Signal and Information Processing Laboratory
Institut für Signal- und Informationsverarbeitung

KalmanNet - Data-Driven Kalman Filtering

Guy Revach[1] **Nir Shlezinger**[2] **Ruud J. G. van Sloun**[3,4] **Yonina C. Eldar**[5]

Partially based on joint work with Adrià López Escoriza[1]

[1]The Institute for Signal and Information Processing (ISI), ETH Zürich, Switzerland

[2]The School of ECE, Ben-Gurion University of the Negev, Beer Sheva, Israel

[3]The EE Dpt., Eindhoven University of Technology, Eindhoven, The Netherlands

[4]Phillips Research, Eindhoven, The Netherlands

[5]The Faculty of Math and CS, Weizmann Institute of Science, Rehovot, Israel

★ *We thank Hans-Andrea Loeliger for helpful discussion and support*

State Estimation

State estimation of a dynamical system in real-time is one of the most fundamental problems in signal processing with countless real-life applications.

In this work we focus on architectures that can be deployed on embedded mobile devices such as drones and vehicular systems with limited computational resources.

Kalman Filter — Classical Approach

- For systems that are well-represented by a fully known linear Gaussian state-space (SS) model, the Kalman filter (KF) is a low complexity optimal solution.
- KF and its variants are *model-based (MB)* algorithms, their performance critically depends on the validity of the key assumption that the underlying SS model is accurately known.
- However, both linearity and accurate knowledge of the model are often not encountered in practice.
- For non-linear models, these approaches are not theoretically optimal, and suffer from severe degradation in the face of strong non-linearity.

Success of SOA NNs

- *SOA* (recurrent) neural networks (NNs) have demonstrated remarkable success in real-life (time series) applications
- NNs can catch the subtleties of the true generative process and replace the need to explicitly characterize the domain of interest.
- They can be trained in an end-to-end model-agnostic manner from a large quantity of data to capture complex dynamics.
- These data-driven architectures lack the interpretability of model-based methods and tend to require many trainable parameters.
- These constraints limit the application of deep NNs for real-time state estimation on hardware-limited mobile devices.

KalmanNet — Our Approach

- We integrated a dedicated recurrent neural network into the KF flow and learned to carry out Kalman filtering under non-linear complex dynamics, with *unknown noise* and *model mismatch*.
- Using the structure of the SS model, we retained data efficiency and interpretability of the classic algorithm, and achieved a low-complexity solution.
- Numerical evaluations shows that we outperform the classic filtering methods
- The design of KalmanNet was found to be scalable to other state estimation tasks; e.g., smoothing, and proved to be suitable for deployments on systems with very limited computational resources - the ETH Zürich autonomous racing car.

Agenda

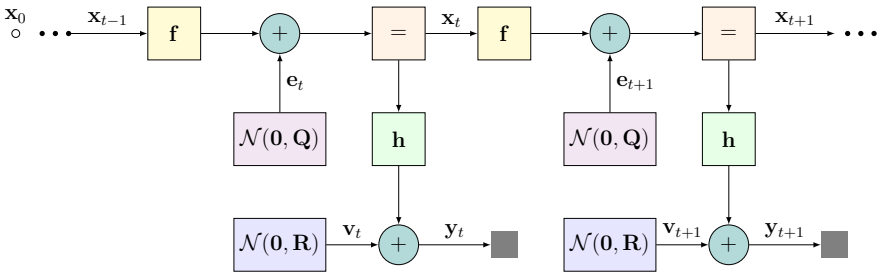
1. Motivation and Abstract
2. State Space Model
3. Classical Kalman Filtering
4. KalmanNet Architecture
5. Extensions and Conclusions
   - Kalman Smoothing
   - Velocity Estimation

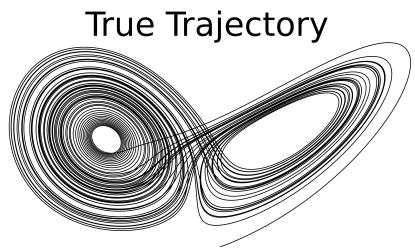
State Space Model

We consider a partially-observable time-invariant dynamical system represented by (possibly) non-linear, Gaussian continuous SS evolution model in discrete time $t \in \mathbb{Z}$:

