MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

# Driver Intention-based Vehicle Threat Assessment using Random Forests and Particle Filtering

Okamoto, K.; Berntorp, K.; Di Cairano, S.

TR2017-090 July 09, 2017

**Abstract**

One of the key technologies to safely operate self-driving vehicles is the threat assessment of other vehicles in the neighborhood of a self-driving vehicle. Threat assessment algorithms must be capable of predicting the future movement of other vehicles. Many algorithms, however, predict future trajectories based only on the model of the dynamics and the environment, which implies that they sometimes make too conservative predictions. This work reduces this conservativeness by capturing the driver intention of other vehicles using a randomforests classifier. Then, the algorithm computes possible future trajectories with a sequential Monte Carlo method, which biases the predicted trajectory by the recognized intention. Lastly, the algorithm calculates the potential threat to the ego vehicle. To evaluate the performance, we conduct numerical simulations and show that the proposed algorithm can accurately capture driver intentions and prevent motion predictions that are too conservative.

*World Congress of the International Federation of Automatic Control (IFAC)*

# Driver Intention-based Vehicle Threat Assessment using Random Forests and Particle Filtering

**Kazuhide Okamoto** [*,1] **Karl Berntorp** [**] **Stefano Di Cairano** [**]

[*] *Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: kazuhide@gatech.edu).*
[**] *Mitsubishi Electric Research Laboratories (MERL), Cambridge MA 02139 USA (e-mail: {karl.o.berntorp,dicairano}@ieee.org)*

**Abstract:** One of the key technologies to safely operate self-driving vehicles is the threat assessment of other vehicles in the neighborhood of a self-driving vehicle. Threat assessment algorithms must be capable of predicting the future movement of other vehicles. Many algorithms, however, predict future trajectories based only on the model of the dynamics and the environment, which implies that they sometimes make too conservative predictions. This work reduces this conservativeness by capturing the driver intention of other vehicles using a random-forests classifier. Then, the algorithm computes possible future trajectories with a sequential Monte Carlo method, which biases the predicted trajectory by the recognized intention. Lastly, the algorithm calculates the potential threat to the ego vehicle. To evaluate the performance, we conduct numerical simulations and show that the proposed algorithm can accurately capture driver intentions and prevent motion predictions that are too conservative.

*Keywords:* autonomous vehicle, intention recognition, machine learning, particle filter, path prediction, random forests, sequential Monte Carlo, supervised learning,

## 1. INTRODUCTION

Passenger vehicles are always required to operate safely, and this requirement also holds for autonomous vehicles. For autonomous vehicles to be accident free, the control system needs to assess the threat of the environment (e.g., static obstacles, pedestrians, and other vehicles) with data from sensors such as radars, lidars, and cameras. When the computed threat is greater than a specified threshold, the vehicle employs its path planning algorithm to compute a path that avoids the potentially unsafe objects. If the threat assessment (TA) is too optimistic, then the vehicle may collide with the objects. By contrast, if the TA is too conservative, then the vehicle will make an unnecessary detour, which will worsen efficiency, comfort, and the safety of the neighborhood traffic. Thus, TA algorithms play a key role in operating self-driving vehicles.

TA algorithms are either deterministic or stochastic. While deterministic TA algorithms predict a single future trajectory and compute the threat level, stochastic methods represent the future trajectories with probability density functions (PDFs), which are estimated using statistical methods such as Monte Carlo (MC) sampling (Broadhurst et al. (2005)). By taking into account the uncertainty, stochastic methods offer safer and more robust performance than deterministic methods. The most famous and popular stochastic motion prediction method is the Kalman filter (KF), which originally assumes a linear system model. This assumption is too strict because many real-world systems are nonlinear, and in that case the KF is sub-optimal. To mitigate this problem, several methods have been proposed based on techniques such as extended Kalman filter (EKF), unscented Kalman filter (UKF), and particle filter (PF). Researchers employ these methods to the state estimation and prediction of vehicle (e.g., interacting multiple model Kalman filter, (Carvalho et al. (2014)), unscented transform-based sampling and switching KF (Veeraraghavan et al. (2006)), and Gaussian processes (GP), (Armand et al. (2013))). Another approach to predict future trajectories is to directly solve the ordinary differential equation (ODE) of the system, which is nonlinear and stochastic. Solving nonlinear stochastic ODE is, however, difficult in general. Thus, solution-approximation methods are proposed such as Markov chain-based method (Althoff et al. (2011)) and zonotopes-based method (Althoff and Dolan (2014)). Note that general stochastic methods, which perform predictions only with models of the dynamics and the environment, sometimes predict future trajectories that rarely occur and are too conservative.

