A LiDAR Error Model for Cooperative Driving Simulations

Michele Segata*, Renato Lo Cigno*, Rahul Kumar Bhadani†, Matthew Bunting†, Jonathan Sprinkle†‡
*Dept. of Information Engineering and Computer Science, University of Trento, Italy
†Dept. of Electrical and Computer Engineering, The University of Arizona, USA
‡National Science Foundation, USA
{msegata,locigno}@disi.unitn.it {rahulbhadani,mosfet}@email.arizona.edu jsprinkl@nsf.gov

***Abstract*—Cooperative driving and vehicular network simulations have done huge steps toward high realism. They have become essential tools for performance evaluation of any kind of vehicular networking application. Yet, cooperative vehicular applications will not be built on top of wireless networking alone, but rather fusing together different data sources including sensors like radars, LiDARs, or cameras. So far, these sensors have been assumed to be ideal, i.e., without any measurement error. This paper analyzes a set of estimated distance traces obtained with a LiDAR sensor and develops a stochastic error model that can be used in cooperative driving simulations. After implementing the model within the PLEXE simulation framework, we show the impact of the model on a set of cooperative driving control algorithms.**

I. Introduction and Related Work

Performance evaluation for cooperative driving and vehicular networking applications has become crucial in proving the benefits of such applications in terms of safety, efficiency, and fuel consumption. The reasons are simple. Simulations are flexible and they can be used to assess any kind of performance, from the wireless physical layer up to the application layer. They are fast, cheap, repeatable, controllable, and they permit parametric studies. It is now a matter of a few minutes to setup a set of simulations exploring hundreds of different parameter combinations and, by using high performance computing clusters, a matter of a few hours to run them and analyze their outcome. Last but not least, they are "safe", in the sense that researchers can show how effective an application is in avoiding or mitigating the damages of an impact without harming real persons. Clearly, simulations use models, thus giving up some real world details for the sake of mathematical representability and lower complexity.

The highest level of realism to evaluate feasibility and performacne is obtained through Field Operational Tests (FOTs), where real vehicles and real hardware are employed [1], [2]. Needless to say, FOTs have huge costs in terms of personnel, equipment, and man-power. It is true that they can provide the highest level of realism, but controlling the environment is an issue, and thus experiments repeatability is not trivial. Moreover, scalability is also an issue. Although FOTs employing hundreds of vehicles exist [3], they are rare, if not unique. Simulation can be used to perform a first broad analysis, and then FOTs can be exploited to verify the results on a subset of configurations. Rather than competing tools, simulations and FOTs complement each other.

The research community can choose between several different simulation frameworks (Veins [4], VSimRTI [5]), mobility simulators (SUMO [6], VISSIM [7]), communication models [8]–[10], just to list a few of them. Yet, in the literature body of vehicular simulation models, sensor studies are missing. It is very difficult to find technical specifications of commercial sensors, and whenever found, the details are clearly not enough. For example, the commercial brochure of the Bosch LRR4 radar[1] indicates an accuracy of $\pm 0.12\,\mathrm{m}$ and $\pm 0.11\,\mathrm{m/s}$ for distance and speed, respectively, but how this error is distributed is not known. In [11], a radar is used to measure distance and relative speed to the vehicle ahead. The authors list a range and a range-rate accuracy of $\pm 0.5\,\mathrm{m}$ and $\pm 0.12\,\mathrm{m/s}$, respectively, but again the distribution of the error is not specified. Mathematical models are not only required in academic research, but also by the automotive industry for system prototyping [12].

Needless to say, sensors play a crucial role in the development of future cooperative driving applications, as they complement the view of the surrounding environment obtained through wireless communication. As an example, sensor data is fundamental to properly implement cooperative maneuvers [13] or to measure the distance to vehicles ahead in automatic emergency braking applications [14]. Faults or measurement errors can have a dramatic impact on the performance.

Following these motivations, we perform a stochastic analysis of real-world LiDAR traces to understand the nature of the error and to develop a model for the community. We perform the analysis for a specific LiDAR model. The derivation of a generic model is out of the scope of this paper, and would be meaningless without a standardization process defining the minimum performance requirements for safety applications. As an example, vendor-independent bit error rate models for IEEE 802.11 exist because the standard, together with MAC and PHY specifications, defines the requirements to be WiFi certified.

Dr. Segata was partially supported by the University of Trento within the framework of young researcher support (Bando Giovani Ricercatori 2018).

[1] https://tinyurl.com/bosch-lrr4, visited the 23rd of August, 2018.

978-1-5386-9428-2/18/$31.00 ©2018 IEEE

The contributions of this paper are the following:

- We analyze real-world LiDAR traces to understand the underlying error model;
- On top of our analysis, we build a stochastic error model that is capable of reproducing measurement errors; and
- We implement the model within the PLEXE simulation framework and show the impact on a set of control algorithms.

II. Dataset Description

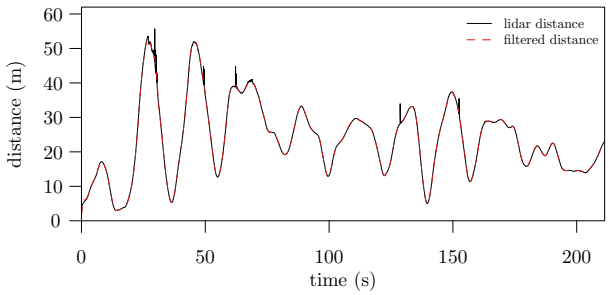
In this section we briefly describe the dataset used to derive the stochastic error model. The dataset is the result of a set of experiments performed in [15]. The paper proposes two autonomous control systems that aims at dampening traffic stop-and-go waves (a.k.a. shock waves). The control systems are implemented on the Cognitive and Autonomous Test (CAT) vehicle developed by University of Arizona and their effectiveness is proven in a set of experiments performed on a circular test track. The experiments are conducted using 22 vehicles, and only one of those (the CAT vehicle) actively dampens stop-and-go waves via the proposed control algorithms, showing the huge benefit that a single, intelligent vehicle can provide to the overall traffic flow.

