HAL
open science

NNAKF: A NEURAL NETWORK ADAPTED KALMAN FILTER FOR TARGET TRACKING

Sami Jouaber, Silvere Bonnabel, Santiago Velasco-Forero, Marion Pilte

To cite this version:

Sami Jouaber, Silvere Bonnabel, Santiago Velasco-Forero, Marion Pilte. NNAKF: A NEURAL NETWORK ADAPTED KALMAN FILTER FOR TARGET TRACKING. ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Jun 2021, Toronto, France. pp.4075-4079, 10.1109/ICASSP39728.2021.9414681 . hal-03463046

HAL Id: hal-03463046

https://unc.hal.science/hal-03463046

Submitted on 2 Dec 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

NNAKF: A NEURAL NETWORK ADAPTED KALMAN FILTER FOR TARGET TRACKING

Sami Jouaber[⋆‡], *Silvère Bonnabel*[⋆§], *Santiago Velasco-Forero*[†] *and Marion Pilté*[‡]

[⋆]MINES ParisTech, Université PSL, Centre de Robotique (CAOR), 75006 Paris, France
[†]MINES ParisTech, Université PSL, Centre de Morphologie
Mathématique (CMM), 77300 Fontainebleau, France
[‡]THALES LAS, Massy, France
[§]Université de Nouvelle Calédonie, Nouméa, France

ABSTRACT

An adaptive three-dimensional Kalman filter for the tracking of maneuvering targets in three dimensions is proposed. In the radar industry, numerous trackers are based on a constant velocity model, with a process noise covariance matrix Q which is adapted in real time to enhance tracking: it is kept at moderate values during straight lines where the constant velocity assumption applies and is increased during maneuvers. In the present paper we advocate a novel method to increase Q during maneuvers (and hence the Kalman gains) based on a recurrent neural network (RNN). The difficulty and the interest of our approach lies in the fact the neural network is trained together with the filter, by backpropagation through the filter, and hence learns the covariance matrix such as to directly maximize the accuracy of the final output.

Index Terms— Recurrent Neural Network (RNN), Kalman Filter, Radar Tracking

1. INTRODUCTION

The Kalman filter (KF) has long been used as an estimator for target tracking in radar applications. However, as the motion of the target is unknown, the Kalman filter has to assume a dynamical model, and radar measurements allow the filter to correct the estimated state. As the motion of the target is unpredictable, a simple yet sensible assumption is that the velocity vector is constant. The process noise covariance matrix Q of the Kalman filter is an indication of discrepancy between the actual motion and the model, which informs the filter how relevant the model is. It is a parameter that must be tuned by the user. Figure 1 illustrates the corresponding trade-off: if Q is set low the filter trusts the model and thus obtains good performance when actual velocity is close to being constant but degrades when maneuvers are performed, whereas setting Q at a higher value makes the filter trust more the radar measurements, which is relevant during maneuvers but leads to degraded performances in straight lines. It is hence quite natural to adapt Q so as to increase it during turns while keeping it moderate during constant speed flight. Such adaptation can be traced back to the work of Castella [1]. The method consists in evaluating the tracking residuals in each coordinate, which are a measure of filter performance and suitability to the model, and to increase Q when they exceed a threshold. Due to its simplicity of implementation, its robustness, and its relevance to the tracking of many targets such as airliners, the method is still very lively within the radar industry[1].

In this paper, building upon the recent advances of deep learning, we propose to use a RNN to dynamically adapt the process noise of a constant velocity Kalman filter, which we called a Neural Network Adapted Kalman Filter (NNAKF). Our approach is the first to our knowledge to use RNN to precisely learn what the adaptation should be in adaptive Kalman filtering for target tracking. Moreover, at a technical level, the NN is trained with the KF in the loop. This is difficult because all the KF operations based on a Riccati equation need to be part of a complex Recurrent NN for which training can be easily unstable. This should be put in contrast with approaches where a NN increases Q directly based on the observations only, as in e.g., [2].

