2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Towards Risk Minimizing Trajectory Planning in On-Road Scenarios

Erik Ward[1], John Folkesson[1]

Abstract— **Trajectory planning for autonomous vehicles should attempt to minimize expected risk given noisy sensor data and uncertain predictions of the near future. In this paper, we present a trajectory planning approach for on-road scenarios where we use a graph search approximation. Uncertain predictions of other vehicles are accounted for by a novel inference technique that allows efficient calculation of the probability of dangerous outcomes for set of modeled situation types.**

I. Introduction

Autonomous vehicles need to navigate road environments and handle complex situations. This includes coping with uncertainties from perception and uncertain predictions of the behavior of nearby road users. It is often necessary to plan ahead in order to make safe decisions, for instance when merging onto the highway, or negotiating an intersection, as a dangerous situation might only manifest several seconds after a dangerous decision is made.

Typically autonomous vehicles choose their actions by optimizing future control inputs to find the next trajectory to be executed in a receding horizon fashion [1]. Optimization criteria include comfort but also collision avoidance to other agents in the scene. Trajectory optimization is hard because we must consider not only the position of the non holonomic vehicle during planning, but also the time we arrive there, and the problem is non-convex in general. Managing uncertainty adds an additional layer of difficulty as we now need to plan in belief space, where we cannot know the result of applying different control actions.

In this paper we propose a receding horizon trajectory planning approach for on-road scenarios, based on the A^* algorithm, designed to achieve long planning horizons, described in Sec. IV, where trajectories are defined in relation to the current road. We model possibly dangerous future situations as belonging to one of the following types: car following, lane changes, merge maneuvers and crossing the path of another vehicle. The method copes with uncertain predictions of nearby vehicles by analyzing the planned state of the autonomous vehicle in relation to predicted states of nearby vehicles, and then estimating the probability of dangerous outcomes of the aforementioned situation types. These include modeling the event that a vehicle in front suddenly brakes. Inference of the probability of these events involve non-linear transformations of uncertain future state variables and the proposed inference algorithm, fast enough for real time application, is based on approximating the infeasible region of the state space of possible future positions and speeds of other vehicles with linear regions. Our inference method ensures that we do not under-estimate the probability of dangerous outcomes and avoids numerical issues of previous such methods.

How a planning algorithm can be designed depends on how we approximate of the belief space. Although the problem can be modeled as a Partially Observable Markov Decision process, we cannot solve these exactly for any practical application. We have chosen to ignore future measurements and not to explicitly model interactions between vehicles during planning, in favor of using a simple, parametric, description of future belief states. We discuss the benefits and downsides of this choice and compare with an approximate, Monte Carlo based, POMDP method in Sec. VI-B.

II. Related Work

In the review by Gonzalez et al. [1] decision making, given a route to follow, is divided into two steps: i) behavioral planning and ii) local planning, where behavioral planning works on higher level of abstraction than local planning. Determining which maneuver a vehicle should perform is a non-convex problem [2], for instance we might decide to pass on the right or the left of another object. This non-convexity cannot be handled directly by continuous optimization solvers as noted in [3], and to use traditional MPC algorithms a behavior planning step is needed to set up a convex problem. E.g. in [4] a suitable gap for a lane change is searched for considering the reachable set of speeds for the ego vehicle, rather than its eventual trajectory, then this gap gives temporal constraints for a QP solver. However, in some related works it is not reasonable to define these two steps, as the optimization of the ego vehicle's trajectory is performed by one search procedure [5], also, different authors have each step work at different levels of abstraction, therefore we choose to use the term *trajectory planning* to encompass both steps i) and ii). Different authors have used different terminology, for instance [2], [6] use *driving strategy* to mean a planning algorithm that searches for a sequence of states that approximate the trajectory the ego vehicle should execute. In [7] a tree search algorithm is used to find a sequence of high level maneuvers for the ego vehicle, and instead *behavior planning* is used. Common to all approaches is comparing planned future states of the ego vehicle (position and speed) against predicted states of other vehicles using different risk assessment functions. In this work we use a similar planning algorithm as [2], extended to handle lane changes in a similar way as the lateral motion options in [6], although with a substantially longer prediction horizon.

