บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

งานวิจัยเรื่องการประยุกต์ใช้ระบบปฏิบัติการหุ่นยนต์สำหรับรถขนส่งสินค้าอัตโนมัติในโรงงานอุตสาหกรรม การศึกษาในครั้งนี้ทางผู้วิจัยได้ศึกษาค้นคว้าข้อมูลและงานวิจัยที่เกี่ยวข้องเพื่อนำมาใช้สำหรับการเป็นองค์ความรู้ในการวิจัยโดยในบทนี้จะนำเสนอรายละเอียดของทฤษฎีและงานวิจัยที่เกี่ยวข้องซึ่งประกอบไปด้วย

2.1 ระบบปฏิบัติการหุ่นยนต์ (Robot Operating System :ROS)

2.2 ทรานสฟอร์ม (Transform :TF)

2.3 ออดโดมิททรี (Odometry)

2.4 การระบุตำแหน่งพร้อมกับการสร้างแผนที่ (Simultaneous Localization and Mapping :SLAM)

2.5 การระบุตำแหน่งด้วยวิธีการมอนติคาร์โลแบบปรับตัว (Adaptive Monte Carlo Localization :AMCL)

2.6 การคำนวณแผนที่ (Costmap)

2.7 การเคลื่อนย้ายฐาน (Move Base)

2.8 ผู้วางแผน (Planner)

2.9 ไลดาร์ (Light Detection and Ranging :Lidar)

2.10 ความเร็วเชิงเส้น

2.11 ความเร็วเชิงมุม

2.12 งานวิจัยที่เกี่ยวข้อง

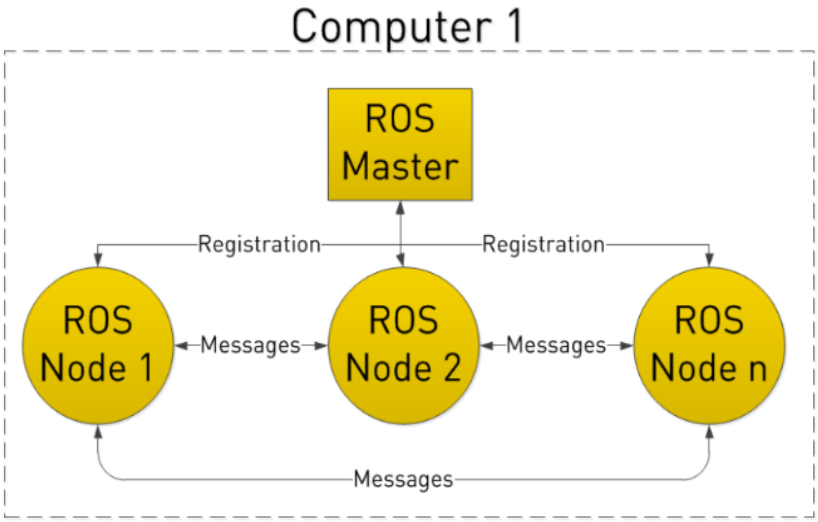
2.1 ระบบปฏิบัติการหุ่นยนต์ (Robot Operating System :ROS)

ระบบปฏิบัติการหุ่นยนต์ เป็นซอฟต์แวร์ชุดหนึ่งที่ (Framework) มีความยืดหยุ่นในการเขียนซอฟต์แวร์หุ่นยนต์ ซึ่งในปัจจุบันระบบปฏิบัติการหุ่นยนต์ทำงานบนระบบปฏิบัติการตระกูลยูนิกซ์ (UNIX-based) ภายในระบบปฏิบัติการหุ่นยนต์นั้นประกอบไปด้วยซอฟต์แวร์ต่างๆ ที่เกี่ยวข้องกับการสร้างหุ่นยนต์เป็นจำนวนมาก เช่น ซอฟต์แวร์ที่ทำหน้าที่ติดต่อกับเซ็นเซอร์ ซอฟต์แวร์ระบบการนำ

ทาง ซอฟต์แวร์การหยิบของสำหรับหุ่นยนต์และซอฟต์แวร์ทางด้านการมองเห็นของหุ่นยนต์

ปัจจุบันระบบปฏิบัติการหุ่นยนต์ถูกนำมาพัฒนาและใช้งานเพิ่มมากขึ้นอย่างต่อเนื่อง มีการนำไปใช้ในทุกระดับทั้งระดับอุตสาหกรรม การศึกษาและการวิจัย โดยมีจำนวนผู้ใช้งานเพิ่มขึ้นอย่างสม่ำเสมอ ซึ่งการประยุกต์ใช้ระบบปฏิบัติการหุ่นยนต์ในด้านต่างๆ เช่น หุ่นยนต์ที่สามารถเดินตามแผนที่แบบอัตโนมัติ หุ่นยนต์ที่สามารถหยิบจับสิ่งของได้ หุ่นยนต์ที่สามารถมีปฏิสัมพันธ์กับผู้ใช้ได้ และหุ่นยนต์ที่มีการใช้งานปัญญาประดิษฐ์ เป็นต้น

กลไกการทำงานของระบบปฏิบัติการหุ่นยนต์จะมีส่วนของโนดหลัก (Master) ทำหน้าที่ในการควบคุมจุดต่อ (Node) อื่นๆทั้งหมดในระบบ ซึ่งการทำงานของโนดหลักนั้นจะไม่เกี่ยวข้องกับการสื่อสารระหว่างจุดต่อแต่จะทำหน้าที่เป็นเพียงผู้ประสานงานและจัดเก็บข้อมูลต่างๆ ที่เกี่ยวข้องกับการติดต่อสื่อสารดังภาพที่ 2.1



ภาพที่ 2.1 ระบบปฏิบัติการหุ่นยนต์

(ที่มา : https://robohub.org/ros-101-intro-to-the-robot-operating-system/)

จุดเด่นอันสำคัญของระบบปฏิบัติการหุ่นยนต์ ได้แก่ความสามารถในเรื่องของระบบหลายภารกิจ (Multitasking) คือกระบวนการต่างๆ สามารถทำงานพร้อมกันได้ จะทำงานทางการสื่อสารระหว่างจุดต่อผ่านข้อมูลจากหัวข้อที่เผยแพร่ (Topic) ในระบบปฏิบัติการหุ่นยนต์ ระบบหนึ่งระบบของระบบปฏิบัติการหุ่นยนต์สามารถมีจุดต่อที่ทำงานได้พร้อมกันจำนวนหลายๆจุดต่อจุดต่อ (Node) หลักการของการพัฒนาซอฟต์แวร์หุ่นยนต์ด้วยระบบปฏิบัติการหุ่นยนต์ คือการสร้างพื้นที่ทำงาน (Workspace) และชุดคำสั่งสำเร็จ (Package) สำหรับการเก็บโปรแกรมที่ต้องการ

2.1.1 การสร้างพื้นที่ทำงาน (Workspace) เป็นไดเร็กทอรีหรือพื้นที่ส่วนตัวที่ใช้เก็บไฟล์ต่างๆ ที่จำเป็นสำหรับการทำงาน แต่ที่แตกต่างจากไดเร็กทอรีทั่วไปคือกฎเกณฑ์ต่างๆ ที่ถูกกำหนดมาโดยระบบปฏิบัติการหุ่นยนต์ ซึ่งผู้พัฒนาจะต้องปฏิบัติตามเพื่อให้การพัฒนาหุ่นยนต์เป็นไปตามต้องการและทำงานได้อย่างถูกต้อง นอกจากนี้ยังมีซอฟต์แวร์ที่เป็นเครื่องมือที่จำเป็นสำหรับการสร้างไฟล์ที่สามารถเรียกให้ประมวลผลได้ เครื่องมือที่ใช้ได้แก่โปรแกรม catkin_make ซึ่งทำหน้าที่แปลรหัสคำสั่งให้เป็นภาษาเครื่องที่พร้อมสำหรับการทำงานและทำหน้าที่เชื่อมโยงโปรแกรมเข้ากับไลบรารี่ต่างๆ

