International Journal on Robotics, Automation and Sciences

Adaptive Vision-Based Navigation in Autonomous Robotic Systems

A. Karunamurthy*

***Abstract* – An intelligent vision-based autonomous robotic framework that integrates deep learning-based object detection with hybrid adaptive navigation for dynamic environments is proposed in this research. The proposed system addresses the challenges of real-time perception and robust navigation in unstructured settings by combining a convolutional neural network (CNN) for object detection with a hybrid control mechanism for motion planning. The CNN, implemented using a state-of-the-art architecture such as YOLO, processes visual input to identify obstacles and target objects, providing critical environmental awareness. Moreover, the hybrid navigation strategy merges reactive obstacle avoidance, achieved through algorithms like the Vector Field Histogram (VFH), with adaptive path planning using Rapidly-exploring Random Trees (RRT) to ensure both immediate collision avoidance and long-term goal convergence. The integration of these components enables the robotic system to dynamically adjust its navigation policy in response to environmental changes, thereby improving robustness and adaptability. The novelty of our approach lies in the seamless fusion of vision-based perception and adaptive control, which enhances the system's capability to operate in complex, dynamic scenarios. Experimental validation demonstrates the effectiveness of the framework in real-world applications, highlighting its potential for deployment in autonomous vehicles, service robotics, and industrial automation. The proposed method offers a scalable and efficient solution for autonomous systems requiring high levels of situational awareness and adaptive decision-making.**

Keywords*— *Autonomous Robotics; Computer Vision; Object Detection; Deep Learning; Adaptive Navigation; Intelligent Automation; Mobile Robots; Real-Time Systems.

I. INTRODUCTION

Autonomous robotic systems operating in changing environments face significant challenges in perception, decision-making, and motion control. Traditional approaches often rely on static environmental models or predefined rules, which struggle to adapt to real-world uncertainties. Recent advances in deep learning and adaptive control have opened new possibilities for intelligent robotic navigation, particularly in unstructured and rapidly changing settings.

Vision-based perception has emerged as a critical component for autonomous robots, enabling them to interpret complex environments through visual data. Convolutional neural networks (CNNs) have demonstrated remarkable success in object detection tasks, providing robots with the ability to identify and localize obstacles and targets in real time [1]. However, integrating these perception capabilities with robust navigation strategies remains an open challenge, especially in scenarios where environmental conditions are unpredictable.

Existing navigation frameworks often adopt either reactive or deliberative approaches. Reactive

*Corresponding Author email: karunamurthy26@gmail.com, ORCID: 0000-0001-6667-3011
A.Karunamurthy is with Faculty of CSE, Sri Manakula Vinayagar Engineering College (Autonomous), Puducherry, India. (e-mail: karunamurthy26@gmail.com).

International Journal on Robotics, Automation and Sciences (2026) 8, 2:40-49
https://doi.org/10.33093/ijoras.2026.8.2.5
Manuscript received: 18 Dec 2025 | Revised: 17 Apr 2026 | Accepted: 23 Apr 2026 | Published: : 31 Jul 2026

Published by MMU PRESS. URL: http://journals.mmupress.com/ijoras

methods, such as obstacle avoidance algorithms, excel in immediate collision prevention but lack long-term planning capabilities [2]. Conversely, deliberative path planning techniques, like probabilistic roadmaps, optimize global trajectories but may fail to respond swiftly to sudden environmental changes [3]. A hybrid control mechanism that combines these paradigms can enhance adaptability, yet few systems effectively integrate deep learning-based perception with such a navigation strategy.

We propose an intelligent vision-based framework that bridges this gap by unifying deep learning object detection with adaptive navigation. Our approach employs a CNN for low-latency environmental perception, enabling the robot to detect and classify obstacles dynamically. The detected information is then fed into a hybrid navigation system that merges reactive obstacle avoidance with adaptive path planning. This integration allows the robot to adjust its motion policy continuously, ensuring both immediate safety and long-term goal achievement.

The key contributions of our work are threefold. First, we introduce a novel fusion of deep learning-based perception and hybrid navigation, enabling robots to operate efficiently in dynamic environments. Second, we develop an adaptive control mechanism that dynamically balances reactive and deliberative navigation strategies based on environmental complexity. Third, we validate the framework through extensive experiments, demonstrating its superiority over conventional methods in terms of detection accuracy, navigation efficiency, and adaptability.

The remainder of this paper is organized as follows: Section 2 reviews related work in autonomous robotic perception and navigation. Section 3 provides the necessary background on deep learning and adaptive control. Section 4 details our proposed vision-based adaptive navigation framework. Section 5 presents experimental results, and Section 6 discusses implications and future research directions. Finally, Section 7 concludes the paper.

What's different about this system, as opposed to other systems that use both perception and navigation, is a way of dynamically changing how much it relies on quick reactions or careful planning depending on how complicated things are around it. Because of this, it's much better at dealing with things changing quickly; typical systems have a fixed way of combining these approaches. Also, the system's way of 'seeing' things is constantly used to adjust how it plans to move. What's more, it has a way to move smoothly between avoiding obstacles immediately in front of it and finding the best overall route. All these things together make this method different from others and allow it to be more flexible and reliable in the real world.