The most basic risk avoidance constraint type in trajectory planning are collision avoidance constraints assuming perfect predictions, implemented by comparing the extent of the ego vehicle with that of other objects [2], [3], [5]. But

The research leading to these results has been carried out within the iQMatic project funded by FFI/VINNOVA.

[1]Robotics Perception and Learning Lab, KTH, Stockholm, Sweden (e-mail: eward@kth.se, johnf@kth.se)

in reality these predictions are uncertain, both from sensor noise and because of the difficulty of predicting the future, so deterministic collision checking is not enough to ensure safety. An ad-hoc solution is to add some extra margins to compensate for uncertainties [4], but using an algorithm which allows constraints on actual risk probabilities allows for more rigorous treatment of risk. Ideally we would like to consider a probability density over all possible future states of other vehicles and the probability of collisions, checked by calculating the probability of overlap between ego and any other vehicle. However, approximating this probability faithfully is computationally expensive and must be done for each proposed trajectory of the ego vehicle [8]. Instead alternative risk measures can be used inside planning algorithms, such as time to collision (TTC) [6] or braking distances [9], however these measures should also account for uncertainty.

In the proposed method, we ignore modeling future measurements and instead assume that we are provided with the distribution over other vehicles states for the time steps in the prediction horizon in advance of planning. We also only find the optimal control sequence from the start state to one of the goal states, rather than a policy, Π, as in a POMDP. In the context of autonomous driving POMDPs have been proposed by [10], [11] and others. Here the belief space is continuous over the possible future states of other agents and the dimensionality therefore grows with the number of modeled agents. This allows modeling of interactions between agents, and modeling of arbitrary futures (rather than the mixture of Gaussian assumption in our proposed approach). Hubmann et. al proposed an on-line approximate POMDP method for autonomous driving [10], where an acceleration policy for the ego vehicle is found. Here the policy is computed only for a few future time steps, considering only belief states that can be reached in this time and approximated with a *Belief Tree*, using the TAPIR library [12]. At a fixed interval the policy is re-computed. This re-planning makes sure that the computed policy is relevant to the current measurements. In Sec. VI-B we compare the proposed method and the method of [10] in an intersection scenario and discuss the differences.

III. Problem overview

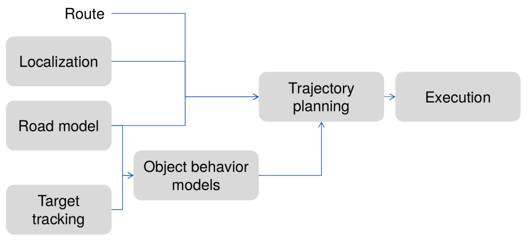

Fig. 1: System architecture for road driving function.

Figure 1 shows a high level view of modules for autonomous on-road operation. A global planning module processes missions to provide a route for the trajectory planning module, which also requires localization, models of nearby lanes and models describing the state and behavior of nearby objects. The trajectory planning module should find a suitable trajectory for the autonomous vehicle, referred to as the *ego vehicle*, to follow:

$$x^e(t),\ t_{now} \leq t \leq t_{now} + T_h$$

where x^e is the state of the ego vehicle and T_h is the length of the prediction horizon. This is done by optimizing a cost function where high risk and uncomfortable trajectories have high cost. Risk is estimated by analyzing uncertain predictions for other nearby vehicles, indexed by i. This planning under uncertainty will be repeated at a fixed interval in a receding horizon approach, while the current best plan is executed by lower level control modules.

