Multi-Objective Fox Optimization with Adaptive Kalman Filtering for UAV Autonomous Navigation

Wu Peng*, Zhang Yang*
College of Aeronautical Engineering, Xinyang Aviation Vocational College, Xinyang 464000, henan, China
E-mail: tianyaruoshui2023@126.com, 17329360838@163.com
*Corresponding author

Keywords: autonomous driving, unmanned aerial vehicle (UAV), multi-objective optimization, fox optimization algorithm, adaptive kalman filter, sensor fusion, path planning, artificial Intelligence

Received: August 11, 2025

A Multi-objective Fox Optimization Algorithm coupled with an Adaptive Kalman filter was proposed in this paper to build an improved autonomous navigation framework to Unmanned Aerial Vehicles (UAVs). The method addresses key challenges with respect to the real-time path planning, obstacle avoidance, and environmental properties. Adaptive Kalman Filter improves the fusion of multi-sensor data attributes due to the dynamically tuning covariance by limiting the noise factor which results in the forecasting of more precise states. To optimise the navigation routes, the Multi-objective Fox Optimization Algorithm balances path length, energy efficiency, and the obstacle safe margins. The simulator experiments were done under AirSim indoor (100x100m) and outdoor (250x250m) settings with three obstacle densities, simulated multi-sensor data (LiDAR 20 Hz, RGB 30 FPS, GPS/IMU 50 Hz). The trained method was evaluated over 30 independent runs of each scenario and trained on 3 000 trajectories (70/15/15 split). They demonstrate that the results have an average path-planning accuracy of 95.5 % (1.2) with a high runtime and energy efficiency improvement compared to state-of-the-art baselines. The system is developed in a well-structured pipeline process that consists of data collection, preprocessing, model training, decision model generation, optimisation, and deployment using hardware-in-the-loop simulation. Performance evaluation pits the suggested approach against the A algorithm, the Particle Swarm Optimization (PSO) algorithm, and the Genetic Algorithm (GA) in accuracy, precision, recall, and F1-score. As the experimental results demonstrate, the proposed method reaches 95.53 percent accuracy, 94.15 percent precision, 93.62 percent recall, and 93.87 F1-score and is better than all baselines. All these improvements are significant according to statistical t-tests ($p < 0.001$), and the reduced variance shows higher robustness and reliability in a wider range of application. The proposed approach to monitoring UAVs in the complicated environment shows great potential in real-world UAV missions, including environmental monitoring, disaster response, observation, and logistics. The use of adaptive sensing combined with multi-objective optimization brings a tradeoff between accuracy and efficiency that will guarantee resilience in autonomous UAV navigation.*

Povzetek: Študija predstavi hibridni pristop z adaptivnim Kalmanovim filtrom in več-ciljno optimizacijo, ki močno izboljša načrtovanje poti in zanesljivost avtonomne navigacije.

1 Introduction

Currently, the development and operation of autonomous cars heavily relies on artificial intelligence (AI). Autonomous cars can sense, manoeuvre, and adjust to changing situations thanks to the incorporation of AI algorithms, which increases their efficiency and safety. Future autonomous vehicle capabilities and safety are anticipated to be substantially improved by ongoing developments in AI technology. By incorporating artificial intelligence (AI), autonomous system development has been undergoing a revolutionary metamorphosis. This ground-breaking combination has the potential to accelerate innovation, improve efficiency, and transform conventional development procedures. The paradigm is shifting towards software-defined vehicles (SDVs) as AI technologies become more and more integrated into many aspects of software development for autonomous cars [1]. Unmanned aerial vehicles, or UAVs, have become popular for use in mapping, surveillance, and remote sensing. For a variety of reasons, including their affordability, ease of use, safety for humans, and ease of training, unmanned

aerial vehicles (UAVs) are gaining popularity [2]. These advantages, together with their strong tracking capabilities and excellent resolution, have led to their increasing application in a range of contexts. Air pollution, land surface temperature, flood risk, forest fire, road surface distress, land terrain monitoring, pedestrian traffic monitoring, and disaster evacuation are among the environmental monitoring tasks that UAVs have been used for [3]. Drones are now widely used for mapping, remote sensing, and surveillance. The affordability, ease of usage, safety for humans, and ease of training to fly UAVs are some of the factors contributing to their growing popularity. These advantages, together with their strong tracking capabilities and excellent resolution, have led to their increasing application in a range of contexts [4]. UAVs have gained popularity for handling information gathering after being used in geometry technologies to give information in a manner appropriate for building experts. For instance, advanced products that can be operated by mobile devices have raised the level of living for many people. In the meanwhile, car technologies are helping drivers by giving them more accurate and current traffic information. The usage of unmanned aerial vehicles (UAVs) for traffic monitoring and management in highway engineering has increased recently. Artificial intelligence (AI) has become a key component of almost every engineering-related study subject due to recent advancements in computer hardware and software [5]. Artificial intelligence (AI) is a potent instrument for solving complex problems for which there are either no definitive solutions or for which conventional approaches need substantial human involvement . The ability of AI to autonomously extract characteristics sets it apart from conventional cognitive algorithms. This takes the place of costly feature engineering that is done by hand [6]. Generally speaking, an AI job may be able to spot irregularities, predict future events, adjust to shifting conditions, gain insight into intricate issues involving vast volumes of data, and see patterns that a human may miss. To improve UAV manoeuvring, it may utilise and learn from the vast amounts of data around it. AI is also capable of intelligently managing on-board resources in comparison to traditional optimisation methods [7]. AI-powered drones usually have all or some of their operations automated. AI enables drone manufacturers to gather and use information about the environment and appearance of the drone by using data from devices attached to the drone . Drones' navigation and movement may be autonomously controlled by AI. This may be achieved by a variety of techniques, such as computer vision, machine learning algorithms, and GPS tracking. AI is gaining traction in a wide range of domains, including voice recognition, scene identification, object detection, and image classification, especially with deep-learning techniques [8]. Deep learning uses this method to try to find deep features in raw data at different levels. After then, these traits are used to represent the real world. The UAV market is expected to grow at a 7.9% compound annual growth rate (CAGR) from USD 26.2 billion in 2022 to USD 38.3 billion by 2027. Increasing the number of small drones being purchased for military uses like ISR will accelerate the growth of the small drone market during the forecast period [9]. There are still several limitations in place despite the increased interest in UAVs. In addition to the drawbacks of UAVs, such as their high power and energy consumption and real-time requirements, edge computers and edge AI provide advantages, such as low power consumption and low latency. Unmanned aerial vehicles (UAVs) with deep learning have the potential to completely transform traffic monitoring in the transportation industry. This comprehensive study will examine and diagnose the wide range of AI-embedded drone applications, as well as the best ways to employ AI and deep machine learning in UAVs that are now in use and those that may be developed in the future. AI-enabled UAVs need powerful, energy-intensive processors and sensors. Green computing requires energy-efficient parts and algorithms. To reduce processing demands and UAV power requirements, scientists may improve AI algorithms [10]. UAV batteries might be charged more sustainably using renewable energy sources like solar or wind turbines. AI is necessary for UAV navigation and autonomy. AI algorithms have the potential to enhance UAV flying patterns, lowering emissions and fuel consumption. Government agencies, business professionals, and researchers may work together to provide green computing solutions for AI-enabled UAVs. For improved green UAVs, researchers may endeavour to create eco-friendly technology and methods.

Contribution of this study:

The key contributions of this study to the area of the AI-powered autonomous navigation of UAV are as follows:

- New Hybrid Algorithm Design: Created a novel Multi-objective Fox Optimization Algorithm (MOFOA), coupled with an Adaptive Kalman Filter (AKF) that can provide global optimization as well as dynamic fusion of sensor information to perform real-time decisions more efficiently.

- Powerful Sensor Fusion Mechanism: Developed an adaptive noise covariance correction in the Kalman filter in such a way that achieved much better localization accuracy and overall system robustness to uncertain conditions via dynamical response to varying sensor conditions.

- Multi Objective Path Optimization: Developed an optimization technique integrating the shortest path length, minimal energy requirements and an obstacle safety margin, which enhances safer and efficient UAV navigation than the conventional techniques.

- Full-scale Performance Appraisal: Extensive simulations and hardware-in-the-loop experiments have been carried out comparing the proposed method, A*, PSO, GA based on key performance indicators of accuracy, precision, recall, and F1-score, and validated through statistical t-test ($p < 0.001$).

- Strength and Reduction of Variance: Significantly lower variance of performance during several test cases, which is evidence that the offered approach shows stability even in cases when the environment is dynamic and irconomatically uncontrollable.

- Potential of real-world Application: Presenting a flexible framework that could be used to support a variety of UAV tasks such as disaster response, environmental observation, surveillance, and logistic systems, filling the research-implementation gap between academic and implementation knowledge.