2.1.2 ชุดคำสั่งสำเร็จ (Package) เมื่อมีพื้นที่สำหรับการทำงานของหุ่นยนต์ภายในการสร้างพื้นที่ทำงานจะทำการสร้างชุดคำสั่งสำเร็จ ที่เก็บหน่วยย่อยของโปรแกรมที่เรียกว่าจุดต่อซึ่งจุดต่อแต่ละจุดต่อมีหน้าที่ทำงานที่แตกต่างกันขึ้นอยู่กับการออกแบบซอฟต์แวร์ของหุ่นยนต์ และแต่ละการสร้างชุดคำสั่งสำเร็จสามารถบรรจุได้หลายๆ จุดต่อเช่นกัน สามารถแก้ไข ปรับปรุง เพิ่มเติมความต้องการต่างๆ ของซอฟต์แวร์ลงไปในจุดต่อความแตกต่างระหว่างการสร้างชุดคำสั่งสำเร็จกับจุดต่อคือการใช้งานการสร้างชุดคำสั่งสำเร็จ จะใช้สำหรับเขียนโปรแกรมและเก็บไฟล์ที่เกี่ยวข้องส่วนจุดต่อจะใช้ในการประมวลผล ผู้พัฒนาซอฟต์แวร์หุ่นยนต์บนระบบปฏิบัติการหุ่นยนต์ จะต้องออกแบบส่วนของจุดต่อที่ต้องการใช้งานและนำไปเขียนในการสร้างชุดคำสั่งสำเร็จ เมื่อเขียนเสร็จทำการ Build หรือ Make ด้วยคำสั่ง catkin_make เพื่อสร้างไฟล์ที่สามารถประมวลผลได้

2.1.3 จุดต่อ (Node) ที่สร้างจากภาษาโปรแกรมต่างๆ เช่น C++ Python Java จะต้องมีการเรียกใช้ไลบรารี่ของระบบปฏิบัติการหุ่นยนต์ เช่น roscpp หรือ rospy เป็นต้น แต่ด้วยการออกแบบของระบบปฏิบัติการหุ่นยนต์ สามารถแบ่งจุดต่อออกเป็นประเภทต่างๆ ได้ดังต่อไปนี้

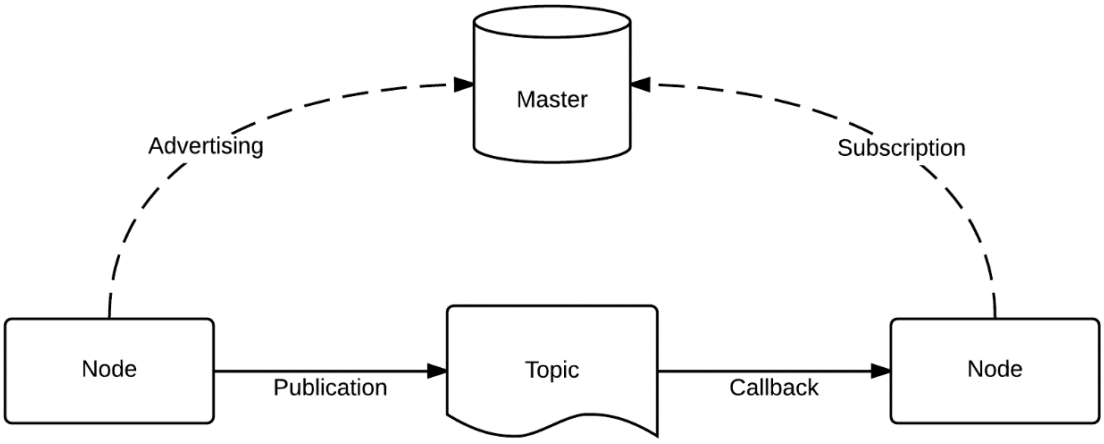
2.1.3.1 จุดต่อที่เป็นผู้ประกาศข้อมูล (Publisher)

การที่ข้อมูลสื่อสารกันระหว่างจุดต่อต่างๆและแต่ละจุดต่อทำหน้าที่เฉพาะตนเอง เช่น จุดต่อที่ชื่อว่า CameraNode ทำการประมวลผลเพื่ออ่านข้อมูลจากกล้องและส่งไปให้กับจุดต่อ ที่ต้องการใช้ข้อมูลต่อไปนี้ BorderNode ทำการสกัดหาขอบภาพเพื่อนำไปคำนวณหาตำแหน่งหรือประเภทของวัตถุที่กล้องจับภาพได้ซึ่งจุดต่อ ที่ชื่อว่า CameraNode จะเป็นผู้ประกาศข้อมูลสร้างข้อมูลเพื่อส่งข้อมูลไปให้กับ BorderNode ที่เป็นผู้ต้องการใช้ข้อมูล ข้อมูลที่ส่งมานั้นจะเรียกว่าข้อมูลจากหัวข้อที่เผยแพร่

2.1.3.2 จุดต่อที่เป็นผู้บอกรับข้อมูล (Subscriber)

ผู้ที่จะใช้ข้อมูลที่ป้อนมาจากผู้ประกาศข้อมูลจะเรียกว่าผู้บอกรับข้อมูล ซึ่งจากตัวอย่างข้างต้นจุดต่อที่ชื่อว่า BorderNode คือจุดต่อที่เป็นผู้บอกรับข้อมูล โดยรอรับข้อมูลที่ส่งมาจาก CameraNode การสื่อสารระหว่างผู้ประกาศข้อมูลกับผู้บอกรับข้อมูล นั้นจะเป็นแบบทางเดียว (One-way Communication) แต่สามารถมีจุดต่อที่เกี่ยวข้องกับข้อมูลจากหัวข้อที่เผยแพร่ได้

หลายๆจุดต่อ สำหรับระบบปฏิบัติการหุ่นยนต์จะใช้โปรโตคอล TCP/IP ในการสื่อสารระหว่างผู้ประกาศข้อมูล (Publisher) และผู้บอกรับข้อมูล (Subscriber) ดังภาพที่ 2.2



ภาพที่ 2.2 การติดต่อระหว่างผู้ประกาศข้อมูล (Publisher) และผู้บอกรับข้อมูล (Subscriber) (ที่มา : https://upload.wikimedia.org/wikipedia/commons/e/e7/ROS-master-node-topic.png)

2.1.3.3 จุดต่อที่เป็นเซอร์วิส (Service)

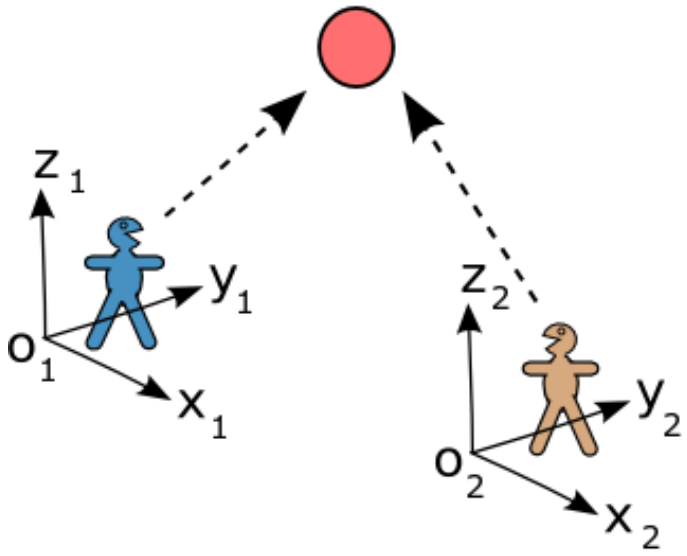
จุดต่อที่เปิดบริการงานบางอย่างเพื่อให้จุดต่ออื่นเรียกให้ทำงานตามวัตถุประสงค์ที่กำหนด เช่น จุดต่อที่เกี่ยวข้องกับแผนที่เปิดบริการให้จุดต่ออื่นๆสามารถเรียกดูแผนที่ได้