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}) + \mathbf{e}_t, \quad \mathbf{e}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}), \quad \mathbf{x}_t \in \mathbb{R}^m. \tag{1a}$$

$$\mathbf{y}_t = \mathbf{h}(\mathbf{x}_t) + \mathbf{v}_t, \quad \mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \quad \mathbf{y}_t \in \mathbb{R}^n. \tag{1b}$$





Classical Kalman Filtering

- The KF was introduced in a pioneering work by R.E. Kalman in the 1960s.
- It is an optimal minimum mean-squared error (MMSE) estimator that is applicable to time varying linear systems with additive white Gaussian noise (AWGN).
- Because it is a recursive linear filter with low-complexity and a sound theoretical basis, it is considered the workhorse of state estimation in discrete-time.
- It was applied to various tracking problems (e.g., radar and a ballistic missile). Most noteworthy was the use of NASA to estimate the position and velocity of a space vehicle in a trip to the moon.

(Extended) Kalman Filter Computation

KF estimates $\mathbf{x}_t$ using only the new observation $\mathbf{y}_t$ and the previous estimate $\hat{\mathbf{x}}_{t-1}$, with constant computational complexity.

- *Prediction* of the *a priori* moments:

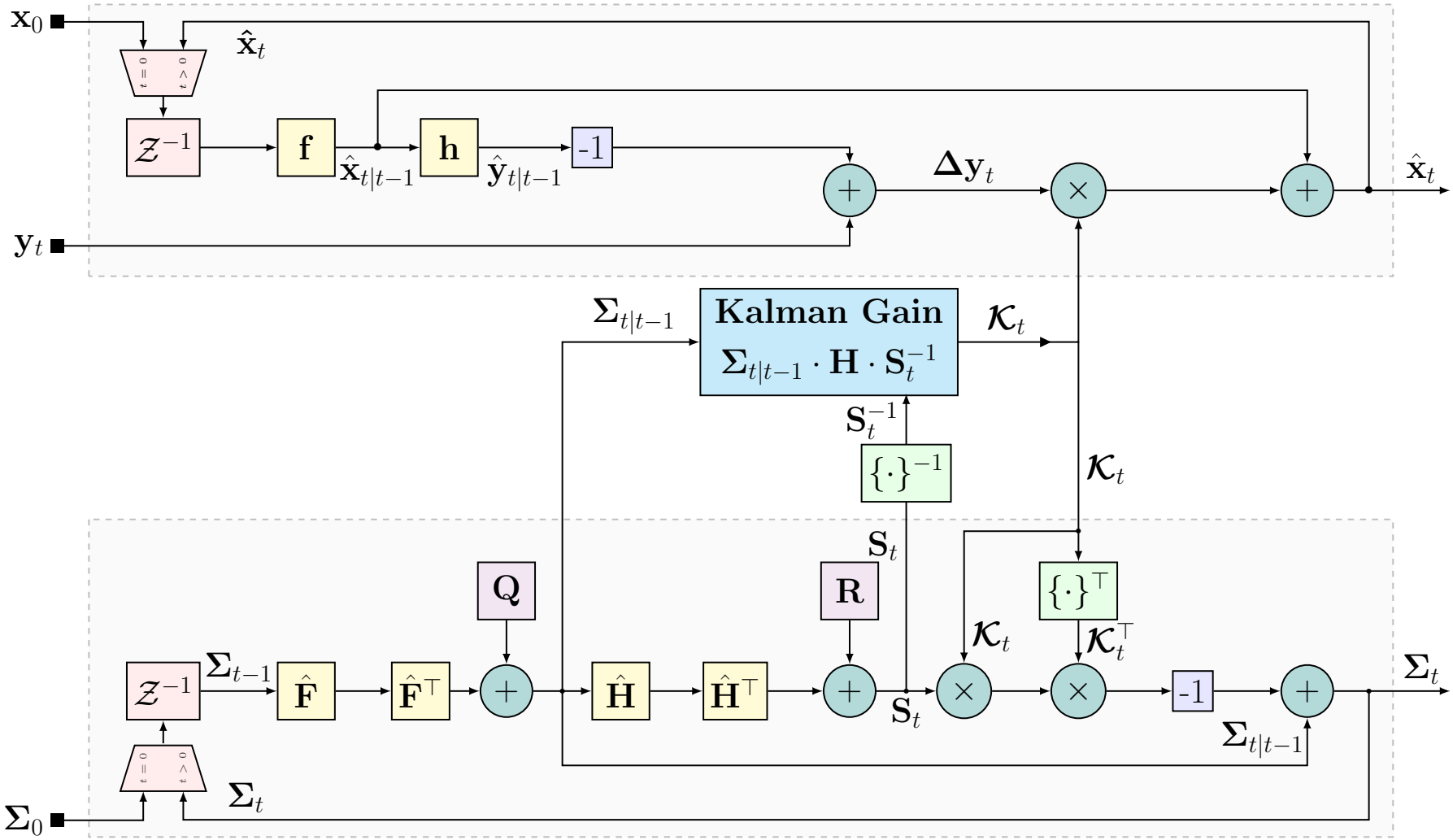
$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{f}\left(\hat{\mathbf{x}}_{t-1}\right), \qquad \Sigma_{t|t-1} = \hat{\mathbf{F}}_t \cdot \Sigma_{t-1} \cdot \hat{\mathbf{F}}_t^{\top} + \mathbf{Q},$$
$$\hat{\mathbf{y}}_{t|t-1} = \mathbf{h}\left(\hat{\mathbf{x}}_{t|t-1}\right), \qquad \mathbf{S}_{t|t-1} = \hat{\mathbf{H}}_t \cdot \Sigma_{t|t-1} \cdot \hat{\mathbf{H}}_t^{\top} + \mathbf{R}.$$

