The FAA's Center of Excellence for UAS Research
ASSURE
Alliance for System Safety of UAS through Research Excellence

A51_A11L.UAS.92 – Best Engineering Practices for Automated Systems: Final Report

June 30, 2025

v 1.1

TECHNICAL REPORT DOCUMENTATION PAGE

1. Report No. A11L.UAS.92_A51	**2. Government Accession No.**	**3. Recipient's Catalog No.**
4. Title and Subtitle Best Engineering Practices for Automated Systems: Report		**5. Report Date** June 30, 2025 **6. Performing Organization Code**
7. Author(s) Houssam Abbas, Ph.D. https://orcid.org/0000-0002-8096-2618 Rakesh Bobba, Ph.D. https://orcid.org/0000-0002-5440-0785 Arun Natarajan, Ph.D. https://orcid.org/0000-0002-1613-7007 Jinsub Kim, Ph.D. https://orcid.org/0000-0002-0119-8632 Phil Smith, Ph.D. https://orcid.org/0000-0003-1965-3717 Steven Weber, Ph.D. https://orcid.org/0000-0002-9235-6922 David Han, Ph.D. https://orcid.org/0000-0001-6437-3007 Shawn Keshmiri, Ph.D. https://orcid.org/0000-0002-7371-1255 Mark Ewing, Ph.D. https://orcid.org/0000-0001-5982-6219 Mark Askelson, Ph.D. https://orcid.org/0000-0002-8521-7158		**8. Performing Organization Report No.**
9. Performing Organization Name and Address Oregon State University University of North Dakota Drexel University University of Kansas Ohio State University		**10. Work Unit No.** **11. Contract or Grant No.**
12. Sponsoring Agency Name and Address U.S. Department of Transportation Federal Aviation Administration Office of Aviation Research Washington, DC 20591		**13. Type of Report and Period Covered** Final Report **14. Sponsoring Agency Code** 5401
15. Supplementary Notes		

16. Abstract

Data on UAS hazards and accidents is scarce, so the team recommend a low-data probabilistic risk assessment method before certifying or authorizing a concept of operations. UAS in urban areas face (micro) climate phenomena that are not as well researched as phenomena facing general aviation, so the team present urban weather models and their effects on UAS flight characteristics and safety separation for informing separation guidance. Key findings indicate that extreme wind shear, turbulence, and vorticity significantly degrade flight performance and aircraft maneuverability, particularly during low-altitude operations in urban environments, calling for investment in advanced flight control systems. UAS density is expected to far exceed general aviation density, so the team propose a balanced or fair method for small UAS air traffic control and motion planning to inform guidance development around maximum allowed densities and flight space partitioning, including around vertiports, and to inform guidance around rules-of-the-road for UAS. UAS use easily available off-the-shelf sensors that are relatively easily spoofed, so the team demonstrate and recommend a watermarking strategy as baseline for spoofing-resilient state estimation. Because UAS face payload restrictions and often use not-yet-optimized-for-UAS sensors, a detect-and-avoid architecture is explored which shares the burden of localization between the UAS and a broader, networked sensing environment which includes Global Navigation Satellite Systems, signal aping techniques, reconfigurable intelligent surfaces and received signal strength-based localization. Empirical results establish object detection performance at various distances to the obstacle, to inform guidance around camera resolution for UAS; the team demonstrate an image quality metric that can inform guidance around added controller safety margins in various vision conditions. The effects of cyber-attacks become significantly more pronounced in swarm or multi-UAS settings, so the team conduct an empirical analysis of those effects, and practical swarm parameter tuning mechanisms, to inform guidance around secure swarm deployments.

17. Key Words UAS; risk assessment; traffic management; motion planning; flight control; urban weather; micro-climate; runtime monitoring; sensors; localization; detect-and-avoid; cyber-physical security; sensor spoofing; swarm; computer vision; object detection; image quality.	**18. Distribution Statement** No restrictions.

19. Security Classification (of this report) Unclassified	**20. Security Classification (of this page)** Unclassified	**21. No. of Pages** 185	**22. Price**

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

TABLE OF ACRONYMS

6-DOF	Six Degrees-of-Freedom
ADS-B	Automatic Dependent Surveillance - Broadcast
AFTM	Air Flow Traffic Management
AI	Artificial Intelligence
AP	Average Precision
AR	Average Recall
BRISQUE	Blind/Referenceless Image Spatial Quality Evaluator
BVLOS	Beyond Visual Line-of-Sight
CA	Collision Avoidance
CBF	Control Barrier Function
CE	Cross-Entropy
CFD	Computational Fluid Dynamics
CLF	Control Lyapunov Function
CLIP	Contrastive Language Image Pretraining
CMA-ES	Covariance Matrix Adaptation - Evolution Strategy
CONOPs	Concept of Operations
CycleGAN	Cycle Consistency Generative Adversarial Network
DAA	Detect and Avoid
DBCNN	Deep Bilinear Convolutional Neural Network
DETR	Detection Transformers
DoS	Denial of Service
DoT	Department of Transportation
D-swarm	Decentralized swarm
EKF	Extended Kalman Filter
eVTOL	electric Vertical Take-Off and Landing
FAA	Federal Aviation Administration
FMEA	Failure Modes and Effect Analysis
FDI	Fault Detection and Isolation
FiReFly	Fair Distributed Receding Horizon Planning for Flying robots
FPR	False-Positive Rate
FTA	Fault Tree Analysis
GAN	Generative Adversarial Network
GNC	Guidance, Navigation, and Control
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HIL	Hardware-in-the-Loop
IMU	Inertial Measurement Unit
IQA	Image Quality Assessment
LiDAR	Light Detection and Ranging
LOC	Loss of Control
LoRa	Long Range

LQR	Linear Quadratic Regulator
LRDD	Long Range Drone Detection
LSTM	Long Short-Term Memory
LTI	Linear Time Invariant
mAP	mean Average Precision
MEMS	Micro-Electro-Mechanical Systems
MILP	Mixed Integer Linear Program
MIMO	Multiple Input Multiple Output
ML	Machine Learning
MTBF	Mean Time Between Failures
PFTC	Passive Fault Tolerant Control
PID	Proportional–Integral–Derivative
PRA	Probabilistic Risk Assessment
PSNR	Peak Signal Noise Ratio
PRREACH	Probabilistic Risk Analysis Using Reachability Analysis
PSU	Provider of Service to the UAM
RF	Radio frequency
RIS	Reconfigurable Intelligent Systems
RL	Reinforcement Learning
RMSE	Root-Mean-Squared-Error
ROC	Receiver-Operating-Characteristic
RSS	Received Signal Strength
SAHI	Slicing Aided Hyper Inference
SIL	Software-in-the-Loop
SME	Subject Matter Expert
SMS	Safety Management System
SNR	Signal-to-Noise Ratio
SRM	Safety Risk Management
SRMP	Safety Risk Management Policy
SSIM	Structural Similarity Index Measure
STAMP	System Theoretic Accident Model Processes
STL	Signal Temporal Logic
sUAS	Small Unmanned Aircraft System
TDC	Total Delay Cost
TPR	True-Positive Rate
UAM	Urban Air Mobility
UAS	Unmanned Aircraft Systems
UE	Unreal Engine
UWB	Ultra-Wideband
YOLO	You Only Look Once

EXECUTIVE SUMMARY

This research establishes empirical and theoretical foundations to inform Federal Aviation Administration (FAA) guidance and policy development for Unmanned Aircraft Systems (UAS) operations in the national airspace, particularly addressing the avoidance and mitigation of automation failures. Key focus areas include probabilistic risk assessment, urban weather modeling, balanced air traffic management and motion planning, advanced flight control, visual perception for detect-and-avoid, sensors and infrastructure for localization, robust inference and state estimation, runtime monitoring, and cyber-physical security. Each area offers evidence-based insights to support the formulation of FAA guidance and modular tools applicable across UAS configurations and operational concepts.

Given the limited availability of comprehensive UAS hazard and accident data, the team demonstrates a low-data probabilistic risk assessment framework that could support FAA safety case evaluations and pre-certification screening of novel concepts of operations. The team recommends that a tool such as PRREACH become a standard part of both FAA evaluations and operator-submitted assurance packages for autonomous UAS approvals, especially in the absence of large-scale accident databases.

Recognizing the unique urban microclimates UAS must navigate, the research presents new urban weather models and associated flight behavior data. These results should inform FAA guidance on safety separation standards and environmental performance thresholds for urban UAS operations. Simulations indicate that horizontal standoff distances from building corners, smooth arc trajectories instead of 90-degree turns, and pre-flight wind assessments using hyperlocal sensors are critical. These strategies align with FAA urban integration goals and support more robust, real-time UAS decision-making tools. Additionally, FAA certification of flight control systems should incorporate CFD-based simulations of control performance in uncertain wind environments, particularly near landing zones.

To address the anticipated high-density UAS traffic in urban areas, the team proposes a balanced or fair method for small UAS air traffic control and motion planning to inform guidance development around maximum allowed densities and flight space partitioning, including around vertiports, and to inform guidance around rules-of-the-road for UAS. The proposed framework shows how traffic deconfliction burdens can be distributed equitably across operators, which should be encouraged in FAA and UAM Service Provider policies. Operators should also be incentivized to adopt distributed planning architectures for local deconfliction in tightly clustered environments, though further investment is needed to understand infrastructure requirements and regulatory enablers.

Additionally, UAS use easily available off-the-shelf sensors that are relatively easily spoofed, so the team demonstrates and recommends a watermarking strategy as a baseline for spoofing-resilient state estimation. FAA and relevant authorities should also evaluate operator safety cases based on their resilience to a catalog of moderate and covert sensor spoofing attacks, as defined by the simulator developed in this effort.

Because UAS face payload restrictions and often use not-yet-optimized-for-UAS sensors, a detect-and-avoid architecture is explored which shares the burden of localization between the UAS and a broader, networked sensing environment which includes Global Navigation Satellite Systems, signal aping techniques, reconfigurable intelligent surfaces (RIS), and received signal strength-based localization. To reach grid-level safety in dense urban corridors, the research recommends pilot deployments of 3D airspace cells (e.g., $10m^3$ to $20m^3$) and standardized fault classification protocols, both of which could shape FAA infrastructure strategy and interagency coordination (e.g., with ICAO). Dynamic geofencing driven by AI-powered signal maps and multi-sensor mesh feedback loops should be considered for future FAA system-level safety assessments.

Empirical results on object detection performance under varying visual conditions can support FAA and industry standards guidance on minimum sensor resolution standards and appropriate controller safety buffers based on visibility metrics. Additionally, a threshold metric is proposed to determine when degraded perception warrants mission termination or rerouting, supporting vision-based safety assurance cases.

As multi-UAS and swarm operations become more prevalent, the research quantifies the impact of cyber-attacks in cooperative flight scenarios and explores tuning strategies to maintain swarm integrity. FAA guidance should reflect the increased vulnerability of swarms to cyber-physical interference and include standardized runtime monitoring requirements for on-board health checks and inter-UAS coordination. The team further recommends that runtime monitors be synthesized from formal specifications to avoid human coding errors which may help shape FAA positions on autonomy and real-time compliance monitoring. Certification approaches should evolve beyond traditional verifiability and instead reward demonstrated performance in simulation and test flights, especially for adaptive and ML-based controllers. A layered control approach could be incorporated into FAA review criteria for autonomy-driven control systems.

Collectively, the research outputs can help the FAA define performance-based safety margins, refine operating domains, establish approval pathways for new concepts of operations, and prioritize research needs related to robust UAS automation, particularly in urban and high-density environments. The research outputs also point to specific areas of investment for future policymaking and research.

To accommodate the numerous topics covered, the following A51 Final Report is structured as a chaptered report, with individual summaries and reference lists per unique topic.

1 INTRODUCTION

This report presents the results of Year 3 of ASSURE project A51. In Year 3, the research was focused on developing new guidance for highly automated and autonomous functionality in small Unmanned Aircraft Systems (UAS) and validating this guidance. The development of new guidance was informed by the automation failures found in Year 1 and the capability of state-of-the-art techniques at mitigating such failures, investigated in Year 2. The research covered all levels of the integration of UAS into the national airspace. The chapters can be read independently of each other, and the entire report can be read in a number of sequences.

2 FAIRNESS IN UAS OPERATIONS

In Urban Air Mobility (UAM) and other more general UAS Concept of Operations (CONOPs), there will be multiple approved UAS operations occurring simultaneously. In Urban Air Mobility (UAM) and other more general UAS Concept of Operations (CONOPs), there will be multiple approved UAS operations occurring simultaneously. The UAS share an airspace but might have competing interests: one UAS getting to its destination as quickly as possible might force others to hold or take longer routes, thus wasting time and energy. It is therefore important to account for some kind of fairness or balance in the UAS' traffic management and motion planning. This would discourage the cases where some operators may be directed to flight paths that are shorter or trajectories that require less energy, at the excessive expense of others, because such imbalance might decrease participation by smaller UAS stakeholders and put their operations at greater risk of running afoul of time or energy limitations. This chapter describes two methods for injecting fairness notions into UAM. The first is through high-level traffic management of UAS flight plans. The second is through multi-UAS motion planning and control. While our work focuses particularly on small-sized UAS, these methods are general to all aircraft, including larger passenger-carrying vehicles.

2.1 Executive Summary

The team demonstrates a negotiated framework for UAS traffic management around and between vertiports. In this framework, the provider of services to the UAM and the UAS operators collaborate seamlessly (and automatically) to plan routes that are safe, meet mission demands, and *distribute the cost of deconfliction fairly over the UAS.* The cost of deconfliction is measured in energy consumption, which is computed using a high-level generic approximation that is available to all participants (thus no knowledge of dynamics or internal operation is required). The team demonstrates feasibility in simulation for up to 50 UAS taking off every hour at 5-minute intervals, thus striking a balance between efficiency and computational feasibility with realistic take-off cadences.

For motion planning, the team demonstrates a fully distributed algorithm that the UAS can run to optimize motion plans subject to a fair or balanced distribution of (normalized) energy consumption. This motion planner can run in real-time for groups of up to 15 UAS, which is a reasonable limit on the number of directly interacting UAS.

Proposed Guidance: The team proposes that air traffic management near vertiports (whether by providers of services to the UAM or public authorities) distribute the cost of traffic deconfliction over all participants, to avoid inadvertently putting most of that cost on a few operators (or repeatedly on the same operators, which the team's research shows can easily happen). This chapter's framework demonstrates the feasibility of this approach in a negotiated setting, but more research is needed to reduce the involvement of the UAS operators further, in case the latter is a regulatory and technical bottleneck.

The team also proposes that UAS operators be incentivized to move towards a more distributed motion planning approach for directly interacting, small groups of UAS, to achieve the robustness benefits of distributed planning, including the maintenance of fair navigation. A distributed framework does place a greater burden on the communication infrastructure, so more research is needed to understand the practical requirements of such an infrastructure.

Prototype software for the methods developed here is available at:

FiReFly repo: https://github.com/sabotagelab/quadratic_dist_opt

Fair-CoPlan repo: https://github.com/sabotagelab/fair_uam

The figures below illustrate the guidance.

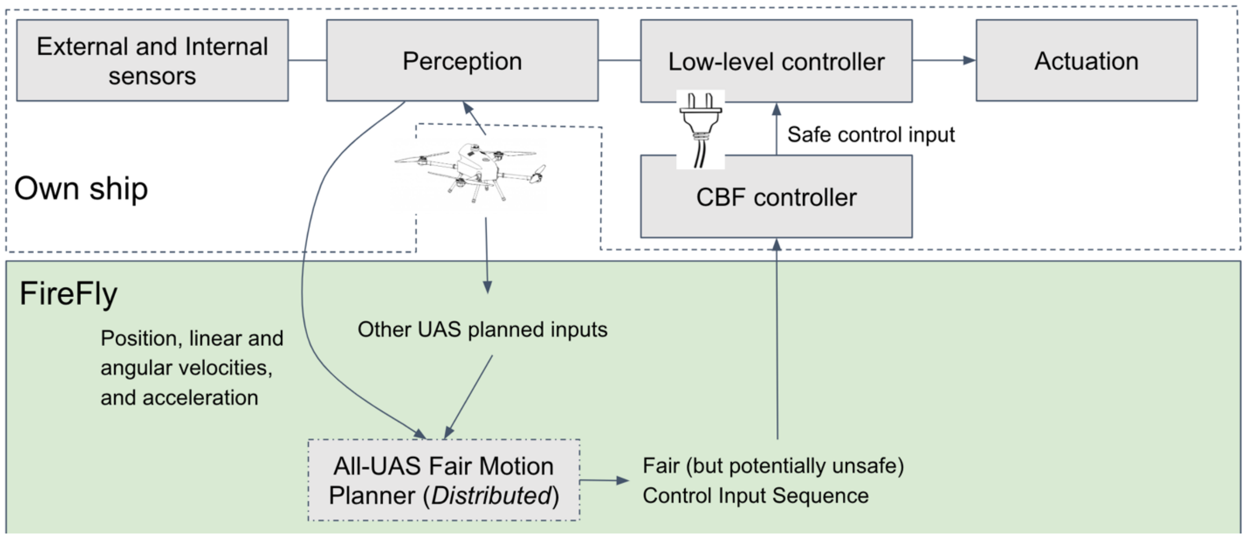

External and Internal sensors
Perception
Low-level controller
Actuation
Safe control input
CBF controller
Own ship
FireFly
Position, linear and angular velocities, and acceleration
Other UAS planned inputs
All-UAS Fair Motion Planner (Distributed)
Fair (but potentially unsafe) Control Input Sequence

Figure 1. Process diagram for FireFly guidance.

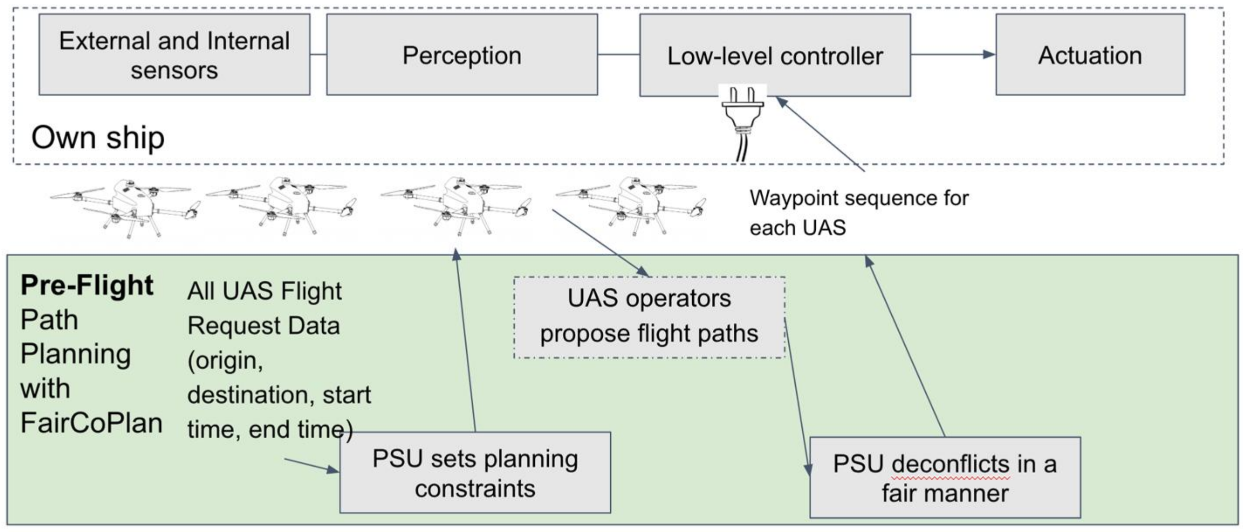

External and Internal sensors
Perception
Low-level controller
Actuation
Own ship
Waypoint sequence for each UAS
Pre-Flight Path Planning with FairCoPlan
All UAS Flight Request Data (origin, destination, start time, end time)
UAS operators propose flight paths
PSU sets planning constraints
PSU deconflicts in a fair manner

Figure 2. Process diagram for FairCoPlan.

2.2 Fair UAS Traffic Management

The team first addressed the problem of fair UAS traffic management. In traditional Air Flow Traffic Management (AFTM), strategic deconfliction is the process of advanced planning and coordination of multiple flight trajectories to prevent conflicts between aircraft. The Vision Concept of Operations proposed by the Federal Aviation Administration (FAA) outlines that UAM operators can develop and file their own operation plan, but then must conform to any changes by the Provider of Service to the UAM (PSU) for centralized strategic deconfliction (FAA 2020). This can result in sub-optimal plans for some flights and generally limits operator flexibility. Indeed, recent FAA reports show an increase in airspace authorization and waiver applications from UAS operators, indicating a desire for greater latitude in planning without always having to file formal requests with a central authority (FAA 2023a).

The team describes a prototype flight planning software that allows greater operator flexibility and optimizes for fair outcomes while maintaining a conflict-free airspace. This solution is called the Fair-CoPlan, and it proceeds in three steps:

1. The PSU constrains take-off and landing choices for flight operators based on current expected traffic at and around vertiports.
2. Operators propose flight plans based on the constraints given by the PSU and known occupancies of en-route flight sectors.
3. The PSU fairly deconflicts the proposed flight trajectories.

The team focused on fairness in terms of change in path length as a result of strategic deconfliction.

2.2.1 Related Work

The team's work builds on the integer programming approach for AFTM first presented in (Bertsimas and Patterson 1998), and expanded upon in (Barnhart et al. 2012; Bertsimas, Lulli, and

Odoni 2011; Chandran and Balakrishnan 2017). In this approach, the flow of air traffic is centrally controlled by adjusting release times to the network or subsequent air sectors in a flight's path. The final flight paths are inflexible to operators. Fair-CoPlan differs in that operators submit their preferred flight paths given centrally provided constraints, and these are fairly adjusted only when conflicts arise. In a distributed approach presented in (Chin, Li, and Pant 2022), the airspace is split into separate regions, and the AFTM problem is solved locally within these regions, with information exchanged between them as necessary. To contrast, Fair-CoPlan considers the airspace as a single region, and path planning for individual flights is handed off to their operators.

The problem of optimizing fair path lengths in motion plans of a single group of UAS is introduced in (Kurtz and Abbas 2020) and reformulated as a distributed Mixed Integer Program in (Brahmbhatt and Oregon State University 2023). Fair-CoPlan also optimizes fair path lengths for single groups of UAS at a time but considers the changing state of the airspace at every new time period and does this through a combined central and distributed Mixed Integer Linear Program (MILP) solution.

Fair path lengths are typically not considered in AFTM formulations. More common notions of fairness surround flight scheduling and include minimization of reversals and overtakings (Bertsimas and Gupta 2016; Chin et al. 2021), and minimal time order deviation (Barnhart et al. 2012). Other works consider fair allocation of departure time slots based on original flight schedules (Mercedes Pelegrín and Hamadi 2023; Vossen et al. 2003). Towards fairness in path planning, the work in (Tang et al. 2021) optimizes for equity in cost reduction from a maximal flight cost. The team's fairness notion differs in that the team looks at fair change in cost from a proposed flight plan, rather than an assumed maximal cost.

2.2.2 Problem Overview

The team discretizes the airspace into a 2D grid. Each unit of the grid is called a *resource* that is either a vertiport or sector of airspace. A vertiport is a collective term for land area or structure designed for UAS operations (FAA 2020). Each sector has a maximum allowed number of UAS, or *capacity*, associated with it. Similarly, each vertiport has set departure and arrival capacities. The team assumes information about resource capacities and current active flights is made available through a publicly available flight database.

The team also assumes UAS can travel between sectors through designated corridors. Multiple UAS can be in the same sector, up to the maximum capacity for the sector, by maintaining safe nose-to-tail separation, or taking altitude-separated passing corridors.

Flight requests need to include departure vertiport and time, and arrival vertiport and time. The problem is to safely route all flights given capacity constraints and the current state of the airspace. Below is an overview of the team's solution:

1. The PSU offers all flight operators the following as choices from which they can plan their trajectories:
    a. Feasible departure times from the origin vertiport within a flexible time window.
    b. Feasible sectors adjacent to the departure vertiport, and times the flight is permitted in those sectors. These give direction of travel from the origin.

c. Feasible arrival times to the destination vertiport within the flexible time window.
d. Feasible arrival sectors adjacent to the arrival vertiport, and times the flight is permitted in them. These give approach direction to the destination.

Feasible sectors and times for departure/arrival are informed by previously approved flight plans that have been filed in the flight database. This initial step of assigning departure and arrival constraints is motivated by the fact that the areas of highest congestion, where risk of midair collision is greatest, are at vertiports and their surrounding areas (FAA 2023b). Thus, this step allows the PSU to pre-manage possible conflicts in these areas before operators formulate their preferred trajectories.

2. Operators propose flight plans by formulating trajectories given the choices provided by the PSU and known en-route occupancies and capacities of flight resources, given previously approved and published flight plans available in the database.
3. The PSU deconflicts the proposed flight trajectories. Because conflicts at and around vertiports have already been managed in Step 1, this step handles anticipated en-route conflicts. After ensuring no conflicts, flights are authorized and their plans filed in the database.

The outcome of Fair-CoPlan is not unlike that of the Flight Plan Routing module of NASA's Future Air Traffic Management Concepts Evaluation Tool (Bilimoria et al. 2001). Both solutions ultimately provide each operator a sequence of waypoints defining the route and include the times the flight may arrive and remain in a sector. The key difference is that in Fair-CoPlan, the route is devised as a *negotiated* solution between PSU and operator and gives greater flexibility to the operator without sacrificing safety of the airspace.

2.2.3 Problem Formulation

In this section, the team discusses the technical components for implementing Fair-CoPlan.

2.2.3.1 Input Data

Fair-CoPlan requires the following input data:

- The set of incoming flight requests F
- The number of discrete time intervals T over which to plan all incoming flight requests
- The set of all airspace resources R divided into:
    - The set of vertiports and each's capacity at each time step t
    - The set of sectors and each's capacity at each time step t
- An occupancy map that gives the number of UAS for each resource r at each time step t
- The minimum time each flight f must spend in each sector
- For each flight f, the maximum of time steps the flight allows for delay

2.2.3.2 Step 1: PSU Sets UAS Operator Planning Choices

This step is formulated and solved as an MILP with the following components:

- **Decision Variables**

    - $c_{r,t}^{f}$: a binary variable that equals 1 if the PSU decides to offer flight *f* the option of being in resource *r* at time *t*, and 0 otherwise
- **Objective**
    - The PSU maximizes the number of sector-time choices offered to all flight operators by maximizing the sum over all $c_{r,t}^{f}$
- **Constraints**
    - Ensure occupancy of any resource at any time does not exceed the capacity
    - Prevent flights from departing earlier or arriving later than their requested departure or arrival times
    - Prevent flights from operations outside of their requested departure or arrival times
    - Prevent flights from departing from their origin if sectors adjacent to the origin cannot be offered as a choice (due to potentially being at capacity)
    - Prevent flights from arriving at their destination if sectors adjacent to the destination cannot be offered as a choice (due to potentially being at capacity)
    - Ensure each flight can remain in an offered sector for at least the duration of the flight's determined minimum time it must spend in that sector

2.2.3.3 Step 2: Operators Plan Their Trajectory

Given the planning constraints computed by the PSU in Step 1, each operator designs their flight plan, which the team formulates as an MILP with the following components:

- **Decision Variables**
    - $u_{r,t}^{f}$: a binary variable that equals 1 if the operator of flight *f* designs their flight plan to arrive in resource *r* by time *t*, and 0 otherwise
- **Objective**
    - The operator minimizes Total Delay Cost (*TDC*) for their proposed flight, following the definition of total delay cost from (Bertsimas and Patterson 1998)
- **Constraints**
    - Follow the PSU restrictions on departure and arrival sectors and times imposed in Step 1
    - Select only one time slot for departure from the origin and only one time slot for arrival to the destination
    - Do not plan to be in a sector at a time when it is expected to be at capacity
    - Ensure that chosen sectors in sequence for the plan are adjacent to each other
    - Ensure only one sector is chosen per time step
    - Ensure that for any sector chosen in the plan, the flight spends the determined minimum time in must spend in that sector

2.2.3.4 Step 3: PSU Performs Fair Deconfliction

Given all operators' proposed plans, the PSU fairly adjusts the plans of any conflicting flights. This step is formulated and solved as a MILP with the following components:

- **Decision Variables**

    - $v^f_{r,t}$: a binary variable that equals 1 if the PSU directs the flight plan of flight *f* to arrive in resource *r* by time *t*, and 0 otherwise
- **Objective**
    - The PSU minimizes TDC over all flights and a fairness term F over all flights according to the formula $TDC + \gamma F$ where the parameter $\gamma > 0$ set by the PSU manages the trade-off between *F* and *TDC*
- **Fairness Term**
    - The fairness term *F* is included in the objective to encourage equal change in path lengths after deconfliction for all flights. The fairness term is defined by the equation $F = max \frac{L(v^f)}{L(u^f)} - min \frac{L(v^f)}{L(u^f)}$ where $L(v^f)$ represents the length of the flight path adjusted by the PSU and $L(u^f)$ is the length of the flight path proposed by the operator of flight *f* in Step 2. The ratio $\frac{L(v^f)}{L(u^f)}$ is the normalized change in path length of flight *f* and represents the "cost" of deconfliction imposed on flight *f*. Minimizing *F* results in similar costs distributed between flights.
- **Constraints**
    - The constraints in this step mirror those in Step 2, but applied over all considered flights

2.2.4 Fair-CoPlan Experimental Setup

The team evaluated the performance of Fair-CoPlan in a fixed airspace, and at increasing flight demands.

The simulated airspace covers 3600km2, which is comparable to the greater metro areas of Boston or Detroit. The airspace is discretized into a 15×15 grid. This is based on speeds of current top-end UAS, which can reach top cruising speeds of 112 km/hour. The team simulated 12 vertiports, four of which are central hub vertiports with departure and arrival capacities of 12 UAS per five minutes. The other eight are smaller vertiports, which are sometimes called vertistops (FAA 2020), with limited operations and smaller capacities of five UAS per five minutes. The team set $T = 18$ with each time step having duration of five minutes. Capacities of sectors adjacent to a vertiport are set to three, while all other sector capacities are set to one. For each flight *f* the team set the minimum time each it must spend in sectors adjacent to vertiports randomly to one or two. For all other sectors, the team set the minimum time each flight must spend in that sector to one. For each flight *f*, the team set the maximum of time steps allowed for delay to three.

The team created demand scenarios of 25, 36, and 50 flights per hour per hub vertiport. Under each of these demand scenarios, the team simulated 10 different multi-hour operating days in which Fair-CoPlan is run every five minutes to serve all incoming requests. Any flights not successfully planned in the planning period - due to infeasibility given the state of the airspace - update and resubmit their requests. These flights are processed first before the new incoming requests in the next planning period. Sample flight paths from a single simulation are shown in Figure 3. In the figure, flight paths (gray lines) generated by Fair-CoPlan under the 36 flights/hr/vertiport hub demand scenario for a 60km $\times$ 60km airspace. The larger green circles are the vertiport hubs, and the smaller blue circles are lower capacity vertistops. Darker lines indicate more frequented flight corridors.

In all simulated days, the team tested different values of γ in Step 3, and evaluated the average change in flight path length against TDC. The team compared these to results to a baseline where $\gamma = 0$. The team also compared Fair-CoPlan against the classical TFMP formulation in [14] in terms of average TDC per flight.

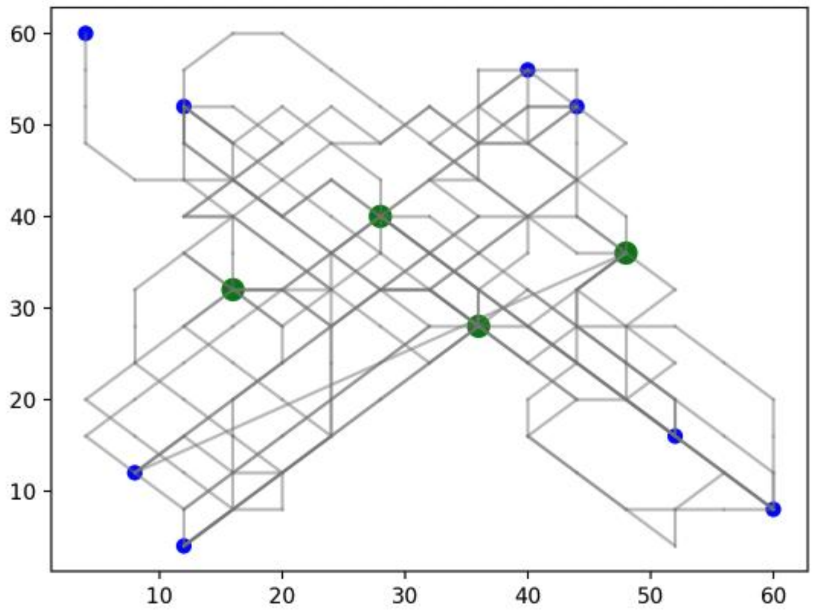


Figure 3. Sample simulated flight paths.

All steps were implemented in Python with each MILP solved using MOSEK version 10.2. Experiments were run on a computer with an 8-core 3.65Ghz processor and access to 32GB RAM.

2.2.5 Fair-CoPlan Results and Discussion

2.2.5.1 Shorter Delays than Classical TFMP

Fair-CoPlan overall yields shorter delays than a traditional approach, which does not consider fairness during planning. This is evidenced by comparing the distribution of average final TDC for Fair-CoPlan and the TFMP formulation in Figure 4. The distribution for Fair-CoPlan skew further left, thus having a lower average TDC than TFMP.

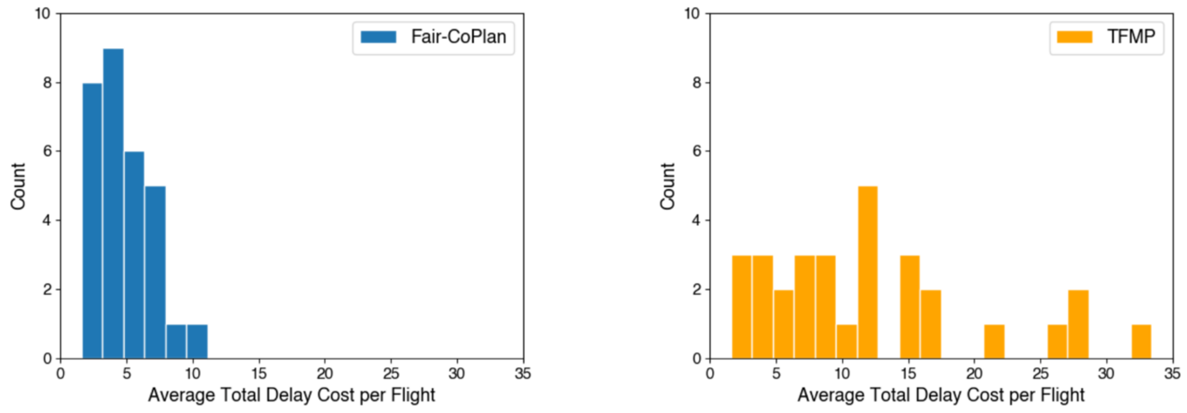


Figure 4. Distribution of Average TDC between Fair-CoPlan and TFMP.

2.2.5.2 Improved Fairness with Minor TDC Trade-offs

Almost all planning periods had some improvement in fairness for all $\gamma > 0$, with more than half of the periods also having an improvement in TDC. This is shown in Figure 5, which plots the difference in final TDC against the difference in fairness between the baseline (where $\gamma = 0$) and Fair-CoPlan with varying values of positive γ. In the figure, each point represents the result from a single planning period. Dashed gray lines indicate equal final TDC or fairness. Higher, positive values (for either fairness or TDC) indicate that Fair-CoPlan does better (in terms of fairness or TDC) than the baseline. In the few cases in which fairness decreases with Fair-CoPlan, this is because fairness is only considered between the rent flights being deconflicted, and these flights may be assigned routes through sectors in a way that unfairly blocks *future* flights. Figure 5 also shows that Fair-CoPlan is most effective for moderate and large demand scenarios in terms of efficiency trade-offs. For the 25 flights/hr scenario, 70-80% of planning periods result in a larger TDC with Fair-CoPlan over the baseline. The other demand scenarios result in larger TDC only 30-50% of the time. This is because with smaller demand, fewer conflicts arise, leading Fair-CoPlan to essentially over-optimize fairness. In low demand scenarios, lower values of γ can mitigate this.

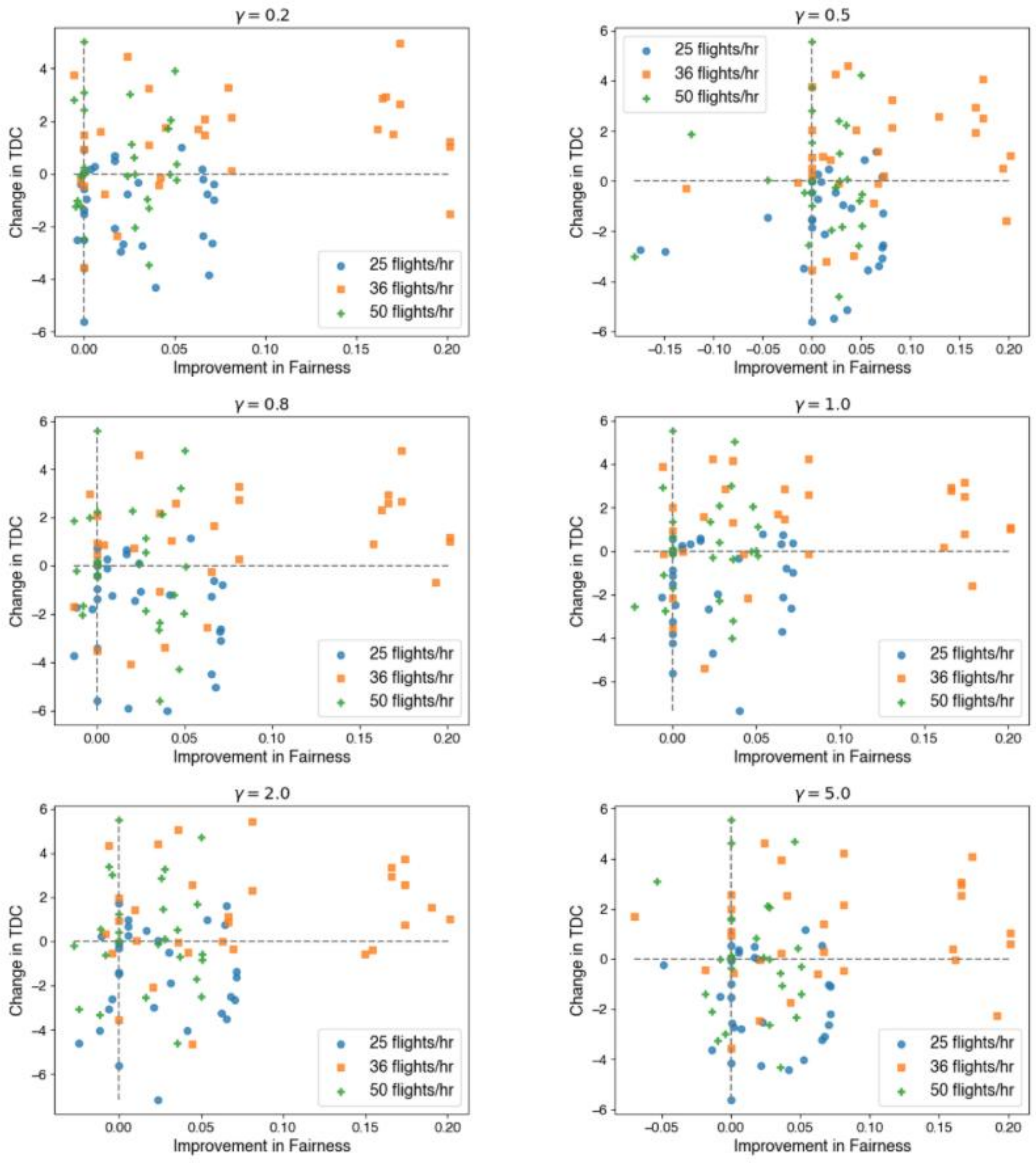


Figure 5. Difference in final TDC against difference in fairness between the baseline and Fair-CoPlan.

2.2.5.3 Operationally Efficient Runtimes

The time to construct and solve all steps of Fair-CoPlan took less than five minutes, with Step 3 being the most time-consuming, as shown in Table 1. This table shows that Fair-CoPlan can run at relatively high frequency and can process all flight requests before the next batch of requests in the following planning period. Note also that average solve times in Step 2 decrease with increasing flight demand, reducing computational burden on the individual operator.

Table 1. Solve times in seconds for Fair-CoPlan.

	Step 1 Solve Times				Step 2 Solve Times (per Operator)				Step 3 Solve Times			
	mean	std	min	max	mean	std	min	max	mean	std	min	max
25 flights/hr	8.1	0.8	7.0	9.3	24.3	6.5	18.6	38.9	84.4	34.1	37.2	136.5
36 flights/hr	12.3	2.0	9.5	16.0	22.5	4.5	17.5	32.8	104.1	47.1	41.9	177.8
50 flights/hr	26.3	4.7	19.2	33.8	15.4	1.4	12.7	17.5	69.1	41.9	31.4	149.1

2.3 Fair Motion Planning and Control for UAS

The team now addresses the problem of fair motion planning and control for UAS sharing a sector of airspace. Current motion planning approaches for multiple robots sharing the same space focus on mission success and safety. These approaches may produce plans that get a few robots to their destination quickly, but have others spend excessive energy loitering or navigating unnecessarily longer paths before reaching their target. This is clearly an unfair outcome for all but the single operator whose robot achieved greater energy efficiency. A fairer plan would distribute the energy expenditure more evenly between the robots, while still fulfilling the mission objectives, and without compromising other mission constraints.

The team presents a distributed motion planning algorithm that explicitly optimizes for fairness in normalized energy consumption across multiple UAS in the same airspace. A distributed algorithm is necessary in settings where a centralized planner is unavailable or has limited capacity, as in the case of untowered air sectors or busy airspaces that must prioritize traditional aircraft. The team integrates this fair motion planning algorithm into a framework that guarantees safe control, relying on Control Barrier Functions (CBFs) in a receding horizon fashion to ensure collision avoidance at every time step. The team calls this solution **FiReFly**, for **F**air Distributed **Re**ceding Horizon Planning for **Fly**ing robots. Example safe *and* fair FiReFly trajectories are shown in Figure 6.

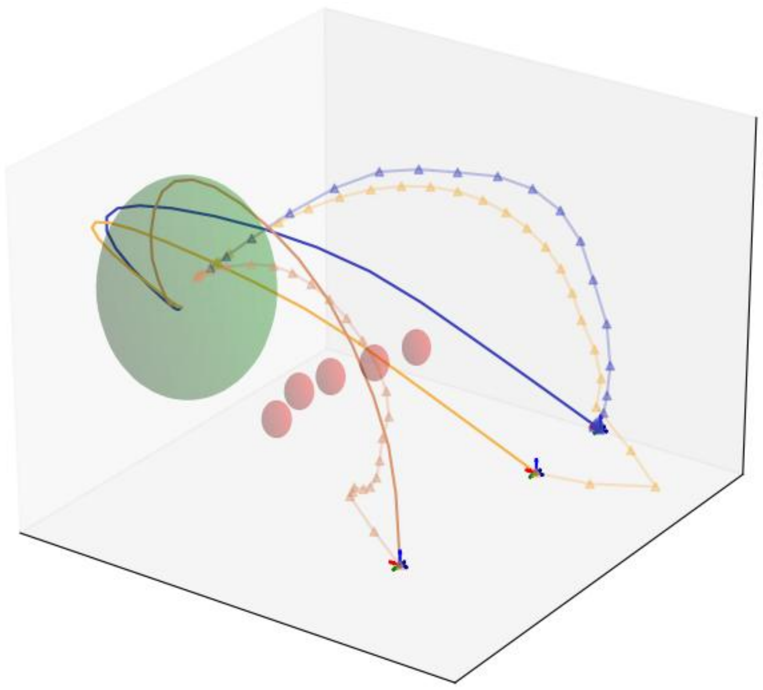


Figure 6. Trajectories for three UAS under FiReFly.

2.3.1 Related Work

An overview of general UAS path planning techniques is presented in the surveys (Israr et al. 2022; Rahman, Sarkar, and Lutui 2025), but these approaches do not consider any notions of fairness. FairFly (Kurtz and Abbas 2020) introduced an offline optimization method for fair UAS motion planning with fairness in terms of flight time. FiReFly is an online optimization for fairness in terms of energy consumption, which is not always proportional to flight time since it depends on flight characteristics. Safety in (Kurtz and Abbas 2020) is guaranteed offline via an expensive global optimization of robustness from (Pant et al. 2018). FiReFly guarantees safety online using CBFs, which have been shown to be capable of real-time control (Ames et al. 2017).

The work in (Brandão et al. 2020) presents different formalizations for fairness in robot planning in terms of locations visited or groups served, which are mathematically different from the team's defined notions for fairness in energy. Fair energy consumption for multi-UAS teams is explored in (Ji et al. 2020), but for multi-target positioning, and uses a different formulation for fairness.

Other energy-aware multi-UAS planners include those in (Buyukkocak, Aksaray, and Yazıcıoğlu 2023; Zhao et al. 2021), but these focus on total energy efficiency of a UAS team, and not fairness. These notions are not reducible to each other; plans that have low total energy can still be unfair to some agents, and vice versa. FiReFly looks at the energy consumption of individual UAS and attempts to distribute it between them equitably.

2.3.2 Problem Overview

2.3.2.1 System Model

The team considers a team of $N > 2$ UAS modeled with discrete-time double integrator dynamics. The team represents the control inputs of a single UAS k as $\boldsymbol{u_k}$, with a particular input at time t given by $\boldsymbol{u_k}[t]$.

2.3.2.2 Reach-Avoid Mission

Each UAS has a starting position and goal position. The UAS must maintain a safe distance between each other or risk collision. The mission space also includes static obstacles. In a Reach-Avoid mission, all UAS must reach their goal position by the end of their set time horizon while avoiding collisions with obstacles and each other.

2.3.2.3 Fairness

For UAS motion planning and control, fairness is a property of the set of control sequences of all N UAS, $\mathbf{u} = (\boldsymbol{u}_1, \dots, \boldsymbol{u_N})$.

2.3.2.4 Problem Statement

Given N UAS and a mission space with obstacles, compute N control sequences such that each UAS reaches its goal position within its time horizon without collisions, and such that fairness is optimized.

2.3.3 Solution Overview

Solving for fair and safe control inputs simultaneously in a single optimization would require satisfaction of multiple non-linear, non-convex constraints, which is computationally prohibitive. The team instead deals with satisfying linear, convex constraints by handling the fairness and safety problems separately. The team takes an iterative, receding horizon approach to solving the problem that proceeds in two steps. First, the team optimizes for fairness at time t, computing N control input sequences (one per UAS) such that each UAS reaches its goal position by the end of its time horizon, and such that fairness over the entire set of trajectories is optimized. Then, the team adjusts the fair input sequences, computing N one-step safe inputs for time t such that progress to the goal is maintained, and the difference between these safe inputs and the first inputs from the fair sequences is minimized. The idea is that the UAS execute safe control inputs at every time step and, because the process is iterative, reference inputs will always be updated to be as fair as possible given the history of inputs.

2.3.4 Fairness Notions

The team presents four fairness notions in terms of energy consumption. The first two notions are adapted from (Kurtz and Abbas 2020) which applied them to flight time as a resource.

2.3.4.1 Energy Variance

The first notion strives to ensure that each UAS consumes a similar amount of energy to complete its mission. Because different missions can require different amounts of energy to complete, it makes little sense to require that all UAS consume the same amount of energy. Therefore, the team uses normalized energies. The team denotes the energy that UAS k would consume if it were the single UAS in the airspace as $\underline{e}_k$. Then, the team defines the normalized energy e_k of UAS k as

$e_k = \frac{1}{\underline{e}_k}\sum_t \boldsymbol{u_k}[t]^2$. Normalized energy is always greater than or equal to one and represents the "energy cost" increase due to the presence of other UAS in the environment. The first fairness notion f_1 is the variance between the normalized energies, given by equation $f_1(u) = Var(e_1, \ldots, e_N)$. Perfect fairness is achieved when $f_1 = 0$, or in other words, when all UAS consume the exact same normalized energy. As such, optimizing for this notion of fairness means minimizing f_1.

Minimizing function f_1 alone might cause problematic behavior as it can force all UAS to consume the same normalized energy, even if there is a solution in which UAS can consume less energy without increasing the others' consumption. The team, therefore, introduces the following fairness notion: $f_2(u) = f_1(u) + \beta Q(u)$ where $Q(u)$ is a term representing the total control energy of the system and $\beta > 0$ is a hyperparameter that manages the trade-off between the energy term and f_1. Function f_2 promotes less control energy if it does not unduly impact fairness, according to f_1.

2.3.4.2 Surge Variance

Large changes of energy consumed from moment to moment (in other words, power consumption surges) are generally undesirable. A third notion of fairness that the team experiments with seeks to distribute such surges evenly across the robots, once they exceed a threshold. Let *M* be the surge threshold. Then a surge occurs for UAS, k, at time *t* if $|e_k[t] - e_k[t-1]| > M$. The total energy surge z_k for UAS, k, is computed as $z_k = \sum_t |e_k[t] - e_k[t-1]| - M$. The resulting fairness notion f_3 is all UAS surges, given by equation $f_3(u) = Var(z_1, \ldots, z_N)$.

The team also defines a notion that combines f_3 with the energy term to reduce overall control energy: $f_4(u) = f_3(u) + \beta Q(u)$.

2.3.5 FiReFly: Fair Motion Planning with Safety Guarantees

The fair control problem of finding control inputs for a team of UAS at time $t > 0$ can be formulated as a convex optimization problem with the following components:

- **Decision Variables**
  - The set of control input sequences of all *N* UAS, $\mathbf{u} = (\boldsymbol{u_1}, \ldots, \boldsymbol{u_N})$
- **Objective**
  - The UAS team maximizes fairness between their control input sequences by minimizing one of the fairness notions defined above
- **Constraints**
  - Ensure all UAS intermediate states between their current state at time *t* and their final state at the end of the time horizon are governed by their system dynamics
  - Ensure that each UAS reaches their goal destination by their set time horizon
  - Ensure all control inputs at every time step are within set control bounds
  - Ensure all previous control inputs executed before time *t* are fixed

The fair motion planner is solved by a distributed algorithm that is a modified version of the algorithm developed in (Pant, Abbas, and Mangharam 2022) which is based on (Razaviyayn et al. 2014).

After fair control input sequences are set, denoted as $\mathrm{u}^{\mathrm{fair}}$, the inputs for the next time step are adjusted for safety through another convex optimization problem with the following components:

- **Decision Variables**
  - The control inputs for the next time step *t* for each UAS, $\mathbf{u}[\mathrm{t}] = (\boldsymbol{u_1}[t], \ldots, \boldsymbol{u_N}[t])$
- **Objective**
  - The UAS team minimizes the magnitude of the input adjustment required for safety
- **Constraints**
  - Ensure all UAS adjust their inputs to maintain a safe distance from obstacles and other UAS. This constraint is called the CBF constraint.
  - Ensure all UAS adjust their inputs to maintain progress towards each's goal position. This constraint is called the Control Lyapunov Function (CLF) constraint.
  - Ensure all control inputs at every time step are within set control bounds

The formulation of this convex optimization problem has the form of a CLF-CBF controller defined in (Ames et al. 2019). The problem is solved centrally following the approach in (Glotfelter, Cortés, and Egerstedt 2017) and in a distributed manner through adapting the approach in (Do Nascimento, Papachristodoulou, and Margellos 2023).

2.3.6 FiReFly Experiments and Results

The team conducts two types of simulation experiments to evaluate FiReFly under different fairness notions. The first investigates how FiReFly performs with an increasing number of obstacles. The second investigates scaling of the number of UAS. Simulations were implemented in Python, and experiments were run on a partition of a high-performance computer cluster using 24 cores with a 2.1 Ghz processor and with access to 50GB RAM.

2.3.6.1 Experiment 1: Increasing Number of Obstacles

In this experiment, the team fixes the number of UAS ($N = 5$) and their starting positions. Obstacles are uniformly randomly generated somewhere along the straight-line path between a UAS's starting position and a single goal area. The team generates 200 trial configurations with the number of obstacles varying from one to five. For each configuration, the team runs FiReFly for all fairness notions with both the central and distributed safe control formulations. The team also runs FiReFly with central safe control and without any fairness notions as a baseline.

Figure 7 shows results for five UAS with an increasing number of obstacles. Mission success rates are shown in the bar plot on the left. The bar plot on the right shows the rate of fairness improvement over the baseline for FiReFly with central safe controller under different notions of fairness. The horizontal lines show the results for FiReFly with a distributed safe controller. The legend on the right plot also applies to the left.

Mission Success: The team defines the mission success rate as the total number of UAS to reach their goal area (within *H* time units) divided by the total number of UAS in all trials. FiReFly with distributed safe control achieves better 100% mission success rates, as shown by the horizontal lines in the left plot in Figure 7. The bars in this figure show that FiReFly with central safe control also improves mission success over the baseline, though the team sees this success rate decrease as the number of obstacles increases.

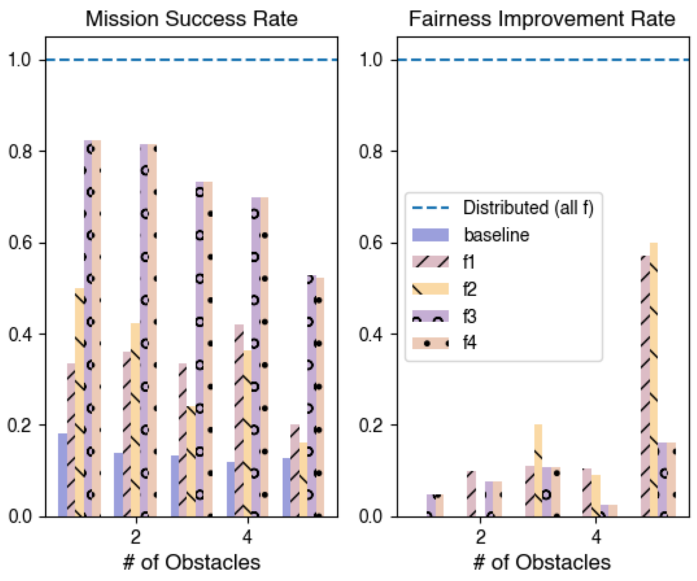


Figure 7. Results for FiReFly Experiment 1.

Fairness: Figure 7 also shows that FiReFly with distributed safe control improves fairness over the baseline in 100% trials. In contrast, FiReFly with central safe control improved fairness over the baseline only a small fraction of the time.

2.3.6.2 Experiment 2: Scaling number of UAS

In this experiment, the team assesses these scaling effects on runtime of FiReFly for UAS team sizes of $N = \{7, 10, 12, 15, 20, 50\}$. The team fixes a single obstacle at the origin and randomly places UAS starting and goal positions around it. For each team size, the team generates 20 of these random configurations and runs FiReFly for all fairness notions. The team focuses on distributed FiReFly in this experiment, given that it had the better performance over the two different safe control formulations in Experiment 1. Mission success rates were nearly 100% for all team sizes.

Runtime: The team compares the runtime of FiReFly to that of FairFly (Kurtz and Abbas 2020), a state-of-the-art UAS motion planner, in Table 2. For both approaches, the reported runtimes are for the time it takes to compute an input for a single timestep. FiReFly is faster than FairFly in all UAS team sizes, and growth in runtime is also slower. In assessing runtimes, timeout is declared after five minutes. To reduce runtime for 50 UAS, the team ran FiReFly without any online fair re-planning and simply using the safe controller to track the initially planned fair trajectory. The team found that while this saves computation time, fairness worsens. Thus, a balance can be achieved by re-planning every few steps.

Fairness: Fairness generally improved as team size grew up to a team size of 15. The team sees this in Figure 6, which shows the mission success rates on the left plot and fairness improvement rates on the right plot for an increasing number of UAS under FiReFly with distributed safe control. Nearly 100% mission success is achieved for all fairness notions.

Table 2. Runtimes in seconds for FairFly and FiReFly.

	FairFly [51]			Distributed FiReFly					
	Total Runtime			Fair Planner Runtime			Safe Control Runtime		
# of UAS	mean	std	max	mean	std	max	mean	std	max
7	35.742	12.132	46.462	**0.005**	0.007	0.031	0.011	0.000	0.012
10	78.644	17.453	104.051	**0.005**	0.006	0.029	0.014	0.001	0.015
12	63.446	29.451	102.503	**0.007**	0.007	0.031	0.019	0.001	0.019
15	109.711	23.950	138.048	**0.010**	0.008	0.033	0.021	0.001	0.024
20	TIMEOUT	TIMEOUT	TIMEOUT	**0.007**	0.009	0.029	0.024	0.001	0.025
50	TIMEOUT	TIMEOUT	TIMEOUT	**0.014**	0.002	0.016	0.053	0.001	0.054

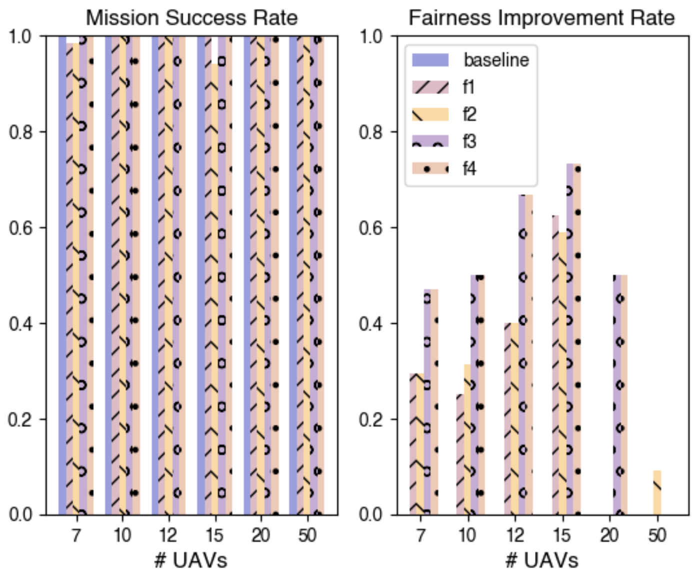


Figure 8. Results for FiReFly Experiment 2.