2.1.3.4 จุดต่อที่ทำงานทั่วไป

จุดต่อที่ทำงานทั่วไปเป็นการสร้างกระบวนการให้ประมวลผลในระบบปฏิบัติการตามความต้องการ เช่น การสั่งควบคุมฮาร์ดแวร์ต่างๆ การประมวลผลอัลกอริทึมต่างๆของหุ่นยนต์ ซึ่งไม่จำเป็นต้องบอกรับข้อมูลตัวข้อมูลจากหัวข้อที่เผยแพร่ใดๆหรือไม่จำเป็นต้องประกาศข้อมูลตัวข้อมูลจากหัวข้อที่เผยแพร่ออกมาหรือเปิดเซอร์วิสต่างๆ

2.2 ทรานสฟอร์ม (Transform :TF)

ทรานสฟอร์มคือระบบพิกัดที่ใช้ในการอ้างอิงถึงจุดใดๆที่ต้องการสังเกตดังภาพที่ 2.3 จากภาพจะเห็นว่ามีผู้สังเกตุ 2 คน มองลูกบอลลูกเดียวกันแต่ยืนคนละที่ ซึ่งแต่ละคนจะมีเฟรม (Frame) จุด x , y , z ที่แตกต่างกัน การแปลงข้อมูลของจุดที่ต้องการสังเกต การแปลงระยะทางที่วัดได้จากจุดใดๆ ในระบบพิกัดหรือเฟรมหนึ่งไปเป็นระยะของจุดในอีกระบบพิกัด ระบบทรานสฟอร์มมีความสำคัญมากในการทำงานของระบบปฏิบัติการหุ่นยนต์สำหรับการนำทาง (ROS Navigation) เนื่องจากทรานสฟอร์ม บอกถึงความสัมพันธ์ระหว่างระบบพิกัดต่างๆ ในตัวหุ่นยนต์



ภาพที่ 2.3 ตัวอย่างระบบพิกัดที่ใช้อ้างอิงจุด
(ที่มา : https://upload.wikimedia.org/wikipedia/commons/c/cb/Frame_of_reference.pn)

การทำงานของทรานสฟอร์มมีอยู่ 2 ภาพแบบดังต่อไปนี้

2.2.1 การแปลงแบบตายตัว (Static Transform)

การใช้งาน static transform ซึ่งสามารถเรียกให้จุดต่อชื่อ static_transfrom _ publisher ในชุดคำสั่งสำเร็จ ซึ่งทรานสฟอร์มสามารถทำงานได้โดนตรงหรืออาจเขียนลงใน launch file

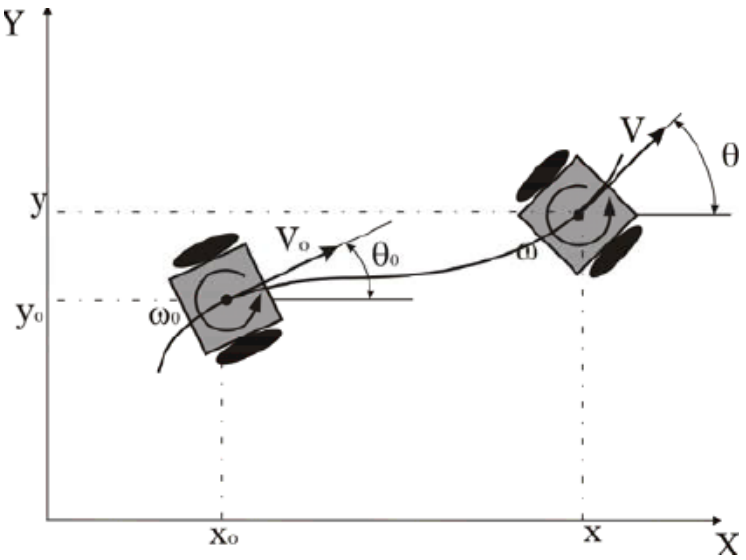
2.2.2 การแปลงแบบพลวัต (Dynamic)

ในกรณีที่ตำแหน่งของจุดที่ต้องการสังเกตมีการเคลื่อนที่จะไม่สามารถใช้การแปลงแบบตายตัวได้ เนื่องจากข้อมูลที่วัดได้มีการเปลี่ยนแปลงตลอดเวลา จึงต้องใช้การแปลงแบบพลวัตซึ่ง

ในระบบปฏิบัติการหุ่นยนต์สำหรับการนำทาง จะต้องใช้หลักการกระจายข้อมูล โดยจะกระจายข้อมูลไปในเวลาที่กำหนดในความเป็นจริงแล้วการกระจายข้อมูลใช้หลักการคล้ายๆ กับการส่งข้อมูลผ่านทางข้อมูลจากหัวข้อที่เผยแพร่ เพียงแต่ไม่จำเป็นต้องมีผู้ประกาศข้อมูล และผู้บอกรับข้อมูลที่จับคู่กัน

2.3 ออดโดมิททรี (Odometry)

การวัดการเคลื่อนที่ของหุ่นยนต์ในระบบพิกัดเฉพาะของตนเอง โดยจะเก็บข้อมูลการเคลื่อนที่ของหุ่นยนต์ ซึ่งมาจากเซ็นเซอร์ต่างๆ ในภาพที่ 2.4 แสดงให้เห็นว่าหุ่นยนต์เคลื่อนที่ไปตามแกน x y ด้วยความเร็วเชิงเส้น และความเร็วเชิงมุมที่แตกต่างกันไปในแต่ละเวลา ซึ่งออดโดมิททรีจะเก็บข้อมูลเหล่านี้ไว้ทั้งหมด



ภาพที่ 2.4 หลักการออดโดมิททรี (Odometry)

(ที่มา:https://www.researchgate.net/profile/Najdan_Vukovic/publication/253585277/figure/fig1/AS:298142923804682@1448094220692/The-model-of-differential-drive-mobile-robot-used-in-simulation.png)

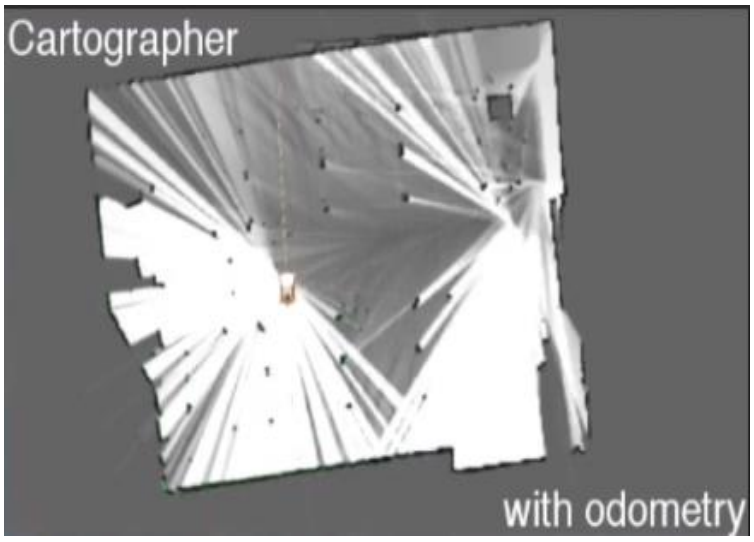
2.4 การระบุตำแหน่งพร้อมกับการสร้างแผนที่ (Simultaneous Localization and Mapping :SLAM)