The autonomous controllers clearly require distance measurements to the car ahead to compute a proper control action. To this purpose, the CAT vehicle mounts a SICK LMS 2D LiDAR. The LiDAR horizontally scans the environment 75 times per second producing a cloud of distance points, where each point is associated with an angle. In the experiments, the authors used the minimum distance point along the trajectory as an estimate of the distance to the front vehicle, and filtered the time series using a Kalman filter to remove the noise. In this paper we study the filtered data, assuming to work with an automotive LiDAR product that already provides filtered, but still imperfect samples. Analyzing the stochastic properties of the raw cloud point samples is left as a future work.

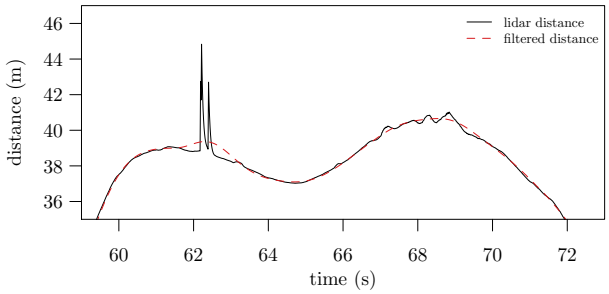
The data collected during the experiment is freely available for download[2]. The dataset includes the data from three experiments, namely experiment A, B, and C. The autonomous control strategies have been tested only in experiments A and C, while in experiment B one of the proposed strategies was performed by a trained human driver without the help of automation. For this reason, only the traces of experiments A and C include LiDAR data that we can use for our purpose.

In addition to the LiDAR traces, the dataset includes data collected from the On-Board Diagnostics (OBD) port, as well as the position and the speed of the vehicles estimated using a 360-degree camera positioned in the center of the circle. By using image processing and clustering techniques, the authors can estimate the position, the speed, and the acceleration of each vehicle on the ring [16]. We decided not to use the distance estimated using the camera as the ground truth, as this also intrinsically includes measurement errors, which could falsify our analysis.

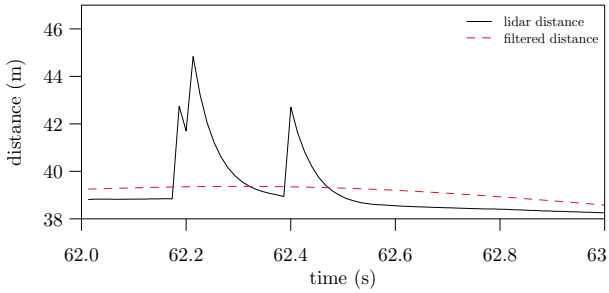
[2] https://doi.org/10.15695/vudata.cee.1



(a) whole trace



(b) detailed view on two different errors



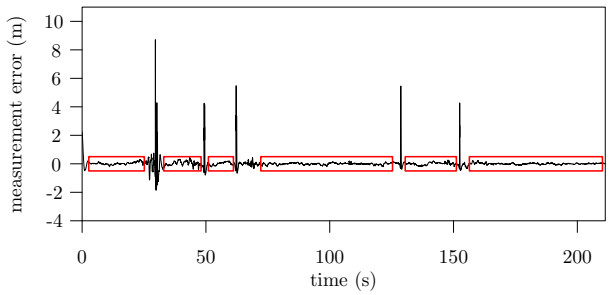
(c) detailed view on shot-noise errors

Figure 1: LiDAR estimated distance and Butterworth filtered distance traces for experiment C.

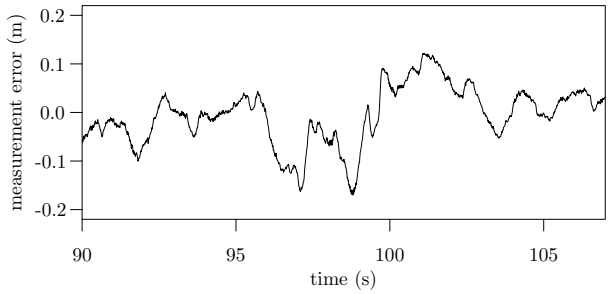
For this reason we use a fifth-order Butterworth low-pass filter with a cutoff frequency of 1 Hz to clean the trace. After filtering the trace we compensate for the delay introduced by the filter to obtain our ground truth. This processing choice is clearly non-optimal, but in the absence of a real ground truth it can still provide some insights on the error process.

III. Traces Analysis

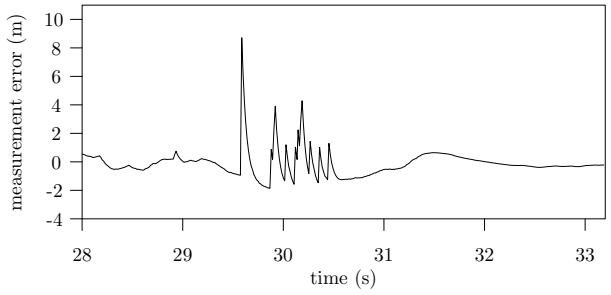
We analyze the estimated LiDAR distance traces recorded during the experiments in [15]. Figure 1 shows the LiDAR distance and the filtered distance traces for the CAT vehicle in experiment C. Figure 1a shows the entire trace. It is not possible to distinguish between the original and the filtered traces, but some errors in the form of spikes can clearly be seen. Figure 1b shows a detailed view on a portion of the trace, where it is possible to understand how the filter acts on the signal. The filter smoothing on spikes is not ideal due to the large error magnitude, while it properly cleans the smaller errors between 66 s and 70 s. Finally, Fig. 1c shows the characteristic shape of spike errors, i.e., a large positive peak error followed by an



