

บทที่ 2

ปฐกัณฑ์วรรณกรรมและงานวิจัยที่เกี่ยวข้อง

ในการวิจัยเรื่องนี้ มุ่งเน้นไปที่การ พัฒนาระบบระบุตำแหน่งระหว่างอาคาร ที่มีความแม่นยำสูง ซึ่งต้องอาศัยการบูรณาการเทคโนโลยีและแนวคิดจากหลากหลายสาขา ดังนั้นในบทนี้จึงมีวัตถุประสงค์เพื่อทบทวนวรรณกรรม เอกสารทางวิชาการ และงานวิจัยที่เกี่ยวข้องกับองค์ความรู้หลักที่ใช้ในการวิจัย เน้นการศึกษาเชิงลึกเกี่ยวกับ เทคโนโลยีการระบุตำแหน่ง และ เทคนิคการรวมข้อมูลเซนเซอร์ ที่จำเป็นต่อการสร้างระบบระบุตำแหน่งที่มีประสิทธิภาพ โดยครอบคลุมตั้งแต่พื้นฐานของระบบกำหนดตำแหน่งด้วยดาวเทียม (GNSS) ไปจนถึงการประมวลผลภาพเพื่อหาตำแหน่ง นอกจากนี้ยังรวมถึงการทำความเข้าใจกับกรอบแนวคิดหลักที่ใช้ในการผสานรวมข้อมูลจากแหล่งต่าง ๆ เพื่อให้ได้ตำแหน่งที่ถูกต้องและต่อเนื่อง ซึ่งเป็นกุญแจสำคัญในการแก้ปัญหาระบบระบุตำแหน่ง งานวิจัยนี้จะทบทวนกรอบแนวคิดและเทคนิคสำคัญดังต่อไปนี้

2.1 ORB-SLAM2

การระบุตำแหน่งและการสร้างแผนที่พร้อมกัน (SLAM) เป็นหัวข้อวิจัยที่ได้รับความนิยมอย่างมากในวงการวิทยาการคอมพิวเตอร์ด้านการมองเห็น (Computer Vision) และหุ่นยนต์ (Robotics) เทคนิค SLAM มีหน้าที่สร้างแผนที่ของสภาพแวดล้อมที่ไม่รู้จักมาก่อน และกำหนดตำแหน่งของเซ็นเซอร์ในแผนที่นั้น โดยให้ความสำคัญกับการทำงานแบบเรียลไทม์เป็นอย่างมาก ในบรรดาประเภทของเซ็นเซอร์ที่ใช้ กล้องถือว่าเป็นอุปกรณ์ที่มีราคาถูก และสามารถให้ข้อมูลที่หลากหลายเกี่ยวกับสภาพแวดล้อม ซึ่งช่วยให้สามารถจดจำสถานที่ได้อย่างแม่นยำและเชื่อถือได้ ดังนั้น SLAM แบบใช้ภาพ หรือ Visual SLAM ซึ่งกล้องเป็นเซ็นเซอร์หลัก จึงได้รับความนิยมอย่างมากในปัจจุบัน การจดจำสถานที่ (Place Recognition) เป็นองค์ประกอบสำคัญของระบบ SLAM เพื่อปิดลูป กล่าวคือ ตรวจจับเมื่อเซ็นเซอร์กลับมาสู่พื้นที่ที่เคยบันทึกไว้แล้ว และแก้ไขข้อผิดพลาดที่สะสมระหว่างการสำรวจ รวมถึงใช้สำหรับระบุตำแหน่งใหม่ของกล้อง เมื่อระบบติดตามตำแหน่งล้มเหลว เช่น จากการถูกบังหรือการเคลื่อนไหวที่รุนแรง หรือเมื่อระบบเริ่มทำงานใหม่

เดิมนั้น Visual SLAM สามารถดำเนินการได้โดยใช้เพียงกล้องแบบ Monocular [1] ซึ่งเป็นการตั้งค่าของเซ็นเซอร์ที่มีราคาถูกที่สุดและมีขนาดเล็กที่สุด อย่างไรก็ตาม เนื่องจากไม่สามารถสังเกตข้อมูลความลึกได้จากกล้องเพียงตัวเดียว ขนาด (scale) ของแผนที่และเส้นทางที่คาดการณ์ไว้จึงไม่สามารถทราบได้ ท้ายที่สุดแต่ไม่ท้ายสุด ระบบ Monocular SLAM ยังประสบปัญหาเรื่องการลอย

ของสเกล (scale drift) และอาจล้มเหลวหากมีการเคลื่อนไหวแบบหมุนล้วน ๆ ระหว่างการสำรวจ อย่างไรก็ตาม หากใช้กล้องแบบสเตอริโอ (Stereo) หรือกล้องที่ให้ข้อมูลสีและความลึก (RGB-D) ปัญหาเหล่านี้ทั้งหมดจะได้รับการแก้ไข และทำให้สามารถพัฒนาโซลูชัน Visual SLAM ที่เชื่อถือได้มากที่สุด โดยมีงานที่เกี่ยวข้องดังนี้

ระบบ Stereo SLAM โดยอิงจากเทคนิค Conditionally Independent Divide and Conquer EKF-SLAM ซึ่งสามารถทำงานในสภาพแวดล้อมขนาดใหญ่กว่าวิธีการอื่น ๆ ที่สำคัญที่สุดมันเป็นระบบ Stereo SLAM แรกที่ใช้ประโยชน์จากทั้งจุดใกล้และจุดไกล โดยใช้การพาราเมเตอร์ไรซ์ความลึกแบบผกผัน (inverse depth parametrization) สำหรับจุดที่อยู่ไกล ได้แสดงให้เห็นจากการทดลองว่า จุดต่าง ๆ สามารถถูกไตรแองกูเลต (triangulated) ได้อย่างน่าเชื่อถือ หากความลึกของจุดนั้นน้อยกว่า 40 เท่าของระยะฐานกล้อง (stereo baseline) ระบบ Stereo SLAM สมัยใหม่ส่วนใหญ่ใช้โครงสร้างแบบ Keyframe-based และทำการปรับแต่งด้วยเทคนิค Bundle Adjustment (BA) ในพื้นที่เฉพาะบางส่วน เพื่อให้สามารถขยายขนาดระบบได้ ได้ทำการปรับแต่งร่วมกันระหว่าง BA (ข้อจำกัดระหว่างจุดกับตำแหน่งของกล้อง) ในหน้าต่างด้านในของคีย์เฟรม และกราฟตำแหน่ง (pose-graph: ข้อจำกัดระหว่างตำแหน่งกล้อง) ในหน้าต่างด้านนอก โดยการจำกัดขนาดของหน้าต่างเหล่านี้ วิธีการนี้สามารถทำงานได้ในเวลาคงที่ (constant time complexity) แต่แลกมาด้วยการไม่สามารถรับประกันความสอดคล้องระดับโลกของแผนที่ได้อย่างสมบูรณ์

RSLAM ใช้การแทนตำแหน่งของจุดสังเกตและท่าทาง (poses) ในเชิงสัมพัทธ์ และทำ Bundle Adjustment แบบสัมพัทธ์ในพื้นที่ที่กำลังทำงานอยู่ (active area) ซึ่งสามารถจำกัดขอบเขตเพื่อให้ทำงานในเวลาแบบคงที่ได้ RSLAM สามารถปิดลูปได้ ซึ่งช่วยให้สามารถขยายพื้นที่ทำงานไปทั้งสองด้านของลูปได้ แต่ไม่ได้บังคับให้มีความสอดคล้องทั่วโลก (global consistency) ระบบ S-PTAM ล่าสุดทำ BA เฉพาะในพื้นที่เฉพาะ แต่ยังขาดความสามารถในการปิดลูปขนาดใหญ่ คล้ายกับแนวทางเหล่านี้ เราทำ BA ในกลุ่มของคีย์เฟรมในพื้นที่เฉพาะ เพื่อให้ความซับซ้อนของการประมวลผลไม่ขึ้นอยู่กับขนาดของแผนที่ และสามารถใช้งานในสภาพแวดล้อมขนาดใหญ่ได้ อย่างไรก็ตาม เป้าหมายของเราคือการสร้างแผนที่ที่มีความสอดคล้องระดับโลก (globally consistent) เมื่อมีการปิดลูป ระบบของเราจะจัดตำแหน่งของทั้งสองฝั่งของลูปให้ตรงกันก่อน (คล้ายกับ RSLAM) เพื่อให้การติดตามสามารถดำเนินต่อไปได้โดยใช้แผนที่เดิม และจากนั้นจึงทำ pose-graph optimization เพื่อลดความเบี่ยงเบนที่สะสมในลูป แล้วจึงตามด้วย Bundle Adjustment แบบเต็มรูปแบบ

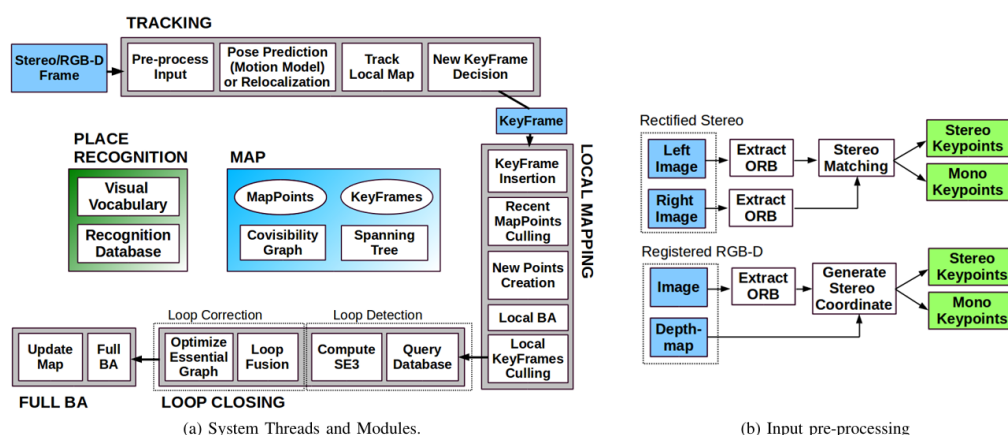
Stereo LSD-SLAM เป็นวิธีแบบกึ่งหนาแน่น (semi-dense) และตรง (direct approach) ซึ่งลดค่าความคลาดเคลื่อนของแสง (photometric error) ในบริเวณของภาพที่มีความชันสูงเนื่องจากไม่พึ่งพาการตรวจจับคุณลักษณะ (features) วิธีนี้จึงคาดว่าจะมีความทนทานมากกว่าในกรณีที่เกิด

การเบลอจากการเคลื่อนไหว (motion blur) หรือในสภาพแวดล้อมที่มีเนื้อผิวน้อย (poorly-textured environments) อย่างไรก็ตาม ในฐานะที่เป็นวิธีแบบตรง (direct method) ประสิทธิภาพของมันอาจลดลงอย่างมากหากมีผลกระทบที่ไม่ได้ถูกรวมไว้ในแบบจำลอง เช่น เอฟเฟกต์ rolling shutter หรือ การสะท้อนแบบไม่ลัมเบิร์ตเซียน (non-lambertian reflectance)

RGB-D SLAM วิธีนี้รวมข้อมูลความลึกทั้งหมดจากเซ็นเซอร์เข้าด้วยกันเป็นโมเดลเชิงปริมาตรแบบหนาแน่น (volumetric dense model) ซึ่งใช้สำหรับติดตามตำแหน่งของกล้องโดยใช้เทคนิค ICP อย่างไรก็ตาม ระบบนี้ถูกจำกัดให้อยู่ในพื้นที่ขนาดเล็ก เนื่องจากการใช้การแทนค่าด้วยรูปแบบเชิงปริมาตร และไม่มีการปิดลูป (loop closing) สามารถทำงานในสภาพแวดล้อมขนาดใหญ่ได้ โดยใช้ rolling cyclical buffer และมีการปิดลูปโดยใช้ การจดจำสถานที่ (place recognition) และการปรับแต่งกราฟท่าทาง (pose graph optimization) ระบบ RGB-D SLAM ที่มีชื่อเสียงและเป็นระบบโอเพนซอร์ส ซึ่งเป็นระบบที่อิงกับคุณลักษณะ (feature-based system) โดยที่ front-end คำนวณการเคลื่อนไหวระหว่างเฟรมแต่ละเฟรมผ่านการจับคู่คุณลักษณะ (feature matching) และ ICP ส่วน back-end ทำการปรับแต่งกราฟท่าทาง (pose-graph optimization) พร้อมข้อจำกัดจากการปิดลูป (loop closure constraints) ที่ได้จากการค้นหาด้วยวิธีฮิวริสติก (heuristic search)

จากนั้นจึงได้มีการพัฒนา ORB-SLAM2 [2] สำหรับกล้อง stereo และ RGB-D ซึ่งถูกพัฒนาต่อยอดจากระบบ ORB-SLAM สำหรับกล้อง monocular ที่ใช้คุณลักษณะ (feature-based) โดยส่วนประกอบหลักของระบบนั้นสามารถดูได้ในรูปที่ 2.1 ระบบประกอบด้วยเธรดหลัก 3 ตัวที่ทำงานแบบขนานกัน 1) การติดตาม (tracking) ใช้ในการระบุตำแหน่งกล้องในทุกเฟรม โดยค้นหาการจับคู่ของคุณลักษณะกับแผนที่ท้องถิ่น และลดค่าความผิดพลาดจากการฉายซ้ำ (reprojection error) ด้วยการประยุกต์ใช้ motion-only BA 2) การทำแผนที่ท้องถิ่น (local mapping) ใช้จัดการแผนที่ท้องถิ่นและปรับแต่งให้เหมาะสม โดยทำการ local BA 3) การปิดลูป (loop closing) ใช้ตรวจจับลูปขนาดใหญ่และแก้ไขความคลาดเคลื่อนที่สะสม (drift) ด้วยการทำ pose-graph optimization เธรดนี้จะเรียกใช้เธรดที่สี่เพื่อทำ full BA หลังจากการทำ pose-graph optimization เพื่อคำนวณโครงสร้างและการเคลื่อนที่ที่เหมาะสมที่สุดของระบบ ระบบได้ฝังโมดูล การจดจำสถานที่ (Place Recognition) ที่อ้างอิงจาก เพื่อใช้สำหรับการหาดำแหน่งใหม่ (relocalization) ในกรณีที่การติดตามล้มเหลว (เช่น เกิดการบังกล้อง) หรือเพื่อเริ่มต้นใหม่ในฉากที่มีการทำแผนที่ไว้แล้ว และเพื่อใช้ในการตรวจจับลูป (loop detection) ระบบจะเก็บ กราฟการมองเห็นร่วม (covisibility graph) ซึ่งเชื่อมโยง keyframe สองตัวที่สังเกตเห็นจุดร่วมกัน และโครงสร้าง ต้นไม้ครอบคลุมน้ำหนักรวมต่ำสุด (minimum spanning tree) ที่เชื่อม keyframe ทั้งหมดเข้าด้วยกัน โครงสร้างกราฟเหล่านี้ช่วยให้สามารถดึงชุดหน้าต่างของ keyframe ในพื้นที่ท้องถิ่นออกมาได้ ทำให้การติดตาม (tracking) และการทำแผนที่ท้องถิ่น (local mapping) ทำงานในเชิงพื้นที่เฉพาะจุดได้ ช่วยให้รองรับการทำงานใน

สภาพแวดล้อมขนาดใหญ่ และยังทำหน้าที่เป็นโครงสร้างสำหรับการทำ pose-graph optimization เมื่อมีการปิดลูป



รูปที่ 1.1 ส่วนประกอบหลักของระบบ ORB-SLAM2

จากการทบทวนแนวคิดและงานวิจัยที่เกี่ยวข้อง พบว่า Visual SLAM โดยเฉพาะอย่างยิ่งระบบที่ใช้กล้องแบบ Stereo หรือ RGB-D เป็นวิธีการที่ น่าเชื่อถือที่สุด ในการสร้างแผนที่และระบุตำแหน่งที่มีความแม่นยำ เนื่องจากสามารถแก้ไขปัญหาระยะ ขนาด (Scale) และ ความคลาดเคลื่อนสะสม (Drift) ของกล้องเดี่ยวได้ ดังนั้น เพื่อให้บรรลุวัตถุประสงค์ของการ พัฒนาระบบระบุตำแหน่งระหว่างอาคาร ที่ต้องการความแม่นยำสูงและมีความทนทานต่อสภาพแวดล้อมจริง งานวิจัยนี้จึงตัดสินใจเลือกใช้ ORB-SLAM2 เป็น กรอบแนวคิดหลักในการจัดการด้าน Visual SLAM เนื่องจาก ORB-SLAM2 ได้รับการพัฒนาให้รองรับการทำงานทั้งแบบ Monocular, Stereo และ RGB-D ซึ่ง สอดคล้องกับแนวคิดการใช้กล้องในการเก็บข้อมูล ORB-SLAM2 จึงถูกนำมาใช้เป็นฐานในการประมวลผลข้อมูลภาพ ก่อนที่จะผสมรวมเข้ากับข้อมูลจาก GNSS ผ่าน Extended Kalman Filter (EKF) เพื่อให้ได้ระบบระบุตำแหน่งที่มีความแม่นยำสูงในสภาพแวดล้อมนอกอาคารที่มีอาคารสูง ซึ่งเป็นส่วนที่จะกล่าวถึงในบทต่อไป

2.2 Extended Kalman Filter

EKF หรือ Extended Kalman Filter เป็นหนึ่งในเทคนิคที่ถูกใช้งานอย่างแพร่หลายในโลกของหุ่นยนต์จริง คุณสามารถพบการประยุกต์ใช้ EKF ได้ตั้งแต่ในรถยนต์ไร้คนขับจนถึงโดรน เนื่องจาก หุ่นยนต์ในโลกจริงมักต้องพึ่งพาเซนเซอร์หลากหลายชนิดเพื่อรับรู้สภาพแวดล้อมรอบตัว แต่เซนเซอร์เหล่านี้มักมีสัญญาณรบกวนและไม่สามารถให้ค่าที่แม่นยำ 100% ได้ ซึ่งเป็นเรื่องปกติในงานวิศวกรรม

EKF จึงเข้ามามีบทบาทสำคัญในการประมวลผลข้อมูล โดยจะนำข้อมูลการสังเกตจากเซนเซอร์ทั้งหมดมาผ่านกระบวนการกรอง เพื่อลดความสั่นไหวและความคลาดเคลื่อนจากสัญญาณรบกวน ผลลัพธ์ที่ได้คือค่าประมาณสถานะ (state) ของหุ่นยนต์ที่แม่นยำขึ้นในแต่ละช่วงเวลา t ขณะหุ่นยนต์เคลื่อนที่ไปในสภาพแวดล้อมจริง “สถานะ” ในที่นี้หมายถึงข้อมูลสำคัญอย่างเช่น “ตำแหน่งของหุ่นยนต์อยู่ที่ไหน” และ “ทิศทางหรือการหันของมันเป็นอย่างไ” ด้วยคุณสมบัติเหล่านี้ EKF จึงกลายเป็นเครื่องมือสำคัญที่ช่วยให้หุ่นยนต์ทำงานได้อย่างมั่นคงและเชื่อถือได้ แม้ในสภาพแวดล้อมที่เต็มไปด้วยความไม่แน่นอน

สถานะ (State) ในบริบทนี้หมายถึงสถานะทางกายภาพของระบบ ซึ่งสามารถอธิบายได้ด้วยตัวแปรเชิงพลวัต (dynamic variables) เช่น ตำแหน่ง ความเร็ว หรือทิศทางของระบบ อย่างไรก็ตาม ข้อมูลการวัดที่ได้จากเซนเซอร์มักมีสัญญาณรบกวน ซึ่งทำให้มีความไม่แน่นอน (uncertainty) ในค่าที่วัดได้

ระบบพลวัต (dynamical system) เป็นระบบที่สถานะมีการเปลี่ยนแปลงตามเวลา โดยทั่วไปจะใช้สมการเชิงอนุพันธ์ (differential equations) ในการจำลองพฤติกรรมของระบบ เหล่านี้ นอกจากนี้ ยังมีสัญญาณรบกวนในพลวัตของระบบเอง (process noise) ซึ่งหมายความว่าเราไม่สามารถจำลองระบบได้อย่างแม่นยำแบบกำหนดแน่นอน (deterministic) แต่ยังสามารถรับข้อมูลวัดที่มีสัญญาณรบกวนได้

คำว่า การกรอง (filtering) หมายถึงกระบวนการกรองเอาสัญญาณรบกวนออกจากข้อมูลการวัด เพื่อให้ได้ค่าประมาณสถานะที่เหมาะสมที่สุดตามข้อมูลที่สังเกตได้

ก่อนที่จะเข้าใจเกี่ยวกับตัว Extended Kalman Filter จะต้องทำความเข้าใจเกี่ยวกับหลักการของตัว Kalman Filter ซึ่งตัว Kalman Filter เป็นอัลกอริทึมที่ใช้ในการประมาณค่าและทำนายสถานะของระบบ โดยคำนึงถึงความไม่แน่นอนและสัญญาณรบกวนถูกสร้างมาให้ใช้กับระบบที่เป็นเชิงเส้น โดย Kalman filter มี 2 ขั้นตอนที่สำคัญ

- 1) Prediction Step เป็นขั้นตอนในการประมาณสถานะถัดไป และ ความแปรปรวน ณ เวลา t ของระบบตามสถานะก่อนหน้า ณ เวลา $t-1$

$$\hat{x}_t = Fx_{t-1} + Bu_t \quad (2.1)$$

$$\hat{P}_t = FP_{t-1}F^T + Q_t \quad (2.2)$$

- 2) Correction Step หรือ Updated step เป็นขั้นตอนแก้ไขการประมาณค่าจากข้อมูลที่ทำการวัด z ณ เวลา t

$$v_t = z_t - H\hat{x}_t \quad (2.3)$$

$$S_t = H\hat{P}_tH^T + R \quad (2.4)$$

$$K_t = \hat{P}_tH^TS_t^{-1} \quad (2.5)$$

$$x_t = \hat{x}_t + K_tv_t \quad (2.6)$$

$$P_t = \hat{P}_t + K_tS_tK_t^T \quad (2.7)$$

เมื่อ	x_t	คือ state of the system
	Q_t	คือ process noise
	R_t	คือ measurement noise
	F	คือ State Transition Matrix
	H	คือ masurement model matrix
	$\hat{P}_t$	คือ Predicted Covariance
	P_t	คือ Estimated Covariance
	u_t	คือ Control input vector
	B	คือ Control input model
	H	คือ Observation model
	v_t	คือ Measurement residual
	S_t	คือ Measurement Prediction Covariance
	K_t	คือ Kalman Gain

สถานะที่คาดการณ์ $\hat{x}_t$ จะถูกประมาณในขั้นแรกโดยอ้างอิงจากสถานะที่คำนวณก่อนหน้านี้ x_{t-1} และต่อมาจะถูก แก้ไข หรือ อัปเดตในขั้นที่สองเพื่อให้ได้การประมาณสถานะสุดท้าย x_t การคำนวณที่คล้ายกันเกิดขึ้นกับความแปรปรวน P

Extended Kalman Filter (EKF) เป็นการประยุกต์ใช้ Kalman Filter ในระบบที่ไม่เป็นเชิงเส้น โดยการทำให้ Linearization ฟังก์ชันที่ไม่เป็นเชิงเส้นในแต่ละขั้นตอนของการประมาณค่า EKF มีสองขั้นตอนหลัก ๆ คือ การทำนายสถานะ (Prediction) และการปรับปรุงสถานะ (Update) โดยใช้ Jacobian Matrix ในการ Linearize ฟังก์ชันที่ไม่เป็นเชิงเส้น

