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

Joint Decision Making and Motion Planning for Road Vehicles Using Particle Filtering

Berntorp, Karl; Di Cairano, Stefano

TR2016-038 June 20, 2016

Abstract

This paper describes a probabilistic framework for real-time, joint decision making and trajectory generation of road vehicles. The selection of desired lane and state profile is posed as a stochastic nonlinear estimation problem. The nonlinear estimator is integrated with a sampling-based motion planner that computes candidate paths and corresponding state trajectories, while accounting for obstacle motion. The motion planner is implemented in receding horizon and repeatedly replans, based on stored candidate trajectories. A simulated autonomous highway-driving example illustrate show vehicle dynamics is naturally handled in the framework. The results show that the framework is capable of making effective online decisions and computing safe trajectories.

IFAC Symposium on Advances in Automotive Control (AAC) 2016

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Joint Decision Making and Motion Planning for Road Vehicles Using Particle Filtering

Karl Berntorp * **Stefano Di Cairano** *

* *Mitsubishi Electric Research Laboratories (MERL), 02139 Cambridge, MA, USA (Email:{karl.o.berntorp,dicairano}@ieee.org)*

Abstract: This paper describes a probabilistic framework for real-time, joint decision making and trajectory generation of road vehicles. The selection of desired lane and state profile is posed as a stochastic nonlinear estimation problem. The nonlinear estimator is integrated with a sampling-based motion planner that computes candidate paths and corresponding state trajectories, while accounting for obstacle motion. The motion planner is implemented in receding horizon and repeatedly replans, based on stored candidate trajectories. A simulated autonomous highway-driving example illustrates how vehicle dynamics is naturally handled in the framework. The results show that the framework is capable of making effective online decisions and computing safe trajectories.

Keywords: path planning, trajectory planning, vehicle dynamics, optimal estimation

1. INTRODUCTION

Decision making and motion planning are two key components for future generations of advanced driver-assistance systems (ADAS). Based on driver desires and current and predicted information, decision making concerns selecting the desired lane and velocity profile. In semi-autonomous mode, decision making can also include decisions about whether to assist the driver or take control over the car. Motion planning, on the other hand, addresses generation of suitable trajectories. Decision making and planning is sometimes modeled separately, but can also be considered jointly (Carvalho et al., 2015). In recent literature, model-predictive control (MPC) is a popular choice (Di Cairano et al., 2013; Carvalho et al., 2015; Ali et al., 2013) for designing ADAS, because it naturally provides a framework for performing joint decision making and trajectory generation. However, MPC is not a stand-alone component for solving the motion-planning problem, since it typically requires a precomputed reference trajectory.

Sampling-based methods, such as rapidly exploring random trees (RRTs) (LaValle, 2006; Karaman and Frazzoli, 2011), are popular and reliable methods for path planning in robotic applications, because they guarantee to provide a path whenever it exists. Traditional RRTs rely on random sampling of the state space. Each generated sample is checked for collision, typically assuming a static environment; if the sample location is collision free, it is added as a node, and collision-free connections are made to surrounding nodes for tree expansion. However, to include dynamics, or even kinematics, is nontrivial. Despite this, sampling-based approaches have been successfully used for motion planning of automotive systems in some cases, both in simulation and experiment. In (Hwan Jeon et al., 2013), time-optimal minimization of a hairpin turn was considered. Full-scale field tests for an online RRT have been performed in (Kuwata et al., 2009). There, an input-based RRT that samples position references to a tracking controller was developed. The method uses the tracking controller to connect position references to each other, thereby generating drivable paths between points. In (Arslan et al., 2016), the approach in (Kuwata et al., 2009) was extended to optimal motion planning.

In this paper, we propose a sampling-based, probabilistic framework for joint decision making and motion planning applied to road vehicles. RRTs are commonly used as path planners by generating samples such that the state space is covered uniformly, and do not include decision making. Our motion planner is entirely simulation based and complex vehicle dynamics is therefore naturally included, unlike most other sampling-based approaches. Here, the vehicle dynamics is modeled using a nonlinear single-track chassis model that incorporates combined-force modeling, seven states in total. Commonly, sampling-based planners generate random states and then connect these through approximate solutions to two-point boundary value problems, which is computationally prohibitive in automotive applications. In contrast, the proposed planner generates random inputs. This reduces the search space from the dimension of the state space to the dimension of the input space (i.e., from seven to two for the considered model), simplifying real-time implementation. In addition, our approach includes decision making in the planner already in the sampling phase. The input samples are guided to distinct lanes and velocity sets using particle filtering (Doucet et al., 2001; Gustafsson, 2010). This can be seen as a de-randomizing step. The proposed planner computes several sets of trajectories that can be used for decision making. Our approach draws inspiration from (Gustafsson et al., 2012; Eidehall and Petersson, 2008), where sequential Monte Carlo is used for threat assessment. This paper extends the application domain of sequential Monte Carlo to motion planning of road vehicles. The approach is explained and demonstrated using a simulated highway-driving example in different scenarios, such as overtaking of vehicles and in case of obstructed lanes.