การสร้างแผนที่และการค้นหาตำแหน่งของตนเองแบบฉับพลัน สำหรับหุ่นยนต์จะเป็นการสร้างแผนที่จากเซ็นเซอร์ต่างๆ การระบุตำแหน่งพร้อมกับการสร้างแผนที่ ส่วนใหญ่ทำงานบนหลักการของสถิติ โดยที่หุ่นยนต์สำรวจไปบริเวณรอบๆ และเก็บข้อมูลของสิ่งกีดขวางโดยรอบเข้ามาด้วยเซ็นเซอร์ที่ละเอียดเช่นไลดาร์ (Lidar) หรือคินเน็คท์ (Kinect) เพื่อสร้างแผนที่สำหรับระบบปฏิบัติการหุ่นยนต์ มีไลบรารี่ที่เป็นของการระบุตำแหน่งพร้อมกับการสร้างแผนที่ หลายแบบ เช่น SLAM gmapping ,hector SLAM ,Cartographer เป็นต้น ความแตกต่างระหว่างสามอย่างนี้คือ

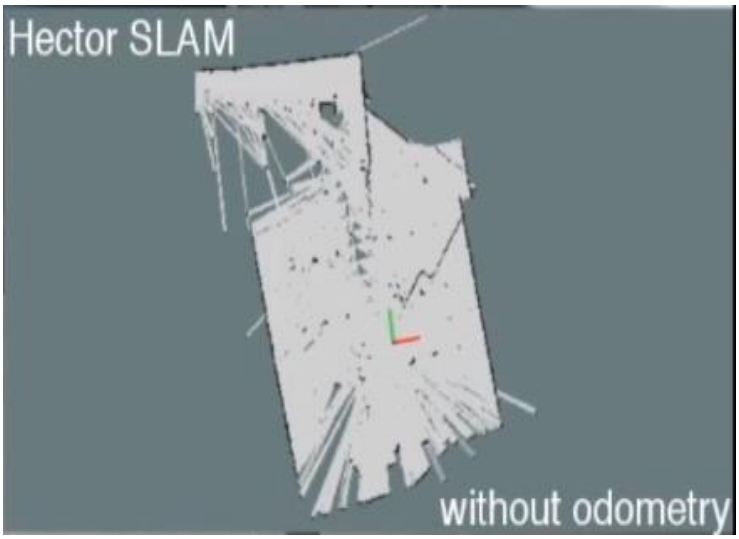
2.4.1 SLAM gmapping และ Cartographer ใช้ออดโดมิททรี ในการสร้างแผนที่ ส่วน Hector SLAM ไม่ใช่ออดโดมิททรีในการสร้างแผนที่

2.4.2 Cartographer สามารถทำแผนที่ได้แบบ 2 มิติ และ 3 มิติ ส่วน SLAM gmapping และ Hector SLAM ทำได้แค่ 2 มิติ

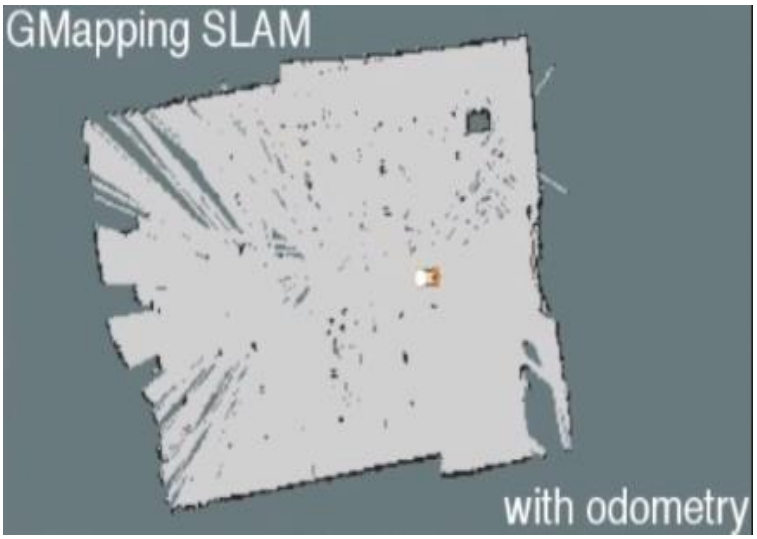
ถึงแม้ว่าการสร้างแผนที่แบบ Cartographer จะสามารถทำได้ทั้งแบบ 2 มิติ และ 3 มิติ แต่แผนที่ที่ได้มานั้นคุณภาพต่ำกว่าแผนที่ที่ถูกสร้างจาก SLAM gmapping และถึงแม้ว่าคุณภาพของแผนที่ที่ถูกสร้างจาก Hector SLAM จะมีคุณภาพต่ำกว่าทั้งสองแบบแต่มีข้อดีคือไม่จำเป็นต้องใช้ออดโดมิททรีในการสร้าง



ภาพที่ 2.5 การสร้างแผนที่จาก Cartographer
(ที่มา: https://www.youtube.com/watch?v=7iM2ynZEuf0)



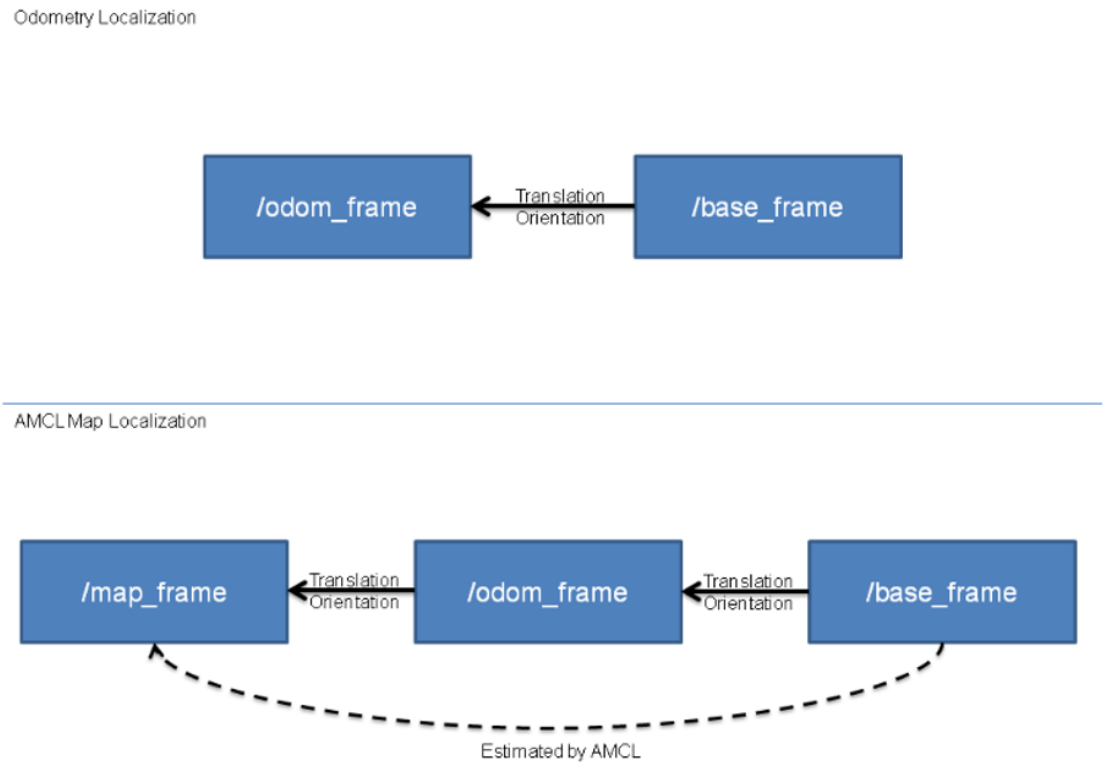
ภาพที่ 2.6 การสร้างแผนที่จาก Hector SLAM
(ที่มา: https://www.youtube.com/watch?v=7iM2ynZEuf0)



