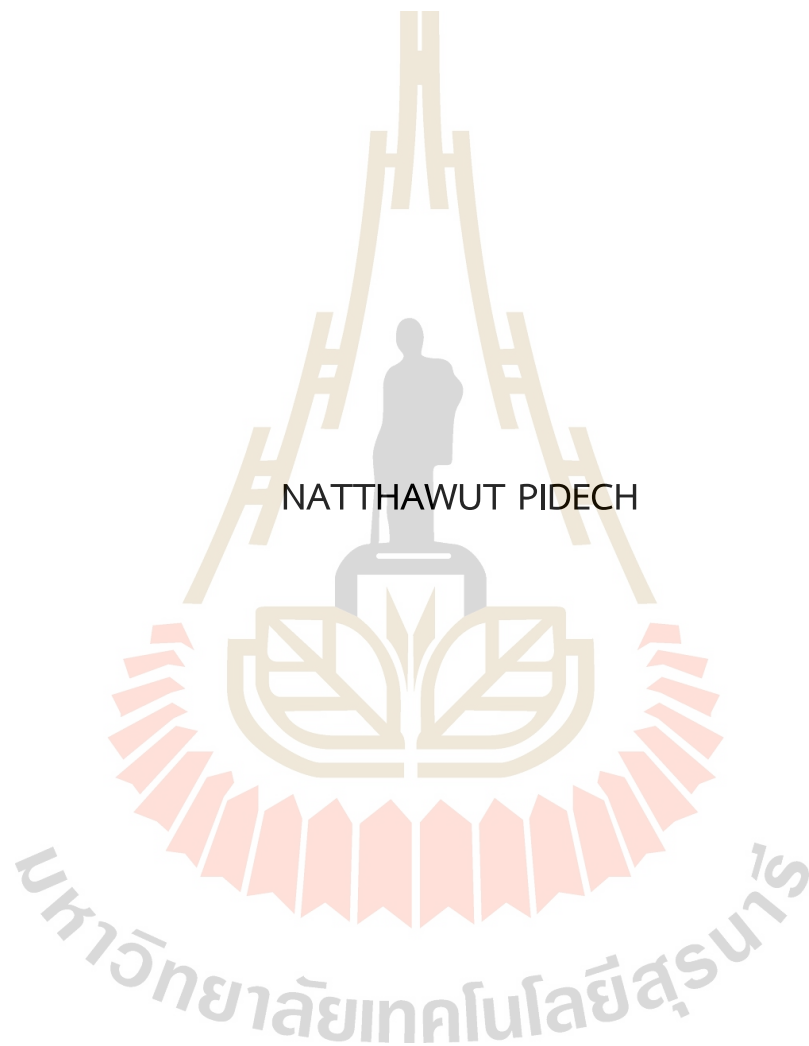


INDOOR SLAM USING LiDAR-BASED LOCALIZATION AND RGB-D
MAPPING ON THE F1TENTH PLATFORM



A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Mechanical and Process System Engineering
Suranaree University of Technology
Academic Year 2025

การสร้างระบบ SLAM ภายในอาคารโดยใช้การระบุตำแหน่งด้วย LiDAR และ
การสร้างแผนที่ด้วยกล้อง RGB-D บนแพลตฟอร์ม F1TENTH



วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมเครื่องกลและระบบกระบวนการ
มหาวิทยาลัยเทคโนโลยีสุรนารี
ปีการศึกษา 2568

INDOOR SLAM USING LIDAR-BASED LOCALIZATION AND RGB-D MAPPING ON THE F1TENTH PLATFORM

Suranaree University of Technology has approved this thesis submitted
in partial fulfillment of the requirements for a Master's Degree.

Thesis Examining Committee



(Asst. Prof. Lt. Dr. Rattawut Vongvit)
Chairperson



(Assoc. Prof. Dr. Atthaphon Ariyarat)
Member (Thesis Advisor)



(Asst. Prof. Dr. Suradet Tantrairatn)
Member



(Asst. Prof. Dr. Sorada Khaengkarn)
Member



(Assoc. Prof. Dr. Yupaporn Ruksakulpiwat)
Acting Vice Rector for Academic Affairs and
Quality Assurance



(Assoc. Prof. Dr. Pornsiri Jongkol)
Dean of Institute of Engineering

ณัฐวุฒิ พิเดช: การสร้างระบบ SLAM ภายในอาคารโดยใช้การระบุตำแหน่งด้วย LiDAR และ
การสร้างแผนที่ด้วยกล้อง RGB-D บนแพลตฟอร์ม F1TENTH

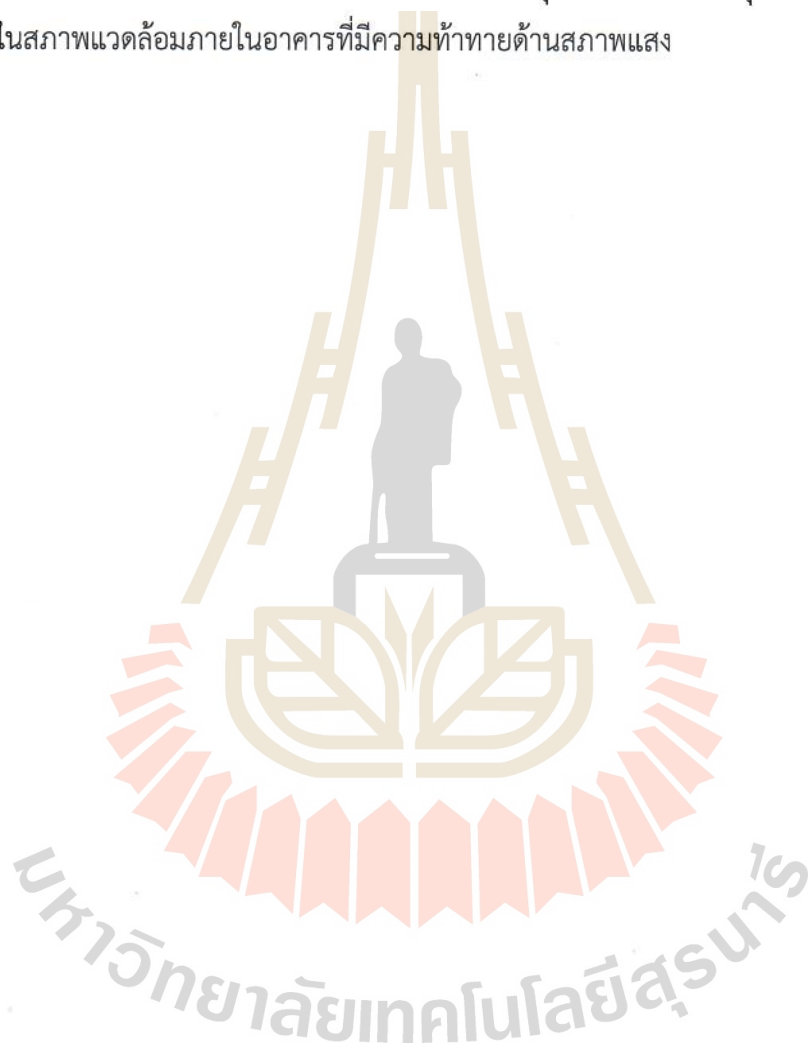
อาจารย์ที่ปรึกษา: รองศาสตราจารย์ ดร.อัฐพล อริยฤทธิ์, 73 หน้า

คำสำคัญ: ระบบระบุตำแหน่งและสร้างแผนที่ SLAM/ หุ่นยนต์เคลื่อนที่/ 2D LiDAR

Simultaneous Localization and Mapping (SLAM) เป็นเทคนิคพื้นฐานที่มีบทบาทสำคัญในงานด้านหุ่นยนต์เคลื่อนที่ โดยช่วยให้หุ่นยนต์สามารถประมาณตำแหน่งของตนเองไปพร้อมกับการสร้างความเข้าใจเกี่ยวกับสภาพแวดล้อมรอบตัว ระบบ SLAM ที่อาศัยการมองเห็น เช่น Visual SLAM และ RGB-D SLAM ได้รับความนิยมอย่างแพร่หลาย เนื่องจากสามารถให้ข้อมูลเชิงโครงสร้างของสภาพแวดล้อมได้อย่างละเอียด อย่างไรก็ตาม ประสิทธิภาพของระบบเหล่านี้ขึ้นอยู่กับสภาพแสงเป็นอย่างมาก ในสภาพแวดล้อมที่มีแสงน้อย แสงสะท้อนสูง หรือคุณภาพการมองเห็นต่ำ ระบบ SLAM ที่ใช้กล้องมักไม่สามารถดึงคุณลักษณะภาพที่เชื่อถือได้ ส่งผลให้เกิดความคลาดเคลื่อนในการระบุตำแหน่ง หรืออาจไม่สามารถทำการระบุตำแหน่งได้อย่างต่อเนื่อง ข้อจำกัดดังกล่าวส่งผลต่อความสามารถในการใช้งานหุ่นยนต์ในระยะยาวภายใต้สภาพแวดล้อมจริง วัตถุประสงค์ของวิทยานิพนธ์ฉบับนี้คือการศึกษาศักยภาพการใช้ SLAM ที่อาศัย LiDAR เป็นแกนหลักในการระบุตำแหน่งที่มีความทนทานต่อสภาพแสงที่ไม่เหมาะสม ขณะเดียวกันยังคงใช้กล้อง RGB-D สำหรับการสร้างข้อมูลเชิงพื้นที่ งานวิจัยนี้มีเป้าหมายเพื่อแสดงให้เห็นว่าสามารถรักษาความแม่นยำในการระบุตำแหน่งได้ด้วย LiDAR ในกรณีที่ระบบ SLAM ที่ใช้กล้องมีประสิทธิภาพลดลง ซึ่งช่วยให้การสร้างข้อมูลเชิงพื้นที่สามารถดำเนินต่อไปได้อย่างสม่ำเสมอ แม้ในสถานะที่ข้อมูลภาพมีข้อจำกัด แนวทางที่นำเสนอถูกพัฒนาและทดสอบบนแพลตฟอร์มหุ่นยนต์เคลื่อนที่อัตโนมัติ F1TENTH โดยใช้เซนเซอร์ LiDAR แบบสองมิติในการดำเนินการ SLAM เพื่อการระบุตำแหน่งอย่างต่อเนื่องและมีเสถียรภาพ พร้อมทั้งติดตั้งกล้อง RGB-D เพื่อเก็บข้อมูลความลึกและสี แทนที่จะใช้ข้อมูลจากกล้องเพื่อการระบุตำแหน่งโดยตรง ข้อมูลความลึกจะถูกแปลงไปยังกรอบพิกัดอ้างอิงระดับโลกโดยอาศัยตำแหน่งที่ได้จาก LiDAR SLAM ระบบทั้งหมดถูกพัฒนาภายใต้สภาพแวดล้อมของ ROS 2 ซึ่งทำหน้าที่จัดการการประสานข้อมูลจากเซนเซอร์ การแปลงกรอบพิกัด และกระบวนการประมวลผลข้อมูล การทดลองถูกดำเนินการในสภาพแวดล้อมที่มีข้อจำกัดด้านแสงเพื่อประเมินพฤติกรรมและประสิทธิภาพของระบบ

ผลการทดลองแสดงให้เห็นว่า SLAM ที่ใช้ LiDAR สามารถรักษาความแม่นยำในการระบุตำแหน่งได้อย่างต่อเนื่องในสภาพแวดล้อมที่การรับรู้ด้วยภาพมีข้อจำกัด ภายใต้สถานะแสงที่ไม่เพียงพอหรือไม่เสถียร ระบบ SLAM ที่อาศัยกล้องมีแนวโน้มเกิดความคลาดเคลื่อนของเส้นทางการเคลื่อนที่ และในบางกรณีไม่สามารถดำเนินการระบุตำแหน่งต่อไปได้ เนื่องจากไม่สามารถดึงคุณลักษณะภาพที่เชื่อถือได้ ในทางตรงกันข้าม ระบบที่นำเสนอสามารถคงความแม่นยำในการ

ประมาณตำแหน่งได้โดยอาศัยข้อมูลจาก LiDAR ทำให้การระบุตำแหน่งดำเนินต่อไปได้โดยไม่สะดุด นอกจากนี้ การใช้ตำแหน่งที่ได้จาก LiDAR เพื่อแปลงข้อมูลจากกล้อง RGB-D ไปยังกรอบพิกัดระดับโลก ช่วยให้การสร้างข้อมูลเชิงพื้นที่ดำเนินไปได้อย่างต่อเนื่อง แม้ข้อมูลภาพจะถูกจำกัด เมื่อสภาพแสงกลับสู่ภาวะปกติ ระบบสามารถกลับมาสร้างข้อมูล point cloud แบบหนาแน่นได้โดยไม่ต้องเริ่มต้นระบบใหม่ ผลลัพธ์ดังกล่าวยืนยันว่า การแยกกระบวนการระบุตำแหน่งออกจากการรับรู้ด้วยภาพช่วยเพิ่มความทนทานของระบบ และสนับสนุนการทำงานของหุ่นยนต์อัตโนมัติในระยะยาวภายในสภาพแวดล้อมภายในอาคารที่มีความท้าทายด้านสภาพแสง



สาขาวิชา วิศวกรรมเครื่องกล

ปีการศึกษา 2568

ลายมือชื่อนักศึกษา สุวิทย์ พินิจ

ลายมือชื่ออาจารย์ที่ปรึกษา สุวิทย์ พินิจ

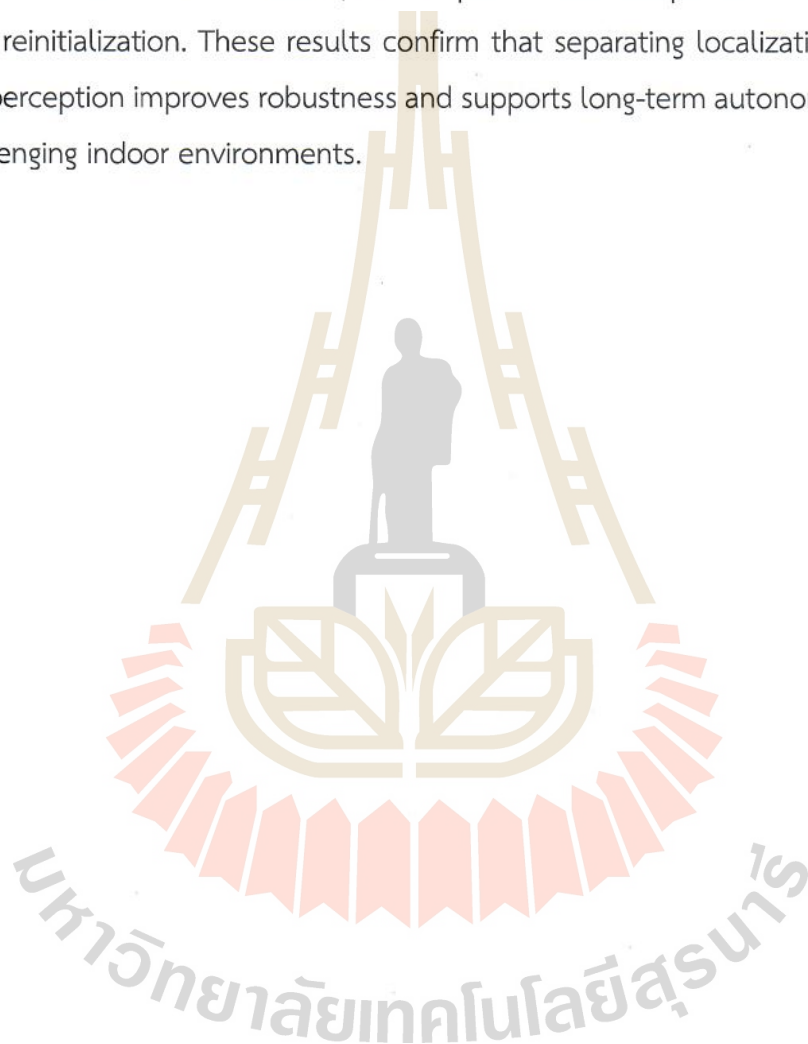
NATTHAWUT PIDECH: INDOOR SLAM USING LiDAR-BASED LOCALIZATION AND
RGB-D MAPPING ON THE F1TENTH PLATFORM

ADVISOR: ASSOC. PROF. ATTHAPHON ARIYARIT, Ph.D., 73 PP.

Keyword: Simultaneous Localization and Mapping (SLAM)/ Mobile robotics/ 2D LiDAR

Simultaneous Localization and Mapping (SLAM) is a fundamental technique in mobile robotics, enabling a robot to estimate its position while understanding the surrounding environment. Vision-based and RGB-D camera-based SLAM systems are widely used due to their ability to provide rich environmental information. However, their performance strongly depends on lighting conditions. Under low-light, high-glare, or visually degraded environments, camera-based SLAM often fails to extract reliable features, leading to localization drift or complete localization failure. This limitation restricts the long-term operation of robots in real-world environments. The objective of this thesis is to investigate the use of LiDAR-based SLAM as a robust localization backbone under poor lighting conditions, while still utilizing an RGB-D camera for point cloud generation. The study aims to demonstrate that reliable localization can be maintained using LiDAR when camera-based SLAM becomes unreliable, allowing consistent spatial data generation despite degraded visual conditions. The proposed approach is implemented on an F1TENTH autonomous ground vehicle platform, A 2D LiDAR sensor is used to perform LiDAR-based SLAM for continuous and robust localization. An RGB-D camera is mounted on the robot to capture depth and color data. Instead of relying on the camera for localization, the depth data is transformed into a global reference frame using the pose estimated from LiDAR-based SLAM. The system is developed using ROS 2, where sensor synchronization, coordinate transformations, and data processing are managed within a unified framework. Experiments are conducted in environments with degraded lighting to evaluate system behavior. The experimental findings demonstrate that LiDAR-based SLAM maintains consistent localization performance in environments where visual sensing is degraded. During conditions of insufficient or unstable lighting, the camera-based SLAM approach exhibited increased trajectory drift and, in some cases, failed to continue localization

due to the inability to extract reliable visual features. In contrast, the proposed system sustained accurate pose estimation by relying on LiDAR data, enabling uninterrupted localization throughout the experiment. By using the LiDAR-derived pose to project RGB-D measurements into the global coordinate frame, spatial data generation remained continuous even when visual information was compromised. Once normal lighting conditions were restored, dense point cloud acquisition resumed without system reinitialization. These results confirm that separating localization from vision-based perception improves robustness and supports long-term autonomous operation in challenging indoor environments.



School of Mechanical Engineering

Academic Year 2025

Student's Signature วิภาดา นิล

Advisor's Signature อ.ดร. อรุณ

ACKNOWLEDGEMENT

The thesis entitled “Robust Indoor SLAM Using LiDAR-Based Localization and RGB-D Mapping on the F1 TENTH Platform” has been successfully completed. The author would like to express sincere gratitude to the individuals and organizations who provided guidance, advice, and continuous support throughout the research process. First and foremost, the author would like to express deepest appreciation to Assoc. Prof. Dr. Atthaphon Ariyarit, the thesis advisor, for granting the opportunity to pursue graduate studies, as well as for his valuable guidance, constructive advice, and constant support in solving problems encountered during the research.

The author would also like to thank all former instructors who imparted fundamental and advanced knowledge, which enabled the researcher to build upon these foundations and successfully complete this research.

Finally, sincere thanks are extended to senior and junior colleagues whose names are not individually mentioned, for both direct and indirect assistance that contributed to the successful completion of this thesis.

