนไปตามลำดับเช่นนี้ และเช่นเดียวกับแบบฟูลสเต็ปหนึ่งเฟส ถ้าหาจ่ายไฟวนกลับทางก็จะทำให้ทิศการหมุนของมอเตอร์กลับทางเช่นกัน การกระตุ้นสเต็ปเปอร์มอเตอร์แบบนี้สามารถเพิ่มแรงบิดได้มากกว่าแบบเวฟ โรเตอร์จะเคลื่อนที่ด้วยแรงดึงอย่างเต็มแรงจากสองขดลวดที่ถูกกระตุ้นพร้อมกัน และต่อไปด้วยแรงดึงจากอีกสองขดลวดถัดไป แต่ก็มีข้อเสียคือต้องใช้กำลังไฟฟ้ามากขึ้นขั้นตอนการทำงานแสดงดังตารางที่ 2

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	ทำงาน	-	-
2	-	ทำงาน	ทำงาน	-
3	-	-	ทำงาน	ทำงาน
4	ทำงาน	-	-	ทำงาน

ตาราง 2 แสดงลำดับการทำงานของขดลวดในแต่ละเฟสของมอเตอร์

แบบฮาล์ฟสเต็ปเป็นรูปแบบที่ผสมผสานระหว่างการกระตุ้นแบบฟูลสเต็ปหนึ่งเฟส และสองเฟส เพื่อเพิ่มจำนวนสเต็ปต่อรอบอีกเท่าตัว ในระบบนี้จะทำการกระตุ้นขดลวดเรียงกันไปเป็นลำดับดังนี้เริ่มจากขดลวดที่ 1 , 1 และ 2, 2 , 2 และ 3,3 ,3 และ 4, 4 , 4 แล้ววนกลับมายังขดที่ 1 แรงบิดที่ได้จากการกระตุ้นแบบนี้จะเพิ่มมากขึ้นอีก เพราะช่วงสเต็ปมีระยะสั้นลง แต่ละสเต็ปเกิดแรงดึงจากขดลวดสองขดที่ถูกกระตุ้นพร้อมกัน ความถูกต้องของตำแหน่งมีเพิ่มมากขึ้น แต่ต้องพึงระวังอีกประการหนึ่งว่าเมื่อการะตุ้นให้ทำงานในรูปแบบนี้จะต้องทำการหมุนถึงสองสเต็ปจึงจะได้ระยะเท่ากับหนึ่งสเต็ปของการควบคุมในสองแบบแรก สำหรับขนาดของกำลังไฟฟ้าต้องใช้ขนาดเท่ากับแบสองเฟสเป็นอย่างน้อยจึงจะเพียงพอขั้นตอนการทำงานต่างๆแสดงดังตารางที่ 3

สเต็ปที่	เฟสที่ 1	เฟสที่ 2	เฟสที่ 3	เฟสที่ 4
1	ทำงาน	-	-	-
2	ทำงาน	ทำงาน	-	-
3	-	ทำงาน		-
4	-	ทำงาน	ทำงาน	-
5	-	-	ทำงาน	-
6	-	-	ทำงาน	ทำงาน
7	-	-	-	ทำงาน
8	ทำงาน	-	-	ทำงาน

ตาราง 3 แสดงลำดับการทำงานของขดลวดในแต่ละเฟสของมอเตอร์ เมื่อได้รับการกระตุ้นแบบฮาล์ฟสเต็ป

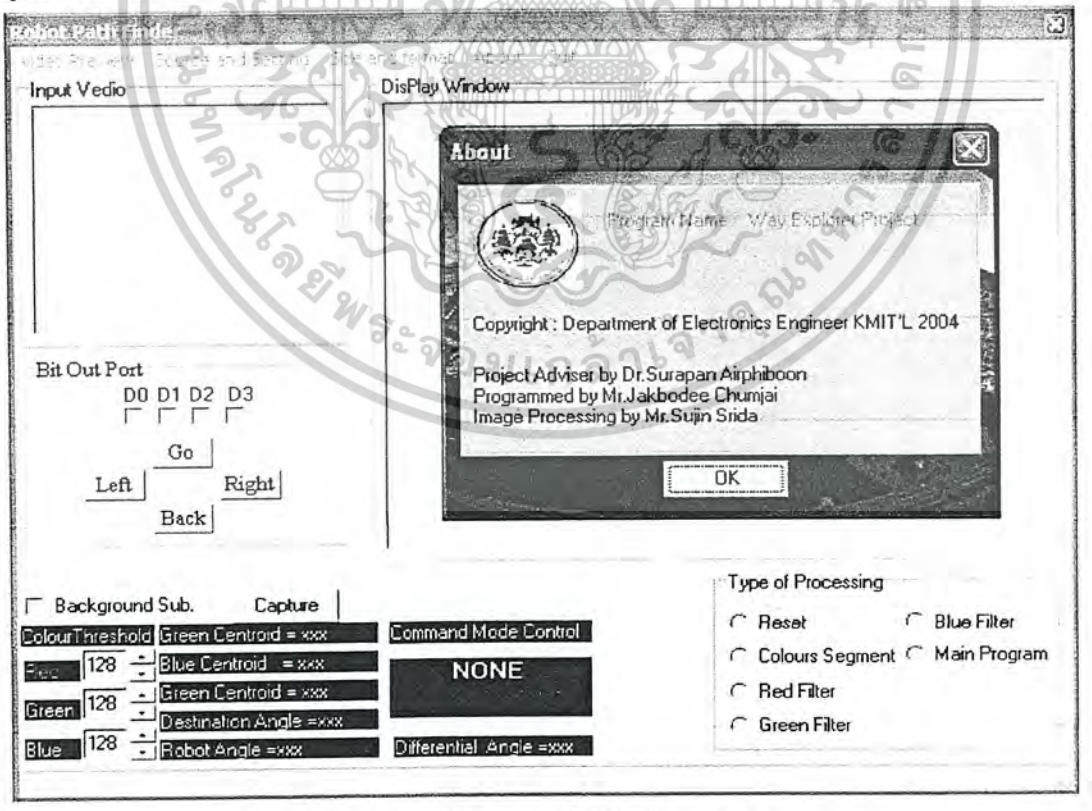
บทที่ 3
การออกแบบและการสร้าง

3.1 แนวคิดในการออกแบบโครงงาน

แนวความคิดการออกแบบโครงงานชุดนี้จะแยกออกเป็น 3 ส่วนใหญ่ๆด้วยกันคือ ส่วนแรกคือส่วนของโปรแกรมที่จะถูกใช้ภายในเครื่องคอมพิวเตอร์ ส่วนที่สอง คือส่วนของตัวหุ่นยนต์ และส่วนที่สาม คือสนามที่จะถูกใช้เป็นที่ทดสอบโครงงานโดยจะสามารถแยกอธิบายทั้งสามส่วนได้ดังนี้

3.2 โปรแกรมภายในเครื่องคอมพิวเตอร์

โปรแกรมที่ใช้งานเพื่อทำการติดต่อและควบคุมตัวหุ่นนั้นเป็นโปรแกรมในลักษณะของกราฟฟิกส์ ยูสเซอร์ อินเตอร์เฟสซ์(Graphics User Interfaces) คือโปรแกรมที่ใช้การแสดงผลแบบรูปภาพเพื่อให้ผู้ใช้งานสามารถทำงานได้โดยไม่ต้องยุ่งกับคำสั่งจำนวนมาก และโปรแกรมที่ใช้ในการสร้างโครงงานนี้คือ บอแลน ซี++ บิลด์เดอร์ 6.0 (BCB 6.0) ซึ่งเป็นโปรแกรมที่ใช้ภาษา ซี เป็นภาษาพื้นฐานในการเขียนโปรแกรม ซึ่งหน้าตาของโปรแกรมที่ได้สร้างมาแล้วนั้นมีรูปร่างหน้าตาตามรปที่3.1ดังนี้



รูป 3.1 แสดงโปรแกรมที่สร้างด้วย BCB 6.0

ส่วนที่เป็นหัวใจหลักของโครงงานนี้คือการประมวลผลภาพหรือ(Image Processing)ซึ่งจะเป็นโปรแกรมสำหรับใช้ในการประมวลสัญญาณภาพที่รับเข้ามาจากกล้องวีดิโอ แล้วจะนำผลที่ได้จากกระบวนการไปทำการควบคุมหุ่นยนต์โดยผ่านทางพอร์ตขนาน (Parallel Port) ของเครื่องคอมพิวเตอร์อีกต่อหนึ่งโดยจะมีวิธีการในออกแบบโปรแกรมดังนี้

3.2.1 การรับภาพจากกล้องวีดิโอ(Image Acquisition)

ก่อนอื่นขออธิบายก่อนว่า โปรแกรม ซี++ บิวล์เดอร์ คือซอฟต์แวร์หรือเครื่องมือที่ใช้สำหรับเพื่อเขียนหรือสร้างแอพพลิเคชัน ซอฟต์แวร์ ซี++ บิวล์เดอร์ นั้นจัดเป็นเครื่องมือเขียนโปรแกรมชนิด วิชัวล์ โปรแกรมมิ่ง(Visual Programming) เช่นเดียวกับโปรแกรม เดลฟาย , วิชัวล์ เบสิก , วิชัวล์ ซี++ เป็นต้น

ซึ่งในการเขียนโปรแกรมจะมีเครื่องมือพื้นฐานเตรียมให้ใช้งานได้อย่างสะดวกเรียกว่า วิชัวล์คอมโพเนนท์ไลบรารี่ VCL (Visual Component Library) และภายในโครงงานก็ได้มีการใช้ VCL ที่มีชื่อว่า ImageEn.[4] ซึ่งนอกจากจะใช้เป็นเครื่องมือในการรับสัญญาณภาพจากกล้องวีดิโอแล้วยังมีคำสั่งที่รองรับการประมวลผลภาพให้อย่างมาก

3.2.2 การควบคุมพอร์ตขนาน(Parallel Port Controlling)

ข้อจำกัดของ ซี++บิวล์เดอร์ ที่ทำงานบนระบบปฏิบัติการวินโดวส์ เอ็กซ์พี และ เอ็นที ก็คือไม่สามารติดต่อสื่อสารกับพอร์ตของเครื่องคอมพิวเตอร์โดยใช้ภาษาเอสเซมบลี่ได้โดยตรงอีกแล้ว (แต่ไม่มีปัญหากับระบบปฏิบัติการวิโดวส์ 95-98) จะต้องใช้การติดต่อกับพอร์ตที่มีอยู่ทางอ้อมแทนโดยติดต่อผ่านซอร์ฟแวร์ไดรเวอร์แทน

ดังนั้นวิธีการที่จะสามารถติดสื่อสารกับพอร์ตต่างๆของเครื่องคอมพิวเตอร์บนระบบปฏิบัติการดังกล่าว นั้นก็ต้องใช้ Non-VCL(มองไม่เห็นขณะทำงาน)ที่มีชื่อว่า TvicHW32[5] เป็นคอมโพเนนต์ที่จะทำหน้าที่ติดต่อกับพอร์ตของเครื่องคอมพิวเตอร์ที่ต้องการได้

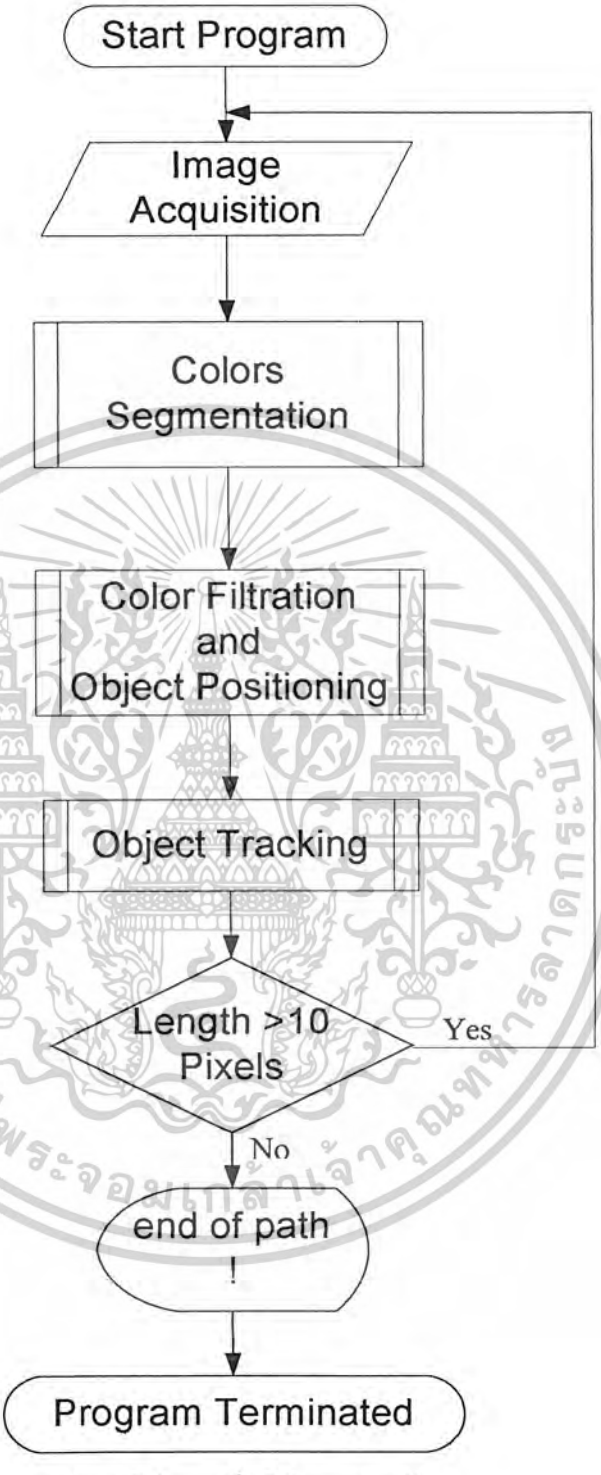
3.2.3 การทำงานของโปรแกรม

ในการเขียนโปรแกรมจะทำการแยกส่วนของโปรแกรมที่ถูกเรียกใช้งานบ่อยๆให้อยู่ในรูปของโปรแกรมย่อยหรือ ซับรูทีน(Sub Routine) เพื่อให้โปรแกรมสามารถแก้ไขได้สะดวกเมื่อเกิดปัญหาขึ้นโดยจะแสดงส่วนของโปรแกรมหลักได้ดังรูปโฟลวชาร์ตต่อไปนี้

---

4 สามารถดาวน์โหลดมาใช้ได้ฟรีจาก www.hicomponents.com / หรือหาได้จาซีดี-รอมของรายงานเล่มนี้

5 http://www.entechtaiwan.com/tools.htm / เป็นรุ่นทดลองใช้30วัน



รูป 3.2 โฟลวชาร์ต โปรแกรมหลัก

Image Acquisition

การทำงาน เริ่มโปรแกรมจะทำการรับภาพจากกล้องวีดิโอ ผ่านคอมโพเนนต์ชื่อ ImageEn. เพื่อนำมาแสดงผลที่หน้าจอก่อนซึ่งจะตั้งความลึกของสีไว้ที่ 24 บิต และความละเอียดของภาพตั้งไว้ที่160 X 120 Pixels และต่อจากนั้นภาพที่ได้จะถูกนำไปทำ Colors Segmentation

Colors Segmentation

เพื่อปรับภาพที่ได้ให้เหมาะสมก่อนนำไปคัดแยกสี เหตุที่ต้องทำก็เพราะในภาพที่รับเข้ามาจากกล้องวีดิโอนั้น ยังมีปัญหาจากแสงที่ไม่เสม่ำเสมอ ที่มาตกกระทบกับวัตถุทำให้สีของวัตถุที่กล้องจับได้นั้นเกิดการผิดเพี้ยน เช่นความมืดของสีที่บริเวณขอบของภาพ สีของวัตถุที่ไม่เท่ากัน เป็นต้น ดังนั้นการทำเซ็กเมนต์ดังกล่าวก็เพื่อให้เกิดความชัดเจนของสีทั้งหมดออกมาเท่ากัน ตลอกทั้งภาพ(มีความสว่างเท่ากันทั้งภาพ)

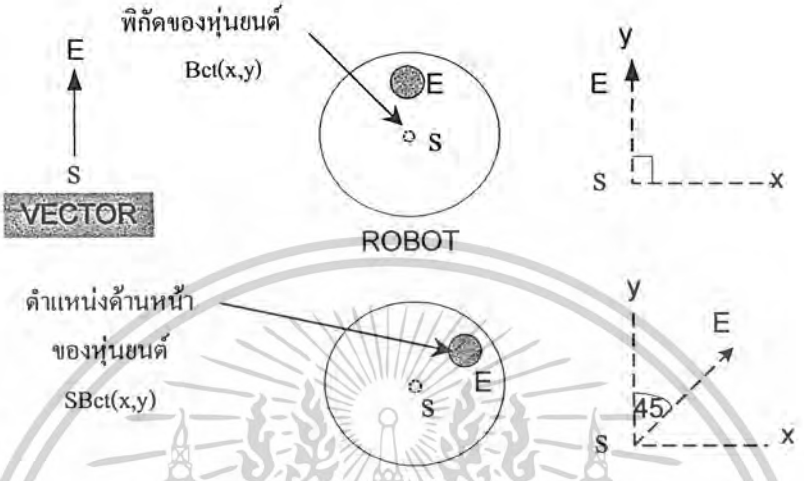
Color Filtration and Object Positioning

ต่อไปหลังจากเราได้ภาพที่เหมาะสมที่จะใช้ในการคัดแยกสีของวัตถุแล้ว เราก็จะทำการกรองสีในส่วนของโปรแกรมย่อย ซึ่งในส่วนนี้ก็จะมีโปรแกรมย่อยอยู่ 3 ส่วนด้วยกันคือ โปรแกรมกรองวัตถุสีแดง โปรแกรมกรองวัตถุสีเขียว และสุดท้าย คือโปรแกรมกรองวัตถุสีน้ำเงิน ซึ่งหลังจาได้ทำการกรองเอาสีทั้งสามออกจากันแล้ว จากนั้นจึงทำการหาตำแหน่งของวัตถุในภาพว่าอยู่ ณ ตำแหน่งในของภาพโดยใช้การอ้างอิงกับจำนวนของพิเซลในภาพนั้นโดยจะหาตำแหน่งเพียง สองสีเท่านั้นคือ สีน้ำเงิน และสี เขียว

Object Tracking

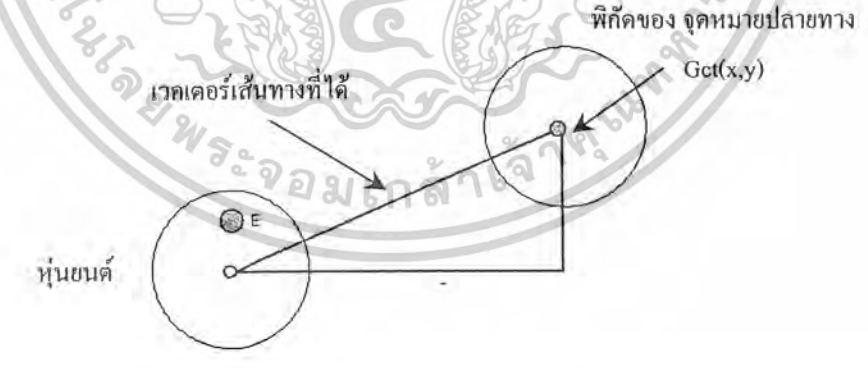
ต่อไปเป็นโปรแกรมสำหรับการติดตามวัตถุ ซึ่งวัตถุที่จะถูกติดตามก็คือสีเขียวและสีน้ำเงินโดยจะทำการรับค่าตำแหน่งของวัตถุทั้งสองจากโปรแกรมย่อยก่อนหน้านี้มาทำการหาเวคเตอร์จากจุดพิกัดของหุ่นยนต์และพิกัดของจุดหมายปลายทาง(หาความชันนั้นเอง)ซึ่งเวคเตอร์นี้จะทำหน้าที่ในการบอกทิศทางของจุดหมายที่หุ่นยนต์จะต้องเดินไปหา ต่อไป แล้วจารู้ได้ไงว่าหุ่นยนต์หันหน้าไปทางไหน? คำตอบนั้นก็คือ ที่ตำแหน่งด้านหน้าของหุ่นยนต์จะมีการติดวงกลมสีเขียวขนาดเล็กเอาไว้เพื่อเป็นตัวบอกพิกัดของตำแหน่งของหน้ารถดังนั้นเมื่อนำพิกัดดังกล่าวมาคิดรวมกับตำแหน่งตัวหุ่นยนต์แล้วหาเวคเตอร์ทิศทางของตัวหุ่นมาอีกหนึ่งเส้นแสดงดังรูปที่ 3.4 หลังจากที่ได้ทำการหาทิศทางของตัวหุ่นยนต์ได้แล้วลำดับต่อไปก็จะทำการสั่งให้หุ่นยนต์เลี้ยวหรือหันหน้าไปในทิศทางเดียวกันกับ พาทเวคเตอร์ (หันไปทิศทางเดียวกับจุด

ปลายทาง) ซึ่งสามารถทำได้โดยการวัดค่ามุมของเวคเตอร์ของทั้งตัวหุ่นและมุมของพาทเวคเตอร์หลังจากนั้นก็หามุมผลต่างของเวคเตอร์ทั้งสองโดยใช้มุมของพาทเวคเตอร์เป็นมุมอ้างอิง



รูป 3.4 แสดงทิศทางการหมุนตัวของหุ่นยนต์

ถ้าผลต่างติดลบแสดงว่ามุมของหุ่นยนต์นั้นมากกว่ามุมของพาทเวคเตอร์ ดังนั้นจึงทำการสั่งให้หนุ่ยนต์หมุนไปทางขวาเข้าหาพาทเวคเตอร์ ซึ่งระหว่างนั้นก็จะตรวจวัดว่า มุมที่ได้เข้าใกล้ ค่า 0 องศาหรือยังทำแบบนี้ไปจนกว่าหุ่นจะหันไปในทิศทางที่ถูกต้องแล้วก็ให้หุ่นหยุดรอคำสั่งต่อไป

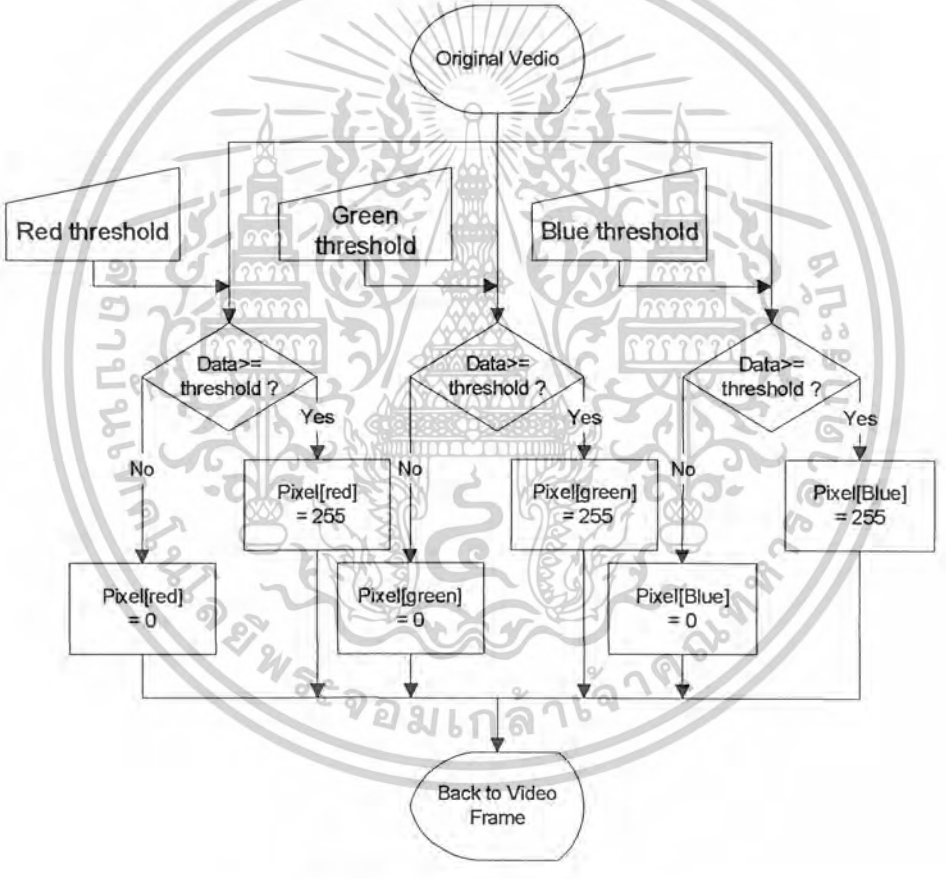


รูป 3.5 แสดงการหาเวคเตอร์เส้นทางและเวคเตอร์ของตัวหุ่น

โปรแกรมส่วนการตัดสินใจ

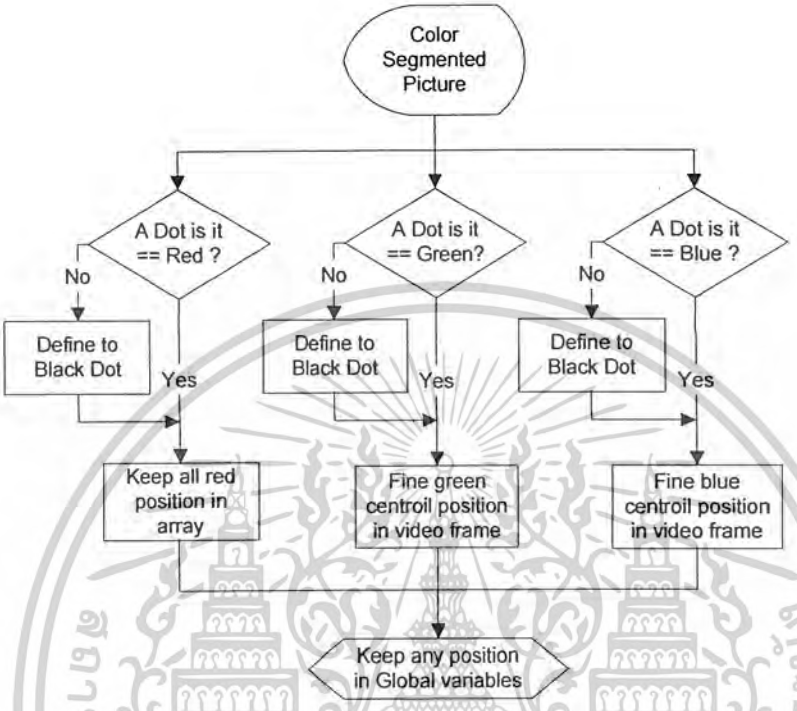
จะเป็นการการเปรียบเทียบระยะทางของพาทเวคเตอร์ (Path Vector) ว่ามีค่าเท่ากับที่กำหนดไว้หรือไม่ โดยที่สามารถหาได้จากสมการ ระยะห่างระหว่างจุด Bct(x,y) และ Gct(x,y) ได้จากสามการ $\text{Length} = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$ ....Pixels ซึ่งถ้าความยาวที่คำนวณได้ตรงตามเงื่อนไขโปรแกรมก็จะสั่งให้รถหยุดทำงานพร้อมกับแสดงข้อความ the end of path .

