DATA-DRIVEN KALMAN-BASED VELOCITY ESTIMATION FOR AUTONOMOUS RACING

Adrià López Escoriza, Guy Revach, Nir Shlezinger, and Ruud J. G. van Sloun

ABSTRACT

Real-time velocity estimation is a core task in autonomous driving, which is carried out based on available raw sensors such as wheel odometry and motor currents. When the system dynamics and observations can be modeled together as a fully known linear Gaussian state space (SS) model, the celebrated Kalman filter (KF) is a low complexity optimal solution. However, both linearity of the underlying SS model and accurate knowledge of it are often not encountered in practice. This work proposes to estimate the velocity using a hybrid data-driven (DD) implementation of the KF for non-linear systems, coined KalmanNet. KalmanNet integrates a compact recurrent neural network in the flow of the classical KF, retaining low computational complexity, high data efficiency, and interpretability, while enabling operation in non-linear SS models with partial information. We apply KalmanNet on an autonomous racing car as part of the Formula Student (FS) Driverless competition. Our results demonstrate the ability of KalmanNet to outperform a state-of-the-art implementation of the KF that uses a postulated SS model, while being applicable on the vehicle control unit used by the car.

Index Terms— Kalman filter, autonomous vehicles.

1. INTRODUCTION

Many emerging technologies carry out real-time state estimation of dynamical systems. For instance, velocity estimation (VE) is a key and central component of autonomous vehicles [1]. The velocity estimates are used by the perception algorithms to compensate for motion when sensing the environment; they are required to provide localization and mapping; and are essential for providing feedback to the vehicle motion control system. VE must therefore provide high-rate, high-quality data. Due to the inertial nature of velocity, directly measuring it involves using expensive external velocity sensors such as optical flow sensors. While these sensors provide robustness against extreme conditions, their high cost limits their applicability for commercial road cars. It is therefore desired to design robust and accurate VE mechanisms that can process sensory data acquired by the vehicle, at a reduced cost, while operating reliably in extreme scenarios like adverse weather without requiring driver attention, thus reaching full (level 4) autonomy [2].

The generic problem of state estimation is typically tackled using either of two leading strategies. The most widely used solution is based on the celebrated Kalman filter (KF) [3]. The KF is the minimum mean-squared error (MMSE) estimator in linear, time-invariant, state space (SS) models with Gaussian noise, and requires the model to be fully known. In many practical setups, including VE in self-driving cars, this model assumption may not hold; the underlying dynamics are complex and non-linear, while domain knowledge often relies on a crude approximation. Well-known model-based (MB) variants of the KF are designed for non-linear dynamics such as the extended Kalman filter (EKF) [4, Ch. 7] and the unscented Kalman filter (UKF) [5] but they are not MMSE optimal and are severely degraded by model mismatch [6]. An alternative strategy is to learn the filter mapping from data. In particular, neural network (NN) architectures such as recurrent neural networks (RNNs) have been shown to learn to carry out time series predication [7]. The work [8] trained an RNN outperforming MB KF architectures. While such data-driven (DD) architectures can learn to capture complex dynamics, they tend to require many trainable parameters even for seemingly simple sequence models [9], and may be computationally prohibitive to implement on limited hardware. In addition, these black-box architectures provide results that are not explainable enough, which makes them unfit for critical applications. These constraints limit the application of highly parameterized deep models for real-time state estimation in
applications embedded on hardware-limited mobile devices such as VE in autonomous vehicles.

