Proceedings of the 12th International IEEE Conference on Intelligent Transportation Systems, St. Louis, MO, USA, October 3-7, 2009

Kalman Filter Process Models for Urban Vehicle Tracking

Carlos Aydos, Bernhard Hengst, William Uther

Abstract— **Faced with increasing congestion on urban roads, authorities need better real-time traffic information to manage traffic. Kalman Filters are efficient algorithms that can be adapted to track vehicles in urban traffic given noisy sensor data. A Kalman Filter process model that approximates dynamic vehicle behaviour is a reusable subsystem for modelling the dynamics of a multi-vehicle traffic system. The challenge is choosing an appropriate process model that produces the smallest estimation errors. This paper provides a comparative analysis and evaluation of Linear and Unscented Kalman Filters process models for urban traffic applications.**

I. Introduction

The Roads and Traffic Authority of New South Wales (RTA) and NICTA are collaborating on a research and development project to estimate urban road traffic in real-time given sparse and noisy sensor data. This can be achieved via a combination of simulation and sensor data. Better real-time estimates of the traffic state support improved traffic control, incident management, travel advisory services and the management of environmental impact. One of the key building blocks of this estimation system is a subsystem that is able to track individual vehicles.

Bayesian filtering is a principled method for state estimation. It tracks state variables by recursively integrating system behaviour and sensor data. It requires two prior probabilistic models: a process model to describe the target dynamics and a sensor model to characterise the noisy sensor measurements. An efficient Bayesian filtering algorithm is the Kalman Filter. A Linear Kalman Filter requires both models to be linear. An extension of the Linear Kalman Filter, the Unscented Kalman Filter relaxes the linearity requirement via an approximation of nonlinear models.

In this paper, we are interested in comparing process models that approximate vehicle dynamics in urban environments. These vehicle models are commonly required across a range of traffic estimation related applications. Some examples of urban traffic applications are: vehicle position tracking from noisy data, smoothing of noisy measurements and vehicle/driver behaviour prediction from noisy data. For the applications cite above, the process models of choice are:

- a constant velocity model (aka white noise acceleration model) [1], [2], [3],
- a constant acceleration model (aka white noise jerk model) [4], [5], and
- a nonlinear car-following model [6].

Our objective is to compare results for a selected set of commonly used process models. Sensor models are designed for individual sensor types and generally are not traffic specific. They are outside the scope of this paper. In this paper, we present the vehicle models considered and the corresponding parameter learning methods. Additionally we describe the datasets for parameter learning and model evaluation. Finally, we present results, analysis and our conclusions.

The contribution of this paper is the comparative analysis and evaluation of multiple Kalman Filter process subsystem models for urban traffic estimation applications.

C. Aydos is with NICTA, the Roads and Traffic Authority of New South Wales and the School of Computer Science and Engineering, University of New South Wales. Mailto: Level 4, 223 Anzac Parade, Kensington NSW 2052, Australia. Telephone +61 2 8306 0436, carlos.aydos@nicta.com.au.

B. Hengst is with NICTA, the School of Computer Science and Engineering, University of New South Wales and the ARC Centre of Excellence for Autonomous Systems. bernhardh@cse.unsw.edu.au.

W. Uther is with NICTA and the School of Computer Science and Engineering, University of New South Wales. william.uther@nicta.com.au.

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

The authors wish to thank Robert Bosch GmbH, the Clearing House Transport and the NGSIM program for their datasets and the Roads and Traffic Authority of New South Wales for their collaboration in the Smart Traffic and Roads project.

II. Tracking Subsystem and Bayesian Filtering

Our tracking subsystem estimates vehicle longitudinal distance d_k from an initial point along a single-lane road at discrete time-steps where k is the time index. The distance is measured to the front-end of the vehicle. Distance estimates are based on sparse and noisy measurements z_k. Optionally, information about the position, speed and length of the vehicle ahead (leader) can be used as well. Our two main problems are: the measurements are not available at every time-step, and when they are available, they contain noise.

A naïve distance estimation approach assumes the last available measurement as the estimate:

$$d_k = z_j \tag{1}$$

where $j \leq k$. A better approach is to use a Bayesian Filter. It offers a range of improvements over the naïve approach:

- all measurements up to the current time are taken into account in the estimate,
- a vehicle behaviour model or *process model* is used to predict the next position of the vehicle given its current position. The estimate is the weighted average of the prediction and the measurement. Alternatively, when no measurement is available, the estimation is the prediction, and
- in addition to vehicle position estimates, it can estimate additional variables (e.g. vehicle speed) that are part of a vehicle state x_k.

978-1-4244-5521-8/09/$26.00 ©2009 IEEE

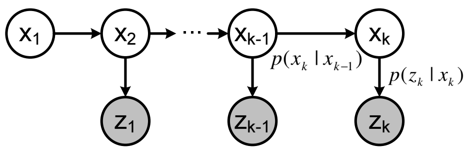


Fig. 1. Bayesian Filter. The white nodes represent the unknown vehicle states or *hidden states* at discrete time intervals. The shaded nodes represent the measurements or *evidence*. The current time is k.

