Maneuvering Target Tracking using the Autoencoder-Interacting Multiple Model Filter

Kirty Vedula*, Matthew L. Weiss*, Randy C. Paffenroth*, Joshua R. Uzarski‡, D. Richard Brown III*
*Worcester Polytechnic Institute, 100 Institute Road, Worcester, MA 01609
‡Soldier Protection and Survivability Directorate, U.S. Army CCDC-SC, Natick, USA 01760
Email: *{kpvedula, mlweiss, rcpaffenroth, drb}@wpi.edu, ‡ joshua.r.uzarski.civ@mail.mil

***Abstract*—This paper considers the problem of tracking and predicting the state of a dynamic system with stochastic dynamics and multiple modes of operation. A well-known approach to this problem is the "interacting multiple model" (IMM) estimator, which uses knowledge of the different modes of operation to update a bank of Kalman Filters (each optimal for a given mode of operation). The IMM combines estimates according to the posterior probability of the different modes. Despite their popularity, IMMs are known to sometimes be slow to detect mode switching, however, which can result in large state estimation errors. This paper addresses this problem by developing an Autoencoder-Interacting Multiple Model (AEIMM) algorithm. The AEIMM effectively embeds an IMM within an autoencoder framework to create a hybrid approach using both deep learning and classical tracking frameworks. The motivation for this approach is that the neural network can perform nonlinear transformations on the measurements to help the IMM more quickly identify mode changes. The effectiveness of the AEIMM is demonstrated in a maneuvering target tracking scenario. Numerical results show that the AEIMM outperforms classical tracking techniques as well as hybrid techniques and a Long Short-Term Memory network in this scenario.**

***Index Terms*—Kalman Filter, Interacting Multiple Model Filter, Autoencoder, Autoencoder-Kalman Filter, Deep Learning, Neural Networks, Target Tracking**

I. Introduction

Target tracking is used in many practical applications to accurately track objects with trajectories that have significant position derivatives of several orders. When not detected and compensated, the maneuvers can degrade the performance of the tracker and might lead to filter divergence [1]. A Kalman Filter (KF), or its non-linear variants such as Extended Kalman Filter (EKF) or particle filter, is commonly employed for tracking maneuvering targets [2].

Accurate tracking is possible only when we model the target motion appropriately. Some common dynamic models are the nearly constant velocity (NCV) model, the nearly constant acceleration (NCA) model and the coordinated turn (CT) model [3]. However, a single model is not always adequate to accurately describe the target's motion. In an Interacting Multiple Model (IMM) filter, two or more models are considered, and the model inaccuracy is addressed by facilitating an interaction between the models for different modes at the beginning of each filter cycle [1]. These are weighed accordingly by the conditional probabilities of switching between model modes. However, the IMM is a heuristic algorithm designed to yield a good performance only within the class of a fixed structure algorithms. It is not known to be generally optimal and does not guarantee robust performance. An IMM may give unsatisfactory results for particular scenarios such as nonlinear target dynamics during a turn [4] and sudden starts and stops of maneuvers such as model switching [5].

Funding 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 #U21-150

While a variety of traditional state estimation and time series prediction techniques, such as the EKF, particle filters, kernel methods and Bayesian inference [6], [7], [8], address state estimation and time series prediction in the non-linear context, the proposed approach in this paper is conceptually justified and distinguished by the use of a neural network. Specifically, leveraging the power of a neural network allows the parameters of the IMM to be learned from data, which is not possible with more traditional techniques. This will be further elaborated upon in II-G.

The main contribution of this paper is the design and implementation of an Autoencoder-Interacting Multiple Model (AEIMM) Filter, as an extension to the Autoencoder-Kalman Filter (AEKF) proposed in [9], [10], to improve IMM state estimation in the above challenging scenarios. In particular, while the AEKF addresses the issue of estimating the measurement noise covariance, the AEIMM addresses both the measurement noise covariance estimation and choosing the appropriate system dynamics. This is discussed further in section II-E.

Autoencoders are a class of unsupervised deep learning algorithms often used for dimensionality reduction. They have been successfully applied for end-to-end communication system design in channels with Gaussian and non-Gaussian noise [11] and for object tracking applications [12]. The AEKF in [9], [10] is a hybrid autoencoder KF algorithm that places a KF in the latent layer of a traditional autoencoder, as depicted in Figure 1. Further technical details of the AEKF are presented in section II-C and [9], [10].