We mitigate this conservativeness using a machine-learning algorithm that performs driver intention recognition (DIR) of other vehicles. DIR approaches found in literature are typically either model-based or data-based. One model-based algorithm is employed by Salvucci (2004) who proposed a mind-tracking architecture with two-point visual driver control model. One of the most classical data-based DIR is hidden Markov models (HMMs)(Kuge et al. (2000); Lefèvre et al. (2015)). Recently, researchers employ supervised-classification algorithms for DIR such as support vector machine (SVM) (Mandalia and Salvucci

[1] This research was performed while at MERL.

(2005)), SVM and Bayesian filters (SVM-BF) (Aoude and How (2009)), and relevance vector machine (RVM) (McCall et al. (2005)).

Several researchers employed DIR for their TA methods.Eidehall and Petersson (2008) employed an iterative MC method that is biased by the driver-preference distribution. Aoude et al. (2010) addressed the problem of TA at intersections and employed an SVM-BF for DIR and explored future trajectories with the closed-loop rapidly-exploring random tree (CL-RRT), which is biased by the recognized driver intention. Laugier et al. (2011) employed a hierarchical HMM for DIR and used GP, which is trained a priori, to predict the future trajectories. Carvalho et al. (2014) employed IMM-KF to predict future trajectories. Our approach modifies these preceding methods and is able to compute future trajectories more computationally efficiently than iterative MC methods, recognize potentially more driver intentions than SVM-based methods, perform online tuning of parameters, and exhibit more versatile future predictions than KF-based methods.

The proposed algorithm follows the following four steps. First, to perform DIR of other vehicles, the algorithm employs random forests (RF), a supervised-classification algorithm proposed by Breiman (2001). The RF classifier, which is an ensemble learning method, creates many small decision trees to solve a given problem and votes for the most popular result. Second, we convert the recognized driver intention to a linear function of the vehicle state by modeling in the road-aligned coordinate frame. Third, we compute possible future trajectories with a sequential Monte Carlo (SMC) method, PF with optimal sampling (Arulampalam et al. (2002)), which biases the particles based on DIR. Fourth, we compute threat level using time to collision (TTC).

## 2. STATISTICAL THREAT ASSESSMENT METHOD WITH DRIVER INTENTION RECOGNITION

This section explains the proposed algorithm. Firstly, we define some terminologies employed in this paper. Secondly, we describe the entire framework of the algorithm. Then, we explain the feature extraction method and classification algorithm for intention recognition. Lastly, we describe the proposed algorithm to efficiently compute possible future trajectories of vehicles.

### *2.1 Notation*

We refer to a vehicle that employs the proposed method as an "ego vehicle" (EV). Suppose that an EV wants to compute the threat of another vehicle in its neighborhood, the region of interest (ROI). The ROI corresponds to the range of the employed sensors (e.g., radars, cameras, or lidars). All the vehicles in the ROI are denoted by "other vehicles" (OVs) (see Fig. 1). Note that OVs can be autonomous, semi-autonomous, or completely human-driven. In addition, we assume that the EV has access to the map of the road (e.g., given by a car-navigation system). We also assume that the EV can measure the following state of the OVs: distance along the road, deviation from the center of the road, and longitudinal and lateral speed. Because these measurements can be estimated from onboard sensors such as cameras, radars, and global positioning systems (GPS), this assumption is reasonable for a vehicle equipped with an advanced driver assist system (ADAS).

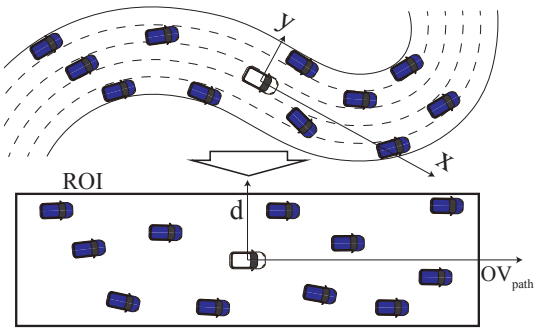



Fig. 1. The EV is the white vehicle in the center, and the OVs are the blue vehicles. Each state of OV is mapped to the road-aligned coordinate, and the shape of the ROI is not affected by the number of lanes and road curvature.

### *2.2 Entire Structure*

As shown in Algorithm 1, the proposed algorithm consists of the following four steps: DIR, intention-to-path conversion, reachable set computation, and threat computation. The first step is OV DIR with the RF algorithm using the data of the driving behavior of OV over the previous several seconds. The second step is to convert the captured driver intention to the corresponding path in the road-aligned frame. The third step, which we call "reachable set computation," predicts possible future trajectories of the OV with a sequential MC method. The last step assesses potential threat of the OV and outputs a threat measure.