2.4 References

Althoff, Matthias, Goran Frehse, and Antoine Girard. 2021. “Set Propagation Techniques for Reachability Analysis.” *Annual Review of Control, Robotics, and Autonomous Systems*. Annual Reviews. https://doi.org/10.1146/annurev-control-071420-081941.

Ames, Aaron D., Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. 2019. “Control Barrier Functions: Theory and Applications.” In *2019 18th European Control Conference (ECC)*, 3420–31. https://doi.org/10.23919/ECC.2019.8796030.

Ames, Aaron D., Xiangru Xu, Jessy W. Grizzle, and Paulo Tabuada. 2017. “Control Barrier Function Based Quadratic Programs for Safety Critical Systems.” *IEEE Transactions on Automatic Control* 62 (8): 3861–76. https://doi.org/10.1109/TAC.2016.2638961.

Bak, Stanley, and Parasara Sridhar Duggirala. 2017. “HyLAA: A Tool for Computing Simulation-Equivalent Reachability for Linear Systems.” In *Proceedings of the 20th International*

Conference on Hybrid Systems: Computation and Control, 173–78. HSCC ’17. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3049797.3049808.

Barnhart, Cynthia, Dimitris Bertsimas, Constantine Caramanis, and Douglas Fearing. 2012. “Equitable and Efficient Coordination in Traffic Flow Management.” *Transportation Science* 46 (2): 262–80.

Bertsimas, Dimitris, and Shubham Gupta. 2016. “Fairness and Collaboration in Network Air Traffic Flow Management: An Optimization Approach.” *Transportation Science* 50 (1): 57–76.

Bertsimas, Dimitris, Guglielmo Lulli, and Amedeo R. Odoni. 2011. “An Integer Optimization Approach to Large-Scale Air Traffic Flow Management.” *Oper. Res.* 59:211–27.

Bertsimas, Dimitris, and Sarah Stock Patterson. 1998. “The Air Traffic Flow Management Problem with Enroute Capacities.” *Oper. Res.* 46 (3): 406–22.

Bilimoria, Karl D., Banavar Sridhar, Shon R. Grabbe, Gano B. Chatterji, and Kapil S. Sheth. 2001. “FACET: Future ATM Concepts Evaluation Tool.” *Air Traffic Control Quarterly* 9 (1): 1–20. https://doi.org/10.2514/atcq.9.1.1.

Brahmbhatt, Khushal and Oregon State University. 2023. “A Distributed Mixed-Integer-Programming Approach to Fair Trajectory Planning of Autonomous Systems.” Master’s Thesis, Oregon State University.

Brandão, Martim, Marina Jirotka, Helena Webb, and Paul Luff. 2020. “Fair Navigation Planning: A Resource for Characterizing and Designing Fairness in Mobile Robots.” *Artificial Intelligence* 282:103259. https://doi.org/10.1016/j.artint.2020.103259.

Buyukkocak, Ali Tevfik, Derya Aksaray, and Yasin Yazıcıoğlu. 2023. “Energy-Aware Planning of Heterogeneous Multi-Agent Systems for Serving Cooperative Tasks with Temporal Logic Specifications.” In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 8659–65. https://doi.org/10.1109/IROS55552.2023.10342064.

Chandran, Bala G., and Hamsa Balakrishnan. 2017. “A Distributed Framework for Traffic Flow Management in the Presence of Unmanned Aircraft.” In *Proceedings of the USA/Europe Air Traffic Management R&D Seminar*, 26–30. Seattle, Washington: ATM Seminar.

Chin, Christopher, Karthik Gopalakrishnan, Maxim Egorov, Antony Evans, and Hamsa Balakrishnan. 2021. “Efficiency and Fairness in Unmanned Air Traffic Flow Management.” *IEEE Transactions on Intelligent Transportation Systems* 22 (9): 5939–51. https://doi.org/10.1109/TITS.2020.3048356.

Chin, Christopher, Max Z. Li, and Yash Vardhan Pant. 2022. “Distributed Traffic Flow Management for Uncrewed Aircraft Systems.” In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 3625–31. Macau, China: IEEE. https://doi.org/10.1109/ITSC55140.2022.9922227.

Do Nascimento, Allan Andre, Antonis Papachristodoulou, and Kostas Margellos. 2023. “A Game Theoretic Approach for Safe and Distributed Control of Unmanned Aerial Vehicles.” In *2023 62nd IEEE Conference on Decision and Control (CDC)*, 1070–75. Singapore, Singapore: IEEE. https://doi.org/10.1109/CDC49753.2023.10383672.

FAA. 2020. “Urban Air Mobility (UAM) Concept of Operations (ConOps).” *Federal Aviation Administration* 2.0:1–28.

———. 2023a. *FAA Airspace Forecast 2022-2043. FAA Airspace Forecast 2022-2043*. Washington, DC: Federal Aviation Administration.

———. 2023b. “FAA Pilot Handbook, Chapter 7, Section 6.” Edited by Federal Aviation Administration. *Aeronautical Information Manual*. Federal Aviation Administration.

Glotfelter, Paul, Jorge Cortés, and Magnus Egerstedt. 2017. “Nonsmooth Barrier Functions With Applications to Multi-Robot Systems.” *IEEE Control Systems Letters* 1 (2): 310–15. https://doi.org/10.1109/LCSYS.2017.2710943.

Israr, Amber, Zain Anwar Ali, Eman H. Alkhammash, and Jari Juhani Jussila. 2022. “Optimization Methods Applied to Motion Planning of Unmanned Aerial Vehicles: A Review.” *Drones* 6 (5). https://doi.org/10.3390/drones6050126.

Ji, Yao, Chao Dong, Xiaojun Zhu, and Qihui Wu. 2020. “Fair-Energy Trajectory Planning for Multi-Target Positioning Based on Cooperative Unmanned Aerial Vehicles.” *IEEE Access* 8:9782–95. https://doi.org/10.1109/ACCESS.2019.2962240.

Kousik, Shreyas, Patrick Holmes, and Ramanarayan Vasudevan. 2019. “Technical Report: Safe, Aggressive Quadrotor Flight via Reachability-Based Trajectory Design.” arXiv. http://arxiv.org/abs/1904.05728.

Kurtz, Connor, and Houssam Abbas. 2020. “FairFly: A Fair Motion Planner for Fleets of Autonomous UAS in Urban Airspace.” In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 1–6. https://doi.org/10.1109/ITSC45102.2020.9294612.

Liu, Jinsun, Challen Enninful Adu, Lucas Lymburner, Vishrut Kaushik, Lena Trang, and Ram Vasudevan. 2023. “RADIUS: Risk-Aware, Real-Time, Reachability-Based Motion Planning.” *Robotics, Science and Systems 2023*. https://doi.org/10.48550/arXiv.2302.07933.

Mercedes Pelegrín, Rémi Delmas, Claudia D’Ambrosio, and Youssef Hamadi. 2023. “Urban Air Mobility: From Complex Tactical Conflict Resolution to Network Design and Fairness Insights.” *Optimization Methods and Software* 38 (6): 1311–43.

Michaux, Jonathan, Qingyi Chen, Challen Enninful Adu, Jinsun Liu, and Ram Vasudevan. 2024. “Reachability-Based Trajectory Design via Exact Formulation of Implicit Neural Signed Distance Functions.”

OpenDataPhilly. n.d. “Census Blocks Dataset.”

Pant, Yash Vardhan, Houssam Abbas, and Rahul Mangharam. 2022. “Distributed Trajectory Planning for Multi-Rotor UAS with Signal Temporal Logic Objectives.” In *2022 IEEE Conference on Control Technology and Applications (CCTA)*, 476–83. https://doi.org/10.1109/CCTA49430.2022.9966096.

Pant, Yash Vardhan, Houssam Abbas, Rhudii A. Quaye, and Rahul Mangharam. 2018. “Fly-by-Logic: Control of Multi-Drone Fleets with Temporal Logic Objectives.” In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, 186–97. https://doi.org/10.1109/ICCPS.2018.00026.

Rahman, Mamunur, Nurul I. Sarkar, and Raymond Lutui. 2025. “A Survey on Multi-UAS Path Planning: Classification, Algorithms, Open Research Problems, and Future Directions.” *Drones* 9 (4). https://doi.org/10.3390/drones9040263.

Razaviyayn, Meisam, Mingyi Hong, Zhi-Quan Luo, and Jong-Shi Pang. 2014. “Parallel Successive Convex Approximation for Nonsmooth Nonconvex Optimization.” In

Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1, 1440–48. NIPS'14. Cambridge, MA, USA: MIT Press.

Sabatino, Francesco. 2015. "Quadrotor Control: Modeling, Nonlinearcontrol Design, and Simulation."

Tang, Hualong, Yu Zhang, Vahid Mohmoodian, and Hadi Charkhgard. 2021. "Automated Flight Planning of High-Density Urban Air Mobility." *Transportation Research Part C: Emerging Technologies* 131:103324.

U.S Department of Transportation (DOT) Federal Aviation Administration (FAA). 2024a. "FAA Report A21__A11L.UAS.69 - Integrating Expanded and Non-Segregated UAS Operations into the NAS: Impact on Traffic Trends and Safety." U.S Department of Transportation (DOT) Federal Aviation Administration (FAA).

———. 2024b. "Part 107 Waviers." *Federal Aviation Administration*. U.S Department of Transportation (DOT) Federal Aviation Administration (FAA).

Vossen, Thomas, Michael Ball, Robert Hoffman, and Michael Wambsganss. 2003. "A General Approach to Equity in Traffic Flow Management and Its Application to Mitigating Exemption Bias in Ground Delay Programs." *Air Traffic Control Quarterly* 11 (4): 277–92.

Zhao, Chenxi, Junyu Liu, Min Sheng, Wei Teng, Yang Zheng, and Jiandong Li. 2021. "Multi-UAS Trajectory Planning for Energy-Efficient Content Coverage: A Decentralized Learning-Based Approach." *IEEE Journal on Selected Areas in Communications* 39 (10): 3193–3207. https://doi.org/10.1109/JSAC.2021.3088669.

3 PRREACH: PROBABILISTIC RISK ANALYSIS USING REACHABILITY ANALYSIS FOR UAS CONTROL

Some commercial UAS operations in the national airspace system first require approval through the FAA's Part 107 waiver process (U.S Department of Transportation (DOT) Federal Aviation Administration (FAA) 2024b, 107). This waiver process requires risk assessment and mitigation strategies. A common definition of risk is the probability of a bad event occurring multiplied by its severity, and this definition is adopted in this chapter. Thus, a low-severity event that occurs frequently, or a high-risk event that occurs rarely, might have comparable risks. A risk assessment must be done for all potential hazard outcome events – e.g., collision with a person or crashing into a building – given potential hazard causes – e.g., a broken rotor, strong winds, or sensor malfunctions. Existing frameworks that evaluate risk of UAS operations use probabilistic risk analysis methodologies (A11L.UAS.69 U.S Department of Transportation (DOT) Federal Aviation Administration (FAA) 2024), commonly known as a Probabilistic Risk Assessment (PRA). PRA relies on conditional probabilities of hazard outcomes given hazard causes; these conditional probabilities must be computed from historical UAS incident data. However, there is very limited data, and this makes real-world implementation of PRA frameworks difficult if not impossible. Furthermore, classical PRA does not include controller-level strategies for in-flight UAS risk mitigation, but relies entirely on pre-flight assessments. Therefore, there is a need for a PRA that can be performed in low-data regimes, and which can be executed pre-flight as well as in-flight. In this chapter, the team demonstrates how automation of control functionalities on-board UAS makes such an approach possible.

3.1 Executive summary

The team used reachability analysis to formalize an evaluation of risk for a UAS' concept of operations. The team calls this new PRA tool PRREACH. Using real data from the city of Philadelphia and common dynamical models of UAS, the team demonstrates that PRREACH can reduce risk of collision with a person by up to 23% through offline, pre-flight risk assessment, and up to 87% through in-flight risk assessment, with the cost being greater control effort and distance offset from the UAS's final destination, but still well within acceptable norms. These results are obtained for a variety of modeled hazard causes like strong winds, broken rotors, and sensor errors.

Proposed guidance: The team proposes that reachability-based PRA, such as demonstrated in PRREACH, become a required part of the evidence submitted by a UAS operator for approval of their CONOPs, at least as long as reliable, clean data on hazard outcomes and hazard causes remains scarce. In the absence of such low-data risk assessments, assurance cases remain incomplete and qualitative at best. The team proposes that a low-data risk analysis tool, such as PRREACH, become a standard part of the authorities' own certification efforts for autonomous UAS operations.

Prototype software available at https://github.com/sabotagelab/PRREACH/tree/main

3.2 Traditional PRA and Limitations

The FAA has established safety and risk management protocols through orders such as FAA Order 8000.369C, which outlines the Safety Management System (SMS), and FAA Order 8040.4B, which defines the Safety Risk Management Policy (SRMP). These frameworks serve as the foundation for assessing and mitigating risks in aviation operations.

According to FAA Order 8040.4B, SRMP delineates the procedural requirements for executing Safety Risk Management (SRM) within the FAA. SRM is one of four integral components of SMS, alongside Safety Policy, Safety Assurance, and Safety Promotion. The SRMP process consists of five essential steps: *(i)* system analysis, *(ii)* hazard identification, *(iii)* risk analysis, *(iv)* risk assessment, and *(v)* risk control.

Previous PRA frameworks for UAS generally follow this standard sequence. The ASSURE-21 project (A11L.UAS.69 U.S Department of Transportation (DOT) Federal Aviation Administration (FAA) 2024), presented such a framework that at a high level works as follows:

For a given CONOPs of a UAS, including its origin, destination, flight times, and other information pertinent to its mission, first enumerate all possible *hazard causes* – undesirable external factors such as strong turbulence, low fuel, faulty sensors – and all possible *hazard outcomes* – adverse results such as collisions with other aircraft, people, or buildings. Then identify the *critical locations* at which to evaluate risk, defined as the likelihood or *probability* of a hazard outcome multiplied against the severity of the outcome. For each of these critical locations, use historical UAS incident data to obtain the conditional probability of each hazard outcome given each hazard cause, and unconditional probability of each hazard outcome at each critical location using

Bayesian statistical methods. With these hazard likelihoods in hand, traffic light tables can be used to determine the risk level at every critical location. Then, fusion rules will aggregate the risk levels to compute an overall assessment of the CONOPs. Finally, a "fly or no fly" decision can be made for the UAS request based on the CONOPs assessment. While this framework provides a mathematically rigorous, quantitative process for assessing risk for UAS-specific operations, it has some limitations.

First, historical UAS incident data required for computation of the conditional probabilities is limited and thus necessitates the use of broad assumptions in the risk assessment. Second, the framework only considers categorical hazard causes. For example, it can model the presence or absence of wind acting on the UAS, but not the wind speed. Third, while risk can be evaluated at many separate locations, the fusion rules ultimately produce a single assessment for the CONOPs. This may result in overly conservative decisions for some CONOPs if very high risk is determined at a particular location that the UAS can simply avoid with a risk-aware controller.

The forthcoming approach PRREACH addresses these limitations. Instead of relying on limited UAS incident data, PRREACH can rely on public and readily available UAS dynamics and open-source spatial data for relating hazard causes to hazard outcomes. PRREACH can incorporate continuous hazard causes, such as wind speed, into its risk assessment. Where previous PRA frameworks average risk over the entire airspace, PRREACH uses reachability analysis to compute risk over only the feasible trajectories of the UAS. Finally, where previous PRA frameworks do not provide methods for in-flight control, PRREACH can produce risk-bounded controllers computed offline before take-off or online during flight.

3.3 Reachability Analysis

Reachability or reachability analysis is the computation of the set of future states of a dynamical system from some initial set of states. These sets are called *reach sets*. Algorithms are available for computing the reach sets of discrete-time and continuous-time linear, non-linear, deterministic, and stochastic systems, with many specializations for various cases. The team refers to (Althoff, Frehse, and Girard 2021) and (Bak and Duggirala 2017) and the references therein for a deeper background on reachability.

Existing control methods for robotic systems that use reachability to ensure safety include the work by (Kousik, Holmes, and Vasudevan 2019), which computes the reach set over all trajectories of a UAS and finds the subset of safe trajectories within the reach set that do not intersect with an obstacle. Similarly, (Liu et al. 2023) finds risk-aware trajectories of a ground vehicle are found by first computing reach sets offline that over-approximate all possible trajectories offline and then using chance-constrained optimization online to find a trajectory that has a chance of collision less than some threshold. The work in (Michaux et al. 2024) also computes reach sets offline and then finds collision-avoidant trajectories within the reach set through online optimization, but uses a neural network to formulate a safety constraint in the optimization.

3.4 PRREACH Operational Steps

PRREACH exists currently as a prototype software solution. We envision PRREACH can be applied by regulators evaluating risk of proposed UAS operations and can aid operators in developing risk mitigation measures. The process for operators would work as follows:

1. If a hazard cause – sensor error, broken rotor, or strong winds – occurs before flight, use a precomputed PRREACH controller to guarantee risk below a set threshold. Otherwise, proceed with nominal flight controllers.
2. If no hazard cause is present initially but occurs later during flight, either
    a. switch to the PRREACH controller that was precomputed *offline* for the hazard cause, OR
    b. if time and compute resources are available, compute a new PRREACH controller *online*.

3.5 PRREACH Technical Components

3.5.1 UAS CONOPs, Dynamics, and Reach Sets

The team defines a UAS CONOP to include the *set of possible initial states* of the UAS, the destination of the UAS, the time horizon over which the UAS must travel from its initial state to the destination, and the *generalized dynamics* of the UAS.

The generalized dynamics of the UAS is a collection of *linear, discrete time* dynamics under each possible hazard cause, including the case when no hazard cause occurs.

Reachability analysis is then used to compute the reach sets over time given the set of initial states and the generalized dynamics.

3.5.2 Hazard Maps and Risk Estimation

PRREACH uses *hazard maps* to model the probability of hazard outcomes occurring at different locations. Hazard maps are based on various *environmental and operational factors*, such as:

- **Population density** (risk of UAS colliding with a pedestrian)
- **Structural density** (risk of UAS impacting buildings or infrastructure)

3.5.3 Markov Process Modeling of UAS Risk

PRREACH computes risk by modeling UAS motion as a *Markov process*:

- Each reach set is treated as a *Markov state.*
- The UAS transitions between Markov states over time.
- At each Markov state, the probability of a hazard outcome occurring in the reach set is computed using the hazard map.
- A hazard outcome occurring is treated as a transition to a terminal Markov state representing mission failure or an emergency landing.

3.5.4 Risk-Bounded Control Optimization

PRREACH includes a control algorithm that *optimizes the UAS control gains* to maintain efficient operation while bounding the computed risk. The algorithm consists of solving an optimization problem to determine a controller gain matrix that is close to that of the original controller associated with the dynamics but is constrained by some appropriate risk threshold. The inputs to the algorithm are:

- The set of initial states of the UAS
- The dynamics associated with the hazard cause in question, taken from the generalized dynamics defined for the UAS
- The hazard maps associated with the hazard outcomes in question
- The risk threshold for the operation, defined by the operation's stakeholders

3.6 PRREACH Implementation and Experiments

3.6.1 Hazard Outcome Maps

The team considered two hazard outcomes maps that represent the probability of collision with a person and the probability of collision with a building. Both hazard maps are represented as degree three polynomials, approximating the population and building density of city blocks in Philadelphia, PA, available through an open data repository of census blocks (OpenDataPhilly, n.d.).

3.6.2 Hazard Causes

For the experiments, the team used the UAS dynamics model from (Sabatino 2015) and modify the dynamics equations appropriately for the following hazard causes:

- **Deficient Rotor**: Reduces overall control of the UAS, which the team modeled by scaling the control matrix in the dynamics by a fixed coefficient
- **Sensor Error**: Reduces accuracy in controlling the UAS's position, which the team modeled by altering the position components in the drift matrix in the dynamic
- **Wind Disturbance:** Pushes the UAS outside of its nominal trajectory. The team modeled the wind disturbance as a vector added to the UAS's state

Under each of the hazard causes, the team computed a Linear Quadratic Regulator (LQR) controller as a baseline controller that is not optimized to reduce the risk of any hazard outcome.

3.6.3 Experimental Setup

The team implemented the operational process for PRREACH described in Section 2.3 and ran two experiments. The first precomputes PRREACH controllers *offline,* assuming the presence of a hazard cause before flight. The second computes PRREACH controllers *online* after a hazard cause occurs.

The team set up a CONOP in which the UAS must maintain its altitude and fly over a group of city blocks whose population and building density are modeled by the hazard outcome maps discussed in Section 2.5.1. The team set the operational risk threshold to the risk evaluation generated by the baseline LQR controller under dynamics when no hazard cause is present. This ensures the resulting PRREACH controllers do not produce trajectories whose overall risk exceeds that of trajectories produced under normal conditions with no hazard cause.

For the first experiment, the team ran the PRREACH control optimization for each hazard outcome map and hazard cause combination. The team called each output result a *PRR-offline* controller. The team compared the trajectory it generates from the initial position with the trajectory generated by the LQR controller in terms of risk and final distance to the target.

For the second experiment, the team ran the PRREACH control optimization online for 100 simulated flights, each from different initial positions, and with a hazard cause occurring at a randomly selected time. The team called each output result a *PRR-online* controller. The team compared the risk and final distance to target of the trajectories generated by PRR-offline and PRR-online controllers from the time the hazard cause occurred.

Simulations were implemented in Python using SciPy for solving the optimization. Experiments were run on a computer with an 8-core 3.2Ghz processor and access up to 32GB RAM.

3.7 PRREACH Experimental Results

Figure 9 shows example trajectories from PRR-offline controllers. On the left of this figure, there are example UAS trajectories over a heatmap representing population density of city blocks in Philadelphia. The blue line shows the nominal trajectory over this area towards the target given by a red dot. The orange line shows the trajectory when the UAS experiences a deficient rotor, causing it to fly over a more densely populated area. The light green and blue rectangles show the reach sets of the UAS with the deficient rotor. On the right, the same heatmap and nominal UAS trajectory are shown. The orange line here shows the UAS trajectory using a PRR-offline controller, which directs the UAS away from the more densely populated regions. This results in the reduced risk evaluation from 0.79 to 0.61 between the left and right plots, at the cost of the UAS on the right being further from the target at the end of the time horizon.

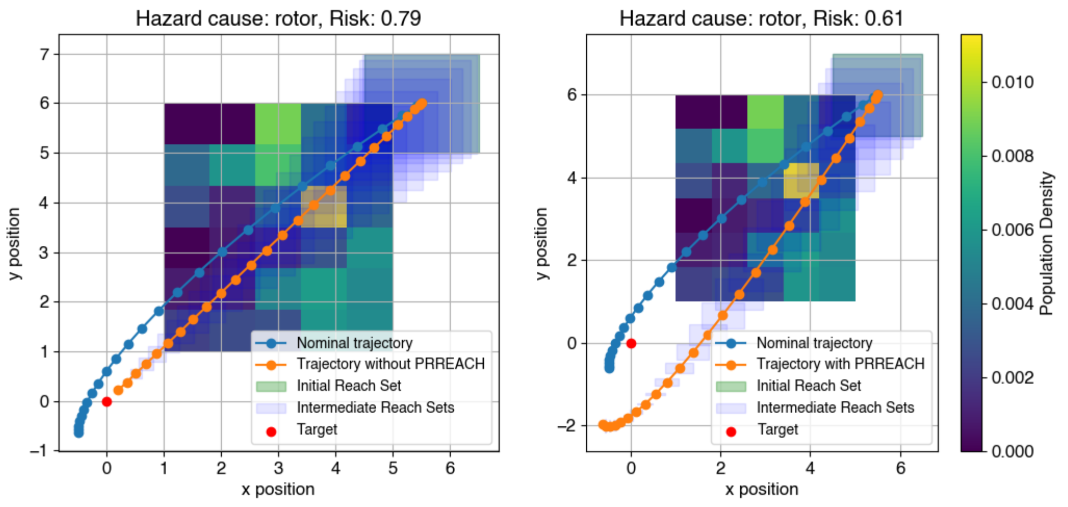


Figure 9. Example result from PRREACH Experiment 1.

3.7.1 Reducing risk for greater distance to target

From the first experiment, the team found that a trajectory produced by the PRR-offline controller has lower initial overall risk than that of the LQR controller. The trade-off for lower risk compared to the LQR controller is a greater distance between the final state in the trajectory and the target destination. Table 3 shows this risk reduction and increase in distance of the PRR-offline controller produced trajectories as a percentage of those of the LQR controller. The trajectories produced by the PRR-offline controller reduce the risk of collision with a person or building in the presence of a hazard cause, but at the expense of greater distance to the target at the end of the time horizon, for all three hazard causes.

Table 3. PRREACH Experiment 1 Results.

	% Reduction in Risk		**% Increased Distance to Target**	
Hazard Cause	**Person**	**Building**	**Person**	**Building**
Deficient Rotor	23.55%	1.62%	598.90%	780.10%
Sensor Error	8.10%	1.40%	33.87%	89.50%
Wind	25.52%	3.27%	3.15%	0.31%

It was found in the team's second experiment that by re-solving the PRREACH control optimization at the time the hazard cause occurs and using the resulting PRR-online controller, rather than simply switching to the PRR-offline controller, the resulting trajectory, on average, will have even lower overall risk from the point the hazard cause occurs. This means that overall risk can be further reduced by re-optimizing the UAS controller with up-to-date information at the time the hazard cause occurs. Furthermore, the distance between the final state in the trajectory and the target destination was on average closer for trajectories produced from the PRR-online than the PRR-offline controller. For the sensor error and wind hazard causes, the trajectories would result in a final position that is closer to the target than that of the LQR controller. Table 4 summarizes these results. The table reports average risk reduction and the increase in final distance to target of the trajectories produced by the PRR-offline and PRR-online controllers as a percentage of the risk and final distance to target of trajectories produced by the LQR controller. Results were averaged over 100 trajectories in which the hazard cause occurred at random times in flight. Trajectories produced by the PRR-online controllers resulted in greater risk reduction than the PRR-offline controllers, as well as lower distance to the target. In some cases, PRREACH trajectories would result in a final distance to the target that is lower than trajectories from the LQR controller, as indicated by the negative values in the table.

Table 4. PRREACH Experiment 2 Results.

	Average % Reduction in Risk				**Average % Increased Distance to Target**			
	Person		**Building**		**Person**		**Building**	
Hazard Cause	**PRR-offline**	**PRR-online**	**PRR-offline**	**PRR-online**	**PRR-offline**	**PRR-online**	**PRR-offline**	**PRR-online**

Deficient Rotor	23.62%	47.84%	3.98%	24.25%	311.07%	42.70%	417.17%	80.46%
Sensor Error	8.46%	43.90%	3.29%	25.03%	37.87%	-27.58%	108.89%	-16.77%
Wind	24.02%	53.46%	3.18%	29.39%	-4.05%	-27.83%	2.16%	-37.19%

Figure 10 visualizes trajectories from a single simulation of this second experiment. The figure shows trajectories produced by an LQR controller (blue), a PRR-offline controller (orange), and a PRR-online controller (green) under the wind hazard. The trajectories depict the UAS's simulated path under each of the controllers after the hazard cause occurs mid-flight. The heatmap shows the building density. Outside the heatmap, there is no risk of collision with a building. The PRREACH controllers produce trajectories that spend less time over buildings as compared to the LQR controller, and the trajectory from the PRR-online controller ends up closest to the target point (red dot).

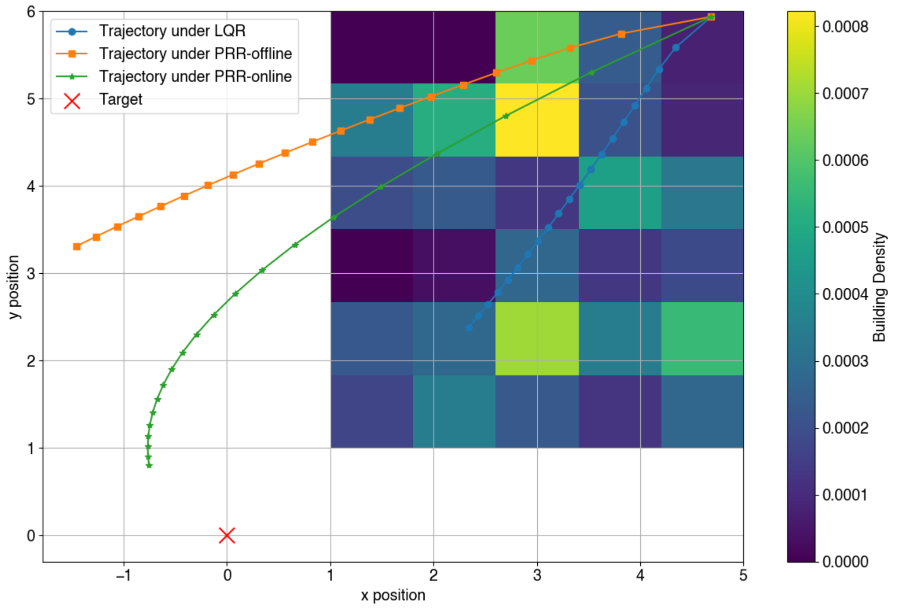


Figure 10. Example result from PRREACH Experiment 2.

3.7.2 Optimization Runtime

Runtime to solve the PRREACH control optimization varies depending on the hazard cause dynamics, with the range of averages for the dynamics the team chose being 8-53 seconds. Over all simulations, the maximum time to compute a controller took almost 400 seconds under the wind hazard cause. The maximum, average, and standard deviation of runtimes in the team's second experiment are reported in Table 5.

Table 5. PRREACH optimization runtimes.

	Person			**Building**		
Hazard Cause	**Average**	**Max**	**Std Dev.**	**Average**	**Max**	**Std Dev.**
Deficient Rotor	8.829	11.457	0.812	9.524	11.070	1.147
Sensor Error	7.994	9.969	1.031	9.424	10.763	0.965
Wind	53.630	396.593	68.337	16.899	38.678	10.774

3.7.3 PRREACH in Practice

For a UAS CONOPs, if some deviation to the target can be tolerated, and thus accounted for with longer flight times, PRREACH provides an offline tool for operators to prepare controllers for mitigating risk of hazard outcomes under different hazard causes. If online compute resources are available, and operations can allow a several-second buffer for computation, using PRREACH online further improves in-flight risk mitigation, in some cases without compromising distance to the target. For dynamics which online computation of PRREACH may take too much time, a PRR-offline controller can still be used for in-flight risk mitigation.

3.8 References

Althoff, Matthias, Goran Frehse, and Antoine Girard. 2021. "Set Propagation Techniques for Reachability Analysis." *Annual Review of Control, Robotics, and Autonomous Systems*. Annual Reviews. https://doi.org/10.1146/annurev-control-071420-081941.

Ames, Aaron D., Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. 2019. "Control Barrier Functions: Theory and Applications." In *2019 18th European Control Conference (ECC)*, 3420–31. https://doi.org/10.23919/ECC.2019.8796030.

Ames, Aaron D., Xiangru Xu, Jessy W. Grizzle, and Paulo Tabuada. 2017. "Control Barrier Function Based Quadratic Programs for Safety Critical Systems." *IEEE Transactions on Automatic Control* 62 (8): 3861–76. https://doi.org/10.1109/TAC.2016.2638961.

Bak, Stanley, and Parasara Sridhar Duggirala. 2017. "HyLAA: A Tool for Computing Simulation-Equivalent Reachability for Linear Systems." In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control*, 173–78. HSCC '17. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3049797.3049808.

Barnhart, Cynthia, Dimitris Bertsimas, Constantine Caramanis, and Douglas Fearing. 2012. "Equitable and Efficient Coordination in Traffic Flow Management." *Transportation Science* 46 (2): 262–80.

Bertsimas, Dimitris, and Shubham Gupta. 2016. "Fairness and Collaboration in Network Air Traffic Flow Management: An Optimization Approach." *Transportation Science* 50 (1): 57–76.

Bertsimas, Dimitris, Guglielmo Lulli, and Amedeo R. Odoni. 2011. "An Integer Optimization Approach to Large-Scale Air Traffic Flow Management." *Oper. Res.* 59:211–27.

Bertsimas, Dimitris, and Sarah Stock Patterson. 1998. "The Air Traffic Flow Management Problem with Enroute Capacities." *Oper. Res.* 46 (3): 406–22.

Bilimoria, Karl D., Banavar Sridhar, Shon R. Grabbe, Gano B. Chatterji, and Kapil S. Sheth. 2001. "FACET: Future ATM Concepts Evaluation Tool." *Air Traffic Control Quarterly* 9 (1): 1–20. https://doi.org/10.2514/atcq.9.1.1.

Brahmbhatt, Khushal and Oregon State University. 2023. "A Distributed Mixed-Integer-Programming Approach to Fair Trajectory Planning of Autonomous Systems." Master's Thesis, Oregon State University.

Brandão, Martim, Marina Jirotka, Helena Webb, and Paul Luff. 2020. "Fair Navigation Planning: A Resource for Characterizing and Designing Fairness in Mobile Robots." *Artificial Intelligence* 282:103259. https://doi.org/10.1016/j.artint.2020.103259.

Buyukkocak, Ali Tevfik, Derya Aksaray, and Yasin Yazıcıoğlu. 2023. "Energy-Aware Planning of Heterogeneous Multi-Agent Systems for Serving Cooperative Tasks with Temporal Logic Specifications." In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 8659–65. https://doi.org/10.1109/IROS55552.2023.10342064.

Chandran, Bala G., and Hamsa Balakrishnan. 2017. "A Distributed Framework for Traffic Flow Management in the Presence of Unmanned Aircraft." In *Proceedings of the USA/Europe Air Traffic Management R&D Seminar*, 26–30. Seattle, Washington: ATM Seminar.

Chin, Christopher, Karthik Gopalakrishnan, Maxim Egorov, Antony Evans, and Hamsa Balakrishnan. 2021. "Efficiency and Fairness in Unmanned Air Traffic Flow Management." *IEEE Transactions on Intelligent Transportation Systems* 22 (9): 5939–51. https://doi.org/10.1109/TITS.2020.3048356.

Chin, Christopher, Max Z. Li, and Yash Vardhan Pant. 2022. "Distributed Traffic Flow Management for Uncrewed Aircraft Systems." In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, 3625–31. Macau, China: IEEE. https://doi.org/10.1109/ITSC55140.2022.9922227.

Do Nascimento, Allan Andre, Antonis Papachristodoulou, and Kostas Margellos. 2023. "A Game Theoretic Approach for Safe and Distributed Control of Unmanned Aerial Vehicles." In *2023 62nd IEEE Conference on Decision and Control (CDC)*, 1070–75. Singapore, Singapore: IEEE. https://doi.org/10.1109/CDC49753.2023.10383672.

FAA. 2020. "Urban Air Mobility (UAM) Concept of Operations (ConOps)." *Federal Aviation Administration* 2.0:1–28.

———. 2023a. *FAA Airspace Forecast 2022-2043*. *FAA Airspace Forecast 2022-2043*. Washington, DC: Federal Aviation Administration.

———. 2023b. "FAA Pilot Handbook, Chapter 7, Section 6." Edited by Federal Aviation Administration. *Aeronautical Information Manual*. Federal Aviation Administration.

Glotfelter, Paul, Jorge Cortés, and Magnus Egerstedt. 2017. "Nonsmooth Barrier Functions With Applications to Multi-Robot Systems." *IEEE Control Systems Letters* 1 (2): 310–15. https://doi.org/10.1109/LCSYS.2017.2710943.

Israr, Amber, Zain Anwar Ali, Eman H. Alkhammash, and Jari Juhani Jussila. 2022. "Optimization Methods Applied to Motion Planning of Unmanned Aerial Vehicles: A Review." *Drones* 6 (5). https://doi.org/10.3390/drones6050126.

Ji, Yao, Chao Dong, Xiaojun Zhu, and Qihui Wu. 2020. “Fair-Energy Trajectory Planning for Multi-Target Positioning Based on Cooperative Unmanned Aerial Vehicles.” *IEEE Access* 8:9782–95. https://doi.org/10.1109/ACCESS.2019.2962240.

Kousik, Shreyas, Patrick Holmes, and Ramanarayan Vasudevan. 2019. “Technical Report: Safe, Aggressive Quadrotor Flight via Reachability-Based Trajectory Design.” arXiv. http://arxiv.org/abs/1904.05728.

Kurtz, Connor, and Houssam Abbas. 2020. “FairFly: A Fair Motion Planner for Fleets of Autonomous UAS in Urban Airspace.” In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 1–6. https://doi.org/10.1109/ITSC45102.2020.9294612.

Liu, Jinsun, Challen Enninful Adu, Lucas Lymburner, Vishrut Kaushik, Lena Trang, and Ram Vasudevan. 2023. “RADIUS: Risk-Aware, Real-Time, Reachability-Based Motion Planning.” *Robotics, Science and Systems 2023*. https://doi.org/10.48550/arXiv.2302.07933.

Mercedes Pelegrín, Rémi Delmas, Claudia D’Ambrosio, and Youssef Hamadi. 2023. “Urban Air Mobility: From Complex Tactical Conflict Resolution to Network Design and Fairness Insights.” *Optimization Methods and Software* 38 (6): 1311–43.

Michaux, Jonathan, Qingyi Chen, Challen Enninful Adu, Jinsun Liu, and Ram Vasudevan. 2024. “Reachability-Based Trajectory Design via Exact Formulation of Implicit Neural Signed Distance Functions.”

OpenDataPhilly. n.d. “Census Blocks Dataset.”

Pant, Yash Vardhan, Houssam Abbas, and Rahul Mangharam. 2022. “Distributed Trajectory Planning for Multi-Rotor UAS with Signal Temporal Logic Objectives.” In *2022 IEEE Conference on Control Technology and Applications (CCTA)*, 476–83. https://doi.org/10.1109/CCTA49430.2022.9966096.

Pant, Yash Vardhan, Houssam Abbas, Rhudii A. Quaye, and Rahul Mangharam. 2018. “Fly-by-Logic: Control of Multi-Drone Fleets with Temporal Logic Objectives.” In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, 186–97. https://doi.org/10.1109/ICCPS.2018.00026.

Rahman, Mamunur, Nurul I. Sarkar, and Raymond Lutui. 2025. “A Survey on Multi-UAS Path Planning: Classification, Algorithms, Open Research Problems, and Future Directions.” *Drones* 9 (4). https://doi.org/10.3390/drones9040263.

Razaviyayn, Meisam, Mingyi Hong, Zhi-Quan Luo, and Jong-Shi Pang. 2014. “Parallel Successive Convex Approximation for Nonsmooth Nonconvex Optimization.” In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, 1440–48. NIPS’14. Cambridge, MA, USA: MIT Press.

Sabatino, Francesco. 2015. “Quadrotor Control: Modeling, Nonlinearcontrol Design, and Simulation.”

Tang, Hualong, Yu Zhang, Vahid Mohmoodian, and Hadi Charkhgard. 2021. “Automated Flight Planning of High-Density Urban Air Mobility.” *Transportation Research Part C: Emerging Technologies* 131:103324.

U.S Department of Transportation (DOT) Federal Aviation Administration (FAA). 2024a. “FAA Report A21__A11L.UAS.69 - Integrating Expanded and Non-Segregated UAS Operations into the NAS: Impact on Traffic Trends and Safety." U.S Department of Transportation (DOT) Federal Aviation Administration (FAA).

———. 2024b. "Part 107 Waviers." *Federal Aviation Administration*. U.S Department of Transportation (DOT) Federal Aviation Administration (FAA).

Vossen, Thomas, Michael Ball, Robert Hoffman, and Michael Wambsganss. 2003. "A General Approach to Equity in Traffic Flow Management and Its Application to Mitigating Exemption Bias in Ground Delay Programs." *Air Traffic Control Quarterly* 11 (4): 277–92.

Zhao, Chenxi, Junyu Liu, Min Sheng, Wei Teng, Yang Zheng, and Jiandong Li. 2021. "Multi-UAS Trajectory Planning for Energy-Efficient Content Coverage: A Decentralized Learning-Based Approach." *IEEE Journal on Selected Areas in Communications* 39 (10): 3193–3207. https://doi.org/10.1109/JSAC.2021.3088669.

4 RUNTIME VERIFICATION

A group of flying UAS, which may or may not be under the purview of a single operator, constitutes a *distributed system:* a system of interacting independent agents. Regardless of such a system's design process, it is necessary to monitor its operation at runtime; this includes each UAS' internal operation (such as patterns of acceleration, battery consumption, and stability), the interaction between UAS (such as safety requirements and the 'rules of the road'), and external events like electromagnetic interference, sudden gust turbulence, etc. Monitoring system-wide behavior (that is, the latter kind of inter-UAS interactions) is as challenging as it is *essential* for runtime safety, given the uncertainty inherent in UAS operations. Today, system-wide runtime monitoring is rarely used (see previous reports from this project). Still, low UAS density partially explains this, but the team notes that what is not monitored is also not known (there is a real problem with under-reporting UAS incidents), and that efficient error-free methods for runtime monitoring of UAS groups are not available. In this chapter, the team demonstrates precisely such a method, including its limitations and recommendations.

4.1 Executive Summary

The team developed a monitor synthesis algorithm for general distributed Cyber-Physical Systems, of which UAS groups are a special case. The team's monitor synthesis algorithms take in a requirement in formal temporal logic and produce a provably correct runtime monitor for the requirement. Critically, the team's monitors can handle drifting clocks in continuous time, run in an incremental fashion, and provide a verdict (safe/unsafe) at every moment in time. This is the first ever monitor synthesis algorithm for this setting. The team's analysis demonstrates that worst-case computational complexity is logarithmic in the size of the formula, exponential in the number of agents, and linear in the frequency of the agents' signals.

Recommended guidance: The team proposes that every UAS have onboard runtime monitors for continuous checking of its own health and operations, including distance to (static and moving) obstacles. These on-board monitors ought to be synthesized automatically from the formal requirements to ensure error-free code. For monitoring inter-UAS operation, the team needs research on establishing the degree to which clock drift can affect UAS operations for common

UAS platforms with off-the-shelf components and developing a regulatory framework for inter-UAS communication on safety-critical tasks (such as monitoring). The team recommends establishing baseline requirements on the on-board compute needed to process the necessary set of monitored signals. Given the correctness guarantees on the monitoring code, it is now possible to focus future analysis on the formal requirements themselves, and whether they capture designer intent comprehensively enough. The figure below illustrates the overall process.

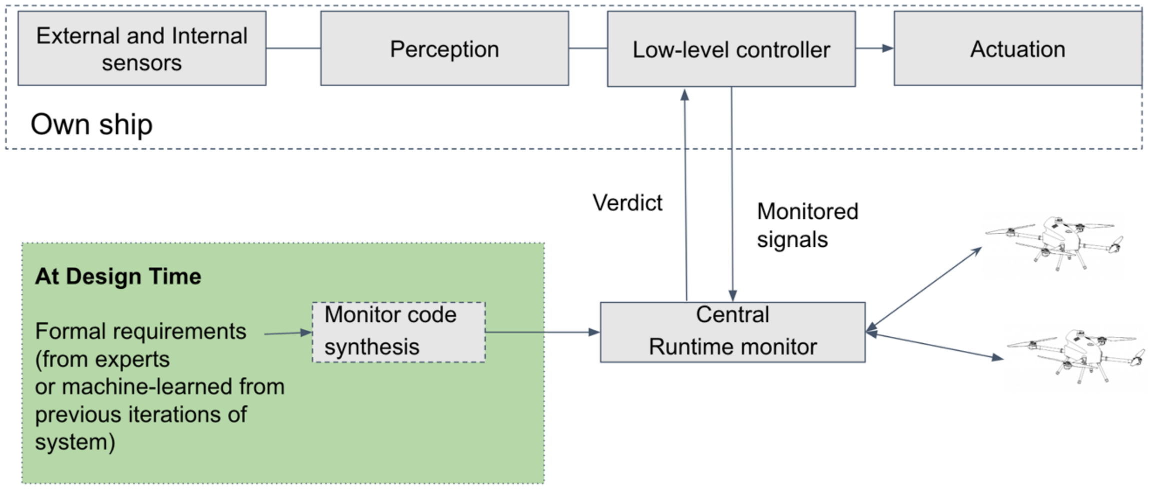


Figure 11. A process diagram for the runtime monitoring guidance.

4.2 Distributed Monitoring

Challenges face runtime monitors in the distributed UAS setup, which is a special case of the more general distributed cyber-physical system setup: first, as UAS include a physical aspect (e.g. the UAS' motion or battery level), the monitors measure and compute over signals in *dense time*, aka continuous time. So, for any finite time span, there are infinitely many "events" in the system. Second, the individual UAS that make up the distributed system each have a local clock, and these clocks drift from each other. Thus, when two UAS report a value of their signal at local time t, these two values are not necessarily synchronous. So, the monitor must find a reasonable interpretation of the temporal constraints, such as "two UAS must fly at each other for at most N seconds." In such a requirement, on which clock are the N seconds to be measured? Third, if a monitor returns only a single possible violation (or satisfaction) of the requirement or spec, the corresponding global state producing this violation may not provide enough information for debugging purposes, since other possible global states (other ways of resolving the concurrency) may reveal more or different violations; rather, a set of possible global states that violate the spec would be more useful.

In this work, the team addresses these challenges for partially synchronous distributed systems like UAS. Such systems use an algorithm like NTP (Mills et al. 2010) to keep their clocks within a known bound ε of each other; signal values occurring within an ε of each other might be concurrent. Given a temporal logic specification whose constraints refer to dense global time, which cannot be measured by the agents, the team's algorithm computes, to arbitrary precision, the set of all possibly concurrent moments that satisfy the specification. The team first generalizes satisfaction signals (Maler and Nickovic 2004) to this partially synchronous setting and express them as a function of *dense-time satcuts*, first introduced by (Koll et al. 2023), which are possibly concurrent moments on the drifting clocks. The team's algorithm then works by analyzing the geometry of partially synchronous multi-dimensional time and performs geometric manipulations on satcut polytopes. Finally, the team derives an online monitoring algorithm with provable approximation guarantees.

The team use Signal Temporal Logic (STL) (Maler and Nickovic 2004) as specification language. STL is widely used for specifying requirements of systems like UAS, such as "At every moment between 0 and 100 ms, a critical separation is followed, 6 to 10.5 ms later, by a negative acceleration." Using a formal temporal logic as a specification language allows us to avoid ambiguity of imprecise specs, as well as produce a monitoring algorithm which does not need to be redesigned when the spec changes.

4.2.1 Related work

Existing logics for specifying properties of distributed systems (Basin et al. 2011; Baumeister et al. 2021; Sen et al. 2004) do not preserve the abstraction of a single synchronized system for the control engineer designing the UAS. More work has been done on monitoring of distributed systems in general, most of it in discrete or logical time, such as (Fabre and Pigourier 2002; Ganguly et al. 2022; Tekken Valapil et al. 2017; Zhao et al. 2001). A result from (Chase and Garg 1998) shows that the complexity of monitoring such a system in general is NP-complete. Two papers (Koll et al. 2023; Momtaz et al. 2023) address monitoring dense-time distributed systems, but only for boolean predicates and not temporal logic specifications. Finally, (Momtaz, Abbas, and Bonakdarpour 2023) does online STL monitoring using an SMT solver, but only returns whether the spec is satisfied at time 0, while the team returns all such (possibly) concurrent moments, thereby computing the entire satisfaction signal. By considering a fragment of STL that still includes all temporal operators, the team provides a custom algorithm that avoids the uncertainty of SMT runtimes and their dependence on the particular SMT encoding. While the team has not yet implemented the algorithm, leaving a comprehensive experimental evaluation to follow-on work, the team is also able to characterize the algorithm's complexity directly in terms of meaningful quantities, like the number of agents and the quality of the team's approximations.

4.2.2 Contributions

In this work, the team provides three primary contributions:

1. Identify a fragment of STL that is amenable to monitoring over partially synchronous systems;
2. Provide an offline monitor that returns all possible satisfactions of a given formula;
3. Produce an online version of said monitor.

4.3 STL formulas

STL allows us to reason about specs that rely on changes over time. There are three common temporal operators that allow it to rely on these changes: $F_{[a,b]}$ (Eventually), saying that something eventually occurs at some moment a to b timesteps in the future; $G_{[a,b]}$ (Always), saying that something always occurs at the moment a to b timesteps in the future; and $U_{[a,b]}$ (Until), saying that something always occurs from now until some handover moment a to b timesteps in the future, at which moment the team has something else occur. The following are some examples of STL formulas:

"Sometime in the next 10 secs, x_1 will be positive and remain positive for at least 30 secs."

$$F_{[0,10]}G_{[0,30]}(x_1 > 0) \tag{1}$$

"At every moment 5 to 20 secs from now, x_1 will be positive within 2 secs."

$$G_{[5,20]}F_{[0,2]}(x_1 > 0) \tag{2}$$

"x_1 will be positive until some moment 5-10 secs from now, when x_2 or x_3 is negative."

$$(x_1 > 0)U_{[5,10]}((x_2 < 0) \vee (x_3 < 0)) \tag{3}$$

These examples are all STL formulas, but not all of them are practically monitorable. The team has identified a fragment of all STL formulas which are monitorable; they do not include example (2). In the team's fragment, temporal operators are only allowed on:

- the root of a formula (e.g., $G_{[0,4]}(x_1 > 0)$);
- both sides of an "or" (e.g., $(F_{[1,2]}(x_1 > 0)) \vee (G_{[3,4]}(x_2 > 0))$);
- one side of an "and" (e.g., $\left(F_{[0,3]}(x_2 > 0)\right) \wedge (x_1 > 0)$);
- inside an "Eventually" (e.g., example (1)); or
- the right side of an "Until" (e.g., $(x_1 > 0)U_{[1,3]}\left(F_{[2,3]}(x_2 > 0)\right)$).

An example of an unallowable formula would be $(F_{[0,2]}(x_1 > 0)) \wedge (F_{[1,3]}(x_2 > 0))$ (temporal operators on both sides of an "and").

With these restrictions on allowable specifications, the team can produce a monitor to identify satisfaction of a spec on a distributed UAS. There is still much flexibility in choosing a valid spec; both examples (1) and (3) are valid with these restrictions.

4.4 Monitoring framework

The team provides an offline monitor for the team's task of monitoring distributed UAS. This monitor is centralized, meaning all agents must communicate with the monitor to determine satisfaction or violation of a given formula. For a given formula with predicates $x_1 > 2$, $x_2 > 0$, etc., agents send the timestamps of when their signals cross their predicate boundaries. E.g., agent x_1 sends timestamps whenever its signal goes above or below 2, agent x_2 sends timestamps whenever its signal goes above or below 0, etc. The monitor uses this data to construct high-dimensional geometric shapes, combining them based on the formula to produce a final shape. The properties of this shape determine whether the team identifies satisfaction or violation of the formula. An example of this combination is shown in **Error! Reference source not found.**9. Combining the two shapes in the left plot produces the shape in the right plot when there is an "or" in the STL formula.

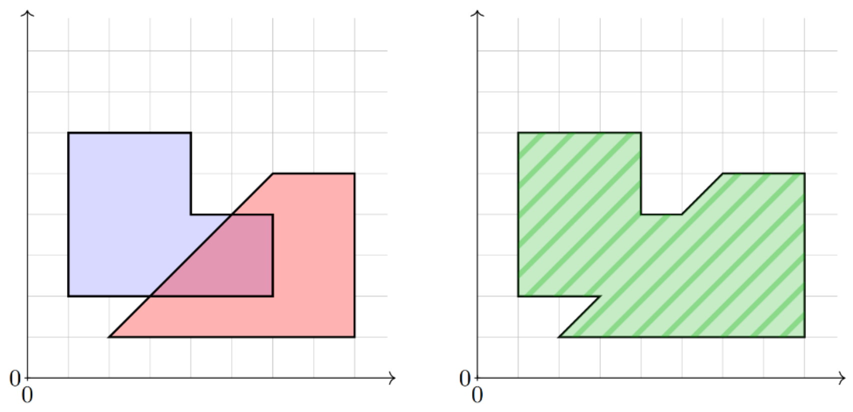


Figure 12. Combining shapes in the central monitor.

The monitor approximates satisfaction of the formula, but provides lower and upper bounds on this approximation. Furthermore, the algorithm for the monitor can further refine the approximation with additional time, allowing for as tight of an approximation as necessary for the application.

In this work, the team has also produced an online version of this offline monitor. At any given moment in time with the signals of the UAS up to this point, the online version can determine if there is satisfaction or violation of the given STL formula, or if more data is necessary for evaluation.

4.5 Complexity

Complexity of the offline monitor is based on the size of the formula, number of agents, and how often each agent's signal goes above or below its predicate boundaries (call this the frequency of the signal). It is logarithmic in the size of the formula (some operators in the formula can be parallelized), exponential in the number of agents (each agent adds a dimension to the combining shapes the team described in the previous section), and linear in the frequency of the agents' signals (every two timestamps of changes in signal value adds an extra combining shape). Further execution time can be added if the team chooses to tighten the resulting approximation.

The online version of the monitor has the same complexity as the offline version at each moment it evaluates satisfaction or violation. This means more regular evaluations lead to more execution time overall (linear in the number of evaluations).

4.6 References

Basin, David, Carlos Caleiro, Jaime Ramos, and Luca Viganò. 2011. "Distributed Temporal Logic for the Analysis of Security Protocol Models." *Theoretical Computer Science* 412 (31): 4007–43.

Baumeister, Jan, Norine Coenen, Borzoo Bonakdarpour, Bernd Finkbeiner, and César Sánchez. 2021. "A Temporal Logic for Asynchronous Hyperproperties." In *International Conference on Computer Aided Verification*, 694–717. Springer.

Chase, Craig M, and Vijay K Garg. 1998. "Detection of Global Predicates: Techniques and Their Limitations." *Distributed Computing* 11:191–201.

Fabre, E., and V. Pigourier. 2002. "Monitoring Distributed Systems with Distributed Algorithms." In *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, 1:411–16. Las Vegas, NV, USA: IEEE. https://doi.org/10.1109/CDC.2002.1184529.

Ganguly, Ritam, Yingjie Xue, Aaron Jonckheere, Parker Ljung, Benjamin Schornstein, Borzoo Bonakdarpour, and Maurice Herlihy. 2022. "Distributed Runtime Verification of Metric Temporal Properties for Cross-Chain Protocols." In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, 23–33. IEEE.

Koll, Charles, Anik Momtaz, Borzoo Bonakdarpour, and Houssam Abbas. 2023. "Decentralized Predicate Detection over Partially Synchronous Continuous-Time Signals." In *International Conference on Runtime Verification*, 213–30. Springer.

Maler, Oded, and Dejan Nickovic. 2004. "Monitoring Temporal Properties of Continuous Signals." In *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, 152–66. Springer.

Mills, David, Jim Martin, Jack Burbank, and William Kasch. 2010. "Network Time Protocol Version 4: Protocol and Algorithms Specification." Internet Engineering Task Force.

Momtaz, Anik, Houssam Abbas, and Borzoo Bonakdarpour. 2023. "Monitoring Signal Temporal Logic in Distributed Cyber-Physical Systems." In *Proceedings of the ACM/IEEE 14th International Conference on Cyber-Physical Systems (with CPS-IoT Week 2023)*, 154–65.

Momtaz, Anik, Niraj Basnet, Houssam Abbas, and Borzoo Bonakdarpour. 2023. "Predicate Monitoring in Distributed Cyber-Physical Systems." *International Journal on Software Tools for Technology Transfer*, 1–16.

Sen, K., A. Vardhan, G. Agha, and G. Rosu. 2004. "Efficient Decentralized Monitoring of Safety in Distributed Systems." In *Proceedings. 26th International Conference on Software Engineering*, 418–27. https://doi.org/10.1109/ICSE.2004.1317464.

Tekken Valapil, Vidhya, Sorrachai Yingchareonthawornchai, Sandeep Kulkarni, Eric Torng, and Murat Demirbas. 2017. "Monitoring Partially Synchronous Distributed Systems Using SMT Solvers." In *International Conference on Runtime Verification*, 277–93. Springer.

Zhao, Feng, Xenofon Koutsoukos, Horst Haussecker, James Reich, Patrick Cheung, and Claudia Picardi. 2001. "Distributed Monitoring of Hybrid Systems: A Model-Directed Approach." In *IJCAI*, 557–64. https://www.researchgate.net/profile/Horst-Haussecker/publication/2375562_Distributed_Monitoring_of_Hybrid_Systems_A_model-directed_approach/links/0a85e532884ac52f96000000/Distributed-Monitoring-of-Hybrid-Systems-A-model-directed-approach.pdf.

