ปัญหาการติดตามเป้าหมายและการหลบหลีกสิ่งกีดขวางของหุ่นยนต์ด้วยวิธีควบคุมแบบฟัซซีอย่างง่าย

TRACKING PROBLEM AND OBSTACLE AVOIDANCE OF MOBILE ROBOT USING SIMPLE FUZZY CONTROL

อภิวิชญ์ แก้วนพรัตน์

APHIWIT KAEWNOPPARAT

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ. 2547

ISBN 974-15-1079-9

ปัญหาการติดตามเป้าหมายและการหลบหลีกสิ่งกีดขวางของหุ่นยนต์ด้วยวิธีควบคุมแบบฟัซซีอย่างง่าย

TRACKING PROBLEM AND OBSTACLE AVOIDANCE OF MOBILE ROBOT USING SIMPLE FUZZY CONTROL

อภิวิชญ์ แก้วนพรัตน์
APHIWIT KAEWNOPPARAT

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2547

ISBN 974-15-1079-9

TRACKING PROBLEM AND OBSTACLE AVOIDANCE OF MOBILE ROBOT USING SIMPLE FUZZY CONTROL

APHIWIT KAEWNOPPARAT

A THESIS SUMMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF
MASTER OF ENGINEERING IN ELECTRICAL ENGINEERING
SCHOOL OF GRADUATE STUDIES
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
2004

ISBN 974-15-1079-9

หัวข้อวิทยานิพนธ์	ปัญหาการติดตามเป้าหมายและการหลบหลีกสิ่งกีดขวางของหุ่นยนต์ด้วยวิธีควบคุมแบบฟัซซีอย่างง่าย
นักศึกษา	นาย อภิวิชญ์ แก้วนพรัตน์
รหัสนักศึกษา	42061006
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมไฟฟ้า
พ.ศ.	2547
อาจารย์ผู้ควบคุมวิทยานิพนธ์	รศ.ดร. ปิติเขต สู้รักษา

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้กล่าวถึงวิธีการแก้ปัญหาการติดตามเป้าหมายของหุ่นยนต์ในขณะที่มีสิ่งกีดขวางด้วยกระบวนการควบคุมแบบฟัซซีอย่างง่ายซึ่งเป็นปัญหาวิจัยหนึ่งที่น่าสนใจ โดยรูปแบบของฟังก์ชันความเป็นสมาชิกออกแบบให้อยู่ในรูปแบบที่ง่ายต่อการนำไปใช้จริง พารามิเตอร์ที่ใช้ในการควบคุมออกแบบโดยอาศัยขนาดของหุ่นยนต์และระยะอันตรกิริยาระหว่างตัวตรวจรับกับสิ่งแวดล้อม เพื่อตรวจสอบการออกแบบจึงจำลองการเคลื่อนที่ของหุ่นยนต์ในสถานการณ์ต่าง ๆ ด้วยคอมพิวเตอร์โดยใช้โปรแกรมวิชวลเบสิก ผลการจำลองที่ได้แสดงถึงแนวทางสู่การสร้างจริงของระบบควบคุมที่นำเสนอ

Thesis Title	Tracking Problem and Obstacle Avoidance of Mobile Robot using Simple Fuzzy Control
Student	Mr. Aphiwit Kaewnopparat
Student ID.	42061006
Degree	Master of Engineering
Programme	Electrical Engineering
Year	2004
Thesis Advisor	Assoc.Prof.Dr.Pitikhate Sooraksa

ABSTRACT

This thesis presents a solution to tracking problem, which is one of interesting research topics, for robot with obstacle avoidance using simple fuzzy control. The proposed forms of membership functions are designed in light of easy-to-implement for the real-world application. Control parameters are considered and assigned based upon size of the robot and interaction distance between sensors and environment. To evaluate the effectiveness of the design, computer simulation using Visual Basic is carried out to validate the robot movement. The results show a guideline possibility for implementation of the system using the proposed design.

กิตติกรรมประกาศ

วิทยานิพนธ์เล่มนี้สำเร็จได้ด้วยด้วยความกรุณาจากอาจารย์ที่ปรึกษา รศ.ดร.ปิติเขต สู้รักษา ที่ให้ความช่วยเหลือ ให้คำชี้แนะ ช่วยแก้ปัญหา ตลอดจนให้ความรู้และประสบการณ์ที่ดีแก่ข้าพเจ้า

ขอขอบพระคุณ ผศ.นภพินทุ์ อนันตรศิริชัย ผศ.ดร.อรรถสิทธิ์ หล่าสกุล กรรมการสอบหัวข้อและเค้าโครงวิทยานิพนธ์ที่ได้กรุณาให้คำแนะนำตลอดจนข้อชี้แนะ จนในที่สุดทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลงได้

สุดท้ายต้องขอขอบคุณ คุณสุชาดา พิพัฒน์เมธินทร์ เพื่อนที่เป็นเสมือนคู่คิดและเป็นกำลังใจที่ดีตลอดมา

สำหรับคุณงามความดีอันใดที่เกิดจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้กับบิดามารดา ซึ่งเป็นที่รักและเคารพยิ่ง ตลอดจนครูอาจารย์ที่เคารพทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้และถ่ายทอดประสบการณ์ที่ดีให้แก่ข้าพเจ้า

อภิวิชญ์ แก้วนพรัตน์

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในการออกแบบระบบที่มีความซับซ้อน มักจะพบปัญหาในเรื่องความยากในการหาแบบจำลองทางคณิตศาสตร์และถ้าหากว่าแบบจำลองนั้นมีความผิดพลาดจะทำให้สมรรถนะของระบบควบคุมไม่ดีไปด้วย แต่สำหรับระบบควบคุมฟัซซีแบบจำลองทางคณิตศาสตร์จะถูกแทนด้วยกฎการควบคุมซึ่งอยู่ในรูปแบบที่สามารถทำความเข้าใจและปรับเปลี่ยนได้ง่ายโดยใช้ประสบการณ์จากผู้เชี่ยวชาญในกระบวนการนั้น ๆ [1] [2] ทำให้ในปัจจุบันการใช้งานระบบควบคุมฟัซซีกำลังได้รับความนิยมเป็นอย่างมาก

ที่ผ่านมามีการนำระบบควบคุมฟัซซีมาประยุกต์ใช้กับหุ่นยนต์ให้ทำงานในลักษณะต่าง ๆ มากมาย เช่น ใช้ในการสร้างแผนที่ [6] ใช้ให้เคลื่อนที่ตามแหล่งกำเนิดแสงที่กำหนด [9] ใช้เป็นระบบช่วยในการจอดรถ [11] ซึ่งสำหรับงานวิจัยที่เกี่ยวกับการเคลื่อนที่ของหุ่นยนต์ในงานสนามนั้นจะพบว่าปัญหาการติดตามเป้าหมายและปัญหาการหลบสิ่งกีดขวางเป็นปัญหาที่พบเป็นประจำ และงานวิจัยที่ผ่านมานั้นใช้วิธีควบคุมแบบฟัซซีที่มีการออกแบบฟังก์ชันสมาชิกที่ยุ่งยาก [3] [8] หรือไม่ก็มีตัวแปรทางภาษามากเกินความจำเป็น [4] [5] [13] มีการนำระบบการเรียนรู้มาใช้ [12] ทำให้ยุ่งยากยิ่งขึ้น และบางงานวิจัยใช้ตัวแปรอินพุตน้อยเกินไป [7] [10] ทำให้หุ่นยนต์ได้รับข้อมูลเพื่อใช้ตัดสินใจน้อยเกินไปทำให้การเคลื่อนที่ไม่มีประสิทธิภาพ ดังนั้นวิทยานิพนธ์ฉบับนี้จึงได้นำเสนอวิธีควบคุมฟัซซีที่เข้าใจง่าย ลดความซับซ้อนลง โดยที่ระบบควบคุมประกอบไปด้วยตัวแปรอินพุตเพียง 4 ตัว และตัวแปรเอาต์พุตเพียง 2 ตัว โดยมีฟังก์ชันความเป็นสมาชิกของอินพุตและเอาต์พุตเป็นรูปสามเหลี่ยมทำให้การปรับปรุงหรือเปลี่ยนแปลงทำได้ง่ายขึ้น ใช้ตัวตรวจวัดอัลตราโซนิคเพียง 3 ตัวเพื่อลดความซับซ้อนและต้นทุนการผลิตของฮาร์ดแวร์ลง ทำให้หุ่นยนต์ใช้เวลาในการประมวลผลน้อย มีการตอบสนองอย่างรวดเร็ว และเหมาะสมที่จะนำมาใช้กับการตัดสินใจของหุ่นยนต์ในการเคลื่อนที่หลบหลีกสิ่งกีดขวางไปยังเป้าหมายที่กำหนด

1.2 ความมุ่งหมายและวัตถุประสงค์ของการวิจัย

ความมุ่งหมายของงานวิจัยนี้ คือ เพื่อออกแบบจำลองการควบคุมให้หุ่นยนต์สามารถเคลื่อนที่ไปยังเป้าหมายได้ในทุก ๆ สถานการณ์หรือในทุก ๆ สภาวะแวดล้อมที่กำหนดให้

วัตถุประสงค์ของงานวิจัยนี้ คือ เพื่อศึกษาการทำงานของระบบควบคุมฟัซซีซึ่งปัจจุบันเป็นที่นิยมใช้งานอย่างกว้างขวางในอุตสาหกรรม ทำให้การศึกษาทางด้านนี้มีความน่าสนใจและ

สามารถนำระบบควบคุมฟัซซีมาประยุกต์กับกรณีของการเคลื่อนที่ของหุ่นยนต์เพื่อให้เกิดผลประโยชน์ในวงกว้าง เพราะในปัจจุบัน มีความนิยมใช้งานหุ่นยนต์อย่างกว้างขวางในอุตสาหกรรม และสำหรับหุ่นยนต์ที่จะต้องมีการเคลื่อนที่จากตำแหน่งหนึ่งไปสู่อีกตำแหน่งหนึ่ง จำเป็นต้องหลบหลีกสิ่งกีดขวางในระหว่างทาง ดังนั้น การพัฒนาหุ่นยนต์ให้มีความสามารถในการตัดสินใจจึงเป็นประโยชน์อย่างยิ่ง

1.3 ขอบเขตของการวิจัย

1. ออกแบบตัวควบคุมฟัซซีที่สามารถนำมาประยุกต์ใช้กับแบบจำลองของหุ่นยนต์เพื่อหลบหลีกสิ่งกีดขวาง และเคลื่อนที่ไปยังเป้าหมายที่กำหนดได้
2. จำลองการทำงานของหุ่นยนต์ในสภาวะแวดล้อมต่าง ๆ จากสถานการณ์ที่อาจเกิดขึ้นจริงได้
3. จำลองสถานการณ์ที่สภาวะแวดล้อมมีการเปลี่ยนแปลง เช่น สิ่งกีดขวางเคลื่อนที่ได้ เป้าหมายที่กำหนดสามารถเคลื่อนที่ได้ เป็นต้น

1.4 ขั้นตอนของการศึกษา

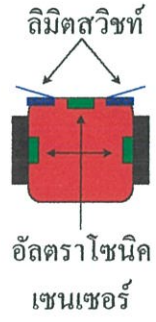
1. ค้นคว้าหาเอกสารและข้อมูลที่เกี่ยวข้อง
2. ศึกษาเอกสารและข้อมูลที่รวบรวมได้ พร้อมทั้งสร้างสมการคณิตศาสตร์สำหรับการเคลื่อนที่ของแบบจำลองของหุ่นยนต์
3. ออกแบบระบบควบคุมที่จะนำมาใช้กับแบบจำลองของหุ่นยนต์
4. ทำการทดลองระบบควบคุมที่ออกแบบไว้โดยจำลองสถานการณ์ต่าง ๆ ขึ้นโดยอาศัยโปรแกรมคอมพิวเตอร์
5. สรุปผลและเรียบเรียงวิทยานิพนธ์

บทที่ 2
ลักษณะกายภาพและแบบการจำลองคณิตศาสตร์ของหุ่นยนต์

2.1 การติดตั้งอัลตราโซนิคเซนเซอร์และลิมิตสวิชท์

หุ่นยนต์ที่เราออกแบบจะใช้อุปกรณ์ตัววัด 2 ชนิด คือ ลิมิตสวิชท์ ใช้ตรวจจับการชน และอัลตราโซนิคเซนเซอร์ ใช้ตรวจวัดระยะห่างระหว่างตัวหุ่นยนต์กับสิ่งกีดขวาง

โดยที่อัลตราโซนิคเซนเซอร์ จะใช้ทั้งหมด 3 ตัว คือ ทำหน้าที่วัดระยะทางระหว่างหุ่นยนต์กับสิ่งกีดขวางทางด้านซ้าย ด้านหน้า และทางด้านขวา โดยค่าที่ตรวจวัดได้นี้จะเป็นค่าที่ใช้เป็นตัวแปรอินพุทของตัวควบคุมฟัซซี ส่วนลิมิตสวิชท์ จะใช้ 2 ตัว ที่ด้านหน้าทางซ้ายหนึ่งตัวและด้านหน้าทางขวาอีกหนึ่งตัว โดยจะทำหน้าที่เป็นตัวเสริมเพื่อป้องกันข้อผิดพลาดที่อาจเกิดขึ้น ดังรูปที่ 2.1



รูปที่ 2.1 การติดตั้งอัลตราโซนิคเซนเซอร์และลิมิตสวิชท์ของหุ่นยนต์จำลอง

2.2 การทำงานของลิมิตสวิชท์

ลิมิตสวิชท์ด้านหน้าทางซ้ายจะทำงานเมื่อหุ่นยนต์เกิดการชนกับสิ่งกีดขวางทางด้านซ้าย ดังรูปที่ 2.2

รูปที่ 2.2 เมื่อหุ่นยนต์ชนกับสิ่งกีดขวางด้านซ้าย

ถ้าลิมิตสวิชท์ด้านซ้ายทำงานแล้วจะให้หุ่นยนต์ถอยหลังเล็กน้อยและหมุนไปทางขวา

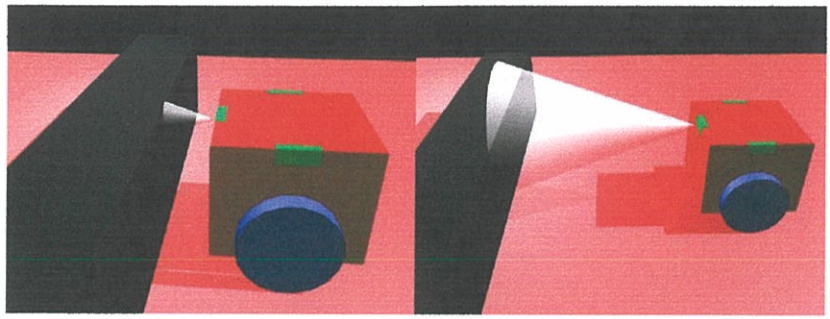
ลิมิตสวิชท์ด้านหน้าทางขวาจะทำงานเมื่อหุ่นยนต์เกิดการชนกับสิ่งกีดขวางทางด้านขวา ดังรูปที่ 2.3

รูปที่ 2.3 เมื่อหุ่นยนต์ชนกับสิ่งกีดขวางด้านขวา

ถ้าลิมิตสวิชท์ด้านขวาทำงานแล้วจะให้หุ่นยนต์ถอยหลังเล็กน้อยและหมุนไปทางซ้าย

2.3 การทำงานของอัลตราโซนิคเซนเซอร์

ในการเลือกใช้ตัวตรวดวัดอัลตราโซนิคต้องเลือกให้สามารถวัดระยะได้อย่างน้อยที่สุด 30 เซนติเมตร ในรูปที่ 2.4 แสดงการตรวจวัดระยะห่างระหว่างสิ่งกีดขวางทางด้านหน้ากับหุ่นยนต์โดยอาศัยอัลตราโซนิคที่อยู่ด้านหน้า



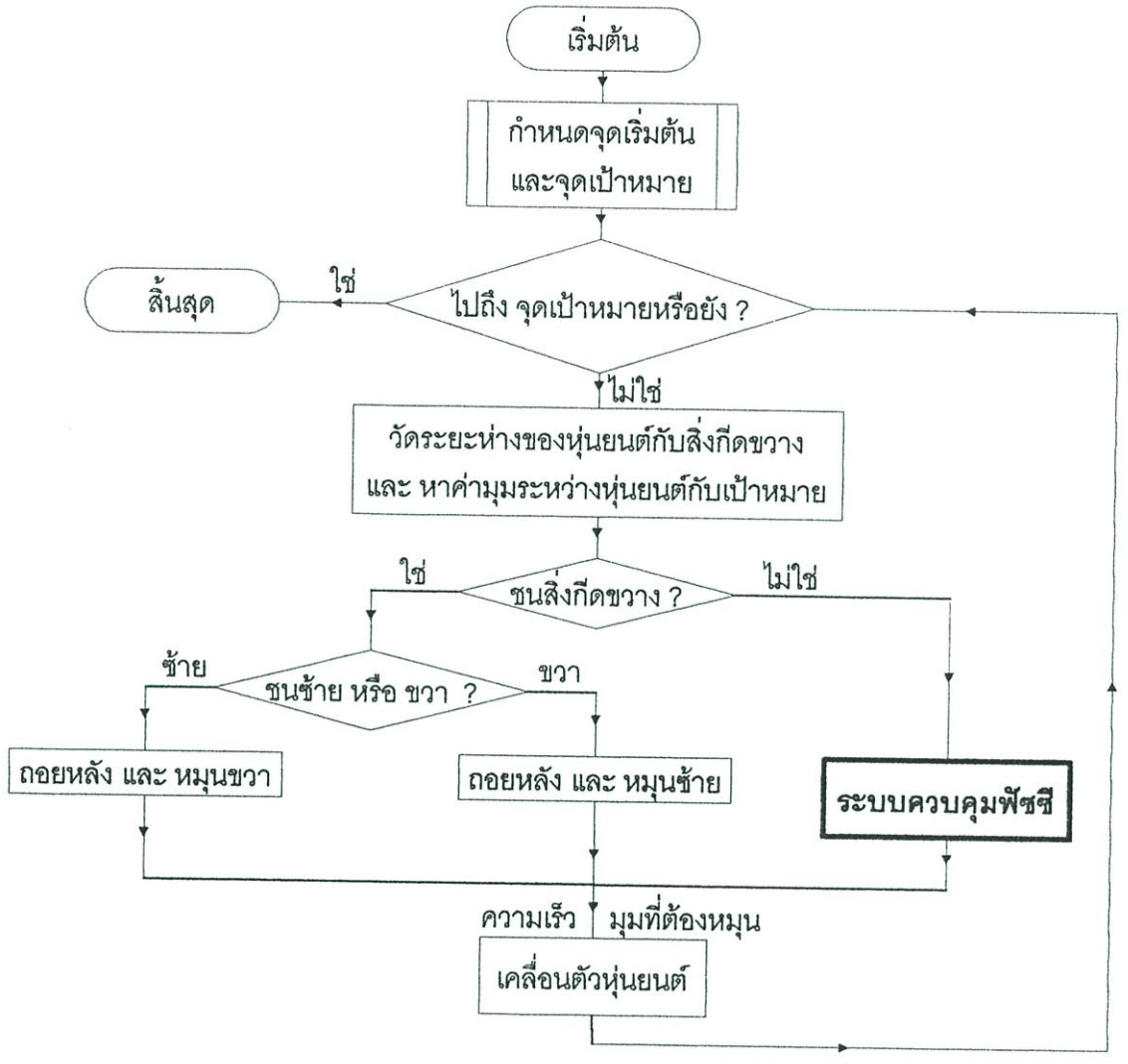
การตรวดวัดของอัลตราโซนิคที่ระยะห่างจากสิ่งแวดล้อม 5 เซนติเมตร

การตรวดวัดของอัลตราโซนิคที่ระยะห่างจากสิ่งแวดล้อม 30 เซนติเมตร

รูปที่ 2.4 การทำงานของอัลตราโซนิคด้านหน้า

2.4 การทำงานของโปรแกรมจำลองการเคลื่อนที่ของหุ่นยนต์

โครงสร้างภาพรวมของขั้นตอนการทำงานของโปรแกรมจำลองการเคลื่อนที่ของหุ่นยนต์แสดงดังรูปที่ 2.5 ในการทดสอบการทำงานได้จำลองตัวหุ่นยนต์ขนาดกว้าง 15 ซม. ยาว 15 ซม. เพื่อให้มีขนาดใกล้เคียงกับหุ่นยนต์ที่มีอยู่จริง ภายในพื้นที่จำกัดขนาด กว้าง 230 ซม. ยาว 150 ซม. เพื่อให้หุ่นยนต์คล้ายกับเคลื่อนที่อยู่ในสนามที่จำลองขึ้น



รูปที่ 2.5 ขั้นตอนการทำงานของโปรแกรมจำลองการเคลื่อนที่ของหุ่นยนต์

2.5 แบบจำลองคณิตศาสตร์และสมการการเคลื่อนที่ของหุ่นยนต์

ในขั้นตอนการตรวจสอบว่าหุ่นยนต์เคลื่อนที่ถึงเป้าหมายที่กำหนดให้แล้วหรือไม่ สามารถตรวจสอบได้จากการที่หุ่นยนต์เคลื่อนที่ไปเข้าใกล้เป้าหมายในระยะที่ยอมรับได้ ดังสมการที่ 2.1

$$\sqrt{(X_t - X_r)^2 + (Y_t - Y_r)^2} < \varepsilon \qquad (2.1)$$

โดยที่ ε คือ ระยะที่ยอมรับได้ ในที่นี้กำหนดให้มีค่าเท่ากับ 0.5 เซนติเมตร

(X_t, Y_t) คือ พิกัดของเป้าหมาย

(X_r, Y_r) คือ พิกัดของหุ่นยนต์

ส่วนในขั้นตอนการเคลื่อนตัวหุ่นยนต์ จะนำค่าความเร็วและมุมการหมุนของหุ่นยนต์ที่ได้จากการดีฟัซซิฟิเคชันมากำหนดตำแหน่งที่หุ่นยนต์จะเคลื่อนที่ไปและมุมของหุ่นยนต์ใหม่ ดังสมการที่ 2.2

$$\begin{aligned} X(t+T_s) &= X(t) + v_x^* T_s \\ Y(t+T_s) &= Y(t) + v_y^* T_s \\ \phi(t+T_s) &= \phi(t) + \theta_{Robot}^* \end{aligned} \tag{2.2}$$

โดยที่ T_s คือ อัตราการแซมปลิงของตรวจวัดอัลตราโซนิค มีค่า 0.1 วินาที

v_x^* คือ ความเร็วหุ่นยนต์ในแกน X หรือ $v_{Robot}^* \cos(\phi(t))$

v_y^* คือ ความเร็วหุ่นยนต์ในแกน Y หรือ $v_{Robot}^* \sin(\phi(t))$

$\phi(t)$ คือ มุมของหุ่นยนต์ก่อนหมุน

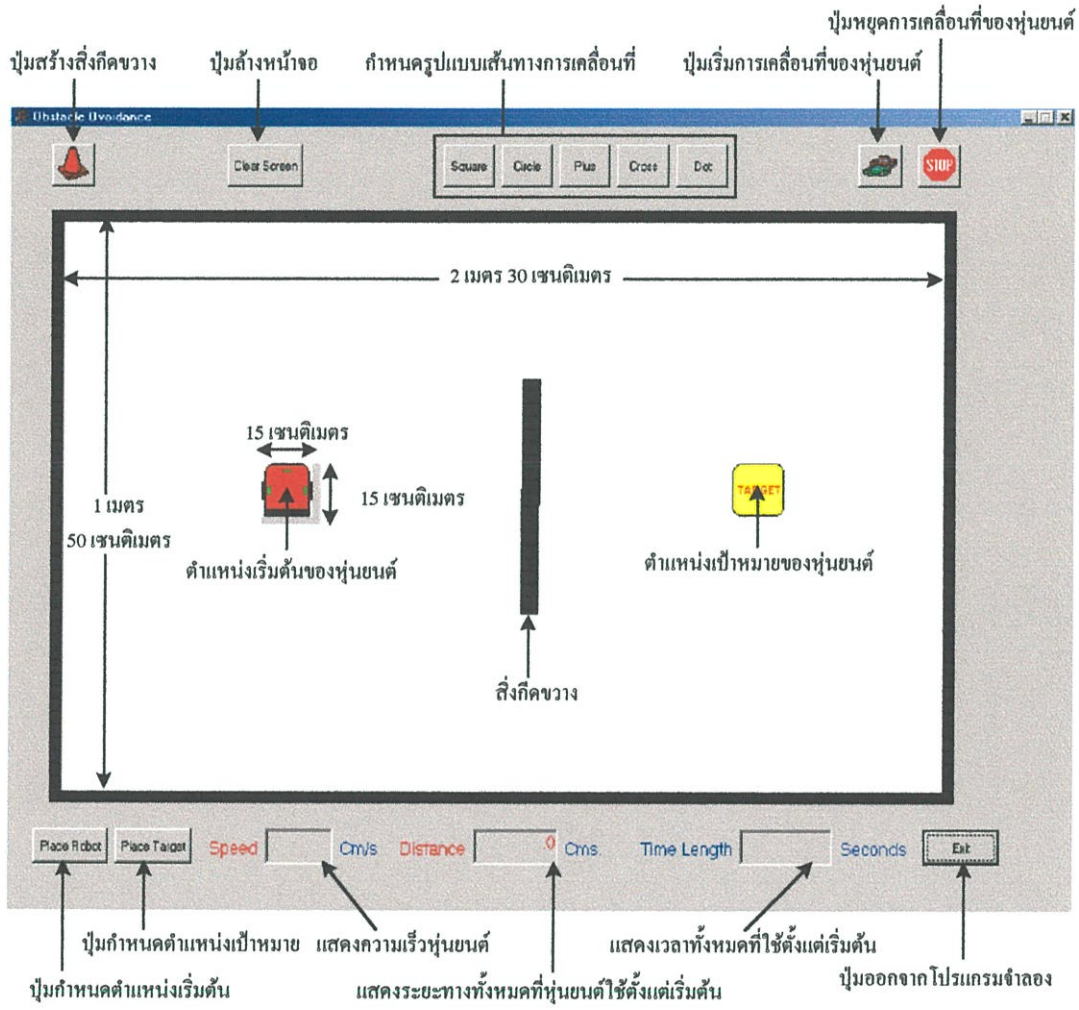
$\phi(t+T_s)$ คือ มุมของหุ่นยนต์หลังจากหมุนแล้ว

$(X(t), Y(t))$ คือ พิกัดของหุ่นยนต์ก่อนเคลื่อนที่

$(X(t+T_s), Y(t+T_s))$ คือ พิกัดที่หุ่นยนต์จะเคลื่อนที่ไป

2.6 หน้าต่างของโปรแกรมการจำลอง

เมื่อโปรแกรมที่ใช้จำลองการทำงานในการเคลื่อนที่ของหุ่นยนต์ เริ่มต้นทำงานจะมีหน้าจอดังรูปที่ 2.6



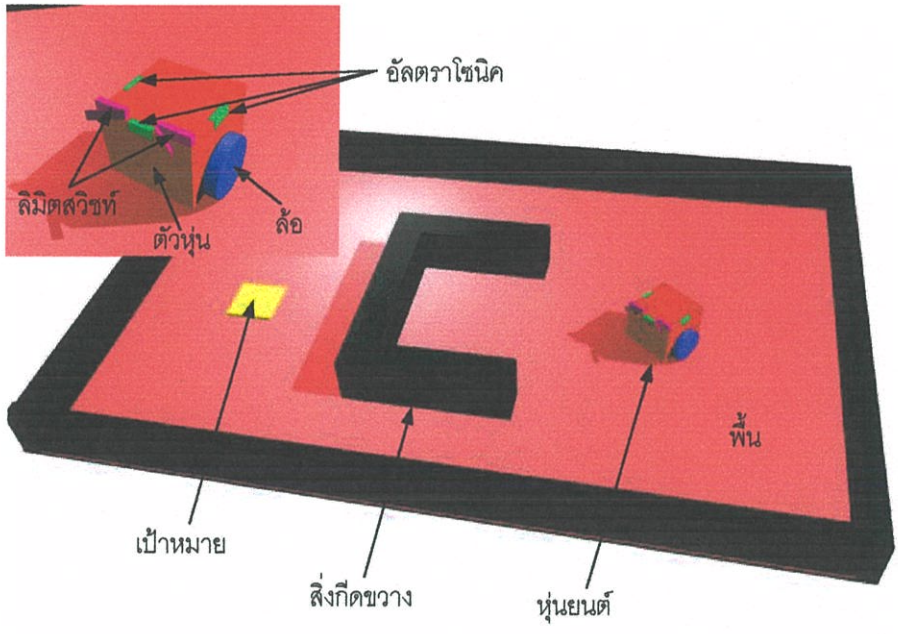
รูปที่ 2.6 หน้าจอเริ่มต้นของโปรแกรมที่ใช้จำลองการทำงานของหุ่นยนต์

ขั้นตอนในการใช้งานโปรแกรมสามารถอธิบายได้ดังนี้

1. สร้างสิ่งกีดขวางที่ต้องการ โดยการกดปุ่มสร้างสิ่งกีดขวางหลังจากนั้นทำการลากเมาส์ในพื้นที่การทำงานเพื่อวาดสิ่งกีดขวาง และเมื่อต้องการลบสิ่งกีดขวางที่วาดไว้ทำได้โดยกดปุ่มล้างหน้าจอ
2. กำหนดตำแหน่งเริ่มต้นของตัวหุ่นยนต์ โดยกดปุ่มกำหนดตำแหน่งเริ่มต้นแล้วคลิกตรงตำแหน่งที่ต้องการในพื้นที่การทำงาน

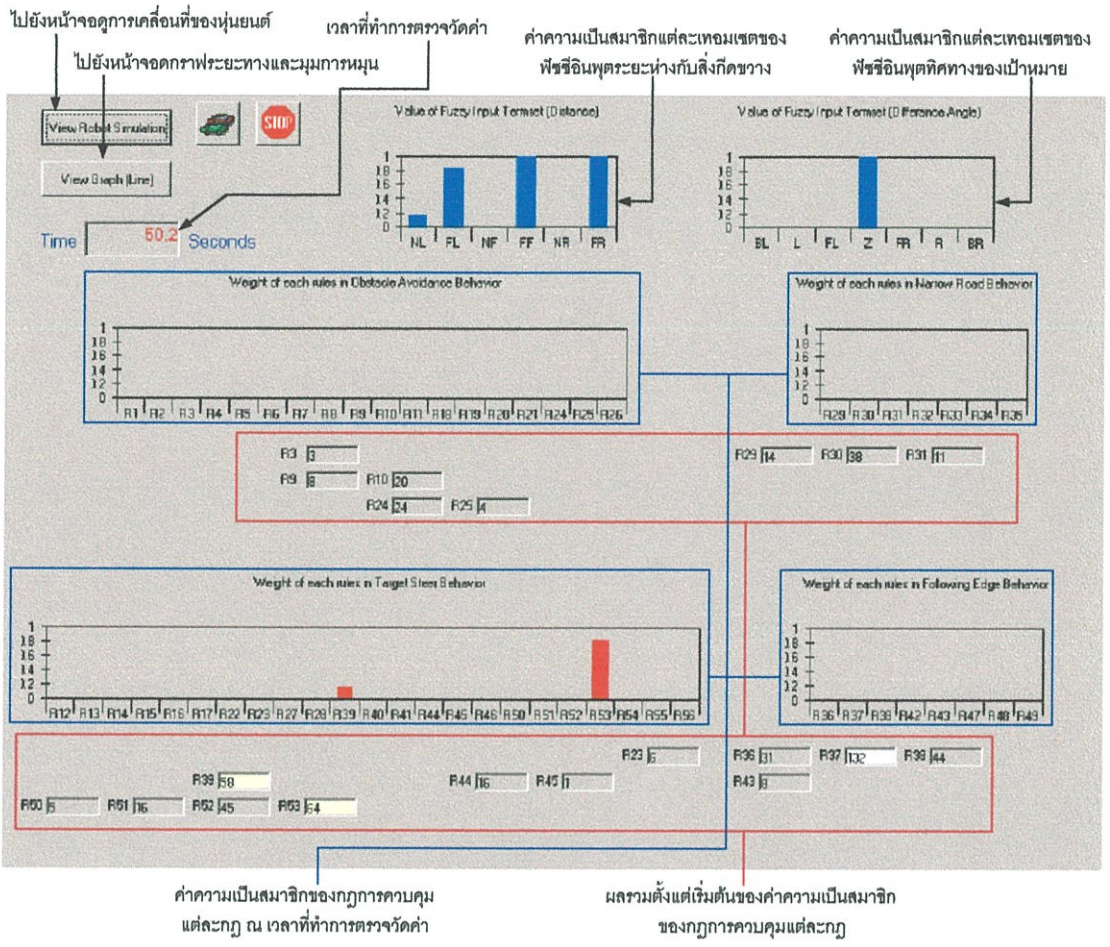
3. กำหนดตำแหน่งเป้าหมาย โดยกดปุ่มกำหนดตำแหน่งเป้าหมายแล้วคลิกตรงตำแหน่งที่ต้องการในพื้นที่การทำงาน
4. เริ่มต้นการทำงาน โดยกดปุ่มเริ่มการเคลื่อนที่ของหุ่นยนต์ ถ้าต้องการหยุดการทำงานทำได้โดยกดปุ่มหยุดการเคลื่อนที่ของหุ่นยนต์
5. ออกจากโปรแกรม โดยกดปุ่มออกจากโปรแกรมจำลอง

