ใบรับรองวิทยานิพนธ์

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์

วิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์)

ปริญญา

วิศวกรรมคอมพิวเตอร์ วิศวกรรมคอมพิวเตอร์

สาขา **ภาควิชา**

เรื่อง การประยุกต์ใช้ระบบนิวโรฟัซซี่กับตัวควบคุมผ่านเน็ตเวิร์คไร้สายสำหรับหุ่นยนต์เคลื่อนที่

A Neuro-Fuzzy Wireless Network-Based Controller for Mobile Robot Control

นามผู้วิจัย นายจิรัฏฐ์ ศรีสบาย

ได้พิจารณาเห็นชอบโดย

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก .....

(ผู้ช่วยศาสตราจารย์ยอดเยี่ยม ทิพย์สุวรรณ์, Ph.D.)

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม .....

(อาจารย์ชัยพร ใจแก้ว, Ph.D.)

หัวหน้าภาควิชา .....

(ผู้ช่วยศาสตราจารย์เขมะฑัต วิภาตะวนิช, Ph.D.)

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์รับรองแล้ว

.....

(รองศาสตราจารย์กัญจนา ธีระกุล, Ph.D)

คณบดีบัณฑิตวิทยาลัย

วันที่ ..... **เดือน** ..... **พ.ศ.** .....

วิทยานิพนธ์

เรื่อง

การประยุกต์ใช้ระบบนิวโรฟัซซี่กับตัวควบคุมผ่านเน็ตเวิร์คไร้สายสำหรับหุ่นยนต์เคลื่อนที่

A Neuro-Fuzzy Wireless Network-Based Controller for Mobile Robot Control

โดย

นายจิรัฏฐ์ ศรีสบาย

เสนอ

บัณฑิตวิทยาลัย มหาวิทยาลัยเกษตรศาสตร์
เพื่อความสมบูรณ์แห่งปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์)
พ.ศ. 2551

จิรัฏฐ์ ศรีสบาย 2551: การประยุกต์ใช้ระบบนิวโรฟัซซี่กับตัวควบคุมผ่านเน็ตเวิร์คไร้สายสำหรับหุ่นยนต์เคลื่อนที่ ปริญญาวิศวกรรมศาสตรมหาบัณฑิต (วิศวกรรมคอมพิวเตอร์) สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก: ผู้ช่วยศาสตราจารย์ยอดเยี่ยม ทิพย์สุวรรณ์, Ph.D. 97 หน้า

การควบคุมผ่านเน็ตเวิร์คหรือ Network-Based Control (NBC) ได้ถูกนำไปประยุกต์ใช้อย่างแพร่หลายแทนการควบคุมแบบเดิม ยกตัวอย่างเช่น การควบคุมอุปกรณ์ระยะไกลผ่านเน็ตเวิร์ค การตรวจสอบการทำงานของอุปกรณ์ผ่านเน็ตเวิร์คเป็นต้น แต่ปัญหาที่สำคัญของระบบควบคุมผ่านเน็ตเวิร์คก็คือดีเลย์ (delay) ที่เกิดขึ้นจากระบบเน็ตเวิร์ค ซึ่งทำให้ระบบเสียเสถียรภาพและระบบมีประสิทธิภาพต่ำลง โดยลักษณะของดีเลย์ที่เกิดขึ้นมีลักษณะกำกวมและไม่สามารถคาดคะเนได้

งานวิจัยนี้ได้นำเสนอวิธีการแก้ปัญหาโดยใช้ระบบ neuro-fuzzy gain scheduling เพื่อเพิ่มประสิทธิภาพให้กับอัลกอริทึม quadratic curve ซึ่งใช้ในตัวควบคุมให้หุ่นยนต์เดินตามเส้นทาง โดยได้มีการประยุกต์ใช้ self–adaptive neuro-fuzzy Inference system (SANFIS) เพื่อช่วยในการจัดกลุ่มสภาวะของเน็ตเวิร์คและใช้ gain scheduler ในการปรับค่าพารามิเตอร์ในตัวควบคุมให้เหมาะสมกับสภาวะของเน็ตเวิร์ค ณ ตอนนั้น โดยระบบทั้งหมดถูกทดสอบกับการเดินตามเส้นทางของหุ่นยนต์ผ่านเน็ตเวิร์คไร้สายซึ่งทำการทดสอบกับทั้งระบบที่จำลองควบคุมหุ่นยนต์ผ่านเน็ตเวิร์ค (simulation) และระบบที่ควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คโดยใช้ดีเลย์จริงซึ่งในการทดสอบได้มีการเปรียบเทียบระหว่างการควบคุมที่ใช้และไม่ใช้ neuro-fuzzy gain scheduling

การทดสอบทั้งระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คจำลองและระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คจริงได้พบว่าการควบคุมที่ประยุกต์ใช้ neuro-fuzzy gain scheduling ทำให้ประสิทธิภาพในระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คดีขึ้น โดยหุ่นยนต์สามารถเดินตามเส้นทางได้แม่นยำขึ้นและเร็วขึ้น

_____ _____ ___/___/___

ลายมือชื่อนิสิต ลายมือชื่ออาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

Jirat Srisabye 2008: A Neuro-Fuzzy Wireless Network-Based Controller for Mobile Robot Control. Master of Engineering (Computer Engineering), Major Field: Computer Engineering, Department of Computer Engineering. Thesis Advisor: Associate Professor Yodyium Tipsuwan, Ph.D. 97 pages.

Network-Based Control (NBC) systems have been widely applied to increase the capability of tradition control systems such as remote control and monitoring of control devices via a network. However, a significant problem that degrades performance of a NBC system is the network-induced delay. In general, network delays are ambiguous and difficult to estimate.

In this research, the neuro-fuzzy NBC gain scheduling scheme is proposed to improve the performance of a quadratic curve controller, which is a path tracking controller for a mobile robot. A SANFIS (Self-Adaptive Neuro-Fuzzy Inference System) is utilized to classify a current network condition in order to handle ambiguity in network behaviors. The SANFIS periodically selects optimal gains including a α gain and a β gain in a quadratic curve controller with respect to the network condition. To illustrate this proposed approach, we applied a mobile robot path tracking control problem over a wireless IP network in a simulation and an experiment and compared the performance of our neuro-fuzzy NBC gain scheduling approach with a traditional approach.

Simulation results and experimental results show that the mobile robot with neuro-fuzzy NBC gain scheduling provides significantly better NBC system performances than the traditional approach. The robot can track the path faster and more accurate.

_____ _____ ___ / ___ / ___

Student's signature Thesis Advisor's signature

กิตติกรรมประกาศ

ข้าพเจ้าขอกราบขอบพระคุณผู้ช่วยศาสตราจารย์ ดร.ยอดเยี่ยม ทิพย์สุวรรณ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก ผู้ให้แนวทางในการวิจัยและข้อเสนอแนะที่เป็นประโยชน์ ตลอดจนให้คำปรึกษาแนะนำและแก้ไขข้อบกพร่องต่างๆในวิทยาพนธ์นี้ และขอกราบขอบพระคุณอาจารย์ ดร.ชัยพร ใจแก้ว อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วมและศาสตราจารย์ ดร.ประภาส จงสถิตย์วัฒนา อาจารย์ผู้แทนบัณฑิตวิทยาลัยที่กรุณาให้ข้อเสนอแนะที่มีประโยชน์เพื่อให้วิทยานิพนธ์นี้สมบูรณ์ยิ่งขึ้น

นอกจากนี้ขอขอบพระคุณ ดร.สุวัชชัย กมลสันติโรจน์ และคุณณภัทร พากเพียรที่ให้คำแนะนำและสนับสนุนให้วิทยานิพนธ์นี้สำเร็จลุล่วงไปด้วยดี

จิรัฏฐ์ ศรีสบาย

ตุลาคม 2551

สารบัญ

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

การประยุกต์ใช้ระบบนิวโรฟัซซี่กับตัวควบคุมผ่านเน็ตเวิร์คไร้สายสำหรับหุ่นยนต์เคลื่อนที่

A Neuro-Fuzzy Wireless Network-Based Controller for Mobile Robot Control

คำนำ

เทคโนโลยีทางด้านเน็ตเวิร์คได้เข้ามามีบทบาทมากขึ้นในชีวิตการทำงานปัจจุบัน ทำให้สามารถประหยัดค่าใช้จ่ายในการเดินสายและทำให้มีความคล่องตัวในการใช้งานสูงขึ้น และเทคโนโลยีทางด้านนี้ยังถูกนำไปประยุกต์ใช้เกี่ยวกับงานทางด้านการควบคุมผ่านเน็ตเวิร์ค ยกตัวอย่างเช่น การควบคุมหุ่นยนต์จากระยะไกล เป็นต้น แต่เนื่องจากระบบควบคุมผ่านเน็ตเวิร์คต้องใช้อินเตอร์เน็ต (Internet) เป็นสื่อกลางในการเชื่อมต่อซึ่งมีปัจจัยหลายอย่างที่ทำให้เกิดผลกระทบโดยตรงกับประสิทธิภาพและเสถียรภาพของระบบควบคุมผ่านเน็ตเวิร์ค ยกตัวอย่างเช่น ดีเลย์ (delay), ความไม่ต่อเนื่องของข้อมูล (jitter), และการสูญหายของข้อมูล (packet loss) เป็นต้น

งานวิจัยนี้จึงได้นำเสนอเทคนิคใหม่ที่ใช้ neuro-fuzzy มาประยุกต์เพื่อเพิ่มประสิทธิภาพตัวควบคุม (controller) ในการควบคุมผ่านเน็ตเวิร์ค ซึ่ง neuro-fuzzy เป็นการประยุกต์รวมเอาข้อดีของทั้ง fuzzy logic และ neural network เข้ามารวมไว้ด้วยกัน โดยที่ fuzzy logic (Zadeh, 1973) นำการใช้เหตุและผลของมนุษย์มาจัดการความกำกวมของดีเลย์ที่เกิดขึ้นและ neural network ใช้เพื่อเพิ่มประสิทธิภาพของระบบด้วยการเรียนรู้ส่วนของระบบที่ไม่ทราบถึงการทำงานที่แน่นอนจากตัวอย่างข้อมูล ยกตัวอย่างเช่น ANFIS (Jang, 1993), FALCON (Lin and Lee, 1991) และ SANFIS (Wang and Lee, 2002) เป็นต้น นอกจากนี้แล้ว neuro-fuzzy ได้ถูกนำไปประยุกต์ในงานวิจัยหลายด้าน ยกตัวอย่างเช่น งานด้านการจัดแบ่งประเภท (classification) (Wang and Lee, 2002), งานทางด้านการควบคุม (Lin and Lee, 1991) เป็นต้น

โดยในงานวิจัยนี้ได้นำเสนอ neuro-fuzzy gain scheduling เพื่อเพิ่มประสิทธิภาพการทำงานให้กับอัลกอริทึม quadratic curve โดยการประยุกต์ใช้ SANFIS (self-adaptive neuro-fuzzy inference system) เป็นวิธีการสำคัญในการจำแนกเงื่อนไขสภาวะเน็ตเวิร์คที่เกิดขึ้นให้กับ gain scheduler ซึ่งเป็นกลไกในวิธีที่นำเสนอเพื่อใช้ในการตัดสินเลือกค่าพารามิเตอร์ alpha (α) และ beta (β) ที่เหมาะสมให้กับอัลกอริทึม quadratic curve เพื่อทำให้ระบบควบคุมผ่านเน็ตเวิร์คทำงาน

ได้อย่างมีประสิทธิภาพสูงสุด โดยระบบทั้งหมดจะถูกทดสอบกับหุ่นยนต์เคลื่อนที่ (mobile robot) เพื่อแสดงประสิทธิภาพและแสดงการทำงานของระบบควบคุมผ่านเน็ตเวิร์ค

วัตถุประสงค์

เพื่อเสนอวิธีการควบคุมผ่านเน็ตเวิร์คไร้สายให้มีประสิทธิภาพที่ดีขึ้นกว่าวิธีเดิมและสามารถนำมาใช้งานกับระบบควบคุมหุ่นยนต์เคลื่อนที่จริงได้

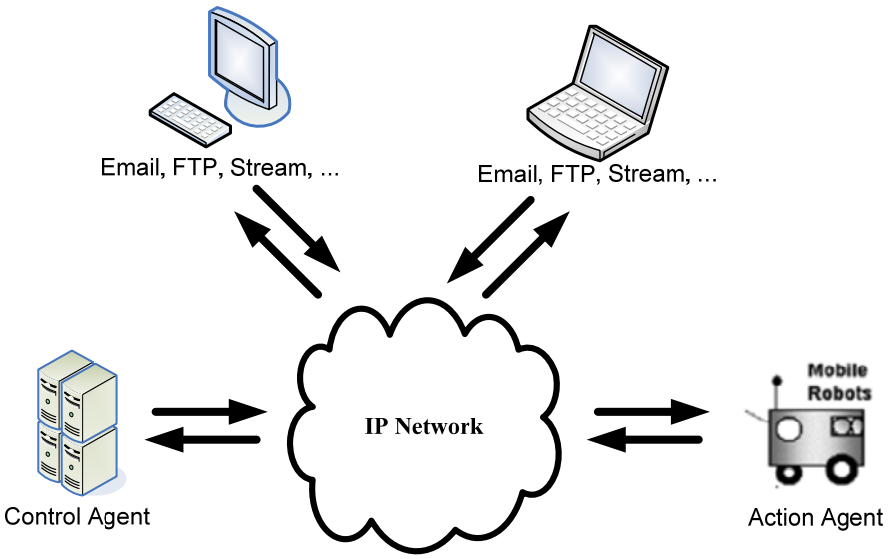
การตรวจเอกสาร

ในหัวข้อนี้จะกล่าวถึงความรู้พื้นฐานและงานวิจัยที่เกี่ยวข้อง ซึ่งประกอบไปด้วย

1) โครงสร้างของระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์ค
2) ผลกระทบของดีเลย์ที่มีต่อประสิทธิภาพของระบบ
3) งานวิจัยด้านระบบควบคุมผ่านเน็ตเวิร์คที่เกี่ยวข้อง
4) การออกแบบตัวหุ่นยนต์
5) รูปแบบทางคณิตศาสตร์ของหุ่นยนต์
6) อัลกอริทึมในการเดินตามเส้นทางของหุ่นยนต์
7) Self-Adaptive Neuro-Fuzzy Inference Systems (SANFIS)
8) การหาตำแหน่งหุ่นยนต์จากภาพ
9) การประมาณค่าตำแหน่งหุ่นยนต์ด้วย Kalman filter

1. โครงสร้างของระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์ค

โครงสร้างของระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คประกอบไปด้วย 3 ส่วนคือ เน็ตเวิร์ค (network) ตัวควบคุม (control agent) และแพลนต์ (plant หรือ action agent) ดังรูปต่อไปนี้



ภาพที่ 1 โครงสร้างของระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์ค

1.1 เน็ตเวิร์ค (network)

เน็ตเวิร์คเป็นสื่อกลางในการรับและส่งข้อมูลและเชื่อมต่อส่วนควบคุมและที่ต้องการควบคุม เข้าด้วยกันซึ่งคำว่าเน็ตเวิร์คในงานวิจัยนี้หมายถึง IP Network หรืออินเตอร์เน็ต (internet) ที่ใช้งานกันทั่วโลก ซึ่งแน่นอนว่าเน็ตเวิร์คที่ใช้เป็นสื่อกลางในการควบคุมหุ่นยนต์ยังถูกใช้ร่วมกับการดำเนินการอื่นๆ ยกตัวอย่างเช่น FTP (file transfer protocol) และ E-mail เป็นต้น ซึ่งถ้าจำนวนผู้เข้าใช้ระบบเน็ตเวิร์คมีจำนวนมาก จะทำให้ระบบควบคุมผ่านเครือข่ายเกิดปัญหาตามมา ซึ่งปัญหาหลักที่ทำให้ประสิทธิภาพการทำงานของระบบควบคุมผ่านเน็ตเวิร์คลดลงก็คือ ดีเลย์ (delay)

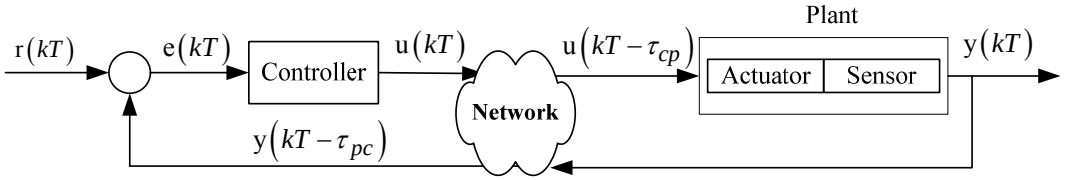
1.2 ตัวควบคุม (controller)

ตัวควบคุมเป็นส่วนที่ประมวลผลอัลกอริทึมบางอย่างเพื่อทำการควบคุมให้กระบวนการทำงานตามคำสั่งที่ได้รับ โดยในงานวิจัยนี้ใช้อัลกอริทึม quadratic curve เป็นตัวควบคุมให้หุ่นยนต์เคลื่อนที่ตามเส้นทางที่ได้รับ

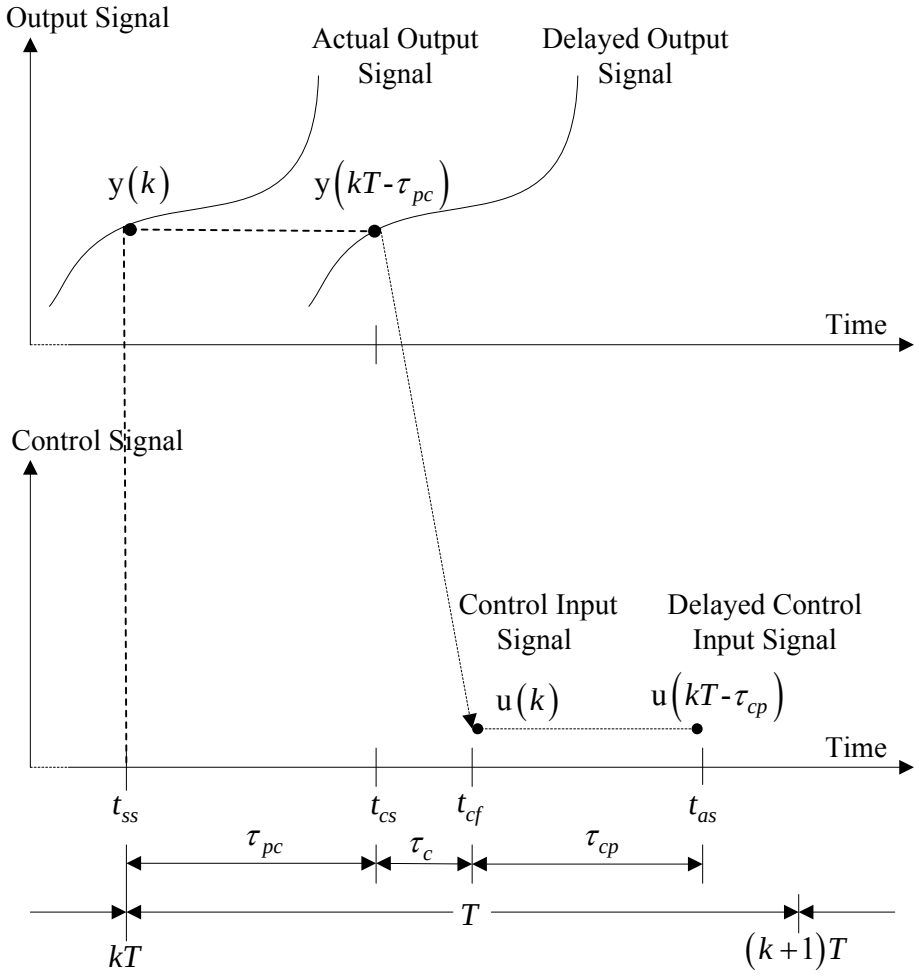
1.3 แพลนต์ (plant)

แพลนต์ในที่นี้หมายถึงหุ่นยนต์ โดยเมื่อแพลนต์ได้รับคำสั่งการควบคุมแล้วจะดำเนินการตามคำสั่งควบคุมที่ได้และจะมีเซ็นเซอร์ (sensor) เพื่อใช้ในการตรวจสอบสถานะการทำงานว่าได้ตามเป้าหมายหรือไม่ ซึ่งในงานวิจัยนี้ใช้กล้องเป็นตัวตรวจสอบการเคลื่อนที่ของหุ่นยนต์และรายงานค่าความผิดพลาด (error) ที่ได้ผ่านเน็ตเวิร์คกลับไปยังตัวควบคุม

2. ผลกระทบของดีเลย์ที่มีต่อประสิทธิภาพของระบบ



ภาพที่ 2 การทำงานของ NBC โดยมีดีเลย์เข้ามาเกี่ยวข้อง



ภาพที่ 3 ลักษณะช่วงเวลาการเกิดดีเลย์ (timing diagram of delay generations)

เนื่องด้วยในการติดต่อสื่อสารของ NBC นั้นตัวเน็ตเวิร์คเองจะถูกใช้เป็นสื่อกลางในการรับส่งข้อมูลและเชื่อมต่อส่วนควบคุมและแพลนต์เข้าด้วยกัน นอกจากนี้ตัวเน็ตเวิร์คยังถูกใช้ร่วมกับการดำเนินการอื่นๆ ยกตัวอย่างเช่น FTP (file transfer protocol), E-mail เป็นต้น โดยลักษณะ

เหตุการณ์ที่แพลนต์ได้รับข้อมูลจากตัวควบคุมหรือตัวควบคุมได้รับข้อมูลการตรวจสอบที่วัดได้จากแพลนต์เป็นไปในลักษณะเหตุการณ์ที่ไม่ต่อเนื่อง เพราะฉะนั้นการออกแบบระบบควบคุมด้วย discrete-time model จะเหมาะสมกว่า continue-time model

ในหัวข้อนี้นำเสนอการทำงานของ NBC ซึ่งมีดีเลย์เข้ามาเกี่ยวข้องและการเกิดดีเลย์ของ NBC ซึ่งนำเสนอดังภาพที่ 2 และ 3 ตามลำดับโดย *u* คือคำสั่งควบคุม (control signal), *r* คือสัญญาณอ้างอิง (reference signal), *y* คือผลลัพธ์ (output signal) และ *e* คือ error *(e = r-y)*

จากภาพที่ 2 ในการดำเนินการควบคุมผ่านเน็ตเวิร์ค ตัวควบคุมจะทำการประมวลผลคำสั่งควบคุม $\mathrm{u}(kT)$ ด้วยเวลา τ_c และส่งไปยังแพลนต์ทุกๆ ช่วงเวลา $t = kT$ โดย *k* คือลำดับของช่วงเวลาและ *T* คือช่วงเวลาที่ใช้ในการสุ่มตัวอย่าง (sampling time) ซึ่งคำสั่งควบคุมจะถูกหน่วงเวลาด้วยดีเลย์ τ_{cp} เพราะฉะนั้นจึงกำหนดให้คำสั่งที่แพลนต์ได้รับผลกระทบจากดีเลย์ τ_{cp} เป็น $\mathrm{u}(kT - \tau_{cp})$ เช่นเดียวกัน หลังจากที่แพลนต์ได้คำสั่งควบคุมแล้วแพลนต์จะทำการตอบรับโดยส่งผลลัพธ์ $\mathrm{y}(kT)$ กลับไปยังตัวควบคุมซึ่งจะถูกหน่วงเวลาด้วยดีเลย์ τ_{pc} จึงกำหนดให้ผลลัพธ์ที่ตัวควบคุมได้รับที่ได้รับผลกระทบจากดีเลย์ τ_{pc} เป็น $\mathrm{y}(kT - \tau_{pc})$ โดยคำอธิบายและการคำนวณหาดีเลย์ τ_{cp}, τ_{pc} และ τ_c มีดังต่อไปนี้

1) τ_{pc} : ดีเลย์ที่เกิดขึ้นระหว่างการส่งผลลัพธ์จากแพลนต์ไปยังตัวควบคุม (plant-to-controller delay) ซึ่งช่วงเวลาที่ใช้สามารถคำนวณได้จากสมการดังต่อไปนี้

$$\tau_{pc} = t_{cs} - t_{ss}, \tag{1}$$

โดย t_{cs} คือเวลาที่ตัวควบคุมเริ่มทำงานและ t_{ss} คือเวลาที่ sensor ของแพลนต์ได้รับสัญญาณจากตัวควบคุม

2) τ_c : ดีเลย์ที่เกิดขึ้นจากการคำนวณ (computational delay) ดีเลย์ที่เกิดขึ้นในส่วนนี้เป็นเวลาที่ตัวควบคุมใช้ในการคำนวณคำสั่งควบคุม ซึ่งช่วงเวลาที่ใช้สามารถคำนวณได้จากสมการดังต่อไปนี้

$$\tau_c = t_{cf} - t_{cs}, \tag{2}$$

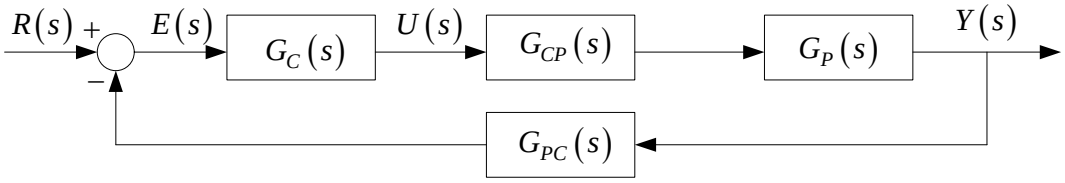
โดย t_{cf} คือเวลาที่ตัวควบคุมคำนวณคำสั่งในการควบคุมเสร็จ แต่ในทางปฏิบัติแล้วดีเลย์ที่เกิดขึ้นจากการคำนวณของตัวควบคุมจะสามารถกำหนดให้เป็นค่าคงที่หรืออาจจะไม่นำมาคิดในการดำเนินก็เป็นได้ เพราะถ้าเทียบสัดส่วนกันกับดีเลย์ที่เกิดจากการติดต่อข้ามเน็ตเวิร์คแล้วดีเลย์ที่เกิดจากการคำนวณจะมีค่าน้อยมาก

3) τ_{cp} : ดีเลย์ที่เกิดขึ้นระหว่างการส่งคำสั่งควบคุมจากตัวควบคุมไปยังแพลนต์ (controller-to-plant delay)

$$\tau_{cp} = t_{as} - t_{cf}, \tag{3}$$

โดย t_{as} คือเวลาที่ actuator ของแพลนต์ได้รับสัญญาณจากตัวควบคุมและเริ่มดำเนินการ

ปัญหาหลักของ NBC คือดีเลย์ที่เกิดขึ้นระหว่างการติดต่อข้ามเน็ตเวิร์ครวมทั้ง τ_{cp} และ τ_{pc} โดยดีเลย์จากที่ได้กล่าวข้างต้นเป็นปัญหาสำคัญยิ่งที่จะมีผลกระทบโดยตรงกับประสิทธิภาพและเสถียรภาพของระบบ



ภาพที่ 4 โครงสร้าง NBC ที่แสดงถึงผลกระทบของดีเลย์ที่มีต่อประสิทธิภาพและเสถียรภาพในการทำงาน

เพื่อแสดงให้เห็นถึงผลกระทบของดีเลย์ที่เกิดขึ้นในการทำงานของ NBC จึงกำหนดโครงสร้าง NBC ควบคุมมอเตอร์ไฟฟ้ากระแสตรง (dc motor) กับตัวควบคุม PI (proportional integral controller) เพื่อใช้ทดสอบประสิทธิภาพและเสถียรภาพซึ่งนำเสนอในรูปแบบของลาปลาซ (Laplace domain) ดังแสดงในภาพที่ 4 โดยมีการกำหนดตัวแปรดังต่อไปนี้

1. $R(s)$ คือสัญญาณอ้างอิง (reference signal) ใน Laplace domain
2. $U(s)$ คือคำสั่งควบคุม (control signal) ใน Laplace domain
3. $Y(s)$ คือผลลัพธ์ (output signal) ใน Laplace domain
4. $E(s) = R(s) - Y(s)$ คือ error ใน Laplace domain
5. $G_C(s)$ คือ PI controller ซึ่งนำเสนอในรูป transfer function ดังต่อไปนี้

$$G_C(s) = \frac{K_P\left(s + \left(K_I / K_P\right)\right)}{s} = \frac{K_P\left(s + z_C\right)}{s}, \tag{4}$$

โดย K_p คือ proportional gain, โดยในที่นี้ให้ค่า $K_p = 0.1701$

K_I คือ integral gain, โดยในที่นี้ให้ค่า $K_I = 0.3780$

6. $G_P(s)$ คือแพลนต์ของมอเตอร์ไฟฟ้ากระแสตรง (DC motor) ซึ่งนำเสนอในรูป transfer function ดังต่อไปนี้

$$G_P(s) = \frac{2029.826}{(s + 26.29)(s + 26.29)}. \tag{5}$$

7. $G_{CP}(s)$ คือดีเลย์เน็ตเวิร์คที่เกิดขึ้นระหว่างการส่งคำสั่งควบคุม $U(s)$ จาก PI controller $G_C(s)$ ไปยังแพลนต์ $G_P(s)$ ซึ่งนำเสนอดังสมการต่อไปนี้

$$G_{CP}(s) = e^{-\tau_{cp}s}. \tag{6}$$

8. $G_{PC}(s)$ คือดีเลย์เน็ตเวิร์คที่เกิดขึ้นระหว่างการส่งผลลัพธ์ $Y(s)$ จากแพลนต์กลับไปยัง PI controller ซึ่งนำเสนอดังสมการต่อไปนี้

$$G_{PC}(s) = e^{-\tau_{pc}s}. \tag{7}$$

9. ค่าดีเลย์ใน (7) และ (8) สามารถนำมาประมาณได้ด้วยวิธี numerator degree zero (Kuo ,1987) ดังสมการดังต่อไปนี้

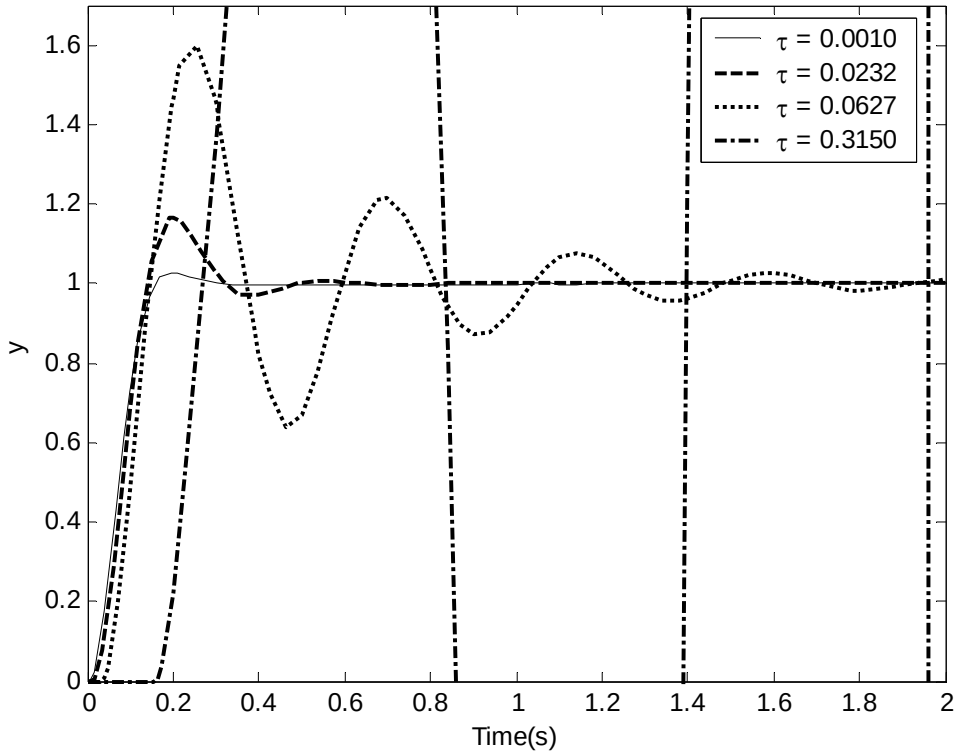
$$e^{-\tau s} \cong \left(1+\left(\tau s/n\right)\right)^{-n}. \tag{8}$$

โดย $\tau_{cp} = \tau_{pc} = \tau/2$

10. สมการ transfer function ของระบบที่มีดีเลย์ในเน็ตเวิร์คมีรูปแบบดังนี้

$$\frac{Y(s)}{R(s)} = \frac{G_C(s)G_{CP}(s)G_P(s)}{1+G_C(s)G_{CP}(s)G_P(s)G_{PC}(s)}. \tag{9}$$

เนื่องด้วยดีเลย์เป็นปัญหาหลักที่ทำให้ประสิทธิภาพการทำงานของระบบลดลง เพื่อแสดงให้เห็นถึงผลกระทบของดีเลย์นั้น (Tipsuwan and Chow, 1999) ทำการทดสอบ step response ของมอเตอร์ไฟฟ้ากระแสตรงด้วยค่าคงที่ของดีเลย์ τ = 0.001, 0.0232, 0.0627 และ 0.3150 เพื่อตรวจสอบผลกระทบที่เกิดขึ้นโดยกำหนดให้ $\tau_{cp} = \tau_{pc} = \tau/2$



ภาพที่ 5 step responses ที่เปลี่ยนแปลงอันเนื่องจากดีเลย์ที่เพิ่มขึ้น

จากภาพที่ 5 แสดงให้เห็นว่าเมื่อดีเลย์มีค่าเพิ่มขึ้นก็จะทำให้ค่า overshoot และ setting time มีค่าสูงขึ้นและเวลาที่มากขึ้นตามลำดับ

เนื่องด้วยดีเลย์เป็นปัญหาหลักที่มีผลต่อเสถียรภาพของระบบเพื่อแสดงให้เห็นถึงผลกระทบของดีเลย์นั้น (Tipsuwan and Chow, 1999) ได้ยกตัวอย่างโดยทำการตรวจสอบเสถียรภาพของระบบด้วยวิธี root locus กับ characteristic equation ดังต่อไปนี้

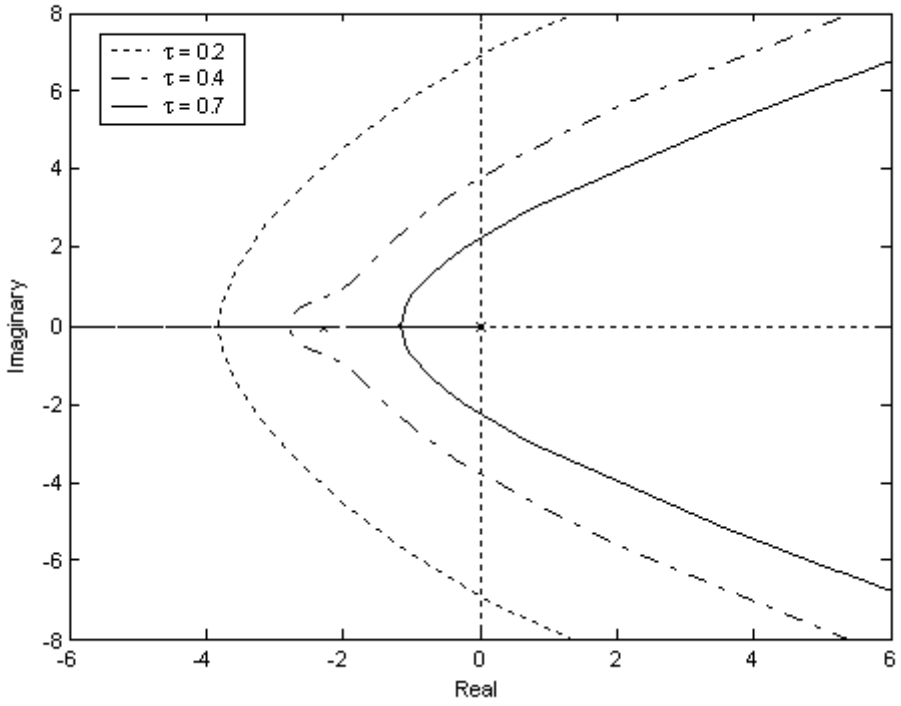
$$1+G_C(s)G_{CP}(s)G_P(s)G_{PC}(s)$$

$$=e^{-\tau s}\frac{2029.826K_P(s+z_C)}{s(s+26.29)(s+2.296)}$$

$$\cong\frac{2029.826K_P n^n(s+z_C)}{\tau^n s(s+26.29)(s+2.296)\left(s+\left(\frac{n}{\tau}\right)\right)^n}. \qquad (10)$$

โดยกำหนดให้ $n = 4$ และทำการตรวจสอบด้วยค่าดีเลย์ τ = 0.2, 0.4 และ 0.7 ตามลำดับ

เนื่องจากการเพิ่มค่าดีเลย์ (τ) ดังภาพที่ 6 รากของสมการที่ (10) จะเคลื่อนที่เข้าสู่ด้านขวาของแกนจินตภาพ (imaginary axis) ซึ่งแสดงถึงขอบเขตของเสถียรภาพของระบบที่ลดลง



