Journal of Robotics Research (JRR)
Volume 3, Issue 2, Date: 31 - December - 2026
ISSN: 2978-4344, DOI: 10.64820/AEPJRR.32.8.33.122026

EFM2: An Enhanced Sensor Fusion and Path-Tracking Control Framework for Omnidirectional Automated Guided Vehicles

Ata Jahangir Moshayedi [1, 2], Arash Sioofy Khoojine [2, 3, *], Amir Sohail Khan [2], Atany Shuvam Roy [2, 4], Xu Dangling [2], Yibin Xie [2], Zeashan Hameed Khan [2, 5], Mehran Emadi Andani [2, 6, *]

[1] Smart Structural Health Monitoring and Control Laboratory, DGUT-CNAM,Dongguan, University of Technology, D1, Daxue Rd., Songshan Lake, Dongguan, 523000, Guangdong, China.

[2] Robotics & Automation Research Lab (RARL), Queens Building, Leicester, LE1 9BH, United Kingdom

[3] Faculty of Economics and Business Administration, Yibin University, Yibin, 644000, Sichuan, China

[4] Department of Computer Science and Engineering, University of Liberal Arts Bangladesh (ULAB), Dhaka, Bangladesh

[5] Interdisciplinary Research Center for Intelligent Manufacturin (IRC-IMR), KFUPM, Dhahran, 31261, Saudi Arabia

[6] Department of Neurosciences, Biomedicine and Movement Sciences, University of Verona, Via Casorati, Verona, Italy

Email: [1] aj.moshayedi@lecnam.com,[2] arashsioofy@yibinu.edu.cn, [3] mrsohail21@gmail.com, [4] atanuroy911@gmail.com

[5] 2469184855@qq.com, [6] 2469184855@qq.com, [7] zeashan.khan@gmail.com,[8] mehranemadi@gmail.com

*Corresponding Author

Abstract— **Omni-directional robot platforms offer enhanced flexibility, maneuverability, and adaptability, making them suitable for diverse applications. This study introduces and evaluates a novel sensor fusion algorithm, Extended Fusion Method 2 (EFM2), designed to integrate inputs from a Magnetometer, Camera (with SURF/SIFT object detection), and Lidar (SLAM-based) to achieve fast, accurate, and reliable navigation. The EFM2 algorithm was assessed through both simulation and real-world experiments across multiple path geometries, including continuous and cut paths, as well as industrial-like rectangular tracks. To provide a comprehensive performance evaluation, a Weighted Overall Performance Index (WOPI) was developed, combining conventional metrics velocity, accuracy, stability, and task success into a single holistic measure. Monte Carlo simulations were employed to account for variability in sensor behavior and path conditions, enabling robust ranking of sensor configurations. Results demonstrate that the optimal configuration Magnet at speed 10, Camera at speed 5 with SURF, and Lidar at speed 10 achieves the highest performance in both simulated and real environments. The study further shows that sensor fusion consistently outperforms single-sensor setups, with EFM2 providing superior balance between speed, accuracy, and reliability. These findings validate the effectiveness of the WOPI framework and EFM2 algorithm for advanced, performance-aware navigation in dynamic robotic applications, while highlighting directions for future improvements, including learning-based weight optimization and integration of additional sensing modalities.**

Keywords— ***Automated Guided Vehicle(AGV), Path Tracking, Omni-directional Robots, EFM2 Algorithm, Automation***

I. INTRODUCTION

Today's industrial automation is increasingly enhanced by Automated Guided Vehicles (AGVs), which serve as an essential element of the modern-day smart factories. AGVs are recognized as flexible and efficient solutions for a wide range of tasks such as material handling, logistics, and production support [1] [2]. These robots come in various configurations, including two-wheel, four-wheel, and six-wheel designs [3]. Recently, omni directional AGVs have gained increasing popularity due to their enhanced flexibility and capability to move smoothly in multiple directions, meeting the complex demands of dynamic industrial environments with minimal constraints [4]. One of the primary tasks for AGVs is path tracking and trajectory planning, which depends on the integration of various sensors and advanced algorithms. Recent AGVs integrate a variety of sensors such as LiDAR, ultrasonic sensors, infrared sensors, cameras, and inertial measurement units (IMUs) to accurately perceive their environment and ensure safe navigation across diverse settings [5][6]. For efficient operation, AGVs leverage advanced algorithms including A* and Dijkstra for path planning, and PID, Model Predictive Control (MPC), or Linear Quadratic Regulator (LQR) for trajectory tracking and control [4] [7].Localization is achieved through techniques like Simultaneous Localization and Mapping (SLAM) and Kalman filtering, while obstacle avoidance relies on methods such as the Dynamic Window Approach and Potential Field algorithms. Together, these sensor technologies and algorithms enable AGVs to move precisely and safely in complex environments. Various surveys have demonstrated the impact of sensor fusion [4] [5]. For example, Lee et al.

Received: 25-7-2026
Revised: 25- 8 -2026
Published: 31-12-2026

This article is freely accessible under the Creative Commons Attribution License, allowing for its use, distribution, and reproduction in any format, as long as the original work is correctly cited. © 2026 The Authors.

(2012) [8] developed a high-precision indoor AGV guidance system using magnetic spots with Hall sensors, encoders, and counters, while Yu Jun et al. (2015) [9] implemented a CCD camera–based omnidirectional visual guidance system for Mecanum-wheel robots, achieving stable performance and underscoring the potential for improvements beyond traditional magnetic navigation. Although magnetic sensors are an older technology, they are still widely used in various applications due to their accuracy in robotics [5]. With the introduction of LiDAR, navigation in larger, less constrained spaces has become more feasible, addressing one of the major challenges in achieving true autonomous control [10]. LiDAR is now recognized as one of the key sensors integrated into modern robot designs. Alongside this, various studies have highlighted the use of Kalman filters, deep learning models, and SLAM as the primary algorithms for navigation and perception[10]. As the survey in the previous paper shows, Yun Su et al. (2021) [11] combined the LiDAR, Inertial Measurement Unit(IMU), and wheel encoders using a tightly coupled odometer increment model and sensor data weighting in Ground optimized LiDAR Odometry and Mapping(GR-LOAM). Measured parameters include LiDAR mapping error, Computation time (ms), where GR-LOAM achieves 0.614 m error outdoors, outperforming LOAM and LIO-mapping, with high-frequency pose updates at 100 Hz.Zhong et al. (2021) [12] presented a comprehensive survey of LiDAR and camera fusion algorithms for 3D vision applications, including depth completion, 3D object detection, and semantic segmentation. The reviewed approaches, such as Sparse2Dense, CSPN++, MVX-Net, and LaserNet++, were evaluated using metrics including Root Mean Square Error (RMSE) and Average Precision (AP). For example, CSPN++ achieved an RMSE of 743.69, while MVX-Net reported a vehicle detection AP of approximately 75.86%. The survey concluded that multi-sensor fusion consistently improves accuracy compared with single-sensor approaches across a wide range of computer vision tasks.Tran et al. (2021) [13] proposed a sensor fusion framework that combines a 2D LiDAR and a 3D ultrasonic sensor using a Bayesian filter for three-dimensional environment mapping. Their evaluation focused on mapping accuracy and map quality, demonstrating that the fusion approach generated more accurate and detailed 2D and 3D maps than either sensor operating independently. However, the limited sensing range of ultrasonic sensors required the robot to remain close to surrounding obstacles.Li et al. (2023) [14] integrated LiDAR, a depth camera, and an Inertial Measurement Unit (IMU) using an Extended Kalman Filter (EKF) for mobile robot localization and map construction. The proposed framework was evaluated using RMSE, Mean Absolute Error (MAE), and mapping completeness. Experimental results showed that sensor fusion produced more complete and accurate maps than single-sensor systems. In addition, the combination of an improved Ant Colony Optimization algorithm with the Dynamic Window Approach (DWA) enabled smoother and shorter path planning, achieving a maximum path-following error of approximately 4 cm. Chen et al. (2023) [15] introduced FUTR3D, a modality-agnostic transformer-based sensor fusion framework that integrates data from 2D cameras, 3D LiDAR, 3D radar, and 4D imaging radar. The model was evaluated on the NuScenes dataset using mean Average Precision (mAP) and the NuScenes Detection Score (NDS). Experimental results showed that FUTR3D outperformed existing state-of-the-art methods, achieving 72.1% NDS and 69.4% mAP for LiDAR-camera fusion.Li et al. (2023) [16] proposed DeepFusion, a deep learning-based framework that combines LiDAR and camera features using InverseAug and Learnable Align techniques for 3D object detection. Evaluations on the Waymo Open Dataset demonstrated significant improvements in Average Precision (AP), with gains of up to 8.9 Level-2 APH. The proposed approach also exhibited strong robustness to sensor noise and out-of-distribution scenarios while significantly improving long-range object detection. Obando Ceron et al. (2023) [17] combined RGB camera images, sparse LiDAR depth information, and surface normals using a Conditional Random Field (CRF) optimized through a conjugate gradient algorithm to generate dense depth maps. Performance was evaluated using RMSE and MAE on the KITTI dataset. The proposed method achieved an RMSE as low as 849.39 mm using 5,500 superpixels, providing a computationally efficient alternative to deep learning-based depth completion methods.Peng et al. (2023) [18] proposed the Vision–Inertial–LiDAR Odometry (VILO) SLAM framework, which tightly couples stereo vision, an IMU, and 2D LiDAR residuals through nonlinear optimization. Experimental results demonstrated position error reductions of 20.8–38.7% and yaw error reductions of up to 48.4% compared with ORB-SLAM2 and VINS-Fusion in challenging corridor environments, while maintaining real-time performance at 10 Hz. Lan et al. (2024) [19] developed a modular multi-sensor fusion system integrating LiDAR, cameras, and an IMU with real-time sensor degradation detection and dynamic weighting. The proposed algorithm provided robust localization in GNSS-denied environments by dynamically adjusting sensor weights. Performance evaluation using translational and rotational RMSE demonstrated localization errors ranging from 0.27 to 0.67 m in complex environments. The system consistently achieved sub-meter localization accuracy and reported a 100% success rate in global localization experiments. Nam et al. (2024) [20] proposed a LiDAR-centric SLAM framework for autonomous mobile robots by integrating multiple 2D LiDARs, stereo cameras, and an IMU using binary pose fusion and factor graph optimization. Experimental results showed translation errors below 0.001% and rotation errors below 0.1%, outperforming existing SLAM approaches while maintaining compatibility with low-cost robotic hardware. Dhanush et al. (2024) [21] developed an omnidirectional surveillance robot equipped with an RP LiDAR and cameras. The system employed Hector SLAM and Adaptive Monte Carlo Localization (AMCL) for localization, face recognition for intruder detection, and YOLOv3 combined with pose estimation and a Two-Stream Spatial Temporal Graph for fall detection. The proposed framework successfully achieved indoor mapping, accurate localization, and real-time detection of falls and intruders. However, the authors reported certain limitations related to hardware precision and the robot's motion speed.Zhu et al. (2024) [22] presented a comprehensive survey of multi-sensor fusion SLAM techniques based on cameras, LiDAR, and IMU sensors. The survey discussed

several state-of-the-art algorithms, including Extended Kalman Filter (EKF) variants and sliding-window optimization approaches such as VINS-Mono and LIO-SAM. The reviewed studies primarily evaluated localization accuracy and robustness, concluding that tightly coupled sensor fusion and optimization-based methods effectively overcome the limitations of individual sensors. Yusefi et al. (2024) [23] merge monocular RGB images (via YOLOv7) and 3D LiDAR point clouds to estimate object distance and enhance obstacle avoidance in UGVs. Measured parameters include distance estimation accuracy beyond 50 meters and successful navigation re-planning to avoid occluded obstacles, proving superior to camera-only or LiDAR-only methods. Koshy (2024) [24] explores localization in lunar-like environments using wheel encoders, Sun sensor, and IMU with fusion methods including Simple Fusion, Kalman Filter (KF), and Linear Weighted fusion. Measured parameters include RMSEs of 38 m (x) and 41 m (y) for weighted fusion, outperforming other methods, and localization errors under 92 m overall, demonstrating improved accuracy with minimal computational cost.A survey paper showed that only few works are reported on AGV with OMNI structure and among the sensor suite (Appendix A), magnetic sensors offer a reliable and cost-effective solution for AGVs operating on fixed, guided paths, but they lack the flexibility needed for dynamic navigation [25]. LiDAR remains the most sought sensor due to its high-precision 3D mapping and robust obstacle detection across varied environments [26]. Camera sensors are gaining popularity for their rich visual data and capability to enhance AI driven navigation and object recognition, although they demand significant processing power and are sensitive to lighting conditions [27]. Ultrasonic and infrared sensors are widely used for close-range detection but are limited by their shorter range and vulnerability to environmental interference. IMUs play a vital role in tracking orientation and movement, complementing other sensors to ensure accurate localization [28].Among various algorithms, the most commonly used algorithms in AGV sensor fusion include the EKF, which is widely applied for nonlinear state estimation and localization [29]. The Particle Filter (PF) is popular for handling non-Gaussian noise and complex, multi-modal distributions [30]. Complementary Filters offer a simple and effective method for fusing data from IMU sensors such as accelerometers and gyroscopes [31]. The Unscented Kalman Filter (UKF) serves as an alternative to EKF, providing improved nonlinear estimation accuracy [32]. The classic Kalman Filter (KF) is used for linear sensor fusion problems [33].Neural Networks and Deep Learning techniques are increasingly used for complex sensor data fusion and pattern recognition. Bayesian Networks are applied for probabilistic fusion and decision making based on sensor data. Among these, EKF and Particle Filter are the most prevalent, balancing performance and computational efficiency in AGV applications [34]. The primary challenge of implementing EKF and Particle Filter on small processors in AGVs lies in balancing their computational demands with the constraints of limited hardware resources. EKF involves complex linearization and matrix operations that can be taxing for low power or low-speed processors, potentially causing slower update rates or reduced estimation accuracy. Particle Filter is even more computationally intensive, as it requires managing and updating numerous particles (samples), demanding significant processing power and memory resources that small processors may struggle to provide in real time.In essence, the key difficulty is executing these sophisticated, resource heavy algorithms efficiently on constrained hardware without compromising accuracy or responsiveness(Appendix B). Considering all the aforementioned points regarding the selection of an OMNI AGV platform and the integration of fusion sensors, this paper aims to introduce a novel algorithm called the Extended Fusion Method 2 (EFM2). This novel approach builds upon our previous positive results with earlier Fusion Algorithms (EFM, EFM1) as reported in our prior work [5]. EFM2 extends these methods by incorporating Simultaneous Localization and Mapping (SLAM) along with additional sensor combinations to achieve highly accurate and robust navigation in challenging environments. By evaluating the EFM2 framework using the Weighted Overall Performance Index (WOPI), this study demonstrates the effectiveness of advanced sensor fusion algorithms and establishes a scalable methodology for quantifying robotic performance on a unified and meaningful scale. The primary aim of this study is to design, implement, and evaluate the EFM2 sensor fusion algorithm on a custom omni-directional robotic platform integrating Magnet, Camera (with SURF/SIFT object detection), and Lidar (SLAM-based) sensors to achieve fast, accurate, and reliable navigation. To provide a comprehensive assessment, the Weighted Overall Performance Index (WOPI) is introduced, aggregating multiple conventional metrics velocity, accuracy, stability, and task success into a single holistic measure. The main objectives of this study are as follows:1)Design and simulate a comprehensive omni-directional robotic platform, including its structural, mechanical, and control architectures.2)Develop and implement the lightweight EFM2 path-tracking sensor fusion algorithm suitable for embedded processors with limited computational resources. 3)Validate the effectiveness of EFM2 through extensive simulation studies using different path geometries and industrial-like experimental scenarios.4)Compare different sensor configurations using the Weighted Overall Performance Index (WOPI) to identify the optimal sensing strategy while providing a unified performance evaluation framework.Investigate the effectiveness of multi-sensor fusion techniques for enhancing robotic navigation under diverse operational conditions and environmental constraints.The authors note that relatively few studies have focused on omni-robot path-tracking algorithms. By integrating the holistic evaluation capabilities of the Weighted Overall Performance Index (WOPI) with the advanced sensor fusion architecture of EFM2, this work establishes a strong foundation for the next generation of performance-aware, sensor-driven robotic systems capable of operating effectively in increasingly complex and dynamic environments. The remainder of this paper is organized as follows. Section 2, Methodology, provides a comprehensive overview of the omni-directional AGV platform, including its design, kinematics, and hardware/software components. The section 3 details the proposed Extended Fusion Method 2 (EFM2) algorithm, presents simulation studies carried out

in CoppeliaSim, and discusses real-world experimental validation. Furthermore, this section includes a comparative analysis of performance using the Weighted Overall Performance Index (WOPI). Finally, Section 4 concludes the paper and outlines directions for future research.

