Robust State Estimation and Deep Learning for Time Series Analysis

by

Matthew L. Weiss

A Dissertation

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

in

Data Science

August 12, 2020

APPROVED:

Professor Randy C. Paffenroth Worcester Polytechnic Institute Advisor	Professor Xiangnan Kong Worcester Polytechnic Institute Committee Member
Professor Jacob R. Whitehill Worcester Polytechnic Institute Committee Member	Dr. Joshua R. Uzarski U.S. Army CCDC-SC External Committee Member
Professor Elke A. Rundensteiner Worcester Polytechnic Institute Program Director	

Abstract

Filtering is the process of recovering a true state, $\mathbf{x}(t)$, at some time t, from a noisy measurement process $\mathbf{z}(t)$, given all noisy measurements up to and including time t. One common filtering technique is the Kalman Filter, which is known to be the optimal filter under certain conditions. However, while the Kalman Filter is shown to be the optimal conditional minimum variance estimator when the random processes involved are Gaussian, many real-world applications involve non-Gaussian processes. In this regard, one Kalman Filter parameter of particular importance is the measurement noise covariance matrix $\mathbf{R}$, which is the covariance of the noisy measurements $\mathbf{z}(t)$ obscuring the true state $\mathbf{x}(t)$. In practice, the precise determination or estimation of $\mathbf{R}$ is difficult and often relies on cross-validation and domain knowledge. To address this problem, we develop a novel deep learning-Kalman Filter hybrid algorithm called the Autoencoder-Kalman Filter (AEKF). The AEKF leverages the computational power of an autoencoder with the well-known mathematical foundation of the Kalman Filter. This merger results in a robust state estimation system, which is able to successfully perform state estimation on a variety of function families and a variety of noise types.

The training of this robust state estimation system is facilitated by a technique called *domain randomization*. The standard application of domain randomization is to the modeling of physical parameters in deep learning systems. In our approach we apply domain randomization to the state estimation of different function families, where each function family is thought of as occupying distinct regions of a Hilbert Space.

In terms of demonstrated applications, we first address the problem of chemical classification using noisy time series measurements collected from chemical sensors. These experiments show that preprocessing noisy time se-

ries data with a Kalman Filter significantly improves early and accurate chemical discrimination. In particular, when Kalman Filter first derivative estimates of a sensor's underlying state are used for feature engineering, early and accurate detection of chemical agents is improved. Building on the success of the Kalman Filter for feature engineering, we explore the AEKF's state estimation capabilities on progressively more general function families and noise types using our variant of domain randomization. Since the AEKF is a deep learning-Kalman Filter hybrid model, we compare its performance against a standard Kalman Filter and a Long-Short Term Memory (LSTM) recurrent neural network. In the majority of our experiments, the AEKF outperforms both the Kalman Filter and the LSTM, particularly in the context of mitigating the effect of outlier measurements.

Specifically, we begin by training three different AEKF models to perform state estimation on exponential, sigmoidal, and sinusoidal functions with added Gaussian, bimodal, and Cauchy noise. Next, we train a single AEKF model to filter all three of these function families on the same noise varieties. Lastly, at the most general level we train the AEKF to filter truncated Taylor Polynomials with the same three noise distributions. Given the success of the AEKF in all these experiments, we develop an extension of it called the Autoencoder-Interacting Multiple Model Kalman Filter and apply it to simulated target tracking problems.

Lastly, given the successful application of the AEKF in a variety of contexts, we derive a theorem which explains the AEKF's outlier mitigation capabilities in terms of learned matrix eigenvalues. This theorem not only provides insight into the workings of the AEKF, but serves as a metric to judge whether or not the AEKF is working correctly. As a result, our theorem is a valuable tool for the design of the AEKF's neural network architecture and more generally

demonstrates how well-understood mathematical models can inform neural network design.

Acknowledgments

First, I would like to thank my advisor, Professor Randy C. Paffenroth. I first approached him on the advice of some people in the mathematical sciences department. The second time we met he mentioned a project involving chemical sensors and the Army. Looking back it has been an amazing and exciting journey over the last four years. My weekly meetings with Randy were the intellectual highlight of my week and I always looked forward to them. Although Randy is an amazing repository of knowledge, his humility allows him to listen while, at the same time, pressing his point when appropriate. If it were not for the many hours he dedicated to helping me with my research I would not be where I am today.

I would like to thank Dr. Joshua R. Uzarski from the U.S. Army Combat Capabilities Development Command Soldier Center (CCDC-SC), who is not only the external member of my dissertation committee, but also has been an advisor to me since the beginning of my PhD program. In addition to supporting my program financially through the ORISE program, working with Josh has given me the opportunity to work with real-world data from day one.

I thank my two WPI committee advisors, Professor Jacob R. Whitehill and Professor Xiangnan Kong, for taking the time to discuss my research one-on-one, for the time they have invested in reading my manuscripts, and for providing valuable feedback.

I would also like to thank Professor Elke A. Rundensteiner for going out of her way to meet with me to discuss the WPI Data Science program when I was first considering applying. I am grateful to my research group members for their weekly presentations, especially Chong Zhou for providing help with both technical and logistical questions throughout my program. Thanks to the Data Science

Program's administrative assistant Mary Racicot for the amazing administrative assistance and guidance.

Thanks to my family for being supportive and understanding of my time constraints during the last four years.

Finally, I appreciate the patience and support of my wife Rashmi, especially considering I began my PhD program two weeks after our wedding.

Papers Contributing to this Dissertation

Published First Author Papers

Matthew L. Weiss, Randy C. Paffenroth, Jacob R. Whitehill, and Joshua R. Uzarski. "Deep Learning with Domain Randomization for Optimal Filtering." *Proceedings of the 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Boca Raton, FL, USA, 2019*

Matthew L. Weiss, Randy C. Paffenroth, and Joshua R. Uzarski. "The Autoencoder-Kalman Filter: Theory and Practice." *Proceedings of the 53rd Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, 2019*

Matthew L. Weiss, Michael S. Wiederoder, Randy C. Paffenroth, Eric C. Nallon, Collin J. Bright, Vincent P. Schnee, Shannon K. McGraw, Michael P. Polcha, and Joshua R. Uzarski. "Applications of the Kalman Filter to Chemical Sensors for Downstream Machine Learning." *IEEE Sensors Journal, Volume 18, No. 13, July 1, 2018*

Published Non-First Author Papers

Huimin Ren, Yun Yue, Chong Zhou, Randy C. Paffenroth, Yanhua Li, and **Matthew L. Weiss**. "Robust Variational Autoencoders: Generating Noise-Free Images from Corrupted Images." *AdvML '20: Workshop on Adversarial Learning Methods for Machine Learning and Data Mining, San Diego, CA, USA 2020*

Michael S. Wiederoder, **Matthew L. Weiss**, Bora Yoon, Randy C. Paffenroth, Shannon K. McGraw, and Joshua R. Uzarski. "Impact of Graphene Nanoplatelet Concentration and Film Thickness on Vapor Detection for Polymer Based Chemiresistive Sensors." *Current Applied Physics, Volume 19, Issue 9, September 2019*

Michael S. Wiederoder, Eric C. Nallon, **Matthew L. Weiss**, Shannon K. McGraw, Vincent P. Schnee, Collin J. Bright, Michael P. Polcha, Randy C. Paffenroth, and Joshua R. Uzarski. "Graphene Nanoplatelet-Polymer Chemiresistive Sensor Arrays for the Detection and Discrimination of Chemical Warfare Agent Simulants." *ACS Sensors 2017*

Papers in Submission

Kirty Vedula, **Matthew L. Weiss**, Randy C. Paffenroth, Joshua R. Uzarski, D. Richard Brown III. "Maneuvering Target Tracking Using the Autoencoder-Interacting Multiple Model Kalman Filter." *Submitted to the 54th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA, 2020*

Working Papers

Matthew L. Weiss, Randy C. Paffenroth, and Joshua R. Uzarski. "Measurement Noise Covariance Analysis for the Autoencoder-Kalman Filter."

Funding and Technical Acknowledgments

Funding for this dissertation was provided by In-house Laboratory Independent Research (ILIR) and Section 219 Innovation Funding at the U.S. Army Combat Capabilities Development Command Soldier Center in Natick, MA. Released for UNLIMITED DISTRIBUTION, PAO# U20-2220.

Based upon our research, subsequent U.S. government grants totaling $649,299 for fiscal year 2020 were secured to continue the work herein, with up to two additional years at similar amounts.

Results in this dissertation were obtained in part using a high-performance computing system acquired through NSF MRI grant DMS-1337943 to WPI.

Executive Summary

The story we tell in this dissertation is one of leveraging the power of deep learning and mathematics in the service of developing a generalized time series state estimation system. In particular, we utilize a deep autoencoder [11], with training inspired by Hilbert Space representations of functions [21, 31], to enhance the filtering capabilities of the Kalman Filter [3, 7, 29, 54, 57, 61, 65].

Specifically, we combine a deep autoencoder and Kalman Filter to produce a hybrid filtering algorithm called the Autoencoder-Kalman Filter (AEKF). The thinking behind this merger is to combine the computational power of deep learning with a theoretically well-established filtering method. The resulting AEKF not only shows improved state estimation over the standard Kalman Filter and a Long Short-Term Memory (LSTM) [23] recurrent neural network (RNN), but also solves the problem of how the Kalman Filter handles outlier measurements. This is demonstrated in Figure 1, which shows an AEKF state estimate of a sinusoidal curve with Cauchy noise.

The training of the AEKF is somewhat non-standard and relies on a technique known as domain randomization [41, 52, 58]. The overarching purpose of domain randomization is to train deep learning models completely in simulation which then generalize to real-world data without any further training or parameter tuning. Instead of training a neural network to learn parameters from a fixed training set and then generalize to a testing set, we utilize domain randomization to train a neural network to learn parameters within a specified range of values, $[a, b]$, in simulation. In some sense, this process can be thought of as learning a covering of a subspace in a Hilbert Space, an idea that will be elaborated on in Section 3.2. As an example, in the context of state estimation, we train the AEKF to learn how to filter

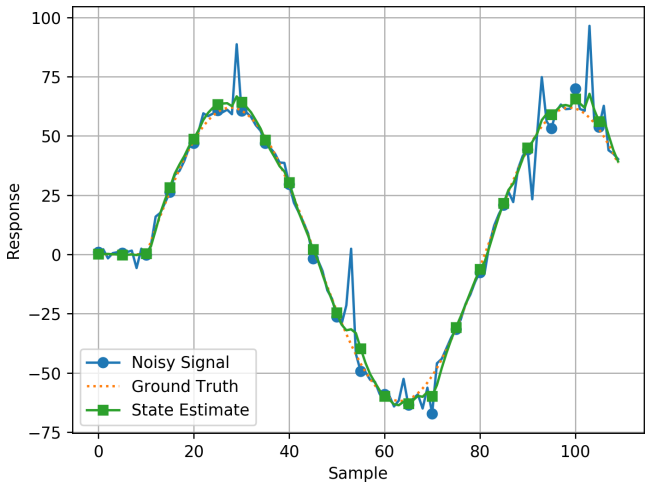


Figure 1: AEKF filtering of a test set sine function with Cauchy noise. Note the state estimate, represented by the solid green line, stays close to the ground truth even in the presence of outliers. This behavior is difficult to achieve with a standard Kalman Filter, which is more easily misled by outlier data.

a family of exponential curves with Gaussian noise. Each training epoch, a random exponential curve, $f(x) = e^{\alpha x}$, is generated by uniformly sampling $\alpha \in [a, b]$. This smooth curve serves as the ground truth or true state in the context of state estimation. A random draw of Gaussian noise is then added to the ground truth curve. The smooth ground truth with added noise serves as a noisy training sample. Each epoch, the AEKF is given a new noisy sample and trained to estimate the ground truth by minimizing the difference between the AEKF's output and the ground truth in the AEKF's cost function. If trained successfully, the AEKF has not generalized to a specific dataset but generalized to a family of functions within the domain of x, the range of α and noise distribution parameters used in training. *Due to the use of domain randomization, during training the AEKF almost certainly never sees the same ground truth and noise sample twice.* Trained in this way, the AEKF is truly learning to generalize to the family of functions it was trained on.

Working towards the goal of a generalized time series state estimation system,

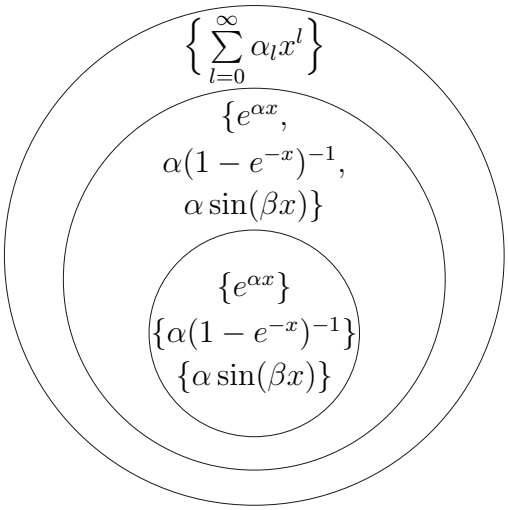


Figure 2: Figure depicting our progressive domain randomization scheme. The concentric circles represent a progressively larger subregion of a Hilbert Space, each including the smaller circles. The smallest circle represents all exponential, sigmoidal, and sinusoidal function families individually while the middle circle represents a single family consisting of all exponential, sigmoidal, and sinusoidal functions. The outermost circle represents the most general class of functions, Taylor Polynomials.

we train a single AEKF to filter progressively larger families of functions, each of which contains the previous family as a subset. This is shown visually in Figure 2, where each circle represents a higher level of generality. The smallest circle represents training three different AEKF models to filter exponential, sigmoidal, and sinusoidal families individually. The next larger circle represents a single AEKF that is trained to filter all three of these families. Lastly, wanting to achieve a higher level of generalization, we train an AEKF to filter truncated Taylor polynomials as shown in the largest circle.

Table 1 shows the results of training two AEKF models, five LSTM models, and two Kalman Filter models on exponential, sigmoidal and sinusoidal curves with Cauchy noise individually (corresponding to the innermost circle in Figure 2) and then testing each model on the same family it was trained upon. Taking the AEKF

Table 1: MSE and ratio results for single curve family models with Cauchy noise. This result demonstrates the AEKF achieves a significantly lower MSE for all three curve families on a noise distribution that is very challenging for the traditional Kalman Filter. Bold values indicate the smallest MSE in each column. Note, NCV and NCA refer to specific Kalman Filter models in the AEKF and standard Kalman Filter and the number following each LSTM model is the sequence length used for that model.

Model	Exp. MSE	Exp. Ratio	Sig. MSE	Sig. Ratio	Sine MSE	Sine Ratio
AEKF NCV	**0.39**	**1.00**	1.16	1.00	3.56	1.00
AEKF NCA	0.43	1.08	**0.89**	**0.77**	**2.96**	**0.83**
LSTM 1	35.90	90.99	45.45	39.16	82.12	23.06
LSTM 10	1.35	3.42	5.72	4.93	21.18	5.95
LSTM 15	5.15	13.04	45.43	39.14	109.56	30.77
LSTM 20	10.66	27.01	183.51	158.11	644.53	180.99
LSTM 25	63.08	159.89	509.68	439.12	835.96	234.74
KF NCV	99.26	251.58	953.21	821.25	2979.70	836.72
KF NCA	212.33	538.18	3541.22	3051.00	13427.75	3770.61

NCV row as an example, the Exp. MSE column is the exponential curve test set state estimation MSE for the AEKF NCV model trained on exponential curves with Cauchy noise, where the MSE is computed by comparing the AEKF NCV estimate against the ground truth. However, the ground truth is only used to compute the MSE and did not affect the AEKF NCV state estimation. The Exp. Ratio column is the ratio of each model's MSE and the AEKF NCV MSE. The next four columns follow similarly for sigmoidal and sinusoidal curve families. The top performing models for each of the three curve families are shown in bold. These results show that either of the two AEKF models outperformed both the LSTM and Kalman Filter models on all three curve families. Note, NCV and NCA refer to specific Kalman Filter models in the AEKF and standard Kalman Filter and the number following each LSTM model is the sequence length used for that model.

Given the AEKF performance in Table 1, we note one of our primary results is the ability of the AEKF to mitigate the presence of outliers when trained with

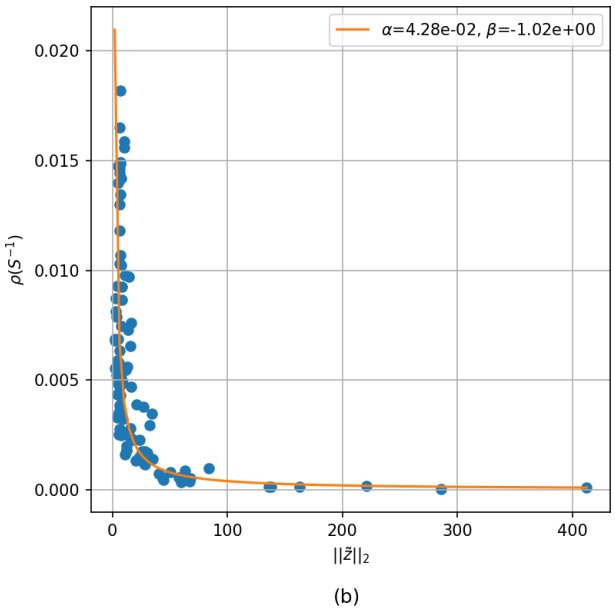


Figure 3: Single test set trial sample plot of $\rho(\mathbf{S}_k^{-1})$ vs. $\|\tilde{\mathbf{z}}_k\|_2$ for the AEKF. The related estimated parameters appear in the legend. This figure is consistent with our hypothesis that $\rho(\mathbf{S}_k^{-1})$ and $\|\tilde{\mathbf{z}}_k\|_2$ share an inverse relationship.

domain randomization using Cauchy noise. The Kalman Filter is known to be optimal if, among other Gaussian assumptions, the noise obscuring the state being estimated is Gaussian. However, noise distributions with large tails, such as the Cauchy distribution, present a problem for the Kalman Filter. As the Kalman Filter's state estimate takes into consideration the measurements it receives, it must be able to distinguish reliable and unreliable measurements. This role is handled by the Kalman Filter's measurement noise covariance matrix $\mathbf{R}$. However, a single measurement noise covariance matrix for a distribution such as the Cauchy distribution is problematic as the long tails allow for values with large variance. The primary reason the AEKF is able to perform well in the presence of these outliers is it learns not a single measurement noise covariance matrix $\mathbf{R}$, but a sequence of them, $\{\mathbf{R}_k\}_{k=1}^N$, one for each measurement. This point-wise measurement noise covariance allows the AEKF to "tell" the Kalman Filter how reliable each measure-

ment in a time series is.

After we present empirical evidence that the AEKF mitigates the effect of outliers in a variety of contexts, in Section 4.5 we prove a theorem which shows the Kalman Filter's outlier mitigation capacity is upper bounded by the largest singular value and eigenvalues of three Kalman Filter matrices, $\mathbf{H}_k^\intercal$, $\mathbf{P}_{k|k\text{-}1}$, and $\mathbf{S}_k^{-1}$ respectively. Specifically,

$$\|\mathbf{K}_k\tilde{\mathbf{z}}_k\|_2 \leq \rho(\mathbf{P}_{k|k\text{-}1})\sigma_1(\mathbf{H}_k^\intercal)\rho(\mathbf{S}_k^{-1})\,\|\tilde{\mathbf{z}}_k\|_2 \tag{1}$$

from which we hypothesize that $\|\tilde{\mathbf{z}}_k\|_2$ and $\rho(\mathbf{S}_k^{-1})$, the largest eigenvalue in $\mathbf{S}_k^{-1}$, share an inverse relationship. *Thus, if the AEKF's behavior is consistent with (1), it is learning the sequence of measurement noise covariance matrices* $\{\mathbf{R}_k\}_{k=1}^N$ *in such at way that large outliers are ignored by the Kalman Filter.* This hypothesis is then confirmed experimentally as shown in Figure 3. As a result, in this dissertation we demonstrate how well-understood mathematical models can inform neural network design.

Road Map

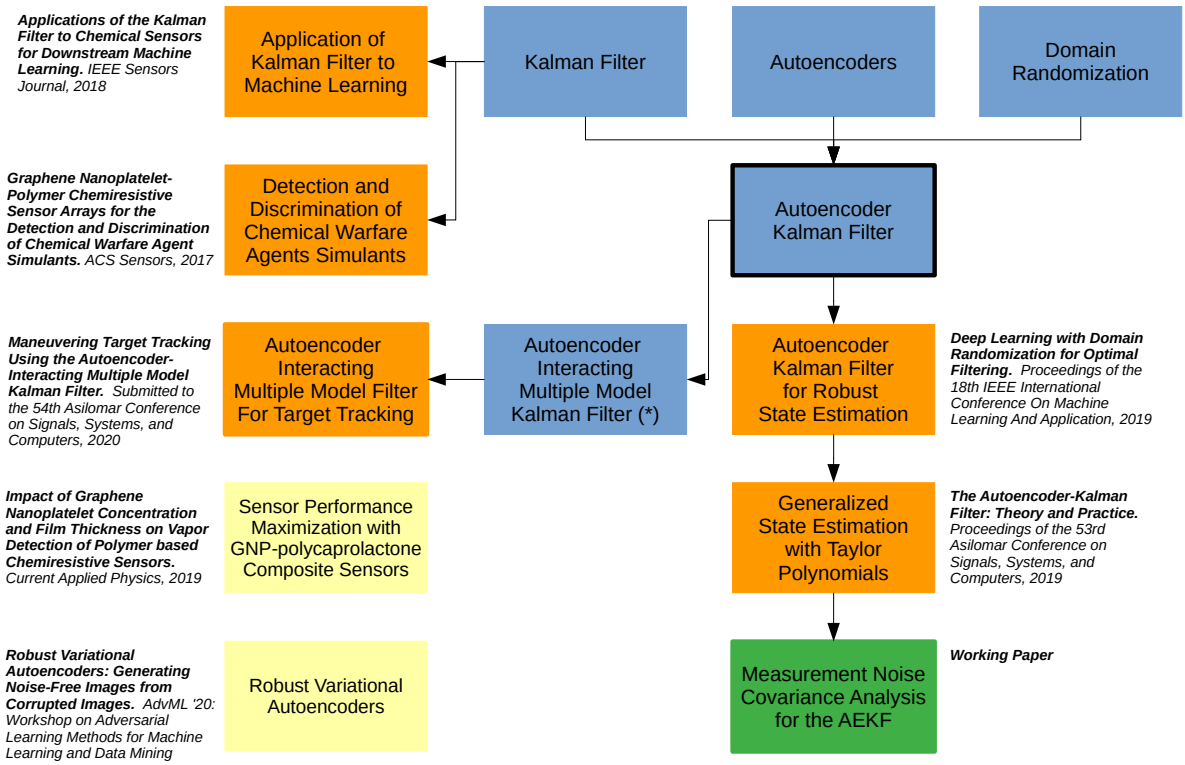


Figure 4: Research Road Map. This figure is a summary of our research over the past four years and can be broken down into algorithms and publications, with the blue boxes indicating algorithms and the orange their application in publications. The black band around the Autoencoder-Kalman Filter box indicates this algorithm is our primary algorithmic contribution and the green box indicates a working paper based upon a mathematical analysis of the AEKF. The papers in yellow did not utilize the algorithms in blue. In summary, based on this dissertation, we have three published first author papers, three published non-first author papers written in collaboration with CCDC-SC scientists and members of the WPI community, a co-first author paper in submission, and one working first author paper.

Contents

1 Introduction

Filtering is the process of recovering a signal, $\mathbf{x}(t)$, from noisy measurements, $\mathbf{z}(t)$. In general, the filtering problem attempts to recover, estimate, or determine some information about a state $\mathbf{x}(t)$ given values of a measurement process $\mathbf{z}(t)$, up to and including time t, where the state is assumed to be obscured by noise present in $\mathbf{z}(t)$. For example, $\mathbf{z}(t)$ may represent noisy time series measurements from a sensor and $\mathbf{x}(t)$ may represent the signal assumed to be the true sensor response obscured by noise in the sensor [3]. Filtering can also be distinguished from smoothing and prediction. In the case of smoothing, an estimate of $\mathbf{x}(t)$ is made using measurements beyond time t. Prediction attempts to estimate $\mathbf{x}(t + \delta t)$, for $\delta t > 0$, using measurements up to and including time t [3].

The problem of separating signal from noise has a long history and these attempts can broadly be broken into the frequency approach and statistical approach. The frequency approach assumes the signal and noise lie in different frequency bands with a certain amount of overlap. Conversely, the statistical approach assumes there are certain shared statistical properties of the signal and noise. Initial contributions to statistical filtering were made by Norbert Weiner and Andrey Kolmogorov [3]. However, these methods assumed the statistical distributions involved were stationary, which imposed a limitation that was subsequently addressed by Rudolf Kalman. In [29] Kalman developed the initial formulation of what came to be known as the Kalman Filter, which did not rely on the above stationary assumption. This initial development, and subsequent additions, led to the Kalman Filter as a mainstay in the communications and control communities. For more on the history of filtering and related references see [3].

One of the Kalman Filter's advantages is the simplicity and elegance of its the-

oretical foundation, where the Kalman Filter can be shown to be the optimal filter under specific conditions. However this comes with a price. The fact that the Kalman Filter can be solved analytically is a result of it belonging to the class of linear filters. That is, *the Kalman Filter can be shown to be the optimal conditional minimum variance estimator among all filters which are an affine transformation of the input measurements when the random variables involved are assumed to be Gaussian and the Kalman Filter's dynamical model is correct*. One particular consequence of the Gaussian assumption is that the measurement noise present in $\mathbf{z}(t)$ is assumed to be Gaussian. As many applications of the Kalman Filter involve estimating processes with non-linear dynamics and non-Gaussian noise, these assumptions present a certain limitation. Thus, subsequent variants of the Kalman Filter, such as the Extended Kalman Filter [3], were developed to address these limitations. Continuing in this tradition, the work presented here can be seen as an extension of the Kalman Filter to domains where the traditional optimality conditions are not necessarily met.

The primary contribution of this research is the Autoencoder-Kalman Filter (AEKF), a novel deep learning-Kalman Filter hybrid algorithm which addresses the difficulty posed by the Kalman Filter when the statistical distribution of the input measurements, $\{\mathbf{z}_k\}_{k=1}^N$, and/or the covariance of these inputs, $\{\mathbf{R}_k\}_{k=1}^N$, is unknown, difficult to estimate, or does not exist. Although there exists previous work addressing Kalman Filter performance under sub-optimal conditions [50], the application of deep learning to filtering [63], and the merger of deep learning and the Kalman Filter [19, 34, 45], to our knowledge, the AEKF distinguishes itself by incorporating all of the following, many of which are original:

- The AEKF addresses the Kalman Filter measurement noise covariance esti-

mation problem by reformulating the problem to one of training a neural network.

- The derivation of a novel algorithm where the encoder portion of an autoencoder is used to map noisy time series measurements to a learned measurement sequence $\{\mathbf{z}_k\}_{k=1}^N$ and a learned measurement noise covariance sequence $\{\mathbf{R}_k\}_{k=1}^N$, which are used as inputs to the Kalman Filter, thereby creating a system with the flexibility of deep learning, and the principled theoretical foundation of the Kalman Filter.

- The demonstration that the merger of an autoencoder and Kalman Filter allows the AEKF to be effectively used even in the context where the theoretical conditions for optimality of the Kalman Filter are not met.

- The leveraging of *domain randomization*, which allows training with simulated data, by-passing the practical difficulty of securing enough data to sufficiently train a deep learning model and avoiding overfitting.

- Numerical demonstrations of the AEKF which show, in the majority of tested cases, it outperforms both a traditional Kalman Filter and state-of-the-art LSTM.

- The restriction of the Kalman Filter's dynamical model to a linear model and the learned Kalman Filter parameters to measurements $\{\mathbf{z}_k\}_{k=1}^N$ and the measurement noise covariance $\{\mathbf{R}_k\}_{k=1}^N$. This allows us to perform a mathematical analysis in which we prove a theorem explaining how the AEKF mitigates the effects of outliers.

In [25, 34, 45, 63] deep learning technology is applied in some way to learn an

aspect of a Kalman Filter's dynamical model. [10] is similar to our work conceptually in that deep learning is used to estimate $\{\mathbf{R}_k\}_{k=1}^N$. However, three separate LSTM modules are used to estimate $\{\mathbf{F}_k\}_{k=1}^N$, $\{\mathbf{Q}_k\}_{k=1}^N$ and $\{\mathbf{R}_k\}_{k=1}^N$ respectively. In [15], the authors apply the combination of a Kalman Filter and variational autoencoder (VAE) [33] to missing data imputation, which requires the combination of prediction and inference. In [19], the authors propose the backprop Kalman Filter (BPKF), which uses a feedforward neural network to learn an $\{\mathbf{R}_k\}_{k=1}^N$ and $\{\mathbf{z}_k\}_{k=1}^N$ which then passes these to a Kalman Filter. In their first experiment, the added noise is Gaussian and the trajectories of the tracked objects have linear-Gaussian dynamics. However, the AEKF developed herein was tested on both Gaussian and non-Gaussian noise. In the second experiment the Extended Kalman Filter was used in place of the standard Kalman Filter. Lastly, while the BPKF passes a linear transformation of the Kalman Filter output directly to its objective function, the AEKF places the decoder layer between the Kalman Filter's output and the final AEKF output, thus the AEKF allows for a non-linear mapping between the Kalman Filter's output and the original measurement space.

Following this introduction, Section 2 presents the theoretical background of the Kalman Filter (2.1), the Interacting Multiple Model Kalman Filter (IMMKF) (2.2), autoencoders (2.3), Long Short-Term Memory (LSTM) recurrent neural networks (RNN) (2.4), the Autoencoder-Kalman Filter (2.5) and the Autoencoder-Interacting Multiple Model Kalman Filter (AEIMMKF) (2.6). This is followed by Section 3 on training, which covers Hilbert Space representation of functions (3.1) and domain randomization (3.2) as they relate to the training of neural networks.

The application and analysis of the algorithms and training methods presented in Sections 2 and 3 begins in Section 4.1, with the application of the Kalman Filter

to feature engineering for accurate and early chemical discrimination. In this work, using noisy chemical sensor time series data, we demonstrate that the accurate and early detection of non-mixture chemical warfare agent simulants and obscurants is improved when the Kalman Filter is used for feature generation.

In Sections 4.2, 4.3, and 4.4, having seen the value of the Kalman Filter in the context of feature engineering and knowing its limitations, we apply the AEKF to filtering problems in regimes where the Kalman Filter is known to be non-optimal. In Section 4.2 we train the AEKF with domain randomization on a family of sigmoidal curves, where we introduce the notion of sequence length for an autoencoder. Here the AEKF achieves lower test set MSE as a function of sequence length. However, in Section 4.2 each measurement noise covariance matrix in $\{\mathbf{R}_k\}_{k=1}^N$ was restricted to $\mathbb{R}^{1\times 1}$. Once we understood how to ensure the AEKF learns a non-singular measurement noise covariance matrix $\mathbf{R}_k \in \mathbb{R}^{n\times n}$ for each entry in $\{\mathbf{R}_k\}_{k=1}^N$, we were no longer restricted to $\mathbb{R}^{1\times 1}$. As a result, in Section 4.3, via domain randomization, the AEKF is trained such that each learned measurement noise covariance matrix is in $\mathbb{R}^{16\times 16}$. Furthermore, the AEKF is no longer restricted to filtering sigmoidal functions, as three separate AEKFs are trained to filter sigmoidal, exponential and sinusoidal curve families, with added Gaussian, bimodal, and Cauchy noise. The three AEKFs correspond to the inner circle of Figure 2. Additionally, in Section 4.3 we train a *single AEKF to learn all three families of functions: sigmoidal, exponential and sinusoidal*, corresponding to the second largest circle in Figure 2. These results indicate a single AEKF can generalize to more than one family of functions and, in most cases, the AEKF obtains better test set MSE than a standard Kalman Filter and LSTM. Based on these results, in Section 4.4 we again train the AEKF with domain randomization to filter a more general class of func-

tions based upon truncated Taylor Polynomials:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_p x^p$$

where each coefficient a_i for $i = \{0, 1, \ldots, p\}$ and the polynomial order p are sampled randomly. As in Section 4.3, Gaussian, bimodal, and Cauchy noise are added to the ground truth truncated Taylor polynomials. The results in this section demonstrate the state estimation capabilities of the AEKF successfully generalize to a larger family of functions than simply exponential, sigmoidal, and sinusoidal.

In the process of performing the tests in Section 4.4, we discovered the AEKF was able to mitigate outliers more effectively than a standard Kalman Filter and LSTM. Subsequent investigation revealed the AEKF had learned the sequence $\{\mathbf{R}_k\}_{k=1}^N$ in such a way as to mitigate the effect of outlier measurements. In Section 4.5, applying matrix analysis to the Kalman Filter, we derive a theorem which explains this outlier mitigation behavior in terms of the eigenvalues of the learned measurement noise covariance matrices $\{\mathbf{R}_k\}_{k=1}^N$. This allows us to hypothesize an empirically testable relation between these learned eigenvalues and the norm of the Kalman Filter's innovation. This hypothesis is then confirmed in subsequent tests. Thus, while a complete understand of "how" the AEKF learns $\{\mathbf{R}_k\}_{k=1}^N$ is still unclear, the results in Section 4.5 provide a theoretical understanding of "what" the AEKF is doing to mitigate outliers.

Building on the state estimation capabilities of the AEKF, in Section 4.6 we extended the AEKF to a new algorithm, the AEIMMKF. Similar to the AEKF, the AEIMMKF places an Interacting Multiple Model Kalman Filter in the latent layer of a deep autoencoder. The IMMKF is similar to a standard Kalman Filter except now there is a bank of M Kalman Filters, where the output of the IMMKF is

a statistical weighting of the M Kalman Filters. In the tracking community, it is common practice to replace a standard Kalman Filter with the IMMKF, so the step from the AEKF to AEIMMKF is a natural one. Training the AEIMMKF on simulated single-turn flight paths with added Gaussian and Cauchy noise, we show the AEIMMKF achieves superior state estimation compared with a standard Kalman Filter, IMMKF, and AEKF.

Lastly, having seen a connection between Kalman Filter state estimation and the Hilbert Space representation of functions from Section 3.1, we investigate the theoretically well-known properties of Hilbert Spaces as they relate to state estimation in Section 4.7. Here the AEKF is replaced by the Hilbert Space Filter, which is a single feedforward neural network that learns a smooth, global estimate of the ground truth instead of a point-wise estimate as the AEKF does. The ideas and results in this section are preliminary and present a promising area of future research.

2 Algorithms

2.1 Kalman Filter

The Kalman Filter [3, 7, 29, 54, 57, 61, 65] estimates the state of a system at time k, $\mathbf{x}_k$, using all measurements, $\mathbf{z}_k$, up to and including time k and is based on the following linear model

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k\text{-}1} + \mathbf{w}_k \tag{2}$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{3}$$

where, for fixed k, $\mathbf{F}_k$ is a linear mapping $\mathbf{F}_k : \mathbb{R}^n \mapsto \mathbb{R}^n$ from the state space to itself which projects the state forward from time k-1 to time k, $\mathbf{x}_{k\text{-}1}$ is the state at time k-1, and $\mathbf{w}_k$ is the process noise at time k. Thus, (2) models the temporal transition between states as the sum of a linear transformation of the previous state, $\mathbf{F}_k \mathbf{x}_{k\text{-}1}$, and an added "noise" or random term, $\mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$ which is assumed to be a zero mean Gaussian process with covariance $\mathbf{Q}_k$. In (3), $\mathbf{H}_k$ is defined as a linear mapping $\mathbf{H}_k : \mathbb{R}^n \mapsto \mathbb{R}^m$ from the state space to the measurement space, $\mathbf{z}_k$ is the actual measurement or observation at time k, and $\mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k)$ is the noise present in the measurements at time k which obscures the state and is assumed to be a zero mean Gaussian process with covariance $\mathbf{R}_k$.

To summarize, equation (2) defines how the state evolves in time and equation (3) models the relationship between the actual state and observed measurements. Based on this model, the goal of the Kalman Filter is to find the optimal estimate of the sequence $\{\mathbf{x}_k\}_{k=1}^N$ given sequences $\{\mathbf{H}_k\}_{k=1}^N$, $\{\mathbf{F}_k\}_{k=1}^N$, $\{\mathbf{z}_k\}_{k=1}^N$, $\{\mathbf{R}_k\}_{k=1}^N$ and $\{\mathbf{Q}_k\}_{k=1}^N$ for $k = \{1, 2, \ldots, N\}$. For reference the four vectors at time k and their

associated names are

- $\mathbf{x}_k \in \mathbb{R}^n$ state estimate vector
- $\mathbf{z}_k \in \mathbb{R}^m$ measurement vector
- $\mathbf{w}_k \in \mathbb{R}^n$ process noise vector
- $\mathbf{v}_k \in \mathbb{R}^m$ measurement noise vector

and the related matrices at time k are similarly defined as

- $\mathbf{F}_k \in \mathbb{R}^{n \times n}$ state transition matrix
- $\mathbf{H}_k \in \mathbb{R}^{m \times n}$ observation matrix
- $\mathbf{Q}_k \in \mathbb{R}^{n \times n}$ process noise covariance matrix
- $\mathbf{R}_k \in \mathbb{R}^{m \times m}$ measurement noise covariance matrix

For a given k, each iteration of the Kalman Filter begins with an estimate of the state $\mathbf{x}_k$, the *a priori* state estimate, based the expected value of (2). The final, or *a posteriori*, estimate of $\mathbf{x}_k$ is then an affine function of the *a priori* state estimate and actual measurement at time k, along with $\mathbf{H}_k$ and $\mathbf{K}_k$. This process is depicted in

Figure 5 and mathematically represented by the series of equations

$$\hat{\mathbf{x}}_{k|k\text{-}1} = \mathbf{F}_k \hat{\mathbf{x}}_{k\text{-}1|k\text{-}1} \tag{4}$$

$$\mathbf{P}_{k|k\text{-}1} = \mathbf{F}_k \mathbf{P}_{k\text{-}1|k\text{-}1} \mathbf{F}_k^{\mathsf{T}} + \mathbf{Q}_k \tag{5}$$

$$\tilde{\mathbf{z}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k\text{-}1} \tag{6}$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k\text{-}1} \mathbf{H}_k^{\mathsf{T}} + \mathbf{R}_k \tag{7}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k\text{-}1} \mathbf{H}_k^{\mathsf{T}} \mathbf{S}_k^{-1} \tag{8}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k\text{-}1} + \mathbf{K}_k \tilde{\mathbf{z}}_k \tag{9}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k\text{-}1} \tag{10}$$

where equations (4) through (10) represent a single iteration of the Kalman Filter and can be broken down into two parts: prediction [(4) - (5)] and update [(6) - (10)].

Equation (4) represents the *a priori* state estimate of $\mathbf{x}_k$ at time k, where the subscript $k|k$-1 indicates an estimate of the state at time k given all data up to and including time k-1. Since the Kalman Filter is estimating the state at time k before having seen the actual measurement at time k, this estimate is referred to as the *a priori* estimate. Equation (5) represents the covariance of the estimate in (4) and is referred to as the *a priori* estimate covariance. Equations (6) and (7) represent the innovation and innovation covariance respectively. The innovation is simply the difference between the actual measurement, $\mathbf{z}_k$, and the *a priori* estimate, $\hat{\mathbf{x}}_{k|k\text{-}1}$, mapped into the measurement space via $\mathbf{H}_k$. Equation (8) is the Kalman Gain, which weights the innovation's contribution to the final state estimate. Lastly, (9) and (10) are the *a posteriori* state estimate and *a posteriori* estimate covariance respectively. The distinction between the *a priori* and *a posteriori* estimates and covariances rests on whether the actual measurement, $\mathbf{z}_k$, has been considered by the

Kalman Filter when estimating the state at time k.

2.1.1 Kalman Filter Optimality Conditions

The presentation of the Kalman Filter optimality conditions in this section and Section 2.1.2 motivates our merger of deep learning and the Kalman Filter in the AEKF. To that end, here we discuss the regime where the Kalman Filter is the optimal filter among a certain class of filters. Furthermore, and more importantly for the purposes of this dissertation, through understanding the Kalman Filter's limitations, we indicate where its performance may be enhanced by deep learning.

Traditionally the derivation of the Kalman Filter begins by assuming the random processes in (2) and (3) are Gaussian. Here we take a different approach and begin by showing the Kalman Filter is the optimal filter among a specific class of filters. We then show, in the Gaussian case mentioned above, the Kalman Filter is the optimal filter, in the sense of conditional minimum variance estimation, among all filters.

