Intelligent Service Robotics (2026) 19:33
https://doi.org/10.1007/s11370-025-00661-7

ORIGINAL RESEARCH PAPER

Development of an integrated estimation and predictive control framework for safe navigation in mobile robots for industrial environments

Gilchrist R. S. Gnimady[1] · Evan Murimi[2] · Anthony K. Muchiri[2] · Samuel Kangwagye[3,4]

Received: 19 May 2025 / Accepted: 10 November 2025

Abstract
The increasing integration of mobile robots in industrial environments has raised critical safety concerns, particularly in shared workspaces with human operators. Effective collision avoidance is essential to prevent accidents and enable smooth navigation in dynamic and unpredictable settings. This paper presents the development of an integrated estimation and predictive control framework for safe navigation in mobile robots for industrial environments. Here, safe navigation refers to improved, accurate motion of the robot (i.e., trajectory tracking, positioning, speed control) and enhanced collision avoidance (i.e., safe navigation around obstacles and humans). The estimation algorithm integrates data from LiDAR, RADAR, and IMU sensors using an Extended Kalman Filter (EKF) to overcome limitations such as LiDAR blind spots and IMU drift. The output of this algorithm is used as input to the MPC, which embeds an obstacle avoidance function directly into its cost function to enable proactive and adaptive collision avoidance. The complete framework is validated through both Gazebo-based simulations and real-world experiments in free-space navigation and navigation through obstacle-rich industrial environments. The results demonstrate that the robot can track reference trajectories while safely avoiding static and dynamic obstacles, including those located within sensor blind zones. The robot consistently maintains safe distances without unnecessary stops, achieving a balance between safety and operational efficiency. The framework offers a cost-effective alternative to high-resolution 360-degree perception systems and is well-suited for deployment in dynamic industrial workspaces.

Keywords Blind spots · Collision avoidance · Mobile robot control

1 Introduction

Recent advancements in robotics have enabled systems to interact more intelligently and dynamically with their environments. When these systems are deployed in industrial settings, however, safety becomes a major concern, especially when human workers and autonomous robots operate in the same workspace [1]. The presence of mobile robots increases the risk of accidents, as collisions can cause serious injuries, slow down production, and interrupt operations. Ensuring safe navigation is therefore essential for any robot deployment strategy. The challenge is amplified by the unpredictable nature of human movement, which requires robots to react in real time while still performing their tasks smoothly [2, 3].

Collision avoidance has long been a central focus in mobile robotics, and numerous methods have been proposed to enhance navigation safety and efficiency. Early approaches relied primarily on LiDAR-based sensing. For instance, Max and Asiya [4] employed the Hokuyo URG-04LX LiDAR for obstacle detection, but blind spots remained a significant limitation. Ueyama et al. [5] developed an affordable LiDAR-based robot, yet the lack of predictive control made it unsuitable for dynamic environments. To improve navi-

✉ Samuel Kangwagye
samkan@mp.aau.dk

1 Department of Mechatronic Engineering, Pan African University Institute for Basic Sciences, Technology and Innovation (PAUSTI) Hosted at Jomo Kenyatta University of Agriculture and Technology (JKUAT), 62000 – 00200 Nairobi, Kenya

2 Department of Mechatronic Engineering, Jomo Kenyatta University of Agriculture and Technology (JKUAT), 62000 – 00200 Nairobi, Kenya

3 Robotics and Automation Group, Department of Materials and Production, Aalborg University, 9220 Aalborg, Denmark

4 Department of Mechanical and Production Engineering, Kyambogo University, 1 Kyambogo, Kampala, Uganda

Published online: 09 February 2026

gation performance, many researchers have adopted Model Predictive Control (MPC). Sudianto et al. [6] demonstrated path planning with MPC in structured environments, while Sani et al. [7] applied nonlinear MPC for smooth navigation in cluttered spaces. However, these methods typically depend on a single LiDAR sensor, limiting their effectiveness in blind zones. Even recent NMPC work with LiDAR data [8] improved obstacle awareness but continued to suffer from occlusions. Control barrier function (CBF)-based formulations have also been proposed to guarantee safety [9], yet they often require expensive 360° LiDAR sensors [10] and increase computational demand, restricting feasibility on embedded platforms. Similar limitations are observed in multi-robot NMPC studies, which frequently rely on idealized simulations that overlook real-world unpredictability [11].

To address these perception gaps, researchers have explored multisensor fusion strategies. Combining heterogeneous data sources such as LiDAR, RADAR, and IMU can provide a more complete and reliable representation of the environment, compensating for blind spots and drift [12]. For example, Elleithy et al. [13] demonstrated real-time obstacle avoidance using ultrasonic and infrared sensors, while Liang et al. [14] combined 2D LiDAR with depth cameras to navigate dense crowds. In addition, other recent approaches have employed dense point-cloud representations within NMPC for multi-robot flocking [15]. While effective, these methods highlight the significant computational burden of processing large amounts of perception data. Despite the promise of both sensor fusion and point-cloud approaches, their integration into lightweight, real-time NMPC frameworks are not fully exploited, particularly in dynamic human-robot shared spaces where proactive, motion-aware avoidance is critical.

Although there has been significant progress in integrating perception and control for autonomous navigation, current MPC-based frameworks still face three key challenges: reliance on single-sensor inputs and associated blind spots, high computational complexity in safety-certified controllers, and limited coupling of multisensor fusion with predictive control under embedded hardware constraints. At the same time, industrial applications demand low-cost, reliable solutions that can operate in real time without depending on expensive high-end sensors or desktop-class computing resources.

Therefore, the main contribution of this paper is the integration of a multisensor information fusion and state estimation framework with a nonlinear MPC controller to achieve robust and safe navigation in dynamic industrial environments. The sensor fusion and estimation framework combine LiDAR and RADAR measurements to overcome blind spots, while IMU data are incorporated through an EKF to improve localization accuracy. The resulting enhanced state estimates are provided as inputs to the NMPC, whose cost function integrates trajectory tracking, control accuracy, and a collision avoidance term. This direct embedding of avoidance in the optimization objective enables the robot to adapt proactively to nearby obstacles and human co-workers, rather than reacting only after hazards are detected. Unlike existing NMPC approaches that rely solely on LiDAR or computationally expensive point-cloud processing, the proposed framework uniquely embeds obstacle distance directly into the cost function while extending perception into LiDAR blind spots using RADAR and IMU through EKF. This combination allows proactive, motion-aware obstacle avoidance with real-time feasibility on low-cost embedded hardware. The effectiveness of the proposed framework is validated through Gazebo-based simulations and real-world experiments, to demonstrate its potential for safe and efficient operation in collaborative settings between humans and mobile robots.

2 Problem setup and preliminaries

2.1 Mobile robot and obstacles on a factory floor

Figure 1 illustrates a typical factory floor scenario where a mobile robot, equipped with sensors: a Hokuyo LiDAR, A121 RADAR, and an IMU, operates in the same space with objects such as human workers. Each sensor plays a critical role in enhancing the robot's perception and navigation capabilities. The Hokuyo LiDAR is selected for its cost-effectiveness, high-precision scanning capabilities, and reliability in industrial environments [16]. It efficiently detects both static and dynamic obstacles within its 270-degree FoV. However, this LiDAR has a 30-degree blind spot, particularly behind the robot, which can result in undetected obstacles. To mitigate this limitation, the A121 RADAR sensor is added to the robot. Unlike LiDAR, RADAR is less affected by environmental conditions such as dust, smoke, or lighting variations, making it well-suited for detecting and tracking moving obstacles, including human workers. The IMU sensor provides linear acceleration, angular velocity, and orientation data, which is essential for accurate robot localization. However, IMU measurements are often affected by drift and noise, leading to gradual inaccuracies over time [17]. To address all these limitations of the three sensors, an information processing algorithm is proposed in Section 3.1.