A Bayesian Filter is a special case [7] of a Bayesian Network and can be represented as a graphical model (see Fig. 1). A Bayesian Filter requires two a priori models. A probabilistic process model and probabilistic sensor model that approximate the vehicle dynamics and sensor measurements respectively. The process model $p(x_k \mid x_{k-1})$ describes the one time-step probabilistic prediction of the vehicle state. The sensor model $p(z_k \mid x_k)$ describes the probability of a measurement position given a true vehicle position. As in a Bayesian Network, these models are conditional probability distributions. As a result, the filter produces probabilistic estimates and we can perform different types of inference on the model. In Bayesian filtering, the desired result is $p(x_k \mid z_{1:k}, x_1)$. That is, we would like to know the current vehicle state given an initial state and all the measurements up to the current time. This inference can be performed by integration:

$$p(x_k \mid z_{1:k}, x_1) = \\ p(z_k \mid x_k) \int p(x_k \mid x_{k-1})\, p(x_{k-1} \mid z_{1:k-1}, x_1)\, dx_{k-1} \quad (2)$$

The derivation of this expression can be found in [8, p.543]. However, the calculation of this expression requires a numerical solution and the storage of an infinitely large vector for the parameters describing the resulting distribution. This makes its use impractical. Diard et al. [7] discuss different assumption scenarios which make the solution of (2) feasible. We are interested in one particular scenario which is the Linear Kalman Filter. The Linear Kalman Filter [9] is the exact solution to (2) when both conditional probability distributions $p(x_k \mid x_{k-1})$ and $p(z_k \mid x_k)$ are linear Gaussian models. As we will see later, the Unscented Kalman Filter [10] approximates the solution of (2) when the conditional probability distributions represent nonlinear models.

A. Linear Kalman Filtering

The discrete-time Linear Kalman Filter (LKF) is commonly expressed in state space form [11, p.208]

$$x_k = F_k x_{k-1} + G_k u_{k-1} + v_k \quad (3)$$
$$z_k = H_k x_k + w_k \quad (4)$$

where F_k is the process model matrix, G_k is the exogenous inputs matrix, H_k is sensor model matrix, u_k are the exogenous inputs, v_k is the process model noise with covariance $\mathbb{E}\left[v_k v_k'\right] = Q_k$ and w_k is the sensor model noise with covariance $\mathbb{E}\left[w_k w_k'\right] = R_k$.

The matrices F_k, G_k and Q_k form the parameter set of the process model conditional probability $p(x_k \mid x_{k-1})$. The matrices H_k and R_k form the parameter set of the sensor model conditional probability $p(z_k \mid x_k)$. These can be expressed as

$$p(x_k \mid x_{k-1}) = \mathcal{N}(x_k; F_k x_{k-1} + G_k u_{k-1}, Q_k) \quad (5)$$
$$p(z_k \mid x_k) = \mathcal{N}(z_k; H_k x_k, R_k) \quad (6)$$

where $\mathcal{N}(x; \mu, \Sigma)$ represents a Gaussian probability distribution on x with mean μ and covariance Σ. The solution of the state space equations above is a closed form solution and widely available in the literature [11, p.208].

B. Unscented Kalman Filtering

The Unscented Kalman Filter (UKF) is an extension of the LKF for when either process or sensor models are nonlinear. This nonlinear filtering problem can be expressed in the general state space form:

$$x_k = f(x_{k-1}, u_{k-1}, v_k, k) \quad (7)$$
$$z_k = h(x_k, u_k, w_k, k) \quad (8)$$

The LKF assumptions are not valid here and the nonlinear transformations required cannot be performed like the linear transformations in (5) and (6). The UKF uses instead an *unscented* transformation. For instance, lets assume a certain Gaussian random variable a has a relationship with another random variable b via a function

$$b = fun(a)$$

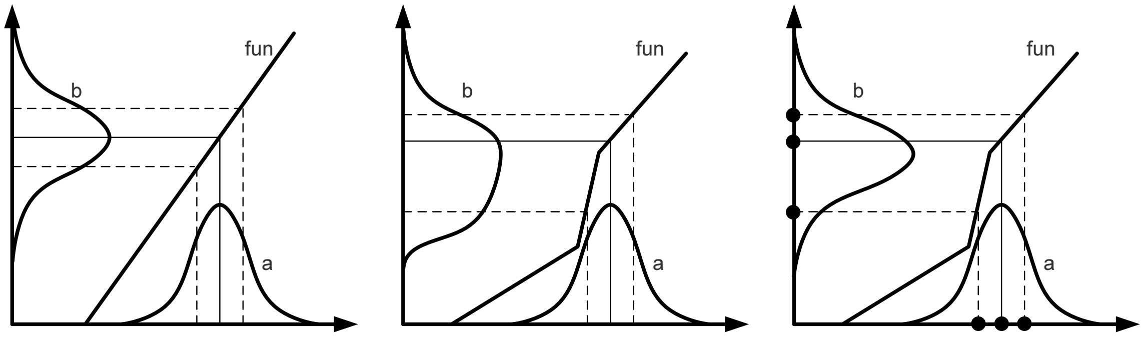
If fun is a linear transformation, the result b is a new Gaussian function (e.g. Fig. 2(a)). If fun is nonlinear such as $f(\cdot)$ or $h(\cdot)$ the result b is a new function with a potentially unbounded number of parameters (e.g. Fig. 2(b)).

The UKF is an approximation method for nonlinear filtering problems. For each nonlinear transformation, the method selects a deterministic sample of *sigma points* from the initial distribution. Subsequently, the nonlinear transformation is applied to each individual sigma point. The transformed set of sigma points can now be used to estimate the parameters of the assumed distribution (e.g. Fig. 2(c)).

Two other obvious choices of nonlinear-capable Bayesian Filters are possible here: the Extended Kalman Filter (EKF) and the Particle Filter (PF). We chose the UKF over the PF for two reasons: the UKF's smaller computation footprint and because a Gaussian vehicle state assumption seems reasonable. We discarded the EKF based on a previous study [12] that claims: "Given its performance and implementation advantages, we conclude that the new filter (UKF) should be preferred over the EKF in virtually all nonlinear estimation applications." A complete description of the UKF algorithm can be found in [10].

III. Datasets

The data we use for learning and evaluating the models originates from three distinct datasets. All three datasets consist of sequences of vehicle positions at $2/3 \approx 0.6667$



(a) Linear Transformation (b) Nonlinear Transformation (c) Unscented Transformation

Fig. 2. Transformation examples

seconds intervals. Each sequence additionally includes the leading vehicle's position, speed and length. The choice of a 0.6667 seconds interval is influenced by a requirement of one of the chosen models as we will see later. The remaining models have no time-step size restriction, and for fair comparison purposes, we carry out all the parameter learning and evaluations on this unique time-step size.

