อากาศยานไร้คนขับสี่ใบพัดนำทางอัตโนมัติในสภาพแวดล้อมที่ไม่รู้จักด้วยการเรียนรู้แบบเสริมกำลังเชิงลึก

นายภูวนาถ เผือกทอง

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมเมคคาทรอนิกส์
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2566

AN AUTONOMOUS NAVIGATION QUADROTOR IN UNKNOWN ENVIRONMENT USING DEEP REINFORCEMENT LEARNING

PHUWANAT PHUEAKTHONG

A Thesis Submitted in Partial Fulfillment of the Requirements for the
Degree of Master of Engineering in Mechatronics Engineering
Suranaree University of Technology
Academic Year 2023

อากาศยานไร้คนขับสี่ใบพัดนำทางอัตโนมัติในสภาพแวดล้อมที่ไม่รู้จัก
ด้วยการเรียนรู้แบบเสริมกำลังเชิงลึก

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

คณะกรรมการสอบวิทยานิพนธ์

.....

(ผศ. ดร.ชัยยุทธ สัมภวะคุปต์)
ประธานกรรมการ

.....

(อ. ดร.จิตติมา วระกุล)
กรรมการ (อาจารย์ที่ปรึกษาวิทยานิพนธ์)

.....

(ผศ. ดร.โศรฎา แข็งการ)
กรรมการ

.....

(รศ. ดร.ยุพาพร รักสกุลพิวัฒน์)
รองอธิการบดีฝ่ายวิชาการและประกันคุณภาพ

.....

(รศ. ดร.พรศิริ จงกล)
คณบดีสำนักวิชาวิศวกรรมศาสตร์

ภูวนาถ เผือกทอง : อากาศยานไร้คนขับสี่ใบพัดนำทางอัตโนมัติในสภาพแวดล้อมที่ไม่รู้จักด้วยการเรียนรู้แบบเสริมกำลังเชิงลึก (AN AUTONOMOUS NAVIGATION QUADROTOR IN UNKNOWN ENVIRONMENT USING DEEP REINFORCEMENT LEARNING) อาจารย์ที่ปรึกษา : อาจารย์ ดร.จิตติมา วระกุล, 128 หน้า.

คำสำคัญ: อากาศยานหลายใบพัดอัตโนมัติ/การเรียนรู้เสริมกำลังเชิงลึก/โดรนในอาคาร

งานวิจัยนี้ต้องการพัฒนาอากาศยานไร้คนขับขนาดเล็กที่สามารถนำทางตนเองได้โดยใช้การเรียนรู้แบบเสริมกำลังเชิงลึกแบบไม่ต้องใช้แผนที่ เนื่องจากในบางภารกิจอากาศยานไร้คนขับต้องทำงานในสภาพแวดล้อมที่ไม่รู้จักมาก่อนหรือไม่มีแผนที่ ซึ่งการเรียนรู้เสริมกำลังเชิงลึกเป็นปัญญาประดิษฐ์ที่เหมาะกับการนำมาใช้งานร่วมด้วยกับภารกิจดังกล่าว โดยอากาศยานไร้คนขับที่ถูกใช้สำหรับการวิจัยเป็นอากาศยานไร้คนขับขนาดเล็ก 4 ใบพัดมีน้ำหนักรวมสูงสุด 1,883 กรัม บินทดสอบในพื้นที่ในอาคารที่มีขนาดสูงสุดเป็นกว้าง 7.96 ยาว 12.07 และสูง 2.5 เมตร อากาศยานไร้คนขับจะถูกฝึกโมเดลการเรียนรู้ในโปรแกรมจำลองก่อนโดยใช้ระบบปฏิบัติการหุ่นยนต์เป็นกรอบการพัฒนาโปรแกรมทั้งหมด อัลกอริทึมที่เลือกใช้คือ Deep Deterministic Policy Gradient (DDPG) เซนเซอร์หลักที่ใช้ในการรับรู้สภาวะของตัวแทน (agent) เป็น LiDAR (Light Detection and Ranging) แบบ 2 มิติ การฝึกโมเดลใช้เวลาทั้งหมด 172 ชั่วโมง ใช้รอบการคำนวณ 5,324 รอบ ใช้จำนวนขั้นการฝึกฝนทั้งหมด 1,794,206 ขั้นเมื่อนำไปทดสอบในสภาพแวดล้อมจำลองที่ไม่เคยเห็นมาก่อน 4 แบบ ตัวแทนสามารถตัดสินใจหาเส้นทางไปจุดหมายได้ และเมื่อนำไปทดสอบกับอากาศยานในสภาพแวดล้อมจริง อากาศยานสามารถระบุตำแหน่งในอาคารได้ด้วยการรวมข้อมูลของไลดาร์ และ IMU ด้วยอัลกอริทึม RF2O และตัวกรองคาลมานแบบขยาย โมเดลสามารถทำงานได้สภาพแวดล้อมจริงที่แตกต่างกัน 2 แบบ โดยเส้นทางที่อากาศยานสร้างขึ้นสามารถพาอากาศยานหลบหลักสิ่งกีดขวางได้ไปจนถึงเป้าหมาย โดยที่เมื่อเทียบกับเส้นทางที่สร้างโดยตัววางแผน A* (A star) แล้วเส้นทางมีความใกล้เคียงกัน ระยะห่างระหว่างขอบใบพัดถึงสิ่งกีดขวางต่ำสุดในสภาพแวดล้อมจริงแบบที่ 1 และ 2 มีค่าเท่ากับ 0.13 และ 0.19 เมตรตามลำดับ

สาขาวิชา วิศวกรรมเมคคาทรอนิกส์ ลายมือชื่อนักศึกษา .....

ปีการศึกษา 2566 ลายมือชื่ออาจารย์ที่ปรึกษา.....

PHUWANAT PHUEAKTHONG : AN AUTONOMOUS NAVIGATION QUADROTOR IN UNKNOWN ENVIRONMENT USING DEEP REINFORCEMENT LEARNING.
THESIS ADVISOR : JITTIMA VARAGUL, Ph.D., 128 PP.

Keywords: Autonomous Copter/Deep Reinforcement Learning/Indoor Drone

This research aims to develop a small aerial vehicle capable of autonomous navigation using deep reinforcement learning in an unknown environment. Since in some missions, the unmanned aircraft must operate in unfamiliar or unmapped environments, Deep reinforcement learning is suitable for tasks. The UAVs used for the research is a quadcopter which has a maximum weight of 1,883 grams. It operated within an indoor area with maximum dimensions of 7.96 meters wide, 12.07 meters long, and 2.5 meters high. The UAVs is trained using deep reinforcement learning models in a simulation program by using a Robot Operating System as framework. The selected algorithm is Deep Deterministic Policy Gradient (DDPG). The primary sensor used for agent perception is a 2D LiDAR. The training process took a total of 172 hours, with 5,324 episodes and 1,794,206 steps. When tested in previously unseen simulated environments, the agent was able to make decisions to find a path to the destination. In real-world environments, the aircraft could determine its position within the building by LiDAR and IMU sensor fusion using the RF2O algorithm and an Extended Kalman Filter (EKF). The model demonstrated effectiveness in two different real-world environments, successfully navigating and avoiding obstacles to reach the target. The generated paths were closely to the path generated by A* Path Planner. The minimum clearance between the rotor edges and obstacles in the actual environments for type 1 and type 2 scenarios were 0.13 meters and 0.19 meters.

School of Mechatronics Engineering
Academic Year 2023

Student's Signature .....
Advisor's Signature .....

กิตติกรรมประกาศ

ผู้วิจัยขอกราบขอบพระคุณผู้มีอุปการะคุณทุกท่านที่ได้กรุณาให้ความช่วยเหลือ คำปรึกษา คำแนะนำ กำลังใจ รวมทั้งการสนับสนุนต่าง ๆ ทั้งในด้านวิชาการ การดำเนินงานจนกระทั่งวิทยานิพนธ์นี้สามารถบรรลุตามวัตถุประสงค์ โดยขอแสดงความขอบคุณบุคคลดังต่อไปนี้

อาจารย์ ดร.จิตติมา วระกุล ผู้เป็นอาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ได้ให้ความรู้ คำแนะนำ คำปรึกษา งบประมาณสนับสนุนในด้านการทำวิจัยและประสบการณ์ในด้านวิชาการและสังคม รวมไปถึงช่วยตรวจสอบบทความวิจัยต่าง ๆ และวิทยานิพนธ์จนสำเร็จลุล่วงสมบูรณ์

อาจารย์ ดร.ณัฐวัฒน์ พิณรัตน์ อาจารย์สาขาวิชาวิศวกรรมอุตสาหการ ที่ได้มอบความสนับสนุนด้านวิชาการ โภชนาการและการทำวิจัย ตลอดจนอำนวยสถานที่และอุปกรณ์ในการทำวิจัยครั้งนี้

นางสาวธัญญ์นภัส จารุโชคภูริวัจน์ เพื่อนบัณฑิตศึกษาที่ช่วยเหลือในการเก็บผลการทดลอง ช่วยให้กำลังใจและคำแนะนำในการปรับปรุงวิทยานิพนธ์จนไปถึงอำนวยความสะดวกในการจัดทำวิทยานิพนธ์

นายซอยี ไวยัน รุ่นน้องบัณฑิตศึกษาที่ช่วยเหลือในการดำเนินงานและเป็นธุระจัดการภารกิจของสัตว์เลี้ยงของผู้วิจัยในกรณีที่มีการดำเนินงานภาคสนามในส่วนของงานวิจัย

นางสาวเณศรา ธรรมนิตย์ ที่ให้การช่วยเหลือในการแก้ไขรูปเล่มวิทยานิพนธ์และอำนวยความสะดวกในการจัดทำรูปเล่มวิทยานิพนธ์และมอบบทเพลงผ่อนคลายให้สดับรับฟัง

เพื่อน พี่ น้อง ในห้องปฏิบัติการวอส์ป ไดนามิคส์ ที่คอยเป็นกำลังใจ เป็นแหล่งความคิดสนับสนุนพื้นที่ สนับสนุนการช่วยเหลือ และน้ำใจสำหรับการทดสอบงานวิจัยนี้จนสำเร็จลุล่วง

บิดา มารดา คุณยาย คุณตา ที่คอยอบรมเลี้ยงดู ให้ความช่วยเหลือและสนับสนุนในเรื่องต่าง ๆ เพื่ออำนวยความสะดวกอย่างถึงที่สุดในการศึกษาและการทำวิจัยนี้ให้เสร็จสมบูรณ์อย่างราบรื่น ไร้กังวล

สุดท้ายนี้ ขอกราบขอบพระคุณอาจารย์และผู้คนทุกท่านที่ไม่ได้กล่าวถึง ณ ที่นี้ ที่มีส่วนช่วยเติมเต็มส่วนประกอบของวิทยานิพนธ์ฉบับนี้ให้สมบูรณ์ตั้งแต่อดีตจนถึงปัจจุบันทั้งทางตรงและทางอ้อม ขอให้ทุกท่านจงประสบพบเจอแต่สิ่งที่มุ่งหวังในชีวิตและความสำเร็จในทุก ๆ ที่ไป

ภูวนาถ เผือกทอง

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

อากาศยานไร้คนขับ (Unmanned Aerial Vehicle : UAVs) หรือโดรน (Drone) มีการนิยมนำมาประยุกต์ใช้ได้หลากหลายในหลายด้าน ไม่ว่าจะเป็นการเกษตร การทหาร การแพทย์ และด้านอื่น ๆ ซึ่งก่อให้เกิดประโยชน์และเพิ่มประสิทธิภาพอย่างมากในการทำงาน ในการใช้งานทั่วไป เมื่อต้องการสั่งให้โดรนเคลื่อนที่ไปทำภารกิจในสถานที่ต่าง ๆ โดรนจะต้องอาศัยเส้นทางที่กำหนดไว้ก่อน (planned path) แล้วบินไปตามเส้นทางนั้น แต่ การบินตามเส้นทางที่ถูกกำหนดไว้เพียงอย่างเดียวไม่สามารถทำให้โดรนสามารถหลบหลีกสิ่งกีดขวางที่ไม่ทราบมาก่อนได้ ทำให้ต้องมีการติดตั้งเซนเซอร์ขึ้นไปบนโดรนด้วยเพื่อตรวจจับสิ่งกีดขวางและทำการหลบหลีก ความซับซ้อนของการหลบหลีกขึ้นอยู่กับอัลกอริทึมที่ใช้ในการหลบหลีก หากโดรนอยู่ในสภาวะแวดล้อมที่มีความซับซ้อน เช่น แนวสายไฟฟ้า ในพื้นที่โรงเรือนเกษตรกรรม เขตงานก่อสร้าง และในอาคารที่ซึ่งมีความซับซ้อนสูงมาก การตรวจจับสิ่งกีดขวางและการหลบหลีกโดยใช้อัลกอริทึมธรรมดาอาจมีประสิทธิภาพที่ไม่เพียงพอที่จะทำให้โดรนสามารถหลบหลีกแล้วกลับมายังเส้นทางเดิมได้ ต้องมีการใช้วิธีการที่มีความอัจฉริยะมากขึ้น เช่น การระบุตำแหน่งและสร้างแผนที่ไปพร้อม ๆ กัน (Simultaneous Localization and Mapping : SLAM) การเรียนรู้เชิงลึก (Deep Learning) และ การเรียนรู้แบบเสริมกำลังเชิงลึก (Deep Reinforcement Learning) เป็นต้น การนำทางโดยใช้ SLAM นั้นจะต้องอาศัยการสร้างแผนที่เอาไว้ก่อน จึงจะนำมาใช้ในการนำทางได้ จึงมีข้อจำกัดสำหรับพื้นที่ขนาดใหญ่ (large-scale environment) ส่วนการเรียนรู้เชิงลึกนั้นจะต้องอาศัยการฝึกโมเดลด้วยชุดข้อมูล (dataset) จำนวนมากและเหมาะกับงานประเภทการตรวจจับและจำแนกมากกว่า ส่วนการเรียนรู้แบบเสริมกำลังเชิงลึกเป็นการเรียนรู้ด้วยตนเองของตัวแทนระบบ (Agent) โดยไม่ต้องการทำการสอน ทำให้ไม่ต้องมีการทำชุดข้อมูลที่ติดฉลาก (label) มาให้ และสามารถประยุกต์ใช้กับการนำทางได้ เหมาะกับงานขนาดใหญ่และไม่ต้องมีแผนที่มาก่อน (unknown environment) แต่จะใช้เวลาในการฝึกฝนโมเดลที่มากกว่าการใช้ SLAM

ด้วยเหตุนี้ งานวิจัยนี้จึงต้องการศึกษาการพัฒนาอากาศยานปีกหมุนสี่ใบพัดที่สามารถนำทางตนเองได้โดยการใช้การเรียนรู้แบบเสริมกำลังเชิงลึก (Deep reinforcement learning) เพื่อที่จะสามารถนำความรู้ไปประยุกต์ใช้กับการนำทางของโดรนในสภาพแวดล้อมอื่น ๆ ที่มีการนำโดรนไปใช้ประโยชน์ได้ เช่น การใช้โดรนสำหรับการตรวจการณ์และเก็บข้อมูลในมุมมองใต้ร่วมไม้ การใช้โดรนในพื้นที่โรงเรือนในร่มเพื่อเฝ้าวัดการเจริญเติบโตของพืชผลเกษตร หรือเฝ้าระวังโรคพืช การใช้โดรนในการสำรวจอาคาร สำรวจเส้นทางสายไฟฟ้า ตรวจตราความปลอดภัยในอาคาร ใช้ในการรักษาความปลอดภัยในชุมชน ใช้ในการทหารและในเขตก่อสร้าง เป็นต้น นอกจากนี้ยังสามารถนำความรู้เรื่องของการใช้งานการเรียนรู้แบบเสริมกำลังเชิงลึก ไปใช้แก้ปัญหากับหุ่นยนต์ในรูปแบบอื่นได้ด้วย เช่น การสอนให้หุ่นยนต์แบบมีขา (ped robot) ทรงตัวได้ การสอนให้แขนกลหยิบชิ้นงานที่ซับซ้อน การนำทางของหุ่นยนต์เคลื่อนที่ในกลุ่มคนในสังคมอย่างระมัดระวัง (social awareness) การนำทางของรถยนต์ไร้คนขับโดยใช้การมองเห็นจากกล้อง (vision self- navigation) เป็นต้น โดยวัตถุประสงค์ ขอบเขตและรายละเอียดของงานวิจัยนี้จะถูกกล่าวถึงต่อไปในส่วนถัดไป

1.2 วัตถุประสงค์

1.2.1 เพื่อพัฒนาอากาศยานไร้คนขับขนาดเล็กที่สามารถนำทางตนเองได้โดยใช้การเรียนรู้แบบเสริมกำลังเชิงลึกแบบไม่ต้องใช้แผนที่ในสภาพแวดล้อมที่ไม่รู้จักมาก่อน

1.3. ขอบเขตของการวิจัย

1.3.1 อากาศยานเป็นแบบอากาศยานปีกหมุนมีจำนวนใบพัด 4 ใบพัด

1.3.2 สภาพแวดล้อมเป็นสภาพแวดล้อมในอาคาร มีความกว้าง 5.75 เมตร ยาว 5.75 เมตร สูง 2.5 เมตร

1.3.3 การนำทางเป็นการนำทางในรูปแบบการเคลื่อนที่ 2 มิติในแนวแกน x และ y โดยรักษาระดับความสูงเทียบกับจุดอ้างอิงไว้

1.3.4 ใช้โมเดลการเรียนรู้แบบเสริมกำลังเชิงลึก (Deep reinforcement learning) ในการสอนโมเดลการนำทางในโปรแกรมจำลองเพื่อทำการเรียนรู้ก่อน จากนั้นจึงนำไปทดสอบในสภาพแวดล้อมจริง

1.3.5 เซนเซอร์สำหรับการตรวจจับสิ่งกีดขวางและสภาพแวดล้อมเป็นไลดาร์ (LiDAR) แบบ 2 มิติ

1.3.5 ทำการบินโดยไม่ใช้สัญญาณจากระบบจีพีเอส (GPS)

1.3.7 ขนาดเส้นผ่านศูนย์กลางใบพัดไม่เกิน 12 นิ้ว

1.3.8 น้ำหนักของอากาศยานไร้คนขับไม่เกิน 2.5 กิโลกรัม

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 ได้ศึกษาการใช้การเรียนรู้แบบเสริมกำลังเชิงลึกสำหรับสำหรับการประยุกต์ใช้ในการนำทาง

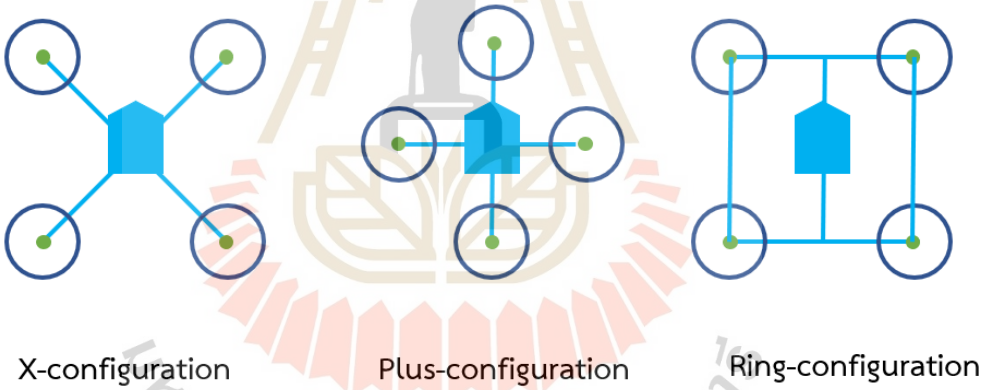
1.4.2 ได้แนวทางในการพัฒนาอากาศยานไร้คนขับอัตโนมัติสำหรับประยุกต์ในงานตรวจสอบและตรวจการณ์ในสถานที่กลางแจ้งและในร่ม

บทที่ 2
ปริทรรศน์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

2.1 การออกแบบอากาศยานสี่ใบพัด

2.1.1 ประเภทของโครงอากาศยานปีกหมุน

ประเภทโครงของอากาศยานปีกหมุน ถูกแบ่งตามจำนวนของใบพัดที่มีการติดตั้งลงไปและรูปแบบการติดตั้ง ส่วนที่ติดตั้งใบพัดที่ยื่นออกจากศูนย์กลางของลำตัวจะถูกเรียกว่า “แขน (Arm)” สำหรับอากาศยานสี่ใบพัดนั้น รูปแบบโครงสร้างหลักจะถูกแบ่งเป็น 3 ประเภท คือ รูปตัวเอ็กซ์ (X-configuration), รูปเครื่องหมายบวก (Plus-configuration) และรูปวงแหวน (Ring-configuration) ดังในรูปที่ 2.1

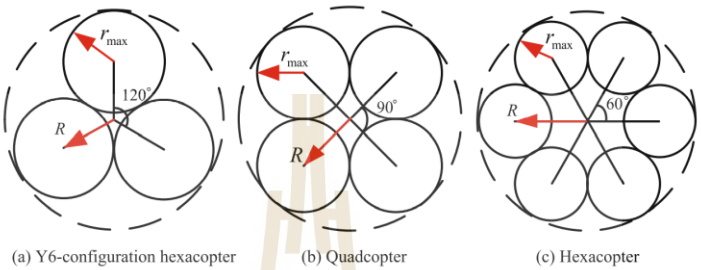


รูปที่ 2.1 ประเภทของโครงอากาศยานปีกหมุนสี่ใบพัด

โครงอากาศยานแบบตัวเอ็กซ์จะให้ความคล่องตัวมากกว่าแบบเครื่องหมายบวก และมีพื้นที่การมองเห็นด้านหน้าเมื่อติดกล้องที่กว้างกว่าแบบเครื่องหมายบวก ส่วนประเภทวงแหวนนั้นจะช่วยลดแรงสั่นสะเทือนจากมอเตอร์และใบพัดที่จะส่งผลต่อเซนเซอร์ของอุปกรณ์ควบคุมการบินได้ แต่จะมีความคล่องตัวที่น้อย

2.1.2 การกำหนดขนาดของโครงสร้างลำตัว

การกำหนดโครงสร้างลำตัวของอากาศยานปีกหมุนหลายใบพัด (Multicopper) นั้น มีความสัมพันธ์กับขนาดของใบพัดที่จะใช้ และจำนวนของใบพัด โดยมุมของการติดตั้งใบพัดมักจะถูกแบ่งให้เท่า ๆ กันดังในรูปที่ 2.2



รูปที่ 2.2 การแบ่งมุมองศาการติดตั้งใบพัด (Quan, Xunhua, และ Shuai, 2020)

เมื่อ r_{max} คือ ขอบเขตรัศมีที่มากที่สุดของใบพัด ส่วน R คือ รัศมีของแขนของโครงสร้างอากาศยานที่เหมาะสม ซึ่งหากใบพัดมีจำนวนทั้งหมด n_r ใบพัด จะได้ความสัมพันธ์ของขนาดใบพัด ขนาดแขนอากาศยาน และจำนวนใบพัดดังสมการต่อไปนี้

$$R = \frac{r_{max}}{\sin\left(\frac{180^{\circ}}{n_r}\right)} \tag{2.1}$$

จากความสัมพันธ์ในสมการข้างต้น จะทำให้สามารถคำนวณขนาดของอากาศยานที่เหมาะสมออกมาได้ ด้วยขนาดที่ได้มาจากสมการข้างต้น จะได้ระยะที่ไม่ชิดกันเกินไปจนกระแสอากาศ (vortex) จากใบพัดแต่ละใบจะไม่ส่งผลถึงกันจนทำให้แรงขับ (thrust) ลดลงไป ซึ่งทำให้ประสิทธิภาพทางด้านอากาศพลศาสตร์สูงขึ้น โดยการกำหนดขอบเขตรัศมีของใบพัดสูงสุด ควรอยู่ในช่วงต่อไปนี้ โดย r_p คือรัศมีของใบพัดที่เลือกใช้

$$r_{max} = 1.05r_p \sim 1.2r_p \tag{2.2}$$

2.1.3 การประมาณน้ำหนักของอากาศยานปีกหมุน

ในการออกแบบอากาศยานแต่ละประเภท หนึ่งในขั้นตอนแรกที่ต้องทำหลังจากการกำหนดความต้องการในการออกแบบคือการประมาณน้ำหนักของอากาศยานที่ทำการออกแบบ

เพราะเกี่ยวข้องกับการกำหนดขนาด ระบบขับดัน (propulsion system) ของอากาศยาน โดยสำหรับอากาศยานปีกหมุนนั้นน้ำหนักเบื้องต้นของอากาศยานสามารถหาได้จากการหาผลรวมของน้ำหนักโดยประมาณของส่วนประกอบหลัก คือ ใบพัด, มอเตอร์ โครงอากาศยาน, แบตเตอรี่, ESC และอุปกรณ์ที่จะติดตั้งบนอากาศยาน (payload) หรือเขียนเป็นความสัมพันธ์ คือ

$$W_{aircraft} = W_{prop} + W_{motor} + W_{batt} + W_{ESC} + W_{af} + W_{payload} + W_{other} \quad (2.3)$$

เมื่อ $W_{aircraft}$ คือ น้ำหนักรวมของอากาศยาน
W_{prop} คือ น้ำหนักของใบพัด
W_{motor} คือ น้ำหนักของมอเตอร์
W_{batt} คือ น้ำหนักของแบตเตอรี่
W_{ESC} คือ น้ำหนักของ ESC (Electronic Speed Controller)
W_{af} คือ น้ำหนักของโครงอากาศยาน
$W_{payload}$ คือ น้ำหนักของวัตถุที่จะบรรทุก
W_{other} คือ น้ำหนักอุปกรณ์อื่น ๆ

(Justin Winslow, 2017) นำเสนอสมการสำหรับการประมาณน้ำหนักของอุปกรณ์และส่วนประกอบต่าง ๆ ของอากาศยานสี่ใบพัดขนาดเล็ก (Micro Aerial Vehicle: MAV) โดยใช้การรวบรวมข้อมูลทางสถิติจากอากาศยานที่มีการสร้างขึ้น แล้วนำมาหาความสัมพันธ์ระหว่างน้ำหนักของส่วนประกอบแต่ละส่วนร่วมกับคุณสมบัติของชิ้นส่วนนั้น โดยสมเหล่านั้นจะถูกนำเสนอในส่วนถัดไปนี้

การประมาณน้ำหนักของใบพัดสามารถหาได้จากความสัมพันธ์ระหว่างน้ำหนักของใบพัดร่วมกับ รัศมีของใบพัด (r_p) ในหน่วยเซนติเมตร, จำนวนแฉกของใบพัด (number of blade : N_b) และ solidity ของใบพัด (θ) โดยใช้สมการต่อไปนี้

$$W_{propeller} = 0.0195R^{2.0589}\theta^{-0.2038}N_b^{0.5344} \quad (2.4)$$

การประมาณน้ำหนักของแบตเตอรี่ สามารถประมาณได้จากความสัมพันธ์ร่วมกับความจุของแบตเตอรี่ (C) ในหน่วย mAh และจำนวนเซลล์ที่ต่ออนุกรมกันของแบตเตอรี่ (N_{cell})

$$W_{battery} = 0.0418C^{0.9327}N_{cell}^{1.0725} \quad (2.5)$$

การประมาณน้ำหนักของมอเตอร์ไร้แปรงถ่าน (Brushless motor) สามารถทำได้โดยใช้ความสัมพันธ์ของน้ำหนักร่วมกับ ความเร็วคงที่ของมอเตอร์ (Kv) ในหน่วยรอบต่อนาทีต่อโวลต์, กระแสสูงสุดที่มอเตอร์ใช้ (I), ความยาวของมอเตอร์ (l_{BL}) ในหน่วยมิลลิเมตร, เส้นผ่านศูนย์กลางภายนอกของมอเตอร์ (d_{BL}) ในหน่วยมิลลิเมตรและกำลังสูงสุดของมอเตอร์ (maximum rated output power : P) โดยมีความสัมพันธ์ตามสมการต่อไปนี้

$$W_{motor} = 0.0109Kv^{0.5122}P^{-0.1902}(\log_{10} l_{BL})^{2.5582}(\log_{10} d_{BL})^{12.8502} \quad (2.6)$$

เมื่อความยาวของมอเตอร์หาได้จากสมการ

$$l_{BL} = 4.8910I^{0.1751}P^{0.2476} \quad (2.7)$$

และเส้นผ่านศูนย์กลางภายนอกของมอเตอร์หาได้จากสมการ

$$d_{BL} = 41.45Kv^{-0.1919}P^{0.1935} \quad (2.8)$$

ข้อมูลของมอเตอร์สามารถหาได้จากข้อมูลที่เตรียมโดยผู้ผลิต โดยการเลือกมอเตอร์จะสอดคล้องกับอัตราส่วนน้ำหนักของอากาศยานและอัตราส่วนแรงขับต่อน้ำหนักที่เลือก (Thrust to weight ratio : T/W) ใช้ โดย (Marcin Biczyski, 2020) เสนอตารางข้อมูลการเลือกใช้ T/W กับประเภทการใช้งานของอากาศยานปีกหมุนไว้ดังรูปที่ 2.3

Thrust-to-weight ratio	Application
2	Slow flight (minimum)
3	Payload transport; photography
4	Surveillance
5+	Aerobatics; high-speed video
7+	Racing

รูปที่ 2.3 การเลือกอัตราส่วนแรงขับต่อน้ำหนักกับการใช้งานของอากาศยาน (Biczyski, 2020)

เมื่อทราบอัตราส่วนแรงขับต่อน้ำหนักแล้วสามารถหาแรงขับที่ต้องการอย่างตอน hover ($T_{r,hover}$) และแรงขับที่ต้องการตอนบินด้วยสภาวะ Wide Open Throttle (WOT) หรือ $T_{r,WOT}$ ต่อมอเตอร์ 1 ตัว ได้ด้วยสมการต่อไปนี้

$$T_{r,hover} = \frac{W_{total}}{n_r} \tag{2.9}$$

$$T_{r,WOT} = \left(\frac{T}{W}\right) \times T_{r,hover} \tag{2.10}$$

การประมาณน้ำหนักของ ESC สามารถประมาณได้จากความสัมพันธ์ร่วมกับกระแสสูงสุดที่ไหลผ่าน ESC ($I_{ESC,max}$) โดยหาได้จากสมการดังนี้

$$W_{ESC} = 0.8013 I_{ESC,max}^{0.9727} \tag{2.11}$$

การประมาณน้ำหนักของโครงอากาศยาน สามารถประมาณได้จากความสัมพันธ์ร่วมกับรัศมีของใบพัด (r_p) ในหน่วยเซนติเมตรและมวลของแบตเตอรี่ ($W_{battery}$) ในหน่วยกรัม โดยหาได้จากสมการดังนี้

$$W_{airframe} = 1.3119 R^{1.2767} W_{battery}^{0.4587} \tag{2.12}$$

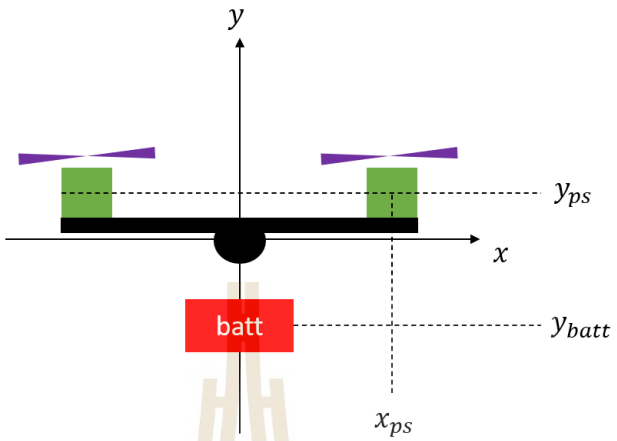
ส่วนน้ำหนักของวัตถุที่บรรทุกขึ้นไปสามารถหาได้จากข้อมูลน้ำหนักของอุปกรณ์ที่ได้จากผู้ผลิต หรือจากการชั่ง เมื่อได้น้ำหนักทั้งหมดแล้ว จะสามารถนำไปประมาณน้ำหนักรวมของอากาศยานได้เพื่อทำการออกแบบต่อไป

2.1.4 โมเมนต์ความเฉื่อยของอากาศยานปีกหมุน

โมเมนต์ความเฉื่อย (Moment of inertia) ของอากาศยานปีกหมุน (J_{copter}) จะบอกถึงความคล่องตัวในการหมุนตัว (rotation) รอบแกน pitch และ roll ของตัวอากาศยานเอง โดยโมเมนต์ความเฉื่อยของอากาศยานปีกหมุนสามารถหาได้จากความสัมพันธ์ของ จำนวนใบพัด (n_r) , มวลของระบบขับดัน (m_{ps}), ระยะทางระหว่างจุดศูนย์ถ่วง (Center of gravity) ของระบบขับดันถึงจุดศูนย์ถ่วงของโดรนในแนวดิ่ง (y_{ps}) และแนวระดับ (x_{ps}) และระยะทางระหว่างจุดศูนย์ถ่วงของแบตเตอรี่ถึงจุดศูนย์ถ่วงของโดรน (y_{batt}) โดยความสัมพันธ์แสดงได้ด้วยสมการดังนี้

$$J_{copter} = n_r m_{ps}\left(y_{ps}^2 + x_{ps}^2\right) + m_{batt} y_{batt}^2 \tag{2.13}$$

ซึ่งมีแผนภาพแสดงระยะทางต่าง ๆ ดังรูปต่อไปนี้



รูปที่ 2.4 แผนผังระยะทางสำหรับคำนวณโมเมนต์ความเฉื่อย

ยิ่งโมเมนต์ความเฉื่อยมีค่าน้อย จะส่งผลให้การหมุนตัวของอากาศยานทำได้ง่ายกว่าโมเมนต์ความเฉื่อยที่มีค่ามาก

2.1.5 การประมาณเวลาการบินของอากาศยาน

การประมาณเวลาการบินสำหรับอากาศยานปีกหมุนที่ใช้ไฟฟ้าเป็นพลังงานจะมีความสัมพันธ์เกี่ยวกับขนาดความจุของแบตเตอรี่และพลังงานที่โดรนมีการใช้ระหว่างบิน การประมาณเวลาการบินจะช่วยให้สามารถกำหนดขอบเขตการทำงานของอากาศยานได้ ในระบบขับเคลื่อนของอากาศยาน (propulsion system) การสูญเสียพลังงานของแบตเตอรี่สามารถหาได้จากผลรวมของการสูญเสียพลังงานในรูปแบบต่าง ๆ โดยแผนผังการส่งผ่านพลังงานของแบตเตอรี่แก่โดรนแสดงดังในรูปที่ 2.5

รูปที่ 2.5 กระบวนการส่งผ่านกำลังงานของระบบขับเคลื่อน

เมื่อพิจารณากระบวนการส่งผ่านกำลังของระบบขับเคลื่อนจะได้ว่า กำลังที่ถูกจ่ายเข้าไปจะนำไปถูกแจกจ่ายดังสมการต่อไปนี้

$$P_{in} = P_M + P_{motor} + P_I + P_c \tag{2.14}$$

เมื่อ P_M คือ กำลังทางกลของใบพัด ซึ่งสามารถหาได้จากความสำคัญของทอร์คการหมุน τ และความเร็วการหมุนของใบพัด ω ดังสมการต่อไปนี้

$$P_M = M \times \omega \tag{2.15}$$

ส่วน P_m คือ กำลังที่สูญเสียไปของ brushless DC motor ซึ่งหาได้จากความสัมพันธ์ของแรงดัน emf ป้อนกลับของมอเตอร์ $\mathrm{V_m}$ และกระแส back emf I_m ดังสมการต่อไปนี้

$$P_m = V_m \times I_m \tag{2.16}$$

จากนั้น P_I คือ กำลังที่สูญเสียไปกับวัสดุโลหะ หรือ iron loss ซึ่งหาได้จากความสัมพันธ์ของสัมประสิทธิ์ของ Steinmetz k , ความหนาแน่นสูงสุดของฟลักซ์ของสนามแม่เหล็ก (peak magnetic flux density) และความที่ f ดังสมการต่อไปนี้

$$P_I = k \times f \times B^2 \tag{2.17}$$

ต่อมา P_c คือ การสูญเสียกำลังในทองแดงซึ่งหาได้จากความสัมพันธ์ของความต้านทานวงจรของระบบขับดัน R_m และกระแสในวงจร ดังสมการต่อไปนี้

$$P_I = I^2 \times R_m \tag{2.18}$$

เมื่อคำนวณกำลังที่เกิดการสูญเสียได้แล้ว ต่อไปเมื่อพิจารณาการบริโภคพลังงาน (energy consumption) ของอากาศยานไร้คนขับ E_{prop} ซึ่งหาได้จากความสัมพันธ์ของกำลังที่เข้าไปและเวลา โดยค่ามีหน่วยเป็นจูล และเวลามีหน่วยเป็นวินาที ดังสมการต่อไปนี้

$$E_{prop} = P_{in} \times t \tag{2.19}$$

ส่วนพลังงานที่มีการเก็บไว้ในแบตเตอรี่สามารถหาได้จากสมการความสัมพันธ์ของแรงดันของแบตเตอรี่ V_{bat} และความจุของแบตเตอรี่ C_{batt} หน่วยเป็น Ah

$$E_{batt} = V_{batt} \times C_{batt} \times 3600 \tag{2.20}$$

และสามารถคำนวณอัตราการใช้พลังงานเป็นร้อยละได้ดังสมการต่อไปนี้

$$C = 100 - \left(\frac{E_{batt} - E_{prop}}{E_{batt}}\right) \times 100 \tag{2.21}$$

เมื่อทราบอัตราการบริโภคพลังงานแล้ว จะสามารถตรวจสอบได้ว่าพลังงานที่มีเพียงพอหรือไม่ และหากอยากทราบเวลาในการใช้งานของแบตเตอรี่สามารถหาได้โดยใช้หลักการสมดุลของพลังงานดังนี้

$$Flight\ time = \frac{V_{batt} \times C_{batt} \times 3600}{P_{in}} \tag{2.22}$$

โดยเวลาในการบิน (flight time) ที่ได้ออกมาจะมีหน่วยเป็นวินาที

2.2 อุปกรณ์ควบคุมและสื่อสารของอากาศยานไร้คนขับ

2.2.1 บอร์ดควบคุมการบิน

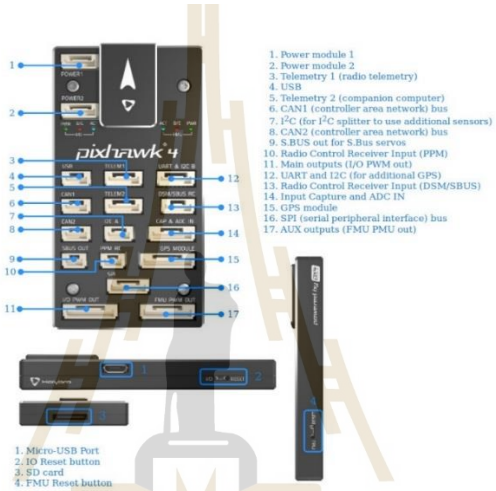
บอร์ดควบคุมการบิน (flight controller board) เป็นบอร์ดอิเล็กทรอนิกส์ที่บรรจุเซนเซอร์และอุปกรณ์สำหรับใส่ซอฟท์แวร์ที่ใช้ในการควบคุมอากาศยานไร้คนขับเอาไว้ ซึ่งสำหรับอากาศยานไร้คนขับขนาดเล็ก (Micro Aerial Vehicle : MAV) ในปัจจุบันมี อุปกรณ์สำหรับควบคุมการบินมีอยู่หลากหลาย เช่น บอร์ด Pixhawk, CUAV, Flight one และ Hobby wing เป็นต้น แต่บอร์ดควบคุมการบินที่มีความน่าสนใจและมักถูกใช้ในงานวิจัยคือ บอร์ด Pixhawk ซึ่งมีถึงเวอร์ชัน 4 ในปัจจุบันโดยบริษัท Holybro จุดเด่นของบอร์ด Pixhawk ที่สำคัญคือการที่โค้ดเป็นแบบเปิดเผยหรือ open-source จึงเป็นที่นิยมกันอย่างแพร่หลายในการพัฒนาอากาศยานไร้คนขับ คุณลักษณะของบอร์ด Pixhawk4 แสดงดังตารางต่อไปนี้

ตารางที่ 2.1 คุณลักษณะของบอร์ดควบคุมการบิน Pixhawk4

หัวข้อคุณลักษณะ	คุณลักษณะ
Main FMU Processor	STM32F765: 32 Bit Arm® Cortex®-M7, 216MHz, 2MB memory, 512KB RAM
IO Processor	STM32F100: 32 Bit Arm® Cortex®-M3, 24MHz, 8KB SRAM
On-board sensors	- Accel/Gyro: ICM-20689 - Accel/Gyro: BMI055 - Magnetometer: IST8310 - Barometer: MS5611
GPS	u-blox Neo-M8N GPS/GLONASS receiver; integrated magnetometer IST8310
Interfaces	- 8-16 PWM outputs (8 from IO, 8 from FMU) - 3 dedicated PWM/Capture inputs on FMU - Dedicated R/C input for CPPM - Dedicated R/C input for Spektrum / DSM and S.Bus with analog / PWM RSSI input - 3 I2C ports - 4 SPI buses - Up to 2 CANBuses for dual CAN with serial ESC Analog inputs for voltage / current of 2 batteries
Power System	- Power module output: 4.9~5.5V - USB Power Input: 4.75~5.25V - Servo Rail Input: 0~36V
Weight and Dimensions	- Weight: 15.8g - Dimensions: 44x84x12mm
Operating temperature	-40 ~ 85°c

รูปร่างของบอร์ด pixhawk4 และพอร์ตในการเชื่อมต่อของบอร์ด pixhawk4 ถูกแสดงดังรูปที่ 2.6 และ 2.7

รูปที่ 2.6 บอร์ด Pixhawk4 (PX4Docs, 2021)



รูปที่ 2.7 พอร์ตการเชื่อมต่ออุปกรณ์ของบอร์ด Pixhawk4 (PX4Docs, 2021)

ซอฟท์แวร์สำหรับบอร์ดควบคุมการบินที่มีการเปิดเผยโค้ดมีอยู่ 2 แห่งหลักคือโค้ดของ Ardupilot และ PX4 โดยจะมีความแตกต่างกันในเรื่องของความสะดวกในการใช้งานและอื่น ๆ และเฟิร์มแวร์ทั้งสองจะใช้โปรแกรมควบคุมภาคพื้นอตกต่างกัน โดย PX4 นั้นจะใช้โปรแกรม QGround Control เป็นซอฟท์แวร์สำหรับสถานีภาคพื้น ซึ่งมีลักษณะดังในรูปที่ 2.8

รูปที่ 2.8 ตัวอย่างหน้าจอของโปรแกรม Q Ground Control

2.2.2 Telemetry module

Telemetry module หรือ Telemetry radio เป็นอุปกรณ์ในการส่งข้อมูลระหว่างอากาศยานไร้คนขับและสถานควบคุมภาคพื้น (ground control station) เพื่อที่จะสื่อสารข้อมูลระหว่างอากาศยานร้คนขับและสถานีควบคุมภาคพื้นระหว่างบินได้ ตัวอย่างของ telemetry module ที่สามารถใช้ได้กับบอร์ด Pixhawk4 คือ RFD900, HKPilo และ Holybro Telemetry radio ดังในรูปที่ 2.9 เป็นต้น

รูปที่ 2.9 Holybro SIK telemetry radio (PX4, 2021)

2.2.3 รีโมทวิทยุบังคับและอุปกรณ์รับสัญญาณ

รีโมทวิทยุบังคับ (RC remote) และอุปกรณ์รับสัญญาณ (radio receiver) เป็นอุปกรณ์สำคัญที่ให้ผู้ใช้สามารถบินควบคุมอากาศยานไร้คนขับแบบ manual ได้ โดยมีการส่งสัญญาณผ่านคลื่นวิทยุ ซึ่งอุปกรณ์รับสัญญาณจะถูกติดตั้งอยู่กับอากาศยานไร้คนขับ และอุปกรณ์รับสัญญาณจะสามารถจ่ายสัญญาณ Pulse Width Modulation หรือ PWM ออกไปให้กับอุปกรณ์ต่าง ๆ บนอากาศยานไร้คนขับได้ และสัญญาณนั้นจะถูกนำไปเข้าอุปกรณ์ควบคุมของอากาศยานไร้คนขับต่อไป ตัวอย่างของอุปกรณ์รับสัญญาณและส่งสัญญาณวิทยุบังคับ แสดงดังในรูปที่ 2.10

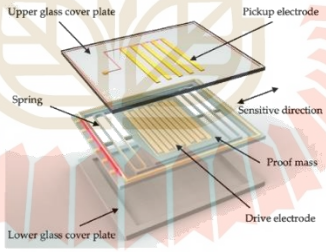
รูปที่ 2.10 อุปกรณ์ส่งและรับสัญญาณวิทยุบังคับ

2.3 อุปกรณ์รับรู้ข้อมูลของอากาศยานไร้คนขับ

2.3.1 Inertial Measurement Unit

Inertial Measurement Unit หรือ IMU เป็นเซนเซอร์ที่ใช้สำหรับตรวจจับการเคลื่อนไหว ไม่ว่าจะเป็นในหุ่นยนต์ ในเรือ หรือแม้แต่ในอากาศยานไร้คนขับ โดย IMU ที่ใช้ในอากาศยานไร้คนขับอย่างน้อยจะต้องสามารถตรวจจับการเคลื่อนไหวใน 6 องศาอิสระได้ เพื่อที่จะใช้ในการคำนวณท่าทางของอากาศยานไร้คนขับแล้วทำการควบคุม โดยใน IMU ที่มี 6 องศาอิสระ จะประกอบด้วยเซนเซอร์ 2 ชนิด คือ เซนเซอร์วัดความเร็วเชิงมุม (gyroscope) และเซนเซอร์วัดความเร่งเชิงเส้น (accelerometer) ตัวอย่างของ IMU ที่ใช้ในบอร์ดควบคุมการบิน Pixhawk 4 คือ BOSCH BMI50 และ Invensense ICM-20689 หลักการทำงานของอุปกรณ์แต่ละตัวเป็นดังต่อไปนี้

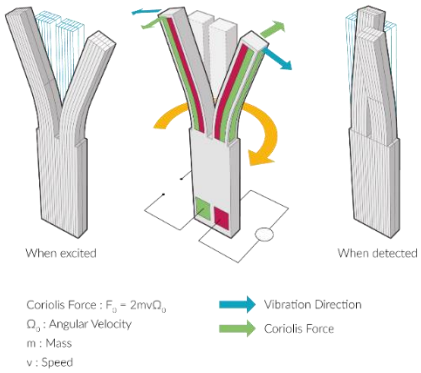
เซนเซอร์วัดความเร่ง สามารถวัดความเร่งตามแนวแกนต่าง ๆ ออกมาได้โดยใช้อุปกรณ์ที่เรียกว่า Micro Electronics Mechanical System (MEMS) ที่อยู่ด้านในดังในรูปที่ 2.11 โดย MEMS จะประกอบด้วย กลไกอิเล็กโทรดที่ขยับได้และเคลื่อนที่ได้อยู่ ซึ่งมวลมีค่าคงที่ เมื่อเซนเซอร์เกิดการเคลื่อนไหว อิเล็กโทรดด้านใน MEMS จะเคลื่อนไหวเข้าหากันและออกจากกันและเกิดการเปลี่ยนแปลงของค่าประจุไฟฟ้าและสามารถคำนวณความเร่งจากสัญญาณที่รับเข้ามาได้ และเมื่อทราบความเร่งจะสามารถนำไปอินทิเกรตเพื่อหาตำแหน่งการเคลื่อนที่ของเซนเซอร์ได้