2.2 Mobile robot modeling

The mobile robot in Fig. 1 is further presented in Fig. 2 showing its (x, y) motion model. The robot consists of two independently driven caster wheels, which determine its motion. The velocities of the left and right wheels, denoted as

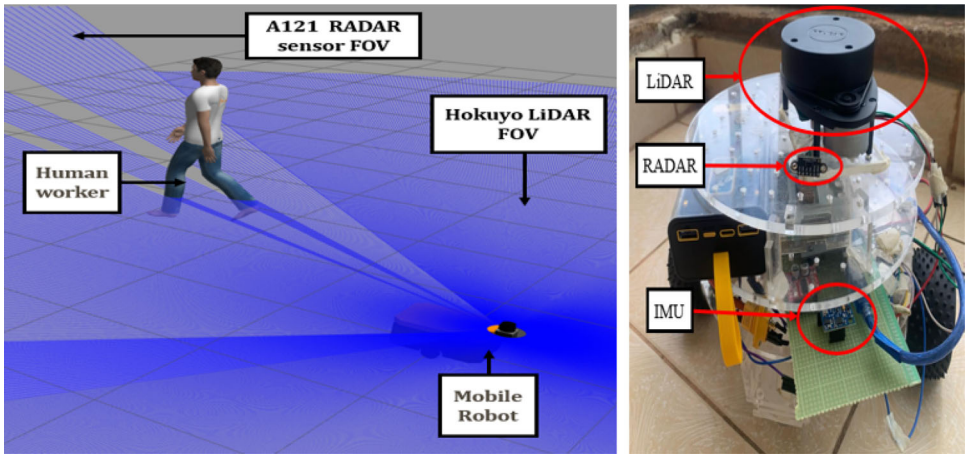

Fig. 1 Mobile robot, in a simulated environment (left) and actual robot (right), equipped with a Hokuyo LiDAR, A121 RADAR, and IMU sensors

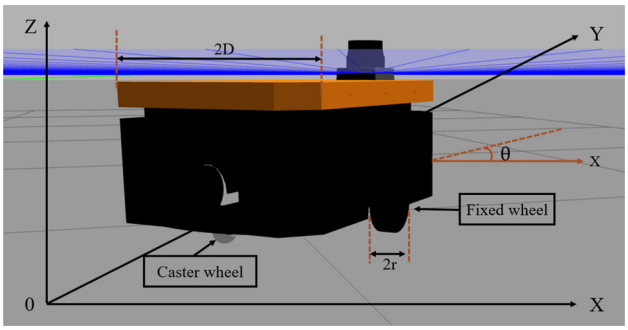

Fig. 2 Nonholonomic wheeled mobile robot

v_l and v_r, respectively, contribute to the overall linear velocity $\hat{v}$ and angular velocity $\hat{\omega}$ of the robot represented as.

$$\hat{v} = (v_l + v_r)/2 \quad \text{and} \quad \hat{\omega} = (v_l - v_r)/D, \tag{1}$$

where D represents half the distance between the wheels [18]. The robot's position in the global coordinate frame, (x, y), corresponds to the center of the driving axis. If θ is the robot's orientation relative to the world frame, the kinematic equations of the robot can be derived as

$$\dot{\hat{x}} = \hat{v}\cos(\theta), \quad \dot{\hat{y}} = \hat{v}\sin(\theta), \quad \text{and} \quad \dot{\theta} = \hat{\omega}. \tag{2}$$

From (2), the state variable vector and control vector are given by

$$\begin{pmatrix} \hat{x} & \hat{y} & \hat{\theta} \end{pmatrix}^T \quad \text{and} \quad u = (v \quad \omega)^T \tag{3}$$

The discrete time equivalent of (2) is obtained by using a simple Euler method for discretization for a time step Δt as

$$x_{k+1} = \hat{x}_k + v_k \cos(\theta_k)\Delta t, \tag{4}$$
$$y_{k+1} = \hat{y}_k + v_k \sin(\theta_k)\Delta t, \tag{5}$$
$$\theta_{k+1} = \hat{\theta}_k + \omega_k \Delta t. \tag{6}$$

The generalized nonlinear system and measurement equations of the mobile robot can be represented in discrete time as

$$X_{k+1} = f(X_k, U_k) \quad \text{and} \quad Z_k = h(X_k), \tag{7}$$

where X_k and U_k are state and input vectors, respectively.

3 Proposed object-aware navigation framework

The framework, illustrated in Fig. 3, is designed to enable safe and efficient navigation for a mobile robot operating in dynamic industrial environments with static and moving obstacles. From Fig. 3, the sensors' measurements are used by the EKF algorithm to provide accurate estimates. These are in turn used as input to the nonlinear MPC. To this end, the estimation algorithm provides accurate robot position and speed, as well as range information, which is utilized in the MPC cost function to enable collision avoidance capability. In addition, the MPC ensures trajectory following.

3.1 EKF-based multisensor information processing algorithm

The sensor information processing algorithm employed in this work is based on the Extended Kalman Filter (EKF) [19–21] as shown in Fig. 3. The system and measurement equations in (7) are re-written to include the noise terms as shown below

$$X_{k+1} = f(X_k, U_k, W_k), \quad W_k \sim \mathcal{N}(0, Q_k), \tag{8}$$
$$Z_k = h(X_k, V_k), \quad V_k \sim \mathcal{N}(0, R_k), \tag{9}$$

where X_k, Z_k, and U_k are the state vector, the measurable output, and the process input, respectively. W_k and V_k are

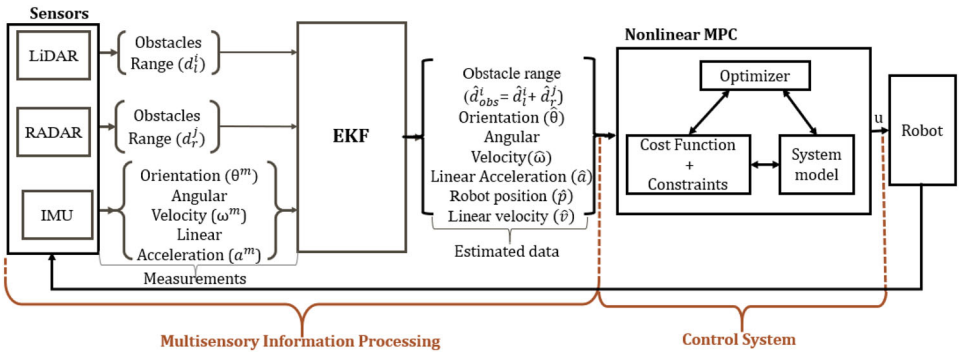

Fig. 3 The proposed object-aware navigation framework

process and white measurement noises, which are assumed to be normally distributed with zero mean and standard deviations. Q_k and R_k are process and measurement covariance matrices, respectively.

3.1.1 State vector formulation

The state vector, which is divided into two parts, is designed as

$$\mathbf{X}_k = \left[\overbrace{p_k \;\; v_k \;\; a_k \;\; q_k \;\; \omega_k}^{\text{robot states}} \;\Big|\; \overbrace{d_{l,k}^i \;\; d_{r,k}^j}^{\text{object ranges}} \right]^T, \tag{10}$$

where the states $p_k = [p_x, p_y]^T$, $v_k = [v_x, v_y]^T$, $a_k = [a_x, a_y]^T$, $q_k = [q_x, q_y, q_w]^T$, and $\omega_k = [\omega_x, \omega_y]^T$, $d_{l,k}^j$ and $d_{r,k}^j$ represent linear position, velocity, acceleration, quarternion orientation, angular velocity, and ranges of the object. The state vector in (10) is divided into two parts. The part of states for robot control, which contains states of the robot and that which contains states for range of the object for collision avoidance.

3.1.2 Estimates and measurements

The system consists of seven states as shown in (10). The sensors attached on the robot can provide five measurements, i.e., measurements for a_k, q_k, and ω_k from IMU sensor, $d_{l,k}^j$ from LiDAR, and $d_{r,k}^j$ from the RADAR, with the RADAR positioned to cover areas within the LiDAR's blind spot. i and j represent the number of detected objects by the LiDAR and the RADAR sensors. On the other hand, position p_k and velocity v_k measurements are estimated using the acceleration measurements through the EKF.

