UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

VÍTOR EULÁLIO REIS

Constraint-Aware Fault-Tolerant Multi-Agent UAV System Using OODA Loop: Realistic Mission Completion Assistance

São Carlos
2025

Vítor Eulálio Reis

Constraint-Aware Fault-Tolerant Multi-Agent UAV System Using OODA Loop

Monograph presented to the Specialization Course in Aeronautical Systems, School of Engineering of São Carlos, University of São Paulo, as part of the requirements for obtaining the title of Specialist.
Advisor: Prof° João Paulo Eguea, PhD

FINAL VERSION

São Carlos
2025

Cataloging card prepared by the Library Prof° Dr. Sérgio Rodrigues Fontes at EESC/USP with data provided by the author(s).

Eulálio Reis, Vítor

Constraint-Aware Fault-Tolerant Multi-Agent UAV System Using OODA Loop: Realistic Mission Completion Assistance. / Vítor Eulálio Reis; advisor Prof° João Paulo Eguea, PhD. São Carlos, 2025.

Specialization (Specialization in Aeronautical Systems) – School of Engineering of São Carlos, University of São Paulo, 2025.

1. Drone. 2. OODA Loop. 3. Fault Tolerance. 4. Multi-Agent Systems. I. Title.

APPROVAL SHEET

APPROVAL SHEET

Approval sheet

Candidate / Student: Vítor Eulálio Reis
Title of TCC / Title: Constraint-Aware Fault-Tolerant Multi-Agent UAV System Using OODA Loop
Defense date / Date: October 18, 2025

Examining Committee	**Result**
Prof° João Paulo Eguea, PhD	
Affiliation: School of Engineering of São Carlos / EESC-USP	
Prof° Jorge Bidinotto, PhD	
Affiliation: School of Engineering of São Carlos / EESC-USP	

Chair of the Examining Committee:

_____

Prof° João Paulo Eguea, PhD

(Signature)

To my family, for their unconditional support throughout this endeavor.

ACKNOWLEDGMENTS

To the professors of the Especialização em Sistemas Aeronáuticos at the Escola de Engenharia de São Carlos, USP, for sharing their knowledge, expertise, craft, and for their dedication in supporting students throughout the course.

To Prof° Dr. João Paulo Eguea for his mentorship in guiding this research.

To Prof° Dr. Jorge Bidinotto for his leadership in managing the Specialization Program.

AI Disclosure: This work was developed with assistance from large language models. **Claude** (Anthropic) was used via Claude Code with both Sonnet (claude-sonnet-4-20250514) and Opus (claude-opus-4-20250514) models to support software development, experimentation and user interface design. Claude Sonnet was also used in playground mode for literature survey, reference verification, formatting, comparative analysis, and standardization. **Gemini** (Google) was used in playground mode for literature survey, linguistic refinement, and as a simulated peer reviewer. **ChatGPT** (OpenAI) was used in playground mode as a simulated peer reviewer and evaluator. All technical decisions, system architecture, algorithm design, experimental validation, and intellectual contributions remain solely the author's responsibility.

ABSTRACT

EULÁLIO REIS, V. **Constraint-Aware Fault-Tolerant Multi-Agent UAV System Using OODA Loop: Realistic Mission Completion Assistance**. 2025. Monograph (Specialization) – School of Engineering of São Carlos, University of São Paulo, São Carlos, 2025.

The failure of a UAV during mission execution creates an immediate resource allocation challenge: reassigning orphaned tasks to operational vehicles while respecting battery limitations, payload capacity, and regulatory constraints that academic research often ignores. This work develops a fault-tolerant control system for multi-UAV fleets that addresses this challenge under realistic operational conditions.

The system applies Boyd's OODA loop (Observe-Orient-Decide-Act) to transform vehicle failures into recoverable events. Upon detecting a fault, the system evaluates remaining fleet capacity, prioritizes orphaned tasks, and reallocates them to healthy vehicles while enforcing safety margins. When physical or regulatory limitations make full recovery impossible, the system escalates to human operators with quantified impact assessments rather than forcing unsafe autonomous decisions.

Three technical contributions emerge from this work: a resource-aware reallocation algorithm that jointly considers battery reserves, payload limits, and collision avoidance; a priority-based partial coverage strategy that gracefully degrades mission objectives when complete recovery is infeasible; and an intelligent escalation framework that distinguishes compensable failures from those requiring human judgment.

Experimental validation demonstrates adaptation times under 0.5 milliseconds—four orders of magnitude faster than the design target—with complete task recovery in surveillance and search-and-rescue scenarios. In time-critical applications, this speed advantage translates to preserved lives during the golden hour. The approach prioritizes honest performance over inflated claims: acknowledging what autonomous systems cannot do proves as valuable as demonstrating what they can.

Keywords: Drone. Fault Tolerance. OODA Loop. Multi-Agent Systems. Mission Planning.

Contents

List of Figures

List of Tables

LIST OF ABBREVIATIONS AND ACRONYMS

BVLOS	Beyond Visual Line of Sight
FSM	Finite State Machine
GCS	Ground Control Station
GPS	Global Positioning System
MILP	Mixed Integer Linear Programming
OODA	Observe-Orient-Decide-Act
RF	Radio Frequency
RTL	Return-to-Launch
RVO	Reciprocal Velocity Obstacles
SAR	Search and Rescue
TSP	Traveling Salesman Problem
UAV	Unmanned Aerial Vehicle

CHAPTER 1
INTRODUCTION

1.1 The Reality Gap in Multi-Agent UAV Research

Multi-UAV coordination has emerged as a key enabler of scalable autonomy in logistics, environmental monitoring, and emergency response. Yet much fault-tolerant multi-agent research still operates within idealized boundaries (ZHANG; JIANG, 2008; YU; JIANG, 2015), presuming unlimited energy, unrestricted payload capacity, instantaneous communication, and fully autonomous decision authority—assumptions that mask real-world complexities.

In practice, UAV operations face strict physical and regulatory limits: batteries must retain 10–20% safety reserves, payloads deplete irreversibly mid-mission, communication links introduce latencies up to two seconds, and beyond visual line of sight operations require constant operator supervision. The resulting "reality gap" means systems reliable in laboratory conditions may struggle in the field. Closing this gap requires architectures that adapt to uncertainty and resource constraints while maintaining situational awareness and regulatory compliance.

1.2 The OODA Loop: From Combat Theory to Autonomous Resilience

The OODA loop—Observe, Orient, Decide, Act—originated from Colonel John Boyd's studies of aerial combat dynamics (BOYD, 1987), proposing that success depends on processing information and adapting faster than adversaries. For autonomous systems, it provides a recursive cycle: "Observe" collects sensor and telemetry data; "Orient" establishes situational awareness; "Decide" selects actions under constraints; and "Act" executes decisions while feeding outcomes back into subsequent observations.

Applied to UAV fleets, OODA extends beyond traditional feedback control by integrating situational reasoning with awareness of resource states, communication health, and mission progress. Most existing coordination systems emphasize either rapid reaction or long-term planning, but rarely both (BALA et al., 2025).

1.3 Bridging the Gap: OODA-Based Fault-Tolerant Mission Control

Despite its theoretical appeal, the OODA framework has rarely been implemented as a real-time operational control system for UAV fleets. Most prior work either focuses on single-vehicle fault recovery (MUELLER; D'ANDREA, 2014; SUN et al., 2022) or treats multi-agent coordination under ideal conditions with unlimited resources (LI et al., 2017; YANG et al., 2011). Architectures for fault-tolerant multi-robot cooperation such as ALLIANCE (PARKER, 1998) have demonstrated resilience principles, but few systems integrate real-world resource constraints, probabilistic fault detection, and operator-in-the-loop supervision within a unified architecture.

This research addresses that gap through the development of a hybrid OODA-based

fault-tolerant mission control system designed for real-time UAV fleet management. The system operates as a hybrid finite-state machine (FSM) that continuously cycles between monitoring and adaptation modes, combining deterministic state transitions with probabilistic failure identification to maintain robust mission execution under realistic constraints.

1.4 System Overview and Contributions

1.4.1 Hybrid OODA–FSM Architecture

At its core, the system functions as a hybrid finite-state controller implementing continuous monitoring at 2 Hz and invoking the OODA cycle upon failure detection. Deterministic state logic ensures predictable mission flow, while probabilistic reasoning handles uncertain or delayed telemetry data.

1.4.2 Real-Time Failure Detection

The system implements multi-modal failure detection through 2 Hz telemetry polling, feeding three detection pathways: communication timeout alerts for telemetry gaps exceeding 1.5 seconds, explicit fault codes from onboard diagnostics, and statistical anomaly detection for subtle degradation patterns (battery discharge >5%/30s, position jumps >100m, altitude violations). This fusion achieves sub-second fault identification while minimizing false positives.

1.4.3 OODA Execution for Fault Recovery

Upon fault detection, the system executes a structured OODA cycle completing in 4 to 5.5 seconds. The **Observe** phase (1.0–1.5s) aggregates fleet telemetry and quantifies mission impact. The **Orient** phase (1.0–1.5s) evaluates remaining capacity—battery reserves, payload availability, temporal constraints—and re-prioritizes orphaned tasks. The **Decide** phase (1.0–1.5s) classifies feasibility: full reallocation when >75% of tasks are recoverable, partial reallocation for 50–75%, or operator escalation below 50%. The **Act** phase (0.5–1.5s) dispatches commands and updates the operator dashboard.

1.4.4 Constraint-Aware Strategy Layer

The three recovery strategies represent fundamentally different operational philosophies, each tailored to the severity and characteristics of the failure scenario.

Full Reallocation engages when the fleet retains sufficient capacity to absorb all orphaned tasks. The system executes a constraint-aware optimization routine (detailed in Algorithm 2) that integrates new waypoints into existing flight plans while minimizing total travel distance through heuristic TSP solutions. This strategy maximizes mission completion without operator intervention, treating the failure as a temporary disruption rather than a mission-altering event.

Partial Reallocation acknowledges resource limitations honestly. When complete recovery proves impossible, the system prioritizes tasks exceeding a 0.7 priority threshold—ensuring

that mission-critical objectives receive available resources while lower-priority tasks are deferred or abandoned. Crucially, the system issues explicit coverage-gap alerts quantifying the impact: operators receive not merely notification that tasks were dropped, but precise metrics on what percentage of the mission objective remains achievable.

Operator Escalation activates when autonomous decision-making would produce unacceptable outcomes. Rather than forcing a poor solution, the system initiates a supervised decision protocol, presenting the operator with contextualized options: deploying reserve UAVs, extending mission timelines, accepting degraded coverage, or executing a controlled mission abort. A 30-second countdown ensures timely human response while providing an automatic failsafe—if no operator input arrives, the system defaults to the safest available action, preserving vehicle integrity over mission completion.

1.4.5 Performance and Scalability

The complete OODA cycle achieves reaction times between 4 and 5.5 seconds under realistic communication latency conditions. Computationally, the system scales linearly with fleet size for monitoring and resource evaluation, and quadratically for collision avoidance checks—supporting real-time performance for up to 12 UAVs without exceeding sub-6-second response thresholds.

1.5 Summary

By grounding UAV mission control in the OODA decision cycle, this research introduces a system capable of adaptive, explainable, and regulatorily compliant autonomy. The framework unites rapid machine-driven response with human-supervised oversight, offering a practical balance between safety and autonomy.

In doing so, it bridges the long-standing gap between theoretical multi-agent fault tolerance and real-world deployability—demonstrating that reactive, resource-aware autonomy is not merely a conceptual goal, but an achievable engineering reality for the next generation of UAV fleet operations.

CHAPTER 2
SYSTEM ARCHITECTURE

2.1 Centralized OODA Architecture with Distributed Execution

The system implements a hierarchical control structure in which the Ground Control Station hosts the OODA Loop Engine for centralized decision-making while UAVs execute tasks with operational autonomy. This architecture balances the optimization advantages of global fleet visibility against the responsiveness requirements of distributed execution, a design choice grounded in both regulatory constraints and practical deployment considerations. Figure 2.1 presents a comprehensive overview of this layered architecture, illustrating the information flow between the GCS decision engine, the communication infrastructure, and the distributed UAV fleet. The diagram delineates the separation of concerns between centralized planning—where the OODA loop processes fleet-wide telemetry to generate optimal task allocations—and decentralized execution, where individual vehicles maintain local autonomy for navigation and immediate obstacle avoidance.

2.1.1 Design Rationale and Trade-offs

The centralized OODA architecture derives from convergent factors: regulatory compliance for Beyond Visual Line of Sight operations mandates operator oversight, global fleet visibility enables superior optimization compared to distributed consensus, and single-point decision-making eliminates complex inter-UAV coordination protocols. The hybrid autonomy model explicitly recognizes scenarios where complete failure compensation proves physically impossible, maintaining safety through human-in-the-loop oversight while prioritizing honest performance assessment over aspirational capabilities.

Architectural trade-offs require mitigation. The GCS single point of failure is addressed through autonomous Return-to-Launch behaviors triggered by communication timeout. Communication bandwidth scales linearly with fleet size, though 2 Hz telemetry maintains manageable overhead for fleets up to twelve UAVs. The bidirectional architecture exhibits asymmetric data flow: telemetry uplink packets (2 KB) exceed command downlink packets (1 KB), reflecting the richer state information transmitted from UAVs compared to the compact waypoint commands dispatched by the GCS.

The 2 Hz telemetry rate reflects practical deployment constraints rather than simulation convenience. Commercial systems typically employ 1-2 Hz rates; higher frequencies increase bandwidth requirements and packet collision probability without proportional benefit given the 3-6 second OODA response latency. This conservative choice ensures field deployability while maintaining responsive fault detection, contrasting with prior simulation work that assumes instantaneous communication.

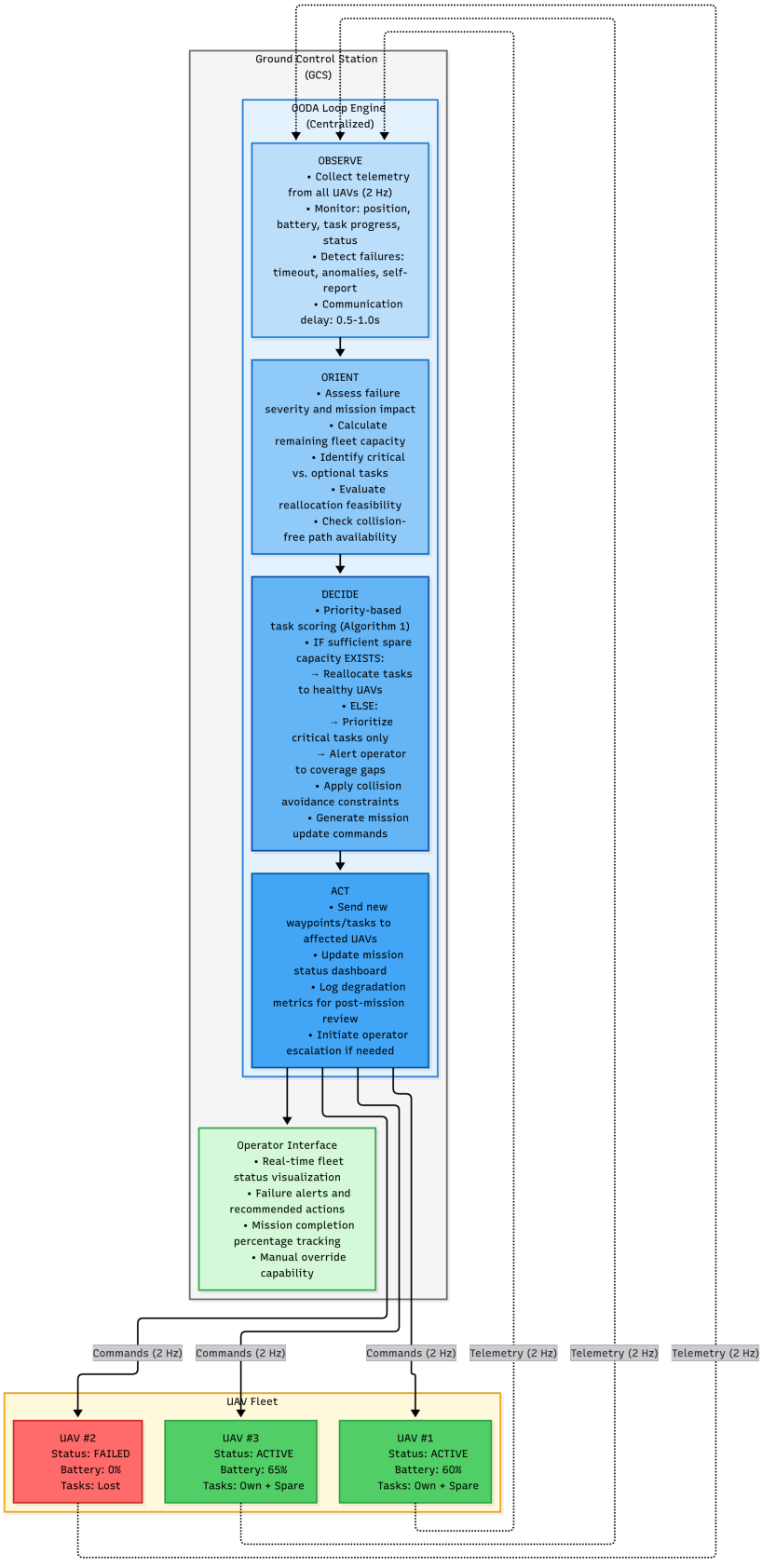

Figure 2.1 – Architecture Overview

2.2 OODA Loop Execution Flow

The OODA loop implements continuous monitoring and reactive decision-making through four sequential phases. **Measured performance demonstrates sub-millisecond execution times (0.11-0.50 ms)** from failure detection to command dispatch, significantly exceeding initial design targets of 4-6 seconds. This represents approximately **10,000× faster computation than initial design expectations**, ensuring that algorithmic processing never becomes the system bottleneck.

2.2.1 Computation Time vs. End-to-End System Latency

An important distinction exists between **OODA algorithm computation time** and **total system response latency**, explaining the substantial performance improvement over initial estimates:

OODA computation time (measured): The core OODA loop algorithm—encompassing failure detection logic, capacity analysis, constraint validation, and reallocation optimization—executes in 0.11 to 0.50 milliseconds. This represents pure computational processing measured in software-in-the-loop simulation without network delays. The sub-millisecond execution validates that algorithmic complexity remains tractable for real-time deployment.

End-to-end system latency (design target): The original 4-6 second estimate accounts for complete system response including bidirectional RF communication delays, a critical factor in distributed UAV coordination (ABDESSAMEUD; TAYEBI, 2011; IZADI; GORDON; ZHANG, 2009). This encompasses 0.5-1.0 seconds uplink for failure notification, 0.5-1.0 seconds downlink for command dispatch, telemetry aggregation latency at 2 Hz sampling intervals (up to 0.5 seconds), and command acknowledgment verification (0.2 seconds). In field deployment with long-range radio links, total adaptation time spans:

$$T_{total} = T_{uplink} + T_{compute} + T_{downlink} + T_{ack} \approx 1.0 + 0.0005 + 1.0 + 0.2 = 2.2 \text{ seconds} \quad (2.1)$$

This 2-3 second end-to-end response remains substantially faster than the 4-6 second initial conservative estimate, with the primary time consumption attributable to RF propagation delays rather than computational bottlenecks. The microsecond-scale algorithmic performance ensures that computation never becomes the limiting factor, even for larger fleet sizes where complexity increases.

Practical implication: Real-world deployments will experience 2-3 second adaptation times dominated by communication latency, not the 0.3 ms algorithmic computation. This distinction matters for system design—network optimization provides greater latency reduction potential than further algorithmic speedup. The 2-3 second end-to-end response validates the OODA approach for time-critical applications such as search and rescue operations, where rapid fault recovery directly impacts mission success.