รูปที่ 2.11 MEMS (Bennett, et. al., 2021)

ไจโรสโคป เป็นเซนเซอร์ที่ใช้วัดความเร็วเชิงมุมของวัตถุเมื่อมีการหมุนเกิดขึ้น โดยใช้หลักการของแรงโคริโอลิส (Coriolis's force) ซึ่งแรงนี้จะทำให้กลไกด้านในมีการบิดและถ่างตามทิศทางของแรงโคริโอลิสตามรูปที่ 2.12 ซึ่งเมื่อความจุของอิเล็กโทรดเปลี่ยนไปจะได้สัญญาณไฟฟ้าออกมาแล้วสามารถนำไปแปลงเป็นค่าความเร็วเชิงมุมในการหมุนได้ โดยสมการที่ใช้คำนวณแรงโคริโอลิส F_0 ร่วมกับมวล ความเร็ว v และความเร็วเชิงมุม Ω_0 เป็นดังต่อไปนี้

$$F_0 = 2mv\Omega_0 \tag{2.22}$$



รูปที่ 2.12 MEMS กับแรงโคริโอลิส (Bennett, et. al., 2021)

เมื่อวัดความเร็วเชิงมุมและความเร่งได้ก็จะสามารถนำข้อมูลดังกล่าวไปประมาณสภาวะของอากาศยานเพื่อทำการควบคุมได้

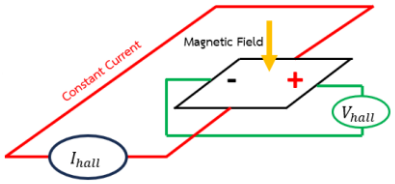
2.3.2 บารอมิเตอร์

บารอมิเตอร์ (barometer) เป็นเซนเซอร์ที่ใช้สำหรับวัดความกดอากาศ ซึ่งในอากาศยานไร้คนขับ การทราบความกดอากาศจะสามารถนำไปหาความสูงของอากาศยานได้ โดยบารอมอเตอร์จะมีวัสดุ piezoelectric ซึ่งทำมาจากสารกึ่งตัวนำ (semi-conductor) โดยความต้านทานจะเปลี่ยนไปเมื่อมีแรงดันมากระทำ ทำให้เราสามารถวัดความกดอากาศที่มากระทำกับเซนเซอร์ได้ ตัวอย่างของเซนเซอร์วัดความกดอากาศ เช่น BMP180 ดังในรูปที่ 2.13

รูปที่ 2.13 เซ็นเซอร์ BMP180

2.3.3 เซ็นเซอร์วัดสนามแม่เหล็ก

เซนเซอร์วัดสนามแม่เหล็ก (magnetometer) เป็นอุปกรณ์สำหรับวัดค่าความเข้มของสนามแม่เหล็กเพื่อนำไปหาทิศทางหรือใช้ประโยชน์อื่น ๆ โดยหลักการของ magnetometer ที่ใช้หลักการ hall effect จะสามารถแสดงได้ดังในรูปที่ 2.14

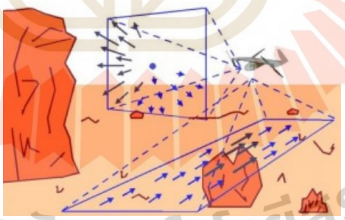


รูปที่ 2.14 หลักการ hall effect

ในวงจรที่มีการไหลของกระแสไฟฟ้า จะมีการไหลของประจุอิเล็กตรอนเกิดขึ้น เมื่อมีสนามแม่เหล็กที่มีความเข้ม B มาตัดผ่าน จะทำให้เกิดแรงขึ้นที่ประจุไฟฟ้าและเกิดการเหนี่ยวนำประจุไฟฟ้าให้เกิดการเลี้ยวเบนออกจากกันเกิดขึ้น ถูกเรียกว่า hall effect ซึ่งจะทำให้แรงดันเปลี่ยนแปลงไป และทำให้ทราบถึงขนาดความเข้มของสนามแม่เหล็ก (magnetic field) ได้

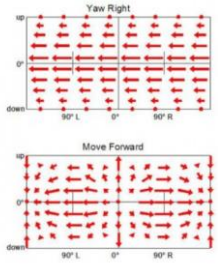
2.3.4 เซนเซอร์ Optical flow

Optical flow sensor เป็นเซนเซอร์สำหรับวัดระยะทางการเคลื่อนที่ของอากาศยานไร้คนขับขนาดเล็ก และนิยมใช้ในพื้นที่ในร่ม (indoor) เซนเซอร์ optical flow อาศัยหลักการของปรากฏการณ์ optical flow ซึ่งเป็นเหตุการณ์ที่เกิดขึ้นเป็นประจำในการมองเห็นของมนุษย์ เช่น เมื่อนั่งรถแล้วมองท้องฟ้าหรือวัตถุอื่นผ่านกระจกจะพบว่า เมื่อรถเกิดการเคลื่อนที่ วัตถุจะมีการเคลื่อนที่แตกต่างกันไป โดยวัตถุที่อยู่ใกล้จะเคลื่อนที่ไว ส่วนวัตถุที่อยู่ไกลจะเลื่อนไปอย่างช้า ๆ



รูปที่ 2.15 ตัวอย่างปรากฏการณ์ optical flow (Centeye, 2023)

สำหรับอากาศยานไร้คนขับ เมื่อติดตั้งกล้องในมุมมองไปที่พื้นแล้วนั้น การไหลของภาพจะเป็นดังลูกศรสีน้ำเงินดังรูปที่ 2.15 ซึ่งการไหลของภาพจะแตกต่างกันไปตามการเคลื่อนที่ดังเช่นลูกศรแสดงทิศทางการไหลของภาพดังในรูปที่ 2.16 ซึ่งภาพด้านบนคือตอนเลี้ยวขวา ส่วนภาพด้านล่างคือตอนเคลื่อนที่ไปด้านหน้า



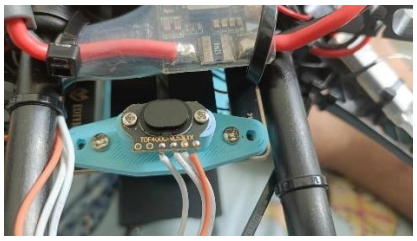
รูปที่ 2.16 ตัวอย่างปรากฏการณ์ optical flow ในการเลี้ยวและเคลื่อนที่ไปด้านหน้า (Centeye, 2023)

เมื่อใช้หลักการนี้ร่วมกับการประมวลผลภาพจะทำให้สามารถวัดการเคลื่อนที่ของวัตถุได้ ซึ่งสำหรับอากาศยานไร้คนขับจะสามารถใช้ในการหาตำแหน่งของอากาศยานไร้คนขับได้ ตัวอย่างของเซนเซอร์ optical flow แสดงดังในรูปที่ 2.17

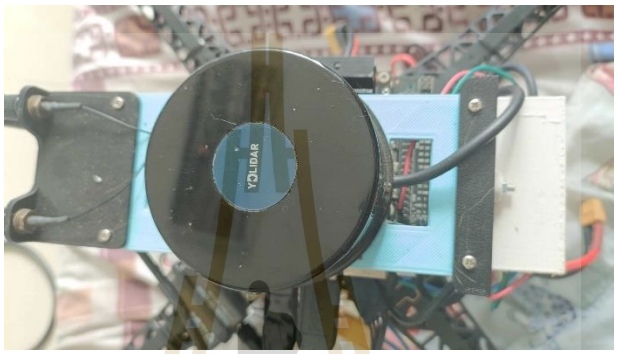
รูปที่ 2.17 PX4Flow optical flow sensor (Centeye, 2023)

2.3.5 เซนเซอร์ตรวจจับวัตถุด้วยแสงและวัดระยะทาง

เซนเซอร์ตรวจจับวัตถุด้วยแสงและวัดระยะทาง (Light Detection And Ranging) หรือ LiDAR เป็นเซนเซอร์ที่สามารถตรวจจับวัตถุวัดระยะห่างจากวัตถุได้โดยใช้แสงเลเซอร์ การคำนวณระยะทางของวัตถุจะใช้การคำนวณความเร็วของแสงในการคำนวณ โดยเมื่อทราบความเร็วการเดินทางของแสงและเวลาที่แสงมีการยิงออกไปแล้วสะท้อนกลับมาก็จะสามารถคำนวณหาระยะทางที่แสงมีการเดินทางไปได้ ในปัจจุบันนั้น LiDAR มีการพัฒนาไปมากตั้งแต่ LiDAR แบบยิงทางตรง ดังในรูปที่ 2.18 LiDAR แบบหมุนที่สามารถตรวจจับวัตถุได้ 360 องศาดังในรูปที่ 2.19 ไปจนถึง LiDAR แบบ3 มิติ



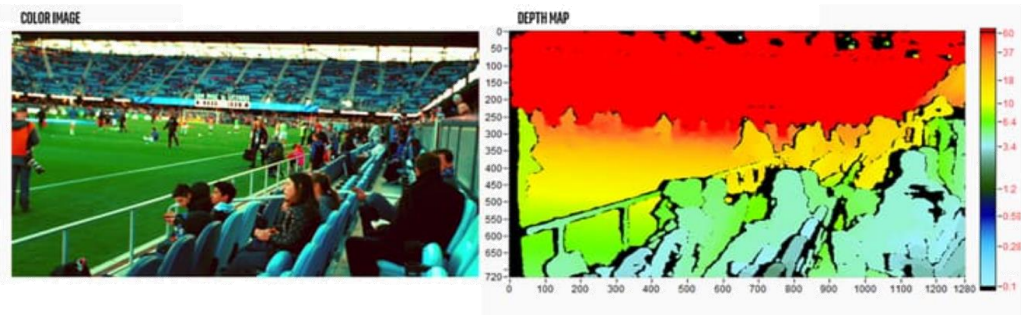
รูปที่ 2.18 LiDAR ทางเดียว



รูปที่ 2.19 LiDAR 360 องศา

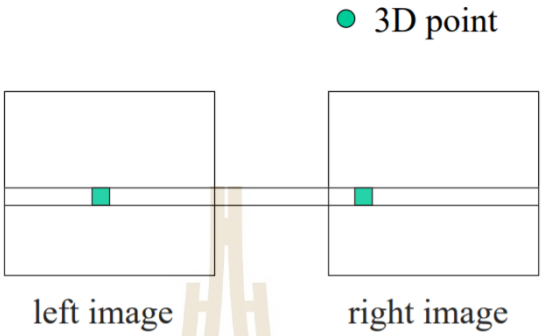
2.3.6 กล้องวัดความลึกแบบสเตอริโอ

กล้องวัดความลึกแบบสเตอริโอ (stereo depth camera) เป็นกล้องที่สามารถสร้างแผนที่ความลึก (depth map) ของภาพที่เห็นออกมาได้ หรือสามารถเรียกได้ว่าวัดความลึกของภาพได้ โดยเมื่อรวมกับค่าสีของภาพที่เป็นระบบ RGB แล้ว กล้องวัดความลึกจะมีพารามิเตอร์เพิ่มเข้ามากลายเป็น RGBD ตัวอย่างของภาพ RGBD แสดงดังในรูปด้านล่าง โดยด้านซ้ายคือภาพ RGB ส่วนด้านขวาแสดงแผนที่ความลึกของภาพโดยที่สีแดงเป็นการบ่งบอกว่าภาพมีความไกลส่วนสีเหลืองหรือสีอื่นที่ไล่เรียงเข้ามาจะมีระยะทางที่ใกล้เข้ามาตามลำดับ

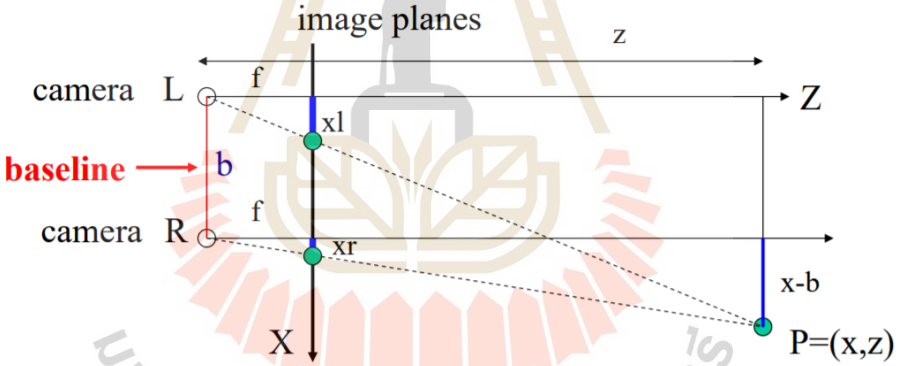


รูปที่ 2.20 ตัวอย่างภาพ RGBD (IntelRealsense, 2019)

หลักการทำงานของกล้องสเตอริโอจะเลียนแบบการรับรู้ความลึกของสายตามนุษย์ โดยสายตาของมนุษย์จะมีระยะห่างเล็กน้อย ในกล้องแบบเสตอริโอก็เช่นกัน กล้องจะมีเลนส์สองเลนส์ที่ติดตั้งห่างกันเล็กน้อยเป็นระยะที่เรียกว่า base line (d) ดังรูปด้านล่างต่อไปนี้



รูปที่ 2.21 เทคนิคสเตอริโอ (IntelRealsense, 2019)



รูปที่ 2.22 การใช้หลักการตรีโกณมิติกับการหาความลึกของรูปภาพ

จากหลักการของตรีโกณมิติและสามเหลี่ยมคล้าย เมื่อทราบพารามิเตอร์ของกล้องดังต่อไปนี้ x_l, x_r, y_l, y_r และ focal length (f) ซึ่งจะได้มาจากการคาลิเบรทตัวกล้องก็จะหาระยะความลึกของภาพได้ดังความสัมพันธ์ดังต่อไปนี้

$$z = f \times \frac{b}{(x_l - x_r)} \tag{2.23}$$

$$x = x_l \times \frac{z}{f} \tag{2.24}$$

$$y = y_l \times \frac{z}{f} \tag{2.25}$$

ในวงการหุ่นยนต์ กล้องวัดความลึกมีการนำไปประยุกต์ใช้กันแพร่หลายในการนำทางและหลบหลีกสิ่งกีดขวางอัตโนมัติ และในอากาศยานไร้คนขับก็สามารถนำไปประยุกต์ใช้ในการนำทางอัตโนมัติได้เช่นกัน ดังเช่น A.G Chabunichev (2020) นำกล้องวัดความลึกแบบสเตอริโอที่มีราคาถูกสำหรับเป็นอุปกรณ์อ้างอิงของการเคลื่อนที่ของหุ่นยนต์อัตโนมัติ ที่มีอากาศยานไร้คัยขับจอดอยู่ด้านบน และ Z. Sumin (2021) ที่นำกล้องวัดความลึกแบบสเตอริโอมาใช้กับหุ่นยนต์เคลื่อนที่และ VSLAM โดยใช้กล้องสเตอริโอสำหรับการประมาณโอโดเมทรีจากภาพ (visual odometry) ซึ่งผลการทดสอบให้ความคลาดเคลื่อนเพียงเล็กน้อย

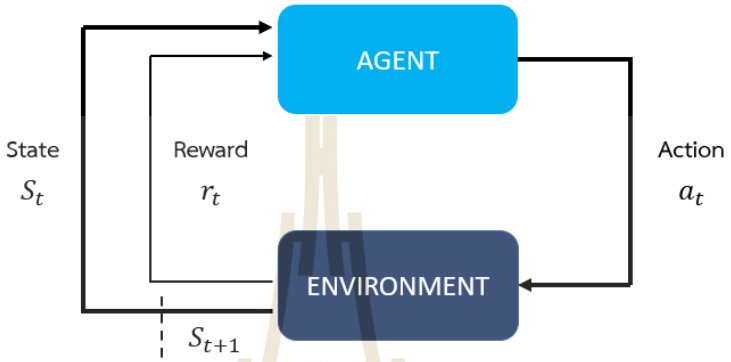
2.4 การเรียนรู้แบบเสริมกำลังเชิงลึก (Deep reinforcement learning)

การเรียนรู้แบบเสริมกำลังเชิงลึก (deep reinforcement learning) เป็นหนึ่งในแขนงของการเรียนรู้ของเครื่อง (machine learning) ในส่วนของการเรียนรู้แบบเสริมกำลัง (reinforcement learning) โดยเป็นการเรียนรู้แบบเสริมกำลังที่นำองค์ความรู้จากการเรียนรู้แบบมีผู้สอน (supervised learning) ซึ่งคือโครงข่ายประสาทเทียมเชิงลึก (deep neural network) มาใช้ในการประเมินค่าที่ออกมาจากฟังก์ชัน (function approximation) ในการทำงานของการเรียนรู้แบบเสริมกำลัง นอกจากนี้ยังมีการนำเทคนิค stochastic gradient descent และ backpropagation มาใช้ด้วยในกระบวนการเรียนรู้ของข้อมูล

2.4.1 แนวคิดการเรียนรู้แบบเสริมกำลังเบื้องต้น

การเรียนรู้แบบเสริมกำลังเป็นการเรียนรู้ที่เลียนแบบการเรียนรู้แบบลองผิดลองถูก (Trial and error) ของมนุษย์หรือสัตว์ต่าง ๆ โดยจะแตกต่างจาก supervised learning ตรงที่ไม่จำเป็นต้องทำฉลาก (label) ให้กับข้อมูล ตัวอย่างแนวคิดของการเรียนรู้แบบเสริมกำลังเช่น การฝึกสุนัข โดยเมื่อผู้ฝึกสุนัขจะฝึกให้สุนัขเก็บลูกบอล เมื่อสุนัขเจ้าลูกบอลได้ ผู้ฝึกจะมอบอาหารให้ สุนัขก็จะเรียนรู้ว่าหากต้องการอาหารจะต้องไปจับลูกบอล แต่หากสุนัขไม่ทำตาม ผู้ฝึกอาจจะลงโทษสุนัขด้วยการไม่ให้ขนม สุนัขก็จะเรียนรู้ว่าหากไม่ไปจับลูกบอลจะไม่ได้ขนม ทำให้สุนัขเกิดการเรียนรู้ไปในตัว หรือในปัญหาการนำทางของหุ่นยนต์ หากต้องการให้หุ่นยนต์นำทางไปถึงเป้าหมายได้โดยไม่ชน สามารถทำได้โดยให้หุ่นยนต์ลองเดินไปในทิศทางต่าง ๆ ด้วยตัวเอง เมื่อเดินได้รับทางที่ไกลขึ้นโดยไม่ชน อาจจะให้คะแนนเป็นรางวัล แต่หากชนจะหักคะแนนก็จะทำให้หุ่นยนต์สามารถเรียนรู้วิธีการไปถึงเป้าหมายโดยไม่ชนได้ ซึ่งปัญหาต่าง ๆ เหล่านี้ในการเรียนรู้แบบเสริมกำลังจะถูกโมเดลให้เข้าไปอยู่ในรูปแบบกระบวนการตัดสินใจแบบมาร์คอฟ (Markov decision process)

กระบวนการตัดสินใจแบบมาร์คอฟ (Markov Decision Processes : MDPs) เป็นกรอบงานทางคณิตศาสตร์ สำหรับการแก้ปัญหาโดยใช้การเรียนรู้แบบเสริมกำลังปัญหาส่วนใหญ่ในการเรียนรู้แบบเสริมกำลังล้วนสามารถนำมาสร้างแบบจำลองใน MDPs ได้ นอกจากนี้ MDPs ยังถูกใช้แก้ปัญหาการหาค่าที่เหมาะสมที่สุด (Optimization) อีกด้วย แผนผังอธิบายการทำงานของ MDPs แสดงดังในรูปที่ 2.23



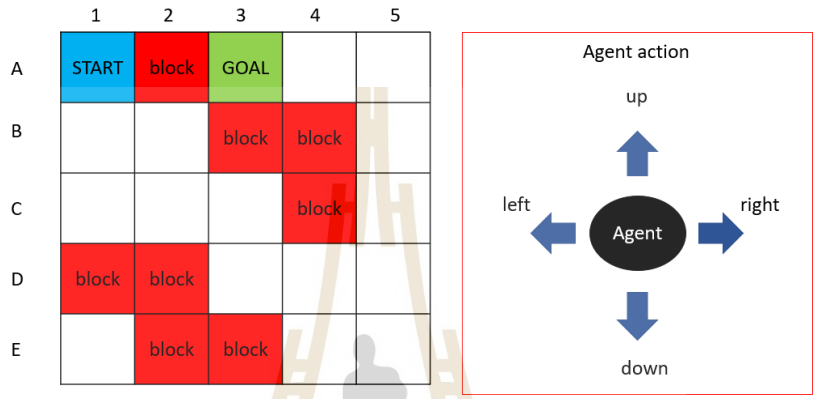
รูปที่ 2.23 แบบจำลองกระบวนการตัดสินใจแบบมาร์คอฟ

คุณสมบัติสำคัญของกระบวนการตัดสินใจแบบมาร์คอฟ (Markov property) คือ เหตุการณ์หรือสิ่งที่เกิดขึ้นในอนาคตจะขึ้นอยู่กับสิ่งที่ทำในปัจจุบันเท่านั้น เหตุการณ์ในอดีตจะไม่ส่งผลถึงอนาคตด้วย โดยกระบวนการตัดสินใจแบบมาร์คอฟจะประกอบด้วย ตัวแทน (agent), สภาพแวดล้อม (environment), สภาวะ (state), รางวัล (reward) และ การกระทำ (action) โดยความหมายของ state, reward และ action มีดังต่อไปนี้

1) State หรือ สภาวะ เป็นเซ็ต (set) ของสภาวะที่อยู่ในสิ่งแวดล้อม
2) Action หรือ การกระทำ เป็นเซ็ต (set) ของการกระทำที่ agent สามารถแสดงออกมาได้ในแต่ละ state
3) Reward หรือ รางวัล เป็นฟังก์ชันที่กำหนดผลของการกระทำของ agent ในแต่ละสภาวะในสภาพแวดล้อม

กระบวนการตัดสินใจแบบมาร์คอฟจะเริ่มต้นโดย ตัวแทนจะต้องรับทราบสภาวะที่ตนอยู่ในสภาพแวดล้อม เช่น ในสภาพแวดล้อมแบบ grid world ดังรูปที่ 2.24 หากตัวแทนมีเป้าหมายที่ต้องเดินจากจุด START ไปถึง GOAL โดยที่ไม่เข้าเดินไปใน block ตัวแทนจะต้องทราบสภาวะตัวเองก่อน จากนั้นตัวแทนจะแสดงการกระทำออกมาเพื่อเดินไปที่ GOAL โดยใน grid world นี้ action ที่ทำได้ของตัวแทนคือ การเดินขึ้น ลง ซ้าย ขวา ซึ่งใน state ที่จุด START นั้น action ที่

ทำได้จะมีเพียงเดินลงเท่านั้น จึงจะสามารถไปต่อได้ หากตัวแทนเดินลงมาจากจุด START เพื่อไปต่อ ตัวแทนจะมีการเปลี่ยนสภาวะกลายเป็นสภาวะ B1 โดยความน่าจะเป็นในการเปลี่ยนจากสภาวะเดิม (s) ไปยังสภาวะใหม่ (s') ด้วยผลจากการกระทำ (a) นั้นเรียกว่า ความน่าจะเป็นการเปลี่ยนผ่านสภาวะ (transitional probability) แทนด้วย $P(s'|s,a)$ และความน่าจะเป็นในการเปลี่ยนผ่านจากสภาวะเดิมไปสภาวะใหม่โดยไม่คิดจากการกระทำจะแทนด้วย $P(s'|s)$



รูปที่ 2.24 ตัวอย่างสภาพแวดล้อมแบบ grid world

เมื่อสภาวะของตัวแทนเปลี่ยนไป ตัวแทนจะได้รับรางวัลหรือบทลงโทษจาก reward function เช่น หากตัวแทนเดินเข้าไปในพื้นที่ Block จะถูกหักคะแนน -1 คะแนน แต่หากตัวแทนสามารถเดินต่อไปได้จะได้คะแนน +1 คะแนน แต่หากเข้าพื้นที่ GOAL ได้จะได้คะแนน +2 คะแนน เป็นต้น โดย reward function สามารถแทนได้ด้วย $R(s,a,s')$ ตัวแทนจะหาทางเดินจาก START ไป GOAL ให้ได้จนเจอเส้นทางที่ดีที่สุดหรือเส้นทางที่ทำให้ได้รางวัลมากที่สุดนั่นเอง และจะได้แผน (policy) ออกมา

ในการเรียนรู้แบบเสริมกำลัง สิ่งที่ต้องการได้ออกมาคือ policy ของตัวแทนในสภาพแวดล้อมต่าง ๆ โดย policy นั้นจะเป็นสิ่งกำหนดพฤติกรรมของตัวแทนในสภาพแวดล้อม policy จะเป็นสิ่งที่บอกตัวแทนว่าในสภาวะต่าง ๆ ควรทำการกระทำแบบใดจึงจะให้ผลลัพธ์ออกมาดีที่สุด ในตอนเริ่มต้นการเรียนรู้ policy จะถูกสุ่มขึ้นมาก่อน ดังนั้น การกระทำของตัวแทนในตอนเริ่มต้นจะถูกสุ่มมา โดยประเภทของ policy ถูกแบ่งออกเป็น 2 ประเภท คือ deterministic policy และ stochastic policy ซึ่งมีความหมายดังต่อไปนี้

1) Deterministic policy คือ policy ที่จะโยง (map) สภาวะให้เข้ากับการกระทำที่เจาะจง (particular action) มักถูกแทนด้วย μ

2) Stochastic policy แตกต่างจาก deterministic policy โดย stochastic policy จะไม่โยงสภาวะของตัวแทนเข้ากับการกระทำที่เฉพาะเจาะจงโดยตรง แต่จะโยงสภาวะกับการกระจายความน่าจะเป็น (probability distribution) ของการกระทำแต่ละแบบ ดังนั้น แม้ว่าตัวแทนจะอยู่ในสภาวะเดิม แต่การกระทำอาจจะแตกต่างกันออกไปก็ได้ตามความน่าจะเป็น stochastic policy มักจะถูกแทนด้วย π โดย policy สำหรับสภาวะที่เวลาหนึ่ง s_t กับการกระทำที่เวลานั้น a_t สามารถเขียนแทนได้ด้วย $\pi(a_t|s_t)$

การเลือกใช้ policy ขึ้นอยู่กับสิ่งแวดล้อมและตัวแทน ซึ่งในการเรียนรู้แต่ละรอบ (Episode) ตัว policy จะถูกปรับปรุงขึ้นเรื่อย ๆ จนเหมาะสมที่สุด (optimal policy) ซึ่งจะให้ผลรวมของรางวัลที่ดีที่สุด โดยจุดสิ้นสุดของแต่ละ episode สามารถทำได้โดยการกำหนดจำนวนการเปลี่ยนสภาวะ (step) หรือกำหนดเงื่อนไขอื่นในการสิ้นสุด episode โดยสภาวะ, การกระทำของตัวแทนและรางวัลที่ตัวแทนได้รับในแต่ละ episode สามารถถูกเรียกได้ว่า วิถี (trajectory) หรือแทนด้วย τ โดย trajectory ตั้งแต่เริ่มต้น (t = 0) ไปจนถึงเวลาสุดท้าย (t = T) สามารถเขียนได้อยู่ในรูปเซ็ต $(s_0, a_0, r_0, s_1, a_1, r_1, \dots, s_t)$

2.4.2 Return และ Discount factor

Return คือ ผลรวมของรางวัลทั้งหมดที่ตัวแทนได้รับในหนึ่งรอบการคำนวณ หรือ episode โดยเขียนแทนด้วยสัญลักษณ์ G ซึ่งเขียนเป็นสมการได้ดังต่อไปนี้

$$G(\tau) = \sum_{t=0}^{T} r_t \tag{2.26}$$

และสำหรับงานที่มีลักษณะเป็น continuous task นั้น จะสามารถเขียน return ได้ในรูปสมการต่อไปนี้

$$G(\tau) = \sum_{t=0}^{\infty} r_t \tag{2.27}$$

แต่หากเป็น continuous time นั้นจะเป็นเรื่องยากที่จะใช้ความสัมพันธ์ดังสมการที่ 2.27 เพื่อหา return ที่มีค่าที่สุดใน trajectory แต่ละชุด จึงต้องมีการเพิ่ม discount factor (γ) เข้าไปในสมการด้วย เพื่อบอกความสำคัญระหว่างรางวัลในอนาคต (future reward) และรางวัลฉับพลันในปัจจุบัน (immediate reward) ค่า discount factor จะมีค่าอยู่ระหว่าง 0 ถึง 1 โดยสมการของ return ที่มี discount factor จะเป็นดังสมการต่อไปนี้

$$G(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t \tag{2.28}$$

เมื่อเวลายิ่งผ่านไป return ก็จะยิ่งลดลง โดยหาก discount factor มีค่าเข้าใกล้ 0 มาก ค่า return ที่ได้จากรางวัลในอนาคตก็จะยิ่งลดลงอย่างรวดเร็วแตกต่างจาก discount factor ที่มีค่าเข้าใกล้ 1

2.4.3 Value function และ Q-function

Value function หรืออีกชื่อเรียกว่า state value function เป็นค่า return ที่ตัวแทนจะได้รับเมื่อเริ่มต้นจากสภาวะนั้นโดยทำตาม policy π สามารถเขียนแทนได้ด้วย $V(s)$ โดยสามารถแสดง value function ได้ดังรูปสมการต่อไปนี้

$$V^{\pi}(s) = [R(\tau)|s_0 = s] \tag{2.29}$$

ซึ่ง return ที่จะนำมาใช้คำนวณ value function นั้น ใน policy ที่เป็น stochastic policy ไม่สามารถคำนวณ return ได้โดยตรงเพราะการกระทำของตัวแทนขึ้นอยู่กับการกระจายตัวของความน่าจะเป็น ดังนั้น การหา value function จาก stochastic policy จะเป็นการคาดเดา (expected return) ซึ่งสามารถคำนวณ value function ได้จากสมการต่อไปนี้

$$V^{\pi}(s) = \mathrm{E}_{\tau\sim\pi}[R(\tau)|s_0 = s] = \sum_i R(\tau_i)\pi(a_i|A) \tag{2.30}$$

Value function จะแตกต่างกันไปตาม policy ที่เลือกใช้ Value function ที่มากที่สุดจะถูกเรียกว่า optimal value function ซึ่งถูกแสดงด้วยสมการต่อไปนี้

$$V^{*}(s) = \max V^{\pi}(s) \tag{2.31}$$

นอกจาก value function แล้ว อีกหนึ่งฟังก์ชันที่สำคัญที่สามารถแสดงความสัมพันธ์ของตัวแทนกับการเรียนรู้ได้อีกฟังก์ชันก็คือ state-action value function หรือเรียกว่า Q function โดย Q function นั้นจะเป็นค่า return ที่จะได้รับเมื่อตัวแทนเริ่มต้นในสภาวะ s และแสดงการกระทำ a โดยสามารถเขียนแทนด้วย $Q(s,a)$ และสามารแสดงได้ในรูปสมการดังต่อไปนี้

$$Q^{\pi}(s,a) = [\,R(\tau)|s_0 = s, a_0 = a] \tag{2.32}$$

คล้ายกับ value function ใน stochastic policy นั้นการกระจายตัวของความน่าจะเป็นในการกระทำของตัวแทนมีผลต่อ Q value นี้ด้วย ดังนั้น return ที่ได้มาโดย Q function นั้นจะสามารถเขียนใหม่ให้อยู่ในรูปสมการต่อไปนี้ได้

$$Q^{\pi}(s,a) = \mathrm{E}_{\tau\sim\pi}[\, R(\tau)|s_0 = s, a_0 = a] \tag{2.33}$$

Q-value ที่ให้ return มากที่สุดสามารถเขียนแสดงสมการได้ดังนี้

$$Q^{*}(s,a) = \max\, Q^{\pi}(s,a) \tag{2.34}$$

ทั้ง value function และ Q function จะถูกนำไปเป็นตัวแปรสำคัญในการเรียนรู้ของโมเดลด้วยซึ่งจะได้กล่าวถึงถัดไป

2.4.4 สมการของเบลล์แมน

สมการของเบลล์แมน (Bellman equation) เป็นสมการที่ใช้แพร่หลายในการเรียนรู้แบบเสริมกำลังสำหรับการแก้ปัญหาแบบมาร์คอฟ เพื่อที่จะหา policy ที่เหมาะสมที่สุด (optimal policy) สมการของเบลล์แมนสามารถแสดงได้ดังสมการต่อไปนี้

$$V(s) = R(s,a,s') + \gamma V(s') \tag{2.35}$$

ซึ่งสมการข้างต้นสามารถใช้ได้กับสภาพแวดล้อมที่เป็น deterministic environment เท่านั้น แต่หากเป็น stochastic environment การกระทำ a อาจไม่ส่งผลให้ตัวแทนเปลี่ยนสภาวะจาก s ไปสภาวะ s' ก็ได้ ดังนั้นจะมีการเพิ่มเทอมของความน่าจะเป็นในการเปลี่ยนสภาวะ(transition probability) และความน่าจะเป็นในการแสดงการกระทำ (action probability) เข้าไปในสมการด้วยดังสมการต่อไปนี้

$$V^{\pi}(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)\,[R(s,a,s') + \gamma V^{\pi}(s')] \tag{2.36}$$

สมการข้างต้นสามารถเรียกอีกชื่อได้เป็น สมการเบลล์แมนเชิงคาดการณ์ (Bellman expectation equation) ของ value function

ในส่วนของสมการเบลล์แมนของ state-action function นั้น จะมีความแตกต่างกับสมการเบลล์แมนของ value function เพียงเล็กน้อยเท่านั้น โดยสมการเบลล์แมนของ state-action value สามารถแสดงได้ดังต่อไปนี้

$$Q^{\pi}(s,a) = \sum_{s'} P(s'|s,a)\,[R(s,a,s') + \gamma \sum_{a'} \pi(a'|s') Q^{\pi}(s',a')] \quad (2.37)$$

ในการหา optimal policy นั้น จะต้องหา optimal value function และ optimal state-action value function โดยหาจากค่าที่ให้ return มากที่สุดของแต่ละตัวแปร ซึ่งจะทำให้สามารถเขียนสมการเบลล์แมนสำหรับตัวแปรที่เหมาะสมทั้งสองได้ดังต่อไปนี้

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} P(s'|s,a)\,[R(s,a,s') + \gamma V^{\pi}(s')] \quad (2.38)$$

ส่วนสมการของเบลล์แมนของ state-action value สามารถเขียนได้ดังสมการต่อไปนี้

$$Q^{\pi}(s,a) = \sum_{s'} P(s'|s,a)\,[R(s,a,s') + \gamma \sum_{a'} \pi(a'|s') Q^{\pi}(s',a')] \quad (2.39)$$

P.Huy, L. Hung และ N. Luan (2018) นำเสนอการใช้การเรียนรู้แบบเสริมกำลังเชิงลึก โดยใช้การหา state-action value ของอากาศยานไร้คนขับในสภาวะที่เป็น discrete ซึ่งมีขนาด 5x5 หน่วย มาทำการเรียนรู้การนำทางและหลบหลีกสิ่งกีดขวางรอบตัวโดยใช้ LiDAR โดยได้นำเสนอการใช้ function approximation ของ Q value มาแทนการหาด้วยวิธีการโดยตรง เพื่อที่จะลดเวลาในการคำนวณและสามารถนำไปประยุกต์ใช้กับสภาวะที่มีขนาดใหญ่ขึ้นได้ด้วย

2.4.5 วิธีการของมอนทิคาร์โล

วิธีการของมอนทิคาร์โล (Monte Carlo (MC) method) เป็นวิธีการในการหา policy ที่เหมาะสมที่สุด (optimal policy) วิธีการหนึ่ง สำหรับสภาพแสดล้อมที่ไม่ทราบแบบจำลองของระบบ (model-free) วิธีการนี้เป็นเทคนิคที่มักใช้ในทางสถิติในการประมาณการคาดการณ์ของตัวแปรสุ่ม (random variable) จากกลุ่มตัวอย่าง โดยปกติแล้วหากมีตัวแปรสุ่ม X แล้ว ค่าคาดการณ์ (expected value) ของตัวแปรสุ่ม ($E(x)$) สามารถหาได้จากสมการดังต่อไปนี้

$$E(x) = \sum_{i=1}^{n} x_i p(x_i) \quad (2.40)$$

แต่หากใช้วิธีการมอนทิคาร์โลจะสามารถประมาณค่าคาดการณ์ได้จากการสุ่มตัวอย่างขึ้นมา N ตัว จากนั้นหาค่าเฉลี่ยออกมาดังสมการต่อไปนี้

$$E_{x\sim p(x)}[X] \approx \frac{1}{N}\sum_i x_i \tag{2.41}$$

สำหรับปัญหาการเรียนรู้แบบเสริมกำลังนั้น ในการประมาณ value function เพื่อที่จะหา optimal policy สามารถประมาณได้จากการหาค่าเฉลี่ยของ return ในหลาย episode จากสมการดังต่อไปนี้

$$V(s) \approx \frac{1}{N}\sum_{i=1}^{N} R_i(s) \tag{2.42}$$

สำหรับการประมาณ state-action value จะคล้ายคลึงกับการหา value action โดยสามารถประมาณได้จากสมการดังต่อไปนี้

$$Q(s,a) \approx \frac{total\ return(s,a)}{N(s,a)} \tag{2.43}$$

จากหลักการของวิธีมอนทิคาร์โลที่ต้องคำนวณ state value หรือ state-action value โดยการเฉลี่ยค่าของทั้ง episode ดังนั้น จำเป็นต้องรอให้การเรียนรู้จบ episode ก่อนจึงจะสามารถหา optimal policy ได้ จึงไม่เหมาะสมกับงานที่ไม่มีจุดสิ้นสุด (non-episodic task)

2.4.6 วิธีการความต่างชั่วคราว

วิธีการความต่างชั่วคราว (Temporal Difference : TD method) เป็นอีกหนึ่งวิธีที่เป็นวิธีการในการหา policy ที่เหมาะสมที่สุด (optimal policy) สำหรับสภาพแสดล้อมที่ไม่ทราบแบบจำลองของระบบ (model-free) แต่มีการพัฒนาจากวิธีมอนทิคาร์โล คือ ไม่ต้องรอให้การคำนวณจบ 1 episode ก่อน ดังนั้นจะสามารถนำไปใช้ในงานที่มีความต่อเนื่อง (continuous task) ได้ โดยสมการสำหรับการประมาณค่า state value นั้นสามารถแสดงได้ดังในสมการต่อไปนี้ เมื่อ α คือ อัตราการเรียนรู้ (learning rate)

$$V(s) = V(s) + \alpha(r + \gamma V(s') - V(s)) \tag{2.44}$$

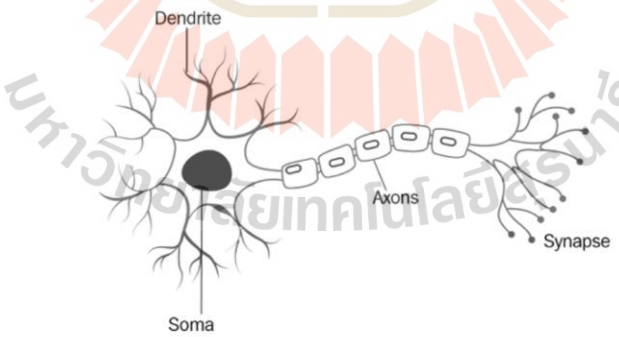
จากสมการข้างต้น พจน์ของ $r + \gamma V(s') - V(s)$ จะมีชื่อเรียกเฉพาะคือ Temporal Difference Error (TD Error) หากพิจารณา state-action value เพื่อหา optimal policy จะได้สมการการประมาณค่า state-action value ได้ดังสมการต่อไปนี้

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a)) \quad (2.45)$$

ในการหา Optimal policy โดยใช้ TD method นั้น จะเริ่มจากการสุ่มค่า Q ของทุก state ก่อนจากนั้นจะทำการอัพเดทและปรับปรุง policy อย่างต่อเนื่องจนเจอ policy ที่เหมาะสมที่สุด ทั้งวิธีการของมอนทิคาร์โลและวิธีการความต่างชั่วคราวนับเป็นอัลกอริทึมคลาสสิคในการแก้ปัญหาของการเรียนรู้แบบเสริมกำลัง ซึ่งในการเรียนรู้แบบเสริมกำลังเชิงลึกจะถูกนำไปใช้งานร่วมกับการเรียนรู้เชิงลึก (Deep learning) ซึ่งจะถูกกล่าวถึงต่อไป

2.4.7 แนวคิดการเรียนรู้เชิงลึก

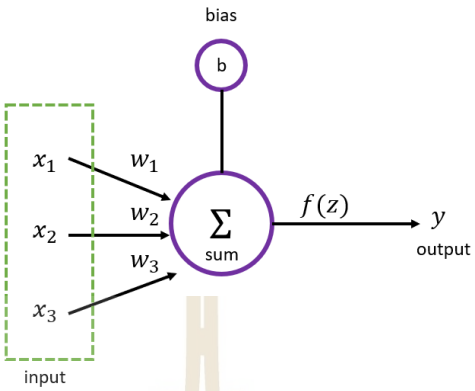
การเรียนรู้เชิงลึก (Deep Learning) เป็นแขนงหนึ่งของการเรียนรู้ของเครื่องจักร (Machine Learning) ซึ่งเกี่ยวข้องกับการใช้งานโครงข่ายประสาทเทียม (Neural Network) การเรียนรู้เชิงลึกเลียนแบบแนวคิดของระบบประสาทของมนุษย์แล้วถูกเรียกว่า โครงข่ายประสาทเทียม (Artificial Neural Network : ANN) โดยในทางชีววิทยาแล้ว ระบบประสาทของมนุษย์ประกอบด้วยเซลล์ประสาทจำนวนมาก โดยหนึ่งเซล์ประสาท (neuron) นั้นมีองค์ประกอบดังในรูปที่ 2.25



รูปที่ 2.25 โครงสร้างเซลล์ประสาทของมนุษย์ (Phongchit, 2018)

เซลล์ประสาทในสมองของมนุษย์มีอยู่ประมาณ 1 แสนล้านเซลล์ ซึ่งจะทำงานเชื่อมต่อกัน โดยส่วนที่มีการเชื่อมต่อกันจะเรียกว่า “ไซแนปส์ (synapse)” จากรูปที่ 2.25 เดนไดร์ท (dendrite) จะเป็นส่วนที่รับสัญญาณเข้ามาในนิวรอน โซมา (soma) คือส่วนกลางของเซลล์ประสาท และ แอกซอน (axon) คือส่วนที่จะส่งข้อมูลออกไปจากนิวรอนไปนิวรอนตัวอื่นผ่านไซแนปส์

ในโครงข่ายประสาทเทียมนั้น แบบจำลองของนิวรอนหนึ่งหน่วยนั้นสามารถแสดงได้ดังในรูปที่ 2.26

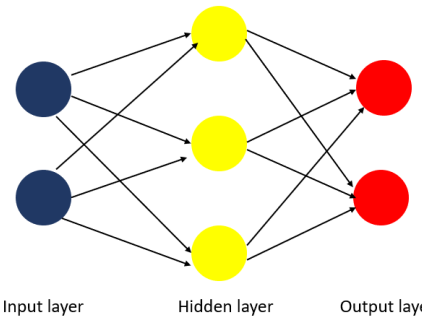


รูปที่ 2.26 แบบจำลองนิวรอนหนึ่งหน่วย

โดย x_i คือ ข้อมูลที่ถูกนำเข้ามา ส่วน y คือ ข้อมูลที่ถูกส่งออกไป w_i คือน้ำหนัก (weight) และ b คือ ไบแอส (bias) ค่าข้อมูลที่เข้ามาจะถูกนำมาคูณกับน้ำหนักเพื่อบอกความสำคัญของข้อมูลแต่ละตัวในการหาผลลัพธ์ และมีการบวกเพิ่มไบแอสเข้าไป โดยหากพิจารณาจากรูปที่ 2.26 จะได้ความสัมพันธ์ดังสมการต่อไปนี้

$$z = (x_1 w_1 + x_2 w_2 + x_3 w_3) + b \tag{2.46}$$

ในโครงข่ายประสาทเทียมนั้น จะประกอบด้วยนิวรอนหลายตัวที่โยงกันเป็นโครงข่าย (network) ดังในรูปที่ 2.27 ซึ่งเป็นโครงข่ายประสาทเทียมแบบพื้นฐาน โดย input layer เป็นชั้นที่รับข้อมูลเข้ามา จะไม่มีการคำนวณในขั้นนี้ hidden layer เป็นชั้นที่อยู่ระหว่างชั้น input และ output โดย hidden layer จะทำการประมวลผลและเรียนรู้ความสัมพันธ์ที่ซับซ้อนของ input และ output



รูปที่ 2.27 แบบจำลองโครงข่ายประสาทเทียมเบื้องต้น

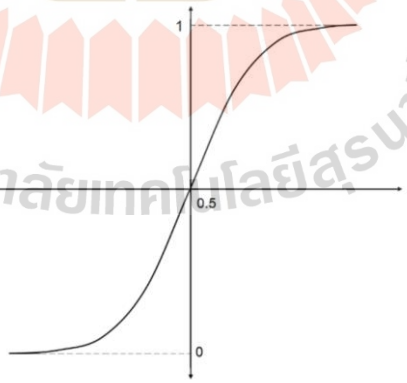
hidden layer สามารถมีกี่ชั้นก็ได้ขึ้นอยู่กับความซับซ้อนของปัญหา โดยหากจำนวนของ hidden layer มีมากกว่า 1 ชั้น โครงข่ายประสาทเทียมนั้นจะถูกเรียกว่า โครงข่ายประสาทเชิงลึก (Deep Neural Network) ส่วน output layer นั้นจะรับค่าออกมาจาก hidden layer โดยหากเป็นปัญหาการจำแนกประเภท (classification) จำนวนนิวรอนในชั้นนี้จะเท่ากับประเภทที่ต้องการจำแนก แต่หากเป็นปัญหาการทำ regression นั้น จำนวนนิวรอนจะมีเพียงตัวเดียว หลังจาก hidden layer ส่งค่าออกมานั้น เพื่อให้เกิดการเรียนรู้รูปแบบของข้อมูลที่ซับซ้อนและมีความคล้ายคลึงกัน จึงจำเป็นต้องเพิ่มความไม่เป็นเชิงเส้น (non-linearity) เข้าไปด้วย โดยใช้ activation function มาเป็นเครื่องมือ

activation function จะมีหลายรูปแบบ แต่รูปแบบที่นิยมใช้กันจะแบ่งเป็น ฟังก์ชันซิกมอยด์ (sigmoid function), ฟังก์ชันแทนเจนต์ไฮเปอร์โบลิค (tanh function), Rectified Linear Unit Function และฟังก์ชันซอฟท์แม็กซ์ (softmax function) โดยรูปแบบสมการของฟังก์ชันแต่ละรูปแบบจะแสดงดังต่อไปนี้

