Using Machine learning to enhance a Kalman filter : application to the measurement of the velocity of a car

Maud Biquard Nathan Doumèche

Mines de Paris - MAREVA

Friday, January 28, 2022

Content

Content

Our supervisor

Colin Parellier
PhD at CAOR

Presentation of the problem

Goal : measure the velocity of a car

2 measures :

- the acceleration with noise (inertial measurement unit),
- the velocity of the wheels with noise, sliding and slipping.

We model sliding and slipping thanks to the Pacejska formula. One of the difficulties is that, for a given acceleration, there are two ways of sliding.

How can we merge these information ?

Merging two measures

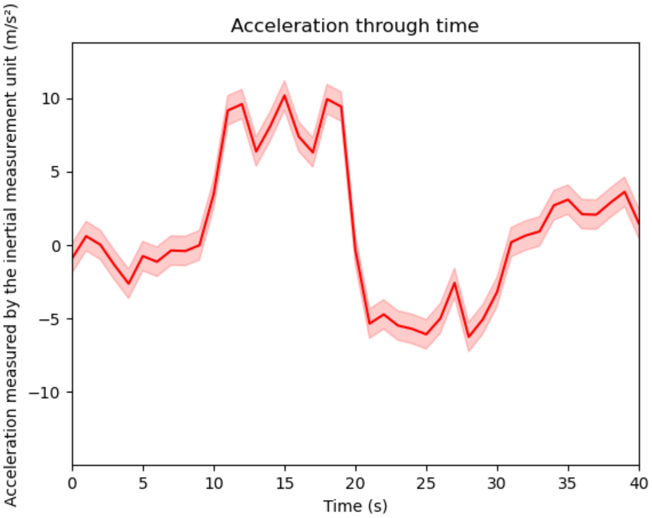


Figure 1 – The acceleration with a uniform noise

Merging two measures

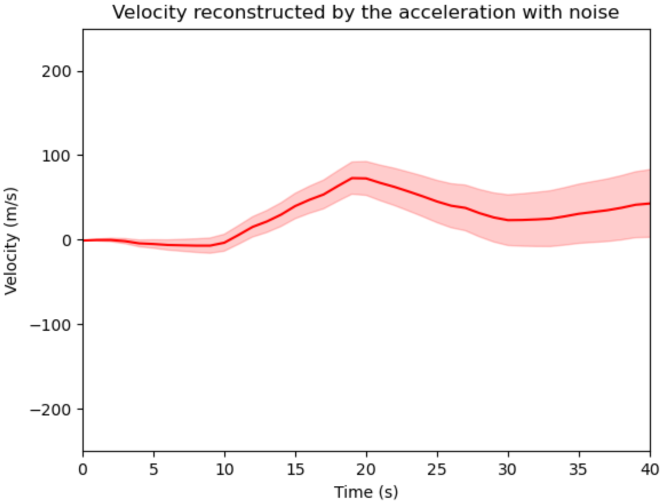
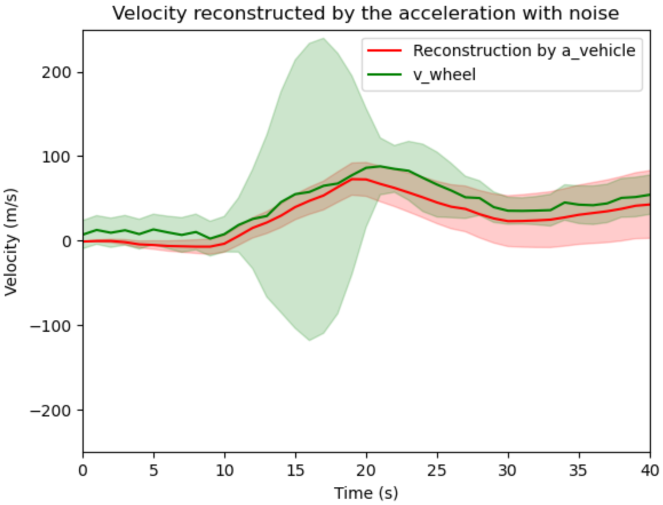


Figure 2 – Reconstructing the velocity by integrating the acceleration : propagation of the noise

Merging two measures



Merging two measures

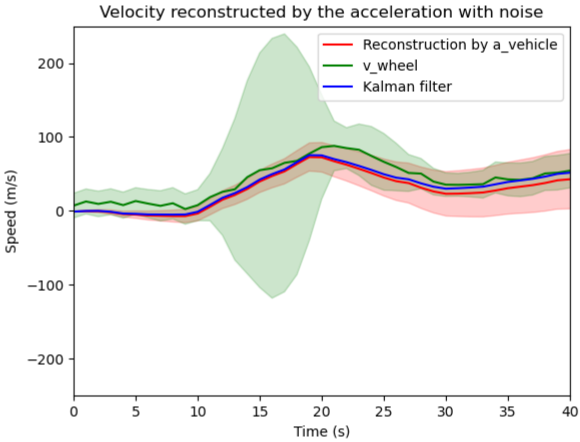


Figure 3 – When the variance of v^{wheel} is low, the Kalman filter gets close to it. Otherwise, it gets close to the reconstruction by integrating the acceleration.