3.2.3.1 **Flowchart Colors Segmentation**



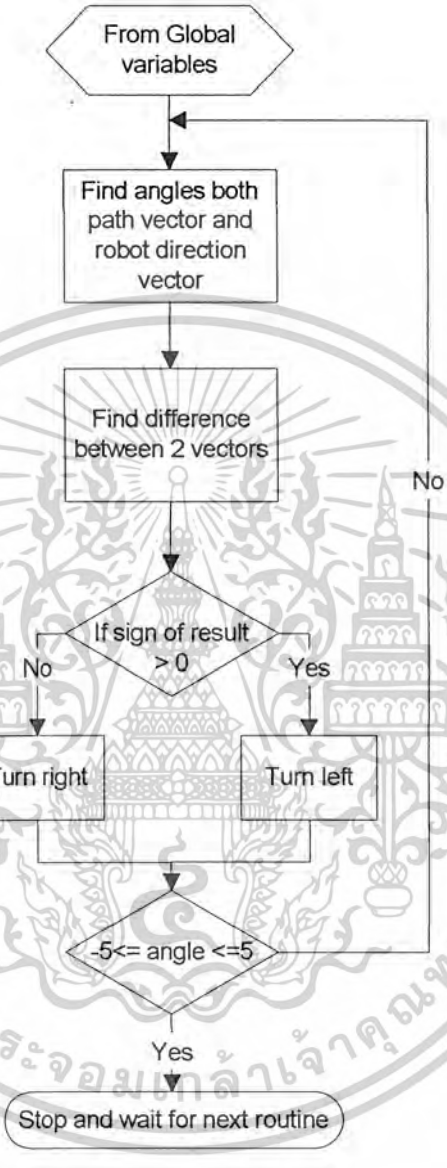
รูป 3.6 โฟลวชาร์ตโปรแกรมแยกส่วนสี

3.2.3.2 **Flowchart Colors Filtration and object positioning**



รูป 3.7 โฟลวชาร์ตโปรแกรมกรองสีและหาตำแหน่งวัตถุ

3.2.3.3 **Flowchart Object Tracking**

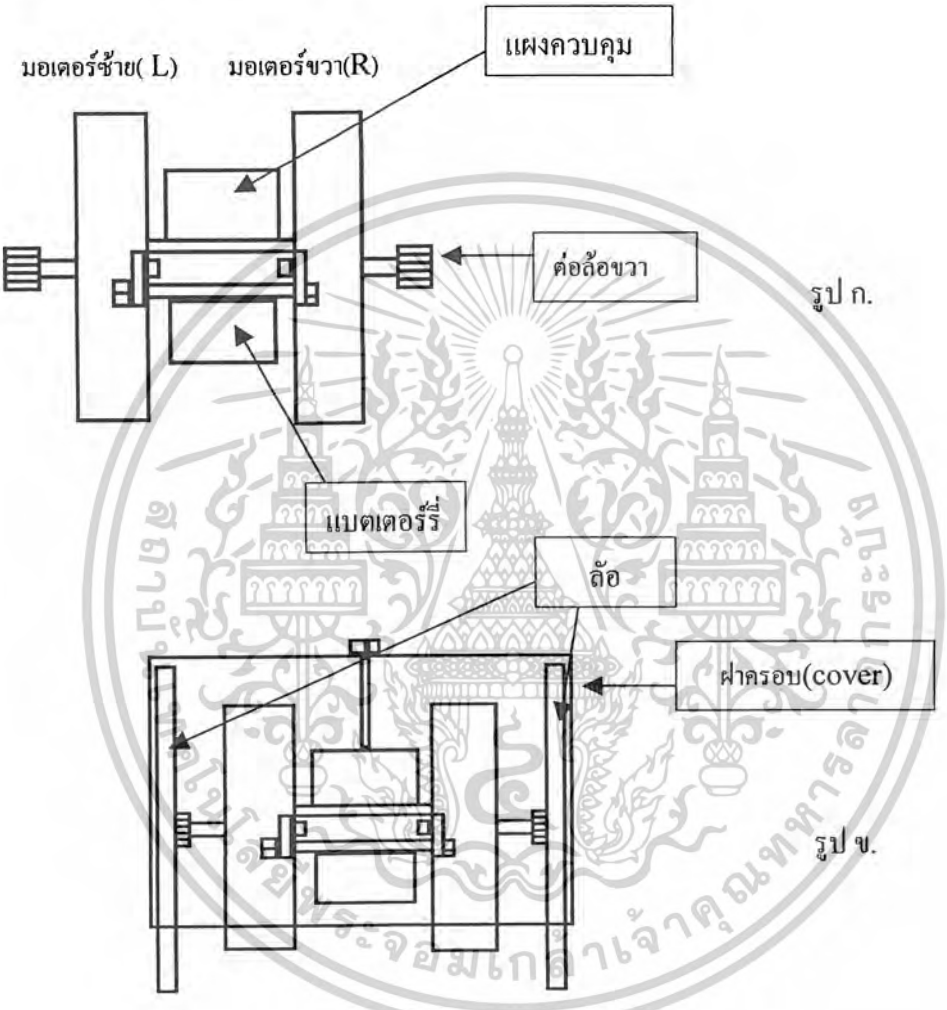


รูป 3.8โฟลวชาร์ตโปรแกรมแทรกกิ้งวัตถุ

3.3 ส่วนของตัวหุ่นยนต์

ขั้นตอนการออกแบบตัวหุ่นนั้นสามารถอธิบายเป็นขั้นตอนได้ดังต่อไปนี้

3.3.1 ลักษณะโครงสร้างตัวหุ่นยนต์



รูป3.9 โครงสร้างภายในใช้สเต็ปปิ้งมอเตอร์(Stepping motor)สองตัว

ก. โครงสร้างภายใน

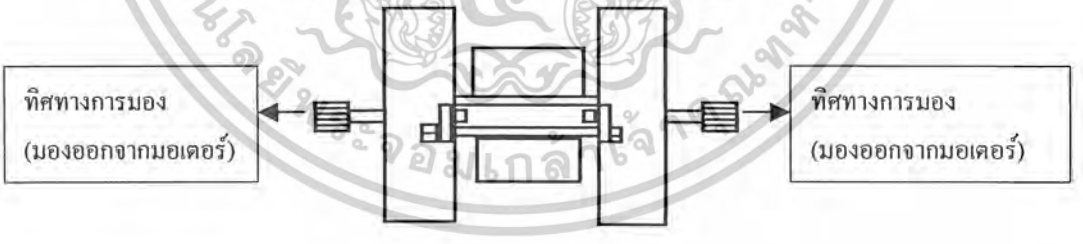
ข. รูปโครงสร้างเมื่อใส่ล้อและสวมฝาครอบ

เพื่อให้ง่ายต่อการควบคุมในพื้นที่แคบและหุ่นยนต์สามารถเคลื่อนที่ในทิศทางต่างๆตามแต่ละสถานการณ์ได้อย่างเหมาะสม ตัวหุ่นจึงต้องสามารถ เดินหน้า , ถอยหลัง , หมุนรอบตัวเอง(ซ้าย , ขวา) เลี้ยวซ้าย , เลี้ยวขวาได้ตามตารางต่อไปนี้

ทิศทางการเคลื่อนที่ของหุ่นยนต์	มอเตอร์	
	ล้อซ้าย (L)	ล้อขวา (R)
เดินหน้า (FORWARD)	ตามเข็ม	ทวนเข็ม
ถอยหลัง (BACKWARD)	ทวนเข็ม	ตามเข็ม
หมุนรอบตัวซ้าย (ROUND_LEFT)	ทวนเข็ม	ทวนเข็ม
หมุนรอบตัวขวา (ROUND_RIGHT)	ตามเข็ม	ตามเข็ม
เลี้ยวซ้ายหน้า (TURN_LEFT_FORWARD)	หยุด	ทวนเข็ม
เลี้ยวซ้ายหลัง (TURN_LEFT_BACKWARD)	หยุด	ตามเข็ม
เลี้ยวขวาหน้า (TURN_RIGHT_FORWARD)	ตามเข็ม	หยุด
เลี้ยวขวาหลัง (TURN_RIGHT_BACKWARD)	ทวนเข็ม	หยุด

ตาราง 4 แสดงการเคลื่อนที่ของหุ่นยนต์สภาวะต่างๆ (ROBOT MOVING ALGORITHM)

ที่สภาวะต่างๆการเคลื่อนที่ของมอเตอร์แต่ละตัวจะใช้การมองออกจากตัวมอเตอร์ จะทำให้ได้สถานะ ทวนเข็ม , ตามเข็ม ตามลำดับ

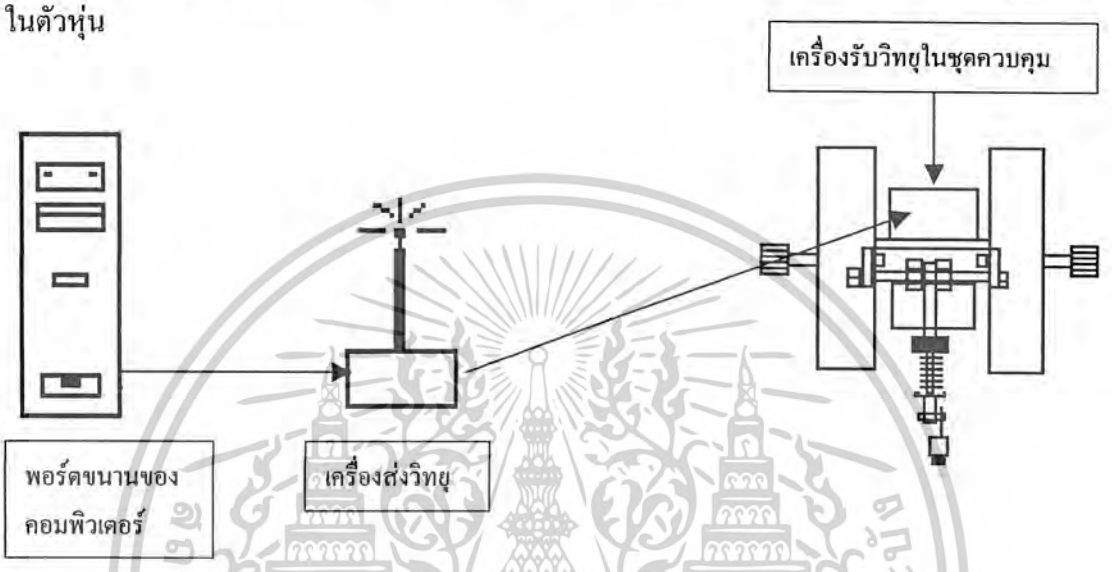


รูป 3.10 แสดงทิศทางการมองการหมุนของแกนมอเตอร์

จากลักษณะการเคลื่อนที่ดังกล่าวและมอเตอร์ทั้งสองตัวเป็นสเต็ปปิ้งมอเตอร์(Stepping Motor) ซึ่งสามารถขับได้หลายวิธีดังที่ได้กล่าวมาแล้วในทฤษฎีข้างต้นดังนั้นในการออกแบบนี้จะใช้การขับสเต็ปปิ้งมอเตอร์แบบฟูลสเต็ปหนึ่งเฟสและใช้ไมโครคอนโทรลเลอร์(Microcontroller) ในการควบคุมในที่นี้จะใช้ไมโครคอนโทรลเลอร์ MCS – 51 ของ ATMEL เบอร์ AT89C2051 ซึ่งมีพอร์ต ศูนย์ (PORT 0) และ พอร์ต 3 (PORT 3) มีหน่วยความจำโปรแกรมภายใน 2 กิโลไบต์ (KB)

3.3.2 **การออกแบบชุดควบคุม**

ชุดควบคุมหุ่นยนต์จะใช้วิทยุรับส่งของรถกระป๋องเป็นช่องส่งข้อมูล โดยรับข้อมูลมาจากพอร์ตขนาน(PARARELL PORT) ของคอมพิวเตอร์ เข้าที่เครื่องส่ง จากนั้นเครื่องรับที่อยู่ในตัวหุ่น

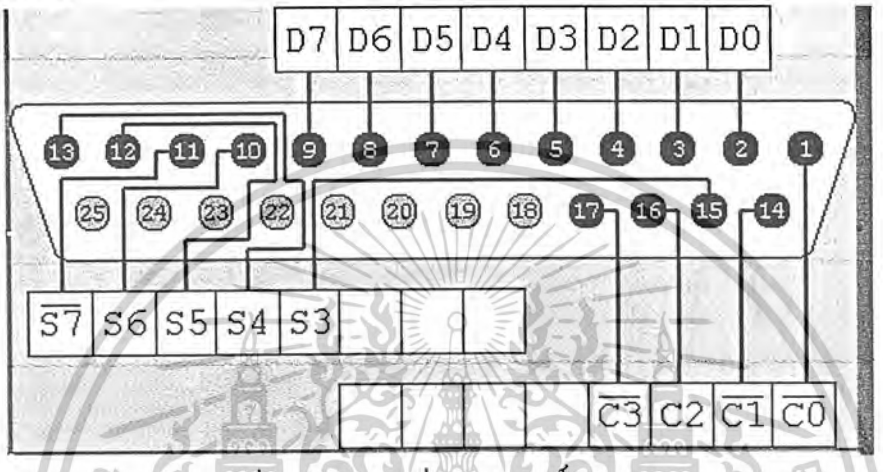


รูป 3.11 การควบคุมหุ่นผ่านทางพอร์ตขนาน (PARARELL PORT)

จะรับสัญญาณควบคุมเพื่อควบคุมตัวหุ่นยนต์ต่อไป แต่สัญญาณควบคุมที่ได้มาจากพอร์ตขนานจะได้มาจากการประมวลผลภาพ (IMAGE PROCESSING) ในเงื่อนไขที่แตกต่างกันซึ่งพอร์ตขนานจะมีมาตรฐานการติดต่อเป็นของตัวเองโดยตัวพอร์ตจะมีขาต่างๆดังต่อไปนี้

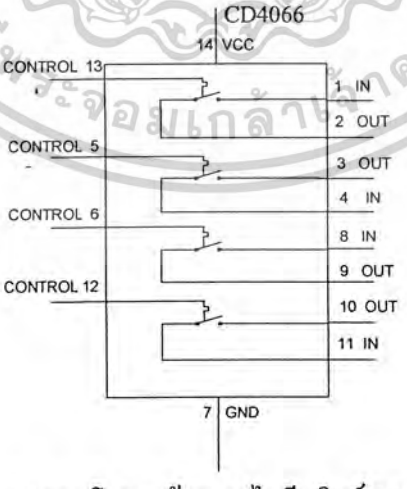
1. ขาข้อมูล 8 ขา [8 OUTPUT PIN DATAPORT]
2. ขาอินพุต 5 ขา [5 INPUT PIN S (1 INVERT) STATUS PORT]
3. ขาควบคุม 4 ขา [4 INPUT PIN (3 INVERT) CONTROL PORT]
4. ขากราวนด์ 8 ขา [8 PIN GROUND]

ในการออกแบบจะใช้ขาข้อมูลเพียง 4 ขา และขา กราวนด์อีก 1 ขา เท่านั้น ก็เพียงพอต่อการส่งข้อมูลแล้ว โดยจะใช้ขา D0 , D1 , D2 , D3 ในการส่งข้อมูลไปที่เครื่องส่งวิทยุ (REMOTE CONTROL:TX) แต่เนื่องจากเครื่องส่งรีโมตจะต้องมีไฟเลี้ยงดังนั้นจึงไม่สามารถที่จะต่อสัญญานจากพอร์ตไปเข้าเครื่องส่งรีโมตได้โดยตรงจึงต้องใช้ ไอซีสวิตช์ (IC SWITCH) เบอร์ CD 4066 ซึ่งมี



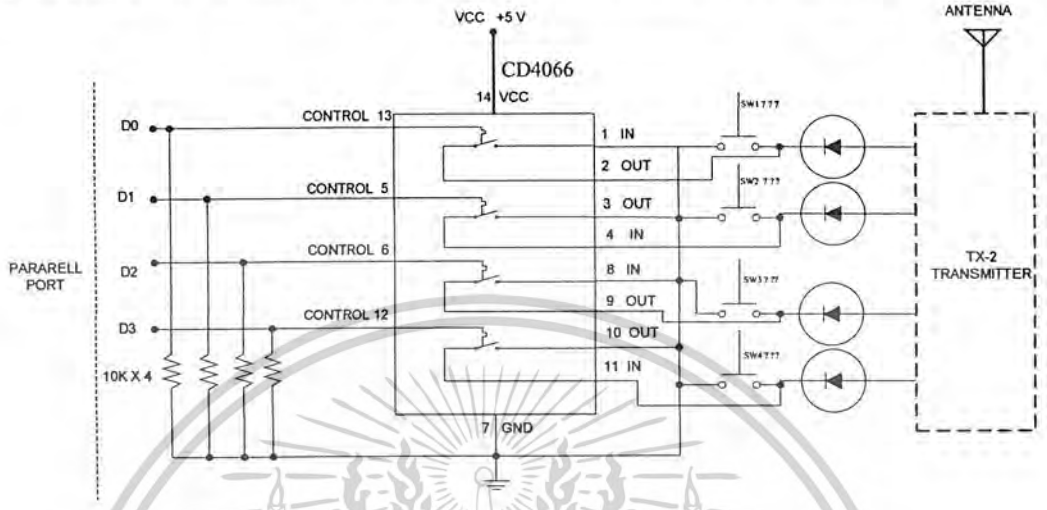
รูป 3.12 แสดงขาต่างๆของพอร์ตขนาน

ภายในมีสวิตช์ทั้งหมด 4 ตัว มีอินพุทเป็นแบบ ทีทีแอล (TTL) และพอร์ตขนานก็มี เอาท์พุทเป็นแบบทีทีแอลจึงต่อกัน แต่อย่างไรก็ตามจะต้องต่อตัวต้านทานพลู-ดาวน์ที่ทุกอินพุทของไอซี CD 4066 เพื่อรอสัญญาณ ลอจิกสูง " 1 " (LOGIC HIGH) ซึ่งสามารถต่อได้ดังรูปได้ดังรูปที่ 3.13 โดยเอาท์พุตของไอซีสวิตช์จะต่อคร่อมสวิตช์ควบคุมของเครื่องส่งวิทยุเพื่อทำงานแทนนาลอกสวิตช์นั่นเอง

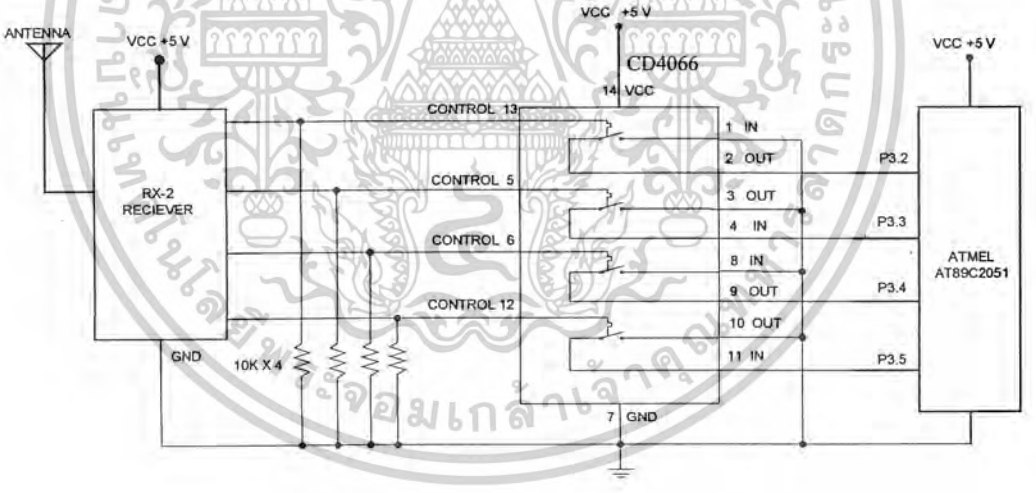


รูป 3.13 แสดงโครงสร้างของไอซีสวิตช์ CD4066

จากโครงสร้างของไอซีสวิตช์จะเห็นว่ามีขาควบคุม (CONTROL) ซึ่งถ้าหากว่าขาควบคุมนี้มีสถานะลอจิกเป็น " 1 " สวิตช์อิเล็กทรอนิกส์ก็จะต่อโดยทำหน้าที่เหมือนสวิตช์ธรรมดา



รูป 3.14 แสดงการต่อพอร์ตขนานเข้ากับเครื่องส่งโดยผ่านไอซี CD 4066



รูป 3.15 การต่อเครื่องรับเข้ากับไมโครคอนโทรลเลอร์ AT89C2051 โดยผ่านไอซี CD 4066

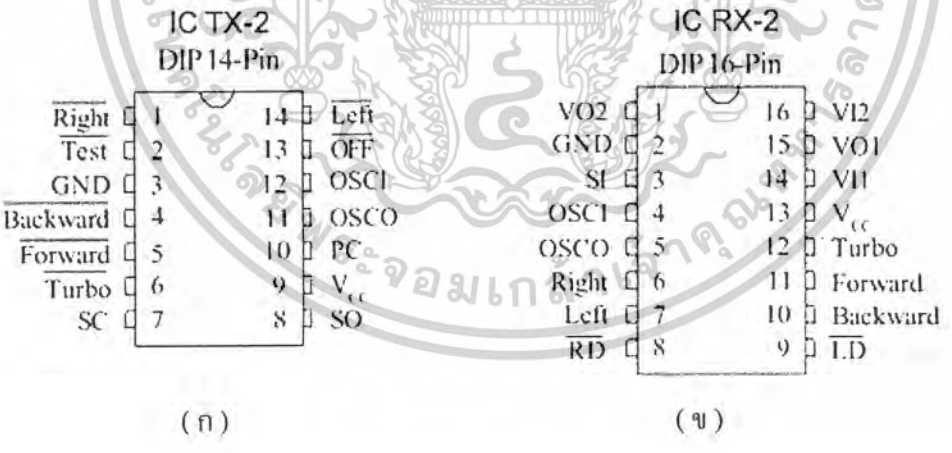
ในการต่อภาครับกับไอซีสวิตช์จะต้องเอาตัวต้านทาน 2.2 กิโลโอห์ม (2.2K) ที่ต่ออนุกรมกับทางเดินสัญญาณออกเพื่อให้กระแสจากขาเอาต์พุต ของภาครับรีโมท (ไอซี RX-2) เพียงพอต่อการเทิน – ออน (TURN – ON) ไอซีสวิตช์และเช่นเดียวกันกับภาคส่งจะต้องต่อตัวต้านทาน พูล – ดาวน์เพื่อให้ขาอินพุตของไอซี CD 4066 รอสัญญาณลอจิกสูง " 1 " (ปรกติเป็นถ้าไม่มีสัญญาณจะเป็นลอจิกต่ำ " 0 ") สำหรับข้อมูล(CONTROL DATA) ที่จะต้องส่งออกมา

จากพอร์ตขนานเพื่อควบคุมตัวหุ่นจะกำหนดให้เป็นดังตารางต่อไปนี้ (ซึ่งจะพิจารณาเพียงแค่ไบต์ต่ำเท่านั้น)

ทิศทางการเคลื่อนที่ของหุ่นยนต์	ขาของพอร์ตขนาน				ข้อมูลที่ต้องส่งออกที่พอร์ต
	D3	D2	D1	D0	
เดินหน้า	1	0	0	1	0X09
ถอยหลัง	0	1	1	0	0X06
หมุนรอบตัว (ซ้าย)	1	0	1	0	0X0A
หมุนรอบตัว (ขวา)	0	1	0	1	0X05
เลี้ยวซ้าย (หน้า)	1	0	0	0	0X08
เลี้ยวซ้าย (หลัง)	0	1	0	0	0X04
เลี้ยวขวา (หน้า)	0	0	0	1	0X01
เลี้ยวขวา (หลัง)	0	0	1	0	0X02

ตาราง 5 แสดงค่าที่ต้องส่งออกพอร์ตที่สถานการณ์เคลื่อนที่ต่างๆ

ค่าต่างๆจากตารางที่ 5 นี้จะถูกส่งผ่านเสาอากาศผ่านอากาศไปยังเครื่องรับที่ชุดควบคุมของตัวหุ่นยนต์ต่อไป โดยมีวงจรและโครงสร้างของไอซีของตัวเครื่องรับและเครื่องส่งเป็นดังนี้



(ก) (ข)

รูป 3.16 แสดงไอซีที่ทำหน้าที่เป็นตัวส่งและตัวรับ

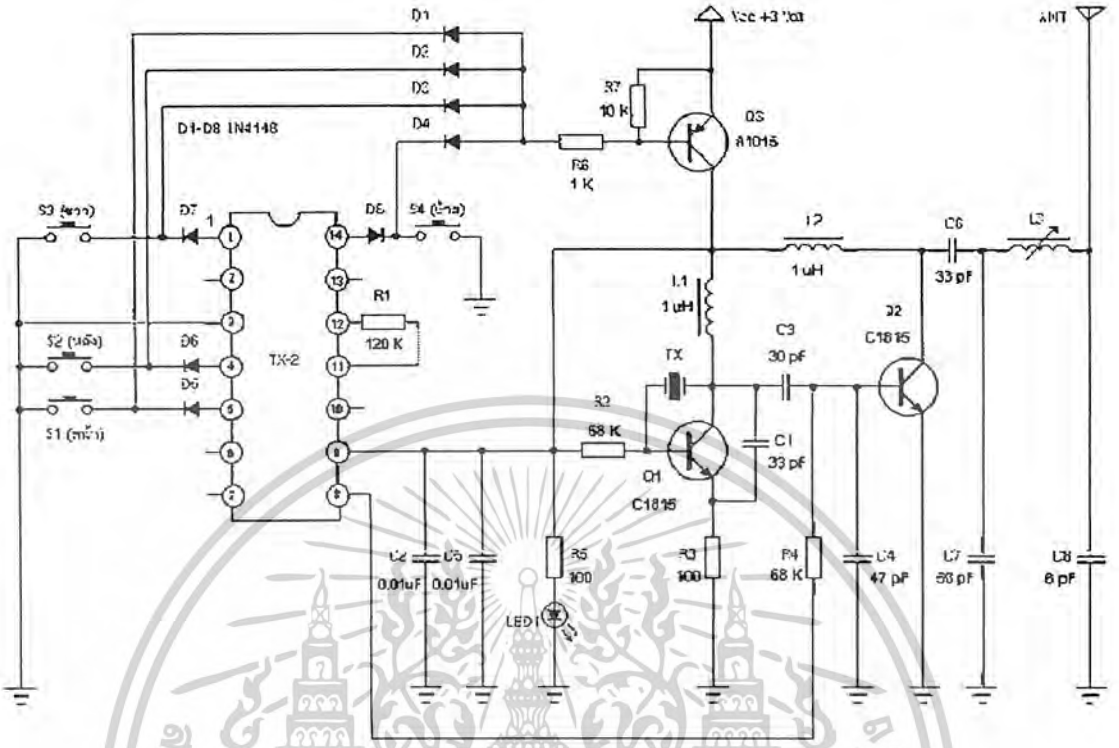
ก) ตัวส่ง

ข) ตัวรับ

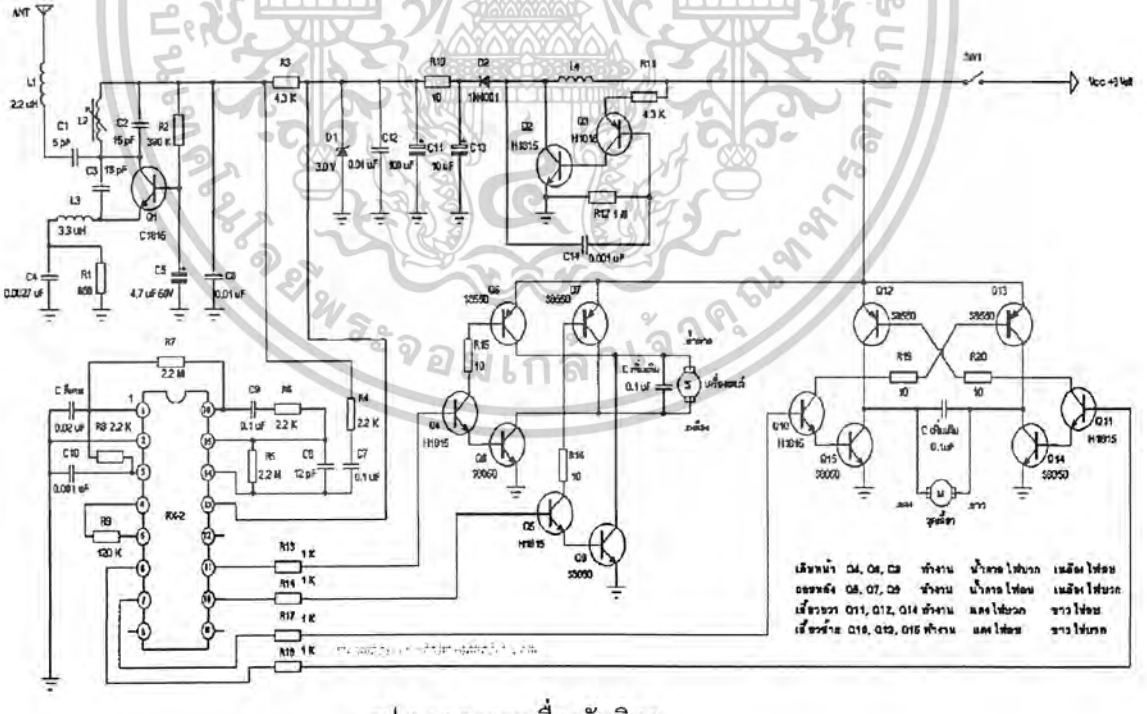
1. transmission circuits TX-2 :			2. accepting circuits RX-2 :		
Pin number	Symbol number	The meritorious can say clearly	Pin number	Symbol number	The meritorious can say clearly
1	RIGHT	When this termination, choice clockwise function.	1	V 02	Uses in phase reverser 2 out-ports which the signal enlarges.
2	TEST	This end uses in to test the pattern.	2	GND	Power source end.
3	GND	Power source negative ground.	3	SI	Coded signal input end.
4	BACKWARD	When this termination, choice backlash function.	4	OSCI	Oscillator input end.
5	FORWARD	When this termination, choice advance function.	5	OSCO	Oscillator out-port.
6	TRUBO	When this termination, choice acceleration function.	6	RIGHT	Clockwise out-port.
7	SC	Belt carrier coded signal out-port.	7	LEFT	Counterclockwise out-port.
8	SO	Does not bring the carrier the coded signal out-port.	8	RD	When this termination, clockwise the function is forbidden.