3.1.3 The nonlinear process model

The robot's motion and sensor dynamics include nonlinear relationships, especially when considering orientation updates using quaternions and the integration of acceleration to update velocity and position. To this end, the full process model update of the system model is given by

$$X_k = \begin{pmatrix} p_{k-1} + v_k \Delta t \\ v_{k-1} + a_k \Delta t \\ a_{k-1} \\ \mathbf{q}_{k-1} + \frac{1}{2} \begin{pmatrix} -q_x \omega_x - q_y \omega_y \\ q_\omega \omega_x \\ q_\omega \omega_y \\ -q_x \omega_y + q_y \omega_x \end{pmatrix} \Delta t \\ \omega_{k-1} \\ d_{l,k-1}^i \\ d_{r,k-1}^j \end{pmatrix}, \tag{11}$$

which consists of the updates for position, velocity, angular velocity, linear acceleration, orientation, and range. The process model in (11) contains nonlinear components, i.e., the quaternion update for orientation. The linearization of these nonlinear components is achieved using the Jacobian matrix in the process model.

Linear terms: From (11), the position and velocity update equation shows that they are linearly dependent on the velocity and the acceleration by Δt. Thus, the partial derivative of position compared to velocity and the velocity compared to the acceleration is also Δt times the identity matrix as shown below.

$$\begin{aligned} &\frac{\partial \mathbf{p}_k}{\partial \mathbf{v}_k} = \Delta t \mathbf{I}, \quad \frac{\partial \mathbf{v}_k}{\partial \mathbf{a}_k} = \Delta t \mathbf{I}, \quad \frac{\partial \mathbf{a}_k}{\partial \mathbf{a}_{k-1}} = \mathbf{I}, \\ &\frac{\partial \omega_k}{\partial \omega_{k-1}} = \mathbf{I}, \quad \frac{\partial \mathbf{d}_{\mathbf{l}k}}{\partial \mathbf{d}_{\mathbf{l}k-1}} = \mathbf{I}, \quad \frac{\partial \mathbf{d}_{\mathbf{r}k}}{\partial \mathbf{d}_{\mathbf{r}k-1}} = \mathbf{I}. \end{aligned} \tag{12}$$

Nonlinear terms: The quaternion update in (11) is nonlinear, necessitating linearization. The partial derivative of orientation update with respect to itself is given by

$$\frac{\partial q_k}{\partial q_{k-1}} = I + \frac{1}{2} \frac{\partial (q_k \otimes \omega_k)}{\partial q_k} \Delta t. \tag{13}$$

Applying the standard definition for quaternion multiplication to (13) gives

$$\frac{\partial \mathbf{q}_k}{\partial \mathbf{q}_{k-1}} = \begin{pmatrix} 1 & -\frac{1}{2}\omega_x \Delta t & -\frac{1}{2}\omega_y \Delta t & 0 \\ \frac{1}{2}\omega_x \Delta t & 1 & 0 & \frac{1}{2}\omega_y \Delta t \\ \frac{1}{2}\omega_y \Delta t & 0 & 1 & -\frac{1}{2}\omega_x \Delta t \\ 0 & -\frac{1}{2}\omega_y \Delta t & \frac{1}{2}\omega_x \Delta t & 1 \end{pmatrix}. \quad (14)$$

Full Jacobian Matrix F_k: The linearization of $f(X_k)$ in (8) to get F_k is based on the partial derivation of (11) as given below

$$F_k = \begin{bmatrix} Ip & \Delta t Iv & 0 & 0 & 0 & 0 & 0 \\ 0 & Iv & \Delta t Ia & 0 & 0 & 0 & 0 \\ 0 & 0 & Ia & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\partial \mathbf{q}_k}{\partial \mathbf{q}_{k-1}} & \frac{\partial \mathbf{q}_k}{\partial \boldsymbol{\omega}_k} & 0 & 0 \\ 0 & 0 & 0 & 0 & I\omega & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & I_l^i & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & I_r^j \end{bmatrix}, \quad (15)$$

where $\frac{\partial(q_k)}{\partial \omega_k}$ represents the Jacobian of the quaternion respective to angular velocity, which shows how the angular velocity affects the quaternion.
The partial derivative of orientation update with respect to the angular velocity is given by (16).

$$\frac{\partial q_k}{\partial \omega_k} = \frac{1}{2}\frac{\partial (q_k \otimes \omega_k)}{\partial \omega_k}\Delta t. \quad (16)$$

Applying the standard definition for quaternion multiplication to (16) gives

$$\frac{\partial \mathbf{q}_k}{\partial \boldsymbol{\omega}_k} = \frac{1}{2}\Delta t \begin{pmatrix} 0 & -q_x & -q_y & -q_z \\ q_w & 0 & -q_y & q_z \\ q_w & q_x & 0 & -q_z \\ q_w & -q_x & q_y & 0 \end{pmatrix}. \quad (17)$$

3.1.4 Linearization of the measurement model

Since the measurement function $h(X_k)$ may also contain nonlinear relationships between the state and the sensor data, a Jacobian matrix H_k is computed, which combines the Jacobians for IMU, LiDAR, and RADAR measurements is given by

$$H_k = [I_{IMU} \;\; I^i_{LiDAR} \;\; I^j_{Radar}]^T. \quad (18)$$

The range measurements and the IMU measurements are directly part of the state vector, thus the Jacobian matrix for the LiDAR, RADAR, and IMU measurements is straightforward and can be written as identity matrices.

3.1.5 Design of system and measurement covariance matrices

The process noise covariance matrix Q_k accounts for the inherent randomness in the robot dynamics modeling and how the actual state might deviate from the predicted state due to unknown factors. This is given by

$$Q_k = \text{diag}[\sigma_p^2 \;\; \sigma_v^2 \;\; \sigma_a^2 \;\; \sigma_q^2 \;\; \sigma_\omega^2 \;\; (\sigma_{dl}^i)^2 \;\; (\sigma_{dr}^j)^2], \quad (19)$$

where σ_p, σ_v, σ_a, σ_ω, and σ_q represent the process noise standard deviations for the position, velocity, acceleration, orientation, angular velocity and ranges, respectively.

The measurement noise covariance matrix R_k captures the uncertainty in the LiDAR, RADAR, and IMU sensor measurements. It accounts for the noise inherent in the measurement process, including sensor drifts, inaccuracies, and environmental noise.

$$R_k = \text{diag}[\sigma_p'^2 \;\; \sigma_v'^2 \;\; \sigma_a'^2 \;\; \sigma_q'^2 \;\; \sigma_\omega'^2 \;\; (\sigma_{dl}'^i)^2 \;\; (\sigma_{dr}'^j)^2], \quad (20)$$

where σ' represents the measurement noise for each sensor.

3.1.6 Prediction and update steps of the EKF algorithm

Having designed the EKF components in the previous subsections, the prediction and update steps for the EKF are computed as follows. The predicted state, $\hat{\mathbf{x}}_{k|k-1}$, and error covariance, $\mathbf{P}_{k|k-1}$, are given by

$$\begin{aligned} &\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}) \;\text{ and }\; \mathbf{P}_{k|k-1} \\ &= \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}_{k-1}^\top + \mathbf{Q}_k. \end{aligned} \quad (21)$$

For the update step, the equations for innovation, $\mathbf{y}_k$, innovation covariance, $\mathbf{S}_k$, Kalman gain, $\mathbf{K}_k$, updated state estimate, $\hat{\mathbf{x}}_{k|k}$, and updated error covariance, $\mathbf{P}_{k|k}$ are, respectively, given by

$$\begin{aligned} &\mathbf{y}_k = \mathbf{z}_k - h(\hat{\mathbf{x}}_{k|k-1}), \quad \mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^\top + \mathbf{R}_k, \\ &\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^\top\mathbf{S}_k^{-1}, \\ &\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\mathbf{y}_k, \quad \text{and} \quad \mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1}. \end{aligned} \quad (22)$$