1) Prediction step

$$\hat{x}_t = f(x_{t-1}, u_t) \quad (2.8)$$

$$\hat{P}_t = F(x_{t-1}, u_t)P_{t-1}F^T(x_{t-1}, u_t) + Q_t \quad (2.9)$$

2) Correction step

$$v_t = z_t - h(x_t) \quad (2.10)$$

$$S_t = H(x_t)\hat{P}_tH^T(x_t) + R_t \quad (2.11)$$

$$K_t = \hat{P}_tH^T(x_t)S_t^{-1} \quad (2.12)$$

$$x_t = \hat{x}_t + K_tv_t \quad (2.13)$$

$$P_t = (I_4 - K_tH(x_t))\hat{P}_t \quad (2.14)$$

เมื่อ f คือ ฟังก์ชันที่เป็นเชิงเส้น
 h คือ ฟังก์ชันโมเดลการวัดแบบไม่เชิงเส้น Matrix

$$F(x_{t-1}, u_t) = \frac{\partial f(x_{t-1}, u_t)}{\partial x} \quad (2.15)$$

$$H(x_t) = \frac{\partial h(x_t)}{\partial x} \quad (2.16)$$

F และ H คือ Jacobians ของ f และ h ซึ่งหาได้จากสมการที่ 2.15 และ 2.16

จากการทบทวนหลักการของ Kalman Filter (KF) และ Extended Kalman Filter (EKF) ข้างต้น แสดงให้เห็นว่า EKF เป็นเครื่องมือที่ขาดไม่ได้ในการจัดการกับ ความไม่แน่นอน (Uncertainty) และ สัญญาณรบกวน (Noise) ซึ่งเป็นสิ่งที่พบได้ปกติในข้อมูลที่ได้จากเซนเซอร์จริง ดังนั้น เพื่อให้ได้ค่า ประมาณสถานะ (State Estimation) ของระบบระบุตำแหน่งที่ แม่นยำและ น่าเชื่อถือที่สุด งานวิจัยนี้จึงเลือกใช้ Extended Kalman Filter (EKF) เป็น เทคนิคหลักในการรวม ข้อมูลเซนเซอร์ (Sensor Fusion) เนื่องจาก ระบบพลวัตของการเคลื่อนที่ของหุ่นยนต์และโมเดลการ วัดจากเซนเซอร์ต่าง ๆ มักไม่เป็นเชิงเส้น (Non-linear) EKF สามารถจัดการกับความไม่เป็นเชิงเส้น เหล่านี้ได้อย่างมีประสิทธิภาพ และทำหน้าที่กรอง (Filtering) สัญญาณรบกวนจากทั้งพลวัตของระบบ (Process Noise) และข้อมูลการวัด (Measurement Noise) จาก GNSS และ ORB-SLAM2 จึงเป็น แกนหลักทางคณิตศาสตร์ในการพัฒนา ระบบระบุตำแหน่งนอกระบบนี้เพื่อให้ได้ค่าตำแหน่งที่มีความ คลาดเคลื่อนต่ำและมีความมั่นคงในการใช้งานจริง

2.3 Odometry

การระบุตำแหน่งของหุ่นยนต์ ทำได้โดยวิธีการที่เรียกว่า Odometry คือ การระบุตำแหน่งที่หุ่นยนต์ เคลื่อนที่ได้ในระยะเวลาทางเท่าไร ตำแหน่งและทิศทางต่างๆของหุ่นยนต์เมื่อเทียบกับจุดเริ่มต้น ลักษณะการทำงานของวิธีการ Odometry จะขึ้นอยู่กับเซนเซอร์ที่นำมาใช้งานกับหุ่นยนต์ โดยส่วน ใหญ่จะนำมาจาก wheel encoder, IMU และ camera

2.3.1 odometry จาก wheel encoder

ลักษณะการทำงานของวิธีการนี้จะนับจำนวนรอบการหมุนที่ล้อของหุ่นยนต์โดยใช้ encoder จากนั้นนำข้อมูลที่ได้มาแปลงกลับเป็นค่าระยะทาง ตำแหน่ง และทิศทางการเคลื่อนที่

ซึ่ง odometry จาก encoder เราคำนวณได้จากวิธีการของ dead reckoning ซึ่งเป็นกระบวนการคำนวณตำแหน่งของหุ่นยนต์ โดยอิงจากตำแหน่งเริ่มต้นที่รู้และการคำนวณระยะทางที่เคลื่อนที่ไปในทิศทางต่างๆในช่วงเวลาที่กำหนด กระบวนการนี้ใช้ข้อมูลเกี่ยวกับความเร็ว ทิศทาง และเวลาในการเดินทางโดยไม่ต้องพึ่งพาการนำทางด้วยเครื่องมือที่เชื่อมต่อกับแหล่งข้อมูลภายนอก สมการที่ใช้อธิบายกระบวนการนี้แสดงให้ดังสมการ 2.17

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \theta \end{bmatrix} \quad (2.17)$$

โดยที่ x_t คือ ตำแหน่งปัจจุบันตามพิกัดแกน x ที่เวลา t (เมตร)
 y_t คือ ตำแหน่งปัจจุบันตามพิกัดแกน y ที่เวลา t (เมตร)
 θ_t คือ ทิศทางปัจจุบันของการหมุนที่เวลา t (เรเดียน)
 x_{t-1} คือ ตำแหน่งก่อนหน้าตามพิกัดแกน x ที่เวลา t (เมตร)
 y_{t-1} คือ ตำแหน่งก่อนหน้าตามพิกัดแกน y ที่เวลา t (เมตร)
 θ_{t-1} คือ ทิศทางก่อนหน้าของการหมุนที่เวลา t (เรเดียน)
 Δx คือ ระยะทางที่เปลี่ยนแปลงตามพิกัดแกน x (เมตร)
 Δy คือ ระยะทางที่เปลี่ยนแปลงตามพิกัดแกน y (เมตร)
 $\Delta \theta$ คือ มุมที่เปลี่ยนแปลง (เรเดียน)

หลักการของ dead reckoning คือ เริ่มจากเราต้องรู้ตำแหน่งที่รู้แน่นอนก่อน และทำการคำนวณหาระยะทางที่เคลื่อนที่ตามทิศทางที่กำหนดในช่วงเวลาหนึ่ง และใช้ความเร็วเฉลี่ยและทิศทางในการคำนวณตำแหน่งใหม่ ซึ่งตำแหน่งของหุ่นยนต์เราจะหาได้จาก จลนศาสตร์การเคลื่อนที่ของหุ่นยนต์แบบ differential drive จะเป็นการคำนวณทางคณิตศาสตร์ของการเคลื่อนที่ของหุ่นยนต์ โดยอิงจากการควบคุมล้อของหุ่นยนต์ หุ่นยนต์ที่ใช้การขับเคลื่อนแบบ differential drive มักจะมีสองล้อที่ขับเคลื่อนอย่างอิสระอยู่ที่แต่ละด้าน และการเคลื่อนที่ของหุ่นยนต์จะถูกควบคุมโดยการปรับความเร็วของล้อทั้งสอง จลนศาสตร์การเคลื่อนที่ของหุ่นยนต์แบบ differential drive สามารถอธิบายได้ด้วยสมการที่ 2.18 และ 2.19

ความเร็วเชิงเส้น (Linear Velocity)

$$v = \frac{v_l + v_r}{2} \quad (2.18)$$

ความเร็วเชิงมุม (Angular Velocity)

$$\omega = \frac{v_r - v_l}{2d_w} \quad (2.19)$$

โดยที่ v_l คือ ความเร็วล้อซ้าย (เมตร/วินาที)

v_r คือ ความเร็วล้อขวา (เมตร/วินาที)

d_w คือ ระยะห่างระหว่างล้อ (เมตร)

การหาตำแหน่ง (X, Y) และท่าทาง (θ) ของหุ่นยนต์สามารถอธิบายได้ด้วยการหาปริพันธ์ของความเร็ว ซึ่งสมการในการหา ตำแหน่งและท่าทางของหุ่นยนต์ สามารถอธิบายได้ดังสมการที่ 2.20 จนถึง สมการที่ 2.22

$$\frac{dx}{dt} = v \cos(\theta) \quad (2.20)$$

$$\frac{dy}{dt} = v \sin(\theta) \quad (2.21)$$

$$\frac{d\theta}{dt} = \omega \quad (2.22)$$

การระบุตำแหน่งของหุ่นยนต์ด้วยวิธี odometry นั้น จะมีความถูกต้อง แม่นยำค่อนข้างต่ำ เกิดจากปัญหาทางกายภาพของตัวเซ็นเซอร์ ปัญหาการลื่นไถล (slip) ของหุ่นยนต์ รวมถึงสัญญาณรบกวนต่างๆ ทำให้ข้อมูลที่ได้มาไม่มีความสมบูรณ์ อีกทั้งในแต่ละช่วงเวลาที่ย่เปลี่ยนไป ระบบจะสะสมความผิดพลาดเพิ่มขึ้นอย่างต่อเนื่องซึ่งทำให้หุ่นยนต์มีความผิดพลาดได้ ดังนั้นจำเป็นต้องมีการประยุกต์ใช้เทคนิคหรือวิธีการอื่นๆ ควบคู่ไปด้วยเพื่อให้หุ่นยนต์มีความถูกต้องและแม่นยำมากขึ้น

2.4 GNSS

ปัจจุบันมีระบบ SATNAV จำนวนมากที่ให้บริการทั่วโลก บางระบบครอบคลุมทั่วโลก ขณะที่บางระบบให้บริการเฉพาะในบางภูมิภาค คำว่า Global Navigation Satellite System (GNSS) ถูกนิยามว่าเป็นการรวมกันของระบบ SATNAV ทั้งหมดและระบบเสริม (augmentations) ของมัน อย่างไรก็ตาม ปัจจุบันคำว่า GNSS ก็มักถูกใช้เพื่อหมายถึงระบบ SATNAV แบบใดระบบหนึ่ง โดยเฉพาะ GNSS มอบข้อมูลตำแหน่งและความเร็วแบบสามมิติที่มีความแม่นยำ ต่อเนื่อง และครอบคลุมทั่วโลกให้แก่ผู้ใช้ที่มีอุปกรณ์รับสัญญาณที่เหมาะสม อีกทั้งยังเผยแพร่ข้อมูลเวลาอ้างอิงตามมาตรฐานเวลา Coordinated Universal Time (UTC) ด้วย ระบบ SATNAV ใช้แนวคิด one-way Time of Arrival (TOA) ranging โดยสัญญาณที่ส่งจากดาวเทียมจะอ้างอิงกับมาตรฐานความถี่อะตอม (atomic frequency standards) ที่มีความแม่นยำสูงบนดาวเทียม ซึ่งถูกซิงโครไนซ์กับฐานเวลาภายในของระบบ ทุกระบบ SATNAV สัญญาณที่ส่งจากดาวเทียมแต่ละดวงจะถูกซิงโครไนซ์ได้สำหรับการวัดระยะกับมาตรฐานเวลาร่วมเดียวกัน ข้อมูลการนำทางจะทำให้เครื่องรับสามารถคำนวณหาตำแหน่งของดาวเทียมในขณะที่สัญญาณถูกส่ง ส่วนโค้ดการวัดระยะทำให้เครื่องรับของผู้ใช้สามารถคำนวณเวลาที่สัญญาณเดินทาง (transit/propagation time) และหาระยะทางระหว่างดาวเทียมกับผู้ใช้ เทคนิคนี้ต้องการให้เครื่องรับมีนาฬิกาด้วย[4] โดยการหาตำแหน่งสามมิติของเครื่องรับด้วยวิธี TOA จำเป็นต้องวัดค่าระยะทางไปยังดาวเทียมอย่างน้อย 4 ดวง หากนาฬิกาในเครื่องรับถูกซิงโครไนซ์กับนาฬิกาของดาวเทียม จะต้องการเพียง 3 การวัดเท่านั้น อย่างไรก็ตาม โดยทั่วไปเครื่องรับการนำทางจะใช้นาฬิกาคริสตัลเพื่อให้ต้นทุน ความซับซ้อน และขนาดของอุปกรณ์ที่ต่ำลง ดังนั้นจึงต้องใช้การวัด 4 ครั้งเพื่อหาค่าละติจูด ลองจิจูด ความสูง และความคลาดเคลื่อนของนาฬิกาที่รับสัญญาณเมื่อเทียบกับเวลาภายในของระบบ แต่ถ้าผู้ใช้ทราบเวลาของระบบหรือความสูงที่แม่นยำอยู่แล้ว ก็สามารถใช้อาวเทียมน้อยกว่า 4 ดวงได้

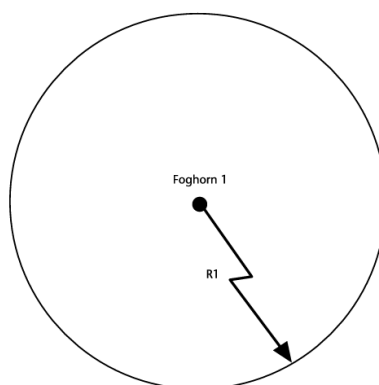
2.4.1 แนวคิดของการวัดระยะทางโดยใช้การวัดเวลาที่สัญญาณเดินทางมาถึง

GNSS ใช้แนวคิด time-of-arrival (TOA) ranging เพื่อหาตำแหน่งของผู้ใช้ แนวคิดนี้คือการวัดเวลาที่สัญญาณซึ่งถูกส่งออกจากแหล่งกำเนิด ที่มีตำแหน่งแน่นอน ใช้เวลาเดินทางมาถึงเครื่องรับของผู้ใช้นานเท่าใด ช่วงเวลานี้เรียกว่า signal propagation time จากนั้นนำเวลานี้ไปคูณกับความเร็วของการแพร่กระจายสัญญาณ (เช่น ความเร็วเสียง หรือความเร็วแสง) เพื่อหาระยะทางระหว่างแหล่งกำเนิดกับเครื่องรับ

1) การหาตำแหน่งแบบสองมิติ (Two-Dimensional Position Determination)

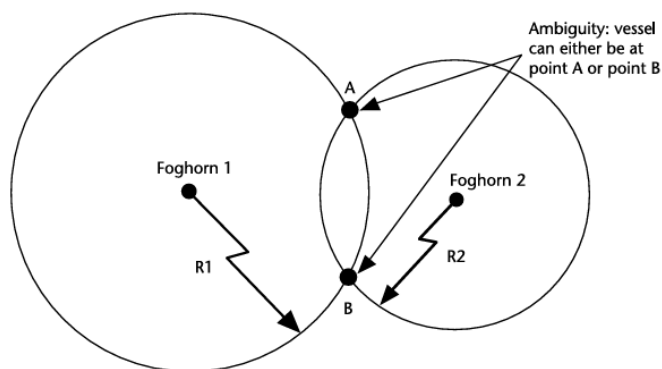
ลองพิจารณากรณีที่นักเดินเรือในทะเลกำลังหาตำแหน่งของเรือโดยอาศัยเสียงสัญญาณจาก foghorn สมมติว่าเรือติดตังนาฬิกาที่มีความแม่นยำและนักเดินเรือมีความรู้โดยประมาณเกี่ยวกับตำแหน่งของเรือ อีกทั้งสมมติว่าเสียงสัญญาณ foghorn ถูกเป่าอย่างแม่นยำตรง

กับเครื่องหมายนาฬิกา (minute mark) และนาฬิกาบนเรือถูกซิงโครไนซ์กับนาฬิกาของ foghorn นักเดินเรือจะจดบันทึกเวลาที่ผ่านไปตั้งแต่เครื่องหมายนาฬิกาที่จนกระทั่งได้ยินเสียงสัญญาณ foghorn เวลาแพร่กระจายของเสียง foghorn (propagation time) คือเวลาที่ใช้ตั้งแต่เสียงออกจาก foghorn จนถึงหูของนักเดินเรือ เวลานี้เมื่อนำไปคูณกับความเร็วเสียง (ประมาณ 335 เมตร/วินาที) จะได้ระยะทางจาก foghorn ถึงตำแหน่งของนักเดินเรือเช่น หากสัญญาณ foghorn ใช้เวลา 5 วินาทีในการเดินทางมาถึงหูของนักเดินเรือ ระยะทางจาก foghorn จะเท่ากับ 1,675 เมตร ให้ระยะนี้แทนด้วย R1 ดังนั้น จากการวัดเพียงครั้งเดียว นักเดินเรือจะรู้ว่าเรืออยู่ที่ใดที่หนึ่งบนวงกลมที่มีรัศมี R1 โดยมีจุดศูนย์กลางที่ตำแหน่งของ foghorn ซึ่งในรูปที่ 2.2 เรียกว่า Foghorn 1

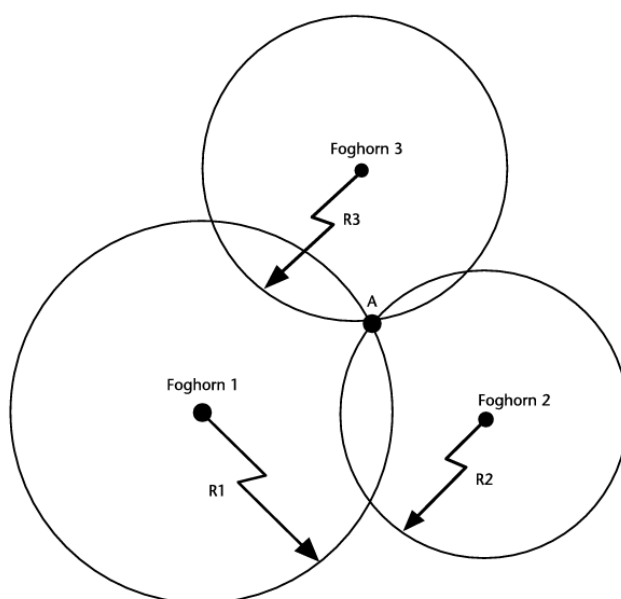


รูปที่ 1.2 การหาค่าระยะทางจากแหล่งกำเนิดเพียงแหล่งเดียว

ในทางสมมติ หากนักเดินเรือสามารถวัดระยะจาก foghorn ตัวที่สองได้พร้อมกันในลักษณะเดียวกัน ตำแหน่งของเรือก็จะต้องอยู่ที่ระยะ R1 จาก Foghorn 1 และระยะ R2 จาก Foghorn 2 ดังแสดงในรูปที่ 2.3 โดยสมมติว่าสัญญาณ foghorn ทั้งสองถูกซิงโครไนซ์กับฐานเวลาเดียวกัน และนักเดินเรือทราบเวลาที่ foghorn ส่งสัญญาณออกมาดังนั้น ตำแหน่งของเรือเมื่อเทียบกับ foghorn ทั้งสองจะต้องอยู่ที่จุดตัดของวงกลมทั้งสองวงที่แทนระยะทาง (range circles) เนื่องจากสมมติว่านักเดินเรือมีความรู้โดยประมาณเกี่ยวกับตำแหน่งเรืออยู่แล้ว เขาจึงสามารถตัดตำแหน่งที่ไม่น่าเป็นไปได้ออกไปได้อีกวิธีในการแก้ความไม่ชัดเจนนี้ คือทำการวัดระยะจาก foghorn ตัวที่สาม เพิ่มเติม ดังแสดงในรูปที่ 2.4



รูปที่ 1.3 ความกำกวมที่เกิดจากการวัดระยะทางจากสองแหล่งกำเนิด

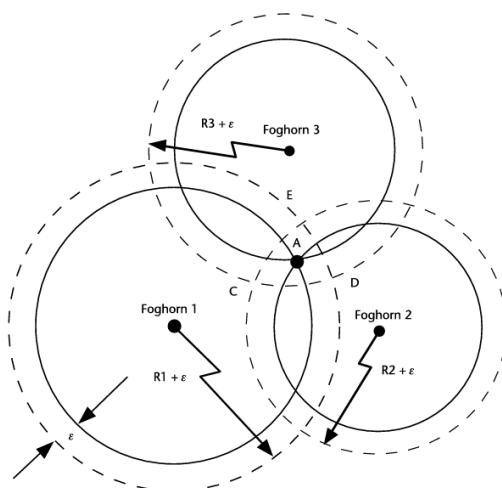


รูปที่ 1.4 การแก้ความกำกวมของตำแหน่งด้วยการวัดเพิ่มเติม

2) ความคลาดเคลื่อนของนาฬิกาการร่วมและการชดเชย (Common Clock Offset and Compensation)

การอธิบายข้างต้นสมมติว่า นาฬิกาบนเรือถูกซิงโครไนซ์อย่างแม่นยำกับฐานเวลาของ foghorn แต่ในความเป็นจริงอาจไม่เป็นเช่นนั้น สมมติว่านาฬิกาบนเรือ เดินนำไป 1 วินาที เมื่อเทียบกับฐานเวลาของ foghorn กล่าวคือ นาฬิกาบนเรือเข้าใจว่าเครื่องหมายที่เกิดขึ้นก่อนจริง 1 วินาที ผลลัพธ์คือช่วงเวลาการแพร่กระจาย (propagation intervals) ที่นักเดินเรือวัดได้จะมากกว่าความเป็นจริง 1 วินาที เนื่องจากความคลาดเคลื่อนนี้ ความคลาดเคลื่อนของเวลา (timing offset) จะเหมือนกันสำหรับการวัด (เป็น common offset) เพราะนาฬิกาที่ผิดพลาดถูกใช้

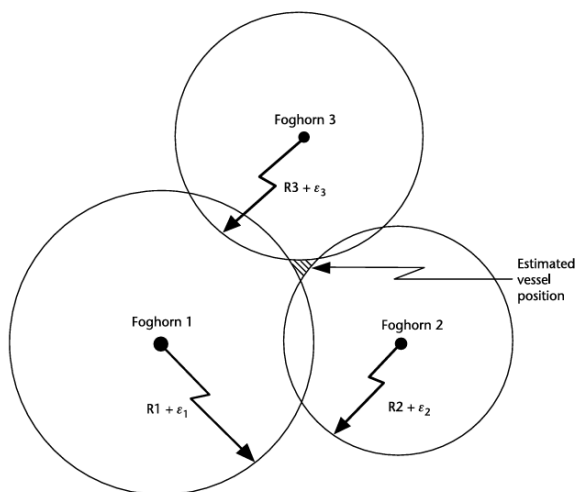
ร่วมกันในการวัดทั้งหมด ความคลาดเคลื่อนด้านเวลานี้จะเท่ากับ ข้อผิดพลาดของระยะทาง (range error) ประมาณ 335 เมตร (เมื่อใช้ความเร็วเสียง 335 m/s) และในรูปที่ 2.5 ถูกแทนด้วยสัญลักษณ์ ϵ การที่ตำแหน่งจุดตัด C, D และ E เบี่ยงเบนออกจากตำแหน่งที่แท้จริงของเรือ (จุด A) เป็นผลจากความคลาดเคลื่อนของนาฬิกาบนเรือ หากสามารถลบหรือชดเชยความคลาดเคลื่อนนี้ได้ วงกลมระยะทางทั้งหมดก็จะตัดกันที่ จุด A ซึ่งคือตำแหน่งที่แท้จริงของเรือ



