Neural Network Aided Kalman Filter to Maximize Accuracy

Varun Niraj Agarwal
School of Computer Science Engineering
Vellore Institute of Technology
Vellore, India
varun.agarwal2019@vitstudent.ac.in

Avaneesh Kanshi
School of Computer Science Engineering
Vellore Institute of Technology
Vellore, India
avaneesh.kanshi2019@vitstudent.ac.in

Navneet Vinod Melarkode
School of Computer Science Engineering
Vellore Institute of Technology
Vellore, India
navneetvinod.melarkode2019@vitstudent.ac.in

Hemanth Krishna
School of Computer Science Engineering
Vellore Institute of Technology
Vellore, India
hemanth.krishna2019@vitstudent.ac.in

Dr. Malaya Kumar Hota
Department of Communication Engineering
School of Electronics and Engineering
Vellore Institute of Technology
Vellore, India
malayakumar.h@vit.ac.in
ORCID: 0000-0002-3669-1611

***Abstract*—With the growing need for data, and ever-growing demand for prediction and error-correction, Kalman Filters are undoubtedly at the forefronts of real-time estimation. While these filters are designed to achieve convergence shortly after getting exposed to the data, the filters might not be able to maximize all the data it can extract from the system.**

In order to extract most of the information that is otherwise unusable by the Kalman Factor, an initial assumption of a pre-trained Machine Learning model that correlates a feedable parameter with the unusable data is made. The feedable parameter is then given to the Kalman Filter along with the other standard parameters which boosts the accuracy by adding another dimension to the filter.

I. Introduction

IN modern day calculations, Kalman Filter (KF) play an integral role. They are extensively used while performing state estimations and linear optimal filtering of non-stationary stochastic processes. As much as the algorithm is the chief estimator, the states and parameters are as necessary as it gets. At present, the KF can be initialized in various ways including but not limited to Probabilistic models such as Gaussian and Bayesian approaches, or even geometric approaches. For cases when enough data isn't available, the KF assumes an initial state of mean and variance. In actuality, the real-world data cannot be mapped to either probabilistic or geometric models [1]. The main pitfall concerning the same lies in the filter's convergence ability as and when required. General trends suggest that the fewer the parameters fed into the filter, the higher the convergence rate would be. The problem is negated when the KF can find and extract all permissible parameters from the surrounding system.

However, if we need the filter to show signs of parity from get-go, it is advisable that the chaotic states first be converted to predictable variables and also be correlated to the initial guesses [2]. Therefore, it is highly important that the KF's superior estimation algorithm be maximized by employing the use of converted chaotic variables, which generally is neglected and isn't made use of as the required components are not salvageable for the KF to make use of. Salvaging these states from the chaotic variables will also aid and assist the estimation process positively and help converge faster than usual.

This paper proposes an alternative method to maximize the abilities of KF by converting a chaotic variable which had no significant use for the estimation into a predictable and feedable variable that adds another state. A simple Neural Network (NN) model with dense layers is incorporated into the KF to convert the order less variable into a semantic variable [3]. The limitations of a standard KF and its state estimation algorithm can successfully be overhauled with the help of a NN aided system that adds an extra dimension of accuracy by outperforming classical filtering methods.

II. Related Work

Several earlier methods have been used to initialize the Kalman Filter without the assumptions. Some of them include

A. Initialization of the Kalman Filter Without Assumptions on the Initial State

The authors in [4] proposed a method to initialize the filter without making any prior assumptions of the initial values. However, the method was limited to a linear space-state form system. The algorithm was corroborated and verified with the help of the problem statement setup. Consider a ball being thrown towards a robot. A box is moved upon the robot's instruction in a way the ball hits the hole in the box. In this scenario, it is not viable for the KF to guess the initial estimates due to the nature of the experiment, that is, a very short burst of estimates where none of them should deviate due to the initial guesses. The filter provides us with measurement which is regularly modified so as to attain a barebone filter pertaining to information. However, updating time is not simple and doesn't work well in linear space.

B. KalmanNet: Neural Network-Aided Adaptive Unscented Kalman Filter for Nonlinear State Estimation

The KF inherits disadvantages such as instability due to linearity and inefficient calculation of the Jacobian matrices causing the performance to degrade in filtering non-linear datasets. Reference [3] introduces a Machine Learning (ML) algorithm which is further used to develop a NN model incorporated into an unscented KF to improve convergence in

uncertain systems. This approach resulted in a huge improvement in computation time while being able to remove uncertainties of a non-linear dataset and filter it out in a similar manner as a linear system.