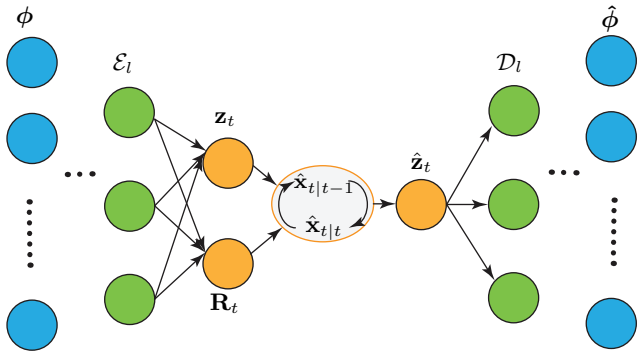

Fig. 1. The Autoencoder-Kalman Filter. Here the measurements, ϕ, are first transformed by the encoder portion of an autoencoder, where each layer of the autoencoder is represented by $\mathcal{E}_l$. The encoder outputs two sequences, $\mathbf{z}_t$ and $\mathbf{R}_t$, which are passed to a KF as measurements and associated measurement noise covariances. The KF's state estimate of $\mathbf{z}_t$, represented by $\hat{\mathbf{z}}_t$, is then mapped back to the measurement space via the decoder portion, whose layers are similarly represented by $\mathcal{D}_l$. The final output of the decoder is the state estimate of the original measurements, $\hat{\phi}$. Note the vertical ellipses represent the number of dimensions in a layer and horizontal ellipses represent the depth of the encoder and decoder. This is further discussed in Section II-C.

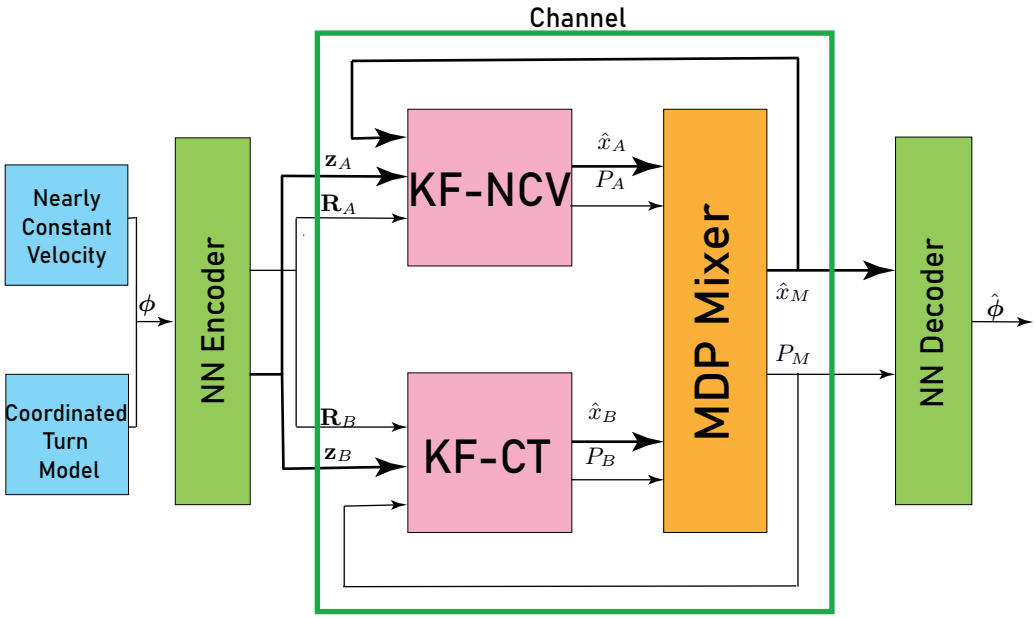

Fig. 2. Block Diagram for the Autoencoder-Interacting Multiple Model (AEIMM) filter.

The AEIMM is conceptually the same as the AEKF except an IMM appears in the autoencoder's latent space. We apply the AEIMM to a simulated flight data tracking problem with NCV and NCA models and compare with the KF, IMM and AEKF algorithms. We also compare our results with a traditional deep learning framework, the Long Short-Term Memory (LSTM) recurrent neural network [13], [14]. The improvements that we obtain using AEIMM in tracking maneuvering targets are particularly useful in some tactical applications such as threat evaluation, flight tracking and financial portfolio management.

II. System Model

Among the standard KF, IMM, AEKF and AEIMM variants, we consider three KF models in this paper, namely KF-NCV, KF-NCA and KF-CT with nearly constant velocity, nearly constant acceleration and coordinated turn models respectively. The states, statistics and covariance matrices vary for these models. In models involving the IMM, we consider a filter bank of two models with the aim to detect the *active* state model since we do not know which KF is providing the optimal updates. We pick our models such that one cannot be in the subspace of another. We also assume for the scope of this paper that we have a point target, that can result in at most a single measurement.

