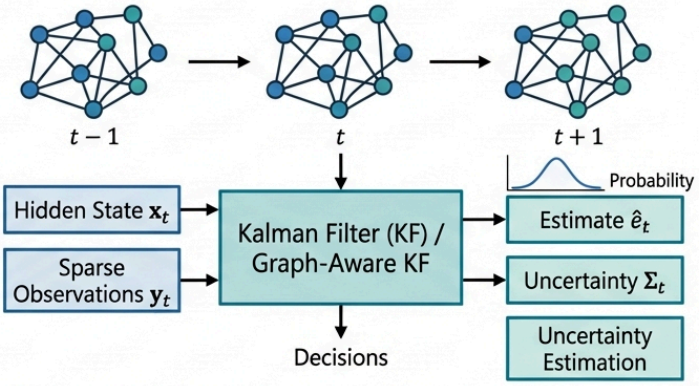
Slide 1 — State Estimation for Dynamical Networks

State Estimation for Dynamical Networks

- **Goal:** infer hidden network state $\mathbf{x}_t$ from sparse noisy observations $\mathbf{y}_t$
- **Why:** traffic, epidemics, power grids, social diffusion, dynamic graphs
- **Output:** estimate + uncertainty (variance/covariance) for decisions
- **We cover:** KF → distributed KF → graph-aware KF → learning + calibration

Takeaway
Accurate state estimation on dynamic networks requires leveraging both temporal dynamics and graph structure to handle sparse data and provide reliable uncertainty estimates.



The transpose, ℓ_2 norm, and gradient are denoted by $\{\cdot\}^T$, $\|\cdot\|_2$, and $\nabla\{\cdot\}$. We let $G = (V, E, W)$ be a weighted, undirected, and connected graph, where $V = \{1, \ldots, N\}$ denotes the set of nodes, and $\Delta = |V|$ is the number of nodes. If there is an edge connecting

Source: Tracking Graph Signals via Neural Aided Kalman Filtering

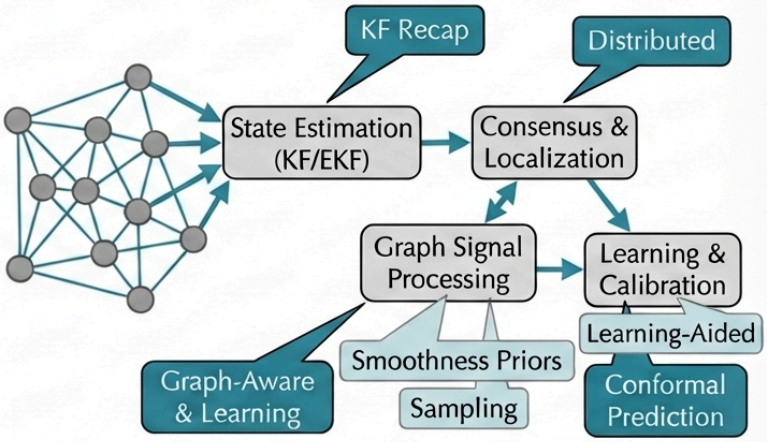
$G_t(V, \xi_t)$, where t is the time index, $V = \{1, \ldots, N\}$ denotes the set of nodes, and $\xi_t \subset V \times V$ is the set of time-varying edges.

Source: A Kalman Filter for Tracking Network Dynamics

Slide 2 — Roadmap

Roadmap

- Part I: **setup + KF recap** (linear/nonlinear)
- Part II: **distributed + scalable** (consensus, localization)
- Part III: **graph-aware filtering** (smoothness priors, sampling)
- Part IV: **learning-aided + conformal calibration**

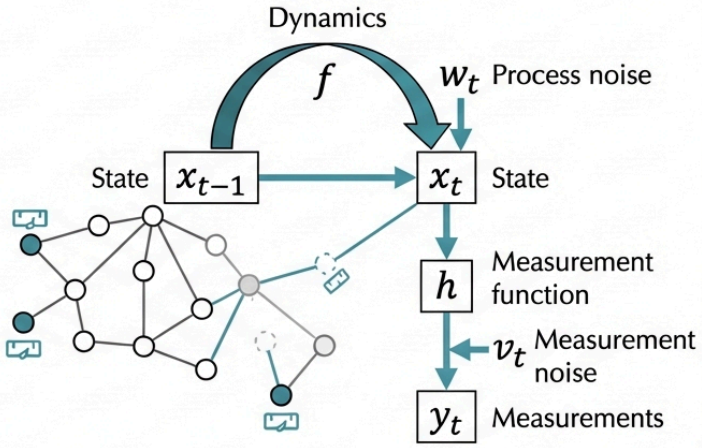


Takeaway
This roadmap advances from fundamental Kalman filtering to distributed, graph-aware, and learning-enhanced state estimation methods.