ภาพที่ 6 การตรวจสอบเสถียรภาพของระบบด้วย root locus

3. งานวิจัยด้านระบบควบคุมผ่านเน็ตเวิร์คที่เกี่ยวข้อง

มีงานวิจัยหลายชิ้นได้นำเสนอวิธีการเกี่ยวกับการแก้ปัญหาที่เกิดขึ้นกับการควบคุมแบบปิดผ่านเน็ตเวิร์ค โดยการใช้เทคนิคเกี่ยวกับการควบคุมเข้ามาจัดการเพื่อให้ประสิทธิภาพของระบบเกิดความมีเสถียรภาพและประสิทธิภาพอยู่เกณฑ์ที่ยอมรับได้ ยกตัวอย่างเช่น

Augmented deterministic discrete-time model method (Halevi and Ray, 1988) เป็นงานที่ใช้วิธีการให้บริการ (state-augmentation) ซึ่งสามารถประยุกต์ใช้ได้ดีกับรูปแบบเน็ตเวิร์คที่มีลักษณะเป็นวงเชื่อมต่อกัน (cyclic service networks) โดยข้อดีหลักๆของงานก็คือสามารถทำได้ง่ายเนื่องจากเป็นการแก้ปัญหาที่ตรงไปตรงมาและสามารถลดการชนกันของ packet โดยการใช้ช่วงที่ว่างของ sampling time

Queuing method (Luck and Ray, 1990) เป็นงานวิจัยที่ใช้หลักการแถวคอย (queue) เพื่อที่จะใช้คาดการและทำการชดเชยดีเลย์ที่เกิดขึ้น โดยดีเลย์ที่เปลี่ยนแปลงจะถูกเปลี่ยนเป็นดีเลย์คงที่

Optimal stochastic control method (Nilsson and Wittenmark, 1998) เป็นงานวิจัยที่พยายามกำหนดปัญหาดีเลย์ในเน็ตเวิร์คเหมือนปัญหา LQG (linear quadratic Gaussian) ซึ่งจากผลที่ได้แสดงให้เห็นว่าสามารถแก้ปัญหาดีเลย์ในเน็ตเวิร์คได้ดีในระบบเชิงเส้น (linear system) กับสภาพเน็ตเวิร์คที่ซับซ้อน

Perturbation method (Walsh, Beldiman, and Bushnell, 1998) เป็นวิธีการแรกที่มีจุดประสงค์หลักคือการแก้ปัญหาดีเลย์ในเน็ตเวิร์คกับระบบไม่เชิงเส้น (nonlinear systems) โดยอาศัยการ perturbation เป็นหลัก

Sampling time scheduling method (Hong, 1995) เป็นงานวิจัยที่มองปัญหาดีเลย์ในเน็ตเวิร์คเป็น discrete time model มากกว่า time-varying model โดยใช้หลักการจัดลำดับของ sampling time

Robust control method (Goktas, 2000) เป็นงานวิจัยที่ทำการโมเดลดีเลย์ในเน็ตเวิร์คโดยใช้วิธีการ first-order Páde approximation และใช้หลักการจัดลำดับของ robust control มาชดเชยผลจากดีเลย์และควบคุมระบบโดยจากสมมุติฐานว่าผลกระทบจากดีเลย์เป็นสัญญาณรบกวนแบบหนึ่ง

Smith predictor method (Sichitiu, 2001) เป็นงานที่ทำการประยุกต์ใช้ Smith predictor (O.J.M. Smith, 1957) ทำนายดีเลย์ที่จะเกิดขึ้นและทำการปรับค่าพารามิเตอร์ของตัวควบคุม

Fuzzy logic method (Lee, 2001) เป็นงานที่ทำการประยุกต์ใช้ Fuzzy logic เป็นตัวควบคุมเพื่อชดเชยผลจากดีเลย์

Event-based control method (Tarn and Xi, 1998) เป็นงานที่ใช้เหตุการณ์ในการกำหนดการทำงานของระบบ (event-based) โดยจะมีการสร้าง motion reference mapping สำหรับเน็ตเวิร์คไว้ก่อนซึ่งแน่นอนว่าข้อดีของงานนี้ก็คือเสถียรภาพของระบบไม่ขึ้นกับค่าดีเลย์

Optimal gain scheduling (Tipsuwan and Chow, 2004) เป็นวิธีการที่ปรับพารามิเตอร์ในตัวควบคุมให้เหมาะสมกับสภาวะเน็ตเวิร์ค ณ ตอนนั้นๆ

งานวิจัยที่ได้กล่าวมาข้างต้นถูกจำกัดไปด้วยข้อกำหนดหรือขอบเขตการดำเนินการบางอย่าง ยกตัวอย่างเช่น มีการกำหนดค่าขอบเขตสูงสุดของดีเลย์ (maximal delay bound) การทำงานยังต้องขึ้นอยู่กับโมเดลดีเลย์ที่ทำการอ้างอิง เป็นต้น ซึ่งในทางปฏิบัติแล้วไม่สามารถกำหนดข้อจำกัดในการทำงานเหล่านี้ได้ เพราะว่าความจริงแล้วลักษณะการเกิดดีเลย์ขึ้นบนเน็ตเวิร์คนั้นมีความซับซ้อนและกำกวม เนื่องจากเราไม่สามารถกำหนดลักษณะที่แน่นอนหรือคาดคะเนการเกิดดีเลย์บนเน็ตเวิร์คได้

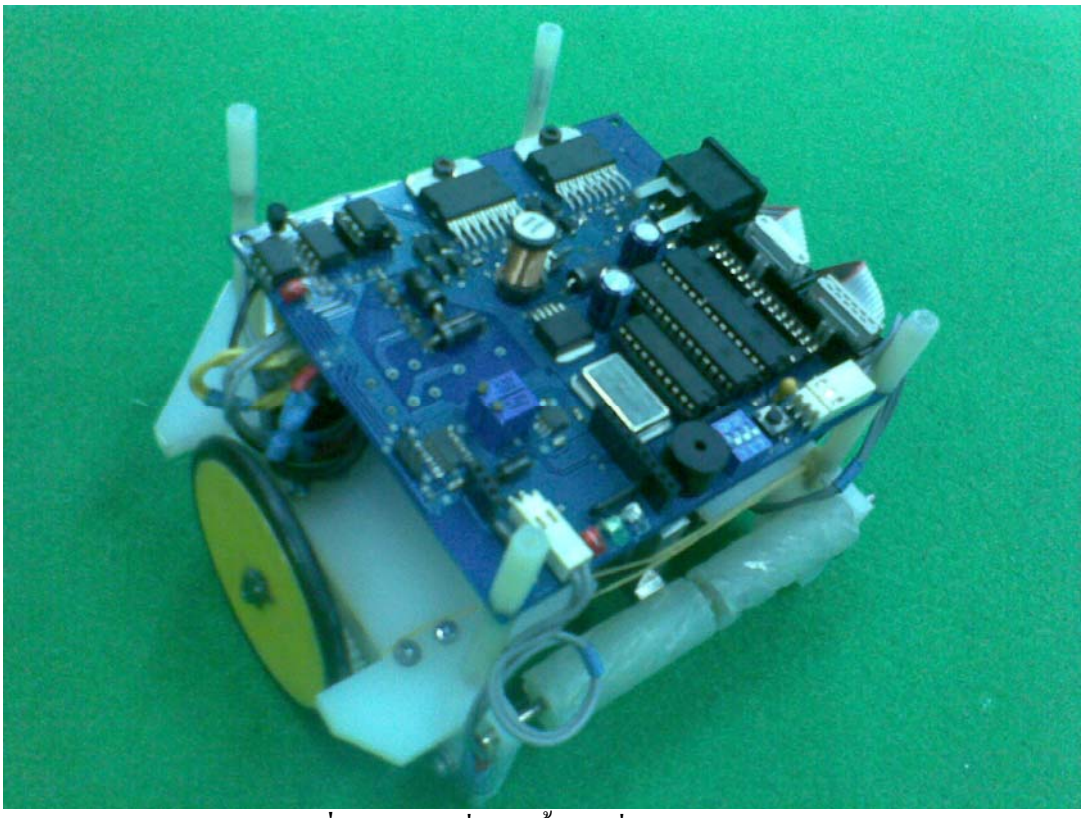
งานวิจัยนี้จึงได้นำเสนอเทคนิคใหม่ที่ใช้ neuro-fuzzy มาประยุกต์เพื่อเพิ่มประสิทธิภาพตัวควบคุม (controller) ในการควบคุมผ่านเน็ตเวิร์ค ซึ่ง neuro-fuzzy เป็นการประยุกต์รวมเอาข้อดีของทั้ง fuzzy logic และ neural network เข้ามารวมไว้ด้วยกัน โดยที่ fuzzy logic นำการใช้เหตุและผลของมนุษย์มาจัดการความกำกวมของดีเลย์ที่เกิดขึ้นและ neural network ใช้เพื่อเพิ่มประสิทธิภาพของระบบด้วยการเรียนรู้ส่วนของระบบที่ไม่ทราบถึงการทำงานที่แน่นอนจากตัวอย่างข้อมูล

4. การออกแบบตัวหุ่นยนต์

ในงานวิจัยนี้ได้ออกแบบและสร้างหุ่นยนต์เคลื่อนที่ได้ (mobile robot) เพื่อใช้ในการทดลองควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คจริงๆ โดยหุ่นยนต์ที่ทำการออกแบบและสร้างเป็นชนิดหุ่นยนต์ 2 ล้อ (differential-drive) ซึ่งแสดงในภาพที่ 7 โดยการออกแบบโครงสร้างหลักๆของตัวหุ่นยนต์แบ่งออกเป็นสองส่วนคือ

4.1 การออกแบบทางเครื่องจักรกล (mechanical design)

4.2 การออกแบบทางอิเล็กทรอนิกส์ (electronic design)



ภาพที่ 7 หุ่นยนต์ที่สร้างขึ้นมาเพื่อใช้ในการทดลอง

4.1 การออกแบบทางเครื่องจักรกล (mechanical design)

เนื่องจากต้องการให้หุ่นยนต์มีน้ำหนักเบาที่สุดเพื่อจะได้ควบคุมได้ง่ายโครงสร้างหุ่นยนต์ส่วนใหญ่จึงทำมาจากพลาสติกวิศวกรรมชนิด POM ซึ่งพลาสติกชนิดนี้มีคุณสมบัติ แข็ง, ลื่น, ไม่อมความชื้น และทนกระแสไฟได้ โดยพลาสติกชนิดนี้เป็นที่นิยมในการทำชิ้นงานแทน

โลหะ ซึ่งชิ้นส่วนของหุ่นยนต์ทั้งหมดใช้โปรแกรม Autodesk Inventor 2008 ในการออกแบบโดยชิ้นงานทั้งหมดถูกผลิตด้วยเครื่อง CNC เพื่อให้ชิ้นงานออกมามีความแม่นยำมากที่สุด

หุ่นยนต์ใช้มอเตอร์เป็นระบบขับเคลื่อนหลัก โดยเลือกใช้มอเตอร์ไฟฟ้ากระแสตรง (DC Motor) ยี่ห้อ Maxon A-max 226802 และใช้ชุดเกียร์ทดซึ่งมีอัตราทด 5:1 และนอกจากนี้มอเตอร์รุ่นนี้ยังมี optical encoder ซึ่งเป็นเซ็นเซอร์วัดความเร็วติดตั้งมาให้ด้วย

4.2 การออกแบบทางอิเล็กทรอนิกส์ (electronic design)

ไมโครโปรเซสเซอร์ที่ใช้ในการควบคุมการทำงานของหุ่นยนต์คือไมโครคอนโทรลเลอร์ตระกูล PIC รุ่น dsPIC30f2010 เหตุผลที่เลือกใช้ไมโครคอนโทรเลอร์ตัวนี้เนื่องจากมีประสิทธิภาพสูงและที่สำคัญคือไมโครคอนโทรเลอร์ตัวนี้มีโมดูลอ่านความเร็วของมอเตอร์จาก optical encoder ได้เลยโดยไม่จำเป็นต้องสร้างวงจรใหม่เพื่อทำการอ่านค่า

สิ่งสำคัญในการควบคุมความเร็วและทิศทางการเคลื่อนที่ของหุ่นยนต์ที่ใช้ล้อขับเคลื่อนนั้นคือการควบคุมความเร็วของล้อแต่ละล้อให้มีความถูกต้องแม่นยำ ซึ่งในการควบคุมความเร็วของล้อหุ่นยนต์ที่ใช้เป็นการควบคุมความเร็วมอเตอร์แบบระบบปิด (closed-loop control) โดยใช้ตัวควบคุมแบบ PI (Proportional-Integral controller) โดยไมโครคอนโทรลเลอร์จะทำการอ่านความเร็วจาก optical encoder แล้วนำข้อมูลที่อ่านได้ไปเปรียบเทียบกับความเร็วที่ต้องการ ค่าความผิดพลาด (error) และผลรวมของความผิดพลาดในอดีตจนถึงปัจจุบัน (integral error) จะถูกนำไปใช้ในการคำนวณเพื่อหาค่า duty cycle ของสัญญาณ PWM (Pulse-Width modulation) ที่เหมาะสมเพื่อที่จะขับมอเตอร์ให้หมุนตามความเร็วที่สั่ง

สมการต่อไปนี้แสดงการทำงานของอัลกอริทึมที่ใช้สำหรับตัวควบคุมแบบ PI control โดยการทำงานของอัลกอริทึมจะทำการปรับค่าของคำสั่งควบคุม (u) ตามค่าความผิดพลาด (e) ระหว่างค่าอ้างอิง (r) และค่าเอาท์พุต (y) ที่ได้อ่านได้จาก encoder ของมอเตอร์

$$u(t) = K_P e(t) + K_I \int_0^t e(\zeta)\, d\zeta. \tag{11}$$

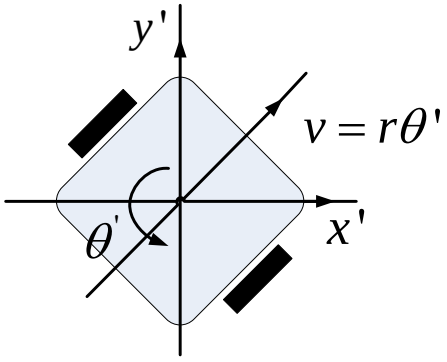
โดย K_P คือ proportional gain และ K_I คือ integral gain

5. รูปแบบทางคณิตศาสตร์ของหุ่นยนต์

งานวิจัยชิ้นนี้ใช้แพลนต์เป็นหุ่นยนต์เคลื่อนที่ (mobile robot) โดยนำหุ่นยนต์ที่ออกแบบและสร้างขึ้นจากหัวข้อที่แล้วไปหาแบบจำลองทางคณิตศาสตร์ ซึ่งรูปแบบทางคณิตศาสตร์ของหุ่นยนต์จะประกอบไปด้วย 3 ส่วน ดังต่อไปนี้

5.1 แบบจำลองทางจลนศาสตร์ของหุ่นยนต์

แบบจำลองทางจลนศาสตร์ของหุ่นยนต์ (kinematics model) คือสมการที่ใช้แปลงความเร็วเชิงเส้นและความเร็วเชิงมุมของหุ่นยนต์ให้เป็นความเร็วของล้อทั้งสองด้านของหุ่นยนต์ ดังรูปต่อไปนี้



ภาพที่ 8 แบบจำลองทางจลนศาสตร์ของหุ่นยนต์

จากภาพที่ 8 จะได้สมการต่อไปนี้

$$\dot{x}_0 \cos\phi + \dot{y}_0 \sin\phi - b\dot{\phi} = r\dot{\theta}_l, \tag{12}$$

$$\dot{x}_0 \cos\phi + \dot{y}_0 \sin\phi + b\dot{\phi} = r\dot{\theta}_r. \tag{13}$$

นำสมการที่ (12) + (13)

$$2(\dot{x}_o \cos\phi + \dot{y}_o \sin\phi) = r(\dot{\theta}_r - \dot{\theta}_l),$$

$$v = \frac{r}{2}(\dot{\theta}_r - \dot{\theta}_l). \tag{14}$$

นำสมการที่ (12) - (13)

$$\dot{\phi} = \frac{r}{W}(\dot{\theta}_r - \dot{\theta}_l). \tag{15}$$

จากสมการที่ (14) และสมการที่ (15) จะได้แบบจำลองทางจลนศาสตร์ของหุ่นยนต์เป็นสมการดังต่อไปนี้ (Koichi Yoshizawa, Hideki Hashimoto, Masayoshi Wada, and Shunji Mori. 1996)

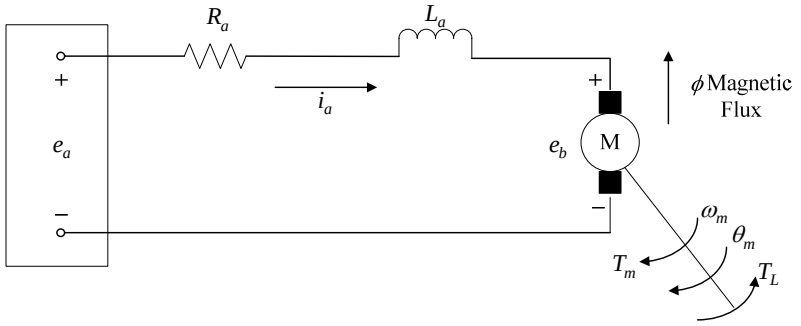
$$\begin{pmatrix} v \\ \omega \end{pmatrix} = \begin{pmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{W} & \frac{-r}{W} \end{pmatrix} \begin{pmatrix} \omega_R \\ \omega_L \end{pmatrix}. \tag{16}$$

ตารางที่ 1 พารามิเตอร์แบบจำลองทางจลนศาสตร์ของหุ่นยนต์

พารามิเตอร์	คำบรรยาย
v	ความเร็วในการเคลื่อนที่ไปข้างหน้าของหุ่นยนต์
w	ความเร็วในการหมุนของหุ่นยนต์
r	รัศมีล้อของหุ่นยนต์
W	ระยะห่างระหว่างล้อทั้งสองข้าง
ω_L	ความเร็วในการหมุนของล้อซ้าย
ω_R	ความเร็วในการหมุนของล้อขวา

5.2 แบบจำลองทางคณิตศาสตร์ของมอเตอร์กระแสตรง

แบบจำลองทางคณิตศาสตร์ของมอเตอร์กระแสตรง (DC motor model) คือรูปแบบการทำงานทางคณิตศาสตร์ของมอเตอร์ไฟฟ้ากระแสตรง โดยมอเตอร์กระแสตรงมีรูปแบบทางวงจรไฟฟ้าดังภาพต่อไปนี้



ภาพที่ 9 วงจรไฟฟ้าของมอเตอร์ไฟฟ้ากระแสตรง

จากภาพวงจรไฟฟ้าของมอเตอร์ไฟฟ้ากระแสตรงจะประกอบด้วยตัวต้านทาน (resistance), ตัวเหนี่ยวนำ (inductance), แรงดันขดลวด (armature voltage) และ แรงดันย้อนกลับ (back EMF) ที่เกิดจากการหมุนโรเตอร์ทำให้เกิดแรงดันไฟฟ้า นอกจากนี้ยังมีพารามิเตอร์สำคัญอื่นๆ ที่ใช้ในมอเตอร์ไฟฟ้ากระแสตรงดังนี้

ตารางที่ 2 พารามิเตอร์ของแบบจำลองทางคณิตศาสตร์ของมอเตอร์กระแสตรง

พารามิเตอร์	คำบรรยาย	พารามิเตอร์	คำบรรยาย
R_a	Armature resistance	i_a	Armature current
L_a	Armature inductance	e_a	Armature voltage
K_b	Back EMF constant	e_b	Back EMF voltage
K_i	Torque constant	T_L	Load torque
J	Moment of inertia	T_m	Motor torque
B	Viscous-friction coefficient	ϕ	Magnetic flux
θ_m	Motor displacement	ω_m	Rotor angular velocity

จากภาพวงจรไฟฟ้าของมอเตอร์ไฟฟ้ากระแสตรงสามารถอธิบายได้ด้วยสมการดังต่อไปนี้

$$T_m = K_i i_a, \tag{17}$$

จากกฎของ Kirchhoff เราสามารถหาอนุพันธุ์อันดับที่ 2 ได้เป็น

$$\frac{di_a(t)}{dt} = \frac{1}{L_a} e_a(t) - \frac{R_a}{L_a} i_a(t) - \frac{1}{L_a} e_b(t), \tag{18}$$

โดย

$$e_b(t) = K_b \omega_m(t), \tag{19}$$

$$\frac{d\theta_m(t)}{dt} = \omega_m(t). \tag{20}$$

จากกฎของนิวตัน ($\sum \Gamma = i_a \omega$)

$$\frac{d^2\theta_m(t)}{dt^2} = \frac{1}{J_m} T_m(t) - \frac{1}{J_m} T_L(t) - \frac{B_m}{J_m} \frac{d\theta_m(t)}{dt}. \tag{21}$$

จากสมการที่ (17) - (21) กำหนดให้ x_1 และ x_2 คือ ω_m และ i_a เพราะฉะนั้นจะได้สมการมอเตอร์กระแสตรงจึงสามารถจัดอยู่ในรูปแบบของเมตริกซ์ดังต่อไปนี้

$$\dot{\mathbf{x}}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{K_b}{L_a} \\ \frac{K_i}{J} & -\frac{B}{J} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} \\ 0 \end{bmatrix} \mathbf{u}(t),$$

$$\mathbf{y}(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}. \tag{22}$$

5.3 แบบจำลองทางพลศาสตร์ของหุ่นยนต์

แบบจำลองทางพลศาสตร์ของหุ่นยนต์ (robot dynamic model) คือสมการที่ใช้อธิบายแรงและการเคลื่อนที่ของหุ่นยนต์ ซึ่งจะพิสูจน์ตามกฎของนิวตันได้ดังต่อไปนี้

$$f_r + f_l = M\dot{v}, \tag{23}$$

$$\frac{W}{2}(f_r - f_l) = I\dot{\omega}. \tag{24}$$

ตารางที่ 3 พารามิเตอร์ของแบบจำลองทางพลศาสตร์ของหุ่นยนต์

พารามิเตอร์	คำบรรยาย
f_r	แรงในการขับเคลื่อนล้อขวา
f_l	แรงในการขับเคลื่อนล้อซ้าย
I	โมเมนต์ความเฉื่อยของโครงสร้างหุ่นยนต์
W	ระยะห่างระหว่างล้อทั้งสองข้าง
M	มวลของหุ่นยนต์

ถ้านำแบบจำลองทางคณิตศาสตร์ของมอเตอร์กระแสตรงมาประยุกต์เข้ากับสมการที่ (23) และ (24) จะได้แบบจำลองทางพลศาสตร์ของหุ่นยนต์ ในเทอมของมอเตอร์ไฟฟ้ากระแสตรงดังนี้

$$(k^2 J_1 + J_2 + \frac{M}{2} r^2)\dot{v} + (k^2 B_1 + B_2 + \frac{k_m^2 k^2}{R_a})v = \frac{k_m k}{R_a} \frac{r(V_{ar} + V_{al})}{2}, \tag{25}$$

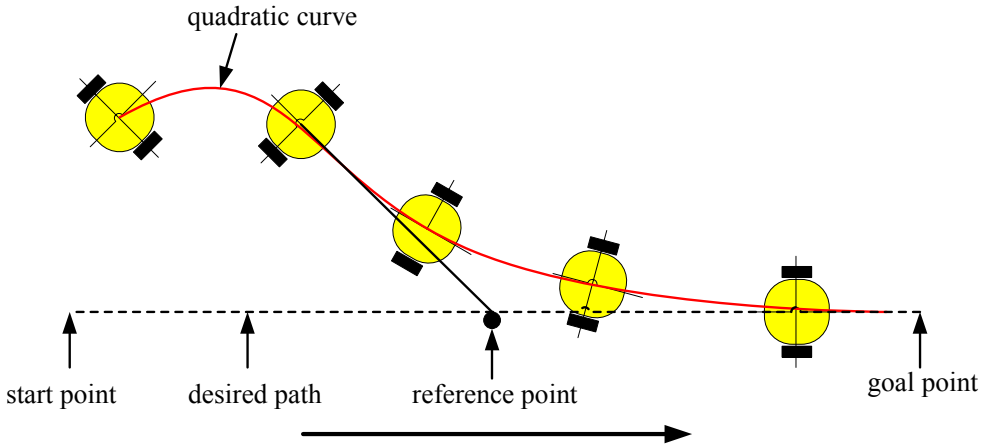
$$(k^2 J_1 + J_2 + \frac{2I}{W^2} r^2)\dot{\omega} + (k^2 B_1 + B_2 + \frac{k_m^2 k^2}{R_a})\omega = \frac{k_m k}{R_a} \frac{r(V_{ar} - V_{al})}{W}. \tag{26}$$

ตารางที่ 4 พารามิเตอร์ของแบบจำลองทางพลศาสตร์ของหุ่นยนต์กับมอเตอร์กระแสตรง

พารามิเตอร์	คำบรรยาย	พารามิเตอร์	คำบรรยาย
J_1	Moment of inertia on left wheel	J_2	Moment of inertia on right wheel
k_m	Torque constant	V_{al}	Input voltage to left DC motor
k	Gear Ratio	R_a	Armature Resistance
V_{ar}	Input voltage to right DC motor	W	Distance between wheels
B_1	Viscous friction constant on left wheel	B_2	Viscous friction constant on right wheel

6. อัลกอริทึมในการเดินตามเส้นทางของหุ่นยนต์

ในส่วนควบคุมของระบบควบคุมผ่านเน็ตเวิร์คในงานวิจัยนี้จะเป็นส่วนที่ทำหน้าที่ควบคุมการเดินของหุ่นยนต์ให้อยู่บนเส้นทางที่กำหนด (path tracking) โดยในงานวิจัยนี้ได้เลือกใช้อัลกอริทึม quadratic curve เป็นอัลกอริทึมที่ใช้ในการควบคุมการเดินของหุ่นยนต์ซึ่งมีลักษณะดังภาพต่อไปนี้



ภาพที่ 10 ลักษณะการเดินตามเส้นทางของอัลกอริทึม quadratic curve

โดยหลักการของอัลกอริทึม คือจะกำหนดให้มีจุดอ้างอิงซึ่งจะเคลื่อนที่บนเส้นทางของหุ่นยนต์ เพราะฉะนั้นจะเกิดระยะห่างระหว่างหุ่นยนต์และจุดอ้างอิงนั้นอยู่เสมอ โดยขั้นตอนการทำงานของอัลกอริทึมนี้จะแบ่งเป็น 2 ขั้นตอน คือ 1) การสร้างเส้นทางเชื่อมต่อระหว่างจุดอ้างอิงและตำแหน่งหุ่นยนต์ 2) หาความเร็วเชิงเส้นและความเร็วเชิงมุมของหุ่นยนต์ ซึ่งมีรายละเอียดของวิธีการดังต่อไปนี้

กำหนดให้เวคเตอร์บอกตำแหน่งของหุ่นยนต์คือ $[x_c, y_c, \theta_c]^T$ และเวคเตอร์บอกตำแหน่งของจุดอ้างอิงคือ $[x_r, y_r, \theta_r]^T$ หาผลต่างที่เกิดขึ้นแล้วแปลงพิกัดจากพิกัดจริง (world coordinate) เป็นพิกัดหุ่นยนต์ (robot coordinate) ได้ดังนี้

$$\begin{pmatrix} e_x \\ e_y \\ e_\theta \end{pmatrix} = \begin{pmatrix} \cos\theta_c & \sin\theta_c & 0 \\ -\sin\theta_c & \cos\theta_c & 0 \\ 0 & 0 & 1 \end{pmatrix} (x_r - x_c) \qquad (27)$$

โดยที่ $e=\left(e_x,e_y,e_\theta\right)$ คือเวคเตอร์ของผลต่างที่เกิดขึ้น x_r คือตำแหน่งของจุดอ้างอิงและ x_c คือตำแหน่งของหุ่นยนต์

สมการเส้นโค้งแบบ quadratic ซึ่งคำนวณได้จากสมการต่อไปนี้

$$y=Ax^2, \tag{28}$$

$$A=sign(e_x)\frac{e_y}{e_x^2}. \tag{29}$$

โดยค่า A คือค่าที่บอกถึงความโค้งของเส้น ยกตัวอย่างเช่น ถ้า A มีค่ามากจะได้ความชันของความโค้งสูงหรือเส้นทางนี้มีความโค้งมาก

หาความเร็วเชิงเส้น v และหาความเร็วเชิงมุม ω ได้จากสมการต่อไปนี้

$$v=sign(e_x)\sqrt{\dot{x}^2(1+4A^2x^2)}, \tag{30}$$

$$\omega=\frac{2A\dot{x}^3}{v^2}, \tag{31}$$

ถ้าคิดแบบไม่ต่อเนื่องในช่วงเวลาสั้นๆ $n\Delta t\le t<(n+1)\Delta t$ โดยให้ A_n คือค่าที่บอกถึงความโค้งของเส้น ณ ช่วงเวลานั้น จะหาค่าความเร็วเชิงเส้น v_n และหาความเร็วเชิงมุม ω_n ได้จากสมการต่อไปนี้

$$v_n=K_n, \tag{32}$$

$$\omega_n=2A_nK_n, \tag{33}$$

โดยที่

$$K_n = sign(e_x)\frac{\alpha}{1+|A_n|}. \tag{34}$$

จากสมการจะเห็นได้ว่าความเร็วเชิงเส้นและความเร็วเชิงมุมจะขึ้นอยู่กับค่าคงที่ α เท่านั้น โดยในการใช้อัลกอริทึมนี้จะต้องกำหนดค่าระยะห่างระหว่างหุ่นยนต์กับจุดอ้างอิงที่มากที่สุด ($d_{\max}$) ให้เหมาะสมเนื่องจากถ้ากำหนดค่า $d_{\max}$ มีค่ามากเกินไป ความแม่นยำในการเดินตามเส้นทางของหุ่นยนต์จะลดลง แต่ถ้ากำหนดให้ $d_{\max}$ มีค่าน้อยเกินไปหุ่นยนต์ก็จะไม่สามารถเดินตามเส้นทางได้

กำหนดให้ β คือค่าคงที่จำนวนจริงบวกใดๆ จะหาค่าระยะห่างระหว่างหุ่นยนต์กับจุดอ้างอิง (d_0) และความเร็วของจุดอ้างอิง ($v_{r,n}$) ที่วิ่งบนเส้นทางได้จากสมการต่อไปนี้

$$d_0 = \frac{d_{\max}}{1+\beta|A_n|}, \tag{35}$$

$$v_{r,n} = \begin{cases} \hat{v}_n d_0 / d_n, & (e_x \geq 0). \\ 0, & (e_x < 0), \end{cases} \tag{36}$$

โดย $\hat{v}_n$ คือความเร็วของหุ่นยนต์

7. Self-Adaptive Neuro-Fuzzy Inference Systems

Self-Adaptive Neuro-Fuzzy Inference Systems (SANFIS) เป็นโครงสร้างที่ถูกนำมาประยุกต์ใช้ในการจำแนกสภาวะเน็ตเวิร์คที่เกิดขึ้นระหว่างการควบคุมผ่านเน็ตเวิร์ค เพื่อเพิ่มประสิทธิภาพการทำงานให้กับตัวควบคุมในงานวิจัยนี้ โดยลักษณะโครงสร้างนำเสนอดังนี้

SANFIS ประกอบด้วย J กฎ ซึ่งสามารถนำเสนอด้วยรูปแบบต่อไปนี้

$$\begin{aligned} &\text{Rule } j\text{: IF } s_1 \text{ is } A_1^j \text{ and } \ldots \text{ and } s_N \text{ is } A_N^j \\ &\qquad \text{THEN } v_1 \text{ is } \theta_k^j \text{ and } \ldots \text{ and } v_M \text{ is } \theta_M^j . \end{aligned} \tag{37}$$

โดย $j = 1,\ldots,J$; $s_i, i = 1,\ldots,N$ และ $v_k,\ k = 1,\ldots,M$ คืออินพุตและเอาท์พุตของ SANFIS ตามลำดับ

จากรูปแบบพื้นฐานข้างต้น ถ้าเลือกฟังก์ชันแสดงความเป็นสมาชิก (membership function) ในรูปแบบ Gaussian สำหรับตัวแปรอินพุตและรูปแบบ defuzzifier แบบ centroid method สมการเอาต์พุตของ SANFIS จะมีรูปแบบดังต่อไปนี้

$$v_k(\mathbf{s}) = \sum_{j=1}^{J} \theta_k^j P_j(\mathbf{s}), \tag{38}$$

โดย $\mathbf{s} = [s_1,\ldots,s_n]^{\mathbf{T}}$, θ_k^j คือค่า singleton membership, $p_j(\mathbf{s})$ คือรูปแบบ FBF (fuzzy basis function) ซึ่งแสดงในสมการต่อไปนี้

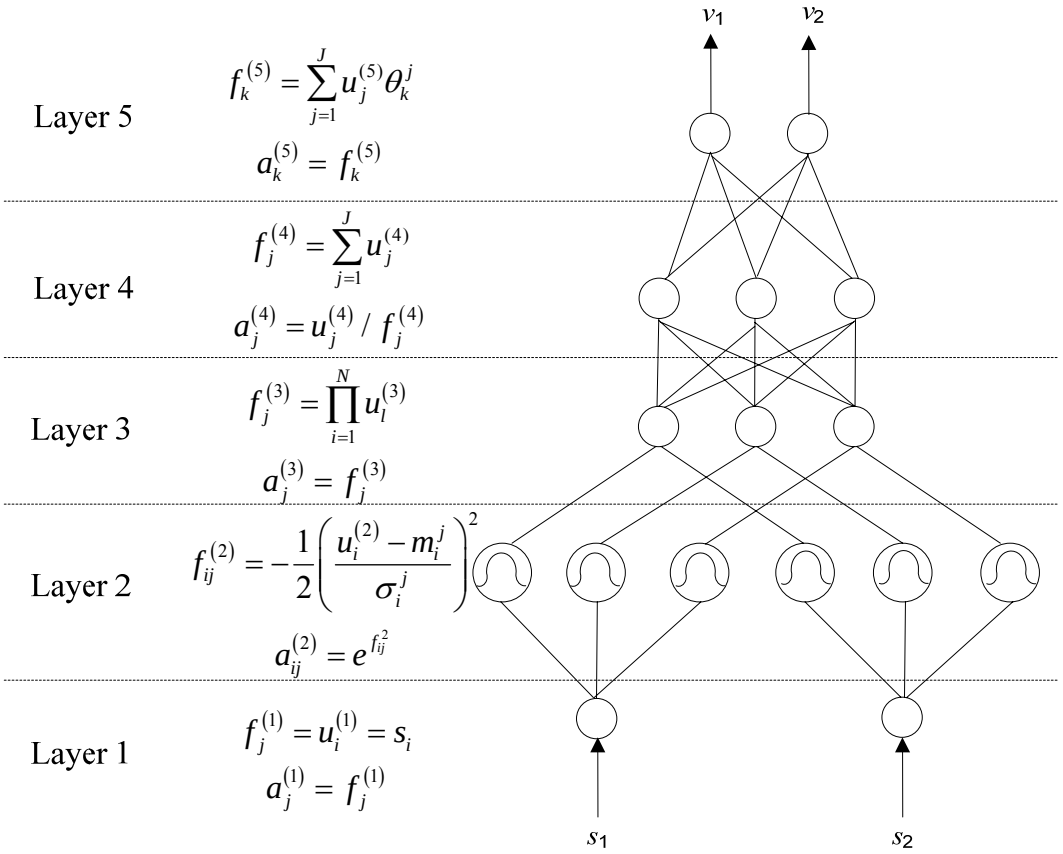
$$P_j(\mathbf{s}) = \frac{\prod_{i=1}^{N} \mu_{A_i^j}(s_i)}{\sum_{j=1}^{J}\prod_{i=1}^{N} \mu_{A_i^j}(s_i)}, \tag{39}$$

โดยที่ $\mu_{A_i^j}(s_i)$ คือฟังก์ชันแสดงความเป็นสมาชิกซึ่งแสดงอยู่ในสมการต่อไปนี้

$$\mu_{A_i^j}(s_i) = \exp\left[-\frac{1}{2}\left(\frac{s_i - m_i^j}{\sigma_i^j}\right)^2\right], \tag{40}$$

โดยที่ m_j^i คือค่าเฉลี่ย (mean) และ σ_i^j คือส่วนเบี่ยงเบนมาตรฐาน (standard deviation) ของฟังก์ชัน Gaussian เทอมที่ j ของอินพุตลำดับที่ i

ตัวอย่างของโครงสร้าง SANFIS ที่ประกอบด้วย 3 กฎฟัซซี่, 2 อินพุต, 2 เอาท์พุตซึ่งจัดเรียงในรูปแบบ 5 เลเยอร์ (ชั้น) แสดงในภาพต่อไปนี้



ภาพที่ 11 โครงสร้าง SANFIS ที่ประกอบด้วย 3 กฎ 2 อินพุต และ 2 เอาต์พุต

ผลลัพธ์ของแต่ละเลเยอร์จะอยู่ในรูป $a(f(.))$ ซึ่ง $u_1^{(l)},\ldots,u_P^{(l)}$ คืออินพุตของโหนดและ l คือลำดับเลเยอร์มีดังต่อไปนี้