5 GUIDANCE FOR ROBUST INFERENCE IN UAS AUTOMATION

In Beyond-Visual-Line-Of-Sight (BVLOS) UAS operations, UAS rely heavily on statistical inference using measurements from onboard sensors, such as real time estimation of the system state. When critical sensors such as Global Positioning Systems (GPS) or Inertial Measurement Units (IMUs) are subject to sensor data quality issues (especially a sensor spoofing attack by an adversary), UAS may experience serious consequences, including loss of stability and handing over control to the adversary. In the literature review on robust inference for UAS automation, the team identified various vulnerabilities in UAS automation that may arise when sensors are subject to malicious spoofing attacks or data quality issues. Most existing mitigation strategies share the limitations that they are designed to counter relatively simple attacks, and they may fail in the presence of a more sophisticated and optimally designed attack (like a replay attack). In addition, most existing mitigation strategies are passive strategies, in that they are just focused on improving the design of the inference algorithms without any attempt to leverage other system components (e.g., actuators).

5.1 Executive Summary

The team identified a set of moderate sensor spoofing attack scenarios which can be launched in the real world by an adversary with moderate capability (e.g., using a low-cost hardware to transmit a Radio Frequency (RF) signal in the vicinity of a target UAS), and how the impact of the attacks on sensor data can be emulated in UAS simulations. The team also demonstrated that an active watermarking-based detector can be much more effective in detecting and mitigating more sophisticated sensor spoofing attacks compared to existing passive mitigation strategies, like anomaly detection, that are most commonly used today. In a physical watermarking countermeasure, the control signals for actuators are designed to embed a small magnitude watermark sequence, and the UAS response to the embedded watermark is analyzed to detect and mitigate sensor spoofing attacks.

Recommended guidance: The team recommends that UAS developers and relevant certification authorities leverage the identified set of moderate sensor spoofing attacks and their simulator; the developers to test their UAS in relevant contexts, and the certification authorities to evaluate the evidence submitted by a UAS operator in support of a CONOPs safety (e.g., “Did they submit evidence that they are resilient against the following attacks?”). The team recommends the use of physical watermarking techniques by UAS developers to enhance resilience against more sophisticated sensor spoofing attacks, like replay attacks, especially when the UAS are planned to be deployed to an adversarial environment. The watermarking-based detector can potentially detect covert sensor spoofing attacks that were designed to avoid detection by passive anomaly detectors.

5.2 Recommended Guidance: Evaluation Of UAS Inference Algorithms

5.2.1 Overview

Robust state inference is key for small UAS operating BVLOS. A broad survey of documented incidents shows that corrupted sensor data—most often through GPS spoofing or jamming and acoustic manipulation of Micro-Electro-Mechanical Systems (MEMS) gyroscopes—regularly propagates through the estimator and leads to loss-of-control events (A51 Task 1). These findings motivate a structured evaluation procedure that regulators and manufacturers can apply before field deployment. The recommended test plan concentrates on the sensor channels whose compromise is both straightforward to execute and severe in consequence—principally Global Navigation Satellite System (GNSS) and MEMS-based inertial sensors—so that estimator robustness is benchmarked against the threats most likely to occur in practice.

Task 2 revealed a critical gap in the current body of research: although the literature abounds with failure anecdotes and ad-hoc countermeasures, no standard, end-to-end procedure exists for evaluating small Unmanned Aircraft Systems (sUAS) state-estimation integrity against the catalogue of known threats. The attack surface is broad—ranging from GNSS spoofing to inertial falsification—so an objective test campaign must begin with a curated subset of threats chosen systematically. For each selected threat, validated attack models are required so that developers can reproduce the conditions in Software-In-The-Loop (SIL) or Hardware-In-The-Loop (HIL) testbeds with traceable fidelity. Addressing these two needs—threat selection, and threat emulation—is therefore essential to move from anecdotal defenses to a certifiable evaluation framework.

5.2.1.1 Threat selection

In the following guidance, a curated set of threats from the literature review in A51 Task 1 is provided. The threats are selected based on practicality and safety impact. Failure due to adversarial manipulation of GPS and IMU dominated the failure stories in the Task 1 report and are identified as practical to launch.

5.2.1.2 Threat emulation

For each threat, a physics-based or signal-level attack model is prescribed that could be used to emulate the impact of a sensor spoofing attack on the measurement data. With this structure, a consistent and traceable pathway for evaluating estimator robustness under representative adversarial conditions is established.

The following subsections set out (i) how an adversary of moderate capability can execute each threat in practice and (ii) how the same effects can be modelled in a SIL/HIL simulator so that estimator robustness can be evaluated.

5.2.2 GNSS ("GPS") Integrity Threats

Many sUAS autopilot systems use the GNSS data as the measurement data for Extended Kalman Filter (EKF) for state estimation. For example, in ArduPilot, its EKF2 module computes the estimate of latitude, longitude, ground speed, and vertical speed of the sUAS using the GNSS measurements (ArduPilot Dev Team 2025). The heavy reliance on these measurements has rendered GPS the most frequently exploited attack surface in the literature.

5.2.2.1 Jamming

Practical feasibility: A GPS-jamming attack can be carried out with surprisingly simple and inexpensive equipment. Off-the-shelf "privacy jammers"—small boxes that cost under USD 200—broadcast a blanket of radio noise on the same frequency that civilian GPS receivers listen to (the L1 band). When one of these devices is activated, drones within roughly 130 m often lose their satellite lock and can no longer calculate position or time (Ferreira 2020, Zidane 2024). The same result has been demonstrated with hobby-grade software-defined radios such as the USRP B210, using less than one watt of transmit power (Le Roy 2019, Deshmukh 2022). Consumer drone GPS units are designed to work with very weak satellite signals, so even a modest increase in background radio noise is enough to make them search for new satellites—or drop the GPS solution entirely—leaving the UAS to rely on less accurate backup sensors.

Simulation model

The impact of GPS jamming on sUAS absolute position $\boldsymbol{p} = [longitude, latitude, altitude]$ can be emulated as shown in Table 6.

Table 6. GPS jamming threat model.

Model	Jammed measurement	Model parameter/Typical range	Description/Reference
Additive white gaussian noise	$\boldsymbol{p_{jam}}(t) = \boldsymbol{p_{True}}(t) + \boldsymbol{n_j}(k)$ $\boldsymbol{n_j}(k) \sim \mathcal{N}(0, \sigma_p^2)$	σ_p^2 is the variance of the additive noise injected due to the jamming attack.	White gaussian noise is added to position measurements (Ferreira 2020, Zidane 2024)
Drop-out (Denial of Service attack)	$\boldsymbol{p_{jam}}(t) = NaN$	N/A	Drops Signal to Noise Ratio (SNR) of the incoming GPS signal below the receiver's acquisition threshold, causing loss of GPS signal (Saputro 2020).

5.2.2.2 Spoofing

Practical feasibility: A GPS-spoofing attack could be carried out by transmitting counterfeit satellite signals that are slightly stronger than the authentic ones received by a sUAS. A moderately resourced adversary can achieve this with a low-cost software-defined radio (e.g., USRP B-210, BladeRF) running publicly available GNSS-signal-generation software (Zidane 2024, Noh 2019).

Simulation model

The impact of GPS spoofing on absolute position $\boldsymbol{p}$ and absolute velocity $\boldsymbol{v}$ measurements can be modeled as shown in Table 7.

Table 7. GPS Spoofing threat model.

Model	Spoofed measurement	Model parameter/Typical range	Description/Reference
Bias injection	$\boldsymbol{p_{spoof}}(t) = \boldsymbol{p_{True}}(t) + \Delta\boldsymbol{p}$	$\Delta\boldsymbol{p}$ is injected absolute position offset typically in the range 5 to 20 m on lateral positions.	$\Delta\boldsymbol{p}$ bias is injected to true absolute position GPS measurements $\boldsymbol{p_{True}}(t)$ (Noh 2019)
Ramp	$\boldsymbol{p_{spoof}}(t) = \boldsymbol{p_{true}}(t) + \Delta\boldsymbol{V_{bias}}t$	$\Delta\boldsymbol{V_{bias}}$ is injected absolute velocity offset typically in the range 0.5 to 1 m/s on lateral velocities, which	$\Delta\boldsymbol{V_{bias}}$ bias is injected to true absolute velocity GPS measurements (Chen 2022)

	$\boldsymbol{v}_{\boldsymbol{spoof}}(t) = \boldsymbol{v}_{true}(t) + \Delta \boldsymbol{V}_{\boldsymbol{bias}}$	adds a ramp on the position measurement.	
Replay	$\boldsymbol{p}_{\boldsymbol{spoof}}(t) = \boldsymbol{P}_{\boldsymbol{true}}(t-\tau)$	τ is recorded measurement delay typically greater or equal to the attack duration.	Genuine absolute position measurements are replaced with previously recorded position GPS measurements (Gaspar 2019)

5.2.3 Gyroscope Integrity Threats

Gyroscopes provide high-rate measurements of sUAS's body angular rates. Autopilots like Ardupilot and PX4 use gyroscope measurements to help predict how the sUAS rotates in 3D space. These measurements are important because they help the system understand and control the vehicle's orientation as it moves forward. The gyroscope measurement noise is much smaller than the noise associated with heading aids such as the magnetometer or GPS yaw; the filter gives far more weight to these high-rate IMU updates and uses the slower sensors mainly to curb long-term drift. As a result, any bias or scaling error introduced into the gyroscope data stream shows up almost immediately in the estimated quaternion (or Euler angles) and then propagates to velocity and position estimates. A corruption of the MEMS gyro, therefore, can quickly compromise every downstream guidance and control function.

5.2.3.1 Jamming

Practical feasibility: Experimental evidence shows that a consumer-grade MEMS gyroscope can be disrupted with a simple acoustic source. Son (2015) demonstrated that directing a narrow-band tone at the sensor's resonant frequency—generated by a laptop, a low-power amplifier, and an inexpensive piezo-ceramic speaker—was sufficient to overwhelm the inertial readings on an AR.Drone 2.0. The attack produced uncontrolled roll- and pitch-angle excursions exceeding 30° and caused loss of control within twelve seconds. The low cost and off-the-shelf nature of the equipment confirms that effective gyro-jamming can be executed by an adversary with only moderate capability. (Son 2015)

Simulation Model

Table 8 shows how the impact of gyroscope jamming on angular rate measurements $\boldsymbol{\omega}$ can be emulated.

Table 8. Gyroscope jamming threat model.

Model	Jammed measurement	Model parameter	Description/Reference
Additive white	$\boldsymbol{\omega}_{\boldsymbol{jam}}(t) = \boldsymbol{\omega}_{True}(t) + \boldsymbol{\omega}_{\boldsymbol{j}}(k)$	σ_ω^2 is power of the injected noise	$\boldsymbol{\omega}_{\boldsymbol{j}}(k)$ a white Gaussian noise is added to angular rate

gaussian noise	$\boldsymbol{\omega_j}(k) \sim \mathcal{N}(0, \sigma_\omega^2)$		measurements (Son 2015, Tu 2018)
Drop-out (Denial of Service attack)	$\boldsymbol{\omega_{jam}}(t) = NaN$	N/A	Drops SNR of the gyroscopic measurement causing the EKF filter to reject gyroscope measurement (Son 2015, Tu 2018).

5.2.3.2 Spoofing

Practical feasibility: Tu (2018) showed that an adversary can use inexpensive ultrasonic hardware to do more than merely saturate a MEMS gyroscope; it can inject *controlled* rate errors such that it could steer the drone without touching its control interface. Using a small array of off-the-shelf ultrasonic transducers (total cost ≈ USD 120), the authors transmitted a "Side-Swing" waveform that imposed a steady average bias on BS NMI055 and BS BMI160 gyroscopes, which are typically used in sUAS such as CUAS and HolyBro Pixhawk variants. The modest cost and readily available hardware indicate that a technically competent hobbyist could carry out an effective gyroscope-jamming attack. Although tracking a moving sUAS is difficult, an attacker can readily align an acoustic beam while the sUAS hovers. Even a short attack window can introduce rate errors that may quickly affect the attitude control system, making gyro-spoofing a significant risk (Tu 2018).

Simulation Model

The impact of gyroscope spoofing on angular rate measurements $\boldsymbol{\omega}$ can be emulated as shown in Table 9.

Table 9. Gyroscope spoofing threat model.

Model	Spoofed measurement	Model parameter	Description/Reference
Bias injection	$\boldsymbol{\omega_{spoof}}(t) = \boldsymbol{\omega_{True}}(t) + \Delta\boldsymbol{\omega}$	$\boldsymbol{\Delta\omega}$ is injected angular rate vector offset typically in the range $0.01 - 0.05\ rad\ s^{-1}$	$\boldsymbol{\Delta\omega}$ bias is injected to true gyro angular rate measurements $\boldsymbol{\omega_{True}}(t)$ (Tu 2018)

5.2.4 Accelerometer Integrity Threats

Accelerometers supply sUAS flight computer with high-rate samples of specific force. For example, in both ArduPilot and PX4, these measurements are first integrated into velocities and then applied to the Extended Kalman filter's prediction step, thereby updating body-frame velocity and, through double integration, position. The nominal accelerometer process-noise density (≈ 0.06 m s^{-2} √Hz) is appreciably lower than the variances assigned to slower aids such as GPS speed, optical flow, or barometric altitude. Consequently, over intervals ranging from milliseconds to several seconds, the filter allows the velocity-driven prediction to dominate the velocity estimate, using external sensors primarily to correct long-term drift. Any bias or scale error injected into the accelerometer stream, therefore, appears immediately as a spurious change in estimated velocity, accumulates into position error, and can tilt the attitude solution through the gravity-vector coupling until the next GPS, vision, or barometric update realigns the filter. Bias faults of only ±0.1 m s^{-2} have been shown to yield position drifts of tens of meters when unmitigated (Kwon 2017), and additive accelerometer errors have been reported to destabilize Extended Kalman-filter fusion if left undetected (Saied 2021).

5.2.4.1 Jamming

Experiments have shown that simply flooding a MEMS accelerometer with acoustic energy at its resonant frequency can overwhelm the measurement channel and lead to loss of control. Son (2015) first noted that the same ultrasonic tone that destabilized a sUAS's gyroscope also injected erratic signals into the onboard accelerometer. Follow-up hardware-in-the-loop and live-flight tests by Jeong (2023) confirmed the risk: when a loudspeaker emitting the sensor's resonant tone was placed near the vehicle, every experiment ended in a crash. Because the attack relies solely on an inexpensive speaker operating in the 2–30 kHz band, a moderately equipped adversary can reproduce the effect whenever close access to the sUAS is possible.

Simulation Model

The impact of accelerometer jamming on body acceleration measurements $\boldsymbol{a}$ can be emulated as shown in Table 10.

Table 10. Accelerometer jamming threat model.

Model	Jammed measurement	Model parameter	Description/Reference
Additive white gaussian noise	$\boldsymbol{a_{jam}}(t) = \boldsymbol{a_{True}}(t) + \boldsymbol{a_j}(k)$ $\boldsymbol{\omega_j}(k) \sim \mathcal{N}(0, \sigma_a^2)$	σ_ω^2 is power of the injected noise	$\boldsymbol{a_j}(k)$ a white Gaussian noise is added to body acceleration measurements (Son 2015).
Drop-out (Denial of Service attack)	$\boldsymbol{a_{jam}}(t) = NaN$	Not applicable	Drops SNR of the accelerometer measurement, causing the EKF filter to reject accelerometer measurements.

5.2.4.2 Spoofing

Acoustic signals can also be shaped to bias or fully control the accelerometer output rather than merely saturating it. Trippel (2017) exposed 20 commercial MEMS accelerometer models, including those used in sUAS, to tone bursts at their resonant frequencies and achieved deliberate output shifts on three-quarters of the devices tested. Digital parts such as the ADXL350 could be forced to report ±10 g for one to two seconds, while analog devices with external ADCs sustained ±1 g readings for more than 30 seconds; in some cases, the attacker could steer the output indefinitely with peak-to-peak amplitudes up to 1 g. The required gear was no more sophisticated than an off-the-shelf ultrasonic speaker and a low-cost amplifier. Although the speaker must be brought within a few tens of centimeters of the drone, successful bias injection can tilt the estimated attitude and accumulate large position errors, making acoustic accelerometer spoofing a credible, low-cost threat.

Simulation Model

The impact of accelerometer spoofing on body acceleration measurements $\boldsymbol{a}$ can be emulated as shown in Table 11.

Table 11. GPS jamming threat model.

Model	Spoofed measurement	Model parameter/Typical range	Description/Reference
Bias injection	$\boldsymbol{a}_{\boldsymbol{spoof}}(t) = \boldsymbol{a}_{True}(t) + \Delta\boldsymbol{a}$	$\Delta\boldsymbol{a}$ is injected absolute position vector offset typically in the range 0.6 to 2 g on each axis (gravitational acceleration).	$\Delta\boldsymbol{a}$ bias is injected to true body acceleration accelerometer measurements $\boldsymbol{a}_{True}(t)$ (Trippel 2017).
Sinusoid	$\boldsymbol{a}_{\boldsymbol{spoof}}(t) = \boldsymbol{a}_{true}(t) + \Delta\boldsymbol{a}_{\boldsymbol{s}}$ $\Delta\boldsymbol{a}_{\boldsymbol{s}} = \mathbf{A}\sin(2\pi \mathrm{f_r} t + \phi)$	$\mathbf{A}$ is amplitude matrix of the injected body acceleration offset typically in the range 0.13 to 2 g on each axis. ϕ phase of the attack signal $\mathrm{f_r}$ represents the resonance frequency.	$\Delta\boldsymbol{a}_{\boldsymbol{s}}$ sinusoid bias is injected to true body acceleration measurements (Jeong 2023).
Overwrite	$\boldsymbol{a}_{\boldsymbol{true}}(t) = \boldsymbol{a}_{\boldsymbol{spoof}}(t)$	$\boldsymbol{a}_{\boldsymbol{spoof}}(t)$ replaces true accelerometer measurements $\boldsymbol{a}_{\boldsymbol{true}}(t)$ typically in the range 0.13 to 2 g.	Genuine absolute body accelerometer measurements are completely replaced by spoofed measurements (Trippel 2017).

5.3 Recommended Guidance: Active Detection for Robust Inference

5.3.1 Overview

The following guidance introduces a lightweight active-detection layer based on physical watermarking. The guidance explains how a low-energy, randomly generated excitation (referred to as watermark) can be superimposed on thrust-and-torque commands, folded into the estimator's covariance propagation, and used for detecting sophisticated sensor spoofing attacks that could otherwise stay covert under existing passive detectors.

This section first summarizes the background motivating the recommended guidance, then presents the rationale for augmenting passive χ^2 monitors with a physical watermark. Subsequent sections describe how small, zero-mean Gaussian excitations, i.e., watermarks, can be injected into the thrust-and-torque commands to enhance detectability of sophisticated sensor spoofing attacks such as replay attacks. Finally, validation evidence from simulations is compiled, quantifying gains in true-positive rate under measurement replay attacks. Taken together, the material equips integrators and regulators with a concise, low-overhead pathway for embedding active defense into existing sUAS inference chains.

5.3.2 Background

sUAS that operate BVLOS depend on onboard sensing and real-time state estimation to satisfy both safety and performance requirements. The survey conducted under A51 Task 1 established that compromised sensor streams—originating from either benign drift or deliberate manipulation—have repeatedly driven Kalman-Filter-based estimators and their attendant control loops to unsafe operating conditions without triggering existing alarms (A51 Task 1). The incidents most frequently involved the GNSS, MEMS IMUs, and Light Detection And Ranging (LiDAR)/optical-flow sensors, highlighting these channels as the primary contributors to state-estimation risk.

Mainstream commercial and open-source autopilots, notably ArduPilot and PX4, rely almost exclusively on passive mechanisms such as χ^2 innovation tests, fixed-window residual monitors, and cumulative-sum detectors to identify anomalous measurements (ArduPilot Dev Team 2025, PX4 Dev Team 2025). Task 2 demonstrated that such detectors can be defeated by adversaries who acquire knowledge of their static detection thresholds and then tailor attack signals to avoid detection (A51 Task 2). The GNSS record-and-replay attack exemplifies this vulnerability: by substituting previously valid pseudo-range data, an attacker can maintain residuals at nominal levels, thereby evading all passive checks (Lenhart 2022). Because these attacks become asymptotically undetectable under purely passive monitoring, a supplementary mechanism that compels the adversary to expose deviations is required.

5.3.3 Active Detection

Active detection techniques via physical watermarking provide a comprehensive solution to identifying and verifying the authenticity of measurement signals, ensuring that the integrity of the information is maintained and protected against unauthorized alterations. The defender intentionally perturbs the data acquisition or control channel (through random excitation, challenge–response patterns, or other “watermarks”) so that malicious manipulation, such as a replay attack, must distort a known fingerprint and thus reveal itself. It transforms the defender from a passive observer into an active interrogator, ensures that any attacker who wishes to remain hidden must pay a quantifiable performance penalty, and, most importantly, restores provable guarantees on state estimation integrity for safety-critical sUAS operations (Satchidanandan 2017). The guidance set forth herein, therefore, concentrates on active detection methodologies, with physical watermarking presented as an implementation-ready exemplar. Detailed recommendations are provided for watermark design, integration into established autopilot architectures.

Physical watermarking: By embedding a known, low-energy watermark—typically a small, random excitation added to control inputs—the operator imprints a hidden signature that any uncompromised measurement must contain. If a replay or injection attack erases or distorts this signature, a simple statistical test on the residue exposes the intrusion. Mo and Sinopoli (Mo 2009) formalized this idea, proving that properly designed physical watermarks guarantee detection of otherwise undetectable replay attacks while allowing an explicit trade-off between control performance and attack detection performance.

Physical watermarking could be adapted to sUAS by superimposing independent and identically distributed (i.i.d.) zero-mean Gaussian watermark signals $\boldsymbol{u_w}(k)$ on each element of the four-channel thrust–torque command vector $\boldsymbol{u_c}(k)$

$$\boldsymbol{u_c}(k) = \left[\, \tau_\phi \; \tau_\theta \; \tau_\psi \; T^b \right], \qquad \boldsymbol{u_w}(k) \; \sim \mathcal{N}(\boldsymbol{0}, \Sigma_w)$$

yielding the watermarked input

$$\widetilde{\boldsymbol{u}}(k) \quad = \boldsymbol{u_c}(k) + \boldsymbol{u_w}(k).$$

where Σ_w is the watermark parameter, τ_ϕ, τ_θ and τ_ψ are the 3D torques, and T^b is the body z-axis thrust, as shown in Figure 10.

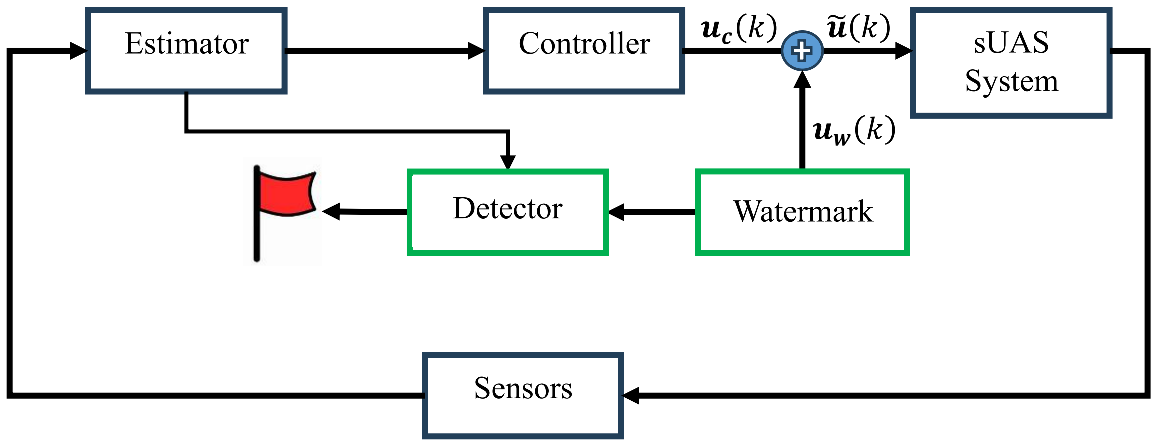


Figure 13. Working principle of physical watermarking.

Because the watermark is generated onboard, the technique requires no extra hardware or inter-UAS communication. Subsequent studies have extended the concept to design optimal watermarks to maximize detection performance, showing that watermarking can be made lightweight enough for embedded autopilots yet powerful enough to detect sophisticated sensor spoofing attacks (Liu 2022, Rubio-Hernan 2017, Satchidanandan 2017).

A stealthy attack, such as a replay attack, becomes detectable once the defender embeds a physical watermark into the control inputs: because the attacker cannot replicate the unpredictable watermark, the estimation residuals under attack acquire a systematic bias that causes the χ^2 statistic to exceed its nominal threshold. However, this active interrogation incurs a trade-off: the watermark's magnitude must be large enough to produce a discernible signature in the sensor stream—improving detection probability—yet small enough that the added perturbation does not unduly degrade control performance. In practice, increasing the watermark variance enhances separation between the genuine and replayed residual distributions (thereby shifting the Receiver-Operating-Characteristic [ROC] curve upward), but it also injects energy into the closed-loop system, lengthening settling times, increasing control effort, and introducing slight oscillations around the setpoint. Thus, the cost of physical watermarking is a modest reduction in tracking fidelity and increased actuator activity, which must be balanced against the required detection sensitivity when tuning the watermark parameters.

VALIDATION

A high-fidelity Six-Degree-Of-Freedom (6-DOF) sUAS simulation environment that captures aerodynamic effects on thrust and torque generation implemented in the *asbQuadcopter* Simulink project was modified and used to validate the proposed active-detection framework (MathWorks 2025). Realistic sensor noise and control loops are emulated, and replay attack—known to bypass conventional passive monitors—is injected. Low-energy physical watermarks are superimposed on the control inputs, and detection performance is evaluated by comparing residual statistics with and without watermarking. Through these experiments, the ability of the active scheme to expose otherwise undetectable adversaries is quantitatively demonstrated.

5.3.3.1 sUAS Closed Loop Model

sUAS system model: The plant is a high-fidelity 6-DOF rigid-body model of a parrot rolling spider mini quadrotor that was adapted from *asbQuadcopter* Simulink project (MathWorks 2025).

- **Rigid-body dynamics** are captured with Earth-fixed positions, Euler attitudes, body-frame velocities, and angular rates (12 states in total). The dynamics is driven by 3D forces and torques generated by the rotor dynamics.

$$\boldsymbol{x}(t) = [x^I\ y^I\ z^I \phi\ \theta\ \psi\ \dot{x}^b\ \dot{y}^b\ \dot{z}^b\ p\ q\ r]$$

- **Rotor/actuator dynamics** capture the aerodynamic effects on thrust and torque generation such as blade flapping, induced drag, and rotor Coriolis effect. These blocks respect the hardware bounds (-500–500 rad s^{-1}) and include a configurable saturation layer so that controller outputs realistically clip before reaching the motors (F. Riether 2016).
- **Aerodynamic and environmental effects** are injected as process-noise terms: white, zero-mean Gaussian sequences routed through dedicated disturbance ports to represent unmodelled aerodynamics and stochastic wind gusts. Their covariance can be scaled or disabled on a run-by-run basis, giving us fine control of uncertainty levels during Monte-Carlo experiments.

The entire plant runs at a 10 kHz fixed-step rate and is wrapped in Simulink Fast-Restart to accelerate large-scale batch simulations.

Sensors are modelled as direct feedback of the full 12 states of the system. In the Simulink implementation, the sensor suite is represented by a single lumped block that returns the entire 12-state vector—position, Euler attitude, body-frame velocities, and angular rates—at **1 kHz,** i.e., ten times slower than the 10 kHz step used for integrating the plant, thereby mimicking the continuous-to-discrete gap present in real flight computers. Each channel is corrupted by independent zero-mean white Gaussian noise whose variances are drawn from Parrot Rolling-Spider-class datasheets (≈ 3 cm for position, 0.15° for attitude, 2 cm s^{-1} for body velocity, 0.02° s^{-1} for angular rate).

Controller architecture: The existing cascaded controller in the *asbQuadcopter* model—a position outer loop driving an attitude Proportional–Integral–Derivative (PID) inner loop—was adapted and retuned to achieve a 0.7 s rise time for a unit step during hover (MathWorks 2025). The controller includes a control-allocation layer that converts desired body torques and thrust into individual motor speed commands using an analytically inverted mixer matrix. Output saturations and rate limiters sit immediately upstream of the actuator blocks to capture real electronic speed controller behavior.

Estimator Model: An EKF is provided as a MATLAB Function block running at 1 kHz within the Simulink loop and fully integrated with the *asbQuadcopter* dynamics: The block outputs the state estimate and the estimation residue vector, which feeds a passive χ^2 anomaly detector.

Anomaly detector model: A conventional χ^2 innovation detector, widely adopted in small-UAS autopilots, has been embedded inside the 1 kHz Simulink loop. At each step the residue vector computed from sensor measurements $\boldsymbol{z}(k)$ and the predicted measurements $\hat{\boldsymbol{z}}(k)$

$$\boldsymbol{r}(k) = \boldsymbol{z}(k) - \hat{\boldsymbol{z}}(k \mid k - 1)$$

is formed by the EKF, and the quadratic statistic

$$\boldsymbol{g}(k) = \boldsymbol{r}^{\top}(k)\boldsymbol{S}^{-1}(k)\boldsymbol{r}(k)$$

is computed, where $\boldsymbol{S}(k)$ denotes the theoretical innovation covariance propagated by the estimator. Under nominal conditions $\boldsymbol{g}(k)$ follows a χ^2 distribution with m degrees of freedom where m denotes the dimension of $z(k)$, i.e., the number of measurements; thresholds are therefore selected from the cumulative χ^2 table to realize a prescribed false-alarm probability α (typically $\leq$ 5 %).

Detection performance for a χ^2 innovation detector is typically assessed with **receiver-operating-characteristic analysis**, which provides a threshold-independent view of how well the statistic $\boldsymbol{g}(k)$ separates nominal behaviour from attacks. In practice, many stochastic realisations of both **benign flight** and **adversarial scenarios** are generated—often by Monte-Carlo simulation—to capture the full distribution of $\boldsymbol{g}(k)$ under each hypothesis. The complete time series of $\boldsymbol{g}(k)$ is logged for every run, after which a notional decision threshold is swept across the range of observed values. For each threshold, the **True-Positive Rate** (TPR) (probability of correctly signaling an attack) and the **False-Positive Rate** (FPR) (probability of raising a false alarm) are tallied, and the resulting TPR–FPR pairs trace out the ROC curve. Curves that bow closer to the upper-left corner—or equivalently exhibit a larger area under the curve—indicate stronger discriminative power.

5.3.3.2 Replay Attack

The experiment implements a record-and-playback replay attack by capturing a segment of genuine sensor outputs during steady-state flight (e.g., hover) and then substituting the live measurements with this recorded buffer. Because each replayed sample corresponds to a previously valid trajectory, the estimator's innovation residuals remain statistically consistent with nominal noise bounds. As a result, the passive χ^2-based detector—relying on fixed residual thresholds—registers no anomaly, rendering the attack effectively invisible. Practical examples could include when a construction sUAS is hovering to inspect a building or a delivery sUAS hovering to drop off a package. In the Simulink simulator, all 12 state measurements are recorded and stored during steady-state hovering. During the attack, the true measurements of the sUAS are replaced by recorded measurements.

5.3.3.3 Physical Watermarking

In the simulator, the watermark was injected *after* the lower-level attitude controller but *before* the external motor mixer and actuator saturations, so that it traversed the full vehicle dynamics exactly as any legitimate command would.

At steady-state hovering, the 3D torques are close to zero and only change to compensate for process noise, while the thrust vector is a constant value canceling the gravitational force to maintain altitude. As a result, the variance of the watermark added to the thrust is orders of magnitude larger than the watermark added to the torques.

$$\Sigma_w = diag\,(0.001\sigma_w^2, 0.001\sigma_w^2, 0.001\sigma_w^2, \sigma_w^2)$$

Hence, the watermark covariance Σ_w is parametrized by the watermark variance added on the thrust σ_w^2.

Introducing a low-energy Gaussian watermark into the thrust-and-torque commands preserves the structure of the existing χ^2 innovation detector while *amplifying* its discriminatory power. Because the watermark is generated onboard, its realization is fully known to the estimator, so the extra excitation is folded into the propagated innovation covariance $\boldsymbol{S}(k)$; under nominal flight, therefore, the statistic $\boldsymbol{g}(k)$ still follows its expected χ^2 distribution and the prescribed false-alarm rate remains unchanged. When an adversary replaces genuine measurements with forged data—via measurement replay—the counterfeit stream cannot replicate the unseen watermark dynamics. This mismatch increases the magnitude of $\boldsymbol{g}(k)$, shifting the ROC curve upward and leftward, widening the gap between true-positive and false-positive rates, thereby making the replay attack detectable. In effect, the watermark acts as a built-in challenge that turns the passive χ^2 test into an *active interrogator*, revealing attacks that would otherwise remain statistically indistinguishable from normal operation.

5.3.3.4 Experiment Setup

In the simulation experiment, to capture the effect of watermark on popular autopilots like Ardupilot and PX4, the simulator was set to follow a similar architecture with a high fidelity 6-DOF system model, full state sensor measurements reflecting physical sensor noise, EKF for full state estimation, χ^2 anomaly detector, and cascaded PID controller.

i) Impact of Watermarks on Attack Detection Performance

In the simulation experiment, the UAS is set to hover at 1m altitude. Two scenarios were simulated. In the non-attack scenario (the null hypothesis), the UAS stays hovering, and the sensor measurements are intact. In the attack scenario (the alternative hypothesis), the replay attack is launched while UAS is set to hover. After three seconds from the attack start, the chi-square anomaly detector decision is checked to evaluate the attack detection performance. 300 Monte Carlo runs were performed for each hypothesis to obtain the ROC curve. The experiment was repeated for each watermark variance tested.

Detection performance is characterized by receiver-operating-characteristic (ROC) curves. After each run, the detection threshold is swept across the full range of $\boldsymbol{g}(k)$; for each threshold, the TPR and FPR are computed, giving a TPR-versus-FPR locus. ROC is after the 3s window to highlight how, without a watermark, replay attacks become asymptotically undetectable, whereas watermarked cases retain detectability.

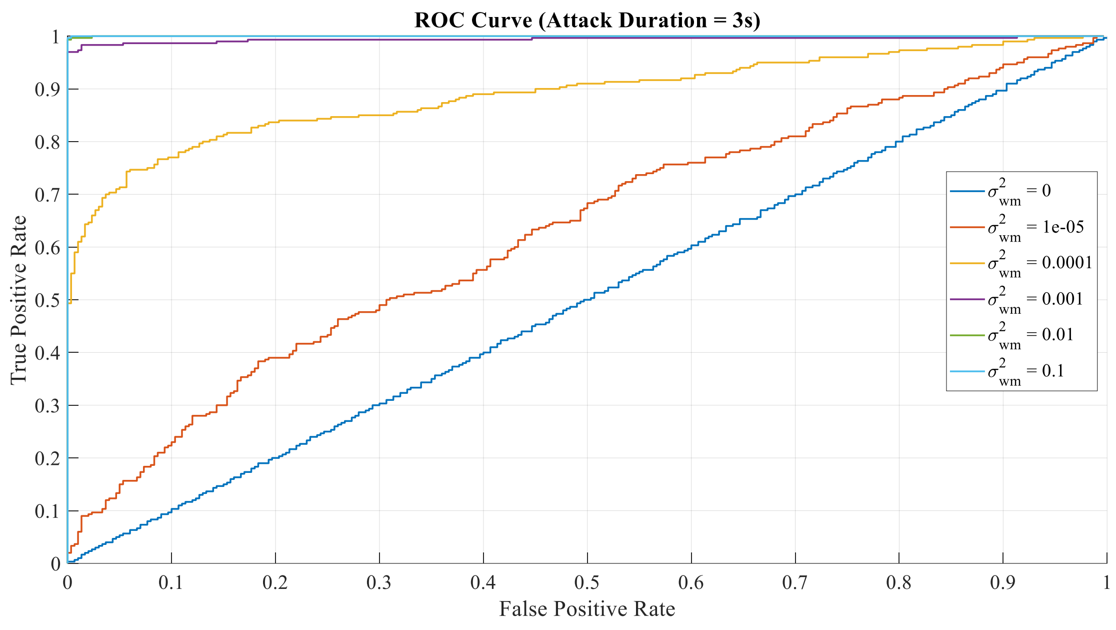
ii) Impact of watermark on controller performance

The impact of embedded watermarks on controller performance was examined using a controlled hovering scenario. The simulated sUAS was commanded to maintain a 1 m hover with zero lateral displacement; process noise representing mild wind and unmodelled aerodynamics, together with sensor-level measurement noise, were set to nominal fair-weather values. After the vehicle reached steady state, the watermark signal was superimposed on the thrust-and-torque command vector, and full state trajectories were logged.

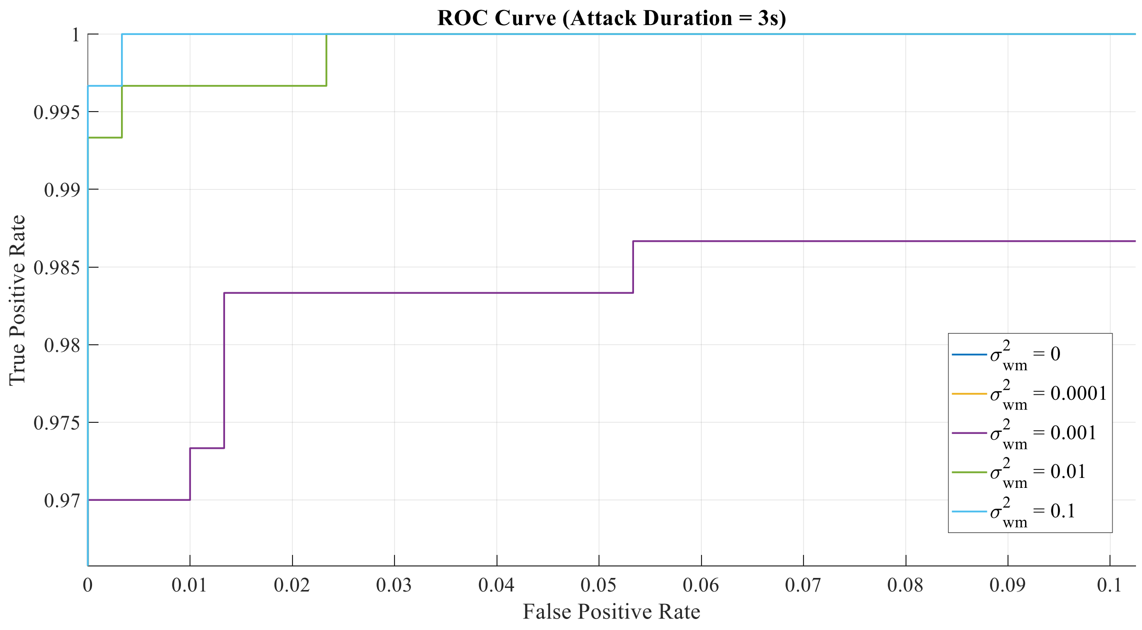
A sweep over watermark variances was then performed. For each setting, the average position and attitude tracking error, as well as the position and attitude trajectory, were compared with the baseline no-watermark hover, establishing the baseline required to select watermark amplitudes that preserve flight tracking fidelity while still embedding a detectable signature.

5.3.3.5 Results

Figure 11 presents ROC curves for the χ^2 innovation detector under a 3s replay-attack scenario, comparing cases with and without physical watermarking. Here, each curve plots the TPR against FPR as the decision threshold varies. In Figure 11a, the blue curve labeled $\sigma_{wm}^2 = 0$ corresponds to the no-watermark baseline; it coincides with the diagonal line ($TPR \approx FPR$), indicating detection performance equivalent to random guessing and confirming the asymptotic undetectability of replay attacks under purely passive monitoring. Adding a very small watermark ($\sigma_{wm}^2 = 10^{-4}$) bends the curve sharply upward, delivering a true-positive rate of roughly 70 % at 5 % false-positive budget, as shown in Figure 11a. Raising the watermark variance to 10^{-3} moves the curve into the upper-left corner depicted in Figure 11b, exceeding 98 % TPR across the relevant FPR range. In short, even modest watermark energy transforms the legacy χ^2 monitor from effectively blind to replay attacks into a detector with near-perfect discrimination against stealth adversaries.



a. Full scale ROC curves.



b. Top right corner of the ROC curves.

Figure 14. ROC curve for baseline and watermarked sUAS input evaluated 3s after the onset of a replay attack affecting all sensor measurements on the sUAS.

The reference tracking error of the controller increases as expected for all controlled outputs as the watermark variance was increased for all control inputs, as shown in Figure 12. This result shows the price in terms of control degradation of adding watermarks to control inputs. Increasing the watermark beyond a certain value could result in significant performance degradation as shown in Figure 12. However, depending on the CONOP, small watermark variances below 0.01 may still offer acceptable tracking error in a mildly adversarial environment while providing significant gain in revealing stealthy adversaries as shown in Figure 11.

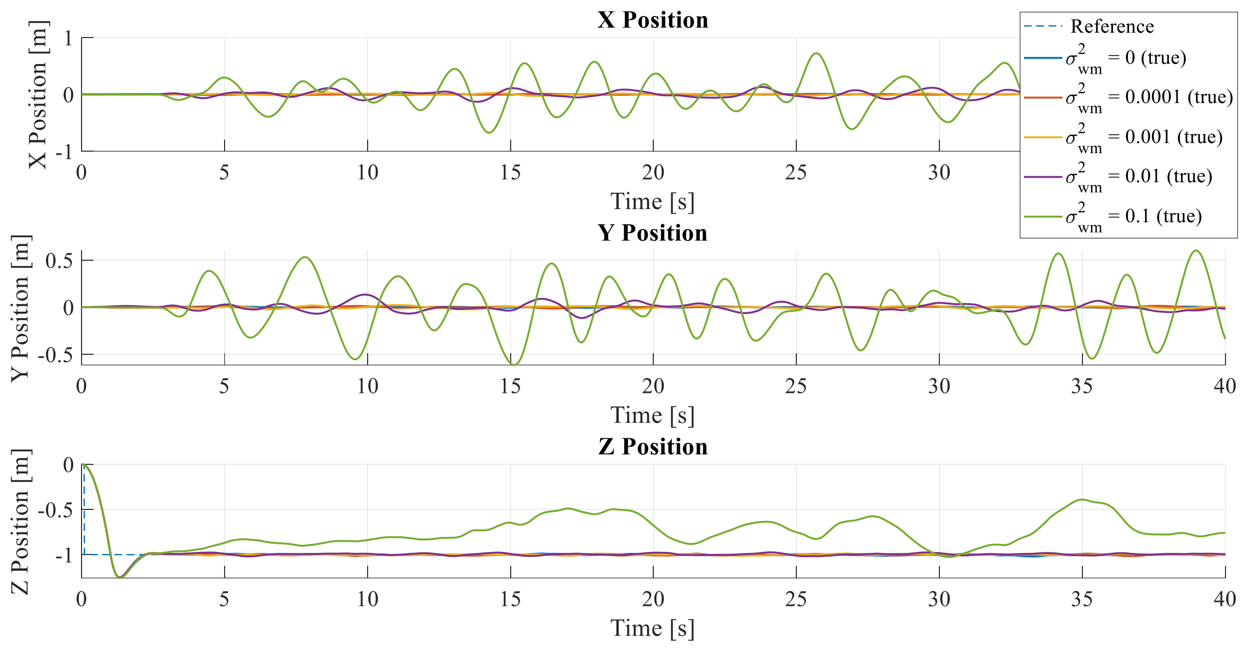


Figure 15. Impact of superimposing σ^2_{wm} i.i.d watermark to the thrust and torque input of the UAS starting at 2.5s on ground truth and reference position plots.

5.4 Conclusion

The team recommended two sets of guidance related to robust inference for UAS. The first guidance provides a standard framework for evaluating robustness of inference techniques in the presence of moderate sensor spoofing attacks, which were identified through an extensive literature review. The measurement falsification model provided in this guidance can be used by UAS engineers/operators to emulate the impact of sensor spoofing attacks in their simulation environments and evaluate resilience of the UAS automation against the attacks. The second guidance recommends the use of a physical watermarking technique to enhance the detectability of sophisticated sensor spoofing attacks, such as the measurement of replay attacks when the UAS is expected to be deployed to an adversarial environment. The team validated the guidance using a Simulink simulator; use of a physical watermarking technique significantly enhanced detectability of the measurement replay attacks in the validation experiment. In the meanwhile, it is worthwhile to note that the physical watermark used in the validation experiment (a white Gaussian process) is the simplest design for watermarks. It is expected that the gain in attack detectability can be made much more significant by considering more advanced designs of the physical watermark, such as dynamic watermarking (Satchidanandan 2017), hidden state generated watermarks (Mo 2015), natural watermarks (Ozel 2017), non-stationary, and non-gaussian watermarks (Rubio-Hernan 2017).

5.5 References

ArduPilot Dev Team. (2025). EKF2—Rate Gyro Bias Stability. ArduPilot Documentation.

Bhatti, J. A., & Humphreys, T. E. (2019). Hostile control of consumer drones via software-defined-radio jamming and spoofing. GPS World, 30 (2), 34–41.

Borio, D., Dovis, F., Kuusniemi, H., & Lo Presti, L. (2016). Impact and detection of GNSS jammers on consumer-grade satellite-navigation receivers. Proceedings of the IEEE, 104 (6), 1233–1245.

Cao, Y., Xiao, C., Cyr, B., Zhou, Y., Park, W., Rampazzi, S., Chen, Q. A., Fu, K., & Mao, Z. M. (2019). Adversarial sensor attack on LiDAR-based perception in autonomous driving. Proceedings of the 26th ACM SIGSAC Conference on Computer and Communications Security (CCS 2019), 2267–2281.

Chen, W., Dong, Y., & Duan, Z. (2018). Attacking altitude estimation in drone navigation. IEEE INFOCOM 2018—Computer Communications Workshops (INFOCOM WKSHPS), Honolulu, HI, 888–893. dblp.org

Choi, H., Lee, W.-C., Aafer, Y., Fei, F., Tu, Z., Zhang, X., Xu, D., & Deng, X. (2018). Detecting attacks against robotic vehicles: A control-invariant approach. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS 2018), 801–816. sigsac.org

Davidson, D., Wu, H., Jellinek, R., Singh, V., & Ristenpart, T. (2016). Controlling UAS with sensor-input spoofing attacks. 10th USENIX Workshop on Offensive Technologies (WOOT 16), August.

Deshmukh, S. and Sharma, V., (2022)"An SDR-based anti-drone system with Detection, Tracking, Jamming, and Spoofing Capabilities," 2022 IEEE Microwaves, Antennas, and Propagation Conference (MAPCON), Bangalore, India, 2022, pp. 388-393, doi: 10.1109/MAPCON56011.2022.10046968.

Feng, Z., Guan, N., Lv, M., Liu, W., Deng, Q., Liu, X., & Yi, W. (2017). Efficient drone hijacking detection using onboard motion sensors. Design, Automation & Test in Europe Conference & Exhibition (DATE 2017), 1414–1419.

Ferreira, R., Gaspar, J., Sebastião, P. et al. Effective GPS Jamming Techniques for UAS Using Low-Cost SDR Platforms. Wireless Pers Commun 115, 2705–2727 (2020). https://doi-org.oregonstate.idm.oclc.org/10.1007/s11277-020-07212-6

Garrett, I., & Gerdes, R. (2020). On the efficacy of model-based attack detectors for unmanned aerial systems. Proceedings of the ACM Workshop on Cyber-Physical Systems Security (AutoSec 2020), 13–24.

Humphreys, T. E., Ledvina, B. M., Psiaki, M. L., O'Hanlon, B. W., & Kintner, P. M. (2012). Assessing civilian GPS spoofing vulnerability: Demonstration at White Sands Missile Range. Navigation, 59 (1), 35–44.

Jeong, J., Kim, D., Jang, J., Noh, J., Song, C., & Kim, Y. (2023). Un-Rocking drones: Foundations of acoustic injection attacks and recovery thereof. 30th Network and Distributed System Security Symposium (NDSS 2023), February.

Le Roy, F., Roland, C., Le Jeune, D., & Diguet, J.-P. (2019). Risk assessment of SDR-based attacks with UAS. In Proceedings of the 16th International Symposium on Wireless Communication Systems (ISWCS) (pp. 222–226). IEEE. https://doi.org/10.1109/ISWCS.2019.8877144

Lenhart, M., Spanghero, M., & Papadimitratos, P. (2022). DEMO: Relay/Replay attacks on GNSS signals. arXiv Preprint, arXiv:2202.10897.

Liu, H., Li, Y., Han, Q.-L., & Ra, T. (2022). Watermark-based proactive defense strategy design for cyber-physical systems with unknown-but-bounded noises. IEEE Transactions on Automatic Control, 67 (4), 1973–1988.

MathWorks. (2025). asbQuadcopter Simulink Project [Simulink model]. Aerospace Blockset Examples, MATLAB R2025a. The MathWorks, Inc.

Mo, Y., & Sinopoli, B. (2009). Secure control against replay attacks. 47th Annual Allerton Conference on Communication, Control, and Computing, 911–918.

Mo, Y., Weerakkody, S., & Sinopoli, B. (2015). Physical authentication of control systems: Designing watermarked control inputs to detect counterfeit sensor outputs. IEEE Control Systems Magazine, 35 (1), 93–109.

Noh, J., Kwon, Y., Son, Y., Shin, H., Kim, D., Choi, J., & Kim, Y. (2019). Tractor Beam: Safe-hijacking of consumer drones with adaptive GPS spoofing. ACM Transactions on Privacy and Security, 22 (2), Article 12.

Ozel, O., Weerakkody, S., & Sinopoli, B. (2017). Physical watermarking for securing cyber-physical systems via packet-drop injections. 2017 IEEE International Conference on Smart Grid Communications (SmartGridComm), 271–276.

PX4 Development Team. (2025). EKF2/ECL Design. PX4 Documentation.

Riether, F. (2016). Agile quadrotor maneuvering using tensor-decomposition-based globally optimal control and onboard visual-inertial estimation (Master's thesis). Massachusetts Institute of Technology.

Rubio-Hernan, J., De Cicco, L., & Garcia-Alfaro, J. (2017). On the use of watermark-based schemes to detect cyber-physical attacks. EURASIP Journal on Information Security, 2017 (1), 8.

Saied, M., Tabikh, A. R., Francis, C., Hamadi, H., & Lussier, B. (2021). An informational approach for fault-tolerant data fusion applied to a UAS's attitude, altitude and position estimation. IEEE Sensors Journal, 21 (24), 27766–27778.

Satchidanandan, B., & Kumar, P. R. (2017). Dynamic watermarking: Active defense of networked cyber-physical systems. Proceedings of the IEEE, 105 (2), 219–240.

Scott, L., & Wing, M. (2016). Protecting civil GPS receivers from personal-privacy jammers: Field measurements and mitigation techniques. Proceedings of the ION GNSS+ 2016, 2377–2389.

Son, Y., Shin, H., Kim, D., Park, Y., Noh, J., Choi, K., Choi, J., & Kim, Y. (2015). Rocking drones with intentional sound noise on gyroscopic sensors. 24th USENIX Security Symposium (USENIX Security 15), 881–896.

Trippel, T., Weisse, O., Xu, W., Honeyman, P., & Fu, K. (2017). WALNUT: Waging doubt on the integrity of MEMS accelerometers with acoustic injection attacks. 2017 IEEE European Symposium on Security and Privacy (EuroS&P), April, 3–18.

Tu, Y., Lin, Z., Lee, I., & Hei, X. (2018). Injected and delivered: Fabricating implicit control over actuation systems by spoofing inertial sensors. 27th USENIX Security Symposium (USENIX Security 18), 1545–1562.

Wesson, K. D., Humphreys, T. E., Shepard, D. P., & Evans, B. L. (2012). An analysis of low-power GPS jammers and their impact on civilian navigation receivers. Proceedings of the ION GNSS 2012, 2609–2619.

Zidane, Y., Silva, J. S., & Tavares, G. (2024). Jamming and Spoofing Techniques for Drone Neutralization: An Experimental Study. Drones, 8(12), 743. https://doi.org/10.3390/drones8120743

6 GUIDANCE FOR SENSORS IN UAS AUTOMATION

The team studied the critical challenge of collision avoidance in UAS operations, focusing specifically on a sensor systems perspective. The team highlighted the important role of sensor modalities and sensing infrastructure in ensuring reliable and safe UAS navigation, especially in dynamic or constrained environments.

Optical sensors (such as cameras) are widely used on UAS platforms due to their ability to capture rich visual information, including images and video feeds. These sensors support key tasks such as object recognition, obstacle detection, and visual navigation. However, they have inherent limitations that constrain their effectiveness in real-world scenarios. Their performance is highly weather-dependent, often degraded by fog, rain, or low-light conditions. Furthermore, they have limited range, are susceptible to obstacle occlusion, and are vulnerable to optical interference. These factors significantly impact the reliability of collision detection and avoidance when relying solely on onboard optical sensing.

Thus, while single-sensor improvements continue, there's a need to design and evaluate networked sensor approaches to detect-and-avoid tasks. Rather than relying solely on onboard sensors, the team envisions UAS as participants in a broader, networked sensing environment that enables more robust and context-aware navigation decisions.

6.1 Executive Summary

The team proposes an augmented sensing infrastructure where UAS can leverage the following emerging wireless positioning and communication technologies to improve accuracy, reliability, and coverage:

- GNSS, which remains the backbone of UAS localization, but struggles in urban canyons and under signal jamming.
- Signal mapping techniques, where UAS exploit known signal landscapes for location inference.
- Reconfigurable Intelligent Systems (RIS) enable programmable reflection and propagation of radio signals, offering greater flexibility in signal routing and enhancement.
- Received Signal Strength (RSS)-based localization provides a low-cost and infrastructure-light alternative for estimating distance and position.

The team discusses the use of ubiquitous 4G/5G-based cellular localization systems in a new geofencing-based approach for localizing UAS flight paths, and how to leverage digital twin-enabled Multiple Input Multiple Output (MIMO) networks, where massive fingerprinting and real-time emulation of physical environments allow for centimeter-level precision in complex spaces. To mitigate complexity, the team proposes that the airspace be quantized into 3D cubic cells, each with a fixed spatial dimension. UAS are restricted to moving between these cells, akin to virtual flight corridors. The team proposes Long Range (LoRa)-based localization systems for GPS-denied environments, particularly for emergency response scenarios where infrastructure is limited or compromised.

Recommended guidance: To achieve the desired levels of safety in a grid airspace using the above combination of localization technology and infrastructure, the team recommends:

- Pilot deployments of 3D cubic cell systems: Begin with grid sizes of approximately 20 meters x 20 meters x 20 meters for suburban and rural trials and 10m x 10m x 10m in urban corridors or high-density UAS routes.
- Develop Artificial Intelligence-powered geofencing managers that integrate signal mapping and sensor health indicators to enforce dynamic no-fly zones.
- Standardize sensing fault classification protocols across UAS platforms and encourage FAA and International Civil Aviation Organization alignment in fault management documentation.
- Incorporate multi-sensor feedback loops where redundancy is shared across swarm participants via local mesh networks.
- Promote open data formats and Application Programming Interfaces to integrate RIS-enhanced infrastructure sensing into both commercial and public UAS operations.

6.2 Introduction and Background

6.2.1 UAS Sensors and Fault Detection

UAS depend heavily on a variety of onboard and external sensors to achieve autonomous functionality, including flight stabilization, obstacle avoidance, altitude control, and route adherence. These sensors include but are not limited to: GPS, IMU, barometers, airspeed sensors, cameras, LiDAR, and radar. Together, they form a sensory suite critical for system awareness, decision-making, and closed-loop control.

However, the reliability of these sensors under various conditions remains a major concern in UAS safety. Several common types of sensor faults can compromise system functionality:

- **Bias Fault**: A constant deviation between the measured and actual values.
- **Drift Fault**: A gradual deviation over time.
- **Freezing Fault**: The sensor output remains constant regardless of actual changes.
- **Loss of Accuracy**: Sporadic inaccuracies due to interference or degradation.

In addition to sensor faults, UAS also face actuator faults, like lock-in-place, float, and hardover conditions, which can severely impact flight behavior. Addressing sensor faults proactively is essential for maintaining control integrity.

Multiple approaches are used for Fault Detection and Isolation (FDI) in UAS:

Hardware Redundancy: Using multiple sensors of the same type to cross-validate readings.

Analytical Redundancy: Employing mathematical models (e.g., observers, Kalman filters) to estimate correct values.

Machine Learning & Neural Networks: Leveraging models such as EMRAN or GA-BP networks for adaptive, nonlinear detection.

Studies have demonstrated success using model-based estimators, sensor fusion, and intelligent switching between redundant systems. Fault-tolerant control can enable recovery by isolating faulty sensors and relying on backups or inferred values. Ongoing research suggests that coupling real-time fault diagnosis with cooperative sensing strategies across UAS swarms can improve detection robustness. Moreover, integrating Failure Modes and Effects Analysis (FMEA) and Fault Tree Analysis (FTA) into system design supports prioritization of sensor reliability upgrades.

In this report, the sensors discussion serves to underscore that resilient collision avoidance is not merely a matter of detection algorithms but also of sensor health, redundancy, and fault handling throughout the UAS's lifecycle

6.2.2 UAS Sensor Roles and Reliability

UAS rely on a suite of internal and external sensors that are critical for safe navigation, automation, and fault-tolerant operation. These include GPS, IMU (Inertial Measurement Unit), LiDAR, radar, cameras, airspeed sensors, and more. The performance and reliability of these sensors directly influence flight stability, control accuracy, and collision avoidance effectiveness. Sensor reliability remains a key concern in UAS, with fault types including bias, drift, freezing, and loss of accuracy. For example, prior studies have shown failure rates of up to 2.2×10^{-2} failures-per-flight-hour for certain UAS platforms, highlighting the importance of both hardware and analytical redundancy. As UAS autonomy increases, so does the need for unsupervised and cooperative sensor fault detection mechanisms.