Sigmoid function

$$f(x) = \frac{1}{1-e^{-x}} \tag{2.47}$$

จากสมการข้างต้น สามารถพลอตเป็นกราฟได้ดังรูปต่อไปนี้

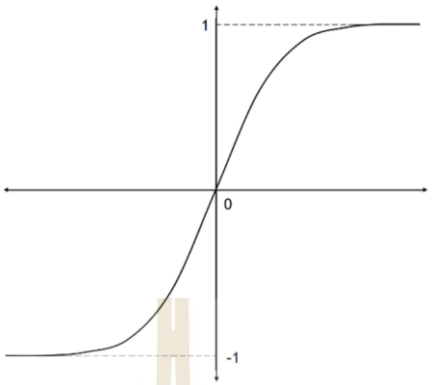


รูปที่ 2.28 Sigmoid function

Tanh function

$$f(x) = \frac{1-e^{-2x}}{1+e^{-2x}} \tag{2.48}$$

จากสมการข้างต้น สามารถพลอตเป็นกราฟได้ดังรูปต่อไปนี้

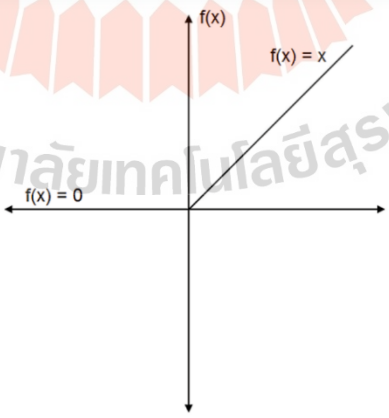


รูปที่ 2.29 Tanh function

Rectified Linear Unit Function (ReLU) function

$$f(x) = \begin{cases} 0 \; for \; x < 0 \\ x \; for \; x \geq 0 \end{cases} \tag{2.49}$$

จากสมการข้างต้น สามารถพลอตเป็นกราฟได้ดังรูปต่อไปนี้



รูปที่ 2.30 ReLU function

Softmax function

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{2.50}$$

โดยฟังก์ชันของซอฟท์แม็กซ์จะให้เป็นความน่าจะเป็นของเอาท์พุทออกมา

การเรียนรู้เชิงลึกด้วยโครงข่ายประสาทเทียมจะใช้วิธีการ forward propagation ส่งผ่านข้อมูลในแต่ละ layer จนนำไปสู่การทำนาย output จาก input ที่เข้ามา แต่การทำ forward propagation นั้น ไม่สามารถบอกได้ว่าผลที่ได้ออกมาจากการทำนายนั้นถูกต้องหรือไม่ โดยกำหนด cost function (J) ขึ้นมาดังสมการต่อไปนี้

$$J = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 \tag{2.51}$$

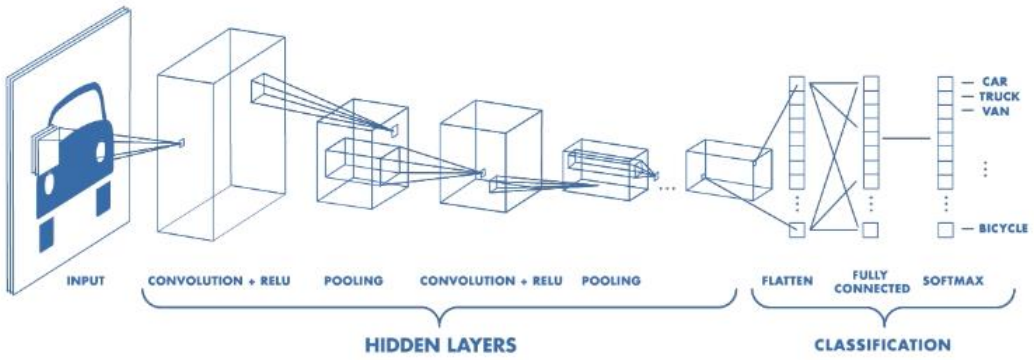
ซึ่ง cost function ที่กำหนดขึ้นนี้ เป็นการหาความคลาดเคลื่อนระหว่างข้อมูลจริง (y_i) และข้อมูลที่ได้จากการทำนาย $(\hat{y}_i)$ โดยใช้การหาความคลาดเคลื่อนแบบค่าเฉลี่ยกำลังสอง (mean squared error) ซึ่งหากค่าคลาดเคลื่อน (loss) จากการฝึกโมเดลมีน้อย แสดงว่าโมเดลมีความแม่นยำในการทำนาย โดยในการเรียนรู้เชิงลึกนั้นจะมีการทำ backpropagation เพื่อทำการปรับน้ำหนักของโมเดลในแต่ละ layer ให้เหมาะสม เพื่อที่จะทำให้ค่าจาก loss function มีค่าน้อยที่สุด โดยใช้หลักการของ gradient descent ในการปรับค่าน้ำหนัก ซึ่งสมการสำหรับการอัพเดทน้ำหนักของโมเดลนั้นสามารถแสดงได้ดังสมการต่อไปนี้ เมื่อ α คืออัตราการเรียนรู้ (learning rate)

$$W = W - \alpha\frac{\partial J}{\partial w} \tag{2.52}$$

การเลือกอัตราการเรียนรู้ที่เหมาะสมจะช่วยให้โมเดลนั้นเรียนรู้ที่จะทำให้ loss function มีค่าน้อยลงได้ไวมากขึ้นและลู่เข้าในที่สุด จนได้โมเดลความสัมพันธ์ระหว่าง input และ output ออกมา

2.4.8 โครงข่ายประสาทแบบคอนโวลูชัน

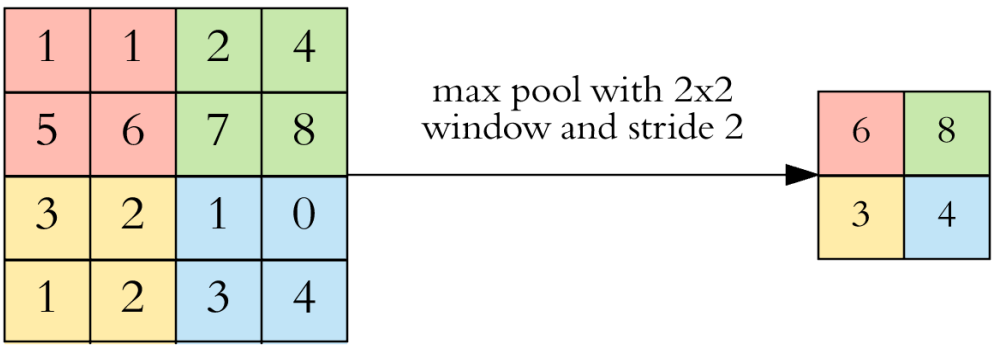
โครงข่ายประสาทแบบคอนโวลูชัน (Convolution Neural Network: CNN) เป็นอัลกอริทึมหนึ่งที่ใช้กันกว้างขวางสำหรับงานด้านการมองเห็นของคอมพิวเตอร์ โดย CNN จะประกอบด้วยชั้นการประมวลผลสามส่วนที่สำคัญ คือ ชั้นคอนโวลูชัน (convolutional layer), ชั้นพูลลิ่ง (pooling layer) และชั้นเชื่อมต่อสมบูรณ์ (fully connected layer) ดังในรูปที่ 2.31



รูปที่ 2.31 ตัวอย่าง Convolutional Neural Network (Phongchit, 2018)

<u>ชั้นคอนโวลูทชัน</u> นั้น เป็นชั้นที่ทำหน้าที่เป็นอย่างแรกของ CNN โดยจะทำหน้าที่แยกฟีเจอร์ (feature) ที่สำคัญออกมาจากรูปภาพ โดยการทำคอนโวลูชั่นกับรูปภาพด้วยตัวกรอง (filter / kernel) จะทำให้ภาพที่ผ่านการคอนโวลูทชันจะเหลือรายละเอียดที่สำคัญเท่านั้นก่อนที่จะนำไปประมวลผลในขั้นต่อไป การเลือกใช้ตัวกรองนั้นสามารถใช้ตัวกรองหลายตัวได้ โดยสิ่งที่ได้ออกมาจะถูกเรียกว่า feature maps ซึ่งจำนวนของ feature maps จะเท่ากับจำนวนของตัวกรองที่ใช้ในการทำคอนโวลูทชัน ในตอนเริ่มต้นนั้นตัวกรองจะถูกสุ่มขึ้นมาก่อน แล้วทำการปรับให้เหมาะสมมากขึ้นผ่านการเรียนรู้และ backpropagation

<u>ชั้นพูลลิ่ง</u> คือ ชั้นที่ใช้สำหรับลดขนาดมิติของภาพหลังจากที่ผ่านการทำคอนโวลูทชันมาแล้ว โดยวิธีการทำพูลลิ่งนั้น มีหลายแบบ เช่น max pooling, mean pooling และ sum pooling โดยตัวอย่างของ max pooling แสดงดังในรูปด้านล่าง โดยเป็นการใช้ตัวกรองขนาด 2x2 กับ feature maps ด้วยลำดับขั้นการเลื่อนตัวกรอง (stride) เป็น 2 แล้วหาค่าสูงสุดในตัวกรองที่เกิดขึ้น จนได้ feature maps ที่มีขนาดลดลงดังในรูปด้านขวามือ



รูปที่ 2.32 Max pooling (Phongchit, 2018)

ชั้นเชื่อมต่อสมบูรณ์ คือ ชั้นที่ทำหน้าที่ในการจำแนกประเภท(classification) ของภาพที่ถูกนำเข้ามาเพื่อบอกว่าภาพนั้นคืออะไร feature maps ที่ได้มาจาก ชั้นคอนโวลูทชันและชั้นพูลลิ่งจะถูกแปลงเป็นเวกเตอร์ (vector) และนำเข้าไปในโครงข่ายประสาทเทียมใน fully connected layer แล้วทำการจำแนกภาพต่อไป

2.4.9 Deep Q Network

Deep Q Network เป็นวิธีการหนึ่งของการเรียนรู้แบบเสริมกำลังเชิงลึก (deep reinforcement learning) จากหลักการใช้ state-action value (Q) เพื่อแยก optimal policy ออกมาของการเรียนรู้แบบเสริมกำลัง จะเห็นว่าการคำนวณ $Q\ value$ โดยตรงนั้นจะต้องคำนวณจากจำนวน state ทั้งหมด แต่หาก state มีจำนวนที่เยอะมาก วิธีการคำนวณ state-action value แบบนี้นั้นจะต้องใช้ทรัพยากรการคำนวณมหาศาล ดังนั้น จึงมีการคิดค้น ฟังก์ชันการประมาณค่า (function approximator) ขึ้นมาสำหรับการประมาณค่า state-action value โดยใช้โครงข่ายประสาทเทียมโดยมีพารามิเตอร์ของโครงข่ายประสาทเทียมคือ θ และ state-action value ที่ถูกประมาณด้วยโครงข่ายประสาทเทียมจะถูกแทนด้วยตัวแปร $Q_\theta(s,a)$

ในการฝึกโมเดลสำหรับ DQN นั้น ข้อมูลที่จำเป็นในการป้อนเข้าไปคือ replay buffer ซึ่งเป็น buffer ที่จะเก็บประสบการณ์ของตัวแทนเมื่อมีการกระทำเกิดขึ้นและมีการเปลี่ยนผ่านจากสภาวะตั้งต้นไปสภาวะถัดไป ซึ่งข้อมูลนั้นสามารถถูกเก็บไว้ได้ในรูปแบบของ (s,a,r,s') โดย replay buffer จะถูกแทนด้วยตัวแปร D ในการคำนวณความคลาดเคลื่อนของการเรียนรู้จะใช้ loss function สำหรับ DQN จะแทนด้วย $L(\theta)$ ซึ่งหาได้จากสมการต่อไปนี้

$$L(\theta) = Q^*(s,a) - Q_\theta(s,a) \tag{2.53}$$

โดย $Q^*(s,a)$ คือ optimal policy ของตัวแทน โดยสามารถหาได้จากสมการของเบลล์แมนต่อไปนี้

$$Q^*(s,a) = r + \gamma \max_{a'} Q^*(s',a') \tag{2.54}$$

เมื่อแทนค่าสมการลงไปใน loss function จะได้สมการของ loss function ขึ้นมาใหม่ดังนี้

$$L(\theta) = r + \gamma \max_{a'} Q(s',a') - Q_\theta(s,a) \tag{2.55}$$

ค่าของ $Q(s', a')$ สามารถประมาณได้โดยใช้โครงข่ายประสาทเทียมเช่นเดียวกันจึงได้สมการต่อไปนี้

$$L(\theta) = r + \gamma \max_{a'} Q_\theta(s', a') - Q_\theta(s, a) \tag{2.56}$$

หากเปลี่ยน loss function เป็น mean squared error จะได้ loss function ดังสมการต่อไปนี้ โดย K คือจำนวนข้อมูลที่นำมาใช้ในการฝึกโมเดล

$$L(\theta) = \frac{1}{K}\sum_{i=1}^{K}(r + \gamma \max_{a'} Q_\theta(s', a') - Q_\theta(s, a))^2 \tag{2.57}$$

แต่จากการประมาณทั้ง $Q_\theta(s', a')$ และ $Q_\theta(s, a)$ ด้วยพารามิเตอร์ θ เดียวกันทำให้การลู่เข้านั้นเป็นเรื่องยาก จึงต้องมีการแยกคำนวณ stat-action value ทั้งสอง ทีละตัวโดย $Q_\theta(s', a')$ จะถูกประมาณด้วยโครงข่ายประสาทเทียมที่ถูกเรียกว่า target network ส่วนโครงข่ายประสาทเทียมที่ใช้ในการประมาณ $Q_\theta(s, a)$ จะถูกเรียกว่า main network และเมื่อนำ loss function ไปคำนวณด้วย backpropagation และ gradient descent จะได้ความสัมพันธ์ในการอัพเดท θ ดังสมการต่อไปนี้

$$\theta = \theta - \alpha \nabla_\theta L(\theta) \tag{2.58}$$

ในการประยุกต์ใช้งาน DQN กับอากาศยานไร้คนขับ สามารถทำได้โดยสร้างสภาวะจำลองในโปรแกรม simulator เพื่อทำการฝึกโมเดลก่อนแล้วทำการส่งผ่าน (transfer) การเรียนรู้ไปยังอุปกรณ์จริง ซึ่ง C. Efe และคณะ (2020) ใช้ DQN ร่วมกับการนำทางของอากาศยานไร้คนขับและทำการโมเดลสภาวะด้วยภาพจากกล้องวัดความลึก (dept camera) และตำแหน่งสัมพัทธ์ (relative position) ของอากาศยานไร้คนขับ แล้วนำไปเทียบกับวิธีการนำทางด้วย potential field สภาวะจำลองถูกสร้างขึ้นในโปรแกรม AirSim simulator โดยมีสภาวะ 10 แบบ แล้วนำไปใส่ในอากาศยานจริง ผลการทดสอบพบว่าอากาศยานสามารถนำทางตัวเองได้โดยใช้ DQN และใช้เวลาใกล้เคียงกับวิธีการ potential field นอกจากนี้เมื่อ C. Yun และคณะ (2020) ทดลองใช้การตรวจจับวัตถุ (object detection) จากกล้องโมโนคูลาร์ (monocular) มาช่วย DQN ในการควบคุมอากาศยานไร้คนขับ พบว่าวิธีนี้ช่วยในการป้องกันการชนได้ดีกว่าการใช้เพียงการเรียนรู้แบบเสริมกำลังและ DQN

Y. Pengyu (2019) ใช้การเรียนรู้เชิงลึกแบบ DQN ร่วมกับหุ่นยนต์เคลื่อนที่ในการทำการนำทางแบบอัตโนมัติ โดยรับภาพ RGB จากกล้องมาเป็นข้อมูลเข้าให้กับ DQN ซึ่งการนำทางสามารถทำได้ แต่ยังมีความคลาดเคลื่อนของตำแหน่งที่เยอะ ส่วน T. Minh (2020) นำเสนออีกวิธีการ

หนึ่งในการรับค่าสภาวะของตัวแทนเพื่อทำงานกับอัลกอริทึมการเรียนรู้เสริมกำลังเชิงลึก โดยใช้กล้อง monocular แล้วแปลงภาพที่เข้ามาให้เป็น dept image ด้วยการเรียนรู้เชิงลึก ซึ่งโมเดลที่ใช้ทำนายคือโมเดล UNet โดยมี Resnet-18 เป็นเอนโคเดอร์ และใช้ pretrained model จาก KITTI dataset ซึ่งผลการทดสอบให้ผลที่ใกล้เคียงกันกับการใช้ LiDAR ในการนำทาง แต่มีข้อเสียคือต้องใช้ทรัพยากรในการคำนวณมากขึ้น เพราะต้องใช้การเรียนรู้เชิงลึกในการทำนายภาพที่เข้ามาเป็น dept image ด้วย

2.4.10 Double Deep Q Network

Double Deep Q Network (DDQN) เป็นอัลกอริทึมที่พัฒนามาจาก DQN โดยต้องการแก้ปัญหาการประมาณค่าที่มากเกินไป (overestimation) ของค่า Q อันเนื่องมาจากสิ่งรบกวน (noise) ในการประมวลผล ซึ่งอาจจะทำให้การเลือกการกระทำของตัวแทนผิดพลาดไป หลักการของ DDQN คือการปรับปรุง target value (y) ให้กลายเป็นสมการดังต่อไปนี้

$$y = r + \gamma Q_{\theta'}(s', \underset{a'}{\operatorname{argmax}}\, Q_{\theta}(s', a')) \tag{2.59}$$

โดย Q function ในสมการนี้จะถูกประมาณด้วย main network θ ก่อนจากนั้นจะมีการเลือก action ออกมา เพื่อคำนวณค่า Q $value$ จาก target network θ' อีกครั้ง จึงจะทำให้ overestimation เกิดได้น้อยลง

C. Ender และคณะ (2019) นำเสนอการประยุกต์ใช้ DDQN กับอากาศยานไร้คนขับ 4 ใบพัด เพื่อนำทางและหลบหลีกสิ่งกีดขวางที่มีการเคลื่อนที่ นั่นคืออากาศยานสี่ใบพัดอีก 2 ลำซึ่งจะเคลื่อนที่แบบสุ่ม ในการโมเดลสภาวะของตัวแทนจะใช้ภาพจากกล้องวัดความลึก และสภาวะเชิงสเกลาร์ของอากาศยาน เช่น ความเร็ว การชน ระยะห่างถึงจุดหมาย โดยใช้ Joint Neural Network (JNN) ในการเชื่อมต่อข้อมูลขาเข้าทั้งสอง ก่อนที่จะนำไปเข้า DDQN จากการทดสอบพบว่าในช่วงแรกของการเรียนรู้ อากาศยานผู้เรียนรู้จะมีการชนกับอากาศยานที่เป็นอุปสรรค แต่เมื่อจำนวนรอบของการเรียนรู้มากขึ้นก็สามารถหลบหลีกอากาศยานที่ทำหน้าที่ขัดขวางได้

2.4.11 Dueling Deep Q Network

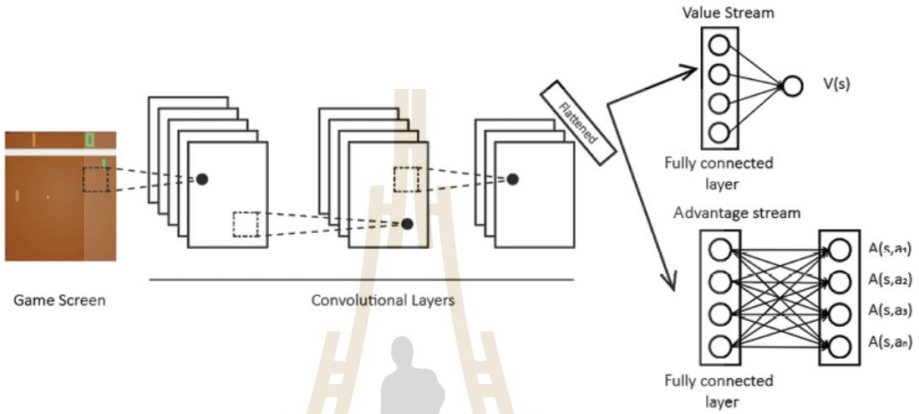
Dueling deep Q network (D-DQN) เป็นอัลกอริทึมหนึ่งที่มีการพัฒนามาจาก DQN โดยมีพารามิเตอร์ที่สำคัญเพิ่มเข้ามาเรียกว่า advantage value $(A(s,a))$ โดยเกิดจากผลต่างของ state-action value และ state value function โดยแสดงความสัมพันธ์ได้ดังสมการต่อไปนี้

$$A(s,a) = Q(s,a) - V(s) \tag{2.60}$$

ซึ่งเมื่อจัดรูปสมการใหม่เพื่อหา state-action value จะได้ว่า

$$Q(s,a) = A(s,a) + V(s) \tag{2.61}$$

สำหรับ advantage value นั้น จะเป็นค่าที่บ่งชี้ว่าการกระทำใดในสภาวะที่ตัวแทนอยู่นั้นไม่มีผลต่อการเปลี่ยนสภาวะของตัวแทน โดยการกระทำที่ไม่มีผลต่อการเปลี่ยนสภาวะของตัวแทนนั้นจะสามารถถูกแยกออกจากการคำนวณได้ เพื่อลดขนาดการคำนวณที่ต้องใช้ได้



รูปที่ 2.33 Architecture ของ Dueling DQN

จากรูปที่ 2.33 เป็นสถาปัตยกรรมของ dueling DQN โดยจะมีการแบ่งแยกข้อมูลที่เข้ามาเป็น 2 ฟังก์ชัน โดยมี value stream function และ advantage stream function ทำหน้าที่ในการประมาณ state value และ advantage value ตามลำดับ โดยใช้การประมาณด้วยโครงข่ายประสาทเทียม เมื่อหน่วยคำนวณทั้งสองทำงานเสร็จสิ้น ค่าที่ได้จะถูกนำมารวมกันด้วย aggregate layer เพื่อทำการประมาณ state-action value ต่อไป แต่เพื่อลดปัญหาการระบุตัวตน (problem of identifiability) จึงต้องทำให้ advantage value ของ action ที่เลือกให้เป็นศูนย์โดยนำ advantage value ที่ได้ไปลบกับค่าเฉลี่ยของ advantage value ทั้งหมดดังสมการต่อไปนี้

$$Q(s,a) = V(s) + (A(s,a) - \frac{1}{\mathcal{A}}\sum_{a'} A(s,a')) \tag{2.62}$$

โดย $\mathcal{A}$ คือ ขนาดของ action space ที่สามารถกระทำได้ เมื่อคำนวณ state-action value ได้แล้วก็สามารถนำไปเลือก action ที่ได้ที่สุดออกมาได้

S. Sang (2019) ได้ลองประยุกต์ใช้ dueling deep Q network และ double deep Q network ทำงานร่วมกัน กลายเป็น double dueling deep Q network เพื่อประยุกต์ใช้

กับการนำทางของอากาศยานไร้คนขับ โดยมีการเคลื่อนที่แบบสามมิติ ในสภาพแวดล้อมที่เป็นป่าไม้และเป็นกล่องวัตถุต่าง ๆ และมีการเทียบผลการบินกับการใช้นักบินที่มีความเชี่ยวชาญในระดับต่าง ๆ ด้วยระบบ Hardware In The Loop (HITL) กับอัลกอริทึมทั้งสามแบบ ผลการทดสอบพบว่า DDQN ไม่สามารถทำให้การนำทางสำเร็จได้ แต่ D-DQN และ D-DDQN สามารถทำได้ แต่ D-DDQN ให้ผลดีที่สุด แต่ยังได้ reward น้อยกว่าการใช้นักบินระดับมีความเชี่ยวชาญสูงบิน แต่ชนะนักบินที่มีความเชี่ยวชาญระดับปานกลาง

2.4.12 Policy Gradient Method

จากวิธีการก่อนหน้านี้ เราสามารถหา optimal policy ได้จากการหา state-action value หรือ Q ของตัวแทน ซึ่งวิธีการดังกล่าวเรียกว่า value based method แต่ข้อจำกัดของวิธีดังกล่าวคือสามารถใช้ได้กับ discrete action space เพียงเท่านั้น แต่ในโลกความเป็นจริงนั้น ปัญหาส่วนใหญ่เป็นปัญหาแบบต่อเนื่อง (continuous task) เช่น การควบคุมความเร็วของรถเป็นต้น ดังนั้น วิธีการแบบ policy gradient จึงถูกพัฒนาขึ้นโดยสามารถหา optimal policy ได้โดยตรงและสามารถใช้ได้กับทั้งปัญหาแบบไม่ต่อเนื่องและปัญหาแบบต่อเนื่อง สิ่งที่ได้ออกมาจากวิธี policy gradient จะออกมาเป็นความน่าจะเป็นของ action แต่ละตัวที่ตัวแทนสามารถกระทำได้ในสภาวะนั้น จะแตกต่างจากวิธีการ value-based ที่สิ่งที่ได้ออกมาจะเป็น stat-action value ของการกระทำในแต่ละสภาวะ โดยในตอนเริ่มแรกความน่าจะเป็นของการกระทำแต่ละตัวจะยังไม่แม่นยำ แต่เมื่อมีการเรียนรู้อย่างต่อเนื่อง ความน่าจะเป็นจะถูกปรับเพิ่มขึ้นเมื่อการกระทำนั้นให้ return ที่มีค่ามากขึ้น สมการในการอัพเดทค่าของโครงข่ายประสาทเทียม θ โดยวิธี gradient descent สามารถเขียนได้ดังสมการต่อไปนี้

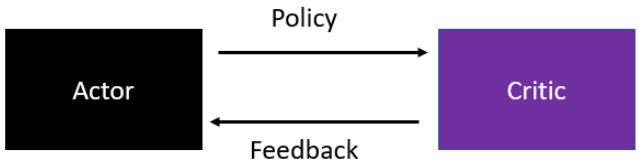
$$\theta = \theta + \frac{1}{N}\sum_{i=1}^{N}[\sum_{t=0}^{T-1}\nabla_\theta \, log\pi_\theta(a_t|s_t)R(\tau)] \tag{2.63}$$

เมื่อ $\log \pi_\theta(a_t|s_t)$ คือ ลอการิทึมของความน่าจะเป็นของการกระทำ a ในสภาวะ s ณ เวลา t และ $R(\tau)$ คือ return ที่ได้มาจาก trajectory ของการเรียนรู้ในแต่ละ episode ซึ่งวิธีการ policy gradient นี้มีชื่อเรียกว่า REINFORCE แต่การใช้วิธีนี้นั้นค่าเกรเดียน $\nabla_\theta J(\theta)$ จะมีความแปรปรวน (variances) สูงในการอัพเดตแต่ละครั้ง ซึ่งจะทำให้เวลาที่ต้องใช้ในการเรียนรู้จนลู่เข้านั้นใช้เวลาที่นานมาก

2.4.13 Actor Critic

Actor Critic เป็นวิธีการหา optimal policy โดยรวมเอาข้อดีของวิธีการแบบ value based และวิธีการแบบ policy based เข้าด้วยกัน วิธีการ Actor Critic ประกอบด้วยโครงข่ายประสาทเทียม 2 ตัว คือ actor network และ critic network โดยหน้าที่ของ actor

network คือการหา optimal policy และ critic network จะทำหน้าที่ประเมิน policy ที่ได้มาจาก actor และเป็นแนวทางให้ actor ทำการปรับปรุง actor ให้ได้ optimal policy ออกมา



รูปที่ 2.34 Actor – Critic

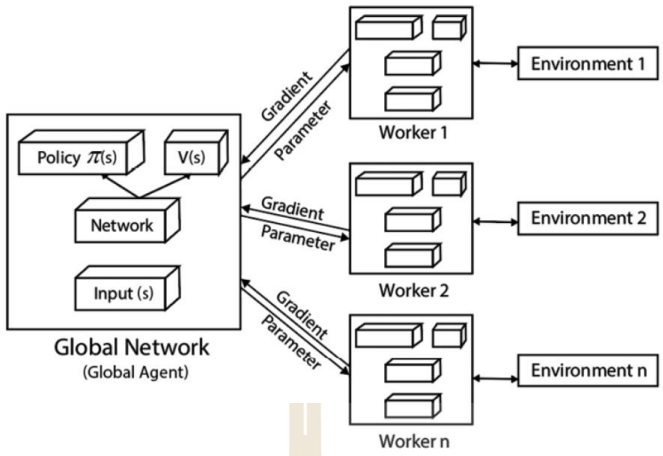
Actor network จะทำการหา optimal policy ด้วยวิธีการ policy gradient ส่วน critic network ใช้วิธีการหา state value $(V(s))$ ในการประเมิน policy ที่ได้ออกมา โดยวิธีการของ critic network จะช่วยลดความแปรปรวนที่เกิดขึ้นจากการอัพเดทค่าได้ การอัพเดทพารามิเตอร์ของโครงข่ายประสาทเทียมจะทำในทุก ๆ ขั้นของ episode ซึ่งจะมีความคล้ายคลึงกับวิธีการผลต่างชั่วคราว (temporal difference) ซึ่งจะใช้วิธีเดียวกันในการเปลี่ยนแปลงฟังก์ชันการคำนวณค่าใช้จ่าย (cost function) ของวิธีการ REINFORCE จนได้ดังสมการต่อไปนี้

$$\nabla_\theta J(\theta) = \nabla_\theta \log \pi_\theta(a_t|s_t)(r + \gamma V_\phi(s'_t) - V_\phi(s_t)) \tag{2.64}$$

เมื่อ $V_\phi(s_t)$ คือ state value ที่มาจากการทำนายในสภาวะที่เวลา t และเช่นเดียวกันกับการเรียนรู้โดยใช้ D-DQN นั้น actor critic สามารถใช้งานร่วมกับ advantage value ได้ ซึ่งอัลกอริทึมดังกล่าวจะถูกเรียกว่า Advanced Actor Critic (A2C) ซึ่งจะได้ว่า policy gradient ของ A2C จะสามารถแสดงได้ดังสมการต่อไปนี้

$$\nabla_\theta J(\theta) = \nabla_\theta \log \pi_\theta(a_t|s_t)A(s,a) \tag{2.65}$$

นอกจากวิธีการ A2C แล้ว อีกวิธีการที่มีการพัฒนามาจาก actor critic และเป็นที่นิยมเป็นอย่างมากคือวิธีการ Asynchronous Advantage Actor-Critic หรือ A3C โดยแนวคิดของ A3C คือ การสร้างตัวแทนหลายตัว (multiple agent) ที่สามารถเรียนรู้ไปพร้อม ๆกับแบบคู่ขนานได้ โดยจะทำการแยกตัวแทนออกเป็นหลายตัวเรียกว่า ตัวแทนทำงาน (worker agent) และมีตัวแทนที่ทำหน้าที่รวบรวมความรู้เรียกว่าตัวแทนกลาง (global agent) ดังสถาปัตยกรรมในรูปด้านล่างนี้



รูปที่ 2.35 สถาปัตยกรรมของ A3C

Loss function ของ A3C สามารถเขียนได้ดังนี้ โดยจะมีการเพิ่มพารามิเตอร์ของเอนโทรปี (entropy) เข้าไปด้วย

$$J(\theta) = \log \pi_\theta(a_t, s_t)\left(r + \gamma V_\phi(s_t') - V_\phi(s_t)\right) + \beta H(\pi(s)) \qquad (2.66)$$

เมื่อ β คือ พารามิเตอรสำหรับควบคุมนัยสำคัญของเอนโทรปี ส่วน $H(\pi)$ คือพจน์ของเอนโทรปีใน policy ที่เกิดขึ้น ซึ่งจะบ่งบอกถึงการสุ่ม (randomness) ของการเรียนรู้ หากเอนโทรปีมาก แสดงว่าตัวแทนมีการสุ่มการกระทำหลายแบบและมีการเรียนรู้ที่หลากหลายมากกว่ากรณีที่เอนโทรปีน้อย จากรูปที่ 2.35 เมื่อ worker agent ทำการคำนวณ gradient ที่เกิดขึ้นเรียบร้อยแล้ว ค่านั้นจะถูกส่งไปที่ global agent โดย gradient ของ worker agent แต่ละตัวจะเป็น asynchronous กัน จากนั้น global agent จะทำการอัพเดทพารามิเตอร์แล้วส่งไปให้ worker agent เพื่อทำการเรียนรู้วนไปจนได้ optimal policy ออกมา เมื่อ N. Ezebuugo (2020) ได้นำเสนอการทดสอบการประยุกต์โมเดล A3C ร่วมกับหุ่นยนต์เคลื่อนเทียบกับ DQN ปรากฏว่า return สะสมจะได้มากกว่าและใช้เวลาในการฝึกโมเดลที่น้อยกว่าด้วยเมื่อเทียบในสภาวะเดียวกัน

X. Jiaqi และคณะ (2019) นำอัลกอริทึม soft actor critic ไปประยุกต์กับ mobile robot สำหรับการทำการนำทางอัตโนมัติด้วย LiDAR โดยเปรียบเทียบกับวิธีดั้งเดิม คือ simultaneous localization and mapping (SLAM) ซึ่งผลการสร้างเส้นทางสำหรับการเดินทางได้ใกล้เคียงกัน แต่การใช้การเรียนรู้แบบเสริมกำลังไม่จำเป็นต้องสร้างแผนที่ไว้ก่อน

2.4.14 Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient (DDPG) เป็นอัลกอริทึมที่พัฒนามาจากวิธีการ actor-critic ซึ่งมี actor สำหรับหา optimal policy และ critic สำหรับให้ผลป้อนกลับเพื่อประเมินและปรับปรุง policy ให้กับตัว actor แต่ใน DDPG จะแตกต่างจาก A2C และ A3C โดยที่ตัว critic network ที่ใช้นั้น DDPG จะใช้ Deep Q Network ในการประเมิน policy นอกจากนี้ จากวิธีที่ policy gradient ใช้ในการหา optimal policy มักจะใช้ policy แบบ stochastic คือ การเลือกการกระทำของตัวแทนจะถูกแบ่งตามความน่าจะเป็น แต่ DDPG นั้นจะใช้ deterministic policy แทน ในการคำนวณ loss function นั้น สมการการคำนวณจะมีความคล้ายคลึงกันในส่วนของ critic network ดังสมการที่ 2.67 แต่ในงานที่การกระทำเป็น continuous action space นั้น เราไม่สามารถหา state-action value ของการกระทำที่เป็นไปได้ทั้งหมดไม่ได้ จึงต้องตัดเทอมของ a' ออกโดยการใช้ target network ใน actor network ซึ่งมีพารามิเตอร์ ϕ' ซึ่ง network ของ a' คือ $\mu_{\phi'}(s')$ ซึ่งจะทำให้เราได้ loss function ใหม่แสดงได้ดังสมการต่อไปนี้

$$J(\theta) = \frac{1}{K}\sum_i \left(r_i + \gamma Q_{\theta'}\left(s'_i, \mu_\phi(s'_i)\right) - Q_\theta(s_i, a_i)\right)^2 \tag{2.67}$$

ซึ่งในการจะอัพเดทตัวของ target critic network นั้นจะมีการคัดลอกพารามิเตอร์ θ ของ main critic network มา ซึ่งจะได้สมการการอัพเดทดังต่อไปนี้

$$\theta' = \theta\tau + (1-\tau)\theta' \tag{2.68}$$

โดย τ คือ ค่า soft replacement และมักถูกกำหนดให้เป็น 0.001

หลังจากกล่าวถึง critic network ไปแล้ว ต่อไปจะเป็นการกล่าวถึง actor network ซึ่งมีการใช้ deterministic policy มาในการหา optimal policy และจากการใช้วิธีการนี้ทำให้ตัวแทนจะเลือกการกระทำที่ดีที่สุดในสภาวะนั้นเพียงอย่างเดียวไม่เสาะหาการกระทำใหม่ ซึ่งเรียกเหตุการณ์นี้ว่า exploration – exploitation dilemma สำหรับวิธีการ DDPG แล้ว การจัดการกับการกระทำที่เป็น continuous action space ด้วย deterministic policy จำเป็นที่จะต้องเพิ่มการรบกวน หรือ noise $\mathcal{N}$ เขาไปใน action space ด้วยซึ่งกระบวนการนี้ถูกเรียกว่า กระบวนการ Ornstein- Uhlenbeck ซึ่งจะทำให้ได้สมการความสัมพันธ์ของ action ออกมาเป็นดังต่อไปนี้

$$a = \mu_\phi(s) + \mathcal{N} \tag{2.69}$$

เมื่อรวมเข้ากับหลักการของ DQN และ actor critic ก่อนหน้าจะได้สมการ loss function ของ DDPG ออกมาดังสมการที่ 2.70

$$J(\theta) = \frac{1}{K}\sum_i \left(y_i - Q_\theta(s_i, a_i)\right)^2 \tag{2.70}$$

เมื่อ

$$y_i = r_i + \gamma Q_{\theta'}(s'_i, \mu_{\phi'}(s'_i)) \tag{2.71}$$

และสมการในการอัพเดท actor network จะได้ออกมาเป็นสมการดังต่อไปนี้

$$\phi' = \tau\phi + (1-\tau)\phi' \tag{2.72}$$

R. Bartomeu และคณะ (2020) ประยุกต์ใช้ DDPG สำหรับการบินตามเส้นทาง (path following) ร่วมกับตัวควบคุมแบบพีไอดีโดยนำเสนอสมการของ noise ที่เพิ่มเข้าไปในการฝึกโมเดลดังสมการที่ (2.73)

$$n_k = n_{k-1} + \theta_n(\mu_n - n_{k-1})\Delta t + \sigma_n dW_t \tag{2.73}$$

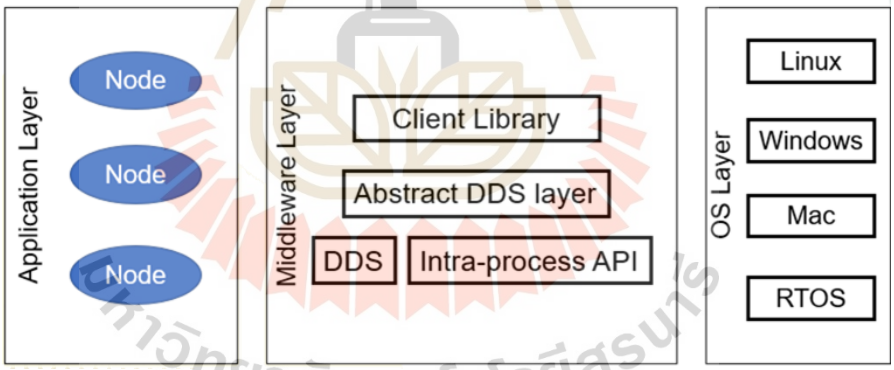
เมื่อ n_k คือค่าของ noise ที่รอบการคำนวณ k ส่วน θ_n คือพารามิเตอร์ที่เกี่ยวข้องอัตราเร็วของ mean conversion μ_n คือ drift term ที่ส่งผลต่อ asymptotic mean ค่า Δt คือค่าของ timestep และ dW_t คือค่ามาตรฐานกระบวนการเวียเนอร์ หลังจากทำการเรียนรู้และทดสอบการบินตามเส้นทางในโปรแกรมจำลองการบิน ปรากฏว่า DDPG ให้ผลใกล้เคียงกับเส้นทางที่กำหนดไว้ในตอนแรก

G. Ricardo และคณะ (2020) นำ DDPG ไปเปรียบเทียบกับวิธีการซอฟท์ actor critic ในการนำทางของอากาศยานสี่ใบพัดโดยใช้ข้อมูลจาก LiDAR ในการตรวจจับวัตถุ และทำการเปรียบเทียบกับ traditional control method ซึ่งพบว่า DDPG ให้ความคลาดเคลื่อนของเส้นทางมากกว่า soft actor critic แต่ไม่สามารถทำได้ดีกว่า traditional control ได้ในสภาวะที่โล่ง แต่เมื่อมีสิ่งกีดขวางตัว traditional control ไม่สามารถทำงานได้ เพราะชนสิ่งกีดขวางแต่ DDPG และ soft actor critic สามารถทำงานได้ โดยมีประสิทธิภาพตามลำดับ

2.5 ระบบปฏิบัติการหุ่นยนต์ (Robot Operating System : ROS)

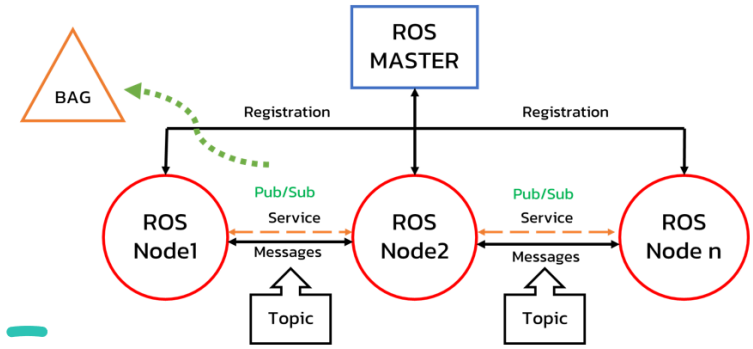
ระบบปฏิบัติการหุ่นยนต์ (Robot Operating System: ROS) ระบบปฏิบัติการหุ่นยนต์ (Robot Operating System : ROS) เป็น framework สำหรับการพัฒนาหุ่นยนต์ที่จะช่วยให้การพัฒนาหุ่นยนต์ที่มีความซับซ้อนมีความสะดวกมากยิ่งขึ้น มีเครื่องมือที่ช่วยในการวิเคราะห์ข้อมูลที่ได้จากหุ่นยนต์ให้ใช้มากมาย นอกจากนี้ ROS ยังสามารถนำไปใช้เพื่อพัฒนาหุ่นยนต์ที่มีคามฉลาดมากมาย เช่น หุ่นยนต์ที่สามารถนำทางตัวเองได้ (Self-Navigation robot) โดยใช้เทคนิค SLAM (Simultaneous Localization and Mapping) ในการสร้างแผนที่และระบุตำแหน่งเพื่อการนำทาง หรืออาจจะนำไปควบคุมแขนกล ทำหุ่นยนต์ที่มีการใช้งานร่วมกับปัญญาประดิษฐ์ต่าง ๆ ก็ได้

ในปัจจุบัน ROS มีการพัฒนามา 2 เวอรชั่น คือ ROS1 และ ROS2 โดย ROS2 เป็นเวอร์ชันใหม่ล่าสุดที่มีการพัฒนาขึ้นมา โดยเปลี่ยนสถาปัตยกรรมจาก ROS1 ให้สามารุนำไปใช้ในเชิงพาณิชย์ได้มากขึ้น ROS2 มีการใช้ Data Distributed Service (DDS) เป็นตัวกลางในการสื่อสารข้อมูล (middleware) ซึ่งสนับสนุนระบบปฏิบัติการเวลาจริง (Real Time Operating System : RTOS) การส่งข้อมูลของ ROS2 มีโมเดลการสื่อสารในรูปแบบของระบบ publish – subscribe ในการสื่อสารข้อมูลกัน โดยโครงสร้างชั้นการทำงานของ ROS2 แสดงดังในรูปที่ 2.36



รูปที่ 2.36 ชั้นของสถาปัตยกรรมของ ROS2 โดย DDS

หลักการทำงานเบื้องต้นของ ROS1 สามารแสดงได้ดังในรูปที่ 2.37 โดย ROS จะแบ่งหน่วยการทำงานของโปรแกรมออกเป็น Node ซึ่ง 1 โหนดจะรับหน้าที่แตกต่างกัน แต่สามารถส่งข้อมูลหากันได้ผ่าน message และ topic นอกจากนี้ยังมี service กับ action เพื่อช่วยอำนวยความสะดวกด้วย โดยใน ROS1 นั้น จะมี ROS Master คอยเป็นจุดลงทะเบียนของโหนดต่าง ๆ เพื่อจับคู่กัน แต่ใน ROS 2 นั้น การใช้ DDS ทำให้โหนดแต่ละโหนดสามารถค้นหากันได้เลยทันทีโดยไม่ต้องมี ROS Master อีกต่อไป

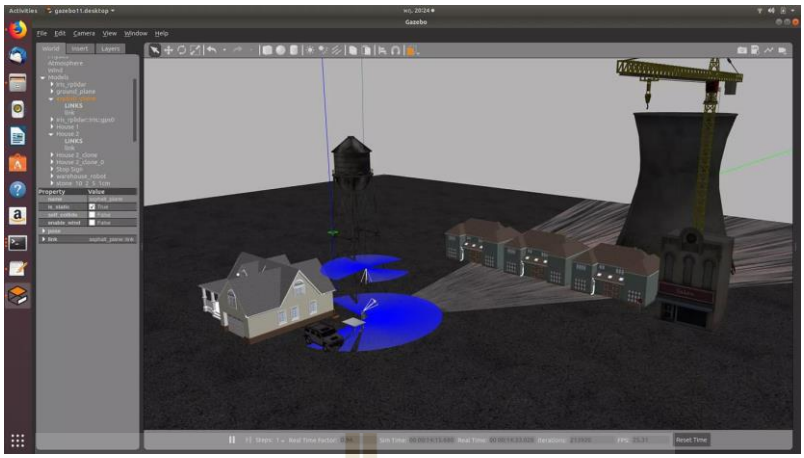


รูปที่ 2.37 การแบ่งหน่วยการทำงานเบื้องต้นของ ROS1

ROS2 ทำให้การพัฒนาหุ่นยนต์ที่ต้องอาศัยการทำงานแบบเวลาจริงทำได้ง่ายและมีประสิทธิภาพมากขึ้น โดย M. Yuya (2019) ได้ทำการทดสอบสมรรถนะของ ROS2 และพบว่าการใช้ ROS2 นั้นสามารถการันตีเวลาในการส่งข้อมูลถึงจุดหมายได้อย่างดีและมรการสูญหายข้องข้อมูลที่น้อยมาก ๆ เมื่อเปรียบเทียบกับการส่งข้อมูลด้วย ROS1 แต่ในการส่งข้อมูลที่มีขนาดใหญ่เกิด 256 กิโลไบต์ในการส่งข้อมูลหนึ่งครั้งนั้น จะใช้เวลามากกว่า ROS1 เพียงเล็กน้อยในหลักมิลลิวินาที นอกจากนี้ ROS2 ยังเพิ่มความน่าเชื่อถือให้กับการทำหุ่นยนต์ด้วยเนื่องจากมีระบบความปลอดภัยทางด้านเครือข่ายและการส่งข้อมูลดังเช่น C. Hang (2020) นำ ROS2 ไปประยุกต์ใช้ในการทำ autonomous driving car ที่มีสเถียรภาพของข้อมูลที่สูงและใช้ในการควบคุมแบบเวลาจริงได้อย่างดี มี latency ที่ต่ำ ซึ่งในวิทยานิพนธ์ฉบับนี้ก็จะมีการนำ ROS2 มาประยุกต์ด้วยเช่นกัน