เลเยอร์ 1: โหนดแต่ละโหนดในเลเยอร์นี้ทำหน้าที่ส่งข้อมูลที่ได้รับมาไปยังเลเยอร์ถัดไป

$$f_j^{(1)} = u_i^{(1)} = s_i \text{ และ } a_j^{(1)} = f_j^{(1)}, \tag{41}$$

เลเยอร์ 2: โหนดแต่ละโหนดในเลเยอร์นี้ประมวลผลการคำนวณด้วยฟังก์ชัน Gaussian

$$f_{ij}^{(2)} = -\frac{1}{2}\left(\frac{u_i^{(2)} - m_i^j}{\sigma_i^j}\right)^2 \text{ และ } a_{ij}^{(2)} = e^{f_{ij}^{(2)}}, \tag{42}$$

เลเยอร์ 3: เลเยอร์นี้ทำหน้าที่เป็นเหตุ (antecedents) ของกฎฟัซซี่ซึ่งนำไปใช้ในเลเยอร์ถัดไป โดยความสัมพันธ์ของโหนดจะนำเสนอในรูปแบบของ AND operation

$$f_j^{(3)} = \prod_{i=1}^{N} u_l^{(3)} \text{ และ } a_j^{(3)} = f_j^{(3)}, \tag{43}$$

เลเยอร์ 4: เลเยอร์นี้จะทำการเฉลี่ยความหนาแน่นของข้อมูล (normalizes firing strengths) ที่ได้จากเลเยอร์ที่ 3 ซึ่งจำนวนโหนดในเลเยอร์ที่ 4 นี้จะมีจำนวนเท่ากับจำนวนโหนดของเลเยอร์ 3

$$f_j^{(4)} = \sum_{j=1}^{J} u_j^{(4)} \text{ และ } a_j^{(4)} = u_j^{(4)} / f_j^{(4)}, \tag{44}$$

โดย $a_j^{(4)}$ คือ FBF ลำดับที่ j

เลเยอร์ 5: เลเยอร์นี้เป็นเลเยอร์สุดท้ายของโมเดลซึ่งจะรับข้อมูลมาจากเลเยอร์ที่ 4 มาประมวลผลเพื่อเปลี่ยนข้อมูลที่ได้ให้อยู่ในรูปแบบของผลลัพธ์ singleton ซึ่งการทำงานของเลเยอร์นี้เปรียบเสมือนกระบวนการ defuzzification

$$f_k^{(5)} = \sum_{j=1}^{J} u_j^{(5)} \theta_l^j \text{ และ } a_k^{(5)} = f_l^{(5)}. \tag{45}$$

อัลกอริทึมในการเรียนรู้ของ SANFIS

ในการใช้งาน SANFIS จำเป็นต้องปรับพารามิเตอร์ภายในเพื่อให้ SANFIS มีโครงสร้างที่เหมาะสมตามความต้องการ การปรับค่าพารามิเตอร์จะต้องอาศัยอัลกอริทึมในการเรียนรู้ซึ่งประกอบด้วยส่วนสำคัญ 2 ส่วนดังนี้

1 Mapping-Constrained Agglomerative (MCA) (J. C. Bezdek, 1999)

2 Fast recursive linear/nonlinear least-square optimization algorithm (Wang and Lee, 2002)

MCA เป็นคลัสเตอร์ริ่งอัลกอริทึมแบบ agglomerative ที่ใช้ในการจัดกลุ่มข้อมูล ซึ่งสามารถแบ่งกลุ่มข้อมูลโดยไม่ต้องระบุจำนวนกลุ่มเริ่มต้น ซึ่งต่างกับคลัสเตอร์ริ่งอัลกอริทึมแบบอื่น ๆ ยกตัวอย่างเช่น C-mean (Ross, 1995) ที่ต้องระบุจำนวนคลัสเตอร์ข้อมูลก่อนการประมวลผล

Fast recursive linear/nonlinear least-square optimization algorithm ใช้เพื่อปรับค่าพารามิเตอร์เพื่อให้โครงสร้าง SANFIS สามารถให้ค่าเอาท์พุตมีค่าใกล้เคียงกับค่าเอาท์พุตที่ต้องการมากที่สุด

Mapping-Constrained Agglomerative (MCA)

MCA เป็นคลัสเตอร์ริ่งอัลกอริทึมที่อาศัยการทดสอบความสอดคล้องของข้อมูลเพื่อใช้ในการจัดแบ่งข้อมูล ซึ่งแนวความคิดหลักในการทดสอบความสอดคล้องของข้อมูลนั้นคือ ถ้าข้อมูลอินพุตนั้นมีระยะห่างที่ใกล้ที่สุดกับคลัสเตอร์ใด ๆ ในอินพุตสเปซแล้วข้อมูลเอาท์พุตนั้นควรมีระยะห่างที่ใกล้ที่สุดกับคลัสเตอร์ลำดับเดียวกันในเอาท์พุตสเปซด้วย จึงจะรวมข้อมูลนั้นเข้าด้วยกัน หากเกิดความไม่สอดคล้องของข้อมูลกับคลัสเตอร์ใด ๆ ที่มีอยู่ก็ทำสร้างคลัสเตอร์ใหม่ขึ้นแทน ซึ่งก่อนการประมวลผลกำหนดให้

- เซตในการเทรนประกอบด้วยเวคเตอร์ order-pair ของอินพุต-เอาท์พุต p ตัว โดย $\left\{\left(\mathbf{s}_i, \mathbf{v}_i\right), \mathbf{s}_i \in \mathbb{R}^N \text{ และ } \mathbf{v}_i \in \mathbb{R}^M, i = 1, 2, \ldots, p\right\}$

- $\mathbf{m}_{Ik}$ และ $\boldsymbol{\sigma}_{Ik}$ เป็นคลัสเตอร์เซ็นเตอร์และส่วนเบี่ยงเบนมาตรฐานของฟังก์ชัน Gaussian ของคลัสเตอร์ที่ k ใด ๆ ในอินพุตสเปซ
- $\mathbf{m}_{Ok}$ และ $\boldsymbol{\sigma}_{Ok}$ เป็นคลัสเตอร์เซ็นเตอร์และส่วนเบี่ยงเบนมาตรฐานของฟังก์ชัน Gaussian ของคลัสเตอร์ที่ k ใด ๆ ในเอาท์พุตสเปซ
- c_k เป็นจำนวนสมาชิกในคลัสเตอร์ที่ k
- $K_{\max}$ เป็นจำนวนคลัสเตอร์ทั้งหมดตั้งต้น

ลำดับขั้นตอนของวิธี MCA

M1) System initialization

เป็นขั้นตอนในการเริ่มต้นระบบโดยการกำหนดค่า $K = 0$

M2) Initialization of seed clusters

เป็นขั้นตอนในการกำหนดค่าเริ่มต้นให้กับแต่ละคลัสเตอร์ โดยมีขั้นตอนย่อยๆดังต่อไปนี้

- กำหนดจำนวนคลัสเตอร์เริ่มต้นในการประมวลผล โดย $K = K_{\max}$
- กำหนดจำนวนสมาชิกเริ่มต้นให้แต่ละคลัสเตอร์ โดย $c_k = 1$
- กำหนดค่าเริ่มต้นของคลัสเตอร์เซ็นเตอร์และส่วนเบี่ยงเบนมาตรฐานแต่ละคลัสเตอร์ในอินพุตสเปซโดย $\mathbf{m}_{Ik} = \mathbf{s}_k$, $\boldsymbol{\sigma}_{Ik} = 0$; $k = 1, \ldots, K$
- กำหนดค่าเริ่มต้นของคลัสเตอร์เซ็นเตอร์และส่วนเบี่ยงเบนมาตรฐานแต่ละคลัสเตอร์ในเอาท์พุตสเปซโดย $\mathbf{m}_{Ok} = \mathbf{v}_k$, $\boldsymbol{\sigma}_{Ok} = 0$; $k = 1, \ldots, K$

M3) Creation of distance matrix

เป็นขั้นตอนสร้างเมตริกซ์ที่เก็บระยะห่างระหว่างคลัสเตอร์โดยทำการหาค่าเริ่มต้นของระยะห่างระหว่างคลัสเตอร์ด้วยซึ่งจัดเรียงอยู่ในรูป triangular matrix โดยกำหนดให้ระยะห่างในอินพุตสเปซเป็น $D_I \in \mathbb{R}^{K \times K}$ และระยะห่างในเอาต์พุตสเปซเป็น $D_O \in \mathbb{R}^{K \times K}$ ตามลำดับ โดยที่

$$d_{Iij} = \left\| \mathbf{m}_{Ii} - \mathbf{m}_{Ij} \right\|, \quad i, j \in \{1, \ldots, K\} \text{ and } i \neq j, \tag{46}$$

$$d_{Oij} = \left\| \mathbf{m}_{Oi} - \mathbf{m}_{Oj} \right\|, \quad i, j \in \{1, \ldots, K\} \text{ and } i \neq j, \tag{47}$$

โดย d_{Iij} คือระยะห่างระหว่างคลัสเตอร์เซ็นเตอร์ i และ j ในอินพุตสเปซ

d_{Oij} คือระยะห่างระหว่างคลัสเตอร์เซ็นเตอร์ i และ j ในเอาท์พุตสเปซ

$\|\bullet\|$ คือ Euclidean norm

M4) Data query

เป็นขั้นตอนในการกำหนดตัวข้อมูล $(\mathbf{s}_i, \mathbf{v}_i)$ ในลำดับที่ $i = K+1 \ldots p$ เพื่อทำการจัดกลุ่มว่าควรอยู่ในคลัสเตอร์ที่ 1 ถึง K หรือไม่

M5) Winner cluster determination

เป็นขั้นตอนการหาคลัสเตอร์เพื่อใช้ในการกำหนดกลุ่มสมาชิกของข้อมูล $(\mathbf{s}_i, \mathbf{v}_i)$ โดยใช้สมการดังต่อไปนี้

$$\text{winner}_I = \gamma \text{ โดยที่ } \left\| \mathbf{m}_{I\gamma} - \mathbf{s}_i \right\| \text{ มีค่าน้อยที่สุด} \tag{48}$$

$$\text{winner}_O = \delta \text{ โดยที่ } \left\| \mathbf{m}_{O\delta} - \mathbf{v}_i \right\| \text{ มีค่าน้อยที่สุด} \tag{49}$$

โดย winner_I คือคลัสเตอร์ที่มีระยะห่างน้อยที่สุดเทียบกับข้อมูล $\mathbf{s}_i$ ในอินพุตสเปซและ winner_O คือคลัสเตอร์ที่มีระยะห่างน้อยที่สุดเทียบกับข้อมูล $\mathbf{v}_i$ ในเอาท์พุตสเปซ

M6) Mapping consistency verification

เป็นขั้นตอนการทดสอบความสอดคล้องกันของข้อมูลที่จะทำการคลัสเตอร์ โดยดูจากคลัสเตอร์ที่มีระยะห่างเทียบกับข้อมูล $\mathbf{s}_i$ ในอินพุตสเปซและคลัสเตอร์ที่มีระยะห่างน้อยที่สุดเทียบกับ

ข้อมูล $\mathbf{v}_i$ ในเอาท์พุตสเปซจะต้องมีลำดับคลัสเตอร์เดียวกัน โดยถ้าค่า $\gamma = \delta$ ให้ดำเนินการในขั้นตอนที่ M6.1 แต่ถ้าเงื่อนไขไม่เป็นจริงให้ข้ามไปทำขั้นตอนที่ M7 ต่อไป

M6.1) ให้ทำการคำนวณค่าตัวแปรที่เกี่ยวข้องของคลัสเตอร์ winner ทั้งอินพุตและเอาท์พุตสเปซ

$$c_{winner} = c_{winner} + 1, \tag{50}$$

สำหรับอินพุตสเปซ

$$temp\_\mathbf{m}_{Iwinnner} = \mathbf{m}_{Iwinnner} + \frac{\mathbf{s}_i - \mathbf{m}_{Iwinner}}{c_{winner}}, \tag{51}$$

$$\boldsymbol{\sigma}^2_{Iwinner} = \frac{\left(c_{winner} - 1\right)\left(\boldsymbol{\sigma}^2_{Iwinner} + \mathbf{m}^2_{Iwinner}\right) + \mathbf{s}_i^2}{c_{winner}} - temp\_\mathbf{m}^2_{Iwinner}, \tag{52}$$

$$\mathbf{m}_{Iwinner} = temp\_\mathbf{m}_{Iwinner}, \tag{53}$$

สำหรับเอาท์พุตสเปซ

$$temp\_\mathbf{m}_{Owinner} = \mathbf{m}_{Owinner} + \frac{\mathbf{v}_i - \mathbf{m}_{Owinner}}{c_{winner}}, \tag{54}$$

$$\boldsymbol{\sigma}^2_{Owinner} = \frac{\left(c_{winner} - 1\right)\left(\boldsymbol{\sigma}^2_{Owinner} + \mathbf{m}^2_{Owinner}\right) + \mathbf{v}_i^2}{c_{winner}} - temp\_\mathbf{m}^2_{Owinner}, \tag{55}$$

$$\mathbf{m}_{Owinner} = temp\_\mathbf{m}_{Owinner}, \tag{56}$$

M6.2) ให้ทำการคำนวณระยะห่างระหว่างคลัสเตอร์เซ็นเตอร์ใหม่ทั้งในอินพุตและเอาท์พุตสเปซ

$$d_{Iij} = \left\| \mathbf{m}_{Ii} - \mathbf{m}_{Ij} \right\|, \quad i, j \in \{1, \ldots, K\} \text{ and } i \neq j, \tag{57}$$

$$d_{Oij} = \left\| \mathbf{m}_{Oi} - \mathbf{m}_{Oj} \right\|, \quad i, j \in \{1, \ldots, K\} \text{ and } i \neq j, \tag{58}$$

M6.3) ให้กลับไปที่ขั้นตอนที่ M4 ใหม่อีกครั้ง

M7) New potential cluster creation

เป็นขั้นตอนในการดูความเป็นไปได้ว่าจะต้องกำหนดคลัสเตอร์ใหม่ขึ้นมาหรือไม่โดยดูจากความสอดคล้องกันของตัวคลัสเตอร์ระหว่างอินพุตสเปซและเอาท์พุตสเปซ นั้นก็คือ การพิจารณคลัสเตอร์เซ็นเตอร์ในอินพุตสเปซว่าเป็นคู่อันดับเดียวกันกับคลัสเตอร์ 2 คลัสเตอร์ที่คลัสเตอร์เซ็นเตอร์มีระยะห่างที่น้อยที่สุดในเอาท์พุตสเปซ ซึ่งกำหนดให้ β และ α เป็นคู่อันดับคลัสเตอร์ที่มีระยะห่างน้อยที่สุดในอินพุตสเปซและ $s_{O\alpha\beta}$ เป็นตัวแปรที่ใช้เก็บค่าระยะห่างระหว่างคลัสเตอร์ β และ α ในเอาท์พุตสเปซ โดยถ้าค่า $s_{O\alpha\beta}$ มีค่าน้อยที่สุดในเอาท์พุตสเปซให้ดำเนินการตามขั้นที่ M7.1 – M7.3 แต่ถ้าไม่ให้ดำเนินการในขั้นตอน M7.4

M7.1) ทำการรวมคลัสเตอร์ β และ α เข้าด้วยกัน ด้วยการเทียบอัตราส่วนของจำนวนสมาชิกโดยหาค่าเฉลี่ยคลัสเตอร์เซ็นเตอร์และส่วนเบี่ยงเบนมาตรฐานใหม่ในคลัสเตอร์ทั้งสอง และหาผลรวมของจำนวนสมาชิกของทั้งสองคลัสเตอร์ด้วยสมการต่อไปนี้

สำหรับอินพุตสเปซ

$$temp\_\mathbf{m}_{I\alpha} = \frac{c_\alpha \mathbf{m}_{I\alpha} + c_\beta \mathbf{m}_{I\beta}}{c_\alpha + c_\beta}, \tag{59}$$

$$\boldsymbol{\sigma}_{I\alpha}^2 = \frac{c_\alpha\left(\boldsymbol{\sigma}_{I\alpha}^2+\mathbf{m}_{I\alpha}^2\right)+c_\beta\left(\boldsymbol{\sigma}_{I\beta}^2+\mathbf{m}_{I\beta}^2\right)}{c_\alpha+c_\beta} - temp\_\mathbf{m}_{I\alpha}^2, \tag{60}$$

$$\mathbf{m}_{I\alpha} = temp\_\mathbf{m}_{I\alpha}, \tag{61}$$

สำหรับเอาท์พุตสเปซ

$$\boldsymbol{\sigma}_{O\alpha}^2 = \frac{c_\alpha\left(\boldsymbol{\sigma}_{O\alpha}^2+\mathbf{m}_{O\alpha}^2\right)+c_\beta\left(\boldsymbol{\sigma}_{O\beta}^2+\mathbf{m}_{O\beta}^2\right)}{c_\alpha+c_\beta} - temp\_\mathbf{m}_{O\alpha}^2, \tag{62}$$

$$temp\_\mathbf{m}_{O\alpha} = \frac{c_\alpha\mathbf{m}_{O\alpha}+c_\beta\mathbf{m}_{O\beta}}{c_\alpha+c_\beta}, \tag{63}$$

$$\mathbf{m}_{O\alpha} = temp\_\mathbf{m}_{O\alpha}, \tag{64}$$

$$c_\alpha = c_\alpha + c_\beta, \tag{65}$$

M7.2) ทำการสร้างคลัสเตอร์ใหม่โดยบันทึกข้อมูล $(\mathbf{s}_i,\mathbf{v}_i)$ ลงในคลัสเตอร์ β และทำการเซตตัวแปรต่างๆดังต่อไปนี้

- เพิ่มจำนวนสมาชิกให้กับคลัสเตอร์ β โดย $c_\beta = 1$
- กำหนดค่าคลัสเตอร์เซ็นเตอร์และส่วนเบี่ยงเบนมาตรฐานของอินพุตสเปซสำหรับคลัสเตอร์ β โดย $\mathbf{m}_{I\beta} = \mathbf{s}_i,\ \boldsymbol{\sigma}_{I\beta} = 0$
- กำหนดค่าคลัสเตอร์เซ็นเตอร์และส่วนเบี่ยงเบนมาตรฐานของเอาท์พุตสเปซสำหรับคลัสเตอร์ β โดย $\mathbf{m}_{O\beta} = \mathbf{v}_i,\ \boldsymbol{\sigma}_{O\beta} = 0$

M7.3) คำนวณระยะห่างของแต่ละคลัสเตอร์ในเมตริกซ์ D_I และ D_O ใหม่ทั้งหมดอีกครั้ง ถ้ายังเหลือคู่ของกลุ่มคลัสเตอร์ที่ยังไม่ได้ทำการทดสอบให้ดำเนินการขั้นตอน M7 อีกครั้งโดยพิจารณาคู่ของกลุ่มคลัสเตอร์ต่อไปที่ยังไม่ได้ทำการทดสอบในอินพุตสเปซ

M7.4) ในกรณีที่ข้อมูล $(\mathbf{s}_i,\mathbf{v}_i)$ ที่นำมาคลัสเตอร์ไม่มีความสอดคล้องกับคลัสเตอร์ใด ๆ ที่มีอยู่ให้เพิ่มจำนวนกลุ่มของคลัสเตอร์ขึ้นใหม่อีกหนึ่งกลุ่มและดำเนินขั้นตอนดังต่อไปนี้

- เพิ่มจำนวนคลัสเตอร์ โดยกำหนดให้ $K = K+1$
- เพิ่มจำนวนสมาชิกให้กับคลัสเตอร์ใหม่ โดยกำหนดให้ $c_K = 1$
- กำหนดค่าคลัสเตอร์เซ็นเตอร์และส่วนเบี่ยงเบนมาตรฐานของอินพุตสเปซสำหรับคลัสเตอร์ K โดย $\mathbf{m}_{IK} = \mathbf{s}_i,\ \boldsymbol{\sigma}_{IK} = 0$
- กำหนดค่าคลัสเตอร์เซ็นเตอร์และส่วนเบี่ยงเบนมาตรฐานของเอาท์พุตสเปซ สำหรับคลัสเตอร์ K โดย $\mathbf{m}_{OK} = \mathbf{v}_i,\ \boldsymbol{\sigma}_{OK} = 0$
- คำนวณระยะห่างของแต่ละคลัสเตอร์เซ็นเตอร์ในเมตริกซ์ D_I และ D_O ใหม่ทั้งหมดอีกครั้ง

M8) Termination of clustering process

ถ้ายังเหลือข้อมูลที่ยังไม่ได้ทำการคลัสเตอร์ให้กลับไปขั้นตอนที่ M4 นอกจากนี้ให้ทำขั้นตอน M9 ต่อไป

M9) Final combination procedure

แบ่งเป็น 2 ขั้นตอนดังต่อไปนี้

M9.1) เป็นขั้นตอนในการตรวจสอบและกำจัดสิ่งรบกวนที่เกิดขึ้นซึ่งจะดูจากจำนวนสมาชิกข้อมูลแต่ละคลัสเตอร์ว่ามีจำนวนน้อยไปหรือไม่ เมื่อทำการเทียบกับจำนวนของสมาชิกของคลัสเตอร์อื่น ๆ โดยกำหนดค่าคงที่ (ε) ที่มีค่าน้อย ๆ เพื่อใช้ในการทดสอบโดยถ้าคลัสเตอร์ที่ทำการทดสอบมีจำนวนสมาชิกของข้อมูลน้อยกว่าค่าคงที่ (ε) จะทำการตรวจสอบว่าคลัสเตอร์ดังกล่าวมีความสอดคล้องกับคลัสเตอร์อันใดที่เหลืออยู่หรือไม่ ถ้าเกิดความสอดคล้องจะทำการรวมเป็นสมาชิกกันระหว่างคลัสเตอร์ ทำการคำนวณหาค่าคลัสเตอร์เซ็นเตอร์และส่วนเบี่ยงเบนมาตรฐานใหม่ แต่ไม่เกิดความสอดคล้องกับคลัสเตอร์อันใด ๆ ที่มีอยู่จะถือว่าคลัสเตอร์ดังกล่าวเป็นสิ่งรบกวนที่เกิดขึ้นก็จะทำการกำจัดทิ้ง ซึ่งขั้นตอนนำเสนอดังต่อไปนี้

ตารางที่ 5 MCA อัลกอริทึมขั้นตอนที่ M9.1

รายละเอียดอัลกอริทึม

FOR $r = 1, \ldots, K$

IF $c_r < \varepsilon$

$\beta = r$

FOR $\alpha = r + 1, \ldots, K$

IF มีความสอดคล้องระหว่างคลัสเตอร์ β และ α

ประมวลผล (59-65)

END

IF ไม่มีความสอดคล้องกับคลัสเตอร์ใด ๆ

ทำการกำจัดคลัสเตอร์ดังกล่าวทิ้ง โดยกำหนดให้ $K = K - 1$

END

M9.2) ในขั้นตอนนี้จะทำการทดสอบความคาบเกี่ยวกัน (overlapping) ของแต่ละคลัสเตอร์ เพื่อที่จะลดความซับซ้อนของข้อมูลให้กับโครงสร้าง SANFIS

ตารางที่ 6 ขั้นตอนที่ M9.2

รายละเอียดอัลกอริทึม

FOR $r = 1,\ldots,K$,

Set $\mathbf{s}_i = m_r$.

FOR $a = r+1,\ldots,K$,

ทำการคำนวณค่า $p_a(\mathbf{s}_i)$ ของคลัสเตอร์ a ด้วยสมการต่อไปนี้

$$p_a(\mathbf{s}_i) = \frac{\prod_{i=1}^{N} \mu_{A_i^j}(s_i)}{\sum_{j=1}^{J}\prod_{i=1}^{N} \mu_{A_i^j}(s_i)}. \qquad (66)$$

โดย $p_a(x) \in [0,1]$ คือค่าความคาบเกี่ยวของคลัสเตอร์ a

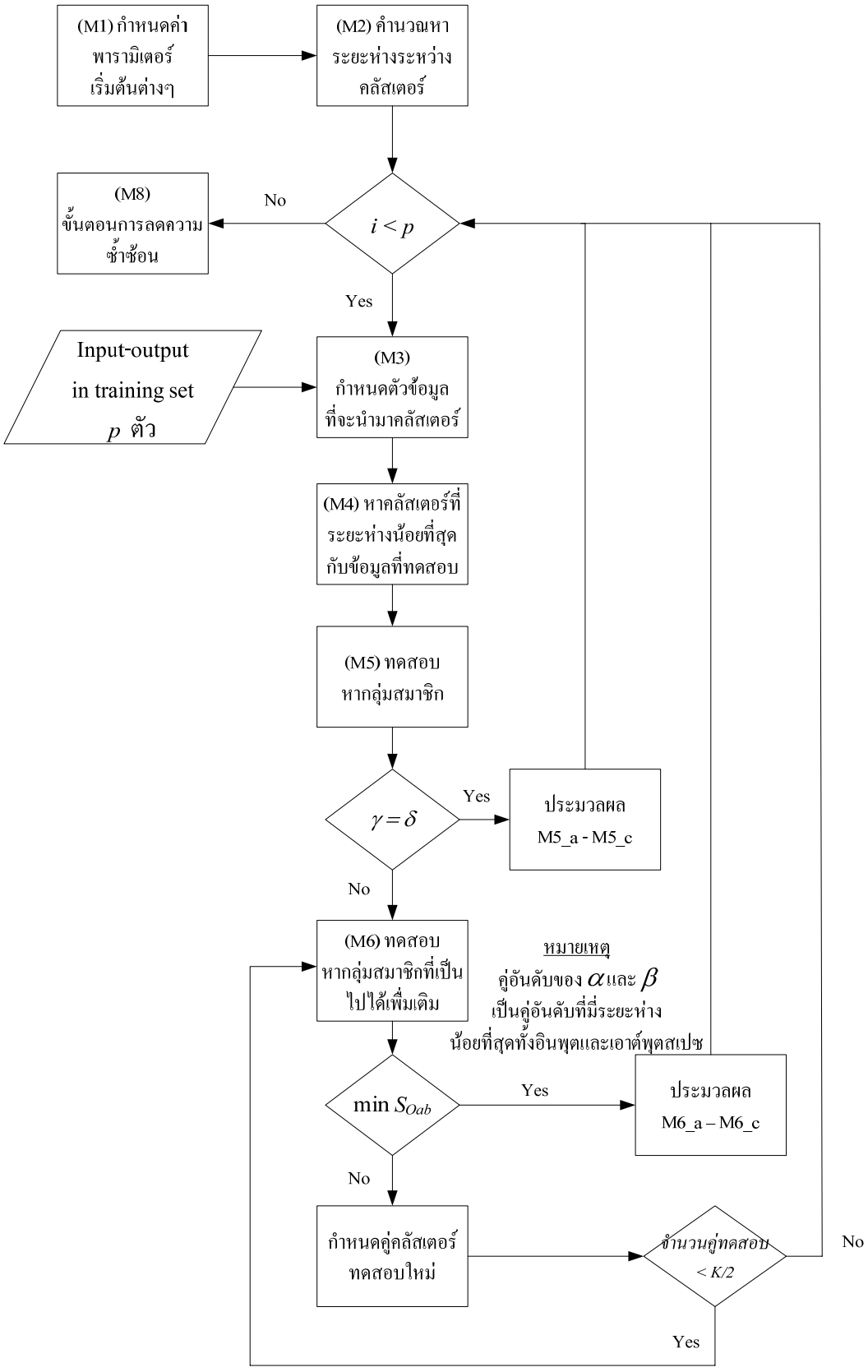
END

ทำการหา $\eta = \arg\max_{r+1\le a \le K} p_a(\mathbf{s})$ โดย η คือคลัสเตอร์ที่มีความคาบเกี่ยวกับคลัสเตอร์ r มากที่สุดเมื่อเทียบกับคลัสเตอร์อื่น ๆ หลังจากนั้นให้ทำการทดสอบความสอดคล้องของคลัสเตอร์ η และ r ซึ่งถ้าเกิดความสอดคล้องกันให้รวมคลัสเตอร์ทั้งสองเข้าด้วยกันโดยประมวลผลสมการ (49-55) และกำหนดให้ $K = K-1$

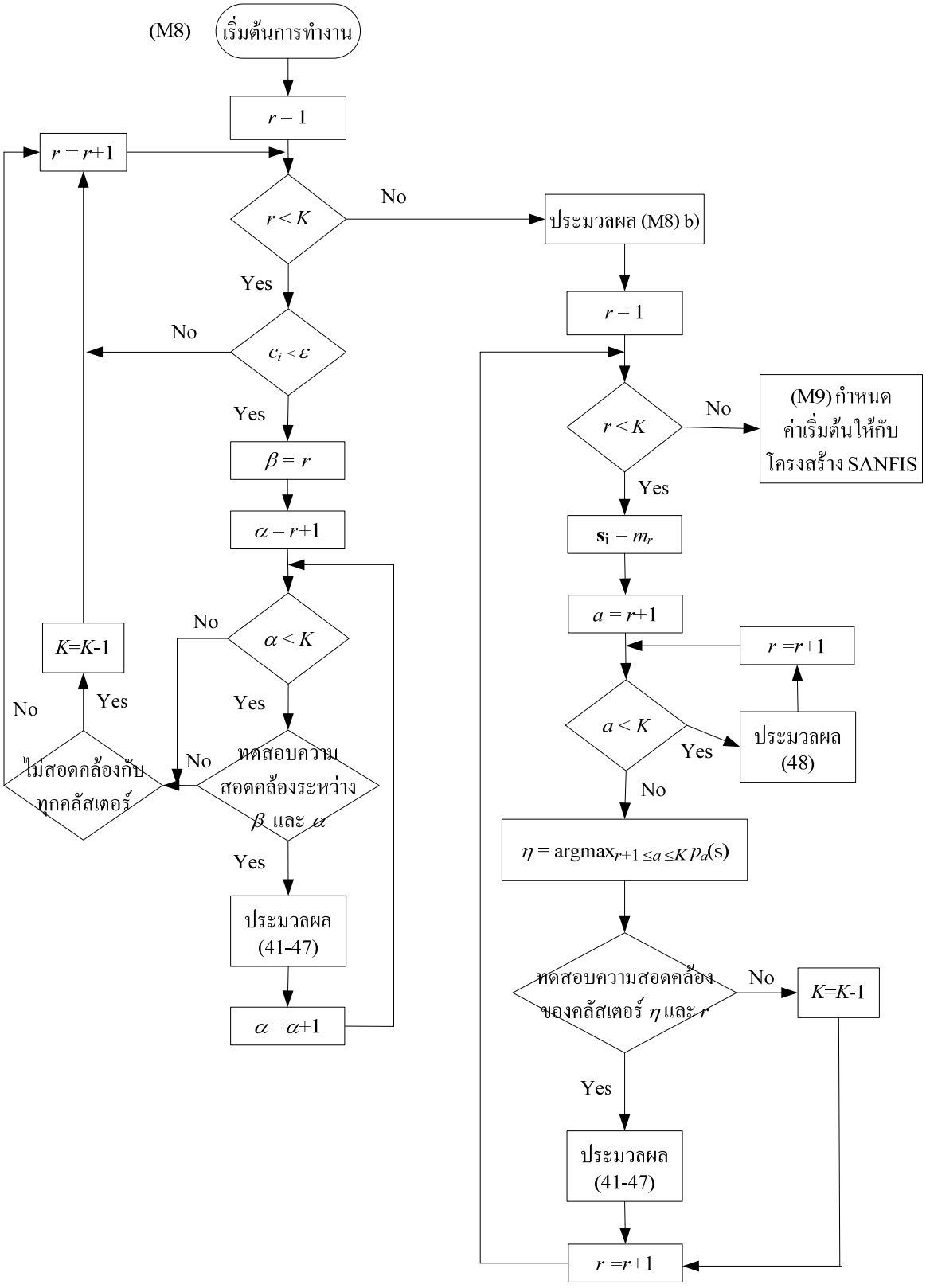
END

M10) SANFIS initialization

นำค่าพารามิเตอร์ที่ได้นำไปใช้เป็นค่าเริ่มต้นใน SANFIS โดยกำหนดให้ $J = C,\ \theta_k = m_{Ok}$ โดยที่ $k = 1,\ldots,M$ และ M คือจำนวนของเอาท์พุต



ภาพที่ 12 Flow chart ของ MCA cluster algorithm



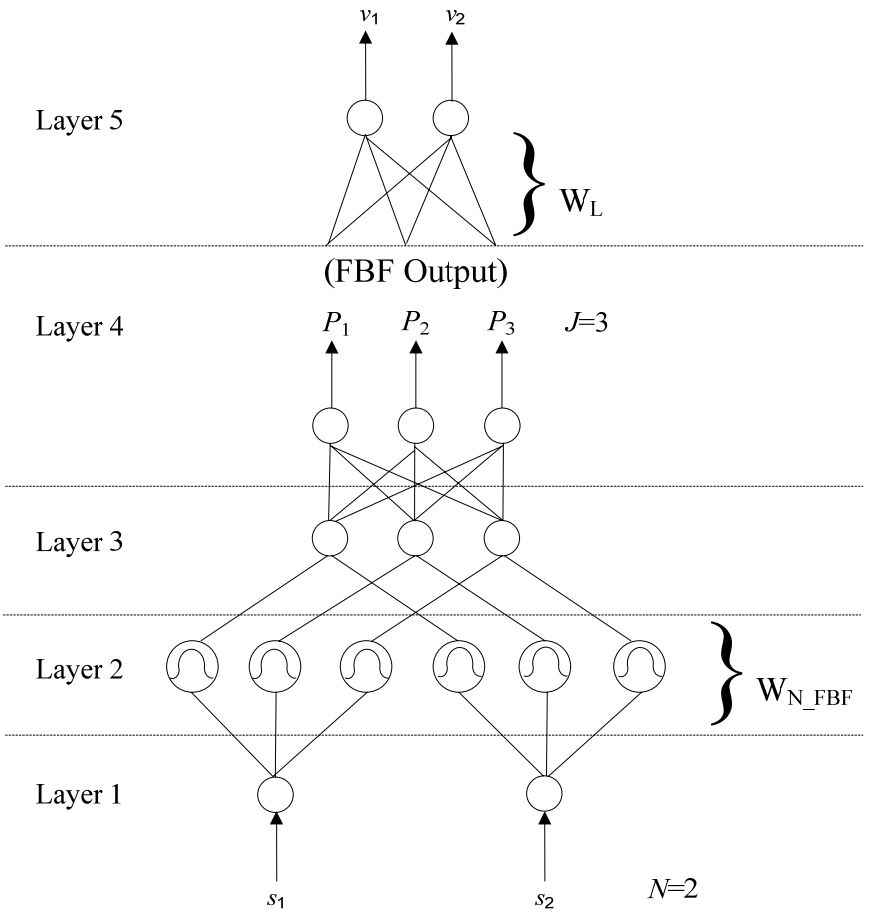
ภาพที่ 13 Flow chart ของ MCA cluster algorithm ขั้นตอน M9 และ M10

Fast Recursive Linear/Nonlinear Least-squares Optimization Algorithm

หลังจากที่ได้ค่าเริ่มต้นพารามิเตอร์ที่เหมาะสมสำหรับ SANFIS ด้วยการใช้อัลกอริทึม MCA แล้วพารามิเตอร์ต่าง ๆ ภายในโครงสร้าง SANFIS ยังจำเป็นต้องได้รับการปรับค่าเพื่อให้ SANFIS สามารถให้ได้ผลลัพธ์จากโมเดลใกล้เคียงกับผลลัพธ์ที่ต้องการมากที่สุด โดยฟังก์ชันหลัก (objective function) ในการปรับค่าพารามิเตอร์สามารถนำเสนอในรูปแบบฟังก์ชัน:

$$\min E\left(\mathrm{W}_{N\_FBF}(t), \mathrm{W}_{L}(t)\right), \tag{67}$$

พารามิเตอร์ของ SANFIS จะถูกพิจารณาให้เป็น weight ซึ่งการเทรนหรือการปรับพารามิเตอร์ตาม objective function จะแบ่งออกเป็นส่วนออกเป็น 2 ส่วนคือ non-linear weight ($\mathrm{W}_{N\_FBF}(t)$) และ linear weight ($\mathrm{W}_{L}(t)$) ดังรูปที่ต่อไปนี้



ภาพที่ 14 Non-linear weight ($\mathrm{W}_{N\_FBF}(t)$) และ linear weight ($\mathrm{W}_{L}(t)$) ของโครงสร้าง SANFIS

1) Non-linear weight ($\mathbf{W}_{N\_FBF}$)

$$\mathbf{W}_{N\_FBF}(t)=\begin{bmatrix} \mathbf{w}_{11} & \mathbf{w}_{21} & \cdots & \mathbf{w}_{N1} \\ \mathbf{w}_{12} & \mathbf{w}_{22} & \cdots & \mathbf{w}_{N2} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{w}_{1J} & \mathbf{w}_{2J} & \cdots & \mathbf{w}_{NJ} \end{bmatrix}, \tag{68}$$

โดยที่ $\mathbf{W}_{N\_FBF}(t)\in\mathbb{R}^{J\times 2n}$, N คือจำนวนของตัวแปรอินพุต, $p=1,\ldots,N$, J คือจำนวนกฎ, $j=1,\ldots,J$, $\mathbf{w}_{pj}(t)=\left[m_p^j,\sigma_p^j\right]^T$, m_p^j และ σ_p^j เป็นคลัสเตอร์เซ็นเตอร์และส่วนเบี่ยงเบนมาตรฐานของอินพุตสเปซที่ได้จากการประมวลผลอัลกอริทึม MCA