We consider a maneuvering target model

$$x_{t+1} = f_k(x_t, u_t) + w_t \tag{1}$$

$$y_t = h_k(x_t) + v_t \tag{2}$$

where x_t and y_t are the target state and observations respectively at time t. u_t is the unknown external input. w_t and v_t are the process and measurement noises and f_k and h_k are the state and observational models at mode k. We present the discrete-time setting here. These models are converted from continuous state-space models by performing a zero-order hold sampling and using linearized discretization [15].

A. Dynamic Models and Examples

1) Nearly Constant Velocity Model: The state vector for the nearly constant velocity model in $\mathbb{R}^2$ is represented by $X = \begin{bmatrix} x_t & y_t & v_t^x & v_t^y \end{bmatrix}^\top$ [1]. Here x_t and y_t are the respective position coordinates and v_t^x and v_t^y the corresponding velocities. It is described by

$$x_t = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x_{t-1} + \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \end{bmatrix} a_t \tag{3}$$

with $a_t \sim \mathcal{N}(0, {\sigma_a}^2)$.

2) Nearly Constant Acceleration Model: The state vector for the nearly constant acceleration model in $\mathbb{R}^2$ is represented by $X = \begin{bmatrix} x_t & y_t & v_t^x & v_t^y & a_t^x & a_t^y \end{bmatrix}^\top$. The position and velocity terms are the same as the NCV model and a_t^x and a_t^y represent the corresponding accelerations. It is described by

$$\begin{aligned} x_t = & \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} x_{t-1} \\ & + \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ \Delta t & 0 \\ 0 & \Delta t \\ 1 & 0 \\ 0 & 1 \end{bmatrix} a_t \end{aligned} \tag{4}$$

with $a_t \sim \mathcal{N}(0, {\sigma_a}^2)$.

3) Coordinated Turn Model: The coordinated turn model has nearly constant speed and turns at a constant rate. Its state vector in $\mathbb{R}^2$ is $X = \begin{bmatrix} x_t & y_t & v_t^x & v_t^y & \omega_t \end{bmatrix}^\top$. We have the turns $\dot{x} = v\cos(h), \dot{y} = v\sin(h)$, where $\dot{h} = \omega$ and we obtain $\ddot{x} = -\omega\dot{y}$ and $\ddot{y} = \omega\dot{x}$, where ω is the turn rate for modeling sharp maneuvers and simulating high coordinated turn rates for short time periods. The

dynamics are given by

$$x_t = \begin{bmatrix} 1 & 0 & \frac{\sin(\omega\Delta t)}{\omega} & -\frac{1-\cos(\omega\Delta t)}{\omega} & 0 \\ 0 & 1 & \cos(\omega\Delta t) & -\sin(\omega\Delta t) & 0 \\ 0 & 0 & \frac{1-\cos(\omega\Delta t)}{\omega} & \frac{\sin(\omega\Delta t)}{\omega} & 0 \\ 0 & 0 & \sin(\omega\Delta t) & \cos(\omega\Delta t) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} x_{t-1} + \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 & 0 \\ 0 & \frac{1}{2}\Delta t^2 & 0 \\ \Delta t & 0 & 0 \\ 0 & \Delta t & 0 \\ 0 & 0 & 1 \end{bmatrix} a_t. \tag{5}$$

with $a_t \sim \mathcal{N}(0, {\sigma_a}^2)$.

B. Kalman Filter and Extended Kalman Filter

For a linear state-space model with Gaussian noise and known process and measurement noise covariances, the optimal solution is provided by the KF with the standard recursive prediction and update equations that calculate estimates and predictions [16]. Since KF literature may be familiar to the readers, we briefly describe the main ideas. Further details can be found in [9], [17], [18], [19], [20].

1) The prediction step consists of projecting the previous state estimate, $\hat{\mathbf{x}}_{t|t}$, forward one step via the linear transformation $\hat{\mathbf{x}}_{t+1|t} = \mathbf{F}_{t+1}\hat{\mathbf{x}}_{t|t}$.
2) The final update performs a linear combination of $\hat{\mathbf{x}}_{t|t}$ and measurement residual $\tilde{\mathbf{z}}_t$, where the relative weighing is determined by Kalman Gain matrix $\mathbf{K}_t$ and the observation matrix $\mathbf{H}_t$.
3) During both the prediction and update stages, the covariances of both state estimates is also calculated.

The EKF is based on linearization of the nonlinear dynamic and measurement equations about the current state estimates using Taylor Series expansions. Assuming initial a priori estimates $\hat{\mathbf{x}}_{0|0}$ and $\mathbf{P}_{0|0} = \Pi_0$, just as in the KF, we have an additional linearizing step with the continuous time update equation $\hat{x}_{t|t-1} = f(\hat{x}_{t-1|t-1})$ with $f_t^T = \nabla_x f^\top(x)$ at $x = \hat{x}_{t|t}$ and other steps similar to KF.