- *Updating* the *a posteriori* moments given the new observed $\mathbf{y}_t$:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t|t-1} + \mathcal{K}_t \cdot \Delta\mathbf{y}_t, \qquad \Delta\mathbf{y}_t = \mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}$$
$$\Sigma_t = \Sigma_{t|t-1} - \mathcal{K}_t \cdot \mathbf{S}_{t|t-1} \cdot \mathcal{K}_t^{\top}, \qquad \mathcal{K}_t = \Sigma_{t|t-1} \cdot \hat{\mathbf{H}}_t^{\top} \cdot \mathbf{S}_{t|t-1}^{-1}.$$

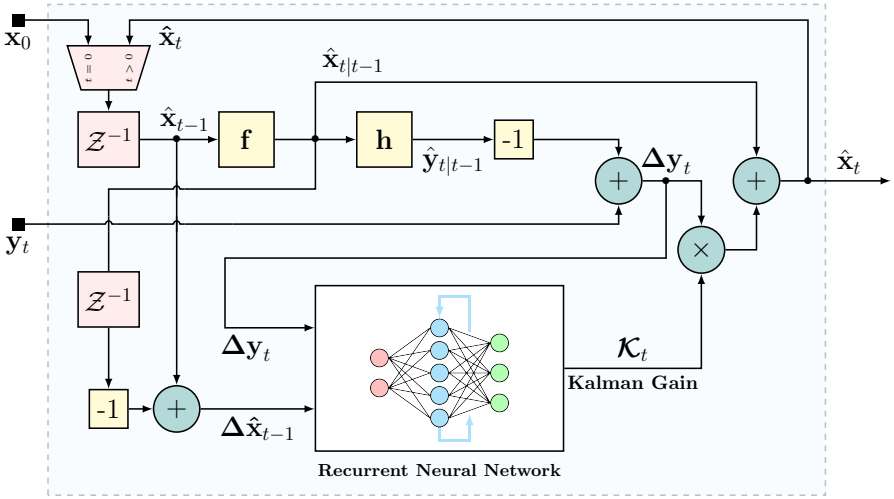
$\hat{\mathbf{F}}_t$ and $\hat{\mathbf{H}}_t$ are the matrices of partial derivatives (i.e., *Jacobians*) of f and h, evaluated at $\hat{\mathbf{x}}_{t|t-1}$.