2. VEHICLE MODELING

The model used in the motion planning is a nonlinear single-track model with lumped right and left wheels; for more model details, see (Berntorp, 2014). Fig. 1 shows the notation and

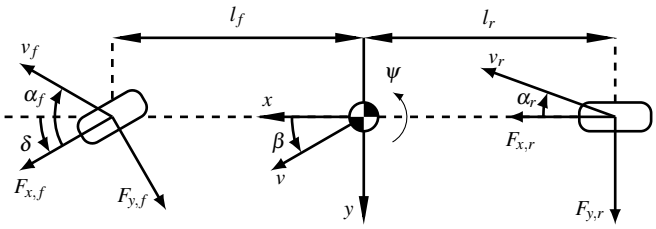

Fig. 1. Notation and geometry of the single-track chassis model used in the motion planning.

geometry. The chassis model has three degrees of freedom, two translational and one rotational:

$$\begin{aligned} \dot{v}_X - v_Y\dot{\psi} &= \frac{1}{m}(F_{x,f}\cos(\delta) + F_{x,r} - F_{y,f}\sin(\delta)), \\ \dot{v}_Y + v_X\dot{\psi} &= \frac{1}{m}(F_{y,f}\cos(\delta) + F_{y,r} + F_{x,f}\sin(\delta)), \\ I_{zz}\ddot{\psi} &= l_f F_{y,f}\cos(\delta) - l_r F_{y,r} + l_f F_{x,f}\sin(\delta), \end{aligned} \tag{1}$$

where m is the vehicle mass; I_{zz} is the vehicle inertia about the z-axis; $\dot{\psi}$ is the yaw rate; δ is the steer angle; v_x, v_y are the longitudinal and lateral velocities at the center of mass; l_f, l_r are the distances from the center of mass to the front and rear wheel base; and $\{F_{x,i}, F_{y,i}\}_{i=f,r}$ are the longitudinal and lateral tire forces acting at the front and rear wheels, respectively.

Slip angles α_f, α_r are introduced following (Pacejka, 2006) and are described by

$$\alpha_f := \delta - \arctan\left(\frac{v_y + l_f\dot{\psi}}{v_x}\right), \tag{2}$$

$$\alpha_r := -\arctan\left(\frac{v_y - l_r\dot{\psi}}{v_x}\right). \tag{3}$$

The lateral nominal tire forces—that is, the forces under pure slip conditions—are computed with a simplified Magic-Formula model (Pacejka, 2006),

$$\begin{aligned} F^0_{y,i} = \mu_{y,i} F_{z,i} \sin\Big(C_{y,i}\arctan\big(B_{y,i}\alpha_i \\ - E_{y,i}(B_{y,i}\alpha_i - \arctan(B_{y,i}\alpha_i))\big)\Big), \end{aligned} \tag{4}$$

$$F_{z,i} = mg(l - l_i)/l,\ i = f, r, \text{ where } l = l_f + l_r. \tag{5}$$

The simplification in (4) lies in that since a single-track model is used, the tire models are assumed symmetric in α_i. Furthermore, μ denotes the friction coefficient and B, C, and E are model parameters. Combined slip is modeled using the friction-ellipse concept:

$$F_{y,i} = F^0_{y,i}\sqrt{1 - \left(\frac{F_{x,i}}{\mu_{x,i}F_{z,i}}\right)^2}, \tag{6}$$

where $F^0_{y,i}$ is computed using (4), given the longitudinal force.

There are different options for choosing inputs to the vehicle model. Here, we have opted for using the steer rate $\dot{\delta}$ and longitudinal acceleration a_x as inputs. The choice of steer rate and acceleration implies that the resulting steer angle and velocity are continuous. Furthermore, it is straightforward to impose constraints on the inputs in the sampling phase with this choice of inputs, ensuring that the computed trajectories provide a smooth and safe ride. We model the steer dynamics as an integrator, but it is straightforward to include more complex dynamics. The acceleration is translated to longitudinal forces by assuming that the front and rear wheel axle provide longitudinal forces that are proportional to the length from the respective center of mass, that is,

$$F_{x,f} = ma_x l_r/l, \quad F_{x,r} = ma_x l_f/l.$$

Thus, the total number of states are

$$x = [X\ Y\ \psi\ \dot{\psi}\ v_x\ v_y\ \delta]^{\mathrm{T}}, \tag{7}$$

where the Cartesian positions X and Y relate to the mass center in the vehicle frame as

$$\begin{aligned} \dot{X} &= v_x\cos(\psi) - v_y\sin(\psi), \\ \dot{Y} &= v_x\sin(\psi) + v_y\cos(\psi), \end{aligned}$$

and the inputs are

$$u = \left[a_x\ \dot{\delta}\right]^{\mathrm{T}}. \tag{8}$$

In the following, we will compactly write (1)–(8) as

$$\dot{x} = f(x, u). \tag{9}$$

There are sources of uncertainty in this model. Mass and tire parameters are not exactly known and inputs are uncertain. However, the range of these uncertainties can typically be specified; for example, the inputs are bounded by their allowed ranges and it is more likely that the inputs are close to zero than saturating for normal driving. The uncertainties are therefore modeled by introducing stochastic process noise w acting on the inputs and states. Standard choices for noise distributions are Gaussian or uniform distributions. Other examples are given in (Gustafsson et al., 2012; Eidehall and Petersson, 2008).

3. JOINT DECISION MAKING AND TRAJECTORY GENERATION

Many sampling-based path planners find geometric paths based on a snapshot of the environment and are applicable in quite general scenarios (LaValle, 2006). However, they often ignore the equations of motion. For automotive systems, the vehicle dynamics significantly restricts the reachable set and should therefore be accounted for. An approach to incorporate equations of motion is to sample inputs and propagate through the system model (Hsu et al., 2002; Frazzoli et al., 2002). While this often works well, it can be very inefficient in certain scenarios, because it does not utilize structure in the motion-planning problem.