Natthawut Pidech

TABLE OF CONTENTS

	Page
ABSTRACT (THAI).....	I
ABSTRACT (ENGLISH).....	III
ACKNOWLEDGEMENT.....	IV
TABLE OF CONTENTS	VI
LIST OF TABLES	IX
LIST OF FIGURES	X
CHAPTER	
I INTRODUCTION	1
1.1 BACKGROUND.....	1
1.2 OBJECTIVES.....	3
1.3 SCOPE.....	3
1.4 EXPECTED BENEFITS.....	4
II LITERATURE REVIEWS.....	5
2.1 LITERATURE REVIEWS	5
2.1.1 Literature Review: Effects of Lighting on Depth Camera Performance.....	5
2.1.2 Drone For Inspecting Inside Building With Object Avoidance And Mapping System (Pawarut Karaked, 2023).....	5
2.1.3 LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping (Tixiao ,S ,. 2020).....	7
2.1.4 A survey on vision-based UAV navigation (Lu, Xue, Xia, & Zhang).....	8

TABLE OF CONTENTS (Continued)

	Page
2.1.5 Development of Autonomous Electric Golf Cart System for Passenger Transportation Between Building (Wongsathon Angkhem, 2023)	9
2.1.6 A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation (Mourikis, A., 2007)	9
2.2 ROBOT	11
2.2.1 F1TNEH	11
2.3 VEDDER ELECTRONIC SPEED CONTROLLER VESC	12
2.4 LOW LEVEL	14
2.5 ROBOT OPERATING SYSTEM ROS	15
2.6 SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM)	18
2.6.1 Coordinate frame conventions.....	19
2.6.2 Simultaneous Localization and Mapping (SLAM).....	21
2.6.3 LiDAR SLAM.....	24
2.6.4 Visual SLAM	25
III RESERCH METHODOLOGY	28
3.1 METHODS OF DATA COLLECTION.....	28
3.2 PROCEDURE.....	29
3.2.1 Study and Literature review.....	29
3.2.2 Configure Jetson and Peripherals	30
3.2.3 Test all sensors.....	31
3.2.4 SLAM testing and configuration parameter.....	31
3.2.5 Merge system	31
3.3 THE OVERALL SYSTEM OPERATION.....	32
3.3.1 Computation Unit: NVIDIA Jetson NX.....	32

TABLE OF CONTENTS (Continued)

	Page
3.3.2 Hardware Configuration.....	33
3.3.3 Software Architecture.....	36
3.3.4 Integration of LiDAR Localization and Depth Mapping.....	38
IV RESULTS AND DISCUSSION.....	40
4.1 RESULTS AND DISCUSSION.....	40
4.1.1 Comparing LiDAR SLAM and RTAB-Map SLAM Against a Ground-Truth Path.....	42
4.1.2 Loop Closure Experimental Results	53
4.1.3 Robustness Under Camera Failure During Operation	54
V CONCLUSION AND RECOMMENDATION	59
5.1 CONCLUSION.....	59
5.2 RECOMMENDATION.....	60
REFERENCES	61
APPENDIX	82
BIOGRAPHY	73

LIST OF TABLES

Table	Page
2.1 Comparison of the advantages and disadvantages of each method	10
4.1 Experimental Localization Data and Absolute Error Measurements test 1	44
4.2 Localization Error Summary Table test 1	44
4.3 Experimental Localization Data and Absolute Error Measurements test 2.....	46
4.4 Localization Error Summary Table test 2	46
4.5 Experimental Localization Data and Absolute Error Measurements test 3	48
4.6 Localization Error Summary Table test 3	48
4.7 Experimental Localization Data and Absolute Error Measurements test 4	49
4.8 Localization Error Summary Table test 4	50
4.9 Experimental Localization Data and Absolute Error Measurements test 5	51
4.10 Localization Error Summary Table test 5	51
4.11 Comparative Error Analysis of RTAB-Map and Odom_SLAM over Five test Runs	52

LIST OF FIGURES

Figure	Page
2.1 The drone's movement path during the positioning system test in case the stereo camera failed to function (Pawrut K., 2023).	6
2.2 Drone trajectory in positioning system test in case LIDAR fails to function. (Pawrut K., 2023).	7
2.3 Mapping results (Xu, X, 2022).	8
2.4 An example of creating an outdoor map using 3D LiDAR (Wongsathon A., 2023)	9
2.5 illustrates the experimental F1TENTH autonomous vehicle platform used in this study. The system integrates a 2D LiDAR sensor and an RGB-D depth camera for perception, an NVIDIA Jetson NX for onboard computation, and a VESC motor controller for vehicle actuation.	11
2.6 ROS2 shows a ROS 2 transform message representing the robot pose relative to the odometry frame, including position and orientation information used in the localization process.	12
2.7 Wheel Mobile Robot Model four-wheel mobile (Zunaidi, I, 2006)	13
2.8 The ROS Master Source (ROS, 2024).	16
2.9 RViz2 terminal	18
2.10 Illustration of multiple reference frames used in a robotic system, showing the relationships between different coordinate transformations within the robot structure (Foote, 2013).	19

LIST OF FIGURES (Continued)

Figure	Page
2.11 Diagram illustrating the coordinate frame hierarchy in a mobile robot, including the map, odometry, base footprint, base link, stabilized base, and laser sensor frames, and their relative transformations within the ROS tf framework (ROS) (M. Quigley., 2009).	20
2.12 illustrates the overall process of Full SLAM, (thrun, 2007)	23
2.13 Example of LiDAR SLAM usage	24
2.14 Illustration of the angles used for computing the distance (A. Zaarane, 2020)..	26
2.15 Example of two cameras mounted as a stereo camera (A. Zaarane, 2020).....	26
3.1 Procedure	29
3.2 System architecture of the F1TENTH platform, illustrating the data flow and control connections between the NVIDIA Jetson Xavier NX, LiDAR sensor, VESC motor and steering controller, onboard actuators, and the pit/host computer via SSH and HDMI interfaces.	30
3.3 The Overall system operation	32
3.4 NVIDIA Jetson NX	33
3.5 VESC.....	34
3.6 2D LiDAR	35
3.7 D435i.....	35
3.8 Coordinate frame hierarchy of the proposed system visualized using the ROS TF tree. The diagram illustrates the transformation relationships between the global map frame, odometry frame, robot base frame, LiDAR frame, and RGB-D camera frames.	36
3.9 Simplified ROS TF frame structure showing the transformation chain from the global map frame to odom, base_link, and the LiDAR (laser) frame.....	37
3.10 Loosely-coupled sensor fusion concept.....	39
4.1 Comparing LiDAR SLAM and RTAB-Map Against a Ground-Truth Path test.....	41

LIST OF FIGURES (Continued)

Figure	Page
4.2 Experimental Environment Setup for Robustness Evaluation Experiment 2	41
4.3 Comparing LiDAR SLAM and RTAB-Map Against a Ground-Truth Path test 1	43
4.4 Comparing LiDAR SLAM and RTAB-Map Against a Ground-Truth Path test 2	45
4.5 Comparing LiDAR SLAM and RTAB-Map Against a Ground-Truth Path test 3	47
4.6 Comparing LiDAR SLAM and RTAB-Map Against a Ground-Truth Path test 4	49
4.7 Comparing LiDAR SLAM and RTAB-Map Against a Ground-Truth Path test 5	50
4.8 presents the loop closure performance of the SLAM system during a closed loop.	54
4.9 Robustness Under Camera Failure During Operation	55
4.10 Robustness Under Camera Failure During Operation During the Moment of camera failure	56
4.11 The system back on, the propose system resumes generating point cloud data.	57

CHAPTER I

INTRODUCTION

1.1 Background

Mobile robots increasingly rely on three-dimensional (3D) perception to understand and interact with their environments. One of the most common representations used for 3D perception is the point cloud, which provides a spatial description of surrounding objects and structures by representing them as collections of points in three-dimensional space. Point clouds are widely used in robotics applications such as environment understanding, obstacle detection, scene analysis, and navigation support. The quality and reliability of point cloud data directly affect the performance and robustness of higher-level robotic systems.

RGB-D cameras have become a popular sensing modality for point cloud generation due to their ability to provide both color (RGB) and depth information at relatively low cost. RGB-D sensors enable dense point cloud generation with rich visual information. These sensors are widely adopted in indoor robotics, and autonomous systems. However, despite their advantages, RGB-D cameras suffer from well-known limitations that significantly degrade depth data quality under certain environmental conditions.

Depth cameras are highly sensitive to factors such as lighting variation, reflective or transparent surfaces, outdoor sunlight interference, motion blur, and sensor noise. In low-texture environments, depth estimation becomes unstable, while in high-glare or reflective scenes, depth measurements may be missing or severely distorted. These issues are especially pronounced in dynamic or challenging environments, where the depth camera may fail to produce consistent and reliable depth information.

In contrast, LiDAR sensors provide accurate and robust distance measurements independent of ambient lighting conditions. LiDAR systems actively emit laser pulses and measure the time of flight to compute distances, allowing them to operate reliably

in low-light, high-glare environments where RGB-D sensors struggle. While LiDAR sensors typically produce sparser point clouds compared to RGB-D cameras, their geometric accuracy and consistency make them well suited for localization and pose estimation tasks. Consequently, LiDAR-based odometry and localization systems are often more reliable than vision-based alternatives in adverse sensing conditions.

However, LiDAR sensors alone may not always provide sufficiently dense spatial information for detailed scene representation. Their limited vertical resolution and angular sampling can result in incomplete or sparse point clouds. As a result, relying solely on LiDAR data may be insufficient for applications that require dense point cloud representations. This limitation motivates the integration of RGB-D and LiDAR sensors, allowing each sensor to compensate for the weaknesses of the other.

Sensor fusion approaches that combine RGB-D cameras and LiDAR sensors have been explored to improve robustness in robotic perception systems. In such systems, LiDAR data is often used to provide reliable motion estimation or pose information, while RGB-D sensors contribute dense depth measurements for point cloud generation. By decoupling pose estimation from depth sensing, the system can remain operational even when the RGB-D camera experiences degraded depth quality.

Simultaneous Localization and Mapping (SLAM) is a foundational capability in mobile robotics that enables a robot to estimate its own pose while incrementally building a representation of the surrounding environment. Traditionally, SLAM systems aim to produce a globally consistent map while simultaneously tracking the robot's trajectory within that map. Over the past two decades, SLAM has evolved from purely geometric approaches using range sensors to more complex multi-sensor systems integrating vision, inertial measurements, and LiDAR data.

LiDAR-based SLAM provides a more robust alternative for localization, as it relies on geometric structure rather than visual appearance. LiDAR sensors are less sensitive to lighting variations and can maintain reliable pose estimation even when RGB-D sensors fail to deliver accurate depth information. As a result, LiDAR-based SLAM is frequently used as a dependable source of localization in challenging environments.

LiDAR-based SLAM can provide stable and accurate localization even when RGB-D sensing conditions are poor. By using LiDAR-based SLAM only for localization, the robot can still estimate its pose correctly while collecting depth data from an RGB-

D camera. This allows point clouds to be generated in a consistent coordinate frame, even when the quality of depth measurements is reduced. Therefore, combining LiDAR-based localization with RGB-D sensing is a practical approach to maintain point cloud generation under challenging depth sensor conditions

1.2 Objectives

1.2.1 To develop an indoor localization system using a loosely-coupled multi-sensor SLAM framework that integrates 2D LiDAR and depth camera data

1.2.2 To compare the resilience and localization performance of the proposed system with the single base sensor system

1.3 Scope

1.3.1 The proposed system consider only planer (2D) localization and does not estimate vertical (Z-axis)

1.3.2 Outdoor operation, GNSS integration, and large-scale environments are outside the scope

1.3.3 The system uses a 2D LiDAR for localization and RGB-D camera for dense point cloud generation other sensors such as stereo cameras, 3D LiDAR, cameras are not considered

1.3.4 The system is evaluated under degraded lighting conditions that negatively affect RGB-D sensing

1.3.5 The generated point cloud maps are intended for visualization and analysis purposes only

1.3.6 Path planning, navigation, and autonomous decision-making using the generated maps are not within the scope

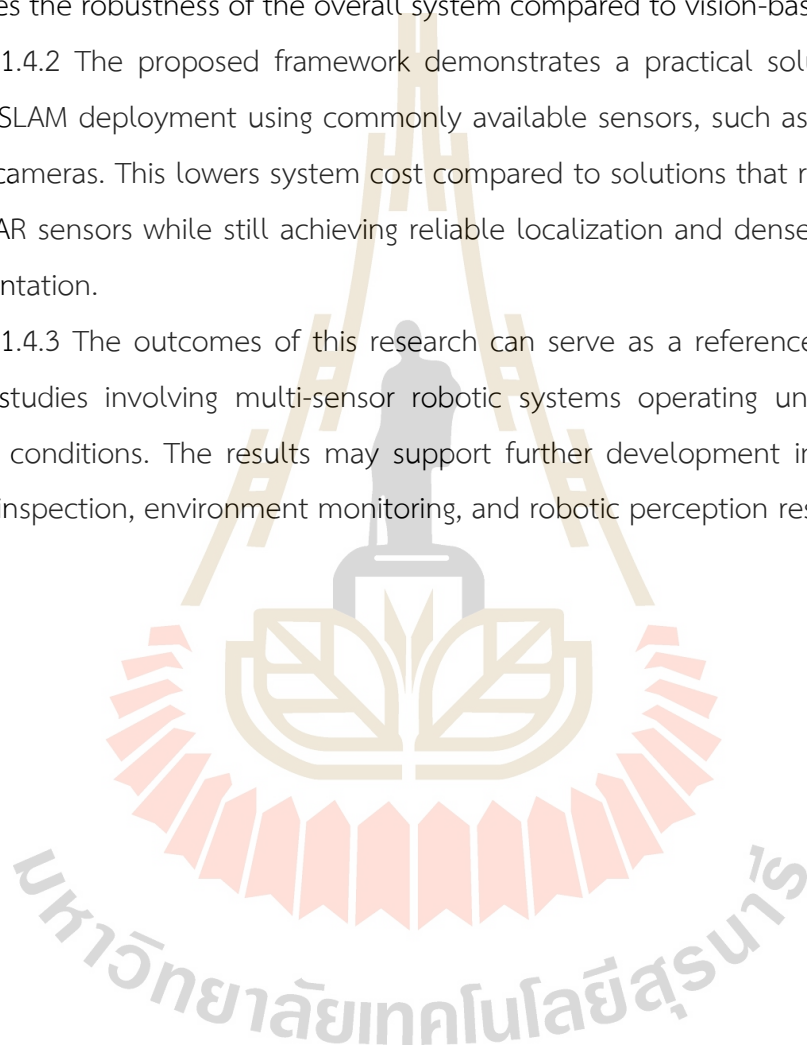
1.3.7 The evaluation focuses on localization robustness, consistency of point cloud generation.

1.4 Expected benefits

1.4.1 The proposed system is expected to improve localization reliability in indoor environments where visual sensing performance is degraded. By relying on LiDAR-based SLAM for localization, the system can maintain stable pose estimation even under poor lighting conditions that negatively affect RGB-D cameras. This increases the robustness of the overall system compared to vision-based SLAM alone.

1.4.2 The proposed framework demonstrates a practical solution for multi-sensor SLAM deployment using commonly available sensors, such as 2D LiDARs and RGB-D cameras. This lowers system cost compared to solutions that rely on high-end 3D LiDAR sensors while still achieving reliable localization and dense environmental representation.

1.4.3 The outcomes of this research can serve as a reference framework for future studies involving multi-sensor robotic systems operating under challenging sensing conditions. The results may support further development in areas such as indoor inspection, environment monitoring, and robotic perception research.



CHAPTER II

LITERATURE REVIEWS

2.1 Literature reviews

2.1.1 Literature Review: Effects of Lighting on Depth Camera Performance

Depth cameras are commonly employed in indoor robotic and autonomous applications, including environment perception, spatial understanding, and navigation-related tasks. Despite their widespread use, the effectiveness of depth sensing is strongly influenced by surrounding environmental factors, especially lighting conditions. Previous research has shown that variations in illumination can significantly affect depth measurement quality, leading to increased noise, reduced accuracy, and unstable system performance. (J. Gené-Mola, 2020) (G. Halmetschlager, 2018) (P. Chen, X, 2025)

2.1.2 Drone For Inspecting Inside Building With Object Avoidance And Mapping System (Pawarut Karaked, 2023)

This research focuses on evaluating an integrated indoor drone system that combines multiple SLAM methods, EKF-based sensor fusion, mapping, and autonomous navigation with obstacle avoidance. The positioning system remains functional even when partial SLAM failure occurs, confirming the effectiveness of the Extended Kalman Filter in filtering abnormal data. The mapping system produces maps with dimensions consistent with the real environment, indicating acceptable accuracy. Additionally, the autonomous flight and obstacle avoidance system successfully calculates and follows flight paths while maintaining safe navigation.

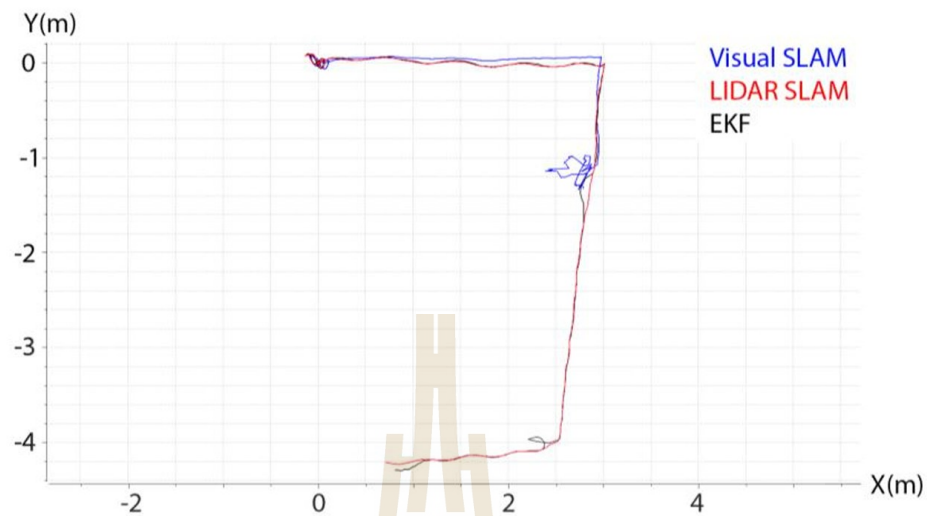


Figure 2.1 The drone's movement path during the positioning system test in case the stereo camera failed to function (Pawrut K., 2023).

In the event that the stereo camera fails to function during testing, a drone will be controlled in a darkened room with controlled lighting. At the start of the test, the room will have sufficient light for the stereo camera to operate. Afterward, all lights in the room will be turned off, making it completely dark and preventing the stereo camera from detecting its surroundings.

The result was that, initially with sufficient lighting, all positioning system, including the drone, functioned correctly. However, when the light was turned off, inaccuracies occurred in the Visual SLAM data. Meanwhile, data from the LiDAR SLAM and EKF remained accurate as in normal circumstances, and the drone itself continued to operate normally.

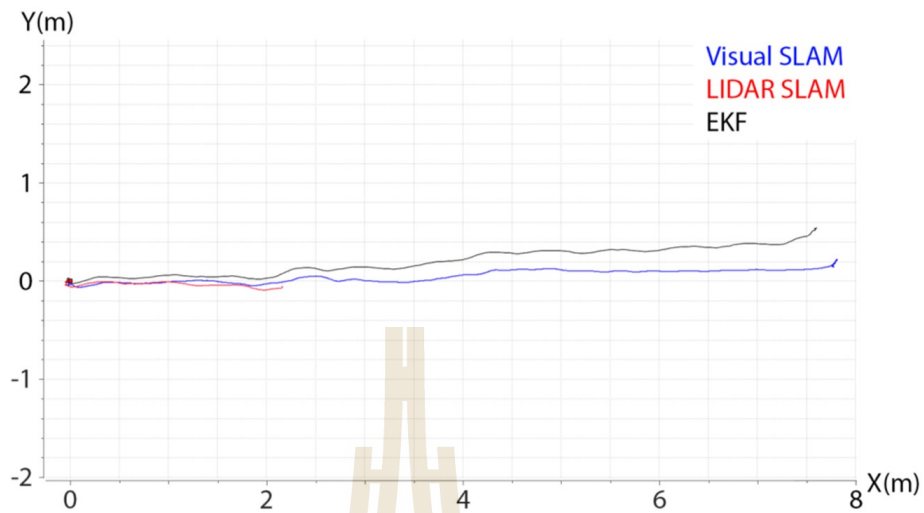


Figure 2.2 Drone trajectory in positioning system test in case LIDAR fails to function. (Pawrut K., 2023).

In the event of LIDAR failure, a test was conducted controlling a drone in a large open room where LIDAR cannot detect all sides simultaneously. In areas where LIDAR could not detect anything at the start of the test, the drone was initially positioned near a wall to ensure the LIDAR SLAM functioned normally. The drone was then controlled to an area where LIDAR could not detect anything, simulating LIDAR SLAM failure.

The result showed that, at the beginning, when LIDAR could still see the wall, the entire positioning system, including the drone, functioned correctly. When simulating LIDAR SLAM damage, Visual SLAM and EKF also worked, even without relying on LIDAR data. A system called Infrared Local Positioning System (IR-LPS) was introduced. This system uses infrared to determine the drone's position, replacing GPS.

2.1.3 LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping (Tixiao ,S ,. 2020)

proposed LIO-SAM, a tightly-coupled LiDAR-inertial odometry framework based on factor graph optimization. The system integrates LiDAR measurements and inertial data within a unified smoothing and mapping architecture to achieve high-accuracy state estimation in real time.

LIO-SAM is built upon a factor graph formulation using incremental smoothing techniques. The framework combines. LiDAR scan matching constraints, IMU pre-integration factors, Loop closure constraints and GPS factors (optional, when available)

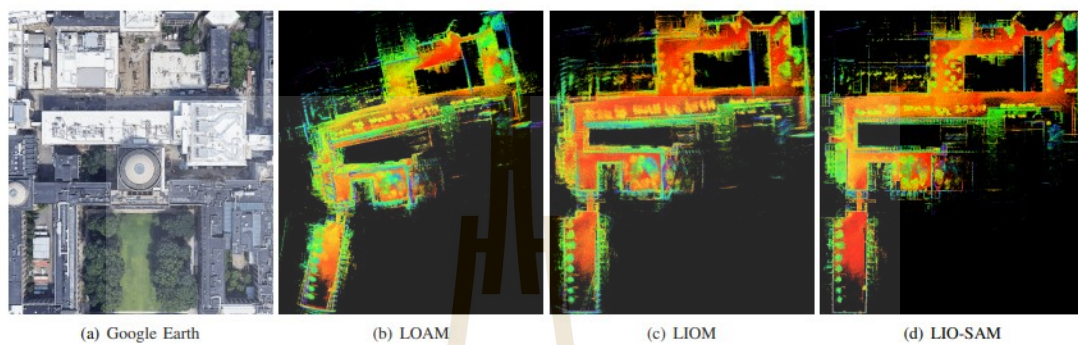


Figure 2.3 Mapping results (Xu, X, 2022).

Mapping results of LOAM, LIOM, and LIO-SAM using the Walking dataset. The map of LOAM in (b) diverges multiple times when aggressive rotation is encountered. LIOM outperforms LOAM. However, its map shows numerous blurry structures due to inaccurate point cloud registration. LIO-SAM produces a map that is consistent with the Google Earth imagery, without using GPS.

2.1.4 A survey on vision-based UAV navigation (Lu, Xue, Xia, & Zhang)

This research addresses the challenges of indoor autonomous drone operation using a method called Simultaneous Localization and Mapping (SLAM), integrated with an Obstacle Avoidance system. SLAM is a process that identifies its own position and maps its environment using distance data from proximity sensors. This research recognizes that both types of data are applicable in solving indoor autonomous drone problems. The localization component provides the drone's current location, while the mapping component monitors and records the environment. This information is then used in conjunction with the Obstacle Avoidance system, which guides the drone safely to its target location.

2.1.5 Development of Autonomous Electric Golf Cart System for Passenger Transportation Between Building (Wongsathon Angkhem, 2023)

The researchers were therefore interested in developing an autonomous electric golf cart controlled by artificial intelligence, using: Mapping and positioning with data from optical rangefinders; creating movement paths with high-resolution maps; using artificial intelligence to detect objects and identify their positions in 3D; and creating movement paths for obstacle avoidance. This information is then used to make decisions and control the golf cart, controlling speed and turning angle according to the curvature of the movement path to reach the destination and evaluating its efficiency.