เพื่อให้เห็นถึงการนำโปรแกรมจำลองการเคลื่อนที่ของหุ่นยนต์ในรูปที่ 2.6 ไปใช้งานจริง จึงได้ทำการสร้างภาพจำลองเป็น 3 มิติขึ้นมาดังแสดงในรูปที่ 2.7

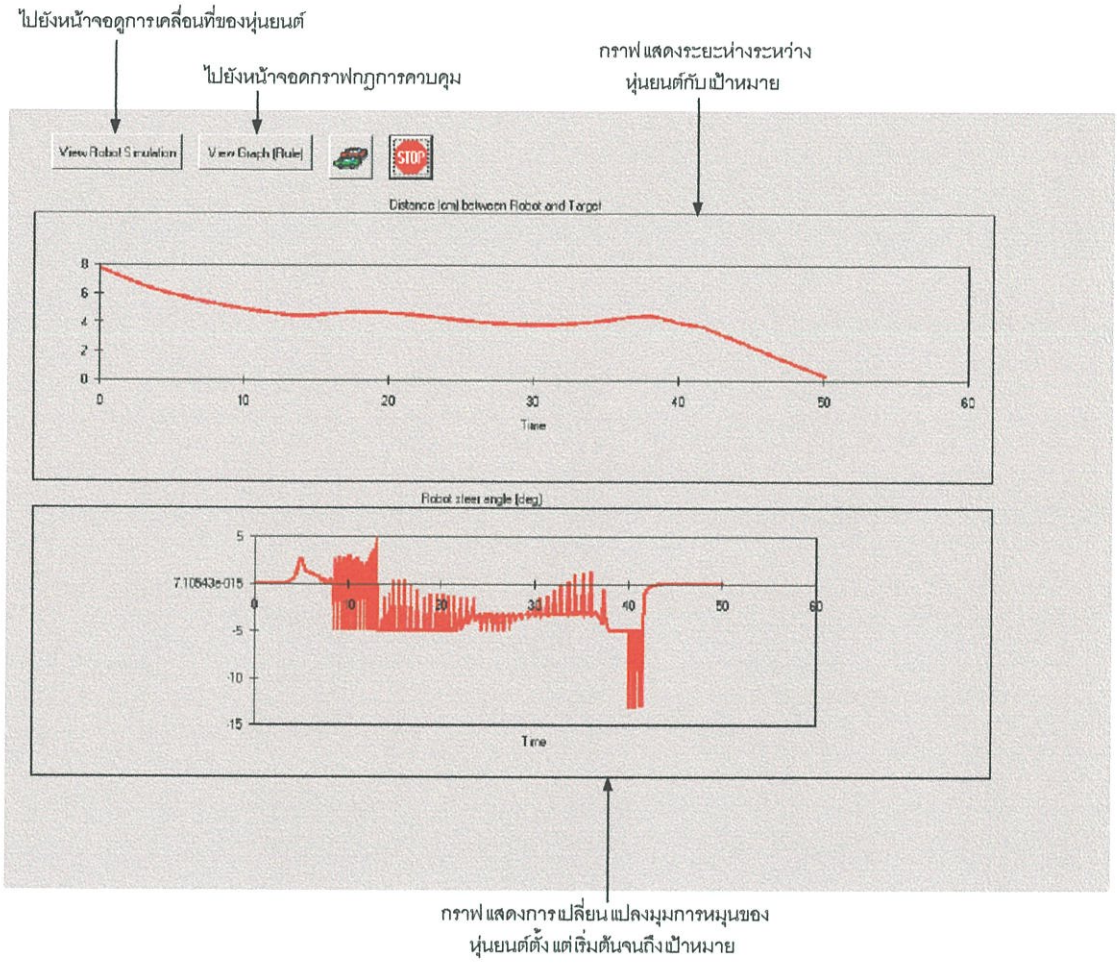


รูปที่ 2.7 หุ่นยนต์และสิ่งกีดขวางที่ใช้ในการจำลองการเคลื่อนที่ของหุ่นยนต์ในรูปแบบ 3 มิติ

นอกจากนี้ยังได้เขียนโปรแกรมในการตรวจดูค่าต่าง ๆ ที่ต้องการเพื่อช่วยให้สามารถตรวจสอบและแก้ไขโปรแกรมได้ง่ายขึ้น เช่น ตรวจดูค่าความเป็นสมาชิกของฟัซซีอินพุต ตรวจดูว่า ณ เวลานั้นมีกฎข้อใดบ้างที่นำมาใช้ในการตัดสินใจของหุ่นยนต์ และตรวจดูว่าตั้งแต่เริ่มต้นกฎการควบคุมแต่ละข้อนั้นมีกฎข้อใดบ้างที่นำใช้งานและข้อใดไม่ได้ใช้งาน และในกฎแต่ละข้อมีการใช้มากน้อยเพียงใด ดังแสดงในรูปที่ 2.8 และยังได้นำค่าระยะห่างระหว่างหุ่นยนต์กับเป้าหมายและการเปลี่ยนแปลงมุมการหมุนมาเขียนเป็นกราฟเพื่อตรวจดูการเคลื่อนที่ของหุ่นยนต์ว่าทำงานถูกต้องหรือไม่ ดังแสดงในรูปที่ 2.9



รูปที่ 2.8 หน้าจอแสดงการตรวจดูค่าความเป็นสมาชิกของฟัซซีอินพุตและกฎการควบคุม



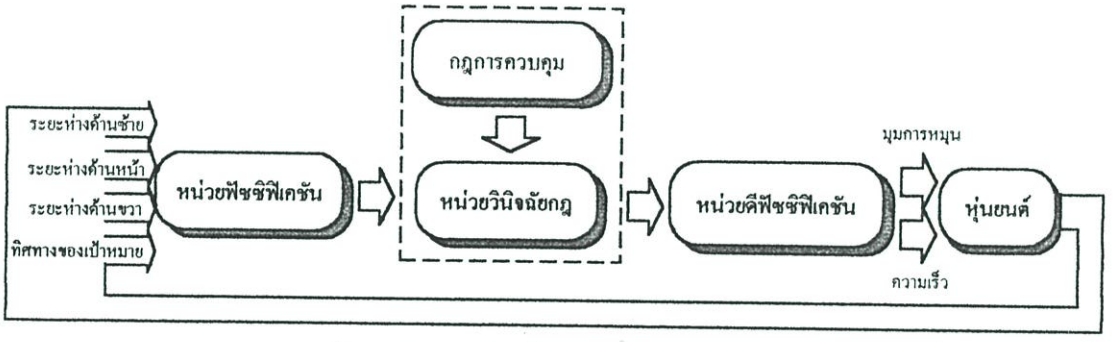
รูปที่ 2.9 หน้าจอแสดงกราฟระยะห่างระหว่างหุ่นยนต์กับเป้าหมายและกราฟการเปลี่ยนแปลงมุมการหมุน

บทที่ 3
การออกแบบตัวควบคุมฟัซซี

ระบบควบคุมฟัซซี จะมีโครงสร้างพื้นฐานที่สำคัญ 4 ส่วน คือ

1. การฟัซซิฟิเคชัน
2. กฎการควบคุมฟัซซี
3. หน่วยวินิจฉัยกฎ
4. การดีฟัซซิฟิเคชัน

แสดงดังรูปที่ 3.1



รูปที่ 3.1 โครงสร้างพื้นฐานของระบบควบคุมฟัซซี

3.1 การฟัซซิฟิเคชัน

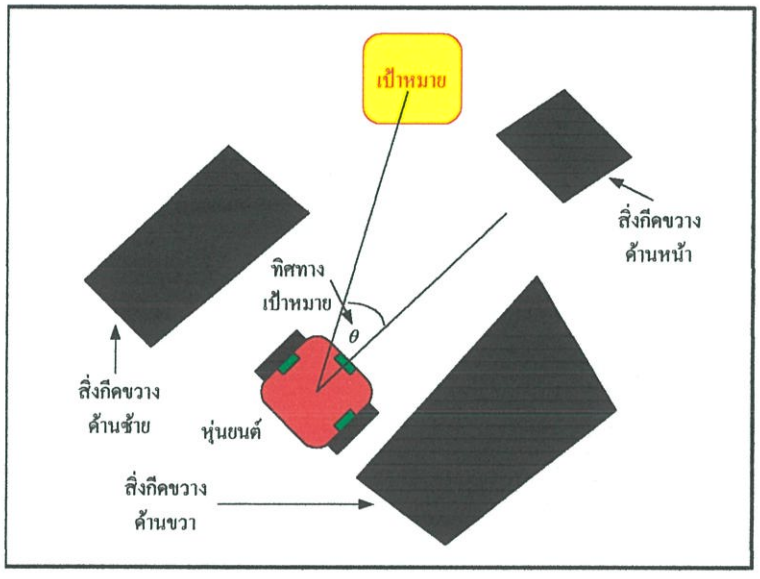
เป็นขั้นตอนการแปลงค่าตัวแปรอินพุต (Input Variables) และค่าตัวแปรเอาต์พุต (Output Variables) ให้อยู่ในรูปค่าความเป็นสมาชิกในระบบฟัซซีโดยใช้ฟังก์ชันความเป็นสมาชิก (Membership Functions) และ ตัวแปรทางภาษา (Linguistic Variables)

สำหรับการเลือกใช้ฟังก์ชันความเป็นสมาชิกนั้นต้องเลือกให้เหมาะสมกับคุณสมบัติของตัวแปร กล่าวคือ หากตัวแปรมีคุณสมบัติเป็นเชิงเส้น ควรจะเลือกใช้ฟังก์ชันความเป็นสมาชิกที่เป็นเชิงเส้น เช่น ฟังก์ชันรูปสามเหลี่ยม รูปสี่เหลี่ยมคางหมู แต่ถ้าหากว่าตัวแปรมีคุณสมบัติไม่เป็นเชิงเส้น ควรจะเลือกใช้ฟังก์ชันความเป็นสมาชิกที่ไม่เป็นเชิงเส้นด้วย เช่น ฟังก์ชันรูปตัว S, π เป็นต้น

ตัวแปรอินพุตที่ใช้ในการฟัซซิฟิเคชันมีด้วยกัน 4 ตัวแปร คือ

1. ระยะห่างระหว่างหุ่นยนต์กับสิ่งกีดขวางทางด้านซ้าย
2. ระยะห่างระหว่างหุ่นยนต์กับสิ่งกีดขวางทางด้านหน้า
3. ระยะห่างระหว่างหุ่นยนต์กับสิ่งกีดขวางทางด้านขวา
4. ทิศทางของเป้าหมาย

ซึ่งตัวแปร 3 ตัวแรกเป็นข้อมูลที่รับมาจากตัวตรวจวัดอัลตราโซนิค 3 ตัว ตามทิศทางที่ติดตั้ง กล่าวคือ อัลตราโซนิคตัวด้านหน้าจะวัดระยะห่างระหว่างหุ่นยนต์กับสิ่งกีดขวางทางด้านหน้า อัลตราโซนิคตัวด้านซ้ายจะวัดระยะห่างระหว่างหุ่นยนต์กับสิ่งกีดขวางทางด้านซ้าย อัลตราโซนิคตัวด้านขวาจะวัดระยะห่างระหว่างหุ่นยนต์กับสิ่งกีดขวางทางด้านขวา แสดงได้ดังรูปที่ 3.2



รูปที่ 3.2 แสดงการหาค่าตัวแปรอินพุตในขั้นตอนการฟัซซิฟิเคชัน

ส่วนตัวแปรทิศทางของเป้าหมาย หาได้จากสมการที่ 3.1 คือ

$$\theta = \phi(\text{robot}) - \phi(\text{target}) \qquad (3.1)$$

โดยที่ θ คือ ทิศทางของเป้าหมาย

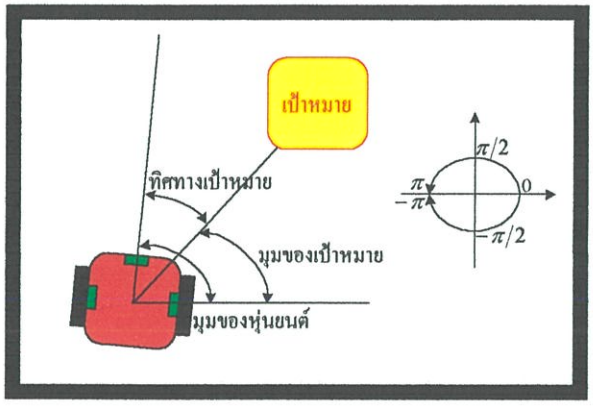
ϕ(robot) คือ มุมของหุ่นยนต์

ϕ(target) คือ มุมของเป้าหมาย

โดยที่ ถ้าทิศทางเป้าหมายมีค่าเป็นลบ แสดงว่าเป้าหมายอยู่ทางด้านซ้ายของหุ่นยนต์

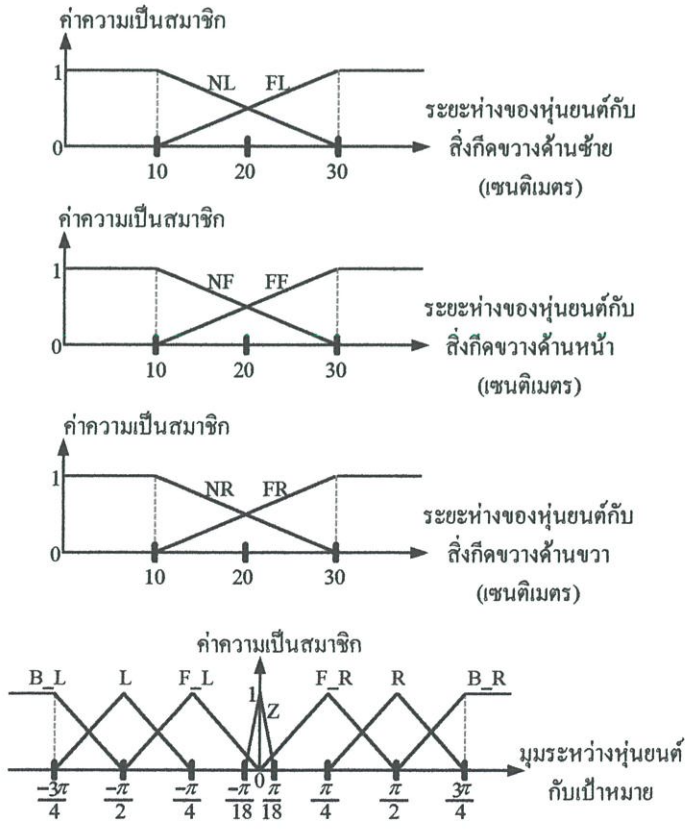
ถ้าทิศทางเป้าหมายมีค่าเป็นบวก แสดงว่าเป้าหมายอยู่ทางด้านขวาของหุ่นยนต์

แสดงการหาทิศทางเป้าหมายดังรูปที่ 3.3



รูปที่ 3.3 การหาทิศทางเป้าหมาย

ฟังก์ชันความเป็นสมาชิกของตัวแปรอินพุตแสดงดังรูปที่ 3.4 ฟังก์ชันความเป็นสมาชิกนี้สร้างขึ้นให้มีรูปแบบที่ง่ายโดยให้ฟังก์ชันเป็นสมการเชิงเส้น ค่าพารามิเตอร์ต่าง ๆ ขึ้นอยู่กับขนาดของหุ่นยนต์และประสบการณ์ของผู้ออกแบบว่าจะกำหนดให้ระยะห่างจากสิ่งกีดขวางแค่ไหนจึงจะถือว่าเข้าใกล้และระยะห่างแค่ไหนถือว่ายังไกลอยู่



รูปที่ 3.4 ฟังก์ชันสมาชิกของตัวแปรอินพุต

ตัวแปรทางภาษาของตัวแปรอินพุตมีความหมายดังนี้

- ตัวแปรระยะห่างด้านซ้าย ระยะห่างด้านหน้าและระยะห่างด้านขวา

NL:Near Left คือ เข้าใกล้สิ่งกีดขวางทางซ้าย , FL:Far Left คือ ไกลจากสิ่งกีดขวางทางซ้าย

NF:Near Front คือ เข้าใกล้สิ่งกีดขวางทางหน้า , FF:Far Front คือ ไกลจากสิ่งกีดขวางทางหน้า

NR:Near Right คือ เข้าใกล้สิ่งกีดขวางทางขวา , FR:Far Right คือ ไกลจากสิ่งกีดขวางทางขวา

สามารถเขียนสมการคณิตศาสตร์ของฟังก์ชันความเป็นสมาชิกของตัวแปรระยะห่างด้านซ้ายจากรูปที่ 3.4 โดยอาศัยหลักการของสมการเส้นตรงทั่วไป ได้ดังสมการที่ 3.2

$$\mu_{NL} = \begin{cases} 1 & ;\text{เมื่อ} \quad \text{ระยะห่างด้านซ้าย} < 10 \\ \dfrac{30 - \text{ระยะห่างด้านซ้าย}}{20} & ;\text{เมื่อ} \quad 10 < \text{ระยะห่างด้านซ้าย} < 30 \\ 0 & ;\text{เมื่อ} \quad \text{ระยะห่างด้านซ้าย} > 30 \end{cases}$$

$$\mu_{FL} = \begin{cases} 0 & ;\text{เมื่อ} \quad \text{ระยะห่างด้านซ้าย} < 10 \\ \dfrac{\text{ระยะห่างด้านซ้าย} - 10}{20} & ;\text{เมื่อ} \quad 10 < \text{ระยะห่างด้านซ้าย} < 30 \\ 1 & ;\text{เมื่อ} \quad \text{ระยะห่างด้านซ้าย} > 30 \end{cases} \qquad (3.2)$$

สามารถเขียนสมการคณิตศาสตร์ของฟังก์ชันความเป็นสมาชิกของตัวแปรระยะห่างด้านหน้าได้ดังสมการที่ 3.3

$$\mu_{NF} = \begin{cases} 1 & ;\text{เมื่อ} \quad \text{ระยะห่างด้านหน้า} < 10 \\ \dfrac{30 - \text{ระยะห่างด้านหน้า}}{20} & ;\text{เมื่อ} \quad 10 < \text{ระยะห่างด้านหน้า} < 30 \\ 0 & ;\text{เมื่อ} \quad \text{ระยะห่างด้านหน้า} > 30 \end{cases}$$

$$\mu_{FF} = \begin{cases} 0 & ;\text{เมื่อ} \quad \text{ระยะห่างด้านหน้า} < 10 \\ \dfrac{\text{ระยะห่างด้านหน้า} - 10}{20} & ;\text{เมื่อ} \quad 10 < \text{ระยะห่างด้านหน้า} < 30 \\ 1 & ;\text{เมื่อ} \quad \text{ระยะห่างด้านหน้า} > 30 \end{cases} \qquad (3.3)$$

สามารถเขียนสมการคณิตศาสตร์ของฟังก์ชันความเป็นสมาชิกของตัวแปรระยะห่างด้านขวา ได้ดังสมการที่ 3.4

$$\mu_{NR} = \begin{cases} 1 & ;\text{เมื่อ ระยะห่างด้านขวา} < 10 \\ \dfrac{30 - \text{ระยะห่างด้านขวา}}{20} & ;\text{เมื่อ } 10 < \text{ระยะห่างด้านขวา} < 30 \\ 0 & ;\text{เมื่อ ระยะห่างด้านขวา} > 30 \end{cases}$$

$$\mu_{FR} = \begin{cases} 0 & ;\text{เมื่อ ระยะห่างด้านขวา} < 10 \\ \dfrac{\text{ระยะห่างด้านขวา} - 10}{20} & ;\text{เมื่อ } 10 < \text{ระยะห่างด้านขวา} < 30 \\ 1 & ;\text{เมื่อ ระยะห่างด้านขวา} > 30 \end{cases} \quad (3.4)$$

- ตัวแปรทิศทางของเป้าหมาย

B_L (Back Left) : คือ เป้าหมายอยู่ด้านหลังทางซ้าย , L (Left) : คือ เป้าหมายอยู่ด้านซ้าย

F_L (Front Left) : คือ เป้าหมายอยู่ด้านหน้าทางซ้าย , Z (Zero) : คือ เป้าหมายอยู่ด้านหน้า

F_R (Front Right) : คือ เป้าหมายอยู่ด้านหน้าทางขวา, R (Right) : คือ เป้าหมายอยู่ด้านขวา

B_R (Back Right) : คือ เป้าหมายอยู่ด้านหลังทางขวา

สามารถเขียนสมการคณิตศาสตร์ของฟังก์ชันความเป็นสมาชิกของตัวแปร ทิศทางของเป้าหมายได้ดังสมการที่ 3.5

$$\mu_{B\_L} = \begin{cases} 1 & ;\text{เมื่อ ทิศทางเป้าหมาย} <= -3\pi/4 \\ ((-\pi/2) - \text{ทิศทางเป้าหมาย})/(\pi/4) & ;\text{เมื่อ } -3\pi/4 < \text{ทิศทางเป้าหมาย} < -\pi/2 \\ 0 & ;\text{เมื่อ } -\pi/2 <= \text{ทิศทางเป้าหมาย} \end{cases}$$

$$\mu_{L} = \begin{cases} (\text{ทิศทางเป้าหมาย} + 3\pi/4)/(\pi/4) & ;\text{เมื่อ } -3\pi/4 < \text{ทิศทางเป้าหมาย} < -\pi/2 \\ ((-\pi/4) - \text{ทิศทางเป้าหมาย})/(\pi/4) & ;\text{เมื่อ } -\pi/2 < \text{ทิศทางเป้าหมาย} < -\pi/4 \\ 0 & ;\text{เมื่อ ทิศทางเป้าหมาย นอกเหนือจากที่กำหนด} \end{cases}$$

$$\mu_{F\_L} = \begin{cases} (\text{ทิศทางเป้าหมาย} + \pi/2)/(\pi/4) & ;\text{เมื่อ } -\pi/2 < \text{ทิศทางเป้าหมาย} <= -\pi/4 \\ (-\text{ทิศทางเป้าหมาย})/(\pi/4) & ;\text{เมื่อ } -\pi/4 < \text{ทิศทางเป้าหมาย} < 0 \\ 0 & ;\text{เมื่อ ทิศทางเป้าหมาย นอกเหนือจากที่กำหนด} \end{cases}$$

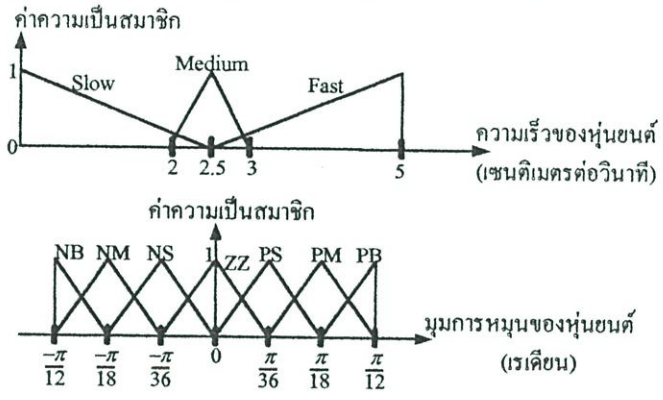
$$\mu_{Z} = \begin{cases} (\text{ทิศทางเป้าหมาย} - (-\pi/18))/(\pi/18)) & ;\text{เมื่อ } -\pi/18 < \text{ทิศทางเป้าหมาย} <= 0 \\ (\pi/18 - \text{ทิศทางเป้าหมาย}/(\pi/18)) & ;\text{เมื่อ } 0 < \text{ทิศทางเป้าหมาย} < \pi/18 \\ 0 & ;\text{เมื่อ ทิศทางเป้าหมาย นอกเหนือจากที่กำหนด} \end{cases} \quad (3.5)$$

$$\mu_{F\_R} = \begin{cases} (\text{ทิศทางเป้าหมาย})/(\pi/4) & ;\text{เมื่อ } 0 < \text{ทิศทางเป้าหมาย} <= \pi/4 \\ (\pi/2 - \text{ทิศทางเป้าหมาย})/(\pi/4) & ;\text{เมื่อ } \pi/4 < \text{ทิศทางเป้าหมาย} < \pi/2 \\ 0 & ;\text{เมื่อ ทิศทางเป้าหมาย นอกเหนือจากที่กำหนด} \end{cases}$$

$$\mu_{R} = \begin{cases} (\text{ทิศทางเป้าหมาย} - \pi/4)/(\pi/4) & ;\text{เมื่อ } \pi/4 < \text{ทิศทางเป้าหมาย} <= \pi/2 \\ (3\pi/4 - \text{ทิศทางเป้าหมาย})/(\pi/4) & ;\text{เมื่อ } \pi/2 < \text{ทิศทางเป้าหมาย} < 3\pi/4 \\ 0 & ;\text{เมื่อ ทิศทางเป้าหมาย นอกเหนือจากที่กำหนด} \end{cases}$$

$$\mu_{B\_R} = \begin{cases} 0 & ;\text{เมื่อ ทิศทางเป้าหมาย} <= \pi/2 \\ (\text{ทิศทางเป้าหมาย} - \pi/2)/(\pi/4) & ;\text{เมื่อ } \pi/2 < \text{ทิศทางเป้าหมาย} < 3\pi/4 \\ 1 & ;\text{เมื่อ } 3\pi/4 <= \text{ทิศทางเป้าหมาย} \end{cases}$$

ฟังก์ชันความเป็นสมาชิกของตัวแปรเอาต์พุตของระบบควบคุมฟัซซีแสดงดังรูปที่ 3.5



รูปที่ 3.5 แสดงฟังก์ชันสมาชิกของตัวแปรเอาต์พุต

ตัวแปรทางภาษาของตัวแปรเอาต์พุต มีความหมายดังนี้

- **ตัวแปรความเร็วของหุ่นยนต์**

  Slow คือ กำหนดให้หุ่นยนต์เคลื่อนที่ช้า

  Medium คือ กำหนดให้หุ่นยนต์เคลื่อนที่ปานกลาง

  Fast คือ กำหนดให้หุ่นยนต์เคลื่อนที่เร็ว

- **ตัวแปรมุมการหมุนของหุ่นยนต์**

  NB : Negative Big คือ กำหนดให้หุ่นยนต์หมุนไปทางซ้ายมาก

  NM: Negative Medium คือ กำหนดให้หุ่นยนต์หมุนไปทางซ้ายปานกลาง

  NS : Negative Small คือ กำหนดให้หุ่นยนต์หมุนไปทางซ้ายเล็กน้อย

  ZZ : Zero คือ กำหนดให้หุ่นยนต์ตรงไปข้างหน้า

  PS : Positive Small คือ กำหนดให้หุ่นยนต์หมุนไปทางขวาเล็กน้อย

  PM: Positive Medium คือ กำหนดให้หุ่นยนต์หมุนไปทางขวาปานกลาง

  PB : Positive Big คือ กำหนดให้หุ่นยนต์หมุนไปทางขวามาก ๆ

สามารถเขียนสมการคณิตศาสตร์ของฟังก์ชันความเป็นสมาชิกของตัวแปรความเร็วของหุ่นยนต์ได้ดังสมการที่ 3.6

$$\mu_{Slow} = \begin{cases} 1 & ;\text{เมื่อ ความเร็วหุ่นยนต์} = 0 \\ \dfrac{0.25 - \text{ความเร็วหุ่นยนต์}}{0.25} & ;\text{เมื่อ } 0 < \text{ความเร็วหุ่นยนต์} < 0.25 \\ 0 & ;\text{เมื่อ ความเร็วหุ่นยนต์} \Rightarrow 0.25 \end{cases}$$

$$\mu_{Medium} = \begin{cases} (\text{ความเร็วหุ่นยนต์} - 0.2) / 0.05 & ;\text{เมื่อ } 0.2 < \text{ความเร็วหุ่นยนต์} < 0.25 \\ \dfrac{0.3 - \text{ความเร็วหุ่นยนต์}}{0.05} & ;\text{เมื่อ } 0.25 < \text{ความเร็วหุ่นยนต์} < 0.3 \\ 0 & ;\text{เมื่อ ความเร็วหุ่นยนต์ นอกเหนือจากที่กำหนด} \end{cases} \quad (3.6)$$

$$\mu_{Fast} = \begin{cases} 0 & ;\text{เมื่อ ความเร็วหุ่นยนต์} \le 0.25 \\ \dfrac{\text{ความเร็วหุ่นยนต์} - 0.25}{0.25} & ;\text{เมื่อ } 0.25 < \text{ความเร็วหุ่นยนต์} < 0.3 \\ 1 & ;\text{เมื่อ ความเร็วหุ่นยนต์} = 0.3 \end{cases}$$

สามารถเขียนสมการคณิตศาสตร์ของฟังก์ชันความเป็นสมาชิกของตัวแปรมุมการหมุนของหุ่นยนต์ได้ดังสมการที่ 3.7

$$\mu_{NB} = \begin{cases} 1 & ;\text{เมื่อ มุมการหมุน} = -(\pi/12) \\ ((-\pi/18) - \text{มุมการหมุน})/(\pi/36) & ;\text{เมื่อ } -(\pi/12) < \text{มุมการหมุน} < -(\pi/18) \\ 0 & ;\text{เมื่อ มุมการหมุน นอกเหนือจากที่กำหนด} \end{cases}$$

$$\mu_{NM} = \begin{cases} (\text{มุมการหมุน} + (\pi/12))/(\pi/36) & ;\text{เมื่อ } -(\pi/12) < \text{มุมการหมุน} \le -(\pi/18) \\ ((-\pi/36) - \text{มุมการหมุน})/(\pi/36) & ;\text{เมื่อ } -(\pi/18) < \text{มุมการหมุน} < -(\pi/36) \\ 0 & ;\text{เมื่อ มุมการหมุน นอกเหนือจากที่กำหนด} \end{cases}$$

$$\mu_{NS} = \begin{cases} (\text{มุมการหมุน} + (\pi/18))/(\pi/36) & ;\text{เมื่อ } -(\pi/18) < \text{มุมการหมุน} \le -(\pi/36) \\ (-\text{มุมการหมุน})/(\pi/36) & ;\text{เมื่อ } -(\pi/36) < \text{มุมการหมุน} < 0 \\ 0 & ;\text{เมื่อ มุมการหมุน นอกเหนือจากที่กำหนด} \end{cases}$$

$$\mu_{ZZ} = \begin{cases} (\text{มุมการหมุน} + (\pi/36))/(\pi/36) & ;\text{เมื่อ } -(\pi/36) < \text{มุมการหมุน} \le 0 \\ ((\pi/36) - \text{มุมการหมุน})/(\pi/36) & ;\text{เมื่อ } 0 < \text{มุมการหมุน} < (\pi/36) \\ 0 & ;\text{เมื่อ มุมการหมุน นอกเหนือจากที่กำหนด} \end{cases} \quad (3.7)$$

$$\mu_{PS} = \begin{cases} (\text{มุมการหมุน})/(\pi/36) & ;\text{เมื่อ } 0 < \text{มุมการหมุน} \le (\pi/36) \\ ((\pi/18) - \text{มุมการหมุน})/(\pi/36) & ;\text{เมื่อ } (\pi/36) < \text{มุมการหมุน} < (\pi/18) \\ 0 & ;\text{เมื่อ มุมการหมุน นอกเหนือจากที่กำหนด} \end{cases}$$

$$\mu_{PM} = \begin{cases} (\text{มุมการหมุน} - (\pi/36))/(\pi/36) & ;\text{เมื่อ } (\pi/36) < \text{มุมการหมุน} \le (\pi/18) \\ ((\pi/12) - \text{มุมการหมุน})/(\pi/36) & ;\text{เมื่อ } (\pi/18) < \text{มุมการหมุน} < (\pi/12) \\ 0 & ;\text{เมื่อ มุมการหมุน นอกเหนือจากที่กำหนด} \end{cases}$$

$$\mu_{PB} = \begin{cases} 1 & ;\text{เมื่อ มุมการหมุน} = (\pi/12) \\ (\text{มุมการหมุน} - (\pi/18))/(\pi/36) & ;\text{เมื่อ } (\pi/18) < \text{มุมการหมุน} < (\pi/12) \\ 0 & ;\text{เมื่อ มุมการหมุน นอกเหนือจากที่กำหนด} \end{cases}$$

3.2 การสร้างกฎการควบคุม และ การวินิจฉัยกฎ

การกำหนดกฎการควบคุมแบบฟัซซีนิยมให้อยู่ในรูปแบบของ IF-THEN Rules ซึ่งแสดงความสัมพันธ์ระหว่างค่าอินพุตและเอาต์พุต รูปแบบโดยทั่วไปของกฎควบคุมฟัซซี ในกรณีของระบบที่มีหลายอินพุตหนึ่งเอาต์พุต (Multi-Input Single-Output System MISO) คือ

$$R^J : \text{IF } x \text{ is } A_J \ldots, \text{ AND } y \text{ is } B_J \text{ , THEN } z = C_J \text{ , } J = 1,2,3,\ldots n$$

โดยที่ x,..., y และ z เป็นตัวแปรฟัซซี A_J ,...,B_J และ C_J เป็นเทอมเซตที่สอดคล้องกับตัวแปรฟัซซี x , y และ z ที่อยู่ในเอกภพ U,...,V และ W ตามลำดับ ประพจน์ฟัซซี x is A_J ..., AND y is B_J ที่ตามหลัง IF นั้นเป็นส่วนของเงื่อนไขที่เรียกว่า "Precondition" หรือส่วนของอินพุตฟัซซี ส่วนประพจน์ฟัซซี $z = C_J$ ที่ตามหลัง THEN นั้นเป็นส่วนผลเนื่องจากเงื่อนไขที่เรียกว่า "Consequent" หรือเป็นส่วนของเอาต์พุตฟัซซี