(a) whole trace

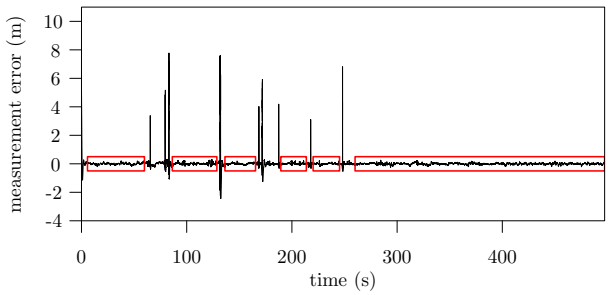


(b) detailed view on a shot-free portion

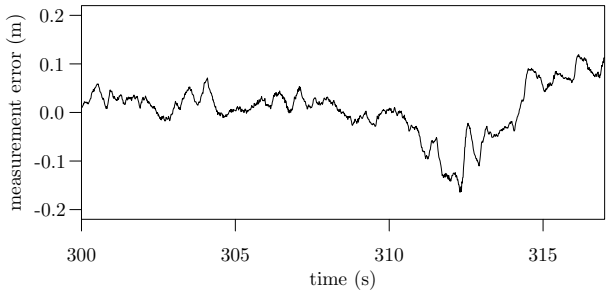


(c) detailed view of shot errors

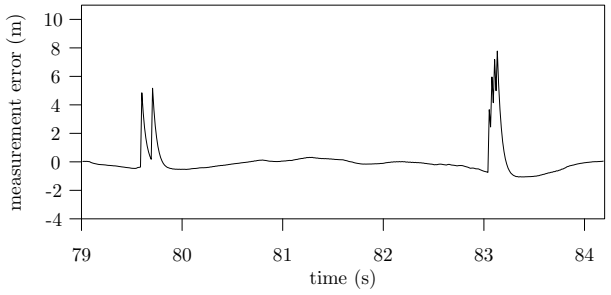
Figure 2: Measurement error trace for experiment C. The highlighted areas in (a) indicates the portions used to perform model estimation of the shot-free error.



(a) whole trace



(b) detailed view on a shot-free portion



(c) detailed view of shot errors

Figure 3: Measurement error trace for experiment A. The highlighted areas in (a) indicates the portions used to perform model estimation of the shot-free error.

exponential decay. This kind of error is known as exponential shot-noise [17]–[19]. Shot-noise peaks are present in the raw LiDAR trace, while the exponential decay is caused by the Kalman filter.

By subtracting the filtered trace from the original one, we obtain an estimate of the measurement error (Figs. 2 and 3). Observing the whole trace (Figs. 2a and 3a) it is possible to clearly spot shot-noise errors, but it is nearly impossible to understand the nature of the error. Observing both a shot (Figs. 2c and 3c) and a shot-free (Figs. 2b and 3b) portion in detail we can start drawing some assumptions. We believe the error is split into two components. Measurement errors in shot-free portions are driven by a correlated stochastic process, while shot-noise errors seem to occur randomly and to quickly fade away. In fact, shot-noise events are often modeled using a homogeneous Poisson point process [18] regardless of the application field.

We thus assume that the error process is of the form

$$\epsilon[k] = \epsilon_c[k] + \epsilon_s[k], \tag{1}$$

where k is the sample index, while ϵ_c and ϵ_s are the processes driving the correlated and the shot errors, respectively.

A. Correlated Noise Estimation

We assume the correlated error to be a first-order autoregressive process (AR(1)) of the form

$$\epsilon_c[k] = \rho\epsilon_c[k-1] + N_c[k], \tag{2}$$

where ρ is the correlation coefficient and N_c the innovation process of the error.

To characterize ϵ_c, we assume N_c has zero mean and we estimate ρ by computing the autocorrelation with a lag of 1 sample on shot-free portions of the error. The considered portions are highlighted by boxes in Figs. 2a and 3a. The autocorrelation for each portion i of ϵ is computed as

$$\rho_i = \frac{\frac{1}{|\epsilon_i|-1}\sum_{k=1}^{|\epsilon_i|-1}\epsilon_i[k]\cdot\epsilon_i[k-1]}{\frac{1}{|\epsilon_i|}\sum_{k=0}^{|\epsilon_i|-1}\epsilon_i[k]^2} \tag{3}$$

Experiment	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6
A	0.997	0.991	0.991	0.983	0.991	0.996
C	0.997	0.993	0.998	0.995	0.996	0.995

Table I: Estimated values of ρ.

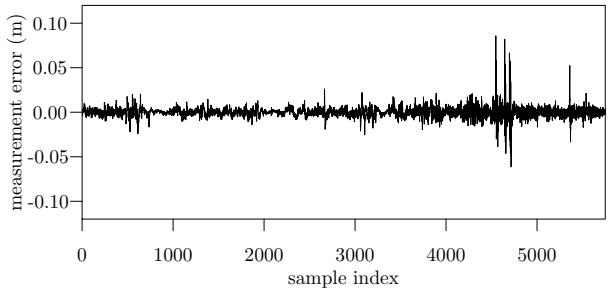


Figure 4: N_c obtained after applying decorrelation (Eq. (4)) on a subset of the samples.

Table I shows the estimated values of ρ for the correlated portions of the trace: they clearly indicate a high correlation. We compute the estimated innovation process N_c by computing

$$N_c[k] = \epsilon_c[k+1] - \bar{\rho}\epsilon_c[k], \quad k = 0, \ldots, |\epsilon_c| - 1 \tag{4}$$

where $\bar{\rho}$ is the average of the values in Table I. We clearly perform this computation only on the portions of the signal highlighted in Figs. 2a and 3a.

Figure 4 shows a subset of the samples in N_c. Although most of the correlation is removed, there is some residual correlation. This suggests that the order of the autoregressive model might be higher than one. In this preliminary work, however, we maintain an AR(1) process as the underlying error model and perform a distribution fitting on N_c. In particular, we use a maximum-likelihood parameter estimation on different distributions and choose the one that maximizes the log-likelihood.