Here we briefly outline the trajectory planning problem under uncertainty by borrowing notation and definitions from [13]. We assume that the state of the environment $x = (x^e, x^i),\ i = 0,..,K$ evolves in discrete time steps for the planning horizon:

$$x_{k+1} = f(x_k, u_k, \omega_k),\ k = 1, ..., N$$

where x_k^i is the state of vehicle i at time $t_{now} + k\Delta t_p$, u_k is a control action for the ego vehicle, ω_k a disturbance, and $N = T_h/\Delta t_p$ where Δt_p is the time discretization. We also get noisy measurements of the environment every time step:

$$y_k = h(x_k, \nu_k)$$

where ν_k models measurement noise. In general, we cannot observe x_k directly and instead we summarize all information available in the information state I_k. The goal is to find an optimal policy $\Pi^* = \{\pi_0^*(I_0), ..., \pi_{N-1}^*(I_{N-1})\}$ for the planning horizon which gives us the control action that minimizes future expected costs, given the information state, for an additive cost function:

$$L(x_{0:N}, \Pi) = l_N(x_N) + \sum_{k=0}^{N-1} l_k(x_k, \pi_k(I_k))$$

To find an optimal policy we plan in *belief space*, and make different assumptions on future measurements to get a tractable problem. Where the belief state, b_k, is defined as $b_k = p(x_k|I_k)$ and I_k is the *history I-state*: $I_k = (I_0, u_{0:k-1}, y_{0:k-1})$. This problem can be reformulated into a Stochastic Dynamic Programming (SDP) problem.

We can use partially observable decision process (POMDP) methods to approximate solutions to the SDP problem or we can solve for a sequence of control actions, in a receding horizon fashion, ignoring future measurements. This latter approach, which we use in this paper, is called open-loop receding horizon control (OLRHC), and here the belief states are “the objects’ open-loop predicted distributions” [13] and feedback is achieved by re-planning. We encourage the reader to consult [13] for further details.

A. Inputs

As in [5] [14], our trajectory planning algorithm describes the state of the autonomous vehicle, in a road aligned coordinate system where the abscissa corresponds to the arc length s along a reference path and the ordinate the signed lateral distance from the path d. We assume that we can measure the current state in relation to the reference path, (s_0^e, d_0^e), exactly and that a list of open-loop predictions for other vehicles is available. Let x_k^i be the pose and speed of vehicle i in an inertial frame of reference, $x_k^i = (x, y, \theta, v)$, $x_{k,r}^i$ the state in relation to reference path r, $x_{k,r}^i = (s_{k,r}^i, d_{k,r}^i, \dot{s}_{k,r}^i)$, and $g(x_{k,r}^i) = x_k^i$ a function that maps between them. The prediction for each other vehicle is expressed as a sequence of mixture of Gaussians, where each Gaussian models the distribution of the vehicle's state in relation to a specific route:

$$p(x_{0:N}^i) = \sum \alpha_j p(x_{j,k}^i),\ k = 0, .., N,\ x_{j,k}^i = g(x_{k,r(j)}^i)$$

where α_j is mixture component weight for Gaussian $p_{r_j}(x_{k,r(j)}^i) = \mathcal{N}(\mu_{r(j),k}^i, \Sigma_{r(j),k}^i)$ describing the state distribution of vehicle i along the route, $r(j)$, of hypothesis j. This representation allows us to express multiple motion hypotheses for a single vehicle, for example in Sec VI-B we model uncertainty in whether or not a vehicle will turn using different α_j values for two hypotheses, each belonging to a different route. The distributions $\mathcal{N}(\mu_{r(j),k}^i, \Sigma_{r(j),k}^i)$ could be estimated with different methods, for example [15], or a simple method such as performing prediction steps of a Kalman Filter. We also assume that we know the longitudinal extent of each vehicle along its route.

The trajectory planning algorithm has access to a road model where it can quickly determine whether or not two routes are adjacent, e.g. to check if a vehicle is to the left or right of us, if one merges with the other or if routes cross. The risk estimation methods described in Sec IV-C are primarily based the distance until two vehicles occupy the same longitudinal position on a route and the vehicles relative speeds.