รูปที่ 1.5 ผลกระทบของความคลาดเคลื่อนนาฬิกาของเครื่องรับต่อการวัด

3) ผลกระทบของความคลาดเคลื่อนในการวัดแบบอิสระต่อความแม่นยำของตำแหน่ง

หากสมมติฐานข้างต้นเกิดขึ้นจริง การวัด TOA (time-of-arrival) ก็จะไม่สมบูรณ์แบบ เนื่องจากมีข้อผิดพลาดจากหลายปัจจัย เช่น ผลกระทบของบรรยากาศ ความคลาดเคลื่อนของนาฬิกา foghorn เมื่อเทียบกับฐานเวลา foghorn เสียงรบกวนอื่น ๆ ที่แทรกเข้ามา ซึ่งแตกต่างจากกรณี ความคลาดเคลื่อนของนาฬิกาบนเรือ ที่กล่าวไปก่อนหน้านี้ เพราะข้อผิดพลาดเหล่านี้มักจะเป็นแบบ อิสระต่อกัน (independent errors) และไม่ใช้ข้อผิดพลาดร่วม (not common) สำหรับทุกการวัด แต่จะส่งผลกระทบต่อการวัดแต่ละครั้งในลักษณะที่แตกต่างกันไป ส่งผลให้การคำนวณระยะทางไม่แม่นยำ ในรูปที่ 2.6 แสดงผลกระทบของ ข้อผิดพลาดอิสระ เหล่านี้ต่อการวัด



รูปที่ 1.6 ผลกระทบของความคลาดเคลื่อนในการวัดแบบอิสระต่อความแน่นอนของตำแหน่ง

ข้อผิดพลาดอิสระเหล่านี้ (เช่น ϵ_1 , ϵ_2 , และ ϵ_3) ส่งผลต่อการหาตำแหน่ง แม้ว่าจะสมมติว่านาฬิกาบนเรือและฐานเวลาของ foghorn ถูกซิงโครไนซ์ก็ตาม แทนที่จะเกิดจุดตัดของวงกลมระยะทั้งสามวงที่เดียวกัน ตำแหน่งของเรือจะอยู่ภายใน พื้นที่สามเหลี่ยมแห่งความคลาดเคลื่อน (triangular error space) แทน

2.4.2 หลักการหาตำแหน่งโดยใช้รหัสวัดระยะ (Ranging Codes) ที่สร้างจาก

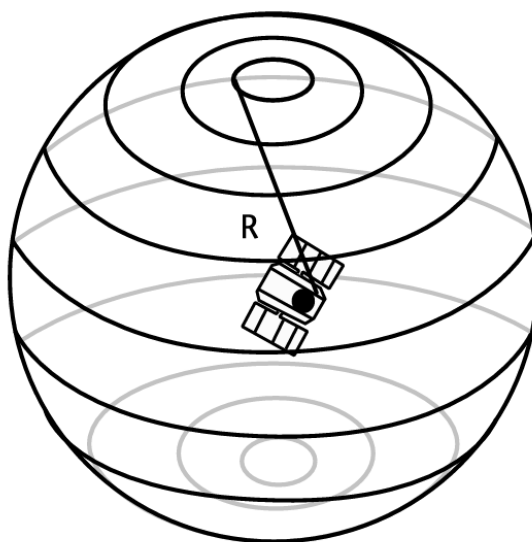
ดาวเทียม

GNSS ใช้การวัด TOA (time-of-arrival) ranging เพื่อหาตำแหน่งของผู้ใช้ โดยการวัด TOA จากดาวเทียมหลายดวง จะสามารถหาตำแหน่งแบบสามมิติได้เทคนิคนี้มีความคล้ายคลึงกับตัวอย่าง foghorn ข้างต้น แต่ความแตกต่างคือ โค้ดวัดระยะของดาวเทียมเดินทางด้วยความเร็วแสงซึ่งประมาณ 3×10^8 เมตร/วินาที และสมมติว่าตำแหน่งดาวเทียมใน ephemerides มีความแม่นยำ (หมายความว่าตำแหน่งดาวเทียมถูกต้องแน่นอน)

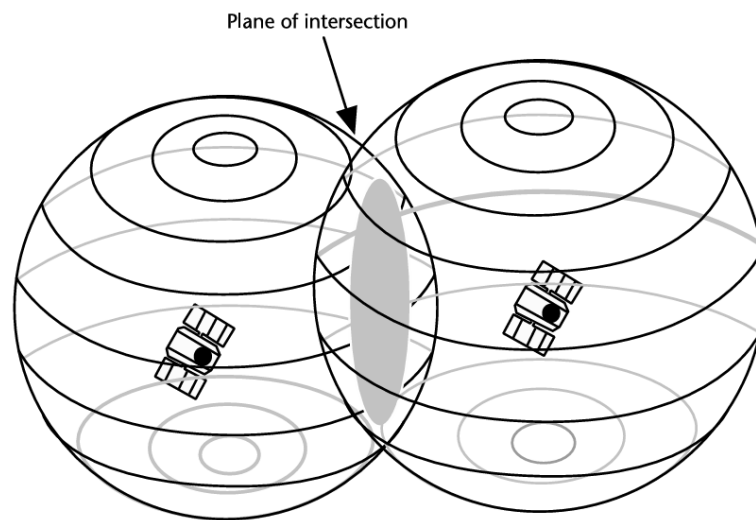
1) การหาตำแหน่งสามมิติด้วยการหาจุดตัดของทรงกลมหลายชุด

สมมติว่ามีดาวเทียมเพียงดวงเดียวที่ส่งสัญญาณวัดระยะ (ranging signal) นาฬิกาบนดาวเทียมควบคุมเวลาการส่งสัญญาณนี้ นาฬิกานี้และนาฬิกาอื่น ๆ บนดาวเทียมแต่ละดวงในกลุ่มดาว SATNAV constellation ถูกซิงโครไนซ์กับ ระบบเวลาในตัว (system time) ซึ่งในที่นี้เรียกว่า system time (เช่น GPS system time) เครื่องรับของผู้ใช้ก็มีนาฬิกาของตัวเอง ซึ่งในตอนนี้สมมติว่าซิงโครไนซ์กับ system time ข้อมูลเวลา (timing information) จะฝังอยู่ในสัญญาณวัดระยะของดาวเทียม ทำให้เครื่องรับสามารถคำนวณได้ว่าสัญญาณออกจากดาวเทียมเมื่อใดตามเวลาของนาฬิกาบนดาวเทียมโดยสังเกตเวลาที่รับสัญญาณ เครื่องรับสามารถคำนวณ เวลาแพร่กระจาย

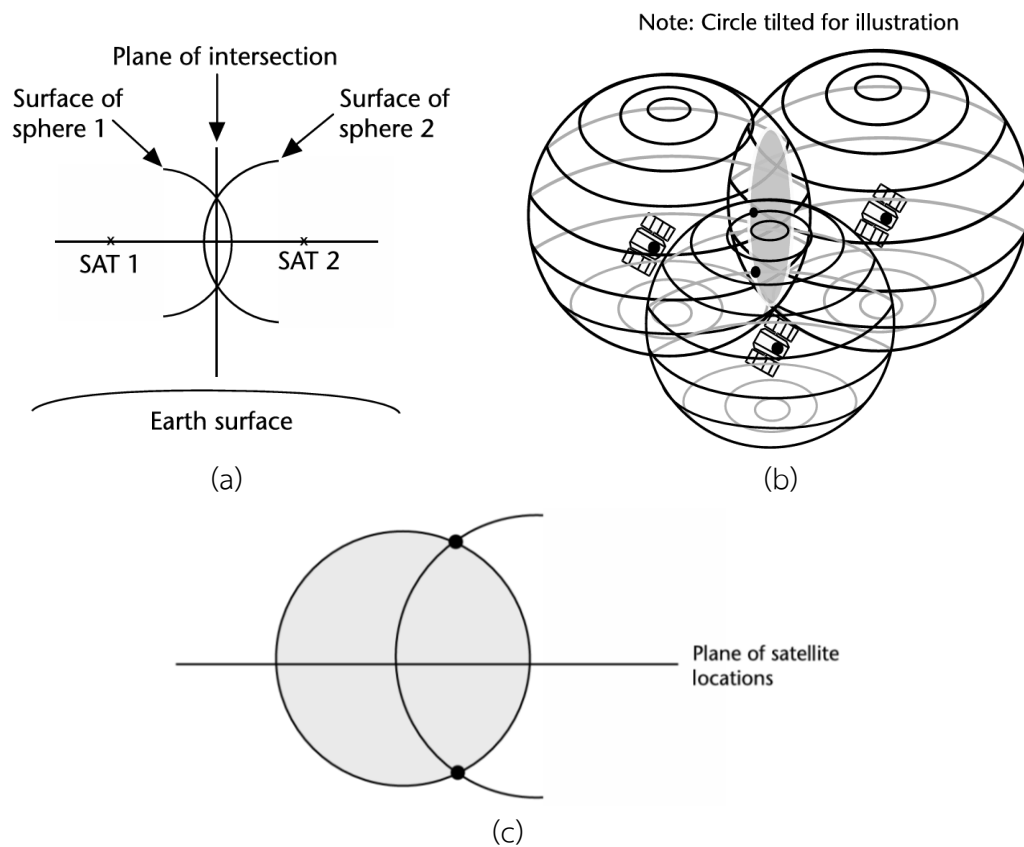
ของสัญญาณ (satellite-to-user propagation time) และเมื่อนำมาคูณกับ ความเร็วแสง จะได้ระยะทางจากดาวเทียมถึงผู้ใช้ (R) ผลลัพธ์ของการวัดนี้ทำให้ผู้ใช้อยู่บนผิวทรงกลม ที่มีดาวเทียมเป็นจุดศูนย์กลาง รูป 2.7 ถ้ามีการวัดพร้อมกันจาก ดาวเทียมดวงที่สอง ผู้ใช้จะอยู่บนผิวทรงกลมอีกอันหนึ่งที่มีดาวเทียมดวงที่สองเป็นศูนย์กลาง ดังนั้นผู้ใช้จะอยู่ บนผิวทรงกลมทั้งสอง ซึ่งจุดที่เป็นไปได้คือเส้นรอบวงของวงกลมที่เป็นระนาบตัดกันของทรงกลมสองลูกดังรูป 2.8 จุดเดียวที่ทรงกลมทั้งสองสัมผัสกัน (เกิดขึ้นเฉพาะเมื่อผู้ใช้และดาวเทียมอยู่ในแนวเดียวกัน ซึ่งไม่ใช่กรณีทั่วไป) ระนาบที่ตัดกันของทรงกลมสองลูกตั้งฉากกับเส้นเชื่อมระหว่างดาวเทียมสองดวง รูป 2.9 (a) ถ้าเพิ่มการวัดจากดาวเทียมดวงที่สาม ผู้ใช้จะอยู่ที่ จุดตัดระหว่างผิวทรงกลมที่สามกับเส้นรอบวงของระนาบตัดกัน ทรงกลมดวงที่สามจะตัดเส้นรอบวงในสองจุด แต่ มีเพียงหนึ่งจุดที่ถูกต้อง สำหรับตำแหน่งผู้ใช้ รูป 2.9 (b) จากมุมมอง 3 มิติ จุดตัดสองจุดเป็น ภาพสะท้อนกัน (mirror images) กับระนาบของดาวเทียม รูป 2.9 (c) สำหรับผู้ใช้บนผิวโลก จุดที่อยู่ด้านล่างจะเป็นตำแหน่งจริง แต่ผู้ใช้ที่อยู่เหนือผิวโลก (เช่น บนเครื่องบินหรือในอวกาศ) อาจใช้ดาวเทียมที่มีมุมเงยติดลบ ทำให้การหาตำแหน่งที่ไม่กำกวมซับซ้อนมากขึ้น ผู้ใช้ในอวกาศอาจอยู่เหนือหรือต่ำกว่าระนาบดาวเทียม และอาจต้องมี ข้อมูลเสริม เพื่อเลือกตำแหน่งที่ถูกต้อง



รูปที่ 1.7 ผู้ใช้อยู่บนผิวทรงกลม



รูปที่ 1.8 ผู้ใช้อยู่บนเส้นรอบวงของวงกลมที่ระบายสี



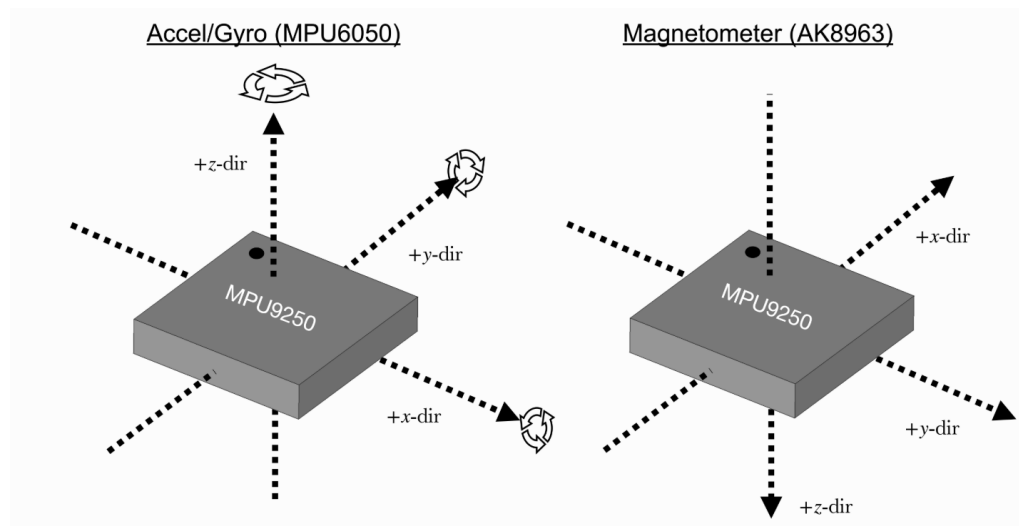
รูปที่ 1.9 (a) ระนาบของจุดตัด (b) ผู้ใช้หนึ่งที่หนึ่งในสองจุดบนวงกลมที่ระบายสี (c) ผู้ใช้หนึ่งที่หนึ่งในสองจุดบนเส้นรอบวงของวงกลม

ระบบกำหนดตำแหน่งด้วยดาวเทียม (GNSS) ถือเป็นเทคโนโลยีหลักและเป็นรากฐานที่ขาดไม่ได้สำหรับการระบุตำแหน่งภายนอกอาคาร (Outdoor Localization) และเป็นส่วนสำคัญของงานวิจัยนี้ในการพัฒนาระบบระบุตำแหน่งระหว่างอาคาร เป็นระบบเดียวที่สามารถให้ค่าพิกัดตำแหน่งสัมบูรณ์ (Global Coordinates) บนพื้นโลกได้โดยอัตโนมัติ ทำให้ระบบสามารถทราบตำแหน่งเริ่มต้น (Initial Position) และตำแหน่งที่แท้จริงในแผนที่โลกได้อย่างรวดเร็ว ซึ่งระบบ Visual SLAM ไม่สามารถให้ได้ด้วยตัวเอง เมื่อใช้ร่วมกับระบบ Visual SLAM (ORB-SLAM2) ข้อมูลตำแหน่งสัมบูรณ์จาก GNSS จะทำหน้าที่เป็นข้อมูลอ้างอิงภายนอก (External Reference) เพื่อช่วยจำกัดความคลาดเคลื่อนสะสม (Drift) ที่เกิดขึ้นจากกระบวนการทำแผนที่และการติดตามตำแหน่งด้วยภาพในระยะยาว อย่างไรก็ตาม แม้ว่า GNSS จะมีความสำคัญอย่างยิ่งในการระบุตำแหน่งภายนอกอาคาร แต่ข้อมูลที่ได้จาก GNSS มักมีความคลาดเคลื่อนสูงเมื่อใช้งานในพื้นที่ระหว่างอาคารสูง (Urban Canyon) เนื่องจากปัญหาการถูกบังสัญญาณ (Obstruction) และสัญญาณหลายเส้นทาง (Multipath)

ดังนั้น เพื่อแก้ไขข้อจำกัดนี้ งานวิจัยนี้จึงต้องใช้เทคนิคการผสานข้อมูลเซนเซอร์ (Sensor Fusion) โดยการนำข้อมูลจาก GNSS มารวมกับข้อมูลที่มีความละเอียดสูงจากกล้อง (Visual SLAM) ผ่าน Extended Kalman Filter (EKF) เพื่อให้ได้ระบบระบุตำแหน่งที่มีความแม่นยำสูง และมีความเสถียร ในสภาพแวดล้อมที่มีสิ่งกีดขวางตามวัตถุประสงค์ของการวิจัย

2.5 Gyroscope and Accelerometer Calibration

ในหัวข้อนี้จะเป็นการปรับเทียบเซนเซอร์ MPU9250 [10] ซึ่งเป็นเซนเซอร์วัดความเฉื่อยแบบ 9 องศาอิสระ (9-DoF) ที่ประกอบด้วยเซ็นเซอร์ตรวจจับความเร่งและไจโรสโคป (MPU6050) และเซ็นเซอร์แม่เหล็กมิเตอร์ (AK8963) ข้อมูลที่อ่านได้จาก MPU6050 และ AK8963 ด้านล่างนี้เป็นองค์ประกอบของระบบพิกัดสำหรับเซ็นเซอร์ทั้งสองตัว



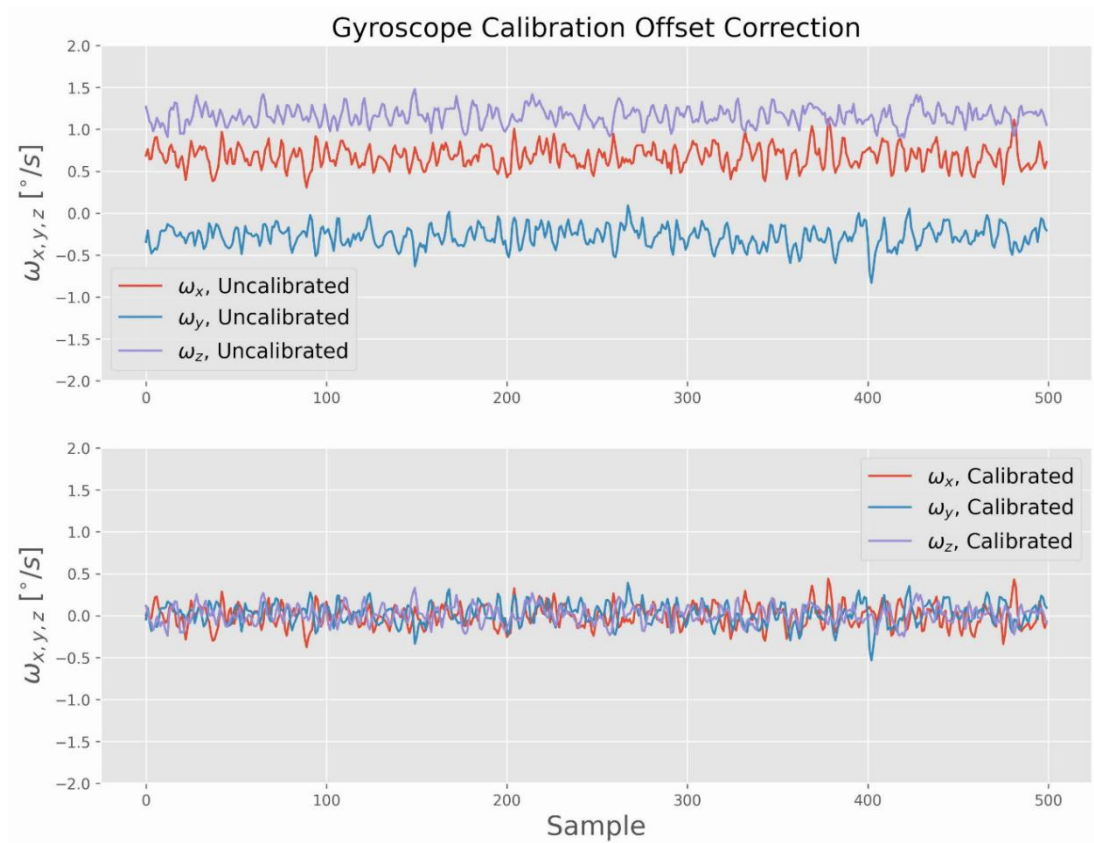
รูปที่ 1.10 ระบบพิกัดของเซ็นเซอร์วัดความเฉื่อย

ระบบพิกัดของแอคเซเลอโรมิเตอร์ใช้ระบบพิกัดคาร์ทีเซียนแบบดั้งเดิม โดยทั่วไปแล้ว ส่วนใจโรสโคปก็ใช้การหมุนทวนเข็มนาฬิกาเป็นทิศทางบวกของการหมุน เช่นเดียวกัน ในขณะที่แม็กเนโตมิเตอร์มีการสลับแกน x และ y และกลับทิศทางของขนาดในแกน z การกำหนดทิศทางตามแบบนี้จะมีผลต่อกระบวนการปรับเทียบ (Calibration) การอ้างอิงระบบพิกัดจะมีความสำคัญอย่างยิ่งในการทำ ความเข้าใจข้อมูลที่อ่านได้จาก MPU9250 และในการพัฒนาวิธีการสำหรับการปรับเทียบ IMU

อัตราการสุ่มตัวอย่าง (Sample Rate) ที่ใช้ก็คือ 8kHz ซึ่งเป็นอัตราการอัปเดตที่เร็วที่สุดสำหรับไจโรสโคป ส่วนอัตราการสุ่มตัวอย่างของแอคเซเลอโรมิเตอร์จะอยู่ที่ประมาณ 1kHz และแม็กเนโตมิเตอร์ (AK8963) มีค่าสูงสุดอยู่ที่ 100Hz อัตราการสุ่มตัวอย่างที่มีผลจริง (Effective Sample Rate) อยู่ที่ประมาณ 400Hz ซึ่งส่งผลให้ค่าที่อ่านได้จากเซ็นเซอร์ AK8963 ซ้ำกันหลายครั้ง

การปรับเทียบ IMU ที่ง่ายที่สุดคือการคำนวณค่าออฟเซต (offset) สำหรับแต่ละแกนของไจโรสโคป โดยไจโรสโคปเป็นเซ็นเซอร์ที่สามารถปรับเทียบได้ง่ายที่สุด เนื่องจากค่าที่คาดว่าจะอ่านได้ภายใต้สภาวะนิ่งนั้นค่อนข้างชัดเจน โดยปกติแล้ว ทั้งสามแกนของไจโรสโคปจะแสดงค่าเป็น 0 องศาต่อวินาที เมื่อ IMU ไม่ได้เคลื่อนไหว วิธีการวัดค่าออฟเซตคือการอ่านค่าข้อมูลจากไจโรขณะที่ IMU อยู่ในสภาวะนิ่ง แล้วนำค่าที่ได้มาใช้เป็น "ออฟเซต" ในการหักลบจากค่าที่อ่านได้ในอนาคต เพื่อชดเชยความคลาดเคลื่อน วิธีนี้ถือเป็นการปรับเทียบเบื้องต้นที่ง่ายที่สุด และเพียงพอสำหรับการใช้งานทั่วไปของไจโรสโคปและ IMU อย่างไรก็ตาม ยังมีขั้นตอนการปรับเทียบไจโรสโคประดับสูงอีกหลายวิธี ผลลัพธ์ที่ได้จะแสดงกราฟของข้อมูลจากไจโรสโคปทั้งสามแกน (Triple-axis Gyroscope) โดย

เปรียบเทียบระหว่างค่าที่ยังไม่ผ่านการปรับเทียบ (Uncalibrated) และค่าที่ผ่านการปรับเทียบแล้ว (Calibrated) ตัวอย่างผลลัพธ์จากการปรับเทียบไจโรสโคปของเราจะแสดงอยู่ด้านล่าง



รูปที่ 1.11 ข้อมูลจากไจโรสโคปเปรียบเทียบระหว่างค่าที่ยังไม่ผ่านการปรับเทียบ (Uncalibrated) และค่าที่ผ่านการปรับเทียบแล้ว (Calibrated)