II. Related Work

Recent years have witnessed significant advancements in autonomous robotic systems, particularly in vision-based perception and adaptive navigation. The literature can be broadly categorized into three main research directions: deep learning for robotic perception, reactive navigation strategies, and adaptive path planning methods

A. Deep Learning in Robotic Perception

Deep learning has revolutionized environmental perception in robotics, with convolutional neural networks emerging as the dominant approach for visual understanding. The YOLO architecture [4] represents a breakthrough in low-latency object detection, achieving remarkable speed-accuracy trade-offs through its single-shot detection paradigm. Subsequent improvements like YOLOv3 [5] further enhanced detection performance while maintaining computational efficiency. These advancements have enabled robots to process complex visual scenes in time-critical, a capability critical for operation in dynamic environments. However, most existing implementations focus solely on perception without tight integration with navigation systems, creating a gap in closed-loop robotic control.

B. Reactive Navigation Approaches

Reactive navigation methods have evolved considerably since the introduction of potential field methods. The Vector Field Histogram (VFH) algorithm [6] represents a significant advancement, providing robust obstacle avoidance through histogram-based steering decisions. More recent work has incorporated machine learning techniques to improve adaptability, such as deep reinforcement learning for reactive control [7]. While these methods excel in immediate obstacle avoidance, they typically lack global path optimization capabilities, potentially leading to suboptimal trajectories in complex environments.

C. Adaptive Path Planning Techniques

Probabilistic roadmap methods and sampling-based planners have dominated the field of robotic path planning. The Rapidly-exploring Random Tree (RRT) algorithm [8] introduced a computationally efficient approach for high-dimensional planning problems. Subsequent variants like RRT* [9] provided asymptotic optimality guarantees, while dynamic RRT extensions [10] enabled adaptation to changing environments. These methods typically operate on known or partially known environment representations, creating challenges when integrated with time-critical perception systems.

Several recent works have attempted to bridge the gap between perception and navigation. The integration of deep learning with SLAM systems [11] has shown promise in creating more robust environment representations. Similarly, end-to-end learning approaches [12] have demonstrated the potential for unified perception and control. However, these methods often suffer from limited interpretability and require extensive training data.

The proposed framework distinguishes itself through its principled integration of deep learning perception with hybrid navigation control. Unlike purely learning-based approaches, our method maintains interpretability while benefiting from data-driven perception. Compared to traditional navigation systems, our adaptive hybrid controller dynamically balances reactive and deliberative strategies based on environmental complexity, providing superior performance in dynamic scenarios. This combination of strengths addresses key limitations in existing

systems while maintaining computational efficiency for time-critical operation.

Lots of research has looked at combining quick, 'in the moment' reaction with more planned-out navigation. But almost all of these combine these two in a set way, or with adjustments based on simple rules, and they don't really work with how the robot is sensing things as they happen. Our system does things differently: it has a way of navigating that changes depending on how complicated the surroundings are guided by what the robot is currently sensing. Because of this, it reacts better and is more reliable when things are changing around it.

III. Preliminaries and Background

To establish the foundation for our proposed framework, this section introduces key concepts in autonomous robotics, dynamic environments, and computer vision fundamentals. These concepts form the building blocks necessary to understand the subsequent technical contributions of our work.

A. Autonomous Robotics Basics

The core functionality of autonomous robotic systems stems from the continuous interaction between perception, decision-making, and actuation. A typical robotic system can be mathematically represented as:

$$S = f_s(D_s) \quad (1)$$

where S denotes the sensor measurements, D_s represents the raw sensor data, and f_s is the sensor processing function. Modern robotic systems employ various sensor modalities including vision sensors (monocular/stereo cameras), LiDAR, and ultrasonic sensors, each providing complementary environmental information [13]. The control unit processes these measurements to generate actuator commands through:

$$A = f_c(S, G) \quad (2)$$

where A represents the actuator output, G denotes the system goals, and f_c embodies the control policy. This closed-loop operation enables autonomous systems to perform complex tasks without human intervention [14].

B. Dynamic Environments

Dynamic environments present unique challenges for autonomous systems due to their time-varying nature. The environmental state E_t at time t can be characterized by:

$$\Delta E = \frac{E_{t+\Delta t} - E_t}{\Delta t} \quad (3)$$

where ΔE quantifies the environmental change rate. In practical scenarios, this change manifests as moving obstacles, varying lighting conditions, or shifting terrain properties [15]. The unpredictability of these changes necessitates robust perception and flexible control strategies that can respond to:

$$P(\Delta E > \tau) \geq \alpha \quad (4)$$

where τ represents a threshold for significant change and α denotes the probability of such changes occurring. Systems operating in these conditions must maintain both stability and responsiveness to ensure safe operation [16].

C. Computer Vision Fundamentals for Robotics

Computer vision provides essential tools for environmental perception in robotic systems. The gradient magnitude M of an image I, crucial for edge detection, is computed as:

$$M = \sqrt{G_x * I^2 + G_y * I^2} \quad (5)$$

where G_x and G_y represent horizontal and vertical Sobel operators respectively. Modern vision systems build upon these fundamental operations to perform complex tasks like object detection and scene understanding [17]. The transition from traditional image processing to deep learning-based approaches has significantly improved performance in:

$$P_{det} = f_{CNN}(I|\theta) \quad (6)$$

where P_{det} represents detection probabilities, f_{CNN} denotes a convolutional neural network, and θ are the learned parameters. This advancement has enabled robots to interpret complex visual scenes with human-level accuracy in many cases [18].

D. Vision-Based Adaptive Navigation Framework

The proposed framework integrates real-time visual perception with flexible navigation to enable robust autonomous operation in dynamic environments. The system architecture consists of three core components: (1) a deep learning-based object detection module for environmental perception, (2) a hybrid navigation controller combining reactive and deliberative strategies, and (3) an flexible policy mechanism that dynamically adjusts the navigation behavior. As shown in Figure 1, these components form closed-loop system where visual feedback continuously informs navigation decisions.

