KALMAN FILTER IS ALL YOU NEED: OPTIMIZATION WORKS WHEN NOISE ESTIMATION FAILS

Anonymous authors
Paper under double-blind review

ABSTRACT

Determining the noise parameters of a Kalman Filter (KF) has been studied for decades. A huge body of research focuses on the task of noise estimation under various conditions, since precise noise estimation is considered equivalent to minimization of the filtering errors. However, we show that even a small violation of the KF assumptions can significantly modify the *effective* noise, breaking the equivalence between the tasks and making noise estimation an inferior strategy. We show that such violations are common, and are often not trivial to handle or even notice. Consequentially, we argue that a robust solution is needed – rather than choosing a dedicated model per problem. To that end, we apply gradient-based optimization to the filtering errors directly, with relation to an efficient parameterization of the symmetric and positive-definite parameters of the KF. In a variety of state-estimation and tracking problems, we show that the optimization improves both the accuracy of the KF and its robustness to design decisions. In addition, we demonstrate how an optimized neural network model can seem to reduce the errors significantly compared to a KF – and how this reduction vanishes once the KF is optimized similarly. This indicates how complicated models can be wrongly identified as superior to the KF, while in fact they were merely more optimized.

1 INTRODUCTION

The Kalman Filter (KF) (Kalman, 1960) is a celebrated method for linear filtering and prediction, with applications in many fields including tracking, guidance, navigation and control (Zarchan and Musoff, 2000; Kirubarajan, 2002). Due to its simplicity and robustness, it remains highly popular – with over 10,000 citations in the last 5 years alone (Google Scholar, 2021) – despite the rise of many non-linear sequential prediction models (e.g., recurrent neural networks). The KF relies on the following model for a dynamic system:

$$\begin{aligned} X_{t+1} &= FX_t + \omega_t \qquad (\omega_t \sim N(0, Q)) \\ Z_t &= HX_t + \nu_t \qquad (\nu_t \sim N(0, R)) \end{aligned} \tag{1}$$

where X_t is the state of the system at time t (whose estimation is usually the goal), and its dynamics are modeled by the linear operator F up to the random noise ω_t with covariance Q; and Z_t is the observation, which is modeled by the linear operator H up to the noise ν_t with covariance R.

To use the KF, one must determine the noise parameters Q, R. The filtering errors (i.e., estimation errors of the states $\{X_t\}$) are minimized when Q and R correspond to the true covariance matrices of the noise (Humpherys et al., 2012). Thus, these parameters are usually determined by noise estimation. In absence of system state data $\{x_t\}$ (the "ground truth"), many methods have been suggested to determine Q and R from observations data $\{z_t\}$ alone (Abbeel et al., 2005; Odelson et al., 2006; Zanni et al., 2017; Park et al., 2019). When ground-truth data *is* available, however, noise estimation trivially reduces to calculation of the sample covariance matrices (Lacey, 1998):

$$\hat{R} \coloneqq Cov(\{z_t - Hx_t\}_t), \qquad \hat{Q} \coloneqq Cov(\{x_{t+1} - Fx_t\}_t). \tag{2}$$

Indeed, as stated by Odelson et al. (2006), "*the more systematic and preferable approach to determine the filter gain is to estimate the covariances from data*". Our work focuses on such problems with ground-truth available for learning (but not for inference after the learning, of course), which was motivated by a real-world Doppler radar estimation problem.

Noise estimation is often not optimal: The equivalence between noise estimation and errors minimization can be proved under the standard KF assumptions – that is, known and linear dynamics and observation models (F, H), with i.i.d and normally-distributed noises $(\{\omega_t\}, \{\nu_t\})$ (Humpherys et al., 2012). However, as put by Thomson (1994), "*experience with real-world data soon convinces one that stationarity and Gaussianity are fairy tales invented for the amusement of undergraduates*" – and linearity and independence can be safely added to this list. Therefore, under realistic assumptions, the covariance of the noise does not necessarily correspond to optimal filtering.

We introduce a case study in the context of radar tracking, where we demonstrate that even using the true covariance of the noise ("oracle" noise-estimation) is sub-optimal in a variety of scenarios – including very simple scenarios with relatively minor violation of the KF assumptions. In Appendices E and F, we also analyze this phenomenon analytically for two private cases (non-linearity in Doppler radar and non-i.i.d noise in lidar), where the violation of a single KF assumption is shown to modify the effective noise. By providing this extensive evidence for the sub-optimality of noise estimation in practical applications of the KF, **we re-open a problem that was considered solved for decades** (Kalman, 1960).

We also show that seemingly small changes in the properties of the scenario may lead to major changes in the desired design of the KF, e.g., whether to use a KF or an Extended KF (Sorenson, 1985). In certain cases, the design choices are easy to overlook (e.g., Cartesian vs. polar coordinates), and are not trivial to make even if noticed. As a result, it is impractical to manually choose or develop a variant of the KF for every problem. Rather, we should assume that our model is sub-optimal, and leverage data to deal with the sub-optimality as robustly as possible.

Optimization *is* optimal: We consider Q and R as model parameters that should be optimized with respect to the filtering errors (i.e., system-state estimation errors) – rather than estimating the noise. While both noise estimation and errors optimization rely on exploitation of data, only the latter explicitly addresses the actual goal of solving the filtering problem.

Gradient-based optimization methods are usually effective in the field of machine learning, but applying them naively to the entries of Q and R may violate the symmetry and positive-definiteness (SPD) constraints of the covariance matrices. Indeed, even works that come as far as optimizing Q and R (instead of estimating the noise) usually apply limited optimization methods, e.g., grid-search (Coskun et al., 2017) or diagonal restriction of the covariance matrices (Formentin and Bittanti, 2014; Li et al., 2019). To address this issue, we use a parameterization based on Cholesky decomposition (Horn and Johnson, 1985), which allows us to apply gradient-based optimization to SPD matrices. This method is computationally efficient compared to other general gradient-based methods for SPD optimization (Tsuda et al., 2005; Tibshirani, 2015).

We demonstrate that the optimization reduces the errors of the KF consistently: over different variants of the KF, over different violations of KF assumptions, over different domains (tracking from radar, video or lidar), over small and large training datasets, and even under distributional shifts between train and test datasets. Furthermore, we show that optimization improves the robustness to design decisions, by shrinking the performance gaps between different variants of the KF.

As explained above, we extensively justify the need to optimize the KF in many practical problems, and suggest a simple solution which is effective, robust, computationally efficient, and relies on standard tools in supervised machine learning. As a result, we believe that in the scope of filtering problems with available ground-truth, whenever the KF assumptions are not strictly-guaranteed, **the suggested optimization method should become the new standard procedure for the KF tuning**.

Unfair comparison: Many learning algorithms have been suggested to address non-linearity in filtering problems, e.g., based on Recurrent Neural Networks (RNN). Such works often use a linear tool such as the KF as a baseline for comparison – with tuning parameters being sometimes ignored (Gao et al., 2019), sometimes based on noise estimation (fa Dai et al., 2020), and sometimes optimized in a limited manner using trial-and-error (Jamil et al., 2020) or grid-search (Coskun et al., 2017). Our findings imply that such a methodology yields over-optimistic conclusions, since the baseline is not optimized to the same level as the learning model. This may result in adoption of over-complicated algorithms with no actual added value. Instead, any learning algorithm should be compared to a baseline that is optimized using a similar method (e.g., gradient-descent with respect to the errors).

Indeed, we consider an extension of the KF based on LSTM, which is the key component in many SOTA algorithms for non-linear sequential prediction in recent years (Neu et al., 2021). For radar tracking with non-linear motion, we demonstrate how the LSTM seems to provide a significant improvement over the KF. Then, we show that the whole improvement comes from optimization of parameters, and *not* from the expressive non-linear architecture. In particular, this result demonstrates the competitiveness of our suggested method versus SOTA sequential prediction models.

Recent works in the area of machine learning have already shown that advanced algorithms often obtain most of their improvement from implementation nuances (Engstrom et al., 2019; Andrychowicz et al., 2020; Henderson et al., 2017). Our work continues this line of thinking and raises awareness to this issue in the domain of filtering problems.

Contribution: We show that the KF is used sub-optimally in a variety of problems; demonstrate that advanced filtering algorithms are often tested unfairly; suggest a simple method to solve both issues, using gradient-based optimization along with Cholesky decomposition; and provide a detailed case study which analyzes the differences between noise estimation and parameters optimization empirically and (for certain private cases) analytically.

Limitations: The main assumption of this work is availability of the ground-truth system-states in the training data, which allows supervised learning. Ground-truth data is often available from simulations, controlled experiments or manual labeling. Within this scope, we show that although noise estimation is straight-forward (Equation 2), it is often not the right task to address.

The paper is organized as follows: Section 2 reviews the KF. Section 3 introduces our method for efficient KF optimization. Section 4 justifies the necessity of KF optimization through a detailed case study. Section 5 presents a neural version of the KF which reduces the errors compared to a standard KF – but not when compared to an optimized KF. Section 6 discusses related works.

2 Preliminaries: the Kalman Filter Algorithm

The KF algorithm (Kalman, 1960; Humpherys et al., 2012) relies on Equation 1 for a dynamic system model. It keeps an estimate of the state X_t, represented as the mean x_t and covariance P_t of a normal distribution. As shown in Figure 1, it alternately predicts the next state using the dynamics model (*prediction* step), and processes new information from incoming observations (*update* or *filtering* step).

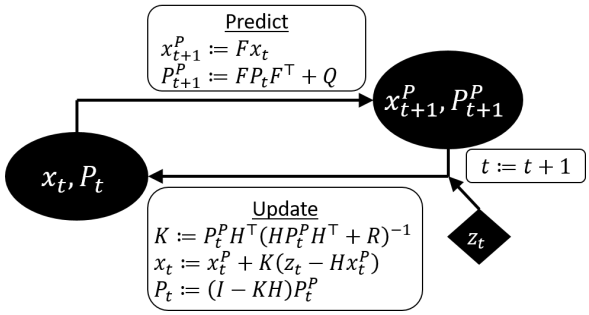


Figure 1: The KF algorithm. The prediction step is based on the motion model F with noise Q, whereas the update step is based on the observation model H with noise R.

The KF yields optimal state estimations – but only under a restrictive set of assumptions (Kalman, 1960), as specified in Definition 2.1. Note that normality of the noise is excluded since it is not necessary for optimality (Humpherys et al., 2012), although it is also often assumed.

Definition 2.1 (KF assumptions)**.** F, H of Equation 1 are constant and known matrices (i.e., linear models of motion and observation); ω_t, ν_t are i.i.d random variables with zero-mean and constant, known covariance matrices Q, R, respectively; and the initial state distribution is known.

Certain assumptions violating in Definition 2.1 can be partially handled by variations of the KF, such as the Extended KF (EKF) (Sorenson, 1985) which replaces the linear models F, H with local linear approximations, and the Unscented KF (UKF) (Wan and Van Der Merwe, 2000) which applies the filtering through sigma-points sampled from the estimated distribution. The use of multiple tracking models alternately is also possible using switching mechanisms (Mazor et al., 1998).

While F and H are usually determined based on domain knowledge, Q and R are often estimated from data as the covariance of the noise. As mentioned in Section 1, this can be done using Equation 2

if ground-truth data is available, or using more sophisticated methods otherwise (Odelson et al., 2006; Feng et al., 2014; Park et al., 2019).

See Appendix A for a detailed introduction of the KF and recurrent neural networks (RNN, LSTM).

3 KF Optimization Using Cholesky Parameterization

The performance of a KF strongly depends on its parameters Q and R (Formentin and Bittanti, 2014). These parameters are usually regarded as estimators of the noise covariance in motion and observation, respectively (Lacey, 1998), and are estimated accordingly. Although optimization has been suggested for the KF in the past (Abbeel et al., 2005), it was often viewed as a fallback solution (Odelson et al., 2006), for cases where direct estimation is not possible (e.g., the true states are unavailable in the data (Feng et al., 2014)). Accordingly, we define our baseline method for this work:

Method 1 (Estimated KF)**.** KF whose parameters Q, R were determined from data using Equation 2.

The preference of noise estimation relies on the fact that the KF – with parameters corresponding to the noise covariances – minimizes the square filtering errors (MSE) of the estimates of the system-states $\{X_t\}$. This holds under Assumptions 2.1 (Humpherys et al., 2012). Hence, equivalently, we could explicitly look for the parameters that minimize the MSE, e.g., using the Adam algorithm (Diederik P. Kingma, 2014). Adam is a popular variant of the well-known gradient-descent algorithm, and has achieved remarkable results in many optimization problems in the field of machine learning in recent years, including in non-convex problems where local-minima exist (Zhong et al., 2020).

Numeric optimization is more complicated than computing two covariance matrices, but is shown to be beneficial in Sections 4 and 5. Stable optimization algorithms are available in several open-source packages, e.g., PyTorch (Paszke et al., 2019), which supports gradient-propagation through matrix-inversion (as needed for the KF computations). As demonstrated in Sections 4,5, the PyTorch implementation of Adam provides stable results on MSE-optimization of a KF. This is conceivable, as the KF is arguably a simple model in terms of mathematical sophistication and number of parameters.

One major challenge in the KF parameters optimization is that both Q and R correspond to covariance matrices, which are symmetric and positive definite (SPD): a naive numeric optimization of their entries may ruin the SPD structure. This difficulty often motivates optimization methods that avoid gradients (Abbeel et al., 2005), or even the restriction of Q and R to be diagonal (Li et al., 2019). Indeed, Formentin and Bittanti (2014) pointed out that "*since both the covariance matrices must be constrained to be positive semi-definite, Q and R are often parameterized as diagonal matrices*".

To allow Adam to optimize the non-diagonal Q and R we use the Cholesky decomposition (Horn and Johnson, 1985), which states that any SPD matrix $A \in \mathbb{R}^{n\times n}$ can be written as $A = LL^\top$, where L is lower-triangular with positive entries along its diagonal. The reversed claim is also true: for any lower-triangular L with positive diagonal, $LL^\top$ is SPD. Now consider the parameterization of A using the following $\binom{n}{2} + n = \frac{n(n+1)}{2}$ parameters: $\binom{n}{2}$ parameters correspond to $\{L_{ij}\}_{1\le j<i\le n}$, and n parameters to $\{\log L_{ii}\}_{1\le i\le n}$. Clearly, the transformation from the parameters to the SPD matrix $A = LL^\top$ is differentiable and outputs a SPD matrix for any realization of the parameters in $\mathbb{R}^{n(n+1)/2}$. Thus, we can apply Adam (or any other gradient-based method) to optimize these parameters without worrying about the SPD constraints. This approach, which is presented more formally in Appendix C and implemented in our code, was used for all the optimizations in this work.