The essence of the problem the Kalman Filter addresses is embodied in (3). Since the true state is obscured by the noise, only the noisy measurements are observed. Thus, the question becomes: what can be said in a principled manner about the true state given only knowledge of the noisy measurements [29]? More precisely, assume a random vector

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{z}_0 \\ \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_N \end{bmatrix} \tag{11}$$

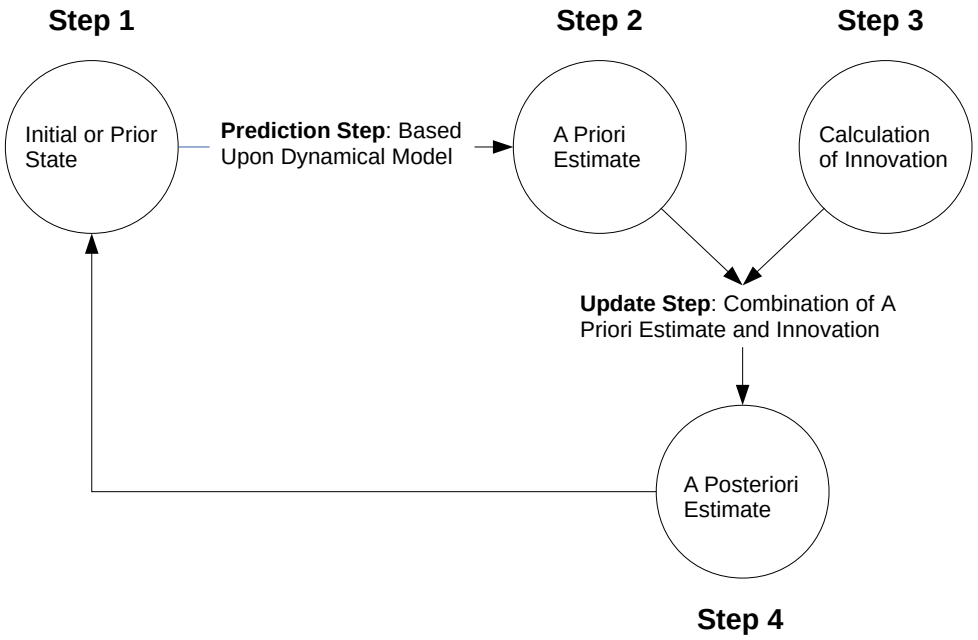


Figure 5: Conceptual diagram depicting a single iteration of the Kalman Filter. Step 1 is the initial input at time zero or output of the previous iteration. Based on this, the Kalman Filter computes the *a priori* estimate at step 2. Using the *a priori* estimate and the measurement, the innovation is calculated at step 3. These are then combined to form the *a posteriori* output in step 4, where the relative weighting between the *a priori* estimate and the innovation is determined by the Kalman Gain (8). The *a posteriori* output is then used as the input to the next iteration at step 1. This process is repeated iteratively until all measurements are exhausted.

where $\mathbf{x}$ and $\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_N$ are considered random variables and thus take on values in a random fashion. If we require that they all vary for the same underlying reasons, whatever that may be, it seems plausible that knowledge of $\{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_N\}$ may allow for a principled estimation of $\mathbf{x}$. Put another way, if all the circumstances affecting $\{\mathbf{x}, \mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_N\}$ are in the same sample space Ω, and thus $\{\mathbf{x}, \mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_N\}$ are functions on Ω, any covariance between $\mathbf{x}$ and $\{\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_N\}$ may, in principle, be utilized to estimate $\mathbf{x}$. This is the idea behind the Kalman Filter [3, 7].

2.1.2 Best Linear Minimum Variance Estimate

The approach here, which follows the derivation given in [3], is to assume the estimate of the true state, $\mathbf{x}$, is a function of the observations, represented here by

$$\hat{\mathbf{x}} = f(\mathbf{z}) \tag{12}$$

where $f(\cdot)$ in (12) is not restricted to a specific functional form, such as linear or non-linear. In the case where $f(\cdot)$ is an affine function of $\mathbf{z}$, the Kalman Filter is the best estimator of any affine filter when the noise processes $\{\mathbf{v}_k\}_{k=1}^N$ and $\{\mathbf{w}_k\}_{k=1}^N$ are mutually uncorrelated, zero-mean, and white and further uncorrelated with the initial state estimate $\mathbf{x}_0$. Here we also see a hint as to why the Kalman Filter is referred to as a linear filter, as the estimate $\hat{\mathbf{x}}$ is defined as a linear transformation of the measurements $\mathbf{z}$ plus an additive term independent of the measurements. Thus, while technically the Kalman Filter is the best *affine* minimum variance estimator (BAMVE), the term best *linear* minimum variance estimator (BLMVE) is most commonly used in practice and we follow this convention.

More formally, rewriting (12) as an affine function of $\mathbf{z}$

$$\hat{\mathbf{x}} = \mathbf{A}\mathbf{z} + \mathbf{b} \tag{13}$$

where $\mathbf{A}$ is a matrix mapping from the measurement space to signal space and $\mathbf{b}$ is a vector in the signal space. The goal is then to solve

$$\arg\min_{\mathbf{A},\mathbf{b}} \mathbb{E}\big[\, \|\mathbf{x} - \hat{\mathbf{x}}\|^2 \,\big] \tag{14}$$

which upon substitution of (13) becomes

$$\arg\min_{\mathbf{A},\mathbf{b}} \mathbb{E}\big[\, \|\mathbf{x} - \mathbf{A}\mathbf{z} - \mathbf{b}\|^2 \,\big] \tag{15}$$

Given random vectors $\mathbf{x}$ and $\mathbf{z}$ with means $\bar{\mathbf{x}}$ and $\bar{\mathbf{z}}$ and covariances

$$\begin{aligned}
\mathrm{Cov}\big(\mathbf{x}, \mathbf{x}\big) &= \mathbf{\Sigma}^{xx} \\
\mathrm{Cov}\big(\mathbf{x}, \mathbf{z}\big) &= \mathbf{\Sigma}^{xz} \\
\mathrm{Cov}\big(\mathbf{z}, \mathbf{x}\big) &= \mathbf{\Sigma}^{zx} \\
\mathrm{Cov}\big(\mathbf{z}, \mathbf{z}\big) &= \mathbf{\Sigma}^{zz}
\end{aligned}$$

it can then be shown the solution to (15) is given by

$$\mathbf{A} = (\mathbf{\Sigma}^{xz}(\mathbf{\Sigma}^{zz})^{-1}) \tag{16}$$

$$\mathbf{b} = \bar{\mathbf{x}} - \mathbf{A}\bar{\mathbf{z}} \tag{17}$$

substituting (16) and (17) into (13) gives

$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + (\boldsymbol{\Sigma}^{xz}(\boldsymbol{\Sigma}^{zz})^{-1})(\mathbf{z} - \bar{\mathbf{z}}) \tag{18}$$

which we will see corresponds to the Kalman Filter's *a posteriori* estimate if we make the follow identifications

$$\begin{aligned}
\bar{\mathbf{x}} &\triangleq \hat{\mathbf{x}}_{k|k\text{-}1} \\
\bar{\mathbf{z}} &\triangleq \mathbf{H}_k\hat{\mathbf{x}}_{k|k\text{-}1} \\
\boldsymbol{\Sigma}^{xz} &\triangleq \mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^{\mathsf{T}} \\
\boldsymbol{\Sigma}^{zz} &\triangleq \mathbf{S}_k
\end{aligned}$$

Up to this point nothing specific to the Kalman Filter has been used. Now we consider the form (18) takes when the linear model in (2) and (3) is assumed. After some lengthy calculations and reintroducing the standard Kalman Filter subscript notation, we arrive at formulas for the recursive estimation of the *a posteriori* state estimate and associated covariance given by

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k\text{-}1} + (\mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^{\mathsf{T}}\mathbf{S}_k^{-1})(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k\text{-}1}) \tag{19}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k\text{-}1} \tag{20}$$

A complete derivation of the above is given in appendix A.1.

We now show how the Kalman Filter is the best conditional minimum variance estimator (BCMVE), for any class of filters defined by f in (12), when a specific assumption is made about the joint distribution between $\mathbf{x}$ and $\mathbf{z}$. Namely, we assume $\mathbf{x}$ and $\mathbf{z}$ are jointly Gaussian, along with the assumptions regarding variables

being uncorrelated above. First, we state a well-known theorem from probability. Suppose we want to estimate the value taken by a random variable $\mathbf{x}$, represented by $\hat{\mathbf{x}}$, given the value taken by another random variable $\mathbf{z}$. In this case, the conditional minimum variance estimate of $\mathbf{x}$ given $\mathbf{z} = z$ is defined as $\hat{\mathbf{x}}$ such that

$$\mathbb{E}\big[\, \|\mathbf{x} - \hat{\mathbf{x}}\|^2 \,|\mathbf{z} = z\big] \leq \mathbb{E}\big[\, \|\mathbf{x} - \mathbf{y}\|^2 \,|\mathbf{z} = z\big] \tag{21}$$

for all vectors $\mathbf{y}$. With this definition of the minimum variance estimate, given jointly distributed random variables $\mathbf{x}$ and $\mathbf{z}$ (regardless of the distribution) and letting $\mathbf{z}$ take on the value $\mathbf{z} = z$, then $\hat{\mathbf{x}}$ is uniquely defined as the conditional mean of $\mathbf{x}$ given $\mathbf{z} = z$. That is,

$$\hat{\mathbf{x}} = \mathbb{E}\big[\mathbf{x}|\mathbf{z} = z\big] \tag{22}$$

Note that equation (18) above represents the *best linear minimum variance estimate* and (a) is not an expectation and (b) is not necessarily the *best minimum variance estimate*. However, we can ask whether there is a statistical distribution where the conditional minimum variance estimate *is an affine function of the variable conditioned on.* If so, perhaps the form of this estimate is similar to (18). It turns out that the this is exactly the case. That is, if $\mathbf{x}$ and $\mathbf{z}$ are jointly distributed Gaussian random variables with means and and covariance as above, then $\mathbf{x}$, conditioned on the fact $\mathbf{z} = z$, is a Gaussian random variable with mean

$$\mathbb{E}\big[\mathbf{x}|\mathbf{z} = z\big] = \bar{\mathbf{x}} + (\boldsymbol{\Sigma}^{xz}(\boldsymbol{\Sigma}^{zz})^{-1})(\mathbf{z} - \bar{\mathbf{z}}) \tag{23}$$

which is the same as (18). This provides insight into why the Kalman Filter is con-

sidered optimal when the above Gaussian assumptions are made. Most generally, the Kalman Filter is an affine function of the measurements $\mathbf{z}$. However, in the Gaussian case, the BCMVE is the same as the BLMVE. Put another way, the answer to the question "what statistical distribution has a best conditional minimum variance of the form in (18)" is the Gaussian distribution.

2.1.3 Kalman Filter's Dynamical Model

The Kalman Filter's *a priori* estimate in (4) is a linear transformation independent of the measurements. For this reason it can be understood as representing the hidden state's dynamics or a model of how the true state transitions from $\mathbf{x}_{k\text{-}1}$ to $\mathbf{x}_k$. Some well-known dynamical models used in the Kalman Filter are the nearly constant velocity (NCV), the nearly constant acceleration (NCA), and the jerk model [4], whose matrix representations, $\mathbf{F}_k$, are modeled by (24), (25), and (26) respectively [4].

$$\begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \tag{24}$$

$$\begin{bmatrix} 1 & dt & \frac{1}{2}dt^2 \\ 0 & 1 & dt \\ 0 & 0 & 1 \end{bmatrix} \tag{25}$$

$$\begin{bmatrix} 1 & dt & \frac{1}{2}dt^2 & \frac{1}{6}dt^3 \\ 0 & 1 & dt & \frac{1}{2}dt^2 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{26}$$

Taking the NCA model as an example, since $\mathbf{F}_k \in \mathbb{R}^3$, the state vector at k-1 consists of the position, first derivative, and second derivative

$$\begin{bmatrix} \hat{\mathbf{x}}_{k\text{-}1|k\text{-}1} \\ \hat{\dot{\mathbf{x}}}_{k\text{-}1|k\text{-}1} \\ \hat{\ddot{\mathbf{x}}}_{k\text{-}1|k\text{-}1} \end{bmatrix} \tag{27}$$

Thus projecting forward in time via $\mathbf{F}_k$ gives

$$\begin{bmatrix} \hat{\mathbf{x}}_{k|k\text{-}1} \\ \hat{\dot{\mathbf{x}}}_{k|k\text{-}1} \\ \hat{\ddot{\mathbf{x}}}_{k|k\text{-}1} \end{bmatrix} = \begin{bmatrix} 1 & dt & \frac{1}{2}dt^2 \\ 0 & 1 & dt \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_{k\text{-}1|k\text{-}1} \\ \hat{\dot{\mathbf{x}}}_{k\text{-}1|k\text{-}1} \\ \hat{\ddot{\mathbf{x}}}_{k\text{-}1|k\text{-}1} \end{bmatrix} \tag{28}$$

$$= \begin{bmatrix} \hat{\mathbf{x}}_{k\text{-}1|k\text{-}1} + \hat{\dot{\mathbf{x}}}_{k\text{-}1|k\text{-}1}(dt) + \hat{\ddot{\mathbf{x}}}_{k\text{-}1|k\text{-}1}(\frac{1}{2}dt^2) \\ \hat{\dot{\mathbf{x}}}_{k\text{-}1|k\text{-}1} + \hat{\ddot{\mathbf{x}}}_{k\text{-}1|k\text{-}1}(dt) \\ \hat{\ddot{\mathbf{x}}}_{k\text{-}1|k\text{-}1} \end{bmatrix} \tag{29}$$

where (29) indicates the state transitions according to the well-known law of kinematics. That is, in the first row of (29) the position is modeled as transitioning according to a second-order time-dependent model, the second row indicates velocity transitions via a linear time-dependent model, and the last row indicates that the acceleration is constant; hence the name, nearly constant acceleration. As mentioned above, one important aspect of Kalman Filter design is selecting a dynamical model that is appropriate to the problem one is modeling. In scenarios where model dynamics are known to change, a fixed $\mathbf{F}_k$ can be problematic. For example, in the context of air traffic control, one may have a plane that flies at a constant velocity for a long period and then exhibits accelerated turning or descending. In the next section we discuss the IMMKF, which address the limitations

of a single dynamical model.

2.2 Interacting Multiple Model Kalman Filter

As discussed in section 2.1, capturing the correct model dynamics with a single Kalman Filter can be problematic in situations where the state being estimated exhibits multiple dynamic modes. For example, when constant velocity motion transitions to accelerated motion as a plane begins turning. These two distinct modes, constant velocity motion and constant acceleration, are modeled by NCV and NCA dynamical models respectively in a Kalman Filter. The IMMKF address this limitation by using multiple Kalman Filters with varying internal parameters, which often include the process noise covariance matrix and dynamical model [4, 5, 39, 48]. While the idea of multiple models is elegant and the idea has a certain simplicity, the essence of the IMMKF is in how it combines, or weights, the multiple models to produce a single state estimate. The next two sections consist of a general overview of the IMMKF followed by a more technical description. Both these sections closely follow [48] and are included here for reference.

2.2.1 General Description of the Interacting Multiple Model Kalman Filter

Figure 6 presents the steps for a single iteration of the IMMKF. Since the IMMKF has M distinct Kalman Filters, step (a) consists of M three-tuples $\left(\hat{\mathbf{x}}^{(i)}_{k\text{-}1|k\text{-}1}, \mathbf{P}^{(i)}_{k\text{-}1|k\text{-}1}, \mu^{(i)}_{k\text{-}1}\right)$, for $i = \{1, 2, \ldots, M\}$. $\hat{\mathbf{x}}^{(i)}_{k\text{-}1|k\text{-}1}$ and $\mathbf{P}^{(i)}_{k\text{-}1|k\text{-}1}$ are either the *a posteriori* state estimate and associated covariance at time k-1 for $i = \{1, 2, \ldots, M\}$ from the previous iteration or the initial estimates. However, $\mu^{(i)}_{k\text{-}1}$ is a new term not seen in the standard Kalman Filter. Recall that one goal of the IMMKF is to allow the use of multiple dynamical models, each defined by one of the M distinct Kalman Filters. Here the

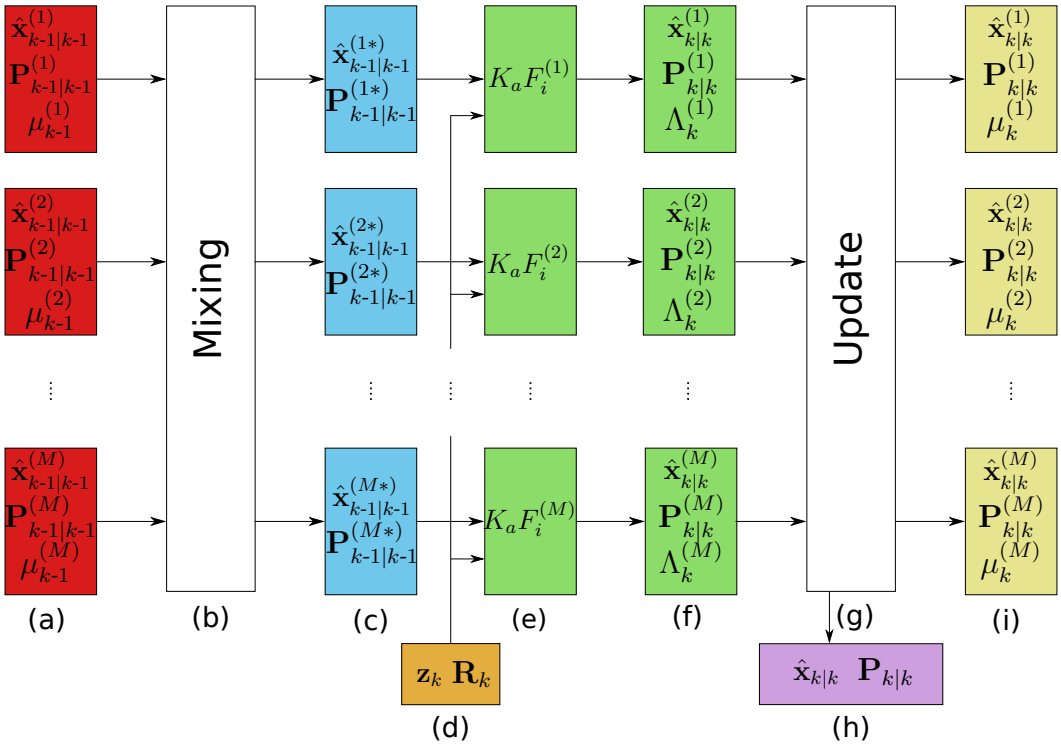


Figure 6: This figure depicts a single iteration of the IMMKF. Step (a) is either the outputs from the prior iteration of the IMMKF or the initial inputs at time zero. In addition to the state estimate and associated covariance, the term $\mu_{k\text{-}1}^{(\cdot)}$ in step (a) represents the probability the state being estimated was in the mode defined by the i^{th} Kalman Filter at time k-1, conditioned on all measurements up to and including time k-1. These probabilities are then used in the mixing step (b) to determine $\hat{\mathbf{x}}_{k\text{-}1|k\text{-}1}^{(*)}$ and $\mathbf{P}_{k\text{-}1|k\text{-}1}^{(*)}$ as a linear combination of the state estimates and associated covariances in step (a) for each of the M Kalman Filters. The output of the mixing step in step (c), along with the measurement and measurement noise covariance matrix in step (d), are then passed as input to the corresponding Kalman Filter in step (e). Note that each of the M Kalman Filters receives the same measurement and measurement noise covariance matrix. The *a posteriori* estimate and associated covariance for each of the M Kalman Filters appear in step (f), along with $\Lambda_k^{(\cdot)}$, which is the likelihood of the measurement for the a given Kalman Filter. The M *a posteriori* estimates and associated covariances are combined in step (g) to form the IMMKF's single *a posteriori* estimate and associated covariance in step (h). Lastly, the M *a posteriori* estimates and associated covariances from step (f), along with the updated probability $\mu_k^{(\cdot)}$, in step (i) are used by the IMMKF at step (a) of the next iteration. *Apart from minor notational differences, this figure is the same as in [48] and is included here for reference.*

term $\mu_{k\text{-}1}^{(i)}$ represents the probability the state being estimated was in the mode defined by the i^{th} Kalman Filter at time k-1, conditioned on all measurements up to and including time k-1. Herein, we sometimes refer to this as simply the IMMKF "being in mode i at time k-1". Although one might consider passing $\hat{\mathbf{x}}_{k\text{-}1|k\text{-}1}^{(i)}$ and $\mathbf{P}_{k\text{-}1|k\text{-}1}^{(i)}$ directly to the corresponding i^{th} Kalman Filter, there is an intermediate mixing at step (b). Here each $\mu_{k\text{-}1}^{(i)}$ (along with other terms) for $i = \{1, 2, \ldots, M\}$ are used to form linear combinations of the terms from step (a). The output of this mixing step is shown in step (c), which can be thought of as the weighted *"a posteriori"* state estimate for each of the M Kalman Filters from step (a). These, along with the measurements $\mathbf{z}_k$ and measurement noise covariance matrix $\mathbf{R}_k$ in step (d), are then input to the M Kalman Filters in step (e), each represented by $K_a F_i^{(i)}$ for $i = \{1, 2, \ldots, M\}$, to produce their respective *a posteriori* estimate and associated covariance. Step (f) shows another three-tuple, with $\hat{\mathbf{x}}_{k|k}^{(i)}$ and $\mathbf{P}_{k|k}^{(i)}$ representing the *a posteriori* output of the i^{th} Kalman Filter at time k. Since there is only one measurement at time k for all M Kalman Filters, the third term, $\Lambda_k^{(i)}$, is the likelihood of the measurement $\mathbf{z}_k$ given the IMMKF is in mode i at time k, which is used to determine the updated $\mu_k^{(i)}$. Similar to step (b), in step (g) the *a posteriori* estimates from step (f) are combined based upon the updated $\mu_k^{(i)}$. In step (h) a single *a posteriori* state estimate and associated covariance, $\hat{\mathbf{x}}_{k|k}$ and $\mathbf{P}_{k|k}$, are output and serve as the IMMKF's output state estimate and covariance at time k. Lastly, at step (i) each of the M *a posteriori* state estimates and associated covariances, along with the updated $\mu_k^{(i)}$, serve as the three-tuples in step (a) at the next iteration.

2.2.2 Mathematical Description of the Interacting Multiple Model Kalman Filter

In this section we present the steps in Figure 6 in more mathematical detail. When implementing the IMMKF a transition matrix, p_{ij}, must be defined prior to implementing the algorithm. This represents the probability of being in mode j at time k given the IMMKF was in mode i at time k-1

$$p_{ij} = Pr(m_k = j | m_{k\text{-}1} = i) \tag{30}$$

where $m_k = j$ means the "IMMKF was in mode j at time k". Also $\mu_{k\text{-}1}^{(i)}$ mentioned above is formally expressed as

$$\mu_{k\text{-}1}^{(i)} = Pr\big(m_{k\text{-}1} = i | \{\mathbf{z}_l\}_{l=1}^{k\text{-}1}\big) \tag{31}$$

where $\mathbf{z}_l$ is the measurement at time l. Algorithmically, the IMMKF proceeds in the following steps:

1. Given the input in Figure 6(a), the first step is to compute the probability the IMMKF was in mode i at time k-1 given it is in mode j at time k, conditioned on all measurements up to and including time k-1. This is expressed

mathematically as

$$\mu_{k\text{-}1}^{(i|j)} = Pr\big(m_{k\text{-}1} = i | m_k = j, \{\mathbf{z}_l\}_{l=1}^{k\text{-}1}\big) \tag{32}$$

$$= \frac{Pr\big(m_k = j | m_{k\text{-}1} = i, \{\mathbf{z}_l\}_{l=1}^{k\text{-}1}\big) Pr\big(m_{k\text{-}1} = i | \{\mathbf{z}_l\}_{l=1}^{k\text{-}1}\big)}{Pr\big(m_k = j | \{\mathbf{z}_l\}_{l=1}^{k\text{-}1}\big)} \tag{33}$$

$$= \frac{Pr(m_k = j | m_{k\text{-}1} = i) Pr\big(m_{k\text{-}1} = i | \{\mathbf{z}_l\}_{l=1}^{k\text{-}1}\big)}{Pr\big(m_k = j | \{\mathbf{z}_l\}_{l=1}^{k\text{-}1}\big)} \tag{34}$$

$$= \frac{p_{ij}\mu_{k\text{-}1}^{(i)}}{\bar{c}_j}, \quad \bar{c}_j = \sum_{i=1}^{M} p_{ij}\mu_{k\text{-}1}^{(i)} \tag{35}$$

where (33) results from applying Bayes' theorem to (32), (34) results from the fact the probability of transitioning from mode i at time k-1 to mode j at time k is independent of $\{\mathbf{z}_l\}_{l=1}^{k\text{-}1}$ (i.e. (30) is independent of $\{\mathbf{z}_l\}_{l=1}^{k\text{-}1}$), and (35) follows from substituting (30) and (31) in the numerator and applying the law of total probabilities to the denominator.

2. Using the mixing probabilities from (35), the next step is to compute the mixed state estimates and associated covariances corresponding to Figure 6(c). This is done according to

$$\hat{\mathbf{x}}_{k\text{-}1|k\text{-}1}^{(j*)} = \sum_{i=1}^{M} \hat{\mathbf{x}}_{k\text{-}1|k\text{-}1}^{(i)} \mu_{k\text{-}1}^{(i|j)} \tag{36}$$

$$\mathbf{P}_{k\text{-}1|k\text{-}1}^{(j*)} = \sum_{i=1}^{M} \left[\mathbf{P}_{k\text{-}1|k\text{-}1}^{(i)} + \big(\hat{\mathbf{x}}_{k\text{-}1|k\text{-}1}^{(i)} \text{-} \hat{\mathbf{x}}_{k\text{-}1|k\text{-}1}^{(j*)}\big)\big(\hat{\mathbf{x}}_{k\text{-}1|k\text{-}1}^{(i)} \text{-} \hat{\mathbf{x}}_{k\text{-}1|k\text{-}1}^{(j*)}\big)^{\mathsf{T}} \right] \mu_{k\text{-}1}^{(i|j)} \tag{37}$$

for $j = \{1, 2, \ldots, M\}$, where $\hat{\mathbf{x}}_{k\text{-}1|k\text{-}1}^{(j*)}$ and $\mathbf{P}_{k\text{-}1|k\text{-}1}^{(j*)}$ are the mixed *a posteriori* state estimate and associated covariance input to the j^{th} Kalman Filter at time k. These, along with the measurements $\mathbf{z}_k$ and measurement noise covariance matrix $\mathbf{R}_k$, are then used by each of the M Kalman Filters to produce their

respective *a posteriori* outputs $\hat{\mathbf{x}}_{k|k}^{(j)}$ and $\mathbf{P}_{k|k}^{(j)}$ for $j = \{1, 2, \ldots, M\}$ at time k in Figure 6(f).

3. In order to produce a single *a posteriori* output in Figure 6(g), the M *a posteriori* outputs from Figure 6(f) are combined in Figure 6(g). These combinations are similar to (36) and (37) using an updated mixing probability $\mu_k^{(j)}$ for $j = \{1, 2, \ldots, M\}$. This updated mixing probability is the *a posteriori* probability of the IMMKF being in mode j at time k, hence it should consider the measurement $\mathbf{z}_k$. The first step in determining the updated mixing probability is to compute $\Lambda_k^{(j)}$, which is the likelihood of measurement $\mathbf{z}_k$ given the IMMKF is in mode j at time k. For each $j = \{1, 2, \ldots, M\}$, this is determined by evaluating the Gaussian probability density function (PDF).

$$\Lambda_k^{(j)} = \mathcal{N}\left(\mathbf{z}_k; \mathbf{H}_k \hat{\mathbf{x}}_{k|k}^{(j)}, \mathbf{S}_k^{(j)}\right) \tag{38}$$

$$= \frac{1}{(2\pi)^{n/2}\left|\mathbf{S}_k^{(j)}\right|} e^{-\frac{1}{2}\left(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k}^{(j)}\right)\left(\mathbf{S}_k^{(j)}\right)^{-1}\left(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k}^{(j)}\right)^{\mathsf{T}}} \tag{39}$$

where, for time k, $\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k}^{(j)}$ is the j^{th} IMMKF's innovation (6), $\mathbf{S}_k^{(j)}$ is the associated innovation covariance (7), and n is the dimensions of $\mathbf{z}_k$. Rewriting

(31) updated for time k gives

$$\mu_k^{(j)} = Pr\big(m_k = j|\{\mathbf{z}_l\}_{l=1}^{k}\big) \tag{40}$$

$$= Pr\big(m_k = j|\mathbf{z}_k, \{\mathbf{z}_l\}_{l=1}^{k\text{-}1}\big) \tag{41}$$

$$= \frac{Pr\big(\mathbf{z}_k|m_k = j, \{\mathbf{z}_l\}_{l=1}^{k\text{-}1}\big) Pr\big(m_k = j|\{\mathbf{z}_l\}_{l=1}^{k\text{-}1}\big)}{Pr\big(\mathbf{z}_k|\{\mathbf{z}_l\}_{l=1}^{k\text{-}1}\big)} \tag{42}$$

$$= \frac{\mathcal{N}\Big(\mathbf{z}_k; \mathbf{H}_k\hat{\mathbf{x}}_{k|k}^{(j)}, \mathbf{S}_k^{(j)}\Big) Pr\big(m_k = j|\{\mathbf{z}_l\}_{l=1}^{k\text{-}1}\big)}{Pr(\mathbf{z}_k)} \tag{43}$$

$$= \frac{\Lambda_k^{(j)} \sum_{i=i}^{M} Pr\big(m_k = j|m_{k\text{-}1} = i, \{\mathbf{z}_l\}_{l=1}^{k\text{-}1}\big) Pr\big(m_{k\text{-}1} = i, \{\mathbf{z}_l\}_{l=1}^{k\text{-}1}\big)}{Pr(\mathbf{z}_k)} \tag{44}$$

$$= \frac{\Lambda_k^{(j)} \sum_{i=i}^{M} p_{ij}\mu_{k\text{-}1}^{(i)}}{Pr(\mathbf{z}_k)} \tag{45}$$

$$= \frac{1}{c}\Lambda_k^{(j)}\bar{c}_j \tag{46}$$

where $\bar{c}_j$ is defined in (35) and $c = \sum_{j=1}^{M} \Lambda_k^{(j)}\bar{c}_j$ is a normalizing constant which ensures $\sum_{j=1}^{M} \mu_k^{(j)} = 1$.

4. Having updated the mixing probabilities the output IMMKF state estimate and covariance are determined in the mixing step, Figure 6(g), as

$$\hat{\mathbf{x}}_{k|k} = \sum_{j=1}^{M} \hat{\mathbf{x}}_{k|k}^{(j)}\mu_k^{(j)} \tag{47}$$

$$\mathbf{P}_{k|k} = \sum_{j=1}^{M} \left[\mathbf{P}_{k|k}^{(j)} + \big(\hat{\mathbf{x}}_{k|k}^{(j)}\text{-}\hat{\mathbf{x}}_{k|k}\big)\big(\hat{\mathbf{x}}_{k|k}^{(j)}\text{-}\hat{\mathbf{x}}_{k|k}\big)^{\mathsf{T}}\right]\mu_k^{(j)} \tag{48}$$

where $\hat{\mathbf{x}}_{k|k}^{(j)}$ and $\mathbf{P}_{k|k}^{(j)}$ are the j^{th} Kalman Filter's *a posteriori* estimate and associated covariance at time k.

5. Lastly, in Figure 6(i) the M individual *a posteriori* state estimates, their associated covariances, and the updated mixing probabilities from Figure 6(f) are passed to the next iteration of the IMMKF.

This concludes our discussion of the Kalman Filter and IMMKF. In sections 2.5 and 2.6 we present our main theoretical contribution where deep autoencoders are combined with the Kalman Filter and IMMKF. First though, we present an overview of autoencoders and LSTMs.

2.3 Autoencoders

An autoencoder [11] is a type of artificial neural network which attempts to copy an input ϕ to an output $\hat{\phi}$, using intermediate mappings which constitute the network's hidden layers. The case where there are multiple hidden layers is referred to as a deep autoencoder. The mappings between the hidden layers are typically non-linear and may increase or decrease the dimensions of the preceding layer's feature space. In practice, a common application of an autoencoder is dimensionality reduction, similar to principal component analysis (PCA). In fact, if the mappings between the hidden layers are linear, an autoencoder will span the same subspace as PCA [11]. However, if they are non-linear, an autoencoder may be used to achieve non-linear dimensionality reduction.

Figure 7 depicts a deep autoencoder with depth, or number of layers, S and varying dimensions per layer, both indicated by ellipsis. The mapping from the input, ϕ, to the smallest hidden layer, or *latent layer*, $\mathbf{z}$ is the encoder and the mapping from the latent layer to the output, $\hat{\phi}$, is the decoder. While it may seem trivial to copy ϕ to $\hat{\phi}$, the essence of the autoencoder in Figure 7 is that it must learn the important features to encode such that an accurate representation of the input can

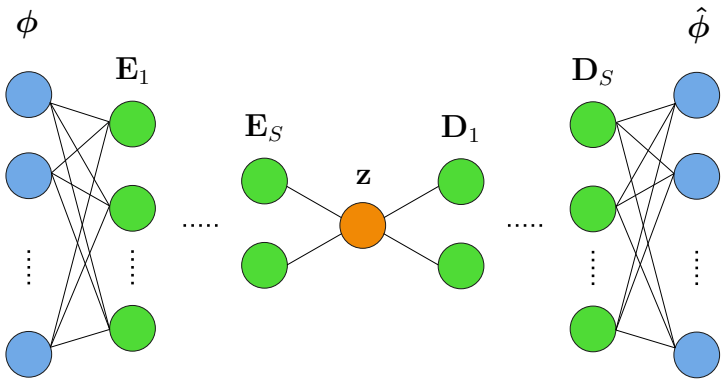


Figure 7: Diagram of a deep autoencoder with input ϕ, output $\hat{\phi}$, a single latent layer $\mathbf{z}$, and S hidden layers in both the encoder and decoder represented by $\mathbf{E}_k$ and $\mathbf{D}_k$ for $k = \{1, 2, \ldots, S\}$ respectively. The stacked discs represent the values in each layer while vertical ellipsis indicate the omitted values and the horizontal ellipsis represent omitted layers.

be reconstructed from the latent layer. In the case of dimensionality reduction, since the latent layer has a smaller feature space than the input, some information is lost in the encoding. Thus, the goal of the autoencoder is to learn the mappings between hidden layers such that the features essential to reconstruction are preserved. In other words, the autoencoder must learn what to throw out and what to keep. Representing the encoder and decoder mappings by $\mathbf{E}_{(\mathbf{W},\mathbf{b})}$ and $\mathbf{D}_{(\mathbf{W},\mathbf{b})}$, the autoencoder's cost function is

$$J(\mathbf{W}, \mathbf{b}) = \arg\min_{\mathbf{W},\mathbf{b}} \left\| \phi - \hat{\phi} \right\|_F^2 \tag{49}$$

$$= \arg\min_{\mathbf{W},\mathbf{b}} \left\| \phi - (\mathbf{D}_{(\mathbf{W},\mathbf{b})} \circ \mathbf{E}_{(\mathbf{W},\mathbf{b})})(\phi)) \right\|_F^2 \tag{50}$$

where $\mathbf{W}$ and $\mathbf{b}$ are the matrices and vectors learned by the autoencoder and the norm is the Frobenius norm squared.

In the case of non-linear mappings, the function mapping between layers is represented by $\sigma(\cdot)$ and is commonly a sigmoid, hyperbolic tangent or rectified

linear unit (ReLU) [49] function, where the argument of the function is an affine transformation of the previous layer's output. For example, the mapping from hidden layer m to $m+1$ would be

$$H_{m+1} = \sigma(\mathbf{W}_m H_m + \mathbf{b}_m) \tag{51}$$

where the individual entries in $\mathbf{W}_m$ and $\mathbf{b}_m$ are learned for each of the hidden layers in the network via backpropagation [53]. Thus, despite the mystification of deep learning, a deep autoencoder is nothing more than the composition of N affine transformations $g(x)$ and activation functions $\sigma(x)$

$$(\sigma_N \circ g_N \circ \sigma_{N-1} \circ g_{N-1} \circ \cdots \sigma_1 \circ g_1)(x) \tag{52}$$

For more details on autoencoders, including many of the common variants, see [11].

2.4 Long-Short Term Memory Recurrent Neural Network

A classic approach in the deep learning literature is an LSTM RNN applied to time series. As we will be comparing the AEKF against an LSTM, here we present some background on LSTMs.

LSTM networks were initially introduced to solve the vanishing gradient problem present in RNNs [23]. As a result, LSTMs are able to learn long term dependencies among data separated in space and time by preserving network error throughout. This facilitates the application of LSTMs to sequence labeling problems [18], where the features of the input vectors can not be assumed to be independent, such as natural language processing (NLP) [64]. LSTMs have also

been applied to time series classification [30] and prediction [66]. In the context of such methods, the AEKF may be seen as contribution to time series state estimation and prediction problems involving deep learning, where the Kalman Filter is used for state estimation and the encoder/decoder are used for non-linear data transformation. An excellent general introduction to the LSTM can be found at *https://colah.github.io/posts/2015-08-Understanding-LSTMs*.

The standard LSTM implementation is composed of four main components: a cell or memory unit and three gates which serve to regulate the flow of information into and out of the cell unit. The cell unit acts as the memory and keeps track of the relation between elements in the input sequence. For example, in the case of a time series the cell would maintain a history of the dependencies between elements of the time series. As the LSTM processes its input, it needs to determine which new values are added to the cell, which are maintained in the cell, and which are removed. This is accomplished by the input, output, and forget gates. Formally, these three gates are defined by the composition of affine transformations and non-linear activations functions as

$$\mathcal{F}_k^{(i)} = \sigma(\mathbf{W}_i x_k + \mathbf{U}_i h_{k-1} + \mathbf{b}_i) \tag{53}$$

$$\mathcal{F}_k^{(o)} = \sigma(\mathbf{W}_o x_k + \mathbf{U}_o h_{k-1} + \mathbf{b}_o) \tag{54}$$

$$\mathcal{F}_k^{(f)} = \sigma(\mathbf{W}_f x_k + \mathbf{U}_f h_{k-1} + \mathbf{b}_f) \tag{55}$$

where, at time k, the input sequence is x_k, the hidden state or LSTM output from the previous iteration is $h_{k\text{-}1}$, $\mathbf{W}$, $\mathbf{U}$, and $\mathbf{b}$ are the weights and biases learned during training, and σ is a sigmoid activation function. The subscripts i, o, and f, indicate which of the three gates (input, output, or forget) the weights and biases belong to. Similarly, the superscript in parenthesis in $\mathcal{F}_k^{(\cdot)}$ indicates which gate the

function corresponds to. Each of the three gate functions takes as input the current input sequence $x_k \in \mathbb{R}^m$ and the previous LSTM output $h_{k\text{-}1} \in \mathbb{R}^n$, and then applies their weights and biases accordingly. The output of all three gates are in $\mathbb{R}^n$.

The cell state is updated based upon the previous cell state and a cell input vector as

$$\mathcal{F}_k^{(c)} = \mathcal{F}_k^{(f)} \odot \mathcal{F}_{k-1}^{(c)} + \mathcal{F}_k^{(i)} \odot \mathcal{F}_k^{(\hat{c})} \tag{56}$$

where $\mathcal{F}_{k-1}^{(c)}$ is the cell state at k-1, $\odot$ is the Hadamard product, and the cell input vector is defined similarly to the gate functions as

$$\mathcal{F}_k^{(\hat{c})} = \hat{\sigma}(\mathbf{W}_c x_k + \mathbf{U}_c h_{k-1} + \mathbf{b}_c) \tag{57}$$

where $\hat{\sigma}$ is now the hyperbolic tangent activation function. Inspection of (56) indicates how the LSTM determines which new information to add to the cell state and which to remove from the cell state. Since the activation function in both $\mathcal{F}_k^{(f)}$ and $\mathcal{F}_k^{(i)}$ is a sigmoid function, combined with the Hadamard product, these vectors weight each entry in $\mathcal{F}_{k-1}^{(c)}$ and $\mathcal{F}_{k-1}^{(\hat{c})}$ respectively between 0 and 1. This provides some intuition into why (53)-(55) are referred to as gate functions.

Finally, the output of the LSTM is given by

$$\mathcal{F}_k^{(h)} = \mathcal{F}_k^{(o)} \odot \hat{\sigma}(\mathcal{F}_k^{(c)}) \tag{58}$$

where the output gate, $\mathcal{F}_k^{(o)}$, acts element-wise on the cell state composed with a hyperbolic tangent function. Other variants such as the peephole LSTM [16, 17] and convolutional LSTM [55] also exist with differences in architecture.

2.5 Autoencoder-Kalman Filter

Before presenting the details of the AEKF we stop to make a point about our philosophical approach in designing the AEKF. The combination of an autoencoder and Kalman Filter merges the computational power of deep learning with the theoretical understanding of a simple and elegant linear system. In our experience, one of the more challenging aspects of working with neural networks is the lack of mathematical principles to guide training. However, if combined with well-known mathematical models such as the Kalman Filter, certain design and training issues can be addressed in the Kalman Filter itself, where the decision was informed by the Kalman Filter's well-known theoretical basis. At the same time, the computational power of deep learning allows a hybrid model such as the AEKF to outperform a traditional Kalman Filter. Wanting to only leverage deep learning for parameters external to the Kalman Filter, it was a specific design decision to only utilize the autoencoder's computational resources for learning or transforming parameters related to the measurements, i.e. only $\{\mathbf{z}_k\}_{k=1}^N$ and $\{\mathbf{R}_k\}_{k=1}^N$. As we will see in section 4.5, "what" the AEKF learns is entirely consistent and can be derived from the mathematical theory of the Kalman Filter.

2.5.1 Technical Details

In the larger picture, the AEKF simply places a Kalman Filter in the latent layer of the standard deep autoencoder [11] shown in Figure 7. However, due to the presence of the Kalman Filter, the roles of the encoder and decoder portions of the AEKF vary slightly from a standard autoencoder. That is, while the encoder and decoder portions of the AEKF ensure that the input and output of the AEKF are in the same space, the goal is not to minimize the difference between the input

and output, but the difference between the ground truth of the input and a state estimate of the input. The question may be asked here how access to the ground truth is possible, given that the goal of the AEKF is state estimation. This is made possible by the use of domain randomization as will be discussed in section 3.2.

With the AEKF we make a slight change in notation from the traditional Kalman Filter. In keeping with the notation that the sequence of measurements passed to the Kalman Filter is $\{\mathbf{z}_k\}_{k=1}^N$, we define the actual measurements, prior to transformation by the encoder, as $\{\boldsymbol{\phi}_k\}_{k=1}^N$, consistent with the notation in Figure 7. Given a single time series sequence of measurements $\{\boldsymbol{\phi}_k\}_{k=1}^N$, with k indexing time, or sample, and each $\boldsymbol{\phi}_k \in \mathbb{R}^p$, we define the stacking of $\{\boldsymbol{\phi}_k\}_{k=1}^N$ into a matrix as $\boldsymbol{\phi}$. Here there are N rows with each row representing a sample in the time series and p columns representing the dimension of each sample, thus $\boldsymbol{\phi} \in \mathbb{R}^{N\times p}$. This notation applies throughout to any variable defined in terms of as sequence, for example $\{\mathbf{z}_k\}_{k=1}^N$, $\{\mathbf{R}_k\}_{k=1}^N$, etc.