The first dataset derives from the *Lankershim Boulevard* and *Peachtree Street* datasets available and documented at the NGSIM program website [13]. These are vehicle trajectories captured by high-mounted video cameras in an urban environment. The images are post-processed into vehicle trajectories at 0.1 s intervals. For each vehicle, its leading and following vehicles are identified on a lane-by-lane basis. We select the 100 longest (distance) trajectories where follower and leader pairs are identified continuously. We also linearly interpolate the values to match the 0.6667 s intervals. In total there are 45 km (follower driving distance) and 104 minutes of traffic data.

The second dataset was recorded by Robert Bosch GmbH in urban peak afternoon traffic in Stuttgart-Vaihingen, Germany. It contains vehicle speed from the vehicle's own wheel speed sensor and the relative speed and offset to the leading vehicle are measured with a radar. Part of the dataset and details are available at the Clearing House Transportation [14]. The remaining part was obtained directly from Robert Bosch GmbH. This dataset was used by several authors for car-following models studies and to our knowledge it is first referenced in [15]. It comprises 3 sequences totalling 4.7 km and 16 minutes of traffic data.

The third dataset originates from simulation. The model road is 900 m long with traffic signals at 500 m and 850 m from the start. The overall speed limit is 60 km/h, with a speed limit restriction of 40 km/h between 200 m and 300 m. We produce 100 sequences using the PARAMICS traffic microsimulation tool and randomly vary the driver aggression and vehicle class settings. It totals 88 km and 146 minutes of traffic data.

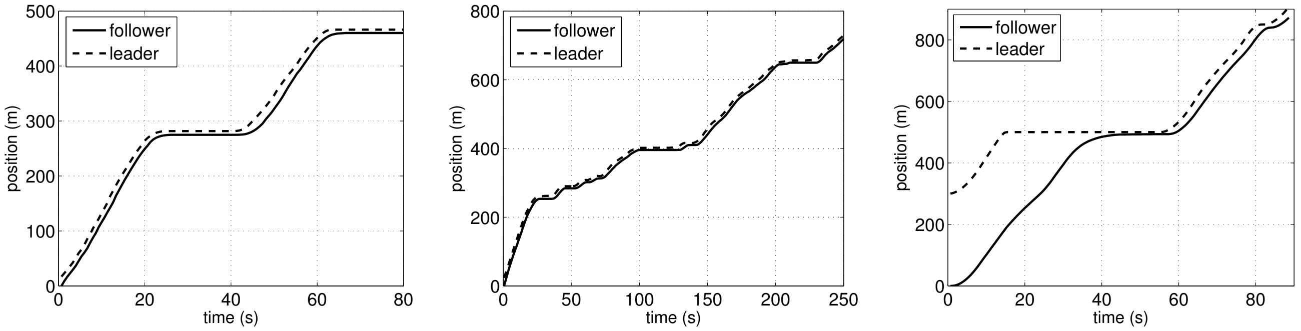
Fig. 3 depicts examples sequence from each dataset.

IV. Evaluation

We use a k-fold cross-validation [8, p.663] method to calibrate and compare the models. For each dataset, a chosen percentage c of the data is randomly selected to calibrate the models and the remainder is used for evaluation. This process repeats k times for each dataset and the evaluation results are aggregated. The values of c and k are displayed in Table I.

In general, practical and mass-deployable traffic sensors have much larger errors than the measuring technique errors used in both our measured datasets. Therefore, we assume vehicles positions in these datasets to contain true vehicle positions for the purpose of this paper.

Sensor models are outside the scope of this paper, but we require noisy sensor data to perform our evaluation. The



(a) NGSIM (b) Bosch (c) Simulation

Fig. 3. Sequence examples for each dataset

TABLE I
VALUES OF c AND k FOR THE k-FOLD CROSS-VALIDATION

Dataset	c	k
NGSIM	90%	10
Bosch	66.66%	3
NGSIM	90%	10

sensor data is simulated by drawing samples from a zero-mean Gaussian distribution and adding them to the true vehicle positions:

$$p(z_k \mid x_k) = \mathcal{N}(x_k, R_k)$$

where $R_k = 5^2$ for all k.

The choice of sensor noise variance 5^2 m is an approximation to that expected from real sensors. This choice has no effect on our comparison results. We tested values of $1, 2^2, 3^2, 4^2, 5^2$ and 6^2 and the model comparison ranking was unchanged across all datasets. Values larger than 6^2 are not considered as they seem very unlikely in practice. For the sake of the comparisons and analysis in this paper we use $R_k = 5^2$.

For each data sequence we simulate several sequences of measurements. For instance, each dataset receives 5 measurement sequences. The first with 100% of the time-stamps having measurements. The second with 75% of the time-stamps having measurements and so forth. More formally, for each vehicle position in each set we simulate a measurement with probability pe with a position drawn from $\mathcal{N}(x_k, R_k)$. The values for pe are 1, 0.75, 0.5, 0.25 and 0.05 for each measurement sequence.

The evaluation consists of applying the filters with corresponding models to the simulated measurements, and if applicable, the position, speed and vehicle length from the leading vehicle. The information about the leader is assumed to be error free, i.e. there are no simulated measurements for the leader data. The result of the sequence is the root mean squared (RMS) error between filter estimation and the assumed ground truth. The RMS error provides an absolute performance indication of the filters [11, p.234].

V. MODELS AND PARAMETER LEARNING

In this section we describe each of the seven models in detail and the parameter learning techniques for each one.