3.2 Design of the collision avoidance embedded nonlinear MPC

3.2.1 Objective function with collision avoidance component

The objective function which incorporates the collision avoidance component and cost function are derived as

$$\min_{u} \sum_{k=0}^{N-1} \left[||x_k - x_k^r||_A^2 + ||u_k - u_k^r||_B^2 + \lambda \sum_{k=0}^{n_{obs}} \phi(\hat{d}_{obs,k}^i) \right], \quad \text{s.t.} \begin{cases} d_{safe} - \hat{d}_{obs,k}^i \leq 0, \forall i \in n_{obs}, \\ p_{x\min} \leq \hat{p}_{x,k} \leq p_{x\max}, \\ p_{y\min} \leq \hat{p}_{y,k} \leq p_{y\max}, \end{cases} \tag{23}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{B} \in \mathbb{R}^{m \times m}$ are positive definite symmetric weighting matrices for trajectory tracking and control smoothness, respectively. d_{safe} is a predefined safety distance threshold. $x_{\min}$, $x_{\max}$, $y_{\min}$, $y_{\max}$ represent the boundaries of the robot's operational workspace, defining the minimum and maximum allowable positions along the x-axis and y-axis, respectively. The objective function consists of the three terms. Two conventional terms are path tracking, $||x_k - x_k^r||_A^2$, and control effort, $||u_k - u_k^r||_B^2$.

In this paper, a third term of the objective function in (23) is proposed as $\lambda \sum_{k=0}^{n_{obs}} \phi(\hat{d}_{obs,k}^i)$, which is a repulsive potential cost that increases sharply as the robot approaches an obstacle and remains small when far away to penalize situations where the robot gets too close to an obstacle. Here, λ is introduced as a tuning parameter to balance trajectory tracking and obstacle avoidance, $\phi(\hat{d}_{obs,k}^i)$ is the avoidance function for the i-th obstacle, n_{obs} is the number of nearest obstacles considered at step k, and $\hat{d}_{obs,k}$ is the estimated distance between the robot and the nearest obstacles at time step k obtained from the multisensor-based ranges information.To this end, the avoidance function is designed as

$$\phi(\hat{d}_{obs,k}^i) = \left(\frac{1}{\hat{d}_{obs,k}^i + 0.1} \right)^2 + 10e^{-\hat{d}_{obs,k}^i}, \tag{24}$$

$$\text{where } \hat{d}_{obs,k}^i = \hat{d}_{l,k}^i + \hat{d}_{r,k}^j. \tag{25}$$

The objective of this novel function is not to minimize the obstacle distance directly but rather to penalize situations where the robot gets too close to an obstacle. This is achieved by introducing a smooth penalty function that remains inactive when the robot is at a safe distance but increases gradually as the robot approaches an obstacle as shown in (24). Specifically, if the obstacle distance $\hat{d}_{obs,k}^i$ is large, the cost contribution remains very small (close to zero), allowing the optimizer to focus primarily on trajectory tracking. However, as the robot moves closer to an obstacle, the penalty function gradually increases, discouraging unsafe trajectories. The penalty grows more significantly as the obstacle gets closer, ensuring that the robot proactively adjusts its path to maintain a safe separation while still following its intended trajectory as smoothly as possible.

To prevent extreme deviations from the desired path or unsafe behavior, a constraint is imposed to ensure that the robot maintains a minimum allowable distance d_{safe} from obstacles at all times. To ensure the robot operates within a predefined workspace, constraints on the robot's position $(\hat{p}_x, \hat{p}_y)$ are imposed to prevent the robot from leaving its designated area, which is essential in structured industrial environments where boundaries must be respected.

The primary decision variables in this formulation are the linear velocity v_k and angular velocity ω_k, which dictate the robot's motion. These variables are dynamically optimized within the predefined limits to ensure smooth navigation and obstacle avoidance. To prevent unsafe or impractical control actions, these variables are bounded within the following ranges as

$$v_k \in [v_{min}, v_{max}] \quad \text{and} \quad \omega_k \in [\omega_{min}, \omega_{max}] \tag{26}$$

where v_{min} and v_{max} define the allowable range for linear velocity, and ω_{min}, ω_{max} set the limits for angular velocity.

4 Validation of the proposed method

4.1 Experiment and simulation protocols

4.1.1 Setups and parameters

The simulation and experimental setups are illustrated in Fig. 1. Both setups were implemented in ROS2, using a differential-drive robot model with a wheelbase of 0.30 m and a wheel radius of 0.05 m. The NMPC was implemented in CasADi with the Ipopt solver and executed at 10 Hz.

For simulation [see Fig. 1(left)], a low-cost A121 RADAR sensor with a FoV of 65° was installed on the robot in addition to the LiDAR and IMU sensors. A moving obstacle was placed within the robot's FoV: it was first detected by the Hokuyo LiDAR, and once it entered the LiDAR's blind spot, the A121 RADAR continued tracking it. Obstacles were represented as $0.5 \times 0.5 \times 0.5$ m cubes, both static and dynamic, distributed along the robot's path.

For the experiments [see Fig. 1 (right)], the framework was executed on a Raspberry Pi 4 (Quad-core Cortex-A72 @ 1.5 GHz, 4 GB RAM). The mobile robot was equipped with a 360° RPLiDAR, an Acconeer A121 RADAR, and a BNO055 IMU sensor. The RPLiDAR provided full-range

scanning, but to emulate the 270° FoV typical of Hokuyo LiDARs used in the proposed algorithm, a custom filtering algorithm was applied to ignore the front-left sector (from $angle\_min = 0$) by setting those ranges to infinity. To compensate for this blind spot, the RADAR sensor was mounted precisely in the occluded region of the LiDAR, ensuring full coverage and reliable obstacle detection. In practice, experimental obstacles consisted of a water bottle and a human leg, which correspond to objects approximately 0.15–0.20 m in diameter. An IMU mounted on the lower platform provided orientation and motion data.

Parameters for (19) and (20) are selected as $\sigma_a' = 0.0019g$, $\sigma_\omega' = 0.14°/s$, and $\sigma_q' = 6\,\mu T$ for the accelerometer, gyroscope, and magnetometer for the BNO055 IMU sensor at a bandwidth of 100 Hz. $\sigma_d'^i = 0.030\,\text{m}$ is for the Hokuyo UST-10LX LiDAR sensor, while $\sigma_d'^j = 0.00505\,\text{m}$ is for the A121 Pulsed Coherent RADAR sensor for multiple profiles at a typical room temperature (25°C). These values were obtained from manufacturer datasheets and then fine-tuned experimentally to reduce estimation error while maintaining filter stability. Assuming negligible process noise, $\sigma = 0.001$ was set for the matrix Q_k. For the MPC, $A = \text{diag}([1, 1, 0.1])$, where higher weights on position coordinates prioritize accurate trajectory tracking, while a lower weight on orientation avoids overly aggressive angular corrections. $B = \text{diag}([0.1, 0.5])$ balances the control effort between linear and angular velocities, with larger values in B leading to smoother but less responsive inputs, and smaller values improving responsiveness at the cost of higher control activity. The robot's linear velocity v is bounded between $[-0.15\,\text{m/s}, 0.15\,\text{m/s}]$ and angular velocity ω is bounded between $[-0.8\,\text{rad/s}, 0.8\,\text{rad/s}]$, which ensures safe and physically feasible maneuvers for the robot. A safety distance d_{safe} tuned to 0.5 m was enforced to guarantee sufficient clearance from obstacles in both simulation and experiment. The sampling time was set to $\Delta t = 0.1$ s. The obstacle avoidance penalty weight λ was tuned to 60, as smaller values of < 30 resulted in the robot approaching obstacles too closely, which is unsafe, while larger values of $>$ 80 caused excessively conservative detours and increased trajectory deviation. The chosen value of $\lambda = 60$ provided a good compromise, consistently maintaining at least 0.5 m clearance while keeping the trajectory root-mean-square error below 0.4 m. Finally, the robot position constraints were set to $0 \leq x_k \leq 4$ to emulate the experimental floor space. These parameter choices represent a practical balance between trajectory accuracy, obstacle avoidance safety, and control effort.

