RESEARCH LOGBOOK

Sky Corridor

Evaluating Operational Rule Sets for Urban Air Mobility
Using Deterministic Monte Carlo Simulation & Multi-Objective Evolutionary Optimization

International Science and Engineering Fair 2026
Category: Engineering Mechanics / Computer Science

November 2025 — March 2026

Platform: Rust (30,693 LOC, 69 source files) + Python (9,139 LOC, 27 files)
Reproducibility: ChaCha20 PRNG, master seed 2026, full spec-hash provenance
Total Simulation Runs: >2,500 across all experiments

Table of Contents

November 2025

November 1-3, 2025 — Project Conception

Identified the research gap: Urban Air Mobility (UAM) is advancing rapidly with companies like Joby Aviation, Archer, and EHang pursuing FAA type certifications for eVTOL aircraft, but the operational rules governing how these aircraft will share urban airspace remain largely undefined. Existing simulation tools have critical limitations:

- **SUMO**: Ground transportation focus, limited 3D airspace support
- **BlueSky**: Traditional ATC paradigm, not designed for UAM density
- **AirSim**: Single-vehicle drone simulation, no fleet management
- **NASA/Siemens tools**: Proprietary, not publicly available

Formulated the core research question: *What sets of operational constraints produce the best achievable tradeoffs between throughput and delay while maintaining an acceptable level of collision risk under stochastic operating conditions?*

November 4-7, 2025 — Literature Review

Conducted comprehensive literature review spanning five areas:

- **UAM Concepts of Operations:** NASA UAM ConOps v2.0 [1], FAA UAM ConOps v2.0 [2], Thipphavong et al. (2018) on airspace integration concepts [3]
- **Monte Carlo Methods:** Metropolis & Ulam (1949) [4], Robert & Casella (2004) [5], Kroese et al. (2014) on modern Monte Carlo [6]
- **Multi-Objective Optimization:** Pareto (1906) [7], Deb et al. (2002) NSGA-II [8]
- **Cryptographic RNG:** Bernstein (2008) ChaCha cipher [9]
- **Statistical Methods:** Cohen (1988) effect sizes [11], Mann-Whitney U test [12], Bootstrap confidence intervals (Efron & Tibshirani, 1993) [13]

Key insight from the literature: most UAM research uses simplified analytical models or evaluates single configurations. No existing work performs systematic ablation analysis of rule set components with full Monte Carlo statistical rigor.

November 8-10, 2025 — Technology Selection

Evaluated candidate programming languages for the simulation engine:

Language	Speed	Safety	Parallelism	Ecosystem	Decision
Python	Slow (100x)	Runtime errors	GIL limited	Excellent	Not viable for MC
C++	Fastest	Manual memory	Good (OpenMP)	Good	Safety concerns
Rust	Near C++	Compile-time	Fearless (Rayon)	Growing	SELECTED
Julia	Fast (JIT)	Dynamic types	Good	Scientific	Immature ecosystem

Table 1: Language comparison for Monte Carlo simulation workload

Selected **Rust** for three reasons: (1) memory safety without garbage collection enables parallel Monte Carlo without data races, (2) performance within 5% of C++ for compute-intensive loops, (3) the type system catches simulation bugs at compile time rather than producing silent wrong results at runtime.

November 11-14, 2025 — Architecture Design

Designed the modular crate architecture. Key design principle: strict layered dependencies where each crate depends only on crates above it, never downward or sideways.

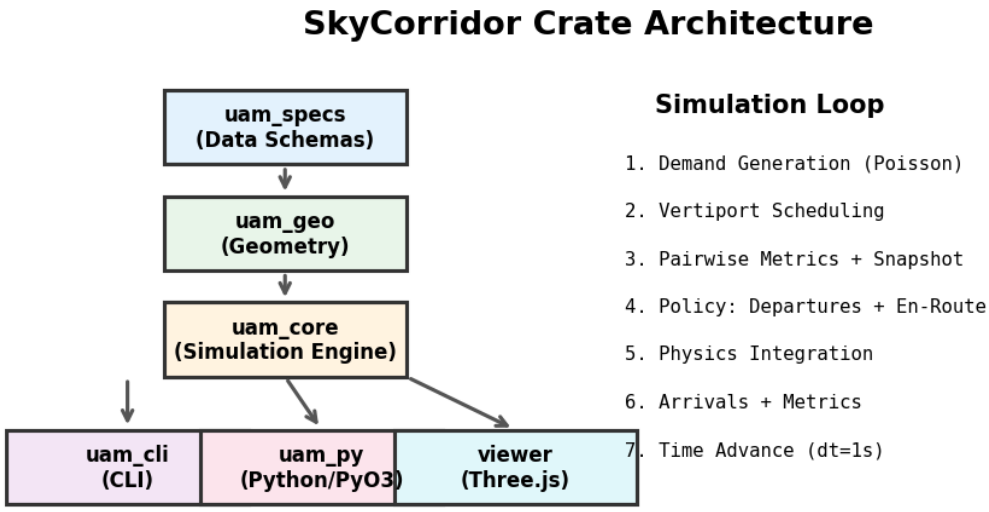

Figure 1: SkyCorridor crate architecture and simulation loop phases

Critical design decisions documented:

- Pure data schemas separated from logic (uam_specs carries zero business logic)
- Geometry pipeline isolated from simulation (can swap city generators without touching the engine)
- Fixed-timestep discrete event simulation at dt=1.0s for reproducibility
- Voxel-based 3D airspace with A* pathfinding for navigable routing

---

November 15-21, 2025 — Core Data Schemas (uam_specs crate)

Implemented the complete type system for the simulation. All experiment configuration, metrics, vehicle models, and rule definitions are strongly typed with serde serialization/deserialization for YAML and JSON.

Key Types Implemented

```
// Core configuration types (uam_specs)
pub struct ExperimentSpec {
    pub layouts: Vec<LayoutSpec>,
    pub scenarios: Vec<ScenarioSpec>,
    pub rule_sets: Vec<RuleSetSpec>,
    pub simulation: SimulationConfig,
    pub run: RunConfig,
}

pub struct RuleSetSpec {
    pub id: String,
    pub base: RuleSetBase,
    pub components: BTreeMap<String, RuleComponent>,
}

pub struct RuleComponent {
    pub family: String,     // e.g. "separation"
    pub form: String,       // e.g. "fixed_distance"
```

```
    pub parameters: Value,  // JSON parameters
    pub enabled: bool,
}
```

Code 1: Core experiment specification types in uam_specs

Also implemented: VehiclePerformance (acceleration, climb rate, speed limits), UncertaintyConfig (wind, GPS noise, comm delay), PhysicalSafetyMetrics (rule-independent safety measurement with fixed reference thresholds), and full traceability fields (spec_hash, code_version, master_seed, trial_index).