อัลกอริทึมที่ใช้ในการปรับค่าพารามิเตอร์ (m_p^j และ σ_p^j) ในส่วนของ non-linear weight จะใช้ Recursive Levenberg and Marquardt (L-M) อัลกอริทึม (Ngia and Sjoberg, 2000) ซึ่งในงานวิจัยชิ้นนี้จะไม่ทำการพิสูจน์ที่มาแต่จะนำเสนอในรูปสมการพร้อมใช้ดังต่อไปนี้

$$\mathbf{W}(t+1)=\mathbf{W}(t)-\mathbf{\Phi}(t+1)\mathbf{J}\left(\mathbf{W}(t)\right)\mathbf{e}\left(\mathbf{W}(t)\right), \tag{69}$$

$$\mathbf{\Phi}(t+1)=\frac{1}{\alpha}\left[\mathbf{\Phi}(t)-\mathbf{\Phi}(t)\mathbf{J}^*\left(\mathbf{W}(t)\right)\left(\mathbf{\Omega}(t)\right)^{-1}\mathbf{J}^{*T}\left(\mathbf{W}(t)\right)\mathbf{\Phi}(t)\right], \tag{70}$$

โดย α คือ forgetting factor; $0.95<\alpha<1$

$$\mathbf{\Omega}(t)=\mathbf{J}^{*T}\left(\mathbf{W}(t)\right)\mathbf{\Phi}(t)\mathbf{J}^*\left(\mathbf{W}(t)\right)+\alpha\mathbf{\Lambda}^*, \tag{71}$$

โดยที่ $\mathbf{\Omega}(t)\in\mathbb{R}^{2\times 2}$, $\mathbf{\Phi}(0)=\rho\mathbf{I}$; ρ เป็นค่าคงที่ที่มีค่ามากๆ ยกตัวอย่างเช่น 1000 และ

$$\mathbf{\Lambda}^* = \begin{bmatrix} 1 & 0 \\ 0 & N\mu \end{bmatrix}, \tag{72}$$

μ เป็นค่าคงที่ที่มีค่าน้อยๆ ยกตัวเช่น 0.0001 และ N คือจำนวนอินพุต

$$\mathbf{J}^*\left(\mathbf{W}(t)\right) = \begin{bmatrix} & & 0 \\ & & \vdots \\ & & 0 \\ \mathbf{J} & \left(\mathbf{W}(t)\right) & 1 \\ & & 0 \\ & & \vdots \\ & & 0 \end{bmatrix} \leftarrow \left(\left(t \bmod N\right)+1\right)\text{th position}, \tag{73}$$

โดยที่ $\mathbf{J}^*\left(\mathbf{W}(t)\right) \in \mathbb{R}^{N\times 2}$

กำหนดให้

$$\mathbf{e}\left(\mathbf{W}(t)\right) = e_j\left(\mathbf{w}_{pj}(t)\right), \tag{74}$$

$$e_j\left(\mathbf{w}_{pj}(t)\right) = P_j\left(\mathbf{w}_{pj}(t)\right) - P_j^k, \tag{75}$$

โดย $P_j\left(\mathbf{w}_{pj}(t)\right)$ คือค่าเอาท์พุตของแต่ละ fuzzy basis function (FBFs), $j = 1,\ldots,J$

$$P_j^k = \begin{cases} 1 & \text{if } j\text{th FBF is connected to node } l^* \text{ Where } l^* = \arg\min\limits_{1\le l\le L}\left(v_k^d - \theta_j\right) \\ 0 & \text{Otherwise} \end{cases} \tag{76}$$

โดยที่ v_k^d คือค่าเอาต์พุตที่ต้องการ (desired output) ลำดับที่ k, $k = 1,\ldots,M$, M คือจำนวนเอาท์พุตและ θ_j คือค่ากลางของคลัสเตอร์ j, $j = 1,\ldots,J$

จุดประสงค์ของสมการ (76) คือการดูความเกี่ยวเนื่องกันระหว่างค่าเอาต์พุตที่ต้องการ (v_k^d) กับค่า θ_j เพื่อใช้กำหนดค่า P_j^k โดยค่า P_j^k จะเท่ากับ 1 ถ้า $v_k^d - \theta_j$ มีค่าน้อยที่สุดเมื่อเทียบกับ P_j^k อื่นๆ

เนื่องด้วยขั้นตอนการปรับค่าพารามิเตอร์ (weight) ของแต่ละ FBF สามารถทำได้โดยไม่กระทบกับฟังก์ชันหลัก (objective function) เพราะฉะนั้นการปรับค่าพารามิเตอร์ของแต่ละ FBF สามารถแยกออกจากกันได้ ดังนั้นกำหนดให้ jacobian matrix ของแต่ละ FBF ใช้รูปแบบสมการดังต่อไปนี้

$$\mathbf{J}_{P_j} = \left[\mathbf{J}_{1j}, \ldots, \mathbf{J}_{Nj} \right]^T, \tag{77}$$

$$\mathbf{J}_{pj} = \frac{\partial e_j\left(\mathbf{w}_{pj}\right)}{\partial \mathbf{w}_{pj}},$$

$$\mathbf{J}_{pj} = \frac{\partial\left(P_j\left(\mathbf{w}_{pj}(t)\right) - P_j^d\right)}{\partial \mathbf{w}_{pj}}, \tag{78}$$

การหา closed form $\mathbf{J}_{pj}$ สำหรับ P_j นำเสนอดังต่อไปนี้

$$P_j = \frac{\exp\left[-\frac{1}{2}\left(\frac{s_1 - m_p^j}{\sigma_p^j}\right)^2\right] \times \ldots \times \exp\left[-\frac{1}{2}\left(\frac{s_N - m_N^j}{\sigma_N^j}\right)^2\right] \rightarrow (\mathrm{A})}{\left(\exp\left[-\frac{1}{2}\left(\frac{s_1 - m_1^1}{\sigma_1^1}\right)^2\right] \times \ldots \times \exp\left[-\frac{1}{2}\left(\frac{s_n - m_n^1}{\sigma_n^1}\right)^2\right]\right) + \ldots + \left(\exp\left[-\frac{1}{2}\left(\frac{s_1 - m_1^J}{\sigma_1^1}\right)^2\right] \times \ldots \times \exp\left[-\frac{1}{2}\left(\frac{s_N - m_N^J}{\sigma_N^J}\right)^2\right]\right) \rightarrow (\mathrm{B})},$$

$$\frac{\partial P_j}{\partial m_p^j} = \frac{(B)(A)\frac{\left(s_p - m_p^j\right)}{\left(\sigma_p^j\right)^2} - (A)(A)\frac{\left(s_p - m_p^j\right)}{\left(\sigma_p^j\right)^2}}{(B)^2},$$

$$= P_j \frac{\left(s_p - m_p^j\right)}{\left(\sigma_p^j\right)^2} - \left(P_j\right)^2 \frac{\left(s_p - m_p^j\right)}{\left(\sigma_p^j\right)^2},$$

$$= \left(P_j - \left(P_j\right)^2\right) \frac{\left(s_p - m_p^j\right)}{\left(\sigma_p^j\right)^2}. \tag{79}$$

$$\frac{\partial P_j}{\partial \sigma_p^j} = \frac{(B)(A)\frac{\left(s_p - m_p^j\right)^2}{\left(\sigma_p^j\right)^3} - (A)(A)\frac{\left(s_p - m_p^j\right)^2}{\left(\sigma_p^j\right)^3}}{(B)^2},$$

$$= P_j \frac{\left(s_p - m_p^j\right)^2}{\left(\sigma_p^j\right)^3} - \left(P_j\right)^2 \frac{\left(s_p - m_p^j\right)^2}{\left(\sigma_p^j\right)^3},$$

$$= \left(P_j - \left(P_j\right)^2\right) \frac{\left(s_p - m_p^j\right)^2}{\left(\sigma_p^j\right)^3}. \tag{80}$$

จากสมการข้างต้นเป็น closed form ของ สำหรับ P_j :

$$\mathbf{J}_{Pj} = \begin{bmatrix} \mathbf{J}m_{Pj}^T \\ \mathbf{J}\sigma_{Pj}^T \end{bmatrix} = \begin{bmatrix} \frac{\partial P_j}{\partial m_1^j} & \frac{\partial P_j}{\partial m_2^j} & \cdots & \frac{\partial P_j}{\partial m_N^j} \\ \frac{\partial P_j}{\partial \sigma_1^j} & \frac{\partial P_j}{\partial \sigma_2^j} & \cdots & \frac{\partial P_j}{\partial \sigma_N^j} \end{bmatrix}$$

$$= \begin{bmatrix} \left(P_j - \left(P_j\right)^2\right)\frac{\left(s_1 - m_1^j\right)}{\left(\sigma_1^j\right)^2} & \left(P_j - \left(P_j\right)^2\right)\frac{\left(s_2 - m_2^j\right)}{\left(\sigma_2^j\right)^2} & \cdots & \left(P_1 - \left(P_1\right)^2\right)\frac{\left(s_N - m_N^j\right)}{\left(\sigma_N^j\right)^2} \\ \left(P_j - \left(P_j\right)^2\right)\frac{\left(s_1 - m_1^j\right)^2}{\left(\sigma_1^j\right)^3} & \left(P_j - \left(P_j\right)^2\right)\frac{\left(s_2 - m_2^j\right)^2}{\left(\sigma_2^j\right)^3} & \cdots & \left(P_1 - \left(P_1\right)^2\right)\frac{\left(s_N - m_N^j\right)^2}{\left(\sigma_N^j\right)^3} \end{bmatrix}. \tag{81}$$

เนื่องด้วยพารามิเตอร์ที่ทำการปรับในส่วนของ non-linear weight นั้นประกอบด้วยพารามิเตอร์ 2 ตัว คือ m_p^j และ σ_p^j จากการหา jacobian matrix ของแต่ละ FBF ($\mathbf{J}_{P_j}$) ประกอบด้วย 2 เวคเตอร์ คือเวคเตอร์ $\mathbf{J}m_{Pj}$ และ $\mathbf{J}\sigma_{Pj}$ โดย $\mathbf{J}m_{Pj}$ เป็น jacobian matrix สำหรับการปรับพารามิเตอร์ m_p^j และ $\mathbf{J}\sigma_{Pj}$ เป็น jacobian matrix สำหรับปรับพารามิเตอร์ σ_p^j เพราะฉะนั้นขั้นตอนในการปรับค่าพารามิเตอร์ของแต่ FBF ควรแยกการประมวลผล โดยเริ่มแรกทำการปรับพารามิเตอร์ m_p^j ด้วย $\mathbf{J}m_{Pj}$ ก่อนโดยกำหนดให้ $\mathbf{J}\ \left(\mathbf{W}(t)\right)=\mathbf{J}m_{Pj}$ แล้วทำการประมวลผล L-M อัลกอริทึมหลังจากนั้นค่อยทำการปรับพารามิเตอร์ σ_p^j โดยกำหนดให้ $\mathbf{J}\ \left(\mathbf{W}(t)\right)=\mathbf{J}\sigma_{Pj}$ ตามลำดับ

2) Linear weight ($\mathbf{W}_L(t)$)

เนื่องด้วยค่าพารามิเตอร์ (weight) ในโมเดลแบ่งออกเป็น 2 ส่วนด้วยกันเพราะฉะนั้นเราจึงเริ่มต้นด้วยการปรับค่าพารามิเตอร์ของ non-linear weight ($\mathbf{W}_{N\_FBF}$) ในแต่ละ FBF ก่อนแล้วจึงค่อยทำการปรับค่าพารามิเตอร์ในส่วนของ linear weight:

$$\mathbf{W}_L(t)=\left[\theta_1,\theta_2,\ldots,\theta_J\right]^T, \tag{82}$$

โดยที่ $\mathbf{W}_L(t)\in\mathbb{R}^J$, $\theta_j;\ j=1,\ldots,J$ เป็นคลัสเตอร์เซ็นเตอร์ของเอาต์พุตสเปซที่ได้จากการใช้อัลกอริทึม MCA

หลังจากทำการปรับพารามิเตอร์ของ non-linear weight ในส่วนของ linear weight สามารถกำหนดฟังก์ชันหลัก (objective function) ในปรับค่าพารามิเตอร์ได้ดังนี้

$$\min E\left(\mathbf{W}_{N\_FBF}^{*},\mathbf{W}_L(t)\right)$$

$$=\frac{1}{2}\sum_{k=1}^{M}\left(v_k(t)-v_k^d(t)\right)^2.$$

$$= \frac{1}{2}\sum_{k=1}^{M}\left(\mathbf{P}_k^T\left(\mathbf{X_k}, \mathbf{W}^*_{N\_FBF}(t)\right)\mathbf{W}_L(t) - v_k^d(t)\right)^2. \qquad (83)$$

โดย $\mathbf{W}^*_{N\_FBF}$ คือค่าพารามิเตอร์ (weight) หลังจากที่ได้ประมวลผล L-M อัลกอริทึม, $v_k(t)$ คือเอาต์พุตที่ได้จากโมเดล, $v_k^d(t)$ คือเอาต์พุตที่ต้องการ (desired output), $\mathbf{P}_k(t)$ คือเวคเตอร์เอาต์พุตของแต่ละ FBF (เลเยอร์ 4) ซึ่งใช้ $\mathbf{W}^*_{N\_FBF}$ ในการประมวลผล

ส่วนของ linear weight จะใช้ recursive least squares (RLS) (Wang and Lee, 2001) อัลกอริทึมในการปรับค่าพารามิเตอร์ (θ) สำหรับสมการพร้อมใช้ของ RLS มีดังต่อไปนี้

$$\mathbf{W}_L(t+1) = \mathbf{W}_L(t) + \mathbf{\Phi}(t+1)\mathbf{P}_k(t)\left[v_k^d(t) - v_k(t)\right], \qquad (84)$$

$$\mathbf{\Phi}(t+1) = \frac{1}{\alpha}\left[\mathbf{\Phi}(t) - \frac{\mathbf{\Phi}(t)\mathbf{P}_k(t)\mathbf{P}_k^T(t)\mathbf{\Phi}(t)}{\alpha + \mathbf{P}_k^T(t)\mathbf{\Phi}(t)\mathbf{P}_k(t)}\right], \qquad (85)$$

โดย $\mathbf{\Phi}(0) = \rho\mathbf{I}$; ρ เป็นค่าคงที่ที่มีค่ามากๆ ยกตัวอย่างเช่น 1000

8. การหาตำแหน่งหุ่นยนต์จากภาพ

เนื่องจากงานวิจัยนี้จะต้องทำการทดสอบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คจริงซึ่งข้อมูลตำแหน่งของหุ่นยนต์จะได้จากการประมวลผลภาพ (image processing) แต่วิธีการที่ใช้ในการหาตำแหน่งหุ่นยนต์นั้นจะต้องมีความเร็วสูงเนื่องจากการควบคุมหุ่นยนต์เป็นงาน real-time

วิธีการประมวลผลภาพเพื่อหาตำแหน่งหุ่นยนต์ของงานวิจัยนี้ได้ปรับปรุงและพัฒนาต่อมาจาก CMVision Color Segmentation (Bruce, J. and Veloso, M, 2002) ซึ่งในปัจจุบันถือว่าเป็นวิธีการในการหากลุ่มก้อนสี (region) ที่มีความเร็วสูงที่สุดโดยขั้นตอนในการหาตำแหน่งหุ่นยนต์จากภาพมีทั้งหมด 5 ขั้นตอนดังต่อไปนี้

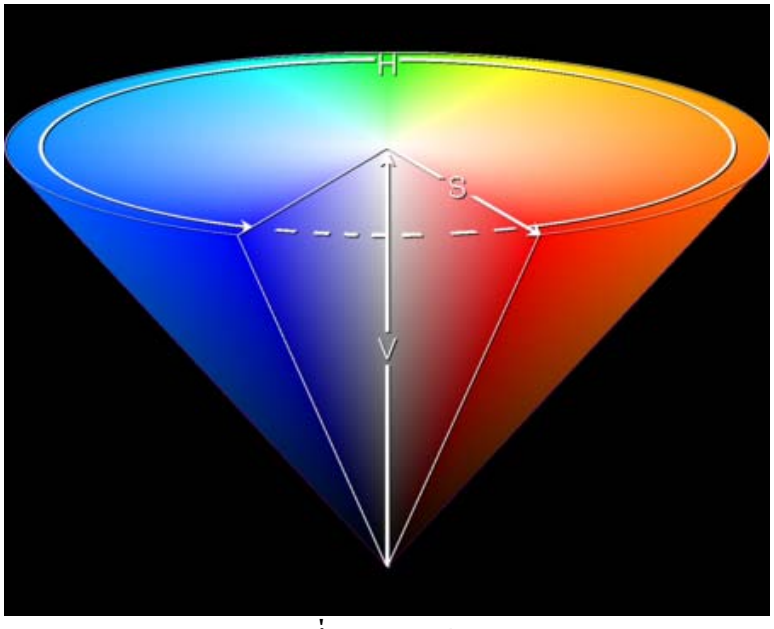
การแปลงระบบสี (color space transformation)
การระบุสี (color threshold)
การเชื่อมต่อพิกเซลที่มีคุณสมบัติเหมือนกัน (connected components)
การคัดเลือกกลุ่มสี (extracting region information)
การรู้จำแบบรูปของหุ่นยนต์ (pattern recognition)

4.1 การแปลงระบบสี (color space transformation)

มาตรฐานของภาพสีที่ใช้อยู่ในปัจจุบันมีอยู่หลายระบบด้วยกัน ทั้งนี้จะขึ้นอยู่กับการนำไปใช้ แต่โดยทั่วไปแล้วทุกมาตรฐานจะมีแนวคิดเดียวกันคือ การแทนจุดสีด้วยจุดที่อยู่ภายในสเปซ (space) 3 มิติ โดยจะมีแกนอ้างอิงสำหรับจุดสีนั้นในสเปซซึ่งแต่ละแกนจะมีความเป็นอิสระต่อกัน ตัวอย่างเช่นในระบบ RGB จะมีแกนสีคือ แกนสีแดง เขียว และน้ำเงินในระบบ HLS จะมีแกนเป็น ค่าสี (hue) ความสว่าง (lightness) และความบริสุทธิ์ของสี (saturation) และตัวอย่างระบบสีที่นิยมใช้กันได้แก่ ระบบ RGB HSV (Hue Saturation Value) และ HLS (Hue Lightness Saturation)

สำหรับในงานวิจัยนี้ได้เลือกใช้ระบบสี HSV เพราะว่าเป็นระบบสีที่ทนต่อการรบกวนของแสงมากที่สุดหรือในอีกมุมหมายถึงแสงสว่างจะมากหรือน้อยจะไม่ส่งผลกระทบกับระบบสีประเภทนี้ แต่ปัญหาที่เกิดขึ้นตามมาคือ ภาพที่ได้มาจากกล้องเป็นภาพที่มีระบบสี RGB โดยมีขนาด

640 x 480 พิกเซล ซึ่งถ้าจะแปลงจากระบบสี RGB เป็นระบบสี HSV ต้องใช้สมการที่ (86) ซึ่งถ้าทำทั้งภาพจะใช้เวลาทั้งหมดประมาณ 45 มิลลิวินาที ซึ่งเป็นเวลาที่มากกว่า sampling time ของระบบควบคุมหุ่นยนต์ซึ่งอยู่ที่ 60 ครั้งต่อวินาที หรือ คาบเวลาเท่ากับ 16.67 มิลลิวินาที งานวิจัยนี้จึงแก้ปัญหาโดยใช้วิธีการแปลงระบบสีโดยใช้ look-up table (LUT) ซึ่งเป็นวิธีการที่สร้างตารางการแปลงระบบสีเก็บไว้ในหน่วยความจำก่อนอยู่แล้ว ผลลัพท์ที่ได้ดีมากโดยวิธีการ LUT ใช้เวลาในการแปลงระบบสีจาก RGB เป็นระบบสี HSV ทั้งภาพอยู่ที่ประมาณ 4 มิลลิวินาที



ภาพที่ 15 ระบบสี HSV

การแปลงสีจากระบบสี RGB เป็น HSV แสดงในสมการต่อไปนี้

$$V = \max(R, G, B),$$

$$S = \frac{V - \min(R, G, B)}{V},$$

$$H = \begin{cases} \dfrac{(G-B)\times 60}{S}, & V = R, \\ 180 + \dfrac{(B-R)\times 60}{S}, & V = G, \\ 240 + \dfrac{(R-G)\times 60}{S}, & V = B, \end{cases} \qquad (86)$$

4.2 การระบุสี (color threshold)

การระบุสี คือการชี้ชัดว่าข้อมูลในแต่ละพิกเซลนั้นเป็นสีอะไร ซึ่งคุณสมบัติที่จะบอกว่าพิกเซลนั้นเป็นสีอะไรจะเป็นไปตามระบบโมเดลสี ยกตัวอย่างเช่น ระบบสี HSV อยู่ภายใน Space 3 มิติ จะมีช่วงของสีที่ตัด 6 ช่วงหรือหมายถึงสีที่สนใจจะต้องมีคุณสมบัติดังนี้

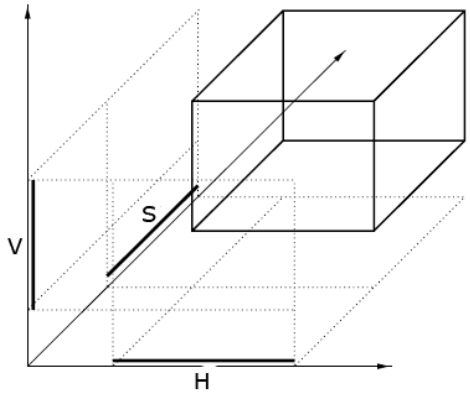
- ค่า Hue มากกว่า Min(H) และ น้อยกว่า Max(H)
- ค่า Saturate มากกว่า Min(S) และ น้อยกว่า Max(S)
- ค่า Value มากกว่า Min(V) และ น้อยกว่า Max(V)

หรือจะเขียนเป็น pseudocode ได้ตามตารางต่อไปนี้

ตารางที่ 7 pseudocode color threshold

รายละเอียด pseudocode

```
if ((H >= Hlowerthresh)
AND (H <= Hupperthresh)
AND (S >= Slowerthresh)
AND (S <= Supperthresh)
AND (V >= Vlowerthresh)
AND (V <= Vupperthresh))
pixel_color = color_class; //Pixel นี้อยู่ใน Class ของสีนี้
```



ภาพที่ 16 แสดงตำแหน่งของช่วงสีที่อยู่ใน space ของระบบสี HSV

เช่น สมมุติว่าถ้าเรากำหนดค่า hue ของสีส้มต้องอยู่ในช่วง 4 ถึง 20 , ค่า saturate ต้องอยู่ในช่วง 100 ถึง 150 และค่า value ต้องอยู่ในช่วง 200 ถึง 250 ถ้าข้อมูลในพิกเซลใดๆมีค่า hue เท่ากับ 6, ค่า saturate เท่ากับ 120 และค่า value เท่ากับ 220 จะสามารถระบุได้ว่าพิกเซลนี้เป็นสีส้ม

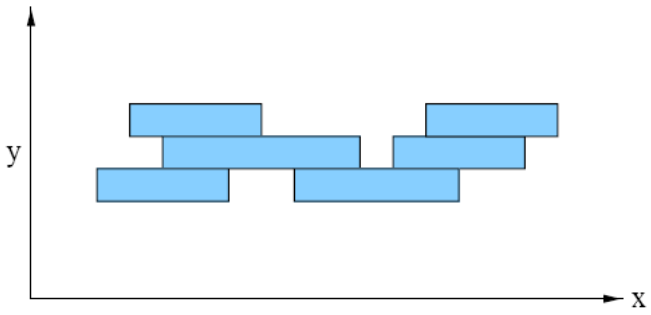
4.3 การเชื่อมต่อพิกเซลที่มีคุณสมบัติเหมือนกัน (connected components)

หลังจากขั้นตอนการระบุสีเราจะสามารถกำหนดได้ว่าภาพที่เข้ามาแต่ละพิกเซลเป็นสีอะไรบ้างและอยู่ในเป็นสีที่เราต้องการหรือเปล่า แต่ข้อมูลที่ได้มายังไม่สามารถใช้ในการประมวลผลหาตำแหน่งของหุ่นยนต์ ซึ่งจะต้องดำเนินการขั้นตอนการเชื่อมต่อพิกเซลที่มีคุณสมบัติเหมือนกันก่อน (connected components)

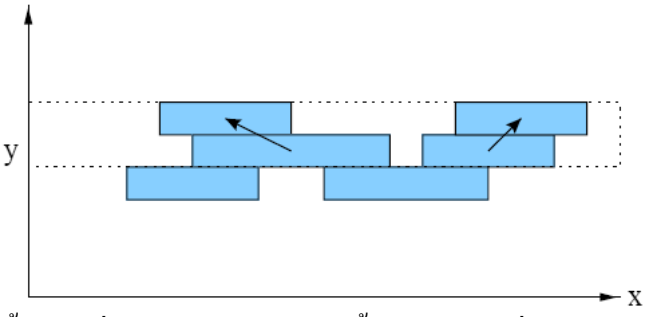
การเชื่อมต่อพิกเซลที่มีคุณสมบัติเหมือนกันคือวิธีการในการทำให้พิกเซลที่เป็นสีเดียวกันและอยู่ติดกันมาทำการเชื่อมต่อเป็นวัตถุเดียวกันโดยจะเรียกวัตถุนี้ว่ากลุ่มสี (region)

การเชื่อมต่อกลุ่มสีสามารถแบ่งเป็นขั้นตอนได้ 2 ขั้นตอน ขั้นตอนแรกคือการใช้อัลกอริทึม Run Length Encode (RLE) (Bruce, J. and Veloso, M, 2002) ซึ่งเป็นวิธีการที่ใช้กันทั่วไปในการค้นหาพิกเซลที่ติดกัน โดยงานวิจัยนี้ได้ทำการปรับแต่งอัลกอริทึมนี้ใหม่โดยจะทำการค้นหาพิกเซลที่ติดกันในแนวนอน (horizontal) อย่างเดียวก่อนโดยจะเรียกกลุ่มของพิกเซลเดียวกันในแถวเดียวกันว่า run (Bruce, J. and Veloso, M, 2002) ซึ่งสาเหตุสำคัญที่ยังไม่เชื่อมต่อพิกเซลในแนวตั้งเนื่องจากการค้นหาข้อมูลทางด้านแนวตั้งลำบากในการเข้าถึงข้อมูลเพราะจริงๆแล้วข้อมูลของภาพในหน่วยความจำเก็บเป็น array มิติเดียว ซึ่งแน่นอนว่าถ้าค้นหาพิกเซลในแนวตั้งจะต้องเกิดการคำนวณตำแหน่งของพอยเตอร์ relative กับขนาดภาพซึ่งเสียเวลามาก

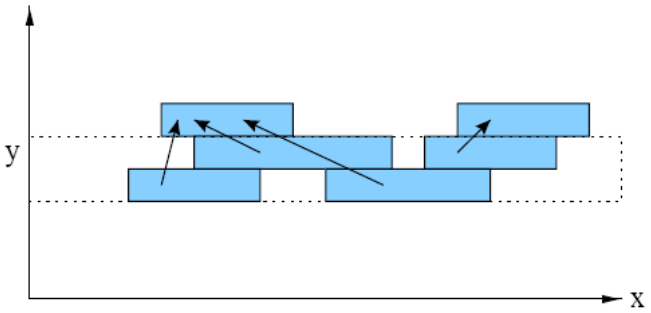
ขั้นตอนต่อมาคือการเชื่อมต่อ runs โดยใช้วิธีการ tree-based union find (Bruce, J. and Veloso, M, 2002) ซึ่งจะทำการค้นหาและเชื่อมต่อ run ที่ติดกันแนวตั้งเท่านั้น ซึ่งมีขั้นตอนดังภาพต่อไปนี้



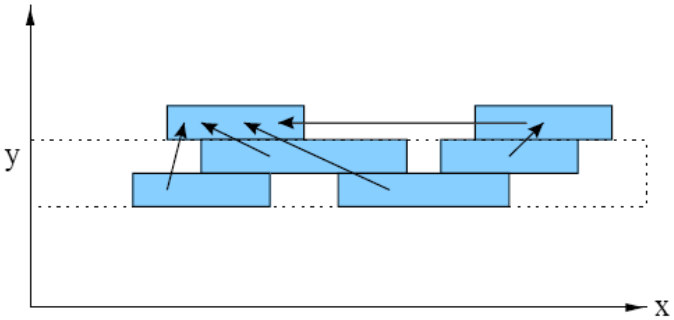
ภาพที่ 17 ขั้นตอนที่ 1: ข้อมูลดิบที่ได้หลังจากอัลกอริทึม RLE



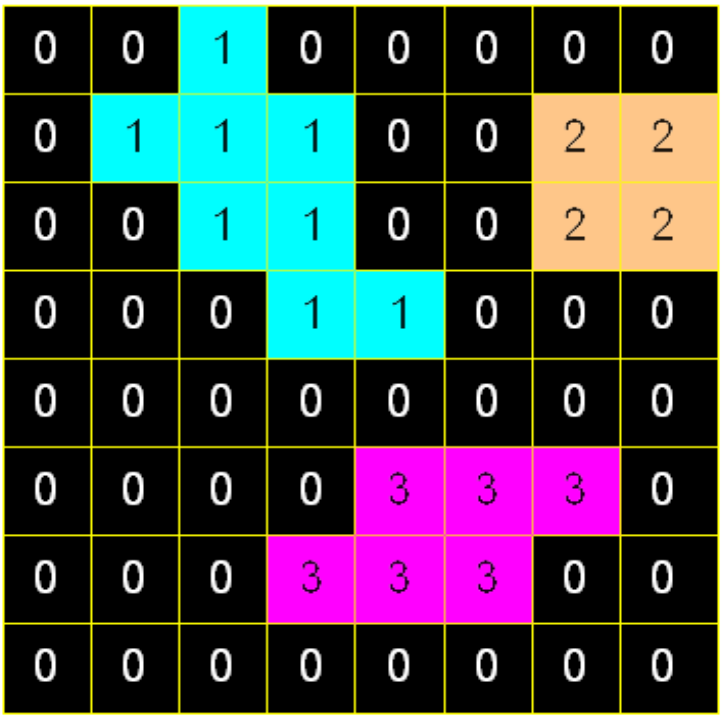
ภาพที่ 18 ขั้นตอนที่ 2: ค้นหา run ในแนวตั้งและทำการเชื่อมต่อกันเป็น Region



ภาพที่ 19 ขั้นตอนที่ 3: เลื่อนลงมาทีละแถวแล้วทำเหมือนขั้นตอนที่ 2



ภาพที่ 20 ขั้นตอนที่ 4: ในกรณีที่พบว่า region เกิดซ้อนทับกัน ให้ลบ Region เดิมทิ้งแล้วสร้าง region ขึ้นมาใหม่



ภาพที่ 21 ตัวอย่างการเชื่อมต่อพิกเซลที่มีคุณสมบัติเหมือนกัน

4.4 การคัดเลือกกลุ่มสี (extracting region information)

หลังจากได้ regions ทั้งหมดแล้ว นำ regions ที่ได้มาหาคุณสมบัติต่างๆ ยกตัวอย่างเช่น ตำแหน่งจุดศูนย์กลาง ขนาด และ สีเฉลี่ย เป็นต้น และในขั้นตอนนี้ยังทำการคัดเลือก region ที่ไม่พึงประสงค์ทิ้ง ซึ่งอาจจะเป็นการกำหนดคุณสมบัติว่าถ้าขนาดของ region น้อยกว่าค่าค่าหนึ่งจะถือว่า region นั้นเกิดจากสัญญาณรบกวน (noise) ให้ทำการลบ region นั้นทิ้งเป็นต้น

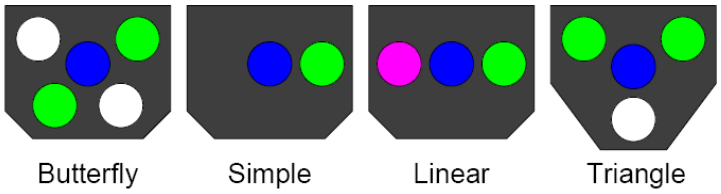
4.5 การรู้จำแบบรูปของหุ่นยนต์ (pattern recognition)

ขั้นตอนนี้เป็นขั้นตอนการหาตำแหน่งทั้งตำแหน่งและมุมของหุ่นยนต์โดยรูปแบบที่งานวิจัยนี้เลือกใช้บ่งบอกตำแหน่งของหุ่นยนต์เป็นแบบ butterfly (Bruce, J. and Veloso, M, 2002) จากการวิเคราะห์พบว่ารูปแบบนี้ให้ตำแหน่งและมุมผิดพลาดน้อยที่สุดดังภาพที่ 23

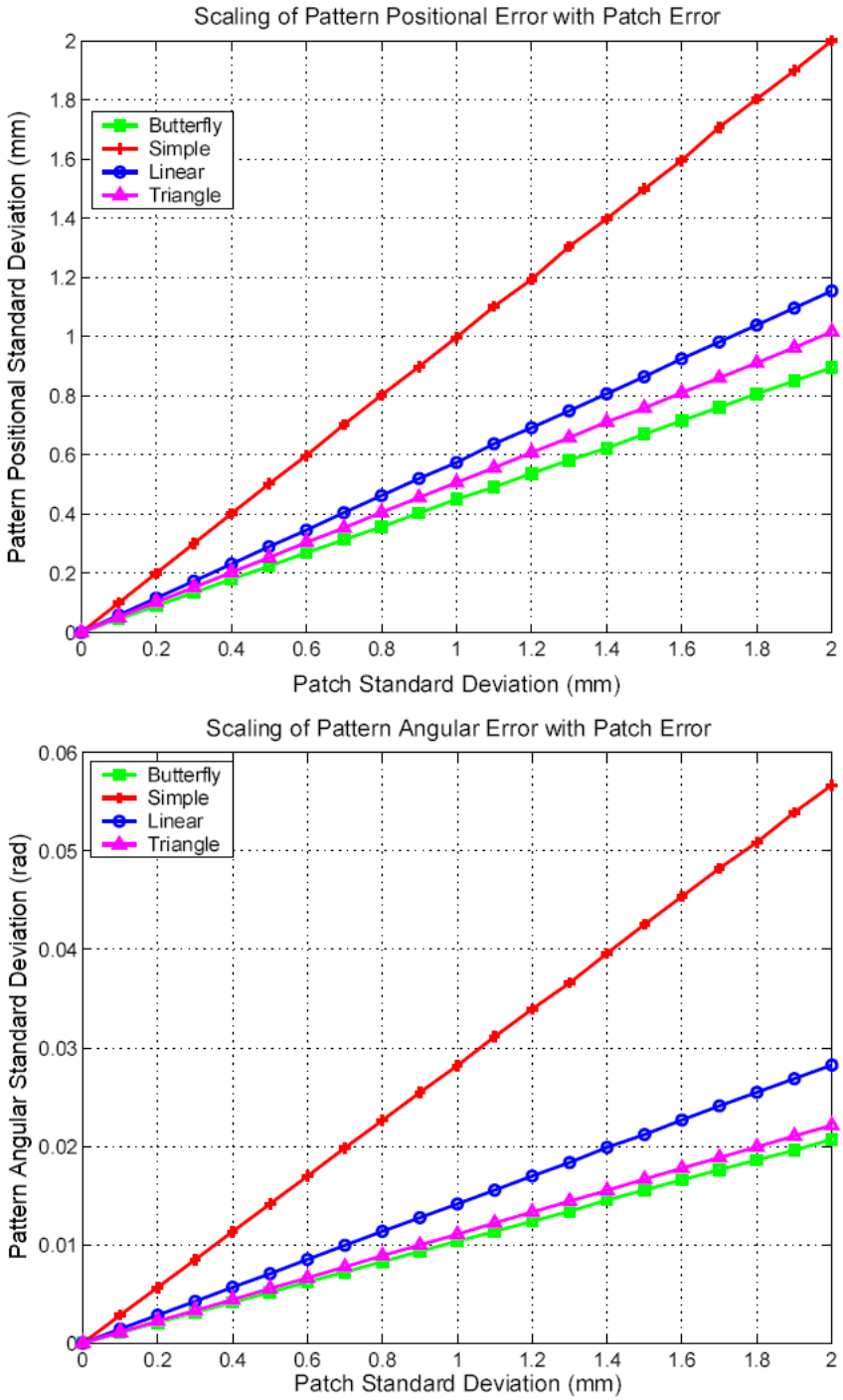
รูปแบบ butterfly ยังมีข้อดีอื่นๆอีกยกตัวอย่างเช่นรูปแบบนี้สามารถชี้ชัดหุ่นยนต์ได้จำนวนมากที่สุดในกรณีที่เราจะหาตำแหน่งของหุ่นยนต์หลายตัวพร้อมกันซึ่งจำนวนหุ่นยนต์ที่แต่ละรูปแบบสามารถชี้ชัดได้แสดงในตารางต่อไปนี้

ตารางที่ 8 จำนวนหุ่นยนต์ที่แต่ละรูปแบบสามารถชี้ชัดได้มากที่สุด

Pattern	2 colors	3 colors	4 colors	n colors
Buterfly	16	64	256	4^n
Simple	2	3	4	n
Linear	1	3	6	$n.(n-1)/2$
Triangle	2	6	12	$n.(n-1)$