จากรูปที่ 2.11 จะเห็นได้อย่างชัดเจนว่า การหาค่าออฟเซตนั้นมีความสำคัญอย่างมากในการลดค่าความคลาดเคลื่อนที่ควรจะเป็นศูนย์ ภายใต้สภาวะที่ IMU อยู่กับที่ (steady state) การปรับเทียบด้วยค่าออฟเซตช่วยให้ค่าที่อ่านได้จากไจโรสโคปสะท้อนความเป็นจริงมากขึ้น โดยเฉพาะอย่างยิ่งเมื่อต้องใช้งานในงานที่ต้องการความแม่นยำสูง

เราสามารถทดสอบการปรับเทียบของไจโรสโคปเพิ่มเติมได้โดยการอินทิเกรตค่าความเร็วเชิงมุม (angular velocity) ที่อ่านได้ตลอดเวลา ในขณะที่มีการหมุนที่ทราบค่าล่วงหน้า เช่น การหมุน Calibration Block ของ IMU ไป 180° แล้วนำค่าที่ได้จากไจโรสโคปมาบวกอินทิเกรตตามช่วงเวลา เพื่อยืนยันว่าไจโรสโคปทำงานได้อย่างถูกต้องและแม่นยำตามที่คาดหวัง

โดยการอินทิเกรตค่าความเร็วเชิงมุมจากไจโรสโคป จะสามารถประมาณค่าการกระจัดเชิงมุม (angular displacement) ได้ดังนี้

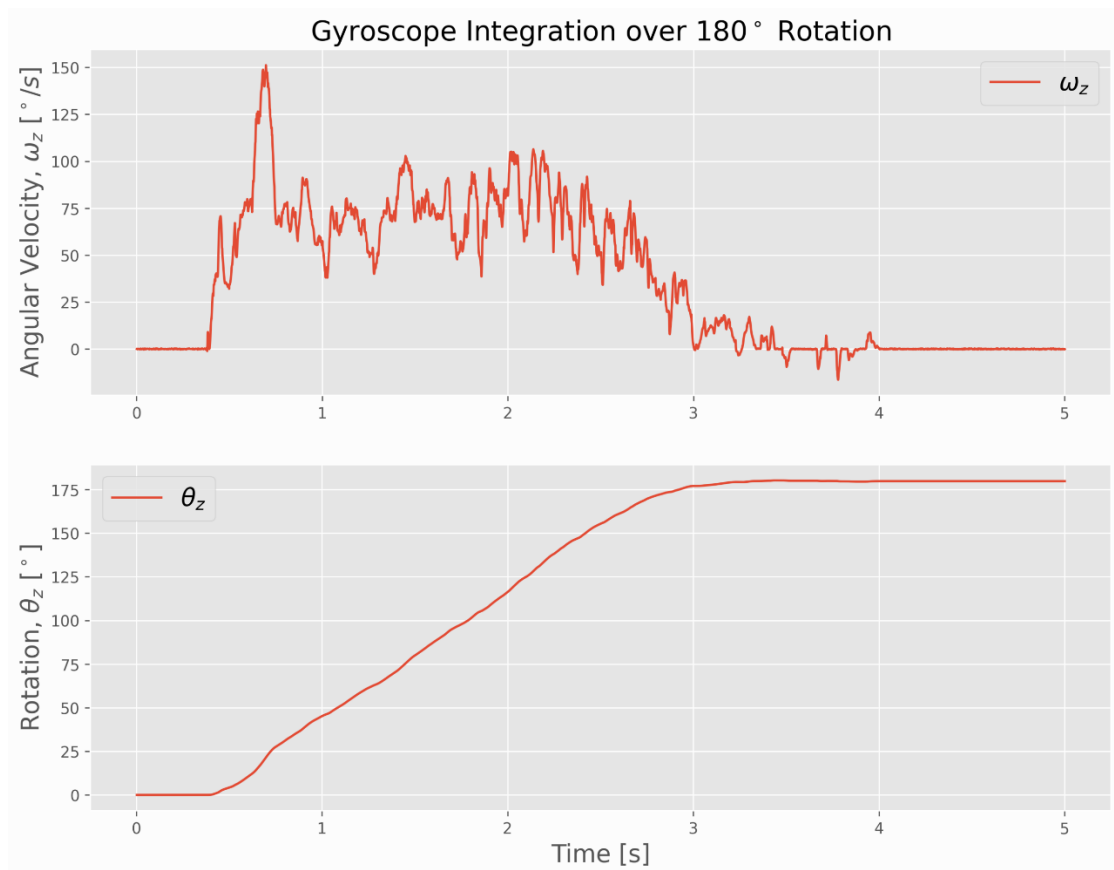
$$\int_{t_1}^{t_2} \omega_i dt = \theta_i \quad (2.27)$$

สำหรับจุดข้อมูลที่เป็นแบบแยกช่วง (discrete points) จำเป็นต้องใช้เทคนิคการอินทิเกรตเชิงตัวเลข (numerical integration) ซึ่งในที่นี้เราใช้วิธีอินทิเกรตแบบแท่งสี่เหลี่ยม (trapezoidal numerical integration):

$$\theta_i = \int_{t_1}^{t_2} \omega_i dt \approx \sum_{j=N}^N \left(\frac{f_i(t_j - 1) + f_i(t_j)}{2} \Delta t_j \right) \quad (2.28)$$

โดยที่ ω_i	คือ อัตราเร็วเชิงมุม (องศา/วินาที)
θ_i	คือ การกระจัดเชิงมุม (องศา)
$f_i(t_j)$	คือ ค่าอัตราเร็วเชิงมุมที่จุดสิ้นสุด (องศา/วินาที)
$f_i(t_j - 1)$	คือ ค่าอัตราเร็วเชิงมุมที่จุดเริ่มต้น (องศา/วินาที)
Δt_j	คือ ขนาดของช่วงเวลา (วินาที)

ด้วยการอินทิเกรตค่าจากไจโรสโคปตัวอย่างการใช้งานการอินทิเกรตไจโรสโคปพร้อมกราฟผลลัพธ์แสดงอยู่ด้านล่าง:

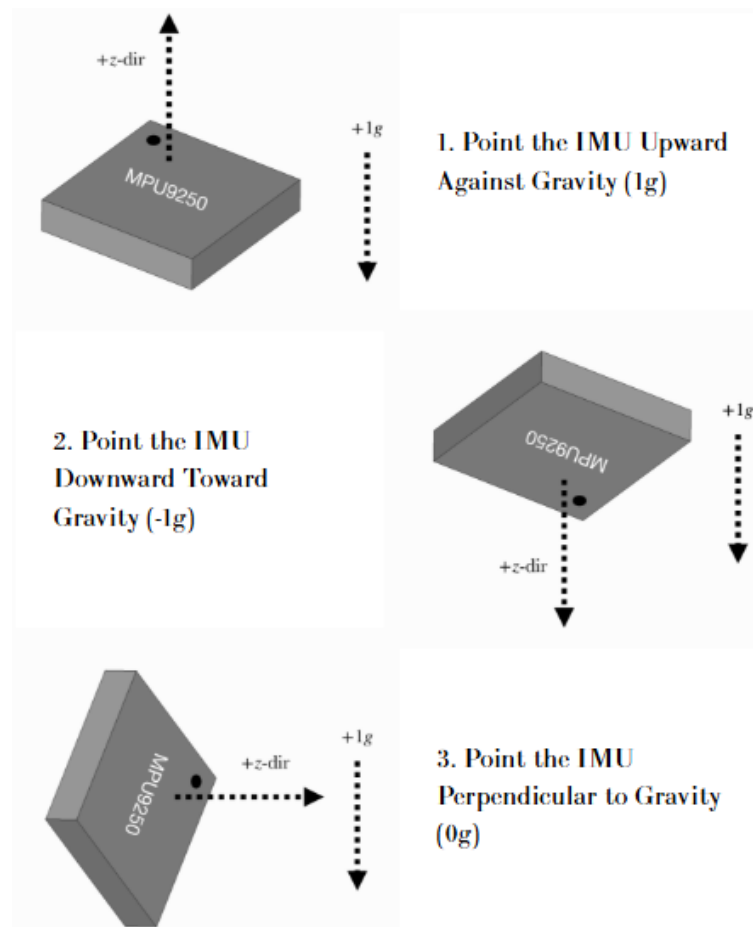


รูปที่ 1.12 การใช้งานการอินทิเกรตไจโรสโคป

ไจโรสโคปประมาณค่าการหมุน 180° ได้ใกล้เคียงที่ประมาณ 179.8° ซึ่งถือว่าค่อนข้างแม่นยำ นอกจากนี้ยังยืนยันได้ว่าทิศทางการหมุนของ IMU ถูกต้องตามจริง เพราะในกรณีนี้การหมุนถูกดำเนินการในทิศทางทวนเข็มนาฬิกา ดังนั้น เราจึงถือได้ว่าการปรับเทียบไจโรสโคปบน MPU9250 (MPU6050) มีความแม่นยำในระดับที่ดี หรือเรียกได้ว่าเป็นการปรับเทียบระดับศูนย์ (zeroth-order calibrated) อย่างไรก็ตาม การทดสอบนี้เป็นการหมุนที่นิ่งและในสภาพแวดล้อมที่เหมาะสมเท่านั้น ในส่วนถัดไป จะเป็นการปรับเทียบแอคเซเลอโรมิเตอร์ในลักษณะที่คล้ายกัน แต่จะการใช้การประมาณระดับหนึ่ง (first-order approximation) ซึ่งซับซ้อนกว่าเล็กน้อยและต้องผ่านการปรับเทียบสามขั้นตอนสำหรับแต่ละแกน แอคเซเลอโรมิเตอร์จะได้รับการตรวจสอบความถูกต้องด้วยการอินทิเกรตเชิงตัวเลขในวิธีที่คล้ายกับไจโรสโคปด้วยเช่นกัน

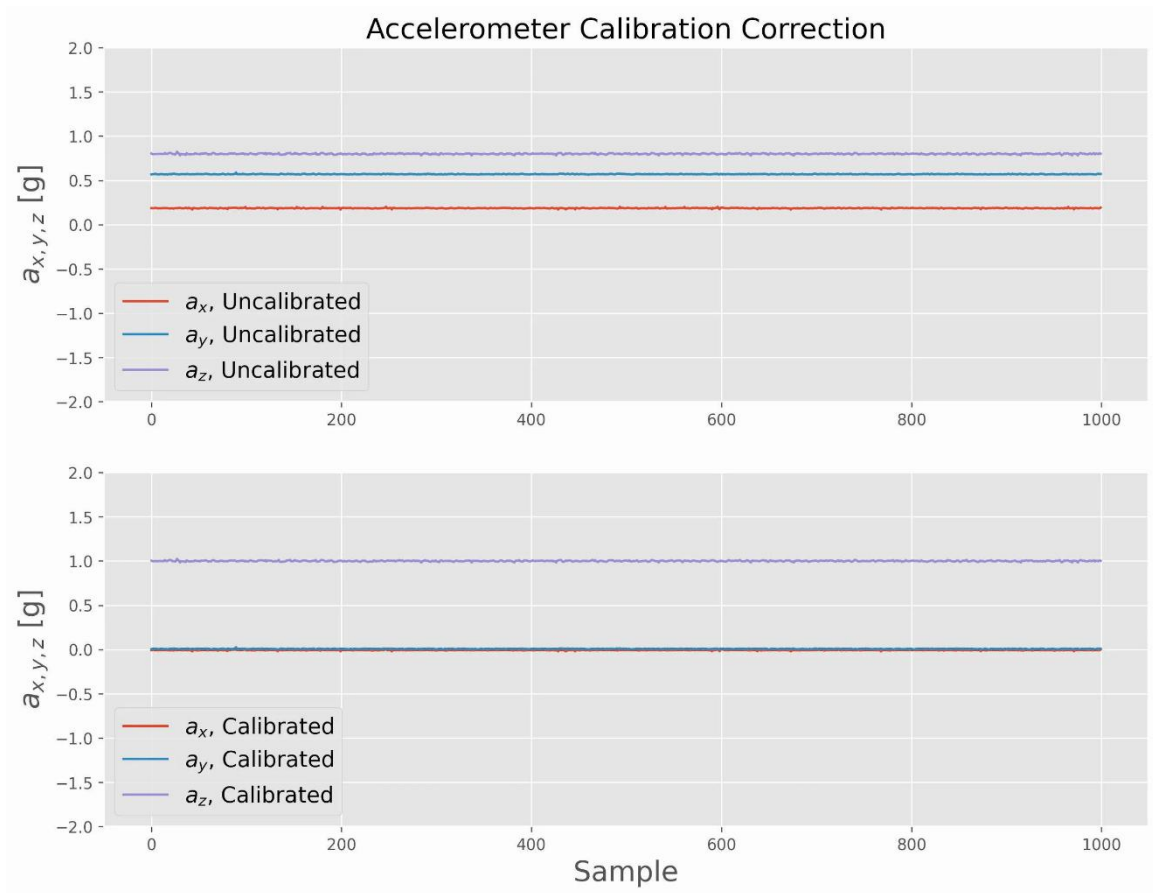
การปรับเทียบแอคเซเลอโรมิเตอร์จำเป็นต้องใช้ประโยชน์จากแรงเร่งเนื่องจากแรงโน้มถ่วง ซึ่งเราสามารถใช้ได้ทั้งในทิศทางบวกและลบของ IMU นอกจากนี้ เรายังสามารถวาง IMU ในแนวตั้งฉากกับแรงโน้มถ่วงเพื่อเก็บค่าการปรับเทียบจุดที่สามได้ วิธีนี้จะทำให้เราได้ค่าที่ไม่ซ้ำกันสามค่า ซึ่งสามารถนำมารวมกันเพื่อสร้างสมการเชิงเส้น (linear fit) ระหว่างค่าที่ได้จากจุดทั้งสามกับค่าที่

เซ็นเซอร์แอคเซเลอโรมิเตอร์แต่ละแกนแสดงออกมา ขั้นตอนการปรับเทียบแอคเซเลอโรมิเตอร์มีดังนี้ (สำหรับแต่ละแกน):



รูปที่ 1.13 ขั้นตอนการปรับเทียบแอคเซเลอโรมิเตอร์

การปรับเทียบทั้งสามจุดนี้ควรทำกับทุกแกนทั้ง 3 แกน โดยค่าทั้งสามจุดจะช่วยให้สามารถทำการฟิตแบบ least-squares สำหรับโมเดลระดับหนึ่ง (first-order model) เพื่อหาค่าปัจจัยการขยาย (scale factor หรือ slope) และค่าออฟเซต (offset หรือ intercept) ของแต่ละแกนในแอคเซเลอโรมิเตอร์ได้อย่างแม่นยำ



รูปที่ 1.14 ข้อมูลจากแอคเซเลอโรมิเตอร์เปรียบเทียบระหว่างค่าที่ยังไม่ผ่านการปรับเทียบ (Uncalibrated) และค่าที่ผ่านการปรับเทียบแล้ว (Calibrated)

การปรับเทียบข้างต้นเป็นการทำกับ IMU โดยให้แกน z ชี้ขึ้นด้านบน ซึ่งเป็นสาเหตุที่เราเห็นค่าแรงเร่งประมาณ 1g ในแกน z และค่าแรงเร่งใกล้เคียงกับศูนย์ในสองแกนที่เหลือ การปรับเทียบแอคเซเลอโรมิเตอร์สามารถตรวจสอบความถูกต้องได้ในลักษณะเดียวกับไจโรสโคป คือโดยการอินทิเกรตเชิงตัวเลข (numerical integration) ความแตกต่างระหว่างการอินทิเกรตของไจโรสโคปกับแอคเซเลอโรมิเตอร์ก็คือ ค่าความเร่งจะถูกอินทิเกรตถึงสองครั้งเพื่อประมาณค่าการกระจัด (displacement) ของ IMU ตามสมการอินทิเกรตดังนี้

$$a_i = \frac{d^2 x_i}{dt^2} \rightarrow \frac{dx_i}{dt} = \int a_i dt \rightarrow x_i = \int \left(\int a_i dt \right) dt \quad (2.29)$$

เช่นเดียวกับก่อนหน้านี้ สมการข้างต้นจำเป็นต้องถูกแปลงให้อยู่ในรูปแบบเชิงดิจิทัล (discretized) เพื่อให้เหมาะกับการใช้งานในสถานการณ์จริง ซึ่งขั้นตอนแรกคือการอินทิเกรตค่าความเร่งเพื่อประมาณค่าความเร็ว (velocity) ก่อน:

$$v_i = \sum_{j=1}^N \left(\frac{a(t_{j-1}) + a(t_j)}{2} \right) \Delta t_j \quad (2.30)$$

ต่อไป ต้องนำอาร์เรย์ความเร็วที่ได้จากการอินทิเกรตครั้งแรก มาทำการอินทิเกรตอีกครั้งเพื่อประมาณค่าการกระจัด (displacement)

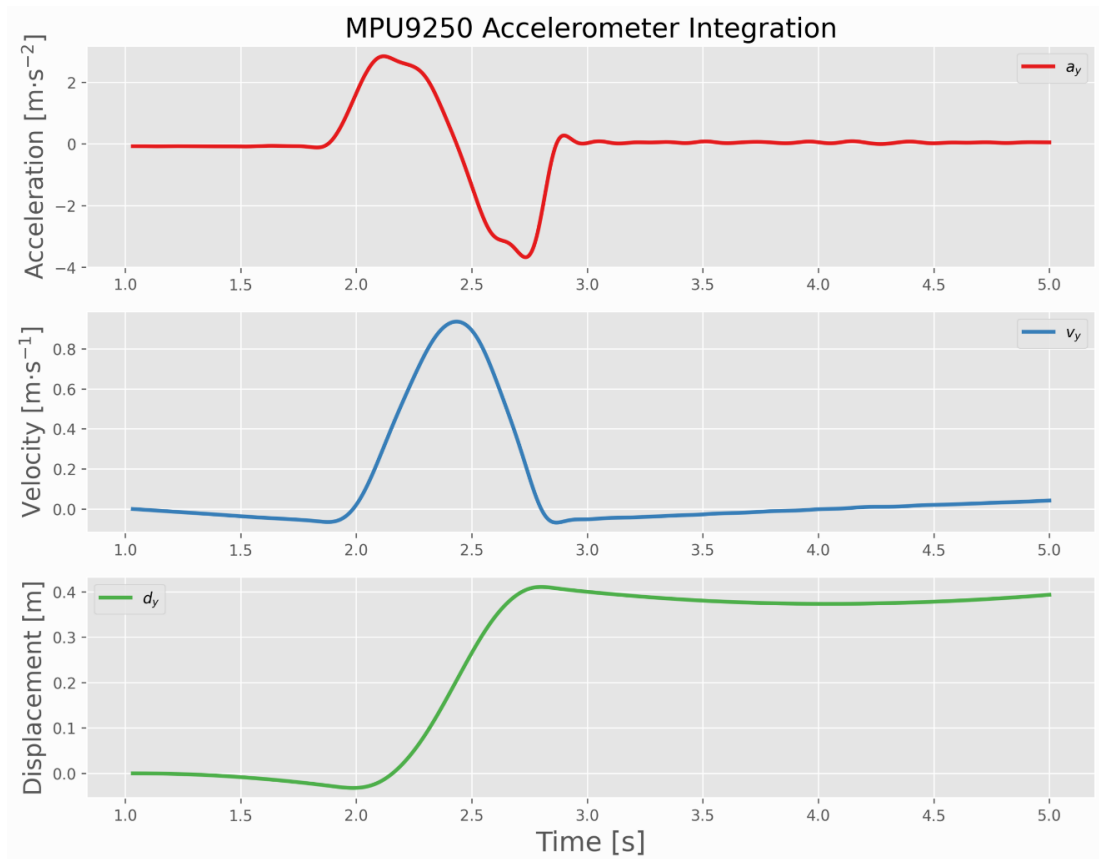
$$x_i = \sum_{j=1}^N \left(\frac{v(t_{j-1}) + v(t_j)}{2} \right) \Delta t_j \quad (2.31)$$

โดยที่ a_i คือ ความเร่ง (เมตร/วินาที²)

v_i คือ ความเร็ว (เมตร/วินาที)

x_i คือ ระยะทาง (เมตร)

การอินทิเกรตครั้งแรกจะได้เวกเตอร์ความเร็ว (velocity vector) ส่วนการอินทิเกรตครั้งที่สองจะได้อาร์เรย์การกระจัด (displacement array) ค่าความเร่งจากแอคเซเลอโรมิเตอร์สามารถนำมาทำการอินทิเกรตในทิศทางเดียวได้ ด้านล่างนี้เป็นการสาธิตการเลื่อน IMU Calibration Block ไปตามเทปวัดระยะในทิศทางเดียว ค่าการกระจัดที่วัดได้จะถูกนำมาเปรียบเทียบกับข้อมูลที่ได้จากการอินทิเกรตสองครั้งของข้อมูลแอคเซเลอโรมิเตอร์ เพื่อใช้ตรวจสอบความถูกต้องของการปรับเทียบแอคเซเลอโรมิเตอร์ต่อไป ตัวอย่างผลลัพธ์สำหรับกรณีทดสอบที่ IMU เลื่อนในทิศทาง +y เป็นระยะทาง 0.5 เมตร จะแสดงดังนี้



รูปที่ 1.15 ผลลัพธ์สำหรับกรณีทดสอบที่ IMU เคลื่อนในทิศทาง +y เป็นระยะทาง 0.5 เมตร

จากรูปข้างต้น เราสามารถสังเกตได้หลายประเด็น ค่าแรงเร่งไม่เป็นศูนย์ แม้จะผ่านการปรับเทียบแล้ว (ซึ่งน่าจะเกิดจากโต๊ะที่ไม่สมดุล) ค่าแรงเร่งเริ่มต้นเป็นบวก (เร่งความเร็ว) ผ่านศูนย์ แล้วกลายเป็นลบ (ลดความเร็ว) ความเร็วมีค่าบวกตลอดเวลา (เคลื่อนที่ไปข้างหน้า) การกระจัดมีค่าบวก (เคลื่อนที่ไปข้างหน้า) ความเร็วและการกระจัดเปลี่ยนแปลงอยู่ตลอด เนื่องจากแรงเร่งไม่สมดุล (โต๊ะไม่เรียบ) การประมาณค่าการกระจัดได้ประมาณ 0.4 เมตร จากระยะทดสอบ 0.5 เมตร ความคลาดเคลื่อนนี้เกิดจากการอินทิเกรตสองครั้งและความไม่แน่นอนของเซ็นเซอร์ รวมถึงการปรับเทียบที่ไม่สมบูรณ์ วิธีแก้ไขคือ ควรตั้ง IMU ให้อยู่ในแนวตั้งฉากกับแรงโน้มถ่วงในระหว่างการปรับเทียบ (โดยใช้ระดับน้ำ) และทำการทดสอบบนพื้นผิวที่เรียบและสมดุลเพื่อเพิ่มความแม่นยำในการวัด

2.6 Magnetometer Hard & Soft Iron Calibration

แมกนีโตมิเตอร์ คือเซ็นเซอร์ที่ใช้วัดความแรงและทิศทางของสนามแม่เหล็กในบริเวณรอบ ๆ ระบบ โดยการวัดสนามแม่เหล็กนี้สามารถนำไปเปรียบเทียบกับแบบจำลองของสนามแม่เหล็กโลก

