# Advanced Sensor Data Fusion in Distributed Systems

Florian Spieß

July 24, 2021

## Implementation

The source code includes the implementations for all the methods shown. All the files are relative paths from the project directory.

- Kalman Filter (kalman.py)
- Information Filter (information.py)
- Linear Sensor and Node (sensor.py)
- Fusion Center (fusion.py)
- Central Kalman Filter (assigment_01/central.py)
- Naive Fusion (assigment_01/naive.py)
- Tracklet Fusion with Equivalent Measurements (assigment_02/tracklet.py)
- Federated Kalman Filter with Relaxed Evolution Model (assigment_02/federated.py)
- Distributed Kalman Filter with Globalization Step (assigment_02/distributed.py)

The simulation is implemented in main.py and can be configured through the parameters provided in lines 62-66.

- $T$ the episode length
- *stepsize* the delay between each measurement
- *sigma* covariance factors for each sensor used for noise term
- *track* the ground-truth trajectory
- *velocity* the ground-truth velocity

# Analysis for the Simulation Results

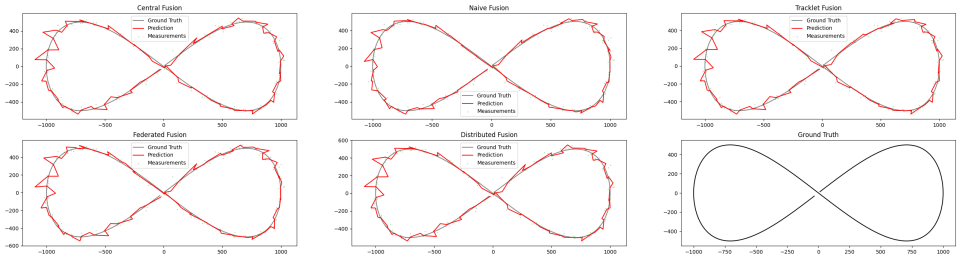


Figure 1: Trajectory and predictions with 10 second interval measurements

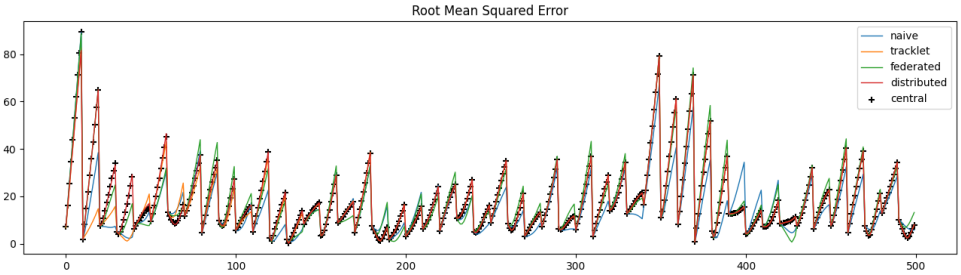


Figure 2: Root Mean Squared Error

The trajectories shown in Figure 1 were produced with identical measurements for each fusion method. In this scenario, the sensor nodes produced a measurement every 10 seconds and predicted the position within 1 second timesteps.

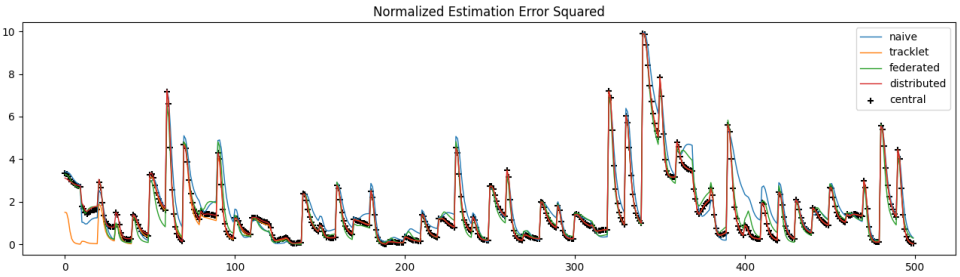


Figure 3: Normalized Estimation Error Squared for each fusion method

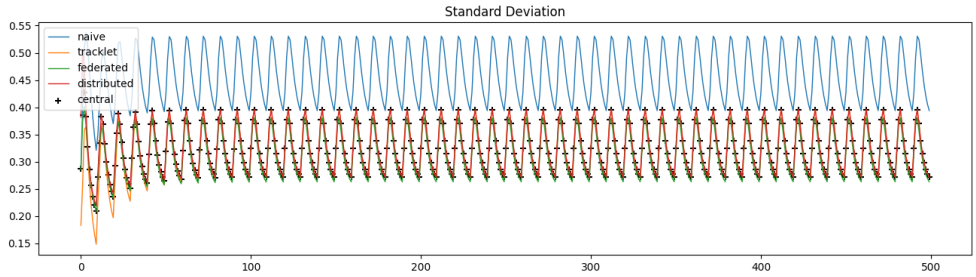


Figure 4: The Standard Deviation for 10 second measurement intervals

Since the naive fusion and federated Kalman Filter are not optimal solutions, they sometimes produce different fusion results compared to the optimal solutions, as shown in Figure 2. However, these errors seem to have nearly the same confidence as compared to other fusion methods, which can be seen in the Normalized Estimation Error Squared (NEES) in Figure 3. Which is given by

$$NEES(k) = (x_k - x_{k|k})^T P_{k|k}^{-1} (x_k - x_{k|k})$$

This error can be interpreted as distance to the ground truth, scaled by the confidence of the prediction. The more confident errors are also penalized more by this metric. Both the distributed Kalman Filter, which also makes use of the relaxed evolution model, and the tracklet fusion are producing optimal predictions, as expected. Optimal, in this case, refers to the central Kalman Filter which does fusion on all raw sensor measurements rather than posterior tracks.

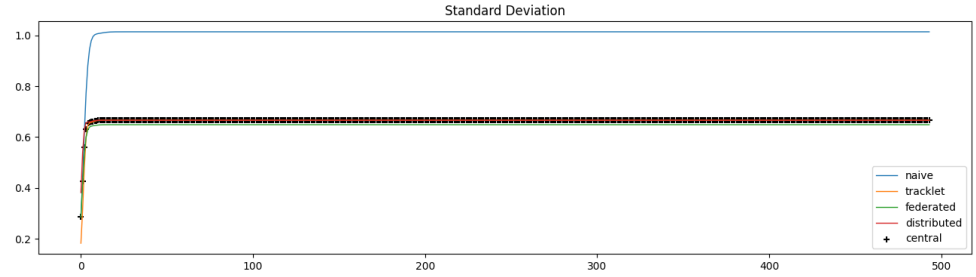


Figure 5: The Standard Deviation for 1 second measurement intervals

Figure 4 shows the standard deviation of the simulation. All methods, except for the naive fusion, produce identical standard deviations. Figure 5 also shows that these deviations properly converge if measurements are available for every timestep.

In summary, the optimal fusion methods are Tracklet Fusion, Central Fusion, and the distributed Kalman Filter. All of these methods produce identical predictions and standard deviations. However, both central fusion and tracklet fusion require full communication at each timestep to produce these results. The distributed Kalman Filter can function with arbitrary communication since it is aware of the other sensor nodes in the network and locally adjusts the covariance.
Central Fusion receives the raw sensor measurements and knows the measurement covariance. These measurements are then filtered using a central Kalman Filter. Tracklet Fusion makes use of the equivalent measurements for the posterior tracks of each sensor nodes. To produce these fisher information parameters, it must know the dynamics model for the measurements to apply it to the posterior tracks to produce priors in the fusion center.