ความรู้เบื้องต้นเกี่ยวกับหุ่นยนต์เคลื่อนที่อัตโนมัติ

Introduction to Autonomous Mobile Robot

พนัส นัถฤทธิ์

ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร พิษณุโลก

E-mail: panusn@nu.ac.th

***บทคัดย่อ* - บทความนี้มีจุดมุ่งหมายเพื่อนำเสนอพื้นฐานความรู้และเทคโนโลยีของหุ่นยนต์เคลื่อนที่ตั้งแต่อดีตจนถึงปัจจุบัน รวมไปถึงการศึกษาทฤษฎีและหลักการต่างๆ ที่เป็นองค์ประกอบสำคัญในการควบคุมหุ่นยนต์ให้สามารถเคลื่อนที่ได้เองโดยอัตโนมัติจากจุดเริ่มต้นไปสู่จุดหมายปลายทางโดยไม่มีการปะทะ เฉี่ยวชนหรือแม้แต่สัมผัสกับสิ่งกีดขวางใดๆ ที่เป็นอุปสรรคขัดขวางการเคลื่อนที่ของหุ่นยนต์ไม่ให้เข้าสู่จุดหมายปลายทางได้**

คำสำคัญ – หุ่นยนต์เคลื่อนที่ สถาปัตยกรรมการควบคุมหุ่นยนต์เคลื่อนที่ การนำทางหุ่นยนต์เคลื่อนที่เข้าสู่จุดหมาย ระบบเซนเซอร์

***Abstract* – This paper addresses the essential knowledge related to mobile robot navigation. In the paper, the relevant information regarding research in mobile robotics area is presented including background and their history, control architecture, navigation, sensor system, and software. The robot architecture is the foundation of control in mobile robot while the navigation system enables the robot to safely navigate to its final destination. Sensors assist the mobile robot to determine its internal status and acquire information of external surrounding. All knowledge described above are finally integrated and programmed into the software to undertake a number of specifics tasks.**

Keywords – Mobile Robot, Mobile Robot Architecture, Mobile Robot Navigation, Sensor System

1. บทนำ

หุ่นยนต์ (Robot) สามารถนิยามได้ว่าเป็นเครื่องจักรที่สามารถรับรู้ (Sense) คิดวางแผน (Plan) และลงมือทำได้ (Act) โครงสร้างของหุ่นยนต์จะมีส่วนประกอบหลายประเภท อาทิ ชิ้นส่วนทางกล อุปกรณ์ทางไฟฟ้า อิเล็คทรอนิคส์และคอมพิวเตอร์ หุ่นยนต์ถูกพัฒนาขึ้นเพื่อปฏิบัติงานแทนมนุษย์ในงานทุกประเภททั้งทางตรงและทางอ้อม ซึ่งโดยทั่วไปแล้วหุ่นยนต์จะถูกใช้ในงานที่ต้องการความถูกต้องแม่นยำ รวมทั้งงานที่เสี่ยงอันตรายเกินกว่าที่มนุษย์จะสามารถดำเนินการได้ ทั้งนี้เนื่องจากหุ่นยนต์ไม่รู้จักเหน็ดเหนื่อยและมีความทนทานต่อเงื่อนไขทางกายภาพสูงกว่ามนุษย์ หุ่นยนต์ไม่เบื่อหน่ายต่องานที่ซ้ำซากจำเจและไม่วอกแวกขณะปฏิบัติงาน

หุ่นยนต์สามารถแบ่งออกได้เป็นสองประเภทหลักๆ คือ หุ่นยนต์เคลื่อนที่ (Mobile robot) และหุ่นยนต์แขนกล (Robot manipulator) โดยที่หุ่นยนต์เคลื่อนที่นั้นเป็นหุ่นยนต์ที่สามารถเคลื่อนตัวเองไปยังที่ต่างๆ ได้ และหุ่นยนต์ประเภทนี้ยังสามารถแบ่งย่อยออกได้อีกเป็นสามประเภทตามสภาพแวดล้อมที่หุ่นยนต์ปฏิบัติงาน คือ หุ่นยนต์เคลื่อนที่บนบก (Land based mobile robot) หุ่นยนต์เคลื่อนที่ในอากาศ (Airborne robot) และหุ่นยนต์เคลื่อนที่ในน้ำ (Surface and sub-sea robot) หุ่นยนต์เคลื่อนที่บนบกเป็นหุ่นยนต์ที่นิยมใช้งานอย่างแพร่หลาย ตัวอย่างของหุ่นยนต์เคลื่อนที่ประเภทนี้ที่พบเห็นโดยทั่วไป คือ หุ่นยนต์เคลื่อนที่แบบล้อ (Wheeled mobile robot) แบบล้อ

ตีนตะขาบ (Tracked mobile robot) และแบบมีขา (Legged mobile robot) ดังแสดงในรูปที่ 1

(ก) (ข) (ค)

รูปที่ 1 หุ่นยนต์เคลื่อนที่บนบก (ก) แบบล้อ (ข) แบบล้อตีนตะขาบ และ (ค) แบบมีขา

ส่วนหุ่นยนต์แขนกลนั้นเป็นหุ่นยนต์ที่สามารถหยิบจับและเคลื่อนย้ายวัตถุ/ สิ่งของที่อยู่ภายในขอบเขตพื้นที่การทำงานของหุ่นยนต์เองเท่านั้น ทั้งนี้เพราะมีข้อจำกัดที่โครงสร้างของหุ่นยนต์ โดยฐานของหุ่นยนต์แขนกลนั้นจะถูกยึดไม่ให้เคลื่อนที่ได้ การทำงานของหุ่นยนต์ประเภทนี้จะทำงานในลักษณะคล้ายกับแขนของมนุษย์

สำหรับบทความนี้จะกล่าวถึงพื้นฐานความรู้และเทคโนโลยีของหุ่นยนต์เคลื่อนที่แบบล้อ โดยเน้นศึกษาทฤษฎี องค์ความรู้และผลงานวิจัยทางด้านหุ่นยนต์เคลื่อนที่ภายในอาคารเป็นหลัก (Indoor mobile robot) ทั้งนี้เนื่องจากในปัจจุบันมีการเติบโตอย่างกว้างขวางของอุตสาหกรรมการผลิตหุ่นยนต์เคลื่อนที่แบบล้อเพื่อใช้งานภายในอาคารสำนักงาน โรงพยาบาล พิพิธภัณฑ์ และภายในครัวเรือน อาทิเช่น หุ่นยนต์เพื่อการศึกษาวิจัยและใช้งานทั่วไป Pioneer robot ซึ่งผลิตโดยบริษัท ActiveMedia Robotics [1] หุ่นยนต์ดูดฝุ่น Roomba ซึ่งผลิตโดยบริษัท iRobot [2] และ ดินสอ[1] หุ่นยนต์บริการที่พัฒนาโดยบริษัท CTAsia Robotics [3] ซึ่งเป็นบริษัทของคนไทย เป็นต้น

2. ความเป็นมาของหุ่นยนต์เคลื่อนที่

หุ่นยนต์เคลื่อนที่มีประวัติความเป็นมาที่ยาวนาน โดยหุ่นยนต์ที่ได้รับการยอมรับว่าเป็นหุ่นยนต์เคลื่อนที่ตัวแรกของโลกนั้นมีชื่อว่า Shakey ซึ่งถูกพัฒนาขึ้นในช่วงปลายทศวรรษที่ 60 โดยสถาบันวิจัย Stanford (SRI: Stanford Research Institute) [4] หุ่นยนต์ Shakey (แสดงดังรูปที่2) สามารถรับรู้ถึงสภาพแวดล้อมบริเวณพื้นที่ปฏิบัติงาน รวมถึงรับรู้ตำแหน่งและขนาดของสิ่งกีดขวางได้ด้วยเซนเซอร์ชนิดต่างๆ ได้แก่ กล้องโทรทัศน์ (TV camera) เลเซอร์วัดระยะ (Laser range finder) และเซนเซอร์ตรวจจับการชน (Bumper) นอกจากนี้ Shakey ยังสามารถคิดและลงมือปฏิบัติงานได้ โดยอาศัยโปรแกรมควบคุมการทำงานที่มีชื่อว่า STRIPS [4]

รูปที่ 2 หุ่นยนต์ Shakey

หุ่นยนต์ Shakey แบ่งการทำงานออกเป็น 3 ส่วนคือ ส่วนควบคุมระดับล่าง (Low-level control) ส่วนควบคุมระดับกลาง (Intermediate-level control) และส่วนควบคุมระดับบน (High-level control) โดยส่วนควบคุมระดับล่างนั้นมีหน้าที่ดูแลฟังก์ชันพื้นฐานของหุ่นยนต์ นั่นคือการเดินหน้าถอยหลัง การหมุนและการเลี้ยว รวมถึงการควบคุมความเร็วและทิศทางการเคลื่อนที่ของหุ่นยนต์ ส่วนควบคุมระดับกลางทำหน้าที่ติดต่อประสานงานระหว่างส่วนควบคุมระดับล่างและส่วนควบคุมระดับบน โดยส่วนควบคุมระดับบนนั้นมีหน้าที่วางแผนและออกแบบเส้นทางการเคลื่อนที่ (Path planning) เพื่อให้หุ่นยนต์สามารถเคลื่อนที่ไปสู่เป้าหมายได้อย่างถูกต้อง สำหรับการวางแผนเส้นทางนั้น ส่วนควบคุมระดับบนจะประมวลผลข้อมูลที่หุ่นยนต์รับรู้จากเซนเซอร์แต่ละชนิด (Sense) แล้วนำมาสร้างเป็นโมเดลจำลองสภาพแวดล้อมบริเวณที่หุ่นยนต์กำลังทำงานอยู่เพื่อใช้ในการวางแผนเส้นทาง (Plan) เมื่อวางแผนเส้นทางเรียบร้อยแล้ว ส่วนควบคุมระดับบนจะส่งเส้นทางดังกล่าวให้กับส่วนควบคุมระดับกลางเพื่อใช้ออกคำสั่งให้ส่วนควบคุมระดับล่างทำการเคลื่อนหุ่นยนต์ไปตามเส้นทางที่วางแผนไว้ต่อไป (Act)