**Algorithm 1** Statistical TA Method with DIR

**for** each $OV_i$ in ROI
1. $\mathcal{DI}_i \leftarrow \mathrm{DIR}(OV_i)$
2. $y_i \leftarrow \mathrm{Intention\_to\_path}(\mathcal{DI}_i)$
3. $ReachableSet_i \leftarrow \mathrm{PF\_with\_TA}(y_i)$
4. $ThreatLevel_i \leftarrow \mathrm{Threat\_level}(ReachableSet_i)$
**end for**

### *2.3 Intention Recognition of Other Vehicles*

After an OV enters the ROI of the EV, the proposed algorithm first estimates the intention of the OV driver. Since our interest is mostly in the lateral motion, we assume that the driver intention ($\mathcal{DI}$) consists of the following three intentions: to change lane to the left, $\mathcal{CL}$; to change lane to the right, $\mathcal{CR}$; and to stay in the lane, $\mathcal{SL}$.

*Feature Extraction:* Since we employ an RF classifier as a supervised-learning classification algorithm, we need training data with $\mathcal{DI}$. To have enough amount of training data, we employ a sliding window approach as follows. We compute the feature values from the following OV state variables: vehicle lateral position, $y$; longitudinal and lateral speed, $v_x$ and $v_y$, respectively. We assume that the EV can observe these state variables by processing information from onboard sensors (e.g., radars and cameras), GPS, and the road map. From the state variables and the road geometry, we compute the following features for

intention recognition: normalized lateral position, $\bar{y}$; normalized longitudinal speed, $\bar{v}_x$; and lateral speed, $v_y$. Each feature is explained in detail below. The raw data of these variables are subject to noise. Thus, for the robustness, we compute the following six statistical values of the features: the minimum, maximum, mean, variance, the difference between the first and the second value, and the difference between the last and the second to last value. Furthermore, to include changes of the values in the feature vector, we divide the dataset to four sub-datasets and compute the same statistical values of each sub-dataset. Therefore, the feature vector we create has $90 (= 3 \times 6 \times (1+4))$ entries.

To train the classifier, we employ a sliding-window approach. Let $N_{\text{total}}$ $(= 4 \times N)$, $N \in \mathbb{N}$, be the number of time steps of a window. We calculate the feature values $\bar{y}(t) \in [-1, 1]$ and $\bar{v}_x(t)$ by $\bar{y}(t_i) = \text{mod}(2y(t_i)/w+1, 2)-1$, where $w$ is the width of the road, $\text{mod}(a, b)$ returns the remainder after division of $a$ by $b$, and

$$\bar{v}_x(t_i) = \frac{v_x(t_i)}{\max_{t_0 \le t \le t_{N_{\text{total}}-1}}(v_x)}, \tag{1}$$

where $\max_{t_0 \le t \le t_{N_{\text{total}}-1}}(v_x)$ denotes the maximum value among $v_x(t_i)$, where $i = 0, 1, \ldots, N_{\text{total}}-1$. Then, we define the feature vector as

$$\begin{aligned} z(\xi, t_s, t_e) = [&\min(\xi(t_i)), \max(\xi(t_i)), \text{mean}(\xi(\text{t}_\text{i})), \\ &\text{Var}(\xi(t_i)), \xi(t_{s+1}) - \xi(t_s), \xi(t_e) - \xi(t_{e-1})] , \end{aligned} \tag{2}$$

and $\xi \in \{\bar{y}, \bar{v}_x, \bar{v}_y\}$, $t_s \in \{t_0, t_N, t_{2N}, t_{3N}\}$, $t_e \in \{t_{N-1}, t_{2N-1}, t_{3N-1}, t_{N_{\text{total}}-1}\}$. Thus, we define our feature vector as:

$$Z = [z_{\text{total}}(\bar{y}), z_{\text{total}}(\bar{v}_x), z_{\text{total}}(v_y)], \tag{3}$$

where

$$\begin{aligned} z_{\text{total}}(\xi) = [&z(\xi, t_0, t_N), z(\xi, t_{N+1}, t_{2N}), \\ &z(\xi, t_{2N+1}, t_{3N}), z(\xi, t_{3N+1}, t_{N_{\text{total}}})] . \end{aligned} \tag{4}$$

*Data Labeling:* Consider Fig. 2 where a driver intends to change lane from lane 1 to lane 2. We define the time the vehicle crosses the lane boundary as $t_{\text{cross}}$ and the vehicle reaches the center of lane 2 as $t_{\text{reach}}$. The window starts at $t_{\text{cross}} - t_1$ and ends at $t_{\text{cross}} + t_2$. The window size, $t_1 + t_2$, needs to be large enough so that the classification algorithm can capture the driver intention. Within the time window, we move a sub-window with length $t_{\text{window}}$, extract features from the state variables in the sub-window, and generate a dataset. We label the data in each sub-window as "Change Lane" (i.e., $\mathcal{CL}$ or $\mathcal{CR}$ depending on the direction) if the data start is between $t_{\text{cross}} - t_1$ and $t_{\text{reach}}$, and $\mathcal{SL}$ if it starts after $t_{\text{reach}}$. Note that when the training data has no lane changes, we cannot define a window as denoted above. Thus, we just extract dataset with length $t_1 + t_2$ and label all the dataset as $\mathcal{SL}$. Algorithm 2 briefly describes the algorithm.

