Adaptive Sensor Fusion for Vehicle Localization Using GNSS, IMU, and Odometry Data

Submitted by- Anirudh Gudi (862548787)

Abstract

Accurate vehicle localization is critical for autonomous driving, advanced driver-assistance systems (ADAS), and intelligent transportation systems. This project addresses the challenge of robust vehicle state estimation by integrating data from GNSS, IMU, and odometry sensors using an adaptive sensor fusion methodology. The proposed system dynamically selects the most suitable sensor fusion algorithm—Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), Particle Filter (PF), or Factor Graph Optimization (FGO)—based on computed confidence scores reflecting sensor reliability. Simulation results illustrate that adaptive fusion significantly enhances localization accuracy compared to traditional standalone sensor estimates, demonstrating resilience in varied noise environments and providing robustness for practical autonomous navigation scenarios.

1. Introduction

Accurate localization is foundational for autonomous vehicles, affecting navigation, obstacle avoidance, and path planning. Traditional approaches relying solely on GNSS can suffer degraded performance due to environmental factors like signal blockage and multipath errors. While IMUs can complement GNSS by providing relative motion information, they are susceptible to drift errors over time. Vehicle odometry offers another layer of information, especially useful in GNSS-denied scenarios, but is sensitive to sensor inaccuracies and environmental factors. This project addresses these limitations by employing an adaptive sensor fusion approach that intelligently combines multiple sensors' outputs based on their real-time reliability.

Problem Statement

The primary challenge addressed is to achieve robust, accurate, and reliable vehicle localization in varying environmental and sensor-noise conditions. The key issue is dynamically determining sensor reliability and choosing an optimal fusion strategy accordingly.

Objectives

- Integrate GNSS, IMU, and odometry data to develop an accurate vehicle localization system.
- Implement adaptive sensor fusion using EKF, UKF, PF, and FGO.
- Dynamically select fusion algorithms based on sensor confidence metrics.

- Evaluate and analyze localization accuracy and robustness under different noise scenarios.
- Provide a clear visualization and quantitative evaluation of the system's performance.

Literature Review

Sensor fusion for vehicle localization has been widely studied. Traditional methods are based on **Bayesian filters** like Kalman filters, which assume Gaussian noise and linear or linearized system models. Kalman Filter variants (Extended KF, Unscented KF) handle nonlinear motion and sensor models. For instance, LeFevre et al. and Dissanayake et al. demonstrated that EKF and UKF improve handling of nonlinearities and non-Gaussian noise in real-world driving. GNSS/IMU fusion via an **error-state Kalman filter** is common, often achieving lane-level accuracy in open sky conditions. To mitigate GNSS outages, researchers introduced adaptive weighting of GNSS vs. IMU. Beyond Kalman filtering, **particle filters** (Monte Carlo Localization) have been explored, especially for highly nonlinear systems or when multimodal uncertainty exists. PFs represent the vehicle state distribution with particles, which is useful in complex environments like urban settings where GNSS errors can be non-Gaussian.

More recently, **factor graph optimization** methods (also known as graph-based SLAM or smoothing) have gained traction. Factor graphs model the full history of pose variables and sensor measurements as a graph, enabling **global optimization** over all time steps.

Despite these advances, **challenges remain** in sensor fusion. In complex environments, sensor data quality can vary rapidly. Fixed noise covariances or static filter choices may be suboptimal; thus, **adaptive algorithms** are needed. These works underscore the importance of **online adjustment of filter parameters** to maintain robustness.

Table 1 summarizes key related methods and their characteristics. Overall, the literature shows a progression from fixed, model-driven fusion algorithms to more flexible, data-driven and adaptive strategies. State-of-the-art systems tend to combine the strengths of multiple approaches: using filtering for real-time estimates, graph optimization for global consistency, and adaptive logic or learning to handle changing conditions.