1.1. Literature Review and Paper's Organization

For mono-target radar tracking, the Kalman filter and its variants [3, 4, 5, 6, 7] have long been brought to bear. As the actual underlying dynamical of the target is never known to the radar tracker, most approaches involving Kalman filters revolve

[1]As far as we can tell from our experience within the radar industry.

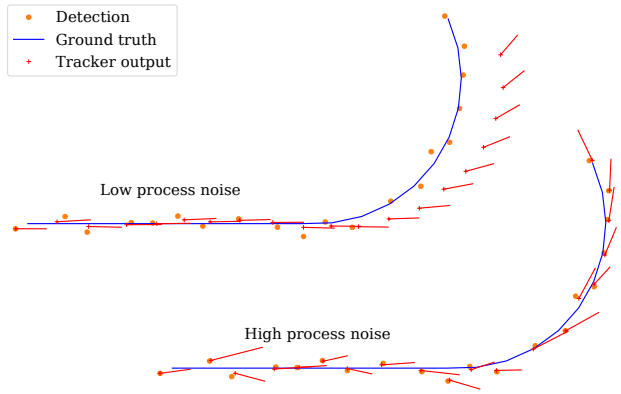

Fig. 1. Two-dimensional example of filtering using a Kalman filter with a Constant Speed model and two different settings of the noise process. Blue line is the real trajectory, orange dots are the detections provided to the filter, and red crosses and lines represent (respectively) the estimated states and the estimated velocity vectors.

around dynamically adapting the uncertainty parameters as in [1], or using multiple dynamical models with weights that are dynamically adapted, as in the Interacting Multiple Models [8] (IMM), which is deemed state of the art.

On the other hand, machine learning algorithms outperform standard methods in many fields. Since the operations composing the KF are differentiable, modern gradient-based optimization algorithms such as ADAM [9], RMSprop [10] or Adadelta [11] can be used to set the filters parameters to fit a given set of trajectories. As stated by Krishnan *et al.* [12] in 2015, and Chin [13] before them, it is also possible to incorporate Neural Networks (NN) into a KF to make it applicable on a wide range of problems. For instance in 2016 Haarnoja *et al.* [14] proposed a deep learning preprocessing to make the filter work with images as input. In 2017 Coskun *et al.* [15] proposed to also add Long Short-Term Memory cells [16] (LSTM) to model the unknown dynamics of the studied state. Later several versions of hybridization between KF and NN were proposed. In 2019, Brossard *et al.* [2] used a Convolutional Neural Network (CNN) to adapt the measurement noise matrix for dead-reckoning of ground vehicles solely based on inertial measurement. In the same year, Ju *et al.* [17] added a NN involving CNN and LSTM to predict accelerations due to proximity to other targets for car trajectories prediction. Still in 2019, Mercat *et al.* [18] proposed a NN for multi-target trajectories forecasting. These algorithms use NN when the specific problem to solve does not fit the required assumptions to properly use a KF (essentially lack of knowledge of dynamics and/or interactions or use of pseudo-measurements). However, for our particular problem measurements are already provided as multivariate normal distributions (MND).

The rest of this paper is organized as follows. In Section 2, KF and the proposed adaptation are described. In Section 3, the dataset, training parameters and results are presented.

2. THE PROPOSED ADAPTATION

2.1. Kalman Filter

To track an aircraft, a sequence of position measurements at times t_n is provided to the filter as MNDs represented by their mean vectors Y_n and covariance matrices R_n. The state of the target is also modeled as a MND represented by X_n and P_n. The KF works as a recursive filter, updating the state for each new measurement as follows.