**Algorithm 2** DIR

**Input**: vehicle lateral position, $y$, longitudinal and lateral speed, $v_x$, $v_y$, over the last $N_{\text{total}}$ steps.
**Output**: $\mathcal{DI} \in \{\mathcal{CL}, \mathcal{CR}, \mathcal{SL}\}$

1: Compute $Z$ in (3)
2: $\mathcal{DI} \leftarrow$ RandomForests($Z$)

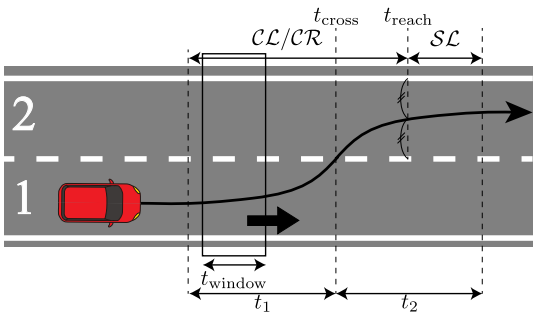



Fig. 2. Window size and data labeling.

*Random Forests* As discussed, the proposed algorithm captures $\mathcal{DI}$ based on the behavior of the vehicle over the previous several seconds. To estimate the intention, we employ RF (Breiman (2001)). While Torkkola et al. (2004) employed an RF classifier for an EV DIR, to the best of our knowledge, RF classifier has never been employed for OV DIR. As we mentioned in Section 1, many DIR algorithms employ SVM. Both SVM and RF are state-of-the-art algorithms for classification tasks, and determining which classifier outperforms the other a priori is difficult since the results depend on the data and the problem. Some research compared the performance of SVM and RF in various problem settings such as medical diagnosis (Statnikov and Aliferis (2007))). The reason why this work employs an RF classifier is that RF has superior performance than SVM in multi-class classification tasks. Although this current paper assumes that the driver has three intentions, the future work will capture the longitudinal intention of drivers (e.g., "to accelerate", "to decelerate", and "to keep a constant speed") simultaneously. To this end, we will need to solve a nine-class $(=3 \times 3)$ classification problem that is computationally heavy for SVM classifiers since they are based on the one-vs-all strategy, implying the need of training nine classifiers. By contrast, RF classifiers do not need to train nine classifiers. One classifier will classify the data to all of the nine intentions. Thus, RF is more favorable for our objective than SVM. Note that we can decouple the longitudinal and lateral dynamics and train separate classifiers. However, the longitudinal and lateral dynamics are in fact coupled, and ignoring this coupling may result in an inferior performance and possibly wrong conclusions about threat level.

### *2.4 Reachable Set Computation*

Having obtained the driver intention of the OV from the RF-based intention-recognition algorithm described above, the proposed TA algorithm computes a reachable set that is possible future trajectories of the OV. Since the intention recognition algorithm effectively reduces the diversion of the prediction problem, our algorithm will predict less conservative trajectories.

*Conversion of $\mathcal{DI}$ to path* First, the algorithm converts the recognized intention to corresponding desired location and velocity profiles in the road-aligned coordinate frame (see Fig. 1). This conversion is possible because, for instance, the intention of changing to the left lane implies an intended lateral position equal to the middle of the lane left to our current lane, and similar for the other options.

Hence, transforming intentions to actual expressions on the road gives us a measurement relation $y_k = Hx_k + n_k$, which is linear because we model everything in the road-aligned frame. Although other statistical motion prediction algorithms can be applied, our motion prediction algorithm employs an SMC method, specifically a PF with optimal sampling (Arulampalam et al. (2002)). We denote the dynamic model of OV as:

$$x_{k+1} = f(x_k) + v_k, \tag{5}$$

$$y_k = Hx_k + n_k, \tag{6}$$

where $v_k \sim \mathcal{N}(v_k; 0_{n_x \times 1}, Q_k)$, $n_k \sim \mathcal{N}(n_k; 0_{n_y \times 1}, R_k)$, and $f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ is the vehicle dynamics, $H \in \mathbb{R}^{n_y \times n_x}$ is an observation matrix, and $Q_k \in \mathbb{R}^{n_x \times n_x}$ and $R_k \in \mathbb{R}^{n_y \times n_y}$ are symmetric positive definite. PFs numerically estimate PDFs $p(x_k|y_{0:k})$ by generating $N_P$ random states $\{x_k^i\}_{i=1}^{N_P}$ at each time step $k$ and assigning a probability weight $w_k^i$, which reflects how well the state explains the observations $y_k$. In previous work we have developed a sampling-based motion planner based on particle filtering (Berntorp and Di Cairano (2016)), in which task specifications $y$ are used to guide the motion planner to the relevant parts of the state space. In this work, we let $y_k$ be the intended path of a driver.