C. Neural Kalman Filtering

The traditional and conventional approach to KF requires exceedingly complicated mathematical equations that are highly unlikely to be implemented into a NN circuit. Reference [2] demonstrates how a simple gradient descent algorithm can be employed for the same tasks the conventional approach aims to solve. Furthermore, the authors compare and contrast both the models by using a simple KF, that justifies the use of Neural architecture. The paper also suggests a Neural implementation of the required complex Kalman equations to support their argument. At the same time, there are several deficiencies with the aforementioned method as the model assumes a full connectivity which are required to implement the matrix methods. To this, the authors suggest that the model be used at the bottom of the pile, thereby eliminating a key drawback, the assumption of linearity.

III. Proposed Methodology

A. Initialization

We define and initialize the state variables, model equation and the different variation matrices

$$x_k = Fx_{k-1} + Bu_{k-1} + w_{k-1} \tag{1}$$

x_k is the state of the system
F is the prediction matrix
H is the control matrix
u_k is the control vector
w_k is the process noise vector (mean: 0, covariance: Q)

B. Prediction

Model equations are used to predict what the next state would be based on the current state. B is the control-input matrix and Q is gaussian covariance matrix.

$$\hat{x}_{\bar{k}} = F\hat{x}_{k-1} + B_{u_{k-1}} \tag{2}$$

$$P_{\bar{k}} = FP_{k-1}F^T + Q \tag{3}$$

$\hat{x}$ is the predicted state of the system
P_k is the covariance matrix of the system variables

C. Update Phase

The final state variables for the current predict-update cycle are achieved by superimposing prediction stage and sensor reading Gaussian curves. Current variables are then set to the cycle's final-state variables.

$$\tilde{y}_k = z_k - H\hat{x}_{\bar{k}} \tag{4}$$

$$K_k = \overline{P_k}H^T(R + H\bar{P}_kH^T)^{-1} \tag{5}$$

$$\hat{x}_{\bar{k}} = \hat{x}_{\bar{k}} + K_k\tilde{y} \tag{6}$$

$$P_k = (1 - K_kH)P_k \tag{7}$$

y_k is the measurement residual
z_k is the measurement vector
v_k is the measurement noise vector (mean: 0, covariance: R)
K is the Kalman gain

D. 2D Model (Traditional Model)

In this model, we use the readings from the velocity sensor and the distance sensor to find the value of the acceleration at any given instance. Newtons laws of motion states that:

$$v^2 - u^2 = 2as \tag{8}$$

Using this we can derive:

$$a = \frac{v(t)^2 - v(t-1)^2}{2(x(t) - x(t-1))} \tag{9}$$

Since *v* has a standard deviation of 2 m/s and *x* has a standard deviation of 50 m/s, and the acceleration will have a standard deviation of 0.64 m/s^2.

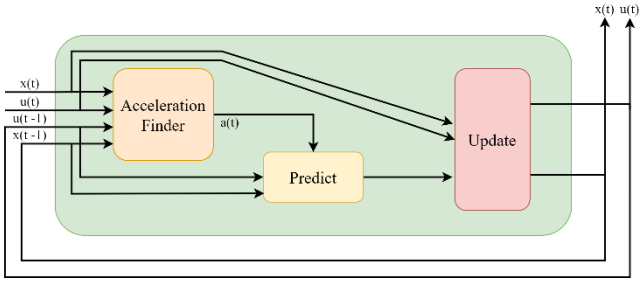

Fig. 1. Traditional Model

Where *x(t)* is the position of the vehicle read with the help of a sensor and *v(t)* is the velocity of the vehicle read from the speedometer. Fig. 1 depicts the functions of the system, that is the acceleration finder which maps to the NN model, the Predict function is ideally the KF and the Update function is the function that feeds the output inputs back to inputs for the NN for prediction of the next value.

TABLE I. Error Estimation in Traditional Model

Parameters	Range	
	Mean Range	***Standard Deviation***
Velocity	0 m/s – 12 m/s	2 m/s
Position	0 m – 1200 m	50 m
Acceleration	0 m/s^2 – 1.5 m/s^2	0.64 m/s^2

E. Long Short-Term Memory (LSTM)

The necessity and the option to select a sequential based machine learning model over other techniques like RNN and CNN, is that we need to hold the state of previous data, KF work and have context to store previous data and modify themselves to predict better, the closest methodology in the machine learning world is the LSTM model that has a memory and can store and learn from its previous values [6]. Using LSTM also prevents the vanishing-gradient problem from occurring which might ruin the efficiency of the KF than improving it.