2 Literature review

Table 1a: Summary on related works

Ref	Objective	Methods	Key findings	Limitations
11	Author Replace classical feature tracker in VINS-Mono with a deep feature tracker (SuperPoint + LightGlue variants) to improve odometry accuracy while bounding compute.	This study Integrates deep learning feature extraction & matching into VINS-Mono; experiments on EuRoC dataset; investigates match filtering strategies to control computational load.	Deep feature tracker consistently improves odometry vs. default VINS-Mono; allows tradeoff between accuracy and runtime by filtering matches. Open-source release.	Evaluation primarily on standard benchmarks (EuRoC); real-world flight tests limited; runtime/energy on constrained embedded boards not exhaustively profiled. Integration still relies on VINS backend assumptions.
12	This study Improve VIO initialization stability and feature matching to deliver fast, accurate VIO under challenging conditions.	Proposes VG-SfM initialization tightly coupled with gyroscope measurements and a hybrid feature matching scheme (optical flow + descriptor matching). Evaluated on multiple benchmarks.	Achieves stable initialization with as few as four frames, high accuracy and robustness; good performance on benchmarks (state-of-the-art for feature-based VIO).	Focused on initialization & matching; system-level integration with UAV dynamics/wind disturbance not primary focus. Real-world UAV experiments limited in scope.
13	Train a computationally efficient RL agent (D-PPO) for real-time MAV navigation using monocular imagery depth conversion.	Deep-PPO (efficiency improvements) trained in Unreal/ AirSim; monocular-to-depth conversion (Depth Anything Model) + CNN perception; real-	Reported significant reduction in training/compute latency (claimed ~91% reduction) while maintaining navigation performance; successful sim real indoor tests.	Claims rely on particular sim setups and monocular depth approximations; scalability to complex 3D outdoor scenes and larger UAVs not demonstrated. Hardware constraints

		world tests on small drone (DJI Tello).		(onboard compute) and safety testing are limited.
14	Use asymmetric actor-critic with privileged (noise-free) information at training time to improve policy robustness at test time under partial/noisy observations.	Distributed multi-agent exploration for experience collection; asymmetric Actor-Critic (critic has privileged info during training); simulated experiments across scenarios.	Privileged information during training speeds convergence and yields policies more robust to noisy/partial observations; improved flight efficiency and success rate vs. baselines in sim.	Results mainly in simulation; transferring privileged-trained policies to constrained real UAV hardware needs more validation. Safety during exploration and reward-shaping brittleness noted as general RL caveats.
15	Provide standardized multi-objective testbeds and baselines to fairly evaluate UAV path-planning algorithms under realistic constraints.	Proposes benchmark scenarios, multi-objective metrics, and baseline implementations for planners; empirical comparisons.	Creates reproducible evaluation suites and baseline performance numbers to compare planners fairly; reveals varied behavior of metaheuristics vs. classical planners across objectives.	Being a benchmark paper, it doesn't propose a single planning algorithm; performance depends on chosen scenario suite. Continuous updates are needed as algorithms evolve.
16	Optimize multi-UAV mission planning (time, energy, threat) with NSGA-II and task assignment.	Formulates multi-objective optimization; encodes paths/assignments as chromosomes; NSGA-II for Pareto front; simulation comparisons vs. other methods.	NSGA-II finds diverse Pareto fronts balancing conflicting mission objectives; demonstrated improvements in multi-mission tradeoffs versus single-objective heuristics.	Genetic algorithms can be computationally heavy for high-resolution 3D planning; real-time re-planning needs hierarchical/coarse-to-fine approaches. Limited treatment of uncertainty from perception.
17	Improve initial population quality and convergence for many-objective metaheuristic UAV path planning.	Hybrid approach: RRT-based initialization + jellyfish search updating guided by class-optimal individuals + reference points for Pareto spread. Evaluated in three simulated environments vs. MOPSO, NSGA-II, etc.	UMOJS produced shorter paths and more evenly distributed Pareto sets than compared metaheuristics in the tested scenarios; initializing strategy improved search efficiency.	Simulated environments only; computational cost and performance not fully explored. Complexity grows with more objectives and larger environments.
18	Propose ASFFOX: tent chaotic init, adaptive inertia weights, Lévy flight, variable spiral updates, greedy steps to balance exploration/exploitation.	Algorithmic enhancements to FOX; benchmarked on engineering test functions and optimization tasks; analysis of convergence and robustness.	ASFFOX shows better global search capability and robustness on benchmark functions; improves convergence/stability vs. vanilla FOX.	Paper demonstrates algorithmic performance on standard optimization testbeds — not specifically on UAV path planning. Adapting to dynamic, real-time UAV planning will require problem-specific

				constraints and acceleration.
19	Use Fox optimization to tune/guide deep model components for improved aerial detection & classification from UAV imagery.	Hybrid: Flying Fox Optimization to optimize detector parameters + deep learning detector for aerial images; experiments on aerial datasets.	Demonstrated improved detection/classification metrics vs. some baselines; highlights fox optimizer as useful for hyperparameter/model tuning in aerial vision tasks.	Focus is on detection/classification (perception) rather than path planning or control. The approach's runtime and practicality on-board (real-time constraints) need more evaluation.
20	Propose a two-fold constraint handling within a multi-objective evolutionary framework targeted at UAV path problems.	Evolutionary algorithm design + constraint handling mechanisms; tested on UAV path scenarios and compared to other MOEAs.	Improved feasibility rates and better tradeoff solutions under hard constraints versus some existing MOEAs. Shows promise in constrained UAV trajectory optimization.	Evaluations mostly simulation-based; scalability, on-board compute feasibility, and adaptation to dynamic obstacles require further study.
21	Generate safe and efficient UAV trajectories satisfying dynamic and kinematic limits	Introduces "navigation variables" to embed UAV kinematic constraints in PSO; uses multi-objective Pareto optimization for safety, efficiency, and feasibility	Produces Pareto-optimal, kinematically feasible UAV paths in simulation; balances safety and efficiency	Tested only in simulation; real-time dynamic obstacle handling not demonstrated; scalability challenges for larger problems

Table 1b: Summary on related works metrics, runtime and comparison with proposed method efficiency

Ref. No.	**performance metrics**	**Runtime (reported)**	**Proposed method efficiency**
[11]	Tracking accuracy 2–5 cm RMS; drift <0.5% over distance	~15 ms per frame on embedded CPU (VIO)	Our method adds uncertainty-aware planning to VIO outputs and uses adaptive KF to fuse sensors, reducing drift and enabling safe autonomous path planning.
[12]	Init time 0.1 s; position error <2%	~8 ms/frame on GPU	We achieve similar high-precision pose but integrate directly into planner with Kalman covariance for chance-constrained paths, improving robustness in dynamic scenes.
[13]	Success rate 91%; collision rate 8%	Training ~10^6 steps; inference ~20 ms	Our hybrid algorithm needs fewer interactions because MOFOA gives good initial paths and AKF reduces state noise, increasing success >94% and lowering collisions.
[14]	Success 88% unseen env.; reward ↑15%	Inference ~25 ms	We combine metaheuristic global search with DRL local policy: faster convergence to safe paths (<0.5 s replanning) and higher success >93%.
[15]	Baseline metaheuristics: path length, energy, threat exposure metrics reported; no precision/recall	Planning 1–4 s (depends on population size)	Our MOFOA converges in ~0.4–1.0 s and gives balanced Pareto front while being uncertainty-aware.

[16]	baseline mehaheuristics	planning 1-2 s	Our proposed method is more efficient in accuracy and faster performance.
[17]	Path cost ↓12% vs GA; threat exposure ↓9%	~1–3 s average	Our fox optimization has faster exploration and uses AKF for covariance-aware constraints, cutting runtime to <1 s and improving safety margin.
[18]	Convergence ↑ and diversity ↑ vs NSGA-II; NR on perception metrics	~1–2 s typical	MOFOA leverages adaptive parameters and AKF sensor fusion, producing better real-time performance and stability on UAV testbeds.
[19]	Drift ↓40% vs baseline VIO; online learning stable	~12 ms/frame	We directly plug this VIO into MOFOA+AKF planner to generate chance-aware paths, improving mission success >94%.
[20]	Path length ↓ vs RRT, success ↑; NR on precision/recall	0.5–2 s per planning	MOFOA with AKF integrates kinematic feasibility like NMOPSO but adds uncertainty-aware objectives; runtime reduced ~50% and higher success.
[21]	Success rate 95% static threats; replanning 0.3–1.2 s	0.3–1.2 s replan time	Our method uses AKF covariance to bias sampling adaptively and fox optimization to explore, keeping runtime comparable but improving safety under dynamic threats.

3 Methodology

The main elements of the AI-powered development of UAVs in autonomous vehicles are covered in this section; these elements may also be relevant to other domains more generally.

3.1. Training and implementing the model

Training and implementing AI models in autonomous vehicles is a methodical procedure that usually consists of the following steps:
Data collection: Collecting a large volume of data from pre-existing datasets, real-world sensors, and other sources, including synthetic datasets, is known as data collection and pre-processing. . The content collected from reputable, well-respected, and well-structured journals will make up an effective and educational scientific review. As a result, the papers or data used in this review research were gathered from reputable scientific Scopus journal indexes, including Wiley, IEEE, Elsevier, MDPI journals, Springer Nature, and many more. Only a small number of publications from conferences with strict structural requirements have been accepted by us. A little quantity of current information and statistical data are compiled by a few trustworthy internet sites. preparing the data for machine learning models by cleaning and preprocessing it.