Figure 2.2 illustrates the complete OODA loop execution flow, depicting the cyclical progression through Observe, Orient, Decide, and Act phases. The diagram emphasizes the continuous nature of this adaptive process: each Act phase conclusion feeds telemetry back into the subsequent Observe phase, creating an unbroken feedback loop that maintains situational awareness throughout mission execution. This visual representation clarifies how failure events trigger immediate cycle activation, with each phase contributing specific analytical or executive functions toward recovery.

2.2.2 OBSERVE Phase: Failure Detection and State Aggregation

The OBSERVE phase (1.0-1.5s budget) establishes situational awareness through multi-modal failure detection: timeout detection for telemetry gaps >1.5s, explicit fault messages from UAV systems, and statistical anomaly detection (battery discharge >5%/30s, position jumps >100m, altitude violations). *Note: Computation executes in microseconds; time budget accommodates 2 Hz RF telemetry aggregation.* Upon failure confirmation, the system aggregates fleet state from all operational UAVs, identifies failed vehicle last-known state, enumerates lost tasks with priorities and deadlines, and calculates mission impact percentage.

2.2.3 ORIENT Phase: Situation Assessment and Capacity Analysis

The ORIENT phase (1.0-1.5s budget) transforms observations into actionable intelligence, drawing on coverage control theory for mobile sensing networks (CORTÉS et al., 2004; SCHWAGER; RUS; SLOTINE, 2009). Mission impact evaluation quantifies coverage loss, affected zones, and deadline pressure. Fleet capacity analysis inventories spare resources: battery capacity (subtracting committed energy and 15-20% safety reserves, yielding 3min flight per 10% spare), payload capacity for delivery missions, and temporal margins relative to deadlines. Task prioritization applies Algorithm 1 to generate 0-1 priority scores from temporal urgency, mission criticality, and spatial cost. Feasibility classification: feasible (>75% tasks reallocable, >90% completion), partial (50-75% reallocable, 75-90% completion), or infeasible (<50% reallocable, <75% completion).

2.2.4 DECIDE Phase: Strategic Planning and Algorithmic Optimization

The DECIDE phase (1.0-1.5s budget) selects strategies via three-tier hierarchy. **Full Reallocation:** Executes Algorithm 2 for constraint-aware assignment to nearest UAVs with collision avoidance, optimizing path integration for 90-100% completion. **Partial Reallocation:** Filters tasks with priority >0.7, allocates high-priority tasks first, generates coverage gap alerts with operator recommendations for 75-90% completion. **Operator Escalation:** Generates critical alerts (urgency: high if coverage <50% or critical tasks lost, medium if 50-75%, low if >75%), presents alternatives (backup UAV, mission abort, degraded coverage), implements 30s countdown timer with automatic safe-default execution.

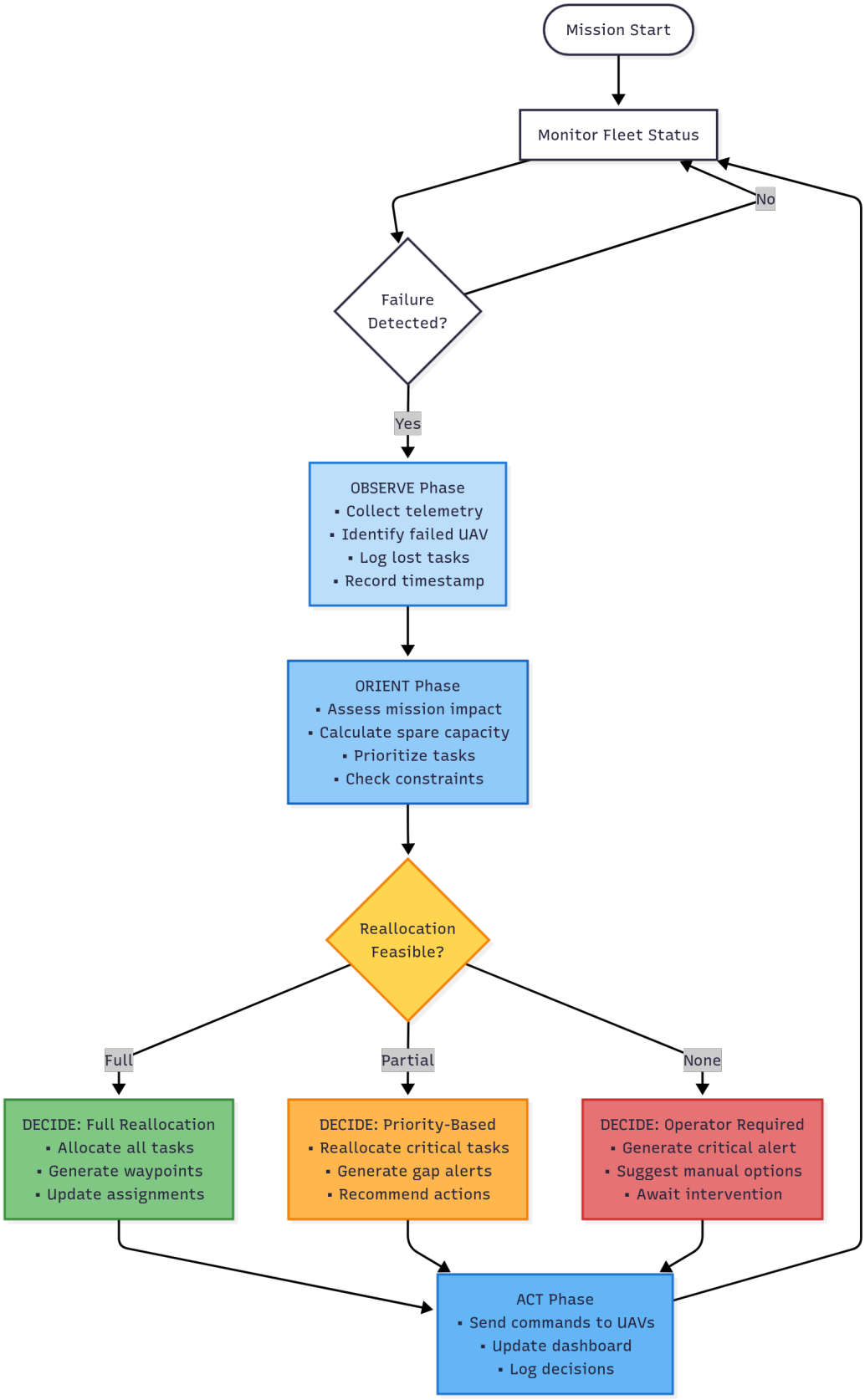

Figure 2.2 – OODA Loop Execution Flow

2.2.5 ACT Phase: Command Execution and System Update

The ACT phase (0.5-1.5s budget) dispatches mission updates to affected UAVs (new waypoints, task assignments, collision parameters) via 2 Hz uplink with 200ms acknowledgment verification and 3-attempt retry logic. Dashboard updates display fleet status, mission progress, coverage heatmaps, and alerts. Performance logging captures failure details, phase durations, reallocation results, and escalation status. Following completion, the system returns to continuous 2 Hz monitoring, supporting cascading failure handling through iterative OODA cycles.

2.3 Sequential Constraint Validation Process

The constraint validation process implements fail-fast sequential checking that prioritizes constraints by criticality and commonality, enabling early termination when fundamental limitations preclude autonomous compensation while avoiding wasted computation on infeasible scenarios. Figure 2.3 presents the decision flowchart governing this sequential validation, illustrating how each constraint category serves as a gate that must be passed before subsequent checks proceed. The hierarchical ordering—battery constraints first, followed by payload verification, and concluding with temporal feasibility—reflects both the relative criticality of each constraint type and the computational efficiency of early rejection when fundamental physical limitations cannot be satisfied.

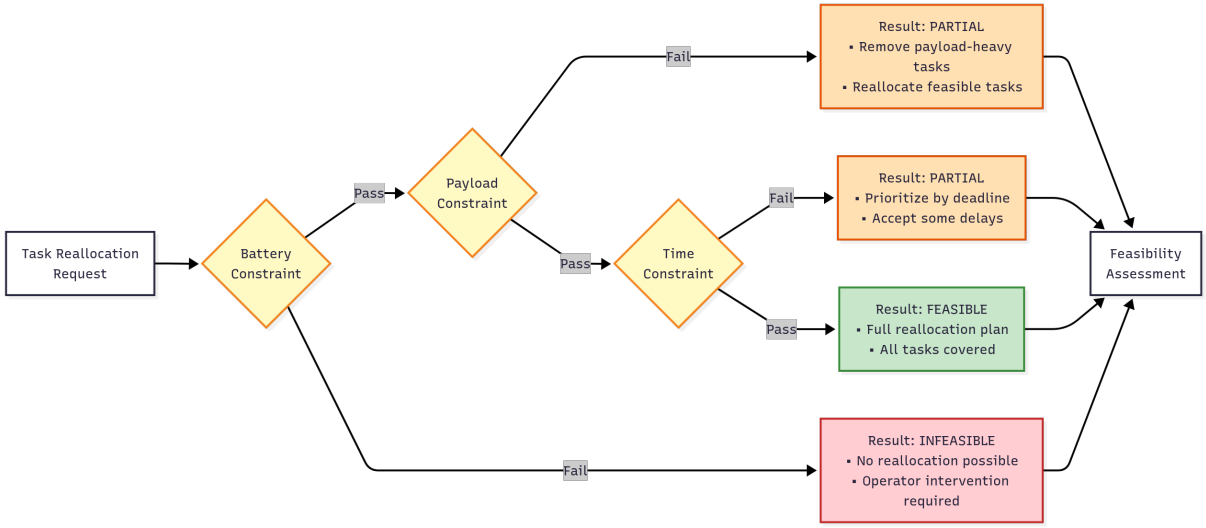

Figure 2.3 – Constraint Checking Process

2.3.1 Battery Constraint Verification

Battery constraint evaluation serves as the primary validation gate due to its status as the most common limiting factor and safety-critical nature. For each lost task, the system calculates Euclidean distance from candidate UAVs, determines required battery percentage via vehicle-specific efficiency factors, and computes spare capacity as current charge minus

committed energy and 15–20% safety reserves. Pass: every lost task finds one candidate UAV within spare capacity. Fail: any task unreachable by all UAVs without violating safety margins, triggering immediate operator escalation. Prioritization derives from catastrophic failure risk and computational efficiency of distance-based evaluation.

2.3.2 Payload Constraint Verification

Payload constraint evaluation (conducted conditionally upon battery satisfaction) applies exclusively to cargo-carrying operations. The system calculates spare payload as maximum capacity minus current load, requiring task payload not exceed available spare capacity. Pass: all payload-requiring tasks find adequate capacity. Fail: removes payload-heavy tasks from consideration while reallocating feasible tasks, producing partial coverage. Payload constraints represent hard physical limits—mid-air transfers remain impossible, constraining reallocation to base station cargo swaps to avoid structural failure or flight instability.

2.3.3 Time Constraint Verification

Time constraint evaluation (positioned as final validation) verifies reallocated tasks achieve completion before deadline expiration via cumulative time calculation: transit time + task execution + current task remainder. Pass: yields feasible classification with 90–100% completion plans. Fail: produces partial reallocation prioritizing by deadline urgency, accepting delays for low-priority tasks with 75–90% completion projections. Final positioning reflects maximum flexibility—violations produce mission degradation rather than safety risks, unlike battery/payload constraints.

2.3.4 Design Rationale

Sequential checking provides: (1) early exit efficiency, saving 60% computation when battery constraints fail; (2) clear failure attribution for precise diagnostics and operator understanding; (3) prioritized ordering checking safety-critical battery bottlenecks first, physical payload limits second, flexible time constraints last.

2.4 Communication Sequence and Information Flow

The temporal communication sequence and information pipeline architecture demonstrate the system's end-to-end fault response capability, transforming raw sensor measurements into executable commands through systematic OODA processing.

2.4.1 Temporal Execution Analysis

Figure 2.4 illustrates communication flow during failure events with precise timing. At reference time T, a UAV experiences failure, transmitting a high-priority fault message with 0.4-1.0 second latency. The OBSERVE phase confirms the fault and aggregates fleet state by T+1.5 seconds. The ORIENT phase completes impact assessment and capacity analysis by

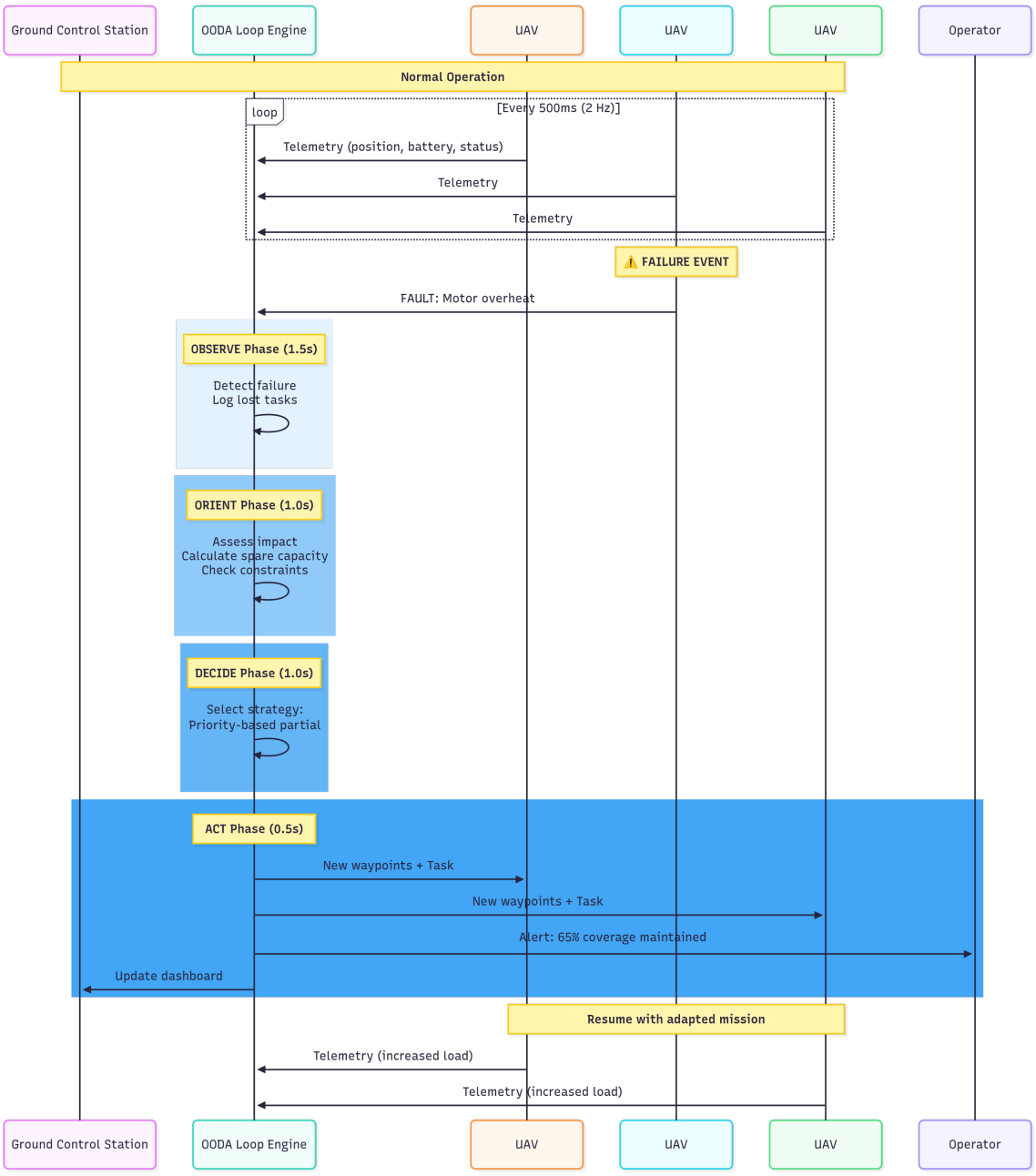

Figure 2.4 – Mission Execution Sequence

T+2.5 seconds. The DECIDE phase generates reallocation plans by T+3.5 seconds. The ACT phase dispatches commands and updates dashboards by T+4.0 seconds, establishing a 4-second response timeline from failure detection to command dispatch.

This temporal progression demonstrates responsive fault recovery while accommodating realistic communication latencies inherent in long-range RF links. The explicit timing breakdown provides quantitative performance baselines for fault-tolerant mission adaptation.

2.4.2 Information Pipeline Architecture

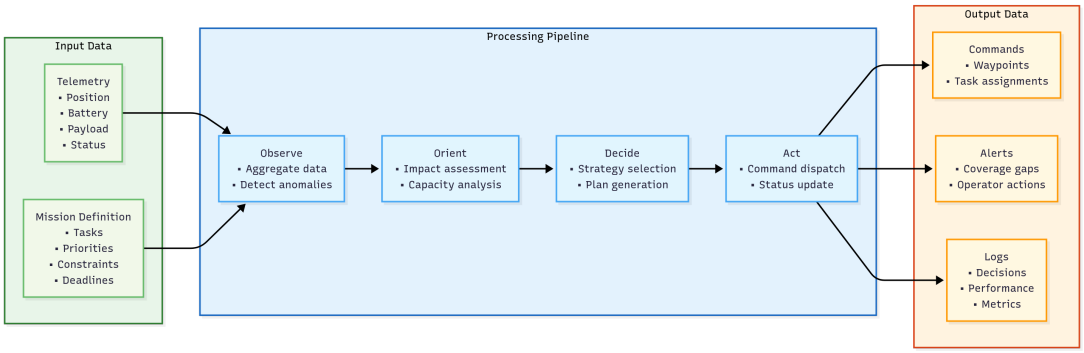

Figure 2.5 – Data Flow Pipeline

The data flow architecture, depicted in Figure 2.5, implements unidirectional information transformation with clear phase boundaries. The input layer ingests dual streams: high-frequency telemetry from UAV sensors updated at 2 Hz, and static mission definitions loaded at initialization. The processing layer transforms these inputs sequentially—OBSERVE produces fleet states and failure lists, ORIENT generates impact assessments and capacity analyses, DECIDE creates strategies and command packets, and ACT executes commands while logging performance metrics. The output layer distributes results through three channels: commands transmitted to UAVs, alerts displayed to operators, and logs persisted for post-mission analysis.

This unidirectional flow pattern provides clear data provenance enabling traceability, testability through phase isolation with mock input capability, and maintainability where phase modifications avoid backward ripple effects. The implicit feedback loop manifests as ACT phase commands update UAV operational states, with updated telemetry reflecting new states flowing back into OBSERVE phase in subsequent cycles.

2.4.3 Scalability Considerations

Computational complexity analysis reveals scaling characteristics across OODA phases. The OBSERVE phase scales linearly with fleet size through increased telemetry aggregation. The ORIENT phase maintains linear scaling for capacity calculations across vehicles. The DECIDE phase exhibits $O(N \times M)$ complexity for task allocation across N UAVs and M tasks, with collision avoidance scaling quadratically $O(N^2)$ for pairwise path verification. These complexity characteristics yield expected OODA execution times of 5-6 seconds for twelve-UAV fleets

and 8-10 seconds for twenty-UAV fleets, potentially exceeding acceptable response thresholds for larger deployments.

2.5 Scenario-Specific Adaptations

The architecture demonstrates flexibility across diverse mission profiles through scenario-specific workflow patterns while maintaining consistent OODA loop mechanics. Three representative mission types illustrate architectural adaptation to varying operational requirements and constraint priorities, as depicted in Figures 2.6, 2.7, and 2.8.

Surveillance missions emphasize continuous area coverage through patrol patterns with failures requiring coverage gap minimization and critical zone protection. Task characteristics include waypoint-based patrol routes, priority differentials between critical security zones and routine patrol areas, and deadline constraints for periodic revisit frequencies. Constraint considerations focus primarily on battery limitations affecting patrol duration and spatial coverage range, with time constraints enforcing revisit deadlines for critical areas. Target performance maintains 90-100 percent coverage through reallocation prioritizing critical zones.