The proposed model is based around a major assumption that there already exists a pre-trained Machine learning model which outputs a predictable variable when provided with a chaotic state. Fig. 2 below depicts the higher-level architecture of the described proposed model which is a generalised architecture.

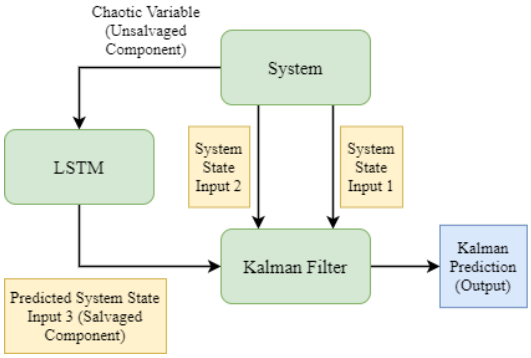

Fig. 2. Proposed Model

For experimental and observational purposes, we have used a vehicle velocity-position state model with the further addition of the heat of the engine. Since KF cannot use the heat to its benefit, the Machine learning model - built with a reverse Long Short-Term Memory Network (LSTM) is used to accurately determine the acceleration with the present state values of velocity and position.

The two-state model, with parameters namely velocity and position were converted to a three-state model with an added variable, acceleration. The error range calculations for the same are shown below:

TABLE II. ERROR ESTIMATION IN PROPOSED MODEL

Parameters	**Range**	
	Mean Range	***Standard Deviation***
Velocity	0 m/s – 12 m/s	2 m/s
Position	0 m – 1200 m	50 m
Acceleration	0 m/s^2 – 1.5 m/s^2	0.2 m/s^2

Fig. 3 below, depicts the experiment model used to apply the proposed model into a real life situation. Here *x(t)* is the position of the vehicle read with a sensor and *v(t)* is the velocity of the vehicle read from the speedometer.

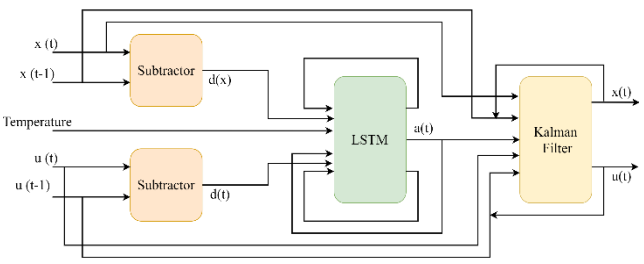

Fig. 3. Proposed Experiment Model

IV. EXPERIMENTAL RESULTS AND OBSERVATIONS

A. *Experiment Environment*

For the experimentation purposes, the scenario is that there is an 2019 Renault Duster Sports Utility Vehicle at an altitude of 560 m above sea level with a surrounding temperature of 302 K and the engine core temperature at 300 K. The vehicle travels for about 10 km in total with a maximum elevation of 500 m hitting a top RPM (rotations per minute) of 3600 during the journey. The temperature, velocity and position have been recorded during this journey. The highest temperature that was achieved was around 468 K.

B. *Assumptions*

1) Correlation: Prerequisite of sequential model which contains multiple LSTM layers to help correlate the engine temperature to the vehicle's acceleration with a standard deviation within 50% of the mean.

2) Acceleration: Initial acceleration of the vehicle is 0 m/s^2 and the KF is used to estimate values only once the vehicle starts moving.

3) Environment Climate: Normal temperature conditions without any rain/humidity/extreme climate.

4) Abnormalities: There are no abnormalities in the vehicle's engine.

C. *Test Results*

Fig. 4 depicts the acceleration readings calculated from the Sequential Model and the Newton's Laws of Motion respectively. From the graph, it is evident that the acceleration value predictions are smoothened and more consistent than its theoretical counterpart.

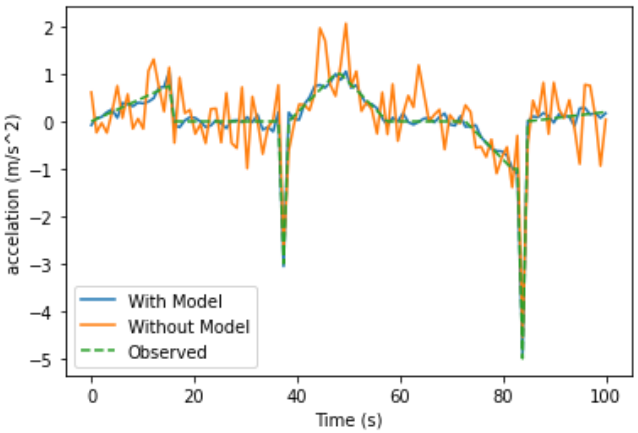

Fig. 4. Acceleration-time Graph

As observed from Fig. 5, without the acceleration parameter, the KF computes velocity with a root mean squared error of 1.659 whereas when the additional parameter, acceleration is provided to the model the root mean squared error falls to 0.429.