A. Models 1 and 2

The first two models we consider are models frequently used in the traffic tracking and data fusion literature: a constant velocity, and a constant acceleration model. Both models assume vehicle states are approximated by a Gaussian probability distribution and their conditional probability distribution is a linear Gaussian model as in (5). The parameters F_k, G_k and Q_k are the same for all k, and from now on, are simply referred to as F, G and Q. The constant velocity and constant acceleration assumptions result in the pre-definitions of F and G. Both models have $G = 0$ since no external information is available here and F is defined as:

$$F^{cv} = \begin{bmatrix} 1 & \delta t \\ 0 & 1 \end{bmatrix} \quad F^{ca} = \begin{bmatrix} 1 & \delta t & \frac{\delta t^2}{2} \\ 0 & 1 & \delta t \\ 0 & 0 & 1 \end{bmatrix}$$

where $\delta t = 0.6667s$ is the time-step size. The process noise covariance Q must be learned from data and we use the Maximum Likelihood Estimation (MLE) method [16, p.315].

Firstly, we determine the joint probability distribution. Following the assumption in Section IV that the datasets contain ground truth vehicle positions we ignore the measurement nodes from the network represented in Fig. 1 and model each learning data sequence as depicted in Fig. 4.

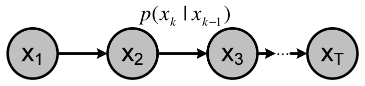


Fig. 4. Bayesian Network with fully known states

The joint probability distribution of this network representing a single sequence is:

$$p(x_1, ..., x_T) = p(x_1) \prod_{k=2}^{T} p(x_k \mid x_{k-1}) \tag{9}$$

where T is the total time interval of the data sequence. Since all our experiments are carried out with known initial conditions we can simplify (9) to:

$$p(x_1, ..., x_T) = \prod_{k=2}^{T} p(x_k \mid x_{k-1}) \tag{10}$$

The joint probability distribution for N sequences is:

$$p\left(x_{1,1}, ..., x_{1,T(1)}, ..., x_{N,1}, ..., x_{N,T(N)}\right) = \prod_{n=1}^{N} \prod_{k=2}^{T(n)} p\left(x_{n,k} \mid x_{n,k-1}\right) \tag{11}$$

The MLE solution for Q is:

$$Q = \frac{\delta - \beta F' - F\beta' + F\gamma F'}{\sum_{n=1}^{N} [T(n) - 1]} \tag{12}$$

where

$$\beta = \sum_{n=1}^{N} \sum_{k=2}^{T(n)} x_{n,k} x'_{n,k-1} \tag{13}$$

$$\gamma = \sum_{n=1}^{N} \sum_{k=2}^{T(n)} x_{n,k-1} x_{n,k-1} \tag{14}$$

$$\delta = \sum_{n=1}^{N} \sum_{k=2}^{T(n)} x_{n,k} x'_{n,k} \tag{15}$$

The derivation of (12) is available in Appendix A.

B. Models 3 and 4

Models 3 and 4 have no parameters assumptions for F and G like models 1 and 2. Models 3 and 4 are chosen with the intent of investigating the questions: If we had the extra information about the state of the leading vehicle, could we improve the process model? What happens with model 1 and 2 if all parameters were learned instead of being partially assumed? Model 3 is a first-order model like model 1 and model 4 is a second-order model like model 2. The exogenous inputs $u_{n,k}$ are the current state of the leading vehicle. All parameters F, G and Q are learned for both models. The parameters are determined using MLE as for models 1 and 2, resulting in:

$$F = \frac{\beta - \eta\zeta^{-1}\varepsilon}{\gamma - \varepsilon'\zeta^{-1}\varepsilon} \quad (16)$$

$$G = \frac{\eta - \beta\gamma^{-1}\varepsilon'}{\zeta - \varepsilon'\gamma^{-1}\varepsilon} \quad (17)$$

$$Q = \frac{\delta - F\beta' - G\eta'}{\sum_{n=1}^{N}[T(n)-1]} \quad (18)$$

where

$$\eta = \sum_{n=1}^{N}\sum_{k=2}^{T(n)} x_{n,k}u'_{n,k-1} \quad (19)$$

$$\zeta = \sum_{n=1}^{N}\sum_{k=2}^{T(n)} u_{n,k-1}u'_{n,k-1} \quad (20)$$

$$\varepsilon = \sum_{n=1}^{N}\sum_{k=2}^{T(n)} u_{n,k-1}x'_{n,k-1} \quad (21)$$

The derivations of (16), (17) and (18) are available in Appendix B.

C. Model 5 and 6

Models 5 and 6 are piecewise linear models. In principle they have the same structure as models 3 and 4, except they are divided in two regimes: one where the vehicle behaviour is constrained by the leading vehicle and one where the vehicle behaviour is not constrained by the leading vehicle. The same learning principles and parameter expressions from models 3 and 4 are applied to each regime separately.

We classify the learning data into 2 classes: leader-constrained and leader-unconstrained. This classification is achieved through the use of a Gipps car-following model [17]. A car-following model (CFM) describes a longitudinal vehicle behaviour given the behaviour of the leading vehicle. Specifically, the Gipps model is a piecewise nonlinear model with a leader-constrained and a leader-unconstrained regimes. Gipps determines the current regime by picking the one with smallest predicted speed. We apply each data point to the Gipps model and classify it according to the regime used by the Gipps model for that particular data point. This results in 2 sets of parameters: $\{F,G,Q\}^{constrained}$ and $\{F,G,Q\}^{unconstrained}$.

D. Model 7

Model 7 is a nonlinear model based entirely on the Gipps CFM. A CFM seems the most appropriate answer to best utilise the extra information about the leading vehicle. We chose the Gipps model from a previous study [18] showing this model to be superior among common CFMs.