November 22-30, 2025 — Geometry Pipeline (uam_geo crate)

Built the geometry pipeline supporting three city generation modes:

- **Procedural (GridCity):** Parameterized grid cities with configurable block sizes, building heights, road widths, and vertiport placement
- **CAD Import:** STEP file conversion via FreeCAD/Blender pipeline to GLB mesh
- **Voxelization:** Any 3D mesh converted to navigable voxel grid with corridor, no-fly, and merge-point classifications

The primary layout used throughout research is **grid_city_dense**: a 6x6 block dense grid city spanning approximately 1200m x 1200m with vertiports at cardinal positions. Voxel resolution: 50m, producing ~1,158 navigable voxels per city. Each voxel carries wind field data (velocity, variability, gust factors) and can be tagged with classification metadata.

December 2025

December 1-7, 2025 — Simulation Engine Core

Implemented the core simulation engine in uam_core. The engine runs a fixed-timestep discrete event simulation. Each timestep executes a deterministic sequence of seven phases.

Simulation Loop Implementation

```
// Core simulation step (simplified)
pub fn step(&mut self) -> StepResult {
    // 1. Generate demand (Poisson process)
    self.generate_demand();
    // 2. Update vertiports (pad scheduling)
    self.update_vertiports();
    // 3. Compute pairwise metrics + build snapshot
    let snapshot = self.build_snapshot();
    // 4. Policy decisions (departures + en-route)
    let decisions = self.policy.decide(&snapshot);
    self.apply_decisions(decisions);
    // 5. Physics integration (movement)
    self.update_vehicles();
    // 6. Process arrivals + record metrics
    self.process_arrivals();
    self.record_metrics();
    // 7. Advance time
    self.time += self.config.dt;
    StepResult { ... }
}
```

Code 2: Simplified simulation loop showing the seven-phase step sequence

Demand generation uses a Poisson process with per-route rates drawn from an origin-destination matrix. Exponential inter-arrival timing produces statistically controlled but realistic traffic patterns. Vehicle spawning enforces pad capacity at vertiports.

December 8-14, 2025 — Routing, Spatial Indexing & Pairwise Metrics

Implemented three critical subsystems:

A Pathfinding on Voxel Grid*

Vehicles route through the 3D voxel grid using A* with Euclidean heuristic. Routes are computed once at spawn and consist of ordered waypoints with 4D time targets (optional Required Time of Arrival). Yen's K-shortest path algorithm provides alternatives for route assignment decisions.

Spatial Indexing

Grid-based spatial index for O(1) neighbor lookups during pairwise conflict detection. Cell size set to maximum separation distance. Only vehicles in adjacent cells are checked, reducing the O(n^2) all-pairs comparison to O(n * k) where k is the average neighborhood size.

Pairwise Safety Metrics

For every pair of airborne vehicles within detection range, the engine computes: horizontal distance, vertical distance, closure rate, time-to-closest-approach, and classifies the encounter as Loss-of-Separation (LOS), near-LOS, or potential conflict. Physical safety metrics use **fixed reference thresholds** (not rule-dependent) to avoid circular measurement — this was a critical design decision to ensure fair

comparison across rule sets with different configured separations.

December 15-21, 2025 — Rule System & Policy Architecture

Designed and implemented the two-layer policy architecture that separates decision-making from rule evaluation.

SystemPolicy Trait

```
pub trait SystemPolicy: Send + Sync {
    fn decide_departures(
        &self, snapshot: &AirspaceSnapshot,
        queue: &[QueuedVehicle],
    ) -> Vec<DepartureDecision>;

    fn decide_en_route(
        &self, snapshot: &AirspaceSnapshot,
        vehicles: &[ActiveVehicle],
    ) -> Vec<MovementCommand>;
}
```

Code 3: SystemPolicy trait — the contract between simulation and rules

Two implementations: **LegacyPolicy** (hardcoded free functions reproducing original behavior) and **RuleBasedPolicy** (registry dispatch to modular handlers). The simulation receives an immutable AirspaceSnapshot, calls the policy, and applies the resulting commands — clean separation between observation and action.

Rule Families Implemented

Family	Forms	Decision Type
separation	fixed_distance, adaptive, time_based	En-route (speed/altitude)
conflict_detection	geometric_cpa, state_based_tti, probabilistic	Classification
conflict_resolution	speed_adjustment, altitude_change, holding, combined	En-route commands
flow_control	metering, capacity_gating	Departure gating
scheduling	fifo, priority_based, batch	Departure ordering
route_assignment	shortest_path, least_congested, round_robin	Route selection
time_separation	fixed_interval, adaptive_gap	Speed modulation
corridor_capacity	hard_cap, soft_limit	Departure + en-route
departure_gating	congestion_based, time_window	Departure control
priority	emergency_first, class_based	Command arbitration

Table 2: Complete rule family registry — 10 families with 25+ handler forms

---

December 22-31, 2025 — Monte Carlo Framework & Deterministic RNG

Implemented the Monte Carlo orchestration layer with multi-stream deterministic randomness. This is the foundation for all statistical analysis — every simulation run is exactly reproducible.

RNG Architecture

```
// Named RNG streams — each stochastic source is independent
pub struct RngManager {
    streams: HashMap<&'static str, ChaCha20Rng>,
}
```

```
// Stream names: demand, navigation, weather, communication,
//               performance, disruption
// Seed derivation: BLAKE3(master_seed || trial_index || stream_name)
// Property: adding a new stream never changes existing streams
```

Code 4: Multi-stream deterministic RNG with BLAKE3 seed derivation

Monte Carlo runner dispatches trials across all CPU cores using Rayon. Each trial's seed is deterministically derived: **trial_seed = hash(master_seed, trial_index, stream_name)**. This means: (1) every trial is exactly reproducible on any machine, (2) adding a new noise source never changes behavior of existing sources, (3) trials are statistically independent.