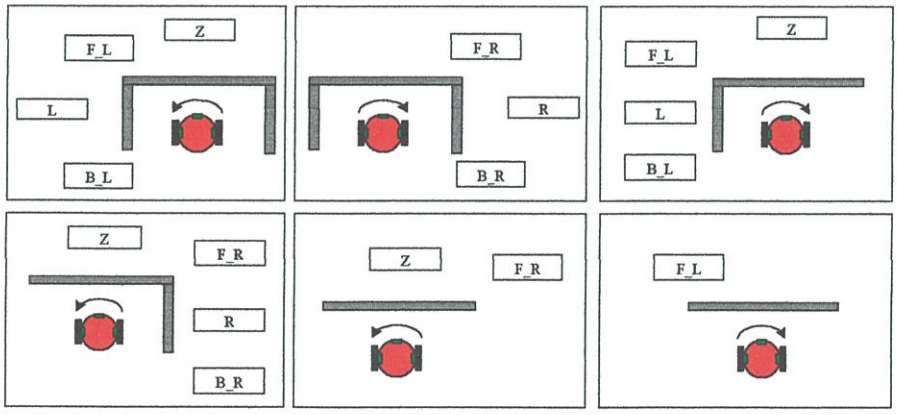
การสร้างกฎที่ใช้ควบคุมระบบจะต้องใช้ทั้งความรู้และประสบการณ์ของผู้ออกแบบ สำหรับจำนวนกฎในระบบควบคุมฟัซซีนั้นขึ้นอยู่กับจำนวนตัวแปรอินพุตและจำนวนตัวแปรทางภาษาหรือเทอมเซตของตัวแปรอินพุตแต่ละตัว ซึ่งในที่นี้ กฎการควบคุมจะมีด้วยกันทั้งหมด 56 กฎ (2x2x2x7 ระยะห่างด้านซ้าย 2 เทอมเซต ระยะห่างด้านหน้า 2 เทอมเซต ระยะห่างด้านขวา 2 เทอมเซต ทิศทางของเป้าหมาย 7 เทอมเซต)

3.2.1 การสร้างกฎการควบคุม (Fuzzy Rules Base)

สร้างจากพฤติกรรมของหุ่นยนต์ที่จะกระทำกับสิ่งกีดขวางในสภาวะแวดล้อมขณะนั้น แบ่งออกได้เป็น 4 พฤติกรรม คือ

- พฤติกรรมเคลื่อนที่เพื่อหลบหลีกสิ่งกีดขวาง

เมื่อหุ่นยนต์เคลื่อนที่เข้าใกล้สิ่งกีดขวางดังรูปที่ 3.6 หุ่นยนต์ควรจะลดความเร็วและเคลื่อนที่เพื่อหลบสิ่งกีดขวางให้ได้ กฎการควบคุมสำหรับพฤติกรรมนี้ แสดงไว้ในตารางที่ 3.1



รูปที่ 3.6 สภาวะแวดล้อมที่ทำให้หุ่นยนต์เกิดพฤติกรรมเคลื่อนที่เพื่อหลบหลีกสิ่งกีดขวาง

ตารางที่ 3.1 กฎการควบคุมเมื่อหุ่นยนต์ มีพฤติกรรมเคลื่อนที่เพื่อหลบหลีกสิ่งกีดขวาง

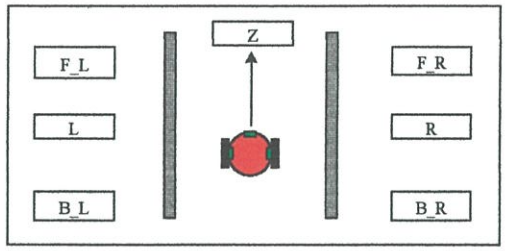
พฤติกรรม เคลื่อนที่เพื่อหลบหลีกสิ่งกีดขวาง						
กฎข้อที่	ระยะห่างด้านหน้า	ระยะห่างด้านซ้าย	ระยะห่างด้านขวา	ทิศทางของเป้าหมาย	มุมการหมุน	ความเร็ว
1	NF	NL	NR	B_L	NB	Slow
2	NF	NL	NR	L	NB	Slow
3	NF	NL	NR	F_L	NB	Slow
4	NF	NL	NR	Z	NM	Slow
5	NF	NL	NR	F_R	PB	Slow
6	NF	NL	NR	R	PB	Slow
7	NF	NL	NR	B_R	PB	Slow
8	NF	NL	FR	B_L	PS	Medium
9	NF	NL	FR	L	PS	Medium
10	NF	NL	FR	F_L	PS	Medium
11	NF	NL	FR	Z	PS	Medium
18	NF	FL	NR	Z	NS	Medium
19	NF	FL	NR	F_R	NS	Medium
20	NF	FL	NR	R	NS	Medium
21	NF	FL	NR	B_R	NS	Medium
24	NF	FL	FR	F_L	PS	Slow
25	NF	FL	FR	Z	ZZ	Slow
26	NF	FL	FR	F_R	NS	Slow

ยกตัวอย่างเพื่อความเข้าใจสำหรับวิธีการเขียนกฎการควบคุมสำหรับกฎข้อที่ 1 จากตารางที่ 3.1 ได้ดังนี้

ถ้า ระยะห่างด้านหน้า เป็น NF ระยะห่างด้านซ้าย เป็น NL
ระยะห่างด้านขวา เป็น NR ทิศทางของเป้าหมาย เป็น B_L
แล้ว มุมการหมุน เป็น NB และ ความเร็ว เป็น Slow

- พฤติกรรมเคลื่อนที่ในทางแคบ ๆ

เมื่อหุ่นยนต์เคลื่อนที่ในทางแคบ ๆ ดังรูปที่ 3.7 หุ่นยนต์ควรวิ่งความเร็วปกติและเคลื่อนที่ไปข้างหน้าโดยที่ไม่สนใจถึงเป้าหมาย กฎการควบคุมสำหรับพฤติกรรมนี้ แสดงไว้ในตารางที่ 3.2



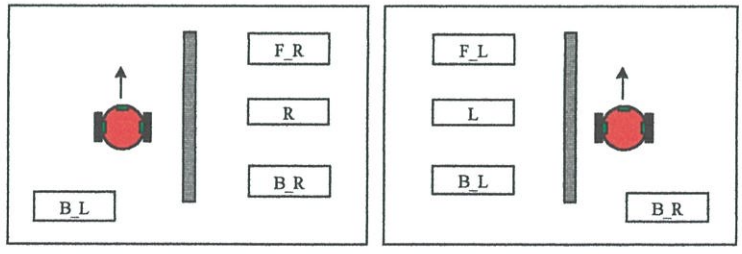
รูปที่ 3.7 สภาวะแวดล้อมที่ทำให้หุ่นยนต์เกิดพฤติกรรมเคลื่อนที่ในทางแคบ ๆ

ตารางที่ 3.2 กฎการควบคุมเมื่อหุ่นยนต์ มีพฤติกรรมเคลื่อนที่ในทางแคบ ๆ

พฤติกรรม เคลื่อนที่ในทางแคบ ๆ						
กฎข้อที่	ระยะห่างด้านหน้า	ระยะห่างด้านซ้าย	ระยะห่างด้านขวา	ทิศทางของเป้าหมาย	มุมการหมุน	ความเร็ว
29	FF	NL	NR	B_L	ZZ	Medium
30	FF	NL	NR	L	ZZ	Medium
31	FF	NL	NR	F_L	ZZ	Medium
32	FF	NL	NR	Z	ZZ	Fast
33	FF	NL	NR	F_R	ZZ	Medium
34	FF	NL	NR	R	ZZ	Medium
35	FF	NL	NR	B_R	ZZ	Medium

- พฤติกรรมเคลื่อนที่ไปตามขอบ

เมื่อเป้าหมายอยู่ฝั่งตรงข้ามสิ่งกีดขวาง ดังรูปที่ 3.8 ดังนั้น เพื่อให้หุ่นยนต์วิ่งไปหาเป้าหมายได้ หุ่นยนต์ควรจะวิ่งไปตามแนวของขอบของสิ่งกีดขวางไปเรื่อย ๆ โดยที่หุ่นยนต์สามารถเคลื่อนที่ด้วยความเร็วสูง ๆ ได้ กฎการควบคุมสำหรับพฤติกรรมนี้ แสดงไว้ในตารางที่ 3.3



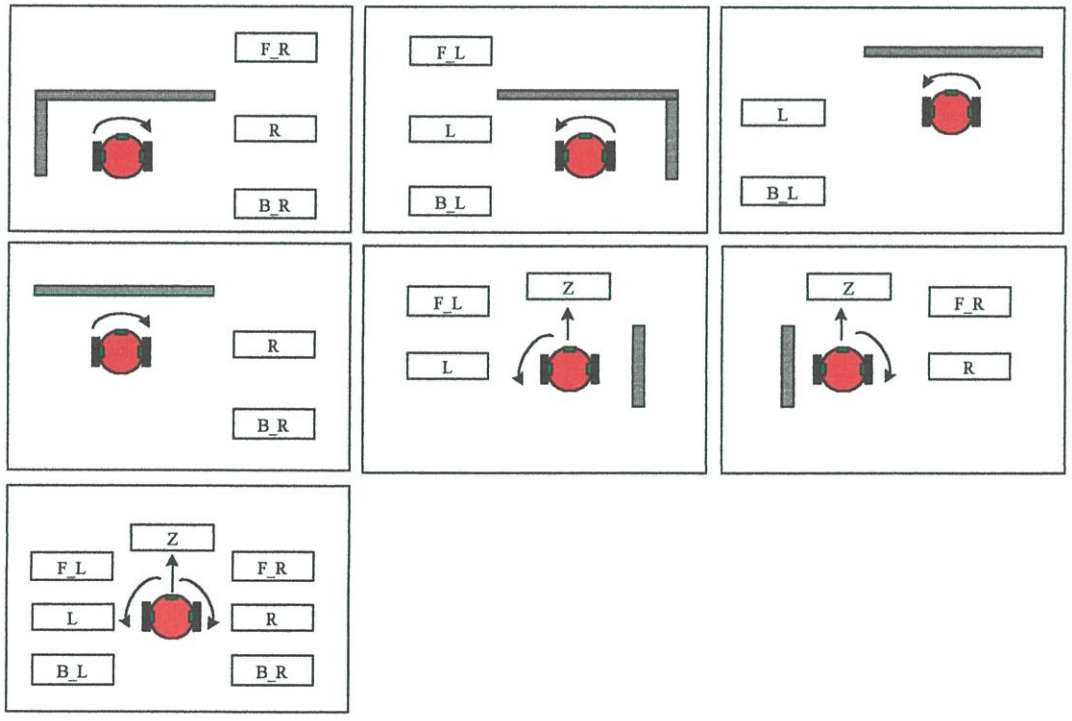
รูปที่ 3.8 สภาวะแวดล้อมที่ทำให้หุ่นยนต์เกิดพฤติกรรมเคลื่อนที่ไปตามขอบ

ตารางที่ 3.3 กฎการควบคุมเมื่อหุ่นยนต์ มีพฤติกรรมเคลื่อนที่ไปตามขอบ

พฤติกรรม เคลื่อนที่ไปตามขอบ						
กฎข้อที่	ระยะห่างด้านหน้า	ระยะห่างด้านซ้าย	ระยะห่างด้านขวา	ทิศทางของเป้าหมาย	มุมการหมุน	ความเร็ว
36	FF	NL	FR	B_L	ZZ	Fast
37	FF	NL	FR	L	ZZ	Fast
38	FF	NL	FR	F_L	ZZ	Fast
42	FF	NL	FR	B_R	NS	Fast
43	FF	FL	NR	B_L	NS	Fast
47	FF	FL	NR	F_R	ZZ	Fast
48	FF	FL	NR	R	ZZ	Fast
49	FF	FL	NR	B_R	ZZ	Fast

- พฤติกรรมเคลื่อนที่ไปยังเป้าหมาย

เมื่อไม่มีสิ่งกีดขวางระหว่างหุ่นยนต์กับเป้าหมาย หรือ ไม่มีสิ่งกีดขวางรอบ ๆ ตัวหุ่นยนต์ ดังรูปที่ 3.9 ดังนั้นจุดประสงค์หลักสำหรับสถานะการณ์นี้ คือให้หุ่นยนต์เคลื่อนที่ไปยังตำแหน่งเป้าหมาย และเพิ่มความเร็วให้มากที่สุดเท่าที่จะทำได้ กฎการควบคุมสำหรับพฤติกรรมนี้ แสดงไว้ในตารางที่ 3.4



รูปที่ 3.9 สภาวะแวดล้อมที่ทำให้หุ่นยนต์เกิดพฤติกรรมเคลื่อนที่ไปยังเป้าหมาย

ตารางที่ 3.4 กฎการควบคุมเมื่อหุ่นยนต์ มีพฤติกรรมเคลื่อนที่ไปยังเป้าหมาย

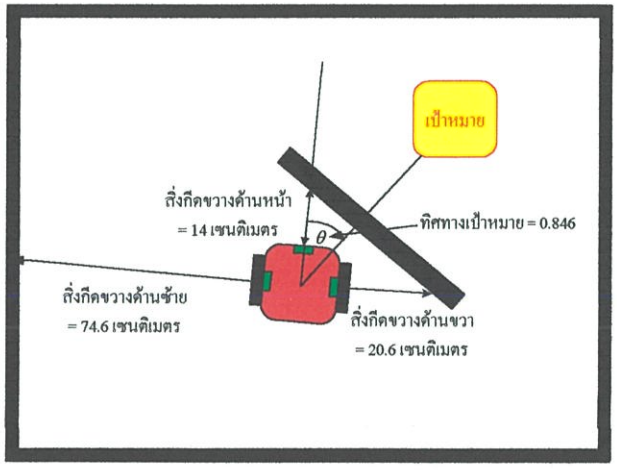
พฤติกรรม เคลื่อนที่ไปยังเป้าหมาย						
กฎข้อที่	ระยะห่างด้านหน้า	ระยะห่างด้านซ้าย	ระยะห่างด้านขวา	ทิศทางของเป้าหมาย	มุมการหมุน	ความเร็ว
12	NF	NL	FR	F_R	PS	Fast
13	NF	NL	FR	R	PM	Fast
14	NF	NL	FR	B_R	PB	Fast
15	NF	FL	NR	B_L	NB	Fast
16	NF	FL	NR	L	NM	Fast
17	NF	FL	NR	F_L	NS	Fast
22	NF	FL	FR	B_L	NB	Fast
23	NF	FL	FR	L	NM	Fast
27	NF	FL	FR	R	PM	Fast
28	NF	FL	FR	B_R	PB	Fast
39	FF	NL	FR	Z	ZZ	Fast
40	FF	NL	FR	F_R	PS	Fast
41	FF	NL	FR	R	PS	Fast
44	FF	FL	NR	L	NS	Fast
45	FF	FL	NR	F_L	NS	Fast
46	FF	FL	NR	Z	ZZ	Fast
50	FF	FL	FR	B_L	NS	Fast
51	FF	FL	FR	L	NS	Fast
52	FF	FL	FR	F_L	NS	Fast
53	FF	FL	FR	Z	ZZ	Fast
54	FF	FL	FR	F_R	PS	Fast
55	FF	FL	FR	R	PS	Fast
56	FF	FL	FR	B_R	PS	Fast

3.2.2 ส่วนการวินิจฉัย (Inference Engine)

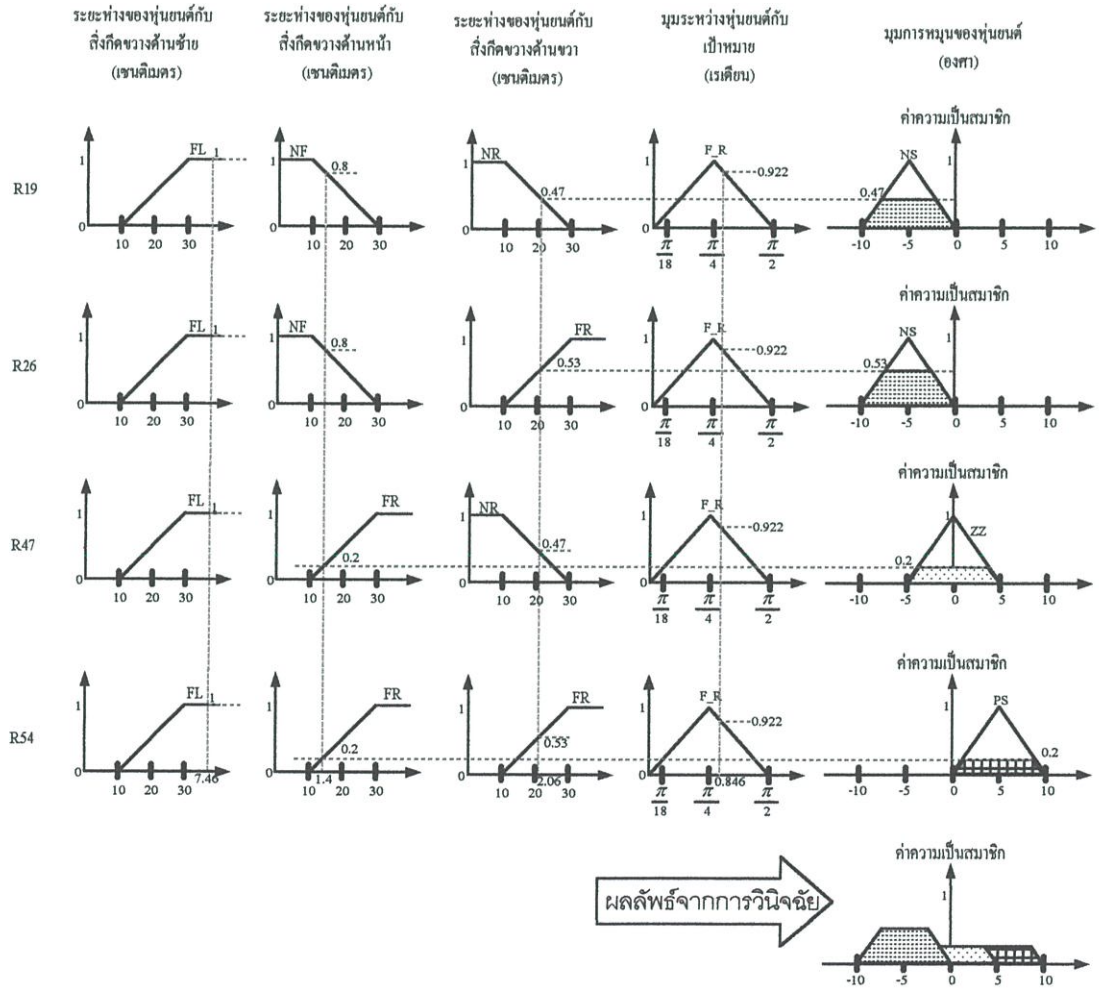
เป็นการนำค่าความเป็นสมาชิกที่ได้รับเข้ามาไปประมวลผลตามกฎที่ระบบได้เรียนรู้หรือออกแบบไว้เพื่อหาคำตอบ

การให้เหตุผลทางฟัซซี (Fuzzy Reasoning) จะเลือกใช้วิธีการของ Mamdani - Min Composition method [2] โดยจะนำค่าความเป็นสมาชิกที่น้อยที่สุดของอินพุตทุกตัวมาเป็นค่าความเป็นสมาชิกในเทอมเซตของเอาต์พุตที่ออกแบบไว้ในกฎการควบคุมข้อนั้น และถ้าค่าความเป็นสมาชิกเทอมเซตของเอาต์พุตมีหลายค่าจะโดยเลือกเอาเฉพาะค่าที่มากที่สุดเท่านั้น

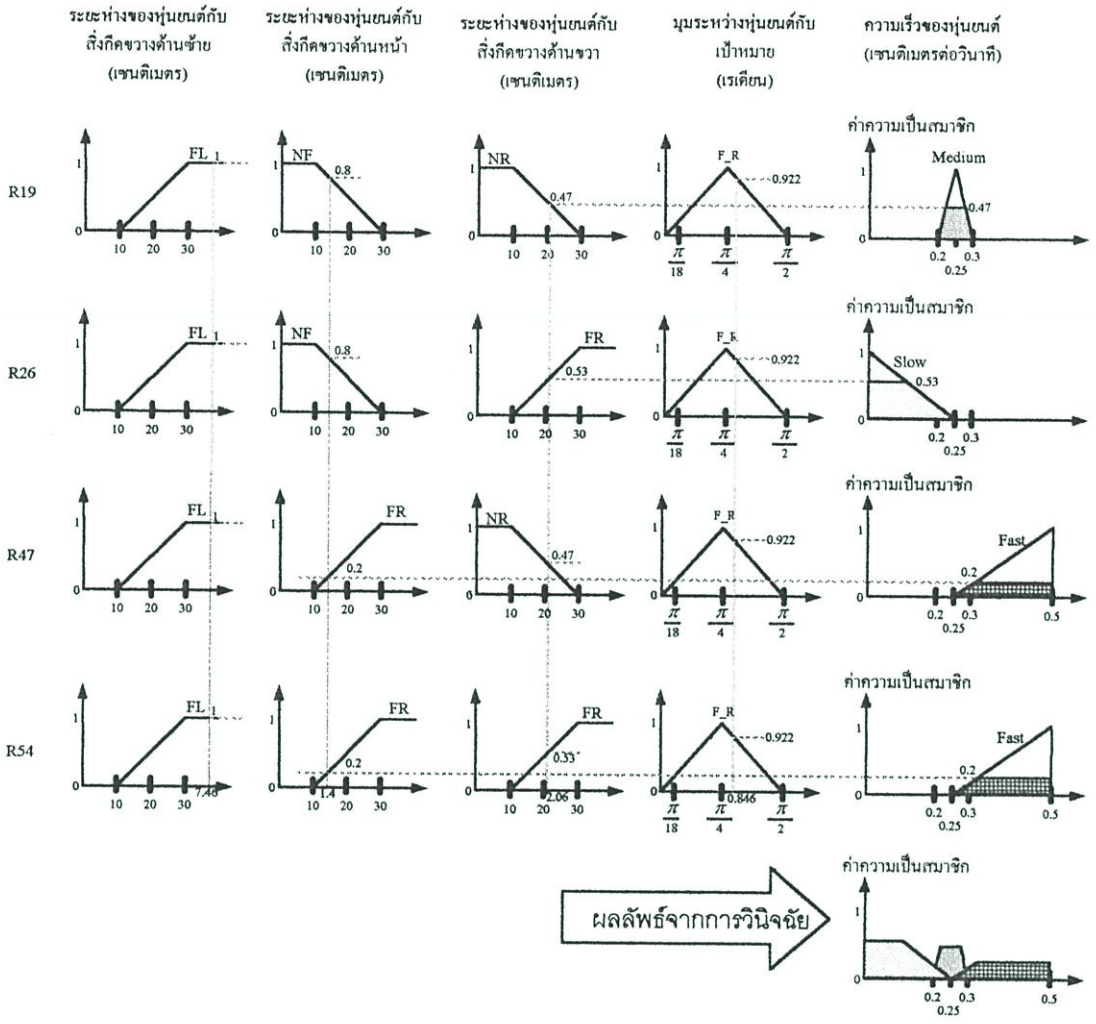
ยกตัวอย่างการหาวินิจฉัยกฎการควบคุมเมื่อ ตัวแปรอินพุตทั้ง 4 มีค่าดังนี้ ระยะห่างระหว่างหุ่นยนต์กับสิ่งกีดขวางทางด้านซ้าย 74.6 เซนติเมตร ระยะห่างระหว่างหุ่นยนต์กับสิ่งกีดขวางทางด้านหน้า 14 เซนติเมตร ระยะห่างระหว่างหุ่นยนต์กับสิ่งกีดขวางทางด้านขวา 20.6 เซนติเมตร ทิศทางของเป้าหมาย 0.846 เรเดียน ดังในรูปที่ 3.10



รูปที่ 3.10 ค่าของตัวแปรอินพุตทั้ง 4 สำหรับใช้วินิจฉัยกฎจากตัวอย่าง



รูปที่ 3.11 การวินิจฉัยกฎการควบคุมสำหรับตัวแปรเอาต์พุตมุมการหมุนของหุ่นยนต์



รูปที่ 3.12 การวินิจฉัยกฎการควบคุมสำหรับตัวแปรเอาต์พุตความเร็วของหุ่นยนต์

3.3 การดีฟัซซิฟิเคชัน

เป็นการแปลงผลจากค่าความเป็นสมาชิกที่ได้จากขั้นตอนของการวินิจฉัยกฎการควบคุม ให้อยู่ในรูปของค่าเอาต์พุต ซึ่งจะเป็นจำนวนจริงที่อยู่ในโดเมนของตัวแปรเอาต์พุต ซึ่งค่านี้เป็นค่าคาดหวัง (Expected Value) ของตัวแปรฟัซซี เพื่อนำไปใช้ในการควบคุมหุ่นยนต์นั่นเอง

เทคนิคการดีฟัซซิฟิเคชัน ก็มีหลายวิธีด้วยกัน แต่ที่ใช้กันมากคือ วิธีการหาค่าศูนย์กลาง (Center Of Area (COA)) [2] และวิธีการหาค่าเฉลี่ยของสมาชิกที่มากที่สุด (Mean Of Maximum (MOM)) สำหรับวิทยานิพนธ์นี้ใช้วิธีวิธีการหาค่าศูนย์กลางในการดีฟัซซิฟิเคชัน

- วิธีการหาค่าศูนย์กลาง

เป็นการประมาณค่าของตัวแปรเอาต์พุต โดยหาจุดศูนย์กลางของพื้นที่ วิธีนี้เป็นที่นิยมใช้มากในกระบวนการควบคุม เนื่องจากใช้ข้อมูลของค่าความเป็นสมาชิกของทุกกฎที่เป็นจริง ทำให้ค่าที่ได้มีความน่าเชื่อถือและเหมาะสมกว่าวิธีอื่น ๆ

สามารถคำนวณความเร็วของหุ่นยนต์ จากการดีฟัซซิฟิเคชันได้จากสมการที่ 3.8

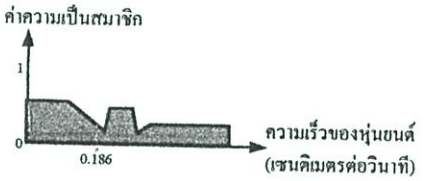
$$v^*_{Robot} = \frac{\sum_{j=1}^{N} \mu(v_j) \cdot v_j}{\sum_{j=1}^{N} \mu(v_j)} \tag{3.8}$$

โดยที่ v^*_{Robot} คือ ความเร็วของหุ่นยนต์จากการดีฟัซซิฟิเคชัน

v_j คือ ค่าเอาต์พุตของตัวแปรความเร็วของหุ่นยนต์ตัวที่ j

$\mu(v_j)$ คือ ค่าความเป็นสมาชิกของตัวแปร$_{v_j}$ ตัวที่ j

N คือ จำนวนข้อมูลที่แซมปลิง (Sampling) จากผลลัพธ์ของการวินิจฉัย ในที่นี้ใช้ 50 ตัว จากตัวอย่างที่แล้ว สามารถหาค่าความเร็วของหุ่นยนต์ได้ดังรูปที่ 3.13



รูปที่ 3.13 แสดงค่าความเร็วของหุ่นยนต์ที่ได้จากการดีฟัซซิฟิเคชัน

ส่วนการคำนวณมุมการหมุนของหุ่นยนต์ สามารถหาได้จากการดีฟัซซิฟิเคชัน ดังสมการที่ 3.9

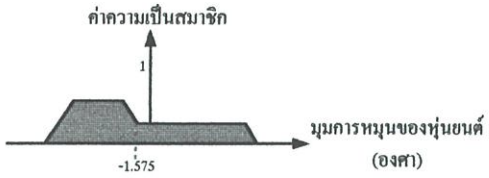
$$\theta^*_{Robot} = \frac{\sum_{j=1}^{N} \mu(\theta_j) \cdot \theta_j}{\sum_{j=1}^{N} \mu(\theta_j)} \tag{3.9}$$

โดยที่ θ^*_{Robot} คือ มุมการหมุนจากการดีฟัซซิฟิเคชัน

θ_j คือ ค่าเอาต์พุตของตัวแปรมุมการหมุนตัวที่ j

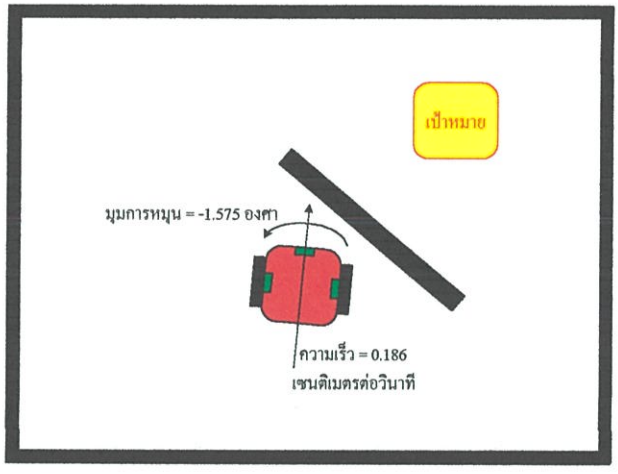
$\mu(\theta_j)$ คือ ค่าความเป็นสมาชิกของ θ_j ตัวที่ j

คือ จำนวนข้อมูลที่แซมปลิงจากผลลัพธ์ของการวินิจฉัย ในที่นี้ใช้ 300 ตัว จากตัวอย่างที่แล้ว สามารถหาค่ามุมการหมุนของหุ่นยนต์ ได้ดังรูปที่ 3.14



รูปที่ 3.14 ค่ามุมการหมุนของหุ่นยนต์ที่ได้จากการดีฟัซซิฟิเคชัน

การเคลื่อนที่ของหุ่นยนต์เมื่อได้รับค่าเอาต์พุต จากการดีฟัซซิฟิเคชันแล้ว แสดงดังรูปที่ 3.15



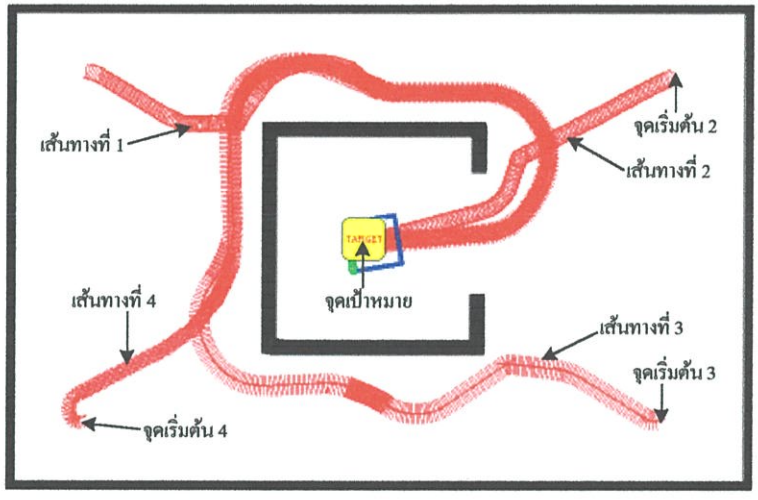
รูปที่ 3.15 ค่าเอาต์พุตจากการดีฟัซซิฟิเคชันในกรณีตัวอย่าง

เพื่อทดสอบให้เห็นถึงประสิทธิภาพของตัวควบคุมฟัซซีที่ได้ออกแบบมาจึงได้ทำการจำลองการทำงานการเคลื่อนที่ของหุ่นยนต์ในสภาวะแวดล้อมต่าง ๆ ที่สร้างขึ้นในหลาย ๆ รูปแบบ ดังจะได้กล่าวถึงในบทถัดไป

บทที่ 4
การจำลองการเคลื่อนที่ของหุ่นยนต์และผลการทดสอบ

เพื่อตรวจสอบประสิทธิภาพของตัวควบคุมฟัซซีที่ออกแบบไว้ในบทที่ 3 จากการเคลื่อนที่ของหุ่นยนต์ จึงได้จำลองสภาวะแวดล้อมของสิ่งกีดขวางให้มีสภาพที่แตกต่างกันไปดังต่อไปนี้

4.1 การทดสอบโดยให้เป้าหมายอยู่ในสิ่งกีดขวางรูปตัวยูโดยทำการเปลี่ยนตำแหน่งเริ่มต้นของหุ่นยนต์



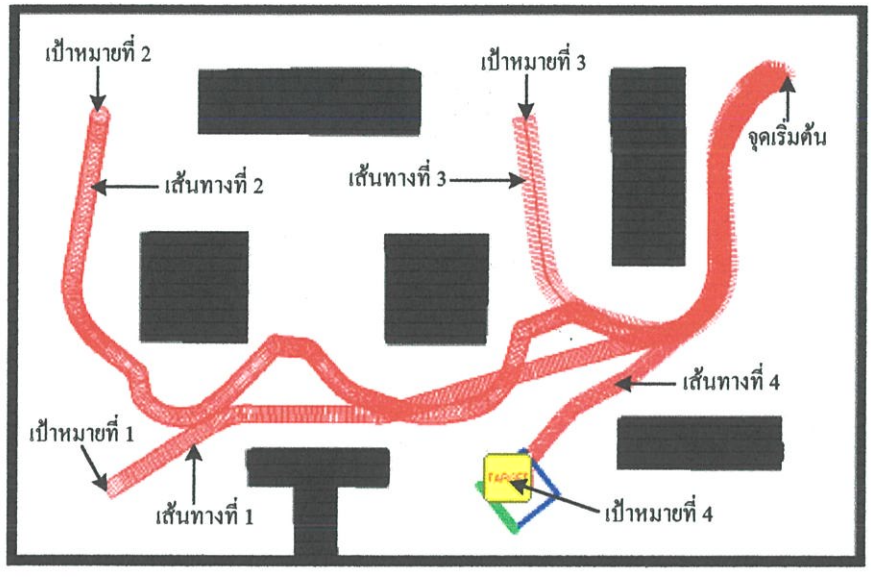
รูปที่ 4.1 เส้นทางการเคลื่อนที่ของหุ่นยนต์กำหนดให้เป้าหมายอยู่ในสิ่งกีดขวางรูปตัวยู