2.6 ระบบจำลองการบิน (Flight simulator) ด้วย Gazebo

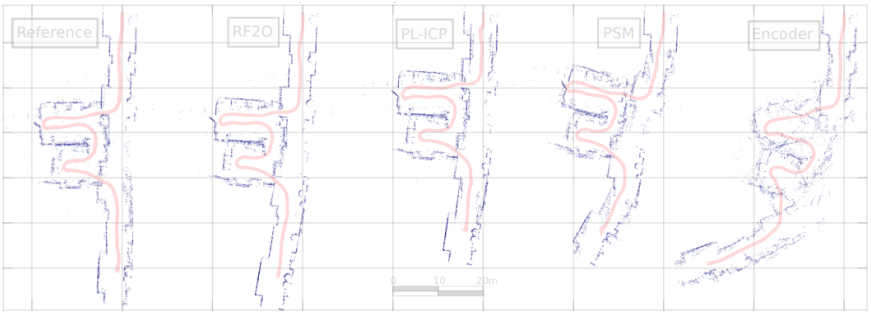
Gazebo เป็นโปรแกรม 3D Dynamic simulator ซึ่งมีความสามารถในการจำลองฟิสิกส์ความเป็นไปได้ต่าง ๆ ของ หุ่นยนต์ ที่มีความซับซ้อน ทั้งสภาพแวดล้อมในและนอกอาคาร นอกจากนี้ยังสามารถเพิ่มปลั๊กอินของเซนเซอร์ต่าง ๆ เข้าไปได้ด้วย เพื่อทำการจำลองการทำงานของหุ่นยนต์ได้อย่างมีประสิทธิภาพ ทำให้สามารถทดสอบการทำงานของอัลกอริทึม, ออกแบบหุ่นยนต์เป็นไปได้อย่างง่ายดาย โดยโปรแกรม Gazebo นั้นสามารถเชื่อมต่อกับซอฟท์แวร์ได้หลากหลาย และยังสามารถใช้งานร่วมกับระบบปฏิบัติการหุ่นยนต์ (Robot Operating System: ROS) ได้อีกด้วย โดยตัวอย่างหน้าตาของโปรแกรม Gazebo เป็นดังในรูปต่อไปนี้



รูปที่ 2.38 ตัวอย่างโปรแกรม Gazebo

2.7 การหาตำแหน่งด้วยอัลกอริทึม RF2O

อัลกอริทึม RF2O หรือ (Range Flow To Odometry) เป็นอัลกอริทึมสำหรับการหาโอโดเมทรี (odometry) ในระราบ 2 มิติตามแนวแกน x และแกน y ที่มีความแม่นยำโดยใช้ข้อมูลจากตัวสแกนเลเซอร์ที่มีการยิงรัศมีไปรอบ ๆ อัลกอริทึมนี้ถูกเสนอโดย J. Mariano ในปี 2016 [29] อัลกอริทึม RF2O จะทำการนำข้อมูลของเลเซอร์รอบทิศทางมาหาการเคลื่อนที่สัมพัทธ์ของจุดแต่ละจุดเพื่อประมาณความเร็วที่เปลี่ยนแปลงไปแล้วเอาความเร็วดังกล่าวทั้งความเร็วเชิงเส้นและความเร็วเชิงมุม มาคำนวณเป็นตำแหน่งที่มีการเปลี่ยนแปลงไปของตัวเซนเซอร์ โดยในงานวิจัยนี้ การหาตำแหน่งของอากาศยานทำได้ยาก เพราะไม่สามารถติดเซนเซอร์บางตัวเหมือนหุ่นยนต์บนพื้น เช่น เอ็นโคเดอร์ได้และยังต้องทำงานโดยที่ไม่มีแผนที่ ดังนั้น จึงไม่สามารถใช้วิธีการทำ SLAM (Simultaneous Localization and Mapping) ได้ อัลกอริทึมนี้จึงมีความเหมาะสมที่สุด โดยคุณ J. Mariano ได้มีการเปรียบเทียบความแม่นยำของอัลกอริทึมนี้เอาไว้กับค่าโอโดเมทรีที่ได้จากเอนโคเดอร์ในหุ่นยนต์เคลื่อนที่ พบว่าอัลกอริทึม RF2O สามารถให้ตำแหน่งที่แม่นยำกว่าได้ดังในรูปต่อไปนี้



รูปที่ 2.39 การเปรียบเทียบแผนที่ที่สร้างจากตำแหน่งของ RF2O และอัลกอริทึมอื่น (J. Mariano, 2016)

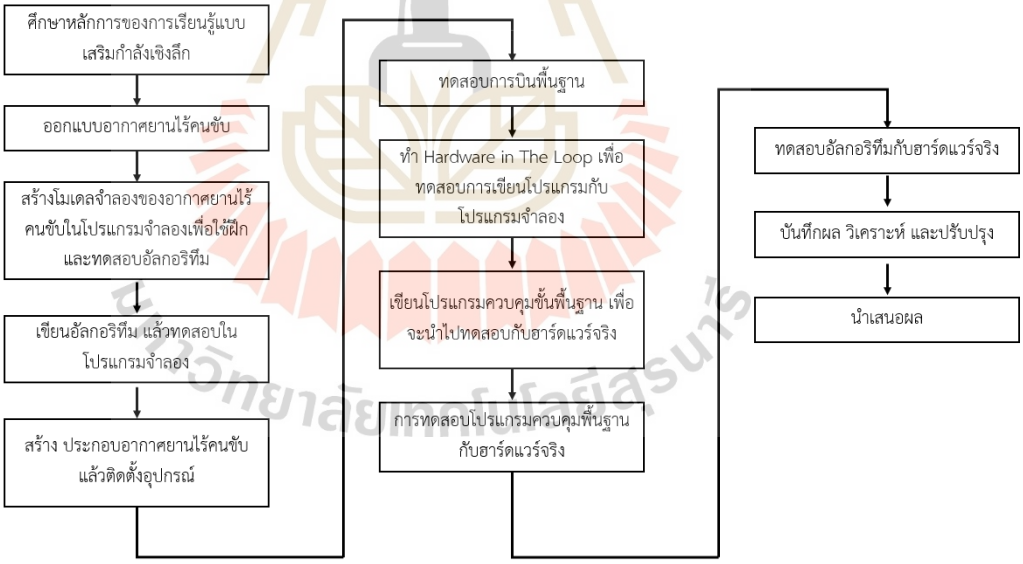
2.8 การรวมข้อมูลของเซนเซอร์สำหรับการประมาณสภาวะ

การรวมข้อมูลของเซนเซอร์ (Sensor Fusion) สำหรับการประมาณสภาวะ (State Estimation) ของหุ่นยนต์เป็นกระบวนการที่จำเป็นสำหรับการพัฒนาหุ่นยนต์ทุกประเภทไม่เว้นแม้แต่อากาศยานไร้คนขับ สำหรับงานวิจัยนี้ กระบวนการประมาณสภาวะจะใช้สำหรับคาดเดาตำแหน่งและท่าทางของอากาศยาน โดยใช้ข้อมูลตำแหน่งจาก RF2O และข้อมูลจาก IMU โดยใช้ตัวกรองคาลมานแบบขยาย (Extended Kalman Filter : EKF) [30] ซึ่งเป็นตัวกรองสำหรับระบบที่ไม่เป็นเชิงเส้น (non-linear) โดยมีขั้นตอนของการทำนาย (prediction step) และขั้นตอนของการอัพเดทค่า (update step) ทำงานประสานกัน โดยในขั้นตอนของการทำนายนั้น ตัวกรองคาลมานแบบขยายจะใช้ข้อมูลที่มีอยู่ของหุ่นยนต์เทียบกับเวลามาทำนายสภาวะปัจจุบันของหุ่นยนต์ก่อน จากนั้นจะทำการอัพเดทสภาวะที่ถูกทำนายโดยการดึงข้อมูลจากเซนเซอร์เข้ามาแล้วทำการปรับค่าเกนของตัวกรองคาลมาน โดยในระหว่างกระบวนการจะมีการคำนวณสิ่งรบกวน (noise) และเมทริกซ์ความเกี่ยวข้องของความแปรปรวน (covariance matrix) และทำการอัพเดทตลอดเวลา เพื่อชดเชยสิ่งรบกวนจากการวัดของเซนเซอร์และสิ่งรบกวนในกระบวนการ ให้การประมาณสภาวะมีค่าความแม่นยำมากที่สุด

บทที่ 3
วิธีการดำเนินการ

3.1 กล่าวนำ

ในหัวข้อวิธีการดำเนินการวิจัยของวิทยานิพนธ์นี้ได้นำเสนอหัวข้อวิธีดำเนินงาน โดยมีรายละเอียด ได้แก่ รายละเอียดการออกแบบของอากาศยานไร้คนขับ การตั้งค่าโปรแกรมควบคุมการบิน การสร้างสภาพแวดล้อมจำลองสำหรับการเขียนโปรแกรมควบคุมอากาศยานไร้คนขับ การใช้ระบบปฏิบัติการหุ่นยนต์ในการเขียนโปรแกรมควบคุมอากาศยานทั้งในโปรแกรมจำลองและในฮาร์ดแวร์จริง แนวทางการทดสอบการบินและในการระบุตำแหน่งในอาคาร การฝึกฝนโมเดลการเรียนรู้แบบเสริมกำลังเชิงลึกและการนำโมเดลที่ได้จากการฝึกไปใช้จริง โดยขั้นตอนของการดำเนินงานแสดงดังในรูปต่อไปนี้



รูปที่ 3.1 ขั้นตอนการดำเนินงานวิจัย

3.2 การออกแบบขนาดของอากาศยานไร้คนขับ

การออกแบบอากาศยานไร้คนขับที่ใช้ในการทดสอบในวิทยานิพนธ์ฉบับนี้ เป็นอากาศยาน 4 ใบพัดขนาดเล็ก โดยมีการตั้งความต้องการในการออกแบบตามขอบเขตของงานวิจัย โดยตารางสรุปข้อมูลความต้องการในการออกแบบเป็นดังต่อไปนี้

ตารางที่ 3.1 ความต้องการพื้นฐานของอากาศยานไร้คนขับสำหรับการออกแบบชุดควบคุม

หัวข้อ	**ค่า**	**หน่วย**
จำนวนใบพัด	4	ใบพัด
รัศมีใบพัด	< 12	นิ้ว
น้ำหนักรวมของอากาศยาน	< 2.5	กิโลกรัม
ระยะเวลาทำการบิน	10	นาที
เพดานบิน	10	เมตร
น้ำหนักบรรทุก	< 1	กิโลกรัม
อุปกรณ์ที่ต้องทำการติดตั้ง	1. ชุดบอร์ดควบคุมการบิน 2. บอร์ด Nvidia Jetson Nano 3. YDLiDAR TG30	-

เมื่อทำการกำหนดความต้องการในการออกแบบดังตารางที่ 3.1 แล้ว จะทำการประเมินน้ำหนักของอากาศยานทั้งหมดโดยใช้สมการที่ 2.3 ถึง 2.12 โดยใช้โปรแกรมภาษา Python ในการคำนวณ ซึ่งจะได้น้ำหนักออกมาที่ ประมาณ 1948.85 กรัม จากนั้นนำน้ำหนักที่คำนวณได้ไปเลือกมอเตอร์และใบพัดตามสมการที่ 2.1 และ 2.2 จะได้ขนาดรัศมีของอากาศยานออกมาเป็น 0.225 เมตร แล้วจึงนำข้อมูลทั้งหมดไปเลือกมอเตอร์ที่จะใช้ ซึ่งต้องมีแรงขับต่อมอเตอร์ในช่วง hover ของอากาศยานได้เป็น 487.22 กรัม ที่เปอร์เซ็นต์คันเร่ง 50 เปอร์เซนต์ โดยมอเตอร์ที่พิจารณาเลือกมาใช้ในโครงการนี้คือ Motor Emax MT2213 935KV จากนั้นได้นำข้อมูลของมอเตอร์ไปเลือกอุปกรณ์ควบคุมความเร็ว (Electronic Speed Controller : ESC) และแบตเตอรี่จากนั้น ใช้เครื่องมือคำนวณ ecalC ในการคำนวณเปรียบเทียบสมรรถนะย้อนกลับ โดยได้ผลลัพธ์ดังในรูปต่อไปนี้

Battery		Total Drive		Multicopter	
Load:	10.77 C	Drive Weight:	1157 g	All-up Weight:	1949 g
Voltage:	14.21 V		40.8 oz		68.7 oz
Rated Voltage:	14.80 V	Thrust-Weight:	2.0 : 1	add. Payload:	1406 g
Energy:	88.8 Wh	Current @ Hover:	21.85 A		49.6 oz
Total Capacity:	6000 mAh	P(in) @ Hover:	323.3 W	max Tilt:	54 °
Used Capacity:	5100 mAh	P(out) @ Hover:	241.3 W	max. Speed:	64 km/h
min. Flight Time:	4.7 min	Efficiency @ Hover:	74.6 %		39.8 mph
Mixed Flight Time:	10.8 min	Current @ max:	64.60 A	est. Range:	- m
Hover Flight Time:	14.0 min	P(in) @ max:	956.1 W		- mi
Weight:	652 g	P(out) @ max:	683.9 W	est. rate of climb:	7.1 m/s
	23 oz	Efficiency @ max:	71.5 %		1398 ft/min
				Total Disc Area:	20.27 dm²
					314.19 in²
				with Rotor fail:	✗

รูปที่ 3.2 ผลลัพธ์ที่ได้จากการคำนวณในโปรแกรม eCalc

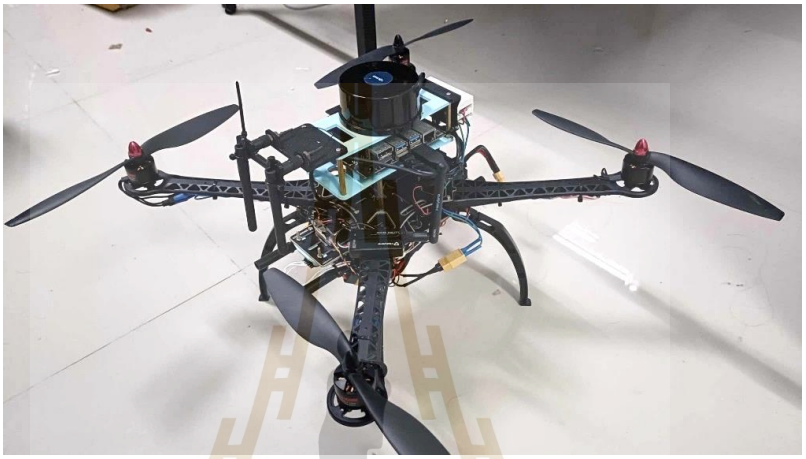
จากการคำนวณ สามารถสรุปรายละเอียดของการออกแบบในขั้นต้น (conceptual design) ได้ดังต่อไปนี้

ตารางที่ 3.2 รายละเอียดของอากาศยานจากการออกแบบขั้นต้น

ตัวแปร	ค่า	หน่วย
จำนวนใบพัด	4	ใบพัด
รัศมีใบพัด	10	นิ้ว
น้ำหนักรวมสูงสุดของอากาศยาน	1.95	กิโลกรัม
ระยะเวลาทำการบินสูงสุด	10.8	นาที
เพดานบิน	10	เมตร
น้ำหนักบรรทุก	0.4	กิโลกรัม
ขนาดเฟรม	450	มิลลิเมตร
ขนาด ESC	40	แอมแปร์
มอเตอร์	Emax MT2212	-
แบตเตอรี่	LiPo 14.8V 4S	-
ความเร็วสูงสุด	17.78	เมตรต่อวินาที

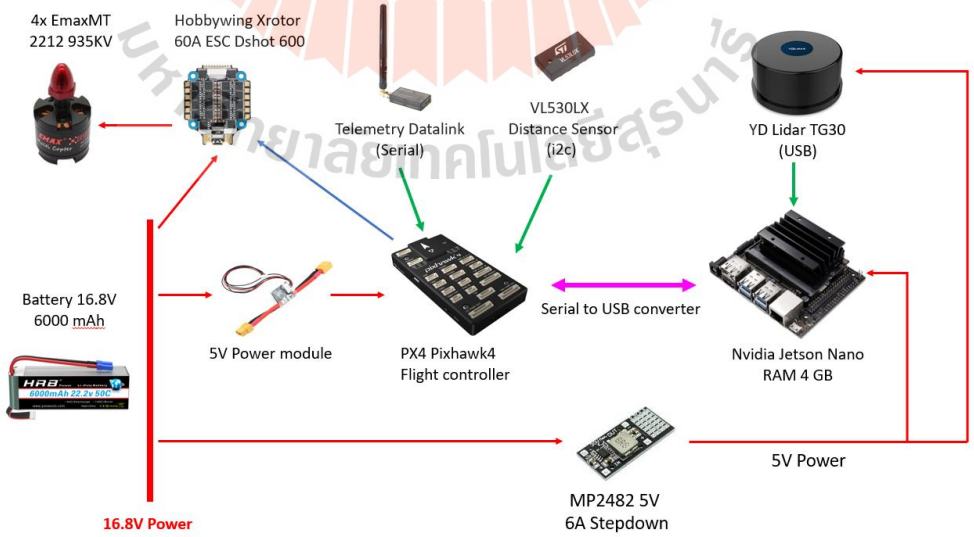
3.3 การประกอบอากาศยานไร้คนขับ

แบบที่ได้จากการออกแบบด้วยโปรแกรม 3 มิติ จะถูกนำไปผลิตและขึ้นรูปชิ้นส่วนด้วยวัสดุที่เหมาะสม แผ่นคาร์บอนคอมโพสิท พลาสติก PLA และพลาสติกไนลอน อื่น ๆ หลังจากประกอบชิ้นส่วนพื้นฐานของอากาศยานไร้คนขับเรียบร้อยแล้ว จะได้รูปทรงของอากาศยานไร้คนขับที่ใช้ในงานวิจัย ดังรูปต่อไปนี้ โดยเมื่อประกอบจริงอากาศยานไร้คนขับมีน้ำหนักรวมอยู่ที่ 1883 กรัม



รูปที่ 3.3 อากาศยานไร้คนขับที่ใช้ในงานวิจัย

โดยแผนผังการเชื่อมโยงส่วนประกอบต่าง ๆ ของอากาศยานไร้คนขับจะแสดงดังในรูปต่อไปนี้



รูปที่ 3.4 แผนผังแสดงการเชื่อมต่ออุปกรณ์ของอากาศยานไร้คนขับ

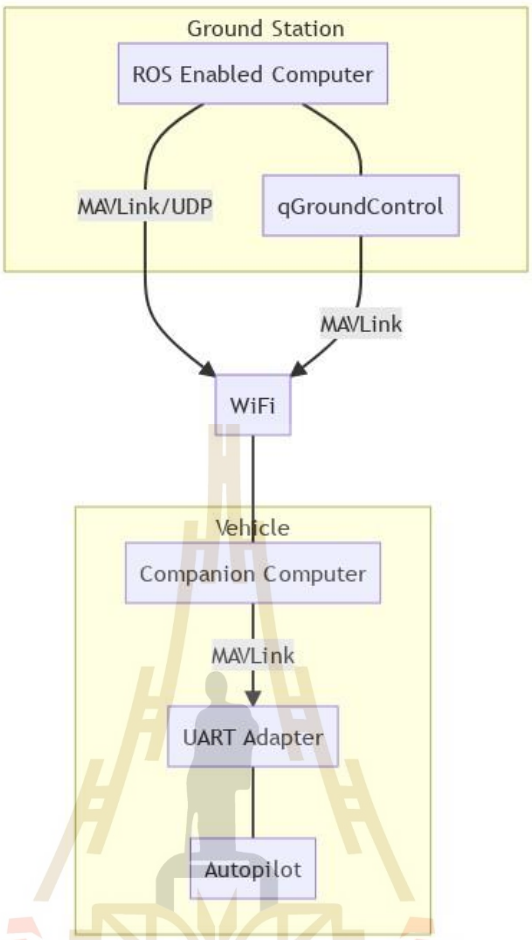
3.4 การตั้งค่าโปรแกรมควบคุมการบิน

โปรแกรมควบคุมการบินเป็นโปรแกรมที่ถูกติดตั้งในบอร์ดควบคุมการบิน (Flight Controller Unit : FCU) สำหรับทำการควบคุมการบิน โดยโปรแกรมควบคุมการบินที่เลือกใช้เป็นโปรแกรมควบคุมการบินแบบเปิด (Open Source) ที่มีชื่อว่า พีเอ็กซ์โฟว์ (PX4) ซึ่งรองรับการใช้งานกับบอร์ด pixhawk4 ที่มีการเลือกใช้ในงานวิจัยครั้งนี้ เมื่อติดตั้งฮาร์ดแวร์และโปรแกรมเรียบร้อยแล้วก่อนที่จะทำการบินได้ ต้องมีการตั้งค่าโปรแกรมควบคุมการบินก่อน เช่น การปรับเทียบขอบเขตค่าข้อมูลของรีโมทควบคุม (radio control calibration), การปรับเทียบเซนเซอร์, แบตเตอรี่, โหมดการบิน และ อื่น ๆ โดยมีโปรแกรม Q Ground Control เป็นส่วนควบคุมภาคพื้น (Ground Control Station : GCS) ใช้สำหรับรับ-ส่งสัญญาณกับอากาศยานและปรับแต่งค่าต่าง ๆ

รูปที่ 3.5 ตัวอย่างหน้าสรุปข้อมูลการปรับแต่งโปรแกรมการบิน

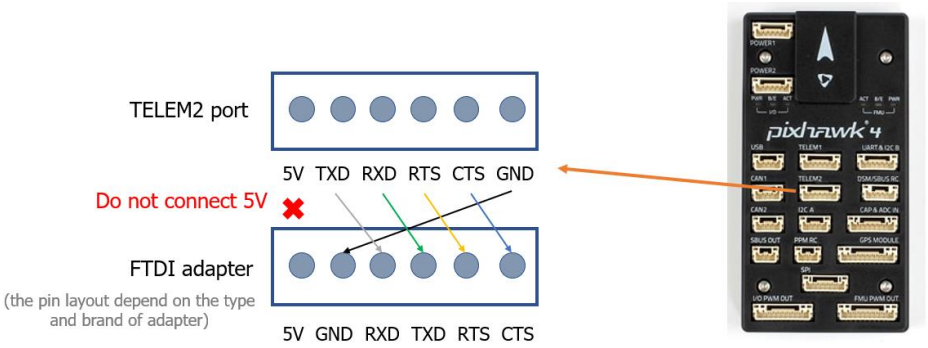
3.5 การเชื่อมต่อระบบปฏิบัติการหุ่นยนต์เข้ากับระบบควบคุมการบิน

การเชื่อมต่อระบบปฏิบัติการหุ่นยนต์เข้ากับระบบควบคุมการบิน จะช่วยอำนวยความสะดวกในการเขียนโปรแกรมควบคุมอุปกรณ์การบินโดยใช้คอมพิวเตอร์ขนาดเล็ก (companion computer) ให้สะดวกมากขึ้น และจะเพิ่มขีดความสามารถของอากาศยานด้วยโดยเฉพาะการเพิ่มระบบประมวลผลภาพ และปัญญาประดิษฐ์เข้าไปให้อากาศยานไร้คนขับ แผนผังการเชื่อมต่อบอร์ดคอมพิวเตอร์เข้าไปทำงานร่วมกับบอร์ดควบคุมการบินเป็นดังในรูปต่อไปนี้



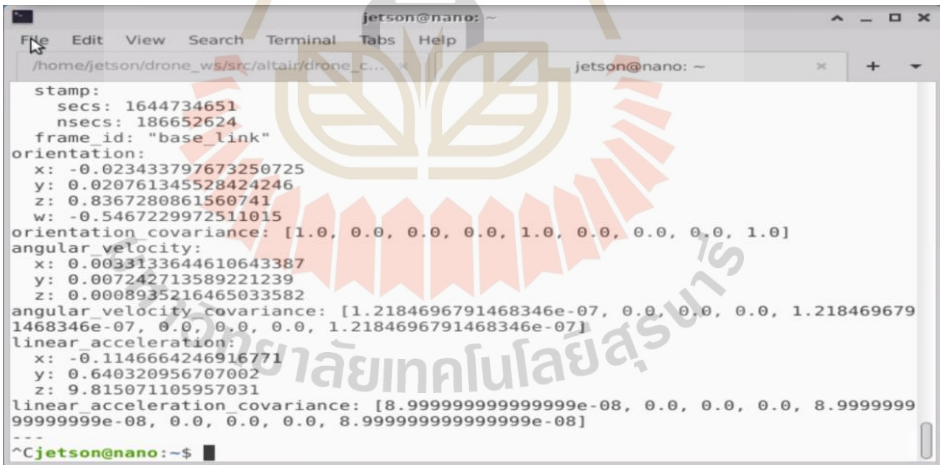
รูปที่ 3.6 แผนผังการเชื่อมต่อบอร์ดคอมพิวเตอร์เข้ากับบอร์ดควบคุมการบิน

บอร์ดควบคุมการบินอัตโนมัติ (Autopilot) จะถูกเชื่อมต่อกับบอร์ดคอมพิวเตอร์ Nvidia Jetson Nano ด้วย FTDI Adapter ด้วยการเชื่อมต่อแบบ UART ดังในรูปที่ 3.16 โดยการสื่อสารข้อมูลระหว่างบอร์ดทั้งสองจะส่งผ่านด้วยโปรโตคอลการส่งข้อมูลสำหรับอากาศยานไร้คนขับขนาดเล็ก (Micro Aerial Vehicle Link : MAVLINK) บนบอร์ด Nvidia Jetson Nano จะมีการติดตั้งระบบปฏิบัติการหุ่นยนต์ (Robot Operating System : ROS) ไว้ด้วย และมีการติดตั้งแพ็คเกจ MAVROS ซึ่งเป็นแพ็คเกจสำหรับการส่งผ่านข้อมูลผ่าน MAVLINK มาเป็นการสื่อสารโดยใช้ message ของ ROS ได้



รูปที่ 3.7 การเชื่อมต่อบอร์ดคอมพิวเตอร์กับบอร์ดควบคุมการบิน

เมื่อทำการเชื่อมต่อบอร์ดทั้งสองเข้ากันได้แล้ว จะสามารถส่งข้อมูลระหว่างกันผ่าน ROS message ได้ ดังในตัวอย่างรูปที่ 3.16 ซึ่งเป็นตัวอย่างการดึงข้อมูล IMU จากบอร์ดควบคุมการบิน นอกจากนี้หากเครื่องคอมพิวเตอร์ควบคุมภาคพื้นมีระบบปฏิบัติการหุ่นยนต์ติดตั้งอยู่ด้วย ก็จะสามารถส่งข้อมูลผ่านผ่าน ROS message ระหว่างคอมพิวเตอร์บนอากาศยานและสถานควบคุมภาคพื้นผ่าน WiFi ได้ด้วย



รูปที่ 3.8 ตัวอย่างการดึงข้อมูล IMU จากบอร์ดควบคุมการบินผ่าน ROS message

ในส่วนของการส่งโปรแกรมควบคุมหรือคำสั่งไปให้อากาศยานไร้คนขับปฏิบัติตามสามารถทำได้แต่ต้องให้อากาศยานเข้าโหมดการบินแบบ OFFBOARD ก่อนถึงจะอนุญาตให้ companion computer สามารถส่งคำส่งเข้าไปควบคุมได้ โดยในงานวิจัยนี้ คำสั่งความเร็วจากการคำนวณของ

โมเดลการเรียนรู้แบบเสริมกำลังเชิงลึกจะถูกส่งไปให้กับบอร์ดควบคุมการบินด้วยโหมด OFFBOARD และโปรโตคอล MAVLINK และระบบปฏิบัติการหุ่นยนต์

3.6 การสร้างโมเดลจำลองของอากาศยานไร้คนขับในโปรแกรม Gazebo

การสร้างโมเดลจำลองของอากาศยานไร้คนขับในโปรแกรมจำลองเป็นสิ่งที่สำคัญยิ่งในงานวิจัยชิ้นนี้ เพราะต้องใช้การทำ software in the loop (SITL) ในการเขียนโปรแกรมเพื่อฝึกโมเดลให้เกิดการเรียนรู้เสริมกำลังเชิงลึก ทดสอบการเขียนโปรแกรมที่ใช้ในการควบคุมและโปรแกรมอื่น ๆ ก่อนนำไปทดสอบจริง โดยโปรแกรมจำลองที่เลือกใช้นั้น คือ โปรแกรม Gazebo ที่อาศัยระบบปฏิบัติการหุ่นยนต์ในการเชื่อมต่อโค้ดโปรแกรมควบคุมการบินของ PX4 เข้ากับระบบจำลองการบินได้ โมเดลที่ถูกนำเข้าไปในแบบจำลองจะถูกสร้างขึ้นมาในรูปของไฟล์ URDF (Unified Robot Description Format) และถูกติดตั้งปลั๊กอินของเซนเซอร์และอากาศพลศาสตร์เข้าไป นอกจากนี้จะมีการจำลองสภาพแวดล้อมสำหรับการฝึกฝนโมเดลและทดสอบการทำงานของโปรแกรมก่อนนำไปใส่อุปกรณ์จริงอีกด้วย

3.7 การทดสอบการบินพื้นฐานในอุปกรณ์จริง

การทดสอบการบินพื้นฐานในอุปกรณ์จริงเป็นขั้นตอนการทดสอบและปรับแต่งสเถียรภาพการบินของอากาศยานไร้คนขับ ในโหมดพื้นฐานได้แก่ โหมดเสถียรภาพ (stabilized flight mode), โหมดควบคุมความสูง (altitude flight mode) และโหมดควบคุมตำแหน่ง (position flight mode) และทำการปรับแต่งพารามิเตอร์ต่าง ๆ เช่น ค่าเกนตัวควบคุมพีไอดี (PID gain) การปรับทริมต่าง ๆ เป็นต้น จนกระทั่งการทำงานของอากาศยานไร้คนขับเป็นไปตามที่ต้องการ

3.8 การทดสอบโปรแกรมในโหมดออฟบอร์ด

การทดสอบโปรแกรมในโหมดออฟบอร์ดเป็นการทดสอบความถูกต้องของการทำงานของอากาศยานไม่ว่า ว่ามีความปลอดภัยและทำงานถูกต้องตามที่ต้องการหรือไม่ โดยการทดสอบจะถูกแบ่งออกเป็นแบบนอกอาคาร (outdoor) และแบบในอาคาร (indoor) โดยการทดสอบนอกอาคารจะใช้ระบบระบุตำแหน่งแบบจีพีเอสในการคำนวณตำแหน่งพื้นถิ่น (local position) ส่วนการทดสอบภายในอาคารจำใช้ข้อมูลของเลเซอร์ในการระบุตำแหน่งหลัก โดยการทดสอบโปรแกรมในโหมดออฟบอร์ดจะมีขั้นตอนการตรวจสอบดังตารางต่อไปนี้

ตารางที่ 3.3 รายละเอียดการตรวจสอบการทำงานในโหมดออฟบอร์ด

ลำดับ	รายละเอียดการตรวจสอบ	ผลที่คาดหมาย / ตัวชี้วัด
1	นำอากาศยานขึ้นบิน แล้วปรับโหมดการบินเป็นโหมดการรักษาตำแหน่ง	อากาศยานต้องสามารถรักษาตำแหน่งของตนเองได้
2	รันโปรแกรมสำหรับการส่งตำแหน่งในโหมดออฟบอร์ดให้กับอากาศยาน	อากาศยานต้องเปลี่ยนโหมดการบินเป็นโหมดออฟบอร์ด
3	ส่งค่าตำแหน่งที่ต้องการให้อากาศยานเคลื่อนที่ไปผ่านโหมดออฟบอร์ด	อากาศยานต้องสามารถเคลื่อนที่ไปตำแหน่งที่ต้องการได้ ตามความเร็วการเคลื่อนที่ที่ถูกกำหนดเอาไว้ (0.2 เมตรต่อวินาที) โดยที่ไม่เสียอาการ
4	ทดสอบหยุดการรันโปรแกรมสำหรับส่งคำสั่งในโหมดออฟบอร์ด	โหมด failsafe ต้องทำงานโดยการเปลี่ยนจากโหมดออฟบอร์ดเป็นโหมดรักษาตำแหน่ง โดยที่ไม่สูญเสียความสูงและสเถียรภาพการบิน
5	นำอากาศยานลงจอด แล้วนำไฟล์บันทึกการบินมาตรวจสอบความถูกต้องของการเปลี่ยนโหมดและความถูกต้องของการควบคุมตำแหน่งการเคลื่อนที่	ข้อมูลที่ถูกบันทึกในไฟล์บันทึกการบินต้องมีความถูกต้อง

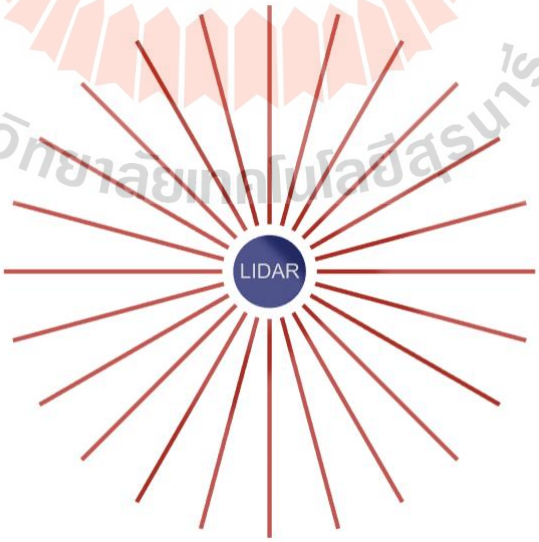
ภาพตัวอย่างการทดสอบการทำงานในโหมดออฟบอร์ดของอากาศยานแสดงดังในรูปต่อไปนี้

รูปที่ 3.9 ลักษณะการทดสอบการทำงานของโหมดออฟบอร์ดนอกอาคาร

รูปที่ 3.10 ลักษณะการทดสอบการทำงานของโหมดออฟบอร์ดในอาคาร

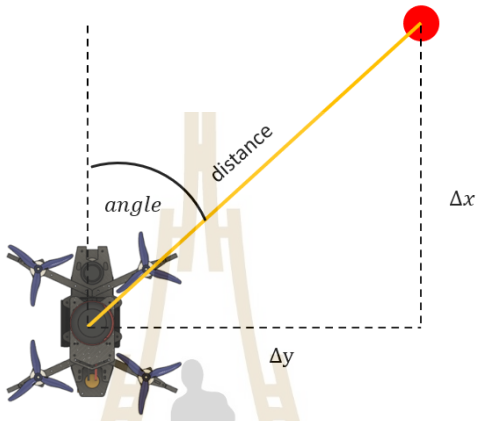
3.9 การออกแบบการรับรู้สภาวะของโมเดลการเรียนรู้เชิงลึก

สำหรับการฝึกโมเดลการเรียนรู้เชิงลึกของงานวิจัยนี้ตามกระบวนการตัดสินใจของมาร์คอฟจะต้องมีการสร้างการรับรู้สภาวะ (S) ให้กับโมเดล เพื่อให้โมเดลสามารถรับรู้สภาวะปัจจุบันและสภาวะที่เปลี่ยนไปของตัวแทนได้ โดยในงานวิจัยนี้ เครื่องมือที่ใช้ในการรับรู้สภาวะ (state) ของตัวแทนที่เลือกใช้คือ LiDAR (Light Detection and Ranging) ซึ่งจะทำการหยิบตำแหน่งของการตรวจจับออกมา 24 ตำแหน่ง โดยแบ่งมุมในการตรวจจับเป็น 15 องศาเท่า ๆ กัน ดังรูปต่อไปนี้



รูปที่ 3.11 การรับรู้สภาวะด้วย LiDAR

นอกจากการรับรู้สภาวะด้วย LiDAR แล้ว (S_{laser}) ยังต้อมีการรับรู้สภาวะด้วยข้อมูลของตำแหน่งของอากาศยานเทียบกับเป้าหมายด้วย โดยจะคิดเป็นระยะทางระหว่างอากาศยานและเป้าหมาย (S_d) รวมถึงมุมของด้านหน้าของโดรนกับจุดเป้าหมาย(S_h) จะทำให้ได้ข้อมูลสภาวะอีก 2 สภาวะ รวมทั้งหมดกับสภาวะที่รับรู้ได้จาก LiDAR เป็นทั้งหมด 26สภาวะ



รูปที่ 3.12 การรับรู้สภาวะด้วย LiDAR

3.10 การออกแบบปริภูมิกระทำของตัวแทน

สำหรับการออกแบบการกระทำของตัวแทน (action space) หรือ a_t เป็นการออกแบบสิ่งที่ตัวแทนสามารถกระทำได้หลังจากที่มีการรับรู้สภาวะแวดล้อมมาแล้ว เพื่อทำการเปลี่ยนเป็นสภาวะแวดล้อมถัดไป โดยการกำหนดปริภูมิการกระทำนั้น จะต้องคำนึงถึงผลลัพธ์ที่ต้องการของอากาศยาน ตัวแทนสามารถกระทำได้หลังจากที่มีการรับรู้สภาวะแวดล้อมมาแล้ว เพื่อทำการเปลี่ยนเป็นสภาวะแวดล้อมถัดไป โดยการกำหนดปริภูมิการกระทำนั้น จะต้องคำนึงถึงผลลัพธ์ที่ต้องการของอากาศยาน ในงานวิจัยนี้ผู้วิจัยได้มีการทดลองใช้ปริภูมิการกระทำ 2 แบบเพื่อหาปริภูมิการกระทำที่เหมาะสมที่สุด โดยมีปริภูมิที่สอดคล้องกับอัลกอริทึม DDPG 2 แบบดังต่อไปนี้

1) การกำหนดปริภูมิการกระทำเป็นความเร็วเชิงเส้นและความเร็วเชิงมุมของอากาศยาน

2) การกำหนดปริภูมิการกระทำเป็นตำแหน่งเป้าหมายที่เปลี่ยนไปในระนาบ 2 มิติของอากาศยาน

ซึ่งในผลการทดสอบเบื้องต้นที่พบนั้น การกำหนดแบบที่ 1) จะทำให้การกระทำที่ออกมามีความแข็งกร้าวเกินไปและต้องอาศัยความถี่ในการประมวลผลรวมทั้งทรัพยากรในการประมวลผลที่สูง

จนเกินไปจึงไม่เหมาะสมกับการนำไปใช้งานในอากาศยานไร้คนขับ ในงานวิจัยนี้จึงเลือกการกำหนดปริภูมิการกระทำเป็นแบบที่ 2) ซึ่งมีรายละเอียดดังต่อไปนี้

การกระทำของตัวแทนถูกกำหนดไว้เป็น 2 การกระทำได้แก่ การเปลี่ยนตำแหน่งการเคลื่อนที่ในแกน X และการเปลี่ยนตำแหน่งการเคลื่อนที่ในแกน Y เทียบกับเฟรมอ้างอิงของอากาศยานเอง โดยการเปลี่ยนแปลงการเคลื่อนที่จะสามารถหาได้จากเอาท์พุทการคำนวณของตัวโมเดลที่มีค่าเป็นแบบต่อเนื่อง โดยจะถูกจำกัดค่าเอาไว้ให้ไม่เกิน 1 เมตรต่อการเปลี่ยนแปลงการเคลื่อนที่ 1 ครั้ง และในการฝึกฝนโมเดลก็จะมีการใส่สิ่งรบกวน (noise) เข้าไปในการกระทำอีกด้วย โดยสามารถเขียนปริภูมิการกระทำดังสมการต่อไปนี้

$$a_t = [\,\Delta X\,, \Delta Y] \tag{3.1}$$

$$|\Delta X| \leq 1.0\ \ , |\Delta Y| \leq 1.0 \tag{3.2}$$

นอกจากพฤติกรรมที่เหมาะสมของอากาศยานและการลดความถี่ในการคำนวณของโมเดลแล้ว การเลือกใช้ปริภูมิแบบดังกล่าวจะทำให้โมเดลการเรียนรู้ไม่ขึ้นอยู่กับลักษณะการเคลื่อนที่ของตัวแทน ดังนั้น การฝึกโมเดล 1 รอบจะสามารถนำโมเดลที่ฝึกไปใช้กับหุ่นยนต์ประเภทใดก็ได้ ต่างจากการระบุปริภูมิเป็นแบบอื่น และยังช่วยให้การควบคุมมีความปลอดภัยมากขึ้นเพราะการตัดสินใจและการควบคุมการเคลื่อนที่ถูกแยกออกจากกัน

3.11 การกำหนดฟังก์ชันรางวัล

ในการกำหนดฟังก์ชันรางวัล (Reward function) สำหรับการฝึกโมเดลจะอ้างอิงกับข้อจำกัดและการแก้ปัญหาของการนำทาง โดยฟังก์ชันรางวัลที่กำหนดขึ้นจะทำให้ตัวแทนได้รางวัลมากขึ้นเมื่อเข้าใกล้เป้าหมายมากขึ้น ในทางกลับกัน หากตัวแทนออกห่างจากเป้าหมาย รางวัลที่ตัวแทนจะได้รับจะลดลง เมื่อเกิดการชนกับสิ่งกีดขวาง ตัวแทนจะเสียคะแนนอย่างมากและสภาพแวดล้อมจะถูกรีเซ็ทใหม่ทันที นอกจากนี้ เพื่อความปลอดภัยจะมีการกำหนดฟังกชันรางวัลของการเว้นระยะห่างออกจากกำพงของตัวแทนอีกด้วย โดยฟังก์ชันรางวัลการเข้าสู่เป้าหมาย (r_{target}), ฟังก์ชันรางวัลของการชน $(r_{collision})$ และฟังก์ชันรางวัลของการเว้นระยะห่างออกจากกำแพง (r_{wall}) ที่ถูกใช้ในงานวิจัยนี้สามารถเขียนได้ดังสมการด้านล่าง

$$r_{target} = \begin{cases} 1\,(d_p - d_c) > 0 \\ -1\,(d_p - d_c) < 0 \end{cases} \tag{3.3}$$

$$r_{collision} = \begin{cases} -200 & collision \\ 200 & goal \end{cases} \quad (3.4)$$

$$r_{wall} = c \cdot (\min(ranges) - d_{sm}) \quad (3.5)$$

เมื่อ

d_p คือ ระยะทางก่อนหน้าระหว่างตัวแทนกับเป้าหมาย [m]

d_c คือ ระยะทางปัจจุบันระหว่างตัวแทนกับเป้าหมาย [m]

d_{sm} คือ ขอบเขตระยะความปลอดภัยอย่างน้อยระหว่างสิ่งกีดขวางกับตัวแทน [m]

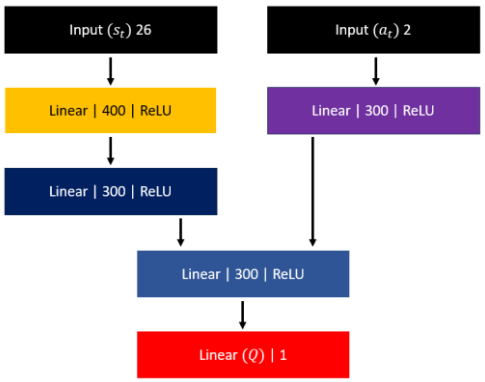
จากฟังก์ชันรางวัลทั้งหมด จะได้ว่า รางวัลของตัวแทนหลังจากผ่านการกระทำ a_t ในสภาวะ s_t จะสามารถหาได้จากสมการดังต่อไปนี้

$$r_t(s_t, a_t) = r_{target} + r_{collision} + r_{wall} \quad (3.6)$$

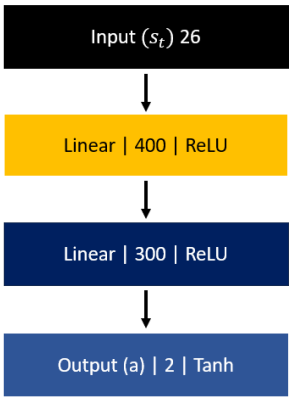
ฟังกันรางวัลทั้งหมดที่กล่าวมาจะถูกนำไปคำนวณรางวัลส่งให้กับโมเดลสำหรับทำการเรียนรู้และประเมินผลเพื่อให้โมเดลสามารถคำนวณหานโยบาย (policy) ที่ดีที่สุดของตัวแทนตามหลักการของการเรียนรู้แบบเสริมกำลังเชิงลึก

3.12 การสร้างโมเดลการเรียนรู้เสริมกำลังเชิงลึก

ในการสร้างโมเดลการเรียนรู้เสริมกำลังเชิงลึก เฟรมเวิร์คการเขียนโปรแกรมที่ใช้ คือ ไลบรารีไพธอร์ช (Pytorch) โดยโมเดลที่ถูกสร้างขึ้นแบ่งเป็น 2 ส่วน ได้แก่ โมเดลสำหรับเครือข่ายคริติก (Critic network) และโมเดลสำหรับเครือข่ายแสดง (Actor network) โดยเครือข่ายแสดงจะมีการรับข้อมูลขาเข้า (input) เป็นสภาวะเพียงอย่างเดียว และเครือข่ายคริติกจะมีการรับสภาวะและการกระทำเข้าไปในระบบเครือข่าย โครงสร้างของเครือข่ายทั้งสองประเภทแสดงดังในรูปต่อไปนี้



รูปที่ 3.13 โครงสร้างโครงข่ายคริติกที่ถูกใช้



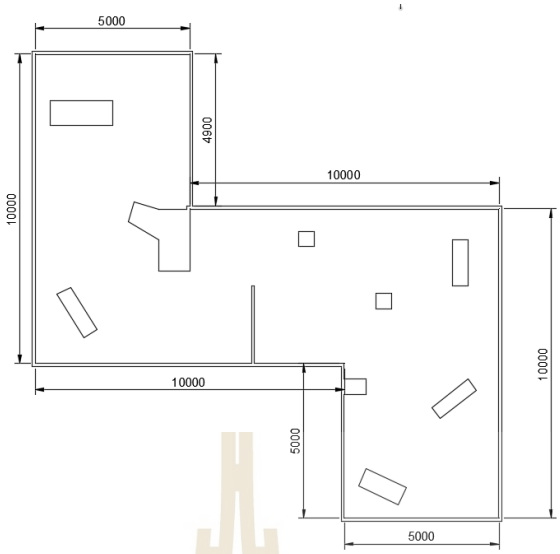
รูปที่ 3.14 โครงสร้างโครงข่ายตัวแสดงที่ถูกใช้

โครงข่ายตัวแสดง (actor network) จะรับข้อมูลขาเข้าเป็นสภาวะของตัวแทนแล้วนำไปประมวลผลหาการกระทำของตัวแทนออกมา โดยในตอนเริ่มแรก ข้อมูลที่รับเข้ามาจำนวน 24 ฟีเจอร์จะถูกนำไปผ่านชั้นเชื่อมต่อแบบเต็ม (fully connected layer) ที่มีจำนวนฟีเจอร์ขาออกเป็น 400 ฟีเจอร์ก่อนแล้วถูกทำแบทชนอร์มัลไลเซชั่น (batch normalization) เพื่อเพิ่มปรับค่าทางสถิติให้ใกล้เคียงก่อนก่อนที่จะผ่านฟังก์ชันกระตุ้นแบบ ReLU (Rectified Linear Unit) แล้วจึงถูกส่งไปยังชั้นเชื่อมต่อแบบเต็มที่ 2 ซึ่งมีจำนวนฟีเจอร์ขาออกเป็น 300 ฟีเจอร์ร่วมกับการทำแบทช์ นอร์มัลไลเซชันและฟังก์ชันกระตุ้นแบบ ReLU จนสุดท้ายถูกแปลงข้อมูลให้เป็นข้อมูลขาออกของการกระทำของตัวแทนออกมาจำนวน 2 ตัวที่ผ่านฟังก์ชันกระตุ้นแบบ Tanh โดยการกระทำที่ออกมาจะถูกส่งไปให้กับตัวแทน