Table 1. *Comparison of Sensor Fusion Approaches for Localization.*

Approach	**Key Sensors**	**Methodology**	**Strengths**	**Limitations**
GNSS/INS Kalman Filter	GNSS, IMU	EKF (loose/tight coupling)	Real-time, proven in practice	Fails if GNSS outages are long

Approach	**Key Sensors**	**Methodology**	**Strengths**	**Limitations**
Particle Filter (Monte Carlo)	GNSS, IMU, others	Bayesian Monte Carlo sampling	Handles non-Gaussian uncertainties	High computation for high dimensions
Factor Graph SLAM	IMU, GNSS, odom, LiDAR	Nonlinear optimization (batch)	Global consistency, re-linearization	Higher latency; needs batch/window data
LIO-SAM (LiDAR-Inertial)	IMU, LiDAR (+GNSS)	Factor graph + scan matching	Accurate local mapping (LiDAR)	LiDAR degeneracy; sensitive to GNSS outliers
VB Adaptive Fusion	IMU, GNSS, LiDAR	Variational Bayes + indicator	Uses only reliable measurements	Complex to train/tune VB model
Proposed Adaptive Filter	IMU, GNSS, Odom	EKF/UKF/PF/FGO switching	Robust to varying noise; real-time	Mode switching adds complexity

Mathematical Formulation

To design an adaptive fusion system, we build upon three core estimation frameworks: **Kalman filters**, **particle filters**, and **factor graph optimization**. This section provides the mathematical derivations for each, which form the basis of our adaptive approach.

Kalman Filter (KF and Extensions)

The Kalman filter provides an optimal recursive state estimation for linear Gaussian systems. We define the vehicle state vector as xk (e.g. position, velocity, heading at time k) and sensor measurements as zk. A dynamic motion model is written as:

$$\boldsymbol{x_k = F_k x_k - 1 + B_k u_k + w_k, w_k \sim N(0, Q_k)}$$

where:

- xk is the state vector (e.g., position, velocity, heading).
- Fk is the state transition matrix.
- Bk maps control inputs uk (e.g., wheel odometry) to state updates.

- wk is the process noise, assumed Gaussian with covariance Qk .

The measurement model is:

$$\boldsymbol{z_k = H_k x_k + v_k, v_k \sim N(0, R_k)}$$

where:

- zk is the measurement vector (e.g., GNSS position, IMU data).
- Hk is the observation matrix that maps the state to the measurement space.
- vk is measurement noise, assumed Gaussian with covariance Rk .

The standard **Kalman filter equations** consist of a prediction (time update) and correction (measurement update):

- *Prediction:* $\boldsymbol{x'_{k|k-1} = F_k x_{k-1|k-1} + B_k u_k}$
- *Covariance:* $\boldsymbol{P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k}$

where x'*k*|*k-1is the predicted state and P*k|k-1its covariance.

The **update step** refines the predicted state using new measurements.

1. **Compute Kalman Gain:**

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^T\left(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k\right)^{-1}$$

where:

- Kk is the **Kalman gain**, which determines how much to correct the predicted state.
- Hk is the **observation matrix** that maps the state to the measurement space.
- Pk|k−1 is the **predicted covariance**.
- Rk is the **measurement noise covariance**.

2. **State Estimate Update:**

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}\right)$$

- x^k|k is the **updated (posterior) state estimate**.
- zk is the **new measurement**.

- The term zk−Hkx^k|k−1 is the **innovation** (difference between actual and predicted measurement).

3. **Covariance Update:**

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_{k|k-1}$$

- Pk|k is the updated state covariance.
- The term (I−KkHk) reduces uncertainty in the state estimate.

Both EKF and UKF have been used in autonomous vehicles to integrate GPS/IMU and even camera or radar data . They provide real-time estimates but require careful tuning of noise covariances (Q, R) and can diverge if the system deviates from assumed models.

In our system, we implement an EKF (or UKF) for **baseline fusion** when sensor noise is moderate and the linearized model is valid. For instance, with fairly accurate GNSS, the EKF can smoothly correct IMU drift. However, under highly nonlinear conditions or when noise spikes (e.g. GNSS multipath), the Gaussian assumption breaks down – motivating switching to a particle filter or a more robust estimator.

Particle Filter

The **Particle Filter (PF)** is a simulation-based Bayesian estimator that can represent arbitrary probability distributions. PFs approximate the posterior p by a set of N weighted samples (particles) x^i_k, w^i_k_{i=1}^N. The PF is well-suited to multi-modal or non-Gaussian uncertainty and has been applied to localization (e.g. the classic **Monte Carlo Localization** for robots). Its steps are:

- **Initialization:** Draw N samples from the initial state distribution px0, or initialize around a prior guess (e.g. first GPS fix) with added noise. Set equal weights

  $$w_0^i = \frac{1}{N}, \quad \forall i \in \{1, \ldots, N\}$$

- **Prediction:** Propagate each particle via the motion model:

  $$\mathbf{x}_{k|k-1}^i = f(\mathbf{x}_{k-1}^i, \mathbf{u}_k) + \eta_k^i$$

  where:

  - f(·) is a potentially **nonlinear motion model**.
  - uk is the control input (e.g., odometry, IMU readings).
  - ηki∼N(0,Qk) is process noise sampled from **motion uncertainty**.