*Remark: By incorporating the driver intention, we bias the reachable set toward the recognized intention. Thus, the reachable set we compute here is more reasonable and less conservative than previous statistical approaches.*

Algorithm 3 summarizes the process. Let $x_0$ be the initial state. The initialization step is to generate $x_1^i$ and $w_{1|0}^i$ according to (12). Let $T$ be the time horizon to perform prediction, then at each time step, perform the following four steps: *measurement update, estimation, resampling*, and *time update*. In the measurement update, given observation $y_k$, the algorithm updates the weights by (13) and (14). Then, the algorithm estimates the state according to (15), where $\delta$ is the Dirac delta function. Next, if some of the weights are too small, we resample particles. At last, we predict state according to (16) and update the weight as (17).

The performance of a PF depends on how to generate particles from $q$ in (16) (Gustafsson (2010)). The optimal proposal is computed as:

$$q(x_{k+1}|x_k^i, y_{k+1}) = \mathcal{N}(x_{k+1}; f(x_k^i) + K_k^i(y_{k+1} - \hat{y}_{k+1}^i), \Sigma_k^{-1}), \tag{7}$$

where

$$K_k^i = Q_k(H_k^i)^\top (H_k^i Q_k (H_k^i)^\top + R_k)^{-1}, \tag{8}$$

$$\Sigma_k = H_k^i R_k (H_k^i)^\top + Q_k, \tag{9}$$

$$\hat{y}_{k+1}^i = Hf(x_k^i). \tag{10}$$

Thus, the weights are updated as the following.

$$w_{k+1}^i \propto w_k^i \, \mathcal{N}(y_{k+1}; \hat{y}_{k+1}^i, H_k^i Q_k (H_k^i)^\top + R_k). \tag{11}$$

A key for guiding the particles using (7) is that we model everything in the road-aligned coordinate frame, resulting in a linear measurement model (6).

### *2.5 Threat Quantification*

As we predict the future trajectories with a PF, we can employ a PDF of future state of an object as a threat measure. Note that since we are investigating a TA method for autonomous vehicles, more complex measures can be an output such as a union of the PDF of all obstacles. The PDF can be leveraged by path planners such as the one proposed by Berntorp and Di Cairano (2016). But, for simplicity, in the numerical simulations in Section 3, we employ TTC defined by:

$$\mathrm{TTC}_i = \Delta t \cdot \inf_{k \in \{1,T\}} \left\{ k | pos_k^i \in \mathcal{B}_{pos_k^{\mathrm{EV}}}(r) \right\}, \tag{19}$$

where $\Delta t$ is the discretization time of the dynamics, $pos_k^i$ is the $i$th predicted position of the OV at time step $k$, and $\mathcal{B}_{pos_k^{\mathrm{EV}}}(r)$ represents the ball of radius $r$ centered at the position of the EV at time step $k$.

**Algorithm 3** PF_with_TA

**Input**: Intended path $y_k$, current state $x_0$, disturbances $v_k$ and $n_k$, where $k = 0, \ldots, T-1$.
**Output**: Particles $x_k^i$ and weight $w_{k|k}^i$, where $i = 1, \ldots, N_P$.

*Initialization.*

$$x_1^i \sim p(x_1|x_0), \; w_{1|0}^i = 1/N_P \tag{12}$$

**for** $k = 1 : T-1$ **do**

*Measurement Update.*

$$w_{k|k}^i = w_{k|k-1}^i p(y_k|x_k^i) \tag{13}$$

$$w_{k|k}^i = w_{k|k}^i / \sum_{i=1}^{N_P} w_{k|k}^i. \tag{14}$$

*Estimate PDF.*

$$\hat{p}(x_{1:k}|y_{1:k}) = \sum_{i=1}^{N_P} w_{k|k}^i \delta(x_{1:k} - x_{1:k}^i), \tag{15}$$

**if** needed **then**

*Resampling.*

**end if**

*Time Update.*

$$x_{k+1}^i \sim q(x_{k+1}|x_k^i, y_{k+1}), \tag{16}$$

$$w_{k+1|k}^i = w_{k|k}^i \frac{p(x_{k+1}^i|x_k^i)}{q(x_{k+1}^i|x_k^i, y_{k+1})}. \tag{17}$$

*Threat Assessment.*

**if** $x_{k+1}^i$ collides with an obstacle **then**

$$w_{k+1|k}^i = 0. \tag{18}$$

**end if**

**end for**

## 3. NUMERICAL SIMULATION

In the previous section, we introduced the proposed TA algorithm. This section evaluates the performance using numerical simulations. First, we evaluate the performance of the RF-based algorithm to recognize OV driver intention and compare it with an SVM-based method. Second, we show the effectiveness of the proposed algorithm in a situation where taking OV driver intention into account significantly impacts the performance.