Lidström and Larson [6] apply the Gipps model as the process model of a Bayesian Filter. Their state-space description of the Gipps model is:

$$\begin{bmatrix} s_k \\ d_k \end{bmatrix} = \begin{bmatrix} g\left(d_{k-1}, s_{k-1}, d^{lea}_{k-1}, s^{lea}_{k-1}, \Theta\right) \\ d_{k-1} + s_{k-1}\delta t \end{bmatrix} + v_k. \quad (22)$$

The state of the vehicle is represented by its distance d and speed s. The speed is determined by the Gipps model $g(\cdot)$ which requires the previous time-step vehicle's state, the leaders' state and a set of intrinsic model parameters Θ. The vehicle's position is defined by an Euler integration and v_k is a zero-mean additive Gaussian noise.

The apparent driver reaction time τ plays a vital role in the Gipps CFM. Its expressions are derived in such way that its discrete time updates are equal to τ. In other words, $\delta t = \tau = 0.6667s$.

We apply two modifications to the above model. One can observe that Gipps [17] uses a second-order Runge-Kutta integration to estimate distance from speed. Therefore we replace the Euler integration by a second-order Runge-Kutta integration. Also, the model above produces biased estimates when the real vehicle parameters do not match the model's parameters. The bias is irrelevant or perhaps even desirable for Lidström and Larson's application, but it is undesirable in ours. The bias can be estimated as a parameter and discounted from the estimate. Hoogendoorn and Ossen [19] studied parameter estimation of simple CFM based Bayesian Filters. The principle is also described in [11], [20]. Instead of introducing an additional bias parameter, we obtain the same effect by modelling the Gipps model intrinsic parameters as stochastic processes with independent increments at each time-step. The advantages of introducing the Gipps model parameters as stochastic processes are: removal of the bias from the estimates, elimination of the need for an artificial bias parameter and the available physical meaning of the Gipps' parameters. The 3 parameters from the Gipps model are the vehicle maximum acceleration a_{max_k}, the vehicle's driver desired speed s_{des_k} and the effective length of the leader l_k. These modifications result in:

$$\begin{bmatrix} a_{max_k} \\ s_{des_k} \\ l_k \\ s_k \\ d_k \end{bmatrix} = \begin{bmatrix} a_{max_{k-1}} \\ s_{des_{k-1}} \\ l_{k-1} \\ g(\cdot) \\ d_{k-1} + \frac{s_k + s_{k-1}}{2}\delta t \end{bmatrix} + v_k \quad (23)$$

where $g(\cdot) = g(d_{k-1}, s_{k-1}, d^{lea}_{k-1}, s^{lea}_{k-1}, a_{max_k}, s_{des_k}, l_k)$.

Note that $g(\cdot)$ uses the non-delayed versions of a_{max}, s_{des} and l. Here we take advantage of the more recent versions available through the top-down processing sequence of (23).

The model parameter we learn is the process model noise covariance $Q = \mathbb{E}[vv']$. Learning the process model noise covariance for a nonlinear stochastic model is not a trivial exercise and outside the scope of this paper. We approximate the process noise covariance from assumptions and experiments. Firstly, we use Gipps' suggestion of parameter variances of 0.3^2, 3.2^2 and 0.3^2 for a_{max}, s_{des} and l respectively. From Gipps' work, this choice of variances represent the population of vehicles from which new vehicle parameters are drawn. In ours, they represent the uncertainty of how these parameters evolve over time. The principles of choice are conceptually different, but the choice seems to work well on both scenarios as we will see in the next section. We also add a noise variance σ_s^2 for the vehicle speed. Finally, we assume that all noises are uncorrelated producing:

$$Q = \begin{bmatrix} 0.3^2 & 0 & 0 & 0 & 0 \\ 0 & 3.2^2 & 0 & 0 & 0 \\ 0 & 0 & 0.3^2 & 0 & 0 \\ 0 & 0 & 0 & \sigma_s^2 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad (24)$$

We determine σ_s^2 empirically from the Kalman innovation consistency test [11, p.234] using the learning dataset with simulated measurements.

VI. Results and discussion

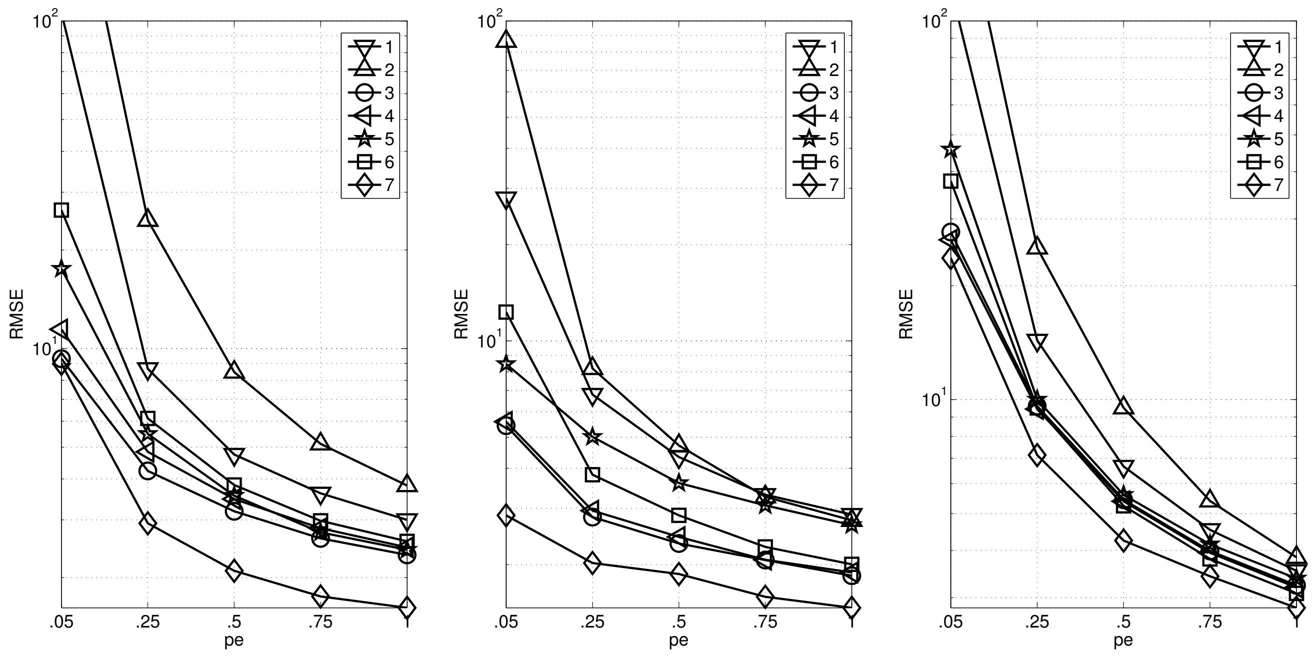
We collected RMS errors for all the evaluation data and aggregated them for each dataset, model type and *pe* measurement sequences. These are depicted in Fig. 5. We also calculated confidence intervals of the RMS errors. We choose not to include the intervals in Fig. 5 because their 1-sigma confidence interval had a maximum size of only 4% relative to the RMS error and would be almost undistinguishable in this Figure resolution. One can observe the that model 7 (Gipps CFM) has the smallest RMS errors over all dimensions. Models 3 and 4 produce sightly larger errors, but it is notable that model 3 has very small errors given its simplicity. Another advantage of model 3 over model 7, is that it is not bound to the time-stamp size.