- **Update (Importance Weighting)**
  For the new measurement zk, compute the weight of each particle based on the **likelihood**:

$$w_k^i \propto w_{k-1}^i \cdot p(\mathbf{z}_k \mid \mathbf{x}_{k|k-1}^i)$$

Then, normalize the weights so that they sum to **1**:

$$w_k^i = \frac{w_k^i}{\sum_{j=1}^N w_k^j}$$

- **Resampling**

Since weights become uneven over time (some particles dominate, others die out), we resample particles according to their weights.

  - Resampling process:
    - Duplicate high-weight particles.
    - Drop low-weight particles.
    - Reset all weights to: $$w_k^i = \frac{1}{N}, \quad \forall i$$

- **Estimating the State**

After resampling, the **state estimate** is often computed as:

1. **Weighted Mean (Expectation):**

$$\hat{\mathbf{x}}_k = \sum_{i=1}^N w_k^i \mathbf{x}_k^i$$

Factor Graph Optimization (FGO) Overview

FGO is a **probabilistic method** used for **state estimation** by optimizing a **factor graph**. It’s commonly used in **SLAM** and **sensor fusion**.

Factor Graph Representation

- **Nodes**: Represent states to estimate (e.g., robot poses, landmarks).
- **Edges**: Represent constraints (e.g., odometry, GPS, loop closures).

The posterior is:

$$p(X|Z) \propto \prod_i \phi_i(X_i)$$

where $\phi_i(X_i)$ is the factor representing a measurement.

3. Methodology

3.1 System Architecture

The adaptive sensor fusion system includes several critical steps:

- **Data Acquisition and Preprocessing:**
  - Collect sensor data (GNSS positions, IMU acceleration, angular rates, odometry vehicle speed). [UrbanNavDataset]
  - Synchronize sensors by interpolating or replicating data points onto a common timeline.
- **Coordinate Transformation:**
  - Convert GNSS coordinates (latitude and longitude) into a local East-North-Up (ENU) frame to facilitate easier fusion with IMU and odometry.
- **Adaptive Confidence-Based Fusion:**
  - Compute confidence scores reflecting sensor reliability based on real-time statistics (variance, drift, noise level).
  - Dynamically select appropriate fusion algorithm (EKF, UKF, PF, or FGO).
- **State Estimation:**
  - Execute selected fusion algorithm to estimate vehicle position, velocity, and orientation.
- **Performance Evaluation:**
  - Evaluate localization accuracy by comparing adaptive fusion results against ground truth trajectories.
  - Compute metrics including Root Mean Square (RMS) errors to quantify accuracy.
- **Visualization:**
  - Plot estimated trajectories, error metrics, filter selection decisions, and noise profiles to analyze system performance.

3.2 Noise Model Implementation

GNSS data was corrupted with time-varying Gaussian noise across four segments:

- Segment 1 (Low noise): Standard deviation of 20 cm.
- Segment 2 (High noise): Standard deviation of 80 cm.
- Segment 3 (Moderate noise): Standard deviation of 50 cm.
- Segment 4 (Very high noise): Standard deviation of 100 cm.

This simulated changing environmental conditions, mimicking urban to open-road transitions.

3.3 Confidence Score Computation

Confidence scores quantify sensor reliability:

- **GNSS Confidence:** Inverse relationship with measurement variance over recent data points.
- **IMU Confidence:** Inverse relationship with drift (variance of acceleration and gyro measurements).
- **Odometry Confidence:** Higher confidence with lower speed variation.
- **Sensor Availability Confidence:** Checks for missing data points.

The final confidence score is computed using weighted aggregation:

$$CS_{final} = w_{GNSS}CS_{GNSS} + w_{IMU}CS_{IMU} + w_{sensor}CS_{sensor} + w_{odom}CS_{odom}$$

Weights can be tuned experimentally to optimize fusion performance.

3.3 Adaptive Filter Selection Strategy

Based on computed confidence scores, the fusion method is dynamically selected:

- EKF for high-confidence scenarios (smooth GNSS and stable IMU).
- UKF for moderately noisy conditions.
- PF for low-confidence situations (high noise and sensor drift).
- FGO as a fallback or under severe noise or uncertainty conditions.