Contemporary approaches to sensor fault detection in UAS include the use of Kalman filters, observer models, neural networks (such as Extended Minimum Resource Allocating Networks), and hardware-based fault isolation schemes. These techniques aim to either detect faults in real time or accommodate them through model-based estimation and intelligent switching between redundant sensor inputs.

In this report, the role of sensors is revisited not only from a collision detection perspective but also from a reliability engineering lens. The framework outlined includes cubic cell quantization and infrastructure-based sensing built upon the foundation of a resilient multi-sensor system.

As part of future implementation, considerations include:

- Fault detection algorithms using observer/Kalman filtering
- Adaptive redundancy and background sensor calibration
- Cooperative detection through networked UAS sensing nodes
- Risk modeling via FMEA and FTA

All of these are used to quantify sensor-related risks and mitigation impact. This integrated approach ensures that UAS sensing is not just responsive but also proactively resilient to sensor anomalies, enabling safer autonomous operation in increasingly complex environments.

6.2.3 The Sensor Perspective on UAS Collisions

As UAS usage expands globally across logistics, infrastructure inspection, disaster relief, and aerial photography, the threat of mid-air collisions, flyaways, or restricted airspace violations becomes more pressing. These incidents are often tied to limitations in environmental perception or a UAS's inability to respond to unforeseen obstacles. Traditionally, the response to collision threats has focused on flight path optimization and reactive control algorithms. However, this report repositions the problem through a sensor's lens, emphasizing the sensing modalities and networked infrastructure necessary to predict, detect, and avoid collisions reliably.

6.2.3.1 Limitations on Optical Sensing

Visible-light optical sensors, such as cameras, remain the backbone of most UAS platforms due to their low cost, compactness, and capacity to capture detailed visual data. They support operations such as object tracking, surveillance, and photogrammetry. Nevertheless, their real-world deployment is significantly hampered by:

- **Environmental Vulnerability**: Inclement weather such as fog, rain, or snow can obscure visibility and disrupt optical signal acquisition.

- **Line-of-Sight Limitations**: Optical sensors require a clear path to detect and classify obstacles, making them less effective in cluttered or dynamic environments.
- **Computational Demands**: Processing image or video feeds for real-time navigation requires high compute resources and robust embedded vision systems.

Thus, while useful, these sensors alone cannot support the high-reliability demands of widespread autonomous UAS deployment.

6.3 Alternative and Complementary Elements of a Sensor Infrastructure

6.3.1 Radar Systems

Radar technology, traditionally used in automotive safety and defense, is becoming increasingly vital in UAS navigation. Unlike optical systems, radars are robust to adverse weather and can detect objects through fog, dust, or even some materials

6.3.1.1 Long-Range Radar

Long-range radars transmit RF signals that reflect off distant objects and return to the receiver, measuring time delay and Doppler shift to estimate range and velocity. In UAS, these radars are ideal for forward-looking operations such as early obstacle detection, flight stabilization, and coordinated swarm behavior. The Continental ARS51x and Texas Instruments AWR1642BOOST offer capabilities including high angular resolution, fast processing time, and integration-ready modules suitable for UAS payloads.

6.3.1.2 Short-Range Radar

Short-range radars are optimized for immediate spatial awareness, such as takeoff/landing assistance or maneuvering in narrow airways. The Decawave DWM1001, which employs Ultra-Wideband (UWB) pulses, achieves decimeter-level precision in enclosed or cluttered spaces. These modules consume low power, are compact, and offer real-time location tracking suitable for swarming drones or warehouse UAS operations.

6.3.2 Wireless Positioning and Communication

Wireless infrastructure offers an untapped potential for UAS positioning and coordination. Modern wireless localization systems use signal characteristics—such as phase, amplitude, Received Signal Strength Indicator, or time-of-flight—to deduce a device's position relative to known transmitters.

6.3.2.1 Global Navigation Satellite Systems

Although GNSS is a foundational tool in UAS navigation, it suffers from severe performance degradation in urban canyons, tunnels, dense forests, or during intentional signal jamming. Standard GPS systems exhibit position errors ranging from 1 to 10 meters, insufficient for high-precision autonomous navigation.

6.3.2.2 Signal Mapping and RSS Localization

Signal mapping constructs a geo-spatial model of signal strength from known transmitters across a region. UAS then compares real-time measurements to the map to infer position. RSS-based localization, while less precise than GPS or UWB, offers a cost-effective method in areas with dense wireless coverage or in GNSS-denied environments.

6.3.2.3 Reconfigurable Intelligent Systems (RIS)

RIS technologies manipulate electromagnetic waves using tunable reflective elements. These passive/low-power surfaces can redirect RF signals to overcome line-of-sight issues or multipath fading. RIS can effectively improve communication robustness and facilitate beam steering for UAS flying in urban, hilly, or indoor zones.

6.3.2.4 Cellular and Base Station Localization

By triangulating signals from multiple 4G/5G base stations, UAS can localize themselves without GNSS. This method supports hybrid positioning systems, where cellular signals act as backups when satellite navigation is unreliable. Cellular localization scales well across urban environments and integrates seamlessly with existing telecom infrastructure.

6.3.2.5 Digital Twin-Enabled MIMO Networks

Massive MIMO systems combined with digital twins—a real-time emulation of the physical world—enable extremely accurate localization. Fingerprinting techniques leverage signal patterns at specific locations to match UAS measurements and determine position with centimeter-level accuracy. This is especially powerful in environments with rich signal diversity and structural complexity.

6.3.2.6 LoRa-Based Localization

LoRa is a low-power, wide-area network protocol that allows for lightweight, long-distance communication. LoRa localization methods are advantageous in GPS-denied environments such as subterranean, mountainous, or post-disaster zones. With a range exceeding 10 km in open terrain, LoRa modules can keep UAS networked in sparse infrastructure conditions.

To synthesize the strengths and trade-offs across the sensor modalities explored, the team presents a comparative performance table. This table includes key evaluation criteria such as accuracy, effective range, power consumption, and known limitations. The intent is to highlight how no single sensor meets all needs, underscoring the importance of a hybrid approach.

Table 12. Sensor System Performance Comparison.

Sensor Type	**Accuracy**	**Range**	**Limitations**
Optical Camera	Low-Medium(image-dependent)	Short(~50m)	Weather/lighting sensitivity
LiDAR Lite v2	High(~1-2cm)	Up to 40m	Requires line of sight, moderate power draw
Long-Range Radar (ARS51x)	Medium(~10-30cm)	Over 250m	Bulky, expensive

Short-Range Radar (DWM1001, UWB)	High(~10-30cm)	100-300m	Indoor bias, limited coverage area
GNSS (ZED-F9P)	Low-Medium(1-10m)	Global	Susceptible to urban canyon signal loss
Digital Twin-Enabled MIMO	Very High (cm-Level)	Environment wide	Requires extensive fingerprinting and simulation
LoRa-Based Localization	Medium (meters)	2-15km	Low data rate, infrastructure dependency

This table supports the selection of sensor technologies tailored to application needs. For example, LoRa may be ideal in emergency response zones, while MIMO fingerprinting is more suited to precision-demanding urban operations. It is advisable to integrate these findings into the UAS system design pipeline and adapt sensor usage dynamically based on mission profile.

6.4 Proposed 3D Cubic Cell Flight Lane Structure

In this proposed system, airspace is quantized into 3D cubic cells, each with a fixed spatial dimension. UAS are restricted to moving within and between these cells, akin to virtual flight corridors.

This concept draws from the notion of digital highways in the sky, akin to how traditional vehicles are guided by lane markers and roadways. By discretizing space into predefined volumes, UAS no longer operate in an unbounded 3D space but instead within a structured grid. These lanes in the sky enable systematic navigation, predictable motion, and collision-free routing.

Each cube acts as a virtual checkpoint and is governed by access control protocols. Only one UAS (or a safety-compliant number) can occupy a cell at a time, and transitions between cells must satisfy precondition checks, including:

- No imminent entry by another UAS;
- Adequate clearance from static or dynamic obstacles;
- Continuous localization updates to verify position;
- Spatial quantization also facilitates discrete vertical layering. UAS can fly at specific altitude levels like floors in a building, therefore ensuring separation of mission types (e.g., delivery vs. emergency response) or traffic density.

Advantages of this method include:

Predictability: With all UAS adhering to grid-aligned paths, movement is no longer random or ad-hoc but coordinated and scheduled.

Deconfliction: Automated systems can monitor and manage which UAS are in which grid zones, issuing commands to delay, reroute, or hold as necessary.

Interoperability: Enables integration into larger UAM ecosystems where drones, air taxis, and emergency UAS share airspace.

Safety Assurance: The grid system supports logical and physical geofencing, rapidly detecting and reacting to violations.

Moreover, each cell can be overlaid with metadata from the environment, such as RF signal maps, obstacle presence, or priority zones. This metadata, updated in real-time, enhances context-aware navigation. In future implementations, artificial intelligence could autonomously allocate cells and forecast traffic congestion across the grid using historical UAS motion data. For instance, machine learning could optimize flight paths to minimize energy usage while avoiding likely bottlenecks.

The implementation of 3D cubic flight lanes thus marks a pivotal shift in how airspace can be safely democratized for mass UAS operations, making use of existing sensing, positioning, and communication technologies to ensure robust, rule-driven autonomy in the sky.

6.4.1 Grid Construction and UAS Tracking

Each cell is assigned a globally unique identifier (Cell ID), typically generated from its position in the 3D matrix using its x, y, and z indices. This enables seamless lookup, referencing, and routing algorithms. The airspace is then partitioned into zones or volumes, each comprising hundreds or thousands of such cells. These volumes can represent air corridors over urban areas, restricted government zones, emergency lanes, or commercial delivery routes.

Once the grid is defined, UAS use a combination of positioning technologies (GNSS, RSS, MIMO, INS, and LoRa) to determine their real-time position. This position is continuously mapped to a grid cell by dividing the UAS's coordinates by the defined cell size and rounding to the nearest integer. Each UAS's state is stored and updated in a centralized or distributed airspace management system, often tied to a UAS Traffic Management platform.

Advanced implementations may incorporate:

- Spatial indexing techniques (e.g., octrees, k-d trees) to optimize position-to-cell mapping.
- Temporal granularity where time slots are assigned to each UAS for specific cells to prevent temporal collisions.
- Flight reservation systems where UAS "book" their path through a series of connected cells.
- Edge computing, enabling local computation of grid occupancy and trajectory planning in real-time with minimal latency.

To ensure safe transitions, the UAS's onboard system must check that the next desired cell is unoccupied and that transition metrics (e.g., speed, altitude, risk factor) are within threshold. If not, rerouting or loitering behavior is triggered. The continuous data fusion of environmental cues with cell assignments supports that dynamic geofencing areas can be instantly restricted or released for safety or emergency events without requiring manual UAS intervention.

6.4.2 Operational Advantages

The adoption of a structured 3D grid brings numerous advantages to real-world UAS operations, transforming previously ad-hoc free flight into a manageable, scalable, and safe aerial ecosystem.

- *Weather-Resilient Navigation*: With radar and RF-based technologies integrated into the sensing loop, UAS are no longer reliant solely on optics or GNSS. Even during signal attenuation or low visibility, accurate cell determination is maintained.
- *Simplified Conflict Resolution*: Instead of computing complex 3D collision vectors, the system only needs to monitor occupancy of adjacent or upcoming cells. This simplifies deconfliction algorithms significantly.
- *Dynamic Mission Planning:* UAS can adapt their missions in-flight. For example, a drone assigned to deliver a package can be rerouted via a different cell corridor if traffic is detected ahead, or if a priority mission (like a medical delivery) is assigned precedence.
- *Smart Zoning*: Specific altitude layers or columns of cells can be designated for specific use cases: commercial, recreational, emergency response, or inspection. This logical zoning reduces interference and streamlines regulation.
- *Redundancy and Handover Support*: As UAS move between grid zones governed by different infrastructure (e.g., moving from a LoRa-enabled rural area to a cellular-enabled urban core), the system supports seamless handover without loss of positional integrity.
- *Digital Twin Integration*: Each grid cell can be linked with a digital twin environment that predicts weather, signal quality, or crowding—guiding UAS to safer, more efficient paths in real time.
- *Regulatory Transparency*: By monitoring which UAS occupy which cells and at what times, regulators gain unprecedented insight into aerial behavior, allowing for data-driven rulemaking and rapid incident response.

The grid system transforms the sky into an organized infrastructure space, not unlike the road and rail systems people rely on. It opens the door to federated, safe, and intelligent management of increasing UAS traffic, laying the groundwork for truly autonomous aerial mobility at scale.

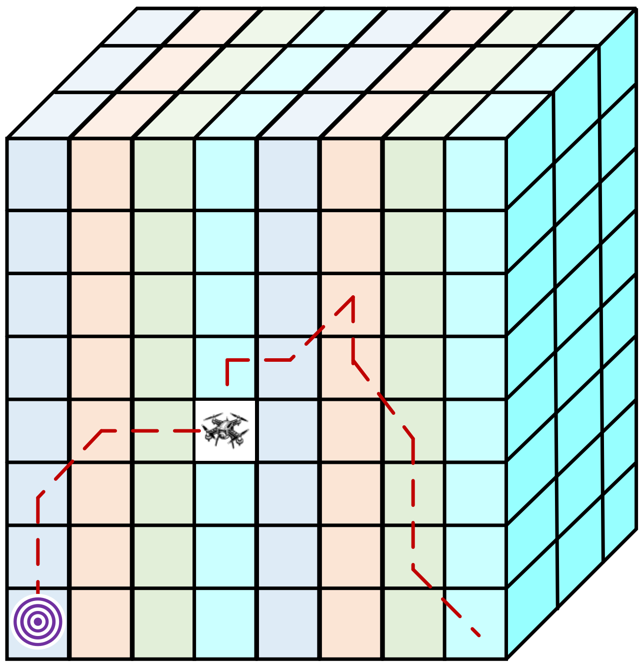


Figure 16. Structured 3D Airspace for UAS Navigation Using Cubic Cell Quantization.

6.5 Conclusion

The evolution of UAS operations is driving a transformation in how we understand and manage airspace. This report has presented a comprehensive examination of how hybrid sensing systems leveraging both onboard sensors and infrastructure-enhanced technologies can dramatically improve UAS collision avoidance and operational reliability. Traditional reliance on standalone optical sensors and GPS is insufficient in modern scenarios involving urban density, GPS-denied zones, and weather-challenged environments. Through a thoughtful integration of radar systems, LoRa localization, digital twin-enabled MIMO, and RSS-based signal mapping, UAS can maintain situational awareness far beyond the limits of their individual components. The proposed approach promotes a cooperative and resilient sensing ecosystem, where UAS dynamically respond to signal environments and infrastructure feedback in real time.

Crucially, the adoption of a 3D cubic cell airspace framework introduces a scalable, modular solution to the problem of airspace crowding and deconfliction. By segmenting the aerial environment into discrete volumes, UAS gain structure and predictability in flight, paving the way for standardized air traffic management and regulatory compliance.

Key enhancements explored in this chapter include:

- Embedding FDI mechanisms at the sensor level using analytical redundancy, Kalman filters, and neural network estimators.
- Implementing real-time geofencing based on hybrid localization inputs, including signal strength, RIS-enhanced coverage, and infrastructure overlays.

- Quantizing space into structured 3D volumes, reducing the probability of mid-air collisions, and supporting UAM models.

This convergence of localization, control, and sensing unlocks the potential for high-density UAS operations without compromising safety or efficiency. It forms the basis of what could evolve into a digitally managed "air traffic grid," accessible to delivery drones, emergency responders, and survey platforms alike.

Adopting this vision can elevate UAS deployment from reactive obstacle avoidance to a proactive, infrastructure-synchronized navigation paradigm that is intelligent, self-correcting, and ready for global scalability.

7 WIND-INDUCED ROTATIONAL DISTURBANCES IN URBAN AIRSPACES

7.1 Executive Summary

The proliferation of urban drone operations has intensified the need to understand and mitigate wind-induced hazards around buildings in complex urban environments. This study employed high-resolution Computational Fluid Dynamics (CFD) simulations across four diverse U.S. cities (New York City, Chicago, Los Angeles, Dallas) to simulate wind variability over complex urban canyons. The University of Kansas integrated these wind fields into flight dynamic models, quantifying control challenges such as oscillatory roll, yaw disturbances, and trajectory tracking errors. The results identified critical urban wind phenomena, such as corner vortices, shear layers, and channeling effects that significantly compromise drone stability, especially at the corners of high-rise structures and complex urban corridors.

The varying levels of trajectory deviation observed under different wind intensities (and reported in Tables 18 and 19 in Chapter 8) illustrate how street-level flow patterns, turbulence, and wind channeling effects can significantly alter UAS behavior—even with adaptive controllers in place. The following guidance therefore strengthens the case for integrating high-resolution urban wind modeling into UAS navigation systems. By simulating site-specific wind fields in advance, operators and systems can better anticipate areas of high disturbance, adjust standoff distances, and improve both trajectory planning and safety margins

Recommended guidance: Based on the CFD simulations of urban wind fields, the following guidance strategies are recommended to improve UAS safety in complex urban environments.

- High-resolution urban wind modeling should be integrated into UAS control simulations to enhance navigation performance. By incorporating site-specific wind field simulations in advance, operators and autonomous systems can better anticipate areas of significant disturbance, adjust standoff distances accordingly, and optimize both trajectory planning and safety margins.

- Sharp 90-degree turns near building corners should be avoided. Instead, smooth arc trajectories or offset corners should be used to reduce exposure to complex wind patterns revealed by the CFD.

- Before flight, operators should perform pre-flight assessments of wind conditions using available hyperlocal sensors or real-time forecast tools to anticipate gust potential and identify periods of calmer conditions.

This guidance can be integrated into distance recommendations and real-time UAS decision support systems. It also aligns with the FAA's urban integration goals, offering a scientifically grounded roadmap for enhancing urban UAS safety and reliability.

7.2 Introduction and Background

The growing use of UAS in urban environments, ranging from package delivery to emergency response, presents a new set of safety challenges. As drones are increasingly deployed in low-altitude airspace, understanding the aerodynamic complexities of urban settings becomes critical. Unlike rural or open environments, cities create unique airflow disruptions due to tall buildings, street canyons, and irregular infrastructure. These structures interact with natural wind patterns to generate intense turbulence, gusts, and shear layers that may compromise UAS stability, navigation, and control.

Urban wind flows are highly sensitive to microscale factors such as building height variability, street orientation, and surface heating. Prior studies (e.g., Chrit and Majdi. 2022, Chrit et al. 2023, Oke, 1987; Coceal et al., 2007; Fernando, 2010) have highlighted how urban morphology can lead to highly localized wind behaviors that standard meteorological tools fail to capture. Microscale modeling approaches like CFD provide the resolution necessary to analyze these effects. Research by Britter and Hanna (2003) and Tominaga and Stathopoulos (2013) demonstrated the effectiveness of CFD in capturing vortex shedding, flow separation, and wake dynamics in complex built environments. These models have also been validated against wind tunnel experiments and field data.

With UAS operations expanding rapidly, especially in metropolitan areas, the FAA has identified urban integration as a key focus. This study supports that effort by predicting wind-induced hazards in urban environments. By combining CFD simulations with control modeling performed by KU and real-world operator feedback, this research can provide actionable guidance to help mitigate the risks of urban drone operations.

7.3 Methodology

This research employed a combination of high-resolution CFD modeling and expert-informed operational analysis to evaluate wind-related risks in urban environments. The methodology consisted of the following core components:

- **Urban Geometry and Model Setup:** 3D models of New York City (Atlantic Heliport, Manhattan), Downtown Chicago, Downtown Los Angeles, and Downtown Dallas were reconstructed using building geometry data from OpenStreetMap. Each model captured key features such as building height, orientation, and density, which are critical to resolving microscale wind flows. The models included representative downtown areas known for existing or proposed UAS operational activity (e.g., vertiports).
- **CFD Wind Field Simulation:** The CFD simulation approach was inspired by Chrit and Majdi (2022), using a coupled mesoscale-microscale framework. The CFD simulations employed the SIMPLE algorithm (Chrit and Majdi 2022), capturing microscale urban wind phenomena such as flow separation, shear, and vortex shedding within the roughness sublayer. The mesh was refined near building edges and rooftops, providing detailed resolution of critical flow structures relevant to UAS stability.
- **Cross-City Case Study Comparison:** The four cities were selected for their variation in urban morphology and climate. A comparative framework was used to prevent site-specific wind guidance and provide robust operational guidance.
- **UAS Control Modeling**: The CFD-derived wind fields were integrated by the University of Kansas team into UAS flight dynamic models to assess control challenges, focusing on roll, pitch, and yaw deviations and quantify the separation distance from building corners.
- **Expert Interviews:** Incorporated insights from UAS and urban wind experts to validate findings and refine guidance.

7.4 Results and Discussion

- **New York City:** CFD simulations of the East 34th Street Vertiport in Manhattan revealed strong wind variability over short distances, driven by the dense building layout and proximity to the East River. Figure 14 shows how the canyon-like structure of narrow streets and high-rise walls created accelerated flows and frequent wind shifts, while vortices formed behind tall structures may destabilize UAS during lateral and vertical maneuvers. Figure 14 also confirms the presence of strong shear, which complicates hover stability and approach paths.

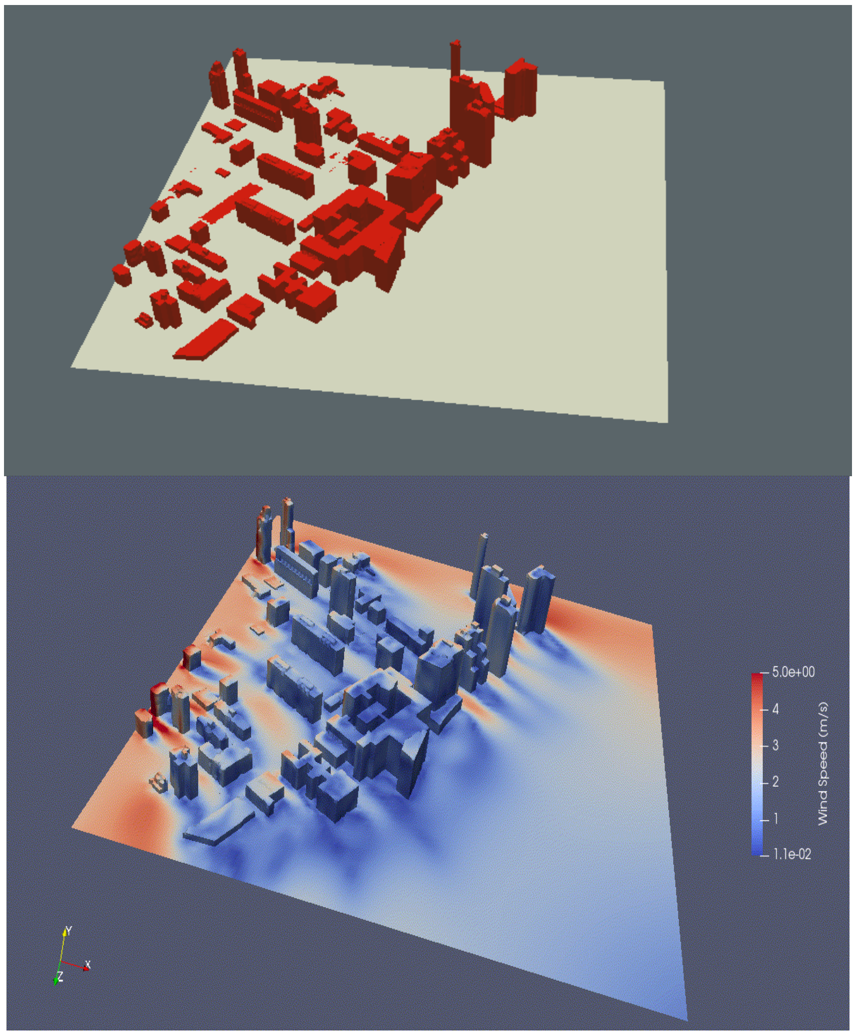


Figure 17. Results from CFD simulation of urban wind patterns in New York City.

- **Chicago:** In Chicago, CFD simulations indicated substantial wind variation. The lake breezes may also introduce additional vertical wind shear, which intensifies in late afternoon hours. Figure 15 highlights the wind acceleration along west-east avenues, creating corridor effects similar to NYC, with notable shear lines present. Wind convergence and Venturi effect (wind acceleration) were especially observed near intersections where buildings of different heights created uneven flow profiles. The transition between wide streets and densely packed zones created localized hazardous zones that must be accounted for during low-altitude UAS operations, especially at building corners, near riverfronts, or near elevated structures.

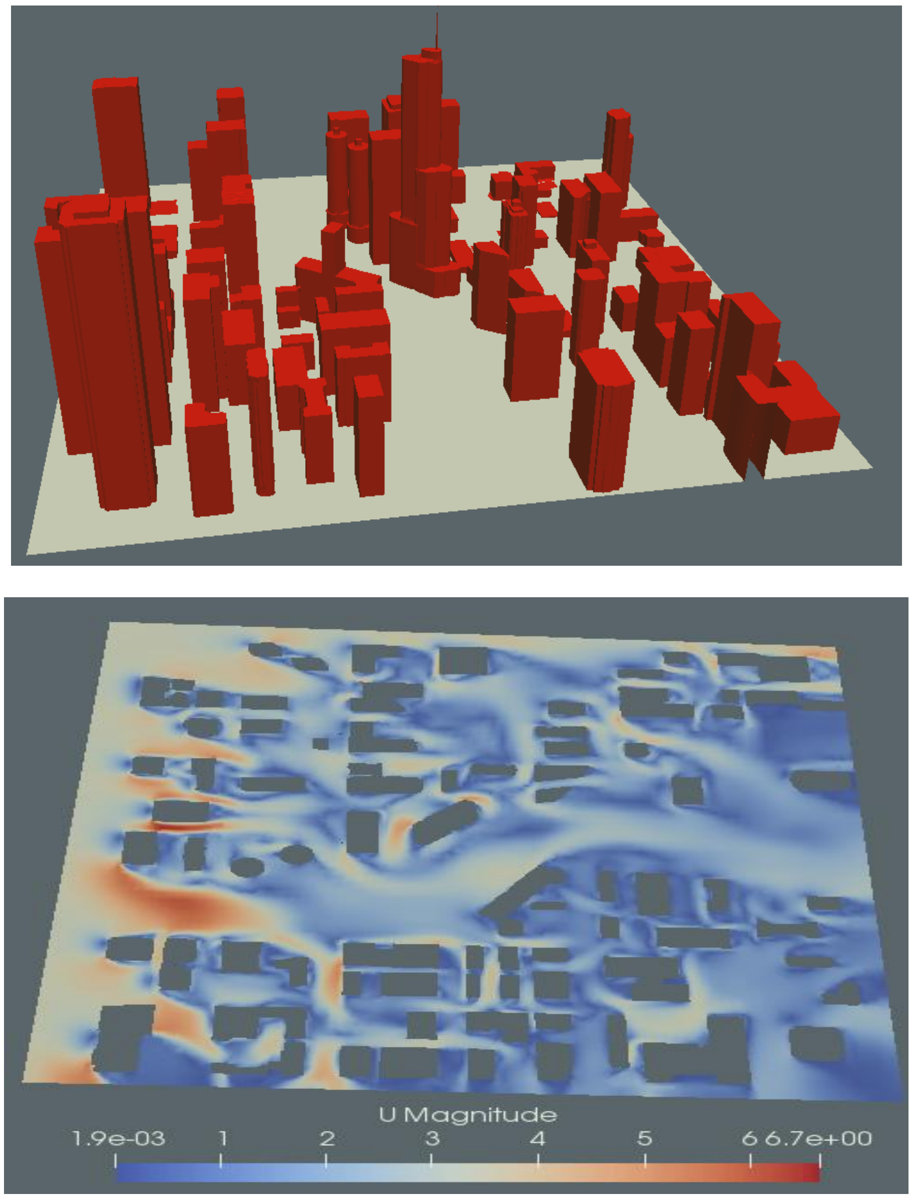
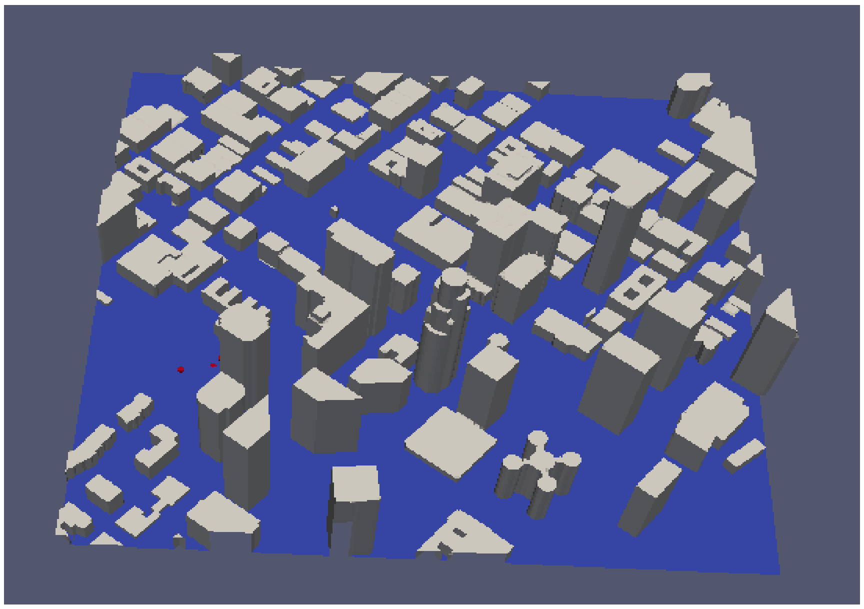


Figure 18. Results from CFD simulation of wind flows in Chicago's downtown area.

- **Los Angeles:** In Los Angeles, the combination of dense downtown zones, coastal marine airflows, and sloped terrain created a complex and layered wind environment. Figure 16 shows results from CFD simulations during an extreme event that revealed rapid shifts in wind speed and direction, particularly along streets, where the channeling effect of the buildings may intensify gusts during sea breeze events.



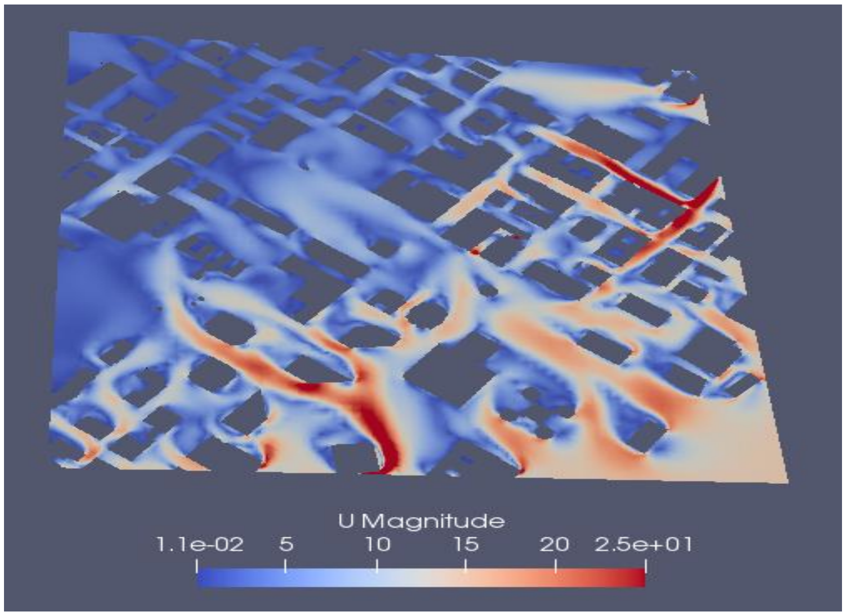


Figure 19. Results from CFD simulation of wind flows in Los Angeles's downtown area.

- **Downtown Dallas:** In Dallas, although the terrain was relatively flat and there were no coastal influences, Figure 17 shows that the moderate-density skyscrapers created localized accelerations and swirling eddies, particularly at building corners. This underscores the need for UAS operators to maintain safe distances from building edges, particularly during storm events when wind-induced instabilities are amplified.

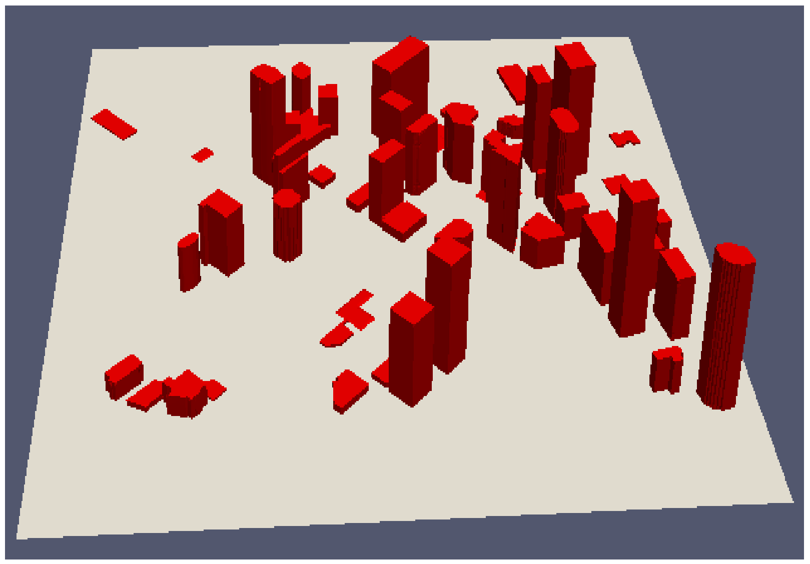
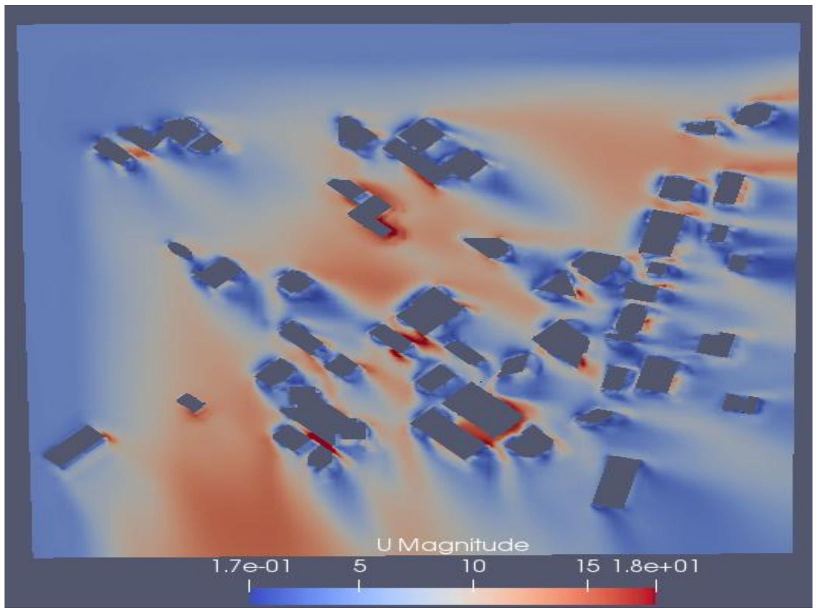



Figure 20. Results from CFD simulation of wind flows in Dallas's downtown area.

7.5 Conclusion

This study confirms that urban wind variability, shaped by building geometry, weather, and terrain, creates significant challenges for small UAS operations. High-resolution CFD simulations in four cities revealed complex patterns like shear layers, corner vortices, and accelerated flows that can destabilize drones near buildings and rooftops. To complement the CFD findings, the University of Kansas used a UAS flight dynamics model to show that standard PID-based controllers struggle with gusts and shear. Path-tracking accuracy also decreased in areas of intersecting wakes, highlighting the need for real-time wind-aware trajectory planning and adaptive control systems.

Together, these insights were used to develop operational guidance for safe lateral and vertical standoff distances, trajectory shaping, and pre-flight wind checks. The outcomes support FAA integration goals and provide a foundation for refining operational standards, real-world testing, and adaptive control development to ensure safe, reliable urban drone operations.

Future research should focus on conducting operational urban wind simulations, real-world flight tests with instrumented drones to validate simulation results and refine guidance. Developing adaptive control systems with real-time wind tuning and expanding CFD analysis to more cities will improve generalizability. Integrating these findings with urban wind sensor networks like the WindAware platform (Chrit and Majdi, 2024) can also enhance mission safety and flexibility.

7.6 References

Chrit, Mohamed, and Majdi Majdi. 2022. "Improving Wind Speed Forecasting for Urban Air Mobility Using Coupled Simulations." *Advances in Meteorology* 2022: Article ID 2629432, 14 pages. (https://doi.org/10.1155/2022/2629432).

Chrit, Mohamed. 2023. "Reconstructing Urban Wind Flows for Urban Air Mobility Using Reduced Order Data Assimilation." *Theoretical and Applied Mechanics Letters* 100451. (https://doi.org/10.1016/j.taml.2023.100451).

Britter, Rex E., and Steven R. Hanna. 2003. "Flow and Dispersion in Urban Areas." *Annual Review of Fluid Mechanics* 35 (1): 469–496. (https://doi.org/10.1146/annurev.fluid.35.101101.161147).

Chrit, Mohamed, and Majdi Majdi. 2024. "Operational Wind and Turbulence Nowcasting Capability for Advanced Air Mobility." *Neural Computing and Applications* 36: 10637–10654. https://doi.org/10.1007/s00521-024-09614-0).

Coceal, Ovidiu, T. G. Thomas, I. P. Castro, and S. E. Belcher. 2007. "Structure of Turbulent Flow over Regular Arrays of Cubical Roughness." *Journal of Fluid Mechanics* 589: 375–409. (https://doi.org/10.1017/S002211200700794X).

Fernando, Harindra J. S. 2010. "Fluid Dynamics of Urban Atmospheres in Complex Terrain." Annual Review of Fluid Mechanics, 42: 365–389. https://doi.org/10.1146/annurev-fluid-121108-145459).

Oke, Timothy R. 1987. *Boundary Layer Climates*. 2nd ed. Routledge. (https://doi.org/10.4324/9780203407219T).

Poulos, Gregory S., et al. 2002. "CASES-99: A Comprehensive Investigation of the Stable Nocturnal Boundary Layer." *Bulletin of the American Meteorological Society* 83 (4): 555–581. https://doi.org/10.1175/1520-0477%282002%29083)

Tominaga, Yasuhiro, and Takashi Stathopoulos. 2013. "CFD Simulation of Near-Field Pollutant Dispersion in the Urban Environment: A Review of Current Modeling Techniques." *Atmospheric Environment* 79: 716–730. (https://doi.org/10.1016/j.atmosenv.2013.07.028).

8 AUTOMATIC FLIGHT CONTROL

8.1 Executive Summary

Failures of automation include the failure of a flight controller to prevent UAS Loss Of Control (LOC) due to exigencies, such as an (onboard) flight control actuator failure or (external) severe weather events. The team declares that fault detection/mitigation by either hardware or software implementations are sub-optimal, because they require prior knowledge of all failure types. As such, the team studied the ability of a range of **resilient** controllers to survive, such as rudder failure and severe weather conditions. The team concludes that adaptive controllers are worthy of further study, and that Machine Learning (ML) algorithms provide superior performance over other implementations. It is not practically possible to test or simulate every combination of failures, uncertainties, environmental conditions, and interactions across the operational design domain as the input/fault space grows combinatorially. Consequently, adaptive controllers that can rapidly and reliably compensate for unmodeled dynamics and disturbances have shown promise for fast, bounded-transient adaptation under uncertainty. However, these methods are still in the developmental phase, and their integration into safety-critical systems will require updated and holistic verification and validation (V&V) methodologies that go beyond traditional software certification practices. Based on Subject Matter Expert (SME) input, this requires a "layered control" approach, when a single adaptive flight controller architecture performs both tracking and transient adaptation.

CFD models of both nominal and extreme wind fields provide excellent representations for the simulation of UAS performance in uncertain weather. In particular, simulations of the performance of a UAS control system responding to simulated wind fields can be an essential element of the process leading to certification of the control system. However, the most important weather-related issue for UAS (and UAM) safety during flight in an urban environment is knowledge of the wind/weather *at the landing zone*. Provisions for weather stations and navigational aids at the landing zone are the most likely mitigations for weather uncertainty to enhance safety. Based on SME input, in addition to safety enhancement, such services would provide support for air ambulance and other emergency vehicle decision-makers as they make go/no-go decisions for deployment.

The figure below illustrates the guidance within the overall architecture of a UAS. The blue components are the new elements of this guidance. Each box (e.g., Perception, Sensors) contains *examples* of what else is running in that module.

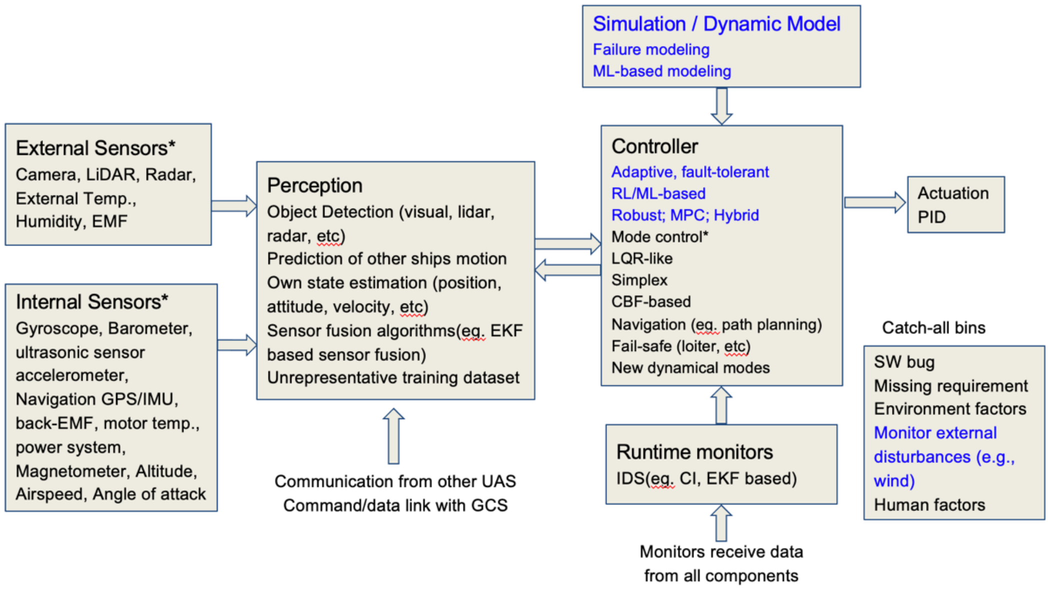

Simulation / Dynamic Model
Failure modeling
ML-based modeling
External Sensors*
Camera, LiDAR, Radar, External Temp., Humidity, EMF
Internal Sensors*
Gyroscope, Barometer, ultrasonic sensor accelerometer, Navigation GPS/IMU, back-EMF, motor temp., power system, Magnetometer, Altitude, Airspeed, Angle of attack
Perception
Object Detection (visual, lidar, radar, etc)
Prediction of other ships motion
Own state estimation (position, attitude, velocity, etc)
Sensor fusion algorithms(eg. EKF based sensor fusion)
Unrepresentative training dataset
Communication from other UAS
Command/data link with GCS
Controller
Adaptive, fault-tolerant
RL/ML-based
Robust; MPC; Hybrid
Mode control*
LQR-like
Simplex
CBF-based
Navigation (eg. path planning)
Fail-safe (loiter, etc)
New dynamical modes
Actuation
PID
Catch-all bins
SW bug
Missing requirement
Environment factors
Monitor external disturbances (e.g., wind)
Human factors
Runtime monitors
IDS(eg. CI, EKF based)
Monitors receive data from all components

Figure 21. Process diagram for flight control guidance. Guidance from this chapter is in blue.

8.2 Introduction and Background

There is a desire in industry to move towards increased automation of UAS operations. Autonomy is defined as the ability to independently choose how to act to achieve goals. An autonomous system is comprised of several subsystems, including sensory, decision-making, communication/navigation, propulsion, and actuation. Although the FAA has issued detailed guidance for the airworthiness certification of Unmanned Aircraft Systems and Optionally Piloted Aircraft (e.g., **FAA Order 8130.34D**), the majority of small UAS operating under 14 CFR Part 107 (i.e., aircraft weighing less than 55 pounds) are governed primarily by operational regulations—such as requirements for remote pilot certification, visual line-of-sight operations, and operating restrictions. These small UAS do not require formal airworthiness certification, unless the intended operations fall outside Part 107 provisions and necessitate a waiver or certification pathway. Because of this exemption, the small UAS market has been flooded by low-quality consumer-grade UAS (a.k.a. drones). Currently, many existing UAS are equipped with low-quality but inexpensive commercial-off-the-shelf systems without proper (or in many cases, minimum) engineering, manufacturing, and quality control practices. The complexity of UAS missions is another factor to be considered, which can lead to failures. In this work, potential automation failures as they relate to automated operations of UAS are studied and explored. Focus is placed on UAS failures related to flight test operations and automated flight control. Guidance on ways to mitigate such failures is provided, then validated using flight tests and simulations.

8.3 Recommended Guidance

In this section, guidance, engineering best practices, and mitigations to address failure in UAS autonomy are presented. In addition, validation using flight tests and simulations is presented in this section.

1) Artificial intelligence (AI)-based flight control offers an approach for passive fault-tolerant control, which can have improved performance over other control techniques. (See Section 8.3.1, 8.3.2 and 8.3.4). The reinforcement learning (RL)-based control method even has the potential to control multiple aircraft without being specifically designed for each aircraft, showing the possibility to handle different aircraft dynamics (See Section 8.3.2).
2) Architectural choices of flight controllers play a major role in ensuring an adaptive controller is adequate and performs as intended. In practice, robust–adaptive architectures, such as L_1 adaptive control, are synthesized to meet trajectory-tracking objectives and to mitigate adverse onboard conditions (e.g., turbulence, sensor/actuator degradation) by providing bounded transients and uniform performance in the presence of uncertainties and unmodeled dynamics . (See Section 8.3.1)
3) Machine learning offers approaches for improving UAS dynamic models.
    a. In Section **Error! Reference source not found.**, the cross-entropy method is presented. The method has the potential to be applied online during flight to capture changes in aircraft dynamics under adverse onboard conditions.
    b. In Section 8.3.4, the use of machine learning and a bank of previous flight test data to model the lateral-directional dynamics of UAS is presented. The approach has the potential to improve UAS dynamic models using existing flight test data collections without the need to perform complex flight test maneuvers for aircraft modeling.
4) Extreme wind shear, turbulence, and vorticity significantly degrade flight performance and aircraft maneuverability, particularly during low-altitude operations in urban environments. The unpredictable nature of these wind patterns can lead to autonomy failures, including loss of trajectory tracking, instability during tight turns, and compromised flight path control. To mitigate these risks, **navigational corridors must be adaptively managed** in response to changing environmental conditions. Under periods of high wind loading, certain corridors may require **temporary restriction or closure** to prevent loss of control. (See Section 8.3.5)
5) Designing guidance, navigation, and control algorithms for aircraft operating in urban environments, care should be exercised to avoid catastrophic failures in navigation and tracking due to weather conditions. (See Section 8.3.5).
6) While simulation-based failure assessment offers a cost-effective and scalable means for evaluating UAS performance, substantial discrepancies remain between simulated outcomes and real-world flight behavior, especially under loss-of-control or adverse onboard conditions. Developing a modular, empirical software toolkit for UAS would provide a low-cost alternative for modeling post-failure dynamics and enable rapid prototyping of recovery strategies. (See Section **Error! Reference source not found.**).

Figure 21 above indicates where the above guidance fits in an overall UAS autonomy diagram. The above guidance are identified using blue font in the diagram

8.3.1 Control Mitigation and Flight Test Validation

Failures in actuation can be mitigated using control methods that are either more robust to failures or sense and adapt to failures. Methods such as these are referred to as fault-tolerant control. Fault-tolerant control is divided into two subsections: Passive Fault Tolerant Control (PFTC) and active fault-tolerant control. PFTC uses a static controller structure that achieves fault tolerance using robust control techniques with a range of expected faults and system uncertainties. Active fault tolerant control instead uses a failure detection and diagnosis algorithm to sense the occurrence of failures and update the controller in real-time via changing parameters.

Two advanced control methods are being investigated as strong candidates for PFTC strategy: L_1 adaptive control and a flight controller trained using AI-based methods. L_1 adaptive control is a novel, robust adaptive control method with the ability to fast adapt without sacrificing robustness (Hovakimyan & Cao, 2010). Two of the main advantages of L_1 adaptive controller over other adaptive methods (e.g. Model Reference Adaptive Control (Butler, 1992)) are: 1) no need for a priori information, and 2) a fast adaptation is achieved by decoupling the adaptation loop from the control loop (Figure 22). An AI-based advanced optimization method allows us to train control policies to perform complex behavior, such as control reconfiguration in flight in the event of loss of effectiveness of a control surface, to achieve similar stability and tracking performance of the system. Long-Short-Term-Memory (LSTM) architecture is used to facilitate the adaptation ability of the controllers, with the help of the memory functionality of LSTM neural networks (Figure 23). This study (Chowdhury & Keshmiri, 2022) has shown that the method has relatively lower performance trade-offs for robustness than other methods, such as LQRs (Figure 24).

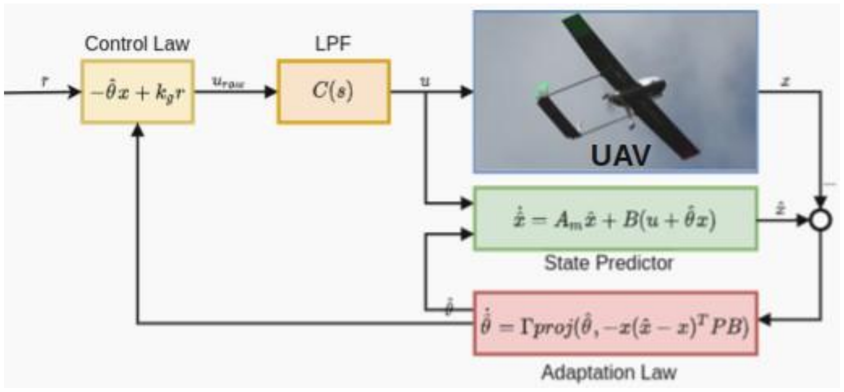

Control Law
LPF
r
$-\hat{\theta}x + k_g r$
u_{raw}
$C(s)$
u
UAV
x
$\dot{\hat{x}} = A_m\hat{x} + B(u + \hat{\theta}x)$
$\hat{x}$
State Predictor
$\hat{\theta}$
$\dot{\hat{\theta}} = \Gamma proj(\hat{\theta}, -x(\hat{x} - x)^T PB)$
Adaptation Law

Figure 22. L_1 adaptive control design.

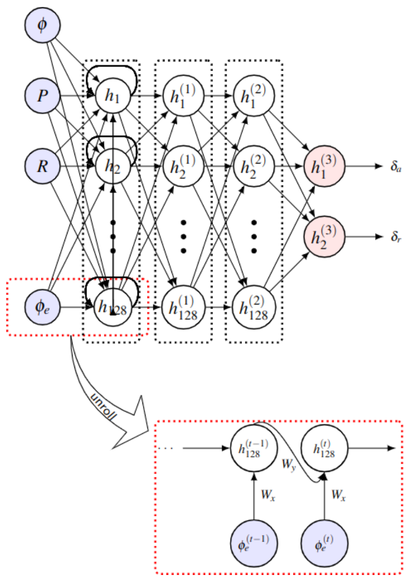


Figure 23. AI-based flight control: LSTM architecture.

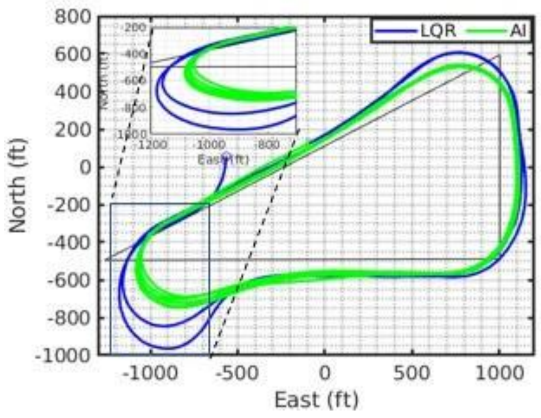
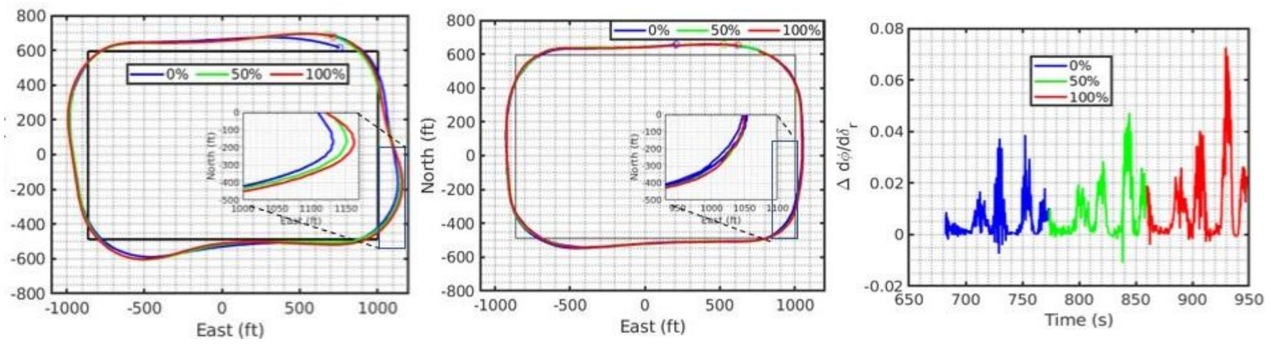


Figure 24. AI-based flight control: Performance comparison of AI- and LQR-based controllers during a tight tracking task.

Figure 25 shows flight test validation and verification of L_1 and AI-based flight control methods. 2D trajectory tracking performance of a racetrack-pattern of both control methods are shown in Figure 25a and Figure 25b, respectively. To check for the controller's adaptive ability to the loss of effectiveness of a control surface, rudder effectiveness was reduced programmatically to three degradation values: 0%, 50%, and 100%, during three consecutive racetrack loops for both controllers. Figure 25c shows uncertainty in $\Delta\frac{d\phi}{d\delta_r}$ estimated by L_1 control method is increased as the rudder degradation is increased, showcasing its adaptive ability. By comparing Figure 25b and Figure 25a, it can be seen that AI-based method outperformed the L_1 control methods by a significant margin in the 2D tracking performance for the same degradation values, showcasing the potentials of AI-based methods as PFTC.



(a) L_1 (b) AI (c) Uncertainty estimated by L_1

Figure 25. Rudder degradation flight test.

8.3.2 Reinforcement-learning flight controller for fixed-wing UAS

Since designing aircraft flight controllers is complex, expensive, and time consuming, the interchangeability of flight controllers between different aircraft platforms has been an active research area. The development of interchangeable, verifiable flight controllers for fixed-wing UAS was studied in (Chowdhury & Keshmiri, 2024b). A model-free deep Reinforcement Learning (RL) algorithm—called proximal policy optimization—trains the RL-based control policy. Instead of using high-fidelity dynamic models, the RL policy-based controller is trained in simulation using an engineering-level dynamic model. The robustness of the flight controller toward uncertainty in the dynamic model is improved using randomization of the dynamic model. An aircraft's six degrees of freedom model is used in training to eliminate the heavy reliance of modern controllers on dynamic models, which are prone to the accuracy of the trim information. The idea of an interchangeable flight controller is developed by incorporating memory functions into the policy using LSTM, a variant of recurrent neural network architecture. The developed flight controller is uniquely verified and validated in actual flight tests using fixed-wing autonomous aircraft. The interchangeable RL-based flight controller is flight-tested on an entirely different aircraft, which is the first of its kind. Its performance is superior to commercial-off-the-shelf flight controllers and LQR-based flight controllers explicitly designed for that platform. Flight test validation and verification data are used to assess flight controller performance and the comparison matrices.

The interchangeability of the developed RL-based control policy in software-in-the-loop simulation tests was demonstrated using three different UAS dynamic models. Specifications of the UAS used in this simulation study are in Table 13. UAS with mass values ranging from 4 to 27.2 kg and cruise speed values ranging from 13.7 to 30.8 m/s were used in this study. The largest UAS among the three, "Argus," has moments of inertia (I_{xx}, I_{yy}, and I_{zz}) about one order of magnitude higher than the others about its three-body axis. Figure 26 shows some of the results from the study where the same RL-based control policy (trained based on the SkyHunter dynamic model) was applied to controlling each UAS in a racetrack pattern tracking task. The controller shows good tracking performances in UAS cases with stable angular rates showcasing the extent of the controller's agnostic ability to significantly different dynamic models.

Table 13. Specifications of UAS Used in Simulation Tests.