Distributions defined on the whole real domain poorly fitted the dataset. For this reason, we also perform fitting on the absolute value of the samples in N_c. The N_c process can then be generated by multiplying the samples drawn from the fitted distribution by $\mathbf{B} \cdot 2 - 1$, where $\mathbf{B}$ is Bernoulli distributed with parameter $p = 0.5$.

The distribution with the highest log-likelihood resulted to be a generalized Pareto distribution with location $\mu = 0$, scale $\sigma = 0.0036$, and shape $\xi = 0.0913$. Our final autoregressive process is thus

$$\begin{aligned} N_c[k+1] = 0.9936 N_c[k] + \\ \mathbf{GP}(\mu = 0, \sigma = 0.0036, \xi = 0.0913) \cdot \\ (\mathbf{B}(p = 0.5) \cdot 2 - 1), \end{aligned} \tag{5}$$

where **GP** and **B** indicate two random number generators according to a generalized Pareto and a Bernoulli distribution, respectively. As the fitting has been performed with Matlab, we report here the probability density function of the generalized Pareto distribution for $\xi > 0$ and $x > \mu$:

$$f(x|\mu, \sigma, \xi) = \frac{1}{\sigma}\left(1 + \xi\frac{x-\mu}{\sigma}\right)^{-1-\frac{1}{\xi}}. \tag{6}$$

To generate generalized Pareto distributed samples, we compute the Cumulative Density Function (CDF) ($F(x)$) and use the inverse CDF method on its inverse ($F^{-1}(x)$):

$$F(x|\mu = 0, \sigma > 0, \xi > 0) = 1 - \left(1 + \frac{\xi x}{\sigma}\right)^{-\frac{1}{\xi}} \tag{7}$$

$$F^{-1}(x) = -\frac{\sigma\left((1-x)^{\xi} - 1\right)}{(1-x)^{\xi}\,\xi}. \tag{8}$$

B. Shot-noise Estimation

The next step is to estimate shot-noise. The number of shots in the dataset is limited, so a proper fitting is not possible. The contribution of this paper is the framework used to estimate the nature of shot-noise. In this preliminary work we estimate the interarrival time of the homogeneous Poisson process, the exponential decay parameter, and the amplitude of the shots for the limited amount of samples we have. By using data points from additional measurement campaigns, we can apply the same method and obtain a more statistically confident estimation.

The Poisson distribution is defined as

$$\mathbf{P}(N = n, \Lambda) = \frac{\Lambda^n}{n!}e^{-\Lambda}. \tag{9}$$

The process computes the probability that the value of the random variable N is n. Λ is the expected value of the random value and it is defined as $\Lambda = \nu\lambda$, where ν is the time span and λ is the average number of occurrences per time unit.

To estimate λ, we simply count the number of shot-noise points in our trace and divide it by the time duration of the trace. Given the nature of shot-noise, we can easily identify them using the derivative of the trace. In particular, we extract the index of shot-noise points using the following set definition:

$$\{k \mid \epsilon_s[k] - \epsilon_s[k-1] > 1\}. \tag{10}$$

Using the traces from experiments A and C, we count 52 samples over 686 s, obtaining $\lambda = 0.0758$. Given that our LiDAR has a sampling rate of 75 Hz, we can compute

$$\Lambda = \nu\lambda = \frac{0.0758}{75} \simeq 0.001. \tag{11}$$

This means that, on average, we will have one shot error every 1000 samples. It is worth noticing that this estimation is not completely correct. Figure 2c clearly shows that shot errors sometimes appear in bursts. A proper way to study them would be to first compute the interarrival time for bursts, and then study the characteristics of each single burst. Given the limited amount of samples, however, the results would be statistically meaningless, so we leave this task for future work.

To estimate the exponential decay, we consider three points of a shot noise. The value of the shot s_k where the shot starts, and the values of the signal before and after the start (s_{k-1} and

23.408	23.560	23.973	23.358	23.913	24.106	22.715

Table II: Estimated values of τ.

s_{k+1}, respectively). The value s_{k-1} is used as the base value, i.e., the value on top of which the shot noise is built. This is assumed to be constant over the three samples. Although this is not exactly true, we can assume that this value does not change significantly in the time span of three samples ($2.7\,\mathrm{ms}$). The value s_{k+1}, instead, is used to estimate the exponential decay parameter τ.

To compute τ we can use the following system of equations:

$$\begin{cases} s_k = s_{k-1} + N_0 \\ s_{k+1} = N_0 e^{-\frac{\tau}{75}} + s_{k-1}, \end{cases}$$

where N_0 is the amplitude of the shot noise. Solving for N_0 and τ we obtain

$$\tau = -\ln\left(\frac{s_{k+1} - s_{k-1}}{s_k - s_{k-1}}\right) \cdot 75 \tag{12}$$

We take a subset of 7 isolated shot noise points, i.e., the ones that do not belong to a burst (Fig. 2c) and use Eq. (12) to compute τ. Table II shows the values obtained. We take their average as the estimated value of the exponential decay, i.e., $\tau = 23.576$.

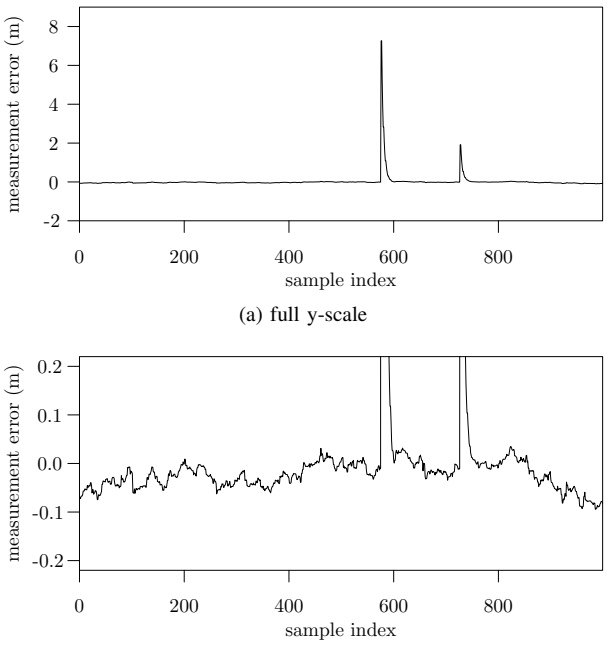
Finally, we need to estimate the amplitude of shot noise. We can obtain shot noise amplitude for isolated shots in a trivial manner, i.e., simply computing $s_k - s_{k-1}$. During bursts, however, this task becomes non trivial, as each shot is the sum of multiple shot noises plus the time varying correlated noise. While it is possible to estimate the decay of each shot, it is not possible to know the value of the correlated noise in that point. For this reason, we only consider isolated shot noise samples and the ones at the beginning of a burst.