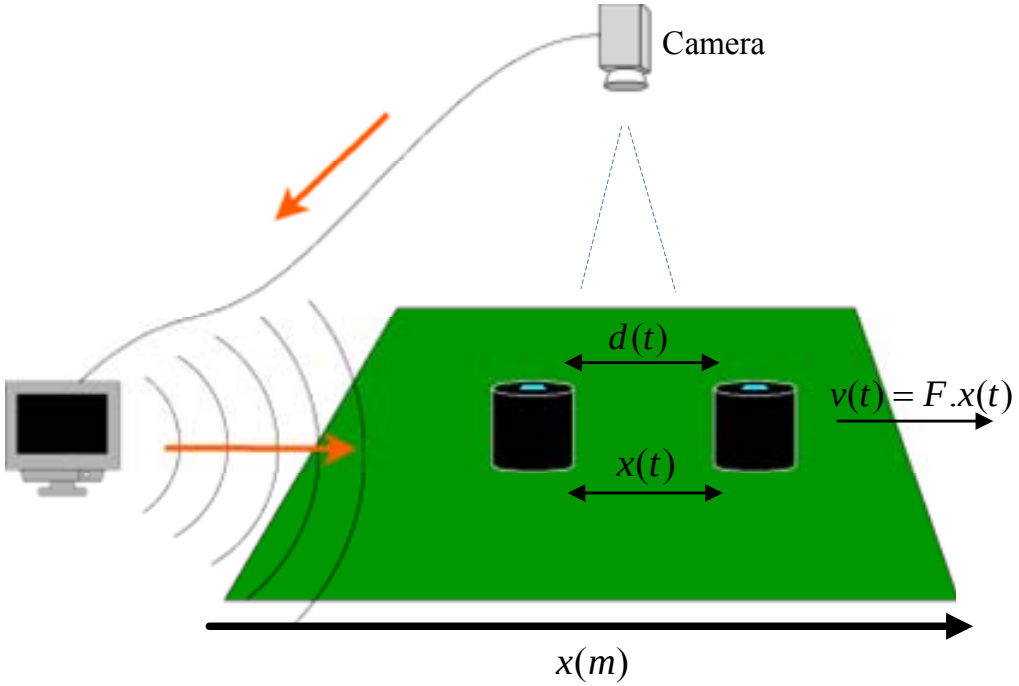
ภาพที่ 22 Pattern ทั้งหมดที่ทำการวิเคราะห์



ภาพที่ 23 เปรียบเทียบค่าความผิดพลาดทั้งตำแหน่งและมุม

9. การประมาณค่าตำแหน่งหุ่นยนต์ด้วย Kalman filter

โดยปกติสิ่งที่เราต้องการรู้เมื่อทำการวิเคราะห์ระบบก็คือ ณ เวลาหนึ่งๆ ระบบมีสถานะ (states) เป็นอย่างไร และสถานะของระบบเปลี่ยนแปลงตามเวลาอย่างไร ในทางปฏิบัติบ่อยครั้ง การหาสถานะของระบบไม่ใช่เรื่องง่าย เพราะมีข้อจำกัดหลายปัจจัย เช่น ความไม่สมบูรณ์ของเซ็นเซอร์ที่ใช้วัดสถานะของระบบและความคลาดเคลื่อนในการวัด ตัวอย่างเช่นในงานวิจัยนี้ข้อมูลที่ใช้ในการทดลองคือตำแหน่งหุ่นยนต์ที่ได้จากการประมวลผลภาพซึ่งแน่นอนว่าตำแหน่งที่ได้มาจะมีสัญญาณรบกวน (noise) เนื่องจากกล้องที่ใช้ในการถ่ายภาพ เป็นต้น วิธีหนึ่งสำหรับหาสถานะของระบบคือการใช้อัลกอริทึม Kalman filter (R.E. Kalman, 1960) โดยอัลกอริทึมนี้ถูกนำมาใช้เป็นครั้งแรกเพื่อประมาณสถานะของระบบนำร่องของยาน Apollo ในการโคจรรอบโลก ปัจจุบันถูกนำมาใช้อย่างแพร่หลายโดยเฉพาะอย่างยิ่งใช้ในการประมวลข้อมูลจากเซ็นเซอร์เพื่อหาค่าที่ดีที่สุด (data fusion) ภายใต้สัญญาณรบกวน (noise) จากหลายแหล่งเพื่อหาค่าประมาณของสถานะของระบบที่ดีที่สุด ตัวอย่างของระบบที่ใช้ Kalman filter ได้แก่ Integrated INS/GPS System (Sohne, W., Heinze, O. and Groten, E., 1994) และระบบการหาตำแหน่งวัตถุจากภาพ (Browning, B., Bowling, M. and Veloso, M., 2002) ซึ่งได้นำมาประยุกต์ใช้กับงานวิจัยนี้ซึ่งแสดงดังรูปภาพต่อไปนี้



ภาพที่ 24 ระบบการหาตำแหน่งของหุ่นยนต์ด้วยกล้อง

กำหนดให้ระบบตามแสดงใน ภาพที่ 24 ประกอบด้วยหุ่นยนต์กำลังวิ่งไปตามแนวแกน x ด้วยความเร็ว $v(t)$ ซึ่งมีค่าแปรผันตรงกับตำแหน่ง $x(t)$ ตามสมการอนุพันธ์ (first order linear differential equation) ต่อไปนี้

$$v(t) = \frac{dx(t)}{dt} = F.x(t), \tag{87}$$

โดย $F.x(t)$ คือฟังก์ชันของระยะทางที่ขึ้นกับเวลา

สมมุติว่าสถานะของระบบที่ต้องการรู้คือตำแหน่งในแนวแกน x ดังนั้น จากการแก้สมการที่ (87) สามารถคำนวณหาตำแหน่ง $x(t)$ ณ เวลาใดๆ ได้คือ

$$x(t) = x(t_0)e^{F.(t-t_0)}, \tag{88}$$

โดยที่ $x(t_0)$ คือตำแหน่งของหุ่นยนต์ในแนวแกน x ที่เวลา $t = t_0$ โดยจะเรียกสมการที่ (87) และ (88) ว่า process model เนื่องจากเป็นสมการที่ (88) สามารถแสดงความสัมพันธ์ของการเปลี่ยน แปลงสถานะของระบบเทียบกับเวลา

นอกจากจะหาตำแหน่งของหุ่นยนต์ได้จากความเร็วของหุ่นยนต์แล้วยังสามารถหาตำแหน่งได้จากได้จากกล้องซึ่งทำการประมวลผลภาพซึ่งได้ตามสมการต่อไปนี้

$$x(t) = d(t), \tag{89}$$

โดยที่ $d(t)$ คือระยะทางที่เปลี่ยนแปลงไปในช่วงเวลาหนึ่ง ซึ่งจะเรียกสมการที่ (89) ว่าเรียกว่า measurement model เนื่องจากเป็นสมการแสดงความสัมพันธ์ระหว่างสถานะของระบบกับค่าที่วัดได้จากเซ็นเซอร์

ทั้งสมการที่ (88) และ (89) เป็นแบบจำลองทางคณิตศาสตร์ที่กำหนดได้ (deterministic system model) แต่ตำแหน่งหุ่นยนต์จริงๆไม่สามารถวิเคราะห์ได้จากทั้ง 2 สมการนี้อย่างได้ถูกต้องและแม่นยำได้เนื่องจากปัจจัยที่เป็นไปได้ดังต่อไปนี้

1. ไม่มีแบบจำลองทางคณิตศาสตร์ใดที่สมบูรณ์ 100%

สมมุติฐานที่ว่าหุ่นยนต์วิ่งด้วยความเร็ว $v(t) = F.x(t)$ ตามสมการที่ (87) เป็นเพียงการประมาณเนื่องจากจริงๆแล้วมีตัวแปรเป็นจำนวนมากที่มีผลต่อความเร็วและการเคลื่อนที่ของหุ่นยนต์ซึ่งไม่สามารถนำมาเขียนเป็นสมการได้อย่างแม่นยำ เช่น การสึกหรอของมอเตอร์และประสิทธิภาพของแบตเตอรี่ เป็นต้น

2. ไม่มีเซ็นเซอร์ใดที่วัดค่าได้สมบูรณ์ 100%

ในทางปฏิบัติเซ็นเซอร์ทุกชนิดมีความคลาดเคลื่อน (measurement noise) จะมากหรือน้อยขึ้นอยู่กับหลายปัจจัย นอกจากนี้ระบบจริงๆ ยังไม่มีเซ็นเซอร์ที่สามารถวัดได้โดยตรง อย่างเช่นระบบหาตำแหน่งจากภาพต้องใช้กล้องในการรับข้อมูลเข้ามาประมวลผลซึ่งก็แน่นอนว่าตัวกล้องเองก็มีผลกับความแม่นยำของตำแหน่งหุ่นยนต์ที่ได้ โดยปัจจัยที่มีผลก็คือ ความโค้งของเลนส์ ความชัดของภาพ เป็นต้น

3. ปัจจัยภายนอกที่ไม่สามารถควบคุมได้

จากตัวอย่างข้างต้นปัจจัยภายนอกที่มีผลต่อความเร็วของหุ่นยนต์ เช่น สภาพพื้นที่หุ่นยนต์วิ่งซึ่งแน่นอนว่าวัสดุของพื้นผิวมีผลกับแรงเสียดทานที่กระทำกับล้อหุ่นยนต์ซึ่งมีผลโดยตรงกับความเร็วของหุ่นยนต์

จากปัญหาการวิเคราะห์ระบบแบบ deterministic ข้างต้น จึงนำไปสู่การวิเคราะห์แบบ stochastic ซึ่งนำความไม่แน่นอน ข้อมูลทางสถิติ และหลักการของความน่าจะเป็นมาพิจารณาร่วมด้วยในตัวอย่างนี้สามารถเขียนสมการที่ (87) และ (89) ได้ใหม่ในรูปของ continuous -time stochastic system model เป็นสมการดังต่อไปนี้

$$\dot{x}(t) = Fx(t) + w(t), \tag{90}$$

$$d(t) = x(t) + v(t). \tag{91}$$

$w(t)$ และ $v(t)$ เป็นตัวแปรสุ่ม (random variable) ใช้เป็นแบบจำลองความไม่แน่นอนในระบบ และจากสมการที่ (90) และ (91) สามารถเขียน discrete-time system model เพื่อหาระยะ $x(t)$ ณ เวลา $t = t_k$ ได้คือ

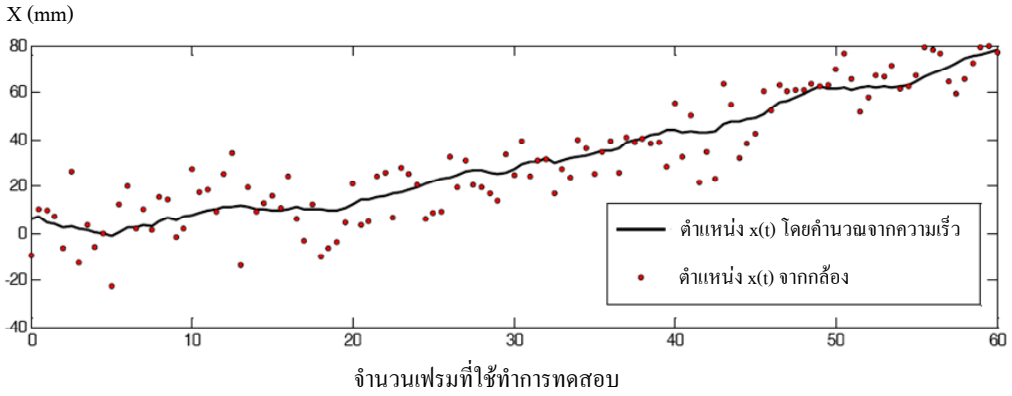
$$x_k = \Phi_{k-1} x_{k-1} + w_k, \tag{92}$$

$$x_k = d_k + v_k, \tag{93}$$

โดยที่ $\Phi_{k-1} = e^{F\Delta t}$, $\Delta t = t_k - t_{k-1}$, x_k และ x_{k-1} คือระยะห่างที่หุ่นยนต์เคลื่อนที่ไปได้ที่เวลา $t = t_k$ และ $t = t_{k-1}$ ตามลำดับ ดังนั้นจะเห็นได้ว่าสมการที่ (92) สามารถหาสถานะของระบบที่เวลา $t = t_k$ ได้จากข้อมูลก่อนหน้า (สถานะของระบบที่เวลา $t = t_{k-1}$) แต่มีปัญหาตามมาคือจะใช้ค่า v_k และ w_k เท่าไรเนื่องจากค่าทั้งสองค่านี้เป็นตัวแปรสุ่ม

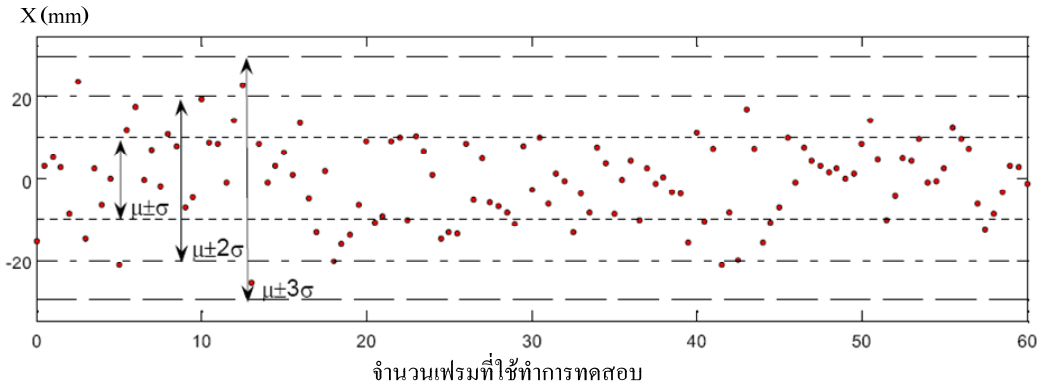
ถ้ากำหนดค่า v_k และ w_k เป็นตัวแปรสุ่มแบบ Gaussian ซึ่งนิยามเป็น $X = N(\mu, \sigma^2)$ หมายความว่า X เป็นตัวแปรสุ่มแบบ Gaussian มีค่า mean เท่ากับ μ และ variance เท่ากับ σ^2 ซึ่งค่า variance เป็นค่าตัวแปรระบุความแปรปรวนของข้อมูลมากน้อยเพียงไร ยิ่ง variance มีค่าสูงความแม่นยำของสถานะของระบบจะยิ่งมีค่าน้อย โดยจะกำหนดให้ค่าความไม่แน่นอนของการเคลื่อนที่เป็น $w_k = N(0,1)$ ซึ่งเป็น Gaussian white noise และกำนดให้ความคลาดเคลื่อน (measurement noise) ของกล้องคือ $v_k = N(0,10)$ ซึ่งเป็นค่าที่ได้จากการทดลองมา

หลังจากกำหนดค่าตัวแปรสุ่มแล้วลองทำการเก็บข้อมูลจริงขึ้นมาทดสอบ ซึ่งภาพต่อไปนี้แสดงผลการเคลื่อนที่ของหุ่นยนต์และค่าที่กล้องวัดตำแหน่งได้เมื่อสุ่มตัวอย่างทุก ๆ 0.0167 วินาที เป็นเวลา 60 ครั้ง



ภาพที่ 25 ตำแหน่งของหุ่นยนต์ที่คำนวณจากความเร็วและที่ได้จากกล้องด้วยกล้อง

ลองนำข้อมูลที่ได้มา plot โดยให้ค่า mean เป็น 0 จะเห็นได้ว่าคุณสมบัติหนึ่งของตัวแปรสุ่มแบบ Gaussian คือ สามารถกำหนดกรอบของค่า noise ได้อย่างเช่นจากข้อมูลตัวอย่างประมาณ 68%, 95% และ 99% ของค่าที่ได้จากการสุ่มทั้งหมดจะอยู่ในช่วง $\mu \pm \sigma$, $\mu \pm 2\sigma$ และ $\mu \pm 3\sigma$ ตามลำดับ ดังแสดงในภาพต่อไปนี้



ภาพที่ 26 camera measurement noise $v_k = N(0,10)$

จากภาพที่ 25 และ 26 จะสังเกตได้ว่าถ้าเรารู้คุณลักษณะทางสถิติของระบบและของเซ็นเซอร์ เราจะสามารถนำข้อมูลทั้งหมดมาใช้ประกอบกับหลักการของความน่าจะเป็นเพื่อหาค่าประมาณที่ดีที่สุดของสถานะของระบบได้ซึ่งเป็นหลักการของ Kalman filter นั่นเอง

Kalman filter คืออัลกอริทึมใช้สำหรับหาค่าประมาณที่ดีที่สุดของสถานะของระบบโดยนำข้อมูลเกี่ยวกับความไม่แน่นอน เช่น ความไม่แน่นอนของกลศาสตร์ของระบบ (system dynamics), ความคลาดเคลื่อนของเซ็นเซอร์ (measurement noise) มาประกอบการพิจารณาบนพื้นฐานของ

ความน่าจะเป็น โดยการใช้ Kalman filter เริ่มต้นด้วยการเขียนแบบจำลองทางคณิตศาสตร์ของระบบให้อยู่ในรูปของ discrete-time system model ด้วยสมการ

$$x_k = \Phi_{k-1} x_{k-1} + w_k, \tag{94}$$

$$z_k = H_k x_k + v_k, \tag{95}$$

สมการที่ (94) และ (95) คือ discrete-time process model และ measurement model ตามลำดับ ค่า v_k และ w_k คือ zero-mean Gaussian White noise ซึ่งหมายความว่า v_k และ w_k เป็นตัวแปรสุ่มแบบ Gaussian มีค่าเฉลี่ยเป็นศูนย์ และกำหนดให้ variance ของ v_k และ w_k มีค่าเป็น Q_k และ R_k ตามลำดับ ซึ่ง White Noise คือ noise ที่เกิดขึ้นอยู่แล้วเป็นปกติในระบบและไม่เกี่ยวข้องกับค่าอื่นๆ

การใช้งาน Kalman filter มีขั้นตอนดังนี้ ในขณะที่เซ็นเซอร์ยังไม่สามารถวัดค่าอะไรได้ สามารถประมาณค่า state ของระบบ x_k เวลา $t = t_k$ ได้จากสมการ

$$\hat{x}_k^- = \Phi_{k-1} \hat{x}_{k-1}, \tag{96}$$

$\hat{x}_k^-$ เป็นการระบุว่า state เป็นค่าประมาณโดยไม่มีค่าที่เซ็นเซอร์วัดได้มาประกอบการพิจารณา การคาดการณ์ไปในอนาคตย่อมมีความไม่แน่นอนซึ่งให้ความคลาดเคลื่อนในการคาดการณ์คือ $e_k^- = \hat{x}_k - \hat{x}_{k-1}^-$ (ความแตกต่างระหว่างสถานะจริงของระบบกับค่าที่เราคาดการณ์จากการคำนวณ)

เนื่องจากเรากำหนดว่าค่า w_k เป็นตัวแปรสุ่มแบบ Gaussian ทำให้ค่า $\hat{x}_k^-$ และ e_k^- เป็นตัวแปรสุ่มแบบ Gaussian ด้วย โดยที่ e_k^- มีค่า mean เป็นศูนย์ และมี variance คือ

$$P_k^- = E[(e_k^-)^2] = \Phi_{k-1}^2 P_{k-1} + Q_k. \tag{97}$$

เราสามารถใช้สมการที่ (96) และ (97) เพื่อคาดการณ์ค่าของ $x(t)$ และความน่าเชื่อถือของการคาดการณ์ (P_k^-) ไปเรื่อยๆ จนกระทั่งเซ็นเซอร์สามารถวัดค่าได้ เมื่อเซ็นเซอร์วัดค่าได้ Kalman

filter จะนำค่าที่วัดได้มาประกอบการพิจารณาเพื่อหาค่าประมาณสถานะของระบบตามสมการต่อไปนี้

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(z_k - H_k\hat{x}_k^-), \tag{98}$$

โดยที่

$$K_k = P_k^- H_k(P_k^- H_k^2 + R_k)^{-1}. \tag{99}$$

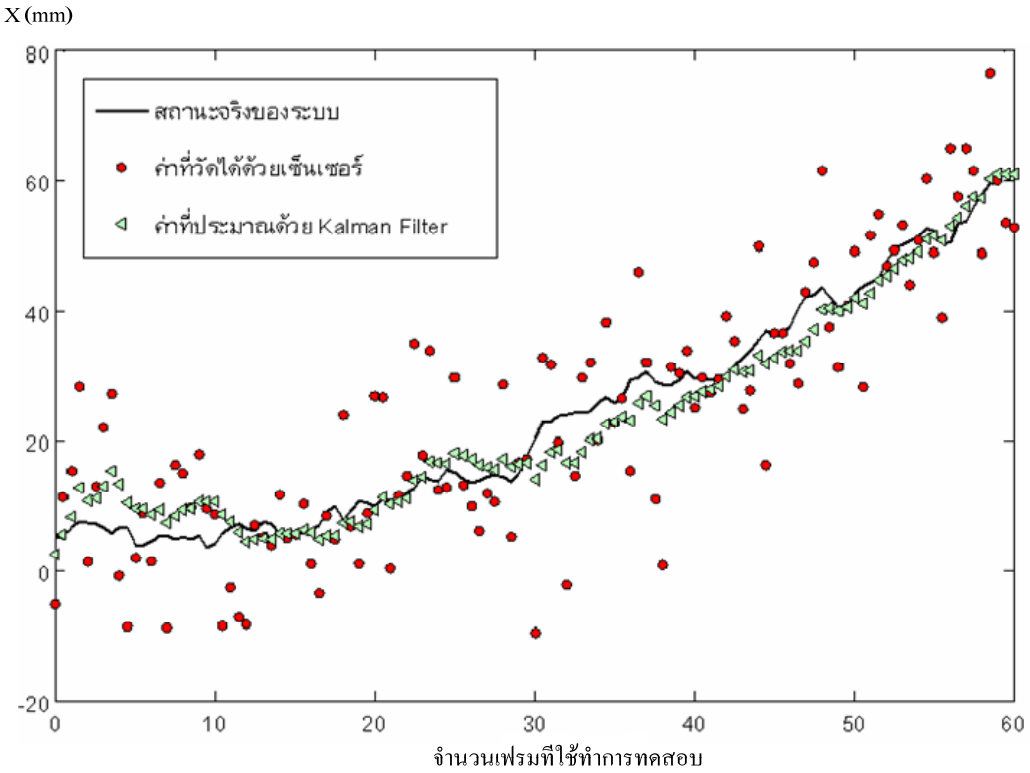
$\hat{x}_k^+$ เป็นการระบุว่า $\hat{x}_k$ เป็นค่าประมาณโดยได้นำจากเซ็นเซอร์ที่วัดได้มาประกอบการพิจารณา โดยสมการที่ (98) คือสมการที่ใช้ Update ค่า $\hat{x}_k^-$ ที่เราคาดการณ์ไว้ล่วงหน้าก่อนที่เซ็นเซอร์จะวัดค่าได้และเมื่อพิจารณาจากสมการที่ (98) และ (99) แล้วจะเห็นว่า $H_k\hat{x}_k^-$ คือ สมการที่ใช้ประมาณค่าที่เซนเซอร์ควรวัดได้ ($\hat{z}_k$) คำนวณโดยใช้ค่าประมาณ $\hat{x}_k^-$ และอัลกอริทึมจะทำงานโดยการหา residual $(z_k - H_k\hat{x}_k^-)$ ซึ่งเป็นผลต่างระหว่างค่าที่เซนเซอร์ควรวัดได้กับค่าที่วัดได้จริง นำมาให้น้ำหนักโดยการคูณด้วย Kalman Gain K_k แล้วนำผลที่ได้มาใช้แก้ไขค่า $\hat{x}_k^-$ ที่ได้คาดการณ์ไว้ล่วงหน้า

จากที่กล่าวมาข้างต้นจะสังเกตได้ว่าค่าที่นำมาใช้แก้ไข $\hat{x}_k^-$ ขึ้นอยู่กับขนาดของ residual และ K_k โดยถ้าผลต่างระหว่างค่าที่คาดการณ์กับค่าที่วัดได้จริงมีมากและข้อมูลมีความน่าเชื่อถือน้อย ค่า Kalman gain ที่คำนวณได้จะมีค่าสูง ทำให้ต้องแก้ไข $\hat{x}_k^-$ มาก ($K_k(z_k - H_k\hat{x}_k^-)$ มีค่ามาก) ในทางกลับกันถ้าผลต่างระหว่างค่าที่คาดการณ์ไว้กับค่าที่วัดได้จริงมีน้อยและข้อมูลมีความน่าเชื่อถือมาก ค่า Kalman gain ที่คำนวณได้จะมีค่าต่ำ ทำให้แก้ไข $\hat{x}_k^-$ เพียงเล็กน้อย ($K_k(z_k - H_k\hat{x}_k^-)$ มีค่าน้อย) ซึ่งจะสังเกตได้ว่าการคำนวณค่า Kalman gain ตามสมการที่ (98) เป็นการคำนวณที่ได้นำข้อมูลทางสถิติของระบบมาพิจารณาแล้วและเป็นค่า gain ที่จะทำให้ $\hat{x}_k^+$ มีค่าต่ำสุดซึ่งลักษณะนี้เป็นลักษณะของ optimal state estimator

ในการหาค่าประมาณโดยใช้สมการที่ (99) จะสามารถพิสูจน์ได้ว่า ค่าความคลาดเคลื่อนของการประมาณ $e_k^- = \hat{x}_k - \hat{x}_{k-1}^-$ เป็นตัวแปรแบบ Gaussian มี ค่า mean เป็นศูนย์ และมี variance คือ

$$P_k^- = E[(e_k^+)^2] = (1 - K_k H_k)^2 P_k^- + K_k^2 R_k. \quad (100)$$

สุดท้ายจะสังเกตการทำงานของ Kalman filter ได้ว่า การทำงานจะเป็นลักษณะของการ Predict $\hat{x}_k^-$ และ Estimate $\hat{x}_k^+$ ซึ่งหมายความว่า Kalman filter จะพยายามทำนายว่าสถานะของระบบ $\hat{x}_k^-$ น่าจะเป็นอย่างไรโดยไม่ใช้ค่าจากเซ็นเซอร์และการทำการประเมินค่า $\hat{x}_k^+$ เมื่อเซ็นเซอร์วัดค่าได้ ซึ่งผลลัพท์ที่ได้แสดงดังภาพต่อไปนี้



ภาพที่ 27 เปรียบเทียบตำแหน่งที่ได้จาก Kalman และตำแหน่งที่ได้จากกล้อง

จากภาพจะเห็นได้ชัดว่า Kalman filter สามารถประมาณสถานะของระบบคือ ตำแหน่งที่หุ่นยนต์เคลื่อนที่ในแนวแกน x ได้อย่างดี สังเกตจากค่าประมาณมีค่าใกล้เคียงกับสถานะจริงและเรียบกว่าค่าที่วัดได้ด้วยเซ็นเซอร์ ที่เป็นเช่นนี้เพราะ Kalman filter ใช้ความรู้เกี่ยวกับ dynamics ของระบบ, คุณลักษณะทางสถิติของระบบและเซ็นเซอร์ และค่าที่วัดได้จากเซ็นเซอร์มาประมวลผลบนหลักการของความน่าจะเป็น เพื่อหาค่าประมาณของสถานะของระบบที่ดีที่สุด

และจุดเด่นอีกอย่างของ Kalman filter คือเป็นการประมวลผลแบบ recursive หมายความว่าใช้เฉพาะข้อมูลจากเวลาก่อนหน้า $t = t_{k-1}$ เพียงค่าเดียวเท่านั้นเพื่อคำนวณสถานะที่เวลาปัจจุบัน $t = t_k$ ข้อดีของการประมวลผลแบบ recursive คือไม่ต้องใช้หน่วยความจำมาก

อุปกรณ์และวิธีการ

อุปกรณ์

ฮาร์ดแวร์ (hardware)

1) เครื่องคอมพิวเตอร์พีซี	3	เครื่อง
2) กล้อง Firewire AVT Stingray	2	ตัว
3) หุ่นยนต์ 2 ล้อ	1	ตัว
4) บอร์ด dsPIC30f2010 ควบคุมมอเตอร์	1	บอร์ด

ซอร์ฟแวร์ (software)

1) Microsoft Window XP Service Pack 2

2) Matlab Version 7.0

3) Microsoft Visual Studio.Net 2005

4) CMVision library

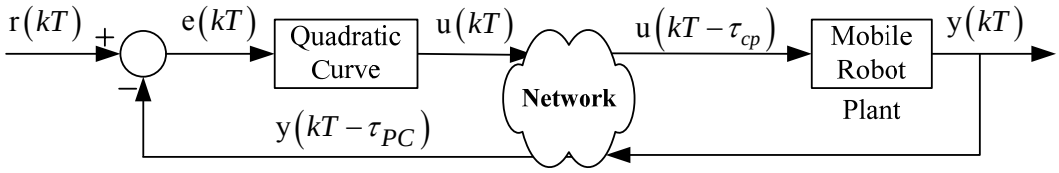
5) OpenCV image processing library

6) AutoCAD Autodesk Inventor 2008

วิธีการ

ในหัวข้อความรู้เบื้องต้นและทฤษฎีที่เกี่ยวข้องได้เสนอรูปแบบการทำงานของระบบนิวโรฟัซซี่กับตัวควบคุมผ่านเน็ตเวิร์คไร้สายสำหรับหุ่นยนต์เคลื่อนที่ รูปแบบทางคณิตศาสตร์ของหุ่นยนต์ อัลกอริทึมในการเดินตามเส้นทางของหุ่นยนต์ การจัดกลุ่มสภาวะเน็ตเวิร์คด้วยระบบนิวโรฟัซซี่โครงสร้างการปรับค่าพารามิเตอร์ส่วนควบคุมให้เหมาะสมและผลกระทบของดีเลย์ที่มีต่อประสิทธิภาพของระบบ

เพื่อศึกษาและวิจัยสมรรถนะของวิธีการที่นำเสนอในงานวิจัยชิ้นนี้จึงจะต้องจำลองการควบคุมหุ่นยนต์ให้เดินตามเส้นทางผ่านเน็ตเวิร์คของใน Matlab\Simulink 7.0 เป็นกรณีตัวอย่างและนอกจากจะทำการจำลองระบบในโปรแกรมแล้วยังนำผลที่ได้ไปทำการทดสอบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คจริงด้วยเพื่อให้เห็นว่าวิธีที่นำเสนอนั้นสามารถนำไปใช้งานจริงได้ ซึ่งรายละเอียดของระบบมีดังต่อไปนี้



ภาพที่ 28 ไดอะแกรมของระบบ NBC ควบคุมหุ่นยนต์ผ่านเน็ตเวิร์ค

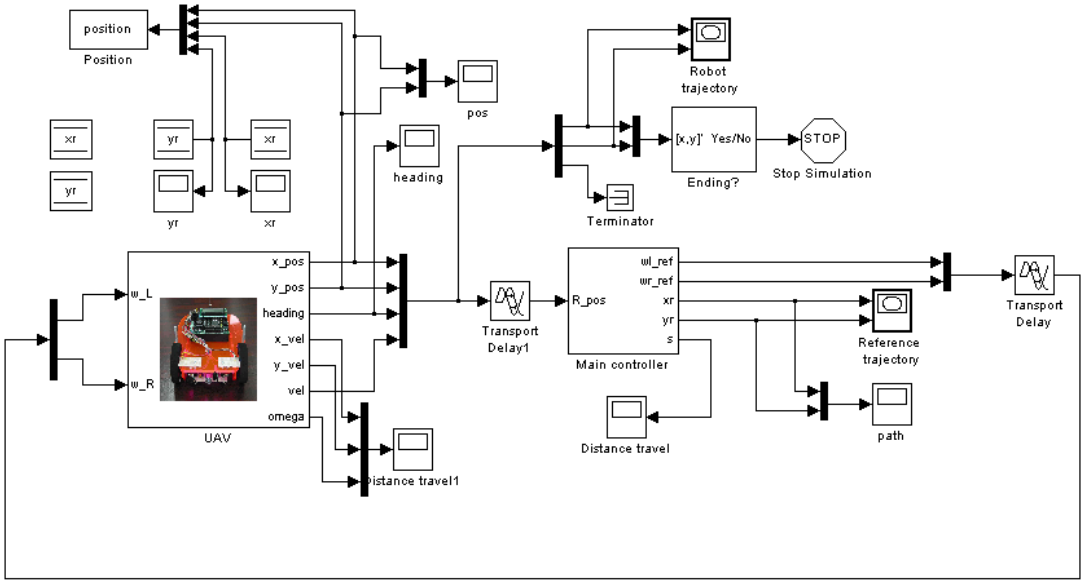
เนื่องด้วยรูปแบบและสัญลักษณ์ในการทำงานของ NBC เป็นตามรูปแบบทั่วไปที่ได้นำเสนอในหัวข้อความรู้เบื้องต้นและทฤษฎีที่เกี่ยวข้องแล้วจึงไม่ขอกล่าวถึงอีก

โดยในการทดสอบสมรรถนะของวิธีการที่นำเสนอในระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คในงานวิจัยนี้ได้แบ่งวิธีการออกเป็น 5 ขั้นตอนดังต่อไปนี้

1. การสร้างแบบจำลองระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์ค
2. การเก็บรวบรวมข้อมูลของสภาวะของเน็ตเวิร์คเพื่อนำมาทำการวิเคราะห์
3. การคลัสเตอร์สภาวะของเน็ตเวิร์คโดยประยุกต์ใช้ SANFISII
4. การหาพารามิเตอร์ของตัวควบคุมที่เหมาะสมสำหรับดีเลย์แต่ละคลัสเตอร์
5. การทดลองควบคุมหุ่นยนต์ผ่านเน็ตเวิร์ค

1. การสร้างแบบจำลองระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์ค

เนื่องจากงานวิจัยชิ้นนี้จะทำการควบคุมหุ่นยนต์ผ่านเน็ตเวิร์ค แต่ก่อนที่จะควบคุมหุ่นยนต์จริงเราจะต้องทดสอบอัลกอริทึมในการเดินตามเส้นทางและจะหาต้องหาค่า β และ α ที่เหมาะสมในการควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คก่อน ซึ่งการทดสอบอัลกอริทึมทั้งหมดจะถูกจำลองในโปรแกรม Matlab/Simulink7.0 ซึ่ง block diagram แสดงในภาพต่อไปนี้



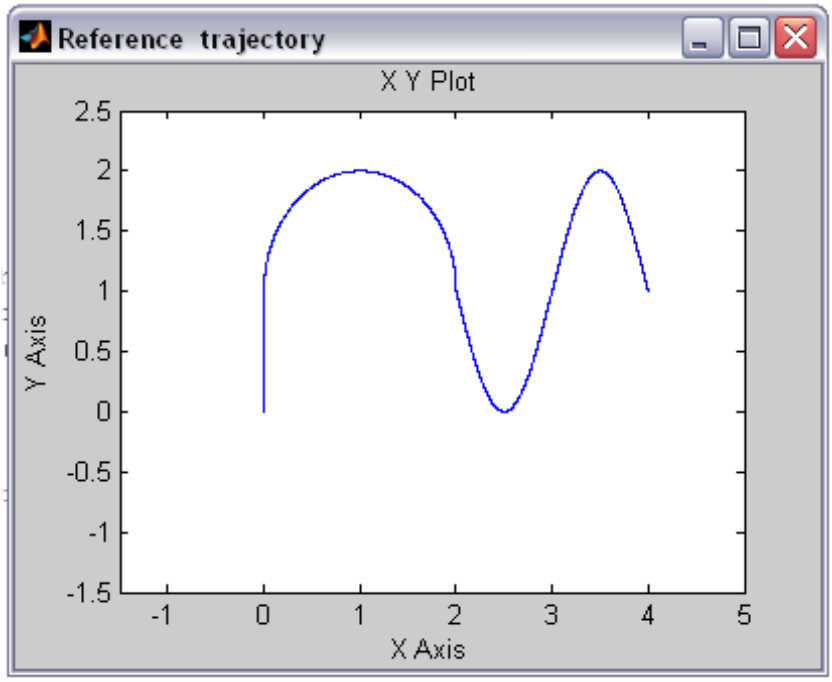
ภาพที่ 29 โปรแกรมจำลองการควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คบนโปรแกรม Matlab 7.0

ในการสร้างแบบจำลองการควบคุมหุ่นยนต์ผ่านเน็ตเวิร์ค อันดับแรกเราต้องหารูปแบบทางคณิตศาสตร์ของหุ่นยนต์ซึ่งแสดงในสมการที่ (25) และ (26) โดยพารามิเตอร์ในแบบจำลองทางคณิตศาสตร์ของหุ่นยนต์จำเป็นต้องมีความแม่นยำเหมือนกับหุ่นยนต์ที่ใช้ในการทดลองจริงมากที่สุดเพราะว่าเราต้องนำค่า α และ β ที่ได้จากการจำลองบนโปรแกรม Matlab7.0 ไปใช้ในระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คจริง ซึ่งค่าพารามิเตอร์ที่หามาได้และใช้ในงานวิจัยได้แสดงในตารางต่อไปนี้

ตารางที่ 9 พารามิเตอร์ของหุ่นยนต์ที่สร้างขึ้นมาเพื่อใช้ในการทดลอง

Parameter	Description	Value
R_a	Armature resistance	4.67 Ω
L_a	Armature inductance	170e-3 H
K_b	Back-Emf constant	14.7e-3 V-sec/rad
K_i	Torque constant	14.7e-3 N-m/A
J	Moment of inertia	42.6e-6 Kg-m2
B	Viscous-friction coefficient	47.3e-6 N-m/rad/sec
W	Distance Between 2 wheels	0.15 m
r	Radius of wheel	0.03 m