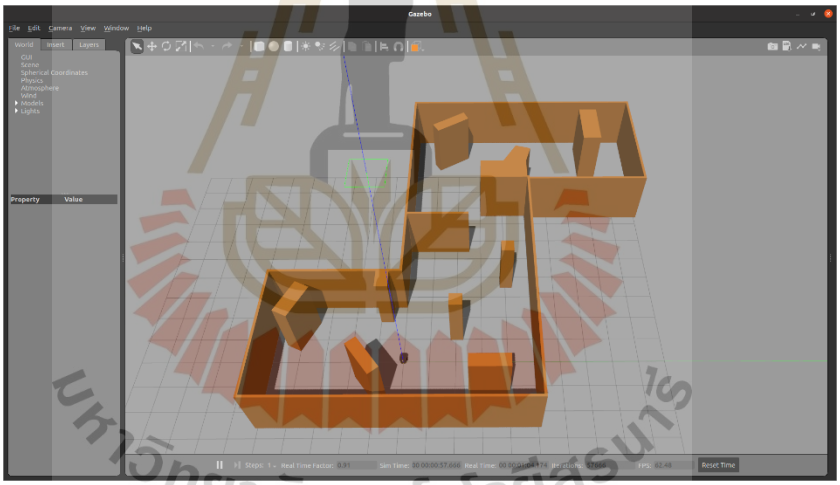
ในส่วนของโครงข่ายคริติก ค่าข้อมูลขาเข้าจะเป็นทั้งสภาวะและการกระทำที่ผ่านมาของตัวแทนโดยแยกออกเป็น 2 ส่วนก่อนแล้วผ่านโครงข่ายประสาทเทียมจนมาบรรจบกันก่อนการคำนวณค่า Q โดยที่จำนวนของฟีเจอร์และฟังก์ชันกระตุ้นสามารถดูได้จากรูปที่ 3.14 โดยข้อมูลขาออกที่ออกมาจากเครือข่ายคริติกจะเป็นค่า Q ของการกระทำที่สภาวะนั้นของตัวแทน ซึ่งจะถูกนำไปประเมินเพื่อปรับนโยบายเรื่อย ๆ จนได้นโยบายที่ดีที่สุดของตัวแทน

3.13 การฝึกโมเดลการเรียนรู้เสริมกำลังเชิงลึกในโปรแกรมจำลอง

การฝึกโมเดลการเรียนรู้แบบเสริมกำลังเชิงลึกจะมีการทำในโปรแกรมจำลอง Gazebo โดยการสร้างสภาพแวดล้อมจำลองสามมิติขึ้นมาด้วยโปรแกรม Fusion360 แล้วทำการส่งออกโมเดลที่ได้แปลงเป็นไฟล์ URDF โดยแบบของสนามจำลองแสดงดังในรูปต่อไปนี้



รูปที่ 3.15 แบบของสนามจำลองที่ถูกใช้ในการฝึกโมเดลพร้อมขนาดเบื้องต้นในหน่วยมิลลิเมตร



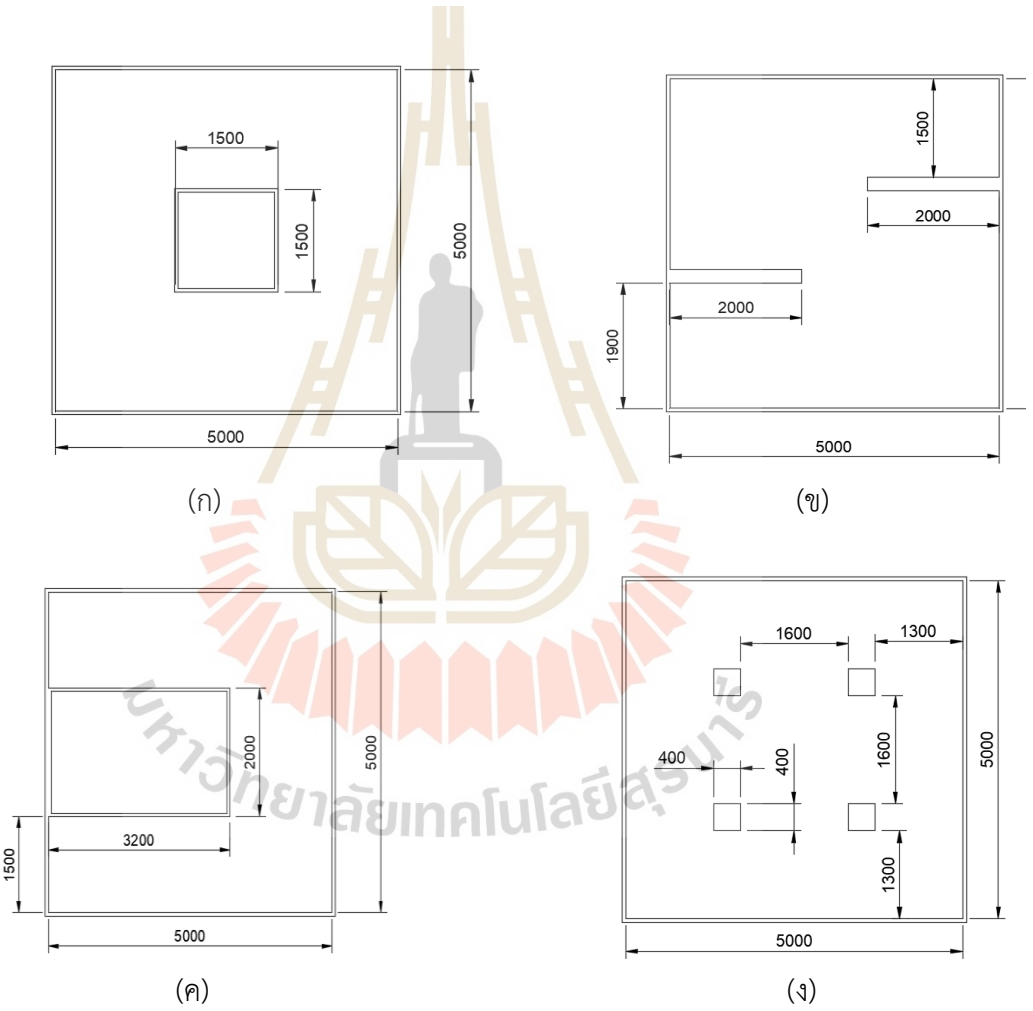
รูปที่ 3.16 สนามที่ถูกนำเข้าโปรแกรม Gazebo ร่วมกับโมเดลของตัวแทน

ในการฝึกโมเดล โปรแกรมจะทำการสุ่มจุดหมายที่ตัวแทนจะต้องไปถึงให้ได้ขึ้นมา จากนั้นจะส่งสัญญาณให้โมเดลเริ่มการฝึกโดยให้ตัวแทนเรียนรู้จากการย้ายตำแหน่งของตัวเองไปเรื่อย ๆ โดยมีรัศมีการเคลื่อนย้ายตำแหน่งไม่เกิน 1 เมตร ซึ่งในแต่ละรอบ (episode) จะมีการกำหนดจำนวนขั้น (step) สูงสุดของการเรียนรู้อยู่ที่ 2000 ขั้นต่อ 1 รอบ ซึ่งหากในรอบเดียวกันตัวแทนสามารถไปถึงเป้าหมายได้ก่อนหมดจำนวนขั้นสูงสุด โปรแกรมจะทำการสุ่มจุดหมายใหม่ขึ้นมาให้ตัวแทน แต่หากระหว่างรอบมีการชนเกิดขึ้นเพียงหนึ่งครั้ง สภาพแวดล้อมทั้งหมดจะถูกเริ่มต้นใหม่ทั้งหมดและขึ้นรอบ

ใหม่โดยทันที ในระหว่างการฝึกฝนจะมีการบันทึกค่าของรางวัลสะสมเอาไว้ หากค่าเฉลี่ยของรางวัลสะสมใน 100 รอบล่าสุดมีค่ามากกว่าค่าเฉลี่ยเดิมก่อนหน้า ตัวโมเดลจะถูกบันทึกและทำการอัพเดทไปเรื่อย ๆ จนกว่าการฝึกจะเสร็จสิ้น เพื่อให้ได้โมเดลหรือนโยบายที่ดีที่สุด

3.14 การทดสอบโมเดลในสภาพแวดล้อมจำลอง

หลังจากการฝึกฝนเสร็จเรียบร้อย โมเดลจะถูกนำไปทดสอบกับสภาพแวดล้อมจำลองก่อน โดยเป็นสนามที่ตัวแทนไม่เคยเห็นมาก่อน ซึ่งประกอบด้วยสนาม 4 สนามดังต่อไปนี้



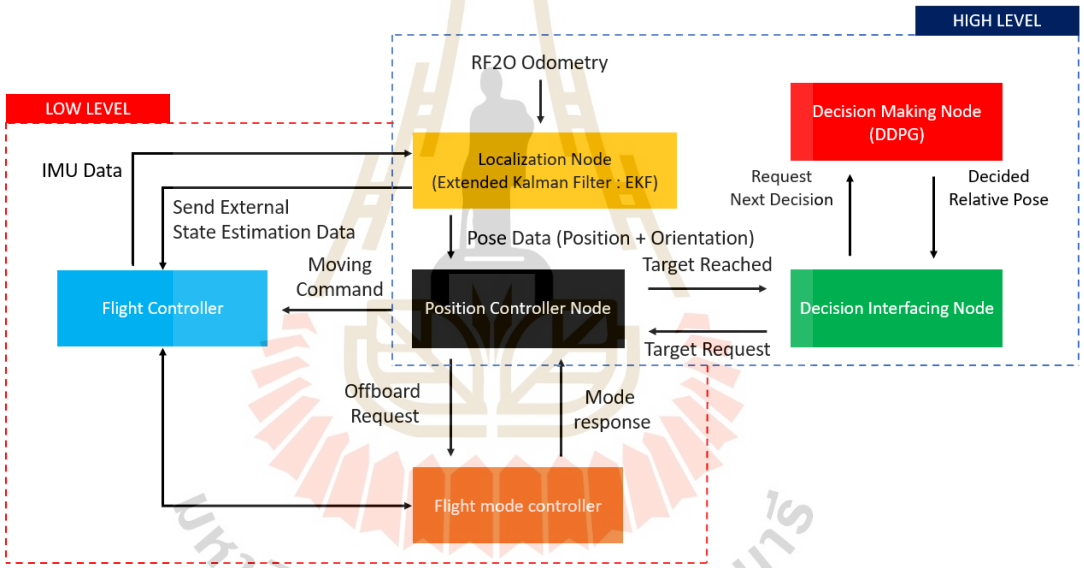
รูปที่ 3.17 แผนผังสนามจำลองที่ใช้ในการทดสอบ (ก) สนาม1 (ข) สนาม2 (ค) สนาม3 (ง) สนาม4

สำหรับการทดสอบในสนามจำลองทั้ง 4 สนาม ตัวแทนจะถูกนำไปปล่อยที่จุดเริ่มต้นของแต่ละสนามแล้วให้เดินทางไปที่จุดหมายที่กำหนดไว้ 1 จุดบนสนาม โดยจะทำการทดสอบทั้งหมด 5 ครั้ง

ต่อ 1 สนามแล้วทำการบันทึกเส้นทางที่ตัวแทนมีการตัดสินใจตั้งแต่ต้นทางจนถึงปลายทางเพื่อดูว่าตัวแทนสามารถตัดสินใจเส้นทางในการเดินทางไปถึงจุดหมายได้อย่างถูกต้องและปลอดภัยหรือไม่

3.15 การทดสอบโมเดลในอากาศยานไร้คนขับจริง

ในการทดสอบโมเดลการเรียนรู้ที่ฝึกได้ในอากาศยานไร้คนขับจริงจะต้องอาศัยการทำงานของส่วนประกอบทางโปรแกรมหลายส่วน ได้แก่ ส่วนของการระบุตำแหน่ง (localization node) ส่วนของการควบคุมโหมดการบิน (flight mode controller node) ส่วนของการควบคุมตำแหน่ง (position control node) ส่วนของการตัดสินใจ (decision making node) และส่วนของการเชื่อมต่อการตัดสินใจ (decision interfacing node) โดยแผนผังแสดงส่วนประกอบโปรแกรมและการส่งข้อมูลหากันระหว่างส่วนโปรแกรมแสดงดังในรูปต่อไปนี้



รูปที่ 3.18 แผนผังส่วนประกอบของโปรแกรมและการส่งข้อมูลที่สำคัญ

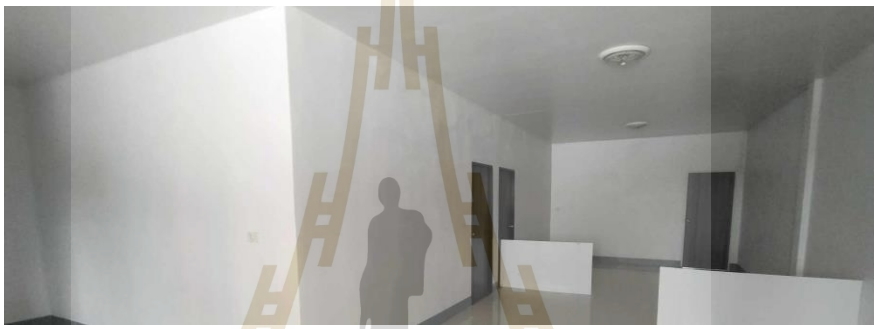
การทำงานของโปรแกรมจะแบ่งเป็นส่วนโปรแกรมระดับล่าง (low level) และโปรแกรมระดับสูง (high level) โดยส่วนของโปรแกรมระดับล่างจะเป็นส่วนของบอร์ดควบคุมการบิน (flight controller) ส่วนโปรแกรมระดับสูงจะอยู่ในคอมพิวเตอร์ควบคุมการบิน (flight computer) ซึ่งโปรแกรมสองส่วนจะมีการส่งข้อมูลไปมาหากันเพื่อทำงานร่วมกันโดยเชื่อมต่อกันผ่านระบบปฏิบัติการหุ่นยนต์และ MAVROS ในขั้นตอนการทดสอบของสภาวะจริงนั้น จะเริ่มต้นด้วยการนำอากาศยานขึ้นบินบนอากาศแล้วบังคับอากาศยานให้อยู่ในจุดที่ต้องการ จากนั้นทำการเปลี่ยนโหมดการบินเข้าสู่โหมดรักษาตำแหน่ง ซึ่งตัวควบคุมการบินจะดึงตำแหน่งจาก Localization node ที่เกิดจากการรวม

(fusion) กันระหว่างข้อมูลจากอัลกอริทึม RF2O และข้อมูลของ IMU ด้วยตัวกรองคาลมานแบบขยาย (Extended Kalman Filter : EKF) เพื่อนำไปใช้เป็นข้อมูลในการระบุตำแหน่งร่วมกับตัวประมาณสภาวะ (state estimation) ด้านในบอร์ดควบคุมการบินอีกชั้น แล้วทำการสร้างสัญญาณควบคุมมอเตอร์ของอากาศยานให้หมุนเพื่อรักษาตำแหน่งและระดับความสูงของอากาศยาน โดยความสูงที่ถูกกำหนดไว้อยู่ที่ 0.3 เมตรหรือ 30 เซนติเมตรจากพื้นดินถึงตัวไลดาร์เซนเซอร์ที่ยิงลงพื้น เมื่อทำการควบคุมตำแหน่งเริ่มต้นได้แล้วผู้วิจัยจะส่งจุดหมายที่อากาศยานต้องไปให้กับตัวคอมพิวเตอร์ควบคุมการบินโดยใช้การส่งสัญญาณผ่านสัญญาณไร้สายไวไฟ (WiFi) จากนั้นคอมพิวเตอร์ควบคุมการบินจะส่งจุดหมายผ่านตัวเชื่อมต่อการตัดสินใจ (decision interfacing node) เพื่อขอการตัดสินใจไปยังหน่วยการตัดสินใจ (decision making node) หน่วยการตัดสินใจที่มีโมเดลการเรียนรู้เสริมกำลังเชิงลึกอยู่จะทำการดึงสภาวะของอากาศยานทั้งหมดที่ต้องใช้ไปเข้าสู่การพิจารณาตามกระบวนการแล้วทำการตัดสินใจการกระทำเป็นตำแหน่งที่ต้องเคลื่อนที่ย้ายไปในแกน X และแกน Y ออกมา จากนั้นตำแหน่งดังกล่าวจะถูกส่งไปให้หน่วยควบคุมตำแหน่งเพื่อเปรียบเทียบกับตำแหน่งปัจจุบัน แปลงพิกัดเฟรมอ้างอิง แล้วทำการส่งคำร้องไปที่หน่วยควบคุมการบินเพื่อเปลี่ยนโหมดการบินเป็นโหมดออฟบอร์ดให้บอร์ดควบคุมการบินสามารถรับคำสั่งการควบคุมจากภายนอกได้ จากนั้นบอร์ดควบคุมการบินจะทำการควบคุมให้อากาศยานเคลื่อนที่ไปตำแหน่งที่ต้องการ จากนั้นหน่วยควบคุมตำแหน่งจะคอยตรวจสอบว่าอากาศยานถึงขอบเขตของตำแหน่งที่ส่งไปหรือไม่ หากอากาศยานไปถึงขอบเขตจุดหมายที่ยอมรับได้แล้ว หน่วยควบคุมตำแหน่งจะส่งสัญญาณไปให้หน่วยเชื่อมต่อการตัดสินใจเพื่อขอการตัดสินใจตำแหน่งถัดไปจากหน่วยการตัดสินใจ แล้วเกิดการวนการทำงานไปจนกระทั่งอากาศยานไร้คนขับสามารถเดินทางไปถึงเป้าหมายได้สำเร็จ โดยมีระยะความเผื่ออยู่ที่ 0.15 เมตรหรือ 15 เซนติเมตร

ในการทดสอบอากาศยานในพื้นที่จริง จะทำการทดสอบในพื้นที่ที่มีขนาดกว้าง x ยาว x สูงสุดที่ 7.96 x 12.07 x 2.5 เมตรตามลำดับ โดยลักษณะของพื้นที่แสดงดังในรูปที่ 3.19 และจะมีการทดสอบโดยการสร้างสิ่งกีดขวางขึ้นมา 2 รูปแบบดังในรูปที่ 3.20 และรูปที่ 3.21

รูปที่ 3.19 ลักษณะพื้นที่สำหรับการทดสอบ (ยังไม่ถูกเพิ่มสิ่งกีดขวาง)

รูปที่ 3.20 สิ่งกีดขวางสำหรับการทดสอบรูปแบบที่ 1



รูปที่ 3.21 สิ่งกีดขวางสำหรับการทดสอบรูปแบบที่ 2

เมื่อทำการทดสอบข้อมูลการตัดสินใจเลือกในเส้นทางและข้อมูลตำแหน่งของอากาศยานจะถูกนำมาเปรียบเทียบกับเส้นทางการเคลื่อนที่ที่หาได้จากการวางแผนเส้นทางในกรณีที่รู้แผนที่มาก่อน โดยเทียบกับอัลกอริทึม A* (A – star algorithm) เพื่อประเมินความสามารถของโมเดลการเรียนรู้เชิงลึก ลักษณะการทดสอบอากาศยานไร้คนขับแสดงดังในรูปที่ 3.22

รูปที่ 3.22 ลักษณะการทดสอบโมเดลในอากาศยานไร้คนขับจริง (ซ้าย) แบบ1 (ขวา) แบบ2

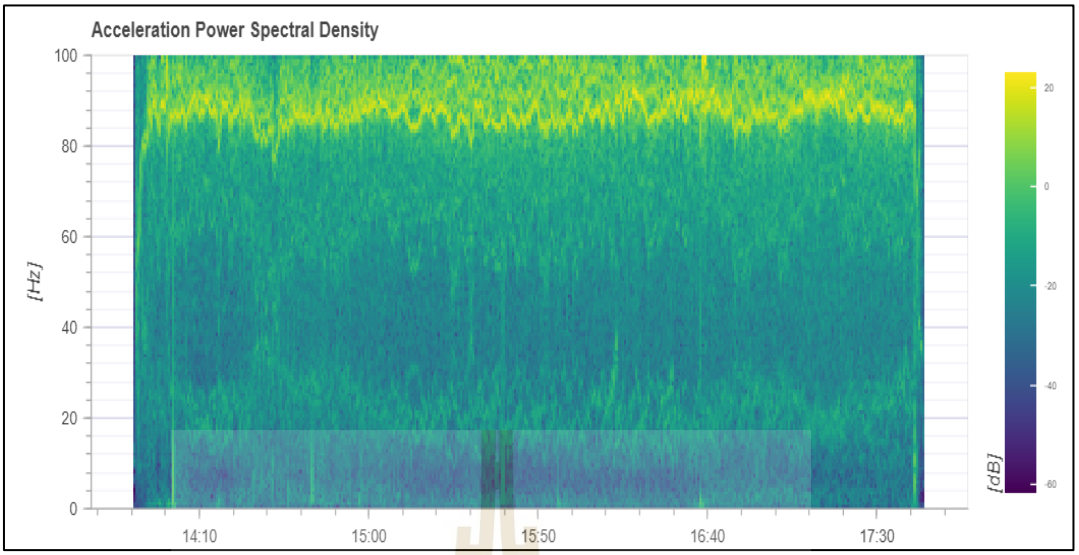
บทที่ 4
ผลการวิจัยและอภิปรายผล

4.1 กล่าวนำ

เนื้อหาในส่วนของการทดสอบจะอ้างอิงตามวิธีการทดสอบที่มีการกล่าวถึงไว้ในบทที่ 3 โดยจะแสดงผลการทดสอบในแต่ละส่วนออกมาในรูปของตาราง รูปภาพ กราฟหรือแผนผังการเปรียบเทียบอื่น ๆ เพื่อนำเสนอข้อมูลของการทดสอบให้ดูง่ายและชัดเจน นอกจากนี้ เนื้อหาในส่วนนี้จะเกี่ยวข้องไปถึงการวิเคราะห์สิ่งที่เกิดขึ้นจากการทดสอบอีกด้วย

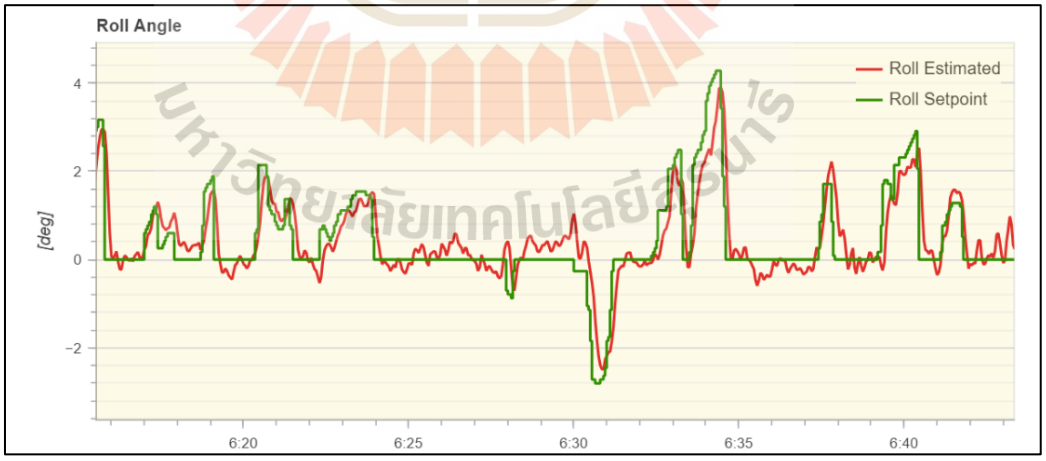
4.2 ผลการทดสอบการบินขั้นพื้นฐานในอุปกรณ์จริง

ผลการทดสอบการบินสิ่งแรกที่สำคัญคือผลการทดที่สามารถบ่งบอกลักษณะการสั่นสะเทือนของอากาศยานได้ เพราะการสั่นสะเทือนเป็นปัญหาที่สำคัญที่สุดของการควบคุมอากาศยานไร้คนขับ เพราะการสั่นสะเทือนที่สูงเกินไปจะทำให้เซนเซอร์อ่านค่าผิดพลาดและทำให้การควบคุมเสียเสถียรภาพไปได้โดยสิ่งแรกที่สามารถพิจารณาได้คือค่าความหนาแน่นสเปคตรัมความเร่งที่แสดงดังต่อไปนี้ โดยเป็นการสแดงภาพรวมการตอบสนองเชิงความถี่ของข้อมูลความเร่งดิบตลอดช่วงเวลาที่ทำการบิน หากค่าความหนาแน่นมีค่ามากหรือเป็นสีเหลืองกระจายไปทั่วแผนภาพแสดงว่ามีการสั่นสะเทือนสูง และหากมีการสั่นสะเทือนอยู่ในช่วงความถี่มากกว่า 80 Hz ตัวควบคุมการบินจะสามารถกรองสัญญาณออกไปได้ซึ่งจากในรูปด้านล่างพบว่า การสั่นสะเทือนของอากาศยานอยู่ในช่วงที่สามารถกรองออกได้และไม่ทำให้เกิดความสูญเสียในการควบคุม

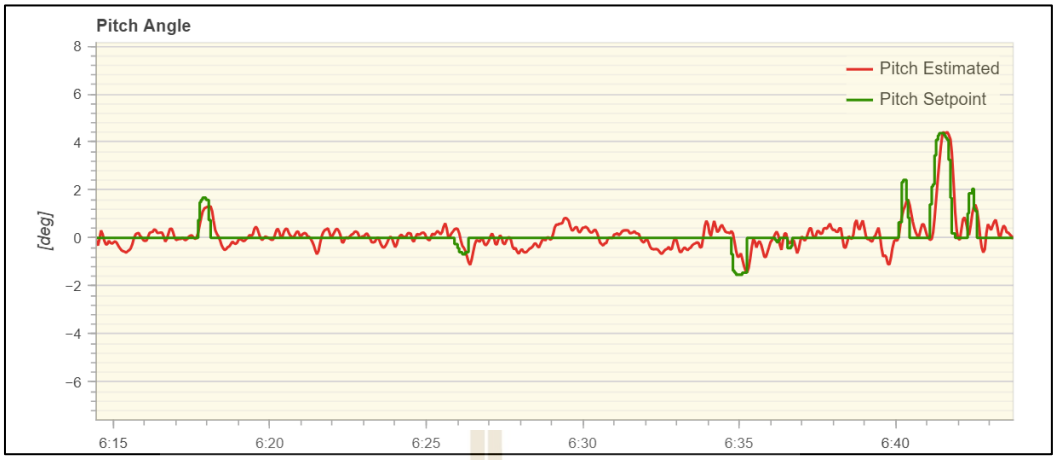


รูปที่ 4.1 ความหนาแน่นสเปคตรัมความเร่งของอากาศยาน

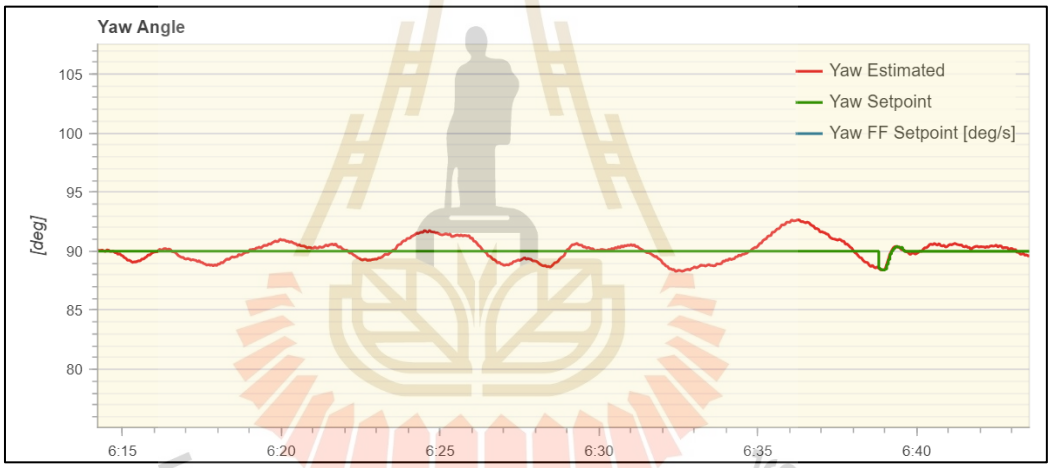
นอกจากการสั่นสะเทือนแล้ว สามารถดูการตอบสนองของการควบคุมท่าทาง (attitude) ของอากาศยานได้โดยให้ความสนใจการควบคุมโหมดรักษาระดับเพราะจะเป็นอย่างมากในการบินอาคารร่วมกับโหมดรักษาตำแหน่ง แต่ผลการทดสอบในโหมดรักษาตำแหน่งจะถูกกล่าวถึงในส่วนของการทดสอบโปรแกรมในโหมดออฟบอร์ด



รูปที่ 4.2 ผลตอบสนองของอากาศยานแนวแกน roll ในโหมดควบคุมความสูง



รูปที่ 4.3 ผลตอบสนองของอากาศยานแนวแกน pitch ในโหมดควบคุมความสูง



รูปที่ 4.4 ผลตอบสนองของอากาศยานแนวแกน yaw ในโหมดควบคุมความสูง

จากผลการทดสอบผบว่าในโหมดรักษาระดับความสูง (altitude hold mode) การตอบสนองของการควบคุมท่าทองของอากาศยานมีความใกล้เคียงกับค่าที่ต้องการ (setpoint) และมีความเร็วในการตอบสนองที่เพียงพอเมื่อทำการบินโดยค่าเกนการควบคุมของ k_p ในแนวแกน roll , pitch และ yaw เป็น 6.5, 6.5 และ 2.8 ตามลำดับ

4.3 ผลการทดสอบโปรแกรมในโหมดออฟบอร์ด

การทดสอบการทำงานของโปรแกรมในโหมดออฟบอร์ดจะใช้การตรวจสอบตามหัวข้อตรวจสอบที่ถูกกล่าวถึงในหัวข้อ 3.9 โดยผลการทดสอบและผลการตัดสินเป็นดังตารางที่ 4.1

ตารางที่ 4.1 ผลการตรวจสอบการทำงานในโหมดออฟบอร์ด

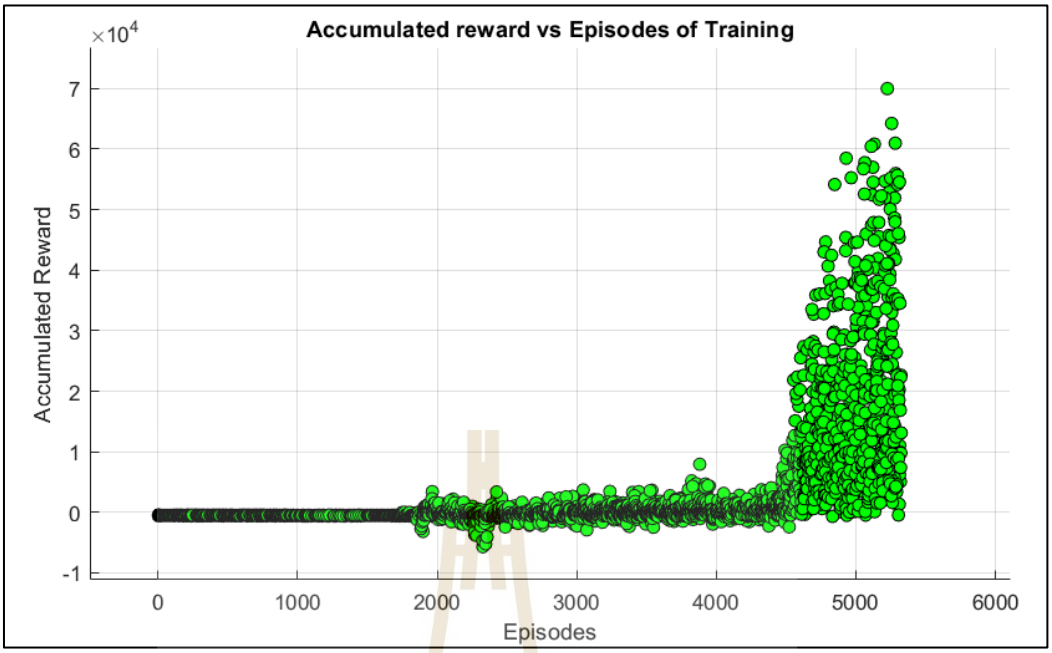
ลำดับ	รายละเอียดการตรวจสอบ	ผลที่คาดหมาย / ตัวชี้วัด	ผลการทดสอบ	การตัดสิน
1	นำอากาศยานขึ้นบินแล้วปรับโหมดการบินเป็นโหมดการรักษาตำแหน่ง	อากาศยานต้องสามารถรักษาตำแหน่งของตนเองได้	อากาศยานสามารถรักษาตำแหน่งของตนเองได้	ผ่าน
2	รันโปรแกรมสำหรับการส่งตำแหน่งในโหมดออฟบอร์ดให้กับอากาศยาน	อากาศยานต้องเปลี่ยนโหมดการบินเป็นโหมดออฟบอร์ด	อากาศยานเปลี่ยนโหมดการบินเป็นโหมดออฟบอร์ดได้โดยไม่สูญเสียความสูงและตำแหน่ง	ผ่าน
3	ส่งค่าตำแหน่งที่ต้องการให้อากาศยานเคลื่อนที่ไปผ่านโหมดออฟบอร์ด	อากาศยานต้องสามารถเคลื่อนที่ไปตำแหน่งที่ต้องการได้ตามความเร็วการเคลื่อนที่สูงสุดที่ถูกกำหนดเอาไว้ (0.2 เมตรต่อวินาที) โดยที่ไม่เสียอาการ	อากาศยานสามารถรับค่าตำแหน่งในโหมดออฟบอร์ดและเคลื่อนที่ไปยังตำแหน่งที่ต้องการได้ โดยที่แกนบวกลบของระบบพิกัดอ้างอิงมีความถูกต้อง	ผ่าน
4	ทดสอบหยุดการรันโปรแกรมสำหรับส่งคำสั่งในโหมดออออฟบอร์ด	โหมด failsafe ต้องทำงานโดยการเปลี่ยนจากโหมดออฟบอร์ดเป็นโหมดรักษาตำแหน่ง โดยที่ไม่สูญเสียความสูงและสเถียรภาพการบิน	โหมด failsafe ทำงานถูกต้องโดยเมื่อหยุดโปรแกรมแล้ว อากาศยานมีการเปลี่ยนโหมดเป็นโหมดรักษาตำแหน่งทันทีโดยที่ไม่เสียอาการการบิน	ผ่าน

ตารางที่ 4.2 ผลการตรวจสอบการทำงานในโหมดออฟบอร์ด (ต่อ)

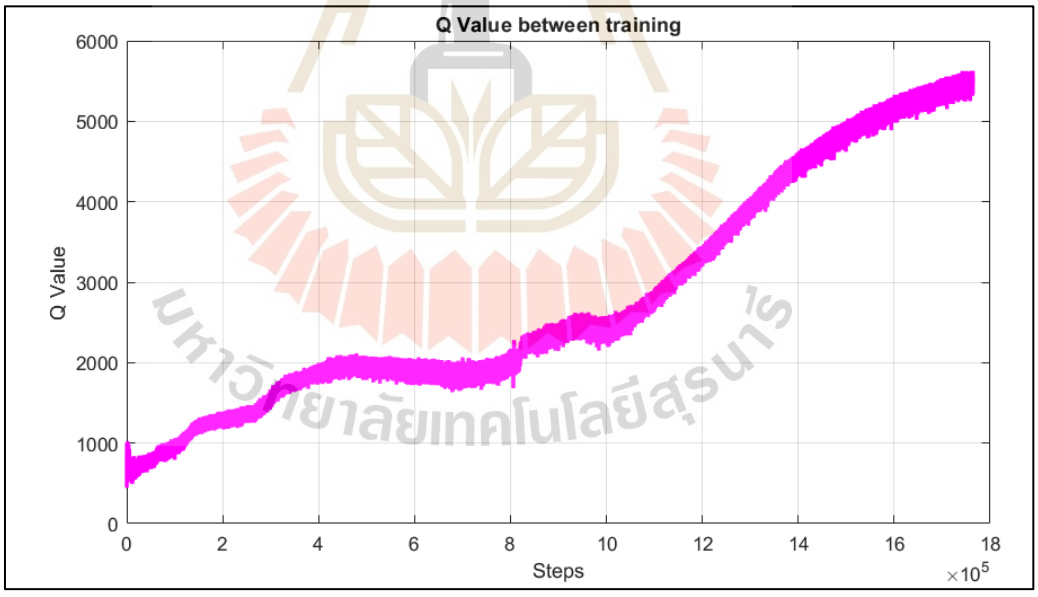
ลำดับ	รายละเอียดการตรวจสอบ	ผลที่คาดหมาย / ตัวชี้วัด	ผลการทดสอบ	การตัดสิน
ลำดับ	รายละเอียดการตรวจสอบ	ผลที่คาดหมาย / ตัวชี้วัด	ผลการทดสอบ	การตัดสิน
5	นำอากาศยานลงจอดแล้วนำไฟล์บันทึกการบินมาตรวจสอบความถูกต้องของการเปลี่ยนโหมดและความถูกต้องของการควบคุมตำแหน่งการเคลื่อนที่	ข้อมูลที่ถูกบันทึกในไฟล์บันทึกการบินต้องมีความถูกต้อง	ข้อมูลของอากาศยานมีการบันทึกสมบูรณ์	ผ่าน

4.4 ผลการฝึกโมเดลการเรียนรู้แบบเสริมกำลังเชิงลึกในโปรแกรมจำลอง

การฝึกฝนโมเดลการเรียนรู้แบบเสริมกำลังเชิงลึกในโปรแกรมจำลองจะทำให้นโยบายการตัดสินใจของตัวแทนมีการพัฒนาอย่างต่อเนื่อง โดยในงานวิจัยครั้งนี้ใช้เวลาในการฝึกฝนโมเดลไปทั้งหมด 172 ชั่วโมง (7 วัน 4 ชั่วโมง) โดยใช้รอบในการคำนวณ (episodes) ไปทั้งหมด 5324 เอพิโสด และใช้จำนวนขั้นไปทั้งหมด 1,764,206 ขั้น โดยกราฟแสดงรางวัลสะสมในแต่ละเอพิโสดของตัวแทนแสดงดังในรูปต่อไปนี้โดยมีค่าสูงสุดอยู่ที่ 69,960 โดยคอมพิวเตอร์ที่ใช้ฝึกโมเดลมี CPU core i5-11300H (11th gen) RAM8GB การ์ดจอ Nvidia MX250



รูปที่ 4.5 รางวัลสะสมของตัวแทนในแต่ละรอบการคำนวณ



รูปที่ 4.6 ค่า Q Value ระหว่างการฝึกโมเดลแต่ละขั้น

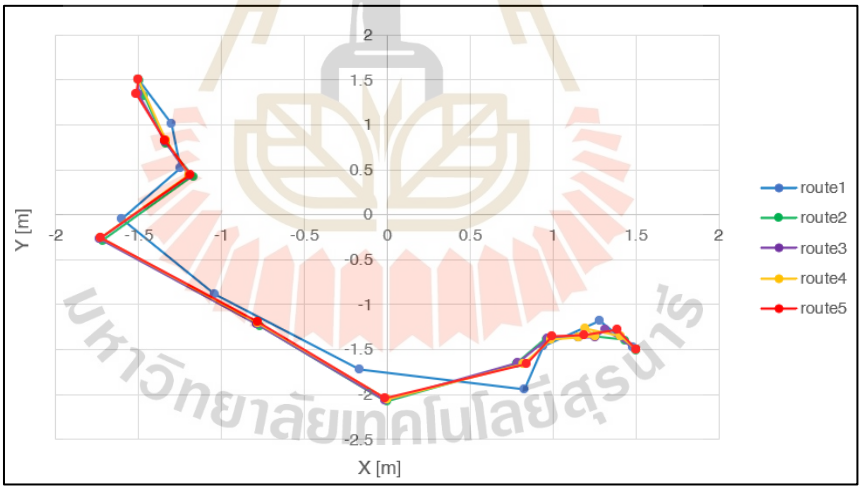
ในระหว่างการฝึกโมเดล ตัวแทนจะค่อย ๆ ทำการปรับพารามิเตอร์ของเครือข่ายต่าง ๆ สำหรับการเรียนรู้ และเมื่อนโยบายมีการพัฒนามากขึ้นค่า Q value ก็จะมีแนวโน้มที่สูงมากขึ้น โดย

จากการฝึกโมเดลในงานวิจัยนี้แนวโน้มของค่า Q Value มีการเพิ่มสูงขึ้นอย่างต่อเนื่องในแต่ละขั้น โดย Q value สูงสุดอยู่ที่ 5,627

จากรูปที่ 4.5 และรูปที่ 4.6 บ่งชี้ว่าโมเดลมีการเรียนรู้ที่พัฒนาขึ้นอย่างก้าวกระโดดในช่วงรอบการคำนวณที่ประมาณ 4500 ซึ่งการที่คะแนนสะสมมีการเพิ่มขึ้นสูง แสดงให้เห็นว่าหุ่นยนต์สามารถตัดสินใจเส้นทางแล้วไปหาจุดหมายได้หลายจุดหมายต่อ 1 รอบ ซึ่งในบางรอบจะไม่มีการชนเกิดขึ้นเลย ตัวแทนสามารถทำงานได้โดยที่ไม่ต้องเริ่มต้นรอบการคำนวณใหม่ สามารถเรียนรู้ได้จนถึงจำนวนขั้นสูงสุด

4.5 ผลการทดสอบโมเดลในสภาพแวดล้อมจำลอง

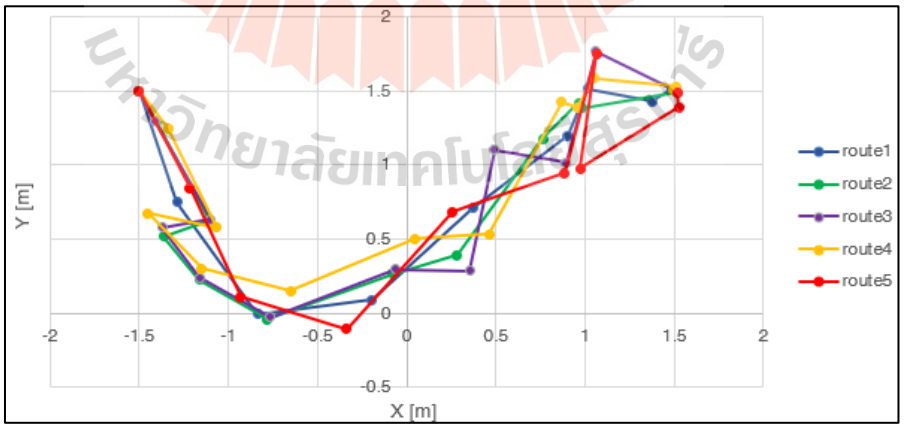
ผลการทดสอบโมเดลในสภาพแวดล้อมจำลองประกอบด้วยผลการตัดสินใจเลือกตำแหน่งการเคลื่อนที่ของตัวแทนใน 4 สภาพแวดล้อมที่ไม่รู้จักมาก่อนโดยสภาพแวดล้อมจะอ้างอิงตามหัวข้อ 3.15 ซึ่งตารางของตำแหน่งการเคลื่อนที่ที่ตัวแทนเลือกและรูปที่นำค่าพิกัดของแกน x และ y มาวาดเป็นกราฟในแต่ละสภาพแวดล้อมแสดงดังต่อไปนี้



รูปที่ 4.7 พิกัดการเคลื่อนที่ของตัวแทนในสภาพแวดล้อมจำลองที่ 1

ตารางที่ 4.3 ตารางบันทึกการเลือกเส้นทางของตัวแทนในสภาพแวดล้อมจำลองที่ 1

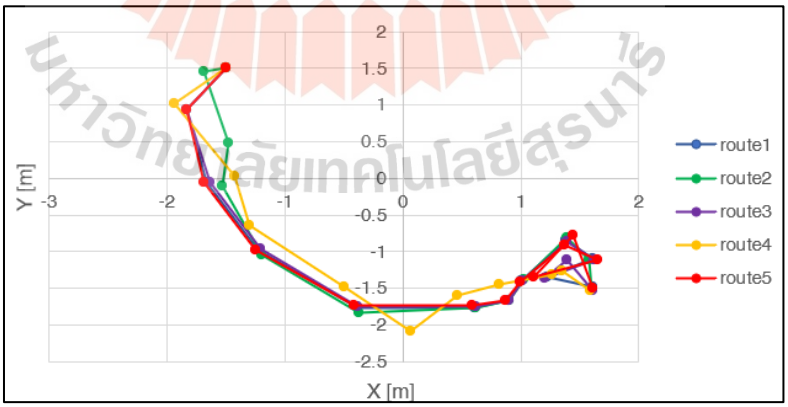
	สภาพแวดล้อมจำลองที่ 1									
ครั้ง	1		2		3		4		5	
ลำดับ	x	y	x	y	x	y	x	y	x	y
1	-1.50	1.50	-1.50	1.50	-1.50	1.50	-1.50	1.50	-1.50	1.50
2	-1.30	1.01	-1.47	1.33	-1.49	1.34	-1.34	0.83	-1.52	1.35
3	-1.25	0.52	-1.34	0.79	-1.34	0.82	-1.20	0.44	-1.35	0.83
4	-1.60	-0.04	-1.18	0.43	-1.20	0.44	-1.73	-0.26	-1.19	0.45
5	-1.04	-0.89	-1.72	-0.28	-1.74	-0.27	-0.78	-1.21	-1.73	-0.25
6	-0.17	-1.72	-0.77	-1.23	-0.79	-1.22	0.00	-2.06	-0.78	-1.20
7	0.83	-1.95	-0.01	-2.07	-0.01	-2.06	0.82	-1.66	-0.01	-2.04
8	0.94	-1.46	0.78	-1.66	0.79	-1.65	0.99	-1.38	0.84	-1.66
9	1.29	-1.18	0.96	-1.38	0.97	-1.38	1.16	-1.36	1.00	-1.36
10	1.49	-1.47	1.15	-1.36	1.15	-1.37	1.26	-1.35	1.19	-1.35
11	1.49	-1.47	1.25	-1.35	1.26	-1.36	1.20	-1.26	1.39	-1.28
12			1.43	-1.39	1.32	-1.28	1.41	-1.35	1.47	-1.51
13			1.51	-1.50	1.48	-1.45	1.51	-1.50		