Cholesky decomposition for covariance matrix parameterization was already suggested by Pinheiro and Bates (1996). However, despite its simplicity, it is not commonly used for gradient-based optimization of SPD matrices in general, and to the best of our knowledge is not used at all for KF optimization in particular. Other methods exist for SPD optimization, e.g., matrix-exponent (Tsuda et al., 2005) and projected gradient-descent with respect to the SPD cone (Tibshirani, 2015). These methods require SVD-decomposition in every iteration, hence are computationally heavy, which may explain why they are not commonly used for KF tuning. The parameterization derived from Cholesky decomposition only requires a single matrix multiplication, and thus is both efficient and easy to implement. This results in our method for KF optimization:

Method 2 (Optimized KF)**.** A KF whose parameters Q, R were determined from data by optimizing the MSE of the state-estimates, using Adam algorithm with relation to a Cholesky parameterization of Q and R, as described in detail in Appendix B.1.

4 KALMAN FILTER CONFIGURATION AND OPTIMIZATION: A CASE STUDY

In this section, we introduce a detailed case study to compare noise estimation and errors optimization in the KF. As mentioned in Section 2, the two are equivalent only under Assumptions 2.1 (Humpherys et al., 2012). Some of these assumptions are clearly violated in realistic scenarios, while other violations may be less obvious. For example, even if a radar's noise is i.i.d in polar coordinates, it is not so in Cartesian coordinates (see Appendix A.4).

The need to rely on many assumptions might explain why there are several extensions and design decisions in a KF configuration. This includes the choice between KF and EKF; the choice between educated state initialization and a simple uniform prior (Linderoth et al., 2011); and certain choices that may be made without even noticing, such as the coordinates of the state representation.

The case study below justifies the following claims:

1. Design decisions in a KF are often nontrivial to make and are potentially significant.
2. Tuning a KF by noise estimation is often highly sub-optimal – even in very simple scenarios.
3. Tuning a KF by optimization improves both accuracy and robustness to design decisions.
4. KF optimization using Method 2 is robust to distributional shifts (Appendix G) and to small training datasets (Appendix H).

These claims imply that the popular KF algorithm may not be exploited to its full potential. Furthermore, many works that compare learning algorithms to a KF baseline conduct an "unfair" comparison, as the learning algorithms are optimized and the KF is not. This may lead to adoption of unnecessarily complicated algorithms, as demonstrated in Section 5. Indeed, in many works the tuning of the baseline KF is either ignored in the report (Gao et al., 2019) or relies on estimation (or knowledge) of the noise (fa Dai et al., 2020; Jamil et al., 2020), as demonstrated extensively in Section 6.

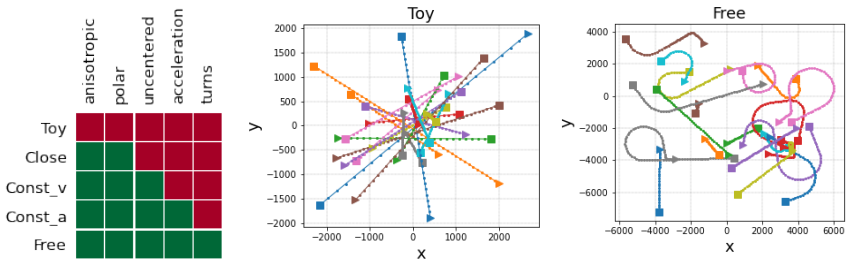
4.1 SETUP AND METHODOLOGY

In the case study of **radar tracking**, each target is represented by a sequence of (unknown) states in consecutive time-steps, and a corresponding sequence of radar measurements. A state $x_{full} = (x_x, x_y, x_z, x_{vx}, x_{vy}, x_{vz})^\top \in \mathbb{R}^6$ consists of 3D location and velocity. We also denote $x = (x_x, x_y, x_z)^\top \in \mathbb{R}^3$ for the location alone (or $x_{\text{target},t} \in \mathbb{R}^3$ for the location of a certain target at time t). An observation $z \in \mathbb{R}^4$ consists of noisy measurements of range, azimuth, elevation and Doppler signal. Note that the former three correspond to a noisy measurement of x in polar coordinates, and the latter one measures the projection of velocity onto the radial direction x. The goal is to minimize the error of the point-estimate of the state: $MSE = \sum_{\text{target}} \sum_t (\tilde{x}_{\text{target},t} - x_{\text{target},t})^2$.

The case study considers 5 types of tracking scenarios (*benchmarks*) and 4 variants of the KF (*baselines*). For each benchmark and each baseline, we use the benchmark training data to produce one estimated **KF** (Method 1) and one optimized KF (**OKF**, Method 2). We then evaluate the errors of both models on the test data of the benchmark (generated using different seeds than the training data). All experiments were run on eight i9-10900X CPU cores on a single Ubuntu machine.

Figures 2b,2c display a sample of trajectories in the simplest benchmark (*Toy*), which satisfies all KF assumptions except for a linear observation model H; and in the complex *Free Motion* benchmark, which violates several assumptions. The other benchmarks are demonstrated in Appendix L. Figure 2a defines each of the 5 benchmarks more formally as a certain subset of the following properties:

- *anisotropic*: horizontal motion is more likely than vertical motion (otherwise motion direction is distributed uniformly).
- *polar*: radar noise is generated i.i.d in polar coordinates (otherwise noise is Cartesian i.i.d, which violates the physics of the system).
- *uncentered*: targets are not forced to concentrate close to the radar.
- *acceleration*: speed change is allowed (through intervals of constant acceleration).
- *turns*: non-straight motion is allowed.



(a) (b) (c)

Figure 2: (a) Benchmarks names (rows) and the properties that define them (columns). Green means that the benchmark satisfies the property. (b,c) Targets in Toy and Free Motion benchmarks (projected onto XY plane).

The 4 baselines differ from each other by using either KF or EKF, with either Cartesian or polar coordinates for representation of R (the rest of the system is always represented in Cartesian coordinates). As mentioned above, each baseline is tuned once by Method 1 and once by Method 2. For the Toy benchmark, the optimal parameters are also derived **analytically** in Appendix E. Appendix G demonstrates the robustness of OKF to distributional shifts by training on one benchmark and testing on another. Appendix H repeats the tests for different sizes of training datasets.

We also repeat the experiment for the problem of **tracking from video**, using MOT20 dataset (Dendorfer et al., 2020) of real-world pedestrians, with train and test datasets taken from separated videos. For this problem we only consider a KF with Cartesian coordinates, since there is no polar component in the problem. See Appendix I for the detailed setup of the video tracking experiments. In addition, we test the OKF in the problem of **self-driving state-estimation from lidar measurements** (Appendix J). These two domains extend the scope of the experimented KF assumptions violations, since they correspond to linear observation models (unlike the radar benchmarks), with both noisy (lidar) and noiseless (MOT20) observations. For a simplified lidar problem, we also derive the optimal parameters **analytically** in Appendix F.

4.2 Results

Design decisions are not trivial: Table 1 summarizes the tracking errors. The left column in each cell corresponds to Method 1 (standard KF), and shows that in each benchmark, the errors strongly depend on the design decisions (R's coordinates and whether to use EKF). In the Toy benchmark, for example, EKF is the best design, since the observation model H is non-linear.

In other benchmarks, however, the winning designs of the non-optimized KF are arguably surprising:

1. Under non-isotropic motion direction (all benchmarks except Toy), EKF is worse than KF despite the non-linear motion. It is possible that since the horizontal prior reduces the stochasticity of H, the advantage of EKF no longer justifies the instability of the derivative-based approximation.

Table 1: Summary of the errors of the various models over the various benchmarks (on out-of-sample test data). Corresponding confidence intervals are available in Figure 14a in the appendix. In the model names, "O" denotes optimized, "E" denotes extended, and "p" denotes polar (e.g., OEKFp is an *extended* KF with *polar* representation of R and *optimized* parameters). For KFp, we also consider an oracle-realization of R according to the true noise of the simulated radar in polar coordinates (available only in polar benchmarks). Note that (1) for any benchmark and any baseline, optimization yields lower errors than estimation; and (2) this remains true even in the oracle variant, where the noise "estimation" is perfect.

Benchmark	KF	OKF	KFp	KFp (oracle)	OKFp	EKF	OEKF	EKFp	OEKFp
Toy	151.7	84.2	269.6	–	116.4	92.8	**79.4**	123.0	109.1
Close	25.0	24.8	22.6	22.5	**22.5**	26.4	26.1	24.5	24.1
Const_v	90.2	90.0	102.3	102.3	**89.2**	102.5	99.7	112.7	102.1
Const_a	107.5	101.6	118.4	118.3	**100.3**	110.0	107.0	126.0	108.7
Free	125.9	118.8	145.6	139.3	**117.9**	135.8	121.9	149.3	120.0

2. Even when the observation noise is polar i.i.d, polar representation of R is not beneficial unless the targets are forced to concentrate close to the radar (last 3 benchmarks in Table 1). It is possible that when the targets are distant, Cartesian coordinates have a more important role in expressing the horizontal prior of the motion.

Since the best variant of the KF for each benchmark seems difficult to predict in advance, a practical system cannot rely on choosing the KF variant optimally, and rather has to be robust to this choice.

Optimization is more accurate and robust: Table 1 shows that for *every* benchmark and *every* baseline in the **radar tracking** problem (20 experiments in total), optimization yielded smaller errors than noise estimation (over an out-of-sample test dataset). Note that OKF wins even in the Toy scenario, under the slightest violation of KF assumptions. In addition, the variance between the baselines reduces under optimization, i.e., the optimization makes the KF more robust to design decisions (which is also evident in Figure 14a in Appendix L).

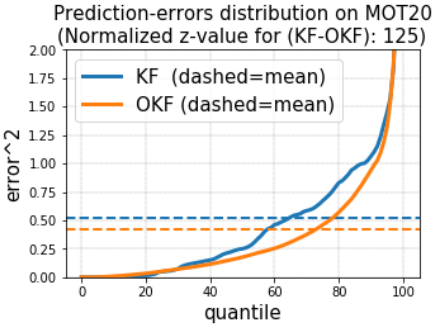
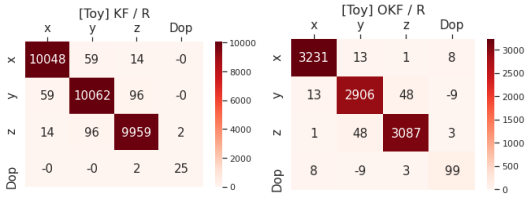


Figure 3: Prediction errors of KF and OKF on 1208 targets of the test data of MOT20 videos dataset. The MSE of OKF is smaller by 18%, with statistical significance of p-value $< 10^{-6}$ over the test samples.

We also studied the performance of a KF with a perfect knowledge of the noise covariance matrix R. Note that in the constant-speed benchmarks, the estimation of $Q = 0$ is already very accurate, hence in these benchmarks the oracle-KF has a practically perfect knowledge of both noise covariances. Nonetheless, Table 1 shows that the oracle yields very similar results to a KF with estimated parameters. This indicates that the limitation of noise estimation in the KF indeed comes from choosing the wrong goal, and not from estimation inaccuracy.

Figure 3 shows that OKF provides significantly better predictions also in **video tracking**, where the MSE is reduced by 18% (see Appendix I for more details). In addition, in **lidar-based state estimation**, Appendix J shows a similar error reduction of 15%, and Lemma F.3 explains the MSE-advantage of OKF **analytically**.



(a) Estimated R (b) Optimized R

Figure 4: The covariance matrix R of the observation noise obtained in a (Cartesian) KF by noise estimation and by optimization, based on the dataset of the Toy benchmark. The axes correspond to the observation variables associated with the matrix entries. Note that the noise estimation is quite accurate, as the true variance of the noise is 100^2 for the positional dimensions and 5^2 for Doppler. The optimization increases the variance associated with the Doppler signal, as predicted by Lemma E.1. The decrease in the other diagonal components is discussed in Appendix E.

OKF is further demonstrated to be robust to major **distributional shifts** in Appendix G, and to **small training datasets** in Appendix H.

Diagnosis of the KF sub-optimality in Toy scenario: The source of the gap between estimated and optimized noise parameters can be studied through the simplest Toy benchmark, where the only violation of KF assumptions is the non-linear observation model H. Since the non-constant entries of H correspond to the Doppler observation, the non-linearity inserts uncertainty to the Doppler observation (in addition to the inherent observation noise). This increases Doppler's effective noise in comparison to the location observation, as shown **analytically** by Lemma E.1 in the appendix. This explanation is consistent with Figure 4: the noise associated with Doppler is indeed increased by the optimization. Note that the non-linearity modifies the effective noise in a delicate way, which would *not* be compensated by a naive trial and error of noise inflation or deflation.

5 Neural Kalman Filter: Is Non-linear Prediction Helpful?

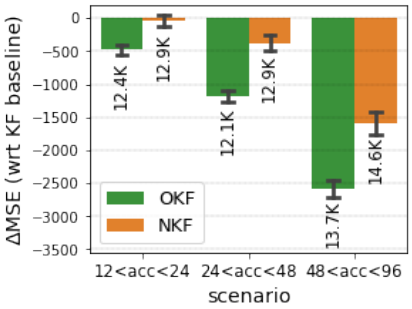
A standard Kalman Filter for a tracking task assumes linear motion, as discussed in Section 2. In this section we introduce the ***Neural Kalman Filter*** tracking model (NKF), which uses LSTM networks to model non-linear motion, while keeping the framework of the KF – namely, the probabilistic representation of the target's state, and the separation between prediction and update steps.

Every prediction step, NKF uses a neural network model to predict the target's acceleration a_t and the motion uncertainty Q_t. Every update step, another network predicts the observation uncertainty R_t. The predicted quantities are incorporated into the standard update equations of the KF, as shown in detail in Figure 7 in Appendix B. For example, the prediction step of NKF is:

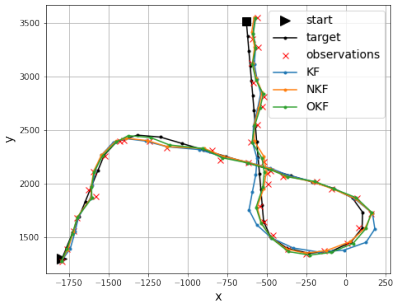
$$x^P_{t+1} = Fx_t + 0.5a_t(\Delta t)^2, \qquad P^P_{t+1} = FP_tF^\top + Q_t$$

where $F = F(\Delta t)$ is the constant-velocity motion operator, and a_t, Q_t are predicted by the LSTM (whose input includes the recent observation and the current estimated target's state). Other predictive features were also attempted but failed to provide significant value, as described in Appendix B. Note that the neural network predicts the acceleration rather than directly predicting the location. This is intended to regularize the predictive model and to exploit the domain knowledge of kinematics.

After training NKF over a dataset of simulated targets with random turns and accelerations (see Appendix B.1 for more details), we tested it on a test dataset. The test dataset is similar to the training dataset, with different seeds and an extended range of permitted accelerations. As shown in Figure 5, NKF significantly reduces the tracking errors compared to a standard KF.