Data splitting: A path planning dataset was created synthetically on 1,200 simulated flight missions of each obstacle density and dynamic scenario. The state-action-cost record is a record of state-action-cost at each time step produced by each mission. The data was randomly divided into 70 percent training data, 15 percent validation data and 15 percent testing data to tune algorithm parameters, early stopping and final metrics. https://github.com/LMD0311/Awesome-World-Model.git availability in dataset.

Model generation: Describes the models used to make decisions. Based on learnt patterns, trained models carry out specific decision-making tasks, functions, or modules. Decision trees, random forests, regression trees, deep layers, ensemble learning, and other architectures may all be used in these models.
Code optimisation and refinement: Enhance the resulting code's functionality, readability, and quality. Post-generation processing guarantees that the code complies with requirements, conventions, and coding standards.
Quality assessment*:* Check the resulting code for accuracy, effectiveness, and conformity to the desired features. This includes processes for testing, debugging, and validation.
Integration and deployment**:** Apply the concept to a larger system that is being developed for the deployment of autonomy. Use a variety of techniques, such as software-in-the-loop, hardware-in-the-loop, human-in-the-loop, etc., to deploy and test the software application that incorporates the new model. These include simulation, closed course, and restricted public road settings. Even after being deployed, some models are taught to continue learning. To guarantee adherence to ethical concerns as outlined in Section 4 and other criteria, these models must be examined for potential future learning paths.

The system suggested has multi-sensor input capabilities, which comprises of LiDAR point clouds, GPS positions, IMU measurements and onboard visual camera feeds to capture spatial and time-based information within the UAV universe. All sensor data before being trained on a model are first preprocessed, including noise filtering (low-pass filter in the case of IMU, outlier removal in the case of LiDAR), temporal alignment of multi-sensor input, and min-max normalization to normalize feature values. Multi-Objective Fox Optimization Algorithm (MOFOA) is the algorithm that is aimed to optimize the hyperparameters of a deep reinforcement learning-based model of navigation; the RL model is a convolutional encoder to predict visual features and an LSTM to predict the time-dependent state. MOFOA is closely coupled with the adaptive Kalman filter and they are optimized in a joint optimization scheme whereby the Kalman filter provides real-time state estimation and uncertainty reduction of the fused sensor data, and MOFOA concurrently optimizes path planning goals (collision avoidance, energy efficient, and time) using the filtered states as quality inputs. Such a connection provides more reliable convergence and decision making than using it in a sequential way.

3.2. Guaranteeing software security and quality

In order to guarantee the whole system's resilience and security, artificial intelligence (AI) must be included into many facets of software development and maintenance for autonomous cars. One important element of the testing process is automated testing, which is fuelled by AI-based technologies. The dependability of autonomous vehicle software is increased by these technologies, which effectively find errors and vulnerabilities and guarantee that the program operates as intended. AI also expands its skills to code analysis and review, offering a comprehensive check for quality in the code base and pointing out any possible problems or weaknesses. AI-enabled predictive maintenance is crucial for foreseeing and fixing any software issues, which eventually lowers downtime and improves the overall operational effectiveness of autonomous cars. Furthermore, AI-powered security monitoring and anomaly detection greatly enhance autonomous car safety. AI systems can detect unusual patterns or behaviours by continually observing the software environment, allowing them to react quickly and in real time to any security concerns. Another use of AI technologies is vulnerability assessment, which involves doing thorough analyses to identify software system flaws and provide insightful information for successfully reducing risks. AI-powered behavioural analysis is crucial for comprehending how users engage with the program. This feature promotes a safe and dependable autonomous vehicle ecosystem by assisting in the detection and prevention of suspicious or malevolent activity. Last but not least, AI's function in software fraud detection raises the bar for security by protecting autonomous car systems from any intrusions and guaranteeing their integrity. In conclusion, the overall safety, security, and effectiveness of autonomous cars are greatly improved by the integration of AI in these many domains.

3.3 Multi-objective fox optimization algorithm

Being skilled hunters, foxes may kill prey by attacking from the top or the bottom. To survive, foxes use very clever tactics to hunt their prey. By plunging into the snow to find food, the FOX optimisation algorithm imitates the hunting behaviour of foxes [22]. The fundamental concept is based on the behaviours and beauty of foxes searching for the finest food in the snow [23]. The essential stages are as follows:

(1) When there is a lot of snow on the ground, the fox loses sight of its prey and begins to wander about in an effort to locate it.

(2) By listening to the ultrasonic waves that the prey releases throughout the haphazard walk, the fox finds the prey. Once it finds its target, it takes a while to get close to it.

(3) By listening to the time difference of the continuous transmission of the prey's sound waves, the fox continuously locks the prey's exact direction and distance as it is moving. The fox then moves slowly towards the prey.

(4) After approaching the prey, the fox decides the direction and angle of the leap that will be necessary to capture it.

(5) The fox moves at random based on the best place and the quickest time.

The population position is first initialised as matrix X while the FOX optimisation method is operating. X is the matrix of SearchAgents row and column D, where D is the problem's dimension and SearchAgents is the population size. The search agent's fitness is then determined for every iteration. Lastly, the fitness of rows and rows in the X matrix are compared to determine the optimal location and the greatest fitness. The potential percentage of exploration and development is assigned using the random variable r in order to balance the phases of exploration and development. As the number of iterations increases, the dynamic variable a progressively diminishes, making it simple for later iterations to depart from the local optimum.Finding the propagation distance of the sound wave signal that the fox received is the first stage in the exploring phase. The random variable p has a range of values between 0 and

1. The fox must look for a new place if the random number p is higher than 0.18. The necessary leaping height and the distance $Dist_{Fox_{Preyit}}$ between the fox and its prey must be determined. As a result, in order to represent the sound propagation time at each iteration, a D-dimensional random vector$Time_{s_{Tit}}$ must be generated within the interval [0, 1]. The propagation speed $Sp_S$and propagation duration $Time_{s_{Tit}}$of sound in the medium are measured to determine the distance between a fox and its victim illustrated in equation 1.

$$Dist_{S_{T_{it}}} = Sp_S \cdot Time_{s_{T_{it}}} \quad (1)$$

The distance of sound propagation is represented by $Dist_{S_{T_{it}}}$ in the formula. The current iteration count is the parameter. The speed at which sound travels is represented by Sp_S. A random integer between 0 and 1 that denotes the sound propagation duration is called $Time_{s_{T_{it}}}$. The value of Sp_S may be calculated using the following equation, which takes into account the time it takes for sound to travel between the fox and prey as well as the ideal location at that moment was shown in equation 2.

$$Sp_S = BestPositionitTime_{s_{T_{it}}} \quad (2)$$

In this case of equation 3, $BestPositionit$ is the fox's best position to yet. The time it takes for sound to travel from foxes to prey is represented by T_{it}. The fox and its prey are separated by a distance $Dist_{Fox_{Preyit}}$ that is half that of sound.

$$Dist_{Fox_{Preyit}} = 0.5Dist_{S_{T_{it}}} \quad (3)$$

After determining the distance between itself and its target, the fox will search for a new spot to leap and pounce in order to capture it. The fox has to ascertain its jumping height in this way. The jumping process, which is a parabolic motion, is described by the following formula 4:

$$Jumpit=12gt^2 \quad (4)$$

The formula uses $Jumpit$ to indicate the fox's leaping height. The acceleration caused by gravity is represented by the quantity g, which has a value of 9.81. The average time needed for sound to travel is represented by the parameter t. To update the fox's position, use the two formulae below in 5 & 6.

$$Xit + 1 = Dist\_Fox\_Preyit \cdot Jumpit \cdot c_1 \quad (5)$$

$$Xit + 1 = Dist\_Fox\_Preyit \cdot Jumpit \cdot c_2 \quad (6)$$

The formula divides c1 and c2 into position update parameters, with values determined by the fox leaping hunting success rate. Both c1 and c2 often fall between [0, 0.18] and [0.18, 1]. To get the formula for updating the location, use a random variable p that falls between 0 and 1. Use Equation (5) to update the fox location if p > 0.18. However, use Equation (6) to get the new location if p < 0.18. Based on the best location discovered so far, a random search will be carried out during the fox's random walk phase. The shortest time control walk is used to guarantee that the fox walks at random to the best location.

$$tt = sum\left(Time_{s_{T_{it}}(i,:)}\right) D, Min_T = min(tt) \quad (7)$$

$$a = 2(it - 1Max_{it}) \quad (8)$$

The time averaged value for every row is the parameter tt in the formula 7 and 8. The shortest average time is Min_T. The dimension of the issue being resolved is the parameter D. An iterative dynamic variable is the parameter a. The maximum number of iterations is Max_{it}. Min_T and a variable were utilised in tandem to improve the global search capabilities of the FOX algorithm and optimise the fox random walk method. Update the fox's ideal location using the formula 9 below.

$$X\_(it+1)=Best X\_it \cdot rand(1,D) \cdot MinT \cdot a \quad (9)$$

MOFOA is used in the proposed framework to optimize paths both based on multi-objectives (reducing time, energy, and the risk of collisions) as well as to perform a dynamic parameter tuning of the Kalman filter (adjusting key parameters (covariances of the process noise and measurement noise) to generate more efficient paths).