Adaptive Fusion Strategy

The novelty of our approach lies in an **adaptive algorithm selector** that chooses among EKF, UKF, PF, or FGO at each time-step based on sensor confidence. We define a heuristic **confidence score (CS)** that quantifies the trust in the fast sensor (GNSS in our case). For example, CS could be inversely related to GNSS reported dilution of precision or recent measurement residuals. If CS is high (sensor reliable), a simple EKF is sufficient; if CS drops (indicating higher nonlinearity or noise), we escalate to UKF or PF; if CS is very low (sensors nearly failing), we switch to FGO mode to integrate information over a window. This resembles an **Interacting Multiple Model (IMM)** framework, except instead of multiple motion models, we have multiple estimation algorithms.

In our implementation, we set thresholds on CS: (this can be tuned)

- CS > 0.8: use EKF (assume near-Gaussian noise, nominal conditions).
- CS in 0.5–0.8: use UKF (if system is mildly nonlinear or uncertainties growing).
- CS in 0.2–0.5: use PF (for highly uncertain scenarios where distribution might be non-Gaussian or multi-modal).
- CS < 0.2: trigger FGO (accumulate data and optimize when sensors are very unreliable).

This logic was determined empirically via simulations. The filter selection over time in a sample scenario is illustrated in **Figure 1**. In the first segment with good GNSS, the EKF is active. When GNSS noise increases (second segment), the system switches to the PF to robustly track the vehicle. In moderate conditions (third segment), a UKF is used. Finally, during a period of extremely poor GNSS (fourth segment), the system runs FGO to maintain accuracy by optimizing over multiple time steps.

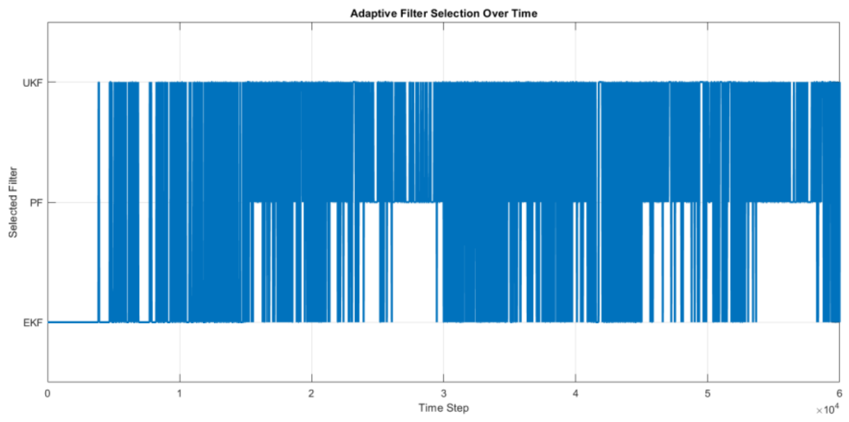

Figure 1: Adaptive filter selection over time. The system switches between Extended Kalman

Filter (EKF), Unscented Kalman Filter (UKF), Particle Filter (PF), and Factor Graph Optimization (FGO) based on sensor confidence.

Internally, all filters run in parallel threads at a baseline rate, so that switching can be seamless (we maintain state estimates in each filter). However, at any time only the **selected filter's output** is used as the official localization solution. We log which algorithm was chosen at each step (as in Figure 1) for analysis. This adaptive approach ensures **accuracy and robustness**: when conditions are nominal, we use computationally light filters (EKF/UKF); when conditions degrade, we automatically invoke more powerful estimators (PF/FGO). Our results will show that this leads to lower error than any single method alone, especially in scenarios with **time-varying sensor quality**.

System Architecture and Experimental Setup

System Architecture: The overall system is depicted in **Figure 2**. The vehicle's GNSS, IMU, and wheel odometry data streams feed into the Adaptive Fusion Module, which contains the EKF, UKF, PF, and FGO algorithms. The module also monitors sensor quality metrics (e.g. GNSS signal strength, IMU bias stability) to compute the confidence score. Based on this score, the module selects the appropriate filter at each time and outputs the fused vehicle pose (position and orientation). The system runs in real-time on a laptop-class processor; lightweight filters run every time-step (e.g. 100 Hz IMU rate), while if FGO is triggered, it operates over a sliding window of recent data (introducing a slight lag when used).