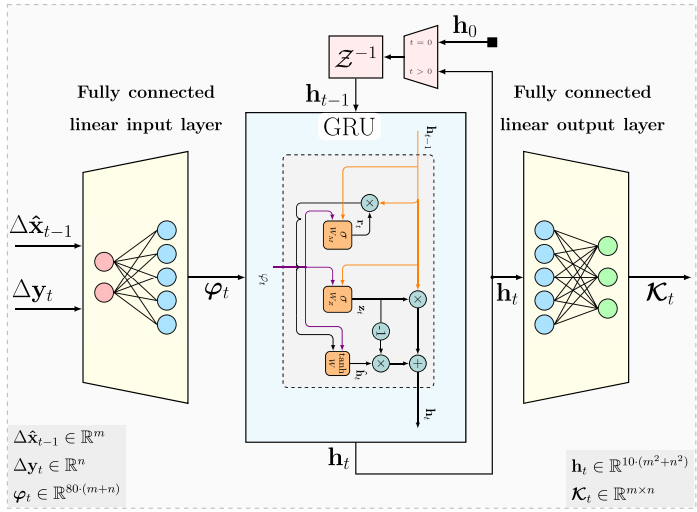
Kalman Filter — Block Diagram



KalmanNet Design Idea - Maintain the Basic KF Flow

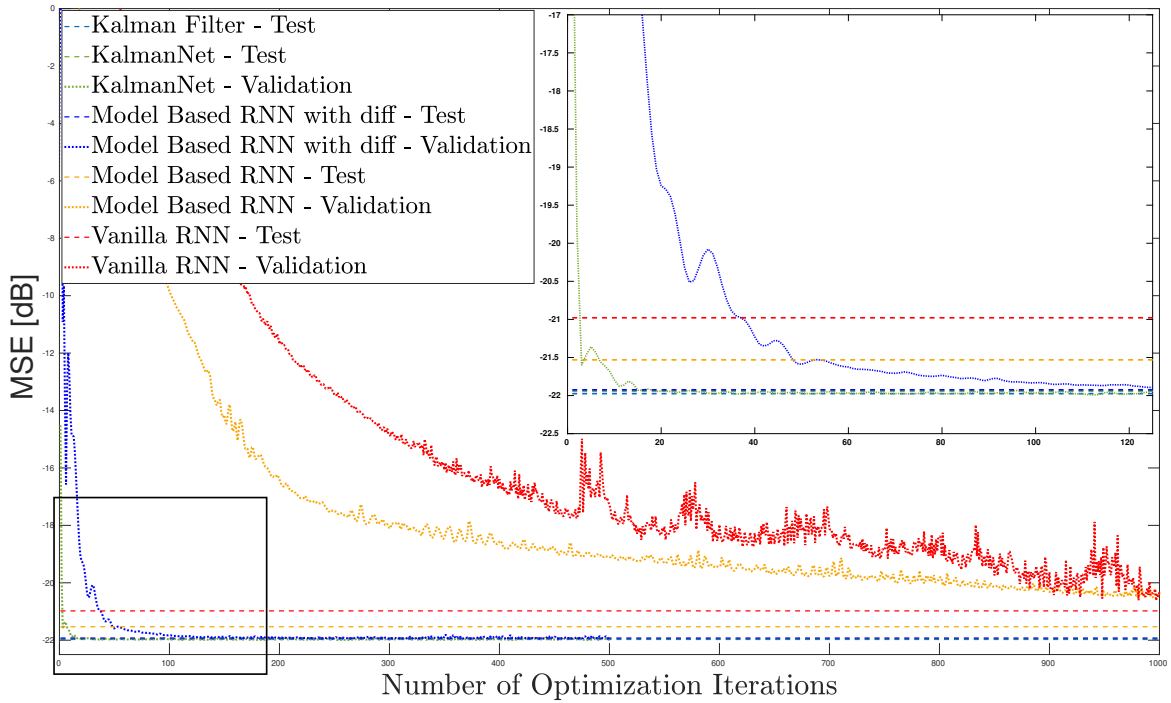
- What should be the SS *internals* that needs to be learned?
  The Kalman gain which depends on unknown second order moments.
- What should be the architecture of such a network?
  The recursive nature of the Kalman gain computation indicates that its learned module should involve an internal memory element as a recurrent NN with low complexity - a gated recurrent unit.





From which input signals will the network learn?

- The difference between subsequent observations
- Innovation
- The difference between subsequent posterior estimates
- The difference between prior to posterior



How will this NN be trained from data?