To reproduce the vehicle dynamics in our simulation, we employ CarSim. [2] This software has sophisticated dynamic models and is frequently used in the automotive industry.

[2] www.carsim.com

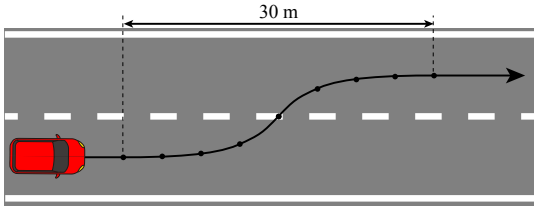



Fig. 3. The path to follow in training dataset. We put points in the transition path, which length is 30 meters independent on speed. We smooth the path by a spline interpolation and extrapolation.

### *3.1 Intention Recognition*

*Training Dataset* In this paper the intentions to be recognized are $\mathcal{CL}$,$\mathcal{CR}$, and $\mathcal{SL}$. Using CarSim, we create a training dataset with the following six speed settings: 80, 90, 100, 110, 120, and 130 kph. Thus, we prepared 18 (=3×6) datasets for training in total. The datasets are recorded at 40 Hz, which is enough to capture the driver intention for our RF classifier. A driver model installed in CarSim controls the vehicle and tries to follow a specified path, which we design a priori (see Fig. 3).

*Performance Measure* As performance measure, we employ precision, recall, and the F1 score: precision = TP/(TP+FP), recall = TP/(TP+FN), $F_1 = 2 \cdot$precision$\cdot$ recall/(precision+recall), where TP, FP, and FN represent "true positive", "false positive", and "false negative". True positive/negative is the number of times the classifier *correctly* identifies the class the given data *do/do not* belong to. False positive/negative is the number of times the classifier *incorrectly* identifies the class the given data *do/do not* belong to.

*Results* We test the proposed intention recognition algorithm with a test dataset, which we generate in the same way as the training dataset, but the speed is 95 kph. As the sliding window parameters, we set $t_1 = 2$ sec, $t_2 = 3$ sec, and $t_{\text{window}} = 1$ sec. The trained RF classifier has 100 classification trees. Table 1 lists the results. The precision of recognizing $\mathcal{CL}$ and $\mathcal{CR}$ are 1.00, but the values of recall are around 0.96 and 0.95, respectively. This result implies that the RF-based method always detects the driver intention when the driver changes lanes, although very few false detection occurs. By contrast, the recall of recognizing $\mathcal{SL}$ is 1.00 but the precision is 0.96. This result implies that the false detection of $\mathcal{SL}$ does not occur, but the algorithm does not always detect $\mathcal{SL}$. To compare the performance, Table 1 also lists the performance of an SVM-based method with a tuned radial basis function kernel. The $F_1$ score of our RF-based method is competitive to that of an SVM-based method. Also, as we discussed in Section 2, SVM-based methods have difficulty in multi-class classification. Although RF does not outperform SVM for lane-change intention recognition, it performs comparably. It is therefore clear that RF is more suitable when generalizing to more complex intention scenarios.

### *3.2 Parallel Driving*

In the previous scenario, we showed that the RF-based classifier we propose has a competitive performance to the SVM-based algorithm in the three-class DIR task and explained the advantage RF has over SVM. In the second numerical simulation, we assume that two vehicles are driving straight in parallel on a three-lane road. Here, we show that taking into account the OV driver intention prevents too conservative predictions that may lead to unnecessary escaping maneuvers.