In the dataset there are 16 points matching this criterion and their average amplitude is 4.364. These points are clearly not enough to draw any conclusion on the distribution of the amplitudes. In here, we make the strong assumption that the amplitude is exponentially distributed with a mean of 4.364.

The Poisson process is time-continuous, so in the sampled process we consider there is a non negligible probability that more than one shot arrives between one sample and the next, accumulating the error on this latter sample. Thus the amplitude of the shot noise on a sample can be modeled with an Erlang-n distribution, where n is the number of shots in the sampling interval, as each shot has an exponentially distributed amplitude. By putting together the interarrival distribution, the decay exponent, and the amplitude distribution, we can construct the shot-noise process as

$$\epsilon_s[k+1] = \mathbf{E}(n = \textbf{Pois}(\Lambda = 0.001), \mu = 4.364) + \epsilon_s[k] e^{-\frac{23.576}{75}}. \tag{13}$$

In Eq. (13), **E** indicates the generator of Erlang-distributed samples, i.e., the sum of n independent exponential distributions with mean μ. The **Pois** random variable extracts the number of shot events occurring at sample k. The Erlang distribution is defined when n is a positive integer; with a little abuse of notation, we set $\mathbf{E}(n = 0, \mu) = 0$.



(a) full y-scale

(b) detailed view on correlated error

Figure 5: Synthetically-generated trace.

As a final remark, it should be noted that the processes defined in this section are valid for a sampling rate of $75\,\mathrm{Hz}$. When changing the sampling rate, the parameters of the distributions should be scaled accordingly.

Figure 5 shows a synthetically-generated trace using the model designed in this section. The figure shows that the model properly handles correlation and shot-noise. The correlated error process in Fig. 5a, however, seems to lead to smaller errors compared to the original process (Figs. 2a and 3a). This might be caused by the low order of the autoregressive process, indicating the need for further future investigation.

IV. Impact on Cooperative Driving Simulations

We implement the model designed in Section III within the Plexe simulator [20]. Plexe is a cooperative driving simulator with a special focus on platooning that features realistic vehicular networking models, as well as realistic vehicle dynamics and platooning control algorithms. The simulator does not include error models either for sensors, or for GPS, and studies using this simulator assume measurements to be error-free [21], [22].

We run two set of simulations using three different control algorithms. In the first set we inject a platoon of 8 cars with the leader following a constant speed profile, i.e., vehicle dynamics are not disturbed. The inter-vehicle distance is set to the steady-state distance of the chosen control algorithm. Ideally, the distance should remain constant throughout the whole simulation, so the outcome will show the impact of the model. In the second set, instead, the leader follows a sinusoidal profile.

	Parameter	Value
mobility	Leader's average speed	100 km/h
	Oscillation frequency	0.2 Hz
	Oscillation amplitude	$\simeq$ 95 to 105 km/h
	Platoon size	8 cars
	Car length	4 m
	Simulation sampling rate	100 Hz
	LiDAR sampling rate	75 Hz
controllers	Engine lag τ	0.5 s
	Stand-still distance d_{st}	2 m
	ACC λ	0.1
	ACC headway time T	0.3 s (10.33 m @ 100 km/h)
	PATH weighting factor C_1	0.5
	PATH bandwidth ω_n	0.2
	PATH damping factor ξ	1
	PATH constant distance d	5 m
	Ploeg headway time T	0.5 s (15.89 m @ 100 km/h)
	Ploeg k_p	0.2
	Ploeg k_d	0.7

Table III: Network and road traffic simulation parameters.

We consider three control algorithms, namely a standard Adaptive Cruise Control (ACC), and two Cooperative Adaptive Cruise Control (CACC) algorithms. All control algorithms implement automated car following, i.e., given contextual information such as distance to the front vehicle, speed and/or acceleration of other vehicles, they compute the acceleration required to reach a certain target, which in this case is simply to maintain a predefined distance. The required acceleration (or control input u_i for vehicle i) is implemented by the engine/braking system with a certain delay. This delay is commonly modeled using a first order lag described by the following differential equation [20], [23], [24]

$$\frac{\mathrm{d}a_i(t)}{\mathrm{d}t} = \frac{1}{\tau}\left(u_i(t) - a_i(t)\right). \tag{14}$$

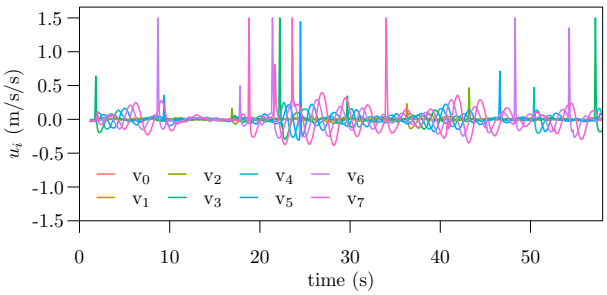
In Eq. (14), $a_i(t)$ and $u_i(t)$ are the current and the desired acceleration of vehicle i, respectively, while τ is the time constant of the delay which is set to $0.5\,\mathrm{s}$.