KalmanNet is trained **offline** using labeled data. The data set is composed of N length T trajectories, denoted $\{\mathbf{Y}_i, \mathbf{X}_i\}_1^N$:

$$\mathbf{Y}_i = \left[\mathbf{y}_1^{(i)}, \ldots, \mathbf{y}_T^{(i)}\right] \qquad \mathbf{X}_i = \left[\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_T^{(i)}\right]$$

With a **MSE loss function**:

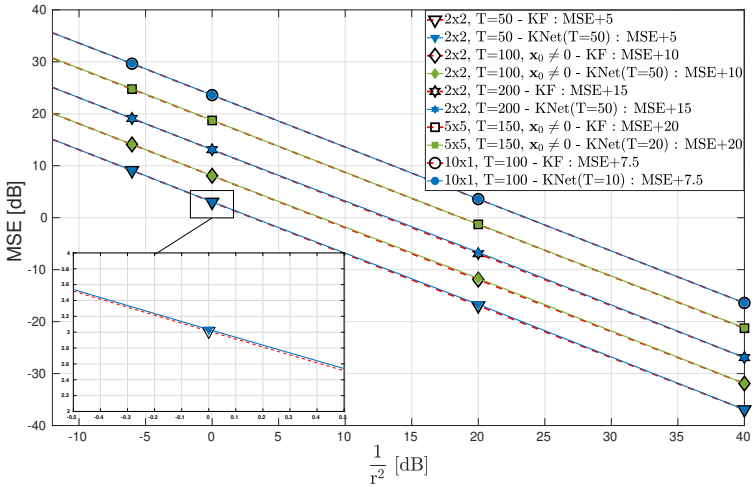
$$\ell_i(\Theta) = \frac{1}{T}\sum_{t=1}^{T}\left\|\Psi_\Theta\left(\hat{\mathbf{x}}_{t-1}^{(i)}, \mathbf{y}_t^{(i)}\right) - \mathbf{x}_t^{(i)}\right\|^2 + \gamma \cdot \|\Theta\|^2,$$

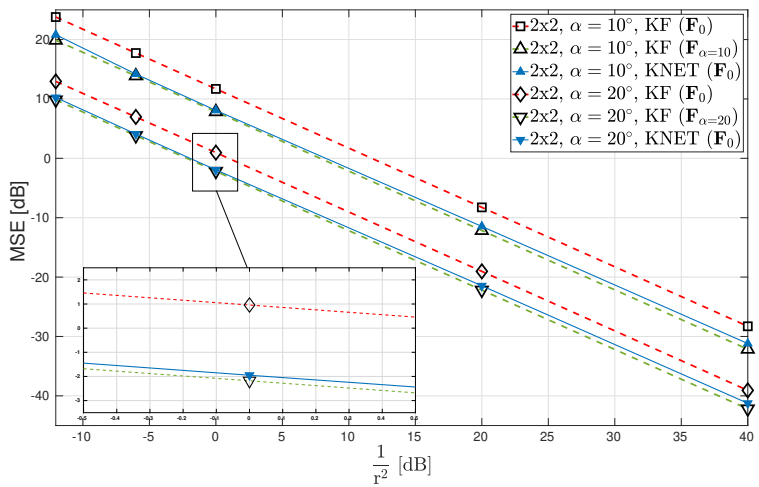
Optimisation through a variant of mini-batch stochastic gradient descent: randomly choose $M < N$ trajectories indexed $i_1^k, \ldots, i_M^k$ from the training set, computing the batch loss as

$$\mathcal{L}_k(\Theta) = \frac{1}{M}\sum_{j=1}^{M}\ell_{i_j^k}(\Theta).$$

Linear Systems

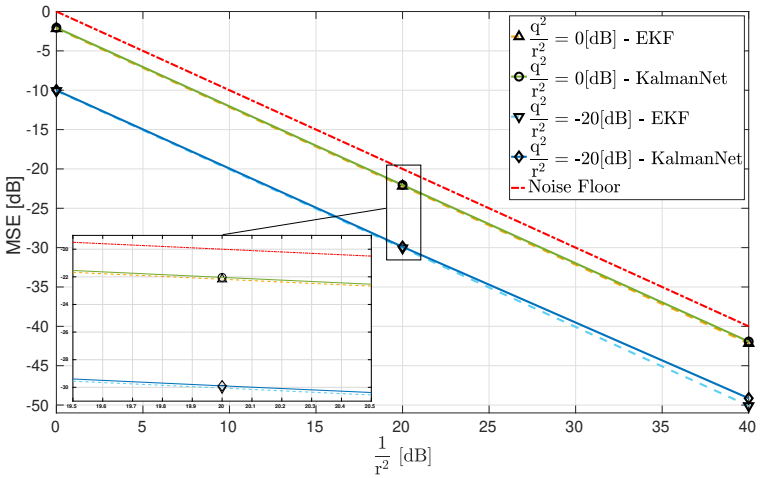
- **Full information** - KalmanNet achieves MMSE lower bound. It is applicable to different trajectory lengths and initial conditions than those on which it was trained - shows that it **learns to filter** rather to reconstruct trajectories from training data.
- **Partial information** - We rotate the evolution matrix **F** by $\alpha = 10°$. While KF opens a noticeable gap from the MMSE, KalmanNet is able to learn to close this gap from data.