Slide 3 — State-space model on a network

State-space model on a network

- **State** on nodes/edges: x_t (e.g., speed per road segment)
- **Dynamics:** $x_t = f(x_{t-1}) + w_t$ (process noise)
- **Measurements:** $y_t = h(x_t) + v_t$ (measurement noise)
- **Key challenge:** sparse sensors + evolving structure



Takeaway: $\{\cdot\}^{\mathrm{T}}$, $_{ii}$, $\|_2$ norm, and gradient are denoted by $\{\cdot\}^{\mathrm{T}}$, $\Delta =$ $\Delta = |V|$ is the number of nodes. If there ... the edge; otherwise, $[W]_{i,j} = 0$. In general, graph signals are

$G_l(V, \xi_l)$, where l is the time index, $V = \{1, \ldots, N\}$ denotes

Source: GSP-KalmanNet: Tracking Graph Signals via Neural-Aided Kalman Filtering (2022)

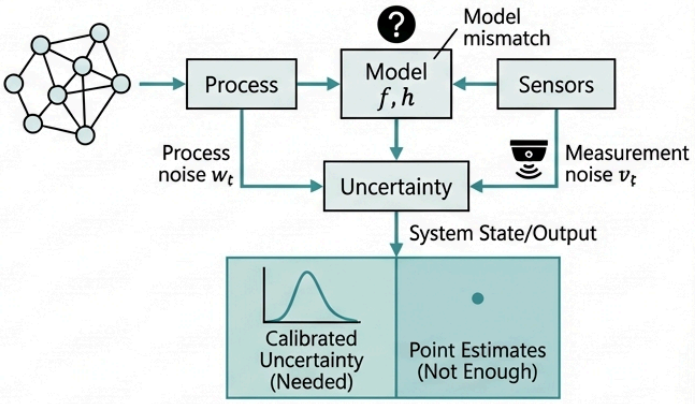
Source: Kalman Filter for Tracking Network Dynamic (2020)

Slide 4 — Sources of uncertainty in estimation

Sources of Uncertainty in Estimation

- Process noise w_t: unmodeled inputs, stochastic interactions
- Measurement noise v_t: sensor errors, missingness
- Model mismatch: wrong f, h or wrong Q, R; drifting topology
- We need calibrated uncertainty, not just point estimates

Takeaway: Uncertainty from multiple sources must be modeled for robust learning and estimation.



Mathematical Notation

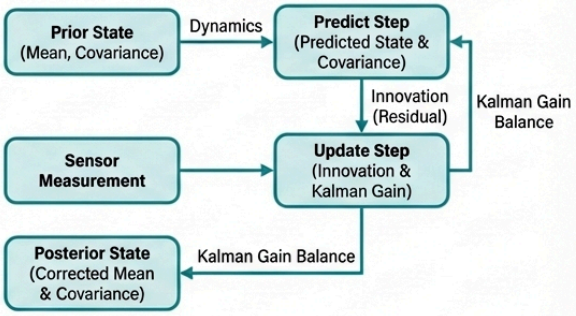
The transpose, ℓ_2 norm, and gradient are denoted by $\{\cdot\}^\top$, $\|\cdot\|_2$, and ∇, respectively.
Source: GSP-KalmanNet_Tracking_Graph_Signals_via_Neural-Aided_Kalman_Filtering.pdf

$\Delta = |V|$ is the number of nodes. If there is an edge, $[W]_{i,j} > 0$; otherwise, $[W]_{i,j} = 0$. In general, graph signals are defined on $G_l(V, \xi_l)$, where l is the time index, $V = \{1, \ldots, N\}$ denotes the set of nodes.
Source: Kalman_Filter_for_Tracking_Network_Dynamic.pdf

Slide 5 — Kalman Filter recap: Predict → Update

Kalman Filter Recap: Predict → Update

- Predict: propagate mean + covariance through dynamics
- Update: innovation (residual) corrects prediction
- Kalman gain balances trust: model vs sensor
- Posterior is Gaussian when linear + Gaussian assumptions hold



Takeaway
The Kalman Filter optimally fuses noisy measurements with dynamic models by iteratively updating its state estimate based on uncertainty, under linear Gaussian conditions.

Source: Kalman Filter for Tracking Network Dynamics (Title)

$G_l(V, \xi_l)$, where l is the time index, $V = \{1, \ldots, N\}$ denotes

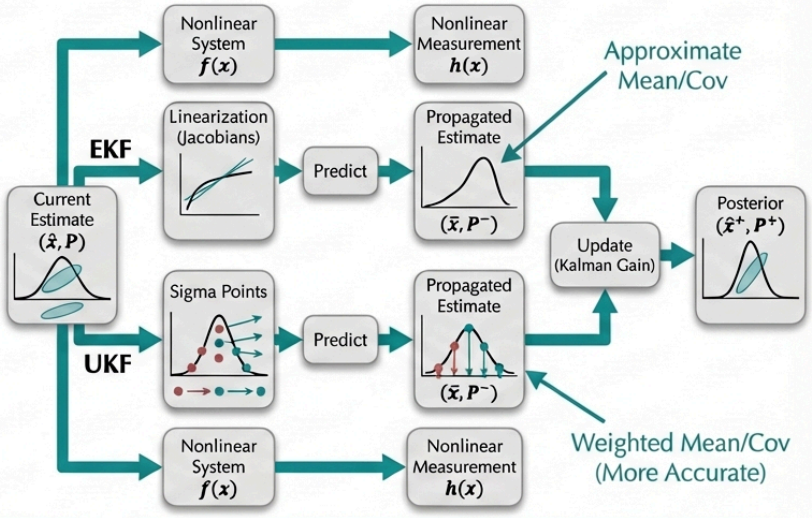
The transpose, ℓ_2 norm, and gradient are denoted by $\{\cdot\}^\top$. $|V|$ is the number of nodes. If there

Source: GSP-KalmanNet_Tracking_Graph_Signals via Neural-Aided Kalman Filtering (2022)

Slide 6 — EKF vs UKF: Nonlinear intuition

EKF vs UKF: Nonlinear Intuition

- **EKF**: linearize f,h (Jacobians) around current estimate
- **UKF**: sigma points propagate uncertainty through nonlinearity
- **Tradeoff**: EKF cheaper; UKF often more robust to nonlinearity
- **Both**: keep the predict/update backbone



Notation: $\{\cdot\}^\top$ for transpose, $\|\cdot\|_2$ for ℓ_2 norm, ∇ for gradient. $|V|$ is the number of nodes.