หุ่นยนต์ Shakey เป็นที่รู้จักและได้รับความนิยมอย่างแพร่หลาย อีกทั้งยังเป็นต้นแบบในการพัฒนาหุ่นยนต์เคลื่อนที่อีกหลากหลายรูปแบบตามมาเป็นลำดับ อาทิ หุ่นยนต์ HILARE

[1] ปัจจุบันมีการใช้งาน “ดินสอ” เป็นหุ่นยนต์เสิร์ฟอาหารภายในภัตตาคาร MK สุกี้ หลายสาขาในกรุงเทพมหานคร

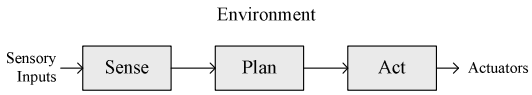
ซึ่งถูกพัฒนาขึ้นที่ห้องปฏิบัติการ LAAS ประเทศฝรั่งเศสในช่วงปลายทศวรรษที่ [5] 70 หุ่นยนต์ Stanford cart ที่ถูกออกแบบและสร้างขึ้นที่มหาวิทยาลัย Stanford ในปี 1977 โดย Hans Moravec ขณะกำลังศึกษาในระดับปริญญาเอก [6] และหุ่นยนต์ CMU Rover ซึ่งถูกพัฒนาขึ้นที่มหาวิทยาลัย Carnegie Mellon (CMU) ในช่วงต้นทศวรรษที่ 80 โดย Hans Moravec เช่นเดียวกัน [7] แต่เป็นการออกแบบและสร้างขึ้นหลังจากสำเร็จการศึกษาและเข้าทำงานที่มหาวิทยาลัย Carnegie Mellon แล้ว

อย่างไรก็ดี ในช่วงต้นของการพัฒนาหุ่นยนต์เคลื่อนที่นั้น การเคลื่อนที่ของหุ่นยนต์ส่วนใหญ่ยังเป็นไปอย่างเชื่องช้าและมีประสิทธิภาพค่อนข้างต่ำ นั่นจึงเป็นสาเหตุให้นักวิจัยจำนวนมากพยายามพัฒนาความสามารถของหุ่นยนต์เพื่อให้เคลื่อนที่ได้เร็วและมีประสิทธิภาพมากขึ้น โดยในช่วงทศวรรษที่ 80 วิชาการทางด้านหุ่นยนต์ได้ถูกพัฒนาขึ้นอย่างก้าวกระโดด หลากหลายทฤษฎีและวิธีการที่ใช้ในการควบคุมการเคลื่อนที่ รวมถึงเทคนิคการออกแบบและสร้างหุ่นยนต์ได้ถูกพัฒนาขึ้น โดยทฤษฎีและวิธีการควบคุมที่ถูกคิดค้นขึ้นนั้นบางส่วนยังคงทันสมัยและยังถูกนำมาใช้งานอยู่ในปัจจุบัน

3. สถาปัตยกรรมการควบคุมหุ่นยนต์เคลื่อนที่

สถาปัตยกรรมการควบคุมหุ่นยนต์เคลื่อนที่ (Mobile Robot Architecture) คือกระบวนการที่ประกอบด้วยการรับข้อมูลของสภาพแวดล้อมบริเวณที่หุ่นยนต์กำลังปฏิบัติงานอยู่ รวมทั้งตำแหน่งและขนาดของสิ่งกีดขวางมาจากเซนเซอร์แล้วจัดการกับข้อมูลดังกล่าวเพื่อนำไปใช้ในการตัดสินใจว่าจะให้หุ่นยนต์ทำการเคลื่อนที่ต่อไปอย่างไร เมื่อตัดสินใจได้แล้วจึงสั่งให้หุ่นยนต์ลงมือปฏิบัติงาน สถาปัตยกรรมการควบคุมหุ่นยนต์เคลื่อนที่สามารถแบ่งออกได้เป็น 3 รูปแบบ คือสถาปัตยกรรมแบบ Deliberative แบบ Reactive และแบบ Hybrid

สถาปัตยกรรมแบบ Deliberative ถูกนำมาใช้ตั้งแต่ยุคแรกของการวิจัยและพัฒนาทางด้านหุ่นยนต์เคลื่อนที่ โดยมีหุ่นยนต์ Shakey [4] และ Stanford Cart [6] เป็นตัวอย่างของหุ่นยนต์ที่ใช้งานสถาปัตยกรรมรูปแบบนี้ สถาปัตยกรรมแบบ Deliberative มุ่งเน้นที่จะสร้างโมเดลจำลองของสภาพแวดล้อมบริเวณที่หุ่นยนต์ทำงานอยู่ โดยโมเดลดังกล่าวจะถูกสร้างขึ้นจากข้อมูลภายในที่หุ่นยนต์มีอยู่ (Pre-stored data) และข้อมูลภายนอกที่หุ่นยนต์รับเข้ามาจากเซนเซอร์ (Sense) เมื่อได้โมเดลแล้วระบบจะนำไปใช้ในการวางแผนเส้นทางและตัดสินใจเลือกการกระทำ (Plan) เช่น เคลื่อนที่เข้าหาจุดหมาย หรือหลบหลีกสิ่งกีดขวาง โดยเมื่อตัดสินใจเลือกการกระทำแล้วระบบจะสั่งให้หุ่นยนต์ทำการเคลื่อนที่ต่อไป (Act) รูปที่ 3 แสดงแผนผังการทำงานของสถาปัตยกรรมแบบ Deliberative



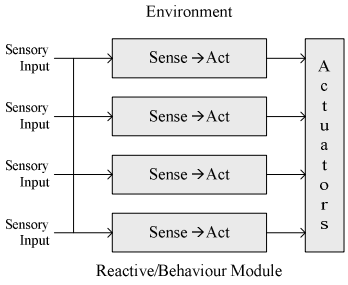
รูปที่ 3 สถาปัตยกรรมการควบคุมหุ่นยนต์เคลื่อนที่แบบ Deliberative

ในการวางแผนเส้นทางและตัดสินใจเลือกการกระทำให้หุ่นยนต์นั้น ระบบจำเป็นต้องมีโมเดลของสภาพแวดล้อมที่มีความถูกต้องแม่นยำสูงและมีความทันสมัยตลอดเวลา ทั้งนี้เนื่องจากหุ่นยนต์จำเป็นที่จะต้องคาดเดาเหตุการณ์ที่กำลังจะเกิดขึ้นเพื่อใช้เป็นข้อมูลในการวางแผนเส้นทางและเลือกการกระทำ ในกรณีที่ระบบมีโมเดลที่ถูกต้องและทันสมัย และมีเวลามากพอที่จะวางแผนและตัดสินใจเลือกการกระทำ หุ่นยนต์จะสามารถทำงานได้อย่างมีประสิทธิภาพ แต่ในโลกแห่งความเป็นจริง สภาพแวดล้อมบริเวณที่หุ่นยนต์ทำงานมักจะมีการเปลี่ยนแปลงอยู่เสมอ อีกทั้งหุ่นยนต์ก็มีการเคลื่อนที่อยู่ตลอดเวลา ทำให้ระบบไม่สามารถที่จะปรับปรุงโมเดลของสภาพแวดล้อมให้ทันสมัยได้ทันต่อการเปลี่ยนแปลงดังกล่าว ซึ่งจะส่งผลให้ระบบไม่สามารถวางแผนเส้นทางและเลือกการกระทำที่เหมาะสมเพื่อตอบสนองต่อการเปลี่ยนแปลงนั้นๆ ได้ จึงส่งผลให้มักเกิดความผิดพลาดจากการใช้งานสถาปัตยกรรมรูปแบบนี้ ยกตัวอย่างเช่น เกิดการชนกับสิ่งกีดขวางที่เคลื่อนตัวเข้ามาขวาง /หรือถูกนำมาวางขวางทิศทางการเคลื่อนที่ของหุ่นยนต์ เนื่องจากหุ่นยนต์ไม่สามารถปรับปรุงโมเดลได้ทันต่อการเข้ามาของสิ่งกีดขวางนั้น จึงไม่สามารถวางแผนเส้นทางการหลบหลีกได้

ยิ่งไปกว่านั้น เนื่องจากสถาปัตยกรรมแบบ Deliberative นั้นมีการดำเนินงานในลักษณะที่ต่อเนื่องกันเป็นลำดับ) Sense → Plan → Act) หากมีโมดูลใดโมดูลหนึ่งเกิดความเสียหายก็จะทำให้ทั้งระบบหยุดทำงานและส่งผลให้หุ่นยนต์ไม่สามารถปฏิบัติงานต่อไปได้ ด้วยเหตุผลดังที่กล่าวมาข้างต้น สถาปัตยกรรมแบบ Deliberative จึงไม่นิยมนำมาใช้ในการ

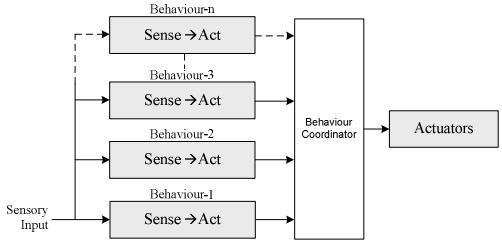
ควบคุมหุ่นยนต์เคลื่อนที่ที่ปฏิบัติงานภายในบริเวณที่มีการเปลี่ยนแปลงของสภาพแวดล้อมตลอดเวลา[8]