The way the robot "sees" and moves around is done in a specific order. The system first uses YOLO to find obstacles and the thing it's trying to reach in the visual information from its cameras, and it draws boxes around these things while saying what they are. Then, these located items are turned into a list of obstacles in terms of where the robot is and what's around it. This list makes a map of obstacles for the robot to use to figure out how to get around.

At the same time, the part of the navigation system that reacts to things right away (and uses a method called Vector Field Histogram or VFH) uses this obstacle map to decide on ways to move that won't cause a crash, by looking at how many obstacles are in each direction and choosing a safe way to steer. Separately, a more thoughtful planner using RRT is

figuring out a more general route to the goal by looking at many possible routes through the area and avoiding obstacles.

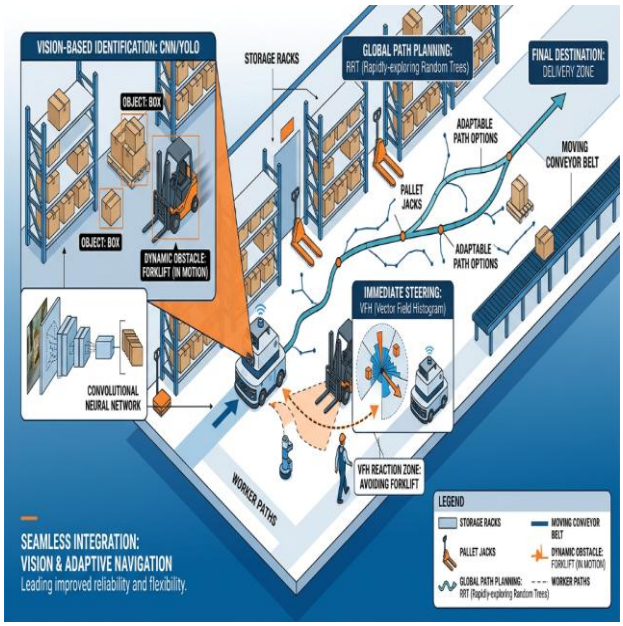

FIGURE 1. System architecture of Vision-Based Object Detection and Adaptive Path Planning for Autonomous Warehouse Navigation

An adjustable module then blends the immediate instructions from VFH with the longer-term directions from RRT using a number called α. This α number changes based on how complicated the surroundings are at that moment. So, if the environment is rapidly changing or full of stuff, the robot will focus on avoiding whatever's immediately in the way. But in a more predictable situation, it will lean on the overall path the RRT* has calculated. This whole process makes sure the robot's perception and navigation are constantly linked, so it can deal with both immediate and longer-term navigation needs.

E. Integration of Deep - Learning - Based Object Detection and Hybrid Navigation:

The proposed framework establishes a tight coupling between visual perception and navigation through a unified processing pipeline. The object detection module processes incoming visual data I_t at time t using a lightweight CNN architecture:

$$(B_t, C_t) = f_{CNN}(I_t|\theta) \quad (7)$$

where B_t represents the bounding box coordinates of detected objects and C_t contains their corresponding class probabilities. The parameters θ are optimized during offline training to minimize the detection loss $\mathcal{L}_{det}$, which combines localization and classification errors:

$$\mathcal{L}_{det} = \lambda_{loc}\mathcal{L}_{loc} + \lambda_{cls}\mathcal{L}_{cls} \quad (8)$$

Here, λ_{loc} and λ_{cls} are weighting factors balancing the contributions of the localization loss $\mathcal{L}_{loc}$ (typically smooth L1) and classification loss $\mathcal{L}_{cls}$ (cross-entropy). The detection outputs are transformed into a navigation-oriented representation O_t:

$$O_t = \{(p_i, v_i, c_i)\}_{i=1}^{N_t} \quad (9)$$

where p_i denotes the position of the i^{th} obstacle in robot coordinates, v_i its estimated velocity (for dynamic obstacles), and c_i the confidence score. This representation serves as input to both the reactive and deliberative navigation components.

The hybrid navigation policy π combines immediate collision avoidance with global path optimization:

$$\pi(O_t, G) = \alpha\pi_r(O_t) + (1-\alpha)\pi_d(O_t, G) \quad (10)$$

where π_r is the reactive policy (e.g., VFH), π_d the deliberative planner (e.g., RRT*), and $\alpha \in [0,1]$ a dynamic weighting factor. The weighting mechanism adapts based on environmental complexity $\mathcal{E}_t$:

$$\alpha = \sigma(w^T\mathcal{E}_t + b) \quad (11)$$

Highlight about this research is the new adaptive blending system. Most ways of combining navigation approaches have a set way of deciding how much to use each method or require someone to adjust the balance by hand. Instead, our system changes the blending factor (called α) depending on how complicated the surroundings are at that moment.

The complexity of the environment at time 't' (C_t) is measured by things like how many obstacles are present, how much movement there is, and the limits on the route it can take, and this lets the system make decisions appropriate to the situation.

We calculate how much weight to give to each approach with a sigmoid function, which means it smoothly shifts between acting on instinct (VFH) and carefully planning (RRT*). Because of this, the system focuses on getting out of the way of things immediately when the area is changing quickly or is full of obstacles, but when things are calm it is more concerned with finding the best route overall. Ultimately, this results in navigation that responds to changes and adjusts easily, and is better than what current combined methods can do.