Model 7 uses a well known and validated CFM. It is used in traffic simulation applications where platoon behaviour depends solely on this model and not on measurements. But how does model 3 behave as a CFM in a platoon simulation scenario with no measurements? Fig. 6 depicts results of a post-analysis on such a scenario. Figs. 6(a) and 6(b) depict an artificially simulated leading vehicle being followed by a platoon simulated with model 3 and 7 respectively. In model 3, we see some under-damped behaviour of the platoon behaviour starting at time 40 s where vehicles move backwards (lines with negative slope) and the shock wave is amplified. Such problems are not observed in model 7 (Fig. 6(b)). Figs. 6(c) and 6(d) depict a pair of artificially simulated traffic signals (simulated as a stopped leading vehicle) and a platoon simulated with models 3 and 7 respectively. In 6(c) we can observe that the linear over-damped behaviour of model 3 is not representative of a platoon behaviour. Again, no problems are observed in model 7 (Fig. 6(d)).

For completeness, we included Figs. 7(a) and 7(b) as position tracking examples of models 3 and 7 respectively.

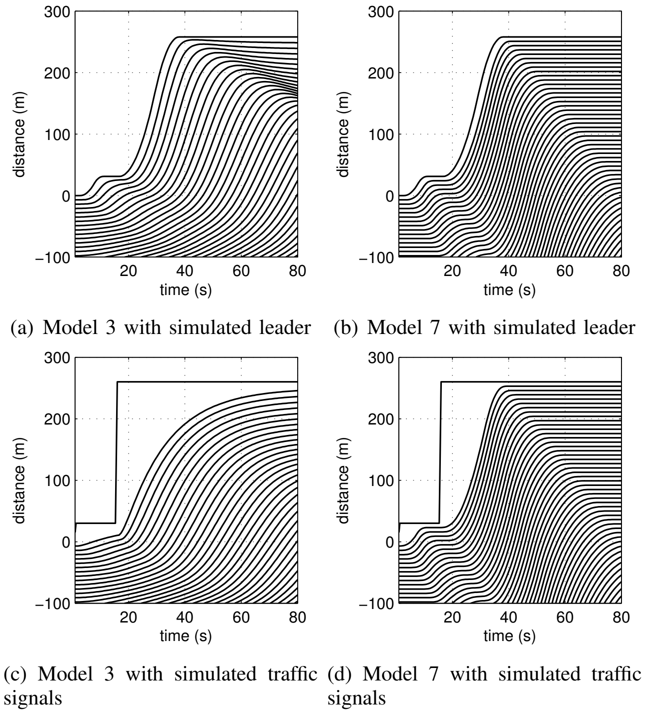
VII. Conclusion

We evaluated the performance of each model by simulating noisy measurements and comparing the position estimates with the dataset trajectories. The model with the smallest RMS errors is a model based on the Gipps CFM. A first-order linear model with exogenous inputs from the position, speed and length of the leader vehicle provided slightly larger errors. A post-analysis reveals this linear model produces a car-following-like behaviour in some situations. Similar to other linear car-following models [21], it suffers in performance when used in dynamic situations typically found in urban scenarios.



(a) NGSIM dataset results (b) Bosch dataset results (c) Simulation dataset results

Fig. 5. Comparison results. Root mean squared errors (*RMSE*) of different models with different measurement densities drawn with probability *pe*.



(a) Model 3 with simulated leader (b) Model 7 with simulated leader

(c) Model 3 with simulated traffic signals (d) Model 7 with simulated traffic signals

Fig. 6. Post-analysis on platoon simulation with no measurements

It is surprising that the linear model produces small errors during evaluation given the very poor behaviour depicted in Fig. 6(c) compared to the more appropriate car-following behaviour seen in Fig. 6(d). However, that scenario is poorly represented in the datasets, which is perhaps a weakness of our study. Nevertheless, this linear model remains a strong model candidate for situations where measurements are dense. The Gipps CFM based model is the most promising model presenting the smallest errors across all scenarios. It also behaves robustly under no-measurement platoon simulations.