เพื่อหาทิศทาง (heading) ของระบบเมื่อเทียบกับทิศเหนือแม่เหล็กของโลกได้ อย่างไรก็ตาม ในการใช้งานจริงส่วนใหญ่ สนามแม่เหล็กที่ถูกวัดจะเป็นการรวมกันระหว่างสนามแม่เหล็กของโลกกับสนามแม่เหล็กที่เกิดจากวัตถุใกล้เคียง ซึ่งมักเรียกกันว่า สัญญาณรบกวนทางแม่เหล็ก (magnetic disturbances) เพื่อให้ได้ค่าทิศทางที่แม่นยำ ผลกระทบจากสัญญาณรบกวนทางแม่เหล็กที่อยู่ใกล้เคียงต้องถูกลดทอนหรือชดเชย ในกรณีของสัญญาณรบกวนภายในระบบที่ไม่เปลี่ยนแปลงตามเวลา สามารถชดเชยได้ด้วยการ ปรับเทียบค่าผิดพลาดแบบฮาร์ดและซอฟต์ไอออน เพื่อให้สามารถแก้ไขค่าที่ได้จากเซ็นเซอร์จากแหล่งที่มาของความคลาดเคลื่อนต่างๆ ได้อย่างเป็นระบบ จำเป็นต้องมีการกำหนดโมเดลของเซ็นเซอร์ขึ้นมา แมกนีโตมิเตอร์มัก จะถูกจำลองด้วยสมการต่อไปนี้ ซึ่งเวกเตอร์วัด $\tilde{m}$ ประกอบด้วยสนามแม่เหล็กของโลกและสนาม แม่เหล็กที่เกิดจากสัญญาณรบกวนจากวัตถุใกล้เคียง

$$\tilde{m} = S_I^{-1}(m_E + m_{e(t)}) + b_{HI} + m_{i(t)} \quad (2.32)$$

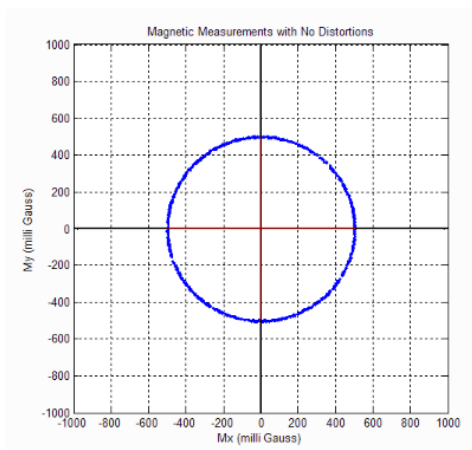
จากโมเดลการวัดนี้ เราสามารถสร้างโมเดลการชดเชยขึ้นมาเพื่อชดเชยผลกระทบจากโลหะแม่เหล็ก แบบแข็ง (hard iron) b_{HI} และแบบอ่อน (soft iron) S_I ซึ่งเป็นความบิดเบือนที่เกิดขึ้นในสนามแม่เหล็กเฉพาะที่ จากแหล่งรบกวนทางแม่เหล็กภายในที่ไม่เปลี่ยนแปลงตามเวลา

$$m_c = S_I(\tilde{m} - b_{HI}) \quad (2.33)$$

$$\begin{bmatrix} m_{cx} \\ m_{cy} \\ m_{cz} \end{bmatrix} = \begin{bmatrix} C_{00} & C_{01} & C_{02} \\ C_{10} & C_{11} & C_{12} \\ C_{20} & C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} \tilde{m}_x - b_{H0} \\ \tilde{m}_y - b_{H1} \\ \tilde{m}_z - b_{H2} \end{bmatrix} \quad (2.34)$$

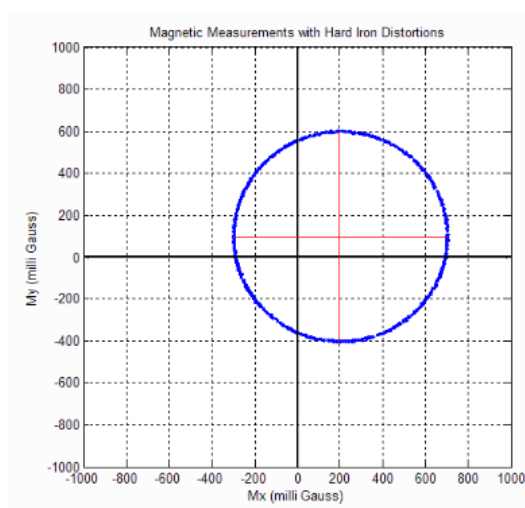
การวัดค่าทางแม่เหล็กจะมีโอกาสเกิดความบิดเบือน (distortion) ซึ่งความบิดเบือนเหล่านี้สามารถแบ่งออกเป็น 2 ประเภทหลัก ได้แก่ ฮาร์ดไอออน (Hard Iron) และ ซอฟต์ไอออน (Soft Iron) ฮาร์ดไอออน (Hard Iron) คือความบิดเบือนที่เกิดจากวัตถุที่สร้างสนามแม่เหล็กขึ้นมาเอง ตัวอย่างเช่น ลำโพง หรือชิ้นส่วนของเหล็กที่มีสนามแม่เหล็ก จะทำให้เกิดฮาร์ดไอออนดิสทอร์ชัน หากวัตถุแม่เหล็กนั้นติดอยู่กับเฟรมอ้างอิงเดียวกันกับเซ็นเซอร์โดยตรง ความบิดเบือนนี้จะส่งผลให้เกิดอคติถาวร (permanent bias) ต่อค่าที่เซ็นเซอร์วัดได้ ซอฟต์ไอออน (Soft Iron) คือความบิดเบือนที่เกิดจากการเบี่ยงเบนหรือการเปลี่ยนแปลงของสนามแม่เหล็กที่มีอยู่เดิม สนามแม่เหล็กจะถูกยืดหรือบิดเบี้ยวตามทิศทางของสนามที่กระทำต่อเซ็นเซอร์ ความบิดเบือนประเภทนี้มักเกิดจากโลหะ เช่น นิกเกิล และเหล็ก โดยทั่วไปแล้ว ความบิดเบือนแบบฮาร์ดไอออนมักจะส่งผลต่อความผิดพลาดรวมที่

ยังไม่ได้รับการแก้ไขมากกว่าความบิดเบือนแบบซอฟต์แวร์ฮาร์ดไอออน ในกรณีที่ไม่มี ความบิดเบือนแบบฮาร์ดไอออนหรือซอฟต์แวร์ฮาร์ดไอออนเกิดขึ้น ค่าที่วัดได้ควรจะมัลักษณะเป็นวงกลมที่มีจุดศูนย์กลางอยู่ที่ $X=0$, $Y=0$ ดังแสดงในรูปที่ 2.10 โดยรัศมีของวงกลมนั้นจะเท่ากับขนาดของสนามแม่เหล็ก



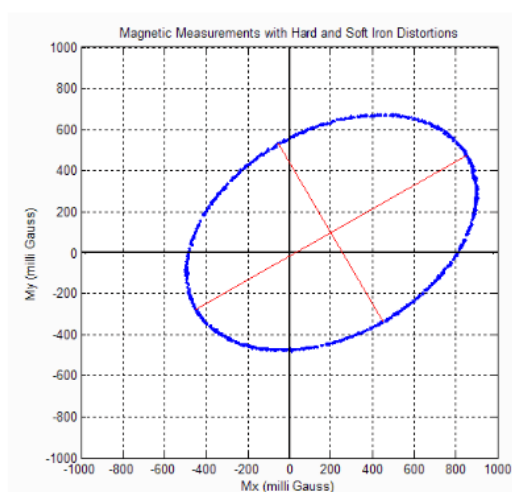
รูปที่ 1.16 ไม่มีความบิดเบือน

ความบิดเบือนแบบฮาร์ดไอออนจะทำให้เกิดอคติถาวรในค่าการวัดสนามแม่เหล็ก ซึ่งส่งผลให้จุดศูนย์กลางของวงกลมเลื่อนออกไป ดังแสดงในรูปที่ 2.11 จุดศูนย์กลางของวงกลมจะอยู่ที่ $X = 200$, $Y = 100$ เมื่อมีความบิดเบือนแบบฮาร์ดไอออนเกิดขึ้น จากข้อมูลนี้สามารถสรุปได้ว่า มีค่าอคติแบบฮาร์ดไอออนเท่ากับ 200 มิลลิแกส (mGauss) ในแกน X และ 100 มิลลิแกส (mGauss) ในแกน Y



รูปที่ 1.17 ความบิดเบือนแบบฮาร์ดไอออน

ความบิดเบือนแบบฮาร์ดไอออนจะทำให้จุดศูนย์กลางของวงกลมเลื่อนออกจากจุดกำเนิดเท่านั้น โดยจะไม่ทำให้รูปร่างของวงกลมเปลี่ยนแปลงแต่อย่างใด ในทางกลับกัน ความบิดเบือนแบบซอฟต์ไอออนจะทำให้สนามแม่เหล็กที่มีอยู่เดิมเกิดการบิดเบี้ยวและแปรเปลี่ยน เมื่อพล็อตค่าที่ได้จากแมกนีโตมิเตอร์ ความบิดเบือนแบบซอฟต์ไอออนจะสามารถสังเกตได้ง่าย เพราะจะทำให้รูปร่างวงกลมเปลี่ยนเป็นรูปวงรี รูปที่ 2.10 แสดงให้เห็นผลกระทบที่เกิดจากทั้งความบิดเบือนแบบฮาร์ดไอออนและซอฟต์ไอออนต่อสัญญาณที่ได้จากแมกนีโตมิเตอร์—โดยวงกลมได้กลายเป็นวงรีที่ไม่ได้อยู่กึ่งกลางที่จุดกำเนิด จุดศูนย์กลางของวงรียังคงอยู่ที่ $X = 200$ และ $Y = 100$ เนื่องจากความบิดเบือนแบบฮาร์ดไอออนยังคงเท่าเดิมวงรีทุกวงจะมีแกนหลัก (major axis) และแกนรอง (minor axis) ซึ่งเป็นแนวยาวและแนวสั้นตามลำดับ ดังที่แสดงในรูปที่ 2.10 วงรีนี้มีแกนหลักเอียงขึ้นจากแนวแกน X ของเฟรมตัวเครื่อง (body frame) ไปประมาณ 30 องศา ซึ่งเป็นผลมาจากความบิดเบือนแบบซอฟต์ไอออน



รูปที่ 1.18 ความบิดเบือนแบบฮาร์ดไอออนและซอฟต์ไอออน

แม้ว่าการพล็อตค่าการวัดสนามแม่เหล็กบนกราฟ 2 มิติจะช่วยให้สามารถมองเห็นผลกระทบจากความบิดเบือนแบบฮาร์ดไอออนและซอฟต์ไอออนได้อย่างชัดเจน แต่ในความเป็นจริงแล้ว ค่าการวัดสนามแม่เหล็กเป็นเวกเตอร์ในรูปแบบ 3 มิติอย่างสมบูรณ์ เมื่อไม่มีความบิดเบือนใด ๆ ค่าการวัดสนามแม่เหล็กแบบสามมิติจะมีลักษณะเป็นทรงกลม โดยมีจุดศูนย์กลางอยู่ที่จุดกำเนิด เช่นเดียวกับในกรณี 2 มิติ ความบิดเบือนแบบฮาร์ดไอออนจะทำให้จุดศูนย์กลางของทรงกลมเลื่อนไปจากจุดกำเนิด ส่วนความบิดเบือนแบบซอฟต์ไอออนจะทำให้ทรงกลมบิดเบี้ยวกลายเป็นทรงรี (ellipsoid)

2.7 Transverse Mercator

การฉายภาพแบบ Transverse Mercator ในรูปแบบต่าง ๆ เป็นระบบพิกัดฉายภาพที่ใช้กันอย่างแพร่หลายที่สุดสำหรับการทำแผนที่ภูมิประเทศทั่วโลกและแผนที่ทางทะเล ระบบนี้เป็น การฉายภาพคงมุม (conformal projection) ซึ่งเส้นเมริเดียนที่เลือกจะถูกฉายออกมาเป็น เส้นตรงที่มีมาตราส่วนคงที่ ก่อนการพัฒนาระบบพิกัด Universal Transverse Mercator (UTM) หลายประเทศในยุโรปได้แสดงให้เห็นประโยชน์ของแผนที่แบบกริดคงมุม (grid-based conformal maps) โดยทำการสร้างแผนที่ประเทศของตนในช่วงระหว่างสงครามโลกครั้งที่หนึ่งและสอง การคำนวณระยะทางระหว่างสองจุดบนแผนที่เหล่านี้สามารถทำได้ง่ายกว่าในภาคสนาม (โดยใช้ ทฤษฎีบทพีทาโกรัส) เมื่อเทียบกับการคำนวณด้วยสูตรตรีโกณมิติที่จำเป็นในระบบพิกัดละติจูด-ลองจิจูดแบบดั้งเดิม ในช่วงหลังสงคราม แนวคิดเหล่านี้ได้ถูกต่อยอดไปสู่ระบบพิกัด Universal Transverse Mercator / Universal Polar Stereographic (UTM/UPS) ซึ่งเป็นระบบแผนที่กริดแบบสากล ครอบคลุมทั่วโลก

การสร้างสมการสำหรับ การฉายภาพแบบ Transverse Mercator ที่อธิบายมาจากการขยายอนุกรม (series expansions) ที่แม่นยำถึงลำดับที่ 3 โดยใช้ third flattening ต่อมาได้ขยายอนุกรมนี้ไปถึงลำดับที่ 6 ทำให้ได้ความแม่นยำแบบ double-precision ครอบคลุมพื้นที่ ภายใน 3,900 กม. จากเส้นเมริเดียนศูนย์กลาง (ประมาณ 57% ของพื้นผิวโลก) ความผิดพลาดจะน้อยกว่า 0.1 มม. สำหรับพื้นที่ ภายใน 7,000 กม. จากเส้นเมริเดียนศูนย์กลาง ซึ่งครอบคลุมประมาณ 89% ของพื้นผิวโลก การสร้างสมการนี้ประกอบด้วย สามขั้นตอนหลัก คือ การฉายภาพคงมุมจากทรงรีไปยังทรงกลม การฉายภาพแบบ Transverse Mercator บนทรงกลม และการปรับแก้การฉายภาพนี้เพื่อให้มีมาตราส่วนคงที่บนเส้นเมริเดียนศูนย์กลาง

พิกัดที่ฉายภาพแล้วถูกกำหนดโดย

$$x_i = \zeta = \xi + i\eta \quad (2.32)$$

$$x = k_0 A \eta \quad (2.33)$$

$$y = k_0 A \xi \quad (2.34)$$

$$A = \frac{a}{1+n} \left(1 + \frac{1}{4} n^2 + \frac{1}{64} n^4 + \frac{1}{256} n^6 \right) \quad (2.35)$$

อนุกรมสำหรับการแปลงระหว่างพิกัดภูมิศาสตร์บนทรงรีและทรงกลม และพิกัดฉายภาพบนทรงรีและทรงกลม ถูกกำหนดในรูปเมทริกซ์ โดยที่ $S(\theta)$ และ N เป็นเวกเตอร์แถวและเวกเตอร์คอลัมน์ที่มีความยาว 6

$$s(\theta) = (\sin 2\theta \quad \sin 4\theta \quad \sin 6\theta \quad \sin 8\theta \quad \sin 10\theta \quad \sin 12\theta) \quad (2.36)$$

$$N = \begin{pmatrix} n \\ n^2 \\ n^3 \\ n^4 \\ n^5 \\ n^6 \end{pmatrix} \quad (2.37)$$

ความสัมพันธ์ระหว่างพิกัดภูมิศาสตร์

$$\lambda' = \lambda \quad (2.38)$$

$$\phi' = \tan^{-1} \sinh (\sinh^{-1} \tan \phi - e \tanh^{-1}(e \sin \phi)) \quad (2.39)$$

แทนที่จะใช้สูตรเชิงวิเคราะห์สำหรับ ϕ' การแปลงระหว่าง ϕ และ ϕ' ใช้การประมาณค่าแบบอนุกรม

$$\phi' = \phi + S(\phi)C_{x,\phi}N \quad (2.40)$$

$$\phi = \phi' + S(\phi')C_{x,\phi}N \quad (2.41)$$

$$C_{x,\phi} = \begin{pmatrix} -2 & \frac{2}{3} & \frac{4}{3} & \frac{-82}{45} & \frac{32}{45} & \frac{4642}{4725} \\ & \frac{5}{3} & \frac{-16}{15} & \frac{-13}{9} & \frac{904}{315} & \frac{-1522}{945} \\ & & \frac{-26}{15} & \frac{34}{24} & \frac{8}{5} & \frac{-24832}{14175} \\ & & & \frac{1237}{630} & \frac{-12}{5} & \frac{-24832}{14175} \\ & & & & \frac{-734}{315} & \frac{109598}{31185} \\ & & & & & \frac{444337}{155925} \end{pmatrix} \quad (2.42)$$

$$C_{\phi,x} = \begin{pmatrix} 2 & \frac{-2}{3} & -2 & \frac{116}{45} & \frac{26}{45} & \frac{-2854}{675} \\ & \frac{7}{3} & -8 & \frac{-227}{45} & \frac{2704}{315} & \frac{2323}{945} \\ & & \frac{5}{56} & \frac{45}{15} & \frac{-1262}{105} & \frac{73814}{2835} \\ & & & \frac{-136}{4279} & \frac{-332}{630} & \frac{-399572}{14175} \\ & & & & \frac{35}{4174} & \frac{-144838}{6237} \\ & & & & & \frac{601676}{22275} \end{pmatrix} \quad (2.43)$$

ความสัมพันธ์ระหว่างพิกัดฉาย

$$\zeta = \zeta' + S(\zeta')C_{\mu,x}N \quad (2.44)$$

$$\zeta' = \zeta + S(\zeta)C_{x,\mu}N \quad (2.45)$$

$$C_{\mu,x} = \begin{pmatrix} \frac{1}{2} & \frac{-2}{3} & \frac{5}{16} & \frac{41}{180} & \frac{-127}{288} & \frac{7891}{37800} \\ & \frac{13}{48} & \frac{-3}{5} & \frac{557}{1440} & \frac{281}{630} & \frac{-1983433}{1935360} \\ & & \frac{61}{240} & \frac{-103}{140} & \frac{15061}{26880} & \frac{167603}{181440} \\ & & & \frac{49561}{161280} & \frac{-179}{168} & \frac{6601661}{7257600} \\ & & & & \frac{34729}{80640} & \frac{-3418889}{1995840} \\ & & & & & \frac{212378941}{319334400} \end{pmatrix} \quad (2.46)$$

$$C_{x,\mu} = \begin{pmatrix} \frac{-1}{2} & \frac{2}{3} & \frac{-37}{96} & \frac{1}{360} & \frac{81}{512} & \frac{-96199}{604800} \\ & \frac{-1}{48} & \frac{-1}{15} & \frac{437}{1440} & \frac{-46}{105} & \frac{1118711}{3870720} \\ & & \frac{-17}{480} & \frac{37}{840} & \frac{209}{4480} & \frac{-5569}{90720} \\ & & & \frac{-4397}{161280} & \frac{11}{504} & \frac{830251}{7257600} \\ & & & & \frac{-4583}{161280} & \frac{108847}{3991680} \\ & & & & & \frac{-20648693}{638668800} \end{pmatrix} \quad (2.47)$$

สูตรด้านล่างนี้อธิบายอัลกอริทึมที่ใช้ในการระบุตัวเลือกโดยประมาณ ใช้ได้ภายในระยะประมาณ 5 องศาจากเส้นเมริเดียนกลางเท่านั้น นี่คือการะยะทางเมริเดียน

$$N = \frac{k_0}{(1 - e^2 \sin^2 \phi)^{1/2}} \quad (2.48)$$

$$R = \frac{k_0(1 - e^2)}{(1 - e^2 \sin^2 \phi)^{3/2}} \quad (2.49)$$

$$t = \tan \phi \quad (2.50)$$

$$\eta = \frac{e^2}{1 - e^2} \cos^2 \phi \quad (2.51)$$

$$\begin{aligned} x = k_0 \lambda \cos \phi &+ \frac{k_0 \lambda^3 \cos^3 \phi}{3!} (1 - t^2 + \eta^2) \\ &+ \frac{k_0 \lambda^5 \cos^5 \phi}{5!} (5 - 18t^2 + t^4 + 14\eta^2 - 58t^2 \eta^2) \\ &+ \frac{k_0 \lambda^7 \cos^7 \phi}{7!} (61 - 479t^2 + 179t^4 + t^6) \end{aligned} \quad (2.52)$$

$$\begin{aligned} y = M(\phi) &+ \frac{k_0 \lambda^2 \sin \phi \cos \phi}{2!} + \frac{k_0 \lambda^4 \sin \phi \cos^3 \phi (5 - t^2 + 9\eta^2 + 4\eta^4)}{4!} \\ &+ \frac{k_0 \lambda^6 \sin \phi \cos^5 \phi (61 - 58t^2 + t^4 + 270\eta^2 - 330t^2 \eta^2)}{6!} \\ &+ \frac{k_0 \lambda^8 \sin \phi \cos^7 \phi (1385 - 3111t^2 + 543t^4 - t^6)}{8!} \end{aligned} \quad (2.53)$$

โดยที่ ตัวแปลพิกัดและจำนวนเชิงซ้อน

- x, y คือ พิกัดฉายภาพบนระนาบ (เมตร)
- ζ คือ จำนวนเชิงซ้อนของพิกัดฉายภาพ
- ξ คือ ส่วนจินตภาพของจำนวนเชิงซ้อน
- η คือ ส่วนจริงของจำนวนเชิงซ้อน
- $\emptyset$ คือ ละติจูด (เรเดียน)
- Λ คือ ลองจิจูด (เรเดียน)
- $\emptyset'$ คือ ละติจูดบนทรงกลม (เรเดียน)
- λ' คือ ลองจิจูด บนทรงกลม (เรเดียน)

ζ' คือ จำนวนเชิงซ้อนที่เกี่ยวข้องกับการแปลงพิกัด

ค่าคงที่และค่าพารามิเตอร์ของโลก

a คือ ความยาวกึ่งแกนเอกของทรงรีโลก (เมตร)

e คือ ความเยื้องศูนย์กลางเชิงตัวเลขของทรงรีโลก

n คือ Third flattening

k_0 คือ ตัวประกอบมาตราส่วนบนเส้นเมริเดียนศูนย์กลาง

A คือ ค่าคงที่ ที่ขึ้นอยู่กับ a และ n (เมตร)

$M(\emptyset)$ คือ ระยะทางเมริเดียน (เมตร)

ตัวแปรและเมทริกซ์ในอนุกรม

N คือ เวกเตอร์คอลัมน์

$s(\theta)$ คือ เวกเตอร์แถว

$C_{x,\emptyset}$ คือ เมทริกซ์สัมประสิทธิ์สำหรับการแปลงจาก $\emptyset$ ไป $\emptyset'$

$C_{\emptyset,x}$ คือ เมทริกซ์สัมประสิทธิ์สำหรับการแปลงจาก $\emptyset'$ ไป $\emptyset$

$C_{\mu,x}$ คือ เมทริกซ์สัมประสิทธิ์สำหรับการแปลงจาก ζ' ไป ζ

$C_{x,\mu}$ คือ เมทริกซ์สัมประสิทธิ์สำหรับการแปลงจาก ζ ไป ζ'

N คือ รัศมีตามแนวตั้งฉาก (เมตร)

R คือ รัศมีความโค้งในแนวเมริเดียน (เมตร)