สำหรับสถาปัตยกรรมการควบคุมหุ่นยนต์เคลื่อนที่แบบ Reactive หรืออาจเรียกอีกอย่างว่าสถาปัตยกรรมแบบ Behaviour based นั้นจะประกอบด้วยโมดูล Sense-Act (หรือโมดูลพฤติกรรม) หลากหลายโมดูลดังแสดงในรูปที่ 4 โดยแต่ละโมดูลจะทำการรับข้อมูลของสภาพแวดล้อมรวมทั้งตำแหน่งและขนาดของสิ่งกีดขวางมาจากเซนเซอร์ (Sense) แล้วทำการตอบสนองทันที (Act) เช่น สั่งให้หุ่นยนต์เคลื่อนที่หลบหลีกสิ่งกีดขวางดังกล่าว ซึ่งจะเห็นได้ว่าในสถาปัตยกรรมรูปแบบนี้ หุ่นยนต์จะไม่มีการคิดหรือวางแผนการทำงานเหมือนสถาปัตยกรรมแบบ Deliberative ทำให้หุ่นยนต์สามารถปฏิบัติงานได้อย่างรวดเร็ว แม้ว่าจะอยู่ในสภาพแวดล้อมที่มีการเปลี่ยนแปลงตลอดเวลา



รูปที่ 4 สถาปัตยกรรมการควบคุมหุ่นยนต์เคลื่อนที่แบบ Reactive

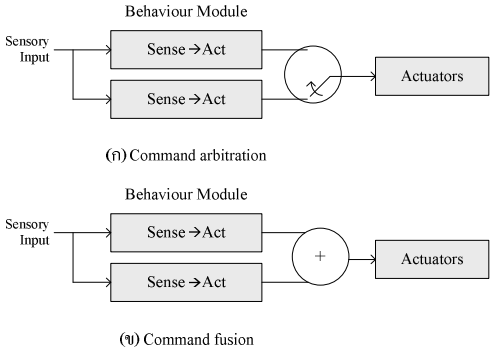
สถาปัตยกรรมรูปแบบนี้แต่ละโมดูลจะเป็นอิสระต่อกันและจะทำงานไปพร้อมๆ กัน มิได้ทำงานต่อเนื่องกันเป็นลำดับเหมือนสถาปัตยกรรมแบบ Deliberative ดังนั้นหากมีโมดูลใดโมดูลหนึ่งเกิดความเสียหาย โมดูลอื่นๆ ที่เหลืออยู่ก็จะยังสามารถทำงานต่อไปได้ จึงไม่ส่งผลให้ทั้งระบบหยุดทำงานและหุ่นยนต์ก็ยังคงปฏิบัติงานได้ต่อไป



รูปที่ 5 แผนภาพแสดงพฤติกรรมที่ใช้ในการเคลื่อนที่ของหุ่นยนต์

ในการออกแบบระบบควบคุมการเคลื่อนที่ของหุ่นยนต์นั้น สิ่งที่สำคัญอย่างหนึ่งก็คือการสร้างพฤติกรรมให้กับหุ่นยนต์ ยกตัวอย่างเช่น พฤติกรรมการเคลื่อนที่เข้าสู่เป้าหมาย ,พฤติกรรมการหลบหลีกสิ่งกีดขวาง ,พฤติกรรมการค้นหา ,พฤติกรรมการหยุดเคลื่อนที่ ฯลฯ โดยพฤติกรรมพื้นฐานเหล่านี้จำเป็นต่อการทำงานของหุ่นยนต์เป็นอย่างยิ่ง รูปที่ 5 แสดงตัวอย่างแผนภาพของระบบพฤติกรรมที่ใช้ควบคุมการเคลื่อนที่ของหุ่นยนต์ จะสังเกตได้ว่าในระบบจะมีโมดูลอยู่หนึ่งโมดูลที่ทำหน้าที่จัดการกับแต่ละพฤติกรรมของหุ่นยนต์ (Behaviour coordinator) ก่อนที่จะส่งผลลัพธ์สุดท้ายไปสั่งให้หุ่นยนต์ลงมือปฏิบัติงาน

การจัดการกับพฤติกรรมของหุ่นยนต์โดยทั่วไปนั้นสามารถแบ่งออกได้เป็น 2 ประเภทหลักๆ คือ การเลือกพฤติกรรม (Arbitration mechanism) และการรวมพฤติกรรม (Fusion mechanism) ดังแสดงในรูปที่ 6 (ก) และ (ข) ตามลำดับ ตัวอย่างของระบบควบคุมการเคลื่อนที่ของหุ่นยนต์ที่ใช้วิธีการเลือกพฤติกรรมได้แก่ การควบคุมหุ่นยนต์แบบ Subsumption [9] ซึ่งนำเสนอโดย Rodney Brooks แห่งสถาบันเทคโนโลยีแห่ง Massachusetts (MIT) ส่วนตัวอย่างของระบบที่ใช้วิธีการรวมพฤติกรรมได้แก่ การควบคุมหุ่นยนต์โดยวิธี Motor Schema [10] ซึ่งคิดค้นโดย Ronald Arkin แห่งสถาบันเทคโนโลยีแห่ง Georgia



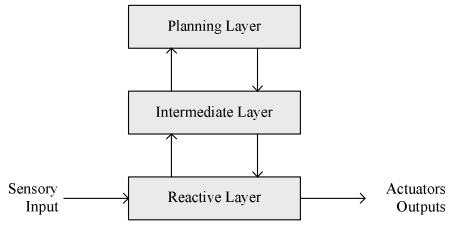
รูปที่ 6 กลไกการจัดการกับแต่ละพฤติกรรมของหุ่นยนต์

ตัวอย่างของหุ่นยนต์ที่ใช้การควบคุมแบบ Subsumption ได้แก่ หุ่นยนต์ Allen ของ Rodney Brooks [9] ซึ่งเป็นหุ่นยนต์ตัวแรกที่ใช้วิธีการควบคุมแบบนี้ หุ่นยนต์ Polly [11] ซึ่งเป็นหุ่นยนต์นำเที่ยวภายในห้องปฏิบัติการระบบปัญญาประดิษฐ์ (AI) ของสถาบัน MIT และหุ่นยนต์ Toto [12] ซึ่งเป็นหุ่นยนต์ที่มีการควบคุมแบบ Subsumption ตัวแรกที่สามารถสร้างแผนที่บริเวณที่หุ่นยนต์ปฏิบัติงานได้ ส่วนหุ่นยนต์ที่ใช้การควบคุมแบบ Motor Schema นั้นได้แก่ หุ่นยนต์ George [13] และ

หุ่นยนต์ Buzz [14] ซึ่งต่างก็ถูกพัฒนาโดย Ronald Arkin และทีมงาน

แม้ว่าสถาปัตยกรรมแบบ Reactive นี้จะสามารถทำงานได้อย่างรวดเร็ว แต่ก็มีข้อจำกัดตรงที่ไม่มีการสร้างโมเดลจำลองของสภาพแวดล้อมเพื่อใช้ในการวางแผนการทำงาน ดังนั้นสถาปัตยกรรมนี้จึงไม่เหมาะสมที่จะใช้ในงานที่ต้องการโมเดลของสภาพแวดล้อมบริเวณที่หุ่นยนต์ทำงานอยู่เพื่อใช้ในการวางแผนและตัดสินใจเลือกการกระทำ เช่น บริเวณพื้นที่ปฏิบัติการที่มีความสลับซับซ้อนมาก

สำหรับสถาปัตยกรรมแบบ Hybrid หรือแบบผสมนั้น จะรวมความสามารถหลักของสถาปัตยกรรมทั้งสองรูปแบบข้างต้นเข้าด้วยกันคือ การวางแผนการทำงานและความรวดเร็วในการการตอบสนอง เพื่อเพิ่มประสิทธิภาพการทำงานของหุ่นยนต์ อย่างไรก็ตามการรวมเอาจุดเด่นของสองสถาปัตยกรรมข้างต้นเข้าด้วยกันนั้นเป็นงานที่ค่อนข้างท้าทาย เนื่องจากว่าโมดูลต่างๆ ของระบบ Reactive นั้นต้องการที่จะตอบสนองการทำงานทันทีโดยใช้ข้อมูลภายนอกที่รับมาจากเซนเซอร์ แต่ระบบ Deliberative นั้นจะมีการวางแผนโดยใช้ทั้งข้อมูลภายในระบบเองและข้อมูลภายนอกที่รับมาจากเซนเซอร์ ดังนั้นในสถาปัตยกรรมรูปแบบนี้จึงจำเป็นต้องมีตัวกลาง (Intermediate layer) เพื่อคอยจัดการการทำงานของทั้งสองระบบให้สอดคล้องกัน ดังแสดงในรูปที่7