II. METHODOLOGY

This section provides a detailed overview of the real OMNI platform used, including its kinematic and dynamic modeling in both simulation and real-world settings. It also covers the hardware and software components, along with an introduction to the proposed EFM2 algorithm.

A. The OMNI AGV platform Design and Modelling:

The platform used is an OMNI-type robot with a square chassis measuring 25 cm × 25 cm, composed of a core computation board and an expansion board for driving control, supporting a total weight of 25 kg. It is equipped with four 100 mm double plastic omnidirectional wheels, each with a 5 cm radius, 6 cm thickness, 16 mm axis width, 6 mm hub, two plates, and eighteen 19 mm rubber rollers on a nylon body. The wheels are mounted at 45° relative to the chassis edges, allowing full 360° omnidirectional maneuverability. Each wheel weighs 275 g, can bear loads up to 20 kg, and enables motion both along and tangential to the wheel axis, transmitting traction, braking, and driving torque while absorbing shocks and vibrations. In addition to the chassis, the design comprises five hardware components Processing Units, Motors and Encoders, Sensor Suite, Power Supply, and Protection, as well as five software modules (Figure 1) which are described as follows.

Figure 1: Overall hardware structure. (A) Robot hardware and software components, (B) Robot platform including wheels, drivers, and chassis, (C) Complete robot platform in real environment, (D) Overall simulated robot in CoppeliaSim, (E) Simulated robot in CoppeliaSim (3D view), (F)Simulated robot in CoppeliaSim (side view).

B. Robot Platform Kinematics & Dynamics:

To precisely control the robotic platform, an accurate kinematic and dynamic model is essential to ensure that the robot behaves as intended. Kinematics translates the desired motion into individual wheel commands, whereas dynamics considers the real-world forces and interactions that influence the robot's movement. Together, these models enable accurate navigation, stable motion, and smooth operation, particularly in complex environments and challenging tasks. Without proper kinematic and dynamic modeling, the robot would experience difficulties in motion planning, control, and adapting to physical constraints [36]. The kinematic and dynamic equations used in this study are summarized in Table 1 and illustrated in Figure 2.

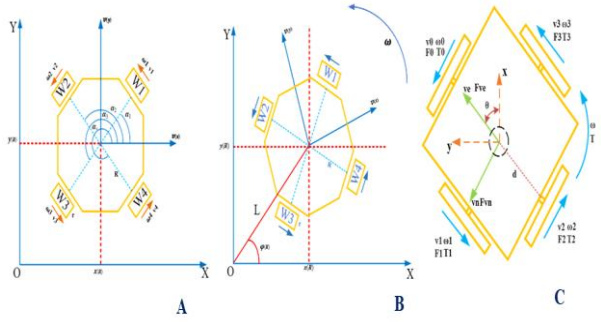

Figure 2: Omni-Robot Platform: Kinematics and Dynamics. Overview of the proposed omni-directional robotic platform. (A) and (B) illustrate the robot's kinematic model, where (W_1), (W_2), (W_3), and (W_4) represent the four omni-wheels; (\alpha_1), (\alpha_2), (\alpha_3), and (\alpha_4) denote the angles between each wheel and the robot reference frame; (\omega_w) represents the angular velocity of the wheels; and (v_w) denotes the linear velocity of the wheels. (C) illustrates the robot's dynamic model, where (T_0), (T_1), (T_2), and (T_3) (N·m) represent the wheel traction torques; (v_e) and (v_n) (m/s) denote the robot's linear velocities; (\omega) (rad/s) is the robot's angular velocity; (F_{ve}) and (F_{vn}) (N) represent the traction forces along the (v_e) and (v_n) directions; (x), (y), and (\theta) define the robot's position and orientation; (d) (m) is the distance between each wheel and the robot's center; (v_0), (v_1), (v_2), and (v_3) (m/s) represent the linear velocities of the wheels; (\omega_0), (\omega_1), (\omega_2), and (\omega_3) (rad/s) denote the wheel angular velocities; and (f_0), (f_1), (f_2), and (f_3) (N) represent the traction forces generated by the wheels.

- **Kinematics of the Robot Platform:** Kinematic modeling describes the motion of a robot based on its geometric structure and the relationship between stationary or moving reference coordinate frames without considering the forces, torques, or moments responsible for the motion. The position and orientation of the omni-wheels have a significant influence on the robot's kinematic model. In this study, the robotic

platform is equipped with four omni-wheels, whose rotational axes intersect at the center of the robot, with adjacent wheels separated by an angle of 90°. The kinematic model establishes the relationship between the robot's body velocities and the angular velocities of its four omni-wheels, which are symmetrically positioned at an angle of 45° relative to the robot's chassis. The robot's state vector is defined in Equation (1). where x and y represent the robot's position in the two-dimensional plane, while θ denotes its orientation (heading) angle. The robot's velocity in the body-fixed reference frame is defined in Equation (2), where vx and vy are the linear velocities along the robot's x- and y-axes, respectively, and ω is the angular velocity about the vertical axis. The angular velocities of the four wheels are defined in Equation (3) and are related to the robot's body velocity through the Jacobian matrix, as shown in Equation (4). The Jacobian matrix considers the wheel radius (r), the distance (l) between the robot's center and each wheel, and the 45° orientation of the omni-wheels. To estimate the robot's body velocity from the measured wheel speeds, inverse kinematics is applied using the pseudo-inverse of the Jacobian matrix, as presented in Equation (5).

- **Dynamics of the Robot Platform:** The dynamics of the omni-directional mobile platform are formulated using the standard rigid-body dynamics framework. The system's mass and inertia are represented in Equation (6), where m denotes the total mass of the platform and Iz represents the moment of inertia about the vertical axis. The overall motion of the robot is governed by the equation of motion presented in Equation (7), where v is the robot's velocity vector, C(v) represents the Coriolis effects, Fr denotes the resistive frictional forces, and B is the input mapping matrix. The relationship between the wheel torques and the resulting body forces is defined in Equation (8), where J is the Jacobian matrix relating the wheel velocities to the platform velocities. Finally, the wheel torques are expressed in Equation (9), where τi represents the torque generated by the corresponding wheel. This dynamic model provides a compact representation of the omni-directional platform, enabling accurate motion modeling, controller design, and analysis of the robot's behavior under different operating and loading conditions.The vector $\boldsymbol{\tau}_i$ (Equation 9) contains the torque commands applied individually to each wheel's motor. These torques generate the driving forces that move and steer the robot.

C. Hardware Components:

Processing Units: The robot operates using a custom-designed PCB, referred to as the motherboard, which connects the various processors and sensors. The motherboard serves as an interface between the main processor, a Raspberry Pi 3B+, auxiliary processors (Arduino Mega and Uno) and other hardware section like, motors, power modules, sensor modules, and secondary processor units, while also allowing expansion for future module integration. Main Processor (Raspberry Pi 3B+), the Raspberry Pi 3B+ serves as the main processor and "brain" of the mobile robot chassis, managing data processing and controlling other processors. It features a 1.4 GHz 64-bit quad-core ARM Cortex A53 CPU, 1 GB LPDDR2 RAM, dual-band 2.4 GHz/5 GHz Wi-Fi, Bluetooth 4.2, Gigabit Ethernet (via USB 2.0), and multiple USB ports. It connects to peripherals such as Arduino UNO, Arduino Mega, RPLidar, and a camera via USB, CSI, and SPI interfaces. Running a ROS-based distributed system, it hosts driver nodes and algorithm modules that communicate through topic subscriptions and publications, making it central to the robot's operation. Auxiliary processors: Arduino Mega/Arduino UNO.

Table 1: Kinematic and dynamic equations for a four-wheel omnidirectional robot platform with wheels placed at 45°.

Aspect	Equation / Description	Notes
Robot Configuration	Four omni-wheels placed symmetrically at 45° relative to the chassis	Enables holonomic movement
State Variables	$x = [x, y, \theta]^T$	x, y: position; θ: orientation
Omni Kinematics Equation	(Section Heading)	
Body Velocity Vector	$v = [vx, vy, \omega]^T$	vx, vy: linear velocity; ω: angular velocity
Wheel Velocities	$\omega w = Jv$, where $\omega w = [\omega_1, \omega_2, \omega_3, \omega_4]^T$	Angular velocities of the four wheels
Jacobian Matrix (J)	$J = (1/r) \times [[-\sqrt{2}/2, \sqrt{2}/2, l],[-\sqrt{2}/2, -\sqrt{2}/2, l],[\sqrt{2}/2, -\sqrt{2}/2, l],[\sqrt{2}/2, \sqrt{2}/2, l]]$	r: wheel radius; l: distance from robot center to each wheel
Inverse Kinematics	$v = J\dagger\, \omega w$	Uses the pseudoinverse of J
Omni Dynamics Equation	(Section Heading)	
Mass and Inertia Matrix (M)	$M = [[m, 0, 0],[0, m, 0],[0, 0, Iz]]$	m: robot mass; Iz: moment of inertia about the z-axis
Equation of Motion	$M\,\tilde{v} + C(v)v + Fr = B\tau$	C(v): Coriolis matrix; Fr: friction forces; B: input mapping matrix
Input Mapping Matrix	$B = J^T$	Maps wheel torques to body forces
Wheel Torques	$\tau = [\tau_1, \tau_2, \tau_3, \tau_4]^T$	Torque applied at each wheel

As it Shown in Figure (1), this design utilizes two microprocessors: Arduino UNO and Arduino Mega. The Arduino UNO serves as the main control unit, managing four motors and their encoders. Each motor-encoder pair connects to dedicated ports (MotorA to MotorD), with motor direction controlled via digital IO pins high level for forward, low level for reverse and speed regulated through PWM signals. Motor speed feedback is obtained from the encoders. The Arduino UNO receives real-time speed commands from the Raspberry Pi 3B+, performs inverse kinematics calculations, adjusts motor speeds accordingly, and sends feedback to the Raspberry Pi via serial communication. The Arduino Mega functions as the sensing unit and currently interfaces with an AGV

magnetic guider sensor through IO connection. It reads sensor data, receives commands from the Raspberry Pi 3B+, and returns processed information through serial communication (Figure 3).

- **Motor and motor Encoder:** The motor has a diameter of 30 mm, a body length of 42 mm, and a total length including the shaft of 85 mm. The shaft itself has a diameter of 6 mm and a length of 35 mm. The mobile robot chassis employs a 12V DC coreless brushed motor, selected for its ease of control. It has a power rating of 17 W, a no-load speed of 8100 RPM, which is reduced to 120 RPM after a 64:1 gearbox reduction, providing the desired torque and speed. The motor operates with a no-load current of 75 mA and a load current of 1400 mA. An incremental optical encoder with a resolution of 12 counts per revolution (CPR) is integrated for precise motion feedback.

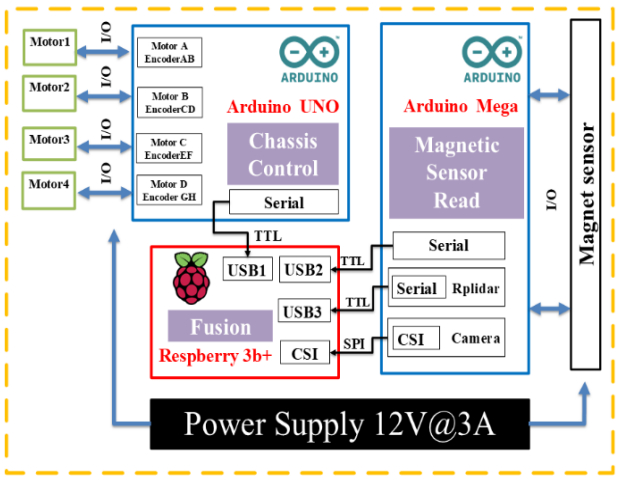

Figure 3: Hardware Connections and Component Relationships of the Designed Robot

- **Sensor Suite:** In the sensor section, the platform utilizes three types of sensors magnetic, vision (camera), and LiDAR selected based on their popularity and effectiveness in sensor fusion configurations. 1) Magnet Sensor: The platform employs the magnetic navigation sensor model AGS-16N (size 200 × 50 × 17 mm), which detects only the N-pole magnetic field using 16 sampling points and can sense weak fields below 100 gauss above the magnetic stripe. It has an effective detection distance of 5 to 55 mm, a response time of 1 ms, and a detection sensitivity of 0.5 mT. The sensor operates over a voltage range of 10–30 V, with a maximum current consumption of 80 mA, and an operating temperature range of -10°C to +50°C. It functions by detecting the magnetic line and sending a signal to the processor whenever the line is sensed, enabling precise navigation along predefined paths. 2) Vision Sensor (Raspberry Pi Camera): The camera has physical dimensions of approximately 24 × 25 × 9 mm, a focal length of 3.04 mm, and an 8-megapixel resolution (3280 × 2464 pixels). This high-definition camera is compatible with all Raspberry Pi versions and supports capturing both HD video and still images. It features a Sony IMX219 sensor with a 1/4-inch CCD and a 73° field of view. The Pi camera supports a single channel and connects via the CSI-2 bus interface, enabling a maximum frame rate of 30 fps. Integrated directly into the mobile robot chassis, it connects to the Raspberry Pi through the native CSI interface for seamless operation. 3) Lidar sensor: As the final sensing component, the robot platform employs the Slamtec A1M8 LiDAR, which is designed for indoor applications and connected to the Raspberry Pi via USB. The sensor measures 98.5 × 70 × 60 mm and weighs approximately 170 g (excluding the battery). It provides a sensing range of 0.15–6 m with a full 360° scanning angle, a distance resolution of less than 0.5 cm, and an angular resolution of less than 1°. The LiDAR acquires more than 2000 sampling points per second at a scanning frequency of 5.5–10 Hz, generating dense point clouds for accurate environmental mapping and navigation.
- **Power Supply, and Protection:** This section covers the battery used, the voltage regulation components, and the protection circuitry. **1) Battery:** The platform is powered by a 6-cell, 22.2-volt, 488.4 Wh Tattu Plus 22000 (25,000 mAh) battery, serving as the main power source for the system. **2) Voltage Regulator:** Since the processors, sensors, and other components operate at different voltages 12V, 5V, and 3.3V, a voltage regulator module is mounted on motherboard to convert the main power supply to the appropriate operating voltages for the Raspberry Pi, Arduino, and other devices, ensuring stable and reliable operation. **3) Fuse:** To protect each section of the robot, a 2A fuse is incorporated on the PCB to ensure overcurrent protection for all components.

D. Software Architecture:

The software architecture of the mobile robot platform is described as follows. It comprises five primary components: Raspberry Pi OS, Arduino IDE with C/C++, OpenCV (Python/C++ bindings), SLAM packages including Hector SLAM and Gmapping, and the Robot Operating System (ROS). Within ROS, the key modules **move_base**, **mag_base**, **camera_base**, and the ROS–Arduino Bridge communicate via ROS topics, ensuring high cohesion within modules and low coupling between them. This modular design enables flexible integration and efficient data exchange across the system.

- **Raspberry Pi OS:** As the primary operating system on the Raspberry Pi 3B+, responsible for high-level data processing, running SLAM algorithms, and real-time control through ROS.