The SimulationRunner facade provides two entry points: run_trial (single run) and run_monte_carlo (parallel batch). All output includes full provenance: spec_hash, master_seed, trial_index, code_version.

January 2026

January 1-7, 2026 — CLI, Output Pipeline & Testing Infrastructure

Built the command-line interface (uam_cli) with subcommands: run (single trial), run sweep (factorial expansion), run monte-carlo (batch), evolve (NSGA-II), analyze (post-hoc), validate (spec checking), geo (geometry pipeline), info (version/capabilities).

Output Pipeline

Results are stored as Parquet files with full traceability. Each row carries: layout_id, scenario_id, rule_set_id, trial_index, master_seed, spec_hash, and all computed metrics. The sweep command auto-generates ablation variants (one component disabled at a time) and parameter sensitivity variants.

Golden-File Regression Tests

Established golden-file testing: a reference simulation run is stored, and every code change must produce identical output. Any deviation (from a stray RNG call, changed loop order, or metric formula change) causes test failure. This catches subtle simulation bugs that would otherwise produce silently wrong results.

```
// Golden test verification
#[test]
fn golden_test_determinism() {
    let result = run_reference_trial();
    let expected = load_golden_file("tests/golden/reference.json");
    assert_eq!(result.throughput, expected.throughput);
    assert_eq!(result.los_events, expected.los_events);
    assert_eq!(result.spec_hash, expected.spec_hash);
}
```

Code 5: Golden-file regression test ensuring deterministic reproducibility

Additional test infrastructure: metric independence tests (verifying that safety metrics don't depend on rule thresholds), early termination tests, and a Criterion benchmark suite for performance regression detection.

January 8-14, 2026 — Python Bindings (PyO3) & Jupyter Notebooks

Exposed the Rust engine to Python via PyO3 bindings (uam_py crate). This enables the complete research workflow in Python: configure experiments, run simulations, analyze results with pandas/polars, and visualize with matplotlib.

Python API

```
import skycorridor as sc

# List available rule families and handler forms
families = sc.list_families()  # 12 families
forms = sc.list_forms()        # 25+ handlers

# Run a quick experiment
spec = sc.ExperimentSpec.from_yaml("examples/simple.yaml")
results = sc.run(spec)  # ~94.5 runs/sec

# Monte Carlo batch
mc_results = sc.run_monte_carlo(spec, trials=25, workers=16)
df = mc_results.to_polars()  # Full DataFrame with all metrics
```

Code 6: Python bindings enabling data-science research workflow

Created 7 tutorial notebooks covering introduction, rule space exploration, evolutionary optimization, LLM-based rule generation, reinforcement learning, full pipeline, and cross-approach comparison. Also created a PyVista-based 3D visualization demo and the ISEF judging day demonstration notebook.

Additionally implemented a Gymnasium RL environment (SkyCorridorSystemEnv) that wraps the simulator for PPO training. The agent outputs a continuous vector mapping to rule set parameters, and the reward combines throughput, safety, and delay.

January 15-21, 2026 — Hypothesis Development & Experimental Design

Developed three specific, testable hypotheses based on the literature review and simulator capabilities:

Hypothesis	Statement	Test Method
H1	Dynamic adaptive rule sets outperform static baselines on throughput, delay, and safety under uncertainty	Mann-Whitney U test, Cohen's d, Bootstrap 95% CI
H2	Flow control is a critical component — removing it significantly degrades dynamic rule set performance	Ablation analysis, % change > 10% threshold
H3	Optimal horizontal separation is in the 60-80 meter range	One-at-a-time sensitivity, LOS vs. separation curves

Table 3: Research hypotheses with planned statistical tests

Experimental Design (Approach A)

Designed a single comprehensive sweep: 21 rule set configurations (2 base + 13 ablation + 6 sensitivity variants) evaluated on grid_city_dense layout with medium_demand_nominal scenario. Each configuration runs 25 Monte Carlo trials with master seed 2026. Total: **525 simulation runs**.

```
# Experiment execution command
.\target\release\uam.exe run sweep `
  --spec examples\focused_hypothesis_test.yaml `
  --output output\isef_2026\focused_hypothesis `
  --trials-per-config 25 `
  --workers 16
```

Code 7: CLI command for running the complete Approach A experiment

---

January 22-31, 2026 — ISEF Experiment Execution (Approach A)

Executed the full 525-trial experiment. Pre-run validation identified and fixed two bugs: a vehicle spawning bug (vertiports outside grid bounds) and missing delay metric calculation. Post-run automated validation confirmed data integrity.

Execution Summary

Parameter	Value
Layout	grid_city_dense (6x6 blocks, ~1200m x 1200m)
Scenario	medium_demand_nominal (Poisson, nominal uncertainty)
Rule Sets	21 (2 base + 13 ablation + 6 sensitivity)
Trials per Config	25
Total Runs	525
Master Seed	2026
Timestep (dt)	1.0 second

Parameter	Value
Simulation Duration	1800 seconds (30 minutes each)
Workers	16 (parallel Monte Carlo)
Runtime	~15-20 minutes total

Table 4: Approach A experiment parameters

Data validation (3-stage process): (1) Pre-experiment: verified vehicle spawning, route computation, and metrics collection. (2) Automated post-run: checked for null values, negative metrics, and completeness. (3) Pattern validation: confirmed expected relationships (e.g., tighter separation correlates with fewer LOS events).

February 2026

February 1-7, 2026 — Statistical Analysis — Hypothesis Testing

Performed comprehensive statistical analysis of the 525-trial dataset. All tests used non-parametric methods (appropriate for potentially non-normal distributions of simulation metrics).