Source: GSP-KalmanNet: Tracking Graph Signals via Neural-Aided Kalman Filtering (2023)

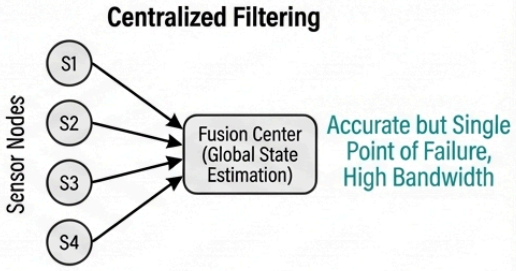
Graph Signals & Weights: $[W]_{i,j} = 0$ if no edge. Graph $\mathcal{G}_l(V, \xi_l)$ at time l, $V = \{1, \ldots, N\}$.

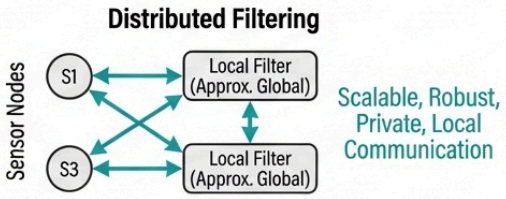
Source: Kalman Filter for Tracking Network Dynamic & GSP-KalmanNet: Tracking Graph Signals (2023)

Slide 7 — Centralized vs Distributed filtering

Centralized vs. Distributed Filtering

- **Centralized:** one fusion center (accurate but not scalable)
- **Distributed:** local filters + neighbor communication
- **Motivation:** scale, robustness, privacy, bandwidth limits
- **Core idea:** approximate global posterior via consensus





Takeaway
Distributed methods enable scalable and robust filtering by exchanging local information to approximate global consensus.

A sensor network with $n = 200$ nodes and $l = 1074$ links.

$$\mathbb{E}[v_k v_l^T] = R_k \delta_{kl}$$

Given the measurements $Z_k = \{z_0, z_1, \ldots, z_k\}$, the state

Source: Distributed Kalman Filter with Embedded Consensus Filters (2008)

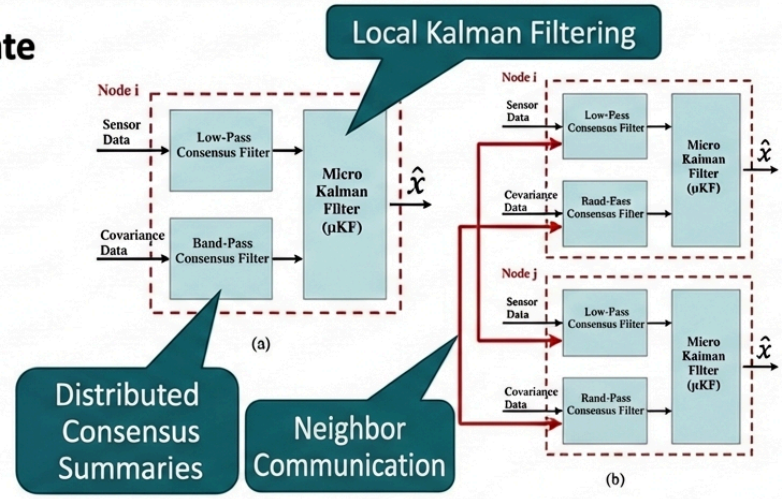
$$r_{i,j,k+1} = r_{i,j,k} + \Delta t$$

Source: Localized Extended Kalman Filter for Scalable Real-Time Traffic State Estimation (2016)

Slide 8 — Distributed Kalman filter with consensus

Distributed Kalman filter with consensus (architecture)

- Each node runs **local predict/update** using local measurements
- **Consensus exchanges summaries** (information form) with neighbors
- Goal: **emulate centralized KF** without a fusion center
- More **consensus rounds** → closer to centralized accuracy



Takeaway

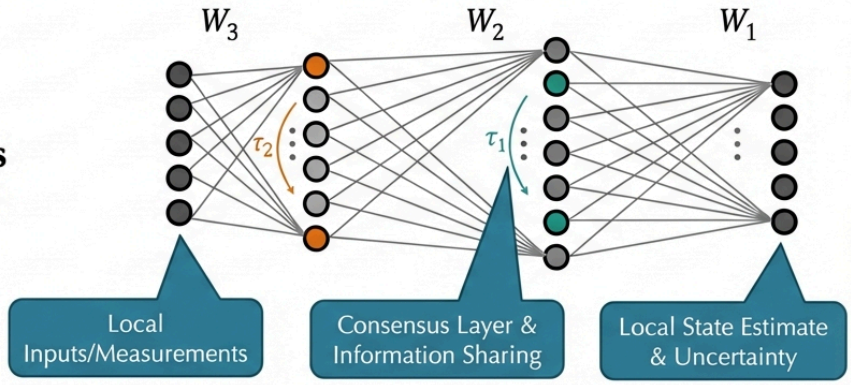
Distributed consensus allows a network of nodes to collaboratively approximate a centralized Kalman filter estimate through localized processing and information exchange.