3.1 Task Specification

In many motion-planning problems there are certain specifications that should be fulfilled. In RRT, specifications are handled by appropriate choice of cost function, evaluated among all feasible paths. This can lead to inefficiency and poor performance in certain applications. While specifications on path level can be accounted for, it is hard to incorporate specifications on general variables while preserving convergence guarantees. In our approach, we model specifications as probabilistic constraints on the allowed motion and therefore only generate samples that are consistent with the specifications. To exemplify, when considering vehicle motion planning, possible specifications are:

- Stay on the road
- Maintain a desired velocity profile
- Drive smoothly, that is, prioritize small steer rates
- Keep safety distance to surrounding obstacles

Specifications are modeled as constraints by introducing them as desired outputs y_d as functions of the states,

$$y_d = h(x). \tag{10}$$

The desired outputs can directly correspond to some or all of the states. In case of specifications on the desired velocity, $y_d = v_{x,d}$. In terms of safety distance, however, the desired output is the preferred distance between suitable points on the ego vehicle (i.e., the vehicle executing the path planner) and obstacles.

In practice, the specifications cannot be tracked perfectly. We therefore add a probabilistic slack on each desired output, here expressed by e. The slack determines how large deviations can be tolerated. Imperfect tracking can occur for several reasons, being time constants, sensor uncertainties, or an overconstrained system.

When considering road-vehicle motion planning and decision making, different specifications have different priorities. For example, it is imperative that the car stays on the road, whenever that is possible, but it is not crucial for operability to exactly track a certain velocity. To illustrate our method, we introduce the following specifications for road-bound motion planning:

Follow a nominal longitudinal velocity v_{nom} This desired output directly corresponds to the longitudinal velocity, that is, $y_{d,1} = v_{\text{nom}}$ and $h_1(x) = v_x$.

Follow the middle lane For the second specification, we assume that the ego vehicle only uses the left-most lanes if needed. If there are no obstacles in the vicinity of the ego vehicle, the goal is to follow the middle of the right lane. The mid-lane error is defined as the lateral distance in the road frame from the vehicle to the middle lane and is denoted by p_e.

When there are obstacles present within the planning horizon, the motion planner needs to make a decision about which lane to follow. We model this as a bimodal relation. For a two-lane road with each lane having width w, the middle of the road can be chosen as the origin. By squaring the error from the middle of the road, we obtain the relation $h_2(x) = p_e^2$. Mid-lane following enforces that $y_{d,2} = (w/2)^2$. Such modeling can be extended to incorporate several lanes.

Maintain a safety margin to vehicles Assume that, within a prespecified radius from the ego vehicle, there are O obstacles with vectors $\{d_j\}_{j=1}^{O}$, which describe the squared distance along each coordinate axis, computed in the frame of the obstacle. The safety requirement is modeled as

$$h_{j+3} = \sigma e^{-d_j^{\mathrm{T}} W d_j}, \tag{11}$$

where σ is a scaling factor and W is a weighting matrix. This function has its peak value σ at $d_j = 0$, that is, at the center of the obstacle. As the arguments increase, (11) decreases in value. Thus, $y_{d,j+3} = 0$ if W is chosen appropriately. This specification will penalize trajectories that are close to surrounding vehicles. An alternative is to only choose the closest predicted obstacle. In this case, $O = 1$.

Maintain a safety distance d_s *to vehicles in the same lane* The safety distance d_s for vehicles in the same lane is velocity dependent. It is chosen as $d_s = v_x\Delta$, where Δ is the time to reach the obstacle when it is standing still. Denote the true distance with d_t. This specification is only invoked when d_t is below d_s, that is, when

$$d_t \leq d_s \tag{12}$$

is fulfilled. If (12) is satisfied, $h_{4+m} = d_s - d_t$ and $y_{d,4+m} = 0$.

The complete vector of desired outputs and state relation, respectively, is

$$y_d = \begin{bmatrix} v_{\text{nom}} & (w/2)^2 & \{0\}_{j=1}^{O} & 0 \end{bmatrix}^{\mathrm{T}}, \tag{13a}$$

$$h(x) = \begin{bmatrix} v_x & p_e^2 & \{\sigma e^{-d_j^{\mathrm{T}} W d_j}\}_{j=1}^{O} & d_s - d_t \end{bmatrix}^{\mathrm{T}}. \tag{13b}$$

When only one lane is to be tracked, the second argument in (13b) reduces to p_e and w in (13a) is set to zero.

3.2 State-Space Exploration using Particle Filtering

Here, we explore the state space using particle-filter (PF) techniques, assuming specifications in the form (13). The rationale for using PFs is that they provide an appealing way to statistically determine which computed trajectory is bad and which one is good, and they include many possibilities for biasing the samples and obtain reliable results with few samples. Furthermore, there is a vast literature on the theoretical properties of PFs (Crisan and Doucet, 2002; Douc et al., 2014). PFs numerically estimate probability distributions $p(x_k|y_{0:k})$ by generating N random states $\{x_k^i\}_{i=1}^{N}$ at each time step k and assigning a probability weight w_k^i, which reflects how well the state explains the observations. The random states are generated by sampling from a proposal density $q(\cdot)$ as

$$x_{k+1}^i \sim q(x_{k+1}|x_k^i, y_{k+1}). \tag{14}$$

The corresponding unnormalized weight is computed as

$$w_{k+1}^i \propto \frac{p(y_{k+1}|x_{k+1}^i)p(x_{k+1}^i|x_k^i)}{q(x_{k+1}^i|x_k^i, y_{k+1})} w_k^i, \tag{15}$$

which is then normalized to give the interpretation of a probability distribution. As in RRT, a node is selected at random using some heuristics π. A PF is then executed for T time steps with sampling time T_s. If particle i ends up in an occupied area, the corresponding weight w_{k+1}^i is set to zero. If all N weights become zero, the PF is terminated and a new node is chosen for tree expansion. The PF provides N state trajectories. For memory efficiency, we only store the weighted mean of the trajectories, which is the minimum mean-square estimate of the desired trajectory.