9	Vcc	Power source	9	LD	When this termination, the counterclockwise
10	PC	Power source control out-port.	10	BACKWARD	Backlash out-port.
11	OSCO	Oscillator out-port.	11	FORWARD	Advance out-port.
12	OSCI	Oscillator input end.	12	TURBO	Acceleration out-port.
13	FOSC	This end uses in to test the pattern.	13	V cc	Power source
14	LEFT	When this meets the place, choice counterclockwise	14	V 11	Uses in phase reverser 1 input end which the signal enlarges.
			15	V 01	Uses in phase reverser 1out-port which the signal enlarges.
			16	V 12	Uses in phase reverser 2 input ends which the signal enlarges.

ตาราง 6 การดูขาIC TX-2 และ RX-2



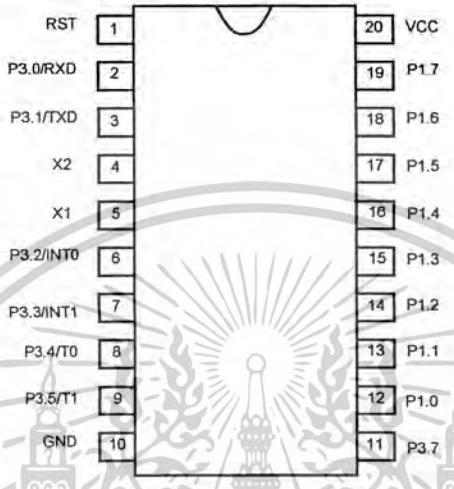
รูป 3.17 วงจรเครื่องส่งวิทยุ



รูป 3.18 วงจรเครื่องรับวิทยุ

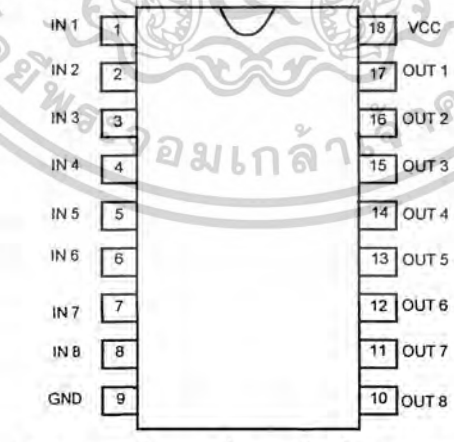
3.3.3 การเขียนโปรแกรมควบคุม

จะใช้ไมโครคอนโทรลเลอร์ MCS-51 ของ ATMAL เบอร์ AT89C2051 และใช้ภาษาแอสเซมบลีในการเขียนโปรแกรมควบคุมซึ่งโครงสร้างของ AT89C2051 เป็นดังนี้

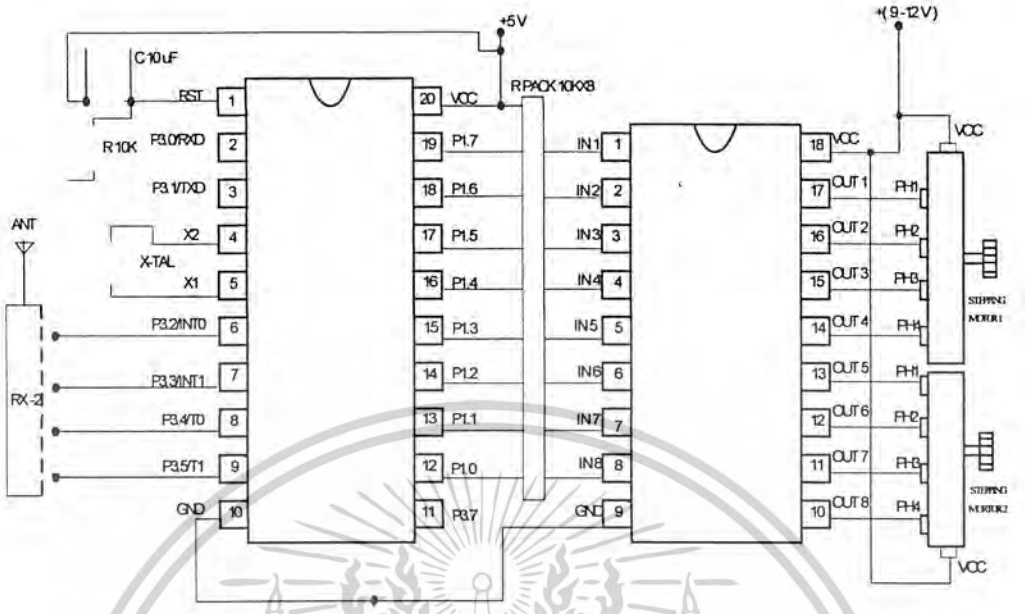


รูป 3.19 แสดงโครงสร้างของ AT89C2051

จากโครงสร้างจะเห็นว่ามีเพียงพอร์ต 1 และ พอร์ต 3 เท่านั้นโดยจะใช้พอร์ต 3 เป็นอินพุตและพอร์ต 1 เป็นเอาต์พุต ซึ่งอินพุตพอร์ต 3 จะรับสัญญาณควบคุมมาจากเครื่องรับวิทยุซึ่งต่อกันดังรูปที่ 3.14 สำหรับเอาต์พุตพอร์ต 1 จะต่อกับไอซีขับสเตปปิ้งมอเตอร์ เบอร์ ULN 2803A ภายในเป็นตัวขับ (DRIVER)แบบดาลิงตัน (DARLINGTON) 8 ช่อง (8 CHANNEL) สามารถขับกระแสได้ 500 มิลลิแอมแปร์ ต่อ1 ช่อง ตัว ULN2803 มีโครงสร้างดังต่อไปนี้



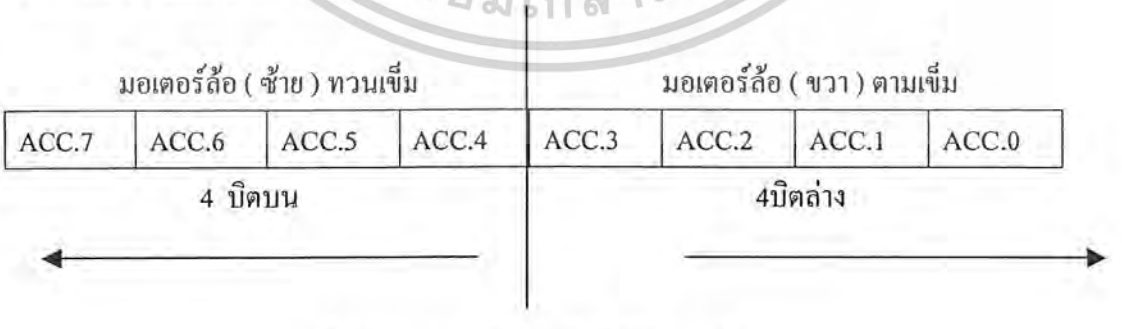
รูป 3.20 แสดงโครงสร้างของไอซี ULN2803A



รูป 3.21 แสดงการต่อไมโครคอนโทรลเลอร์กับไอซีขับมอเตอร์ (ULN2803A)

เนื่องจากว่าภายในตัวไมโครคอนโทรลเลอร์ AT89C2051 มีรีจีสเตอร์ที่กระทำทางคณิตศาสตร์และลอจิกเพียงตัวเดียวคือ รีจีสเตอร์ เอ (REGISTER A : ACC) แต่สเตปปิ้งมอเตอร์ที่ใช้มีสองตัวดังนั้นจึงต้องใช้วิธีการตามตารางที่ 1 เพื่อให้เกิดการเลื่อนบิตตามลักษณะการเคลื่อนที่แบบต่างๆดังต่อไปนี้

เมื่อตัวหุ่นเคลื่อนที่ไปข้างหน้า (FORWARD) สามารถเขียนสถานการณ์เลื่อนบิตของรีจีสเตอร์เอ(ACC) โดยจะแบ่งการเคลื่อนที่เป็นการเคลื่อนที่ของมอเตอร์ล้อซ้ายและมอเตอรีล้อขวาให้เป็น 4 บิตบนและ 4 บิตล่างตามลำดับซึ่งสามารถแสดงได้ดังนี้



รูป 3.22 แสดงการเลื่อนบิตในรีจีสเตอร์ เอ

จะเห็นได้ว่าบิต ACC.4 จะต้องเลื่อนไปหาบิต ACC.7 พร้อมกันกับ ที่บิต ACC.3 จะต้องเลื่อนไปหา ACC.0 ซึ่งสามารถกำหนดรูปแบบการเริ่มต้นได้โดย

สเตปที่	มอเตอร์ล้อซ้าย (ทวนเข็ม)				มอเตอร์ล้อขวา (ตามเข็ม)			
1	0	0	0	1	1	0	0	0
2	0	0	1	0	0	1	0	0
3	0	1	0	0	0	0	1	0
4	1	0	0	0	0	0	0	1

ตาราง 7 แสดงการเลื่อนบิตในสถานะเดินหน้า (FORWARD)

ซึ่งสามารถกำหนดค่าเริ่มต้นของมอเตอร์ซ้ายและมอเตอร์ขวาได้เป็น STEP_A = 00010000B และ STEP_B = 00001000B ตามลำดับสำหรับในอีก 7 สถานะของการเคลื่อนที่ที่เหลือก็ใช้การพิจารณาเช่นเดียวกับการเคลื่อนที่แบบเดินหน้า (FORWARD) เพียงแต่เปลี่ยนการเลื่อนบิตตามตารางที่ 1 (ทวนเข็ม หรือ ตามเข็ม) สุดท้ายจะได้ค่าเริ่มต้นมี 4 ค่าดังนี้

STEP_A = 00010000B

STEP_B = 00001000B

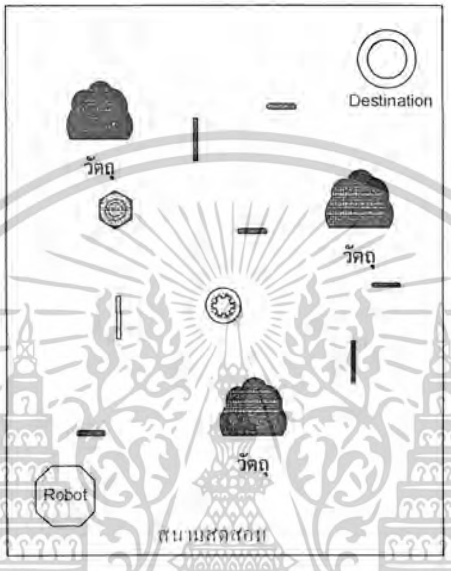
STEP_C = 00000001B

STEP_D = 10000000B

โดยแต่ละสถานะจะต้องกำหนดค่าการเลื่อนบิดให้เหมาะสมเพียงสองค่าในแต่ละสถานะเท่านั้น ซึ่งค่าเริ่มต้นทั้ง 4 ค่านี้จะครอบคลุมการเคลื่อนที่ทั้ง 8 สถานะซึ่งสามารถนำไปเขียนแอสเซมบลีได้เลยจะได้โค้ด (CODE) แอสเซมบลีดังในภาคผนวก

3.4 สนามที่ใช้ในการทดสอบ

ในส่วนของสนามจำลองที่ใช้ทดสอบนั้นเป็นสนามที่ออกแบบให้สีของพื้นหลังมีสีอ่อนกว่าวัตถุและตัวหุ่นยนต์นั้นคือมีสีขาว เพื่อให้โปรแกรมสามารแยกแยะระหว่างวัตถุที่มีสีเข้มกว่าได้อย่างชัดเจนโดยที่ลักษณะของสนามดังกล่าวสามารถแสดงได้ดังต่อไปนี้



รูป 3.23 แสดงลักษณะของสนามและวัตถุ ทางด้านบน(Top View)

รูป 3.24 แสดงลักษณะของสนามและกล้อง ทางด้านข้าง(Side View)

โดยที่สนามด้านบนจะติดกล้องเว็ปแคมรุ่นVideo Blaster from Creative และตั้งความละเอียดไว้ที่ 160 x120 พิเซล โดยติดต่อกับคอมพิวเตอร์ผ่าน พอร์ต ยูเอสบี

บทที่ 4
การทดลอง และผลการทดลอง

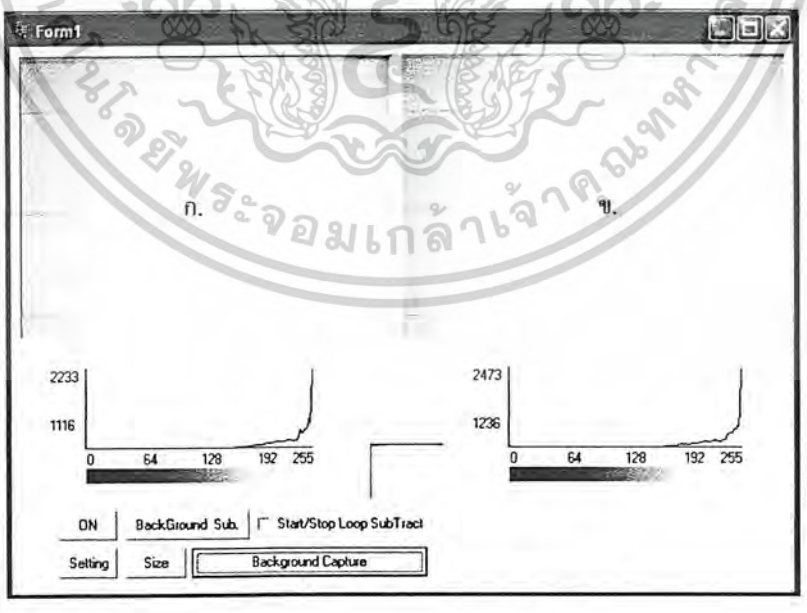
4.1 การทดลองส่วนโปรแกรมประมวลผลภาพ

ในการออกแบบโปรแกรมเกี่ยวกับส่วนประมวลผลภาพในโครงงานเล่มนี้โปรแกรมเมอร์ได้ทำการค้นคว้า (ค้นและคว้ามา)ข้อมูลหลายอย่างเกี่ยวกับ อิมเมจ โปรเซสซิ่ง และการใช้งานโปรแกรม ซี++ บิวล์เดอร์ ทั้งจากรายงานของรุ่นพี่และเว็ปไซค์จำนวนมาก เพื่อหาวิธีการออกแบบทำงานของโปรแกรม และอัลกอริทึม (Algorithm) ของการเขียนซอร์ฟแวร์ แล้วได้นำมาปรับแต่งและพัฒนาให้เหมาะสมกับจุดประสงค์ของโครงงาน ดังนั้นโปรแกรมที่สร้างขึ้นมาเพื่อทดลองในบทนี้มีทั้งที่ถูกใช้และไม่ได้นำมาใช้ในโปรแกรมแต่ก็ถือว่าเกี่ยวข้องกับโครงงานทั้งสิ้นดังนั้นจะขอเริ่มการทอลองดังต่อไปนี้

4.1.1 การลบเฉดสีของพื้นหลัง(Background Subtraction)

เหตุผลที่นำมาแสดงให้ดูก็เพราะว่า ในกรณีที่พื้นหลังของภาพเป็นสีขาว และแสงที่มาตกกระทบบริเวณพื้นหลังของภาพนั้นไม่คงที่ตลอดทั้งภาพ จะทำให้เกิดเงาในบริเวณที่โดนแสงน้อยจึงทำให้เมื่อนำภาพมาประมวลผลจะทำให้โปรแกรมทำงานได้ไม่มีเสถียรภาพ ดังนั้น จึงต้องใช้วิธีการลบส่วนที่มืดของพื้นหลังออกไป โดยมีวิธีการดังนี้

-ทำการจับภาพพื้นหลัง(Background Capture)ที่ยังไม่ได้ใส่วัตถุมาเก็บไว้ดังรูป 4.1



รูป 4.1 แสดงโปรแกรมลบเฉดสีพื้นหลัง

จากรูป 4.1(ก.) แสดงภาพจากกล้องวีดิโอ และรูป4.1(ข.) รูปพื้นหลังที่จับมาจากกล้องวีดิโอ

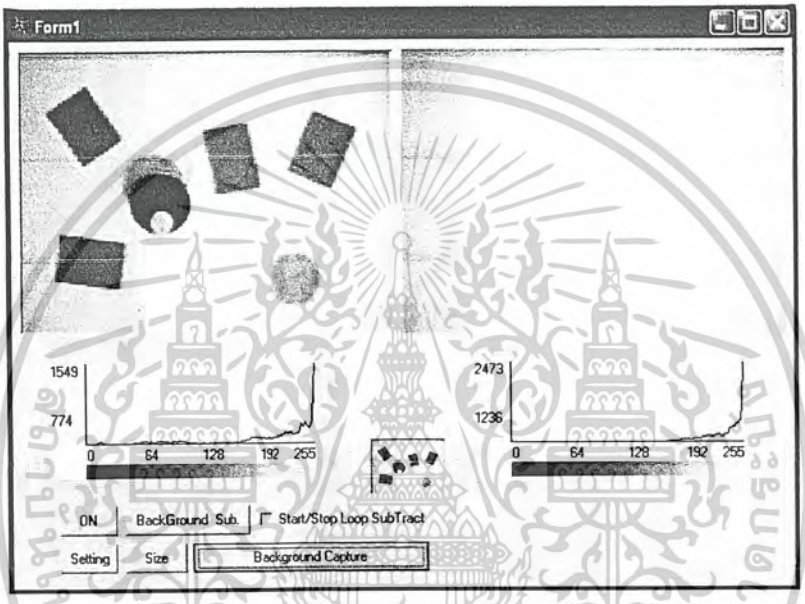
-เมื่อได้ทำการจับภาพพื้นหลังได้แล้วต่อไปทำการลบฉากหลัง (Background Subtraction) ด้วยสมการ $New \quad image = 255 - |x - y|$

เมื่อ 255 คือระดับความสว่างสูงสุด(สีขาว)

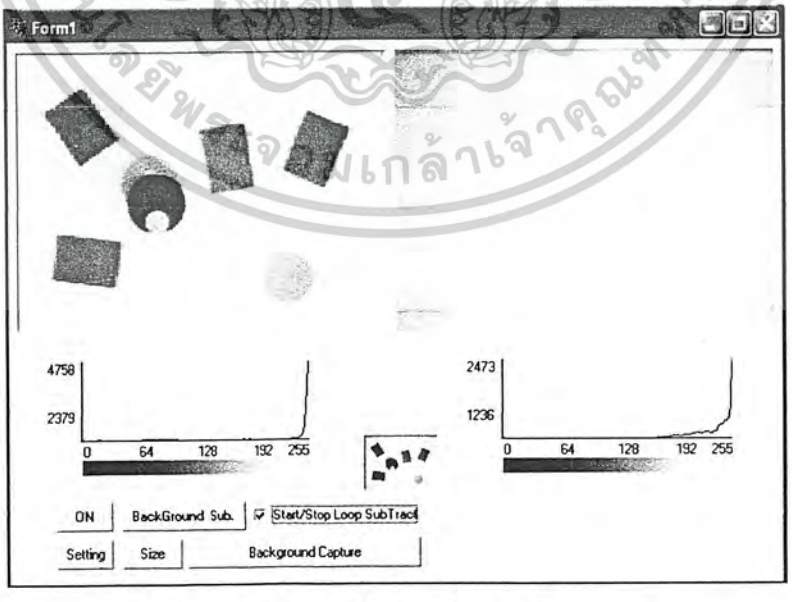
X คือ พิกัด(x_0,y_0)ใดๆของฉากหลังที่จับได้

Y คือ พิกัด(X_0,Y_0)ใดๆของภาพใหม่ที่มีวัตถุอยู่ด้วย

และจะได้ผลออกมาดังรูป 4.3



รูป 4.2 ภาพจากกล้องเมื่อใส่วัตถุเข้าไป

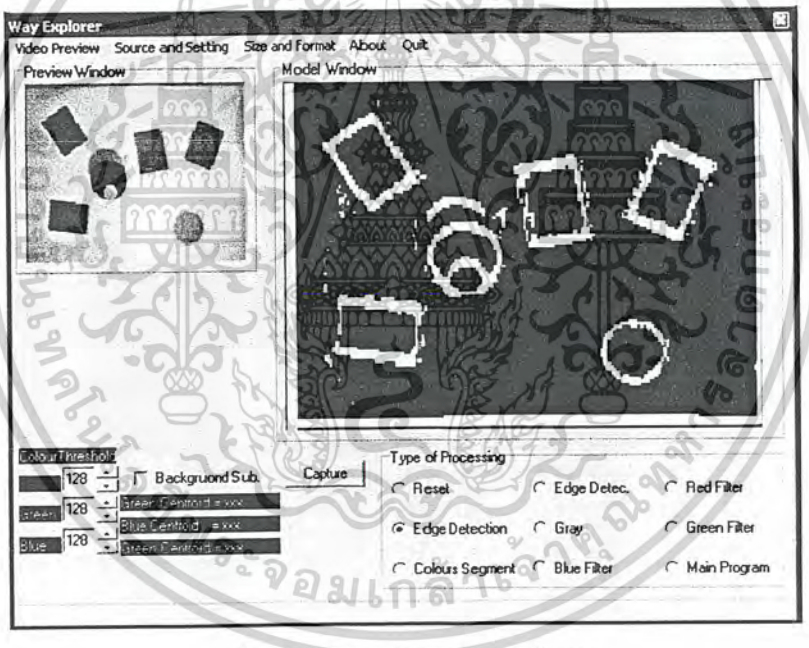


รูป 4.3 แสดงผลการทำ Background Subtraction

4.1.2 การทดลองหาขอบภาพ(Edge Detection)

ในอิมเมจโปรเซสซิ่งที่ต้องการให้โปรแกรมสามารถตรวจจับวัตถุ(Object Tracking) และ/หรือรู้จักวัตถุ(Object Recognition) จำเป็นอย่างยิ่งที่จะต้องใช้การหาขอบภาพเพื่อใช้หาลักษณะเด่นที่จำเป็นที่อยู่ในภาพก่อนการประมวลผล เพื่อเปรียบเทียบกับฐานข้อมูลหรือภาพต้นแบบ (Model Picture) แต่เนื่องจากว่าการตรวจจับวัตถุในลักษณะนี้ จะต้องใช้การออกแบบโปรแกรมที่ซับซ้อนมากจนเกินความรู้ที่มีอยู่ ดังนั้นการหาขอบภาพนี้จึงไม่ได้ถูกนำมาใช้ในโครงงานนี้ด้วยแต่ก็ได้ทำการทดสอบให้เห็นดังรูป 4.4 โดยได้ใช้เทมเพลทของ โซเบล ดังนี้ โดยใช้ค่าเทรชโฮลด์ 128

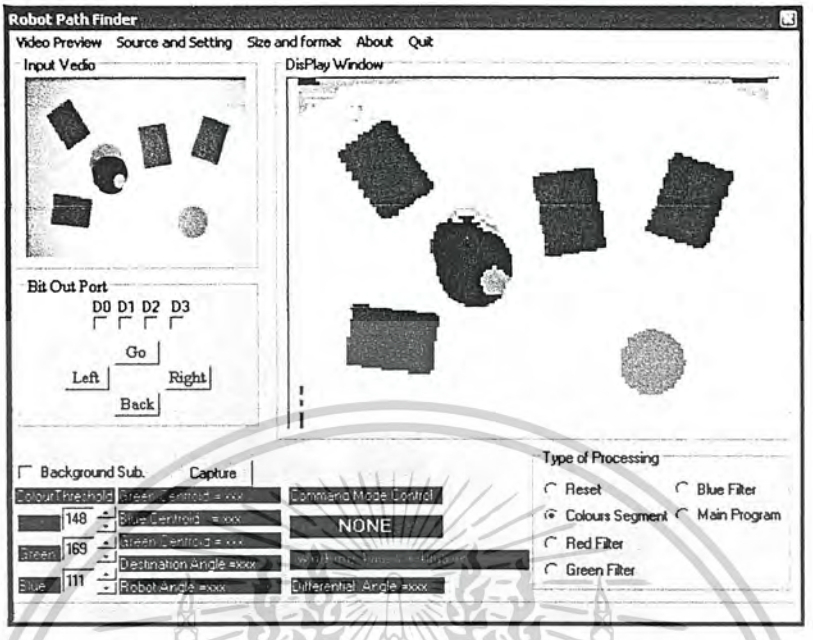
$$h1 = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, \qquad h2 = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}.$$



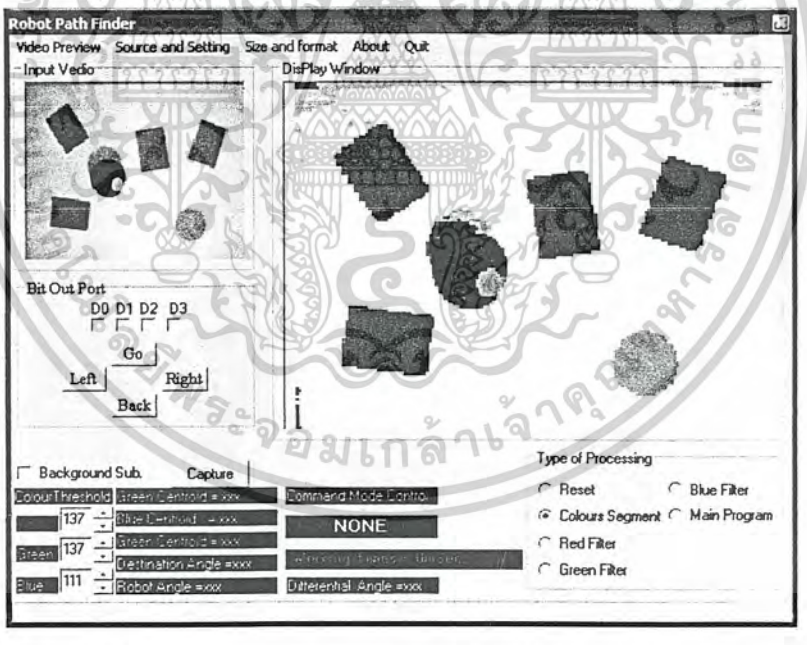
รูป 4.4 การหาขอบภาพ แบบ โซเบล

4.1.3 การแยกสี(Colors Segmentation)