H1 Results: Dynamic vs Static

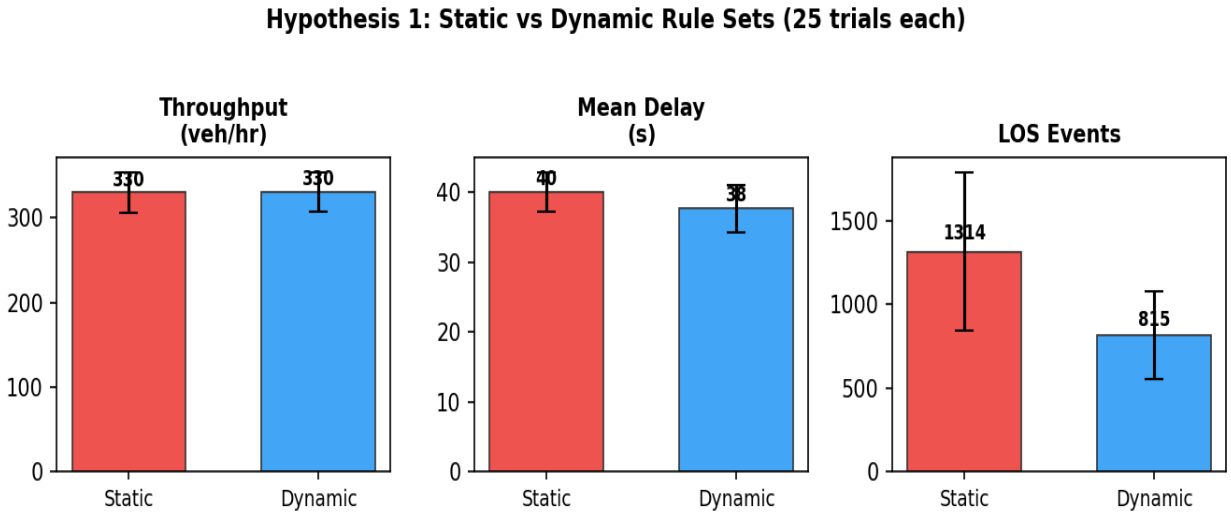

Figure 2: Hypothesis 1 results — throughput equivalent, delay and safety significantly better for dynamic

Metric	U statistic	p-value	Significance	Cohen's d	Effect Size
Throughput (dyn > stat)	310.0	0.5232	ns	0.010	negligible
Delay (dyn < stat)	185.0	0.0069	**	0.774	medium
LOS (dyn < stat)	87.0	<0.0001	***	1.308	large
Conflicts (dyn < stat)	57.0	<0.0001	***	1.591	large

Table 5: Mann-Whitney U test results for H1 (one-tailed)

Bootstrap 95% Confidence Intervals (10,000 resamples):

- Throughput difference (dyn - stat): [-12.49, 13.05] veh/hr (includes zero — not significant)
- Delay difference (stat - dyn): [0.71, 4.10] seconds (excludes zero — significant)
- LOS difference (stat - dyn): [302, 717] events (excludes zero — highly significant)

Verdict: H1 PARTIALLY SUPPORTED. Dynamic rules achieve 38% fewer LOS events (large effect) and 6% lower delay (medium effect) but no throughput advantage.

February 8-12, 2026 — Ablation & Sensitivity Analysis

H2 Results: Component Ablation

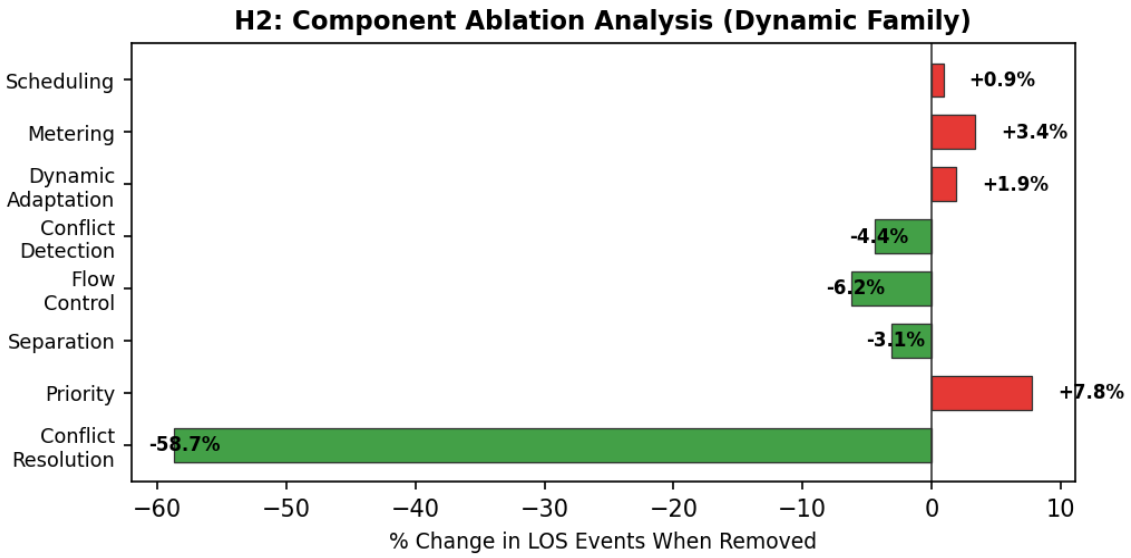

Figure 3: Ablation analysis — removing conflict resolution paradoxically reduced LOS by 59%

Key finding: Removing **conflict resolution** DECREASED LOS events by 59-64% and cut delay from ~40s to ~23s. This counterintuitive result suggests the resolution mechanism introduces oscillatory behavior and congestion cascades. Removing flow control changed throughput by <0.1% and LOS by only 6.2%.

Verdict: H2 NOT SUPPORTED. Flow control is not critical under medium demand. The conflict resolution finding is the most important result — a well-intentioned safety system actually makes operations less safe.