The choice of proposal density (14) is nontrivial and there is a plethora of alternatives readily available. Here, we guide the samples using the conditional distribution

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

This choice leads to the weight update

$$w_{k+1}^i \propto p(y_{k+1}|x_k^i) w_k^i. \tag{17}$$

Eq. (17) implies that the weight is independent of the sample x_k^i. The proposal (16) is optimal in the sense that it maximizes the effective number of samples (all other alternatives will lead to increased variance of the weights), but it is generally difficult to sample from exactly. However, for a linear, Gaussian measurement relation in the form

$$y_k = Hx_k + e_k,$$

the expression is analytic. For a nonlinear measurement relation a linearized version can be used, leading to

$$q(x_{k+1}|x_k^i, y_{k+1}) = \mathcal{N}\left(x_{k+1}|\hat{x}_{k+1}^i, (\Sigma_{k+1}^i)^{-1}\right) \tag{18}$$

where $\mathcal{N}(x|\mu,\Sigma)$ is the Gaussian density given mean μ and covariance Σ,

$$
\begin{aligned}
\hat{x}^i_{k+1} &= f(x^i_k) + L^i_k(y_{k+1} - \hat{y}^i_{k+1}),\\
\Sigma^i_{k+1} &= \left((H^i_k)^{\mathrm{T}} R^{-1}_{k+1} H^i_k + Q^{-1}_k\right)^{-1},\\
L^i_k &= Q_k (H^i_k)^{\mathrm{T}} (H^i_k Q_k (H^i_k)^{\mathrm{T}} + R_{k+1})^{-1},\\
\hat{y}^i_{k+1} &= H^i_k f(x^i_k),\\
H^i_k &= \left.\frac{\partial h}{\partial x}\right|_{f(x^i_k)},
\end{aligned}
$$

and Q_k is the second-order moment of the process noise. The likelihood in (17) is approximated as

$$
p(y_{k+1}|x^i_k) = \mathcal{N}\left(y_{k+1}|\hat{y}^i_{k+1}, H^i_k Q_k (H^i_k)^{\mathrm{T}} + R_{k+1}\right). \tag{19}
$$

3.3 Online Execution

The framework provides decision making and motion planning over a horizon constrained by the sensing information. The road to be traversed is assumed to be given by a higher-level logic, such as a car-navigation system. RRTs plan paths toward a predefined goal, then choose the best path along all those that have reached the goal point. Here, the goal is created by inserting waypoints in each lane at a distance constrained by the map geometry, current vehicle velocity, and predicted motion of obstacles. More complex goal-generation methods can be applied to the proposed motion planner. The RRT plans a trajectory toward one or several waypoints P for time Δt but only applies it for $\delta t \leq \Delta t$; that is, the planned trajectory is applied in receding horizon, similar to MPC.

3.4 Collision Checking

A collision-checking functionality returns `True` if the vehicle position (possibly with added safety margin) is unoccupied at time t_k. Otherwise, it returns `False`. The collision checking incorporates time, which is nonstandard. This implies that a larger portion of the generated tree can be reused in the next planning cycle. It also suppresses the need for reevaluation of nodes, which otherwise is needed when replanning in dynamic environments (Kuwata et al., 2009). The inclusion of time necessitates prediction of obstacles, given their measured (estimated) states at the beginning of every planning cycle, but the number of obstacles in the vicinity of the autonomous vehicle is typically much less than the number of nodes that have to be checked in the reevaluation.

Algorithm 1 provides the PF-based exploration phase and the complete motion planner is given in Algorithm 2.

Remark 1. By exploring the state space using task specifications, decision making is naturally incorporated into the algorithm. In standard RRT a tree containing a uniformly distributed set of points are generated and a user-designed cost function picks out the best trajectory in the tree as a last step in the algorithm. In the limit, the tree contains a continuoum of nodes and discrete decision making is therefore hard. With PF-based exploration, trajectories for different possible decisions (e.g., which lane to follow) are generated by viewing the decision process as a nonlinear, possibly multimodal, estimation problem. The generated tree therefore contains distinct set of trajectories, with each set corresponding to a possible decision.