C. AEKF

A detailed description of the AEKF, along with a comparison to other deep learning-KF hybrid models, appears in [9], [10]. Here we present a brief overview and some details on the training of the AEKF.

In the AEKF, shown in Figure 1, the actual measurements are now represented by $\boldsymbol{\phi}$, which is mapped via a composition of neural network layers to two new variables: $\mathbf{z}_t$ and $\mathbf{R}_t$. These comprise the input measurements and associated measurement noise covariance passed to the KF. The output of the KF, $\hat{\mathbf{z}}_t$, is then mapped back to the measurement space via the decoder portion of the AEKF. Here the output, $\hat{\boldsymbol{\phi}}$, is a filtered version of the original input $\boldsymbol{\phi}$. At this point the question of how to train the AEKF may arise. Since the encoder and decoder portions of the AEKF are trained via back propagation, the cost function should compare $\boldsymbol{\phi}$ and $\hat{\boldsymbol{\phi}}$ in some way. However, since the actual ground truth is not known this appears problematic. The solution to this is to train with simulated data where the ground truth is known. In the context of deep learning, this technique is known as domain randomization [21], [9], [10].

In the context of domain randomization, the loss function for this architecture is given by the following.

$$\mathcal{L} = \min_{\boldsymbol{\theta}} \sum_i \left\| \boldsymbol{\phi}_i^{(g)} - \hat{\boldsymbol{\phi}}_i(\boldsymbol{\phi}_{i-1}, \boldsymbol{\phi}_{i-2}, \dots) \right\|_F^2 \tag{6}$$

where i indexes the training sample, $\boldsymbol{\phi}_i^{(g)}$ is the ground truth, $\hat{\boldsymbol{\phi}}_i(\boldsymbol{\phi}_{i-1}, \boldsymbol{\phi}_{i-2}, \dots)$ the AEKF's estimate of $\boldsymbol{\phi}_i$ and $\boldsymbol{\theta}$ represents the parameters learned by the encoder and decoder portions. We propose to leverage domain randomization as done in [9], [10] which allows training over a range of parameter values in simulation, thus overcoming the need to have knowledge of the real-data ground truth to train a deep learning model.

D. Interacting Multiple Model

We consider the problem of tracking the states of the systems with two or more dynamic models. We summarize the algorithm briefly here and direct the reader to [1] for a detailed treatment.

Considering the state model in (2), we can model the behavior at each mode as a time-invariant Markov chain with a transition probability matrix (TPM). The IMM is an iterative algorithm that estimates the blended states and covariances at each step given measurements from two or more KFs using their initial conditions, states and their associated covariances. For the state $x_{t-1|t-1}^{i-1}$, covariance $P_{t-1|t-1}^{i-1}$ at a given mode $i \in 1, \dots, N_r$, each step of the IMM filter performs the following:

1) Calculates mixing probabilities $\{\mu_{t-1|t-1}^{ij}\}_{i,j=1}^{N_r}$, mixed estimates $\{\hat{x}_{t-1|t-1}^{0i}\}_{i=1}^{N_r}$ and covariances $\{\Sigma_{t-1|t-1}^{0i}\}_{i=1}^{N_r}$.
2) Calculates predicted estimates and covariances from mixed estimates in the previous step for i^{th} model, $i \in 1, \dots, N_r$.
3) Calculates updated estimates and covariances from the predicted estimates for i^{th} model, $i = 1, \dots, N_r$ and calculates the updated mode probability.
4) Calculates output state and covariance estimates.

E. AEIMM

The AEIMM is conceptually similar to the AEKF in that the KF portion of the AEKF is replaced by an IMM, as shown in Figure 2. Both the AEKF and AEIMM address the issue of estimating measurement noise covariances. However, in the case of the AEIMM, we allow the encoder portion to learn different measurements and their associated measurement noise covariances *for each of the two KFs in the AEIMM*, represented by $\mathbf{z}_A, \mathbf{z}_B$ and $\mathbf{R}_A, \mathbf{R}_B$ respectively. The primary motivation for this feature is that learning different measurements and associated measurement noise covariances for each KF in the IMM will assist the Markov Decision Process (MDP) mixer in weighting the appropriate KF. This, in turn, will help the IMM weight the system dynamics appropriately. The multiple encoder outputs are passed to two KFs which output $\hat{x}_A, P_A$ and $\hat{x}_B, P_B$ respectively. These are then passed into a Markov Decision Process mixer that returns $\hat{x}_M, P_M$. Lastly, $\hat{x}_M, P_M$ are (a) passed back to the IMM for the next iteration and (b) passed to the decoder, which maps these values to the final output $\hat{\boldsymbol{\phi}}$.