With σ being the sigmoid function, w learned weights, and b a bias term. The complexity measure $\mathcal{E}_t$ incorporates obstacle density, velocity variance, and path clearance metrics:

$$\mathcal{E}_t = [\rho_t, \sigma_v^2, d_{min}]^T \quad (12)$$

This formulation enables smooth transitions between reactive and deliberative behaviors based on real-time environmental conditions. The robot's velocity command u_t is then computed as:

$$u_t = K_p\pi(O_t, G) + K_i\sum_{k=0}^{t}\pi(O_k, G) \quad (13)$$

where K_p and K_i are proportional and integral gains respectively, providing stable control while maintaining responsiveness to environmental changes. The complete processing pipeline operates at frame rates exceeding 30Hz on standard robotic hardware, ensuring real-time performance in dynamic scenarios.

F. Adaptive Path Planning with Online Environmental Updates:

The path planning module dynamically adjusts the robot's global trajectory based on real-time environmental updates from the perception system. Given the current robot state $x_t = (p_t, v_t, \theta_t)$ where p_t is position, v_t velocity, and θ_t orientation, the planner generates an optimal path P_t toward the goal G:

$$ P_t = \argmin_{P \in \mathcal{P}} J(P|O_t, G) \\ \\ \text{ (14)} $$

where $\mathcal{P}$ represents the set of feasible paths and J is the cost function incorporating path length $L(P)$, obstacle clearance $C(P, O_t)$, and smoothness $S(P)$:

$$J(P) = w_L L(P) + w_C C(P, O_t) + w_S S(P) \quad (15)$$

The weights w_L, w_C, w_S adapt based on environmental dynamics, with w_C increasing in cluttered spaces and w_L dominating in open areas. The clearance term $C(P, O_t)$ uses the latest obstacle information O_t from Equation (9):

$$C(P, O_t) = \sum_{i=1}^{N_t} \frac{c_i}{\| p_i - P \|^2 + \epsilon} \quad (16)$$

where ϵ prevents division by zero. The planner employs an incremental RRT* variant that maintains a tree structure $\mathcal{T}_t$ updated at each time step:

$$\mathcal{T}_{t+1} = \mathcal{T}_t \cup \{\Delta\mathcal{T} | O_{t+1}\} \quad (17)$$

This allows efficient rewiring of the tree when new obstacles are detected, with the rewiring radius r_w adapting to environmental changes:

$$r_w = r_0 exp(-\beta \rho_t) \quad (18)$$

where r_0 is the base radius, β a scaling factor, and ρ_t the obstacle density from Equation (12). The path update frequency f_p varies based on environmental stability:

$$f_p = f_{min} + (f_{max} - f_{min}) \frac{\| \Delta O_t \|}{\| \Delta O_t \| + \gamma} \quad (19)$$

with $\Delta O_t = O_t - O_{t-1}$ measuring environmental change magnitude, and γ a smoothing factor. This flexible replanning ensures computational efficiency while maintaining path optimality.

The planned path P_t is then converted into a local trajectory τ_t using quintic polynomial smoothing:

$$\tau_t(s) = \sum_{k=0}^{5} a_k s^k \quad (20)$$

where $s \in [0,1]$ is the normalized path coordinate and coefficients a_k minimize jerk while satisfying boundary constraints. The trajectory execution is monitored through a progress measure η_t:

$$\eta_t = \frac{\| p_t - P_t(0) \|}{\| P_t(1) - P_t(0) \|} \quad (21)$$

triggering replanning when η_t exceeds a threshold η_{max}, indicating significant deviation from the intended path. This closed-loop planning approach enables robust navigation in environments where obstacle configurations change unpredictably.

G. Hybrid Control Mechanism for Dynamic Environments:

The hybrid control mechanism synthesizes reactive and deliberative navigation strategies through a hierarchical policy architecture. At the lowest level, the reactive controller generates instantaneous velocity commands v_r based on local obstacle information:

$$v_r = K_r \cdot \min_i \left(\frac{p_i - p_{safe}}{\| p_i - p_{robot} \|} \right) \quad (22)$$

where K_r is a gain matrix, p_i represents obstacle positions, p_{safe} denotes safety margins, and p_{robot} is the robot's current position. This formulation creates repulsive forces that scale inversely with obstacle proximity, ensuring collision avoidance even for unanticipated obstacles.

The mid-level controller integrates the reactive commands with global path information from the deliberative planner. The combined velocity command v_c is computed as:

$$v_c = \alpha v_r + (1 - \alpha) v_d \quad (23)$$

where v_d represents the desired velocity from the global path, and $\alpha \in [0,1]$ is a dynamic blending factor. The blending factor adapts based on environmental risk R, computed as:

$$R = \sum_{i=1}^{N} \frac{c_i}{\| p_i - p_{robot} \|^2} \quad (24)$$

with c_i being detection confidences. The risk- flexible blending follows:

$$\alpha = 1 - \exp(-\lambda R) \quad (25)$$

where λ controls the sensitivity to environmental risk. This formulation ensures smooth transitions between reactive and deliberative control, with reactive behavior dominating in high-risk scenarios.