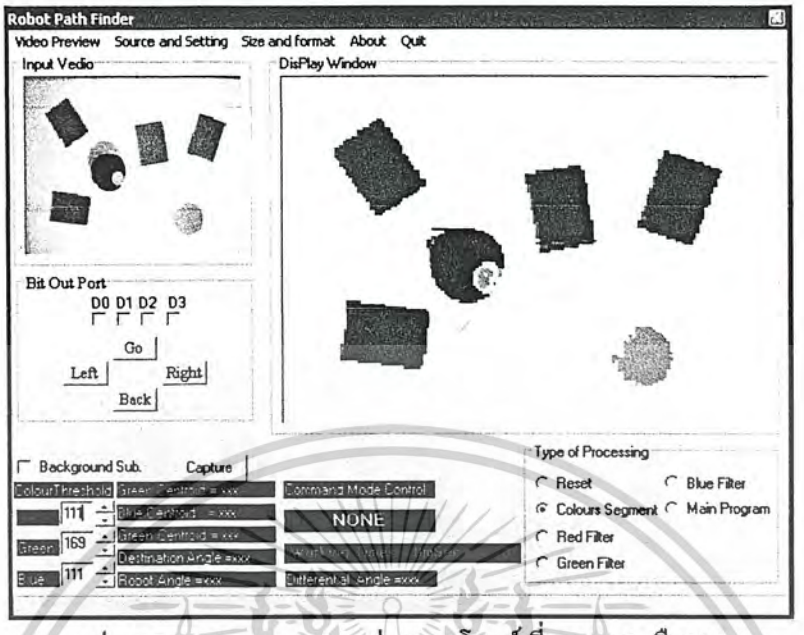
ก่อนที่จะทำการนำภาพไปแทรกกิ้งวัตถุ จะต้องปรับให้ภาพมีความเหมาะสมก่อนจะนำไปประมวลผลซึ่งขั้นตอนแรกก็คือการแยกส่วนสี เพื่อภาพให้มีความแตกต่างกันระหว่างวัตถุแต่ละกลุ่มและวัตถุกับฉากหลังอย่างชัดเจนมากขึ้น แสดงรูปที่4.5 โดยการกำหนดค่าเทรชโฮลด์สีนั้น แต่ละสีก็จะใช้ค่าเทรชโฮลด์ที่ต่างกันเพื่อที่จะให้เกิดความสมบูรณ์ของแต่ละสีให้มากที่สุด ซึ่งการแยกส่วนสีแดง สีเขียวและสีน้ำเงิน แสดงในรูป 4.5, 4.6 ,4.7 ตามลำดับต่อไปนี้



รูป 4.5 Red Segmentation ค่า เทรชโฮลด์ ที่เหมาะสมคือ148



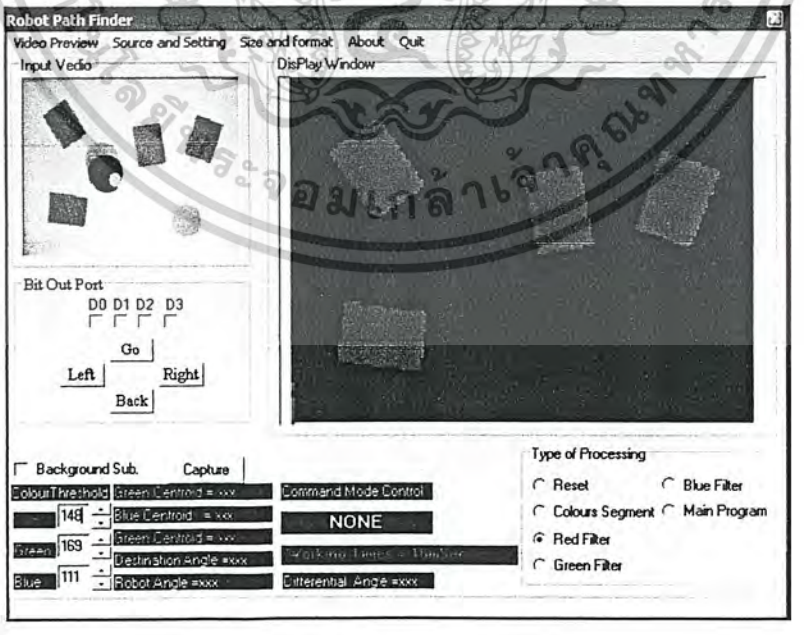
รูป 4.6 Green Segmentation ค่า เทรชโฮลด์ ที่เหมาะสมคือ137



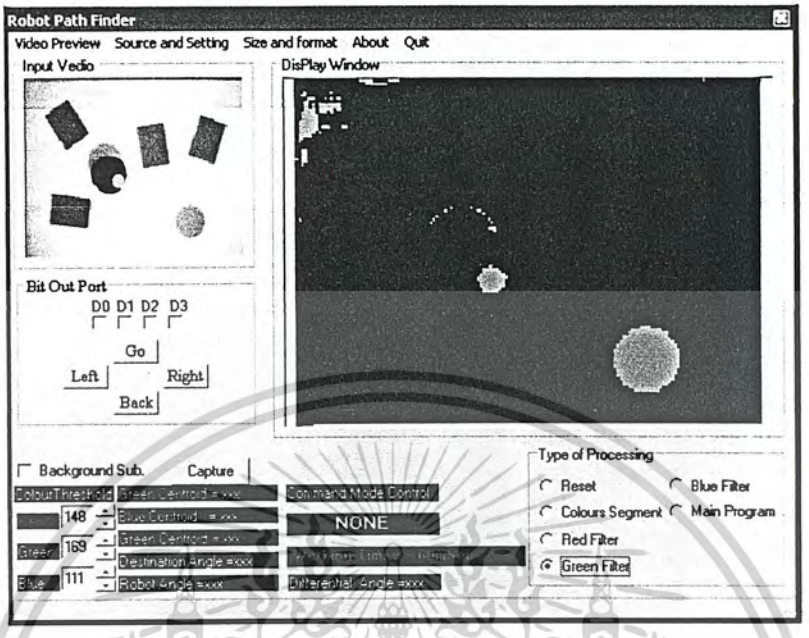
รูป 4.7 Blue Segmentation ค่า เทรชโฮลด์ ที่เหมาะสมคือ111

4.1.4 การกรองสีและหาตำแหน่งวัตถุ(Color Filtration and Object Positioning)

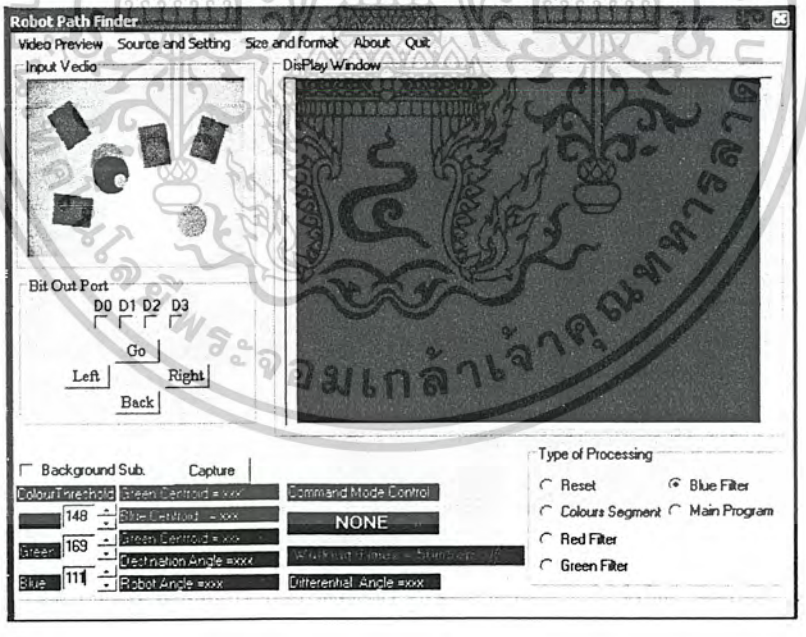
หลังจากที่ได้ทำ(Color Segmentation) ไปแล้วและได้ปรับแต่ค่าเทรชโฮลด์ให้กับแต่ละสีอย่างเหมาะสมแล้ว ขั้นตอนต่อไปจะนำผลที่ได้มากรองเอาเฉพาะที่ต้องการมาประมวลผล เพื่อเก็บตำแหน่งศูนย์กลางของวัตถุ สีเขียว และสีน้ำเงิน ส่วนสีแดงจะเก็บค่าทุกตำแหน่งที่เป็นสีแดงไว้ในอาเรย์ เพื่อใช้ในการแทรกกิ้งในโปรแกรมย่อยลำดับต่อไป แต่ในที่นี้ไม่ได้แสดงค่าที่หามาได้ไว้ แสดงผลการกรองสีแดง สีเขียว และ สีน้ำเงิน ดังรูปที่ 4.8, 4.9, 4.10 ตามลำดับต่อไปนี้



รูป 4.8 Red Filtration



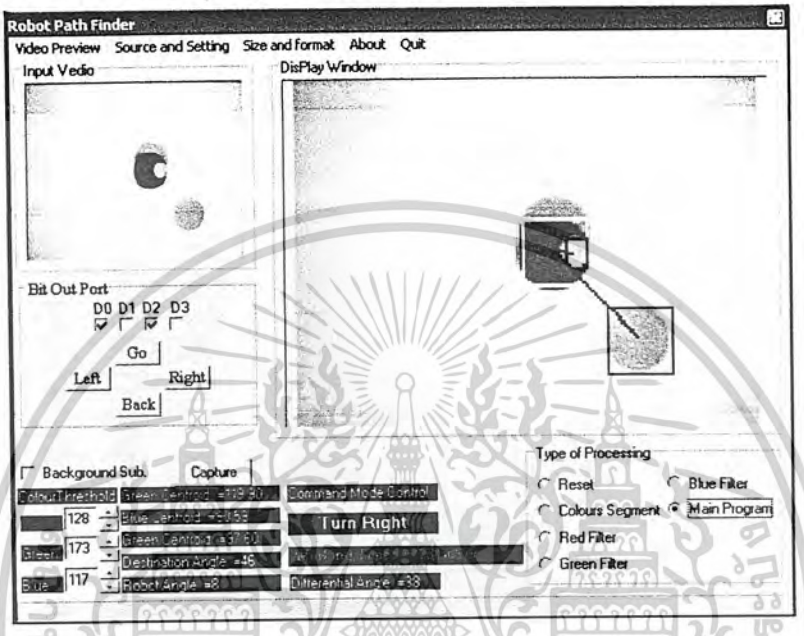
รูป 4.9 Green Filtration



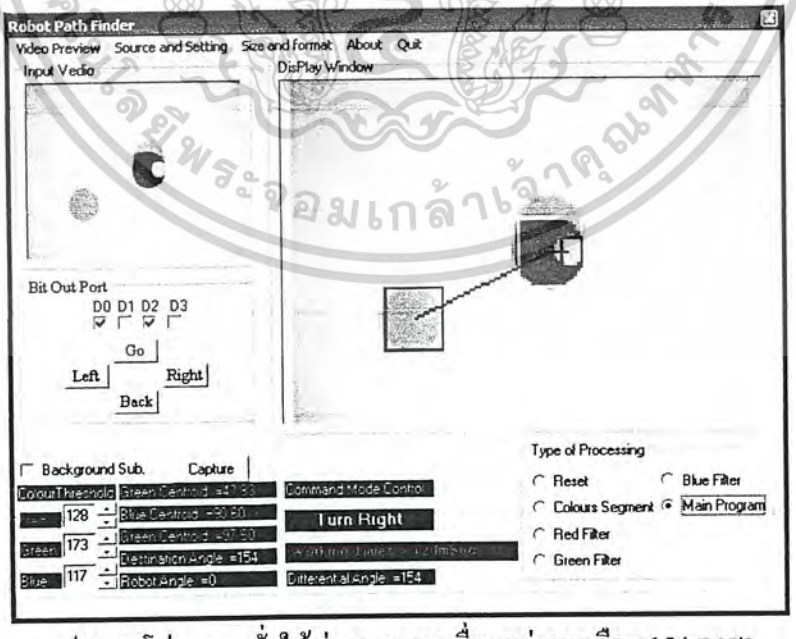
รูป 4.10 Blue Filtration

4.1.5 **ทดลองควบคุมการหมุนของหุ่นยนต์**

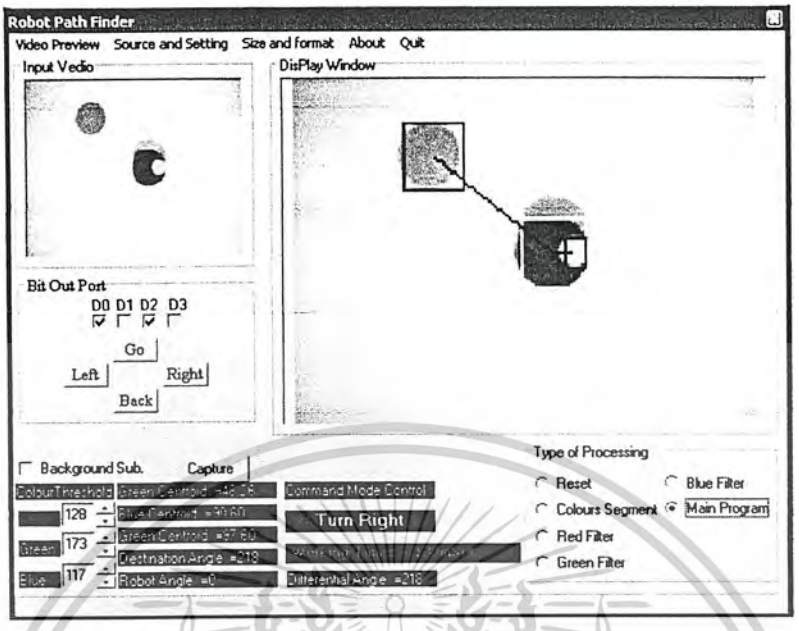
ขั้นตอนต่อไปเป็นการทอลองควบคุมการหมุนของหุ่นยนต์และการออกคำสั่งควบคุมโดย การทดลองจะทดลองให้หุ่นหันหน้าไปในมุมต่างๆคือ 0,90,180 และ270 องศาเพื่อให้เห็นการตัดสิน ใจของโปรแกรมว่าจะสั่งให้ตัวหุ่นหมุนซ้ายหรือหมุนขวาถ้าจุดหมายปลายทางเคลื่อนที่รอบตัวหุ่น



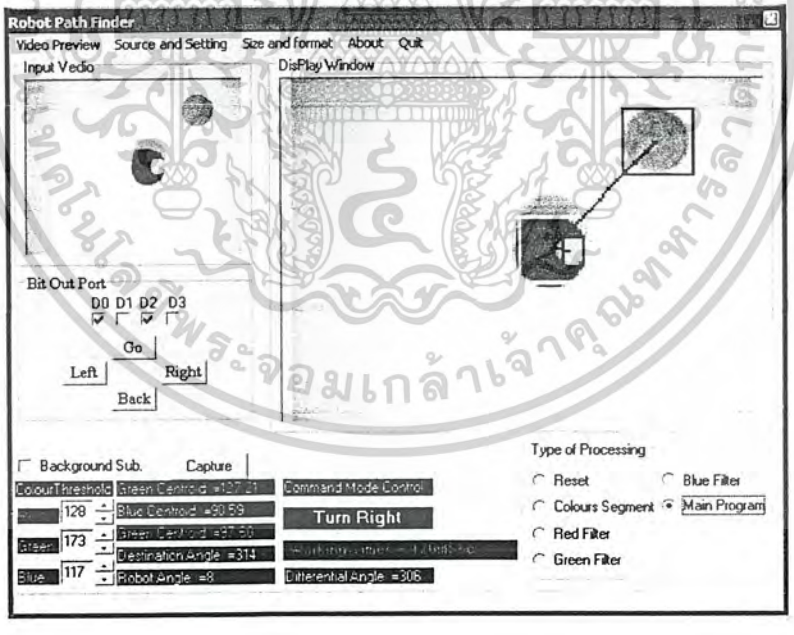
รูป 4.11 โปรแกรมสั่งให้หุ่นหมุนขวาเมื่อผลต่างมุมคือ +38 องศา



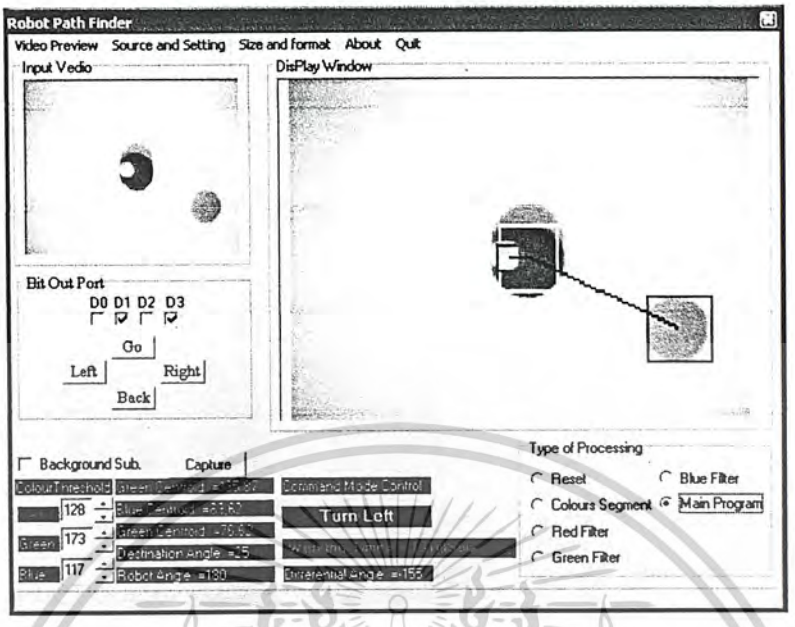
รูป 4.12 โปรแกรมสั่งให้หุ่นหมุนขวาเมื่อผลต่างมุมคือ +154 องศา



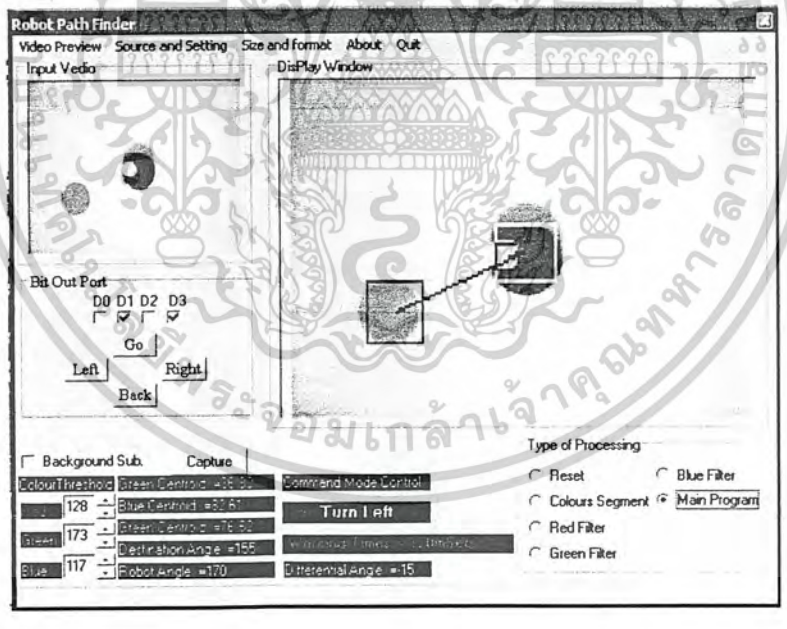
รูป 4.13 โปรแกรมสั่งให้หุ่นหมุนขวาเมื่อผลต่างมุมคือ +218 องศา



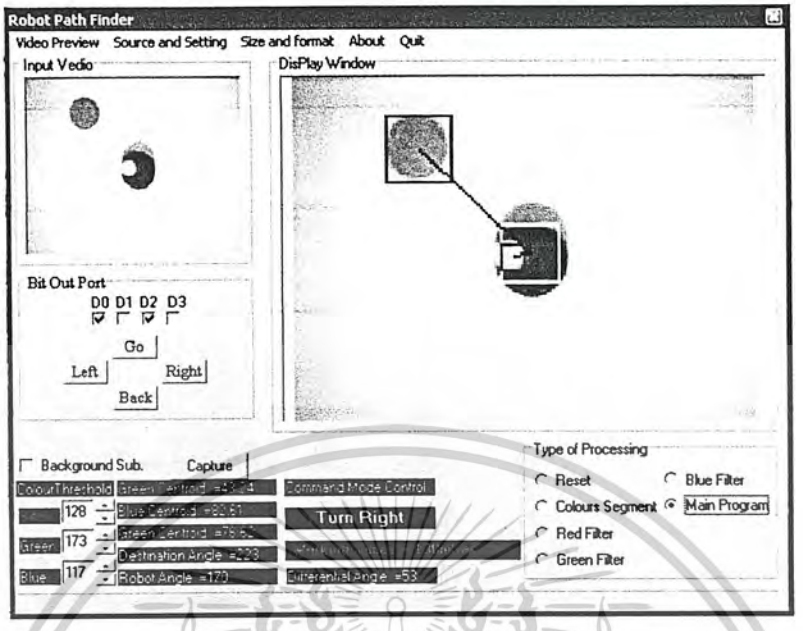
รูป 4.14 โปรแกรมสั่งให้หุ่นหมุนขวาเมื่อผลต่างมุมคือ +306 องศา



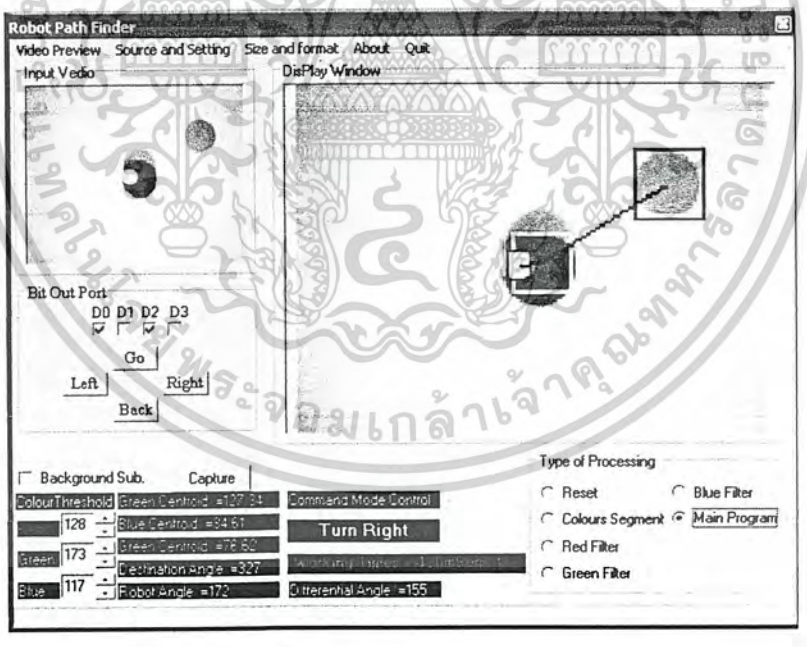
รูป 4.15 โปรแกรมสั่งให้หุ่นหมุนซ้ายเมื่อผลต่างมุมคือ -155 องศา



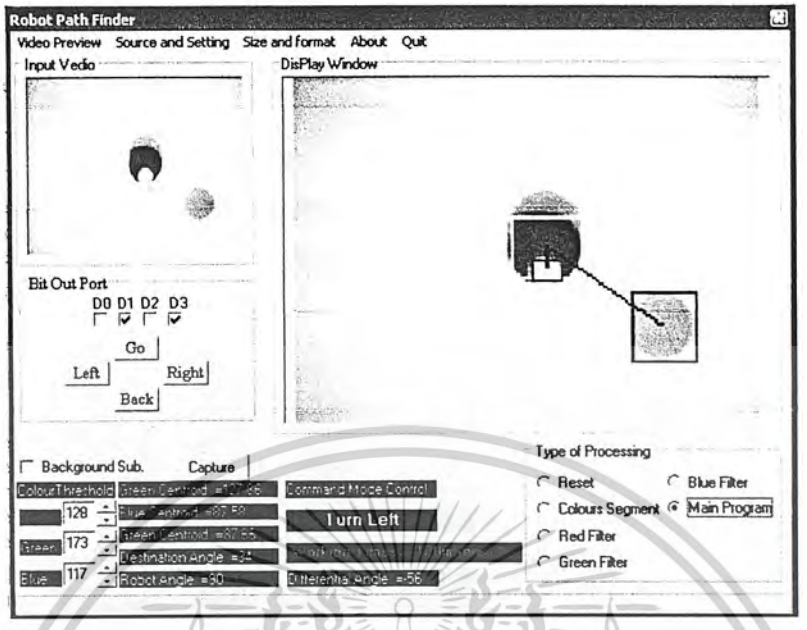
รูป 4.16 โปรแกรมสั่งให้หุ่นหมุนซ้ายเมื่อผลต่างมุมคือ -15 องศา



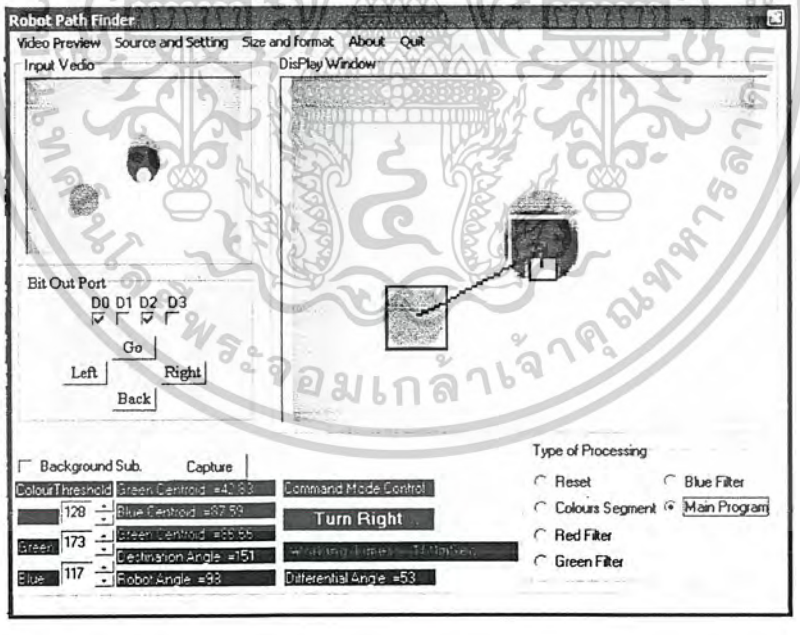
รูป 4.17 โปรแกรมสั่งให้หุ่นหมุนขวาเมื่อผลต่างมุมคือ +53 องศา



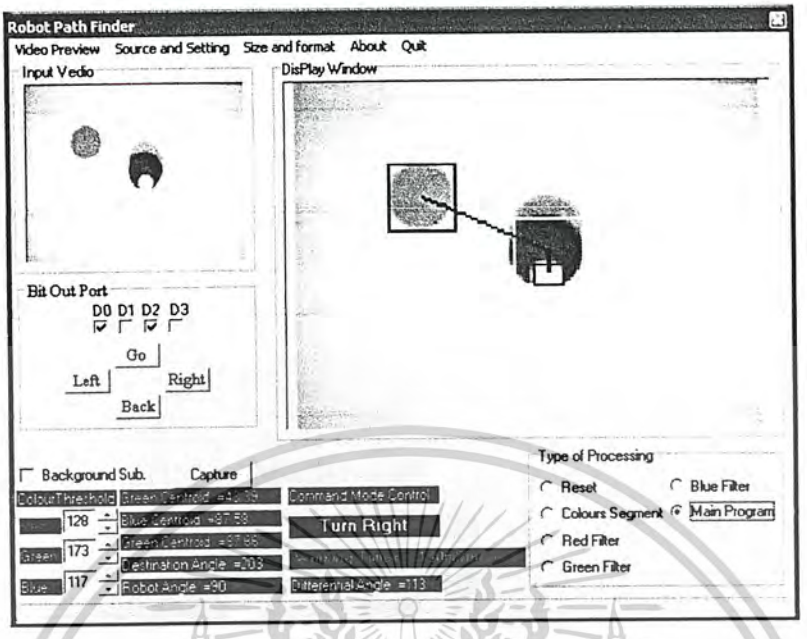
รูป 4.18 โปรแกรมสั่งให้หุ่นหมุนขวาเมื่อผลต่างมุมคือ +155 องศา