Olfati-Saber — Distributed Kalman Filter with Embedded Consensus Filters (CDC 2005/2007) Distributed filtering

Slide 9 — DKF: algorithm loop per time step

DKF: algorithm loop per time step

- Step 1: **local predict** (model propagation)
- Step 2: **local update** (measurement fusion)
- Step 3: L **rounds of consensus** (share & average information)
- Output: **local estimate** + uncertainty at each node



Takeaway

DKF integrates local model propagation, measurement updates, and network consensus to achieve accurate distributed state estimation.

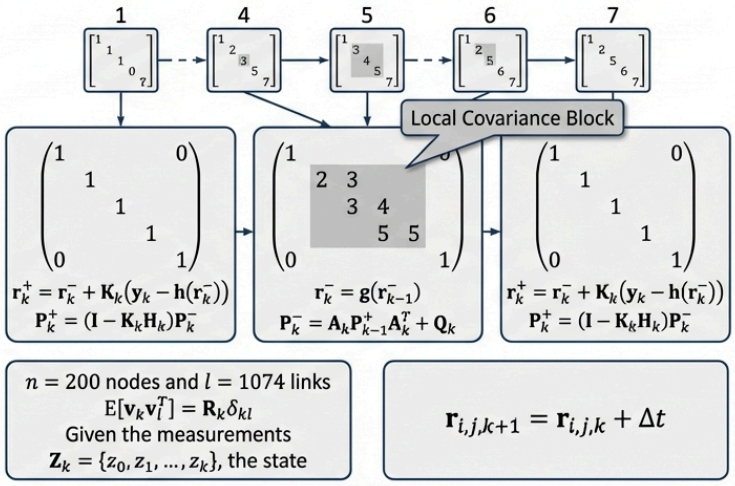
Source: Distributed Kalman Filter with Embedded Consensus Filters (Olfati-Saber, 2005/2007)

Distributed filtering

Slide 10 — Scalability: Localization idea (Traffic)

Scalability: Localization Idea (Traffic)

- Full covariance is $O(n^2)$: too expensive for large networks
- Localization assumes far nodes weakly correlated
- Keep local covariance blocks or taper with distance
- Result: real-time filtering at scale



$n = 200$ nodes and $l = 1074$ links
$\mathrm{E}[\mathbf{v}_k\mathbf{v}_l^T] = \mathbf{R}_k\delta_{kl}$
Given the measurements
$\mathbf{Z}_k = \{z_0, z_1, \ldots, z_k\}$, the state

$$\mathbf{r}_{i,j,k+1} = \mathbf{r}_{i,j,k} + \Delta t$$

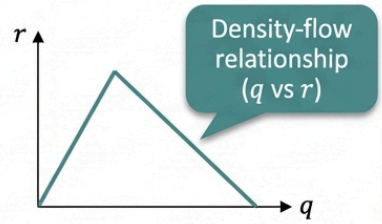
Takeaway:
Localization exploits weak spatial correlations to make filtering scalable and real-time.

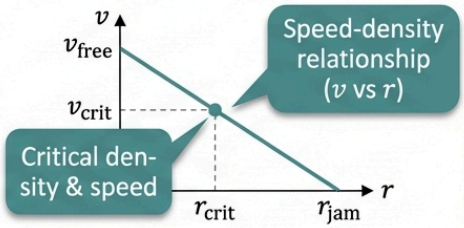
van Hinsbergen et al. — Localized EKF for Scalable Real-Time Traffic State Estimation (IEEE T-ITS 2012) Distributed filtering

Slide 11 — Traffic estimation with sparse sensors

Traffic Estimation with Sparse Sensors

- **State**: density/speed per road segment (high-dimensional)
- **Sensors**: sparse + noisy; coverage varies over space
- **Filtering** yields a complete map of state over the network





The graph-Laplacian matrix L is defined as $L = D - A$, where D is the diagonal degree matrix.
[Source: Shuman et al., "The emerging field of signal processing on graphs" (2013)]

Takeaway

Efficient traffic stat. estimation requires handling sparse, noisy data through filtering methods that account for spatial variations and uncertainty.

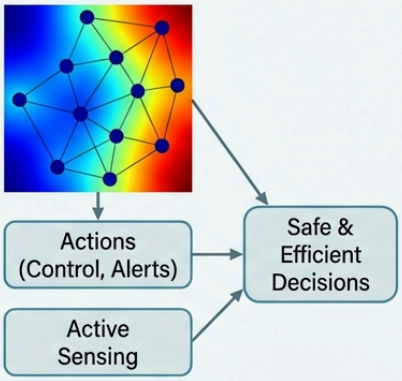
van Hinsbergen et al. — Localized EKF for Scalable Real-Time Traffic State Estimation (IEEE T-ITS 2012) Distributed filtering

Slide 12 — Posterior variance map: where we don't know

Posterior Variance Map: Where We Don’t Know

Symmetries in Data / Tasks...

- Variance is lowest near sensors, higher far from observations;
- Use uncertainty to prioritize actions (control / alerts);
- Also guides active sensing: where to place new sensors;
- Good UQ prevents overconfident wrong decisions



Takeaway

Reliable posterior variance quantification is crucial for safe, efficient decision-making and optimal data acquisition.

Graph Laplacian Definitions

$D = \mathrm{diag}(d_i)$, where d_i is the degree of vertex i;

$W_{ij} =$ weight of the edge; otherwise, $W_{ij} = 0$.

The Laplacian $L = D - W$, where D = diag(d_i) and W is the adjacency matrix.