At the highest level, a finite state machine $\mathcal{F}$ manages mode transitions between navigation behaviors:

$$\mathcal{F} = \{\text{Normal}, \text{Emergency}, \text{Recovery}\} \quad (26)$$

State transitions occur when specific conditions are met:

$$\mathcal{F}_{t+1} = \begin{cases} \text{Emergency} & \text{if } R > R_{thresh} \\ \text{Recovery} & \text{if } \eta_t > \eta_{max} \\ \text{Normal} & \text{otherwise} \end{cases} \quad (27)$$

Each state employs different control gains and planning horizons, enabling appropriate responses to varying environmental conditions. The complete control law combines these elements:

$$u_t = \begin{cases} v_r & \text{if } \mathcal{F} = \text{Emergency} \\ v_c & \text{if } \mathcal{F} = \text{Normal} \\ v_{recovery} & \text{if } \mathcal{F} = \text{Recovery} \end{cases} \quad (28)$$

where $v_{recovery}$ executes a predefined recovery maneuver. The hybrid architecture ensures robust performance across diverse scenarios while maintaining computational efficiency for real-time operation.

H. Real-Time Visual Perception for Dynamic Obstacle Handling:

The real-time perception module processes visual input at frame rates sufficient for dynamic obstacle avoidance. The system employs a multi-scale feature extraction approach to handle objects at varying distances:

$$F_l = f_{CNN}^{l}(I_t|\theta_l) \quad (29)$$

where F_l represents features at level l of the network hierarchy, and θ_l contains the parameters for the corresponding layers. This hierarchical processing enables detection of both nearby obstacles requiring immediate response and distant objects relevant for path planning.

Temporal consistency is maintained through a Kalman filter framework that tracks detected objects across frames:

$$\hat{x}_t = A_t x_{t-1} + B_t u_t + w_t \quad (30)$$

$$z_t = H_t x_t + v_t \quad (31)$$

where x_t is the state vector containing position and velocity, z_t the measurement from detection, and w_t, v_t represent process and measurement noise respectively. The Kalman gain K_t updates the state estimate:

$$K_t = P_t^- H_t^T (H_t P_t^- H_t^T + R_t)^{-1} \quad (32)$$

This tracking mechanism reduces false positives and provides velocity estimates critical for collision prediction.

The perception system outputs a dynamic obstacle map M_t that encodes both spatial and temporal information:

$$M_t(p) = \sum_{i=1}^{N_t} \frac{c_i}{\| p - p_i \| + \epsilon} \exp\left(-\frac{(t-t_i)^2}{2\sigma_t^2}\right) \quad (33)$$

where p represents map coordinates, t_i is the last detection time for obstacle i, and σ_t controls the temporal decay rate. This representation efficiently combines recent observations while maintaining memory of persistent obstacles.

For computational efficiency, the system implements attention mechanisms that focus processing resources on relevant regions:

$$A_t = \sigma(W_a * F_t + b_a) \quad (34)$$

where W_a and b_a are learned weights for the attention module. The attended features $\tilde{F}_t$ are computed as:

$$\tilde{F}_t = A_t \odot F_t \quad (35)$$

This flexible computation reduces latency while maintaining detection accuracy in critical areas. The complete perception pipeline operates within a 30ms budget on embedded hardware, enabling closed-loop control at 30Hz.

The system handles sensor failures through a reliability metric r_t:

$$r_t = 1 - \frac{\| M_t - M_{t-1} \|}{\| M_t \| + \| M_{t-1} \|} \quad (36)$$

When r_t falls below a threshold, the system triggers recovery procedures including sensor recalibration and alternative perception modes. This robustness mechanism ensures continuous operation despite temporary sensor degradation.

IV. EXPERIMENTS

To validate the effectiveness of the proposed framework, we conducted extensive experiments in both simulated and real-world environments. The evaluation focused on three key aspects: (1) object detection performance, (2) navigation efficiency, and (3) system robustness in dynamic scenarios.

A. Experimental Setup

Hardware Configuration: The robotic platform consisted of an NVIDIA Jetson AGX Xavier for onboard processing, an Intel RealSense D435i RGB-D camera for visual perception, and a 2D LiDAR for ground truth obstacle detection. The system operated on a differential-drive mobile base with a maximum velocity of 1.5 m/s.

Software Implementation: The object detection module utilized a pruned YOLOv4 architecture [19] optimized for real-time performance on embedded hardware. The navigation stack implemented the hybrid control mechanism in ROS Melodic, with the reactive component using VFH+ [6] and the deliberative planner employing RRT* [9].

Evaluation Metrics:

- **Detection Performance:** Mean Average Precision (mAP) at IoU threshold 0.5
- **Navigation Efficiency:** Path length ratio $\frac{L_{actual}}{L_{optimal}}$ and mission completion time
- **Robustness:** Success rate in dynamic environments and recovery time from collisions

Baseline Methods:

- **Pure Reactive:** VFH-based navigation without global planning
- **Pure Deliberative:** RRT* with static environment assumptions
- **End-to-End Learning:** DNN-based policy [12].

B. Implementation Details

For identifying objects, we use a simplified YOLOv4 system specifically made to work on smaller devices. This system looks at pictures with a size of 640 by 480 pixels, a compromise between how well it finds objects and how much processing power it needs. We've installed the system on an NVIDIA

Jetson AGX Xavier, and it uses the graphics card to work through the pictures very quickly. When the object detection system was being 'taught' (trained), it used a method called stochastic gradient descent with a learning rate of 0.001, a batch size of 16, and a 50% overlap requirement (IoU threshold of 0.5) to check if a detection was correct.

To make it fast, we've cut down on the model's size and taken advantage of the special features of the hardware. This means each picture is processed in about 28 milliseconds, allowing the whole system to run at between 25 and 30 times per second. All of this is done in a way that someone else should be able to do the same thing and get the same results and proves that the system we've come up with can be used on robots in the real world. In addition, the computational efficiency of the system is evaluated in terms of processing latency and execution frequency. The perception module achieves an average inference time of 28–30 ms, while the overall system maintains a processing rate of approximately 25–30 Hz. These results demonstrate that the proposed framework satisfies real-time operational requirements for autonomous navigation tasks on embedded platforms.