ตารางที่ 4.1 ระยะทางและเวลาของหุ่นยนต์จากรูปที่ 4.1

จุดเริ่มต้นที่	เส้นทางที่	ระยะทาง(ซม.)	เวลา(วินาที)
1	1 □	234.4	73.8
2	2 ○	109.3	27.6
3	3 +	417	128
4	4 ×	305.4	91.1

ในรูปที่ 4.1 และตารางที่ 4.1 แสดงให้เห็นว่า เส้นทางการเคลื่อนที่ในเส้นทางที่ 1 และ 2 นั้น หุ่นยนต์สามารถเคลื่อนที่ไปยังเป้าหมายโดยใช้ระยะทางและเวลาที่น่าพอใจ แต่สำหรับเส้นทางที่ 3 และ 4 จะพบข้อเสียตัวควบคุมฟัซซีที่ได้นำเสนอนี้ คือ เมื่อหุ่นยนต์ต้องหลบหลีกสิ่งกีดขวางโดยมีเป้าหมายอยู่ตรงหน้า จากการออกแบบกฎการควบคุมได้กำหนดให้หุ่นยนต์หลบไปทางซ้ายเสมอ ทำให้ต้องเคลื่อนที่โดยใช้ระยะทางและเวลาที่มากกว่าที่ควรจะเป็น

4.2 การทดสอบโดยให้หุ่นยนต์เริ่มต้นจากจุดเดียวไปยังเป้าหมายที่ต่างกันในสิ่งกีดขวางที่วางกระจัดกระจายกันอยู่



รูปที่ 4.2 เส้นทางการเคลื่อนที่ของหุ่นยนต์ในพื้นที่ที่สิ่งกีดขวางวางกระจัดกระจายกันอยู่

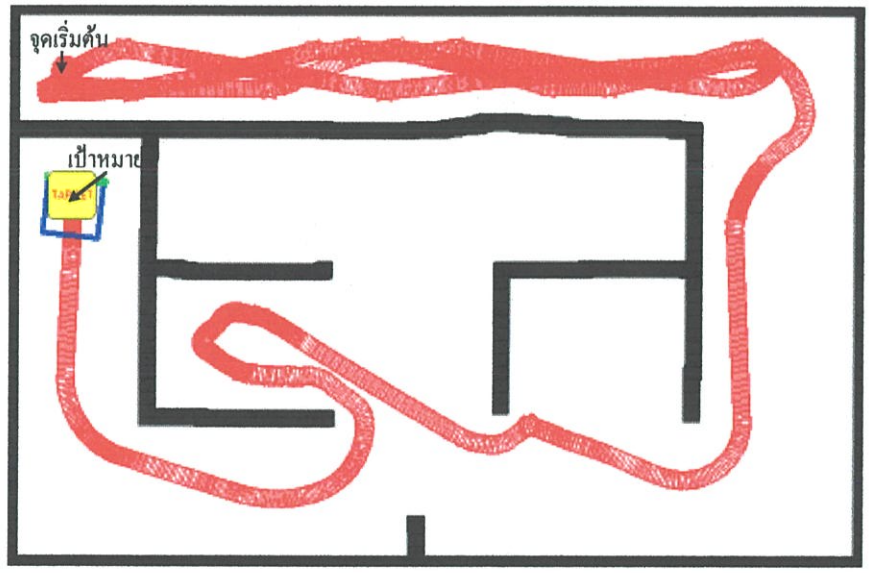
ตารางที่ 4.2 ระยะทางและเวลาของหุ่นยนต์จากรูปที่ 4.2

เป้าหมายที่	เส้นทางที่	ระยะทาง(ซม.)	เวลา(วินาที)
1	1 □	234.5	65.7
2	2 ○	336.8	106.8
3	3 +	167.6	52.6
4	4 ×	143.9	41.6

ในเส้นทางที่ 1, 3 และ 4 หุ่นยนต์สามารถเคลื่อนที่ไปยังเป้าหมายโดยใช้ระยะทางและเวลาที่น่าพอใจแต่สำหรับเส้นทางที่ 2 จะพบข้อเสียเช่นเดียวกับกรณีที่ได้กล่าวจากหัวข้อที่ผ่านมา ดังรูปที่ 4.2 และ ตารางที่ 4.2

4.3 การทดสอบให้หุ่นยนต์เคลื่อนที่ในพื้นที่ที่สิ่งกีดขวางที่วางอย่างสลับซับซ้อนคล้ายเขาวงกต

เพื่อทดสอบประสิทธิภาพในการทำงานของหุ่นยนต์ จึงจำลองสิ่งแวดล้อมที่มีสิ่งกีดขวางวางอย่างสลับซับซ้อนซึ่งมีลักษณะคล้ายกับเขาวงกต จากเส้นทางการเคลื่อนที่ของหุ่นยนต์ในรูปที่ 4.3 พบว่าหุ่นยนต์ตัดสินใจผิดพลาดทำให้เคลื่อนที่วนกลับไปยังจุดเริ่มต้นอยู่หนึ่งรอบ แต่รอบที่สองก็สามารถตัดสินใจได้อย่างถูกต้อง เนื่องจากข้อมูลที่หุ่นยนต์ได้รับเปลี่ยนไปทำให้การตัดสินใจเปลี่ยนไปด้วย

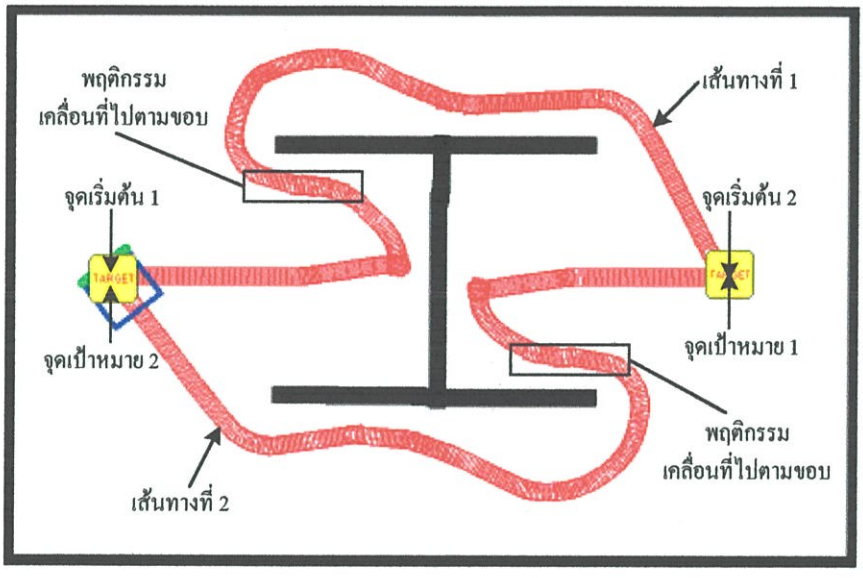


รูปที่ 4.3 เส้นทางการเคลื่อนที่ของหุ่นยนต์ในสิ่งแวดล้อมที่มีสิ่งกีดขวางวางอย่างสลับซับซ้อน

จากรูปจะเห็นว่าเส้นทางที่ใช้ไม่ใช่เส้นทางที่ดีที่สุดเนื่องจากหุ่นยนต์ต้องเสียเวลาเมื่อเคลื่อนที่เข้าไปในสิ่งกีดขวางรูปตัวยู ทั้งนี้ก็เพราะว่าหุ่นยนต์ไม่สามารถที่จะรับรู้ว่าเส้นทางข้างหน้าจะมีสิ่งกีดขวางเป็นอย่างไรจนกว่าที่จะเข้าไปใกล้นั่นเอง

4.4 การทดสอบให้หุ่นยนต์เคลื่อนที่ไปยังเป้าหมายที่อยู่หลังสิ่งกีดขวางรูปตัวยู

เป็นการจำลองเพื่อทดสอบว่าเมื่อหุ่นยนต์เคลื่อนที่เข้าไปในสิ่งกีดขวางรูปตัวยูแล้วจะสามารถเคลื่อนที่ออกมาโดยที่ไม่เคลื่อนที่วนรอบอยู่ภายในสิ่งกีดขวางรูปตัวยูได้หรือไม่ โดยการทดสอบจะให้หุ่นยนต์เคลื่อนที่ 2 เที่ยว คือ ทั้งไปและกลับ โดยให้มีสิ่งกีดขวางรูปตัวยู 2 ตัวหันหลังชนกัน ดังในรูปที่ 4.4

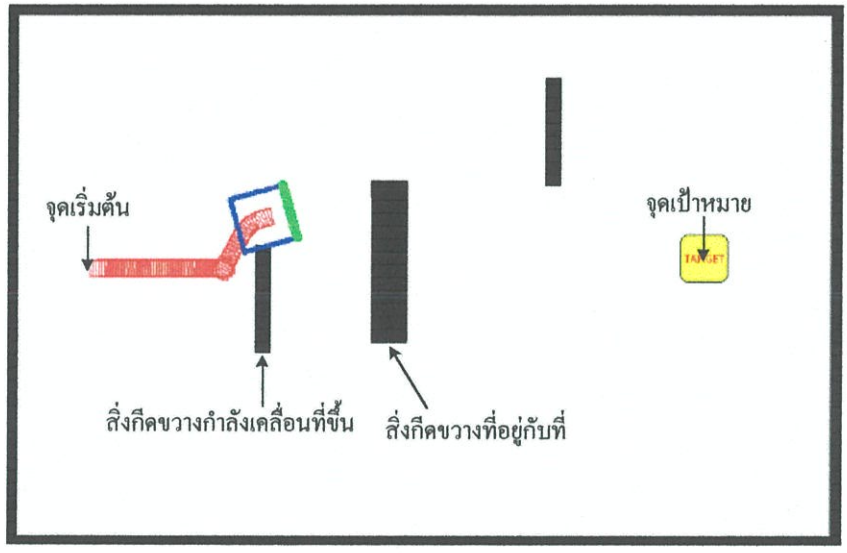


รูปที่ 4.4 เส้นทางการเคลื่อนที่ของหุ่นยนต์เมื่อเคลื่อนที่ไปในสิ่งกีดขวางรูปตัวยู

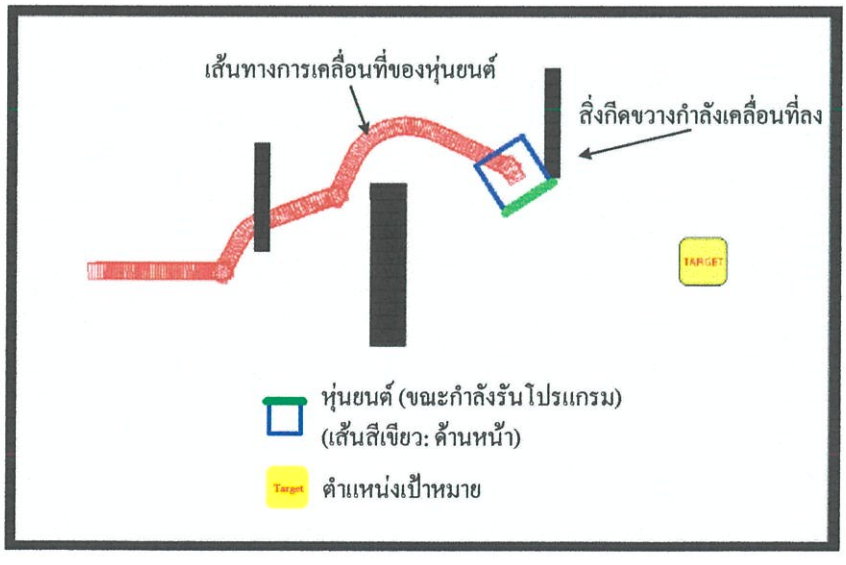
จากรูปที่ 4.4 เมื่อหุ่นยนต์เคลื่อนที่เข้าไปในสิ่งกีดขวางรูปตัวยู หุ่นยนต์จะใช้พฤติกรรมหลบหลีกสิ่งกีดขวางจนกระทั่งหันหลังให้กับสิ่งกีดขวางแล้วหลังจากนั้นจะใช้พฤติกรรมเคลื่อนที่ไปตามขอบเพื่อที่จะเคลื่อนที่ออกมาจากสิ่งกีดขวางรูปตัวยู หลังจากที่ออกมาภายนอกสิ่งกีดขวางรูปตัวยูแล้วก็จะใช้พฤติกรรมเคลื่อนไปยังตำแหน่งเป้าหมายต่อไป

4.5 การทดสอบให้หุ่นยนต์เคลื่อนที่หลบหลีกสิ่งกีดขวางที่เคลื่อนที่ได้

เป็นการจำลองให้มีสิ่งกีดขวางที่เคลื่อนที่ได้ 2 ตัว เคลื่อนที่ขึ้นลงอยู่ระหว่างจุดเริ่มต้นของหุ่นยนต์กับเป้าหมาย โดยที่สิ่งกีดขวางทางซ้ายกำลังเคลื่อนที่ขึ้น ส่วนสิ่งกีดขวางทางขวากำลังเคลื่อนที่ลง ทั้งสองเคลื่อนที่ด้วยความเร็ว 2.5 เซนติเมตรต่อวินาทีเท่ากัน เส้นทางการเคลื่อนที่ของหุ่นยนต์ แสดงดังรูปที่ 4.5 และ 4.6



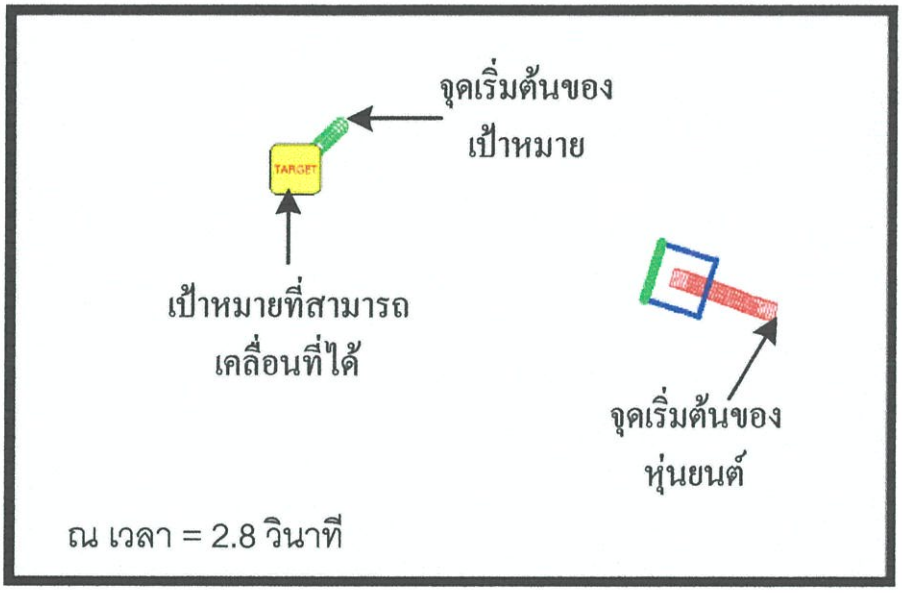
รูปที่ 4.5 หุ่นยนต์กำลังเคลื่อนที่หลบหลีกสิ่งกีดขวางที่กำลังเคลื่อนที่ขึ้นไปด้านบน



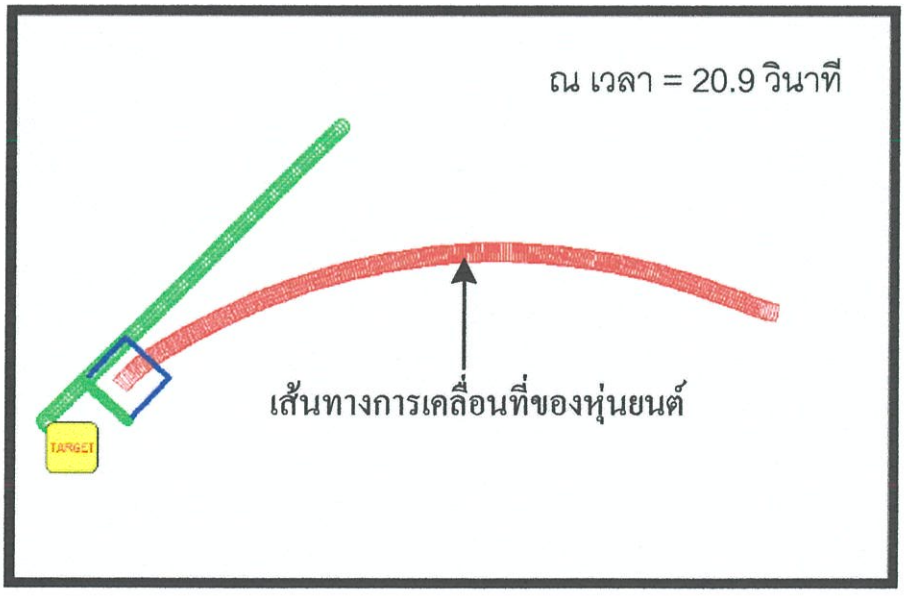
รูปที่ 4.6 หุ่นยนต์กำลังเคลื่อนที่หลบหลีกสิ่งกีดขวางที่กำลังเคลื่อนที่ลงมาด้านล่าง

4.6 การทดสอบให้หุ่นยนต์เคลื่อนที่ไล่จับเป้าหมายที่สามารถเคลื่อนที่ได้

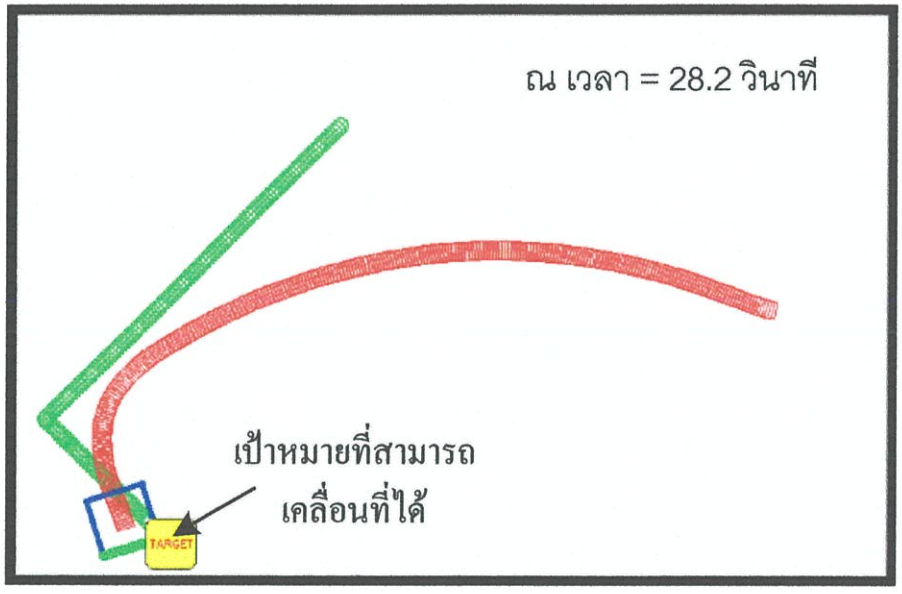
เป็นการจำลองให้หุ่นยนต์เคลื่อนที่ไปยังเป้าหมายที่สามารถเคลื่อนที่ได้ โดยเป้าหมายเคลื่อนที่ด้วยความเร็ว 2 เซนติเมตรต่อวินาที เส้นทางการเคลื่อนที่ของหุ่นยนต์ แสดงดังรูปที่ 4.7 ถึง 4.10



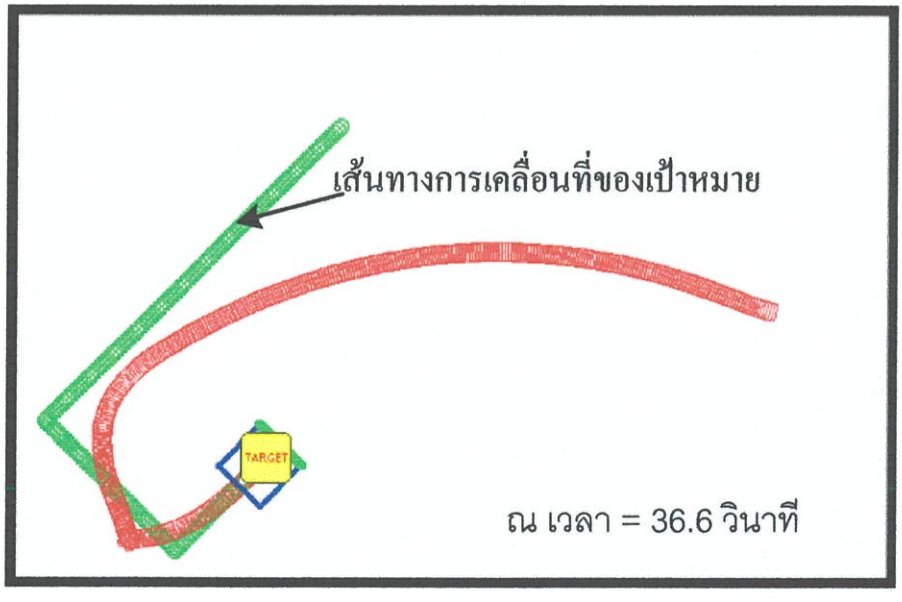
รูปที่ 4.7 หุ่นยนต์เริ่มเคลื่อนที่เข้าหาเป้าหมายที่เคลื่อนที่ได้



รูปที่ 4.8 เป้าหมายเปลี่ยนทิศทางการเคลื่อนที่ครั้งแรก



รูปที่ 4.9 เป้าหมายเปลี่ยนทิศทางการเคลื่อนที่ครั้งที่ 2

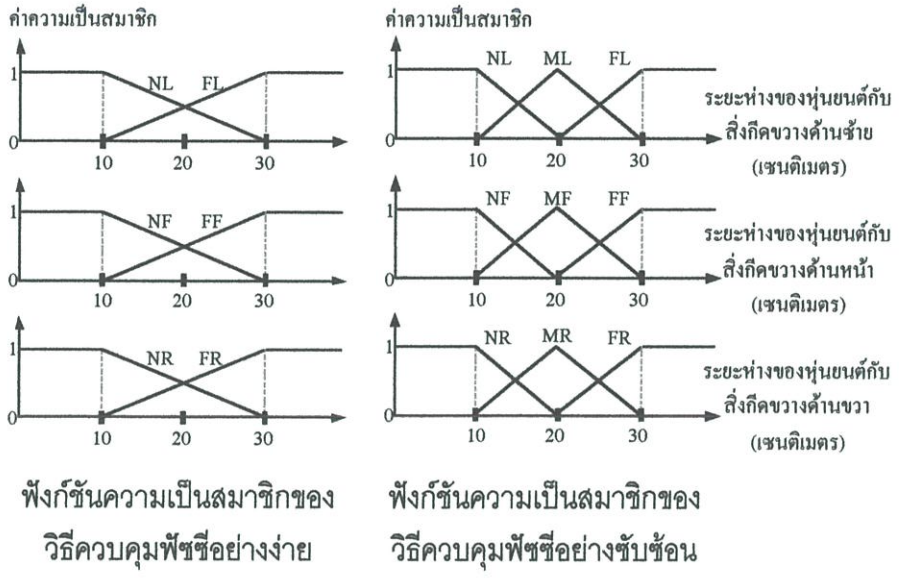


รูปที่ 4.10 เมื่อหุ่นยนต์สามารถเคลื่อนที่ไปถึงเป้าหมายได้สำเร็จ

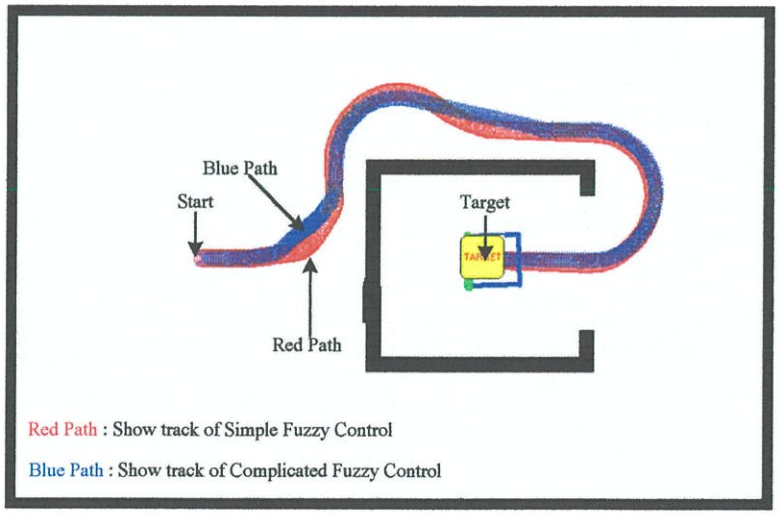
การจำลองในหัวข้อ 4.5 และ 4.6 ทำให้เห็นว่า ระบบควบคุมฟัซซีสามารถนำมาใช้กับหุ่นยนต์ให้เคลื่อนที่ในสภาพแวดล้อมที่มีการเปลี่ยนแปลงได้ เนื่องจากการทำงานของหุ่นยนต์จะรับข้อมูลสภาพแวดล้อม ณ เวลานั้น จึงทำให้การตัดสินใจเป็นไปในลักษณะเวลาจริง (Realtime)

4.7 การทดสอบเปรียบเทียบระหว่างวิธีควบคุมฟัซซีอย่างง่ายกับอย่างซับซ้อน

เป็นการจำลองเพื่อเปรียบเทียบวิธีควบคุมฟัซซีอย่างง่ายที่ออกแบบไว้กับวิธีควบคุมฟัซซีที่ซับซ้อน ซึ่งมีฟังก์ชันความเป็นสมาชิกดังในรูปที่ 4.10



รูปที่ 4.11 ฟังก์ชันความเป็นสมาชิกของวิธีควบคุมฟัซซีอย่างง่ายและอย่างซับซ้อน



รูปที่ 4.12 เส้นทางการเคลื่อนที่ของหุ่นยนต์ของวิธีควบคุมฟัซซีอย่างง่ายและอย่างซับซ้อน

ตารางที่ 4.3 ระยะทางและเวลาของการเคลื่อนที่ของหุ่นยนต์ด้วยวิธีควบคุมฟัซซีอย่างง่าย และอย่างซับซ้อนจากรูปที่ 4.12

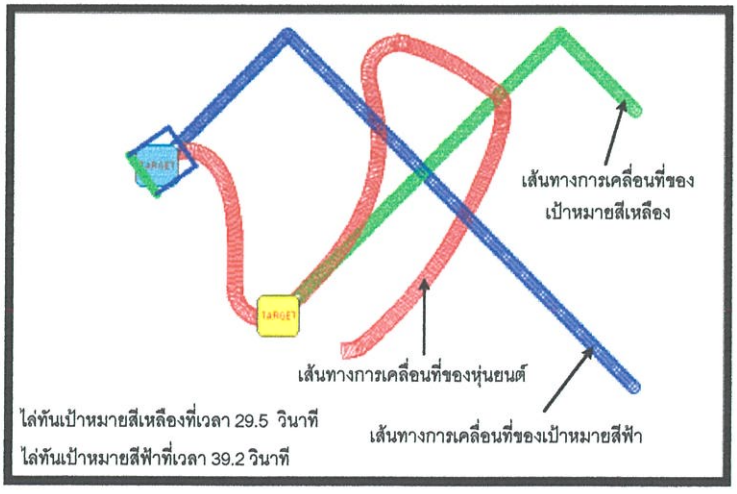
Path	Distance(Cm.)	Time(S.)
Simple Fuzzy Control (Red Path)	243	66.2
Complicated Fuzzy Control (Blue Path)	238.1	67

จากรูปที่ 4.12 และตารางที่ 4.3 จะเห็นได้ว่าวิธีควบคุมฟัซซีอย่างง่ายจะใช้เวลาในการเคลื่อนที่ไปยังเป้าหมายน้อยกว่า เนื่องจากมีการเปลี่ยนแปลงมุมการหมุนของหุ่นยนต์น้อยกว่า ทำให้ไม่เสียเวลากับการหมุนตัวของหุ่นยนต์ แต่จะใช้ระยะทางในการเคลื่อนที่มากกว่า ก็เพราะว่าวิธีควบคุมฟัซซีอย่างง่ายจะต้องเคลื่อนที่เข้าไปใกล้กับสิ่งกีดขวางมากกว่าวิธีควบคุมฟัซซีอย่างซับซ้อนจึงจะเริ่มหลบหลีกสิ่งกีดขวาง หรือกล่าวอีกอย่างได้ว่าวิธีควบคุมฟัซซีอย่างซับซ้อนสามารถหลบหลีกสิ่งกีดขวางได้ก่อนที่จะเข้าใกล้สิ่งกีดขวางนั่นเอง

4.8 การทดสอบให้หุ่นยนต์เคลื่อนที่ไล่จับเป้าหมายที่เคลื่อนที่ได้ 2 เป้าหมาย

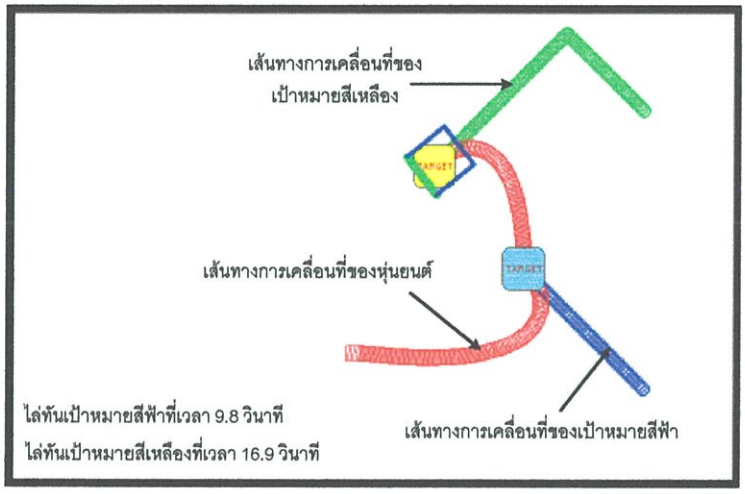
เป็นการจำลองให้หุ่นยนต์เคลื่อนที่ไปยังเป้าหมายที่มีมากกว่าหนึ่งเป้าหมาย โดยในที่นี้แบ่งการจำลองเป็น 2 ลักษณะ คือ 1. จับเป้าหมายตามลำดับ ดังในรูปที่ 4.13 และ 2. จับเป้าหมายที่อยู่ใกล้กว่าก่อน ดังในรูปที่ 4.14 โดยเส้นทางการเคลื่อนที่ของหุ่นยนต์เป็นสีแดง เส้นทางการเคลื่อนที่สีเขียวเป็นของเป้าหมายสีเหลือง และเส้นทางการเคลื่อนที่สีน้ำเงินเป็นของเป้าหมายสีฟ้า

4.8.1 จับเป้าหมายตามลำดับ (สีเหลืองก่อนสีฟ้า)



รูปที่ 4.13 เส้นทางการเคลื่อนที่ของหุ่นยนต์ในการไล่จับเป้าหมายทั้งสองตามลำดับ

4.8.2 จับเป้าหมายที่อยู่ใกล้กว่าก่อน



รูปที่ 4.14 เส้นทางการเคลื่อนที่ของหุ่นยนต์ในการไล่จับเป้าหมายที่ใกล้กว่าก่อน

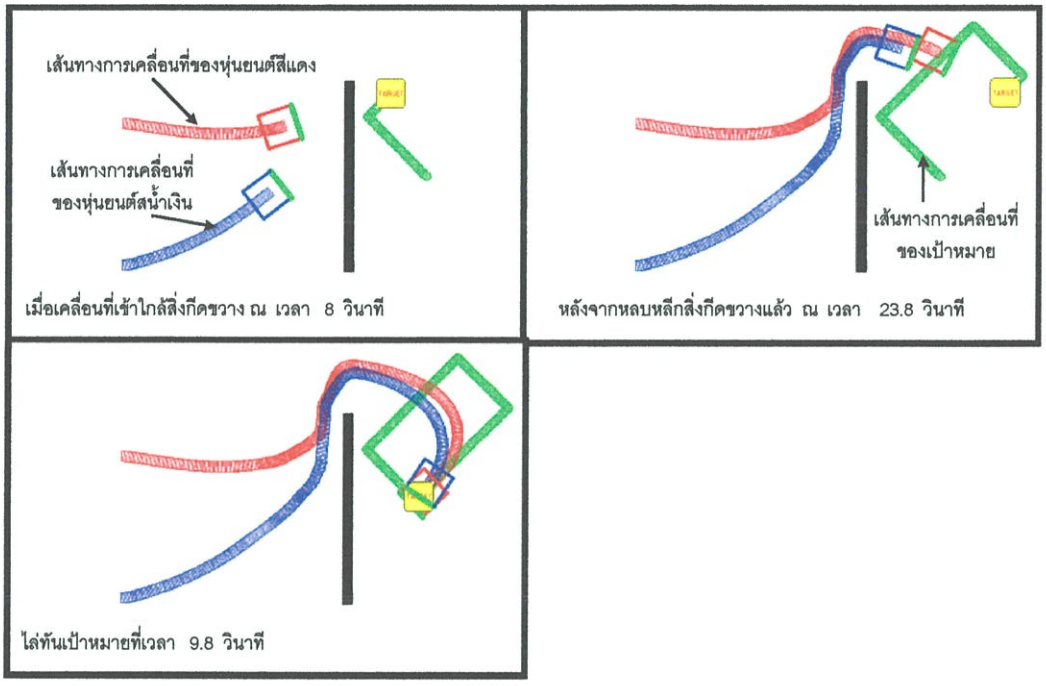
จากรูปที่ 4.13 และ 4.14 เมื่อกำหนดให้ทั้ง 2 กรณีมีจุดเริ่มต้นของหุ่นยนต์และเป้าหมายทั้งสองเหมือนกัน จะเห็นว่าในตอนเริ่มต้นหุ่นยนต์จะใกล้กับเป้าหมายสีฟ้ามากกว่า ทำให้กรณีที่ไล่จับเป้าหมายตามลำดับ หุ่นยนต์จะใช้เวลามากกว่า เนื่องจากต้องไล่จับสีเหลืองให้ได้ก่อนแล้วค่อยจับสีฟ้า แต่กรณีที่ไล่จับเป้าหมายที่ใกล้กว่าก่อนจะสามารถจับสีฟ้าที่อยู่ใกล้กว่าสีเหลืองได้เลยแล้วค่อยไล่จับสีเหลืองต่อไป ทำให้หุ่นยนต์จะใช้เวลาน้อยกว่านั่นเอง