Table 1. DIR performance (RF: left, SVM: right).

| | RF | | | SVM | | |
|---|---|---|---|---|---|---|
| | $\mathcal{CL}$ | $\mathcal{CR}$ | $\mathcal{SL}$ | $\mathcal{CL}$ | $\mathcal{CR}$ | $\mathcal{SL}$ |
| precision | 1.000 | 1.000 | 0.963 | 0.991 | 1.000 | 0.992 |
| recall | 0.964 | 0.946 | 1.00 | 1.000 | 0.982 | 0.996 |
| $F_1$ | 0.982 | 0.972 | 0.981 | 0.995 | 0.991 | 0.994 |

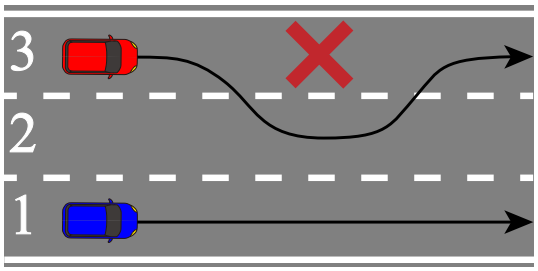



Fig. 4. The lane 3 has an obstacle, and the OV (red) avoids it by changing lane to lane 2 and back to lane 3. The EV (blue) assesses the threat while the OV is avoiding the obstacle.

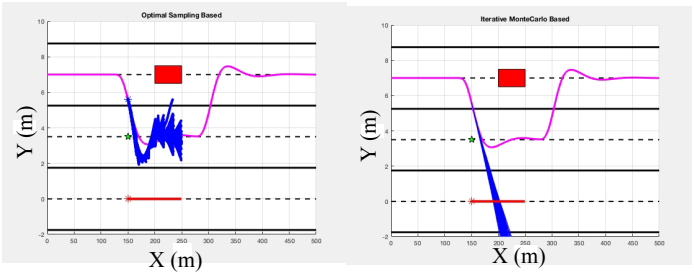
We assume that an EV is driving on lane 1 and an OV is driving on lane 3 (see Fig. 4). Furthermore, we assume that the OV finds an obstacle (e.g., a bursted tire, a broken car, or a pitfall) in the middle of the lane 3 and because the obstacle is very large, the OV tries to avoid it by changing lane to lane 2 and back to lane 3. We investigate the performance of the proposed TA method for the EV while the OV is performing this maneuver. Note that for simplicity we neglected the uncertainty of the EV future trajectories.

The data of the OV is generated with CarSim a priori, and the classification tree for the intention recognition is trained with the same database as the one trained for the first numerical simulation. The dynamics employed for motion prediction in this scenario is a single-track vehicle model (Berntorp et al. (2014)). We also converted the recognized intention to the path as described in Algorithm 4. We set the parameters as $t_1 = 2.0$ sec, $t_2 = 3.0$ sec, and $t_{\text{window}} = 1$ sec (See Fig. 2). The number of particles was $2^6$. We compared the performance of the proposed algorithm with an MC-based method inspired by Eidehall and Petersson (2008) without taking into account the $\mathcal{DI}$ and biasing the proposal probability, i.e., a naive MC approach. The number of cases for the MC was $2^8$. Both methods compute the TTC when the threat of collision exists. Figures 5 shows the predicted path of the proposed method and iterative MC method at one time step. Knowing the OV driver intention, the proposed method does not predict future collision. By contrast, the method without DIR predicted collisions, which leads to unnecessary escaping maneuver of the EV.

**Algorithm 4** Intention_to_path

**Input**: current $y$ coordinate, $\bar{y}$; $\mathcal{DI} \in \{\mathcal{CL}, \mathcal{CR}, \mathcal{SL}\}$
**Output**: intended path at time step $k$, $y_k$

$\eta_2$, $\eta_3$ ←coordinate of the center of lane2, lane3
**if** $\mathcal{DI} == \mathcal{CL}$ **then**
    $y_k$ ←$\eta_3$
**else if** $\mathcal{DI} == \mathcal{CR}$ **then**
    $y_k$ ←$\eta_2$
**else**
    **if** $|\bar{y} - \eta_3| < |\bar{y} - \eta_2|$ **then**
        $y_k$ ←$\eta_3$
    **else**
        $y_k$ ←$\eta_2$
    **end if**
**end if**




(a) Proposed Method (b) MC

Fig. 5. The EV drives straight along with $y = 0$. The red straight line is the predicted path of the EV. The blue asterisk is the current position of the OV. The blue lines are the reachable set of the OV, and the green star indicates the OV driver intention. The red box is the obstacle for the OV to avoid. The magenta line is the OV path, which is generated using CarSim a priori.

## 4. CONCLUSION

This paper addressed a new statistical algorithm to assess the threat of the neighborhood traffic of a vehicle. Firstly, the algorithm recognizes the intention of the OV driver using an RF classifier with behavioral data over the previous several seconds. After recognizing the driver intention, the algorithm computes possible future trajectories of the OV using a particle filter with optimal sampling. In a numerical simulation, the algorithm successfully recognized the intention of the driver and successfully prevented producing too conservative predictions.