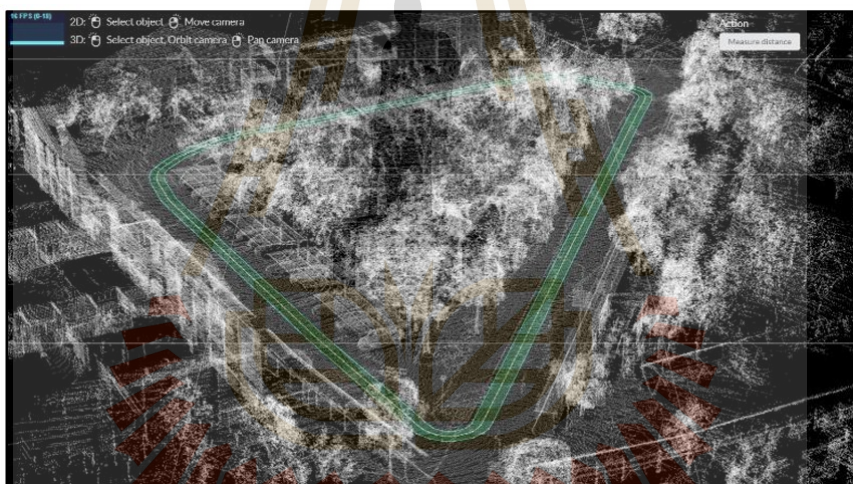


Figure 2.4 An example of creating an outdoor map using 3D LiDAR (Wongsathon A., 2023)

2.1.6 A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation (Mourikis, A., 2007)

The key innovation of MSCKF lies in its state representation and update strategy. Instead of explicitly estimating the positions of visual landmarks within the state vector, the algorithm maintains a sliding window of past camera poses. Feature observations are used to impose multi-state constraints across these poses. IMU data is used for state propagation, while visual measurements provide correction updates.

MSCKF is commonly categorized as a loosely-coupled or filtering-based visual–inertial fusion framework.

Table 2.1 Comparison of the advantages and disadvantages of each method

Method	Advantage	Disadvantage
EKF	It can operate even when one of the sensors is unable to function due to various sensor limitations	The location tracking system lacks accuracy in identifying locations
separation between odometry estimation and mapping (loosely-Coupled Sensor)	High localization accuracy without vision	not fully exploited
Tightly-coupled	Increased computational complexity compared to loosely-coupled systems	may affect the entire estimation pipeline because all constraints are jointly optimized
Depth camera	Gained dense of information, including color and depth.	Sensitive to light and easily disturbed
SLAM	They are able to perform tasks immediately, have freedom in carrying out tasks, and can perform tasks with precision	High-precision sensors are required, as well as computers with high computing power
3D LiDAR	Captures height, shape, and structure of the environment	Often unnecessary for indoor environment Much higher price
2D LiDAR	Great for localization Much cheaper than 3D LiDAR	Planer laser scanner (Measures distance on a single horizontal plane) Wake for full environment understanding

2.2 Robot

2.2.1 F1TNEH

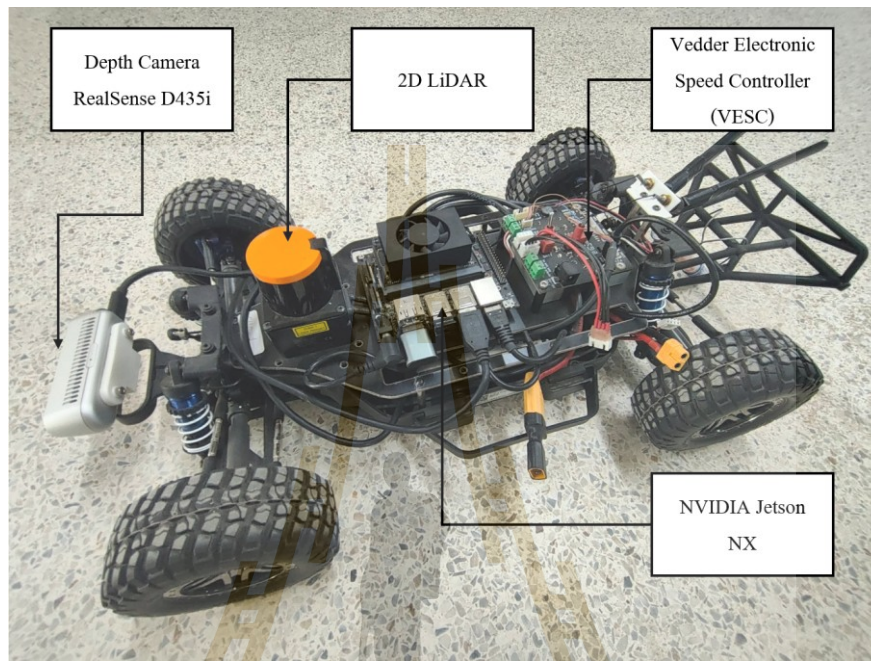


Figure 2.5 illustrates the experimental F1TENTH autonomous vehicle platform used in this study. The system integrates a 2D LiDAR sensor and an RGB-D depth camera for perception, an NVIDIA Jetson NX for onboard computation, and a VESC motor controller for vehicle actuation.

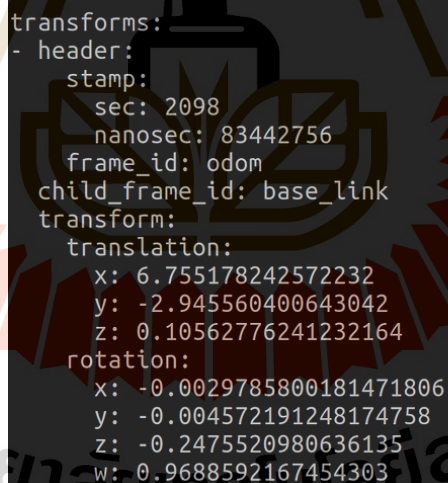
F1TENTH is a compact autonomous vehicle platform developed for research and educational purposes in the field of self-driving systems. Built at a one-tenth scale, the platform integrates onboard sensors, computing hardware, and open-source software to support the development and evaluation of autonomous driving algorithms. It is commonly used for studying core tasks such as environment perception, vehicle localization, mapping, path planning, and control, providing a practical and cost-effective testbed for autonomous vehicle research.

The NVIDIA Jetson NX is a core processing board that serves as a computer operating system, including SLAM and control systems, and supports various sensor protocols for connecting different sensors. VESC (Vedder Electronic Speed Controller)

is an open-source motor controller widely used in robotic and autonomous systems for precise control of brushless DC (BLDC), VESC has been extensively adopted in robotics due to its flexibility, robustness, and support for advanced motor control techniques. 2D LiDAR Provides planar laser scans for LiDAR-based SLAM and localization. Depth Camera (Intel RealSense D435i) Used for RGB-D perception, dense point cloud generation, and visual/depth-based SLAM

2.3 Vedder Electronic Speed Controller VESC

It is used to control BLDC motors by adjusting the power supplied to the motor windings using a technique called Field-Oriented Control (FOC). This allows the motor to operate smoothly by controlling the stator current in relation to the rotor's magnetic field. In the Robot Operating System (ROS), data obtained from VESC is used to calculate the robot's position. This is represented by a topic named /odom.



```

transforms:
- header:
  stamp:
    sec: 2098
    nanosec: 83442756
  frame_id: odom
  child_frame_id: base_link
  transform:
    translation:
      x: 6.755178242572232
      y: -2.945560400643042
      z: 0.10562776241232164
    rotation:
      x: -0.0029785800181471806
      y: -0.004572191248174758
      z: -0.2475520980636135
      w: 0.9688592167454303

```

Figure 2.6 ROS2 shows a ROS 2 transform message representing the robot pose relative to the odometry frame, including position and orientation information used in the localization process

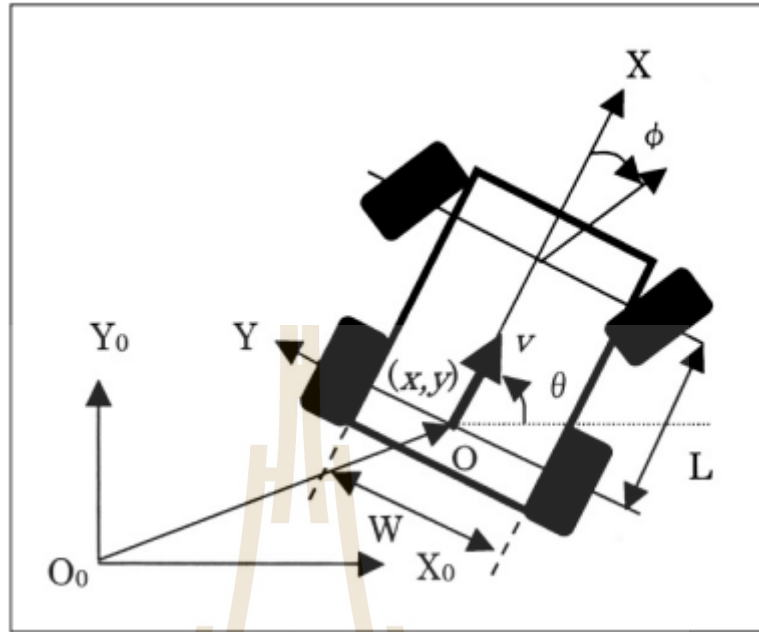


Figure 2.7 Wheel Mobile Robot Model four-wheel mobile (Zunaidi, I, 2006)

illustrates the kinematic model of a four-wheeled mobile robot. The variables x and y denote the robot's position in the navigation coordinate frame, while θ represents the robot's orientation. The steering angle is denoted by ϕ , and v corresponds to the linear velocity of the vehicle. The parameter L indicates the distance between the front and rear axles, and W defines the lateral separation between the left and right wheels.

$$x(k+1) = x(k) + v(k) \cdot \cos(\theta)(k) \cdot \Delta k \quad (2.1)$$

$$y(k+1) = y(k) + v(k) \cdot \sin(\theta)(k) \cdot \Delta k \quad (2.2)$$

$$\theta(k+1) = \theta(k) + \frac{v_l(k) - v_r(k)}{W} \cdot \Delta k \quad (2.3)$$

$$v(k) = \frac{v_l(k) + v_r(k)}{2} \quad (2.4)$$

The robot's planar position and heading are estimated using incremental wheel encoder measurements, as described by Equation (2.6). The position update is obtained by integrating the robot's forward velocity along its heading direction, whereas the orientation update is derived from the difference between the left and

right wheel velocities. The linear velocity $V(k)$ is calculated as the average speed of the left and right rear wheels. Here, k represents the current sampling index, $k + 1$ denotes the subsequent time step, and Δk is the fixed sampling interval. (Zunaidi, I., 2006)

Robot speed and position will be published in the topic /odom via the message nav_msgs/Odometry

Odometry data is not without its challenges. Odometry errors accumulate overtime leading to drift and inaccuracies in the robot's estimated tractor (Thrun, S., 2001) (Grisetti, G., 2010)

2.4 Low level

Low-level control refers to the basic hardware control of robotic systems, involving direct communication with sensors, actuators, and hardware interfaces. This allows the robot to perform basic tasks such as movement and receiving signals from sensors. This research focuses on.

Motor Control: A Vedder Electronic Speed Controller (VESC) is used to control the motors for robot movement. The VESC receives speed commands from the remote control via software and converts them into motor driving signals. The speed depends on user input or automatic commands. For steering, PWM (Pulse Width Modulation) signals are used to adjust the turning radius. Odometry values can be received from the VESC and can be used to approximate the position of the F1 TENTH robot using the odometry data found in the /odom topic.

LiDAR: 2 D LiDAR is installed on the F1 TENTH for sensing the surrounding environment. In low-level operations, this involves measuring the distances of the surrounding environment and converting this data into a usable format for Simultaneous Localization and Mapping (SLAM). In low-level operations, it connects directly to the hardware, receiving distance and angle data from the LiDAR and publishing/scanning it.

RealSense Depth Camera: A depth camera provides a three-dimensional environment. At the low level, it connects directly to the camera hardware to capture depth and Red-Green-Blue (RGB) data and processes this data. The point cloud data,

or depth map, is generated directly from the RealSense camera and published in topics such as `/camera/depth/points` or `/camera/depth/image`. This data is then used for further mapping.

Hardware Abstraction Layer (HAL): This layer integrates system communication with various sensors, including the depth camera, VESC controller, and LiDAR. It utilizes the Robot Operating System (ROS) to isolate hardware-level communication with sensors through built-in drivers and designated topics for high-level communication, such as `/camera/depth/points`, eliminating the need for manual USB communication management.

Communication Protocols (e.g., I2C, CAN Bus, USB): VESC communication utilizes the CAN bus for high-speed, real-time, low-level motor control. This includes programming and sending/receiving data from the VESC via CAN, such as speed commands or distance measurements. The LiDAR unit communicates via I2C or USB at a low level; the driver receives raw data for publication in ROS messages. RealSense communicates via USB on the onboard computer for video and depth data processing.

Real-Time Systems: The F1TENTH operates in a real-time environment, requiring communication between sensors and the processor. The processing board is the NVIDIA Jetson NX, which demands sufficient and fast processing power.

2.5 Robot Operating System ROS

Robot Operating System (ROS) is an open-source software framework that supports the development of robotic systems by offering a standardized set of libraries, tools, and communication mechanisms. It is built on top of a conventional operating system, most commonly Ubuntu Linux, and enables modular and distributed robot software design (M. Quigley, 2009) (Berka, Z. 2020)

In the F1TENTH platform, ROS serves as the central software infrastructure that integrates sensing, computation, and control components. It manages data exchange and coordination among multiple nodes, including the 2D LiDAR, depth camera, and motor controller (VESC). By using a publish–subscribe communication model, ROS allows these components to operate independently while sharing information in real time. ROS (2024)

The ROS framework consists of several core components that support message passing, hardware abstraction, and system coordination, which are essential for autonomous robotic applications.

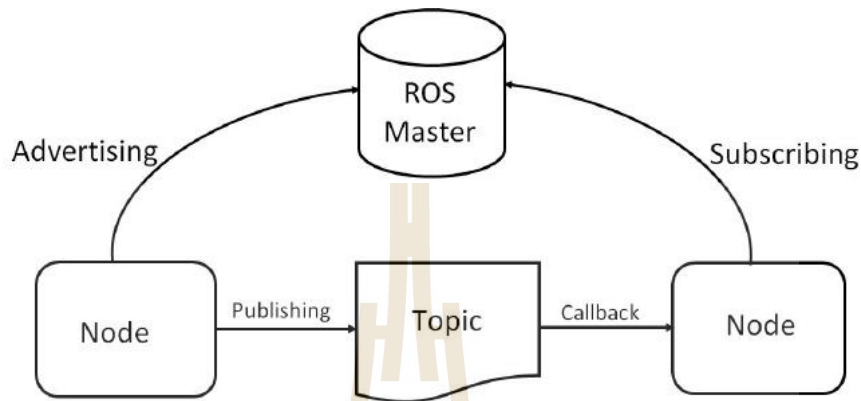


Figure 2.8 The ROS Master Source (ROS, 2024).

Nodes are standalone processes that make up the functional units of a robotic system. Each node is typically designed to handle a specific task, such as processing sensor data or managing actuator control. Communication between nodes occurs through mechanisms like topics, services, or actions. In this communication model, the node that transmits data is referred to as a publisher, while the node that receives the data is known as a subscriber. A single node may perform both roles simultaneously. In addition, ROS allows nodes to be developed using multiple programming languages, including Python and C++, enabling flexible system design and integration.

Topics serve as logical communication channels that allow nodes to exchange data within the ROS ecosystem. Data are transmitted in the form of messages over these channels, enabling nodes to publish information while others subscribe to receive it. A node may both send and receive data through different topics as needed. This publishes–subscribe communication approach operates asynchronously, making it well suited for continuous data streams such as LiDAR measurements, camera imagery, velocity instructions, and motor control signals.

Messages define the data formats exchanged between nodes and act as the fundamental units of communication in ROS. Each message specifies the structure of the transmitted data, which may include numerical values, floating-point variables, or collections such as arrays. For instance, information obtained from a LiDAR sensor can be organized and transmitted as an array within a message.

Unlike topics, which are designed for continuous data transmission, services provide a synchronous communication mechanism based on a request–reply pattern. Each service is defined by two message types: one that carries the request and another that returns the response. This form of communication is typically used for brief, discrete operations, such as querying a node’s status or triggering a specific action within the robotic system.

The ROS Master acts as a coordination service that oversees communication within the system. It keeps track of active nodes, manages information about publishers, subscribers, and available services, and allows nodes to locate one another. Through this mechanism, nodes are able to form connections dynamically during runtime and exchange data efficiently.

ROS structures its software components into packages that group related resources together. A package may include source code written in C++ or Python, along with launch scripts, message definitions, configuration files, and supporting data. These packages serve as the primary unit for organizing and distributing ROS-based software, and they can be shared and reused by other developers through version control platforms such as GitHub.

Launch files provide a convenient way to start multiple nodes and load their associated configuration files at the same time. They are commonly used to specify parameters required by individual nodes, ensuring consistent system setup. These files are typically written using XML or YAML formats, allowing structured and readable system configuration.

The Parameter Server functions as a centralized storage system that allows nodes to access and retrieve shared configuration parameters. It enables nodes to obtain necessary values without hardcoding them into the source code, since parameters can be defined or adjusted during runtime. This approach supports flexible system initialization and simplifies configuration management across the robotic application.

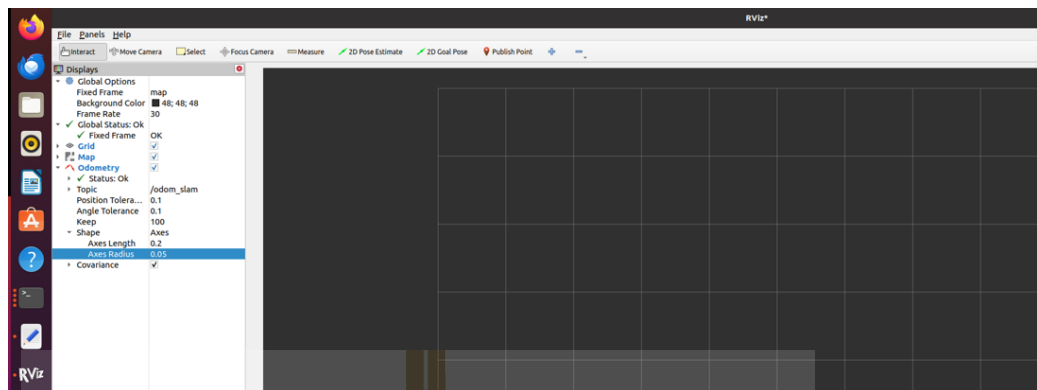


Figure 2.9 RViz2 terminal

RViz (ROS Visualization) is a three-dimensional visualization utility that presents sensor outputs, robot models, maps, and other system data in real time. It allows developers to observe how a robot interprets its surroundings and how different components of the system interact within the operating environment.

Gazebo is a robotics simulation platform that integrates with ROS to model realistic physical environments, including factors such as lighting conditions and sensor behavior. It enables accurate simulation of robot dynamics and sensor outputs, allowing developers to evaluate and refine robotic algorithms in a virtual setting before deploying them on real-world hardware.

In addition to tools that support software development, ROS includes the tf library, also known as the Transform Library, which represents various components of a robotic system as independent coordinate frames. This library manages the spatial relationships between these frames, enabling consistent tracking of positions and orientations across the system.

2.6 Simultaneous Localization and Mapping (SLAM)

Simultaneous Localization and Mapping (SLAM) refers to a computational process in which a robot or autonomous system constructs a representation of an unfamiliar environment while concurrently estimating its own position within that space. This capability is essential for autonomous navigation, particularly in situations where global positioning systems are unavailable, such as indoor environments, enclosed areas, or building interiors. SLAM consists of two tightly coupled processes:

localization, which determines the system's current pose, and mapping, which involves capturing and updating information about the surrounding environment. These two processes operate in parallel and continuously inform one another.

In this research, a comprehensive literature review was carried out covering SLAM techniques, LiDAR sensing, sensor fusion methods, depth camera technologies, robotic control systems, and data communication within ROS. The purpose of this review was to establish a solid theoretical foundation for the proposed system. Based on these principles, various testing methodologies were designed, and appropriate data were collected to support a complete and reliable evaluation. The goal was to develop a system capable of accurately estimating position within environments suitable for camera-based operation (REP, 103).

2.6.1 Coordinate frame conventions

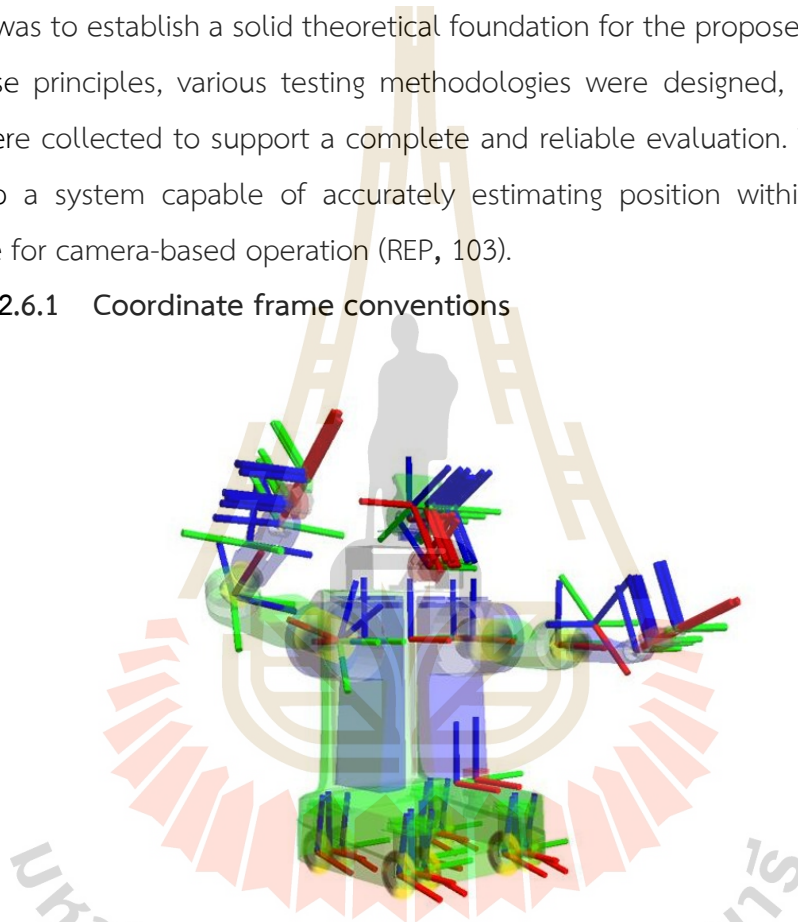


Figure 2.10 Illustration of multiple reference frames used in a robotic system, showing the relationships between different coordinate transformations within the robot structure (Foote, 2013)

The functioning of SLAM is fundamentally based on coordinate frame conventions. A coordinate frame defines a spatial reference system used to describe positions and orientations within an environment. It provides a consistent way to represent motion and spatial relationships and is composed of several key elements:

Origin: A fixed reference point within the coordinate system that serves as the basis for all position measurements.