(a)



(b)

Figure 5: (a) Relative tracking errors (lower is better) with relation to a standard KF, over targets with different ranges of acceleration. The error-bars represent confidence intervals of 95%. The label of each bar represents the corresponding *absolute* MSE ($\times 10^3$). In the training data, the acceleration was limited to 24-48, hence the other ranges measure generalization. While the Neural KF (NKF) is significantly better than the standard KF, its advantage is entirely eliminated once we optimize the KF (OKF). (b) A sample target and the corresponding models outputs (projected onto XY plane). The standard KF has a difficulty to handle some of the turns.

At this point, it seems that the non-linear architecture of NKF provides better accuracy in the non-linear problem of radar tracking. However, Figure 5 shows that by shifting the baseline from a naive KF to an optimized one, we *completely eliminate* the advantage of NKF, and in fact reduce the errors even further. In other words, in this experiment the benefits of NKF come *only* from optimization and *not at all* from the expressive architecture. By overlooking the sub-optimality of noise estimation in the KF, we would wrongly adopt the over-complicated NKF.

Note that the optimized KF (OKF) also generalizes well to targets with different accelerations than observed in the training, which indicates a certain robustness to distributional shifts. Appendix K extends the experiment to additional variants of NKF, to another tracking benchmark, and to the evaluation metric of likelihood (NLL) in addition to estimation error (MSE). Note that high likelihood score is important for the matching task in a multi-target tracking problem.

Of course, our results do not imply that neural-networks in general cannot be superior to a KF: only that when comparing the two, if the KF is not optimized similarly to the neural model, the experimental results may be very misleading. As discussed in Section 6, this wrong *methodology* is not uncommon in the literature.

6 Related Work

Noise estimation: When tuning a KF, the system-states are often unavailable in the data (Formentin and Bittanti, 2014). Thus, estimation of the KF noise parameters from observations alone has been studied for decades, addressed using various methods such as EM (Shumway and Stoffer, 2005) and others (Odelson et al., 2006; Feng et al., 2014; Park et al., 2019). However, if the states *are* available, noise estimation reduces to Equation 2 and is considered a solved problem (Odelson et al., 2006). In this case, we show that although noise estimation is easy, it is often not the right task to address.

Many works addressed the problem of non-stationary noise estimation (Zanni et al., 2017; Akhlaghi et al., 2017). However, as demonstrated in Sections 4,5, in certain cases stationary methods are highly competitive if tuned correctly – even in problems with complicated dynamics.

Optimization: In this work we apply gradient-based optimization to the KF with respect to its errors. Optimization without gradients computation was already suggested in Abbeel et al. (2005). In practice, "optimization" of the KF is often handled manually using trial-and-error (Jamil et al., 2020) or a grid-search over possible values of Q and R (Formentin and Bittanti, 2014; Coskun et al., 2017). In other cases, Q and R are restricted to be diagonal (Li et al., 2019; Formentin and Bittanti, 2014).

Gradient-based optimization of SPD matrices in general was suggested in Tsuda et al. (2005) using matrix-exponents, and is also possible using projected gradient-descent (Tibshirani, 2015) – both rely on SVD-decomposition. In this work, we apply gradient-based optimization using the parameterization that was suggested in Pinheiro and Bates (1996), which requires a mere matrix multiplication, and thus is both efficient and easy to implement.

Neural Networks (NNs) in filtering problems: Section 5 presents a RNN-based extension of the KF, and demonstrates how its advantage over the linear KF vanishes once the KF is optimized. The use of NNs for non-linear filtering problems is very common in the literature, e.g., in online tracking prediction (Gao et al., 2019; Dan Iter, 2016; Coskun et al., 2017; fa Dai et al., 2020; Ullah et al., 2019), near-online prediction (Kim et al., 2018), and offline prediction (Liu et al., 2019b). In addition, while Bewley et al. (2016) apply a KF for video tracking from mere object detections, Wojke et al. (2017) add to the same system a NN that generates visual features as well. NNs were also considered for related problems such as data association (Liu et al., 2019a), model-switching (Deng et al., 2020), and sensors fusion (Sengupta et al., 2019).

In many works that consider NNs for filtering problems, a KF is used as a baseline for comparison. However, while the NN parameters are typically optimized with respect to the filtering errors, the KF parameters tuning is sometimes ignored (Gao et al., 2019; Bai et al., 2020; Zheng et al., 2019), sometimes based on estimation (or knowledge) of the noise (fa Dai et al., 2020; Aydogmus and Aydogmus, 2015; Revach et al., 2021), and sometimes optimized in a limited manner as mentioned above (Jamil et al., 2020; Coskun et al., 2017; Ullah et al., 2019). Our findings imply that this methodology is wrong, since the baseline is not optimized to the same level as the learning model. Hussein (2014) explicitly discusses the sensitivity of EKF performance to the noise model accuracy, and suggests the solution of a NN with supervised learning – without considering the same supervised learning for the EKF.

7 Summary

Through a detailed case study, we demonstrated both analytically and empirically the fragility of the KF assumptions, and how the slightest violation of them may change the effective noise in the problem – leading to significant and non-trivial changes in the optimal noise parameters. We addressed this problem using optimization tools from supervised machine learning, and suggested how to apply them efficiently to the SPD parameters of the KF.

We demonstrated the accuracy of our method over different violations of KF assumptions, in different domains (radar tracking, video tracking and lidar-based state estimation), with relation to different variants of the KF, over small and large training datasets, and even under distributional shifts between train and test datasets. Indeed, once we acknowledged the need for optimization and applied the Cholesky parameterization, the optimization itself performed robustly over all these scenarios. In light of this evidence, we recommend to use our method as the default procedure for the KF tuning in presence of ground-truth data, whenever the KF assumptions are not strictly-guaranteed.

We also demonstrated one of the consequences of using a sub-optimal KF: the common methodology of comparing learning filtering algorithms to classic variants of the KF is misleading, as it essentially compares an optimized model to a non-optimized one. We argued that the baseline method should be optimized similarly to the researched one, e.g., using optimization rather than noise estimation.

REPRODUCIBILITY

All the experiments in this work are reproducible using our code, including data generation, models training and results analysis. The complete proofs for the theoretical results are available in Appendices E and F.

REFERENCES

Pieter Abbeel, Adam Coates, Michael Montemerlo, Andrew Ng, and Sebastian Thrun. Discriminative training of kalman filters. *Robotics: Science and systems*, pages 289–296, 06 2005. doi: 10.15607/RSS.2005.I.038.

S. Akhlaghi, N. Zhou, and Z. Huang. Adaptive adjustment of noise covariance in kalman filter for dynamic state estimation. In *2017 IEEE Power Energy Society General Meeting*, pages 1–5, 2017. doi: 10.1109/PESGM.2017.8273755.

Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters in on-policy reinforcement learning? a large-scale empirical study, 2020.

Zafer Aydogmus and Omur Aydogmus. A comparison of artificial neural network and extended kalman filter based sensorless speed estimation. *Measurement*, 63:152–158, 2015. ISSN 0263-2241. doi: https://doi.org/10.1016/j.measurement.2014.12.010. URL https://www.sciencedirect.com/science/article/pii/S0263224114006071.

Yu-ting Bai, Xiao-yi Wang, Xue-bo Jin, Zhi-yao Zhao, and Bai-hai Zhang. A neuron-based kalman filter with nonlinear autoregressive model. *Sensors*, 20(1):299, 2020.

S. T. Barratt and S. P. Boyd. Fitting a kalman smoother to data. In *2020 American Control Conference (ACC)*, pages 1526–1531, 2020. doi: 10.23919/ACC45564.2020.9147485.

Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016. doi: 10.1109/ICIP.2016.7533003.

S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House Radar Library, Boston, 1999.

W. R. Blanding, P. K. Willett, and Y. Bar-Shalom. Multiple target tracking using maximum likelihood probabilistic data association. In *2007 IEEE Aerospace Conference*, pages 1–12, 2007. doi: 10.1109/AERO.2007.353035.

Chaw-Bing Chang and Keh-Ping Dunn. Radar tracking using state estimation and association: Estimation and association in a multiple radar system, 04 2019.

Zhaozhong Chen et al. Kalman filter tuning with bayesian optimization, 2019.

S. Kumar Chenna, Yogesh Kr. Jain, Himanshu Kapoor, Raju S. Bapi, N. Yadaiah, Atul Negi, V. Seshagiri Rao, and B. L. Deekshatulu. State estimation and tracking problems: A comparison between kalman filter and recurrent neural networks. *ICONIP*, 2004.

Huseyin Coskun, Felix Achilles, Robert DiPietro, Nassir Navab, and Federico Tombari. Long short-term memory kalman filters:recurrent neural estimators for pose regularization. *ICCV*, 2017. URL https://github.com/Seleucia/lstmkf_ICCV2017.

Philip Zhuang Dan Iter, Jonathan Kuck. Target tracking with kalman filtering, knn and lstms, 2016. URL http://cs229.stanford.edu/proj2016/report/IterKuckZhuang-TargetTrackingwithKalmanFilteringKNNandLSTMs-report.pdf.

JP DeCruyenaere and HM Hafez. A comparison between kalman filters and recurrent neural networks. *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, 4:247–251, 1992.

Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. Mot20: A benchmark for multi object tracking in crowded scenes, 2020. URL https://motchallenge.net/data/MOT20/.

Lichuan Deng, Da Li, and Ruifang Li. Improved IMM Algorithm based on RNNs. *Journal of Physics Conference Series*, 1518:012055, April 2020. doi: 10.1088/1742-6596/1518/1/012055.

Jimmy Ba Diederik P. Kingma. Adam: A method for stochastic optimization, 2014. URL https://arxiv.org/abs/1412.6980.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep policy gradients: A case study on ppo and trpo. *ICLR*, 2019. URL https://openreview.net/pdf?id=r1etN1rtPB.

Hai fa Dai, Hong wei Bian, Rong ying Wang, and Heng Ma. An ins/gnss integrated navigation in gnss denied environment using recurrent neural network. *Defence Technology*, 16(2):334–340, 2020. ISSN 2214-9147. doi: https://doi.org/10.1016/j.dt.2019.08.011. URL https://www.sciencedirect.com/science/article/pii/S2214914719303058.

B. Feng, M. Fu, H. Ma, Y. Xia, and B. Wang. Kalman filter with recursive covariance estimation—sequentially estimating process noise covariance. *IEEE Transactions on Industrial Electronics*, 61(11):6253–6263, 2014. doi: 10.1109/TIE.2014.2301756.

Simone Formentin and Sergio Bittanti. An insight into noise covariance estimation for kalman filter design. *IFAC Proceedings Volumes*, 47(3):2358–2363, 2014. ISSN 1474-6670. doi: https://doi.org/10.3182/20140824-6-ZA-1003.01611. URL https://www.sciencedirect.com/science/article/pii/S1474667016419646. 19th IFAC World Congress.

C. Gao, H. Liu, S. Zhou, H. Su, B. Chen, J. Yan, and K. Yin. Maneuvering target tracking with recurrent neural networks for radar application. *2018 International Conference on Radar (RADAR)*, pages 1–5, 2018.

Chang Gao, Junkun Yan, Shenghua Zhou, Bo Chen, and Hongwei Liu. Long short-term memory-based recurrent neural networks for nonlinear target tracking. *Signal Processing*, 164, 05 2019. doi: 10.1016/j.sigpro.2019.05.027.

Matthieu Geist and Olivier Pietquin. Kalman filtering colored noises: the (autoregressive) moving-average case. In *MLASA 2011*, pages 1–4, Honolulu, United States, December 2011. URL https://hal-supelec.archives-ouvertes.fr/hal-00660607.

Google Scholar. Citations since 2017: "a new approach to linear filtering and prediction problems", 2021. URL https://scholar.google.com/scholar?as_ylo=2017&hl=en&as_sdt=2005&sciodt=0,5&cites=5225957811069312144&scipsc=.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep Reinforcement Learning that Matters. *arXiv preprint arXiv:1709.06560*, 2017. URL https://arxiv.org/pdf/1709.06560.pdf.

Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 1997. URL https://direct.mit.edu/neco/article/9/8/1735/6109/Long-Short-Term-Memory.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

Jeffrey Humpherys, Preston Redd, and Jeremy West. A fresh look at the kalman filter. *SIAM Review*, 54(4):801–823, 2012. doi: 10.1137/100799666.

Ala A. Hussein. Kalman filters versus neural networks in battery state-of-charge estimation: A comparative study. *International Journal of Modern Nonlinear Theory and Application*, 2014.

Faisal Jamil et al. Toward accurate position estimation using learning to prediction algorithm in indoor navigation. *Sensors*, 2020.

S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004. doi: 10.1109/JPROC.2003.823141.

R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960. ISSN 0021-9223. doi: 10.1115/1.3662552. URL https://doi.org/10.1115/1.3662552.

Chanho Kim, Fuxin Li, and James M. Rehg. Multi-object tracking with neural gating using bilinear lstm. *ECCV*, September 2018.

Yaakov Bar-Shalom X.-Rong Li Thiagalingam Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory, Algorithms and Software*. John Wiley and Sons, Inc., 2002. doi: 10.1002/0471221279.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Tony Lacey. Tutorial: The kalman filter, 1998. URL "http://web.mit.edu/kirtley/kirtley/binlustuff/literature/control/Kalmanfilter.pdf".

S. Li, C. De Wagter, and G. C. H. E. de Croon. Unsupervised tuning of filter parameters without ground-truth applied to aerial robots. *IEEE Robotics and Automation Letters*, 4(4):4102–4107, 2019. doi: 10.1109/LRA.2019.2930480.

M. Linderoth, K. Soltesz, A. Robertsson, and R. Johansson. Initialization of the kalman filter without assumptions on the initial state. In *2011 IEEE International Conference on Robotics and Automation*, pages 4992–4997, 2011. doi: 10.1109/ICRA.2011.5979684.

Hu Liu et al. Kalman filtering attention for user behavior modeling in ctr prediction. *NeurIPS*, 2020.

Huajun Liu, Hui Zhang, and Christoph Mertz. Deepda: Lstm-based deep data association network for multi-targets tracking in clutter. *CoRR*, abs/1907.09915, 2019a. URL http://arxiv.org/abs/1907.09915.

Jingxian Liu, Zulin Wang, and Mai Xu. Deepmtt: A deep learning maneuvering target-tracking algorithm based on bidirectional lstm network. *Information Fusion*, 53, 06 2019b. doi: 10.1016/j.inffus.2019.06.012.

E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan. Interacting multiple model methods in target tracking: a survey. *IEEE Transactions on Aerospace and Electronic Systems*, 34(1):103–123, 1998.

A. Paulo Moreira, Paulo Costa, and José Lima. New approach for beacons based mobile robot localization using kalman filters. *Procedia Manufacturing*, 51:512–519, 2020. URL https://www.sciencedirect.com/science/article/pii/S2351978920319296. 30th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM2021).