รูปที่ 7 สถาปัตยกรรมการควบคุมหุ่นยนต์เคลื่อนที่แบบผสม

เมื่อไรก็ตามที่หุ่นยนต์ประสบเหตุการณ์ที่ไม่ได้ถูกคาดหมายหรือวางแผนไว้โดยระบบ Deliberative (หรือ Planning layer ในรูปที่7) ระบบ Reactive ก็จะเริ่มตอบสนองต่อเหตุการณ์นั้นๆ ทันทีโดยไม่สนใจว่าระบบ Deliberative จะมีการวางแผนเส้นทางไว้อย่างไร ยกตัวอย่างเช่นในกรณีที่มีสิ่งกีดขวางเคลื่อนตัวเข้ามาขวาง /หรือถูกนำมาวางขวางทิศทางการเคลื่อนที่ของหุ่นยนต์ ระบบ Deliberative จะไม่สามารถปรับปรุงโมเดลให้ทันสมัยเพื่อวางแผนเส้นทางการหลบหลีกได้ทัน อย่างไรก็ตามระบบ Reactive จะเข้ามาทำหน้าที่แทนด้วยการสั่งให้หุ่นยนต์ทำการหลบหลีกสิ่งกีดขวางดังกล่าวทันทีโดยไม่ต้องรอการวางแผนเส้นทางใหม่จากระบบ Deliberative ในทางกลับกันเมื่อหุ่นยนต์ต้องการเคลื่อนที่จากตำแหน่งปัจจุบันไปสู่เป้าหมายใหม่ซึ่งอยู่อีกสถานที่หนึ่ง ระบบ Deliberative จะคอยชี้นำให้ระบบ Reactive ทำงานอย่างเหมาะสมและมีประสิทธิภาพโดยการออกแบบเส้นทางที่สั้นและปลอดภัยที่สุดให้กับระบบ Reactive เพื่อใช้ในการเคลื่อนหุ่นยนต์ไปสู่จุดหมายปลายทางต่อไป ตัวอย่างของระบบควบคุมการเคลื่อนที่ของหุ่นยนต์ที่ใช้สถาปัตยกรรมแบบผสม ได้แก่ ระบบ AuRA ของ Ronald Arkin [15] ระบบ Atlantis ที่พัฒนาโดย Erran Gat [16] ระบบ Planner-Reactor ของ Damian Lyons และทีมงาน [17] และระบบ PRS ของ Michael Georgeff [18]

4. การนำทางหุ่นยนต์เคลื่อนที่เข้าสู่จุดหมาย

การนำทางหุ่นยนต์เคลื่อนที่เข้าสู่จุดหมาย (Mobile Robot Navigation) เป็นหนึ่งในหัวข้อวิจัยหลักของงานวิจัยและพัฒนาทางด้านหุ่นยนต์เคลื่อนที่ โดยในขณะกำลังเคลื่อนที่เข้าสู่จุดหมายนั้น หุ่นยนต์มักประสบปัญหาการไม่รู้ว่าตัวเองอยู่ที่ตำแหน่งใดภายในพื้นที่ปฏิบัติงาน (Where am I?) ซึ่งจะส่งผลต่อเนื่องไปถึงปัญหาการที่หุ่นยนต์ไม่รู้ว่าขณะนี้ตัวเองอยู่ห่างจากจุดหมายปลายทางเท่าไรและจะเคลื่อนที่ต่อไปในทิศทางใดจึงจะสามารถเข้าสู่จุดหมายปลายทางได้สำเร็จ เพื่อความเข้าใจที่ชัดเจนขึ้นจะขอยกตัวอย่างกรณีที่หุ่นยนต์ถูกสั่งให้เคลื่อนที่เข้าหาสถานีเชื่อมต่อกับเครื่องชาร์จแบตเตอรี่ (Docking station) ซึ่งถูกกำหนดให้เป็นจุดหมายปลายทางของหุ่นยนต์ ดังนั้นหุ่นยนต์จึงจำเป็นต้องเคลื่อนที่ไปยังจุดหมายดังกล่าวเพื่อให้บรรลุหน้าที่ กรณีเช่นนี้สามารถแยกพิจารณาเป็นสถานการณ์ต่างๆ ที่อาจเกิดขึ้นได้ดังนี้

สถานการณ์ที่หนึ่ง สมมติว่าหุ่นยนต์มีแผนที่ของพื้นที่ปฏิบัติงาน (Pre-defined map) และรู้พิกัดของจุดหมายปลายทาง ซึ่งก็คือตำแหน่งของ Docking station รวมทั้งรู้ตำแหน่งปัจจุบันของตัวหุ่นยนต์เอง การเคลื่อนที่เข้าสู่จุดหมายจะสามารถทำได้โดยการวางแผนเส้นทางระหว่างตำแหน่งปัจจุบันของหุ่นยนต์กับพิกัดของจุดหมายปลายทาง แล้วจึงเคลื่อนที่หุ่นยนต์ไปตาม

เส้นทางที่ออกแบบไว้ ปัญหาลักษณะนี้ถูกเรียกว่า ปัญหาการวางแผนเส้นทาง) Path planning problem)

สถานการณ์ที่สอง สมมติว่าหุ่นยนต์มีแผนที่ของพื้นที่ปฏิบัติงานและรู้พิกัดของจุดหมายปลายทาง แต่หุ่นยนต์ไม่รู้ว่าขณะนี้ตัวมันเองอยู่ที่ตำแหน่งใดในพื้นที่ปฏิบัติงาน สิ่งแรกที่หุ่นยนต์ต้องทำก็คือ การหาตำแหน่งของตัวเองภายในแผนที่ ซึ่งปัญหาลักษณะนี้ถูกเรียกว่า ปัญหาการระบุตำแหน่งของตัวหุ่นยนต์เอง) Localization problem) เมื่อหุ่นยนต์สามารถระบุตำแหน่งของตัวเองได้แล้วก็จะสามารถเคลื่อนที่เข้าสู่จุดหมายปลายทางได้โดยการวางแผนเส้นทางการเคลื่อนที่เหมือนกับในสถานการณ์แรก

สถานการณ์ที่สาม สมมติว่าหุ่นยนต์มีแผนที่ของพื้นที่ปฏิบัติงานและรู้ตำแหน่งปัจจุบันของตัวหุ่นยนต์เอง แต่หุ่นยนต์ไม่รู้พิกัดของ Docking station ดังนั้นหุ่นยนต์จึงจำเป็นต้องเคลื่อนที่ไปรอบๆ พื้นที่ปฏิบัติงานเพื่อทำการค้นหา Docking station อย่างไรก็ตามเนื่องจากมีแผนที่ของพื้นที่ปฏิบัติงานอยู่ หุ่นยนต์สามารถใช้แผนที่ดังกล่าวในการวางแผนเส้นทางเพื่อให้การค้นหา Docking station นั้นครอบคลุมพื้นที่ทั้งหมด ซึ่งปัญหาลักษณะนี้ถูกเรียกว่า ปัญหาการสืบค้น (Coverage problem)

สถานการณ์ที่สี่ สมมติว่าหุ่นยนต์ไม่มีแผนที่ของพื้นที่ปฏิบัติงาน ในกรณีนี้หุ่นยนต์จำเป็นที่จะต้องสร้างแผนที่ขึ้นเองในขณะที่กำลังเคลื่อนที่ไป ปัญหาดังกล่าวถูกเรียกว่า ปัญหาการสร้างแผนที่ (Mapping problem) อย่างไรก็ตามการไม่มีแผนที่มิได้แปลว่าหุ่นยนต์ไม่สามารถรู้ตำแหน่งของตัวเองได้ ทั้งนี้เนื่องจากว่าหุ่นยนต์อาจจะประมาณค่าตำแหน่งของตัวเองได้ ตัวอย่างเช่นในกรณีที่เซนเซอร์ของหุ่นยนต์ตรวจพบสถานที่หรือวัตถุบางประเภทที่มีลักษณะเด่นหรือใช้เป็นจุดสังเกตได้)Landmark) เช่น มุมของห้อง ประตูทางเข้า หรือ Docking station) จุดหมายปลายทาง (ซึ่งสถานที่หรือวัตถุเหล่านี้อาจมีการกำหนดพิกัดให้ระบบทราบไว้ล่วงหน้าแล้ว (Pre-defined position) หุ่นยนต์จึงสามารถใช้พิกัดดังกล่าวในการอ้างอิงเพื่อประมาณค่าตำแหน่งของตัวเองได้

สถานการณ์สุดท้าย สมมติว่าหุ่นยนต์ไม่มีแผนที่ของพื้นที่ปฏิบัติงานและไม่สามารถรู้หรือประมาณได้ว่าขณะนี้ตัวเองอยู่ที่ตำแหน่งใดในพื้นที่ปฏิบัติงาน หุ่นยนต์จำเป็นต้องเริ่มต้นการทำงานโดยการสร้างแผนที่ขึ้นมาในขณะที่กำลังเคลื่อนที่ไปพร้อมกับพยายามหาตำแหน่งของตัวเองรวมทั้งทำการค้นหาจุดหมายปลายทางไปด้วย ปัญหาลักษณะนี้เรียกว่า ปัญหาการสร้างแผนที่และระบุตำแหน่งของตัวเองในเวลาเดียวกัน (Simultaneous Localization And Mapping (SLAM) problem) โดยที่ปัญหาดังกล่าวเป็นปัญหาที่ค่อนข้างวกวนคล้ายกับปัญหาที่ว่า “ไก่กับไข่อะไรเกิดก่อน” ทั้งนี้เพราะว่าการที่จะสร้างแผนที่นั้นหุ่นยนต์จำเป็นที่จะต้องรู้ตำแหน่งของตัวเองก่อนเพื่อใช้เป็นพิกัดอ้างอิง อย่างไรก็ตามการที่หุ่นยนต์จะรู้ตำแหน่งของตัวเองได้นั้น หุ่นยนต์จำเป็นต้องมีแผนที่ ดังนั้นหุ่นยนต์จึงจำเป็นต้องทำงานทั้งสองอย่างไปพร้อมๆ กัน