Evolution/Propagation:

$$X_{n|n-1} = F_n X_{n-1|n-1} \tag{1}$$

$$P_{n|n-1} = F_n P_{n-1|n-1} F_n^T + Q_n \tag{2}$$

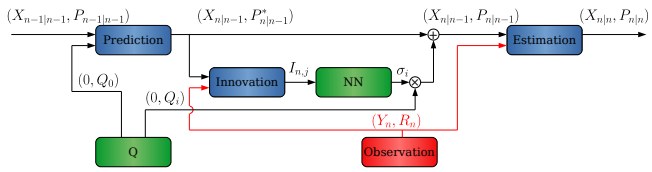

Fig. 2. Schematic representation of the proposed filter (NNAKF). Input data is represented in red, and trainable components of the filter are represented in green.

Gain computation:

$$Z_n = Y_n - H_n X_{n|n-1} \tag{3}$$
$$S_n = H_n P_{n|n-1} H_n^T + R_n \tag{4}$$
$$K_n = P_{n|n-1} H_n^T S_n^{-1} \tag{5}$$

Estimation:

$$X_{n|n} = X_{n|n-1} + K_n Z_n \tag{6}$$
$$P_{n|n} = (I - K_n H_n) P_{n|n-1} \tag{7}$$

Here we consider an evolution governed by a system of linear differential equations $\frac{dX}{dt} = AX$, A depending on the chosen evolution model. The process noise is modeled as a Brownian motion with a magnitude given by a covariance matrix $\tilde{Q}$, such as :

$$\Delta t_n = t_n - t_{n+1} \tag{8}$$
$$F_n = e^{A\Delta t_n} \tag{9}$$
$$Q_n = \int_0^{\Delta t_n} e^{As} \tilde{Q} e^{A^T s} ds \tag{10}$$

Since every operation composing the KF is differentiable, the matrix $\tilde{Q}$ can be automatically tuned [12] given a dataset using gradient based optimization algorithms as $\tilde{Q} = WW^T$ where W is a matrix full of trainable parameters. The KF can then be trained in a similar fashion to RNNs, for instance using backpropagation through time [19].

2.2. Neural Network Adaptation for Kalman Filters (NNAKF)

Tuning the process noise can be tricky [20]. If it is low, the estimation of the filter will be great when the trajectory respects the chosen model, but it will be poor when it doesn't. If the process noise is high, the filter will mostly trust the measurement and will become noise-sensitive. Sometimes an acceptable trade-off cannot even be found. Figure 1 illustrates the influence of the process noise parameter on the tracking.

The proposed adaptation works by adding more or less noise after propagation. To this aim, a set of $N+1$ covariance matrices $(\tilde{Q}_i)_{i\in[|0,N|]}$ is tuned, N being a hyper-parameter of the filter. A first evolution step is made to give a first approximation $(X_{n|n-1}, P^*_{n|n-1})$ of the state at time t_n, using $\tilde{Q}_0$ as Brownian motion magnitude. The normalized squared innovation $I_{n,j}$ is then computed on each axis just like for the Castella adaptation [1] :

$$\forall j \in \{x, y, z\}, I_{n,j} = \frac{Z_{n,j}^2}{S_{n,j,j}} \tag{11}$$

These values are used as inputs to a NN that outputs N coefficients bounded between 0 and 1 via a sigmoid function. An additional process noise is used, depending on the coefficients σ_i, such as the total magnitude of the Brownian motion added to the evolution between t_{n-1} and t_n is:

$$\tilde{Q} = \tilde{Q}_0 + \sum_{i=1}^{N} \sigma_i \tilde{Q}_i \tag{12}$$

A schematic representation of the NNAKF is given in Figure 2. This adaptation was inspired by the Castella adaptation, and aims to add more noise when and only when the chosen model does not match with the trajectory. The main differences with the classical adaptation are as follows. First, adaptation is more complex, and it is able to keep memory of its past thanks to the LSTM cell. Then, we adapt the full process noise covariance matrix, not only the diagonal part related to speed.