Axes: Three mutually perpendicular directions, typically labeled X, Y, and Z, which define orientation in three-dimensional space.

Unit of Measurement: A defined scale used to quantify distances along each axis.

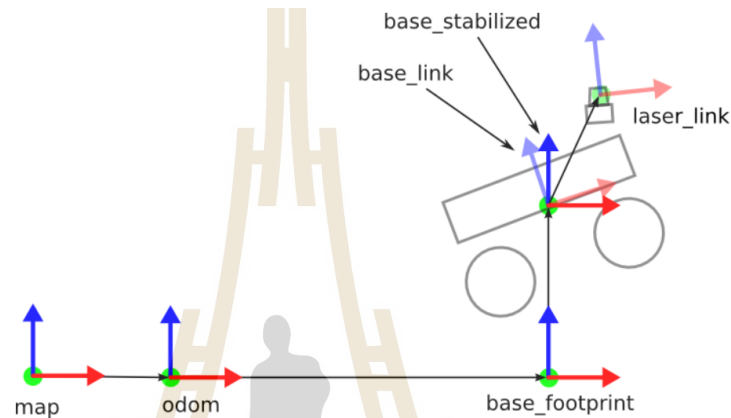


Figure2.11 Diagram illustrating the coordinate frame hierarchy in a mobile robot, including the map, odometry, base footprint, base link, stabilized base, and laser sensor frames, and their relative transformations within the ROS tf framework (ROS) (M. Quigley., 2009).

base_link represents the coordinate frame attached to the robot itself and moves consistently with the robot's orientation and direction of motion. The exact location of this frame depends on the specific robot platform and its hardware configuration. ROS (2024).

odom refers to a globally referenced frame derived from odometry data that reflects the robot's movement over time. While it provides smooth and continuous position estimates, accumulated drift errors inevitably occur and cannot be corrected retroactively. (Ref 103)

map denotes a fixed reference frame associated with the surrounding environment, providing a global coordinate system that is independent of the robot's motion.

2.6.2 Simultaneous Localization and Mapping (SLAM)

In SLAM operation, the `/scan` topic obtained from the LiDAR sensor is continuously used to observe and interpret the surrounding environment. This data allows the system to determine the robot's position relative to its environment by analyzing the geometric information captured in each scan. The processed localization information is referenced within the `/map` frame, which represents a global coordinate system of the environment.

At the same time, the `/odom` frame serves as an additional reference for estimating the robot's motion based on incremental movement. These two coordinate frames operate together to determine the position of the `/base_link`, which represents the robot's physical reference frame. By combining information from both `/map` and `/odom`, the SLAM system can maintain a consistent estimate of the robot's pose over time.

Based on this formulation, the SLAM problem at a given time step t consists of several key elements. One of the primary components is the robot's pose, denoted as X_T which in this research corresponds to the F1TENTH platform. The robot pose can be represented in either two-dimensional or three-dimensional vector form, depending on the application. Furthermore, the sequence of robot poses over time can be combined to describe the robot's entire trajectory, denoted as X_T , where T represents the total number of time steps. This trajectory can be expressed as:

$$X_T = \{x_0, x_1, x_2, \dots, x_T\} \quad (2.5)$$

For a SLAM system to operate, an initial robot pose X_0 is required. In situations where the system begins operation in an environment with no prior information, this initial pose is typically defined as a reference position. All subsequent pose estimates are then computed relative to this initial reference frame.

Odometry refers to data that describes the relationship between the robot's positions at different time steps. This information represents the robot's motion from time $t-1$ to time t , or more generally, the change in position between any two-time instances. Odometry data may include measurements such as traveled distance, velocity, acceleration, or a combination of multiple motion-related variables.

This information is commonly denoted as u_t , and the complete sequence of odometry measurements over time can be expressed as:

$$U_T = \{u_0, u_1, u_2, \dots, u_T\} \quad (2.6)$$

From a theoretical perspective, if the odometry measurements were free of noise, the sequence U_T could be directly integrated to recover the robot's trajectory X_T . However, in real-world applications, odometry data inevitably contains noise and uncertainty. As a result, trajectories estimated using odometry alone tend to drift over time and gradually deviate from the true path of the robot.

Objects represent abstract descriptions of the surrounding environment and may correspond to landmarks, physical objects, surfaces, or other environmental features, depending on the type of sensor used for perception. These objects are used to model the environment within the SLAM framework and are typically assumed to be static. Each object is described by its spatial location, which does not vary over time. The collection of all detected objects in the environment can be represented as:

$$m = \{m_1, m_2, m_3, \dots, m_n\} \quad (2.7)$$

The relationship between the robot's pose at a given time t , denoted as x_t , and the set of environmental objects m is captured through sensor measurements. These measurements describe the relative distances or spatial relationships between the robot and each detected object at time t . Such observations provide information about how far the robot is from the surrounding objects and are essential for correcting pose estimates. This measurement relationship can be expressed as:

$$Z_{n:T} = \begin{Bmatrix} Z_{1:1} & Z_{2:1} & A_{3:1} & \cdots & Z_{n:1} \\ Z_{1:2} & Z_{2:2} & Z_{3:2} & \cdots & Z_{n:2} \\ Z_{1:3} & Z_{2:3} & Z_{3:3} & \cdots & Z_{n:3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ Z_{1:T} & Z_{2:T} & Z_{3:T} & Z_{n:T} & Z_{n:T} \end{Bmatrix} \quad (2.8)$$

Based on the components described above, SLAM can be summarized as a process that simultaneously estimates the robot's trajectory X_T and constructs a representation of the environment m_n . This process integrates motion information obtained from odometry data U_T with sensor observation $Z_{n:T}$. The complete formulation, which aims to estimate both the full trajectory and the entire map, is commonly referred to as **Full SLAM**. This relationship can be expressed probabilistically as:

$$P(X_T, m_n | Z_{n:T}, U_T) \quad (2.9)$$

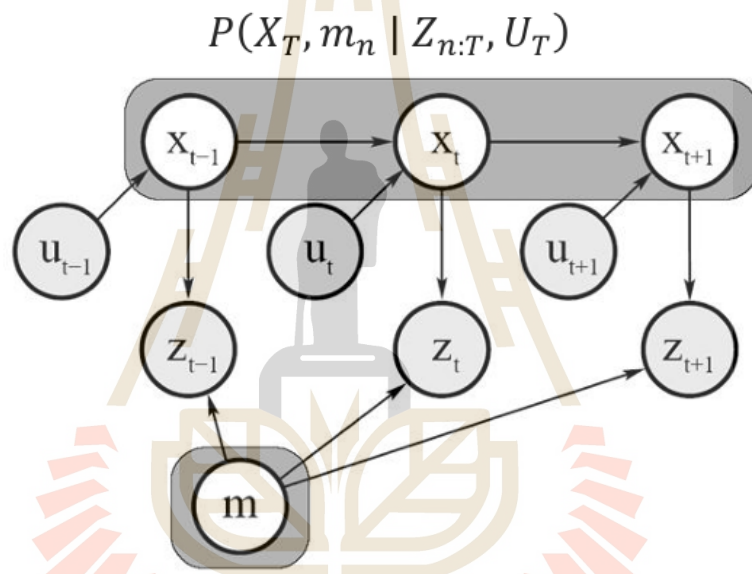


Figure 2.12 illustrates the overall process of Full SLAM, (thrun, 2007)

In Full SLAM, the system computes the entire sequence of robot poses X_T together with all environmental features m_n that can be inferred from the available data. While this comprehensive estimation provides a detailed representation of both motion and environment, it is not always necessary in practical applications. In many scenarios, the robot or user may only require the robot's current pose rather than the complete trajectory history.

To address this requirement, an alternative formulation known as **Online SLAM** has been developed. Unlike Full SLAM, Online SLAM focuses on estimating only the current robot pose x_T instead of the full trajectory X_T . By marginalizing past states,

Online SLAM reduces computational complexity while still maintaining sufficient localization accuracy for real-time operation. This formulation can also be described using a probabilistic framework derived from the Full SLAM representation.

In this research, two SLAM approaches are employed: LiDAR-based SLAM for processing laser range measurements, and odometry data to support motion estimation. However, odometry measurements are inherently imperfect. Errors in odometry accumulate over time, leading to drift and increasing inaccuracies in the estimated robot trajectory, as reported in previous studies. This limitation highlights the importance of incorporating sensor-based observations, such as LiDAR measurements, to correct accumulated motion errors and improve localization reliability. (Thrun, s., 2001)(Trun, s.,2007)

2.6.3 LiDAR SLAM

LiDAR SLAM is a type of SLAM technique that is fundamentally based on a principle known as scan matching. This approach operates using laser-based distance sensing devices, commonly referred to as LiDAR (Light Detection and Ranging). LiDAR functions through a laser scanning process in which distances between the sensor and surrounding objects are measured. This is achieved by emitting laser pulses toward objects and calculating the distance based on the time taken for the reflected light to return to the sensor. (Saeefi. S, 2020)

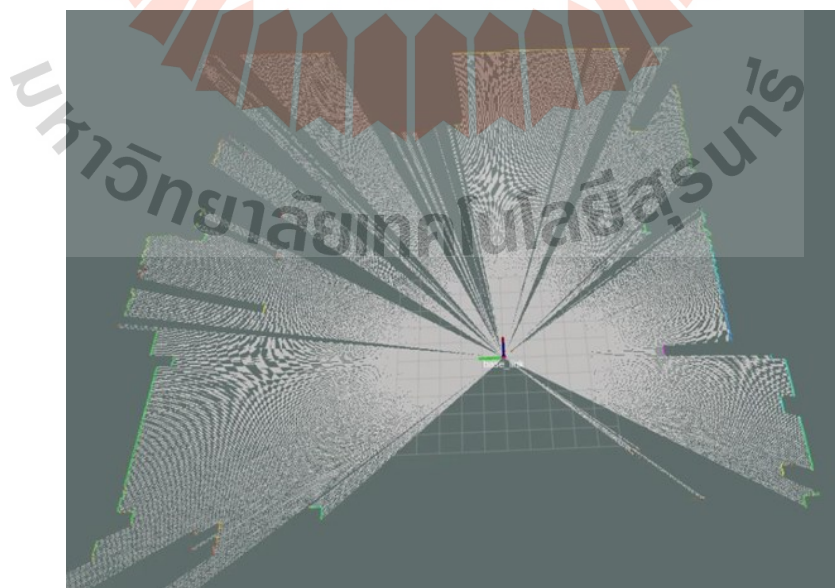


Figure 2.13 Example of LiDAR SLAM usage

There are several types of LiDAR sensors, ranging from single-direction distance-measuring LiDARs, rotating LiDARs capable of scanning surrounding objects in a two-dimensional plane, to advanced LiDAR systems that can capture three-dimensional information of the environment. Despite these differences, all LiDAR sensors operate on the same fundamental physical principle: the propagation and reflection of light.

In LiDAR SLAM, consecutive laser scans are aligned by matching geometric features between scans. This scan matching process allows the system to estimate the robot's motion over time while simultaneously constructing a representation of the surrounding environment. Because LiDAR directly measures spatial geometry, LiDAR SLAM is generally robust to lighting conditions and can operate reliably in environments where visual features are limited or inconsistent.

$$Distance = \frac{\text{speed of light} \times \text{Time of flight}}{2} \quad (2.10)$$

2.6.4 Visual SLAM

Visual SLAM is a SLAM approach that relies on image data for processing. In this method, the SLAM system extracts visual features from images, where features typically correspond to regions with noticeable changes in pixel intensity or color, such as edges and corners of objects within the scene. These features are tracked across consecutive frames using feature tracking techniques, which enables the system to estimate motion, construct a map, and determine the robot's position simultaneously.

One advantage of this approach is that Visual SLAM can be implemented using various types of cameras. When stereo cameras are employed, additional depth information can be obtained for each detected feature. Stereo cameras are specifically designed for three-dimensional imaging by capturing two images simultaneously from slightly different viewpoints. The captured image pairs are processed to extract visual features based on surface color, texture, intensity, and structural information present in both images. These features are then used to generate point cloud representations for each view.

Subsequently, corresponding points that represent the same physical locations in the real world are matched between the point clouds. This

correspondence allows the system to compute the distance from the camera to each point and reconstruct a three-dimensional representation of the environment. The resulting 3D information enhances both mapping accuracy and localization performance.

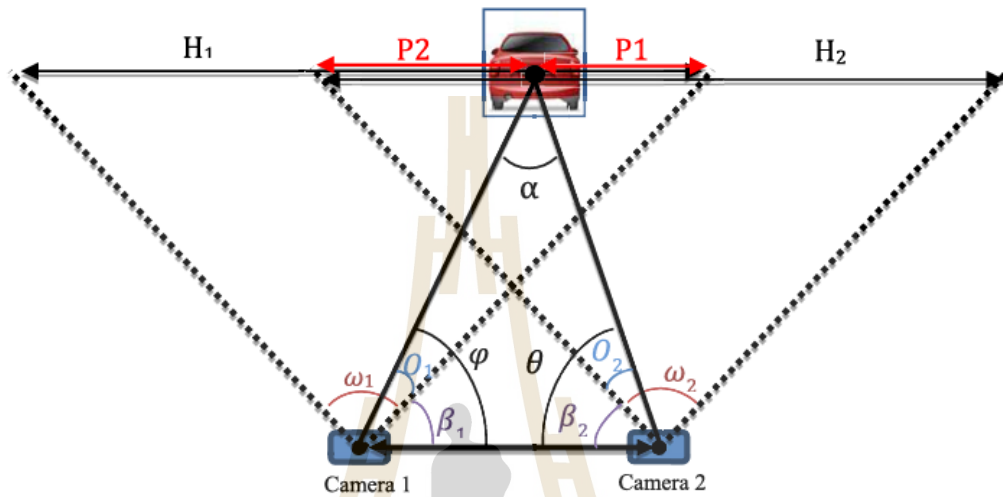


Figure 2.14 Illustration of the angles used for computing the distance (A. Zaarane, 2020)

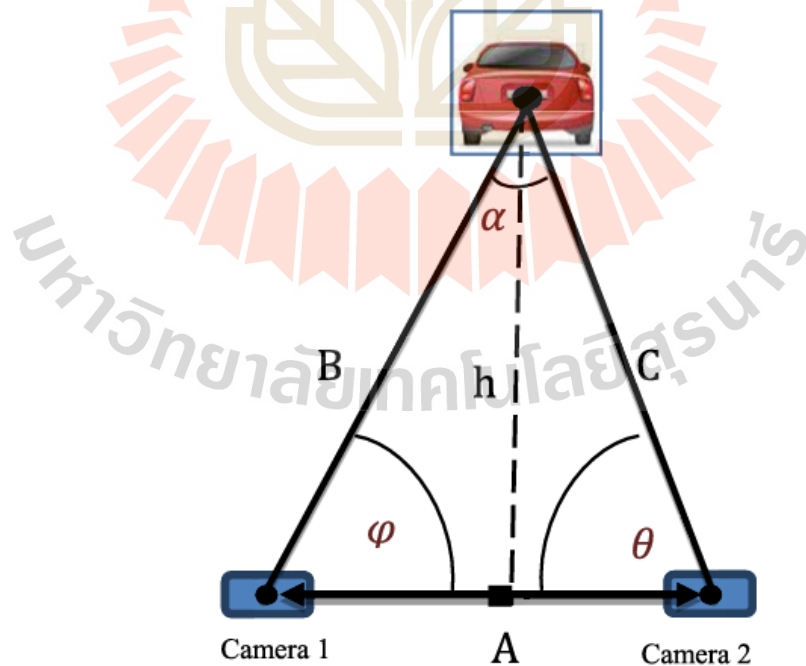


Figure 2.15 Example of two cameras mounted as a stereo camera (A. Zaarane, 2020)

The distance can be calculated following equation.

$$h = \frac{A \sin \left(P_2 \cdot \left(\frac{\omega_2}{H_2} + \beta_2 \right) \sin(P_1) \cdot \left(\frac{\omega_1}{H_1} + \beta_1 \right) \right)}{\sin \left(180 - \left(P_2 \cdot \frac{\omega_2}{H_2} + \beta_2 + P_1 \cdot \frac{\omega_1}{H_1} + \beta_1 \right) \right)} \quad (2.11)$$



CHAPTER III

RESERCH METHODOLOGY

3.1 Methods of data collection

During the experiments, all data generated by the system, including both processed and raw sensor data, were transmitted through the Robot Operating System (ROS) framework. ROS provides a standardized communication mechanism that allows different system components to exchange information in real time.

To record experimental data, a built-in ROS tool known as rosbag was utilized. Rosbag enables the recording of ROS messages published on selected topics and stores them in files with the bag format. This functionality allows comprehensive logging of sensor measurements, localization outputs, and system states throughout the experiment.

In addition to data recording, rosbag supports data playback, which makes it possible to replay previously recorded experiments offline. This capability is particularly useful for repeated analysis, algorithm evaluation, and performance comparison under identical conditions without requiring the physical robot to be rerun. As a result, rosbag provides a reliable and reproducible method for collecting and analyzing experimental data in this research.

3.2 Procedure

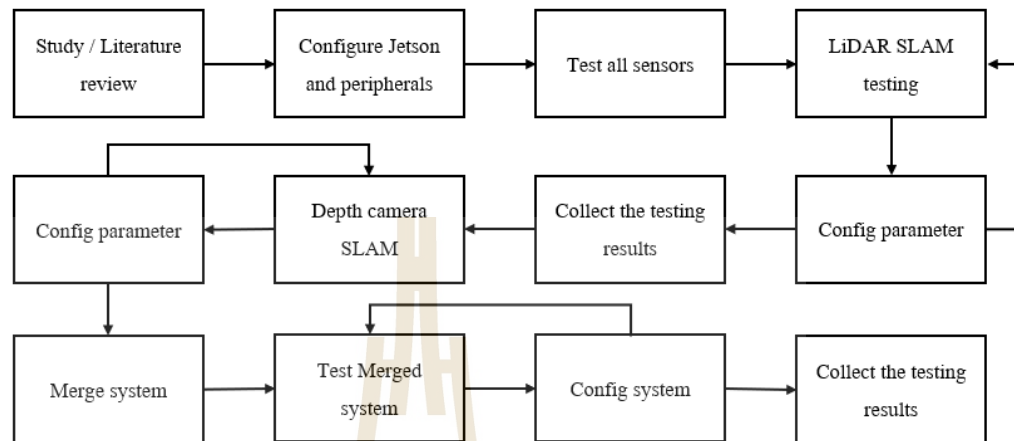


Figure 3.1 Procedure

3.2.1 Study and Literature review

A comprehensive literature review was conducted based on academic papers, technical documents, and previous research related to SLAM, LiDAR, sensor fusion, and depth camera technologies. In addition, studies concerning robotic control systems and data communication within the Robot Operating System (ROS) framework were reviewed.

The purpose of this review was to gain a clear understanding of the underlying principles and theoretical foundations relevant to this research. Particular attention was given to the strengths and limitations of different sensing modalities, as well as their suitability for localization and mapping tasks. Furthermore, existing experimental methodologies and data collection strategies were examined to support the design of appropriate testing procedures.

This review provided essential guidance for selecting suitable sensors, designing experimental scenarios, and collecting relevant data, ultimately contributing to a complete research framework capable of accurately estimating robot position in environments suitable for camera-based operation.

3.2.2 Configure Jetson and Peripherals

This stage focuses on configuring the NVIDIA Jetson Xavier NX and its associated peripheral devices for use on the F1TENTH autonomous platform. The setup process includes establishing remote communication with the Jetson via Secure Shell (SSH) and connecting an external monitor to enable system monitoring and configuration.

These steps allow the Jetson to execute autonomous programs on the F1TENTH platform while maintaining reliable communication with a laptop for development, debugging, and system control. The following figure illustrates the data and hardware connections used within the system architecture.

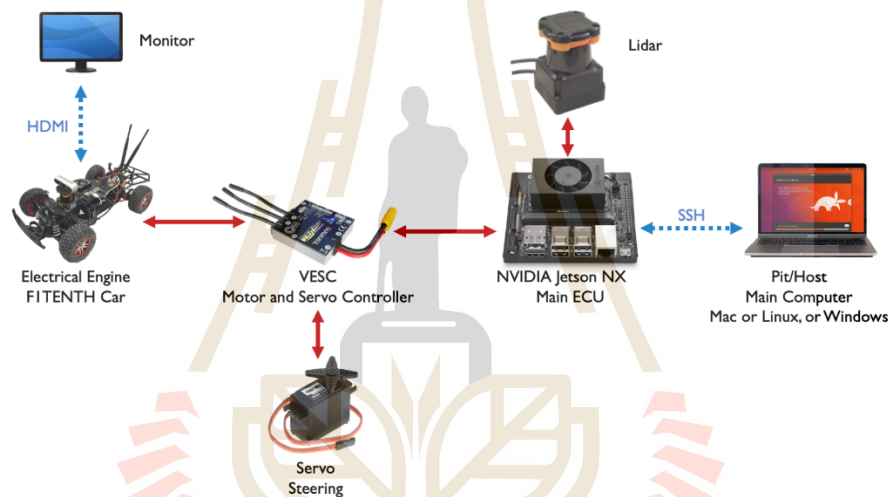


Figure 3.2 System architecture of the F1TENTH platform, illustrating the data flow and control connections between the NVIDIA Jetson Xavier NX, LiDAR sensor, VESC motor and steering controller, onboard actuators, and the pit/host computer via SSH and HDMI interfaces.

The NVIDIA Jetson Xavier NX serves as the central processing unit of the entire system. It is responsible for issuing control commands to the Vedder Electronic Speed Controller (VESC), which manages the steering servo and the brushless motor of the F1TENTH platform. In addition to control tasks, the Jetson NX receives sensory data from the Light Detection and Ranging (LiDAR) sensor and the depth camera, which are connected via USB or Ethernet interfaces.

