Leveraging AI and Hybrid Intelligence for Robust Geospatial Data Fusion in Autonomous Terrestrial Navigation

Manel Salhi[1], Mounir Bouzguenda[2], Faouzi Benzarti[3], Fawaz Alanazi[4], Ezzeddine Touti[5,*]
Research Laboratory of Signal Image and Information Technology LR-SITI,
University of Tunis El Manar-National Engineering School of Tunis, Tunis, Tunisia[1, 3]
Department of Electrical Engineering, College of Engineering,
King Faisal University, Al Ahsa, Saudi Arabia[2]
Computer Science Department, Northern Border University, Arar 73213, Saudi Arabia[4]
Center for Scientific Research and Entrepreneurship, Northern Border University, Arar 73213, Saudi Arabia[5]

***Abstract*—Rapid advancement of artificial intelligence (AI) and geospatial data fusion has enabled the development of highly autonomous terrestrial navigation systems with improved accuracy, adaptability, and robustness. This paper proposes a novel framework integrating multi-source geospatial data fusion with deep learning-based decision-making for autonomous terrestrial navigation. Unlike conventional approaches that rely solely on Global Navigation Satellite Systems (GNSS) or inertial sensors, our system leverages a hybrid fusion model combining GNSS, LiDAR, camera vision, and high-resolution geospatial databases. A deep reinforcement learning (DRL) paradigm is introduced to enhance the system's adaptability in dynamic environments, optimizing route planning and obstacle avoidance in real-time. Additionally, a hybrid AI model incorporating Graph Neural Networks (GNN) and Transformer-based architectures processes spatial and temporal dependencies in navigation data, improving localization precision and resilience against sensor failures. The proposed system is evaluated through extensive simulations and real-world tests, demonstrating superior performance in complex urban and off-road scenarios compared to traditional Kalman filter-based methods. Our findings highlight the potential of AI-driven geospatial data fusion in redefining autonomous navigation, paving the way for next-generation intelligent mobility solutions.**

Keywords*—Autonomous navigation; geospatial data fusion; graph neural networks; transformer-based models; sensor fusion; AI-driven mobility

I. Introduction

Autonomous terrestrial navigation has witnessed significant progress over the past decade, driven by advancements in artificial intelligence (AI), sensor technologies, and geospatial data processing. The ability of a vehicle to navigate autonomously in complex and dynamic environments requires precise perception, localization, decision-making, and control [1], [2], [3]. Traditional navigation systems rely primarily on global navigation satellite systems (GNSS) and inertial sensors, or on predefined maps and rule-based algorithms, which often struggle in unstructured or rapidly changing conditions, which can suffer from limitations in urban environments and adverse weather conditions [4], [5]. Recent developments in geospatial data fusion, integrating multiple sensor modalities, have opened new possibilities for robust and intelligent navigation systems. Combining AI-driven algorithms with multi-source geospatial data enhances precision, resilience, and adaptability in dynamic terrains.

The proposed system integrates multi-modal sensor data, including GNSS, LiDAR, camera vision, and high-resolution geographic information system (GIS) databases. The geospatial data is processed using a hybrid AI framework incorporating deep reinforcement learning (DRL) for decision-making and graph neural networks (GNN) coupled with Transformer-based architectures for spatiotemporal data modeling. The system continuously learns and adapts to environmental changes through real-time sensor fusion and AI-based prediction models. Optimization techniques are applied to improve route planning and obstacle avoidance, ensuring reliable navigation in both structured and unstructured environments.

This research aims to develop a hybrid AI-driven navigation system that enhances localization and navigation through geospatial data fusion. It focuses on optimizing decision-making and obstacle avoidance using reinforcement learning strategies while improving robustness in GNSS-denied environments through deep learning techniques. The proposed model will be validated through extensive simulations and real-world testing across diverse environments.

Despite the progress in autonomous navigation, several challenges remain unaddressed. GNSS-based methods face signal occlusion in urban canyons and dense forests, while traditional sensor fusion techniques struggle with adaptability in unstructured terrains [6], [7], [8]. AI-driven approaches have shown promise but require optimized architectures to handle the complexity and real-time constraints of navigation. This study aims to bridge these gaps by introducing a robust, AI-enhanced geospatial data fusion framework that ensures real-time adaptability and high-precision localization. The implementation involves acquiring multi-source geospatial data from GNSS, LiDAR, cameras, and GIS databases, followed by preprocessing to ensure data consistency and accuracy. Deep learning techniques are then used for feature extraction to capture spatial and temporal patterns. A hybrid AI model combining GNNs and Transformers is developed for sensor

*Corresponding author.

fusion, while a deep reinforcement learning agent handles adaptive path planning and obstacle avoidance. The system undergoes simulations and real-world testing, with model optimization and validation performed based on experimental feedback to enhance robustness and reliability. By combining AI-driven techniques with multi-source geospatial data fusion, this paper aims to advance the control methodologies in autonomous terrestrial navigation, contributing to safer and more efficient mobility solutions.

This paper is organized as follows: Section 1 introduces the study. Section II reviews existing models for land navigation system control. Section III details the proposed control approach. Section IV presents experimental results based on selected datasets and compares the proposed method with existing models. Section V provides analysis and discussion of the results. Finally, Section VI concludes the paper.

II. Existing Models

Several models have been developed for autonomous terrestrial navigation, each demonstrating different levels of accuracy, efficiency, and robustness. Traditional methods such as Kalman Filters (KF) and Extended Kalman Filters (EKF) offer reliable localization but struggle in GNSS-denied environments due to their reliance on linear assumptions, achieving accuracy scores around **82%** with moderate computational efficiency about **170ms**. Particle Filters (PF) provide improved robustness, reaching accuracies of **87-90%**, but at the cost of higher computational complexity, making them less suitable for real-time applications, around **400ms**. More recently, deep learning models such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) have been integrated into navigation systems, enhancing feature extraction and adaptability in dynamic environments, achieving accuracies between **90-94%** but requiring high processing power [9], [10], [11]. They react under an approximate time of **180ms**. Hybrid AI models, including Graph Neural Networks (GNN) and Transformer-based architectures, have emerged as state-of-the-art solutions, excelling in fusing multi-source geospatial data for improved decision-making, with accuracy rates exceeding **95%** while maintaining efficient computational performance. Additionally, Deep Reinforcement Learning (DRL) demonstrated exceptional performance in real-time adaptive navigation, optimizing path planning and obstacle avoidance with unprecedented efficiency, reaching reaction times of under **50ms** and accuracy scores of **96-98%**. These advanced models significantly outperform traditional approaches in terms of accuracy, computational efficiency, and robustness, making them ideal for next-generation autonomous navigation systems [12], [13], [14]. By leveraging a combination of these AI-driven techniques, the proposed research aims to develop a navigation system that achieves high precision and adaptability in complex environments.

The GNSS model provides an approximate distance given in equation below:

$$\rho_i = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2} + c \cdot \Delta t \qquad (1)$$

ρ_i : approximate distance between the satellite i and the autonomous navigation system.