In this work we study VE for self-driving cars using hybrid MB/DD state estimation [10]. Our design is based on the recent KalmanNet algorithm [11], which combines the soundness and low complexity of the classic KF while exploiting the model-agnostic nature of NNs to mitigate its dependence on accurate knowledge of the SS model. Our hybrid approach allows obtaining reliable velocity estimates in real time without relying on expensive velocity sensors. The method is demonstrated on a full scale autonomous race car, illustrated in Fig. 1, which can accelerate from 0 to 100 [km/h] in 2.1 s and reaches lateral accelerations of 1.7 g. KalmanNet operates at reduced complexity with a limited number of trainable parameters. It can thus be applied on the hardware-limited Vehicle Control Unit (VCU) of the autonomous car, as opposed to previously proposed purely DD estimators whose applicability was limited by their complexity [8]. We train the hybrid MB/DD KalmanNet to track the velocity of the car based solely on sensory inputs acquired from inertial measurement units (IMUs), wheel odometry, and motor currents; i.e., without relying on expensive velocity sensors. The resulting system is shown to notably outperform State-of-the-Art (SOA) KF with equivalent sensor setups, while approaching the accuracy achievable with costly sensory data. Our results demonstrate the potential of combining MB and DD methods in autonomous systems, while providing a proof of concept for KalmanNet in a challenging real-life scenario.

The rest of this paper is organized as follows: Section 2 discusses the system model, and Section 3 details the KalmanNet velocity estimator. Our experimental study is presented in Section 4, and Section 5 concludes the paper.

2. SYSTEM MODEL AND PRELIMINARIES

2.1. Velocity Estimation Problem Formulation

We consider the real-time VE problem in which at every time instance $t$ our goal is to estimate a velocity vector $\mathbf{x}_t$ from a vector of sensory measurements $\mathbf{y}_t$ that are acquired by the autonomous vehicle. In particular,

$$\mathbf{x}_t = \begin{pmatrix} \mathbf{v}_t^T & \mathbf{a}_t^T \end{pmatrix}^T \in \mathbb{R}^m, \quad m = 5$$

where, $\mathbf{v}_t = (v_{x,t}, v_{y,t})^T$ and $\mathbf{a}_t = (a_{x,t}, a_{y,t})^T$ denotes the velocity and acceleration along the $x - y$ axis, respectively, and $\mathbf{a}_t$ denotes the rotational (yaw) rate along the $z$-axis. For the estimation task, we consider $n = 18$ raw measurements that are collected by sensors on the autonomous car. Two IMUs measure yaw rate and accelerations $\mathbf{y}_{IMU} = (\dot{\psi}, \ddot{a}_x, \ddot{a}_y)^T$. Four motor encoders measure the rotational velocity of the $i$-th wheel $\dot{\psi}_i$. One steering angle sensor measures the steering angle of the wheels $\delta_i$. One current sensor measures the torque $T_i$ of the $i$-th wheel. The data from the wheel encoders is fused to get a velocity estimate of the $i$-th wheel. The car is also equipped with an optical flow-based velocity sensor and a GNSS velocity sensor, which are used only for validation and ground truth (GT). The sensor setup is depicted in Fig. 2.

2.2. State Space Model

To track the velocity, we assume that the evolution of the velocity state vector $\mathbf{x}_t$ together with the observed sensory data $\mathbf{y}_t$, can be described by a SS model. Specifically, as in [12], we assume that the evolution of the velocity is described by a set of continuous-time differential equations:

$$\dot{\mathbf{a}}_t = \mathbf{q}_{a,t}, \quad \dot{\psi}_t = \mathbf{q}_{\psi,t}$$

where the change in the accelerations is modeled as an additive white Gaussian noise (AWGN) ($\mathbf{q}_a$ and $\mathbf{q}_\psi$). Using standard techniques [13] we obtain the discrete-time evolution model:

$$\mathbf{x}_t = \mathbf{F}_t \cdot \mathbf{x}_{t-1} + \mathbf{q}_t, \quad \mathbf{q}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}).$$

Here, $\mathbf{Q}$ is an unknown covariance matrix, and $\mathbf{F}_t$ is a time-dependent evolution matrix with $\Delta t = 5$ [ms]

$$\mathbf{F}_t = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 \\ -\dot{\psi}_{t-1} \cdot \Delta t & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$  

We assume a partially observable system, where the observations vector $\mathbf{y}_t$ is generated from $\mathbf{x}_t$ via

$$\mathbf{y}_t = \mathbf{h}(\mathbf{x}_t) + \mathbf{n}_t, \quad \mathbf{n}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}).$$