Figure 8 presents the AEKF and is similar to Figure 7 except that the single latent layer has been replaced by $\mathbf{z}$, $\mathbf{R}$, K_aF_i, and $\hat{\mathbf{z}}$, which represent the learned measurements, learned measurement noise covariance matrix, Kalman Filter, and Kalman Filter's *a priori* estimate of the learned measurements respectively. The sequence $\{\boldsymbol{\phi}_k\}_{k=1}^N$ mentioned above is shown in the first column and represented by the variable $\boldsymbol{\phi}$. The input $\boldsymbol{\phi}$ is passed to the encoder portion of the AEKF represented by $\mathbf{E}_1$ through $\mathbf{E}_S$ where S is the number of layers in the encoder portion. The encoder portion then outputs two variables: $\mathbf{z}$ and $\mathbf{R}$, which represent the sequences $\{\mathbf{z}_k\}_{k=1}^N$ and $\{\mathbf{R}_k\}_{k=1}^N$ respectively. From the encoder's perspective these are just two variables resulting from the encoder portion's transformation of $\boldsymbol{\phi}$. However, from the point of view of the Kalman Filter placed in the AEKF's

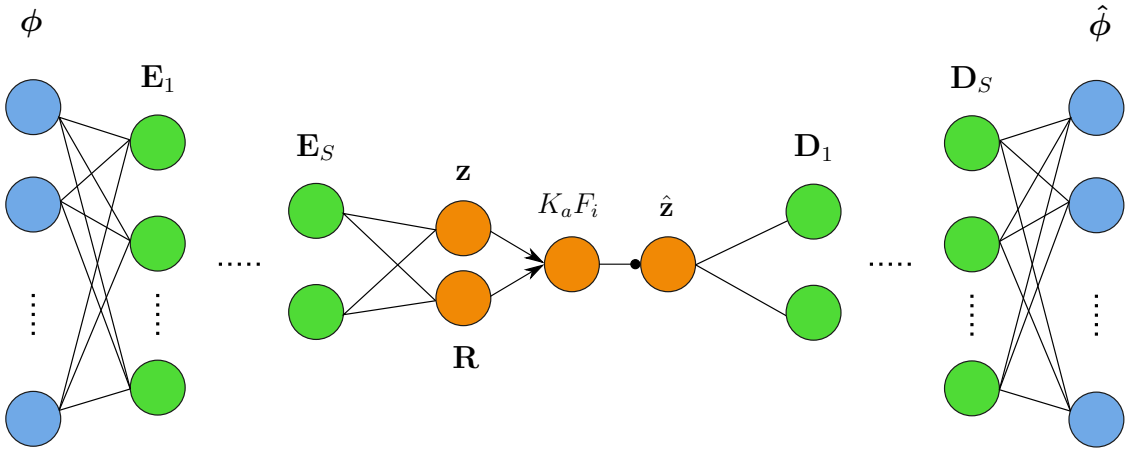


Figure 8: Diagram of the Autoencoder-Kalman Filter which is the same as Figure 7 apart from the presence of a Kalman Filter, and related variables, in the latent layer. Here the output of the encoder results in two variables, $\mathbf{z}$ and $\mathbf{R}$, which are the measurements and measurement noise covariance matrices passed to the Kalman Filter, represented by K_aF_i. The Kalman Filter's *a priori* estimate of $\mathbf{z}$, $\hat{\mathbf{z}}$, is then passed to the decoder, which maps $\hat{\mathbf{z}}$ to the final output $\hat{\phi}$. The lines with arrows to the left of the Kalman Filter indicate the Kalman Filters inputs without any transformation and the line with a disc indicates the linear transformation $\mathbf{H}$.

latent layer, they represent the measurements and associated measurement noise covariance matrices of the Kalman Filter. Thus, the AEKF's goal is to learn a transformation of ϕ, via the encoder portion, such that $\{\mathbf{z}_k\}_{k=1}^N$ and $\{\mathbf{R}_k\}_{k=1}^N$ allow the Kalman Filter to accurately filter ϕ. However, the output of the Kalman Filter, $\hat{\mathbf{z}}$, is not the actual filtered version of ϕ as $\hat{\mathbf{z}}$ is in the AEKF's latent space, which is not necessarily in the same space as ϕ. Thus, the decoder portion with layers $\mathbf{D}_1$ through $\mathbf{D}_S$ maps $\hat{\mathbf{z}}$ from the AEKF's latent dimension to the original measurement dimension. The AEKF's cost function then compares the AEKF output, $\hat{\phi}$, with the ground truth of ϕ and updates the encoder and decoder weights and biases via backpropagation. Note, in Figure 8, the lines between the encoder and decoder layers represent the standard composition of affine transformations and non-linear activations in a neural network, the lines with arrows to the left of the

Kalman Filter indicate the Kalman Filter's inputs without any transformation, and the line with a disc indicates the linear transformation $\mathbf{H}$. The horizontal ellipses represent the omitted hidden layers, and the vertical ellipses represent each layer's omitted samples.

In slightly more mathematical detail, the AEKF is summarized as follows:

1. Apply a transformation, $\mathbf{E}(\cdot)$, to $\boldsymbol{\phi}$ using the encoder portion of the AEKF, where $\mathbf{E}(\boldsymbol{\phi}) = \mathbf{E}_S \circ \mathbf{E}_{S\text{-}1} \circ \cdots \circ \mathbf{E}_1(\boldsymbol{\phi})$
2. Apply two affine transformations, $\mathbf{z} = \mathbf{W}_z\mathbf{E}(\boldsymbol{\phi}) + \mathbf{b}_z$ and $\mathbf{L} = \mathcal{T} \circ (\mathbf{W}_L\mathbf{E}(\boldsymbol{\phi}) + \mathbf{b}_L)$, to $\mathbf{E}(\boldsymbol{\phi})$, where $\mathcal{T}$ is a stacking operation described below.
3. Compute $\mathbf{R} = \mathbf{L}^\intercal\mathbf{L}$ and pass $\mathbf{z}$ and $\mathbf{R}$ to the Kalman Filter.
4. Return the Kalman Filter's *a priori* estimate, $\hat{\mathbf{z}} = K_a F_i(\mathbf{z}, \mathbf{R})$
5. Apply an affine transformation, $\hat{\mathbf{z}}_p = \mathbf{W}_p\hat{\mathbf{z}} + \mathbf{b}_p$, to $\hat{\mathbf{z}}$
6. Apply a transformation, $\mathbf{D}(\cdot)$, to $\hat{\mathbf{z}}_p$ using the decoder portion of the AEKF, where $\mathbf{D}(\hat{\mathbf{z}}_p) = \mathbf{D}_S \circ \mathbf{D}_{S\text{-}1} \circ \cdots \circ \mathbf{D}_1(\hat{\mathbf{z}}_p)$
7. Apply an affine transformation, $\hat{\boldsymbol{\phi}} = \mathbf{W}_o\mathbf{D}(\hat{\mathbf{z}}_p) + \mathbf{b}_o$, to $\mathbf{D}(\hat{\mathbf{z}}_p)$
8. Compute $||\boldsymbol{\phi}_{true} - \hat{\boldsymbol{\phi}}||_F^2$, where $\boldsymbol{\phi}_{true}$ is the ground truth of $\boldsymbol{\phi}$, and backpropagate the errors, updating the weights and biases.

Steps (1) and (2) contain the essence of how the AEKF addresses the problem of learning transformations related to the original measurements $\boldsymbol{\phi}$. $\mathbf{E}(\boldsymbol{\phi})$ in step (1) is the result of multiple non-linear and affine function compositions, which transform the original measurements, $\{\boldsymbol{\phi}_k\}_{k=1}^N$, producing a new basis, $\{\mathbf{z}_k\}_{k=1}^N$, onto which the Kalman Filter's state estimate, $\{\hat{\mathbf{x}}_k\}_{k=1}^N$, is an orthogonal projection,

as indicated by (13). However, the *bases are not fixed but learned by the AEKF*, as $\{\mathbf{z}_k\}_{k=1}^N$ is a function of the parameters learned by the AEKF. Thus, if the AEKF shows improved performance over a standard Kalman Filter, it is finding a transformation of the measurements that produces a more accurate orthogonal projection. Furthermore, while in practice only the actual measurements are available, the compositional nature of the autoencoder's encoder mapping allows the input dimensions to be mapped to an arbitrary number of output dimensions. Thus, we not only use the encoder to learn a transformation of $\{\boldsymbol{\phi}_k\}_{k=1}^N$, resulting in an input to the Kalman Filter $\{\mathbf{z}_k\}_{k=1}^N$, but also the covariance of this input $\{\mathbf{R}_k\}_{k=1}^N$. It is important to note that in the Kalman Filter the only term affected by $\{\mathbf{z}_k\}_{k=1}^N$ is the *a posteriori* output (9). Thus, if the AEKF were to only learn $\{\mathbf{z}_k\}_{k=1}^N$ the Kalman Gain (8) would not be affected by the encoder portion of the AEKF. This fact plays an essential role in AEKF and will be discussed below.

2.5.2 Autoencoder-Kalman Filter Formalism

Given an input $\boldsymbol{\phi} \in \mathbb{R}^{N \times p}$, the layer-by-layer mathematical model discussed above for the AEKF is broken down into four parts:

$$
\text{Encoder} \begin{cases}
\mathbf{E}_1 = \sigma\big(\mathbf{W}_1^{(e)}\boldsymbol{\phi} + \mathbf{b}_1^{(e)}\big) & (59)\\
\mathbf{E}_2 = \sigma\big(\mathbf{W}_2^{(e)}\mathbf{E}_1 + \mathbf{b}_2^{(e)}\big) & (60)\\
\vdots & \\
\mathbf{E}_S = \sigma\big(\mathbf{W}_S^{(e)}\mathbf{E}_{S-1} + \mathbf{b}_S^{(e)}\big) & (61)\\
\mathbf{z} = \mathbf{W}_z\mathbf{E}_S + \mathbf{b}_z & (62)\\
\mathbf{L} = \mathcal{T} \circ (\mathbf{W}_L\mathbf{E}_S + \mathbf{b}_L) & (63)\\
\mathbf{R} = \mathbf{L}^{\mathsf{T}}\mathbf{L} & (64)
\end{cases}
$$

$$
\text{Kalman Filter} \begin{cases}
\hat{\mathbf{z}} = K_a F_i(\mathbf{z}, \mathbf{R}) & (65)\\
\hat{\mathbf{z}}_p = \mathbf{W}_p\hat{\mathbf{z}} + \mathbf{b}_p & (66)
\end{cases}
$$

$$
\text{Decoder} \begin{cases}
\mathbf{D}_1 = \sigma(\mathbf{W}_1^{(d)}\hat{\mathbf{z}}_p + \mathbf{b}_1^{(d)}) & (67)\\
\mathbf{D}_2 = \sigma(\mathbf{W}_2^{(d)}\mathbf{D}_1 + \mathbf{b}_2^{(d)}) & (68)\\
\vdots & \\
\mathbf{D}_S = \sigma(\mathbf{W}_S^{(d)}\mathbf{D}_{S-1} + \mathbf{b}_S^{(d)}) & (69)
\end{cases}
$$

$$
\text{Output} \begin{cases}
\hat{\boldsymbol{\phi}} = \mathbf{W}_o\mathbf{D}_S + \mathbf{b}_o & (70)
\end{cases}
$$

where each symbol to the left of the equals sign, for example $\hat{\mathbf{z}}$, is the stacked sequence $\{\mathbf{z}_k\}_{k=1}^N$ as mentioned previously. Equations (59)-(61) represent the encoder portion of the AEKF, which takes $\boldsymbol{\phi}$ in the original measurement space as input. Here $\mathbf{E}_l$ represents the l^{th} hidden layer, for $l = \{1, 2, \ldots, S\}$, in the encoder portion. $\mathbf{W}_l^{(e)}$ and $\mathbf{b}_l^{(e)}$ represent the weights and biases, respectively, of the l^{th}

hidden layer and (e) indicates that the given variable belongs to the encoder portion. Each layer's element-wise activation function is represented by σ. Equations (62) and (63) consist of affine transformations of the encoder output, $\mathbf{E}_S$, with the subscripts z and L on the weights and biases indicating which output variable, $\mathbf{z}$ or $\mathbf{L}$, they correspond to. However, the affine transformation in (63) is composed with a function $\mathcal{T}$, which maps its input vector to an upper triangular matrix with positive diagonal elements. More formally, for $\mathbf{R} \in \mathbb{R}^{m \times m}$

$$\mathcal{T} : \mathbb{R}^{\frac{m^2+m}{2}} \mapsto \mathbb{R}^{m \times m} \tag{71}$$

where $\frac{m^2+m}{2}$ is the number of terms necessary to form an upper triangular matrix in $\mathbb{R}^{m \times m}$. Taking $m = 3$ as an example and writing the input to $\mathcal{T}$ as

$$\boldsymbol{a}^{\mathsf{T}} = [a_1, a_2, a_3, a_4, a_5, a_6] \tag{72}$$

the result of $\mathcal{T} \circ \boldsymbol{a}$ is

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \end{bmatrix} \mapsto \begin{bmatrix} a_1 & a_2 & a_3 \\ 00 & a_4 & a_5 \\ 00 & 00 & a_6 \end{bmatrix} \tag{73}$$

where the matrix on the right side of (73) is $\mathbf{L}$ in (63), from which a symmetric positive definite $\mathbf{R}$ is formed in (64). (65) is the Kalman Filter mapping of $\mathbf{z}$ and $\mathbf{R}$, which returns the Kalman Filter's *a priori* estimate (mapped into the measurement space) and is followed by an affine transformation in (66). In (66), the subscript p

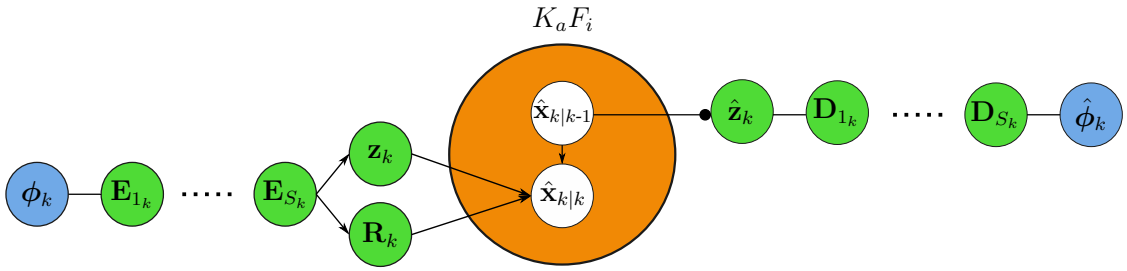


Figure 9: Diagram depicting the path of a single sample, ϕ_k, through the Autoencoder-Kalman Filter. The meaning of the figure elements are the same as in Figure 8. The encoder portion maps ϕ_k to $\mathbf{z}_k$ and $\mathbf{R}_k$ which, along with the *a priori* estimate at time k, are used to compute the Kalman Filter's *a posteriori* estimate at time k. However, it is the *a priori* estimate at time k that is passed to the decoder portion of the AEKF. Thus, the final state estimate, $\hat{\phi}_k$, is independent of ϕ_k. However, $\mathbf{z}_{k-1}$ and $\mathbf{R}_{k-1}$ do affect ϕ_k since $\hat{\mathbf{x}}_{k-1|k-1}$ is used to determine $\hat{\mathbf{x}}_{k|k-1}$. This architecture forces the encoder portion to learn a mapping such that the Kalman Filter's *a priori* estimate, mapped through the decoder, accurately estimates ϕ_k having only seen $\{\phi_l\}_{l=1}^{k-1}$. In this light, while our design of the AEKF uses a fixed dynamical system, the AEKF can be thought of as learning a transformation which attempts to minimize (74) constrained by this fixed dynamical system.

stands for "post" Kalman Filter affine transformation. The decoder portion (67)-(69), taking $\hat{\mathbf{z}}_p$ as input, maps back to the original measurement space. Here the notation is the same as the encoder portion with the exception of (d) replacing (e). Lastly, while keeping the dimensions the same, an affine transformation is applied to the decoder output in (70), resulting in the final AEKF output $\hat{\phi}$.

The AEKF loss function given by (74) utilizes the Frobenius norm. The loss is computed using the difference between the ground truth ϕ_{true} and $\hat{\phi}$ as previously mentioned.

$$J(\mathbf{W}, \mathbf{b}) = \arg\min_{\mathbf{W},\mathbf{b}} \left\| \phi_{true} - \hat{\phi} \right\|_F^2 \tag{74}$$

Since $\hat{\phi}$ is based on the Kalman Filter's *a priori* estimate, $\hat{\phi}$ never "sees" the true ϕ. When ϕ and $\hat{\phi}$ are compared for a fixed k, $\hat{\phi}_k$ is the AEKF's *"a priori" prediction* of ϕ_k, never having seen $\mathbf{z}_k$ and $\mathbf{R}_k$, and thus being independent of ϕ_k. This is

shown in Figure 9, which tracks the full path of a single sample, $\boldsymbol{\phi}_k$, through the AEKF. The input $\boldsymbol{\phi}_k$ passes through the encoder which results in a transformed measurement $\mathbf{z}_k$ and transformed measurement noise covariance matrix $\mathbf{R}_k$. *However, the Kalman Filter in the latent layer only uses* $\mathbf{z}_k$ *and* $\mathbf{R}_k$ *for the a posteriori estimate*, as shown by the lines with arrows from $\mathbf{z}_k$ and $\mathbf{R}_k$ to $\hat{\mathbf{x}}_{k|k}$. The *a priori* estimate of $\mathbf{z}_k$, which is independent of $\mathbf{z}_k$, is given by $\hat{\mathbf{z}}_k = \mathbf{H}_k\hat{\mathbf{x}}_{k|k\text{-}1}$, where the line with a disc indicates the linear transformation $\mathbf{H}_k$. $\hat{\mathbf{z}}_k$ is then passed to decoder which produces the final output $\hat{\boldsymbol{\phi}}_k$. Thus, for $k = \{1, 2, \ldots, N\}$, $\hat{\boldsymbol{\phi}}_k$ is a function of the first k-1 measurements $\{\boldsymbol{\phi}_l\}_{l=1}^{k-1}$. This is different from a traditional autoencoder where $\hat{\phi}_k$, the autoencoder's reconstruction of the input, has "seen" what it is trying to encode during training, i.e. the true $\boldsymbol{\phi}_k$.

Of the three phases in the AEKF, *only the Kalman Filter is making a prediction, i.e, projecting forward in time*. Both the encoder and decoder are simply non-linear transformations of their respective input sequences at a fixed time. As a result, passing $\hat{\mathbf{z}}_p$ to the decoder forces the AEKF to learn a transformation of the measurements that minimizes the AEKF cost function, having only seen the Kalman Filter's *a priori* estimate.

2.6 Autoencoder-Interacting Multiple Model Kalman Filter

The AEIMMKF is conceptually very similar to the AEKF. The primary difference is the Kalman Filter in the latent layer is replaced with an IMMKF. This is shown in Figure 10, which is identical to Figure 8 apart from the M Kalman Filters in the latent layer, representing the IMMKF. Note the line with the disc represents the linear transformation $\mathbf{H}$ applied to (75). Note that each of the M Kalman Filters receive the same inputs $\mathbf{z}$ and $\mathbf{R}$. The box around the M Kalman Filters indicates

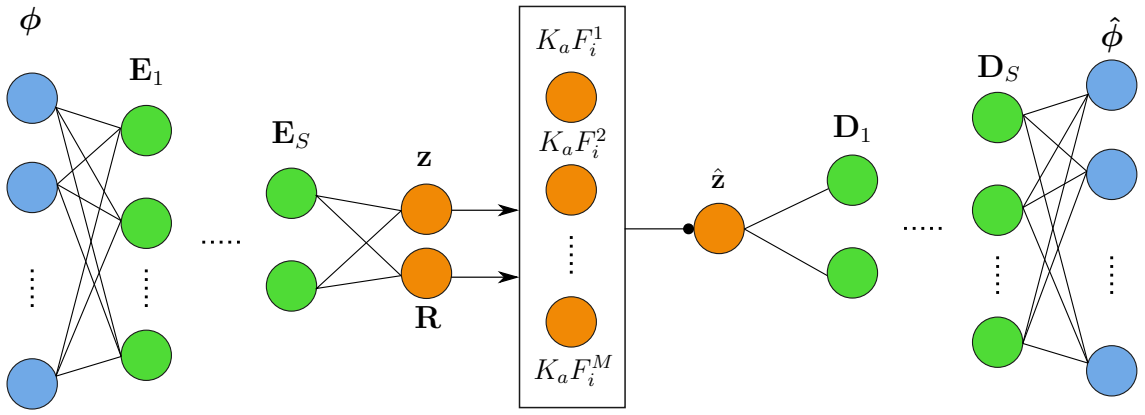


Figure 10: Diagram of the AEIMMKF which is the same as Figure 8 apart from the IMMKF in the latent layer. Note that each of the M Kalman Filters in the IMMKF receive the same $\mathbf{z}$ and $\mathbf{R}$ as input. The box around the IMMKF represents the entire process from Figure 6 with the ellipses indicating the omitted Kalman Filters. Since the AEKF passes the Kalman Filter's *a priori* estimate to its decoder portion, a single *a priori* estimate for the IMMKF must be generated. However the standard IMMKF does not do this. Here we compute a single *a priori* estimate in a manner similar to how the IMMKF computes the single *a posteriori* estimate via (75). This is then mapped by $\mathbf{H}$ to produce $\hat{\mathbf{z}}$

the entire process shown in Figure 6 occurs in the AEIMMKF's latent layer with the ellipses indicating the omitted Kalman Filters. Just as in the IMM, each of the M Kalman Filters are given the same $\mathbf{z}$ and $\mathbf{R}$ and the final estimate, $\hat{\mathbf{z}}$, is the IMMKF's "*a priori* estimate" mapped into the measurement space via $\mathbf{H}$. However, a close inspection of Section 2.2 shows there is no single IMMKF *a priori* estimate, as each of the M Kalman Filter's *a priori* estimate is only used internally in the IMMKF. On the other hand, there is a single *a posteriori* state estimate, given as the linear combination in (47). Since the AEKF passes the *a priori* estimate of the Kalman Filter to its decoder portion, we need a single *a priori* estimate for the IMMKF. This is generated as a linear combination according to

$$\hat{\mathbf{x}}_{k|k-1} = \sum_{j=1}^{M} \hat{\mathbf{x}}_{k|k-1}^{(j)} \mu_{k-1}^{(j)} \tag{75}$$

which is a modification of (47) with the j^{th} Kalman Filter's *a priori* estimate given by $\hat{\mathbf{x}}_{k|k-1}^{(j)}$ and $\mu_{k-1}^{(j)}$ is the mode probability (31) before updating in (46) for $j = \{1, 2, \ldots, M\}$.

3 Training

3.1 Hilbert Space Representations of Functions

In this section we present a summary of Hilbert Space theory, which serves as a background for understanding our application of domain randomization in Section 3.2 and the Hilbert Space Filter in Section 4.7. Although the functions in Section 4.7 are restricted to $\mathbb{R}$, the Hilbert Space presentation here is general and focuses on the canonical Hilbert Space $\mathcal{L}_2(\Omega)$, which is the set of all functions $f(x) : \mathbb{F} \mapsto \mathbb{F}$, where $\mathbb{F}$ is either $\mathbb{R}$ or $\mathbb{C}$, which satisfy

$$\int_{\Omega \in \mathbb{F}} \left(|f(x)|^2 dx \right)^{1/2} < \infty \tag{76}$$

The presentation of Hilbert Spaces in this section is drawn from [21, 31] and the reader is directed therein for further details. Additionally, while the ideas in this section technically apply to separable Hilbert Spaces, we drop "separable" and simply write Hilbert Space.

What is special about Hilbert Space representations of functions is that a large class of functions can be exactly represented by the countably infinite sum

$$f(x) = \sum_{i=0}^{\infty} \langle f(x), \varphi_i \rangle \varphi_i \tag{77}$$

where $\langle \cdot, \cdot \rangle$ is the inner product in $\mathcal{L}_2(\Omega)$ and $\{\varphi_i\}_{i=1}^{\infty}$ are a set of orthonormal func-

tions. These orthonormal functions are analogous to the canonical basis vectors $\boldsymbol{e}_1 = (1, 0, 0)$, $\boldsymbol{e}_2 = (0, 1, 0)$, and $\boldsymbol{e}_3 = (0, 0, 1)$ in $\mathbb{R}^3$ in that they have unit norm, are mutually orthogonal, and form a basis in the Hilbert Space. The concept of norm and orthogonality for functions is based on a generalization of the vector inner product in $\mathbb{R}^n$ to $\mathcal{L}_2(\Omega)$ explained below. Just as any vector in $\mathbb{R}^3$ can be represented as a linear combination of the canonical basis vectors, any function in $\mathcal{L}_2(\Omega)$ can be represented by (77). This ability to represent any vector in $\mathbb{R}^3$ stems from the fact the vectors in the set $\{\boldsymbol{e}_i\}_{i=1}^3$ are a basis for $\mathbb{R}^3$. From this it follows that the inner product of vectors in $\{\boldsymbol{e}_i\}_{i=1}^3$ obey

$$\langle \boldsymbol{e}_i, \boldsymbol{e}_j \rangle = \delta_{i,j} \tag{78}$$

where $\delta_{i,j}$ is the Kronecker delta function, which is 1 if $i = j$ and zero otherwise. In Hilbert Space there is the analogous idea for the orthonormal functions mentioned above

$$\langle \varphi_i, \varphi_j \rangle = \delta_{i,j} \tag{79}$$

The difference between (78) and (79) is that each φ_i in $\{\varphi_i\}_{i=1}^\infty$ is no longer a vector in a finite dimensional vector space but a function in a Hilbert Space, which can be thought of as an "uncountably infinite dimensional" vector. The term "uncountably infinite dimensional" comes from the fact functions in a Hilbert Space can be thought of as a vector with an uncountably infinite number of elements. Furthermore, the inner product for two (possibly complex) functions $f(x), g(x) \in \mathcal{L}_2(\Omega)$ is now defined as

$$\langle f, g \rangle = \int_{\Omega \in \mathbb{F}} f(x)\overline{g(x)}dx \tag{80}$$

Based on this, we can think of the inner product $\langle f(x), \varphi_i \rangle$ as the projection of

the function $f(x)$ onto the i^{th} orthonormal basis function φ_i, directly analogous to vector projection onto $\boldsymbol{e}_i$ in $\mathbb{R}^3$.

A key idea of Hilbert Space function representation is an uncountably infinite object, $f(x)$, can be represented by a countably infinite weighted sum of orthonormal basis functions $\{\varphi_i\}_{i=1}^{\infty}$, where the weighting is done by $\{\alpha_i\}_{i=1}^{\infty}$, with each $\alpha_i = \langle f(x), \varphi_i \rangle$. Just as a vector in $\mathbb{R}^N$ can be written as

$$\boldsymbol{r} = \sum_{i=1}^{N} \beta_i \boldsymbol{e}_i \tag{81}$$

where $\{\beta_i\}_{i=1}^{N}$ are the coordinates of $\boldsymbol{r}$ in $\mathbb{R}^N$, the sequence $\{\alpha_i\}_{i=1}^{\infty}$ above represents the "coordinates" of $f(x)$ on an orthonormal basis $\{\varphi_i\}_{i=1}^{\infty}$ in a Hilbert Space.

Some common families of orthonormal basis functions are the Legendre and Hermite Polynomials [35]. Once a particular family of functions is chosen, each φ_i in $\{\varphi_i\}_{i=1}^{\infty}$, for fixed domain $x \in [a, b]$, is uniquely determined. At this point the only parameters that differentiate functions in a Hilbert Space are the coefficients $\{\alpha_i\}_{i=1}^{\infty}$. Given that functions in a Hilbert Space are differentiated by their coefficients $\{\alpha_i\}_{i=1}^{\infty}$, we can think of different subspaces of Hilbert Space as the regions where different function families "live", where the different regions are defined by the different values of $\{\alpha_i\}_{i=1}^{\infty}$. This is shown in Figure 11, where for visualization purposes only two dimensions are shown. The regions representing exponential, sigmoidal, and sinusoidal functions individually are indicated by circles with their respective names. These three function families are subsumed under the more general family of Taylor polynomials. Experiments in Sections 4.2, 4.3, 4.4, 4.6, and 4.7 progressively train an AEKF and Hilbert Space Filter (introduced in Section 4.7) to perform state estimation on more general classes of functions in these families.

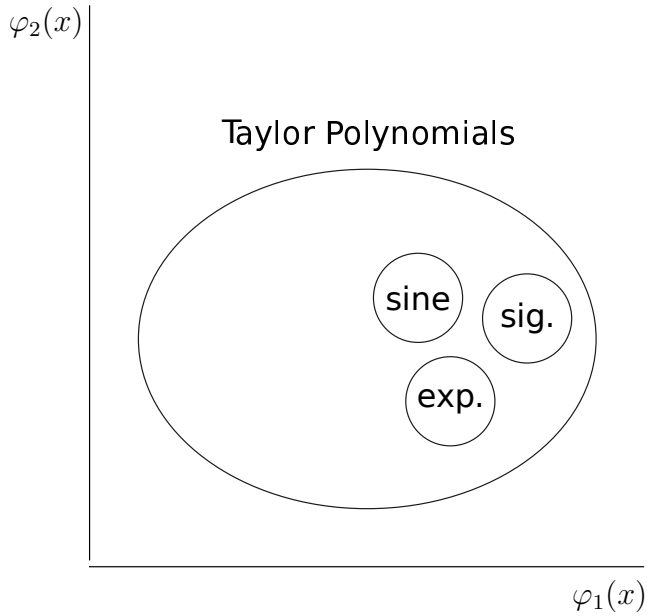


Figure 11: Curve families as occupying different regions of Hilbert Space. The region of Hilbert Space occupied by exponential, sigmoidal, and sinusoidal functions are each represented by the circle with the respective name. At a higher level of generality, these three families are each a subset of all functions representable by a Taylor polynomial. In Hilbert Space, each family is distinguished by the coefficients $\{\alpha_i\}_{i=1}^{\infty}$ projected onto the orthonormal basis functions $\{\varphi_i\}_{i=1}^{\infty}$.

3.2 Domain Randomization

One difficulty when training deep learning models is the requirement that large amounts of training data be made available. This requirement can be mitigated by the use of simulated data, which effectively allows for infinite training data. However, the differences between models trained on simulated data and the real world, known as the *reality gap*, are often difficult to overcome. Even when tuning parameters in simulation to match the real world, a technique known as *system identification*, physical quantities such as wear-and-tear and fluid dynamics often remain unmodeled [58].

Domain randomization [41, 52, 58] attempts to solve these problems and bridge the reality gap. The basic philosophy of domain randomization is that *in order to*

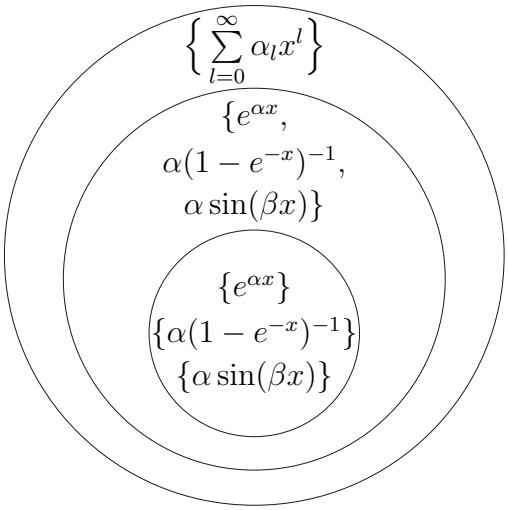


Figure 12: Figure depicting our progressive domain randomization scheme. The concentric circles represent a progressively larger subregion of a Hilbert Space, each including the smaller circles. The smallest circle represents all exponential, sigmoidal, and sinusoidal function families individually while the middle circle represents a single family consisting of all exponential, sigmoidal, and sinusoidal functions. The outermost circle represents the most general class of functions, Taylor Polynomials.

avoid overfitting a particular set of parameters in a deep learning model, one should randomize those parameters during training. If there is enough variability in the simulated model, the real world is just one among the many variations learned in simulation. Thus, the goal is to achieve a high enough degree of variability in simulation such that the real world being modeled is present among the variations. Of course, the range of parameter randomization will be bounded and informed by domain knowledge. The hope is that by randomizing training parameters, the learned model will be robust enough to perform well on test cases whose parameters fall within the range of the randomized training parameters. Put another way, domain randomization deals with the question of how to cover a particular parameter space by randomizing a parameter over its entire domain. Since the data is simulated, access to the ground truth *for training* is not problematic.

As an example, consider a neural network which learns a function $f(\mathbf{x};\boldsymbol{\theta}) : \mathbf{x} \mapsto \mathbf{y}$, where $\mathbf{x}$ is the input feature vector, $\mathbf{y}$ the target, and $\boldsymbol{\theta}$ a parameter vector to be learned that correctly maps $\mathbf{x}$ to $\mathbf{y}$. If we assume $\mathbf{x} \in \mathbb{R}^M$, $\boldsymbol{\theta} \in \mathbb{R}^N$, and $\mathbf{y} \in \mathbb{R}^P$, domain randomization consists of training the neural network on simulated data by randomly sampling a simulated input $\hat{\mathbf{x}} \in \mathbb{R}^M$, with known target $\hat{\mathbf{y}} \in \mathbb{R}^P$, each training epoch. The network is then trained to learn $\boldsymbol{\theta}$ such that a cost function, such as

$$\|\hat{\mathbf{y}} - f(\hat{\mathbf{x}};\boldsymbol{\theta})\|_F^2 \tag{82}$$

is minimized. If the randomization of $\hat{\mathbf{x}}$ during training covers $\mathbb{R}^M$ such that (82) is below a threshold $\epsilon > 0$, a model that has learned to successfully map $\hat{\mathbf{x}} \mapsto \hat{\mathbf{y}}$ during training, should generalize for a new input $\hat{\mathbf{x}} + \delta\hat{\mathbf{x}}$ and new target $\hat{\mathbf{y}} + \delta\hat{\mathbf{y}}$ such that

$$\|(\hat{\mathbf{y}} + \delta\hat{\mathbf{y}}) - f(\hat{\mathbf{x}} + \delta\hat{\mathbf{x}};\boldsymbol{\theta})\|_F^2 \approx \|\hat{\mathbf{y}} - f(\hat{\mathbf{x}};\boldsymbol{\theta})\|_F^2 \tag{83}$$

for $\delta\hat{\mathbf{x}}$ sufficiently close to $\hat{\mathbf{x}}$ and $\delta\hat{\mathbf{y}}$ sufficiently close to $\hat{\mathbf{y}}$.

In the context of this dissertation, our use of domain randomization can be thought of as training a neural network to perform state estimation for increasingly more general families of functions in Hilbert Space. That is, each of the concentric circles in Figure 12 represent a larger region of a Hilbert Space, each of which includes the contained smaller circles. The inner most circle represents exponential, sigmoidal, and sinusoidal function families separately. Moving to the next larger circle, a single family consisting of exponential, sigmoidal, and sinusoidal functions is depicted. Lastly, the outermost circle represents the family of Taylor Polynomials, which includes exponential, sigmoidal, and sinusoidal functions as a subset. Thinking of Hilbert Spaces in this way is useful for conceptualizing our use of domain randomization.

4 Applications and Analysis

In this section we discuss applications of the Kalman Filter, LSTM, AEKF, AEIMMKF, and Hilbert Space Filter to various state estimation problems along with a mathematical analysis of the AEKF. Section 4.1 addresses the application of the standard Kalman Filter to feature engineering in the context of time series classification. In Sections 4.2, 4.3, and 4.4 we address the question of robust filtering by applying the AEKF to time series filtering with a variety of curve families and noise types, within the context of domain randomization. Building on the success in Sections 4.2-4.4, in Section 4.5 we derive a theorem concerning the AEKF in the context of mitigating the effects of outlier measurements. In Section 4.6 we present an application of the AEIMMKF to target tracking in a simulated environment. Lastly, in Section 4.7 we present preliminary state estimation results based upon explicitly leveraging the functional representation properties of Hilbert Spaces.

The overarching narrative of our applications is depicted in Figure 13. Figure 13(a) is a ASR-9 air traffic control (ATC) radar and represents the traditional application of the Kalman Filter to tracking problems. In our work, we make a novel contribution to the chemical sensor community by applying the Kalman Filter outside this traditional setting. Specifically, the Kalman Filter is applied to the feature engineering of chemical sensor response time series data for downstream machine learning. Here we leverage the state estimation capabilities of the Kalman Filter to improve early detection and discrimination of chemical agents. This is represented by the chemical sensor array in Figure 13(b). Based upon the success here, and noting the limitations of the Kalman Filter discussed above, we combine the Kalman Filter with deep learning and domain randomization for the purpose of creating a generalized time series state estimation framework in Sections 4.2-4.4.

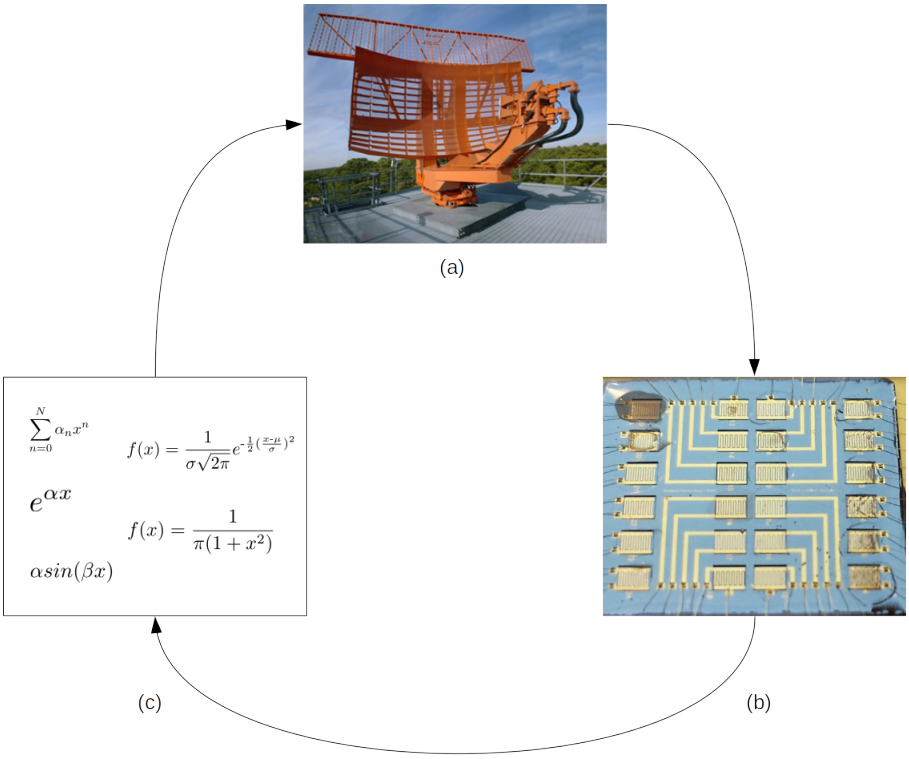


Figure 13: Traditionally the Kalman Filter is used for target tracking applications represented by the ASR-9 ATC radar in (a). In Section 4.1 we apply the Kalman Filter to feature engineering of chemical sensor array time series data for early detection of chemical agents, represented by the chemical sensor array in (b). Based on the success of the Kalman Filter in this context, we leverage deep learning, the Kalman Filter, and domain randomization to train the AEKF and AEIMMKF to filter families of functions of increasing generality, with various added noise types as depicted in (c). Finally, we come full circle by applying the AEKF and AEIMMKF to ATC tracking problems with simulated data. Source: Figure (a) https://commons.wikimedia.org/wiki/File:ASR-9_Radar_Antenna.jpg

This is depicted in Figure 13(c), which shows a variety of the function families and noise types we train the AEKF and AEIMMKF on. Lastly, returning to Figure 13(a), we train an AEKF and AEIMMKF on simulated flight paths demonstrating these algorithms are applicable in the context of ATC tracking problems. Thus, the applications in this dissertation begin with applying the Kalman Filter in a non-traditional setting, demonstrate the power of merging the Kalman Filter with

deep learning and domain randomization, and end with a simulated real-world application of the techniques developed herein.

4.1 Kalman Filter For Early Detection

The detection of chemical agents is an ever-present issue in a variety of application domains. Yet many of the existing chemical agent detection technologies are limited by size, power, and weight. In this context, low-powered miniaturized sensors with discrimination capabilities for numerous diverse chemicals are highly desirable [62]. In addition, if a sensor is only able to characterize known targets after a set exposure time, then the usefulness of the sensor is diminished in real-world scenarios.

To alleviate the difficulties of designing sensors under such constraints, advanced signal processing techniques may be used. Here optimizations in software allow strict constraints to be met in hardware. Thus, from a sensor design perspective, the Kalman Filter's low computational footprint makes it an ideal candidate for use in miniaturized, low resource sensors.

Furthermore, sensors which meet the above-mentioned design constraints are only maximally effective if they can alarm early, with low false alarm rates. Yet many of the standard features used for chemical classification with sensor time series data require data to be processed batch-wise with pre-determined exposure times. That is, all or a large portion of a time series must be considered, making early detection problematic. In this context, the sensor is only characterized for a scenario where the analyte exposure time is fixed and known. For example, in the case where maximum sensor response is used for classification, up to thirty seconds of the sensor time series data post-analyte exposure must be considered

[42, 43]. In other cases the entire time series is required such as when exponential curve fitting is applied to response and recovery regions, area of response and recovery regions are used as features [42, 43], and when feature engineering via dimensionality reduction with PCA is performed on the entire time series [62].

The work in this section proposes a solution to the design constraint and early detection problem by demonstrating early and accurate chemical analyte detection is possible when chemical sensor time series data is preprocessed by a Kalman Filter. As will be shown, the use of filtered first derivatives estimated with the Kalman Filter are the key to this procedure. The decision to use filtered first derivatives, along with the filtered first-order data, is based on the assumption when analytes are first introduced to a sensor the rate at which the sensor reacts is more relevant to classification.

We emphasize that the ideas proposed in this section make a unique contribution to the chemical sensor community in two ways.