รูปที่ 4.8 พิกัดการเคลื่อนที่ของตัวแทนในสภาพแวดล้อมจำลองที่ 2

ตารางที่ 4.4 ตารางบันทึกการเลือกเส้นทางของตัวแทนในสภาพแวดล้อมจำลองที่ 2

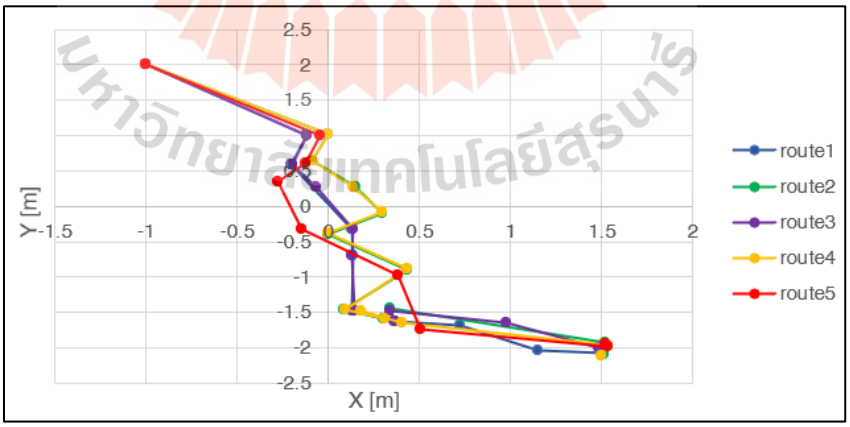
สภาพแวดล้อมจำลองที่ 2										
ครั้ง	1		2		3		4		5	
ลำดับ	x	y	x	y	x	y	x	y	x	y
1	-1.50	1.50	-1.50	1.50	-1.50	1.50	-1.50	1.50	-1.50	1.50
2	-1.29	0.75	-1.40	1.30	-1.41	1.30	-1.34	1.25	-1.21	0.85
3	-0.84	-0.01	-1.11	0.63	-1.10	0.63	-1.06	0.58	-0.93	0.11
4	-0.19	0.09	-1.36	0.52	-1.37	0.58	-1.45	0.68	-0.34	-0.11
5	0.37	0.71	-1.16	0.22	-1.16	0.24	-1.15	0.30	0.26	0.68
6	0.90	1.20	-0.78	-0.04	-0.76	-0.03	-0.65	0.15	0.88	0.95
7	1.02	1.51	-0.08	0.27	-0.06	0.29	0.05	0.50	1.07	1.75
8	1.38	1.43	0.28	0.39	0.36	0.29	0.46	0.53	0.97	0.97
9	1.49	1.51	0.76	1.18	0.49	1.10	0.87	1.43	1.53	1.39
10			0.96	1.42	0.89	1.02	0.96	1.39	1.52	1.49
11			0.98	1.38	1.06	1.77	1.05	1.59		
12			1.52	1.49	1.51	1.50	1.51	1.53		
13							1.49	1.51		



รูปที่ 4.9 พิกัดการเคลื่อนที่ของตัวแทนในสภาพแวดล้อมจำลองที่ 3

ตารางที่ 4.5 ตารางบันทึกการเลือกเส้นทางของตัวแทนในสภาพแวดล้อมจำลองที่ 3

สภาพแวดล้อมจำลองที่ 3										
ครั้ง	1		2		3		4		5	
ลำดับ	x	y	x	y	x	y	x	y	x	y
1	-1.50	1.50	-1.50	1.50	-1.50	1.50	-1.50	1.50	-1.50	1.50
2	-1.83	0.94	-1.68	1.46	-1.84	0.94	-1.94	1.02	-1.84	0.94
3	-1.67	-0.06	-1.47	0.49	-1.64	-0.06	-1.43	0.02	-1.69	-0.06
4	-1.23	-0.98	-1.53	-0.10	-1.21	-0.97	-1.30	-0.64	-1.25	-0.97
5	-0.39	-1.76	-1.20	-1.05	-0.38	-1.76	-0.50	-1.49	-0.41	-1.74
6	0.61	-1.76	-0.37	-1.83	0.62	-1.75	0.07	-2.08	0.58	-1.74
7	0.89	-1.66	0.62	-1.77	0.90	-1.67	0.46	-1.60	0.87	-1.67
8	1.01	-1.40	0.90	-1.66	1.01	-1.40	0.82	-1.45	0.99	-1.42
9	1.38	-0.85	1.02	-1.38	1.39	-0.86	1.26	-1.31	1.37	-0.90
10	1.61	-1.09	1.39	-0.82	1.62	-1.11	1.35	-1.27	1.65	-1.12
11	1.23	-1.34	1.57	-1.11	1.20	-1.36	1.59	-1.53	1.11	-1.35
12	1.61	-1.48	1.60	-1.51	1.39	-1.12			1.44	-0.77
13					1.61	-1.52			1.61	-1.49



รูปที่ 4.10 พิกัดการเคลื่อนที่ของตัวแทนในสภาพแวดล้อมจำลองที่ 4

ตารางที่ 4.6 ตารางบันทึกการเลือกเส้นทางของตัวแทนในสภาพแวดล้อมจำลองที่ 4

สภาพแวดล้อมจำลองที่ 4										
ครั้ง	1		2		3		4		5	
ลำดับ	x	y	x	y	x	y	x	y	x	y
1	-1.00	2.00	-1.00	2.00	-1.00	2.00	-1.00	2.00	-1.00	2.00
2	-0.12	1.01	0.00	1.02	-0.12	1.01	0.00	1.03	-0.04	1.01
3	-0.20	0.59	-0.09	0.65	-0.20	0.59	-0.09	0.65	-0.12	0.61
4	0.13	-0.32	0.15	0.27	-0.07	0.27	0.14	0.28	-0.28	0.36
5	0.14	-0.69	0.29	-0.09	0.14	-0.32	0.30	-0.08	-0.15	-0.32
6	0.13	-1.49	0.00	-0.39	0.13	-0.69	-0.01	-0.38	0.38	-0.97
7	0.36	-1.62	0.43	-0.90	0.14	-1.48	0.43	-0.89	0.51	-1.74
8	0.72	-1.68	0.09	-1.47	0.37	-1.62	0.09	-1.47	1.53	-1.98
9	1.15	-2.03	0.30	-1.59	0.34	-1.49	0.31	-1.60	1.52	-1.95
10	1.51	-2.07	0.34	-1.45	0.98	-1.65	0.18	-1.48		
11			1.52	-1.92	1.49	-2.01	0.40	-1.65		
12			1.51	-2.10			1.53	-1.97		
13							1.50	-2.12		

จากผลการทดสอบ พบว่า ในแต่ละสภาพแวดล้อมจำลอง ตัวแทนสามารถหาเส้นทางไปสู่จุดหมายได้โดยที่เส้นทางที่สร้างขึ้นในแต่ละครั้งจะมีความใกล้เคียงกัน โดยบริเวณเส้นโค้งเป็นบริเวณที่ตัวแทนมีการเลือกเส้นทางอ้อมเพื่อหลบหลีกสิ่งกีดขวาง จำนวนขั้นที่ใช้ขั้นต่ำสำหรับเดินทางจากจุดเริ่มต้นถึงจุดหมายเป็นจำนวนอย่างน้อย 9 ขั้นและสูงสุดที่ 13 ขั้น ระยะทางในช่วงเริ่มต้นจะมีระยะทางที่ยาว แต่ในช่วงท้ายจะมีระยะทางที่สั้นเพื่อต้องการปรับตำแหน่งของตัวแทนให้ใกล้กับจุดหมายมากที่สุด

4.6 ผลการทดสอบโมเดลในอากาศยานไร้คนขับจริง

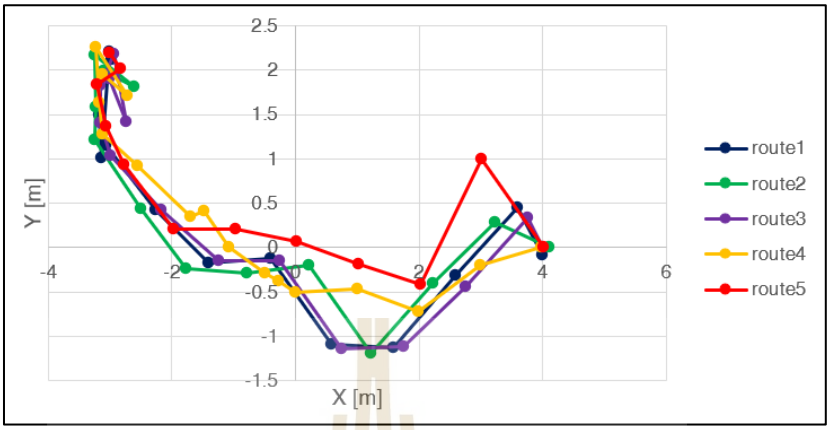
ในการทดสอบโมเดลในอากาศยานไร้คนขับจริง อากาศยานถูกทดสอบในสภาวะแวดล้อม 2 แบบ โดยได้รับการทดสอบด้วยจุดเริ่มต้นและจุดหมายเดียวกันจำนวน 5 ครั้ง ต่อหนึ่งสภาพแวดล้อม การคำนวณจะเกิดขึ้นบน GPU ของบอร์ด Nvidia Jetson Nano RAM4GB โดยในสภาพแวดล้อมแบบที่ 1 ตารางข้อมูลการตัดสินใจของตัวแทนในอากาศยานมีการตัดสินใจเคลื่อนที่ไป

ตามพิกัดดังตารางที่ 4.6 และกราฟแสดงเส้นทางการตัดสินใจจะแสดงในรูปที่4.11 โดยในสภาพแวดล้อมที่1จุดเริ่มต้นอยู่ที่พิกัด (4.0 , 0.0) และจุดหมายปลายทางคือ (-3.0, 2.0)

ตารางที่ 4.7 ตารางบันทึกการเลือกเส้นทางของอากาศยานในสภาพแวดล้อมจริงที่ 1

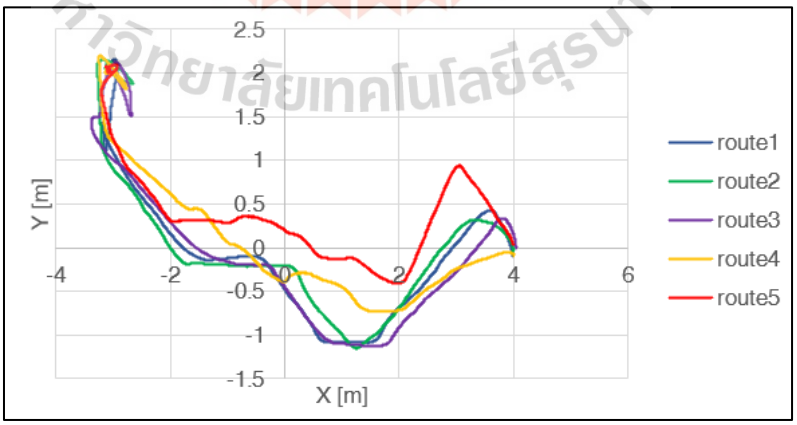
สภาพแวดล้อมจริงที่ 1										
ครั้ง	**1**		**2**		**3**		**4**		**5**	
ลำดับ	**x**	**y**	**x**	**y**	**x**	**y**	**x**	**y**	**x**	**y**
1	3.99	-0.08	4.11	0.00	4.00	0.00	3.98	0.00	4.01	0.00
2	3.59	0.45	3.22	0.29	3.75	0.33	3.00	-0.20	3.02	1.00
3	2.59	-0.31	2.22	-0.40	2.75	-0.44	2.00	-0.72	2.03	-0.42
4	1.59	-1.12	1.22	-1.19	1.75	-1.12	1.00	-0.47	1.03	-0.19
5	0.59	-1.09	0.22	-0.20	0.75	-1.14	0.00	-0.50	0.02	0.07
6	-0.41	-0.12	-0.78	-0.28	-0.25	-0.15	-0.28	-0.37	-0.98	0.21
7	-1.41	-0.17	-1.78	-0.24	-1.25	-0.15	-0.49	-0.29	-1.98	0.20
8	-2.27	0.43	-2.50	0.44	-2.17	0.42	-1.07	0.01	-2.78	0.93
9	-3.07	1.15	-3.25	1.21	-3.00	1.04	-1.48	0.42	-3.07	1.37
10	-3.18	1.49	-3.24	1.58	-3.15	1.40	-1.71	0.34	-3.22	1.84
11	-3.14	1.01	-3.25	2.17	-3.16	1.83	-2.56	0.92	-2.82	2.02
12	-3.00	2.21	-2.61	1.82	-2.93	2.18	-3.12	1.28	-3.01	2.19
13	-2.98	2.12	-3.11	1.99	-2.73	1.42	-3.18	1.64		
14					-3.01	1.90	-3.23	2.26		
15							-2.72	1.71		
16							-3.12	1.95		

โดยหากนำข้อมูลการตัดสินใจของอากาศยานทั้ง 5 ครั้งมาพลอตร่วมกันจะได้กราฟดังในรูปต่อไปนี้



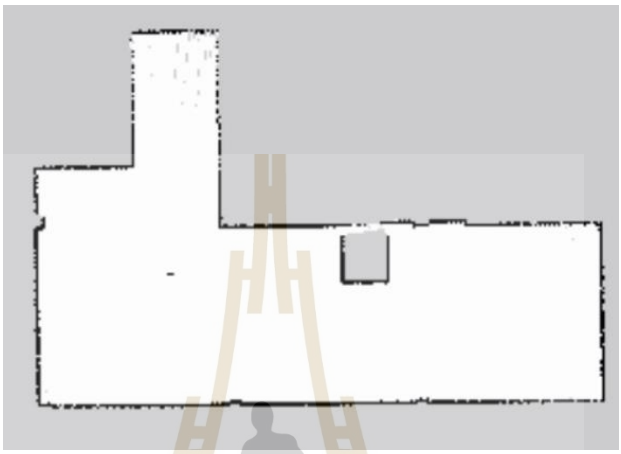
รูปที่ 4.11 พิกัดการเคลื่อนที่ของตัวแทนในสภาพแวดล้อมจริงที่ 1 ที่ถูกตัดสินใจ

โดยเส้นทางการเคลื่อนที่จริงของอากาศยานเมื่อทำตามการตัดสินใจในแต่ละขั้นจากจุดเริ่มต้นไปถึงจุดหมายสามารถแสดงเปรียบเทียบกันได้ดังรูปต่อไปนี้ ซึ่งจะมีความคล้ายคลึงกับเส้นทางที่ตัดสินใจออกมาแต่ว่าเส้นทางจะมีความต่อเนื่องเพราะเป็นตำแหน่งการเคลื่อนที่จริงที่มีจุดข้อมูลมากกว่า โดยเส้นทางที่ 5 จะมีความใกล้ชิดกับสิ่งกีดขวางมากกว่าเส้นทางอื่น ช่วงระยะห่างระหว่างสิ่งกีดขวางกับเส้นทางที่น้อยที่สุดคือ 0.39 เมตรโดยระยะห่างระหว่างสิ่งกีดขวางในเส้นทางที่ 5 จะอยู่ที่ 0.13 เมตร แต่จะเป็นเพียงช่วงสั้นก่อนที่จะมีการเพิ่มระยะห่างทำมุมออกมาอีกในภายหลังจากที่เคลื่อนที่ผ่านจุดดังกล่าวไปได้ 1 วินาทีแล้ว



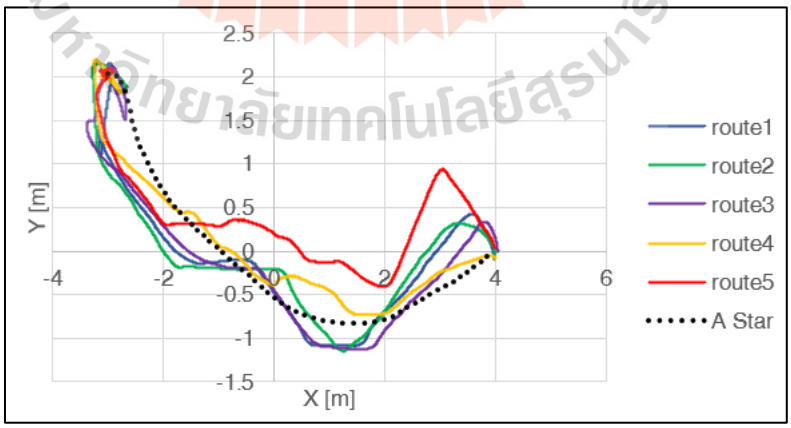
รูปที่ 4.12 พิกัดการเคลื่อนที่ของอากาศยานในสภาพแวดล้อมจริงที่ 1 ที่เกิดขึ้นจริง

ผลที่ได้จากการทดสอบโมเดลการเรียนรู้เชิงลึกจะถูกนำไปเทียบกับเส้นทางที่สร้างขึ้นโดยตัววางแผนแบบ A* (A star path planner) ที่ใช้ในกรณีที่ทราบข้อมูลแผนที่มาก่อน โดยแผนที่ถูกสร้างขึ้นโดยเทคนิค SLAM ซึ่งสำหรับสภาพแวดล้อมจริงที่ 1 รูปภาพของแผนที่ที่บันทึกมาได้แสดงดังในรูปต่อไปนี้



รูปที่ 4.13 แผนที่ที่ถูกสร้างขึ้นของสภาพแวดล้อมจริงที่ 1

เมื่อได้แผนที่แล้ว ตัววางแผนจะได้รับจุดเริ่มต้นและจุดหมายแบบเดียวกับอากาศยาน และจะสามารถสร้างเส้นทางที่เหมาะสมที่สุดออกมาได้ เมื่อพลอตเทียบกับเส้นทางที่อากาศยานใช้เดินทางจริงจะได้ดังในรูป

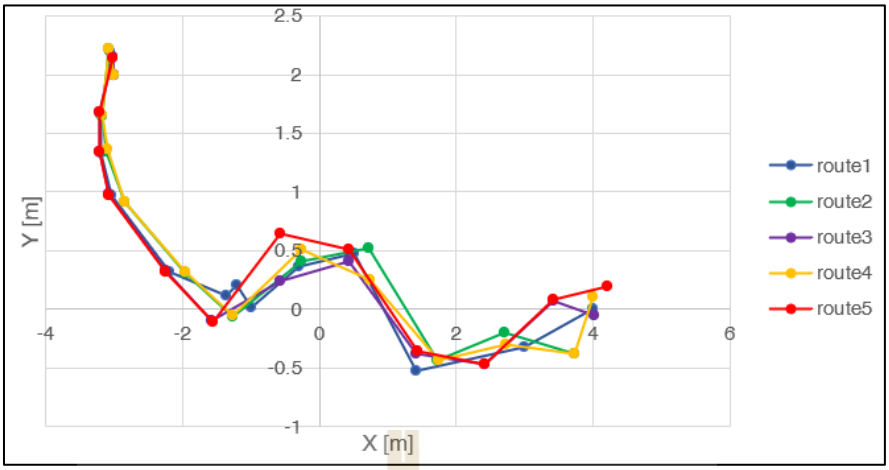


รูปที่ 4.14 เส้นทางจริงของอากาศยานเทียบกับเส้นทางของตัววางแผน A* ในสภาพแวดล้อมจริงที่ 1

จากผลการเปรียบเทียบพบว่า เส้นทางที่สร้างจากตัววางแผนแบบ A* ให้แนวโน้มของเส้นทางใกล้เคียงกับเส้นทางที่ได้จากการตัดสินใจของอากาศยานแต่เส้นทางที่สร้างจากอากาศยานจะใช้ระยะทางที่มากกว่าและมีความรายเรียบ (smooth) น้อยกว่าเส้นทางที่สร้างจากตัววางแผนที่มีข้อมูลของแผนที่อยู่ก่อน ถัดจากสภาพแวดล้อมที่ ไปแล้ว ในสภาพแวดล้อมแบบที่ 2 ตารางข้อมูลการตัดสินใจของตัวแทนในอากาศยานมีการตัดสินใจเคลื่อนที่ไปตามพิกัดดังตารางที่ 4.7 และสามารถนำไปวาดกราฟได้ดังรูปที่ 4.15 โดยในสภาพแวดล้อมที่ 2 จุดเริ่มต้นอยู่ที่พิกัด (-3.0 , 2.0) และจุดหมายปลายทางคือ (4.0, 0.0)

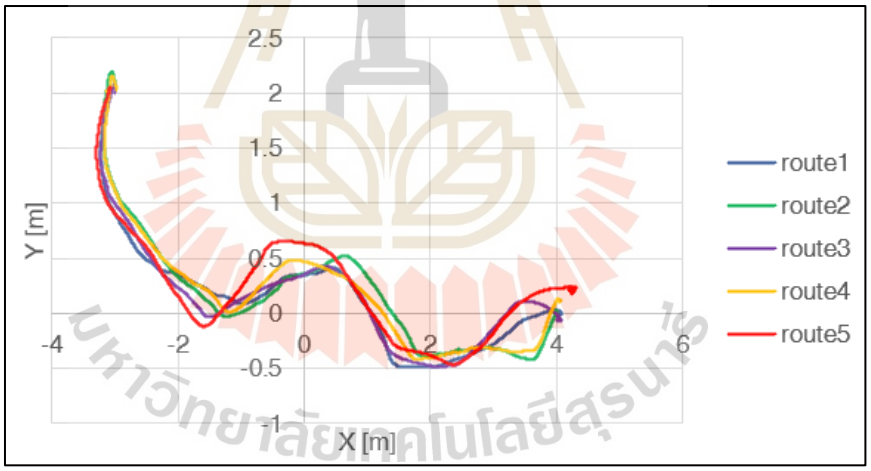
ตารางที่ 4.8 ตารางบันทึกการเลือกเส้นทางของอากาศยานในสภาพแวดล้อมจริงที่ 2

สภาพแวดล้อมจริงที่ 2										
ครั้ง	**1**		**2**		**3**		**4**		**5**	
ลำดับ	**x**	**y**	**x**	**y**	**x**	**y**	**x**	**y**	**x**	**y**
1	-3.00	2.00	-3.00	2.00	-3.00	2.00	-3.00	2.00	-3.00	2.00
2	-3.06	2.19	-3.08	2.22	-3.02	2.15	-3.07	2.22	-3.21	1.68
3	-3.20	1.66	-3.18	1.65	-3.20	1.68	-3.18	1.65	-3.21	1.35
4	-3.19	1.34	-3.11	1.34	-3.21	1.35	-3.10	1.36	-3.07	0.97
5	-3.04	0.97	-2.85	0.91	-3.07	0.98	-2.85	0.92	-2.25	0.32
6	-2.19	0.32	-1.95	0.31	-2.25	0.33	-1.95	0.32	-1.56	-0.11
7	-1.36	0.11	-1.27	-0.06	-1.56	-0.10	-1.26	-0.05	-0.57	0.64
8	-1.21	0.21	-0.27	0.41	-0.58	0.24	-0.26	0.51	0.43	0.51
9	-0.99	0.02	0.73	0.52	0.42	0.41	0.74	0.25	1.43	-0.36
10	-0.30	0.37	1.73	-0.44	1.43	-0.38	1.74	-0.44	2.43	-0.47
11	0.51	0.48	2.72	-0.20	2.43	-0.47	2.73	-0.30	3.43	0.09
12	1.41	-0.52	3.72	-0.37	3.43	0.07	3.73	-0.37	4.22	0.19
13	3.00	-0.33			4.03	-0.06	4.00	0.10		
14	4.00	0.00								



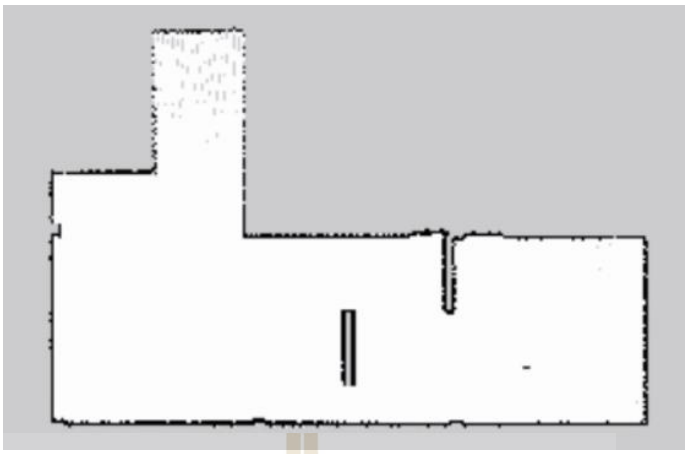
รูปที่ 4.15 พิกัดการเคลื่อนที่ของตัวแทนในสภาพแวดล้อมจริงที่ 2 ที่ถูกตัดสินใจ

เส้นทางการเคลื่อนที่จริงของอากาศยานในสภาพแวดล้อมที่ 2 ในแต่ละครั้งของการทดสอบแสดงดังในรูปต่อไปนี้



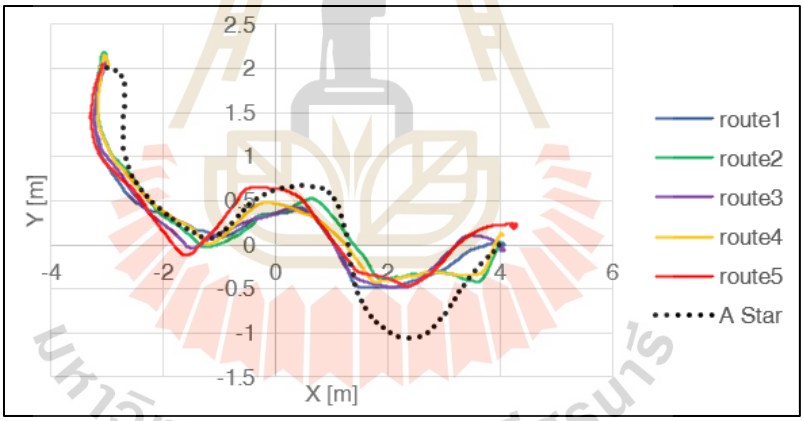
รูปที่ 4.16 พิกัดการเคลื่อนที่ของอากาศยานในสภาพแวดล้อมจริงที่ 2 ที่เกิดขึ้นจริง

เช่นเดียวกับผลการทดสอบในสภาพแวดล้อมที่ 1 เส้นทางของสภาพแวดล้อมที่2จะถูกนำไปเทียบกับตัววางแผน A* โดยแผนที่ถูกสร้างขึ้นโดยเทคนิค SLAM ที่บันทึกมาได้ของสภาพแวดล้อมที่ 2 แสดงดังในรูปต่อไปนี้



รูปที่ 4.17 แผนที่ที่ถูกสร้างขึ้นของสภาพแวดล้อมจริงที่ 2

เมื่อนำแผนที่ที่ได้ ไปสร้างเส้นทางจากจุดเริ่มต้นถึงจุดปลายทางด้วยตัววางแผนแบบ A* แล้วนำมาพลอตเปรียบเทียบกับเส้นทางจริงของอากาศยานจะได้ดังในรูปต่อไปนี้



รูปที่ 4.18 เส้นทางจริงของอากาศยานเทียบกับเส้นทางของตัววางแผน A* ในสภาพแวดล้อมจริงที่ 2

เมื่อเปรียบเทียบผลของการสร้างเส้นทางพบว่าในช่วงเริ่มต้นระยะจาก x เป็น -3.5 ถึง 1.0 เส้นทางที่ได้ของโมเดลมีความใกล้เคียงกับเส้นทางของตัววางแผน A* มากสำหรับการทำการหลบหลีกสิ่งกีดขวางส่วนแรก แต่ในส่วนของการหลบหลีกสิ่งกีดขวางส่วนที่สอง ตัววางแผน A* มีการใช้รัศมีวงโค้งที่มากกว่าเพื่อหลบหลีกโดยมีระยะห่างจากสิ่งกีดขวางประมาณ 1 เมตรซึ่งถึงเป็นระยะที่ค่อนข้างปลอดภัยมาก ส่วนเส้นทางของโมเดลมีระยะห่างจากสิ่งกีดขวางโดยเฉลี่ย 0.45 เมตรซึ่งยังอยู่ในระยะที่ปลอดภัย ระยะห่างระหว่างใบพัดกับกำแพงใกล้สุดคือ 0.19 เมตร ในการเปรียบเทียบเส้นทางที่เกิดจากการตัดสินใจของการเรียนรู้แบบเสริมกำลังเชิงลึกกับเส้นทางที่อ้างอิงจากตัววางแผน A* จะใช้การ

เปรียบเทียบหาค่า RMSE (Root Mean Square Error) โดยตารางค่า RMSE ของเส้นทางในสภาพแวดล้อมทดสอบจริงทั้งสองแสดงดังในตารางที่ 4.8 หากเส้นทางการทดสอบครั้งใดมีการเบี่ยงออกจากเส้นแนวโน้มของเส้นทางที่ได้จากตัววางแผน A* มาก จะมีค่าของ RMSE ที่สูงกว่าเส้นทางที่มีการเบี่ยงเบนน้อยกว่า ในสภาพแวดล้อมที่สองค่า RMSE จะอยู่ในช่วงใกล้เคียงกันแสดงให้เห็นว่าเส้นทางที่ได้จากการตัดสินใจมีความใกล้เคียงกัน ส่วนในสภาพแวดล้อมที่หนึ่งเส้นทางการตัดสินใจของโมเดล DDPG จะมีความใกล้เคียงกันน้อยกว่าสภาพแวดล้อมที่สอง โดยค่า RMSE น้อยที่สุดอยู่ที่ 0.315 ซึ่งเส้นทางที่ได้จะมีแนวโน้มใกล้เคียงกับเส้นทางที่ได้จากตัววางแผน A* มากแต่ในเส้นทางลำดับที่ 5 มีค่า RMSE ที่ 0.635 ซึ่งให้เส้นทางที่ออกห่างจากแนวเส้นทางของ A* มากที่สุดและทำให้การบินมีระยะห่างน้อยกว่าสิ่งกีดขวางน้อยที่สุด

ตารางที่ 4.9 ตารางการเปรียบเทียบค่า RMSE ของเส้นทางการเคลื่อนที่

เส้นทางที่	**สภาพแวดล้อมที่ 1 [m]**	**สภาพแวดล้อมที่ 2 [m]**
1	0.516	0.458
2	0.505	0.586
3	0.447	0.532
4	0.315	0.564
5	0.635	0.568
ค่าเฉลี่ย	0.484	0.542
ค่าสูงสุด	0.635	0.568
ค่าต่ำสุด	0.315	0.458

จากตารางที่ 4.8 ค่าเฉลี่ยของ RMSE ของเส้นทางในสถาพแวดล้อมที่ 1 มีค่าน้อยกว่าสถาพแวดล้อมที่ 2 โดยในสภาพแวดล้อมที่ 2 เส้นทางที่ได้จากตัววางแผน A* ในช่วงการหลบหลีกสิ่งกีดขวางที่สองมีการเว้นระยะห่างจากสิ่งกีดขวางในตอนหลบหลีกออกมาประมาณ 0.48 เมตร แต่เส้นทางที่สร้างออกมาจาก DDPG มีระยะห่างเฉลี่ยที่ประมาณ 0.2 เมตรวัดจากขอบนอกของใบพัด มีค่าแตกต่างกันประมาณ 0.28 เมตรซึ่งคิดเป็น 2.4 เท่า สาเหตุเกิดจากตอนเปลี่ยนผ่านสิ่งกีดขวางที่หนึ่งไปสิ่งกีดขวางที่สองไลดาร์ไม่สามารถฉายเลเซอร์ไปทั่วขอบเขตจริงทั้งหมดของสิ่งกีดขวางได้อย่างสมบูรณ์ โดยส่วนของขอบนอกด้านขวาบางส่วนสามารถเล็ดลอดจากการตรวจจับของไลดาร์ออกไปได้ด้วยมาจากมุมการรับสภาวะของไลดาร์ถูกแบ่งไว้ที่ 15 องศาในแต่ละลำแสง โมเดลตัวแทนจึงไม่ได้ตัดสินใจให้อากาศยานมีการเบี่ยงตัวออกมาอีก หากต้องการแก้ปัญหาส่วนนี้ สามารถทำได้โดยลดมุม

การตรวจจับของสภาวะที่ได้จากไลดาร์ลงมาให้มีความละเอียดมากขึ้น แต่ในงานวิจัยนี้ ด้วยข้อจำกัดด้านกานคำนวณของฮาร์ดแวร์คอมพิวเตอร์จึงไม่สามารถทำได้ และหากมีการใช้คอมพิวเตอร์ที่มีกำลังการประมวลผลสูงขึ้นต้องคำนึงถึงเรื่องของน้ำหนักที่จะเพิ่มขึ้นด้วย หากเพิ่มความสามารถของอุปกรณ์คอมพิวเตอร์และอุปกรณ์ขับเคลื่อนได้จะสามารถทำให้สมรรถนะการทำงานของอากาศยานและโมเดลตัวแทนทำงานได้ดีมากขึ้น

บทที่ 5
สรุปผลและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

วิทยานิพนธ์นี้นำเสนอการพัฒนาอากาศยานไร้คนขับสี่ใบพัดนำทางอัตโนมัติในสภาพแวดล้อมที่ไม่รู้จักด้วยการเรียนรู้แบบเสริมกำลังเชิงลึกด้วยข้อมูลจากไลดาร์ 2 มิติให้อากาศยานเคลื่อนที่ไปยังจุดหมายที่ต้องการได้โดยคงระดับความสูงเอาไว้ อากาศยานต้องสามารถสร้างเส้นทางที่ปลอดภัยไปถึงจุดหมายได้และสามารถทำการบินในอาคารได้ งานวิจัยมีรายละเอียดสรุป ดังนี้

สร้างอากาศยานสี่ใบพัดที่มีเซนเซอร์ไลดาร์ 2 มิติ ติดตั้งอยู่ มีบอร์ดควบคุมการบินและคอมพิวเตอร์ควบคุมการบินทำงานร่วมกันโดยใช้ระบบปฏิบัติการหุ่นยนต์และแพ็คเกจ MAVROS เพื่อสามารถทำให้คอมพิวเตอร์ควบคุมการบินสั่งการการเคลื่อนที่ของอากาศยานให้ไปตำแหน่งที่ระบุได้ในโหมดควบคุมการบินแบบออฟบอร์ดอย่างถูกต้องและปลอดภัย

นำค่าจากเซนเซอร์ไลดาร์และ IMU มาสร้างระบบระบุตำแหน่งและท่าทางของอากาศยานไร้คนขับด้วยอัลกอริทึม RF2O และตัวกรองคาลมานแบบขยาย (Extended Kalman Filter) แล้วนำค่าตำแหน่งที่ได้ส่งให้บอร์ดควบคุมการบินเพื่อระบุตำแหน่งสำหรับการทำงานในโหมดการบินอัตโนมัติ ซึ่งระบบระบุตำแหน่งสามารถทำงานได้อย่างถูกต้องตามแกนพิกัดอ้างอิงและสามารถใช้นำทางในอาคารได้

ฝึกฝนโมเดลการเรียนรู้แบบเสริมกำลังเชิงลึกด้วยโมเดล Deep Deterministic Policy Gradient โดยใช้การรับรู้สภาวะจากไลดาร์ ตำแหน่งของอากาศยานและจุดหมายในสภาพแวดล้อมจำลอง ใช้เวลาในการฝึกโมเดลไปทั้งหมด 172 ชั่วโมง มีรอบการคำนวณทั้งหมดที่ 5324 รอบการคำนวณโดยใช้จำนวนขั้นในการเรียนรู้ไปทั้งหมด 1,764,206 ขั้น ด้วยคะแนนสะสมสูงสุดต่อ 1 รอบการคำนวณเป็น 69,960 คะแนน ตัวโมเดลสามารถตัดสินใจการกระทำเป็นการเคลื่อนที่สัมพัทธ์ของอากาศยานเพื่อไปถึงจุดหมายได้

นำโมเดลที่ได้จากการฝึกฝนไปทดสอบกับสภาพแวดล้อมจำลองทั้งหมด 4 สภาพแวดล้อมที่แตกต่างกันและตัวแทนไม่เคยเห็นมาก่อนจำนวน 5 รอบต่อสภาพแวดล้อม ตัวแทนสามารถหาเส้นทางที่ปลอดภัยจากจุดเริ่มต้นไปยังจุดหมายได้โดยใช้จำนวนขั้นในการตัดสินใจอย่างน้อย 9 ขั้นและสูงสุดที่ 13 ขั้นในพื้นที่ขนาดไม่เกิน 5.0 x 5.0 เมตร โดยเส้นทางในแต่ละรอบมีความใกล้เคียงกัน

นำโมเดลที่ผ่านการทดสอบในสภาพแวดล้อมจำลองไปใช้กับอากาศยานจริงในพื้นที่จริงขนาดกว้าง x ยาว x สูง สูงสุดเป็น 7.96 x 12.07 x 2.5 เมตร ในสภาพแวดล้อมที่ไม่เคยเห็นมาก่อนโดยมี

สิ่งกีดขวาง 2 รูปแบบ อากาศยานสามารถตัดสินใจนำทางตนเองไปถึงจุดหมายได้ด้วยความเร็วสูงสุดไม่เกิน 0.2 เมตรต่อวินาที ความสูง 0.3 เมตรได้อย่างสำเร็จและปลอดภัย โดยระยะทางที่ใกล้ที่สุดระหว่างขอบใบพัดกับสิ่งกีดขวางในสภาพแวดล้อมที่ 1 และสภาพแวดล้อมที่ 2 จะมีค่าเท่ากับ 0.13 เมตรและ 0.19 เมตรตามลำดับ นอกจากนี้เมื่อนำเส้นทางที่ตัดสินใจได้จากโมเดลไปเทียบกับเส้นทางที่หาโดยตัววางแผนแบบ A* ในกรณีที่ทราบแผนที่มาก่อนจะได้เส้นทางที่ใกล้เคียงกันแต่อัลกอริทึม A* จะมีการเพื่อระยะความปลอดภัยระหว่างสิ่งกีดขวางที่มากกว่า โดยค่า RMSE ของเส้นทางที่ได้จากการตัดสินใจของตัวแทนกับเส้นทางจากตัววางแผน A* ในสภาพแวดล้อมที่ 1 มีค่าเฉลี่ยอยู่ที่ 0.484 เมตรและในสภาพแวดล้อมที่ 2 มีค่าเฉลี่ยนอยู่ที่ 0.542 เมตร

5.2 ข้อเสนอแนะ

5.2.1 ในการระบุตำแหน่งสามารถนำเซนเซอร์ระบุตำแหน่งภายนอกเช่นกล้องตรวจจับความเคลื่อนไหว (motion capture) หรือระบบคลื่นความถี่ช่วงกว้างมาระบุตำแหน่งเพิ่มได้

5.2.2 การฝึกโมเดลการการเรียนรู้เสริมกำลังเชิงลึกใช้เวลาในการฝึกเป็นเวลานาน สามารถย่นระยะเวลาได้โดยใช้การฝึกแบบคู่ขนานแล้วนำผลลัพธ์มารวมกัน

5.2.3 โมเดลที่ถูกฝึกในงานวิจัยนี้เหมาะกับพื้นที่ที่ไม่มีความซับซ้อนมาก หากต้องการใช้ในพื้นที่ที่มีความซับซ้อนต้องเพิ่มความละเอียดในการตรวจจับของสภาวะของเซนเซอร์ได้ เช่น เพิ่มความละเอียดของมุมตรวจจับหรือใช้เซนเซอร์ประเภทกล้องเข้ามาตรวจจับสภาวะแทนได้.

5.2.4 สามารถปรับมุมการตรวจจับของไลดาร์ให้แคบลงมามากกว่า 15 องศาได้เพื่อประสิทธิภาพของการตรวจจับที่ดีขึ้น ซึ่งจะทำให้จำนวนสภาวะที่รับสู่โมเดลมีความละเอียดขึ้น ทำให้ประสิทธิภาพในการสร้างเส้นทางการหลบหลีกสิ่งกีดขวางมีความปลอดภัยมากขึ้น แต่ต้องเพิ่มกำลังประมวลผลของคอมพิวเตอร์ซึ่งอาจจะทำให้มีน้ำหนักมากขึ้นได้

รายการอ้างอิง

Alex. (2018). Pixhawk 4, PX4 Autopilot Announced. Retrieved from Dronetrest: https://blog.dronetrest.com/pixhawk-4-announced/

Arun, F. I., Waluyo, P. T., & Siswo, R. A. (2019). A Review of Accelerometer Sensor and Gyroscope Sensor in IMU Sensors on Motion Capture. Journal of Engineering and Applied Sciences.

Bennett, J., Vyhnalek, B., Greenall, H., Bridge, E., Gotardo, F., Forstner, S., . . . Bowen, W. (2021). Precision Magnetometers for Aerospace Applications: A Review. Sensors Volume 21(16):5568.

Camci, E., Campolo, D., & Kayacan, E. (2020). Deep Reinforcement Learning for Motion Planning of Quadrotors Using Raw Depth Images. Proceeding of 2020 International Joint Conference on Neural Networks (IJCNN), (pp. 1-7).

Centeye. (2023). Introduction to optical flow. Retrieved from Centeye: https://www.centeye.com/technology/optical-flow/

Cetin, E., Barrado, D., Munoz, G., Macias, M., & Pastor, E. (2019). Drone Navigation and Avoidance of Obstacles Through Deep Reinforcement Learning. Proceeding of 2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC), (pp. 1-7).

Chan, C. W., & Kam, T. Y. (2020). A procedure for power consumption estimation of multirotor unmanned aerial vehicle. Proceeding of of Physics Conference Series vol. 1509.

Chen, Y., González-Prelcic, N., & Heath, R. W. (2020). Collision-Free UAV Navigation with a Monocular Camera Using Deep Reinforcement Learning. Proceeding of 2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP), (pp. 1-6).

Chibunichev, A. G., Makarov, A. P., & Poliakova, E. V. (2020). Using A Low-Cost Stereo Camera For Autonomous Navigation Of Mobile Robot. Int. Arch. Photogram Remote Sens. Spatial Inf. Sci., XLIII-B1-2020, 423-427.

Eness, B. (2018). Mastering Reinforcement Learning with Python. UK: Packt.

Grando, R. B., Jesus, J. C., & Drews-Jr, L. J. (2020). Deep Reinforcement Learning for Mapless Navigation of Unmanned Aerial Vehicles. Proceeding of 2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), (pp. 1-6).

IntelRealsense. (2019). What is dept camera. Retrieved from IntelRealsense: https://www.intelrealsense.com/beginners-guide-to-depth/

Jaimez, M., Monroy, J. G., & Gonzalez-Jimenez, J. (2016). Planar odometry from a radial laser scanner. A range flow-based approach. IEEE International Conference on Robotics and Automation (ICRA), (pp. 4479-4485). Stockholm. Sweden.

Maria, R. I., & Isabel, R. (2004). Kalman and Extended Kalman Filters: Concept, Derivation . Retrieved from Researchgate: https://www. researchgate.net/ publication/ 288 8846_

Maxim, L. (2020). Deep Reinforcement Learning Hands-On Second Edition. UK: Packt.

Nwaonumah, E., & Samanta, B. (2020). Deep Reinforcement Learning For Visual Navigation of Wheeled Mobile Robots. Proceeding of 2020 SoutheastCon, (pp. 1-8).

Pham, H. X., La, H. M., Seifer, D. F., & Van, N. (2018). Reinforcement Learning for Autonomous UAV Navigation Using Function Approximation. Proceeding of 2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), (pp. 1-6).

Phongchit, N. (2018, September 19). Convolutional Neural Network (CNN). Retrieved from Medium: https://medium.com/@natthawatphongchit/

PX4. (2021, July 15). Telemetry. Retrieved from PX4: https://docs.px4.io/v1.9.0/en/telemetry/

PX4Docs. (2021, July 15). PX4Docs. Retrieved from PX4: https://docs.px4.io/master/en/flight_controller/pixhawk4.html

Quan, Q., Xunhua, D., & Shuai, W. (2020). Multicopter Design and Control Practice. Singapore: Springer.

Ravichandiran, S. (2020). Deep Reinforcement Learning with Python, Second Edition. UK: Packt.

Rubi, B., Morcego, B., & Perez, R. (2020). A Deep Reinforcement Learning Approach for Path Following on a Quadrotor. Proceeding of 2020 European Control Conference (ECC), (pp. 1092-1098).

Shin, S., Kang, Y., & Kim, Y. (2019). Automatic Drone Navigation in Realistic 3D Landscapes using Deep Reinforcement Learning. Proceeding of 2019 6th International Conference on Control, Decision and Information Technologies (CoDIT), (pp. 1072-1077).

Stable flying and hovering drone. (2020). Retrieved from Ingenieerburo: https://www.technik-consulting.eu/en/analysis/stable_drone.html

Tran, M. Q., & Ly, N. Q. (2020). Mobile Robot Planner with Low-cost Cameras Using Deep Reinforcement Learning. Proceeding of 2020 7th NAFOSTED Conference on Information and Computer Science (NICS), (pp. 54-59).

Winslow, J. M., Hrishikeshavan, V., & Chopra, I. (2017). Design Methodology for Small-Scale. Journal of Aircraft Volume 55, 1062-1070.

Xiang, J., Li, Q., Dong, X., & Ren, Z. (2019). Continuous Control with Deep Reinforcement Learning for Mobile Robot Navigation. Proceeding of 2019 Chinese Automation Congress (CAC), (pp. 1501-1506).

Yue, P., Xin, J., Zhao, H., Liu, D., Shan, M., & Zhang, J. (2019). Experimental Research on Deep Reinforcement Learning in Autonomous navigation of Mobile Robot. Proceeding of 2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA), (pp. 1612-1616).

Zhang, S., Lu, S., He, R., & Bao, Z. (2021). Stereo Visual Odometry Pose Correction through Unsupervised Deep Learning. Sensors, 21(14):4735.

ภาคผนวก ก

รายชื่อบทความที่ได้รับการเผยแพร่ระหว่างการศึกษา

รายชื่อบทความที่ได้รับเผยแพร่ระหว่างการศึกษา

P. Phueakthong and J. Varagul. (2021). A Development of Mobile Robot Based on ROS2 for Navigation Application. Proceeding of 2021 International Electronics Symposium (IES), Surabaya, Indonesia, 2021, pp. 517-520, doi: 10.1109/IES53407.2021.9593984.

P. Phueakthong, J. Varagul and N. Pinrath. (2022). Deep Reinforcement Learning Based Mobile Robot Navigation in Unknown Environment with Continuous Action Space. Proceeding of 2022 5th International Conference on Intelligent Autonomous Systems, Dalian, China, 2022, pp. 154-158, doi: 10.1109/ICoIAS56028.2022.9931272.

P. Phueakthong, J. Varagul and N. Pinrath. (2023). Mobile Robot Indoor Localization Based on the Fusion of UWB with Laser Odometry and IMU Sensor, 2023 The Annual Conference on Engineering and Information Technology (ACEAIT2023), Osaka, Japan, 2023, pp. 335-344.

Saw Yi Wan Yan, K. Chamniprasart, P. Phueakthong and N. Pinrath. (2023). Autonomous Mobile Robot for Material Handling in an Industrial Plant. Proceeding of 2023 The Annual Conference on Engineering and Information Technology (ACEAIT2023), Osaka, Japan, 2023, pp. 368-379.

P. Singcharoenkit, P. Phueakthong, A. Lonklang, J. Varagul and K. Chamniprasart. (2022). An Implementation of Object Tracking Methods on Pan and Tilt Manipulator for Teacher Tracking in Hybrid Classroom. Proceeding of 2021 WRFER International Conference, Phuket, Thailand, 2022, pp. 41-44.