ภาพที่ 2.7 การสร้างแผนที่จาก SLAM gmapping
(ที่มา: https://www.youtube.com/watch?v=7iM2ynZEuf0)

2.5 การระบุตำแหน่งด้วยวิธีการมอนติคาร์โลแบบปรับตัว (Adaptive Monte Carlo Localization :AMCL)

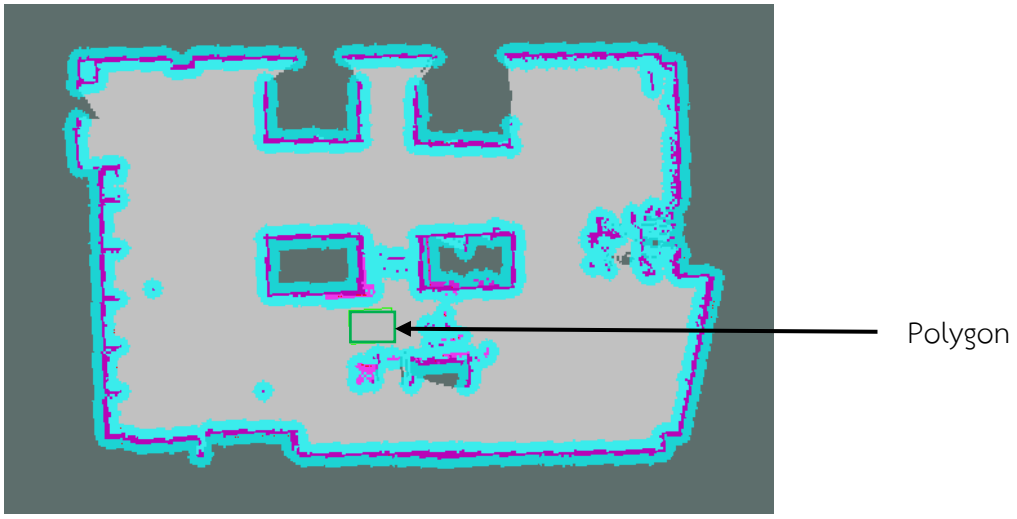
การระบุตำแหน่งด้วยวิธีการมอนติคาร์โลแบบปรับตัวเป็นไลบรารี่ที่ทำหน้าที่ในการหาตำแหน่ง ณ ปัจจุบันของหุ่นยนต์ นำเสนอโดย Dieter Fox ด้วยอัลกอริทึมที่ดัดแปลงมาจาก Monte Carlo Localization ใช้หลักการทำงานเชิงสถิติที่เรียกว่า Particle Filter ซึ่งใช้หลักการของ Genetic Algorithm คืออัลกอริทึมที่เลียนแบบหลักการคัดสรรของธรรมชาติภาพที่ 2.8 ในส่วนของออดโดมิททรีในการทรานสฟอร์มของเฟรมจะทำได้เพียงแปลงข้อมูลจาก /base_frame ที่เป็นข้อมูลตำแหน่งและทิศทางของรถแบบตายตัวไปยัง /odom_frame เป็นข้อมูลตำแหน่งและทิศทางของรถที่เปลี่ยนไปตามการเคลื่อนที่ แต่การระบุตำแหน่งด้วยวิธีการมอนติคาร์โลแบบปรับตัว สามารถทำการเชื่อมเข้ากับเฟรมของแผนที่ได้ ดังนั้นข้อดีของการระบุตำแหน่งด้วยวิธีการมอนติคาร์โลแบบปรับตัวคือการแก้ความผิดพลาดที่เกิดจากออดโดมิททรีทำให้ทราบตำแหน่งและทิศทางของหุ่นยนต์ในแผนที่ได้อย่างถูกต้อง



ภาพที่ 2.8 การแปลงทรานสฟอร์มที่มาจากการทำการระบุตำแหน่งด้วยวิธีการมอนติคาร์โลแบบปรับตัว
(ที่มา: http://wiki.ros.org/amcl)

2.6 การคำนวณแผนที่ (Costmap)

การหลบหลีกสิ่งกีดขวางที่มีอัลกอริทึมใช้หลักการของการคำนวณแผนที่ โดยจะคำนวณระยะอันตรายที่ยื่นออกมาจากสิ่งกีดขวางภาพที่ 2.9 เป็นการแสดงให้เห็นหลักการคำนวณของการคำนวณแผนที่ ส่วนที่เป็นกรอบสี่เหลี่ยมเรียกว่าโพลีกอน (Polygon) คือส่วนล่างของหุ่นยนต์ ส่วนอินเฟลชั่น (Inflation) คือภาพคล้ายก้อนเมฆที่ยื่นออกมา สีชมพูหมายถึงระยะอันตรายห้ามให้หุ่นยนต์ผ่านเข้าไปเด็ดขาด ส่วนสีฟ้าเป็นส่วนที่พอรับได้ ส่วนที่ไม่ใช่ศูนย์กลางยังพอผ่านไปได้ การทำงานของการคำนวณแผนที่จะอ่านค่าจากข้อมูลจากหัวข้อที่เผยแพร่ของไลดาร์หรือคินเน็คท์เพื่อนำข้อมูลของเซ็นเซอร์ที่อ่านได้มาคำนวณหาต้นทุนของแผนที่



ภาพที่ 2.9 ระยะปลอดภัยของการคำนวณแผนที่ (Costmap)

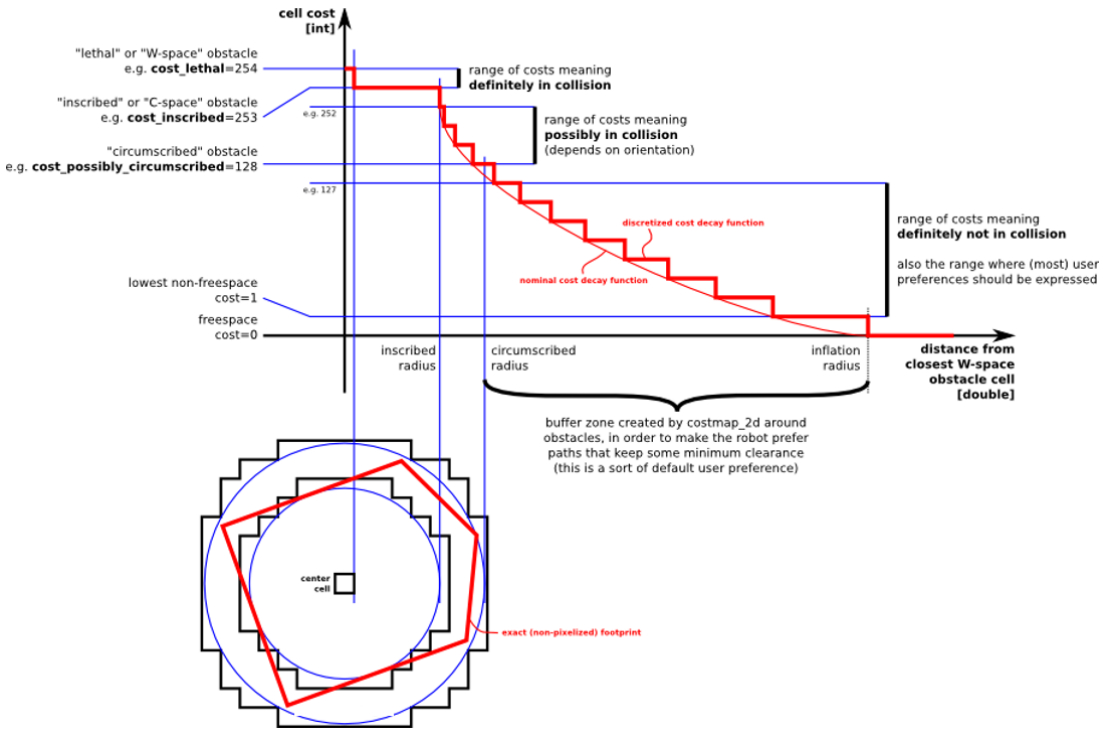
การนำข้อมูลอินเฟลชั่นจากการคำนวณแผนที่เข้ามาคำนวณโดยจะไม่หาเส้นทางเดินที่ทำให้จุดศูนย์กลางของหุ่นยนต์ผ่านเข้าไปในส่วนของอินเฟลชั่น (Inflation) แต่ถ้าหากต้องเดินไปใกล้ๆหรือเฉียดในกรณีทางแคบ มันจะสามารถให้บางส่วนของหุ่นยนต์ผ่านไปได้ จากภาพ 2.10 กราฟแกนตั้งคือค่าต้นทุนที่คำนวณได้ โดยจะมีค่าที่จะใช้เป็นฐานในการคำนวณได้แก่