All communication and data exchange within the system are handled through the **ROS 2** framework. A laptop acting as the pit or host computer is used to

remotely access the onboard system, enabling monitoring, debugging, and control during operation. This host computer also supports real-time visualization of system data through a connected monitor using an HDMI interface.

3.2.3 Test all sensors

The objective of this testing phase is to verify the operational integrity and data accuracy of the primary perception and localization suite—specifically the 2D LiDAR, the RGB-D depth camera, and the wheel-encoder-based odometry. The validation process follows a tiered approach, beginning with individual sensor stream verification and concluding with integrated motion testing. Verification commences by initializing the Robot Operating System (ROS) environment to ensure all sensor nodes are publishing to their designated topics. Using the `rostopic echo` or `ros2 topic echo` command, the frequency and consistency of data on `scan` (LiDAR), `camera`, `depth`, `image_raw` (Depth Camera), and `odom` (Odometry) are monitored. A consistent publishing rate.

The 2D LiDAR is tested using a scan-matching algorithm (e.g., SLAM Toolbox). Key parameters, specifically `scan_buffer_size` and `minimum_time_interval`, are tuned to handle high-speed data intake.

3.2.4 SLAM testing and configuration parameter

This research phase focuses on the independent optimization of LiDAR-based and Depth-based SLAM systems. The objective is to achieve a synchronized state where both modalities exhibit minimal drift and high map fidelity, ensuring they are prepared for sensor fusion.

3.2.5 Merge system

The core of the navigation stack relies on a LiDAR-centric SLAM approach (e.g., utilizing a LiDAR Odometry and Mapping framework). The justification for using LiDAR as the exclusive localization base the Depth Camera as the volumetric data source.

3.3 The overall system operation.

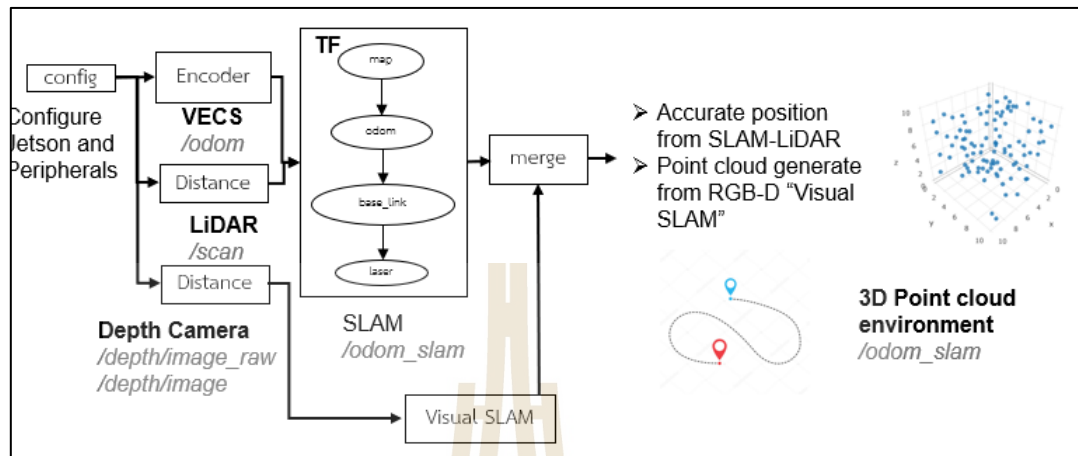


Figure 3.3 The Overall system operation

Methodology used to develop and evaluate a multi-sensor SLAM system on the F1TENTH autonomous robotic platform. The proposed approach integrates 2D LiDAR-based SLAM for robust localization with depth camera-based sensing for dense environmental mapping, particularly under challenging lighting conditions where vision-based localization becomes unreliable

The overall system configuration consists of three main components: hardware configuration, software architecture, and sensor integration. These components work together to enable localization, mapping, and data fusion in real time.

3.3.1 Computation Unit: NVIDIA Jetson NX

An NVIDIA Jetson NX is used as the primary onboard computing unit. This embedded computer is responsible for processing LiDAR scan data, Acquiring and processing depth camera images, executing SLAM algorithms, Managing ROS2 node communication, Broadcasting and receiving TF transformation



Figure 3.4 NVIDIA Jetson NX

The Jetson NX is selected due to its high-performance CPU and GPU capabilities, which enable real-time processing of depth images and point clouds while maintaining low power consumption. The system runs Ubuntu Linux as the operating system, with ROS 2 installed as the middleware framework.

The Jetson is mounted centrally on the chassis to reduce vibration and ensure stable cable routing to connected sensors.

3.3.2 Hardware Configuration

Computation Unit An NVIDIA Jetson NX is used as the main onboard computer. It performs sensor data processing, SLAM computation, and communication between software nodes. The Jetson platform is selected due to its balance between computational performance and power efficiency, making it suitable for real-time robotic applications.

Actuation and Odometry Source Vehicle motion is controlled using a Vedder Electronic Speed Controller (VESC), which drives the electric motor and steering servo. Wheel encoder data obtained from the VESC is used to estimate vehicle motion, producing raw odometry information that represents incremental displacement over time.

Sensor Mounting and Frame Alignment All sensors, including the LiDAR and depth camera, are rigidly mounted on the vehicle chassis. Their relative positions are defined using static transforms to ensure consistent spatial relationships between sensor frames and the robot base frame.

3.3.2.1 Actuation and Odometry Source: VESC Controller



Figure 3.5 VESC

The vehicle's propulsion and steering are controlled using a Vedder Electronic Speed Controller (VESC). The VESC interfaces with the electric motor and steering servo and provides feedback from wheel encoders.

Wheel encoder measurements are used to estimate the rotational speed of the drive wheels. From these measurements, incremental motion information is computed and published as odometry data (/odom). This odometry represents relative motion based on wheel rotation and is subject to cumulative drift over time due to wheel slip and surface irregularities.

Despite these limitations, wheel odometry provides high-frequency motion estimates that are essential for short-term motion tracking and sensor synchronization.

3.3.2.2 2D LiDAR Sensor

A 2D LiDAR sensor is mounted on the top of the vehicle, aligned with the forward direction of motion. The LiDAR emits laser beams in a horizontal plane and measures the distance to surrounding obstacles based on time-of-flight or phase-shift principles.



Figure 3.6 2D LiDAR

The LiDAR publishes distance measurements as a laser scan (/scan), which is used as the primary input for LiDAR-based SLAM. The LiDAR frame is statically transformed to the robot base frame to ensure accurate spatial alignment during localization.

3.3.2.3 Depth Camera Sensor

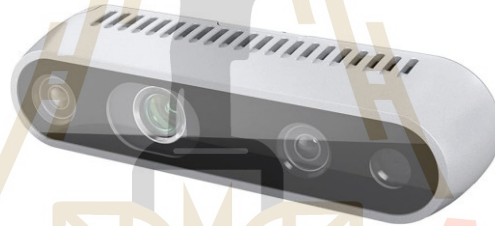


Figure 3.7 D435i

A depth camera (Intel RealSense D435i) is mounted at the front of the vehicle to capture three-dimensional information about the environment. The camera provides depth images in which each pixel corresponds to the distance between the camera and observed surfaces.

The camera includes both infrared sensors and an RGB sensor; however, in this research, the depth stream is the primary data source. The depth camera is rigidly mounted, and its coordinate frame is defined relative to the base_link frame through a static transformation.

3.3.2.4 Sensor Mounting and Coordinate Frame Alignment

Accurate sensor fusion requires precise knowledge of the relative positions and orientations of all sensors. Each sensor is mounted rigidly to minimize mechanical vibration and displacement during motion.

Static transformations are defined between:

base_link and LiDAR frame

base_link and camera frame

Camera frame and camera optical frames

These transformations are published using ROS 2 static transform publishers and remain constant throughout operation. Correct frame alignment ensures that LiDAR scans, depth point clouds, and odometry data can be consistently represented in a shared coordinate system.

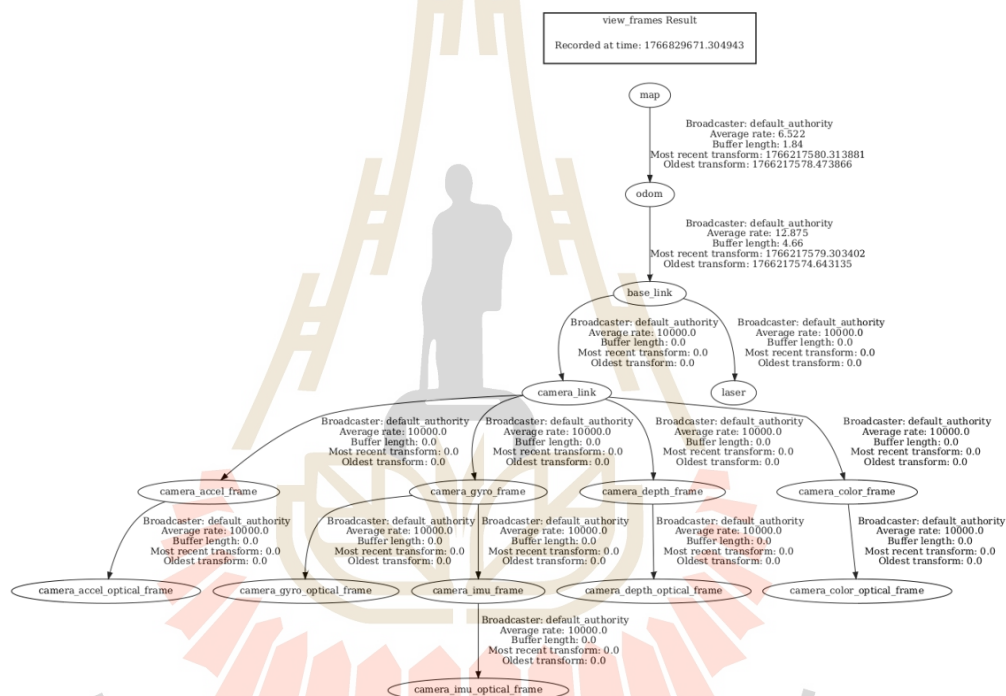


Figure 3.8 Coordinate frame hierarchy of the proposed system visualized using the ROS TF tree. The diagram illustrates the transformation relationships between the global map frame, odometry frame, robot base frame, LiDAR frame, and RGB-D camera frames.

3.3.3 Software Architecture

The software framework used in this research is the Robot Operating System 2 (ROS 2). ROS 2 provides a modular and distributed architecture that enables communication between different software components through topics, services, and the TF transformation system. The TF framework is used to manage coordinate

transformations between frames such as map, odom, base_link, and sensor frames. This ensures that data from different sensors can be fused consistently within a common reference frame.

3.3.3.1 LiDAR-Based Localization

A 2D LiDAR sensor is used to measure distances to surrounding obstacles in a horizontal plane. The LiDAR continuously publishes range measurements as a laser scan, which represents the environment as a set of angular distance samples.

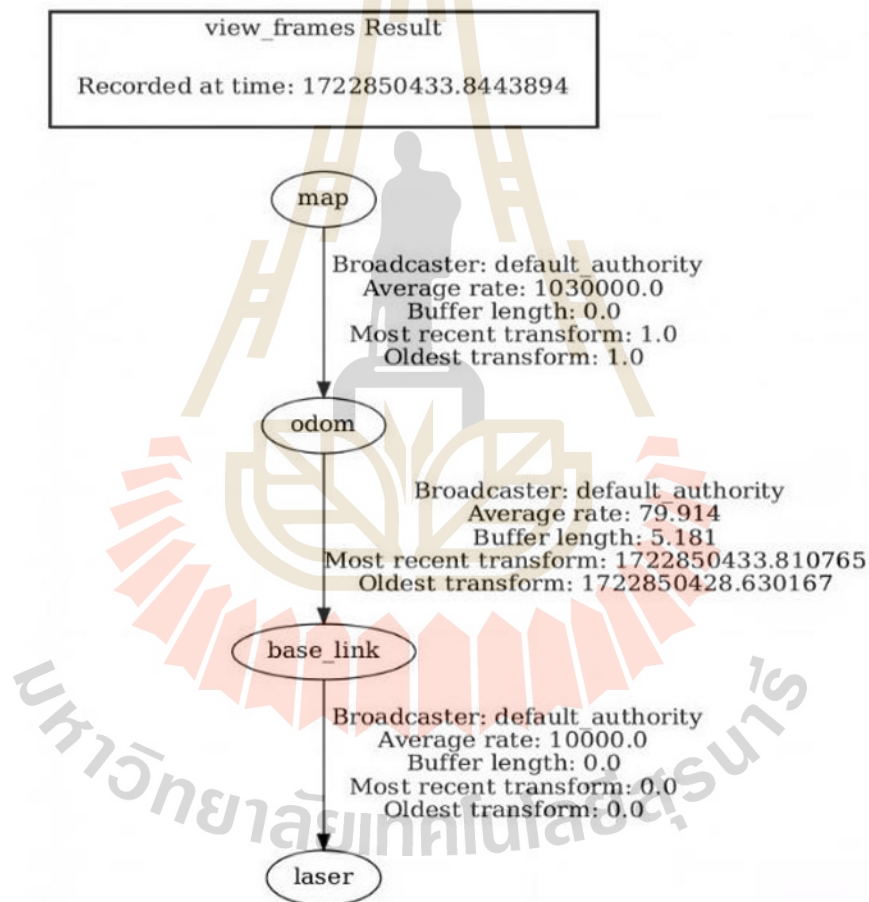


Figure 3.9 Simplified ROS TF frame structure showing the transformation chain from the global map frame to odom, base_link, and the LiDAR (laser) frame.

The LiDAR is configured to operate at a fixed scanning frequency and is aligned with the robot's forward direction. The LiDAR data is published in the sensor frame and transformed into the robot base frame using a static TF.

In this research, LiDAR-based SLAM is employed primarily for localization rather than dense mapping. The SLAM algorithm estimates the robot's pose by matching consecutive laser scans against a growing map of the environment.

The output of the LiDAR SLAM system includes, A 2D occupancy grid map representing the environment, A pose estimate of the robot relative to the map, A transformation between the map and odom frames

This localization output is published as `/odom_slam` and provides a drift-corrected estimate of the robot's motion. Because LiDAR measurements are largely invariant to lighting conditions, the localization remains stable even when visual sensors degrade.

3.3.3.2 LiDAR-Based Localization

Depth images are converted into 3D point clouds using a pinhole camera model. Each pixel in the depth image is reprojected into a 3D point based on its depth value and the camera's intrinsic parameters.

This process produces a dense point cloud that represents the local geometry of the environment in the camera coordinate frame. The point cloud is then transformed into the global reference frame using TF information.

3.3.4 Integration of LiDAR Localization and Depth Mapping

The proposed system adopts a loosely-coupled sensor fusion approach. In this framework, each sensor processes its data independently, and the results are combined at the pose level rather than at the raw measurement level.

LiDAR-based SLAM provides the robot pose estimate, while the depth camera provides dense spatial information. The two subsystems communicate through shared coordinate frames rather than tightly integrated measurement models. The pose output from the LiDAR SLAM system (`/odom_slam`) is used to position the depth-based point clouds in the global map frame. This ensures that the dense map follows the accurate trajectory estimated by LiDAR, even when depth data quality degrades. By decoupling localization and dense mapping, the system prevents visual drift from affecting the robot's estimated trajectory.

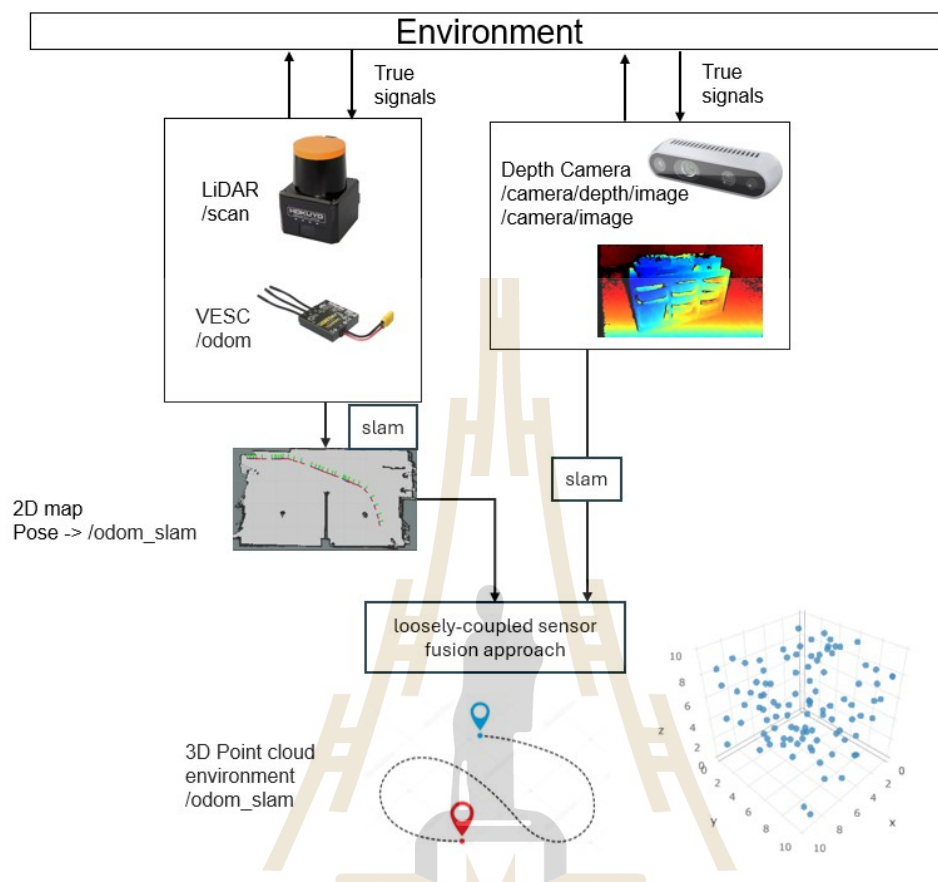


Figure 3.10 Loosely-coupled sensor fusion concept

CHAPTER IV

RESULTS AND DISCUSSION

4.1 Results and Discussion

Reliable localization is a critical capability for autonomous robotic systems, as it directly affects navigation accuracy, mapping quality, and overall system safety. Although many SLAM techniques have shown strong performance in controlled environments, real-world operation introduces additional challenges such as sensor noise, environmental variability, and temporary sensor unavailability. As a result, evaluating a SLAM system solely under ideal conditions is insufficient to fully characterize its performance.

This study conducts a systematic experimental evaluation of multiple SLAM-based localization approaches using a known reference trajectory as ground truth. The evaluation focuses not only on trajectory accuracy during normal operation, but also on system behavior when sensor input is partially lost. By doing so, the experiments aim to reflect more realistic operational scenarios encountered in practical robotic applications.

Experiment 1 Trajectory Estimation under Normal Conditions. The first experiment evaluates localization performance during standard operation, where all sensors function normally. Five independent test runs were performed using different trajectory layouts to assess consistency across varying motion patterns.

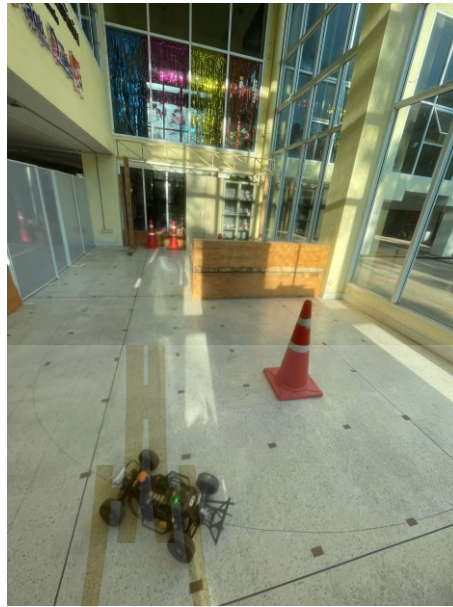


Figure 4.1 Comparing LiDAR SLAM and RTAB-Map Against a Ground-Truth Path test

Experiment 2 Loop Closure Activation Test. The second experiment investigates the loop closure capability of the SLAM systems. In this setup, the robot follows a closed-loop trajectory and returns to a previously visited location



Figure 4.2 Experimental Environment Setup for Robustness Evaluation Experiment 2

Experiment 3: Robustness Evaluation under Sensor Degradation The third experiment examines system robustness by simulating sensor failure conditions. During robot operation, the camera input is intentionally disabled while the robot continues to move.

An important distinction in this evaluation is the architectural difference between the tested systems. One approach relies heavily on camera-based perception

for both localization and mapping, while the proposed system separates these responsibilities by using LiDAR data for localization and depth data primarily for mapping. This experimental design allows the study to highlight how sensor dependency and system architecture influence accuracy, robustness, and fault tolerance.

The following sections present detailed results and discussions for each experiment, beginning with trajectory comparison under normal conditions and followed by robustness analysis during camera interruption.

4.1.1 Comparing LiDAR SLAM and RTAB-Map SLAM Against a Ground-Truth Path

Experiment 1 evaluates localization performance of two SLAM pipelines. LiDAR-based SLAM (`odom_slam`) and RTAB-Map SLAM (RTABMAP). Both systems estimate the robot trajectory in a 2D plane and are compared against a true path reference. The goal is to assess how closely each SLAM output follows the ground-truth trajectory during normal operation, and to identify where and why deviations occur.