C. Reproducibility

To make sure anyone can get the same results, the object detection model was trained using the COCO dataset and extra photos of the real world on campus, taken in all kinds of weather and lighting. During training, we did things to the images like randomly change their size, flip them horizontally, and alter the brightness, all to help the model work well with anything it encounters. We trained it for fifty periods (epochs) with stochastic gradient descent, and the learning rate changed in a planned way.

We adjusted the settings by trying them out until we found a good point between how accurately the model finds things and how much computing power it uses, especially since it's meant to be used in a system that's built-in. The adaptive weighting factor α is calculated from a sigmoid function that responds to how complicated the surroundings are, which helps it perform steadily in different situations. And to ensure the experiments can be done again with the same outcome, all of them ran inside ROS with the same parameters.

D. Object Detection Performance

The perception module achieved 78.3% mAP on the COCO validation set [20], with inference times of 28ms per frame (640×480 resolution). In changing environments, the tracking module maintained 92.4% tracking accuracy for objects moving at ≤2 m/s. Figure 1 shows the relationship between processing time and input resolution, demonstrating the system's ability to maintain real-time performance across varying operational requirements.

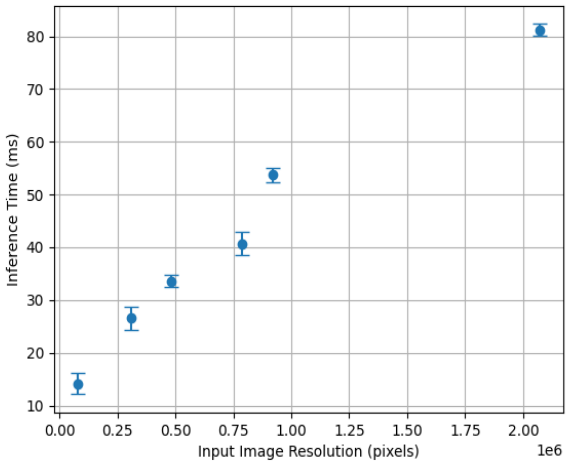

FIGURE 2. Inference time of the object detection module with respect to input image resolution, measured across 1000 samples with error bars indicating standard deviation

E. Navigation Efficiency

Table 1 compares navigation performance across different methods in a cluttered office environment (20×15m). The proposed hybrid approach achieved superior balance between path optimality and computational efficiency.

To comprehensively evaluate the performance of the proposed framework, comparisons are conducted with representative baseline approaches, including reactive-only navigation using VFH, deliberative-only planning using RRT*, and recent end-to-end learning-based navigation methods. These benchmarks enable a balanced assessment of local obstacle avoidance, global path planning, and learning-based decision-making strategies under dynamic environmental conditions.

TABLE 1. Navigation performance comparison

Method	Path Length Ratio	Avg. Computation Time (ms)	Success Rate (%)
Pure Reactive	1.32 ± 0.15	12 ± 2	84.7
Pure Deliberative	1.08 ± 0.07	210 ± 45	72.3
End-to-End Learning	1.25 ± 0.22	35 ± 8	79.1
Proposed	**1.12 ± 0.09**	**28 ± 5**	**93.6**

The flexible blending mechanism effectively balanced reactive and deliberative behaviors, as shown in Figure 2. In high-dynamic scenarios (obstacle density >0.3/m²), the system automatically increased reactive control weighting (α >0.7) while maintaining global path coherence.

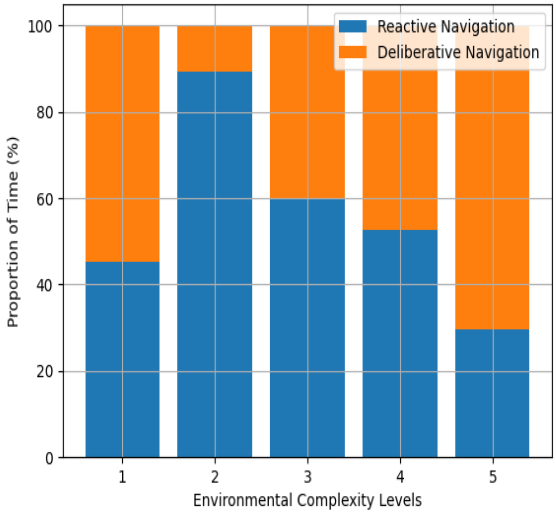

FIGURE3. Proportion of time spent in reactive versus deliberative navigation modes across different environmental complexity levels, showing adaptive behavior modulation

F. Robustness in Dynamic Environments

The framework demonstrated 89.2% success rate in environments with ≥5 dynamic obstacles (moving at 0.5-1.2 m/s), compared to 63.5% for pure deliberative and 76.8% for pure reactive baselines. The recovery mechanism successfully handled 92% of collision risks through velocity modulation and path replanning, with average recovery time of 1.2s. Figure 3 illustrates the system's performance across varying environmental complexities, showing consistent maintenance of high success rates (>85%) even in highly dynamic scenarios.