- **Arduino IDE with C/C++**: Used to program the Arduino UNO and Mega boards.
- **OpenCV (Python/C++ bindings)**: Offers comprehensive computer vision capabilities, facilitating image capture and processing from the Raspberry Pi Camera to support visual navigation.
- **SLAM Packages (Hector SLAM, Gmapping):** ROS-compatible SLAM algorithms that process LiDAR data for mapping and localization, enabling autonomous navigation through dynamic occupancy grid maps.
- **ROS (Robot Operating System):** ROS serves as the middleware framework enabling modular communication between distributed nodes. It integrates sensor data, SLAM processing, and coordinates navigation tasks, supporting peripherals like RPLIDAR and the Raspberry Pi Camera. The ROS modules consist of move_base, mag_base, camera_base, and the ROS–Arduino Bridge, which communicate via ROS topics. These modules communicate through ROS topics, enabling high cohesion within modules and low coupling between them. This modular design supports flexible integration and efficient data exchange across the system, as described in the following (Figure 4). 1) Move base: The core module of the ROS Navigation Stack, managing 2D navigation by processing odometry, sensor inputs, and target poses to generate safe velocity commands. Key components include: 2) Global Planner: Uses algorithms such as A* and Dijkstra to plan optimal paths on global costmaps generated from static or SLAM maps. 3) Local Planner: Implements approaches like the Dynamic Window Approach (DWA) for real-time velocity command generation, enabling local path tracking and obstacle avoidance. 4) Recovery Behaviors: Triggered when navigation issues arise, these behaviors clear obstacles from costmaps or rotate the robot to restore navigability. The system maintains coordinate transformations between robot and map frames using the tf package, ensuring precise localization and movement control. 5) Mag base: This module processes data from a 16-bit digital magnetic sensor array divided into three groups: left (sensors 1–5), middle (6–11), and right (12–16). By comparing readings from these groups, the robot determines whether it should turn left, right, or proceed straight, generating corresponding speed commands. 6) Camera_base: The camera module captures RGB images, converts them to grayscale, and isolates regions of interest to improve processing efficiency. Image preprocessing includes Gaussian blur, erosion, dilation, and Sobel edge detection. Feature detectors such as SURF and SIFT extract keypoints, which are spatially grouped into left and right clusters. The system averages these points, calculates deviation, and employs a PID controller to convert visual feedback into velocity commands. 7) ROS Arduino Bridge: Facilitates serial communication between the Raspberry Pi and Arduino boards (UNO and Mega), enabling real-time data exchange for motor control and sensor feedback.
- **Serial Communication Libraries:** Facilitate synchronized, real-time communication between the Raspberry Pi and Arduino boards.

E. OMNI Platform Simulation in CoppeliaSim:

Since the used Omni platform are not are not available in the CoppeliaSim library, based on mentioned kinect and dynamice relation a custom model was developed (**Figure 5**). Instead of simply adding rollers which complicates the simulation the model leverages the wheel's motion characteristics, where the hub and roller axes are perpendicular. It consists of two spheres of different sizes connected by a rotating joint acting as the drive shaft: a smaller sphere represents the hub, while a larger sphere in contact with the ground acts as the roller, attached via a perpendicular rotating joint. Then the rectangular chassis (25 cm × 25 cm × 0.5 cm) was created, with four omnidirectional wheels mounted at the corners, each oriented toward the chassis center and spaced 90° apart. The simulation comprehensively covers mechanical, electrical, and control system aspects, following a structured development process. After constructing the omnidirectional wheel model and assembling the chassis, key components such as the frame, PCB, and LiDAR were integrated based on the actual robot's specifications. Sensor performance was then simulated, with a focus on LiDAR accuracy and response times. Radar scripting was adapted from an existing LiDAR driver script and thoroughly tested for effectiveness. During integration, mechanical and electrical components were precisely aligned and tested to ensure stable operation. **ROS** was initialized alongside **CoppeliaSim** to enable communication and control, with successful plugin loading confirming proper integration. Mapping was performed using classical SLAM with **GMapping** and **RViz**, where LiDAR scan data generated maps for real-time navigation visualization. The robot's pose and velocity data were synchronized and published to **ROS**, enabling manual navigation and map creation via keyboard control. The **move_base** package was configured for global and local path planning by combining LiDAR, map, and localization data. Parameters were set via configuration files, and navigation tests confirmed the robot's ability to reach targets while avoiding obstacles. Finally, the control system integrated **CoppeliaSim**, **ROS**, and Python. Lua scripts in **CoppeliaSim** published sensor data and controlled motors based on velocity commands from **ROS**, while Python scripts handled target goal publishing. This closed-loop system enabled autonomous navigation. All aspects of the robot platform are simulated in the **CoppeliaSim** environment, following the specifications detailed in our previously reported work on the OMNI robot.

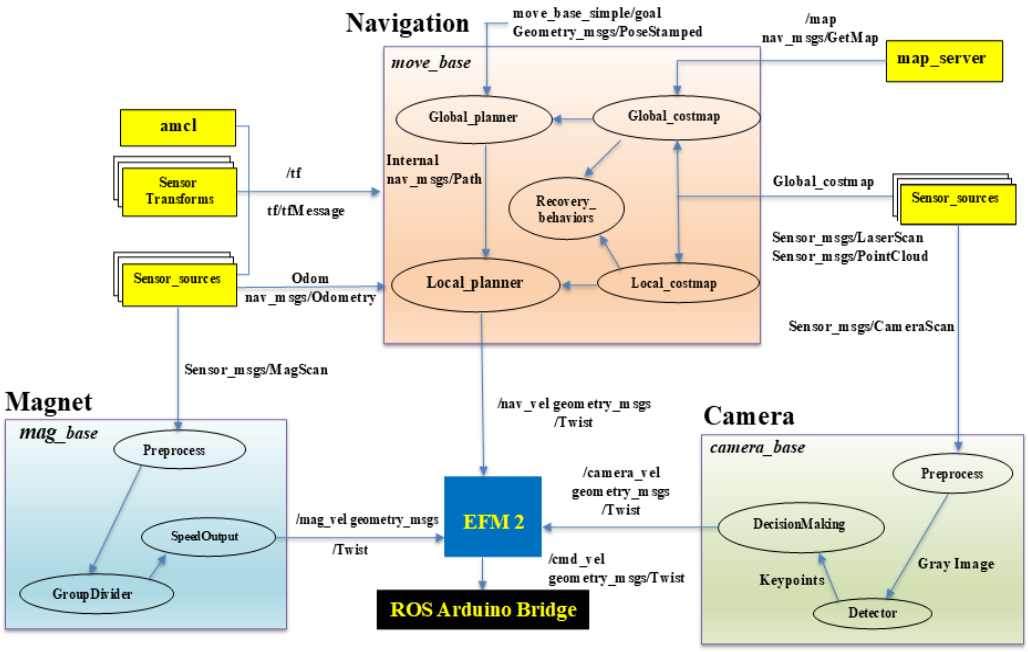

Figure 4: Overall software structure

F. Extended Fusion Algorithm (EFM2) Algorithm:

The proposed novel algorithm named Extended Fusion Algorithm (EFM2) is a hierarchical, multi-sensor navigation framework developed to enhance mobile robot robustness by integrating multiple perception modules and control strategies. The process begins with an initialization phase in which all essential hardware and software components are configured. The camera module a Raspberry Pi camera connected via CSI interface is initialized in the ROS environment using Python and the cv2 library, enabling real-time video capture from /dev/video0. The 16-channel magnetic sensor is configured by assigning Arduino Mega pins and executing initialization routines through the magSensorInit () function.

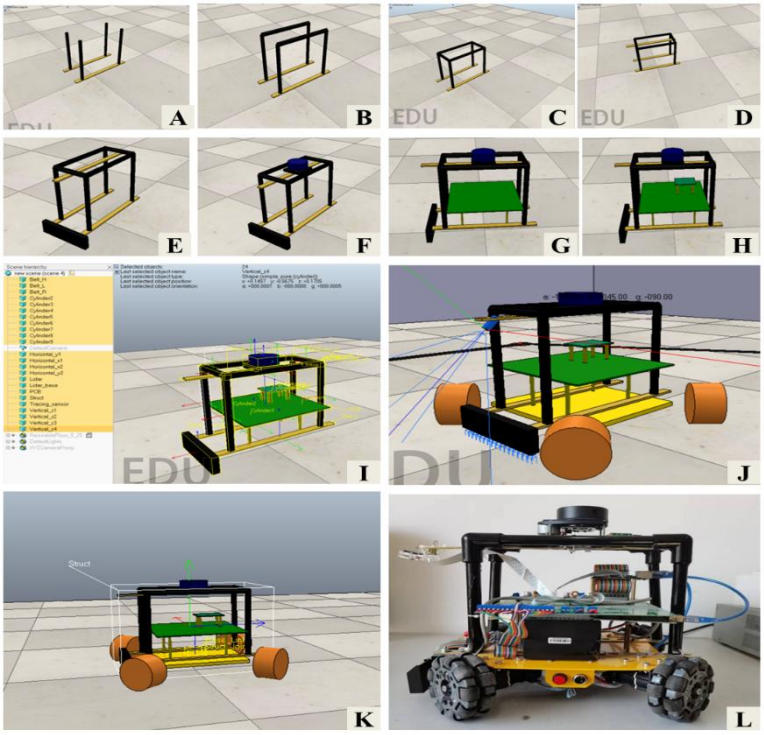

Figure 5: OMNI Platform: Simulation Structure and Workflow in CoppeliaSim, featuring omnidirectional wheels, a rectangular chassis, and integrated sensors (LiDAR, PCB, and frame) based on actual specifications. (A)–(D) Main structure assembly for mounting the camera and LiDAR; (E)–(H) Sections for the magnetic sensor, LiDAR, and motherboard; (I)–(J) Complete robot platform in CoppeliaSim; (K) Overall simulated platform; (L) The real robot platform.

The LiDAR unit, connected via USB, is launched using the **rplidar_ros** package, enabling 360° scanning for mapping and obstacle detection. Controller parameters such as PID gains (Kp, Ki, Kd) are tuned empirically to ensure stable navigation control. In addition, system parameters including camera resolution and frame rate, magnetic sensor thresholds, drivetrain geometry, encoder resolution, and both local and global costmap settings are optimized at this stage (see **Table efm2**). Particular attention is given to fine-tuning camera parameters such as image width and height, frame rate, camera matrix, and distortion coefficients; UNO micro controller settings including communication port, baud rate, and data transfer rates; drivetrain parameters like wheel diameter, track width, and encoder resolution; control parameters such as PID gains; and navigation parameters involving local and global cost maps, global and local planners, and **move_base** configurations. These configurations directly influence the performance of the algorithm. In operation, the fusion process follows a hierarchical priority scheme in which the magnetic sensor provides the primary navigation reference, followed by the camera, and then the LiDAR. Each sensor contributes data through its dedicated processing algorithm, and the integrated results guide the robot's movement decisions in real time. After initialization, the algorithm operates in a continuous loop governed by a three-level priority structure. Magnetic sensor data has the highest priority; when a valid magnetic path is detected, the corresponding navigation module directs the robot along the track. If magnetic data is unavailable or unreliable, control shifts to camera-based navigation. In this mode, captured images are processed by first converting them to grayscale using **cv2.cvtColor**, followed by Gaussian blur via **cv2.GaussianBlur**, thresholding with **cv2.threshold**, and morphological operations using **cv2.erode** and **cv2.dilate** to isolate navigation features. The largest contour is identified with **cv2.findContours** and **max(contours, key=cv2.contourArea)**, its moments are calculated using **cv2.moments**, and the centroid coordinates (**cX, cY**) are determined. Lines are drawn to mark the centroid (**cv2.line**) as well as the contour area (**cv2.drawContours**) for visualization. Path deviation is then measured, and a PID controller adjusts motor speeds to maintain accurate navigation.If both magnetic and camera inputs fail, the algorithm resorts to LiDAR-assisted navigation, which leverages odometry and point cloud data to avoid obstacles and move toward target way points. When the LiDAR confirms arrival at a designated location, the robot executes predefined site-specific actions; otherwise, it continues navigation. If all sensor inputs fail, an emergency stop routine halts the robot, logs the event, and issues a safety alert.

Table 2: ***Parameters configuration of the proposed Extended Fusion 2 (EFM2) algorithm.***

Section	Parameter	Value	Section	Parameter	Value
Camera	image_width	150	PID	Kp	10
	image_height	200		Kd	12
	camera_name	camerav1_150x200		Ki	0
	camera_FPS	20		Ko	50
				accel_limit	1.0
Arduino	port	/dev/uno	Drivetrain	wheel_diameter	0.115
	baud	9600		wheel_track	0.50
	timeout	0.1		encoder_resolution	7392
	rate	30		gear_reduction	1.0
	sensorstate_rate	10		motors_reversed	True
	use_base_controller	True			
	base_controller_rate	50			
Local costmap	update_frequency	5.0	Global planner	recovery_behavior_enabled	true
	publish_frequency	2.0		controller_frequency	2
	obstacle_range	2.5		controller_patience	2
	raytrace_range	2.5		planner_frequency	2
	static_map	false		planner_patience	2
	width	2.5		conservative_reset_dist	6.0
	height	2.5		oscillation_timeout	10.0
	resolution	0.05		oscillation_distance	0.15
Local planner	acc_lim_x	2.0	Global costmap	update_frequency	2.0
	acc_lim_y	0.0		publish_frequency	1.0
	acc_lim_theta	2.0		obstacle_range	5.0
	max_trans_vel	0.1		width	15.0
	min_trans_vel	0		height	15.0
Move base	global_frame	map	Move base	robot_base_frame	base_link
	rolling_window	true		observation_sources	scan

Each sensor module functions as an independent subroutine. In the Magnetic Sensor Algorithm, the sixteen channels are grouped into left, middle, and right zones, with their summed values determining whether the robot should turn or move forward. After initialization, the **magSensorRead()** function reads 16-bit sensor data, which is then analyzed by the **msgDeal()** function. This function divides the data into the three zones (left, straight, right) and compares their values to determine the robot's movement direction: left, right, or straight. To enhance camera based navigation, feature detection algorithms such as Speeded Up Robust Features (SURF) or Scale Invariant Feature

Transform (SIFT) may be incorporated. In the LiDAR SLAM algorithm, while the underlying SLAM concepts are critical for autonomous navigation, the document primarily emphasizes that the Slamtec ROS toolkit is used to interface with the LiDAR hardware. Unlike the camera and magnetic sensor modules, the specific algorithmic details such as pseudocode or explicit function calls are not fully elaborated. It is understood, however, that LiDAR data contributes to path planning and obstacle avoidance whenever magnetic and vision inputs fail, maintaining safe and continuous navigation. Overall, the EFM2 fusion algorithm integrates prioritized sensor data streams magnetic sensing first, vision second, and LiDAR third within a modular, hierarchical framework. This approach allows flexible adaptation to varied environmental conditions while maintaining robust navigation performance. Embedded safety mechanisms, including emergency stop routines and event logging, ensure that the robot can prevent uncontrolled operation even if all sensor inputs fail.

Algorithm 1: Extended Fusion Algorithm (EFM2)

```
1:  Phase 1: Init & Parameters
2:    CALL: init_camera(), init_magnetic(), init_lidar(), init_pid(), init_ros()
3:    Set thresholds, limits, gains, tracing_state <- 'MAGNETIC', running <-
TRUE
5:  Phase 2: Navigation Loop, while running do
7:     mag <- READ magnetic
8:     if mag available & DetectMagnetic() then CALL: magnetic_nav(mag)
9:     else
10:       cam <- READ camera; features <- extract(cam)
11:       if features then CALL: camera_nav(features)
12:       else lidar <- READ lidar
14:         if lidar then CALL: lidar_nav()
15:         else CALL: emergency_stop()
16:         end if
17:       end if
18:    end if
19:  end while Phase 3: Functions
21: function MAGNETIC_NAV(mag)
22:   Partition: L[1:5], M[6:10], R[11:16]; SL, SM, SR
23:   if SL > max(SM,SR) then TurnLeft()
24:   else if SR > max(SL,SM) then TurnRight()
25:   else GoStraight(), end if, end function
28: function CAMERA_NAV(features)
29:   K <- SURF/SIFT; partition KL, KR; e <- path_error
30:   u <- PID(e); AdjustMotorSpeed(u)
31: end function
32:   function LIDAR_NAV()
33:     pos<-Encoder(); dist<-Target(pos); obs<-Lidar(); rp<-Odom()
34:       if obs<=OBSTACLE_RANGE then avoid_obstacle()
35:   else
36:     if dist<thr then station_protocol(); running<-FALSE
37:     else continue_nav(),
38:    end if
39:  end if
40: end function
41: function EMERGENCY_STOP()
42:   SafeShutdown(); LogResults(); return 'STOP'
43: end function
```

G. Metrics and testing environment:

The robot's performance was evaluated using conventional navigation metrics and a newly proposed Weighted Overall Performance Index (WOPI), which integrates these metrics to reflect their combined effect on performance.