Here, $\mathbf{R}$ is an unknown covariance matrix. The hidden states are observed by the sensors in the following way: accelerations and rotation rate are noisy measurements directly sensed

Fig. 2: Sensor setup: two IMUs, four motor encoders + current sensors, a steering angle sensor, and two dedicated velocity sensors that are used for validation and target generation.
by the two IMUs
\[
\left( \hat{\psi}_t, \hat{\omega}_t, \hat{\alpha}_t \right)^T = I_{3 \times 3} \cdot \left( \psi_t, \alpha_t, \omega_t \right)^T + n_{IMU,t} \tag{6}
\]

As in [14], measurements from the wheel encoders are fused to get an estimate of the wheel velocity

\[
\hat{v}_{i,t} = \left[ \cos(\delta_{i,t}), \sin(\delta_{i,t}) \right]^T \cdot \omega_{i,t} \cdot \frac{R_i}{SR(T_{i,t}) + 1}. \tag{7}
\]

Here, SR(·) is a function that maps the torque to its slip ratio under low slip conditions (Pacejka magic tire model [15]), and \( R_i \) is the radius of the \( i \)-th wheel. A noisy observation for the car velocity is now assumed to be given by

\[
i^*_t = \arg \min_{i \in \{1, \ldots, 4\}} \{ \hat{v}_{i,x,t} \}
\]

\[
\hat{v}_{i^*_t,t} = \hat{v}_t + \left[ \dot{\psi}_{t-1} \cdot p_{i^*_t,y}, -\dot{\psi}_{t-1} \cdot p_{i^*_t,x} \right]^T + n_{v,t}. \tag{8b}
\]

Here, \( n_{v,t} \) is AWGN, \( p_{i,x} \) and \( p_{i,y} \) are constants denoting the positions of the \( i \)-th wheel in the car frame, and \( i^*_t \) is the index of the wheel with the lowest estimated x-axis velocity; i.e., slip. The reason that the velocity update only takes the single wheel with the smallest absolute estimated slip stems from the fact that the slip ratio calculation is fairly accurate at low slips, but uncertain at high slips. By using this model we can use \( n = 8 \) fused observation as input to the filter.

2.3 Kalman Filtering in Velocity Estimation

We compare the performance of KalmanNet to a Mixed Kalman Filter (MKF) [8]. It is a MB approach that uses a combination of EKF, UKF, and a chi-squared test for detecting outlier measurements. MKF is the filter that was embedded in the autonomous car (pilatus), it is well tuned and optimized, it was extensively tested and proved to be successful in multiple FS competitions around Europe in 2019 and during several testing sessions in different kind of tracks in 2020, and is therefore considered SOA for VE in autonomous racing cars. In the MKF the state is first propagated with an EKF step, then a chi-squared test is done on the measurements. The update step is performed via EKF by default except for the wheelspeed measurements, which, due to the strong non-linearity of the model, are updated with a UKF step.

3. HYBRID MB/DD VELOCITY ESTIMATION

Our hybrid velocity estimator is based upon our KalmanNet architecture presented in detail in [11]. We first briefly describe KalmanNet, then explain how it was applied to VE and discuss its properties. Finally we discuss the gains of KalmanNet compared to related architectures on a conceptual level.

3.1 KalmanNet

KalmanNet is built upon the flow of the KF. However, as opposed to classic KF that utilizes full domain knowledge to compute the Kalman gain (KG), KalmanNet learns the KG from data by training a compact RNN in an end-to-end (E2E) manner using GT state vectors. This adaptability to the data provides KalmanNet with robustness to model mismatch and removes the need to know the noise covariance matrices. A block diagram of KalmanNet, where the learned KG is integrated in the overall KF flow, is illustrated in Fig. 3. Compared with E2E black-box architectures, KalmanNet inherits the interpretability from the KF, providing explainable results, by learning how to perform Kalman Filtering rather than directly learning the dependencies between states and observations. These capabilities of KalmanNet were extensively evaluated in [11] for linear system models and in [16] for nonlinear system models.