Dominic A. Neu, Johannes Lahann, and Peter Fettke. A systematic literature review on state-of-the-art deep learning methods for process prediction. *CoRR*, abs/2101.09320, 2021. URL https://arxiv.org/abs/2101.09320.

Brian Odelson, Alexander Lutz, and James Rawlings. The autocovariance least-squares method for estimating covariances: Application to model-based control of chemical reactors. *Control Systems Technology, IEEE Transactions on*, 14:532 – 540, 06 2006. doi: 10.1109/TCST.2005.860519.

Sebin Park et al. Measurement noise recommendation for efficient kalman filtering over a large amount of sensor data. *Sensors*, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Dong-liang Peng and Yu Gu. Imm algorithm for a 3d high maneuvering target tracking. *International Conference in Swarm Intelligence*, pages 529–536, 06 2011. doi: 10.1007/978-3-642-21524-7_65.

José C. Pinheiro and Douglas M. Bates. Unconstrained parameterizations for variance-covariance matrices. *Statistics and Computing*, 6:289–296, 1996.

Guy Revach, Nir Shlezinger, Xiaoyong Ni, Adria Lopez Escoriza, Ruud J. G. van Sloun, and Yonina C. Eldar. Kalmannet: Neural network aided kalman filtering for partially known dynamics, 2021.

B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech house Boston, 2004.

David E. Rumelhart et al. Learning representations by back-propagating errors. *Nature*, 1986. URL https://www.nature.com/articles/323533a0.

A. Sengupta, F. Jin, and S. Cao. A dnn-lstm based target tracking approach using mmwave radar and camera sensor fusion. *2019 IEEE National Aerospace and Electronics Conference (NAECON)*, pages 688–693, 2019. doi: 10.1109/NAECON46414.2019.9058168.

Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications (Springer Texts in Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2005. ISBN 0387989501.

Harold Wayne Sorenson. *Kalman Filtering: Theory and Application*. IEEE Press, 1985.

D. J. Thomson. Jackknifing multiple-window spectra. In *Proceedings of ICASSP '94. IEEE International Conference on Acoustics, Speech and Signal Processing*, volume vi, pages VI/73–VI/76 vol.6, 1994. doi: 10.1109/ICASSP.1994.389899.

Ryan Tibshirani. Proximal gradient descent and acceleration, 2015. URL https://www.stat.cmu.edu/~ryantibs/convexopt-F15/lectures/08-prox-grad.pdf.

Koji Tsuda, Gunnar Rätsch, and Manfred K. Warmuth. Matrix exponentiated gradient updates for on-line learning and bregman projection. *Journal of Machine Learning Research*, 6(34):995–1018, 2005. URL http://jmlr.org/papers/v6/tsuda05a.html.

Israr Ullah, Muhammad Fayaz, and DoHyeun Kim. Improving accuracy of the kalman filter algorithm in dynamic conditions using ann-based learning module. *Symmetry*, 11(1), 2019. ISSN 2073-8994. doi: 10.3390/sym11010094. URL https://www.mdpi.com/2073-8994/11/1/94.

Ashish Vaswani et al. Attention is all you need. *NeurIPS*, 2017.

E. Wan. Sigma-point filters: An overview with applications to integrated navigation and vision assisted control. *IEEE*, pages 201–202, 2006. doi: 10.1109/NSSPW.2006.4378854.

E. A. Wan and R. Van Der Merwe. The unscented kalman filter for nonlinear estimation. *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, pages 153–158, 2000. doi: 10.1109/ASSPCC.2000.882463.

Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649, 2017. doi: 10.1109/ICIP.2017.8296962.

L. Zanni, J. Le Boudec, R. Cherkaoui, and M. Paolone. A prediction-error covariance estimator for adaptive kalman filtering in step-varying processes: Application to power-system state estimation. *IEEE Transactions on Control Systems Technology*, 25(5):1683–1697, 2017. doi: 10.1109/TCST.2016.2628716.

Paul Zarchan and Howard Musoff. *Fundamentals of Kalman Filtering: A Practical Approach*. American Institute of Aeronautics and Astronautics, 2000.

Tianyu Zheng, Yu Yao, Fenghua He, and Xinran Zhang. An rnn-based learnable extended kalman filter design and application. In *2019 18th European Control Conference (ECC)*, pages 3304–3309, 2019. doi: 10.23919/ECC.2019.8796088.

Hui Zhong, Zaiyi Chen, Chuan Qin, Zai Huang, Vincent W. Zheng, Tong Xu, and Enhong Chen. Adam revisited: a weighted past gradients perspective. *Frontiers of Computer Science*, 2020.

CONTENTS

1 Introduction **1**

2 Preliminaries: the Kalman Filter Algorithm **3**

3 KF Optimization Using Cholesky Parameterization **4**

4 Kalman Filter Configuration and Optimization: a Case Study **5**

5 Neural Kalman Filter: Is Non-linear Prediction Helpful? **7**

6 Related Work **8**

7 Summary **9**

A Preliminaries: Extended **15**

B Problem Setup and Implementation Details **17**

C Cholesky Gradient Optimization **20**

D Related Work: Extended **20**

E Theoretical Analysis: Estimation vs. Optimization in the Toy Benchmark **21**

F Theoretical Analysis: Estimation vs. Optimization in the Presence of Non-i.i.d Noise **23**

G Generalization Tests: Training and Testing on Different Benchmarks **26**

H Sensitivity of KF-Optimization to Train Dataset Size **27**

I Optimized Kalman Filter for Video Tracking **28**

J Optimized Kalman Filter for Lidar-based State Estimation in Self Driving **29**

K Neural Kalman Filter: Extended Experiments **30**

L Detailed Results **31**

A PRELIMINARIES: EXTENDED

A.1 MULTI-TARGET RADAR TRACKING

In the problem of multi-target radar tracking, noisy observations (or *detections*) of aerial objects (or *targets*) are received by the radar and are used to keep track of the objects. In standard radar sensors, the signal of an observation includes the target range and direction (which can be converted into location coordinates x, y, z), as well as the Doppler shift (which can be converted into the projection of velocity v_x, v_y, v_z onto the location vector).

The problem consists of observations-trackers assignment, where each detection of a target should be assigned to the tracker representing the same target; and of location estimation, where the location of each target is estimated at any point of time according to the observations (Chang and Dunn, 2019). In the fully online setup of this problem, the assignment of observations is done once they are received, before knowing the future observations on the targets (though observations of different targets may be received simultaneously); and the location estimation at any time must depend only on previously received observations.

Assignment problem: The assignment problem can be seen as a one-to-one matching problem in the bipartite graph of existing trackers and newly-received observations, where the edge between a tracker and an observation represents "how likely the observation z_j is for the tracker trk_i". In particular, if the negative-log-likelihood is used for edges weights, then the total cost of the match represents the total likelihood (under the assumption of observations independence):

$$C(\text{match}) = \sum_{(i,j)\in\text{match}} -\log P(z_j|trk_i) = -\log \prod_{(i,j)\in\text{match}} P(z_j|trk_i) \tag{3}$$

The assignment problem can be solved using standard polynomial-time algorithms such as the Hungarian algorithm (Kuhn, 1955). However, the assignment can only be as good as the likelihood information fed to the algorithm. This is a major motivation for trackers that maintain probabilistic representation of the target, rather than a merely point estimate of the location. A common example for such a tracking mechanism is the Kalman filter discussed below.

A.2 KALMAN FILTER

Kalman Filter (KF) is a widely-used method for linear filtering and prediction originated in the 1960s (Kalman, 1960), with applications in many fields (Zarchan and Musoff, 2000) including object tracking (Kirubarajan, 2002). The classic model keeps an estimate of the monitored system state (e.g., location and velocity), represented as the mean x and covariance P of a normal distribution (which uniquely determine the PDF of the distribution). The mechanism (see Figure 1) alternately applies a *prediction step*, where a linear model $x := Fx$ predicts the next state; and an *update step* (also termed *filtering step*), where the information of a new observation z is incorporated into the estimation (after a transformation H from the observation space to the state space).

Note that KF compactly keeps our knowledge about the monitored target at any point of time, which allows us to estimate both whether a new observation corresponds to the currently monitored target, and what the state of the system currently is. However, KF strongly relies on several assumptions:

1. **Linearity**: both the state-transition of the target f and the state-observation transformation h are assumed to be linear, namely, $f(x) = F \cdot x$ and $h(x) = H \cdot x$. Note that the Extended KF described below, while not assuming linearity, still assumes known models for transition and observation.
2. **Normality**: both state-transition and state-observation are assumed to have a Gaussian noise with covariance matrices Q and R respectively. As a result, the estimates of the state x and its uncertainty P also correspond to the mean and covariance matrix of a Normal distribution representing the information regarding the target location.
3. **Known model**: F, Q, H, R are assumed to be known.

While F and H are usually determined manually according to domain knowledge, the noise model parameters R, Q are often estimated from data (as long as the the true states are available in the data). Specifically, they are often estimated from the sample covariance matrix of the noise: $R \coloneqq Cov(Z - HX), Q \coloneqq Cov(\Delta X) = Cov(\{x_{t+1} - Fx_t\}_t)$ (Lacey, 1998).

Two non-linear extensions of KF – *Extended Kalman Filter* (EKF) (Sorenson, 1985) and *Unscented Kalman Filter* (UKF) (Wan and Van Der Merwe, 2000) – are also very popular in problems of non-linear state estimation and navigation (Wan, 2006). EKF replaces the linear prediction (F) and observation (H) models with non-linear known models f and h, and essentially runs a standard KF with the local linear approximations $F = \frac{\partial f}{\partial x}|_{x_t}, H = \frac{\partial h}{\partial x}|_{x_t}$, updating in every step t. UKF does not pass the state distribution x, P through the motion equations as a whole distribution, since such distribution transformation is unknown for general nonlinear state-transition. Instead, it samples *sigma points* from the original Gaussian distribution, passes each point through the nonlinear transition, and uses the resulting points to estimate the new Gaussian distribution. *Particle filters* go farther and do not bother to carry a Gaussian PDF: instead, the points themselves (*particles*) can be seen as representatives of the distribution.

Whether linear or not, a single simple model is hardly enough to represent the motion of any aerial target. A common approach is to maintain a switching model that repeatedly chooses between several mode, each controlled by a different motion model. Common simple models the Constant Velocity (CV), Constant Acceleration (CA), and Coordinated Turn Left or Right (CTL and CLR). Note that in all these models the prediction operator F is linear. A popular switching mechanism is the *Interactive Multiple Model* (IMM) (Mazor et al., 1998), which relies on knowing the transition-probabilities between the various modes.

In addition to the challenge of predicting mode-transitions, the standard models in use are often too simple to represent the motions of modern and highly-maneuvering targets, such as certain unmanned aerial vehicles, drones and missiles. Many approaches have been recently attempted to improve tracking of such objects, as discussed in Section D.

A.3 Recurrent Neural Networks

Neural networks (NN) are parametric functions, usually constructed as a sequence of matrix-multiplications with some non-linear differentiable transformation between them. NNs are known to be able to approximate complicated functions, given that the right parameters are chosen. Optimization of the parameters of NNs is a field of vast research for decades, and usually relies on gradient-based methods, that is, calculating the errors of the NN with relation to some training-data of inputs and their desired outputs, deriving the errors with respect to the network's parameters, and moving the parameters against the direction of the gradient.

Recurrent neural networks (RNN) (Rumelhart et al., 1986) are NN that are intended to be iteratively fed with sequential data samples, and that pass information (the *hidden state*) over iterations. Every iteration, the hidden state is fed to the next copy of the network as part of its input, along with the new data sample. *Long Short Term Memory* (LSTM) (Hochreiter and Schmidhuber, 1997) is an architecture of RNN that is particularly popular due to the linear flow of the hidden state over iterations, which allows to capture memory for relatively long term. The parameters of a RNN are usually optimized in a supervised manner with respect to a training dataset of input-output pairs.

A.4 Noise Dependence by Change of Coordinates

One of the violations of the KF assumptions in Section 4 is the non-i.i.d nature of the noise. The interest in this violation is increased by the fact that the noise is simulated in an i.i.d manner, which makes the violation non-trivial to notice.

The violation is caused by the difference of coordinates: while the state of the target is represented in Cartesian coordinates, the noise is generated in polar ones. As a result, polar dimensions with higher noise are assigned to the same Cartesian dimensions in consecutive time-steps, creating dependence between the noise in these time-steps in Cartesian coordinates.

For example, consider a radar with a noiseless angular estimation (i.e., all the noise is radial), and consider a low target (in the same height of the radar). Clearly, most of the noise will be in the XY

plane – both in the current time-steps and in the following ones (until the target gets far from the plane). In other words, the noise in the following time-steps statistically-depends on the current time-step, and is not i.i.d.

B PROBLEM SETUP AND IMPLEMENTATION DETAILS

Code: The code for this work is available here, and naturally contains implementation details that may be missing in the descriptions below.

Targets simulation: Our simulated dataset consists of episodes of tens to hundreds of seconds. Each episode contains around 20 targets. Each target appears at a certain point of time, has certain acceleration and speed ranges, and alternately performs straight motions (in constant speed or with acceleration) and turns (horizontally or vertically) until it disappears. The time progresses in discrete time-steps of size dt. The space is assumed to be 3D and homogeneous (no ground).

The randomization in an episode is expressed through the number of targets, each target's acceleration and speed ranges, target appearing time, initial target state, number of turns, directions of turns (left/right/up/down), length of turns, length of straight intervals, whether accelerating in straight intervals and what the acceleration is.

Radar simulation: Every time-step, the radar generates detections of the available targets. Each detection contains location information (x, y, z) and a Doppler measurement corresponding to the projection of velocity onto the location vector. Each detection has an additive i.i.d Gaussian noise in polar coordinates, i.e., range, azimuth, elevation, and Doppler. The radar itself is assumed to be a singular point located in the origin $(0, 0, 0)$. Recall that we assume a homogeneous space with no ground.

Tracking test framework: The predictive models (*trackers*) can be tested in a mere prediction framework (where each target is tested by itself), or in a full tracking framework (where multiple targets are tracked simultaneously). In the latter, we use a tracking system (*solver*) consisting of a matcher and multiple trackers. Every time-step during a test episode, the solver runs the following procedures:

1. **Observe**: receive observations from the radar.
2. **Assign**: use the matcher (based on the standard Hungarian Algorithm (Kuhn, 1955), with relation to the corresponding trackers' likelihood functions) to assign observations to trackers; create a new tracker for any unassigned observation; and delete any tracker whose target was not observed for several time-steps.
3. **Update**: update the location estimation of the trackers using the newly-assigned observations.
4. **Predict**: predict the target location in the next time-step (to allow more accurate assignment of the next batch of observations).