- **Conventional navigation metrics:** The robot's performance was evaluated using two sets of metrics. The first set consists of conventional metrics, as shown in Table 3, which include Target Velocity, Path Length, Running Time, Average Velocity, Body Change, Minimum Error, Maximum Error, Average Error, and Success Rate. As Table 3 shows, Target Velocity represents the commanded speed of the robot during operation, expressed in centimeters per second (cm/s). The Path Length denotes the total distance traveled along the planned route, measured in centimeters. Running Time captures the duration required to complete the path, measured in seconds, while Average Velocity reflects the robot's actual mean speed throughout the traversal. To assess the robot's maneuverability, Body Change quantifies the angular rotation of the robot's body during navigation, expressed in radians, with minimum, maximum, and average values recorded. Navigation accuracy is evaluated by measuring the deviation from the intended path through Minimum Error, Maximum Error, and Average Error, all expressed in meters.
- **Weighted Overall Performance Index (WOPI):** The proposed novel index is designed to evaluate the combined impact of conventional metrics on robot performance, enabling more accurate and holistic comparisons. This index also highlights the relative importance of each metric, indicating which factors have greater priority in determining overall performance.

The experimental dataset $\mathbf{D} = \{\mathbf{x_i}\}_{i=1}^{N}$ comprised **N** trials across robot path geometries under sensor configurations. Each observation $\mathbf{x_i}$ contained four feature categories: path characteristics (shape type **G** and length **L**), temporal metrics (running time **T** and actual velocity $\mathbf{V_a}$), performance measures (positional error **ε** and body orientation change $\mathbf{\Delta\theta}$), and completion status (binary success $\mathbf{S \in \{0,1\}}$). Data preprocessing involved normalization of success rates and velocity ratios:

$$S_{\text{norm}} = \frac{S - \min(S)}{\max(S) - \min(S)} \quad (1)$$

$$V_{\text{ratio}} = \frac{V_a}{V_t} \quad (2)$$

where $\mathbf{V_t}$ denotes the target velocity. Outliers were removed using a $\mathbf{3\sigma}$ threshold on temporal metrics ($\mathbf{T > \mu_t + 3\sigma_t}$), affecting **2.4%** of observations. Derived variables included path completion percentage and a stability metric

$$1 - \frac{|\Delta\theta|}{\pi}$$

with $\mathbf{\Delta\theta \in [-\pi, \pi]}$ ensuring normalization.

The composite performance metric **WOPI** was formulated as a convex combination of four normalized sub-metrics:

$$\text{WOPI} = w_1 f_1(x) + w_2 f_2(x) + w_3 f_3(x) + w_4 f_4(x) \quad (3)$$

subject to

$$\sum_{k=1}^{4} w_k = 1, w_k \in [\ell_k, u_k] \quad (4)$$

The component functions were defined as

$$f_1(x) = \min\left(\frac{V_a}{V_t}, 1\right) \text{(Velocity achievement ratio)} \quad (5)$$

$$f_2(x) = \frac{1}{1+\varepsilon} \text{(Positional error metric)} \quad (6)$$

$$f_3(x) = S_{\text{norm}} \text{(Normalized success rate)} \quad (7)$$

$$f_4(x) = 1 - \frac{|\Delta\theta|}{\pi} \text{(Orientation stability index)} \quad (8)$$

Initial weight boundaries **[ℓ_k,u_k]** were derived from expert elicitation, with **$w_1 \in [0.15,0.35]$**, **$w_2 \in [0.15,0.30]$**, **$w_3 \in [0.25,0.50]$**, and **$w_4 \in [0.10,0.25]$**.

The evaluation employed three complementary approaches. First, Monte Carlo simulation generated **500** weight vectors **$\mathbf{w}^{(m)} \sim \mathbf{U}([\ell_k,u_k])$**, computing WOPI scores for all trials via

$$\text{WOPI}_i^{(m)} = \mathbf{f}(x_i)^T \mathbf{w}^{(m)} \quad (9)$$

where

$$\mathbf{f}(x_i) = [f_1(x_i), f_2(x_i), f_3(x_i), f_4(x_i)] \quad (10)$$

Second, differential evolution optimized **w*** by maximizing the mean WOPI:

$$w^* = \arg\max_w \left(\frac{1}{N}\sum_{i=1}^{N} f(x_i)^T w\right) (11)$$

using a population size of **15** and mutation factor **0.8**. Third, **k-means clustering (k = 3)** grouped trials based on standardized features

$$z_i = \frac{x_i - \mu}{\sigma} \quad (12)$$

minimizing the within-cluster variance

$$\min_{\{C_j\}} \sum_{j=1}^{3} \sum_{z_i \in C_j} \| z_i - \mu_j \|_2^2 \quad (13)$$

Methodological robustness was verified through three approaches. Sensitivity analysis employed Sobol indices

$$S_k = \frac{\text{Var}_w(\mathbb{E}[\text{WOPI} \mid w_k])}{\text{Var}(\text{WOPI})} \quad (14)$$

to quantify weight importance. Temporal validation used Dynamic Time Warping between trajectories **p** and **q**:

$$\text{DTW}(p,q) = \min_{\pi} \sum_{(i,j)\in\pi} \| p_i - q_j \|_2 \quad (15)$$

where **π** denotes a warping path. Statistical significance of cluster differences was tested via the Kruskal–Wallis statistic

$$H = \frac{12}{N(N+1)} \sum_{j=1}^{3} \frac{R_j^2}{n_j} - 3(N+1) \quad (16)$$

with post-hoc Dunn's tests (**$\alpha = 0.05$**, Bonferroni-corrected).

The pipeline was implemented using vectorized operations for WOPI computation

$$\text{WOPI}_{\text{all}} = \mathbf{F}\mathbf{w}^T, \mathbf{F} \in \mathbb{R}^{N\times 4} \quad (17)$$

where **F** stacks all **$\mathbf{f}(x_i)$**.

Algorithm 2. WOPI-Based Path Planning Framework

```
Input: Raw dataset D of N trials; path geometries G = {rectangle, circle, playground, spiral, 8-shape}; sensor configs C = {magnet, vision, fusion}; weight bounds W = [wmin, wmax]
Output: optimal weights w*; cluster assignments; performance metrics
procedure DataPreprocessing(D)
    for each trial x_i ∈ D do
        x_i.Vratio ← x_i.Vactual / x_i.Vtarget
        x_i.Snorm ← (x_i.S − minS) / (maxS − minS)
        x_i.Stability ← 1 − (|x_i.Δθ| / π)
        if x_i.T > (μT + 3σT) then
            D.remove(x_i)
        end if
    end for
    D ← D.drop_incomplete()
    return D
end procedure
function CalculateWOPI(x, w)
    f1 ← min(x.Vratio, 1);  f2 ← 1 / (1 + x.ε);  f3 ← x.Snorm;  f4 ← x.Stability
    return w1f1 + w2f2 + w3f3 + w4f4
end function
procedure MonteCarloSimulation(D, W)
    top_performers ← [ ]
    for m ← 1 to 1000 do
        w_m ← RandomSample(W)
        for each x_i ∈ D do
            x_i.WOPI ← CalculateWOPI(x_i, w_m)
        end for
        top_performers.append(arg max(MeanWOPI(w_m)))
    end for
    return top_performers
end procedure
```

H. Test environment and Experiment:

This study evaluates five distinct paths P1 (Rectangle), P2 (Circle), P3 (Playground), P4 (Spiral), and P5 (8-shape) designed for robotic navigation testing, each with unique geometric characteristics (Table 3). Together, these diverse paths provide a comprehensive environment for evaluating navigation and control algorithms. As illustrated in Figure 8 for each path, the robot's performance was tested under both normal and complex conditions, the normal path being the original layout without cuts or obstacles, and the complex path including added obstacles and gaps to increase the challenge.As shown in Figure 6 and Table 3, five navigation paths were designed for robotic experiments, each with normal and complex versions. The paths vary in shape, length, and curvature, ranging from a quasi-rectangular track to an intricate figure 8 route. Complex paths incorporate cylindrical obstacles (60 cm height, 10 cm diameter) and multiple gaps of varying lengths to simulate challenging environments. These modifications aim to test robot navigation and obstacle avoidance under more demanding conditions, with the number of obstacles and gap sizes increasing alongside path complexity.

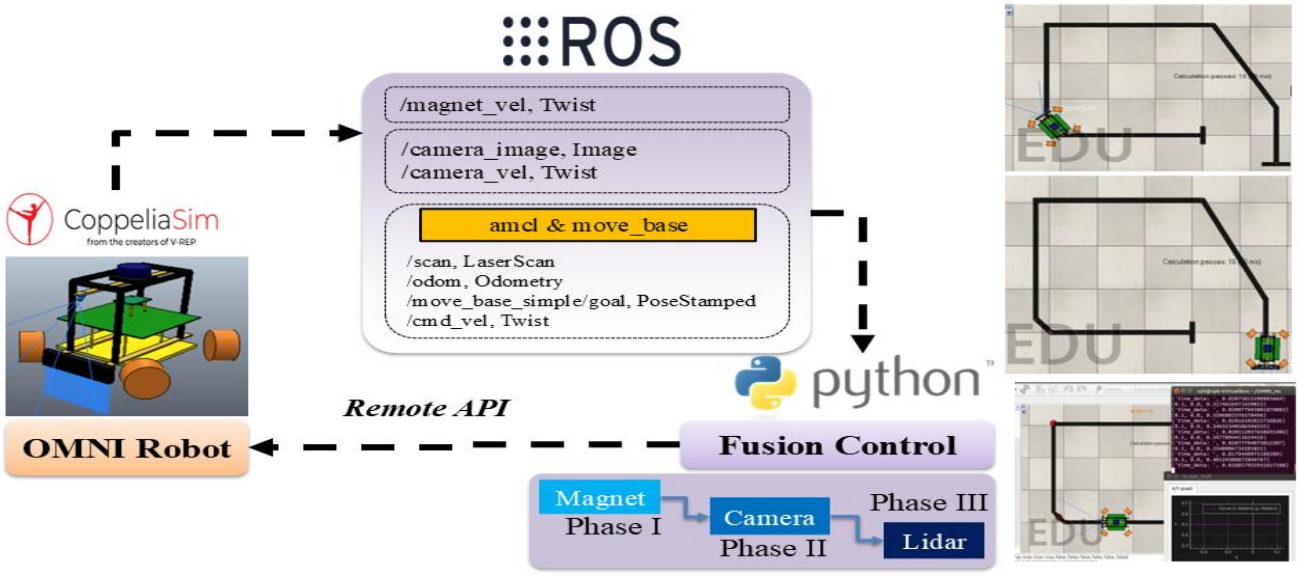

Figure 6: EFM2 implementation in CoppeliaSim simulation, integrated and coordinated with ROS and Python for control and communication.

Table 3: Specifications of normal and complex navigation paths with obstacles and gaps. P1 (Rectangle), P2 (Circle), P3 (Playground), P4 (Spiral), P5 (8-shape) and TL (Total Length).

Path	Description	TL (cm)	Radius (cm)	Details	Obstacles	Gaps
P1	Quasi-rectangular	675	N/A	Various bends	2	3 gaps: 0.20, 0.75, 0.40
P2	Circular route	800	127.3	Radius of circle	3	3 gaps: 0.20, 0.95, 0.80
P3	Running-track shape	800	63.7	200 cm straights	2	3 gaps: 0.30, 1.00, 1.00
P4	Spiral (2 semicircles)	785	100 / 150	Different radii	2	3 gaps: 0.30, 0.80, 1.00
P5	Figure-8 (two circles)	2250	200	130 cm gap	5	6 gaps: 0.30, 0.45, 1.80, 0.30, 0.45, 2.25

I. Fusion algorithm implementation in Simulator:

The Fusion algorithm operates through the coordinated integration of CoppeliaSim, Python, and ROS, forming a comprehensive framework for autonomous robot navigation in a simulated environment. The simulation implementation is shown in **Figure 7**. As shown, CoppeliaSim provides the primary simulation platform, modeling the robot's movement and sensor perception under various road conditions. The robot's guidance system fuses data from magnetic sensors, cameras, and LiDAR, dynamically switching between guidance modes in real time to adapt to environmental complexity, while continuously publishing sensor readings and motion commands to ROS topics. ROS serves as the central processing and communication hub. Leveraging the **amcl** and **move_base** packages, it calculates the robot's motion trajectory and real-time control information by processing map data, live LiDAR point clouds, and the robot's pose. Additionally, ROS functions as a bridge between the simulation and control layers, aggregating data from CoppeliaSim and Python and publishing the robot's motion commands to the *xxx_vel* topic as **Twist** messages. Python hosts the Fusion algorithm and implements the motion control logic. It subscribes to the *xxx_vel* topic to identify the current guidance mode (**magnet_vel**, **camera_vel**, or **cmd_vel**), extracts the **Twist** message information, and converts it into motor control parameters. These parameters are then transmitted to CoppeliaSim via the remote API, allowing the robot to execute smooth, mode-adaptive motion in the simulated environment. Through this collaborative architecture, the Fusion algorithm achieves robust, sensor-fused navigation while maintaining real-time responsiveness and seamless integration across simulation, control, and communication layers.

III. EXPERIMENT SECTION AND ANALYSIS:

The experimental evaluation was carried out through both simulation and real testing on the designed platform, with performance metrics analyzed in each stage. Each experimental configuration was repeated five times (N = 5), and the reported results represent the corresponding aggregate performance. In the simulation phase, three categories of tests were conducted: continuous path tracking across five predefined trajectories, cut path tracking, and cut-path tracking with obstacles. These tests examined the performance of individual sensors (magnet, camera with FAST, camera with SURF, and LiDAR) as well as their combinations under different conditions. Building on the

simulation outcomes, five experiments were performed in the real environment using the rectangle path to validate practical performance. The rectangle path was specifically chosen due to its close resemblance to the real-world experimental setup and its relevance to practical applications. These experiments investigated individual sensor capabilities, multi-sensor fusion, speed optimization, parameter tuning, and robustness in industrial-like scenarios. Together, the two phases provided a comprehensive assessment of the proposed system in both controlled and realistic environments. It should noted in presenting the results throughout this study, sensor configurations and their associated speeds are expressed using a consistent notation protocol. Specifically, M denotes the Magnet sensor, C denotes the Camera, and L denotes the Lidar. For each configuration, the speed parameter was systematically varied in increments from 5 to 25, allowing performance to be evaluated under a range of operating conditions.

A. Simulation Environment Experiments:

As outlined in the simulation phase, three categories of tests were designed. The first category focused on continuous paths, which included five predefined trajectories: PATH 1 (Rectangle), PATH 2 (Circle), PATH 3 (Playground), PATH 4 (Spiral), and PATH 5 (8-shape). In addition, a cut-path test, both with and without obstacles, was performed on the rectangular path. The results are summarized as follows:

Simulation Step 1: Continuous Path: Seven configurations were tested to evaluate the performance of individual sensors (magnet, camera with FAST, camera with SURF, and LiDAR) as well as their combinations. The experiments included: (P1-1) magnet only, (P1-2) camera with FAST, (P1-3) camera with SURF, (P1-4) magnet + camera (FAST), (P1-5) magnet + camera (SURF), (P1-6) sensor fusion with FAST, and (P1-7) sensor fusion with SURF. Since no obstacles were present in this scenario, the LiDAR was not active; therefore, the results of P1-6 and P1-7 were equivalent to those of the corresponding magnet–camera combinations (Table 4). Analysis of Table 4 shows that the Magnetometer (Mag) alone achieved a 100% success rate across all paths, demonstrating high reliability. The average velocity closely matched the target (approximately 25–26 cm/s), indicating stable control. Errors were moderate, with average error values ranging from 1.09 cm on P2 to 2.83 cm on P1, and the highest error observed at 17.76 cm on P4, reflecting strong robustness and precision, particularly for simpler paths (P1–P3).In contrast, the Camera (FAST) setup showed more variable performance. While it achieved **100%** success on most paths, **P1** and **P5** had lower success rates of **80%** and **60%**, respectively, highlighting its sensitivity to path complexity and limitations in camera-based tracking. Errors were larger than those for Mag-only, with average errors reaching **14.44 cm** on **P3** and **10.4 cm** on **P1**. Velocity tracking also degraded on **P1** and **P5**, with the average velocity dropping to **8.56 cm/s** on **P5** despite long runtime. Camera (SURF) performed more reliably than FAST, maintaining a **100%** success rate across all paths and providing better velocity tracking closer to target values. However, error levels remained relatively high, with average errors between **6.05 cm** and **12.58 cm**, suggesting that SURF offers more stable navigation but less precision compared to Mag.