2.8 Covariance matrix

เมทริกซ์ความแปรปรวนร่วมคือเมทริกซ์สี่เหลี่ยมขององค์ประกอบที่แสดงถึงค่าความแปรปรวนร่วมระหว่างตัวแปรแต่ละคู่ในชุดข้อมูลหรือเวกเตอร์ที่กำหนดให้ เพื่อให้เข้าใจความแปรปรวนร่วมได้ดีขึ้น เราต้องมาทำความเข้าใจความแปรปรวนกันก่อน ความแปรปรวนอธิบายถึงความแปรผันของค่าในตัวแปร ขึ้นอยู่กับระยะห่างระหว่างค่าต่างๆ สูตรเพื่อทำความเข้าใจว่าความแปรปรวนคำนวณได้อย่างไร

กรณีรู้ค่าเฉลี่ย

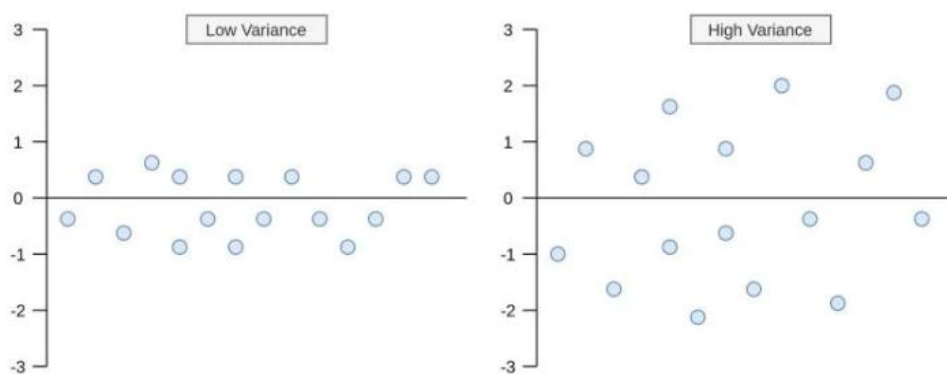
$$var(x) = \frac{\sum_i^n (x_i - \mu)^2}{N} \quad (2.54)$$

กรณีไม่รู้ค่าเฉลี่ย

$$var(x) = \frac{\sum_i^n (x_i - \bar{x})^2}{N - 1} \quad (2.55)$$

โดยที่	$var(x)$	คือ ความแปรปรวนของชุดข้อมูล x
	x_i	คือ ข้อมูลแต่ละตัว ในชุดข้อมูล
	μ	คือ ค่าเฉลี่ยของประชากร
	$\bar{x}$	คือ ค่าเฉลี่ยของกลุ่มตัวอย่าง
	N	คือ ขนาดของประชากร

ในสูตรนี้ ค่าแต่ละค่าในตัวแปรจะลบออกจากค่าเฉลี่ยของตัวแปรนั้น หลังจากนั้นนำค่าความแตกต่างยกกำลังสองแล้ว จะถูกหารด้วยจำนวนค่า (N) ในตัวแปรนั้น เกิดอะไรขึ้นเมื่อความแปรปรวนต่ำหรือสูง ดังที่เห็นในภาพด้านล่าง เกิดอะไรขึ้นเมื่อค่าความแปรปรวนต่ำหรือสูง



รูปที่ 1.19 แสดงความแปรปรวนของข้อมูล

ความแปรปรวนร่วมคือการหาค่าที่บ่งชี้ว่าตัวแปรสองตัวนี้แปรผันกันอย่างไร ในสูตรความแปรปรวนร่วม ค่าของตัวแปรทั้งสองจะถูกคูณด้วยผลต่างจากค่าเฉลี่ย สามารถดูได้ในสูตรด้านล่าง กรณีรู้ค่าเฉลี่ย

$$cov(x, y) = \frac{\sum_i^n (x_i - \mu_x) \cdot (y_i - \mu_y)}{N} \quad (2.56)$$

โดยที่ $cov(x, y)$ = ความแปรปรวนร่วมของประชากร

x_i = ข้อมูลแต่ละตัวในชุดที่หนึ่ง

y_i = ข้อมูลแต่ละตัวในชุดที่สอง

μ_x = ค่าเฉลี่ยของประชากรชุดที่หนึ่ง

μ_y = ค่าเฉลี่ยของประชากรชุดที่สอง

N = ขนาดของประชากร

กรณีไม่รู้ค่าเฉลี่ย

$$cov(x, y) = \frac{\sum_i^n (x_i - \bar{x}) \cdot (y_i - \bar{y})}{N - 1} \quad (2.57)$$

โดยที่ $cov(x, y)$ = ความแปรปรวนร่วมของตัวอย่าง

x_i = ข้อมูลแต่ละตัวในชุดที่หนึ่ง

y_i = ข้อมูลแต่ละตัวในชุดที่สอง

$\bar{x}$ = ค่าเฉลี่ยของตัวอย่างของชุดข้อมูล x

$\bar{y}$ = ค่าเฉลี่ยของตัวอย่างของชุดข้อมูล y

N = ขนาดของตัวอย่าง

ความแตกต่างเพียงอย่างเดียวระหว่างความแปรปรวนและความแปรปรวนร่วมคือ เราใช้ค่าและค่าเฉลี่ยของตัวแปรสองตัวแทนที่จะเป็นตัวเดียวในความแปรปรวนร่วม ความแปรปรวนร่วมสามารถคำนวณได้ระหว่างตัวแปรสองตัวเท่านั้น เมทริกซ์ความแปรปรวนร่วมจึงหมายถึงค่าความแปรปรวนร่วมของตัวแปรแต่ละคู่ในข้อมูลหลายตัวแปร นอกจากนี้ ความแปรปรวนร่วมระหว่างตัวแปรเดียวกันจะเท่ากับค่าความแปรปรวน ดังนั้น เส้นทแยงมุมจะแสดงความแปรปรวนของแต่ละตัวแปร สมมติว่ามีตัวแปรสองตัวคือ x y และ z ในชุดข้อมูลของเรา เมทริกซ์ความแปรปรวนร่วมควรมีลักษณะตามสูตรด้านล่าง

$$cov(x, y, z) = \begin{bmatrix} var(x) & cov(x, y) & cov(x, z) \\ cov(x, y) & var(y) & cov(y, z) \\ cov(x, z) & cov(y, z) & var(z) \end{bmatrix} \quad (2.58)$$

เมทริกซ์สมมาตรแสดงความแปรปรวนร่วมของตัวแปรแต่ละคู่ ค่าเหล่านี้ในเมทริกซ์ความแปรปรวนร่วมแสดงขนาดและทิศทางการกระจายตัวของข้อมูลหลายตัวแปรในปริภูมิหลายมิติ การควบคุมค่าเหล่านี้ช่วยให้เราสามารถรวบรวมข้อมูลเกี่ยวกับการกระจายตัวของข้อมูลระหว่างสองมิติได้

2.9 ระบบปฏิบัติการหุ่นยนต์ (Robot Operating System)

ก่อนที่จะมีระบบปฏิบัติการหุ่นยนต์ นักออกแบบหุ่นยนต์และนักวิจัยหุ่นยนต์ทุกคนต้องใช้เวลามากในการออกแบบซอฟต์แวร์ฝังตัวภายในหุ่นยนต์ รวมถึงตัวฮาร์ดแวร์เอง ซึ่งจำเป็นต้องอาศัยทักษะด้านวิศวกรรมเครื่องกล อิเล็กทรอนิกส์ และการเขียนโปรแกรมแบบฝังตัว โดยทั่วไปโปรแกรมที่ออกแบบในลักษณะนี้จะมีความคล้ายคลึงกับการเขียนโปรแกรมแบบฝังตัว คล้ายกับอิเล็กทรอนิกส์ มากกว่าที่จะเป็นหุ่นยนต์ในความหมายที่แท้จริงที่สุด ดังเช่นที่เราอาจพบเห็นในปัจจุบันในหุ่นยนต์บริการ มีการนำโปรแกรมกลับมาใช้ซ้ำจำนวนมาก เนื่องจากโปรแกรมเหล่านี้มีความเชื่อมโยงอย่างแน่นกับฮาร์ดแวร์พื้นฐาน แนวคิดหลักของระบบปฏิบัติการหุ่นยนต์คือการหลีกเลี่ยงการประดิษฐ์ขึ้นมาใหม่ซ้ำแล้วซ้ำเล่า และนำเสนอฟังก์ชันมาตรฐานในการทำการแยกฮาร์ดแวร์ออกเป็นส่วนๆ เช่นเดียวกับระบบปฏิบัติการทั่วไปสำหรับพีซี ดังนั้นจึงมีชื่อที่คล้ายกัน ROS เป็นผู้อำนวยความสะดวกให้กับโครงการหุ่นยนต์ นักวิจัยหรือวิศวกรในฝ่ายวิจัยและพัฒนาที่ใช้ ROS ไม่ต้องเสียเวลาสร้างระบบนิเวศใหม่ให้กับโครงการหุ่นยนต์ใหม่แต่ละโครงการอีกต่อไป ซึ่งถือเป็นผลประโยชน์ทางการเงินอีกด้วย ROS ได้รับการพัฒนาและดูแลโดย Willow Garage บริษัทจากแคลิฟอร์เนีย ก่อตั้งขึ้นในปี พ.ศ. 2549 โดย Scott Hassan แนวคิดของพวกเขาคือ หากเราต้องการเห็นหุ่นยนต์เข้าถึงบ้านของเรา จำเป็นต้องเร่งการวิจัยโดยการจัดหาฮาร์ดแวร์และซอฟต์แวร์พื้นฐานที่แข็งแกร่งและเป็นโอเพนซอร์ส



รูปที่ 1.20 รูปบริษัท Willow Garage

โดยปรัชญาของ ROS สามารถสรุปได้เป็นหลักการดังนี้

Peer to Peer: หุ่นยนต์ที่มีความซับซ้อนเพียงพอประกอบด้วยคอมพิวเตอร์หรือบอร์ดคอมพิวเตอร์หลายตัวที่เชื่อมต่อผ่านอินเทอร์เน็ต และบางครั้งอาจมีคอมพิวเตอร์นอกบอร์ดสำหรับงานคำนวณที่ต้องใช้การประมวลผลอย่างหนัก สถาปัตยกรรมเพียร์ทูเพียร์ที่เชื่อมต่อกับระบบบัพเฟอร์และระบบค้นหา ช่วยให้แต่ละส่วนประกอบสามารถโต้ตอบโดยตรงกับส่วนประกอบอื่นๆ ได้ ทั้งแบบซิงโครนัสและอะซิงโครนัสตามต้องการ

Multi-language: ROS เป็นกลางทางภาษา และสามารถเขียนโปรแกรมได้หลายภาษา ข้อกำหนดของ ROS ทำงานที่เลเยอร์การส่งข้อความ การเชื่อมต่อแบบเพียร์ทูเพียร์จะดำเนินการใน XML-RPC ซึ่งมีอยู่ในหลายภาษา เพื่อรองรับภาษาใหม่ คลาส C++ จะถูกห่อหุ้มใหม่หรือถูกเขียนคลาสเพื่อให้สามารถสร้างข้อความได้ ข้อความเหล่านี้จะอธิบายไว้ใน IDL (Interface Definition Language)

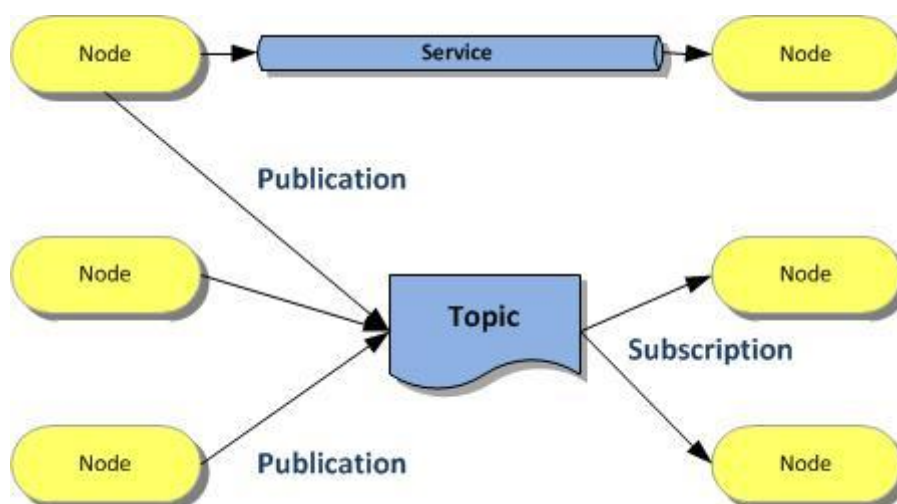
Tools-based: แทนที่จะใช้สภาพแวดล้อมรันไทม์แบบโมนอลิธิก ROS ได้นำการออกแบบไมโครเคอร์เนลมาใช้ ซึ่งใช้เครื่องมือขนาดเล็กจำนวนมากเพื่อสร้างและรันส่วนประกอบต่างๆ ของ ROS เมื่อคุณได้ศึกษาบทช่วยสอน ROS แล้ว คุณจะได้เรียนรู้การใช้คำสั่งต่างๆ ที่ใช้จัดการโหนดและข้อความ แต่ละคำสั่งสามารถรันได้ ข้อดีของระบบนี้คือปัญหาที่เกิดขึ้นกับไฟล์ปฏิบัติการหนึ่งจะไม่ส่งผลกระทบต่อไฟล์ปฏิบัติการอื่นๆ ซึ่งทำให้ระบบมีความแข็งแกร่งและยืดหยุ่นกว่าระบบที่ใช้สภาพแวดล้อมรันไทม์แบบรวมศูนย์

Thin: เพื่อต่อสู้กับการพัฒนาอัลกอริทึมที่พันกันยุ่งเหยิงน้อยหรือมากกับระบบปฏิบัติการหุ่นยนต์ และยากต่อการนำมาใช้ซ้ำในภายหลัง นักพัฒนา ROS จึงตั้งใจให้ไดรเวอร์และอัลกอริทึมอื่นๆ รวมอยู่ในไฟล์ปฏิบัติการแบบสแตนด์อโลน วิธีนี้ช่วยให้สามารถนำกลับมาใช้ซ้ำได้สูงสุด และเหนือสิ่งอื่นใดคือช่วยลดขนาดไฟล์ วิธีนี้ทำให้ ROS ใช้งานง่าย เนื่องจากความซับซ้อนของไลบรารีถูกจัดวางอยู่ในไลบรารี การจัดวางแบบนี้ยังเอื้อต่อการทดสอบยูนิตอีกด้วย สุดท้ายนี้ ROS ใช้โค้ด (ไดรเวอร์และอัลกอริทึม) จากโครงการโอเพนซอร์สอื่นๆ

Free and open source: เราได้อธิบายเหตุผลของตัวเลือกนี้ไว้แล้ว อย่างไรก็ตาม โปรดทราบว่าสถาปัตยกรรมที่เลือกนั้นสอดคล้องกับตัวเลือกนั้น ROS ส่งข้อมูลระหว่างโมดูลโดยใช้การสื่อสารระหว่างกระบวนการ ส่งผลให้โมดูลไม่จำเป็นต้องเชื่อมโยงกันภายในกระบวนการเดียว ทำให้สามารถใช้สิทธิ์การใช้งานที่แตกต่างกันได้ หลักการพื้นฐานของระบบปฏิบัติการหุ่นยนต์คือการรันไฟล์ปฏิบัติการจำนวนมากแบบขนาน ซึ่งจำเป็นต้องสามารถแลกเปลี่ยนข้อมูลได้ทั้งแบบซิงโครนัสและอะซิงโครนัส ตัวอย่างเช่น ระบบปฏิบัติการหุ่นยนต์จำเป็นต้องสอบถามข้อมูลจากเซ็นเซอร์หุ่นยนต์ที่ความถี่ที่กำหนด (เช่น เซ็นเซอร์วัดระยะอัลตราซาวนด์หรืออินฟราเรด เซ็นเซอร์วัดความดัน เซ็นเซอร์วัดอุณหภูมิ ไจโรสโคป เครื่องวัดความเร่ง กล้อง ไมโครโฟน ฯลฯ) ดึงข้อมูลเหล่านี้ ประมวลผล

(ดำเนินการที่เรียกว่าการรวมข้อมูล) ส่งต่อไปยังอัลกอริทึมการประมวลผล (เช่น การประมวลผลเสียง การมองเห็นแบบเทียม SLAM – การระบุตำแหน่งและการทำแผนที่พร้อมกัน ฯลฯ) และสุดท้ายคือการควบคุมมอเตอร์ กระบวนการทั้งหมดนี้ดำเนินการอย่างต่อเนื่องและแบบขนาน ยิ่งไปกว่านั้น ระบบปฏิบัติการหุ่นยนต์ยังต้องบริหารจัดการการแย่งชิงทรัพยากรเพื่อให้มั่นใจว่าสามารถเข้าถึงทรัพยากรของหุ่นยนต์ได้อย่างมีประสิทธิภาพ

แนวคิดที่รวบรวมไว้ใน ROS ภายใต้ชื่อ “ROS Computation Graph” เพื่อให้สามารถบรรลุวัตถุประสงค์เหล่านี้ได้อธิบายไว้ด้านล่างนี้ แนวคิดเหล่านี้เป็นแนวคิดที่ระบบใช้งานขณะที่ระบบกำลังทำงาน ในขณะที่ระบบไฟล์ ROS ที่อธิบายไว้ในหัวข้อก่อนหน้านี้เป็นแนวคิดแบบคงที่



รูปที่ 1.21 ROS Computation Graph

Node: ROS จัดการกับปัญหานี้ทั้งหมดโดยใช้แนวคิดพื้นฐานง่ายๆ แนวคิดแรกคือแนวคิดเกี่ยวกับโหนดใน ROS โหนดคืออินสแตนซ์ของไฟล์ปฏิบัติการ โหนดอาจเทียบเท่ากับเซ็นเซอร์ มอเตอร์ อัลกอริทึมการประมวลผล หรือการตรวจสอบ และอื่นๆ ทุกโหนดที่เริ่มทำงานจะประกาศตัวเองต่อ Master ซึ่งย้อนกลับไปยังสถาปัตยกรรมไมโครเคอร์เนล ซึ่งทรัพยากรแต่ละรายการจะเป็นโหนดอิสระ ด้วยระบบโหนด ฟังก์ชันพื้นฐานสามารถเป็นมาตรฐาน และสามารถพัฒนาบล็อกเทคโนโลยีได้อย่างรวดเร็ว และนำกลับมาใช้ใหม่ แก้ไข หรือปรับปรุงได้อย่างง่ายดาย

Master: บริการประกาศและลงทะเบียนโหนด ซึ่งทำให้โหนดต่างๆ สามารถค้นหาและแลกเปลี่ยนข้อมูลกันได้ Master ดำเนินการผ่าน XMLRPC Master ประกอบด้วยส่วนประกอบที่มีการใช้งานอย่างแพร่หลายที่เรียกว่า Parameter Server ซึ่งดำเนินการในรูปแบบของ XMLRPC

เช่นกัน และตามชื่อก็บอกอยู่แล้วว่าเป็นฐานข้อมูลส่วนกลางที่โหนดต่างๆ สามารถจัดเก็บข้อมูลและแชร์ข้อมูลร่วมกันทั่วทั้งระบบ

Topics: ข้อมูลมีการแลกเปลี่ยนแบบอะซิงโครนัสผ่าน topic และแบบซิงโครนัสผ่าน service topic คือระบบการขนส่งข้อมูลที่ใช้ระบบสมัครสมาชิก/เผยแพร่ โหนดอย่างน้อยหนึ่งโหนดสามารถเผยแพร่ข้อมูลไปยัง topic และโหนดอย่างน้อยหนึ่งโหนดสามารถอ่านข้อมูลใน topic นั้นได้ในอีกแง่หนึ่ง topic เปรียบเสมือนบัสข้อความแบบอะซิงโครนัส คล้ายกับฟีด RSS แนวคิดของบัสแบบหลายต่อหลายแบบอะซิงโครนัสนี้มีความสำคัญอย่างยิ่งในสถานการณ์ของระบบแบบกระจาย topic ถูกพิมพ์ ซึ่งหมายความว่าประเภทของข้อมูลที่เผยแพร่ (ข้อความ) จะถูกจัดโครงสร้างในลักษณะเดียวกันเสมอ โหนดส่งและรับข้อความใน topic

Messages: ข้อความคือโครงสร้างข้อมูลแบบผสม ข้อความประกอบด้วยชนิดข้อมูลพื้นฐาน (สตริงอักขระ บูลีน จำนวนเต็ม จุดลอยตัว ฯลฯ) และข้อความ (ข้อความที่เป็นโครงสร้างแบบเรียกซ้ำ) ตัวอย่างเช่น โหนดที่เป็นตัวแทนของมอเตอร์เซอร์โวของหุ่นยนต์จะเผยแพร่สถานะของตนในหัวข้อใดหัวข้อหนึ่ง (ขึ้นอยู่กับวิธีที่คุณตั้งโปรแกรม) พร้อมข้อความที่มีข้อมูล เช่น จำนวนเต็มแทนตำแหน่งของมอเตอร์ จุดลอยตัวแทนอุณหภูมิ จุดลอยตัวแทนความเร็ว และอื่นๆ

Service: หัวข้อ (topic) คือวิธีการสื่อสารแบบอะซิงโครนัสที่ใช้สำหรับการสื่อสารแบบหลายต่อหลาย บริการนี้ตอบสนองความต้องการที่แตกต่างกัน นั่นคือการสื่อสารแบบซิงโครนัสระหว่างสองโหนด แนวคิดนี้คล้ายคลึงกับการเรียกโปรซีเจอร์ระยะไกล

Bag: เป็นรูปแบบสำหรับจัดเก็บและเล่นข้อมูลข้อความ กลไกนี้ทำให้สามารถรวบรวมข้อมูลที่วัดโดยเซ็นเซอร์และเล่นซ้ำได้หลายครั้งตามต้องการเพื่อจำลองข้อมูลจริง นอกจากนี้ยังเป็นระบบที่มีประโยชน์อย่างมากสำหรับการดีบักระบบหลังจากเหตุการณ์เกิดขึ้น

2.10 Coordinate Frames for Mobile Platforms

นักพัฒนาไดรเวอร์ โมเดล และไลบรารีจำเป็น[12]ต้องมีข้อตกลงร่วมกันสำหรับเฟรมพิกัดเพื่อให้สามารถบูรณาการและนำส่วนประกอบซอฟต์แวร์กลับมาใช้ใหม่ได้ดีขึ้น ข้อตกลงร่วมกันสำหรับเฟรมพิกัดจะให้ข้อมูลจำเพาะสำหรับนักพัฒนาที่สร้างไดรเวอร์และโมเดลสำหรับฐานเคลื่อนที่ในทำนองเดียวกัน นักพัฒนาที่สร้างไลบรารีและแอปพลิเคชันสามารถใช้ซอฟต์แวร์ของตนกับฐานเคลื่อนที่ต่าง ๆ ที่เข้ากันได้กับข้อมูลจำเพาะนี้ได้ง่ายขึ้น

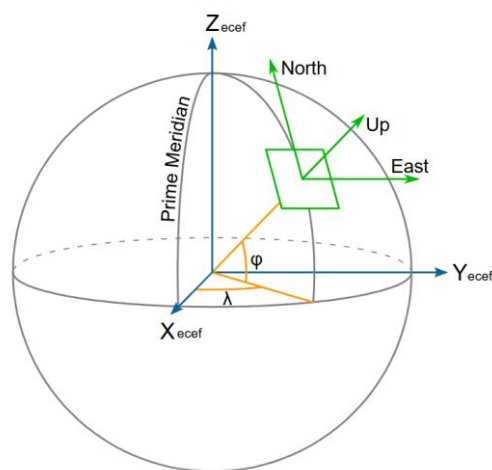
Base_link กรอบพิกัดที่เรียกว่า base_link จะถูกแนบเข้ากับฐานหุ่นยนต์เคลื่อนที่อย่างแน่นหนา โดยสามารถแนบ base_link เข้ากับฐานได้ในตำแหน่งหรือทิศทางใดก็ได้ตามต้องการ