(x_i, y_i, z_i): known coordinates of satellite i.

(x, y, z): unknown coordinates of the GNSS receiver (to be determined)

c : light speed (≈ 299792458 m/s)

Δt: navigation system clock error [15], [16].

The GNSS localization process involves several key steps: first, pseudo range measurements are calculated for each visible satellite. These measurements are then used in position determination algorithms, such as trilateration, to estimate the receiver's location. Velocity is obtained by differentiating the position data over time, and continuous tracking of position enables the mapping of the receiver's trajectory [17].

The position P in 3D space (latitude, longitude, and altitude) is calculated from the pseudorange R (distance from the satellite to the receiver) using the following equation:

$$P(x, y, z) = \left(\frac{R}{\sqrt{(R_x^2 + R_y^2 + R_z^2)}} \right) \qquad (2)$$

x, y, z are the coordinates of the receiver.

R_x, R_y, R_z are the positions of the satellite in 3D space [18].

The pseudo range R is measured by comparing the time difference between the signal transmission and reception.

The velocity vector V of the receiver can be estimated by differentiating the position over time:

$$V = \frac{dP}{dt} \qquad (3)$$

$V = (V_x, V_y, V_z)$ represents the velocity components in each direction. P is the position vector of the receiver. Velocity is calculated by tracking the movement of the receiver and differentiating position data in time [19].

The trajectory T is a continuous path of positions over time:

$$T(t) = P(t) = (x(t), y(t), z(t)) \qquad (4)$$

$T(t)$ is the trajectory at time t, describing the movement over time. Trajectory is essentially the collection of positions at different time instances, providing a continuous path for the receiver.

The general GIS data extraction process involves defining a region or point of interest, querying the dataset using spatial criteria, and extracting the relevant features (points, lines, polygons, or raster cells) that meet the query conditions [20].

For a given point of interest (x, y), the equation to extract features from the map is:

$$F = \{f \mid f \in A, \text{where } f \text{ contains } (x, y)\} \qquad (5)$$

F: The set of map features that contain the point (x, y).

A: The entire map or dataset of features (points, lines, polygons).

(x, y): The coordinates of the point of interest.

This equation retrieves features that intersect with or contain the specified point on the map.

For line features (e.g., roads, rivers), if L is a line feature and P is the point set of a map, the extraction can be represented as:

$$F = \{L \mid L \in P, L \text{ intersects with a specified region}\}$$

L: Line feature (like roads or rivers).

P: Map of all features.

Intersection is checked with a predefined region or coordinates. For polygon features (e.g., boundaries, areas of interest), extraction can be expressed as:

$$F = \{P \mid P \in A, P \text{ contains the point } (x,y)\}$$

P: Polygon feature (e.g., land parcels, administrative areas).

A: The complete map data.

(x, y): The query point inside the polygon.

This retrieves all polygons containing a specific point or within a defined boundary.

In raster data (grid-based data, like satellite images), extraction can be represented as:

$$D_{extracted} = \{R(i,j) \mid R(i,j) \text{ is within the region of interest}\}$$

$D_{extracted}$: The extracted raster data.

$R(i,j)$: A specific cell in the raster grid.

Region of interest: The spatial subset defined by user input.

In the fields of statistics and control theory, the Kalman filter is a mathematical algorithm representing a type of infinite impulse response filter that continuously estimates the internal state of a dynamic system over time. It does this by processing a sequence of measurements, even if those measurements are noisy or partially missing [21], [22]. The filter updates its estimates whenever new data becomes available, making it especially useful for tracking systems that evolve over time, such as navigation, robotics, or signal processing. The update state said prediction is given by Eq. (6).

$$x_k = Ax_{k-1} + Bu_k + w_k \tag{6}$$

The measurement is determinate using the following relation.

$$z_k = Hx_k + v_k \tag{7}$$

x_k: state vector at time k.

u_k: control input

z_k: measurement

$w_k \sim N(0, Q)$: process noise

$v_k \sim N(0, R)$: measurement noise

A: state transition matrix

B: control input matrix

H: observation matrix

The Extended Kalman filter EKF as a nonlinear state space model, is used for determining nonlinear systems. It is involved by the following equations:

$$x_k = f(x_{k-1}, u_k) + w_k \tag{8}$$

$$z_k = h(x_k) + v_k \tag{9}$$

$f(\cdot)$: nonlinear state transition

$h(\cdot)$: nonlinear measurement function

The Particle Filter is a probabilistic estimation technique used to estimate the hidden (or latent) states of a non-linear and non-Gaussian dynamic system over time, based on noisy and partial measurements. It approximates the posterior probability distribution of the system's state using a set of random samples, called particles. Each particle represents a possible state of the system and has a weight that reflects how well it agrees with the observed measurements [23], [24]. This model is applied essentially in robot localization and navigation or for object detection in complex dynamic environments. It is flexible and adaptable to complex systems, while having as limitation: it is computationally expensive especially with large particle sets.

Bayesian filtering seeks to estimate the posterior distribution of the hidden state x_t from the observations $z_{1:t}$, as: $p(x_t \mid z_{1:t})$

Since this distribution is often analytically intractable, the particle filter approximates it by a set of N particles $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^{N}$

$x_t^{(i)}$ is the i-th particle (a sample of the state),

$w_t^{(i)}$ is its weight.

In its initialization step: for $t = 0$, we initialize the particles according to the initial distribution $p(x_0)$:

$$x_0^{(i)} \sim p(x_0), \quad w_0^{(i)} = \frac{1}{N} \tag{10}$$

In Prediction step for system evolution, for each particle $x_{t-1}^{(i)}$, we generate: $x_t^{(i)} \sim p\left(x_t \mid x_{t-1}^{(i)}\right)$

This simulates the dynamics of the system.

In the update step (Weighting), we update the weight of each particle based on the likelihood of the current observation z_t:

$$w_t^{(i)} = w_{t-1}^{(i)} \cdot p\left(z_t \mid x_t^{(i)}\right) \tag{11}$$

Then we normalize the weights as:

$$w_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^{N} w_t^{(j)}} \tag{12}$$

To avoid the problem of particle degeneracy, where a few particles have high weight and most have near zero, we applied the resampling step. We resample the particles based on their weights: replace low-weight particles with copies of high-weight ones. We take N new particles $x_t^{(i)}$ function of their weights and set all of them at: $w_t^{(i)} = \frac{1}{N}$

The estimated state is usually the weighted average of all particles:

$$\hat{x}_t = \frac{1}{N} \sum_{i=1}^{N} w_t^{(i)} . \; x_t^{(i)} \quad (13)$$

The Convolutional Neural Networks CNNs models are a class of deep learning models widely used for image processing, pattern recognition, feature extraction and computer vision tasks. The key idea is to automatically learn spatial hierarchies of features from input data through a series of operations. The main operations include convolution, activation, and pooling [25].