4.2 Ground truths

To validate the EKF-based algorithm, a ground truth based on the Euclidean distance **d** between the robot (p_r) and the obstacles (p_w) for the Gazebo Model State was computed. This computation is performed at each simulation time step to continuously validate the proposed method's tracking accuracy in the LiDAR blind spot. The ground truth based on Gazebo position and velocity model state was also used to validate the proposed method of tracking the robot's position and velocity accuracy derived by integrating the estimated acceleration.

For the control system, the robot trajectory is derived from the Gazebo simulation model, which continuously updates the robot's state based on odometry feedback. Since odometry readings provide real-time estimates of the robot's position, the robot trajectory can be considered as ground truth, representing the actual movement executed in response to control commands. The MPC trajectory is generated by integrating the robot's unicycle kinematic model, which transforms control inputs (linear velocity v and angular velocity ω) into a sequence of predicted positions (x, y) over a finite horizon N as developed in (4).

Ground truth for experiments: For independent pose validation, floor markers were laid out and their coordinates recorded with a tape measure. At each marker, the estimated position from the EKF was logged and compared to the known ground truth. The markers checked NMPC tracking: trials always started at the origin mark (0, 0, 0) and ended at the goal mark (4, 0, 0), where the tape-measured final pose was contrasted with the controller's target.

4.3 Estimation performance evaluation of the multisensor-based EKF information processing algorithm

Simulation scenario 1: The worker is initially placed within the LiDAR's FoV. As the robot follows a forward trajectory around the stationary worker, it simulates typical operational movements. Over time, as the robot continues its path, the worker transitions from being within the LiDAR's FoV to a blind spot. Results are presented in Fig. 4.

The gaps in LiDAR coverage depicted by the dotted red line, indicated by the discrepancies between the LiDAR scan range and the ground truth, show the blind spot of this LiDAR sensor from 220 s to 475 s (the area where the person was not detected by the LiDAR sensor while the ground truth is continuously plotting the trajectory). On the other hand,

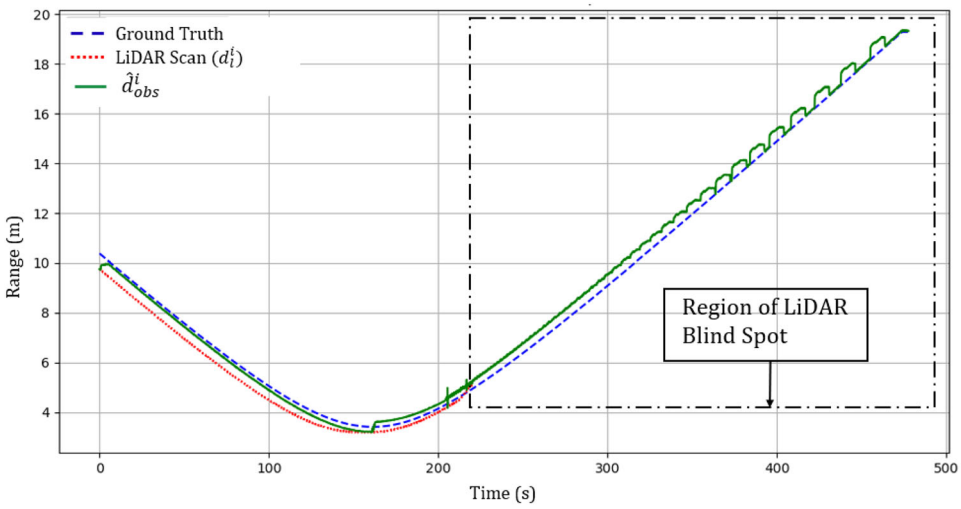

Fig. 4 Comparison of the range estimated by the EKF with the ground truth and the LiDAR raw data

the proposed algorithm can continuously track the human (green line). Moreover, there is alignment between the estimated range and the ground truth. From the 0 s to 220 s area where the LiDAR sensor (dotted red line) is detecting, the trend of the estimated range is closer to the ground truth than the LiDAR scan. This shows the precision of the proposed EKF algorithm, affirming its effectiveness in using data from multiple sensors to provide an accurate estimation of the object's position and also mitigate the LiDAR blind spot. The mean error for the LiDAR measurements is 0.435 with a standard deviation of 0.177, while the estimated range shows a reduced mean error of 0.25 and standard deviation of 0.11. This indicates that the multisensor framework provides a more accurate representation of the measured values.

Simulation scenario 2: In this scenario, the robot navigates a forward trajectory at varying speeds. The reference point for the proposed method aligns with the one used in the Gazebo simulator, ensuring consistency in tracking and evaluation metrics. This simulation focuses exclusively on measuring the accuracy and reliability of the robot's position and velocity estimations as speed varies.

The position error comparison and tracking performance plots in Fig. 5 reveal significant insights into the effectiveness of the proposed estimation method. The position tracking in Fig. 5 (left) illustrates that the multisensor method position trend closely aligns with the ground truth, maintaining consistent accuracy, while the direct integration of the IMU data diverges significantly. The IMU raw direct integration trend shows how noisy and unreliable the output of this raw data can be. The position error graph in Fig. 5 (right) shows that the error for the proposed multisensor method position remains considerably lower (close to 0 m) and more stable over time than the direct integration from the IMU measurement, which experiences a sharp increase in error over time with a large margin of error from 25 s. The IMU position measurements show a high mean error of 6.4, and a standard deviation of 5.13, while the multisensor method significantly reduces the mean error to 0.367 and the standard deviation to 0.434. This contrast indicates that the estimation algorithm significantly improves the accuracy of position estimates.

The velocity tracking plot in Fig. 6 (left) shows that the proposed multisensor method velocity closely matches the ground truth throughout the timeline. The velocity obtained by direct integration from the raw IMU measurement trend is relatively lower than the ground truth over time from 25 s. The velocity error comparison in Fig. 6 (right) indicates that the estimated velocity consistently maintains lower error rates than the direct integration from the IMU measurements, exhibiting significant spikes and variability. The IMU velocity measurements show a high mean error of 0.07, and a standard deviation of 0.061, while the estimated velocity significantly reduces a bit the mean error to 0.038125 and the standard deviation to 0.0373. However, discrepancies are observed in Fig. 6 (right) during rapid velocity changes, and at higher operational speed scenarios, some spikes in the proposed method highlight potential areas for improvement. These discrepancies are important as they highlight the limitations of the current data integration method under certain dynamic conditions.

Experiment scenario: The robot was driven manually from its start pose to the goal (4, 0, 0) while the multisensor algorithm was running. Several deliberate stops were inserted along the path; the resulting flat segments in the estimated trajectory allowed to verify that the position output remained stable (i.e., no drift) whenever the robot was stationary.

Figure 7 compares, over the entire 260 s experiment, the minimum range reported at every time step by three sources: the ground-truth 360°LiDAR (blue), the Hokuyo LiDAR (green), and the proposed multisensor estimate (orange, dashed). Whenever the closest obstacle falls inside the Hokuyo's 90° blind zone, the Hokuyo LiDAR cannot detect it; its algorithm therefore returns the minimum range mea-