F. LSTM

An LSTM is a type of recurrent neural network (RNN) initially developed to solve the vanishing gradient problem in RNNs [13]. As they are able to learn long term temporal dependencies, LSTMs are well suited for time series classification, prediction and state estimation. For our purposes, the LSTM is included for comparison as the AEKF and AEIMM both leverage deep learning.

G. Model Justification

Here we would like to take a moment and provide some justification for designing a neural network-Kalman Filter hybrid algorithm as opposed to (a) a purely neural network-based algorithm and (b)

other state estimation algorithms such as particle filters. Note, the comments below apply equally to the AEIMM.

First, as we demonstrate below, the AEKF or AEIMM outperforms an LSTM in both single and multiple turn simulated tracking problems. Similar results were also observed for different state estimation problems involving Gaussian and non-Gaussian noise in [9], [10]. Additionally, with the AEKF we are training a neural network to learn a measurement noise covariance matrix for each element of a time series, and not simply a single measurement noise covariance matrix for the entire signal. This is the key to the AEKF's ability to outperform standard Kalman Filter algorithms and an LSTM. Furthermore, we were able to derive a theorem that explains how the encoder portion of the AEKF should behave in the context of learning the measurement noise covariance matrix. This was due to the fact the Kalman Filter can be analyzed with standard matrix analysis. From the point of view of the LSTM, it is difficult to determine what part of the algorithm is analogous (if anything) to the Kalman Filter's measurement noise covariance matrix, making such an analysis problematic. In the context of the AEIMM, if we know how a given problem's model dynamics vary (as we do in the examples herein) we can input these dynamical models into the AEIMM's IMM. In the case of the LSTM, how does one introduce this domain knowledge? In the larger scheme of things, the AEKF benefits from a mutually supportive combination of the power of a neural network and mathematically and physically principled domain knowledge.

The question of why we did not consider a non-linear extension of the Kalman Filter to deal with the non-linear domain is related to our use of a neural network. By introducing a neural network the AEKF can be trained for each given application domain. That is, the encoder and decoder portions of the AEKF will learn different transformations for different problems and/or datasets. Thus it is more adaptive than an Extended Kalman Filter, while more structured than a particle filter.

Lastly, while the Kalman Filter is known to be optimal when the random variables involved are Gaussian, this requires a correctly chosen state transition matrix, observation matrix, process noise covariance matrix and measurement noise covariance matrix, represented by $\mathbf{F}$, $\mathbf{H}$, $\mathbf{Q}$ and $\mathbf{R}$ respectively. Thus, even in the case of all processes involved being Gaussian, the Kalman Filter performance is still constrained by how well the choices of $\mathbf{F}$, $\mathbf{H}$, $\mathbf{Q}$ and $\mathbf{R}$ match the real problem domain. The addition of a neural network in the AEKF allows for the possibility that the $\mathbf{z}$ and $\mathbf{R}$ learned by the AEKF are transformed in such as way as to improve Kalman Filter performance given fixed $\mathbf{F}$, $\mathbf{H}$ and $\mathbf{Q}$.

III. Results

A. Test Protocol

We test our approach 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). Gaussian noise is then added to these smooth ground truth flight paths. Using these noisy simulated flight paths, we compare the state estimation capabilities of the following models: KF, IMM, AEKF, AEIMM and LSTM. Each model's state estimation is then compared with the actual ground truth and the corresponding RMSE is reported for each model. Note that in this phase, the ground truth is used *only for model evaluation* and does not affect each model's state estimate in any way. As the state estimate and ground truth are two dimensional, the RMSE is calculated by taking the square root of the Frobenius norm between the state estimate and ground truth.

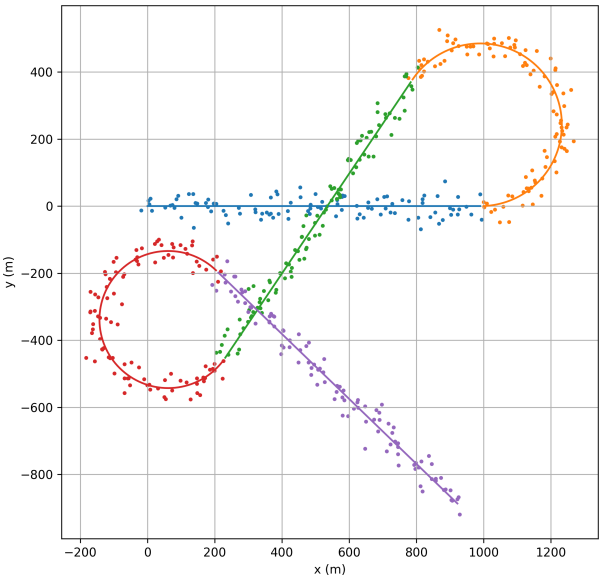

Fig. 3. Sample Simulated Two-Turn Flight Path with Gaussian Noise.