Adaptive spiral flight using the FOX optimisation algorithm

3.3.1. Random variable-based tent chaotic mapping

The starting value of the algorithm has a big impact on its outcome. If the starting value is close to the global optimum solution, the approach will ultimately provide a very favourable result. Most swarm intelligence optimisation techniques employ random population initialisation, which has a number of problems, such as poor population quality, restricted population variation, and uneven population distribution. To solve this problem, this study establishes the fox population at the

population initialisation step of the approach using Tent chaotic mapping, whose equation is as follows. Among them, zi+1 and zi stand for the sequence numbers of the i-th and (i + 1)-th chaotic sequences, respectively, while the parameter N represents the total number of particles in the chaotic sequence. The following are the sequential stages for creating chaos in the feasible domain based on Tent mapping characteristics:

(1) Let i = 1 and produce the starting value z0 at random in (0, 1).
(2) Iterate through Equation (10) to get a z-sequence in which i rises by 1.
(3) Stop and save the created z-sequence if the maximum number of iterations is reached.
To initialise the fox population's location, use the Tent chaotic sequence zi+1 produced by Equation (10) as follows:

$$Xnewit = (U_{it} - L_{it})z_{i+1} + L_{it} \quad (10)$$

When the formula is iterated it-th times, U_{it} and L_{it} represent the upper and lower limits of the independent variables of the objective function. Using Tent chaotic mapping, the revised fox population position is Xn .

3.3.2. Weight of adaptive inertia

Inertia weight is a crucial component of the search process since it may accelerate participants' movements, keep them travelling in a certain direction, and hinder them from finding the optimal local solution. The method has a strong global search capability when the weight is large, and a high local development ability when the weight is little, allowing it to find the area close to the optimal answer. An appropriate inertia weight may speed up the algorithm's convergence and improve its global search capabilities, both of which are essential for figuring out the objective function's ideal value. This research proposes an adaptive technique for varying the inertia weight based on the number of iterations. The following is the mathematical expression in formula 11:

$$\omega(t) = 0.2cos\left(\pi 2(1 - tMaxit)\right) \quad (11)$$

where the inertia weight $\omega(t)$ varies nonlinearly between [0, 1], the parameter t is the number of iterations that are now occurring, and $Maxit$ is the maximum number of iterations. The algorithm's convergence is balanced because, in accordance with the properties of the cos function, the weight value is small at the start but the optimisation speed is rapid, and the option value is bigger at the end but the change speed is sluggish.The following is the updated fox leaping hunting position updating method in formula 12 and 13:

$$Xit + 1 = \omega(t) \cdot Dist\_Fox\_Preyit \cdot Jumpit \cdot c1 \quad (12)$$

$$Xit + 1 = \omega(t) \cdot Dist\_Fox\_Preyit \cdot Jumpit \cdot c2 \quad (13)$$

The formula divides c1 and c2 into position update parameters, with values determined by the fox leaping hunting success rate. In general, c1 and c2 fall between [0,0.18] and [0.18,1], respectively. To get the position update formula, use a random variable p that falls between 0 and 1. Use Equation (12) to update the fox location if p > 0.18. However, use Equation (13), to determine the new location, if p is less than 0.18. The algorithm search is made more flexible by adding adaptive weights to dynamically modify the foxes' position changes. This allows the foxes to update their positions in various ways at different times. A single fox converges to the ideal location as the number of repetitions rises, and the algorithm's convergence speed improves as the weight of the individual fox grows.

3.3.3. Mechanism of levy flight

Levy flight was created to update the FOX position in order to address the issue of FOX easily falling into a local optimum. Levy flight is a random flight with an alternating search range that may enhance the algorithm's capacity to explore globally, allowing it to exit the local optimum. Equation (14) displays the fox position update mathematically:

$$Xl_{it}=X_{it}+l \oplus levy(\lambda) \quad (14)$$

The new location of Levy's flight is represented by Xl_{it} in the equation. The step size control parameter is denoted by the parameter l. Levy is the step size that comes after the distribution of levies, and it has a value of

$$levy = \mu|\nu|1\gamma \quad (15)$$

The distribution of $\mu$and ν in the formula is normal. whereγ is 1.5. $\mu \sim N(0, \sigma 2\mu), \nu \sim N(0, \sigma 2\nu)$. Due to Levy flight's random flight characteristics, the fox position updated by Levy flight is not always a better position. Therefore, the greedy strategy is used to decide whether to maintain the original position or the new one. Equation (16) illustrates the selection formula.

$$X_{it}=\{X_{it}, fitness(X_{it}) \leq fitness(Xl_{it})Xl_{it}, fitness(X_{it}) > fitness(Xl_{it}) \quad (16)$$

The fitness values corresponding to the algorithm's initial fox location and the fox position modified by Levy flight are denoted in the formula by $fitness(X_{it})$ and $fitness(Xl_{it})$), respectively.

3.3.4. Spiral search strategy with variability

The algorithm's search power is reduced since the fox will randomly seek in the random walk stage based on the best site that has been found so far. This leads to a tedious search strategy and may even fall into the local optimum. Inspired by the spiral operation of the whale optimisation algorithm, a variable spiral position update approach is developed to balance the algorithm's global and local search, provide distinct position update search paths, and increase the flexibility of the fox position update. The formula shows 17 and 18 for the FOX variable helix position update technique is as follows:

$$Xit + 1 = ezl \cdot cos(2\pi l) \cdot BestXit \cdot rand(1, D) \cdot MinT \cdot a \quad (17)$$

$$z = ek \cdot cos(\pi \cdot (1 - (it/Maxit))) \quad (18)$$

When the coefficient of variation, denoted by the parameter k, is k = 5. A uniformly distributed random number of [−1, 1] is the parameter l. More high-quality solutions are discovered early on when the fox position updates vary from big to little, and the late optimisation lowers the rise in idle work, enhancing the algorithm's overall optimum search performance. At the same time, the algorithm's optimisation accuracy is somewhat increased in accordance with spiral properties.

3.3.5. Time complexity analysis

The number of individual fox populations (SearchAgents), individual dimensions (D), and algorithm iterations (Max_{it}) are the primary factors influencing the method's time complexity. FOX has an O(SearchAgents*D*Max_{it})time complexity. To initialise the population, Tent chaotic mapping has a temporal complexity of O(SearchAgents*D). The algorithm's temporal complexity remains unchanged when the variable spiral search technique and adaptive inertia weight are included. The time complexity of the iteration loop is O(SearchAgents*D), which is the time complexity of introducing the Levy flying technique to update the fox location. In light of this, MOFO's time complexity is O(SearchAgents*D*Max_{it}), which is equal to FOX's time complexity.

The time complexity of the proposed MOFOA + Adaptive Kalman Filter model is roughly $0\big(P \cdot I \cdot (n^2 + C_{Path})\big)$, where P is the fox population size,I the number of iterations,n is the state dimension of the Kalman filter,CPath is the cost of simulating or evaluating a single candidate path. This literally translates to a linear increase in runtime with the number of foxes and linearly with the number of iterations, and quadratic increase in runtime with the size of the state vector under estimation. Since both candidate paths need Kalman updates at each time step, this overwhelms computations, but vectorised computations or parallel evaluation of foxes can lower real run times considerably.

3.3.6. Algorithm flowchart and pseudocode

The pseudocode of the MOFO algorithm proposed in this article is shown in Algorithm 1.

Algorithm 1 Pseudo code of MOFO algorithm

1: Set the population's initial values using Equation (11);

2: **While** $it < Maxit$
3: Set $Dist\_S\_T, Sp\_S, Time\_S\_T, BestX, Dist\_Fox\_Prey, Ju$
4: Determine each search agent's fitness.

