GNSS-Denied Search with Heterogeneous UAV-UGV Swarm

Rasheed Sheikh Muhammd Muneeb Hamid

13th December 2025

1 Project Summary: GNSS-Denied UAV Search Simulation

1.1 Overview

This project simulates and optimizes a **Cooperative UAV Search Mission** in a GNSS-denied environment. In such environments, UAVs cannot rely on GPS for positioning and suffer from navigation drift (accumulating position error) over time. To maintain effective coverage, UAVs must periodically visit support infrastructure (**UGVs** or **Beacons**) to reset their drift and recharge their batteries.

The goal is to find the optimal configuration of support nodes (UGVs and Beacons) and the optimal fleet size (5–13 UAVs) to cover more than 97% of the target area within 100 minutes at the lowest possible cost.

2 System Architecture

2.1 Initialization: Static Placement (MILP)

- **File**: `functions/milp_placement.py`
- **Algorithm**: Mixed-Integer Linear Programming (using `PuLP`)
- **Purpose**: Generates a mathematically optimal *initial* placement of UGVs and Beacons based purely on geometric coverage. This provides a suitable starting point for dynamic optimization.

2.2 Dynamic Simulation (The "Judge")

- **File**: `functions/mission_optimizer.py`
- **Algorithm**: Centroidal Voronoi Tessellation (CVT)

2.2.1 Key Mechanics

- **CVT Movement**: UAVs continuously move toward the centroid of their Voronoi cell, ensuring optimal area distribution.
- **Drift Modeling**: UAVs accumulate position error linearly (approximately 0.5 m/s) when outside the range of a support node.
- **Probabilistic Coverage**: A grid cell is considered covered only if confidence exceeds 95%. Confidence is a function of UAV drift.
- **State Machine**:

  $$\texttt{SEARCH} \rightarrow \texttt{RETURN\_DRIFT} \rightarrow \texttt{RETURN\_BATTERY} \rightarrow \texttt{CHARGING}$$

2.3 Optimization Pipeline (VNS)

- **File**: `functions/vns_optimizer.py`
- **Algorithm**: Variable Neighborhood Search (VNS)

2.3.1 Workflow

1. **Evaluation**: Each solution is tested using five Monte Carlo simulations with randomized initial positions and initial battery levels. A solution is considered valid only if at least 80% of runs succeed.
2. **Neighborhood Operators**:
   - **Swap**: Exchanges a selected node with an unselected candidate.
   - **Shift**: Moves a node to a nearby location.
   - **Shake**: Random perturbation to escape local optima.
   - **Add/Remove**: Adjusts the number of support nodes to balance cost and performance.

3 Key Constraints and Parameters

- **Area**: Square grid, default size 5500×5500 m (even though it can be changed in the constants.)
- **UAVs**: 5 to 13 agents
- **Drift Tolerance**: Drift must be reset when it reaches a threshold (e.g., 100 m), any areas scanned with drift equal to or greater than the threshold receive a zero increase in coverage confidence.
- **Battery**: UAVs compute full 3D energy costs and must physically return to a UGV for battery swapping
- **Cost Function**: Infrastructure cost (UGVS + Beacons) + UAV cost + Time based operational cost

4 Mathematical Formulation

4.1 Mixed-Integer Linear Programming (MILP)

The initialization phase solves a facility location problem to minimize infrastructure cost while ensuring coverage.

Objective Function

$$\min \sum_{i \in I} C_{UGV} x_i + \sum_{j \in J} C_{Beacon} y_j$$

Decision Variables

- $x_i \in \{0, 1\}$: UGV placement at candidate location i
- $y_j \in \{0, 1\}$: Beacon placement at candidate location j

Constraints

1. **Coverage Constraint**

$$\sum_{i \in I} a_{ik} x_i + \sum_{j \in J} b_{jk} y_j \geq 1 \quad \forall k \in K$$

2. **Minimum UAVs**

$$\sum_{u \in U} z_u \geq N_{\min}$$

3. **Battery Feasibility**

$$R_{\text{swap}} \leq \frac{E_{\text{cap}}(1 - R_{\text{reserve}}) v_{\min}}{P_{\text{avg}}}$$

4.2 Bayesian Coverage Update

The confidence of grid cell k is updated using a Bayesian rule.

Detection Probability

$$P_{\text{det}}(t) = \max\left(0, 1 - \frac{\delta(t)}{\delta_{\text{tol}}}\right)$$

Update Rule

$$P(C_k \mid z_t) = 1 - (1 - P(C_k \mid z_{t-1}))\,(1 - P_{\text{det}}(t))$$

A cell is considered covered when $P(C_k) \geq 0.95$.

4.3 Drift Dynamics

$$\delta(t + \Delta t) = \delta(t) + r_{\text{drift}} \Delta t$$

When entering support coverage:

$$\delta(t) \to 0$$

4.4 CVT Energy Function

$$J = \sum_{i=1}^{N} \int_{V_i} \rho(q) \|q - p_i\|^2 dq$$

$$p_i[k+1] = p_i[k] + k_p (C_i - p_i[k])$$

4.5 Battery Consumption Model

$$P(t) = \begin{cases} P_{\text{hover}} & v \approx 0 \\ P_{\text{cruise}} & v > 0 \\ P_{\text{climb}} & v_z > 0 \\ P_{\text{descent}} & v_z < 0 \end{cases}$$

$$E_{\text{total}} = \int_0^T P(t)\, dt$$

4.6 VNS Objective Function

$$J_{\text{VNS}} = w_1 C_{\text{infra}} + w_2 C_{\text{ops}} + w_3 T_{\text{mission}} + w_4 E_{\text{fleet}} + P_{\text{penalty}}$$

5 Outputs

- **JSON Results**: `vns_result.json`
- **Visualizations**:
  - `vns_solution_map.png`
  - `vns_coverage_heatmap.png`
  - `vns_dashboard.png`
  - `candidates_generated.png`
  - `milp_solution.png`
- **Animation**: Optional GIF of mission execution

6 Drift Plotting in VNS Dashboard

Each UAV tracks its own drift value. The maximum drift represents the worst-case drift among all UAVs at a given time step. Drift may exceed the threshold if a UAV does not visit a support node in time, as the drift variable is not explicitly capped.

7 Confidence Update Formula

When a UAV visits a grid cell, confidence is increased by:

$$\text{gain} = \max(0, 100 - \text{drift})$$

Higher drift results in lower confidence gain. A cell is considered covered when confidence reaches 95%.