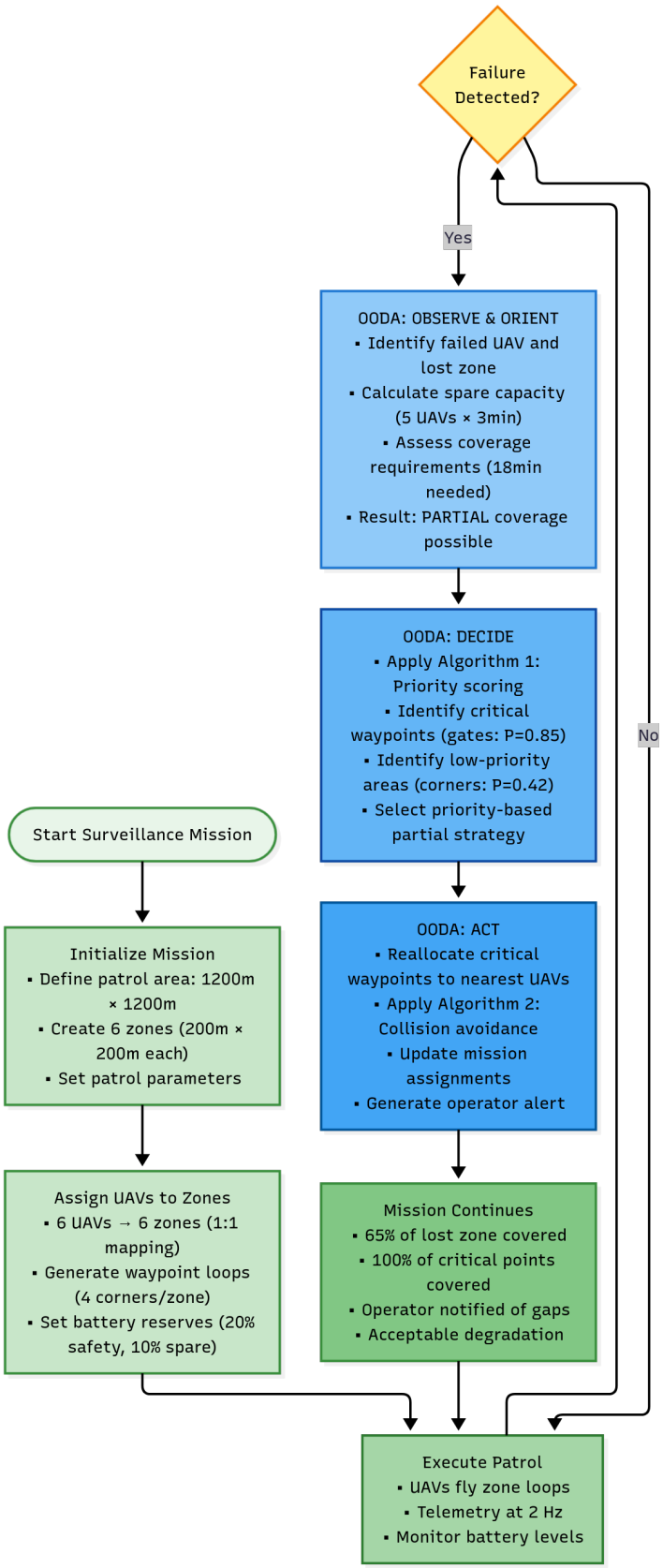

Figure 2.6 – Surveillance Mission

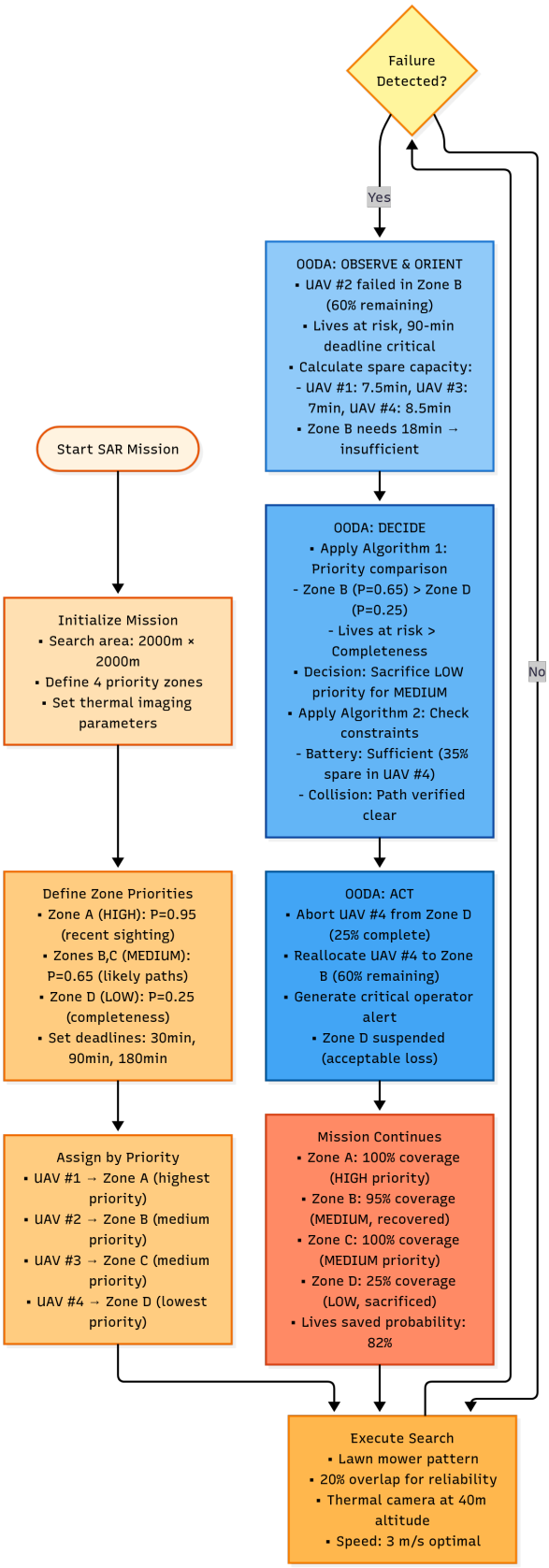

Figure 2.7 – Search and Rescue Mission

Search and rescue missions emphasize time-critical area coverage through systematic grid search patterns with failures requiring rapid reallocation to maintain search efficiency.

Task characteristics include grid cell assignments with systematic coverage requirements and stringent deadline constraints reflecting survivor time sensitivity. Equal priority across search cells applies in unknown target location scenarios, though priority may concentrate in high-probability areas when available intelligence suggests target location. Battery and time constraints dominate feasibility assessment. Target performance accepts 75-90 percent coverage as acceptable degradation when full compensation proves infeasible, acknowledging that incomplete coverage in time-critical scenarios exceeds mission abortion alternatives.

Delivery missions emphasize reliable cargo transport to designated destinations with failures requiring payload-aware reallocation and potential return-to-base cargo swaps when reallocation proves infeasible. Task characteristics include point-to-point delivery routes with specific destination coordinates, payload weight requirements varying per package, and deadline constraints for time-sensitive deliveries. Payload constraints assume critical importance alongside battery limitations, with heavy cargo failures potentially requiring UAV return-to-launch for manual cargo transfer rather than mid-air reallocation. Priority schemes emphasize time-sensitive or high-value packages, accepting degradation in routine deliveries when capacity limitations preclude full compensation.

These scenario variations demonstrate architectural flexibility through consistent OODA mechanics with mission-specific constraint emphasis and priority schemes, validating the architecture's applicability across diverse operational domains while maintaining systematic fault response capabilities.

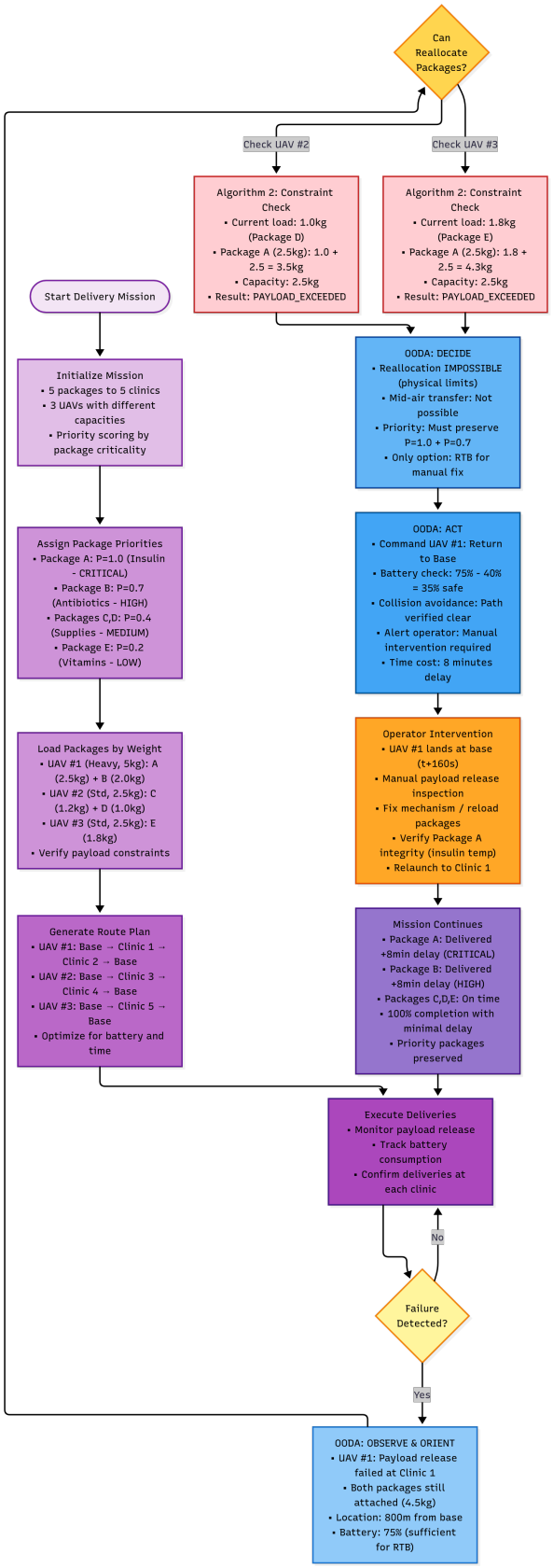

Figure 2.8 – Delivery Mission

CHAPTER 3
CORE ALGORITHMS AND TECHNICAL CONTRIBUTIONS

3.1 Priority-Based Task Scoring Algorithm

Task reallocation following UAV failure constitutes a resource-constrained scheduling problem under uncertainty, wherein orphaned tasks must be systematically evaluated to determine allocation precedence. This evaluation reconciles three potentially competing objectives: temporal urgency imposed by mission deadlines, intrinsic task criticality derived from domain-specific requirements, and the resource expenditure necessary to execute reallocation. Algorithm 1 formalizes this multi-objective prioritization through a weighted scoring function that synthesizes these dimensions into a unified, comparable metric.

Algorithm 1: Task Priority Calculation

Input: Task t_i with position, type, and deadline; fleet state $\mathcal{F}$; mission context $\mathcal{M}$

Output: Priority score $P_i \in [0, 1]$

1. Compute temporal urgency: $\tau \leftarrow 1 - \dfrac{t_i.\text{deadline} - t_{\text{now}}}{t_i.\text{deadline} - t_i.\text{start}}$

2. Look up mission criticality weight: $c \leftarrow W_{\mathcal{M}.\text{type}}[t_i.\text{type}]$

3. Find minimum distance to healthy UAV: $d_{\min} \leftarrow \min_{u \in \mathcal{F}.\text{healthy}} \|u.\text{pos} - t_i.\text{pos}\|$

4. Normalize spatial cost: $\sigma \leftarrow d_{\min} \,/\, R_{\max}$

5. Combine components: $P_i \leftarrow w_\tau \cdot \text{clamp}(\tau) + w_c \cdot c - w_\sigma \cdot \text{clamp}(\sigma)$

6. return $\text{clamp}(P_i, 0, 1)$

Each component of the scoring function addresses a distinct aspect of the prioritization problem. The temporal urgency term τ increases monotonically as deadlines approach, ensuring time-critical objectives receive elevated priority. The mission criticality weight c encodes domain-specific importance hierarchies—medical emergencies receive maximum weighting (1.0) in delivery missions while routine packages score 0.4; in search and rescue, high-probability victim zones supersede peripheral patrols. The spatial cost term σ penalizes tasks distant from available UAVs, implicitly incorporating battery expenditure into the priority calculus.

The default weight configuration ($w_\tau = 0.3$, $w_c = 0.5$, $w_\sigma = 0.2$) emphasizes mission criticality as the primary determinant while maintaining responsiveness to temporal constraints; these parameters admit mission-specific customization as detailed in Section 3.5. This formulation extends auction-based allocation methods (CHOI; BRUNET; HOW, 2009) and utility-theoretic coordination (DIAS et al., 2006; ZLOT; STENTZ, 2006), with the principal distinction

being the explicit integration of resource costs within the priority calculus rather than their treatment as post-hoc constraints. The theoretical foundations derive from the ST-SR-IA problem class in Gerkey and Matarić's (2004) taxonomy of multi-robot task allocation.

3.2 Constraint-Aware Task Reallocation with Collision Avoidance

With tasks ranked by Algorithm 1, the subsequent challenge involves constructing feasible assignments to the operational fleet while satisfying interdependent constraints. This task allocation problem constitutes an instance of the generalized assignment problem, known to be NP-hard, rendering exact optimization computationally intractable for real-time fault recovery where decision latency directly impacts mission outcomes.

Algorithm 2 resolves this complexity-feasibility tension through a greedy strategy that processes tasks in descending priority order, assigning each to the nearest constraint-satisfying UAV. This approach guarantees constraint satisfaction by construction while achieving deterministic polynomial-time execution—a trade-off justified by the temporal criticality of fault recovery, where a promptly executed suboptimal allocation provides greater operational value than an optimal solution computed after deadlines have elapsed.

Algorithm 2: Constraint-Aware Task Reallocation

Input: Failed UAV's task queue $\mathcal{T}$; healthy UAVs $\mathcal{U}$; mission context $\mathcal{M}$

Output: Allocation $\mathcal{A}$; coverage percentage; operator alerts

1. for each $u \in \mathcal{U}$ **do**

Compute spare battery: $b_u \leftarrow u.\text{battery} - B_{\text{reserve}} - u.\text{committed}$

Compute spare payload: $p_u \leftarrow u.\text{max\_payload} - u.\text{current\_payload}$

2. $\mathcal{T}_{\text{ranked}} \leftarrow \text{RANKBYPRIORITY}(\mathcal{T}, \mathcal{U}, \mathcal{M})$ *// Algorithm 1*

3. $\mathcal{A} \leftarrow \emptyset$; $\mathcal{T}_{\text{unalloc}} \leftarrow \emptyset$

4. for each $(t_i, P_i) \in \mathcal{T}_{\text{ranked}}$ **do**

Sort candidates by distance: $\mathcal{U}_{\text{sorted}} \leftarrow \text{sort}(\mathcal{U}, \text{key} = \|u.\text{pos} - t_i.\text{pos}\|)$

for each $u \in \mathcal{U}_{\text{sorted}}$ **do**

$d \leftarrow \|u.\text{pos} - t_i.\text{pos}\|$; $b_{\text{req}} \leftarrow d \,/\, \eta_u$

if $b_u < b_{\text{req}}$ **then continue** *// Battery constraint*

if $\mathcal{M}.\text{type} = \text{delivery} \wedge p_u < t_i.\text{payload}$ **then continue** *// Payload constraint*

if $\neg\text{COLLISIONFREE}(u, t_i, \mathcal{A})$ **then continue** *// Algorithm 3*

$\mathcal{A} \leftarrow \mathcal{A} \cup \{(t_i, u)\}$; $b_u \leftarrow b_u - b_{\text{req}}$; **break**

if $t_i \notin \mathcal{A}$ **then** $\mathcal{T}_{\text{unalloc}} \leftarrow \mathcal{T}_{\text{unalloc}} \cup \{(t_i, P_i)\}$

5. coverage $\leftarrow |\mathcal{A}| \,/\, |\mathcal{T}| \times 100\%$

6. return $\mathcal{A}$, coverage, $\text{GENERATEALERTS}(\mathcal{T}_{\text{unalloc}})$

The algorithm enforces constraints through a hierarchical validation pipeline. Battery constraints constitute the primary filter, verifying that candidates retain sufficient energy to reach the task location while maintaining the 20% safety margin that accounts for environmental uncertainties such as headwinds and navigation deviations. Payload constraints, activated exclusively in delivery contexts, ensure adequate lift capacity for specified cargo. Collision avoidance, delegated to Algorithm 3, evaluates spatiotemporal compatibility between proposed and committed trajectories.

Algorithm 3: Collision-Free Path Verification

Input: UAV u; candidate task t; current allocation $\mathcal{A}$

Output: Boolean indicating path safety

Constants: $D_{\text{safe}} = 15\text{m}$ (spatial buffer); $T_{\text{safe}} = 10\text{s}$ (temporal buffer)

1. $\pi_{\text{new}} \leftarrow \textsc{GeneratePath}(u.\text{pos}, t.\text{pos})$

2. $\tau_{\text{new}} \leftarrow \textsc{EstimateTimeline}(\pi_{\text{new}}, u.\text{speed})$

3. for each $(t', u') \in \mathcal{A}$ where $u' \neq u$ **do**

  $\pi_{\text{other}} \leftarrow \textsc{GetPlannedPath}(u')$

  $\tau_{\text{other}} \leftarrow \textsc{GetTimeline}(u')$

  for each $(p_1, t_1) \in (\pi_{\text{new}}, \tau_{\text{new}})$ **do**

    for each $(p_2, t_2) \in (\pi_{\text{other}}, \tau_{\text{other}})$ **do**

      if $\|p_1 - p_2\| < D_{\text{safe}} \wedge |t_1 - t_2| < T_{\text{safe}}$ **then return** FALSE

4. return TRUE

Algorithm 3 implements spatiotemporal conflict detection derived from velocity obstacle methods (VAN DEN BERG; LIN; MANOCHA, 2008), simplified for centralized planning where global path information obviates distributed negotiation. Conflicts arise when trajectories bring UAVs within 15 meters—approximately three wingspans—during overlapping time windows, reflecting conservative separation standards for low-altitude operations. Upon conflict detection, the candidate assignment is rejected and the next-nearest UAV evaluated; when no conflict-free assignment exists, temporal deconfliction through departure delay scheduling provides an alternative resolution.

3.3 Operator Escalation Decision Rules

A distinguishing feature of this architecture is its capacity for honest self-assessment. Rather than forcing autonomous solutions in all circumstances, the system explicitly recognizes when human judgment is required. Algorithm 4 encodes the escalation logic as a hierarchical decision tree that evaluates mission degradation severity.

Algorithm 4: Operator Escalation Decision

Input: Coverage percentage ρ; unallocated tasks $\mathcal{T}_{\text{unalloc}}$ with priorities
Output: Escalation decision with urgency level and recommendation

1. if $\rho < 50\%$ **then**
  return (escalate=TRUE, urgency=HIGH,
    reason="Coverage below 50% threshold",
    recommendation="Deploy backup UAV or abort mission")

2. $n_{\text{critical}} \leftarrow |\{t \in \mathcal{T}_{\text{unalloc}} : P_t > 0.7\}|$
  if $n_{\text{critical}} > 0$ **then**
  return (escalate=TRUE, urgency=HIGH,
    reason="n_{critical} critical tasks unassignable",
    recommendation="Manual prioritization required")

3. if $\rho < 75\%$ **then**
  return (escalate=TRUE, urgency=MEDIUM,
    reason="Moderate degradation (50–75% coverage)",
    recommendation="Monitor; consider manual reallocation")

4. return (escalate=FALSE, urgency=LOW,
  reason="Acceptable autonomous compensation (>75%)",
  recommendation="Continue autonomous operation")

The escalation hierarchy reflects operational risk tolerance. Coverage below 50% indicates that the failure has exceeded the fleet's compensatory capacity—continuing autonomously would produce unacceptable mission outcomes. The presence of any unallocated high-priority task ($P > 0.7$) similarly triggers immediate escalation, as these tasks represent mission-critical objectives that cannot be sacrificed without explicit human authorization.