ปัญหาทั้งหมดที่กล่าวถึงข้างต้นคือส่วนประกอบของปัญหาการนำทางหุ่นยนต์เคลื่อนที่เข้าสู่จุดหมาย โดยที่แต่ละปัญหานั้นได้ถูกหยิบยกขึ้นมาทำการศึกษาวิจัยอย่างกว้างขวาง ส่งผลให้มีการเผยแพร่บทความทางวิชาการที่นำเสนอเทคนิคและวิธีการใหม่ๆ ที่ใช้ในการแก้ปัญหาดังกล่าวออกมาเป็นจำนวนมาก อย่างไรก็ตามในบทความนี้จะขอยกเอาปัญหาการวางแผนเส้นทาง (Path planning problem) และปัญหาการระบุตำแหน่งของตัวหุ่นยนต์เอง(Localization problem) มานำเสนอ รวมทั้งจะขยายขอบเขตการนำเสนอให้ครอบคลุมไปถึงเทคนิคและวิธีการหลบหลีกสิ่งกีดขวาง (Obstacle avoidance techniques) เนื่องจากเรื่องดังกล่าวนั้นถือได้ว่าเป็นเรื่องหลักของงานวิจัยและพัฒนาทางด้านหุ่นยนต์เคลื่อนที่ โดยอย่างน้อยที่สุดหุ่นยนต์เคลื่อนที่ขั้นพื้นฐานทุกตัวจะต้องจัดการกับปัญหาดังกล่าวได้ อย่างไรก็ตามรายละเอียดของปัญหาอื่นๆ ที่มิได้ถูกนำเสนอไว้ในบทความนี้สามารถศึกษาเพิ่มเติมได้จากวรรณกรรมและบทความทางวิชาการต่างๆ ทางด้านหุ่นยนต์เคลื่อนที่8] , [23-19

4.1 การระบุตำแหน่งของหุ่นยนต์เคลื่อนที่

โดยทั่วไปแล้วการระบุตำแหน่งของหุ่นยนต์ (Localization) สามารถทำได้โดยใช้วิธีการพื้นฐานที่เรียกว่า Odometry ซึ่งก็คือการระบุว่าหุ่นยนต์เคลื่อนที่ไปเป็นระยะทางเท่าใด อยู่ที่ตำแหน่งและทิศทางใดเมื่อเทียบกับตำแหน่งเริ่มต้น การทำงานของ Odometry จะนำเซนเซอร์ประเภท Optical Encoder มาใช้ในการนับจำนวนรอบของการหมุนที่แต่ละล้อของหุ่นยนต์ จากนั้นจะนำข้อมูลที่ได้มาแปลงกลับเป็นค่าระยะทาง ตำแหน่งและทิศ

ทางการเคลื่อนที่ คำว่า Odometry มีรากศัพท์มาจากคำในภาษากรีก 2 คำ คือคำว่า *hodos* ซึ่งมีความหมายว่า “การเดินทาง” และคำว่า *metros* ซึ่งแปลว่า “การวัด” ดังนั้นคำว่า Odometry นี้จึงมีความหมายว่า “การวัดระยะการเดินทาง” [8]

อย่างไรก็ตาม การระบุตำแหน่งของหุ่นยนต์ด้วยวิธี Odometry นั้นมีความถูกต้องแม่นยำค่อนข้างต่ำ ทั้งนี้เกิดจากปัญหาทางกายภาพของตัวเซนเซอร์เอง ปัญหาการลื่นไถล (Slip) รวมถึงปัญหาสัญญาณรบกวนขณะปฏิบัติงาน จึงทำให้ระบบไม่สามารถให้ข้อมูลที่ถูกต้องสมบูรณ์ได้ในแต่ละช่วงเวลาที่หุ่นยนต์เคลื่อนที่ไป ดังนั้นเมื่อหุ่นยนต์เคลื่อนที่ไปเป็นระยะทางที่มากขึ้น ระบบก็จะยิ่งสะสมความผิดพลาดของข้อมูลมากขึ้นเรื่อยๆ เช่นกัน ซึ่งอาจส่งผลให้หุ่นยนต์ทำงานผิดพลาดได้ ดังนั้นจึงจำเป็นต้องมีการประยุกต์ใช้เทคนิคอื่นๆ ควบคู่ไปกับวิธี Odometry เพื่อชดเชยความผิดพลาดดังกล่าวและเพื่อเพิ่มประสิทธิภาพของการทำงานให้ดียิ่งขึ้น

โดยปกติแล้ว การระบุตำแหน่งของหุ่นยนต์ด้วยวิธี Odometry จะใช้พิกัดของจุดเริ่มต้นของหุ่นยนต์เป็นตำแหน่งอ้างอิงในการคำนวณหาตำแหน่งและระยะทางการเคลื่อนที่ของหุ่นยนต์ สำหรับวิธีการระบุตำแหน่งวิธีอื่นๆ ที่ถูกพัฒนาขึ้นเพื่อใช้งานควบคู่ไปกับวิธี Odometry นั้นจะมีขั้นตอนการทำงานที่สลับซับซ้อนมากขึ้น ยกตัวอย่างเช่น ในกรณีที่หุ่นยนต์มีแผนที่ของพื้นที่ปฏิบัติงาน (Pre-defined map) และมีเซนเซอร์สำหรับตรวจจับระยะของวัตถุ/หรือสิ่งกีดขวาง อาทิ เลเซอร์วัดระยะ (Laser range finder) และ/หรืออัลตราโซนิคเซนเซอร์ (Ultrasonic sensor) หุ่นยนต์จะใช้เซนเซอร์ดังกล่าวในการหาทิศทางและระยะห่างระหว่างตัวหุ่นยนต์เองกับวัตถุ/หรือสิ่งกีดขวางที่อยู่ในบริเวณพื้นที่ปฏิบัติงาน เช่น ระยะห่างจากมุมห้องด้านขวามือ ระยะห่างจากกำแพงด้านหน้าและประตูทางเข้า เป็นต้น จากนั้นจะนำข้อมูลที่ได้มาเปรียบเทียบกับตำแหน่งของมุมห้องด้านขวามือ กำแพงด้านหน้าและประตูทางเข้าที่ระบุอยู่ในแผนที่ที่หุ่นยนต์มีอยู่ ด้วยวิธีการดังกล่าวหุ่นยนต์จะสามารถประมาณค่าตำแหน่งของตัวเองภายในพื้นที่ปฏิบัติงานได้ สำหรับตัวอย่างงานวิจัยเกี่ยวกับการหาตำแหน่งของหุ่นยนต์โดยใช้แผนที่ดังที่กล่าวมานั้น สามารถหาดูได้จาก [24-29]

อย่างไรก็ตามในกรณีที่หุ่นยนต์ไม่มีแผนที่ของพื้นที่ปฏิบัติงาน แต่รู้ตำแหน่งที่แน่นอนของวัตถุบางอย่างที่ใช้เป็นจุดสังเกต (Landmark) หุ่นยนต์ก็ยังสามารถใช้ทิศทางและระยะห่างระหว่างตัวเองกับ Landmark ดังกล่าวที่ได้มาจากเซนเซอร์ในการประมาณค่าตำแหน่งได้ ยกตัวอย่างเช่น หากหุ่นยนต์รู้พิกัดของ Docking station ซึ่งถูกระบุไว้ล่วงหน้าในโปรแกรมควบคุม (Pre-defined position) รวมทั้งรู้ทิศทางและระยะห่างระหว่างตัวหุ่นยนต์เองและ Docking station ซึ่งได้มาจากเซนเซอร์ หุ่นยนต์จะสามารถใช้การคำนวณตรีโกณมิติในการประมาณค่าตำแหน่งปัจจุบันของตัวเองได้ เป็นต้น ตัวอย่างของวิธีการหาตำแหน่งของหุ่นยนต์โดยใช้ Landmark นั้นสามารถหาดูได้จาก[30-33]

4.2 การวางแผนเส้นทางการเคลื่อนที่ของหุ่นยนต์

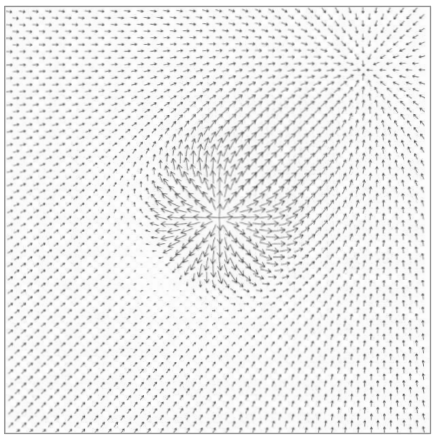
ในการปฏิบัติงานของหุ่นยนต์นั้นจำเป็นต้องมีการวางแผนเส้นทางการเคลื่อนที่ (Path Planning) จากจุดเริ่มต้นไปสู่จุดหมายปลายทางเพื่อความสะดวก รวดเร็วและเพื่อความปลอดภัยในการปฏิบัติงาน ซึ่งการวางแผนเส้นทางการเคลื่อนที่นั้น หุ่นยนต์จำเป็นต้องรู้ตำแหน่งปัจจุบันของตัวเองและพิกัดของจุดหมายปลายทางที่ต้องการเคลื่อนที่ไป โดยที่ตำแหน่งทั้งสองนั้นจำเป็นที่จะต้องอยู่ในกรอบของการอ้างอิงเดียวกัน (แผนที่เดียวกัน) การวางแผนเส้นทางนั้นสามารถทำได้ง่ายหากหุ่นยนต์มีแผนที่ของพื้นที่ปฏิบัติงานรวมทั้งรู้ข้อมูลของทั้งสองตำแหน่งข้างต้น เมื่อดำเนินการวางแผน ออกแบบและตัดสินใจเลือกเส้นทางที่เหมาะสมที่สุดแล้ว หุ่นยนต์จะทำการเคลื่อนที่ไปตามเส้นทางดังกล่าวเพื่อเข้าสู่จุดหมายต่อไป

ตัวอย่างของวิธีการวางแผนเส้นทางการเคลื่อนที่โดยใช้แผนที่ ได้แก่ การออกแบบเส้นทางโดยวิธี Visibility graph [34,35] การวางแผนเส้นทางด้วย Voronoi diagram [36] เทคนิคแบบ Cell decomposition [37] การค้นหาเส้นทางที่เหมาะสมโดยอาศัยทฤษฎี A* [38]และ D* search algorithm [39] การวางแผนเส้นทางการเคลื่อนที่โดยวิธี Dynamic programming algorithm [40] และการนำเอา Wavefront algorithm มาประยุกต์ใช้ในการวางแผนเส้นทาง [41,42] เป็นต้น