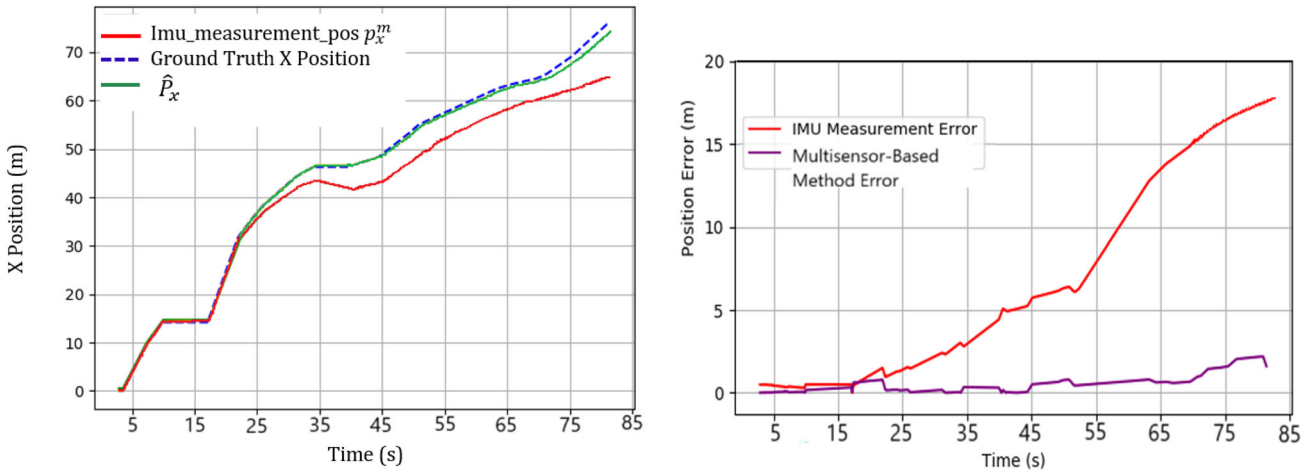

Fig. 5 Left: Comparison of the estimated position and the ground truth, alongside the direct position integration from IMU measurements. Right: absolute position errors of both methods relative to the ground truth

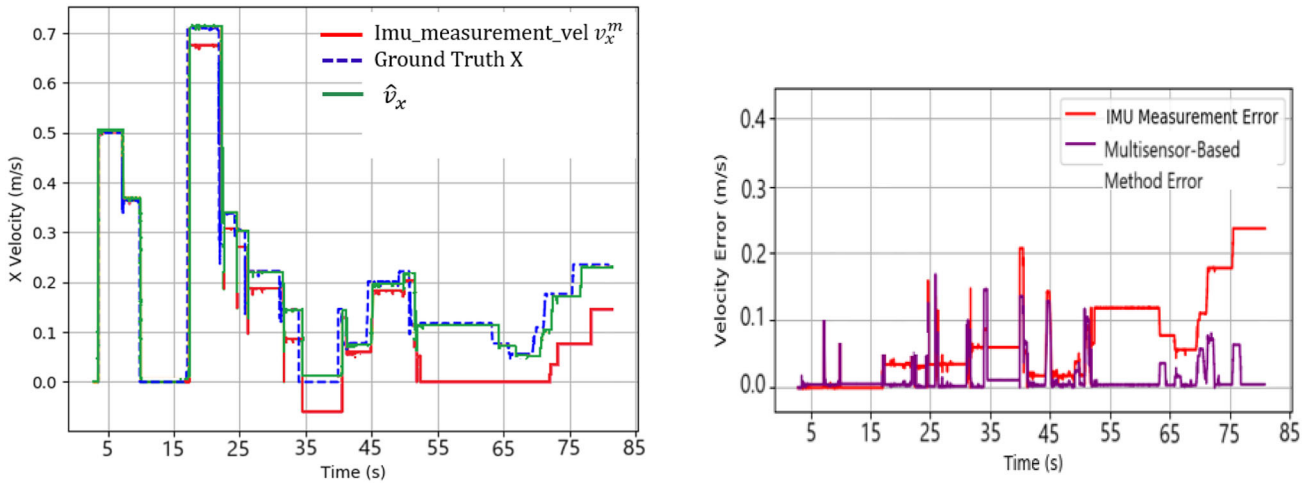

Fig. 6 Left: Comparison of the estimated velocity, the ground truth, and direct velocity integration from IMU measurements. Right: Velocity errors of both methods relative to the ground truth

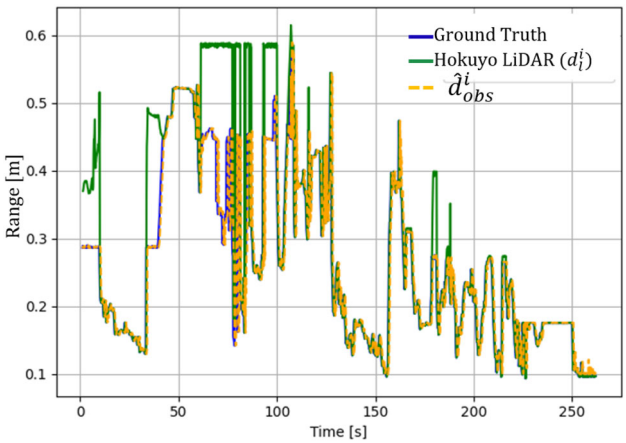

Fig. 7 Comparison of the estimated range with the ground truth and the Hokuyo LiDAR raw data

sured elsewhere in the scan and the green trace abruptly rises. These excursions, clearly visible around 55 s, 95 s, and 130 s do not appear in the ground-truth signal. In contrast, the multisensor curve remains tightly aligned with the ground truth throughout because the A121 RADAR continues to observe the obstacle inside the blind spot and the EKF seamlessly propagates this information into the fused range estimate. This behavior proves the benefit of adding a RADAR sensor to cover Hokuyo LiDAR occlusions.

A further aspect of the estimation results is the uncertainty information provided by the EKF. Alongside the state estimates, the EKF outputs the covariance matrix, which quantifies the confidence in position, velocity, and obstacle range estimates. Although the NMPC does not explicitly incorporate these covariance values into its cost function, their influence is indirectly addressed through the conservative safety distance $d_{\text{safe}} = 0.5$ m and the repulsive potential

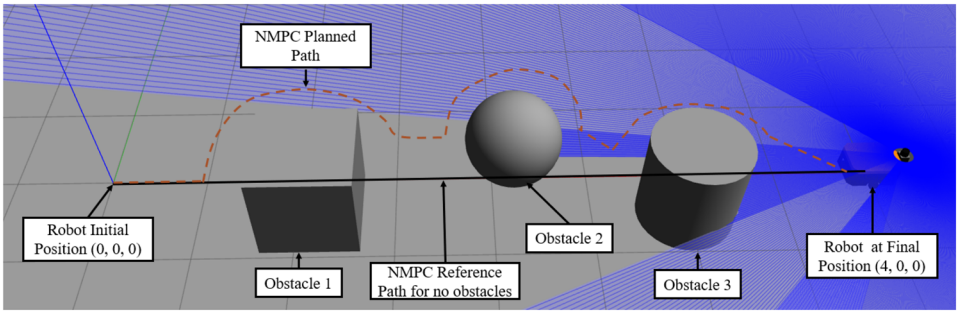

Fig. 8 Description of the simulated scenario

defined in (23) and (24). This design choice ensures that even under higher estimation uncertainty, such as during IMU drift or reduced sensor coverage, the controller maintains safe clearance from obstacles and preserves robust navigation performance.

4.4 Validation of the entire method

In this section, the EKF algorithm is connected to the MPC system and the entire control system is tested with simulations and experiments.

The simulation scenario incorporates moving and static objects as shown in Fig. 8 to emulate an active industrial floor, challenging the proposed method's adaptability to unstructured settings with unpredictable obstacles as shown in the simulation video https://youtu.be/PKRo4y3FnfQ?si=Nj4OtykR55WkpeUe.

Figure 8 shows the graphical display of the mobile robot executing a trajectory tracking and collision avoidance. The key elements of this simulation include the robot's initial position, the reference path (represented by the black line), obstacles along the path, and the robot's final position. The reference path represents the ideal trajectory the robot would follow in the absence of obstacles. However, the presence of obstacles 1 to 3 with an approximate dimension of 0.5 m $\times$ 0.5 m $\times$ 0.5 m along this path requires the MPC to modify the trajectory dynamically to ensure collision avoidance.