3. EXPERIMENTS

3.1. Dataset

Filter	Type 1	Type 2	Type 3	Type 4
Oracle	19.26	15.50	13.31	13.94
KF	46.24	48.34	55.55	46.62
Castella	40.72	44.11	50.16	42.33
Vaidehi	45.01	47.57	55.58	45.56
LSTM-KF	69.69	71.40	74.65	75.04
NNAKF (ours)	**38.13**	**41.64**	**46.40**	**41.39**

Table 1. Average error of the estimated horizontal position, values are given in meters.

Filter	Type 1	Type 2	Type 3	Type 4
Oracle	8.52	8.87	8.91	8.63
KF	31.57	31.88	32.38	46.13
Castella	31.93	32.23	32.73	45.52
Vaidehi	31.92	32.22	32.70	45.79
LSTM-KF	31.30	34.33	35.26	**42.22**
NNAKF (ours)	**26.98**	**27.37**	**27.95**	45.68

Table 2. Average error of the estimated altitude, values are given in meters.

Filter	Type 1	Type 2	Type 3	Type 4
Oracle	0.56	0.45	0.34	0.41
KF	3.65	3.67	4.02	3.64
Castella	2.47	2.66	3.32	2.54
Vaidehi	3.69	3.71	4.04	3.68
LSTM-KF	17.80	20.59	39.79	29.01
NNAKF (ours)	**1.95**	**2.24**	**3.17**	**2.12**

Table 3. Average error of the estimated horizontal speed, values are given in meters per seconds.

Filter	Type 1	Type 2	Type 3	Type 4
Oracle	1.42	0.39	0.20	0.07
KF	17.53	6.60	5.26	1.43
Castella	13.26	5.55	4.37	1.22
Vaidehi	17.51	6.46	5.14	1.41
LSTM-KF	78.23	71.11	66.93	22.94
NNAKF (ours)	**11.07**	**5.10**	**3.88**	**1.18**

Table 4. Average error of the estimated heading, values are given in degrees (°).

The dataset is constructed using simulations. First, the non-noisy 3-D trajectories are built using three randomly generated flight commands: turn rate, horizontal acceleration and vertical speed. To generate any of these commands, we start from a Gaussian white noise with a given magnitude, then it is processed by a low-pass convolution filter with a given cut-off frequency to get a smooth signal, and finally it is set to zero when its absolute value is under a given threshold. This process leads to a signal that can stay at zero and sometimes exhibit a peak with positive or negative values. Three parameters are used to control the average frequency of occurrence, length and magnitude of the peaks. In addition to these three parameters for each of the three flight commands, two more are used to bound the ground speed.

Altogether, the simulator can randomly generate a realistic trajectory using a set of eleven parameters. Four different combinations were used in the generated datasets with equal proportions to ensure the diversity of the trajectories. Then the detections of the aircraft by a sensor are simulated by adding a Gaussian noise to the position of the trajectory at different selected times. For each of the four types of trajectory, 1500 trajectories were generated, 1000 of them were used as training dataset, 250 as validation dataset, and 250 as testing dataset. All trajectories are 10-minutes long and detections were made at a regular time step of $4s$.

3.2. Filters

Different variations of Kalman filters were compared:

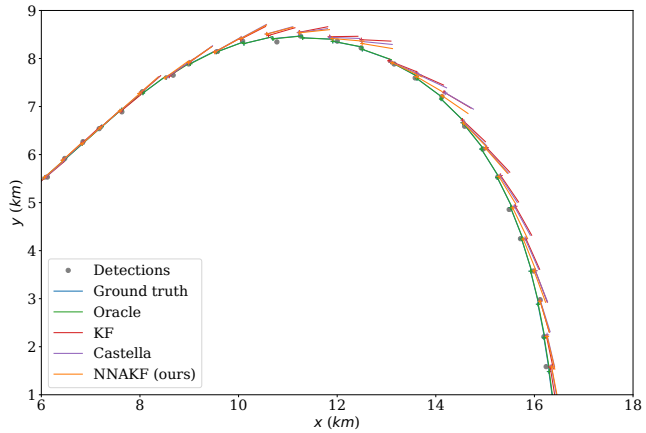

Fig. 3. Example of tracking results projected in the horizontal plane. For each filter, estimated position is represented by a cross and the estimated velocity by a speed vector. Oracle's output and the ground truth are almost indistinguishable. It seems on this particular trajectory that the NNAKF adapts quicker to the turn while it is stable during straight lines.