Source: Extended Kalman Filter for Graph Signals in Nonlinear Dynamic Systems (2019)

Kalman Filter Update

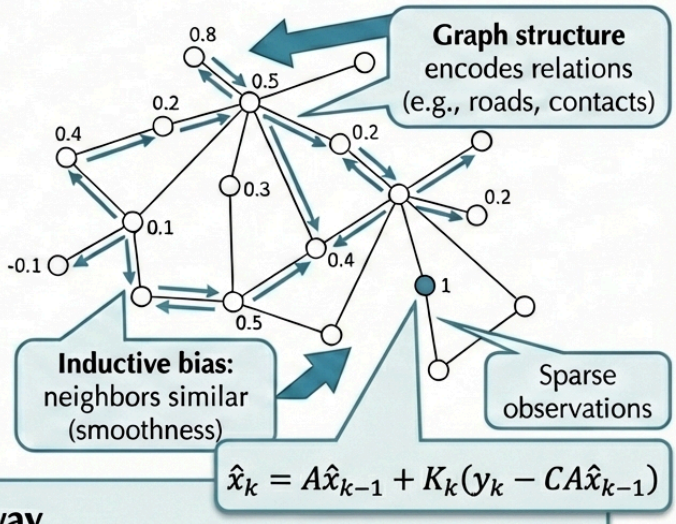
$$\hat{x}_k = A\hat{x}_{k-1} + Bu_{k-1} + K_k(y_k - CA\hat{x}_{k-1})$$

Source: Kalman Filtering Over Graphs: Theory and Applications (2018)

Slide 13 — Graph signal viewpoint: state lives on nodes

Graph Signal Viewpoint: State Lives on Nodes

- Treat x_t as a **graph signal**: one value per node.
- **Graph structure** encodes relations (roads, contacts, interactions).
- Many states are **smooth**: neighbors similar (**inductive bias**).
- Helps when observations are **sparse**



$$\hat{x}_k = A\hat{x}_{k-1} + K_k(y_k - CA\hat{x}_{k-1})$$

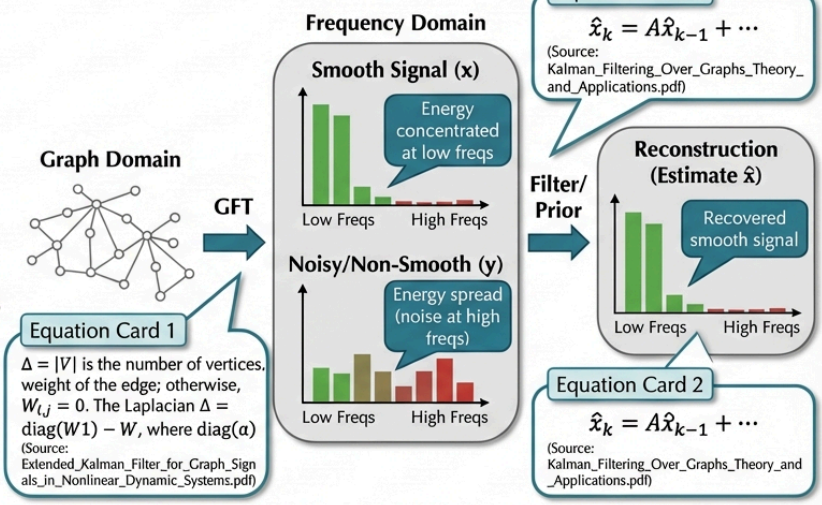
Takeaway

Viewing data as a graph signal allows leveraging structure and smoothness for better modeling, especially with sparse data.

Slide 14 — Smoothness prior + Graph Fourier intuition

Smoothness prior + GFT intuition

- **Graph Laplacian L** defines smoothness: $\mathbf{x^T L\, x}$
- **GFT** decomposes into graph frequencies (eigenvectors of **L**)
- **Smooth signals** concentrate energy at **low graph frequencies**
- This becomes a powerful **prior** for estimation



Takeaway

Leveraging the smoothness prior via GFT allows effective signal estimation by focusing on low-frequency components where information is concentrated.

Slide 15 — Kalman filtering over graphs: core idea

Kalman filtering over graphs: core idea

- **Classic KF** ignores graph structure → wastes information
- **Graph-KF** injects graph priors (smoothness / spectral models)
- Enables reconstruction with few sensors
- **Key:** combine temporal dynamics + spatial graph regularization

Takeaway

Graph-KF leverages both time-series data and network topology to improve state estimation, especially with sparse sensors.

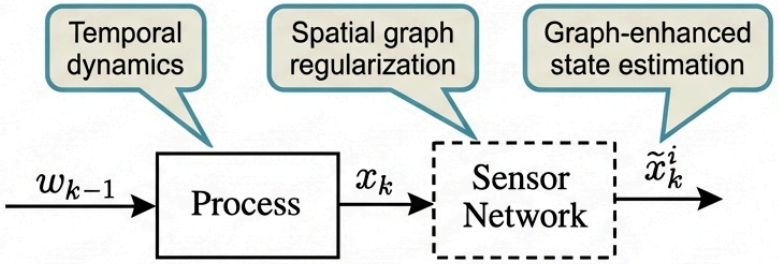


Fig. 1. State estimation using a wireless sensor network.

The transpose, ℓ_2 norm, and gradient are denoted $\{\cdot\}^\top$, $\Delta = |V|$ is the number of nodes. If there is an edge; otherwise, $[\mathbf{W}]_{i,j} = 0$.