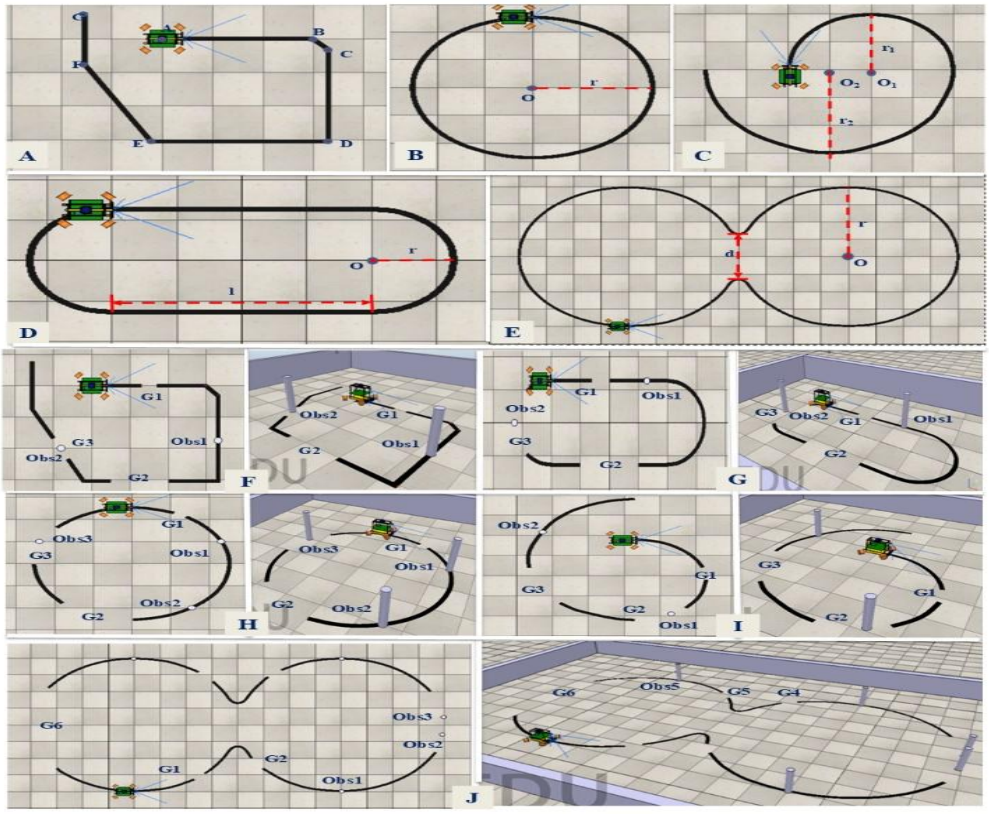

Figure 7: Simulation experiments Over paths : (A)–(E) Continuous paths – Path 1 (Rectangle), Path 2 (Circle), Path 3 (Playground), Path 4 (Spiral), and Path 5 (8-shape); (F)–(J) Complex paths with gaps (G1–G6) and obstacles (OBs1–OBs5) – Path 1 (Rectangle), Path 2 (Circle), Path 3 (Playground), Path 4 (Spiral), and Path 5 (8-shape)

Table 4: Robot performance of OMNI robot Simulation Step 1 on five continuous paths: P1 (Rectangle), P2 (Circle), P3 (Playground), P4 (Spiral), P5 (8-shape).

Sensor	Path Shape	Target Velocity	RT(s)	Average Velocity	Body Change			Error (cm)		
					Min	Max	Avg	Min	Max	Avg
M	P1	25	26.35	25.61	-3.137	3.139	0.49	0	14.438	2.836
	P2	25	30.66	26.09	-3.139	3.126	-0.035	0.024	3.005	1.092
	P3	25	32.15	24.88	-3.139	3.141	-0.105	0.069	7.95	1.531
	P4	25	30.21	25.99	-3.132	3.133	-0.031	0.038	17.76	1.682
	P5	25	85.1	26.44	-3.135	3.141	0.072	0	14.696	2.253
C (FAST)	P1	10	49.04	13.76	-3.138	3.139	-0.244	0.005	31.054	10.409
	P2	25	30.35	26.36	-3.14	3.14	-0.1	0.75	11.012	7.232
	P3	25	33.86	23.63	-3.14	3.138	-0.513	0.085	24.874	14.438
	P4	30	28.15	27.89	-3.13	3.122	-0.12	3.081	32.239	8.45
	P5	5	262.95	8.56	-3.141	3.14	0.122	0.014	32.069	7.105
C (SURF)	P1	30	23.4	28.85	-3.136	3.117	-0.178	0	28.265	9.093
	P2	30	28.96	27.62	-3.139	3.135	-0.029	0.574	9.791	6.099
	P3	25	29.9	26.76	-3.141	3.136	-0.442	0.176	27.71	12.584
	P4	30	27.83	28.21	-3.137	3.133	-0.023	2.351	32.158	7.357
	P5	5	251.48	8.95	-3.14	3.141	0.034	0	31.309	6.052
M + C (FAST)	P1	M 20, C 10	29.64	22.77	-3.136	3.137	0.328	0.005	14.132	2.738
	P2	M 25, C 20	30.34	26.37	-3.124	3.139	-0.052	0.068	3.568	1.316
	P3	M 25, C 15	32.9	24.32	-3.14	3.141	-0.149	0.154	6.656	1.321
	P4	M 25, C 10	30.6	25.65	-3.131	3.133	0.006	0.066	12.069	1.649
	P5	M 10, C 5	142.52	15.79	-3.138	3.135	0.012	0.064	14.699	2.194
M + C (SURF)	P1	M 20, C 20	30.6	22.06	-3.137	3.137	0.017	0.005	15.083	2.888
	P2	M 25, C 20	30.34	26.37	-3.137	3.125	-0.048	0.07	3.508	1.246
	P3	M 25, C 15	32.13	24.9	-3.137	3.141	-0.707	0.08	7.686	1.254
	P4	M 25, C 10	30.6	25.65	-3.135	3.128	-0.006	0.018	10.297	1.615
	P5	M 10, C 5	140.5	16.01	-3.141	3.134	0.017	0.016	14.845	2.175

Note: M = Magnet, C = Camera, L = LiDAR. The success rate for all experiments was 100%

Simulation Step 2: Cut Path (Rectangle): Based on the successful outcomes of the fusion approach in the previous tests, this step examined only sensor fusion. Two configurations were evaluated: (P2-1) fusion with FAST and (P2-2) fusion with SURF (Table 6). Sensor fusion produced clear improvements. The combination of Mag and Camera (FAST) maintained 100% success across all paths and significantly reduced errors compared to camera-only runs, such as an average error of 1.32 cm on P3 versus 14.44 cm with FAST alone. Velocity tracking was well-aligned with targets, balancing robustness and accuracy, demonstrating strong synergy between Mag and FAST for stable navigation. Similarly, the Mag and Camera (SURF) fusion achieved 100% success across all paths, with very low errors often approaching Mag-only performance, for example, 1.25 cm on P3 and 1.61 cm on P4. Running times and velocities were well-balanced, reflecting high consistency. Overall, this configuration offered the most stable performance, combining both accuracy (low errors) and reliability (100% success).

Table 5: Simulation Step 2: Fusion performance on cut Path 1 (Rectangle), Length = 675 cm.

Fusion Type	Config	RT (s)	AV (cm/s)	Body (rad)			Err (cm)		
				Min	Max	Avg	Min	Max	Avg
Fusion (FAST)									
M10,C20,L10	37.66	17.92	-3.1414	3.1415	0.351	0.0053	8.2831	2.4951	2.4951
M10,C20,L15	37.45	18.02	-3.1404	3.1349	-0.380	0.0529	9.3833	2.9656	2.9656
M10,C20,L20	37.60	17.95	-3.1413	3.1391	-0.368	0.0053	8.6476	2.4713	2.4713
M10,C20,L25	37.80	17.86	-3.1404	3.1348	-0.693	0.0053	9.3786	2.7836	2.7836
M10,C20,L30	37.45	18.02	-3.1414	3.1337	-0.688	0.0053	9.1887	2.4205	2.4205
Fusion (SURF)									
M10,C20,L10	39.15	39.15	-3.1349	3.1396	-0.749	0.0053	9.2422	2.5621	2.5621
M10,C20,L15	37.83	37.83	-3.1346	3.1406	-0.463	0.0053	8.6593	2.9168	2.9168
M10,C20,L20	37.40	37.40	-3.1366	3.1399	-0.484	0.0055	8.6176	2.6797	2.6797
M10,C20,L25	37.13	37.13	-3.1388	3.1357	-0.501	0.0053	8.5464	2.6926	2.6926
M10,C20,L30	37.24	37.24	-3.1415	3.1410	0.470	0.0054	8.4994	2.6964	2.6964
Note: M = Magnet, C = Camera, L = LiDAR. The success rate for all experiments was 100%									

Table 5 presents the results of the sensor fusion experiments on PATH 1 (Rectangle) with a path length of 675 cm and a 100% success rate. The table compares two fusion configurations, FAST and SURF, across varying LIDAR values while keeping magnetometer and camera settings constant.For Fusion (FAST), the target velocity ranges from 37.45 to 37.80 cm/s, with running times between 17.86 and 18.02 s. The body change values remain fairly consistent, with minimum values around -3.14 rad, maximums near 3.14 rad, and average body change ranging from -0.693 to 0.350

rad. The error metrics indicate average errors between 2.42 and 2.97 cm, while maximum errors reach approximately 8.28 to 9.38 cm. For Fusion (SURF), the target velocity is slightly higher for some settings (up to 39.15 cm/s), with running times matching the target velocities exactly, which might indicate a direct correlation in simulation. Body change metrics are comparable to FAST fusion, with minimum values around -3.13 rad, maximums near 3.14 rad, and average body change between -0.748 and 0.470 rad. Average errors are slightly higher or comparable, ranging from 2.56 to 2.92 cm, while maximum errors are in a similar range to FAST (8.5–9.24 cm).Comparing both fusion methods, FAST and SURF demonstrate similar trends in body change and maximum errors, but SURF exhibits slightly higher target velocities and slightly more consistent average errors in some LIDAR configurations.

Simulation Step 3: Cut Path with Obstacle: To further assess robot performance under more challenging conditions, this category introduced obstacles while testing the fusion algorithm. The two experiments included: (P3-1) fusion with FAST and (P3-2) fusion with SURF (Table 6).

Table 6: Fusion performance on cut paths with obstacles.

Path	NOO	PL	NG	NBG	RT	AV	Body			Error		
							Min	Max	Avg	Min	Max	Avg
Fusion (FAST)												
P1	2	675	2	1	54.81	12.32	-3.142	3.142	-0.313	0.01	40.65	7.84
P2	3	800	2	1	107.21	7.462	-3.138	3.142	0.166	0.04	44.26	11.86
P3	2	800	2	1	65.65	12.19	-3.139	3.140	-0.041	0.06	52.88	9.51
P4	2	785	2	1	61.60	12.74	-3.138	3.137	0.198	0.03	61.06	11.72
P5	5	2250	4	2	212.15	10.61	-3.138	3.138	-0.038	0.10	44.16	7.74
Fusion (SURF)												
P1	2	675	2	1	56.70	11.91	-3.134	3.141	0.168	0.01	41.85	8.94
P2	3	800	2	1	74.31	10.77	-3.135	3.132	-0.330	0.03	40.39	12.10
P3	2	800	2	1	68.23	11.73	-3.141	3.141	-0.725	0.05	44.54	8.68
P4	2	785	2	1	82.15	9.56	-3.135	3.138	0.276	0.03	50.52	8.92
P5	5	2250	4	2	210.40	10.69	-3.139	3.139	0.010	0.04	42.58	6.92

Note: NOO = Number of Obstacles, PL = Path Length (cm), NG = Number of Gaps, NBG = Number of Big Gaps, RT = Running Time (s), AV = Average Velocity (cm/s). The success rate for all experiments was 100%.

As the table shown, Both Fusion (FAST) and Fusion (SURF) achieved a 100\% success rate across all five paths, demonstrating robust navigation in obstacle-laden environments. For Fusion (FAST), the paths show average velocities ranging from 7.46 cm/s (P2) to 12.74 cm/s (P4). Running times increase with path length and obstacle complexity, with the longest runtime observed for P5 (212.15 s) due to its longer path and higher number of obstacles. Body changes remain consistent across all paths, with minimum and maximum values near ±3.14 rad, while average body change is small, indicating controlled maneuvering. Error metrics show average errors from 7.74 cm (P5) to 11.86 cm (P2), with maximum errors reaching up to 61.06 cm (P4), reflecting occasional deviations in more complex paths.For Fusion (SURF), average velocities are slightly lower on some paths (e.g., 9.56 cm/s on P4) but more consistent overall. Running times are generally similar to FAST, except P2 and P4, where SURF is faster (74.31 s and 82.15 s) compared to FAST (107.21 s and 61.6 s, respectively). Body change remains stable, comparable to FAST. Average errors are generally lower on SURF, ranging from 6.92 cm (P5) to 12.1 cm (P2), while maximum errors are slightly reduced in most paths, indicating improved stability during obstacle navigation.

B. Real Environment Experiments:

Following the simulation tests, a second set of experiments was conducted in a real-world environment to evaluate the robot's practical performance. The experiments were performed on a rectangular path with a total length of 680 cm (6.8 m), as shown in Figure 8.For all real-environment tests, the target operating speed of the AGV was systematically varied from 5 to 25 cm/s in increments of 5 cm/s for each sensor and fusion configuration (Tables 6–10), with the configuration Magnet 10, Camera 5, and LiDAR 10 cm/s (M10, C5, L10) selected as the reference speed combination for the final comparative and industrial-scenario experiments (Table 11), based on its superior performance identified in Experiment Steps 3–4. As it shown in Figure 8 the path consisted of seven station points, labeled A through F, with point A as the starting position and point F as the endpoint. Station B was further subdivided into two sub-points, B1 and B2. The path was marked with a green line placed on a white floor surface, enabling reliable detection by the onboard camera due to the strong contrast between the colors. Experiments were conducted indoors under artificial overhead LED lighting at an approximate illuminance of 300–400 lux, with no direct sunlight or significant shadowing on the test track. The guide line was a matte green tape applied to a matte white vinyl floor surface, chosen to maximize color contrast for camera-based detection. Occasional localized glare from the floor surface required adaptive HSV thresholding in the camera_base module to maintain reliable line segmentation under these lighting conditions. As it mentioned, the path consisted of seven station points: A, B1, B2, C, D, E, and F, with B1 and B2 representing two closely spaced sub-points located along the AB segment of the path (as shown in Figure 8). The distances between stations were measured as follows: A to B1 = 170 cm, B1 to B2 = 20 cm, B2 to C = 110 cm, C to D = 190 cm, D to E = 120 cm, and E to F = 70 cm. To further evaluate the robot's performance, a complex path test was designed by introducing four faults in the line. These included a 15 cm cut in the line (T1) requiring magnet and LiDAR detection, a 10 cm cut in the line (T2) also requiring LiDAR detection, a 20

cm segment without a magnetic marker (T3) expected to be detected by the camera, and a 13 cm cut in the line (T4) requiring LiDAR detection.The fault lengths (10–20 cm) were selected to exceed the effective detection distance of the AGS-16N magnetic sensor (5–55 mm), ensuring a reliable loss of magnetic signal at each fault and forcing a handover to the camera or LiDAR module. At the same time, the fault lengths were kept smaller than one chassis length (25 cm), allowing the robot to bridge each gap using odometric continuation while the alternate sensor re-acquired the path, thereby enabling a controlled comparison of the detection and recovery capability of each sensor modality. Prior to conducting the experiments, the environment was mapped using the file map.pgm, with a resolution of 0.05 meters per pixel. The map origin was set at coordinates (0.0, 0.0, 0.0), with Negate set to 0. Occupancy thresholds were defined such that cells with a probability greater than 0.65 were classified as occupied, while those below 0.196 were considered free.. Five experiments were designed as follows:

Real Experiment Step 1 (Continuous Line Tracking): This experiment focused on determining the boundary output speed and identifying the optimal output speed for each individual sensor. Baseline performance data for each sensor was obtained (Table 7).