KF: this is the regular Kalman Filter, using a constant speed (CS) model, *i.e.*, $A = \begin{pmatrix} 0 & I \\ 0 & 0 \end{pmatrix}$.

Castella [1]: A simple adaptation of the KF with CS model is performed based on the normalized square innovations of (11), by increasing noise parameter on the corresponding speed component.

Vaidehi: this variation of the KF proposed by Vaidehi *et al.* [21] uses a NN that takes the innovation, gain and correction to produce an additional correction to the estimation provided by a KF with CS model. The original filter was made to track four targets at once, but can easily be adaptated to a mono-target problem.

LSTM-KF: this filter replaces the evolution process by a NN. It differs from the LSTM-KF proposed by Coskun [15] by only using NNs for evolution ($LSTM_F$) and process noise ($LSTM_Q$), measured state and measurement noise being provided by the dataset. This filter, originally intended to be used for human pose estimation on camera recordings, works without any prior knowledge of the target's kinematic. It is similar to the Mnemonic Kalman Filter proposed by [22].

NNAKF: this is a KF with CS model and the proposed adaptation explained in Section 2. The hyper-parameter N was empirically set to 10.

Except for the LSTM-KF, these filters all rely on a CS model as a first-order approximation of the model. All of these were trained with the Adadelta optimization algorithm and the average over time of the following loss function.

$$\mathcal{H}_n = \|X_{n|n} - \bar{Y}_n\|^2_{P_{n|n}} + ln|P_{n|n}| \tag{13}$$

The operator $\|\cdot\|_{\mathbf{\Sigma}}$ denotes the Mahalanobis distance associated with the covariance matrix Σ and $\bar{Y}_n$ is the true state including position and speed. Minimizing this loss boils down to maximizing the loglikelihood of the ground truth according to the output of the filter. The gradient is computed on the 4000 trajectories of the training dataset by batches of size 50, and the loss is monitored on the 1000 trajectories of the validation dataset. Finally, the testing dataset is used to evaluate the trained filters in the next subsection. We also made use of the following alternate filter, for comparison purposes:

Oracle: This is an extended Kalman filter with a more complex state (position, norm of velocity, heading) whose dynamical model exactly matches with the trajectory being performed. It gives an idea of the minimum error one could reach using a filter having a perfect knowledge of the flight commands at all times, which is of course impossible.

3.3. Results

All compared filters were implemented in Keras/Tensorflow as custom recurrent layers and trained end to end. The results shown in Tables 1 to 4 are average errors of the compared filters on different metrics. An example of tracked trajectory can be found in Figure 3. The different types of trajectories in the dataset are separated since error can substantially vary from one to another. The proposed filter seems to outperform the other filters most of the time. The improvement due to the Vaidehi correction looks incremental, even compared to Castella's adaptation. The poor results of the LSTM-KF can be explained by

its lack of prior knowledge of the aircraft motion, *i.e.*, CS model. Another limitation of this filter is that it requires a regular time step between measurements.

4. CONCLUSION

In this paper, a variation of a Kalman filter is proposed, using NN to adapt the process noise by the mean of an LSTM cell. By combining prior knowledge of the approximated target motion, memory of the integrated RNN, and precision of the automatic tuning by gradient-based optimization algorithms, the proposed filter with NNAKF is able to outperform classical and neural-network-aided KF based on CS model. The poor results of LSTM-KF on our problem seems to show that future work should focus on improving existing reliable models used in state-of-the-art filters, such as the IMM [8] or the IEKF [6, 7].

5. REFERENCES

[1] Frank R Castella, "An Adaptive Two-Dimensional Kalman Tracking Filter," *IEEE Transactions on Aerospace and Electronic Systems*, , no. 6, pp. 822–829, 1980.