2.6.1 cost_lethal คือค่าตั้งแต่ cost_inscribed ถึงค่านี้จะเป็นเขตอันตราย ห้ามผ่านเด็ดขาด หากจุดศูนย์กลางของหุ่นยนต์ผ่านเข้าในนี้ จะติดขัดแน่นอน

2.6.2 cost_inscribed คือค่าตั้งแต่ cost_possibly_circumscribed ถึงค่านี้ อาจจะมีโอกาสชนหรือติดขัดขึ้นอยู่กับทิศทางของหุ่นยนต์

2.6.3 cost_possibly_circumscribed คือค่าตั้งแต่พื้นที่ว่างเปล่าถึงค่านี้จะไม่ชนอย่าง

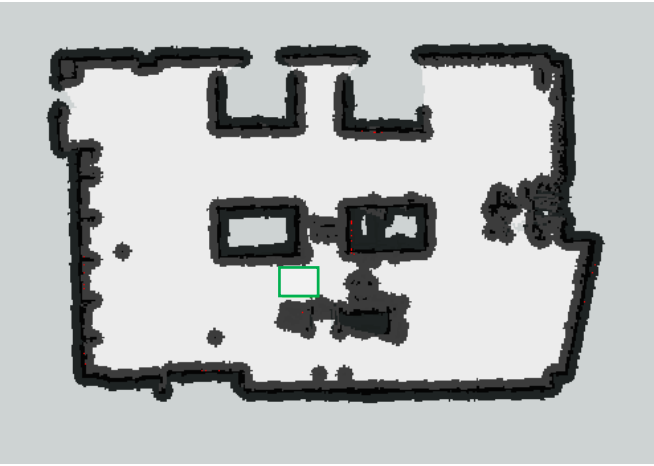
แน่นอนโดยหลักการแล้วผู้วางแผน (Planner) จะพยายามหาเส้นทางเดินใกล้ที่สุดและไกลจากเขตติดขัดมากที่สุดและเรียกเส้นทางเหล่านั้นว่าเหมาะสมที่สุด



ภาพที่ 2.10 หลักการทำงานของอินเฟลชั่น (Inflation)
(ที่มา: http://wiki.ros.org/ costmap_2d)

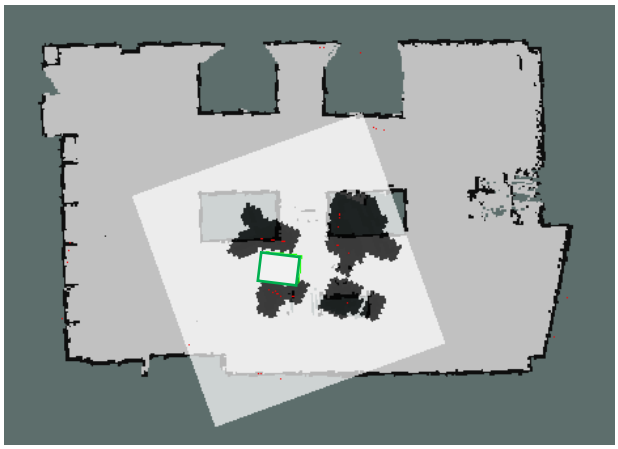
การเก็บข้อมูลของการคำนวณแผนที่จะเป็นภาพแบบของเซลล์ขนาดเล็กที่อยู่ในแผนที่ ก่อนการคำนวณแผนที่จะสร้างอินเฟลชั่นนั้น จะมีขั้นตอนของการล้างค่าและทำเครื่องหมายโดยจะทำการรับข้อมูลจากเซ็นเซอร์แบบกวาดเข้ามาแล้วตรวจสอบสิ่งกีดขวาง หากพบสิ่งกีดขวางก็จะทำเครื่องหมาย ส่วนการล้างค่าจะทำการตรวจสอบบริเวณโดยรอบแบบกวาดและจะมีระยะเวลาการล้างค่าไม่ได้ทำบ่อยเท่ากับการทำเครื่องหมาย หากพบว่าเป็นตำแหน่งเดิมที่เคยมีสิ่งกีดขวางเปลี่ยนเป็นไม่มีสิ่งกีดขวางจะทำการล้างค่า

โกลเบิลโคสต์แมฟ (global_costmap) เป็นการสร้างสิ่งกีดขวางโดยการนำแผนที่ที่ได้จาก hector SLAM มาทำการอินเฟลชั่น (inflation) เพื่อเป็นขอบเขตในการเคลื่อนที่ของรถโดยมีระยะห่างจากกำแพงหรือสิ่งกีดขวางที่กำหนด จากภาพที่ 2.11 ส่วนที่เป็นกรอบสี่เหลี่ยมเรียกว่าโพลีกอน (Polygon) คือส่วนล่างของรถและแถบพื้นที่สีดำจะเป็นขอบเขตที่รถไม่สามารถเข้าไปได้



ภาพที่ 2.11 แผนที่ที่ถูกสร้างจากโกลบอลโกลเบิลโคสต์แมฟ (global_costmap)

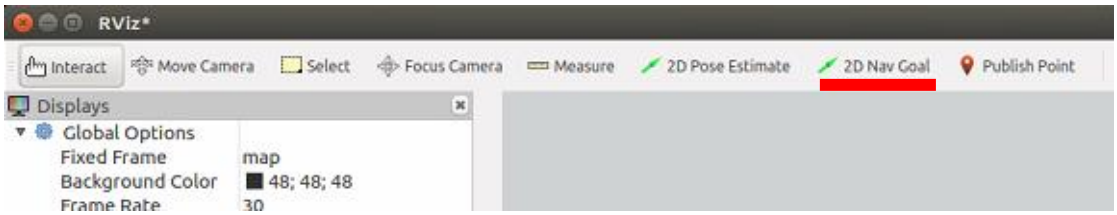
โลเคิลโคสต์แมฟ (local_costmap) เป็นการสร้างสิ่งกีดขวางในปัจจุบันที่ถูกพบในการแสกนของไลดาร์มาทำการอินเฟลชั่น (inflation) เพื่อเป็นขอบเขตในการเคลื่อนที่ของรถในระยะใกล้ จากภาพที่ 2.12 ส่วนที่เป็นกรอบสี่เหลี่ยมเรียกว่าโพลีกอน (Polygon) คือส่วนล่างของรถและแถบพื้นที่สีดำจะเป็นสิ่งกีดขวางที่เกิดขึ้นในระยะใกล้



ภาพที่ 2.12 แผนที่ที่ถูกสร้างจากโลเคิลโคสต์แมฟ (local_costmap)