Odom กรอบพิกัดที่เรียกว่า odom เป็นกรอบที่คงที่ทั่วโลก ตำแหน่งของแพลตฟอร์มเคลื่อนที่ในกรอบ odom สามารถเลื่อนไปมาได้ตามเวลาโดยไม่มีขอบเขต เลื่อนไปมานี้ทำให้กรอบ odom ไร้ประโยชน์ในการใช้เป็นข้อมูลอ้างอิงทั่วโลกในระยะยาว อย่างไรก็ตาม ตำแหน่งของหุ่นยนต์

ในกรอบ odom รับประกันว่าจะต่อเนื่อง ซึ่งหมายความว่าตำแหน่งของแพลตฟอร์มเคลื่อนที่ในกรอบ odom จะพัฒนาไปอย่างราบรื่นเสมอโดยไม่มีการกระโดดแยกจากกัน ในการตั้งค่าทั่วไป กรอบ odom จะถูกคำนวณโดยอิงจากแหล่งโอโดเมตรี เช่น โอโดเมตริล้อ โอโดเมตรีภาพ หรือหน่วยวัดแรงเฉื่อย กรอบ odom มีประโยชน์ในการใช้เป็นข้อมูลอ้างอิงในท้องถิ่นในระยะสั้นที่แม่นยำ แต่การเลื่อนไปมาทำให้ไม่เหมาะสำหรับใช้เป็นข้อมูลอ้างอิงในระยะยาว

Map กรอบพิกัดที่เรียกว่าแผนที่เป็นกรอบพิกัดโลกที่คงที่ โดยมีแกน Z ชี้ขึ้นด้านบน ตำแหน่งของแพลตฟอร์มเคลื่อนที่เมื่อเทียบกับกรอบแผนที่ไม่ควรเลื่อนไปตามเวลามากนัก กรอบแผนที่ต่อเนื่อง ซึ่งหมายความว่าตำแหน่งของแพลตฟอร์มเคลื่อนที่ในกรอบแผนที่สามารถเปลี่ยนแปลงได้เป็นระยะๆ ตลอดเวลา ในการตั้งค่าทั่วไป ส่วนประกอบการระบุตำแหน่งจะคำนวณท่าทางของหุ่นยนต์ในกรอบแผนที่ใหม่อย่างต่อเนื่องตามการสังเกตจากเซ็นเซอร์ จึงจัดการดริฟต์ แต่ทำให้เกิดการกระโดดแบบแยกจากกันเมื่อมีข้อมูลเซ็นเซอร์ใหม่มาถึง กรอบแผนที่นั้นมีประโยชน์ในฐานะข้อมูลอ้างอิงทั่วโลกในระยะยาว แต่การกระโดดแบบแยกจากกันในตัวประมาณตำแหน่งทำให้กรอบอ้างอิงนี้ไม่เหมาะสำหรับการรับรู้และการดำเนินการในพื้นที่ กรอบพิกัดแผนที่สามารถอ้างอิงได้ทั่วโลกหรืออ้างอิงตำแหน่งเฉพาะของแอปพลิเคชัน สิ่งที่สำคัญที่สุดคือต้องมีการบันทึกการเลือกตำแหน่งอ้างอิงอย่างชัดเจนเพื่อให้ผู้ใช้หลีกเลี่ยงความสับสน ในกรณีที่มีข้อกำหนดเฉพาะของแอปพลิเคชันที่ไม่สามารถตอบสนองข้อกำหนดข้างต้นได้ ก็ยังควรปฏิบัติตามให้มากที่สุด ตัวอย่างแอปพลิเคชันที่ไม่สามารถตอบสนองความต้องการข้างต้นได้ เช่น หุ่นยนต์ที่เริ่มทำงานโดยไม่มีอุปกรณ์อ้างอิงภายนอก เช่น GPS เข็มทิศ หรือเครื่องวัดความสูง แต่หากหุ่นยนต์ยังมีเครื่องวัดความเร่งอยู่ ก็สามารถเริ่มต้นแผนที่ที่ตำแหน่งปัจจุบันโดยให้แกน z ชี้ขึ้นได้ หากหุ่นยนต์มีทิศทางเข็มทิศขณะเริ่มต้น หุ่นยนต์ก็สามารถกำหนดทิศทาง x ตะวันออก และ y เหนือได้และหากหุ่นยนต์มีการประเมินความสูงเมื่อเริ่มต้นใช้งาน หุ่นยนต์ก็สามารถกำหนดค่าเริ่มต้นความสูงที่ MSL ได้ ขอแนะนำให้ปฏิบัติตามอนุสัญญาข้างต้นสำหรับสภาพแวดล้อมที่ไม่มีโครงสร้าง ในสภาพแวดล้อมที่มีโครงสร้าง การจัดตำแหน่งแผนที่ให้สอดคล้องกับสภาพแวดล้อมอาจมีประโยชน์มากกว่า ตัวอย่างของสภาพแวดล้อมที่มีโครงสร้าง เช่น ภายในอาคารสำนักงาน ซึ่งโดยทั่วไปจะเป็นเส้นตรงและมีวิธีการระบุตำแหน่งทั่วโลกที่จำกัด โดยแนะนำให้จัดตำแหน่งแผนที่ให้สอดคล้องกับอาคาร โดยเฉพาะอย่างยิ่งหากทราบเค้าโครงอาคารล่วงหน้า ในทำนองเดียวกัน ในสภาพแวดล้อมภายในอาคาร ขอแนะนำให้จัดตำแหน่งแผนที่ที่ระดับชั้น ในกรณีที่คุณทำงานบนหลายชั้น การมีกรอบพิกัดหลายกรอบสำหรับแต่ละชั้นอาจสมเหตุสมผล หากมีความคลุมเครือ ให้ย้อนกลับไปที่แนวทางปฏิบัติสำหรับสภาพแวดล้อมที่ไม่มีโครงสร้างข้างต้น หรือหากมีความรู้เกี่ยวกับสภาพแวดล้อมล่วงหน้าอย่างจำกัด แนวทางปฏิบัติที่ไม่มีโครงสร้างยังคงใช้ได้ ในสภาพแวดล้อมที่มีโครงสร้าง

Earth เฟรมนี้ได้รับการออกแบบมาเพื่อให้หุ่นยนต์หลายตัวสามารถโต้ตอบกันได้ในเฟรมแผนที่ที่แตกต่างกัน หากแอปพลิเคชันต้องการเพียงแผนที่เดียว ไม่จำเป็นต้องมีเฟรมพิกัดโลก ในกรณีที่ทำงานด้วยแผนที่หลายแผนที่พร้อมกัน จำเป็นต้องปรับแต่งเฟรมแผนที่ odom และ base_link สำหรับหุ่นยนต์แต่ละตัว หากทำงานหุ่นยนต์หลายตัวและเชื่อมโยงข้อมูลระหว่างกัน เฟรม transform_ids จะยังคงเป็นมาตรฐานสำหรับหุ่นยนต์แต่ละตัวได้ หากเขียนเฟรม transform_ids ของหุ่นยนต์ตัวอื่นใหม่ หากอ้างอิงเฟรมแผนที่ทั่วโลก ผู้เผยแพร่จากโลกไปยังแผนที่ที่สามารถเป็นผู้เผยแพร่การแปลงแบบคงที่ได้ มิฉะนั้น การแปลงจากโลกไปยังแผนที่โดยปกติจะต้องคำนวณโดยใช้การประมาณตำแหน่งทั่วโลกปัจจุบันและลบตำแหน่งที่ประมาณไว้ในแผนที่ปัจจุบันเพื่อให้ได้ตำแหน่งที่ประมาณของจุดกำเนิดของแผนที่ ในกรณีที่ไม่มีทราบตำแหน่งสัมบูรณ์ของกรอบแผนที่ในขณะที่เริ่มต้นใช้งาน กรอบแผนที่สามารถแยกออกจากกันได้จนกว่าจะสามารถประเมินการประมาณตำแหน่งทั่วโลกได้อย่างเหมาะสม ซึ่งจะดำเนินการในลักษณะเดียวกับที่หุ่นยนต์สามารถดำเนินการในกรอบแผนที่ได้ก่อนที่จะเริ่มการระบุตำแหน่งในกรอบแผนที่



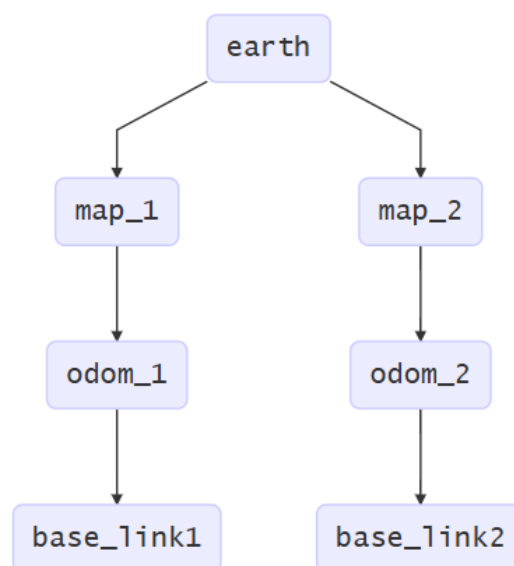
รูปที่ 1.22 การแสดงภาพของโลกที่มีศูนย์กลางเป็นโลกที่คงที่พร้อมกรอบแผนที่เชิงสัมพัทธ์

เราได้เลือกการแสดงผลแบบตันไม้เพื่อแนบเฟรมพิกัดทั้งหมดในระบบหุ่นยนต์เข้าด้วยกัน ดังนั้นเฟรมพิกัดแต่ละเฟรมจะมีเฟรมพิกัดหลักหนึ่งเฟรมและเฟรมพิกัดย่อยจำนวนเท่าใดก็ได้ เฟรมที่อธิบายไว้ดังนี้



รูปที่ 1.23 ภาพแสดงความสัมพันธ์ระหว่างเฟรม

เฟรมแผนที่เป็นพารามิเตอร์ของ odom และ odom เป็นพารามิเตอร์ของ base_link แม้ว่าสัญชาตญาณจะบอกว่าทั้ง map และ odom ควรแนบกับ base_link แต่ก็ไม่สามารถทำได้เนื่องจากเฟรมแต่ละเฟรมสามารถมีพารามิเตอร์ได้เพียงอันเดียว ตัวอย่างของกรอบพิกัดเพิ่มเติมที่มีศักยภาพคือกรอบพิกัดที่แสดงระดับความสูงของความดันสำหรับยานพาหนะบินได้ ระดับความสูงของความดันเป็นการประมาณระดับความสูงโดยอิงจากการประมาณค่าความดันบรรยากาศร่วมกัน ในการใช้งานการบิน สามารถวัดระดับความสูงของความดันได้อย่างแม่นยำโดยใช้เพียงเครื่องวัดความสูงจากบรรยากาศ เครื่องวัดความสูงอาจเคลื่อนที่ตามเวลาเหมือนโอโดเมตริก แต่จะเคลื่อนที่ในแนวตั้งเท่านั้น เพื่อให้มีประโยชน์ กรอบ pressure_altitude อาจถูกแทรกระหว่างกรอบ odom ที่สอดคล้องกับแรงเฉื่อยและกรอบแผนที่ จำเป็นต้องมีตัวประมาณเพิ่มเติมเพื่อประมาณค่าออฟเซตของ pressure_altitude จากแผนที่ แต่กรอบพิกัดพิเศษนี้สามารถรองรับฟังก์ชันการทำงานเพิ่มเติมได้และไม่ทำลายการแยกส่วนที่สรุปไว้ข้างต้น



รูปที่ 1.24 รูป tf ที่มีหุ่นยนต์สองตัวที่ใช้แผนที่ต่างกันเพื่อระบุตำแหน่งและมีกรอบโลกร่วมกัน

ไดอะแกรมด้านบนใช้ ID เฟรมที่แตกต่างกันเพื่อความชัดเจน อย่างไรก็ตาม เพื่อให้สามารถนำกลับมาใช้ใหม่ได้สูงสุด แนะนำให้ใช้ ID เฟรมมาตรฐานบนหุ่นยนต์แต่ละตัวและใช้สคริปต์เพื่อส่งต่อข้อมูลออกจากหุ่นยนต์ เมื่อข้อมูลถูกส่งต่อ ID เฟรมควรได้รับการกำหนดใหม่เพื่อแยกแยะว่า ID เฟรมมาจากหุ่นยนต์ตัวใดและอ้างอิงถึงหุ่นยนต์ตัวใด การแปลงจาก odom เป็น base_link จะถูกคำนวณและออกอากาศโดยแหล่ง odometry หนึ่งแหล่งการแปลงจากแผนที่เป็น base_link คำนวณโดยส่วนประกอบการระบุตำแหน่ง อย่างไรก็ตาม ส่วนประกอบการระบุตำแหน่งจะไม่ออกอากาศการแปลงจากแผนที่เป็น base_link แต่จะรับการแปลงจาก odom เป็น base_link ก่อน และใช้ข้อมูลนี้เพื่อออกอากาศการแปลงจากแผนที่เป็น odom การแปลงจากพื้นโลกเป็นแผนที่จะเผยแพร่และกำหนดค่าแบบคงที่โดยเลือกกรอบแผนที่ หากไม่ได้กำหนดค่าโดยเฉพาะ ตัวเลือกสำรองคือใช้ตำแหน่งเริ่มต้นของยานพาหนะเป็นจุดเริ่มต้นของกรอบแผนที่ หากแผนที่ไม่มีการอ้างอิงทางภูมิศาสตร์เพื่อรองรับการแปลงแบบคงที่แบบง่าย โมดูลการระบุตำแหน่งสามารถทำตามขั้นตอนเดียวกันกับการเผยแพร่ค่าออฟเซตที่ประมาณจากแผนที่ไปยังกรอบ Odom เพื่อเผยแพร่การแปลงจากพื้นโลกไปยังกรอบแผนที่ เมื่อหุ่นยนต์เดินทางเป็นระยะทางไกล คาดว่าหุ่นยนต์จะต้องเปลี่ยนตำแหน่งไปมาระหว่างแผนที่ต่างๆ ในบริบทกลางแจ้ง กรอบพิกัดของแผนที่จะเป็นการประมาณค่าแบบยुकิตของบริเวณใกล้เคียง อย่างไรก็ตาม การประมาณค่าแบบยुकิตจะล้มเหลวในระยะทางที่ไกลขึ้นเนื่องจากความโค้งของโลก ในบริบทในร่ม อาจเป็นการเปลี่ยนตำแหน่งระหว่างอาคารสองหลัง โดยแต่ละหลังมีแผนที่ก่อนหน้านี้ซึ่งคุณกำลังนำทางอยู่ หรือหุ่นยนต์อยู่บนชั้นใหม่ของอาคาร เป็นความรับผิดชอบของผู้มีอำนาจในการกำหนดกรอบตำแหน่งเพื่อกำหนดกรอบ odom ใหม่ให้เหมาะสมเมื่อเคลื่อนที่ไปมาระหว่างแผนที่ การใช้งานทั่วไปในการคำนวณแผนที่เป็นกรอบ odom เป็นผลลัพธ์ของการลบ odom ไปยัง base_link จากแผนที่ที่แก้ไขตำแหน่งไปยัง base_link จะดูสิ่งนี้โดยปริยายเมื่อตัวเลือกกรอบแผนที่เปลี่ยนแปลง เมื่อเปลี่ยนผ่านระหว่างแผนที่ กรอบวัดระยะทางไม่ควรได้รับผลกระทบ นโยบายการเก็บรักษาข้อมูลสำหรับข้อมูลที่รวบรวมในกรอบวัดระยะทางควรได้รับการปรับแต่งเพื่อให้ข้อมูลเก่าหรืออยู่ห่างไกลถูกทิ้งก่อนที่จะข้อผิดพลาดของตำแหน่งที่รวมเข้าด้วยกันจะสะสมมากพอที่จะทำให้ข้อมูลไม่ถูกต้อง นโยบายเหล่านี้อาจแตกต่างกันอย่างมาก ขึ้นอยู่กับคุณภาพของการวัดระยะทางของหุ่นยนต์ ยานยนต์ที่มีล้อที่มีตัวเข้ารหัสความละเอียดสูงซ้ำซ้อนหลายตัวจะมีอัตราการดริฟต์ที่ต่ำกว่ามากและสามารถเก็บข้อมูลไว้ได้นานขึ้นหรือระยะทางไกลกว่าหุ่นยนต์บังคับเลี้ยวแบบมีล้อเลื่อนซึ่งมีข้อเสนอแนะแบบวงเปิดเมื่อเลี้ยวเท่านั้น มีบริบทอื่นๆ ที่จะส่งผลต่อนโยบายการเก็บรักษาข้อมูลที่เหมาะสม เช่น หุ่นยนต์ถูกเคลื่อนย้ายโดยแรงจูงใจภายนอก หรือสมมติฐานของสภาพแวดล้อมแบบคงที่ ตัวอย่างเช่น หุ่นยนต์ในลิฟต์ ซึ่งสภาพแวดล้อมภายนอกมีการเปลี่ยนแปลงระหว่างที่เข้าและออกจากลิฟต์ ปัญหาส่วนใหญ่เหล่านี้มาจากสมมติฐานของสภาพแวดล้อมแบบคงที่ ซึ่งการสังเกตอยู่ในกรอบเฉื่อยเดียวกันกับหุ่นยนต์ ในกรณีดังกล่าว จำเป็นต้องใช้ข้อมูลเชิงความหมาย

เกี่ยวกับสภาพแวดล้อมและวัตถุในนั้นเพื่อจัดการข้อมูลถาวรอย่างถูกต้อง ไม่ว่าจะอย่างไรก็ตาม กรอบเนื้อควรคงอยู่อย่างต่อเนื่องเสมอ หากยานพาหนะเดินทางเป็นระยะทางไกลพอจนทำให้ ระยะทางจากจุดกำเนิดของเฟรม Odom ถึงยานพาหนะเข้าใกล้ความแม่นยำของจุดลอยตัวสูงสุด อาจสังเกตเห็นประสิทธิภาพที่ลดลงสำหรับข้อมูลที่ใช้จุดลอยตัวที่คงอยู่ในเฟรม Odom โดยเฉพาะอย่างยิ่งข้อมูลจุดลอยตัว 32 บิตที่ใช้ในสิ่งต่างๆ เช่น กลุ่มจุด หากพบระยะทางตามลำดับนี้ อาจ จำเป็นต้องรีเซ็ตจุดกำเนิดของเฟรม Odom อย่างเป็นระบบ หากต้องการความแม่นยำในระดับ เซนติเมตร ระยะทางสูงสุดไปยังเฟรม Odom จะอยู่ที่ประมาณ 83 กม. ไม่มีวิธีแก้ปัญหามาตรฐาน สำหรับปัญหานี้ ระบบที่มีปัญหานี้จะต้องดำเนินการแก้ไข วิธีแก้ปัญหาคือเป็นไปได้ ได้แก่ เฟรมพิกัด เพิ่มเติมเพื่อคงข้อมูลสิ่งกีดขวางหรือจัดเก็บข้อมูลสิ่งกีดขวาง หรือการใช้ความแม่นยำที่สูงขึ้น

2.11 Camera Calibration

การปรับเทียบกล้อง (Camera Calibration) เป็นกระบวนการที่มีความสำคัญอย่างยิ่งในการ ประมวลผลภาพเชิงคอมพิวเตอร์และการมองเห็นของหุ่นยนต์ เนื่องจากเป็นขั้นตอนที่ใช้ในการหา ค่าพารามิเตอร์ภายใน (intrinsic parameters), พารามิเตอร์ภายนอก (extrinsic parameters) และ ค่าความบิดเบี้ยวของเลนส์ (distortion parameters) ของกล้อง พารามิเตอร์เหล่านี้ช่วยให้สามารถ แปลงพิกัดของจุดในโลกสามมิติ (3D) ไปยังพิกัดในระนาบสองมิติ (2D) ของเซนเซอร์ภาพได้อย่าง ถูกต้อง อีกทั้งยังช่วยในการแก้ไขความบิดเบี้ยวของเลนส์ซึ่งเกิดจากข้อจำกัดทางกายภาพของการผลิต เลนส์

พารามิเตอร์ภายในของกล้องใช้สำหรับอธิบายคุณสมบัติภายในที่เกี่ยวข้องกับลักษณะทาง เรขาคณิตของการสร้างภาพ ซึ่งได้แก่ ความยาวโฟกัส (focal length) ในแนวแกน x และ y หรือ f_x และ f_y โดยค่าความยาวโฟกัสที่สูงจะส่งผลให้มุมมองแคบลง ขณะที่ค่าที่ต่ำจะให้มุมมองที่กว้างขึ้น อีกส่วนหนึ่งที่สำคัญคือจุดศูนย์กลางของภาพ (optical center) หรือค่าพิกัด c_x และ c_y ซึ่งเป็น ตำแหน่งที่แนวแสงหลักตัดผ่านเซนเซอร์ภาพ ปกติแล้วตำแหน่งนี้อาจมีการเยื้องจากจุดกึ่งกลางของ ภาพประมาณ 20 พิกเซล เนื่องจากความไม่สมบูรณ์ในการจัดวางเลนส์และเซนเซอร์ เช่น การเลื่อน หรือเอียงเล็กน้อย การปรับเทียบที่ถูกต้องจึงช่วยชดเชยข้อบกพร่องดังกล่าวเพื่อคงความแม่นยำใน การทำงานของกล้อง

$$\text{intrinsics properties} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.59)$$

นอกจากนั้น ยังมีพารามิเตอร์ที่เกี่ยวข้องกับความบิดเบี้ยวของเลนส์ซึ่งเรียกว่า distortion coefficients ซึ่งสามารถแบ่งได้เป็นหลายประเภท ได้แก่ ความบิดเบี้ยวเชิงรัศมี (radial distortion)

ซึ่งทำให้ภาพบิดเบี้ยวในลักษณะบวมกลาง (barrel distortion) หรือบับกลาง (pincushion distortion) ความบิดเบี้ยวเชิงสัมผัส (tangential distortion) ที่เกิดจากการเอียงของเลนส์และเซนเซอร์ที่ไม่ได้จัดวางในแนวเดียวกัน รวมถึงการบิดเบี้ยวที่เกิดจากปัจจัยอื่น เช่น thin prism distortion และ skew distortion ซึ่งพบได้บ่อยในเลนส์มุมกว้าง