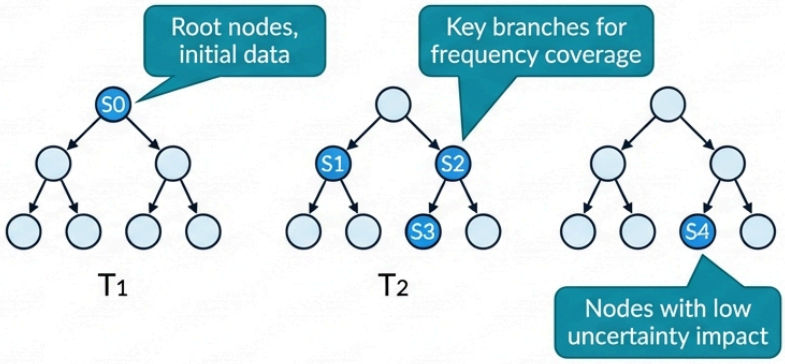
In general, graph signals are $\mathrm{G}_l(V, \xi_l)$, where there l is the time index, $V = \{1, \ldots, N\}$ denotes

Source: GSP-KalmanNet: Graph Signal Processing-based Kalman Fifter Network for Tracking Dynamic Graph Signals

Slide 16 — Sensor placement and sampling on graphs

Sensor placement and sampling on graphs

- Not all sensor nodes are equal (identifiability matters)
- Choose nodes that cover informative graph frequencies
- Goal: minimize posterior uncertainty quickly
- Variance maps + sampling theory guide placement



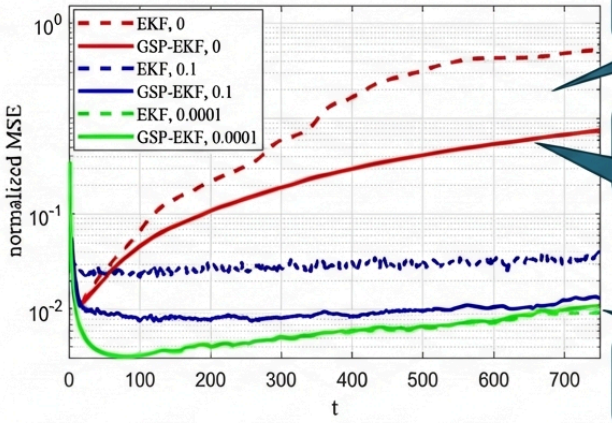
Takeaway

Effective sensor placement on graphs prioritizes identifiable nodes covering informative graph frequencies to rapidly minimize posterior uncertainty.

Slide 17 — Graph EKF: nonlinear dynamics on graph signals

Graph EKF: Nonlinear dynamics on graph signals

- **When dynamics/measurements nonlinear:** use EKF machinery
- **Linearize locally** but keep graph-structured priors
- Useful for **nonlinear diffusion/interaction models**
- Still yields **estimate + uncertainty** (approx.)



Standard EKF fails with high noise/uncertainty.

Graph priors (GSP-EKF) significantly improve performance.

Even with low assumed variance, GSP-EKF maintains lower error.

Takeaway

Takeaway: Graph EKF combines local linearization with graph signal priors to handle nonlinear dynamics, offering improved performance over standard EKF, especially under uncertainty.

Sagi et al. — EKF for Graph Signals in Nonlinear Dynamic Systems (ICASSP 2023)

Slide 18 — Graph UKF: sigma points for graph signals

Graph UKF: sigma points for graph signals

- **Avoid Jacobians** by propagating sigma points
- **Graph structure** can reduce effective dimension (spectral/low-rank)
- Often **more stable** than EKF under stronger nonlinearities
- Still keeps **KF predict/update** backbone

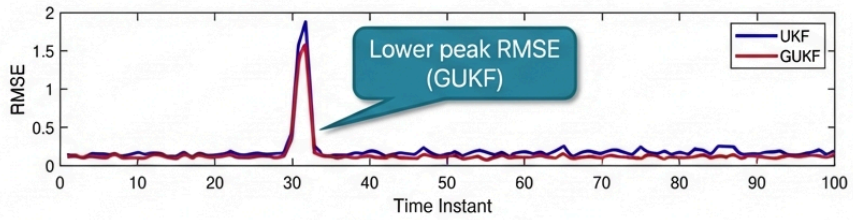


Fig. 1. RMSE with $p = 1$.

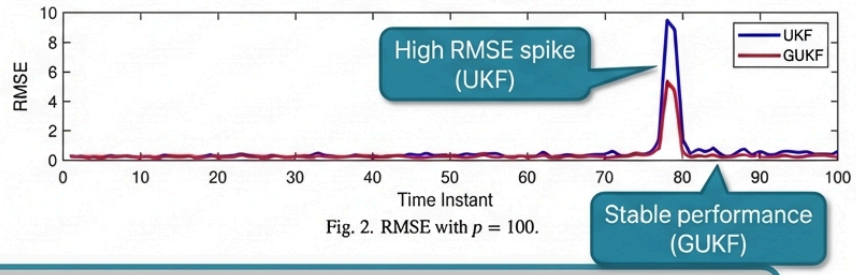


Fig. 2. RMSE with $p = 100$.

Takeaway

Graph UKF leverages graph structure for stable, Jacobian-free nonlinear estimation by propagating sigma points.

