Multi-Drone Agricultural Monitoring System

Theoretical Framework & Implementation Logic

Development Team

December 8, 2025

Contents

1 Introduction

This document outlines the theoretical foundations and decision-making logic behind the Multi-Drone Agricultural Monitoring System. The system deploys swarms of autonomous drones to monitor farmland, assess drought risks using meteorological data, and validate data integrity through multi-sensor fusion.

2 Theoretical Background

2.1 Drought Probability Model

The system estimates drought risk based on the methodology proposed in *"DroughtCast: A Machine Learning Forecast of the United States Drought Monitor"* (Brust et al., 2021).

Rather than relying on single inputs, the model aggregates multiple meteorological indices:

- **SPI (Standardized Precipitation Index)**: Measures rainfall deficit.
- **SMI (Soil Moisture Index)**: Assesses water availability in the root zone.
- **VCI (Vegetation Condition Index)**: Evaluates crop health via satellite imagery proxies.
- **TCI (Temperature Condition Index)**: Accounts for thermal stress.

The final probability $P_{drought}$ is derived from a weighted non-linear combination of these features, normalized to a $[0, 1]$ range using a logistic function:

$$P(y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \ldots + \beta_n X_n)}} \tag{1}$$

2.2 Sensor Fusion (Inverse-Variance Weighting)

To mitigate sensor noise and errors, the system employs **Inverse-Variance Weighting** for merging probabilities from multiple drones. This method gives more weight to precise sensors (low variance) and less weight to noisy ones.

Given a set of measurements x_i with known variances σ_i^2, the fused estimate $\hat{x}$ and its variance $\hat{\sigma}^2$ are calculated as:

$$w_i = \frac{1}{\sigma_i^2} \tag{2}$$

$$\hat{x} = \frac{\sum_i w_i x_i}{\sum_i w_i} \tag{3}$$

$$\hat{\sigma}^2 = \frac{1}{\sum_i w_i} \tag{4}$$

This ensures that the final fused probability is statistically the most reliable estimate possible given the available data.

2.3 Fault Detection Logic

The system automatically detects malfunctioning sensors using statistical hypothesis testing. A sensor reading x_{sensor} is flagged as **faulty** if it deviates significantly from the expected model prediction x_{model} or the consensus of other drones.

The fault condition is triggered if:

$$|x_{sensor} - x_{model}| > (2 \cdot \sigma_{noise} + \delta_{threshold}) \tag{5}$$

Where:

- σ_{noise} is the standard deviation of the sensor (noise floor).
- $\delta_{threshold}$ is a configurable safety margin (e.g., 15%).

If a fault is detected, the system automatically dispatches an **Auditor Drone** from the reserve pool to verify the reading.

3 Frequently Asked Questions (FAQ)

3.1 Probabilty & Data Merging

Q: How do we merge probabilities from multiple drones?

A: We use a technique called **Sensor Fusion** (specifically, Inverse-Variance Weighting). Imagine Drone A reports a drought risk of 60% with high confidence (low noise), and Drone B reports 40% with low confidence (high noise). Instead of just averaging them to 50%, the system "trusts" Drone A more. The mathematical formula (see Section 2.2) ensures that the final combined probability leans closer to the more accurate sensor.

Q: What happens if the Model and the Sensor disagree?

A: The system trusts the drone's physical sensor reading *unless* the deviation is statistically impossible (see Fault Detection).

- **Small Disagreement**: The system fuses the model prediction and sensor reading mathematically to find a middle ground.
- **Large Disagreement**: The system flags the sensor as potentially faulty and dispatches a second drone (Auditor) to double-check.

3.2 Drone Behavior & Allocation

Q: How are drones assigned to areas?

A: Assignment is based on **Risk Prioritization**. 1. **High Risk Areas ($> 70\%$):** Get multiple drones (up to 3) to ensure frequent monitoring. 2. **Medium Risk Areas ($40-70\%$):** Get at least 1 drone. 3. **Low Risk Areas:** Monitored if resources allow.