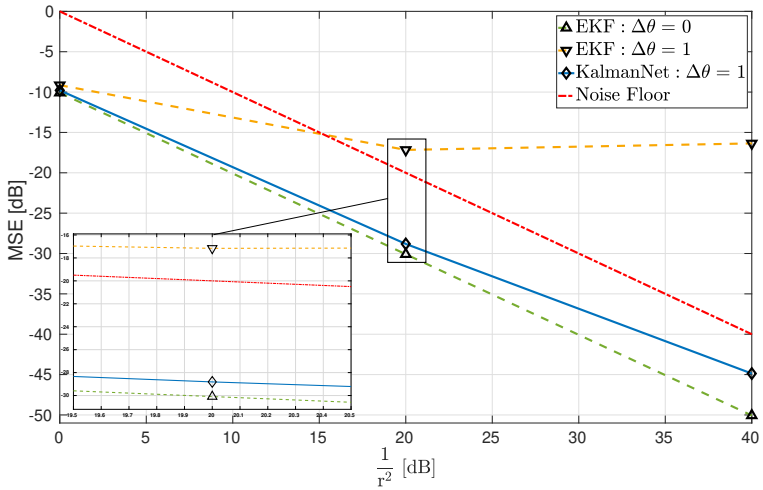




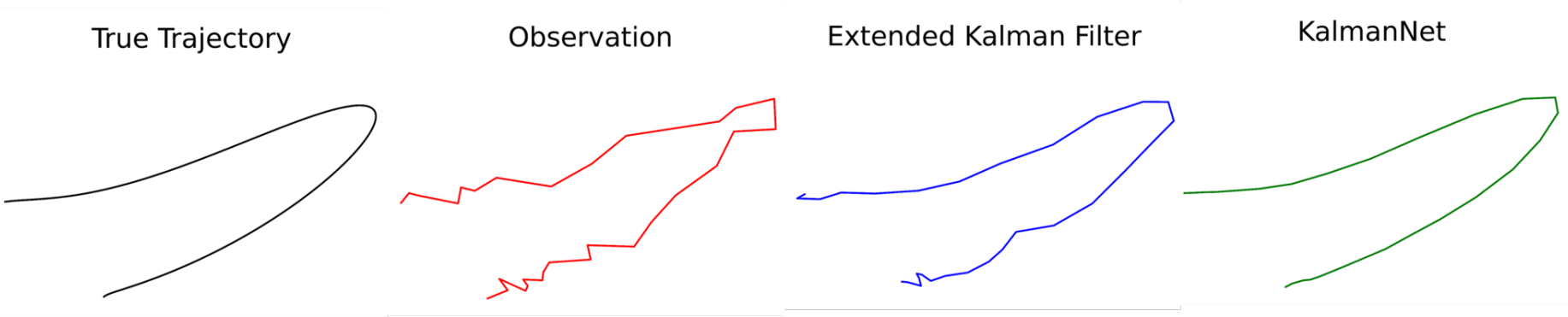
Lorenz Attractor - Non-Linear chaotic system