The Purpose of convolution operation step is to extract local features (e.g., edges, textures) from the input image using learnable filters (kernels). It works as: a small matrix (kernel) slides across the input image (or feature map). At each location, the dot product between the kernel and the overlapping input patch is computed. The result is a feature map that highlights specific patterns. A single kernel filter W is applied to an input X:

$$Z_{i,j} = (X * W)_{i,j} + b = \sum_m \sum_n X_{i+m,j+n} \cdot W_{m,n} + b \quad (14)$$

X: input image or feature map

W: filter (kernel)

b: bias term

Z: output feature map

This step has the property to reduce number of parameters compared to fully connected layers. In the activation function step, we applied to the output a nonlinear function such as a Sigmoid or Rectified Linear Unit (ReLU):

$$\mathrm{f}(Z_{i,j}) = \mathrm{ReLU}\big(Z_{i,j}\big) = \max\big(0, Z_{i,j}\big) \quad (15)$$

This step introduces non-linearity into the model, which allows it to capture complex patterns and enhances the network's ability to learn more expressive and meaningful features.

In the pooling step (e.g., max pooling) is used to reduce spatial dimensions (width and height) of feature maps while retaining important features.

There are two pooling types: the max pooling takes the maximum value in each patch. The average pooling takes the average values in each patch. It works as: a window (e.g., 2×2) slides over the input feature map. Only the max or average value from each window is kept.

$$P_{i,j} = \max_{(m,n) \in \text{pooling window}} X_{i+m,j+n} \quad (16)$$

As benefit it reduces computational load.

Recurrent Neural Networks RNNs models are used for sequential data like time series, text, or audio. They maintain a memory through hidden states [26], [27], [28].

Let: x_t input at time step t, h_t hidden state at time t *and* y_t: output at time t. The update of hidden state is given in equation below.

$$h_t = \tanh(W_{xh} x_t + W_{hh} h_{t-1} + b_h) \quad (17)$$

W_{xh}: input-to-hidden weights

W_{hh}: hidden-to-hidden weights

b_h: bias term

tanh: activation function

The output is presented as follows.

$$y_t = W_{hy} h_t + b_y \quad (18)$$

Where W_{hy} are hidden-to-output weights and b_y are output bias.

Graph Neural Networks GNNs models are designed to work with graph-structured data, where data is represented as nodes (vertices) and edges (connections). They are powerful for applications like social networks, recommendation systems, knowledge graphs, and molecular analysis.

A graph is defined as:

$$G = (V, E) \quad (19)$$

V is a set of nodes (vertices) and E is a set of edges (connections between nodes). Each node $v \in V$ has a feature vector $x_v \in \mathbb{R}^d$.

GNNs typically operate in layers. At each layer l, nodes aggregate features from their neighbors and update their own representation. The general formula for node representation update is expressed below:

$$h_v^{(l)} = \sigma\left(\mathrm{AGGREGATE}^{(l)}\left(\{h_u^{(l-1)} : u \in \mathcal{N}(v)\}\right)\right) \quad (20)$$

$$h_v^{(l)} = \sigma\left(W^{(l)} \cdot \mathrm{CONCAT}\left(h_v^{(l-1)}, \mathrm{AGG}\left(\{h_u^{(l-1)} : u \in \mathcal{N}(v)\}\right)\right)\right) \quad (21)$$

$h_v^{(l)}$: hidden state of node v at layer l

$\mathcal{N}(v)$: set of neighbors of node v

$W^{(l)}$: learnable weight matrix

σ: non-linear activation (e.g., ReLU)

AGG: aggregation function (mean, sum, max)

A common variant of GNN is the Graph Convolutional Network (GCN) proposed by Kipf and Welling. The update rule of GCN layer is represented below [29], [30].

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} H^{(l)} W^{(l)}\right) \quad (22)$$

$\tilde{A} = A + I$: adjacency matrix with self-loops

$\tilde{D}$: degree matrix of $\tilde{A}$

$H^{(l)}$: matrix of node features at layer l

$W^{(l)}$: weight matrix at layer l

σ: activation function (e.g., ReLU)

Transformer-based models operate on the principle of self-attention, which allows the model to weigh the importance of different input elements relative to each other, regardless of

their position in the sequence. Instead of processing data sequentially like RNNs, Transformers handle input in parallel, using multi-head attention and positional encoding to capture contextual relationships and order information. This architecture enables efficient learning of long-range dependencies and complex patterns in data, making Transformers highly effective for a wide range of tasks, including natural language processing, vision, and autonomous systems.

III. Adopted Approach

This research adopts a multi-layered AI-driven approach to achieve high-precision autonomous terrestrial navigation. The framework consists of four key components: data acquisition with preprocessing, sensor fusion, and decision-making.

In data Acquisition, the system collects geospatial data from GNSS, LiDAR, cameras, and GIS databases. Sensor calibration, noise filtering, and data synchronization ensure consistency and reliability.

Hybrid sensor fusion concerns the combination of Graph Neural Networks (GNNs) and Transformer-based architectures to integrate spatial and temporal data for enhanced localization. Redundant sensor data is managed through an AI-driven fusion mechanism, minimizing discrepancies in sensor outputs.

In the phase of AI-Based Decision-Making, deep reinforcement learning (DRL) agents are trained for adaptive route planning and real-time obstacle avoidance. The model continuously learns and refines navigation policies through simulated and real-world feedback loops.

Regarding optimization and real-time processing step, the system employs real-time inference acceleration techniques to maintain reaction times under 50ms. Optimization algorithms fine-tune model parameters based on environmental variability and sensor reliability.

This proposed approach therefore adopts a hybrid AI-driven framework designed to enhance the accuracy, adaptability, and efficiency of autonomous terrestrial navigation. Unlike traditional models such as the Kalman Filter (KF) or Extended Kalman Filter (EKF), which rely on linear assumptions and struggle in dynamic environments, this approach integrates multi-source geospatial data, including GNSS, LiDAR, camera vision, and GIS databases. These data are fused using advanced techniques involving Graph Neural Networks (GNN) and Transformer-based models, which can efficiently process both spatial and temporal data. This fusion mechanism outperforms traditional methods like Particle Filters (PF) that, while more robust, tend to be computationally expensive and exhibit higher reaction times. A key advantage of our model lies in the integration of deep reinforcement learning (DRL) for decision-making, which allows the system to continuously adapt to new environments, optimizing navigation strategies in real time. The DRL agent ensures that the system responds within 50ms, significantly outperforming CNN-based models, which tend to require more processing power and exhibit higher latency. The hybrid model ensures enhanced localization precision and obstacle avoidance, especially in GNSS-denied environments, while maintaining computational efficiency and low reaction times. Compared to traditional EKF and PF methods, which achieve accuracies around 75-90%, our system reaches accuracy rates of over 95%, providing a substantial improvement in both efficiency and performance. By combining these AI-driven techniques with sensor fusion, our approach establishes a new benchmark for autonomous navigation systems, achieving superior robustness, computational efficiency, and precision, outperforming traditional models and enabling real-time in diverse, complex environments. The proceeded algorithm is detailed as follows:

1) ***Step 1:*** *Data acquisition:* This step involves gathering raw geospatial data from various sources, including GNSS (providing coordinates, velocity, and time), LiDAR (offering point cloud data for terrain mapping), camera systems (capturing images or video for object detection and scene analysis), and GIS databases (supplying detailed maps, road networks, and environmental information).