Figure 9 (left) presents a comparative analysis of the MPC-planned trajectory in two scenarios: one without obstacles (blue solid line) and one with obstacles (red dashed line). The obstacles, marked as black dots, disrupt the initial reference trajectory, requiring the controller to generate an alternative path to avoid collisions while maintaining trajectory tracking. A safe distance of 0.5m was enforced around each obstacle, ensuring the robot maintained sufficient clearance while navigating around them. The deviation of the red dashed line from the blue solid line illustrates how the controller dynamically modifies the trajectory to respect this safety constraint. The final position of the robot, indicated by a red '$\times$' marker, shows successful completion of the path while avoiding all obstacles. This visualization highlights the controller's ability to balance obstacle avoidance with trajectory tracking accuracy, ensuring the robot reaches its goal efficiently while maintaining a predefined safe distance from obstacles. Additionally, the use of a multisensor-based estimation algorithm provides 360-degree environmental coverage, effectively eliminating blind spots and ensuring that obstacles from all directions are detected and incorporated into the controller's avoidance strategy.

Figure 9 (right) illustrates the control actions executed by the controller. The top plot represents the linear velocity (in blue), while the bottom plot shows the angular velocity (in red) over 325 time steps. Initially, the robot maintains a relatively constant linear velocity as it follows the reference trajectory. However, as it encounters obstacles, the controller dynamically adjusts the angular velocity to navigate around them while ensuring collision avoidance. These adjustments result in three distinct fluctuations in angular velocity, each corresponding to the presence of an obstacle in the simulation. A noticeable drop in linear velocity occurs when the robot slows down or momentarily stops due to sudden obstacle avoidance placed very close to the robot and path replanning, highlighting the controller's real-time responsiveness. In the final stages, both velocities gradually decrease as the robot reaches its target position, ensuring a smooth and controlled stop. This result effectively demonstrates the proposed controller's ability to generate safe and adaptive control commands while maintaining trajectory tracking performance in a dynamic environment such as a factory floor.

Experiment scenario: Starting from (0, 0, 0), the robot followed the commands generated by the controller to reach (4, 0, 0) in the presence of two static obstacles and one slow-moving obstacle placed along the nominal path, as shown in the experimental video https://youtu.be/xvyHcYzQo6M?si=dfUhdXOJAnSwa6xJ.

Figure 10 (left) illustrates the trajectory of the robot in the presence of two static points at coordinates (1, 0, 0) and (2, 0, 0), along with a moving obstacle that suddenly appears in the robot's path to the coordinate (3, 0, 0). The robot effectively adjusts its trajectory at each point where the obstacle is detected, demonstrating the functionality of the developed obstacle avoidance system. This capability is critical for ensuring safe navigation in dynamic environ-

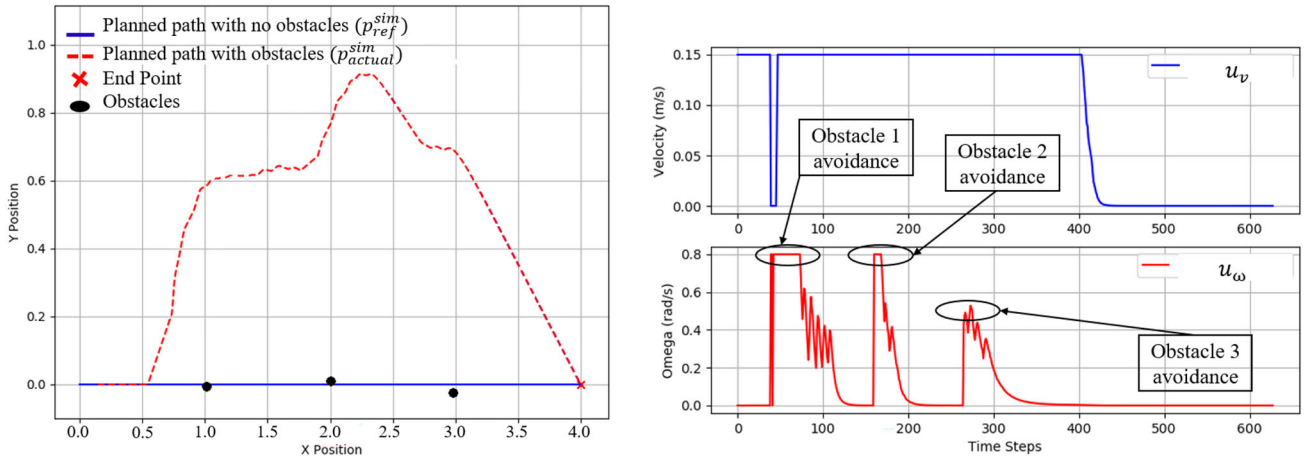

Fig. 9 Left: comparison of the planned path with and without obstacles against the position of the obstacles. Right: robot control actions in an environment with statics and moving obstacles

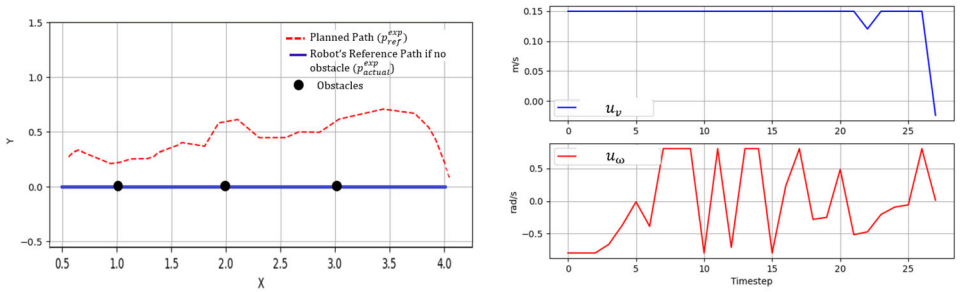

Fig. 10 Left: Planned trajectory. Right: robot control actions in an environment with static and moving obstacles

ments. Toward the end of the maneuver, the robot attempts to approach the goal, indicating that the trajectory modification was successful while maintaining the objective of reaching the target. In terms of control actions, the linear velocity remains almost constant throughout the motion, as shown in Fig. 10 (right). Meanwhile, the angular velocity varies dynamically in response to the maneuvers required to avoid the obstacles. This responsiveness reflects the system's effectiveness in adapting to changes in the environment.

In addition to trajectory performance, the proposed framework was also evaluated in terms of computational efficiency and suitability for embedded implementation. All experimental trials were executed on a Raspberry Pi 4 (Quad-core Cortex-A72 @ 1.5 GHz, 4 GB RAM). The EKF update step required less than 26 ms per cycle, while the NMPC optimization (CasADi with Ipopt) converged within 40–55 ms depending on obstacle density. This enabled the full estimation control loop to run reliably at 10 Hz ($\Delta t = 0.1$ s), which was sufficient for safe operation at the low translational speeds considered in the experiments (± 0.15 m/s). These results confirm that the proposed framework achieves real-time performance on embedded hardware, highlighting its potential for practical deployment in low-speed robotic applications.

Comparison of simulation and experiment results: Statistical values presented in Table 1 are computed for the simulation results in Fig. 9 (left) and experiment results in Fig. 10 (left). The simulation exhibits a higher mean error 0.50m and maximum deviation 1.02m from the reference path than the experimental case (mean error of 0.35m and maximum deviation of 0.75m). This significant difference can be attributed to the nature and shape of the obstacles used in each environment. In the simulation, the robot is required to avoid larger geometric objects with an approximate dimension of 0.5 m $\times$ 0.5 m $\times$ 0.5 m such as *cubes, cylinders, and spheres*, which occupy more physical space. Although the obstacles are plotted as center points in the graph, the robot perceives and reacts to the full volume of these objects, as detected through LiDAR and RADAR sensors. This forces the controller to generate paths with larger deviations, particularly in the Y-direction, to ensure safe avoidance.

In contrast, the experimental environment featured narrower obstacles such as *human legs and plastic bottles*. These objects require less spatial clearance, which can be approximated to 0.15–0.2m, allowing the robot to follow a path that remains closer to the reference trajectory. Consequently, the experimental trajectory records a lower RMSE of 0.40m and a smaller SD of 0.20m. It is important to note that the lower RMSE and SD in the experimental case do not imply

Table 1 Summary of Trajectory Tracking Metrics

Metric	p^{sim}_{actual} vs p^{sim}_{ref}	p^{exp}_{actual} vs p^{exp}_{ref}	p^{sim}_{actual} vs p^{exp}_{actual}
Mean error (m)	0.50	0.35	0.28
Max error (m)	1.02	0.75	0.56
RMSE (m)	0.62	0.40	0.23
SD (m)	0.32	0.20	0.17

greater system consistency or accuracy. Rather, they reflect the less intrusive shapes and sizes of the obstacles encountered. In simulation, larger obstacles naturally result in more pronounced deviations, which elevate both RMSE and SD.