IV. Graph Search Approximation

Deciding on when to change lane, or when drive onto a section of the road where two lanes merge is a non-convex problem, in particular we can often choose to go in front of, or behind, another vehicle with a local optima for each choice. By allowing only a sampled set of state values for our solution we convert the trajectory optimization into a graph search problem: find the sequence of state values, out of a finite set, that has the lowest cost, satisfy our constraints and reach a state at the end of the prediction horizon. We construct a lattice over allowed states by sampling s, d, $v = \dot{s}$ and time and define *motion primitives* that model our control actions u that transition from a state (s, d, v) at time k to another state at time $k+1$, Δt_p seconds after k, thus forming a graph representation over possible state sequences. We have used a simplified model for how the vehicle moves in the road aligned coordinate system:

$$u_1 = \ddot{s}$$
$$u_2 = \dot{d}$$

where motion primitives consists of different pre-defined accelerations $u_1 \in \mathcal{A}$ and lateral speeds $u_2 \in \mathcal{D}$. The speed, v, is restricted to be non-negative bounded above by v_{max} and d is scaled to be between -1 and 1 where -1 is one lane width to the right of the reference path and 1 one lane width to the left of it.

The set of allowed speeds, $\mathcal{V}$, is any speed reachable starting from $v = 0$ that is non-negative and less than v_{max} using any acceleration in $\mathcal{A}$ during the entire time of a planning step assuming instantaneous change of acceleration between steps. The set of allowed longitudinal positions is the set of distances traveled for time steps $k = 0, ..., N$ starting from $s = 0$ with an initial speed in $\mathcal{V}$ and using accelerations in $\mathcal{A}$. In our simplified model we allow instantaneous change in acceleration and lateral speed which is clearly not realistic. This error is assumed to amount in at most a longitudinal position error of d_{min} which is subtracted from distance calculations in collision checking. Planned lateral speeds are converted to non-holonomic motion by the execution module, see Sec. V.

A. *Cost function*

In the A^* algorithm we find a set of edges that give us the smallest total edge cost to traverse from a start node to any end node. We let the edge costs be the expected cost of applying action u from state x_k^e and arriving in state x_{k+1}^e. The result of the graph search is a state sequence minimizing the sum of costs for all state transitions:

$$\begin{aligned} c = \ & \textstyle\sum_{k=0}^{k=N-1} c_v(x_k^e, x_{k+1}^e) + c_u(u_k) + c_l(x_{k+1}^e) + \\ & c_d(x_k^e, x_{k+1}^e, p(x_{k:k+1})), \end{aligned}$$

Where c_v penalizes speed in excess of a reference speed, v_{ref}, and speeds that result in longer travel times than v_{ref}, c_u penalizes large control inputs, c_l penalizes lane changes and c_d penalizes high risk. Note that for the distribution $p(x_{k:k+1})$, $x_{k:k+1}$ includes the states $x_{k:k+1}^i$ $\forall i$. The cost terms are calculated as follows, given constant parameters $w_v, w_d, w_a, w_{\Delta d}, c_{lc}, w_C, w_M, w_{f,k}$:

$$\begin{aligned} c_v = & w_v H(v_k - v_{ref})(v_k - v_{ref})^2 + \\ & w_d H(v_{ref} - (v_{k+1} + v_k)/2)((v_{k+1} + v_k)/2 - v_{ref})^2 \end{aligned}$$

Where $H(x)$ is the Heaviside step function.

$$c_u = w_a u_1^2 + w_{\Delta d} u_2^2$$

$$c_l = \begin{cases} c_{lc} & \text{if } d_{k+1} \text{ is not in the center of a lane} \\ 0 & \text{otherwise} \end{cases}$$

$$c_d = w_C P(C_{k:k+1}) + w_M P(M_{k:k+1}) + w_{f,k} P(F_{k:k+1}) \quad (1)$$

Where $C_{k:k+1}$ is the event that there is a collision during a merge, lane change or crossing during time steps k to $k+1$, $M_{k:k+1}$ is that a merge, or lane changes is unsafe and $F_{k:k+1}$ is that we are following a vehicle to close. How these probabilities are calculated, and how weights w_C,w_M and $w_{f,k}$ set are listed in Sec. IV-C. In addition to costs there are also constraints, which invalidate a potential move by the A^* algorithm:

$$|u_2| \leq r_{\Delta d}(s_{k+1} - s_k) \quad (2)$$
$$a_{lat,k:k+1} \leq a_{lat,max} \quad (3)$$

The first constraint disallows any lateral motion that is larger than the fraction $r_{\Delta d}$ of the longitudinal motion and the second disallows any speed that results in too high lateral acceleration.