2021 International Electronics Symposium (IES)

A Development of Mobile Robot Based on ROS2 for Navigation Application

Phuwanat Phueakthong
School of Mechatronics Engineering
Suranaree University of Technology
Nakhon Ratchasima 30000, Thailand
Email: phuwanat.aerod@gmail.com

Jittima Varagul
School of Manufacturing Engineering
Suranaree University of Technology
Nakhon Ratchasima 30000, Thailand
Email: jittima@sut.ac.th

2021 International Electronics Symposium (IES) | 978-1-6654-4346-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/IES53407.2021.9593984

***Abstract*—This paper proposes an automatic navigation mobile robot using Robot Operating System2 (ROS2) with low-cost embedded hardware. Utilizing Data Distribution Service (DDS) in ROS2 makes the ROS2 more safe and reliable than ROS1. Cartographer and Navigation2 projects in ROS2 are used for Simultaneous Localization and Mapping (SLAM) with 2D LIDAR and navigation, respectively. Micro-ROS which utilizes DDS for eXtremely Resource-Constrained Environments micro-XRCE-DDS is used for communication between main embedded computer and microcontroller replaces ROS serial communication which is less reliable. The experiments prove that the robot can perform mapping and navigation tasks. A robot can generate a global trajectory in a static map to the goal point, can re-plan the local path in the local map area to avoid coming dynamic obstacles during the mission and navigate itself to reach the goal.**

***Keywords*—ROS2; navigation; autonomous mobile robot; micro-ROS; SLAM**

I. Introduction

Nowadays, mobile robot with autonomous navigation system is used in various works, such as delivery robot in medical work and in industrial factory. Robot Operating System (ROS) is a popular framework that is used in these clever robots. ROS provides useful packages for autonomous robot, such as mapping, localization and navigation package, also provides useful tools for robot development. Using ROS simply the autonomous robot invention. Recently, ROS has been improved to the new version which is called ROS2 and the original ROS version is called ROS1. ROS2 is designed to support real-time robotic systems and, increase reliability and enhance safe data communication [1]. These results come from utilizing Data Distribution Service (DDS) which is an industrial-standard communication middleware [2], in ROS2.

For automate driving, ROS2 provides tools for developed and evaluate performances of the autonomous mobile robot, for example, remote control, remote monitoring tools and simulations. The important part of autonomous navigation robot is Simultaneous Localization and Mapping (SLAM) which obtains data from sensors, such as LIDAR laser scan data, sonar distance and depth cameras data to build a map and localize the robot in the map. In this research, Cartographer is used to generating a 2D grid map for robot navigation.

In this essay, we have invented a differential drive mobile robot with cheap hardware. The robot has Raspberry PI4 RAM 4 GB as the main processor, YD Lidar X4 Laser scanner as a data collector, Motor with encoder and use Raspberry Pi Pico RP2040 microcontroller as motor controller unit.

This article is structured as follows. Section II explains methodology which is the details of design, system organization and software components. Experiments, results and analysis are in section III followed by the final conclusion in section IV.

II. Methodology

A. Mechanical Design and hardware architecture

The robot is divided into three layers to contains all components. The top plate supports LIDAR. The mid-plate and bottom plate support Raspberry Pi4 computer and Pi Pico with motor driver board, respectively. The design from Fusion360 software is in Figure 1.

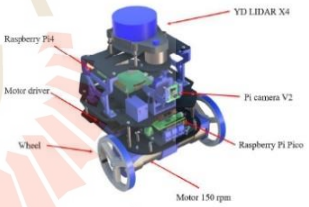


Figure 1. A design of robot.

The selected motor is 150 rpm without load, torque 1.8 kgf-cm and there is quadratic encoder built-in with it. It is used with a wheel which has diameter of 85.1 millimeters. Both wheels are attached at the front of the robot and there is a caster ball locate at the backward of a robot. The Raspberry Pi camera v2.0 is installed in the front for surveying.

B. Elecronic devices integration

Electronic devices consist of the computer, sensors, microcontroller and motors of robot. Details of these devices

are in the following. Figure 2 illustrated the hardware architecture of the robot.

The computer used is Raspberry Pi4 RAM4GB with 64GB micro SD-card. It has 40 GPIO pinouts. It is the main processor for robot. LIDAR, Pi camera V2.0 and Raspberry Pi Pico microcontroller connect with it.

The microcontroller used is Raspberry Pi Pico with RP2040 chip. It has dual cores ARM Cortex-M0+ processor with 264KB on-chip SRAM, 2MB onboard QSPI flash and 16 PWN channels. Pico waits for command from Raspberry Pi to control speed of motors and wait for pulse signal from wheel encoders then publish it to Raspberry Pi.

The LIDAR used is YD Lidar X4 which has minimum and maximum ranges are 0.12 to 10 meters. The scan frequency is 6-12 Hz. and 360 degrees scan angle. YD Lidar X4 provides a separate data transfer and power supply USB port which allows users can power the LIDAR with an external power source separately from Raspberry Pi. YD LIDAR X4 requires 12800 bps communication baud rates to work properly.

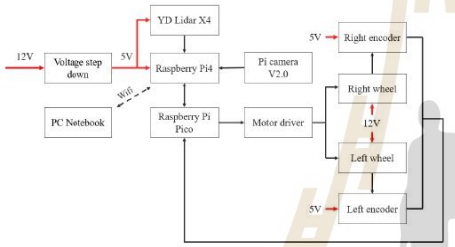


Figure 2. Hardware architecture.

The motor requires 12V power and a built-in PIC16F quadrature encoder which requires 5V power supply. Motor's rated load torque and rated load speed are 1.8 kgf-cm and 112 ± 12 rpm. Gear ratio is 30:1. Motor's power supply is from a 3S LiPo battery.

C. *Robot Operating System2 (ROS2)*

Robot Operating System 2 (ROS) is the second generation Of ROS. ROS2 builds upon Data Distribution Service (DDS) [3]. ROS2 is cleaner than the older version [4]. DDS provides distributed discovery feature (not centralized in ROS1) that allows each ROS2 node can discover each other without ROS master and DDS make ROS2 support real-time operation. [3] propose comparative data about capabilities and performance of ROS2 over ROS1, such as data transmission, supported platforms, Quality of Service (QoS), number of threads and real-time characteristics. ROS2 provides navigation2 and cartographer package which is relevant packages for mobile robot navigation.

ROS navigation2 can be applied in the autonomous navigation mobile robot to find safe way from A to point B. It consists of package for localization, path planning, dynamic obstacle avoidance and etc. Require input for navigation2 are robot's TF (transform) which explains the relative of the reference frame of robot, robot odometry data, sensor data source and map. Then, it will send out the motor's velocity command to control robot [5].

ROS2 used is ROS Foxy Fitzroy which is compatible with Ubuntu 20.04. ROS2 will be installed on both PC (Ubuntu 20.04) and Raspberry Pi4 (Ubuntu mate 20.04).

D. *Micro-ROS (ROS for micro controllers)*

Micro-ROS is ROS for microcontroller. Micro-ROS allows microcontroller can use all major core concepts of ROS with C or C++ programming. Client API of micro-ROS in MCU based on ROS2 client library [6].

Even DDS implementation is lightweight, but the memory footprint is still large too much to bear in embedded systems [9]. This problem leads micro-ROS to use micro-XRCE-DDS (DDS for eXtremely Resource-Constrained Environments) middleware by eProsima which is a middleware for embedded systems. It supports WiFi, 6LoWPAN, Bluetooth, serial transport and UDP communications. Micro-ROS is supported by RTOSes, FreeRTOS, Zephyr and NuttX. In this paper, micro-ROS will be added to Raspberry Pi Pico to create micro-ROS node in Pico which waits for the command from Raspberry Pi to control speed of motor and obtain feedback from encoders.

E. *Mapping with Cartographer*

Cartographer is an open-source package for real-time simultaneous localization and mapping (SLAM) from Google for various sensor configuration and platforms in 2D and 3D [7]. Cartographer uses graph optimization algorithm which use lower computing resource than particle filter method [8].

F. *AMCL and Navigation*

ROS2 navigation stack provides AMCL (Adaptive Monte Carlo Localization) package for localization which use map data, sensors data and odometry data from robot with particle filters, Monte Carlo Localization (MCL) and Kullback-Leibler Distance (KLD) sampling method to estimate pose of robot in an environment [10]. The number of samples will be chosen by KLD using consideration of uncertainty [11]. The number of samples influences the efficiency of particle filters. AMCL will scatter particles randomly in map. When the robot moved, the particles will move together. The particle which matched the sensor data will obtain the higher weight than others. The process occurs repeatedly and the particles will gradually converge to the actual robot position over time [12].

In this paper, grid-based map is used to represent or model the environment of robot. Map data is a static map. It will be taken by path planner to generate trajectory for robot. ROS2 divides path planner into global path planner and local path planner. The global planner will generate path for known obstacles over a map. Nav2Fn is a plugin for global planner in ROS 2 which use A* or Dijkstra's algorithm. The selected algorithm is A* algorithm which is more efficient algorithm for finding the shortest path than Dijkstra's algorithm [13]. In

local path planning, local planner is used to avoiding dynamic obstacles which there is no on map. The selected local planner is DWB local planner which is the upgraded version of DWA local planner in ROS1.

III. Experiment and Results

A completed robot for testing in shown in Figure3. Laboratory area is testing environment. Testing are map generation test and navigation test.

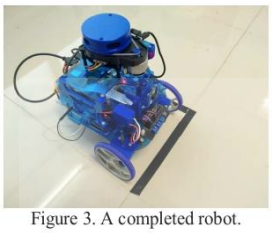


Figure 3. A completed robot.

A. Map generation

A robot was controlled by teleoperation from PC to explore the area. Cartographer used odometry data, and laser scan data as input for map generation. After process, result map is shown in Figure 4. The black area is walls and white area is space. Map size is 448×343 pixels. It consist of, .pgm and .yaml file which are the map picture and map description file, respectively.

Figure 4. Map from exploration.

B. Navigation test

We divided navigation test into two parts, navigation in only static map test and navigation in static map with unknown or dynamic obstacle. During navigation, Rviz2, a software for 3D data visualization in ROS2, was used to visualize the data and send navigation goal to a robot. When navigation2 is activated, a map which obtained in prior was called and used to create global cost map and local cost map as shown in Figure 5. Planner in ROS2 uses global cost map to plan trajectory of a robot until the end of operation and uses local cost map to plan trajectory of a robot to avoid coming obstacle. Local cost map is only a small area around a robot. We used only 3×3 meters local cost map. Selected position tolerance of navigation goal in Navigation2 controller parameters of x and y coordinates is 25 centimeters and a robot footprint radius is 15 centimeters

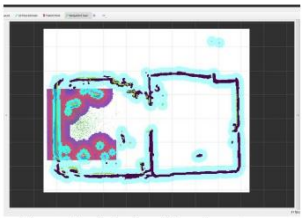
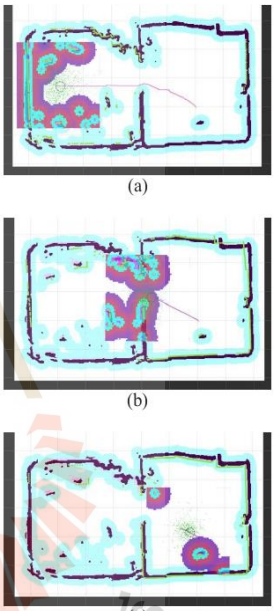


Figure 5. global and local cost map.

We set start point of the robot in boundary as in Figure 3 and sent navigation goal to a robot. This point is the same for all test cases. After it obtained a goal point, it generated a global path as in Figure 6 and started moving.

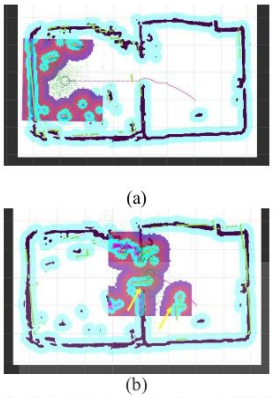


(a) (b) (c)

Figure 6. (a) A robot start moving. (b) A robot followed path. (c) A robot approached to destination.

Above figures illustrate a moving of robot follow a path from global planner without unknown obstacle. When we added new unknown obstacles to the map, a local planner plays an important role. When a robot detected unknown obstacles. Local planner tried to generate new trajectory in local area to avoid obstacles as in Figure 7. Compared to Figure 6(b) the

yellow arrow in Figure 7(b) points to new added dynamic obstacles.



(a)

(b)

Figure 7. (a) A robot started moving to goal. (b) A robot tried to avoid unknown obstacles.

From 5 times testing in both navigation case, the results of position and travel time of a robot in each navigation case are shown in Table I and Table II.

TABLE I. POSITION ERROR OF GOAL AND MOVING TIMES OF ROBOT IN STATIC MAP WITHOUT UNKNOWN OBSTACLES.

Experiment	Position error		Times [s]
	x [cm]	*y [cm]*	
1	8.30	-2.54	14.63
2	0.72	-3.39	14.42
3	11.88	-14.01	14.60
4	5.42	-7.01	15.03
5	8.45	-12.17	14.93
average	6.96	-7.82	14.72

TABLE II. POSITION ERROR OF GOAL AND MOVING TIMES OF ROBOT IN STATIC MAP WITH UNKNOWN OBSTACLES.

Experiment	Position error		Times [s]
	x [cm]	*y [cm]*	
1	1.39	4.57	15.62
2	6.79	6.91	15.77
3	5.97	14.99	15.99
4	2.60	9.85	15.59
5	5.21	11.80	16.09
average	4.392	9.62	15.81

A robot can automatically navigate itself to destination without collision. A robot can detect unknown obstacles and re-plan the trajectory in local cost map to avoid all obstacles. With the dynamic obstacles, a robot took a longer time to reach the destination. A maximum of goal position tolerances is 14.99 centimeters in Table II which is in range of defined x and y goal tolerances.

IV. CONCLUSSION

In this article, a design, software system of an autonomous navigation robot based on ROS2 with low-cost embedded hardware is presented. ROS2 and micro-ROS is implemented in embedded computer and microcontroller of robot. A vehicle is capable of Teleoperation with command from PC, Mapping with Cartographer, Localizing with AMCL and Navigation with Navigation2 project. Selected algorithm for global path planning is A* algorithm and selected planner for local path planner is DWB planner. A robot can navigate to the goal, avoid the dynamic obstacle in a map and reached the goal point successfully with acceptable x and y position tolerances. The maximum tolerance of goal position is 14.99 centimeters.

REFERENCES

[1] Y. Tang et al., "Response Time Analysis and Priority Assignment of Processing Chains on ROS2 Executors," IEEE Real-Time Systems Symposium (RTSS), pp. 231-243, 2020.

[2] H. Cui, J. Zhang and W. R. Norris, "An Enhanced Safe and Reliable Autonomous Driving Platform using ROS2," IEEE International Conference on Mechatronics and Automation (ICMA), pp. 290-295, 2020.

[3] Y. Maruyama, S. Kato and T. Azumi, "Exploring the performance of ROS2," International Conference on Embedded Software (EMSOFT), pp. 1-10, 2016.

[4] C. Heggem, N. M. Wahl and L. Tingelstad, "Configuration and Control of KMR iiwa Mobile Robots using ROS2," 3rd International Symposium on Small-scale Intelligent Manufacturing Systems (SIMS), pp. 1-6, 2020.

[5] S. Macenski, F. Martín, R. White and J. G. Clavero, "The Marathon 2: A Navigation System," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2718-2725, 2020.

[6] Features and Architecture of Micro-ROS, "micro.ros.org" [Online]. Available: https://micro.ros.org//docs/overview/features/

[7] Cartographer with ROS Integration, "https://google-cartographer-ros.readthedocs.io/en/latest/."

[8] X. Zhang, J. Lai, D. Xu, H. Li, and M. Fu, "2D Lidar-Based SLAM and Path Planning for Indoor Rescue Using Mobile Robots," Journal of Advanced Transportation, 2020.

[9] S. Dehnavi, D. Goswami, M. Koedam, A. Nelson and K. Goossens, "Modeling, implementation, and analysis of XRCE-DDS applications in distributed multi-processor real-time embedded systems," 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1148-1151, 2021.

[10] I. Wasisto, N. Istiqomah, I. K. N. Trisnawan and A. N. Jati, "Implementation of Mobile Sensor Navigation System Based on Adaptive Monte Carlo Localization," 2019 International Conference on Computer, Control, Informatics and its Applications (IC3INA), 2019, pp. 187-192

[11] R. Mishra and A. Javed, "ROS based service robot platform," 4th International Conference on Control, Automation and Robotics (ICCAR), pp. 55-59, 2018.

[12] L. Zhi and M. Xuesong, "Navigation and Control System of Mobile Robot Based on ROS," IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), pp. 368-372, 2018.

[13] Z. Li, Y. Xiong and L. Zhou, "ROS-Based Indoor Autonomous Exploration and Navigation Wheelchair," 10th International Symposium on Computational Intelligence and Design (ISCID), 2017, pp. 132-135, 2017.

The 5th International Conference on Intelligent Autonomous Systems

Deep Reinforcement Learning Based Mobile Robot Navigation in Unknown Environment with Continuous Action Space

Phuwanat Phueakthong
School of Mechatronics Engineering
Suranaree University of Technology
Nakhon Ratchasima, Thailand
phuwanat.aerod@gmail.com

Jittima Varagul
School of Manufacturing Automation and Robotics Engineering
Suranaree University of Technology
Nakhon Ratchasima, Thailand
jittima@sut.ac.th

Nattawat Pinrath
School of Industrial Engineering
Suranaree University of Technology
Nakhon Ratchasima, Thailand
nattawat.p@g.sut.ac.th

2022 5th International Conference on Intelligent Autonomous Systems (ICoIAS) | 978-1-6654-9838-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICOIAS56028.2022.9931272

***Abstract*—This work aims to propose the use of deep reinforcement learning for mobile robot navigation and obstacle avoidance in previously unknown areas or without pre-made maps with continuous action control to increase the capabilities of mobile robots beyond conventional map-based navigation. Deep reinforcement learning is used to enable the robot to learn how to make decisions and interact with the environment observed from sensor data to safely navigate itself to its destination. The robot has a two-dimensional laser scanner, ultrasonic sensors and odometry sensor. Deep Deterministic Policy Gradient, which can function in continuous action space, was chosen as the deep reinforcement learning model. The robot is trained and tested in a Gazebo simulator with Robot Operating System. After the training process, the robot is put to the challenge to complete a waypoint navigation mission in four unknown areas as part of an assessment. The results indicate that the mobile robot is adaptable and has the capability of traveling to the specified waypoints and completing the job without the need for a pre-drawn route or an obstacle map in unidentified environments with the minimum success rate of 69.7 percent.**

***Keywords*—deep reinforcement learning; autonomous navigation; DDPG**

I. Introduction

Navigation is one of the key abilities of autonomous mobile robot; it enables the robots to operate in an unknown or partially known environment; where the robot can react to static obstacles or unpredictable dynamic events [1]. The general aim of navigation is to identify the proper path from the starting point to the target point by avoiding obstacles. Robot navigation is a complex technological problem.

In the past two decades, there are many proposed solutions and techniques, One of famous approach is Simultaneous Localization and Mapping (SLAM) is an algorithm process of a robot, which involves perceiving the environment using sensors and estimating the position of itself in the environment simultaneously [2]. However, traditional map-based navigation frameworks tend to have low computational problems due to the large number of computational errors. Additionally, the navigation performance of traditional map-based navigation frameworks is dependent on the quality of the world map. which is very sensitive to sensor interference This requirement may limit the navigation system's ability to handle unknown environments[3].

In the past few years, the rapid rise of deep learning and deep reinforcement technologies has resulted in new ideas for implementing deep learning or deep reinforcement learning frameworks that can learn navigation strategies directly from raw sensor inputs. In [4-6] proposed the navigation system by vision-based for mobile robots by applied deep CNN models. The results show the navigation system is able to navigate the robot in the unknown environments. In addition, the Laser-based approach uses laser sensors to obtain obstacle data then used deep reinforcement such as Deep Q-Network (DQN) to learning method into the robot path planning. In [7] proposed navigation system based on DQN reinforcement learning in maze environment. The environment data obtained by LiDAR sensor and defined one of the three states: move forward, turn left and turn right possible motion the result shown the mobile robot can move towards its target without colliding with obstacles in unknown environments same as research can also be found in the references [8-10]. The DQN reinforcement is a technique designed for discrete action space Therefore, DQN encounters problems when the action space is continuous. One of solution is discretizing the action space. This may work in some situation but cannot bring out the ideal solution. Therefore, in this experiment we presented Deep Deterministic Policy Gradient (DDPG) with mobile robot navigation in unknown environment. The experimental results show that the DDPG can make the robot complete the navigation task without a prepared map.

II. Problem Formulation

Problem-solving with DRL requires the problem set which can work in a reinforcement learning framework. Three important components must be specified are state space, action space and rewards. In this study, state and action referred to limitations of the real robot. More descriptions of state, action and reward specifications are in the following.

A. State Space

In this study, the states of mobile robot were specified from movement and sensor capabilities of the robot in Fig.1. The robot is a differential drive model which uses two wheels and motors to drive itself. Each motor has a quadratic encoder

which can be the source of position and orientation data of the robot. There is a 2D laser scanner inside the body to measure the range of obstacles around the robot. In front of the robot, there were three ultrasonic range sensors to detect some obstacles which out of the detection zone of laser scanner.

Figure 1. Real robot model (Lapras robot) with all sensors, devices and parts.

The states were represented by the laser ranges (S_{laser}), ultrasonic ranges (S_{ul}) robot's heading (S_h) and Euclidean distance from the robot to destination (S_d). The laser ranges were discretized to 24 ranges of data to reduce the computational cost. The total number of states were 29 which consists of 24 ranges data, 3 ultrasonic range data and 2 relative pose data. All states were observed in every step of learning in the DRL algorithm.

B. Action Space

When we control a robot in real world, almost speed command is continuous value. In order to achieve better performance of the robot while navigation and smooth behavior, the continuous action space was selected to control the robot in this research. The action space consists of the linear velocity and angular velocity of the robot. The limit of the linear velocity was [-0.8,0.8] meters per second and the limit of angular velocity was [-0.5,0.5] radian per second. The outputs from the neural network will be restricted in both ranges and sent to control the robot's movement.

C. Reward Function

In deep reinforcement learning, the reward tells the agent that the action a_t gives good or bad results when it is in state s_t. Definition of the reward $r_t(s_t, a_t)$ function has an effect on the performance of learning. The better reward function can converge the policy rapidly. When the distance between the robot and the goal point decreased, the reward of 1 score was given but in opposite directions, the negative reward of -1 score was given. When the robot collides with the wall, the agent will obtain the penalty of -200 score but when the robot can reach the goal, the reward of 200 score was given. In addition, if the robot can maintain itself far from the wall more than safety margin (d_{sm}), the agent will obtain the reward due to the minimum distance from the wall measured by the laser scanner. The summary of reward setting are as follows:

$$r_{target} = \begin{cases} 1 & (d_p - d_c) > 0 \\ -1 & (d_p - d_c) < 0 \end{cases} \quad (1)$$

$$r_{safety} = c_2 \cdot (min(ranges) - d_{sm}) \quad (2)$$

$$r_{collision} = \begin{cases} -200 & collision \\ 200 & goal \end{cases} \quad (3)$$

The value of c_2 is a real number reward factor. The target of the agent is maximizing these rewards and learns the optimal policy.

III. Algorithm

A. Deep Deterministic Policy Gradient (DDPG)

In this study, the autonomous agent is trained using Deep Deterministic Policy Gradient (DDPG) proposed in [11]. DDPG is an off-policy, model-free algorithm that is designed for the environment with continuous action space. DDPG adopts actor critic algorithm based on Deterministic Policy Gradients (DPG) and deep Q networks to solve continuous control tasks . The actor of DDPG uses a policy gradient to estimate the policy of the agent and the critic uses Q function. The networks for actor and critic are policy network and deep Q network, respectively. Both actor and critic networks consist of their target network. DDPG actor learns deterministic policy (μ) instead of a stochastic policy (π) like other actor-critic algorithms. The deterministic policy maps the state to one action and tells the robot how to move. After the actor network select an action, the critic network will evaluate an action with deep Q learning.

DDPG inherits the idea of using an experience replay buffer and using target network from DQN variants and adapted it to the actor critic framework. The replay buffer uses to store states, actions, rewards, and all transitions. During the learning process, the data from replay buffer is sampled to train the network. It can reduce the instability of training due to the correlation presented in the sequence of the state observation . Using the target network is proposed in Double Deep Q Network (DDQN) , which gives a better result than typical DQN. The main steps of DDPG for robot navigation in this research are explain as follows:

The first step, when the robot observed state information (s_t) from environment, the actor network receives the states and returns the action (a_t) from parameterized policy (μ_ϕ). Before action will be sent to the robot, in continuous action space, it is necessary to add noise ($\mathcal{N}$) to action produced by actor network to solve the exploration-exploitation dilemma and prevent that a deterministic policy always selecting the same action and hard to explore new actions. The popular noise generator is Ornstein-Uhlenbeck random process [16]. With additional noise, the policy can explore new actions like stochastic policy. The modified action can be represented by the equation below.

$$a_t = \mu_\phi(s_t) + \mathcal{N}_t \quad (4)$$

After that, the new actions are converted into linear and angular velocity and the commands are sent to the robot speed control node. When the robot transits to new states, the rewards and information of the next state are observed and the

tuple of (s_t, a_t, r_t, s_{t+1}) is collected to the experience replay buffer.

Next, A minibatch of K transitions is extracted randomly and put into the network to learn and update the policy. Extracted sample is input for the critic network to calculate the state-action value (Q). The critic network's parameters (θ) are optimized and updated by the minimization of mean squared loss between target Q value and predicted Q value as following loss function J:

$$J(\theta) = \frac{1}{K}\sum_{i=1}^{K}(y_i - Q_\theta(s_i, a_i))^2 \quad (5)$$

where:

$$y_i = r_i + \gamma Q_{\theta'}(s_{i+1}, \mu_{\phi'}(s_{i+1})) \quad (6)$$

The discount factor γ usually close to 0.99. The update equation for parameters of main critic network by performing gradient descent is:

$$\theta \leftarrow \theta - \alpha \nabla_\theta J(\theta) \quad (7)$$

Where α is learning rate of gradient descent. Then, update the actor network using deterministic policy gradient:

$$\nabla_\phi J \approx \frac{1}{K}\sum_i \nabla_a Q_\theta(s,a)\,|_{s=s_i, a=\mu(s_i)} \nabla_\phi \mu_\phi(s)|_{s_i} \quad (8)$$

Next, update target actor network parameter (ϕ') and target critic network parameter (θ'). The parameter of target critic network is updated by a method called soft replacement as in the equation (9) which soft replacement parameter (τ) is much less-than 1($\tau \ll 1$). This means that target value is delayed for changing which improves stability of learning [11].

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta' \quad (9)$$

Similar to target critic network, the target actor network's parameter (ϕ') can be updated by soft replacement as in equation below.

$$\phi' \leftarrow \tau\phi + (1-\tau)\phi' \quad (10)$$

All processes work repeatedly until the end of episode. The episode ends when the robot collides with the obstacle, or the time is up. After one episode end, new episode begins again until the number of episodes reaches the defined maximum episodes.

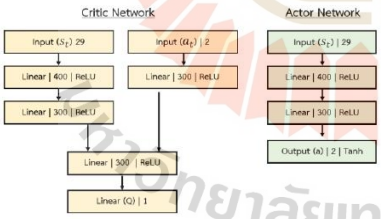


Figure 2. Network architectures of (left) critic and (right) actor networks

B. Network Architectures

In this study, the DDPG network consists of two actor networks (main and target network) and two critic networks. All 29 inputs of state are passed to the model. The proposed architecture of the networks is shown in the Fig. 2.

In the critic network, the input state is connected to two hidden fully connected layers with 400 and 300 neurons, respectively. The input action is connected to 300 neurons fully connected layer and it is concatenated with the previous layer before the estimation of Q value. The input action of the critic network is generated from the actor network with two hidden fully connected layers. The output from actor network is processed by Tanh activation function. Outputs are converted to linear and velocity of the robot.

IV. Experiment and Result

In this paper, the training and evaluation processes were performed in simulated environments. The selected simulator was the Gazebo simulator which fully supports programming with Robot Operating System (ROS). The robot model that was created refers to the real robot in Fig.1 with all the sensor plugins. The agent was trained in one training environment but was evaluated in four different unknown fields.

A. Training Process

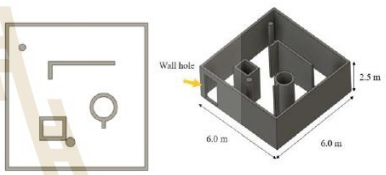


Figure 3. The 3D model and simple dimension of stage for the training process

The training stage is a 6.0 m × 6.0 m × 2.5 m room as shown in Fig. 3. There was the various shapes of obstacles in the area. There was a small wall hole in a side of the area to disturb observations by laser scanner .The robot model is in Fig. 4. The yellow part is the reference object for the ultrasonic range sensor. The laser scanner is a blue part inside the body of a robot. The maximum range of laser and ultrasonic is 12.0 m and 1.0 m, respectively.

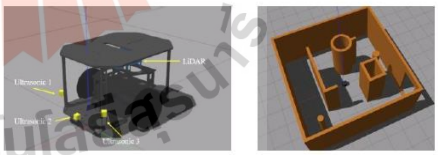


Figure 4. (left) The robot model and (right) virtual environment in Gazebo

The neural network programming framework is PyTorch. The simulator was run on Ubuntu 20.04 with Nvidia GeForce MX250 graphics card and Intel Core i5 CPU. The training finished at 3500 steps and the maximum step per episode was 60000 steps. The agent learned to navigate itself to reach a random goal from the starting point which was set at the center of the map (0.0, 0.0). The accumulated reward obtained during the training process is shown in Fig. 5. And the average Q value for every training iteration steps is shown in Fig. 6.

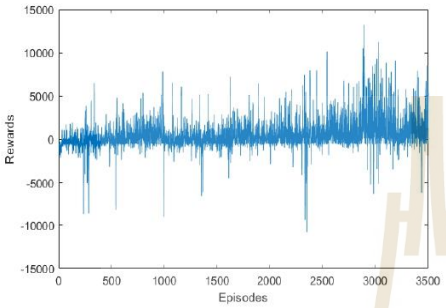


Figure 5. Accumulated reward of training process. The total episode is 3500 episode which maximum step is 60000 steps per episode.

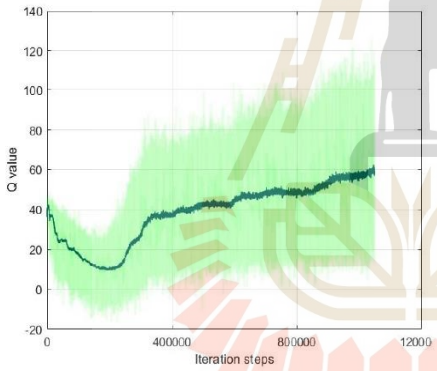


Figure 6. Average Q-value of traning process in every learning iteration steps.

B. Evaluation

After the training, the robot was tested for navigation ability in unknown environments. There were four testing environments with different format of obstacle as shown in Fig. 7. The task of robot was navigation following all desired waypoints to complete the mission safely with only learned policy from the training process.

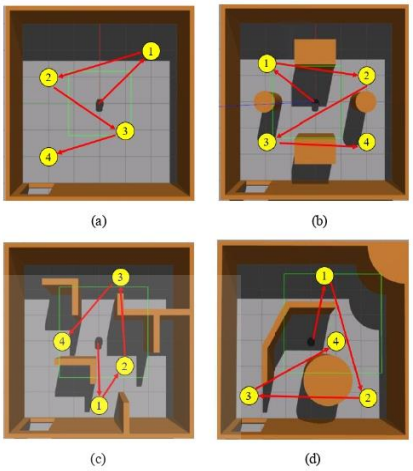


Figure 7. Four different unknown stages for testing consists of (a) test stage1. (b) test stage2. (c) test stage3 and (d) test stage4.

The robot was tested 200 times per stage and the mission success if the robot can reach all waypoints. If the robot was unable to reach all waypoints or missing only one point, the mission failed. The success rates of autonomous navigation in each stage are shown in Table I.

TABLE I. THE SUCCESS RATE OF NAVIGATION IN UNKNOWN STAGE

Stage	Success Rate [%]
1	100.0
2	70.5
3	82.5
4	69.7

In open space as in the test stage1, the robot the success rate of navigation is 100 percent. When the obstacle was added to the field, the robot collided with the wall in some rounds and cannot reach all target points. The robot moved on with uncertain path and the decision was not the same. The paths of a robot in each stage which obtained maximum rewards are shown in Fig. 8.

The hard case of robot to make the decision was the case that the destination point was in front of a robot, but it was blocked by the wall. In this case, the robot may take more time than in other situations for finding the way to detour the wall. The sharp edge points in the path occur when the robot stopped and turned back with the acute angle to change its heading.

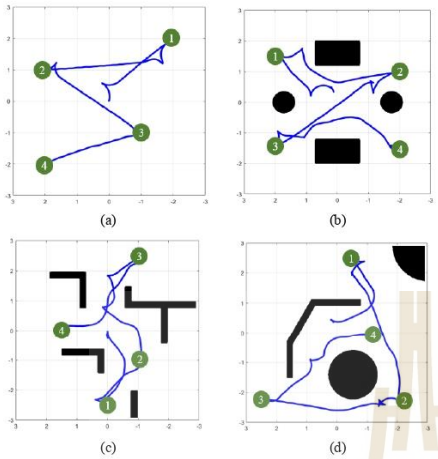


Figure 8. Path of navigation of robot in all test stages. (a) stage1 (b) stage2 (c) stage3 and (d) stage4.

V. Conclusion and Future Work

In this research, reinforcement learning is utilized for autonomous mobile robot navigation in model-free environment with continuous speed command control. The robot model comes with a laser scanner, ultrasonic distance sensors and an odometry sensor that provide state information for the robot's decision-making. A modern deep reinforcement learning approach called Deep Deterministic Policy Gradient (DDPG) is used to instruct the robot in the virtual world how to interact with its surroundings. After training, a navigation task in uncharted areas was evaluated on the robot, and it attempted to navigate in accordance with the acquired policy using continuous values of linear and angular velocity. The findings of the training and evaluation in virtual environments demonstrated that even when the autonomous agent was in an unfamiliar environment, it could learn to go to specified waypoints without colliding with anything else and find its way to the desired location. The minimal navigation success percentage is 69.7%. Implementation and assessment of the reference model, a realistic mobile robot, will be the focus of future study.

References

[1] Y.C. Kim, S.B. Cho, S.R. Oh, Map-building of a real mobile robot with GA-fuzzy controller, International Journal of Fuzzy Systems 4 (2) (2002) 696–703.

[2] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Trans. on Robotics, 32(6):1309–1332, Dec 2016.

[3] K. Zhu and T. Zhang, ''Deep reinforcement learning based mobile robot navigation: A review,'' Tsinghua Sci. Technol., vol. 26, no. 5, pp. 674–691, Oct. 2021.

[4] H. Kanayama, T. Ueda, H. Ito, and K. Yamamoto, ''Two-mode mapless visual navigation of indoor autonomous mobile robot using deep convolutional neural network,'' in Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII), Honolulu, HI, USA, Jan. 2020, pp. 536–541, doi: 10.1109/SII46433.2020.9025851.

[5] T. Xue and H. Yu, ''Model-agnostic metalearning-based text-driven visual navigation model for unfamiliar tasks,'' IEEE Access, vol. 8, pp. 166742–166752, Sep. 2020, doi: 10.1109/ACCESS.2020.3023014.

[6] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, ''Target-driven visual navigation in indoor scenes using deep reinforcement learning,'' Sep. 2016, arXiv:1609.05143. [Online]. Available: http://arxiv.org/abs/1609.05143

[7] L. Tai and M. Liu, "A robot exploration strategy based on q-learning network," in Proceedings of the 2016 IEEE international conference on real-time computing and robotics (RCAR), pp. 57–62, IEEE, Angkor Wat, Cambodia, June 2016.

[8] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in Proceedings of the 2017 IEEE international conference on robotics and automation (ICRA), pp. 3389–3396, IEEE, Marina sands bay, Singapore, May 2017.

[9] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine, "Collective robot reinforcement learning with distributed asynchronous guided policy search," in Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 79–86, IEEE, Vancouver, BC, Canada, September 2017.

[10] T. Tai, S. Li, and M. Liu, "A deep-network solution towards model less obstacle avoidance," in Proceedings of the International Conference on Intelligent Robots and Systems(IROS), pp. 2759–2764, Daejeon, Korea (South), October 2016.

[11] L. Timothy, H. Jonathan, P. Alexander, H. Nicolas, E. Tom, T. Yuvai, S. David, W. Daan, "Continuous control with deep reinforcement learning," Sep. 2015, arXiv:1509.02971. [Online]. Available: https://arxiv.org/abs/1509.02971

[12] S. David, L. Guy, H. Nicolas, D. Thomas, and R. Martin. "Deterministic policy gradient algorithms," in Proceedings of the 31st International Conference on International Conference on Machine Learning, 2014, Beijing, China, pp. 387-395.

[13] Z. Ma, Q. Huo, T. Zhang, J. Hao and W. Wang, "Deep Deterministic Policy Gradient Based Energy Management Strategy for Hybrid Electric Tracked Vehicle With Online Updating Mechanism," in IEEE Access, vol. 9, pp. 7280-7292, 2021, doi: 10.1109/ACCESS.2020.3048966.

[14] H. Sasaki, T. Horiuchi and S. Kato, "A study on vision-based mobile robot learning by deep Q-network," 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), 2017, pp. 799-804, doi: 10.23919/SICE.2017.8105597.

[15] H. Hado, G. Arthur and S. David, ''Deep Reinforcement Learning with Double Q-learning,'' Sep. 2015, arXiv:1509.06461v3. [Online]. Available: https://arxiv.org/abs/1509.06461.

[16] J. Nauta, Y. Khaluf, and P. Simoens, "Using the Ornstein-Uhlenbeck process for random exploration," in Proceedings of the 4th International Conference on Complexity, Future Information Systems and Risk (COMPLEXIS 2019), Heraklion, Greece, 2019, pp. 59–66.

ACEAIT-20234

Mobile Robot Indoor Localization Based on the Fusion of UWB with Laser Odometry and IMU Sensor

Phuwanat Phueakthong[a], Jittima Varagul[b], Nattawat Pinrath[c]

[a] Department of Mechatronics Engineering , Suranaree University of Technology, Thailand

[b] Department of Manufacturing Automation and Robotics Engineering, Suranaree University of Technology, Thailand

[c] Department of Industrial Engineering, Suranaree University of Technology, Thailand

E-mail address: phuwanat.aerod@gmail.com [a] , jittima@sut.ac.th [b] , nattawat.p@sut.ac.th[c]

Abstract

In this work, we present the fusion of sensor data from Ultra-Wideband positioning sensor, Inertial measurement unit and odometry from a laser scanner to achieve accurate and low drift 2D localization data of the mobile robot for resource constraint hardware. The Extended Kalman Filter (EKF) is utilized for fusing the data from measurement. The laser odometry generated from the RF2O algorithm and the UWB raw data is processed to the position coordinate by trilateration approach and is filtered by Kalman Filter before fusing in EKF. The system is performed in two different environments, showing that the fusion approach can give reliable and low drift pose data of the target mobile robot.

Keywords: robot localization, sensor fusion, UWB, mobile robot

1. Background/ Objectives and Goals

The localization problem is one of the challenges for autonomous mobile robot researchers and developers. The positioning system is the primary key for successful navigation. For outdoor navigation, the GNSS (Global Navigation Satellite System) technology is the best positioning system and it can reach centimeter-level accuracy by using RTK-GPS (Real Time Kinetic Global Positioning System). Unfortunately, this technology cannot be used for indoor navigation due to satellite signal limitations. For indoor navigation, it is necessary to find other solutions. The popular solution is Simultaneous Localization and Mapping (SLAM) but this method requires the odometry data of robot to get high performance. Generally, odometry can be calculated from the encoder data with the dead reckoning approach, but it can contain high error by wheel slipping and accumulation of error from integration in dead reckoning. In order to reduce the problem, multi sensor fusion is proposed to combine the data from different sources of sensors to estimate accurate odometry. The odometry data from IMU (Inertial Measurement Unit) [1], VIO (visual inertia odometry) [2,3] and Laser odometry [4] can be used to fuse by state estimation algorithm like EKF (Extended Kalman

Filter). In addition, to reduce the accumulation of errors in odometry, the data from an absolute positioning system such as UWB (Ultra-Wideband) technology that operates similarly to GPS is used for fusion. The UWB positioning system uses radio signals over a wide range of frequencies, each with signal having a very short wavelength, allowing for precise positioning. [5] proposed to fuse UWB positioning data with IMU and got reliable path tracking but using wheel odometry still have hazardous for slipping in some surface. [6,7] proposed to use UWB positioning with visual inertial odometry and can obtain smooth localization system without cumulative error in the global frame, but the visual inertial odometry requires high computational resources and high-quality camera.

In this paper, we focus on the positioning system for indoor mobile robot by using sensor fusion algorithm with UWB positioning sensor, IMU and 2D laser odometry which consumes low computational resources to get accurate pose data with low drift and cumulative error.

2. Methods

In this research, three main sensors are UWB sensor, 2D LiDAR laser scanner and IMU. All these sensors were integrated into the mobile robot which is shown in Fig. 1 and Fig. 2. The selected LiDAR is YDLIDAR TG30 which has a maximum range of 30 meters around itself. It was installed inside the mobile robot. The selected UWB module is the low-cost DW1000 module which is mounted on the ESP32 microcontroller. The UWB modules are divided into two types, the anchor module and tag module. The tag module was installed on top of the mobile and the anchor modules were mounted on the base station. The selected IMU is Adafruit 9DOF IMU which has the FXOS8700 3Axis accelerometer with magnetometer and the FXAS21002 3 axis gyroscope.

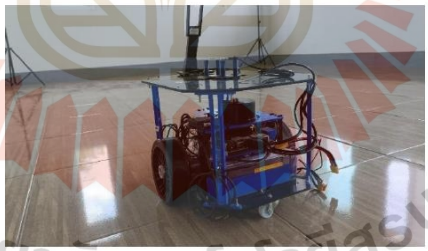


Fig. 1: The mobile robot for research

In addition, the robot has wheel encoder sensor on two sides of motor but did not fuse in the localization algorithm. The Robot Operating System2 (ROS2) is utilized for system development to operate the robot and integrate all sensor data with localization algorithm. The high-level computer is Nvidia Jetson Nano Ram 4GB and the low-level microcontroller

is Teensy4.0 600 MHz ARM Cortex M7 microcontroller.

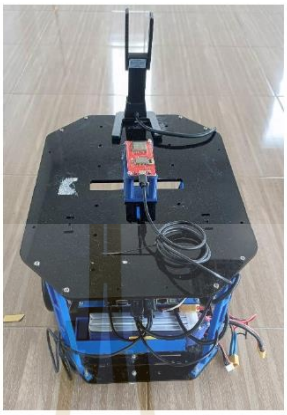


Fig. 2: The UWB module on top of mobile robot

2.1. UWB Positioning System

In order to obtain position of the tag module on the robot, the principle of the measurement is similar to the satellite positioning system. The multiple anchors were arranged in the area with known location. The ESP32 microcontroller which is in the tag module can obtain the range information from tag to all four of base station by time difference of arrival (TDOA) approach. When range measurement between tag and anchor is available at least three anchors, the tag's location can be estimated by trilateration algorithm [8]. Suppose (x_i, y_i) denotes the x and y coordinates of the n anchor $(n = 1,2,\dots,n)$ and the coordinate of the tag is (x_t, y_t). The 2D distance between each anchor and unknown tag can be calculated using the following expression.

$$d_i = \sqrt{(x_t - x_i)^2 + (y_t - y_i)^2} \tag{1}$$

The equation of trilateration-based least square for finding tag coordinate is as follows.

$$\begin{pmatrix} x_t \\ y_t \end{pmatrix} = (A^T A)^{-1} A^T b \quad , \tag{2}$$

Where,

$$A = \begin{pmatrix} 2(x_1 - x_2) & 2(y_1 - x_2) \\ \vdots & \vdots \\ 2(x_1 - x_n) & 2(y_1 - x_n) \end{pmatrix} \tag{3}$$

And

$$b = \begin{pmatrix} d_2^2 - d_1^1 + x_1^2 - x_2^2 + y_1^2 - y_2^2 \\ \vdots \\ d_n^2 - d_1^1 + x_1^2 - x_n^2 + y_1^2 - y_n^2 \end{pmatrix} \quad (4)$$

After calculation, the result of coordinate point may have high noise and deviation. In this paper, the Kalman filter is used to track the position of tag module and reduce the noise position. Kalman filter is used to estimate to the state of a system at time k ($\hat{x}_k$) by using the knowledge of prior state at time $k-1$ ($\hat{x}_{k-1}$) as in the following formula.

$$\hat{x}_k = F\hat{x}_{k-1} + Bu_{k-1} + w_{k-1} \quad (5)$$

Where F is the state transition matrix, B is the control input matrix and w_k is the dynamic system noise. The state transition matrix refers to 2D dynamics model and kinematic equation of mobile robot. The relation matrix of mobile robot's position (x, y) and velocity $(\dot{x}, \dot{y})$ can be written as following.

$$x_k = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{1}{2}(\Delta t)^2 & 0 \\ 0 & \frac{1}{2}(\Delta t)^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} \begin{bmatrix} \ddot{x}_{k-1} \\ \ddot{y}_{k-1} \end{bmatrix} + w_{k-1} \quad (6)$$

The error covariance matrix in prediction state(P_k^-) is predicted by the equation as follows.

$$P_k^- = FP_{k-1}F^T + Q \quad (7)$$

Where Q is the process noise covariance in prediction step of Kalman filter.

After the prediction step, the measurement data from the UWB sensor is added to the process to correct and update the estimation. In the update step, the Kalman gain (K_k) is calculated from transformation matrix (H) , previous estimated error covariance matrix and the measurement noise covariance (R) as in the following equation.

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \quad (8)$$

In order to use measurement data to update the estimation, the Kalman gain and measurement model are added to the update equation as follows.

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \quad (9)$$

Where,

$$z_k = H\hat{x}_k + v_k \quad (10)$$

In this process, the updated error covariance matrix(P_k) for next time step is calculated as

follows.

$$P_k = (I - K_k H) P_{\bar{K}} \tag{11}$$

With the Kalman filter, the noise of position estimation was reduced. The filtered position data is sent to the Extended Kalman Filter (EKF) based sensor fusion algorithm to get accurate location of mobile robot.

2.2. Laser odometry

In this paper, the odometry data from laser scanner is utilized instead of the odometry from wheel encoder which is prone to inaccurate overtime due to wheel slipping. The RF2O (Range Flow-based 2D Odometry) approach which proposed by [4] is used to calculate the pose and velocity of mobile robot. The RF2O is a lightweight and robust algorithm which is compatible with our robot that has resource constraints. The RF2O performs with high performance in general environments but in some environments with low feature of reference and high laser noise, the RF2O can give low accuracy odometry.