4.9 การทดสอบให้หุ่นยนต์ 2 ตัวเคลื่อนที่ไล่จับเป้าหมายเคลื่อนที่ได้เป้าหมายเดียวกัน

เป็นการจำลองเพื่อทดสอบว่าสามารถนำวิธีการควบคุมฟัซซีไปใช้กับหุ่นยนต์มากกว่าหนึ่งตัวในเวลาเดียวกันได้หรือไม่ นั่นคือจะให้หุ่นยนต์ทั้งสองใช้วิธีการควบคุมฟัซซีเหมือนกันไปยังเป้าหมายที่เคลื่อนที่ได้เป้าหมายเดียวกัน โดยจะแบ่งเป็น 2 กรณี คือ 1. หุ่นยนต์ทั้งสองไม่ได้หลบกันเอง 2. หุ่นยนต์ทั้งสองหลบกันเองด้วย ดังแสดงในรูปที่ 4.15 และ 4.16 ตามลำดับ โดยเส้นทางการเคลื่อนที่สีแดงเป็นของหุ่นยนต์สีแดง เส้นทางการเคลื่อนที่สีน้ำเงินเป็นของหุ่นยนต์สีน้ำเงินและเส้นทางการเคลื่อนที่สีเขียวเป็นของเป้าหมาย

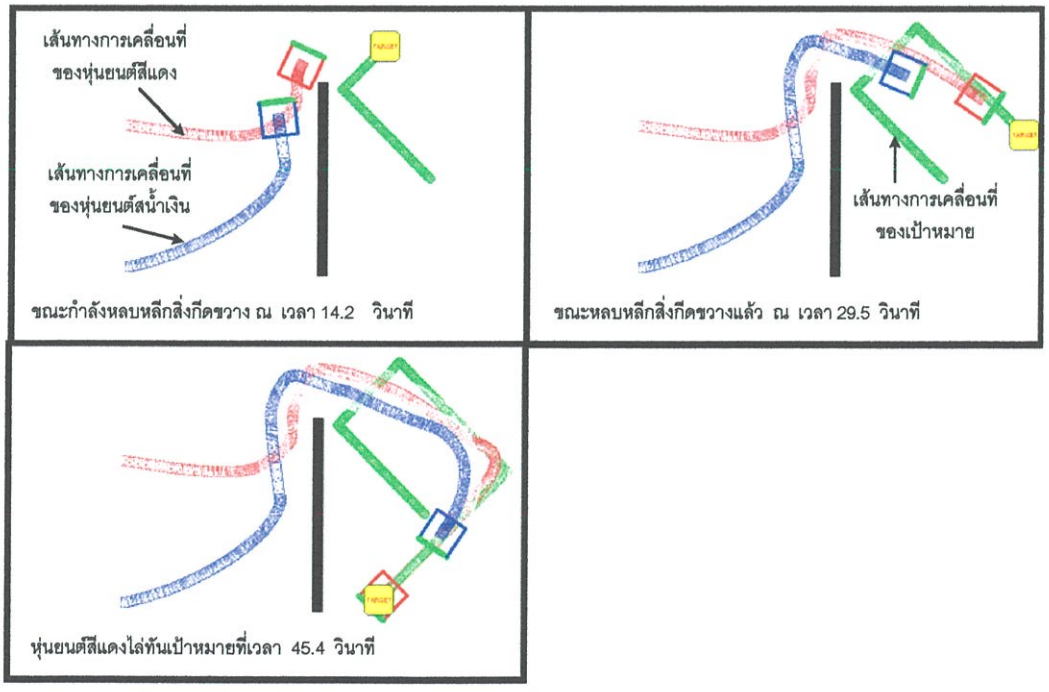
จากการทดลองนี้จะเห็นได้ว่าวิธีการควบคุมฟัซซีสามารถนำไปใช้กับหุ่นยนต์ได้มากกว่าหนึ่งตัว ดังนั้นจึงกล่าวได้ว่าสามารถนำวิธีการควบคุมฟัซซีไปใช้กับหุ่นยนต์ที่ต้องการทำงานร่วมกันเป็นทีมได้

4.9.1 หุ่นยนต์ทั้งสองไม่ได้หลบกันเอง



รูปที่ 4.15 เส้นทางการเคลื่อนที่ของหุ่นยนต์ทั้งสองที่ไม่ได้หลบกันเองไล่จับเป้าหมาย

4.9.2 หุ่นยนต์ทั้งสองหลบกันเองด้วย



รูปที่ 4.16 เส้นทางการเคลื่อนที่ของหุ่นยนต์ทั้งสองไล่จับเป้าหมายโดยที่วิ่งหลบกันเองด้วย

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

วิทยานิพนธ์ฉบับนี้กล่าวถึงวิธีการแก้ปัญหาการติดตามเป้าหมายของหุ่นยนต์ในขณะที่มีสิ่งกีดขวางด้วยกระบวนการควบคุมแบบฟัซซีอย่างง่ายซึ่งมีรูปแบบของฟังก์ชันความเป็นสมาชิกออกแบบให้อยู่ในรูปแบบที่ง่ายต่อการนำไปใช้จริงและการกำหนดค่าพารามิเตอร์ที่ใช้ในการควบคุมออกแบบอาศัยขนาดของหุ่นยนต์และระยะอันตรกิริยาระหว่างตัวตรวจรับกับสิ่งแวดล้อม

ในบทแรกกล่าวถึงความเป็นมาและความสำคัญของระบบควบคุมฟัซซี ความมุ่งหมายและวัตถุประสงค์ของการวิจัย ขอบเขตของการวิจัย ขั้นตอนของการศึกษา

บทถัดมากล่าวถึงลักษณะกายภาพของหุ่นยนต์ เช่น การติดตั้งอัลตราโซนิคเซนเซอร์และลิมิตสวิชท์ การทำงานของลิมิตสวิชท์และการทำงานของอัลตราโซนิคเซนเซอร์ ขั้นตอนการทำงานของโปรแกรมจำลองการเคลื่อนที่ของหุ่นยนต์ หน้าต่างและขั้นตอนในการใช้งานของโปรแกรมการจำลอง ในหัวข้อสุดท้ายกล่าวถึงแบบจำลองคณิตศาสตร์และสมการการเคลื่อนที่ของหุ่นยนต์ จะนำค่าความเร็วและมุมการหมุนของหุ่นยนต์ที่ได้จากการดีฟัซซิฟิเคชันมากำหนดตำแหน่งที่หุ่นยนต์จะเคลื่อนที่ไปและมุมการวางตัวของหุ่นยนต์ที่ตำแหน่งใหม่

ในส่วนของการออกแบบตัวควบคุมฟัซซี กล่าวถึง โครงสร้างพื้นฐานที่สำคัญของระบบควบคุมฟัซซี 4 ส่วน คือ ส่วนของการฟัซซิฟิเคชันซึ่งเป็นขั้นตอนการแปลงค่าตัวแปรอินพุตและค่าตัวแปรเอาต์พุตให้อยู่ในรูปค่าความเป็นสมาชิกในระบบฟัซซีโดยใช้ฟังก์ชันความเป็นสมาชิกรูปสามเหลี่ยมและตัวแปรทางภาษา โดยในวิทยานิพนธ์นี้ใช้ตัวแปรอินพุต 4 ตัวแปร คือ ระยะห่างระหว่างหุ่นยนต์กับสิ่งกีดขวางทางด้านซ้าย ทางด้านหน้า ทางด้านขวา และทิศทางของเป้าหมาย มีตัวแปรเอาต์พุต 2 ตัวแปร คือ ความเร็วและมุมการหมุนของหุ่นยนต์ ในส่วนของกฎการควบคุมฟัซซีเป็นการแสดงความสัมพันธ์ระหว่างตัวแปรฟัซซีอินพุตและตัวแปรฟัซซีเอาต์พุตในรูปแบบของ IF-THEN ในการสร้างกฎการควบคุมอาศัยพฤติกรรมของหุ่นยนต์ที่จะกระทำกับสิ่งกีดขวางในสภาวะแวดล้อมขณะนั้นโดยแบ่งออกได้เป็น 4 พฤติกรรม คือ เคลื่อนที่เพื่อหลบหลีกสิ่งกีดขวาง เคลื่อนที่ในทางแคบ ๆ เคลื่อนที่ไปตามขอบ เคลื่อนที่ไปยังเป้าหมาย ส่วนของหน่วยวินิจฉัยกฎเป็นการนำค่าความเป็นสมาชิกที่ได้รับเข้ามาไปประมวลผลตามกฎที่ระบบได้เรียนรู้หรือออกแบบไว้เพื่อหาคำตอบโดยใช้วิธีการของ Mamdani - Min ส่วนของการดีฟัซซิฟิเคชันเป็นการแปลงผลที่ได้จากขั้นตอนของการวินิจฉัยกฎการควบคุมให้อยู่ในรูปของค่าเอาต์พุต ซึ่งจะเป็นจำนวนจริงที่อยู่ในโดเมนของตัวแปรเอาต์พุตเพื่อนำไปใช้ในการควบคุมหุ่นยนต์นั่นเอง

การทดสอบการเคลื่อนที่ของหุ่นยนต์เพื่อตรวจสอบประสิทธิภาพของตัวควบคุมฟัซซีที่ออกแบบไว้ โดยจำลองสภาวะแวดล้อมของสิ่งกีดขวางให้มีสภาพที่แตกต่างกันไป เช่น ให้เป้าหมายอยู่ในสิ่งกีดขวางรูปตัวยูโดยทำการตำแหน่งเริ่มต้นของหุ่นยนต์ ให้หุ่นยนต์เริ่มต้นจากจุดเดียวไปยังเป้าหมายที่ต่างกันในสิ่งกีดขวางที่วางกระจัดกระจายกันอยู่ ซึ่งการทดสอบทั้งสองนี้ทำให้เห็นว่าการเคลื่อนที่ของหุ่นยนต์บางครั้งจะใช้ระยะทางและเวลาที่มากกว่าที่ควรจะเป็น และเพื่อทดสอบให้สภาวะแวดล้อมซับซ้อนขึ้นยังได้จำลองให้สิ่งกีดขวางวางอย่างสลับซับซ้อนคล้ายเขาวงกตอีกด้วย และเพื่อทดสอบการเคลื่อนที่ของหุ่นยนต์ในสภาวะแวดล้อมที่มีการเปลี่ยนแปลงจึงจำลองให้สิ่งกีดขวางและเป้าหมายสามารถเคลื่อนที่ได้ นอกจากนี้แล้วยังได้ทำการทดสอบโดยให้หุ่นยนต์เคลื่อนที่ไปยังเป้าหมายที่มีมากกว่าหนึ่งเป้าหมายและให้หุ่นยนต์มากกว่าหนึ่งตัวที่ใช้วิธีการควบคุมฟัซซีเหมือนกันวิ่งไปหาเป้าหมายเดียวกันอีกด้วย

จากผลการจำลองจะเห็นได้ว่าหุ่นยนต์สามารถเคลื่อนที่หลบหลีกสิ่งกีดขวางและเคลื่อนที่ไปถึงเป้าหมายที่กำหนดไว้ได้ในกรณีศึกษาต่าง ๆ ไม่ว่าสิ่งกีดขวางจะอยู่กับที่หรือสิ่งกีดขวางเคลื่อนที่ได้ แม้กระทั่งเมื่อเป้าหมายเคลื่อนที่ได้ ดังนั้นจึงสามารถกล่าวได้ว่าระบบควบคุมฟัซซีอย่างง่ายที่นำเสนอในวิทยานิพนธ์นี้ที่นำมาใช้เพื่อช่วยในการตัดสินใจของหุ่นยนต์ น่าจะสามารถนำไปประยุกต์ใช้งานกับหุ่นยนต์ที่สร้างขึ้นมาตามแบบจำลองที่ได้นำเสนอข้างต้นได้

5.2 ข้อเสนอแนะ

แนวทางในการนำวิทยานิพนธ์นี้ไปใช้สำหรับงานวิจัยอื่น ๆ ในอนาคตนั้น โดยอาจจะทำการพัฒนาโปรแกรมการจำลองการทำงานให้หุ่นยนต์เคลื่อนที่ในสภาพแวดล้อมที่เป็นแบบ 3 มิติ เพื่อให้มีความใกล้เคียงกับความเป็นจริงมากยิ่งขึ้น หรืออาจจะนำระบบควบคุมฟัซซีในวิทยานิพนธ์นี้ไปประยุกต์ใช้งานจริงกับหุ่นยนต์ที่สร้างขึ้นมาตามแบบจำลองเพื่อพิสูจน์ถึงประสิทธิภาพอย่างเป็นรูปธรรมของระบบควบคุมที่ได้ออกแบบไว้ ณ ที่นี้

เอกสารอ้างอิง

[1] Chin-Teng L., and George Lee C.S. **Neural Fuzzy Systems.** International Edition. New Jersey : Prentice-Hall, Inc. 1996.

[2] Tsoukalas L., and Uhrig R. **Fuzzy And Neural Approaches In Engineering.** New York : John Wiley & Sons, Inc. 1996.

[3] Martinez A., Tunstel E., Jamshidi M. "Fuzzy Logic Based Collision Avoidance for a Mobile Robot" **International Conf. on Industrial Fuzzy Control and Intelligent Systems.**, Dec 1993. pp.66-69.

[4] Wei Li. "Perception-Action Behavior Control of a Mobile Robot in Uncertain Environments Using Fuzzy Logic" **Proc. IEEE/RSI/GI Intelligent Robots and Systems.**, vol. 1, 1994. pp.439-446.

[5] Wei Li. and Xun Feng. "Behavior Fusion for Robot Navigation in Uncertain Environments Using Fuzzy Logic" **IEEE International Conf. on Systems Man and Cybermetics.**, vol. 2, Oct 1994. pp.1790-1796.

[6] Tunstel E. and Jamshidi M. "Fuzzy logic and Behavior Control Strategy for Autonomous Mobile Robot Mapping" **Proc. IEEE Conf. on Fuzzy Systems.**, vol. 1, June 1994. pp.514-517.

[7] Ho Yim. and Butler A.C. "Motion Planning Using Fuzzy Logic Control With Minimum Sensors" **Proc. IEEE International Symp. on Intelligent Control.**, Aug 1995. pp.558-564.

[8] Ramirez-Serrano A. and Boumedine M. "Real-Time Navigation in Unknown Environments Using Fuzzy Logic and Ultrasonic Sensing" **IEEE International Symp. on Intelligent Control.**, Sept 1996. pp.26-30.

[9] Da Silva I.N., Gomide F.A.C., Do Amaral W.C. "Navigation of Mobile Robots Using Fuzzy Logic Controllers" **International Workshop on Advanced Motion Control.**, June-July 1998. pp.346-349.

[10] Lee T.H., Lam H.K., Leung F.H.F., Tam P.K.S. "A Practical Fuzzy Logic Controller for the Path Tracking of Wheeled Mobile Robots" **IEEE Control Systems Magazine.**, vol. 5, April 26 – May 1, 2003. pp.5045-5050.

[11] Li T.H.S., Shih-Jie Chang. "Autonomous Fuzzy Parking Control of a Car-Like Mobile Robot" **IEEE Transactions on Systems, Man and Cybernetics., Part A**,vol.33, no.4, July 2003. pp.451-465.

[12] Cang Ye., Yung N.H.C., Danwei Wang. "A Fuzzy Controller with Supervised Learning Assisted Reinforcement Learning Algorithm for Obstacle Avoidance" **IEEE Transactions on Systems, Man and Cybernetics., Part B**, vol. 33, no.1, Feb 2003. pp.17-27.

[13] Anmin Zhu., Yang S.X. "A Fuzzy Logic Approach to Reactive Navigation of Behavior–Based Mobile Robots" **Proc. IEEE International Conf. on Robotics and Automation.**, vol. 23, no.2, April 2003. pp.60-65.