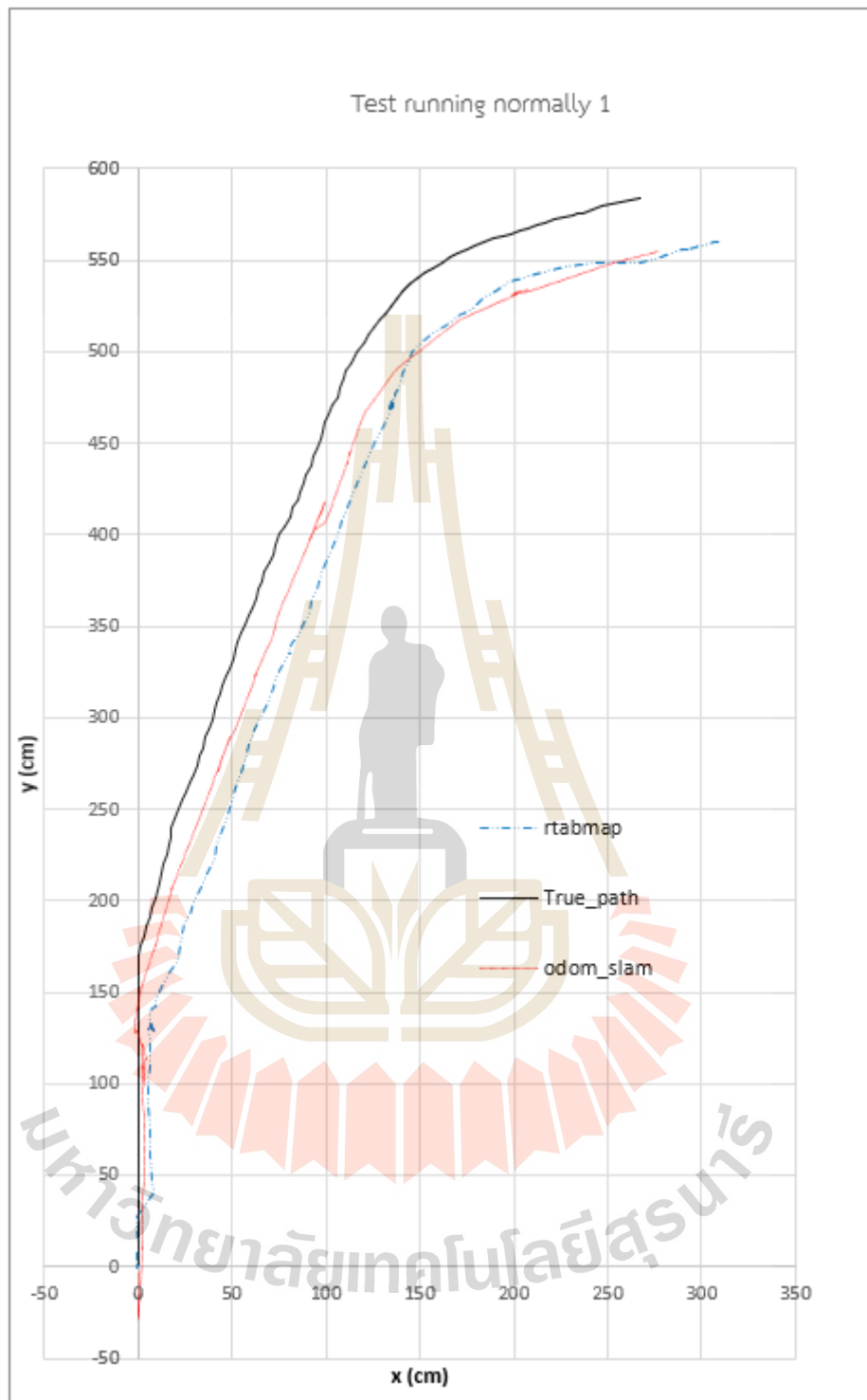


Figure 4.3 Comparing LiDAR SLAM and RTAB-Map Against a Ground-Truth Path test 1

Table 4.1 Experimental Localization Data and Absolute Error Measurements test 1

true path		rtab_odom		odom_slam		Error	Error
X	Y	x	y	x	y	rtab	odom_slam
0.0	0.0	0.0	0.0	0.0	-3.3	0.0	3.3
0.0	10.0	0.3	10.6	2.2	10.1	0.7	2.2
0.0	40.0	7.9	40.6	2.7	38.8	8.0	3.0
0.0	90.0	5.8	88.3	2.3	90.7	6.0	2.4
0.0	170.0	11.1	150.5	-2.0	133.2	22.4	36.8
11.0	210.0	29.2	199.5	18.5	208.0	21.0	7.8
40.0	299.0	59.1	285.8	50.9	292.0	23.2	13.0
61.0	360.0	82.7	340.9	71.7	345.9	28.9	17.7
88.0	428.0	108.4	407.6	99.8	418.2	28.9	15.3
106.0	475.0	126.2	452.7	118.4	461.7	30.1	18.2
129.0	517.0	146.0	500.1	137.0	489.9	24.0	28.2
175.0	556.0	183.9	529.6	207.1	533.8	27.8	39.1
208.0	567.5	227.3	546.9	247.4	546.1	28.2	44.8
247.0	579.5	275.7	550.5	253.1	548.1	40.8	32.0
267.0	584.0	310.2	560.6	277.7	554.9	49.2	31.0

Table 4.2 Localization Error Summary Table test 1

Metric	rtab_odom	odom_slam
Max Error	49.18	44.83
Mean Error	22.61	19.65
RMSE	26.31	24.23

In the first test, the robot motion involved two turns. The first turn had a relatively straight trajectory. From the graph, it can be observed that both paths show accumulated error over distance. The red and blue lines represent odom_slam and rtab_odom respectively. At 90 cm on the y-axis, from the table, rtab has an error of 6

cm, while odom_slam has only 2.4 cm error. This indicates that rtab_odom begins to deviate earlier compared to odom_slam when compared with the true path.

In the next part of the test, which is a straight path followed by a second turn, error accumulation gradually increases until it becomes more significant. From Table 4.2, it can be seen that the Mean Error and RMSE of odom_slam are lower than those of rtab_odom.

The primary source of localization error originates from cumulative drift due to odometry integration, exacerbated during rotational motion where orientation uncertainty propagates into significant positional deviation. The RTAB-Map (depth-only) configuration exhibits earlier divergence from ground truth due to weaker geometric constraints and delayed loop closure correction, while the odom_slam approach maintains lower overall error due to more stable pose estimation and reduced reliance on post-hoc corrections.

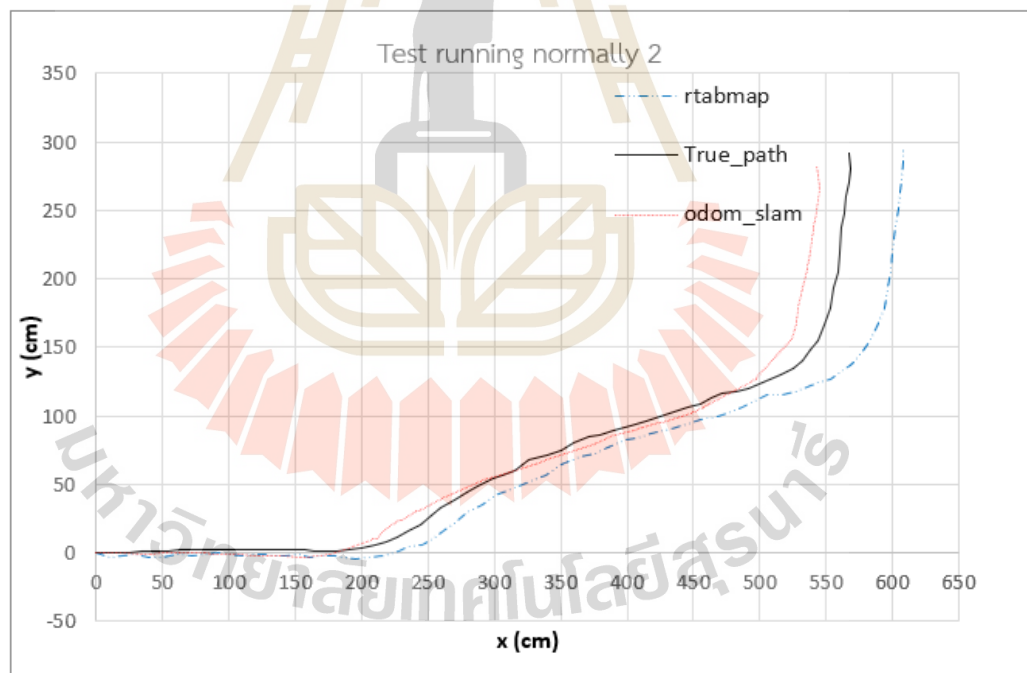


Figure 4.4 Comparing LiDAR SLAM and RTAB-Map Against a Ground-Truth Path test 2

Table 4.3 Experimental Localization Data and Absolute Error Measurements test 2

true path		rtab_odom		odom_slam		Error	Error
X	Y	x	y	x	y	rtab	odom_slam
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
124.0	2.0	127.1	-0.9	121.0	1.8	4.2	3.1
172.0	1.0	195.1	-3.8	176.5	-0.6	23.6	4.8
227.0	11.0	235.0	4.7	217.2	16.7	10.2	11.3
270.0	39.0	279.8	29.6	262.4	40.7	13.6	7.8
287.0	49.0	290.4	34.9	295.0	54.3	14.5	9.6
371.0	85.0	376.4	72.1	374.9	79.3	13.9	7.0
415.0	96.0	408.0	83.5	407.5	90.3	14.3	9.5
455.0	109.0	486.9	106.3	453.4	104.3	32.0	4.9
525.0	135.0	577.1	146.2	523.4	154.5	53.3	19.6
556.0	193.0	599.2	206.0	535.6	207.5	45.1	25.0
562.0	237.0	604.7	251.6	540.9	240.1	45.1	21.3
567.0	292.0	608.2	296.3	542.7	283.2	41.5	25.8

Table 4.4 Localization Error Summary Table test 2

Metric	rtab_odom	odom_slam
Max Error	53.26	25.84
Mean Error	20.75	9.98
RMSE	27.23	13.18

In Test 2, localization accuracy remains stable during the initial straight trajectory, where both methods exhibit low error due to minimal orientation change and consistent sensor observations. However, significant divergence occurs during the final turning segment, where orientation errors propagate into large positional deviations. The RTAB-Map approach shows substantial drift due to its reliance on weak geometric constraints and lack of immediate correction mechanisms, resulting in rapid

error accumulation. In contrast, the odom_slam method maintains significantly lower error across all metrics, indicating more robust pose estimation and better handling of rotational motion. The results highlight that turning maneuvers are the dominant source of localization error, especially for loosely coupled and depth-dependent SLAM systems.

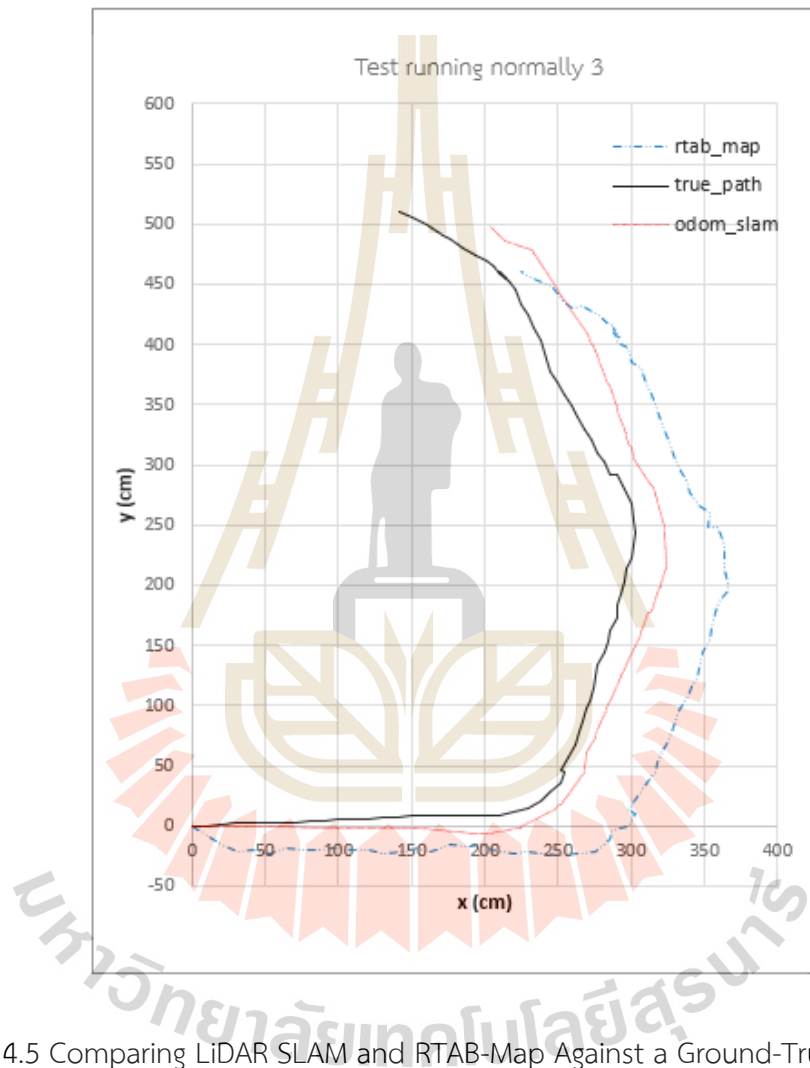


Figure 4.5 Comparing LiDAR SLAM and RTAB-Map Against a Ground-Truth Path test 3

In Test 3, the trajectory consists of a continuous curved path, which introduces sustained orientation changes and eliminates stable motion segments. This results in significant error accumulation for both methods, particularly due to orientation error propagation. The RTAB-Map (depth-only) approach exhibits severe drift, with error increasing exponentially over time due to weak geometric constraints and lack of effective correction mechanisms such as loop closure. In contrast, the

odom_slam method demonstrates comparatively stable performance, maintaining lower error despite degradation. The results highlight that continuous curvature represents a worst-case scenario for loosely coupled and depth-dependent SLAM systems, where small orientation errors compound rapidly into large positional deviations.

Table 4.5 Experimental Localization Data and Absolute Error Measurements test 3

true path		rtab_odom		odom_slam		Error	Error
x	y	x	y	x	y	rtab	odom_slam
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
40.0	2.0	41.6	-19.2	52.1	-0.3	21.2	12.3
120.0	6.0	130.1	-23.1	130.2	-1.8	30.9	12.8
190.0	8.5	219.7	-22.5	192.6	-6.0	43.0	14.8
238.0	21.0	285.0	-10.8	248.0	14.3	56.7	12.0
254.0	45.0	311.5	37.7	268.8	48.1	57.9	15.1
272.0	104.0	332.2	96.0	283.4	97.2	60.7	13.3
273.0	193.0	354.7	166.0	316.8	188.0	86.1	44.1
303.0	244.0	358.7	248.2	322.5	244.9	55.9	19.5
263.0	340.0	321.5	331.8	296.2	326.7	59.1	35.8
239.0	404.0	314.5	357.5	274.3	398.5	88.7	35.7
225.0	433.0	288.5	409.2	252.5	440.3	67.8	28.4
142.0	510.0	255.2	460.6	203.3	497.8	123.5	62.5

Table 4.6 Localization Error Summary Table test 3

Metric	rtab_odom	odom_slam
Max Error	123.48	62.55
Mean Error	50.09	20.42
RMSE	60.68	26.68

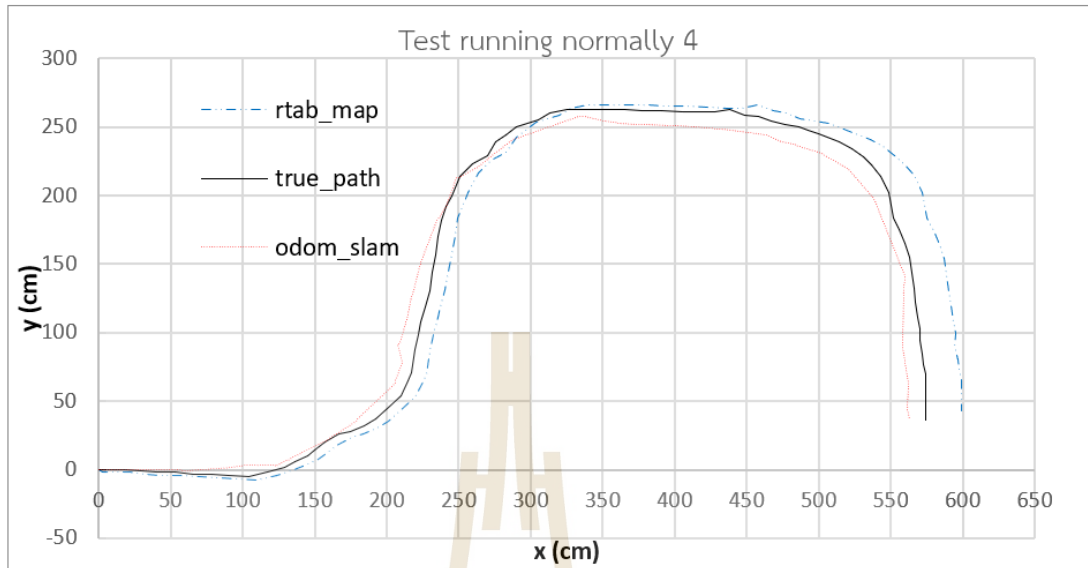


Figure 4.6 Comparing LiDAR SLAM and RTAB-Map Against a Ground-Truth Path test 4

Table 4.7 Experimental Localization Data and Absolute Error Measurements test 4

true path		rtab_odom		odom_slam		Error	Error
X	Y	x	y	x	y	rtab	odom_slam
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
30.0	-1.0	33.2	-3.0	31.2	0.0	3.8	1.5
91.0	-4.0	84.2	-5.1	93.3	2.1	6.9	6.6
129.0	2.0	123.2	-4.3	129.5	6.5	8.6	4.5
145.0	10.0	143.5	3.5	134.6	9.3	6.7	10.4
175.0	10.0	145.6	23.9	176.1	33.4	32.5	23.4
200.0	45.0	201.0	35.8	203.9	60.7	9.2	16.2
234.0	157.0	243.0	144.9	224.4	153.3	15.1	10.3
245.0	200.0	252.9	194.1	248.7	213.0	9.8	13.5
336.0	263.0	339.9	263.3	333.5	257.6	3.9	6.0
486.0	250.0	487.4	256.3	483.3	237.2	6.4	13.1
543.0	214.0	566.2	214.8	522.4	217.7	23.2	21.0
552.0	184.0	575.5	183.6	539.9	193.8	23.5	15.6
574.0	36.0	599.4	42.8	562.9	37.4	26.3	11.2

Table 4.8 Localization Error Summary Table test 4

Metric	rtab_odom	odom_slam
Max Error	32.54	23.41
Mean Error	11.73	10.21
RMSE	15.24	12.35

Test 4, both localization methods demonstrate improved performance due to the smoother trajectory and reduced motion complexity compared to previous tests. Error remains relatively low during the initial straight segment and increases gradually along the curved path. The RTAB-Map (depth-only) approach exhibits a consistent outward drift, indicating systematic bias in pose estimation likely due to accumulated orientation error and weaker geometric constraints. The odom_slam method maintains slightly better accuracy, with lower mean error and RMSE, suggesting more stable handling of gradual curvature. The results indicate that while both methods perform adequately under moderate motion conditions, turning segments still contribute to error accumulation, though to a lesser extent than in more complex trajectories.

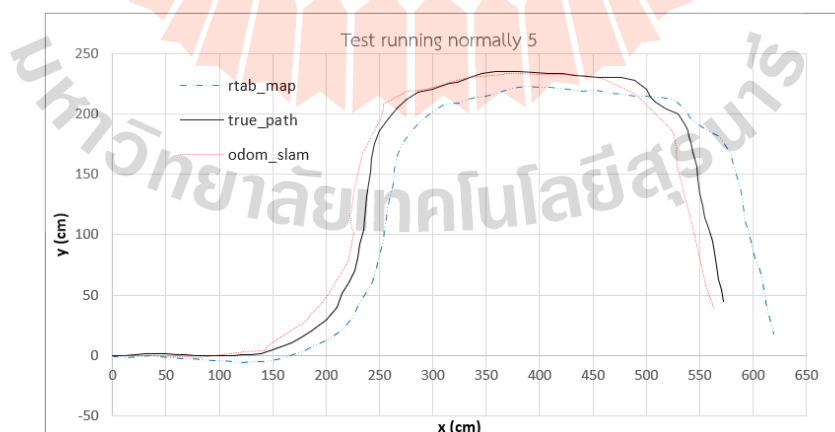


Figure 4.7 Comparing LiDAR SLAM and RTAB-Map Against a Ground-Truth Path test 5

Table 4.9 Experimental Localization Data and Absolute Error Measurements test 5

true path		rtab_odom		odom_slam		Error	Error
X	Y	x	y	x	y	rtab	odom_slam
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
60.0	1.0	65.0	-2.3	65.7	-1.1	6.0	6.1
113.0	0.0	112.0	-4.7	117.0	2.2	4.8	4.6
176.0	15.0	189.1	7.8	175.6	25.2	14.9	10.2
215.0	51.0	231.0	42.5	199.7	48.5	18.1	15.5
227.0	70.0	247.5	72.2	221.5	79.9	20.6	11.4
237.0	122.0	256.3	113.0	224.0	121.8	21.3	13.0
242.0	152.0	264.1	153.4	233.9	166.9	22.1	17.0
267.0	205.0	280.8	186.3	253.5	207.7	23.2	13.8
300.0	220.0	310.6	207.7	290.1	219.2	16.2	9.9
335.0	230.0	349.8	214.3	334.8	229.9	21.6	0.3
408.0	233.0	439.4	219.0	406.8	232.5	34.4	1.3
433.0	232.0	465.9	217.4	438.6	231.8	36.0	5.6
476.0	230.0	500.5	214.4	458.1	229.3	29.1	17.9
530.0	200.0	569.0	181.4	522.5	188.0	43.2	14.2
547.0	157.0	588.7	135.6	529.0	157.4	46.9	18.0
562.0	95.0	602.1	80.8	544.7	101.6	42.6	18.5
572.0	45.0	619.1	17.8	562.9	38.9	54.4	11.0

Table 4.10 Localization Error Summary Table test 5

Metric	rtab_odom	odom_slam
Max Error	43.20	17.89
Mean Error	20.77	9.38
RMSE	29.25	12.06

In Test 5, both localization methods perform well during the initial straight and moderately curved segments, maintaining low error due to stable motion

and consistent sensor observations. However, significant divergence occurs during the final trajectory segment, where a change in direction introduces increased orientation uncertainty. The RTAB-Map (depth-only) approach exhibits substantial drift, driven by accumulated error and limited correction capability, particularly during motion transitions. In contrast, the odom_slam method maintains lower error across all metrics, demonstrating more robust handling of orientation changes and improved trajectory consistency. These results reinforce that directional transitions and accumulated drift are key contributors to localization error, especially in loosely coupled and depth-dependent systems.