[2] Martin Brossard, Axel Barrau, and Silvère Bonnabel, "AI-IMU Dead-Reckoning," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 585–595, 2020.

[3] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 03 1960.

[4] Arthur Gelb, *Applied Optimal Estimation*, MIT press, 1974.

[5] Axel Barrau and Silvère Bonnabel, "The Invariant Extended Kalman Filter as a Stable Observer," *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1797–1812, 2016.

[6] Marion Pilté, Silvère Bonnabel, and Frédéric Barbaresco, "Tracking the Frenet-Serret Frame Associated to a Highly Maneuvering Target in 3D," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 1969–1974.

[7] Marion Pilté, Sami Jouaber, Silvère Bonnabel, Frédéric Barbaresco, Marc Fragu, and Nicolas Honoré, "Invariant Extended Kalman Filter Applied to Tracking for Air Traffic Control," in *2019 International Radar Conference (RADAR)*. IEEE, 2019, pp. 1–6.

[8] Henk AP Blom and Yaakov 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.

[9] Diederik P Kingma and Jimmy Ba, "Adam: A Method for Stochastic Optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[10] Tijmen Tieleman and Geoffrey Hinton, "RMSprop: Divide the Gradient by a Running Average of its Recent Magnitude," *COURSERA Neural Networks Mach. Learn*, 2012.

[11] Matthew D Zeiler, "Adadelta: An Adaptive Learning Rate Method," *arXiv preprint arXiv:1212.5701*, 2012.

[12] Rahul G. Krishnan, Uri Shalit, and David Sontag, "Deep Kalman Filters," 2015.

[13] Leonard Chin, "Application of Neural Networks in Target Tracking Data Fusion," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 30, no. 1, pp. 281–287, 1994.

[14] Tuomas Haarnoja, Anurag Ajay, Sergey Levine, and Pieter Abbeel, "Backprop KF: Learning Discriminative Deterministic State Estimators," in *Advances in Neural Information Processing Systems*, 2016, pp. 4376–4384.

[15] Huseyin Coskun, Felix Achilles, Robert DiPietro, Nassir Navab, and Federico Tombari, "Long Short-Term Memory Kalman Filters: Recurrent Neural Estimators for Pose Regularization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5524–5532.

[16] Sepp Hochreiter and Jürgen Schmidhuber, "Long Short-Term Memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[17] Ce Ju, Zheng Wang, Cheng Long, Xiaoyu Zhang, Gao Cong, and Dong Eui Chang, "Interaction-Aware Kalman Neural Networks for Trajectory Prediction," *arXiv preprint arXiv:1902.10928*, 2019.

[18] Jean Mercat, Thomas Gilles, Nicole El Zoghby, Guillaume Sandou, Dominique Beauvois, and Guillermo Pita Gil, "Multi-Modal Simultaneous Forecasting of Vehicle Position Sequences using Social Attention," *arXiv preprint arXiv:1910.03650*, 2019.

[19] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the Difficulty of Training Recurrent Neural Networks," in *International conference on machine learning*, 2013, pp. 1310–1318.

[20] Pieter Abbeel, Adam Coates, Michael Montemerlo, Andrew Y Ng, and Sebastian Thrun, "Discriminative Training of Kalman Filters.," in *Robotics: Science and systems*, 2005, vol. 2, p. 1.

[21] V Vaidehi, N Chitra, M Chokkalingam, and CN Krishnan, "Neural Network Aided Kalman Filtering for Multitarget Tracking Applications," *Computers & Electrical Engineering*, vol. 27, no. 2, pp. 217–228, 2001.

[22] Steffen Jung, Isabel Schlangen, and Alexander Charlish, "A Mnemonic Kalman Filter for Non-Linear Systems With Extensive Temporal Dependencies," *IEEE Signal Processing Letters*, vol. 27, pp. 1005–1009, 2020.