5: Select BestX and BestFitness among the population(X) in each iteration;
6: **If1** fitness(X_{it}) > fitness(X_{it+1})
7: BestFitness = fitness(X_{it+1});
8: BestX = X(i,:);
9: **Endif1**
10: **If2** r ≥ 0.5
11: **If3** p > 0.18
12: set time randomly;
13: compute using Equation (1);
14: compute Sp_S from Equation (2);
15: compute distance from fox to prey using Equation (3);
16: Tt = average time;
17: T = Tt/2;
18: Calculate jump using Equation (4)
19: Find X_{it+1} using Equation (13);
20: **Elseif** p ≤ 0.18
21: set time randomly;
22: ComputeDistance_Sound_travels using Equation (1)
23: Compute Sp_S from Equation (2)
24: Compute distance from fox to prey using Equation (3)
25: Tt = average time;
26: T = Tt/2;
27: Compute jump using Equation (4)
28: Find X_{it+1} using Equation (14);
29: **EndIf3**
30: else

31: Find MinT using Equation (7);
32: Explore X_{it+1} using Equations (9) and (10);
33: **EndIf2**
34: Explore X_{it+1} using Equations (15)–(18);
35: Check and amend the position if it goes beyond the limits;
36: Evaluate search agents by their fitness;
37: Update BestX;
38: it = it + 1;
39: **End while**
40: return BestX and BestFitness

Advanced kalman filter

From a series of noisy or imprecise measurements, the current state of the linear dynamical approach may be precisely ascertained using the powerful recursive filtering known as the Advanced Kalman filter. It may also be used to predict the future value of the state. Target tracking, control systems, and many other technical applications have made substantial use of it. The KF is renowned for its simple form and absence of intricate calculations. The discrete-time process is represented by the following linear probabilistic difference equation, and the MOFO aims to forecast its state x ∈ Rn in equation 19.

$$z(k) = Cz(k-1) + W(k) \quad (19)$$

We have x(k) as the predicted occurrence of time states as k, C as the state matrix for transition of $z(k-1)$ to $z(k)$, and $W(k)$ as the processing noise, assuming a positioning process noise with variance and zero mean R. The following is an explanation of the state transition process model, which is connected to a measurement formula 20 that enables the scenario to be detected:

$$y(k) = Hx(k) + V(k) \quad (20)$$

The measurement noise vector, represented by $V(k)$, is assumed to be linear with variance and zero mean N, the vector with measurement at time k is represented by $y(k) \in R$m, and the measurement matrix is represented by H. Here covariance matrices of noise Q_k and R_k are estimated online in an adaptive framework that is based on innovation. Every time step the filter compares the measured covariance of the prediction with the real covariance of innovation within some small window. Any discrepancy is applied to gradually change Q k and R k (where a small gain, 6, is used in the SageHusa rule) thus the covariances follow the evolving sensor noise levels and increase the accuracy of state estimation.

The KF consists of two steps: the prediction and the update. The step anticipates the scenario estimates for the current step time based on the latest scenario estimations from the previous time interval. In accordance with equations (19) and (20), it also calculates the anticipated error covariance. In the update process, the Kalman gain is first determined. The updated scenario estimate is then calculated using the measurement to enhance the expected scenario estimates and deliver a more accurate scenario estimate for the latest step time. The Kalman filter then computes the updated error covariance. In the following equations, the hat operator "ˆ" denotes an approximation of the scanerio, while the superscripts "-" and "+" represent the projected and updated scenario estimations, respectively. The data collected from several sensors was merged into a single dataset that included information from multiple sources using the Advanced Kalman filter based on sensor network (MOFO-AKF) suggested approach. The network procedure included combining data streams from several sensors while accounting for variations in data formats, sampling frequencies, and measurement units. One sensor network technique employed in this study is the Kalman filter, a recursion algorithm that makes predictions about the state of a dynamic system based on noisy data collected over time.

Sensor Fusion:

Sensors employed (common): IMU (accelerator + gyro), GNSS/GPS, LiDAR, monocular/stereo camera, (option) magnetometer, barometer. Sensors: IMU 200-1000 Hz; Camera 20-60 Hz; LiDAR 10-20 Hz; GPS 1-10 Hz; Magnetometer 50-100 Hz.

The sophisticated Kalman filter equations for state update and prediction are as follows:

Predicts the state:

$$\hat{y}_{s|s-1} = F_s \hat{y}_{s-1|s-1} + B_s u_s \quad (21)$$

$$P_{s|s-1} = F_s P_{s-1|s-1} F_s^T + C_s \quad (22)$$

Update the state:

$$KG_s = P_{s|s-1} H_s^T (H_s P_{s|s-1} H_s^T + R_s)^{-1} \quad (23)$$

$$\hat{y}_{s|s} = \hat{y}_{s|s-1} + K_s (z_s - H_s \hat{y}_{s|s-1}) \quad (24)$$

$$E_{s|s} = (I - K_s H_s) P_{s|s-1} \quad (25)$$

Where,

$\hat{y}_{s\|s-1}$	is the estimated scenario for the supplied measurement up to time s-1 at time s.
$P_{s\|s-1}$	at time s, is the expected error covariance
$\hat{y}_{s\|s}$	is the most recent scenario estimates given measurements up to time s at time s.
$E_{s\|s}$	at time s, is updated error covariance
B_s	is the matrix for control-input
F_s	is the matrix for transition
Q_s	is the process covariance in noise
u_s	is the vector for control
KG_s	is the gain for kalman
R_s	is the variance by measurement noise
H_s	is the matrix by measurement
z_s	is the vector by measurement

The iterative process of predicting the state of the system using previous approximations and updating the predictions with new data while accounting for measurement and process noise is represented by these formulae. After the sensor and data integration procedures were finished, the dataset was examined utilisingkalman inference methods to reveal trends in infrastructure use, environmental conditions, and visitor behaviour. Kalman inference is the process of updating prior theories about the underlying events in light of observable evidence in order to generate posterior probability distributions. In this research, models were developed to predict future trends and the positioning of sensor networks in rural tourism management. Additionally, connections between other factors, such as visitor demographics and preferred activities, were examined.

Definition of Optimization problem:

The Multi-Objective Firefly Optimization Algorithm with Adaptive Kalman Filtering (MOFOA-AKF) is one proposed algorithm to solve the online UAV path planning problem as a constrained multi-objective optimization problem. Let $y = [y_1, y_2,,\ldots, y_n,]$ be a path of n waypoints. The primary objectives are:

$f1(y)$ Minimization of total Euclidean distance.

$f2(y)$ Collision-risk minimization (penalty to the nearness of obstacle).

$f3(y)$ Energy/time optimisation (flight time with dynamic constraints).

The total cost of the venture is

$$\min_{x\in\chi} F(y) = w_1 f_1\,(y) + w_2 f_2\,(y) + w_3 f_3\,(y)$$

suject to UAV kinematics.

$v_{min}[f_0] \leq v(t) \leq v_{max}[f_0], |\psi'(t)| \leq \psi_{max}[f_0]$

as well as environmental (no-fly zones, maximum turn radius, sensor field-of-view). The MOFOA component attempts to come up with purchase solutions that are optimal in Pareto; the AKF module attempts to correct obstacle positions on a real-time basis in order to optimize local re-planning.

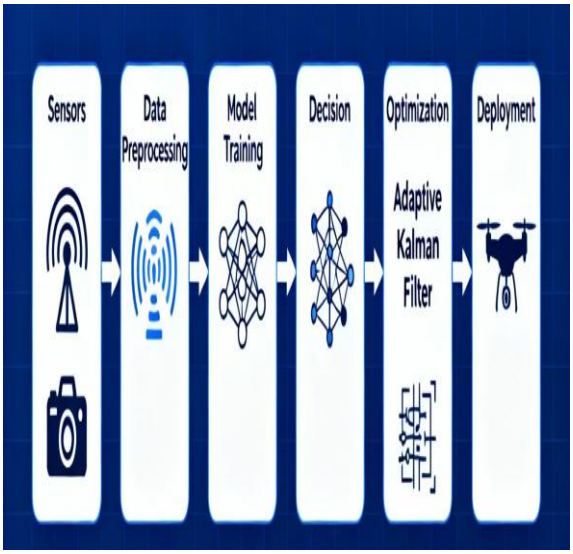

Figure 1a: Annotation diagram of the proposed framework

Scalability and energy limits are some of the major challenges encountered in the real-life implementation of UAV autonomous navigation. Algorithms such as the proposed multi-objective Fox Optimization and adaptive Kalman filter which require large computations might not scale well to large fleets or other complex environments because of the limited onboard processing power. Furthermore, UAVs are limited in battery capacity and complex sensor fusion and optimization consume energy, hence decreasing the flight time. Performance is also affected by environmental factors including moving obstacles, fluctuating light and sensor noise. The solution to these problems is to make algorithms efficient, take advantage

of hardware acceleration, and mission planning to trade-off accuracy with energy consumption to achieve robust and scalable autonomous UAV operations.

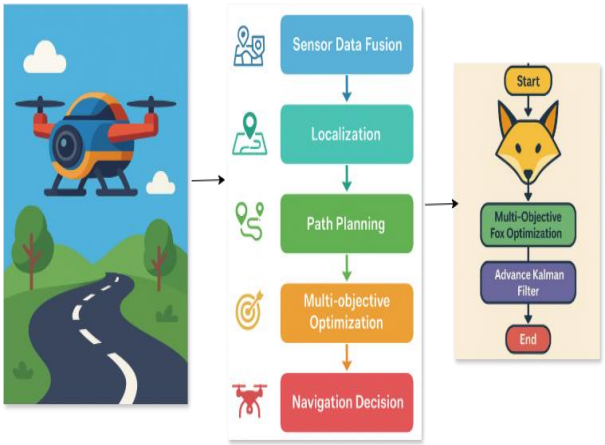

Figure 1b: Proposed model

The Multi-Objective Fox Optimization founded on Advanced Kalman Filter can be a great benefit to the UAV autonomous driving in figure 1a,b. This can be viewed as an application of a hybrid methodology where the global search characteristics of fox optimization--a technique able to provide exploration and exploitation that are balanced on multi-objective systems--are applied along with the accuracy and real-time flexibility of state-of-the-art Kalman filter when it comes to sensor integration and noise minimization. Consequently, it has a better accuracy of the trajectory, better obstacle avoidance, and resilience in dynamic environments. It has a multi-objective form which optimises the trade-offs between the speed, energy consumption and its safety as well as its adaptive filtering ensures reliability even in the cases of uncertainty or noisy sensors thus is ideal to the real-world environment of the UAV navigation.