4.3 การหลบหลีกสิ่งกีดขวางของหุ่นยนต์เคลื่อนที่

สิ่งที่สำคัญที่สุดอย่างหนึ่งในขณะที่หุ่นยนต์กำลังปฏิบัติหน้าที่คือการเคลื่อนที่เข้าสู่จุดหมายได้อย่างปลอดภัย ไม่มีการปะทะ ชนหรือแม้แต่สัมผัสกับสิ่งกีดขวางใดๆ ที่อยู่ในบริเวณ

พื้นที่ปฏิบัติงาน ดังนั้นการหลบหลีกสิ่งกีดขวาง (Obstacle Avoidance) จึงถือเป็นอีกหนึ่งเรื่องหลักของงานวิจัยและพัฒนาทางด้านหุ่นยนต์เคลื่อนที่ วิธีการหลบหลีกสิ่งกีดขวางของหุ่นยนต์เคลื่อนที่นั้นมีหลากหลายวิธี เริ่มตั้งแต่วิธีพื้นฐานซึ่งสามารถทำได้โดยง่ายไปจนถึงวิธีที่มีความยุ่งยากสลับซับซ้อน ตัวอย่างของทฤษฎีและวิธีการหลบหลีกสิ่งกีดขวางที่นิยมนำมาใช้ได้แก่ ทฤษฎี Potential Field Method [43]



รูปที่ 8 ทฤษฎี Potential Field Method - แรงดึงดูดจะดูดอนุภาคเข้าหาเป้าหมายซึ่งตั้งอยู่ทางด้านบนขวาของรูป ในขณะที่แรงผลักจะดันอนุภาคออกจากสิ่งกีดขวางซึ่งตั้งอยู่ตรงกลางของรูป [10]

ทฤษฎี Potential Field Method นั้นถูกพัฒนาขึ้นโดย Osama Khatib แห่งมหาวิทยาลัย Stanford เป็นทฤษฎีที่ได้รับความนิยมเป็นอย่างมากเนื่องจากมีขั้นตอนการทำงานที่ง่ายและสามารถปรับเปลี่ยนไปใช้งานกับหุ่นยนต์เคลื่อนที่ได้หลากหลายรูปแบบ รวมทั้งยังสามารถนำไปใช้งานกับหุ่นยนต์แขนกลได้อีกด้วย [43] หลักการทำงานของทฤษฎีนี้แสดงดังรูปที่ 8 โดยสมมติให้หุ่นยนต์เป็นอนุภาคเล็กๆ อนุภาคหนึ่งซึ่งจะถูกดึงดูดโดยเป้าหมาย (Goal) แต่จะถูกผลักไสโดยสิ่งกีดขวาง (Obstacles) ดังนั้นเมื่อหุ่นยนต์เริ่มต้นการทำงาน หุ่นยนต์จะเคลื่อนที่ไปได้เองตามแรงดึงดูดเข้าหาเป้าหมายโดยจะไม่มีการชนเกิดขึ้น เนื่องจากจะมีแรงอีกชุดหนึ่งคอยผลักหุ่นยนต์ให้ออกห่างจากสิ่งกีดขวาง แรงดึงดูดดังกล่าวจะหมดลง (มีค่าเป็นศูนย์) เมื่อหุ่นยนต์เคลื่อนที่เข้าสู่เป้าหมายเป็นที่เรียบร้อย อย่างไรก็ตาม ทฤษฎี Potential Field Method นั้นมีข้อเสียอยู่เช่นกัน ยกตัวอย่างเช่น อาจเกิดสถานการณ์ที่แรงดึงดูดกับแรงผลักมีขนาดเท่ากันและมีทิศทางตรงข้ามกันพอดี ในกรณีนี้จะทำให้แรงลัพธ์ที่กระทำต่อหุ่นยนต์มีค่าเป็นศูนย์ซึ่งส่งผลให้หุ่นยนต์ไม่สามารถเคลื่อนที่ต่อไปได้ ปัญหาเช่นนี้เรียกว่า Local minima problem [44] แต่ถึงแม้จะมีปัญหาดังกล่าว ทฤษฎี Potential Field Method ก็ยังคงเป็นที่นิยมใช้กันอย่างแพร่หลาย รวมทั้งมีนักวิจัยจำนวนมากได้พยายามเอาชนะปัญหาดังกล่าวโดยการประยุกต์ใช้เทคนิคการควบคุมอื่นๆ เพิ่มเติมเข้าไป [45-47]

นอกจากวิธี Potential Field Method แล้ว ยังมีวิธีการหลบหลีกอีกหลายวิธีที่นิยมนำมาใช้ในการควบคุมการเคลื่อนที่ของหุ่นยนต์ ยกตัวอย่างเช่น วิธี Vector Field Histogram (VFH) [48] ซึ่งถูกนำเสนอโดย Johann Borenstein และทีมงาน วิธี VFH+ [49] และวิธี VFH* [50] ซึ่งถูกพัฒนาต่อยอดมาจากวิธี VFH เพื่อเพิ่มประสิทธิภาพการทำงานให้ดียิ่งขึ้น ทฤษฎี Motor Schema [10] ซึ่งมีพื้นฐานมาจากวิธี Potential Field Method ทฤษฎี Bug algorithm [51] และวิธี Dynamic Window Approach (DWA) [52] เป็นต้น

5. เซนเซอร์สำหรับหุ่นยนต์เคลื่อนที่

วิวัฒนาการทางด้านเทคโนโลยีของเซนเซอร์เกิดขึ้นอย่างรวดเร็วในช่วงทศวรรษที่ผ่านมา ส่งผลให้เซนเซอร์มีราคาถูกลงและสามารถหามาใช้งานได้ง่ายยิ่งขึ้น โดยทั่วไปแล้วเซนเซอร์สามารถแบ่งออกได้เป็นสองประเภทตามลักษณะการทำงานคือ เซนเซอร์ประเภท Passive (P) กับเซนเซอร์ประเภท Active (A) เซนเซอร์ประเภท Passive จะสามารถรับทราบข้อมูลต่างๆ ของสภาพแวดล้อมได้โดยไม่ต้องส่งสัญญาณใดๆ ออกไปกระตุ้น ในขณะที่เซนเซอร์ประเภท Active นั้นจำเป็นต้องส่งสัญญาณออกไปแล้วรอการสะท้อนกลับมาของสัญญาณดังกล่าว ตัวอย่างของเซนเซอร์ประเภท Passive และ Active ที่พบเห็นโดยทั่วไปได้แก่ กล้องวิดีโอ (Computer vision) และเลเซอร์วัดระยะ (Laser range finder) ตามลำดับ

นอกจากนี้เซนเซอร์ยังสามารถแบ่งย่อยออกไปได้อีกเป็นสองประเภทตามลักษณะหน้าที่คือ เซนเซอร์ที่ใช้ตรวจสอบสถานภาพภายในตัวหุ่นยนต์เอง (Proprioceptive) เช่น ปริมาณพลังงานที่เหลืออยู่ หรือความเร็วในการเคลื่อนที่ กับเซนเซอร์ที่ใช้ตรวจสอบสภาพแวดล้อมบริเวณที่หุ่นยนต์ปฏิบัติงานอยู่ (Exteroceptive) เช่น ระยะห่างระหว่างหุ่นยนต์กับสิ่งกีดขวาง

เป็นต้น ตารางที่ 1 แสดงรายการของเซนเซอร์ที่ถูกนำมาใช้งานร่วมกับหุ่นยนต์เคลื่อนที่

ตารางที่ 1 เซนเซอร์ชนิดต่างๆ ที่ใช้งานร่วมกับหุ่นยนต์เคลื่อนที่ [37]

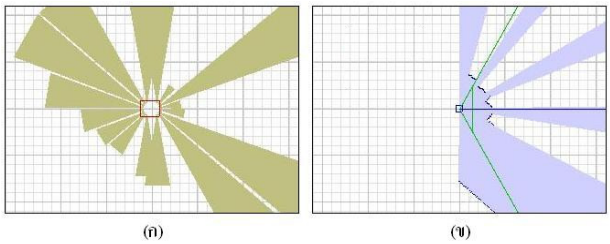
Classification	Sensor System	PC or EC	A or P
Tactile sensors (detection of physical contact or closeness; security switches)	Contact switches, bumpers	EC	P
	Optical barriers	EC	A
	Noncontact proximity sensors	EC	P
Wheel/motor sensors (wheel/motor speed and position)	Brush Encoders	PC	P
	Potentiometers	PC	P
	Synchros, resolvers	PC	A
	Optical encoders	PC	A
	Magnetic encoders	PC	A
	Inductive encoders	PC	A
	Capacitive encoders	PC	A
Heading sensors (orientation of the robot in relation to a fixed reference frame)	Compass	EC	P
	Gyroscopes	PC	P
	Inclinometers	EC	A/P
Ground based beacons (localization in a fixed reference frame)	GPS	EC	A
	Active Optical or RF beacons	EC	A
	Active ultrasound beacons	EC	A
	Reflective beacons	EC	A
Active ranging (reflectivity, time-of-flight, and geo-metric triangulation)	Reflectivity sensors	EC	A
	Ultrasonic sensor	EC	A
	Laser rangefinder	EC	A
	Optical triangulation (1D)	EC	A
	Structures light (2D)	EC	A
Motion/speed sensors (speed relative to objects)	Doppler radar	EC	A
	Doppler sound	EC	A
Vision-based sensors (visual ranging, whole-image analysis, segmentation, object recognition)	CCD/CMOS camera(s)	EC	P
	Visual ranging packages	EC	P
	Object tracking packages	EC	P