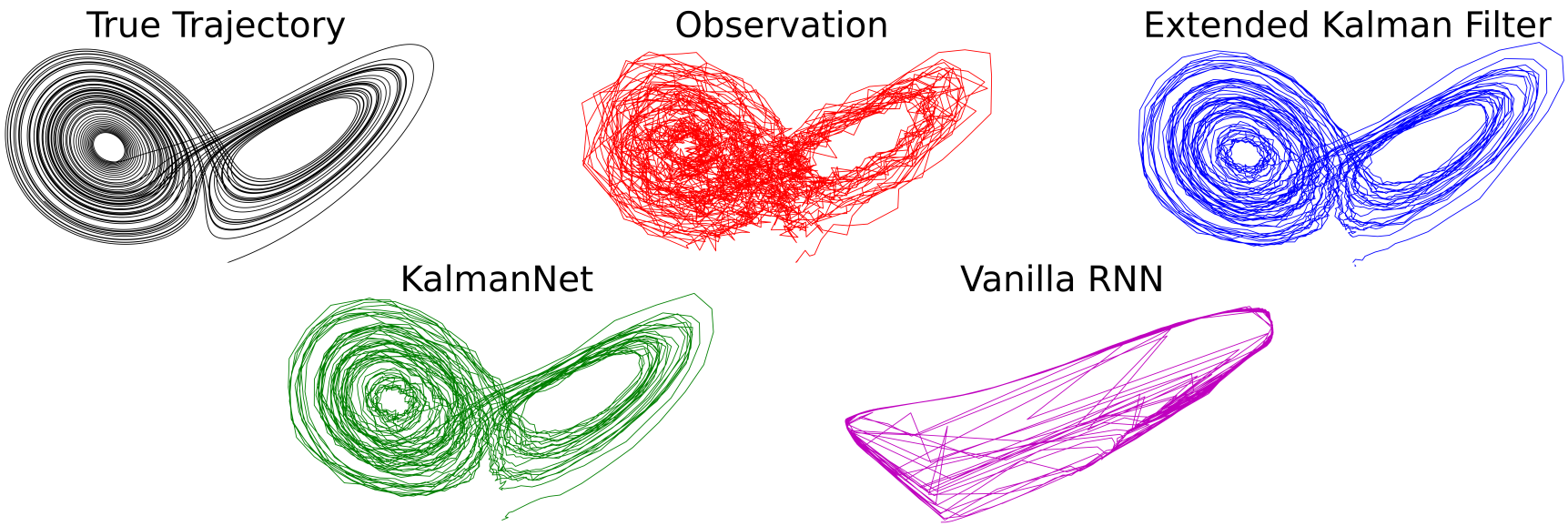
- Set of differential equations that simulate complex dynamics. KalmanNet recovers the extended Kalman filter (EKF) results in a scenario in which it is nearly optimal.
- We rotate the observation matrix **H** by $\theta = 1°$. While EKF almost performs worse than the noise floor, KalmanNet is able to learn to close this gap from data.





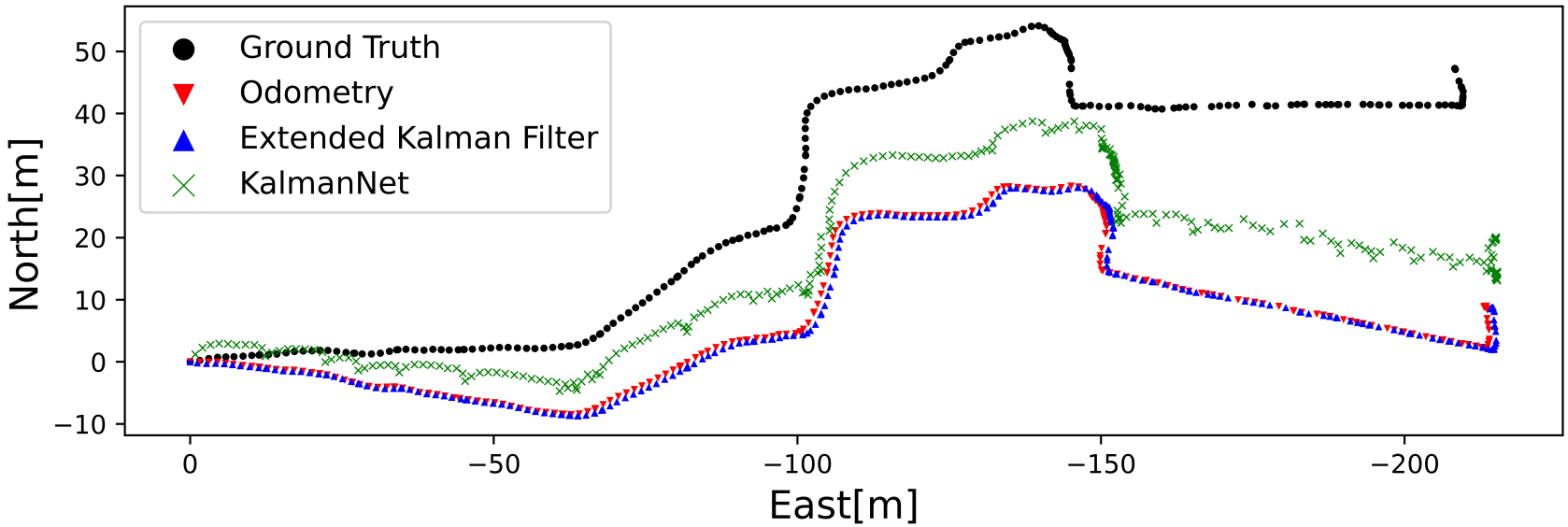
Non-Linear Decimation — KalmanNet Outperforms