Name of UAS	SkyHunter	Boreas	Argus
Wingspan [m]	1.8	2.4	3.6
Mass [kg]	4.0	5.4	27.2
Cruise speed [m/s]	13.7	16.8	30.8
I_{xx} [kg-m^2]	0.43	0.39	4.72
I_{yy} [kg-m^2]	0.73	0.35	6.36
I_{zz} [kg-m^2]	0.31	0.33	4.24

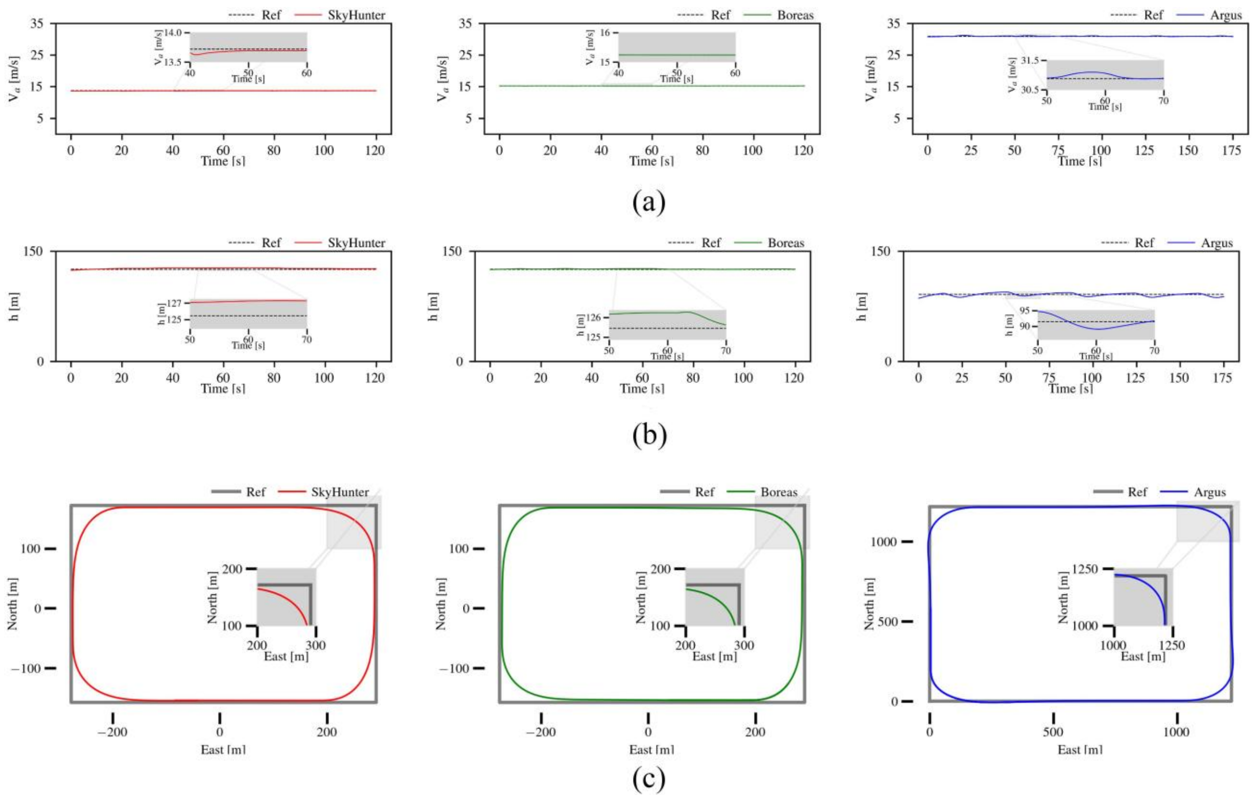


Figure 26. Interchangeability simulation tests. (a) Airspeed. (b) Altitude. (c) 2-D trajectory.

Pixhawk is one of the most popular and powerful autopilots of choice for small UAS platforms. Therefore, it was natural for the team to compare it with the developed RL-based autopilot system. Figure 27 shows the comparative airspeed, altitude, and 2-D trajectory tracking performance between these two autopilots. From the performance metrics given in Table 14, the team can see that the team's RL-based autopilot outperforms the Pixhawk autopilot in every metric and by a significant margin in airspeed and altitude tracking. The RL-based autopilot has about one order of magnitude better airspeed tracking and five times better altitude tracking than the Pixhawk-based autopilot. Pixhawk uses PID-based single-input and single-output control for airspeed, which suffers from hang-off errors, as seen in Figure 27(a). Figure 27(b) shows that a maximum drop of altitude of about 10 m for the Pixhawk autopilot and about 3 m for the team's autopilot. Figure 27(c) shows that Pixhawk autopilot has a maximum cross-track error of about 40 m, while the SkyHunter is coming out of the southwest and southeast corners of the racetrack pattern (counterclockwise flight), experiencing tailwind and crosswind, respectively. In contrast, the team's autopilot has a maximum cross-track error of about 30 m at those corners. Figure 27(c) also shows that the team's autopilot generates significantly more consistent 2-D trajectories, proving its better disturbance rejection capabilities.

Table 14. Performance Metrics of the Comparison Flight Test with Pixhawk autopilot.

UAS	Nominal Speed [m/s]	Wind Dir. & speed [m/s]	Criteria	Pixhawk	RL
SkyHunter	14.0	W 4.5 to 6.3	Airspeed tracking RMSE [m/s]	2.0	0.2
			Altitude tracking RMSE [m]	4.7	0.9
			Roll tracking RMSE [deg]	4.7	4.4

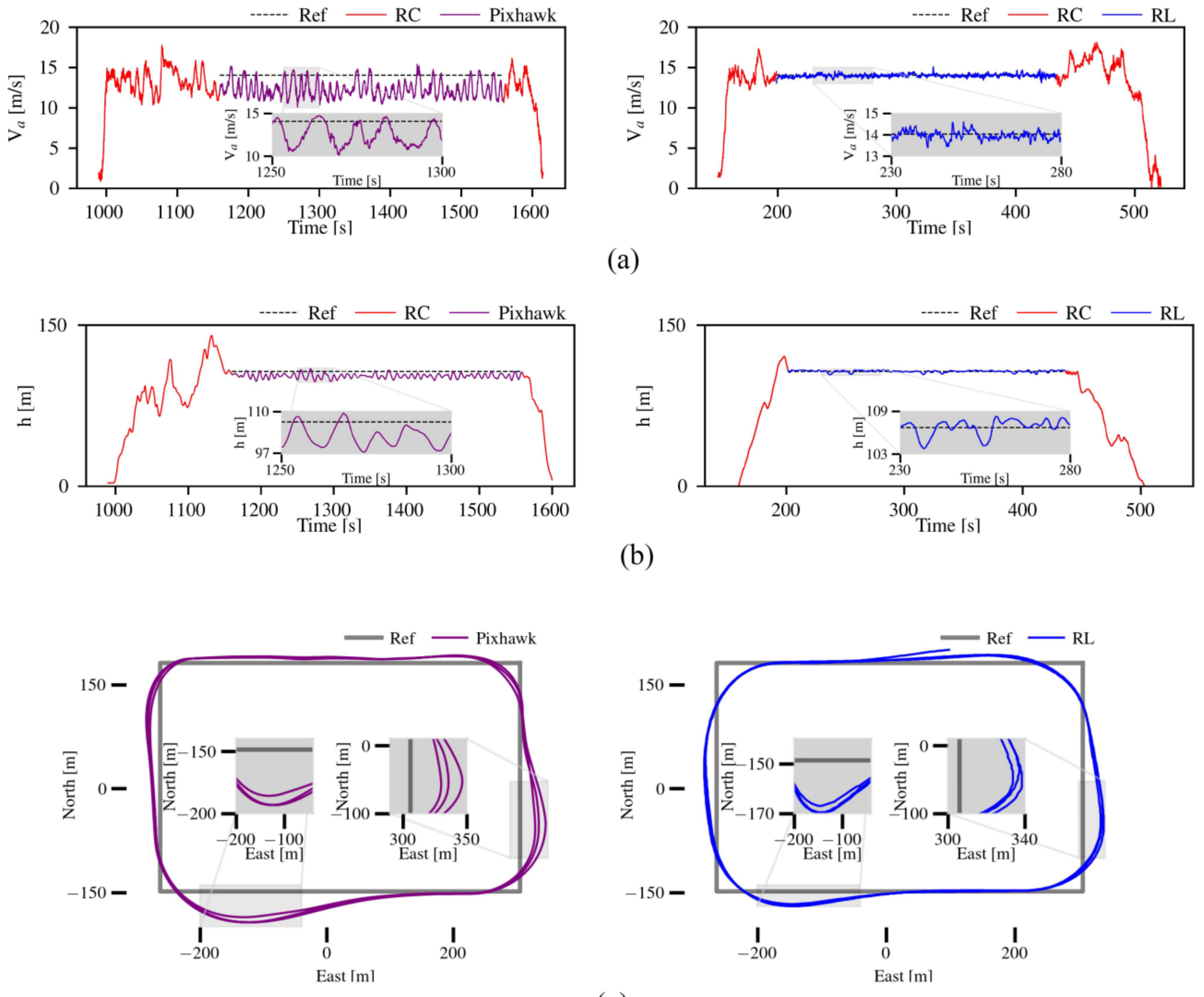


Figure 27. Comparison flight test between Pixhawk and RL-based autopilots. UAS: SkyHunter, Wind: W 4.5 to 6.3 ms^{-1}. (a) Airspeed. (b) Altitude. (c) 2-D trajectory.

An LQR-based autopilot for each target UAS was developed. These base LQR autopilots had been flight tested and tuned more than 200 times (Kim et al., 2020) to achieve the highest base performance. Figure 28 compares the flight test results with the LQR-based autopilot. The flight test was conducted using Boreas UAS. The guidance parameters, which were tuned specifically for the base autopilot, were kept the same for both autopilot cases to remove the impact of the guidance on the controller's performance comparison result. Table 15 shows that the RL-based autopilot performs similarly to LQR in altitude [Figure 28(b)], roll, and 2-D [Figure 28(c)] tracking and outperforms the LQR in airspeed tracking [Figure 28(a)] by one order of magnitude.

Table 15. Performance Metrics of the Comparison Flight Test With LQR-Based Autopilot.

UAS	Nominal Speed [m/s]	Wind Dir. & speed [m/s]	Criteria	LQR	RL
Boreas	16.0	SE 2.2 to 3.6	Airspeed tracking RMSE [m/s]	1.7	0.4
			Altitude tracking RMSE [m]	0.7	0.7
			Roll tracking RMSE [deg]	6.3	6.8

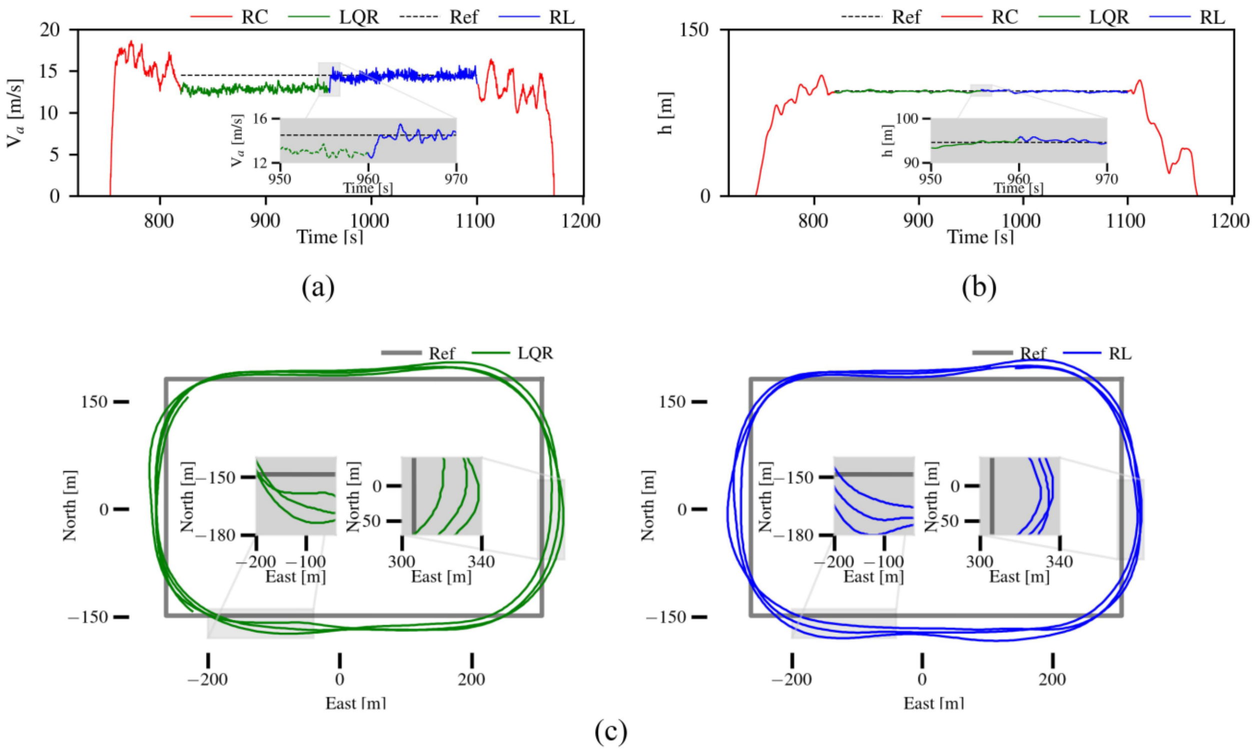


Figure 28. Comparison flight test between LQR and RL-based autopilots. UAS: Boreas, Wind: SE 2.2 to 3.6 ms^{-1}. (a) Airspeed. (b) Altitude. (c) 2-D trajectory.

8.3.3 Cross-Entropy (CE) method for failure modeling

Cross-Entropy (CE), a derivative-free optimization method with real-time performance, is used to continuously fit parameters of a locally Linear Time Invariant (LTI) dynamic model to the nonlinear changes in the aircraft's dynamics due to failure events (see Figure 30) (Chowdhury & Keshmiri, 2024a). The method uses initial model parameters from a physics-based model and predetermined standard deviations to define a multivariate Gaussian distribution (Figure 29). At the beginning of each cycle of operation, the method uses a moving window of flight data, samples a number of parameter sets, and performs LTI simulations on the flight data using parallel computing processes (Figure 31). The method then evaluates each simulated trajectory using an objective function, ranks the sampled parameter sets, and picks the top p% of the best performing parameter sets to update the distribution parameters for the next iteration. The method repeatedly updates its distribution (Figure 31) until either error tolerances or a convergence criterion is met. The method then shifts the flight data window by one timestep and repeats the cycle. The flight data window length can be varied between 0.5-3 seconds, where a lower window size results in faster adaptation and higher generalization error.

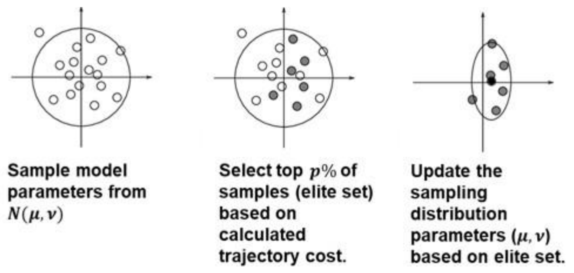

Sample model parameters from $N(\mu, \nu)$
Select top $p\%$ of samples (elite set) based on calculated trajectory cost.
Update the sampling distribution parameters (μ, ν) based on elite set.

Figure 29. CE method for System Identification: Main steps of the algorithm.

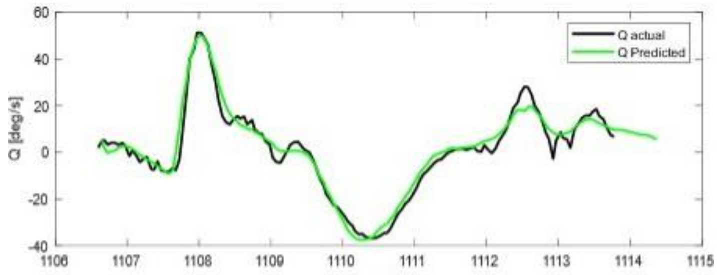


Figure 30. CE method for System Identification: Prediction of pitch-rates (Q) during a stall condition.

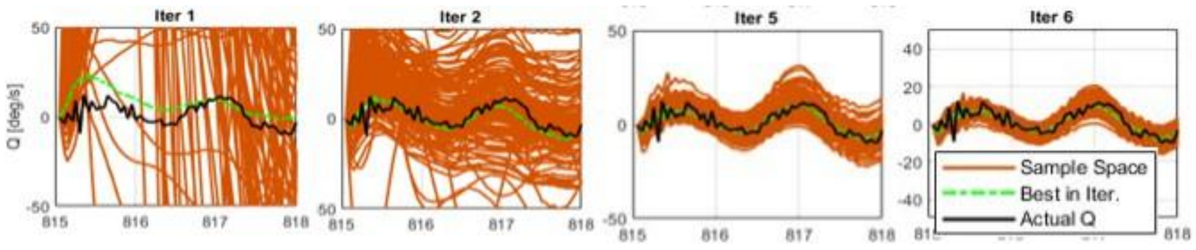


Figure 31. CE method for System Identification: Progress iterations.

8.3.4 *Data-driven Machine Learning Methods for Improving Aircraft Dynamics Modeling*

Low-fidelity engineering-level dynamic models are commonly employed while designing uncrewed aircraft flight controllers due to their rapid development and cost-effectiveness. However, during adverse conditions or complex path-following missions, the uncertainties in low-fidelity models often result in suboptimal controller performance. Aircraft system identification techniques offer alternative methods for finding higher fidelity dynamic models but can be restrictive in flight test requirements and procedures. This challenge is exacerbated when there is no pilot on board. The team introduced, in (Benyamen et al., 2024), data-driven ML to enhance the fidelity of aircraft dynamic models, overcoming the limitations of conventional system identification. A large dataset from twelve previous flights is utilized within an ML framework to create an LSTM model for the aircraft's lateral-directional dynamics. The LSTM model showed some improved modeling over LTI and 6-DOF dynamic models, as seen in Figure 32. A deep RL-based flight controller is developed using a randomized dynamic domain created using the LSTM and physics-based models to quantify the impact of LSTM dynamic model improvements on controller performance. The RL controller performance is compared to other modern controller techniques in actual flight tests in the presence of exogenous disturbances and noise, assessing its tracking capabilities and its ability to reject disturbances. The RL controller with a randomized dynamic domain outperforms a linear quadratic regulator controller and an L_1 adaptive controller. Notably, it demonstrated up to 72% improvements in lateral tracking when the aircraft had to follow challenging paths and during intentional adverse onboard conditions. The improved tracking performance for the RL-based controller flight compared to the other controllers in a racetrack pattern is seen in Figure 33 and Table 16. Improved RL-controller flight tracking in a more challenging (sharp turn) flight maneuver is shown in Figure 34 and Table 17.

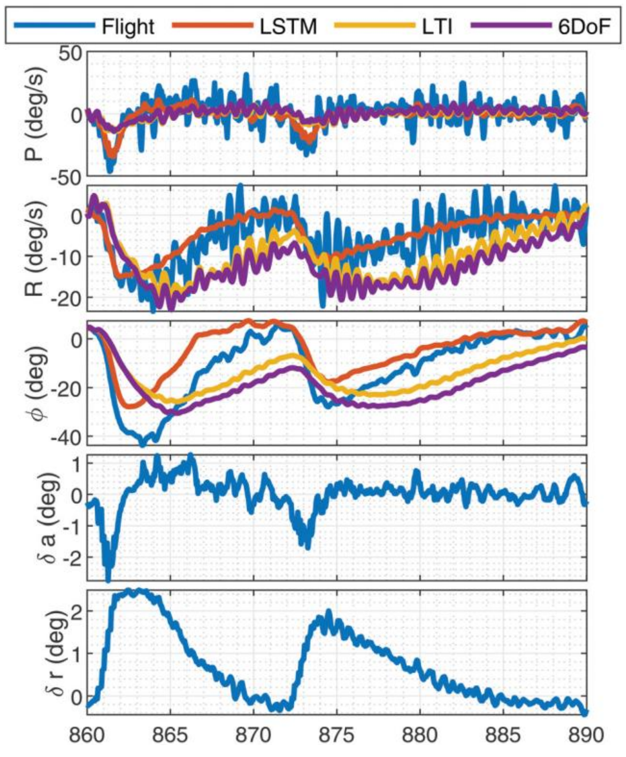
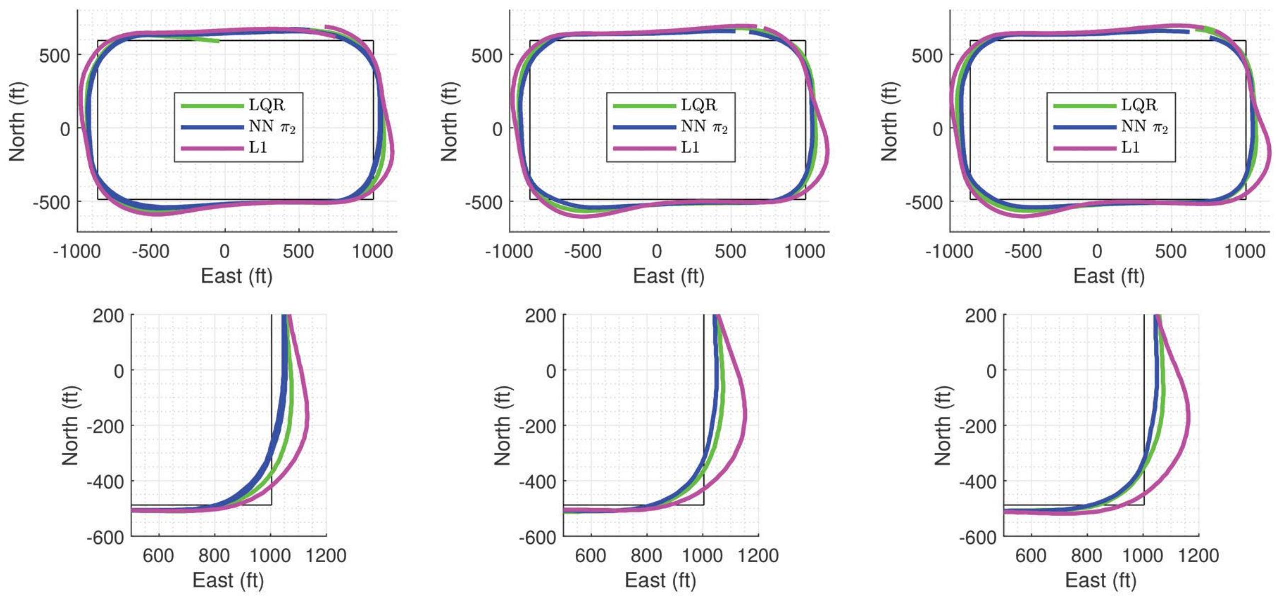


Figure 32. LSTM model showing improved modeling of roll rate (P) and yaw rate (R) compared to the LTI and the 6 degree-of-freedom models.



(a) δ_r effectiveness: 100% (b) δ_r effectiveness: 50% (c) δ_r effectiveness: 0%

Figure 33. Improved flight performance for RL-based controller flight (NN π_2), compared to LQR and L_1 flight controllers. Flight test results presented for different rudder effectiveness.

Table 16. Maximum tracking error for the RL-based, LQR, and L_1 controllers under different rudder effectiveness values.

Controller	Rudder Effectiveness	Maximum Tracking Error (ft)
RL-based (NN π_2)	100 %	50 East
LQR	100 %	71 East
L_1	100 %	128 East
RL-based (NN π_2)	50 %	45 East
LQR	50 %	69 East
L_1	50 %	147 East
RL-based (NN π_2)	0 %	45 East
LQR	0 %	69 East
L_1	0 %	158 East

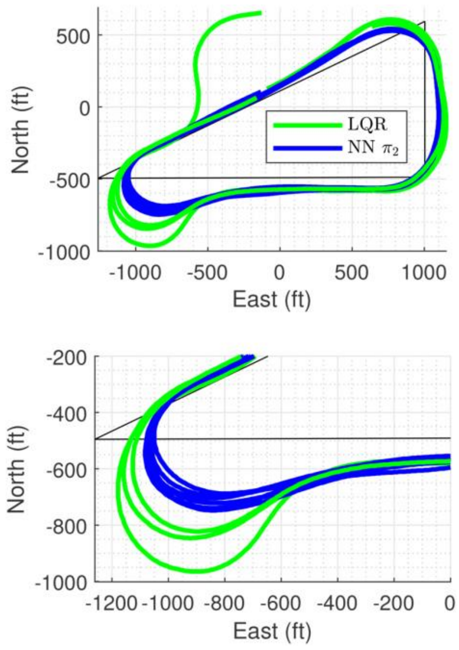


Figure 34. Improved flight performance for RL-based controller flight (NN π_2), compared to LQR controller in a more challenging (sharp turn) flight pattern.

Table 17. Maximum tracking error for the RL-based and LQR controllers in a more challenging (sharp turn) flight pattern.

Controller	Maximum Tracking Error (ft)
RL-based (NN π_2)	254 South
LQR	740 South

8.3.5 Robust and adaptive control techniques can be developed to tackle the impact of adverse onboard conditions, such as weather

H_∞ control, a branch of modern control theory, has been utilized to reduce the impact of external disturbances, such as wind. The lightweight design, low operational ceiling, and high wind-to-cruise velocity ratio of UAS make them particularly vulnerable to such disturbances. To address these challenges, in (Kucuksayacigil et al., 2025) the team developed an H_∞ flight controller that emphasizes the difficulties small UAS face operating in high wind and wind shear conditions. H_∞ controllers are generally categorized into structured and unstructured types. Structured controllers relate to fixed-order designs. The team demonstrated that structured H_∞ tracking controllers can be designed by augmenting the original plant with filters. Once the optimal controller is derived for the augmented plant, it can be combined with the filter to obtain the optimal controller for the original plant. The resultant H_∞ controller inherits the order of the selected filter.

UAS often face significant variability in wind speed and direction due to airflow deflection caused by buildings in urban environments. This variability can impact flight stability and control. Key concerns are wind speed, shear, and vorticity variability. Sudden changes in winds can lead to unpredictable UAS behavior. Wind direction changes and erratic wind patterns can make it difficult for the UAS to maintain its intended flight path.

Simulation results showing deviations along the flight path without any wind incorporated into the system appear in the first two rows of Table 18. Although these deviations are within an acceptable range, when wind is incorporated, deviations along the streets are larger than the acceptable ranges (see the 3rd and 4th row of Table 18). These deviations can lead to catastrophic crashes, with the largest deviation measuring 59.44 ft. along 2nd Avenue. Consequently, a Proportional Integral guidance control acting on the waypoints has been integrated into the guidance to improve the tracking behavior of the aircraft. The guidance control has reduced the deviations along all streets, which can be observed in the 5th and 6th row of Table 18. The deviation along 2nd Avenue has been decreased by 53.43% when the guidance control is applied and fixed-order H_∞ controller is used. (The wind data near TSS Heliport, New York, obtained by CFD, have been used for the simulations. The wind data in Figure 35 represent the instantaneous wind along the flight path given in Figure 36 at an elevation of 289 feet.) The aircraft successfully tracked the desired path, with deviations remaining within acceptable thresholds despite wind shear. However, the deviations highlighted by the blue rectangles in Figure 36 represent potential crash zones, which could arise from unmodeled dynamics or oscillations in the longitudinal or lateral motion of the UAS.

Table 18. Deviations from the reference trajectory in feet. (GC: Guidance control).

			Deviations along the streets			
Wind	GC	Controller Used	37th St	34th St	FDR Dr	2nd Ave
Off	Off	H_∞	3.74 ft	3.36 ft	3.35 ft	2.43 ft
Off	Off	LQT	4.68 ft	4.60 ft	2.39 ft	2.85 ft
On	Off	H_∞	42.71 ft	46.26 ft	40.87 ft	53.79 ft
On	Off	LQT	46.45 ft	51.35 ft	46.67 ft	59.44 ft
On	On	H_∞	14.57 ft	23.64 ft	17.48 ft	25.05 ft
On	On	LQT	18.13 ft	29.12 ft	16.42 ft	31.01 ft

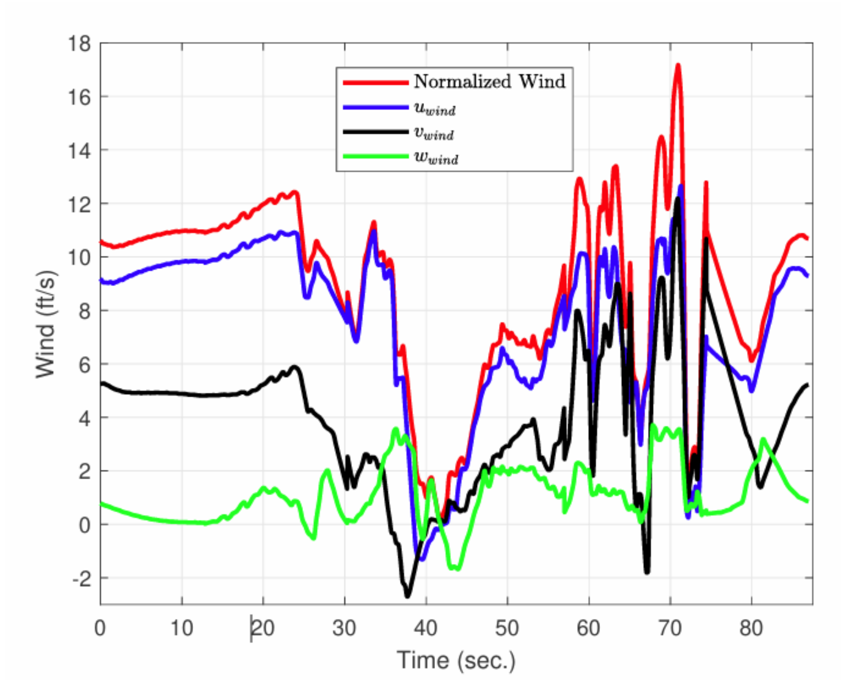


Figure 35. Wind used in controller simulations. Normalized wind, and translational impact of wind in the body coordinate system: u, v, and w.

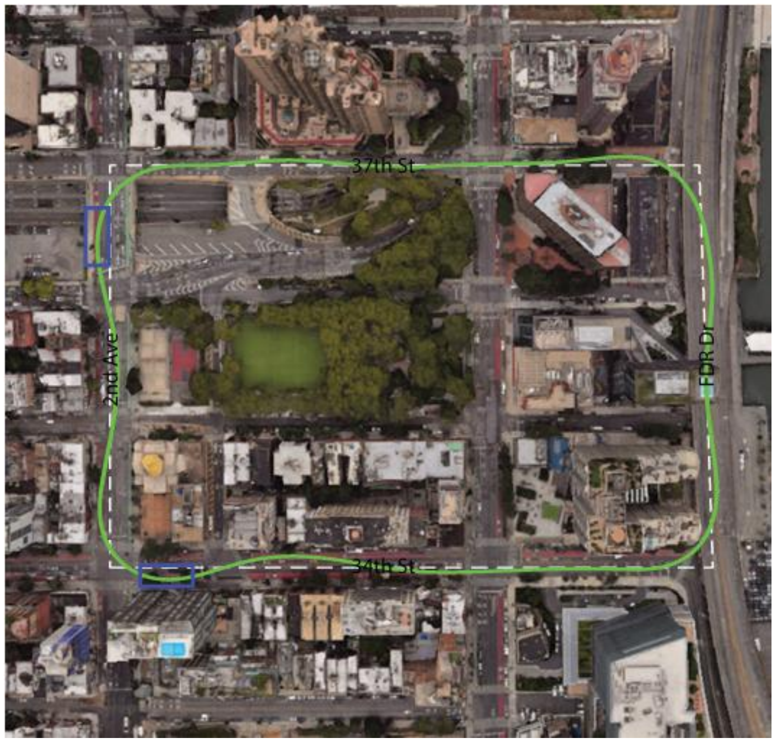


Figure 36. Dashed white path is the reference flight path, and the green (–) path is the simulated 2D flight path with NY wind given in Figure 35 when the fixed-order H_∞ controller and guidance control is applied. The possible crash areas are marked with blue rectangles.

Wind amount was increased by 25% and 50 % from its original values, as shown in Figure 35, to evaluate the effects of higher wind shear. The results of different wind scales are presented in Table 19. Along 34th Street, deviations from the flight path increased by 113.92% and 197.84% for wind increases of 25% and 50%, respectively. Similarly, the deviations along 2nd Avenue increased by 71.18% and 152.22% for the same wind increments. Such deviations from the predefined path can cause a catastrophic failure, as the deviations are more than the maximum width of the street (see Figure 37 and Figure 38). Moreover, these results demonstrate that the increase in deviation is not proportional to the rise in wind amount. Such nonlinearities should be considered when planning flights on days with adverse wind conditions.

The UAS path-tracking performance significantly degrades when the wind magnitude increases by factors of 1.25 and 1.5. When the wind magnitude is raised by 1.25 or greater factors, the aileron and rudder rates must exceed their acceptable limits to track the path with permissible deviations. High wind conditions adversely impact the guidance, resulting in significant deviations from the desired path, which can lead to a catastrophic crash. These substantial errors (Table 19) can be interpreted as failures of the UAS autonomy algorithm. When designing guidance, navigation, and control algorithms for aircraft operating in urban environments, care should be exercised to avoid catastrophic failures in tracking due to weather conditions. Potential measures to avoid failures are to monitor the environmental conditions (e.g., wind conditions) and to properly plan the flight trajectories given the environmental conditions. Depending on the UAS capabilities, flight may not be permitted if the environmental conditions exceed certain thresholds.

Table 19. Deviations from the reference trajectory with different scales of wind data in Figure 35. (GC: Guidance Control).

			Deviations along the streets			
Wind Scale	GC	Controller Used	37th St	34th St	FDR Dr	2nd Ave
x1	On	H_∞	14.57 ft	23.64 ft	17.48 ft	25.05 ft
x1.25	On	H_∞	15.89 ft	50.57 ft	19.59 ft	42.88 ft
x1.5	On	H_∞	27.63 ft	70.41 ft	27.88 ft	63.18 ft

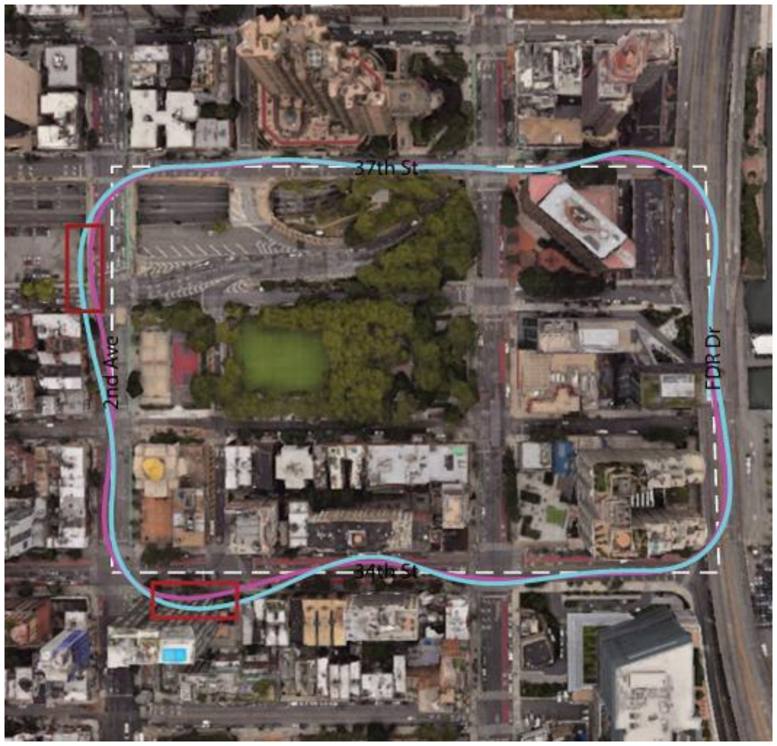


Figure 37. 2D flight path with NY wind in scale of 1.25 (in magenta –) and 1.5 (in blue –) of the wind data in Figure 35 with the fixed-order H_∞ controller and guidance control. The failure areas are marked with red rectangles.

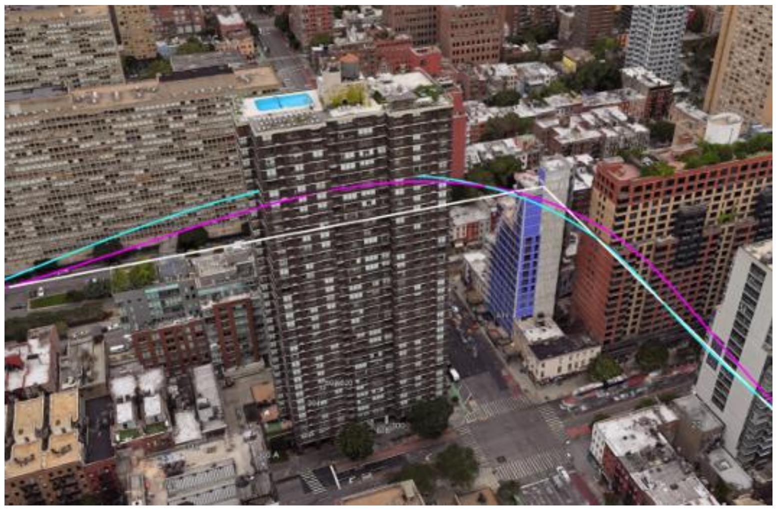


Figure 38. 3D view of the failure of guidance with NY wind in scale of 1.25 (in magenta –) and 1.5 (in blue –) of the wind data in Figure 35 with the fixed-order H_∞ controller and guidance control. This figure is a closer view of the crash area: red rectangle area on the lower left part of Figure 37.

8.3.6 Dynamic Modeling Representation Failures

Finding an accurate estimation of uncertainties in the physics-based dynamic model of aircraft in adverse onboard conditions is the precursor to developing an effective failure mitigation plan and control strategy. The nonlinear and unsteady dynamics of aircraft in adverse onboard conditions involve inertial coupling between the longitudinal and lateral-directional motions. Additionally, in the absence of Mean Time Between Failures (MTBF) and onboard diagnostic systems, there is no a priori information about what has failed and to what extent. Advanced onboard diagnostic systems are heavy and expensive, prohibiting their applications in small UASs. A more suitable approach is the data-driven dynamic modeling approach, based on the generated input-output data, which has minimum reliance on prior knowledge of the system dynamics, can learn from the observed data, and scales naturally with the system's complexity.

Failure testing could be completed with lower cost and difficulty through a simulation-based approach. However, the different impact of failures in simulation versus reality is a sparsely researched topic. Due to this, the comparison between simulation and real flight was performed for the afore-mentioned loss of effectiveness failure. The simulation was performed in a hardware-in-the-loop simulation environment using an engineering-level physics-based model of the aircraft and the same guidance, navigation, and control algorithms used in the flight test. This limits the difference between simulation and flight to differences in dynamic model and differences in operating condition, as no wind or sensor noise is present in the simulation. The differences are highlighted below in Figure 39 and Figure 40. In the flight test, the reduction of aileron effectiveness increased tracking error for roll angle, leading to decreased lateral tracking at 70% degradation and complete loss of tracking at 15%. However, simulation results do not follow this same trend. As aileron effectiveness is decreased in simulation, instead of a large increase of roll angle tracking error, the phase delay in the tracking of commanded roll angle reference is increased. At 70%, this delay is so small that the change in performance is negligible. However, at 15%, the delay increases to a point such that the aircraft begins to oscillate laterally, constantly overshooting the commanded path instead of tracking it. These differences emphasize the importance of improved modeling of these failures and increasing understanding of failure impact on aircraft dynamics. Developing a modular, empirical software toolkit (similar to the USAF's DATCOM) tailored for modeling UAS even in post-failure dynamics would provide a low-cost alternative for modeling UAS post-failure and enable rapid prototyping of recovery strategies.

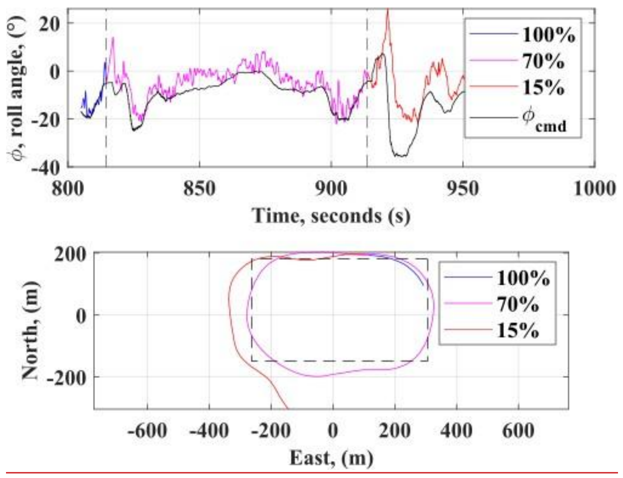


Figure 39. Lateral state and path tracking of degradation during flight test.

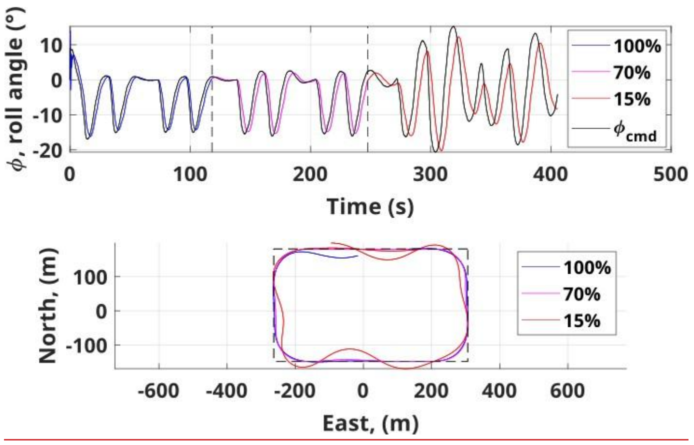


Figure 40. Lateral state and path tracking of degradation during hardware-in-the-loop test.

8.4 Conclusions

The following conclusions may be drawn from the foregoing results:

1. Adaptive controllers offer a cost-effective alternative to heavy and expensive hardware-based failure detection systems. Fault-tolerant control strategies mitigate actuator failures either passively—using robust designs like L_1 adaptive control—or actively through real-

time fault detection and controller reconfiguration. L_1 adaptive control enables rapid adaptation without requiring prior fault knowledge by decoupling the adaptation and control loops. AI-based controllers, such as those using LSTM networks, autonomously reallocate control in-flight and manage complex nonlinear failure modes. Compared to traditional methods like LQR, both approaches demonstrate superior robustness and adaptability in degraded flight scenarios. Validation flight tests under rudder effectiveness degradations (0%, 50%, 100%) confirmed their adaptive capabilities, with L_1 control showing increased estimated uncertainty and AI-based methods achieving higher trajectory tracking accuracy—highlighting their potential as passive fault-tolerant control solutions.

2. Developing interchangeable flight controllers for fixed-wing UAS is an alternative for popular PID controllers with limited robustness in the LOC. A model-free RL-based controller, trained using proximal policy optimization with randomized dynamics, demonstrated robust performance across significantly different UAS platforms and in the presence of adverse onboard conditions. Incorporating LSTM enabled memory-based adaptation, allowing transferability without retraining. Flight tests showed that the RL controller outperformed both commercial Pixhawk and platform-specific LQR controllers, especially in airspeed and altitude tracking. Software-in-the-loop tests confirmed generalization across UAS with varying mass, inertia, and flight characteristics. Overall, the RL-based autopilot achieved superior resiliency, tracking consistency and disturbance rejection, validating its potential for scalable, adaptive UAS control.
3. Machine learning based modeling techniques like real-time CE method can make flight controller dynamically aware to dramatic changes in the dynamics of UAS. They can extract nonlinear and unsteady behavior of aircraft in the LOC using flight data. The approach balances fast adaptation with generalization. Low-fidelity models often underperform under adverse conditions, leading to suboptimal control. To overcome this, an LSTM-based dynamic model trained on twelve prior flights was developed to capture lateral-directional dynamics more accurately than LTI or 6-DOF models. This model was integrated into a randomized domain for training a deep RL controller. The RL controller was validated in real flights, outperforming LQR and L_1 adaptive controllers, especially under disturbances. It achieved up to 72% improvement in lateral tracking in complex maneuvers. Performance gains were demonstrated through racetrack and sharp-turn flight tests.
4. The team's flight tests and simulation-based research (in collaboration with the North Dakota University Team) indicate that extreme wind shear, turbulence, and vorticity significantly degrade flight performance and aircraft maneuverability, particularly during low-altitude operations in urban environments. The unpredictable nature of these wind patterns can lead to autonomy failures, including loss of trajectory tracking, instability during tight turns, and compromised flight path control. The study reveals that conventional control strategies, such as Linear Quadratic Gaussian controllers, often fail to maintain precise trajectory tracking when subjected to abrupt changes in wind dynamics. However, integrating robust control techniques, H-infinity ($H\infty$) control, with adaptive

guidance control algorithms demonstrates a significant improvement in maintaining stability and trajectory accuracy.

While simulation-based failure assessment offers a cost-effective and scalable means for evaluating UAS performance, substantial discrepancies remain between simulated outcomes and real-world flight behavior, especially under loss-of-control or adverse onboard conditions. These discrepancies undermine the reliability of simulation results for safety-critical tasks such as certification and operational risk assessment. The absence of standardized, open-source, and inexpensive empirical modeling tools for UAS—comparable to the USAF's DATCOM for conventional aircraft—limits the accessibility of validated aerodynamic and failure-mode models tailored to small UAS platforms. Developing a modular, empirical software toolkit for UAS would provide a low-cost alternative for modeling post-failure dynamics and enable rapid prototyping of recovery strategies.

8.5 References

Benyamen, Hany, Md Chowdhury, and Shahin Keshmiri. 2024. "Data-Driven Aircraft Modeling for Robust Reinforcement Learning Control Synthesis With Flight Test Validation." *Journal of Dynamic Systems, Measurement, and Control*, 146 (6): 061105. https://doi.org/10.1115/1.4065804.

Butler, Hoke. 1992. *Model Reference Adaptive Control: From Theory to Practice*. Prentice-Hall, Inc.

Chowdhury, Md, and Shahin Keshmiri. 2022. "Design and Flight Test Validation of an AI-Based Longitudinal Flight Controller for Fixed-Wing UAS." In *2022 IEEE Aerospace Conference (AERO)*, 1–12.

Chowdhury, Md, and Shahin Keshmiri. 2024a. "A Unified Inner-Outer Loop Reinforcement Learning Flight Controller for Fixed-Wing Aircraft." In *2024 International Conference on Unmanned Aircraft Systems (ICUAS),* 556–563. https://doi.org/10.1109/ICUAS60882.2024.10556883.

Chowdhury, Md, and Shahin Keshmiri. 2024b. "Interchangeable Reinforcement-Learning Flight Controller for Fixed-Wing UAS." *IEEE Transactions on Aerospace and Electronic Systems* 60 (2): 2305–2318. https://doi.org/10.1109/TAES.2024.3351608.

Eckenhoff, Kevin, Patrick Geneva, and Guoquan Huang. 2019. "Sensor-Failure-Resilient Multi-IMU Visual-Inertial Navigation." In *Proceedings*, 3542–3548.

Fei, Fengjun, Zhiqiang Tu, Dongyan Xu, and Xiangyu Deng. 2020. "Learn-to-Recover: Retrofitting UAVs with Reinforcement Learning-Assisted Flight Control under Cyber-Physical Attacks." In *Proceedings*, 7358–7364.

Freeman, Paul, and Gary Balas. 2014. "Actuation Failure Modes and Effects Analysis for a Small UAV." In *Proceedings*, 1292–1297.

Hovakimyan, Naira, and Chengyu Cao. 2010. *L_1 Adaptive Control Theory: Guaranteed Robustness with Fast Adaptation*. SIAM.

Kim, Ae Ra, Shahin Keshmiri, Allison Blevins, Deepak Shukla, and Wenbo Huang. 2020. "Control of Multi-Agent Collaborative Fixed-Wing UAS in Unstructured Environment." *Journal of Intelligent & Robotic Systems* 97 (1): 205–225. https://doi.org/10.1007/s10846-019-01057-3.

Kucuksayacigil, Gokhan, Shahin Keshmiri, and Mohamed Chrit. 2025. "A Robust Flight Controller Design: Investigating Guidance Failures Near TSS Heliport in Challenging Wind Conditions." In *2025 International Conference on Unmanned Aircraft Systems (ICUAS)*, 1167–1174. https://doi.org/10.1109/ICUAS65942.2025.11007899.

Nan, Fei, Shuangfu Sun, Philipp Foehn, and Davide Scaramuzza. 2021. "Nonlinear MPC for Quadrotor Fault-Tolerant Control." arXiv:2109.12886. https://arxiv.org/abs/2109.12886.

Prochazka, Karel, Tobias Ritz, and Hugo Eduardo. 2019. "Over-Actuation Analysis and Fault-Tolerant Control of a Hybrid Unmanned Aerial Vehicle." In *5th CEAS Conference on Guidance, Navigation, and Control*.

9 PERCEPTION

During the first two years of the project, the team's efforts focused on evaluating and forecasting the perception capabilities of UAS, specifically their effectiveness in detecting and avoiding other drones operating in small airspace environments with high UAS density, such as urban centers. Early in this phase, the team identified a key limitation in publicly available drone image datasets—namely, a lack of diversity, particularly in images depicting small drones captured at long distances. To address this gap, the team conducted a targeted data collection campaign, acquiring a substantial volume of long-range drone imagery. These efforts culminated in the release of the Long Range Drone Detection (LRDD) dataset, versions 1 and 2 [1, 2].

To further enhance dataset diversity, the team explored the generation of synthetic drone imagery using graphical simulation platforms such as Unreal Engine (UE) [6]. The team then evaluated the detection performance of state-of-the-art object detection models, including the You Only Look Once (YOLO) series [10], trained on a combination of the LRDD datasets and synthetically generated images. Experimental results demonstrated that training with the team's curated datasets significantly improved detection accuracy, especially for small drones at extended distances.

9.1 Executive Summary

In the third year, the team expanded the team's dataset by collecting an additional 30,000 real-world images of drones captured at long range; labeling of this new data is currently in progress. In parallel, the team generated 30,000 synthetic images, which the team plans to incorporate into the next release of the LRDD dataset. To improve the realism of these synthetic images and reduce artifacts introduced by the simulation process, the team applied domain adaptation using CycleGAN, translating them from the synthetic domain to a more photorealistic representation. Both real and enhanced synthetic images will be integrated into LRDD version 3, scheduled for release later this year.

Recognizing that most current detection algorithms reduce input resolution (e.g., to 640×640), resulting in loss of fine-grained image detail, the team investigated techniques to preserve high-resolution information. One such method, Slicing Aided Hyper Inference (SAHI) [20], processes high-resolution images by slicing them into smaller patches compatible with the input dimensions of existing object detectors. This approach avoids down sampling and significantly improves detection performance by preserving pixel-level details. Given the increased computational load introduced by patch-based processing, the team further explored the use of CLIP [22], a popular vision-language model, as a pre-filtering mechanism. The team assessed its effectiveness in selectively identifying image patches likely to contain drones, thereby reducing the number of patches that require full object detection inference and improving overall efficiency.

Beyond detection, the team's work also addressed the broader issue of UAS self-assessment under degraded visual conditions. Just as human drivers may choose to pull over during dense fog when visibility is compromised, autonomous UAS must be capable of evaluating their perception quality and making decisions accordingly. This self-assessment is critical for safe operation in adverse conditions such as heavy rain, smoke, fog, or snow. To support this capability, the team investigated several image quality assessment metrics that correlate with object detection performance and can be applied without requiring reference images. The team identified a set of metrics suitable for onboard use in real-time, enabling autonomous systems to assess the reliability of their visual inputs and adjust their behavior accordingly.

Recommended guidance: The team gives the following vision-related guidance and its reasons.

- **Enhance long-range data set to assess vision-based perception algorithm's performance as a function of the target distance**

To address limitations in existing datasets—particularly in long-range detection, environmental variability, and accurate range estimation—the team developed the LRDD dataset during the first two years of the project. In the third year, the team released LRDDv2, an enhanced version containing 39,516 images, including over 8,000 frames with precise ground-truth distance annotations. LRDDv2 significantly expands the dataset's diversity in terms of weather conditions, lighting variations, occlusion scenarios, and background complexity. Most notably, it introduces long-range distance data, enabling detailed analysis of vision-based perception algorithm performance as a function of target distance. This makes LRDDv2 the first drone detection dataset to explicitly support performance benchmarking over long-range scenarios, filling a critical gap in current drone perception research. The team's effort continues in further expansion and enhancement of LRDD datasets.

- **Project future vision-based algorithm performance**

Using the team's LRDDv2 dataset, the team evaluated drone detection performance by implementing the state-of-the-art object detector YOLO. The team's results showed that detection accuracy is highest at close range (<100 ft), with performance generally declining significantly at longer distances. When incorporating the SAHI framework, the team observed similar trends; however, the accuracy drop at longer ranges was much less severe, suggesting SAHI's potential in enhancing long-range detection. Despite its advantages, SAHI introduces a significant computational overhead. To preserve image resolution, SAHI divides each image into 20+ overlapping patches and applies a full object detector to each one, increasing processing time and computational load by approximately 20x compared to standard detection pipelines. To address this challenge, the team is exploring a more lightweight alternative using pre-trained contrastive models. This method rapidly evaluates image patches for potential object presence by generating contrast scores, which are then combined into an attention heatmap. The heatmap guides a focused object detection pass only on likely regions, greatly reducing computational demand while maintaining accuracy. The team believes this approach is well-suited for future UAS detection systems, especially those operating under tight weight and power constraints. The team's ongoing research continues to refine this methodology for real-world deployment.

- **Develop a metric to determine perception quality threshold for terminating UAS operations for safety**

Certain adverse environmental conditions—such as rainstorms, heavy fog, snow, or sandstorms—can significantly degrade camera-based perception, potentially requiring the termination of UAS operations for safety. To understand the impact of such degradation on drone detection, the team first fine-tuned YOLOv5 on the team's real-world dataset, LRDD, and evaluated the model's robustness by introducing salt-and-pepper noise at varying Signal-to-Noise Ratio (SNR) levels to the LRDD test set. The results revealed that even minor reductions in SNR led to substantial drops in detection accuracy, highlighting the sensitivity of vision-based models to image quality degradation. This finding highlights the need for an onboard image quality assessment metric that enables UAS systems to autonomously evaluate perception reliability in real time and make informed operational decisions. Traditional image quality metrics such as SNR, Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index Measure (SSIM) are not practical in UAS operations, as they require access to both pristine and degraded versions of the same scene, which are rarely available in real-world settings. To overcome this, the team explored two no-reference image quality metrics: Blind/Referenceless Image Spatial Quality Evaluator (BRISQUE) and Deep Bilinear Convolutional Neural Network (DBCNN). To evaluate the effectiveness of these metrics, the team created a synthetic dataset using Unreal Engine 4.27 and AirSim, simulating a 3D urban environment under four weather conditions: clear, light fog, moderate fog, and heavy fog. The team then fine-tuned YOLOv11 on clear-weather images and assessed detection performance under increasing fog intensity. As expected, detection accuracy declined with worsening fog conditions. Both BRISQUE and DBCNN showed a reasonable correlation with detection performance, suggesting potential for real-world application. However, some inconsistencies—particularly under light fog conditions— indicate the need for further analysis and refinement of no-reference image quality metrics for UAS perception systems.

The data created in this project is available upon request from the PI David Han at Drexel.

9.2 Long-Range Drone Detection (LRDD) Dataset

Previously, the team published the LRDD dataset [1], a comprehensive training dataset aimed to alleviate the scarcity of practical drone detection datasets. This dataset encompassed a set of different UAS types, flight patterns, variations in altitude, and environmental conditions. It is also suitable for urban operational scenarios due to its inclusion of different backgrounds and distant targets captured within complex city environments. The dataset consists of 22,500 images having a resolution of 1920x1080 pixels and depicts multiple drones within certain frames. As the dataset was intended to capture videos of drones at substantial distances, it led to image frames where the drones occupied 50 or fewer pixels. It also showcases various weather conditions, illumination variations, and scenarios with occlusion and drone blending with the background. Existing datasets for object detection using drone images pose limitations, including limited volume, insufficient variety in terms of lighting, climatic conditions, backgrounds, camera angles, and blending. Additionally, many of these datasets lack long-range images featuring smaller drones. One significant limitation of currently available drone datasets, including LRDD, is the lack of object range information, which is essential for effective collision avoidance. This omission poses a major challenge in training computer vision algorithms for autonomous drone operations, where spatial awareness and distance estimation are critical for safe navigation.

Since publishing LRDDv1, the team has expanded the dataset and published LRDDv2 [2], comprising 39,516 images, and offering diverse environments and scenarios. Among these, 8,000 annotated images include range information, enhancing the dataset's utility for developing models capable of estimating distances to drones. Distinctive in its emphasis on long-range drone imagery, these images are crucial for the development of detection models that effectively identify drones at great distances, where they appear as mere specks against vast backgrounds. In continuation, the team enumerates the most significant challenges that the presented dataset covers as follows.

Long Distance Images The dataset features a substantial number of images capturing drones up to distances of 350 ft from the camera. This inclusion of long-range imagery is vital for enhancing prediction models specifically designed for distant UAS detection, a key requirement for real-world applications. Such diverse distance coverage ensures that detection systems trained on this dataset are well-prepared for practical surveillance challenges. An illustrative example of long-distance drone imaging within this dataset is presented in Figure 35.

Moving Camera Reflecting realistic operational conditions, the dataset incorporates sequences where not only the drone is in motion, but the camera is also moving. This addition presents a real-world challenge that enables the training of more robust detection models.

Occlusion LRDDv2 contains a variety of cases where drones are partially or fully occluded by objects such as trees or bridges. Figure 36 displays an instance of a drone that is partially obscured by tree leaves.

Background Blending The team's dataset includes a range of images where drones blend into the background. Figure 37 demonstrates this challenge with a drone operation in an urban park setting, where the UAS's appearance is congruent with the environment, necessitating advanced pattern recognition for detection.