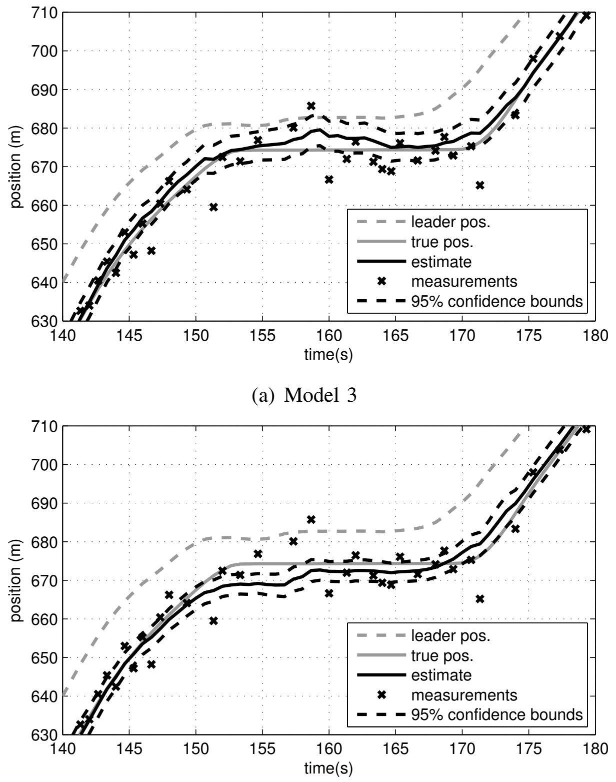
VIII. Future Work

The results of this paper form a building block for a wider scale real-time estimation system. We are interested in tracking multiple vehicles simultaneously and the next step is to track a platoon of vehicles. Improving the parameter learning of the nonlinear Gipps model is another aspect we are investigating. Wills et al. [22] recently showed how this can be achieved using the an Expectation-Maximisation algorithm with promising results.

Appendix

A. Finding the MLE of Models 1 and 2

The joint probability distribution in (11) is a multiplication of many Gaussian distributions and therefore is also a Gaussian distribution [8, p.502]. To find the parameters MLEs from a Gaussian distributed sample we must find the partial derivative of the likelihood function of the joint probability distribution with respect to the desired parameter. The solution is the root of the partial derivative [16, p.315].



(a) Model 3

(b) Model 7

Fig. 7. Tracking examples for part of the sequence shown in Fig 3(b). They correspond to scenarios with $pe = 0.5$. Estimation values are overlaid with true position values, measurements and 95% confidence bounds.

To find the MLE of the parameter Q we start by determining the likelihood function of the joint probability distribution (11):

$$L(Q \mid \mathbf{x}) = \prod_{n=1}^{N} \prod_{k=2}^{T(n)} p\left(x_{n,k} \mid x_{n,k-1}, Q\right) \tag{25}$$

The probability term in (25) is a Gaussian distribution:

$$p\left(x_{n,k} \mid x_{n,k-1}, Q\right) = \frac{1}{\sqrt[z]{2\pi}\sqrt{|\, Q \,|}} \exp\left(-\frac{d_{n,k}{}' Q^{-1} d_{n,k}}{2}\right) \tag{26}$$

where z is the size of the vector $x_{n,k}$ and $d_{n,k} = x_{n,k} - F x_{n,k-1}$.

In this case, it is easier [16, p.317] to use the natural logarithm of the likelihood function to calculate the derivative of (25). The resulting log likelihood function is:

$$\ell(F, G, Q \mid \mathbf{x}) = -\frac{z \log(2\pi)}{2} \sum_{n=1}^{N} [T(n) - 1] - \frac{\log |\, Q \,|}{2} \sum_{n=1}^{N} [T(n) - 1] - \sum_{n=1}^{N} \sum_{k=2}^{T(n)} \frac{d_{n,k}{}' Q^{-1} d_{n,k}}{2} \tag{27}$$

Now we take the partial derivative with respect to the parameter Q. In this case, it is easier to take the partial derivative with respect to Q^{-1}:

$$\frac{\partial \ell(F, G, Q \mid \mathbf{x})}{\partial Q^{-1}} = \frac{Q}{2} \sum_{n=1}^{N} [T(n) - 1] - \frac{1}{2} \sum_{n=1}^{N} \sum_{k=2}^{T(n)} d_{n,k} d_{n,k}{}' \tag{28}$$

where we used the identities:

$$\frac{\partial \log \mid X \mid}{\partial X^{-1}} = -X \text{ and } \frac{\partial a' X a}{\partial X} = a a'$$

Next we find the root of (28):

$$Q = \frac{\sum_{n=1}^{N} \sum_{k=2}^{T(n)} d_{n,k} d_{n,k}{}'}{\sum_{n=1}^{N} [T(n) - 1]} \tag{29}$$

Expanding the term $d_{n,k} d_{n,k}{}'$ finally gives us:

$$Q = \frac{\delta - \beta F' - F \beta' + F \gamma F'}{\sum_{n=1}^{N} [T(n) - 1]}$$

where β, γ and δ are defined in (13), (14) and (15).

B. Finding the MLEs of Models 3, 4, 5 and 6

To find the MLEs of parameter F, G and Q we borrow all definitions from Appendix A up to the log likelihood in (27). The only difference here are the exogenous inputs required which makes $d_{n,k} = x_{n,k} - F x_{n,k-1} - G u_{n,k-1}$.

We start by taking the partial derivatives with respect to F and G:

$$\frac{\partial \ell (F, G, Q \mid \mathbf{x})}{\partial F} = \sum_{n=1}^{N} \sum_{k=2}^{T(n)} \left(Q^{-1} d_{n,k} x'_{n,k-1} \right) \tag{30}$$

$$\frac{\partial \ell (F, G, Q \mid \mathbf{x})}{\partial G} = \sum_{n=1}^{N} \sum_{k=2}^{T(n)} \left(Q^{-1} d_{n,k} u'_{n,k-1} \right) \tag{31}$$

where we used the identity:

$$\frac{\partial (a - Xb)' W (a - Xb)}{\partial X} = -2W (a - Xb) b'$$

By finding the roots and expanding the term $d_{n,k} d_{n,k}{}'$ we get:

$$F = [\beta - G \varepsilon] \gamma^{-1} \text{ and } G = \left[\eta - F \varepsilon' \right] \zeta^{-1} \tag{32}$$

where η, ζ and ε are defined in (19), (20) and (21).