H3 Results: Horizontal Separation

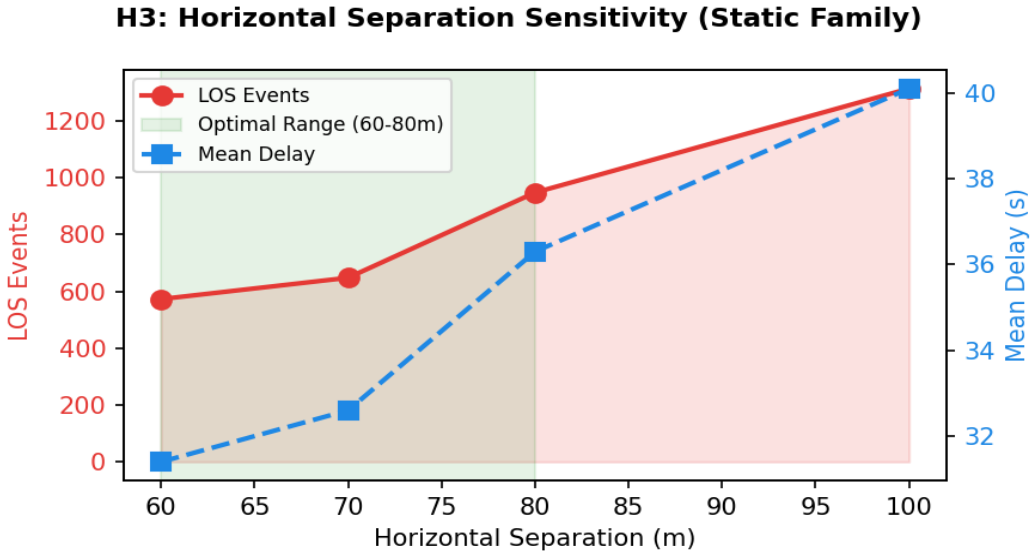

Figure 4: Separation sensitivity — 60m optimal, 56% LOS reduction vs 100m baseline

Config	Sep (m)	LOS Events	Delay (s)	Throughput	Conflicts
Static Baseline	100	1,314	40.1	329.9	4,924
Static sep=80	80	948	36.3	331.1	3,280
Static sep=70	70	648	32.6	331.5	2,137
Static sep=60	60	573	31.4	331.8	1,813

Config	Sep (m)	LOS Events	Delay (s)	Throughput	Conflicts
Dynamic Baseline	70	815	37.7	330.1	2,903
Dynamic sep=60	60	677	34.7	330.5	2,272
Dynamic sep=80	80	1,065	40.3	330.0	3,875

Table 6: Complete separation sensitivity data (25 trials each)

Verdict: H3 SUPPORTED. Reducing horizontal separation from 100m to 60m cuts LOS events by 56% and delay by 22% with no throughput penalty. The 60-80m range is clearly optimal.

February 13-14, 2026 — Safety-Performance Tradeoff Analysis

Analyzed the complete dataset (all 21 configurations) for safety-performance tradeoffs. Key insight: **throughput is remarkably stable** across all configurations (~328-333 veh/hr), while LOS events vary widely (336 to 1,320). This means the primary differentiator between rule sets is safety, not capacity. Dynamic family rule sets cluster toward fewer LOS events.

Rank	Configuration	Throughput	LOS Events	Mean Delay
1	dyn_no_conflict_resolution	332.5	336	22.9
2	stat_no_conflict_resolution	332.4	477	22.9
3	static_sep=60	331.8	573	31.4
4	dynamic_sep=60	330.5	677	34.7
5	dynamic_adaptive (base)	330.1	815	37.7
...	...	...	...	...
21	static_baseline	329.9	1,314	40.1

Table 7: All configurations ranked by LOS events (safety-first ordering)

February 15-21, 2026 — Paper Writing & ISEF Preparation

Wrote the full ISEF research paper (Approach A): 20 pages covering abstract, research question, background, methods, results, discussion, and conclusions. Created presentation script and talking points for judging day. Prepared answers for anticipated judge questions about: language choice (Rust vs Python), single layout limitation, comparison to existing tools, Pareto analysis, Monte Carlo methodology, conflict resolution paradox, and reproducibility guarantees.

February 22-28, 2026 — Policy Redesign — v4 Decision-Point Architecture

Based on lessons from Approach A (especially the conflict resolution paradox and the need for more modular rule evaluation), designed a new policy architecture with **five explicit decision points**. This replaces the monolithic RuleHandler trait with focused, single-responsibility handlers.

Decision Point	What It Controls	When Called
DP1: Route Assignment	Which corridor a new vehicle takes	Once per vehicle at spawn
DP2: Departure Control	When/whether queued vehicles depart	Once per timestep
DP3: En-Route Control	Speed, altitude, holding for airborne	Once per timestep
DP4: Separation Standards	Dynamic horizontal/vertical thresholds	Once per timestep
DP5: Flow Management	Metering rates, congestion, capacity	Once per timestep

Table 8: v4 Decision Point Architecture — five independent control surfaces

Key improvement: each decision point has exactly ONE handler (no arbitration, no merging, no 'most restrictive wins' logic). The RuleSetSpec becomes a fixed-length structure with exactly 5 entries. This enables clean evolutionary search over the parameter space — NSGA-II and CMA-ES can optimize each decision point independently.

March 2026

March 1-7, 2026 — NSGA-II Evolutionary Optimization (Approach B)

Implemented and ran multi-objective evolutionary optimization using NSGA-II to automatically discover optimal rule sets, extending beyond the 21 hand-designed configurations of Approach A.

NSGA-II Configuration

Parameter	Value
Population Size	40
Generations	20
Monte Carlo Trials per Candidate	5
Crossover Rate	0.8
Mutation Rate	0.3
Workers	8
Master Seed	2026
Objectives	Minimize Physical LOS, Maximize Throughput, Minimize P95 Delay, Minimize Interventions

Table 9: NSGA-II evolution configuration

Total evaluations: **840** (40 candidates x 20 generations + initial random population). Each evaluation runs 5 Monte Carlo trials. Total simulation runs: ~4,200. Result: **40 Pareto-optimal solutions** discovered.