In the scope of this work we are mainly interested in the tracker module. Note that the tracker affects the matching accuracy through the likelihoods it feeds to the matcher; and affects the estimation accuracy directly in the prediction and update operations. To solve both matching and estimation tasks, the tracker is required to provide a good probabilistic model of the target location (namely, a PDF of the location) in addition to a point estimate.

We measure the estimation accuracy using the standard Mean Square Error (MSE) of the target location after the update step. For matching accuracy, for each target we consider the tracker that covers most of the target's observations, and count the percent of observations assigned to that tracker.

Trackers: To support the procedures described above, a tracker must implement prediction and update steps, as well as exporting the current point estimate of the target location and the likelihood of any observation whenever queried.

All the trackers in this work are conceptually based on Kalman Filter (described in Section A.2). The following elements characterize the exact nature of the tracker:

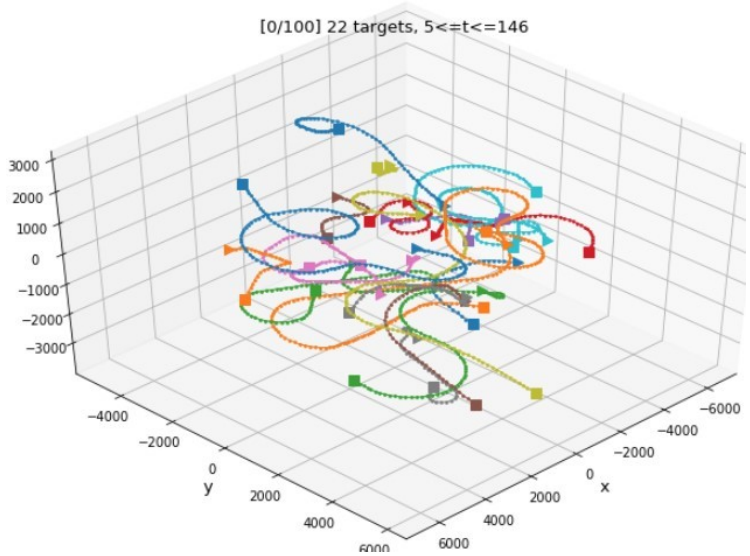


Figure 6: A sample of targets trajectories.

1. **KF/EKF**: the difference is in the observation model H, as explained in Section A.2.
2. **Coordinates for representation of the observation noise** R: while we represent the state in Cartesian coordinates ($(x, y, z, vx, vy, vz)^\top$) and measure the errors accordingly, the natural coordinates for the noise of the radar are polar, as the observation errors are simulated independently for range, azimuth, elevation and Doppler. In particular, in Cartesian coordinates the errors are not independent, hence R is not inherently constant. When implementing R in polar coordinates, we have to transform it to Cartesian coordinates every time-step according to the recent observation.
3. **Determining the noise parameters** Q, R **by estimation/optimization**: by estimation of the noise we mean $Q = Cov(X_{t+1} - FX_t), R = Cov(Z_t - HX_t)$ over the training data $\{X_t, Z_t\}$. By optimization we mean that $R \in \mathbb{R}^{4\times4}, Q \in \mathbb{R}^{6\times6}$ are intended to minimize the loss of the predictions over the training data. Note that in the optimization, we represent R, Q using lower-triangular decomposition so that both matrices remain SPD.
4. **Acceleration prediction**: replace the linear motion model Fx with $Fx + 0.5a \cdot dt^2$, where a is predicted using a LSTM that receives the observations and estimated states as recurrent inputs.
5. Q **prediction**: dynamically predict the motion model noise Q using an LSTM. We use the same LSTM as for a, except for the head layer that outputs Q itself. For simplicity, we assume that Q is diagonal when dynamically predicted.
6. R **prediction**: dynamically predict the observation noise R using a neural network that receives the recent observations and estimated states as inputs. For simplicity, we assume that R is diagonal when dynamically predicted.
7. H **prediction**: the observation model $z = h(x)$ of a radar is a non-linear function of x. Extended KF handles this through a linear approximation of h around the currently-estimated state x. By H-prediction, we aim to learn to choose H dynamically in an optimal way to improve the tracking errors.
8. **Mixture-of-Gaussians representation**: KF assumes Gaussian errors (in prediction and observation), hence the distribution of the state is always normal, which is somewhat limiting. We try to represent the distribution as a mixture of Gaussians (i.e., a linear combination of a few Gaussians). The Gaussians are intended to correspond to possible modes of motion (e.g., constant speed vs. a turn), and thus can predict the dynamic acceleration (see above) differently from each other. In addition, the amplitude of each Gaussian (i.e., the probability assigned to it) is predicted every step along with the accelerations.

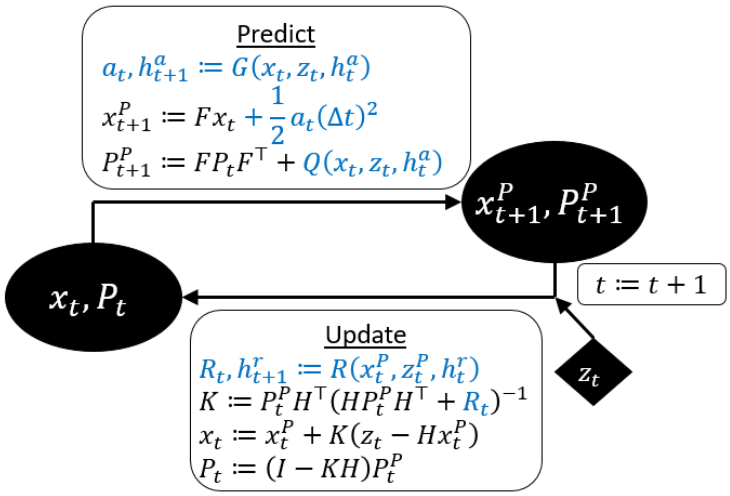


Figure 7: The Neural Kalman Filter (NKF) algorithm of Section 5. The differences from Figure 1 are marked. G, Q, R are neural networks of the type LSTM, and h^a, h^r are their hidden states (more accurately, G and Q are different heads of the same network, and thus share the same hidden states h^a). In addition to the raw x_t, z_t, the networks are also fed with certain manually-crafted features (e.g., the projection of $z_t - x_t$ on the perpendicular direction $x_t^\perp$).

Optimization of the noise parameters (R, Q) was implemented as part of the Optimized Kalman Filter (OKF) described in Section 4. This feature was found to make the tracker both more accurate and more robust to design choices such as KF vs. EKF or coordinates of R as described above. A neural-network-based Dynamic prediction of acceleration and noise was implemented as part of the Neural Kalman Filter (NKF) described in Section 5. The NKF architecture is also described in detail in Figure 7.

The last two features in the list failed to provide successful results: dynamic prediction of H often yielded unstable predictions, and a mixture representation had insignificant effects on the prediction. Furthermore, beyond 2 Gaussians in the mixture, it seemed that any additional Gaussian became practically inactive, with amplitude being constantly near zero.

B.1 Training

Training data: A training dataset consists of targets, each consisting of a sequence of states (e.g., locations and velocities) along with a sequence of observations. In particular, the target state ground-truth is assumed to be known and available to learn from. This can be achieved either by a simulated environment or by internal targets sensors (e.g., GPS) in a controlled experiment in a physical environment. Note that the training dataset considers the targets independently of each other, and does not refer to the notion of episodes with possibly-simultaneous targets. Targets assignment in such episodes can be handled by the Hungarian algorithm as mentioned above. The test dataset differs from the training dataset in the seeds used to generate the trajectories and the observations, and in certain experiments also in some configuration parameters such as the permitted range of acceleration.

Noise estimation: For models whose parameters are determined by noise estimation (e.g., the standard KF), the training reduces to calculation of R and Q from the data as the sample covariance matrices (Equation 2). Note that $\{x_t\}$ and $\{z_t\}$ from Equation 2 correspond to the states and observations mentioned above.

Optimization (error minimization): For models whose parameters are determined by optimization (e.g., OKF and NKF), we use the Adam algorithm (Diederik P. Kingma, 2014), as implemented by the PyTorch package, with initial learning rate of 0.01 (reduced by 50% every 150 training steps)

and no weight-decay. Every training epoch goes once over each training target. In the beginning of each epoch, batches of 10 targets are randomly drawn.

Every training step corresponds to a single batch, where the training loss is aggregated additively over the targets in the batch and over the time-steps of each target. That is, for each target in the batch, we run the tracker on the target observations; after each observation receive the corresponding state estimate from the tracker; and add the corresponding error to the loss.

The explicit training loss is the Mean Square Error (MSE) of the estimated location (which is part of the estimated state) after the update step.

In the case study experiments on the Doppler radar domain (Section 4), each benchmark included 1500 simulated targets for training and 1000 targets for testing. A single training epoch was sufficient for convergence of the optimized models. The optimization lasted a few minutes per benchmark (scenario) – for all the 4 baselines in parallel (OKF, OKFp, OEKF, OEKFp).

C Cholesky Gradient Optimization

Section 3 discusses an efficient method to apply gradient-based optimization to SPD matrices, based on parameterization of the SPD matrix using Cholesky decomposition (Pinheiro and Bates, 1996). This section presents this method more formally.

We consider an objective function $f(x; A)$ to be minimized, where x is the input and $A \in \mathbb{R}^{n\times n}$ are parameters formed as a SPD matrix. We define $A(L) \coloneqq LL^\top$ and

$$(L(\theta))_{ij} \coloneqq \begin{cases} 0 & \text{if } i < j \\ e^{\theta_{n(n-1)/2+i}} & \text{if } i = j \\ \theta_{(i-2)(i-1)/2+j} & \text{if } i > j \end{cases}$$

where $\theta \in \mathbb{R}^{n(n+1)/2}$. We also denote the optimization update rule by $u(\theta, g)$, where θ are the current parameters and g is the current gradient. For example, for the classic gradient descent, $u(\theta, g) = \theta - \alpha g$. Using these notations, the optimization method becomes as simple as Algorithm 1. This algorithm (with Adam's update rule) was used for all the KF optimizations in this work.

Algorithm 1: Cholesky Gradient Optimization

Input: objective f; update rule u; data X; initial parameters $\theta_0 \in \mathbb{R}^{n(n+1)/2}$;
Output: SPD matrix $A \in \mathbb{R}^{n\times n}$;
$\theta \leftarrow \theta_0$;
for *training epoch* **do**
  for *data sample x in X* **do**
    $g \leftarrow \nabla_\theta f(x; A(L(\theta)))$;
    $\theta \leftarrow u(\theta, g)$;
Return $A(L(\theta))$;

Note that the diagonal of $L(\theta)$ could be replaced by any increasing positive transformation. In fact, we could even drop the positive transformation and allow non-positive diagonal entries in L: this would only eliminate the uniqueness of the parameterization, and would allow A to be positive *semi*-definite.

D Related Work: Extended

Kalman Filter parameters tuning: Estimation of the noise parameters in KF has been researched for decades. A popular framework for this problem is estimation from data of observations $\{z\}$ alone (Odelson et al., 2006; Feng et al., 2014; Park et al., 2019), since the ground-truth of the states $\{x\}$ is often unavailable in the data (Formentin and Bittanti, 2014). In this work we assume that the ground-truth is available.

Many works addressed the problem of non-stationary noise estimation (Zanni et al., 2017; Akhlaghi et al., 2017). However, as demonstrated in Sections 4,5, in certain cases stationary methods are highly competitive if tuned correctly – even in problems with complicated dynamics.

Optimization of KF parameters with respect to tracking errors was already suggested in Abbeel et al. (2005), using a method that avoids gradients computation. In practice, "optimization" of KF parameters with respect to the errors is often handled manually using trial-and-error (Jamil et al., 2020), or using a grid-search over possible values of Q and R (Formentin and Bittanti, 2014; Coskun et al., 2017). Alternatively, the optimization is often simplified by restricting Q and R to be diagonal (e.g., see the experiments in Barratt and Boyd (2020) and Li et al. (2019)). Formentin and Bittanti (2014) pointed out explicitly that "since both the covariance matrices must be constrained to be positive semi-definite, Q and R are often parameterized as diagonal matrices".

Gradient-based optimization of SPD matrices in general was suggested in Tsuda et al. (2005) using matrix-exponents, and is also possible using projected gradient-descent (Tibshirani, 2015) – both rely on SVD-decomposition. In this work, we apply gradient-based optimization using the parameterization that was suggested in Pinheiro and Bates (1996), which requires a mere matrix multiplication, and thus is both efficient and easy to implement.

Neural networks for tracking: While comparison between RNN and KF goes back to DeCruyenaere and Hafez (1992); Chenna et al. (2004), the contribution of neural networks to non-linear motion tracking has been widely researched mostly in recent years, in the context of both radar and other sensors. Liu et al. (2019b) focus on an offline framework, where a bidirectional LSTM receives a full, normalized trajectory as an input, and improves the target locations estimations of a UKF in a 2-dimensional space. The LSTM is trained to predict the residual errors of the UKF. Kim et al. (2018) focus on a "near-online" framework, where assignment of an observation is decided after looking ahead a certain number of observations. They operate in the visual domain, incorporate both visual and motion information together to make decisions, and suggest a new variant of LSTM, where the hidden state h interacts with the new input x through the multiplication hx rather than linear combination of the two, and thus can be interpreted as a dynamic model rather than hidden memory.

Gao et al. (2018) focus on an online framework, similarly to our work, where assignments and estimations are done immediately with the receipt of the observations. For the private case of a one-dimensional recurrent input signal, they use two LSTMs for estimation of the target location – one for the prediction step and one for the update step – to achieve improved estimation. Gao et al. (2019) expand this work and learn to predict both the location estimate and its (one-dimensional) variance according to the negative-log-likelihood loss, which yields similar accuracy but provides uncertainty estimation. Dan Iter (2016) consider online tracking of single vehicles on the 2-dimensional video screen of a vehicle camera (the KITTI object tracking benchmark). Since they assume zero observation error, they focus on prediction error optimization. Their LSTM yields a better point-prediction than KF, but does not improve the end-to-end tracking, which they hypothesize that could be caused by insufficient uncertainty estimation. Coskun et al. (2017) suggest a combination of LSTM and KF for the task of human pose estimation, learned on basis of the square estimation-errors. In particular, three LSTM modules are used to predict the parameters F, Q, R of a KF.

Liu et al. (2019a) train an LSTM for data association, i.e., the assignment task. Sengupta et al. (2019) address the problem of sensors fusion using a deep neural network. They incorporate information from a monocular camera and a mmWave radar, and show how to handle failures in either of the sensors. Deng et al. (2020) consider standard models as building blocks for IMM, and replace the mode-transition matrix with an LSTM that estimates the transition probabilities dynamically. Peng and Gu (2011) replace the standard building blocks of IMM with modes that are dedicatedly-designed to track highly-maneuvering targets.