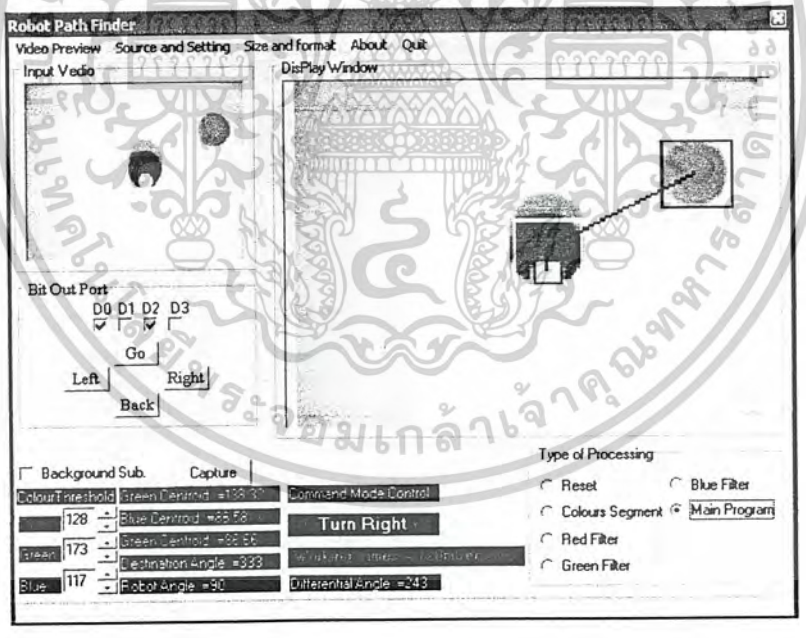
รูป 4.19 โปรแกรมสั่งให้หุ่นหมุนซ้ายเมื่อผลต่างมุมคือ -56 องศา



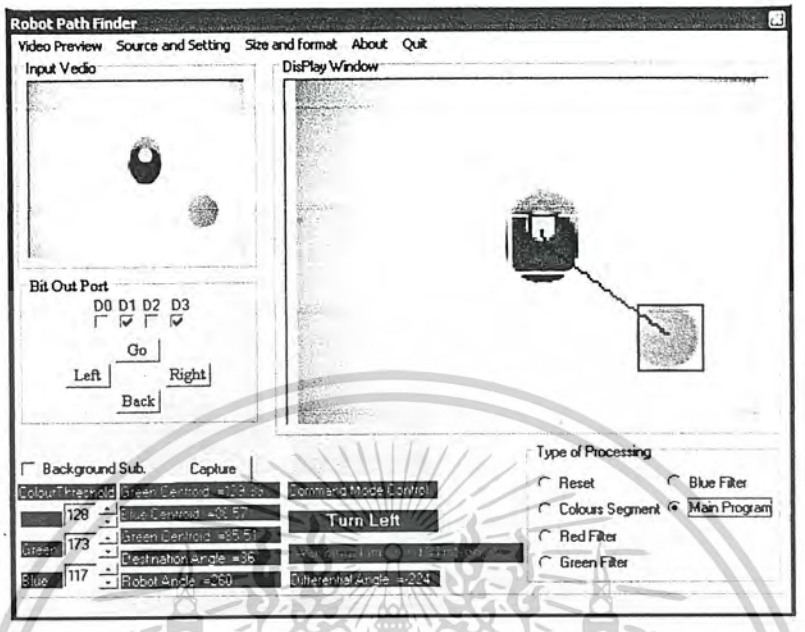
รูป 4.20 โปรแกรมสั่งให้หุ่นหมุนขวาเมื่อผลต่างมุมคือ +53 องศา



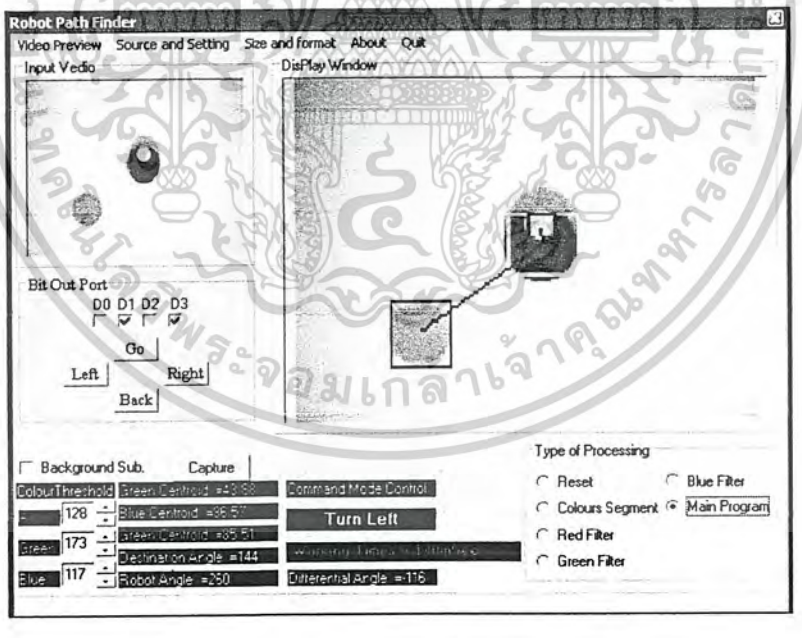
รูป 4.21 โปรแกรมสั่งให้หุ่นหมุนขวาเมื่อผลต่างมุมคือ +113 องศา



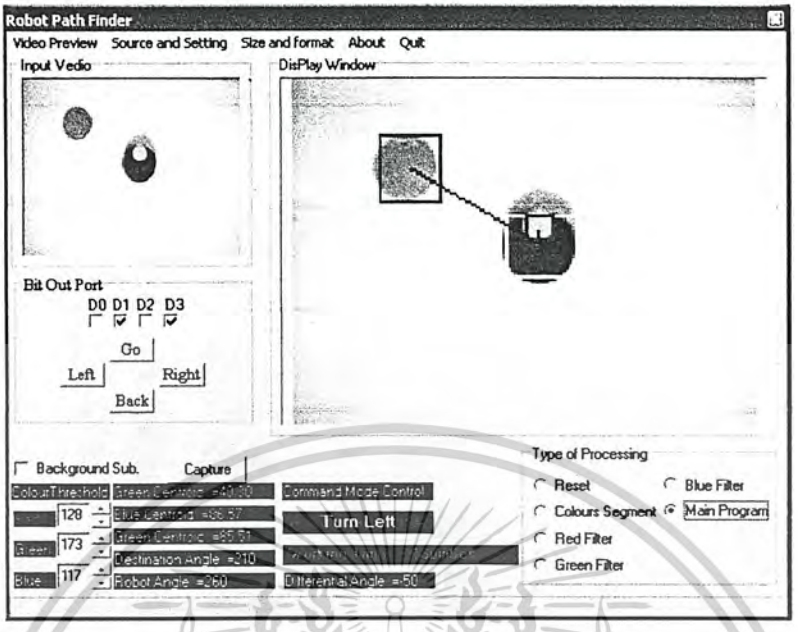
รูป 4.22 โปรแกรมสั่งให้หุ่นหมุนขวาเมื่อผลต่างมุมคือ +243 องศา



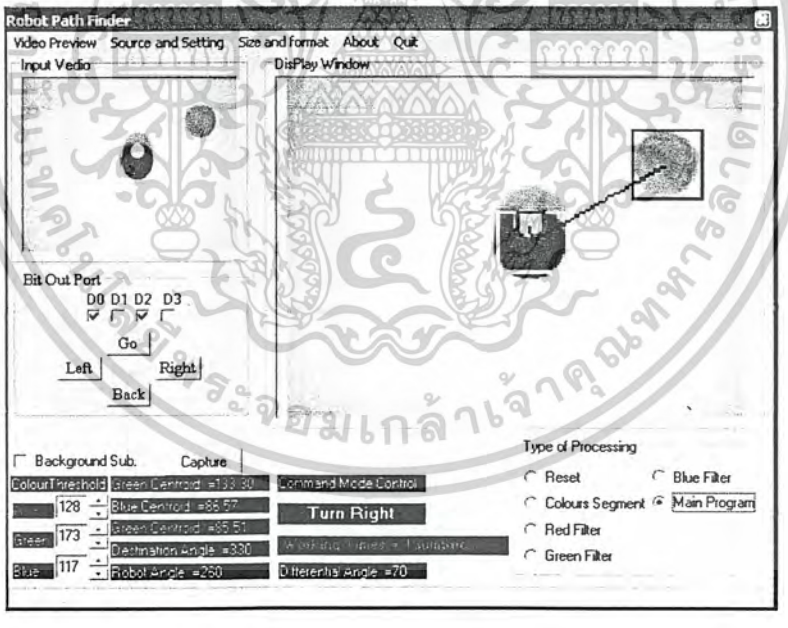
รูป 4.23 โปรแกรมสั่งให้หุ่นหมุนซ้ายเมื่อผลต่างมุมคือ -224 องศา



รูป 4.24 โปรแกรมสั่งให้หุ่นหมุนซ้ายเมื่อผลต่างมุมคือ -116 องศา



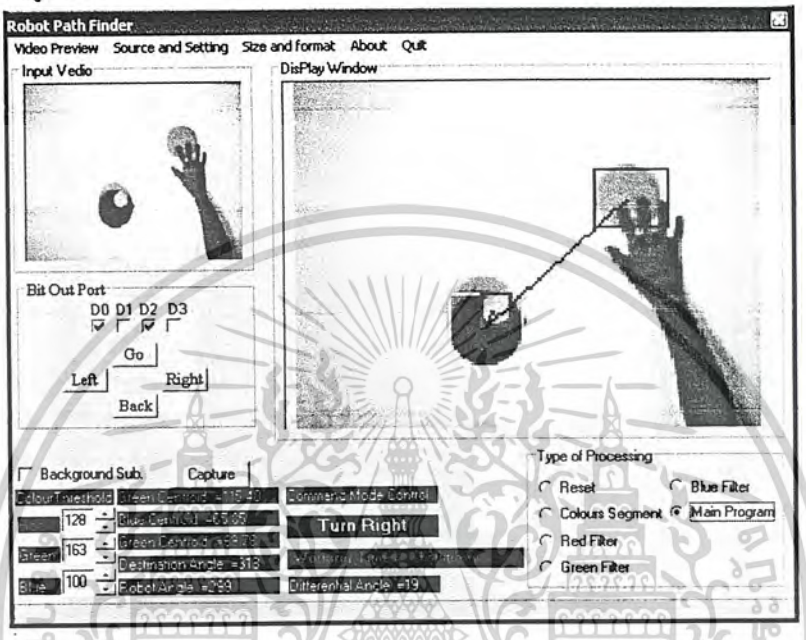
รูป 4.25 โปรแกรมสั่งให้หุ่นหมุนซ้ายเมื่อผลต่างมุมคือ -50 องศา



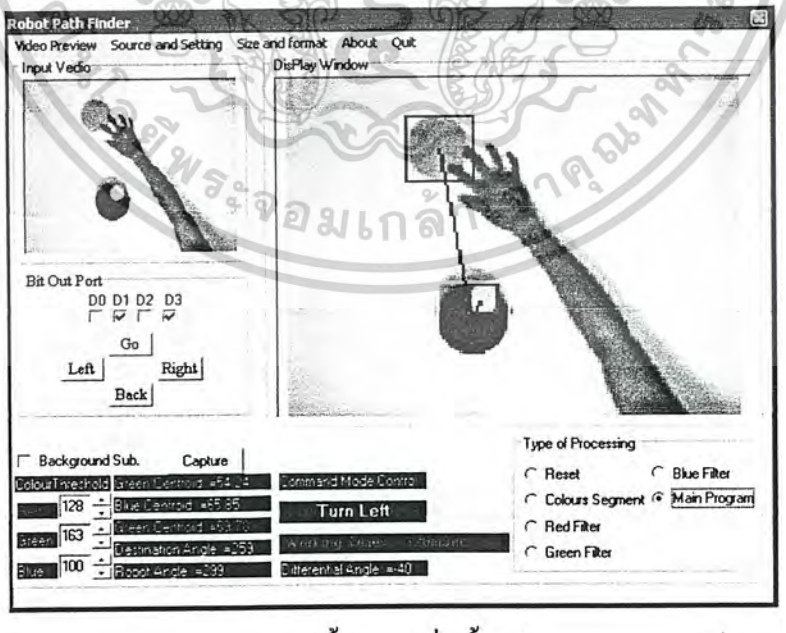
รูป 4.26 โปรแกรมสั่งให้หุ่นหมุนขวาเมื่อผลต่างมุมคือ +70 องศา

4.1.6 การติดตามวัตถุ(Object Tacking)

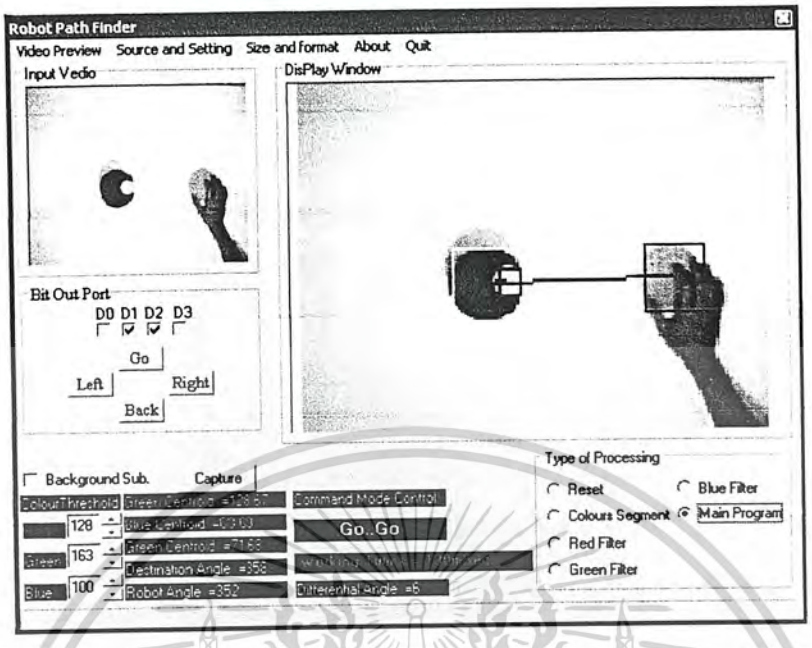
หลังจากที่เราได้ทำการทดลองหาว่าโปรแกรมจะสั่งงานอย่างไรเมื่อให้วัตถุเคลื่อนที่รอบหุ่น (โดยที่หุ่นหยุดนิ่ง) ต่อไปเราจะแสดงผลการติดตามวัตถุโดยปล่อยให้หุ่นเคลื่อนที่เข้าหาวัตถุอย่างอัตโนมัติ แสดงรูปที่ต่อไปนี้



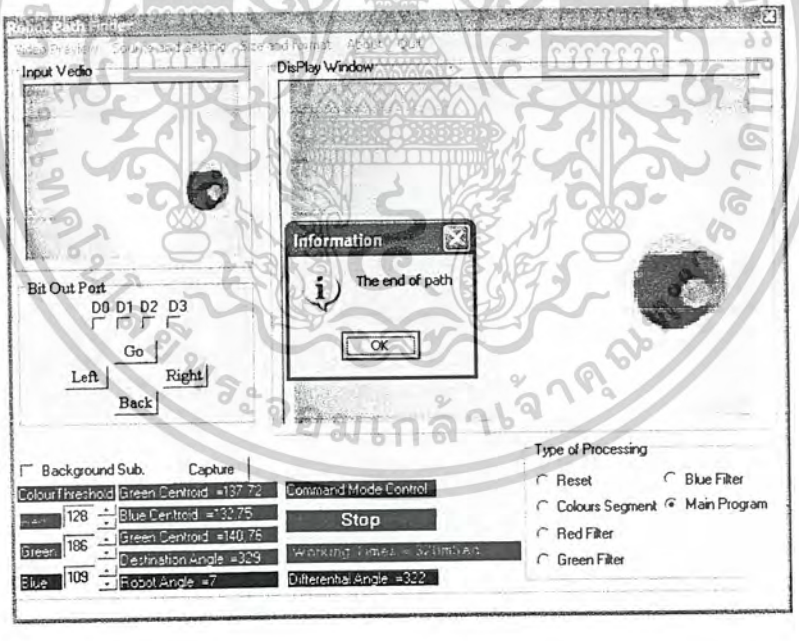
รูป 4.27 แสดงการแทรกกิ้งและคำสั่งเลี้ยวขวาควบคุมหุ่นยนต์



รูป 4.28 แสดงการแทรกกิ้งและคำสั่งเลี้ยวซ้ายควบคุมหุ่นยนต์



รูป 4.29 แสดงการแทรกกิ้งและคำสั่งเดินหน้าควบคุมหุ่นยนต์



รูป 4.30 แสดงข้อความเมื่อหุ่นยนต์มาถึงจุดหมายปลายทางและคำสั่งหยุด

4.2 ผลการทดลอง

4.2.1 ผลการทดลองการลบพื้นหลัง

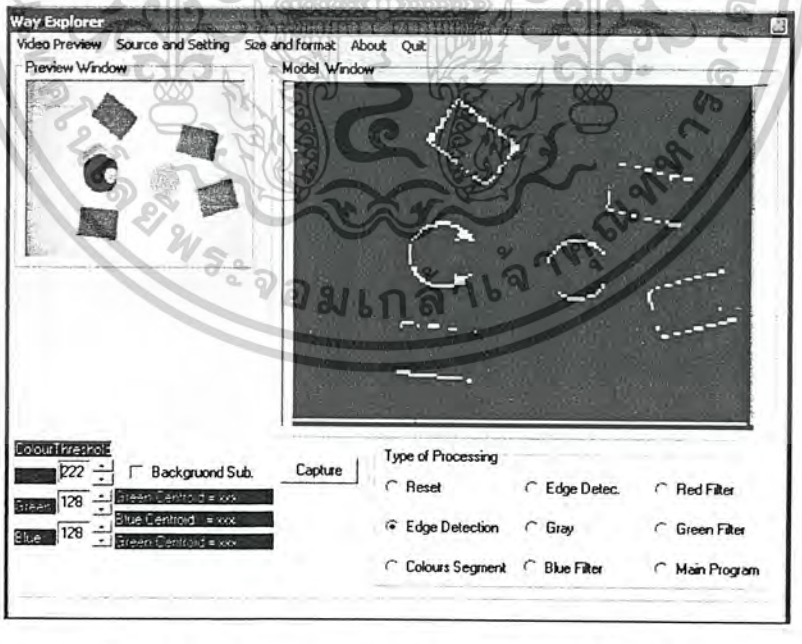
จากผลการทดลองการลบพื้นหลังในรูป 4.1 ถึง 4.3 จะเห็นว่าสามารถแก้ไขปัญหาการเกิดเงาที่บริเวณขอบของฉากพื้นหลังได้อย่างดี แต่ว่าผลการทำงานก็จะส่งผลต่อความสว่างของตัววัตถุที่อยู่ในภาพให้มีความสว่างที่ลดลงจากเดิมเล็กน้อย เมื่อเปรียบเทียบกับภาพก่อนการประมวลผล

4.2.2 ผลการทดลองการหาขอบภาพ

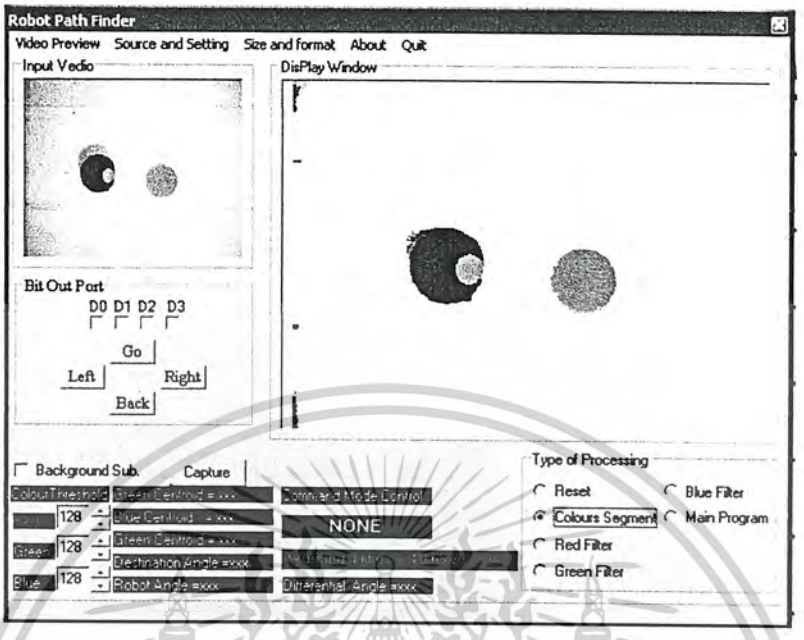
จากผลการหาขอบภาพในรูป 4.4 จะเห็นว่าขอบของภาพที่ได้มีขนาดที่ค่อนข้างใหญ่มาก เนื่องจากการเลือกใช้ค่าเทรชโฮลด์ที่ไม่เหมาะสมเท่าที่ควร(ค่าที่ใช้ในที่นี้คือ 128) ถ้าหากต้องการให้การหาขอบภาพมีขนาดที่เล็กลง(ความผิดพลาดน้อยลง) ก็สามารถทำได้โดยการเพิ่มระดับค่าเทรชโฮลด์ให้มากขึ้นดังรูปที่ 4.31

4.2.4 ผลการทดลองการแยกสี

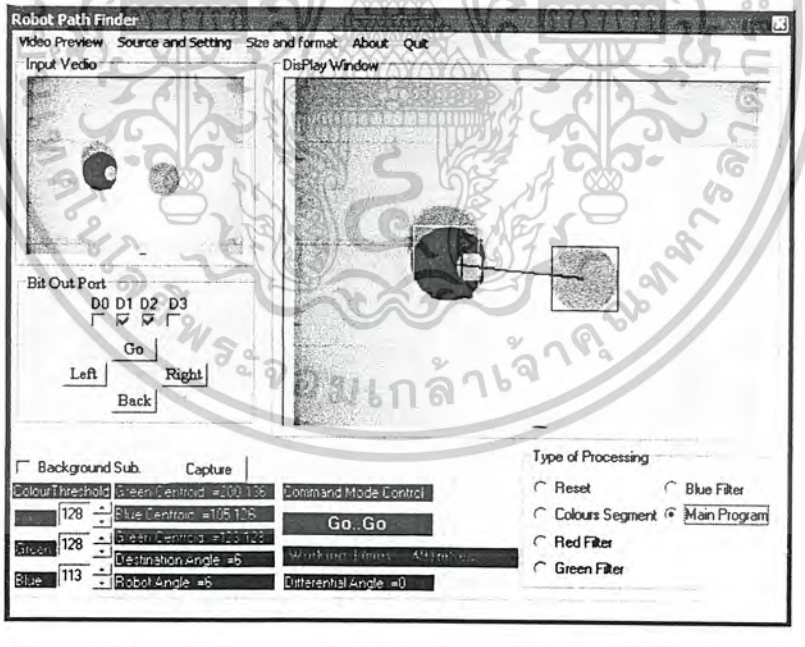
การแยกสีในรูปที่ 4.5 ถึง 4.7 แม้จะมีผลของสัญญาณรบกวนที่เกิดขึ้นหลังจากการแยกสีแต่ก็สามรถถูกกำจัดออกไปได้โดยการกรองสีในโปรแกรมย่อยที่อยู่ในลำดับต่อ ซึ่งถือว่าโปรแกรมใช้เวลาทำงานได้รวดเร็วมากที่ความละเอียด 160x120 ใช้ความเร็ว เท่ากับ 0 millisecond (วัดค่าไม่ได้ที่ฐานเวลา 1 มิลลิเซคคัน)และที่ความละเอียดภาพที่ 320x240 ใช้ความเร็ว 10 millisecond ดังรูป 4.32



รูป 4.31 การหาขอบภาพ โดยเลือกค่าเทรชโฮลด์เท่ากับ 222



รูป 4.32 แสดงการทำการแยกสี ที่ความละเอียดภาพ 320 x 240



รูป 4.33 เมื่อทดลองเพิ่มความละเอียดภาพเป็น 320 x 240

4.2.5 ผลการทดลองการกรองสีและหาตำแหน่งวัตถุ

การกรองสีและหาตำแหน่งวัตถุในรูป 4.8 ถึง 4.10 ในการกรองสีในภาพแต่ละสีนั้นสามารถทำการกรองเฉพาะสีที่ต้องการอกกมาได้ค่อนข้างได้ผลเป็นเป็นที่น่าพอใจแม้ว่าจะมีสัญญาณรบกวรหลุดลอดออกมาปะบนกับภาพแต่ก็ไม่เป็นปัญญหาในการหาตำแหน่งวัตถุของโปรแกรมแต่อย่างใด ถ้าหากว่าขนาดของกลุ่มสัญญาณรบกวนมีขนาดน้อยกว่าขนาดของวัตถุ เวลาที่ใช้ในการกรองสีแต่ละสีนั้นสังเกตว่าจะใช้เวลาการทำงานที่ไม่เท่ากัน เพราะว่า เงื่อนไขที่สีแต่ละสีต้องทำนั้นต่างกัน

4.2.6 ผลการทดลองควบคุมการหมุนของหุ่นยนต์

ในการทดลองควบคุมหมุนของตัวหุ่นยนต์ดังรูป 4.11 ถึง 4.26 จะกำหนดให้หุ่นหยุดอยู่กับที่แม้ว่าจะมีการสั่งงานจากคอมพิวเตอร์ก็ตาม แล้วทำการทดลองเลื่อนจุดปลายทางไปรอบบริเวณตัวหุ่นแล้วดูว่าโปรแกรมจะสั่งให้หุ่นหมุนไปทิศทางใด ปรากฏว่าการสั่งงานยังได้ดีเท่าที่ควรคือ มีปัญหาในกรณีที่ตัวหุ่นและตัววัตถุทำมุมที่มากกว่า 180 องศา(วัดมุมทวนเข็มนาฬิกา)กับแกนนอนซึ่งก็คือแทนที่จะสั่งให้หุ่นเลี้ยวซ้าย เช่นในรูป 4.13 และ 4.14 แต่กลับเลี้ยวขวา ซึ่งเป็นจุดที่ต้องทำการแก้ไขในโปรแกรมให้มีการตรวจสอบเงื่อนไขให้มากขึ้นก็จะสามารถแก้เหตุการณ์นี้ได้

4.2.7 ผลการทดลองการติดตามวัตถุ

รูป 4.27 ถึง 4.30 แสดงการแทรกกิ้งหรือว่าการติดตามวัตถุโดยที่จะปล่อยให้หุ่นยนต์ทำการวิ่งตามวัตถุที่เคลื่อนที่ซึ่งแม้ว่าจากผลการทดลองที่ 4.2.6 อาจดูว่าจะทำให้การติดตามวัตถุนั้นไม่มีประสิทธิภาพแต่นั้นเป็นเพียงเหตุการณ์ที่จะเกิดขึ้นเพียงครั้งเดียวคือตอนที่โปรแกรมถูกรันท์ขึ้นมาในตอนแรกเท่านั้น ซึ่งเมื่อตัวหุ่นได้หันหน้าเข้าหาวัตถุแล้วการแทรกกิ้งก็จะเป็นไปได้ด้วยดีดังรูปที่ได้แสดงไว้

บทที่ 5
สรุปและวิจารณ์ผล

5.1 การทดลองโปรแกรมประมวลผลภาพ

จากการทดลองเขียนโปรแกรมประมวลผลภาพที่ได้ทดลองไปแล้วในบทที่ 4 นั้นก็เพื่อต้องการแสดงให้เห็นว่ามีกี่ขั้นตอนในการออกแบบโปรแกรมและเกิดผลอย่างไรบ้างเมื่อโปรแกรมทำงาน ซึ่งในการเขียนโปรแกรมแล้วจำเป็นอย่างยิ่งที่จะต้องทำการพิสูจน์ว่าโปรแกรมหรือวิธีการที่คิดขึ้นมานั้นสามารถทำงานได้อย่างที่ตั้งใจหรือไม่ ซึ่งเป็นส่วนที่สำคัญในการพัฒนาโปรแกรมต่างๆ

ในการทดลองที่แสดงในบทที่ 4 ที่ผ่านมานั้นวิธีการลบพื้นหลังหรือว่า (Background Subtraction)และการหาขอบภาพ(Edge Detection) นั้นไม่ได้นำมาใช้งานร่วมในโครงงานก็เพราะวิธีการลบเงาพื้นหลังนั้นหลังจากเพิ่มเข้าไปแล้วทำให้ภาพที่ได้ผิดพลาดไปจากที่ต้องการหลังจากนำภาพนั้นไปทำการแยกสี ดังนั้นวิธีการลบพื้นหลังจะไม่นำมาใช้ร่วมกับโปรแกรมย่อยอื่นๆ ส่วนการหาขอบภาพนั้น ก็เป็นเพราะว่าไม่จำเป็นต้องใช้ในวิธีการติดตามวัตถุแบบกรองสี แต่ก็ได้นำมาแสดงไว้เผื่อว่ารุ่นน้องจะนำแนวคิดที่ได้ไปพัฒนาให้ดีขึ้นต่อไป

ในหัวข้อที่ 4.1.3 เป็นต้นมาจะเป็นขั้นตอนการทดลองส่วนของโปรแกรมย่อยต่างๆที่ได้นำมาใช้งานในโปรแกรมประมวลผลภาพซึ่งก็ถือว่าตัวโปรแกรมนั้นสามารถทำงานได้เป็นอย่างดีไม่ว่าจะเป็นส่วนของโปรแกรมการแยกสี , โปรแกรมกรองสีและหาตำแหน่งวัตถุ , และสุดท้ายโปรแกรมควบคุมหุ่นและการตรวจจับวัตถุ ซึ่งผลที่ได้ทดลองออกมานับว่าดีมากทีเดียว

5.2 ปัญหาและแนวทางการแก้ไข

ปัญหาที่พบในการเขียนและการทำงานของโปรแกรมในส่วนของการประมวลผลภาพนั้นสามารถแยกออกเป็นปัญหาหลังใหญ่ๆดังนี้คือ ปัญหาจากการเขียนโปรแกรม และ5.2.2 ปัญหาจากแสงรบกวนภายนอกที่มีผลต้อการทำงานของโปรแกรม โดยสามารถแสดงให้เห็นดังต่อไปนี้

5.2.1 ปัญหาจากการเขียนโปรแกรม

เนื่องจากการเขียนโปรแกรมด้วยภาษา ซี++ บิลด์เดอร์ สำหรับการสร้างโครงงานนี้นั้นถือว่าเป็นการเขียนโปรแกรมที่ทำงานบนระบบวินโดวส์ครั้งแรกในชีวิตการเรียน ถึงแม้ว่าผู้สร้างโครงงานเองจะมีพื้นฐานของ การเขียนโปรแกรม ภาษาซีบน ดอส แต่กว่าจะเขียนโปรแกรมให้ทำงานได้เข้าที่เข้าทางก็ต้องใช้เวลานานหลายอาทิตย์ ต่อไปจะเป็นปัญหาที่เกิดขึ้นในการเขียนโปรแกรม

ปัญหาที่ 1 .