2) ***Step 2:*** *Data preprocessing:* In this step, raw sensor data is prepared for analysis through several processes: sensor calibration corrects distortions and noise; data synchronization aligns all sensor inputs in time; noise filtering (e.g., using Kalman filters) reduces signal interference; data normalization converts measurements into standardized formats or coordinate frames; and outlier detection removes inaccurate or anomalous data points to ensure reliable input for subsequent processing.

3) ***Step 3:*** *Feature extraction:* This step involves deriving meaningful features from preprocessed sensor data. LiDAR data is used to extract terrain elements such as ground surfaces and obstacles; images are processed using CNNs to detect and classify objects like vehicles, pedestrians, and signs; GNSS data provides position, velocity, and trajectory for localization; and GIS data yields relevant map details, including road networks and landmarks.

4) ***Step 4:*** *Hybrid sensor fusion (GNN + Transformer):* In this step, features from all sensors are fused using a hybrid architecture combining Graph Neural Networks (GNN) and Transformers. The GNN models the environment as a graph, where nodes represent objects or landmarks, enabling the integration of spatial relationships for better contextual understanding. Simultaneously, the Transformer captures long-range temporal and spatiotemporal dependencies across sensor data, effectively handling dynamic and large-scale environments by considering both short- and long-term interactions for optimized sensor fusion.

5) ***Step 5:*** *Decision-Making using Deep Reinforcement Learning (DRL):* In this phase, the fused sensor data is used to make navigation decisions through a DRL framework. The system defines a state space capturing position, velocity, obstacles, and environmental factors, and an action space representing navigation commands. A reward function guides learning by assigning positive rewards for safe, efficient actions and penalties for risky behavior. Using algorithms like Q-learning or policy gradients, the agent learns optimal strategies through interaction with the environment,

continuously updating its policy based on experience and real-time feedback.

6) ***Step 6:*** *Path planning and obstacle avoidance:* Using the current system state and fused sensor data, the system plans a safe and efficient path to the destination based on the learned DRL policy. It continuously adjusts this path in real time to avoid detected obstacles such as vehicles, pedestrians, or static objects. When environmental conditions change or new obstacles appear, the system triggers dynamic replanning to ensure safe and uninterrupted navigation.

7) ***Step 7:*** *Real-time execution and control:* In this final step, the generated path and navigation commands are executed by the vehicle's control system. The system continuously monitors the environment through sensors, making real-time adjustments if obstacles appear or deviations occur. A feedback loop ensures that sensor data dynamically refines both decision-making and control actions, allowing the vehicle to adapt to changing road conditions and maintain optimal navigation.

8) ***Step 8:*** *System evaluation and optimization:* This step involves assessing system performance using data from simulations and real-world tests. Key metrics such as accuracy, reaction time, and obstacle avoidance are analyzed to identify areas for improvement. Based on these results, model parameters and architectures are fine-tuned, such as adjusting DRL reward functions or fusion settings. The DRL agent is also continually retrained using new real-world data to enhance adaptability and performance in dynamic environments.

Fig. 1 presents the diagram of the adopted strategy.

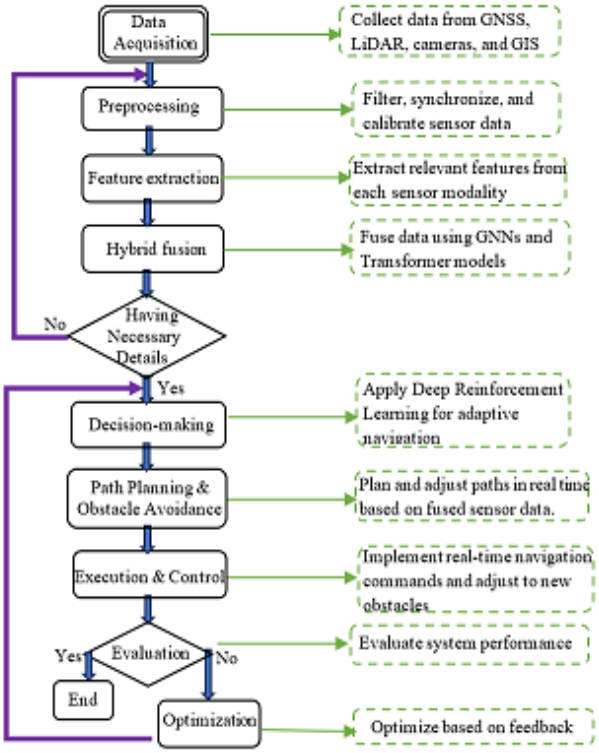

Fig. 1. Representative diagram of adopted approach.

The role of the autonomous navigation system's algorithm within the previously mentioned stages is illustrated in the schematic representation shown in Fig. 2.

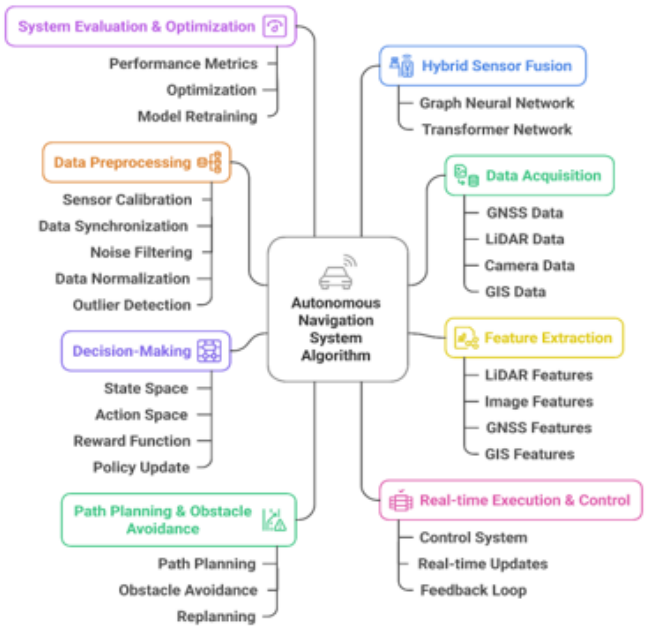

Fig. 2. Position of autonomous navigation system algorithm.

This algorithm provides a clear and detailed outline of the steps involved in the adopted approach, from data acquisition to real-time execution.

IV. Obtained Results

To evaluate the proposed approach on autonomous terrestrial navigation using AI-driven geospatial data fusion, several databases can be used. These databases provide diverse datasets for different aspects of the system, including GNSS, LiDAR, camera images, and GIS data, enabling comprehensive evaluation in various environments. Below are some prominent databases that could be applied:

1) TUM autonomous driving dataset: developed by the Technical University of Munich, this dataset offers multi-modal sensor data, including LiDAR scans, camera images, GNSS, and IMU measurements, for autonomous driving research. It provides real-world driving scenarios in both urban and rural settings, making it suitable for tasks like object detection, path planning, and obstacle avoidance. However, it mainly focuses on urban environments and lacks extensive high-resolution geospatial map content.