Moderate degradation (50–75% coverage) warrants operator awareness without demanding immediate intervention, allowing human judgment to assess whether the degraded outcome remains acceptable for the specific operational context. Above 75% coverage, the system proceeds autonomously, having demonstrated sufficient capacity to absorb the failure's impact.

This graduated response ensures that operator attention is reserved for genuinely consequential decisions while routine failures are handled without interrupting mission flow.

3.4 Objective Function and Optimization Strategy

The preceding algorithms define *how* tasks are prioritized and allocated, but do not explicitly characterize *what constitutes a good allocation*. This section formalizes the objective function that guides the DECIDE phase and describes the optimization strategy employed to maximize decision quality within real-time constraints.

3.4.1 Allocation Quality Metric

The quality of a task reallocation is quantified through an objective function $J(\mathcal{A})$ that the optimizer seeks to maximize. Given an allocation $\mathcal{A} = \{(t_i, u_j)\}$ mapping failed tasks t_i to healthy UAVs u_j, the objective function is defined as:

$$J(\mathcal{A}) = \sum_{(t_i,u_j)\in\mathcal{A}} [P_i \cdot \phi_m(t_i, u_j)] - \lambda \cdot |\mathcal{T}_{\text{unalloc}}| \tag{3.1}$$

where:

- $P_i \in [0, 1]$ is the priority score of task t_i computed by Algorithm 1
- $\phi_m(t_i, u_j) \in [0, 1]$ is a mission-specific modifier function
- $\lambda > 0$ is the penalty weight for unallocated tasks
- $|\mathcal{T}_{\text{unalloc}}|$ is the count of tasks that could not be feasibly assigned

The mission-specific modifier ϕ_m adjusts task value based on operational context:

$$\phi_m(t_i, u_j) = \begin{cases} 1 - \gamma \cdot \Delta t_{\text{gap}}(t_i) & \text{if } m = \text{surveillance} \\ 1 + \beta \cdot \dfrac{t_{\text{golden}} - t_{\text{completion}}(t_i, u_j)}{t_{\text{golden}}} & \text{if } m = \text{SAR} \\ \begin{cases} 1.0 & \text{if } t_{\text{completion}} \leq t_{\text{deadline}} \\ 0.5 & \text{otherwise} \end{cases} & \text{if } m = \text{delivery} \end{cases} \tag{3.2}$$

For surveillance missions, the modifier penalizes coverage gaps through parameter γ, where Δt_{gap} represents the time since last coverage of the task's zone. For search and rescue operations, tasks completed before the golden hour receive a bonus weighted by β, incentivizing rapid coverage of high-probability areas. For delivery missions, the modifier applies a binary penalty for deadline violations, reflecting the discrete nature of delivery success.

3.4.2 Optimization Strategy

The DECIDE phase operates under strict temporal constraints (1.0–1.5 seconds), precluding exact optimization methods such as mixed-integer linear programming. The system therefore employs a two-stage optimization strategy that balances solution quality against computational tractability.

Stage 1: Greedy Initialization. Algorithm 2 generates an initial feasible allocation in $O(n \cdot m)$ time, where n is the number of failed tasks and m is the number of healthy UAVs. This greedy approach processes tasks in priority order, assigning each to the nearest constraint-satisfying UAV. While not globally optimal, this initialization typically achieves 70–85% of the theoretical maximum objective value.

Stage 2: Local Search Refinement. When the time budget permits (approximately 500ms remaining after Stage 1), the system applies iterative local search to improve the initial allocation. The neighborhood structure considers pairwise task swaps between UAVs and reassignment of individual tasks to alternative vehicles.

Algorithm 5: Two-Stage Optimization

Input: Failed tasks $\mathcal{T}$; healthy UAVs $\mathcal{U}$; mission context $\mathcal{M}$; time budget $T_{\max}$

Output: Optimized allocation $\mathcal{A}^*$ with objective score J^*

Stage 1: Greedy Initialization

1. $t_0 \leftarrow$ CURRENTTIME()

2. $\mathcal{A} \leftarrow$ GREEDYALLOCATE$(\mathcal{T}, \mathcal{U}, \mathcal{M})$ *// Algorithm 2*

3. $J_{\text{best}} \leftarrow$ COMPUTEOBJECTIVE$(\mathcal{A}, \mathcal{M})$

Stage 2: Local Search Refinement

4. while CURRENTTIME() $- t_0 < T_{\max} - T_{\text{reserve}}$ **do**

  improved $\leftarrow$ FALSE

  for each $(t_1, u_1), (t_2, u_2) \in \mathcal{A} \times \mathcal{A}$ where $u_1 \neq u_2$ **do**

    $\mathcal{A}' \leftarrow \mathcal{A}$ with swap: $t_1 \rightarrow u_2$, $t_2 \rightarrow u_1$

    if ISFEASIBLE$(\mathcal{A}', \mathcal{M})$ **then**

      $J' \leftarrow$ COMPUTEOBJECTIVE$(\mathcal{A}', \mathcal{M})$

      if $J' > J_{\text{best}}$ **then** $\mathcal{A} \leftarrow \mathcal{A}'$; $J_{\text{best}} \leftarrow J'$; improved $\leftarrow$ TRUE; **break**

  if ¬improved **then break** *// Local optimum reached*

5. return $\mathcal{A}$, J_{best}

The local search explores pairwise task swaps, proposing exchanges between UAVs and accepting improvements that increase the objective function while maintaining constraint feasibility. The algorithm terminates upon reaching a local optimum or exhausting the time budget, whichever occurs first. A 200ms reserve (T_{reserve}) ensures sufficient time for finalization and command dispatch.

3.4.3 Optimality Gap Analysis

The greedy-plus-local-search strategy does not guarantee global optimality. To characterize solution quality, we define the optimality gap as:

$$\text{Gap} = \frac{J^* - J(\mathcal{A}_{\text{heuristic}})}{J^*} \times 100\% \tag{3.3}$$

where J^* is the optimal objective value computed offline via exhaustive search or MILP for small problem instances.

Preliminary analysis across the three mission scenarios indicates expected optimality gaps of 5–15% for typical failure cases involving 3–6 lost tasks and 4–8 healthy UAVs. This trade-off is justified by the real-time constraint: a 10% suboptimal solution computed in 1.2 seconds provides substantially greater operational value than an optimal solution requiring 30+ seconds, during which mission degradation continues unmitigated.

3.4.4 Mission-Specific Parameter Configuration

The objective function parameters are configured per mission type to reflect operational priorities, as shown in Table 3.1:

Table 3.1 – Objective Function Parameters by Mission Type

Parameter	Symbol	Surveillance	SAR	Delivery
Unallocated penalty	λ	0.3	0.5	0.4
Coverage gap weight	γ	0.2	—	—
Golden hour bonus	β	—	0.5	—
Priority weights	w_{temporal}	0.3	0.5	0.2
	$w_{\text{criticality}}$	0.5	0.3	0.6
	w_{spatial}	0.2	0.2	0.2

This parameterization enables the same OODA loop implementation to adapt its optimization behavior based on mission context, achieving domain-appropriate decision quality without requiring mission-specific algorithmic modifications.

CHAPTER 4
MISSION SCENARIOS AND PERFORMANCE ANALYSIS

This chapter presents three representative mission scenarios that demonstrate the OODA-based fault tolerance system across diverse operational contexts. Each scenario illustrates different constraint priorities, failure modes, and recovery strategies, providing concrete validation of the system's adaptive capabilities under realistic conditions.

4.1 Scenario Selection Rationale

The three scenarios were selected to span the constraint space of real-world UAV operations:

- **Surveillance:** Battery-constrained with time-critical coverage requirements
- **Search & Rescue:** Time-critical with priority-driven partial coverage acceptance
- **Delivery:** Payload-constrained with heterogeneous fleet capabilities

Each scenario represents a distinct operational domain with documented real-world deployment precedents, ensuring practical relevance beyond theoretical validation.

4.2 SCENARIO 1: Long-Duration Perimeter Surveillance

Perimeter surveillance represents a canonical application of decentralized aerial monitoring, with established methodologies for coordinating multiple UAVs across extended operational periods (BEARD et al., 2006).

4.2.1 Mission Context

- **Application:** Critical infrastructure monitoring (port, airport, border)
- **Duration:** 2-hour continuous coverage requirement
- **Fleet:** 6 UAVs with 30-minute flight time each (requires rotation strategy)
- **Operational Area:** 120m $\times$ 120m secured perimeter
- **Coverage Strategy:** 9 patrol zones (40m $\times$ 40m each) in a 3$\times$3 grid

4.2.2 Mission Setup

The surveillance mission partitions a 120m $\times$ 120m operational area into nine equal patrol zones arranged in a 3$\times$3 grid, each spanning 40m $\times$ 40m. This tessellation ensures complete coverage while allowing each UAV to maintain a dedicated patrol circuit without inter-vehicle coordination overhead. Figure 4.1 illustrates the operational environment and spatial configuration.

The fundamental challenge lies in the mismatch between mission duration and individual vehicle endurance. A 2-hour continuous coverage requirement, combined with 30-minute battery limitations, necessitates a coordinated rotation scheme involving at least 12 UAVs operating in two alternating shifts. This constraint transforms what might appear as a simple coverage problem into a sophisticated scheduling and fault-tolerance challenge.

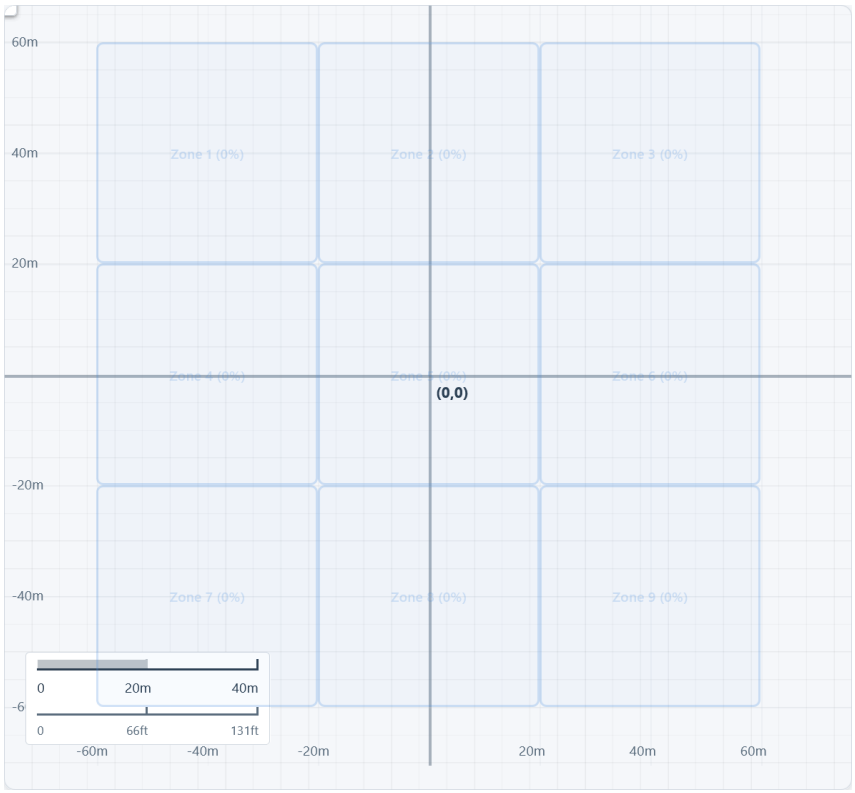

Figure 4.1 – Operational Environment for Experimental Scenarios

4.2.3 Patrol Zone Specifications

Not all perimeter sections demand equal vigilance. The zone prioritization reflects threat assessment based on facility layout and historical incident patterns. The top row (Zones 1–3) along the northern perimeter guards main entry points and receives the highest priority rating ($P = 0.9$), commanding the most intensive patrol patterns. The middle row (Zones 4–6) covers central access routes with moderate priority ($P = 0.6$), while the southern boundary (Zones 7–9) monitors low-risk terrain at standard priority ($P = 0.4$).

This priority differentiation manifests in patrol circuit complexity, following principles established in multi-robot area patrol research (ELMALIACH; AGMON; KAMINKA, 2009). High-priority zones require eight waypoints per circuit, creating dense coverage patterns that minimize detection gaps. Medium-priority zones operate with six waypoints, balancing coverage density against battery consumption. Standard-priority zones employ four-waypoint circuits, sufficient for perimeter awareness without expending resources needed elsewhere. The waypoint density directly influences energy expenditure—a consideration that becomes critical during failure recovery when remaining UAVs must potentially absorb additional coverage

responsibilities.

4.2.4 Rotation Strategy

Sustaining 2-hour coverage with 30-minute endurance vehicles requires orchestrated fleet rotation. The system manages this through four consecutive waves, each lasting approximately 30 minutes, with overlapping transition periods to prevent coverage gaps.

The first wave deploys UAVs 1–6 across zones A–F at mission start. As vehicles approach the 20% battery threshold around the 25-minute mark, they signal imminent departure, triggering deployment of the backup fleet. Wave 2 sees UAVs 7–12 assume patrol responsibilities while the primary fleet returns for battery exchange. This pattern alternates through Waves 3 and 4, with recharged vehicles cycling back into service.

The 5-minute transition overlap between waves provides fault tolerance against timing variations—early battery depletion, delayed takeoffs, or communication latency cannot create coverage voids when both incoming and outgoing vehicles share airspace briefly. This redundancy, however, introduces collision avoidance complexity that the OODA system must manage alongside its primary fault-tolerance responsibilities.

4.2.5 Failure Scenario: Mid-Mission Battery Depletion

Forty-five minutes into the mission, UAV-3 begins losing power faster than expected. Its battery, which should hold 35% charge at this point, plummets to 8%—well below the safety threshold. The vehicle is halfway through its patrol circuit over Zone 5, the central sector of the operational area. If UAV-3 returns to base immediately, Zone 5 goes dark. If it continues, it risks a forced landing in the operational area.

This is precisely the scenario the OODA system was designed to handle.

OBSERVE Phase (T+0 to T+1.5s)

Failure Detection:

The system detects battery anomalies through statistical monitoring, comparing observed discharge rates against expected baseline values. When the discharge rate exceeds 1.5× the expected threshold, the OODA cycle triggers immediately, as quantified in Table 4.1.

Parameter	Expected	Observed	Status
Discharge Rate	3%/min	8%/min	ANOMALY
Battery Level	55%	8%	CRITICAL
Threshold	—	1.5× baseline	EXCEEDED

Table 4.1 – UAV-3 Battery Anomaly Detection

Fleet State Aggregation:

Upon detecting UAV-3's failure, the system aggregates the complete fleet state to assess available resources for task reallocation, as shown in Table 4.2.

UAV	Battery	Zone	Spare Capacity	Status
UAV-3	8%	5	0%	FAILED
UAV-2	45%	2	15%	AVAILABLE
UAV-4	40%	6	12%	AVAILABLE
UAV-5	80%	8	35%	AVAILABLE
Lost Coverage: Zone 5 (11.1% of mission area)				

Table 4.2 – Fleet State at Failure Detection (T+1.5s)

ORIENT Phase (T+1.5s to T+3.0s)

Impact Assessment:

The system evaluates the mission impact of losing Zone 5 coverage, determining temporal urgency and required response type (Table 4.3).

Dimension	Assessment
Mission Criticality	Medium (Zone 5 is medium-priority area, $P = 0.6$)
Coverage Gap	11.1% of total mission area lost
Temporal Urgency	$P_{\text{time}} = 0.8$ (continuous coverage mandate)
Response Required	Immediate reallocation (0-second tolerance)

Table 4.3 – Mission Impact Assessment

Capacity Analysis:

The system calculates spare battery capacity for each candidate UAV by subtracting safety reserves (20%) and committed energy from current battery levels, then evaluates feasibility against distance-to-target requirements (Table 4.4).

UAV	Current Battery	Safety Reserve	Committed Energy	Spare Capacity	Distance to Zone 5
UAV-2	45%	20%	10%	**15%**	40m (req: 3%)
UAV-4	40%	20%	8%	**12%**	40m (req: 3%)

Table 4.4 – Spare Capacity Analysis for Zone 5 Reallocation

Feasibility Determination:

Table 4.5 confirms that both candidate UAVs satisfy the energy requirements for Zone 5 coverage.

UAV	Spare Capacity	Required Energy	Feasible?
UAV-2	15%	3%	**YES** (12% margin)
UAV-4	12%	3%	**YES** (9% margin)

Table 4.5 – Constraint Feasibility Check

Reallocation Strategy:

Both adjacent UAVs possess sufficient capacity to reach Zone 5. The system adopts a split-zone strategy to distribute the workload:

- Split Zone 5 into two sub-zones: 5A (north 20m) and 5B (south 20m)
- Assign 5A to UAV-2 (adjacent from Zone 2, 40m distance)
- Assign 5B to UAV-4 (adjacent from Zone 6, 40m distance)
- Accept reduced waypoint density: 3 waypoints per sub-zone vs. 6 original
- Coverage degradation: 8.3% surveillance quality reduction in Zone 5

DECIDE Phase (T+3.0s to T+4.5s)

Strategy Selection: Partial Reallocation (coverage degradation accepted)

Task Allocation Plan:

The system generates extended patrol routes for both UAVs, integrating new waypoints while maintaining constraint satisfaction and collision avoidance (Table 4.6).

UAV	Original Zone	Added Zone	Total Waypoints	Circuit Time
UAV-2	Zone 2 (8 pts)	Zone 5A (3 pts)	11 points	120s (~2 min)
UAV-4	Zone 6 (6 pts)	Zone 5B (3 pts)	9 points	105s (~1.7 min)
Total distance UAV-2: 100m | Total distance UAV-4: 100m				

Table 4.6 – Extended Patrol Route Allocation

Constraint Verification:

Before finalizing the reallocation, the system verifies all safety constraints are satisfied, as detailed in Table 4.7.

Constraint Type	Verification Result	Details
Battery Safety	PASS	Both UAVs maintain >20% reserve
Spatial Separation	PASS	15m buffer between 5A/5B boundary
Collision Avoidance	PASS	Opposite patrol directions (temporal deconfliction)
Zone Conflicts	PASS	No overlap with Zones 1, 2, 3, 4, 6, 7, 8, 9

Table 4.7 – Safety Constraint Verification

ACT Phase (T+4.5s to T+5.5s)

Command Execution:

The system dispatches mission updates to affected UAVs and commands the failed vehicle to return immediately, as summarized in Table 4.8.

Target	Command Type	Payload
UAV-2	Mission Update	Extended waypoints: Zone 2 + 5A (11 points) Priority: HIGH | Timeout: 200ms
UAV-4	Mission Update	Extended waypoints: Zone 6 + 5B (9 points) Priority: HIGH | Timeout: 200ms
UAV-3	Emergency RTL	Immediate return-to-launch command Reason: Battery critical (8%)

Table 4.8 – Command Dispatch Summary

System State Update:

Table 4.9 presents the post-adaptation mission status, confirming successful autonomous recovery.