The difference between the three algorithms is in the design and the performance. The ACC algorithm is defined in [23] and its implementation details can be found in [20]. It follows a constant time-headway spacing policy and uses a radar/LiDAR system to estimate the distance and the relative speed to the vehicle ahead. Its steady-state distance is defined as

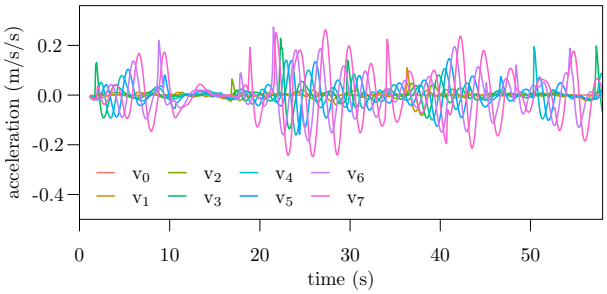
$$d = T \cdot v + d_{\text{st}}, \tag{15}$$

where T is the time headway in s, v is the cruising speed in $\mathrm{m/s}$, and d_{st} is the stand-still distance (i.e., when speed is zero) in m. The chosen headway time T is $0.3\,\mathrm{s}$. This choice of the parameter makes the ACC string-unstable, as the required condition for string stability of this control system is $T \geq 2\tau$ [23]. The reason to have the system in unstable conditions is to see how it reacts to LiDAR-induced perturbation at steady state.

The two CACC algorithms we consider are the PATH's CACC [25] and the Ploeg's CACC [24]. PATH's CACC follows a constant distance spacing policy, i.e., the distance does not depend on the cruising speed as in Eq. (15). This is made possible by the use of Inter-Vehicle Communication (IVC).



(a) control input u_i



(b) acceleration

Figure 6: Impact of the error model on the control dynamics of an unstable ACC in steady-state conditions.

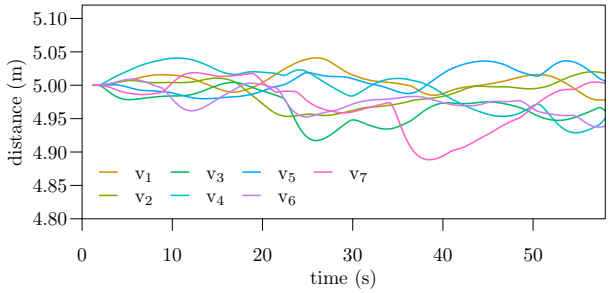
Through IVC, vehicles exchange information such as current acceleration and speed, making it possible to improve system's reaction time and thus safely reduce the distance. In particular, PATH's CACC exploits leader and front vehicle information to compute the control action. The constant inter-vehicle distance is set to $5\,\mathrm{m}$. Ploeg's CACC, instead, exploits a constant time-headway spacing policy as the ACC. Differently from the latter, the time headway T is much smaller, as the control system exploits the desired acceleration of the front vehicle as a feed-forward term. For Ploeg's CACC, T is set to $0.5\,\mathrm{s}$. Differently from the ACC, this control algorithm is string-stable even for such a small headway time.

All the algorithms exploit a radar or a LiDAR system to estimate distance and relative speed to the front vehicle. In this preliminary work, we assume the relative speed to be correct, and we focus our attention on distance errors.

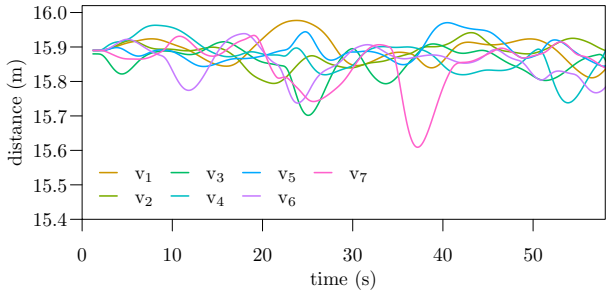
With respect to the dynamics, we assume the control system to run at $100\,\mathrm{Hz}$, while the LiDAR runs at $75\,\mathrm{Hz}$. This simply means that every $40\,\mathrm{ms}$ one LiDAR sample is kept constant, as the control loop requires values faster than the rate the sensor can produce them. With respect to communication, instead, we assume vehicles to send wireless beacons including control data with a rate of $10\,\mathrm{Hz}$, although with no packet losses. Table III summarizes the remaining simulation parameters.

A. Steady-state Simulations

Figure 6 shows the desired acceleration u_i and the actual acceleration over time for the 8 vehicles in the simulation using a non-cooperative ACC. From the control input plot (Fig. 6a) it is possible to spot positive acceleration spikes caused by LiDAR shot-errors. These errors introduce perturbations in the system which, due to the unstable configuration, tend to

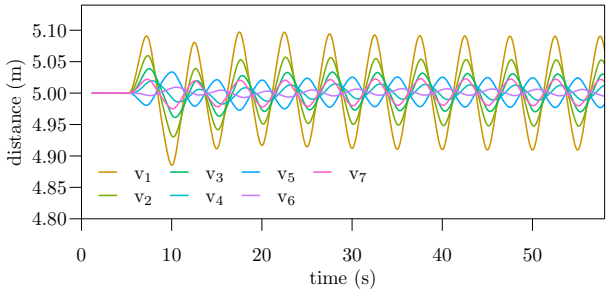


(a) PATH's CACC

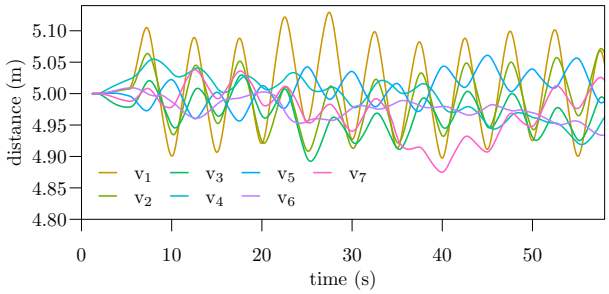


(b) Ploeg's CACC

Figure 7: Impact of the error model on the dynamics of two different CACCs in steady-state conditions. v_i indicates the bumper to bumper distance of vehicle i to vehicle $i-1$.