The comparison between the simulation and experimental trajectories further reveals a relatively small RMSE of 0.23m between the two, indicating that both trajectories follow a similar general trend, despite environmental differences in obstacle characteristics and perception. It is observed from Figs. 9 (left) and 10 (left) that neither the simulation nor the experimental trajectories begin precisely at the origin. This misalignment likely stems from initialization delays and synchronization issues in data logging, such as the system beginning to record only after the robot has moved or after the multisensor algorithm has stabilized. These minor time shifts contribute to the initial offsets seen in both trajectories.

The results demonstrate that the proposed EKF and NMPC framework achieves safe navigation with both static and dynamic obstacles, while maintaining computational feasibility on a Raspberry Pi platform. In comparison with recent state-of-the-art methods, LiDAR-only NMPC approaches such as [8] achieve accurate trajectory tracking but are vulnerable to blind-spot limitations. Gerdpratoom and Yamamoto [15] addressed flock navigation using decentralized NMPC with local point-cloud data for obstacle avoidance. While their strategy relies on point-cloud processing, our framework incorporates the obstacle distance directly into the NMPC objective function, allowing the controller to proactively adapt the trajectory by accounting for obstacle motion as part of the optimization. This design reduces reliance on dense point-cloud computation and enables embedded real-time execution. From a practical standpoint, the achieved RMSE of 0.40 m is within the enforced 0.5 m safety margin, confirming that the system preserves safe clearance in real-world operation.

Nonetheless, some limitations should be noted. The experiments were conducted at relatively low robot speeds of ± 0.15 m/s and in a confined laboratory space of 4×4 m, which may not capture all dynamics of larger industrial settings. Additionally, while the EKF provides covariance estimates of uncertainty, these were not explicitly incorporated into the NMPC cost function. Finally, solver performance was validated at 10 Hz, but operation at higher speeds or with denser obstacle fields may require further optimization. These limitations, however, represent opportunities for extending the framework to faster platforms, larger workspaces, and uncertainty-aware control formulations.

5 Conclusion

This paper presented a methodology for enhancing collision avoidance in industrial mobile robots. By incorporating obstacle avoidance directly into the MPC cost function and using the multisensor-based system for detection, the framework enables real-time trajectory adjustments, ensuring safe and efficient robot navigation in dynamic workspaces, even in the LiDAR blind spots. Simulation and experimental results confirm that the proposed framework demonstrated its ability to track reference trajectories precisely, dynamically adjust the robot's path in response to obstacles, and maintain a safe operating margin. The system prioritizes safety in industrial environments by halting movement only when no feasible path exists, resuming navigation once an obstacle-free trajectory is available. The proposed method effectively balances trajectory tracking, real-time obstacle avoidance, and sensor-based perception, demonstrating its potential for deployment in industrial settings where safety and operational efficiency are paramount. However, IMU-based state estimation remains susceptible to drift over extended operation and during maneuvers, which could affect long-term navigation accuracy. Future work will focus on adaptive filtering techniques to mitigate IMU drift and enhance long-term stability.

Acknowledgements This work was financially supported in part by Kyambogo University competitive research grants scheme (KYU09-24/25) and in part by the African Union through Pan African University Institute of basic Science, Technology and Innovation (PAUSTI), Jomo Kenyatta University of Agriculture and Technology.

References

1. Takala J, Hämäläinen P, Saarela KL, Yun LY, Manickam K, Jin TW, Heng P, Tjong C, Kheng LG, Lim S et al (2014) Global estimates of the burden of injury and illness at work in 2012. J Occup Environ Hyg 11(5):326–337
2. Sisbot EA, Marin-Urias LF, Alami R, Simeon T et al (2007) A human aware mobile robot motion planner. IEEE Trans Rob 23(5):874–883
3. Lasota PA, Fong T, Shah JA et al (2017) A survey of methods for safe human-robot interaction, Foundations and Trends ® in Robotics, 5(4): 261–349
4. Calcroft M, Khan A (2022) Lidar-based obstacle detection and avoidance for autonomous vehicles using raspberry pi 3b, in 2022 UKACC 13th International Conference on Control (CONTROL). IEEE, 24–29
5. Ueyama Y, Sago T, Kurihara T, Harada M (2022) An inexpensive autonomous mobile robot for undergraduate education Integration of arduino and hokuyo laser range finders, IEEE Access, 10:79 029–79 040
6. Sudianto AI, Muslim MA, Rusli M (2022) Collision avoidance system with model predictive control for mobile robot navigation, in, 2022 International Conference of Science and Information Technology in Smart Administration (ICSINTESA). IEEE 204–209
7. Sani M, Robu B, Hably A (2021) Dynamic obstacles avoidance using nonlinear model predictive control, in IECON 2021–47th annual conference of the IEEE industrial electronics society. IEEE, 1–6
8. Ismael OY, Almaged M, Abdulla AI (2024) Nonlinear model predictive control-based collision avoidance for mobile robot. J Robot Control (JRC) 5(1):142–151
9. Jian Z, Yan Z, Lei X, Lu Z, Lan B, Wang X, Liang B (2023) Dynamic control barrier function-based model predictive control to safety-critical obstacle-avoidance of mobile robot, in 2023 IEEE International Conference on Robotics and Automation (ICRA). Ieee, 3679–3685
10. Arnold E, Al-Jarrah OY, Dianati M, Fallah S, Oxtoby D, Mouzakitis A (2019) A survey on 3d object detection methods for autonomous driving applications. IEEE Trans Intell Transp Syst 20(10):3782–3795
11. Mikumo R, Ichihara H (2017) Dynamic collision avoidance among multiple mobile robots: A model predictive control approach, in, 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE). IEEE 1136–1137
12. Wang Z, Wu Y, Niu Q (2019) Multi-sensor fusion in automated driving: A survey. Ieee Access 8:2847–2868
13. Alajlan AM, Almasri MM, Elleithy KM (2015) Multi-sensor based collision avoidance algorithm for mobile robot, in 2015 Long Island Systems, Applications and Technology. IEEE 1–6
14. Liang J, Patel U, Sathyamoorthy AJ, Manocha D (2020) Real-time collision avoidance for mobile robots in dense crowds using implicit multi-sensor fusion and deep reinforcement learning, arXiv:2004.03089
15. Gerdpratoom N, Yamamoto K (2025) Decentralized nonlinear model predictive control-based flock navigation with real-time obstacle avoidance in unknown obstructed environments, Frontiers in Robotics and AI, 12, [Online]. Available: https://doi.org/10.3389/frobt.2025.1540808
16. Cooper MA, Raquet JF, Patton R (2018) Range information characterization of the hokuyo ust-20lx lidar sensor, in Photonics, 5(2) MDPI:12
17. Yan H, Shan Q, Furukawa Y (2018) Ridi: Robust imu double integration, in Proceedings of the European conference on computer vision (ECCV): 621–636
18. Chen H, Ding S, Chen X, Wang L, Zhu C, Chen W (2014) Global finite-time stabilization for nonholonomic mobile robots based on visual servoing. Int J Adv Rob Syst 11(11):180
19. Samuel K, Choi J (2018) Improved imm filter for tracking a highly maneuvering target with mixed system noises. Int J Control Autom Syst 16:2763–2771
20. Samuel K, Choi JW (2018) Correlated system noise error tracking filter design for a sharply maneuvering ground target, in 2018 57th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE). IEEE:626–631
21. Choi JW, Samuel K (2019) Robust ukf-imm filter for tracking an off-road ground target. Int J Control Autom Syst 17:1149–1157