Algorithm 1 Particle Filter for Tree Expansion

Initialize: Set $\{x^i_0\}_{i=1}^N = x_0$, $\{w^i_0\}_{i=1}^N = 1/N$, `Success` $\leftarrow$ `True`
for $k \leftarrow 0$ to $T-1$ **do**
  for $i \leftarrow 1$ to N **do**
    Generate new particles using (18):
    $$x^i_{k+1} \sim p(x_{k+1}|x^i_k, y_{k+1})$$
    if x^i_{k+1} is collision free **then**
      Compute weights using (17), (19):
      $$\bar{w}^i_{k+1} = w^i_k p(y_{k+1}|x^i_k)$$
    else
      Set $\bar{w}^i_{k+1} = 0$
    end if
  end for
  Set $N_w = \sum_{i=1}^N w^i_{k+1}$
  if $N_w = 0$ **then**
    `Success` $\leftarrow$ `False`
    Terminate algorithm
  end if
  Normalize: $w^i_{k+1} = \bar{w}^i_{k+1} / \sum_{j=1}^N \bar{w}^j_{k+1}$
  Set $N_{\mathrm{eff}} = 1/(\sum_{i=1}^N (w^i_k)^2)$
  if $N_{\mathrm{eff}} \leq \gamma N$ **then**
    Draw N samples with replacement, where the probability of drawing x^i_{k+1} is w^i_{k+1}
    Set $w^i_{k+1} = 1/N, \quad \forall i \in \{1,\ldots,N\}$
  end if
  Compute weighted mean:
  $$x_{k+1} = \sum_{i=1}^N w^i_{k+1} x^i_{k+1}$$
  Set $x_{0:k+1} = \{x_{0:k}, x_{k+1}\}$
end for
Return: $\{x_{0:T}, \texttt{Success}\}$

Algorithm 2 RRT with PF-Based Input Sampling

Input: Current vehicle state x_0, time t_0, waypoints, environment, stored tree
Set $t = t_0$
while $t \leq t_0 + \delta t$ **do**
  Select a node in the tree using heuristics π
  Execute Algorithm 1
  if `Success` **then**
    Add $x_{0:T}$ to the tree
  end if
end while
Compute best trajectory for each trajectory set
Choose decision
Apply trajectory for δt s and repeat from Line 1

4. SIMULATION STUDY

We present results from a simulation where the ego vehicle navigates on a two-lane road using Algorithm 2. The nominal desired speed of the ego vehicle is 25 m/s. Other vehicles on the road track either of the two lanes using state-feedback lane-keeping steering controllers and drive with constant velocity between 18.5–23 m/s. This implies that during parts of the simulation, both lanes are blocked by vehicles. The planner must therefore find trajectories that slow down and stay behind until an opening appears. When no obstacles are within the planning horizon, the goal is to track the right lane. However, as soon as an obstacle is detected, the planner computes trajectories for both lanes and then chooses the one with associated lowest cost.

Table 1. Model parameters in (1)–(6), from (Berntorp, 2014).

Notation	Value	Unit
l_f	1.3	m
l_r	1.5	m
m	2 100	kg
I_{zz}	3 900	kgm^2
g	9.81	ms^{-2}
Notation	Front	Rear
μ_x	1.20	1.20
μ_y	0.935	0.961
B_y	8.86	9.30
C_y	1.19	1.19
E_y	−1.21	−1.11

4.1 Parameters

For the simulation results, we use a planning horizon of $\Delta t = 3$, and $\delta t = 1$, meaning that the computed trajectory is at least 3 s long and is applied for a third of the computed horizon. The tree expansion uses $N = 50$ particles and the discretization of the dynamics is done with a time step $T_s = 0.1$ s. A trajectory is considered to have reached its goal when it reaches the middle of the right lane $\Delta t v_x$ m down the road (either of the lanes in case of obstacles), where v_x is taken as the vehicle velocity at the beginning of the planning phase. The values of Δt and δt reflect that onboard sensors can detect long-range obstacles with reasonable accuracy (Δt), but the accuracy is more reliable at shorter distances (δt). The cost C in the RRTs for each node is $C = 10(v_x - v_{\text{nom}})^2 + p_e^2 + d_t$. The noise matrices Q and R are

$$Q = \text{diag}([4^2, (30\pi/180)^2]), \quad R = \text{diag}([2^2, 2^2, 2^2, 3^2]),$$

where diag$(\cdot)$ is the diagonal matrix. Also, σ and W in (11) are chosen as $\sigma = 4$, $W = \text{diag}([5, 1])$. The desired safety distance is set to $d_s = 2v_x$. The model parameters used in this paper are shown in Table 1 .

4.2 Results

Fig. 2 shows four snapshots from an overtaking situation. In (a), at $t = 20$ s, when an obstacle is within the planning horizon Δt, waypoints are inserted in both lanes and the planner tries to connect to both of them. To stay behind the obstacle, the vehicle has to slow down, whereas it can maintain speed by changing lane. To maintain speed and overtake the obstacle gives a lower cost in this case. In (b), at $t = 25$ s, the ego vehicle maintains the left lane. In (c), at $t = 33$ s, Algorithm 2 computes a trajectory to safely reenter the right lane. Finally, (d) shows a snapshot at $t = 40$ s, where the ego vehicle has safely reentered the right lane, after the overtaking is concluded. It is clear that the algorithm provides safe and smooth trajectories. In this scenario, the algorithm decides to switch lane before it gets too close to the obstacle; that is, the safety-distance limit (12) is not activated, and (13b) without the last element is therefore used to guide the inputs (8).

Fig. 3 displays the bodyslip angle and longitudinal velocity for the period of time corresponding to the overtaking in Fig. 2. The bodyslip gives an indication of vehicle stability. For 25 m/s, a rule of thumb is that β should not exceed $\beta \approx 7$ deg (Kiencke and Nielsen, 2005). The planner fulfills this, and the nominal velocity 25 m/s is closely tracked throughout.