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

$$r(t+dt) = \big(x(t) + v_x dt\big)\hat{i} + \big(y(t) + v_y dt\big)\hat{j} \tag{7}$$

$$v(t+dt) = v_x\hat{i} + v_y\hat{j} \tag{8}$$

where $r(t) = (x(t), y(t))$ is the position vector at time t and $v_0 = (v_x, v_y)$ is the initial velocity vector at the beginning of the segment. The coordinated turn model is similarly defined by

$$r(t+dt) = R\cos\Big(\frac{\|v_0\|\,(kdt)}{R}\Big)\hat{i} + R\sin\Big(\frac{\|v_0\|\,(kdt)}{R}\Big)\hat{j} \tag{9}$$

$$v(t+dt) = -v_0\sin\Big(\frac{\|v_0\|\,(kdt)}{R}\Big)\hat{i} + v_0\cos\Big(\frac{\|v_0\|\,(kdt)}{R}\Big)\hat{j} \tag{10}$$

where R is the radius 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 $\theta = \frac{v_0 t}{R}$ we can express the total angle turned through at the k^{th} increment as $k\delta\theta = \frac{v_0}{R}k\delta t$ with $k = 0, 2, ..., N-1$. Note that (9) and (10) assume rotation about the origin starting from $\theta = 0$. Thus, appropriate translation and rotations were applied to the results of (9) and (10) 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 3.

While we use simulated flight paths to demonstrate the efficacy of the AEIMM, it should be noted the AEIMM is designed in a general manner to make it applicable to any application or scenario where an IMM is appropriate.

B. 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 noise $\mathcal{N}(0, 20)$. Each of the three flight segments lasts 10 seconds with a