2.7 การเคลื่อนย้ายฐาน (Move Base)

การหาเส้นทางการเคลื่อนที่ไปสู่ปลายทางให้กับหุ่นยนต์ และสั่งให้เคลื่อนที่ไปด้วยความเร็วและทิศทางที่กำหนด การสั่งให้หุ่นยนต์เคลื่อนที่ไปยังจุดหมายปลายทางนั้นสามารถทำได้โดยการสั่งทางโปรแกรมอาร์วิช (rviz) โดยการกดปุ่มชื่อว่า 2D nav Goal ดังภาพที่ 2.13 เมื่อคลิกที่ปุ่มแล้วนำมาลากใน Grid โดนหันทิศทางไปในทิศปลายทางที่ต้องการ จะปรากฏภาพลูกศรของปลายทางที่ต้องการให้หุ่นยนต์เคลื่อนที่หรือการทำงานตามภาพแบบของ MoveBaseAction ซึ่งเป็นการเขียนโปรแกรมเข้าไปในจุดต่อโดยตรง



ภาพที่ 2.13 แสดงปุ่ม 2D nav Goal บนโปรแกรมอาร์วิช

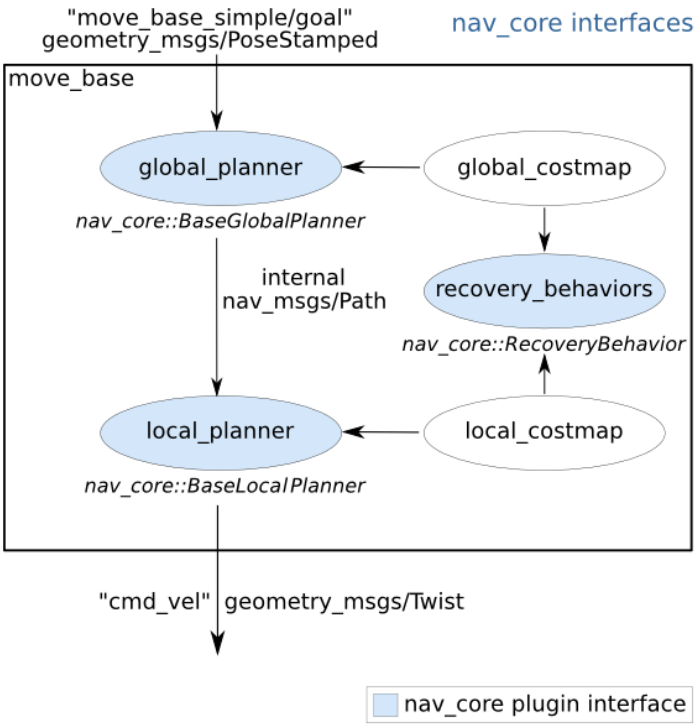
2.8 ผู้วางแผน (Planner)

เป็นอัลกอริทึมภายในจุดต่อทำหน้าที่ค้นหาเส้นทางและสั่งการหุ่นยนต์ให้เคลื่อนที่ไปสู่เป้าหมายที่ต้องการผู้วางแผนมี 2 ประเภท คือ

2.8.1 โกลเบิลแพลนเนอร์ (global_planner) คืออัลกอริทึมที่จะสร้างเส้นทางให้หุ่นยนต์ไปสู้เป้าหมายที่ต้องการ โดยจะสร้างเส้นทางตั้งแต่ต้นทางไปจนถึงปลายทาง โดยโกลเบิลแพลนเนอร์สามารถใช้แผนที่ที่ได้จากแผนที่แบบตายตัวหรือจากข้อมูลจากหัวข้อที่เผยแพร่ขึ้นอยู่กับลักษณะของสิ่งแวดล้อม หากเป็นสิ่งแวดล้อมที่มีการเปลี่ยนแปลงไม่บ่อยครั้ง ก็สามารถใช้แผนที่แบบตายตัวจาก Map server ได้

2.8.2 โลเคิลแพลนเนอร์ (local_planner) คืออัลกอริทึมที่จะสร้างเส้นทางให้หุ่นยนต์ไปสู่เป้าหมายในระดับย่อยหรือเส้นทางที่ได้จากโกลเบิลแพลนเนอร์ ซึ่งโลเคิลแพลนเนอร์จะต้องทำหน้าที่หลบหลีกสิ่งกีดขวางที่อยู่ในระยะใกล้และพยายามเคลื่อนที่ไปตามเส้นทางด้วยระยะที่สั้นที่สุด โดยปกติแล้วโลเคิลแพลนเนอร์จะไม่ใช้แผนที่แบบตายตัว เนื่องจากเน้นการหลบหลีกสิ่งกีดขวางขณะที่ทำการเคลื่อนที่

2.8.3 หลักการค้นหาเส้นทางด้วยผู้วางแผน (Planner)



ภาพที่ 2.14 หลักการค้นหาเส้นทางด้วยผู้วางแผน (Planner)

(ที่มา : http://wiki.ros.org/nav_core)

จากภาพที่ 2.14 จะเห็นว่าการเคลื่อนย้ายฐาน (Move Base) เป็นโปรแกรมที่ทำงานร่วมกันระหว่าง nav_core และการคำนวณแผนที่ (Costmap) ซึ่งใน nav_core ประกอบไปด้วยโปรแกรม 3 ส่วนดังคือ โกลเบิลแพลนเนอร์ โลเคิลแพลนเนอร์ และรีคัฟเวอรี่บีเฮฟวิเออร์ (recovery _behaviors) โดยมีหลักการทำงานเมื่อโกลเบิลแพลนเนอร์ได้รับข้อมูลจุดหมายของรถ จะนำข้อมูลการหลบหลีกสิ่งกีดขวางจากโกลเบิลแพลนเนอร์ มาคำนวณเส้นทางร่วมกับรีคัฟเวอรี่บีเฮฟวิเออร์ ซึ่งรีคัฟเวอรี่บีเฮฟวิเออร์จะรับข้อมูลทั้งโกลเบิลโคสต์แมฟและโลเคิลโคสต์แมฟมาคำนวณด้วย จากนั้นโลเคิลโคสต์แมฟจะรับเส้นทางมาคำนวณเป็นค่าความเร็วของรถและส่งข้อมูลเป็นหัวข้อที่เผยแพร่ชื่อว่า cmd_vel ในข้อมูลจากหัวข้อที่เผยแพร่ cmd_vel คือข้อมูลความเร็วเชิงเส้นและความเร็วเชิงมุม

2.9 ไลดาร์ (Light Detection and Ranging :Lidar)

การทำงานของไลดาร์นั้นจะยิงลำแสงเลเซอร์ลงไปยังพื้นผิวที่ต้องการเก็บข้อมูล หลักการคือการวัดระยะเวลาในการเดินทางของลำแสงเลเซอร์จากจุดเริ่มต้นไปยังวัตถุเป้าหมายและระยะเวลาที่ลำแสงเลเซอร์สะท้อนกลับมายังเซ็นเซอร์เริ่มต้น ซึ่งจะช่วยให้ทราบถึงสภาพแวดล้อมโดยรอบของพื้นที่ที่จะทำการเคลื่อนที่ของรถขนส่งสินค้าอัตโนมัติภาพที่ 2.15 แสดงฮาร์ดแวร์ของไลดาร์

ภาพที่ 2.15 ฮาร์ดแวร์ของไลดาร์ (Lidar)