- We apply the Kalman Filter to the preprocessing of chemical sensor time series data for the purpose of feature engineering for downstream machine learning, where the Kalman Filter is used to estimate the first derivatives of chemical sensor time series. As a result, classification error rates less than 10% are achieved within two seconds of sensor exposure to an analyte. Under the same conditions non-Kalman Filtered and moving average filtered datasets take more than twice as long to achieve the same result.

- We introduce the Kalman Filter as an additional tool for use in the preprocessing of chemical sensor time series data, not limited to machine learning. In this context, the Kalman Filter has an advantage over more traditional preprocessing methods, such as standardization and moving average. This

advantage stems from the fact the Kalman Filter allows greater control over the balance between smoothing and reaction time of chemical sensor time series data.

4.1.1 Dataset

The dataset used for our analysis and validation was comprised of data collected from real sensors in a laboratory setting and is the same as used in [62]. Data collection consisted of exposing a twelve-sensor array of polymer-graphene nanoplatelet coated sensors to the following eight interferents and five organophosphates: acetone, antifreeze, diesel, ethanol, hexane, Round Up, toluene, water, dimethyl ethylphosphonate (DEMP), diisopropyl methylphosphonate (DIMP), dimethyl methylphosphonate (DMMP), triethyl phosphate (TEP), and trimethyl phosphate (TMP). For each analyte 100 trials were performed with each trial consisting of three regions: baseline, response, and recovery. The baseline region lasted sixty seconds where no analyte was present. The response region was initiated by the introduction of the analyte and lasted for thirty seconds. The recovery region then ran for 180 seconds during which time the analyte was purged from the sensor apparatus. The twelve polymer-graphene nanoplatelet coatings consisted of the following polymers: polycaprolactone (PCL), poly(4-vinylphenol-co-methyl methacrylate) (PVPH-MMA), polyvinyl alcohol (PVA), polyisobutylene (PIB), poly(1-vinylpyrrolidone)-graft-(1-triacontene) (PVPyd-gT), nafion, polyepichlorohydrin (PECH), poly(vinylphosphonic acid) (PVPA), polyacenaphthylene (PACN), polytetrafluoroethylene (PTFE), poly(ethylene-co-vinyl acetate) (PEVA) and poly(4-vinylphenol) (PVPH). In Figure 13(b) an actual twenty-four-sensor array is shown, where twelve of the twenty four sensors are coated with the above twelve polymers. Although each of the uncoated sensors

in the array are the same, once each are coated with a polymer they effectively become unique sensors. That is, given exposure to the same chemical, sensors coated with different polymers will produce different responses. The details on where these analytes and polymer coatings were acquired can be found in [62].

The sampling frequency for the sensor time series was 15 Hz, resulting in approximately 4040-4050 samples per 270 second time series. For consistency, 4000 samples were used for each trial during classification. In terms of classification accuracy this has no substantial effect as the majority of classification occurs at the beginning of the response region. The final dataset was a $1300 \times 4000 \times 12$ tensor where 1300 is the total number of trials, 4000 the number of samples per trial, and 12 the number of sensors in the sensor array.

4.1.2 Data Preprocessing

In machine learning applications one of the most important steps is data preprocessing. The importance rests on the fact that the ultimate measure of a machine learning model is performance on, or generalization to, a yet unseen future dataset. To this end, successful generalization depends on the correct "definition" of distance. Since only one dataset was available it was essential to determine a well-principled data preprocessing scheme to maximize the possibility the techniques presented here would generalize on future datasets. Furthermore, this method is not limited to chemical sensors but applicable to any time series data.

Various techniques for preprocessing chemical sensor data exist in the chemical sensing literature. These include global scaling methods such as z-scoring (or autoscaling) and mean centering along with local scaling methods such as vector normalization[22]. Additionally, techniques for baseline drift correction such

as differential, relative, and fractional techniques are common [22]. In this section preprocessing consisted of the following three steps: vector normalization, z-scoring, and Kalman Filtering.

With twelve sensors, each of the 4000 samples in a trial can be thought of a point in twelve-dimensional space. As a result, each trial is a dataset with dimensions $\mathbb{R}^{4000\times12}$, where 4000 represents the number of samples and 12 the number of sensors. Representing the data set by $\mathbf{x} \in \mathbb{R}^{4000\times12}$, $\mathbf{x}_{ij}$ represents the entry in the i^{th} row and j^{th} columns. Vector normalizing the trials normalizes each 12-dimensional sample (row) to a unit vector by dividing each of the twelve sensor responses by the $\ell2$, or Euclidean, norm of the sample according to

$$\hat{x}_{ij} = \frac{x_{ij}}{\sqrt{\sum_{j=1}^{12} x_{ij}^2}} \tag{84}$$

where $\hat{x}_{ij}$ is the vector normalized component, x_{ij} the original or untransformed component, and $\sqrt{\sum_{j=1}^{12} x_{ij}^2}$ the $\ell2$-norm of the i^{th} row. In geometrical terms, this is a non-linear projection that maps a sample in $\mathbb{R}^{12}$ onto the surface of a unit hypersphere in $\mathbb{R}^{12}$. Projecting the data in this way reduces the effects of analyte concentration and is suggested for classification problems, under the assumption all sensors have the same concentration dependence [22]. For this to be a well principled approach, we make a second assumption that varying the concentration by a factor of α should only scale the 12-dimensional sample vector by a factor of α and not rotate the vector.

Following vector normalization, each feature was z-scored according to the formula:

$$\tilde{x}_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \tag{85}$$

where $\tilde{x}_{ij}$ is the z-scored component, x_{ij} the original or untransformed component, and μ_j and σ_j the mean and standard deviation of the j^{th} sensor (column) respectively.

Although it is a common practice to z-score such that each column has zero mean and unit standard deviation, our method was to use a subset of each sensor time series to determine the mean and standard deviation used in z-scoring. This subset was the ten seconds (150 samples) immediately before analyte introduction. The reasons for this are twofold.

First, using the entire time series to compute the sensor mean and standard deviation would cause each of the twelve sensors in a given trial to be shifted by the mean of the *entire response* and not just the mean of the baseline region. As a result, the sensor readings in the baseline region would be shifted by a value different from the baseline drift and baseline drift would not be corrected. Secondly, calculating the sensor mean and standard deviation using the entire time series takes the sensor response to the analyte into account. In our context, z-scoring is a global transformation which attempts to correct for noise inherent in the sensors, unrelated to their particular analyte response. Thus, the most principled approach was to z-score only during the very end of the baseline period, approximating the sensor's true baseline mean and standard deviation as closely as possible. Additionally, the last ten seconds before analyte introduction were used to extend the recovery region in the event the previous analyte had not completely desorbed from the sensor. Note that all references to z-scored data below refer to data that has been first vector normalized and then z-scored.

Lastly, the Kalman Filter was applied to the z-scored data, generating Kalman

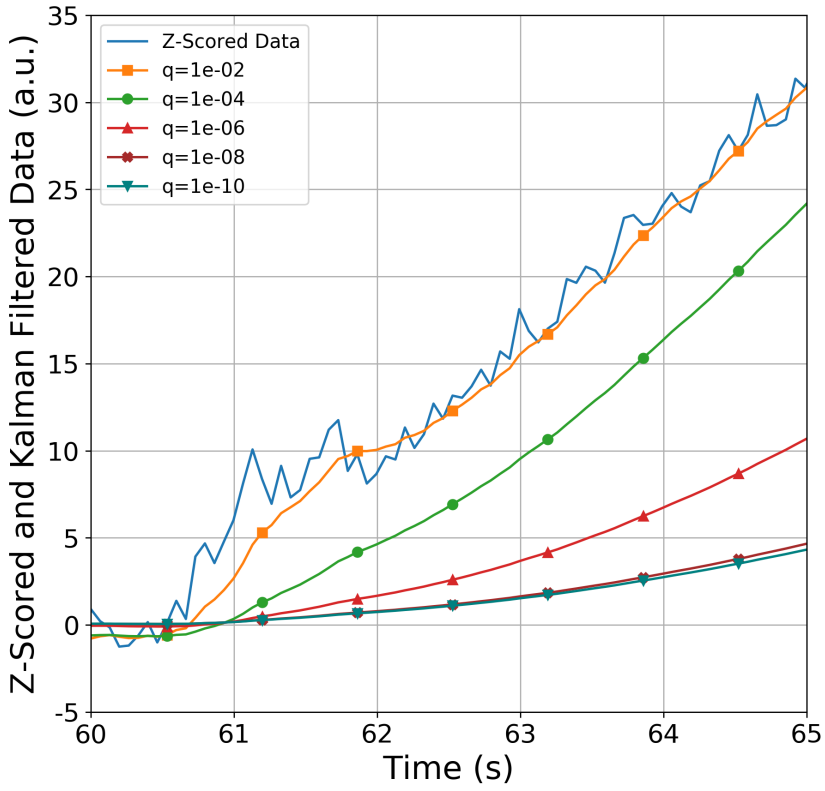


Figure 14: Comparison of smoothing vs. reaction time for Kalman Filtered data for the first five seconds after analyte introduction, across six process noise covariance matrices $\mathbf{Q}_k$ (with measurement noise covariance matrix $\mathbf{R}_k$ fixed as the identity matrix $\mathbf{I}$). The q values in the legend indicate the diagonal elements of $\mathbf{Q}_k$, which can be expressed as $\mathbf{Q}_k = q\mathbf{I}$. The blue response represents the z-scored data and the orange response, corresponding to the largest q value, follows the z-scored response closest and reacts the quickest. This is due to the fact the largest q value gives the least weight to the dynamical model and hence more weight to the measurements than the other filtered responses with smaller q values. At the other extreme, the responses with the two smallest q values are the smoothest but react slowest since they give the most weight to the dynamical model. Although only one q value greater than 1e-08 is shown, for $q \geq$ 1e-08 the Kalman Filtered signals overlap and show no improvement in filtering.

Filtered and Kalman Filtered first derivative datasets. The first derivatives follow from the fact we are using the NCV model in the Kalman Filter. Referring back to (27), if we ignore the second derivative term, the Kalman Filter's state estimate for the NCV model is given by

$$\begin{bmatrix} \hat{\mathbf{x}}_{k|k} \\ \hat{\dot{\mathbf{x}}}_{k|k} \end{bmatrix} \tag{86}$$

where we see that both the state and state's first derivative are estimated.

The decision to Kalman Filter after z-scoring is based on the fact it is computationally more efficient to apply the Kalman Filter to data having the same units. The reason for this is the optimal values of both $\mathbf{Q}_k$ and $\mathbf{R}_k$ may depend on the units of the data processed by the Kalman Filter. If different sensors used different units, it would have been necessary to determine the optimal $\mathbf{Q}_k$ and $\mathbf{R}_k$ individually for each sensor. As the Kalman Filter is a recursive algorithm it is more computationally expensive than the z-scoring in (85). Therefore, the most principled approach was to Kalman Filter the z-scored dimensionless data, which would allow using one value of $\mathbf{Q}_k$ and one value of $\mathbf{R}_k$ for all trials and sensors. Furthermore, vector normalizing after z-scoring would result in the sensor response signal not having zero mean and unit standard deviation in the ten second region immediately before analyte introduction used to calculate each μ_j and σ_j. As a result, baseline drift would not be corrected.

As mentioned above, an important feature of the Kalman Filter is the ability to control the inverse relationship between sensor response filtering and reaction time. This relationship is controlled by the values of the process noise covariance matrix $\mathbf{Q}_k$ and measurement noise covariance matrix $\mathbf{R}_k$ mentioned previously. Here $\mathbf{Q}_k$ and $\mathbf{R}_k$ are diagonal matrices, with their respective diagonal values rep-

resented by q and r. Thus, we can represent the matrices as $\mathbf{Q}_k = q\mathbf{I}$ and $\mathbf{R}_k = r\mathbf{I}$ where $\mathbf{I}$ is the identity matrix. Here it is the ratio of q and r which determines the balance between smoothing and reaction time. Figure 14 demonstrates this inverse relationship, all in relation to the z-scored response. Here the q value in the legend indicates the value used for the diagonal elements in $\mathbf{Q}_k$. For all trials, $r = 1.0$. Smaller diagonal entries of $\mathbf{Q}_k$, relative to r, indicate greater confidence in the Kalman Filter's dynamical model than the measured value. This naturally leads to a smoother curve since the Kalman Filter is not as influenced by the measurements as it is by the dynamical model. As a result, the filtered signal will not react as quickly, as it needs to see more measurements before it is pulled away from the linear model assumed by the Kalman Filter. Based on cross-validation comparison of classification accuracy, the value of $q = 1e\text{-}08$ was used during classification.

It should be emphasized the reason the Kalman Filtered signal does not follow the z-scored signal after analyte introduction in Figure 14 is a consequence of choosing the parameters q and r to maximize classification accuracy and is not a defect of the Kalman Filter. It is important to keep in mind the Kalman Filter was used as a tool for feature engineering a filtered dataset and filtered first derivatives dataset for early and accurate analyte detection, specifically the filtered first derivatives, as these produced the most accurate early analyte classification as discussed below. The fact the chosen values of $q = 1e\text{-}08$ and $r = 1.0$ maximize classification accuracy is an indication that applying the Kalman Filter with these parameters retains the features in the data necessary for accurate classification, while removing noise that would otherwise contribute to misclassification. In other contexts, $\mathbf{Q}_k$ and $\mathbf{R}_k$ would be chosen differently as discussed in this section's summary.

Figure 15 shows the time series response of a sensor coated with PCL exposed to the organophosphate TEP for both the z-scored and Kalman Filtered data. The larger plot shows both signals ten seconds before and after analyte introduction whereas the inlay plot shows the entire time series. The ten seconds after analyte introduction are significant as all machine learning was performed on data within this window.

4.1.3 Machine Learning

The machine learning pipeline we used consisted of five steps: creating bins of the full time series, reshaping the dataset, generating folds for cross validation, dimensionality reduction with PCA, and classification.

In order to perform *early detection*, bins, or subsets, of the full sensor time series were created. These bins consisted of data from the first N-seconds after analyte introduction for all twelve sensors. In total, ten bins were generated, with the first bin containing one second of data after analyte introduction and each subsequent bin increasing by one second. That is, the second bin was two seconds of data after analyte exposure, the third was three seconds after analyte introduction, and so on. As an example, in the case of a 2-second bin, the mapping from the original dataset to the bin was $\mathbb{R}^{4000\times12} \rightarrow \mathbb{R}^{30\times12}$; where the thirty samples in the bin resulted from the fact that the sampling frequency during data acquisition was 15 Hz (15 samples per second $\times$ 2 seconds $= 30$ samples).

To reduce variance, or the possibility of overfitting, 5-fold stratified cross validation was performed, resulting in five 80/20 training/testing splits. The advantage of this method is that each data point in the dataset is included in fitting and evaluating the machine learning model.

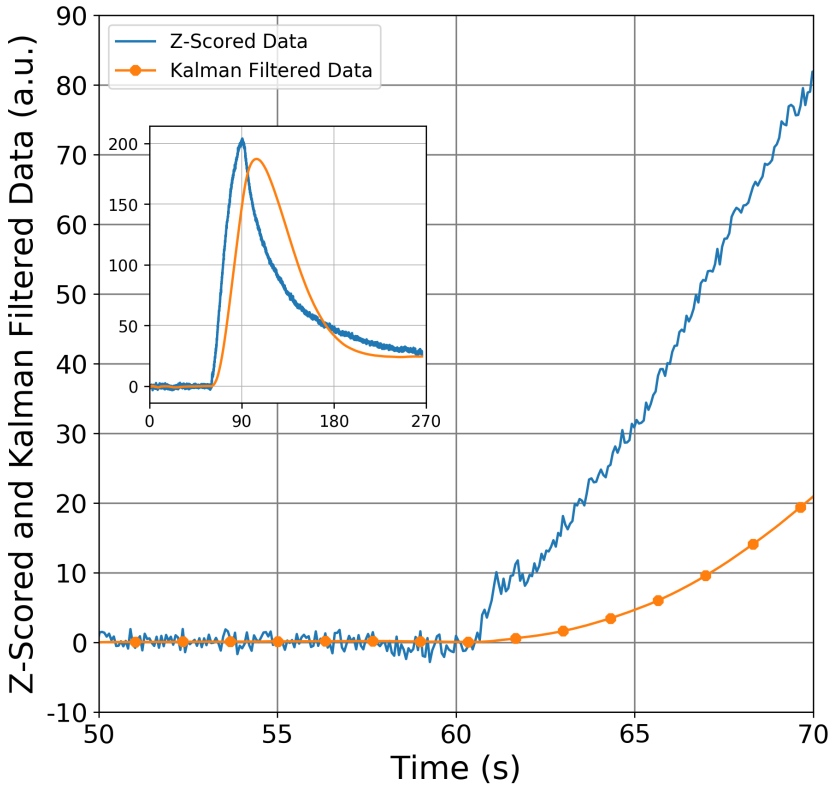


Figure 15: Example of Kalman Filter applied to organophosphate TEP on a sensor coated with PCL. The noisy blue signal is the z-scored sensor data whereas the orange signal is the same data after application of the Kalman Filter. The larger plot displays both responses ten seconds before and after analyte introduction whereas the inlay plot depicts the entire time series. The Kalman Filtered signal not following the z-scored signal closely after analyte introduction is a consequence of selecting the Kalman Filter parameters $\mathbf{Q}_k = 1e\text{-}8\mathbf{I}$ and $\mathbf{R}_k = \mathbf{I}$ to maximize classification accuracy. Note the values along the y-axis are dimensionless since both responses were z-scored.

PCA was used to reduce the dimensionality of the dataset and find the linear combinations of features which capture the most variance in the dataset. To avoid data from the training set influencing the testing set, the PCA model was fit using only the training data after which both the training and testing data were projected onto the principal components. In all machine learning models, the first twenty principal components of the respective data sets were used. The number of principal components was chosen using cross validation and a support vector classifier (SVC) [6, 8] with 20 principal components achieved the lowest classification error on the z-scored data set among all combinations of classifiers and principal components. To make a fair comparison, the z-scored data was used to ensure we did not select the number of principal components to maximize the accuracy of Kalman Filtered data, while possibly decreasing accuracy of the z-scored data.

The final step was to evaluate the classification accuracy of each dataset by machine learning. Three machine learning models were selected for classification: k-nearest neighbors (KNN) [12, 14], SVC and linear discriminant analysis (LDA) [13], and implemented in Scikit Learn [47], a Python machine learning library. These three were chosen because they are (a) standard machine learning models and (b) make different assumptions about the dataset used for classification. KNN makes no assumption about the statistical distribution of the data, but it does assume that the distance between points in the dataset is a predictor for classification. SVC is a linear classifier and will perform better on linearly separable data, whereas LDA, also a linear classifier, adds the additional assumption the dataset was drawn from a Gaussian distribution. For a modern treatment of the above classification algorithms see *The Elements of Statistical Learning* [20] with references therein. All programming was done in Python, drawing heavily on the

Numpy [60], Scikit Learn, and Matplotlib [26] libraries.

For KNN, the 5 nearest neighbors were used with uniform weighting on the distances. For SVC, the penalty parameter C was set to 1.0 which is the default value used in Scikit Learn. For each of the five cross-validation folds, all three machine learning models were fit with the training set, and model evaluation was computed on the testing set using accuracy as a metric. For each of the ten bins the test error, given by the equation $error = 1.0\text{-}accuracy$, for the z-scored dataset, moving average dataset, Kalman Filtered dataset, and Kalman Filtered first derivatives dataset were compared across all three machine learning models. For the moving average, the average was calculated using the five preceding data points. In the context of early detection, this was the most principled approach as using an equal number of points before and after the central value would require waiting for the data after the central value.

As our algorithm involves Kalman Filtered data being passed to machine learning algorithms, it is important to consider error propagation between the Kalman Filter and machine learning algorithms. Since the Kalman Filter is smoothing the data, presumably the filtered data has less error than the unfiltered data. The best measure of how the Kalman Filter's error propagates is determined by comparison of classification accuracy on the filtered and unfiltered datasets discussed below.

4.1.4 Results

In terms of early detection, both Kalman Filtered datasets achieved lower classification error than the z-scored and moving average datasets for the first five bins, with lowest error achieved on the Kalman Filtered first derivatives dataset. Among the three classifiers, SVC achieved the best results, with the Kalman Filtered first derivative dataset

		Bin Classification Error (%)				
		Bin 1	**Bin 2**	**Bin 3**	**Bin 4**	**Bin 5**
KNN	ZS	74 +/- 3	56 +/- 3	46 +/- 3	23 +/- 2	10 +/- 1
	MA	76 +/- 2	56 +/- 2	47 +/- 3	25 +/- 2	11 +/- 1
	KF	40 +/- 3	33 +/- 4	25 +/- 2	25 +/- 3	21 +/- 4
	KF1D	28 +/- 3	20 +/- 2	14 +/- 1	12 +/- 1	10 +/- 1
LDA	ZS	62 +/- 4	48 +/- 2	40 +/- 4	23 +/- 2	14 +/- 2
	MA	62 +/- 3	48 +/- 2	42 +/- 2	24 +/- 2	14 +/- 1
	KF	25 +/- 3	20 +/- 4	18 +/- 3	18 +/- 1	14 +/- 1
	KF1D	**13 +/- 2**	**9 +/- 2**	9 +/- 1	9 +/- 1	8 +/- 1
SVC	ZS	60 +/- 3	43 +/- 3	29 +/- 2	13 +/- 1	7 +/- 1
	MA	62 +/- 3	43 +/- 3	32 +/- 2	17 +/- 1	7 +/- 2
	KF	25 +/- 4	19 +/- 3	14 +/- 3	12 +/- 2	10 +/- 2
	KF1D	14 +/- 2	**9 +/- 2**	**6 +/- 1**	**5 +/- 1**	**5 +/- 1**

Table 2: KNN, LDA and SVC classification error for z-scored (ZS), moving average (MA), Kalman Filtered (KF) and Kalman Filtered first derivative (KF1D) datasets for each of the five bins.

passing below the 10% classification error threshold two seconds after analyte exposure and achieving non-overlapping standard deviations with the z-scored and moving average datasets for the first four bins. The results from SVC are shown in Figure 16. For full comparison, results from all three classifiers are shown in Table 2.

In order to better understand these results, and the trade-off between signal smoothing and reaction time, it is necessary to focus on the processes noise and measurement noise covariances $\mathbf{Q}_k$ and $\mathbf{R}_k$. Figures 17(a) and 17(b) show the time series response of three sensors for a single trial of TEP five seconds before and after analyte introduction for both z-scored and Kalman Filtered data. From the perspective of machine learning, the Kalman Filtered data in Figure 17(b) produces better classification results since it is less noisy. Whereas not all sensors react at the same time, the unique response of just one sensor to a given analyte is enough for accurate classification. Of the three sensors, the two coatings that stand out

as reacting quickly to TEP are PCL and PVPH. Noting the black horizontal line at sixty seconds indicates the analyte introduction point, both of these sensors show a sharp change in their response within a second after analyte exposure. However, this comes at a cost. Whereas both the z-scored and Kalman Filtered signals react within the first second, noting the different y-axis scales in Figures 17(a) and 17(b), we see the z-scored signal grows much more quickly. Referring back to Figure 15 shows this clearly, as both z-scored and Kalman Filtered signals appear on the same figure. An explanation for this difference in y-axis scales is given below.

Relatedly, some justification for using an NCV dynamical model in the Kalman Filter is necessary. Given the sensor time series is non-linear in time, how can the NCV model accurately perform state estimation and capture the important features of the sensor response? The answer is based upon two factors: (1) *locally* the time series can be approximated as linear and (2) the Kalman Filter works by finding a balance between the actual data being filtered and the dynamical model. Related to the first point, the validity of applying a linear model to a non-linear signal depends on the sampling rate from which the signal was generated. Although mathematically a limit can be taken to an arbitrarily small size, in the case of chemical sensors this size is restricted by the time between samples since the dataset is discrete. If the sample rate is small enough, approximating the time series as piecewise linear between samples is an accurate model. In our case, the sampling frequency was 15 Hz which was small enough to justify modeling the time series as linear between samples. Additionally, visual inspection of the sensor response curves indicate higher order derivatives are very small. This gave further confidence that the linearization provided a principled estimate of the true signal. The decision not to use an Unscented Kalman Filter [28] was based on the

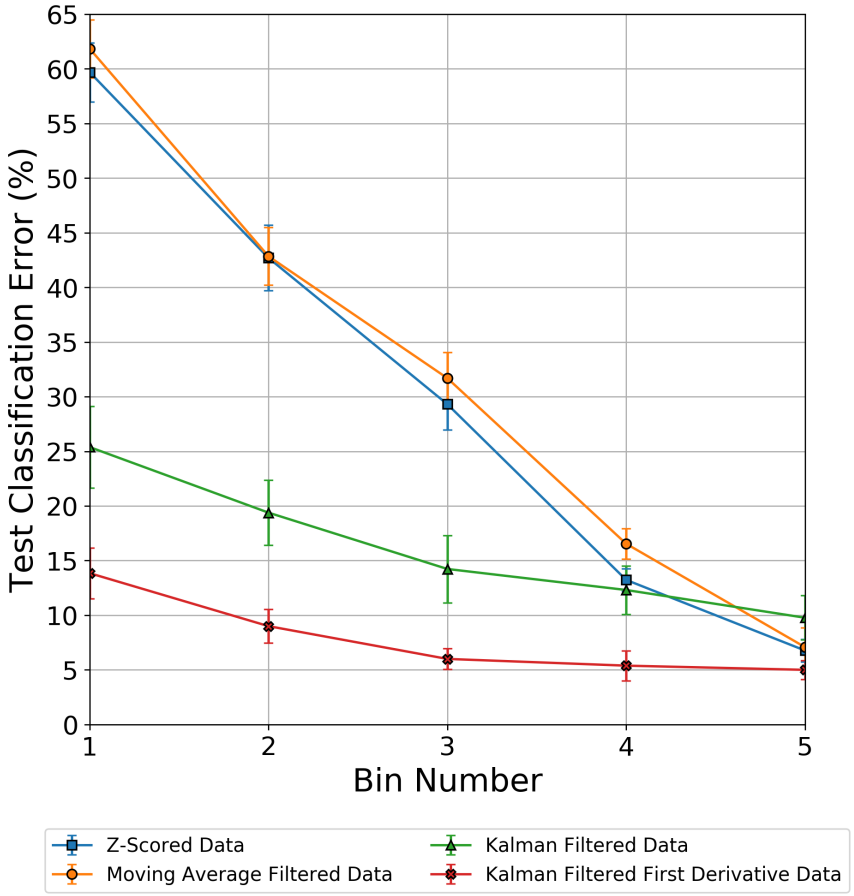


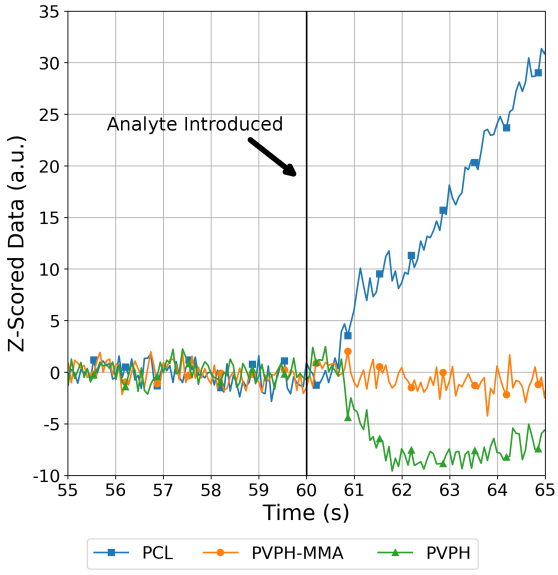
Figure 16: Support vector classifier test error for z-scored, moving average, Kalman Filtered, and Kalman Filtered first derivative datasets for each of the first five bins. The error bars represent the test error standard deviation across the five cross validation folds. Lower test error is achieved for both Kalman Filtered datasets for each of the first four bins, with the Kalman Filtered first derivatives dataset achieving an error of less than 10% by the second bin.

fact the present model worked well, as indicated by the results. However, the application of the methods presented here to more complex data may require the use of an Unscented Kalman Filter. Secondly, since the Kalman Filter finds a balance between the data and dynamical model, if a more flexible non-linear dynamical model was used the Kalman Filter would follow the data too closely, resulting in the Kalman Filter fitting to noise, as well as the underlying signal.

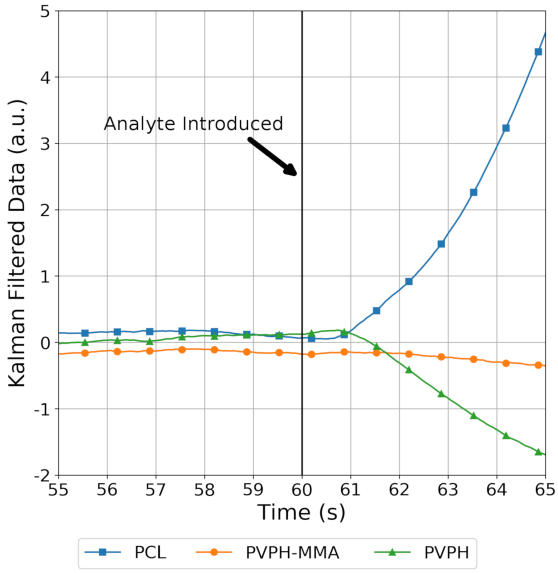
This smoothing-reaction trade-off is one of the main points of our research and deserves further discussion. Consider the z-scored and Kalman Filtered signals in Figures 17(a) and 17(b). Up until the sixty second mark, when the analyte is introduced, the Kalman Filtered signal is flat while the z-scored signal oscillates about the baseline. After analyte introduction, when the z-scored signal begins to change, the Kalman Filtered signal does not react immediately as it has not seen enough data to indicate the change is not a result of noise. Only after seeing several z-scored data points indicating an upward trend does the Kalman Filtered signal turn upwards. This adaptability is one of the Kalman Filter's most important features. Although the dynamical model is linear, the Kalman Filter is flexible enough to adapt to changes presented by the data without being influenced too quickly by noise in the data. Conversely, a more flexible non-linear dynamical model in the Kalman Filter increases the chances of overfitting the data by fitting too closely to the noise. It is this balance between a linear dynamical model and adaptability to empirical data that makes the Kalman Filter such a powerful tool.

4.1.5 Summary

In this section we demonstrated the Kalman Filter is a valuable machine learning preprocessing tool in the context of accurate and early analyte classification with



(a)



(b)

Figure 17: (a) displays the z-scored time series for PCL, PVPH-MMA and PVPH coated sensors five seconds before and after exposure to TEP, with the vertical line at 60 seconds indicating when the analyte was introduced. (b) displays the same scenario after applying the Kalman Filter to the same dataset. From the perspective of machine learning, the less noisy data will produce better classification results. Whereas the PCL and PVPH sensors react within one second, the PVPH-MMA sensor is much slower to react. Note, the y-axes are scaled differently.

chemical sensor time series data. As a recursive filter, the Kalman Filter is able to process data online and construct classification features in real-time. This would not be possible if a predetermined feature engineering scheme, such as maximum resistance change, area of time series curve, or a known sensor exposure time was required for classification. As a result, the Kalman Filter is an excellent candidate for an advanced signal processing technique which allows strict power requirements and concept of operations to be met in hardware. This is particularly true in remote or low resource sensing environments.

Furthermore, classification with the Kalman Filtered datasets outperformed datasets constructed with more traditional preprocessing and filtering methods such as z-scoring and moving average filter. In particular, for the first five bins, Kalman Filtered first derivatives datasets were shown to outperform the z-scored and moving average datasets using KNN, LDA, and SVC classification algorithms. Amongst the two Kalman Filtered datasets, SVC classification of the Kalman Filtered first derivatives dataset demonstrated the lowest classification error, passing below 10% classification error after only two seconds of data post analyte exposure. After five seconds post analyte exposure, all datasets showed no significant differences in analyte classification error. Although the particular focus of our research is on an array of semi-selective chemiresistive vapor sensors, we expect the results herein are applicable to a wide variety of chemical sensor types.

In this section, the Kalman Filter was used as a preprocessing tool for downstream machine learning. As such, the values of the process noise and measurement noise covariances were chosen to minimize classification error. For other non-machine learning applications, the balance between process noise and measurement noise covariances may be motivated by different metrics. For example,

in the determination of binding kinetics, Kalman Filtered data will provide less signal noise and subsequently more accurate fit of kinetic models. Hence, the values of $\mathbf{Q}_k$ and $\mathbf{R}_k$ would be determined using a different criterion than classification accuracy.

4.2 Autoencoder-Kalman Filter with Sequence Length

In addition to its application to chemical sensors in Section 4.1, the Kalman Filter is one of the most widely used algorithms in signal processing and has seen numerous applications in fields such as target tracking [46] and financial data processing [36]. However, as alluded to above, one of the greatest difficulties in implementing the Kalman Filter is the tuning, or estimation, of the measurement noise covariance matrix $\mathbf{R}_k$ [38, 40].

Traditionally, the optimal value for $\mathbf{R}_k$ is tuned using cross-validation. In these contexts, the determination of a *good* result is often based on experience and the domain knowledge of engineers [38]. In Section 4.1, $\mathbf{R}_k$ was tuned by cross-validation with classification accuracy as the metric. The approach in this section is to train the AEKF with domain randomization, leveraging deep learning to learn the values of both $\{\mathbf{z}_k\}_{k=1}^N$ and $\{\mathbf{R}_k\}_{k=1}^N$, where a measurement noise covariance matrix is learned for each measurement, as opposed to a single $\mathbf{R}_k$ for the entire sequence $\{\mathbf{z}_k\}_{k=1}^N$. Furthermore, since the AEKF combines deep learning and the Kalman Filter, the AEKF is compared with both an LSTM and standard Kalman Filter.

4.2.1 Simulation Environment

One possible application of the AEKF is to chemical sensor time series data, in much the same way a standard Kalman Filter was applied in Section 4.1. In this

section, we simulate sensor responses using the sigmoidal family of functions. Thus, the simulated dataset consists of sigmoidal curves, constituting the ground truth, $\boldsymbol{\phi}_g$, with added noise, $\boldsymbol{\phi}_n$, to create the simulated measurement data $\boldsymbol{\phi} = \boldsymbol{\phi}_g + \boldsymbol{\phi}_n$. The curves representing $\boldsymbol{\phi}_g$ were given by

$$\boldsymbol{\phi}_g(x; \alpha, \beta, \gamma) = \frac{\alpha}{1 + e^{-\beta(x-\gamma)}} \tag{87}$$

with $x \in [0, 100]$. The randomization occurred over the parameter α, with fixed values of $\beta = 0.15$ and $\gamma = 60.0$. With fixed β and γ, α was sampled uniformly such that $\boldsymbol{\phi}_g(x; \alpha, \beta, \gamma)$ was in $[0, 100]$. An additional ten samples were added to the beginning of each curve to simulate a zero-mean baseline preceding the simulated sensor response.

4.2.2 Noise

The simulated noise added to the training curves consisted of Gaussian noise, bimodal noise and noise from actual sensors. Gaussian noise was chosen based on the Kalman Filter optimality conditions which are met when Gaussian noise is present. In order to test the AEKF using non-Gaussian noise, bimodal noise was selected. Lastly, to include real-world noise in our evaluation, sensor noise was included.

Gaussian Noise The Gaussian noise was sampled from a standard Gaussian distribution $\mathcal{N}(0, 1)$.

Bimodal Noise In general, a multimodal or mixture distribution is a continuous probability distribution with two or more modes. The general equation for the

PDF of a mixture model is

$$f(x) = \sum_{i=1}^{N} w_i f_i(x) \tag{88}$$

where $\sum_{i=1}^{N} w_i = 1$ and $f_i(x)$ is the PDF of the i^{th} distribution in the mixture model for $i = \{1, 2, \ldots, N\}$. In our case, to generate non-Gaussian noise, we used a bimodal mixture model comprised of two equally weighted Gaussian distributions, resulting in the bimodal mixture model PDF

$$f(x) = \frac{1}{2} f_1(x) + \frac{1}{2} f_2(x) \tag{89}$$

Sampling from this bimodal distribution proceeds by randomly drawing each sample from either $f_1(x)$ or $f_2(x)$, with $\frac{1}{2}$ probability of drawing from either distribution. Although $f_1(x)$ or $f_2(x)$ are both Gaussian, the resulting sample drawn from the bimodal distribution is not Gaussian. Thus, adding this sampling to the ground truth results in a simulated signal with non-Gaussian noise. The two Gaussian distributions used to generate the bimodal noise had means -3.0 and 3.0 and a common covariance of 1.0

Sensor Noise In the case of actual sensor noise, 1,000,000 sensor baseline noise measurements were generated using chemiresistive vapor sensors. These sensors consisted of commercial interdigitated electrode arrays with 10 micron wide gold electrodes spaced 10 microns apart on a glass substrate. The electrodes were coated with a graphene nanoplatelet-polymer film by airbrushing the dissolved mixture on a hotplate with the pattern defined by a shadow mask. For each randomly generated training curve, the added sensor noise was randomly sampled from the 1,000,000 sensor baseline noise measurements in a non-contiguous manner.

4.2.3 Model Parameters

Kalman Filter Recall from 2.1 the Kalman Filter has four matrices of primary importance: $\mathbf{F}_k, \mathbf{Q}_k, \mathbf{H}_k$, and $\mathbf{R}_k$. In terms of $\mathbf{F}_k$, our tests utilized the NCV (24) and NCA (25) models. Recall $\mathbf{F}_k$ in (4) represents how the model projects the *a posteriori* estimate at time k-1, forward one step to time k. The NCV model assumes this projection is linear in time and the NCA assumes it is quadratic in time. In both NCV and NCA models $dt = 1.0$, as this is the reciprocal of the sampling frequency of 1 Hz for the simulated curves which consisted of 110 samples (100 response samples and 10 zero-mean baseline samples). Note that the dimensions of $\mathbf{H}_k$ and $\mathbf{Q}_k$ are dependent on the chosen model although, in our case, the values were not. For $\mathbf{H}_k$ we assumed the mapping was identity and for $\mathbf{Q}_k$ we assumed a diagonal matrix. Note that $\mathbf{R}_k$ is learned by the AEKF. However, it is important to note that in the AEKF's Kalman Filter layer, both $\mathbf{z}_k$ and $\mathbf{R}_k$ are in $\mathbb{R}^1$ for all $k = \{1, 2, \ldots, N\}$. This restriction is due to the difficulty of learning a non-singular $\mathbf{R}_k$ as $\mathbf{S}_k$ must be invertible in (8). This difficulty was eventually overcome and forms the basis of Section 4.3.

Autoencoder-Kalman Filter The AEKF consisted of input and output layers, three hidden layers in both the encoder and decoder, the affine transformations before and after the Kalman Filter, and the Kalman Filter layer itself, corresponding to (59)-(70). During AEKF training, each epoch consisted of passing a single simulated sensor response, $\phi = \phi_g + \phi_n$, to the AEKF with weights updated after each epoch. As the input has a feature space of 1, each $\mathbf{R}_k$ in $\{\mathbf{R}_k\}_{k=1}^N$ learned by the AEKF was a scalar. It should be emphasized, because domain randomization was used, that *each epoch a new simulated ground truth curve and a new random draw from*

the given distribution were generated. Thus, it is highly unlikely the AEKF ever saw the same training curve or noise sample more than once.

To include a feature analogous to sequence length in an LSTM (discussed below), we introduce the notion of sequence length for an autoencoder. Considering a standard autoencoder with an input $\boldsymbol{\phi} \in \mathbb{R}^{N\times 1}$, an affine transformation of each of the N inputs composed with a non-linear transformation is mapped to each of the dimensions in the subsequent layer. Here the dimension of the input feature space is 1. However, in the case of time series data, where the past has some relation with the future, the autoencoder's reconstruction of the k^{th} element of the time series, $\boldsymbol{\phi}_k$, may be improved if some entries preceding $\boldsymbol{\phi}_k$ are included in the row corresponding to $\boldsymbol{\phi}_k$. More formally, introducing the notion of sequence length into an autoencoder reshapes the original input matrix $\boldsymbol{\phi} \in \mathbb{R}^{N\times 1}$ to $\tilde{\boldsymbol{\phi}} \in \mathbb{R}^{(N-s+1)\times s}$, where each row now consists of s entries: the entry to be reconstructed and the s-1 preceding entries. This is shown for an input of size 10 and sequence length of 3 in Figure 18.

In the AEKF the reshaping of the input to facilitate a sequence length occurs before $\boldsymbol{\phi}_k$ is passed to the first encoder layer. As the Kalman Filter in these experiments has measurements $\mathbf{z} \in \mathbb{R}^1$, the encoder maps $\tilde{\boldsymbol{\phi}} \in \mathbb{R}^{(N-s+1)\times s}$ to $\mathbb{R}^{(N-s+1)\times 1}$. From this point, the AEKF behaves the same as if there was no sequence length, passing the Kalman Filter output through the decoder and comparing $\hat{\boldsymbol{\phi}}$ with $\boldsymbol{\phi}_{true}$ as in (74).

Long Short-Term Memory Recurrent Network The LSTM used for testing consisted of an input layer, two hidden layers, and an output affine layer mapping back to the input dimensions. As with the AEKF, each epoch consisted of passing a single simulated sensor response to the LSTM, with weights updated each epoch.

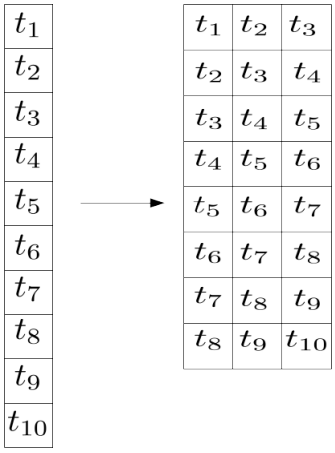


Figure 18: Example of an autoencoder sequence length transformation for a sequence length of 3. Here the preceding two time series samples are included in the row representing the element ϕ_k of the input vector. This ostensibly allows for more information about past history to be utilized when 'reconstructing' the input ϕ_k.