Table 4.11 Comparative Error Analysis of RTAB-Map and Odom_SLAM over Five test Runs

	cm									
	test 1		test 2		test 3		test 4		test 5	
	rtab	odom_slam	rtab	odom_slam	rtab	odom_slam	rtab	odom_slam	rtab	odom_slam
Max Error	49.18	44.83	53.26	25.84	123.48	62.55	32.54	23.41	43.20	17.89
Mean Error	22.61	19.65	20.75	9.98	50.09	20.42	11.73	10.21	20.77	9.38
RMSE	26.31	24.23	83.51	24.23	83.51	24.23	15.24	12.35	29.25	12.06

The figure test running normally shows three path true_path (black solid line): will be the reference trajectory along with odom_slam (red dotted line) LiDAR SLAM estimate and finally rtabmap (orange dashed dotted line) RTAB-Map estimate in the unit of centimeter.

Overall, both SLAM trajectories follow the same global shape as the true path, meaning both systems successfully track the robot's general motion trend. However, both estimated paths sit consistently below the true path for a large portion of the run, indicating a systematic offset rather than random noise. Odometry data can be noisy, particularly in dynamic or unstructured environments. This noise impacts the accuracy of the SLAM system, making it difficult for the robot to maintain a reliable and consistent map of its surroundings (Davison, 2003)

Table 4.11 presents the quantitative comparison of localization performance between RTAB-Map and Odom_SLAM over five experimental trials, evaluated using maximum error, mean error, and RMSE in centimeters. Across most test scenarios, Odom_SLAM demonstrates lower mean error and RMSE values

compared to RTAB-Map, indicating more consistent trajectory estimation. In particular Test 4 and Test 5 show noticeably smaller RMSE values for Odom_SLAM (12.35 cm and 12.06 cm, respectively) compared to RTAB-Map (15.24 cm and 29.25 cm). Mean error values follow a similar trend, suggesting that Odom_SLAM maintains closer alignment with the ground truth trajectory in these trials. This suggests that in moderately structured environments or under relatively stable motion conditions, the LiDAR-based SLAM approach provides robust incremental pose estimation.

However, the results also reveal variability in performance. In Test 3, RTAB-Map exhibits a significantly higher maximum error (123.48 cm) and RMSE (83.51 cm), indicating a substantial deviation from the reference trajectory during that run. These deviations imply that RTAB-Map may be more sensitive to environmental characteristics such as feature sparsity, motion irregularities, or sensor noise.

4.1.2 Loop Closure Experimental Results

In the upper image (before loop matching), slight misalignment can be observed in the highlighted region. As the robot moves along the path, small pose estimation errors accumulate due to incremental scan matching and odometry integration. These errors result in a minor positional offset between newly acquired LiDAR scans and the previously constructed map. Once the robot revisits the same physical area, the SLAM system identifies similarity between the current scan and earlier observations. This triggers loop closure detection, where an additional constraint is introduced into the pose graph. After this constraint is incorporated, global optimization is performed to reduce accumulated drift.

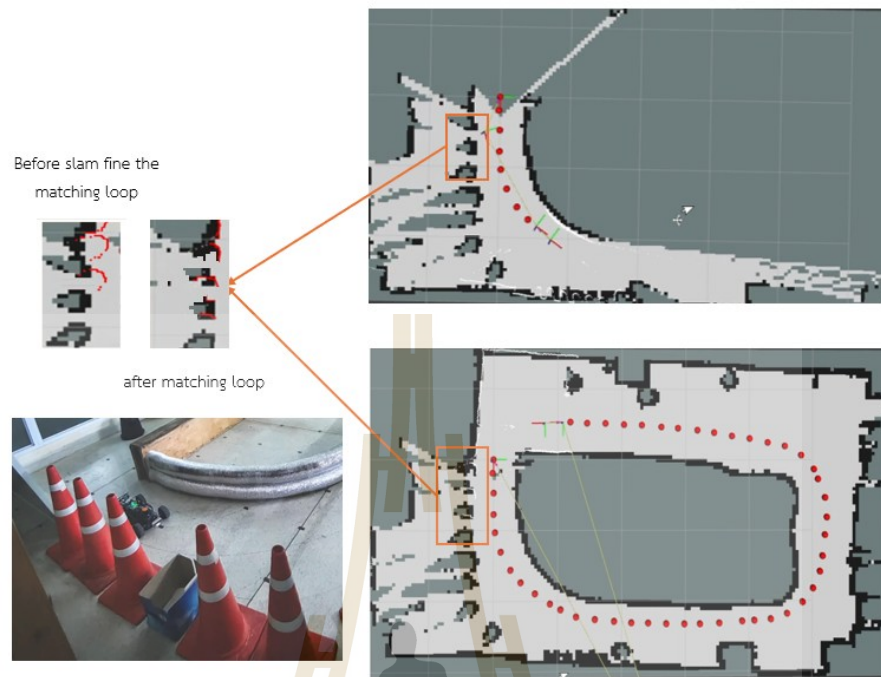


Figure 4.8 presents the loop closure performance of the SLAM system during a closed loop.

The lower image (after loop matching) demonstrates improved alignment within the highlighted region. Structural features, such as wall boundaries and corners, show better registration compared to the pre-closure state. Rather than correcting only the final pose, the optimization process redistributes accumulated error across the entire trajectory, resulting in a more globally consistent map.

These results confirm that the SLAM framework successfully detects revisited locations and applies pose graph optimization to mitigate long-term drift. Although minor residual discrepancies may remain, the overall structural consistency of the map is significantly improved following loop closure.

4.1.3 Robustness Under Camera Failure During Operation

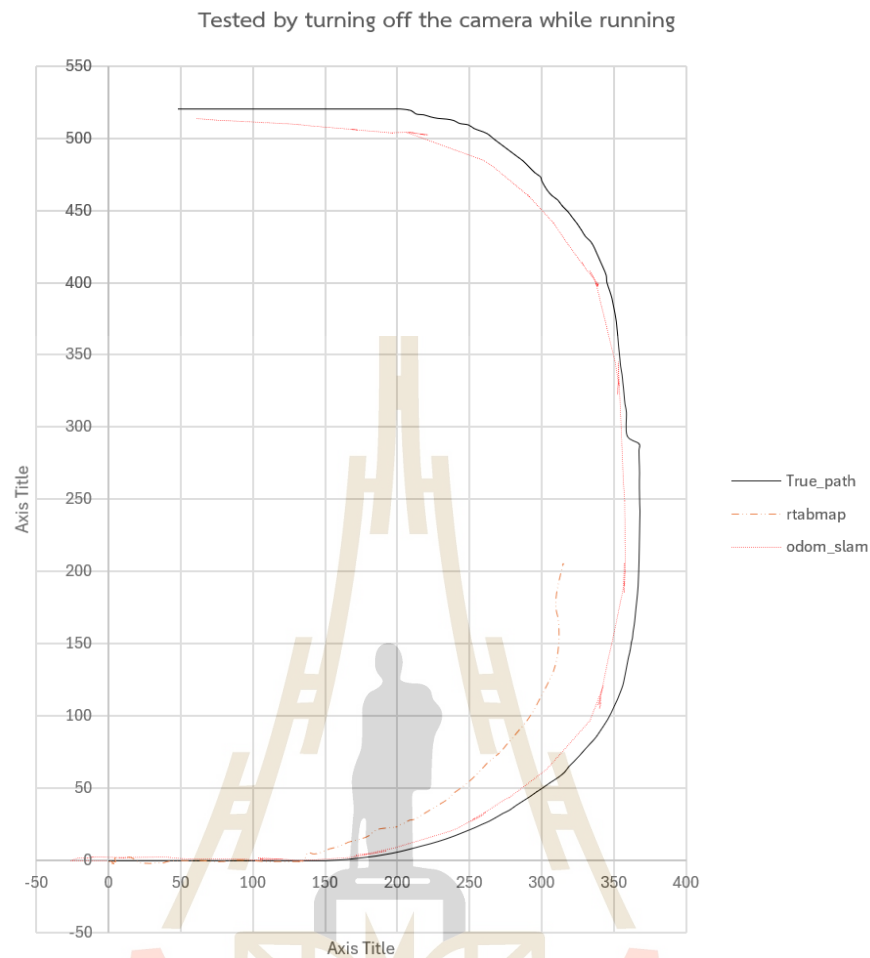


Figure 4.9 Robustness Under Camera Failure During Operation

The second experiment investigates system robustness when a critical sensor becomes unavailable during active operation. In practical robotic deployments, vision-based sensors are prone to failure due to lighting changes, occlusions, hardware instability, or communication interruptions. Therefore, this experiment intentionally disables the camera while the robot is in motion to evaluate how different SLAM approaches respond to sensor loss and whether they can maintain localization and recover mapping functionality once the camera input is restored. This experiment compares the behavior of RTAB-Map SLAM, which relies heavily on visual or depth data for perception and pose estimation and the proposed system, which separates responsibilities by using LiDAR-based SLAM for localization and a depth camera primarily for mapping.

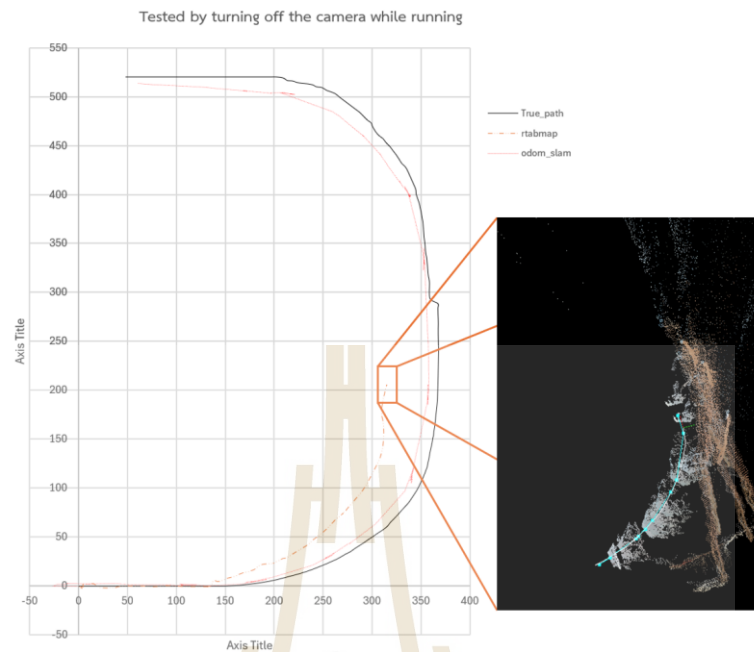


Figure 4.10 Robustness Under Camera Failure During Operation During the Moment of camera failure

When the camera is turned off during operation, the plotted trajectories show a clear difference between the two approaches. The true path continues smoothly, representing the actual motion of the robot. The LiDAR-based localization component in the proposed system continues to estimate the trajectory, whereas the RTAB-Map trajectory stops updating or diverges significantly from the true path. After the camera is turned back on, the proposed system resumes generating point cloud data while maintaining a continuous and stable trajectory. This indicates that localization remains uninterrupted throughout the sensor failure, and mapping functionality is restored once depth data becomes available again.

RTAB-Map relies on visual and depth observations to generate odometry constraints and to expand its pose graph. When the camera input is removed, the system loses its primary source of environmental information. As a result, RTAB-Map is unable to compute new motion estimates or register new observations, which leads to a stagnation of the estimated trajectory. This behavior is reflected in the results, where the RTAB-Map path no longer follows the true trajectory once the camera is disabled.

This outcome highlights a fundamental limitation of vision-dependent SLAM systems: without sufficient perceptual input, the system cannot maintain observability of motion. In such cases, pose estimation either halts or becomes unreliable, particularly if no alternative odometry source is available to compensate for the missing sensor data.

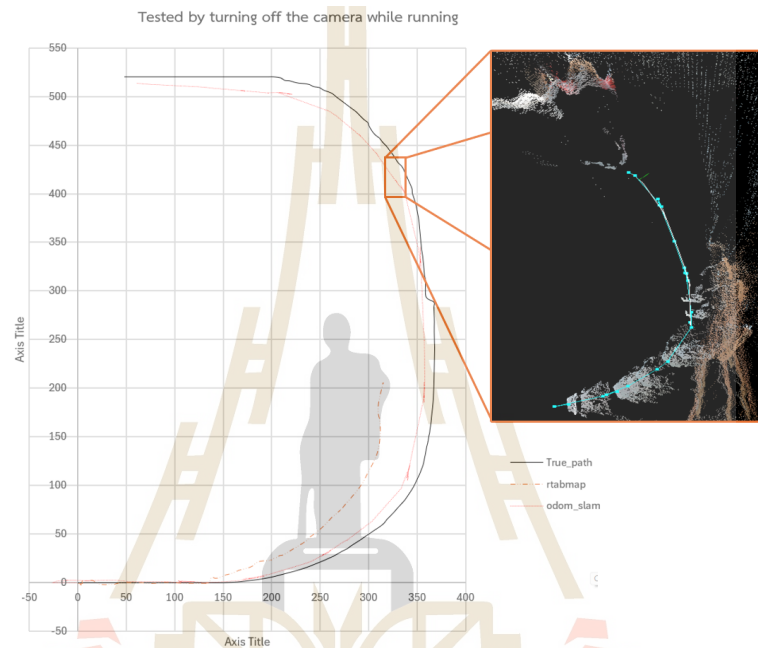


Figure 4.11 The system back on, the propose system resumes generating point cloud data.

In contrast, the proposed system maintains localization throughout the experiment because pose estimation is handled independently by LiDAR SLAM. LiDAR-based localization relies on geometric features extracted from laser scans, such as walls and structural edges, which remain observable even when the camera is inactive. As long as the surrounding environment provides sufficient geometric structure, the LiDAR SLAM module can continue matching scans and estimating motion.

This separation between localization and mapping allows the system to remain operational despite the loss of visual perception. While depth-based mapping temporarily pauses due to the absence of camera input, the robot's pose estimation remains consistent and continuous.

Once the camera is re-enabled, the proposed system immediately resumes generating point cloud data from the depth sensor. Because the localization pipeline never stopped, newly acquired depth measurements can be correctly associated with the current robot pose. This results in a coherent continuation of the map without introducing large discontinuities or misalignments.

The absence of abrupt pose jumps or distorted point clouds suggests that the coordinate frame relationships remain stable throughout the experiment. This indicates effective synchronization between LiDAR-based localization and camera-based mapping once sensor input is restored.

The results of this experiment demonstrate that tightly coupled vision-based SLAM systems are more vulnerable to sensor dropout during motion. When visual data is unavailable, pose estimation may fail entirely, preventing continued navigation. In contrast, the proposed architecture improves fault tolerance by decoupling localization from visual perception. By relying on LiDAR for pose estimation and using the depth camera as a supplementary mapping sensor, the system can tolerate temporary camera failures without losing localization.

Such robustness is particularly important for real-world robotic applications, where uninterrupted sensor availability cannot always be guaranteed. Experiment 2 confirms that disabling the camera during operation causes RTAB-Map SLAM to lose its ability to update the robot trajectory, resulting in a stalled or inaccurate pose estimate. The proposed system, however, continues to operate due to its reliance on LiDAR-based localization. When camera input is restored, mapping resumes seamlessly without disrupting localization. These results demonstrate that separating localization and mapping across different sensor modalities enhances system reliability under sensor failure conditions.

CHAPTER V

CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

This thesis investigated the design and evaluation of a robust indoor SLAM framework that integrates LiDAR-based localization with RGB-D-based mapping on the F1 TENTH autonomous platform. The main goal of the study was to enhance localization reliability and maintain map continuity in indoor environments where visual sensors alone may be affected by lighting conditions, occlusions, or temporary sensor failure.

The experimental results presented in Chapter 4 compared two SLAM approaches: a LiDAR-only SLAM system and an RGB-D SLAM system implemented using RTAB-Map. Both systems were evaluated against a predefined ground-truth trajectory. Under normal operating conditions, the results showed that each method was capable of tracking the robot's motion; however, trajectory deviations became more pronounced during extended motion and curved path segments. These errors reflect the cumulative nature of localization uncertainty and the limitations of relying on a single sensing modality.

A robustness evaluation was conducted by disabling the camera during runtime to simulate a loss of visual information. In this scenario, the RGB-D SLAM system was unable to continue updating its trajectory and mapping process due to the absence of visual features. In contrast, the proposed system maintained stable localization using LiDAR data, allowing the robot to continue navigating the environment. When the camera was reactivated, dense map generation resumed without requiring system reinitialization. This behavior demonstrates the advantage of decoupling localization and mapping tasks across different sensors.

The findings indicate that LiDAR provides consistent and reliable localization in feature-degraded environments, while RGB-D sensors significantly enhance environmental representation when visual data is available. By combining these complementary sensing capabilities, the proposed system achieves greater robustness and operational continuity compared to single-sensor SLAM approaches.

In summary, this research confirms that a hybrid SLAM architecture is effective for indoor autonomous navigation, particularly in scenarios involving sensor degradation or partial sensor failure. The proposed approach improves system resilience and mapping reliability, making it suitable for real-world robotic applications. Future work may extend this framework through tighter sensor fusion, dynamic weighting strategies, and validation in more complex and dynamic indoor environments.

5.2 RECOMMENDATION

5.2.1 Due to the relatively high operating speed of the vehicle, localization performance may be negatively affected. Rapid motion increases the difficulty of accurately estimating the robot's position, as sensor measurements and odometry updates may not be captured with sufficient temporal resolution, leading to increased localization error.

5.2.2 A higher-performance processing board is recommended to improve system responsiveness and computational efficiency. Enhanced processing capability would allow faster sensor data handling, more stable SLAM computation, and reduced latency during real-time operation.

REFERENCES

- G. Halmetschlager-Funek, M. Suchi, M. Kampel, and M. Vincze, "An empirical evaluation of ten depth cameras: Bias, precision, lateral noise, different lighting conditions and materials, and multiple sensor setups in indoor environments," *IEEE Robotics & Automation Magazine*, vol. 25, no. 3, pp. 1–1, Aug. 2018, doi: 10.1109/MRA.2018.2852795.
- P. Chen, X. Zhao, L. Zeng, L. Liu, and S. Liu, "A review of research on SLAM technology based on the fusion of LiDAR and vision," *Sensors*, vol. 25, no. 5, p. 1447, 2025, doi: 10.3390/s25051447.
- J. Gené-Mola, J. Llorens, J. R. Rosell-Polo, and E. Gregorio, "Assessing the performance of RGB-D sensors for 3D fruit crop canopy characterization under different operating and lighting conditions," *Sensors*, vol. 20, no. 24, p. 7072, 2020, doi: 10.3390/s20247072.
- Drone For Inspecting Inside Building With Object Avoidance And Mapping System (Pawarut Karaked, 2023)
- T. Shan and B. Englot, "LIO-SAM: Tightly-Coupled Lidar Inertial Odometry via Smoothing and Mapping," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5135–5142.
- Kirchner, N., & Furukawa, T. (2005). Infrared localization for indoor uavs. Paper presented at the Proc. Of the 1st Int. Conf. on Sensing Technology
- Y. Lu, J. Xue, G. Xia, and L. Zhang, "A survey on vision-based UAV navigation," *Journal of Field Robotics*, 21(1), 21-32
- Mourikis, A. I., & Roumeliotis, S. I. (2007). A multi-state constraint Kalman filter for vision-aided inertial navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3565–3572).

Development of Autonomous Electric Golf Cart System for Passenger Transportation Between Building (Wongsathon Angkhem, 2023)

Zunaidi, I., Kato, N., Nomura, Y., & Matsui, H. (2006). Positioning system for 4-wheel mobile robot: Encoder, gyro and accelerometer data fusion with error model method. *CMU Journal*, 5(1), 1–14.

M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Y. Ng, “ROS: An open-source Robot Operating System,” in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Software*, Kobe, Japan, 2009.

ROS.(2024). REF-0105: Standard Unit of Measure and Coordinate Conventions

REP 103, Standard Unit of Measure and Coordinate Conventions

Foote, T. (2013). tf: The transform library. Paper presented at the 2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)

Thrun, S. (2007). Simultaneous localization and mapping. In *Robotics and cognitive approaches to spatial mapping* (pp. 13-41): Springer.

Thrun, S., Fox, D., Burgard, W., & Dellaert, F. (2001). Robust Monte Carlo localization for mobile robots. *Artificial intelligence*, 128 (1-2), 99-141.

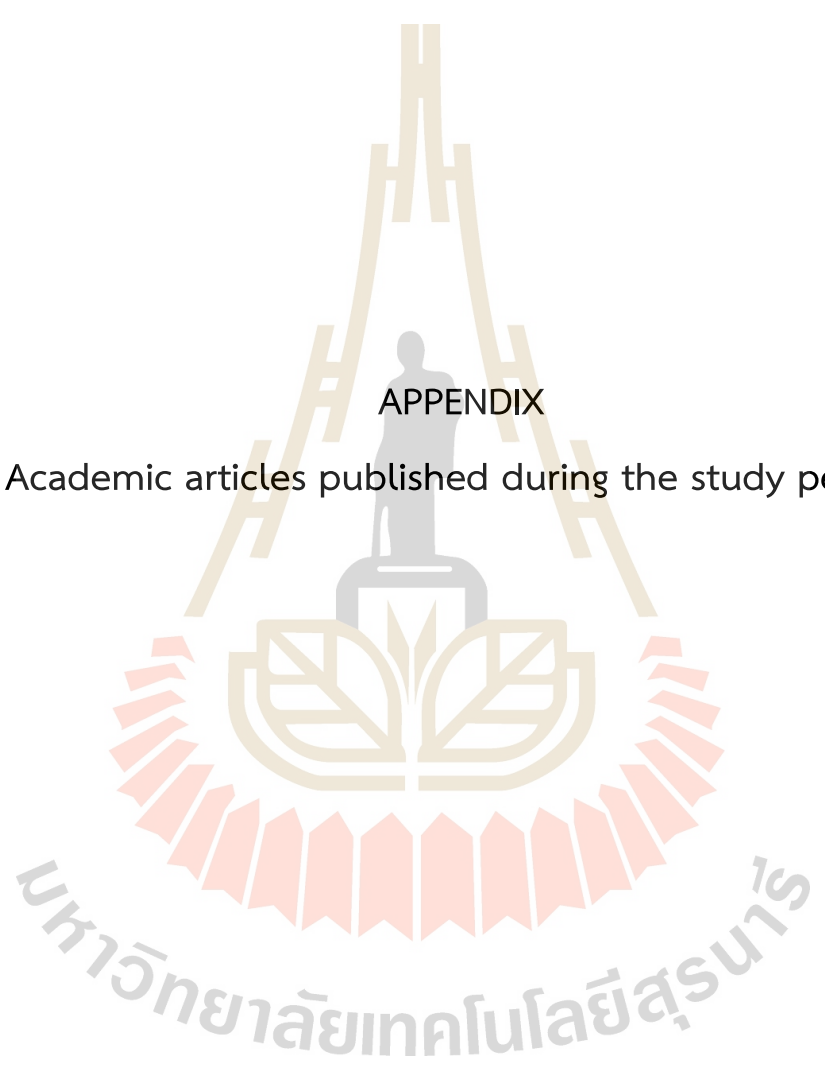
Grisetti, G., Kummerle, R., Stachniss, c., & Burgard, W. (2010). A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4), 31-43

Davison. (2003, October). Real-time simultaneous localization and mapping with single camera. In *proceedings Ninth IEEE International Conference on Computer Vision* (pp. 1403-1410). IEEE.