A combination of KF with attention models (Vaswani et al., 2017) was also suggested (Liu et al., 2020) in the context of click-through rate prediction in e-commerce, where vital information for prediction may be concentrated in certain time-steps on which the model should focus.

E Theoretical Analysis: Estimation vs. Optimization in the Toy Benchmark

Section 4 demonstrates that the optimal noise parameters in KF often differ from the covariance of the noise. Here we analyze this effect for the private case of the Toy benchmark, where all KF assumptions (Definition 2.1) hold – except for the non-linear observation model $H(X)$.

Definition E.1 (The toy Doppler problem)**.** The toy Doppler problem is the filtering problem modeled by Equation 1 with the system-state space $X = (x, y, z, u_x, u_y, u_z)^\top \in \mathbb{R}^6$ and the following parameters:

$$F = \begin{pmatrix} 1 & & & 1 & & \\ & 1 & & & 1 & \\ & & 1 & & & 1 \\ & & & 1 & & \\ & & & & 1 & \\ & & & & & 1 \end{pmatrix} \qquad H = H(X) = \begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & x/r & y/r & z/r \end{pmatrix}$$

$$Q = \mathbf{0} \in \mathbb{R}^{6\times 6} \qquad R = \begin{pmatrix} \sigma_x^2 & & & \\ & \sigma_y^2 & & \\ & & \sigma_z^2 & \\ & & & \sigma_D^2 \end{pmatrix}$$

where $r = \sqrt{x^2 + y^2 + z^2}$ and $\sigma_x, \sigma_y, \sigma_z, \sigma_D > 0$.

As mentioned in Section 4, this toy problem is highly simplified. For example, the targets only move in exact straight lines, and the noise of the radar is drawn i.i.d in *Cartesian* coordinates. Yet, despite the major simplifications, we show below that the remaining violation of observation linearity leads to sub-optimality of noise estimation when determining the KF parameters.

Since the true X is unknown to the KF, we assume that the KF approximates $H(X)$ using $\tilde{H}$, which relies on the current estimates of x, y, z:

$$\tilde{H} = \begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & \hat{x}/\hat{r} & \hat{y}/\hat{r} & \hat{z}/\hat{r} \end{pmatrix}$$

Lemma E.1 (Effective observation noise in the toy Doppler problem)**.** Consider the toy Doppler problem with the approximated observation model $\tilde{H}$. At a certain time-step, assume that the positional estimation errors (i.e., the estimation errors of x, y, z) are independent of the target velocity u. Then, the observation noise corresponding to $\tilde{H}$ (the *effective* observation noise) is described by the following covariance matrix:

$$\tilde{R} = \begin{pmatrix} \sigma_x^2 & & & \\ & \sigma_y^2 & & \\ & & \sigma_z^2 & \\ & & & \sigma_D^2 + C \end{pmatrix} = R + \begin{pmatrix} 0 & & & \\ & 0 & & \\ & & 0 & \\ & & & C \end{pmatrix} \tag{4}$$

for some $C > 0$.

Proof. Denoting $\tilde{x} := x/r$ (and similarly for $\tilde{y}, \tilde{z}$), we can rewrite $\tilde{H}$ as

$$\tilde{H} = \begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ & & 1 & & & \\ & & & \tilde{x} + d\tilde{x} & \tilde{y} + d\tilde{y} & \tilde{z} + d\tilde{z} \end{pmatrix}$$

where $d\tilde{x}, d\tilde{y}, d\tilde{z}$ are the corresponding (normalized) estimation errors. Note that in the private case where x, y, z are estimated by the observations z_x, z_y, z_z, these errors coincide with the corresponding entries of the observation noise ν (up to the normalization by r).

By shifting the observation model in Equation 1 from H to $\tilde{H}$, we receive

$$Z = H \cdot X + \nu = \tilde{H}X + \begin{pmatrix} \nu_x \\ \nu_y \\ \nu_z \\ \nu_D - d\tilde{x}u_x - d\tilde{y}u_y - d\tilde{z}u_z \end{pmatrix} = \tilde{H}X + \begin{pmatrix} \nu_x \\ \nu_y \\ \nu_z \\ \nu_D - d\tilde{r}^\top \cdot u \end{pmatrix}$$

that is, the observation noise corresponding to $\tilde{H}$ is $\tilde{\nu} = (\nu_x, \nu_y, \nu_z, \nu_D - d\tilde{r}^\top \cdot u)^\top$.

$d\tilde{x}, \nu_x$ are both independent of the velocity u_x (the former by the lemma's assumption, and the latter by the model of Equation 1 in Definition E.1). Thus, $Corr(d\tilde{x} \cdot u_x, \nu_x) = E(d\tilde{x} \cdot u_x \cdot \nu_x) = E(d\tilde{x}\nu_x)E(u_x) = 0$, and similarly for ν_y, ν_z. Hence, by denoting $C = Var(d\tilde{r}^\top \cdot u) > 0$, the covariance matrix of $\tilde{\nu}$ is indeed $\tilde{R}$. □

Lemma E.1 provides the theoretical explanation for Figure 4b in Section 4, where we see that the optimization of R increases the variance of the Doppler signal compared to the variance of the positional signal. Figure 4b also shows a decrease in the positional variance (not only an increase in Doppler's variance), which is not explained by the analysis above. This is caused by the fact that when $Q \equiv 0$, the absolute values of R have quite minor importance compared to the relative values between the components – hence the optimization increases Doppler's variance compared to positional variance, but is quite indifferent to their scale. Indeed, re-scaling of the optimized R provided indistinguishable empirical results.

Note that in this private case of Toy benchmark, the gap between estimation and optimization could be easily solved by modifying the estimation from $R \coloneqq Cov(\{z_t - H(x_t) \cdot x_t\})$ to $R \coloneqq Cov(\{z_t - H(z_t) \cdot x_t\})$. However, this correction does not guarantee optimal results beyond the Toy benchmark, and in particular did not demonstrate significant improvement in the other benchmarks defined in Section 4. Indeed, while certain private cases can be modeled and solved analytically through effective noise estimation, optimization of the parameters provides a more robust solution.

F Theoretical Analysis: Estimation vs. Optimization in the Presence of Non-i.i.d Noise

The KF model assumes i.i.d noise for both the motion model and the observation model. In cases where the noise is known to follow another specific model, a dedicated solution can sometimes be developed. For example, if the noise is auto-regressive with a known order p, an adjusted KF model may consider the last p values of the noise itself as part of the system state (Geist and Pietquin, 2011). However, we often do not know the actual model of the noise, or we do not know how to develop the corresponding analytical solution, or we are not aware at all of the violation of the i.i.d assumption. In any of these cases, the default practical choice is to keep the original KF model. However, the covariance matrices of the noise are not guaranteed anymore to be the optimal values of the KF parameters Q and R.

For example, consider the problem of a moving target in a two-dimensional space, with observation noise drawn i.i.d in polar coordinates (similarly to the lidar model of Appendix J). For simplicity of the analysis we represent the state using only the location $x = (x_1, x_2)^\top \in \mathbb{R}^2$, and assume a model of no-motion up to an isotropic noise. Furthermore, we assume that the observation noise has only a radial component.

Definition F.1 (The simplified lidar problem)**.** The simplified lidar problem is the filtering problem modeled by Equation 1 with the following parameters:

$$F = H = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad Q = \begin{pmatrix} q & 0 \\ 0 & q \end{pmatrix} \qquad R_{polar} = \begin{pmatrix} r_0 & 0 \\ 0 & 0 \end{pmatrix}$$

for some $q, r_0 > 0$, where the observation noise ν_t is drawn i.i.d from $N(0, R_{polar})$ in *polar* coordinates. We also assume that the initial system-state x_0 is drawn from an isotropic distribution, i.e., follows a density function of the form $f((x_0)_1^2 + (x_0)_2^2)$, which is invariant to the target direction.

At a first glance, this simplistic problem may seem to satisfy the KF assumptions. In particular, both sources of noise are i.i.d. However, the observation noise is drawn i.i.d in *polar* coordinates, while the system state is estimated (and the prediction error is measured) in *Cartesian* coordinates. As discussed in Appendix A.4, the i.i.d property is *not* invariant to the coordinates transformation!

Lemma F.1. The observation noise in the simplified lidar problem is not i.i.d in Cartesian coordinates.

Proof. Denote the system state at time t by $x_t = ((x_t)_1, (x_t)_2)^\top$, and denote $\tan\theta = ((x_t)_2/(x_t)_1)$. From Definition F.1, by direct coordinates transformation, the observation noise is drawn from the

distribution $\nu_t \sim N(0, R(\theta))$, where

$$R(\theta) = \begin{pmatrix} r_0 \cos^2(\theta) & r_0 \cos(\theta)\sin(\theta) \\ r_0 \cos(\theta)\sin(\theta) & r_0 \sin^2(\theta) \end{pmatrix}$$

This noise covariance matrix varies every time-step with θ, hence the noise ν_t is not identically-distributed in Cartesian coordinates. Furthermore, consecutive time-steps are likely to have similar values of θ (according to the motion model), hence the noise is not independent either. □

If we are aware of the particular noise model of the problem, we can estimate R_{polar} in polar coordinates and transform it to the Cartesian $R(\theta)$ dynamically every time-step (e.g., using our the current estimate of θ, since the true θ is unknown). This is indeed what we do in the KFp model in Section 4. If we are not aware, however, we would simply estimate a constant covariance matrix R in Cartesian coordinates.

Lemma F.2 (Noise estimation in the simplified lidar problem)**.** Let R be the sample covariance matrix of the observation noise, calculated by Equation 2 with respect to a dataset of i.i.d targets in the simplified lidar problem. Then, as the number of targets in the data grows, R converges almost surely to

$$R_{est} = \begin{pmatrix} r_0/2 & 0 \\ 0 & r_0/2 \end{pmatrix}$$

Proof. From Definition F.1, the distribution of the initial system-states is isotropic, and so is the motion model Q. Hence, for any target at any time-step, the distribution of its direction θ (where $\tan\theta = ((x_t)_2/(x_t)_1))$ follows the uniform distribution: $\theta \sim [0, 2\pi)$. Now we can calculate:

$$E_\theta\left[(R_{est})_{11}\right] = E_\theta\left[r_0 \cos^2\theta\right] = \int_0^{2\pi} \frac{r_0}{2\pi}\cos^2\theta d\theta = \frac{r_0}{2}$$

$$E_\theta\left[(R_{est})_{22}\right] = E_\theta\left[r_0 \sin^2\theta\right] = \int_0^{2\pi} \frac{r_0}{2\pi}\sin^2\theta d\theta = \frac{r_0}{2}$$

$$E_\theta\left[(R_{est})_{12}\right] = E_\theta\left[(R_{est})_{21}\right] = E_\theta\left[r_0 \cos\theta\sin\theta\right] = \int_0^{2\pi} \frac{r_0}{2\pi}\cos\theta\sin\theta d\theta = 0$$

and since the targets are i.i.d, by the Law of Large Numbers we have

$$R \xrightarrow{\text{a.s.}} R_{est} = \begin{pmatrix} r_0/2 & 0 \\ 0 & r_0/2 \end{pmatrix}$$

□

So far we saw that the actual observation noise is $R(\theta)$ (where θ changes every time-step), while the constant covariance matrix that would be obtained by noise estimation is (asymptotically) R_{est}. Assuming that we constraint ourselves to a constant matrix (invariant to θ) in Cartesian coordinates, can we do better than R_{est}? More specifically, due to the radial symmetry of the problem, we can denote w.l.o.g

$$R_{opt}(r) = \begin{pmatrix} r & 0 \\ 0 & r \end{pmatrix}$$

for some $r > 0$, and ask what value of r would result in minimal square filtering errors of the KF.

Lemma F.3 (Optimization in the simplified lidar problem)**.** Assume that at a certain time-step in the simplified lidar problem, the state of the system is known to be normally-distributed with mean and covariance

$$x_0 = (x_1, x_2)^\top \qquad P_0 = \begin{pmatrix} p & 0 \\ 0 & p \end{pmatrix}$$

and that a new observation $z = (x_1 + dx_1, x_2 + dx_2)^\top$ is received. Denote by x the point-estimate of the KF following a filtering step with respect to noise parameters of the form $R_{opt}(r)$. Then, $r = \frac{pr_0}{2p+r_0}$ minimizes the expected square error (MSE) of x. Furthermore, $(R_{opt})_{ii} = r < (R_{est})_{ii}$ (for $i \in \{1, 2\}$), i.e., the optimal value of r is smaller than the value obtained by noise estimation.

Proof. As the true model of the observation noise is $R(\theta)$, by applying the KF update step (specified in Figure 1) we obtain the new true mean of the system-state distribution:

$$x_{true} = x_0 + P_0 H^\top (H P_0 H^\top + R)^{-1}(z - H x_0) = x_0 + P_0 (P_0 + R)^{-1}(z - x_0)$$
$$= x_0 + \begin{pmatrix} \frac{r_0 \sin^2\theta + p}{p + r_0} & -\frac{r_0 \cos\theta \sin\theta}{p + r_0} \\ -\frac{r_0 \cos\theta \sin\theta}{p + r_0} & \frac{r_0 \cos^2\theta + p}{p + r_0} \end{pmatrix} \begin{pmatrix} dx_1 \\ dx_2 \end{pmatrix} = \begin{pmatrix} x_1 + \frac{(r_0 \sin^2\theta + p)dx_1 - (r_0 \cos\theta \sin\theta)dx_2}{p + r_0} \\ x_2 + \frac{(r_0 \cos^2\theta + p)dx_2 - (r_0 \cos\theta \sin\theta)dx_1}{p + r_0} \end{pmatrix}$$

where θ corresponds to the (unknown) true direction of the target. However, due to the restricted form of $R_{opt}(r)$ (which is independent of θ), the actual estimate would be

$$x = (x_1 + \frac{p}{p + r} dx_1, x_2 + \frac{p}{p + r} dx_2)^\top$$

The expected error of the point-estimate x can be decomposed into two terms: the bias of x, and the variance of the system-state distribution. More formally, we can write $MSE = MSE_{var}(P_{true}) + MSE_{bias}(x, x_{true})$, where we can only control the latter. Specifically, we denote $a(r) \coloneqq p/(p + r)$ and look for the value of a that minimizes the MSE. We use the identities $\sin 2\theta = 2\cos\theta \sin\theta$ and $E\left[dx_1^2\right] = E\left[dx_2^2\right] = q$ (the latter is by the motion model in Definition F.1).