-ปัญหาที่พบหลังจากการทดสอบการทำงานของโปรแกรมครั้งแรกคือเรื่องของการเข้าถึงเข้ามูลรูปภาพโดยตรงจากใช้คำสั่ง Canvas ซึ่งใช้เวลาในการเข้าถึงข้อมูลภาพได้ช้ามากๆซึ่งในอิมเมจโปรเซสซิ่งจำเป็นต้องใช้เวลาในการประมวลผลให้น้อยที่สุดเท่าที่จะทำได้เพื่อว่าขณะที่โปรแกรมสั่งให้เคลื่อนที่แล้วขณะเดียวกันก็ตรวจหาตำแหน่งของหุ่นยนต์ตามไปด้วย หากว่าโปรแกรมใช้เวลาในการประมวลผลที่นานเกิน ไปจะทำให้การระบุตำแหน่งของตัวหุ่นยนต์ผิดพลาดได้

การแก้ไขปัญหาที่ 1.

- เปลี่ยนคำสั่งจากการเข้าถึงข้อมูลโดยตรงด้วยคำสั่ง Canvas เปลี่ยนเป็นการใช้คำสั่งการเข้าถึงข้อมูลทางอ้อมคือคำสั่ง Scanline ซึ่งเป็นการอ้างตำแหน่งข้อมูลแบบพอนย์เตอร์และเก็บข้อมูลในหน่วยความจำแทนการเก็บข้อมูลไว้ในอาเรย์ซึ่งทำให้การทำงานโดยรวมของระบบเร็วขึ้นมาก

ปัญหาที่ 2.

- การที่ใช้โปรแกรม ซี++ บิลด์เดอร์ให้ทำงานบน ระบบวินโดวส์ เอ็กซ์พี นั้นจะมีปัญหาอยู่(เฉพาะที่เจอตอนนี้)เกี่ยวกับการเขียนโปรแกรมสั่งงานติดต่อกับพอร์ตของเครื่องคอมพิวเตอร์ซึ่งเจ้าตัววินฯเอ็กซ์พีนั้นไม่ยอมให้ ซี++ บิลด์เดอร์ ติดต่อกับพอร์ตดังกล่าวได้โดยตรงโดยใช้แอสเซมบลี่(ไม่พบปัญหานี้ในวินโดวส์ 95-98 และโปรแกรมอื่นที่ไม่ใช้ ซี++บิลด์เดอร์ บน เอ็กซ์พี) จะต้องทำการติดต่อผ่านคอมโพเนนต์(ที่ต้องไปหาโหลดในอินเตอร์เน็ตให้ลำบาก) ซึ่งเป็นการไม่สะดวกในการเสาะหาและเสียเวลาอย่างมากเพราะเป็นรุ่นทดลองใช้30วัน

การแก้ไขปัญหาที่ 2.

- ทำการฟอร์แมตฮารด์ดิสก์คอมพิวเตอร์ที่ได้ลงคอมโพเน็นนั้นไปแล้วใหม่(ก็จะใช้ได้อีก30วัน) หรือ ย้ายคอมโพเน็นไปลงเครื่องอื่น หรือไปซื้อเวอร์ชันเต็มมาใช้งาน หรือวิธีสุดท้ายก็คือ เขียนไดรว์เวอร์ของพอร์ตที่จะต้องการติดต่อขึ้นมาเอง

ปัญหาที่ 3.

-ในอันที่จริงแล้วโครงงานชุดนี้เดิมที่มีชื่อว่าหุ่นยนต์ค้นหาเส้นทางอัตโนมัติแต่เหตุที่ต้องทำการเปลี่ยนแปลงเป็นชื่อหุ่นยนต์ตรวจจับวัตถุ(Object Tracking Robot) ก็เพราะว่าวิธีที่ใช้ในการคิดหาเส้นทาง(ดูภาคผนวก)นั้นผิดตั้งแต่ตอนแรกเลยเพราะถึงแม้ว่าแล้วคิดในการหาเส้นทางให้หุ่นยนต์วิ่งหลบหลีกวัตถุที่ได้คิดไว้ในภาคผนวกนั้นจะมีแนวโน้มเป็นไปได้ในการหาเส้นทางก็ตามที่เมื่อเมื่อมาทดลองถอดอัลกอริทึ่ม(Algorithm)เพื่อจะเขียนโปรแกรมปรากฏว่าโปรแกรมมีความซับซ้อนมากเกินไปที่จะทำให้หุ่นยนต์สามารถหาเส้นทางที่จากกองวัตถุที่เป็นสีแดงได้เหตุที่เป็นเช่นนี้ก็เพราะ วิธีที่คิดไว้แต่แรกไม่สามารถหาตำแหน่งของวัตถุที่เป็นสีแดงทุกก่อนได้เหมือนกับการหาตำแหน่งของวัตถุสีอื่นๆ จึงทำให้การคิดหาเส้นทางที่คิดไว้ในตอนแรกใช้ไม่ได้ในทางปฏิบัติ

การแก้ไขปัญหาที่ 3.

- การแก้ไขปัญหาที่เกิดขึ้นจะต้องทำการระบุทุกตำแหน่งของวัตถุสีแดงแต่ละก้อนให้ได้เสียก่อนซึ่งวิธีการก็คือ ต้องทำการแยกส่วนภาพในส่วนของโปรแกรมย่อยการกรองสีแดงซึ่งวิธีการแยกส่วนภาพที่ว่านี้จะต้อใช้วิธีที่แตกต่างกับวิธีที่ใช้ในโครงงานซึ่งอาจจะเป็นวิธีการของ Hausdarff Transform[6] เป็นต้นจึงจะสามารถทำการจัดการกับกลุ่มของวัตถุสีแดงที่อยู่ในภาพให้สามารระบุตำแหน่งของแต่ละจุดได้

5.2.2 ปัญหาจากแสงรบกวนภายนอก

ในระบบที่ต้องใช้การประมวลภาพนั้นจำเป็นอย่างยิ่งที่จะต้องให้มีแสงที่กล้องจับเข้ามานั้นเพียงพอหรือเสม่ำเสมอเพราะไม่เช่นนั้นแล้วภาพที่ได้อาจเกิดเหตุการณ์ของแสงที่มากเกินไปหรือน้อยเกินไปจะทำให้การทำงานของระบบผิดพลาดได้

---

6 Hausdarff Transform คือหนึ่งในวิธีการของ การแปลงเชิงระยะทาง (Distance Transform)

ปัญหาที่ 4.

- การทดลองในตอนแรกได้มีการใช้เพียงแสงจากภายนอกหรือหลอดไฟที่อยู่ภายในห้องเพียงอย่างเดียวผลที่ได้คือกล้องจับภาพออกมานั้นยังมืดเกินไปแต่ผู้ทดลองก็ได้ทำการปรับแต่งค่าพารามิเตอร์ต่างที่สามารถปรับได้จากกล้องซึ่งก็ทำให้แก้ไขสถานะการณ์เวลานั้นมาได้ แต่พอมาทำการทดลองอีกวันหนึ่งซึ่งอยู่ภายในห้องเดียวกันกับที่ได้ทดลองและปรับแต่งกล้องไปแล้วทีนี้เกิดแสงที่ได้มานั้นสว่างมากเกินไปจนทำให้ไม่สามารถมองเห็นวัตถุจากกล้องได้ซึ่งทำให้เห็นปัญหาที่เกิดขึ้นว่าจะต้องมีการควบคุมสภาวะแสงภายในสนามที่ใช้ทดสอบให้คงที่จึงจะสามารถแก้ไขปัญหานี้ได้

การแก้ไขปัญหาที่ 4.

- ทำการติดตั้งหลอดไฟเพื่อควบคุมแสงภายในสนามให้เหมาะสม

ปัญหาที่ 5.

- ปัญหาสุดท้ายที่จะกล่าวต่อไปนี้คือ ความไม่แน่นอนของกล้องที่มีการปรับแสงภายในตัวเองแบบอัตโนมัติ ซึ่งอันนี้ก็มีปัญหาอย่างมากเช่นกันเพราะว่าในตอนแรกที่ได้ทดลองโปรแกรมจะเห็นอีกว่าแสงที่ได้มีการเปลี่ยนแปลงอย่างไม่คงที่และสีที่กล้องจับออกมาได้ก็ไม่ถูกต้องเท่าที่ควร ซึ่งทำให้ต้องหาสาเหตุอยู่นานกว่าจะหาทางแก้ได้

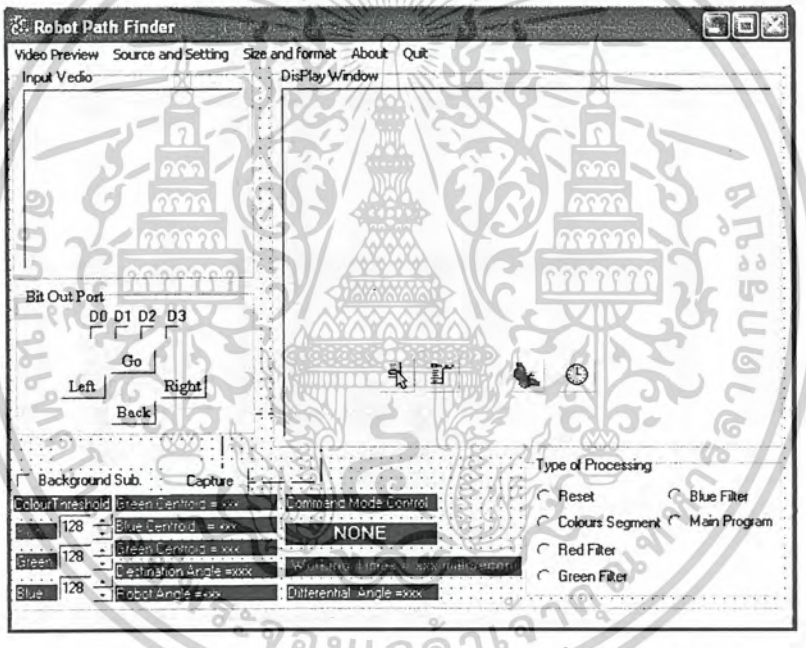
การแก้ไขปัญหาที่ 5.

- ทำการยกเลิกการปรับแสงแบบอัตโนมัติจากซอร์ฟแวร์ไดร์เวอร์ของกล้อง

หนังสืออ้างอิง

1. ไพโรจน์ ไววานิชกิจ , "เปิดโลกการสื่อสารไร้สาย" , ซีเอ็ดยูเคชั่น , 245 หน้า , 2540
2. สุนทร วิทูสุรพจน์ , "การใช้งานไมโครคอนโทรลเลอร์ตระกูล 8051" , ซีเอ็ดยูเคชั่น , 180 หน้า , 2537
3. สมยศ จุณณะปิยะ, "การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ ตระกูล MCS51", ซีเอ็ดยูเคชั่น ,325 หน้า , 2540
4. Adrian Low. ," Introductory computer vision and image processing ",McGRAW-HILL, 243 p.,1992.
5. Yang GZ and Gillies DF ., "Computer Vision", Department of Computing, Imperial College.
6. Jarrod , H. "C++Builder™ 5 Developer's Guide ", SAMS Publishing, 2001, 1472-1478.
7. http://www.doc.ic.ac.uk/~gzy. ,"Image Processing and Edge Detection"
8. http://www.eng.monash.edu.au., "Fast Robust Colour Vision for the Monash Humanoid"

ภาคผนวก



แสดง ฟอร์มและคอมโพเนนต์ของโปรแกรมที่ใช้ในโครงงาน

```
//---------------------------------------------------------------------
//                                  Video Camera Setting
// Whiteness=13;Brightness=0;Exposure=auto;Contrast=255;Satulation=2
//---------------------------------------------------------------------
#include <vcl\vcl.h>
#pragma hdrstop

#include "AboutBox1.h"
#include "Math.h"
#include "ImageProc1.h"
#include "systdate.h"
#include <shellapi.h>
//---------------------------------------------------------------------
#pragma link "Grids"
#pragma link "ievect"
#pragma link "ieview"
#pragma link "imageenproc"
#pragma link "imageenview"
#pragma link "videocap"
#pragma resource "*.dfm"

#include "..\tvichw32.h"

TCheckBox *CPinWrite[5];
BOOL      ActiveHW = FALSE;
short     nPort;
HANDLE    HW32;

TForm1 *Form1;
Byte ForbiArea[160][120],PixSlope[160][120];
int GctX,GctY,BctX,BctY,BGctX,BGctY,StrX=0,StrY=0,StpX=0,StpY=0;
int StartX=0,StartY=0,StopX=0,StopY=0,incre=0;
int Zeta1=0,Zeta2=0,SStartX=0,SStopX=0,SStartY=0,SStopY=0,BStartX=2,
unsigned long StpTime,StrTime;

float Slope=0;
double temp=0,tmpX,tmpY,line=0;
bool null=false;

//---------------------------------------------------------------------
//                                  Function
//---------------------------------------------------------------------
void Capture(void)
{
        Form1->Image1->Picture->Bitmap->Assign(Form1->ImageEnView1-
}

void Sub(void)
{
        TRect r=Form1->ImageEnVideoView1->GetVideoSize();
        TColor PixelBuf2,PixelBuf1;
   for (int y=0; y<=(int(r.Bottom)); y++)
   {
        byte* LinePtr2=(byte*)Form1->Image1->Picture->Bitmap->ScanL
        byte* LinePtr1=(byte*)Form1->ImageEnView1->Bitmap->ScanLine
```

```
          for (int x=0; x<=(int(r.Right))*3; x++)
              LinePtr1[x] =255-(abs( LinePtr2[x] - LinePtr1[x]));
    }
    Form1->ImageEnView1->Refresh();
}

//---------------------------------------------------------------------
void PixelVec(void)
{
              /* Find Pixel on Slope path vector*/
             //Find Reference Slope
              TRect r=Form1->ImageEnVideoView1->GetVideoSize();

              // First && Second Quadrant for Slope Unvaluable
            if((GctX-BctX)==0 && (GctY>BctY))
              {
                for(int y=0; y<=int(r.Bottom);y++)
                    for(int x=0;x<=int(r.Right);x++)
                    {
                      if(y>=BctY && GctY>=y)
                        {PixSlope[x][y]=255;}
                       else{PixSlope[x][y]=0;}

                    }
              }
              // Third && Fourth Quadrant for Slope Unvaluable
            else if((GctX-BctX)==0 && (GctY<BctY))
              {
                for(int y=0; y<=int(r.Bottom);y++)
                    for(int x=0;x<=int(r.Right);x++)
                    {
                     if(y<=BctY && GctY<=y)
                       {PixSlope[x][y]=255;}
                      else{PixSlope[x][y]=0;}

                    }
              }
              // Rigth Plane Quadrant for Slope equal to Zero
            else if((GctY-BctY)==0 && (GctX>BctX))
              {
                for(int y=0; y<=int(r.Bottom);y++)
                    for(int x=0;x<=int(r.Right);x++)
                    {
                      if(x>=BctX && GctX>=x)
                         {PixSlope[x][y]=255;}
                      else{PixSlope[x][y]=0;}

                    }
              }
              // Left Plane Quadrant for Slope equal to Zero
            else if((GctY-BctY)==0 && (GctX<BctX))
              {
                for(int y=0; y<=int(r.Bottom);y++)
                    for(int x=0;x<=int(r.Right);x++)
                    {
```

```
                if(x<=BctX && GctX<=x)
                   {PixSlope[x][y]=255;}
                else{PixSlope[x][y]=0;}

            }
        }

      // Any Slope != Infinity or Zero
    if((GctX-BctX)!=0 && (GctY-BctY)!=0)
        {
          Slope=(GctY-BctY)/(GctX-BctX);
        }

      // First Quadrant (+,+);Slp=+
    if((GctX > BctX) && (GctY > BctY))
        {
          for(int y=0; y<=int(r.Bottom);y++)
              for(int x=0;x<=int(r.Right);x++)
              {
                if(x>BctX && x<GctX && y<GctY && y>BctY)
                  {
                    float Slp=(y-BctY)/(x-BctX);
                    if(Slope==Slp)
                      {PixSlope[x][y]=255;}
                    else{PixSlope[x][y]=0;}
                  }
                else{PixSlope[x][y]=0;}
              }
        }

      // Second Quadrant (-,+);Slp=-
    else if((GctX < BctX) && (GctY > BctY))
        {
          for(int y=0; y<=int(r.Bottom);y++)
              for(int x=0;x<=int(r.Right);x++)
              {
                if(x<BctX && x>GctX && y>GctY && y<BctY)
                  {
                    float Slp=(y-BctY)/(x-BctX);
                    if(Slope==Slp)
                      {PixSlope[x][y]=255;}
                    else{PixSlope[x][y]=0;}
                  }
                else{PixSlope[x][y]=0;}
              }
        }

      //  Third Quadrant (-,-);Slp=+
    else if((GctX < BctX) && (GctY < BctY))
        {
          for(int y=0; y<=int(r.Bottom);y++)
              for(int x=0;x<=int(r.Right);x++)
              {
                if(x<BctX && x>GctX && y<GctY && y>BctY)
                  {
```

```
                                float Slp=(y-BctY)/(x-BctX);
                                if(Slope==Slp)
                                  {PixSlope[x][y]=255;}
                                else{PixSlope[x][y]=0;}
                              }
                            else{PixSlope[x][y]=0;}
                          }
                      }

                    // fourth Quadrant (+,-);Slp=-
                  else if((GctX > BctX) && (GctY < BctY))
                      {
                        for(int y=0; y<=int(r.Bottom);y++)
                            for(int x=0;x<=int(r.Right);x++)
                            {
                              if(x>BctX && x<GctX && y>GctY && y<BctY)
                                {
                                  float Slp=(y-BctY)/(x-BctX);
                                  if(Slope==Slp)
                                    {PixSlope[x][y]=255;}
                                  else{PixSlope[x][y]=0;}
                                }
                              else{PixSlope[x][y]=0;}
                            }
                      }
}
//---------------------------------------------------------------------------

void stop(void)
{
        Form1->WPin2->Checked = false;
        Form1->WPin3->Checked = false;
        Form1->WPin4->Checked = false;
        Form1->WPin5->Checked = false;
}

void go(void)
{
        Form1->WPin2->Checked = false;
        Form1->WPin3->Checked = true;
        Form1->WPin4->Checked = true;
        Form1->WPin5->Checked = false;
}

void back(void)
{
        Form1->WPin2->Checked = true;
        Form1->WPin3->Checked = false;
        Form1->WPin4->Checked = false;
        Form1->WPin5->Checked = true;
}

void turnL(void)
{
        Form1->WPin2->Checked = false;
```

```
        Form1->WPin3->Checked = true;
        Form1->WPin4->Checked = false;
        Form1->WPin5->Checked = true;
}

void turnR(void)
{
        Form1->WPin2->Checked = true;
        Form1->WPin3->Checked = false;
        Form1->WPin4->Checked = true;
        Form1->WPin5->Checked = false;
}

//---------------------------------------------------------------------
void BluePlot(void)
{
                Form1->Label7->Caption="Blue Centroid  ="+IntToStr(Bct
                Form1->ImageEnView1->Bitmap->Canvas->Pen->Color = clYe
                Form1->ImageEnView1->Bitmap->Canvas->MoveTo(BStartX,BS
                Form1->ImageEnView1->Bitmap->Canvas->LineTo(BStopX,BSt
                Form1->ImageEnView1->Bitmap->Canvas->MoveTo(BStopX,BSt
                Form1->ImageEnView1->Bitmap->Canvas->LineTo(BStopX,BSt
                Form1->ImageEnView1->Bitmap->Canvas->MoveTo(BStopX,BSt
                Form1->ImageEnView1->Bitmap->Canvas->LineTo(BStartX,BS
                Form1->ImageEnView1->Bitmap->Canvas->MoveTo(BStartX,BS
                Form1->ImageEnView1->Bitmap->Canvas->LineTo(BStartX,BS

}
//---------------------------------------------------------------------

void RedPlot(void)
{
                Form1->Label6->Caption="Green Centroid  ="+IntToStr(Gc
                Form1->Label7->Caption="Blue Centroid  ="+IntToStr(Bct
                Form1->ImageEnView1->Bitmap->Canvas->Pen->Color = clBl
                Form1->ImageEnView1->Bitmap->Canvas->MoveTo(BctX,BctY)
                Form1->ImageEnView1->Bitmap->Canvas->LineTo(GctX,GctY)
                Form1->ImageEnView1->Bitmap->Canvas->MoveTo(BctX,BctY)
                Form1->ImageEnView1->Bitmap->Canvas->LineTo(SGctX,SGct

}
//---------------------------------------------------------------------

void GreenPlot(void)
{
                Form1->Label6->Caption="Green Centroid  ="+IntToStr(Gc
                Form1->ImageEnView1->Bitmap->Canvas->Pen->Color = clRe
                Form1->ImageEnView1->Bitmap->Canvas->MoveTo(StartX,Sta
                Form1->ImageEnView1->Bitmap->Canvas->LineTo(StopX,Star
                Form1->ImageEnView1->Bitmap->Canvas->MoveTo(StopX,Star
                Form1->ImageEnView1->Bitmap->Canvas->LineTo(StopX,Stop
                Form1->ImageEnView1->Bitmap->Canvas->MoveTo(StopX,Stop
                Form1->ImageEnView1->Bitmap->Canvas->LineTo(StartX,Sto
                Form1->ImageEnView1->Bitmap->Canvas->MoveTo(StartX,Sto
                Form1->ImageEnView1->Bitmap->Canvas->LineTo(StartX,Sta

                Form1->Label8->Caption="Green Centroid  ="+IntToStr(SG
```