3.2 KalmanNet Velocity Estimator

We adapt KalmanNet architecture to the task of VE by integrating the postulated SS model detailed in Section 2.2. Note that this integration involves only setting the mapping blocks \( f \) and \( h \), as knowledge of the noise statistics is not required by KalmanNet, which learns to compute the KG directly from data. We include a prepossessing block that performs sensor calibration and frame transformations based on the design specifications of the autonomous car and ignores the first time steps of the sensors to use them for bias calibration. We use a single gated recurrent unit cell with a hidden state of size proportional to the size of \( Q \) and \( R \). This reduced architecture allows the system to run in the VCU of the car at the high
3.3. Discussion

While classical KF architectures are proven robust and stable solutions for most estimation problems, they are limited by the model mismatch inherent to most real world applications. This model mismatch usually imposes an error floor on the accuracy of the estimator making them sub-optimal for challenging applications like autonomous driving. KalmanNet can overcome this error floor by adapting to the application through data, learning how to obtain the optimal performance of the KF. Secondly, as mentioned in Section 1, the pivotal role of VE in the autonomous system imposes a requirement for robustness and stability. While E2E DD architectures can accurately learn the complex dynamics of the system, the lack of explainability in their results makes them a risky approach for such critical applications. Taking the best of both worlds, KalmanNet fuses the interpretability of the classical KF with the adaptability of RNN to provide the autonomous system with accurate and reliable estimations for velocity.

4. EXPERIMENTS AND RESULTS

All the experiments were carried out using real data collected by the autonomous race car *pilatus*, developed by the members of AMZ Driverless across three consecutive FS seasons. Datasets were created from real data obtained from noisy sensors over both testing and competition runs on FS style tracks. The raw sensor measurements have different sampling frequencies of 200 [Hz] (IMU1 and steering angle), 125 [Hz] (IMU2), and 100 [Hz] (wheel speeds and torques). These raw measurements are hardware time-synced by sampling at a constant rate of 200 [Hz] (zero-order hold) and used as input to KalmanNet. Following the approach used in [8], we generate the reference for KalmanNet using the MKF detailed in Section 2.3 in combination with the costly external velocity sensors described in Section 2.1. The target of the KalmanNet approach should thus obtain similar results but do so without relying on expensive velocity sensors. The output from the MKF was also post-processed using a non-causal, Gaussian moving average filter to obtain a smoothed, non-delayed target, referred to as the reference. As a benchmark, we consider the MKF operating with the same sensors as KalmanNet; i.e., without the external velocity sensors. The mean-squared error of KalmanNet computed over the entire dataset compared to MKF is summarized in Table 1, while Figs. 5-6 show the tracking of the lateral velocity $v_y$ and the yaw rate $\dot{\psi}$ for data collected in a test run in Tuggen (Switzerland) in August 2020. We observe in Table 1 that KalmanNet outperforms the baseline in 4 of the 5 trackable states. Of particular interest are the gains in tracking $v_y$, since, due to the model inaccuracy at high slips, this is typically the most challenging variable to trace. Fig. 5 shows how while the baseline gets mislead by the inaccurate wheel-speed (WS) measurements, KalmanNet obtains a smoother estimation much closer to the reference; and the numerical evaluation over the entire dataset shows a 3.63 [dB] gain over the baseline. Fig. 6 shows that KalmanNet also manages to get a more accurate trajectory when we have accurate sensors, achieving a 4.42 [dB] gain over the baseline.

5. CONCLUSIONS

In this work we studied VE in autonomous racing cars using KalmanNet architecture. As KalmanNet uses a relatively compact NN it is applicable to computationally limited devices such as the VCU of most cars. Our experimental results show that KalmanNet approaches the performance of the SOA MKF with additional external velocity sensors, and outperforms it when working with the same sensors.
6. REFERENCES