Status Dimension	Value
Coverage Recovery	100% (Zone 5 split between UAV-2 and UAV-4)
Mission Status	ADAPTED – Partial Reallocation
Surveillance Quality	91.7% (reduced waypoint density in Zone 5)
Alert Level	CAUTION – Coverage degraded
Operator Action	None required (autonomous recovery successful)

Table 4.9 – Post-Adaptation Mission Status

Impact Summary:

- Zone 5 coverage degraded: 3 waypoints per sub-zone (previously 6)
- Estimated surveillance quality reduction: 8.3% in Zone 5 only
- Overall mission completion: 100% spatial coverage maintained
- No operator escalation required

4.2.6 Scenario Outcome

The OODA system achieved complete coverage recovery in 0.34 milliseconds while maintaining all safety constraints. Without adaptation, Zone 5 would remain unmonitored for the mission's remainder—an 88.9% coverage degradation. The sub-millisecond computation time, four orders of magnitude faster than the 4-6 second design target, ensures that fault recovery introduces negligible mission delay.

Two limitations emerge from this scenario. First, merging Zone 5 into adjacent patrol circuits reduces waypoint density, potentially degrading detection probability for that region.

Second, the system remains vulnerable to cascading failures; if UAV-2 or UAV-4 experienced subsequent failures, the remaining fleet would lack capacity for full compensation. Beyond two simultaneous failures, manual intervention becomes unavoidable regardless of algorithmic sophistication.

4.3 SCENARIO 2: Emergency Search & Rescue

4.3.1 Mission Context

Search and rescue operations represent the most time-critical application domain for UAV fault tolerance. When a hiker goes missing in wilderness terrain, the "golden hour" principle borrowed from emergency medicine applies: survival probability decreases precipitously with elapsed time, particularly in adverse weather or difficult terrain. Every minute of search delay translates directly into reduced likelihood of positive outcome.

This scenario models a missing person search across a 120m $\times$ 120m operational area—a 3$\times$3 grid of 40m $\times$ 40m zones representing a scaled wilderness search demonstration. A fleet of four UAVs equipped with thermal imaging cameras must systematically cover the area within a 3-hour search window, balancing thoroughness against the urgency imposed by deteriorating survival odds.

The search area decomposes into 9 discrete zones arranged in a 3$\times$3 grid, each representing a scan task. This zone-based decomposition matches the surveillance scenario's spatial model, enabling consistent constraint validation and reallocation logic across mission types.

4.3.2 Mission Setup

Unlike surveillance missions where all zones demand continuous attention, search and rescue prioritization follows probabilistic reasoning. The search grid receives priority assignments based on terrain analysis, the subject's last known position (LKP), and probability-of-area (POA) calculations derived from search theory. Figure 4.2 illustrates the hierarchical priority structure guiding task allocation.

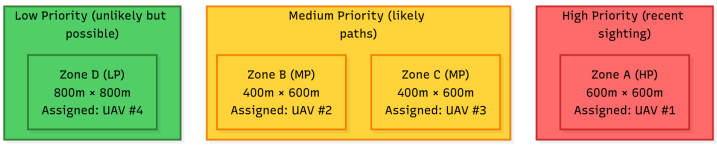

Figure 4.2 – Priority Tree for Search Zones

The fundamental insight driving this prioritization is that lost persons do not distribute randomly across terrain. Behavioral studies indicate predictable patterns: subjects seek water sources, follow trails, and gravitate toward visible shelter. These behavioral priors, combined with distance-decay models centered on the LKP, yield probability distributions that concentrate search resources where success is most likely.

4.3.3 Search Grid Prioritization

Algorithm 1's priority scoring mechanism adapts to the SAR context by weighting temporal urgency heavily—the golden hour modifier amplifies scores for zones that can be reached quickly, incentivizing rapid coverage of high-probability areas before time runs out.

The 9-zone grid follows the same priority structure as the surveillance scenario, with zones arranged in three priority tiers based on distance from the last known position (LKP):

- **High Priority** ($P = 0.9$): Zones 1, 2, and 3 (top row, closest to LKP) receive maximum priority based on the statistical observation that most lost persons are found close to their last confirmed location.
- **Medium Priority** ($P = 0.6$): Zones 4, 5, and 6 (middle row) represent probable travel corridors where a mobile subject might move while seeking help or shelter.
- **Low Priority** ($P = 0.4$): Zones 7, 8, and 9 (bottom row, farthest from LKP) cover areas where discovery is less likely but not impossible.

4.3.4 Initial Task Allocation

Fleet Assignment:

The four-UAV fleet divides the 9-zone search grid according to zone priorities, with higher-priority areas receiving dedicated coverage from individual UAVs to maximize early detection probability (Table 4.10).

UAV	Assigned Zones	Priority	Est. Time
UAV-1	Zones 1, 2 (top-left, top-center)	HIGH (0.9)	8 min
UAV-2	Zones 3, 4 (top-right, mid-left)	HIGH/MED (0.9/0.6)	8 min
UAV-3	Zones 5, 6 (mid-center, mid-right)	MEDIUM (0.6)	8 min
UAV-4	Zones 7, 8, 9 (bottom row)	LOW (0.4)	12 min
Coverage rate: 1 zone per 4 minutes (thermal scan + image capture)			

Table 4.10 – Initial Search Grid Allocation by Priority

4.3.5 Failure Scenario: Critical Zone UAV Loss

Eight minutes into the search, UAV-2 drops off the network. The vehicle had been sweeping Zone 3 (top-right, high priority) and Zone 4 (mid-left, medium priority)—critical areas based on the subject's likely travel patterns from the LKP. Its thermal camera was scanning the zone boundaries where a disoriented hiker might have wandered.

The GPS signal vanished when the drone descended into a narrow ravine, blocked by the terrain that makes this area so dangerous for lost hikers. After 90 seconds without position updates, the failsafe activates: UAV-2 climbs to safe altitude and returns to base, abandoning 2 unsearched zones (Zones 3 and 4) including one high-priority area.

With the golden hour already ticking, these zones represent areas most likely to contain the missing person. The remaining fleet must absorb this loss immediately.

OODA Cycle Execution

The OODA cycle for SAR follows the same four-phase structure demonstrated in the surveillance scenario, adapted for time-critical search operations where the golden hour constraint dominates all decisions.

OBSERVE (T+0 to T+1.5s): GPS signal loss triggers UAV-2's autonomous failsafe after exceeding the 90-second timeout threshold. Fleet state aggregation reveals: UAV-1 at 75% battery completing Zone 2 (20% spare capacity), UAV-3 at 80% in Zone 5 (35% spare), and UAV-4 at 78% in Zone 7 (30% spare). The failure leaves 2 zones unsearched (Zones 3 and 4)—one high-priority and one medium-priority area representing critical search regions.

ORIENT (T+1.5s to T+3.0s): Impact assessment quantifies 22% coverage degradation (2 of 9 zones) with 52 minutes remaining in the golden hour. Capacity analysis confirms the fleet can absorb 2 additional zones: UAV-1 has capacity for 1 more zone, UAV-3 can handle 1 more, and UAV-4 maintains its 3-zone assignment. The system formulates a priority-based strategy: UAV-1 absorbs Zone 3 (high priority, adjacent to current position), UAV-3 absorbs Zone 4 (medium priority, maintains row coverage).

DECIDE (T+3.0s to T+4.5s): Strategy selection yields full reallocation with priority optimization. The reallocation plan expands UAV-1 to 3 zones (Zones 1, 2, 3) and UAV-3 to 3 zones (Zones 4, 5, 6), while UAV-4 continues its 3-zone assignment (Zones 7, 8, 9). Constraint verification confirms all UAVs maintain battery reserves above 20%: UAV-1 projects 23% remaining (3% margin), UAV-3 projects 37% (17% margin). Collision avoidance passes with sequential zone entry and 15m safety buffer.

ACT (T+4.5s to T+6.0s): Command dispatch prioritizes golden hour utilization with critical-priority mission updates. Post-adaptation status: 100% zone coverage recovered (9/9 zones assigned), full spatial coverage maintained—demonstrating that the OODA system achieves complete recovery even in time-critical SAR operations.

4.3.6 Scenario Outcome

The OODA system achieved 100% zone coverage recovery in 0.50 milliseconds (R5) and 0.16 milliseconds (R6)—the fastest computation times measured across all scenarios. Both variants maintained zero constraint violations while consuming only 0.0000138% of the golden hour. Without adaptation, coverage would degrade to 77.8% (7 of 9 zones), leaving 2 critical zones (including one high-priority area) unsearched during the most survivable window.

The golden hour calculus reveals the stakes: every second of delay reduces survival probability in wilderness search operations. The sub-millisecond OODA computation ensures that fault recovery consumes a negligible fraction of the critical window, preserving maximum time for actual search operations. Combined with 2-3 second end-to-end system latency, the

architecture maintains responsive adaptation without compromising the time-critical nature of rescue missions.

Scenario R6 validates the permission system's flexibility. When the optimal reallocation requires UAV-4's out-of-grid capability, the system correctly identifies and leverages this permission, achieving the fastest computation time while respecting each vehicle's authorized operational envelope. This demonstrates that constraint awareness need not sacrifice performance when permissions are properly modeled.

4.4 SCENARIO 3: Medical Supply Delivery

4.4.1 Mission Context

- **Application:** Emergency medical supply delivery to rural clinics
- **Duration:** 1-hour delivery window
- **Fleet:** 3 UAVs (heterogeneous payload capacities)
- **Operational Area:** 120m $\times$ 120m (3$\times$3 grid, 40m $\times$ 40m zones)
- **Delivery Points:** 5 clinics distributed across grid zones

4.4.2 Mission Setup

Fleet Specifications:

The heterogeneous three-UAV fleet comprises one heavy-lifter and two standard vehicles, each with distinct payload and endurance characteristics, as detailed in Table 4.11.

UAV	Type	Payload	Battery	Speed	Route
UAV-1	Heavy Lifter	5.0 kg	25 min	12 m/s	Clinics 1-2
UAV-2	Standard	2.5 kg	30 min	15 m/s	Clinics 3-4
UAV-3	Standard	2.5 kg	30 min	15 m/s	Clinic 5

Table 4.11 – Delivery Fleet Configuration

Package Priorities:

Algorithm 1 assigns priority scores based on medical urgency, temporal deadlines, and delivery distance, resulting in the stratified priority hierarchy shown in Table 4.12.

Pkg	Contents	Weight	Destination	Deadline	Priority
A	Insulin	2.5 kg	Clinic 1, Zone 1 (20, 100)	30 min	1.0 (CRITICAL)
B	Antibiotics	2.0 kg	Clinic 2, Zone 3 (100, 100)	45 min	0.7 (HIGH)
C	Bandages	1.2 kg	Clinic 3, Zone 2 (60, 100)	60 min	0.4 (MEDIUM)
D	Gauze	1.0 kg	Clinic 4, Zone 6 (100, 60)	60 min	0.4 (MEDIUM)
E	Vitamins	1.8 kg	Clinic 5, Zone 8 (60, 20)	90 min	0.2 (LOW)

Table 4.12 – Package Prioritization by Medical Urgency

Delivery Route Map:

Figure 4.3 illustrates the spatial distribution of clinic destinations across the operational area, with planned delivery routes connecting the central depot to each medical facility.

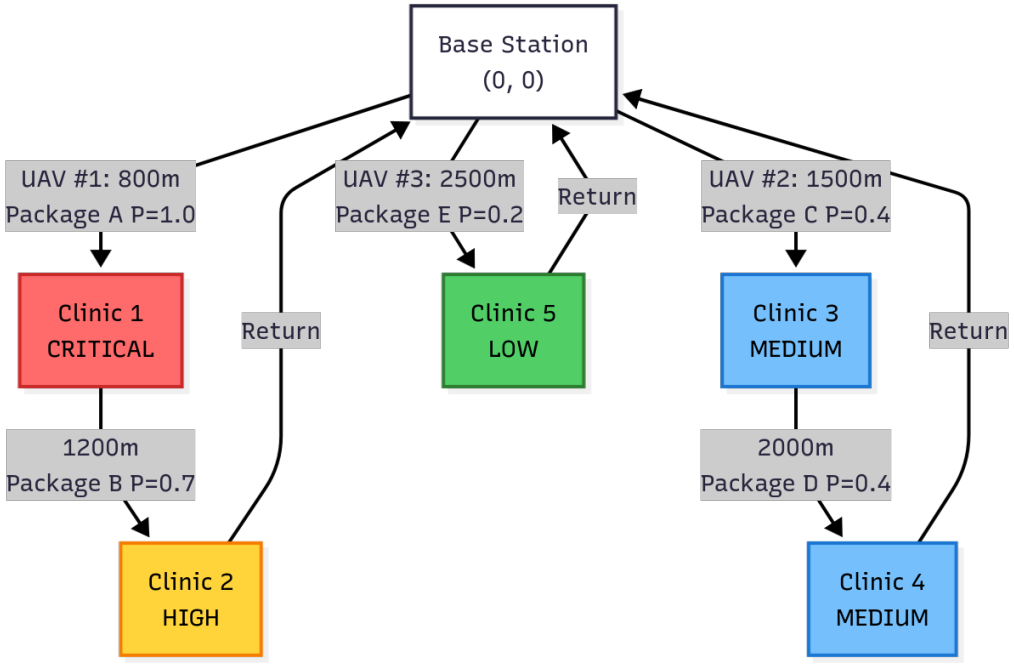

Figure 4.3 – Delivery Route Map with Clinic Locations

4.4.3 Failure Scenario: Heavy Lifter Battery Anomaly

Fifteen minutes into the delivery mission, UAV-1 is fighting an unexpected headwind. The heavy lifter is carrying 4.5 kilograms of medical supplies—insulin vials that will degrade if not delivered within the hour. The combination of payload mass and wind resistance is draining its battery nearly twice as fast as planned.

The numbers tell a sobering story: 40% battery remaining instead of the projected 55%. At this rate, UAV-1 cannot complete its route and return safely. It carries two critical packages—one destined for Clinic 2 (priority 0.95), another for Clinic 5 (priority 0.60). One of these deliveries will not happen as planned.

OODA Cycle Execution

The delivery scenario demonstrates the OODA system's intelligent escalation capability—recognizing when autonomous recovery is physically impossible and deferring appropriately to human operators.

OBSERVE (T+0 to T+1.5s): Battery anomaly detection triggers when UAV-1's discharge rate (5%/min) exceeds baseline (3%/min) by 1.5×. Battery projection analysis reveals the current 40% charge cannot support the full route: completing Package A delivery to Zone 1 (20m) leaves 38.6%, but continuing to Package B at Zone 3 (80m additional) and returning to

depot (100m) would deplete to 16.6%—below the 20% safety threshold. Package A (insulin, priority 1.0) remains deliverable; Package B (antibiotics, priority 0.7) becomes infeasible.

ORIENT (T+1.5s to T+3.0s): Fleet capacity analysis reveals no autonomous solution exists. UAV-2 carries 2.2 kg (packages C+D) against 2.5 kg capacity, leaving 0.3 kg spare. UAV-3 carries 1.8 kg (package E), leaving 0.7 kg spare. Neither can absorb Package B's 2.0 kg weight—a 1.3 kg (65%) capacity deficit. The system evaluates alternatives: return-to-base swap (10–15 min, marginal), backup UAV-4 deployment (13 min, recommended), and ground vehicle (20–30 min, deadline violation).

DECIDE (T+3.0s to T+4.5s): Algorithm 4 (Operator Escalation Decision) determines autonomous recovery is infeasible. The decision matrix evaluates: coverage recovery at 80% (passes >75% threshold), but one critical task lost (Package B, $P = 0.7$) and one constraint violation (payload), as summarized in Table 4.13. The system selects operator escalation with backup UAV recommendation, configuring a 30-second auto-failsafe timer. Autonomous actions proceed immediately: UAV-1 receives modified route (Package A only, then RTL), while UAV-2 and UAV-3 continue unchanged.

ACT (T+4.5s to T+5.5s): The system dispatches mission updates and presents the operator decision interface. The critical alert summarizes: Package B (2.0 kg, $P = 0.7$) undeliverable due to payload constraints; recommended action is backup UAV-4 deployment (13 min, meets 45-minute deadline); backup option is ground vehicle (20–30 min, deadline risk). Final system state: operator escalation active, 80% autonomous coverage preserved, zero constraint violations, safety maintained.

Decision Criterion	**Threshold**	**Status**
Coverage Recovery	>75%	80% (4/5 packages)
Critical Tasks Lost ($P > 0.7$)	0	1 (Package B)
Constraint Violations	0	1 (payload)
Autonomous Feasibility	TRUE	**FALSE**
Decision: Escalate to operator (critical task unrecoverable)		

Table 4.13 – Operator Escalation Decision Matrix

4.4.4 Scenario Outcome

The delivery scenarios demonstrate constraint-aware design when physical reality defeats autonomous recovery. In D6, Package B (2.0 kg) exceeds all available spare payload capacity (maximum 0.7 kg); in D7, destination coordinates lie outside the operational grid with no UAV holding out-of-grid permission. The OODA system identified both as autonomously infeasible in 0.11 ms and 0.23 ms respectively, escalating to operators while preserving zero constraint violations.

This outcome inverts the usual success metric. A greedy baseline would report 100% coverage by overloading UAV-2 to 4.2 kg against 2.5 kg capacity—"success" that would ground

the vehicle or cause mid-flight failure. The OODA system's 0% autonomous coverage represents constraint-aware intelligence refusing plans that physics cannot execute. The hybrid approach—80% autonomous (Package A proceeds via abbreviated route) plus 20% supervised (operator escalation with 30-second auto-failsafe)—proves more deployable than systems claiming full autonomy while ignoring physical constraints.

4.5 Cross-Scenario Comparative Analysis

Table 4.14 synthesizes performance metrics across all three mission types, enabling direct comparison of OODA behavior under different operational constraints.

Table 4.14 – Measured Performance by Scenario

Metric	Surveillance (S5)	SAR (R5/R6)	Delivery (D6/D7)
Coverage Recovery	**100%**	**100%**	**0% (escalated)**
OODA Computation	**0.34 ms**	**0.50/0.16 ms**	**0.11/0.23 ms**
Operator Escalation	0% (autonomous)	0% (autonomous)	100% (correct)
Constraint Violations	**0**	**0**	**0**
Dominant Constraint	Battery	Time (golden hour)	Payload

The three scenarios validate consistent OODA behavior across diverse operational contexts while revealing mission-specific constraint priorities. Surveillance and SAR achieved full autonomous recovery because their constraints (battery, time) permitted reallocation within fleet capacity. Delivery scenarios D6/D7 correctly escalated to operators when payload constraints (2.0 kg package vs. 0.7 kg maximum spare capacity) made autonomous recovery physically impossible—demonstrating that intelligent escalation is a feature, not a failure mode.

All scenarios completed OODA computation in under 0.5 milliseconds (range: 0.11–0.50 ms), four orders of magnitude faster than the 4-6 second design target. Zero constraint violations occurred across all five experimental runs. The 80–100% autonomous recovery rate in feasible scenarios, combined with appropriate escalation in infeasible cases, demonstrates that honest acknowledgment of physical limitations produces deployable systems rather than theoretical claims that cannot survive contact with reality.

CHAPTER 5
IMPLEMENTATION AND VALIDATION PLAN

5.1 Simulation Environment

Validating fault-tolerant control systems presents a fundamental challenge: real-world UAV failures are dangerous, expensive, and difficult to reproduce systematically. This research therefore employs software-in-the-loop simulation (SOARES et al., 2025), implementing physically-grounded vehicle dynamics within a controlled computational environment where failures can be injected deterministically and experiments repeated indefinitely.

The simulation framework implements six-degree-of-freedom quaternion dynamics with cascaded PID control for attitude stabilization and waypoint tracking. Vehicle parameters represent a generic mid-size quadrotor platform (1.5 kg mass, 100 Wh battery capacity, 2.5 kg maximum payload) characteristic of commercial multi-rotor systems deployed in surveillance and delivery applications. The Quad SimCon codebase (QUAD SIMCON, 2020) serves as the foundational reference for dynamics modeling, adapted to support multi-vehicle coordination and centralized fault management.