Furthermore, the training curve and noise sampling were randomized in the same manner as during AEKF training. Corresponding to the notation above, the input to the LSTM is a 3-dimensional tensor of the form $\mathbb{R}^{N\times s\times 1}$, where s corresponds to the LSTM sequence length and is analogous to the AEKF sequence length. That is, when using an LSTM for time series prediction, the LSTM sequence length is the number of samples prior to time t taken as input in the prediction of the sample at time t. The reason sequence length is a parameter is that the LSTM performance was very poor with a sequence length of 1. In order to be more principled in our comparison we tested the LSTM with sequence lengths greater than 1, which improved LSTM performance significantly.

4.2.4 Training and Testing Protocols

In our evaluation five models were compared: AEKF NCA, AEKF NCV, Kalman Filter NCA, Kalman Filter NCV, and LSTM. The NCV and NCA following the

model names indicate the dynamical model used in the corresponding AEKF or Kalman Filter. We describe the training protocol for the AEKF and LSTM models and the testing protocol for all five models below.

In the case of the Kalman Filter there is no training. However, for the AEKF and LSTM models a training protocol was required. It is important to emphasize that all AEKF and LSTM models were trained with domain randomization. As mentioned above, the randomization occurred over the parameter α in (87). Taking training with Gaussian noise as an example, each epoch an α was selected uniformly such that the ground truth curve was within $[0, 100]$ and consisted of 100 points. Next, 100 points were sampled from a Gaussian distribution and added to the curve. This noisy curve was then passed to the AEKF or LSTM. Another important point is that because we are using domain randomization, the *ground truth is known*. Thus, it can be utilized in the AEKF and LSTM objective function. When training with domain randomization, the AEKF's and LSTM's learned parameters are updated by comparing the model's prediction, $\hat{\boldsymbol{\phi}}$, against the *ground truth* of the original input, $\boldsymbol{\phi}_{true}$, and not the noisy input, $\boldsymbol{\phi}$, itself as shown in (74).

As mentioned in Section 3.2, since training with domain randomization involves the use of simulated curves with added noise, the ground truth curve is known during training. The idea here is to train the AEKF and LSTM to fit the ground truth on simulated data, with the view that this learning will generalize to fitting the ground truth on curves in a testing set. *However, the crucial point is that while the ground truth is used to train the AEKF, it does not affect model predictions on the testing set during evaluation.*

Domain randomization avoids overfitting specifically because it randomizes the training over the entire problem domain. Since it is extremely unlikely that the

same curve with the exact same noise was seen twice during training, there is no risk of overfitting with domain randomization. In terms of data snooping, domain randomization is similar to giving students several practice tests and then showing them the solutions so they can make adjustments and learn. Yet for the final exam, the students are given a completely different set of test questions which they have never seen before, and are asked to provide the "truth." As with overfitting, since the curves in the test set were never seen during training, data snooping is avoided. The process described above was repeated for the AEKF NCV, AEKF NCA, and LSTM models, on Gaussian, bimodal, and real sensor noise, for sequence lengths 1-10.

From one point of view, domain randomization obscures the distinction between training and testing sets. However, in order to compare all five models on the same data, twenty fixed curves were generated via domain randomization with the same parameter ranges used during training. These twenty curves were then used to test the trained AEKF and LSTM models and the Kalman Filter (discussed below). Taking the AEKF NCV model trained with Gaussian noise and a sequence length of 5 as an example, this model was given each of the twenty test curves and, for each test curve, the filtered output was compared with the known ground truth and test MSE computed. The average test MSE for the twenty curves was then used to assign a single numerical value to this model.

In the case of both Kalman Filter models, the values of $\mathbf{Q}_k$ and $\mathbf{R}_k$ (both diagonal matrices with constant values) were the parameters to be "optimized." This was done using cross validation, by selecting the best MSE from 1,000 (q, r) pairs on each of the twenty test set curves, where q and r are the diagonal values of $\mathbf{Q}_k$ and $\mathbf{R}_k$. It should be noted the methods used to select these values were chosen

to allow the Kalman Filter to perform as best as possible. The selection of q and r were based upon the lowest MSE when comparing the Kalman Filter's estimate to the ground truth for each of the test curves. However, in practice, the ground truth would not be known as this is the reason for implementing the Kalman Filter. This selection process was done to find the best performing Kalman Filter, however unprincipled in practice, and to demonstrate the AEKF outperformed the Kalman Filter even when the Kalman Filter "cheated."

One crucial aspect of training with domain randomization is to ensure the ground truth training curves and noise samples are truly random. To ensure this, both the Numpy [60] and Tensorflow [2] random seeds were left unset.

4.2.5 Numerical Results

In this section, test MSE vs. sequence length plots, along with the corresponding data in table form, are shown for all five models, three noise types, and ten sequence lengths. Figures 19 through 21 present the results for Gaussian, bimodal and sensor noise respectively. Tables 3 through 5 contain the corresponding test MSE results with standard deviation. In most cases, the AEKF models outperformed the two Kalman Filter models and LSTM. What is of interest is to note that with Gaussian noise, the standard Kalman Filter models outperformed the AEKF models for low sequence lengths, but with increasing sequence length the AEKF performed better. The fact the Kalman Filter performed better for low sequence lengths is not surprising given the Kalman Filter optimality conditions mentioned previously. However, with bimodal noise, the LSTM surpasses the Kalman Filter models at larger sequence lengths and the AEKF models have lower MSE for all sequence lengths. This result is indicative of the superior performance of the

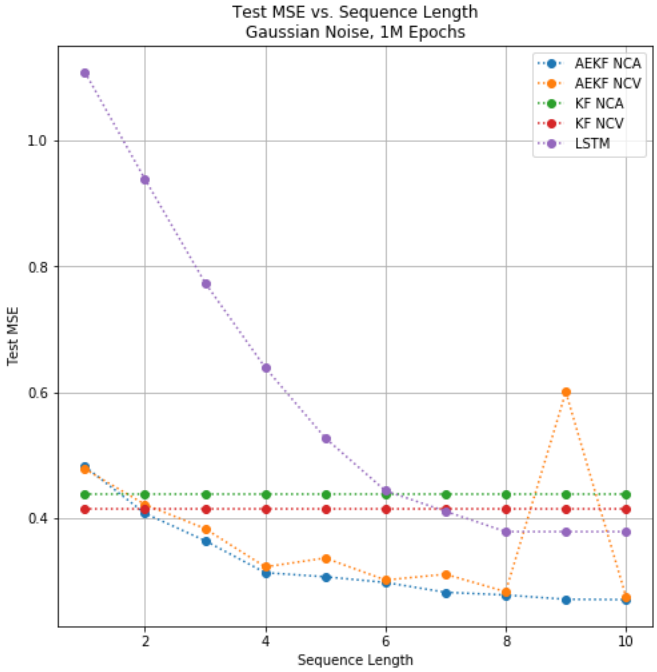


Figure 19: Test MSE vs. Sequence Length for Gaussian Noise. Here we see both AEKF models and the LSTM show a trend of decreasing MSE with sequence length, with the AEKF models outperforming the LSTM. Also, while initially the Kalman Filter outperforms the AEKF models, from a sequence length of three onwards the AEKF models achieve lower test set MSE than the Kalman Filter models. The "blip" for the AEKF NCV at sequence length 9 presents a problem we never completely resolved. However, once we were able to learn a positive definite measurement noise covariance matrix $\mathbf{R}_k$ for arbitrary dimensions we followed this track. Note, the same Kalman Filter results are shown for each sequence length for consistency as the Kalman Filter does not have a sequence length.

AEKF over the Kalman Filter and LSTM. However, looking at Figures 19 and 21 we see several "blips." This, we believe, is related to the fact the AEKF has a larger standard deviation than the LSTM. The "blips" are even more pronounced with the real sensor results, where for the AEKF NCV model it is not just a single spike. Despite these results, we are confident the issue is a technical matter and not a fundamental flaw of the AEKF. This is supported by the successful bimodal results. However, we never successfully resolved this issue as we discuss below although

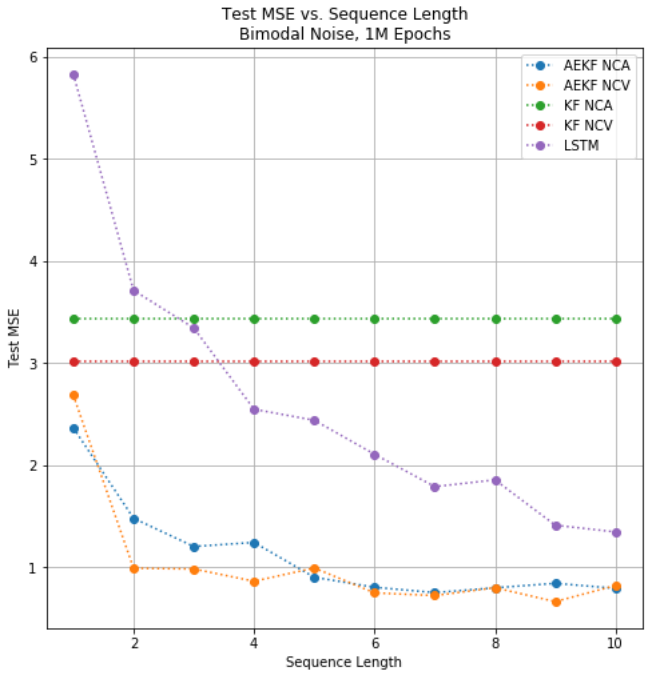


Figure 20: Test MSE vs. Sequence Length for Bimodal Noise. Here we see a similar trend as in Figure 19 except the AEKF outperforms the Kalman Filter models even with a sequence length of one. This is not surprising as the Kalman Filter is known to be suboptimal in the case of non-Gaussian noise processes. We also notice with bimodal noise there were no "blips." However, this could not consistently be reproduced.

given the results in this section it may be a direction of future research.

4.2.6 Summary

In this section we demonstrated the AEKF, with both NCV and NCA models, achieves better state estimation in most cases, in terms of MSE, compared to the

Table 3: Test MSE vs. Sequence Length for Gaussian Noise Results

Sequence Length	AEKF NCA	AEKF NCV	KF NCA	KF NCA	LSTM
1	0.48 ± 0.14	0.48 ± 0.18	0.44 ± 0.12	$\mathbf{0.42 \pm 0.11}$	1.11 ± 0.19
2	$\mathbf{0.41 \pm 0.11}$	0.42 ± 0.11	0.44 ± 0.12	0.42 ± 0.11	0.94 ± 0.16
3	$\mathbf{0.36 \pm 0.11}$	0.38 ± 0.11	0.44 ± 0.12	0.42 ± 0.11	0.77 ± 0.17
4	$\mathbf{0.31 \pm 0.09}$	0.32 ± 0.11	0.44 ± 0.12	0.42 ± 0.11	0.64 ± 0.17
5	$\mathbf{0.31 \pm 0.09}$	0.34 ± 0.12	0.44 ± 0.12	0.42 ± 0.11	0.53 ± 0.13
6	0.3 ± 0.1	$\mathbf{0.3 \pm 0.09}$	0.44 ± 0.12	0.42 ± 0.11	0.44 ± 0.11
7	$\mathbf{0.28 \pm 0.08}$	0.31 ± 0.1	0.44 ± 0.12	0.42 ± 0.11	0.41 ± 0.11
8	0.28 ± 0.09	$\mathbf{0.28 \pm 0.08}$	0.44 ± 0.12	0.42 ± 0.11	0.38 ± 0.1
9	$\mathbf{0.27 \pm 0.08}$	0.6 ± 0.27	0.44 ± 0.12	0.42 ± 0.11	0.38 ± 0.1
10	$\mathbf{0.27 \pm 0.1}$	0.28 ± 0.08	0.44 ± 0.12	0.42 ± 0.11	0.38 ± 0.1

Table 4: Test MSE vs. Sequence Length for Bimodal Noise Results

Sequence Length	AEKF NCA	AEKF NCV	KF NCA	KF NCA	LSTM
1	$\mathbf{2.37 \pm 1.87}$	2.69 ± 1.25	3.43 ± 1.14	3.02 ± 1.01	5.83 ± 1.93
2	1.48 ± 1.13	$\mathbf{0.99 \pm 0.97}$	3.43 ± 1.14	3.02 ± 1.01	3.71 ± 1.87
3	1.2 ± 1.57	$\mathbf{0.98 \pm 1.22}$	3.43 ± 1.14	3.02 ± 1.01	3.34 ± 1.35
4	1.24 ± 1.15	$\mathbf{0.86 \pm 1.11}$	3.43 ± 1.14	3.02 ± 1.01	2.55 ± 1.77
5	$\mathbf{0.9 \pm 1.15}$	0.99 ± 1.68	3.43 ± 1.14	3.02 ± 1.01	2.44 ± 1.83
6	0.8 ± 1.02	$\mathbf{0.75 \pm 0.82}$	3.43 ± 1.14	3.02 ± 1.01	2.11 ± 1.4
7	0.75 ± 0.76	$\mathbf{0.72 \pm 0.61}$	3.43 ± 1.14	3.02 ± 1.01	1.79 ± 1.64
8	0.8 ± 0.84	$\mathbf{0.8 \pm 0.66}$	3.43 ± 1.14	3.02 ± 1.01	1.86 ± 0.95
9	0.84 ± 0.79	$\mathbf{0.66 \pm 0.58}$	3.43 ± 1.14	3.02 ± 1.01	1.41 ± 0.96
10	$\mathbf{0.79 \pm 0.67}$	0.82 ± 0.83	3.43 ± 1.14	3.02 ± 1.01	1.35 ± 1.04

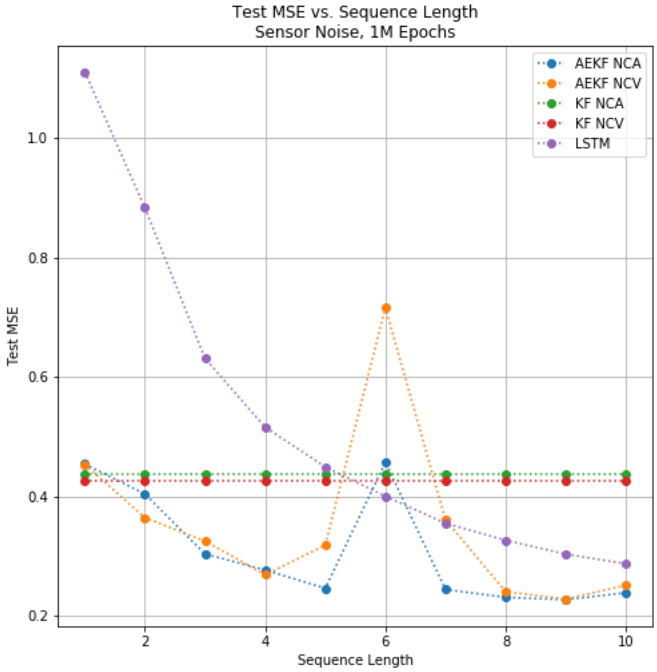


Figure 21: Test MSE vs. Sequence Length for Sensor Noise. The trends here are very similar to Figure 19, although the issue with the "blips" is more pronounced here. The fact the Kalman Filter initially outperforms the AEKF models may indicate the sensor noise was close to Gaussian.

Kalman Filter with NCV and NCA models and the LSTM, with simulated Gaussian, bimodal and, actual sensor noise. This result addresses the two common difficulties of the Kalman Filter: (a) tuning, or estimation, of the measurement noise covariance $\mathbf{R}_k$ and (b) improving the Kalman Filter even when optimality conditions are not met. Although the issue with the "blips" was never fully resolved, a new direction of research greatly improved AEKF performance. Once we were able to learn a sequence of non-singular measurement noise covariance matrices, $\{\mathbf{R}_k\}_{k=1}^{N}$, this no longer limited the dimensions of the AEKF's Kalman Filter space to $\mathbb{R}^1$ and greatly improved the AEKF state estimation capabilities. The consequences of this are explored in the next section.

Table 5: Test MSE vs. Sequence Length for Sensor Noise Results

Sequence Length	AEKF NCA	AEKF NCV	KF NCA	KF NCA	LSTM
1	0.45 ± 0.19	0.45 ± 0.17	0.44 ± 0.15	$\mathbf{0.42 \pm 0.15}$	1.11 ± 0.23
2	0.4 ± 0.15	$\mathbf{0.36 \pm 0.13}$	0.44 ± 0.15	0.42 ± 0.15	0.88 ± 0.27
3	$\mathbf{0.3 \pm 0.12}$	0.32 ± 0.12	0.44 ± 0.15	0.42 ± 0.15	0.63 ± 0.22
4	0.28 ± 0.1	$\mathbf{0.27 \pm 0.11}$	0.44 ± 0.15	0.42 ± 0.15	0.52 ± 0.18
5	$\mathbf{0.25 \pm 0.09}$	0.32 ± 0.14	0.44 ± 0.15	0.42 ± 0.15	0.45 ± 0.16
6	0.46 ± 0.36	0.72 ± 0.63	0.44 ± 0.15	0.42 ± 0.15	$\mathbf{0.4 \pm 0.13}$
7	$\mathbf{0.24 \pm 0.09}$	0.36 ± 0.22	0.44 ± 0.15	0.42 ± 0.15	0.35 ± 0.12
8	$\mathbf{0.23 \pm 0.08}$	0.24 ± 0.1	0.44 ± 0.15	0.42 ± 0.15	0.33 ± 0.11
9	$\mathbf{0.23 \pm 0.09}$	0.23 ± 0.09	0.44 ± 0.15	0.42 ± 0.15	0.3 ± 0.1
10	$\mathbf{0.24 \pm 0.09}$	0.25 ± 0.1	0.44 ± 0.15	0.42 ± 0.15	0.29 ± 0.12

4.3 Deep Learning with Domain Randomization for Robust State Estimation

Building on the results in Section 4.2, in this section we no longer restrict the functions the AEKF is trained on to sigmoidal functions, but include exponential and sinusoidal functions as well. In this context we train the AEKF in two different ways. First, similar to Section 4.2 but expanding the function families, we train three separate AEKF models on exponential, sigmoidal, and sinusoidal curve families respectively, which we refer to as *single family models*. Secondly, we also train a single AEKF model on all three of these curve families, which we refer to as *multiple family models*. That is, during training this AEKF model saw training samples from all three curve families. The results from both these tests prove to be a significant step towards a generalized state estimation system. In this section, we also

introduce AEKF models trained on Cauchy noise as this tests the AEKF's ability to perform state estimation in the presence of significant outliers. Perhaps most importantly though, we no longer limit the AEKF's Kalman Filter measurement and measurement noise covariance dimensions to $\mathbb{R}^1$. As mentioned in Section 4.2, and indicated by (8), the AEKF must learn a positive definite measurement noise covariance matrix $\mathbf{R}_k$ for each element of $\{\mathbf{R}_k\}_{k=1}^N$. However, when the research in Section 4.2 was conducted, we had not solved the technical issue of learning a positive definite matrix. While the addition of a sequence length to the AEKF did improve AEKF state estimation, it came at the cost of the "blips" discussed in Section 4.2. As mentioned above, we were never able to fully solve this problem but once we were able to learn positive definite measurement noise covariance matrices, we put the sequence length idea aside. As will be seen in this section, even without the use of sequence length, when the AEKF's Kalman Filter's measurement space is extended to $\mathbb{R}^N$ the AEKF still outperforms the standard Kalman Filter and LSTM.

4.3.1 Simulation Environment

All models were trained with domain randomization in the same manner as in Section 4.2. In addition to the sigmoidal curves in (87), the exponential and sinusoidal curve families were generated, respectively, by

$$\phi_g(x;\alpha) = e^{\alpha x} - 1 \tag{90}$$

$$\phi_g(x;\alpha,\beta) = \alpha \sin \beta x \tag{91}$$

where $x \in [0, 100]$. For (90) and (91), the range of α was such that $\phi_g \in [0, 100]$. In the case of (91), an additional parameter (also sampled uniformly), β, was added

to allow variation in the frequency. The sigmoidal function parameters were the same as those in Section 4.2.

4.3.2 Noise

The simulated noise added to the training curves was the same as in Section 4.2, except the sensor noise was replaced by Cauchy noise. However, the parameters of the Gaussian and bimodal noises were changed. This was done to test the AEKF state estimation capabilities in more challenging noise scenarios. The Gaussian noise was sampled from a Gaussian distribution $\mathcal{N}(0, 5)$ and two Gaussian distributions used to generate the bimodal noise (89) had means -15.0 and 15.0 and a common covariance of 5.0.

Cauchy Noise Cauchy noise was drawn from a standard Cauchy distribution with PDF

$$f(x) = \frac{1}{\pi(1 + x^2)} \tag{92}$$

4.3.3 Model Parameters

Kalman Filter The Kalman Filter parameters were the same as in Section 4.2.

Autoencoder-Kalman Filter Apart from the dimensions of the AEKF's Kalman Filter, the network architecture here is the same as in Section 4.2. In all experiments $\mathbf{z}_k \in \mathbb{R}^{16}$ and $\mathbf{R}_k \in \mathbb{R}^{16\times 16}$ for all $k = \{1, 2, \ldots, N\}$. However, each $\mathbf{R}_k$ was not learned directly as alluded to in Section 2.5. In order to construct $\{\mathbf{R}_k\}_{k=1}^N$ from $\{\mathbf{L}_k\}_{k=1}^N$, each $\mathbf{L}_k \in \mathbb{R}^{16\times 16}$ was an upper triangular matrix with positive diagonal entries. However, the input to the function $\mathcal{T}$ in (63) was a vector in $\mathbb{R}^{136}$, which is the minimum number of parameters necessary to learn an upper triangular matrix

in $\mathbb{R}^{16\times16}$. Learning $\mathbf{L}_k$ in this way guarantees $\mathbf{R}_k = \mathbf{L}_k^\intercal\mathbf{L}_k$ is symmetric positive definite, which, in turn, guarantees (7) is nonsingular [3]. This is proven in Section 4.5.1.

Long Short-Term Memory Recurrent Network The dimensions, parameters, and testing procedure for the LSTM is the same as in Section 4.2, with the exception we now consider five sequence lengths: 1, 10, 15, 20, 25.

4.3.4 Test Protocol

The training and hyperparameter selection protocols here are the same as in Section 4.2. To evaluate how the optimal AEKF, LSTM, and KF models compared with one another, each of the five optimal models were tested on the same 100 test curves. These test curves were generated in the same manner as in Section 4.2.

4.3.5 Numerical Results

Single Family Model Results Tables 6, 7, and 8 show test set MSE and associated ratio for each model and all three noise types. The values in the ratio column are calculated by dividing all elements of each column by the AEKF NCV MSE value in that column. This provides a standard basis of comparison. For a given row, each MSE/ratio pair corresponds to the test set evaluation for models trained on the corresponding curves. For example, the cell corresponding to row AEKF NCV and column Exp. MSE is the test set MSE for the AEKF NCV model trained on exponential curves. For the LSTM models the number in the model column corresponds to the sequence length. For each column, the smallest MSE value is in bold.

Apart from one sinusoidal test in Table 7, either the AEKF NCV or AEKF NCA model achieves the lowest MSE in all tests. While in all cases either the AEKF and LSTM outperform the KF, the difference between the Kalman Filter MSE values for Gaussian and bimodal noise is significant, while this difference for the AEKF is not as significant. The increased MSE for bimodal noise is consistent with the Kalman Filter optimality conditions discussed in Section 2.1. Recall, one of the primary motivations for developing the AEKF was to apply the Kalman Filter to contexts involving non-Gaussian noise. Furthermore, while in general the LSTM MSE decreases with sequence length, the AEKF was able to perform well without the notion of sequence length in Section 4.2. Noting the LSTM MSE decreases with sequence length, the reader may wonder why we stopped at a sequence length of 25. First, as our curves only had 110 samples, going beyond a sequence length of 25 would place strict limits on the number of points being estimated. Secondly, although not studied formally, the training time for the LSTM grew with sequence length and the higher performing models took longer to run than the AEKF. One future direction of this work is to study the time complexity of the AEKF and LSTM models.

The most interesting result is related to the tests with Cauchy noise, where the AEKF clearly outperforms the other models. Unsurprisingly, the Kalman Filter does poorly since Cauchy noise is likely to have large deviations from the mean. These deviations, or "spikes", in the measurements draw the Kalman Filter's state estimate far from the mean and result in a large MSE. Interestingly, with Cauchy noise, the LSTM MSE increases with sequence length after a sequence length of 10. This may indicate the LSTM is taking an average of the N preceding sequence length points (along with correcting for the curve) in its state estimation. The rea-

son for this is the "spikes" are sparsely spaced and only a large sequence length is likely to encounter multiple spikes, significantly increasing the MSE. This is a possible explanation for why the LSTM MSE increases with sequence length with Cauchy noise, whereas it decreased with Gaussian and bimodal noise. Here we wish to make explicit that we observed the LSTM MSE on Cauchy data varies greatly over different test sets. Recall that the MSE values in Tables 6, 7, and 8 are the average MSE values of each model evaluated on the 100 curves in the testing datasets. As a result, given a different testing dataset, the LSTM will likely achieve a different MSE than the results in Table 8. This high sensitivity to different testing datasets is most likely due to the fact a small fraction of Cauchy data testing datasets will include very large outliers, which the Cauchy distribution is prone to have. However, it should also be noted all models were evaluated on the same testing datasets and the AEKF did not show the same sensitivity to outliers as the LSTM. Additionally, this result demonstrates the LSTM does not generalize well on data with large outliers. In summary, the tests with Cauchy noise (a) demonstrate the AEKF's ability to handle non-Gaussian noise and (b) significantly outperform the other methods considered here. Fig. 22 shows an example of AEKF performance on a test set sine curve with Cauchy noise. This ability of the AEKF to mitigate outliers is one of its most important features and is given mathematical justification in Section 4.5.

Multiple Family Model Results Table 9 shows results for the models trained on all three curve families and added Gaussian noise with a slightly different MSE/ratio format. Here, the ratio column is the ratio between the multiple family model MSE and the single family model MSE. Table 10 presents the same results for the case of bimodal noise.

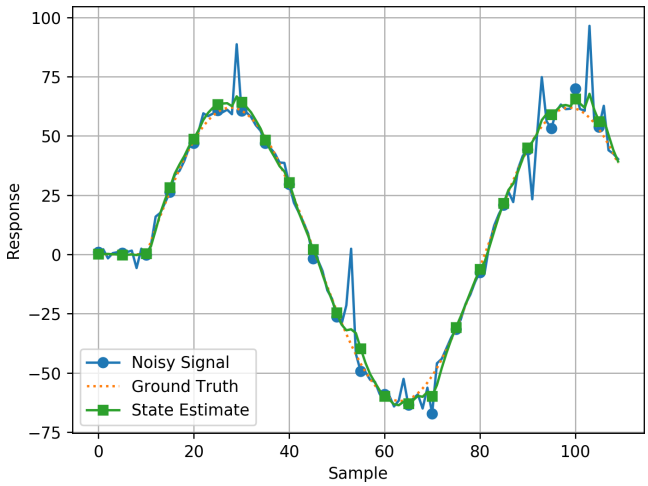


Figure 22: AEKF filtering of a test set sine function with Cauchy noise. Note the state estimate, represented by the solid green line, stays close to the ground truth even in the presence of outliers. This behavior is difficult to achieve with a standard Kalman Filter, which is more easily misled by outlier data.

The MSE columns in Tables 9 and 10 are the test set MSE for each multiple family model evaluated on a separate test set for exponential, sigmoidal, and sinusoidal curves. In both tables the best performing models are similar to those in the single family tests. However, the overall MSE is larger in the multiple family model case. This is not surprising though, as both the single family model and multiple family model AEKFs were trained for 100,000 epochs each. Since the multiple family model AEKFs are learning to filter three families of functions, it is possible more training epochs would improve these results. Furthermore, all ratio values in Tables 9 and 10 never exceed three, the number of curve families being learned, further indicating increasing the number of training epochs may improve performance.

Table 6: MSE and ratio results for single curve family models with Gaussian noise. Here the AEKF consistently achieves the lowest test set MSE. The values in the ratio column are calculated by dividing all elements of each column by the AEKF NCV MSE value in that column. Bold values indicate the smallest MSE in each column. Note, NCV and NCA refer to specific Kalman Filter models in the AEKF and standard Kalman Filter and the number following each LSTM model is the sequence length used for that model.

Model	Exp. MSE	Exp. Ratio	Sig. MSE	Sig. Ratio	Sine MSE	Sine Ratio
AEKF NCV	**1.52**	**1.00**	1.90	1.00	5.70	1.00
AEKF NCA	1.57	1.04	**1.85**	**0.98**	**5.17**	**0.91**
LSTM 1	16.03	10.57	16.06	8.45	32.03	5.62
LSTM 10	3.05	2.01	5.12	2.70	13.05	2.29
LSTM 15	2.67	1.76	4.48	2.36	9.73	1.71
LSTM 20	2.52	1.66	4.29	2.26	7.89	1.38
LSTM 25	2.54	1.67	4.27	2.25	7.66	1.34
KF NCV	4.04	2.67	4.96	2.61	8.94	1.57
KF NCA	4.51	2.97	5.83	3.07	9.01	1.58

Table 7: MSE and ratio results for single curve family models with bimodal noise. This was the first test which evaluated the AEKF, LSTM, and Kalman Filter on non-Gaussian noise. Apart from models trained and tested on sinusoidal curves, the AEKF outperforms both the LSTM and Kalman Filter. Even in the sinusoidal case, the lower MSE achieved by the LSTM is a small improvement compared to the AEKF's lower MSE in the exponential and sigmoidal cases. Unlike the AEKF and LSTM, the Kalman Filter's bimodal MSE increases by a much larger factor over its Gaussian MSE. Bold values indicate the smallest MSE in each column. Note, NCV and NCA refer to specific Kalman Filter models in the AEKF and standard Kalman Filter and the number following each LSTM model is the sequence length used for that model.

Model	Exp. MSE	Exp. Ratio	Sig. MSE	Sig. Ratio	Sine MSE	Sine Ratio
AEKF NCV	2.37	1.00	**2.96**	**1.00**	13.07	1.00
AEKF NCA	**2.35**	**0.99**	3.14	1.06	12.59	0.96
LSTM 1	79.36	33.52	107.45	36.32	199.00	15.22
LSTM 10	5.10	2.15	7.94	2.68	21.90	1.67
LSTM 15	4.77	2.02	6.96	2.35	15.11	1.16
LSTM 20	4.71	1.99	6.64	2.25	11.86	0.91
LSTM 25	4.66	1.97	6.24	2.11	**11.50**	**0.88**
KF NCV	23.77	10.04	30.15	10.19	57.15	4.37
KF NCA	36.86	15.57	45.02	15.22	65.14	4.98

Table 8: MSE and ratio results for single curve family models with Cauchy noise. This is the most significant result of this section. It demonstrates the AEKF achieves a significantly lower MSE for all three curve families on a noise distribution that is very challenging for the traditional Kalman Filter. A possible explanation for the increasing LSTM MSE with sequence length is that the LSTM may be utilizing the average value of the samples in its sequence length. As the Cauchy distribution is subject to "spikes" these would skew the average, especially for long sequence lengths. In contrast, the AEKF is not averaging preceding samples, but is learning which samples to ignore and which to keep, regardless of their deviation from the ground truth. Bold values indicate the smallest MSE in each column. Note, NCV and NCA refer to specific Kalman Filter models in the AEKF and standard Kalman Filter and the number following each LSTM model is the sequence length used for that model.

Model	Exp. MSE	Exp. Ratio	Sig. MSE	Sig. Ratio	Sine MSE	Sine Ratio
AEKF NCV	**0.39**	**1.00**	1.16	1.00	3.56	1.00
AEKF NCA	0.43	1.08	**0.89**	**0.77**	**2.96**	**0.83**
LSTM 1	35.90	90.99	45.45	39.16	82.12	23.06
LSTM 10	1.35	3.42	5.72	4.93	21.18	5.95
LSTM 15	5.15	13.04	45.43	39.14	109.56	30.77
LSTM 20	10.66	27.01	183.51	158.11	644.53	180.99
LSTM 25	63.08	159.89	509.68	439.12	835.96	234.74
KF NCV	99.26	251.58	953.21	821.25	2979.70	836.72
KF NCA	212.33	538.18	3541.22	3051.00	13427.75	3770.61

Table 9: MSE and ratio results for multiple curve family models with Gaussian noise. Here, the ratio column is the ratio of the multiple family model MSE and the corresponding single family model MSE. These results are consistent with those in Table 6 in that the models with the lowest MSE are either the AEKF NCV or AEKF NCA. While the absolute MSE is higher for each model compared with Table 6, this is not surprising as the models presented here were trained for 100,000 epochs as well, yet had to learn three different curve families instead of a single family. Furthermore, all ratio values never exceed three, the number of curve families being learned, indicating increasing the number of training epochs may improve performance. The number after the LSTM models indicates the sequence length and the Kalman Filter results from Table 6 are included for comparison.

Model	Exp. MSE	Exp. Ratio	Sig. MSE	Sig. Ratio	Sine MSE	Sine Ratio
AEKF NCV	**2.71**	**1.79**	3.85	2.02	8.10	1.42
AEKF NCA	2.82	1.79	**3.44**	**1.85**	**7.47**	**1.44**
LSTM 1	16.93	1.06	16.78	1.04	33.23	1.04
LSTM 10	5.00	1.64	6.28	1.23	14.91	1.14
LSTM 15	3.92	1.47	5.55	1.24	11.68	1.20
LSTM 20	3.65	1.45	5.47	1.27	9.25	1.17
LSTM 25	3.17	1.25	5.74	1.34	8.71	1.14
KF NCV	4.04	1.00	4.96	1.00	8.94	1.00
KF NCA	4.51	1.00	5.83	1.00	9.01	1.00

4.3.6 Summary

The results in this section build upon the success of Section 4.2. By using more challenging noise distribution parameters for Gaussian and bimodal noises, the addition of Cauchy noise, and training a single AEKF on multiple curve families, we moved one step closer to generalizing the state estimation capabilities of the AEKF.

In this section both single and multiple family models were limited to a finite number of well-known function families. Building on the results here, in the next section we expand the class of functions more generally by exploring training and testing on random truncated Taylor polynomials, up to some fixed degree p, with

Table 10: MSE and ratio results for multiple curve family models with bimodal noise. Here, the ratio column is the ratio of the multiple family model MSE and the corresponding single family model MSE. These results are consistent with those in Table 7 in that the models with the lowest MSE are either the AEKF NCV, AEKF NCA, and LSTM with large sequence length. While the absolute MSE is higher for each model compared with Table 7, this is not surprising as the models presented here were trained for 100,000 epochs as well, yet had to learn three different curve families instead of a single family. Furthermore, all ratio values never exceed three, the number of curve families being learned, indicating increasing the number of training epochs may improve performance. The number after the LSTM models indicates the sequence length and the Kalman Filter results from Table 7 are included for comparison.

Model	Exp. MSE	Exp. Ratio	Sig. MSE	Sig. Ratio	Sine MSE	Sine Ratio
AEKF NCV	**5.36**	**2.26**	8.15	2.76	17.74	1.36
AEKF NCA	5.38	2.29	**7.04**	**2.24**	17.08	1.36
LSTM 1	84.00	1.06	111.18	1.03	209.60	1.05
LSTM 10	8.90	1.75	11.45	1.44	25.21	1.15
LSTM 15	6.98	1.46	8.82	1.27	17.98	1.19
LSTM 20	6.56	1.39	9.43	1.42	**13.69**	**1.15**
LSTM 25	6.63	1.42	9.19	1.47	14.55	1.27
KF NCV	23.77	1.00	30.15	1.00	57.15	1.00
KF NCA	36.86	1.00	45.02	1.00	65.14	1.00

arbitrary coefficients. That is, for

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_p x^p \tag{93}$$

we randomly select a polynomial degree of $p \leq P$ with random coefficients $\{a_0, a_1, \cdots, a_p\}$ (within a predetermined range), with noise drawn from a variety of distributions, each training epoch.

4.4 Generalizing Robust State Estimation with Domain Randomization

In Section 4.3 it was shown that an AEKF with NCV and NCA models outperforms both a standard Kalman Filter with NCV and NVA models and an LSTM recurrent neural network on simulated exponential, sigmoidal, and sinusoidal curves with Gaussian, bimodal, and Cauchy noise.

In this section we build on the results in Section 4.3 as a next step in the direction of developing a general time series state estimation system where we apply domain randomization to a more general family of functions. Instead of limiting the family of functions to three specific families, we propose to train an AEKF on the family of functions defined by a p^{th} degree truncated Taylor Polynomial with random coefficients

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_p x^p \tag{94}$$

That is, training will consist of generating training curves with $p + 1$ random coefficients within a predetermined range. While it is unlikely the exact p^{th} order truncated Taylor Polynomial for well-known functions will be randomly selected during training (e.g. exponential and sinusoidal curves), the motivation is that by covering the function space with a large variety of random truncated Taylor Polynomials, those functions learned by the AEKF will provide a covering of the parameter space. As a result, well known functions will be well approximated.

Generating the simulated data begins by first generating a ground truth curve

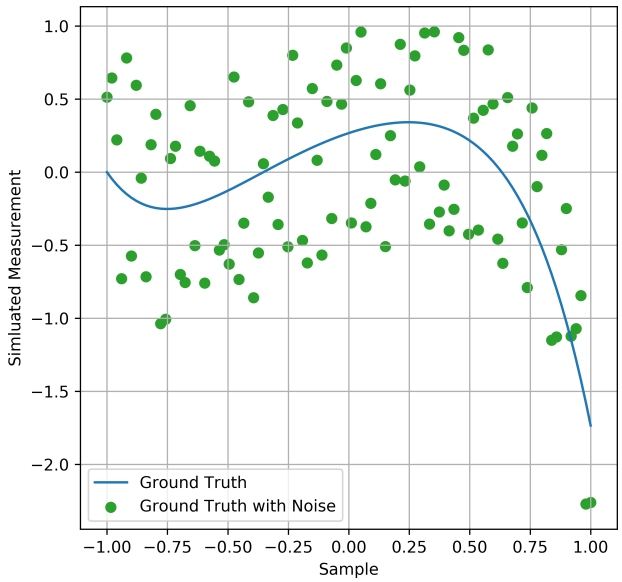


Figure 23: Example of a simulated training curve with bimodal noise. The smooth line is the ground truth truncated Taylor Polynomial ϕ_{true} with the points representing the ground truth with added noise ϕ_{noise}. At each training epoch a new simulated curve was generated and passed to the AEKF. As a result, the AEKF effectively never saw the same curve twice during training.

according to the following

$$\phi_{true} = \sum_{l=0}^{p} a_l x^l \tag{95}$$

$$p \sim \mathcal{U}\{3, 4, 5, 6, 7\} \tag{96}$$

$$a_l \sim \mathcal{U}[-1, 1) \tag{97}$$

where (95) is a random truncated Taylor Polynomial of order p, and p and a_l are sampled uniformly according to (96) and (97) respectively. Next, synthetic noise, ϕ_{noise}, is added to ϕ_{true} by sampling from a given probability distribution, resulting in the simulated training data $\phi = \phi_{true} + \phi_{noise}$. In our experiments we considered Gaussian, bimodal, and Cauchy added noise. Figure 23 shows an example of a simulated baseline curve with added bimodal noise.

4.4.1 Experimental Setup

Apart from training on random truncated Taylor Polynomials and the synthetic noise distribution parameters used, the experimental procedures used in this section are identical to Section 4.3 and are briefly summarized here.

In our experiments, an AEKF, LSTM, and standard Kalman Filter, each with three noise types, Gaussian, bimodal, and Cauchy, were compared. In both the AEKF and standard Kalman Filter, an NCV dynamical model was used. The Gaussian noise was drawn from $\mathcal{N}(0, 0.2)$, bimodal noise consisted of two Gaussian distributions $\mathcal{N}(0.5, 0.2)$ and $\mathcal{N}(-0.5, 0.2)$, and the Cauchy noise was drawn from a standard Cauchy distribution.

The AEKF and LSTM were both trained using domain randomization in the same way as in Section 4.2. The test set was randomly generated in the same manner as the training and hyperparameter selection sets and consisted of 1,000 simulated noisy curves. The final MSE for each model was determined by averaging the mean squared error for each of the 1,000 curves in the test set.

As in Section 4.2, the Kalman Filter parameters $\mathbf{Q}_k$ and $\mathbf{R}_k$ were chosen based on optimal performance on the test set.

4.4.2 Experimental Results

For each model, three separate experiments were performed with Gaussian, bimodal, and Cauchy noise. The test set MSE results are presented in Table 11, with the lowest MSE in bold for each noise type. The number in parentheses following the MSE is the MSE ratio between the given model and the AEKF. In the case of Gaussian noise, the Kalman Filter achieved the lowest MSE of the three models. However, the primary motivation for the AEKF is to demonstrate the Kalman

Table 11: Test Set MSE Results for each of the three models and noise types. While the Kalman Filter performs the best with Gaussian noise, for Bimodal noise both the AEKF and LSTM achieve lower test set MSE than the Kalman Filter, with the LSTM performing slightly better. Here the AEKF shows significantly better filtering capabilities is in the presence of Cauchy noise. The fact that Cauchy noise has large outliers explains why the Kalman Filter performs poorly. While both the AEKF and LSTM do significantly better than the Kalman Filter, the mechanism by which this occurs is well understood in the Kalman Filter, while not so for the LSTM. This is due to the fact the Kalman Filter has a well established mathematical foundation, something we leverage in Section 4.5 to prove a theorem regarding the AEKF's ability to mitigate outliers.

	Gaussian	**Bimodal**	**Cauchy**
AEKF	0.011 (1.00)	0.020 (1.00)	**0.174 (1.00)**
LSTM	0.027 (2.46)	**0.019 (0.93)**	0.217 (1.25)
KF	**0.010 (0.88)**	0.042 (2.07)	409.8 (2,361)

Filter can be successfully applied to state estimation in cases where the noise is non-Gaussian. For bimodal noise the AEKF and LSTM are effectively equal in performance while the Kalman Filter's MSE is twice as large. This indicates, even when the Kalman Filter is allowed to data snoop, the AEKF and LSTM models show better performance. Lastly, the main result of this section is the performance on Cauchy noise. Here the AEKF outperforms both the LSTM and the Kalman Filter, with the Kalman Filter's MSE significantly larger. Thus, despite some specific differences, these results are consistent with the general trend in 4.3 and indicate that the AEKF's state estimation capabilities generalize from exponential, sigmoidal, and sinusoidal families of curves to the more general truncated Taylor Polynomials used here. A sample figure showing the AEKF state estimate of a random truncated Taylor Polynomial, along with the ground truth and noise, is shown in Figure 24.