We also keep a **Reserve Pool** (approx. 10% of the fleet) idle at the base. These reserves are only launched if a sensor failure is detected and an Auditor is needed immediately.

Q: Why do drones sometimes hover instead of moving?

A: This is usually due to the **Geofencing Safety Protocol**. If a drone drifts near the boundary of its assigned circular area, it enters a "Hold" or "Correction" state. It stops exploring and applies a velocity vector towards the center of the area to ensure it doesn't fly into restricted airspace or crash into a neighboring sector.

4 Implementation Analysis

4.1 Expected vs. Actual Implementation

The initial design specifications (outlined in `expected_work.md` and the reference paper *Drought-Cast*) proposed an end-to-end ML pipeline. Below is a comparison of the theoretical design versus the current simulated implementation.

Feature	Expected Design	Current Implementation
Drought Prediction	LSTM Neural Network trained on historical Kaggle meteorological data (SPI, SMI, VCI).	**Probabilistic Simulation**: Uses a stochastic model to generate realistic probability distributions (`drought_probability_model.py`) mimicking real-world variance.
Data Sources	Real-time satellite imagery (MODIS) and weather station APIs.	**Simulated Environs**: Synthetic data generation to allow testing without live internet dependencies.
Coverage Optimization	`scipy.optimize` for perfect non-overlapping circles.	**Greedy Allocation**: Dynamic assignment based on immediate risk priority and proximity (`area_allocation.py`).
Fault Detection	Statistical hypothesis testing (2σ deviation).	**Implemented exactly as designed**. Compares sensor readings against the model's baseline prediction.

Table 1: Design vs. Implementation Comparison

4.2 Key Code Implementations

The core logic for our **Sensor Fusion** and **Fault Detection** is critical to the system's reliability. Below are the specific functions driving these behaviors.

4.2.1 Sensor Fusion Logic

We use **Inverse-Variance Weighting** to merge conflicting probability estimates. This ensures that a drone with a noisy sensor (high variance) has less influence on the final decision than a drone with a precise sensor.

```python
def fuse_probabilities(prob_a, var_a, prob_b, var_b):
    """
    Merges two probability estimates using inverse-variance weighting.
    Formula: (w1*x1 + w2*x2) / (w1 + w2) where w = 1/variance
    """
    weight_a = 1.0 / var_a
    weight_b = 1.0 / var_b

    fused_prob = (weight_a * prob_a + weight_b * prob_b) / (weight_a + weight_b
)
    fused_var = 1.0 / (weight_a + weight_b)

```

```
    return fused_prob, fused_var
```

Listing 1: Inverse-Variance Fusion Logic

4.2.2 Fault Detection Threshold

The system flags a sensor as "Faulty" if its reading deviates from the expected model prediction by more than dynamic threshold (driven by the sensor's known noise profile).

```
def detect_fault(sensor_val, model_val, sensor_noise_std):
    # Threshold is 2 * standard_deviation (95% confidence interval)
    # plus a small base margin (0.15)
    dynamic_threshold = (2 * sensor_noise_std) + 0.15

    deviation = abs(sensor_val - model_val)

    if deviation > dynamic_threshold:
        return True  # FAULT DETECTED
    return False
```

Listing 2: Adaptive Fault Detection

5 Future Work & Limitations

5.1 Current Limitations

- **Simulated Weather:** The system currently does not ingest live weather API data; it relies on a sophisticated probability generator.
- **2D Movement:** Drones currently operate at a fixed altitude. True 3D terrain following is not yet implemented.
- **Simplified Physics:** Battery consumption is modeled as a linear drain, ignoring wind resistance and varied payload mass.

5.2 Future Roadmap

1. **Live LSTM Integration:** Replace the `drought_probability_model.py` with the pre-trained PyTorch LSTM model referenced in the *DroughtCast* paper.
2. **Swarm Communication:** Implement mesh networking so drones can share "Auditor" roles locally without routing every request through a central base station.
3. **Hardware Deployment:** Port the ROS nodes to physical PX4-based drones for real-world field testing.