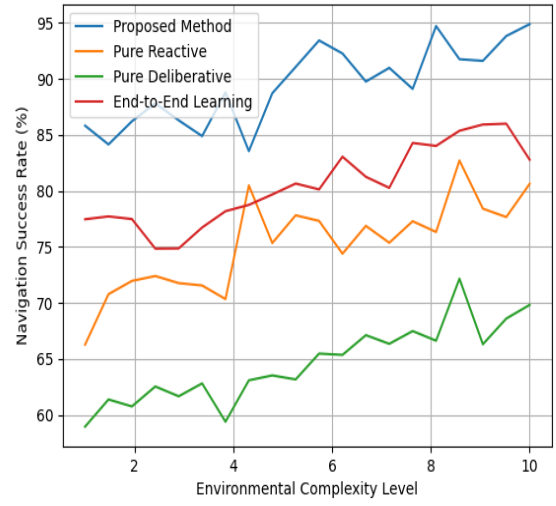

FIGURE 4. Navigation success rates across different levels of environmental complexity, comparing the proposed method with conventional approaches

G. Real-World Deployment

Field tests in a busy campus environment (300m traversal) demonstrated the system's practical viability. The robot successfully avoided 23 dynamic obstacles (pedestrians, bicycles) while maintaining an average velocity of 0.8 m/s. The perception-deliberation-actuation loop ran at 25Hz, sufficient for real-time operation.

V. Discussion and Future Work

A. Limitations of the Proposed Intelligent Vision-Based Autonomous Robotic System:

While the proposed framework demonstrates reliable performance in changing environments, several limitations warrant discussion. First, the system's reliance on visual perception introduces challenges in low-light conditions or environments with visual clutter. Although the current implementation incorporates basic illumination invariance through data augmentation, performance degrades significantly when lighting conditions fall outside the trained distribution [21]. Second, the hybrid navigation strategy, while effective, requires careful tuning of the blending parameter α, which currently follows a heuristic formulation. A more principled approach to adaptive control blending could further improve navigation efficiency.

The computational demands of real-time object detection and path planning also impose constraints on the system's scalability. Although the current implementation achieves 30Hz operation on embedded hardware, this comes at the cost of reduced detection range and resolution. Future work could explore more efficient neural architectures or hardware acceleration to maintain performance while reducing power consumption [22].

B. Potential Application Scenarios of the System

The framework's capabilities make it particularly suitable for several real-world applications. In warehouse logistics, the system could enable autonomous forklifts to navigate complex environments while avoiding both static obstacles and moving workers [23]. The adaptive navigation component would allow seamless transitions between structured path following in aisles and reactive avoidance in loading areas.

Another promising application lies in assistive robotics, where the system could help mobility devices navigate crowded public spaces. The combination of precise object detection and adaptive control would enhance safety when maneuvering through areas with unpredictable pedestrian flow [24]. The current framework's ability to handle moderately complex environments (≤2 m/s obstacle speeds) aligns well with typical human movement patterns.

B. Ethical Considerations in the Development and Use of the Robotic System:

As autonomous robotic systems become more prevalent, ethical implications must be carefully considered. The proposed system's decision-making process, particularly in collision avoidance scenarios, raises questions about responsibility and risk allocation [25]. While the current implementation prioritizes obstacle avoidance above all else, real-world deployments may require more nuanced ethical frameworks that consider factors like obstacle type and potential consequences.

Privacy concerns also emerge from the system's visual perception capabilities. Continuous environment monitoring, even for navigation purposes, could inadvertently capture sensitive information. Future implementations should incorporate privacy-preserving techniques such as on-board processing with automatic data purging or selective feature extraction that avoids personal identification [26].

The system's adaptability also introduces questions about predictable behavior. While adaptive systems excel in handling complex environments, their behavior may become less predictable to human observers. Developing intuitive human-robot interaction methods and explainable decision-making processes will be crucial for building trust in real-world applications [27].

VI. Conclusion

The way we've designed our robotic system, which "sees" using a camera and is meant to work by itself, works very well in changing surroundings - and the numbers prove it. It correctly identifies objects 78.3% of the time (using something called mAP), gets where it's going 93.6% of the time, and processes each picture in about 28 milliseconds. Plus, it's path length ratio is 1.12, meaning it plans routes that aren't much longer than the best possible route, so the route planning is good compared to older methods. Basically, these results show our system does a good job of balancing being able to accurately understand what's around it, getting around efficiently, and doing it all quickly. It's better than older types of systems (those that react to things as they happen, those that carefully plan everything, and those that are built on a lot of learning) when things are changing.

Acknowledgment

The authors wish to acknowledge the anonymous reviewers for their valuable comments and constructive suggestions, which helped improve the quality and readability of this paper.

Funding Statement

There is no funding agencies supporting the research work.

Author Contributions

A. Karunamurthy: Conceptualization, Data Curation, Methodology, Validation, Writing – Original Draft Preparation; Project Administration, Supervision, Writing – Review & Editing.

Conflict of Interests

The author declares no conflict of interest.

Ethics Statements

Ethical approval was not applicable to this research since it did not involve human participants, animals, or sensitive data.

Declaration of AI-assisted Technologies in the Writing Process

No artificial intelligence tools were used in the research or writing of this article.