To show the algorithm's ability to track conflicting objectives, we present results from a scenario where there are two slower moving vehicles in both lanes, ahead of the ego vehicle. The obstacle in the right lane has velocity $v_x = 21.7$ m/s and the obstacle in the left lane has velocity $v_x = 22.5$ m/s. Fig. 4 displays snapshots of the scenario. Since both lanes are blocked, the motion planner has to compute trajectories that slow down and maintain the desired safety distance, see (12). Fig. 5 displays the distance to the closest of the two vehicles and the resulting velocity profile until overtaking is concluded. Up to $t = 25$ s, the ego vehicle maintains the right lane, tracking the middle of the lane and desired velocity $v_{\text{nom}} = 25$ m/s. At $t = 25$ s, the planner chooses to move to the left lane, since v_{nom} can be more closely tracked if switching lane. At $t = 30$ s, the safety criterion (12) is met and the input sampler is now also considering maintaining a safety distance corresponding to two seconds, see Sec. 4.1, that is, (13b) is used to guide the inputs (8). As seen from the figure, the inputs are generated such that the vehicle eventually travels at the same speed as the obstacle. After $t \approx 46$ s (d), the obstacle in the left lane is so far ahead the right-lane obstacle that an opening to reenter the right lane appears. Therefore, v_{nom} can be tracked again.

5. CONCLUSION

We presented a sampling-based method for real-time, joint decision making and motion planning. The proposed method is an input-based RRT that incorporates particle filtering as a means to probabilistically choose the control inputs according to specifications, hence achieving an efficient and nonsparse tree in the regions of most interest. The method generates drivable paths by construction. An enabler for this is the introduction of task specifications, which allows a reformulation of the exploration phase in the motion planning as a nonlinear estimation problem. Because the specifications are handled by a PF, highly nonlinear and/or conflicting objectives can be modeled. In a range of applications, there are specifications that the resulting trajectory should aim to fulfill. Our framework is therefore significantly more general than the particular example presented here.

Simulations on an autonomous-vehicle driving example showed that the method is capable of generating distinct sets of trajectories, according to task specifications, which can be used for online decision making. The ability of handling conflicting objectives was shown through an scenario where both lanes were blocked, forcing the method to prioritize trajectories that maintained a desired safety distance instead of maintaining constant speed.

The method is intended for online motion planning. The focus has therefore not been on finding optimal solutions. However, with the introduction of PF, there are relations to optimal estimation that can be further explored. It is future work to fully investigate convergence properties of our algorithm for different parameter choices. The algorithm is probabilistic and is therefore well suited to incorporate different aspects of uncertainty, such as sensor or estimation uncertainty.

REFERENCES

Ali, M., Falcone, P., Olsson, C., and Sjöberg, J. (2013). Predictive prevention of loss of vehicle control for roadway departure avoidance. *IEEE Trans. Intell. Transp. Syst.*, 14(1), 56–68.

Arslan, O., Berntorp, K., and Tsiotras, P. (2016). Sampling-based algorithms for optimal motion planning using closed-loop prediction. *ArXiv e-print: 1601.06326*.