$$\begin{aligned}
MSE_{bias}(x(a), x_{true}) &= E||x - x_{true}||^2 \\
&= E\left[\left((a - \frac{r_0 \sin^2\theta + p}{p + r_0})dx_1 + \frac{r_0 \sin(2\theta)/2}{p + r_0} dx_2\right)^2\right. \\
&\quad \left. + \left((a - \frac{r_0 \cos^2\theta + p}{p + r_0})dx_2 + \frac{r_0 \sin(2\theta)/2}{p + r_0} dx_1\right)^2\right] \\
&= E\left[dx_1^2\left(a^2 - 2a\frac{r_0 \sin^2\theta + p}{p + r_0} + C_1\right) + \frac{r_0^2 \sin^2(2\theta)/4}{(p + r_0)^2} dx_2^2 + A_1 dx_1 dx_2\right. \\
&\quad \left. + dx_2^2\left(a^2 - 2a\frac{r_0 \cos^2\theta + p}{p + r_0} + C_2\right) + \frac{r_0^2 \sin^2(2\theta)/4}{(p + r_0)^2} dx_1^2 + A_2 dx_1 dx_2\right] \\
&= 2qa^2 - 2qa\frac{r_0 + 2p}{p + r_0} + q(C_1 + C_2) + q\frac{r_0^2 \sin^2(2\theta)/2}{(p + r_0)^2}
\end{aligned}$$

where $C_{1,2}$ are independent of a, and $A_{1,2}$ are multiplied by $E[dx_1 dx_2] = 0$ and vanish. To minimize we calculate

$$0 = \frac{\partial MSE_{bias}(x(a), x_{true})}{\partial a} = 4q \cdot a - 2q\frac{2p + r_0}{p + r_0}$$

which gives us

$$a = \frac{p + r_0/2}{p + r_0}$$

Note that the expression of MSE_{bias} above clearly diverges as $|a| \to \infty$, hence the only critical point necessarily corresponds to a minimum of the MSE. Thus, the optimal MSE is given when substituting the following r in R_{opt}:

$$r = p/a - p = \frac{p^2 + pr_0 - (p^2 + pr_0/2)}{p + r_0/2} = \frac{pr_0}{2p + r_0}$$

For the last part of the lemma, we recall that $(R_{est})_{ii} = r_0/2$ and compare to r directly:

$$(R_{est})_{ii} - (R_{opt})_{ii} = r_0/2 - r = \frac{(pr_0 + r_0^2/2) - pr_0}{2p + r_0} = \frac{r_0^2/2}{2p + r_0} > 0$$

□

Lemma F.3 shows that independently of the value of x_0 (i.e., independently of the current state of the system or the current direction of the target), the expected filtering error would benefit from reduction

of the noise parameter to $(R_{opt})_{ii} = r < (R_{est})_{ii}$. Hence, noise estimation is not an optimal strategy for parameters tuning in this problem.

The simplified lidar problem can be seen as a simplification of the problem experimented in Appendix J, with a degenerated motion model. In the experiments, the optimization of the KF reduced the MSE by 14.9%, and the learned parameters indeed decreased the observation noise, as shown in Figure 8.

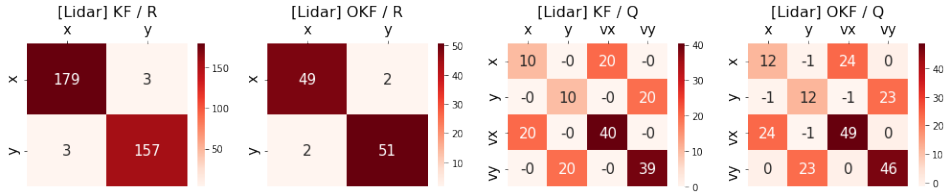


Figure 8: The observation noise (R) and prediction noise (Q) matrices obtained by noise estimation (KF) and by optimization (OKF), for the problem of state-estimation from lidar measurements (described in Appendix J).

In summary, in this section we introduce a simplified lidar problem. While at first glance the problem seems to follow the KF assumptions, the i.i.d-noise assumption is in fact violated by the natural polar coordinates of the sensor. Consequentially, the optimal values of the R parameters do not correspond anymore to the covariance of the observation noise, but rather should be reduced. In other words, **the non-i.i.d nature of the noise reduces the *effective* noise**. This theoretic result is consistent with the experiments presented in Appendix J.

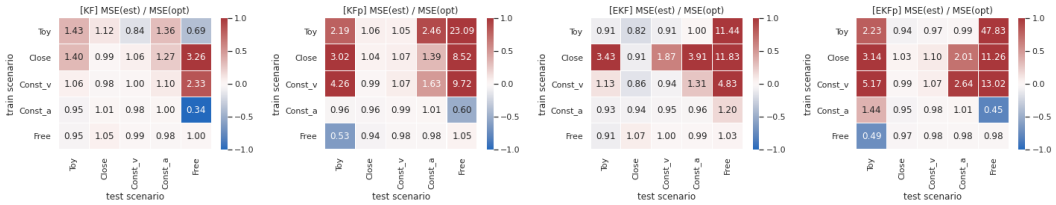
G Generalization Tests: Training and Testing on Different Benchmarks

In the case-study of Section 4, we demonstrate the robustness of an Optimized KF (OKF) over a variety of benchmarks representing different tracking scenarios – where in every benchmark, OKF (Method 2) obtained lower estimation errors than a standard KF (Method 1), **over an out-of-sample test data**. This demonstrates that OKF did not overfit the training data, but still relies on representativeness of the training data, i.e., on the assumption that the training data and the test data are taken from the same distribution.

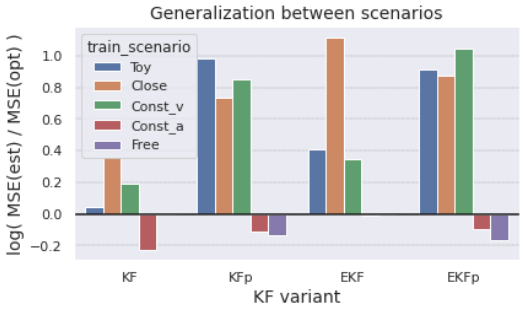
A stronger generalization ability is demonstrated in Section 5, where OKF beats both KF and NKF not only on out-of-sample test data with the same settings, but also **on a test data with different settings**. Specifically, we consider changes of factors 0.5 and 2 in the scale of the targets acceleration. This essentially changes the motion of the targets, including turns sharpness, speed changes and typical scale of speed. In terms of the linear motion model of KF, the acceleration corresponds to the motion noise Q, hence we essentially changed the noise after it had been learned, which poses a significant generalization challenge. Thus, the results of Section 5 provide a significant evidence for the robustness of OKF to certain distributional shifts.

In this section we present a yet stronger evidence for the robustness of OKF – not over a parametric distributional shift, but **over entirely different benchmarks**. Specifically, we consider the 5 benchmarks (or scenarios) of Section 4. For every pair (train-scenario, test-scenario), we train both KF and OKF on data of the train-scenario, then test them on data of the test-scenario. For every such pair of scenarios, we measure the generalization advantage of OKF over KF through $MSE_ratio = MSE(KF)/MSE(OKF)$ (where $MSE_ratio > 1$ indicates positive advantage). To measure the total generalization advantage of a model trained on a certain scenario, we calculate the geometric mean of MSE_ratio over all the test-scenarios (or equivalently, the standard mean over the logs of the ratios). Note that the logarithmic scale is indeed more natural for a symmetric view of the ratio of the two MSE scores.

This test is quite noisy, since a model optimized for a certain scenario may legitimately be inferior in other scenarios. Yet, considering all the results together in Figure 9, it is evident that optimization of KF yields more robust models in most cases, and even when it *is* inferior to noise-estimation, it does not lose by a large margin.



(a) $MSE_ratio = MSE(KF)/MSE(OKF)$ for every KF-baseline (KF,KFp,EKF,EKFp defiend in Section 4), and for every pair of train-scenario and test-scenario. The colormap is logarithmic ($\sim log(MSE_ratio)$), where red values represent advantage to OKF ($MSE_ratio > 1$).



(b) For every train-scenario, MSE_ratio is averaged over all the test-scenarios and is shown in a logarithmic scale. Positive values indicate advantage to OKF.

Figure 9: Generalization tests: optimization vs. noise estimation under distributional shifts.

H Sensitivity of KF-Optimization to Train Dataset Size

Each benchmark in the case-study of Section 4 has 1500 targets in its train data. Training from these datasets demonstrated significant advantage to optimization of KF over noise-estimation. However, one may argue that on smaller datasets, optimization may struggle harder than noise-estimation, e.g., due to numeric instability with little data. Furthermore, the standard optimization procedure "wastes" a portion of the train data as a validation set, which does not contribute directly to the training, and thus may increase the sensitivity to the amount of data.

To test the sensitivity of KF optimization to the amount of data, we repeated some of the tests of Section 4 using smaller subsets of the train datasets – beginning from as few as 20 targets per dataset. Figure 10 demonstrates that the advantage of OKF over KF holds consistently over the various sizes of the train datasets, although it indeed increases with the amount of data. In fact, in the Free Motion benchmark, the state-estimation errors of the standard KF monotonously *increase* with the amount of data.

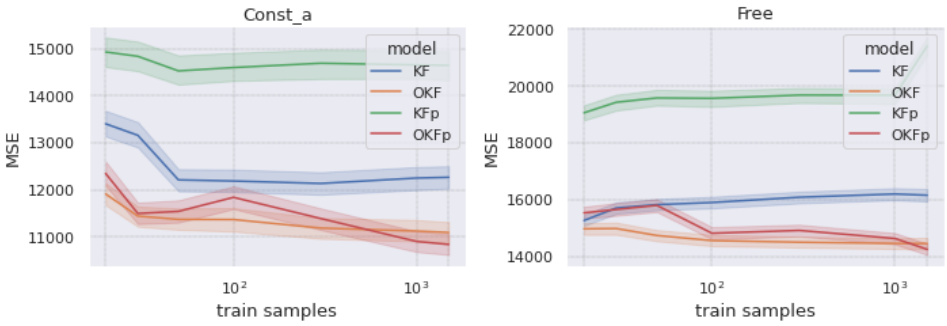


Figure 10: The advantage of OKF over KF remains positive – although sometimes smaller – even when the train data reduces to as few as 20 targets. The shadowed areas correspond to 95% confidence-intervals.

I OPTIMIZED KALMAN FILTER FOR VIDEO TRACKING

Radar tracking is an interesting domain with unique properties such as non-linear observation model and radial observation noise. In this domain, Section 4 demonstrates the sub-optimality of KF with estimated noise parameters, compared to a KF with optimized parameters. In this section, we compare the two in the popular and arguably-simpler domain of video tracking, using the MOT20 dataset (Dendorfer et al., 2020) (available under *Creative Commons Attribution-NonCommercial-ShareAlike 3.0* License).

MOT20 includes several videos with multiple targets (mostly pedestrians) to track. The dataset also includes the ground-truth location and size of the targets in every frame of the videos. We consider these ground-truth states as direct observations, i.e., we assume to have a detector with zero observation error (which in particular makes the observation model linear, in contrast to the radar tracking problem).

More specifically, we characterize a target using the state $X = (x, y, w, h, vx, vy) \in \mathbb{R}^6$ (two-dimensional location, size and velocity), and an observation as $Z = (x, y, w, h)$. Note that x, y, w, h are provided in the dataset and we derive vx, vy from x, y. The observation model H is known as described above, and for the motion model F we assume constant velocity and constant target size, leading to:

$$H = \begin{pmatrix} 1,0,0,0,0,0 \\ 0,1,0,0,0,0 \\ 0,0,1,0,0,0 \\ 0,0,0,1,0,0 \end{pmatrix}, F = \begin{pmatrix} 1,0,0,0,1,0 \\ 0,1,0,0,0,1 \\ 0,0,1,0,0,0 \\ 0,0,0,1,0,0 \\ 0,0,0,0,1,0 \\ 0,0,0,0,0,1 \end{pmatrix}$$

Q and R are estimated or optimized with relation to the train data, as described in Section 4. For the train data we use MOT20-01,MOT20-02,MOT20-03, and for the test data MOT20-05. In particular, our test data comes from an entirely different video than the train data, hence the testing is less prone to overfit. The goal function for the optimization is the MSE (mean square error) of the location x, y after the prediction step (note that in absence of observation noise, the error of the update step is simply zero).

As shown in Figure 3, the optimized model (OKF) reduces the prediction errors (MSE) by 18% in comparison to a KF with standard noise estimation. The significance level of the results is also indicated by the huge z-value $z = 125$, corresponding to the errors difference (calculated as $z = mean(\Delta err)/std(\Delta err) \cdot \sqrt{N_{samples}}$). The results indicate that the sub-optimality of noise estimation – as well as the benefits of parameters optimization – are a cross-domain phenomenon and are not limited to radar tracking or to non-linear observation models.

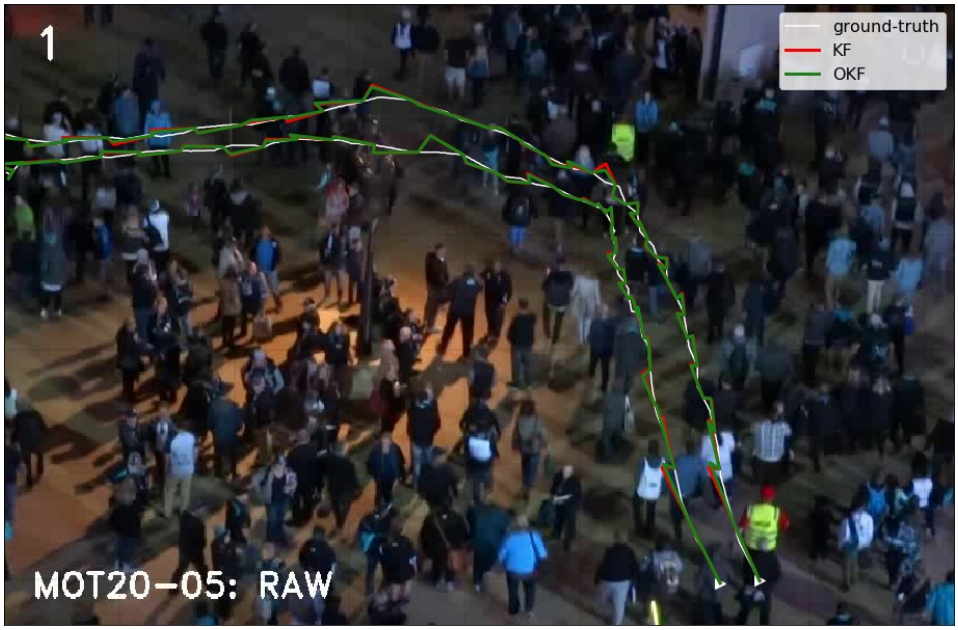


Figure 11: A sample of 2 targets in the first frame of the test video in MOT20. The true trajectories of the targets are shown along with the predictions of KF and OKF (each prediction is done one time-step in advance).

J Optimized Kalman Filter for Lidar-based State Estimation in Self Driving

Another domain in which we test the Optimized Kalman Filter (OKF) of Section 4 is state-estimation in self-driving based on lidar measurements with respect to known landmarks (Moreira et al., 2020). For simplicity, we focus on location estimation and on a single landmark (as the matching problem from multiple landmarks is out of the scope of this work).