Illumination Challenges Collected under different lighting conditions, from bright daylight to the low light of the evening, the dataset also tackles common issues like glare. Figure 38 exemplifies such a scenario where glare affects the visibility of the drone.

Weather Conditions The dataset features images taken in various weather conditions, enhancing the robustness of detection under different meteorological scenarios. The weather categories are as follows: 1) Sunny, 2) Clear, 3) Cloudy, and 4) Rainy. Figure 39 illustrates an example captured on a cloudy day.

Multiple Object Detection In many instances, the dataset presents the challenge of detecting multiple drones within a single scene. Figure 40 highlights this by showing two drones, with one drone blending into the urban architecture.

Diverse Backgrounds LRDDv2 extends across different environments, from urban to rural settings, which aids in training models on a wide array of backgrounds. The main background categories are City, Grass, Sky, and Water. Figure 41 portrays an example captured in a rural setting that also includes multiple drones, thus addressing two mentioned challenges. Figure 42 displays the background distribution within the proposed dataset.

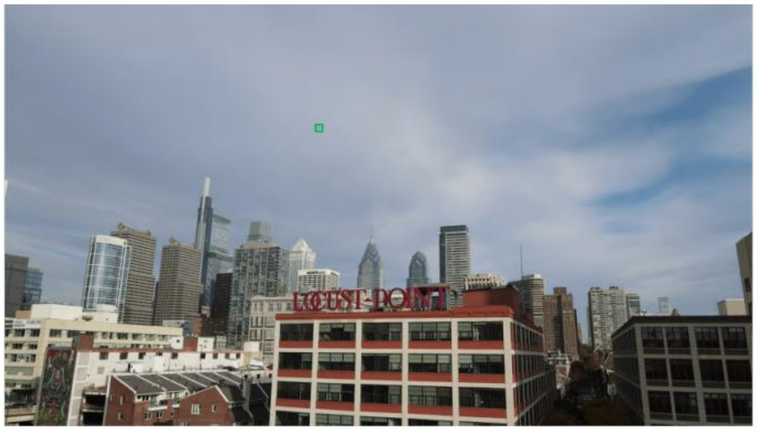


Figure 41. Example of a drone captured at a long distance.

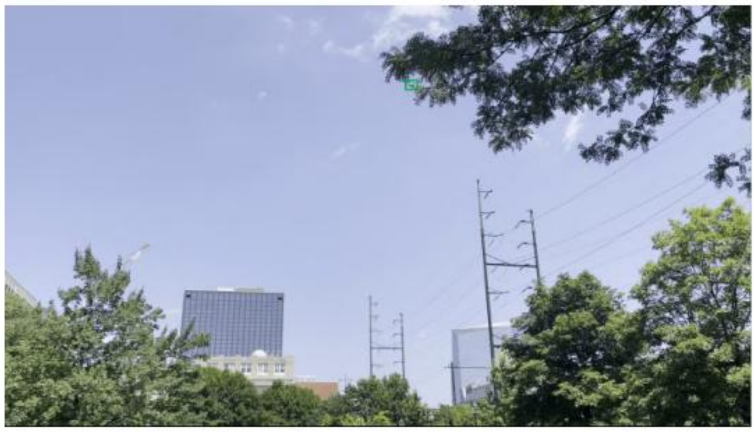


Figure 42. Example of a drone partially obscured by tree leaves.

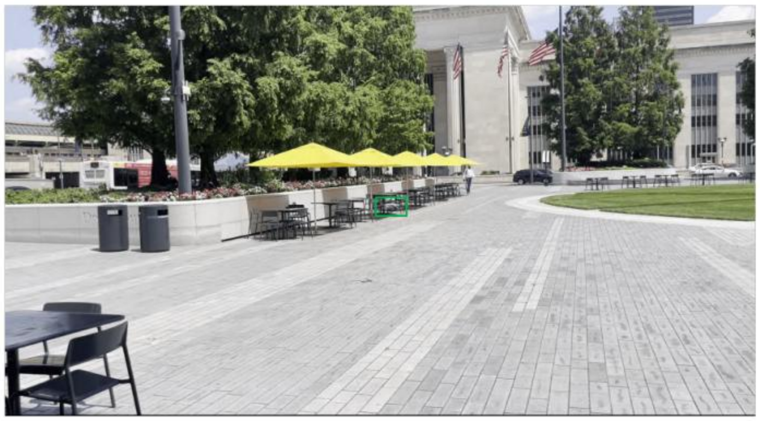


Figure 43. Example of UAS blending into an urban park environment, illustrating the challenge of detecting drones against complex backgrounds within the LRDDv2 dataset.

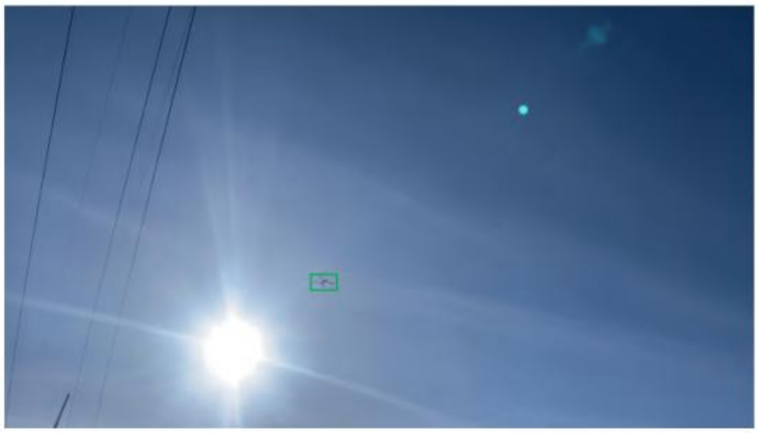


Figure 44. Example of a drone image featuring a glare challenge.

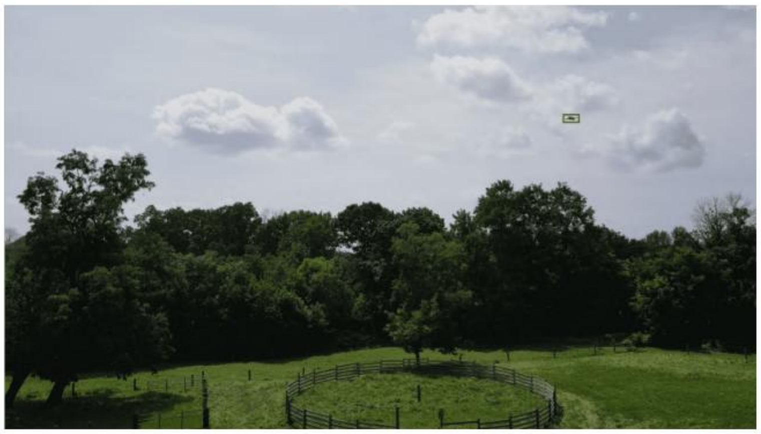


Figure 45. Example of a drone captured in a rural environment under cloudy weather conditions.

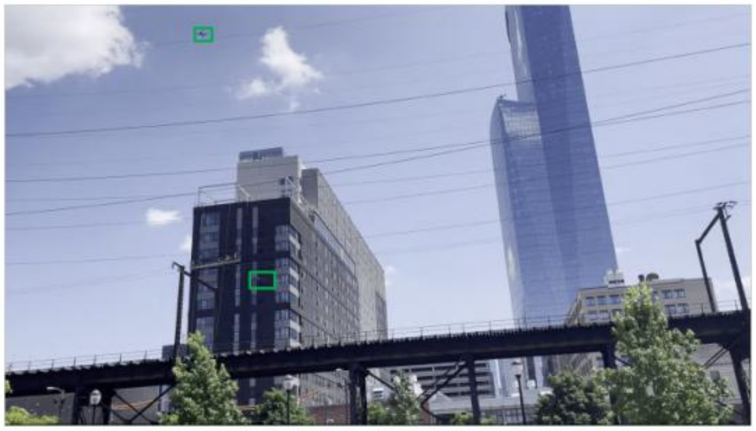


Figure 46. Example of Complex UAS Detection Scenario from LRDDv2: Aerial view showing two drones, with one exhibiting near-perfect background blending against an urban structure, posing a significant challenge for detection algorithms.

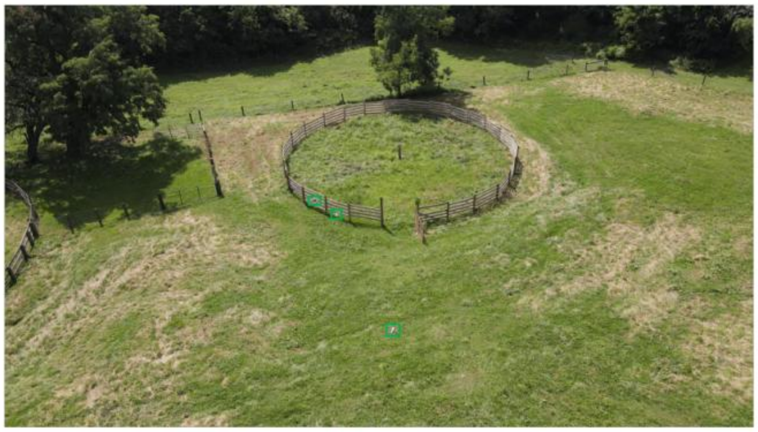


Figure 47. Example showcasing three flying drones in a rural environment, addressing two challenges: diverse surroundings and the presence of multiple drones.

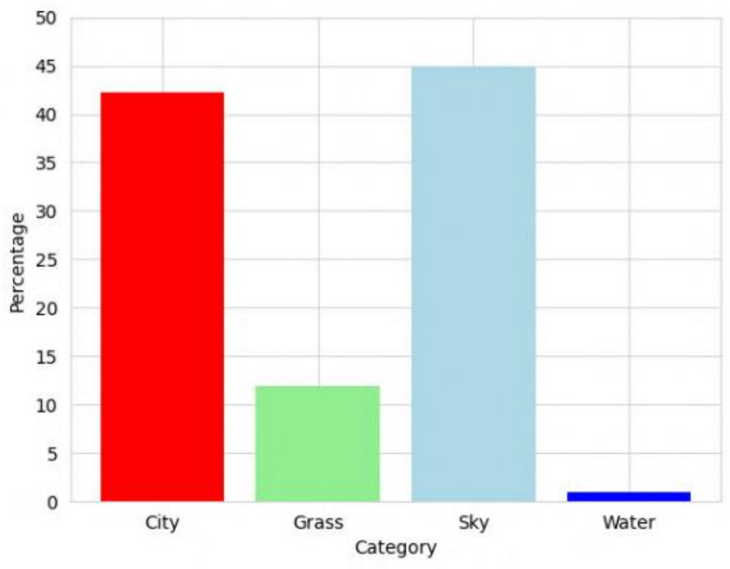


Figure 48. Distribution of Background by percentage.

9.2.1 Data Collection

The images in the dataset were collected using the DJI Mavic Air 2 drone, capable of collecting 1080p 30 fps video footage, as well as using cellular cameras (iPhone 12/15 Pro Max and Google Pixel 6) capable of capturing 30 fps 1080p video footage. In both cases the camera collected video footage of other drones, which were the DJI Mini 3 and the DJI Mavic Pro. The footage includes many scenarios, including multiple and single drone frames, diverse backgrounds, and a wide range of distances from the camera.

9.2.2 Data Preprocessing

To extract the images in the dataset, frames from the video footage are sampled at regular intervals of 10 fps. These images are then synchronized with the range information collected from the flight logs of the drones. The range information collected by the drone includes height and distance from the home point where the flight began. In the case of a stationary cellular camera collecting footage at ground level, the absolute distance from the drone is calculated with the following formula:

$$Distance_{Absolute} = \sqrt{Distance^2_{Horizontal} + Height^2_{Drone}}$$

For aerial footage, both the drone and camera start from the same home position. The horizontal distance is calculated using GPS coordinates (latitude and longitude) of the drone and camera, applying the haversine formula. The absolute distance is then determined using the following equations:

$$\Delta Height = \left|Height_{Target} - Height_{Camera}\right|$$

$$Distance_{Absolute} = \sqrt{Distance^2_{Horizontal} + (\Delta Height)^2}$$

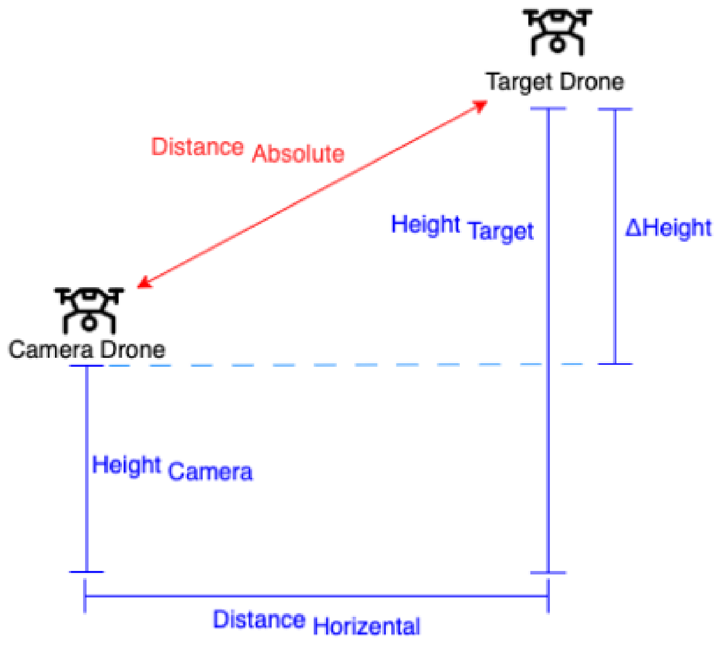


Figure 49. Illustration of distance calculation in the dataset in the case of the footage being collected by an aerial drone.

Figure 43 shows an illustration of the distance calculation process, highlighting the relationship between the horizontal distance, height delta, and absolute distance in the case of footage being collected by an aerial drone. Data was labelled using the YOLO format. Both manual and assisted bounding box generation methods were used. For the automated bounding box generation, a sample of manually labelled images was used as training data for a YOLOv5 algorithm to automatically detect possible matches. This process speeds up labelling and allows for the boxes to merely be tweaked by the labeler. For the manual labelling and bounding box correction, the open-source Python-based software LabelImg was used. This tool provides a simple GUI to create and edit bounding boxes and then automatically generate the labels in YOLO format.

9.2.3 Dataset Benchmarking

The team evaluated the YOLOv8m model for performance improvements brought about by the LRDD v2 data through a strict benchmarking mechanism across a number of datasets that included Drone-vs-Bird [3], Detfly [4], and UAS-Detect [5]. This detailed study focused on the adaptation and realization of YOLOv8 model under different datasets and detection scenarios. The benchmarking process focused on two primary metrics: mean Average Precision (mAP) at IoU threshold 0.5 (mAP@50) and mAP measured over IoU thresholds from 0.5 to 0.95 (mAP@50-95), offering details both on the precision of detection at a common threshold as well as a more fine-grained assessment across a range of detection accuracies.

The experimental results presented in Table 20 show the performance of the YOLOv8m model in detecting drones across different datasets, which include Detfly and UAS-Detect, when trained with different dataset combinations: Drone-vs-Bird, LRDDv2, and both datasets.

As shown in Table 20, the model's performance increases in all measures while it is being trained on the union dataset of Drone-vs-Bird and LRDDv2, yielding the highest mAP@50 and mAP@50-95 scores for both the Detfly and UAS-Detect evaluation datasets. Training the model exclusively on LRDDv2 yields better inference outcomes on both evaluation datasets compared to using the Drone vs Birds dataset alone; however, the most effective results are obtained when training incorporates both datasets together. This enhancement highlights the complementary nature of the Drone-vs-Bird and LRDDv2 datasets. While the Drone-vs-Bird dataset provides a broad range of general drone and bird images, the LRDDv2 dataset introduces specific challenges inherent to drone detection, including long-range detection capabilities, varied lighting, and scenarios involving blending and occlusion.

The improvement in model performance on the Detfly dataset in the mAP@50 metric is from 0.376 to 0.463, and for mAP@50-95, it increased from 0.14 to 0.26 when trained on aggregated datasets, allows a conclusion that the model acquired better skills to generalize over diverse scenarios. Equally on the UASDetect dataset, training with the two datasets provides a noticeable boost that increases the mAP@50 from 0.510 to 0.644 and the mAP@50-95 from 0.22 to 0.32. The improvements represent the strength brought by the LRDDv2 dataset to the model when dealing with challenging drone detection scenarios.

The results provide strong evidence for the benefit of using varied training data, especially the LRDDv2 dataset that is specifically tailored towards the subtle difficulties in drone detection at long range. The improved performance on both Detfly and UAS-Detect evaluation datasets emphasizes the significance of challenging and comprehensive datasets in the development of drone detection technologies.

Table 20. YOLOv8 detection accuracy on Detfly and UAS-Detect using different training datasets.

Training Dataset	Evaluation Dataset	mAP@50	mAP@50-95
Drone-vs-Bird	Detfly	0.376	0.14
LRDDv2	Detfly	0.458	0.27
Drone-vs-Bird+ LRDDv2	Detfly	0.463	0.26
Drone-vs-Bird	UAV-Detect	0.510	0.22
LRDDv2	UAV-Detect	0.562	0.30
Drone-vs-Bird+ LRDDv2	UAV-Detect	0.644	0.32

9.2.4 Detection Probability Versus Bounding Box Area

An essential component of drone detection algorithm evaluation is the investigation of how detection likelihood correlates with bounding box dimensions, which serve as a proxy for the UAS's range from the capture device. In this study, the team leveraged the YOLOv8m model trained on three scenarios: the Drone-vs-Bird dataset, the LRDDv2 dataset, and a combination of

Drone-vs-Bird + LRDDv2 datasets, to explore this relationship. Figure 44 visualizes the probability of detection by bounding box area across these scenarios.

The figure demonstrates a dependency of detection rates on bounding box sizes. As the distance increases (represented by smaller bounding boxes for a fixed object), the detection probability decreases across all training scenarios. YOLOv8 trained on Drone-vs-Bird alone shows the lowest detection probabilities, especially for smaller bounding boxes. In contrast, training on LRDDv2 yields better results, and the highest detection rates are observed when YOLOv8 is trained on the combination of Drone-vs-Bird + LRDDv2, showcasing the complementary strengths of both datasets.

This pronounced decline in detection probability with decreasing bounding box sizes highlights the critical need for datasets tailored to long-range drone detection. The development and refinement of models to improve their long-range detection capabilities is indispensable for addressing the challenges posed by detecting distant UAS in real-world surveillance applications.

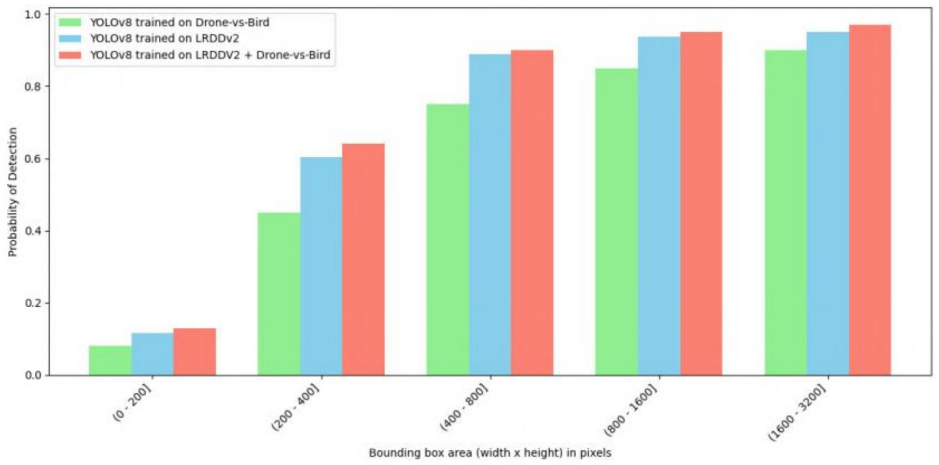


Figure 50. Detection Probability by Bounding Box area (pixel): The graph shows the YOLOv8m model's detection probability as a function of bounding box size.

9.2.4.1 Progress Toward LRDDv3

Since publishing LRDDv2, the team has continued to collect data to improve the size and diversity of the dataset. Particularly, version 2 had a large number of city and sky backgrounds, but limited grass and water backgrounds. Additionally, the number of images taken during unfavorable weather was limited, and the number of images with drone distance information was only a small portion of the dataset. To fill in these gaps, the team collected an additional 30k images, all with distance information. These images also contain many collection days in rainy and snowy weather conditions, and with water and grass backgrounds. Figures 45-47 show examples of the increased diversity in images collected for the final version of LRDD. The team is currently processing the range metadata and labeling drone instances in the collected images and will publish the completed version 3 of the dataset later this year.

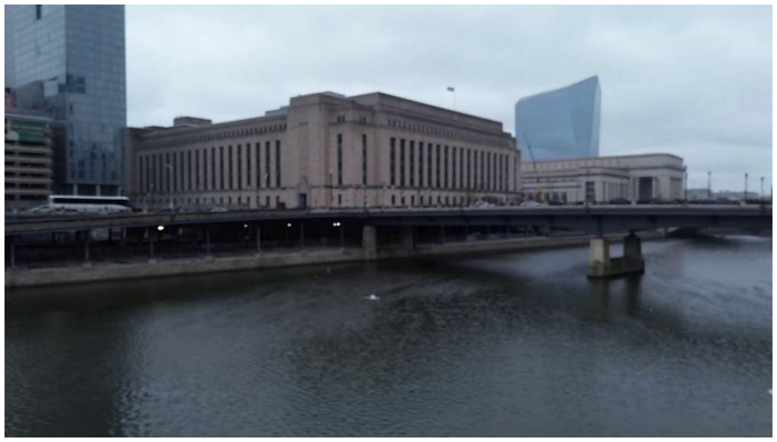


Figure 51. Example of a drone captured over a river in rainy weather conditions. Rain droplets on the drone camera lens produced blurry images, reflecting the conditions.

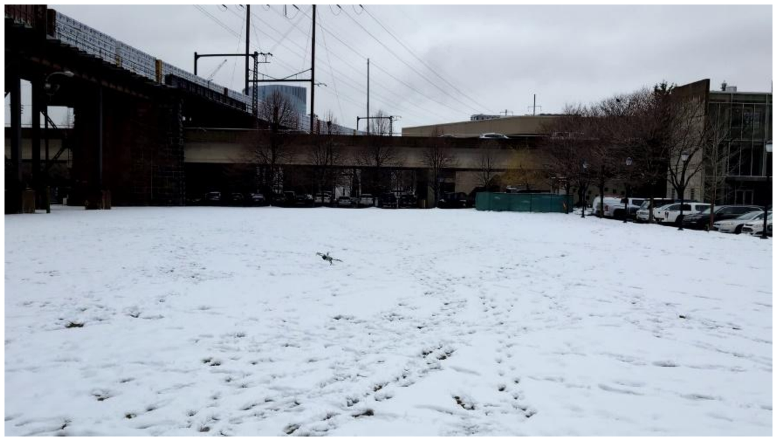


Figure 52. Example of a drone captured in snowy conditions.

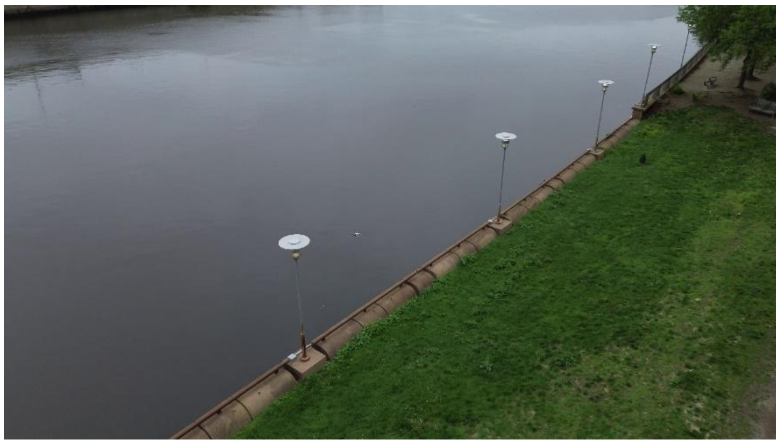


Figure 53. Example of a drone captured with water and grass in the background.

9.3 Simulated Image Data

9.3.1 Generation of synthetic images and their utility proven in the 2nd year effort

To further expand dataset diversity beyond what can be achieved through real-world image collection alone, the team explored the use of simulation as a complementary approach. This strategy not only supports the development of more robust detection algorithms but also enables the generation of datasets that encompass a broader range of conditions and operational scenarios. In the project's second year, the team utilized UE version 4.27 [6] to simulate drone imagery across diverse backgrounds and viewpoints. The use of simulation also allowed a bypass of manual annotation—bounding boxes and other labels essential for training are automatically generated with precise accuracy, as all ground-truth information is inherently known within the simulated environment.

As described in detail in the second year report, a 3D forest environment was created in UE using publicly available maps, and five drone models—designed in SolidWorks and imported via the DataSmith plugin—were placed into the scene to simulate diverse real-world scenarios. A Python script using the AirSim module then positioned a virtual camera at random locations to capture annotated images and segmentation maps, allowing automatic extraction of bounding boxes for training data. The 3D models of the drones are depicted in Figure 48. A sample generated image and its scene segmentation are shown in Figure 49, while Figure 50 shows an image capturing a drone in a forest in low lighting conditions. The 20,000-image UE dataset includes 500 images per drone model across four environments, with varied lighting (day and night), distances, and random orientations, positioning the target drone within the central region of each image to reflect real-world data distributions. Figure 51 shows the distribution of the drone's bounding box location in the 20,000 Unreal Engine images.

The usefulness of the simulated dataset was evaluated using YOLOv8m by measuring mAP on three real-world datasets: DetFly, Drone Detection, and UAS Detect. Training with UE-generated simulated data improved detection performance compared to training without it, particularly when using an 80/20 training-validation split.

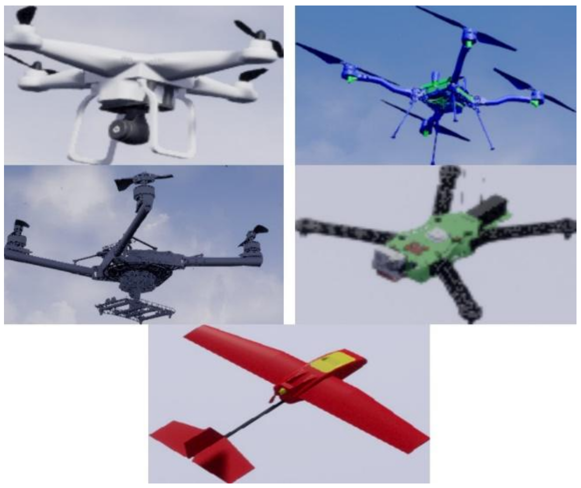


Figure 54. Five drone models.

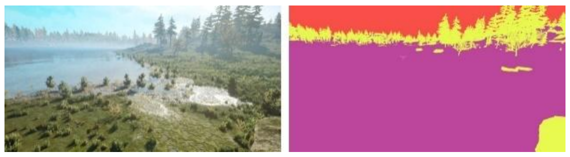


Figure 55. Generated image and the segmentation map.

Figure 56. Sample of a night-time generated image.

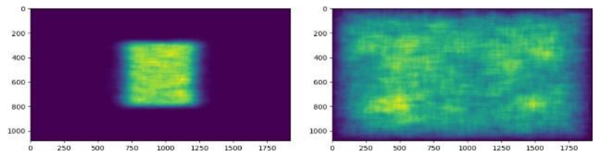


Figure 57. Bounding box heatmap.

Table 21. Performance of YOLOv8m trained with or without simulated data.

Trained Models	DetFly		Drone Detection		UAV Detect	
	mAP50	mAP50-95	mAP50	mAP50-95	mAP50	mAP50-95
DvB50	0.462	0.22	0.66	0.282	0.58	0.288
DvB50+UE	0.475	0.24	0.66	0.282	0.574	0.306
DvB80	0.461	0.226	0.645	0.292	0.61	0.328
DvB80+UE	0.487	0.238	0.677	0.3	0.575	0.304

9.3.2 Generative Adversarial Network (GAN) for dataset augmentation

The second phase of the team's augmentation process addresses the critical issue of domain shift between synthetic and real-world drone imagery. While synthetic data is valuable for scaling and diversifying training datasets, the visual differences between domains can impair model performance. This section outlines how the team mitigated this issue using CycleGAN [15] for unsupervised image-to-image translation. While synthetic data provides a scalable and controlled means of generating diverse training samples, a persistent challenge is the domain shift between synthetic and real-world imagery. This shift arises from visual discrepancies such as texture, lighting, and noise patterns, which can lead to a drop in model performance when transitioning from training on synthetic data to deployment in real-world settings.

9.3.3 CycleGAN For Bridging the Domain Gap

Generative Adversarial Networks (GANs) [16] are a class of deep learning models composed of two competing networks: a generator and a discriminator that are trained simultaneously in a minimax game. The generator attempts to produce realistic data samples, while the discriminator tries to distinguish between real and generated data. This adversarial training process enables GANs to generate high-quality, realistic images, making them especially useful for tasks involving domain adaptation.

Cycle Consistency GAN (CycleGAN), a type of GAN, is particularly well-suited for the team's needs as it enables unpaired image-to-image translation, a critical feature given the lack of corresponding real-synthetic image pairs in drone detection datasets. It ensures cycle consistency, meaning the translated image can be mapped back to its original form. This mechanism preserves the structural integrity of objects (e.g., drones) while allowing the translated images to adopt the appearance characteristics of the target (real-world) domain. As a result, CycleGAN helps reduce the domain gap without altering underlying annotations, making it ideal for enhancing the utility of synthetic datasets.

9.3.3.1 Implementation Details

The team trained CycleGAN using 6,500 synthetic images generated in the UE dataset and 5,500 real images curated from various drone-related datasets. To ensure compatibility with high-resolution inputs, the team adjusted the model's load size to 1080 and crop size to 360. The real images were selected to closely resemble the synthetic environments in terms of background and object appearance, facilitating more effective domain transfer [17]. The result of this translation process is referred to as the **UE_CycleGAN** dataset, a set of synthetic images visually aligned with real-world conditions while retaining precise annotations from simulation. Figure 52 illustrates examples of synthetic images and their corresponding translations into the real domain using CycleGAN, highlighting the enhanced visual realism achieved through domain adaptation.

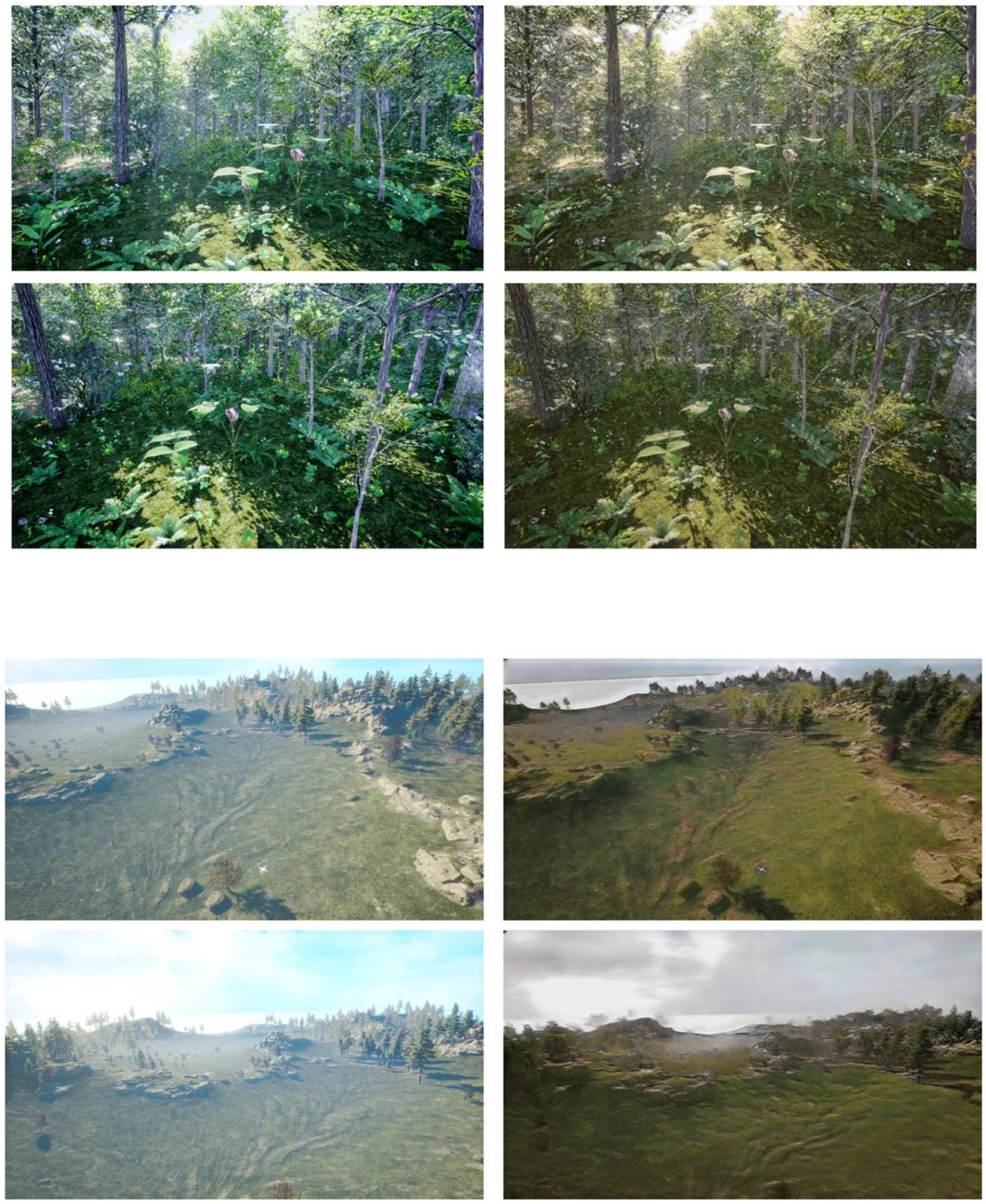


Figure 58. Comparison of synthetic images (left) and their CycleGAN-translated real-domain counterparts (right).

9.3.3.2 Impact on Drone Detection

As demonstrated in the experiments in Table 22, models trained on real datasets augmented with UE_CycleGAN images consistently outperformed those trained on synthetic or real data alone. This performance boost was particularly significant when the available real training data was limited. The integration of CycleGAN-translated data not only improved mAP scores across multiple real-world test datasets (DetFly, UAS Detect, Drone Detection, and New Batch) but also enhanced the generalization ability of the YOLOv8 detection model. The results presented in Table 22 conclusively demonstrate that including both synthetic and translated images in the real dataset enhances drone detection capabilities. Notably, the most substantial improvement is observed when augmenting real data with translated data. These findings emphasize the significance of the proposed approach in enhancing drone detection performance.

Table 22. Comparison of YOLOv8m performance on real data with and without augmentation using synthetic and translated data.

Trained Models	DetFly		Drone Detection		UAV Detect		New Batch	
	mAP50	mAP50-95	mAP50	mAP50-95	mAP50	mAP50-95	mAP50	mAP50-95
DvB50	0.462	0.22	**0.66**	**0.282**	0.58	0.288	0.3825	0.18
DvB50+UE	0.475	0.24	0.66	0.282	0.574	0.306	0.3904	**0.2087**
DvB50+UE_CycleGAN	**0.524**	**0.262**	0.614	0.263	**0.601**	**0.334**	**0.3954**	0.1957
DvB80	0.461	0.226	0.645	0.292	0.61	0.328	0.3774	0.2237
DvB80+UE	0.487	0.238	0.677	0.3	0.575	0.304	0.41	0.2161
DvB80+UE_CycleGAN	**0.51**	**0.246**	**0.694**	**0.311**	**0.612**	**0.352**	**0.4173**	**0.2455**

9.3.4 Additional synthetic dataset generation for urban settings

The team generated an additional synthetic drone detection dataset using Unreal Engine 4.27 and the AirSim plugin, simulating three distinct urban environments to increase scene diversity and realism. Four different drone models were used to capture variations in shape, size, and appearance. Images were rendered under four environmental settings—clear (clean), light fog, moderate fog, and heavy fog—to systematically study the impact of weather-induced visibility degradation on detection performance. The motivation behind creating this dataset was twofold: first, to overcome the limitations of real-world data collection, which is often costly, time-consuming, and difficult to control; and second, to enable a controlled evaluation of how environmental factors such as fog affect both image quality and object detection accuracy. The resulting dataset contains approximately 40,000 high-resolution images and serves as a scalable and customizable benchmark for developing and testing robust vision-based drone detection models. In Figure 53, the drone models are shown, and the team can see samples of simulated images in Figure 54.

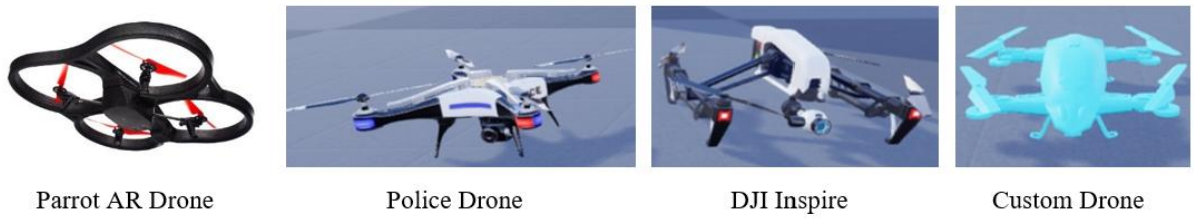


Figure 59. Different drone models used in urban environments.

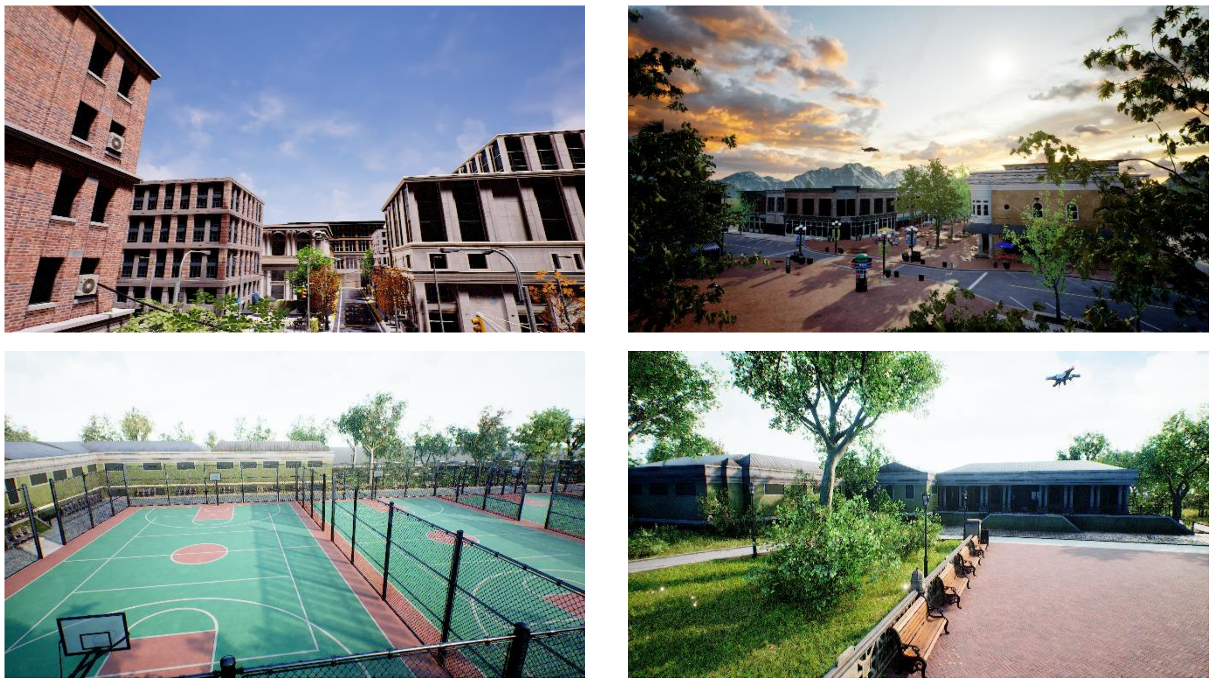


Figure 60. Samples of generated images in urban environments.

9.4 Methods for Long-Range Detection

The creation of a diverse dataset of drone images allowed the team to effectively explore different approaches for long-range detection and analyze their performance at long range. Existing approaches to scene perception mostly rely on supervised learning techniques using popular models such as YOLO [18], a CNN based framework that is widely used in industry. More recently, Detection Transformers (DETRs) such as Deformable DETR have risen to form the current state of the art in detection [19]. However, detecting small drones from long distances remains a significant challenge to these models due to the minimal size of the drones within the image frame. Conventional detection methods struggle with this task, as small drones typically occupy only a few pixels, making accurate classification difficult. While higher resolution cameras can capture more details, the challenge persists with traditional object detection algorithms. For example, state-of-the-art algorithms like the YOLO series require fixed input image resolutions, often much smaller than those provided by high-resolution cameras. In the case of YOLOv5, 4K images must be downscaled to 640 × 640 pixels. This downscaling drastically reduces the pixel size of small objects, such as shrinking a 20×20 pixel drone to just 4×6 pixels, significantly degrading detection accuracy.

9.4.1 Slicing-Aided Hyper Inference (SAHI)

To address the challenge of resolution loss in conventional object detectors, the SAHI [20] method has emerged as a promising solution. SAHI enhances object detection by dividing high-resolution images into smaller overlapping patches that match the input resolution of object detectors like YOLO or SSD, as shown in Figure 55. By ensuring that each patch fits within the detector's required resolution, SAHI eliminates the need for downscaling, preserving crucial pixel information. This approach enhances the detection of small, distant objects by maintaining their visibility and detail within each patch, leading to more accurate identification. SAHI can be combined with other deep learning models and image enhancement algorithms to further improve detection performance.

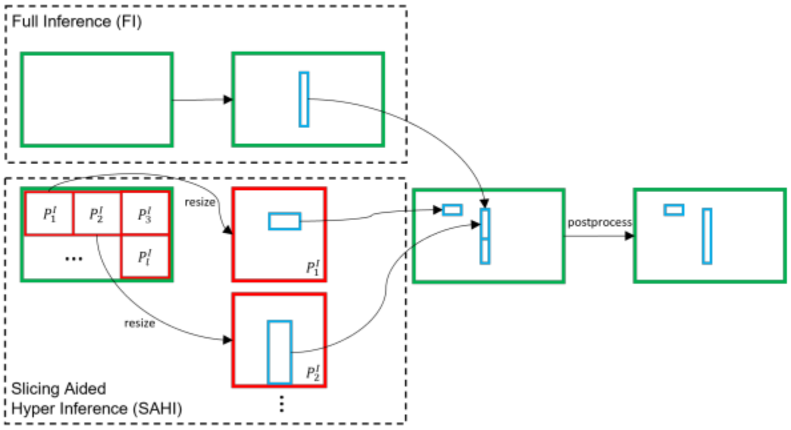


Figure 61. Overview of the Slicing-Aided Hyper Inference (SAHI) approach.

To analyze the effectiveness of SAHI at long-range drone detection, the team implemented SAHI with a YOLO v5 detection model and tested its performance on the team's LRDD v1 dataset [21]. The baseline model was tested on both full images and image patches. This approach helps determine whether any performance gains are solely attributable to slicing images during the testing phase, even though the model was not trained on sliced patches from the dataset. The SAHI model is evaluated using three different approaches: full images, image patches, and a combination of both. Each approach is assessed separately to determine performance. The detection results are presented in Table 23. Both models were evaluated using the LRDD test set. Average Precision (AP) and Average Recall (AR) were calculated for various bounding box sizes to assess performance variations based on the number of pixels occupied by the object in the image. The objects were categorized into three groups as follows:

- Small objects: The area is less than 32 × 32 pixels (i.e., area < 1024 pixels).
- Medium objects: The area is between 32×32 and 96×96 pixels (i.e., 1024 ≤ area ≤ 9216 pixels).

- Large objects: The area is greater than 96 × 96 pixels (i.e., area ≥ 9216 pixels).

Table 23. Detection results of baseline and SAHI model trained and tested on LRDD dataset.

Model	Evaluation Dataset	AP @50 All	AP @50-95 All	AP @50 Small	AP @50-95 Small	AP @50 Medium	AP @50-95 Medium	AP @50 Large	AP @50-95 Large	AR @50-95 Small	AR @50-95 Medium	AR @50-95 Large	FPS
Baseline	Full Image	0.572	0.388	0.443	0.248	0.957	0.722	0.998	0.920	0.274	0.756	0.935	31.25
Baseline	Image Patch	0.413	0.200	0.378	0.147	0.641	0.423	0.122	0.100	0.184	0.461	0.104	2.38
SAHI	Full Image	0.482	0.308	0.323	0.165	0.936	0.637	0.990	0.888	0.189	0.682	0.682	34.39
SAHI	Image Patch	0.933	0.547	0.924	0.485	0.978	0.485	0.809	0.490	0.541	0.800	0.518	2.38
SAHI	Both	0.931	0.540	0.921	0.455	0.986	0.723	0.994	0.861	0.515	0.778	0.886	2.27

The overall performance of the SAHI model shows significant improvement compared to the baseline model. As expected, the key factor in this improvement is the model's ability to detect small objects more effectively through slicing inference. The results indicate that simply applying patches to the baseline model does not enhance its performance. For the model to achieve higher accuracy, it must be trained on a fine-tuned dataset that includes sliced patches of the full images.

Additionally, the performance for medium-sized objects remains relatively similar, with a slight improvement observed due to slicing inference. This suggests that while most medium targets can be detected using full-image inference, image patches contribute positively to the detection process. For large objects, the performance of the two models was comparable.

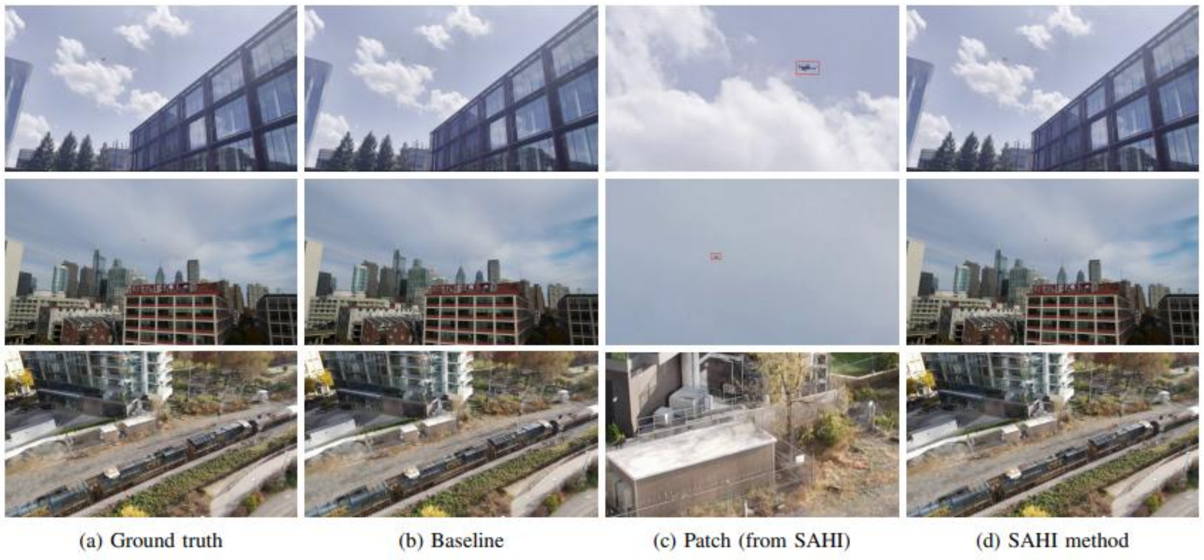


Figure 62. Examples of detection from the baseline and SAHI models on the LRDD dataset. (a) True label (b) No detection from the baseline model (c) Sliced patch from the SAHI method that correctly detected the target (d) Reconstructed detection from the SAHI method that correctly detected the target.

A more detailed analysis can be conducted by examining a few examples from the LRDD dataset and comparing the performance of the baseline and SAHI models. Figure 56 shows that the baseline model cannot detect targets when the drones are too small. In contrast, the SAHI model successfully detects these small targets. Additionally, the team can observe the sliced patches generated by the SAHI method, which reveal how the model identified the targets without the loss of image quality, resulting in significantly improved performance. This performance improvement, however, is accompanied by additional computational cost. The SAHI model requires multiple inferences per image compared to a single image processing in the baseline model, as it processes both multiple image patches and the full image, thereby requiring more computational power. This poses a challenge to real-time detection.

To determine the impact of patch sizes on drone detection, the team conducted experiments with different numbers of sliced patches in each instance. Table 24 presents the detection results on the LRDD test set, comparing the SAHI model with various numbers of image patches. As expected, increasing the number of sliced patches significantly improves detection performance for small objects. However, the accuracy for medium and large drones remains unchanged, once again demonstrating that the models do not heavily rely on image patches for detecting these larger objects. The performance improvement peaks at around 20 sliced patches. At this point, the SAHI model has acquired the necessary details from the images, and adding more patches only increases the computational cost without further benefits.

Table 24. Detection results for different numbers of sliced patches on LRDD dataset.

Model	Number of Slices	AP_{50} All	AP_{50} Small	AP_{50} Medium	AP_{50} Large
Baseline	-	0.572	0.443	0.957	0.998
SAHI	4	0.772	0.696	0.977	0.998
SAHI	9	0.893	0.861	0.979	0.997
SAHI	20	0.931	0.921	0.986	0.994
SAHI	36	0.934	0.922	0.981	0.990

Table 25. Detection results for baseline and SAHI model trained on LRDD and tested across different. datasets

Model	Test Dataset	AP_{50} All	AP_{50} Small	AP_{50} Medium	AP_{50} Large
Baseline	DvB	0.351	0.175	0.572	0.671
SAHI	DvB	0.817	0.880	0.780	0.766
Baseline	DetFly	0.330	0.062	0.396	0.283
SAHI	DetFly	0.441	0.439	0.473	0.441
Baseline	Synthetic	0.177	0.140	0.296	0.406
SAHI	Synthetic	0.296	0.239	0.429	0.604

To analyze the effectiveness of the SAHI method in detecting small drones, the team evaluated its performance across various datasets. The experimental results presented in Table 25 show the performance of both the baseline and SAHI models, which were trained exclusively on the LRDD dataset and tested on the Drone vs. Birds, DetFly, and GAN-translated Synthetic datasets using a zero-shot approach. Note that, due to the extensive number of video images in the Drones vs. Birds dataset, only a portion of the dataset was used, with a focus on including target drones of varying sizes. The results show a significant improvement in detecting small and medium-sized drones with the SAHI model across all datasets. For Drone vs. Birds and DetFly, there is a remarkable increase in AP50 Small, where the baseline model struggled to detect them effectively. Additionally, the method even enhanced the detection of large drones, highlighting how sliced patches aid in detecting larger targets where full-image inference falls short.

9.4.2 SAHI Performance vs Range

In addition to the analysis of SAHI on LRDDv1, the team performed testing on the 8k images with range information in LRDDv2 to gain a better understanding of how detection performance changes with drone distance from the camera. For this experiment, the team trained a YOLO v8 model on the 26k LRDDv1 and v2 images that did not contain range information and tested its performance with and without SAHI on the 8k images with range information.

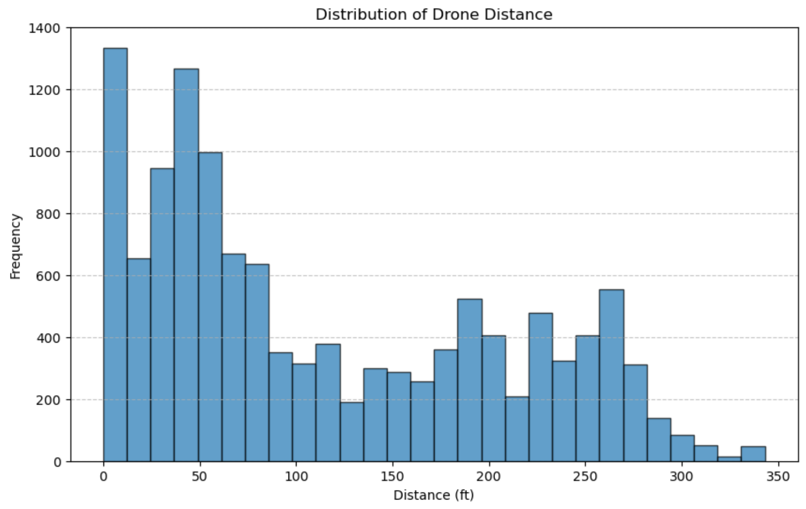


Figure 63. Drone distance distribution of the LRDD v2 dataset.

Figure 57 shows the distribution of drone distances present in the LRDD v2 dataset. This presents an additional difficulty in detection at long range, as both the number of pixels capturing the drone and the number of available samples decrease with distance from the camera. Table 26 shows the detection results across all ranges for both the baseline and SAHI models. Similar to the team's test results on LRDDv1, SAHI improves detection results substantially over the baseline.

Table 26. Detection results for baseline and SAHI models trained on LRDDv1 and v2 and tested on LRDD v2 distance data

Model	AP_{50}	AP_{50-95}
Baseline	0.403	0.609
SAHI	0.456	0.703

Figures 58 and 59 show the detection results across different drone distances for both the baseline and SAHI model implementations. The detection performance decreases sharply for the baseline model, but this trend is mitigated with the implementation of SAHI.

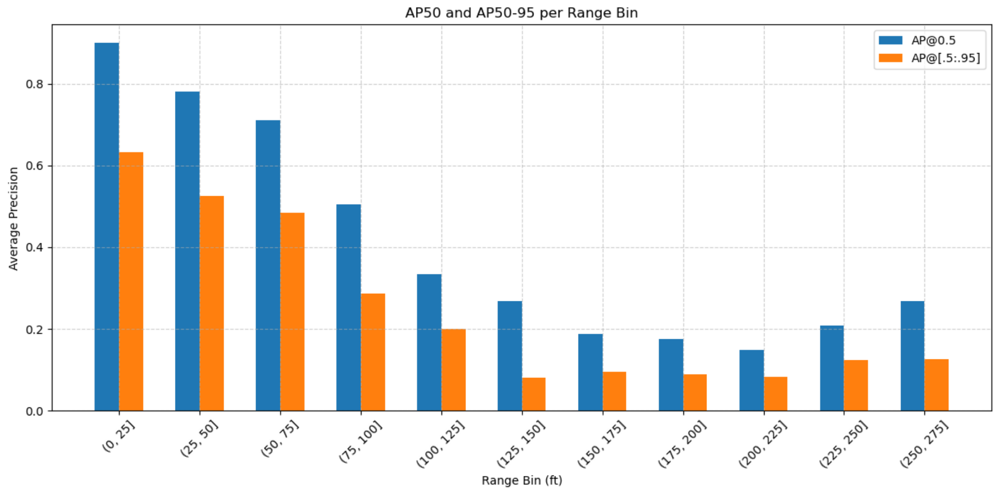


Figure 64. Detection performance of baseline YOLO across different range values in the LRDD v2 dataset.

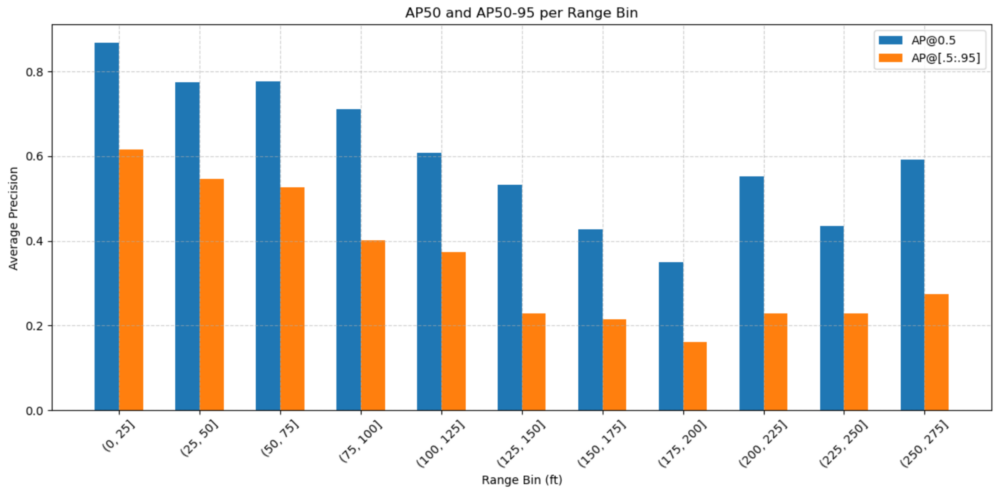


Figure 65. Detection performance of YOLO+SAHI across different range values in the LRDD v2 dataset.

These results show that without implementing SAHI, standard detection methods quickly become inaccurate at distances over 100ft. However, with the implementation of SAHI, detection can remain somewhat accurate at long range but still struggles at longer distances. Some of the drop in performance can be attributed to the lack of data at long-range, which the team anticipates will be somewhat mitigated with the release of version 3 of the team's LRDD dataset, potentially allowing for accurate detections at ranges of up to 300ft. Further improvements could be found by using higher-resolution cameras, which come at the additional costs of weight and processing power.