Equations in (32) are a system of two equations with two variables. By substitution we find the solutions:

$$F = \frac{\beta - \eta \zeta^{-1} \varepsilon}{\gamma - \varepsilon' \zeta^{-1} \varepsilon} \tag{33}$$

$$G = \frac{\eta - \beta \gamma^{-1} \varepsilon'}{\zeta - \varepsilon' \gamma^{-1} \varepsilon} \tag{34}$$

The solution for Q is analogous to the one in steps (28) and (29). But the expansion of the term $d_{n,k} d_{n,k}{}'$ containing the exogenous inputs results in:

$$Q = \frac{\delta - \beta F' - \eta G' - F \beta' + F \gamma F'}{\sum_{n=1}^{N} [T(n) - 1]} + \frac{F \varepsilon' G' - G \eta' + G \varepsilon F' + G \zeta \, G'}{\sum_{n=1}^{N} [T(n) - 1]} \tag{35}$$

Using the roots in (32) we can recognise that:

$$F \gamma F' = \beta F' - G \varepsilon' G' \tag{36}$$

$$G \zeta G' = \eta G' - G \varepsilon' F' \tag{37}$$

Substituting (36) and (37) in (35) we finally get:

$$Q = \frac{\delta - F \beta' - G \eta'}{\sum_{n=1}^{N} [T(n) - 1]}$$

References

[1] Z. Qiu, D. An, D. Yao, D. Zhou, and B. Ran, "An adaptive Kalman predictor applied to tracking vehicles in the traffic monitoring system," in *Proc. IEEE Intelligent Vehicles Symposium*, Las Vegas, USA, 2005.

[2] H. Veeraraghavan, O. Masoud, and N. P. Papanikolopoulos, "Computer vision algorithms for intersection monitoring," *IEEE Trans. Intell. Transp. Syst.*, vol. 4, no. 2, pp. 78–89, 2003.

[3] V. Punzo, D. J. Formisano, and V. Torrieri, "Nonstationary Kalman Filter for estimation of accurate and consistent car-following data," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1934, pp. 3–12, 2005.

[4] A. Polychronopoulos, M. Tsogas, A. J. Amditis, and L. Andreone, "Sensor fusion for predicting vehicles path for collision avoidance systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 3, pp. 549–562, 2007.

[5] K. Robert, "Night-time traffic surveillance: A robust framework for multi-vehicle detection, classification and tracking," in *IEEE International Conference on Advanced Video and Signal Based Surveillance*, Genoa, Italy, September 2009.

[6] K. Lidström and T. Larsson, "Model-based estimation of driver intentions using particle filtering," in *11th International IEEE Conference on Intelligent Transportation Systems*, Beijing, China, 2008.

[7] J. Diard, P. Bessière, and E. Mazer, "A survey of probabilistic models using the Bayesian Programming methodology as a unifying framework," in *Proc. 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems*, Singapore, 2003. [Online]. Available: http://cogprints.org/3755/

[8] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2002.

[9] R. E. Kalman, "A new approach to linear filtering and predictive problems," *Transactions ASME, Journal of Basic Engineering*, vol. 82, p. 3445, 1960.

[10] S. J. Julier and J. K. Uhlmann, "A new extension of the Kalman Filter to nonlinear systems," in *International Symposium on Aerospace/Defense Sensing, Simulation, and Controls*, Orlando, Florida, USA, 1997, pp. 182–193.

[11] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. John Wiley & Sons, Inc., 2001.

[12] S. Julier, J. Uhlmann, and H. Durrant-Whyte, "A new method for the nonlinear transformation of means and covariances in filters and estimators," *IEEE Trans. Autom. Control*, vol. 45, no. 3, pp. 477–482, 2000.

[13] (2009, May) Next Generation Simulation (NGSIM) program. [Online]. Available: http://www.ngsim.fhwa.dot.gov/

[14] (2009, May) German Aerospace Center (DLR) - Clearing House Transport. [Online]. Available: http://daten.clearingstelle-verkehr.de/191/

[15] D. Manstetten, W. Krautter, and T. Schwab, "Traffic simulation supporting urban control system development," in *4th World Congress on Intelligent Transport Systems*, Berlin, Germany, 1997.

[16] G. Casella and R. L. Berger, *Statistical Inference - Second Edition*. Duxbury, 2002.

[17] P. Gipps, "A behavioural car-following model for computer simulation," *Transportation Research Part B: Methodological*, vol. 15, no. 2, pp. 105–111, April 1981.

[18] S. Panwai and H. Dia, "Comparative evaluation of microscopic car-following behavior," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, pp. 314–325, 2005.

[19] S. P. Hoogendoorn and S. Ossen, *Transportation and Traffic Theory: Flow, Dynamics and Human Interaction: Proceedings of the 16th International Symposium on Transportation and Traffic Theory*. Emerald Group Publishing, 2005, ch. Parameter Estimation and Analysis of Car-Following Models, pp. 245–266.

[20] E. A. Wan and R. V. D. Merwe, *Kalman Filtering and Neural Networks*. Wiley, 2001, ch. The unscented Kalman filter, pp. 221–280.

[21] W. Helly, "Simulation of bottlenecks in single-lane traffic flow," in *Proc. Symposium on the Theory of Traffic Flow*, Michigan, USA, 1961.

[22] A. Wills, T. B. Schön, and B. Ninness, "Parameter estimation for discrete-time nonlinear systems using EM," in *Proc. 17th IFAC World Congress*, Seoul, Korea, jul 2008, pp. 1–6.