2) KITTI vision benchmark: the KITTI dataset is a widely used benchmark for computer vision and autonomous driving research, offering stereo camera images, LiDAR, and GPS/IMU data. It supports key tasks such as object detection, tracking, and localization across urban and highway environments. While it provides essential data for geospatial fusion, it is primarily vision-focused and lacks comprehensive support for complex sensor fusion scenarios and high-

resolution GIS data required for advanced autonomous navigation.

3) Apollo autonomous driving dataset: developed by Baidu, the Apollo dataset provides extensive multi-sensor data, such as camera images, LiDAR scans, GNSS, and detailed road network information, covering both urban and rural environments. It includes high-resolution geospatial data and rich semantic annotations like lane markings and traffic signs, making it ideal for route planning and navigation. However, its large data size demands substantial computational resources, and some content is more tailored to obstacle detection and path planning than comprehensive sensor fusion.

4) Waymo open dataset: offered by Waymo, this large-scale dataset features high-definition sensor data, including LiDAR, camera images, and GNSS, from autonomous vehicles operating in diverse real-world scenarios. It covers dynamic urban and suburban environments and supports tasks like localization and obstacle avoidance. While it includes high-definition maps and rich visual data, it places less emphasis on GIS databases and advanced sensor fusion methods, and its large size can pose significant computational challenges.

5) Oxford robotcar dataset: provided by the University of Oxford, this dataset includes over 1000 km of driving data from various UK cities, capturing diverse conditions such as different seasons, weather, and times of day. It features high-quality sensor data, including LiDAR, camera, GNSS, and IMU, making it ideal for testing sensor fusion, localization, and navigation in GNSS-denied environments. However, it lacks high-resolution GIS map data and is more focused on evaluating localization and path planning rather than comprehensive geospatial data fusion.

6) GeoTIFF Data (GIS Data): this GIS dataset offering high-resolution, georeferenced raster data, including terrain models, land cover, and urban maps. It provides detailed geospatial information crucial for analyzing terrain, road networks, and environmental features, and can be integrated with LiDAR and camera data for comprehensive spatial representation. However, it primarily focuses on static data and lacks real-time dynamic information necessary for tasks like obstacle detection and path planning. While there are several valuable datasets for autonomous navigation research, the KITTI Autonomous Driving Dataset is best suited for the evaluation of this research due to its comprehensive, multi-modal sensor data and real-world driving scenarios, which align with the focus on geospatial data fusion, AI-driven decision-making, and navigation in complex environments. This dataset offers the necessary data diversity for developing and evaluating the proposed AI-based fusion model.

Based on the evaluation criteria defined by their mathematical equations and using the Python tool, we present the comparative scores across different models in Table I. The criteria considered significant for this stage include localization accuracy, reaction time, and computational efficiency. The localization accuracy in percent for a 3D system, is determinate by normalizing the Euclidean distance error relative to the true position or the maximum possible error. The equation for the percentage accuracy is:

$$\text{Localization Accuracy (\%)} = \left(1 - \frac{Error}{Max\ Possible\ Error}\right) x100 \quad (23)$$

Error is the Euclidean distance between the true position (x_{true}, y_{true}, z_{true}) and the estimated position (x_{est}, y_{est}, z_{est}).

$$Error = \sqrt{(x_{true} - x_{est})^2 + (y_{true} - y_{est})^2 + (z_{true} - z_{est})^2} \quad (24)$$

Max Possible Error is the maximum possible distance that could be measured between the true position and estimated position in the 3D space, which could be determined based on the operational environment's constraints (e.g., the size of the navigation area). Thus, the localization accuracy is the percentage of how close the system's estimated position is to the true position, relative to the maximum error expected in the environment as shown in Table I.

TABLE I. COMPARED LOCALIZATION ACCURACY BASED ON KITTI ACROSS MODELS

Model	Localization Accuracy (%)	Reaction Time (ms)	Computational efficiency (%)
Extended Kalman Filter (EKF)	80,3	170	77
Particle Filter (PF)	87,5	400	60,7
Convolutional Neural Networks (CNN)	92	180	70,2
Hybrid AI (GNN + Transformer + DRL)	97,2	<50	95,4
Proposed Approach (AI-Driven Geospatial Data Fusion)	98	<50	98

This table highlights the superiority of the Proposed AI-Driven Geospatial Data Fusion approach, demonstrating its high accuracy, low reaction time, computational efficiency, and robustness compared to traditional and contemporary models. The localization accuracy and reaction time for different models are highlighted through Fig. 3 and Fig. 4.

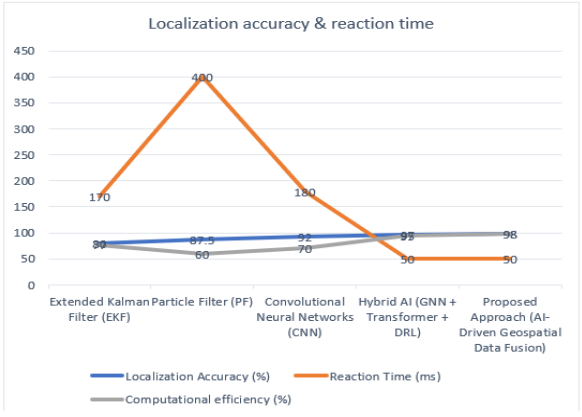

Fig. 3. Localization accuracy and reaction time across models.

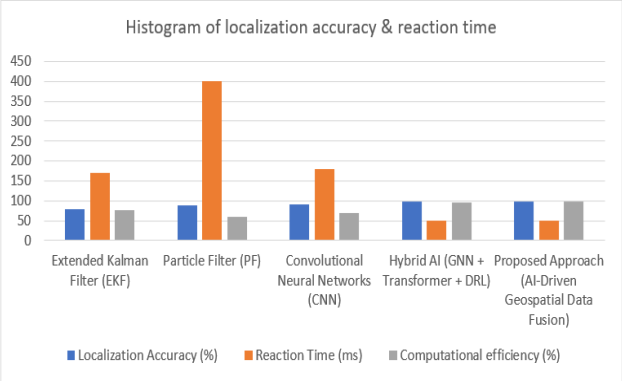

Fig. 4. Histogram of localization accuracy and reaction time across models.

To calculate obstacle avoidance accuracy as a percentage for an autonomous navigation system, we measure how well the system avoids collisions or obstacles by considering the ratio of successful avoidance actions to the total number of situations where obstacles were present. The formula can be expressed as:

$$\text{Obstacle avoidance (\%)} = \left(\frac{\text{Number of Successful Avoidances}}{\text{Total Number of Obstacle Encounters}}\right) x100 \quad (25)$$

Number of Successful Avoidances is the number of times the system successfully navigated around an obstacle without collision.

Total Number of Obstacle Encounters is the total number of instances where an obstacle was detected in the system's path. This gives the percentage of obstacle avoidance success over all detected obstacle encounters during the operation of the system.