2.3. Sensor fusion

Sensor fusion is a method to combine data from multiple sensors to estimate the highly accurate state of system with low noise. This paper applies the EKF (Extended Kalman Filter)-based sensor fusion algorithm to fuse the position from UWB sensor, velocity from RF2O odometry and the orientation from IMU to get accurate pose data of the mobile robot. The input from UWB is absolute position data. The input from RF2O is the linear velocity and angular velocity and the input data from IMU is yaw angle around the normal axis. The state vector of robot consists of position (x, y), orientation (ϕ) and velocity in linear and angular type (v, ω). The motion model of estimation can be written as follows.

$$\hat{x}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \phi_{k-1} \\ v_{k-1} \\ \omega_{k-1} \end{bmatrix} + \begin{bmatrix} \cos(\phi)\,\Delta t & 0 \\ \sin(\phi)\,\Delta t & 0 \\ 0 & \Delta t \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_{k-1} \\ \omega_{k-1} \end{bmatrix} + w_{k-1} \tag{12}$$

In the EKF, the H matrix is replaced by the Jacobian matrix H_j to linearize non-linear function. The calculated Jacobian matrix from the motion function is shown as follows.

$$H_j = \begin{bmatrix} 1 & 0 & -v\sin(\phi)\Delta t & \cos(\phi)\,\Delta t \\ 0 & 1 & v\cos(\phi)\Delta t & \sin(\phi)\,\Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

There are three observation sources from UWB, IMU and laser odometry. The observation

model of the UWB filtered point data and measurement matrix (h) can be written as follows.

$$z_k = H\hat{x}_k + v_k \tag{14}$$

$$h_{uwb} = \begin{bmatrix} x_{uwb} \\ y_{uwb} \end{bmatrix} \tag{15}$$

$$z_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix} \hat{x}_k + v_k \tag{16}$$

The observation model of laser odometry is similar to the form of UWB but the input data is linear and angular velocity of robot.

$$h_{laser} = \begin{bmatrix} v_{laser} \\ \omega_{laser} \end{bmatrix} \tag{17}$$

$$z_k = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \hat{x}_k + v_k \tag{18}$$

The observation model from IMU is yaw orientation of IMU from Madgwick's filter [9] can be written as follows.

$$h_{laser} = [\phi_{imu}] \tag{19}$$

$$z_k = [0 \quad 0 \quad 1 \quad 0 \quad 0]\hat{x}_k + v_k \tag{20}$$

The predict state of EKF is similar to common Kalman filter but the state equation and process noise covariance update of prediction is linearized using the Jacobian matrix as follows.

$$\mathrm{P_k^-} = H_j \mathrm{P}_{k-1} H_j^T + Q \tag{21}$$

The update equation in EKF update step are as in follow.

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{22}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \tag{23}$$

$$P_k = (I - K_k H) P_K^- \tag{24}$$

3. Experiment and Results

In the experiment, the mobile robot was tested in two environments. The first environment is simple environment in square room with size $8 \times 4 \times 2.5$ m. The second environment is in a place that is more complicated than the first. Fig.3 and Fig.4 illustrate the first and second test environment, respectively.

3.1. Experimental Setup

In both test fields, there are 4 UWB anchors installed around the place in rectangle shape with known coordinate point. When the power of all UWB modules is on and the transceiver

is activated, the ESP32 microcontroller which is attached to the UWB tag can obtain the range data between all anchor and tag. These data will be calculated to the pose of robot and filtered by Kalman filter before sending to fuse with data from other sources. The software is divided into small nodes. Each node has its own responsibility, but all nodes can exchange data between the nodes using data distribution service (DDS) in ROS2. When all sensor nodes and control node calculation nodes are ready, the robot can move to the target point for testing. In the first environment, the coordinate and arrangement of UWB base is shown in Fig. 5. The starting point of mobile robot in the UWB frame is at (x = 1.2, y = 2.4). The robot was controlled by teleoperation to the desired point. After the robot stops in each point, the real pose data of robot from real measurement were recorded to compare with the measurement data from sensor after the mission of robot is finished. In the second environment, the test procedure is similar to testing in the first environment, but the second test field is in a larger area and there are objects and areas which can reduce the performance of RF2O odometry estimation. The coordinate of the UWB anchor is equal to the points in first environment. The starting point of robot in second environment is at ($x = 1.8, y = 0.9$). In order to collect data, the ROS2 bag is utilized to record all topics for post processing.

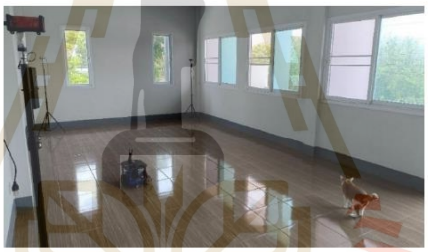


Fig. 3: Test environment1 (simple)

Fig. 4: Test environment2 (complex)

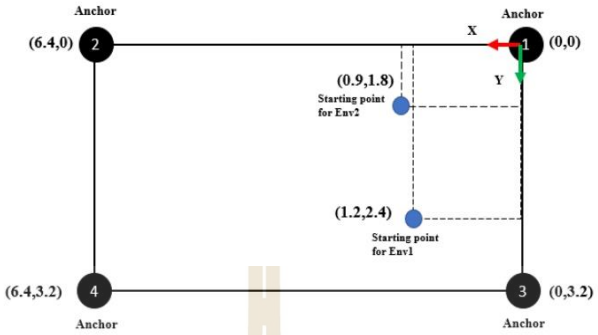


Fig. 5: Anchor and starting tag coordinate for testing.

3.2. Result

From testing, the comparison of mobile robot pose data from each source in each environment is shown in the Fig. 6 and Fig. 7.

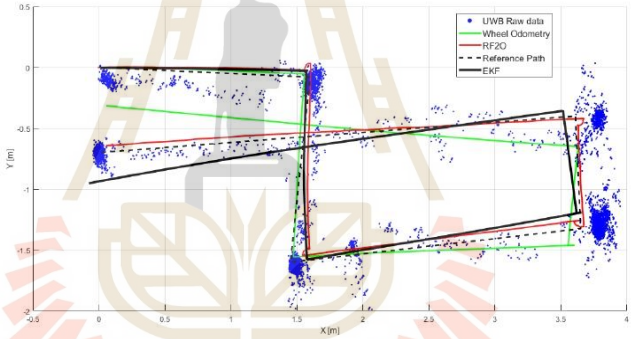


Fig. 6: Position of a robot from each source in environment1

In the first simple environment, the odometry from RF2O looks closely to the actual reference path and is more accurate than the fused odometry with EKF. The odometry from wheel encoder is highly drifting. In the plot, the area that raw data point from UWB is highly dense is the area where the mobile robot stops around 10 seconds and turns to the next waypoint.

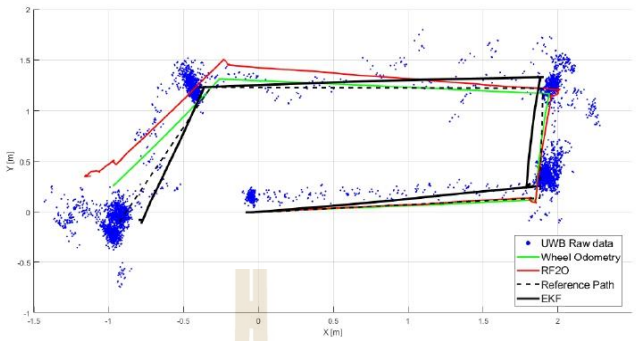


Fig. 7: Position of a robot from each source in environment 2

Fig. 7 illustrates that in the second test environment, the performance of laser odometry from RF2O is reduced by the noise of measurement and surrounding complex scene. Both laser and wheel odometry drifted over time but the pose from EKF fusion can still maintain the position of robot close to the actual reference path. All results show that the fusion of sensor data from IMU, laser odometry and UWB sensor can give reliable pose data and reduce odometry drift problem. The result of error of estimation in both environments are shown in Table 1.

Table 1: The error of robot position estimation from testing

Position Error [cm]	Environment 1			Environment 2		
Value / method	RF2O	Wheel Odometry	EKF	RF2O	Wheel odometry	EKF
Average [cm]	**4.20**	12.66	7.10	16.99	7.65	**7.22**
Max. [cm]	**16.27**	38.91	26.20	50.45	36.40	**15.84**
Min. [cm]	0.72	**0.20**	1.31	0.36	**0.22**	1.22
Std dev. [cm]	**3.39**	11.31	6.77	16.52	8.55	**3.55**

4. Conclusion and Future work

This paper proposed the EKF-based sensor fusion of position data from UWB sensor, the orientation data from IMU and the laser odometry data from the RF2O algorithm to obtain pose data of mobile robot in an indoor environment. From the experiment, we can achieve the fused data from the EKF that is close to actual reference path and has low drifting. Future work will focus on implementation of this approach with the mapping and navigation of the mobile robot in complex area such as industrial plant with the industrial grade autonomous mobile robot to prove the performance of the system in real application and improve it again.

References

M. Brossard, A. Barrau and S. Bonnabel. (2020). AI-IMU Dead-Reckoning. *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 585-595. doi: 10.1109/TIV.2020.2980758.

T. Qin, P. Li and S. Shen. (2018). VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004-1020. doi: 10.1109/TRO.2018.2853729.

C. Forster, M. Pizzoli and D. Scaramuzza. (2014). SVO: Fast semi-direct monocular visual odometry. *Proceeding of IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, pp. 15-22. doi: 10.1109/ICRA.2014.6906584.

M. Jaimez, J. G. Monroy and J. Gonzalez-Jimenez. (2016). Planar odometry from a radial laser scanner. A range flow-based approach. *Proceeding of IEEE International Conference on Robotics and Automation (ICRA)*. Stockholm, Sweden, pp. 4479-4485. doi: 10.1109/ICRA.2016.7487647.

X. Ruan, S. Liu, D. Ren and X. Zhu. (2018). Accurate 2D Localization for Mobile Robot by Multi-sensor Fusion. *Proceeding of IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC)*, Chongqing, China, pp. 839-843. doi: 10.1109/ITOEC.2018.8740490.

H. Sadruddin, A. Mahmoud and M. Atia. (2019). An Indoor Navigation System using Stereo Vision, IMU and UWB Sensor Fusion. *IEEE SENSORS*, Montreal, QC, Canada, pp. 1-4. doi: 10.1109/SENSORS43011.2019.8956942.

J. -R. Zhan and H. -Y. Lin. (2022). Improving Visual Inertial Odometry with UWB Positioning for UAV Indoor Navigation. *Proceeding of 26th International Conference on Pattern Recognition (ICPR)*, Montreal, QC, Canada, pp. 4189-4195. doi: 10.1109/ICPR56361.2022.9956500.

B. Silva, Z. Pang, J. Åkerberg, J. Neander and G. Hancke. (2014). Experimental study of UWB-based high precision localization for industrial applications. *Proceeding of IEEE International Conference on Ultra-WideBand (ICUWB)*, Paris.

Madgwick, S. (2010). An efficient orientation filter for inertial and inertial / magnetic sensor arrays.

ACEAIT-20235
AUTONOMOUS MOBILE ROBOTS FOR MATERIAL HANDING IN AN INDUSTRIAL PLANT

Saw Yi Wai Yan[a*], Kontorn Chamniprasart[a], Phuwanat Phueakthong[a], Nattawat Pinrath[b]

[a] Department of Mechatronics Engineering, Suranaree University of Technology, Thailand

[b] Department of Industrial Engineering, Suranaree University of Technology, Thailand

E-mail address: achirawat051042@gmail.com [a]*, nattawat.p@sut.ac.th [b]

Abstract

Industrial mobile robots are one of the most common handling equipment in manufacturing. It is a way of increasing production efficiency while significantly reducing operating coast and facilitating because of the inflexibility of the Automated Guided Vehicle (AGV) with regard to modifying the robot's route. Because the AGV commonly uses magnetic tape embedded under concrete to create a path for the robot. When changing the route or adding the production line, need to destroy the area to reengineer causing an increase in spending. Therefore, Autonomous mobile robots respond to the problem involved with Simultaneous Localization and Mapping (SLAM) techniques, it can be used to located and create Maps. And AMR moves according to maps and localization, it causes the main features is the flexibility to change routes. In this research will present various systems and usability of AMR for material handing in an industry.

Keywords: Industrial mobile robot, Automated guided vehicle, Autonomous mobile robot, SLAM

1. Background/ Objectives and Goals

Automated guided vehicles are becoming one of the most common smart handling tools in the manufacturing industry. Nowadays, Industrial plants prefer to use Automatic guided vehicles (AGV) to transport materials from warehouses to target points to wait for the production process, the AGV has been designed to be more efficient in the industry. [3,6,7] AGV working principle uses magnetic sensors to detect magnetic tapes under the concrete floor. Use computer principles to control the AGV to follow the route created, which can be customized according to work requirements. Some AGVs use QR codes instead of Magnetic tape [1]

It is impossible to deny that one of the main points of the AGV is routing. Sticking magnetic tape or QR codes on the ground provides a fixed route. It is difficult to change the route if you want to change routes or add routes, you must destroy the ground to dislodge the system. There will be a supplement in each case. It is hard to Computer programming and time-

consuming to edit. When the AGV is used for a long period of time, the magnetic tapes or QR codes degrade. AGV performance declines and faults may result.

The main problem with implementing AGV is its inflexibility. If an industrial plant wants to change or add a production line, it will need to repair the whole system. This increases the cost; therefore, the Autonomous Mobile Robot responds to the problem involved. The main characteristic is the flexibility to change the route. The aim of the research is to present the application of AMR in industrial plants.

2. Methods

2.1.System architecture

Fig. 1 illustrates the shape of robots that our team created together with the company. It has the following Important Components: There are Navigation Sensors installed in front and back of robots. There are safety sensors installed both in the front and back of the robots. Our model has a Monitor used for control and Monitoring. There is a charger. Installed surround Safety bumper. Fig. 1 shows the component of the robot.

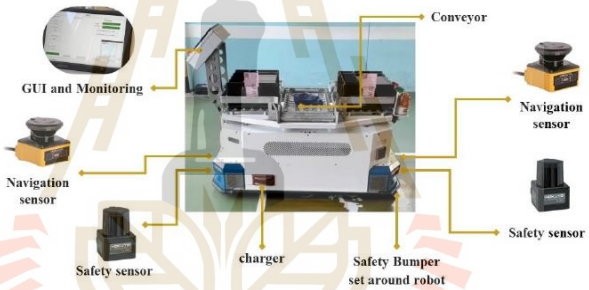


Fig. 1: Component of robot

Fig. 2 shows robot architecture consists of High level and Low levels. Low levels are used to connect the sensor and actuator to data transmission with each other such as sending orientation from the IMU command filter, feedback transmission of an encoder, and sending commands on motor control. The High-level sections contain a user interface where the algorithm resources are required to be processed. The queue for the robot is set up via GUI and WEB Master and then the queue is then handled in the route planner process. First, we check the robot state observer for other operations. The next step is the robot's Pose checker, which checks the robot for being in a position and orientation. Then the process is Dijkstra's Route Planning. Is an algorithm used to determine the path of the robot to get the short path.

Finally, Path collinearity remover steps remove points in the same line. Then select the mode of operation, AMR mode or AGV mode sends a command to navigation software.

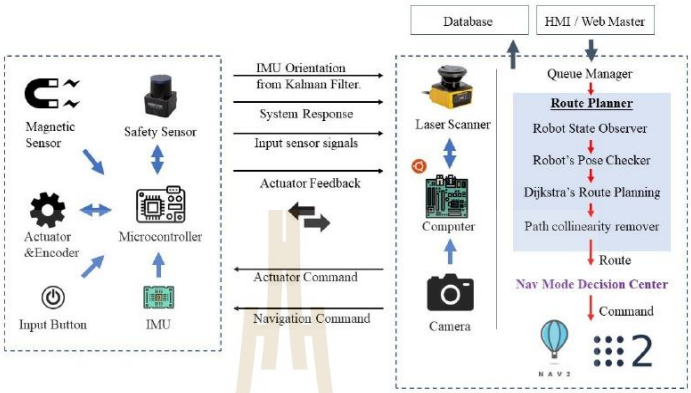


Fig. 2: Robot system architecture

2.2. Hardware

The IMU is a multi-sensor device detecting acceleration, angular velocity, angle as well as magnetic field. The small outline makes it perfectly suitable for industrial retrofit applications such as condition monitoring and predictive maintenance. A sensor measures 3-axis angle, angular velocity, acceleration, and magnetic field. Its strength lies in the algorithm which can calculate three-axis angles accurately.

LIDAR sensor is a laser-powered distance sensing device with high accuracy in measuring distance and can operate in low light or reflected light. A LIDAR sensor emits a laser beam at an object and measures its distance by measuring the time it takes for the light to return after reflecting on the sensor. LIDAR sensors are often used in research, robotics, and autonomous vehicles. High-precision and highly efficient instruments are used in various science and technology applications. LIDAR utilizes a range that rotates clockwise, enabling 360° full-scan detection of the surrounding environment and producing a map of the area. The sample rate of LiDAR directly decides whether the robot can map quickly and accurately. LIDAR improves the internal optical design and algorithm system to make the sample rate up to 8000 times/second. Computers are important devices used for calculating large amounts of complex mathematical data. The researcher used a computer with a CORE I5 CPU.

A microcontroller is an electronic device that consists of a processor with memory and signal amplifiers used to control and process data in various electronic systems. Microcontrollers are used to control motion. Sensor Data Collection and data transmission. The researcher uses

an ARM Cortex M7 32-bit 600 MHz. It is appropriate in terms of resources and processing speed.

The servo motor is a device that converts electrical energy into mechanical energy for use in the propulsion of the robot. It is precise and can control the rotation with high precision. There is an electronic control system inside the motor to control the position and rotation speed of the motor to be precise. The researcher used a 400-watt servo motor with a rated speed of 3000 rpm.

A safety sensor is a device used to detect the status or movement of objects or people to prevent danger or accidents. For the robot, it uses a 2D scanner for measuring the distance between the sensor and its surroundings. Faster response, 66 mms, and can be connected to a control system to trigger an alarm or stop it automatically. to prevent the danger that may occur in various situations.

A bumper is a device that helps in matters of safety. The behavior of the bumper is like a switch. When an object collides with the bumper, it sends an electrical signal to the robot to automatically alert or stop working. The bumper is the last safety device that will work. because it prevents the robot from colliding with the object. HMI and Web servers are part of receiving commands for the operation of the robot and monitoring its operation. Including information about the robot for users to know.

2.3. Odometry

Is the process of calculating the position or direction of a mobile robot. Using sensors connected on the Robot body to measure rotational motion. Use the data obtained to calculate the position of the Robot relative to the initial position. Odometry works well in highly dynamic environments but in an environment of constant change, the data is not as accurate as it should be. Therefore, SLAM (Simultaneous Localization and Mapping) is required to increase the accuracy of the position of the robot or vehicle.

2.4. Mapping

Simultaneous Localization and Mapping (SLAM) is the process of locating the robot's position and creating a map. It uses measurement data from the Lidar sensor to use its surroundings to estimate a map of the robot's environment and movement. Collect data own movement from Odometry and collects environment feature such as angles or block. Based on detection by the Lidar sensor. The SLAM process requires mechanisms such as Extend Kalman Filter (EKF) to update the data and estimate the degree of uncertainty of the robot's position and data in the environment. This process continues while the robot moves to obtain

information about the environment and can be used to create a map in the end. It works in conjunction with the cartographer algorithm, which is used to create 2D maps by extracting and removing important features. The Cartographer algorithm uses feature points detection and tracking to extract functionality.[4]

2.5. Localization

Robot localization is the process of determining where a mobile robot is located with respect to its environment. Localization is one of the most fundamental competencies required by an autonomous robot as the knowledge of the robot's own location is an essential precursor to making decisions about future actions. In Robotics research, Adaptive Monte Carlo Algorithm is commonly used to create a map and direct the robot to the desired location based on information from the LIDAR Sensor to calculate the probability that the robot will be in that position. The algorithm employs random sampling to estimate the potential position of the robot and uses this value to determine the direction of the mobile robot at each period. In order to implement AMCL on mobile robots, to provide and comprehensive sensor data must be prepared as required. and continuously update the robot model to improve the AMCL statistical model over time. The statistical model uses data from sensors to estimate the probabilities of the robot's current position. This model will be continuously updated as new information is received.[2]

2.6. Navigation

Navigation in mobile robot is a process that enables the robot to move efficiently to the desired destination. The ROS2 Navigation Stack is an important module for mobile robot management. It consists of several modules which work together to enable the robot to move efficiently and safely in its environment. In addition, ROS2 Navigation Stack is a versatile tool that can be customized by adding or removing modules used in the Navigation Stack as the case may be.[5]

2.7. Route Planning

Route Planning is the process of planning a route suitable for moving from the starting position to the destination position. It uses information such as a Regions Maps, Destination, Environmental information and other variables to plan a route that can move safely and efficiently within a given environment. This Route planner uses the A* algorithm to help with route planning in order to find the shortest way from the starting point to the destination.

2.8. Command System

The part of command system. At first, the user selects a task for the robot and adds that task to the queue. Confirm the queue and enter the correct password. The task is sent to the robot

then the robot functions depending on the order it receives. While running, the status and task are shown on the monitoring menu when the robot has finished the task, it deleted completed tasks from the queue. Next step is to store task information to the database and completed tasks are shown in the history menu. (Fig. 3)

2.9. Safety System

The safety system is robotic collision sensors. There are two safety features, a safety sensor and bumper switch. Safety sensors are connected to the front and back of the robot. The first case, the robot moves forward with the front safety sensors if it activated the robot stopped moving. The second case, the robot turns when one of the security sensors is activated and the robots stops moving. The third case, the robot turns back if the back safety sensor is activated the robot stopped moving. Another important case is the safety sensor is not working. The robot collides with an object and the bumper switch activated then the robot will stop working as well. (Fig. 4)

2.10. Data storage system

Is a storage device used for storage of various data. The objective is to store the information in a secure form and can access information quickly and efficiently. Use Cloud Storage to store data online. The collected data for example, all robot tasks, recording the number, date, time and information on the task.

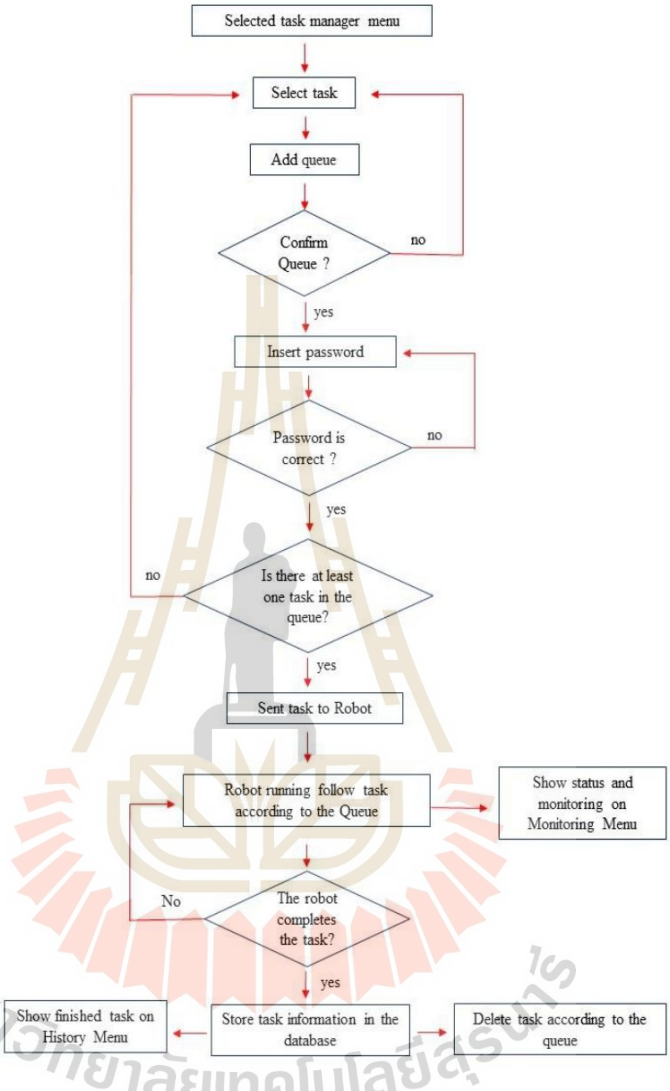


Fig. 3: flowchart of command system

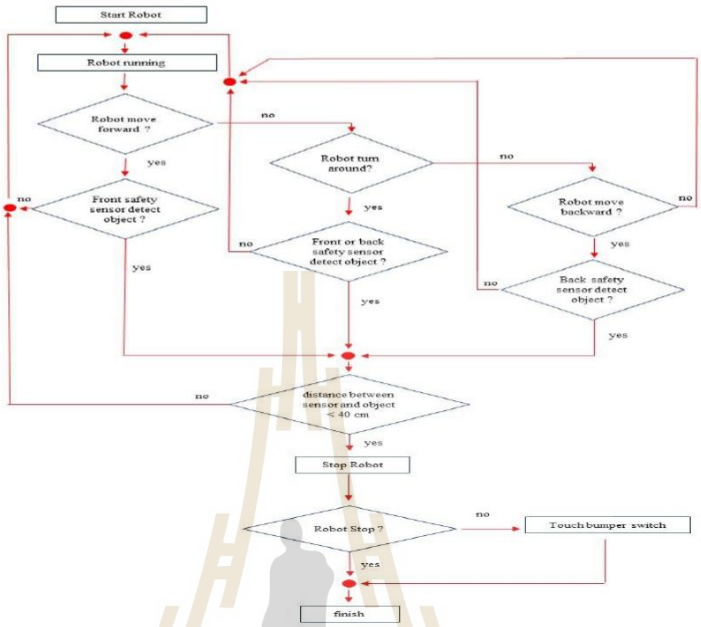


Fig. 4: flowchart of safety system

3. Results

3.1 Mapping

Data collection for creating maps uses a robot remote control method. By entering teleportation mode, then activate the cartographer to enter map creation mode. And then control the robot to move to every area we are interested in completely. In this research, the area we have mapped is an area in an industrial plant. The scanned area was 70**x**25 m. (Fig. 5)

Fig. 5: The map for the testing

a. Navigation

For navigation testing, the robot was tested in a real factory environment. The temperature is 32 degrees Celsius. Almost area is indoor condition but there are some factory gates between the route and the robot's LiDAR must face the sunlight from outside. The sunlight did not affect the navigation of the robot due to the sunlight resistance of LiDAR. When the orders are submitted, the route management or route planning system will send the overall route in the robot's lane to the navigation stack of robot. From testing, the robot can track the path of navigation and can keep itself within predetermined lane. Fig. 6 illustrates the predetermined route of navigation system and Fig 7 illustrates the example of robot during operation. During operation, the green siren was power on the buzzer will have been activated.

Fig .6: Route of navigation system

Fig .7: Robot during operation

In order to dock to the destination, the robot is necessary to stop at the entry point and rearrange the orientation before docking and change the operation mode to docking mode. Fig 8(a) illustrates the example of robot at the entry point and Fig 8(b) illustrates that how the robot docked at the station. From utilizing the automated guided vehicle style in docking, the gap between robot chassis and the station roller can be less than 3 centimeters and the tolerance in longitudinal direction is less than 2 centimeters.

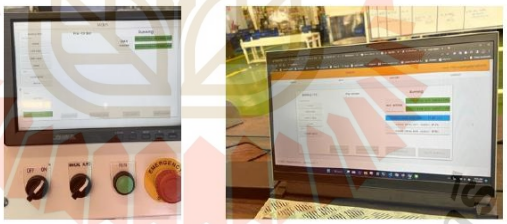
(a) (b)

Fig. 8: (a) The robot at the entry point of station (b) The robot after docking.

b. GUI and Web master

Fig. 9(a) shows the GUI screen on the robot. It consists of two parts. The first part, the touch screen is used to add tasks in the robot queue and can be added the multiple tasks. After a task is selected, press confirm queue and switch to auto. Then press run button, the robot works as a function of the task in the queue. The second part, the switch part has a switch to turn on and off the robot. Robotic operation mode selection button has two modes: manual and auto. There is a run button to start working.

From Fig. 9(b) shows the web master as a website that is able to access the robot from anywhere with a specific URL. When we login, we can order the robot's queue via this web master. Then the queue will to the robot's screen. And then wait for confirmation that the robot will keep moving.



(a) (b)

Fig. 9: (a) the GUI on the HMI of robot (b) the GUI of web master on web browser

3.4 Safety

Testing the safety of the robot we tested it in a real environment. A total of 15 tests were conducted, divided into tests according to the characteristics of the robot's motion, that is the

robot moves, turns, and backs up. The results are 100 safety percentages. Table1 shows the recording of the experiment.

Table1. Result of safety sensor testing

Motion	Test	Front safety sensor	Back safety sensor	Result
Forward	1	✓	-	Robot stops
	2	✓	-	Robot stops
	3	✓	-	Robot stops
	4	✓	-	Robot stops
	5	✓	-	Robot stops
Turn around	1	✓	✓	Robot stops
	2	✓	✓	Robot stops
	3	✓	✓	Robot stops
	4	✓	✓	Robot stops
	5	✓	✓	Robot stops
Backward	1	-	✓	Robot stops
	2	-	✓	Robot stops
	3	-	✓	Robot stops
	4	-	✓	Robot stops
	5	-	✓	Robot stops
safety percentage				100%

4. Conclusion

In this research, the main components of AMR are used for transporting materials into Industrial plants. The main purpose of AMR is to transport materials from warehouses to the various production lines of industrial plants. The results can be divided into several sections. Regarding the collection of maps using the teleoperation mode by keep the size of the map 70x25m. The AMR can be located and navigate to different points on the map. Moreover, it can receive commands from the user, both the GUI on the robot and web master, which can be run from anywhere via a URL. The AMR motion safety system has been tested in several cases. And the ability to prevent accidents by collisions around 100 percent. At the end of each task, the data can be stored in database to analyze the efficiency or number or tasks performed each day. The AMR developed by the researcher is actually used in industrial plants. In the future, researchers will develop a more effective AMR that will work in multiple environments.

5. Acknowledgement

The research of Autonomous Mobile Robots for Material handing in an Industrial plant has been successful. Due to the excellent courtesy and support from TRIPLE A ENGINEERING & SUPPLY COMPANY LIMITED.

References

Ang, J. L. F., Lee, W. K., Ooi, B. Y., & Ooi, T. W. M. (2020). Location Sensing using QR codes via 2D camera for Automated Guided Vehicles. In *2020 IEEE Sensors Applications Symposium (SAS)*. Doi: 10.1109/SAS48726.2020.9220022

Fox, D. (2001). KLD-Sampling: Adaptive Particle Filters. In *NIPS'01: Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*. pp. 713-720.

Georlette, V., Melgarejo, J. S., Bette, S., Point, N., & Moeyaert, V. (2022). Work-in-Progress: Using Li-Fi to control Automated Guided Vehicles. Steps towards an industrial market product. In *2022 IEEE 18th International Conference on Factory Communication Systems (WFCS)*. doi: 10.1109/WFCS53837.2022.9779166

Hess, W., Kohler, D., Rapp, H., & Andor, D. (2016). Real-Time Loop Closure in 2D LIDAR SLAM. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. doi: 10.1109/ICRA.2016.7487258

Macenski, S., Martin, F., White, R., & Clavero, J. G. (2020). The Marathon 2: A Navigation System. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 2718–2725.

Sasamoto, H., Valazquez, R., Gutierrez, S., Cardona, M., Ghavifekr A. A., & Visconti, P. (2021). Modeling and Prototype Implementation of an Automated Guided Vehicle for Smart Factories. In *2021 IEEE International Conference on Machine Learning and Applied Network Technologies (ICMLANT)*. doi: 10.1109/ICMLANT53170.2021.9690543

Zaman, U. K., Aqeel, A. B., Naveed, K., Asad, U., Nawaz, H., & Gufran, M. (2021). Development of Automated Guided Vehicle for Warehouse Automation of a Textile Factory. In *2021 international Conference on Robotics and Automation in Industry (ICRAI)*. doi: 10.1109/ICRAI54018.2021.9651360

AN IMPLEMENTATION OF OBJECT TRACKING METHODS ON PANAND TILT MANIPULATORFOT TEACHER TRACKING IN HYBRID CLASSROOM

[1]PHAKASINEE SINGCHAROENKIT, [2]PHUWANAT PHUEAKTHONG, [3]APHILAK LONKLANG, [4]JITTIMA VARAGUL, [5]KONTORN CHAMNIPRASART

[1,2,3,4,5]School of Mechanical Engineering, Suranaree University of Technology,
111 Suranaree Mueang Nakhon Ratchasima Thailand 30000
E-mail: [1]phakasinee.s@gmail.com,[2]jittima@sut.ac.th, [3]kontorn@sut.ac.th

Abstract - Hybrid Classroom due to the pandemic of COVID-19, the key to success for a digital classroom of the Suranaree University of Technology is broadcasting the teaching and learning activities onsite classroom via zoom application. One problem about the students who take this classroom online can not meet their teacher because the integrated camera on the classroom computer is stacked on the table in front of the class. The reason why there can not sense to teachers acts in the classroom. This paper aims to implement the object tracking method to the pan and tilt manipulator, a camera integrated. The results show that the three selected methods can achieve this task. The best accuracy for teacher tracking is the CSRT method with an IoU of 0.77 at 410 x 308 pixels.

Keywords - Teacher Tracking, Object Tracking, Camera Tracking, Pan and Tilt

I. INTRODUCTION

Due to the pandemic of COVID-19, physical distancing is the critical rule for university teaching and learning activities. Laboratories are essentials classes for the engineering education field. Online classrooms are not the key to success for these cases. A hybrid classroom was selected. They are focusing on large size industrial robot laboratories. These are the essential laboratory for undergraduate students who are majoring in a mechatronics engineering curriculum. The teacher assistants need to use the monitoring camera to present the movement of robots and robot teaching situations. Sometimes the size of the robot is a large size, consequenctly the detail of robot cound not be collected.

Due to the above problem, the aim of this research is to develop the tracking base for teacher monitoring camera. Real-time video will be used for image processing to achive the tracking task and will be recorded for e-courseware stuff. By implementing the three onshelf tracking algoritm to the pan and tilt angle base which paning by DC stepper and tilting by servo motor.

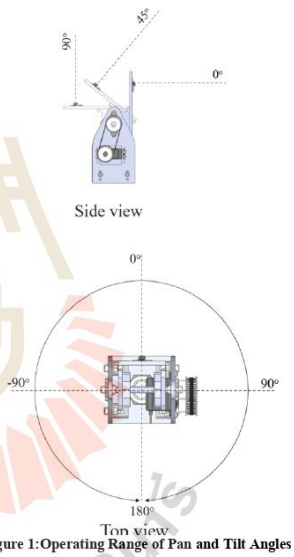


Figure 1:Operating Range of Pan and Tilt Angles

II. SYSTEM DESCRIPTION

A. Hardware configuration

The range of the pan angle is from 0 – 360 degrees left to right and the tilt angle range is from 0 – 90 degrees from ground to air as in Fig.1.

There are three main components as in Fig.2. Using the robot operating system (ROS) network to be a server for data communication. Firstly, the image processing module are included with raspberry pi and pi-camera. The second is the input command, this module will help the user set the parameter for tracking algorithm and starting the system.

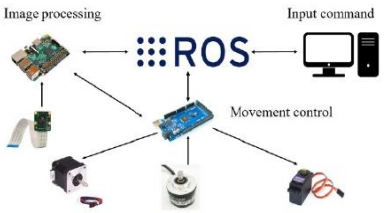


Figure 2: Hardware Overview

Input Command will be sent to the image processing module via ROS network. The tracking data is a region of interest (ROI) from the user. After a user created the ROI from the raw picture, the ROI data will be sent to the image processing module for making the decision. The making decision algorithm was integrated into the Raspberry Pi 4 controllers. Python programming was developed be a computational module. The last section, the movement module, will be received the position command from raspberry pi 4 to activate the pan and tilt base. An Arduino mega controller was selected to control these two motors at the same time. P-controller is integrated into the stepper control algorithm for stability movement.

B. Software descriptions

Robot Operating System(ROS) is commonly used for the robot task framework to develop the software and hardware related to the robots. They were using of ROS Noetic version and Ubuntu 20.04 focal fossa as the system environment. Python language is used for algorithm development. The imageprocessing software is developed based on the OpenCV version 4.3. Moreover, the last section is motors controllers, Arduino IDE, to create the P-controller programming for stepper motor and servo motor controller.The Arduino controller received the orientation from previous software via ROS node and controlled the motor simultaneously. The maximum range of pan limit is -180 degrees and 180 degrees, andthe overall pan angle is 360 degrees. P-controller is the optimized controller for use in this position control case because the speed of the stepper motor to achieve the task is still slow.

The system workflow of this system can be presented as the flowchart in Fig.3.

1) The real-time picture, captured from pi-camera, with desired quality of pixel. The more pixels of the picture, the more processing time need to use in the image processing period.A size of 410x308 pixels or lower were selected to deal with. The captured picture will be sent to the next node throughthe ROS master server.

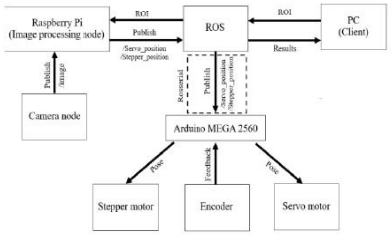


Figure3:System WorkFlow

2) Processing node received the captured picture and compared it with ROI from a user. In this case, we selected the teacher as ROI for the tracking system during the class periodby marking a blue square bounding box. Three methods of tracking system were implemented into this system, KCF, MOSSE, and CSRT. For easy monitoring of the operation of the image processing module, a square bounding box is marked by green bounding box as in Fig.8.

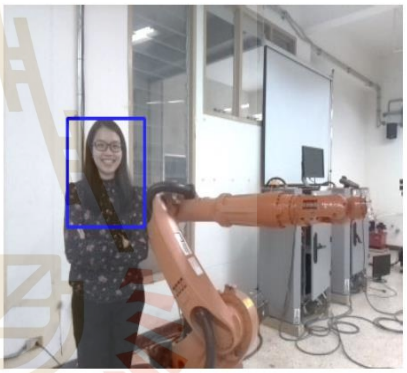


Figure 4:Example of Target Selection

3) Using the mathematical equation to compute [6] the centroid of the tracking object in vertical and horizontal directions. Coordinate from this equation will be the centroid of a tracking box.

4) If the center of the tracking box is not located in the center of the camera frame, the pan and tilt commands will send the movement angle to the movement controller to respond to this situation. The automatic control concept is to center the teacher in the middle of the camera frame.

5) Movement Controller receives the angle data from the ROS node and takes action to the pan and tilt angle with each motor controller.

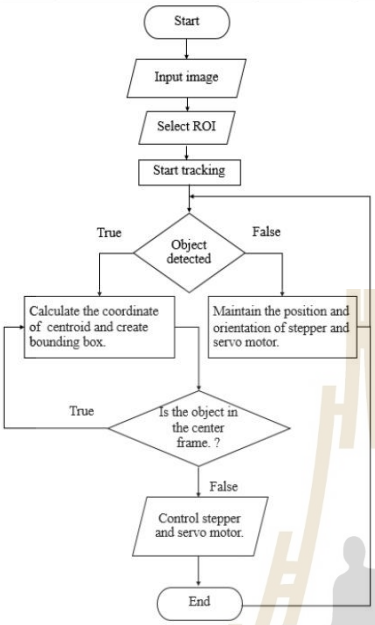


Figure 5: Algorithm Work Flow

C. Object tracking method

The solution of the teacher tracking method has to use the object tracking method. In this research, we implement on-shelf object tracking algorithms to perform this task. KCF, MOSSE, and CSRT were selected for the accuracy test. This group of algorithms usesa low level of computational cost. KCF [1],[2] (Kernelized Correlation Filters) is the object tracking algorithmthat uses the correlation value to match the sample.In object tracking, the correlation value between the ROI patch in the future frame and the original translated patch will be the highest. The KCF algorithm tends to be more accurate than the MOSSE algorithm.

MOSSE [5] (Minimum Output Sum of Squared Error) is the algorithm that used the MOSSE filter, which can discriminatebetween the ROI and the background image. This algorithm performs well in the change of rotation, light, brightness, and object scale.The MOSSE algorithm tends to much faster than KCF and CSRT,the accuracy less than KCF and CSRT algorithm.

CSRT[3],[4](Channel and Spatial Reliability Tracker)is the objecttracking model which improvedthe Discriminative Correlation Filter (DCF)algorithm with spatial and channel reliability. The spatial reliability map makes the CSRT can adjust the filter size, which makes the CSRT model better than the DCF algorithm and can handle non-rectangular shape ROI.The CSRT algorithm tends to be more accurate than the KCF algorithm but slightly slower than KCF.

III. EXPERIMENT RESULTS

A. Evaluation method

For the method of testing the tracking method, accuracy on pan and tilt angle manipulator. By using the three algorithms mentioned in the previous chapter. We divide the captured picture into three parts of the experiment, including 205x154 308x231, and 410 x308 pixels, respectively. The target moves at the same track, and time is the controlled parameter of this experiment. Then measuring the [7] Intersection over Union (IoU) of each track is used to determine the first accuracy of the tracking method. IoU is one measuring value to measure the intersection of area between the ideal frame and the actual frame. The output value of IoUfalls into the range of 0 to 1 (equation 1).

$$\text{Intersection over union (IOU)} = \frac{\text{Intersect area}}{\text{Union area}} \quad Eq.1$$

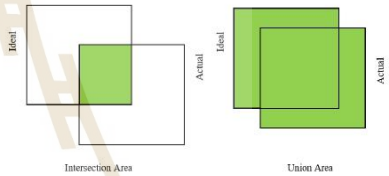


Figure 6: Intersection over Union (IoU)

If the value of IoU is higher and close to 1 is a sign that the actual and ideal area is located in the same frame. If the value of IoU is one shows that the tracking area and the ROI area are located in the same frame.

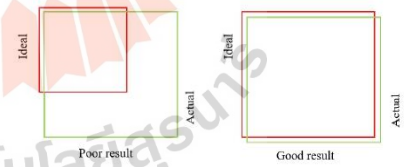


Figure 7: Result of the Value of IoUvs. Accuracy

In the second accuracy test, the difference between the actual frame and ideal is compared. In this test, the Euclidean Distance Expression in Eq.2 for measuring the distance between two centroids was used.

$$d(x,y) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad Eq.2$$

On the other hand,compared withthe IoU value, the value of the Euclidean distance must be closed to Zero, the accuracy of the tracking method will be better than others. For easy understanding of this value, the percent of centroid error (C.E.) is calculated by Eq.3.

$$C.E. = \left| \frac{d_{actual} - d_{ideal}}{d_{actual}} \right| X\ 100\% \qquad Eq.3$$

B. Results

The result of IoU and C.E. of the experiment was shown in Tables 1 and 2. The example of the tracking method is shown in Fig.8. The red bounding box represents the ideal ROI, and the green one represents the actual one.

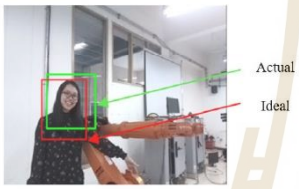


Figure 8:Intersection over Union of the Actual and Ideal Frame

From Table 1, The result shows that if the number of pixels is increasing, the more accuracy in IoU. MOSSE algorithm returns the highest value of IoU in 205x154 pixels condition. On the other hand, CSRT returns the highest value of the IoU.

Image Pixel	Model	(IOU)
205x154	CSRT	0.74
	KCF	0.71
	MOSSE	0.86
308x231	CSRT	0.79
	KCF	0.72
	MOSSE	0.70
410x308	CSRT	0.77
	KCF	0.73
	MOSSE	0.69

Table1: Intersection over Union (IoU)Results

From Table 2, the result shows that the centroid error of the tracking methods is in the same way—the overall centroid error of each tracking condition not over than 2% error. KCF returns the highest centroid error in all conditions of image pixels.

Image Pixel	Model	C.E. [%]
205x154	CSRT	1.79
	KCF	1.59
	MOSSE	0.30
308x231	CSRT	1.33
	KCF	1.72
	MOSSE	1.64
410x308	CSRT	0.88
	KCF	1.60
	MOSSE	1.16

Table2: Centroid error (C.E) Results

IV. CONCLUSION

This paper presents the accuracy testing result from the three on-shelf tracking object algorithmsto the teacher tracking task in the robot laboratory class in university. The three selected algorithms are KCF, MOSSE, and CSRT. By implementing the tracking algorithm results to the pan and tilt manipulator,the camera is integrated to track the teacher in a laboratory.Stepper motor and servo motorare integrated to control pan and tilt motionsfor centering the ROI in the middle of the frame.The result shows that the accuracy of the CSRT both in IoU and centroid errorreturns the accuracy of IoU value of 0.77 and centroid error of 0.88 in a condition of 410x380 pixels. In conclusion, the CSRT algorithm is the best choice in three selected algorithms to deal with teacher tracking in the laboratory classroom.

REFERENCES

[1] Henriques,et al., "High-Speed Tracking with Kernelized Correlation Filters", 2014.
[2] George, Jose and Mathew, "Performance Evaluation of KCF based Trackers using VOT Dataset" , 2018.
[3] Lukezic, et al.,, "Discriminative Correlation Filter Tracker with Channel and Spatial Reliability" , 2019.
[4] Wei Mi and Tsuen Yang, "Comparison of Tracking Techniques on 360-Degree Videos" , 2019.
[5] Bolme,et al., "Visual Object Tracking using Adaptive Correlation Filters", 2010.
[6] George, Jose and Mathew, "Performance Evaluation of KCF based Trackers using VOT Dataset" , 2018.
[7] Adrian Rosebrock, "Intersection over Union (IoU) for object detection", 2018, from https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

★★★

ประวัติผู้เขียน

นายภูวนาถ เผือกทอง เกิดเมื่อวันที่ 3 ตุลาคม พ.ศ. 2540 จบการศึกษาในระดับมัธยมศึกษาตอนปลายจากโรงเรียนหัวหินวิทยาลัย อ.หัวหิน จ.ประจวบคีรีขันธ์ และสำเร็จการศึกษาวิศวกรรมศาสตรบัณฑิตในหลักสูตรวิศวกรรมอากาศยาน มหาวิทยาลัยเทคโนโลยีสุรนารีในปี พ.ศ. 2562 และทำงานในตำแหน่งวิศวกรออกแบบอากาศยานไร้คนขับ ที่บริษัท อาร์วี คอนเน้กซ์ จำกัด ซึ่งในระหว่างทำงาน ในปี พ.ศ. 2563 ได้เริ่มเข้าศึกษาต่อในระดับปริญญาโท หลักสูตรวิศวกรรมเมคคาทรอนิกส์ ณ มหาวิทยาลัยเทคโนโลยีสุรนารี เพื่อศึกษาหาความรู้เพิ่มเติมในด้านที่เกี่ยวข้อง และต่อมาได้ลาออกจากบริษัทในปี พ.ศ. 2564 เพื่อทุ่มเทเวลากับงานวิจัยที่เหลือโดยระหว่างนั้นได้เป็นที่ปรึกษาให้กับบริษัท ทริปเปิล เอ เอนจิเนียร์ริง แอนด์ ซัพพลาย จำกัด ในการสร้างหน่วยวิจัยด้านหุ่นยนต์อัตโนมัติ ณ อุทยานวิทยาศาสตร์ภาคตะวันออกเฉียงเหนือตอนล่าง