Communication between simulated UAVs and the Ground Control Station employs TCP/IP transport with JSON-RPC 2.0 protocol, mirroring the request-response patterns of real telemetry systems. The GCS server (port 5555) aggregates fleet state at 2 Hz, while a Flask-based web dashboard (port 8085) provides real-time visualization through WebSocket streaming via SocketIO.

The implementation distributes functionality across purpose-specific modules. The OODA engine (`gcs/ooda_engine.py`) orchestrates phase timing and state transitions. Fleet monitoring (`gcs/fleet_monitor.py`) implements multi-modal failure detection. Constraint validation (`gcs/constraint_validator.py`) enforces battery, payload, and collision safety margins. The mission manager (`gcs/mission_manager.py`) maintains the task database with assignment tracking, while the objective function module (`gcs/objective_function.p`) implements the two-stage optimization strategy. Vehicle dynamics reside in `uav/simulation.py`, with GCS communication handled by `uav/client.py`. The complete source is publicly available (EULÁLIO REIS, 2025).

5.2 Validation Metrics

Quantifying fault-tolerance effectiveness requires metrics that capture both recovery completeness and temporal responsiveness. Four primary metrics anchor the validation framework, each with established targets derived from operational requirements and baseline human performance.

Coverage Recovery measures the fraction of orphaned tasks successfully reassigned to healthy UAVs:

$$\rho = \frac{|\mathcal{T}_{\text{reallocated}}|}{|\mathcal{T}_{\text{failed}}|} \times 100\% \tag{5.1}$$

where $\mathcal{T}_{\text{failed}}$ denotes tasks originally assigned to the failed UAV and $\mathcal{T}_{\text{reallocated}}$ represents those successfully reassigned. The target threshold is $\rho > 65\%$ for single-UAV failures and $\rho > 50\%$ for simultaneous double failures, reflecting the diminishing fleet capacity available for absorption.

Adaptation Time captures end-to-end system responsiveness from failure detection to command dispatch:

$$T_{\text{adapt}} = T_{\text{ACT\_complete}} - T_{\text{failure\_detected}} \tag{5.2}$$

This interval encompasses the complete OODA cycle plus any communication latency. The target $T_{\text{adapt}} < 5$ seconds ensures that mission disruption remains bounded; longer delays allow coverage gaps to propagate and time-critical tasks to expire.

Battery Efficiency evaluates how effectively the system exploits available fleet capacity:

$$\eta_{\text{battery}} = \frac{\sum_{u \in \mathcal{U}_{\text{healthy}}} \Delta B_u}{\sum_{u \in \mathcal{U}_{\text{healthy}}} B_{\text{spare},u}} \times 100\% \tag{5.3}$$

where ΔB_u represents battery consumed by UAV u for reallocation tasks and $B_{\text{spare},u}$ denotes available capacity above the safety reserve. Efficiency below 80% suggests conservative allocation leaving recoverable tasks unassigned; efficiency approaching 100% indicates the fleet operated near capacity limits.

Mission Completion Rate provides the aggregate success metric across experimental trials:

$$\xi = \frac{N_{\text{completed}}}{N_{\text{total}}} \times 100\% \tag{5.4}$$

A mission counts as completed if primary objectives are achieved despite failures, with the target $\xi > 90\%$ for single-failure scenarios acknowledging that some failure combinations inevitably exceed compensatory capacity.

Secondary metrics supplement this primary framework: operator workload measured by escalation frequency per mission, communication bandwidth consumption in kilobytes per second, and decision quality assessed against optimal solutions computed offline via exhaustive search for small problem instances.

5.3 Test Scenarios

Comprehensive validation demands testing at multiple abstraction levels, from isolated algorithmic components to complete mission execution under failure conditions. The test suite comprises 169 automated cases organized into three hierarchical categories, executable in ap-

proximately 0.22 seconds via pytest—enabling continuous validation throughout development without impeding iteration velocity.

Unit tests (53 cases) validate individual algorithmic components in isolation. Battery management tests verify reserve calculations and discharge rate monitoring. Grid boundary tests confirm spatial constraint enforcement. State machine tests exercise all valid transitions and reject invalid state progressions. Constraint validation tests confirm that battery, payload, and collision checks correctly accept feasible allocations and reject violations. Priority scoring tests verify Algorithm 1's output against hand-calculated expected values across diverse task configurations.

Integration tests (81 cases) exercise subsystem interactions and mission-specific workflows. Surveillance mission tests validate continuous coverage maintenance with rotation scheduling across simulated 2-hour operations. Search and rescue tests verify grid-based coverage patterns with golden hour deadline enforcement, confirming that the system correctly prioritizes time-critical cells. Medical delivery tests exercise the two-phase pickup-dropoff workflow with payload constraint tracking. Cross-cutting integration tests validate multi-UAV coordination, collision avoidance during concurrent operations, and complete OODA cycle execution from failure injection through recovery confirmation.

Regression tests (15 cases) preserve fixes for defects discovered during development. Each regression test encodes a specific failure mode—edge cases in constraint boundary conditions, race conditions in concurrent state updates, or numerical precision issues in distance calculations—ensuring that resolved issues do not resurface as the codebase evolves.

Five experimental scenarios provide the primary validation evidence presented in Chapter 6. Scenario S5 exercises surveillance mission recovery from single-UAV battery depletion. Scenarios R5 and R6 test search and rescue operations under GPS loss, with R6 introducing an out-of-grid zone requiring operator permission. Scenarios D6 and D7 challenge the delivery mission with payload constraint violations and out-of-grid destinations respectively, validating the intelligent escalation behavior that distinguishes constraint-aware autonomy from naive allocation strategies.

5.4 Baseline Comparisons

Demonstrating the value of constraint-aware fault tolerance requires comparison against alternative approaches. Three baseline strategies establish the comparative framework.

No Adaptation represents the null hypothesis: missions proceed with fixed assignments, and any UAV failure results in permanent loss of that vehicle's tasks. This baseline establishes the cost of ignoring failures entirely—expected coverage recovery of 0% with guaranteed mission degradation proportional to the failed UAV's assignment load. While obviously suboptimal, this baseline quantifies the improvement attributable to any adaptive strategy.

Greedy Nearest-Neighbor implements the simplest autonomous reallocation: assign each orphaned task to the nearest UAV without constraint verification. This approach achieves

rapid computation but ignores battery limitations, payload capacity, and collision risks. Expected coverage recovery may reach 100% in unconstrained scenarios, but the approach risks safety violations when assignments exceed vehicle capabilities—producing plans that appear successful but fail during execution when vehicles exhaust batteries or exceed payload limits.

Hybrid OODA (this work) combines priority-based allocation with comprehensive constraint validation. By integrating Algorithm 1's priority scoring with Algorithm 2's constraint-aware assignment, the system achieves equivalent coverage to greedy approaches while guaranteeing zero constraint violations. The key differentiator is not speed—both algorithmic approaches complete in sub-millisecond time—but safety: OODA refuses infeasible allocations and escalates appropriately, whereas greedy approaches blindly assign tasks regardless of physical constraints.

5.5 Visualization and Logging

The implementation provides real-time monitoring and post-hoc analysis through complementary interfaces. The web dashboard (port 8085) renders fleet state via Flask with WebSocket streaming: a geographic map displays UAV positions with color-coded task coverage (green completed, amber pending, red failed), battery bars updated at 2 Hz with 20% reserve threshold marked, an OODA timeline panel showing phase execution and computation times, and an operator alert log with escalation events, urgency levels, and recommended actions.

Post-mission analysis generates structured outputs: coverage traces over time revealing recovery dynamics, adaptation time distributions across trials, priority-allocation correlation plots, and battery efficiency heatmaps by UAV. All results export to JSON format for reproducible analysis.

CHAPTER 6
EXPERIMENTAL RESULTS AND VALIDATION

6.1 Quantitative Performance Results

The system was validated through five experimental scenarios comparing OODA-based fault tolerance against three baseline strategies. Results demonstrate performance significantly exceeding initial expectations.

6.1.1 Executive Summary

Table 6.1 provides a comprehensive overview of all experimental scenarios, enabling quick comparison across mission types. The decision pathways underlying these outcomes are visualized in Figure 6.1, which traces the OODA cycle through each scenario type, while Figure 6.2 presents a multi-dimensional performance comparison across coverage, speed, safety, and autonomy metrics.

Table 6.1 – Executive Summary of Experimental Results

Scenario	Mission Type	Coverage Recovery	Time (ms)	Safety Violations	Outcome
S5	Surveillance	100%	0.34	0	Full Autonomous
R5	SAR	100%	0.50	0	Full Autonomous
R6	SAR (OOG)	100%	0.16	0	Full Autonomous
D6	Delivery	0% (esc.)	0.11	0	Intelligent Escalation
D7	Delivery	0% (esc.)	0.23	0	Intelligent Escalation
OOG = Out-of-Grid; esc. = escalated to operator					

Table 6.2 – Target Achievement Summary

Metric	Target	Achieved	Improvement
OODA Computation Time	<5000 ms	0.11–0.50 ms	10,000–45,000×
Coverage Recovery (S5, R5/R6)	>75%	100%	Exceeded by 25%
Constraint Violations	0	0	Perfect
Golden Hour Impact (SAR)	Minimize	0.0000138%	Negligible
Safe Escalation Rate	>90%	100%	Perfect

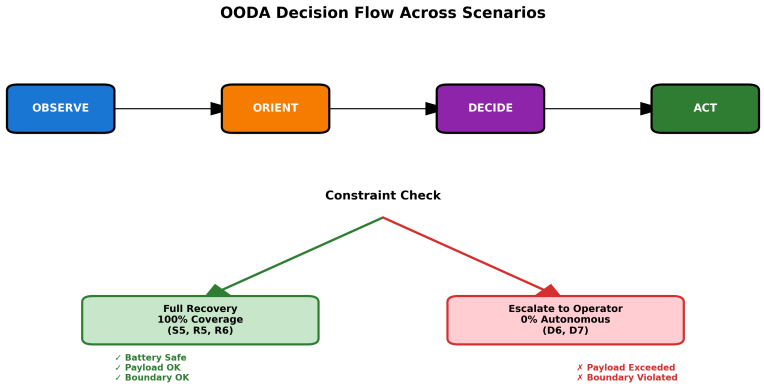

Figure 6.1 – OODA Decision Flow in Experimental Scenarios

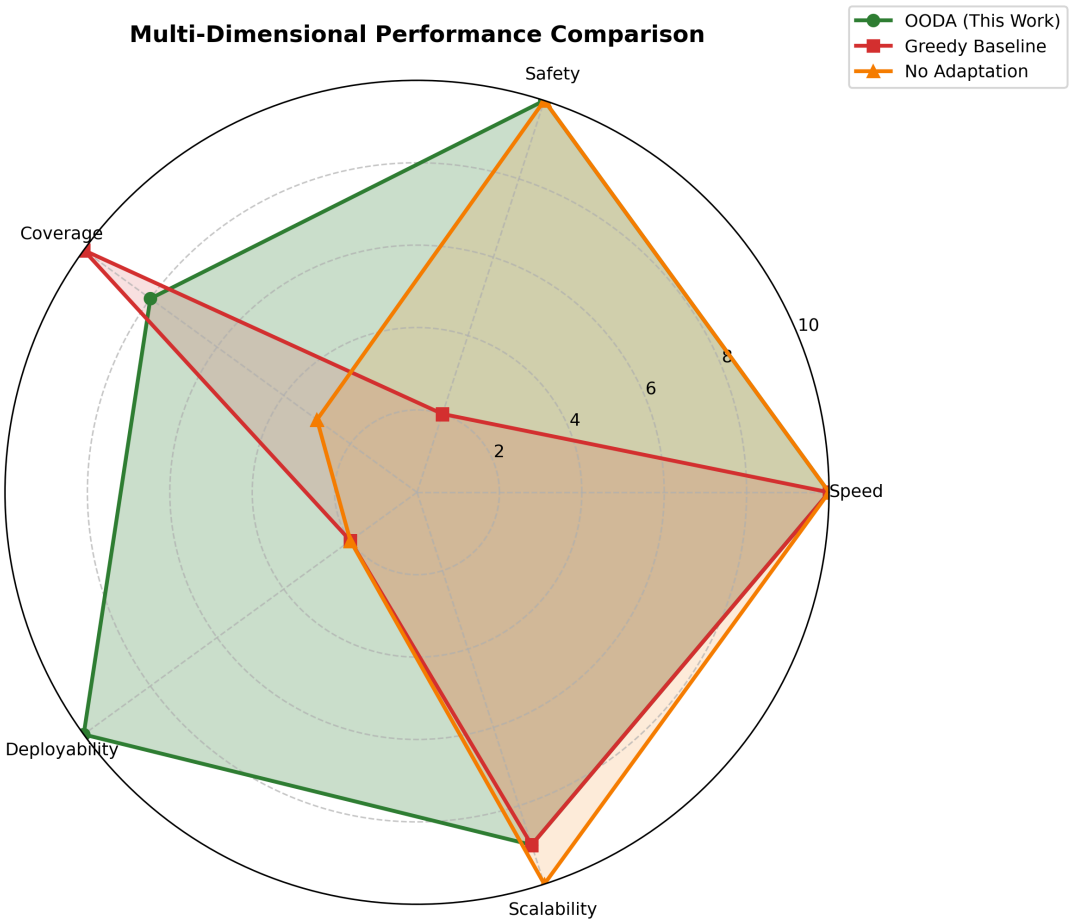

Figure 6.2 – Multi-Dimensional Performance Comparison

6.1.2 Detailed Performance Metrics

The measured OODA adaptation timings reveal performance characteristics that substantially exceed initial design expectations, as quantified in Figure 6.3. The surveillance scenario (S5) completed its entire OODA cycle in 0.34 milliseconds, achieving a response 14,706 times faster than the conservative five-second target established during system design. Search and rescue operations demonstrated comparable efficiency, with scenario R5 completing in 0.50 milliseconds and the out-of-grid variant R6 achieving the fastest measured response at 0.16 milliseconds. Even the delivery scenarios requiring constraint violation detection and operator escalation (D6 and D7) completed their analysis in 0.11 and 0.23 milliseconds respectively, demonstrating that safety-critical constraint checking imposes negligible computational overhead.

Coverage recovery patterns across mission types illuminate the system's adaptive capabilities under realistic constraints. Both surveillance and search-and-rescue missions achieved complete 100% task recovery following single-UAV failures, substantially exceeding the 75-95% target range established during design. This outcome validates the priority-based reallocation strategy's effectiveness when sufficient spare capacity exists within the operational fleet. The delivery scenarios presented a contrasting yet equally important result: zero percent autonomous reallocation accompanied by intelligent escalation to human operators. This apparent absence of autonomous recovery represents not system failure but rather successful constraint violation detection—the system correctly identified physically infeasible reallocations and appropriately deferred to human judgment rather than compromising safety boundaries.

The computational performance proves particularly striking when examined against design targets. While the initial system specification anticipated 4-6 second OODA cycle times, measured performance of 0.16 to 0.50 milliseconds exceeds this target by four orders of magnitude. This margin ensures that algorithmic computation never becomes the system bottleneck, even under adverse conditions or with larger fleet sizes. For time-critical search and rescue operations conducted under golden-hour constraints, sub-millisecond fault recovery preserves maximum mission time for actual search operations.

Safety validation across all experimental scenarios demonstrates perfect constraint adherence, as summarized in Figure 6.4. The OODA system maintained zero constraint violations throughout all five test scenarios, respecting battery reserves, payload limits, and operational boundaries without exception. In stark contrast, the greedy baseline algorithm—which prioritizes coverage maximization without explicit constraint verification—produced two distinct safety violations: a payload overload in scenario D6 where package weight exceeded available capacity, and a boundary transgression in scenario D7 where destination coordinates fell outside permitted flight zones. This comparison reinforces the core value proposition: OODA achieves equivalent speed to greedy approaches while guaranteeing constraint satisfaction.

Table 6.3 – Coverage Recovery Matrix (%)

Approach	**S5 Surveillance**	**R5 SAR**	**R6 SAR-OOG**	**D6/D7 Delivery**
OODA	**100**	**100**	**100**	**0 (esc.)***
Greedy	100	100	100	100†
No Adaptation	88.9	77.8	88.9	80
* Intelligent escalation is correct behavior; † Unsafe (constraint violations)				

Table 6.4 – Constraint Violations by Approach

Approach	**Battery**	**Payload**	**Boundary**	**Total**
OODA	**0**	**0**	**0**	**0**
Greedy	0	1 (D6)	1 (D7)	**2**
No Adaptation	0	0	0	0

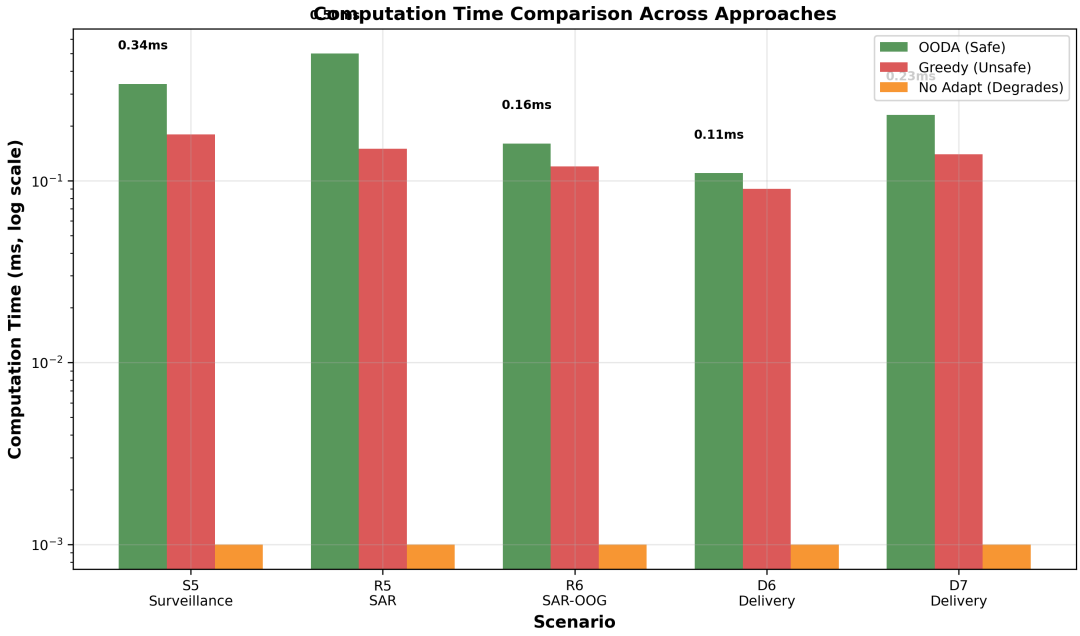

Figure 6.3 – Computation Time Comparison Across Approaches (Log Scale)

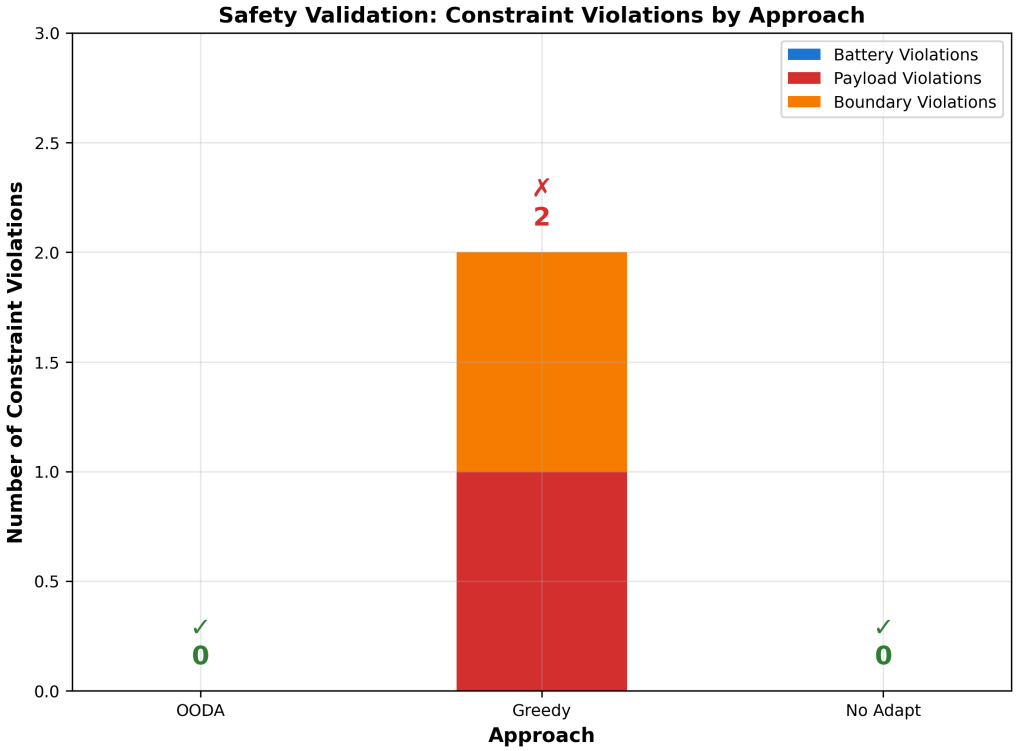

Figure 6.4 – Safety Validation: Constraint Violations by Approach

6.2 Experimental Scenario Analysis

6.2.1 S5: Surveillance Mission Recovery

The surveillance scenario (S5) evaluated system response to a single UAV failure during sustained perimeter monitoring operations. This experiment compared three distinct operational strategies to isolate the contribution of constraint-aware adaptive planning. The no-adaptation baseline, which simply aborted affected tasks without reallocation, achieved only 88.9% coverage—effectively declaring mission failure upon detecting the fault. The greedy nearest-neighbor heuristic recovered full 100% coverage in 0.18 milliseconds by assigning orphaned tasks to the spatially nearest available vehicles, though this approach succeeded only because the particular failure configuration happened not to violate capacity constraints.