(a) w/o error model



(b) w/ error model

Figure 8: Impact of the error model on the dynamics of the PATH's CACC under sinusoidal disturbance. v_i indicates the bumper to bumper distance of vehicle i to vehicle $i-1$.

be amplified by the following vehicles. This can be seen in particular by observing Fig. 6b, which shows that, close to shot-errors, there are some sinusoids that gets amplified by vehicles at the tail of the platoon.

Figure 7 shows the actual inter-vehicle distance between the vehicles in the platoon for the PATH and the Ploeg CACCs. The imperfect LiDAR introduces some disturbance, causing the inter-vehicle distance to float around the steady-state value. The absolute error is in the order of 10-30 cm, so the impact is limited. Yet, the shot LiDAR errors, which are always positive, do not permit to the control system to stabilize the distance around the target value.

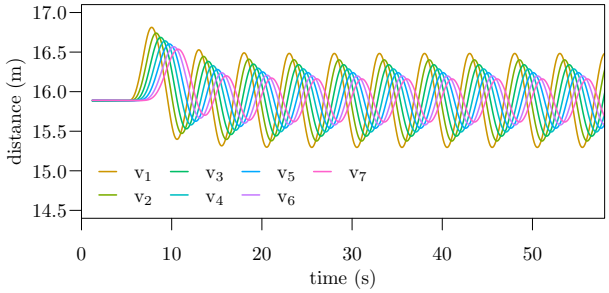
B. Sinusoidal Simulations

Figure 8 shows the results for the PATH's CACC for the sinusoidal scenario, without and with the LiDAR error model. Without the error model (Fig. 8a) the system properly dampens the oscillations down the stream of vehicles, showing the classic string-stable behavior. The error model (Fig. 8b) introduces further disturbance, making it difficult to observe a clear smoothing of the oscillations. Still, the system is pretty robust to the errors introduced by the LiDAR, as the maximum tracking error is in the order of 10 cm.

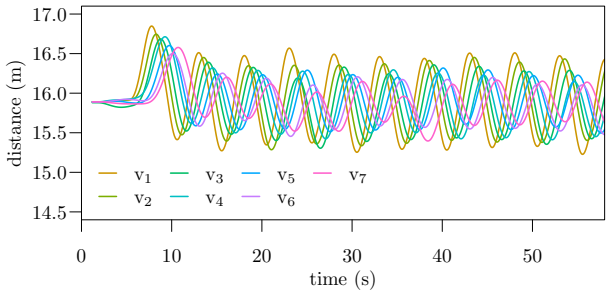
Similar conclusions hold for the Ploeg's CACC (Fig. 9). The LiDAR error model disturbs the ideal string-stable behavior of Fig. 9a, but the impact is again limited. In particular, the system still properly dampens the oscillations with only minor errors.

V. Concluding Discussion and Future Work

In this paper we analyzed a set of LiDAR distance traces recorded during real-world experiments. After filtering the



(a) w/o error model



(b) w/ error model

Figure 9: Impact of the error model on the dynamics of the Ploeg's CACC under sinusoidal disturbance. v_i indicates the bumper to bumper distance of vehicle i to vehicle $i-1$.

traces to obtain the ground truth, we analyzed the stochastic properties of the error. We have shown that the error is composed of a correlated and a shot-noise process and we separately analyzed the two components, deriving stochastic properties and parameters for their synthetic generation. Finally, we implemented the model within the Plexe simulation

framework showing its impact on three different control algorithms and two driving scenarios. In both the steady-state and the sinusoidal scenario the impact of the model is limited. However, the model can definitely help in spotting instabilities in control systems, as we have shown for the ACC in the steady-state scenario.

The model presented in this work is preliminary and we made some simplifying assumptions, so the impact of the model might be underestimated. In our future work, we plan to improve the model and get rid of these assumptions, which we summarize here.

First, the lack of a ground truth is a major problem. The use of a filter to obtain such a ground truth injects the dynamics of the filter in the error model as, for example, the time correlation.

Second, we performed our analysis on the Kalman-filtered points. While this is reasonable, filtering introduces delay which cannot be analyzed in the absence of a synchronized ground truth. Delays have huge impact on the performance of control systems, so proper modeling is fundamental.

Third, we assumed a first order autoregressive model for the correlated component of the error model. After removing the correlation, the dataset still showed some residual correlation which we did not account for. As a result, the synthetically-generated traces have smaller absolute errors.

Fourth, the relative speed has been assumed to be perfectly know. In reality, the relative speed is the derivative in time of the distance, and the derivative is highly influenced by errors. How to properly compute and filter the relative speed to be included in the model is yet an open issue.

Finally, we considered shot-noise peaks to be independent, while in the dataset they occur in bursts. As shot-noise amplitude is relatively large, the occurrence of bursts can lead to a large overestimation of the actual distance and might cause a wrong control decision.

Addressing all these open points is fundamental to obtain a trustworthy stochastic model, not only for LiDARs, but for all automotive sensors.

REFERENCES

[1] H. Stübing, M. Bechler, D. Heussner, T. May, I. Radusch, H. Rechner, and P. Vogel, "Sim TD: A Car-to-X System Architecture for Field Operational Tests," *IEEE Communications Magazine*, vol. 48, no. 5, pp. 148–154, May 2010.

[2] R. Stahlmann, A. Festag, A. Tomatis, I. Radusch, and F. Fischer, "Starting European Field Tests for Car-2-X Communication: the DRIVE C2X Framework," in *18th ITS World Congress and Exhibition*, Orlando, FL, October 2011.

[3] B. Cheng, A. Rostami, and M. Gruteser, "Experience: Accurate Simulation of Dense Scenarios with Hundreds of Vehicular Transmitters," in *22nd ACM International Conf. on Mobile Computing and Networking (MobiCom 2016)*, New York, NY, October 2016, pp. 271–279.