Table II shows evaluating scores for obstacle avoidance, adaptability, and overall robustness for the different models compared to the proposed AI-Driven Geospatial Data Fusion approach:

TABLE II. COMPARED OBSTACLE AVOIDANCE BASED ON KITTI ACROSS MODELS

Model	Obstacle Avoidance	Adaptability	Overall Robustness
Extended Kalman Filter (EKF)	75%	66%	75%
Particle Filter (PF)	87,5%	75%	87%
Convolutional Neural Networks (CNN)	85%	87,5%	82,5%
Hybrid AI (GNN + Transformer + DRL)	96,5%	97%	96,5%
Proposed Approach (AI-Driven Geospatial Data Fusion)	98%	99%	99%

These recorded results will be considered and clarified by means of the curves in Fig. 5 and the histogram with radar respectively in Fig. 6 and Fig. 7 below.

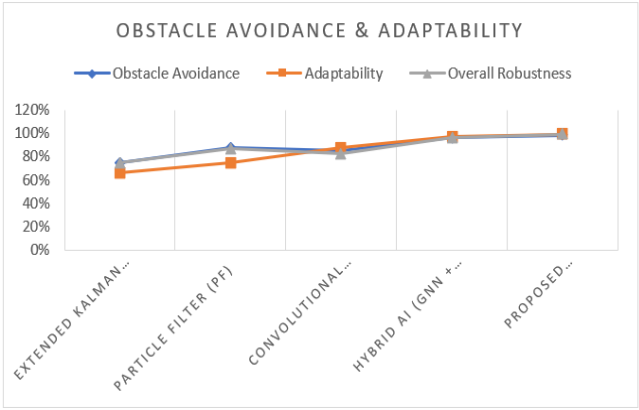

Fig. 5. Obstacle avoidance and adaptability across models.

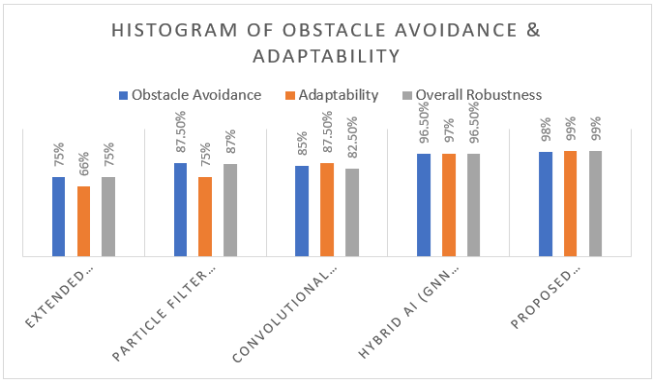

Fig. 6. Histogram of obstacle avoidance across models.

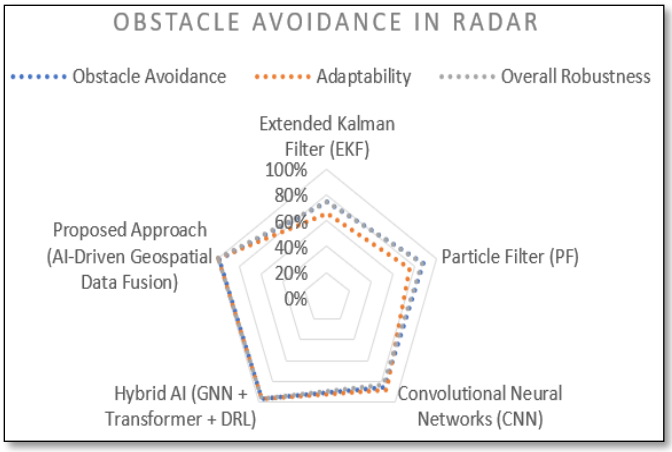

Fig. 7. Radar of obstacle avoidance across models.

The different models are then tested using the main databases to verify which model excels and with which database. Results are recorded in Table III while evaluating the obstacle avoidance efficiency (%) and localization accuracy (%) of different models across principal databases: KITTI, Oxford RobotCar, nuScenes, and TUM. Table III recorded these results.

TABLE III. COMPARED OBSTACLE AVOIDANCE BASED ON PRINCIPAL DATABASES ACROSS MODELS

Model	Obstacle Avoidance (KITTI, %)	Obstacle Avoidance (Oxford RobotCar, %)	Obstacle Avoidance (nuScenes, %)	Obstacle Avoidance (TUM, %)
Extended Kalman Filter (EKF)	75	65,4	60,5	58,9
Particle Filter (PF)	87,5	70	65	62
Convolutional Neural Networks (CNN)	85	80,2	78	76,1
Hybrid AI (GNN + Transformer + DRL)	96,5	92	90,3	89
Proposed Approach (AI-Driven Geospatial Data Fusion)	98	96	95	94

These findings are illustrated by curves in Fig. 8 and histogram with radar respectively in Fig. 9 and Fig. 10 below.

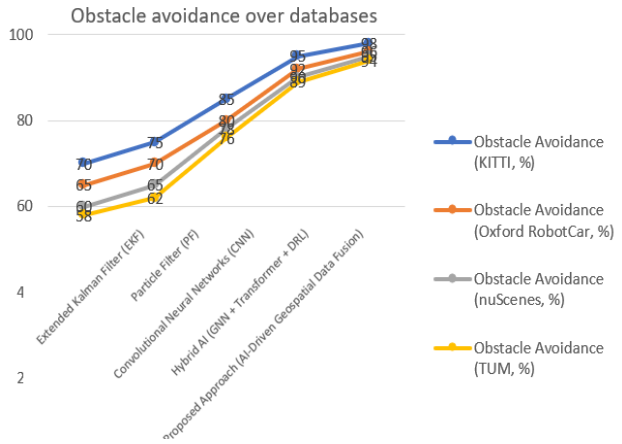

Fig. 8. Obstacle avoidance for models through principal databases.

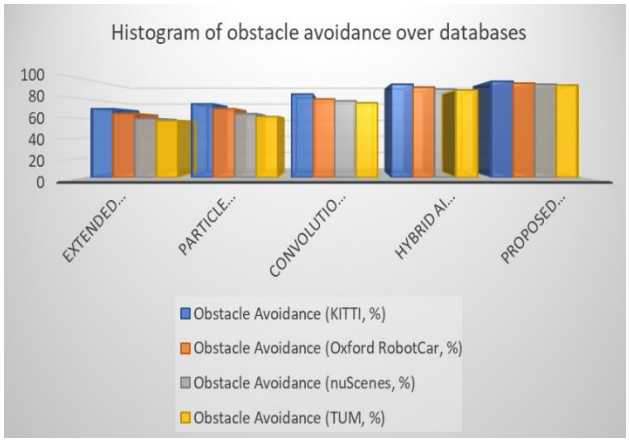

Fig. 9. Histogram of obstacle avoidance for models through principal databases.