9.4.3 Attention Maps for Fast Region Proposal

While SAHI is very effective at finding small objects in images, it comes with a major downside with regard to computation requirements. Because SAHI must run an object detector for every individual patch, the inference time scales with the number of patches. This means that for high-resolution images, SAHI can take up to 12-20 times longer than a normal object detector such as YOLO, making it largely infeasible for use on embedded systems such as UAS, where power and weight are highly constrained. Even on larger systems with more computational power, this issue still limits frame rates, hindering capabilities where detection speed is important.

However, there are options for improving long-range detection that can expand on the SAHI concept but are more efficient. Recently, multimodal foundation models such as Contrastive Language-Image Pretraining (CLIP) [22] have become popular due to their flexible use cases and fast inference speed. One use case is to run CLIP to generate attention heat maps [23, 24], effectively marking regions of interest for deeper investigation. As shown in Figure 60, attention maps operate on image patches similar to SAHI; however, instead of running a heavy object detector on each patch, they instead use a much faster contrastive model, such as CLIP, to quickly find similarities between the visual features of the patch and a provided text input, such as “drone”. These patch-text similarities are then merged and smoothed to generate a final attention heatmap for a given text prompt.

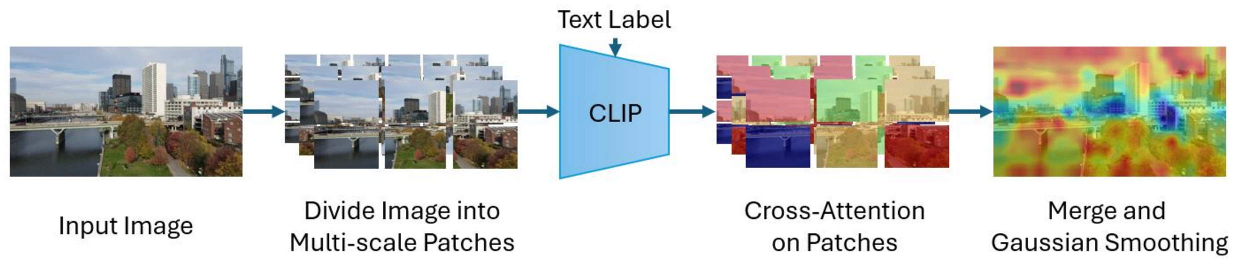

Text Label
CLIP
Input Image
Divide Image into Multi-scale Patches
Cross-Attention on Patches
Merge and Gaussian Smoothing

Figure 66. Overview of the attention map generation method.

When applied to long-range drone detection, attention maps can be used for fast region proposal as an alternative to SAHI by generating a minimal number of patches to use in conjunction with a full object detector such as YOLO. Figure 61 shows an example of this method when applied to the LRDD dataset, where the attention heatmap effectively highlights the drone in the image, allowing for a single patch to quickly be extracted and fed to YOLO for final drone detection.

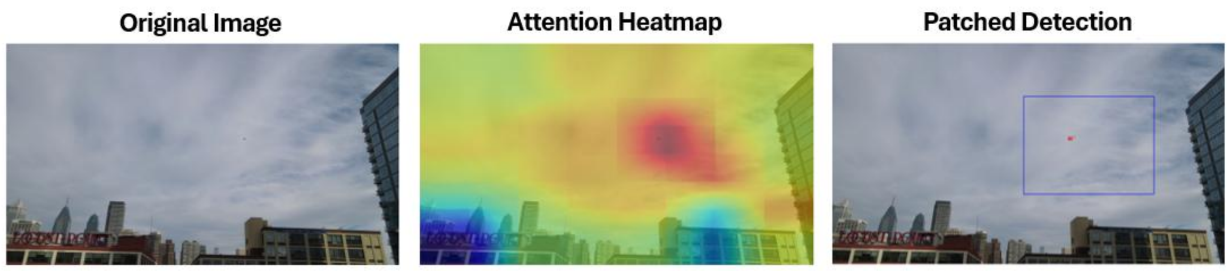


Figure 67. Example application of attention maps for region proposal, quickly generating a single patch to give to YOLO for final detection.

While methods like SAHI are not viable for deployment in lightweight UAS with limited processing power, the use of attention maps may allow for fast patch extraction for full-resolution detection in embedded systems, making cameras much more viable for real world applications. As SAHI can remain effective out to ranges of ~250ft for 1080p video, the team anticipates that patched detection methods, such as attention maps, can become viable for onboard drone detection at long distances.

Beyond detection, the team's work also addressed the broader issue of UAS self-assessment under degraded visual conditions. Just as human drivers may choose to pull over during dense fog when visibility is compromised, autonomous UAS must be capable of evaluating their perception quality and making decisions accordingly. This self-assessment is critical for safe operation in adverse conditions such as heavy rain, smoke, fog, or snow. To support this capability, the team investigated several image quality assessment metrics that correlate with object detection performance and can be applied without requiring reference images. The team identified a set of metrics suitable for onboard use in real-time, enabling autonomous systems to assess the reliability of their visual inputs and adjust their behavior accordingly.

9.5 Effect of image degradation on drone detection and image quality metrics

Accurate self-assessment and prediction of perception performance are critical for safe autonomous UAS operations under adverse environmental conditions. In situations of degraded visibility—such as fog, smoke, or heavy rain—an autonomous UAS must be able to evaluate the quality of its sensory input, assess its detection performance, and determine whether it is safe to continue operation or if it should pause until conditions improve. To support this capability, the team first evaluated the baseline performance of drone detection algorithms under various types of noisy input. The team then investigated a set of existing image quality metrics to assess their potential utility for onboard perception self-assessment in UAS platforms.

9.5.1 Image Quality Metric

To quantify the impact of degradation on visual data and its influence on detection performance, the team applied a set of Image Quality Assessment (IQA) metrics. These are grouped into two categories: traditional metrics and deep-learning-based metrics.

9.5.2 *Traditional Metrics*

Traditional image quality metrics include PSNR and SSIM, both widely used to evaluate the similarity between degraded and reference images.

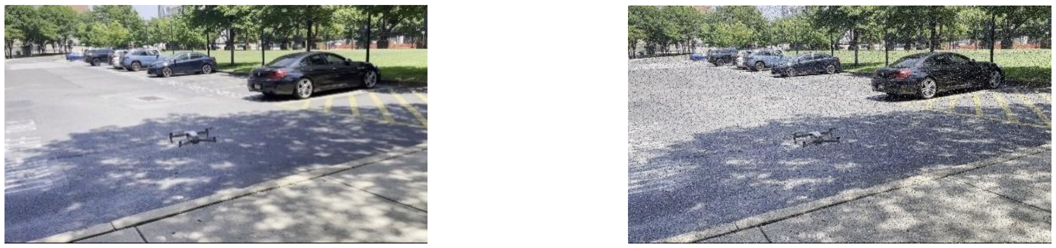
- **PSNR** measures the ratio between the maximum possible power of a signal (image) and the power of corrupting noise. A higher PSNR generally indicates better visual quality.
- **SSIM** evaluates perceived image quality by comparing luminance, contrast, and structure between two images, producing a value between -1 and 1 (higher is better).

To study the effect of image degradation, the team introduced salt and pepper noise to a subset of the LRDD dataset [1]. The corrupted images were then used to evaluate the performance of a YOLOv8 model trained on clean data. Table 27 demonstrates the results.

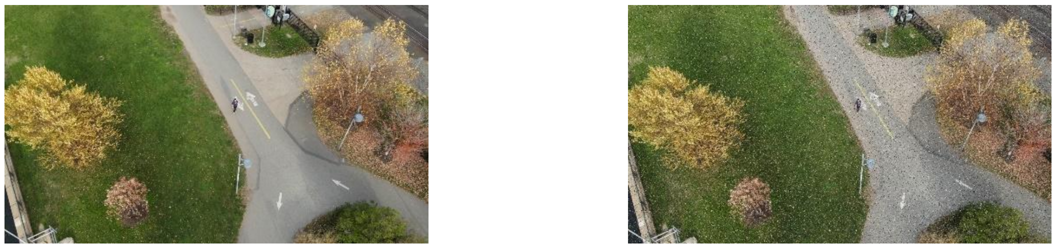
Table 27. Effect of introducing different levels of salt and pepper noise to the LRDD test set on drone detection performance.

SNR	SNR (dB)	AP 50	AP 50- 95	Mean PSNR (dB)	Mean SSIM
Non-degraded set	Non-degraded set	0.9614	0.699	-	-
0.99	30.55	0.394	0.266	20.53	0.5379
0.98	24.66	0.268	0.17	17.6	0.3497
0.97	21.27	0.182	0.114	15.9	0.2572
0.96	18.9	0.1233	0.0761	14.7	0.2045
0.95	17.09	0.06516	0.039	13.8	0.1706

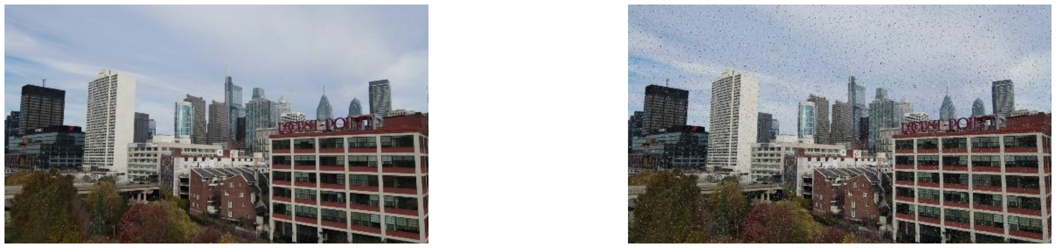
The table demonstrates that as noise increases (i.e., SNR decreases), both image quality and YOLOv8m detection performance degrade significantly. Starting from a high AP50 of 0.9614 on the non-degraded set, performance steadily drops to 0.06516 at the lowest SNR level. This trend is mirrored in the image quality metrics, with PSNR decreasing from 20.53 dB to 13.8 dB and SSIM falling from 0.5379 to 0.1706. The strong correlation between lower image quality and reduced detection accuracy highlights the sensitivity of YOLOv8m to noise and underscores the importance of high-fidelity inputs for reliable object detection. Figure 62 shows some clean and noisy samples with their corresponding SSIM values.



SSIM = 0.177



SSIM = 0.257



SSIM = 0.538

Figure 68. Samples of clean and noisy images with their SSIM value.

9.5.3 Deep-Learning Based Metrics

Traditional IQA metrics—such as PSNR and SSIM—are widely used but rely on the availability of a reference image to quantify degradation levels. In the context of drone self-assessment for perception performance, such reference images are typically unavailable. Moreover, these conventional metrics often fail to capture perceptual degradations or disruptions in high-level semantic features that are critical to the performance of deep learning models. To address these limitations, the team incorporated deep-learning-based no-reference IQA metrics, which do not require a ground truth image and are better aligned with both human visual perception and the sensitivity of downstream tasks such as object detection.

The team specifically examined the following:

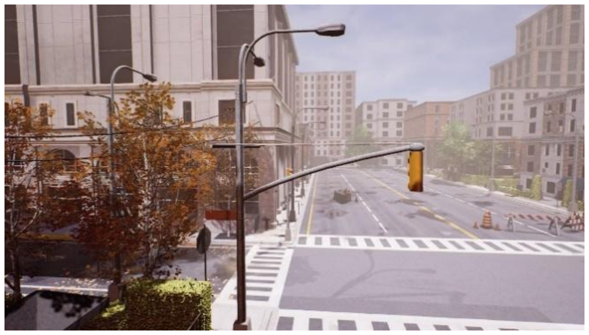
- **BRISQUE (Blind/Referenceless Image Spatial Quality Evaluator)**: A no-reference IQA metric that extracts natural scene statistics in the spatial domain to assess image quality without requiring a reference image. It has demonstrated robust performance in diverse degradation scenarios, making it especially practical for real-world settings where ground-truth images are unavailable [25].

- **DBCNN (Deep Bilinear Convolutional Neural Network)**: A data-driven approach that predicts perceptual quality scores by learning a deep representation of images and comparing them to human opinion scores. Its bilinear structure effectively models complex interactions between features and has shown strong correlation with subjective evaluations of quality, especially for distortions affecting deep network performance [26].

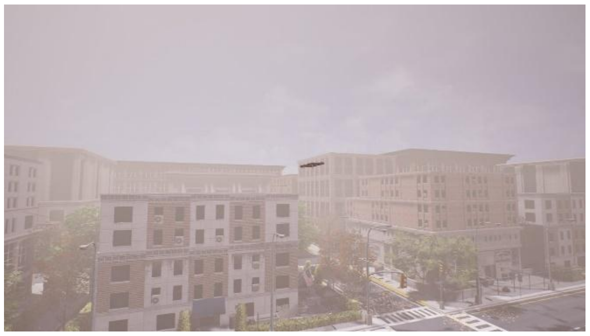
9.6 Adverse Weather Effect

Adverse weather conditions, such as fog and rain, present substantial challenges for vision-based drone detection systems. These effects degrade image quality by introducing visual noise, reducing visibility, and distorting both low-level textures and high-level semantic features, which are critical for reliable object detection.

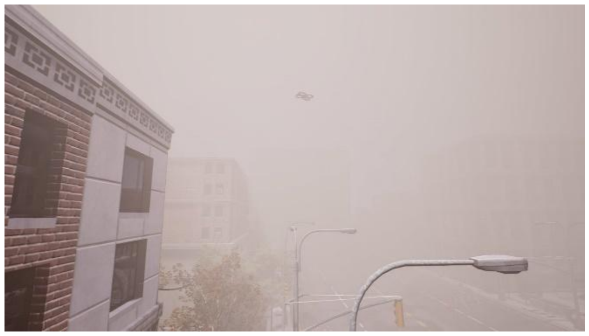
To systematically evaluate the impact of weather-induced degradation, the team modeled fog through controlled simulations in virtual environments rendered with Unreal Engine 4.27 and the AirSim plugin. Each condition was evaluated separately to isolate its effect on image quality and downstream detection performance. The team simulated three fog intensity levels—light, moderate, and heavy—within a photorealistic virtual environment. For each fog level, a separate test set was generated from the same environment. A YOLOv11 [27] model, previously fine-tuned on clear (fog-free) images, was then evaluated on each of these fog-specific test sets to assess the impact of varying fog densities on detection performance. Examples of images from each fog intensity level can be seen in Figure 63.



Light Fog



Moderate Fog



Heavy Fog

Figure 69. Samples of different fog intensity simulated images.

To assess the severity of visual degradation, the team computed both BRISQUE and DBCNN scores for each fog level. Additionally, the detection performance of YOLOv11 was recorded for each set. The results are demonstrated in Table 28.

Table 28. Effect of fog intensity on drone detection and image quality metrics.

Datasets	Weather	mAP50	mAP50-95	#images	BRISQUE	DBCNN
Urban City (Synthetic)	Clear	0.974	0.905	585	13.2802	0.6184
	Light Fog	0.951	0.871	593	10.5788	0.6462
	Moderate Fog	0.938	0.834	423	19.2834	0.6301
	Heavy Fog	0.548	0.412	636	26.1640	0.5743

Detection performance of YOLOv11 consistently declines with increasing fog intensity, with mAP50 dropping from 0.974 under clear conditions to 0.548 in heavy fog. This aligns with the general expectation that adverse weather degrades object detection accuracy. However, the trends in IQA metrics are less straightforward. While BRISQUE scores generally increase with fog, indicating lower perceptual quality, light fog yields a surprisingly lower BRISQUE score than clear images. Similarly, DBCNN scores do not decrease consistently with fog intensity. These inconsistencies may arise because light fog can reduce high-frequency noise and enhance local smoothness, which BRISQUE might interpret as improved quality. Further analysis using a broader set of degradation types, such as rain, snow, and low-light conditions, is needed to better understand and calibrate IQA metrics for autonomous perception assessment.

9.7 Conclusion and Guidance

9.7.1 Required Perception Field of View

As noted in the Year 1 report, the team's study recommends that UAS be equipped with onboard perception systems offering a 4π steradian field of view, enabling full 360-degree coverage of the surrounding 3D space for complete situational awareness. This recommendation parallels the design principles of autonomous ground vehicles, which typically require a 2π steradian (planar) field of view to achieve comprehensive 2D situational awareness. Assuming that detect and avoid maneuvers are performed independently by each UAS, it is critical that these systems can perceive potential threats from any direction in 3D space. Therefore, full 4π steradian sensory coverage is essential for ensuring safe autonomous operation, particularly in high-density airspace where UAS may approach from unpredictable angles.

9.7.2 Drone Detection Training Dataset

Currently available drone image datasets lack the variation and volume needed to effectively train machine learning–based detection algorithms for DAA scenarios. Except for LRDD v2, few—if any—public datasets include target range annotations, which are critical for full situational awareness. There is a pressing need to expand existing datasets to incorporate greater diversity in drone types, lighting conditions, weather variations, drone densities, the presence of other aerial objects, and a wide range of background scenes. For robust perception performance, it is essential that these variations be well-represented to reflect the complexities of real-world operating environments. While the LRDD dataset series addresses some of these gaps, the breadth of variation required for comprehensive training of perception algorithms far exceeds what is currently available. Therefore, the curation of additional datasets—with a clear emphasis on diversity—is strongly recommended. The team's study also highlights the value of augmenting real-world datasets with synthetic data generated through simulation, coupled with advanced generative methods such as GAN models. However, such synthetic data must be generated with care to ensure it meaningfully enhances detection performance and does not introduce artifacts or biases that could impair model reliability.

9.7.3 Drone Detection Algorithm Performance

As demonstrated through the team's implementation of SAHI, drone detection performance can be significantly enhanced by dividing high-resolution images into smaller patches, enabling more accurate identification of small targets. The team's current evaluation shows that applying SAHI can yield detection accuracies exceeding 60% at a range of 125 feet for small drones such as those manufactured by DJI. With ongoing advancements in compact RGB sensor technology, the resolution of cameras onboard UAS is expected to improve substantially in the coming years, further supporting the viability of high-resolution detection strategies.

Moreover, the computational overhead introduced by SAHI can potentially be mitigated through approaches like the one explored in the team's study using CLIP, which enables selective patch processing. This could drastically reduce inference time, paving the way for real-time onboard processing of high-resolution imagery. Although the team's current findings reflect the limitations of existing detection ranges for small UAS, future improvements in sensor resolution and processing efficiency are expected to significantly extend this range. The team strongly recommends continued research into alternative and efficient detection methods for sUAS, particularly those capable of leveraging ultra-high-resolution imagery, such as 8K video streams, for enhanced long-range perception.

9.7.4 Image Quality Metric

The team's study investigated two image quality metrics as potential tools for estimating onboard perception performance in UAS operations. While these metrics demonstrated promising correlations with detection performance, the analysis was limited to simulated fog conditions. Further research is recommended to develop or identify robust and generalizable metrics suitable for UAS self-assessment across a wider range of degraded sensory environments. Such metrics are critical to enabling autonomous systems to determine when conditions are safe for continued operation. Ultimately, a validated self-assessment metric could serve as the foundation for future FAA regulatory guidelines, providing a standardized criterion for UAS operation under reduced visibility or other adverse conditions.

9.8 Team Publications

A list of publications completed as part of this project is presented below:

1. A. Rouhi, H. Umare, S. Patal, R. Kapoor, N. Deshpande, S. Arezoomandan, P. Shah, and D. Han, "Long-range drone detection dataset," in 2024 IEEE International Conference on Consumer Electronics (ICCE). IEEE, 2024, pp. 1–6.
2. A. Rouhi, S. Patel, N. McCarthy, S. Khan, H. Khorsand, K. Lefkowitz, and D. Han. "LRDDv2: Enhanced Long-Range Drone Detection Dataset with Range Information and Comprehensive Real-World Challenges," in 2024 International Symposium of Robotics Research (ISRR). IEEE, 2024, pp. 1-6.
3. Arezoomandan, S., Klohoker, J., Han, D.K. (2025). Data Augmentation Pipeline for Enhanced UAS Surveillance. In: Antonacopoulos, A., Chaudhuri, S., Chellappa, R., Liu, CL., Bhattacharya, S., Pal, U. (eds) Pattern Recognition. ICPR 2024. Lecture Notes in Computer Science, vol 15306. Springer, Cham. https://doi.org/10.1007/978-3-031-78172-8_24.
4. Benyamen, H., Chowdhury, M., & Keshmiri, S. (2024). Data-Driven Aircraft Modeling for Robust Reinforcement Learning Control Synthesis With Flight Test Validation. Journal

of Dynamic Systems, Measurement, and Control, 146(6), 061105. https://doi.org/10.1115/1.4065804
5. Bowes, R., Benjamen, H., & Keshmiri, S. (2023). System Identification-based Fault Detection and Dynamic Inversion Control of an Uncrewed Aerial System. *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, 1286–1293. https://doi.org/10.1109/ICUAS57906.2023.10156557
6. Chowdhury, M., & Keshmiri, S. (2024a). A Unified Inner-Outer Loop Reinforcement Learning Flight Controller for Fixed-Wing Aircraft. *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, 556–563. https://doi.org/10.1109/ICUAS60882.2024.10556883
7. Chowdhury, M., & Keshmiri, S. (2024b). Interchangeable Reinforcement-Learning Flight Controller for Fixed-Wing UAS. *IEEE Transactions on Aerospace and Electronic Systems*, *60*(2), 2305–2318. https://doi.org/10.1109/TAES.2024.3351608
8. H. Khorsand, S. Arezoomandan, D. K. Han, "Enhanced long-range UAS detection: Leveraging slicing aided hyper inference with YOLOv8," in *Proceedings of the International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA, January 11-24, 2025.
9. Kucuksayacigil, G., Keshmiri, S., & Chrit, M. (2025). A Robust Flight Controller Design: Investigating Guidance Failures Near TSS Heliport in Challenging Wind Conditions. 2025 International Conference on Unmanned Aircraft Systems (ICUAS), 1167–1174. https://doi.org/10.1109/ICUAS65942.2025.11007899
10. S. Arezoomandan, J. Klohoker and D. K. Han, "Analyzing the Efficacy of Synthetic Images in Unmanned Aerial Vehicle Detection," 2024 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2024, pp. 1-6, doi: 10.1109/ICCE59016.2024.10444259.

9.9 References

[1] A. Rouhi, H. Umare, S. Patal, R. Kapoor, N. Deshpande, S. Arezoomandan, P. Shah, and D. Han, "Long-range drone detection dataset," in 2024 IEEE International Conference on Consumer Electronics (ICCE). IEEE, 2024, pp. 1–6.

[2] A. Rouhi, S. Patel, N. McCarthy, S. Khan, H. Khorsand, K. Lefkowitz, and D. Han. "LRDDv2: Enhanced Long-Range Drone Detection Dataset with Range Information and Comprehensive Real-World Challenges," in 2024 International Symposium of Robotics Research (ISRR). IEEE, 2024, pp. 1-6.

[3] A. Coluccia, A. Fascista, A. Schumann, L. Sommer, A. Dimou, D. Zarpalas, M. M´endez, D. de la Iglesia, I. Gonz´alez, J.P. Mercier, et al.: Drone vs. bird detection: Deep learning algorithms and results from a grand challenge. Sensors 21(8), 2824 (2021). https://doi.org/10.3390/s21082824, https://doi.org/10.3390/ s21082824

[4] YOLOv8 DetFly 02 Dataset (Mar 2023), https://universe.roboflow.com/ yolov8-drone-detection/yolov8 detfly-02 9. Jiang, N., Wang, K., Peng, X., Yu, X., W

[5] UAS Detect Dataset. https://universe.roboflow.com/get/uav-detect-pfiqs (jan 2023), https://universe.roboflow.com/get/uav-detect-pfiqs, visited on 2023-06-25

[6] Epic Games, Unreal Engine, Version 4.27, Epic Games, 2021. [Software]. Available: https://www.unrealengine.com/

[7] Dassault Systèmes, SolidWorks, Version 2023, Dassault Systèmes, 2023. [Software]. Available: https://www.solidworks.com/

[8] S. Shah, D. Dey, C. Lovett, and A. Kapoor, AirSim: High-fidelity visual and physical simulation for autonomous vehicles, Microsoft Research, 2017. [Software]. Available: https://github.com/microsoft/AirSim

[9] Brian KS Isaac-Medina, Matt Poyser, Daniel Organisciak, Chris G Willcocks, Toby P Breckon, and Hubert PH Shum. Unmanned aerial vehicle visual detection and tracking using deep neural networks: A performance benchmark. In Proceedings of the IEEE/CVF Interna- tional Conference on Computer Vision, pages 1223–1232, 2021.

[10] Ultralytics, YOLOv8: Cutting-edge object detection and segmentation model, Ultralytics, 2023. [Software]. Available: https://github.com/ultralytics/ultralytics

[11] Y. Zheng, Z. Chen, D. Lv, Z. Li, Z. Lan, and S. Zhao, "Air-to-air visual detection of micro-uavs: An experimental evaluation of deep learning," IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 1020–1027, 2021.

[12] GET, "uav detect dataset," https://universe.roboflow.com/get/uavdetect-pfiqs , jan 2023, visited on 2023-10-15. [Online]. Available: https://universe.roboflow.com/get/uav-detect-pfiqs

[13] D. Detection, "Drone detection dataset," https://universe.roboflow.com/drone-detection-7usmm/drone-detectionw29we, May 2023, visited on 2023-10-14. [Online]. Available: https://universe.roboflow.com/drone-detection-7usmm/drone-detectionw29we

[14] S. Arezoomandan, J. Klohoker and D. K. Han, "Analyzing the Efficacy of Synthetic Images in Unmanned Aerial Vehicle Detection," 2024 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2024, pp. 1-6, doi: 10.1109/ICCE59016.2024.10444259.

[15] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 2223–2232.

[16] I. Goodfellow et al., "Generative adversarial nets," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), Dec. 2014, pp. 2672–2680.

[17] Arezoomandan, S., Klohoker, J., Han, D.K. (2025). Data Augmentation Pipeline for Enhanced UAS Surveillance. In: Antonacopoulos, A., Chaudhuri, S., Chellappa, R., Liu, CL., Bhattacharya, S., Pal, U. (eds) Pattern Recognition. ICPR 2024. Lecture Notes in Computer Science, vol 15306. Springer, Cham. https://doi.org/10.1007/978-3-031-78172-8_24.

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 779– 788.

[19] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, “Deformable DETR: Deformable transformers for end-to-end object detection,” in *Proceedings of the 9th International Conference on Learning Representations*, virtual event, May 3-7, 2021.

[20] F. C. Akyon, S. O. Altinuc, and A. Temizel, “Slicing aided hyper inference and fine-tuning for small object detection,” in *Proceedings of the IEEE International Conference on Image Processing*, Bordeaux, France, October 16-19, 2022, pp. 966-970.

[21] H. Khorsand, S. Arezoomandan, D. K. Han, “Enhanced long-range UAS detection: Leveraging slicing aided hyper inference with YOLOv8,” in *Proceedings of the International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA, January 11-24, 2025.

[22] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, “Learning transferable visual models from natural language supervision,” in *Proceedings of the 38th International Conference on Machine Learning, PMLR*, virtual event, July 18-24, 2021, vol. 139, pp. 8748-8763.

[23] T. Lüddecke, and A. Ecker, “Image segmentation using text and image prompts,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, New Orleans, LA, USA, June 19-24, 2022, pp. 7086-7096.

[24] F. Wang, J. Mei, and A. Yuille, “SCLIP: Rethinking self-attention for dense vision-language inference,” *arXiv preprint, arXiv:2312.01597v4*, 2023.

[25] A. Mittal, A. K. Moorthy and A. C. Bovik, "No-Reference Image Quality Assessment in the Spatial Domain," in IEEE Transactions on Image Processing, vol. 21, no. 12, pp. 4695-4708, Dec. 2012, doi: 10.1109/TIP.2012.2214050.

[26] W. Zhang, K. Ma, J. Yan, D. Deng and Z. Wang, "Blind Image Quality Assessment Using a Deep Bilinear Convolutional Neural Network," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 30, no. 1, pp. 36-47, Jan. 2020, doi: 10.1109/TCSVT.2018.2886771.

[27] G. Jocher and J. Qiu, Ultralytics YOLOv11, Version 11.0.0, 2024. [Online]. Available: https://github.com/ultralytics/ultralytics

10 CYBERSECURITY AND CYBER-PHYSICAL SECURITY

Cybersecurity is a constant back-and-forth of new attacks and defenses, and it can be difficult to keep up with the latest developments. In the near future, attacks will likely become more common, and proper measures against them will be needed as UAS are increasingly used, particularly for safety-critical applications. Despite this shifting landscape, existing UAS engineering practices, standards, and tools remain largely rooted in traditional safety and reliability paradigms that do not account for adversarial behaviors.

Specifying failures resulting from security attacks can be tricky because a single attack can potentially cause many failures and have different levels of success and outcomes depending on attacker strength or defenses in place. These failures can include crashes, triggering of a safety mode that forces drones to land, attacker control or takeover of drones, disruption and breaking apart of a swarm, and exfiltration of data which may later be used for attacks. So, there's a need for an updated overview of widely feasible attacks and correspondingly widely available defenses.

Autonomous UAS swarms are emerging as a potentially useful technology in many domains such as UAM, agriculture, forestry, delivery, surveying, warehouse automation, and structural maintenance. Decentralized swarms, aka d-swarms, are a type of swarm where each drone acts more individually, providing advantages for scaling and adaptability as well as some natural path-finding ability in the presence of obstacles. It is important to investigate how to create robust d-swarms. Collision avoidance and path planning methods exist, but they have limitations, especially during an attack, due to increased computation and information demands. Flocking-based swarms with low-latency collision avoidance can avoid some of these limitations and may be useful as a fallback mode of operation, but this appears to be under-investigated. Possible reasons may be their difficulty of tuning, decreased precision and efficiency, and that attacks are often not a primary concern. An issue hindering investigation is that there is limited open-source code or recipes for tuning large, realistic d-swarms. The team developed guidance for d-swarms in this report.

As UAS swarms are a newer technology, there is insufficient work regarding the safe and secure usage of swarms. Decentralized UAS swarms will typically rely on path planning and Collision Avoidance (CA) methods to ensure efficient, coordinated, and collision-free flight [3-8]. These methods are susceptible to attacks such as jamming and spoofing, and they can take seconds to compute in complex environments. To bridge the gap and provide guidance, the team investigated how to develop and evaluate swarms that are resilient against attacks and avoid collisions.

10.1 EXECTUIVE SUMMARY

Updated threat and defense assessment: The team updated the threat assessment of attacks based on the severity of outcomes and likelihood. Based on this, the team recommends a practical set of minimal defenses with large coverage that most practitioners should consider using (e.g., sensor fusion, fuzz testing, safety mode). On top of this, depending on the use case, the team provides recommendations for various levels of additional defenses.

Recommended Guidance for defense assessment: Based on the team's updated assessment of widely available attacks, the team recommends the use of the following defenses that provide wide coverage of common attacks: sensor fusion and redundancy, a safety mode, rapid message validation, software testing and requirement validation, path planning and navigation algorithms, adversarial training for computer vision algorithms, and delay handling. However, the team highly recommends that designers conduct a comprehensive threat and vulnerability assessment tailored to the specific UAS operational context as a foundational step for more efficient and effective selection and deployment of defenses.

Safety and Robustness of UAS swarms: The team developed open-source code for tuning flocking distributed swarms for rotary UAS. The team developed methods and a recipe for robust tuning and evaluation, including a novel attack, focusing on low-latency avoidance and the Vicsek algorithm. Lastly, the team identified limitations and areas for future work as well as guidance for when and how low-latency tuned d-swarms should be used in their current state.

Recommended guidance for swarms: While path planning and collision avoidance methods are recommended for many use cases, no matter what methods are chosen, the limits of these should be identified, and fallback modes of operation should be used in the case of attacks or challenging scenarios. The team recommends a hierarchical fallback mode structure. The first level is a simple fallback mode, like hovering or landing. In cases where these are not appropriate (such as unavailability of a good landing spot for a large number of drones), the team recommends that different settings of swarm parameters be stored, to choose the best adapted to the current flight situation, thus allowing minimal continued operations while maintaining a tolerable safety standard. The next level is to use a slow swarm velocity or hybrid flocking-pathfinding methods. Finally, and as a last resort, it is worth exploring the use of UAS tolerate some light collisions or bumps.

10.2 Introduction

The team find it useful to first summarize the outcomes of the first two years of this project to place the team's work in context.

*10.2.1 **Task 1 Overview: Identification of Security Failures***

Task 1 involved a comprehensive literature review and structured interviews with SMEs to catalog failure modes across multiple components of autonomous UAS. From a security standpoint, several recurrent failure types were identified:

- *Spoofing Attacks*: Falsified signals were shown to compromise GPS, Automatic Dependent Surveillance – Broadcast (ADS-B), Remote ID, and inertial sensors, misleading the UAS about its position or operational environment.
- *Jamming Attacks*: Denial-of-service tactics were found to disable or degrade GPS, communication links, and other sensors through signal saturation or acoustic interference.
- *Software-level Compromise*: It was found that insecure interfaces between system components could allow malicious alteration of sensor data or control commands.
- *Adversarial Sensor Manipulation*: Physical-layer attacks on MEMS-based IMUs and vision systems were shown to produce subtle but unsafe control behaviors.
- *Systemic Design Weaknesses*: Many systems were found to lack redundancy, fail-safe fallback mechanisms, or authenticated data paths, increasing susceptibility to single points of failure.

These findings indicated that most current UAS designs presume benign environments and do not incorporate adversarial models into the system engineering process. Security-relevant failures were also underrepresented in incident reporting databases and simulation-based evaluations.

*10.2.2 **Task 2 Overview: Gaps in Existing Design and Evaluation Frameworks***

In Task 2, the research team assessed whether existing engineering practices, standards, and tools could have mitigated the failures identified in Task 1. It was concluded that:

- Security was rarely embedded into the early design stages of UAS automation, resulting in reactive or fragmented countermeasures.
- Probabilistic Risk Assessment frameworks, though useful for modeling stochastic failures, did not account for intelligent adversaries or coordinated multi-vector attacks.
- Verification and validation tools lacked adversarial realism, making them insufficient to test the efficacy of cyber-physical security mechanisms.
- Guidance documents and standards were either missing for specific threat classes (e.g., sensor spoofing) or too generic to address the nuanced challenges of autonomous UAS.

These gaps highlighted a pressing need for structured design guidance and best engineering practices that explicitly integrate cybersecurity into autonomous UAS design. Such guidance must also be tailored to the operational risk level and tested under realistic conditions to ensure feasibility and effectiveness.

10.3 **Updated List of Broad Defenses**

As Task 1 and Task 2 highlight, there are numerous possible attacks and defenses, and it may be impractical to consider all of them. Many attacks are difficult to achieve, while many defenses have significant overhead or complexity. In this guidance, the security landscape is simplified to attacks that are most likely (baseline threats) and defenses that are practical but provide broad mitigations against those attacks. Table 29 shows a summary of the attacks and how the defense covers them.

*10.3.1 **Baseline Threats:***

- **GPS Jamming or Spoofing:** Prevents GPS signal acquisition or overrides authentic GPS signals.
- **Wi-Fi and Message Jamming or Denial of Service (DoS):** Prevents communication.
- **Message Spoofing, Fake drones (e.g., ADS-B and Remote ID):** Creates fake drones or false drone positions.
- **Adversarial Attacks on Computer Vision:** Adversarial patches or imagery to cause misclassifications in visual perception. This category also includes direct energy attacks, such as laser strikes intended to temporarily blind or permanently damage camera sensors.
- **Adversarial Drones:** Use of drones to cause interference, such as collision or path manipulation.
- **General Software, Design, and Requirement Threats:** Bugs, vulnerabilities, and oversights in software design and implementation. This can include third-party software.
- **Laser Pointers:** Lasers aimed at cockpits of large UAS, such as electric Vertical Take-Off and Landing (eVTOL) vehicles with human passengers, can cause eye damage.

- **Vertiport Denial:** Denial: Congestion (traffic backups) or electronic interference at Vertiports can prevent or delay landing, which is problematic for aircraft with limited battery life.

10.3.2 Recommended Baseline Defenses:

- **Sensor Fusion and Redundancy:** A broad defense that combines readings from redundant and complementary sensors to determine the current state. Kalman Filters are commonly used already in state estimation, but without a specific focus on security concerns. They are typically meant to reduce the impact of faulty or noisy sensors. However, newer approaches that incorporate cross-sensor validation to detect anomalous readings before they are fused are emerging. If a sensor is identified as compromised (e.g., via an anomaly detection system), it should be temporarily or permanently excluded (blacklisted) from the fusion process to prevent the corruption of the state estimate. This guidance is about taking such malicious sensor manipulation into account when designing sensor fusion and redundancies.
- **Safety Mode:** Fallback mode of operation if the UAS is under severe attack and cannot operate normally. May include hovering, landing, or simplified navigation algorithms. These typically exist but may not account for adversarial inducement of such modes.
- **Rapid Message Validation:** Special message format and processing to quickly determine validity and mitigate DoS and fake drone information. This validation may include cryptographic or other authentication techniques.
- **Software Testing and Requirement Validation:** Fuzz testing, unit testing, and supply chain provenance for third-party software to prevent bugs and software attacks. In particular, algorithms that the software implements should also be fuzz tested, i.e., semantic fuzzing of drone behavior.
- **Path Planning and Navigation Algorithms:** Proper design and testing of path planning to avoid collisions and adapt to scenarios such as adversarial drones.
- **Adversarial Training for Computer Vision:** Mitigate the effect of adversarial imagery on misclassifications. In general, any ML models used should be evaluated for their susceptibility to adversarial manipulations and mitigations (like adversarial training) applied.
- **Laser Filters:** Cockpit windows or provided glasses may include laser filters to prevent health risks to passengers.
- **Delay Handling:** Delays should be taken into consideration when estimating the required battery life. Rerouting should be expected, and efficient loitering algorithms may be beneficial.

10.3.3 Ideal Additional Defenses

- **Anomaly Detection:** Implement a detection capability to detect deviations from normal operations, especially those induced by attackers. For instance, detecting spoofing or compromised software so that actions can be taken, such as sensor fusion or safety mode.

Additionally, the identification of the threat landscape aids in creating more secure systems in the future.

- **Forensic Replay:** Using data recorders to enable the forensic replay of incidents. This allows teams to identify causes of bugs or failures, reconstruct events for anomaly investigation, improve future designs, and better characterize the threat landscape..
- **Jamming Resistant Sensors:** Advanced sensors and algorithms that make use of beamforming or directional information can mitigate jamming if it is a likely threat.
- **Shielded Hardware:** Some spoofing and jamming attacks can involve acoustic or electromagnetic interference, and proper shielding can mitigate them if expected.

Table 29**.** Recommended Defenses and Coverage.

Attack	**Defense**
GPS and Sensor Jamming	Sensor Fusion and Redundancy, Safety Mode
Message Spoofing and DoS	Sensor Fusion and Redundancy, Safety Mode, Rapid Message Validation
Computer Vision Adversarial Attack	Sensor Fusion and Redundancy, Safety Mode, Computer Vision Adversarial Training
Adversarial Drones	Safety Mode, Path Planning Algorithms
Software and Requirement Attacks or Bugs	Sensor Fusion and Redundancy, Safety Mode, Software Testing and Requirement Validation
Laser Pointers	Laser Filters
Vertiport Denial	Delay Handling

10.4 Swarm Guidance

Decentralized UAS swarms, or d-swarms for short, can efficiently accomplish tasks that individual drones cannot by working cooperatively and covering larger areas. Swarms are emerging as a useful technology in domains such as agriculture, forestry, delivery, surveying, warehouse automation, and structural maintenance. Unexpected turbulence, moving obstacles, or attacks like jamming [1-2] can severely disrupt the performance of swarms in safety-critical scenarios, like near-people operation.

As UAS swarms are a newer technology, there is insufficient work regarding the safe and secure usage of swarms. Decentralized UAS swarms will typically rely on path planning and CA methods to ensure efficient, coordinated, and collision-free flight [3-8]. These methods are susceptible to attacks such as jamming and spoofing, and they can take seconds to compute in complex environments. To bridge the gap and provide guidance, the team investigated how to develop and evaluate swarms that are resilient against attacks and avoid collisions.

10.4.1 Introduction

Three weak points are identified that can cause path planning algorithms to fail in safety-critical scenarios. First, in unexpected scenarios with high obstacle density, high turbulence, or moving obstacles, the planning methods may not be able to calculate trajectories fast enough to react. Second, in jamming attacks, communication that may be required to coordinate plans is blocked [1-2]. Lastly, the information needed to calculate trajectories is susceptible to sensor and message spoofing [9-11], which can lead to incorrect plans. This information may include the positions, velocities, orientations, and plans of other drones.

To handle the weak points, a fallback mode of operation can be used when problematic scenarios are detected. Hovering in place or landing are commonly used fallbacks [9], but these halt operations, which may be undesirable. It should be noted that these specific fallback maneuvers are most applicable to multirotor UAS that can hover; other aircraft types, such as fixed-wing UAS, would require different fallback strategies. Instead, the guidance proposes that flocking d-swarms can be used as a fallback mode that enables continued operation with tolerable degradation in performance. At their core, flocking d-swarms do not have explicit path planning or CA, and the drones rely on simple dynamics equations to calculate repulsive and cohesive forces to determine their next command [12]. This allows for swarms to be designed with reduced computation, sensor information, and communication requirements, covering the weak points of planning methods. Despite their simplicity, flocking swarms can adapt rapidly and have some natural path-finding ability in the presence of obstacles, such that they are still capable of handling complex scenarios.

Unfortunately, there is a lack of existing methods for creating reliable and efficient flocking d-swarms, particularly while taking attacks into consideration. As part of validation, these methods are developed. To start, an attack-resilient and scalable explicit CA is added to a Vásárhelyi flocking swarm algorithm [16] to improve anti-collision guarantees. This is a simple, low-latency CA mechanism that is based on distance thresholds. However, the CA increases the nonlinearity of the swarm and the potential for deadlocks, which creates optimization challenges for tuning the swarm parameters. To successfully tune the swarm, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm is used with carefully designed fitness score functions and multiple tuning stages. The optimization and experiments are performed on SwarmLab [13], a realistic simulator. The resulting swarms complete all of the test maps without collisions, including ones not seen during tuning.

To test and improve the resilience of the proposed swarm, an attack against the swarm algorithm is developed. The attack spoofs a target drone to move in an oscillating pattern to exploit blind spots in the CA for increased collision risk. Furthermore, the attack takes advantage of the CA to trigger deadlocks, which causes the swarm to get stuck and have increased flight time and energy consumption. When the attack is applied, collisions are consistently experienced on all maps. Inspired by adversarial training in computer vision and natural language processing [14-15], where models are trained against attacks to achieve resilience while maintaining performance in the general case, adversarial tuning is performed with the attack. The results show that adversarial tuning mitigates almost all collisions while only minor performance loss occurs in the non-attack setting.

Overall, the results validate the guidance for the usage of flocking swarms with CA as a fallback mode to safely enable continued operation in challenging scenarios where preferred methods like path planning may fail.

10.4.2 Tuning Approach

The goal of the optimization is to tune the swarm parameters so that the swarm can complete the given maps while maintaining desirable swarm behavior. Swarm behavior is judged by map completion time, visually, and by the fitness score. The fitness score is a weighted combination of scores like collisions, deadlocks, and cohesion.

The SwarmLab simulator [13] is used to test the swarm on various maps. SwarmLab is an open-source, realistic simulator for UAS swarms. Additionally, it provides a reference implementation of a Vásárhelyi swarm, which has its parameters initialized similarly to the results of [16], which developed the algorithm and validated it in the real world.

An overview of the optimization setup is shown in Figure 64. The CMA-ES algorithm is used to tune the swarm, specifically the python library from [17]. CMA-ES is an evolutionary algorithm well-suited for this type of nonlinear black box optimization. A population size of 40 and a step size of 0.1 are used.

The tuning process starts by normalizing the initial swarm parameter values to [0,1] to initialize the CMA-ES sampler. The sampler returns a population of parameters, which are then denormalized and swapped into the swarm algorithm to run simulations to calculate fitness scores. The scores of the top results are passed back to the CMA-ES optimizer so it can update parameter estimates for the next iteration of sampling. The fitness scores used are described in the following sections.

To improve generalization and avoid overfitting, the final fitness score is averaged from running the swarm on three different maps. The maps are shown in Figure 65. Moreover, on each map, the worst-case score out of five random seeds is chosen. The seeds randomize the initial starting positions of the drones within a small area. Furthermore, random noise is added to the position and velocity of the drones as regularization and to mimic environmental or sensor noise.

At the end of tuning, swarms from the top 20 scores are further tested to select the final result. They are run on two unseen maps in addition to the three maps used in tuning. Seven seeds are used, including six guaranteed unseen seeds and a specific seed that is identified to score poorly. As before, the worst-case scores of the maps are averaged to get the final score.

The design constants are shown in Table 30, the tuning settings in Table 31, and the initial parameters and ranges are shown in Table 32. These are largely based on the work of [16] for swarms of 15 drones with 6 m/s velocity. Some minor modifications were made due to the map changes and robustness focus.

Each iteration of CMA-ES requires running *seeds x maps x population* simulations. To optimize efficiently, Message Passing Interface is used to parallelize calls to the simulation within a population. Using 60 Intel Xeon Platinum 8480 CPUs, it takes about 1.5 days to tune the three stages.

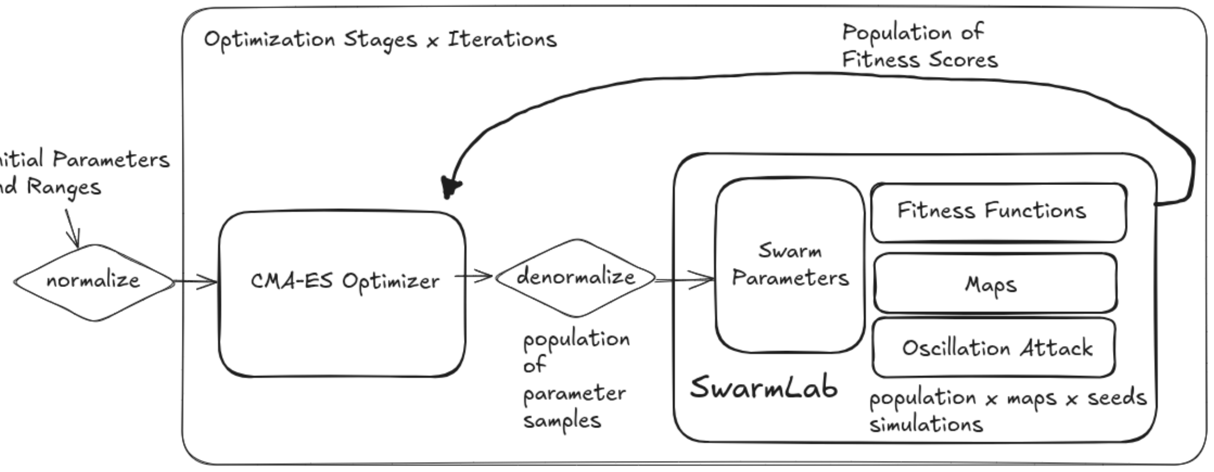

Optimization Stages x Iterations
Population of Fitness Scores
Initial Parameters and Ranges
normalize
CMA-ES Optimizer
denormalize
population of parameter samples
Swarm Parameters
Fitness Functions
Maps
Oscillation Attack
SwarmLab
population x maps x seeds simulations

Figure 70. Swarm Tuning Overview.

Table 30. Swarm Design Constants. The * indicates multiple values were used in experiments, so a representative value is given.

Parameter	Value	Description
n_agent	15	Drones in swarm
drone_mass	0.3 kg	Drone mass
r_col	*0.5 m	Collision radius of drone
max_a	10 m/s²	Max allowed acceleration, caps velocity command
max_v	6 m/s	Max allowed velocity, caps velocity command
v_ref	*6 m/s	Velocity component towards target
r_swarm	120 m	Desired max swarm size
neigh_r	80 m	Max distance of neighbors to use in calculations
max_neigh	10	Max neighbors to use in calculations
n_clust	4	Desired max number disjoint drone clusters
soft_dist	4.5 m	Desired minimum inter-drone distance on top of CA
dl_count	5	Number of time steps for deadlock detection
dl_micro	0.2 m/s	Size of random movement for deadlock avoidance
attack_type	0.5	Probability for each attack variant
attack_freq	*1.5 s	Time between switching spoof direction for attack
attack_dev	5 m	Size of spoofing distance for attack
env_noise	*1.5 %	Max % of velocity added as noise to velocity command
data_noise	*1.5 %	Max % of velocity, position added to drone values during calculations
dt	0.01 s	Size of simulator time step
end_time	*90 s	Cutoff in seconds to end simulation
goal_threshold	60 m	End sim early if all drones within distance to goal
stable_time	*1.5 s	Time before applying scoring so initial drone positions stabilize

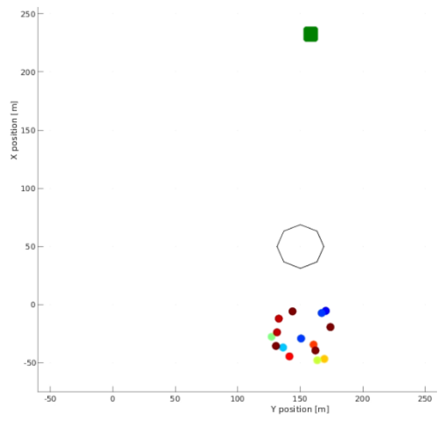
Table 31. Tuning and Evaluation Settings. Tuning and parameter selection settings for the 3 stages are described along with the final evaluation settings. The time outs (*end_time*) align with *v_ref* values. For tuning, population size is 40 and step size is 0.1.

	Tuning Iterations	*end_time*	*v_ref*	*stable_time*	*r_col*	*CA_dist*	*CA_intersect*	*data_noise env_noise*	*attack_freq*
Stage 1	85	90	6	1.5	0.7	learned	learned	1.5 %	0
Stage 2	30	90	6	1.5	0.7	0.7	-0.5	1.5 %	0

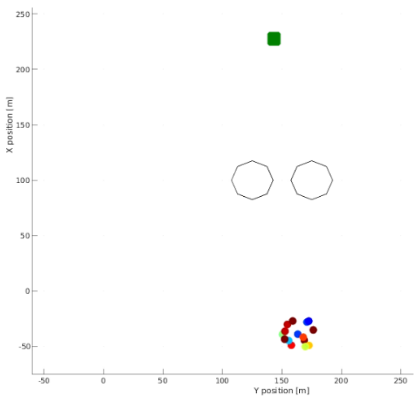
Stage 3	65	150	5	1.5	0.7	0.7	-0.5	1.5 %	1.5
Evaluation	-	90, 150	6, 5	4	0.5	0.7, 1	-0.1	1.0 %	0, 1.2

Table 32**.** Vásárhelyi Swarm Parameters. The * indicates the values were fixed. For the initial set, the ˆ indicates the parameters are new and were set with a couple trial and error runs. The Stage 3 ranges are taken during parameter selection from the top 20 parameters. Table S3 in [16] describes the effect of each parameter.

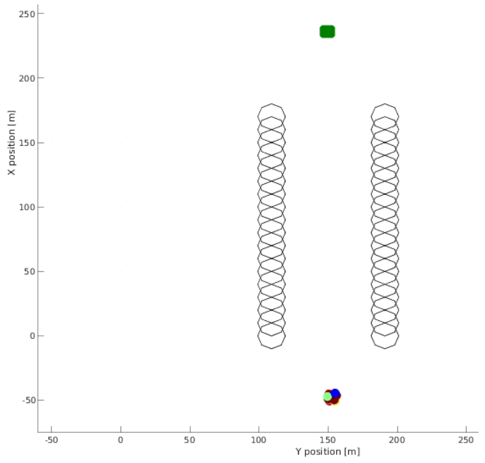
Parameter	Bounds	Initial	Stage 1	Stage 2	Stage 3	Stage 3 Ranges
p_rep	[0.001, 5.0]	0.03	0.65	0.72	0.87	[0.7099, 0.8747]
p_coh	[0.001, 5.0]	0.03	0.21	0.03	0.018	[0.0014, 0.0242]
r0_rep	[0.001, 100.0]	25	10.37	14.09	20.55	[18.5005, 21.3524]
r0_fric	[0.001, 200.0]	85.3	82.58	92.67	96.1	[91.8145, 97.5842]
C_fric	[0.001, 5.0]	0.05	0.02	0.12	0.13	[0.0232, 0.1315]
v_fric	[0.001, 10.0]	0.63	2.78	3.29	3.58	[3.3702, 3.5846]
p_fric	[0.001, 20.0]	3.2	2.71	3.31	3.21	[3.059, 3.4023]
a_fric	[0.001, 20.0]	4.16	4.52	4.74	5.07	[4.9174, 5.4394]
r0_shill	[0.001, 30.0]	0.3	1.5	1.24	0.16	[0.0798, 1.0468]
v_shill	[0.001, 100.0]	13.6	11.56	13.6	14.22	[13.2303, 14.5150]
p_shill	[0.001, 20.0]	3.55	2.98	2.54	1.82	[1.5481, 2.4057]
a_shill	[0.001, 20.0]	3.02	7.8	8.26	7.73	[7.2683, 8.1493]
vel_obs_scale	[0.001, 5.0]	2^	1.77	1.94	2.21	[2.0221, 2.2572]
CA_dist	[0.001, 10.0]	2.5^	0.58	0.7*	0.7*	-
CA_intersect	[-1.0, 1.0]	-0.25^	-0.56	-0.5*	-0.5*	-
Fitness Score						[0.1287, 0.1681]



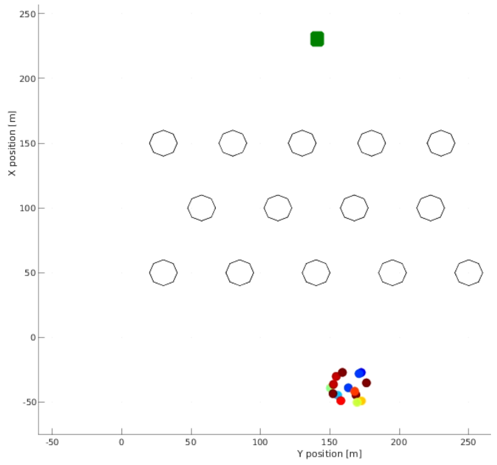
Single



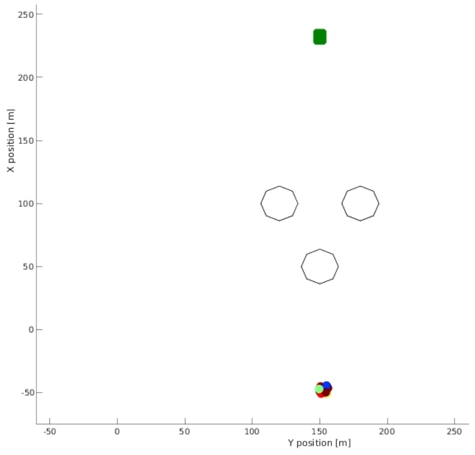
Bottleneck



Channel



Forest



Triple

Figure 71. Maps. The green square is the goal position. Start position is randomized within a small area. Single, Channel, and Bottleneck are used in tuning. All of the maps are used in evaluation and best parameter selection

10.4.3 Collision Avoidance Mechanism

A simple, attack resilient CA mechanism is added to improve the anticollision guarantees of the swarm. If a drone's calculated velocity vector intersects with another object, and the object is within a threshold distance, then the velocity vector is set to 0. To determine what counts as an intersection, another distance threshold is used to check if an object is within a certain distance of the velocity vector. The thresholds are added to the swarm algorithm as new parameters.

The CA increases the possibility of deadlocks in the swarm, particularly if its thresholds have high values. To handle light deadlocks, small random velocity perturbations are introduced if a drone is detected to have 0 velocity for 5 time steps in a row. While this approach is effective for minor deadlock situations, swarms should be optimized to avoid deadlocks. More advanced deadlock avoidance maneuvers are out of scope for current validation.

To verify the CA effectiveness, the SwarmFuzz attack is applied to a 15-drone Vásárhelyi swarm by running 100 random seed initializations on a single obstacle map, closely following the original attack [18]. Using CA, there are 0 obstacle collisions compared to 42 seeds with obstacle collisions while not using CA.

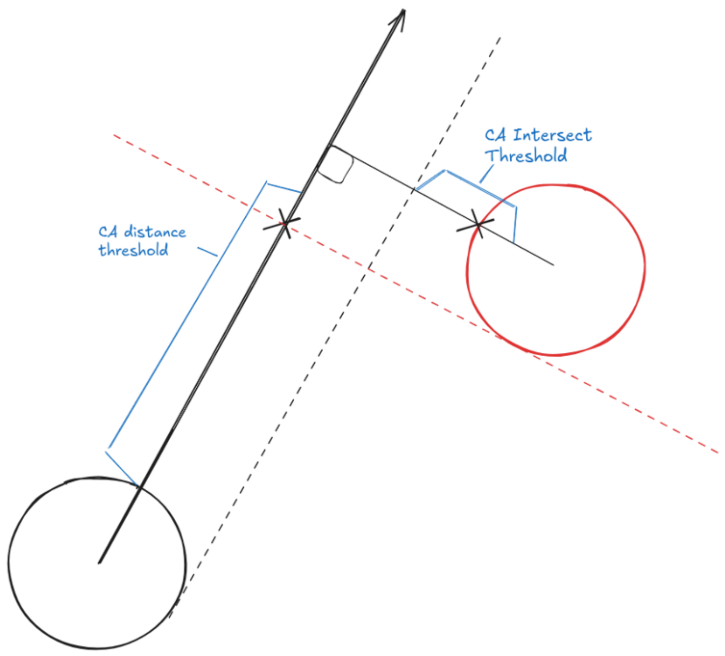


Figure 72. Collision Avoidance Mechanism. The red drone is within a CA Intersection threshold distance of the black drone's velocity vector. The intersection with the velocity vector is also within the CA Distance threshold. In this case the CA triggers for the black drone, setting its velocity vector to 0.