[4] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Trans. on Mobile Computing*, vol. 10, no. 1, pp. 3–15, January 2011.

[5] B. Schünemann, "V2X Simulation Runtime Infrastructure VSimRTI: An Assessment Tool to Design Smart Traffic Management Systems," *Elsevier Computer Networks*, vol. 55, no. 14, pp. 3189–3198, October 2011.

[6] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent Development and Applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, December 2012.

[7] M. Fellendorf and P. Vortisch, "Microscopic Traffic Flow Simulator VISSIM," in *Fundamentals of Traffic Simulation*, ser. International Series in Operations Research & Management Science, J. Barceló, Ed., 2010, vol. 145, pp. 63–93.

[8] F. Hagenauer, F. Dressler, and C. Sommer, "A Simulator for Heterogeneous Vehicular Networks," in *6th IEEE Vehicular Networking Conf. (VNC 2014), Poster Session*, Paderborn, Germany, December 2014, pp. 185–186.

[9] C. Sommer, S. Joerer, M. Segata, O. K. Tonguz, R. Lo Cigno, and F. Dressler, "How Shadowing Hurts Vehicular Communications and How Dynamic Beaconing Can Help," *IEEE Trans. on Mobile Computing*, vol. 14, no. 7, pp. 1411–1421, July 2015.

[10] T. Mangel, O. Klemp, and H. Hartenstein, "A Validated 5.9 GHz Non-Line-of-Sight Path-Loss and Fading Model for Inter-Vehicle Communication," in *11th International Conf. on ITS Telecommunications (ITST 2011)*, St. Petersburg, Russia, August 2011, pp. 75–80.

[11] B. Alrifaee, M. Reiter, J. P. Maschuw, F. Christen, L. Eckstein, and D. Abel, "Satellite- and Map-based Long Range Cooperative Adaptive Cruise Control System for Road Vehicles," *IFAC Proceedings Volumes*, vol. 46, no. 21, pp. 732–737, September 2013.

[12] R. Rasshofer and K. Gresser, "Automotive Radar and Lidar Systems for Next Generation Driver Assistance Functions," *Advances in Radio Science*, vol. 3, pp. 205–209, May 2005.

[13] M. Segata, B. Bloessl, S. Joerer, F. Dressler, and R. Lo Cigno, "Supporting Platooning Maneuvers through IVC: An Initial Protocol Analysis for the Join Maneuver," in *11th IEEE/IFIP Conf. on Wireless On demand Network Systems and Services (WONS 2014)*, Obergurgl, Austria, April 2014, pp. 130–137.

[14] M. Segata and R. Lo Cigno, "Automatic Emergency Braking - Realistic Analysis of Car Dynamics and Network Performance," *IEEE Trans. on Vehicular Technology*, vol. 62, no. 9, pp. 4150–4161, October 2013.

[15] R. E. Stern, S. Cui, M. L. Delle Monache, R. Bhadani, M. Bunting, M. Churchill, N. Hamilton, H. Pohlmann, F. Wu, B. Piccoli, B. Seibold, J. Sprinkle, and D. B. Work, "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments," *Transportation Research Part C: Emerging Technologies*, vol. 89, pp. 205–221, April 2018.

[16] F. Wu, R. Stern, M. Churchill, M. L. Delle Monache, K. Han, B. Piccoli, and D. Work, "Measuring trajectories and fuel consumption in oscillatory traffic: Experimental results," in *Transportation Research Board 96th Annual Meeting (TRB 2017)*, Washington, DC, January 2017.

[17] M. Brigham and A. Destexhe, "Nonstationary filtered shot-noise processes and applications to neuronal membranes," *Physical Review E*, vol. 91, no. 6, June 2015.

[18] P. Ilhe, É. Moulines, F. Roueff, and A. Souloumiac, "Nonparametric estimation of mark's distribution of an exponential Shot-noise process," *Electronic journal of statistics*, vol. 9, no. 2, pp. 3098–3123, June 2015.

[19] L. Wolff and B. Lindner, "Method to calculate the moments of the membrane voltage in a model neuron driven by multiplicative filtered shot noise," *Physical Review E*, vol. 77, no. 4, April 2008.

[20] M. Segata, S. Joerer, B. Bloessl, C. Sommer, F. Dressler, and R. Lo Cigno, "PLEXE: A Platooning Extension for Veins," in *6th IEEE Vehicular Networking Conf. (VNC 2014)*, Paderborn, Germany, Dec. 2014, pp. 53–60.

[21] G. Giordano, M. Segata, F. Blanchini, and R. Lo Cigno, "A Joint Network/Control Design for Cooperative Automatic Driving," in *9th IEEE Vehicular Networking Conf. (VNC 2017)*, Torino, Italy, November 2017, pp. 167–174.

[22] S. Santini, A. Salvi, A. S. Valente, A. Pescapè, M. Segata, and R. Lo Cigno, "A Consensus-based Approach for Platooning with Inter-Vehicular Communications and its Validation in Realistic Scenarios," *IEEE Trans. on Vehicular Technology*, vol. 66, no. 3, pp. 1985–1999, Mar. 2017.

[23] R. Rajamani, *Vehicle Dynamics and Control*, 2nd ed., 2012.

[24] J. Ploeg, B. Scheepers, E. van Nunen, N. van de Wouw, and H. Nijmeijer, "Design and Experimental Evaluation of Cooperative Adaptive Cruise Control," in *IEEE International Conf. on Intelligent Transportation Systems (ITSC 2011)*, Washington, DC, Oct. 2011, pp. 260–265.

[25] R. Rajamani, H.-S. Tan, B. K. Law, and W.-B. Zhang, "Demonstration of Integrated Longitudinal and Lateral Control for the Operation of Automated Vehicles in Platoons," *IEEE Trans. on Control Systems Technology*, vol. 8, no. 4, pp. 695–708, July 2000.