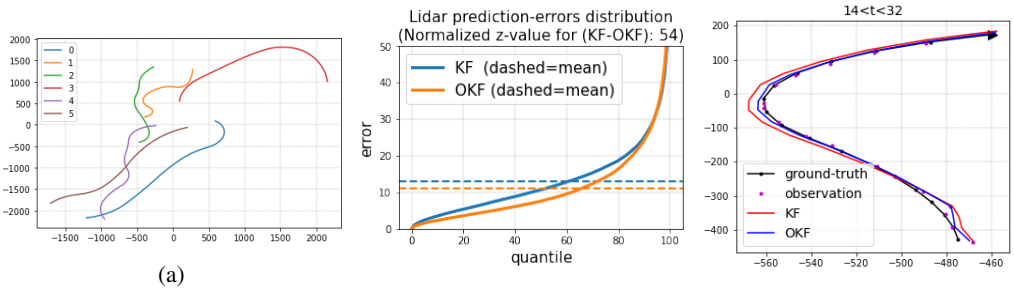
We formulate the problem as follows: the target state is $X = (x, y, vx, vy) \in \mathbb{R}^4$ (two-dimensional location and velocity), and an observation is $Z = (x, y)$. The dynamics are simulated as follows: each target trajectory consists of several intervals, in each one the target has certain (positive or negative) acceleration and certain lateral acceleration ("turn magnitude"), both drawn randomly at the beginning of the interval. The observation noise is drawn i.i.d in polar coordinates. We simulated a corresponding dataset of 2000 targets as demonstrated in Figure 12a, and split them to train set (1400 targets) and test set (600 targets).

From the KF point-of-view, the dynamics and observations are modeled by:

$$H = \begin{pmatrix} 1, 0, 0, 0 \\ 0, 1, 0, 0 \end{pmatrix}, F = \begin{pmatrix} 1, 0, 1, 0 \\ 0, 1, 0, 1 \\ 0, 0, 1, 0 \\ 0, 0, 0, 1 \end{pmatrix}$$

Note that the lidar problem is inherently different from the radar problem (Section 4) since the observation model is linear (due to the absence of the Doppler signal); and is different from video tracking (as formulated in Appendix I) since the observations are not assumed to be noiseless. On the other hand, similarly to the case of the radar, the Cartesian representation of the observations prevents their noise from being i.i.d, since the observation noise is only i.i.d in polar coordinates (see Appendix A.4).

We trained the OKF of Section 4 with respect to the MSE of the estimated target location, as described in Appendix B.1; and trained a standard KF by estimating the noise as in Equation 2. Then we tested both models on the test data. The optimization significantly reduced the errors by 14.9%, as shown in Figure 12b. This joins the promising results from the two other domains (radar tracking and video tracking), and demonstrates the generality of our method.



(a) (b) (c)

Figure 12: (a) A sample of 6 targets trajectories from the simulated self-driving data. (b) Prediction errors of KF and OKF on 600 targets of the test data. OKF is clearly more accurate, as also indicated by the huge z-value corresponding to the errors difference (calculated as $z = mean(\Delta err)/std(\Delta err) \cdot \sqrt{N}$). (c) the tracking of KF and OKF on a sample target trajectory, focused on the time interval $14 < t < 32$ for visual clarity.

K NEURAL KALMAN FILTER: EXTENDED EXPERIMENTS

Section 5 demonstrates that even if a neural-network (NN) based model provides more accurate predictions than a standard KF – its advantage may be entirely vanished when compared to an optimized KF instead. It is shown for the specific NKF model in a free-motion benchmark in the Doppler radar problem. This is sufficient to demonstrate the error in the common methodology that compares optimized NN-based models to non-optimized variants of KF.

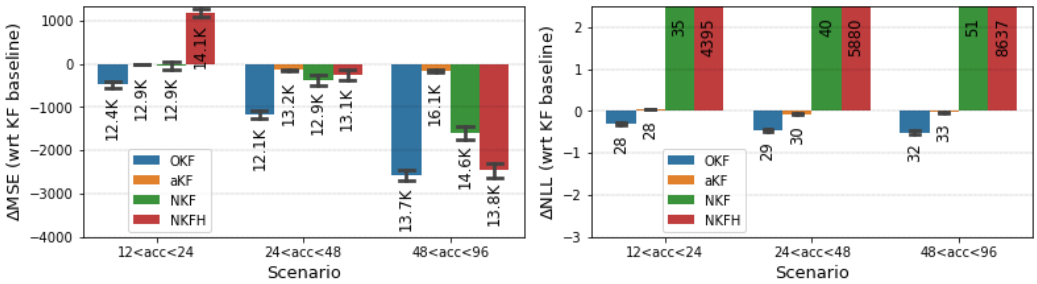
To show that these results are not exclusive to the experiments of Section 5, we present extended experiments in this section. Specifically, we consider 2 different benchmarks – the free-motion benchmark of Section 5 and Const_a benchmark of Section 4 (in which the targets may have acceleration but not turns). We also consider 3 different neural models:

- Predicted-acceleration KF (aKF): a variant of NKF without dynamic prediction of the covariance matrices Q and R in every step (in terms of Appendix B: a tracker with elements 3, 4).
- Neural KF (NKF): the model used in Section 5 and shown in Figure 7 (in terms of Appendix B: a tracker with elements 4, 5, 6).
- Neural KF with H-prediction (NKFH): a variant of NKF that also predicts the observation model H dynamically in every step (in terms of Appendix B: a tracker with elements 4, 5, 6, 7).

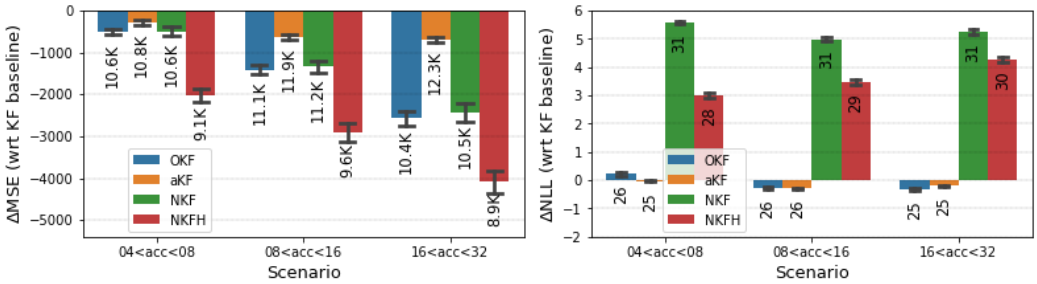
For each benchmark and each model, we trained the model on train data with a certain range of targets acceleration (note that acceleration affects both speed changes and turns sharpness), and tested it on targets with three different acceleration ranges – the original one, a smaller one and a larger one. That is, some of the test datasets correspond to distributional shifts. For each model we train two variants – one with Cartesian representation of the observation noise R, and one with polar representation (as in the baselines of Section 4) – and we select the best among the two variants using the MSE scores on the validation data (which is a portion of the data assigned for training).

We measure the results using (1) the Mean Square Error (MSE) of the estimated location after each update step; and (2) the Negative-Log-Likelihood (NLL) of the true system-state with relation to the estimated state-distribution, after the prediction step. Note that NLL is important for matching observations to targets in the multi-target tracking problem (which is out of the scope of this work).

Figure 13a shows that in the free-motion benchmark, all the 3 neural models improve the MSE in comparison to a standard KF, but lose to an optimized one (OKF). Furthermore, while OKF has the best NLL scores, the more complicated models NKF and NKFH increase the NLL in orders of magnitude. This issue may be handled by explicitly optimize the NLL in addition to the MSE, but such multi-loss optimization is out of the scope of this work. Note that the instability of NKFH is expressed in poor generalization to lower accelerations in addition to the extremely high NLL score.



(a) Free-motion benchmark



(b) Const_a benchmark (no turns)

Figure 13: The *relative* MSE and NLL results of various models in comparison to the standard KF model. The textual labels specify the *absolute* MSE and NLL. Note that certain bars of NLL are of entirely different scale and thus are cropped in the figure (their values can be seen in the labels). In each benchmark, the models were trained with relation to MSE loss, on train data of the middle acceleration-range: the two other acceleration ranges in each benchmark correspond to generalization over distributional shifts.

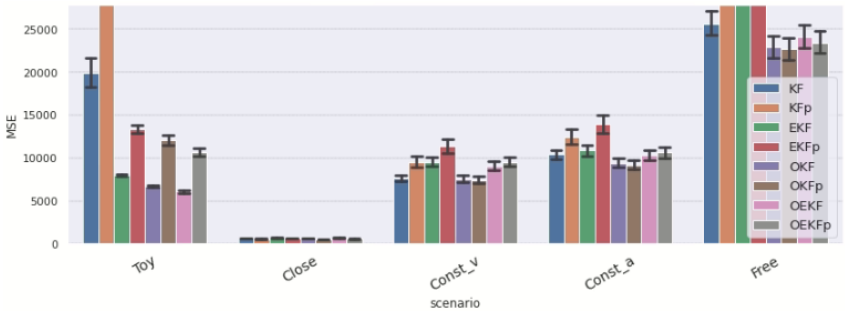
Figure 13a shows that in Const_a benchmark, all the 3 neural models improve the MSE in comparison to a standard KF, but only NKFH improves in comparison to OKF as well. Note that while NKFH does better in this case than in the free-motion benchmark, it still suffers from very high NLL.

In summary, by comparing the optimized neural models to the standard KF model, all the 3 models would be found superior to the KF in both benchmarks in terms of MSE. However, when shifting the baseline to an *optimized* KF, we find that the neural aKF and NKF are in fact inferior to OKF, and that the comparison between NKFH and OKF depends on the selected benchmark and metric.

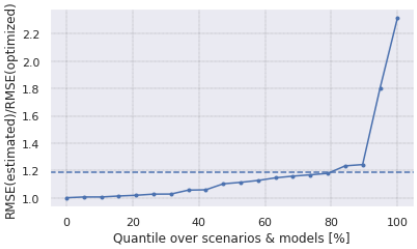
Of course, our results do not imply that neural-networks in general cannot be superior to a KF: only that when comparing the two, one must optimize the KF similarly to the neural model to avoid misleading experimental results. As discussed in Section 6, the wrong methodology of using a non-optimized KF baseline is very common in the literature.

L Detailed Results

Figures 14-17 below provide additional details about the experiments discussed in Sections 5,4.

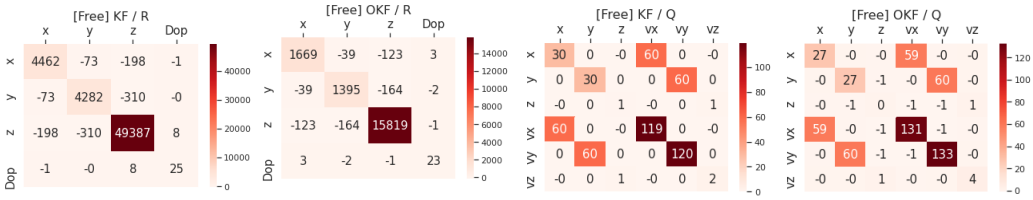


(a)



(b)

Figure 14: (a) Errors of different tracking models over different benchmarks. This is a different visualization of the results of Table 1. The error-bars correspond to confidence intervals of 95%. Note that the optimized models (the last four) tend to yield better and more stable results than the estimated models (the first four). (b) The RMSE ratio between the estimated models and the optimized ones, over all the benchmarks and designs discussed in Section 4. Note that all the ratios are above 1, i.e., all the models in all the benchmarks had smaller errors when tuned by optimization. The horizontal line represents the average ratio.



(a) R (estimated) (b) R (optimized) (c) Q (estimated) (d) Q (optimized)

Figure 15: The observation noise (R) and prediction noise (Q) matrices obtained in a (Cartesian) KF by noise estimation and by optimization with relation to MSE, based on the dataset of the Free-motion benchmark.

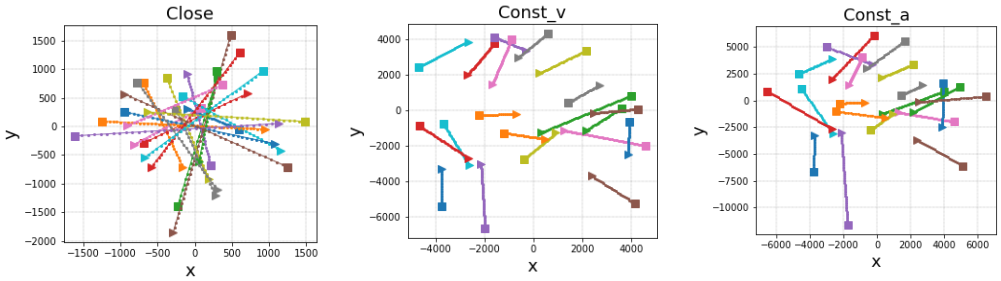


Figure 16: Samples of targets trajectories in the various benchmarks described in Section 4, projected onto the horizontal plane. This is an extension of Figures 2b,2c for the rest of the benchmarks.

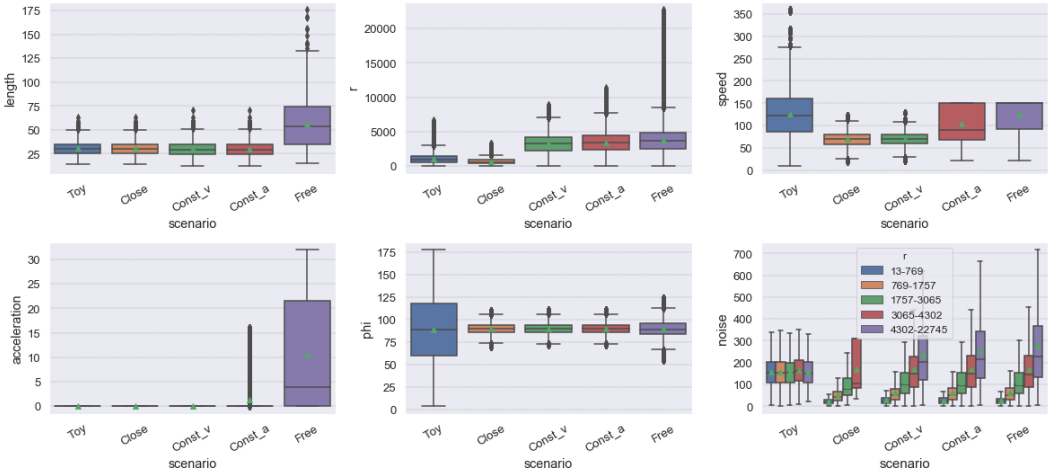


Figure 17: Descriptive statistics of the various benchmarks of Section 4: duration of targets trajectories; distance from the radar; targets speed; targets acceleration; motion direction (90 degrees correspond to horizontal motion); and observation errors vs. distance from radar.