B. Heuristic

For our heuristic in the A^* algorithm we use the optimal cost of going to an end state ($k = N$) assuming that a lateral move will happen between time steps k, $k+1$ if d_k is not in the center of a lane and that there are no other vehicles. These costs are computed off line for each s, v, k combination and stored in a look-up table. This under-estimates the cost of getting to a goal state, since c_l and c_d are always larger than, or equal to zero, the heuristic is admissible.

C. Risk inference

Instead of calculating the probability of collision by checking the probability that our vehicle is at the same position as another vehicle at a future time step using the predicted future positions, $p(x^i_{0:N}), \forall i$, we model some limited future reactions implicitly by defining different events, and minimizing their probability of occurrence. In this way we model risk in terms of the outcomes of three typical occurrences in on-road driving: **i)** we are following a vehicle, **ii)** we are changing lane or merging in front of a vehicle or **iii)** we are crossing the path of a vehicle. We assume that any vehicle behind us, in our current lane, is responsible for its own safety, so if we brake to avoid a collision, we are not responsible if the vehicle behind us cannot brake. During lane changes, we must have sufficient distance between us and the vehicle behind us in the lane we are changing to, but after this point, the vehicle behind us is responsible for its safety.

In the following we describe how events, $C_{k:k+1} = \neg c_{k:k+1,OK}$, $M_{k:k+1} = \neg(f_{k:k+1,OK} \wedge lc_{k:k+1,OK})$ and $F_{k:k+1} = \neg f_{k:k+1,OK}$ are defined in eq. 1:

i) Consider the case where the vehicle in front of us would suddenly brake hard, and we are unable to brake in the next planning cycle, Δt_r seconds after the current time step, in order to avoid a future collision. If we are able to brake at time $t_k + \Delta t_r$, when the vehicle in front brakes at time t_k, then condition $f_{k:k+1,OK}$ is true.

ii) An upstream vehicle is too close when we change lane, or merge onto its lane. This happens if the new follower vehicle has to brake with a deceleration more than a threshold, a_{follow}, after a reaction time, T_r, in order to maintain a time gap, T_{gap}, to our vehicle. This is the same condition as in [16]. If the ego vehicle enters a new lane at some time in between times k, $k+1$ and the new follower vehicle is sufficiently far behind it, then the condition $lc_{k:k+1,OK}$ is true.

iii) The ego vehicle occupies the same space as another vehicle and they collide, for example during a left turn across traffic, when following or during a lane change. Here we do not consider any reaction of the other vehicle, or the ego vehicle, and calculate the probability of collision by checking the probability that any part of the other vehicle is inside the critical section at the same time as our vehicle. If there is no collision, then condition $c_{k:k+1,OK}$ is true.

At the next re-planning step in the prediction horizon we must be able to brake, however for future time steps, if $f_{k,k+1,OK}$ is false, we can avoid a collision by braking earlier. Therefore, we set $w_{f,k},\ k > 0$ to a substantially lower value than $w_{f,0}$ in eq. 1.

To calculate $P(f_{k,k+1,OK})$, $P(lc_{k,k+1,OK})$ and $P(c_{k:k+1,OK})$ we determine the closest leading and following vehicle for subdivided time steps $t_k + \Delta t_r, ..., t_{k+1}$, assuming that mean position of vehicles is sufficient to determine which vehicle is the closest. Only one lane change or merge can happen during $k : k+1$, for the other types of situations we assume probabilities are independent across time steps. **i)-iii)** correspond to indicator functions on relative distances and speeds obtained by transforming the state of follower and/or lead vehicles to the coordinate system of ego's reference path, r_e to get $s^i_{r_e}, v^i_{r_e}$. Because we assume control actions are followed exactly by ego s^e, v^e are known and $(s^i_{r_e}, v^i_{r_e})$ is distributed according to a bivariate Normal that can calculated from $p_{r(j)}(x^i_{k,r(j)})$ by adding a constant to the mean.