ภาคผนวก

โปรแกรมที่ใช้ในการทดสอบการจำลองในบทที่ 4

```
'Declare Variables in Fuzzification
Dim Distance_FrontObs, Distance_LeftObs, Distance_RightObs
Dim NF, FF                              'near far Front Term Set
Dim NL, FL                              'near far Left Term Set
Dim NR, FR                              'near far Right Term Set
Dim B_L, L_L, F_L, Z_Z, F_R, R_R, B_R   'Term Of Target Oreintation
'Declare Variables in Rule Base
Dim Rule(56)
Dim nb(), nm(), ns(), zz(), ps(), pm(), pb()   'Minimum of Output Steer Angle
Dim s(), m(), f()                       'Minimum of Output Robot Speed
Dim MaxNB, MaxNM, MaxNS, MaxZZ, MaxPS, MaxPM, MaxPB
Dim MaxS, MaxM, MaxF
'Declare Variables in Defuzzification
Dim Fraction, Denominator
Dim RobotSpeed, SteerAngle
Dim MemberSpeed, MemberSteer
Dim SteerAngleDegree, SteerAngleOut, SpeedOut
Dim Slow, Medium, Fast
Dim NegB, NegM, NegS, Zero, PosS, PosM, PosB
'Declare variables for Robot and Target
Dim TargetDirection, RobotDirection     'Direction of Target and Direction of Robot
Dim DistanceX, DistanceY                'Distance X Axis,Y Axis Between Robot and Target
Dim TargetAngle, DifferAngle, RobotRotateAngle
Dim RobotAngle, OldRobotAngle, NewRobotAngle
Dim RobotStep, RobotStepX, RobotStepY   'Step of Robot,in X-Axis and Y-Axis
'Declare variables for check repeat rotate loop
Dim Sum_SteerLeft, Sum_SteerRight
'Declare Constant
Const PI = 3.14159265358979
Const Tolerant = 0.05                   'Tolerant for accomplish
'Declare Boolean
Dim PressMouse As Boolean, CreatObs As Boolean
Dim PlaceRbt As Boolean, PlaceTgt As Boolean
Dim LimitSwitchLeft, LimitSwitchRight As Boolean
'Declare flag to stop graphic routines in Do Loops.
Dim DoFlag As Boolean
'Declare TimeLength Display
Dim TimeLengthMS, TimeLength
'Declare Distance Display
Dim DistanceSum
'Declare Creat Trackline
Dim Trackline
'Declare Number of Loop
Dim LoopCount
```

```
Private Sub Form_Load()
'Initiation of program
    DoFlag = False
    RobotStep = 0.01
    Time.Interval = 100
'Initiattion of Robot
    Robot.ScaleMode = 7
    Robot.Width = 1.5
    Robot.Height = 1.5
    Robot.Left = 0 + 10
    Robot.Top = 0 + 10
    FrontLine.Visible = False
    LeftLine.Visible = False
    RightLine.Visible = False
    BehindLine.Visible = False
'Initiation of Target
    Target.Left = Area.ScaleWidth - (Target.Width + 10)
    Target.Top = Area.ScaleHeight - (Target.Height + 10)
'Initiattion of Area Space
    Area.ScaleMode = 7
    Area.Width = 23
    Area.Height = 15
    Area.DrawWidth = 20
    Area.Line (0, 0)-Step(Area.ScaleWidth, 0)
    Area.Line -Step(0, Area.ScaleHeight)
    Area.Line -Step(-Area.ScaleWidth, 0)
    Area.Line -Step(0, -Area.ScaleHeight)
    Area.DrawWidth = 1
    Area.ScaleMode = vbCentimeters
'Initiate Trackline
    Trackline = 1
    LoopCount = 0
End Sub
```

```
Private Sub ExitButton_Click()
    Unload Me
    End
End Sub
```

```
Private Sub ClrScrButton_Click()
'Cleer Screen
    Area.Cls
    Area.ScaleMode = vbPixels
    Area.DrawWidth = 20
    Area.Line (0, 0)-Step(Area.ScaleWidth, 0)
    Area.Line -Step(0, Area.ScaleHeight)
    Area.Line -Step(-Area.ScaleWidth, 0)
    Area.Line -Step(0, -Area.ScaleHeight)
    Area.DrawWidth = 1
    Area.ScaleMode = vbCentimeters
```

```
    DoFlag = False
    DistanceSum = 0
    LblDistance.Caption = FormatNumber(DistanceSum, 1)
End Sub
---
Private Sub RunButton_Click()
    DoFlag = True
    Robot.Visible = False
    CheckTargetAngle
    FrontLine.Visible = True
    LeftLine.Visible = True
    RightLine.Visible = True
    BehindLine.Visible = True
    TimeLengthMS = 0
End Sub
---
Private Sub StopButton_Click()
    DoFlag = False
    Robot.Visible = True
    FrontLine.Visible = False
    LeftLine.Visible = False
    RightLine.Visible = False
    BehindLine.Visible = False
    LblDistance.Caption = FormatNumber(DistanceSum, 1)
End Sub
---
Private Sub ObsButton_Click()
    'Creat Obstacle
    CreatObs = True
End Sub
---
Private Sub Area_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As Single)
    PressMouse = True
    If PlaceRbt Then
        Robot.Left = X - (Robot.Width / 2)
        Robot.Top = Y - (Robot.Height / 2)
    End If
    If PlaceTgt Then
        Target.Left = X - (Target.Width / 2)
        Target.Top = Y - (Target.Height / 2)
    End If
End Sub
---
Private Sub Area_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As Single)
    If PressMouse And CreatObs Then
        Area.Line (X - 0.2, Y - 0.2)-(X + 0.2, Y + 0.2), vbBlack, BF
    End If
End Sub
---
Private Sub Area_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As Single)
```

```
    PressMouse = False
    CreatObs = False
    PlaceRbt = False
    PlaceTgt = False
End Sub
---------------------------------------------------------------------------------------------------------------------
Private Sub PlaceRobot_Click()
'Place Robot Posotion
    PlaceRbt = True
    CheckTargetAngle
    OldRobotAngle = TargetAngle
    DoFlag = False
    Robot.Visible = True
    FrontLine.Visible = False
    LeftLine.Visible = False
    RightLine.Visible = False
    BehindLine.Visible = False
    DistanceSum = 0
    LblDistance.Caption = FormatNumber(DistanceSum, 1)
End Sub
---------------------------------------------------------------------------------------------------------------------
Private Sub PlaceTarget_Click()
'Place Target Position
    PlaceTgt = True
    CheckTargetAngle
    OldRobotAngle = TargetAngle
    DoFlag = False
    FrontLine.Visible = False
    LeftLine.Visible = False
    RightLine.Visible = False
    BehindLine.Visible = False
    DistanceSum = 0
    LblDistance.Caption = FormatNumber(DistanceSum, 1)
End Sub
---------------------------------------------------------------------------------------------------------------------
Private Sub Time_Timer()
  If DoFlag = True Then
    If ((Abs(DistanceX) > Tolerant) Or (Abs(DistanceY) > Tolerant)) Then
      CheckTargetAngle
      FindOldRobotDirection
      CheckFrontObs
      CheckLeftObs
      CheckRightObs
      CheckEdge
        If LimitSwitchLeft = True Then
          SpeedOut = -0.01
          SteerAngleOut = PI / 10
        GoTo IswOn:
        End If
        If LimitSwitchRight = True Then
```

```
            SpeedOut = -0.01
            SteerAngleOut = -PI / 10
          GoTo IswOn:
          End If
        CheckDifferAngle
        FuzzifierInput
        Rules
        Defuzzifier
        CheckRotateRepeat
IswOn:  ChangeRobotAngle
        FindNewRobotDirection
        RobotMove
        CreatRobotLine
        LoopCount = LoopCount + 1
        If (LoopCount Mod 2) = 0 Then
          CreatTrackLine
        End If
        DistanceSum = DistanceSum + (Abs(SpeedOut) * 10)
        TimeLengthMS = TimeLengthMS + 100
      End If
      TimeLength = TimeLengthMS / 1000
      LblTime.Caption = TimeLength
      LblDistance.Caption = FormatNumber(DistanceSum, 1)
      LblSpeed.Caption = FormatNumber(SpeedOut * 10, 3)
  End If
End Sub
```

```
Public Sub CheckTargetAngle()
  DistanceX = Target.Left - Robot.Left
  DistanceY = Target.Top - Robot.Top
  If DistanceX = 0 Then
    TargetAngle = PI / 2
  Else
    TargetAngle = Abs(Atn(DistanceY / DistanceX))
  End If
  If ((DistanceX <= 0) And (DistanceY <= 0)) Then
    TargetAngle = TargetAngle - PI
    TargetDirection = 1       'Left Up
  ElseIf ((DistanceX > 0) And (DistanceY <= 0)) Then
    TargetAngle = -TargetAngle
    TargetDirection = 2       'Right UP
  ElseIf ((DistanceX <= 0) And (DistanceY > 0)) Then
    TargetAngle = PI - TargetAngle
    TargetDirection = 3       'Left Down
  ElseIf ((DistanceX > 0) And (DistanceY > 0)) Then
    TargetAngle = TargetAngle
    TargetDirection = 4       'Right Down
  End If
End Sub
```

```
Public Sub FindOldRobotDirection()
  If (-PI <= OldRobotAngle And OldRobotAngle <= -PI / 2) Then
    RobotDirection = 1      'Left Up
    RobotAngle = OldRobotAngle + PI
  ElseIf (-PI / 2 < OldRobotAngle And OldRobotAngle <= 0) Then
    RobotDirection = 2      'Right Up
    RobotAngle = -OldRobotAngle
  ElseIf (PI / 2 < OldRobotAngle And OldRobotAngle <= PI) Then
    RobotDirection = 3      'Left Down
    RobotAngle = PI - OldRobotAngle
  ElseIf (0 < OldRobotAngle And OldRobotAngle <= PI / 2) Then
    RobotDirection = 4      'Right Down
    RobotAngle = OldRobotAngle
  End If
End Sub
```

```
Private Sub CheckFrontObs()
  Dim X, Y, StartX, StartY
Select Case RobotDirection
  Case 1                'left up
    RobotStepX = RobotStep * Cos(RobotAngle)
    RobotStepY = RobotStep * Sin(RobotAngle)
    StartX = (Robot.Left + Robot.Width / 2) - ((Robot.Height / 2) * Cos(RobotAngle))
    StartY = (Robot.Top + Robot.Height / 2) - ((Robot.Height / 2) * Sin(RobotAngle))
    X = StartX
    Y = StartY
      Do Until (Area.Point(X, Y) = vbBlack)
        X = X - RobotStepX
        Y = Y - RobotStepY
      Loop
    X = X - StartX
    Y = Y - StartY
    Distance_FrontObs = Sqr(X ^ 2 + Y ^ 2)
  Case 2                'right up
    RobotStepX = RobotStep * Cos(RobotAngle)
    RobotStepY = RobotStep * Sin(RobotAngle)
    StartX = (Robot.Left + Robot.Width / 2) + ((Robot.Height / 2) * Cos(RobotAngle))
    StartY = (Robot.Top + Robot.Height / 2) - ((Robot.Height / 2) * Sin(RobotAngle))
    X = StartX
    Y = StartY
      Do Until (Area.Point(X, Y) = vbBlack)
        X = X + RobotStepX
        Y = Y - RobotStepY
      Loop
    X = X - StartX
    Y = Y - StartY
    Distance_FrontObs = Sqr(X ^ 2 + Y ^ 2)
  Case 3                'left down
    RobotStepX = RobotStep * Cos(RobotAngle)
    RobotStepY = RobotStep * Sin(RobotAngle)
```

```
        StartX = (Robot.Left + Robot.Width / 2) - ((Robot.Height / 2) * Cos(RobotAngle))
        StartY = (Robot.Top + Robot.Height / 2) + ((Robot.Height / 2) * Sin(RobotAngle))
        X = StartX
        Y = StartY
          Do Until (Area.Point(X, Y) = vbBlack)
            X = X - RobotStepX
            Y = Y + RobotStepY
          Loop
        X = X - StartX
        Y = Y - StartY
        Distance_FrontObs = Sqr(X ^ 2 + Y ^ 2)
    Case 4                  'right down
        RobotStepX = RobotStep * Cos(RobotAngle)
        RobotStepY = RobotStep * Sin(RobotAngle)
        StartX = (Robot.Left + Robot.Width / 2) + ((Robot.Height / 2) * Cos(RobotAngle))
        StartY = (Robot.Top + Robot.Height / 2) + ((Robot.Height / 2) * Sin(RobotAngle))
        X = StartX
        Y = StartY
          Do Until (Area.Point(X, Y) = vbBlack)
            X = X + RobotStepX
            Y = Y + RobotStepY
          Loop
        X = X - StartX
        Y = Y - StartY
        Distance_FrontObs = Sqr(X ^ 2 + Y ^ 2)
    End Select
End Sub
```

```
Private Sub CheckLeftObs()
    Dim X, Y, StartX, StartY
    FindOldRobotDirection
Select Case RobotDirection
    Case 1                  'left up
        RobotAngle = (PI / 2) - RobotAngle
        RobotStepX = RobotStep * Cos(RobotAngle)
        RobotStepY = RobotStep * Sin(RobotAngle)
        StartX = (Robot.Left + Robot.Width / 2) - ((Robot.Width / 2) * Cos(RobotAngle))
        StartY = (Robot.Top + Robot.Height / 2) + ((Robot.Width / 2) * Sin(RobotAngle))
        X = StartX
        Y = StartY
          Do Until (Area.Point(X, Y) = vbBlack)
            X = X - RobotStepX
            Y = Y + RobotStepY
          Loop
        X = X - StartX
        Y = Y - StartY
        Distance_LeftObs = Sqr(X ^ 2 + Y ^ 2)
    Case 2                  'right up
        RobotAngle = (PI / 2) - RobotAngle
        RobotStepX = RobotStep * Cos(RobotAngle)
```

```
        RobotStepY = RobotStep * Sin(RobotAngle)
        StartX = (Robot.Left + Robot.Width / 2) - ((Robot.Width / 2) * Cos(RobotAngle))
        StartY = (Robot.Top + Robot.Height / 2) - ((Robot.Width / 2) * Sin(RobotAngle))
        X = StartX
        Y = StartY
          Do Until (Area.Point(X, Y) = vbBlack)
            X = X - RobotStepX
            Y = Y - RobotStepY
          Loop
        X = X - StartX
        Y = Y - StartY
        Distance_LeftObs = Sqr(X ^ 2 + Y ^ 2)
    Case 3                  'left down
        RobotAngle = (PI / 2) - RobotAngle
        RobotStepX = RobotStep * Cos(RobotAngle)
        RobotStepY = RobotStep * Sin(RobotAngle)
        StartX = (Robot.Left + Robot.Width / 2) + ((Robot.Width / 2) * Cos(RobotAngle))
        StartY = (Robot.Top + Robot.Height / 2) + ((Robot.Width / 2) * Sin(RobotAngle))
        X = StartX
        Y = StartY
          Do Until (Area.Point(X, Y) = vbBlack)
            X = X + RobotStepX
            Y = Y + RobotStepY
          Loop
        X = X - StartX
        Y = Y - StartY
        Distance_LeftObs = Sqr(X ^ 2 + Y ^ 2)
    Case 4                  'right down
        RobotAngle = (PI / 2) - RobotAngle
        RobotStepX = RobotStep * Cos(RobotAngle)
        RobotStepY = RobotStep * Sin(RobotAngle)
        StartX = (Robot.Left + Robot.Width / 2) + ((Robot.Width / 2) * Cos(RobotAngle))
        StartY = (Robot.Top + Robot.Height / 2) - ((Robot.Width / 2) * Sin(RobotAngle))
        X = StartX
        Y = StartY
          Do Until (Area.Point(X, Y) = vbBlack)
            X = X + RobotStepX
            Y = Y - RobotStepY
          Loop
        X = X - StartX
        Y = Y - StartY
        Distance_LeftObs = Sqr(X ^ 2 + Y ^ 2)
    End Select
End Sub
-------------------------------------------------------------------------------------------------------------------------------------------------------
Private Sub CheckRightObs()
    Dim X, Y, StartX, StartY
    FindOldRobotDirection
Select Case RobotDirection
    Case 1                  'left up
```

```
        RobotAngle = (PI / 2) - RobotAngle
        RobotStepX = RobotStep * Cos(RobotAngle)
        RobotStepY = RobotStep * Sin(RobotAngle)
        StartX = (Robot.Left + Robot.Width / 2) + ((Robot.Width / 2) * Cos(RobotAngle))
        StartY = (Robot.Top + Robot.Height / 2) - ((Robot.Width / 2) * Sin(RobotAngle))
        X = StartX
        Y = StartY
          Do Until (Area.Point(X, Y) = vbBlack)
            X = X + RobotStepX
            Y = Y - RobotStepY
          Loop
        X = X - StartX
        Y = Y - StartY
        Distance_RightObs = Sqr(X ^ 2 + Y ^ 2)
    Case 2                  'right up
        RobotAngle = (PI / 2) - RobotAngle
        RobotStepX = RobotStep * Cos(RobotAngle)
        RobotStepY = RobotStep * Sin(RobotAngle)
        StartX = (Robot.Left + Robot.Width / 2) + ((Robot.Width / 2) * Cos(RobotAngle))
        StartY = (Robot.Top + Robot.Height / 2) + ((Robot.Width / 2) * Sin(RobotAngle))
        X = StartX
        Y = StartY
          Do Until (Area.Point(X, Y) = vbBlack)
            X = X + RobotStepX
            Y = Y + RobotStepY
          Loop
        X = X - StartX
        Y = Y - StartY
        Distance_RightObs = Sqr(X ^ 2 + Y ^ 2)
    Case 3                  'left down
        RobotAngle = (PI / 2) - RobotAngle
        RobotStepX = RobotStep * Cos(RobotAngle)
        RobotStepY = RobotStep * Sin(RobotAngle)
        StartX = (Robot.Left + Robot.Width / 2) - ((Robot.Width / 2) * Cos(RobotAngle))
        StartY = (Robot.Top + Robot.Height / 2) - ((Robot.Width / 2) * Sin(RobotAngle))
        X = StartX
        Y = StartY
          Do Until (Area.Point(X, Y) = vbBlack)
            X = X - RobotStepX
            Y = Y - RobotStepY
          Loop
        X = X - StartX
        Y = Y - StartY
        Distance_RightObs = Sqr(X ^ 2 + Y ^ 2)
    Case 4                  'right down
        RobotAngle = (PI / 2) - RobotAngle
        RobotStepX = RobotStep * Cos(RobotAngle)
        RobotStepY = RobotStep * Sin(RobotAngle)
        StartX = (Robot.Left + Robot.Width / 2) - ((Robot.Width / 2) * Cos(RobotAngle))
        StartY = (Robot.Top + Robot.Height / 2) + ((Robot.Width / 2) * Sin(RobotAngle))
```

```
        X = StartX
        Y = StartY
          Do Until (Area.Point(X, Y) = vbBlack)
            X = X - RobotStepX
            Y = Y + RobotStepY
          Loop
        X = X - StartX
        Y = Y - StartY
        Distance_RightObs = Sqr(X ^ 2 + Y ^ 2)
    End Select
End Sub
```

```
Public Sub CheckEdge()
    Dim Zeta, FrontLeftAng, FrontRightAng, LeftAngle, RightAngle
    Dim Diag
    Dim FrontLeftX, FrontLeftY, FrontRightX, FrontRightY
    Diag = Sqr(((0.5 * Robot.Width) ^ 2) + ((0.5 * Robot.Height) ^ 2))
    Zeta = Atn((Robot.Width / 2) / (Robot.Height / 2))
    FrontLeftAng = OldRobotAngle - Zeta
    FrontRightAng = OldRobotAngle + Zeta
    ' if It Over Circle  or 360 Degree
    If FrontLeftAng < -PI Then
        FrontLeftAng = FrontLeftAng + (2 * PI)
    End If
    If FrontRightAng > PI Then
        FrontRightAng = FrontRightAng - (2 * PI)
    End If
    If (-PI <= FrontLeftAng And FrontLeftAng <= -PI / 2) Then
        LeftAngle = FrontLeftAng + PI
        FrontLeftX = (Robot.Left + Robot.Width / 2) - Diag * Cos(LeftAngle)
        FrontLeftY = (Robot.Top + Robot.Height / 2) - Diag * Sin(LeftAngle)
    ElseIf (-PI / 2 < FrontLeftAng And FrontLeftAng <= 0) Then
        LeftAngle = -FrontLeftAng
        FrontLeftX = (Robot.Left + Robot.Width / 2) + Diag * Cos(LeftAngle)
        FrontLeftY = (Robot.Top + Robot.Height / 2) - Diag * Sin(LeftAngle)
    ElseIf (PI / 2 < FrontLeftAng And FrontLeftAng <= PI) Then
        LeftAngle = PI - FrontLeftAng
        FrontLeftX = (Robot.Left + Robot.Width / 2) - Diag * Cos(LeftAngle)
        FrontLeftY = (Robot.Top + Robot.Height / 2) + Diag * Sin(LeftAngle)
    ElseIf (0 < FrontLeftAng And FrontLeftAng <= PI / 2) Then
        LeftAngle = FrontLeftAng
        FrontLeftX = (Robot.Left + Robot.Width / 2) + Diag * Cos(LeftAngle)
        FrontLeftY = (Robot.Top + Robot.Height / 2) + Diag * Sin(LeftAngle)
    End If
    If (-PI <= FrontRightAng And FrontRightAng <= -PI / 2) Then
        RightAngle = FrontRightAng + PI
        FrontRightX = (Robot.Left + Robot.Width / 2) - Diag * Cos(RightAngle)
        FrontRightY = (Robot.Top + Robot.Height / 2) - Diag * Sin(RightAngle)
    ElseIf (-PI / 2 < FrontRightAng And FrontRightAng <= 0) Then
        RightAngle = -FrontRightAng
```

```
        FrontRightX = (Robot.Left + Robot.Width / 2) + Diag * Cos(RightAngle)
        FrontRightY = (Robot.Top + Robot.Height / 2) - Diag * Sin(RightAngle)
    ElseIf (PI / 2 < FrontRightAng And FrontRightAng <= PI) Then
        RightAngle = PI - FrontRightAng
        FrontRightX = (Robot.Left + Robot.Width / 2) - Diag * Cos(RightAngle)
        FrontRightY = (Robot.Top + Robot.Height / 2) + Diag * Sin(RightAngle)
    ElseIf (0 < FrontRightAng And FrontRightAng <= PI / 2) Then
        RightAngle = FrontRightAng
        FrontRightX = (Robot.Left + Robot.Width / 2) + Diag * Cos(RightAngle)
        FrontRightY = (Robot.Top + Robot.Height / 2) + Diag * Sin(RightAngle)
    End If
    If Area.Point(FrontLeftX, FrontLeftY) = vbBlack Then
        LimitSwitchLeft = True
    Else
        LimitSwitchLeft = False
    End If
    If Area.Point(FrontRightX, FrontRightY) = vbBlack Then
        LimitSwitchRight = True
    Else
        LimitSwitchRight = False
    End If
End Sub
```

```
Private Sub CheckDifferAngle()
    'Find Difference Between Target Angle and Robot Angle
    Select Case TargetDirection
    Case 1                  'left up
        Select Case RobotDirection
            Case 1                      'left up
                If Abs(TargetAngle) > Abs(OldRobotAngle) Then           'DifferAngle Negative Target Locate Left of Robot
                    DifferAngle = -Abs(Abs(TargetAngle) - Abs(OldRobotAngle))
                Else                                                    'DifferAngle Positive
                    DifferAngle = Abs(Abs(TargetAngle) - Abs(OldRobotAngle))
                End If
            Case 2                                                      'right up
                DifferAngle = -Abs(Abs(TargetAngle) - Abs(OldRobotAngle))
            Case 3                                                      'left down
                DifferAngle = (2 * PI) - (Abs(TargetAngle) + Abs(OldRobotAngle))
            Case 4                                                      'right down
                If Abs(TargetAngle) + Abs(OldRobotAngle) < PI Then
                    DifferAngle = -(Abs(TargetAngle) + Abs(OldRobotAngle))          'Target Locate Left Behind of Robot
                Else
                    DifferAngle = (2 * PI) - (Abs(TargetAngle) + Abs(OldRobotAngle))    'Target Locate Right Behind of Robot
                End If
        End Select
    Case 2                      'right up
        Select Case RobotDirection
            Case 1                          'left up
                DifferAngle = Abs(Abs(OldRobotAngle) - Abs(TargetAngle))        'Target Right Of Robot
            Case 2                          'right up
```

```
            If Abs(TargetAngle) > Abs(OldRobotAngle) Then                    'DifferAngle Negative Target Locate Left of Robot
              DifferAngle = -Abs(Abs(TargetAngle) - Abs(OldRobotAngle))
            Else                                                             'DifferAngle Positive
              DifferAngle = Abs(Abs(TargetAngle) - Abs(OldRobotAngle))
            End If
          Case 3                    'left down
            If Abs(TargetAngle) + Abs(OldRobotAngle) < PI Then               'Target Locate Left Behind of Robot
              DifferAngle = -(Abs(TargetAngle) + Abs(OldRobotAngle))         'DifferAngle Negative
            Else                                                             'Target Locate Right Behind of Robot
              DifferAngle = (2 * PI) - (Abs(TargetAngle) + Abs(OldRobotAngle))  'DifferAngle Positive
            End If
          Case 4                    'right down
            DifferAngle = -(Abs(TargetAngle) + Abs(OldRobotAngle))
        End Select
    Case 3                  'left down
        Select Case RobotDirection
          Case 1                    'left up
            DifferAngle = -((2 * PI) - (Abs(TargetAngle) + Abs(OldRobotAngle))) 'DifferAngle Negative
          Case 2                    'right up
            If Abs(TargetAngle) + Abs(OldRobotAngle) < PI Then               'Target Locate Right Behind of Robot
              DifferAngle = Abs(TargetAngle) + Abs(OldRobotAngle)
            Else                                                             'Target Locate Left Behind of Robot
              DifferAngle = -((2 * PI) - (Abs(TargetAngle) + Abs(OldRobotAngle)))
            End If
          Case 3                    'left down
            If Abs(TargetAngle) > Abs(OldRobotAngle) Then                    'DifferAngle Positive Target Locate Right of Robot
              DifferAngle = Abs(Abs(TargetAngle) - Abs(OldRobotAngle))
            Else                                                             'DifferAngle Negative
              DifferAngle = -Abs(Abs(OldRobotAngle) - Abs(TargetAngle))
            End If
          Case 4                    'right down
            DifferAngle = Abs(Abs(TargetAngle) - Abs(OldRobotAngle))
        End Select
    Case 4                  'right down
        Select Case RobotDirection
          Case 1                    'left up
            If Abs(TargetAngle) + Abs(OldRobotAngle) < PI Then               'Target Locate Right Behind of Robot
              DifferAngle = Abs(TargetAngle) + Abs(OldRobotAngle)
            Else                                                            'Target Locate Left Behind of Robot
              DifferAngle = -Abs((2 * PI) - (Abs(TargetAngle) + Abs(OldRobotAngle)))
            End If
          Case 2                    'right up
            DifferAngle = Abs(TargetAngle) + Abs(OldRobotAngle)
          Case 3                    'left down
            DifferAngle = -Abs(Abs(TargetAngle) - Abs(OldRobotAngle))
          Case 4                    'right down
            If Abs(TargetAngle) > Abs(OldRobotAngle) Then                    'DifferAngle Positive Target Locate Right of Robot
              DifferAngle = Abs(Abs(TargetAngle) - Abs(OldRobotAngle))
            Else                                                             'DifferAngle Negative
              DifferAngle = -Abs(Abs(OldRobotAngle) - Abs(TargetAngle))
```

```
            End If
        End Select
    End Select
End Sub
---------------------------------------------------------------------------------------------------------------------------
Private Sub FuzzifierInput()
    Dim Obs_Near, Obs_Medium, Obs_Far
    Obs_Near = 1
    Obs_Medium = 2 * Obs_Near
    Obs_Far = 3 * Obs_Near
    'Fuzzification Input Of Fuzzy
    'Fuzzify Distance of Front Obstacle
    'Near Term
    If (Distance_FrontObs < Obs_Near) Then
      NF = 1
    ElseIf (Obs_Near <= Distance_FrontObs And Distance_FrontObs <= Obs_Far) Then
      NF = (Obs_Far - Distance_FrontObs) / Obs_Medium
    Else
      NF = 0
    End If
    'Far Term
    If (Obs_Far < Distance_FrontObs) Then
      FF = 1
    ElseIf (Obs_Near <= Distance_FrontObs And Distance_FrontObs <= Obs_Far) Then
      FF = (Distance_FrontObs - Obs_Near) / Obs_Medium
    Else
      FF = 0
    End If
    'Fuzzify Distance of Left Obstacle
    'Near Term
    If (Distance_LeftObs <= Obs_Near) Then
      NL = 1
    ElseIf (Obs_Near <= Distance_LeftObs And Distance_LeftObs <= Obs_Far) Then
      NL = (Obs_Far - Distance_LeftObs) / Obs_Medium
    Else
      NL = 0
    End If
    'Far Term
    If (Obs_Far <= Distance_LeftObs) Then
      FL = 1
    ElseIf (Obs_Near <= Distance_LeftObs And Distance_LeftObs <= Obs_Far) Then
      FL = (Distance_LeftObs - Obs_Near) / Obs_Medium
    Else
      FL = 0
    End If
    'Fuzzify Distance of Right Obstacle
    'Near Term
    If (Distance_RightObs <= Obs_Near) Then
      NR = 1
    ElseIf (Obs_Near <= Distance_RightObs And Distance_RightObs <= Obs_Far) Then
```

```
    NR = (Obs_Far - Distance_RightObs) / Obs_Medium
  Else
    NR = 0
  End If
  'Far Term
  If (Obs_Far <= Distance_RightObs) Then
    FR = 1
  ElseIf (Obs_Near <= Distance_RightObs And Distance_RightObs <= Obs_Far) Then
    FR = (Distance_RightObs - Obs_Near) / Obs_Medium
  Else
    FR = 0
  End If
  'Fuzzification Target Orientation
  If (DifferAngle <= ((-3 * PI) / 4)) Then
    B_L = 1
  ElseIf (((-3 * PI) / 4) < DifferAngle And DifferAngle <= (-PI / 2)) Then
    B_L = ((-PI / 2) - DifferAngle) / (PI / 4)
  Else
    B_L = 0
  End If
  If (((-3 * PI) / 4) < DifferAngle And DifferAngle <= (-PI / 2)) Then
    L_L = (DifferAngle + ((3 * PI) / 4)) / (PI / 4)
  ElseIf ((-PI / 2) < DifferAngle And DifferAngle <= (-PI / 4)) Then
    L_L = ((-PI / 4) - DifferAngle) / (PI / 4)
  Else
    L_L = 0
  End If
  If ((-PI / 2) < DifferAngle And DifferAngle <= (-PI / 4)) Then
    F_L = (DifferAngle + (PI / 2)) / (PI / 4)
  ElseIf ((-PI / 4) < DifferAngle And DifferAngle <= 0) Then
    F_L = (-DifferAngle) / (PI / 4)
  Else
    F_L = 0
  End If
  If ((-PI / 18) < DifferAngle And DifferAngle <= 0) Then
    Z_Z = (DifferAngle + (PI / 18)) / (PI / 18)
  ElseIf (0 < DifferAngle And DifferAngle < (PI / 18)) Then
    Z_Z = ((PI / 18) - DifferAngle) / (PI / 18)
  Else
    Z_Z = 0
  End If
  If (0 < DifferAngle And DifferAngle <= (PI / 4)) Then
    F_R = DifferAngle / (PI / 4)
  ElseIf ((PI / 4) < DifferAngle And DifferAngle <= (PI / 2)) Then
    F_R = ((PI / 2) - DifferAngle) / (PI / 4)
  Else
    F_R = 0
  End If
  If ((PI / 4) < DifferAngle And DifferAngle <= (PI / 2)) Then
    R_R = (DifferAngle - (PI / 4)) / (PI / 4)
```

```
    ElseIf ((PI / 2) < DifferAngle And DifferAngle < (3 * PI / 4)) Then
      R_R = ((3 * PI / 4) - DifferAngle) / (PI / 4)
    Else
      R_R = 0
    End If
    If ((3 * PI / 4) <= DifferAngle) Then
      B_R = 1
    ElseIf ((PI / 2) < DifferAngle And DifferAngle < (3 * PI / 4)) Then
      B_R = (DifferAngle - (PI / 2)) / (PI / 4)
    Else
      B_R = 0
    End If
End Sub
```

```
Private Sub Rules()
    Rule(1) = Min(NF, NL, NR, B_L)
    Rule(2) = Min(NF, NL, NR, L_L)
    Rule(3) = Min(NF, NL, NR, F_L)
    Rule(4) = Min(NF, NL, NR, Z_Z)
    Rule(5) = Min(NF, NL, NR, F_R)
    Rule(6) = Min(NF, NL, NR, R_R)
    Rule(7) = Min(NF, NL, NR, B_R)
    Rule(8) = Min(NF, NL, FR, B_L)
    Rule(9) = Min(NF, NL, FR, L_L)
    Rule(10) = Min(NF, NL, FR, F_L)
    Rule(11) = Min(NF, NL, FR, Z_Z)
    Rule(12) = Min(NF, NL, FR, F_R)
    Rule(13) = Min(NF, NL, FR, R_R)
    Rule(14) = Min(NF, NL, FR, B_R)
    Rule(15) = Min(NF, FL, NR, B_L)
    Rule(16) = Min(NF, FL, NR, L_L)
    Rule(17) = Min(NF, FL, NR, F_L)
    Rule(18) = Min(NF, FL, NR, Z_Z)
    Rule(19) = Min(NF, FL, NR, F_R)
    Rule(20) = Min(NF, FL, NR, R_R)
    Rule(21) = Min(NF, FL, NR, B_R)
    Rule(22) = Min(NF, FL, FR, B_L)
    Rule(23) = Min(NF, FL, FR, L_L)
    Rule(24) = Min(NF, FL, FR, F_L)
    Rule(25) = Min(NF, FL, FR, Z_Z)
    Rule(26) = Min(NF, FL, FR, F_R)
    Rule(27) = Min(NF, FL, FR, R_R)
    Rule(28) = Min(NF, FL, FR, B_R)
    Rule(29) = Min(FF, NL, NR, B_L)
    Rule(30) = Min(FF, NL, NR, L_L)
    Rule(31) = Min(FF, NL, NR, F_L)
    Rule(32) = Min(FF, NL, NR, Z_Z)
    Rule(33) = Min(FF, NL, NR, F_R)
    Rule(34) = Min(FF, NL, NR, R_R)
    Rule(35) = Min(FF, NL, NR, B_R)
```

```
Rule(36) = Min(FF, NL, FR, B_L)
Rule(37) = Min(FF, NL, FR, L_L)
Rule(38) = Min(FF, NL, FR, F_L)
Rule(39) = Min(FF, NL, FR, Z_Z)
Rule(40) = Min(FF, NL, FR, F_R)
Rule(41) = Min(FF, NL, FR, R_R)
Rule(42) = Min(FF, NL, FR, B_R)
Rule(43) = Min(FF, FL, NR, B_L)
Rule(44) = Min(FF, FL, NR, L_L)
Rule(45) = Min(FF, FL, NR, F_L)
Rule(46) = Min(FF, FL, NR, Z_Z)
Rule(47) = Min(FF, FL, NR, F_R)
Rule(48) = Min(FF, FL, NR, R_R)
Rule(49) = Min(FF, FL, NR, B_R)
Rule(50) = Min(FF, FL, FR, B_L)
Rule(51) = Min(FF, FL, FR, L_L)
Rule(52) = Min(FF, FL, FR, F_L)
Rule(53) = Min(FF, FL, FR, Z_Z)
Rule(54) = Min(FF, FL, FR, F_R)
Rule(55) = Min(FF, FL, FR, R_R)
Rule(56) = Min(FF, FL, FR, B_R)
'Find MAX of Negative Big
ReDim nb(5)
nb(1) = Rule(1)
nb(2) = Rule(2)
nb(3) = Rule(3)
nb(4) = Rule(15)
nb(5) = Rule(22)
MaxNB = Max(nb())
'Find MAX of Negative Medium
ReDim nm(4)
nm(1) = Rule(4)
nm(2) = Rule(16)
nm(3) = Rule(23)
nm(4) = Rule(26)
MaxNM = Max(nm())
'Find MAX of Negative Small
ReDim  ns(14)
ns(1) = Rule(17)
ns(2) = Rule(18)
ns(3) = Rule(19)
ns(4) = Rule(20)
ns(5) = Rule(21)
ns(6) = Rule(36)
ns(7) = Rule(37)
ns(8) = Rule(38)
ns(9) = Rule(42)
ns(10) = Rule(44)
ns(11) = Rule(45)
ns(12) = Rule(50)
```

```
ns(13) = Rule(51)
ns(14) = Rule(52)
MaxNS = Max(ns())
'Find MAX of Zero
ReDim zz(11)
zz(1) = Rule(29)
zz(2) = Rule(30)
zz(3) = Rule(31)
zz(4) = Rule(32)
zz(5) = Rule(33)
zz(6) = Rule(34)
zz(7) = Rule(35)
zz(8) = Rule(39)
zz(9) = Rule(46)
zz(10) = Rule(53)
zz(11) = Rule(25)
MaxZZ = Max(zz())
'Find MAX of Positive Small
ReDim ps(14)
ps(1) = Rule(8)
ps(2) = Rule(9)
ps(3) = Rule(10)
ps(4) = Rule(11)
ps(5) = Rule(12)
ps(6) = Rule(40)
ps(7) = Rule(41)
ps(8) = Rule(43)
ps(9) = Rule(47)
ps(10) = Rule(48)
ps(11) = Rule(49)
ps(12) = Rule(54)
ps(13) = Rule(55)
ps(14) = Rule(56)
MaxPS = Max(ps())
'Find MAX of Positive Medium
ReDim pm(3)
pm(1) = Rule(13)
pm(2) = Rule(27)
pm(3) = Rule(24)
MaxPM = Max(pm())
'Find MAX of Positive Big
ReDim pb(5)
pb(1) = Rule(5)
pb(2) = Rule(6)
pb(3) = Rule(7)
pb(4) = Rule(14)
pb(5) = Rule(28)
MaxPB = Max(pb())
'Find MAX of Slow
ReDim s(10)
```

```
s(1) = Rule(1)
s(2) = Rule(2)
s(3) = Rule(3)
s(4) = Rule(4)
s(5) = Rule(5)
s(6) = Rule(6)
s(7) = Rule(7)
s(8) = Rule(24)
s(9) = Rule(25)
s(10) = Rule(26)
MaxS = Max(s())
'Find MAX of Medium
ReDim m(16)
m(1) = Rule(8)
m(2) = Rule(9)
m(3) = Rule(10)
m(4) = Rule(11)
m(5) = Rule(18)
m(6) = Rule(19)
m(7) = Rule(20)
m(8) = Rule(21)
m(9) = Rule(23)
m(10) = Rule(27)
m(11) = Rule(29)
m(12) = Rule(30)
m(13) = Rule(31)
m(14) = Rule(33)
m(15) = Rule(34)
m(16) = Rule(35)
MaxM = Max(m())
'Find MAX of Fast
ReDim f(30)
f(1) = Rule(12)
f(2) = Rule(13)
f(3) = Rule(14)
f(4) = Rule(15)
f(5) = Rule(16)
f(6) = Rule(17)
f(7) = Rule(22)
f(8) = Rule(28)
f(9) = Rule(32)
f(10) = Rule(36)
f(11) = Rule(37)
f(12) = Rule(38)
f(13) = Rule(39)
f(14) = Rule(40)
f(15) = Rule(41)
f(16) = Rule(42)
f(17) = Rule(43)
f(18) = Rule(44)
```

```
    f(19) = Rule(45)
    f(20) = Rule(46)
    f(21) = Rule(47)
    f(22) = Rule(48)
    f(23) = Rule(49)
    f(24) = Rule(50)
    f(25) = Rule(51)
    f(26) = Rule(52)
    f(27) = Rule(53)
    f(28) = Rule(54)
    f(29) = Rule(55)
    f(30) = Rule(56)
    MaxF = Max(f())
End Sub
```

```
Private Function Max(MaxTerm())
    Dim Temp
    Dim SizeMax
    SizeMax = UBound(MaxTerm)
    maximum = 0
    Dim Count
    For Count = 1 To SizeMax
        If MaxTerm(Count) > maximum Then
            maximum = MaxTerm(Count)
        End If
    Next
    Max = maximum
End Function
```

```
Private Function Min(Fnt, Lft, Rgt, Ang)
    Dim Minimum
    Minimum = 1
    If (Minimum >= Fnt) Then
        Minimum = Fnt
    End If
    If (Minimum > Lft) Then
        Minimum = Lft
    End If
    If (Minimum > Rgt) Then
        Minimum = Rgt
    End If
    If (Minimum > Ang) Then
        Minimum = Ang
    End If
    Min = Minimum
End Function
```

```
Private Sub Defuzzifier()
    DefuzzifierOut1
    DefuzzifierOut2
```

```
End Sub
```

```
Private Sub DefuzzifierOut1()
    Fraction = 0
    Denominator = o
    For SteerAngle = -15 To 15 Step 0.1
        FuzzifierOut1
        MinOperationOut1
        MaxOperationOut1
        Fraction = Fraction + (SteerAngle * MemberSteer)
        Denominator = Denominator + MemberSteer
    Next
    If Denominator = 0 Then
        SteerAngleOut = 0
    Else
        SteerAngleDegree = Fraction / Denominator
        SteerAngleOut = (SteerAngleDegree * PI) / 180          'Tranformation Degree to Radius
    End If
End Sub
```

```
Private Sub FuzzifierOut1()
    'Fuzzification Robot Steer Angle
    'Negative Big Term
    If (-15 <= SteerAngle And SteerAngle <= -10) Then
        NegB = (-10 - SteerAngle) / 5
    Else
        NegB = 0
    End If
    'Negative Medium Term
    If (-15 <= SteerAngle And SteerAngle <= -10) Then
        NegM = (SteerAngle + 15) / 5
    ElseIf (-10 <= SteerAngle And SteerAngle <= -5) Then
        NegM = (-5 - SteerAngle) / 5
    Else
        NegM = 0
    End If
    'Negative Small Term
    If (-10 <= SteerAngle And SteerAngle <= -5) Then
        NegS = (SteerAngle + 10) / 5
    ElseIf (-5 <= SteerAngle And SteerAngle <= 0) Then
        NegS = (-SteerAngle) / 5
    Else
        NegS = 0
    End If
    'Zero Term
    If (-5 < SteerAngle And SteerAngle <= 0) Then
        Zero = (SteerAngle + 5) / 5
    ElseIf (0 < SteerAngle And SteerAngle < 5) Then
        Zero = (5 - SteerAngle) / 5
    Else
```

```
        Zero = 0
    End If
    'Positive Small Term
    If (0 <= SteerAngle And SteerAngle <= 5) Then
        PosS = SteerAngle / 5
    ElseIf (5 <= SteerAngle And SteerAngle <= 10) Then
        PosS = (10 - SteerAngle) / 5
    Else
        PosS = 0
    End If
    'Positive Medium Term
    If (5 <= SteerAngle And SteerAngle <= 10) Then
        PosM = (SteerAngle - 5) / 5
    ElseIf (10 <= SteerAngle And SteerAngle <= 15) Then
        PosM = (15 - SteerAngle) / 5
    Else
        PosM = 0
    End If
    'Positive Big Term
    If (10 <= SteerAngle And SteerAngle <= 15) Then
        PosB = (SteerAngle - 10) / 5
    Else
        PosB = 0
    End If
End Sub
```

```
Private Sub MinOperationOut1()
    If NegB > MaxNB Then
        NegB = MaxNB
    End If
    If NegM > MaxNM Then
        NegM = MaxNM
    End If
    If NegS > MaxNS Then
        NegS = MaxNS
    End If
    If Zero > MaxZZ Then
        Zero = MaxZZ
    End If
    If PosS > MaxPS Then
        PosS = MaxPS
    End If
    If PosM > MaxPM Then
        PosM = MaxPM
    End If
    If PosB > MaxPB Then
        PosB = MaxPB
    End If
End Sub
```

```
Private Sub MaxOperationOut1()
    If (-15 <= SteerAngle And SteerAngle <= -10) Then
        If NegB > NegM Then
            MemberSteer = NegB
        ElseIf NegB <= NegM Then
            MemberSteer = NegM
        End If
    End If
    If (-10 <= SteerAngle And SteerAngle <= -5) Then
        If NegM > NegS Then
            MemberSteer = NegM
        ElseIf NegM <= NegS Then
            MemberSteer = NegS
        End If
    End If
    If (-5 <= SteerAngle And SteerAngle <= 0) Then
        If NegS > Zero Then
            MemberSteer = NegS
        ElseIf NegS <= Zero Then
            MemberSteer = Zero
        End If
    End If
    If (0 <= SteerAngle And SteerAngle <= 5) Then
        If Zero > PosS Then
            MemberSteer = Zero
        ElseIf Zero <= PosS Then
            MemberSteer = PosS
        End If
    End If
    If (5 <= SteerAngle And SteerAngle <= 10) Then
        If PosS > PosM Then
            MemberSteer = PosS
        ElseIf PosS <= PosM Then
            MemberSteer = PosM
        End If
    End If
    If (10 <= SteerAngle And SteerAngle <= 15) Then
        If PosM > PosB Then
            MemberSteer = PosM
        ElseIf PosM <= PosB Then
            MemberSteer = PosB
        End If
    End If
End Sub
```

```
Private Sub DefuzzifierOut2()
    Fraction = 0
    Denominator = 0
    For RobotSpeed = 0 To 0.05 Step 0.001
        FuzzifierOut2
```

```
        MinOperationOut2
        MaxOperationOut2
        Fraction = Fraction + (RobotSpeed * MemberSpeed)
        Denominator = Denominator + MemberSpeed
    Next
    If Denominator = 0 Then
        SpeedOut = 0.05
    Else
        SpeedOut = Fraction / Denominator
    End If
End Sub
```

```
Private Sub FuzzifierOut2()
    'Fuzzification Robot Speed
    'Slow Term
    If (RobotSpeed <= 0) Then
        Slow = 1
    ElseIf (0 < RobotSpeed And RobotSpeed <= 0.025) Then
        Slow = 1 - (RobotSpeed / 0.025)
    Else
        Slow = 0
    End If
    'Medium Term
    If (0.02 < RobotSpeed And RobotSpeed <= 0.025) Then
        Medium = (RobotSpeed / 0.005) - 4
    ElseIf (0.025 < RobotSpeed And RobotSpeed <= 0.003) Then
        Medium = 6 - (RobotSpeed / 0.005)
    Else
        Medium = 0
    End If
    'Fast Term
    If (RobotSpeed >= 0.05) Then
        Fast = 1
    ElseIf (0.025 < RobotSpeed And RobotSpeed <= 0.05) Then
        Fast = (RobotSpeed / 0.025) - 1
    Else
        Fast = 0
    End If
End Sub
```

```
Private Sub MinOperationOut2()
    If Slow > MaxS Then
        Slow = MaxS
    End If
    If Medium > MaxM Then
        Medium = MaxM
    End If
    If Fast > MaxF Then
        Fast = MaxF
    End If
```

```
End Sub
```

```
Private Sub MaxOperationOut2()
  If (RobotSpeed >= 0 And RobotSpeed < 0.02) Then
    MemberSpeed = Slow
  End If
  If (RobotSpeed >= 0.02 And RobotSpeed < 0.025) Then
    If Slow > Medium Then
      MemberSpeed = Slow
    ElseIf Slow <= Medium Then
      MemberSpeed = Medium
    End If
  End If
  If (RobotSpeed >= 0.025 And RobotSpeed < 0.03) Then
    If Medium > Fast Then
      MemberSpeed = Medium
    ElseIf Medium <= Fast Then
      MemberSpeed = Fast
    End If
  End If
  If (RobotSpeed >= 0.03 And RobotSpeed <= 0.05) Then
    MemberSpeed = Fast
  End If
End Sub
```

```
Private Sub ChangeRobotAngle()
  RobotRotateAngle = SteerAngleOut
  RobotStep = SpeedOut
  If (-PI <= OldRobotAngle And OldRobotAngle <= -PI / 2) Then      'left up
    If RobotRotateAngle < 0 Then   'Rotate Left
      NewRobotAngle = OldRobotAngle + RobotRotateAngle
    Else                           'Rotate  Right
      NewRobotAngle = OldRobotAngle + RobotRotateAngle
    End If
  ElseIf (-PI / 2 < OldRobotAngle And OldRobotAngle <= 0) Then     'right up
    If RobotRotateAngle < 0 Then   'Rotate Left
      NewRobotAngle = OldRobotAngle + RobotRotateAngle
    Else                           'Rotate  Right
      NewRobotAngle = OldRobotAngle + RobotRotateAngle
    End If
  ElseIf (PI / 2 < OldRobotAngle And OldRobotAngle <= PI) Then     'left down
    If RobotRotateAngle < 0 Then   'Rotate Left
      NewRobotAngle = OldRobotAngle + RobotRotateAngle
    Else                           'Rotate  Right
      NewRobotAngle = OldRobotAngle + RobotRotateAngle
    End If
  ElseIf (0 < OldRobotAngle And OldRobotAngle <= PI / 2) Then      'right down
    If RobotRotateAngle < 0 Then   'Rotate Left
      NewRobotAngle = OldRobotAngle + RobotRotateAngle
    Else                           'Rotate  Right
```

```
            NewRobotAngle = OldRobotAngle + RobotRotateAngle
        End If
    End If
    'if It Over Circle  or 360 Degree
    If NewRobotAngle < -PI Then
        NewRobotAngle = NewRobotAngle + (2 * PI)
    End If
    If NewRobotAngle > PI Then
        NewRobotAngle = NewRobotAngle - (2 * PI)
    End If
    OldRobotAngle = NewRobotAngle
End Sub
```

```
Public Sub FindNewRobotDirection()
    If (-PI <= NewRobotAngle And NewRobotAngle <= -PI / 2) Then
        RobotDirection = 1        'Left Up
        RobotAngle = NewRobotAngle + PI
    ElseIf (-PI / 2 < NewRobotAngle And NewRobotAngle <= 0) Then
        RobotDirection = 2        'Right UP
        RobotAngle = -NewRobotAngle
    ElseIf (PI / 2 < NewRobotAngle And NewRobotAngle <= PI) Then
        RobotDirection = 3        'Left Down
        RobotAngle = PI - NewRobotAngle
    ElseIf (0 < NewRobotAngle And NewRobotAngle <= PI / 2) Then
        RobotDirection = 4        'Right Down
        RobotAngle = NewRobotAngle
    End If
End Sub
```

```
Public Sub RobotMove()
    RobotStepX = RobotStep * Cos(RobotAngle)
    RobotStepY = RobotStep * Sin(RobotAngle)
    Select Case RobotDirection
    Case 1
        ' Move the robot left up
        Robot.Move Robot.Left - RobotStepX, Robot.Top - RobotStepY
    Case 2
        ' Move the robot right up
        Robot.Move Robot.Left + RobotStepX, Robot.Top - RobotStepY
    Case 3
        ' Move the robot left down
        Robot.Move Robot.Left - RobotStepX, Robot.Top + RobotStepY
    Case 4
        ' Move the robot right down
        Robot.Move Robot.Left + RobotStepX, Robot.Top + RobotStepY
    End Select
End Sub
```

```
Public Sub CreatRobotLine()
    Dim Zeta, FrontLeftAng, FrontRightAng, Ang
```

```
Dim LineWidth
Dim FrontLeftX, FrontLeftY, FrontRightX, FrontRightY
Dim BehindLeftX, BehindLeftY, BehindRightX, BehindRightY
LineWidth = Sqr(((0.5 * Robot.Width) ^ 2) + ((0.5 * Robot.Height) ^ 2))
Zeta = Atn((Robot.Width / 2) / (Robot.Height / 2))
FrontLeftAng = OldRobotAngle - Zeta
FrontRightAng = OldRobotAngle + Zeta
' if It Over Circle  or 360 Degree
If FrontLeftAng < -PI Then
   FrontLeftAng = FrontLeftAng + (2 * PI)
End If
If FrontRightAng > PI Then
   FrontRightAng = FrontRightAng - (2 * PI)
End If
If (-PI <= FrontLeftAng And FrontLeftAng <= -PI / 2) Then
   Ang = FrontLeftAng + PI
   FrontLeftX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Ang)
   FrontLeftY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Ang)
   BehindRightX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Ang)
   BehindRightY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Ang)
ElseIf (-PI / 2 < FrontLeftAng And FrontLeftAng <= 0) Then
   Ang = -FrontLeftAng
   FrontLeftX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Ang)
   FrontLeftY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Ang)
   BehindRightX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Ang)
   BehindRightY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Ang)
ElseIf (PI / 2 < FrontLeftAng And FrontLeftAng <= PI) Then
   Ang = PI - FrontLeftAng
   FrontLeftX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Ang)
   FrontLeftY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Ang)
   BehindRightX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Ang)
   BehindRightY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Ang)
ElseIf (0 < FrontLeftAng And FrontLeftAng <= PI / 2) Then
   Ang = FrontLeftAng
   FrontLeftX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Ang)
   FrontLeftY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Ang)
   BehindRightX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Ang)
   BehindRightY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Ang)
End If
If (-PI <= FrontRightAng And FrontRightAng <= -PI / 2) Then
   Ang = FrontRightAng + PI
   FrontRightX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Ang)
   FrontRightY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Ang)
   BehindLeftX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Ang)
   BehindLeftY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Ang)
ElseIf (-PI / 2 < FrontRightAng And FrontRightAng <= 0) Then
   Ang = -FrontRightAng
   FrontRightX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Ang)
   FrontRightY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Ang)
   BehindLeftX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Ang)
```

```
        BehindLeftY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Ang)
    ElseIf (PI / 2 < FrontRightAng And FrontRightAng <= PI) Then
        Ang = PI - FrontRightAng
        FrontRightX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Ang)
        FrontRightY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Ang)
        BehindLeftX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Ang)
        BehindLeftY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Ang)
    ElseIf (0 < FrontRightAng And FrontRightAng <= PI / 2) Then
        Ang = FrontRightAng
        FrontRightX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Ang)
        FrontRightY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Ang)
        BehindLeftX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Ang)
        BehindLeftY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Ang)
    End If
    Dim FrontRobot, LeftRobot, RightRobot, BehindRobot As Line
    Set FrontRobot = FrontLine
        FrontRobot.X1 = FrontLeftX
        FrontRobot.X2 = FrontRightX
        FrontRobot.Y1 = FrontLeftY
        FrontRobot.Y2 = FrontRightY
        FrontRobot.BorderColor = vbGreen
        FrontRobot.BorderStyle = 6
        FrontRobot.BorderWidth = 10
    Set LeftRobot = LeftLine
        LeftRobot.X1 = BehindLeftX
        LeftRobot.X2 = FrontLeftX
        LeftRobot.Y1 = BehindLeftY
        LeftRobot.Y2 = FrontLeftY
        LeftRobot.BorderColor = vbBlue
        LeftRobot.BorderStyle = 6
        LeftRobot.BorderWidth = 5
    Set RightRobot = RightLine
        RightRobot.X1 = FrontRightX
        RightRobot.X2 = BehindRightX
        RightRobot.Y1 = FrontRightY
        RightRobot.Y2 = BehindRightY
        RightRobot.BorderColor = vbBlue
        RightRobot.BorderStyle = 6
        RightRobot.BorderWidth = 5
    Set BehindRobot = BehindLine
        BehindRobot.X1 = BehindRightX
        BehindRobot.X2 = BehindLeftX
        BehindRobot.Y1 = BehindRightY
        BehindRobot.Y2 = BehindLeftY
        BehindRobot.BorderColor = vbBlue
        BehindRobot.BorderStyle = 6
        BehindRobot.BorderWidth = 5
End Sub
```

```
Public Sub CreatTrackLine()
```

```
Dim Zeta, FrontLeftAng, FrontRightAng, Ang
Dim LineWidth
Dim FrontLeftX, FrontLeftY, FrontRightX, FrontRightY
Dim BehindLeftX, BehindLeftY, BehindRightX, BehindRightY
LineWidth = 0.3 * Sqr(((0.5 * Robot.Width) ^ 2) + ((0.5 * Robot.Height) ^ 2))
Zeta = Atn((Robot.Width / 2) / (Robot.Height / 2))
FrontLeftAng = OldRobotAngle - Zeta
FrontRightAng = OldRobotAngle + Zeta
' if It Over Circle  or 360 Degree
If FrontLeftAng < -PI Then
   FrontLeftAng = FrontLeftAng + (2 * PI)
End If
If FrontRightAng > PI Then
   FrontRightAng = FrontRightAng - (2 * PI)
End If
If (-PI <= FrontLeftAng And FrontLeftAng <= -PI / 2) Then
   Ang = FrontLeftAng + PI
   FrontLeftX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Ang)
   FrontLeftY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Ang)
   BehindRightX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Ang)
   BehindRightY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Ang)
ElseIf (-PI / 2 < FrontLeftAng And FrontLeftAng <= 0) Then
   Ang = -FrontLeftAng
   FrontLeftX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Ang)
   FrontLeftY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Ang)
   BehindRightX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Ang)
   BehindRightY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Ang)
ElseIf (PI / 2 < FrontLeftAng And FrontLeftAng <= PI) Then
   Ang = PI - FrontLeftAng
   FrontLeftX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Ang)
   FrontLeftY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Ang)
   BehindRightX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Ang)
   BehindRightY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Ang)
ElseIf (0 < FrontLeftAng And FrontLeftAng <= PI / 2) Then
   Ang = FrontLeftAng
   FrontLeftX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Ang)
   FrontLeftY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Ang)
   BehindRightX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Ang)
   BehindRightY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Ang)
End If
If (-PI <= FrontRightAng And FrontRightAng <= -PI / 2) Then
   Ang = FrontRightAng + PI
   FrontRightX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Ang)
   FrontRightY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Ang)
   BehindLeftX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Ang)
   BehindLeftY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Ang)
ElseIf (-PI / 2 < FrontRightAng And FrontRightAng <= 0) Then
   Ang = -FrontRightAng
   FrontRightX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Ang)
   FrontRightY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Ang)
```

```
    BehindLeftX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Ang)
    BehindLeftY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Ang)
  ElseIf (PI / 2 < FrontRightAng And FrontRightAng <= PI) Then
    Ang = PI - FrontRightAng
    FrontRightX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Ang)
    FrontRightY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Ang)
    BehindLeftX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Ang)
    BehindLeftY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Ang)
  ElseIf (0 < FrontRightAng And FrontRightAng <= PI / 2) Then
    Ang = FrontRightAng
    FrontRightX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Ang)
    FrontRightY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Ang)
    BehindLeftX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Ang)
    BehindLeftY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Ang)
  End If
  Select Case Trackline
    Case 1   'Square Trackline
      Area.Line (FrontLeftX, FrontLeftY)-(FrontRightX, FrontRightY), vbRed
      Area.Line (FrontRightX, FrontRightY)-(BehindRightX, BehindRightY), vbRed
      Area.Line (BehindRightX, BehindRightY)-(BehindLeftX, BehindLeftY), vbRed
      Area.Line (BehindLeftX, BehindLeftY)-(FrontLeftX, FrontLeftY), vbRed
    Case 2   'Circle Trackline
      Area.Circle (Robot.Left + Robot.Width / 2, Robot.Top + Robot.Height / 2), 0.8 * LineWidth, vbRed
    Case 3   'Plus Trackline
      Dim FrontX, FrontY, BehindX, BehindY
      Dim LeftX, LeftY, RightX, RightY
      Select Case RobotDirection
      Case 1
        FrontX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Abs(OldRobotAngle))
        FrontY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Abs(OldRobotAngle))
        BehindX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Abs(OldRobotAngle))
        BehindY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Abs(OldRobotAngle))
        LeftX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(PI / 2 - Abs(OldRobotAngle))
        LeftY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(PI / 2 - Abs(OldRobotAngle))
        RightX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(PI / 2 - Abs(OldRobotAngle))
        RightY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(PI / 2 - Abs(OldRobotAngle))
        Area.Line (FrontX, FrontY)-(BehindX, BehindY), vbRed
        Area.Line (LeftX, LeftY)-(RightX, RightY), vbRed
      Case 2
        FrontX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Abs(OldRobotAngle))
        FrontY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Abs(OldRobotAngle))
        BehindX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Abs(OldRobotAngle))
        BehindY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Abs(OldRobotAngle))
        LeftX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(PI / 2 - Abs(OldRobotAngle))
        LeftY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(PI / 2 - Abs(OldRobotAngle))
        RightX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(PI / 2 - Abs(OldRobotAngle))
        RightY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(PI / 2 - Abs(OldRobotAngle))
        Area.Line (FrontX, FrontY)-(BehindX, BehindY), vbRed
        Area.Line (LeftX, LeftY)-(RightX, RightY), vbRed
      Case 3
```

```
            FrontX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Abs(OldRobotAngle))
            FrontY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Abs(OldRobotAngle))
            BehindX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Abs(OldRobotAngle))
            BehindY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Abs(OldRobotAngle))
            LeftX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(PI / 2 - Abs(OldRobotAngle))
            LeftY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(PI / 2 - Abs(OldRobotAngle))
            RightX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(PI / 2 - Abs(OldRobotAngle))
            RightY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(PI / 2 - Abs(OldRobotAngle))
            Area.Line (FrontX, FrontY)-(BehindX, BehindY), vbRed
            Area.Line (LeftX, LeftY)-(RightX, RightY), vbRed
        Case 4
            FrontX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(Abs(OldRobotAngle))
            FrontY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(Abs(OldRobotAngle))
            BehindX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(Abs(OldRobotAngle))
            BehindY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(Abs(OldRobotAngle))
            LeftX = (Robot.Left + Robot.Width / 2) + LineWidth * Cos(PI / 2 - Abs(OldRobotAngle))
            LeftY = (Robot.Top + Robot.Height / 2) - LineWidth * Sin(PI / 2 - Abs(OldRobotAngle))
            RightX = (Robot.Left + Robot.Width / 2) - LineWidth * Cos(PI / 2 - Abs(OldRobotAngle))
            RightY = (Robot.Top + Robot.Height / 2) + LineWidth * Sin(PI / 2 - Abs(OldRobotAngle))
            Area.Line (FrontX, FrontY)-(BehindX, BehindY), vbRed
            Area.Line (LeftX, LeftY)-(RightX, RightY), vbRed
        End Select
    Case 4    'Cross Trackline
        Area.Line (FrontLeftX, FrontLeftY)-(BehindRightX, BehindRightY), vbRed
        Area.Line (FrontRightX, FrontRightY)-(BehindLeftX, BehindLeftY), vbRed
    Case 5    'Dot Trackline
        Area.Circle (Robot.Left + Robot.Width / 2, Robot.Top + Robot.Height / 2), 0.08 * LineWidth, vbRed
  End Select
End Sub
-----------------------------------------------------------------------------------------------------------------------
Private Sub SquareTrack_Click()
  Trackline = 1
End Sub
-----------------------------------------------------------------------------------------------------------------------
Private Sub CircleTrack_Click()
  Trackline = 2
End Sub
-----------------------------------------------------------------------------------------------------------------------
Private Sub PlusTrack_Click()
  Trackline = 3
End Sub
-----------------------------------------------------------------------------------------------------------------------
Private Sub CrossTrack_Click()
  Trackline = 4
End Sub
-----------------------------------------------------------------------------------------------------------------------
Private Sub DotTrack_Click()
  Trackline = 5
End Sub
```