นอกจากตัวหุ่นยนต์ที่ต้องหาพารามิเตอร์แล้ว เราจะต้องออกแบบเส้นทางที่ให้หุ่นยนต์เดินเพื่อทดสอบประสิทธิภาพด้วย โดยเส้นทางที่เลือกใช้จะประกอบไปด้วยเส้นตรง เส้นโค้ง และโค้งหักศอก เพื่อให้ที่จะให้ครอบคลุมเส้นทางทุกรูปแบบที่เป็นไปได้ ซึ่งเส้นทางที่ออกแบบเป็นดังภาพต่อไปนี้



ภาพที่ 30 เส้นทางที่ให้หุ่นยนต์เดินเพื่อทดสอบประสิทธิภาพ

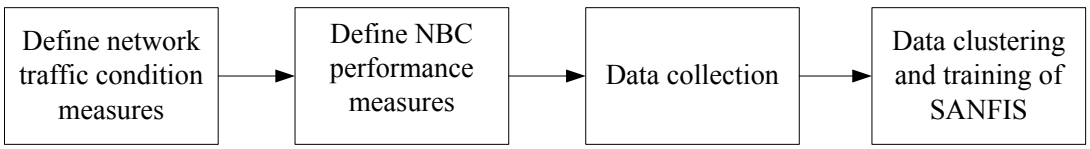
2. การเก็บรวบรวมข้อมูลของสภาวะของเน็ตเวิร์คเพื่อนำมาทำวิเคราะห์

เนื่องด้วยลักษณะของการเกิดดีเลย์ที่เกิดขึ้นในเน็ตเวิร์คไม่อาจคาดคะเนลักษณะที่แน่นอนได้ เพื่อเป็นการแสดงให้เห็นถึงลักษณะดังที่กล่าวมา เราต้องทำการเก็บข้อมูลดีเลย์ round trip time (RTT) ระหว่าง Intelligent Mechatronics Lab (IML) มหาวิทยาลัยเกษตรศาสตร์ไปยังสถานที่ต่างๆ เพื่อนำมาวิเคราะห์และตรวจสอบดูว่าสภาวะเน็ตเวิร์คแต่ละที่เป็นอย่างไร โดยเลือกสถานที่ที่จะใช้เก็บข้อมูลโดยดูจากระยะทางที่ต่างๆกัน ซึ่งสถานที่ที่สุ่มเลือกเพื่อใช้ในการเก็บข้อมูลดีเลย์เพื่อนำมาวิเคราะห์มีทั้งหมด 4 สถานที่ ดังต่อไปนี้

1. **http://clamps.informatik.uni-bremen.de/** ประเทศเยอรมัน
2. **http://www.cs.cmu.edu/** ประเทศสหรัฐอเมริกา
3. **http://www.nlict.zju.edu.cn/** ประเทศจีน
4. **http://www.cmu.ac.th/** จังหวัดเชียงใหม่ ประเทศไทย

3. การคลัสเตอร์สภาวะของเน็ตเวิร์คโดยประยุกต์ใช้ SANFIS

ขั้นตอนการคลัสเตอร์โดยใช้ SANFIS ประกอบด้วย 4 ขั้นตอนสำคัญดังต่อไปนี้



ภาพที่ 31 ขั้นตอนการทำงานของ SANFIS

3.1 Define network traffic condition measure

กำหนดให้ $d_{RTT}(k)$ คือดีเลย์ Round Trip Time (RTT) ที่ทำการวัดทุกๆ sampling time $t = kT$ โดย $d_{RTT}(k)$ เป็นตัวแปรที่ใช้ในการคำนวณตัวแปรเงื่อนไขของสภาวะเน็ตเวิร์คในเวคเตอร์ $\mathbf{q}(t)$ มีดังต่อไปนี้

q_1 คือการคำนวณหา median ของดีเลย์จากค่า $d_{RTT}(k)$ จำนวน N ค่า

q_2 คือการคำนวณหา variance ของดีเลย์จากค่า $d_{RTT}(k)$ จำนวน N ค่า

q_3 คือการคำนวณหา exponential moving average ของดีเลย์จากค่า $d_{RTT}(k)$ จำนวน N ค่า ซึ่ง สามารถคำนวณได้จากสมการดังต่อไปนี้

$$ema(k) = \left[d_{RTT}(k) - \left(2ema(k-1)/N\right)\right] + ema(k-1). \tag{101}$$

โดยค่าเริ่มต้นของ $ema(k)$ มีค่าเท่ากับ mean ของดีเลย์จากค่า $d_{RTT}(k)$ N ค่า และในงานวิจัยชิ้นนี้กำหนดให้ N = 10 นอกจากนี้เงื่อนไขของสภาวะเน็ตเวิร์คยังสามารถปรับเปลี่ยนได้ขึ้นอยู่กับความเหมาะสมกับการทำงาน

สำหรับในงานวิจัยนี้สาเหตุที่เลือกใช้ค่า mean variance และ exponential moving average เนื่องจากทำการทดสอบดูแล้วว่าคุณสมบัติของดีเลย์เหล่านี้มีผลต่อประสิทธิภาพของระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คในระบบจำลองมากกว่าคุณสมบัติอื่นๆ

3.2 Define NBC performance measures

เพื่อการทดสอบว่าการทำงานของหุ่นยนต์ว่ามีประสิทธิภาพเช่นไรกับสภาวะเงื่อนไขเน็ตเวิร์ค ($\mathbf{q}(t)$) ที่กำลังเผชิญอยู่ ดัชนีที่ใช้ในการตรวจวัดมีดังต่อไปนี้

$$J = \lambda J_1 + (1-\lambda) J_2, \tag{102}$$

$$J_1 = \frac{1}{T}\int_0^T \min \sqrt{(x(t)-x_{path})^2 + (y(t)-y_{path})^2}\, dt, \tag{103}$$

$$J_2 = T = \int_0^T dt. \tag{104}$$

โดยที่สมการที่ (103) เป็นค่าที่บ่งบอกประสิทธิภาพของการวิ่งตรงเส้นทางของหุ่นยนต์ (J_1) และสมการที่ (104) เป็นเวลาทั้งหมดที่หุ่นยนต์ใช้เดินตามเส้นทาง (J_2) ซึ่งในที่นี้ถือว่าเป็นค่าที่บ่ง

บอกถึงประสิทธิภาพของระบบด้วย โดยค่าบ่งบอกประสิทธิภาพของระบบรวมเกิดจากการนำค่าบ่งบอกประสิทธิภาพทั้ง J_1 และ J_2 มารวมกันและสามารถเลือกให้น้ำหนักที่ค่า λ ในสมการที่ (102)

ค่า λ สามารถปรับเปลี่ยนไปตามระบบที่ทำการทดสอบ โดยในงานวิจัยนี้ได้เลือกใช้ค่า λ เท่ากับ 0.972 ซึ่งได้มาจากการทดลองระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คจริง

3.3 Data collection

การที่จะนำ SANFISII ไปประยุกต์ใช้กับงานใดๆจำเป็นต้องทำการเทรนนิ่งระบบจากตัวอย่างข้อมูลก่อนเพื่อให้ระบบมีประสิทธิภาพการทำงานสูงสุด โดยนำดีเลย์ที่เป็นตัวอย่างในการวิเคราะห์ทั้งหมดทำการสุ่มตัวอย่างเป็นช่วงๆ และนำดีเลย์ทุกช่วงไปหา median (q_1) variance (q_2) และ exponential moving average (q_3)

หลังจากนั้นนำ q_1, q_2 และ q_3 ทุกๆช่วงเข้าไปหาค่าบ่งบอกประสิทธิภาพ (J) ต่อเนื่องในโปรแกรมจำลอง Matlab/Simulink7.0 เพื่อสร้างเทรนนิ่งเซต

สมมุติว่าเทรนนิ่งเซตมีสมาชิกจำนวน n ตัว,

$$\left\{(\mathbf{q}_i, J_i) \text{ โดย } \mathbf{q} = [q_1, q_2, q_3] \text{ และ } J \in \mathbb{R}^+, i = 1, 2, \ldots, n\right\}$$

3.4 Data clustering and training of SANFIS

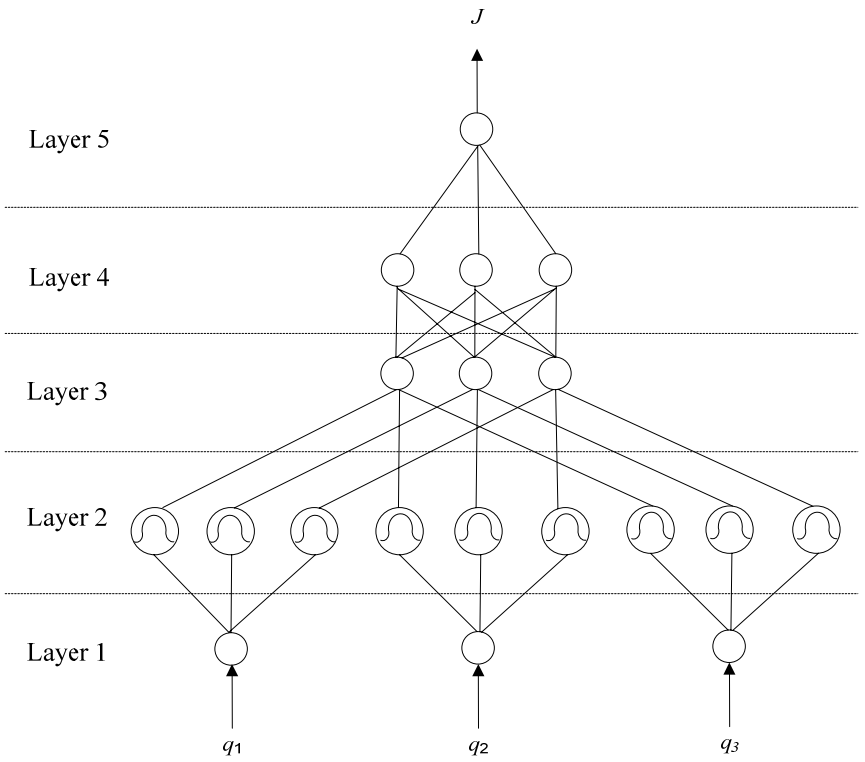
หลังจากเสร็จสิ้นขั้นตอน Data collection จะได้เทรนนิ่งเซตมาโดยมีจำนวนสมาชิกทั้งสิ้น n ตัว ซึ่งในขั้นตอนนี้จะเทรนนิ่งเซตทั้งหมดมาทดสอบตามขั้นตอนต่อไปนี้

1) ทำการประมวลผลอัลกอริทึม MCA เพื่อกำหนดค่าเริ่มต้นของโครงสร้างภายในของ SANFISII โดยอัลกอริทึม MCA จะทำการคลัสเตอร์สภาวะของเน็ตเวิร์คและค่าบ่งบอกประสิทธิภาพ (J)

สมมุติว่าหลังจากการประมวลผลด้วยอัลกอริทึม MCA ได้กลุ่มข้อมูลทั้งหมด 3 คลัสเตอร์ ซึ่งบ่งบอกถึงจำนวนกฎที่ใช้กำหนดโครงสร้าง SANFIS มีดังต่อไปนี้

- เลเยอร์ที่ 2 มี 3 x 3โหนด ซึ่งเท่ากับจำนวนคลัสเตอร์ × จำนวนอินพุต (m , σ , θ)
- เลเยอร์ที่ 3 มี 3 โหนด ซึ่งเท่ากับจำนวนคลัสเตอร์
- เลเยอร์ที่ 4 มี 3 โหนด ซึ่งเท่ากับจำนวนคลัสเตอร์

โดยภาพโครงสร้าง SANFIS หลังการประมวลอัลกอริทึม MCA นำเสนอดังภาพต่อไปนี้



ภาพที่ 32 โครงสร้าง SANFIS หลังประมวลผล MCA algorithm
ที่ประกอบด้วย 3 กฎ 3 อินพุตและ 1 เอาต์พุต

2) ทำการประมวลผล fast recursive liner/nonlinear least-squares optimization ซึ่งเป็นขั้นที่สองเพื่อการเทรนพารามิเตอร์ (m , σ , θ) ให้ได้ค่าผลลัพธ์จากโครงสร้าง SANFISII กับผลลัพธ์ที่ต้องการใกล้เคียงมากที่สุด

4. การหาพารามิเตอร์ของตัวควบคุมที่เหมาะสมสำหรับดีเลย์แต่ละคลัสเตอร์

ในการหาค่า α และ β ที่เหมาะสมสำหรับแต่ละคลัสเตอร์นั้น มีขั้นตอนในการดำเนินการดังต่อไปนี้

1) กำหนดช่วงทดสอบ α และ β คือ $0.1, 0.125, \ldots, 0.5$ และ $0.1, 0.125, \ldots, 0.4$ ตามลำดับ
2) ทำการจำลองระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คเพื่อทดสอบหาประสิทธิภาพ J โดยใช้ค่า α และ β ที่กำหนดข้างต้น โดยกำหนดกลุ่มของสภาวะเน็ตเวิร์คให้แตกต่างกันตามที่ได้ทำการคลัสเตอร์ในขั้นตอนที่แล้ว
3) คำนวณหาค่าเฉลี่ยของ J จากข้อมูลทุกตัวในแต่ละกลุ่มของสภาวะเน็ตเวิร์คที่เป็นผลกับ α และ β

$$\overline{J}_k = \frac{\sum_{i=1}^{p_k} J_{k_i}^{(\alpha,\beta)}}{p_k} \tag{105}$$

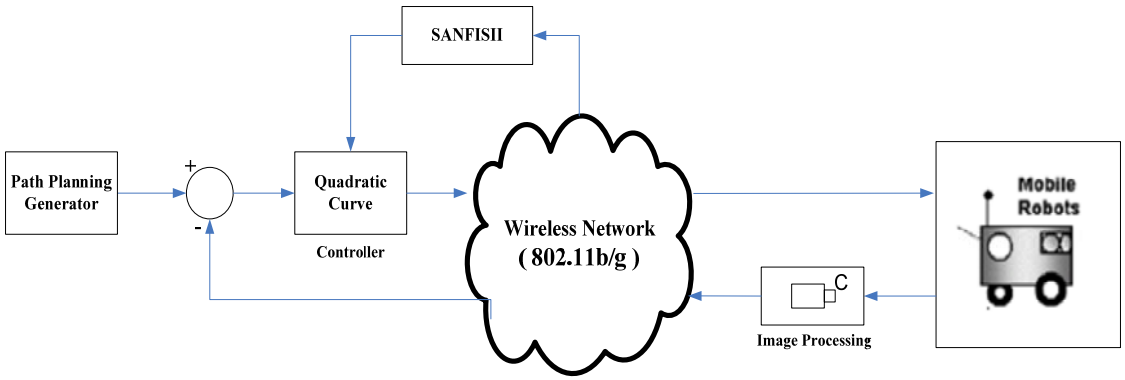
โดยที่ p คือจำนวนข้อมูลในกลุ่มสภาวะเน็ตเวิร์ค (คลัสเตอร์) ที่ k

k คือดัชนีระบุกลุ่มสภาวะเน็ตเวิร์ค (คลัสเตอร์)

4) ทำการสร้างกราฟเพื่อหาค่า α และ β ที่ทำให้ค่าบ่งบอกประสิทธิภาพ J ในแต่ละคลัสเตอร์ของสภาวะเน็ตเวิร์คมีค่าน้อยสุด

5. การทดสอบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์ค

ขั้นตอนนี้เป็นการนำค่า α และ β ที่ได้มาจาก SANFISII มาทดสอบกับระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คจริงโดยโครงสร้างของระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คเป็นดังภาพต่อไปนี้



ภาพที่ 33 ระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์ค

ขั้นตอนทำงานของระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คเริ่มต้นจากการตรวจสอบสภาวะเน็ตเวิร์ค ณ ขณะนั้นว่าอยู่ในคลัสเตอร์ใด โดยใช้วิธีการวัดดีเลย์ด้วยวิธีการเดียวกับขั้นตอนที่ 2 โดยจะใช้จำนวนข้อมูลที่ได้จากในการตรวจสอบ 10 ครั้ง และนำกลุ่มของข้อมูลที่ได้ไปหา Median (q_1) Variance (q_2) และ Exponential moving average (q_3)

หลังจากนั้นนำ $\mathbf{q}(t)$ เข้าไปตรวจสอบด้วย SANFIS ว่าตกอยู่ในคลัสเตอร์ใดและใช้ค่า α และ β เท่าใด หลังจากนั้นนำค่า α และ β ที่ได้เข้าไปปรับใช้กับอัลกอริทึม quadratic curve ที่ใช้อยู่งานบนโปรแกรมตัวควบคุมซึ่งโปรแกรมจะทำการส่งคำสั่งควบคุมผ่านเน็ตเวิร์คไปยังหุ่นยนต์

ในส่วนกระบวนการที่ต้องการควบคุมจะมีกล้องจับภาพและหาตำแหน่งการเคลื่อนที่ของหุ่นยนต์และส่งผลลัพท์ที่ได้ผ่านเน็ตเวิร์คกลับมายังตัวควบคุมเพื่อประมวลผลต่อไป

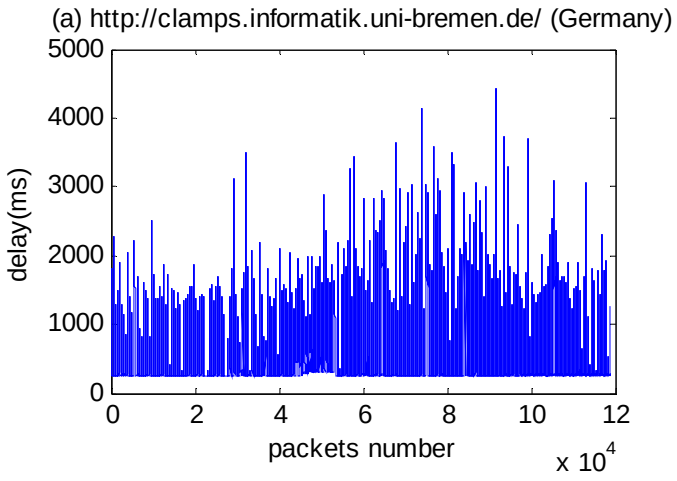
ผลและวิจารณ์

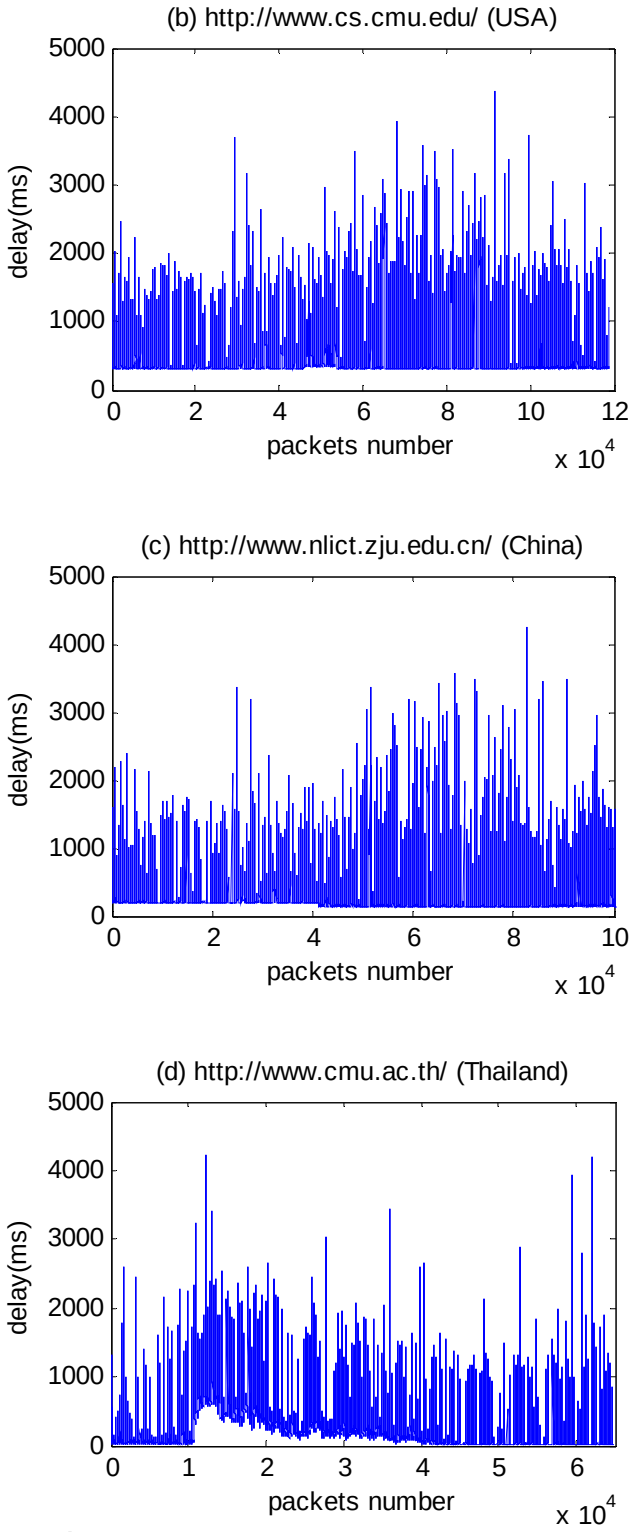
ผล

งานวิจัยนี้ได้ทำการเก็บข้อมูลดีเลย์ระหว่างห้องปฏิบัติการจักรกลอัจฉริยะ ภาควิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเกษตรศาสตร์ไปยังสถานที่ต่างๆ โดยทำการเก็บข้อมูลวันศุกร์ เสาร์ และอาทิตย์เป็นเวลาต่อเนื่องกัน 3 วัน ซึ่งสถานที่ที่สุ่มเลือกเพื่อใช้ในการเก็บข้อมูลดีเลย์เพื่อนำมาวิเคราะห์มีทั้งหมด 4 สถานที่ ดังต่อไปนี้

a. **http://clamps.informatik.uni-bremen.de/** ประเทศเยอรมัน
b. **http://www.cs.cmu.edu/** ประเทศสหรัฐอเมริกา
c. **http://www.nlict.zju.edu.cn/** ประเทศจีน
d. **http://www.cmu.ac.th/** จังหวัดเชียงใหม่ ประเทศไทย

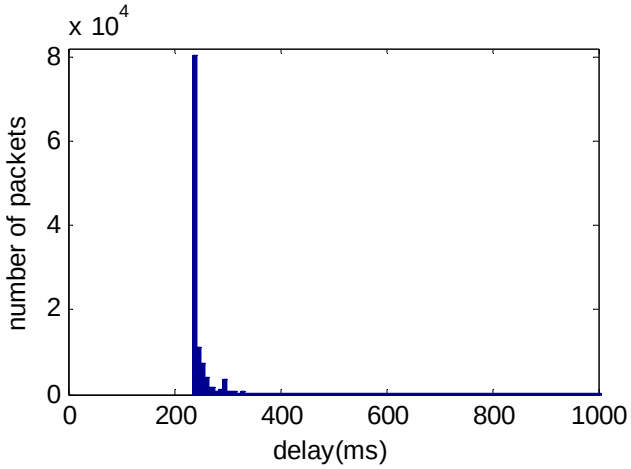
โดยรายละเอียดของข้อมูลดีเลย์จากทั้ง 4 ที่แสดงในภาพต่อไปนี้



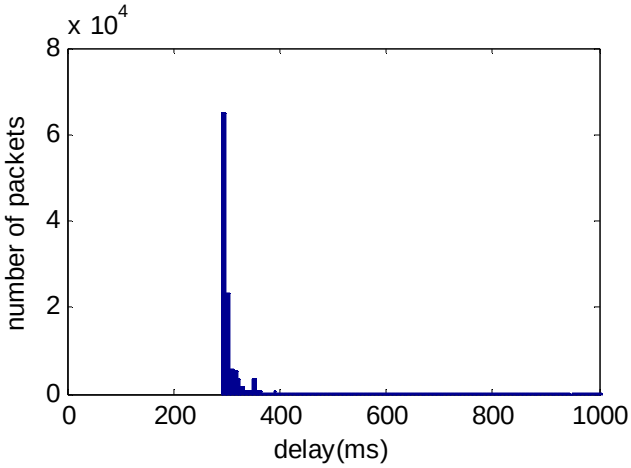


ภาพที่ 34 รายละเอียดข้อมูลดีเลย์จากสถานที่ต่างๆ

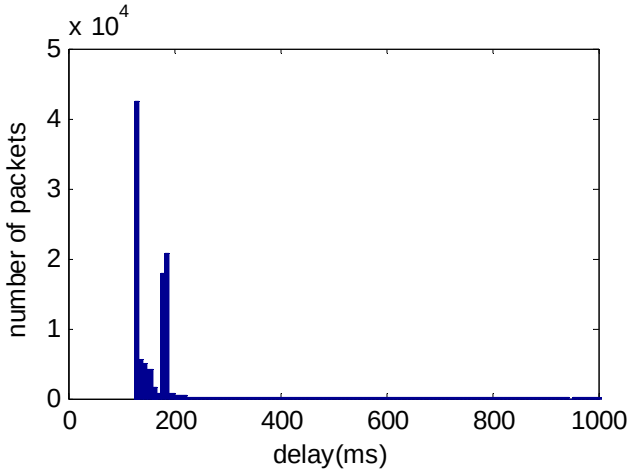
ซึ่งถ้าเรานำข้อมูลดีเลย์ของแต่ละที่มาวิเคราะห์หาการกระจายตัวจะได้ผลตามภาพต่อไปนี้



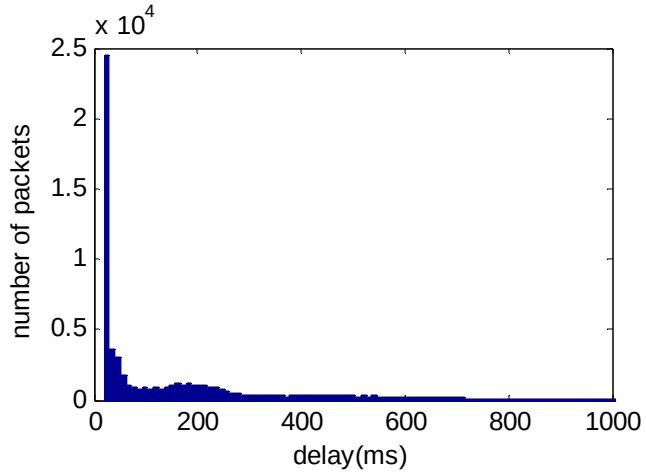
(a) **http://clamps.informatik.uni-bremen.de/** (Germany)



(b) **http://www.cs.cmu.edu/** (USA)



(c) **http://www.nlict.zju.edu.cn/** (China)

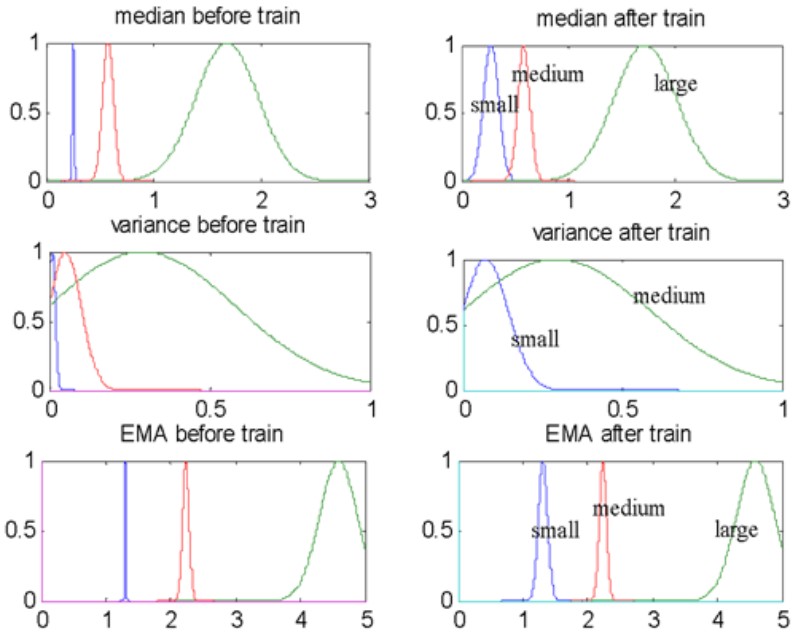


(d) **http://www.cmu.ac.th/** (Thailand)

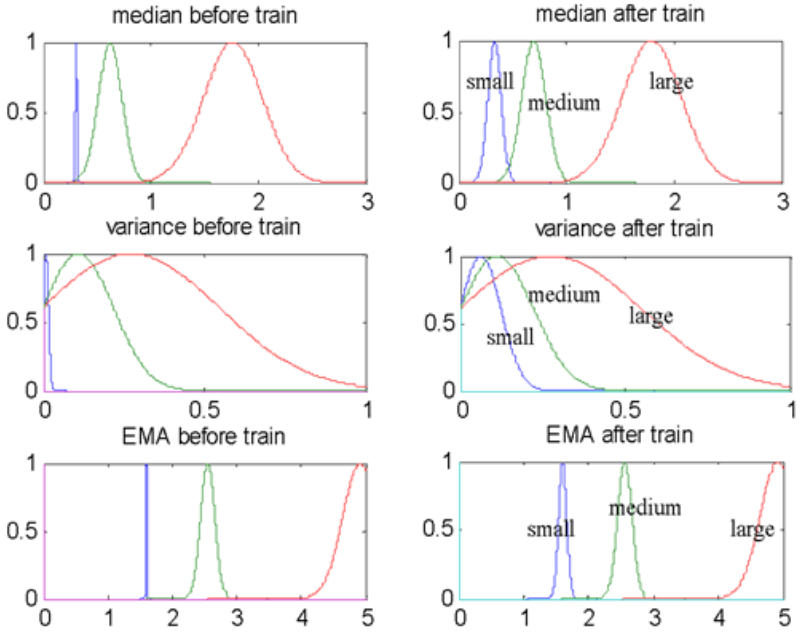
ภาพที่ 35 การกระจายตัวของดีเลย์ของแต่ละสถานที่ต่างๆ

จากการวิเคราะห์ข้อมูลเบื้องต้นจะพบว่าลักษณะการกระจายตัวของดีเลย์แต่ละสถานที่ไม่เหมือนกัน ทำให้การประเมินหรือการทำนายการเกิดดีเลย์นั้นเป็นไปได้ยาก

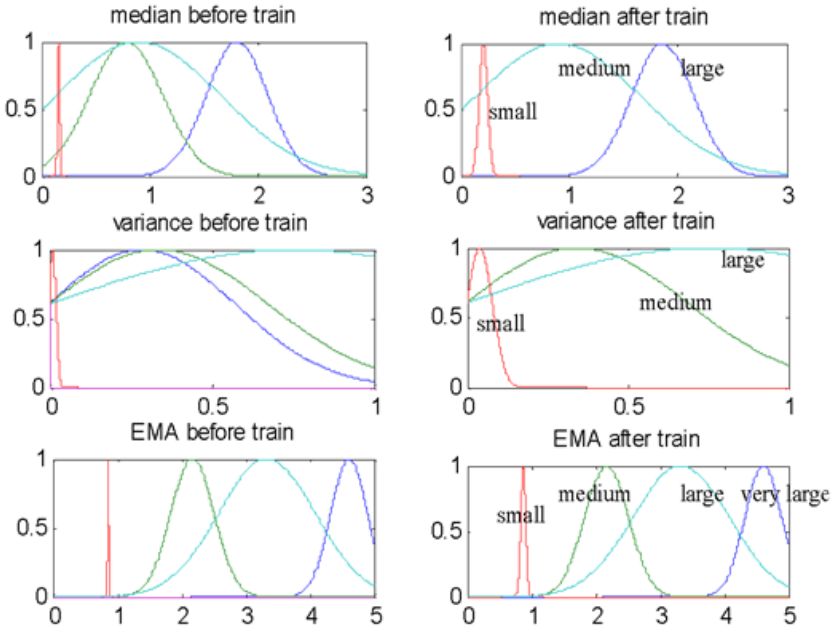
จากนั้นนำข้อมูลดีเลย์ที่ได้มาคลัสเตอร์ด้วยอัลกอริทึม MCA และเรียนรู้ด้วยอัลกอริทึม fast recursive least-squares optimization โดยภาพที่ 36 แสดงฟังก์ชันแสดงความเป็นสมาชิก (membership function) ที่ได้จากอัลกอริทึม MCA ซึ่งแสดงถึงการคาบเกี่ยวกันค่อนข้างมาก ซึ่งทำให้ไม่สามารถกำหนด rule based ที่ใช้ในการอธิบายสภาวะเน็ตเวิร์คได้อย่างชัดเจน แต่หลังจากการประมวลผลด้วยอัลกอริทึม fast recursive least-squares optimization จะเห็นได้ว่าฟังก์ชันแสดงความเป็นสมาชิกมีการคาบเกี่ยวกันน้อยลงและมีช่วงที่กว้างขึ้นซึ่งจะทำให้สามารถกำหนด rule based และสามารถอธิบายสภาวะของเน็ตเวิร์คได้ง่ายขึ้น



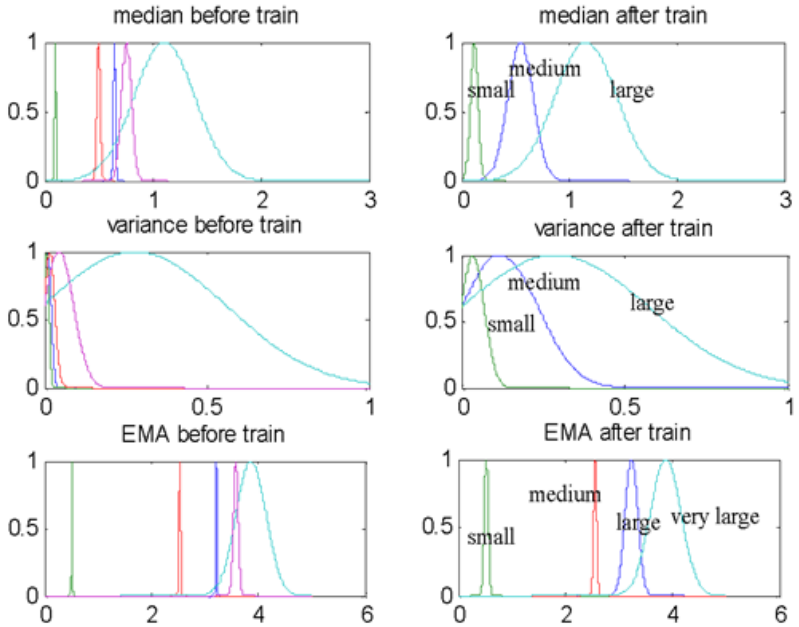
(a) http://clamps.informatik.uni-bremen.de/ (Germany)



(b) http://www.cs.cmu.edu/ (USA)



(c) http://www.nlict.zju.edu.cn/ (China)



(d) http://www.cmu.ac.th/ (Thailand)

ภาพที่ 36 ฟังก์ชันแสดงความเป็นสมาชิกก่อนการเรียนรู้และหลังการเรียนรู้

จากฟังก์ชันแสดงความเป็นสมาชิกและชุดของข้อมูลที่ได้มาหลังจากการเรียนรู้ด้วยอัลกอริทึม fast Recursive least-squares optimization จะสามารถสร้างกฎเพื่อใช้ทำการแบ่งสภาวะของเน็ตเวิร์คโดยดูจากเอาท์พุต (y) โดยแต่ละชุดข้อมูลต้องทำการ normalized เอาท์พุตให้อยู่ในช่วง [0,1] ก่อน โดยจากคลัสเตอร์ที่ได้หลังจากทำการเรียนรู้ด้วยอัลกอริทึม fast recursive least-squares optimization ของทุกๆสถานที่ที่ทำการเก็บมูลได้ออกมา 3 คลัสเตอร์ เพราะฉะนั้นเพื่อความง่ายต่อการเข้าใจเราจะแบ่งสภาวะเน็ตเวิร์คเป็น ดี(good) ปานกลาง(medium) และแย่ (bad) โดยจะทำการแบ่งสภาวะเน็ตเวิร์คเป็น 3 ช่วงเท่าๆกันจากช่วง [0,1] ซึ่งจะกฎที่นิยามแสดงในสมการต่อไปนี้

$$\text{Class of Network} = \begin{cases} \text{Good, if } 0 \le J \le 0.33 \\ \text{Medium, if } 0.33 < J \le 0.67 \\ \text{Bad, if } J > 0.67 \end{cases} \tag{106}$$

โดยกฎในการแบ่งสภาวะของแต่ละชุดข้อมูลแสดงในตารางต่อไปนี้

ตารางที่ 10 rule based จากข้อมูลชุด **http://clamps.informatik.uni-bremen.de/**

Rule	Rule based
1	IF q_1 is small and q_2 is medium and q_3 is small THEN *network* is *good*
2	IF q_1 is large and q_2 is large and q_3 is large THEN *network* is *bad*
3	IF q_1 is medium and q_2 is medium and q_3 is medium THEN *network* is *medium*

ตารางที่ 11 rule based จากข้อมูลชุด **http://www.cs.cmu.edu/**

Rule	Rule based
1	IF q_1 is small and q_2 is small and q_3 is small THEN *network* is *good*
2	IF q_1 is medium and q_2 is medium and q_3 is medium THEN *network* is *medium*
3	IF q_1 is large and q_2 is large and q_3 is large THEN *network* is *bad*