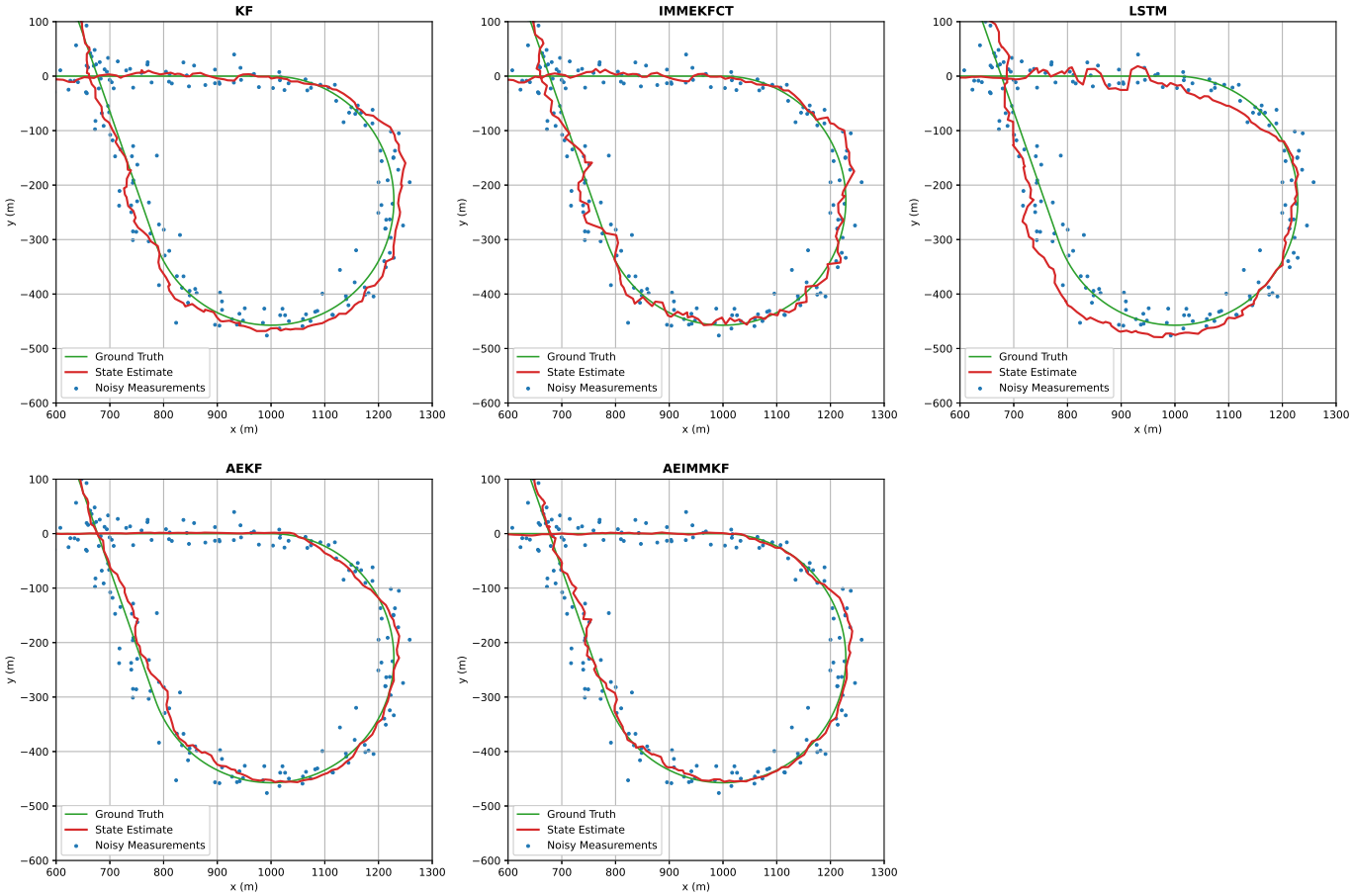

Fig. 4. Turn segment from a test set sample trial. Here the KF, IMM and LSTM estimates are, generally, less smooth than the AEKF and AEIMM estimates. Furthermore, the KF, IMM and LSTM have more difficulty estimating the ground truth on the turn than the AEKF and AEIMM. Note the RMSE increases monotonically from KF, IMM, LSTM, AEKF through AEIMM.

sampling frequency of 10 Hz. The transition probabilities for models involving the IMM are 0.95 and 0.05.

We compare five models: KF, IMM, AEKF, AEIMM and LSTM. The KF and AEKF models consist of NCV models with process noise covariance given by the second term in (3). The IMM consists of NCV and CT models, with the process noise covariance given by the second terms in (3) and (5) respectively. Lastly, the AEIMM consists of NCV and NCA models, with the process noise covariance given by the second terms in (3) and (4) respectively. The reason the CT model was not used in the AEIMM is the dimensions of the Kalman Filters in both the AEKF and AEIMM were $\mathbb{R}^8$. This is a direct result of the fact the actual measurements in $\mathbb{R}^2$ can be mapped to and from $\mathbb{R}^8$ via the encoder and decoder portions of the AEKF and AEIMM. As the standard CT model is defined in $\mathbb{R}^2$, generalizing this to $\mathbb{R}^8$ proved difficult.

The test set consists of 1000 simulated single turn flight paths. For each model, the reported RMSE is computed by averaging the RMSE on each of the 1000 test paths computed using the ground truth and state estimates. Results are shown in Table I, where the RMSE ratio is the ratio of each model's RMSE to the KF's RMSE. Here the models, from highest to lowest RMSE, are KF, IMM, AEKF and AEIMM. The fact that the IMM shows better performance than the KF is not surprising. However, both these models show improvement when combined with a neural network. For visualization, an example of one test trial with the ground truth, noisy simulated measurements and state estimate is shown in Figure 4.

TABLE I
SINGLE TURN TEST RMSE RESULTS.

Model	RMSE	Ratio
KF	10.90	1.00
IMM	10.68	0.98
LSTM	11.15	1.02
AEKF	9.49	0.87
AEIMM	8.19	0.75

C. Multiple Turn Results

The multiple turn flight path results were based upon the same parameters as the single turn results apart from what is presented below. In the case of the multiple turns, the turn radius was between 120 and 180 meters with each of the flight paths (three constant velocity segments and two turn segments) lasting 6 second with a sampling frequency of 10 Hz.

The parameters for the KF, IMM, AEKF, AEIMM and LSTM, the number of sample curves in the test set and model evaluation procedures were the same as in the single turn case apart from using multiple turn flight paths. Similar trends to the single turn case were

TABLE II
MULTIPLE TURN TEST RMSE RESULTS.

Model	RMSE	Ratio
KF	13.54	1.00
IMM	10.27	0.76
LSTM	14.72	1.09
AEKF	10.68	0.79
AEIMM	10.00	0.74

found among the each model's performance on the test set and are shown in Table II.