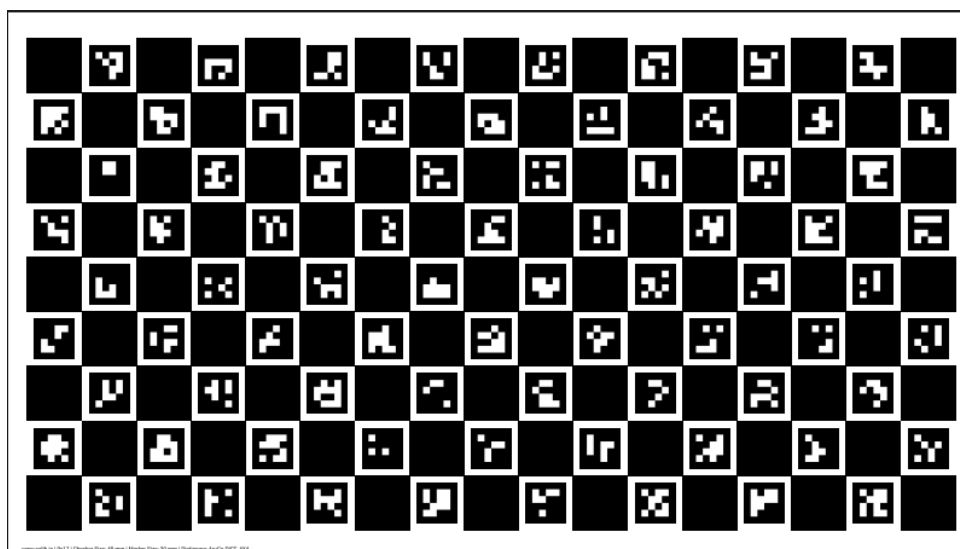
พารามิเตอร์ภายนอกของกล้องทำหน้าที่อธิบายตำแหน่งและทิศทางของกล้องเมื่อเปรียบเทียบกับฉากหรือกล้องตัวอื่น โดยทั่วไปประกอบด้วยค่าการหมุน (rotation) ซึ่งบ่งบอกถึงการหันทิศทางของกล้อง และค่าการเลื่อน (translation) ที่บ่งบอกตำแหน่งของกล้องเมื่อเทียบกับจุดอ้างอิง พารามิเตอร์เหล่านี้มีความสำคัญเป็นพิเศษในระบบที่ต้องใช้กล้องหลายตัว เช่น การมองเห็นแบบสเตอริโอ (stereo vision) ซึ่งต้องอาศัยการปรับภาพจากกล้องแต่ละตัวให้อยู่ในตำแหน่งที่สอดคล้องกันอย่างแม่นยำ

กระบวนการปรับเทียบกล้องประกอบด้วยหลายขั้นตอนเพื่อให้ได้ค่าพารามิเตอร์ที่ถูกต้อง โดยวิธีที่นิยมใช้คือการปรับเทียบโดยใช้มาร์กเกอร์ (marker-based calibration) เช่น มาร์กเกอร์แบบ Charuco ซึ่งเป็นการผสมผสานระหว่างกระดานหมากรุกกับมาร์กเกอร์ Aruco วิธีนี้สามารถวางมาร์กเกอร์ในตำแหน่งต่าง ๆ ทั้งใกล้ ไกล และมุมที่หลากหลายเพื่อให้ครอบคลุมทุกทิศทางของกล้อง และเพิ่มความแม่นยำในการประมาณค่าพารามิเตอร์ นอกจากนี้ การปรับเทียบที่ทำในสภาพแวดล้อมที่มีความนิ่งและเสถียวยังช่วยลดความผิดพลาดและเพิ่มความสามารถในการทำซ้ำได้อย่างสม่ำเสมอ อีกทั้งยังสามารถปรับกระบวนการให้เหมาะสมกับกล้องแต่ละรุ่น เพื่อเพิ่มความน่าเชื่อถือในการนำไปใช้งานจริง

ในการแก้ไขความบิดเบี้ยวของเลนส์ จำเป็นต้องเลือกใช้แบบจำลองการบิดเบี้ยว (distortion models) ที่เหมาะสม ซึ่ง OpenCV ได้ให้กรอบการทำงานมาตรฐานที่สามารถปรับแต่งได้ตามความต้องการ สำหรับกล้องที่มีมุมมองปกติ (Normal Field of View: NFOV) สามารถใช้แบบจำลองภาพเชิงมุมมองมาตรฐาน (standard perspective model) โดยพิจารณาค่าสัมประสิทธิ์ของ radial distortion และ tangential distortion ในกรณีของกล้องมุมกว้าง (Wide Field of View: WFOV) จะใช้แบบจำลองภาพเชิงมุมมองแบบขยาย (extended perspective model) ซึ่งสามารถรองรับพารามิเตอร์การบิดเบี้ยวได้มากถึง 14 ค่า ได้แก่ ค่าสัมประสิทธิ์ของ radial distortion, tangential distortion, thin prism distortion และ tilt parameters ส่วนกรณีเลนส์ฟิชอาย (fisheye lenses) ซึ่งมีความบิดเบี้ยวรุนแรงและมีลักษณะเป็นวงกลม จะใช้แบบจำลองเฉพาะ อย่างไรก็ตามโดยทั่วไปมักไม่เลือกใช้แบบจำลองนี้ เนื่องจากการรองรับเลนส์ฟิชอายใน OpenCV ยังไม่สมบูรณ์เพียงพอต่อการใช้งาน และแบบจำลองมุมมองแบบขยาย (Extended Perspective Model) ก็สามารถตอบโจทย์การใช้งานในกล้องมุมกว้างได้อย่างมีประสิทธิภาพอยู่แล้ว

การใช้พารามิเตอร์ภายใน (intrinsic), พารามิเตอร์ภายนอก (extrinsic) และพารามิเตอร์การบิดเบือนภาพ (distortion parameters) ตามที่อธิบายไว้ในส่วนการปรับเทียบกล้อง (Camera Calibration) การปรับเทียบสเตอริโอ (stereo calibration) จะทำการจัดแนวกล้องทั้งสองให้สัมพันธ์กัน จากพารามิเตอร์เหล่านี้ ระบบจะคำนวณเมทริกซ์การปรับแก้ภาพ (rectification matrices) ซึ่งช่วยให้โหนด StereoDepth สามารถสร้างการประมาณค่าความต่างของภาพสเตอริโอ (stereo disparity) และการประเมินความลึก (depth estimation) ได้อย่างแม่นยำ

แนะนำให้แสดงบอร์ด charuco บนทีวีหรือจอมอนิเตอร์แบบแบนที่มีขนาดใหญ่ โดยหน้าจอที่มีขนาดใหญ่จะดีกว่า เนื่องจากสามารถทำให้มาร์กเกอร์ charuco ปรากฏในภาพได้มากขึ้น ซึ่งมักช่วยปรับปรุงความแม่นยำในการปรับเทียบกล้อง (calibration) ทั้งนี้ ขึ้นอยู่กับขนาดของหน้าจอ เราขอแนะนำให้แสดงบอร์ด charuco ดังกล่าวในโหมดเต็มหน้าจอ (full-screen)



รูปที่ 1.25 บอร์ด charuco

เมื่อแสดงบอร์ด charuco ควรทำให้มาร์กเกอร์และช่องสี่เหลี่ยม (squares) คมชัดและมองเห็นได้อย่างชัดเจน โดยควรคำนึงถึงสิ่งต่อไปนี้ หลีกเลี่ยงการใช้หน้าจอที่สว่างเกินไปหรือมืดเกินไป หลีกเลี่ยงแสงจ้าหรือแสงแดดที่ส่องตรงไปยังหน้าจอและแสดงบอร์ด charuco ในโหมดเต็มหน้าจอ (full-screen)

2.12 robot localization package

ซอฟต์แวร์ robot_localization ในระบบปฏิบัติการหุ่นยนต์ (Robot Operating System: ROS)[13] เป็นหนึ่งในเครื่องมือสำคัญที่ใช้สำหรับการประมาณตำแหน่งและทิศทาง (localization) ของหุ่นยนต์ โดยทำงานผ่านการผสานข้อมูลจากเซนเซอร์หลายชนิดเข้าด้วยกัน (sensor fusion) เช่น

ข้อมูลจากโอโดมเมตรี (odometry), หน่วยวัดความเฉื่อย (IMU) และระบบระบุตำแหน่งบนโลก (GPS) การผสมผสานข้อมูลจากหลายแหล่งนี้ช่วยให้ได้ค่าการประมาณตำแหน่งที่มีความแม่นยำและมีเสถียรภาพมากกว่าการใช้เพียงเซนเซอร์ใดเซนเซอร์หนึ่ง

แพ็คเกจ `robot_localization` รองรับอัลกอริทึมการกรองสถานะ (state estimation filters) ที่ใช้กันอย่างแพร่หลาย ได้แก่ Extended Kalman Filter (EKF) และ Unscented Kalman Filter (UKF) โดยมีโหนดหลักที่เรียกว่า `ekf_localization_node` และ `ukf_localization_node` ซึ่งทั้งสองโหนดใช้ชุดพารามิเตอร์เกือบทั้งหมดร่วมกัน ความแตกต่างอยู่ที่วิธีการประมาณค่าของตัวกรองเท่านั้น เนื่องจากพารามิเตอร์ส่วนใหญ่ถูกออกแบบมาเพื่อควบคุมการประมวลผลข้อมูลก่อนที่จะถูกส่งเข้าสู่ตัวกรองหลัก โหนดเหล่านี้จึงต้องอาศัยการตั้งค่าผ่านไฟล์ configuration และ launch file ซึ่ง `robot_localization` ก็ได้จัดเตรียมไฟล์ต้นแบบ (template) ไว้เพื่ออำนวยความสะดวกแก่ผู้ใช้งาน

ในการผสานข้อมูลจาก GPS แพ็คเกจนี้ได้มีการจัดเตรียมโหนดที่เรียกว่า `navsat_transform_node` ซึ่งมีหน้าที่แปลงข้อมูล GPS ให้อยู่ในเฟรมที่สอดคล้องกับตำแหน่งเริ่มต้นและการวางตัวของหุ่นยนต์ในเฟรมโลกของระบบ สิ่งนี้ช่วยให้การรวมข้อมูล GPS เข้ากับข้อมูลจากเซนเซอร์อื่น ๆ ทำได้ง่ายและเป็นระบบมากขึ้น อย่างไรก็ตาม สิ่งสำคัญที่ผู้ใช้งานต้องตระหนักคือ ข้อมูล GPS มีลักษณะไม่ต่อเนื่องและมักเกิดการกระโดดของค่า (jumps) เป็นระยะ ซึ่งอาจส่งผลกระทบต่อการทำงานของโมดูลการนำทาง (navigation modules)

เพื่อแก้ไขปัญหาดังกล่าว แนวทางหนึ่งคือการใช้โหนด state estimation ของ `robot_localization` มากกว่าหนึ่งตัว โดยโหนดแรกจะใช้สำหรับผสานข้อมูลที่มีความต่อเนื่องสูง เช่น ข้อมูลโอโดมเมตรีและ IMU โดยตั้งค่าพารามิเตอร์ `world_frame` ให้ตรงกับค่า `odom_frame` และใช้เฟรมนี้สำหรับการวางแผนเส้นทางและการควบคุมการเคลื่อนที่ของหุ่นยนต์ ขณะเดียวกันสามารถรันโหนด state estimation อีกตัวเพื่อผสานข้อมูลจากทุกแหล่ง รวมถึง GPS โดยตั้งค่า `world_frame` ของโหนดนี้ให้ตรงกับ `map_frame` ซึ่งช่วยให้ได้ข้อมูลการประมาณตำแหน่งที่สัมพันธ์กับระบบพิกัดแผนที่และยังสามารถนำไปใช้ร่วมกับข้อมูลโหนดแรกได้อย่างมีประสิทธิภาพ

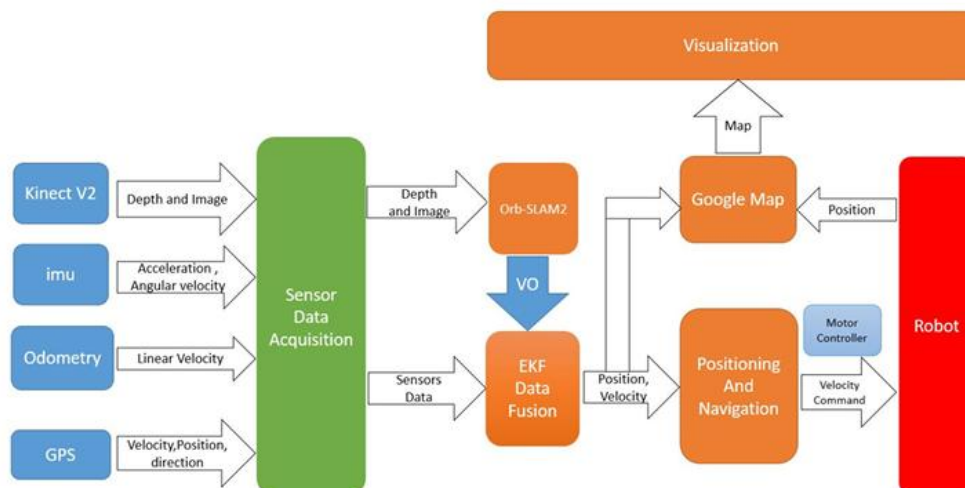
ในบางกรณี ผู้ใช้งานอาจเลือกที่จะผสานข้อมูล GPS เข้ากับโหนด state estimation เพียงตัวเดียว ขึ้นอยู่กับข้อกำหนดและลักษณะการใช้งานจริง แต่ไม่ว่าจะใช้วิธีใด การออกแบบระบบ localization โดยอาศัยแพ็คเกจ `robot_localization` ล้วนมีความยืดหยุ่นสูงและสามารถปรับแต่งเพื่อรองรับการทำงานของหุ่นยนต์ในสภาพแวดล้อมที่แตกต่างกันได้อย่างเหมาะสม

2.13 การระบุตำแหน่งของหุ่นยนต์เคลื่อนที่โดยอาศัยข้อมูลจาก GPS, IMU และระบบวัดระยะด้วยภาพ

หนึ่งในเทคโนโลยีหลักที่ใช้สำหรับการระบุตำแหน่งภายนอกอาคารคือ ระบบกำหนดตำแหน่งบนพื้นโลก (GPS) ถึงแม้ว่าจะสามารถรับข้อมูลตำแหน่งจาก GPS ได้ตลอดทั้งปี แต่กระบวนการส่งสัญญาณนั้นสามารถถูกรบกวนได้ง่าย ซึ่งอาจเกิดขึ้นได้จากหลายสาเหตุ ปัจจัยทั่วไปที่ทำให้เกิดการรบกวน ได้แก่ การที่สัญญาณถูกบดบังโดย อาคารสูง อุโมงค์ หรือถนนยกระดับ รวมถึงสิ่งกีดขวางอื่น ๆ ในหลายพื้นที่ที่อาจขัดขวางการรับสัญญาณของอุปกรณ์ GPS ทำให้สัญญาณที่ได้รับอ่อนลงหรือไม่สามารถรับสัญญาณได้เลย นอกจากนี้ สภาพแวดล้อมโดยรอบยังส่งผลต่อสัญญาณได้อีกด้วย เช่น การหน่วงของสัญญาณในโทรโพสเฟียร์ (troposphere) ซึ่งเป็นอีกสาเหตุหนึ่งที่ทำให้เกิดความคลาดเคลื่อนของสัญญาณ ส่งผลให้เกิดการ ไม่ตรงกันของเวลา (time synchronization error) ระหว่างตัวรับสัญญาณ GPS กับดาวเทียม ซึ่งทำให้ ความแม่นยำของการระบุตำแหน่งลดลง

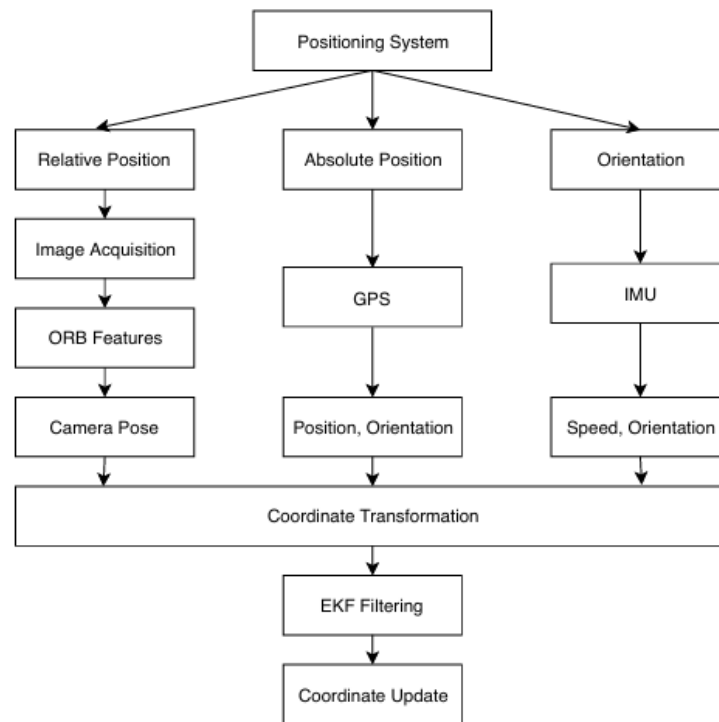
เพื่อปรับปรุงความแม่นยำของการระบุตำแหน่งสำหรับระบบนำทางของหุ่นยนต์เคลื่อนที่ ในงานวิจัย [3] ได้นำเสนอวิธีการระบุตำแหน่งโดยอาศัย ตัวกรองคาลมานแบบขยาย (Extended Kalman Filter: EKF) และนำไปใช้งานบน ระบบปฏิบัติการหุ่นยนต์ (Robot Operating System: ROS) ในขั้นตอน การทำนาย (prediction step) สถานะของหุ่นยนต์จะถูกประมาณค่าด้วยข้อมูลจาก เซนเซอร์แสง (optical sensor) และ หน่วยวัดความเฉื่อย (IMU) จากนั้นสถานะที่คาดการณ์ไว้จะถูกปรับแก้ (corrected) ด้วยข้อมูลที่ได้จาก GPS เพื่อให้เกิดความสัมพันธ์แบบเสริมกัน (complementary relationship) วิธีการนี้สามารถทำให้เราได้รับข้อมูลตำแหน่งที่ แม่นยำยิ่งขึ้น ขณะเดียวกันยังช่วย ลดข้อผิดพลาดสะสม (cumulative error) ที่เกิดขึ้นจากการใช้ข้อมูลของ วิชาวล โอโดเมทรี (visual odometry) และ IMU เมื่อหุ่นยนต์เคลื่อนที่ในสภาพแวดล้อมขนาดใหญ่ได้อีกด้วย

เมื่อกล่าวถึงระบบระบุตำแหน่งภายนอกอาคาร (Outdoor Positioning System) เทคโนโลยีที่เป็นที่รู้จักกันอย่างแพร่หลายคือ ระบบกำหนดตำแหน่งบนพื้นโลก (GPS) อย่างไรก็ตาม ความแม่นยำของ GPS สำหรับผู้ใช้งานทั่วไป อยู่ที่ประมาณ 3 เมตร ซึ่งยังไม่เพียงพอ สำหรับการใช้งานในด้าน การนำทางของหุ่นยนต์ และถึงแม้ว่าจะมี GPS ความแม่นยำสูง ในปัจจุบัน แต่ก็ยังคงมี ราคาสูงเกินไป สำหรับการใช้งานทั่วไปในหลายกรณี นอกจากนี้ สัญญาณ GPS ยังไวต่อสภาพแวดล้อมภายนอก ทำให้เกิดปัญหาในการรับสัญญาณ ดังนั้นจึงจำเป็นต้องอ้างอิงข้อมูลจาก แหล่งข้อมูลอื่นเพิ่มเติม เพื่อเพิ่มความแม่นยำ ระบบระบุตำแหน่งที่ใช้การบูรณาการข้อมูลจากเซนเซอร์หลายตัว (multi-sensor fusion) โดย สถาปัตยกรรมของระบบ แสดงดังใน รูปที่ 2.23



รูปที่ 1.26 สถาปัตยกรรมของระบบ

ในช่วงไม่กี่ปีที่ผ่านมา งานวิจัยด้าน SLAM (Simultaneous Localization and Mapping) มักนิยมใช้ กล้องโมโนคูลาร์ (monocular camera) ที่มีต้นทุนต่ำ หรือ กล้องสเตอริโอ (stereo camera) ที่มีความแม่นยำสูงในการวัดตำแหน่ง มักถูกนำมาใช้ในสภาพแวดล้อมภายนอกอาคาร ในด้านของอัลกอริทึม ORB-SLAM2 เป็นระบบ SLAM ที่อาศัยการตรวจจับคุณลักษณะ (feature-based SLAM system) ซึ่งสามารถใช้งานร่วมกับกล้องทั้งแบบ โมโนคูลาร์ สเตอริโอ และ RGB-D ได้ และมีความสามารถในการระบุตำแหน่งที่มีความแม่นยำสูง สำหรับการพัฒนากล้อง RGB-D นั้นวิธีการวัดความลึก (depth measurement) ได้รับการปรับปรุงจากเดิมที่ใช้วิธี การเข้ารหัสแสง (light coding) มาเป็นเทคนิค Time-of-Flight (TOF) ที่ทันสมัยมากขึ้น เพื่อใช้ในการให้ข้อมูล วิวลีโอโดเมทรี (visual odometry) แก่ระบบ แผนผังการทำงานของระบบระบุตำแหน่งแสดงไว้ใน รูปที่ 2.24 โดยระบบนี้อาศัยข้อมูลจากหลายเซนเซอร์เพื่อระบุตำแหน่งของหุ่นยนต์ได้อย่างแม่นยำมากขึ้น



รูปที่ 1.27 แผนผังการทำงานของระบบระบุตำแหน่งที่ใช้ข้อมูลจากเซนเซอร์ทั้งสามชนิด

เริ่มจาก GPS ซึ่งทำหน้าที่หลักในการให้ข้อมูลเกี่ยวกับ พิกัดตำแหน่งสัมบูรณ์ (absolute position coordinates) และ มุมหัวหุ่นยนต์ (heading angle) ในโลกจริง จากนั้น วิซวลโอดเมทรี (Visual Odometry) จะทำการ ดึงจุดคุณลักษณะ (feature points) จากภาพ แล้วทำการ จับคู่และติดตาม (matching & tracking) จุดเหล่านั้นระหว่างเฟรมภาพต่าง ๆ เพื่อคำนวณ ระยะทางสัมพัทธ์ของการเคลื่อนที่ ของหุ่นยนต์ ขณะเดียวกัน เซนเซอร์ IMU จะให้ข้อมูลเกี่ยวกับ ความเร่ง (acceleration) และ ทิศทางการหมุน (orientation) ของหุ่นยนต์ จากนั้นข้อมูลทั้งหมดที่ได้จากแต่ละเซนเซอร์จะถูก ประมวลผลและแปลงให้อยู่ในระบบพิกัดเดียวกัน (coordinate system) เพื่อให้สามารถใช้งานร่วมกันได้ สุดท้ายแล้ว ข้อมูลจากเซนเซอร์ทั้งหมดจะถูก รวมเข้าด้วยกันโดยใช้ตัวกรองคาลมานแบบขยาย (Extended Kalman Filter: EKF) เพื่อทำการ คำนวณหาตำแหน่งใหม่ของหุ่นยนต์ (new position) ซึ่งมีความแม่นยำและเชื่อถือได้มากขึ้น โดยในช่วงเริ่มต้นของการเคลื่อนที่ Visual Odometry ให้ข้อมูลที่มีความน่าเชื่อถือสูง แต่เมื่อระยะทางที่หุ่นยนต์เคลื่อนที่เพิ่มขึ้น ความคลาดเคลื่อนของข้อมูลจะสะสมมากขึ้น ซึ่งเกิดจากปัจจัยทางสิ่งแวดล้อม ส่งผลให้ข้อมูลระยะทางและทิศทาง เริ่มมีความคลาดเคลื่อน อย่างไรก็ตาม การใช้ข้อมูลจาก GPS ช่วยจำกัดความคลาดเคลื่อนของระยะทางให้อยู่ในช่วงที่สามารถ รักษาความต่อเนื่องของเส้นทางบนแผนที่ได้ และ

ข้อมูลจาก IMU สามารถให้ข้อมูลทิศทางที่น่าเชื่อถือ ดังนั้นจึงได้ทำการ หลอมรวมข้อมูลเสริมจากทั้งสองแหล่งเข้าด้วยกัน ซึ่งช่วยให้ ความแม่นยำของการระบุตำแหน่งของระบบดีขึ้นอย่างมีนัยสำคัญ