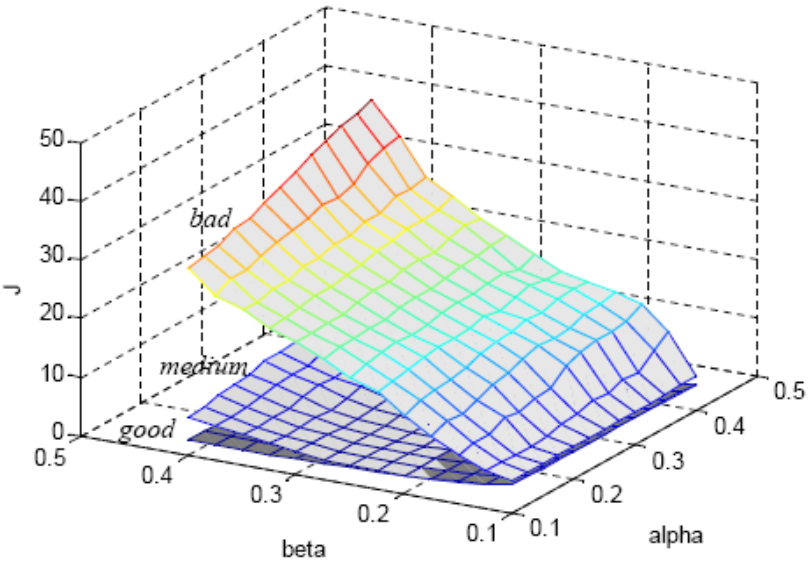
ตารางที่ 12 rule based จากข้อมูลชุด **http://www.nlict.zju.edu.cn/**

Rule	Rule based
1	IF q_1 is large and q_2 is small and q_3 is very large THEN *network* is *bad*
2	IF q_1 is medium and q_2 is medium and q_3 is large THEN *network* is *medium*
3	IF q_1 is small and q_2 is small and q_3 is small THEN *network* is *good*
4	IF q_1 is medium and q_2 is medium and q_3 is very large THEN *network* is *medium*

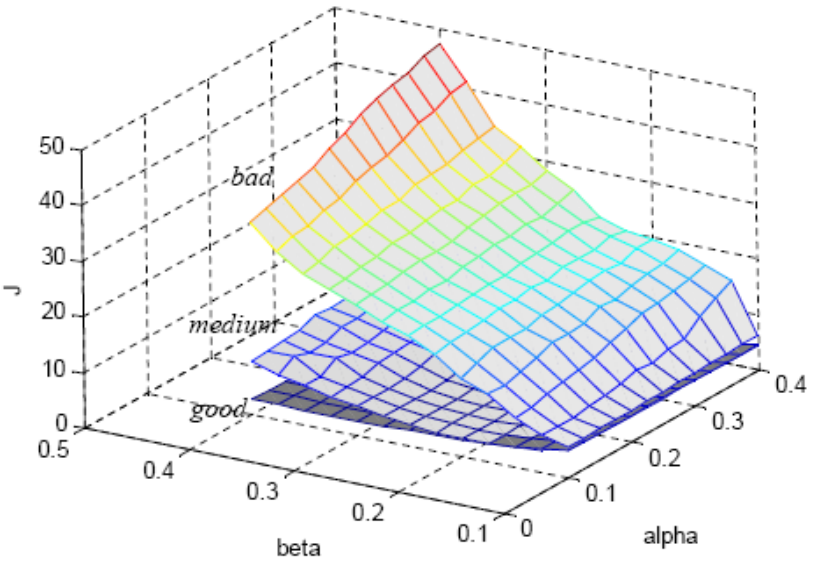
ตารางที่ 13 rule based จากข้อมูลชุด **http://www.cmu.ac.th/**

Rule	Rule based
1	IF q_1 is medium and q_2 is small and q_3 is large THEN *network* is *medium*
2	IF q_1 is small and q_2 is small and q_3 is small THEN *network* is *good*
3	IF q_1 is medium and q_2 is small and q_3 is medium THEN *network* is *medium*
4	IF q_1 is large and q_2 is medium and q_3 is very large THEN *network* is *bad*
5	IF q_1 is large and q_2 is small and q_3 is very large THEN *network* is *bad*

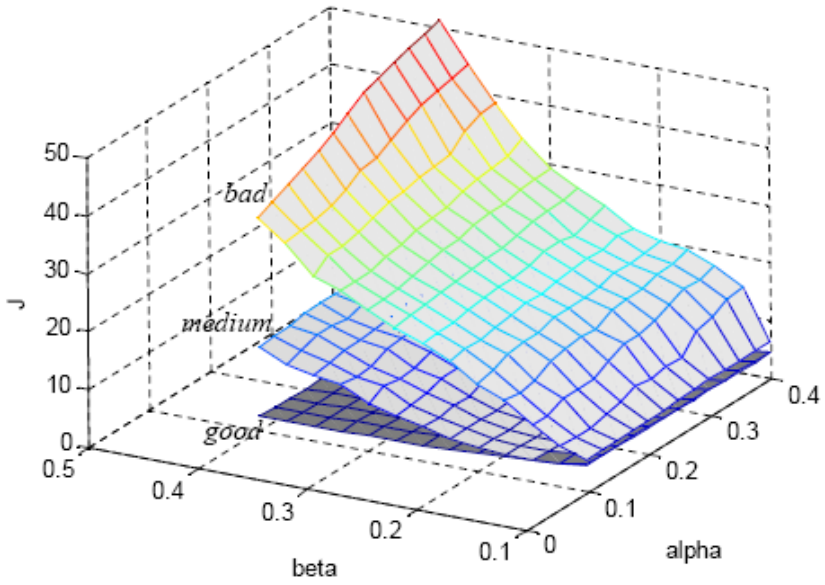
หลังจากนั้นจะทำการหาค่า α และ β ที่เหมาะสมกับแต่สภาวะเน็ตเวิร์คเพื่อใช้ปรับค่าให้กับตัวควบคุม quadratic curve ซึ่งได้นำเสนอวิธีการในการหาค่าที่เหมาะสมไว้ในหัวข้อก่อนนี้ โดยกราฟต่อไปนี้แสดงถึงค่า α และ β ที่มีผลต่อค่าบ่งบอกประสิทธิภาพ (J) ในแต่ละกลุ่มของสภาวะเน็ตเวิร์คสำหรับแต่ละสถานที่



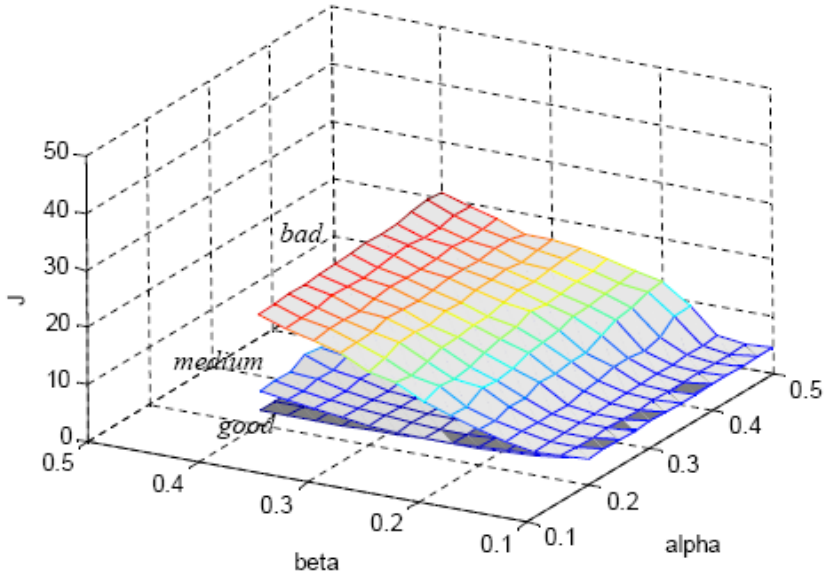
(a) **http://clamps.informatik.uni-bremen.de/ (Germany)**



(b) **http://www.cs.cmu.edu/ (USA)**



(c) http://www.nlict.zju.edu.cn/ (China)



(d) http://www.cmu.ac.th/ (Thailand)

ภาพที่ 37 ค่าบ่งบอกประสิทธิภาพ J ที่สอดคล้องกับค่า α และ β ของแต่ละสถานที่

จากกราฟในภาพแสดงถึงค่า α และ β ที่มีผลต่อ J ในแต่ละกลุ่มของสภาวะเน็ตเวิร์คในแต่ละคลัสเตอร์ โดยค่า α และ β ที่เหมาะสมที่สุดคือค่าที่น้อยที่สุดในแต่ละคลัสเตอร์ตามที่ได้กล่าวมาข้างต้น ซึ่งค่าที่เหมาะสมสำหรับแต่ละคลัสเตอร์ได้นำเสนอดังตารางต่อไปนี้

ตารางที่ 14 ค่า α และ β ที่เหมาะสมสำหรับ **http://clamps.informatik.uni-bremen.de/**

Gain \ Cluster	G	M	B
α	0.3	0.150	0.1
β	0.375	0.4	0.1

ตารางที่ 15 ค่า α และ β ที่เหมาะสมสำหรับ **http://www.cs.cmu.edu/**

Gain \ Cluster	G	M	B
α	0.3	0.175	0.1
β	0.4	0.4	0.1

ตารางที่ 16 ค่า α และ β ที่เหมาะสมสำหรับ **http://www.nlict.zju.edu.cn/**

Gain \ Cluster	G	M	B
α	0.375	0.15	0.1
β	0.4	0.4	0.1

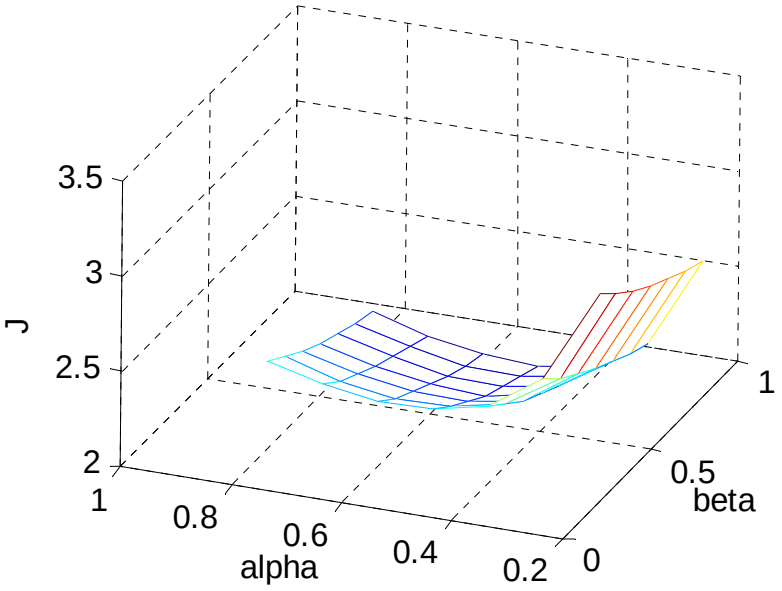
ตารางที่ 17 ค่า α และ β ที่เหมาะสมสำหรับ **http://www.cmu.ac.th/**

Gain \ Cluster	G	M	B
α	0.4	0.225	0.125
β	0.4	0.4	0.275

หลังจากได้ค่า α และ β ที่เหมาะสมสำหรับแต่ละคลัสเตอร์แล้ว นำค่าพารามิเตอร์ที่ได้มาทำการทดสอบเพื่อแสดงให้เห็นประสิทธิภาพในการควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คโดยการประยุกต์ใช้ SANFIS กับตัวควบคุม quadratic curve โดยทำการทดสอบบนโปรแกรม Matlab7.0 และจะทำการสุ่มดีเลย์ตลอดเส้น โดยจะทำการทดสอบด้วยกัน 2 แบบคือ

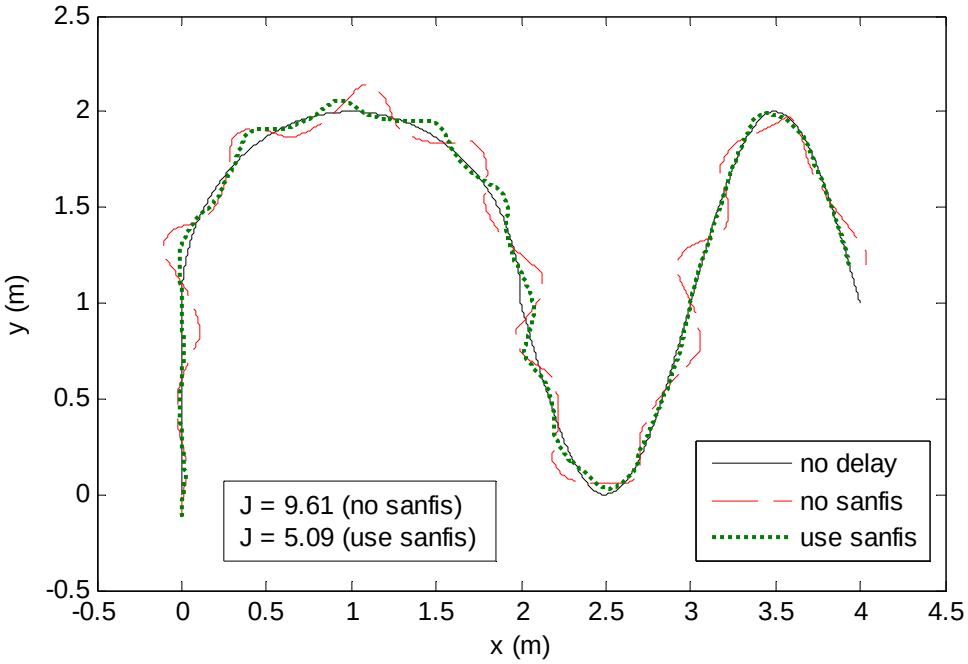
1. การควบคุมผ่านเน็ตเวิร์คแบบใช้ neuro-fuzzy gain scheduling
2. การควบคุมผ่านเน็ตเวิร์คแบบไม่ใช้ neuro-fuzzy gain scheduling

ซึ่งการควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คแบบไม่ใช้ neuro-fuzzy gain scheduling จะเลือกใช้ค่าพารามิเตอร์ α และ β ที่ดีที่สุดในสภาวะที่ไม่เกิดดีเลย์โดยค่า α และ β จะหาได้จากกราฟแสดงประสิทธิภาพขณะที่ไม่มีดีเลย์ซึ่งแสดงในกราฟต่อไปนี้

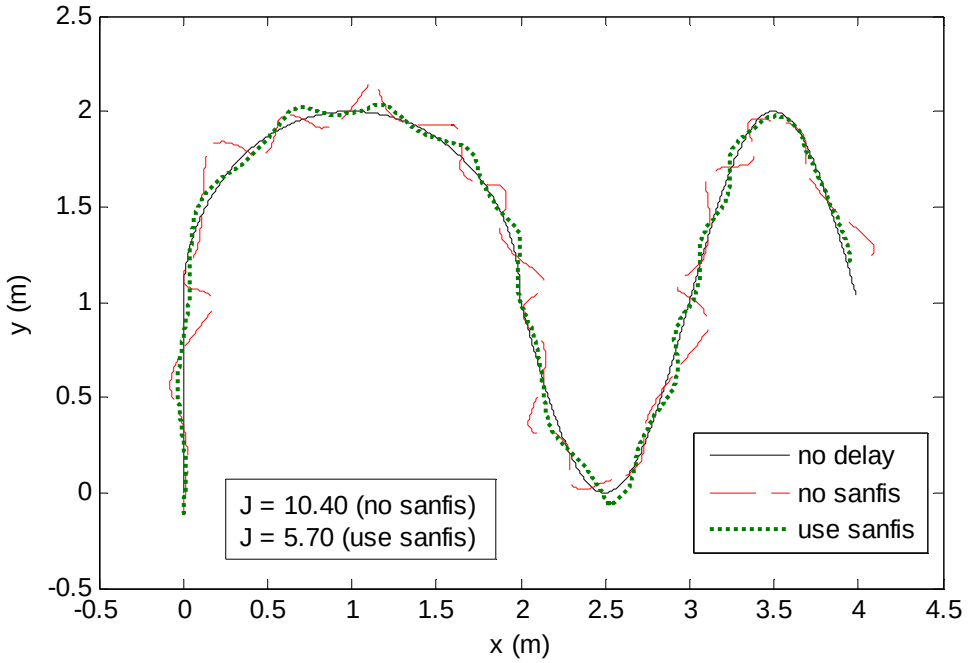


ภาพที่ 38 ค่าบ่งบอกประสิทธิภาพ J ที่สอดคล้องกับค่า α และ β ในขณะที่ไม่มีดีเลย์

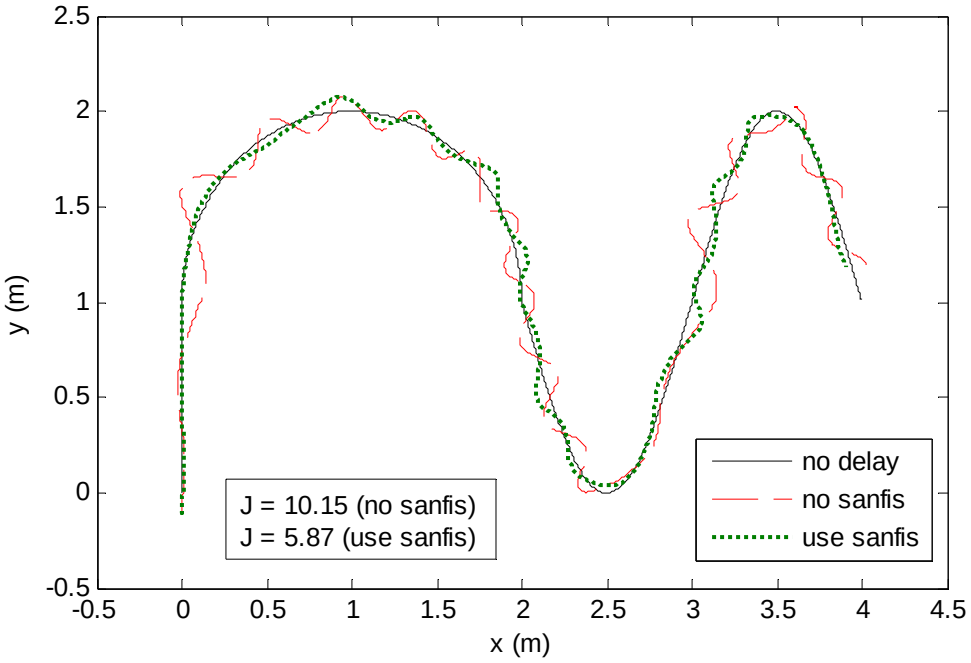
โดยผลการทดสอบการควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คสำหรับข้อมูลดีเลย์แต่ละชุดนำเสนอดังภาพต่อไปนี้



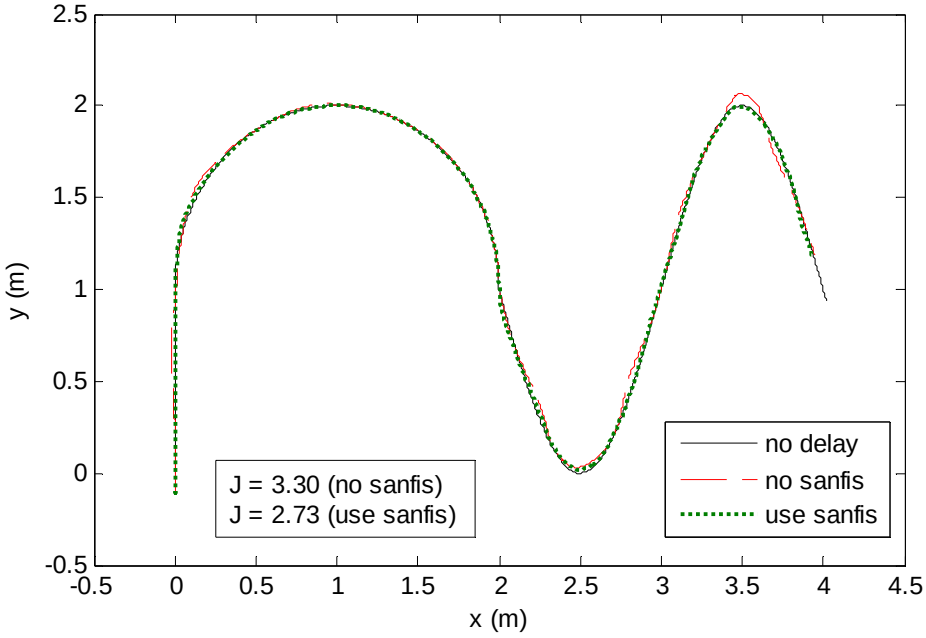
(a) **http://clamps.informatik.uni-bremen.de/ (Germany)**



(b) **http://www.cs.cmu.edu/ (USA)**



(c) **http://www.nlict.zju.edu.cn/ (China)**



(d) **http://www.cmu.ac.th/ (Thailand)**

ภาพที่ 39 ผลการทดสอบของการเดินตามเส้นทางของหุ่นยนต์

จากการทดสอบระบบนิวโรฟัซซี่กับตัวควบคุมผ่านเน็ตเวิร์คไร้สายสำหรับหุ่นยนต์เคลื่อนที่ทั้งการทดสอบบนโปรแกรมจำลอง Matlab7.0 จะเห็นได้ว่าค่า α และ β ที่ได้มาทำให้

ประสิทธิภาพของระบบดีขึ้นโดยสังเกตจากการเดินตามเส้นทางของหุ่นยนต์จากภาพที่ 39 และถ้านำผลที่ได้จากการทดสอบมาหาค่าบ่งบอกประสิทธิภาพซึ่งค่าที่ได้ถูกนำเสนอในตารางต่อไปนี้

ตารางที่ 18 แสดงค่าบ่งบอกประสิทธิภาพ (*J*) หลังจากทำการทดสอบระบบ

การทดสอบ\ ค่าบ่งบอกประสิทธิภาพ	without Neuro-Fuzzy	with Neuro-Fuzzy
http://clamps.informatik.uni-bremen.de/	9.61	5.09
http://www.cs.cmu.edu/	10.40	5.70
http://www.nlict.zju.edu.cn/	10.15	5.87
http://www.cmu.ac.th/	3.30	2.73

สังเกตจากค่าบ่งบอกประสิทธิภาพ (J) ซึ่งได้นำเสนอตามตารางที่ 18 ซึ่งจะเห็นจากค่าบ่งบอกประสิทธิภาพว่าการควบคุมแบบใช้ neuro-fuzzy NBC gain scheduling ในระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คไร้สายทำให้ค่าบ่งบอกประสิทธิภาพมีค่าลดลงซึ่งหมายถึงหุ่นยนต์สามารถเดินตามเส้นทางได้แม่นยำขึ้นและใช้เวลาในการเดินตามเส้นทางน้อยลง เมื่อเปรียบเทียบกับการควบคุมแบบไม่ใช้ neuro-fuzzy NBC gain scheduling ซึ่งแสดงให้เห็นถึงประสิทธิภาพที่ดีขึ้นภายใต้การเกิดดีเลย์บนเน็ตเวิร์ค

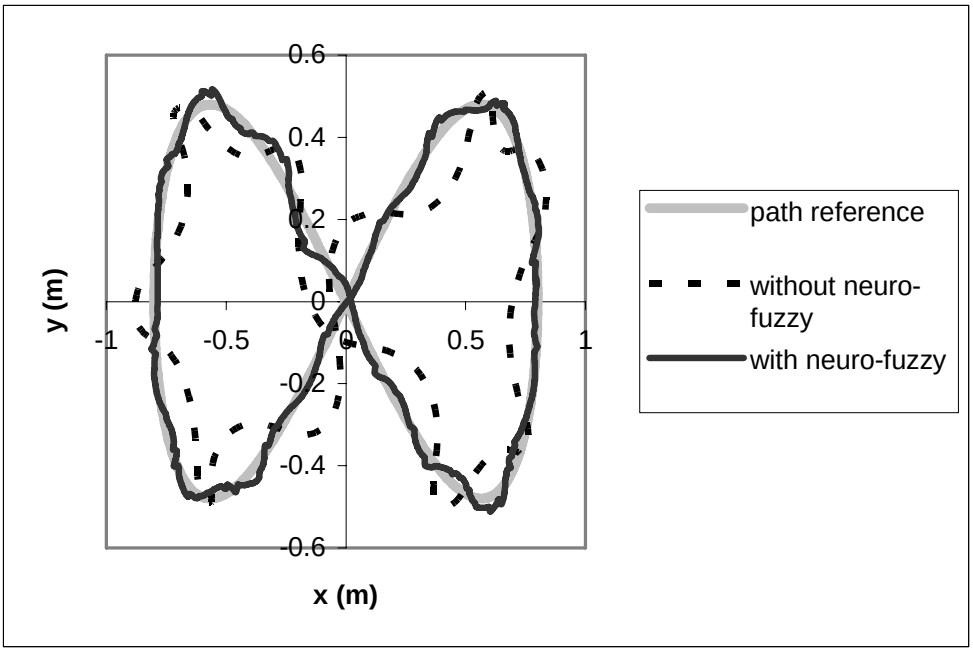
การทดสอบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์ค ไร้สาย

การทดสอบเพื่อแสดงให้เห็นประสิทธิภาพในควบคุมผ่านเน็ตเวิร์คโดยการประยุกต์ใช้ SANFIS กับตัวควบคุม quadratic curve โดยใช้ระบบทดสอบที่ได้อธิบายในวิธีการข้างต้นและเลือกใช้ชุดข้อมูล delay จาก http://www.nlict.zju.edu.cn/ สำหรับการทดลองควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คจริง เนื่องจากการวิเคราะห์เบื้องต้นพบว่าข้อมูลชุดนี้มีการกระจายตัวมากที่สุดจึงน่าจะเป็นกรณีศึกษาได้ดีที่สุด

โดยทำการทดสอบ 2 ลักษณะคือ

1) การควบคุมผ่านเน็ตเวิร์คแบบใช้ neuro-fuzzy gain scheduling
2) การควบคุมผ่านเน็ตเวิร์คแบบไม่ใช้ neuro-fuzzy gain scheduling

โดยผลการทดสอบนำเสนอในภาพต่อไปนี้



ภาพที่ 40 ผลการทดสอบการควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คไร้สาย

สังเกตจากภาพที่ 40 ซึ่งจะเห็นจากค่าบ่งบอกประสิทธิภาพว่าการควบคุมแบบใช้ neuro-fuzzy NBC gain scheduling ในระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คไร้สายทำให้หุ่นยนต์สามารถเดินตามเส้นทางได้แม่นยำขึ้นเมื่อเปรียบเทียบกับการควบคุมแบบไม่ใช้ neuro-fuzzy NBC gain scheduling ซึ่งแสดงให้เห็นถึงประสิทธิภาพที่ดีขึ้นภายใต้การเกิดดีเลย์บนเน็ตเวิร์ค

วิจารณ์

การควบคุมผ่านเน็ตเวิร์คหรือ network-based control (NBC) ได้ถูกนำไปประยุกต์ใช้อย่างแพร่หลายแทนการควบคุมแบบเดิม ยกตัวอย่างเช่น การควบคุมอุปกรณ์ระยะไกลผ่านเน็ตเวิร์ค การตรวจสอบการทำงานของอุปกรณ์ผ่านเน็ตเวิร์คเป็นต้น แต่ปัญหาที่สำคัญของระบบควบคุมผ่านเน็ตเวิร์คก็คือดีเลย์ (delay) ที่เกิดขึ้นจากระบบเน็ตเวิร์ค ซึ่งทำให้ระบบเสียเสถียรภาพและระบบมีประสิทธิภาพต่ำลง โดยลักษณะของดีเลย์ที่เกิดขึ้นมีลักษณะกำกวมและไม่สามารถคาดคะเนได้

ถ้าเรามาวิเคราะห์ข้อมูลดีเลย์ในภาพที่ 34 และข้อมูลการกระจายตัวของดีเลย์ในภาพที่ 35 จะเห็นได้ว่าข้อมูลที่เก็บมาแต่ละที่มีลักษณะที่ไม่เหมือนกันเลย และถ้ามาพิจารณาลักษณะการกระจายตัวของดีเลย์ในภาพที่ 35 จะสังเกตได้ว่ารูปแบบการกระจายตัวของทุกๆที่ไม่สามารถอธิบายได้ด้วยรูปแบบการกระจายตัวปกติ ยกตัวอย่างเช่น การกระจายตัวแบบ exponential หรือการกระจายตัวแบบ poisson สาเหตุเนื่องจากสภาวะของเน็ตเวิร์คคลุมเครือและยากที่จะอธิบายได้

จากข้อมูลดิบในแต่ละที่ถ้ามาวิเคราะห์สภาวะเน็ตเวิร์คในข้อมูลแต่ละชุด จะสังเกตได้ว่าข้อมูลชุดที่ (a) และข้อมูลชุดที่ (b) ซึ่งนั่นก็คือดีเลย์จากที่ http://clamps.informatik.uni-bremen.de/ ประเทศเยอรมัน และดีเลย์จากที่ http://www.cs.cmu.edu/ ประเทศสหรัฐอเมริกา การกระจายตัวของข้อมูลจะมีลักษณะใกล้เคียงกันมากซึ่งสังเกตได้จากภาพที่ 35(a) และ 35(b) โดยชุดข้อมูลเยอรมันดีเลย์ส่วนใหญ่จะอยู่ที่ประมาณ 230 มิลลิวินาทีและจากชุดข้อมูลสหรัฐอเมริกาจะอยู่ที่ประมาณ 300 มิลลิวินาที ซึ่งถือว่าเป็นค่าดีเลย์ที่ค่อนข้างสูงในระบบควบคุม สาเหตุที่ทำให้ดีเลย์มีค่าสูงเนื่องมาจากระยะทางที่ไกล แต่ถ้าสังเกตการกระจายตัวของข้อมูลในภาพที่ 35(a), 35(b) และจากลักษณะของคลัสเตอร์ที่ได้หลังจากการเรียนรู้ในภาพที่ 36(a), 36(b) จะเห็นว่าการกระจายตัวของฟังก์ชันแสดงความเป็นสมาชิกของเทรนนิ่งเซตทั้งค่า mean variance และ exponential moving average ของทั้ง 2 กลุ่มนี้มีค่าต่ำโดยดูได้จากฟังก์ชันแสดงความเป็นสมาชิกที่แคบซึ่งก็หมายถึงสภาวะเน็ตเวิร์คจากมหาวิทยาลัยเกษตรศาสตร์ไปยังทั้ง 2 ที่มีเสถียรภาพค่อนข้างสูง

หากวิเคราะห์สภาวะเน็ตเวิร์คในข้อมูลชุดที่ (c) นั่นก็คือดีเลย์จากที่ http://www.nlict.zju.edu.cn/ ประเทศจีน จากภาพที่ 35(c) จะเห็นว่าลักษณะดีเลย์ของชุดข้อมูลนี้จะหนาแน่นอยู่ 2

ช่วง คือช่วงประมาณ 170 มิลลิวินาทีและช่วง 200 มิลลิวินาที ซึ่งมีค่าน้อยกว่าที่เยอรมันนีและสหรัฐอเมริกาเนื่องจากระยะทางที่ใกล้กว่า แต่จากลักษณะของคลัสเตอร์ที่ได้หลังจากการเรียนรู้ในภาพที่ 36(c) จะเห็นได้ว่าการกระจายตัวของฟังก์ชันแสดงความเป็นสมาชิกมีค่าสูงมากอาจจะเป็นเพราะว่ามีผู้ใช้เน็ตเวิร์คในเส้นทางนี้มีมาก

และถ้ามาวิเคราะห์สภาวะเน็ตเวิร์คในข้อมูลชุดที่ (d) นั่นก็คือดีเลย์จากที่ http://www.cmu.ac.th/ จังหวัดเชียงใหม่ จากภาพที่ 35(d) และภาพที่ 34(d) จะเห็นว่าลักษณะดีเลย์ของชุดข้อมูลนี้ส่วนใหญ่ดีเลย์จะมีค่าประมาณ 30 มิลลิวินาทีเนื่องจากระยะทางที่ใกล้มาก แต่ข้อมูลชุดนี้มีกระจายตัวเป็นช่วงกว้างมากจากช่วงประมาณ 30 ถึง 300 มิลลิวินาทีซึ่งจากข้อมูลอาจเป็นไปได้ว่าจำนวนคนที่ใช้งานในเส้นทางนี้เปลี่ยนแปลงตามช่วงของเวลาอย่างชัดเจน

จากสภาวะเน็ตเวิร์คในแต่ละที่ที่แตกต่างกันสิ้นเชิง งานวิจัยนี้ได้นำเสนอวิธีการแก้ปัญหาดังกล่าวโดยใช้ระบบ neuro-fuzzy gain scheduling เพื่อเพิ่มประสิทธิภาพให้กับอัลกอริทึม quadratic curve ซึ่งเป็นตัวควบคุมให้หุ่นยนต์เดินตามเส้นทางโดยได้มีการประยุกต์ใช้ self–adaptive neuro-fuzzy Inference system (SANFIS) เพื่อช่วยในการจัดกลุ่มสภาวะของเน็ตเวิร์คและใช้ gain scheduler ในการปรับค่าพารามิเตอร์ในตัวควบคุมให้เหมาะสมกับสภาวะของเน็ตเวิร์ค ณ ตอนนั้น โดยระบบทั้งหมดถูกทดสอบกับการเดินตามเส้นทางของหุ่นยนต์ผ่านเน็ตเวิร์คไร้สายซึ่งทำการทดสอบกับทั้งระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คจำลอง (simulation) และระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คจริงซึ่งในการทดสอบได้มีการเปรียบเทียบระหว่างการควบคุมที่ใช้และไม่ใช้ neuro-fuzzy gain scheduling

การทดสอบทั้งระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คจำลองและระบบระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คจริงได้พบว่าการควบคุมที่ประยุกต์ใช้ neuro-fuzzy gain scheduling ทำให้ประสิทธิภาพในระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คดีขึ้น โดยหุ่นยนต์สามารถเดินตามเส้นทางได้แม่นยำขึ้นและเร็วขึ้น

สรุปและข้อเสนอแนะ

สรุป

จากการทดสอบระบบนิวโรฟัซซี่กับตัวควบคุมผ่านเน็ตเวิร์คไร้สายสำหรับหุ่นยนต์เคลื่อนที่ทั้งการทดสอบบนโปรแกรมจำลอง Matlab/Simulink7.0 และการทดสอบระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คจริงจะเห็นได้ว่าค่า α และ β ที่ได้มาการจากการเรียนรู้ด้วย SANFIS ทำให้ประสิทธิภาพของระบบดีขึ้นโดยสังเกตจากการเดินตามเส้นทางของหุ่นยนต์จากภาพที่ 39 และ 40 หรือสังเกตจากค่าบ่งบอกประสิทธิภาพซึ่งได้นำเสนอตามตารางที่ 18 ซึ่งจะเห็นจากค่าบ่งบอกประสิทธิภาพว่าการควบคุมแบบใช้ neuro-fuzzy NBC gain scheduling ในระบบควบคุมหุ่นยนต์ผ่านเน็ตเวิร์คไร้สายทำให้ค่าบ่งบอกประสิทธิภาพมีค่าลดลงซึ่งหมายถึงหุ่นยนต์สามารถเดินตามเส้นทางได้แม่นยำขึ้นและใช้เวลาในการเดินตามเส้นทางน้อยลง เมื่อเปรียบเทียบกับการควบคุมแบบไม่ใช้ neuro-fuzzy NBC gain scheduling ซึ่งแสดงให้เห็นถึงประสิทธิภาพที่ดีขึ้นภายใต้การเกิดดีเลย์บนเน็ตเวิร์ค

ข้อเสนอแนะ

เพื่อเพิ่มประสิทธิภาพของระบบให้สามารถตอบสนองกับสภาวะเน็ตเวิร์คที่เปลี่ยนอยู่ตลอด ระบบควรสามารถปรับเปลี่ยนพารามิเตอร์ต่างๆได้โดยการทำคลัสเตอร์ริ่งแบบ online ซึ่งเป็นแนวทางการทำวิจัยที่น่าสนใจในอนาคต

เอกสารและสิ่งอ้างอิง

Brown, Robert Grove and Patrick Y.C. Hwang. 1997. Introduction to random signals and applied Kalman filtering. **John Wiley & Sons Book**.

Bruce, J. , Veloso, M. 2003. Fast and accurate vision-based pattern detection and identification. pp. 1277-1282. **Robotics and Automation 2003, ICRA '03.**

Chan, H. and H. Ozguner. 1995. Closed-loop control of systems over a communication network with queues, vol. 62, pp. 493-510. **International Journal Control.**

Chow, M. Y. and Y. Tipsuwan. 2001. Network-based control systems: a tutorial, vol. 3, pp. 1593 - 1602. **IEEE IECON '01 Industrial Electronics Society.**

Goktas, F. 2000. Distributed control of systems over communication networked control system, **Ph.D. dissertation**, **University Pennsylvania**, **Philadelphia, PA**

Gupta, M. M., L. Jin. and N. Homma. 2003. **Static and Dynamic Neural Networks** John Wiley & Sons 722 p.