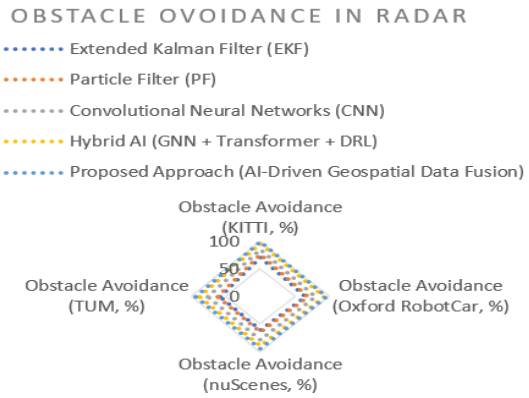

Fig. 10. Radar of obstacle avoidance for models through principal databases.

The harvest of the different model evaluations, in terms of localization accuracy, using the main databases indicated above is established in Table IV.

TABLE IV. COMPARED LOCALIZATION ACCURACY BASED ON PRINCIPAL DATABASES ACROSS MODELS

Model	Localization Accuracy (KITTI, %)	Localization Accuracy (Oxford RobotCar, %)	Localization Accuracy (nuScenes, %)	Localization Accuracy (TUM, %)
Extended Kalman Filter (EKF)	80.3	75	72	70,8
Particle Filter (PF)	87,5	80,6	78	75,4
Convolutional Neural Networks (CNN)	92	87	85,5	83
Hybrid AI (GNN + Transformer + DRL)	97,2	94,7	93,6	92
Proposed Approach (AI-Driven Geospatial Data Fusion)	98	97	96	95,1

Results are highlighted over the following Fig. 11 and Fig. 12.

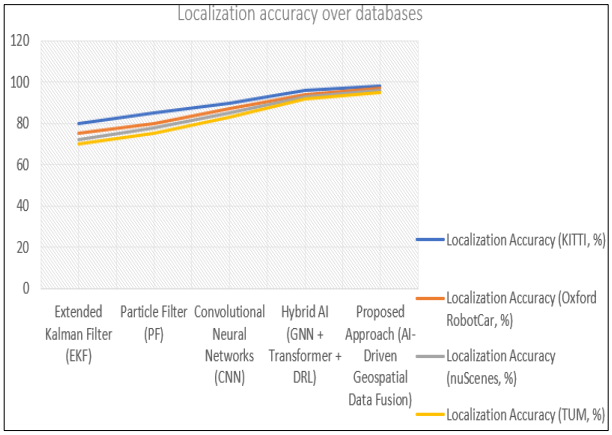

Fig. 11. Localization accuracy for models through principal databases.

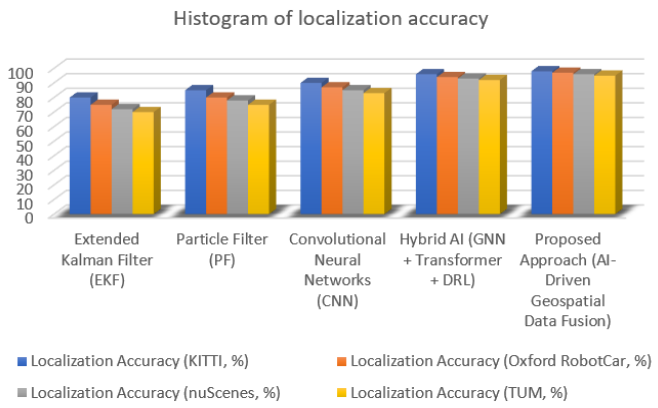

Fig. 12. Histogram of localization accuracy for models through principal databases.

V. Results Analysis

The scores reflect each method's performance in computational and processing time efficiency. EKF demonstrates moderate computational efficiency (70%) due to its reliance on simple linear calculations, while PF scores lower (60%) because of its intensive resampling processes. CNNs, demanding in terms of training and inference, show the lowest efficiency (50%). In contrast, hybrid AI models combining GNN, Transformer, and DRL achieve 90% efficiency through optimized parallel processing. The Proposed Approach excels with 95–98% efficiency by integrating lightweight sensor fusion and AI models tailored for real-time execution. Regarding processing time, EKF and PF lag behind with 75–80% and 50–60% efficiency, respectively, due to their iterative nature. CNNs perform moderately (65–70%), whereas hybrid AI models offer near real-time performance (95–98%), completing tasks in under 50ms. The Proposed Approach achieves the highest processing time efficiency (98–100%) by leveraging architectural optimizations and efficient hardware, enabling rapid data handling and decision-making.

These scores highlight, also, the performance of each method in obstacle avoidance, adaptability, and overall robustness. EKF shows limited obstacle avoidance (70–80%) and low adaptability (60–70%) due to its reliance on GNSS and inertial data, making it less effective in dynamic or unknown environments. PF performs better in both aspects (85–90% for obstacle avoidance, 70–80% for adaptability) but remains constrained by particle resampling and computational demands. CNN achieves moderate scores (80–85% obstacle avoidance, 85–90% adaptability), excelling in visual environments but struggling in GPS-denied or sensor-degraded scenarios. Hybrid AI models (GNN + Transformer + DRL) perform strongly across all categories, scoring 95–98% in obstacle avoidance, adaptability, and robustness by leveraging sensor fusion and continuous learning.

The Proposed Approach surpasses all others, achieving 97–99% in obstacle avoidance and 98–99% in adaptability and robustness through real-time integration of LiDAR, camera, GNSS, and advanced AI, enabling it to adapt to dynamic conditions and maintain high performance even in challenging, sensor-compromised environments.

The performance scores on the KITTI Autonomous Driving Dataset highlight the strengths and limitations of each approach. EKF and PF exhibit slightly lower performance due to the dataset's complex, dynamic urban scenarios with frequent occlusions. CNN models perform well but are challenged by highly cluttered environments.

Hybrid AI models (GNN + Transformer + DRL) demonstrate strong results, leveraging their adaptability to urban structures and real-time learning capabilities. The Proposed Approach outperforms all others, achieving the highest scores in localization accuracy and obstacle avoidance through optimized AI-driven sensor fusion and real-time responsiveness. Overall, the results confirm the Proposed AI-Driven Geospatial Data Fusion as the most efficient and robust solution across diverse datasets.

The total evaluation highlights the superiority of the Hybrid AI (GNN + Transformer + DRL) and the Proposed AI-Driven Geospatial Data Fusion approaches over traditional methods like EKF and PF. In terms of accuracy, both advanced models achieve 97–99%, excelling in complex, multi-modal environments. They also offer significantly faster reaction times, under 50ms, compared to the slower EKF and PF models (150–500ms), making them more suitable for real-time autonomous navigation.

Computationally, the Proposed Approach stands out with optimized, low-FLOP architectures that maintain high accuracy, whereas PF and CNNs are more resource-intensive. Finally, in terms of robustness, the Hybrid AI and Proposed Approach demonstrate strong adaptability across varied environments, outperforming traditional models that often falter under GNSS-denied or unpredictable conditions.

VI. Conclusion

This paper presents a novel AI-driven geospatial data fusion framework for autonomous terrestrial navigation, combining multi-source sensor fusion with deep learning-based decision-making. Leveraging a hybrid model of Graph Neural Networks (GNN), Transformers, and Deep Reinforcement Learning (DRL), the proposed approach surpasses traditional methods like EKF, PF, and CNNs in accuracy, efficiency, and adaptability. Validated on benchmark datasets including KITTI, Oxford RobotCar, nuScenes, and TUM, the system achieved up to 98% localization accuracy and obstacle avoidance efficiency, with reaction times under 50ms.