IV. CONCLUSION

We introduce a new deep learning Kalman Filter hybrid framework the Autoencoder-Interacting Multiple Model, as an extension to the Autoencoder-Kalman Filter, to solve challenging maneuvering target tracking problems. We provide a proof-of-concept demonstration with simulated flight tracking data and compare it against state-of-the-art methods in tracking such as the Interacting Multiple Model, traditional deep learning methods such as Long Short-Term Memory RNN along with the Autoencoder-Kalman Filter. Generally, our results indicate the combination of deep learning and traditional filtering techniques outperform traditional approaches by themselves. Specifically, we demonstrate, among the five methods considered, the Autoencoder-Interacting Multiple Model shows the best performance on both the single and multiple turn datasets.

Given the success of the AEIMM presented in this paper, some pertinent future research questions are:

1) If we replace the IMM's MDP with a neural network and keep the KFs intact, does the neural network mix or augment the outputs from the IMM's bank of KFs?
2) Is it better to use two neural networks on the encoder side, one for each dynamic model that output $\mathbf{z}_A, \mathbf{R}_A$ and $\mathbf{z}_B, \mathbf{R}_B$ respectively instead of a single encoder?

REFERENCES

[1] Yaakov Bar-Shalom, X. Rong Li, and Thiagalingam Kirubarajan, *Estimation with Appications to Tracking and Navigation*, Wiley, New York, 2001.

[2] H. A. P. Blom and Y. Bar-Shalom, "The interacting multiple model algorithm for systems with markovian switching coefficients," *IEEE Transactions on Automatic Control*, vol. 33, no. 8, pp. 780–783, 1988.

[3] Pravas R Mahapatra and Kishore Mehrotra, "Mixed coordinate tracking of generalized maneuvering targets using acceleration and jerk models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 36, no. 3, pp. 992–1000, 2000.

[4] Michael Roth, Gustaf Hendeby, and Fredrik Gustafsson, "Ekf/ukf maneuvering target tracking using coordinated turn models with polar/cartesian velocity," in *17th International Conference on Information Fusion (FUSION)*. IEEE, 2014, pp. 1–8.

[5] M. E. Farmer, Rein-Lien Hsu, and A. K. Jain, "Interacting multiple model (imm) kalman filters for robust high speed human motion tracking," in *Object recognition supported by user interaction for service robots*, 2002, vol. 2, pp. 20–23 vol.2.

[6] Cédric Richard, José Carlos M Bermudez, and Paul Honeine, "Online prediction of time series data with kernels," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 1058–1067, 2008.

[7] Amrit Singh Bedi, Alec Koppel, Ketan Rajawat, and Brian M Sadler, "Nonstationary nonparametric online learning: Balancing dynamic regret and model parsimony," *arXiv preprint arXiv:1909.05442*, 2019.

[8] Amir-massoud Farahmand, Sepideh Pourazarm, and Daniel Nikovski, "Random projection filter bank for time series data," in *Advances in Neural Information Processing Systems*, 2017, pp. 6562–6572.

[9] M. Weiss, R. C. Paffenroth, J. R. Whitehill, and J. R. Uzarski, "Deep learning with domain randomization for optimal filtering," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019, pp. 1779–1786.

[10] M. L. Weiss, R. C. Paffenroth, and J. R. Uzarski, "The autoencoder-kalman filter: Theory and practice," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, 2019, pp. 2176–2179.

[11] K. Vedula, R. Paffenroth, and D. R. Brown, "Joint coding and modulation in the ultra-short blocklength regime for bernoulli-gaussian impulsive noise channels using autoencoders," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 5065–5069.

[12] B. Beşbınar and A. A. Alatan, "Visual object tracking with autoencoder representations," in *2016 24th Signal Processing and Communication Application Conference (SIU)*, 2016, pp. 2041–2044.

[13] Sepp Hochreiter and J Urgen Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] Carlton Downey, Ahmed Hefny, Byron Boots, Geoffrey J Gordon, and Boyue Li, "Predictive state recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 6053–6064.

[15] C.T. Chen, *Linear System Theory and Design*, Oxford series in electrical and computer engineering. Oxford University Press, 2013.

[16] Steven M. Kay, *Fundamentals of Statistical Signal Processing: Estimation Theory*, Prentice-Hall, Inc., USA, 1993.

[17] Brian D.O. Anderson and John B. Moore, *Optimal Filtering*, Dover Publications, 1979.

[18] R E Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[19] Dan Simon, *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*, Wiley, Hoboken, 1st edition, 2006.

[20] Peter A. Ruymgaart, TSu T. Soong, and Tsu Soong T., *Mathematics of Kalman-Bucy Filtering*, vol. 136, Springer-Verlag, Berlin Heidelberg New York Tokyo, 1st edition, 1985.

[21] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.