As shown in Table 7, the performance metrics obtained from real experiments on a continuous rectangular path using the Magnet (MAG), Camera, and LiDAR sensors at different target speeds are presented. The Magnet (MAG) sensor demonstrates high reliability at low speeds, achieving a **100%** success rate at target speeds of **5** and **10 cm/s**. Its average velocity closely matches the target velocity, indicating stable motion control. However, the tracking error increases with speed, rising from **1.95 cm** at **5 cm/s** to **7.85 cm** at **20 cm/s**. The success rate gradually decreases at higher speeds, dropping to **60%** at **20 cm/s** and **0%** at **25 cm/s**, highlighting its limitations under high-speed operating conditions. The body orientation change remains relatively stable, with average values ranging from approximately **−0.678** to **−0.737**, indicating consistent maneuverability. The Camera sensor performs reliably only at the lowest target speed of **5 cm/s**, where it achieves a **100%** success rate. However, it fails to complete the experiment at **10 cm/s**, resulting in a success rate of **0%**. Although its average velocity closely follows the target speed at **5 cm/s**, the average tracking error of **3.5 cm** is higher than that of the Magnet sensor. Furthermore, the absence of successful experiments at higher speeds indicates that the Camera sensor is not suitable for fast robot motion, most likely because of image processing and feature-tracking limitations.

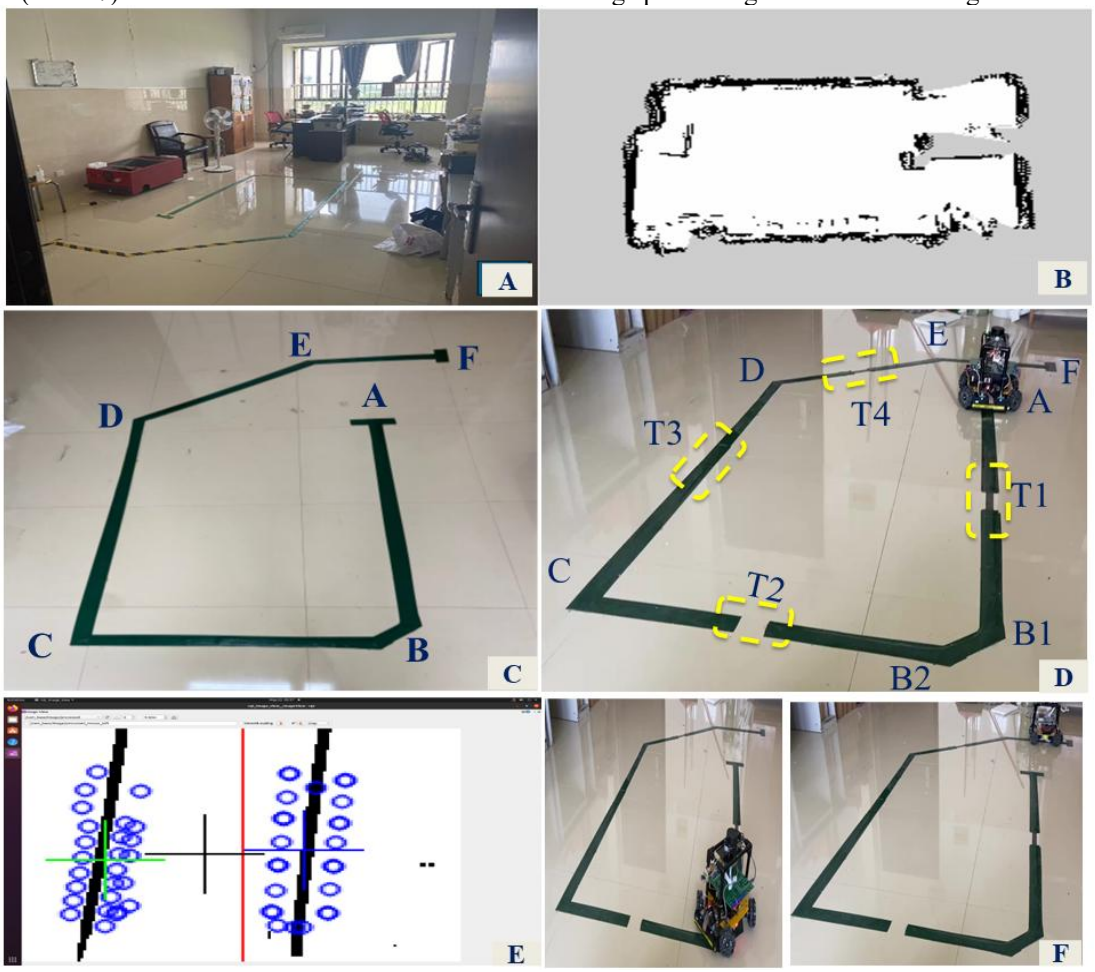

Figure 8: Real-environment robot performance tests were conducted in a controlled indoor setting. Figure X illustrates the experimental setup: (A) the overview of the indoor test area, (B) the map generated using GMapping, (C) the rectangular path with a total length of 680 cm (6.8 m), and (D) the complex rectangular path, also measuring 680 cm (6.8 m). Within the complex path, four gap segments were introduced between points A and B to evaluate sensor responses: T1 = 15 cm, T2 = 10 cm, T3 = 20 cm, and T4 = 13 cm. (E)The line feature detected by camera(Surf) F:The robot performance over EFM2.

Table 7: Real Experiment Step 1: Performance on rectangle continuous path for different sensors at varying target speeds

Sensor	Ave RT (s)	AV Speed (cm/s)	Body (Deg)			Err (cm)		
			Min	Max	Avg	Min	Max	Avg
Magnet								
M5	117	5.94	-1.194	0.010	-0.678	0	2.75	1.95
M10	62	11.20	-1.341	0.031	-0.703	0	4.35	1.85
M15	48	14.47	-1.373	0.049	-0.713	0	8.15	6.25
M20	40	16.54	-1.380	0.009	-0.737	0	9.65	7.85
M25	---	---	---	---	---	---	---	---
Camera								
C5	108	6.43	-1.227	0.016	-0.621	0	7.50	3.50
C10	---	---	---	---	---	---	---	---
LiDAR								
L5	107	6.49	-1.234	0.0004	-0.652	0	10	4.5
L10	65	10.69	-1.228	0.0070	-0.685	0	16	12
L15	50	13.90	-1.218	0.0170	-0.712	0	48	22
L20	42	16.54	-1.227	0.0330	-0.721	0	80	65
L25	42	16.54	-1.268	0.0010	-0.763	0	115	85
Note: M = Magnet, C = Camera, L = LiDAR. The success rate for all experiments was 100%.								

The LiDAR sensor demonstrates reliable performance at low and moderate speeds, achieving success rates between **80%** and **100%** for target speeds ranging from **5** to **15 cm/s**. Nevertheless, its tracking error increases significantly as the robot speed increases, reaching **65 cm** at **20 cm/s** and **85 cm** at **25 cm/s**. Although the average velocity remains close to the desired value at lower speeds, its accuracy gradually deteriorates at higher speeds. In contrast, the body orientation change remains relatively consistent, with average values ranging from approximately **−0.652** to **−0.763**, indicating stable robot motion despite the increase in tracking error.

Experiment Step 2 (Sensor Speed Combination and Fusion): In this stage, the output speeds of individual sensors were combined to construct multiple fusion algorithms with different parameters, and the corresponding data were collected. The results showed that multi-sensor data fusion significantly outperformed individual sensors. However, the speed combinations of different modules were not always optimal, and in some cases, expected speed contributions had no effect on specific algorithm modules. This finding led to the design of Experiment 3, which refined the speed combination Table 8.

Table 8: Experiment 2 for the Fusion Algorithm: Continuous path of 695 cm with the same speed for all sensors

Sensor	Ave RT (s)	AV Speed (cm/s)	Body (deg)			Err(cm)		
			Min	Max	Avg	Min	Max	Avg
M5, C5, L5	111	6.26	-1.269	0.058	-0.616	0	2.25	1.00
M10, C10, L10	60	11.58	-0.269	0.011	-0.620	0	3.75	1.50
M15, C15, L15	45	15.44	-1.260	0.024	-0.661	0	7.50	5.50
M20, C20, L20	38	18.20	-1.338	0.015	-0.725	0	9.00	7.25
M25, C25, L25	38	18.20	-1.226	0.081	-0.771	0	10.25	8.00
Note: M = Magnet, C = Camera, and L = LiDAR. The success rate for all experiments was 100%.								

Table 8 presents the performance of the fusion algorithm on a continuous 695 cm path, with all sensors (Magnet, Camera, Lidar) operating at the same target speeds. The results show that at low speeds (5 and 10 cm/s), the fusion algorithm achieves perfect performance, maintaining a 100\% success rate, average speed close to the target, minimal errors (1–1.5 cm average), and stable body changes (average around -0.616 to -0.620 rad). As the target speed increases to 15 cm/s, performance begins to decline: the average error rises to 5.5 cm, and success rate drops to 80\%, although body change remains relatively stable. At 20 cm/s, the fusion algorithm exhibits further degradation with a 7.25 cm average error and 60\% success rate, while at 25 cm/s the average error reaches 8.0 cm and the success rate falls to 40\%.

Real Experiment Step 3 (Optimal Fusion Algorithm): This experiment further tested the fusion algorithms to determine the most effective speed combination for sensor integration. An effective set of module speeds was identified, though further validation was needed to confirm its optimality Table 9.As it shown, the performance of individual sensors on a continuous path of 695~cm is summarized. The Magnet sensor (M10), operating at a target speed of 10~cm/s, achieved excellent performance by completing the path in 60~s with an average velocity of 11.58~cm/s. It maintained stable body orientation, with an average body change of approximately -0.619, while achieving a low average tracking error of 1.25~cm and a success rate of 100%. In contrast, no experimental results are available for the Camera sensor (C5) at 5~cm/s or the LiDAR sensor (L10) at 10~cm/s, indicating that these experiments were either not performed or their results were not recorded.

Table 9: Experiment 3 for the Fusion Algorithm: Continuous path (695 cm)

Sensor	Target Speed (cm/s)	PL (cm)	RT (s)	AV (cm/s)	Body			Err		
					Min	Max	Avg	Min	Max	Avg
M10	10	695	60	11.58	-1.235	0.019	-0.619	0	3.50	1.25
C5	5	695	---	---	---	---	---	---	---	---
L10	10	695	---	---	---	---	---	---	---	---

Note: *PL = Path Length, RT = Running Time, AV = Average Velocity. M = Magnet, C = Camera, and L = LiDAR. The success rate for all experiments was 100%.*

Real Experiment Step 4 (Fusion Algorithm with Increased Speed): Building on Experiment 3, this step verified whether the identified speed combination was truly optimal by increasing each sensor's speed by 5 cm/s and collecting new experimental data. The modified configuration provided further insight into system performance under adjusted speed conditions Table 10.

Table 10: Experiment 5 for the Fusion Algorithm (M10, C5, L10).

Sensor	Target Speed (cm/s)	RT (s)	AV (cm/s)	Body			Err		
				Min	Max	Avg	Min	Max	Avg
M15, C10, L15	**15**	**47**	**14.78**	**-1.244**	**0.022**	**-0.651**	**0**	**6.50**	**4.75**

Note: RT = Running Time, AV = Average Velocity. M = Magnet, C = Camera, and L = LiDAR. The success rate for all experiments was 100%.

Table 10 presents the results for the fusion algorithm at increased speeds over a 695 cm continuous path. The combined sensors (Magnet 15, Camera 10, Lidar 15) achieved an average speed of 14.78 cm/s, completing the path in 47 seconds. Body changes remained stable, with an average of -0.6513 degrees, while errors were moderate, with an average of 4.75 cm. The success rate was 85%, indicating that the higher speed slightly reduced reliability compared to lower-speed experiments, but the fusion approach still maintained acceptable navigation accuracy and stability.

Real Experiment Step 5 (Cut Path – Industrial Scenario Simulation): The final experiment tested the best-performing fusion algorithm in a more complex environment simulating real industrial conditions. Sections of the guiding line were deliberately removed to assess robustness. As expected, single sensor algorithms failed under these conditions, while the Fusion algorithm continued to operate normally, demonstrating strong adaptability in complex scenarios Table 11.

Table 11: Experiment 5 for the Fusion Algorithm (M10, C5, L10).

Path	L (cm)	RT (s)	AV (cm/s)	Body Min	Body Max	Body Avg	Err Min	Err Max	Err Avg
P	**695**	**60**	**11.58**	**-1.235**	**0.019**	**-0.619**	**0**	**3.50**	**1.25**
CP	**695**	**63**	**11.03**	**-1.273**	**0.021**	**-0.627**	**0**	**3.50**	**1.25**

Note: P = Continuous Path, CP = Cut Path, L = Length, RT = Running Time, and AV = Average Velocity. The sensor speeds were fixed at M10, C5, and L10. The success rate for all experiments was 100%.

Table 11 presents the performance of the fusion algorithm in two different scene types broken and complete over a 695 cm continuous path with sensors set at Mag 10, Cam 5, and Lidar 10. Both scenarios achieved a 100% success rate. The average running time was slightly higher for the broken scene (63 s) compared to the complete scene (60 s), resulting in a slightly lower average speed (11.03 cm/s vs. 11.58 cm/s). Body changes were minimal and nearly identical between the two scenes, and the error metrics were also the same, with an average error of 1.25cm. Overall, the Fusion Algorithm demonstrated robust and consistent navigation performance regardless of minor path irregularities, maintaining accuracy and full reliability. This section presents the performance evaluation of the tested configurations in both the practical and simulation datasets, followed by comparative analysis, uncertainty assessment through Monte Carlo simulations, and configuration grouping via clustering. The evaluation process began by applying the selected sensor fusion and feature extraction algorithms to both the practical and simulation datasets. Each dataset contained multiple configurations of sensors and processing parameters, such as variations in magnetometer, camera, and LiDAR settings for the practical tests, and different combinations of feature detectors and descriptors (e.g., FAST, SURF) for the simulation runs. All configurations were subjected to the same pipeline: raw sensor data were first preprocessed, features were extracted using the assigned method, and then the Weighted Overall Performance Index (WOPI) was computed as the aggregate measure of quality.

C. WOPI Index performance:

As the second evaluation metric in this study, the Weighted Overall Performance Index (WOPI) was calculated for each configuration according to the method described in Algorithm 2. The WOPI values from all test runs were then averaged to obtain the mean WOPI score. In this section, two approaches are investigated: evaluating the impact of conventional performance metrics and performing additional analyses to compare practical and simulation results, assess result stability using Monte Carlo simulations, and group configurations with similar performance profiles through clustering. For the first approach, rankings were obtained by ordering the configurations from the highest to lowest mean WOPI values. Further analyses were then conducted to compare practical versus simulation results, evaluate results stability via Monte Carlo simulations, and identify clusters of configurations with similar performance characteristics. This structured approach ensures that the reported results are directly comparable across datasets and that the conclusions are supported by both quantitative metrics and statistical robustness checks. The simulation and real-world datasets were resampled to 1,000 instances, and the WOPI formula was evaluated using Monte Carlo simulations. The optimized weights were calculated based on four normalized component metrics: velocity (f_1), inverse error (f_2), success rate (f_3), and stability (f_4), with weights w_1, w_2, w_3, w_4 optimized to maximize the separation between high- and low-ranked configurations within each dataset. The resulting optimized coefficients are presented in Table 12.

Table 12: Optimized WOPI weights for each dataset.

Dataset	w_1 (Velocity)	w_2 (Inverse Error)	w_3 (Success)	w_4 (Stability)
Practical	0.28	0.32	0.24	0.16
Simulation	0.25	0.35	0.20	0.20

These results indicate that, for both datasets, **Inverse Error (w_2)** is the most influential metric in determining WOPI, followed by **Velocity (w_1)**. **Success rate** and **stability**, while less heavily weighted, still contribute to overall performance.

Table 13: Comparison of configurations ranked by mean WOPI for both practical and simulation datasets.

Simulation Dataset	Mean WOPI	Rank	Practical Dataset	Mean WOPI	Rank
M10+C5 (P:SURF)	0.9023	1	M5, C10, L5 (P, CP:SURF)	0.9278	1
M25+C10 (P:SURF)	0.8987	2	M5, C5, L5 (P:SURF)	0.9198	2
M10+C5 (P:FAST)	0.8983	3	M25, C10, L10 (P:SURF)	0.9135	3
M25+C10 (P:FAST)	0.8967	4	M10, C10, L10 (P:SURF)	0.8871	4
M10, C10, L10 (CP:FAST)	0.8591	5	M15, C10, L15 (P:SURF)	0.8870	5
C10 (P:SURF)	0.8578	6	M15, C15, L15 (P:SURF)	0.8869	6
M10, C10, L15 (CP:SURF)	0.8553	7	M10 (P)	0.7600	7
C10 (P:FAST)	0.7084	8	C5 (P:SURF)	0.7195	8
M10 (P)	0.7074	9	L10 (P)	0.5432	9
---	---	--	M20, C20, L20 (P:SURF)	0.3569	10
Note: M = Magnet, C = Camera, L = LiDAR, P = Continuous Path, and CP = Cut Path.					