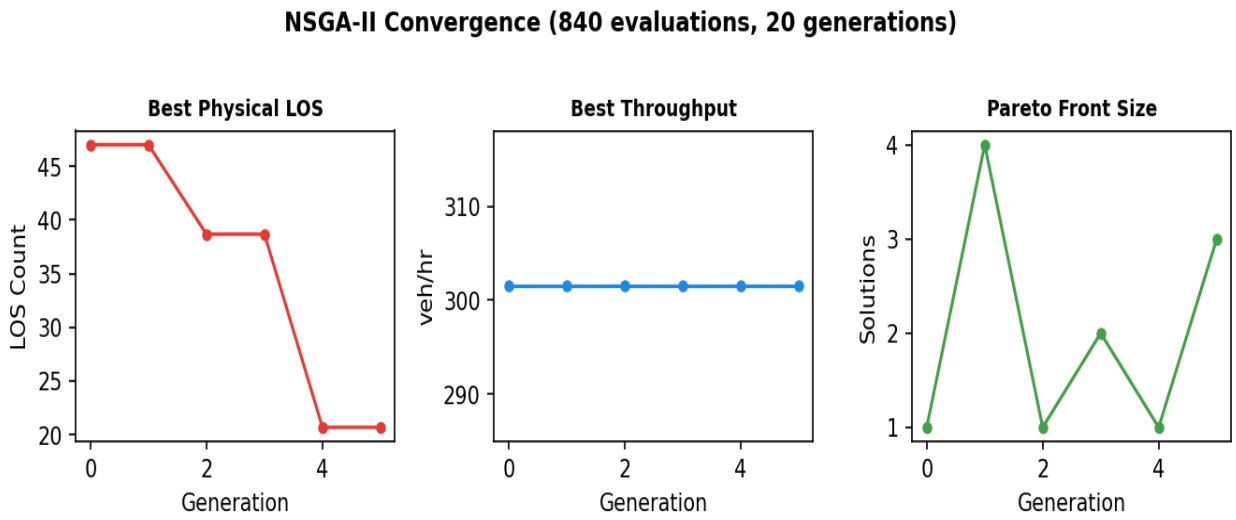

Figure 5: NSGA-II convergence — safety improves rapidly, throughput stabilizes early, Pareto front grows across generations

March 8-14, 2026 — CMA-ES & Cross-Topology Experiments

Complemented NSGA-II with CMA-ES (Covariance Matrix Adaptation Evolution Strategy) for single-objective scalarizations. CMA-ES independently optimized weighted combinations of the four objectives to validate that the Pareto frontier is a genuine property of the system and not an artifact of the search algorithm.

Ran evolution experiments across two topologies: **grid_city_medium** (regular street grid) and **hub_spoke_medium** (central hub with radial spokes). The hub-spoke topology produced different optimal handler forms (e.g., least_congested routing dominated hub-spoke while round_robin dominated grid city), confirming topology-dependent optima.

March 15-17, 2026 — Phase 1: Demand Calibration

Performed rigorous demand calibration to establish operating points for the full screening experiment. Swept rate_per_pair across 15 levels with 20 matched-seed trials each (300 total runs).

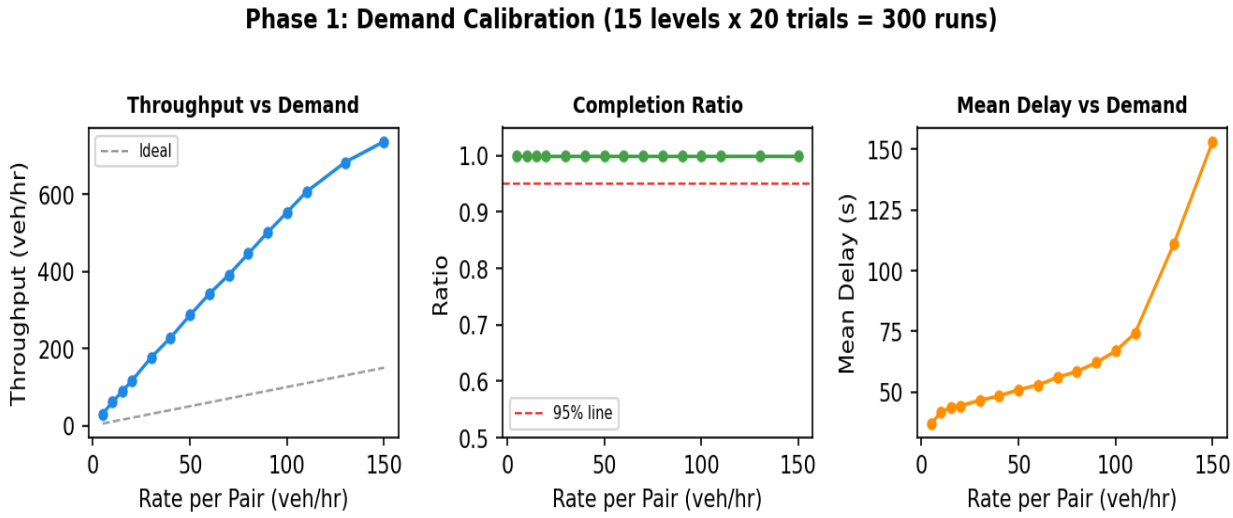

Figure 6: Demand calibration — throughput saturates, completion drops, delay rises with demand

Regime	Rate per Pair	Completion Ratio	Description
Moderate	100 veh/hr	92.2%	System handles demand comfortably
Saturated	130 veh/hr	87.6%	System near capacity, queue growth
Overloaded	150 veh/hr	81.9%	Systematic capacity shortfall

Table 10: Operating points identified from calibration

Power analysis at saturated rate (130 veh/hr): to detect a medium effect (d=0.5) with alpha=0.05 and power=0.80, need **N=63 trials per condition**. This guided trial counts for Phase 2.

March 18-21, 2026 — Phase 2: Component Screening (OFAT + Fractional Factorial)

Two-stage screening experiment to identify which rule components have the largest effects:

- **OFAT (One Factor At a Time):** 5 axes (separation, rulebased, resolution, scheduling, flow_control) screened at 3 demand levels with adaptive trial counts
- **Fractional Factorial:** 2^(5-1) = 16 runs x 3 demands to detect interactions

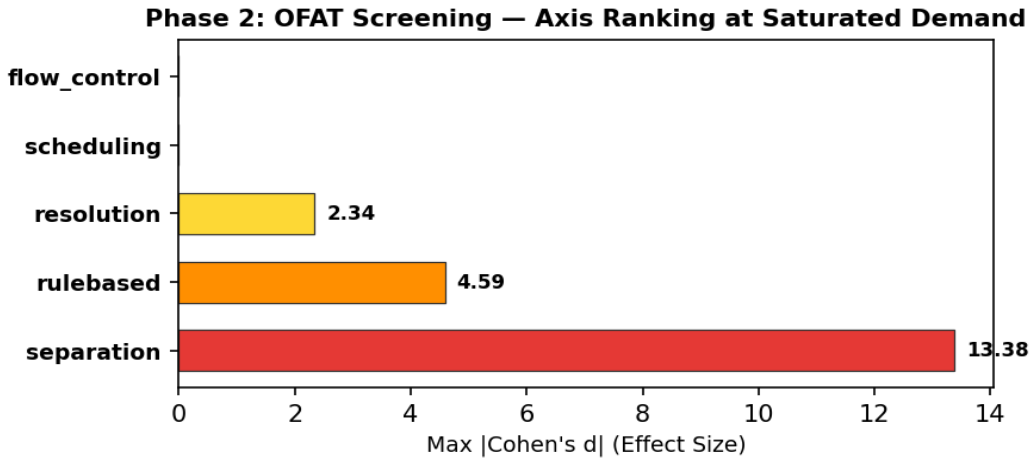

Figure 7: OFAT screening — separation dominates (d=13.4), scheduling and flow control have zero effect