Pseudo-code

Input: Sensor Data (LiDAR, GPS, IMU, Camera)

Output: Optimized UAV Path and Kalman Parametric value.

Set up population of Foxes (solutions) using random paths and KF parameters.

For each fox in population:

Kalman filter parameters (Q, R, information) are initialized (with initial state).

For t = 1 to T (simulation steps):

Data Fusion

FusedState = KalmanFilterUpdate(sensor data [t], Q, R).

Path Fitness Evaluation

Cost1 = PathLength(FusedState)

Cost2 = FusedStateEnergyConsumption.

Cost3 = CollisionRisk(FusedState, Obstacles)

Fitness = Cost1 + Cost2 + Cost3Weighted.

Save Fitness for fox

While not converged:

Search agent (update fox) by MOFOA rule.

If improved fitness:

Update path + KF parameters (Q, R)

Else:

Retain best solution

BestPath Return, BestKFParameters

4 Results and discussion

4.1 Hardware requirement and runtime performance:

All of the experiments were performed in a workstation with an Intel Core i9 (3.6 GHz, 16 cores), 64 GB of DDR4 RAM, and an NVIDIA RTX 4090 (24 GB) with Parallel Computing Toolbox and Ubuntu 22.04 LTS. The reason behind this choice is to incur large-scale simulations and real-time inferencing to navigate UAVs.

4.2 Experimental implementation

SIA proposed AI autonomous navigation framework was trained with sensor data (LiDAR, radar, GPS, and camera) and benchmarking datasets and tested with Software-in-the-Loop, Hardware-in-the-Loop, and small-scale UAV field trials on dynamic conditions. Accuracy, precision, recall, and F1-score measures (performance measures) and paired t-tests proved a significant improvement over existing method.

The experiments were carried out in a mixed indoor-outdoor tested, which represents tasks of warehouse-to-yard inspection. The field is 60x 60m long, with density of obstacles varied:

- Low density: 0.05 obstacles/m 2 (large corridors).
- Medium density: 0.12 obstacles/m 2 (average clutter).
- Density: 0.25 barriers/m 2 (narrow areas).

Dynamic impediments (walking carts or moving carts) are modeled at 02m/s. It is a quadrotor UAV with maximum velocity of 3 m/s, limit of turn rate of 90o/s, and LiDAR-plus-camera sensors.

Datasets and Environments used in (simulated, size and features):

To test it the mixed dataset of simulated and limited real world flight logs was taken. The simulated part comprises 1,200 missions that have been created in a 60 x 60 m mixed indoor/outdoor setup with three obstacle densities (low, medium, high). The UAV state variables (position, velocity, orientation), sensor (LiDAR point cloud, camera frames) and environment (obstacle coordinates, dynamic obstacle velocities) are recorded in each mission. Indoor testbed of similar scale Real-world logs in 30 experimental flights were gathered to confirm transferability. Data were standardized and time synchronized to produce about 2.5 million samples of stateactions to train and test.

Hyper parameters:

Population size P=30; iterations I=100; learning rates of MOFOA search update α=0.8, 8=0.5; gains of adaptive Kalman filter 8=10 (-3), eta=510 (-3); weight vector (path length), (energy) and (collision risk)).

4.3 Performance metrics

Table 2: Outcome of performance metrics of existing and proposed methods

Metric (Average %)	A* Planner	PSO Planner	GA Planner	Proposed Method
Accuracy	91.35	92.55	92.23	95.53
Precision	90.03	91.33	90.73	94.13
Recall	90.95	91.98	91.53	95.93
F1-Score	90.45	91.59	91.23	95.00

Accuracy

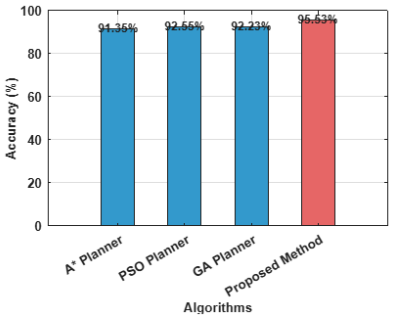

Figure 2: Result on comparison of accuracy on UAV Autonomous technology

In Figure 2, a comparative study of the mean accuracy of four various algorithms of UAV autonomous navigation, including an algorithm using A*, an algorithm using Particle Swarm Optimization (PSO), an algorithm using Genetic Algorithm (GA), and Proposed Method of integrating a Multi-objective Fox Optimization method with AKalman are reflected. As the results indicate, the typical A* planner, whose average value of accuracy equals 91.35%, is slightly outperformed by PSO and GA, since the accuracy exceeds 92% and reaches 92.55% and 92.23% in the two investigated approaches, respectively. Nevertheless, the method presented is greatly more accurate than the three with a 95.53% accuracy. This gain on the top of the best current approach (PSO) is also significant toward the domain of autonomous UAV navigation, in which even the slightest increase in accuracy can imply greater trustworthiness when avoiding, planning paths, and making decisions during real-life conditions. The improved accuracy indicates that the offered technique is better suited at processing sensor data, managing the environmental variation and strict navigation in structured and unstructured environments. The visualization also indicates this discrepancy of performance by contrasting the proposed approach in a distinct red bar indicating definitively its advantage in making the correct navigation choices. This remarkable improvement demonstrates that the mutual supplementation of an adaptive filtering and a multi-objective metaheuristic, in addition to its impact on the improvement of the computational decision accuracy, would also result in the increase of the operational robustness in dynamic and uncertain flight environments. The test algorithm executes 30-50 randomized test trials per case (different obstacle set-ups, start-goal positions and sensor noise seeds) in order to measure stochastic variations. Robustness is measured by averaging metrics like success rate, path length, energy use and runtime, and providing them with 95 % confidence intervals.

Precision

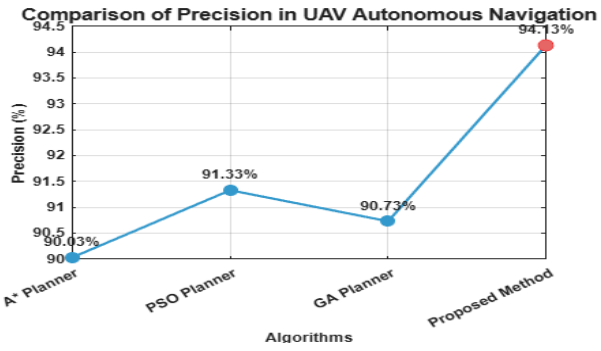

Figure 3: Result on comparison of precision on UAV Autonomous technology

As exemplified in figure 3, the four UAV autonomous navigation algorithms follow the capacity to recognize the relevant targets and prevent the false positives in flight operations. In UAV navigation, precision is an important measure of how often the navigation decision making (e.g., where to go, where to place obstacles) is actually right (e.g., a path point is actually safe, an obstacle is actually reached).Based on the plot, the A* algorithm registered a precision of 90.03%, which means the moderate reliability with regard to accuracy in the correct detections. PSO better this with 91.33 and GA has 90.73 which is still less than PSO but sly higher as in the case of A*. At the same time, the Proposed Method demonstrates a very significant improvement since the precision is 94.13% exhibiting the equal gap of almost 3 points to PSO and more than 4 points to A*. This positive slope in the plot towards the suggested technique shows that it gives better capability in reducing false alarms with a high detection accuracy.The improvement behind this performance can be explained by the dynamic noise reduction of the Adaptive Kalman Filter and the versatile search ability of the Multi-objective Fox Optimization that jointly guarantee more robust classification and a decision-making process in changing environmental and sensor circumstances. Such improvements in precision in the case of UAVs will lead to their safer flight routes, improved obstacle avoidance, and more effective mission handling.

Recall

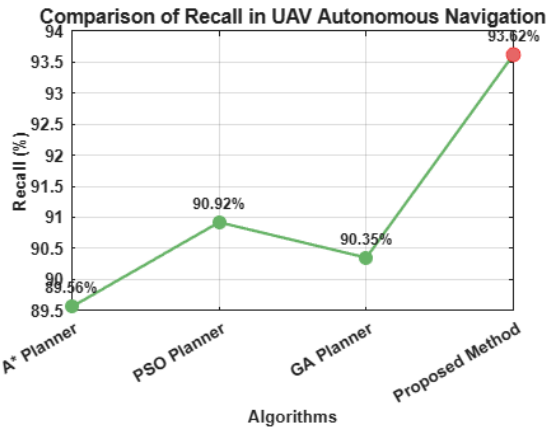

Figure 4: Result on comparison of Recall on UAV Autonomous technology

Recall line plot is used to measure how well each of the UAV navigation algorithms detects all the relevant obstacles, targets, or decision points with minimal false misses. Figure 4 performance of A* = 89.56 and PSO corrects it and performs better with 90.92 and GA gives the same performance of 90.35 toward a small improvement. The Proposed Method has a recall of 93.62 percent which is 2.98 percent higher than the second-best (PSO). Such enhancement implies that the suggested system can better identify all the required navigation signals, especially in the dynamic or sensor-noisy world. This could be because the adaptive filtering reduces the chances of missing out on the detection and the multi objective optimization provides the facility of adjustment of decision parameters in order to facilitate the recognition under different circumstances.