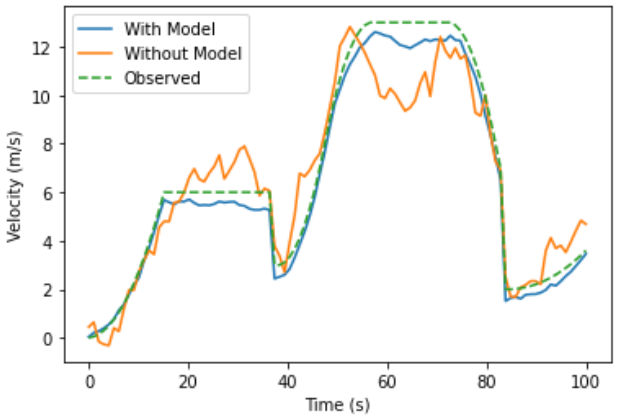

Fig. 5. Velocity-time Graph

Similarly in Fig. 6, the root mean squared error while determining position falls from 52.703 to 14.414 when acceleration values are provided to the model. With the aforementioned values of error metrics, we see that our model clearly outperforms the traditional KF for the given system.

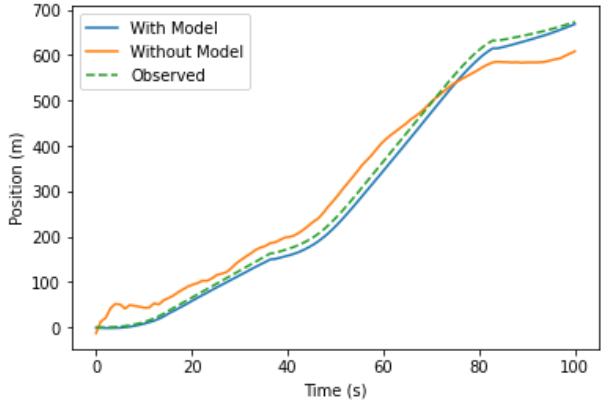

Fig. 6. Position-time Graph

V. Conclusion

The simulation results indicate the stark difference between using a higher degree initialized KF in contrast to the standard KF. With our simulation, the emphasis on the underutilization of chaotic variables is highlighted. While the results show exceptional improvement from the setup of a traditional KF, it cannot be used in all situations. The model lacks proper intuition unlike the KF. The proposed model is heavily dependent on the output of the NN model which converts a chaotic state into a predictable variable for the KF. The initial state of the environment is a major prerequisite for the successful run of the NN. This pitfall paves the way for future development along this front of the model. Future research related to this model should strive to achieve independence between the initial state and the NN model responsible.

References

[1] Z. Shi, "Incorporating Transformer and LSTM to Kalman Filter withEM algorithm for state estimation," no. May, 2021. Available: http://arxiv.org/abs/2105.00250.

[2] B. Millidge, A. Tschantz, A. Seth, and C. Buckley, "Neural Kalman Filtering," 2021. Available: http://arxiv.org/abs/2102.10021.

[3] G. Revach, N. Shlezinger, X. Ni, A. L. Escoriza, R. J. G. van Sloun, and Y. C. Eldar, "KalmanNet: Neural Network Aided Kalman Filtering for Partially Known Dynamics," 2021, [Online]. Available: http://arxiv.org/abs/2107.10043.

[4] M. Linderoth, K. Soltesz, A. Robertsson, and R. Johansson, "Initialization of the Kalman filter without assumptions on the initial state," *Proc. - IEEE Int. Conf. Robot. Autom.*, no. 1, pp. 4992–4997, 2011, doi: 10.1109/ICRA.2011.5979684.

[5] R. C. Wilson and L. H. Finkel, "A neural implementation of the Kalman filter," *Adv. Neural Inf. Process. Syst. 22 - Proc. 2009 Conf.*, pp. 2062–2070, 2009.

[6] R. Zhan and J. Wan, "Neural network-aided adaptive unscented Kalman filter for nonlinear state estimation," *IEEE Signal Process. Lett.*, vol. 13, no. 7, pp. 445–448, 2006, doi: 10.1109/LSP.2006.871854.