Future work will include simultaneous recognitions of lateral and longitudinal driver intentions. The RF-based classifier we employ is expected to outperform SVM-based methods in this multi-class classification scenario. Furthermore, online learning of the reachable set computation parameters, $Q_k$ and $R_k$, is a possible research topic. In addition, we will generalize the algorithm and provide a more thorough comparison with other recently proposed methods.

## REFERENCES


Althoff, D., Wollherr, D., and Buss, M. (2011). Safety assessment of trajectories for navigation in uncertain and dynamic environments. In *Int. IEEE Conf. Robot. and Auto.*, 5407–5412. Shanghai, China.

Althoff, M. and Dolan, J.M. (2014). Online verification of automated road vehicles using reachability analysis. *IEEE Trans. Robotics*, 30(4), 903–918.

Aoude, G.S. and How, J.P. (2009). Using support vector machines and Bayesian filtering for classifying agent intentions at road intersections. *Tech. Rep. ACL09-02*.

Aoude, G.S., Luders, B.D., Lee, K.K., Levine, D.S., and How, J.P. (2010). Threat assessment design for driver assistance system at intersections. In *Int. IEEE Annu. Conf. Intelli. Trans. Sys.*, 1855–1862. Madeira Island, Portugal,.

Armand, A., Filliat, D., and Ibanez-Guzman, J. (2013). Modelling stop intersection approaches using Gaussian processes. In *Int. IEEE Conf. Intelli. Trans. Sys.*, 1650–1655. The Hague, Netherlands.

Arulampalam, M.S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. signal process.*, 50(2), 174–188.

Berntorp, K. and Di Cairano, S. (2016). Particle filtering for online motion planning with task specifications. In *Amer. Control Conf.* Boston, MA.

Berntorp, K., Olofsson, B., Lundahl, K., and Nielsen, L. (2014). Models and methodology for optimal trajectory generation in safety-critical road-vehicle manoeuvres. *Veh. Syst. Dyn.*, 52(10), 1304–1332.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1), 5–32.

Broadhurst, A., Baker, S., and Kanade, T. (2005). Monte Carlo road safety reasoning. In *IEEE Proc. Intelli. Veh. Symp.*, 319–324. Las Vegas, NV.

Carvalho, A., Gao, Y., Lefevre, S., and Borrelli, F. (2014). Stochastic predictive control of autonomous vehicles in uncertain environments. In *Int. Symp. on Adv. Veh. Cont.* Tokyo, Japan.

Eidehall, A. and Petersson, L. (2008). Statistical threat assessment for general road scenes using Monte Carlo sampling. *IEEE Trans. Intelli. Trans. Sys.*, 9(1), 137–147.

Gustafsson, F. (2010). Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 25(7), 53–82.

Kuge, N., Yamamura, T., Shimoyama, O., and Liu, A. (2000). A driver behavior recognition method based on a driver model framework. Technical report, SAE Tech. Paper.

Laugier, C., Paromtchik, I.E., Perrollaz, M., Yong, M., Yoder, J.D., Tay, C., Mekhnacha, K., and Nègre, A. (2011). Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety. *IEEE Intelli. Trans. Sys. Mag.*, 3(4), 4–19.

Lefèvre, S., Carvalho, A., and Borrelli, F. (2015). Autonomous car following: A learning-based approach. In *IEEE Intelli. Veh. Symp.*, 920–926. Seoul, Korea.

Mandalia, H.M. and Salvucci, M.D.D. (2005). Using support vector machines for lane-change detection. In *Proc. Human Fact. and Ergo. Soc. Annu. Meet.*, volume 49, 1965–1969. SAGE Publications.

McCall, J.C., Trivedi, M.M., Wipf, D., and Rao, B. (2005). Lane change intent analysis using robust operators and sparse Bayesian learning. In *IEEE Comp. Soc. Conf. Comp. Vision and Pattern Recog. -Workshops*, 59–59. San Diego, CA.

Salvucci, D.D. (2004). Inferring driver intent: A case study in lane-change detection. In *Proc. Human Fact. and Ergo. Soc. Annu. Meet.*, volume 48, 2228–2231. SAGE Publications, New Orleans, LA.

Statnikov, A. and Aliferis, C.F. (2007). Are random forests better than support vector machines for microarray-based cancer classification? In *Proc. of the AMIA Annu. Symp.*, 686–690. Amer. Med. Infor. Associ., Chicago, IL.

Torkkola, K., Massey, N., and Wood, C. (2004). Driver inattention detection through intelligent analysis of readily available sensors. In *IEEE Intelli. Trans. Sys. Conf.*, 326–331.

Veeraraghavan, H., Papanikolopoulos, N., and Schrater, P. (2006). Deterministic sampling-based switching Kalman filtering for vehicle tracking. In *IEEE Intelli. Trans. Sys. Conf.*, 1340–1345.