นิพนธ์ต้นฉบับ

ปัญหาการติดตามเป้าหมายและการหลบหลีกสิ่งกีดขวางของหุ่นยนต์ด้วยวิธีควบคุมแบบฟัซซีอย่างง่าย

อภิวิชญ์ แก้วนพรัตน์ [1] และ ปิติเขต สู้รักษา [2]

Abstract

Kaewnopparat, A.[1] and Sooraksa, P.[2]

Tracking problem and obstacle avoidance of mobile robots using simple fuzzy control

Songklanakarin J. Sci. Technol., 2004, 26(5) :

This paper presents a solution to the tracking problem, which is one of the interesting research topics, for a robot with obstacle avoidance via simple fuzzy control. The proposed forms of membership functions are designed in light of ease of implementation for real-world application. Control parameters are considered and assigned based upon size of the robot and interaction distance between sensors and environment. To evaluate the effectiveness of the design, computer simulation using Visual Basic is carried out to validate the robot movement. The results show high possibility for implementation of the system using the proposed design.

Key words : mobile robot, robot navigation, fuzzy logic control, obstacle avoidance

[1]Department of Electrical Engineering [2]Department of Information Engineering, Faculty of Engineering, King Mongkut's Institute of Technology Ladkrabrang, Ladkrabrang, Bangkok 10520 Thailand.

[1]นักศึกษาปริญญาโท สาขาวิชาวิศวกรรมไฟฟ้า [2]Ph.D.(Information Engineering) รองศาสตราจารย์, ภาควิชาวิศวกรรมสารสนเทศ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง เขตลาดกระบัง กรุงเทพ 10520
Corresponding e-mail: s2061006@hotmail.com
รับต้นฉบับ 18 กุมภาพันธ์ 2547 รับลงพิมพ์ 21 เมษายน 2547

บทคัดย่อ

อภิวิชญ์ แก้วนพรัตน์ และ ปิติเขต สู้รักษา
ปัญหาการติดตามเป้าหมายและการหลบหลีกสิ่งกีดขวางของหุ่นยนต์
ด้วยวิธีควบคุมแบบฟัซซีอย่างง่าย
ว. สงขลานครินทร์ วทท. 2547 26(5) :

บทความนี้กล่าวถึงวิธีการแก้ปัญหาการติดตามเป้าหมายของหุ่นยนต์ในขณะที่มีสิ่งกีดขวางด้วยกระบวนการควบคุมแบบฟัซซีอย่างง่ายซึ่งเป็นปัญหาวิจัยหนึ่งที่น่าสนใจ โดยรูปแบบของฟังก์ชันความเป็นสมาชิกออกแบบให้อยู่ในรูปแบบที่ง่ายต่อการนำไปใช้จริง พารามิเตอร์ที่ใช้ในการควบคุมออกแบบโดยอาศัยขนาดของหุ่นยนต์และระยะอันตรกิริยาระหว่างตัวตรวจรับกับสิ่งแวดล้อม เพื่อตรวจสอบการออกแบบจึงจำลองการเคลื่อนที่ของหุ่นยนต์ในสถานการณ์ต่าง ๆ ด้วยคอมพิวเตอร์โดยใช้โปรแกรมวิชวลเบสิก ผลการจำลองที่ได้แสดงถึงความเป็นไปได้อย่างสูงของระบบควบคุมที่นำเสนอว่าสามารถนำไปใช้งานจริงได้

เมื่อไม่นานมานี้มีการประยุกต์ใช้การควบคุมแบบฟัซซี (Lin and George Lee, 1996; Tsoukalas and Uhrig, 1996) กับหุ่นยนต์ให้ทำหน้าที่ต่างๆ เช่น ใช้ในการสร้างแผนที่ (Tunstel and Jamshidi, 1994) ใช้ให้เคลื่อนที่ตามแหล่งกำเนิดแสงที่กำหนด (Da Silva *et al.*, 1998) ซึ่งการเคลื่อนที่ของหุ่นยนต์ในงานสนามนั้น จะพบว่าปัญหาการติดตามเป้าหมายและปัญหาการหลบสิ่งกีดขวางนั้นเป็นปัญหาที่พบเป็นประจำ และงานวิจัยที่ผ่านมานั้นพบว่าการแก้ปัญหาดังกล่าวโดยวิธีควบคุมแบบฟัซซีได้ออกแบบฟังก์ชันสมาชิกที่ยุ่งยาก (Ho Yim and Butler, 1995) หรือไม่ก็มีตัวแปรทางภาษามากเกินความจำเป็น (Martinez *et al.*, 1993; Wei Li, 1994) ดังนั้นบทความนี้จึงได้นำเสนอระบบควบคุมฟัซซีที่เข้าใจง่าย ลดความซับซ้อนลง โดยที่ระบบควบคุมประกอบด้วย ตัวแปรอินพุตเพียง 4 ตัว และตัวแปรเอาต์พุตเพียง 2 ตัว โดยมีฟังก์ชันความเป็นสมาชิกของอินพุตและสมาชิกของเอาต์พุตเป็นรูปสามเหลี่ยม ใช้ตัวตรวจวัดอัลตราโซนิคเพียง 3 ตัว ทำให้หุ่นยนต์ใช้เวลาในการประมวลผลน้อย มีการตอบสนองอย่างรวดเร็ว และเหมาะสมที่จะนำมาใช้กับการตัดสินใจของหุ่นยนต์ในการเคลื่อนที่หลบหลีกสิ่งกีดขวางไปยังเป้าหมายที่กำหนด

แบบจำลองหุ่นยนต์และการออกแบบตัวควบคุมฟัซซี

1. แบบจำลองหุ่นยนต์

หุ่นยนต์ที่ออกแบบนี้ ใช้อุปกรณ์ตัวตรวจวัด 2 ชนิด คือ ลิมิตสวิตช์ (Limit Switch) และตัวตรวจวัดอัลตราโซนิค (Ultrasonic Sensor) โดยที่ตัวตรวจวัดอัลตราโซนิค ใช้ตรวจวัดระยะทางระหว่างหุ่นยนต์กับสิ่งกีดขวาง ใช้ทั้งหมด 3 ตัว คือ ด้านหน้า ทางซ้าย และทางขวา โดยค่าที่ตรวจวัดได้นี้จะเป็นค่าที่ใช้เป็นตัวแปรอินพุตของตัวควบคุมฟัซซี ส่วนลิมิตสวิตช์ใช้ตรวจจับการชนจะใช้ 2 ตัว คือ ที่ด้านหน้าทางซ้ายหนึ่งตัวและด้านหน้าทางขวาอีกหนึ่งตัว ดังแสดงใน Figure 1 โดยจะทำหน้าที่เป็นชุดตรวจวัดสำรองเพื่อใช้ในกรณีที่หุ่นยนต์เกิดการชนกับสิ่งกีดขวางโดยใช้หลักสามัญสำนึกคือ ถ้าลิมิตสวิตช์ด้านซ้ายทำงานแล้ว ให้หุ่นยนต์ถอยหลังและหมุนไปทางขวา และถ้าลิมิตสวิตช์ด้านขวาทำงาน แล้วให้หุ่นยนต์ถอยหลังและหมุนไปทางซ้าย

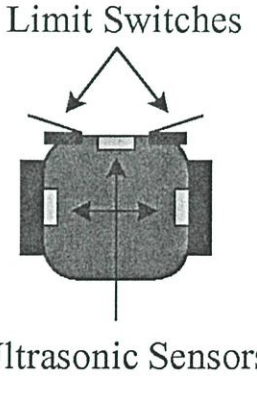


Figure 1. Installation of Ultrasonic Sensors and Limit Switches

2. การออกแบบตัวควบคุมฟัซซี

ในบทความนี้ระบบควบคุมฟัซซีใช้โครงสร้างพื้นฐานที่สำคัญ 4 ส่วน คือ ส่วนการฟัซซิฟิเคชัน (Fuzzification Unit) ส่วนกฎการควบคุมฟัซซี (Fuzzy Rules Base) หน่วยวินิจฉัยกฎ (Inference Engine) และส่วนการดีฟัซซิฟิเคชัน (Defuzzification Unit) ดังแสดงใน Figure 2

2.1 การฟัซซิฟิเคชัน

เป็นขั้นตอนการแปลงค่าตัวแปรอินพุตและค่าตัวแปรเอาต์พุตให้อยู่ในรูปค่าความเป็นสมาชิกในระบบฟัซซีเซต โดยอาศัยฟังก์ชันความเป็นสมาชิกและตัวแปรทางภาษาตัวแปรอินพุตที่ใช้ในการฟัซซิฟิเคชันมีด้วยกัน 4 ตัวแปร คือ

1) ระยะห่างระหว่างหุ่นยนต์กับสิ่งกีดขวางทางด้านซ้าย

2) ระยะห่างระหว่างหุ่นยนต์กับสิ่งกีดขวางทางด้านหน้า

3) ระยะห่างระหว่างหุ่นยนต์กับสิ่งกีดขวางทางด้านขวา

4) ทิศทางของเป้าหมาย

ซึ่งตัวแปร 3 ตัวแรกเป็นข้อมูลที่รับมาจากตัวตรวจวัดอัลตราโซนิค แสดงได้ดัง Figure 3 ส่วนตัวแปรทิศทางของเป้าหมาย หาได้จากสมการที่ 1 คือ

$$\theta = \phi(\text{robot}) - \phi(\text{target}) \qquad (1)$$

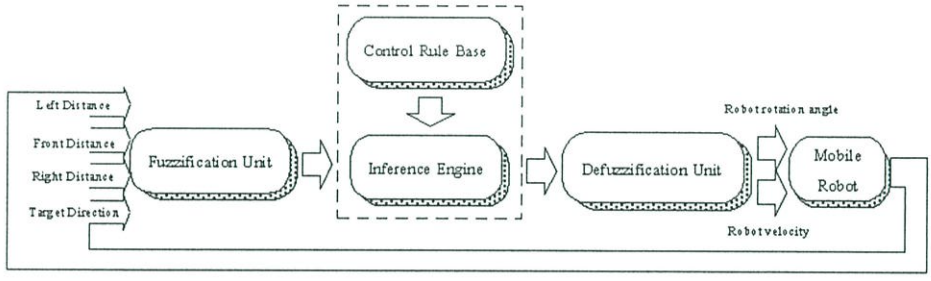


Figure 2. Structure of Fuzzy Control

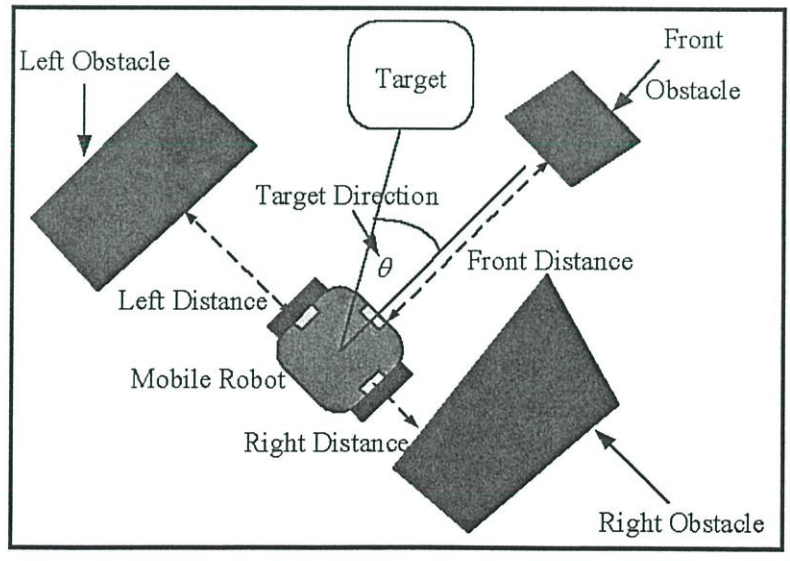


Figure 3. Acquired information for fuzzification of input variables

โดยที่ θ คือ ทิศทางของเป้าหมาย
ϕ(robot) คือ มุมของหุ่นยนต์
ϕ(target) คือ มุมของเป้าหมาย

ถ้าทิศทางเป้าหมายมีค่าเป็นลบ แสดงว่าเป้าหมายอยู่ทางด้านซ้ายของหุ่นยนต์ ถ้าทิศทางเป้าหมายมีค่าเป็นบวก แสดงว่าเป้าหมายอยู่ทางด้านขวาของหุ่นยนต์ โดยการหาทิศทางเป้าหมาย แสดงดัง Figure 4

ฟังก์ชันความเป็นสมาชิกของตัวแปรอินพุตทั้ง 4 นั้น แสดงดัง Figure 5

ตัวแปรทางภาษาของตัวแปรอินพุต มีความหมายดังนี้

- สำหรับตัวแปรระยะห่างด้านซ้าย ตัวแปรระยะห่างด้านหน้า และตัวแปรระยะห่างด้านขวา

NL: Near Left คือ เข้าใกล้สิ่งกีดขวางทางซ้าย
FL: Far Left คือ ไกลจากสิ่งกีดขวางทางซ้าย
NF: Near Front คือ เข้าใกล้สิ่งกีดขวางทางหน้า
FF: Far Front คือ ไกลจากสิ่งกีดขวางทางหน้า
NR: Near Right คือ เข้าใกล้สิ่งกีดขวางทางขวา
FR: Far Right คือ ไกลจากสิ่งกีดขวางทางขวา

- สำหรับตัวแปร ทิศทางของเป้าหมาย

B_L: Back Left คือ เป้าหมายอยู่ด้านหลังทางซ้าย
L: Left คือ เป้าหมายอยู่ด้านซ้าย
F_L: Front Left คือ เป้าหมายอยู่ด้านหน้าทางซ้าย
Z: Zero คือ เป้าหมายอยู่ด้านหน้า
F_R: Front Right คือ เป้าหมายอยู่ด้านหน้าทางขวา
R: Right คือ เป้าหมายอยู่ด้านขวา
B_R: Back Right คือ เป้าหมายอยู่ด้านหลังทางขวา

ฟังก์ชันความเป็นสมาชิกของตัวแปรเอาต์พุตของระบบควบคุมฟัซซีแสดงดัง Figure 6 ตัวแปรทางภาษาของตัวแปรเอาต์พุตมีความหมายดังนี้

- สำหรับตัวแปร ความเร็วของหุ่นยนต์

Slow คือ ให้หุ่นยนต์เคลื่อนที่ช้า
Medium คือ ให้หุ่นยนต์เคลื่อนที่ปานกลาง
Fast คือ ให้หุ่นยนต์เคลื่อนที่เร็ว

- สำหรับตัวแปร มุมการหมุนของหุ่นยนต์

NB: Negative Big คือ ให้หุ่นยนต์หมุนซ้ายมากๆ
NM: Negative Medium คือ ให้หุ่นยนต์หมุนซ้ายปานกลาง
NS: Negative Small คือ ให้หุ่นยนต์หมุนซ้ายเล็กน้อย
ZZ: Zero คือ ให้หุ่นยนต์ตรงไปข้างหน้า
PS: Positive Small คือ ให้หุ่นยนต์หมุนขวาเล็กน้อย
PM: Positive Medium คือให้หุ่นยนต์หมุนขวาปานกลาง
PB: Positive Big คือ ให้หุ่นยนต์หมุนขวามากๆ

2.2 การสร้างกฎการควบคุมแบบฟัซซี

กฎการควบคุมแบบฟัซซี ถูกกำหนดให้อยู่ในรูปแบบของกฎ "ถ้า-แล้ว" หรือ "IF-THEN" ซึ่งแสดงความสัมพันธ์ระหว่างค่าอินพุตและเอาต์พุต หลักการในการสร้างกฎการควบคุมนั้นสร้างจากพฤติกรรมของหุ่นยนต์ที่กระทำกับสิ่งกีดขวางในสภาวะแวดล้อมขณะนั้น แบ่งเป็น

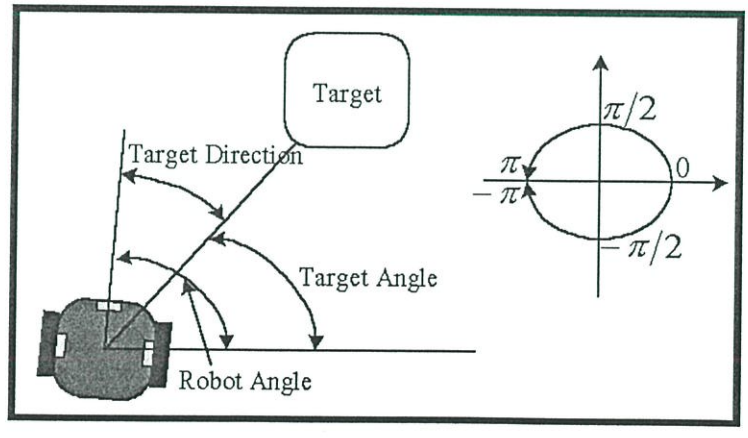


Figure 4. Calculation of target direction

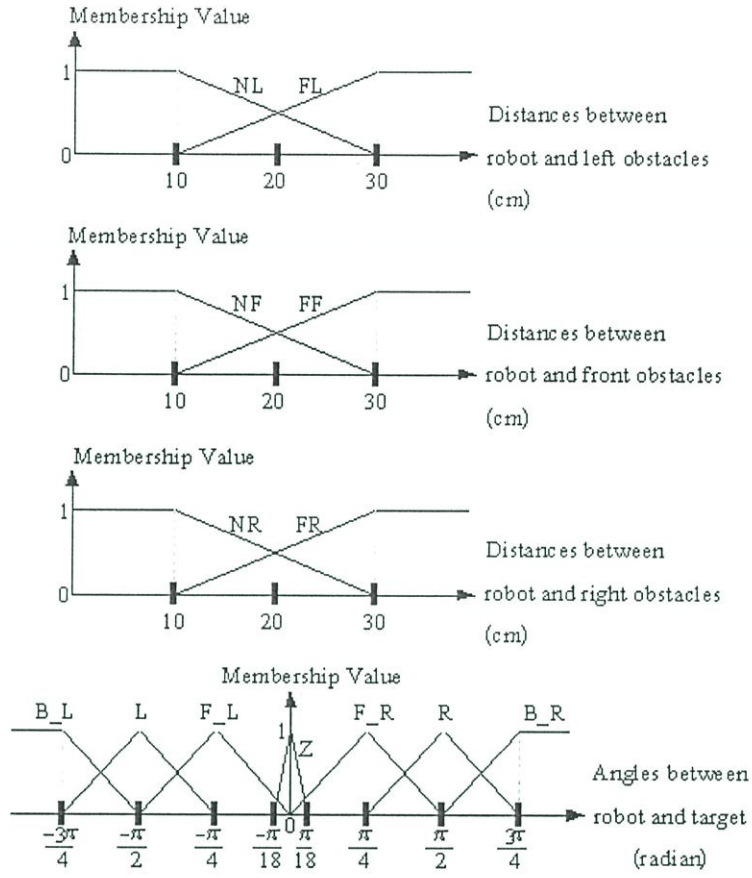


Figure 5. Membership Function of Input Variables

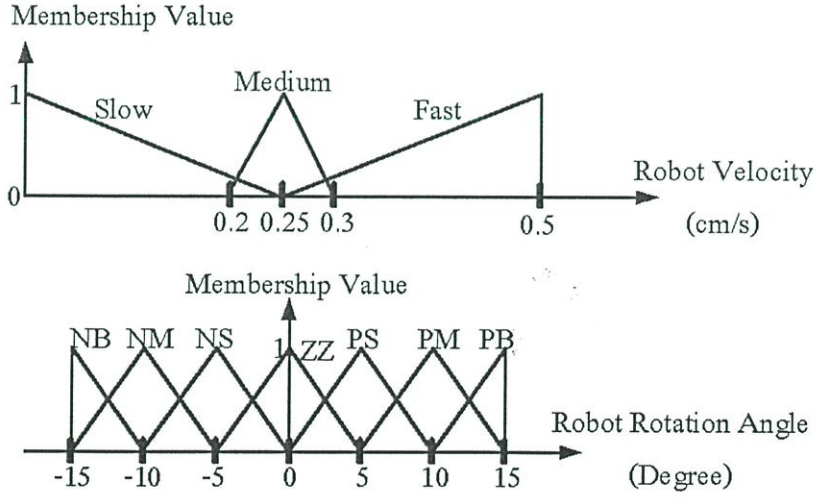


Figure 6. Membership Function of Output Variables

4 พฤติกรรม คือ

1) พฤติกรรมเคลื่อนที่หลบหลีกสิ่งกีดขวาง

เมื่อหุ่นยนต์เคลื่อนที่เข้าใกล้สิ่งกีดขวาง หุ่นยนต์ควรจะเคลื่อนที่เพื่อหลบสิ่งกีดขวางให้ได้ และลดความเร็วให้เคลื่อนที่ช้าลงเพื่อป้องกันตนเองในกรณีที่ตัวตรวจวัดทำงานไม่ทันเนื่องจากข้อจำกัดทางกายภาพ