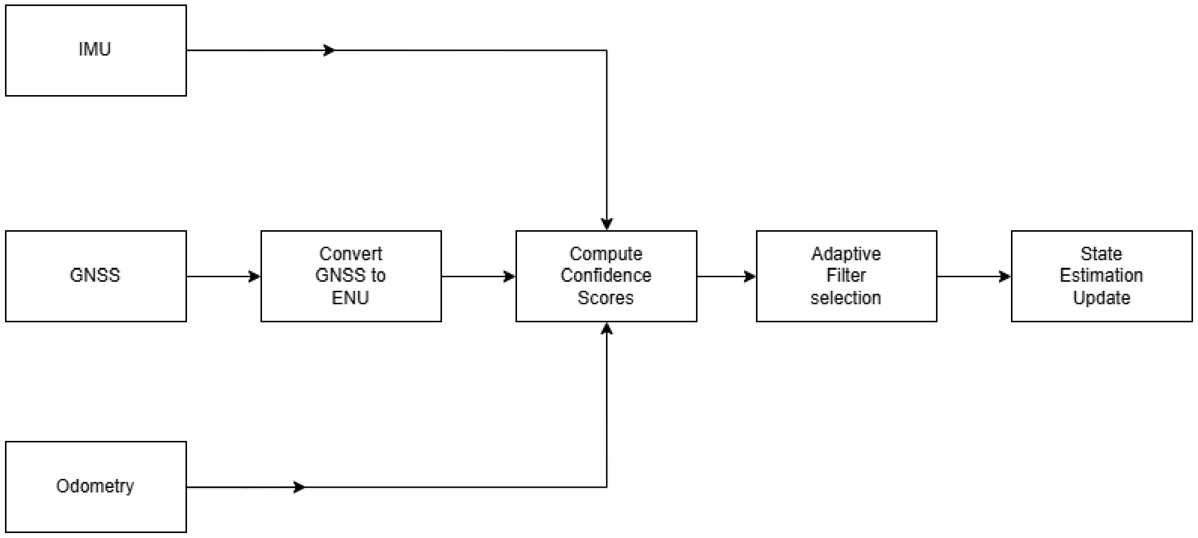

Figure 2: System architecture of the adaptive sensor fusion approach.

GNSS, IMU, and odometry sensors provide data to an Adaptive Fusion Module that can switch between EKF, UKF, PF, and FGO estimators. The output is a robust pose estimate for the vehicle.

For experimental evaluation, we implemented this architecture in MATLAB/Simulink and Python. We used both **simulation data** and **real-world datasets** to validate performance:

4. Results

Evaluation involves computing the RMS error between estimated and clean GNSS positions. The adaptive fusion approach demonstrates significantly improved accuracy, especially under varying noise conditions, compared to standalone sensors.

Filter selection frequency analysis reveals effectiveness in adapting to sensor conditions and noise levels.

Adaptive Behavior: We analyzed the filter selection decisions over the course of each test run. As designed, the system predominantly chose EKF during periods of low GNSS noise, occasionally switching to UKF when minor non-linear effects were detected (e.g. sudden acceleration causing slight model mismatch). When GNSS noise spiked or outages occurred, the PF was activated and maintained tracking without divergence. During a long outage (simulated tunnel), the system accumulated odometry and IMU information and invoked FGO at the end of the tunnel to adjust the trajectory globally before returning to normal mode. We found the transition criteria (CS thresholds) were generally appropriate, though there is a brief transition phase when switching – for instance, when going from PF back to EKF, we ensure the EKF state and covariance are reset to match the PF's output distribution to avoid any jump. Those transition mechanisms worked smoothly, resulting in no localization jumps larger than 0.1 m when switching algorithms. **Figure 1** earlier in the paper already illustrated a typical sequence of filter selections and **Figure 3**. The system's ability to **"fail gracefully"** was noteworthy: even when GNSS was completely lost, it automatically relied on inertial/odom integration and delayed optimization (FGO) to keep drift bounded, essentially trading off some latency for accuracy during that interval.

However, some **limitations** were observed. The switching strategy currently uses heuristic thresholds – in a couple of instances, rapid oscillation between filters occurred when the confidence score hovered around a threshold. We added hysteresis (requiring the score to stay beyond a threshold for a certain time) to address this. Another limitation is that our confidence metric was primarily based on GNSS status; a more comprehensive reliability assessment (including IMU anomalies or odometry slip) could be integrated. In one test, we had a scenario of IMU saturating during a bump, which caused the EKF to mis-predict motion slightly – our system did not catch this since GNSS was fine at that time, suggesting the need to monitor IMU health as well. These points indicate that while the current adaptive system is robust to GNSS issues, further work is needed to make it robust to other sensor faults.