10.4.4 Fitness Functions

To optimize a swarm, a fitness score that captures the desired characteristics is needed. Several scores are combined and weighted as shown in Table 33. Weights can be arbitrary and may require trial and error, but with analysis of swarm behaviors and the convergence process, a successful weighting can be found. The following subsections describe the scores used.

10.4.4.1 Collisions:

cols is the total collisions encountered in a simulation. A single collision will result in 0.9, and additional collisions will follow a square root curve starting from that point, saturating to 1 after 910 collisions. Obstacle and drone collisions are treated equally.

$$score = \begin{cases} 0 & cols = 0 \\ 0.9 & cols = 1 \\ min(1, 0.9 + (\frac{-0.01+0.011*cols}{1000})^{1/2}) & cols > 1 \end{cases}$$

10.4.4.2 Deadlocks:

dl is the fraction of time steps in a simulation that have any deadlocks. This is piecewise linear score, where the first part rapidly increases to 0.9 as 5% of time steps have deadlocks. The second part slowly saturates to 1, where 95% of time steps have deadlocks.

$$score = \begin{cases} min(1, \frac{0.9}{0.05} * dl) & dl \leq 0.05 \\ min(1, 0.9 + \frac{0.1}{0.95} * dl) & dl > 0.05 \end{cases}$$

10.4.4.3 Inter-drone Distance:

soft_dist is the minimum desired inter-drone distance on top of the CA distance threshold *CA_dist*. *min_dist* is the smallest inter-drone distance of each drone at each time step, averaged across all drones and time steps. The score increases by a square root curve as *min_dist* approaches *CA_dist.*

$$score = \begin{cases} 0 & min\_dist \geq (soft\_dist + CA\_dist) \\ 1 & min\_dist \leq CA\_dist \\ min(1, (\frac{soft\_dist-(min\_dist-CA\_dist)}{soft\_dist})^{1/2}) & CA\_dist < min\_dist < (soft\_dist + CA\_dist) \end{cases}$$

10.4.4.4 Distance-to-goal:

goal_dist is the average distance of a drone to *goal_thresh* at the end of the simulation. The score normalizes this by the map length and applies a square root curve. The maps used all have the same length and similar start and goal locations.

$$score = min(1, (\frac{goal\_dist}{map\_length})^{1/2})$$

10.4.4.5 Velocity:

vel is the difference between the velocity of a drone and *v_ref*, averaged across all drones and time steps. The score is linear and normalizes the range to [0,1].

$$score = min(1, \frac{vel}{v\_ref})$$

10.4.4.6 Cohesion:
coh is the area of the square that contains the swarm, averaged across time steps. The maximum and minimum desired areas are needed to normalize the score. *min_sz* is an estimate of the minimum desired area, found by packing all drones into a square while using the desired minimum inter-drone distance *soft_dist*. *max_sz* gets the area difference between the maximum desired area and *min_sz*, reshapes the area difference as a square, and gets the length. *coh_diff* similarly compares the area difference between *coh* and *min_sz* to get the length. Finally, the score normalizes *coh* and applies an exponent curve.

$$min\_sz = (n\_agent^{1/2} * (2 * r\_col + soft\_dist) - soft\_dist)^2$$

$$max\_sz = (r\_swarm^2 - min\_sz)^{1/2}$$

$$coh\_diff = (abs(coh - min\_sz))^{1/2}$$

$$score = min(1, (\frac{coh\_diff}{max\_sz})^4)$$

10.4.4.7 Clusters:
clust is the number of disconnected clusters of drones, averaged across time steps. There is no penalty until beyond the maximum desired number of clusters *n_clust*. The score takes the difference from *n_clust* and normalizes. Clusters are found using a union-find algorithm based on [19]. The algorithm input is an adjacency matrix of the swarm, where edges between drones are created if their distance is within an arbitrary distance *r_clust*. The team chose *r_clust* to be *2 * soft_dist*.

$$score = min(1, max(0, \frac{clust - n\_clust}{n\_agent - n\_clust}))$$

10.4.4.8 Alignment:
This score measures how aligned the velocity vectors of the drones are. Alignment is measured within each cluster of the swarm and then averaged across clusters and time steps, resulting in *align*. Within a cluster, the signs of the dot products between drones are averaged. If the average is below an arbitrary 0.3, then the cluster is considered unaligned. Alignment is binary.

$$score = min(1, align)$$

10.4.5 Tuning Stages

Three tuning stages are used to arrive at the final swarm parameters. The first stage is focused on completing the maps while avoiding collisions. Fitness scores are simplified to improve the ease of optimization. The second stage focuses on improving swarm behavior and robustness to noisy behavior. This is primarily accomplished by adding the inter-drone distance score. The last stage applies adversarial tuning to further improve robustness and resilience against swarm algorithm attacks. The attack is described in the following section. The different tuning settings between the stages are shown in Table 33, and behavior differences can be seen in Figure 67.

Table 33**.** Fitness Score Weightings.

Score	**Stage 1**	**Stage 2**	**Stage 3**
Collisions	0.275	0.25	0.25
Deadlocks	0.205	0.2	0.2
Inter-drone Distance	0	0.15	0.15
Distance-to-goal	0	0.2	0.2
Velocity	0.21	0.1	0.1
Cohesion	0.21	0.1	0.1
Number of clusters	0.05	0.05	0.05
Alignment	0.05	0.05	0.05

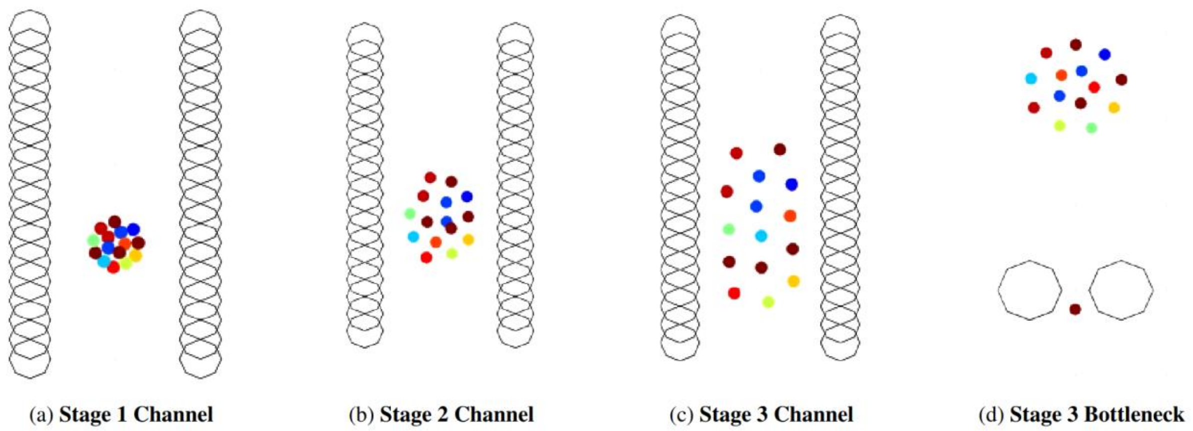


Figure 73. Swarm Behaviors. Selected maps highlighting the behavior of the swarms from different stages. Stage 3 Bottleneck shows a time out.

10.4.6 Adversarial Tuning

Since the main goal of the swarm is to be a robust fallback mode, an attack is crafted for testing and adversarial tuning, which targets the weak points of the swarm. One weakness is that a drone may still be allowed to move to a position that will intersect another drone's velocity. For example, in Figure 68, the red drone is allowed to move to intersect the velocity of the black drone. CA will still trigger for the black drone, but the intersection may occur far within the CA threshold. Due to inertia, braking may not occur in time to avoid collision. The swarm optimizes against this scenario by increasing repulsion and inter-drone distance, but ultimately the CA does not provide guarantees against these collisions. A second issue is that the swarm is susceptible to deadlocks if the CA can be frequently triggered, which will increase energy usage or get the swarm stuck. A third issue is that since drone behavior is correlated with other nearby drones in the swarm, a single drone can have a large effect on the behavior of the swarm. SwarmFuzz demonstrates this with a swarm propagation vulnerability attack.

An oscillation attack is developed to exploit the weaknesses of the swarm. The attack is simple, efficient enough to be used during adversarial tuning, does not require map awareness, and is plausible in the real world. The attack spoofs the position of a single drone so that it continuously moves left and right relative to the goal over some cycle period. The forward and backward positions are similarly spoofed. This attack has several effects:

- It creates constant collision and deadlock risks where the drone crosses the velocity vectors of others.
- It causes inefficient movement due to oscillation, braking, and acceleration, which wastes the limited energy of the drones.
- The oscillation of the target drone can propagate to the rest of the swarm, magnifying the effect.

Two attack variants are used, front and middle, depending on which drone is chosen to spoof. For the front attack, at each time step the drone closest to the goal is spoofed. For the middle attack, the drone closest to the middle of the swarm is chosen. The idea is that the front attack is meant to induce more slowdown and deadlocks, especially in maps with narrow passages, since the front drone will impede the whole swarm. In the middle attack, the target drone is more likely to be included in the velocity calculations of other drones, so it may have greater capability for attack propagation and collisions.

10.4.7 Results

The baselines are the initial reference Vásárhelyi parameters and the results of the three tuning stages. One evaluation goal is to verify that the tuning stages improve robustness and behavior of the swarm, judged by collisions, deadlocks, completion time, fitness score, and visual analysis of behavior. Another goal is to verify the threat of the oscillation attack and the effect of adversarial tuning.

10.4.7.1 Behavior and Robustness

Key results are shown in Table 34. The initial Vásárhelyi baseline does not perform well, as shown by numerous timeouts, collisions, very slow completion times, and a poor score. Stage 1 tuning succeeds in its goals of completing the maps and calibrating the CA parameters. It experiences no collisions, deadlocks, or time outs when the CA is active or inactive, despite only being tuned with an active CA. The main difference with CA is that the score is slightly improved, which is due to a slightly better inter-drone distance score. This means the CA did trigger in some cases to maintain distance from other drones, and ultimately, this did not have a negative effect on completion times.

Despite the good performance of Stage 1, the Channel map in Figure 67 shows that the behavior is not ideal. The cohesion was far too strong, leaving little room for unexpected situations. Stage 2 tuning added the inter-drone distance score to improve this. Similar to Stage 1, in Stage 2, the results show that CA slightly improves over no CA, but they both have no time outs, collisions, or deadlocks. A significant difference compared to Stage 1 is that the time is slower due to higher repulsion.

Stage 3 was tuned with adversarial tuning always active and a velocity of 5 m/s instead of 6 m/s to improve convergence. Again, there are no collisions or deadlocks, despite being tuned at a slower velocity. Noticeably, the results are the same between CA and no CA. This indicates that CA never needed to activate. The reason is seen in the Channel map, where the inter-drone distance is even further increased, and drones have a very uniform distance. However, this does cause an issue, as seen in the Bottleneck map, where all the time outs occurred. The repulsion forces are too high to always squeeze through the gap, leaving the final drone stuck. The previous drones can push through because the drones behind them help with their repulsive force. Once drones go through, they rapidly continue, leaving the communication radius, such that they do not help pull drones through the gap with cohesion. Aside from timing out on Bottleneck, which may be seen more as a design restriction issue, the performance is roughly equivalent to Stage 2.

10.4.7.2 Adversarial Tuning

Results of the oscillation attack applied to Stages 2 and 3 are shown in Table 35. Stage 2 with and without CA suffers dramatically, with most seeds encountering collisions, greatly reduced scores, and some slow down compared to no attack. In contrast, Stage 3 has almost no seeds with collisions, minor score loss, and negligible slowdown.

The main source of collisions and time outs is due to "compression" effects near obstacles. When attempting to split around an obstacle, some drones have difficulty deciding which way to split. In these situations, the oscillation attack is able to push drones closer together and closer to obstacles. Once the oscillating drone moves on, the compressed drones rapidly expand out due to repulsion forces. The expanding drones can more easily and abruptly cross into the velocity vectors of other drones to cause collisions. As for time outs, similar to the Bottleneck issue with Stage 3, the expanding drones quickly leave the communication range, and the final drone gets stuck behind an object.

The compression and expansion effect appears to be one reason why the middle attack variant is noticeably stronger than the front attack. Aside from generally affecting more drones, the target of the middle attack is more likely to remain near obstacles since most drones will slow and congregate there. This allows the attack to potentially compress drones and keep them stuck for longer periods of time.

Table 34. Results Without Attack. Testing the different stages with and without the CA active at velocity of 6 m/s and *CA_dist* of 0.7 m. Results are averaged across 5 maps with 6 seeds each. The maximum # of seeds is 6.

	Initial	**Stage 1**	**Stage 1**	**Stage 2**	**Stage 2**	**Stage 3**	**Stage 3**
	No CA	**No CA**	**CA**	**No CA**	**CA**	**No CA**	**CA**
vg # Seeds w/ Collision	3.4	0	0	0	0	0	0
vg Deadlock	0	0	0	0	0	0	0
vg # Seeds w/ Time Out	2.4	0	0	0	0	1.2	1.2
vg Time	114.86	45.69	45.64	52.75	52.55	68.13	68.13

Avg Score	0.32	0.092	0.083	0.031	0.028	0.037	0.037

Table 35. Results With Attack. Testing the different stages with and without the CA active at velocity of 6 m/s. *CA_dist* of 0.7 m is used for all except the last two columns where 1 m is used as indicated by CA=1. Results are averaged across 5 maps with 6 seeds each. The maximum # of seeds is 6.

	Stage 2 Front No CA	**Stage 2 Middle No CA**	**Stage 2 Front CA**	**Stage 2 Middle CA**	**Stage 3 Front No CA**	**Stage 3 Middle No CA**	**Stage 3 Front CA**	**Stage 3 Middle CA**	**Stage 3 Front CA=1**	**Stage 3 Middle CA=1**
Avg # Seeds w/ Collision	4.20	5.40	3.80	4.80	0.20	0.60	0.20	0.20	0.40	0.40
Avg Deadlock	0.000	0.000	0.108	0.128	0.000	0.000	0.008	0.053	0.064	0.037
Avg # Seeds w/ Time Out	0.00	0.00	0.20	0.60	1.20	1.20	1.40	1.40	1.20	1.00
Avg Time	53.30	54.21	56.69	57.71	68.31	69.84	68.35	69.81	68.28	69.67
Avg Score	0.219	0.272	0.219	0.263	0.048	0.064	0.049	0.059	0.068	0.062

10.4.8 Conclusion

Task 4 describes methods for tuning robust flocking d-swarms and validates their potential as a fallback mode for safely continuing operations in challenging scenarios where preferred methods like path planning may fail. A Vásárhelyi flocking swarm is modified with CA to improve guarantees against collision and tuned to successfully complete a variety of maps. A novel attack is developed to further test the swarm and demonstrate that adversarial tuning can be effective, similar to adversarial training.

10.5 References

[1] Karel Pärlin, Muhammad Mahtab Alam, and Yannick Le Moullec. Jamming of uav remote control systems using software defined radio. In 2018 International Conference on Military Communications and Information Systems (ICMCIS), pages 1–6, 2018.

[2] Renato Ferreira, João Gaspar, Pedro Sebastião, and Nuno Souto. Effective gps jamming techniques for uavs using low-cost sdr platforms. Wirel. Pers. Commun., 115(4):2705–2727, December 2020.

[3] Xin Zhou, Jiangchao Zhu, Hongyu Zhou, Chao Xu, and Fei Gao. Ego-swarm: A fully autonomous and decentralized quadrotor swarm system in cluttered environments. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pages 4101–4107, 2021.

[4] Xin Zhou, Xiangyong Wen, Zhepei Wang, Yuman Gao, Haojia Li, Qianhao Wang, Tiankai Yang, Haojian Lu, Yanjun Cao, Chao Xu, and Fei Gao. Swarm of micro flying robots in the wild. Science Robotics, 7(66):eabm5954, 2022.

[5] Jur van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In Cédric Pradalier, Roland Siegwart, and Gerhard Hirzinger, editors, Robotics Research, pages 3–19, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[6] Li Wang, Aaron D. Ames, and Magnus Egerstedt. Safety barrier certificates for collisions-free multirobot systems. IEEE Transactions on Robotics, 33(3):661–674, 2017.

[7] Enrica Soria, Fabrizio Schiano, and Dario Floreano. Distributed predictive drone swarms in cluttered environments. IEEE Robotics and Automation Letters, 7(1):73–80, 2022.

[8] Dingjiang Zhou, Zijian Wang, Saptarshi Bandyopadhyay, and Mac Schwager. Fast, on-line collision avoidance for dynamic vehicles using buffered voronoi cells. IEEE Robotics and Automation Letters, 2(2):1047–1054, 2017.

[9] Juhwan Noh, Yujin Kwon, Yunmok Son, Hocheol Shin, Dohyun Kim, Jaeyeong Choi, and Yongdae Kim. Tractor beam: Safe-hijacking of consumer drones with adaptive gps spoofing. ACM Trans. Priv. Secur., 22(2), April 2019.

[10] Harshad Sathaye, Martin Strohmeier, Vincent Lenders, and Aanjhan Ranganathan. An experimental study of GPS spoofing and takeover attacks on UAS. In 31st USENIX Security Symposium (USENIX Security 22), pages 3503–3520, Boston, MA, August 2022. USENIX Association.

[11] Qianyun Zhang, Zhendong Wang, Biyi Wu, and Guan Gui. A robust and practical solution to ads-b security against denial-of-service attacks. IEEE Internet of Things Journal, 11(8):13647–13659, 2024.

[12] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH ’87, page 25–34, New York, NY, USA, 1987. Association for Computing Machinery.

[13] Enrica Soria, Fabrizio Schiano, and Dario Floreano. Swarmlab: a matlab drone swarm simulator. In 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 8005–8011, 2020.

[14] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. Proceedings of the AAAI Conference on Artificial Intelligence, 34(05):8018–8025, Apr. 2020.

[15] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net, 2017.

[16] Gábor Vásárhelyi, Csaba Virágh, Gergo Somorjai, Tamás Nepusz, Agoston E. Eiben, and Tamás Vicsek. Optimized flocking of autonomous drones in confined environments. Science Robotics, 3(20):eaat3536, 2018.

[17] Masahiro Nomura and Masashi Shibata. cmaes : A simple yet practical python library for cma-es. ArXiv preprint arXiv:2402.01373, 2024.

[18] Yingao Elaine Yao, Pritam Dash, and Karthik Pattabiraman. Swarmfuzz: Discovering gps spoofing attacks in drone swarms. In 2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), pages 366–375, 2023

[19] Raphaël Candelier. Fast clusterization of adjacency matrices. https://www.raphael.candelier.fr/?blog=Adj2cluster, 2011. [accessed 26-May-2025].

11 INTERVIEW STUDY

An interview study was conducted with SMEs to gather additional insights regarding design guidance and best engineering practices to ensure safe automation for UAS. Below, the major inputs from each SME are listed, followed by a summary of the recommended guidance identified based on those inputs.

11.1 Description of SMEs

The SMEs who were interviewed included:

- Federal government employee with 25 years of experience with runtime verification for safety-critical systems.
- Industry expert with 15 years of experience with the design and use of autonomous helicopters by the military.
- Academic researcher with expertise in meteorology as well as guidance, navigation and control including a focus on sUAS for 18 years.
- Federal government employee with 20 years of experience with sUAS and 13 years of experience focused on UAM operations. His expertise deals with the impact of winds and convective weather on UAS operations.
- Expert with 20 years of experience focused on research and development for aviation weather. This includes 15 years of experience with sUAS and 15 years of experience with larger eVTOLs.
- Expert with 17 years of experience in the development and applications of formal methods to evaluate risk associated with safety-critical systems.
- Expert with over 20 years of experience with flight controllers and flight dynamics, as well as certification of new autopilot systems and associated display systems. This includes 10 years of experience with sUAS and 10 years of experience with UAM aircraft.

11.2 Findings

The following sections provide a summary of the major inputs from seven interviews that were conducted. One focused on runtime monitoring and verification, one on the design and use of autonomous helicopters, three on supporting weather decision making, and one on the use of probabilistic risk assessments.

11.2.1 Runtime Monitoring and Verification

Runtime monitoring is the practice of writing code that runs on board the UAS, whose purpose is to monitor functionality at runtime and flag potential errors. The code is simply called a *monitor.*

Runtime verification is a special way of doing monitoring. It consists of two parts: 1) The usage of formal logic (for example, Boolean logic or temporal logic) for specifying correct and incorrect behavior during runtime, and 2) the automatic generation of code – a so-called monitor- that runs on board the system and *monitors* whether the requirement is being violated.

To ensure safety, runtime monitoring and verification are important to complement compliance with design standards for safety-related systems such as IEC 61150, which covers electrical, electronic, and programmable electronic devices (Bellairs, 2019).

Key points emphasized in this interview regarding runtime monitoring and verification included the following. First, runtime monitoring should monitor for both causes of undesirable system states as well as the undesirable system states themselves (outcomes). This may include monitoring the external environment (such as monitoring winds to make sure they don't exceed operational limitations, or monitoring the RF spectrum to ensure there is adequate bandwidth to support communications), not just monitoring the state of the aircraft and ground control station.

Second, various types of systems engineering approaches to hazard analyses can be applied to predict possible hazards, ranging from Leveson's System-Theoretic Accident Model and Processes (STAMP) to FMEA to FTA (Leveson, 2011; Rausand & Høyland, 2004). The results of such analyses can then be used to identify different needs for monitoring observable states (causes or outcomes of hazards).

Third, detection of faults can involve direct sensing of such things as non-responsive control surfaces, overheating of an engine, excessive vibration, instability, weather conditions, or imminent collision detection. It can also be based on inferences using strategies such as fault tolerant voting or calculation of fuel reserves relative to the planned trajectory.

Fourth, redundancies are an important part of both monitoring for the detection of a fault and fault mitigation. Such redundancies can be based on two identical capabilities or comparison of the results from two distinctly different methods for sensing or inferring some state (e.g., GPS readings vs. dead reckoning that disagree on location, or disagreement of LIDAR and vision-based sensors regarding the presence of an obstacle).

Fifth, it can be useful to conduct a zonal analysis to define fault containment regions. This can help determine the placement of monitors. A caution, however, is the need to ensure that such monitors do not interfere with the system under observation, which in some cases needs to be onboard the vehicle and in others can involve remote sensors on a base station.

Sixth, based on discussions with flight operators planning to operate eVTOLs for UAM operations, there remains a need for the FAA to define clear guidance with a path for manufacturers and flight operators to identify and prevent potential hazards, including the incorporation of runtime monitoring and verification that can trigger appropriate mitigations. Such runtime monitoring and verification can provide an additional layer of safety as a means to certify new capabilities when the applicable traditional certification processes are not fully developed, with the process of monitoring for faults triggering associated mitigations being certified rather than the software or hardware itself.

Seventh, for onboard monitoring systems, the computational load on the system needs to be considered.

Eighth, detection of a problem based on runtime monitoring needs to be linked to appropriate mitigations. When possible, this should be proactive, discovering a problem that is beginning to develop rather than waiting until some subsystem fails. There are numerous focus areas where data may indicate that a subsystem failure is likely to fail, such as indicators of battery health (Kulkarni, Mehta et al. 2024), signs of failure in the UAM propulsion system (Montazeri et al., 2025) and the health of electric powertrains (Kulkarni & Corbetta, 2019). Such health monitoring can be used to alert the remote pilot to initiate an action or to trigger an autonomous response to initiate some mitigation before a system failure occurs. There are model-based, data-drive and hybrid methods that have been studied.

Recommended Guidance – Runtime Monitoring and Verification. The most significant insight focuses on the sixth input: Approval or certification of automation from a safety perspective can be based not solely on the level confidence established for the automation itself, but also based on determining that there is a highly reliable monitoring process (using automation or in some cases human supervision) that can detect some failure of the automation and can reliably trigger a sufficiently safe mitigation. The need to link detection of a problem based on runtime monitoring to appropriate mitigations is also a significant consideration, including proactive detection of a developing problem rather than waiting until some subsystem fails.

The other inputs are important but more widely recognized, such as application of accepted established hazard analysis techniques with standards for safety-related systems, compliance with standards for safety-related systems, and incorporation of cross checks of sensors or system performance based on different sensing systems.

11.2.2 Flight Control, Flight Dynamics, and Certification

This SME has over 20 years of experience with the design, evaluation, and certification of flight controllers, including the evaluation of the airworthiness of autonomous systems with embedded AI. This includes the assessment of performance with windy conditions in urban canyons.

His inputs included:

- It is important to have the “right” controller for an application, such as the ability of a controller to track its flight path in the face of wind/gusts in urban canyons.
- Control power, accuracy, and bandwidth play into the usage of battery energy for small UAS using electrical power for control of flight path, where controller response can lead to significant changes in battery use, endurance, current draw, motor controller wear, actuator/motor wear, etc. This needs to be considered in the design and monitored during operations. The type of controller can have a huge impact on the endurance of the aircraft.

Depending on the controller design, the power to control the path can affect the battery life. The amount of command activity at an actuator level working to keep the aircraft on station can wear out the battery.

- Knowledge of the controller performance with gusty cross-winds needs to support go/no go decisions by the remote pilot and, when applicable, by the autonomous controller to avoid potential crash zones.
- For rotorcraft, the maximum lateral flight path control can be assessed as part of certification. This can include evaluating the maximum lateral control in a hover to control error to the tolerance needed when counteracting a wind gust, and the ability to do a lateral step maneuver in gusty conditions. Responses to both vertical and lateral gusts to maintain the flight path need to be studied. FAA standards need to specify minimum performance requirements in such conditions. If the designers knew what performance tests would be used, they could design better controllers.
- A key question focuses on the governance of the aircraft. Does the automation decide to abort or does the remote pilot? The design of the runtime architecture can be hierarchical, layered to determine whether some limit is going to be exceeded and decide not to continue to protect the flight path integrity. This is especially important in an m-n architecture with one supervisor managing multiple aircraft. One flight operator has demonstrated this with automation that decides to continue or return to base, or in the extreme, to land on its own in a nearby safe area. They've had tens of thousands of flights and only had to do this twice.
- There are potential benefits from having AI control the flight path and help control detect and avoid. The demands could exceed the human pilots' capabilities, so AI is needed.
- The FAA would be open to an AI-based controller if the system bounds behavior to safe boundaries. This can be done with an architecture that limits system performance using a deterministic layer in which a deterministic executive controller wraps around a non-deterministic controller to bound system behavior and thus ensure safety (ASTM TR2-EB, 2020; ASTM AC377: Autonomy design)

Recommended Guidance – Flight Control, Flight Dynamics, and Certification. First, performance-based standards are needed to certify controllers and to inform the operators and designers of the controllers regarding performance limits. This includes evaluation of flight path adherence in the face of winds, wind gusts, and down drafts. The basis for such testing is well understood at this point. Second, a certified downward-sensing perceptual system is needed to support the remote pilot or the automation to ensure a clear landing area. Third, the automation needs to monitor flight performance to detect significant deviations from the intended flight path due to winds and wind gusts. In some cases, such deviations can alert the pilot. In others, it may be appropriate for the automation to abort the mission autonomously. Finally, AI offers an opportunity to safely increase performance by embedding it in an architecture with a deterministic layer that limits excursions by the AI.

11.2.3 Autonomous Aircraft

This interview was conducted with an industry expert with 15 years of experience with the design and use of autonomous helicopters by the military (Blackhawks, Bell helicopters, and helicopters more generally). These insights are important in terms of understanding the state of the art for military applications, which could provide technologies that can be adapted for UAM aircraft through technology transfer.

The interview focused on current military capabilities, which can be characterized as fully functional Beta-test systems that are undergoing further testing and that are available commercially with the understanding that such a purchase is of a Beta-test version (Near Earth Autonomy, 2025). The range of autonomous capabilities is indicated in Figure 68.

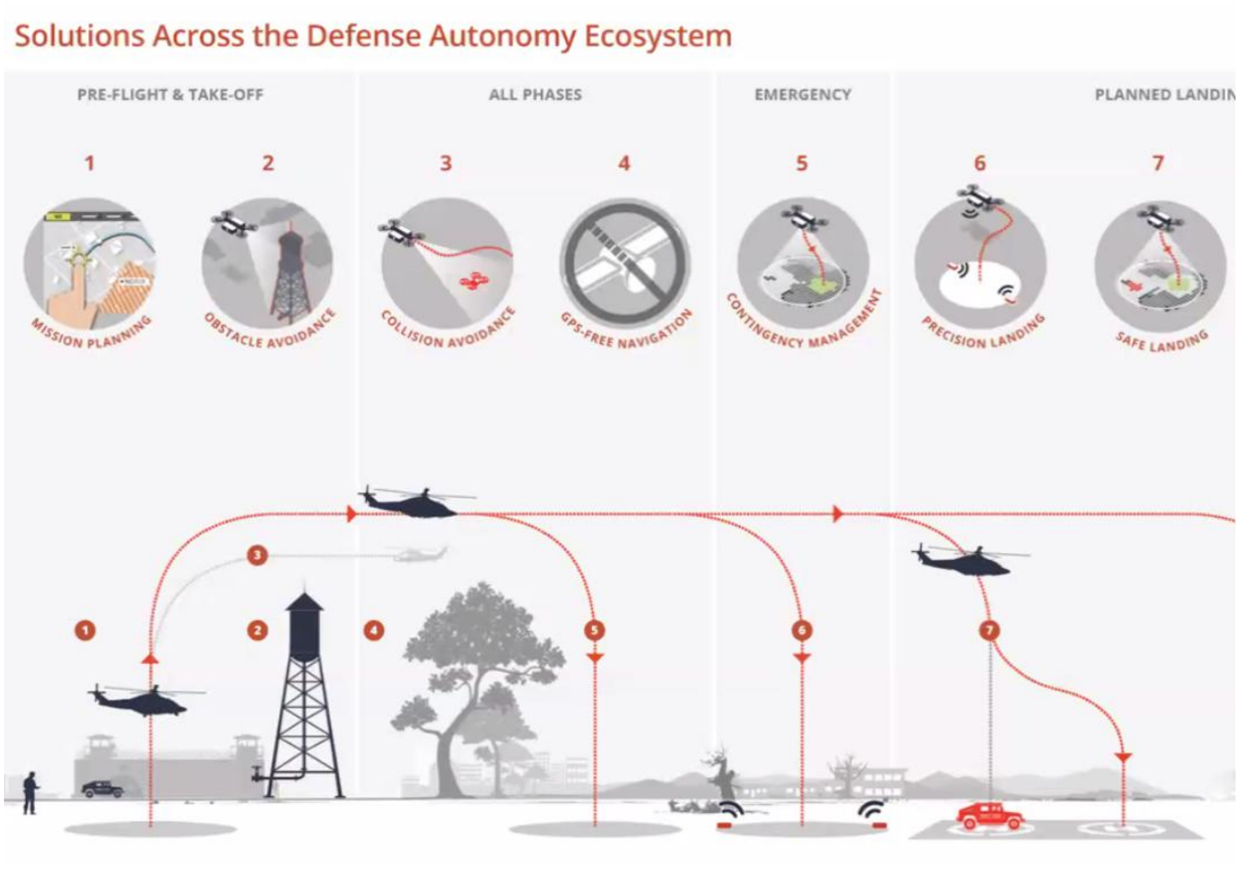


Figure 74. Autonomous capabilities for military helicopters (from Near Earth Autonomy, 2025).

These autonomous capabilities are highly relevant to supporting UAM operations with remote pilots when there is a loss of command and control communications, as well as the need for increased autonomy in Midterm and Mature Operations. This includes:

- Autonomous control for takeoff and precision landing.
    - Navigation in GPS-denied environments.
    - Contingency planning considering wind data and terrain if bingo fuel state is detected.
- Autonomous diversions.
    - Detection that a landing pad is unavailable due to some obstacle.

    - Search for an alternate unplanned landing site.
    - Planning an alternative trajectory.
- Detect and avoid.
- Resistance to cyberattacks.

Autonomous Control for Takeoff and Precision Landing. In addition to autonomous autoflight and autolanding using GPS coupled with inertial sensing, a second class of sensors and processing provides redundancy using visual dead reckoning odometry that includes matching data to satellite images and terrain maps.

Autonomous Diversions. These systems use an integration of short-range LiDAR and a vision system to determine whether the planned landing site is clear. The same technology is used by an autonomous search algorithm that can search the area around the planned landing site for an alternative location that is suitable for landing.

Detect and Avoid. The DAA system incorporates multiple dissimilar sensors, providing redundancy if one should fail, and it will autonomously adapt if a failure is detected with one of the sensors. This contingency management can include modifying the flight trajectory as necessary to adapt to the available set of sensors. This includes a forward-looking long-range LIDAR system, a short-range downward-looking LIDAR system, doppler radar, and a vision system.

Resistance to Cyberattacks. This capability focuses on limiting the size of the message set that can be transmitted from the ground (abort the mission; retask the aircraft) and using a secure, encrypted channel that does not communicate with a continuous stream. This does not make such systems immune to all kinds of cyberattacks but limits the range of attacks to which they are susceptible. The philosophy is that the less that is communicated, the less vulnerable the aircraft is.

There may be aspects of this technology that are applicable top UAM operations if access to such designs is not limited by national security considerations. Such technology transfer may also be limited by cost constraints.

Recommended Guidance – Autonomous Aircraft. The main insight provided by this interview is the degree to which the military has already made advances in the design and operation of autonomous helicopter-sized vehicles. While some of the technology clearly has counterparts in currently developed UAM technologies, opportunities for additional technology transfer merit investigation.

11.2.4 Meteorology: Weather Sensing, Modeling, and Decision Making – SME 1

Three of the SMEs offered expertise regarding meteorology (convective weather and winds) as it pertains to the operation of UAS. From an automation perspective, this relates to technology that supports weather sensing and modeling, as well as the translation of meteorological input to facilitate preflight and real-time decision-making. Concerns include wind flows, convective weather, and turbulence. One of the SMEs was specifically an expert in the performance of Guidance, Navigation, and Control (GNC) models relative to wind fields and storm activity. The inputs from the latter are summarized below.

Of particular note were the following inputs:

- Convective weather and winds in the boundary layer for urban areas have spatial and temporal features that are different from higher altitude weather patterns and flows.
- Filling information gaps. There are many urban areas where the boundary layer is not currently well-equipped for weather sensing. To close the data gaps and provide the data needed to support weather decision making, it would be valuable to equip drones used operationally as weather sensors and to share these data over a network, both to provide data for developing more accurate weather models as well as to support real-time decision making.
    - There is a need to better understand how to collect reliable weather and wind data using UAS.
    - Buildings can amplify and distort wind fields around buildings and would similarly benefit from data collected by drones.
    - There is a gap in access to turbulence data that could be filled with observations provided opportunistically by drones.
    - Weather models will need to be developed for specific urban areas. For example, in Houston, there are sea breezes off the gulf and bay almost every day in the summer, resulting in turbulence.
    - Weather data from the vicinity around an urban area needs to be collected as well, as such inputs support forecasting of the weather that will develop in the urban area itself.
    - Appropriate weather models need to be developed based on sufficient data.
    - Historical weather and wind data could be used to make decisions about the placement of vertiports.
- In terms of supporting decision-making by the flight operators, integration of the potential firehose of information into effective displays is critical. There is a need to better understand how to provide such integrated weather and wind displays in a usable format. There is a need to provide adequate training to flight planners and pilots so that they can effectively interpret the displayed weather information when making decisions.
- Although UAS flights may be relatively short, such as 20 minutes, unexpected convective weather and winds can develop in that time frame. Unexpected storm motion and intensity can arise, especially if there is already some activity in the vicinity.
- While more robust GNC capabilities for sUAS in the presence of very "windy" conditions in an urban environment can improve the ability of an sUAS to survive adverse weather conditions, the primary missing ingredient for safety is the availability of good, actionable convective weather and wind information for operators to make preflight decisions about proceeding with a mission, planning a route or diverting a flight.
- UAS and UAM safety is enhanced by improvements in GNC robustness and, especially for fixed-wing sUAS, also by the availability of reserve propulsive capability to "power through" potential upsets due to adverse weather conditions.
- Accurate wind observations are particularly important during descents, as the thermodynamics can be particularly challenging at that phase of a flight. Consistent with

this, one flight operator has indicated that their primary concerns are down-drafts and out-of-date knowledge of collision obstacles (e.g., new buildings) along the path of a package delivery route, indicating areas that need to be further addressed. Similarly, a study at one major hospital has indicated that the primary concern is knowledge of weather conditions at the landing site. In particular, many air ambulance flights are called off to the detriment of patients with critical health problems due to the uncertainty of weather conditions at the landing site.

- If UAM flight operators have to rely on weather data from a limited number of sites, such as nearby airports without more local weather and wind data at vertiports and other landing sites, their ability to make effective go/no go decisions will be limited.
- Standards need to be provided regarding weather sensors and their housing on drones, as well as weather modeling techniques. Machine learning models could be very useful, but they depend highly on the quality of the data used to train and test the model. Of particular concern with such models is the scale of motion captured in the data available.

Recommended Guidance - Meteorology: Weather Sensing, Modeling, and Decision Making – SME 1. Several areas were emphasized that provide design guidance. A major conclusion is that poor weather decision-making (go/no go and diversion decisions) is more likely to result in incidents than a lack of increasingly adaptive GNC systems. This suggests the need for a focus on providing sufficiently informative local weather data and model-based forecasts (which could also include input from locally knowledgeable meteorologists). This includes ensuring adequate distribution of accurate weather sensors (including placement at launch sites and the use of UAS to feed a network with weather data) to provide the data. It also includes developing effective interfaces to support decision-making that integrate the "firehose" of data that needs to be available. Increasing the robustness of GNC capabilities to deal with windy conditions can improve performance, but helping to ensure good weather decision-making is essential.

11.2.5 Meteorology: Weather Sensing, Modeling, and Decision Making – SME 2

This SME is a researcher with a Federal Government organization with 20 years of experience with sUAS and 13 years of experience focused on UAM operations. His expertise deals with the impact of winds and convective weather on UAS operations and GNC technology, including microweather in the boundary layer. His input emphasized the following:

- The level of understanding of weather at the level of granularity required for urban UAS operations is still limited. This lack of knowledge could contribute to operational failures if not resolved.
- It is not sufficient to rely on reactive robust aircraft control systems to ensure safety. Such technologies may handle 95% of the situations that may arise, but significant risk remains with that approach alone.
- To provide increased data, there is a need to implement lightweight weather sensing technologies that can be carried on the UAS. Improved, reliable sensing networks then need to be developed to collect this data. A barrier to the availability of such networks is the proprietary concerns, including hesitancy due to liability regarding claims regarding the data. Standards are needed.

- A major problem affecting current helicopter operations in many locations is an insufficient number of weather stations in many locations, often limited to airports and hospitals. Such a limitation would be a major concern for UAS operations. Wind data from an airport in the vicinity is insufficient for making decisions about landing at a hospital. These considerations have a major impact on air taxi operations. They “can’t be worrywarts and never fly a mission, but even a handful of incidents could cripple the industry.”
- Unexpected downdrafts are one of the biggest concerns. Zipline has the best evidence to understand this.
- The considerations of insurance adjusters and the public have to be considered.
- For missions like air ambulances, risk management needs to be considered on a case-by-case basis. If at the time a go/no go decision has to be made, if there isn’t sufficient information, they will have to err on the side of caution.
- In the spectrum of low risk to high risk, there is a wide fuzzy area in the middle. This area can be reduced by increasing the robustness of the GNC technologies. “They can make control more robust, but they can still be overwhelmed.”
- However, the contribution of robust controllers can be “lost in the noise”, meaning that these technologies “won’t solve the problem of a lack of adequate predictability to decide to launch.” There is a need to ensure that adequate data is collected and can feed effective predictive models that integrate and display the data in a way that supports more informed decision-making for launch decisions. However, we “don’t understand sensor fusion well enough yet.” Real-time data could be used to make decisions while a flight is enroute to avoid a hazard that has developed.
- “I don’t see as much real-world testing as I think we need.”
- The designers of facilities need to be aware of aviation concerns. Wedging in aviation as an afterthought isn’t effective. The needs of the aviation community need to filter back to infrastructure development.
- Machine learning may have potential to support weather modeling, but it has to have good data to train the system.

Recommended Guidance - Meteorology: Weather Sensing, Modeling, and Decision Making – SME 2. This second SME with expertise in meteorology and GNC provided the same conclusions as SME 1. First, that weather decision-making is critical and needs a major focus in terms of the availability of data and the technology to support that decision-making. Second, that more robust GNC capabilities can improve performance and thus safety when windy conditions are encountered, but such technological support will not improve safety as much as ensuring good weather decision-making.

11.2.6 Meteorology: Weather Sensing, Modeling, and Decision Making – SME 3

The third SME has 20 years of experience with research and development focused on aviation weather and weather modeling. This includes 15 years of experience with sUAS and 15 years of experience with larger eVTOLs.

His major inputs were as follows:

- Industry hasn't done enough to understand how to support weather decision-making effectively for UAM. "For operations starting in the next 2-4 years, we are not well prepared. We need to use an urban site as a testbed that is well-equipped and exercise real-life scenarios."

  There are some examples of such progress. Zipline provides one such example, using its aircraft as weather sensors. Zipline Team (2025) discusses their approach:

  > "The Zip measures position, velocity, orientation, and acceleration. If you have all that information, you can back out the forces that must be acting on the drone, which tells you which direction and how fast the wind is blowing."
  >
  > "For example, one of [their] top priorities was to keep Zips away from gust fronts. … When a thunderstorm forms, rain falls to the ground very quickly because it's cold. When it does, it brings a lot of air with it. When the air hits the ground, it spills in all directions, creating a gust front … This portion of the storm is powerful enough that U.S. air traffic controllers ground commercial planes to avoid them. U.S. airports have wind shear detection systems that predict where they may happen." "Gust fronts happen the moment a thunderhead forms, and massive, 50-mph updrafts and downdrafts can force the aircraft to the ground."

  Another example of progress in this regard is a project summarized by Tinnesz (2022) that has instrumented an **urban weather sensing infrastructure testbed** in Hampton, Virginia, to demonstrate:

  > "delivery of more granular weather data and forecast services for low project altitude urban and suburban flight to enable weather aware Uncrewed Aerial Systems (UAS) beyond-visual-line-of-sight (BVLOS) operations and the AAM (Advanced Air Mobility) industry".
  > "Two innovative MetroWeather Doppler Lidars, from Kyoto-based venture MetroWeather, Inc., will provide 'MRI-like' wind and potential cloud height information covering over 30-40 area miles and up to 6,000 feet above the ground. The test bed will integrate observations from these lidars, 30 ground sensors, satellite data, and potentially a radar to provide services that will provide greater certainty about where and when it is safe for AAM vehicles to operate. The testbed and future systems will inform which routes will provide the most advantageous weather and inform locations selection for terminal operations. The service will enable improvement of UAS and eVTOL power management, payload weight estimates, travel times, and flight separation services to improve airspace traffic management".

  This work is based on the conclusions that: "We need an adaptive, affordable sensor array to complement weather satellites and weather radar technology and to provide more accurate data to train machine learning models to reliably provide the operational picture required to safely navigate the urban airspace."

- There are a number of advances that need to be made:
  - "The weather data and information need to guide decisions about what to do in the next half hour. Do we abort? Do we change the team's flight plan?"
  - "Local meteorologist input is absolutely necessary for urban flight operations. Someone who appreciates urban weather and can translate it into impacts and the limitations on operations. You need to do the homework before you launch. And the meteorologist needs to understand flight operations and the limitations of aircraft and build a trusted two-way relationship with the flight operators."
  - "Flight planning needs to have a good forecast. You need real-time high resolution forecasts. And you need to be monitoring changes in conditions while airborne to decide whether to continue or abort. If a storm is evolving nearby, you need to start paying attention to it. We'd like more automation, but that is way down the road."
  - "All small aircraft should have met sensors onboard. Every drone needs to have as meteorology sensors onboard. This is essential. If the team could share this information, it would fill things in. Then you could use the data with a simulation to improve the modeling."
  - "We need more work translating into urban canyons."
  - "You need to be monitoring the remaining battery capacity. These aircraft have limited operational range and may have to divert or come back because of the draw on the battery."
  - "You need a reporting system that gets input even if there is no injury to people or damage to the vehicle or property, so everybody can learn. This includes understanding how weather is a contributing factor."
  - "It would help a forensic analysis after an incident if the aircraft collected data along the way. Weather can be derived from some of those parameters."
  - "You need to understand what kinds of weather you can operate in to develop a business case."
  - "We need to study operational decision making. This includes the flight operators, the weather folks, and the regulators."
  - "We need to embed weather in the regulator's policy and guidance documents."
  - "The pilot needs to be certified with respect to the urban weather, especially the remote pilot."
  - "Guidance on separation distance from buildings is necessary, but it depends on the weather conditions. If wind speed increases or the flow changes, you may have to have an increased separation from buildings for a specific vehicle."
  - "Thunderstorms create a whole set of hazards: Wind and wind gusts, turbulence, hail, icing, lightning, microbursts. You need to monitor for short-term changes."
  - You need onboard microscale simulations to provide sufficiently fast computations. "An example of this is provided by FastEddy, a resident-GPU model, meaning that all prognostic calculations are carried out in an accelerated manner on GPU with CPU utilization strictly limited to model configuration and input/output of

modeling results. This resident [onboard] GPU approach shows tremendous early potential for achieving faster-than-real-time microscale simulations … that can support tactical responses to the weather (NCAR, 2025).

- “AI is a two-edged sword. There are opportunities to identify different types of situations, but it can’t be applied blindly. AI is too much of a buzzword, with developers simply saying “I can do this” without a real understanding of the meteorology and the application. It needs to be used thoughtfully.”

Recommended Guidance - Meteorology: Weather Sensing, Modeling, and Decision Making – SME 3. The interview with SME 3 echoes the recommendations of SMEs 1 and 2 but provides a number of important additional insights. The list above should be reviewed, as many of them provide important guidance regarding priority areas to consider.

11.2.7 Application of Probabilistic Risk Assessments

This SME works for a Federal Government organization and has 17 years of experience in the development and applications of formal methods to evaluate risk associated with safety-critical systems. He was asked to provide insights on risk assessment techniques that the FAA could use to evaluate the risks associated with the use of UAS. The scope includes both small UAS and larger eVTOLs for use in UAM. The larger eVTOLs could either have a pilot onboard or a remote pilot. Of particular interest is the assessment of the risks associated with particular safety automation, such as software for Detect and Avoid or control software to manage flight if a motor fails.

He provided input on the following questions.

1. Is there value in applying a PRA to evaluate the overall risk associated with a particular scenario based on a specified CONOPs?

   ANSWER: This is a very valid approach when you can get good data for the probabilities being dealt with. This is the problem with the security side. There is a long history of work in reliability engineering with really good data, but the same techniques applied to fields without data don't yield the same thing. Understanding the limits of the applicability of the methodology is important. Otherwise, it is a house on sand.

   The stoplight FAA risk tables have analogs in other organizations with high-risk environments.

2. Is there value in applying a PRA to assess the risk associated with an individual safety automation technology? (such as Detect and Avoid software) Does this need to be scenario-based?
   - Evaluating automation alone?
   - Evaluating human-automation interaction?
     - ANSWER: Yes, it is. Again, getting the data is the challenge. But it is a perfectly valid way to find what risk is being dealt with.

3. Is there value in just developing relevant fault trees?

- ANSWER: This comes from the world of hardware, where it was extraordinarily successful, but less successful in software. Collins and GE safety folks have internal techniques that they do not share that use fault trees. It is also used in safety analyses for aerospace engineering.

  Fault trees are perfectly valid. Other approaches that have been developed since, such as STAMP, are maybe more attuned to a more software-centric world. What we have seen in disciplines like formal methods is that fault trees get used in specific aspects. FAA regulations call for many different safety analyses, due to the thought that other techniques may be complementary in nature.

4. At what level of detail should the fault tree be developed?
   - ANSWER: Analyzing a particular subsystem of a larger subsystem is an approach to make the use of fault trees more tractable. This also helps address the concern that large, fine-grained fault trees are large and difficult to parse and understand by humans.

5. What impact do simplifying assumptions have on the validity of a PRA?
   - Assumption of a single fault?
   - Assumption of independence of probabilities associated with particular event, action or decision nodes in the fault tree?
     - ANSWER: Often, the assertion is that we will plug in a number and will validate assumptions and estimates going forward, but then that validation is not done. It is very important to have a feedback loop to evaluate assumptions. Mistakes in estimates can lead to very wrong behavior prediction. A feedback loop for validation is essential.
     - When the PRA methodology is used by experienced analysts, it is a useful thing.

6. Estimation of probabilities
   - Expert elicitation. What are the important considerations regarding the methods for eliciting probabilities from domain experts?
     - ANSWER: For probability elicitation, when working with experts, it is important to make sure they can operate in a free manner since they are often wary about their own intellectual property. They don't want to tell you anything that could be revealed to a competitor. This is a problem in eliciting probabilities from experts who work for companies that might use such information in developing products. This is a problem with experts in aerospace since the best experts work for companies with lots of IP to protect. Even when paying the experts, it is still very tricky.
     - Experts will have spent entire careers working on systems with a single architectural pattern. You need to pick SMEs that are familiar with the design or pattern of the system under test. Otherwise, there is a tendency

for SMEs to propose to change the architecture and, therefore, the decision or event tree to match their expertise.

7. Estimation of probabilities based on frequencies from historical data
    - Should confidence intervals be considered?
        - ANSWER: Generally, we like having confidence intervals. It matters regarding what the confidence level is and the width of the interval.

8. What considerations need to be addressed if a simulation is used to generate probabilities?
    - ANSWER: A concern is the validation of the simulation. That is usually the issue in simulations. Lots of people only know Simulink. It is important to understand the limitations of what you are modeling and remember the old adage that "all models are wrong, but some models are useful". Focus on the fidelity of the physics is great, but the Simulink build doesn't result in an implementable controller, e.g., due to computational delay. This issue is particularly acute in robotics, and some developers don't use simulations since they would rather believe you have to build it and tweak it. That approach limits the scope of the evaluation.
    - Knowing the limits of the simulation is all-important.

9. Should the branches be represented as discrete paths or as probability density functions?
    - ANSWER: In many cases, the paths can be discretized. However, in some situations, it may be necessary to consult an expert in the "continuous world."

10. Quantification of outcomes (consequences associated with branches of the fault tree)
    - If the outcome for a particular branch is actually a range of possible outcomes, should an average be used (vs. worst case or best case) or should these be reflected as separate branches?
        - ANSWER: Often, the analyst is overwhelmed after a couple of levels of the fault tree. For them, worst-case and best-case are better than an average. Usually, the worst case is of greatest interest.

11. Should other methods (such as FMEA) be considered for use to complement PRAs?
    - ANSWER: But guidelines for the FAA identify twelve types of analysis that are mandated, such as a zonal analysis. Usually, some incident happened and the FAA adds an additional methodology. For example, when computers were added to flight in the 1980s, McDonald Douglas put all three redundant flight computers in tails of aircraft. But condensation in the tail shorted all three computers, so, after that, the computers had to be partitioned into separate fault containment regions to meet requirements resulting from a zonal analysis.
    - It is useful to have complementary techniques as they highlight different things and stress different issues, e.g., cascading failures.

Recommended Guidance – Application of PRAs. Recommended Guidance – Application of PRAs. Several points were emphasized in this interview. First, to be tractable, PRAs and fault tree analyses often focus on specific subsets or aspects of the system. Second, in addition to being sufficiently informative, it is valuable to indicate the confidence intervals associated with point estimates for probabilities. A worst-case analysis is one important approach to indicate such ranges and can include ranges for outcomes as well as probabilities. Third, it is useful to apply a combination of risk assessment methods as a complement to each other. And fourth, PRAs and fault trees require data. When insufficient data is available, elicitation or probabilities from experts is a possible substitute. Regardless of the source of the probabilities and confidence intervals, as operational experience accumulates for a system or subsystem, this should be used as feedback to adjust analyses.

11.3 Interview Study - Conclusion

These interviews guided the design and use of safety automation in a number of areas. In many cases, gaps in the team's current knowledge to ensure safe operations were also identified. Key inputs included:

- Redundancy in the design of safety-critical functions is very important. This can include identical backups for hardware or software that can be employed if the primary fails. Redundancy through the incorporation of different methods providing diverse backups (e.g., navigation using GPS/GNSS vs. an Inertial Navigation System vs visual dead reckoning odometry) is also highly recommended.
- Runtime monitoring is necessary for safety-critical systems, with effective mitigations should some subsystem fail.
- Runtime monitoring should focus on both causes of undesirable system states as well as the undesirable system states themselves (outcomes). When possible, designs for early detection of a potential system failure should be integrated, as a wider range of solutions to ensure safety is available with such a proactive approach.
- Performance-based standards are needed to certify aircraft controllers and to inform the operators and designers of these controllers regarding performance limits. This includes evaluation of flight path adherence in the face of winds, wind gusts, and down drafts.
- A certified downward-sensing perceptual system is needed to support the remote pilot or the automation to ensure a clear landing area.
- The automation needs to monitor flight performance to detect significant deviations from the intended flight path due to winds and wind gusts. In some cases, such deviations can alert the pilot. In others, it may be appropriate for the automation to abort the mission autonomously.
- For onboard monitoring systems, the computational load on the system needs to be evaluated.
- A combination of safety analysis methods should be applied to provide complementary assessments. An example, in addition to PRAs, FMEA, and fault tree analysis, is a zonal analysis to ensure there are effective fault containment regions.

- To be tractable, it is often necessary to limit PRAs and fault tree analyses to specific aspects of a system.
- Confidence intervals should be incorporated in a PRA or fault tree analysis to indicate the level of uncertainty with point estimates for probabilities.
- The application of machine learning needs to be carefully evaluated. While there are opportunities for its application, as one SME noted, “It can’t be applied blindly. AI is too much of a buzzword, with developers simply saying ‘I can do this’ without a real understanding of the … application. It needs to be used thoughtfully.”
- There are opportunities to apply AI safely by embedding it in a hierarchical system with a deterministic layer that prevents the AI from unacceptable excursions.

Identified gaps included:

- There are many urban areas where the boundary layer is not currently well equipped for weather sensing. To close the data gaps and provide the data needed to support weather decision making, it would be valuable to equip drones used operationally as weather sensors and to share these data over a network, both to provide data for developing more accurate weather models as well as to support real-time decision making.
- In terms of supporting decision-making by the flight operators, integration of the potential firehose of weather information into effective displays is critical. There is a need to better understand how to provide such integrated weather and wind displays in a usable format.
- There is a need to provide adequate training to flight planners and pilots so that they can effectively interpret the displayed weather information when making decisions.
- A primary missing ingredient for safety is the availability of good, actionable convective weather and wind information for operators to make preflight decisions about proceeding with a mission, planning a route, or diverting a flight.
- Standards need to be provided regarding weather sensors and their housing on drones, as well as weather modeling techniques. Weather sensing, modeling, and decision making need to be embedded in the regulator’s policy and guidance documents.
- Local meteorologist input is necessary for urban flight operations, someone who appreciates urban weather and can translate it into impacts and the limitations on operations. The meteorologist needs to understand flight operations and the limitations of aircraft and build a trusted two-way relationship with the flight operators.
- Clear guidance from the FAA and standards organizations is required, providing a path for manufacturers and flight operators to identify and prevent potential hazards through the incorporation of runtime monitoring and verification that can trigger appropriate mitigations.
- Operational data should be collected on the performance of flights as feedback for risk assessments.

11.4 Conclusion

The interviews place the findings of other chapters in context and highlight challenges facing the implementation of identified approaches and guidance.

11.5 References

Bellairs, Robert. 2019. “What Is IEC 61508? Determining Safety Integrity Levels (SILs).” *Perforce Software*. (https://www.perforce.com/blog/qac/what-iec-61508-safety-integrity-levels-sils).

Leveson, Nancy. 2011. *Engineering a Safer World: Systems Thinking Applied to Safety*. Cambridge: MIT Press.

NCAR. 2025.“FastEddy.” *FastEddy Research Applications Laboratory*. (https://ral.ucar.edu/projects/fasteddy).

Near Earth Autonomy. 2025. “Introducing Firefly.” *Firefly (Miniaturized Flight Systems). Near Earth Autonomy*. (https://www.nearearth.aero/firefly).

Rausand, Marvin, and Arnljot Høyland. 2004. *System Reliability Theory: Models, Statistical Methods, and Applications*. 2nd ed. Wiley.

Tinnesz, Lauren. 2022. “TruWeather Solutions Prototypes Urban Weather Sensing Infrastructure.” *TruWeather Solutions*. (https://truweathersolutions.com/truweather-solutions-prototypes-urban-weather-sensing-infrastructure).

Zipline Team. 2025. “How Zipline Turned Its Aircraft into Flying Weather Stations.” *Zipline Newsroom*. (https://www.flyzipline.com/newsroom/stories/articles/how-zipline-turned-its-aircraft-into-flying-weather-stations).

12 CONCLUSION

This project has examined a very wide range of components, functionalities, and tasks for UAS. The guidance developed in each research area stands on its own, and implementation will contribute to improving the safety of individual UAS and of the overall integration of UAS into the national airspace. The systemic level research efforts, in risk assessment, traffic management, and robust inference, are modular to an extent where the guidance can be applied across a range of lower-level decisions. E.g., regardless of the particular default controller on a UAS, the proposed risk analysis methodology is applicable. The environmental research, in modeling urban climate, provides practical data for evaluating CONOPs in urban airspace.