2) พฤติกรรมเคลื่อนที่ในทางแคบๆ

เมื่อหุ่นยนต์เคลื่อนที่เข้าไปในทางแคบๆ หุ่นยนต์ควรเคลื่อนที่ตรงไปข้างหน้าเพียงอย่างเดียวโดยที่ไม่สนใจทิศทางของเป้าหมายและใช้ความเร็วในระดับปกติ เพราะหุ่นยนต์ควรรีบพาตนเองออกจากระยะประชิด

3) พฤติกรรมเคลื่อนที่ไปตามขอบ

เมื่อทิศทางของเป้าหมายอยู่ฝั่งตรงข้ามสิ่งกีดขวางและเพื่อให้หุ่นยนต์วิ่งไปหาเป้าหมายได้ หุ่นยนต์ควรจะวิ่งไปตามขอบของสิ่งกีดขวางไปเรื่อยๆ จนกระทั่งผ่านพ้นสิ่งกีดขวางนั้น แล้วจึงเคลื่อนที่ไปยังเป้าหมายต่อไป โดยหุ่นยนต์สามารถเคลื่อนที่ด้วยความเร็วปกติได้

4) พฤติกรรมเคลื่อนที่วิ่งไปหาเป้าหมาย

เมื่อไม่มีสิ่งกีดขวางระหว่างหุ่นยนต์กับเป้าหมาย หรือไม่มีสิ่งกีดขวางรอบๆ ตัวหุ่นยนต์ ในกรณีนี้หุ่นยนต์สามารถทำตามจุดประสงค์หลักคือ วิ่งเข้าหาเป้าหมายและใช้ความเร็วให้มากที่สุดเท่าที่จะทำได้

ยกตัวอย่างกฎการควบคุม ในกรณีที่หุ่นยนต์ตรวจพบสิ่งกีดขวางในระยะใกล้ทั้ง 3 ทิศทาง และเป้าหมายอยู่ด้านหน้า ดังนั้นหุ่นยนต์จะมีพฤติกรรมเคลื่อนที่หลบหลีกสิ่งกีดขวาง เขียนกฎการควบคุมได้ดังนี้ ถ้าระยะห่างด้านซ้าย = NL, ระยะห่างด้านหน้า = NF, ระยะห่างด้านขวา = NR, ทิศทางของเป้าหมาย = Z แล้วความเร็วของหุ่นยนต์ = Slow และมุมการหมุนของหุ่นยนต์ = NM

2.3 การวินิจฉัยกฎ

เป็นการนำค่าความเป็นสมาชิกที่ได้รับเข้ามาไปประมวลผลตามกฎการควบคุมของระบบควบคุมฟัซซีที่ได้ออกแบบไว้ ในที่นี้จะเลือกใช้วิธีการ Mamdani - Min (Tsoukalas and Uhrig, 1996) ซึ่งเป็นวิธีที่ใช้เวลาประมวลผลเร็วและมีการคำนวณที่ไม่ซับซ้อน

2.4 การดีฟัซซิฟิเคชัน

เป็นการแปลงผลจากค่าความเป็นสมาชิกที่ได้จากขั้นตอนของการวินิจฉัยกฎการควบคุมให้อยู่ในรูปของค่าเอาต์พุตโดยเป็นจำนวนจริงที่อยู่ในโดเมนของตัวแปรเอาต์พุต ซึ่งค่านี้เป็นค่าที่นำไปใช้ในการควบคุมหุ่นยนต์นั่นเอง

เทคนิคการดีฟัซซิฟิเคชันมีหลายวิธีด้วยกัน สำหรับระบบควบคุมแบบฟัซซีในบทความนี้จะเลือกใช้วิธีหาค่าศูนย์กลาง (Tsoukalas and Uhrig, 1996) ซึ่งวิธีนี้เป็นที่นิยมใช้มากในกระบวนการควบคุม เนื่องจากใช้ข้อมูลของค่าความเป็นสมาชิกของทุกกฎที่เป็นจริงทำให้ค่าที่ได้มีความน่าเชื่อถือและเหมาะสมกว่าวิธีอื่นๆ สามารถคำนวณความเร็วของหุ่นยนต์ จากการดีฟัซซิฟิเคชันได้จากสมการที่ 2

$$v^{\bullet}_{Robot} = \frac{\sum_{j=1}^{N} \mu(v_j) \cdot v_j}{\sum_{j=1}^{N} \mu(v_j)} \qquad (2)$$

โดยที่ $v^{\bullet}_{Robot}$ คือ ความเร็วของหุ่นยนต์จากการดีฟัซซิฟิเคชัน

v_j คือ ค่าเอาต์พุตของตัวแปรความเร็วของหุ่นยนต์ตัวที่ j

$\mu(v_j)$ คือ ค่าความเป็นสมาชิกของตัวแปร v_j ตัวที่ j

N คือ จำนวนข้อมูลที่สุ่มตัวอย่างทั้งหมด

ส่วนการคำนวณมุมการหมุนของหุ่นยนต์ สามารถหาได้จากการดีฟัซซิฟิเคชัน ดังสมการที่ 3

$$\theta^{\bullet}_{Robot} = \frac{\sum_{j=1}^{N} \mu(\theta_j) \cdot \theta_j}{\sum_{j=1}^{N} \mu(\theta_j)} \qquad (3)$$

โดยที่ $\theta^{\bullet}_{Robot}$ คือ มุมการหมุนจากการดีฟัซซิฟิเคชัน

θ_j คือ ค่าเอาต์พุตของตัวแปรมุมการหมุนตัวที่ j

$\mu(\theta_j)$ คือ ค่าความเป็นสมาชิกของ θ_j ตัวที่ j

สำหรับการทดสอบการออกแบบตามหลักการของแบบจำลองหุ่นยนต์และการออกแบบตัวควบคุมฟัซซี จะนำเสนอในหัวข้อการทดสอบด้วยการจำลองผล

การทดสอบด้วยการจำลองผล

โครงสร้างภาพรวมของขั้นตอนการทำงานของโปรแกรมจำลองการเคลื่อนที่ของหุ่นยนต์แสดงดัง Figure 7

ในการทดสอบการทำงานได้จำลองตัวหุ่นยนต์ขนาดกว้าง 15 ซม. ยาว 15 ซม. ภายในพื้นที่จำกัด ขนาดกว้าง 230 ซม. ยาว 150 ซม. ส่วนขั้นตอนการตรวจสอบว่าหุ่นยนต์เคลื่อนที่ถึงเป้าหมายที่กำหนดให้แล้วหรือไม่ สามารถตรวจสอบได้จากการที่หุ่นยนต์เคลื่อนที่ไปเข้าใกล้เป้าหมายในระยะที่ยอมรับได้ ดังสมการที่ 4

$$\sqrt{(X_t - X_r)^2 + (Y_t - Y_r)^2} < \varepsilon \qquad (4)$$

โดยที่ ε คือ ระยะที่ยอมรับได้

(X_t, Y_t) คือ พิกัดของเป้าหมาย

(X_r, Y_r) คือ พิกัดของหุ่นยนต์

ในขั้นตอนการเคลื่อนตัวหุ่นยนต์จะนำค่าความเร็วและมุมการหมุนของหุ่นยนต์ที่ได้จากการรดีฟัซซิฟิเคชันมากำหนดตำแหน่งที่หุ่นยนต์จะเคลื่อนที่ไปและมุมของหุ่นยนต์ใหม่ ดังสมการที่ 5

$$\begin{aligned} X(t+T_s) &= X(t) + v_x^* T_s \\ Y(t+T_s) &= Y(t) + v_y^* T_s \\ \phi(t+T_s) &= \phi(t) + \theta_{Robot}^* \end{aligned} \qquad (5)$$

โดยที่ T_s คือ อัตราการตรวจวัดของเซนเซอร์ มีค่า 0.1 วินาที

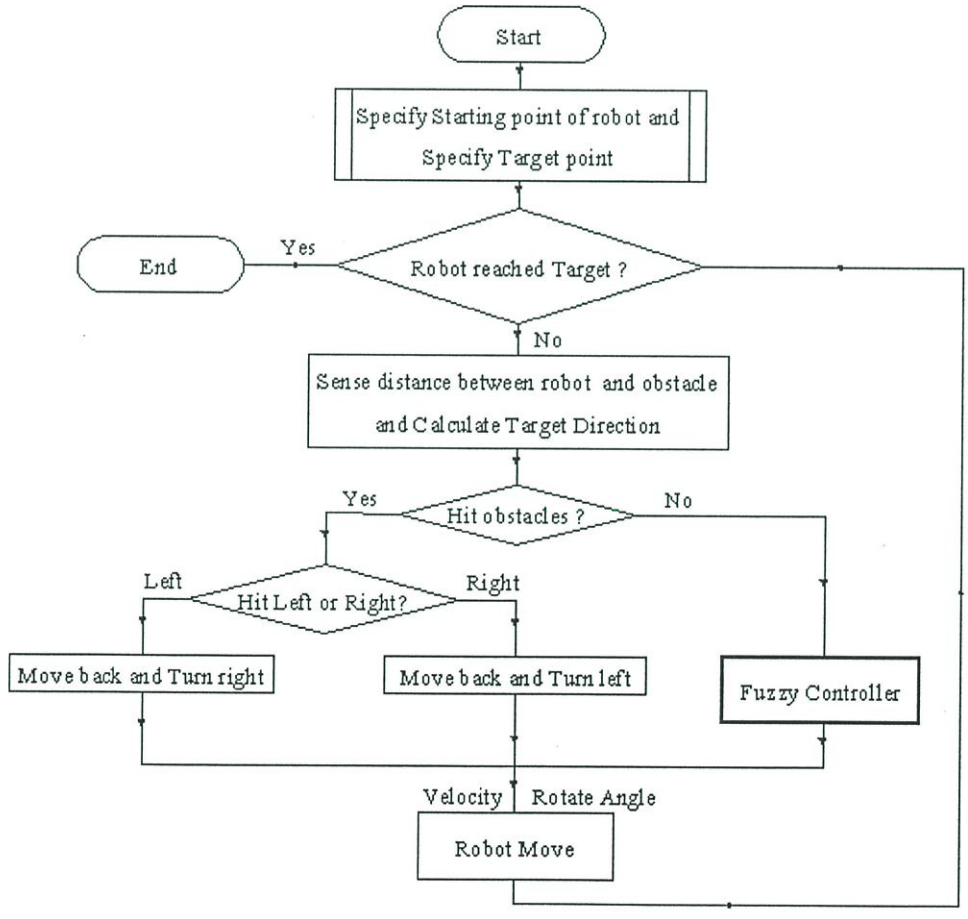


Figure 7. Block diagram of simulation program for robot navigation

v_x^* คือ ความเร็วหุ่นยนต์ในแกน X หรือ $v_{Robot}^* \cos(\phi(t))$

v_y^* คือ ความเร็วหุ่นยนต์ในแกน Y หรือ $v_{Robot}^* \sin(\phi(t))$

$\phi(t)$ คือ มุมของหุ่นยนต์ก่อนหมุน

$\phi(t+T_s)$ คือ มุมของหุ่นยนต์หลังจากหมุนแล้ว

$(X(t),Y(t))$ คือ พิกัดของหุ่นยนต์ก่อนเคลื่อนที่

$(X(t+T_s),Y(t+T_s))$ คือ พิกัดที่หุ่นยนต์จะเคลื่อนที่ไป

1. เป้าหมายอยู่ในสิ่งกีดขวางรูปตัวยูโดยทำการเปลี่ยนตำแหน่งเริ่มต้นของหุ่นยนต์

ใน Figure 8 และ Table 1 แสดงให้เห็นว่าเส้นทางการเคลื่อนที่ในเส้นทางที่ 1 และ 2 นั้น หุ่นยนต์สามารถเคลื่อนที่ไปยังเป้าหมายโดยใช้ระยะทางและเวลาที่น่าพอใจ แต่สำหรับเส้นทางที่ 3 และ 4 จะพบข้อเสียของตัวควบคุมฟัซซีที่ได้นำเสนอนี้คือ เมื่อหุ่นยนต์ต้องหลบหลีกสิ่งกีดขวางโดยมีเป้าหมายอยู่ตรงหน้า จากการออกแบบกฎการควบคุมได้กำหนดให้หุ่นยนต์หลบไปทางซ้ายเสมอ ทำให้ต้องเคลื่อนที่โดยใช้ระยะทางและเวลาที่มากกว่าที่ควรจะเป็น

2. เริ่มต้นจากจุดเดียวกันไปยังเป้าหมายที่ต่างกันในสิ่งกีดขวางวางกระจัดกระจายอยู่

ในเส้นทางที่ 1, 3 และ 4 หุ่นยนต์สามารถเคลื่อนที่ไปยังเป้าหมายโดยใช้ระยะทางและเวลาที่น่าพอใจ แต่สำหรับเส้นทางที่ 2 จะพบข้อเสียเช่นเดียวกับการจำลองหัวข้อที่ก่อนหน้านี้ ดังใน Figure 9 และ Table 2

3. สิ่งกีดขวางวางอย่างสลับซับซ้อนคล้ายเขาวงกต

เพื่อทดสอบประสิทธิภาพในการทำงานของหุ่นยนต์ จึงจำลองสิ่งแวดล้อมที่มีสิ่งกีดขวางวางอย่างสลับซับซ้อนซึ่งมีลักษณะคล้ายกับเขาวงกต จากเส้นทางการเคลื่อนที่ของหุ่นยนต์ใน Figure 10 พบว่าหุ่นยนต์ตัดสินใจผิดพลาดทำให้เคลื่อนที่วนกลับไปยังจุดเริ่มต้นอยู่หนึ่งรอบ แต่รอบที่สองก็สามารถตัดสินใจได้อย่างถูกต้องเนื่องจากข้อมูลที่หุ่นยนต์ได้รับเปลี่ยนไปทำให้การตัดสินใจเปลี่ยนไปด้วย

4. เป้าหมายอยู่หลังสิ่งกีดขวางรูปตัวยู

เป็นการจำลองเพื่อทดสอบว่าหุ่นยนต์สามารถเคลื่อนที่ออกมาจากสิ่งกีดขวางรูปตัวยูโดยที่ไม่เคลื่อนที่วนรอบอยู่ภายในได้หรือไม่ ผลการทดสอบดังแสดงใน Figure 11 หุ่นยนต์จะใช้พฤติกรรมเคลื่อนที่ไปตามขอบเพื่อที่จะเคลื่อนที่ออกมาจากสิ่งกีดขวางรูปตัวยู หลังจากที่ออกมาภายนอกสิ่งกีดขวางแล้วก็จะเคลื่อนไปยังตำแหน่งเป้าหมาย

Table 1. Distance and time of robot from Figure 8

Start Point	Path	Distance (Cm.)	Time (S.)
1	1 □	234.4	73.8
2	2 ○	109.3	27.6
3	3 +	417	128
4	4 ×	305.4	91.1

Table 2. Distance and time of robot from Figure 9

Start Point	Path	Distance (Cm.)	Time (S.)
1	1 □	234.5	65.7
2	2 ○	336.8	106.8
3	3 +	167.6	52.6
4	4 ×	143.9	41.6

Table 3. Distance and time of robot from Figure 15

Path	Distance (Cm.)	Time (S.)
Simple Fuzzy Control (Red Path)	246.5	72
Complicated Fuzzy Control (Blue Path)	238.4	79

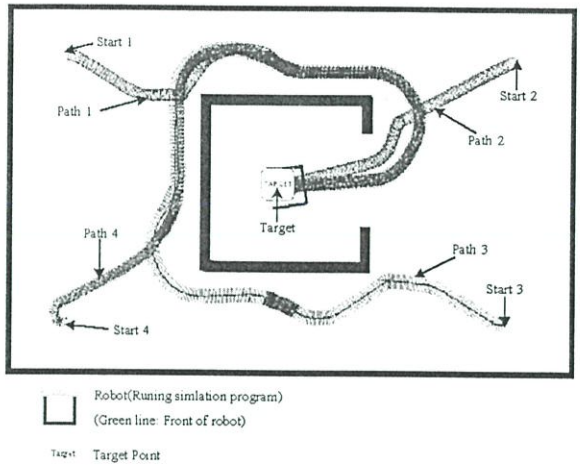


Figure 8. Robot tracking line from several starting points to reach the same target inside U-shaped Obstacle

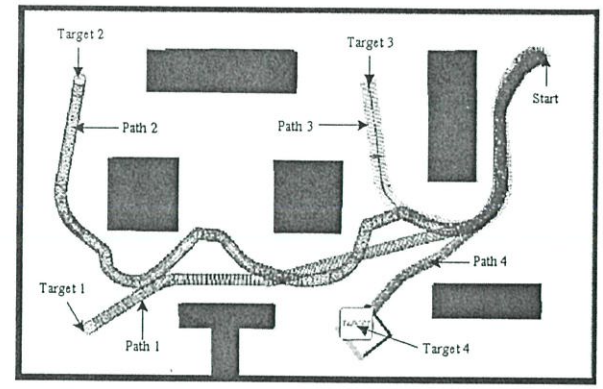


Figure 9. Robot tracking line from same start point to reach the several target points in cluttered environment

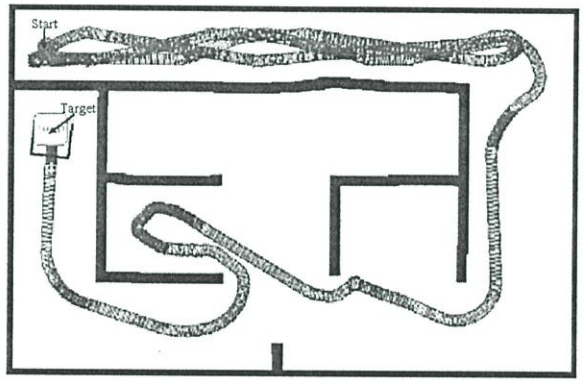


Figure 10. Robot motion in complicated environment like the maze

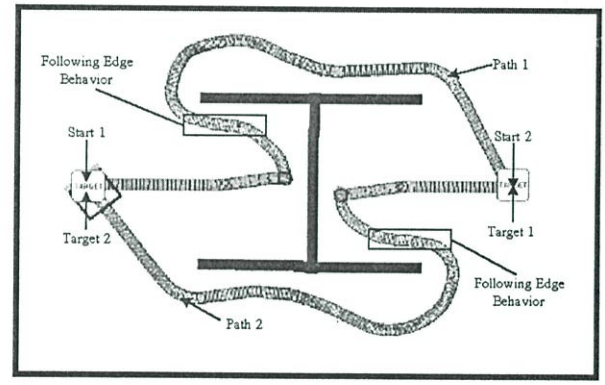


Figure 11. Robot escape from U-shaped obstacle and target position is located at the backside

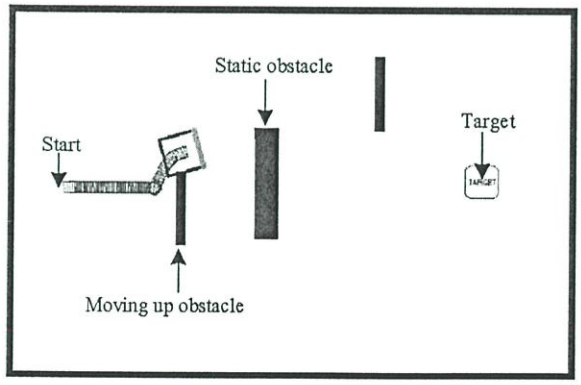


Figure 12.1 Robot avoiding the first obstacle (moving up)

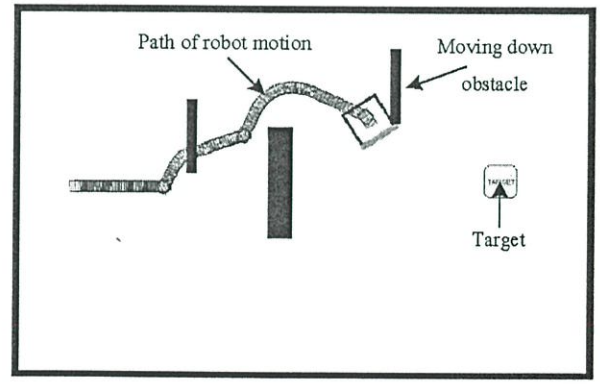


Figure 12.2 Robot avoiding the second obstacle (moving down)

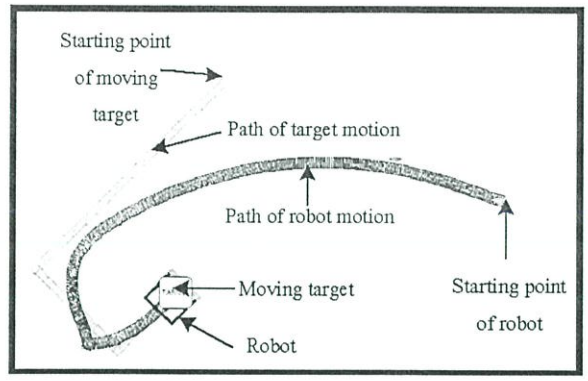


Figure 13. Robot catching the moving target

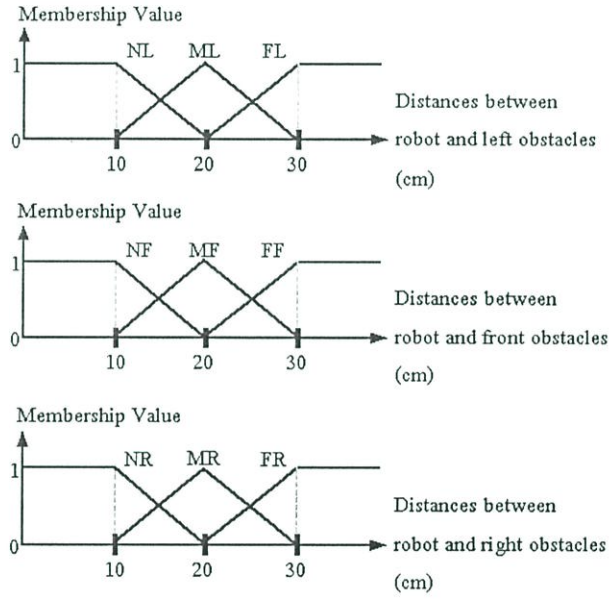


Figure 14. Membership function of input variables for complicated fuzzy control

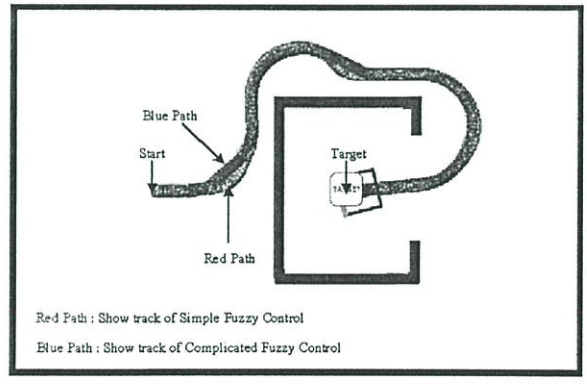


Figure 15. Compare robot tracking line of simple and complicated fuzzy control

ต่อไป

5. หลบหลีกสิ่งกีดขวางที่เคลื่อนที่ได้

เป็นการจำลองให้มีสิ่งกีดขวางที่เคลื่อนที่ได้ 2 ตัว เคลื่อนที่ขึ้นลงอยู่ระหว่างจุดเริ่มต้นของหุ่นยนต์กับเป้าหมาย โดยที่สิ่งกีดขวางทางซ้ายกำลังเคลื่อนที่ขึ้น ส่วนสิ่งกีดขวางทางขวากำลังเคลื่อนที่ลง ทั้งสองเคลื่อนที่ด้วยความเร็ว 2.5 เซนติเมตรต่อวินาทีเท่ากัน เส้นทางการเคลื่อนที่ของหุ่นยนต์แสดงดัง Figure 12.1 และ 12.2

6. เป้าหมายสามารถเคลื่อนที่ได้

เป็นการจำลองให้หุ่นยนต์เคลื่อนที่ไปยังเป้าหมายที่สามารถเคลื่อนที่ได้ โดยเป้าหมายเคลื่อนที่ด้วยความเร็ว 2 เซนติเมตรต่อวินาที เส้นทางการเคลื่อนที่ของหุ่นยนต์แสดงดัง Figure 13

การจำลองในข้อ 5 และ 6 ทำให้เห็นว่า ระบบควบคุมฟัซซีสามารถนำมาใช้กับหุ่นยนต์ให้เคลื่อนที่ในสภาพแวดล้อมที่มีการเปลี่ยนแปลงได้ เนื่องจากการทำงานของหุ่นยนต์จะรับข้อมูลสภาพแวดล้อม ณ เวลานั้นจึงทำให้การตัดสินใจเป็นไปในลักษณะเรียลไทม์

7. เปรียบเทียบวิธีควบคุมฟัซซีแบบง่ายกับแบบซับซ้อน

สำหรับวิธีการควบคุมฟัซซีแบบซับซ้อนนั้นได้ทำการจำลองขึ้นโดยกำหนดให้ฟังก์ชันความเป็นสมาชิกของแต่ละตัวแปรอินพุตระยะห่างกับสิ่งกีดขวาง ทั้งด้านหน้าทางซ้ายและทางขวามีความซับซ้อนขึ้น กล่าวคือ แต่ละตัวแปรอินพุตมีเทอมเซตเพิ่มขึ้นเป็น 3 เทอมเซต ดังแสดงใน Figure 14

การเปรียบเทียบวิธีควบคุมฟัซซีอย่างง่ายที่ได้ทำการออกแบบไว้ในบทความนี้กับวิธีการควบคุมฟัซซีแบบซับซ้อน โดยการจำลองให้หุ่นยนต์เคลื่อนที่ในสภาพแวดล้อมที่เหมือนกันและมีจุดเริ่มต้นและตำแหน่งเป้าหมายเดียวกัน

จากผลการจำลองใน Figure 15 พบว่าวิธีการควบคุมฟัซซีแบบง่ายใช้ระยะทางในการเคลื่อนที่ไปยังเป้าหมายมากกว่าแต่จะใช้ระยะเวลาที่น้อยกว่าดังแสดงใน Table 3 ทั้งนี้เนื่องมาจากการเพิ่มเทอมเซตของฟังก์ชันสมาชิกของตัวแปรอินพุตของวิธีการควบคุมฟัซซีแบบซับซ้อนทำให้หุ่นยนต์สามารถเคลื่อนที่หลบสิ่งกีดขวางได้ก่อนที่จะเข้าไปใกล้กับสิ่งกีดขวาง ส่วนเหตุที่ทำให้วิธีการควบคุมฟัซซีแบบซับซ้อนใช้เวลาในการเคลื่อนที่มากกว่านั้น เนื่องจากหุ่นยนต์ต้องทำการเปลี่ยนแปลงมุมการหมุนมากขึ้นนั่นเอง โดยเมื่อดูจากกราฟที่แสดงใน Figure 16 จะเห็นว่าวิธีการควบคุมฟัซซีแบบง่ายมีการเปลี่ยนแปลงมุมการหมุนน้อยกว่าวิธีการควบคุมฟัซซีแบบซับซ้อน

สรุปผล

จากผลการจำลองจะเห็นได้ว่าหุ่นยนต์สามารถเคลื่อนที่หลบหลีกสิ่งกีดขวางและเคลื่อนที่ไปถึงเป้าหมายที่กำหนดไว้ได้ในกรณีศึกษาต่างๆ ไม่ว่าสิ่งกีดขวางจะอยู่กับที่หรือสิ่งกีดขวางเคลื่อนที่ได้ แม้กระทั่งเมื่อเป้าหมายเคลื่อนที่ได้ ดังนั้นจึงสามารถกล่าวได้ว่าระบบควบคุมฟัซซีอย่างง่ายที่นำเสนอในบทความนี้ที่นำมาใช้เพื่อช่วยในการ

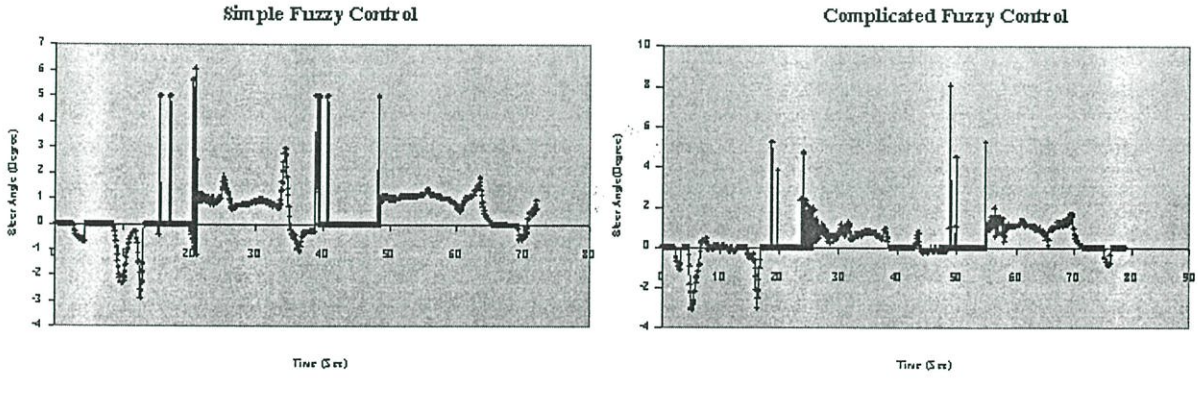


Figure 16. Comparision of robot steer angle of simple and complicated fuzzy control

ตัดสินใจของหุ่นยนต์น่าจะสามารถนำไปประยุกต์ใช้งานกับหุ่นยนต์ที่สร้างขึ้นมาตามแบบจำลองที่ได้นำเสนอข้างต้นได้

เอกสารอ้างอิง

Da Silva, I.N., Gomide, F.A.C. and Do Amaral, W.C. 1998. Navigation of Mobile Robots using Fuzzy Logic Controllers, 5th International Workshop on Advanced Motion Control, Coimbra, Portugal, June 29 - July 1, 1998: 346-349.

Ho Yim and Butler,A.C. 1995. Motion Planning Using Fuzzy Logic Control With Minimum Sensors, Proc. IEEE International Symp. on Intelligent Control, Monterey, CA USA, Aug 27-29, 1995: 558-564.

Lin, C.T. and George Lee, C.S. 1996. Neural Fuzzy Systems. Prentice Hall, Inc., New Jersey.

Martinez, A., Tunstel, E. and Jamshidi, M. 1993. Fuzzy Logic Based Collision Avoidance for a Mobile Robot, 3rd Intl. Conf. on Industrial Fuzzy Control and Intelligent Systems, Houston, TX USA, Dec. 1-3, 1993: 66-69.

Tsoukalas, L.H. and Uhrig, R.E. 1996. Fuzzy And Neural Approaches In Engineering. John Wiley & Sons, Inc., New York.

Tunstel, E. and Jamshidi, M. 1994. Fuzzy logic and Behavior Control Strategy for Autonomous Mobile Robot Mapping, Proc. IEEE Conf. on Fuzzy Systems, Orlando, FL USA, June 26-29, 1994: 514-517.

Wei Li. 1994. Perception-Action Behavior Control of a Mobile Robot in Uncertain Environments Using Fuzzy Logic, Proc. IEEE/RSI/GI Intelligent Robots and Systems, Munich, Germany, Sept 12-16, 1994: 439-446.

ประวัติผู้เขียน

ชื่อ-นามสกุล	อภิวิชญ์ แก้วนพรัตน์
วัน เดือน ปีเกิด	3 สิงหาคม 2521 ที่ อ.หาดใหญ่ จ.สงขลา
ที่อยู่	201 ถ.นิพัทธ์สงเคราะห์ 2 อ.หาดใหญ่ จ.สงขลา 90110
	โทร. 0-7421-1431 , 0-9777-0175
ประวัติการศึกษา	2542 วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมไฟฟ้า - โทรคมนาคม มหาวิทยาลัยสงขลานครินทร์
ความชำนาญเฉพาะด้าน	1.) ระบบโทรคมนาคม
	2.) การออกแบบระบบควบคุมในอุตสาหกรรม
	3.) การเขียนโปรแกรมคอมพิวเตอร์