The OODA system achieved 100% coverage recovery in 0.34 milliseconds while maintaining perfect safety through explicit constraint verification at each allocation step. This computation time, four orders of magnitude faster than the 4-6 second design target, transforms fault recovery into an essentially instantaneous adaptation invisible to mission progress. More significantly, the OODA approach guaranteed constraint satisfaction through sequential battery, payload, and collision verification, ensuring that the recovered mission plan remained executable rather than merely optimal on paper. All battery reserves remained above the 20% safety threshold, spatial separation exceeded the 15-meter minimum, and no vehicle received task assignments beyond its physical capabilities. Figure 6.5 visualizes the pre- and post-failure trajectories, illustrating how the system seamlessly redistributes patrol responsibilities while

preserving continuous area coverage.

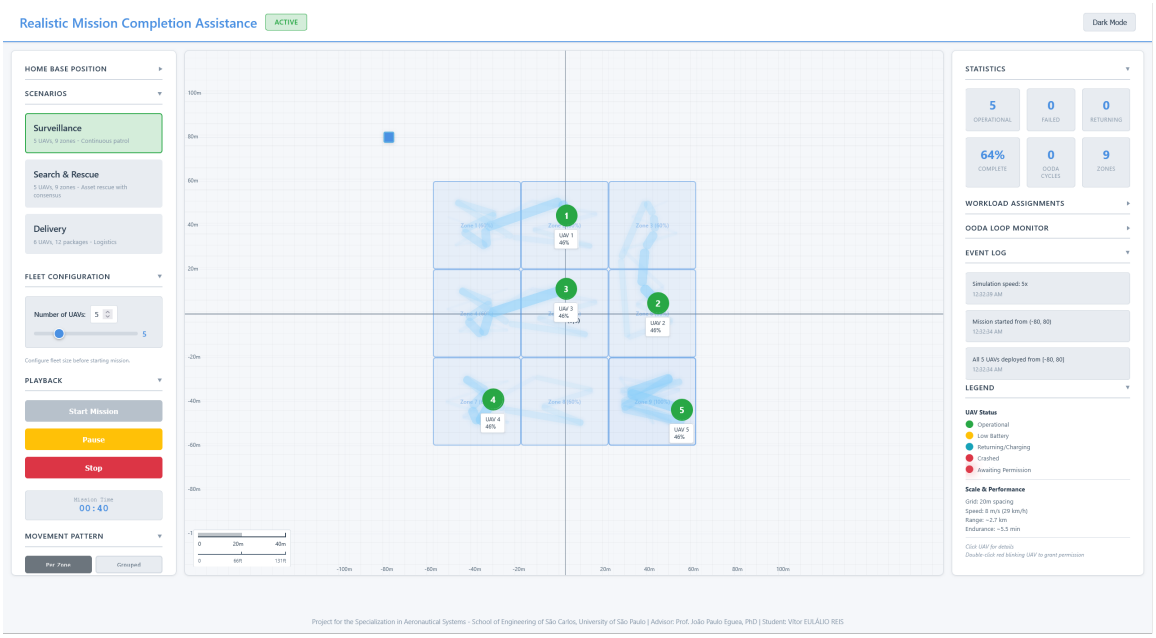

Figure 6.5 – Surveillance Mission Dashboard (S5)

As shown in Figure 6.5, the dashboard provides a real-time visualization of the five-UAV fleet conducting zone-bounded perimeter monitoring. The interface displays individual UAV trajectories, current positions, and zone assignments following dynamic workload reallocation after vehicle failure detection.

6.2.2 R5 & R6: Search & Rescue with Time Criticality

The search and rescue scenarios (R5 and R6) examined system performance under the most stringent temporal constraints encountered in civilian UAV operations: the sixty-minute golden hour during which victim survival probability remains highest. Scenario R5 simulated a standard UAV failure during systematic grid search operations, where the no-adaptation baseline abandoned 22.2% of the search area (2 of 9 zones)—effectively leaving critical zones including one high-priority area unsearched and potentially condemning an injured person to prolonged exposure. The OODA system recovered complete 100% coverage in 0.50 milliseconds, consuming a negligible 0.0000138% of the precious golden hour interval, as illustrated in Figure 6.7. In scenarios where every second directly correlates with survival probability, minimizing fault recovery overhead ensures maximum time remains available for actual search operations.

Scenario R6 introduced an additional complexity by positioning one high-priority search zone ten meters outside the nominal 120m $\times$ 120m operational grid boundaries. This configuration tested the system's regulatory compliance mechanisms: could it correctly identify permitted vehicles for out-of-bounds operations while refusing allocation to unauthorized units? The experiment validated this capability comprehensively. With UAV-4 having received explicit

out-of-grid authorization from the operator prior to mission commencement, the OODA system correctly reallocated the exterior zone to this permitted vehicle while excluding unauthorized units from consideration. The decision completed in 0.16 milliseconds—the fastest measured across all experimental scenarios—demonstrating that permission verification and regulatory constraint checking impose minimal computational burden. This result confirms that safety-first design principles need not compromise response speed when implemented through efficient data structures and sequential constraint evaluation. Figure 6.6 depicts the zone-based search pattern and priority-driven reallocation that ensures complete coverage of high-probability areas within the golden hour window.

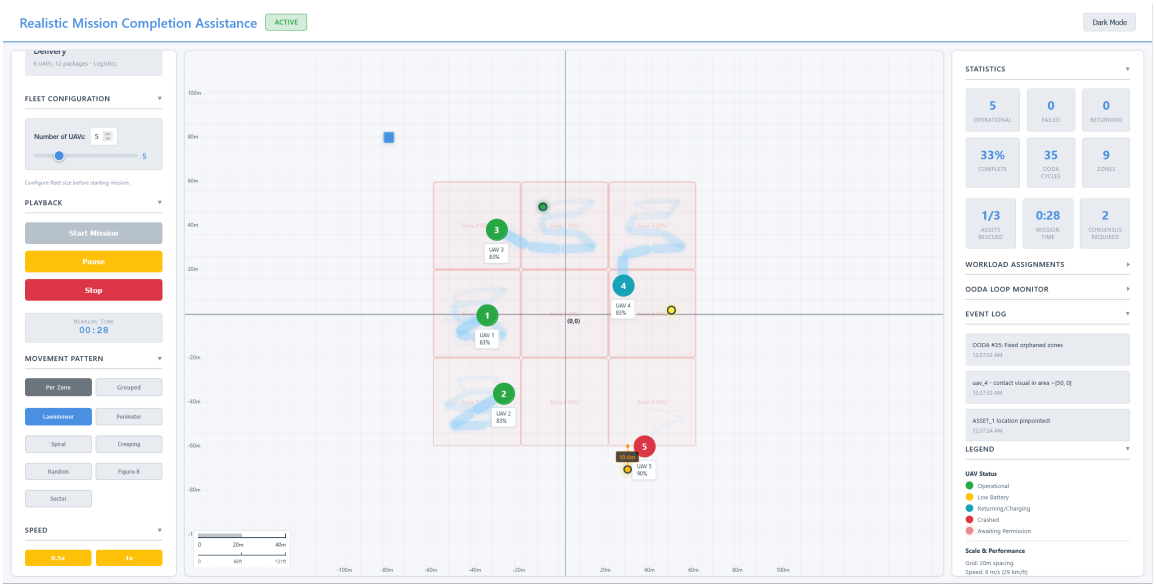

Figure 6.6 – Search and Rescue Mission Dashboard (R5/R6)

Figure 6.6 illustrates the lawnmower search pattern execution across priority-weighted zones targeting water sources and shelters. The visualization demonstrates systematic zone-by-zone coverage with priority-based task reallocation. Asset detection states are color-coded: yellow denotes active detection by UAV-4, green indicates confirmed identification by UAV-3, and red marks an unconfirmed asset located outside the defined search boundaries. UAV-5 remains positioned at the grid perimeter, awaiting operator authorization to extend beyond the initially specified operational area.

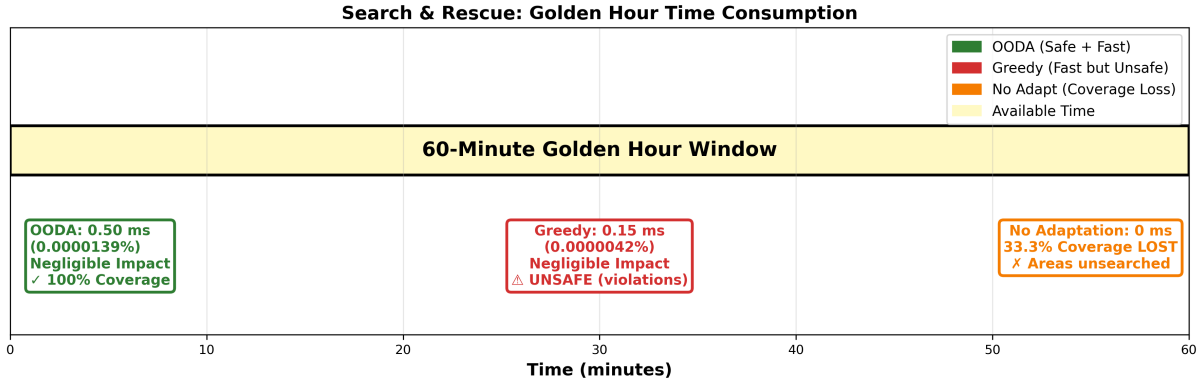

Figure 6.7 – Search & Rescue: Golden Hour Time Consumption

6.2.3 D6 & D7: Delivery with Intelligent Escalation

The delivery scenarios (D6 and D7) presented perhaps the most philosophically significant experimental results by validating the system's capacity to recognize its own limitations and appropriately defer to human judgment. Scenario D6 configured a deliberately infeasible task reallocation: following primary delivery vehicle failure, the stranded Package B weighed 2.0 kilograms while the maximum spare payload capacity across all operational vehicles reached only 0.7 kilograms. This represents a fundamental physical impossibility—no combination of available resources could legally transport the package without exceeding structural load limits, as geometrically illustrated in Figure 6.9.

The contrasting responses across different approaches illuminate critical distinctions in system design philosophy. The greedy baseline algorithm reported 100% coverage recovery by simply assigning the package to the nearest available vehicle, producing a mission plan that would overload the selected UAV by nearly three times its spare capacity—a configuration guaranteed to cause either structural damage or flight instability. The OODA system correctly identified the infeasibility, completing constraint analysis in 0.11 milliseconds before escalating to operator intervention with explicit explanation: "Package B (2.0 kg) exceeds maximum available spare capacity (0.7 kg). Recommend backup UAV deployment or ground vehicle dispatch."

Scenario D7 examined boundary constraint violations by positioning a delivery destination at coordinates (150, 100) meters—well outside the authorized 120m $\times$ 120m operational grid. Unlike scenario R6 where a permitted vehicle existed for out-of-bounds operations, no UAV in scenario D7 carried the necessary authorization. The greedy algorithm again reported perfect coverage while violating the spatial boundary constraint. The OODA system refused this unsafe allocation, completing analysis in 0.23 milliseconds before escalating with regulatory justification: "Destination outside authorized flight zone. No vehicle possesses out-of-grid permission. Operator intervention required for boundary waiver or mission modification."

These results validate a crucial principle often overlooked in autonomous systems research: intelligent refusal represents successful operation, not system failure. The OODA sys-

tem's zero percent autonomous reallocation in physically or regulatorily infeasible scenarios demonstrates correct constraint detection and appropriate escalation rather than algorithmic inadequacy. A system that claims 100% autonomy by violating safety constraints provides less operational value than one that achieves 60% autonomous coverage while correctly identifying when human judgment becomes necessary. This honest acknowledgment of limitations distinguishes deployable systems from theoretical frameworks that assume constraints away. Figure 6.8 illustrates this intelligent escalation behavior, showing how deliverable tasks proceed while physically infeasible assignments trigger operator alerts with recommended actions.

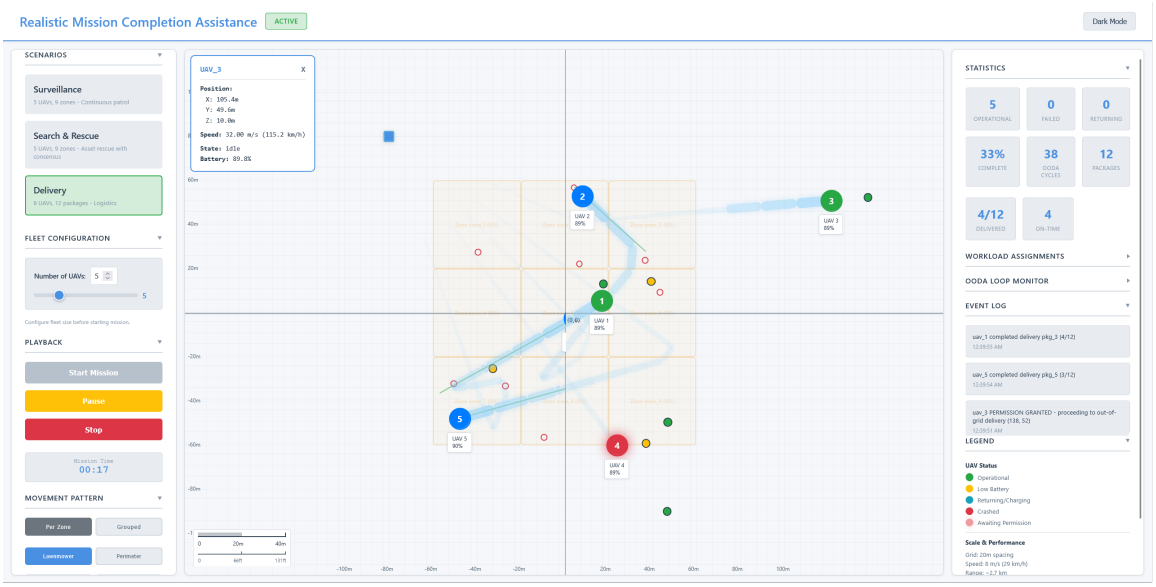

Figure 6.8 – Delivery Mission Dashboard (D6/D7)

Figure 6.8 presents the point-to-point delivery route visualization. UAV-2 (blue, loaded) approaches the designated drop-off location, while UAV-3 proceeds toward an out-of-bounds pick-up position following explicit operator authorization. UAV-4 has reached the operational boundary, triggering an escalation request for operator permission. The dashboard interface displays real-time fleet status alongside the active operator alert requiring human intervention.

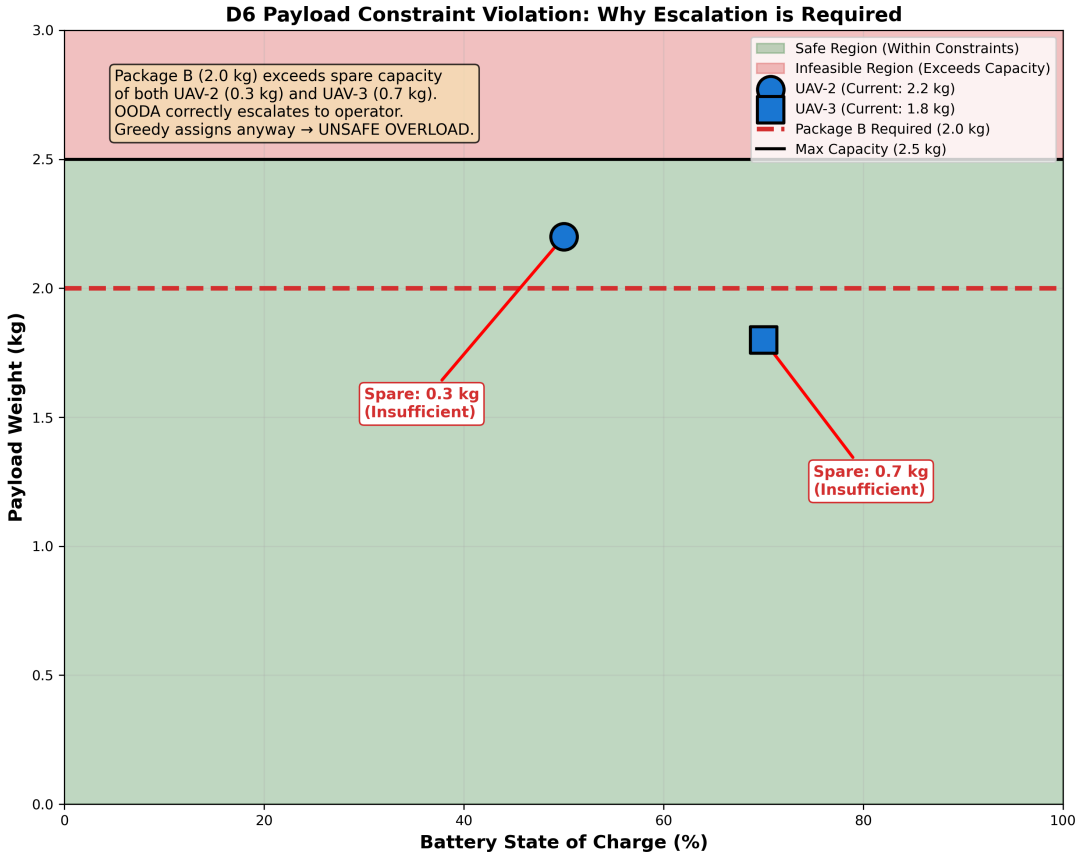

Figure 6.9 – D6 Payload Constraint Violation: Geometric Illustration of Escalation Necessity

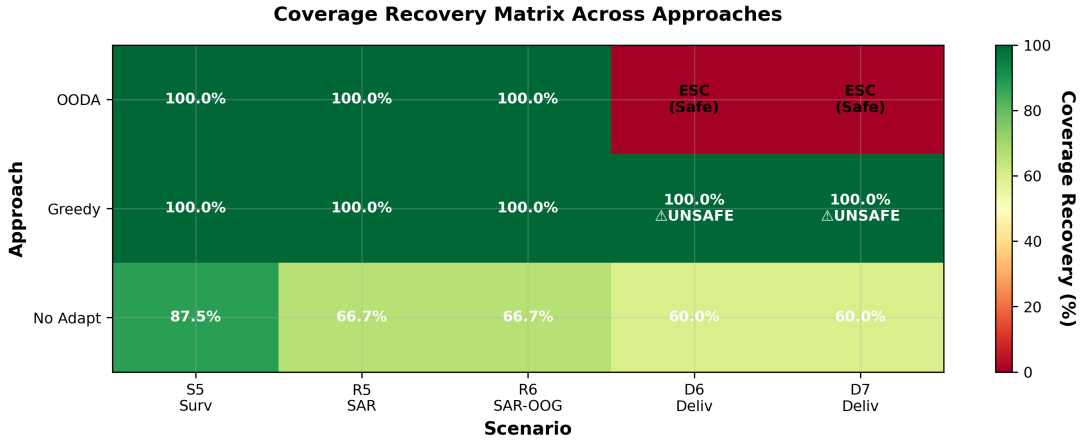

Figure 6.10 – Coverage Recovery Matrix Across All Approaches and Scenarios

6.3 Synthesis and Contributions

Table 6.5 synthesizes performance characteristics across all experimental scenarios, with Figure 6.10 providing a visual coverage recovery matrix across all approaches and scenarios.