Table 13 presents a detailed comparison of sensor configurations ranked according to their mean Weighted Overall Performance Index (WOPI) for both simulation and practical datasets. The WOPI metric integrates multiple performance factors, including velocity, inverse error, success rate, and stability, to provide a comprehensive assessment of each configuration's effectiveness. In the simulation dataset, the top-ranked configurations, such as M10+C5 (P: SURF) and M25+C10 (P: SURF), achieved the highest WOPI scores, indicating superior overall performance across all evaluated metrics. These configurations combine multiple sensors and optimized path strategies, contributing to enhanced detection, tracking, and navigation performance. Mid-ranked configurations, such as M10, C10, L10 (CP: FAST), show moderate performance, while single-sensor or minimally configured setups, like C10 (P: FAST) and M10 (P), yield the lowest WOPI values, highlighting the limitations of reduced sensing capability in simulation. For the practical dataset, top-performing configurations, including M5, C10, L5 (P, CP: SURF) and M5, C5, L5 (P: SURF), achieve the highest mean WOPI scores, reflecting the effectiveness of combining multiple sensors under real-world conditions. Unlike simulation results, certain configurations, such as M10 (P), perform comparatively lower in practice due to real-world variability, sensor noise, and environmental complexity. Lower-ranked practical configurations, including L10 (P) and M20, C20, L20 (P:SURF), demonstrate reduced performance, emphasizing the influence of sensor selection and path planning on practical WOPI outcomes. The comparison between simulation and practical datasets reveals both consistencies and differences. High-ranking configurations in simulation generally maintain strong performance, indicating robustness of multi-sensor combinations in practical scenarios. However, some mid- to low-ranked configurations showed deterioration in real-world conditions, due to the effects of environmental nonlinearities.

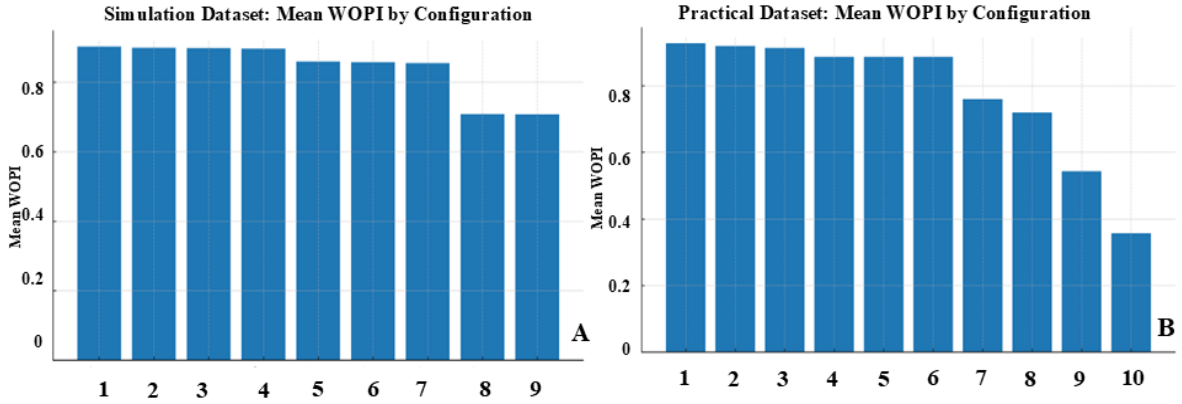

Figure 9: The WOPI reasult over the robot perfromance (A): Simulation data Mean WOPI score 1:Magnet10+Camera5 (P:SURF), 2:Magnet25+ Camera10 (P:SURF), 3:Magnet10+Camera5 (P:FAST), 4:Magnet25+ Camera10 (P:FAST), 5:Magnet10, Camera10, Lidar10 (P:FAST), 6:Camera10 (P:SURF), 7:Magnet10, Camera10, Lidar15 CameraCameraP (SURF), 8:Camera10 (P:FAST), 9:Magnet 10 (P). (B): Practical data Mean WOPI scores, 1:Magnet5, Camera10, Lidar5 (P,SURF), 2:Magnet5, Camera5, Lidar5 (P:SURF), 3:Magnet25, Camera10, Lidar10 (P:SURF), 4:Magnet10, Camera10, Lidar10 (P:SURF), 5:Magnet15, Camera10, Lidar15 (P:SURF), 6:Magnet15, Camera15, Lidar15 (P:SURF), 7:Magnet10(P), 8:Camera5 (P:SURF), 9:Lidar10(P), 10:Magnet20, Camera20, Lidar20 (P:SURF)

Multi-sensor configurations with a combination of continuous (P) and cut (CP) paths tend to perform consistently well across both datasets, suggesting these setups provide greater robustness against real-world uncertainties than configurations with fewer sensors or suboptimal path strategies. Overall, the table provides a comprehensive overview of fusion sensor combinations, path types, and experimental context influence system performance as measured by WOPI. While simulation offers a controlled benchmark for evaluation, practical experiments reveal the nuances and challenges of deploying these configurations in real-world conditions. Figure 10 illustrates the mean WOPI scores of corresponding configurations across the practical and simulation datasets.

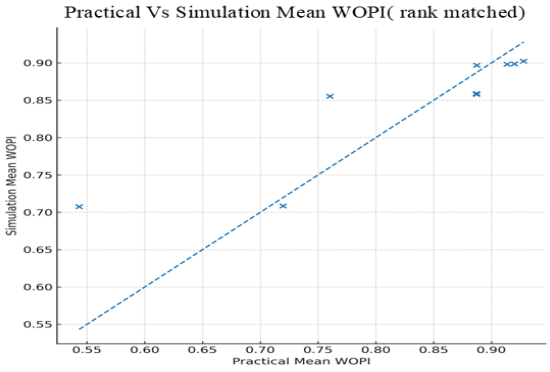

Figure 10: Comparison of mean WOPI scores for each configuration in both practical and simulation.

Overall, the ranking trends are consistent: top-performing configurations in the practical dataset, such as M5,C10,L5 and M5,C5,L5, also achieve high scores in simulation, while lower-performing configurations like M20,C20,L20 and Lidar remain at the bottom in both evaluations. The close alignment with the identity line indicates strong agreement between the two evaluation modes. Monte Carlo simulations (Figure 11A and B) indicate that top configurations such as M5,C10,L5 in the practical dataset and M10,C5 (P:SURF) in the simulation dataset show tight, high-valued distributions, suggesting stable performance under small perturbations. Lower-ranked configurations like M25,C25,L25 and M10 exhibit wider spreads and lower central values. Clustering analysis (Figure 11C and D) groups configurations by performance. In the practical dataset, top clusters are dominated by M5,C10,L5, M5,C5,L5, and M10,C10,L10, whereas in the simulation dataset, the best clusters contain M10,C5 (P:SURF) and M25+C10 (P:SURF). Lower clusters in both datasets contain underperformers like M20,C20,L20 in practical and M10 in simulation.

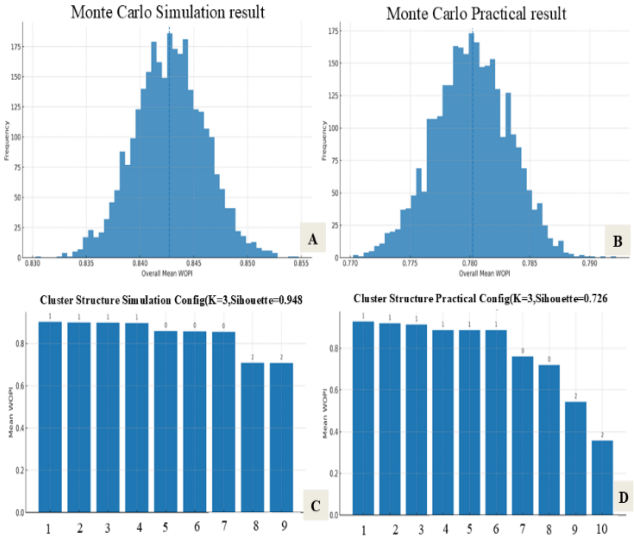

Figure 11: Monte Carlo and Cluster analysis (A) Monte Carlo simulation for the practical data (B) Monte Carlo simulation for the Simulation data, (C): Cluster analysis of configuration in simulation data, 1: Magnet10 + Camera5 (P:FAST), 2: Magnet25 + Camera10 (P:SURF), 3: Magnet10 + Camera5 (P:FAST), 4: Magnet25 + Camera10 (P:FAST), 5: Magnet10, Camera10, Lidar10 (CameraP:FAST) , 6: Camera10 (P:SURF), 7: Magnet10, Camera10, Lidar15 (P:SURF), 8: Camera10 (P:FAST), 9: Magnet 10 (P).(D) Cluster analysis of configration in practical data, 1: Magnet5, Camera10, Lidar5 (P:SURF), 2: Magnet5, Camera5, Lidar5 (P:SURF), 3: Magnet25, Camera10 , Lidar10(P:SURF), 4: Magnet10, Camera10 , Lidar10 (P:SURF), 5: Magnet15, Camera10 , Lidar15 (P:SURF), 6: Magnet15, Camera15 , Lidar15 (P:SURF), 7: Magnet10 (P), 8: Camera5 (P:SURF), 9: Lidar10 (P), 10: Magnet20, Camera20 , Lidar20 (P:SURF)

IV. CONCLUSION

Omni robot platforms represent a new generation of robots that are increasingly being adopted across various applications due to their flexibility, maneuverability, and adaptability. In this study, we designed and tested an omni robot platform to investigate the performance of a novel fusion algorithm, termed as EFM2, developed as a novel algorithm for sensor fusion, designed to prioritize inputs from three different sensors: the Magnet, the Camera (integrated with object detection algorithms such as SURF and SIFT), and the Lidar (utilizing SLAM). By combining and weighting these sensor modalities, EFM2 enables the robotic platform to achieve faster and more timely responses in dynamic environments. A series of experiments, conducted in both simulation and real-world environments, were performed to

evaluate the effectiveness of the EFM2 algorithm and to validate its advantages over conventional sensor fusion approaches. To evaluate the EFM2 algorithm, two complementary approaches were considered. First, the robot's performance was tested in simulation across five different paths (PATH 1 (Rectangle), PATH 2 (Circle), PATH 3 (Playground), PATH 4 (Spiral), and PATH 5 (8-shape)), including both continuous and cut paths, with and without obstacles. This allowed for a controlled assessment of adaptability under varying conditions. Second, to approximate real-world industrial applications, experiments were conducted on a rectangular path in a real environment, representing the commonly used line following patterns in industrial settings.In order, to assess the performance beyond conventional metrics such as Target Velocity, Running Time, Average Velocity, Body Change (minimum, maximum, and average), and Error (minimum, maximum, and average) a new composite index, the Weighted Overall Performance Index (WOPI), was introduced. This index aggregates the contributions of all conventional metrics into a single measure, providing a more comprehensive evaluation of system performance. The WOPI combines multiple robot performance metrics velocity, accuracy, stability, and task success into a single score. Each metric is normalized and assigned a weight reflecting its importance, determined through expert input and optimization. The final WOPI value provides a holistic measure of system effectiveness, enabling fair comparison across trials and configurations. Then, using Monte Carlo simulation, 1,000 data samples were resampled to account for variability in sensor behavior and path conditions. For each resampled instance, WOPI was calculated, and the sensor configurations were ranked based on their aggregated performance. The results from both real-world tests and simulations indicate that the configuration combining a Magnet sensor at speed 10, a Camera at speed 5 using SURF object detection, and a Lidar at speed 10 achieved the best overall performance on both continuous and cut paths. This ranking is further confirmed when evaluated using the WOPI metric.The performance on conventional parameters was analyzed across multiple paths and scenarios. In the first simulation step, continuous path experiments showed that the Magnetometer (Mag) alone exhibited excellent reliability, achieving a 100\% success rate and stable velocity tracking, although errors slightly increased on more complex paths. Camera-only setups were less robust, with FAST demonstrating low success rates and higher average errors, while SURF performed better but remained less accurate than Mag. Sensor fusion significantly improved performance; the combination of Mag and FAST achieved 100\% success with reduced errors, though longer runtimes were observed on very long paths, whereas Mag and SURF fusion provided the most stable and accurate navigation, maintaining consistently low errors, reliable velocity tracking, and full success across all paths. In the second simulation step, both fusion methods (FAST and SURF) achieved 100\% success and low average errors, with SURF fusion showing slightly more consistent errors and marginally higher target velocity tracking, indicating better overall stability and precision. In the third simulation step, cut-path experiments with obstacles revealed that FAST fusion achieved slightly higher peak velocities, but SURF fusion maintained more consistent speed, lower errors, and smoother trajectories, offering the best balance between speed, accuracy, and reliability for complex paths. Real-world experiments further confirmed these findings. In continuous line-tracking experiments, the Magnet (MAG) sensor demonstrated high reliability at moderate speeds but failed under high-speed conditions. The Camera sensor was effective only at very low speeds, whereas the LiDAR sensor performed satisfactorily at low and moderate speeds but exhibited significantly larger tracking errors as the speed increased. Overall, increasing the target speed resulted in higher tracking errors and lower success rates for all individual sensors, highlighting the need for sensor fusion or more advanced control strategies. The sensor fusion experiments showed that combining data from multiple sensors significantly improved navigation accuracy, reliability, and stability at low and moderate speeds. Although the fusion algorithm maintained excellent performance under these conditions, its accuracy gradually declined at higher speeds, demonstrating the inherent challenges of high-speed autonomous navigation. The optimal fusion configurations further indicated that the Magnet sensor provided highly reliable and accurate performance on specific paths, while multi-sensor fusion ensured smoother and more

robust navigation over a wider range of operating conditions. Additional experiments conducted at increased target speeds confirmed that the proposed fusion algorithm could maintain stable navigation with only a slight degradation in performance.