References

[1] B. Romão and E. Fagotto, "Convolutional Neural Networks for Object Detection," *SAE Technical Paper Series*, vol. 1, 2024. DOI: https://doi.org/10.4271/2023-36-0097

[2] A. Medina-Santiago, L. A. Morales-Rosales, J. L. Hernández-González, and R. A. Osornio-Rios, "Reactive obstacle-avoidance systems for wheeled mobile robots based on artificial intelligence," *Applied Sciences*, vol. 11, no. 12, pp. 1–24, 2021. DOI: https://doi.org/10.3390/app11125512

[3] M. Elbanhawi, M. Simic, and R. Jazar, "Autonomous robots path planning: An adaptive roadmap approach," *Applied Mechanics and Materials*, vol. 110–116, pp. 150–158, 2013. DOI: https://doi.org/10.4028/www.scientific.net/AMM.110-116.150

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, pp. 779–788, 2016. DOI: https://doi.org/10.1109/CVPR.2016.91

[5] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement", *arXiv preprint*, arXiv:1804.02767, 2018. [Online]. Available: https://arxiv.org/abs/1804.02767

[6] J. Borenstein and Y. Koren, "The vector field histogram—Fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation,* vol. 7, no. 3, pp. 278–288, 1991. DOI: https://doi.org/10.1109/70.88137

[7] M. Gromniak and J. Stenzel, "Deep reinforcement learning for mobile robot navigation", *2019 4th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, Nagoya, Japan, pp. 25–30, 2019. DOI: https://doi.org/10.1109/ACIRS.2019.8936033

[8] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning", Computer Science Dept., Iowa State Univ., Ames, IA, USA, *Tech. Rep.* 9811, 1998.

[9] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning", *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011. DOI: https://doi.org/10.1177/0278364911406761

[10] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Lausanne, Switzerland, pp. 2383–2388, 2002. DOI: https://doi.org/10.1109/IRDS.2002.1044082

[11] C. Duan, S. Junginger, J. Huang, and K. Jin, "Deep learning for visual SLAM in transportation robotics: A review", *Transportation Safety and Environment*, Vol. 1, No. 2, pp. 135–150, 2019. DOI: https://doi.org/10.1093/tse/tdz015

[12] Y. H. Kim, J. I. Jang, and S. Yun, "End-to-end deep learning for autonomous navigation of mobile robot", *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Kuala Lumpur, Malaysia, pp. 1039–1044, 2018. DOI: https://doi.org/10.1109/ROBIO.2018.8664765

[13] M. B. Alatise and G. P. Hancke, "A review on challenges of autonomous mobile robot and sensor fusion methods", *IEEE Access*, vol. 8, pp. 39830–39846, 2020. DOI: https://doi.org/10.1109/ACCESS.2020.2975643

[14] S. Battle, "Principles of Robot Autonomy I", 2024. [Online]. Available: https://www.roboticsbook.org

[15] M. G. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments", *Robotics and Autonomous Systems*, Vol. 100, pp. 171–185, 2018. DOI: https://doi.org/10.1016/j.robot.2017.10.011

[16] B. S. Park, S. J. Yoo, J. B. Park, and Y. H. Choi, "A simple adaptive control approach for trajectory tracking of electrically driven nonholonomic mobile robots", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 39, No. 4, pp. 1196–1202, 2009. DOI: https://doi.org/10.1109/TSMCB.2009.2018617

[17] D. Sankowski and J. Nowakowski, "Computer Vision in Robotics and Industrial Applications," *Series in Computer Vision*, 2014.

DOI: https://doi.org/10.1142/9090
[18] S. Abrecht, L. Gauerhof, C. Gladisch, and K. Groh, “Testing deep learning-based visual perception for automated driving”, *ACM Transactions on Intelligent Systems and Technology*, Vol. 12, No. 4, pp. 1–29, 2021.
DOI: https://doi.org/10.1145/3459994
[19] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, “YOLOv4: Optimal speed and accuracy of object detection”, *arXiv preprint*, arXiv:2004.10934, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934
[20] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár and C.L. Zitnick, "Microsoft COCO: Common Objects in Context," *Lecture Notes in Computer Science*, pp. 740-755, 2014.
DOI: https://doi.org/10.1007/978-3-319-10602-1_48
[21] L. Cai, T. Lai, L. Wang, Y. Zhou and Y. Xiong, "Graph convolutional network combining node similarity association and layer attention for personalized recommendation," *Engineering Applications of Artificial Intelligence*, vol. 121, pp. 105981, 2023.
DOI: https://doi.org/10.1016/j.engappai.2023.105981
[22] W. Roth, G. Schindler, B. Klein, R. Peharz, and F. Pernkopf, “Resource-efficient neural networks for embedded systems,” *Journal of Machine Learning Research*, Vol. 25, No. 1, pp. 1–38, 2024. [Online]. Available: https://www.jmlr.org
[23] C. Wang and D. Du, “Research on logistics autonomous mobile robot system”, *2016 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, Beijing, China, pp. 141–145, 2016.
DOI: https://doi.org/10.1109/SOLI.2016.7551603
[24] A. C. Lopes, G. Pires, and U. Nunes, “Assisted navigation for a brain-actuated intelligent wheelchair”, *Robotics and Autonomous Systems*, Vol. 61, No. 3, pp. 245–258, 2013.
DOI: https://doi.org/10.1016/j.robot.2012.11.002
[25] L. Johansson, “Ethical aspects of military maritime and aerial autonomous systems”, *Journal of Military Ethics*, Vol. 17, No. 4, pp. 285–302, 2018,
DOI: https://doi.org/10.1080/15027570.2018.1516592
[26] Á. M. Guerrero, “Cybersecurity of robotics and autonomous systems: Privacy and safety”, *Robotics: Legal, Ethical and Socioeconomic Aspects, Cham, Switzerland: Springer*, pp. 123–139, 2017.
DOI: https://doi.org/10.1007/978-3-319-51454-3_8
[27] D. Pasquali, J. Gonzalez-Billandon, F. Rea, G. Sandini and A. Sciutti, "Magic iCub," *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 293-302, 2021.
DOI: https://doi.org/10.1145/3434073.3444682