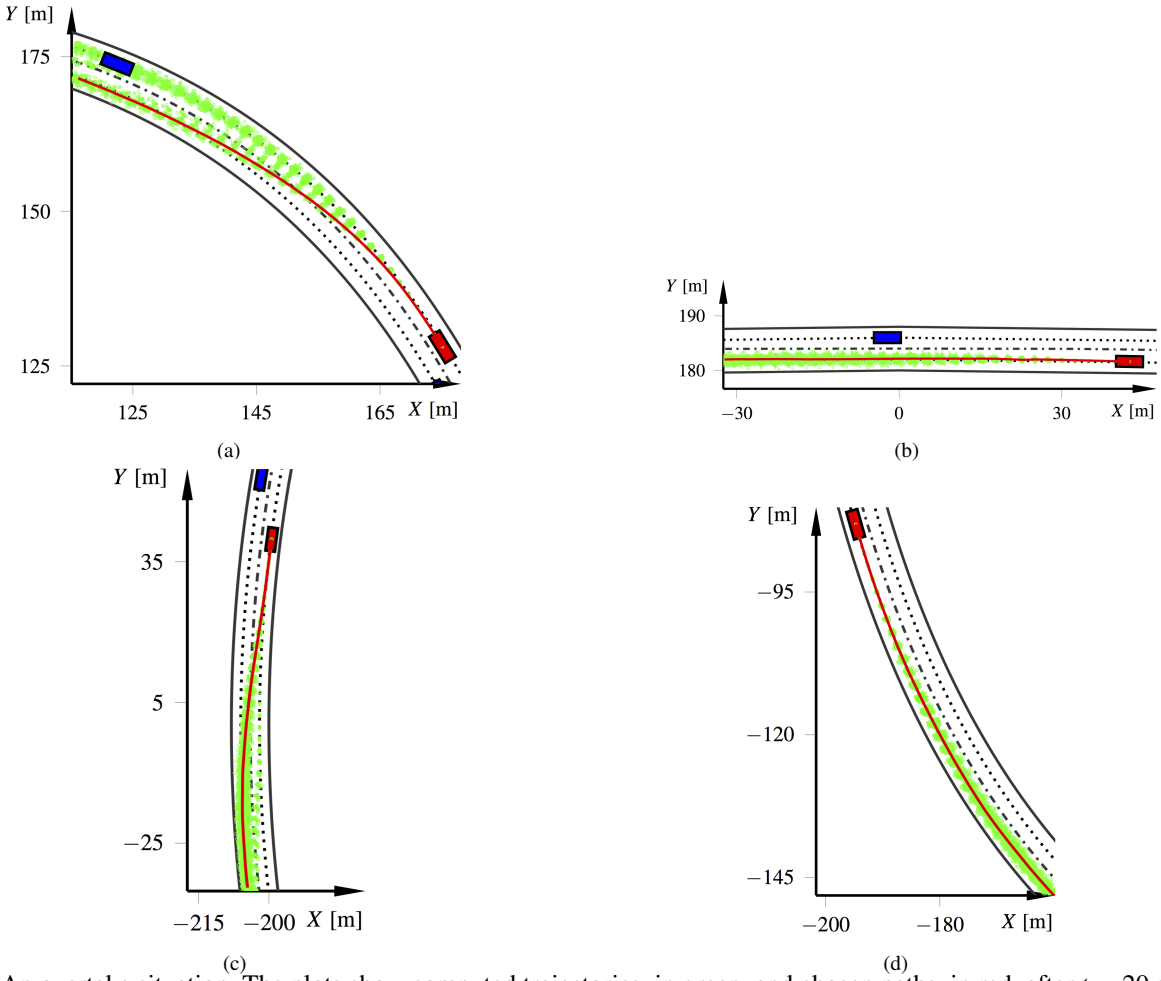

Fig. 2. An overtake situation. The plots show computed trajectories, in green, and chosen paths, in red, after $t = 20$ s (a), $t = 25$ s (b), $t = 33$ s (c), and $t = 40$ s (d).

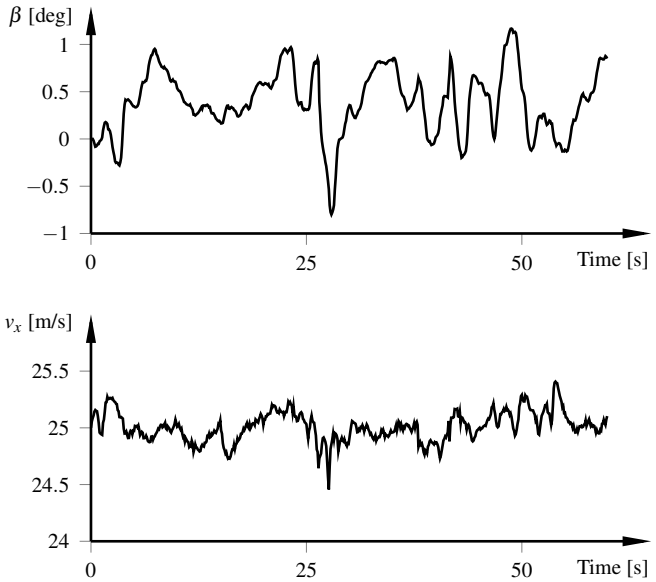

Fig. 3. The bodyslip angle β and computed longitudinal velocity v_x for one realization, corresponding to Fig. 2.

Berntorp, K. (2014). *Particle Filtering and Optimal Control for Vehicles and Robots*. Ph.D. thesis, Department of Automatic Control, Lund University, Sweden.

Carvalho, A., Lefvre, S., Schildbach, G., Kong, J., and Borrelli, F. (2015). Automated driving: The role of forecasts and uncertainty—A control perspective. *European J. Control*, 24, 14–32.

Crisan, D. and Doucet, A. (2002). A survey of convergence results on particle filtering methods for practitioners. *IEEE Trans. Signal Process.*, 50(3), 736–746.

Di Cairano, S., Tseng, H., Bernardini, D., and Bemporad, A. (2013). Vehicle yaw stability control by coordinated active front steering and differential braking in the tire sideslip angles domain. *IEEE Trans. Control Syst. Technol.*, 21(4), 1236–1248.

Douc, R., Moulines, E., and Olsson, J. (2014). Long-term stability of sequential Monte Carlo methods under verifiable conditions. *The Annals of Applied Probability*, 24(5), 1767–1802.

Doucet, A., De Freitas, N., and Gordon, N. (2001). An introduction to sequential Monte Carlo methods. In *Sequential Monte Carlo methods in practice*, 3–14. Springer.

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

Frazzoli, E., Dahleh, M.A., and Feron, E. (2002). Real-time motion planning for agile autonomous vehicles. *J. Guidance, Control, and Dynamics*, 25(1), 116–129.