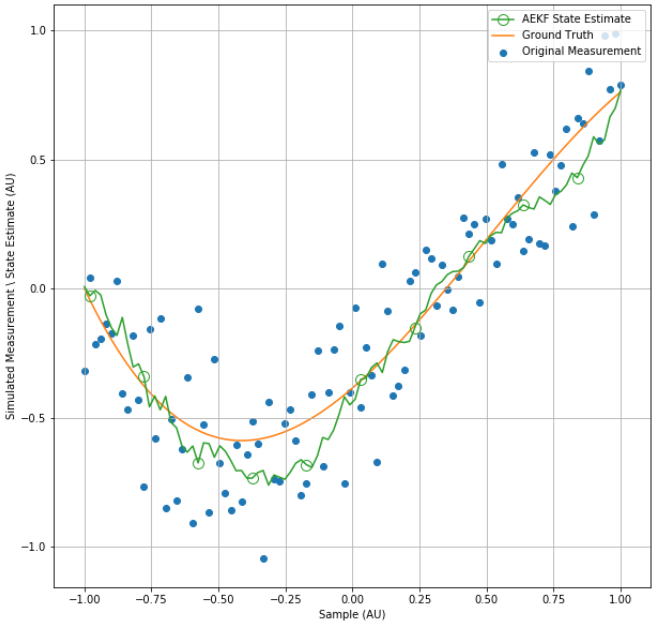


Figure 24: Sample AEKF state estimate of a random truncated Taylor Polynomial ground truth curve with added Gaussian noise.

4.4.3 Summary

In this section we extended the state estimation capabilities of the AEKF, demonstrated in Section 4.3, to a larger class of functions, namely truncated Taylor Polynomials. A next step is to investigate the mathematics behind how the AEKF is able to perform well on Cauchy data, that is data with large outliers, while the standard Kalman Filter, and to a lesser extent the LSTM, are unable to do so. This will be the subject of the next section.

4.5 Measurement Noise Covariance Analysis for the Autoencoder-Kalman Filter

The results in Section 4.4 indicate the LSTM, like the AEKF, is able to effectively mitigate the influence of outliers in Cauchy noise. However the exact mechanism

how this occurs in the LSTM is not as transparent as in the AEKF case. This is due to the fact the AEKF is relying on the well-established theoretical understanding of the Kalman Filter. Specifically, the Kalman Gain $\mathbf{K}_k$ in (9) weights the contribution of the innovation $\tilde{\mathbf{z}}_k$ to the *a posteriori* estimate $\hat{\mathbf{x}}_{k|k}$. Intuitively, if the eigenvalues of $\mathbf{K}_k$ are very small the Kalman Filter will effectively ignore the innovation and vice versa. More precisely, it is the largest eigenvalue of $\mathbf{S}_k^{-1}$ in (8) that provides an upper bound on the norm of the innovation sequence by $\|\tilde{\mathbf{z}}_k\|_2$. Since $\mathbf{S}_k$ depends upon $\mathbf{R}_k$, it is natural to investigate the relation between the outlier mitigation properties of the AEKF and the fact that it is learning $\{\mathbf{R}_k\}_{k=1}^N$.

In order to gain some insight into the outlier mitigation properties of the AEKF, analyzing the Kalman Filter mathematically, in this section we derive a theorem which suggests an empirically testable criteria indicating how the AEKF should behave. If the AEKF does in fact behave as the theorem suggests, this would demonstrate the AEKF behavior can be explained theoretically. We then provide experimental results confirming that the above predicted behavior does in fact occur. Thus, although the question of "how" the AEKF performs its function may still be uncertain, we have moved one step towards understanding by showing "what" it does is consistent with a mathematical analysis of the Kalman Filter.

4.5.1 Mathematical Introduction

Throughout this section we apply standard matrix analysis techniques to the Kalman Filter to aid our understanding of the AEKF. Thus, while the mathematics herein are well-known, their application towards the understanding of deep learning is novel.

In the sections below, $M_{m,n}(\mathbb{R})$ represents the set of all $m \times n$ matrices over the

real numbers and $M_{n,n}(\mathbb{R})$ the set of all $n \times n$ square matrices over the real numbers. While most of the proofs apply to both vectors in $\mathbb{R}^n$ and $\mathbb{C}^n$, we restrict ourselves to vectors and matrices in $\mathbb{R}^n$ and $M_{m,n}(\mathbb{R})$ respectively. That is, the scalar entries in all vectors and matrices are restricted to real numbers. The primary reference used herein is [24].

Before beginning our mathematical analysis we briefly review the motivation for having the AEKF learn $\{\mathbf{R}_k\}_{k=1}^N$. In a standard Kalman Filter, the input is a sequence of noisy measurements $\{\mathbf{z}_k\}_{k=1}^N$, modeled as a sequence of random variables, and a single matrix $\mathbf{R}$, which represents the covariance of $\{\mathbf{z}_k\}_{k=1}^N$. Ideally, $\{\mathbf{z}_k\}_{k=1}^N$ is a zero-mean Gaussian sequence with $\mathbf{R}$ representing the known covariance. In this case, the statistics are complete as the mean and covariance are fully known. However, this is not always the case, as the noisy measurements may not be well-modeled by a Gaussian random variable. Here, estimation techniques are often used to determine the best $\mathbf{R}$ [40, 44, 51]. Even if $\mathbf{R}$ can be well approximated, a single matrix is assumed to capture the covariance for all measurements in $\{\mathbf{z}_k\}_{k=1}^N$, which can be problematic. For example, if the distribution of $\{\mathbf{z}_k\}_{k=1}^N$ is modeled as an α-stable distribution, the covariance is only defined for the Gaussian case, $\alpha = 2$. In the case of a Cauchy distribution, $\alpha = 1$, both the mean and covariance are undefined. However, the sample covariance can always be estimated, even when the population covariance does not exist. But even this approach has problems when dealing with distributions with large tails such as the Cauchy distribution. Thus, the problem of estimating $\mathbf{R}$ for a variety of distributions is non-trivial.

In order to address these issues, the AEKF learns a point-wise sequence of measurement noise covariance matrices, $\{\mathbf{R}_k\}_{k=1}^N$, instead of a single measurement noise covariance ma-

trix. This avoids the problem of both unknown and non-existent covariances, as well as poorly estimated sample covariances, as it learns a (potentially) unique covariance matrix for each measurement, providing the Kalman Filter with more flexibility.

4.5.2 Learning R

In the AEKF, learning $\{\mathbf{R}_k\}_{k=1}^N$ is not simply a matter of learning matrices of the correct dimensions, but it is constrained by properties of the Kalman Filter. In this section we address the requirements for learning $\{\mathbf{R}_k\}_{k=1}^N$ based on these properties.

From (8) we see (7) must be non-singular, which is guaranteed if $\mathbf{S}_k$ is positive definite. In [3, p. 39] it was claimed $\mathbf{S}_k$ is positive definite if $\mathbf{R}_k$ in (7) is positive definite without proof. Here we fill in the details of this claim.

$\mathbf{H}_k\mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^\mathsf{T}$ is Positive Semi-Definite If $\mathbf{P}_{k|k\text{-}1} \in M_{n,n}(\mathbb{R})$ is positive semi-definite, $\mathbf{H}_k\mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^\mathsf{T}$ is positive semi-definite, where $\mathbf{H}_k \in M_{m,n}(\mathbb{R})$ and is non-zero.

Proof. $\mathbf{P}_{k|k\text{-}1}$ is positive semi-definite since it is a covariance matrix. Given a non-zero $\mathbf{v} \in \mathbb{R}^m$, define $\mathbf{u} = \mathbf{H}_k^\mathsf{T}\mathbf{v} \in \mathbb{R}^n$. Since $\mathbf{P}_{k|k\text{-}1}$ is positive semi-definite $\mathbf{u}^\mathsf{T}\mathbf{P}_{k|k\text{-}1}\mathbf{u} \geq 0$ and we write

$$\mathbf{u}^\mathsf{T}\mathbf{P}_{k|k\text{-}1}\mathbf{u} = (\mathbf{v}^\mathsf{T}\mathbf{H}_k)\mathbf{P}_{k|k\text{-}1}(\mathbf{H}_k^\mathsf{T}\mathbf{v}) \tag{98}$$

$$= \mathbf{v}^\mathsf{T}(\mathbf{H}_k\mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^\mathsf{T})\mathbf{v} \geq 0 \tag{99}$$

which shows $\mathbf{H}_k\mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^\mathsf{T}$ is positive semi-definite. □

$\mathbf{S}_k$ is Positive Definite Given positive semi-definite $\mathbf{H}_k\mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^\mathsf{T} \in M_{m,m}(\mathbb{R})$ and positive definite $\mathbf{R}_k \in M_{m,m}(\mathbb{R})$, $\mathbf{S}_k = \mathbf{H}_k\mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^\mathsf{T} + \mathbf{R}_k$ is positive definite.

Proof. Define a non-zero vector $\mathbf{v} \in \mathbb{R}^m$. Writing

$$\mathbf{v}^\mathsf{T}\mathbf{S}_k\mathbf{v} = \mathbf{v}^\mathsf{T}(\mathbf{H}_k\mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^\mathsf{T} + \mathbf{R}_k)\mathbf{v} \tag{100}$$

$$= \mathbf{v}^\mathsf{T}(\mathbf{H}_k\mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^\mathsf{T})\mathbf{v} + \mathbf{v}^\mathsf{T}\mathbf{R}_k\mathbf{v} \tag{101}$$

$$\geq \mathbf{v}^\mathsf{T}\mathbf{R}_k\mathbf{v} > 0 \tag{102}$$

which follows from the fact $\mathbf{R}_k$ is positive definite. □

From this we conclude $\mathbf{S}_k$ is non-singular if $\mathbf{R}_k$ is positive definite.

Learning Positive Definite $\mathbf{R}_k$ The procedure for learning a positive definite $\mathbf{R}_k$ was discussed in Sections 2.5 and 4.3.3 without presenting its mathematical justification. The justification is a statement of Cholesky decomposition [24, p. 441] and included here for completeness: Given that $\mathbf{R}_k \in M_{m,m}(\mathbb{R})$ is symmetric, $\mathbf{R}_k$ is positive definite if and only if there is an upper triangular matrix $\mathbf{L}_k \in M_{m,m}(\mathbb{R})$ with positive diagonal entries where $\mathbf{R}_k = \mathbf{L}_k^\mathsf{T}\mathbf{L}_k$. Furthermore, $\mathbf{L}_k$ is unique and real. Thus, in the AEKF, $\mathbf{R}_k$ is learned by first learning an upper triangular matrix $\mathbf{L}_k \in M_{m,m}(\mathbb{R})$ with positive diagonal entries and then computing $\mathbf{R}_k = \mathbf{L}_k^\mathsf{T}\mathbf{L}_k$.

4.5.3 Scaling the Innovation Sequence

The purpose of $\mathbf{R}_k$ in the Kalman Filter is to provide information on which measurements are reliable and which are unreliable, or which are not outliers and which are. More formally, in (9), the innovation $\tilde{\mathbf{z}}_k$ is scaled by the Kalman gain

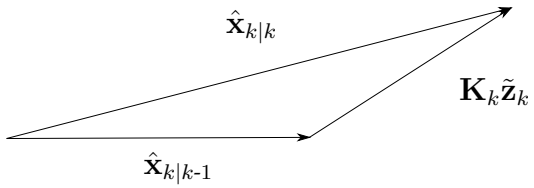


Figure 25: Geometric interpretation of the Kalman Filter's *a posteriori* estimate (9). This figure facilitates a geometric intuition into how the AEKF weights the innovation. We consider a measurement an outlier, or unreliable, if its corresponding innovation has a large norm $\|\tilde{\mathbf{z}}_k\|_2$ and vice-versa for a reliable measurement. In this scenario, if the matrix $\mathbf{K}_k$ does not scale $\tilde{\mathbf{z}}_k$, such that $\|\mathbf{K}_k\tilde{\mathbf{z}}_k\|_2 \ll \|\tilde{\mathbf{z}}_k\|_2$, the *a posteriori* estimate $\hat{\mathbf{x}}_{k|k}$ will be dominated by the outlier measurement $\mathbf{z}_k$. Similarly, if a reliable measurement is scaled such that $\|\mathbf{K}_k\tilde{\mathbf{z}}_k\|_2 \ll \|\tilde{\mathbf{z}}_k\|_2$ the measurement's contribution will be disproportionately reduced. In this case, finding a single $\mathbf{R}_k$, for all measurements presents itself as problematic, especially in cases when the measurement processes distribution is not known, does not exist, or there are significant outliers in the measurements. Since $\mathbf{K}_k$ is a function of $\mathbf{R}_k$ in the AEKF, the intuition is the AEKF learns an $\mathbf{R}_k$ in order that $\mathbf{K}_k$ scales $\tilde{\mathbf{z}}_k$ appropriately.

$\mathbf{K}_k$, which itself is a function of $\mathbf{R}_k$. In this section we take a closer look at this scaling.

Taking the ℓ_2-norm of (9) and applying the triangle inequality gives

$$\left\|\hat{\mathbf{x}}_{k|k}\right\|_2 \leq \left\|\hat{\mathbf{x}}_{k|k\text{-}1}\right\|_2 + \|\mathbf{K}_k\tilde{\mathbf{z}}_k\|_2 \tag{103}$$

This allows us to think about the scaling of $\tilde{\mathbf{z}}_k$ geometrically as shown in Figure 25. If the Kalman Filter is going to ignore a measurement, $\|\mathbf{K}_k\tilde{\mathbf{z}}_k\|_2$ should be very small and vice-versa if a measurement's contribution is significant. Accordingly, we focus on the Kalman gain, $\mathbf{K}_k$, and its effect on $\|\mathbf{K}_k\tilde{\mathbf{z}}_k\|_2$.

The Kalman Gain The Kalman gain, (8), is comprised of three covariance matrices: $\mathbf{P}_{k|k\text{-}1}$, $\mathbf{H}_k\mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^\mathsf{T}$, and $\mathbf{R}_k$. These represent the uncertainty in $\hat{\mathbf{x}}_{k|k\text{-}1}$, $\mathbf{H}_k\hat{\mathbf{x}}_{k|k\text{-}1}$, and $\mathbf{z}_k$ respectively. For example, a large entry in $\mathbf{R}_k$ indicates a large uncertainty in the corresponding entry of $\mathbf{z}_k$ and vice-versa for a small entry. Since the inno-

vation, $\tilde{\mathbf{z}}_k$, is the difference between $\mathbf{z}_k$ and $\mathbf{H}_k\hat{\mathbf{x}}_{k|k\text{-}1}$, $\tilde{\mathbf{z}}_k$ should be weighted by the uncertainty in *both* $\mathbf{z}_k$ and $\mathbf{H}_k\hat{\mathbf{x}}_{k|k\text{-}1}$. The term $\mathbf{S}_k^{-1} = (\mathbf{H}_k\mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^{\mathsf{T}} + \mathbf{R}_k)^{-1}$ in (8) does exactly this, where $\tilde{\mathbf{z}}_k$ is weighted inversely by the covariance of $\mathbf{z}_k$ and $\mathbf{H}_k\hat{\mathbf{x}}_{k|k\text{-}1}$. Thus, if there is a high level of uncertainty in either $\mathbf{z}_k$, $\mathbf{H}_k\hat{\mathbf{x}}_{k|k\text{-}1}$, or both, the effect of $\tilde{\mathbf{z}}_k$ on $\hat{\mathbf{x}}_{k|k}$ in (9) will be reduced accordingly. For high levels of confidence in $\mathbf{H}_k\hat{\mathbf{x}}_{k|k\text{-}1}$ and $\mathbf{z}_k$, a similar analysis follows with the opposite scaling. Similarly, $\mathbf{P}_{k|k\text{-}1}$ in (8) weights the contribution of $\tilde{\mathbf{z}}_k$ based on the reliability of the *a priori* estimate $\hat{\mathbf{x}}_{k|k\text{-}1}$.

4.5.4 Spectral Analysis of the Kalman Gain

This section presents the derivation of our theorem concerning the spectral analysis of the Kalman Gain. After a general introduction to the matrix norm, we begin with some preliminary definitions and lemmas after which we derive the theorem. The definitions and lemmas are principally drawn from [24] and shown here for completeness with some minor additions.

Matrix Norms Just as a vector norm, such as the ℓ_2-norm above, represents the length of a vector, in some scenarios the matrix norm can be thought of as representing the "length" of a matrix. Alternatively, when viewed as a linear operator the matrix norm can be thought of as the largest amount by which a matrix can scale or stretch a vector [59, p. 19]. Corresponding to these two ways of conceptualizing the matrix norm there are element-wise matrix norms and operator matrix norms. Element-wise matrix norms view $M_{m,n}(\mathbb{R})$ as an $m \times n$ dimensional vector space and then apply a well-known vector norm, such as the ℓ_p-norms. For $p = 2$,

the ℓ_2, or Frobenius, element-wise norm is defined for $\mathbf{A} \in M_{m,n}(\mathbb{R})$ as [24, p. 341]

$$\|\mathbf{A}\|_2 = |tr(\mathbf{A}\mathbf{A}^{\mathsf{T}})|^{\frac{1}{2}} = \Big(\sum_{i=1}^{m}\sum_{j=1}^{n}(a_{ij})^2\Big)^{\frac{1}{2}} \tag{104}$$

In this section we focus on the operator matrix norm. The operator matrix norm, instead of considering the element-wise "size" of a matrix, views a matrix $\mathbf{A}$ as a linear operator acting on a vector $\mathbf{x}$ and is defined in terms of the upper bound on the ratio of the norms $\|\mathbf{A}\mathbf{x}\|$ and $\|\mathbf{x}\|$. Note, herein we drop the word "matrix" when referring to the element-wise matrix norm and operator matrix norm.

Definition 1. *A matrix norm is defined as a function $|||\cdot||| : M_{m,n}(\mathbb{R}) \mapsto \mathbb{R}^+$ which, for all matrices $\mathbf{A}, \mathbf{B} \in M_{m,n}(\mathbb{R})$, satisfies the following axioms*

$$|||\mathbf{A}||| \geq 0 \tag{105}$$

$$|||\mathbf{A}||| = 0 \iff \mathbf{A} = 0 \tag{106}$$

$$|||a\mathbf{A}||| = |a|\,|||\mathbf{A}|||, \quad \forall a \in \mathbb{R} \tag{107}$$

$$|||\mathbf{A} + \mathbf{B}||| \leq |||\mathbf{A}||| + |||\mathbf{B}||| \tag{108}$$

Additionally, for $\mathbf{A}, \mathbf{B} \in M_{n,n}(\mathbb{R})$ some matrix norms satisfy [24, p. 341]

$$|||\mathbf{A}\mathbf{B}||| \leq |||\mathbf{A}|||\,|||\mathbf{B}||| \tag{109}$$

Apart from (109), referred to as the submultiplicativity property, the axioms of the matrix norm are identical to the vector norm case.

Definition 2. *For $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} \in M_{m,n}(\mathbb{R})$, and a vector norm $\|\cdot\|$ on $\mathbb{R}^n$ the operator norm*

on $M_{m,n}(\mathbb{R})$ is matrix norm defined as

$$|||\mathbf{A}||| = \sup\{\|\mathbf{Ax}\| : \mathbf{x} \in \mathbb{R}^n \textit{ with } \|\mathbf{x}\| = 1\} \tag{110}$$

$$= \max_{\|x\|=0} \frac{\|\mathbf{Ax}\|}{\|\mathbf{x}\|} \tag{111}$$

$$= \max_{\|x\|=1} \|\mathbf{Ax}\| \tag{112}$$

Definition 3. *$\mathbf{A} \in M_{n,n}(\mathbb{R})$ is orthogonally diagonalizable if it can be written as $\mathbf{A} = \mathbf{U\Lambda U}^\mathsf{T}$, where $\mathbf{U} \in M_{n,n}(\mathbb{R})$ is an orthogonal matrix and $\mathbf{\Lambda} \in M_{n,n}(\mathbb{R})$ is a diagonal matrix with the eigenvalues of $\mathbf{A}$ along the diagonal.*

Lemma 1. *For $\mathbf{A} \in M_{m,n}(\mathbb{R})$ and $\mathbf{y} \in \mathbb{R}^n$ with a vector norm $\|\cdot\|$ on $\mathbb{R}^n$ and operator norm $|||\cdot|||$ on $M_{m,n}(\mathbb{R})$ $\|\mathbf{Ay}\| \leq |||\mathbf{A}||| \, \|\mathbf{y}\|$*

Proof. Given $\mathbf{y} \in \mathbb{R}^n$ and applying definition 2 gives

$$|||\mathbf{A}||| = \max_{\|x\|=1} \|\mathbf{Ax}\| \tag{113}$$

$$\geq \left\| \mathbf{A} \frac{\mathbf{y}}{\|\mathbf{y}\|} \right\| \tag{114}$$

$$= \frac{1}{\|\mathbf{y}\|} \|\mathbf{Ay}\| \tag{115}$$

which gives $\|\mathbf{Ay}\| \leq |||\mathbf{A}||| \, \|\mathbf{y}\|$. □

Lemma 2. *For $\mathbf{A} \in M_{m,n}(\mathbb{R})$ and if the vector norm in (112) is the ℓ_2-norm, the resulting operator norm is $|||\mathbf{A}|||_2 = \sigma_1(\mathbf{A})$ where $\sigma_1(\mathbf{A})$ is the largest singular value of $\mathbf{A}$ and $|||\mathbf{A}|||_2$ is referred to as the spectral norm.*

Proof. For $\mathbf{A}$ with singular value decomposition $\mathbf{A} = \mathbf{U\Sigma V}^\mathsf{T}$, where $\mathbf{U}$ and $\mathbf{V}$ are

orthogonal, $\mathbf{\Sigma}$ is a diagonal matrix $\text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$ containing the singular values of $\mathbf{A}$, and $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_n$.

$$
\begin{aligned}
\left\|\left|\mathbf{A}\right|\right\|_2 &= \max_{\|x\|_2=1} \|\mathbf{A}\mathbf{x}\|_2 && (116)\\
&= \max_{\|x\|_2=1} \left\|\mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}}\mathbf{x}\right\|_2 && (117)\\
&= \max_{\|x\|_2=1} \left\|\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}}\mathbf{x}\right\|_2 && (118)\\
&= \max_{\|\mathbf{V}\mathbf{y}\|_2=1} \|\mathbf{\Sigma}\mathbf{y}\|_2 && (119)\\
&= \max_{\|\mathbf{y}\|_2=1} \|\mathbf{\Sigma}\mathbf{y}\|_2 && (120)\\
&\leq \max_{\|\mathbf{y}\|_2=1} \|\sigma_1(\mathbf{A})\mathbf{y}\|_2 && (121)\\
&= \sigma_1(\mathbf{A}) \max_{\|\mathbf{y}\|_2=1} \|\mathbf{y}\|_2 && (122)\\
&= \sigma_1(\mathbf{A}) && (123)
\end{aligned}
$$

where we use the fact for orthogonal $\mathbf{U} \in M_{n,n}(\mathbb{R})$ and $\mathbf{x} \in \mathbb{R}^n$ that $\|\mathbf{U}\mathbf{x}\|_2 = \|\mathbf{x}\|_2$ and that $\mathbf{U}^{-1} = \mathbf{U}^{\mathsf{T}}$. However the unit vector which maximizes (120) is $\boldsymbol{e}_1$, the canonical unit vector corresponding to $\sigma_1(\mathbf{A})$. Therefore,

$$
\begin{aligned}
\max_{\|\mathbf{y}\|_2=1} \|\mathbf{\Sigma}\mathbf{y}\|_2 &= \max_{\|\boldsymbol{e}_1\|_2=1} \|\mathbf{\Sigma}\boldsymbol{e}_1\|_2 && (124)\\
&= \max_{\|\boldsymbol{e}_1\|_2=1} \|\sigma_1(\mathbf{A})\|_2 && (125)\\
&= \sigma_1(\mathbf{A}) && (126)
\end{aligned}
$$

from which it follows $\left\|\left|\mathbf{A}\right|\right\|_2 = \sigma_1(\mathbf{A})$. □

Note, although both the spectral norm in Lemma 2 and the Frobenius norm (104) have the subscript 2, they are different matrix norms as indicated by the number of

vertical bars. Specifically, the Frobenius norm is an element-wise norm while the spectral norm is an operator norm.

Lemma 3. *If* $\mathbf{A} \in M_{n,n}(\mathbb{R})$ *is symmetric and positive semi-definite then* $\|\mathbf{A}\|_2 = \rho(\mathbf{A})$ *where* $\rho(\mathbf{A})$ *is the largest eigenvalue of* $\mathbf{A}$ *referred to as the spectral radius of* $\mathbf{A}$*.*

Proof. Given $\mathbf{A} \in M_{n,n}(\mathbb{R})$ its singular value decomposition is $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}$ with $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V} \in M_{n,n}(\mathbb{R})$ and

$$\mathbf{A}\mathbf{A}^\mathsf{T} = \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\mathsf{T} \tag{127}$$

However, since $\mathbf{A}$ is symmetric it is orthogonally diagonalizable [24, p. 133]. Choosing orthonormal $\mathbf{U}$ from (127) for the diagonalization

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\mathsf{T} \tag{128}$$

from which we get

$$\mathbf{A}\mathbf{A} = \mathbf{U}\mathbf{\Lambda}^2\mathbf{U}^\mathsf{T} \tag{129}$$

Since $\mathbf{A}$ is symmetric $\mathbf{A}\mathbf{A}^\mathsf{T} = \mathbf{A}\mathbf{A}$ and

$$\mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^\mathsf{T} = \mathbf{U}\mathbf{\Lambda}^2\mathbf{U}^\mathsf{T} \tag{130}$$

By definition the entries of $\mathbf{\Sigma}$ are greater than zero. Using the fact $\mathbf{A}$ is positive semi-definite, $\lambda_i \geq 0$ for each eigenvalue in $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$. From this it follows

$$\sigma_i = \lambda_i \tag{131}$$

for $i = \{1, 2, \ldots, n\}$. As a result $\sigma_1(\mathbf{A}) = \rho(\mathbf{A})$ and we get

$$\left\|\left|\mathbf{A}\right|\right\|_2 = \rho(\mathbf{A}) \tag{132}$$

□

Theorem 1. *For vector norm $\|\cdot\|_2$ and operator norm $\|\|\cdot\|\|_2$, given $\mathbf{K}_k \in M_{n,m}(\mathbb{R})$, $\tilde{\mathbf{z}}_k \in \mathbb{R}^m$, $\mathbf{P}_{k|k\text{-}1} \in M_{n,n}(\mathbb{R})$, $\mathbf{H}_k \in M_{m,n}(\mathbb{R})$, and $\mathbf{S}_k^{-1} \in M_{m,m}(\mathbb{R})$ from Section 2.1*

$$\left\|\mathbf{K}_k\tilde{\mathbf{z}}_k\right\|_2 \leq \rho(\mathbf{P}_{k|k\text{-}1})\sigma_1(\mathbf{H}_k^{\mathsf{T}})\rho(\mathbf{S}_k^{-1})\left\|\tilde{\mathbf{z}}_k\right\|_2 \tag{133}$$

Proof. Using Lemma 1 and $\mathbf{K}_k = \mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^{\mathsf{T}}\mathbf{S}_k^{-1}$ we write

$$\begin{aligned}\left\|\mathbf{K}_k\tilde{\mathbf{z}}_k\right\|_2 &\leq \left\|\left|\mathbf{K}_k\right|\right\|_2\left\|\tilde{\mathbf{z}}_k\right\|_2 &(134)\\ &= \left\|\left|\mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^{\mathsf{T}}\mathbf{S}_k^{-1}\right|\right\|_2\left\|\tilde{\mathbf{z}}_k\right\|_2 &(135)\end{aligned}$$

Applying the submultiplicativity property of the matrix norm (109) and Lemma 2 gives

$$\begin{aligned}\left\|\mathbf{K}_k\tilde{\mathbf{z}}_k\right\|_2 &\leq \left\|\left|\mathbf{P}_{k|k\text{-}1}\right|\right\|_2\left\|\left|\mathbf{H}_k^{\mathsf{T}}\right|\right\|_2\left\|\left|\mathbf{S}_k^{-1}\right|\right\|_2\left\|\tilde{\mathbf{z}}_k\right\|_2 &(136)\\ &= \sigma_1(\mathbf{P}_{k|k\text{-}1})\sigma_1(\mathbf{H}_k^{\mathsf{T}})\sigma_1(\mathbf{S}_k^{-1})\left\|\tilde{\mathbf{z}}_k\right\|_2 &(137)\end{aligned}$$

Lastly, since $\mathbf{P}_{k|k\text{-}1}$ and $\mathbf{S}_k^{-1}$ are symmetric positive semi-definite matrices applying Lemma 3 we get

$$\left\|\mathbf{K}_k\tilde{\mathbf{z}}_k\right\|_2 \leq \rho(\mathbf{P}_{k|k\text{-}1})\sigma_1(\mathbf{H}_k^{\mathsf{T}})\rho(\mathbf{S}_k^{-1})\left\|\tilde{\mathbf{z}}_k\right\|_2 \tag{138}$$

□

Theorem 1 gives us an upper bound on $\|\mathbf{K}_k\tilde{\mathbf{z}}_k\|_2$ completely in terms of the spectral radii of $\mathbf{P}_{k|k\text{-}1}$ and $\mathbf{S}_k^{-1}$, the largest singular value of $\mathbf{H}_k^\mathsf{T}$, and the ℓ_2-norm of $\tilde{\mathbf{z}}_k$. Furthermore, at time k the only term on the right hand side of (138) which is a function of the learned measurement noise covariance matrix $\mathbf{R}_k$ is $\mathbf{S}_k^{-1}$. *Thus, if the AEKF is learning* $\mathbf{R}_k$ *correctly, the scaling of* $\tilde{\mathbf{z}}_k$ *by* $\mathbf{K}_k$ *is such that the a priori estimate at time* $k+1$ *will not be unduly influenced by outlier measurements, while giving proper weighting to reliable measurements.*

4.5.5 Testable Criteria

In order to investigate the relation between $\rho(\mathbf{S}_k^{-1})$ and $\|\tilde{\mathbf{z}}_k\|_2$ we rewrite (138) as

$$\rho(\mathbf{S}_k^{-1}) \geq \frac{\|\mathbf{K}_k\tilde{\mathbf{z}}_k\|_2}{\|\tilde{\mathbf{z}}_k\|_2}\left(\rho(\mathbf{P}_{k|k\text{-}1})\sigma_1(\mathbf{H}_k^\mathsf{T})\right)^{-1} \tag{139}$$

which, for fixed $\tilde{\mathbf{z}}_k$, $\mathbf{P}_{k|k\text{-}1}$ and $\mathbf{H}_k^\mathsf{T}$, indicates two things. First, the scaling of $\tilde{\mathbf{z}}_k$ by $\mathbf{K}_k$ is upper bounded by $\rho(\mathbf{S}_k^{-1})$. Secondly, from this we see there is an inverse relationship between the upper bound $\rho(\mathbf{S}_k^{-1})$ and $\|\tilde{\mathbf{z}}_k\|_2$. Based on this we hypothesize that the equality relationship between $\rho(\mathbf{S}_k^{-1})$ and $\tilde{\mathbf{z}}_k$ is of the form

$$\rho(\mathbf{S}_k^{-1}) = \alpha_k \|\tilde{\mathbf{z}}_k\|_2^{\beta_k} \tag{140}$$

Equation (140) presented here is an experimentally testable criteria based upon the above mathematical analysis. As the validity of (140) will be determined by data, in order to estimate a single α and β, for all $k = 1, \ldots, N$, we perform linear regression on the log-log transformation of (140) [27]. The first step is to write

$$\rho(\mathbf{S}_k^{-1}) = (\epsilon_k)\alpha \|\tilde{\mathbf{z}}_k\|_2^{\beta} \tag{141}$$

where the error incurred by estimating each α_k and β_k with the same α and β is accounted for by the random term ϵ_k. Taking the log of both sides of (141) gives

$$\log \rho(\mathbf{S}_k^{-1}) = \beta \log \|\tilde{\mathbf{z}}_k\|_2 + \log(\alpha) + \log(\epsilon_k) \tag{142}$$

and then taking the expectation of both sides of (142) results in

$$\mathbb{E}\big[\log \rho(\mathbf{S}_k^{-1})\big] = \beta \log \|\tilde{\mathbf{z}}_k\|_2 + \log(\alpha) \tag{143}$$

where it is assumed $\mathbb{E}\big[\log(\epsilon_k)\big] = 0$. We are now in a position to estimate α and β using linear regression. In particular, we are interested in β as we want to test whether the AEKF, by learning $\{\mathbf{R}_k\}_{k=1}^N$, scales outliers via $\rho(\mathbf{S}_k^{-1})$ in inverse proportion to $\|\tilde{\mathbf{z}}_k\|_2$. Looking at (141), the value of β indicates what kind of relation there is between $\rho(\mathbf{S}_k^{-1})$ and $\|\tilde{\mathbf{z}}_k\|_2$. Specifically,

- $\beta = 0 \implies$ no relationship
- $\beta < 0 \implies$ inversely proportional relationship
- $\beta > 0 \implies$ directly proportional relationship

Thus, we hypothesize that for the AEKF $\beta < 0$ and for the Kalman Filter $\beta \approx 0$. Thus, the upper bound on $\|\mathbf{K}_k\tilde{\mathbf{z}}_k\|_2$ decays inversely with respect to $\|\tilde{\mathbf{z}}_k\|_2$ in the AEKF, while it is fixed for the Kalman Filter.

4.5.6 Experimental Setup

Datasets All experiments were performed on simulated data with domain randomization as in previous sections. However, here the simulated dataset was generated using two-dimensional ground truth curves. Of course, the number of di-

mensions is not limited to two but this was selected to ensure $\mathbf{R}_k$ in the standard Kalman Filter was not a scalar.

Our ground truth curves consisted of random truncated Taylor polynomials as in Section 4.4. In this section, only Cauchy noise was used as one of the standing questions from Section 4.4 was to explain why the AEKF performed much better than a standard Kalman Filter in terms of mitigating outliers.

Model Parameters and Test Procedures Training the AEKF consisted of training a variety of models with varying layer sizes and learning rates. Optimization was done using the Adam optimizer [32] with ReLU [49] as the activation function. The testing set consisted of 1,000 random curves, generated as in previous sections, that were saved to disk. This is to ensure final evaluation of all AEKF models was done on the same dataset.

In terms of the AEKF's internal Kalman Filter, the measurement vectors were in $\mathbb{R}^2$, $\mathbb{R}^4$, and $\mathbb{R}^8$, the process noise covariance matrix was a diagonal matrix $\mathbf{Q}_k = 1e\text{-}2\mathbf{I}$, and an NCV dynamical model was used. Recall, as the AEKF first processes the actual measurements via the encoder portion, the dimensions of the Kalman Filter in the AEKF's latent layer do not have to match the measurement dataset dimension. That is, given a sequence of measurements $\{\boldsymbol{\phi}_k\}_{k=1}^N$, with each $\boldsymbol{\phi}_k \in \mathbb{R}^m$, the encoder portion of the AEKF transforms this via

$$E : \{\boldsymbol{\phi}_k\}_{k=1}^N \mapsto \left[\{\mathbf{z}_k\}_{k=1}^N, \{\mathbf{R}_k\}_{k=1}^N\right] \tag{144}$$

where each $\mathbf{z}_k \in \mathbb{R}^n$ and $\mathbf{R}_k \in M_{n,n}(\mathbb{R})$ and where m does not necessarily equal n. It is then the role of the decoder portion of the AEKF to ensure the output of the Kalman Filter portion is mapped back to the original measurement dimensions.

Table 12: Estimates of α and β in (141) along with correlation coefficients between $\log \|\tilde{\mathbf{z}}_k\|_2$ and $\log \rho(\mathbf{S}_k^{-1})$ and p-values (given a null hypothesis that the slope is zero). α and β are estimated, via linear regression, from the log transformed data based upon (143), while the correlation coefficient indicates the strength of the linear relationship between $\log \|\tilde{\mathbf{z}}_k\|_2$ and $\log \rho(\mathbf{S}_k^{-1})$. The number in the model column indicates the dimensions of the Kalman Filter measurement space and the MSE column indicates the average MSE over each of the 1,000 test set samples. The MSE is computed using the actual data and the fitted curve defined by the appropriate α and β.

MODEL	MSE	α	β	Corr. Coeff.	p-value
AEKF 2	2.75e-01	2.96e-03	-4.80e-01	-7.61e-01	1.26e-12
AEKF 4	2.44e-01	3.68e-03	-6.12e-01	-8.85e-01	1.03e-15
AEKF 8	2.25e-01	7.06e-02	-1.05e+00	-9.08e-01	3.04e-16
KF 2	5.72e+01	2.14e-02	5.06e-04	1.53e-02	2.71e-01

The standard Kalman Filter's process noise covariance matrix, $\mathbf{Q}_k$, and dynamical model were the same as the AEKF's. The measurement noise covariance, $\mathbf{R}_k$, was estimated from the data by

$$\mathbf{R}_k = \frac{(\mathbf{z} - \mathbf{z}_\mu)^\intercal (\mathbf{z} - \mathbf{z}_\mu)}{N - 1} \tag{145}$$

where $\mathbf{z} \in \mathbb{R}^{N \times 2}$ is the measurement vector and $\mathbf{z}_\mu \in \mathbb{R}^2$ is the mean of $\mathbf{z}$. Although it is impossible to write down a single $\mathbf{R}$, as this does not exist for the Cauchy distribution, for a finite dataset drawn from a Cauchy distribution a sample covariance can be estimated as in (145). However, the presence of outliers still makes using a single $\mathbf{R}$ problematic. *In this light, the fact the AEKF learns a point-wise covariance matrix sequence* $\{\mathbf{R}_k\}_{k=1}^N$ *is one of its most appealing features*.

Table 13: Estimates of α and β in (141) along with correlation coefficients between $\log \|\tilde{\mathbf{z}}_k\|_2$ and $\log \rho(\mathbf{P}_{k|k\text{-}1})$ and p-values. See the caption in table 12 for details.

MODEL	MSE	α	β	Corr. Coeff.	p-value
AEKF 2	2.75e-01	7.85e+01	2.29e-02	5.41e-02	4.16e-01
AEKF 4	2.44e-01	1.21e+02	-6.93e-03	-1.81e-02	5.07e-01
AEKF 8	2.25e-01	1.60e+02	-3.27e-02	-5.04e-02	5.13e-01
KF 2	5.72e+01	7.68e+01	9.39e-02	1.40e-01	2.52e-01

4.5.7 Experimental Results

Recall our hypothesis is that for the AEKF $\beta < 0$ and for the Kalman Filter $\beta \approx 0$, which both follow from the discussion above. That is, since the AEKF should learn small $\rho(\mathbf{S}_k^{-1})$ for unreliable data and vice-versa for reliable data, we predict an inverse relationship between $\|\tilde{\mathbf{z}}_k\|_2$ and $\rho(\mathbf{S}_k^{-1})$. This would be supported by $\beta < 0$. For the Kalman Filter, since $\rho(\mathbf{S}_k^{-1})$ is independent of $\tilde{\mathbf{z}}_k$, we predict the slope in the log-log space to be approximately zero. This would indicate $\|\tilde{\mathbf{z}}_k\|_2$ does not vary significantly with $\rho(\mathbf{S}_k^{-1})$.

Figures 26(a) and 26(b) plot $\rho(\mathbf{S}_k^{-1})$ vs. $\|\tilde{\mathbf{z}}_k\|_2$ for a sample test set trial for the Kalman Filter and AEKF respectively. Similarly, Figures 26(c) and 26(d) show the associated log-log plots, corresponding to (143), for both the Kalman Filter and AEKF. Notice in Figures 26(a) and 26(b), for the AEKF there is a clear inverse relationship, while for the Kalman Filter there is a near-zero slope relationship. Corresponding to this, in Figures 26(c) and 26(d) we see for the AEKF there is a clear linear relationship, while for the Kalman Filter the only pattern that emerges is effectively a flat line. This trend was observed across all trials in the data set, the statistics of which are shown in Table 12. Here the reported values of MSE, α, β, the correlation coefficient, and p-value are the average of each parameter over the

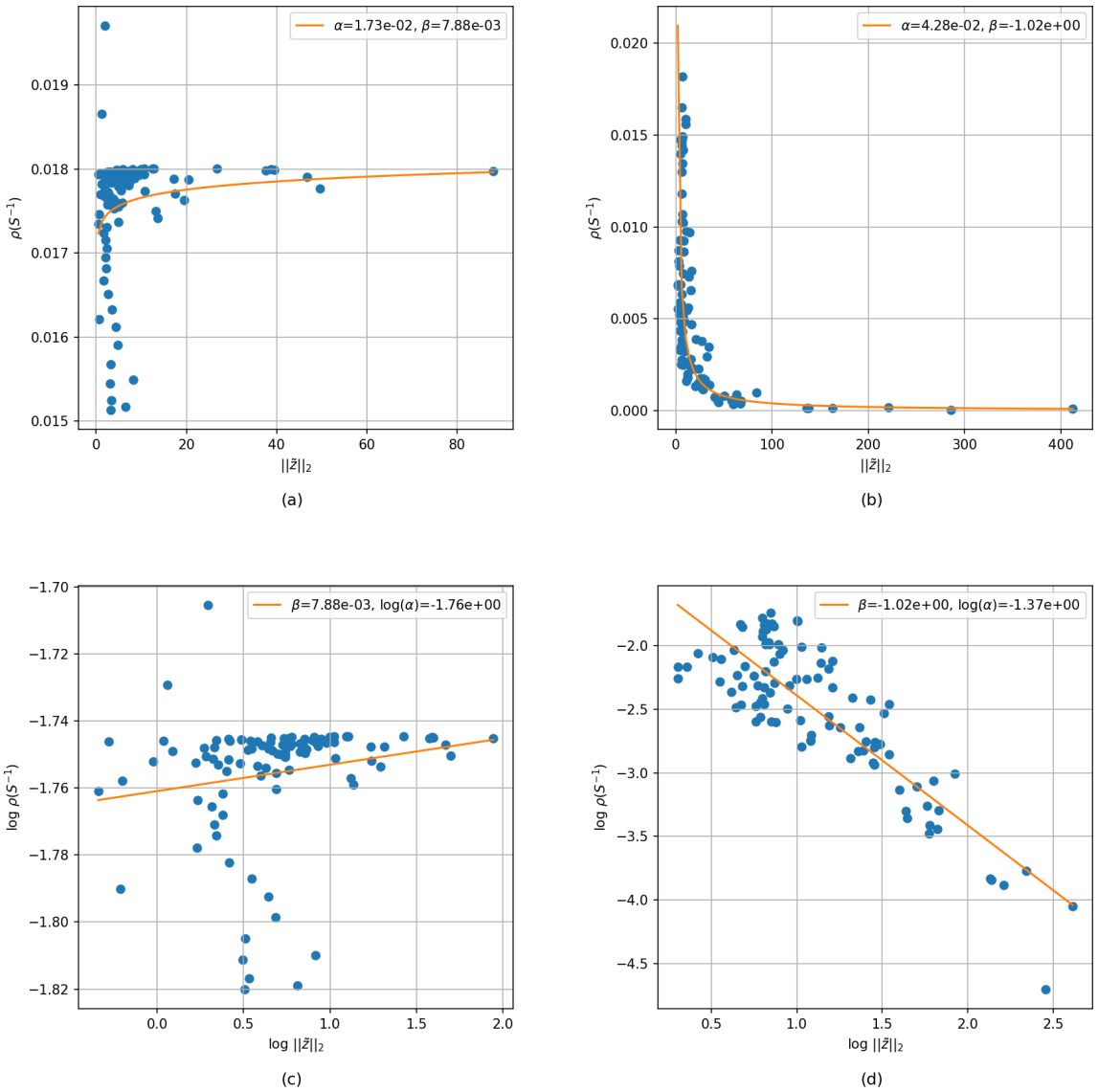


Figure 26: Single test set trial sample plot of $\rho(\mathbf{S}_k^{-1})$ vs. $\|\tilde{\mathbf{z}}_k\|_2$ for (a) Kalman Filter and (b) AEKF, along with the corresponding log-log plots for (c) the Kalman Filter and (d) AEKF. The related estimated parameters appear in the legends. The correspondence between AEKF plots (b) and (d) is consistent with our hypothesis in that an inverse relation in the $\rho(\mathbf{S}_k^{-1})$ vs. $\|\tilde{\mathbf{z}}_k\|_2$ space should correspond to a linear relationship in the log-log space. This is not observed for the Kalman Filter as both (a) and (c) indicate no relationship as the slope is effectively zero.