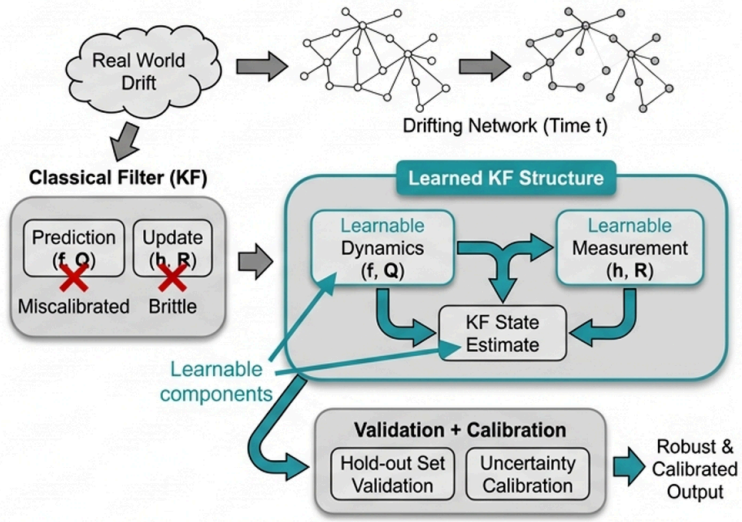
"Li et al. — Unscented Kalman filter of graph signals (Automatica 2023)"

Distributed filtering

Slide 19 — Model mismatch is the norm (why learning helps)

Model mismatch is the norm (why learning helps)

- Real networks drift; **Q,R** and **f,h** often misspecified
- Classical filters can become brittle or miscalibrated
- **Idea:** keep KF structure, learn the hard parts
- Need validation + uncertainty calibration



Takeaway

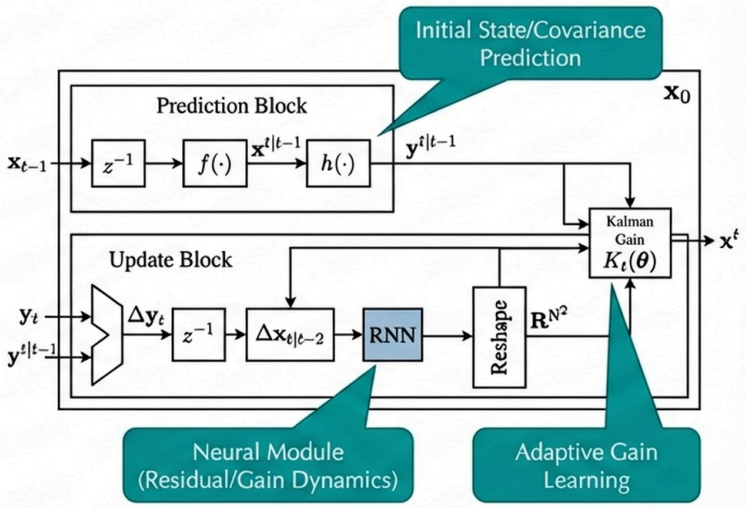
By integrating learning into the Kalman Filter's established structure, we can overcome the limitations of misspecified models and achieve robust, calibrated performance in dynamic environments.

Motivation

Slide 20 — GSP-KalmanNet: neural-aided filtering on graphs

GSP-KalmanNet: neural-aided filtering on graphs

- **Neural module**: learns components (e.g., gain / residual dynamics)
- **Graph structure**: helps generalize across nodes
- Retains **sequential filtering** behavior (tracking)
- **Goal:** improve **accuracy** under mismatch



Takeaway

Combines Combines sequential Kalman filtering with a neural module and graph structure for enhanced tracking accuracy under model mismatch.

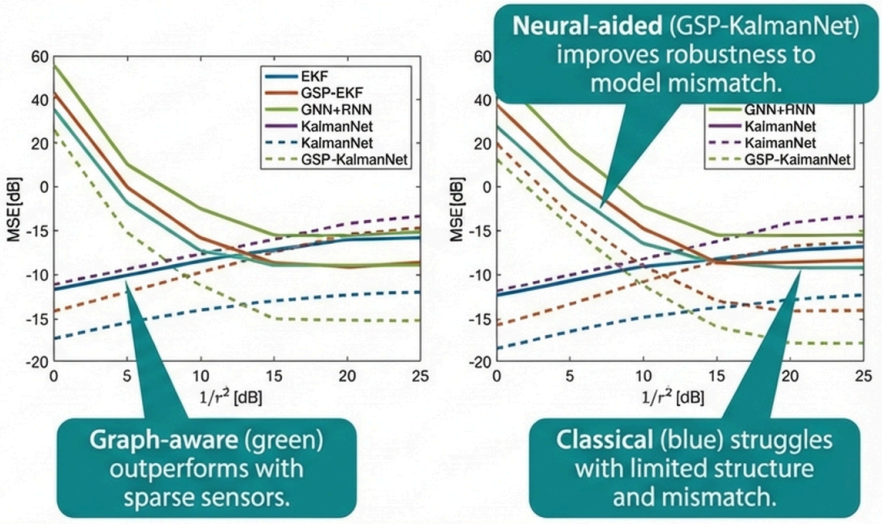
Buchnik et al. — GSP-KalmanNet: Tracking Graph Signals via Neural-Aided Kalman Filtering (IEEE TSP 2024)

Distributed filtering

Slide 21 — Results and tradeoffs: classical vs graph vs neural

Results + tradeoffs: classical vs graph vs neural

- **Graph-aware methods** shine when sensors are sparse sparse
- **Neural-aided methods** help when model mismatch dominates
- **Cost**: training + risk of miscalibration
- Use **uncertainty tools** (calibration) for safety



Takeaway

Takeaway

Optimal tracking requires balancing graph structure knowledge, model fidelity, and the costs of training and calibration.