Hong, S. H. 1995. Scheduling algorithm of data sampling times in the integrated communication and control systems, vol. 3, pp. 225-230. **IEEE Transactions on Control Systems Technology.**

Jang, J.-S. R. 1993. ANFIS: adaptive-network-based fuzzy inference system, vol. 23, pp. 665-685. **IEEE Transactions on Systems, Man and Cybernetics.**

Kalman, R.E. 1960. A new approach to linear filtering and prediction problems, p. 35-45. **Transaction of the ASME, Journal of Basic Engineering.**

Koichi Yoshizawa, Hideki Hashimoto, Masayoshi Wada, and Shunji Mori. 1996. Path Tracking Control of Mobile Robots Using a Quadratic Curve, vol. 40, pp. 58-63. **IEEE Transactions on Intelligent Vehicles.**

Kuo, B. C. 1987. **Automatic Control Systems**, 5 ed. Englewood Cliffs, NJ: Prentice-Hall

Leonard, A. McGee and Stanley F. Schmidt. 1985. Discovery of the Kalman Filter as a practical tool for aerospace and industry, id86847. **NASA Technical Memorandum**.

Lin, C.-T. and C. S. G. Lee. 1991. Neural-network-based fuzzy logic control and decision system, vol. 40, pp. 1320-1336. **IEEE Transactions on Computers.**

Luck, R. and A. Ray. 1994. Experimental verification of a delay compensation algorithm for integrated communication and control systems, vol. 59, pp. 1357-1372. **International Journal of Control.**

Mauer, G. F. and C. Fernando. 1999. Remote controlled mobile imaging in a high temperature tunnel environment, pp. 212-213. **The 20th IEEE Real-Time Systems Symposium.**

Ngia, L.S.H. and Sjoberg, J. 2000. Efficient training of neural nets for nonlinear adaptive filtering using a recursive Levenberg-Marquardt algorithm, vol. 48, pp. 1915-1927. **IEEE Transactions on Signal Processing.**

Nilsson, J. 1998. Real-time control systems with delays, **Ph.D. dissertation, Lund Institute of Technology, Lund, Sweden.**

Ross, J. T. 1995. **Fuzzy Logic with Engineering Applications** McGraw-Hill 600 p.

Tipsuwan, Y. and M.-Y. Chow. 2004. On the gain scheduling for networked PI controller over IP network, vol. 9, pp. 491-498. **IEEE/ASME Transaction on Mechatronics.**

Walsh, G. C., H. Ye and L. G. Bushnell. 2002. Stability analysis of networked control systems, vol. 10, pp. 438-446. **IEEE Transactions on control system Technology.**

Wang, J.-S. and C. S. G. Lee. 2002. Self-adaptive neuro-fuzzy inference systems for classification applications, vol. 10, pp. 790-802. **IEEE Transactions on Fuzzy Systems.**

Zhang, W. B. M. S. and P. S. M. 2001. Stability of networked control systems, vol. 21, pp. 84-99. **IEEE Control Systems Magazine.**

ภาคผนวก

ภาคผนวก ก

โปรแกรมคำนวณอัลกอริทึม MCA

```
cluster_data.m

clear all;
clc;

%load q(t)
load mevama_data_1.mat;

%load v(t)
load J_01.mat;

jt = [t J(:,3)];
x  = jt(:,[1:3]); % input data
y  = jt(:,4);     % output data

i_set = x; % input set
o_set = y; % output set

% Step M1: Initialize the system with zero cluster
K=0;
[N,col_i]=size(i_set); % N = number of rows
[N,col_o]=size(o_set); % col_i,o = number of colums

% Step M2: Initialize data
K_max = 4;
K     = K_max;
cluster_i = cell(K_max,1); % for input Kmax cluster
cluster_o = cell(K_max,1); % for output Kmax cluster
c_k = ones(K,1); % ck = 1 for all cluster
m_i = i_set(1:K,:) % input space of Kmax number

for i=1:K
    %i
    %ui = {i_set(i,:)}
    %uo = {o_set(i,:)}

    %ui = i_set(i,:)
    %uo = o_set(i,:)
    cluster_i(i,1) = {i_set(i,:)}; % cluster center = each data of Kmax
    cluster_o(i,1) = {o_set(i,:)};
end

v_i = zeros(K,col_i);
m_o = o_set(1:K,:);
v_o = zeros(K,col_o);

% Step M3: Creation of distance matrix
D_i = zeros(K,K);
D_o = zeros(K,K);
D_closest_i = inf;
D_closest_o = inf;

for c1=[1:K]
    for c2=[c1:K]
        c1;
        c2;
        m_i(c1,:)
        m_i(c2,:)
        D_i(c1,c2)=norm(m_i(c1,:)-m_i(c2,:));
        D_o(c1,c2)=norm(m_o(c1,:)-m_o(c2,:));
        if ((D_i(c1,c2)<D_closest_i) && (c1 ~= c2))
            D_closest_i = D_i(c1,c2);
            closest_cluster_i = [c1 c2];
        end
        if ((D_o(c1,c2)<D_closest_o ) && (c1 ~= c2))
            D_closest_o=D_o(c1,c2);
            closest_cluster_o = [c1 c2];
        end
    end
end
```

```
% Step M4: Data Query
Initial_K = K;

for d1=[Initial_K+1:N] % Start at K+1 (data that are not the cluster seeds)
    d1
    x1 = i_set(d1,:)
    y1 = o_set(d1)

    % Step M5: Winner cluster determination
    % Compute distance between the input (output) and
    % cluster centers.
    cost_i = zeros(N,1);
    cost_o = zeros(N,1);
    cost_i_min = inf;
    cost_o_min = inf;
    winner_i = 1;
    winner_o = 1;

    for c1 = [1:K]
        c1;
        m_i(c1,:);
        x1;
        cost_i(c1) = norm((m_i(c1,:)-x1));
        cost_o(c1) = norm((m_o(c1,:)-y1));
        if cost_i(c1)<cost_i_min
            cost_i_min = cost_i(c1);
            winner_i = c1;
        end
        if cost_o(c1)<cost_o_min
            cost_o_min = cost_o(c1);
            winner_o = c1;
        end
    end

    winner_i;
    winner_o;
    % Step M6: Mapping consistency verification

    if winner_i == winner_o
    % a) Update the parameters of the winner cluster (centers and variances)
    % for both input and output spaces and the counter
        c_k(winner_i)   = c_k(winner_i)+1;
        temp_mi_winner  = m_i(winner_i,:)+
         ((x1-m_i(winner_i,:))/c_k(winner_i));
        v_i(winner_i,:) = ((((c_k(winner_i)-
1)*(v_i(winner_i,:)+(m_i(winner_i,:).^2)))+(x1.^2))/c_k(winner_i))-
(temp_mi_winner.^2);
        m_i(winner_i,:) = temp_mi_winner;

        temp_mo_winner  = m_o(winner_o,:)+((y1-m_o(winner_o,:))
                          /c_k(winner_o));
        v_o(winner_o,:) = ((((c_k(winner_o)-1)*
(v_o(winner_o,:)+(m_o(winner_o,:).^2)))+(y1.^2))/c_k(winner_o))-
(temp_mo_winner.^2);
        m_o(winner_o,:) = temp_mo_winner;

        cluster_i{winner_i,1} = [cluster_i{winner_i,1}; x1];
        cluster_o{winner_o,1} = [cluster_o{winner_o,1}; y1];

    else
    % Step M7: New potential cluster creation
        alpha = closest_cluster_i(1);
        beta  = closest_cluster_i(2);
        out_c_from_alpha = closest_cluster_o(1);
        out_c_from_beta  = closest_cluster_o(2);
        is_search = zeros(K,K);
        is_found  = 0;

        for search_count = [1:((K^2-K)/2)]
            search_count;
            if (alpha==out_c_from_alpha)&&(beta==out_c_from_beta)
                % a) Merge beta seed cluster to alpha seed cluster by
                % computing their
```

```
                % cumulative counter and weighted average centers and
                % variances.
                c_k(alpha);
                m_i(alpha,:);
                c_k(beta);
                m_i(beta,:);

                v_i(alpha,:);
                v_i(beta,:);


temp_mi_alpha=(c_k(alpha)*m_i(alpha,:)+c_k(beta)*m_i(beta,:))/(c_k(alpha)+c_
k(beta));

v_i(alpha,:)=((c_k(alpha)*(v_i(alpha,:)+(m_i(alpha,:).^2))+c_k(beta)*(v_i(be
ta,:)+(m_i(beta,:).^2)))/(c_k(alpha)+c_k(beta)))-(temp_mi_alpha.^2);
                m_i(alpha,:)=temp_mi_alpha;


temp_mo_alpha=(c_k(alpha)*m_o(alpha,:)+c_k(beta)*m_o(beta,:))/(c_k(alpha)+c_
k(beta));

v_o(alpha,:)=((c_k(alpha)*(v_o(alpha,:)+(m_o(alpha,:).^2))+c_k(beta)*(v_o(be
ta,:)+(m_o(beta,:).^2)))/(c_k(alpha)+c_k(beta)))-(temp_mo_alpha.^2);
                m_o(alpha,:)=temp_mo_alpha;

                c_k(alpha)=c_k(alpha)+c_k(beta);
                cluster_i{alpha,1} = [cluster_i{alpha,1};
cluster_i{beta,1}];
                cluster_o{alpha,1} = [cluster_o{alpha,1};
cluster_o{beta,1}];


                % b) Store the incoming data [x,y] to cluster beta by
                % setting
                cluster_i{beta,1} = [x1];
                cluster_o{beta,1} = [y1];
                %cluster_i(beta,2) = {x_cell(d1,:)}
                c_k(beta) = 1;
                m_i(beta,:) = x1;
                v_i(beta,:) = 0;
                m_o(beta,:) = y1;
                v_o(beta,:) = 0;

                is_found = 1;

                break;
                % b) Update the distance meterics for the input and output
spaces

            else
                % If there remains unsearched paired seed clusters, repeat
Step M7
                % by searching the next closest pair of seed cluster in the
input
                % space
                is_search(alpha,beta) = 1;
                D_closest_i = inf;
                D_closest_o = inf;
                D_i;
                for c1 = [1:K]
                    for c2 = [c1:K]
                        c1;
                        c2;
                        D_i(c1,c2);
                        D_closest_i;
                        is_search(c1,c2);
                        if (D_i(c1,c2)<D_closest_i)&&(is_search(c1,c2)==0)&&
(c1 ~= c2)
                            D_closest_i=D_i(c1,c2);
                            closest_cluster_i=[c1 c2];
                        end
```

```
                        if (D_o(c1,c2)<D_closest_o)&&(is_search(c1,c2)==0)&&
(c1 ~= c2)
                            D_closest_o=D_o(c1,c2);
                            closest_cluster_o=[c1 c2];
                        end
                    end
                end
                alpha = closest_cluster_i(1);
                beta  = closest_cluster_i(2);
                out_c_from_alpha = closest_cluster_o(1);
                out_c_from_beta  = closest_cluster_o(2);

            end %end for (alpha==out_c_from_alpha)&&(beta==out_c_from_beta)
        end %search_count=[1:((K^2-K)/2)]

        if (is_found==0)
        % Increase K (K=K+1)
            K = K+1;
            c_k(K) = 1;
            m_i(K,:) = x1;
            v_i(K,:) = 0;
            m_o(K,:) = y1;
            v_o(K,:) = 0;
            cluster_i{K,1} = [x1];
            cluster_o{K,1} = [y1];

        end

    end% end for winner_i == winner_o

    % b) Update the distance meterics for the input and output spaces
    D_closest_i=inf;
    D_closest_o=inf;

    for c1=[1:K]
        for c2=[c1:K]
            D_i(c1,c2)=norm(m_i(c1,:)-m_i(c2,:));
            D_o(c1,c2)=norm(m_o(c1,:)-m_o(c2,:));
            if ((D_i(c1,c2)<D_closest_i) && (c1 ~= c2))
                D_closest_i=D_i(c1,c2);
                closest_cluster_i=[c1 c2];
            end
            if ((D_o(c1,c2)<D_closest_o) && (c1 ~= c2))
                D_closest_o=D_o(c1,c2);
                closest_cluster_o=[c1 c2];
            end
        end
    end
end % end for d1=[Initial_K+1:N]

% Step M9: Final combination procedure
% a)negligible weight to the majorseed cluster
epsilon = 3;
c1      = 1;
K_temp  = K;
while (c1 < K)
    c_k;
    m_i;
    v_i;
    m_o;
    v_o;
    cluster_i;
    cluster_o;
    D_i;
    D_o;
    c1;
    new_c_k = [];
    new_D_o = [];
    new_D_i = [];
    new_m_i = [];
    new_v_i = [];
    new_m_o = [];
    new_v_o = [];
```

```
new_cluster_i = cell(1,1);
new_cluster_o = cell(1,1);
Ktemp_flag = 0;
clus_size = 0;
for c1=[1:K_temp] %i=c1
    c1;
    K;
    K_temp;
    discard_flag = 0;
    c_k(c1);
    if (c_k(c1)<epsilon)
        % Find the cluster that is closest to cluster i
        D_closest_i = inf;
        D_closest_o = inf;

        %check distance input
        for c2=[1:K_temp]
            K;
            K_temp;
            c1;
            c2;
            D_i;
            D_i(c1,c2);
            D_closest_i;
          if (D_i(c1,c2)<D_closest_i) && (c1 ~= c2) && (D_i(c1,c2) ~= 0)
                D_closest_i=D_i(c1,c2);
                closest_cluster_i=[c1 c2];
            end
        end

        for c2=[1:K_temp]
            c2;
            c1;
            D_i;
            D_i(c2,c1);
            D_closest_i;
        if (D_i(c2,c1)<D_closest_i) && (c1 ~= c2) && ((D_i(c2,c1) ~= 0))
                D_closest_i=D_i(c2,c1);
                closest_cluster_i=[c2 c1];
            end
        end

        %check distance output
        for c2=[1:K_temp]
          if (D_o(c1,c2)<D_closest_o) && (c1 ~= c2) && (D_o(c1,c2) ~= 0)
                D_closest_o=D_o(c1,c2);
                closest_cluster_o=[c1 c2];
            end
        end

        for c2=[1:K_temp]
        if (D_o(c2,c1)<D_closest_o) && (c1 ~= c2) && ((D_o(c2,c1) ~= 0))
                D_closest_o=D_o(c2,c1);
                closest_cluster_o=[c2 c1];
            end
        end

        alpha = closest_cluster_i(1);
        beta = closest_cluster_i(2);
        out_c_from_alpha = closest_cluster_o(1);
        out_c_from_beta = closest_cluster_o(2);
        % if the mapping consistency holds, merge cluster c1 and c2

        if (alpha==out_c_from_alpha)&&(beta==out_c_from_beta)
            % the mapping consistency holds
            % a) Merge beta seed cluster to alpha seed cluster by
            % computing their
            % cumulative counter and weighted average centers and
            % variances.
            c_k;
            m_i;
            v_i;
            m_o;
```

```
                v_o;
                cluster_i;
                cluster_o;


temp_mi_alpha=(c_k(alpha)*m_i(alpha,:)+c_k(beta)*m_i(beta,:))/(c_k(alpha)+c_
k(beta));

v_i(alpha,:)=((c_k(alpha)*(v_i(alpha,:)+(m_i(alpha,:).^2))+c_k(beta)*(v_i(be
ta,:)+(m_i(beta,:).^2)))/(c_k(alpha)+c_k(beta)))-(temp_mi_alpha.^2);
                m_i(alpha,:)=temp_mi_alpha;


temp_mo_alpha=(c_k(alpha)*m_o(alpha,:)+c_k(beta)*m_o(beta,:))/(c_k(alpha)+c_
k(beta));

v_o(alpha,:)=((c_k(alpha)*(v_o(alpha,:)+(m_o(alpha,:).^2))+c_k(beta)*(v_o(be
ta,:)+(m_o(beta,:).^2)))/(c_k(alpha)+c_k(beta)))-(temp_mo_alpha.^2);
                m_o(alpha,:)=temp_mo_alpha;

                c_k(alpha) = c_k(alpha) + c_k(beta);
                cluster_i{alpha,1} = [cluster_i{alpha,1};
cluster_i{beta,1}];
                cluster_o{alpha,1} = [cluster_o{alpha,1};
cluster_o{beta,1}];

                % reduce cluster when j = beta
                new_c_k = [];
                new_m_i = [];
                new_v_i = [];
                new_m_o = [];
                new_v_o = [];
                new_cluster_i = cell(1,1);
                new_cluster_o = cell(1,1);
                clus_size =0;
                for j=[1:K_temp]% reduce cluster when j = beta
                    j;
                    if (j~=beta)
                        beta;
                        clus_size =clus_size+1;
                        new_c_k=[new_c_k; c_k(j)];
                        new_m_i=[new_m_i; m_i(j,:)];
                        new_v_i=[new_v_i; v_i(j,:)];
                        new_m_o=[new_m_o; m_o(j,:)];
                        new_v_o=[new_v_o; v_o(j,:)];
                        new_cluster_i{clus_size,1} = cluster_i{j,1};
                        new_cluster_o{clus_size,1} = cluster_o{j,1};
                    end
                end

                K = K-1;
                K_temp = K; %assiged to c_k,m_i,m_o
                c_k = new_c_k;
                m_i = new_m_i;
                v_i = new_v_i;
                m_o = new_m_o;
                v_o = new_v_o;
                cluster_i = new_cluster_i;
                cluster_o = new_cluster_o;

                % calculate the new distance for each node
                clear D_i;
                clear D_o;
                D_i=zeros(K,K);
                D_o=zeros(K,K);
                for j=[1:K]
                    j;
                    for c2=[j:K]
                        c2;
                        D_i(j,c2)=norm(m_i(j,:)-m_i(c2,:));
                        D_o(j,c2)=norm(m_o(j,:)-m_o(c2,:));
                    end
                end
```

```
                c1 = 1;
                break; %end while loop

            else %if (alpha==out_c_from_alpha)&&(beta==out_c_from_beta)
                % discard cluster i
                discard_flag = 1;
            end

            if (Ktemp_flag == 0)
                K_temp = K;
                Ktemp_flag = 1;
            end

            K = K-1;
        end %if (c_k(c1)<epsilon)

        if (discard_flag==0)
            clus_size = clus_size + 1;
            new_c_k = [new_c_k; c_k(c1)];
            new_m_i = [new_m_i; m_i(c1,:)];
            new_v_i = [new_v_i; v_i(c1,:)];
            new_m_o = [new_m_o; m_o(c1,:)];
            new_v_o = [new_v_o; v_o(c1,:)];
            new_cluster_i{clus_size,1} = cluster_i{c1,1};
            new_cluster_o{clus_size,1} = cluster_o{c1,1};
        end

    end % end of for c1=[1:K]
end %end of while

c_k = new_c_k;
m_i = new_m_i;
v_i = new_v_i;
m_o = new_m_o;
v_o = new_v_o;

cluster_i = new_cluster_i;
cluster_o = new_cluster_o;
clear D_i;
clear D_o;
D_i=zeros(K,K);
D_o=zeros(K,K);
for c1=[1:K]
    for c2=[c1:K]
        D_i(c1,c2) = norm(m_i(c1,:)-m_i(c2,:));
        D_o(c1,c2) = norm(m_o(c1,:)-m_o(c2,:));
    end
end
D_i;
D_o;

% b)
new_c_k = [];
new_D_o = [];
new_D_i = [];
new_m_i = [];
new_v_i = [];
new_m_o = [];
new_v_o = [];

new_cluster_i = cell(1,1);
new_cluster_o = cell(1,1);

merge_flag = 0;
[y_t,i_c_k] = sort(c_k);
i=0;
size(c_k,1);
while (i<K) && (size(c_k,1) > 2)
    i;
    i = i+1;
    x1 = m_i(i_c_k(i),:);
    p = zeros(K,1);
```

```
    sum_mu_product = 0;
    mu_product=zeros(K,1);
    if (merge_flag==0)
        for j=[1:K]
            j;
            mu = exp(-0.5*(((x1-m_i(i_c_k(j),:)).^2)./v_i(i_c_k(j),:)));
            if (j==i)
               mu_product(j)=0;
            else
               mu_product(j)=prod(mu);
            end
            sum_mu_product=sum_mu_product+mu_product(j);
        end

        a=[];
        for j=[1:K]
            p(j)=mu_product(j)/sum_mu_product;
            a=[a p(j)];
        end
        [tmp,l]=max(a);
        closest_test = [i_c_k(i) i_c_k(l)];
        % check if consistency holds for cluster l and cluster i, merge
cluster
        % In order to check the consistency, we expect that cluster l and
        % cluster i in the input space should have the closest distance.
        % Therefore, the cluster l and and i in the output space should have
        % the closest distance too.
        D_closest_o = inf;
        for c1=[1:K]
            for c2=[c1:K]
                c1;
                c2;
                if ((D_o(i_c_k(i),c2)<D_closest_o) && (c1 ~= c2) &&
(i_c_k(i) == c1))
                    D_closest_o = D_o(c1,c2);
                    closest_cluster_o = [c1 c2];
                end
            end
        end

        for c1=[1:K]
            for c2=[c1:K]
                c1;
                c2;
                if ((D_o(c1,i_c_k(i))<D_closest_o) && (c1 ~= c2) &&
(i_c_k(i) == c2))
                    D_closest_o=D_o(c1,c2);
                    closest_cluster_o=[c1 c2];
                end
            end
        end
        alpha = i_c_k(l);
        beta = i_c_k(i);
        % The consistency holds -> merge cluster
        if ((alpha==closest_cluster_o(1)) && (beta==closest_cluster_o(2)))
|| ((beta==closest_cluster_o(1)) && (alpha==closest_cluster_o(2)))
            if( ((mean(cluster_o{i_c_k(l),1}) > 9) &&
(mean(cluster_o{i_c_k(i),1}) >9)) &&  ((mean(cluster_o{i_c_k(l),1}) < 12) &&
(mean(cluster_o{i_c_k(i),1}) <12))) || ((mean(cluster_o{i_c_k(l),1}) < 9) &&
(mean(cluster_o{i_c_k(i),1}) <9)) ||((mean(cluster_o{i_c_k(l),1}) >15) &&
(mean(cluster_o{i_c_k(i),1}) >15))

temp_mi_alpha=(c_k(alpha)*m_i(alpha,:)+c_k(beta)*m_i(beta,:))/(c_k(alpha)+c_
k(beta));

v_i(alpha,:)=((c_k(alpha)*(v_i(alpha,:)+(m_i(alpha,:).^2))+c_k(beta)*(v_i(be
ta,:)+(m_i(beta,:).^2)))/(c_k(alpha)+c_k(beta)))-(temp_mi_alpha.^2);
                m_i(alpha,:)=temp_mi_alpha;


temp_mo_alpha=(c_k(alpha)*m_o(alpha,:)+c_k(beta)*m_o(beta,:))/(c_k(alpha)+c_
k(beta));
```

```
v_o(alpha,:)=((c_k(alpha)*(v_o(alpha,:)+(m_o(alpha,:).^2))+c_k(beta)*(v_o(be
ta,:)+(m_o(beta,:).^2)))/(c_k(alpha)+c_k(beta)))-(temp_mo_alpha.^2);
                m_o(alpha,:)=temp_mo_alpha;

                c_k(alpha)=c_k(alpha)+c_k(beta);

                cluster_i{alpha,1} = [cluster_i{alpha,1};
cluster_i{beta,1}];
                cluster_o{alpha,1} = [cluster_o{alpha,1};
cluster_o{beta,1}];
                clus_size = 0;

                for j=[1:K]
                    j;
                    if (j~=beta)
                        clus_size = clus_size + 1;
                        new_c_k=[new_c_k; c_k(j)];
                        new_m_i=[new_m_i; m_i(j,:)];
                        new_v_i=[new_v_i; v_i(j,:)];
                        new_m_o=[new_m_o; m_o(j,:)];
                        new_v_o=[new_v_o; v_o(j,:)];
                        new_cluster_i{clus_size,1} = cluster_i{j,1};
                        new_cluster_o{clus_size,1} = cluster_o{j,1};
                        %new_cluster_i{clus_size,2} = cluster_i{j,2}
                    end
                end

                K = K-1;
                i = 0;
                c_k = new_c_k;
                m_i = new_m_i;
                v_i = new_v_i;
                m_o = new_m_o;
                v_o = new_v_o;

                cluster_i = new_cluster_i;
                cluster_o = new_cluster_o;

                clear D_i;
                clear D_o;
                D_i=zeros(K,K);
                D_o=zeros(K,K);

                for c1=[1:K]
                    for c2=[c1:K]
                        c1;
                        c2;
                        D_i(c1,c2)=norm(m_i(c1,:)-m_i(c2,:));
                        D_o(c1,c2)=norm(m_o(c1,:)-m_o(c2,:));
                    end
                end

                new_c_k = [];
                new_D_o = [];
                new_D_i = [];
                new_m_i = [];
                new_v_i = [];
                new_m_o = [];
                new_v_o = [];
                new_cluster_i = cell(1,1);
                new_cluster_o = cell(1,1);
                merge_flag = 0;
                [y_t,i_c_k]=sort(c_k);
            end %(((mean(cluster_o{i_c_k(l),1}) > 9) &&
(mean(cluster_o{i_c_k(i),1}) >9)) &&  ((mean(cluster_o{i_c_k(l),1}) < 12) &&
(mean(cluster_o{i_c_k(i),1}) <12))) || ((mean(cluster_o{i_c_k(l),1}) < 9) &&
(mean(cluster_o{i_c_k(i),1}) <9)) ||((mean(cluster_o{i_c_k(l),1}) >15) &&
(mean(cluster_o{i_c_k(i),1}) >15))
        end %((alpha==closest_cluster_o(1)) && (beta==closest_cluster_o(2)))
|| ((beta==closest_cluster_o(1)) && (alpha==closest_cluster_o(2)))
    end% (merge_flag==0)
end %while (i<K) && (size(c_k,1) > 2)
```

```
cluster_i
```

ภาคผนวก ข

โปรแกรมคำนวณอัลกอริทึม
fast recursive linear/nonlinear least-square optimization

train_data.m

```
%close all;
load s1_cluster_3_kmax_4.mat;

x = i_set;
y = o_set;

%load load_parameter;
% Input format:
% X=[x1 x2 x3 ... xcol_len], where xi (n*1) is an input vector
% Y=[y1 y2 y3 ... ycol_len], where yk (m*1) is an input vector
% A user must initialize weight first
% For SANFIS (output is Sugeno-type
%------Initialize variables-----%
%X = x([1:2:29000],:); %input data
%Y = y([1:2:29000],:); %output data
X = x;
Y = y;
[row_len,X_len]=size(X);
[row_len,Y_len]=size(Y);
J = size(m_i,1); % J-Number of Cluster
n = size(m_i,2); % n-Number of Input
m = size(y,2);   % m-Number of Output

%--------Nonlinear weight-------%
W_N = [];
for a=1:J
    temp = [];
    for b=1:n
       temp = [temp m_i(a,b) v_i(a,b)];
    end
    W_N = [W_N; temp];
end
W = W_N ;% W=[wp1; wp2;... wpJ]; J(number of rules)*(i(number of inputs)*2)

%--------Linear weight----------%
W_L = m_o;
theta = W_L' ;% theta=[theta1; theta2; ... thetaJ];
% in program m(number of outputs)*J(number of rules)

%--------Initial lambda-----------------------%
lambda = cell(J,1);
delta = ones(J,1).*0.09;
%delta = 0.0001;
for j=1:J
    lambda{j} = eye(2);
    lambda{j}(2,2) =  n*delta(j);
end
v_i = v_i.^0.5;

% config parameter ONLY!!!
zeta = 0.25 %0.000025
max  = 40

% parameter
k         = 1.0005;
s_lamda   = 0.995;  %0.999;
s_lamda_r = 0.595;  %0.995;

%---matrix weight for phi(t-1)--%
phi = cell(J,2);

%---------Initial P(0)----------%
for j=1:J
   phi{j,1} = eye(n)*1;
   phi{j,2} = eye(n)*1;
end
phi_L = eye(J)*1;
one_time = 0;
sse = [];
```

```
ind_arr = [];


for epoch = 1:max
    epoch
    e_k = [];
    e_k2 = [];
    e_FBF = cell(J,1);
    for i_count = 1 : int32(0.2*row_len)
        i_count;
        sta(i_count) = 0;
    end

    for in_count=1:int32(0.2*row_len)

        % pick an input xi, and y
        %x1 = X(in_count,:);
        %y1 = Y(in_count,:);
        in_count;
        ind = randint(1,1,[1,int32(0.2*row_len)-1]);

        if sta(ind)==0
            sta(ind)= 1;
            ind_arr = [ind_arr; ind];
        else
             ind = randint(1,1,[1,int32(0.2*row_len)-1]);
             while (sta(ind) == 1)
                ind = randint(1,1,[1,int32(0.2*row_len)-1]);
             end
             sta(ind) = 1;
             ind_arr = [ind_arr; ind];

        end
        %ind_arr(in_count);
        ind  ;
        x1 = X(ind_arr(in_count),:);
        y1 = Y(ind_arr(in_count),:);
        % Compute the output of FBFs

        % Compute output of layer 1
        a1=x1  ;

        % Compute output of layer 2
        u=a1;
        M=n;
        % M is the size of inputs
        % J is the number of rules
        W=zeros(J,M,2);
        a2=zeros(J,M);
        % Weight in W(:,:,1) is the mean in the Gaussian membership function
        % Weight in W(:,:,2) is the variance in the Gaussian membership
function
        for c1=[1:J]
            mu=exp(-0.5*(((a1-m_i(c1,:))./v_i(c1,:)).^2));
            %tmp=-0.5*((u(c1)-W(c1,c2,1))/(W(c1,c2,2)))^2;
            a2(c1,:)=mu;
        end

        % Compute output of layer 3
        u=a2;
        a3=zeros(J,1);
        for c2=[1:J]
            a3(c2)=prod(u(c2,:));
        end

        % Compute output of layer 4
        u=a3;
        a4=zeros(J,1);
        sum_rules=sum(u);
        for c2=[1:J]
            a4(c2)=u(c2)/sum_rules;
        end
        %  [C,I] = max(a4); %Maximized
```

```
        % Compute output of layer 5

        u=a4;

        for c1=[1:m]
            tmp=0;
            for c2=[1:J]
                tmp= tmp + (u(c2)*theta(c1,c2));
            end
            a5(c1)=tmp;
            e = y1(c1)-tmp;
            e_k = [e_k ;e];
        end

        % optimization of linear weights
        % omega_t=J_star*phi*J_star'+alpha*tri;
        % phi=(1/alpha)*(phi-phi*J_star*inv(omega_t)*J_star*phi);
        % W=W-phi*J'*e;

        % optimization of nonlinear weights
        P=a4;
        col = 2*M;
        Jm = zeros(J,col);
        Jm_center = zeros(J,M);
        Jm_var = zeros(J,M);
        % find W_N_FBF
        col_c = 1;
        for c2=[1:n]

            for c1=[1:J]
                (P(c1)-P(c1).^2);
                Jm(c1,col_c)=(P(c1)-P(c1).^2)*(x1(c2)-
m_i(c1,c2))/(v_i(c1,c2).^2);
                Jm_center(c1,c2) = Jm(c1,col_c);
            end

            for c1=[1:J]
                Jm(c1,col_c+1)=(P(c1)-P(c1).^2)*((x1(c2)-
m_i(c1,c2)).^2)/(v_i(c1,c2).^3);
                Jm_var(c1,c2) = Jm(c1,col_c+1);
            end
            col_c = col_c + 2;

        end

        %Recursive L-M algorithm
        d_theta = [];
        for j= 1:J
            d_theta = [d_theta ; abs(y1 - theta(j))];
        end
        [C,I] = min(d_theta); %Maximized
        error = zeros(J,1);
        error = a4;
        error(I) = error(I) - 1;
        for j=1:J
            e_FBF{j} = [e_FBF{j} ; error(j)];
        end
        pos = mod(in_count,M)+1  ;
        for j=1:J
            last_col = zeros(M,1);
            last_col(pos,1) = 1;
            Jm_star = [Jm_center(j,:)' last_col];
            if (mod(in_count,n) ==0)
                if (in_count > (2*n+1)) %&& (onetime ~= 1)
                    omega = (Jm_star'*phi{j,1}*Jm_star) + s_lamda*lambda{j};
                    phi_t_t = (phi{j,1} -
phi{j,1}*Jm_star*inv(omega)*Jm_star'*phi{j,1})./s_lamda;
                    rp =
(Jm_center(j,:).*error(j))*phi_t_t*Jm_center(j,:)'.*error(j);
                    %ra = (0.5*n)*( sum(e_FBF(in_count-(2*n+1):in_count-
n).^2) - sum(e_FBF(in_count-n+1:in_count).^2))
                    first = sum(e_FBF{j}(in_count-(2*n+1):in_count-n).^2);
```

```
                    second = sum(e_FBF{j}(in_count-n+1:in_count).^2);
                    ra = (0.5/n)*( first - second );

                    if (ra > (zeta*rp))
                        delta(j) =  delta(j).*k;
                        lambda{j}(2,2) = n*delta(j);

                    else
                        if (ra < ((1-zeta)*rp))
                            delta(j) = delta(j)./k;
                            lambda{j}(2,2) = n*delta(j)  ;
                        end
                    end
                    % if j == J
                    %    onetime = 1;
                    %end
                end
            end

            omega = (Jm_star'*phi{j,1}*Jm_star) + s_lamda*lambda{j};
            phi_t = (phi{j,1} -
phi{j,1}*Jm_star*inv(omega)*Jm_star'*phi{j,1})./s_lamda   ;
            aa = phi_t*Jm_center(j,:)'.*error(j);
            m_i(j,:) = m_i(j,:) - (phi_t*Jm_center(j,:)'.*error(j))';
            phi{j,1} = phi_t;
        end

        for j=1:J
            last_col = zeros(M,1);
            last_col(pos,1) = 1;
            Jm_star = [Jm_var(j,:)' last_col];
            omega = (Jm_star'*phi{j,2}*Jm_star) + s_lamda*lambda{j};
            phi_t = (phi{j,2} -
phi{j,2}*Jm_star*inv(omega)*Jm_star'*phi{j,2})./s_lamda;
            bb=(phi_t*Jm_var(j,:)'.*error(j));
            v_i(j,:) = v_i(j,:) - (phi_t*Jm_var(j,:)'.*error(j))';
            phi{j,2} = phi_t;
        end

        %***************** Compute P_k(t) with W*_FBF
*************************%
        a1=x1;
        % Compute output of layer 1

        % Compute output of layer 2
        u=a1;
        M=n;
        % M is the size of inputs
        % J is the number of rules
        W=zeros(J,M,2);
        a2=zeros(J,M);
        % Weight in W(:,:,1) is the mean in the Gaussian membership function
        % Weight in W(:,:,2) is the variance in the Gaussian membership
function
        for c1=[1:J]
           % mu=exp(-0.5*(((a1-m_i(c1,:)).^2)./(v_i(c1,:).^2)));
           mu=exp(-0.5*(((a1-m_i(c1,:))./v_i(c1,:)).^2));
            %tmp=-0.5*((u(c1)-W(c1,c2,1))/(W(c1,c2,2)))^2;
            a2(c1,:)=mu;
        end

        % Compute output of layer 3
        u=a2;
        a3=zeros(J,1);
        for c2=[1:J]
            a3(c2)=prod(u(c2,:));
        end

        % Compute output of layer 4
        u=a3;
        a4=zeros(J,1);
        sum_rules=sum(u);
        for c2=[1:J]
```

```
            a4(c2)=u(c2)/sum_rules;
        end

        u=a4;
        for c1=[1:m]
            tmp=0;
            for c2=[1:J]
                tmp=tmp+(u(c2)*theta(c1,c2));
            end
            a5(c1)=tmp;
            e = y1(c1)-tmp;
            e_k2 = [e_k2 ;e];
        end

        phi_L_t = (phi_L -
((phi_L*a4*a4'*phi_L)./(s_lamda+(a4'*phi_L*a4)))./s_lamda;
        theta = theta + (phi_L_t*a4.*e)';
        phi_L = phi_L_t ;
    end
    %clc;
    sse = [sse ; mean(e_k2)]
    s_lamda
    epoch;
    min(sse);
    %plot(sse);
    s_lamda = (s_lamda_r*s_lamda)+(1-s_lamda_r);
        if (size(sse,1) > 2)
            if (sse(size(sse,1)) > sse(size(sse,1)-1)) && (one_time == 0)
                delta = ones(J,1).*0.000001;
                k=1.001;
                one_time = 1;
            end
        end
end
```

ประวัติการศึกษา และการทำงาน

ชื่อ –นามสกุล	จิรัฏฐ์ ศรีสบาย
วัน เดือน ปี ที่เกิด	8 พฤษภาคม 2526
สถานที่เกิด	มหาชัย จังหวัดสมุทรสาคร
ประวัติการศึกษา	วิศวกรรมศาสตรบัณฑิต (วิศวกรรมไฟฟ้า) มหาวิทยาลัยเกษตรศาสตร์
ตำแหน่งหน้าที่การงานปัจจุบัน	
สถานที่ทำงานปัจจุบัน	
ผลงานดีเด่นและรางวัลทางวิชาการ	- อันดับที่ 3 Robocup Thailand Championship 2005 - อันดับที่ 3 Robocup Thailand Championship 2007 - อันดับที่ 2 Robocup Thailand Championship 2008 - อันดับที่ 3 World Robocup 2008 เมืองซูโจว ประเทศจีน
ทุนการศึกษาที่ได้รับ	