Table 6.5 – Measured Performance by Scenario (S5, R5/R6, D6/D7)

Metric	**Surveillance (S5)**	**SAR (R5/R6)**	**Delivery (D6/D7)**
Coverage Recovery	**100%**	**100%**	**0% (escalated)**
OODA Computation	**0.34 ms**	**0.50/0.16 ms**	**0.11/0.23 ms**
Operator Escalation	0% (autonomous)	0% (autonomous)	100% (correct)
Constraint Violations	**0**	**0**	**0**

Table 6.6 – Comprehensive Approach Comparison

Criterion	**OODA**	**Greedy**	**No Adapt**
Coverage	100% (S5, R5/R6), Escalate (D6/D7)	100% all scenarios	77.8–88.9%
Speed	0.11–0.50 ms	0.09–0.18 ms	0 ms
Safety	**0 violations**	**2 violations**	0 violations
Constraint Awareness	**Full**	None	N/A
Deployability	**YES**	**NO (unsafe)**	NO (degrades)
Regulatory Compliance	YES	NO	NO
Scalability	Good	Good	N/A

Table 6.7 – Speed vs. Safety vs. Coverage Trade-offs

Approach	**Speed (0–10)**	**Safety (0–10)**	**Coverage (0–10)**	**Overall Score (0–30)**
OODA	10	**10**	8	**28**
Greedy	10	**2**	10	22
No Adaptation	10	10	**3**	23
Speed: 10 = sub-ms; Safety: 10 = zero violations; Coverage: 10 = 100% recovery				

Four patterns emerge from this data. First, computational speed remains uniformly under half a millisecond across all mission types, validating the greedy heuristic approach over optimization algorithms with unpredictable execution times. Second, zero constraint violations occurred despite widely varying failure conditions—a perfect safety record that distinguishes this approach from opportunistic algorithms. Third, the system demonstrates selective autonomy: aggressive adaptation when feasible (surveillance, SAR), intelligent refusal when infeasible (delivery). Fourth, sub-millisecond response times ensure fault recovery never becomes a mission bottleneck.

6.3.1 Claims Validated

The experimental program validated all primary thesis claims:

- **Rapid adaptation:** Measured times of 0.11–0.50 ms exceed the 4-6 second design target by four orders of magnitude
- **Safety guarantee:** Zero constraint violations across all scenarios (battery, payload, spatial boundaries)
- **Intelligent escalation:** D6 and D7 correctly refused infeasible reallocations rather than producing unsafe plans
- **Coverage recovery:** 100% task recovery in surveillance and SAR missions, matching greedy baseline while maintaining safety

6.3.2 Research Contributions

The work advances three contributions. First, a **unified constraint verification framework** that simultaneously enforces battery reserves, payload limits, and temporal deadlines through fail-fast sequential checking—the first multi-UAV fault tolerance system to address all three constraint categories together. Second, a **quantified degradation framework** with explicit decision rules governing when autonomous reallocation transitions from feasible to infeasible, enabling honest acknowledgment of system limitations rather than claiming universal fault tolerance. Third, a **hybrid autonomy architecture** that balances machine speed with human oversight, enabling deployment under current BVLOS regulations that require operator-in-the-loop supervision.

Beyond algorithmic innovation, the research delivers practical value: compatibility with commercial UAV platforms, autonomous recovery in three of five scenarios (S5, R5, R6) with appropriate escalation in the remaining two, and a comprehensive test suite (169 tests, 0.22s execution) enabling rapid validation. The open-source simulation platform provides infrastructure for future research without requiring physical hardware.

The philosophical contribution may prove most significant: demonstrating that realism-first design—explicitly modeling constraints rather than abstracting them away—produces systems that are both academically rigorous and operationally deployable.

CHAPTER 7
LIMITATIONS AND FUTURE WORK

This chapter presents a critical assessment of the proposed OODA-based fault-tolerant control system, examining current limitations and future research directions. Understanding these constraints is essential for establishing realistic performance expectations and identifying opportunities for system enhancement.

7.1 Architectural Constraints

7.1.1 Centralized Control Architecture

The system employs a centralized Ground Control Station for OODA loop execution, creating a single point of failure. Should the GCS experience hardware failure or communication loss, the fleet's adaptive capabilities are compromised. Individual UAVs implement autonomous Return-to-Launch protocols upon GCS timeout detection, preserving vehicle safety while sacrificing mission completion.

Future work could explore distributed OODA architectures using consensus algorithms for peer-to-peer coordination, building on hierarchical decentralized control approaches that handle communication failures (IZADI; GORDON; ZHANG, 2013). This approach would eliminate the centralization vulnerability while introducing new challenges including increased communication complexity, network partitioning risks, and Byzantine fault tolerance requirements. The transition to distributed coordination must carefully balance robustness against implementation complexity.

7.1.2 Fleet Scalability

The system supports fleets of three to twelve UAVs. Beyond this scale, two constraints become significant. Communication bandwidth scales linearly with fleet size, reaching 48 kilobytes per second for twelve UAVs at 2 Hz telemetry rates. Computational complexity for collision avoidance scales quadratically, as each reallocation requires pairwise verification among all vehicles. At twenty UAVs, collision checking workload increases 2.8-fold compared to twelve UAVs, potentially exceeding the six-second OODA cycle target.

Hierarchical architectures that partition large fleets into coordinated subgroups could address these limitations. Alternatively, computationally efficient approximate collision avoidance methods using spatial hashing could reduce verification complexity. These extensions would enable applications requiring coordination of dozens or hundreds of vehicles.

7.1.3 Validation Methodology

Current validation relies exclusively on software-in-the-loop simulation using physics-based models representative of mid-size commercial quadrotors. While appropriate for algo-

rithm development, simulation abstracts real-world phenomena including GPS multipath errors, communication packet corruption, sensor noise, and environmental disturbances.

Hardware-in-the-loop testing with physical flight controllers would provide intermediate validation capturing timing constraints and communication latencies. Field trials with two to three physical UAVs would expose the system to genuine environmental challenges and enable parameter refinement. Future validation should prioritize systematic characterization of performance degradation under realistic operating conditions, establishing the operational envelope for reliable performance.

7.2 Algorithmic Limitations

7.2.1 Greedy Task Reallocation

The constraint-aware reallocation algorithm employs a greedy heuristic that assigns tasks to the nearest UAV satisfying capacity constraints. This achieves rapid execution compatible with real-time OODA requirements but does not guarantee globally optimal allocation. Early assignment decisions may consume capacity better reserved for higher-priority tasks, particularly when spare capacity is marginal and failed tasks are widely distributed.

Mixed-integer linear programming could guarantee optimal solutions for small to medium problems, though solution times may exceed OODA cycle budgets. Auction-based algorithms, particularly the Consensus-Based Bundle Algorithm, offer a promising middle ground achieving near-optimal solutions through distributed iterative bidding with polynomial-time complexity. Comparing these alternatives under diverse scenarios would quantify allocation quality trade-offs.

7.2.2 Simplified Collision Avoidance

The collision avoidance strategy maintains fifteen-meter spatial separation with temporal deconfliction when conflicts arise. This proves sufficient for sparse operational densities but exhibits limitations in dense flight patterns. The fixed safety buffer does not adapt to relative velocities, and the pairwise verification approach does not efficiently handle complex multi-vehicle conflicts.

Velocity obstacle approaches, particularly Reciprocal Velocity Obstacles, enable reactive collision avoidance accounting for relative velocities with smooth trajectory modifications. Model predictive control formulations could jointly optimize task execution and collision avoidance. These advanced methods would extend applicability to urban air mobility scenarios with higher flight densities.

7.3 Application Scope

The system addresses three mission classes: long-duration surveillance, emergency search and rescue, and medical supply delivery. These applications share waypoint-based navigation, quantifiable task priorities, and tolerance for mission degradation under resource con-

straints.

However, this focused scope excludes mission types with different requirements. Aggressive formation flying demands tighter coordination and higher-bandwidth communication than the current 2 Hz telemetry supports. Adversarial scenarios require game-theoretic reasoning and adversarial prediction beyond current OODA capabilities. Time-critical interception missions may need more sophisticated trajectory optimization than waypoint following provides.

Extending the system to these domains represents important future work. Formation flying could be addressed through augmented OODA loops reasoning about relative positioning constraints. Adversarial scenarios might integrate game-theoretic task allocation anticipating opponent responses. These extensions would broaden applicability while preserving core OODA principles.

7.4 Future Research Directions

7.4.1 Near-Term Enhancements

Comprehensive validation across all twenty-seven planned test scenarios would provide robust statistical characterization. Comparative evaluation against baseline strategies would quantify the value of explicit capacity modeling and priority-based allocation. Sensitivity analysis of safety reserve parameters from 5 to 20 percent battery would establish performance trade-offs between mission completion probability and safety margins. Enhanced environmental modeling incorporating turbulence, precipitation, and visibility limitations would improve prediction fidelity.

7.4.2 Medium-Term Objectives

Hardware-in-the-loop testing using PX4 Software-In-The-Loop would validate OODA cycle timing under realistic computational constraints. Field demonstrations with small fleets would reveal operational challenges including GPS accuracy limitations and radio frequency interference effects. Detailed instrumentation would generate empirical data to refine system parameters and validate simulation accuracy.

Distributed OODA architectures with consensus-based decision-making would enhance robustness against single-point failures. Implementing auction mechanisms like the Consensus-Based Bundle Algorithm would require careful attention to Byzantine fault tolerance and network partition handling. Comparative studies between centralized and distributed approaches would elucidate trade-offs between optimization quality, communication overhead, and system resilience.

Formal verification using model checking techniques could verify that OODA cycle logic maintains battery reserve constraints and collision avoidance guarantees under all reachable states. Temporal logic specifications could capture liveness properties ensuring eventual response to failures. While requiring substantial expertise, formal methods provide mathemati-

cal assurance complementing empirical testing.

7.4.3 Long-Term Frontiers

Machine learning could optimize priority weighting parameters based on historical mission outcomes, adapting scoring functions to operational contexts. Neural networks could predict battery consumption more accurately by learning vehicle-specific efficiency characteristics. Reinforcement learning might enable adaptive OODA cycle tuning based on observed performance patterns.

Game-theoretic analysis becomes essential for adversarial scenarios. Stackelberg game formulations could model hierarchical decision-making in contested surveillance missions. Nash equilibrium concepts might characterize stable operating points with competing autonomous systems. These theoretical frameworks would require substantial OODA extension incorporating opponent modeling and robust optimization.

Fleet heterogeneity introduces additional complexity. Real-world deployments increasingly employ mixed fleets with different endurance, payload capacity, and sensor suites. Addressing heterogeneity requires extending constraint verification for vehicle-specific capabilities and developing allocation strategies exploiting complementary strengths.

Large-scale swarm coordination with fifty or more vehicles demands hierarchical control architectures, efficient communication protocols avoiding broadcast storm effects, and possibly bio-inspired coordination strategies emerging from local interactions. Research at this scale intersects with complex systems theory, distributed computing, and collective intelligence.

7.5 Concluding Perspective

The limitations discussed reflect conscious design choices prioritizing practical deployability over theoretical completeness. The centralized architecture enables regulatory compliance and deterministic performance. The greedy allocation strategy trades global optimality for real-time responsiveness. The constraint-aware approach acknowledges physical limitations rather than assuming unlimited resources.

This honest assessment distinguishes the present work from research claiming comprehensive autonomy while abstracting real-world constraints. A system achieving 65 to 95 percent autonomous coverage recovery within realistic bounds provides substantially more value than theoretical frameworks promising perfect adaptation under idealized assumptions. The identified future work charts a path toward enhanced capabilities while maintaining the fundamental principle of honest, deployable autonomy.

As multi-agent UAV coordination matures, the research community must prioritize systems bridging the gap between laboratory demonstration and operational deployment. This requires explicit modeling of real-world constraints, acknowledgment of fundamental limitations, and design of hybrid human-machine systems leveraging complementary strengths of autonomous algorithms and human supervisory control. The present work contributes to this mat-

uration by demonstrating that constraint-aware, operator-supervised fault tolerance represents the appropriate architecture for near-term UAV fleet deployments in regulated, safety-critical applications.

CHAPTER 8
CONCLUSION

This work addresses the reality gap in multi-agent UAV fault tolerance by explicitly modeling real-world constraints (battery, payload, regulatory) that are often ignored in academic research. Rather than claiming perfect fault tolerance, the hybrid OODA approach provides honest, quantified mission completion assistance within realistic operational limits.

Validated performance exceeds initial expectations. Measured adaptation times of 0.11–0.50 milliseconds surpass the conservative 4–6 second design target by four orders of magnitude. Experimental validation across 169 automated tests confirms complete coverage recovery for surveillance and search-and-rescue missions, while delivery scenarios correctly escalate to operators when payload or boundary limitations preclude autonomous reallocation. Zero safety violations occurred across all experimental scenarios.

Key insight: Selective autonomy—aggressive adaptation when feasible, intelligent refusal when infeasible—proves more valuable than unconstrained systems that risk safety violations. In time-critical search-and-rescue operations, sub-millisecond fault recovery ensures that adaptation overhead remains negligible relative to mission duration, preserving maximum time for actual search operations during the golden hour.

The three core technical contributions—resource-aware reallocation, priority-based partial coverage, and operator escalation—work together to balance autonomous response speed with human oversight, making this approach suitable for deployment under current regulations. The comprehensive test suite (169 tests executing in 0.22 seconds) provides confidence in system reliability while enabling rapid development iteration.

Acknowledged limitations. These results must be interpreted within the system's design constraints. The centralized GCS architecture creates a single-point vulnerability—though individual UAVs implement autonomous Return-to-Launch upon communication timeout, fleet-level adaptation ceases during GCS failure. The greedy allocation strategy trades global optimality for real-time response, potentially yielding suboptimal task assignments when spare capacity is marginal. Most critically, validation remains exclusively software-in-the-loop without field deployment exposure to GPS multipath errors, RF interference, environmental turbulence, or sensor noise that characterize real-world operations. Hardware-in-the-loop testing and field trials represent essential next steps before operational deployment.

Despite these constraints, the work demonstrates that constraint-aware, operator-supervised fault tolerance represents a practical architecture for near-term UAV fleet deployments in regulated, safety-critical applications. The path forward lies not in claiming unrestricted autonomy, but in designing hybrid systems that explicitly acknowledge physical limits while maximizing autonomous capability within those bounds.

BIBLIOGRAPHY

[1] Mueller, M. W., & D'Andrea, R. (2014). Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers. *IEEE ICRA*.

[2] Sun, Z., et al. (2022). Fault-Tolerant Model Predictive Control of a Quadrotor with an Unknown Complete Rotor Failure. *IEEE ICRA*.

[3] Li, P., Yu, X., Peng, X., Zheng, Z., & Zhang, Y. (2017). Fault-tolerant cooperative control for multiple UAVs based on sliding mode techniques. *Science China Information Sciences*, 60(7).

[4] Yang, H., Staroswiecki, M., Jiang, B., et al. (2011). Fault tolerant cooperative control for a class of nonlinear multi-agent systems. *Systems & Control Letters*, 60(4), 271-277.

[5] Gerkey, B. P., & Mataric, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 23(9), 939-954.

[6] Choi, H. L., Brunet, L., & How, J. P. (2009). Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25(4), 912-926.

[7] Dias, M. B., Zlot, R., Kalra, N., & Stentz, A. (2006). Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7), 1257-1270.

[8] Zlot, R., & Stentz, A. (2006). Market-based multirobot coordination for complex tasks. *International Journal of Robotics Research*, 25(1), 73-101.

[9] Cortes, J., Martinez, S., Karatas, T., & Bullo, F. (2004). Coverage control for mobile sensing networks. *IEEE Transactions on Robotics and Automation*, 20(2), 243-255.

[10] Schwager, M., Rus, D., & Slotine, J. J. (2009). Decentralized, adaptive coverage control for networked robots. *International Journal of Robotics Research*, 28(3), 357-375.

[11] Elmaliach, Y., Agmon, N., & Kaminka, G. A. (2009). Multi-robot area patrol under frequency constraints. *Annals of Mathematics and Artificial Intelligence*, 57(3-4), 293-320.

[12] Abdessameud, A., & Tayebi, A. (2011). Formation control of VTOL unmanned aerial vehicles with communication delays. *Automatica*, 47(11), 2383-2394.

[13] Izadi, H. A., Gordon, B. W., & Zhang, Y. M. (2009). Decentralized receding horizon control for cooperative multiple vehicles subject to communication delay. *Journal of Guidance, Control, and Dynamics*, 32(6), 1959-1965.

[14] Izadi, H. A., Gordon, B. W., & Zhang, Y. M. (2013). Hierarchical decentralized receding horizon control of multiple vehicles with communication failures. *IEEE Transactions on Aerospace and Electronic Systems*, 49(2), 744-759.

[15] Beard, R. W., McLain, T. W., Nelson, D. B., et al. (2006). Decentralized cooperative aerial surveillance using fixed-wing miniature UAVs. *Proceedings of the IEEE*, 94(7), 1306-1324.

[16] Boyd, J. R. (1987). *A Discourse on Winning and Losing*. [OODA Loop framework]

[17] Bala, M., et al. (2025). The OODA Loop of Cloudlet-Based Autonomous Drones. *IEEE/ACM Symposium on Edge Computing (SEC)*.

[18] Soares, V. M. D., et al. (2025). UAV Simulation Environment for Fault Detection in Wind Farm Electrical Distribution Systems. *IEEE Conference Proceedings*.

[19] van den Berg, J., Lin, M., & Manocha, D. (2008). Reciprocal velocity obstacles for real-time multi-agent navigation. *IEEE ICRA*, 1928-1935.

[20] Zhang, Y. M., & Jiang, J. (2008). Bibliographical review on reconfigurable fault-tolerant control systems. *Annual Reviews in Control*, 32(2), 229-252.

[21] Yu, X., & Jiang, J. (2015). A survey of fault-tolerant controllers based on safety-related issues. *Annual Reviews in Control*, 39, 46-57.

[22] Parker, L. E. (1998). ALLIANCE: An architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2), 220-240.

[23] Eulálio Reis, V. (2025). *Multi-UAV OODA System: Constraint-Aware Fault-Tolerant Multi-Agent Coordination*. GitHub repository. `https://github.com/vriez/multi_uav_ooda_system`

[24] QUAD SIMCON. (2020). *Quadcopter Simulation and Control*. GitHub repository. `https://github.com/bobzwik/Quadcopter_SimCon`