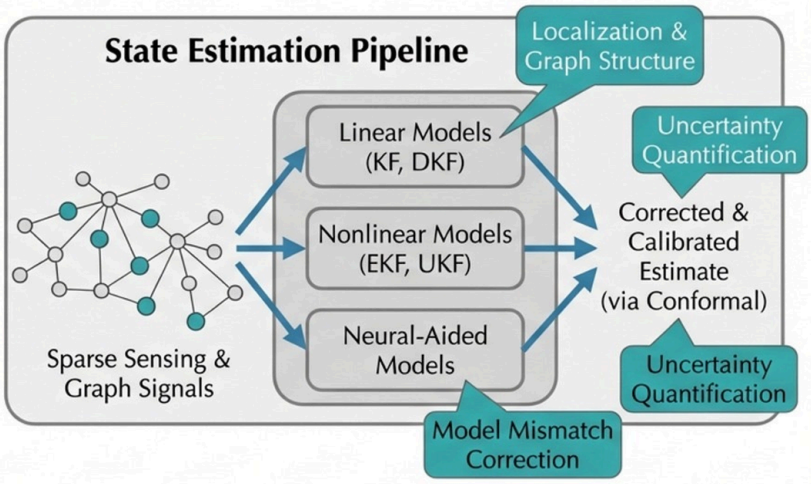
Buchnik et al. — GSP-KalmanNet: Tracking Graph Signals via Neural-Aided Kalman Filtering (IEEE TSP 2024)

Distributed filtering

Slide 22 — Summary: what to use when

Summary: what to use when + takeaways

- Linear + good model: **KF / DKF**; add localization for scale
- Sparse sensing on graphs: **graph-KF** + smart sensor placement
- Nonlinear: **EKF/UKF**; graph EKF/UKF for graph signals
- Mismatch: **neural-aided** + conformal calibration



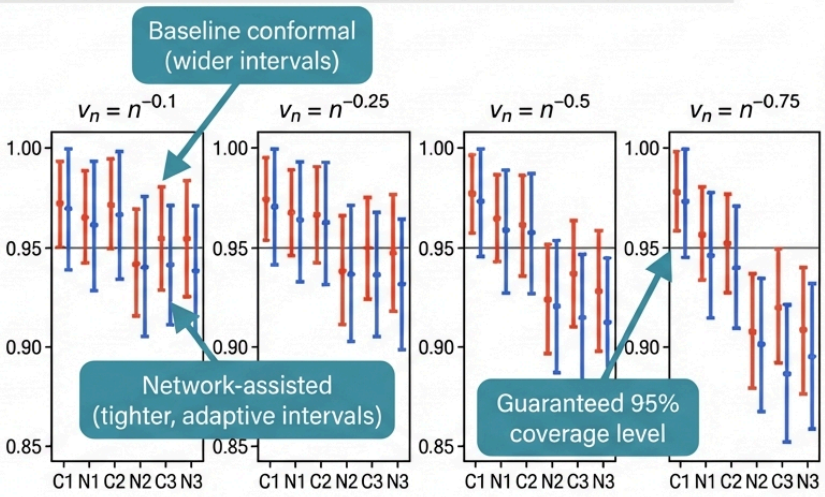
Takeaway: Choose the right filter and sensor placement based on linearity, graph structure, and model uncertainty, using neural aids for mismatch.

Module summary

Slide 23 — Calibrated uncertainty: conformal layer on networks

Calibrated uncertainty: conformal layer on networks

- **Model-based uncertainty** can be miscalibrated under mismatch
- **Conformal prediction** gives finite-sample coverage guarantees
- **Network-assisted conformal** can tighten intervals using graph
- **Practical safety layer** on top of predictors/filters



Takeaway

Conformal layers leverage graph structure to provide certified, **tighter** uncertainty intervals for reliable network-based predictions.

Slide 24 — Tracking a dynamic network (graph changes)

Tracking a Dynamic Network (Graph Changes)

- Sometimes topology evolves: edges/weights change over time
- Can treat graph parameters as part of the state
- Harder: higher uncertainty + identifiability issues
- Connects to joint state+parameter estimation

Algorithm 1: Kalman filter for tracking network dynamic

Input: Compute input, filter for dynamic network, horizons, via input, static, compute filter for all networks

Initialization: $\mathbf{x} = c_0, l_1, b_2, \mathbf{x}_5, s_1, ..., h_i, h_o,$

Algorithm Steps:

1: 1. Precomputation step:
$D = \mathbf{m}(\mathbf{x}(t) - U_1, B_i(U_{m+t-1}, R_i, S_k)$

2: 2. Prediction step:
$S_i = \mathbf{w}_t \mathbf{x}'_{n+1} - R_o((\mathbf{w}(t)_{k+1}, \mathbf{x}_i)$

3: Update step:
$S_i = \mathbf{w}_t \mathbf{x}'_{n+1} - R_o((\mathbf{w}(t)_{k+1}, \mathbf{x}_i)$

4: 4. Thresholding step:
$\hat{y} = A^{-1} + B_i^{-1} \mathbf{x}_i(\mathbf{w}_t, R_n + \gamma)$

5. Thresholding step:
$E_{vp} = A_e \leq \mathbf{E}(\mathbf{m}^{m-1} \mid \gamma_j \to 1)$

5: 5. Thresholding propagation step: $A_n = A_n U_{n-1}, A_t, R_i - M_n$

Output: All the smaller input cus networks

Core filter loop for state & parameter estimates

Topology change detection & updates

Source: Dabush & Routtenberg — Kalman Filter for Tracking Network Dynamic (ICASSP 2024)

Takeaway Jointly estimating state and evolving network parameters increases complexity but is essential for accurate tracking.