NCLT — North Campus Long-Term Dataset

GPS and odometry sensor data of a *Segway* moving through the campus of Michigan Uni. with ground truth positioning available.
Model: 2D model with constant velocity. State: $\mathbf{x} = (p_x, v_x, p_y, v_y)^\top$.

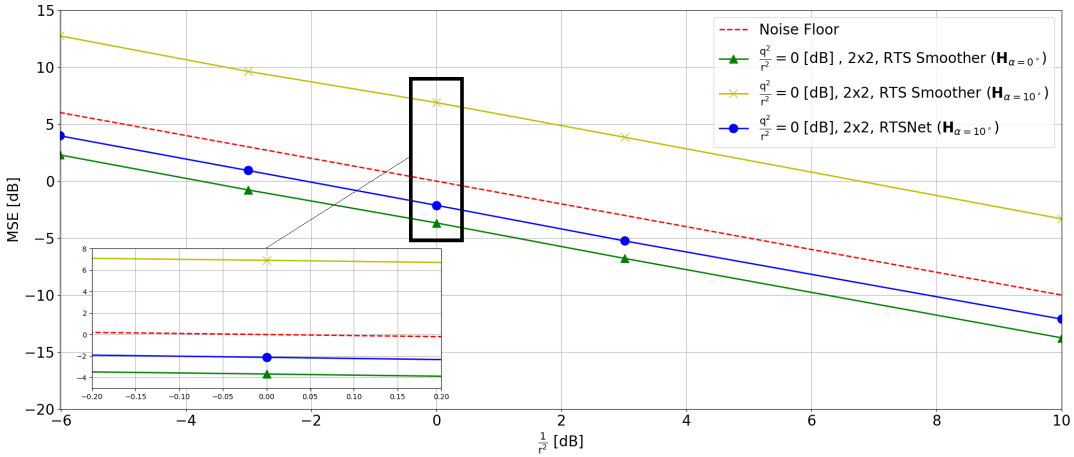


Kalman Smoothing - Offline Estimation (with William Ni)

We utilize the structure of the RTS smoother and add a *backward*-pass to the *forward*-pass. We learn two gains simultaneously; the *forward gain* and the *backward gain*; using two recurrent neural networks in a cascade.

$$\hat{\mathbf{x}}_{t|T} = \hat{\mathbf{x}}_{t|t} + \mathcal{K}_{s,t} \cdot \Delta\mathbf{x}^{+}_{t+1|T}, \tag{3a}$$

$$\Delta\mathbf{x}^{+}_{t+1|T} = \hat{\mathbf{x}}_{t+1|T} - \hat{\mathbf{x}}_{t+1|t}. \tag{3b}$$



Velocity Estimation for Autonomous Racing Car

Embedding KalmanNet in the ETH Zürich autonomous (driver-less) racing car as a velocity estimation module raises various challenges:

- Critical for performance.
- Critical for safety $\rightarrow$ robust to observation outliers.
- Sensors have different sensing frequencies.
- Deployment on a low-end computer with limited computational resources.

Conclusions

- KalmanNet combines deep learning with the classic KF.
- Our design identifies the SS-model-dependent computations of the KF, replacing them with a dedicated NN; i.e., Kalman gain.
- Doing so enables KalmanNet to carry out real-time state estimation in the same manner as the KF, while learning to overcome model mismatches and non-linearities.
- KalmanNet uses a relatively compact NN that can be trained with a relatively small data set, and infers with a reduced complexity, making it applicable for high dimensional SS models and computationally limited devices.