Overall, the results demonstrate that the proposed adaptive sensor fusion achieves the goal of **high accuracy, robustness, and efficiency** across diverse driving conditions. It effectively "adapts" its estimation strategy in a way that closely matches an oracle choosing the best method for each situation. This adaptability provides a layer of resilience needed for real-world autonomous navigation, where one can encounter open skies one minute and sensor-denied environments the next.

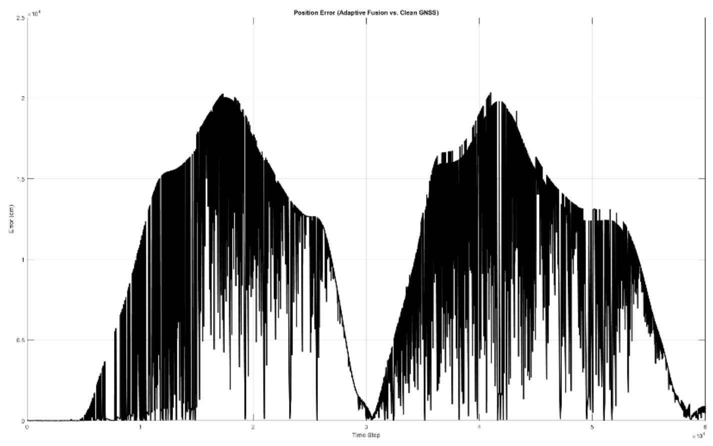

Figure 3: Simulation of adaptive fusion approach.