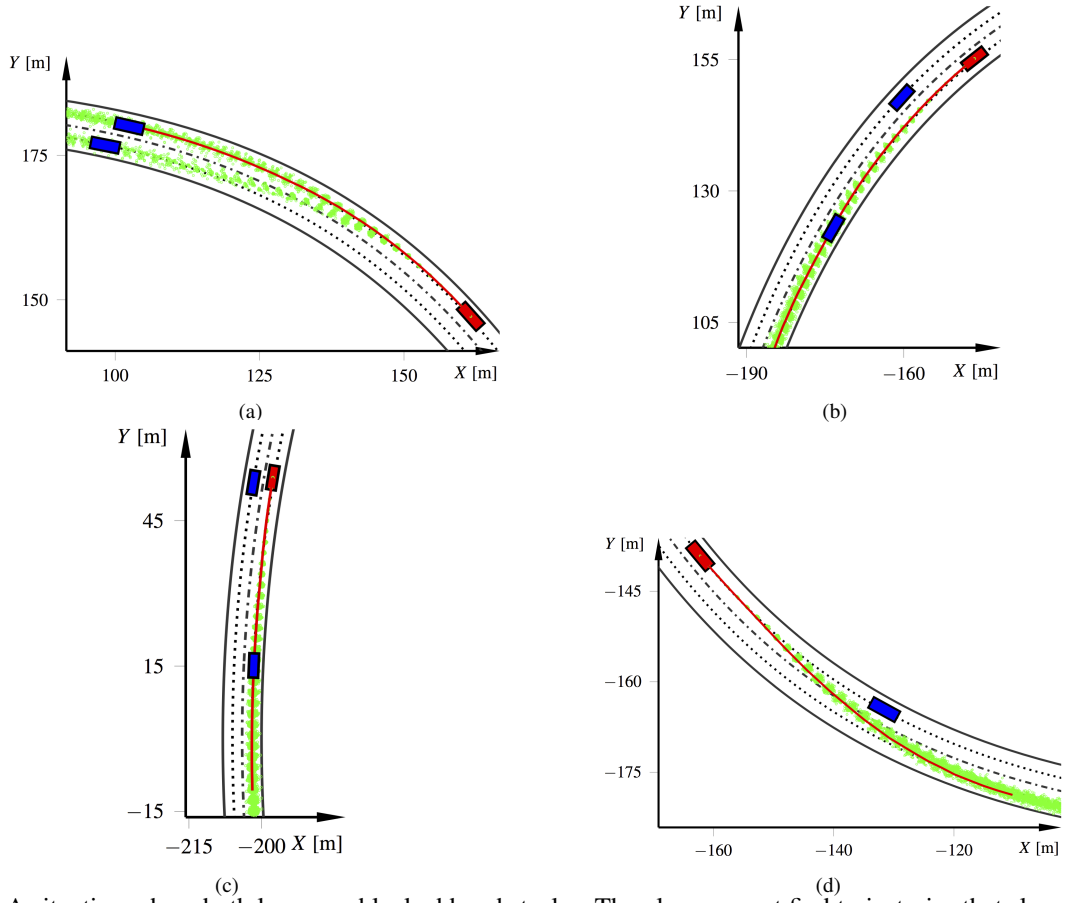

Fig. 4. A situation where both lanes are blocked by obstacles. The planner must find trajectories that slow down and stay behind, with maintained safety distance, until an opening to increase velocity and eventually overtake the vehicles appears.

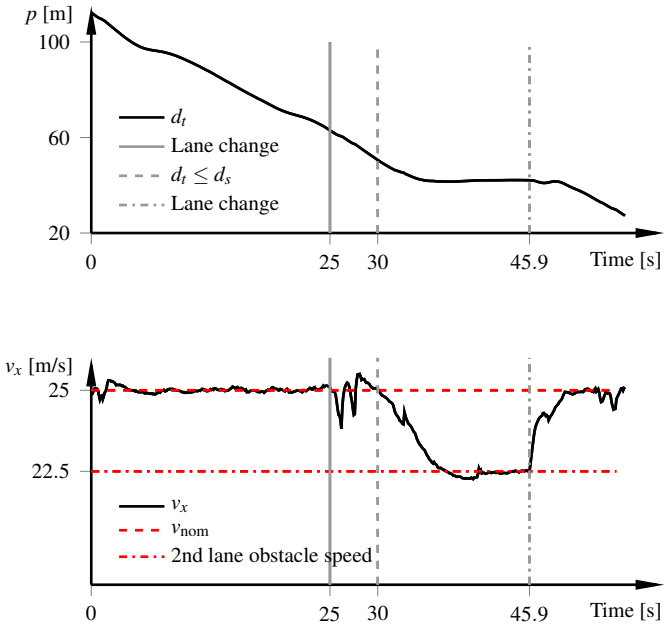

Fig. 5. Distance to closest vehicle and resulting trajectory where obstacles are present in both lanes. At $t = 25$, the ego vehicle changes to the left lane, at $t = 30$, the safety distance threshold is met, activating (12). At $t = 45.9$, the vehicle reenters the right lane and deactivates (12).

Gustafsson, F. (2010). Particle filter theory and practice with positioning applications. *IEEE Aerosp. Electron. Syst. Mag.*, 25(7), 53–82.

Gustafsson, F., Orguner, U., Schön, T.B., Skoglar, P., and Karlsson, R. (2012). Navigation and tracking of road-bound vehicles using map support. In *Handbook of Intelligent Vehicles*, 397–434. Springer.

Hsu, D., Kindel, R., Latombe, J.C., and Rock, S. (2002). Randomized kinodynamic motion planning with moving obstacles. *Int. J. Robotics Research*, 21(3), 233–255.

Hwan Jeon, J., Cowlagi, R.V., Peters, S.C., Karaman, S., Frazzoli, E., Tsiotras, P., and Iagnemma, K. (2013). Optimal motion planning with the half-car dynamical model for autonomous high-speed driving. In *Am. Control Conf.* Washington, DC.

Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *Int. J. Robotics Research*, 30(7), 846–894.

Kiencke, U. and Nielsen, L. (2005). *Automotive Control Systems—For Engine, Driveline and Vehicle*. Springer-Verlag, Berlin Heidelberg, Germany, 2 edition.

Kuwata, Y., Karaman, S., Teo, J., Frazzoli, E., How, J., and Fiore, G. (2009). Real-time motion planning with applications to autonomous urban driving. *IEEE Trans. Control Syst. Technol.*, 17(5), 1105–1118.

LaValle, S.M. (2006). *Planning Algorithms*. Cambridge University Press, Cambridge, United Kingdom.

Pacejka, H.B. (2006). *Tire and Vehicle Dynamics*. Butterworth-Heinemann, Oxford, United Kingdom, 2nd edition.