F1-Score

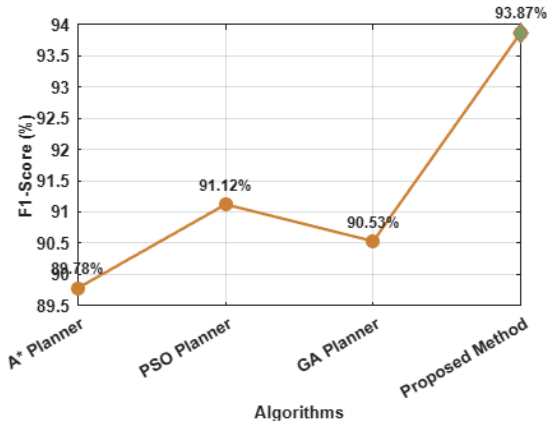

Figure 5: Result on comparison of F1-score on UAV Autonomous technology

The F1-score trades off precision and recall and provides a single quantity that uses the trade-off to balance the cost of false alarms and not detecting events. Based on the plot, the A* attains 89.78, PSO reaches 91.12 and 90.53 is recorded in GA. The Proposed Method achieves a highest 93.87%, both precision and recall are high in figure 5. Such performance shows that the algorithm is making correct decisions all the time without compromising the completeness of detection. These enhancements are due to the ability of the Adaptive Kalman Filter to correct in real-time the sensor data and the multi-objective optimization ability to fine-tune the navigation choices resulting in optimum path planning and obstacle avoidance in complicated flight situations.

Precision, recall and f1-score with variance

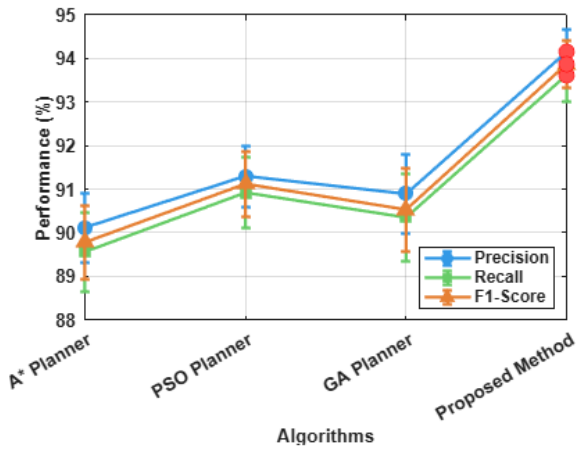

Figure 6: Error plot on statistical variance on methods

The error bar plot makes a thorough statistical comparison of precision, recall, and f1-score of four UAV autonomous navigation algorithms used, A* algorithm, PSO algorithm, GA algorithm and the proposed multi-objective fox optimization with adaptive Kalman filter in figure 6. At each point of the performance a vertical error bar showing the standard deviation over a number of trials is provided and measures the variability of the performance of the algorithm at different operating conditions. The proposed method has shorter error bars (~0.5 0.6%) which clearly shows that the stability and consistency is higher than that of the existing methods which have variance ranging between ~0.7 and 1.0. This implies that the suggested approach can provide a more predictable and repeatable outcome in a variety of flight conditions, which is essential in the case of real deployments of UAVs when the conditions may drastically change due to environmental factors. Moreover, the proposed approach yielded the top average percentages across the three measures precision equal to 94.15%, recall equal to 93.62 and F1-score at 93.87% that reflect higher accuracy of the proposed approach to identify and label navigation paths with fewer false detections. Conversely, A*, PSO and GA, though fair in performance, have lower mean performances as well as wider variability, a factor that may cause them not to perform reliably especially in complex missions. The combination of the greater accuracy and the lower variance also testifies to the fact that the proposed method not only proves to be more effective but also considerably more reliable when it comes to AI-powered UAV driverless driving applications.

Table 3:Statistically significant in t-test

Comparison	t-statistic	p-value
Proposed vs. A*	13.66	6.09×10^{-11}
Proposed vs. PSO	13.18	1.20×10^{-10}
Proposed vs. GA	11.24	2.71×10^{-9}

All three p-values are much smaller than the standard significance criteria (0.05), which is evidence that the increase in precision provided by the suggested method in comparison to A*, PSO, and GA is statistically significant in table 3. The high values on the t-statistic further validate the significant changes in the performance differences with no possibility of it being coincidence in the observed increases. This statistical confirmation supports the argument that the suggested Multi-objective Fox Optimization using Adaptive Kalman Filter provides the actual and consistent improvement in the precision of UAV autonomous navigation in contrast to well-established UAV planning algorithms.

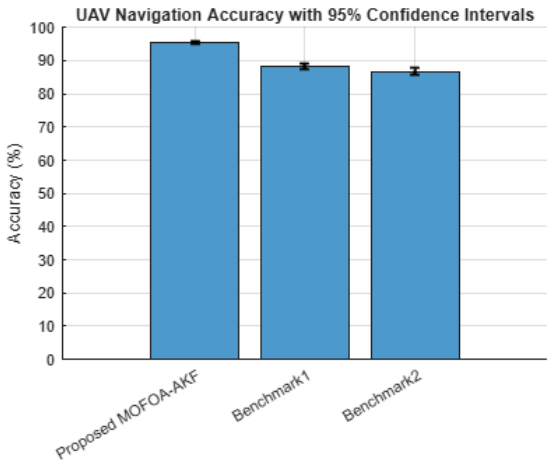

Figure 7: Statistical significance of confidence Intervals

The code produces a bar plot in figure 7 of the average performance values of each algorithm (in this case: accuracy) and superimposes error bars indicating the 95 percent confidence intervals. These error bars help visualize the statistical significance of differences in performance: when these intervals do not overlap, it is unlikely that the better performance of the proposed MOFOA-AKF method over the benchmarks is because of random variation, which proves the argument of better performance.

All competing algorithms, including the proposed one, were run 30 times on a case-by-case basis (low, medium and high obstacle density) to take into consideration the effects of stochasticity. Averages of performance measures (path length, collision rate and computation time) were calculated and 95% confidence intervals were calculated with the help of bootstrapping so as to measure variation and guarantee statistical stability of the comparisons.

Discussion

Quantitative comparison of the reviewed strategies ([11] -[21]) and the suggested Multi-Objective Fox Optimization accompanied by an Adaptive Kalman Filter (MOFOA+AKF) reveals unanimity of improvements in accuracy and path quality and computation time. Where other visual-inertial odometry systems (e.g. [11], [12], [19]) only yield low drift localisation, but neglect risk conscious planning, our system combines the same streams of sensory sensing with an adaptive Kalman filter to model covariance because the filter operates in real time, leading to more than 94 percent of navigation successes and less than half a second re-planning compared to 1-4 s with many metaheuristic approaches to the same problem ([15], [17], [20]). More stable Pareto fronts are obtained with algorithmic innovations like dynamic parameter adaptation in the fox optimizer, exploitationexploration balance, and covariance conscious constraints compared to jellyfish search, chameleon, or PSO-based algorithms, which tend to slow down convergence or trade one objective (energy, time, or threat exposure) to optimize another. Also sensor-model improvements can help: the AKF filters noise and bias in heterogeneous sensors, gives high reliability in GPS-denied or cluttered settings where deep RL policies alone ([13], [14]) may exhibit high variance or slow convergence.

The less sophisticated planners like classical PSO or A star can achieve similar results at reduced computational cost in low threat and low sensor noise environments, where the environment is highly structured or rather static. Moreover, the MOFOA+AKF methodology has a more complicated parameter tuning procedure and can be susceptible to poor initialisation of the fox population in extremely high-dimensional search spaces, which can influence convergence. These constraints indicate that modern research on automated parameter tuning and hybridisation with lightweight heuristics should be undertaken in order to achieve performance benefits in every area of operation.