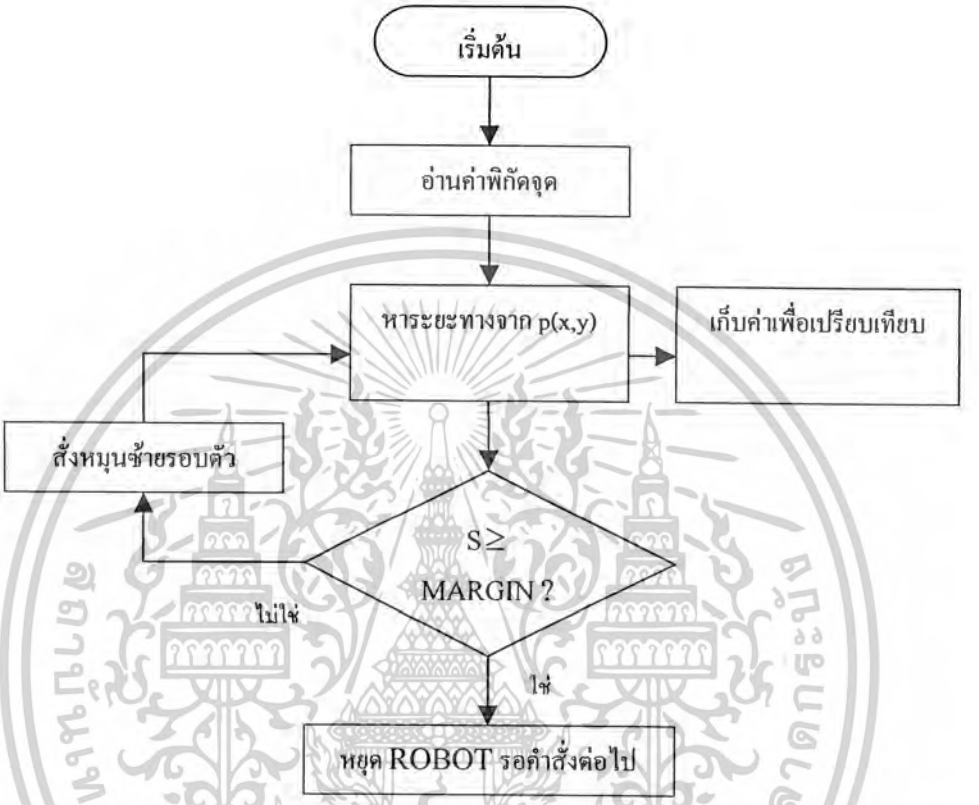
```
                Form1->ImageEnView1->Bitmap->Canvas->MoveTo(SStartX,SS
                Form1->ImageEnView1->Bitmap->Canvas->LineTo(SStopX,SSt
                Form1->ImageEnView1->Bitmap->Canvas->MoveTo(SStopX,SSt
                Form1->ImageEnView1->Bitmap->Canvas->LineTo(SStopX,SSt
                Form1->ImageEnView1->Bitmap->Canvas->MoveTo(SStopX,SSt
                Form1->ImageEnView1->Bitmap->Canvas->LineTo(SStartX,SS
                Form1->ImageEnView1->Bitmap->Canvas->MoveTo(SStartX,SS
                Form1->ImageEnView1->Bitmap->Canvas->LineTo(SStartX,SS
}
//---------------------------------------------------------------------
void StrTimer(void)
{
            StrTime = GetTickCount();      //GetTick. is func. Retrie
}

void StpTimer(void)
{
            StpTime = GetTickCount();
            Form1->Label14->Caption=" Working Times = "+IntToStr(Stp'
}
//---------------------------------------------------------------------
float ZetaValue(int GctXs,int GctYs,int BctXs,int BctYs )
{
           float ZetaBuf=0;

           if((GctXs-BctXs)==0 && (GctYs>BctYs))
             {ZetaBuf=90;}
           else if((GctYs-BctYs)==0 && (GctXs>BctXs))
             {ZetaBuf=0;}
           else if((GctXs-BctXs)==0 && (GctYs<BctYs))
             {ZetaBuf=270;}
           else if((GctYs-BctYs)==0 && (GctXs<BctXs))
             {ZetaBuf=180;}
             // fourth Quadrant (+,-);Slp=-
           else if((GctXs > BctXs) && (GctYs < BctYs))
             {
               ZetaBuf=atan2((GctYs-BctYs),(GctXs-BctXs));
               ZetaBuf=ZetaBuf*57.3;
               ZetaBuf=ZetaBuf+360;
             }
            //  Third Quadrant (-,-);Slp=+
           else if((GctXs < BctXs) && (GctYs < BctYs))
             {
               ZetaBuf=atan2((GctYs-BctYs),(GctXs-BctXs));
               ZetaBuf=ZetaBuf*57.3;
               ZetaBuf=ZetaBuf+360;
             }
            // Second Quadrant (-,+);Slp=-
           else if((GctXs < BctXs) && (GctYs > BctYs))
             {
               ZetaBuf=atan2((GctYs-BctYs),(GctXs-BctXs));
               ZetaBuf=ZetaBuf*57.3;
             }
            // First Quadrant (+,+);Slp=+
           else if((GctXs > BctXs) && (GctYs > BctYs))
```

```
                {
                  ZetaBuf=atan2((GctYs-BctYs),(GctXs-BctXs));
                  ZetaBuf=ZetaBuf*57.3;
                }
        return(ZetaBuf);
}

//---------------------------------------------------------------------

void BlueFunct(void)
{               /*---Colour Threshold---*/
              int lengthX=40,lengthY=40;
              StrX=0,StrY=0,StpX=0,StpY=0;
              TColor PixelBuf2,PixelBuf1;
              TRect r=Form1->ImageEnVideoView1->GetVideoSize();
              int w = StrToInt(Form1->Edit3->Text);
              for (int y=0; y<=(int(r.Bottom)); y++)
                  {
                    byte* LinePtr=(byte*)Form1->ImageEnView1->Bitmap
                    for (int x=0; x<=(int(r.Right)*3); x++)
                        {
                         if (LinePtr[x] > w)
                            LinePtr[x] = 255;
                         else
                            LinePtr[x] = 0 ;

                         byte* Pixel=LinePtr+x*3;
                        if(Pixel[0]>128 && Pixel[1]<100 && Pixel[2]<100
                        {Pixel[0]=255;Pixel[1]=0;Pixel[2]=0;}
//                                              White
                        if(Pixel[0]>250 && Pixel[1]>250 && Pixel[2]>250
                        || Pixel[0]<100 && Pixel[1]<100 && Pixel[2]>
                        {Pixel[0]=0;Pixel[1]=0;Pixel[2]=0;}
                        }
                  }

              /*---Find an Object Boundary---*/
              //X Bound
              for (int y=0; y<=(int(r.Bottom)-2); y++)
                   {byte* LinePtr=(byte*)Form1->ImageEnView1->Bitmap
                    for (int x=1; x<=(int(r.Right)); x++)
                        {byte* PrePi=LinePtr+((x-1)*3);
                         byte* Pixel=LinePtr+x*3;
                         PixelBuf2=Pixel[0];
                         PixelBuf1=PrePi[0];
                         if(abs(PixelBuf2-PixelBuf1)!=0)
                            {
                            if((-1*(PixelBuf2-PixelBuf1))!= abs(Pixe
                              {StrX=x;}//keep start edge
                            else if((-1*(PixelBuf2-PixelBuf1))== abs
                              {StpX=x; //keep position end of edge
                               if((lengthX<=(StpX-StrX))&&((StpX-Str
                                 {lengthX=(StpX-StrX);BStartX=StrX;B
                              }
                            }
```

```
                    }
                }

            //Y Bound
            int y=1;
            for(int x=0; x<int(r.Right)  ; x++)
                {
                do{   byte* LinePtr=(byte*)Form1->ImageEnView1->B
                      byte* LinePtr2=(byte*)Form1->ImageEnView1->
                      byte* Pixel=LinePtr+x*3;
                      PixelBuf2=Pixel[0];
                      Pixel=LinePtr2+x*3;
                      PixelBuf1=Pixel[0];
                      if(abs(PixelBuf2-PixelBuf1)!=0)
                        {
                        if((-1*(PixelBuf2-PixelBuf1))!= abs(Pixe
                          StrY=y;//keep start edge
                        else if((-1*(PixelBuf2-PixelBuf1))== abs
                          {StpY=y; //keep position end of edge
                          if((lengthY<=(StpY-StrY))&&((StpY-Str
                            {lengthY=(StpY-StrY);BStartY=StrY;B.
                          }
                        }
                      y++;
                    }while(y<=int(r.Bottom)); y=1;
                  }
                  //BluePlot();
}
//---------------------------------------------------------------------

void RedFunct(void)
{
            TColor PixelBuf2,PixelBuf1;
            TRect r=Form1->ImageEnVideoView1->GetVideoSize();
             int w = StrToInt(Form1->Edit1->Text);
             for (int y=0; y<=(int(r.Bottom)); y++)
                {
                 byte* LinePtr=(byte*)Form1->ImageEnView1->Bitma
                 for (int x=0; x<=(int(r.Right)*3); x++)
                    {
                  if (LinePtr[x] > w)
                      LinePtr[x] = 255;
                  else
                      LinePtr[x] = 0 ;

                     byte* Pixel=LinePtr+x*3;
                  if(Pixel[0]<100 && Pixel[1]<100 && Pixel[2]>180
                      {Pixel[0]=0;Pixel[1]=0;Pixel[2]=255;}
//                                      White
                  if(Pixel[0]>250 && Pixel[1]>250 && Pixel[2]>250
                     || Pixel[0]>200 && Pixel[1]<100 && Pixel[2]<
                      {Pixel[0]=0;Pixel[1]=0;Pixel[2]=0;}

                     }
                }
```

```
                /*---Find forbidden area of red pixel--*/
              /*   for (int y=0; y<=(int(r.Bottom)); y++)
                   {byte* LinePtr=(byte*)Form1->ImageEnView1->Bitma
                   for (int x=0; x<=(int(r.Right)); x++)
                       {
                        byte* Pixel=LinePtr+x*3;
                        PixelBuf1=Pixel[2];
                        if(PixelBuf1!=0)
                           {ForbiArea[x][y]=255;}
                        else if(PixelBuf1==0)
                           {ForbiArea[x][y]=0;}
                       }
                   }  */
                  // RedPlot();
}
//---------------------------------------------------------------------
void GreenFunct(void)
{
             int lengthX=40,lengthY=40,SmallX=14,SmallY=14;
             TColor PixelBuf2,PixelBuf1;
             TRect r=Form1->ImageEnVideoView1->GetVideoSize();
              int w = StrToInt(Form1->Edit2->Text);
              for (int y=0; y<=(int(r.Bottom)); y++)
                  {
                    byte* LinePtr=(byte*)Form1->ImageEnView1->Bitma
                    for (int x=0; x<=(int(r.Right)*3); x++)
                       {
                     if (LinePtr[x] > w)
                       LinePtr[x] = 255;
                     else
                       LinePtr[x] = 0 ;

                       byte* Pixel=LinePtr+x*3;
                     if(Pixel[0]<100 && Pixel[1]>250 && Pixel[2]<100
                       {Pixel[0]=0;Pixel[1]=255;Pixel[2]=0;}
//                                          White
                     if(Pixel[0]>250 && Pixel[1]>250 && Pixel[2]>250
                       || Pixel[0]<100 && Pixel[1]<100 && Pixel[2]>
                       {Pixel[0]=0;Pixel[1]=0;Pixel[2]=0;}

                       }
                  }
             /*---Find an Object Boundary---*/
             //X Bound of Destination
             for (int y=0; y<=(int(r.Bottom)-2); y++)
                 {byte* LinePtr=(byte*)Form1->ImageEnView1->Bitmap
                  for (int x=0; x<=(int(r.Right)); x++)
                      {byte* PrePi=LinePtr+((x-1)*3);
                       byte* Pixel=LinePtr+x*3;
                       PixelBuf2=Pixel[1];
                       PixelBuf1=PrePi[1];
                       if(abs(PixelBuf2-PixelBuf1)!=0)
                         {
                          if((-1*(PixelBuf2-PixelBuf1))!= abs(Pixe
                           {StrX=x;}
```

```
                    else if(((-1*(PixelBuf2-PixelBuf1))== ab
                     {StpX=x; //keep position end of edge
                       if(lengthX<=(StpX-StrX)&& lengthX+3>=
                         {lengthX=(StpX-StrX);StartX=StrX;St
                       }
                     }
                  }
            }
          //Y Bound of Destination
          int y=1;
          for(int x=0; x<int(r.Right)  ; x++)
               {
               do{   byte* LinePtr=(byte*)Form1->ImageEnView1->B
                     byte* LinePtr2=(byte*)Form1->ImageEnView1->
                     byte* Pixel=LinePtr+x*3;
                     PixelBuf2=Pixel[1];
                     Pixel=LinePtr2+x*3;
                     PixelBuf1=Pixel[1];
                     if(abs(PixelBuf2-PixelBuf1)!=0)
                       {
                       if((-1*(PixelBuf2-PixelBuf1))!= abs(Pixe
                        {StrY=y;}//keep start edge
                       else if(((-1*(PixelBuf2-PixelBuf1))== ab
                        {StpY=y; //keep position end of edge
                         if(lengthY<=(StpY-StrY)&&lengthY+3>=(
                           {lengthY=(StpY-StrY);StartY=StrY;St
                         }
                       }
                  y++;
               }while(y<=int(r.Bottom)); y=1;
            }

          //X Bound of Robot
          for (int y=(BStartY); y<=(BStopY); y++)
           {byte* LinePtr=(byte*)Form1->ImageEnView1->Bitmap
            for (int x=BStartX; x<=BStopX; x++)
              {byte* PrePi=LinePtr+((x-1)*3);
               byte* Pixel=LinePtr+x*3;
               PixelBuf2=Pixel[1];
               PixelBuf1=PrePi[1];
               if(abs(PixelBuf2-PixelBuf1)!=0)
                 {
                  if((-1*(PixelBuf2-PixelBuf1))!= abs(Pixe
                   {StrX=x;}
                  else if(((-1*(PixelBuf2-PixelBuf1))== ab
                   {StpX=x; //keep position end of edge
                     if(SmallX<=(StpX-StrX)&&SmallX+2>=(St
                       {SmallX=(StpX-StrX);SStartX=StrX;SS
                     }
                  }
               }
           }

          //Y Bound of Robot
          y=BStartY;
```



รูป A.10 แสดงบล๊อกไดอะแกรมกาเคลื่อนที่เข้าหาจุด

```
                for(int x=BStartX; x<=BStopX  ; x++)
                    {
                    do{   byte* LinePtr=(byte*)Form1->ImageEnView1->B
                          byte* LinePtr2=(byte*)Form1->ImageEnView1->
                          byte* Pixel=LinePtr+x*3;
                          PixelBuf2=Pixel[1];
                          Pixel=LinePtr2+x*3;
                          PixelBuf1=Pixel[1];
                          if(abs(PixelBuf2-PixelBuf1)!=0)
                             {
                             if((-1*(PixelBuf2-PixelBuf1))!= abs(Pixe
                               {StrY=y;}//keep start edge
                             else if(((-1*(PixelBuf2-PixelBuf1))== ab
                               {StpY=y; //keep position end of edge
                                if(SmallY<=(StpY-StrY)&&SmallY+2>=(St
                                  {SmallY=(StpY-StrY);SStartY=StrY;SS
                                }
                             }
                          y++;
                        }while(y<=BStopY); y=BStartY;
                    }
                    //GreenPlot();
}
//---------------------------------------------------------------------
//                          Button handle routine
//---------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner): TForm(Owner)
{ /*Call Parallel port driver from TvicHW32*/
  HW32 = 0;
      HW32 = OpenTVicHW32(HW32,"TVICHW32","TVicDevice0");
  ActiveHW = GetActiveHW(HW32);
  if (ActiveHW) nPort = (Word) GetLPTBasePort(HW32);
  else
  {
    nPort = 0x378;
    ShowMessage(" The driver TVicHW32 can not be found!");
  }
}
//---------------------------------------------------------------------
void __fastcall TForm1::FormActivate(TObject *Sender)
{    /*define parallel bits*/
     CPinWrite[ 2] = WPin2;
     CPinWrite[ 3] = WPin3;
     CPinWrite[ 4] = WPin4;
     CPinWrite[ 5] = WPin5;

}
//---------------------------------------------------------------------
void __fastcall TForm1::WPin1Click(TObject *Sender)
{
  for (Byte nPin = 2; nPin<=5; nPin++)
     SetPin(HW32,nPin,(BOOL)CPinWrite[nPin]->Checked);
}
//---------------------------------------------------------------------
```

```
void __fastcall TForm1::Button1MouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{             /*Trun Left*/
        turnL();
}
//---------------------------------------------------------------------

void __fastcall TForm1::Button1MouseUp(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
        stop();
}
//---------------------------------------------------------------------

void __fastcall TForm1::Button4MouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{             /*Step Forward*/
        go();
}
//---------------------------------------------------------------------

void __fastcall TForm1::Button2MouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{             /*Step Backward*/
        back();
}
//---------------------------------------------------------------------

void __fastcall TForm1::Button3MouseDown(TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{             /*Trun Right*/
        turnR();
}
//---------------------------------------------------------------------

void __fastcall TForm1::VideoPreview1Click(TObject *Sender)
{
/* Enable Video Preview window */
            null=false;
            ImageEnVideoView1->ShowVideo =VideoPreview1->Enabled;
            Image1->Picture->Bitmap = new Graphics::TBitmap;
            Image1->Picture->Bitmap->PixelFormat=pf24bit;
            ImageEnView1->Bitmap->PixelFormat=pf24bit;
            ImageEnVideoView1->OnVideoFrame=0;
}
//---------------------------------------------------------------------

void __fastcall TForm1::SourceandSetting1Click(TObject *Sender)
{
        ImageEnVideoView1->DoConfigureSource();
}
//---------------------------------------------------------------------

void __fastcall TForm1::Sizeandformat1Click(TObject *Sender)
{
```

```
        ImageEnVideoView1->DoConfigureFormat();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::About1Click(TObject *Sender)
{
        AboutBox->ShowModal();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::EnablePreview1Click(TObject *Sender)
{
  /* Show image processed window */
        if (EnablePreview1->Enabled)
        {
            ImageEnVideoView1->OnVideoFrame=ImageEnVideoView1VideoFr
        }
        else
        ImageEnVideoView1->OnVideoFrame=0;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::DiablePreview1Click(TObject *Sender)
{
        ImageEnVideoView1->OnVideoFrame=0;
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button5Click(TObject *Sender)
{       // must only keep non object background
        null=true;
        Capture();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Act
{
        if (MessageBox(NULL,"Program will be terminating ! Do you re
            {stop();Application->Terminate();}
        else Abort();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Quit1Click(TObject *Sender)
{
/*When User Click on Quit menu ,Ask again to confirm one */
        if (MessageBox(NULL,"Program will be terminating ! Do you re
            {stop();Application->Terminate();}
        else Abort();
}
//---------------------------------------------------------------------------

void __fastcall TForm1::ImageEnVideoView1Job(TObject *Sender, TIEJob
      int per)
{    /* For show label when check devices*/
```

```
    switch( job ) {
        case iejNOTHING:
        Label1->Caption="";
         break;
            case iejVIDEOCAP_CONNECTING:
        Label1->Caption="Connecting devices...";
         break;
    }
   Application->ProcessMessages();


}
//---------------------------------------------------------------------
//                              Type of Processing
//---------------------------------------------------------------------
void __fastcall TForm1::ImageEnVideoView1VideoFrame(TObject *Sender,
      TIEDibbitmap *Bitmap)
{  /*Selecting Image Processing methods*/

        TColor PixelBuf2,PixelBuf1,Blue,Red,Green;
     Bitmap->CopyToTBitmap(ImageEnView1->Bitmap);
        ImageEnView1->Update();
        TRect r=ImageEnVideoView1->GetVideoSize();
                                  //do not thing
      if(RadioButton3->Checked)
        {
        StrTimer();
        Bitmap->CopyToTBitmap(ImageEnView1->Bitmap);
        ImageEnView1->Update();
       // Sleep(1000);  /*Test Timer*/
        StpTimer();
        }
                                  //Blue Pass filtration && Tracking
      else if(RadioButton6->Checked)
        {
          StrTimer();
          BlueFunct();
          StpTimer();
        }
                                  //Red Pass filtration
      else if(RadioButton7->Checked)
        {
          StrTimer();
          RedFunct();
          StpTimer();
        }
                                 // Green Colour Filtration && Trackin
      else if(RadioButton8->Checked)
        {
          StrTimer();
          GreenFunct();
          StpTimer();
        }
                                  //Main Programe
      else if(RadioButton9->Checked)
```

```
{
     StrTimer();
     BlueFunct();

     Bitmap->CopyToTBitmap(ImageEnView1->Bitmap);  // refresh
     ImageEnView1->Update();

     GreenFunct();
    // PixelVec();    //find all pixel path for object avoidi

     Bitmap->CopyToTBitmap(ImageEnView1->Bitmap);
     ImageEnView1->Update();

    // RedFunct();

     Bitmap->CopyToTBitmap(ImageEnView1->Bitmap);
     ImageEnView1->Update();
     /*controlling robot*/
     Zeta2=ZetaValue(GctX,GctY,BctX,BctY);
     Zeta1=ZetaValue(SGctX,SGctY,BctX,BctY);


     Label9->Caption="Destination Angle  ="+IntToStr(Zeta2);
     Label10->Caption="Robot Angle  ="+IntToStr(Zeta1);
     Label11->Caption="Differential Angle  ="+IntToStr(Zeta2-

     line=sqrt((GctX-BctX)*(GctX-BctX))+((GctY-BctY)*(GctY-B
    if(15<=abs(Zeta2-Zeta1))
      {
       if((Zeta2-Zeta1)>0)
          {
           turnR();
           Label13->Caption="Turn Right";
          }
        else if((Zeta2-Zeta1)<0)
          {
           turnL();
           Label13->Caption="Turn Left";
          }
      }
    else if((14>=(Zeta2-Zeta1)) || ((-14)<=(Zeta2-Zeta1)))
          {
            if((11<(Zeta2-Zeta1)))
            {for(int delay=0;delay<10;delay++)
              {
               turnR();
              }
            }
            if((-11)>(Zeta2-Zeta1))
            {for(int delay=0;delay<10;delay++)
              {
               turnL();
              }
            }
           if((-10)<=(Zeta2-Zeta1)&&(10>=(Zeta2-Zeta1)))
```

```
                    {for(int delay=0;delay<10;delay++)
                      {
                       go();
                       Label13->Caption="Go..Go";
                      }
                    }
                  }
          if(line<=10)
               {stop();Label13->Caption="Stop";
                MessageDlg("The end of path",mtInformation, TMsgDlgB
                RadioButton9->Checked=false;
               }

               BluePlot();
               GreenPlot();
               RedPlot();
               StpTimer();
     }
                              // color segmentation
     else if(RadioButton5->Checked)
     {
          StrTimer();
          int w = StrToInt(Edit1->Text);
          byte* LinePtr;

              for (int y=0; y<=(int(r.Bottom)); y++)
                  {
                  LinePtr=(byte*)ImageEnView1->Bitmap->ScanLine[y];
                  for (int x=0; x<=(int(r.Right)*5); x++)
                      {
                   if (LinePtr[x] >w)
                       LinePtr[x] = 255;
                   else
                       LinePtr[x] = 0 ;
                      }
                  }
          ImageEnView1->Refresh();
            StpTimer();
     }
        ImageEnView1->Fit();
                             //background sub.
     if(CheckBox1->Checked)
          {
             if(null==true)
             {
                 Sub();
             }
             else
             {
                MessageDlg("You Must Background Capture first",mtErr
                CheckBox1->State=cbUnchecked;
             }
          }

}
```

```
//---------------------------------------------------------------------------
```

A. อัลกอริทึมการเคลื่อนที่ของหุ่นยนต์ภายใต้การประมวลผลภาพ

การที่ตัวหุ่นจะเคลื่อนที่จากจุดเริ่มต้น (Start) ไปจนถึงจุดสิ้นสุด (Destination) โดยผ่านสิ่งกีดขวางต่างๆที่วางตัวแบบสุ่ม(Random) จะต้องมีอัลกอริทึมการเคลื่อนที่ของตัวหุ่นที่สัมพันธ์กับกล้องที่จับภาพที่ตำแหน่งต่างๆในเวลาใดๆซึ่งแบ่งเป็น 2 ขั้นตอนดังนี้

A.1 การหาเส้นทางที่เป็นไปได้จากจุดเริ่มจนถึงจุดสิ้นสุด

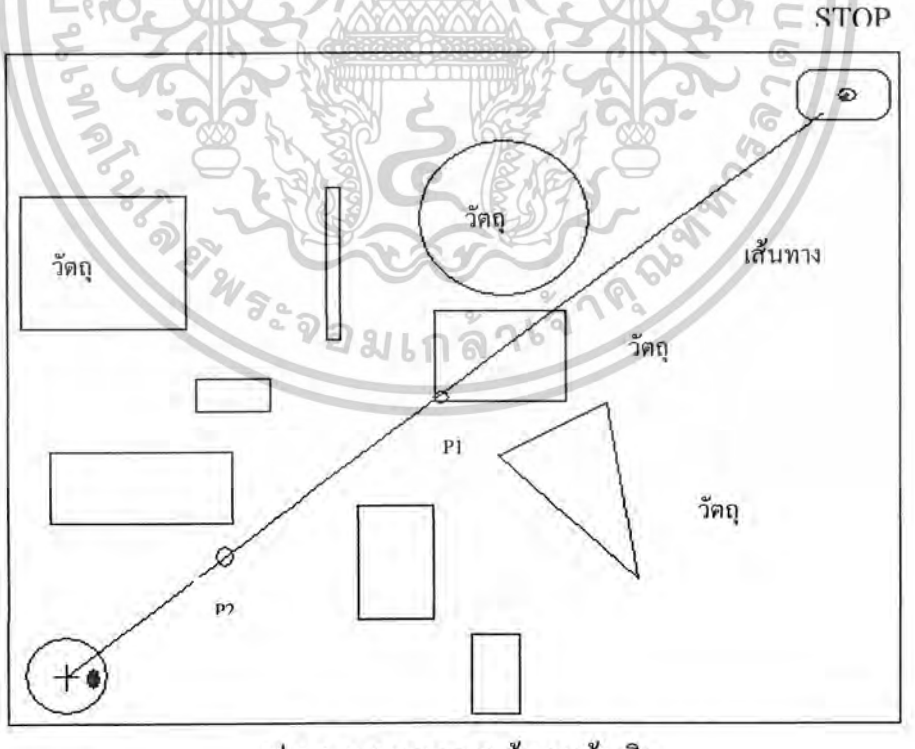
A.2 การเดินทางไปตามเส้นที่เลือกไว้ซึ่งแบ่งเป็น

A.2.1 การหมุนตัวของหุ่นยนต์ (เพื่อให้มีทิศทางตรงกับจุดที่จะเดินไป)

A.2.2 การเคลื่อนที่เข้าหาจุด

A.1 การหาเส้นทางที่เป็นไปได้จากจุดเริ่มต้นจนถึงจุดสิ้นสุด

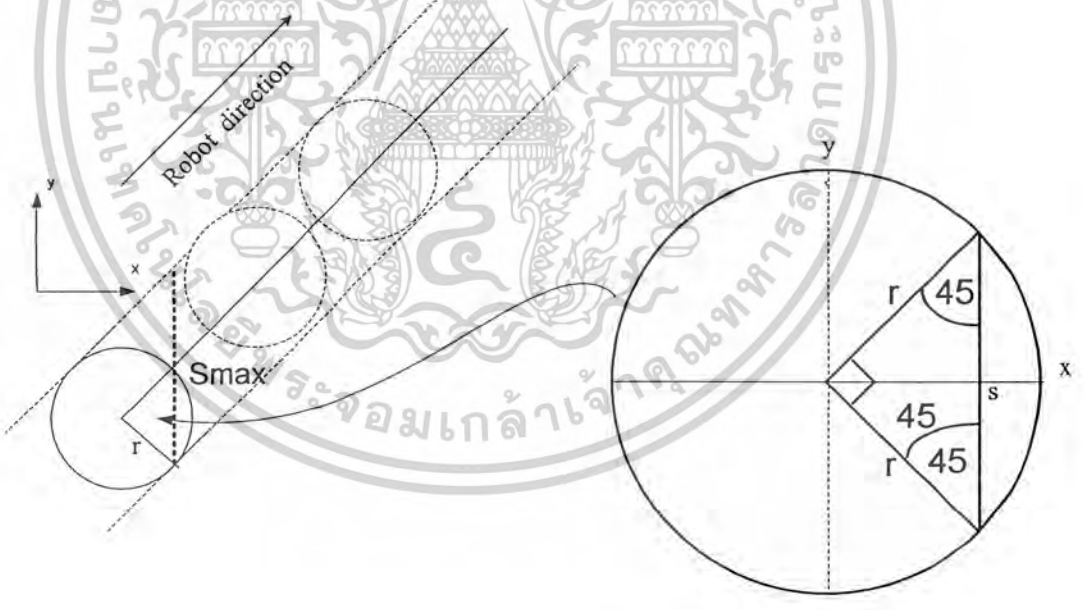
โดยมีข้อกำหนดว่าเส้นทางที่หุ่นยนต์จะเคลื่อนที่ผ่านไปได้จะต้องมีความกว้างของเส้นทางมากกว่าหรือเท่ากับความกว้างของตัวหุ่นยนต์ จากข้อกำหนดข้างต้นจะเป็นไปได้ก็ต่อเมื่อใช้การแสกนเส้นในแกน x และในแกน y โดยการใช้ค่าความกว้างของหุ่นเป็นค่าอ้างอิงและเลื่อนจุดคำนวณไปทีละ Pixel และทำเครื่องหมาย (mark) จุดที่เป็นไปได้ ซึ่งจุดนี้คือจุดบนเส้นทางนั่นเอง พิจารณาจากรูปที่ 3.18 เริ่มจากการขีดเส้นตรงจากจุดเริ่มต้นจนถึงจุดสิ้นสุด เส้นทางนี้จะเป็น



รูป A.1 แสดงการลากเส้นทางอ้างอิง

เส้นทางแรกที่ใช้ในการพิจารณาหาเส้นทางที่เป็นไปได้แต่เส้นนี้ไม่จำเป็นต้องขนานหรือตั้งกับแกน x หรือ y และในการแสกนจะแสกนค่าตำแหน่งภาพที่เป็นลำดับซึ่งขนานกับแกน x หรือ y เท่านั้น ดังนั้นเมื่อทิศทางการเคลื่อนที่ของหุ่นที่ไม่ขนานหรือตั้งฉากกับแนว x , y จึงจะต้องหาระยะ (s) ซึ่งอยู่ในแนวแกน x , y ที่ทำให้ตัวหุ่นเคลื่อนที่ผ่านไปได้