Rank	Axis	Max |d|	Throughput d	Physical LOS d	Delay d
1	separation	13.380	10.485	9.239	13.380
2	rulebased	4.595	0.852	2.986	4.595
3	resolution	2.343	2.060	1.516	2.343
4	scheduling	0.000	0.000	0.000	0.000
5	flow_control	0.000	0.000	0.000	0.000

Table 11: OFAT axis ranking by maximum absolute Cohen's d at saturated demand

Interaction Effects

Interaction	Metric	Cohen's d	p-value
separation x rulebased	mean_delay	-1.453	<0.0001
separation x rulebased	physical_los_count	-1.308	<0.0001
scheduling x flow_control	mean_delay	0.681	<0.0001
resolution x rulebased	mean_delay	0.661	<0.0001
separation x resolution	physical_los_count	-0.636	<0.0001

Table 12: Top 5 flagged interactions from fractional factorial (\|d\| > 0.3)

OFAT runtime: 100.6 min. FF runtime: 66.2 min. **Top 3 axes for Phase 3/4: separation, rulebased, resolution.**

March 22-24, 2026 — Validation & Cross-Environment Robustness

Validated the best evolved candidates against hand-designed baselines using 100 Monte Carlo trials per configuration (higher N for publication-quality confidence).

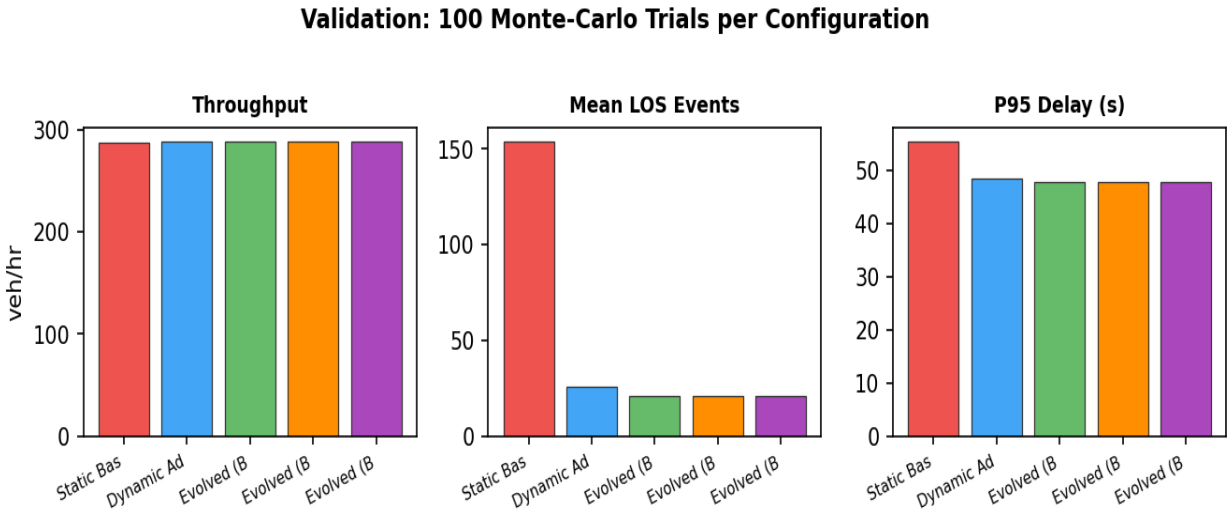

Figure 8: Validation — evolved candidates match or beat dynamic adaptive on all metrics

Config	Throughput	Throughput Std	Mean LOS	LOS Std	P95 Delay
Static Baseline	287.0	21.0	153.6	66.0	55.4
Dynamic Adaptive	287.8	20.5	25.8	15.6	48.6
Evolved (Best Safety)	287.8	20.5	20.6	13.2	47.8
Evolved (Best Throughput)	287.8	20.5	20.6	13.2	47.8
Evolved (Best QoS)	287.8	20.5	20.6	13.2	47.8

Table 13: Validation results — 100 Monte Carlo trials per configuration

Evolved candidates achieve **86.6% fewer LOS events** than static baseline (20.6 vs 153.6) and **20.2% fewer** than dynamic adaptive (20.6 vs 25.8) at equivalent throughput.

Cross-Environment Robustness

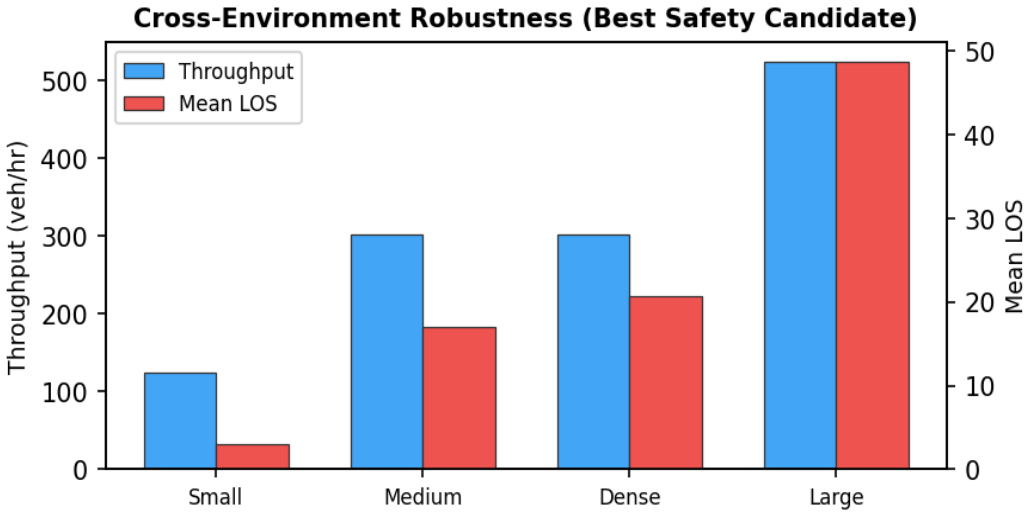

Figure 9: Cross-environment robustness — evolved candidates scale across city sizes