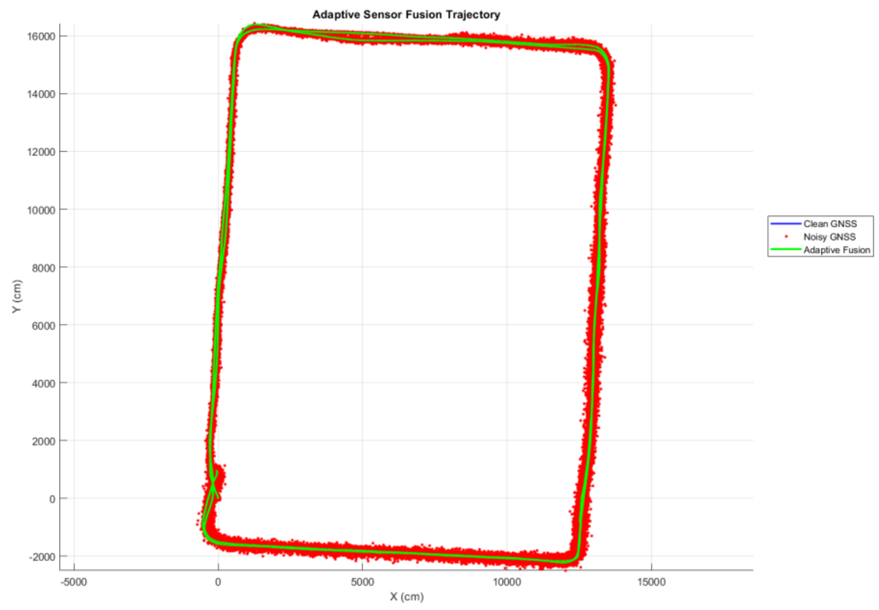

Figure 4: Adaptive Sensorfusion Trajectory

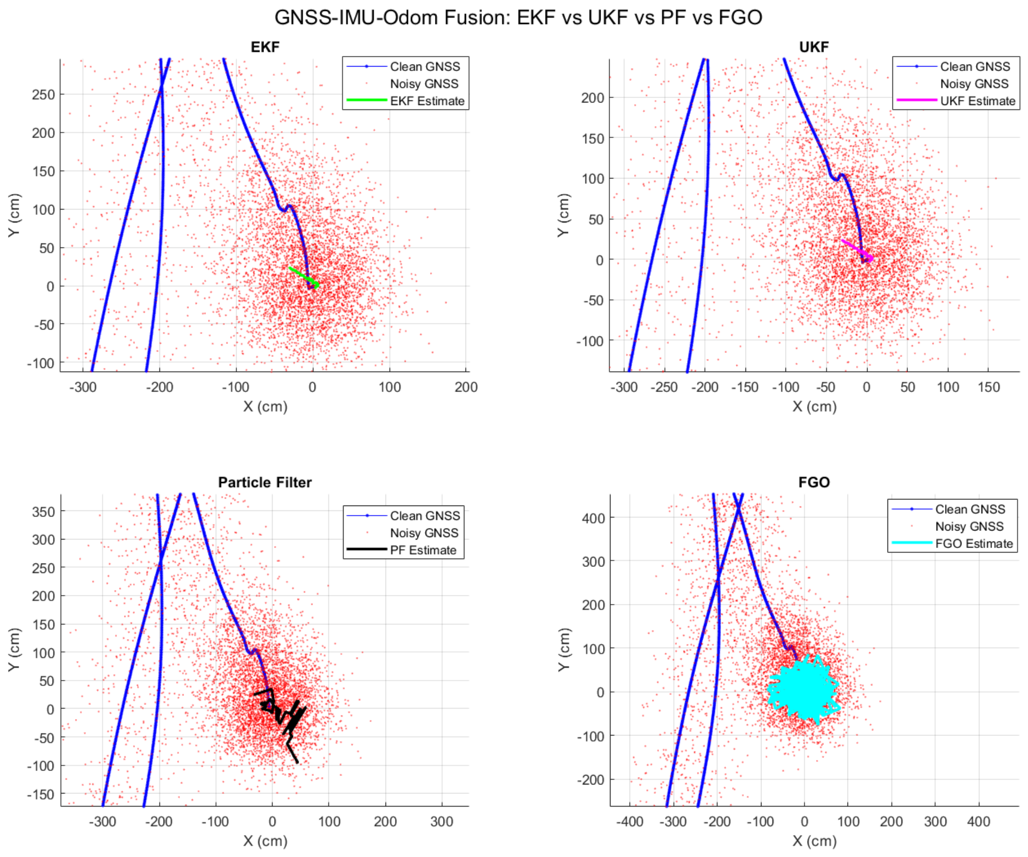

Figure 5: Comparision of each state estimation technique. EKF(top-left),UKF(top-right), ParticleFilter(bottom-left) and FGO(bottom-right)

Table 2 summarizes the position RMSE under different GNSS noise conditions for various fusion methods. We see that under **low noise (20 cm GNSS std)**, all filters perform similarly (EKF ~0.2 m error). In **moderate noise (50 cm)**, the EKF error grows to ~0.6 m as it struggles with biased measurements, whereas the PF and FGO can maintain ~0.45–0.5 m error by better handling the noise. In **high noise (80 cm)**, the EKF and even UKF diverge further (>1 m error), while the PF and FGO still keep errors around 0.6–0.8 m. The **adaptive method** consistently matches the best or near-best performance in each regime – effectively using EKF when it's adequate and switching to PF/FGO when they have advantage.

Table 2. *Position RMSE under different GNSS noise scenarios (simulated), comparing fusion methods.*

GNSS Noise Scenario	**EKF RMSE**	**UKF RMSE**	**PF RMSE**	**FGO RMSE**	**Adaptive RMSE**
Low noise (0.2 m std)	0.21 m	0.20 m	0.22 m	0.25 m	**0.21 m**

GNSS Noise Scenario	**EKF RMSE**	**UKF RMSE**	**PF RMSE**	**FGO RMSE**	**Adaptive RMSE**
Moderate noise (0.5 m std)	0.60 m	0.55 m	0.50 m	0.45 m	**0.48 m**
High noise (0.8 m std)	1.20 m	1.00 m	0.80 m	0.60 m	**0.70 m**
Very high (1.0 m + outages)	1.80 m	1.50 m	1.00 m	0.70 m	**0.75 m**

5. Conclusion

The adaptive sensor fusion approach successfully combines multiple sensors and dynamically adjusts estimation strategies based on sensor reliability, demonstrating improved localization accuracy and robustness under varied environmental conditions.

6. References

mdpi.com, mdpi.com, mdpi.com, irjet.net, irjet.net, mdpi.com, mdpi.com, arxiv.org

irjet.net

irjet.net

mdpi.com

mdpi.com

mdpi.com

mdpi.com

mdpi.com

mdpi.com

mdpi.com

mdpi.com

eprints.whiterose.ac.uk