The Kalman filter

We follow Fox et al. [2005].

Hypothesis : $t \mapsto a_t$ et $t \mapsto v_t^{wheel}$ are gaussian processes.

Prediction :

$$\begin{cases} \hat{v}_{t|t-1} & = \hat{v}_{t|t-1} + a_t & \quad Prediction \\ P_{t|t-1} & = P_{t-1|t-1} + Q_t & \quad Predicted\ covariance \end{cases}$$

Update :

$$\begin{cases} y_t & = v_t^{wheel} - \hat{v}_{t|t-1} & \quad Innovation \\ S_t & = P_{t|t-1} + \boldsymbol{R_t} & \quad Innovation\ Covariance \\ K_t & = P_{t|t-1} S_t^{-1} & \quad Optimal\ Kalman\ gain \\ \hat{v}_{t|t} & = \hat{v}_{t|t-1} + K_t y_t & \quad Update \\ P_{t|t} & = (I - K_t) P_{t|t-1} & \quad Update\ covariance \end{cases}$$

Content

Datas

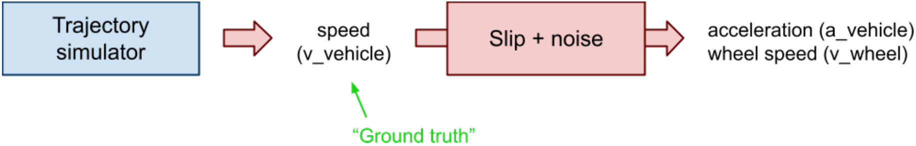


Figure 4 – Collecting the data

We call Classical Kalman filter the Kalman filter taken with a constant R.
We evaluate its performance by computing its Mean Squared Error :

$$MSE = \sum_{1 \leq t \leq T} (v_t^{vehicle} - \tilde{v}_t)^2.$$

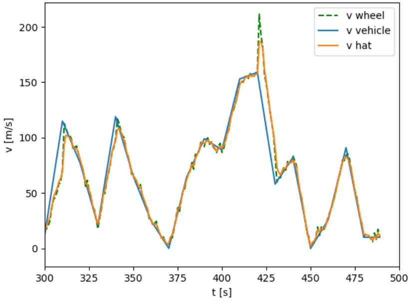


Figure 5 – Result of the Kalman filter by taking $Q_t = 1$ and $R_t = 1$. $MSE = 136.6$

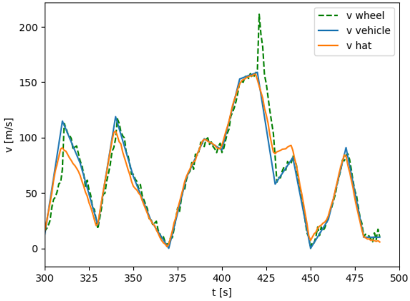


Figure 6 – Result of the Kalman filter by taking $Q_t = 1$ and $R_t = 150$. $MSE = 41.8$

Evaluating the minimal loss

The system we considered contains noise by nature (because of the captors). We evaluate its covariance : it is $P_{k|k}$. While learning, the loss can not converge towards 0. Let's compute a lower bound for the L^2-error. For a trajectory i, let $\mathcal{L}_i$ be its loss. $\tilde{v}_t$ is the velocity computed by the neural network.

$$\mathcal{L}(i) = \mathbb{E}\left(\sum_{1\leq t\leq T}\left(v_t^{vehicle}(i) - \tilde{v}_t(i)\right)^2\right)$$

Strong law of large numbers : $\bar{\mathcal{L}} = \frac{1}{N}\sum_{1\leq i\leq N}\mathcal{L}(i) \xrightarrow{a.s.} \mathbb{E}(\mathcal{L})$

The average loss during learning is therefore approximately $\mathbb{E}(\mathcal{L})$.

$$\begin{aligned}
\mathbb{E}\left(\mathcal{L}\right) &= \sum_{1\leq t\leq T}\mathbb{E}\left(\left(v_t^{vehicle} - \tilde{v}_t\right)^2\right) \\
&\geq \sum_{1\leq t\leq T}\mathbb{E}\left(\mathbb{E}\left(\left(v_t^{vehicle} - \hat{v}_{t|t}\right)^2 |t\right)\right) \text{ the Kalman filter is optimal in } L^2 \\
&= \sum_{1\leq t\leq T}\mathbb{E}\left(P_{t|t}\right) \text{ by definition of } P_{t|t}
\end{aligned}$$