6. ซอฟต์แวร์สำหรับหุ่นยนต์เคลื่อนที่

การเขียนโปรแกรมเพื่อควบคุมการทำงานของหุ่นยนต์นั้นเป็นงานที่ท้าทาย ยิ่งไปกว่านั้นการเชื่อมต่อระหว่างโปรแกรมที่พัฒนาขึ้นกับฮาร์ดแวร์บนตัวหุ่นยนต์นั้นก็ถือได้ว่าเป็นงานที่ยุ่งยากเช่นเดียวกัน ดังนั้นซอฟต์แวร์หลากหลายชนิดจึงได้ถูกพัฒนาขึ้นมาเพื่อนำเสนอช่องทางการเขียนโปรแกรมและการเชื่อมต่อที่ง่ายขึ้น โดยในปี 1965 ซอฟต์แวร์ชื่อ DENDRAL [53] ซึ่งถือได้ว่าเป็นระบบผู้เชี่ยวชาญ (Expert system) ระบบแรกได้ถูกพัฒนาขึ้น หลังจากนั้นซอฟต์แวร์หลากหลายรูปแบบที่ใช้งานกับหุ่นยนต์เคลื่อนที่ก็ถูกพัฒนาขึ้นตามมา อาทิเช่น CARMEN [54] ARIA [55] Miro [56] Player/Stage [57] และ Orca [58]

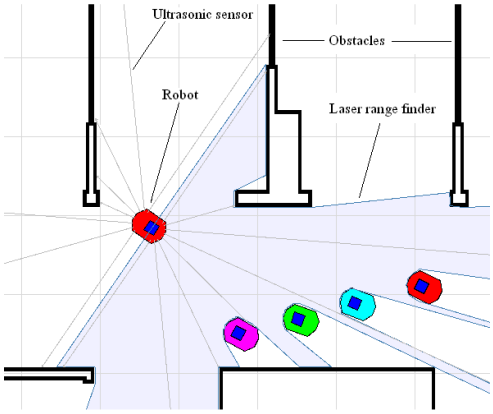
ซอฟต์แวร์ CARMEN นั้นถูกพัฒนาขึ้นมาสำหรับใช้ควบคุมหุ่นยนต์ทีละตัว ไม่สามารถใช้ควบคุมหุ่นยนต์หลายตัวให้ปฏิบัติงานพร้อมกันได้ CARMEN ประกอบด้วยฟังก์ชั่นพื้นฐานของการนำทางหุ่นยนต์เคลื่อนที่เข้าสู่จุดหมาย อาทิเช่น ฟังก์ชั่นควบคุมการทำงานของตัวหุ่นยนต์และเซนเซอร์ ฟังก์ชั่นการหลบหลีกสิ่งกีดขวาง การระบุตำแหน่งและการวางแผนเส้นทางการเคลื่อนที่ ในขณะที่ซอฟต์แวร์ ARIA นั้นถูกพัฒนาขึ้นมาใช้งานเฉพาะกับหุ่นยนต์ที่สร้างโดยบริษัท ActiveMedia Robotics เช่น Pioneer robot [1] เท่านั้น สำหรับซอฟต์แวร์ Miro นั้นใช้สถาปัตยกรรม CORBA (Common Object Request Broker Architecture) ในการเชื่อมต่อระหว่างฮาร์ดแวร์แต่ละส่วนภายในหุ่นยนต์ รวมทั้งใช้ในการติดต่อสื่อสารระหว่างหุ่นยนต์ตัวหนึ่งกับหุ่นยนต์ตัวอื่นๆ ที่ปฏิบัติงานร่วมกัน ถึงแม้ว่า Miro จะสามารถควบคุมการทำงานของหุ่นยนต์หลายๆ ตัวพร้อมกันได้ แต่กลับไม่ได้รับความนิยมเท่าที่ควร ผิดกลับซอฟต์แวร์ Player/Stage ซึ่งเป็นที่ยอมรับในวงกว้างโดยนักวิจัยและพัฒนาทางด้านหุ่นยนต์เคลื่อนที่ ในขณะที่ซอฟต์แวร์ Orca นั้นยังอยู่ในช่วงกำลังพัฒนา ซึ่งในปัจจุบันนี้ต้องยอมรับว่าซอฟต์แวร์ที่เป็นที่นิยมอย่างสูงในสังคมของนักวิจัยทางด้านหุ่นยนต์เคลื่อนที่นั้น คือ Player/Stage

Player/Stage เป็นซอฟต์แวร์ที่ถูกออกแบบมาเพื่อใช้ควบคุมหุ่นยนต์ เซนเซอร์ รวมถึงฮาร์ดแวร์ต่างๆ ภายในตัวหุ่นยนต์ การทำงานของ Player/Stage จะเป็นไปในลักษณะของ Middleware ที่เชื่อมโยงการทำงานของคอมพิวเตอร์บนตัวหุ่นยนต์กับฮาร์ดแวร์ต่างๆ เข้าด้วยกัน ซอฟต์แวร์ Player/Stage นั้นรองรับการเขียนโปรแกรมหลากหลายภาษา อาทิ ภาษา C หรือ C++ และภาษา Java เป็นต้น ซอฟต์แวร์ Player/Stage มาพร้อมกับโปรแกรมจำลองสถานการณ์แบบ 2 มิติและ 3 มิติชื่อ Stage และ Gazebo ตามลำดับ นอกจากนี้แล้ว Player/Stage ยังมีเครื่องมือที่ชื่อว่า PlayerV ซึ่งใช้แสดงสถานะของอุปกรณ์ต่างๆ ภายในตัวหุ่นยนต์ เช่น สถานะของเซนเซอร์ ในรูปแบบของภาพกราฟิก 2 มิติ ดังแสดงในรูปที่ 9



รูปที่ 9 PlayerV –)ก (แสดงตำแหน่งของหุ่นยนต์และสถานะของอัลตราโซนิคเซนเซอร์ (Ultrasonic sensor)) ข (แสดงสถานะของเลเซอร์วัดระยะ (Laser range finder) [59]

การพัฒนาโปรแกรมเพื่อควบคุมการทำงานของหุ่นยนต์นั้น หากเป็นการทดสอบบนหุ่นยนต์ของจริงก็อาจจะมีความเสี่ยงต่อการเกิดความเสียหายหากโปรแกรมยังไม่สมบูรณ์พอ แต่หากเป็นการทดสอบบนสถานการณ์จำลอง (Simulation) ปัญหาดังกล่าวก็จะหมดไป อีกทั้งยังช่วยให้การพัฒนาโปรแกรมนั้นเป็นไปได้อย่างรวดเร็วเนื่องจากการทำงานบนสถานการณ์จำลองนั้นจะช่วยลดเวลาในการทำการทดสอบโปรแกรมลง ดังนั้นจึงไม่น่าแปลกใจเลยว่า Stage หรือโปรแกรมจำลองสถานการณ์แบบ 2 มิตินั้นถูกนำมาใช้งานอย่างแพร่หลาย โปรแกรมควบคุมหุ่นยนต์ที่ถูกพัฒนาขึ้นเพื่อทดสอบการทำงานบนโปรแกรมจำลองสถานการณ์ดังกล่าวสามารถนำไปใช้ควบคุมหุ่นยนต์ภายใต้สถานการณ์จริงได้ทันทีโดยไม่ต้องมีการเปลี่ยนแปลงแก้ไขใดๆ หรือหากจำเป็นต้องมีการแก้ไขก็เป็นการแก้ไขเพียงเล็กน้อยเท่านั้น รูปที่1 0 แสดงตัวอย่างการทำงานของโปรแกรมจำลองสถานการณ์ Stage



รูปที่ 10 โปรแกรมจำลองสถานการณ์ Stage simulator] 59[

7. บทสรุป

บทความนี้กล่าวถึงพื้นฐานความรู้ที่เกี่ยวข้องกับการวิจัยและพัฒนาทางด้านหุ่นยนต์เคลื่อนที่ เริ่มตั้งแต่ประวัติความเป็นมาของหุ่นยนต์เคลื่อนที่ สถาปัตยกรรมการควบคุม และเทคนิคการนำทางหุ่นยนต์เคลื่อนที่เข้าสู่จุดหมายซึ่งประกอบด้วยวิธีการระบุตำแหน่งของหุ่นยนต์ วิธีการวางแผนเส้นทางการเคลื่อนที่ รวมทั้งวิธีการหลบหลีกสิ่งกีดขวาง นอกจากพื้นฐานความรู้ดังกล่าวแล้ว ในบทความยังได้กล่าวถึงเทคโนโลยีของเซนเซอร์ด้วย เนื่องจากหุ่นยนต์จำเป็นต้องใช้เซนเซอร์เพื่อรับรู้ข้อมูลของสภาพแวดล้อมบริเวณที่ปฏิบัติงานอยู่ รวมทั้งตำแหน่งและขนาดของสิ่งกีดขวาง เพื่อให้การเคลื่อนที่เข้าสู่จุดหมายปลายทางเป็นไปอย่างถูกต้องและปลอดภัย ซึ่งองค์ความรู้ทั้งหมดที่ได้กล่าวมานั้นจะถูกรวบรวมเข้าด้วยกันเพื่อพัฒนาเป็นโปรแกรมที่ใช้สำหรับควบคุมการทำงานของหุ่นยนต์ให้สำเร็จลุล่วงอย่างมีประสิทธิภาพต่อไป

8. เอกสารอ้างอิง

[1] ActivMedia. Pioneer 3 General Purpose Robot. 2008 [cited; Available from: http://www.activrobots.com/ROBOTS/p2dx.html.

[2] iRobot. Roomba. 2009 [cited 2010; Available from: http://www.irobot.com/uk/home_robots.cfm.

[3] Robotics, C.A. Service Robot. 2009 [cited 2011; Available from: http://www.ctasia-robotics.com/home.php.

[4] Nilsson, N.J., Shakey the robot. 1984, Artificial Intelligence Center, Computer Science and Technology Division, Stanford Research Institute.