In the cut-path industrial scenario simulations, the fusion configuration consisting of Magnet~10, Camera~5, and LiDAR~10 achieved a 100\% success rate on both continuous and cut paths. The results showed only minor differences in average speed, running time, body orientation, and positional error, demonstrating the robustness and accuracy of the proposed fusion approach even in environments containing path discontinuities and irregularities. Overall, the experimental results demonstrate that sensor fusion consistently outperforms individual sensors in terms of tracking accuracy, reliability, and navigation stability. Among the evaluated methods, the combination of the Magnet sensor with SURF-based fusion achieved the best overall performance in both simulation and real-world experiments. While the Magnet sensor alone is sufficient for relatively simple or moderate navigation tasks, camera-only configurations, particularly those using FAST, are not recommended because of their limited robustness. Therefore, the proposed multi-sensor fusion strategy provides a more reliable and stable solution for autonomous navigation under diverse operating conditions, although high-speed navigation remains a challenging problem that requires further investigation. Weighted Overall Performance Index (WOPI) framework has proven to be an effective and reliable method for benchmarking robotic path planning performance across diverse sensor configurations. The results showed that in the practical dataset, the top-performing configuration (mag 10 cam 5 lidar 10) achieved a mean WOPI of 0.928, while in the simulation dataset, M10+C5 (P: SURF) reached a mean WOPI of 0.902, with both displaying stable distributions under Monte Carlo analysis. Conversely, lower-ranked setups such as M20, C20, L20 (P: SURF) and standalone Lidar sensors consistently underperformed, with mean WOPI scores below 0.36 and 0.54, respectively. These performance trends were further reinforced through clustering analysis, where high-performing configurations grouped distinctly from weaker ones, and through weight optimization, which highlighted inverse error and velocity as the most influential contributors to overall performance. The strong alignment between simulation and real-world evaluations underscores the framework's robustness and generalizability. Collectively, these findings not only validate the WOPI approach as a comprehensive metric for robotic navigation assessment but also provide actionable insights for selecting and optimizing sensor configurations in future applications. During the experiment especially in the real experiment section several challenges were encountered during the real-world trials. First, lighting variations occasionally affected the camera's ability to accurately detect the green tape, especially under reflective conditions, requiring adaptive thresholding adjustments. Second, slight misalignment in the placement of the magnetic induction bar introduced small errors in detection, necessitating frequent recalibration. Mechanical wear in the drive system introduced inconsistencies in velocity control, leading to gradual drift in straight sections. Finally, while the current tests were conducted without obstacles, real-world deployments will require robust obstacle detection and avoidance, which remains an important focus for future work. Nevertheless, the calculated WOPI indicated close alignment between simulation predictions and real-world outcomes, validating the robustness of the EFM2 fusion algorithm. Experimental evaluations demonstrated that EFM2, when assessed through WOPI, delivers superior performance by effectively balancing competing performance factors, making it a robust candidate for advanced robotic applications in complex environments.Future work will focus on several directions. First, the weighting strategy in WOPI could be dynamically optimized through learning-based approaches, allowing the index to adapt to different operational contexts and mission priorities. Second, extending EFM2 with deep learning-based perception modules could enhance the system's ability to handle unstructured environments and adapt to real-world uncertainties. Third, although the current evaluations were promising, large-scale real-world testing is needed to further validate robustness under diverse and unpredictable conditions. Finally, integrating additional sensing modalities, such as inertial measurement units (IMUs) or

event-based cameras, along with advanced optimization techniques for parameter tuning, may further enhance both navigation accuracy and overall robotic efficiency. By combining WOPI's holistic evaluation framework with EFM2's enhanced fusion architecture, this research lays a strong foundation for the next generation of performance-aware, sensor-driven robotic systems capable of thriving in increasingly complex and dynamic operational environments. While the results validate the applicability and effectiveness of both WOPI and EFM2, there remains significant scope for further research and refinement. Future work will focus on several directions. First, the weighting strategy in WOPI could be dynamically optimized through learning-based approaches, allowing the index to adapt to different operational contexts and mission priorities. Second, extending EFM2 with deep learning-based perception modules could enhance the system's ability to handle unstructured environments and adapt to real-world uncertainties. Third, although the current evaluations were promising, integrating additional sensing modalities, such as inertial measurement units (IMUs) or event-based cameras, along with advanced optimization techniques for parameter tuning, may further enhance both navigation accuracy and overall robotic efficiency.

V. REFERENCES

[1] El Mastor H. Impact of automated guided vehicles on logistics operations in manufacturing enterprise "X". Ph.D. thesis, Kauno technologijos universitetas., 2025.

[2] Moshayedi AJ, Jinsong L, Liao L. AGV (Automated Guided Vehicle) Robot: Mission and Obstacles in Design and Performance. Journal of Simulation and Analysis of Novel Technologies in Mechanical Engineering 2019 12(4):5–18.

[3] Karur K, Sharma N, Dharmatti C, Siegel JE. A Survey of Path Planning Algorithms for Mobile Robots. Vehicles 2021 3(3):448–468.

[4] Moshayedi AJ, Roy AS, Sambo SK, Zhong Y, Liao L. Review on: The service robot mathematical model. EAI Endorsed Transactions on AI and Robotics 2022 1(1):1–19.

[5] Grilo A, Costa R, Figueiras P, Gonçalves RJ. Analysis of AGV indoor tracking supported by IMU sensors in intra-logistics process in automotive industry. In 2021 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC), IEEE2021 pp. 1–7.

[6] Shuo S. Multi-AGV Path Planning Method via Reinforcement Learning and Particle Filters. arXiv preprint arXiv:2403.18236 2024.

[7] Moshayedi AJ, Li J, Liao L. Simulation study and PID tune of automated guided vehicles (AGV). In 2021 IEEE international conference on computational intelligence and virtual environments for measurement systems and applications (CIVEMSA), IEEE2021 pp. 1–7.

[8] Lee SY, Yang HW. Navigation of automated guided vehicles using magnet spot guidance method. Robotics and Computer-Integrated Manufacturing 2012 28(3):425–436.

[9] Jun YU, Xing WU, Weiliang S. Development of omnidirectional vision- guided AGV based on Mecanum wheel. Machine Design and Manufacturing Engineering 2015 .

[10] Jiaxin S, Zhiming Z, Yongqing S, Zheng Z. Design and Implementation of Indoor Positioning and Navigation System of Mobile Robot Based on ROS and Lidar. Mach. Electron 2018 36(2018):76–80.

[11] Su Y, Wang T, Shao S, Yao C, Wang Z. GR-LOAM: LiDAR-based sensor fusion SLAM for ground robots on complex terrain. Robotics and Autonomous Systems 2021 140:103759.

[12] Zhong H, Wang H, Wu Z, Zhang C, Zheng Y, et al. A survey of LiDAR and camera fusion enhancement. Procedia Computer Science 2021 183:579–588.

[13] Tran TQ, Becker A, Grzechca D. Environment mapping using sensor fusion of 2d laser scanner and 3d ultrasonic sensor for a real mobile robot. Sensors 2021 21(9):3184.

[14] Li, J., Hu, B., & Noland, T. L. (2013). Classification of tree species based on structural features derived from high density LiDAR data. *Agricultural and forest meteorology*, *171*, 104-114.

[15] Chen X, Zhang T, Wang Y, Wang Y, Zhao H. Futr3d: A unified sensor fusion framework for 3d detection. In proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2023 pp. 172–181.

[16] Li Y, Yu AW, Meng T, Caine B, Ngiam J, et al. Deepfusion: Lidar-camera deep fusion for multi-modal 3d object detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022 pp. 17182–17191.

[17] Obando-Ceron JS, Romero-Cano V, Monteiro S. Probabilistic multi-modal depth estimation based on camera–LiDAR sensor fusion. Machine Vision and Applications 2023 34(5):79.

[18] Peng G, Zhou Y, Hu L, Xiao L, Sun Z, et al. Vilo SLAM: Tightly coupled binocular vision–inertia SLAM combined with LiDAR. Sensors 2023 23(10):4588.

[19] Lan Z, Wang J, Shen Z, Fang Z. Highly robust and accurate multi-sensor fusion localization system for complex and challenging scenarios. Measurement 2024 235:114851.

[20] Van Nam D, Danh PT, Park CH, Kim GW. Fusion consistency for industrial robot navigation: An

integrated SLAM framework with multiple 2D LiDAR-visual-inertial sensors. Computers and Electrical Engineering 2024 120:109607.
[21] Dhanush D, Raghul K, Shri Ramya K, Mukund P, Ganesan M. An omni-directional self driving robot for indoor surveillance. In 2024 International Conference on Signal Processing, Computation, Electronics, Power and Telecommunication (IConSCEPT), IEEE2024 pp. 1–6.
[22] Zhu J, Li H, Zhang T. Camera, LiDAR, and IMU based multi-sensor fusion SLAM: A survey. Tsinghua Science and Technology 2023 29(2):415–429.
[23] Yusefi A, Durdu A, Toy I. Camera/LiDAR Sensor Fusion-based Autonomous Navigation. In 2024 23rd International Symposium INFOTEH-JAHORINA (INFOTEH), IEEE2024 pp. 1–6.
[24] Koshy AS. Multi-Sensor Fusion for Localization of a Lunar Micro Rover Using Non-Vision Sensors.
[25] Khan, A. S., & Zhang, J. (2025). Lightweight AI for Cultural Pattern Recognition: Safeguarding South Asian Regional Embroidery Heritage. In *Artificial Intelligence and Applications*.
[26] Bhargava A, Suhaib M, Singholi AS. A review of recent advances, techniques, and control algorithms for automated guided vehicle systems. Journal of the Brazilian Society of Mechanical Sciences and Engineering 2024 46(7):419.
[27] Pham TMT, Yang B, Yang J. Evaluating the robustness of lidar-based 3d obstacles detection and its impacts on autonomous driving systems. arXiv preprint arXiv:2408.13653 2024.
[28] Moshayedi AJ, Xie Y, Sharifdoust M, Khan AS. Evaluating OMNI Robot Navigation with SLAM in CoppeliaSim: Hemangiomas and Nonhomogeneous Paths. Journal of Robotics Research (JRR) 2024 1(1):7–14.
[29] Xu H, Li Y, Lu Y. Research on indoor AGV fusion localization based on adaptive weight EKF using multi-sensor. In Journal of Physics: Conference Series, IOP Publishing2023 p. 012028.
[30] Wilts T, Badri-Hoeher S. Enhanced state estimation based on particle filter and sensor data with non-gaussian and multimodal noise. IEEE Access 2021 9:60704–60712.
[31] Gui P, Tang L, Mukhopadhyay S. MEMS based IMU for tilting measurement: Comparison of complementary and kalman filter based data fusion. In 2015 IEEE 10th conference on Industrial Electronics and Applications (ICIEA), IEEE2015 pp. 2004–2009.
[32] Moshayedi AJ, Zanjani SM, Xu D, Chen X, Wang G, et al. Fusion BASED AGV robot navigation solution comparative analysis and VREP simulation. In 2022 8th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS), IEEE2022 pp. 1–11.
[33] Yuan C, Wang Y, Liu J. Research on multi-sensor fusion-based AGV positioning and navigation technology in storage environment. In Journal of Physics: Conference Series, IOP Publishing2022 p. 012052.
[34] Ullah I, Youn J, Han YH. Multisensor data fusion based on modified belief entropy in Dempster–Shafer theory for smart environment. IEEE Access 2021 9:37813–37822.
[35] Moshayedi, A. J., Reza, K. S., Khan, A. S., & Nawaz, A. (2023). Integrating virtual reality and robotic operation system (ROS) for AGV navigation. *EAI Endorsed Transactions on AI and Robotics*, *2*(1), e3-e3.
[36] Guyonneau, R., Mercier, F., & Boucher, V. (2024). Robotic system for indoor illuminance map generation. *Journal of Building Engineering*, *86*, 108800.
[37] Moshayedi, A. J., Li, J., Khan, A. S., Khan, Z. H., Khoojine, A. S., Hu, J., & Andani, M. E. (2026). Navigating the field: SLAM implementation for service robots in sports environment. In *Robotics and Artificial Intelligence in Sports Medicine and Sports Services* (pp. 241-275). Academic Press.

VI. APPENDIX

A. *Appendix A: List of Notations and Abbreviations*

Table 14: Table A1. List of Notations and Abbreviations

Abbr	Full Form	Abbr	Full Form
AGV	Automated Guided Vehicle	AI	Artificial Intelligence
AMCL	Adaptive Monte Carlo Localization	ANOVA	Analysis of Variance
AP	Average Precision	APH	Average Precision with Heading
API	Application Programming Interface	AVG	Average
BLE	Bluetooth Low Energy	CCD	Charge-Coupled Device
CNN	Convolutional Neural Network	CP	Continuous Path
CPR	Counts per Revolution	CRF	Conditional Random Field
CSI	Camera Serial Interface	CSPN	Cross Stage Partial Network
DC	Direct Current	DTW	Dynamic Time Warping
Dijkstra	Dijkstra's Shortest Path Algorithm	DWA	Dynamic Window Approach
EFM	Extended Fusion Method	EFM1	Extended Fusion Method 1
EFM2	Extended Fusion Method 2	FAST	Features from Accelerated Segment Test
FOV	Field of View	FPS	Frames Per Second
FUTR3D	Modality-agnostic Transformer-based Fusion Framework	GNSS	Global Navigation Satellite System

GPS	Global Positioning System	IMU	Inertial Measurement Unit
IR	Infrared	KF	Kalman Filter
LiDAR	Light Detection and Ranging	LIO	Lidar-Inertial Odometry
LIO-SAM	Lidar-Inertial Odometry via Smoothing and Mapping	LQR	Linear Quadratic Regulator
MPC	Model Predictive Control	MAE	Mean Absolute Error
NDS	NuScenes Detection Score	ORB-SLAM2	Oriented FAST and Rotated BRIEF SLAM v2
PCB	Printed Circuit Board	PID	Proportional–Integral–Derivative
PF	Particle Filter	PWM	Pulse Width Modulation
RFID	Radio-Frequency Identification	RGB	Red-Green-Blue
RMSE	Root Mean Square Error	ROS	Robot Operating System
RP	Raspberry Pi	SIF	Semantic Information Fusion
SIFT	Scale-Invariant Feature Transform	SLAM	Simultaneous Localization and Mapping
SPI	Serial Peripheral Interface	SURF	Speeded-Up Robust Features
UKF	Unscented Kalman Filter	UWB	Ultra-Wideband
V-REP	Virtual Robot Experimentation Platform	VILO	Visual–Inertial–Lidar Odometry Framework
VINS-Mono	Visual-Inertial Navigation System for Monocular Camera	VINS-Fusion	Visual-Inertial Navigation System with Multi-Sensor Fusion
YOLO	You Only Look Once	GR-LOAM	Ground-optimized Lidar Odometry and Mapping
A*	A-star Path Planning Algorithm		

B. Appendix B: Comparison of Sensor Types Used in AGVs

Table 15: Table B1. Sensor Types Used in AGVs

Sensor Type	Usage in AGVs	Advantages	Limitations	Demand Level
LiDAR	High-resolution 3D mapping and obstacle detection	Accurate distance measurement; 360° coverage	Expensive; sensitive to weather conditions	Very High — dominant in modern AGVs
Camera	Visual perception, object recognition, and path planning	Rich environmental data; supports AI-based analysis	Sensitive to lighting conditions; requires high computational resources	Increasing — growing with AI integration
Magnet	Following magnetic tapes or wires for navigation	Reliable on fixed paths; low cost; unaffected by lighting conditions	Limited to structured environments; inflexible	Moderate — niche in fixed-route AGVs
Ultrasonic Sensors	Short-range obstacle detection	Inexpensive; simple; effective at close range	Limited range and resolution; susceptible to interference	Moderate — often used as supplementary sensors
Infrared	Proximity detection and obstacle avoidance	Low cost; low power consumption	Affected by ambient light; limited sensing range	Moderate — used in simple systems
IMUs	Orientation and motion sensing	Supports dead reckoning and navigation stability	Subject to drift over time; requires periodic calibration	High — essential for localization

C. Appendix C: Comparison of Path Tracking Algorithms (EKF, PF and UKF for AGVs)

Table 16: Table C1. Path Tracking Algorithms for AGVs

Algorithm	Description	Advantages for Path Tracking
EKF	Nonlinear state estimation, widely used for localization	Accurate real-time pose estimation with high computational efficiency
PF	Handles complex noise and multimodal probability distributions	Robust performance in dynamic and uncertain environments
Complementary Filter	Combines IMU sensor data (accelerometer and gyroscope)	Provides smooth orientation estimation with low computational complexity
UKF	Improved nonlinear state estimation compared with EKF	Higher estimation accuracy for highly nonlinear motion
Neural Networks / Deep Learning	Learns complex patterns from sensor data for sensor fusion	Adapts effectively to complex, nonlinear path-tracking scenarios

D. Appendix D: Omini Platform Hardware Components and Specifications

Table 17: Table D1. Hardware Components and Specifications

Component	Type	Key Specifications

Processors	Raspberry Pi 3B+	Central processing unit for high-level data processing and computation.
	Arduino UNO/Mega	Low-level motion control and sensor data acquisition. The Arduino UNO controls four motor–encoder pairs, while the Arduino Mega reads data from the AGV Guider.
Chassis	Motor	12 V DC coreless motor; 17 W; 8100 RPM (no-load); 120 RPM output speed; 1400 mA load current; 64:1 gearbox.
	Wheel	Omni-directional wheel; 100 mm diameter; 18 rollers; nylon body; rubber rollers; 20 kg load capacity.
	Encoder	Incremental optical encoder; A/B phase; 12 CPR (counts per revolution).
Sensors	Camera	Raspberry Pi Camera Module; CSI-2 interface; maximum resolution of **3280 × 2464** pixels; up to 30 fps.
	RPLiDAR	Slamtec A1M8; sensing range of 0.15–6 m; 0°–360° field of view; sampling rate greater than 2000 Hz; scan rate of 5.5–10 Hz.
	AGV Guider	Magnetic stripe sensor; 16 detection units; operating voltage of 10–30 V; sensitivity of 0.5 mT; detection distance of 5–55 mm; response time of 1 ms.
Power	Battery	12 V batteries, separately supplied for the main control board and the chassis board.
	Regulator	12 V to 5 V voltage regulator module for the Raspberry Pi and Arduino boards.
	Fuse	2 A fuse for over-current protection.