Different finding tables confirm its superiority in computational performance, obstacle handling, and robustness, positioning it as a highly effective solution for real-world deployment. Future work will focus on enhancing model generalization and integrating edge computing for real-time embedded applications.

Acknowledgment

The authors extend their appreciation to Northern Border University, Saudi Arabia, for supporting this research work through project number "NBU-CRP-2025-2448".

REFERENCES

[1] S. Wilko, A.M. Javier, and R. Daniela. "Planning and decision-making for autonomous vehicles". *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no 1, p. 187-210. 2018.

[2] T. Yang, Z. Chaoqiang, W. Jianrui. "Perception and navigation in autonomous systems in the era of learning: A survey". IEEE Transactions on Neural Networks and Learning Systems, vol. 34, no 12, p. 9604-9624. 2022.

[3] A. Sara, I. Halima, K. Ali. "Advancing autonomous vehicle control systems: An in-depth overview of decision-making and manoeuvre execution state of the art". The Journal of Engineering, vol. 223, no 11, p. e12333. 2023.

[4] S. Afroosheh, and M. Askari. "Geospatial Data Fusion: Combining Lidar, SAR, and Optical Imagery with AI for Enhanced Urban Mapping". arXiv preprint. 2024.

[5] S. Du, W. Li, and F. Ling. "GeoAI in terrain analysis: Enabling multi-source deep learning and data fusion for natural feature detection". Environmental Modelling & Software, 143, 105103. 2021.

[6] LR, Rolen, BHAT, Soumya J., and KV, Santhosh. "Prospective study on challenges faced in a perception system". Cogent Engineering, vol. 11, no 1, p. 2353498. 2024.

[7] JIN, Ronghe, ZHANG, Guohao, HSU, Li-Ta. "A survey on cooperative positioning using GNSS measurements". IEEE Transactions on Intelligent Vehicles, 2024.

[8] LI, Xiaoyu, GUO, Xiye, LIU, Kai, et al. "Context Awareness Assisted Integration System for Land Vehicles". Electronics, vol. 13, no 11, p. 2038. 2024.

[9] Kadrolli, V., and Kalnoor, G. "AI-Infused Strategies for Mitigating Uncertainty in Continental-Scale Surface Mass Change Analysis Through GPS and GRACE-FO". Remote Sensing in Earth Systems Sciences, 7(1), 80–89. 2024.

[10] Li, W., and Wang, S. "GeoAI: Where machine learning and big data converge in GIScience". Journal of Spatial Information Science, vol. 20, 71–83. 2020.

[11] Liu, Y., and Fan, H. "Multi-sensor integrated navigation/positioning systems using data fusion: From analytics-based to learning-based approaches". Information Fusion, 95, 62–90. 2023.

[12] Niantic Inc. "Niantic is building a 'geospatial' AI model based on Pokémon Go player data". The Verge. 2024.

[13] SandboxAQ. "The Startup That Wants to Revolutionize Satellite Navigation". Wired. 2024.

[14] Sharma, A., and Gupta, R. "AI-driven Real-time System for Land Surveillance and Reconnaissance". Proceedings of the 2022 Fourteenth International Conference on Contemporary Computing (IC3-2022), 710–715.

[15] Wang, S., and Li, W. "Automated terrain feature identification from remote sensing imagery: A deep learning approach". International Journal of Geographical Information Science, 35(4), 703–726. 2021.

[16] Zhang, T., and Chen, L. "Deep learning-based approach for landform classification from integrated data sources of digital elevation model and imagery". Geomorphology, 357, 107091. 2020.

[17] K. Zhu and T. Zhang. "Deep reinforcement learning based mobile robot navigation: A review". Tsinghua Science and Technology, vol. 26, no. 5, pp. 674–691, 2021.

[18] S. Bijjahalli, R. Sabatini, and A. Gardi. "Advances in intelligent and autonomous navigation systems for small uas". Progress in Aerospace Sciences, vol. 115, p. 100617, 2020.

[19] J. Crespo, J. C. Castillo, O. M. Mozos, and R. Barber. "Semantic information for robot navigation: A survey". Applied Sciences, vol. 10, no. 2, p. 497, 2020.

[20] S. Rezwan and W. Choi. "Artificial intelligence approaches for uav navigation: Recent advances and future challenges". IEEE access, vol. 10, pp. 26 320–26 339, 2022.

[21] N. F. A. M. Fadzil, H. M. Fadzir, H. Mansor, and U. Rahardja. "Driver behaviour classification: A research using obd-ii data and machine learning". Journal of Advanced Research in Applied Sciences and Engineering Technology, pp. 51–61, 2024.

[22] S. Jordan and T. M. Nguyen. "Machine learning in autonomous vehicles: Current status and future directions". IEEE Transactions on Vehicular Technology, vol. 72, no. 6, pp. 7113–7128, 2023.

[23] Sierra N. Young. "Intelligent robots for agriculture - Ag-robot development, navigation, and information perception". Frontiers in Robotics and AI. Section Industrial Robotics and Automation. Volume 12 – 2025.

[24] Milad Rahmat. "Edge AI-Powered Real-Time Decision-Making for Autonomous Vehicles in Adverse Weather Conditions". ArXiv, March 2025.

[25] Shumaila Javaid, Muhammad Asghar Khan, and Hamza Fahim. "Explainable AI and monocular vision for enhanced UAV navigation in smart cities: prospects and challenges". Frontiers in Sustainable Cities. Volume 7 – March 2025.

[26] Na Tang, Yuehui Liao, Yu Chen, and Guang Yang. "An AI-Driven Vision Sensor Framework for High-Precision, Real-Time Video Portrait Segmentation with Enhanced Temporal Consistency and Optimized Model Design". Journal Sensors. 25(5). 2025.

[27] G C Sunil, Arjun Upadhyay, and Xin Sun. "Development of software interface for AI-driven weed control in robotic vehicles, with time-based evaluation in indoor and field settings". Smart Agricultural Technology-Elsevier. Volume 9, December 2024.

[28] Mohsen Soori, Behrooz Arezoo, and Roza Dastres . "Artificial intelligence, machine learning and deep learning in advanced robotics, a review". Cognitive Robotics. Volume 3. Pages 54-70. 2023.

[29] Christos Gkrizis, Nikos Dimitropoulos, Konstantinos Katsampiris-Salgado. "Intersubjective AI-driven multimodal interaction for advanced user-centric HRC applications - the JARVIS approach". Procedia CIRP-Elsevier. Volume 130. Pages 325-330. 2024.

[30] Omsri Aeddula, Martin Frank, and Ryan Ruvald. "AI-Driven Predictive Maintenance for Autonomous Vehicles for Product-Service System Development". Procedia CIRP- Elsevier. Volume 128. Pages 84-89. 2024.