Evaluating the minimal loss

Therefore, during the training phase, we expect the average loss to converge towards $\sum_{1 \leq t \leq T} \mathbb{E}\left(P_{t|t}\right) > 0$. In fact, we can estimate this loss thanks to a well-trained neural network. Once again, we use the Monte-Carlo method :

$$\frac{1}{N} \sum_{1 \leq i \leq N} \sum_{1 \leq t \leq T} P_{t|t}(i) \xrightarrow{p.s.} \sum_{1 \leq t \leq T} \mathbb{E}\left(P_{t|t}\right).$$

The result is that, during the training phase, the loss can not converge towards something smaller thant 3.1. In practive, we can't simulate the Kalman filter thanks to a neural network and $\tilde{v}_t \neq \hat{v}_{t|t}$. There is what is called an approximation error because of the approximation of R by the neural network. It is therefore normal to compute an average loss during training greater than the minimal loss.

The first network

- Takes a whole trajectory and returns $\hat{R} = (\hat{R}_t)_{1 \leq t \leq T}$.
- $\Rightarrow$ Downside : it needs to see the future.

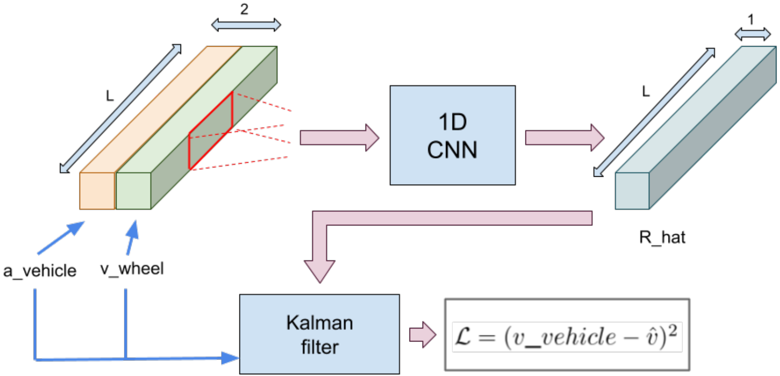


Figure 7 – First network

First network : training and results

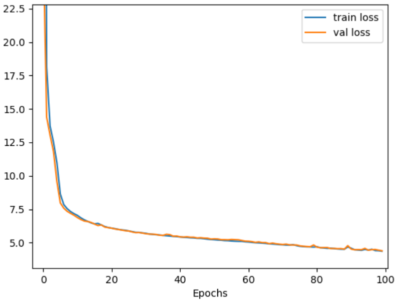


Figure 8 – Loss for the first network

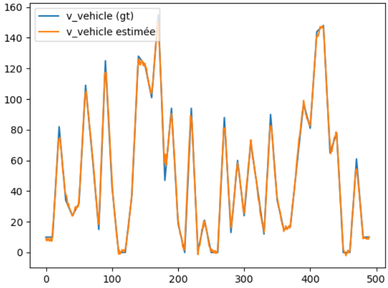


Figure 9 – $v^{vehicle}$ estimated by a Kalman filter with $\hat{R}_t$ given by the network. MSE is the mean on the whole test set : $MSE = 4.66$.

Real-time

You can't see the future for real uses.
Idea : estimate R_t depending on $a_u^{vehicle}$ and v_u^{wheel} for $t-p \leq u \leq t-1$.

- Same training but

$$\mathcal{L} = \sum_{1 \leq t \leq T} w_t \left(v_t^{vehicle} - \hat{v}_t\right)^2 \text{ with } w_t = \begin{cases} 1 & \text{si } t < T \\ K & \text{si } t = T \end{cases}.$$

- Inference : we only use $\hat{R}_T$ for the Kalman filter.

Real-time network : training and results

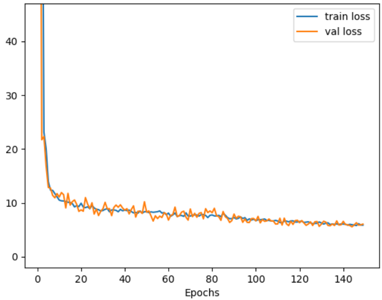


Figure 10 – Loss for the real time network

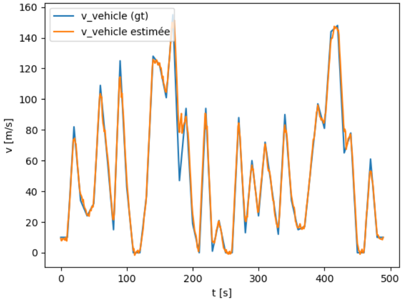


Figure 11 – $v^{vehicle}$ estimated by a Kalman filter, $p = 30$. $\hat{R}_t$ computed by the network. MSE is the mean on the whole test set : $MSE = 7.92$.