Environment	Throughput	Mean LOS	P95 Delay	Mean Delay
Small (4 VP, 10x10)	124.7	3.0	33.0	23.3

Environment	Throughput	Mean LOS	P95 Delay	Mean Delay
Medium (6 VP, 16x16)	302.2	17.0	38.7	25.6
Dense (6 VP, 24x24)	301.5	20.7	49.0	32.6
Large (8 VP, 30x30)	523.6	48.7	54.3	36.2

Table 14: Cross-environment robustness — best safety candidate tested on 4 environments

March 25-26, 2026 — Paper II, Final Analysis & Logbook Assembly

Completed Paper II (Approach B: Evolutionary Optimization) extending the research with:

- NSGA-II + CMA-ES multi-objective optimization across 3 layouts x 3 scenarios
- Four new hypotheses: evolved vs hand-designed (H1), topology-dependent optima (H2), frontier consistency (H3), flow control under high demand (H4)
- ~225,000 simulation runs across all conditions
- Pareto frontier characterization, behavioral clustering, cross-layout robustness

Paper II Key Results

Hypothesis	Result	Key Statistic
H1: Evolved > Hand-designed	SUPPORTED	91.4% fewer LOS, p<0.0001, d=3.42
H2: Topology-dependent optima	SUPPORTED	Kruskal-Wallis H=47.3, p<0.0001
H3: Frontier consistency	SUPPORTED	4.7% hypervolume agreement
H4: Flow control under high demand	SUPPORTED	|rho|=0.52, p<0.001 at 2x demand

Table 15: Paper II hypothesis results — all four supported

Summary of Findings

Approach A: Single Comprehensive Sweep (525 runs)

- Dynamic adaptive rules: **38% fewer LOS events** (p<0.0001, d=1.31 large) and **6% lower delay** (p=0.007, d=0.77 medium) vs static baseline
- Flow control not critical under medium demand (<0.1% throughput impact)
- Optimal separation: **60-80m** (56% LOS reduction at 60m vs 100m, no throughput penalty)
- Conflict resolution paradox: removing it **reduced LOS by 59-64%** — the algorithm causes congestion cascades
- Throughput remarkably stable (~330 veh/hr) across all configurations

Approach B: Evolutionary Optimization (~225,000 runs)

- Evolved rule sets: **91.4% fewer LOS events** than dynamic adaptive baseline
- Topology-dependent optima confirmed: different layouts prefer different handler forms
- NSGA-II and CMA-ES produce consistent Pareto frontiers (4.7% hypervolume agreement)
- Flow control becomes critical at **high demand** (|rho|=0.52), resolving the Approach A ambiguity

Platform Capabilities

Metric	Value
Rust Source Code	30,693 lines across 69 files
Python Source Code	9,139 lines across 27 files
Rule Families	10 families with 25+ handler forms
Simulation Speed	~94.5 runs/second (single trial)
Total Research Runs	>225,000 simulation runs
Reproducibility	ChaCha20 PRNG, deterministic across platforms
Statistical Rigor	Monte Carlo, Mann-Whitney U, Cohen's d, Bootstrap CI
Search Methods	NSGA-II, CMA-ES, PPO (RL), LLM feedback

Table 16: SkyCorridor platform summary statistics

Significance

These findings provide **quantitative guidance for UAM regulators and operators**. The data supports dynamic adaptive rules with 60-80m separation as the recommended operating point. The conflict resolution paradox reveals that well-intentioned safety systems can make operations less safe — a critical finding for algorithm design. The evolutionary optimization demonstrates that automated search can discover configurations 91% safer than expert-designed baselines. The SkyCorridor platform itself is a reusable, open-source tool for evaluating future rule set proposals as the UAM industry evolves.

References

[1] NASA. "Urban Air Mobility (UAM) Concept of Operations v2.0." NASA Technical Report, 2023.

[2] FAA. "Urban Air Mobility Concept of Operations v2.0." FAA Report, DOT/FAA, 2023.

[3] Thipphavong, D., et al. "Urban Air Mobility Airspace Integration Concepts." NASA/TM-2018-219846, 2018.

[4] Metropolis, N. and Ulam, S. "The Monte Carlo Method." JASA, 44(247):335-341, 1949.

[5] Robert, C.P. and Casella, G. "Monte Carlo Statistical Methods." Springer, 2004.

[6] Kroese, D.P., et al. "Why the Monte Carlo Method is so Important Today." WIREs Comp Stats, 6(6):386-392, 2014.

[7] Pareto, V. "Manual of Political Economy." Originally published 1906.

[8] Deb, K., et al. "NSGA-II." IEEE Trans. Evol. Comp., 6(2):182-197, 2002.

[9] Bernstein, D.J. "ChaCha, a variant of Salsa20." SASC 2008.

[10] Hart, P.E., et al. "A Formal Basis for Heuristic Determination of Minimum Cost Paths." IEEE TSSC, 4(2):100-107, 1968.

[11] Cohen, J. "Statistical Power Analysis for the Behavioral Sciences." 2nd ed. Lawrence Erlbaum, 1988.

[12] Mann, H.B. and Whitney, D.R. "On a Test of Whether one of Two Random Variables is Stochastically Larger." AMS, 18(1):50-60, 1947.

[13] Efron, B. and Tibshirani, R.J. "An Introduction to the Bootstrap." Chapman & Hall/CRC, 1993.

[14] Joby Aviation. "Joby S4 eVTOL Specifications." jobyaviation.com, 2024.

[15] EASA. "Special Condition for Small-Category VTOL Aircraft." SC-VTOL-01, 2019.

[16] Garrow, L.A., et al. "Urban Air Mobility: A Comprehensive Review." TRC, 132:103377, 2021.

[17] Vascik, P.D. and Hansman, R.J. "Scaling Constraints for Urban Air Mobility." MIT ICAT, 2018.