1,000 trails in the test set. The MSE is computed using the actual data and the fitted curve defined by the appropriate α and β. In addition to β, the correlation coef-

ficient is included as β was estimated on the assumption of a linear relationship between $\log \rho(\mathbf{S}_k^{-1})$ and $\log \|\tilde{\mathbf{z}}_k\|_2$, and the correlation coefficient characterizes the strength of this linear relationship. The p-value tests the null hypothesis that the slope is zero.

The results in Table 12 are consistent with our hypothesis that for the AEKF $\beta < 0$ and for the Kalman Filter $\beta \approx 0$. Furthermore, these results are not surprising given that we know, for a standard Kalman Filter, $\rho(\mathbf{S}_k^{-1})$ is independent of $\mathbf{z}_k$. However, $\rho(\mathbf{S}_k^{-1})$ is somewhat affected by $\mathbf{H}_k$ and $\hat{\mathbf{x}}_{k|k-1}$, which may account for β not being equal to zero. Still, the size of the β values for the Kalman Filter models provide empirical evidence that a standard Kalman Filter is not adjusting its scaling of $\tilde{\mathbf{z}}_k$ based upon $\mathbf{z}_k$. In the AEKF case, we see a clear inverse relation between $\|\tilde{\mathbf{z}}_k\|_2$ and $\rho(\mathbf{S}_k^{-1})$. The correlation coefficients also support our hypothesis in that all AEKF values are between -0.76 and -0.91 while all Kalman Filter values show little correlation between $\|\tilde{\mathbf{z}}_k\|_2$ and $\rho(\mathbf{S}_k^{-1})$. Also note the AEKF and Kalman Filter MSE values are two orders of magnitude apart. This difference in performance can be explained by the fact that in the AEKF, the scaling of $\|\tilde{\mathbf{z}}_k\|_2$ by $\mathbf{K}_k$ is bound from above by the learned $\rho(\mathbf{S}_k^{-1})$.

Given that $\mathbf{H}_k$ is fixed in the AEKF and Kalman Filter models, (138) indicates there is the possibility the scaling of $\|\tilde{\mathbf{z}}_k\|_2$ also results from $\rho(\mathbf{P}_{k|k\text{-}1})$. To demonstrate this alone is not responsible for the scaling of reliable and unreliable measurements, we show in Table 13 that $\rho(\mathbf{P}_{k|k\text{-}1})$ does not follow the same inverse relationship with $\|\tilde{\mathbf{z}}_k\|_2$ as $\rho(\mathbf{S}_k^{-1})$. In Table 13 the β values for the AEKF are at least an order of magnitude less than those for $\rho(\mathbf{S}_k^{-1})$. The Kalman Filter results are as expected as $\rho(\mathbf{P}_{k|k\text{-}1})$, similar to $\rho(\mathbf{S}_k^{-1})$, does not depend on $\mathbf{z}_k$.

From these results we conclude there is significant empirical evidence to support our

claim the AEKF learns $\{\mathbf{R}_k\}_{k=1}^N$ *such that*

$$\rho(\mathbf{S}_k^{-1}) \propto \|\tilde{\mathbf{z}}_k\|_2^{\beta}, \quad \beta < 0 \tag{146}$$

while this behavior is, unsurprisingly, not observed for a standard Kalman Filter.

Comparing $\|\mathbf{K}_k\tilde{\mathbf{z}}_k\|_2$ in the Kalman Filter and AEKF It is an interesting property of the Kalman Filter that $\mathbf{P}_{k|k\text{-}1}$ and $\mathbf{S}_k$, and hence $\mathbf{K}_k$, can be completely determined independent of the measurements $\mathbf{z}_k$. With this in mind we substitute (138) into (103)

$$\begin{aligned} \left\|\hat{\mathbf{x}}_{k|k}\right\|_2 &\leq \left\|\hat{\mathbf{x}}_{k|k\text{-}1}\right\|_2 + \|\mathbf{K}_k\tilde{\mathbf{z}}_k\|_2 &(147)\\ &\leq \left\|\hat{\mathbf{x}}_{k|k\text{-}1}\right\|_2 + \rho(\mathbf{P}_{k|k\text{-}1})\rho(\mathbf{S}_k^{-1})\sigma_1(\mathbf{H}_k^{\mathsf{T}})\left\|\tilde{\mathbf{z}}_k\right\|_2 &(148)\end{aligned}$$

For fixed $\mathbf{H}_k$, in a standard Kalman Filter, since $\mathbf{P}_{k|k\text{-}1}$ and $\mathbf{S}_k$ are independent of $\mathbf{z}_k$, the scaling of $\|\tilde{\mathbf{z}}_k\|_2$ in (148) is also independent of $\mathbf{z}_k$. This helps explain why a standard Kalman Filter, with fixed $\mathbf{R}_k$, is unable to mitigate large outliers, such as those seen in the Cauchy noise draws in Sections 4.3 and 4.4. As the coefficient that could possibly reduce the effect of outliers is determined independent of the measurements, the upper bound on the *a posteriori* estimate is unaffected by the actual measurements $\{\mathbf{z}_k\}_{k=1}^N$. However, in the AEKF, since both $\{\mathbf{z}_k\}_{k=1}^N$ and $\{\mathbf{R}_k\}_{k=1}^N$ are learned, the scaling of $\|\tilde{\mathbf{z}}_k\|_2$ in (148) *is a function of* $\mathbf{z}_k$. This allows for the possibility the AEKF will scale reliable and unreliable values of $\|\tilde{\mathbf{z}}_k\|_2$ accordingly.

Here it should be pointed out the innovation, $\tilde{\mathbf{z}}_k = \mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k\text{-}1}$, in (148) is not the same for the standard Kalman Filter and AEKF. Following the notation in Section 2.5.2, if we label the actual measurements as ϕ, then for the Kalman Filter,

$\mathbf{z} \triangleq \boldsymbol{\phi}$. In the AEKF, based upon (59)-(62), $\mathbf{z}$ is not equivalent to $\boldsymbol{\phi}$ but is given by (62). That is

$$\mathbf{z} = \mathcal{A}_z \circ \mathbf{E}_S \circ \mathbf{E}_{S-1} \circ \cdots \circ \mathbf{E}_1(\boldsymbol{\phi}) \tag{149}$$

where $\mathcal{A}_z$ is the affine transformation in (62). Based on this analysis, in the AEKF, any dependence of the scaling of $\tilde{\mathbf{z}}_k$ by $\mathbf{K}_k$ on the actual measurements $\boldsymbol{\phi}_k$ is mediated through $\mathbf{R}_k$.

4.5.8 Summary

In this section we hypothesized an experimentally testable criteria to help explain the AEKF's outlier mitigation abilities, based upon a theorem derived from a matrix analysis of the Kalman Filter. Experiments demonstrated the AEKF does in fact behave as indicated by the theorem, leading to some theoretical insight into "what" the deep learning portions of the AEKF are doing. Specifically, the AEKF learns an inverse relationship between the spectral radius of $\mathbf{S}_k^{-1}$, $\rho(\mathbf{S}_k^{-1})$, and the norm of the innovation $\|\tilde{\mathbf{z}}_k\|_2$.

4.6 Maneuvering Target Tracking Using the Autoencoder-Interacting Multiple Model Kalman Filter

(Co-written with Kirty P. Vedula and Professor Donald R. Brown of Worcester Polytechnic Institute)

In this section we apply the AEIMMKF from Section 2.6 to simulated scenarios where target dynamics are known to be represented by multiple dynamical models. Building on the success of the AEKF, the AEIMMKF is seen as an augmentation in the same sense that the standard IMMKF is an augmentation of the

standard Kalman Filter. In particular, while the AEKF addresses the issue of estimating the measurement noise covariance, the AEIMMKF addresses both the measurement noise covariance estimation and choosing the appropriate system dynamics as discussed in Section 2.2

In this section, we apply the AEIMMKF to a simulated flight data tracking problem where the IMMKF consists of two NCA dynamical models. We compare the AEIMMKF against a standard Kalman Filter, IMMKF, LSTM, and AEKF algorithms. The improvements that we obtain using AEIMMKF in tracking maneuvering targets are particularly useful in some tracking applications such as air traffic control as discussed in [4].

4.6.1 Test Protocol

We train the above models using domain randomization on simulated flight paths consisting of constant velocity segments interspersed with coordinated turns. All simulated flight paths begin with constant velocity motion in the horizontal direction. At each turn, the corresponding turn radius is chosen randomly (within a predefined range), along with the turn direction (clockwise or counter clockwise). Either Gaussian or Cauchy noise is then added to these smooth ground truth flight paths, which results in our simulated flight path for domain randomization. Each of the four models' state estimation is then compared with the actual ground truth and the corresponding MSE is reported for each model. Note that in this phase, the ground truth is used *only for model evaluation as in previous sections* and does not affect each model's state estimate in any way. As the state estimate and ground truth are two dimensional, the MSE is calculated by taking the square root of the Frobenius norm between the state estimate and ground truth.

The simulated flight paths are based on physical models according to the kinematics equations for constant linear motion and coordinated turns. The (discrete) time evolution position and velocity vectors for the linear model are defined as

$$r(t+dt) = \big(x(t) + v_x dt\big)\hat{i} + \big(y(t) + v_y dt\big)\hat{j} \tag{150}$$

$$v(t+dt) = v_x\hat{i} + v_y\hat{j} \tag{151}$$

where $r(t) = (x(t), y(t))$ is the position vector at time t, $v_0 = (v_x, v_y)$ is the initial velocity at the beginning of the segment, and $\hat{i}$ and $\hat{j}$ represent the unit vectors in the horizontal and vertical directions respectively. The coordinated turn model is similarly defined by

$$r(t+dt) = R\cos\Big(\frac{\|v_0\|_2\,(kdt)}{R}\Big)\hat{i} + R\sin\Big(\frac{\|v_0\|_2\,(kdt)}{R}\Big)\hat{j} \tag{152}$$

$$v(t+dt) = -\|v_0\|_2\sin\Big(\frac{\|v_0\|_2\,(kdt)}{R}\Big)\hat{i} + \|v_0\|_2\sin\Big(\frac{\|v_0\|_2\,(kdt)}{R}\Big)\hat{j} \tag{153}$$

where R is the radius of the turn, $\|v_0\|_2$ is initial speed at the beginning of the turn, and k indexes the increment of the full turn. That is, for a turn with an angle θ, dividing the turn into N equal increments gives $\theta = N\delta\theta$. Since $\theta = \frac{\|v_0\|_2 t}{R}$ we can express the total angle turned through at the k^{th} increment as $k\delta\theta = \frac{\|v_0\|_2}{R}k\delta t$ with $k = 0, 2, ..., N-1$. Note that (152) and (153) assume rotation about the origin starting from $\theta = 0$. Thus, appropriate translation and rotations were applied to the results of (152) and (153) to ensure the turns occurred at the correct location and were continuous, up to the first derivative, with their incoming and outgoing linear segments. A sample flight path with Gaussian noise is shown in Figure 27.

While we use simulated flight paths to demonstrate the efficacy of the AEIMMKF, it should be noted the AEIMMKF is designed in a general manner to make it ap-

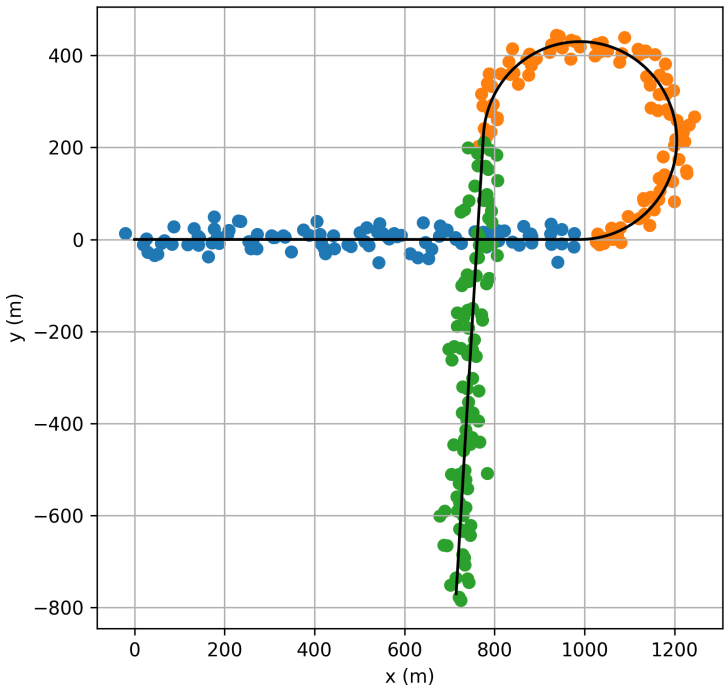


Figure 27: Sample Simulated Flight Path with Gaussian Noise. Each leg of the path are indicated by the different colors.

plicable to any application or scenario where an IMMKF is appropriate.

4.6.2 Single Turn Results

Here we present results for single turn flight paths with an initial velocity in the horizontal direction of 100 m/s, a turn radius uniformly selected between 200 and 300 meters and added Gaussian or Cauchy noise. The Gaussian noise was drawn from $\mathcal{N}(0, 20)$ while the Cauchy noise was drawn from a standard Cauchy distribution which was then scaled by a factor of 5. Each of the three flight segments lasts 10 (simulated) seconds with a sampling frequency of 10 Hz. The transition probabilities, defined in (30), for the AEIMMKF and IMMKF are 0.9 and 0.1. That is, the probability the IMMKF is in one NCA mode at time $k + 1$ given it was in the same NCA mode at time k is 0.9, while the probability the IMMKF switches

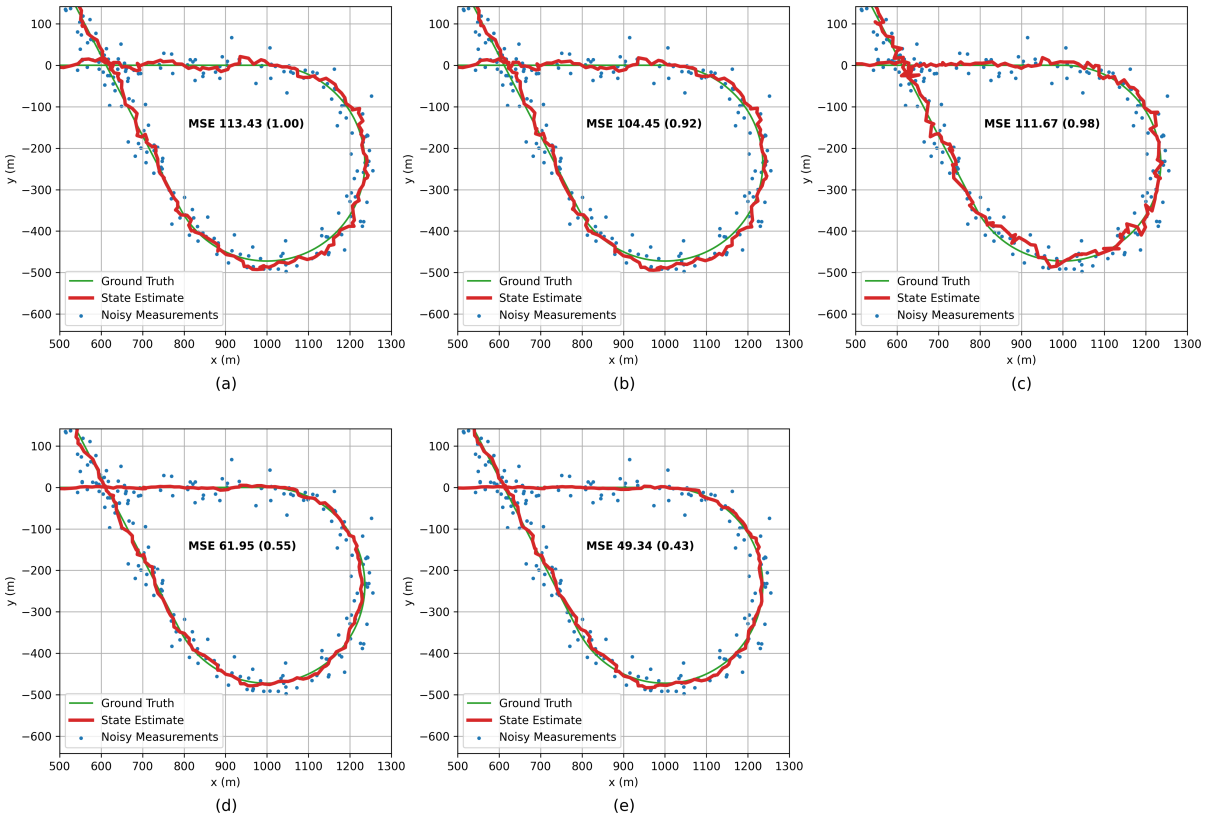


Figure 28: Turn segment from a Gaussian noise test set sample trial. Here the (a) Kalman Filter, (b) IMMKF, and (c) LSTM estimates have large MSE values and, generally, are less smooth than the (d) AEKF and (e) AEIMMKF estimates. Furthermore, the Kalman Filter, IMMKF, and LSTM have more difficulty estimating the ground truth on the turn than the AEKF and AEIMMKF. The number in parenthesis following the MSE is the ratio of the given model's MSE to the Kalman Filter's MSE.

between NCA modes is 0.1. The process noise covariance matrices for both the AEIMMKF and IMMKF were $0.5\mathbf{I}$ and $0.05\mathbf{I}$. The Kalman Filter and AEKF models consisted of single NCA models with process noise covariance matrix $0.5\mathbf{I}$.

For the models trained on Gaussian noise, the test set consisted of 1000 simulated single turn flight paths. For each model, the reported MSE is computed by averaging the MSE on each of the 1000 test paths computed using the ground truth and state estimates. Results for the Gaussian noise tests are shown in Table

Table 14: Single turn test set MSE results with Gaussian noise. The top two performing models are the AEKF and AEIMMKF while Kalman Filter, IMMKF, and LSTM performances are all comparable. The MSE ratio is the ratio of each models MSE to the Kalman Filters MSE.

Model	MSE	Ratio
KF	118.74	1.00
IMMKF	113.97	0.96
LSTM	124.33	1.05
AEKF	90.09	0.76
AEIMMKF	**67.06**	**0.56**

14, where the MSE ratio is the ratio of each model's MSE to the Kalman Filter's MSE. Here the models, from highest to lowest MSE, are the LSTM, Kalman Filter, IMMKF, AEKF, and AEIMMKF. The fact that the IMMKF shows better performance than the Kalman Filter is not surprising. The LSTM performance was very close to both the Kalman Filter and IMM. However, all these models are improved upon by the AEKF and AEIMMKF. For visualization, a turn segment of one trial from the Gaussian noise test set with the ground truth, noisy simulated measurements, and state estimate is shown in Figure 28.

We also trained and tested the same five models with Cauchy noise and noticed some differences in the results. Apart from the presence of Cauchy noise, the test set used here was generated in the same manner as the Gaussian test set. Unsurprising, the Kalman Filter and IMM performed very poorly. Compared with the Gaussian noise tests, the LSTM performed much better relative to the Kalman Filter and IMM. However, the LSTM model still has an MSE approximately three times greater than both the AEKF and the AEIMMKF, with the AEIMMKF achieving the lowest MSE of all five models. The test set MSE results for these tests are shown in Table 15.

Table 15: Single turn test set MSE results with Cauchy noise. The top two performing models were the AEKF and AEIMMKF, with the LSTM having an MSE approximately three times greater than the AEKF and AEIMMKF. Unsurprisingly, both the Kalman Filter and IMM performed poorly. The MSE ratio is the ratio of each models MSE to the Kalman Filters MSE.

Model	MSE	Ratio
KF	4,202,564	1.0
IMMKF	2,177,604	0.52
LSTM	1513.95	3.6e-4
AEKF	515.16	1.23e-4
AEIMMKF	**454.04**	**1.08e-4**

4.6.3 Summary

Based on the success of the AEKF in Sections 4.2, 4.3, and 4.4, in this section we extended the AEKF to the AEIMMKF, to solve challenging maneuvering target tracking problems. We provided a proof-of-concept demonstration with simulated flight tracking data and compare it against state-of-the-art methods in tracking such as the IMMKF. These results further indicate the combination of deep learning and traditional filtering techniques outperform traditional filtering approaches by themselves.

4.7 Hilbert Space Filter

In the AEKF, the state estimate of ϕ, represented by $\hat{\phi}$, is a point-wise sequence $\{\hat{\phi}_k\}_{k=1}^N$. Thus, N samples or points are needed to represent the state estimate of a noisy sequence $\{\phi_k\}_{k=1}^N$. However, the ground truth that the AEKF is training on is a smooth function. The question then arises whether we can represent this ground truth function with a set of parameters $\{\alpha_i\}_{i=1}^P$, where P is less than N. One possible answer to this question is to use Hilbert Space representations of functions from Section 3.1.

Given the orthonormal basis functions $\{\boldsymbol{\varphi}_i\}_{i=1}^{\infty}$ discussed in 3.1, if a neutral network can be trained to learn α_i for $i = \{1, 2, \ldots, P\}$ for $P < \infty$, then each $\hat{\boldsymbol{\phi}}_k$ in $\{\hat{\boldsymbol{\phi}}_k\}_{k=1}^{N}$ would no longer be a point-wise representation of the ground truth, but a global representation. That is, an approximation of the actual functional form of the ground truth is being learned, not simply a point-by-point estimate of it. This is the idea behind the Hilbert Space Filter, an ongoing area of research whose preliminary results we present here.

Note that while strictly speaking each element of $\{\boldsymbol{\varphi}_i\}_{i=1}^{\infty}$ is a function, since in computation we sample each at a finite number of points, we use the bold vector notation in this section. Furthermore, since we will be representing the vector $\hat{\boldsymbol{\phi}}$ as a linear combination of the orthonormal vectors in $\{\boldsymbol{\varphi}_i\}_{i=1}^{\infty}$, the bold vector notation lends further intuition.

Since the sum in (77) contains a countably infinite number of terms, although all the coefficients α_i can not be computed in practice, the truncated version of (77) can be shown to well-approximate $f(x)$ by

$$\left\| f(x) - \sum_{i=1}^{P} \alpha_i \boldsymbol{\varphi}_i(x) \right\|_2^2 < \epsilon \tag{154}$$

where $\|\cdot\|_2^2 = \langle \cdot, \cdot \rangle$ is the $\mathcal{L}_2$-norm in (80) and $\epsilon > 0$. It is this truncated Hilbert Space representation of a function we use in the Hilbert Space Filter. Although the state estimation of the function families in Sections 4.2 through 4.4 can be conceptualized as living in different regions of Hilbert Space, the actual coefficients $\{\alpha_i\}_{i=1}^{P}$ of the Hilbert Space representation of these functions was never explicitly learned by a neural network. With the Hilbert Space Filter, we explicitly learn these coefficients representing the functional form of the state estimation.

4.7.1 Test Protocol

Similarly to how the AEKF was trained with domain randomization, the Hilbert Space Filter learns a mapping from noisy measurements $\boldsymbol{\phi}_k$ to $\hat{\boldsymbol{\phi}}_k = \sum_{i=0}^{P} \alpha_i^k \boldsymbol{\varphi}_i(x)$, using the same cost function (74) as the AEKF, where $\{\alpha_i\}_{i=1}^{P}$ is the final output of the neural network. However, in the AEKF and LSTM models each row of the input $\boldsymbol{\phi}$ represents a sample from the time series and the columns represent the number of sensors or channels per sample. The Hilbert Space Filter is somewhat different in that it is mapping a discrete sampling of a noisy function to a representation of its ground truth, characterized uniquely by a small number of parameters $\{\alpha_i\}_{i=1}^{P}$, where P is much less than the discrete sampling.

In our experiments, each row of $\boldsymbol{\phi}$ was a randomly generated truncated Taylor Polynomial (as in Section 4.4) with added Gaussian noise also drawn from $\mathcal{N}(0, 0.2)$. Each function defined by a row of $\boldsymbol{\phi}$ was evaluated on 100 sampled points in $[-1, 1]$, thus the domain of each row in $\boldsymbol{\phi}$ is defined as $x \in \mathbb{R}^{100\times 1}$. With 128 randomly generated functions per epoch, the input to the Hilbert Space Filter was $\boldsymbol{\phi} \in \mathbb{R}^{128\times 100}$. The network consisted of an input layer, two hidden layers, and an output layer, with each mapping between layers consisting of a composition of an affine transformation and a leaky ReLU [37] activation function. It was trained for 100,000 epochs with a learning rate of 10^{-3}. Representing the mapping of the entire network as $\hat{\boldsymbol{\phi}}$, we express the function learned by the Hilbert Space Filter as

$$\hat{\boldsymbol{\phi}} \triangleq \alpha \circ H_2 \circ H_2 \circ \boldsymbol{\phi}(x) \tag{155}$$

where H_i is the i^{th} hidden layer and α the output layer that produces the coefficients $\{\alpha_i\}_{i=1}^{P}$ in (154) for each row in $\boldsymbol{\phi}$. For the corresponding orthonormal

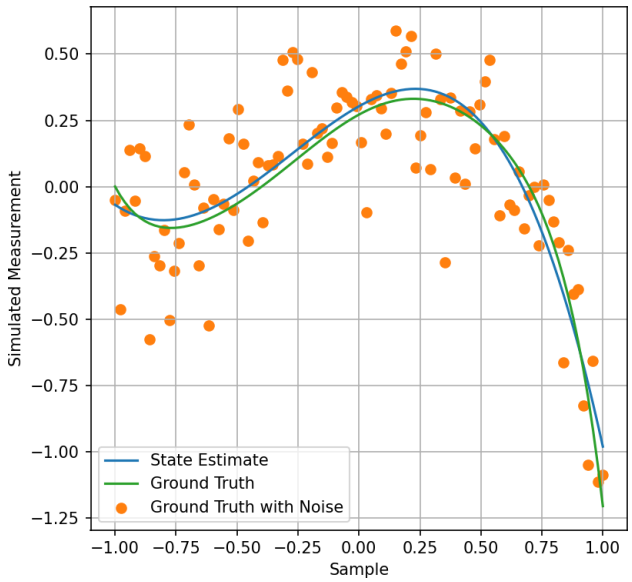


Figure 29: Sample plot of Hilbert Space Filter state estimation. Not only is the state estimate in this figure smoother than the AEKF estimate in Figure 24, but only five coefficients were learned to represent the entire estimate here. This is in contrast to learning a point-wise estimate for each sample in the case of the AEKF and AEIMMKF.

basis functions, $\{\boldsymbol{\varphi}_i\}_{i=1}^{P}$, we chose the Legendre Polynomials. Evaluating each $\{\boldsymbol{\varphi}_i\}_{i=1}^{P}$ on x defined above we get the final Hilbert Space Filter approximation of the ground truth

$$\hat{\boldsymbol{\phi}} = \sum_{i=0}^{P} \alpha_i \boldsymbol{\varphi}_i(x) \tag{156}$$

where the network is trained against the ground truth via (74).

4.7.2 Results

Although this work is in the preliminary stages, in Figure 29 we present an early result which shows the state estimate, $\hat{\boldsymbol{\phi}}$, is a smooth function that closely matches the actual ground truth in the presence of Gaussian noise. This is in contrast to the AEKF's point-wise estimation of the true state. Furthermore, only five coefficients, $\{\alpha_i\}_{i=1}^{5}$, were learned by the neural network for each function in $\boldsymbol{\phi}$. These

early results indicate that even with significant truncation of (77), accurate function representation can be achieved. However, when trained on Cauchy data, the Hilbert Space Filter does not perform well, as it is misled by periodic outlier data points. We claim this results from the fact that, unlike the Kalman Filter, there is no equivalent to $\{\mathbf{R}_k\}_{k=1}^N$, which scales the innovation inversely to the eigenvalues of $\{\mathbf{R}_k\}_{k=1}^N$, in the Hilbert Space Filter. One possible approach to this issue is a merger of the Hilbert Space Filter and Kalman Filter. In this scenario, the Kalman Filter's dynamical model would be replaced, or supplemented, by the Hilbert Space Filter, while the Kalman Filter itself would "weight" measurements based upon $\{\mathbf{R}_k\}_{k=1}^N$.

4.7.3 Summary

Although not fully developed, we believe working out the details of the Hilbert Space Filter and its potential to improve state estimation is an important next step in extending the research presented in this dissertation.

5 Future Work

5.1 Algorithms

5.1.1 Sequence Length for Autoencoder-Kalman Filter

Although the "blips" in Section 4.2 were never fully resolved, the fact the AEKF outperformed both the Kalman Filter and LSTM indicate AEKF sequence length is a worthwhile direction of future research.

5.1.2 Learning the Kalman Filter's Dynamical Model

In the AEKF, only parameters related to the measurement process, $\{\mathbf{z}_k\}_{k=1}^N$ and $\{\mathbf{R}_k\}_{k=1}^N$, were learned. The reason for restricting ourselves to only these two parameters was based upon the assumption the encoder and decoder portions of the AEKF would learn a mapping of the input measurements $\{\phi_k\}_{k=1}^N$ that was suitable for the Kalman Filter's given dynamical model. However, the success of the AEIMMKF demonstrates the use of multiple dynamical models improves state estimation. Thus, an extension of this research is to also learn the sequences $\{\mathbf{F}_k\}_{k=1}^N$, $\{\mathbf{Q}_k\}_{k=1}^N$, and $\{\mathbf{H}_k\}_{k=1}^N$ in the AEKF and AEIMMKF. In the case of the AEIMMKF, each of the M Kalman Filters would learn a unique $\{\mathbf{F}_k\}_{k=1}^N$ and $\{\mathbf{Q}_k\}_{k=1}^N$, and possibly $\{\mathbf{H}_k\}_{k=1}^N$.

5.1.3 Deep Interacting Multiple Model Kalman Filter

Building on the above, we identify three possible areas where deep learning can potentially be leveraged in the IMMKF, resulting in a deep IMMKF. These three potential avenues for further research are:

1. The four probabilities in (30), (31), (35), and (46) could be replaced by a composition of affine and sigmoid functions as in (51).

2. The linear combinations in (36), (37), (47), and (48) could similarly be replaced by a composition of affine transformations and non-linear activation functions.

3. The Gaussian PDF in (39) could be replaced by a function learned by a neural network, constrained to have the properties of a PDF.

5.1.4 Domain Randomization and Covering

Given the success of domain randomization with the AEKF, further exploration of this technique is an important direction of future research. First, our work on domain randomization sampled parameters uniformly in their parameter range. However, Active Domain Randomization [41] learns a parameter sampling strategy and shows improvement over simply sampling uniformly over a parameter range. To this end, research into alternatives to uniform sampling with domain randomization is an area of future investigation.

A second question is to ask how well either a random sampling or a fixed discretization of a parameter space is sufficient for generalization. For example, in our experiments herein we trained the AEKF for a fixed number of epochs, finding a balance between generalization error and runtime, where too few epochs leads to poor model generalization and too many leads to long training times. To perform a systematic analysis of the number of training epochs needed for a given problem, test set generalization would need to be compared against a range of training epoch sizes. Furthermore, the approach we used was to uniformly sample parameters within a predefined range. For example, if using domain randomization to

learn parameters $\{\alpha_k\}_{k=1}^N$

$$f(x) = \sum_{k=1}^{N} \alpha_k x^k \tag{157}$$

where each α_i is assumed to be in $[a, b]$, at each training epoch a value for each entry in the sequence $\{\alpha_k\}_{k=1}^N$ was uniformly sampled between $[a, b]$. When the number of epochs or equivalently random parameter samples is large enough, it is presumed the space of functions represented by (157) is well covered by the trained model. Ultimately, this is determined based upon test set generalization. Another approach would be to discretize the domain being trained over and each epoch randomly select one point from this discretization. That is, for each α_k in $\{\alpha_k\}_{k=1}^N$, partition $[a, b]$ into M equally spaced intervals, where each interval has length

$$\epsilon = \frac{b-a}{M} \tag{158}$$

resulting in $M + 1$ points defined at the interval boundaries and given by

$$p_k = \big(a, a+\epsilon, a+2\epsilon, \ldots, a+(M\text{-}1)\epsilon, b\big) \tag{159}$$

where each entry in p_k is a possible value of α_k for $k = \{1, 2, \ldots, N\}$. The Cartesian product formed by

$$p_1 \times p_2 \times \cdots \times p_N \tag{160}$$

then forms a discretization grid of the space $\mathbb{R}^N$, where each element randomly sampled from this space is a unique $\{\alpha_k\}_{k=1}^N \in \mathbb{R}^N$ used to fit the function in (157). In terms of generalization, the question then becomes how large must M be to achieve a given generalization error.

5.1.5 Comparing Function Families with Domain Randomization

Investigation into how well models trained on one class of functions (e.g. truncated Legendre Polynomials) generalize when evaluated on a different class of functions (e.g. truncated Taylor Polynomials) is a future direction of research.

5.1.6 Extending the Theoretical Understanding of the AEKF

Building on the theoretical and experimental results in Section 4.5, which demonstrate "what" the AEKF is learning, a next step would be to understand "how" the encoder is accomplishing this. Specifically, it is important to understand the inverse relationship between $\rho(\mathbf{S}_k^{-1})$ and $\|\tilde{\mathbf{z}}_k\|_2$. This addresses a more general question related to the theoretical foundations of deep learning. An initial approach would be to understand how an AEKF with linear transformations between layers learns this inverse relationship. If successful, it may be possible to extend the investigation to affine transformations and non-linear activation functions. Furthermore, given the above discussion on Hilbert Space representations of functions, it may prove useful to ask if neural networks can be modeled as linear operators in Hilbert Space. Given the well-established theory of linear operators on Hilbert Spaces, insight may be gained from this approach.

5.1.7 Hilbert Space Filter

Given the preliminary results in Section 4.7 further investigation into the state estimation capabilities of the Hilbert Space Filter is a promising direction of future research. As mentioned in Section 4.7, a combination of the Hilbert Space Filter and something analogous to the measurement noise covariance matrix in the Kalman Filter may allow the Hilbert Space Filter to perform well in the presence of outliers.

5.2 Applications

5.2.1 Outlier Detection

The results in 4.5 indicate the AEKF learns a sequence $\{\mathbf{R}_k\}_{k=1}^N$ such that the largest eigenvalues of each $\mathbf{R}_k$ has an inverse relationship to the ℓ_2 norm of the innovation (6) given by $\rho(\mathbf{S}_k^{-1}) \propto \|\tilde{\mathbf{z}}_k\|_2^\beta$ where $\beta < 0$. Given this, the AEKF is an excellent candidate to identify outliers in $\{\mathbf{z}_k\}_{k=1}^N$ by the eigenvalues of the corresponding covariance matrix $\{\mathbf{R}_k\}_{k=1}^N$. For a dataset with labeled outliers, after training the AEKF with domain randomization, the optimal threshold, based on the eigenvalues of $\{\mathbf{R}_k\}_{k=1}^N$, for considering a point an outlier can be learned from a training set and then evaluated on a testing set.

5.2.2 State Estimation and Association with Air Traffic Control Radar Data

Given the performance of the AEIMMKF on simulated flight data with constant turns, a next step would be to train an AEIMMKF in simulation and test on actual ATC flight data. While this experiment was considered in the above experiments, ultimately the difficulty of finding radar datasets with "ground truth" proved difficult. With real-world datasets, what constitutes "ground truth" is the fact one measurement is more reliable than another as there is no analytical ground truth as with domain randomization. For example, we considered the MANV dataset [56], a marine radar dataset from the German Aerospace Centre. For our purposes, what constituted the ground truth in the MANV dataset deviated significantly enough from radar measurements that using it for an estimate of the true state was problematic. Furthermore, the radar measurements for the multiple targets in the MANV dataset were not associated [9]. That is, the radar measurements for target 1 and target 2 were not labeled as such. This leads to another future extension

of the AEKF/AEIMMKF, where deep learning is leveraged to assist with solving the association problem. One possible source of real-world data is The Open Sky Network [1], which is a Swiss non-profit organization that provides open access, real-world air traffic data.

6 Conclusion

The primary algorithmic contribution of this dissertation is a novel deep learning-Kalman Filter hybrid algorithm, the Autoencoder-Kalman Filter (AEKF). Leveraging the well-known mathematical foundation of the Kalman Filter and the computational power of an autoencoder, we demonstrated that the AEKF outperforms a standard Kalman Filter and Long Short-Term Memory (LSTM) recurrent neural network on a variety of state estimation tasks. In the larger picture, the AEKF is envisioned as an integral part of a generalized state estimation system, which is robust to a variety of function and noise types. A major component of this generalized state estimation system was the use of domain randomization informed by the function representation properties of Hilbert Spaces. Paralleling the traditional use of domain randomization as modeling physical parameters, via domain randomization we trained the AEKF to perform state estimation on a variety of function families, which can be conceptualized as living in different subspaces of a Hilbert Space.

The most important design choice with the AEKF was to have the autoencoder portion of the AEKF learn a sequence of measurement noise covariance matrices $\{\mathbf{R}_k\}_{k=1}^{N}$. In addition to allowing the AEKF to achieve better state estimation than with a single fixed $\mathbf{R}$ for $k = \{1, 2, \ldots, N\}$, this design decision allowed for the derivation a theorem that provided an upper bound on the AEKF's scaling of outlier measurements in terms of matrix eigenvalues learned by the AEKF. Thus, in addition to providing insight into how the AEKF is learning to mitigate the influence of outliers, the results of this theorem provide a metric by which to measure whether the AEKF is performing properly or not. This, in turn, informs the design of the AEKF's neural network components.

In terms of applications, we first applied a standard Kalman Filter to the feature engineering of chemical sensor time series response data. Most importantly, in the context of early detection of chemical agents, we demonstrated that classification on datasets preprocessed by the Kalman Filter achieved better chemical discrimination than unfiltered datasets. Building on this success, the AEKF was applied to a variety of function families in noise regimes where the Kalman Filter was known to be suboptimal. First, three AEKF models were trained to filter exponential, sigmoidal, and sinusoidal curves with added Gaussian, bimodal, and Cauchy noise. Moving towards a higher level of generalization, we next trained a single AEKF to filter all three of these function families on the same noise types. Lastly, at the highest level of generalization, we trained a single AEKF to filter truncated Taylor Polynomials with the same added noise types. Our final application was an extension of the AEKF, the Autoencoder-Interacting Multiple Model Kalman Filter (AEIMMKF), applied to simulated target tracking problems. In all the above tests, the AEKF and AEIMMKF outperformed both a standard Kalman Filter and an LSTM in the majority of our experiments. In particular, the AEKF and AEIMMKF consistently demonstrated superior outlier mitigation capabilities in state estimation problems with Cauchy noise.

Future work related to algorithm development consists of allowing the AEKF to learn other Kalman Filter parameters, such as the first derivative of the transformed measurements $\hat{\mathbf{z}}_k$ and the Kalman Filter's dynamical model $\mathbf{F}_k$. Building on the design philosophy of the AEKF, utilizing the power of deep learning to learn parameters of the Interacting Multiple Model Kalman Filter is a viable direction of research. Other promising areas of research include exploring the limits of and formalizing the understanding of domain randomization, furthering the theoreti-

cal understanding of the Kalman Filter, and building upon the preliminary results of the Hilbert Space Filter in Section 4.7. In terms of applications, given its superior outlier mitigation performance, the AEKF is an excellent candidate for application to anomaly detection problems. Based upon the success of the AEIMMKF in simulated tracking problems, the application of the AEIMMKF to real-world air traffic control datasets holds significant promise.

A Appendices

A.1 Derivation of Kalman Filter as the BLMVE

This derivation follows that presented in [3] while filling in some of the mathematical details.