	Classical Kalman	Future Kalman	Real-time Kalman
MSE	35 - 45	4.66	7.92

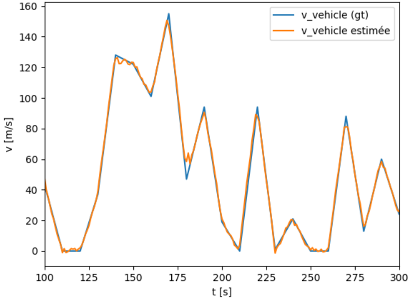


Figure 12 – First network (sees the future)

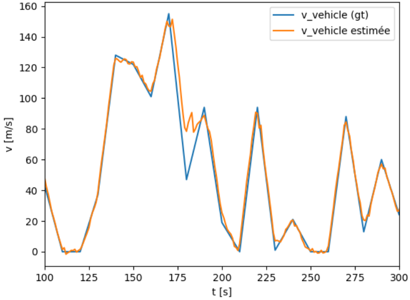


Figure 13 – Real-time network

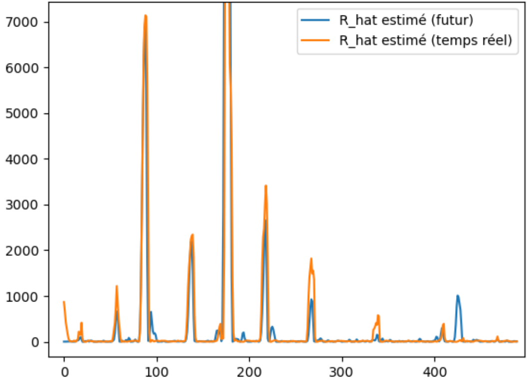


Figure 14 – Comparison between R_t returned by the future and the real-time networks

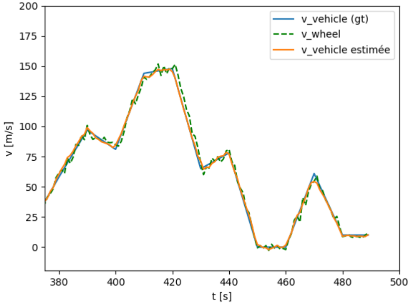


Figure 15 – First network (sees the future)

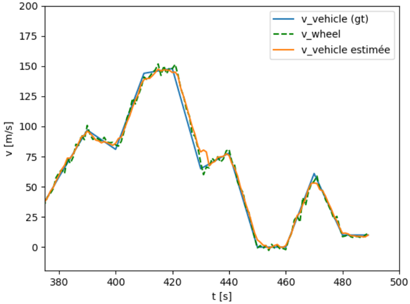
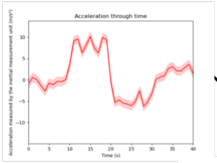


Figure 16 – Real-time network

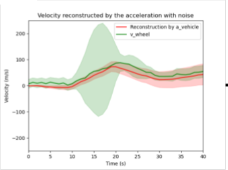
Content

Overview

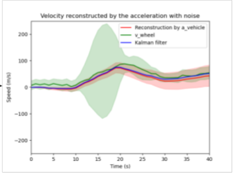
Inertial measurement unit :
Acceleration with noise



Q (random error) :
Determined by adjusting the captor



Kalman filter



Wheel speed :
Noise + sliding and slipping

R (random error + sliding and slipping) :
Computed by a CNN

Bibliography

Dieter Fox, Sebastian Thrun, and Wolfram Burgard. *Probabilistic Robotics*. The MIT Press, 2005.

Appendix 1 : Network structure

```
Sequential(
(0): Conv1d(2, 16, kernel_size=(3,), stride=(1,), padding=(1,))
(1): ReLU()
(2): Conv1d(16, 32, kernel_size=(5,), stride=(1,), padding=(2,))
(3): ReLU()
(4): Conv1d(32, 64, kernel_size=(5,), stride=(1,), padding=(2,))
(5): ReLU()
(6): Conv1d(64, 64, kernel_size=(5,), stride=(1,), padding=(2,))
(7): ReLU()
(8): Conv1d(64, 64, kernel_size=(5,), stride=(1,), padding=(2,))
(9): ReLU()
(10): Conv1d(64, 1, kernel_size=(3,), stride=(1,), padding=(1,))
(11): ReLU()
    )
```