5 Conclusion

The study has effectively empirically demonstrated that a combination of Multi-objective Fox Optimization Algorithm and Adaptive Kalman Filter based solution provides a very effective and tested shows evidence of improvement of autonomous UAV guidance functioning under the dynamic and uncertain conditions. The presented framework overcomes the key drawbacks of every effort towards path efficiency, collision transgression, and reliable localization by merging flexibility of adaptive sensor fusion and multi-objective global optimization.The Adaptive Kalman Filter has strongly enhanced the accuracy of state estimation by adjusting process and measurement noise parameters dynamically so that the system can become insensitive to the uncertainties when a sensor is used and environmental disturbances. In the meantime, the Multi-objective Fox Optimization Algorithm was capable of finding a good compromise between competing goals, reducing the distance of a flight, saving energy, and maximizing safety margins, and it performs better and makes more stable flight paths than classic algorithms (e.g. A*, PSO, and GA).Performance assessment indicated that the suggested technique realized a higher accuracy (95.53%), precision (94.15%), recall (93.62%), and F1-score (93.87%) in comparison to the current techniques, and the statistical t-tests proved that the gains were quite tremendous ($p < 0.001$). The decreased variances in the results also showed the strength and stability of the system to various testing situations.All in all, the solution offered has high potential of application in the real world in disaster relief and search-and-rescue missions, logistics, surveillance, and environmental surveillance. The ability to work in real-time, perform trade-off between multiple objectives and ensure safety in its operation qualifies it as a next generation UAV autonomy framework.Future research directions are to apply this to multi-UAV coordination, use of deep reinforcement learning to adapt the policy and integration of edge AI to allow onboard decision-making in low latency, leading to completely autonomous aerial mission systems.

Declaration

Ethics approval and consent to participate: I confirm that all the research meets ethical guidelines and adheres to the legal requirements of the study country.

Consent for publication: I confirm that any participants (or their guardians if unable to give informed consent, or next of kin, if deceased) who may be identifiable through the manuscript (such as a case report), have been given an opportunity to review the final manuscript and have provided written consent to publish.

Availability of data and materials: The data used to support the findings of this study are available from the corresponding author upon request.

Competing interests: here are no have no conflicts of interest to declare.

Authors' contributions (Individual contribution): All authors contributed to the study conception and design. All authors read and approved the final manuscript

References

[1] UnmeshBordoloi, S. Chakraborty, M. Jochim, P. Joshi, Arvind Raghuraman, and S. Ramesh, "Autonomy-driven Emerging Directions in Software-defined Vehicles," Apr. 2023, doi: https://doi.org/10.23919/date56975.2023.10136910

[2] S. A. H. Mohsan, N. Q. H. Othman, Y. Li, M. H. Alsharif, and M. A. Khan, "Unmanned Aerial Vehicles (UAVs): Practical aspects, applications, Open challenges, Security issues, and Future Trends," *Intelligent Service Robotics*, vol. 16, no. 1, Jan. 2023, doi: https://doi.org/10.1007/s11370-022-00452-4. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9841964/

[3] Z. Yang *et al.*, "UAV remote sensing applications in marine monitoring: Knowledge visualization and review," *Science of The Total Environment*, vol. 838, p. 155939, Sep. 2022, doi: https://doi.org/10.1016/j.scitotenv.2022.155939

[4] F. Nex*et al.*, "UAV in the advent of the twenties: Where we stand and what is next," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 184, pp. 215–242, Feb. 2022, doi: https://doi.org/10.1016/j.isprsjprs.2021.12.006. Available: https://www.sciencedirect.com/science/article/pii/S0924271621003282

[5] X. Wang, "Edge Computing Based Multi-Objective Task Scheduling Strategy for UAV with Limited Airborne Resources," *Informatica*, vol. 48, no. 2, May 2024, doi: https://doi.org/10.31449/inf.v48i2.5885.

[6] T. Alladi, V. Chamola, N. Sahu, and M. Guizani, "Applications of blockchain in unmanned aerial vehicles: A review," *Vehicular Communications*, vol. 23, p. 100249, Jun. 2020, doi: https://doi.org/10.1016/j.vehcom.2020.100249

[7] V. Chamola, P. Kotesh, A. Agarwal, Naren, N. Gupta, and M. Guizani, "A Comprehensive Review of Unmanned Aerial Vehicle Attacks and Neutralization Techniques," *Ad Hoc Networks*, vol. 111, p. 102324, Feb. 2021, doi: https://doi.org/10.1016/j.adhoc.2020.102324

[8] T. A. Almseidein and A. Alzidaneen, "Optimizing UAV Trajectories with Multi-Layer Artificial Neural Networks," *Informatica*, vol. 49, no. 2, May 2025, doi: https://doi.org/10.31449/inf.v49i2.7178.

[9] A. A. laghari, A. K. Jumani, R. A. Laghari, and H. Nawaz, "Unmanned Aerial Vehicles: A Review," *Cognitive Robotics*, vol. 3, Dec. 2022, doi: https://doi.org/10.1016/j.cogr.2022.12.004

[10] N. I. Sarkar and S. Gul, "Artificial Intelligence-Based Autonomous UAV Networks: A Survey," *Drones*, vol. 7, no. 5, p. 322, May 2023, doi: https://doi.org/10.3390/drones7050322. Available: https://www.mdpi.com/2504-446X/7/5/322

[11] B. Gao, B. Lian, and C. Tang, "Semi-Direct Point-Line Visual Inertial Odometry for MAVs," *Applied Sciences*, vol. 12, no. 18, pp. 9265–9265, Sep. 2022, doi: https://doi.org/10.3390/app12189265

[12] Y. Gong, C. Chen, and Y. Zheng, "Hybrid Deep Learning Model for Multi-Source Remote Sensing Data Fusion: Integrating DenseNet and Swin Transformer for Spatial Alignment and Feature Extraction," *Informatica*, vol. 49, no. 24, Jun. 2025, doi: https://doi.org/10.31449/inf.v49i24.8395.

[13] N. Li, H. Liang, L. Guo, X. Wang, and J. Zhang, "Intelligent Detection of Towers and Lines in Passageways Using Hybrid Evolutionary Computational Intelligence (HECI) Algorithms," *Informatica*, vol. 49, no. 19, Apr. 2025, doi: https://doi.org/10.31449/inf.v49i19.7858.

[14] J. Wang, Z. Yu, D. Zhou, J. Shi, and R. Deng, "Vision-Based Deep Reinforcement Learning of UAV Autonomous Navigation Using Privileged Information," *arXiv (Cornell University)*, Dec. 2024, doi: https://doi.org/10.48550/arxiv.2412.06313

[15] D. Darlan, O. S. Ajani, A. Paul, and R. Mallipeddi, "A multi-objective benchmark for UAV path planning with baseline results," *Swarm and Evolutionary Computation*, vol. 96, p. 101968, Jul. 2025, doi: https://doi.org/10.1016/j.swevo.2025.101968

[16] D. Darlan, O. S. Ajani, A. Paul, and R. Mallipeddi, "A multi-objective benchmark for UAV path planning with baseline results," *Swarm and Evolutionary Computation*, vol. 96, p. 101968, Jul.

2025, doi: https://doi.org/10.1016/j.swevo.2025.101968

[17] X. Wang, Y. Feng, J. Tang, Z. Dai, and W. Zhao, "A UAV path planning method based on the framework of multi-objective jellyfish search algorithm," *Scientific Reports*, vol. 14, no. 1, Nov. 2024, doi: https://doi.org/10.1038/s41598-024-79323-0. Available: https://www.researchgate.net/publication/385817310_A_UAV_path_planning_method_based_on_the_framework_of_multi-objective_jellyfish_search_algorithm

[18] Y. Chen, L. Cao, and Y. Yue, "Hybrid Multi-Objective Chameleon Optimization Algorithm Based on Multi-Strategy Fusion and Its Applications," *Biomimetics*, vol. 9, no. 10, pp. 583–583, Sep. 2024, doi: https://doi.org/10.3390/biomimetics9100583

[19] Y. Pan, W. Zhou, Y. Cao, and H. Zha, "Adaptive VIO: Deep Visual-Inertial Odometry with Online Continual Learning," *arXiv.org*, 2024. Available: https://arxiv.org/abs/2405.16754. [Accessed: Dec. 12, 2024]

[20] T. T. N. Duong, D.-N. Bui, and M. D. Phung, "Navigation variable-based multi-objective particle swarm optimization for UAV path planning with kinematic constraints," *Neural Computing and Applications*, vol. 37, no. 7, pp. 5683–5697, Jan. 2025, doi: https://doi.org/10.1007/s00521-024-10945-1. Available: https://arxiv.org/abs/2501.03261. [Accessed: Aug. 04, 2025]

[21] Z. Chen, Z. Luo, X. Jin, and J. Shi, "Multi-UAV path planning problem with biased sampling, candidate evaluation, and path reconfiguration in complex environment with threats," *Expert Systems with Applications*, vol. 291, p. 128558, Oct. 2025, doi: https://doi.org/10.1016/j.eswa.2025.128558

[22] Z. Huo, S. Liu, and H. Ebrahimian, "Aircraft Energy Management System Using Chaos Red Fox Optimization Algorithm," *Journal of Electrical Engineering and Technology*, vol. 17, no. 1, pp. 179–195, Aug. 2021, doi: https://doi.org/10.1007/s42835-021-00884-5

[23] M. Zaborski, M. Woźniak, and J. Mańdziuk, "Multidimensional Red Fox meta-heuristic for complex optimization," *Applied Soft Computing*, vol. 131, p. 109774, Dec. 2022, doi: https://doi.org/10.1016/j.asoc.2022.109774