[5] LAAS. HILARE: LAAS's first mobile robot. 2003 [cited 2010 13 September]; Available from: http://homepages.laas.fr/matthieu/robots/hilare.shtml.

[6] Earnest, L. Stanford Cart. 2005 [cited 2010 13 September]; Available from: http://www.stanford.edu/~learnest/cart.htm.

[7] Moravec, H.P., The CMU Rover, in National Conference of Artificial Intelligence (AAAI-82). 1982. p. 377-380.

[8] Mataric, M.J., The Robotics Primer. 2007: MIT Press.

[9] Brooks, R.A., A Robust Layered Control System for a Mobile Robot. IEEE Journal of Robotics and Automation, 1986. RA-2(1): p. 14-23.

[10] Arkin, R.C., Behavior-Based Robotics. 1 ed. 1998: Bradford Book, MIT Press. 491.

[11] Horswill, I., Polly, A Vision-Based Artificial Agent, in The Eleventh National Conference on Artificial Intelligence (AAAI-93). 1993: Washington, DC. p. 824-829.

[12] Mataric, M.J., Integration of Representation into Goal Driven Behavior-Based Robots. IEEE Transactions on Robotics and Automation, 1992. 8(3): p. 304-321.

[13] Arkin, R.C. and R.R. Murphy, Autonomous Navigation in a Manufacturing Environment. IEEE Transactions on Robotics and Automation, 1990. 6(4): p. 445-454.

[14] Arkin, R.C., et al., Buzz: An Instantiation of a Schema-based Reactive Robotic System, in International Conference on Intelligent Autonomous Systems (IAS-3). 1993: Pittsburgh, PA. p. 418-427.

[15] Arkin, R.C., Integrating Behavioral, Perceptual, and World Knowledge in Reactive Navigation. Robotics and Autonomous System, 1990. 6: p. 105-122.

[16] Gat, E., Reliable Goal-Directed Reactive Control of Autonomous Mobile Robots. 1991, Virginia Polytechnic Institute and State University.

[17] Lyons, D.M. and A.J. Hendriks, Planning for Reactive Robot Behavior, in IEEE International Conference on Robotics and Automation. 1992: Nice, France. p. 2675-2680.

[18] Georgeff, M.P. and A.L. Lansky, Reactive Reasoning and Planning, in AAAI-87. 1987. p. 677-682.

[19] Smith, R.C. and P. Cheeseman, On the Representation and Estimation of Spatial Uncertainty. International Journal of Robotics Research, 1986. 5(4): p. 56-68.

[20] Leonard, J.J. and H.F. Durrant-Whyte, Simultaneous Map Building and Localization for an Autonomous Mobile

Robot, in IEEE/RSJ International Workshop on Intelligent Robots and Systems. 1991. p. 1442-1447.
[21] Bailey, T. and H. Durrant-Whyte, Simultaneous Localization and Mapping: Part II. IEEE Robotics and Automation Magazine, 2006: p. 108-117.
[22] Durrant-Whyte, H. and T. Bailey, Simultaneous Localization and Mapping (SLAM): Part I. IEEE Robotics and Automation Magazine, 2006. 13(2): p. 99-110.
[23] Siciliano, B. and O. Khatib, Springer Handbook of Robotics. 2008: Springer-Verlag Berlin Heidelberg.
[24] Curran, A. and K.J. Kyriakopoulos, Sensor-based self-localization for wheeled mobile robots. Journal of Robotic Systems, 1995. 12(3): p. 163-176.
[25] Horn, J. and G. Schmidt, Continuous Localization of a Mobile Robot based on 3D-Laser-Range-Data, Predicted Sensor Images, and Dead-Reckoning. Robotics and Autonomous Systems, 1995. 14: p. 99-118.
[26] Castellanos, J.A., et al., Experiments in Multisensor Mobile Robot Lacalization and Map Building, in the 3rd IFAC Symposium on Intelligent Autonomous Vehicles. 1998. p. 173-178.
[27] Arras, K.O., N. Tomatis, and R. Siegwart, Multisensor on-the-fly Localization using Laser and Vision, in the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems. 2000: Takamatsu, Japan. p. 462-467.
[28] Cassandre, A.R., L.P. Kaelbling, and J.A. Kurien, Acting under Uncertainty: Discrete Bayesian Models for Mobile-Robot Navigation, in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 1996. p. 963-972.
[29] Fox, D., W. Burgard, and S. Thrun, Markov Localization for Mobile Robots in Dynamic Environments. Journal of Artificial Intelligence Research, 1999. 11: p. 391-427.
[30] Sugihara, K., Some Location Problems for Robot Navigation using a Single Camera. Computer Vision, Graphics, and Image Processing, 1988. 42(1): p. 112-129.
[31] Hertzberg, J. and F. Kirchner, Landmark-based Autonomous Navigation in Sewerage Pipes, in the First Euromicro Workshop on Advance Mobile Robots. 1996. p. 68-73.
[32] Betke, M. and L. Gurvits, Mobile Robot Localization using Landmarks. IEEE Transactions on Robotics and Automation, 1997. 13(2): p. 251-263.
[33] Colios, C.I. and P.E. Trahanias, A Framework for Visual Landmark Identification based on Projective and Point-Permutation Invarian Vectors. Robotics and Autonomous Systems, 2001. 35(1): p. 37-51.
[34] Latombe, J.-C., Robot Motion Planning. 1991: Kluwer Academic Publishers.
[35] Fujimura, K., Time-Minimum Routes in Time-Dependent Networks. IEEE Transactions on Robotics and Automation, 1995. 11(3): p. 343-351.
[36] Garrido, S., et al., Path Planning for Mobile Robot Navigation using Voronoi Diagram and Fast Marching, in IEEE/RSJ International Conference on Intelligent Robots and Systems. 2006. p. 2376-2381.
[37] Siegwart, R. and I.R. Nourbakhsh, Introduction to Autonomous Mobile Robots. 2004: MIT Press.
[38] Shmoulian, L. and E. Rimon, A*-DFS: An Algorithm for Minimizing Search Effort in Sensor based Mobile Robot Navigation, in IEEE International Conference on Robotics and Automation. 1998: Leuven, Belgium.
[39] Stentz, A., The Focussed D* Algorithm for Real-Time Replanning, in International Joint Conference on Artificial Intelligence (IJCAI). 1995.
[40] Sutton, R.S., Planning by Incremental Dynamic Programming, in the Ninth Conference on Machine Learning. 1991: Morgan-Kaufmann. p. 353-357.
[41] Al-Jumaily, A. and C. Leung, Wavefront Propagation and Fuzzy based Autonomous Navigation. International Journal of Advanced Robotic Systems, 2005. 2(2): p. 93-102.
[42] Nattharith, P. and R. Bicker, Mobile Robot Navigation using a Behavioural Strategy, in Control and Application (CA 2009), H. Hu, Editor. 2009: Cambridge, UK.
[43] Khatib, O., Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. The International Journal of Robotics Research, 1986. 5(1): p. 90-98.
[44] Koren, Y. and J. Borenstein, Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation, in IEEE Conference on Robotics and Automation. 1991. p. 1398-1404.
[45] Huq, R., G.K.I. Mann, and R.G. Gosine, Behavior-Modulation Technique in Mobile Robotics Using Fuzzy Discrete Event System, in IEEE Transactions on Robotics. 2006. p. 903-916.
[46] Huq, R., G.K.I. Mann, and R.G. Gosine, Mobile Robot Navigation using Motor Schema and Fuzzy context dependent behavior modulation. Applied Soft Computing, 2008. 8: p. 422-436.
[47] Nattharith, P., Behaviour Modulation using Fuzzy Logic Control for Mobile Robot Navigation, in the 3rd CUTSE International Conference (CUTSE 2011). 2011. p. 110-115.
[48] Borenstein, J. and Y. Koren, The Vector Field Histogram - Fast Obstacle Avoidance for Mobile Robots. IEEE Journal of Robotics and Automation, 1991. 7(3): p. 278-288.
[49] Ulrich, I. and J. Borenstein, VFH+: Reliable Obstacle Avoidance for Fast Mobile Robots, in IEEE International Conference on Robotics and Automation. 1998: Leuven, Belgium. p. 1572-1577.
[50] Ulrich, I. and J. Borenstein, VFH*: Local Obstacle Avoidance with Look-ahead Verification in IEEE International Conference on Robotics and Automation (ICRA). 2000. p. 2505-2511.
[51] Lumelsky, V. and T. Skewis, Incorporating Range Sensing in the Robot Navigation Function. IEEE Transactions on Systems, Man and Cybernetics, 1990. 20(5): p. 1058-1069.
[52] Fox, D., W. Burgard, and S. Thrun, The Dynamic Window Approach to Collision Avoidance. IEEE Robotics and Automation Magazine, 1997. 4(1): p. 23-33.
[53] NASA. A Short History of Robots. 2008 [cited; Available from: http://prime.jsc.nasa.gov/ROV/history.html.
[54] Montemerlo, M., N. Roy, and S. Thrun, Perspectives on Standardization in Mobile Robot Programming: The Carnegie Mellon Navigation (CARMEN) Toolkit., in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2003.
[55] ActivMedia. Advanced Robotics Interface for Applications: ARIA. 2006 [cited; Available from: http://activrobots.com/SOFTWARE/aria.html.
[56] Utz, H., et al., Miro-Middleware for Mobile Robot Applications. IEEE Transactions on Robotics and Automation, 2002. 18(4): p. 493-497.
[57] Gerkey, B.P., et al., Most Valuable Player: A Robot Device Server for Distributed Control, in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2001. p. 1226-1231.
[58] Kaupp, T. Orca Robotics. 2005 [cited; Available from: http://orca-robotics.sourceforge.net/.
[59] Vaughan, R.T. and B.P. Gerkey, Really Reusable Robot Code and the Player/Stage Project. Springer Tracts on Advanced Robotics, 2006.