We begin by showing the general conditions of an affine minimum variance estimator with quadratic loss. This result, combined with the model contained in (2) and (3), is then shown to result in the Kalman Filter. Recall from section 2.1.1 that all we need to assume is $\mathbf{x}$ and $\mathbf{z}$ vary for the same underlying reasons, i.e. they are functions of events in the same sample space. Assuming there is some non-zero covariance between them, the goal of the Kalman Filter is to use the observations $\mathbf{z}$ to estimate the unknown state $\mathbf{x}$.

Given random vectors $\mathbf{x}$ and $\mathbf{z}$ with means $\bar{\mathbf{x}}$ and $\bar{\mathbf{z}}$ and covariances

$$\begin{aligned}
\mathrm{Cov}\big(\mathbf{x}, \mathbf{x}\big) &= \mathbf{\Sigma}^{xx} \\
\mathrm{Cov}\big(\mathbf{x}, \mathbf{z}\big) &= \mathbf{\Sigma}^{xz} \\
\mathrm{Cov}\big(\mathbf{z}, \mathbf{x}\big) &= \mathbf{\Sigma}^{zx} \\
\mathrm{Cov}\big(\mathbf{z}, \mathbf{z}\big) &= \mathbf{\Sigma}^{zz}
\end{aligned}$$

the best linear minimum variance estimator of $\mathbf{x}$ given $\mathbf{z}$ is

$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + (\mathbf{\Sigma}^{xz}(\mathbf{\Sigma}^{zz})^{-1})(\mathbf{z} - \bar{\mathbf{z}}) \tag{161}$$

Proof. The terms *best linear minimum variance estimator* tell us that the estimator, $\hat{\mathbf{x}}$, is going to be an linear (technically affine as mentioned in Section 2.1.2) function of the observations $\mathbf{z}$ and be quantified by minimizing the variance. More specifically,

given an estimator

$$\hat{\mathbf{x}} = \mathbf{A}\mathbf{z} + \mathbf{b} \tag{162}$$

where $\mathbf{A}$ is a matrix and $\mathbf{b}$ is a vector, the goal is then to solve

$$\arg\min_{\mathbf{A},\mathbf{b}} \mathbb{E}\big[\, \|\mathbf{x} - \mathbf{A}\mathbf{z} - \mathbf{b}\|^2 \,\big] \tag{163}$$

where all vector norms $\|\cdot\|$ in this section are the ℓ_2-norm. We begin by defining the term inside the norm in (163) as $S = \mathbf{x} - \mathbf{A}\mathbf{z} - \mathbf{b}$ and proceed to determine the mean and variance of S. The mean is

$$S_\mu = \bar{\mathbf{x}} - \mathbf{A}\bar{\mathbf{z}} - \mathbf{b} \tag{164}$$

Computing the variance we get

$$\begin{aligned} S_\sigma &= \mathbb{E}\big[(\mathbf{x} - \mathbf{A}\mathbf{z} - \mathbf{b} - \bar{\mathbf{x}} + \mathbf{A}\bar{\mathbf{z}} + \mathbf{b})(\mathbf{x} - \mathbf{A}\mathbf{z} - \mathbf{b} - \bar{\mathbf{x}} + \mathbf{A}\bar{\mathbf{z}} + \mathbf{b})^\mathsf{T}\big] && (165)\\ &= \mathbb{E}\big[\big((\mathbf{x} - \bar{\mathbf{x}}) - \mathbf{A}(\mathbf{z} - \bar{\mathbf{z}})\big)\big((\mathbf{x} - \bar{\mathbf{x}}) - \mathbf{A}(\mathbf{z} - \bar{\mathbf{z}})\big)^\mathsf{T}\big] && (166)\\ &= \boldsymbol{\Sigma}^{xx} - \boldsymbol{\Sigma}^{xz}\mathbf{A}^\mathsf{T} - \mathbf{A}\boldsymbol{\Sigma}^{zx} + \mathbf{A}\boldsymbol{\Sigma}^{zz}\mathbf{A}^\mathsf{T} && (167) \end{aligned}$$

where (167) results from expanding (166) and applying the linearity of the expectation operator. Next we show a simple result that follows from the linearity of the trace and expectation operators and the definition of covariance. Given vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, the covariance of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ is defined as

$$\mathrm{Cov}\big(\boldsymbol{\alpha}, \boldsymbol{\beta}\big) = \mathbb{E}\big[\boldsymbol{\alpha}\boldsymbol{\beta}^\mathsf{T}\big] - \mathbb{E}\big[\boldsymbol{\alpha}\big]\mathbb{E}\big[\boldsymbol{\beta}^\mathsf{T}\big] \tag{168}$$

Applying the trace operator to both sides of (168) and using the linearity of the

trance and expectation operators

$$tr(\mathrm{Cov}\big(\boldsymbol{\alpha},\boldsymbol{\beta}\big)) = tr(\mathbb{E}\big[\boldsymbol{\alpha}\boldsymbol{\beta}^{\mathsf{T}}\big]) - tr(\mathbb{E}\big[\boldsymbol{\alpha}\big]\mathbb{E}\big[\boldsymbol{\beta}^{\mathsf{T}}\big]) \tag{169}$$

$$= \mathbb{E}\big[|\langle\boldsymbol{\alpha},\boldsymbol{\beta}\rangle|\big] - tr(\mathbb{E}\big[\boldsymbol{\alpha}\big]\mathbb{E}\big[\boldsymbol{\beta}^{\mathsf{T}}\big]) \tag{170}$$

If $\boldsymbol{\alpha} = \boldsymbol{\beta}$, rearranging terms we get

$$\mathbb{E}\big[\,\|\boldsymbol{\alpha}\|^2\,\big] = tr(\mathrm{Var}\big[\boldsymbol{\alpha}\big]) + tr(\mathbb{E}\big[\boldsymbol{\alpha}\big]\mathbb{E}\big[\boldsymbol{\alpha}^{\mathsf{T}}\big]) \tag{171}$$

We are now in a position to determine $\mathbf{A}$ and $\mathbf{b}$. Applying (171) to the expectation in (163) and writing $S = \mathbf{x} - \mathbf{A}\mathbf{x} - \mathbf{b}$

$$\mathbb{E}\big[\,\|S\|^2\,\big] = tr(\mathrm{Var}\big[S\big]) + tr(\mathbb{E}\big[S\big]\mathbb{E}\big[S^{\mathsf{T}}\big]) \tag{172}$$

$$= tr(S_\sigma) + tr(S_\mu S_\mu^{\mathsf{T}}) \tag{173}$$

$$= tr(\boldsymbol{\Sigma}^{xx} - \boldsymbol{\Sigma}^{xz}\mathbf{A}^{\mathsf{T}} - \mathbf{A}\boldsymbol{\Sigma}^{zx} + \mathbf{A}\boldsymbol{\Sigma}^{zz}\mathbf{A}^{\mathsf{T}}) + \|\bar{\mathbf{x}} - \mathbf{A}\bar{\mathbf{z}} - \mathbf{b}\|^2 \tag{174}$$

At this point we apply a "trick" by adding and subtracting $\boldsymbol{\Sigma}^{xz}(\boldsymbol{\Sigma}^{zz})^{-1}\boldsymbol{\Sigma}^{zx}$ to the trace term in (174).

$$\mathbb{E}\big[\,\|S\|^2\,\big] = tr(\boldsymbol{\Sigma}^{xx} - \boldsymbol{\Sigma}^{xz}\mathbf{A}^{\mathsf{T}} - \mathbf{A}\boldsymbol{\Sigma}^{zx} + \mathbf{A}\boldsymbol{\Sigma}^{zz}\mathbf{A}^{\mathsf{T}} + \boldsymbol{\Sigma}^{xz}(\boldsymbol{\Sigma}^{zz})^{-1}\boldsymbol{\Sigma}^{zx} - \boldsymbol{\Sigma}^{xz}(\boldsymbol{\Sigma}^{zz})^{-1}\boldsymbol{\Sigma}^{zx}) +$$
$$\|\bar{\mathbf{x}} - \mathbf{A}\bar{\mathbf{z}} - \mathbf{b}\|^2 \tag{175}$$

$$= tr(\boldsymbol{\Sigma}^{xx} - \boldsymbol{\Sigma}^{xz}(\boldsymbol{\Sigma}^{zz})^{-1}\boldsymbol{\Sigma}^{zx}) +$$
$$tr(\mathbf{A}\boldsymbol{\Sigma}^{zz}\mathbf{A}^{\mathsf{T}} - \boldsymbol{\Sigma}^{xz}\mathbf{A}^{\mathsf{T}} - \mathbf{A}\boldsymbol{\Sigma}^{zx} + \boldsymbol{\Sigma}^{xz}(\boldsymbol{\Sigma}^{zz})^{-1}\boldsymbol{\Sigma}^{zx}) + \|\bar{\mathbf{x}} - \mathbf{A}\bar{\mathbf{z}} - \mathbf{b}\|^2 \tag{176}$$

$$= tr(\boldsymbol{\Sigma}^{xx} - \boldsymbol{\Sigma}^{xz}(\boldsymbol{\Sigma}^{zz})^{-1}\boldsymbol{\Sigma}^{zx}) +$$
$$tr((\mathbf{A} - \boldsymbol{\Sigma}^{xz}(\boldsymbol{\Sigma}^{zz})^{-1})\boldsymbol{\Sigma}^{zz}(\mathbf{A}^{\mathsf{T}} - (\boldsymbol{\Sigma}^{zz})^{-1}\boldsymbol{\Sigma}^{zx})) + \|\bar{\mathbf{x}} - \mathbf{A}\bar{\mathbf{z}} - \mathbf{b}\|^2 \tag{177}$$

where we assume $(\mathbf{\Sigma}^{zz})^{-1}$ exists. Additionally, we use the fact $\mathbf{\Sigma}^{xz} = (\mathbf{\Sigma}^{zx})^{\mathsf{T}}$. The reason for the above "trick" was to isolate terms involving $\mathbf{A}$ and $\mathbf{b}$. From this we see (177) is minimized when

$$\mathbf{A} = \mathbf{\Sigma}^{xz}(\mathbf{\Sigma}^{zz})^{-1} \tag{178}$$

$$\mathbf{b} = \bar{\mathbf{x}} - \mathbf{A}\bar{\mathbf{z}} \tag{179}$$

Thus S is given by

$$S = \mathbf{x} - (\mathbf{\Sigma}^{xz}(\mathbf{\Sigma}^{zz})^{-1})\mathbf{z} - \bar{\mathbf{x}} - \mathbf{A}\bar{\mathbf{z}} \tag{180}$$

$$= \mathbf{x} - \left[(\mathbf{\Sigma}^{xz}(\mathbf{\Sigma}^{zz})^{-1})(\mathbf{z} - \bar{\mathbf{z}}) + \bar{\mathbf{x}}\right] \tag{181}$$

and the linear minimum variance estimator of $\mathbf{x}$ is

$$\hat{\mathbf{x}} = \bar{\mathbf{x}} + (\mathbf{\Sigma}^{xz}(\mathbf{\Sigma}^{zz})^{-1})(\mathbf{z} - \bar{\mathbf{z}}) \tag{182}$$

□

Up to this point nothing specific to the Kalman Filter has been used. Now we consider the form (182) takes when the linear model in (2) and (3) is assumed. Additionally we assume $\mathbf{x}_0$ has mean and covariance $\bar{\mathbf{x}}_0$ and $\mathbf{P}_0$ respectively and $\{\mathbf{w}_k\}_{k=1}^N$ and $\{\mathbf{v}_k\}_{k=1}^N$ are zero-mean white processes with covariance $\{\mathbf{Q}_k\}_{k=1}^N$ and $\{\mathbf{R}_k\}_{k=1}^N$ respectively, for all $k \geq 0$. Further, we assume $\mathbf{x}_0$ is independent of both $\{\mathbf{w}_k\}_{k=1}^N$ and $\{\mathbf{v}_k\}_{k=1}^N$.

Next we compute the covariances in (182), where we introduce the superscript k to account for the index and the variables $\mathbf{x}_k$ are now the values taken by the

estimator $\mathbf{x}$ at time k.

$$\begin{aligned}
\boldsymbol{\Sigma}_k^{xx} &= \mathbb{E}\big[(\mathbf{x}_k - \bar{\mathbf{x}}_k)(\mathbf{x}_k - \bar{\mathbf{x}}_k)^{\mathsf{T}}\big] && (183)\\
&= \mathbb{E}\big[(\mathbf{F}_k\mathbf{x}_{k\text{-}1} + \mathbf{w}_k - \mathbf{F}_k\bar{\mathbf{x}}_{k\text{-}1})(\mathbf{F}_k\mathbf{x}_{k\text{-}1} + \mathbf{w}_k - \mathbf{F}_k\bar{\mathbf{x}}_{k\text{-}1})^{\mathsf{T}}\big] && (184)\\
&= \mathbb{E}\big[\big(\mathbf{F}_k(\mathbf{x}_{k\text{-}1} - \bar{\mathbf{x}}_{k\text{-}1}) + \mathbf{w}_k\big)\big(\mathbf{F}_k(\mathbf{x}_{k\text{-}1} - \bar{\mathbf{x}}_{k\text{-}1}) + \mathbf{w}_k\big)^{\mathsf{T}}\big] && (185)\\
&= \mathbb{E}\big[\mathbf{F}_k(\mathbf{x}_{k\text{-}1} - \bar{\mathbf{x}}_{k\text{-}1})(\mathbf{x}_{k\text{-}1} - \bar{\mathbf{x}}_{k\text{-}1})^{\mathsf{T}}\mathbf{F}_k^{\mathsf{T}}\big] + \mathbb{E}\big[\mathbf{F}_k(\mathbf{x}_{k\text{-}1} - \bar{\mathbf{x}}_{k\text{-}1})\mathbf{w}_k^{\mathsf{T}}\big] + \\
&\quad \mathbb{E}\big[\mathbf{w}_k(\mathbf{x}_{k\text{-}1} - \bar{\mathbf{x}}_{k\text{-}1})^{\mathsf{T}}\mathbf{F}_k^{\mathsf{T}}\big] + \mathbb{E}\big[\mathbf{w}_k\mathbf{w}_k^{\mathsf{T}}\big] && (186)\\
&= \mathbf{F}_k\mathbb{E}\big[(\mathbf{x}_{k\text{-}1} - \bar{\mathbf{x}}_{k\text{-}1})(\mathbf{x}_{k\text{-}1} - \bar{\mathbf{x}}_{k\text{-}1})^{\mathsf{T}}\big]\mathbf{F}_k^{\mathsf{T}} + \mathbf{Q}_k && (187)\\
&= \mathbf{F}_k\boldsymbol{\Sigma}_{k\text{-}1}^{xx}\mathbf{F}_k^{\mathsf{T}} + \mathbf{Q}_k && (188)
\end{aligned}$$

where the cross terms in (186) are zero due to the independence assumptions mentioned above. Applying the definition of covariance from (168) to the cross terms

$$\begin{aligned}
\mathbb{E}\big[\mathbf{F}_k(\mathbf{x}_{k\text{-}1} - \bar{\mathbf{x}}_{k\text{-}1})\mathbf{w}_k^{\mathsf{T}}\big] &= \mathbf{F}_k\mathbb{E}\big[\mathbf{x}_{k\text{-}1}\mathbf{w}_k^{\mathsf{T}}\big] - \mathbf{F}_k\mathbb{E}\big[\bar{\mathbf{x}}_{k\text{-}1}\mathbf{w}_k^{\mathsf{T}}\big] && (189)\\
&= \mathbf{F}_k\mathbb{E}\big[\mathbf{x}_{k\text{-}1}\mathbf{w}_k^{\mathsf{T}}\big] - \mathbf{F}_k\bar{\mathbf{x}}_{k\text{-}1}\mathbb{E}\big[\mathbf{w}_k^{\mathsf{T}}\big] && (190)\\
&= \mathbf{F}_k\mathbb{E}\big[\mathbf{x}_{k\text{-}1}\mathbf{w}_k^{\mathsf{T}}\big] && (191)
\end{aligned}$$

where all terms involving $\mathbb{E}\big[\mathbf{w}_k^{\mathsf{T}}\big]$ are zero. (191) is zero as follows. From (2) we have the following

$$\begin{aligned}
\mathbf{x}_k &= \mathbf{F}_k\mathbf{x}_{k\text{-}1} + \mathbf{w}_k && (192)\\
\mathbf{x}_{k\text{-}1} &= \mathbf{F}_{k\text{-}1}\mathbf{x}_{k\text{-}2} + \mathbf{w}_{k\text{-}1} && (193)
\end{aligned}$$

Substituting (193) into (192)

$$\mathbf{x}_k = \mathbf{F}_k(\mathbf{F}_{k\text{-}1}\mathbf{x}_{k\text{-}2} + \mathbf{w}_{k\text{-}1}) + \mathbf{w}_k \tag{194}$$

$$= \mathbf{F}_k\mathbf{F}_{k\text{-}1}\mathbf{x}_{k\text{-}2} + \mathbf{F}_k\mathbf{w}_{k\text{-}1} + \mathbf{w}_k \tag{195}$$

If we define

$$\mathbf{\Gamma}_{k,n} = \mathbf{F}_k\mathbf{F}_{k-1}\cdots\mathbf{F}_{n+1} \tag{196}$$

where $n \leq k$ and $\mathbf{\Gamma}_{l,l} = \mathbf{I}$ we can express (195) as

$$\mathbf{x}_k = \mathbf{\Gamma}_{k,k\text{-}2}\mathbf{x}_{k\text{-}2} + \mathbf{\Gamma}_{k,k\text{-}1}\mathbf{w}_{k\text{-}1} + \mathbf{\Gamma}_{k,k}\mathbf{w}_k \tag{197}$$

$$= \mathbf{\Gamma}_{k,k\text{-}2}\mathbf{x}_{k\text{-}2} + \sum_{l=k\text{-}1}^{k} \mathbf{\Gamma}_{k,l}\mathbf{w}_l \tag{198}$$

Continuing the recursion to $\mathbf{x}_0$ corresponds to $n = 0$ in (196) and we get

$$\mathbf{x}_k = \mathbf{\Gamma}_{k,0}\mathbf{x}_0 + \sum_{l=1}^{k} \mathbf{\Gamma}_{k,l}\mathbf{w}_l \tag{199}$$

Writing $\mathbf{x}_{k\text{-}1}$ in terms of (199) and substituting this into (191)

$$\mathbb{E}\big[\mathbf{F}_k(\mathbf{x}_{k\text{-}1} - \bar{\mathbf{x}}_{k\text{-}1})\mathbf{w}_k^{\mathsf{T}}\big] = \mathbf{F}_k\mathbb{E}\big[\mathbf{x}_{k\text{-}1}\mathbf{w}_k^{\mathsf{T}}\big] \tag{200}$$

$$= \mathbf{F}_k\mathbb{E}\big[\big(\mathbf{\Gamma}_{k\text{-}1,0}\mathbf{x}_0 + \sum_{l=1}^{k\text{-}1} \mathbf{\Gamma}_{k\text{-}1,l}\mathbf{w}_l\big)\mathbf{w}_k^{\mathsf{T}}\big] \tag{201}$$

$$= \mathbf{F}_k\mathbf{\Gamma}_{k\text{-}1,0}\mathbb{E}\big[\mathbf{x}_0\mathbf{w}_k^{\mathsf{T}}\big] + \mathbf{F}_k\sum_{l=1}^{k\text{-}1} \mathbf{\Gamma}_{k\text{-}1,l}\mathbb{E}\big[\mathbf{w}_l\mathbf{w}_k^{\mathsf{T}}\big] \tag{202}$$

Since $\mathbf{x}_0$ and $\{\mathbf{w}_k\}_{k=1}^N$ are assumed independent, from the definition of covariance and the zero-mean assumption on $\{\mathbf{w}_k\}_{k=1}^N$, the first term in (202) is zero.

Additionally, since we also assume $\{\mathbf{w}_k\}_{k=1}^N$ is a zero-mean white process and $\mathrm{Cov}\big(\mathbf{w}_l, \mathbf{w}_k\big) = 0$ for $l \neq k$, using the definition of covariance and noting the summation index is $l < k$, each expectation in the right hand sum is zero. Furthermore, in (186) we get $\mathrm{Var}\big[\mathbf{w}_k\big] = \mathbb{E}\big[\mathbf{w}_k\mathbf{w}_k^{\mathsf{T}}\big] = \mathbf{Q}_k$ from the definition of covariance and the fact $\{\mathbf{w}_k\}_{k=1}^N$ is zero-mean and white. From this we see that

$$\boldsymbol{\Sigma}_k^{xx} = \mathbf{F}_k\boldsymbol{\Sigma}_{k\text{-}1}^{xx}\mathbf{F}_k^{\mathsf{T}} + \mathbf{Q}_k \tag{203}$$

which gives us an iterative technique which allows for the recursive calculation of $\boldsymbol{\Sigma}_k^{xx}$ for $k > 0$ given $\boldsymbol{\Sigma}_0^{xx}$. Applying the same procedure to $\boldsymbol{\Sigma}_k^{zz}$

$$\begin{aligned}
\boldsymbol{\Sigma}_k^{zz} &= \mathbb{E}\big[(\mathbf{z}_k - \bar{\mathbf{z}}_k)(\mathbf{z}_k - \bar{\mathbf{z}}_k)^{\mathsf{T}}\big] && (204)\\
&= \mathbb{E}\big[(\mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k - \mathbf{H}_k\bar{\mathbf{x}}_k)(\mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k - \mathbf{H}_k\bar{\mathbf{x}}_k)^{\mathsf{T}}\big] && (205)\\
&= \mathbb{E}\big[\big(\mathbf{H}_k(\mathbf{x}_k - \bar{\mathbf{x}}_k) + \mathbf{v}_k\big)\big(\mathbf{H}_k(\mathbf{x}_k - \bar{\mathbf{x}}_k) + \mathbf{v}_k\big)^{\mathsf{T}}\big] && (206)\\
&= \mathbb{E}\big[\mathbf{H}_k(\mathbf{x}_k - \bar{\mathbf{x}}_k)(\mathbf{x}_k - \bar{\mathbf{x}}_k)^{\mathsf{T}}\mathbf{H}_k^{\mathsf{T}}\big] + \mathbb{E}\big[\mathbf{H}_k(\mathbf{x}_k - \bar{\mathbf{x}}_k)\mathbf{v}_k^{\mathsf{T}}\big] + \\
&\quad \mathbb{E}\big[\mathbf{v}_k(\mathbf{x}_k - \bar{\mathbf{x}}_k)^{\mathsf{T}}\mathbf{H}_k^{\mathsf{T}}\big] + \mathbb{E}\big[\mathbf{v}_k\mathbf{v}_k^{\mathsf{T}}\big] && (207)\\
&= \mathbf{H}_k\mathbb{E}\big[(\mathbf{x}_k - \bar{\mathbf{x}}_k)(\mathbf{x}_k - \bar{\mathbf{x}}_k)^{\mathsf{T}}\big]\mathbf{H}_k^{\mathsf{T}} + \mathbf{R}_k && (208)\\
&= \mathbf{H}_k\boldsymbol{\Sigma}_k^{xx}\mathbf{H}_k^{\mathsf{T}} + \mathbf{R}_k && (209)
\end{aligned}$$

where the cross terms are derived in the same manner as in the case of $\boldsymbol{\Sigma}_k^{xx}$, with $\mathbf{H}_k$ replacing $\mathbf{F}_k$ and $\mathbf{v}_k$ replacing $\mathbf{w}_k$. Since $\{\mathbf{v}_k\}_{k=1}^N$ shares the same independence and whiteness assumptions as $\{\mathbf{w}_k\}_{k=1}^N$ the result follows. Similarly, $\mathrm{Var}\big[\mathbf{v}_k\big] = \mathbb{E}\big[\mathbf{v}_k\mathbf{v}_k^{\mathsf{T}}\big] = \mathbf{R}_k$ Thus,

$$\boldsymbol{\Sigma}_k^{zz} = \mathbf{H}_k\boldsymbol{\Sigma}_k^{xx}\mathbf{H}_k^{\mathsf{T}} + \mathbf{R}_k \tag{210}$$

Lastly we determine $\mathbf{\Sigma}_k^{xz}$

$$\mathbf{\Sigma}_k^{xz} = \mathbb{E}\big[(\mathbf{x}_k - \bar{\mathbf{x}}_k)(\mathbf{z}_k - \bar{\mathbf{z}}_k)^{\mathsf{T}}\big] \tag{211}$$

$$= \mathbb{E}\big[(\mathbf{x}_k - \bar{\mathbf{x}}_k)(\mathbf{H}_k\mathbf{x}_k - \mathbf{H}_k\bar{\mathbf{x}}_k)^{\mathsf{T}}\big] \tag{212}$$

$$= \mathbb{E}\big[(\mathbf{x}_k - \bar{\mathbf{x}}_k)(\mathbf{x}_k - \bar{\mathbf{x}}_k)^{\mathsf{T}}\big]\mathbf{H}_k^{\mathsf{T}} \tag{213}$$

recognizing the expectation in (213) as $\mathbf{\Sigma}_k^{xx}$ gives

$$\mathbf{\Sigma}_k^{xz} = \mathbf{\Sigma}_k^{xx}\mathbf{H}_k^{\mathsf{T}} \tag{214}$$

Writing (161) with an added subscript k

$$\hat{\mathbf{x}}_k = \bar{\mathbf{x}}_k + (\mathbf{\Sigma}_k^{xz}(\mathbf{\Sigma}_k^{zz})^{-1})(\mathbf{z}_k - \bar{\mathbf{z}}_k) \tag{215}$$

we can get expressions for $\bar{\mathbf{x}}_k$ and $\bar{\mathbf{z}}_k$ from (2) and (3)

$$\bar{\mathbf{x}}_k = \mathbf{F}_k\bar{\mathbf{x}}_{k\text{-}1} \tag{216}$$

$$\bar{\mathbf{z}}_k = \mathbf{H}_k\bar{\mathbf{x}}_k \tag{217}$$

from (203), (210), and (214) we have expressions for the covariance matrices. If we identify (203) with the *a priori* estimate covariance $\mathbf{P}_{k|k\text{-}1}$, (210) with the innovation covariance $\mathbf{S}_k$, and substitute (216) and (217) we can write (215) as

$$\hat{\mathbf{x}}_k = \mathbf{F}_k\bar{\mathbf{x}}_{k\text{-}1} + (\mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^{\mathsf{T}}\mathbf{S}_k^{-1})(\mathbf{z}_k - \mathbf{H}_k\bar{\mathbf{x}}_k) \tag{218}$$

Lastly, if we identity $\bar{\mathbf{x}}_k$ and $\bar{\mathbf{x}}_{k\text{-}1}$ in (216) with the *a priori* estimate at time k and the

a posteriori estimate at time k-1 respectively we can rewrite (218) as

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k\text{-}1} + (\mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^\mathsf{T}\mathbf{S}_k^{-1})(\mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k\text{-}1}) \tag{219}$$

which we see is the Kalman Filter's *a posteriori* state estimate in (9). Given an initial state estimate $\mathbf{x}_0$ and initial covariance $\mathbf{P}_0$, where we define $\hat{\mathbf{x}}_{1|0} = \mathbf{F}_1\mathbf{x}_0$ and $\mathbf{P}_{1|0} = \mathbf{F}_1\mathbf{P}_0\mathbf{F}_1^\mathsf{T} + \mathbf{Q}_1$, substituting these into (219) and iterating for $k > 0$ we get the Kalman Filter's recursive *a posteriori* state estimation.

A similar process for calculating the covariances above is used to determine the *a posteriori* covariance.

$$\begin{aligned}
\mathbf{P}_{k|k} &= \mathbb{E}\big[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})^\mathsf{T}\big] && (220)\\
&= \mathbb{E}\big[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k\text{-}1} - \mathbf{K}_k\tilde{\mathbf{z}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k\text{-}1} - \mathbf{K}_k\tilde{\mathbf{z}}_k)^\mathsf{T}\big] && (221)\\
&= \mathbb{E}\big[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k\text{-}1})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k\text{-}1})^\mathsf{T}\big] - \mathbb{E}\big[\mathbf{K}_k\tilde{\mathbf{z}}_k(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k\text{-}1})^\mathsf{T}\big] - \\
&\quad \mathbb{E}\big[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k\text{-}1})\tilde{\mathbf{z}}_k^\mathsf{T}\mathbf{K}_k^\mathsf{T}\big] + \mathbb{E}\big[\mathbf{K}_k\tilde{\mathbf{z}}_k\tilde{\mathbf{z}}_k^\mathsf{T}\mathbf{K}_k^\mathsf{T}\big] && (222)\\
&= \mathbf{P}_{k|k\text{-}1} - \mathbf{K}_k\mathbf{H}_k\mathbf{P}_{k|k\text{-}1}^\mathsf{T} - \mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^\mathsf{T}\mathbf{K}_k^\mathsf{T} + \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^\mathsf{T} && (223)\\
&= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k\text{-}1}^\mathsf{T} - \mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^\mathsf{T}\mathbf{K}_k + \mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^\mathsf{T}\mathbf{S}_k^{-1}\mathbf{S}_k\mathbf{K}_k^\mathsf{T} && (224)\\
&= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k\text{-}1}^\mathsf{T} - \mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^\mathsf{T}\mathbf{K}_k + \mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^\mathsf{T}\mathbf{K}_k^\mathsf{T} && (225)\\
&= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k\text{-}1} && (226)
\end{aligned}$$

where we used the fact $\mathbf{P}_{k|k\text{-}1} = \mathbf{P}_{k|k\text{-}1}^\mathsf{T}$ and $\mathbf{K}_k = \mathbf{P}_{k|k\text{-}1}\mathbf{H}_k^\mathsf{T}\mathbf{S}_k^{-1}$

References

[1] The Open Sky Network. `https://opensky-network.org/`, 2020.

[2] ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., CORRADO, G. S., DAVIS, A., DEAN, J., DEVIN, M., GHEMAWAT, S., GOODFELLOW, I., HARP, A., IRVING, G., ISARD, M., JIA, Y., JOZEFOWICZ, R., KAISER, L., KUDLUR, M., LEVENBERG, J., MANÉ, D., MONGA, R., MOORE, S., MURRAY, D., OLAH, C., SCHUSTER, M., SHLENS, J., STEINER, B., SUTSKEVER, I., TALWAR, K., TUCKER, P., VANHOUCKE, V., VASUDEVAN, V., VIÉGAS, F., VINYALS, O., WARDEN, P., WATTENBERG, M., WICKE, M., YU, Y., AND ZHENG, X. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] ANDERSON, B. D., AND MOORE, J. B. *Optimal Filtering*. Dover Publications, 1979.

[4] BAR-SHALOM, Y., RONG LI, X., AND KIRUBARAJAN, T. *Estimation with Appications to Tracking and Navigation*. Wiley, New York, 2001.

[5] BLAIR, W. D., AND WATSON, G. A. IMM algorithm for solution to benchmark problem for tracking maneuvering targets. *Acquisition, Tracking, and Pointing VIII 2221*, July 1994 (1994), 476–488.

[6] BOSER, B. E., GUYON, I. M., AND VAPNIK, V. N. A Training Algorithm for Optimal Margin Classifiers. *Proceedings of the fifth annual workshop on Computational learning theory* (1992), 144–152.

[7] CATLIN, D. E. *Estimation, Control, and the Discrete Kalman Filter*. Springer-Verlag, New York, 1989.

[8] C.CORTES, AND V.VAPNIK. Support Vector Networks. *Machine Learning 20*, 3 (1995), 273–297.

[9] CHAW-BING CHANG, K.-P. D. *Applied State Estimation and Association*, 1st ed. MIT Press, Cambridge, 2016.

[10] COSKUN, H., ACHILLES, F., DIPIETRO, R., NAVAB, N., AND TOMBARI, F. Long Short-Term Memory Kalman Filters: Recurrent Neural Estimators for Pose Regularization. *Proceedings of the IEEE International Conference on Computer Vision 2017-Octob* (2017), 5525–5533.

[11] COURVILLE, I. G., BENGIO, Y., AND AARON. *Deep Learning*. MIT Press, 2016.

[12] COVER, T., AND HART, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory 13*, 1 (1967), 21–27.

[13] FISHER, R. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics 7*, 2 (1936), 179–188.

[14] FIX, E., AND HODGES, J. L. Discriminatory Analysis - Nonparametric Discrimination: Consistency Properties. *International Statistical Review 57*, 3 (2016), 238–247.

[15] FRACCARO, M., KAMRONN, S., PAQUET, U., AND WINTHER, O. A disentangled recognition and nonlinear dynamics model for unsupervised learning. *Advances in Neural Information Processing Systems 2017-Decem*, section 5 (2017), 3602–3611.

[16] GERS, F. A., AND SCHMIDHUBER, J. LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks 12*, 6 (2001), 1333–1340.

[17] GERS, F. A., SCHRAUDOLPH, N. N., AND SCHMIDHUBER, J. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research 3*, 1 (2003), 115–143.

[18] GRAVES, A. *Supervised Sequence Labelling with Recurrent Neural Networks*, frist ed. Springer-Verlag, Berlin, Heidelberg, 2012.

[19] HAARNOJA, T., AJAY, A., LEVINE, S., AND ABBEEL, P. Backprop KF: Learning Discriminative Deterministic State Estimators. *arXiv:1605.07148 [cs.LG]*, Nips (2017).

[20] HASTIE, T., TIBSHIRANT, R., FRIEDMAN, J., TIBSHIRANI, R., AND FRIEDMAN, J. *The Elements of Statistical Learning*, second ed. Springer, New York, 2016.

[21] HELMBERG, G. *Introduction to Spectral Theory in Hilbert Space*. Dover Books on Mathematics. Dover Publications, 1997.

[22] HIERLEMANN, A., AND GUTIERREZ-OSUNA, R. Higher-Order Chemical Sensing. *Chemical Reviews 108*, 2 (2008), 563–613.

[23] HOCHREITER, S., AND URGEN SCHMIDHUBER, J. Long Short-Term Memory. *Neural Computation 9*, 8 (1997), 1735–1780.

[24] HORN, R. A., AND JOHNSON, C. R. *Matrix Analysis*. Matrix Analysis. Cambridge University Press, 2013.

[25] HOSSEINYALAMDARY, S. Deep Kalman filter: Simultaneous multi-sensor integration and modelling; A GNSS/IMU case study. *Sensors (Switzerland) 18*, 5 (2018).

[26] HUNTER, J. D. Matplotlib: A 2D graphics environment. *Computing in Science and Engineering 9*, 3 (2007), 99–104.

[27] JOHN NETER, D., KUTNER, M. H., AND NACHTSHEIM, C. J. *MP Applied Linear Regression Models-Revised Edition with Student CD*. McGraw-Hill Education, 2004.

[28] JULIER, S. J., AND UHLMANN, J. K. New extension of the Kalman filter to nonlinear systems. *Signal Processing, Sensor Fusion, and Target Recognition VI 3068*, July 1997 (1997), 182.

[29] KALMAN, R. E. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME Journal of Basic Engineering 82*, Series D (1960), 35–45.

[30] KARIM, F., MAJUMDAR, S., DARABI, H., AND CHEN, S. LSTM Fully Convolutional Networks for Time Series Classification. *IEEE Access 6* (2017), 1662–1669.

[31] KENNEDY, R. A., AND SADEGHI, P. *Hilbert Space Methods in Signal Processing*. Hilbert Space Methods in Signal Processing. Cambridge University Press, 2013.

[32] KINGMA, D. P., AND BA, J. L. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* (2015), 1–15.

[33] KINGMA, D. P., AND WELLING, M. Auto-Encoding Variational Bayes. *arXiv:1312.6114v10 [stat.ML]*, Ml (2013), 1–14.

[34] KRISHNAN, R. G., SHALIT, U., AND SONTAG, D. Deep Kalman Filters. *arXiv:1511.05121 [stat.ML]* (2015).

[35] LEBEDEV, N. *Special Functions and Their Applications*. Dover Publications, 2018.

[36] LU, D. W. Agent Inspired Trading Using Recurrent Reinforcement Learning and LSTM Neural Networks. *arXiv:1707.07338 [q-fin.CP]* (2017).

[37] MAAS, A. L., HANNUN, A. Y., AND NG, A. Y. Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing* (2013).

[38] MATISKO, P., AND HAVLENA, V. Optimality tests and adaptive Kalman filter. *IFAC Proceedings Volumes (IFAC-PapersOnline) 16*, PART 1 (2012), 1523–1528.

[39] MAZOR, E., AVERBUCH, A., BAR-SHALOM, Y., AND DAYAN, J. Interacting Multiple model Methods in Target Tracking: A Survey. *IEEE Transactions On Aerospace And Electronic Systems 34*, 1 (1998).

[40] MEHRA, R. K. On the Identification of Variances and Adaptive Kalman Filtering. *IEEE Transactions on Automatic Control AC-15*, 2 (1970), 175–184.

[41] MEHTA, B., DIAZ, M., GOLEMO, F., PAL, C. J., AND PAULL, L. Active Domain Randomization. *arXiv:1904.04762 [cs.LG]* (2019).

[42] NALLON, E. C., SCHNEE, V. P., BRIGHT, C., POLCHA, M. P., AND LI, Q. Chemical Discrimination with an Unmodified Graphene Chemical Sensor. *ACS Sensors 1*, 1 (2016), 26–31.

[43] NALLON, E. C., SCHNEE, V. P., BRIGHT, C. J., POLCHA, M. P., AND LI, Q. Discrimination Enhancement with Transient Feature Analysis of a Graphene Chemical Sensor. *Analytical Chemistry 88*, 2 (2016), 1401–1406.

[44] ODELSON, B. J., RAJAMANI, M. R., AND RAWLINGS, J. B. A new autocovariance least-squares method for estimating noise covariances. *Automatica 42*, 2 (2006), 303–308.

[45] PARLOS, A. G., MENON, S. K., AND ATIYA, A. F. An algorithmic approach to adaptive state filtering using recurrent neural networks. *IEEE TRANSACTIONS ON NEURAL NETWORKS 12*, 6 (2001), 1411–1432.

[46] PEARSON, J., AND STEAR, E. Kalman Filter Applications in Airborne Radar Tracking. *IEEE Transactions on Aerospace and Electronic Systems AES-10*, 3 (1974), 319–329.

[47] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND DUCHESNAY, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[48] PLETT, G. L. ECE5550: Applied Kalman Filtering - The Interacting-Multiple-Model Kalman Filter. `http://mocha-java.uccs.edu/ECE5550/`, 2018.

[49] R HAHNLOSER, R. SARPESHKAR, M A MAHOWALD, R. J. DOUGLAS, H. S. S. Digital Selection and Analogue Amplification Coexist in a Cortex-Inspired Silicon Circuit. *Nature 405* (2000), 947–951.

[50] RAITOHARJU, M., PICHÉ, R., AND NURMINEN, H. A Systematic Approach for Kalman-type Filtering with non-Gaussian Noises. *2016 19th International Conference on Information Fusion (FUSION)* (2016), 1853 – 1858.

[51] RAJAMANI, M. R., AND RAWLINGS, J. B. Estimation of the disturbance structure from data using semidefinite programming and optimal weighting. *Automatica 45*, 1 (2009), 142–148.

[52] REN, X., LUO, J., SOLOWJOW, E., OJEA, J. A., GUPTA, A., TAMAR, A., AND ABBEEL, P. Domain randomization for active pose estimation. *arXiv:1903.03953v1 [cs.CV]* (2019).

[53] RUMELHART, D. E., HINTON, G. E., AND WILLIAMS, R. J. Learning Representations By Back-Propagating Errors. *Nature 323* (1986), 533–536.

[54] RUYMGAART, P. A., SOONG, T. T., AND T., T. S. *Mathematics of Kalman-Bucy Filtering*, 1st ed., vol. 136. Springer-Verlag, Berlin Heidelberg New York Tokyo, 1985.

[55] SHI, X., CHEN, Z., WANG, H., YEUNG, D. Y., WONG, W. K., AND WOO, W. C. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *arXiv:1506.04214v2 [cs.CV]* (2015).

[56] SHRADHA, J., BAUM, M., AND HEYMANN, F. A Marine Radar Dataset for Multiple Extended Target Tracking. *1st Maritime Situational Awareness Workshop* (2019), 7.

[57] SIMON, D. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*, 1st ed. Wiley, Hoboken, 2006.

[58] TOBIN, J., FONG, R., RAY, A., SCHNEIDER, J., ZAREMBA, W., AND ABBEEL, P. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. *arXiv:1703.06907 [cs.RO]* (2017).

[59] TREFETHEN, L. N., AND BAU, D. *Numerical Linear Algebra*, 1 ed. Society for Industrial and Applied Mathematics, 1997.

[60] VAN DER WALT, S., COLBERT, S. C., AND VAROQUAUX, G. The NumPy array: A structure for efficient numerical computation. *Computing in Science and Engineering 13*, 2 (2011), 22–30.

[61] WELCH, G., AND BISHOP, G. An Introduction to the Kalman Filter. *In Practice 7*, 1 (2006), 1–16.

[62] WIEDERODER, M. S., NALLON, E. C., WEISS, M., MCGRAW, S. K., SCHNEE, V. P., BRIGHT, C. J., POLCHA, M. P., PAFFENROTH, R., AND UZARSKI, J. R. Graphene Nanoplatelet-Polymer Chemiresistive Sensor Arrays for the Detection and Discrimination of Chemical Warfare Agent Simulants. *ACS Sensors 2*, 11 (2017), 1669–1678.

[63] YADAIAH, N., AND SOWMYA, G. Neural Network Based State Estimation of Dynamical Systems. *The 2006 IEEE International Joint Conference on Neural Network Proceedings* (2006), 1042–1049.

[64] YOUNG, T., HAZARIKA, D., PORIA, S., AND CAMBRIA, E. Recent Trends in Deep Learning Based Natural Language Processing. *arXiv:1708.02709 [cs.CL]* (2017).

[65] ZARCHAN, P., AND MUSOFF, H. *Fundamentals of Kalman Filtering: A Practical Approach*, 4th ed. American Institute of Aeronautics and Astronautics, Inc., 2015.

[66] ZHANG, Q., WANG, H., DONG, J., ZHONG, G., AND SUN, X. Prediction of Sea Surface Temperature using Long Short-Term Memory. *IEEE Geoscience and Remote Sensing Letters 14*, 10 (2017), 1745–1749.