Berka, Z. (2020). *Mapping road makings and localization of F110 autonomous car using camera*. Retrieved from https://people.ciirc.cvut.cz/~klapajar/studenti/F3-DP-2020-Berka-Zdenek-Mapping_road_makings_and_localization_of_f110_autonomous_car_using_camera.pdf

- Saeefi. S., Liu, L., & Lee, S. (2020). *A review of SLAM algorithms and their applications*. *Journal of Open Source Software*, 5(52), 2783. <https://doi.org/10.21105/joss.02783>
- [22] A. Zaarane, I. Slimani, W. Al Okaishi, I. Atouf, and A. Hamdoun, "Distance measurement system for autonomous vehicles using stereo camera," *Array*, vol. 5, p. 100016, 2020, doi: 10.1016/j.array.2020.100016.



The logo of Sakon Nakhon Rajabhat University is a large, faint watermark in the background. It features a central figure of a person standing on a pedestal, holding a book. Above the figure is a stylized 'A' shape composed of multiple 'H' characters. Below the figure is a circular emblem with a book and a sunburst. The entire logo is surrounded by a ring of red and orange triangles. The name of the university is written in Thai script at the bottom.

APPENDIX

Academic articles published during the study period.

List of Publications

Natthawut Pidech, Tharathep Phiboom, Suradet Tantriratn, Auraluck, Atthaphon Ariyarat
(2024). Enhancing SLAM on F1TENTH Using Data Fusion Kalman Filter Algorithm.
14th International Conference on Mechanical Engineering (ICOME), 10Th -13th
December 2024 at Pattaya Thailand.



DRC0011

Enhancing SLAM on FITENTH Using Data Fusion Kalman Filter Algorithm

Natthawut Pidech¹, Tharathep Phiboon¹, Suradet Tantriratn¹, Auraluck¹, and Atthaphon Ariyarat^{1,*}

¹School of Mechanical Institute of Engineering Suranaree University of Technology, 111, Maha Witthayalai Rd, Suranari Mueang Nakhon Ratchasima District Nakhon Ratchasima, 30000, Thailand

* Corresponding Author: ariyarat@sut.ac.th

Abstract. In the context of autonomous vehicles, localization is complicated to achieve the real-time Simultaneous Localization and Mapping (SLAM). Originally, the SLAM process updates a map of an unknown environment and keeps tracking its location within that environment simultaneously by operating coordinate frames convention of map, odom and base_link frames. The conventional SLAM systems suffer the accumulation of errors over a time, it leads significant inaccuracies. Hence, this work purposed to improve the SLAM abilities of a one-tenth scale (FITENTH) autonomous racing car and developed the method using the data fusion between the Inertial Measurement Unit (IMU), and the teleoperation information of the vehicle from the VESC Motor and Servo Controller which provided information to the base_link for better localization to enhance the SLAM accuracy. To combine data fusion, this work utilized an extended Kalman filter (EKF). The Kalman filter algorithm is a kind of model that is widely used in state of estimation problems where noise and uncertainties occur. The EKF is used for the non-linearities in the dynamic system and measurement models. Thereby, the SLAM performance improved mean absolute deviation 40 percent compared with VESC as source of odometry. The basic findings prove that the localization error decreases, and the precision of map construction increases. Additionally, the robust navigation system is also enhanced.

Keywords: Simultaneous Localization and Mapping SLAM, Localization, Data fusion, FITENTH, Extended Kalman Filter EKF

1. Introduction

Simultaneous Localization and Mapping (SLAM) is a critical technology in robotics and autonomous systems, enabling a robot to build a map of an unknown environment while simultaneously tracking its own location. SLAM's effectiveness relies heavily on the accuracy of the odometry data, which provides estimates of the robot's motion based on sensor inputs such as controller.

However, odometry data is not without its challenges. Odometry errors accumulate over time, leading to drift and inaccuracies in the robot's estimated trajectory [1,2]. This drift results in a growing discrepancy between the robot's estimated position and its actual location, which can significantly

degrade the quality of the SLAM solution. Moreover, odometry data can be noisy, particularly in dynamic or unstructured environments. This noise impacts the accuracy of the SLAM system, making it difficult for the robot to maintain a reliable and consistent map of its surroundings [3].

IMU data provides information on orientation, acceleration, and angular velocity, while odometry offers position estimates. Combining these sources improves the overall accuracy of the state estimation by integrating complementary data and EKF helps in reducing drift by continually updating the state estimate with measurements from both odometry and IMU, leading to more accurate trajectory estimation over time.

Our research focuses on implementing this data fusion strategy on the FITenth platform, seeking to achieve more accurate and reliable SLAM performance. By combining odometry with IMU data and refining the Kalman filter approach, we aim to overcome the limitations of traditional odometry and enhance the overall effectiveness of the SLAM system.

2. FITENTH

This system has been developed on FITENTH autonomous. The FITENTH features standardized equipment and is specifically designed for competitions involving autonomous vehicles systems. The lists of the relevant hardware and software equipment for FITENTH are shown in Table 1.

Table 1. List of relevant hardware and software equipment.

Equipment	Module
IMU	Internal VESC 6MkV
LIDAR	Hokuyo UST-10LX
Computing module	NVIDIA Jetson NX
Motor Controller	VESC 6MkV
OS	ROS2 Foxy (under Ubuntu Linux 20.04)

2.1. VESC 6MkV

The VESC 6Mkv is an advanced electronic speed controller that interfaces with various sensor types. It provides critical data for navigation and control systems through its integration with sensors. Specifically, it outputs information via the '/odom' topic, to compute the vehicle's position and orientation. The integration process involves position update compute the change in position ' $(\Delta x, \Delta y)$ ' using velocity v and orientation θ as in Eq. (1), (2).

$$\Delta x = v \cdot \cos(\theta) \cdot \Delta t \quad (1)$$

$$\Delta y = v \cdot \sin(\theta) \cdot \Delta t \quad (2)$$

Update the vehicle's orientation based on the angular velocity from Eq. (3) and the new position x_{new}, y_{new} is computed by adding the changes in position to the previous position x_{old}, y_{old} as Eq. (4), (5).

$$\theta_{new} = \theta_{old} + \omega \cdot \Delta t \quad (3)$$

$$x_{new} = x_{old} + \Delta x \quad (4)$$

$$y_{new} = y_{old} + \Delta y \quad (5)$$

The node publishes the computed odometry information as a `nav_msgs/Odometry` message on the '/odom' topic this message includes `geometry_msgs/Pose` message containing the position (x, y, z) and

orientation. The linear and angular velocities of the vehicle, published as a `geometry_msgs/Twist` message.

Additionally, it interfaces with the `/sensors/imu` topic, supplying data from the Inertial Measurement Unit (IMU). The `/vesc_driver_node` processes raw IMU data from the VESC by applying filtering techniques to reduce noise and isolating the actual linear acceleration from the total acceleration. The result is a detailed representation of the vehicle's dynamics, which is published on the `/sensors/imu` topic, allowing other components in the ROS system to use this information for tasks like localization, navigation, and control.

2.2. LiDAR

The Hokuyo UST-10LX is a compact and lightweight 2D LiDAR sensor used in F1TENTH for crucial role in SLAM by providing range measurements of the environment and to localize the robot.

2.3. ROS

The whole application is targeted at Robot Operating System (ROS), which operates on F1/10. The individual processes used in the system are represented as ROS nodes which communicate with each other via publishing and subscribing various data messages in so called topics [4]. ROS2 Foxy under Ubuntu Linux 20.04 have been used as operating system.

3. Coordinate frames convention

The Transformation Tree view frames result depicting the transformation tree illustrates the hierarchical relationship between different coordinate frames in the system. The map frame represents the global reference frame in which the robot's position is defined over time. 'odom' frame tracks the robot's movement based on the commands received from the remote control, using teleop data to update the robot's position relative to its starting point [5]. The base_link frame corresponds to the robot's body or chassis, serving as the origin for all the sensors attached to the robot [6]. Lastly, the scan frame is associated with the LiDAR, providing distance measurements relative to the base_link. This transformation chain allows for accurate localization by transforming sensor data from the scan frame to the global map frame through the intermediate odom and base_link frames, ensuring that all sensor data is consistent and aligned within a unified coordinate system.

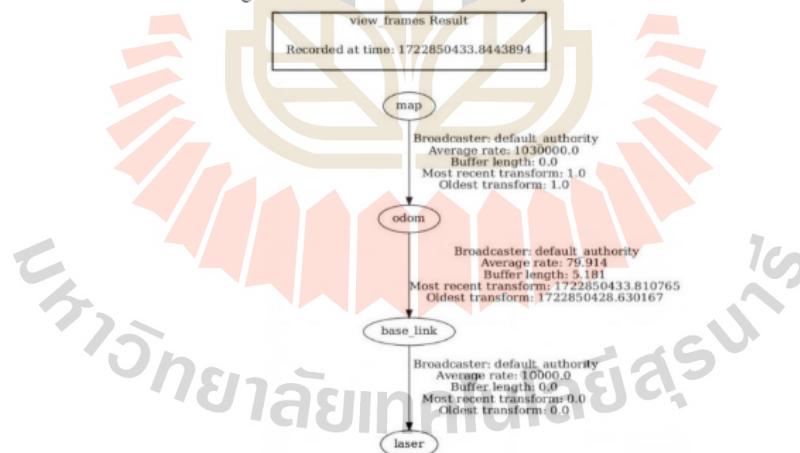


Figure 1. Transformation Tree

According to Figure 1. Map to Odometry (map $\rightarrow$ odom): Establishes where the robot is on the global map based on odometry data. This transformation accounts for drift and errors in odometry over time. Odometry to Base Link (odom $\rightarrow$ base_link): Describes the robot's position and orientation in its local frame relative to its origin. This transformation updates with every movement of the robot based on odometry. Base Link to Laser (base_link $\rightarrow$ laser): Maps the laser scanner's measurements into the robot's local frame. This is crucial for interpreting the laser data relative to the robot's position and orientation.

$$P_{map}(t) = T_{odom \rightarrow map}(t) \cdot T_{base \rightarrow odom}(t) \cdot T_{laser \rightarrow base} \cdot P_{laser} \quad (6)$$

The mathematical transformations between the frames as in Eq. (6) has involved a series of matrix multiplications that account for the robot's movement and sensor placement. These transformations allow SLAM algorithms to localize the robot in global map.

4. Simultaneous Localisation and Mapping SLAM

In this paper employed the SLAM capabilities provide by the slam_toolbox package, slam_toolbox is open-source under an LGPLv2.1 at https://github.com/SteveMacenski/slam_toolbox.git and is available in every current ROS distribution. This software package provides an approach for multi-session mapping and localization [7]. This paper operates on asynchronous mapping mode by transformation between consecutive frames. If the robot's pose at time t is represented by $T_{odom \rightarrow base}(t)$, the transformation between the robot's pose at time $t - 1$ and t is Eq. (7).

$$T_{t-1 \rightarrow t} = T_{odom \rightarrow base}(t) \cdot T_{odom \rightarrow base}(t-1)^{-1} \quad (7)$$

Mathematics of Pose Graph Optimization in Eq. (8) a graph where each node x_i represent the robot's pose, and each edge e_{ij} represent a relative transformation between two poses.

$$\min_x \sum_{i,j} \|T_{ij} - T(x_i, x_j)\|^2 \quad (8)$$

Where $T_{i,j}$ is the observed transformation and $T(x_i, x_j)$ is the estimated transformation based on the poses x_i and x_j .

5. Extended Kalman Filter (EKF)

Provides EKF and Unscented Kalman Filter (UKF) implementations for sensor fusion in ROS. It's commonly used for integrating odometry, IMU, and GPS data [8].

5.1. State Space Representation

In this application, the robot's state vector X_k typically includes its position, velocity, and orientation at time step k . The state-space model can be defined as in Eq. (9). Where: X_k is represented the state vector at time k , $f(\cdot)$ is the nonlinear state transition function, u_{k-1} represents control inputs, and W_{k-1} is the process noise, assumed to be Gaussian with covariance Q_{k-1} .

$$X_k = f(X_{k-1}, u_{k-1}) + W_{k-1} \quad (9)$$

The measurement model is represented as Eq. (10) where: Z_k is the measurement vector at time k , $h(\cdot)$ is the nonlinear measurement function, and W_{k-1} is the measurement noise, assumed to be Gaussian with covariance R_k .

$$Z_k = h(X_{k-1}) + V_k \quad (10)$$

5.2. Fusion of Odometry and IMU Data

For robot localization, data from both the odometry “/odom” and the IMU “/sensors/imu” are fused. The odometry provides velocity and position information, while the IMU contributes angular velocity and acceleration data, which are critical for determining orientation and handling dynamic movements.

6. Integration with SLAM

This study aims to enhance the SLAM localization utilizing a data fusion approach with an EKF algorithm.

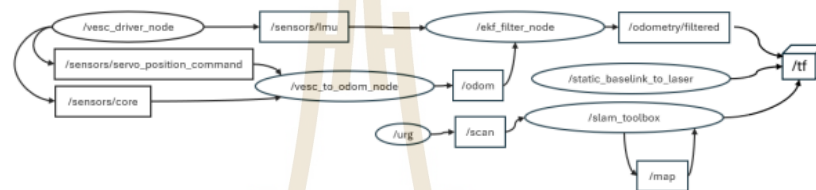


Figure 2. overall system rqt_graph

6.1. SLAM Execution and Evaluation:

Following Figure 2. After obtaining the filtered odometry data from the EKF, eventually integrate this data into the SLAM algorithm. The EKF outputs a refined odometry estimate, which combines the odometry data with suffers from the drift occurring error accumulate overtime. On the other hand, IMU offers measurements of linear acceleration and angular velocity, providing a real-time assessment of the vehicle's orientation and motion. IMU can detect rotational changes (such as heading drift) that odometry might miss or misinterpret. In enhanced process the IMU helps correct the cumulative error in the odometry data by providing an independent reference for rotational movements and accelerations.

7. Result and discussion

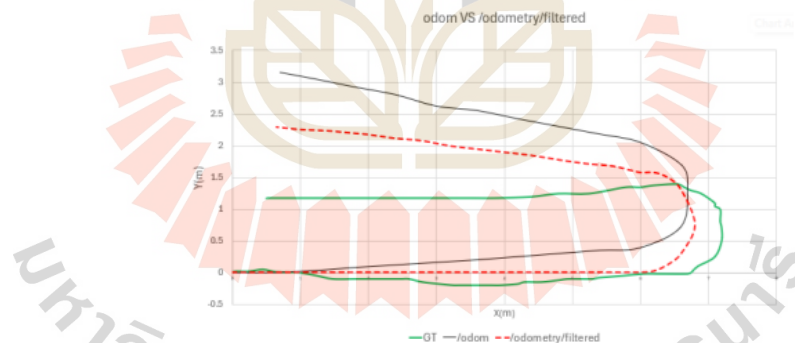


Figure 3. The figure compares the paths obtained from two data sources from /odom/filtered and /odom topic with GT (Ground Truth) /odom/filtered (Red Dashed Line) Figure 3. follows the GT path closely, with some small deviations, especially in the curved sections. The mean absolute deviation from GT for

/odometry/filtered is 0.14 meters, with a maximum deviation of 0.58 meters. This relatively low error suggests that /odometry/filtered is a reasonably accurate method for tracking the GT path.

/odom (Black Line), on the other hand, shows substantial deviations from the GT path. In particular, it consistently deviates above GT along the y-axis and diverges further as it progresses, reaching deviations of more than 3 meters in the curved section. The mean absolute deviation for /odom is 3.07 meters, with a maximum deviation of 6.75 meters, indicating a significant discrepancy from the GT path.

According to Figure 4. EKF_slam (Red Dashed Line) closely follows the GT path across the entire route, maintaining small deviations. The calculated mean absolute deviation from GT for EKF_slam is 0.06 meters, with a maximum deviation of 0.38 meters. This low error suggests that EKF_slam provides a high-fidelity approximation of the true path.

odom_slam (Black Line), however, exhibits a significant deviation from GT, especially in the straight segments. It deviates upward by approximately 1 to 1.5 meters in the middle section, indicating a consistent offset from GT. The mean absolute deviation for odom_slam is 0.10 meters, with a maximum deviation of 0.50 meters, showing less accuracy compared to EKF_slam but still relatively close on average.

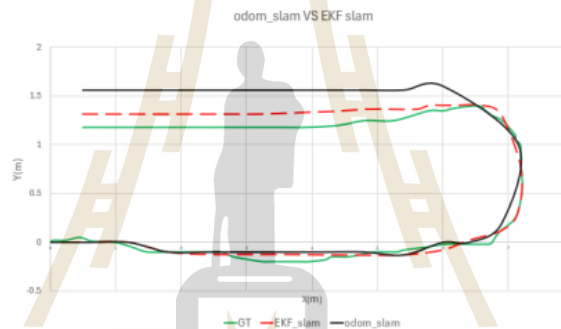


Figure 4. The figure compares the paths obtained from two methods EKF_slam, and odom_slam, with GT(Ground Truth)

7.1. Performance Comparison of SLAM system with Different Odometry Sources

This study evaluated the performance of two odometry data source the results highlighted distinct differences in performance between the two systems. When SLAM was used with the '/odom' frame, with a mean absolute deviation for odom_slam is 0.10. In comparison, the SLAM system using the '/odometry/filtered' frame have mean absolute deviation from GT for EKF_slam is 0.06 meters, with a maximum deviation of 0.38 meters.

It is important to note that the comparison was made based on tests conducted on different paths. While these differences path inherently introduce variations in environmental conditions, the results still strongly suggest that the '/odometry/filtered' system offers a clear advantage in reducing positional error, this finding is significant as it highlights the robustness of the '/odometry/filtered' approach, even when subjected to varied conditions.

The observed differences in performance have practical implications for SLAM applications in environments where precise navigation is critical. The reduced error rate associated with the '/odometry/filtered' frame demonstrates its potential to enhance SLAM performance, particularly in more challenging environments where raw odometry data might lead to significant drift.

This study's results also suggest that. For future SLAM implementations, incorporation sensor fusion techniques, such as those used in 'odometry/filtered', could provide more reliable navigations. This could be especially valuable in scenarios where revisiting locations or retracing paths in common, and where cumulative errors could otherwise degrade the overall system performance.

8. Conclusion

Despite the tests being conducted on different paths, which introduced environmental variations, the superior performance of the 'odometry/filtered' system is evident. This suggests that the sensor fusion techniques employed in the 'odometry/filtered' frame offer a robust solution for minimizing positional errors in SLAM applications. The reduced error rate underscores the effectiveness of integrating filtered odometry data to enhance SLAM performance.

The practical implications of these findings are considerable. For SLAM implementations where precise navigation is crucial, incorporating advanced sensor fusion techniques, like those used in the 'odometry/filtered' frame, could significantly improve system reliability. This approach is especially beneficial in scenarios involving repeated visits to locations or path retracing, where cumulative errors could otherwise compromise overall system accuracy. Future SLAM systems should therefore consider the integration of sensor fusion methods to achieve more accurate and reliable navigation, enhancing performance in challenging conditions.

9. References

- [1] Thrun, S., Fox, D., Burgard, W., & Dellaert, F. (2001). Robust Monte Carlo localization for mobile robots. *Artificial intelligence*, 128(1-2), 99-141.
- [2] Grisetti, G., Kümmerle, R., Stachniss, C., & Burgard, W. (2010). A tutorial on graph-based SLAM. *IEEE Intelligent Transportation Systems Magazine*, 2(4), 31-43.
- [3] Davison. (2003, October). Real-time simultaneous localisation and mapping with a single camera. In *Proceedings Ninth IEEE International Conference on Computer Vision* (pp. 1403-1410). IEEE.
- [4] Berka, Z. (2020). *Mapping road markings and localization of F110 autonomous car using camera*. Retrieved from https://people.ciirc.cvut.cz/~klapajar/studenti/F3-DP-2020-Berka-Zdenek-Mapping_road_markings_and_localisation_of_f110_autonomous_car_using_camera.pdf
- [5] Robotic Operating System (ROS). (2016). REP 103: Package naming and versioning conventions. Retrieved from <https://www.ros.org/reps/rep-0103.html>
- [6] Robotic Operating System (ROS). (2016). REP 105: Package dependency management. Retrieved from <https://www.ros.org/reps/rep-0105.html>
- [7] Saeedi, S., Liu, L., & Lee, S. (2020). *A review of SLAM algorithms and their applications*. *Journal of Open Source Software*, 5(52), 2783. <https://doi.org/10.21105/joss.02783>
- [8] Cummings, M., & Smith, J. (2016). *robot_localization*: Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) implementations for sensor fusion in ROS. Retrieved from https://github.com/cra-ros-pkg/robot_localization

BIOGRAPHY

Mr. Natthawut Pidech was born on June 13, 1999. earned Senior High School Diploma from Grace Christian Academy in Kankakee, Illinois in the 2018, enrolled in the International Program of the School of Mechanical Engineering at Suranaree University of Technology, graduated with a Bachelor of Engineering, he pursued further studies in the 2022, enrolling in the Graduate Program of Mechanical Engineering and Process Systems at Suranaree University of Technology.

During graduate studies, presented the following research paper: "Enhancing SLAM on F1 TENTH Using Data Fusion Kalman Filter Algorithm" at the 14th TSME International Conference on Mechanical Engineering, held from December 10th–13th, 2024, in Pattaya, Thailand.