จากนั้นจะทำการคำนวณค่าระยะทางที่เป็นไปได้ครั้งแรกซึ่งอยู่บนเส้นทางอ้างอิง สามารถทำได้โดยการหาค่าบนเส้นทางอ้างอิงว่ามีค่าใดบ้างที่ซ้อนทับกับค่าของวัตถุแล้วก็เก็บค่าตำแหน่งไว้จากนั้นจึงคำนวณระยะทางจากตัวหุ่นไปหาค่าตำแหน่งที่เก็บและเปรียบเทียบหาค่าที่น้อยที่สุดก็จะได้จุดแรกบนเส้นทางอ้างอิงที่พบวัตถุนับจากตัวหุ่น ดูรูปที่ 3.18 ประกอบจุดที่คำนวนได้ในที่นี้คือจุด P1 นั่นเอง อย่างไรก็ตามเส้นทางที่หามาได้ไม่จำเป็นต้องเป็นเส้นทางที่เป็นไปได้เสมอไปเพราะอาจจะมีวัตถุวางอยู่ใกล้กับเส้นทางอ้างอิงจึงต้องมีการคำนวณระยะระยะเผื่อที่หุ่นจะสามารถข้ามไปได้ อีกครั้งพิจารณาจากรูปที่ 3.18 เส้นระหว่างตัวหุ่นกับจุด P1 ที่หามาได้จะเห็นได้ว่ามีจุดที่หุ่นไม่สามารถเคลื่อนที่ผ่านไปได้นั่นก็คือจุด P2 เมื่อมาถึงขั้นตอนนี้เราจะใช้การคำนวณระยะเผื่อที่หุ่นสามารถผ่านไปได้โดยการคำนวณจะบอกได้ว่าจุดใดบนเส้นทางอ้างอิงที่หุ่นไม่สามารถผ่านไปได้ดังนี้



รูป A.2 ค่าสูงสุดในการแสกนแนว x และแนวเส้นทาง

จากรูปที่ 3.19 ค่า s สูงสุดในแนว x หรือ y ที่ทำให้เกิดเส้นทางที่ทำให้หุ่นเคลื่อนที่ผ่านไปได้จะเกิดขึ้นเมื่อทิศทางของหุ่นทำมุมกับแนว x , y เท่ากับ 45° โดยระยะทางสูงสุดหาได้จาก

$$S^2 = r^2 + r^2$$

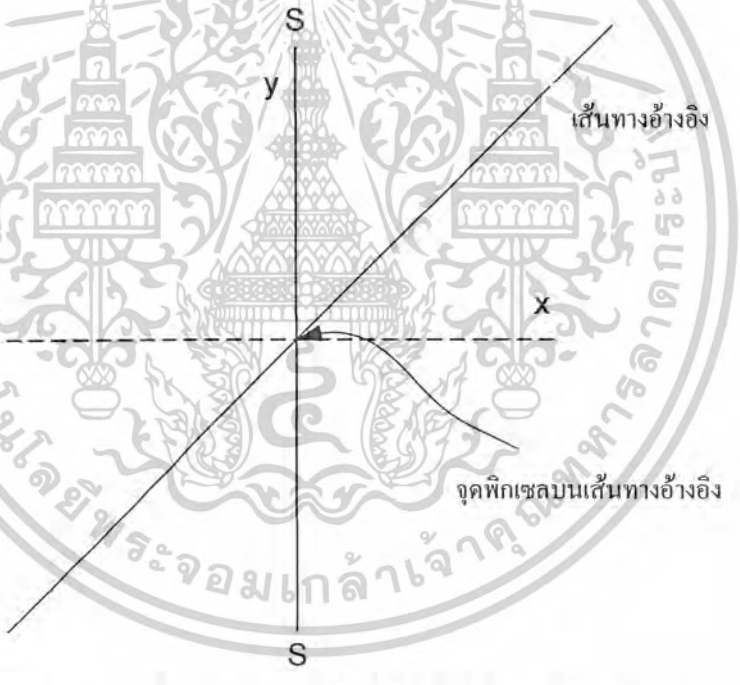
$$S = r\sqrt{2}$$

โดยที่ r คือ รัศมีของหุ่น

ค่า S คือค่าระยะทางที่จะใช้ในการคำนวณระยะความกว้างของเส้นทางที่น้อยที่สุดที่หุ่นยังสามารถผ่านไปได้ดูรูปที่ 3.20 ประกอบจุดP(x , y)คือจุดใดๆที่อยู่บนเส้นทางอ้างอิงโดยให้จุด P(x , y)เป็นจุดกึ่งกลางของระยะเผื่อทั้งสองด้านของเส้นทางแล้วแสกนตามแนว x หรือ y

ดังนั้นระยะบนแกนสูงสุด $S_{max} = 2S$

$= r2\sqrt{2}$



รูป A.3 แสดงระยะเผื่อ S ในการคำนวณเส้นทาง

เมื่อคำนวนเสร็จดังข้างต้นจะทราบจุดและเส้นทางที่เป็นไปได้เส้นแรกจุดดังกล่าวจะถูกกำหนดให้เป็น P1 จากนั้นการหาจุด P2 , P2 , P3 , P4 ,... Pn จะใช้วิธีการแสกนแนวเส้นตามแกนx และ y สลับกันไปโดยการแสกนจะใช้ค่าความกว้างของหุ่นเป็นค่าอ้างอิง เช่น ความกว้างของหุ่นเท่ากับ 3 พิกเซลจะมีการแสกนดังรูปที่ 3.21

จากรูปที่ 3.21 เริ่มแสกนในแกน x ก่อน แสกนเส้นที่ 1 , 2 , 3 มีความยาวสูงสุดเท่ากับความยาวของเส้นที่ 2 (ความยาวน้อยที่สุด) เท่ากับ 12 พิกเซล เก็บค่า x=1 ซึ่งเป็นพิกเซลตรงกลางไว้

จากนั้นจึงแสกนในแนวแกน y โดยการแสกนเส้นที่ 0 , 1 , 2 ก่อนจะได้ความยาวสูงสุดเท่ากับ 9 เก็บค่าไว้เปรียบเทียบ

ตามด้วยการแสกนเส้นที่ 1,2,3 จะได้ความยาวสูงสุดเท่ากับ 9

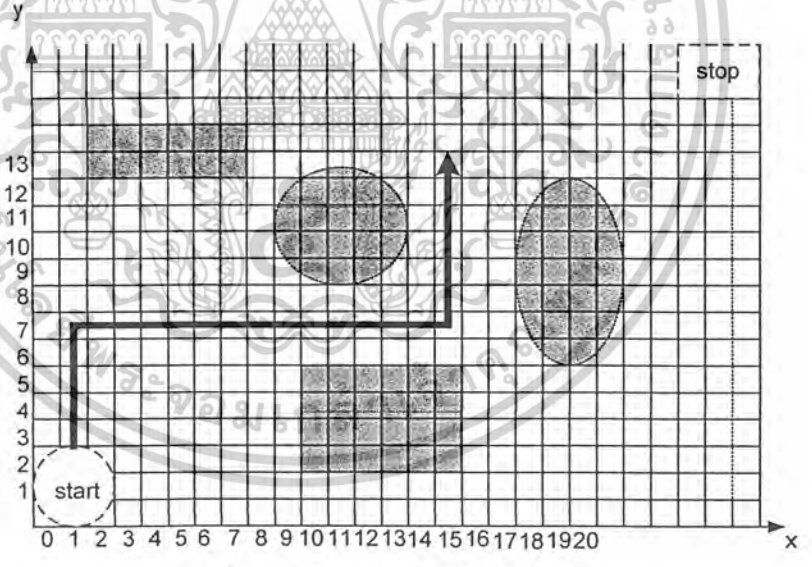
จากนั้นจะเป็นเส้นที่ 2,3,4 และ 3,4,5 ทำเช่นนี้เรื่อยไปจนถึง 10,11,12

แล้วจึงนำค่าสูงสุดของกลุ่มทั้งหมดซึ่งจะได้ความยาวสูงสุดที่กลุ่มพิกเซล 6,7,8

เท่ากับ 17 พิกเซล จึงเก็บค่ากลางของกลุ่มไว้ นั่นคือ y = 7

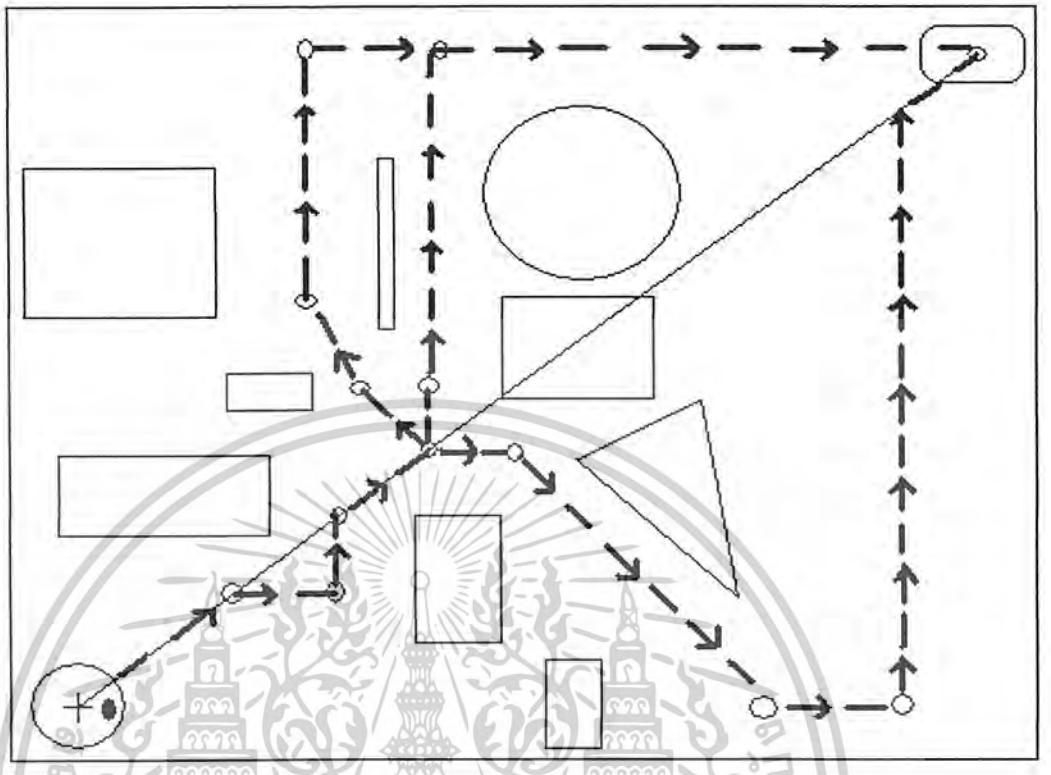
นำค่า x และ y ที่ได้มาเข้าคู่อันดับ จะได้ ตำแหน่ง แรกบนเส้นทางคือ P1(x,y)=(1,7)

จากนั้นสลับกลับมาแสกนที่แกนx จากค่า 1ถึง 17 ซึ่งจะได้ กลุ่มสูงสุดคือ 14,15,16 ซึ่งค่ากลางคือ 15 นำมาสร้างคู่อันดับจะได้ P2(x,y)=(15,7) ซึ่งเป็นจุดที่สองที่ต้องอยู่บนเส้นทาง



รูป A.4 การหาเส้นทางโดยการแสกน x ,y สลับกัน

โดยวิธีข้างต้นทั้งหมดที่กล่าวมาเมื่อพิจารณาดังรูปที่ 3.18 จะทำให้ได้เส้นทางที่เป็นไปได้ดังรูปที่ 3.22 จากรูปจะเห็นได้ว่าทั้งสามเส้นทางจะมีจุดอยู่บนเส้นทางอย่างแน่นอนเราสามารถหาค่าความยาวของเส้นทางทั้งสามได้โดยการหาเส้นทางย่อยจุดต่อจุดแล้วนำมาบวกกันจากนั้นจึงนำเส้นทางทั้งสามมาเปรียบเทียบหาเส้นทางที่สั้นที่สุดเส้นทางเส้นนั้นคือเส้นทางที่หุ่นยนต์จะต้องเคลื่อนที่ไปแบบจุดต่อจุด



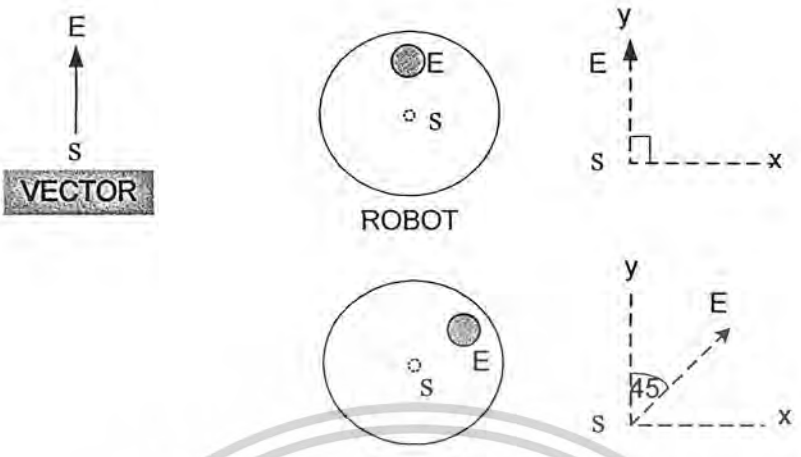
รูป A.5 แสดงเส้นทางที่เป็นไปได้ทั้งหมดโดยอ้างอิงจากรูปที่ 3.18

A.2 การเดินทางไปตามเส้นที่เลือกไว้ซึ่งแบ่งเป็น

เมื่อหาเส้นทางได้แล้วดังที่ได้กล่าวมาข้างต้นในเส้นทางจะมีจุดต่างกำกับไว้เป็นลำดับเพื่อให้หุ่นเคลื่อนที่เข้าหาเป้าหมายโดยผ่านจุดต่อจุดซึ่งจุดที่เชื่อมต่อกันทุกจุดจะมีระยะทางและมุมต่างกัน เช่น จากจุด P0 ไป P1(P0 --> P1) มีมุมที่หุ่นจะต้องหมุนไปเพื่อด้านหน้าของหุ่นตรงกับทิศทางที่หุ่นจะต้องเคลื่อนที่ไปและจุดที่หุ่นจะต้องเคลื่อนที่เข้าหา

A.2.1 การหมุนตัวของหุ่นยนต์ (เพื่อให้มีทิศทางตรงกับจุดที่จะเดินไป)

เนื่องจากหุ่นมีทิศทางการเคลื่อนที่ทิศทางเดียวเท่านั้นและใช้การหมุนรอบตัวเองเพื่อเบี่ยงเบนทิศทางไปในทิศทางที่กำหนด เช่น การการหมุนรอบตัวของหุ่นไปทางขวา 45° โดยสมมุติเดิมทีหุ่นมีมุมอ้างอิงอยู่ที่ 90° ดังรูปที่ 3.23 ซึ่งมีบล๊อกไดอะแกรมการทำงานดังรูปที่ 3.24 โดยอัลกอรึทึมการหมุนจะต้องสอดคล้องกับภาพที่อ่านค่ามาได้ในเวลานั้นๆ



รูป A.6 แสดงเวกเตอร์และรูปด้านบนของหุ่นเมื่อหมุนไปทางขวา 45°

การที่หุ่นจะหมุนไปที่มุมใดๆได้นั้นจะต้องสร้าง**มุมอ้างอิง**(ϕ_{ref})ขึ้นมาหนึ่งมุมและเมื่อหุ่นเริ่มหมุนตัวไปทางขวาก็จะสุ่มอ่าน**มุมใหม่**(ϕ_{new}) มาเพื่อเปรียบเทียบกับมุมอ้างอิง โดยนำค่ามุมอ้างอิงมาลบมุมใหม่จะได้**มุมผลลัพธ์**(ϕ_{result})ดังนี้

$$\phi_{result} = \phi_{ref} - \phi_{new}$$

ซึ่งหากค่า ϕ_{result} < 45° ตัวหุ่นก็จะหมุนต่อไปแต่ถ้าหาก $\phi_{result} \geq 45°$ จะสั่งให้หุ่นหยุดเพื่อรอคำสั่งต่อไป ดังบล็อกไดอะแกรมในรูปที่ 3.23 อัลกอริทึมการหมุนซ้ายก็ทำนองเดียวกันเพียงแต่ส่งคำสั่งหมุนซ้ายออกไปที่พอร์ตคอมพิวเตอร์เท่านั้น

การคำนวนมุมอ้างอิงหาค่าได้จากจุด s และ E ซึ่งจะเป็นมุมเริ่มต้นที่ตัวหุ่นหยุดรออยู่ครั้งแรกหาได้จาก

$$\phi_{ref} = \tan^{-1}\left[\frac{y_E - y_S}{x_E - x_S}\right]$$

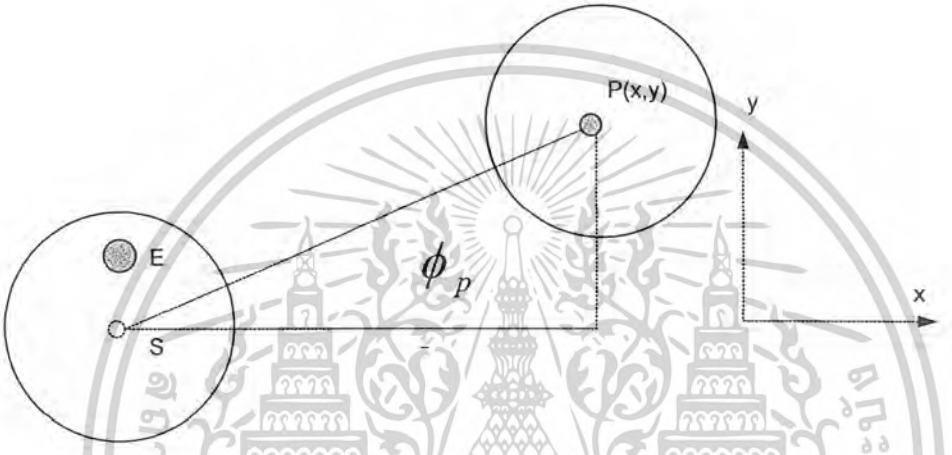
การคำนวณหาค่ามุม ϕ_{new} จะเป็นค่ามุมเมื่อจุด E เลื่อนไปที่ตำแหน่งต่างๆ จึงคำนวณหาได้เหมือนกับค่า ϕ_{ref} แต่จะต่างกันที่เมื่อเวลาผ่านไปจุดE จะเปลี่ยนไป

$$\phi_{new} = \tan^{-1}\left[\frac{y_E - y_S}{x_E - x_S}\right]$$

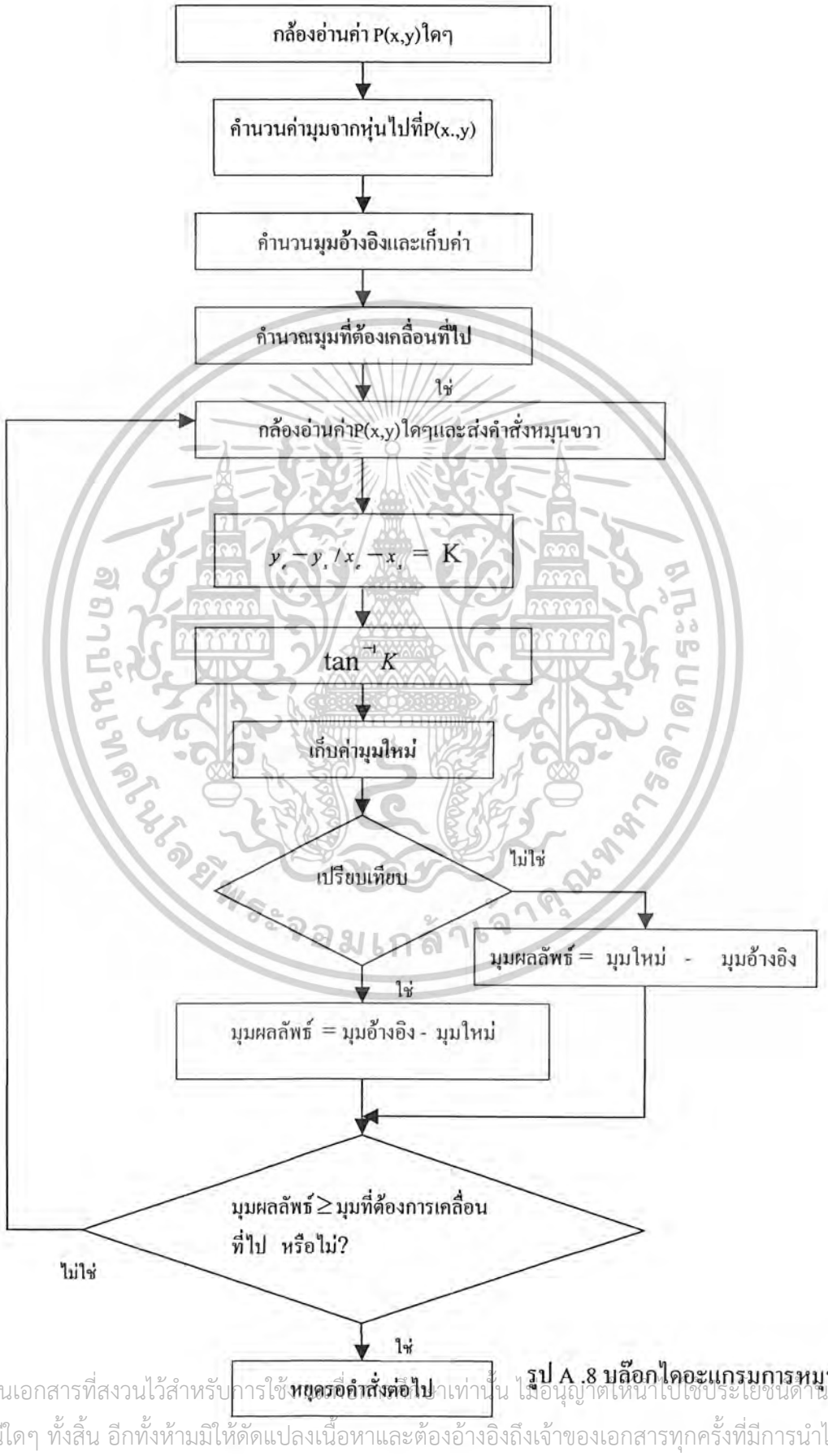
มุมที่ต้องการเคลื่อนที่ไป(ϕ_{track})**คือมุมต่างระหว่างมุมอ้างอิงกับมุมของจุด P ใดๆ**(ϕ_p)หาได้จากการใช้จุดศูนย์กลาง (S) และจุดP(x , y) ใดๆที่ตัวหุ่นต้องเคลื่อนที่เข้าไปหา(เป็นจุดที่อยู่บนเส้นทางซึ่งหาไว้แล้วตั้งแต่แรกนั่นเอง)

$$\phi_{track} = \phi_{ref} - \phi_p$$

โดย ϕ_p หาได้จาก $$\phi_p = \tan^{-1}\left[\frac{y_P - y_S}{x_P - x_S}\right]$$



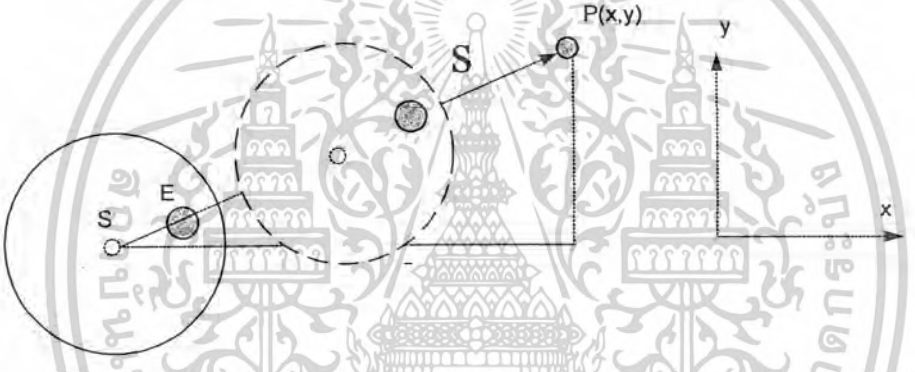
รูป A.7 แสดงมุม ϕ_p



รูป A .8 บล๊อกไดอะแกรมการหมุนขวา

A.2.2 การเคลื่อนที่เข้าหาจุดP(x,y)ใดๆ

เมื่อตัวหุ่นหันไปตรงกับจุดที่ต้องการเดินเข้าหาแล้วต่อไปก็คือการเดินเข้าหาจุด P(x,y)สามารถทำได้โดยการสั่งให้เคลื่อนที่ไปข้างหน้าเข้าหาจุด P แล้วก็สุ่มวัดระยะห่างจากจุดPและจุด S ที่อยู่บนตัวหุ่นยนต์ซึ่งเมื่อเวลาผ่านไประยะทางจะสั้นลง โดยทุกครั้งที่มีการคำนวณหาระยะทางจะเปรียบเทียบทุกครั้งว่าระยะห่างน้อยกว่าค่าที่ตั้งไว้หรือยังถ้ายังก็สั่งให้เดินหน้าต่อแต่หากน้อยกว่าค่าที่ตั้งแล้วให้หยุดเพื่อรอคำสั่งต่อไปรูปการเคลื่อนที่เข้าหาจุดแสดงดังรูปที่ 3.26



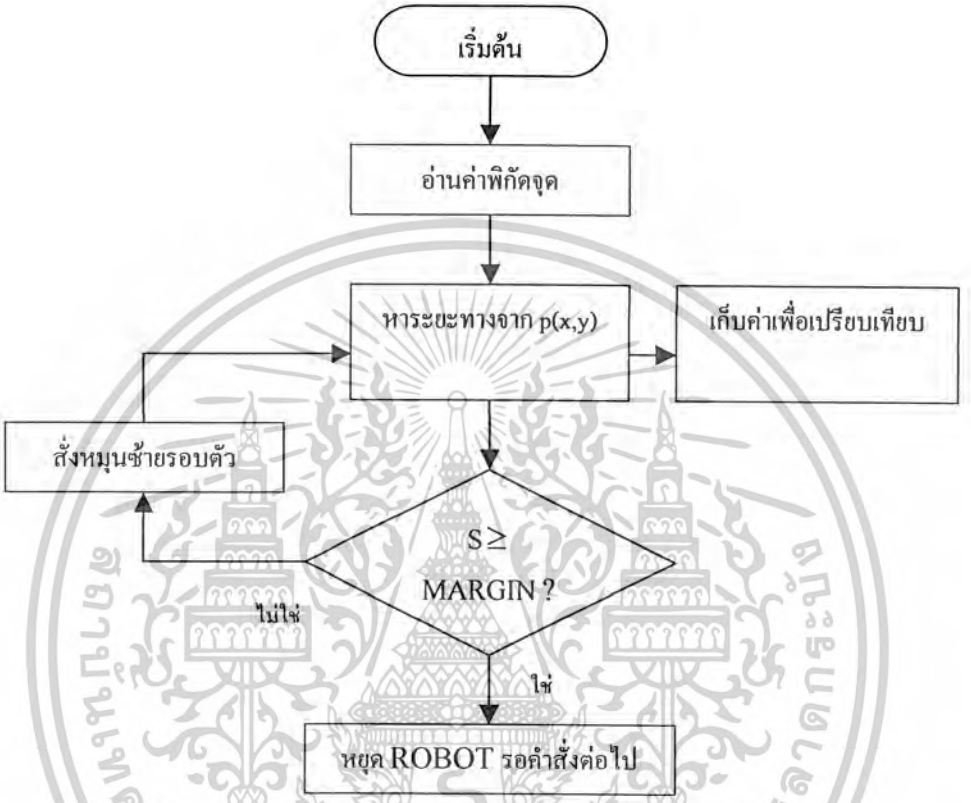
รูป A.9 แสดงการเคลื่อนเข้าหาจุด P(x , y)

หุ่นจะหยุดรอคำสั่งก็ต่อเมื่อจุด S ทับซ้อนอยู่ที่จุด P(x,y)พอดี

การคำนวณระยะทาง S นั้นสามารถทำได้โดย

$$S = \sqrt{(x_P - x_S)^2 + (y_P - y_S)^2}$$

ซึ่งมีบล๊อกไดอะแกรมการทำงานดังรูปที่ 3.27



รูป A.10 แสดงบล๊อกไดอะแกรมกาเคลื่อนที่เข้าหาจุด