2.10 ความเร็วเชิงเส้น

ความเร็วที่วัตถุใช้ในการเดินทางเป็นวงกลม สำหรับการเคลื่อนที่แบบวงกลมในแนวระดับจะมีค่าคงตัวเสมอ ตลอดระยะเวลาของการเคลื่อนที่ และไม่คำนึงถึงแรงเสียดทางจากอากาศ ดังสมการ

$$v_{th} = \frac{2\pi R}{t}$$

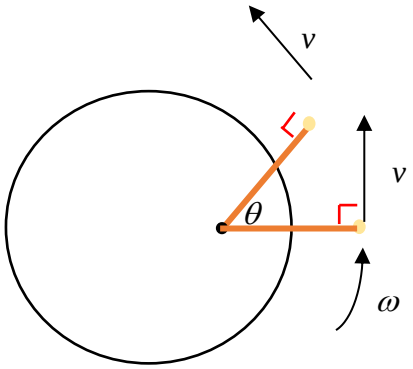
เมื่อ v_{th} คือ ความเร็วเชิงเส้น

R คือ รัศมีของล้อ

t คือ เวลา

2.11 ความเร็วเชิงมุม

มุมการเคลื่อนที่ของวัตถุที่เคลื่อนที่ไปตามหนึ่งหน่วยเวลา จากภาพที่ 2.16 หากวัตถุเคลื่อนที่กวาดมุมไปได้ θ ในเวลา t ใดๆ



ภาพที่ 2.16 มุมการเคลื่อนที่ของวัตถุที่เคลื่อนที่

จะสามารถหาความเร็วเชิงมุมได้ จากสมการนี้

$$\omega = \frac{\theta}{t}$$

เมื่อ ω คือ ความเร็วเชิงมุม

t คือ เวลา

θ คือ องศา

การคำนวณความเร็วของรถ นอกจากรถจะมีความเร็วเชิงเส้นแล้วยังมีความเร็วเชิงมุมอีกด้วย ดังนั้นจึงมีการคำนวณความเร็วเชิงเส้นและเชิงมุมดังสมการต่อไปนี้

$$v = v_{linear} + v_{th} \quad \text{และ} \quad v_{th} = \omega R$$

ดังนั้น

$$v = v_{linear} + \omega R$$

เมื่อ v_{linear} มาจากข้อมูลจากหัวข้อที่เผยแพร่ cmd_vel เป็นความเร็วเชิงเส้น

R ซึ่งในที่นี่คือครึ่งหนึ่งของระยะห่างล้อทั้งสองข้างของรถที่ใช้ในการขับเคลื่อน

2.11 งานวิจัยที่เกี่ยวข้อง

พงศ์เทพ ดวงมาศ (2550) การศึกษาและพัฒนาโปรแกรมระบบควบคุมอัจฉริยะสำหรับเอจีวีชนิดแบบลากจูง โครงสร้างของเอจีวีประกอบด้วย 3 ล้อโดยล้อขับเคลื่อนอยู่ตรงด้านหน้าซึ่งมีมอเตอร์ไฟฟ้ากระแสตรงเป็นตัวขับเคลื่อนและมีมอเตอร์ไฟฟ้ากระแสตรงอีกตัวหนึ่งเป็นตัวบังคับเลี้ยว และมี

ตัวตรวจจับตำแหน่งหรือเอนโค้ดเดอร์เป็นตัวตรวจสอบมุมในการเลี้ยว และมีล้อตามอีก 2 ล้อซึ่งแต่ละล้อมีตัวตรวจวัดตำแหน่งเพื่อตรวจสอบการเคลื่อนที่ โดยใช้ระบบนำร่อง แบบพิกัดตำแหน่ง X, Y ซึ่งเป็นการนำร่องที่ยืดหยุ่นสามารถปรับเปลี่ยนตามสายการผลิตได้ง่าย ไม่ยุ่งยากซับซ้อน ใช้พีแอลซีเป็นตัวควบคุมการทำงานของเอจีวีและใช้คอมพิวเตอร์เป็น ตัวควบคุมและรับส่งข้อมูลร่วมกับพีแอลซี การควบคุมนี้ถูกออกแบบโดยใช้ตัวควบคุมแบบพีดี เพื่อควบคุมตำแหน่งการเคลื่อนที่ของเอจีวี ในการทำงานของเอจีวีจะเริ่มต้นด้วยการป้อน เส้นทางเดินในโปรแกรมระบบควบคุมอัจฉริยะจากคอมพิวเตอร์แล้วส่งถ่ายข้อมูลจากคอมพิวเตอร์ไปพีแอลซีบนเอจีวี โดยใช้โปรโตคอลแบบมอดบัสผ่านระบบเครือข่ายไร้สายจากนั้น เอจีวีก็จะเคลื่อนที่ตามเส้นทางที่ได้รับจากผู้ใช้งาน ซึ่งตัวควบคุมแบบพีดีเป็นตัวชดเชย ค่าความเบี่ยงเบนจากค่าที่ถูกกำหนดเป้าหมายไว้ โดยในส่วนของโปรแกรมระบบควบคุม อัจฉริยะสำหรับเอจีวีจะประกอบด้วย 3 ส่วนหลัก คือ การควบคุมแบบบังคับด้วยมือ การ-ควบคุมแบบอัตโนมัติและในส่วนสุดท้ายคือส่วนแสดงผลซึ่งมีการแสดงผลด้วยกราฟเส้นและ ภาพจากกล้องซีซีดีไร้สาย ซึ่งการทดลองของงานวิจัยนี้จะให้เอจีวีเคลื่อนที่ด้วยภาพแบบเส้นทาง เดินแบบเส้นตรง ตัวเอส วงกลม และสี่เหลี่ยมโดยการเคลื่อนที่แบบเส้นตรงมีความแม่นยำ เฉลี่ย 96.478%, แบบตัวเอสมีความแม่นยำเฉลี่ย 98.5475%, แบบวงกลมมีความแม่นยำเฉลี่ย 98.7668% โดยมีความสามารถในการซ้ำตำแหน่งเดิมเฉลี่ย 97.335%, แบบสี่เหลี่ยมมีความแม่นยำเฉลี่ย 98.122% แล้วมีความสามารถในการซ้ำตำแหน่งเดิมเฉลี่ย 99.964%

ศุภมร ศรีบุญแก้วและพจน์ ตั้งงามจิตต์ (2560) งานวิจัยนี้มีวัตถุประสงค์เพื่อปรับปรุงการระบุตำแหน่งและสร้างแผนที่ 3 มิติด้วยอัลกอริทึม RGBDSLAM ให้ รองรับกับสภาพแวดล้อมที่มีวัตถุเคลื่อนที่ โดยมีแนวคิดที่จะกรองภาพที่มีวัตถุเคลื่อนที่ออกไปก่อนที่ภาพนั้นจะถูกนำ เข้าสู่กระบวนการระบุตำแหน่งและสร้างแผนที่ โดยการกรองภาพนั้นจะใช้วิธีคำนวณการติดตามการเคลื่อนที่ของ Lucas และ Kanade ในการวิเคราะห์คุณลักษณะของการเคลื่อนที่ ซึ่งได้มีการแบ่งภาพที่จะนำ มาสร้างแผนที่ออกเป็น 2 ชนิด คือ ภาพที่มีวัตถุเคลื่อนที่และภาพที่ไม่มีวัตถุเคลื่อนที่ ในวิธีการที่นำเสนอนี้ ภาพที่มีวัตถุเคลื่อนที่จะถูกกรองออกไปก่อน เข้าสู่กระบวนการ RGBDSLAM ทำให้ผลการทดลองที่ประเมินความถูกต้องและแม่นยำของการระบุตำแหน่งและสร้างแผนที่ 3 มิติจากการเปรียบเทียบค่าความผิดพลาดเฉลี่ย (RMSE) ดีขึ้นเมื่อเทียบกับวิธีการเดิมโดยผลที่ได้จากวิธีการเดิม มีค่าความผิดพลาดเฉลี่ย 0.1 ส่วนวิธีที่ได้พัฒนาขึ้นมีค่าความผิดพลาดเฉลี่ย 0.03