For condition $f_{k,k+1,OK}$ we need to calculate a_{req}: the required acceleration to avoid a collision with a vehicle in front, with state $(s^i, v^i)^T$, that brakes with constant acceleration, a_{brake}, at the current time when our vehicle has a reaction time T^e_r. If a_{req} is lower than the maximum deceleration, a_{min} we cannot avoid a collision. From $a_{req} < a_{min}$ we get the decision boundary:

$$a_{req} = l(s^i_{r_e}, v^i_{r_e}) = \frac{-(\tilde{v}^e)^2}{2\left(\tilde{s}^i - (\tilde{v}^i)^2/(2a_{brake}) - \tilde{s}^e\right)} = a_{min}$$

as long as $a_{min} \geq a_{brake}$, where $\tilde{v}, \tilde{s}$ denote speed and longitudinal position after T_r, respectively. This result has many similarities with the calculation of safe distances with time delays from [17]. Let $\mathbf{1}_{f_{k:k+1,OK}}(s^i_{r_e}, v^i_{r_e})$ be an indicator function for $a_{req} \geq a_{min}$. [1]

For condition $lc_{k:k+1,OK}$ we calculate the required acceleration of a follower vehicle to maintain a time gap T_g after we change into its lane in front of it. Assuming the new follower vehicle reacts after T^f_r the required acceleration is:

$$a_{react} = f(s^i_{r_e}, v^i_{r_e}) = \begin{cases} \frac{-(v_e - v^i_{r_e})^2}{2(\tilde{s}^e - \tilde{s}^i_{r_e} - v_e T_g)}, & \text{if } v_e - v^i_{r_e} < 0 \\ 0, & \text{otherwise} \end{cases}$$

This does consider that the initial time gap $(\tilde{s_e} - \tilde{s}^i_{r_e})/v^i_{r_e}$ is large enough however. Therefore, we use the indicator function:

$$\begin{aligned} &(1 - \mathbf{1}_{lc_{k:k+1,OK}}(s^i_{r_e}, v^i_{r_e})) = \\ &s^e - rear^e - front^i < s^i_{r_e} \leq s^e + front^e + rear^i \\ &\vee\ \ \tilde{s_e} - \tilde{s}^i_{r_e}/v^i_{r_e} < T_g \\ &\vee\ \ f(s^i_{r_e}, v^i_{r_e}) < a_{follow} \end{aligned} \tag{4}$$

which is the disjunction of the conditions: i) vehicles are in collision, where $rear^i + front^i$ is the length of vehicle i , ii) the time gap to the new follower vehicle is initially to small after the reaction time and iii) the required deceleration $-a_{react}$ is too large. [2]

Since $l(s^i_{r_e}, v^i_{r_e})$ and $f(s^i_{r_e}, v^i_{r_e})$ are nonlinear we cannot calculate the resulting distributions of the outputs $p(a_{req})$ and $p(a_{react})$ exactly. In [16] the unscented transform was used in order to approximate $p(a_{react})$ for the case of an upstream vehicle behind ego during a lane change so that

[1]We have used $a_{min} = a_{brake} = -5m/s^2$.

[2]We have used $T_g = 0.8s$, $T^f_r = 1.0s$ and $a_{follow} = -1.0m/s^2$, the same parameter values as in [16].

probability of a constraint violation $P(a_{react} < a_{follow})$ could be computed. By fitting a β distribution to the resulting mean and variance of a_{react}, and with good parameters for the unscented transform it is possible to approximate $p(a_{react})$ quite well, however there are no guarantees that the probability $P(a_{react} < a_{follow})$ is not underestimated. Another problem is that handling the different regions of the input space differently for the disjunction in Eq. 4 not trivial with the unscented transform and can lead to large errors.

Our proposed solution is to forgo estimation of distribution parameters for $p(a_{req})$, $p(a_{react})$ and instead approximate $P(\mathbf{1}_{f_{k:k+1,OK}} = 0)$ or $P(\mathbf{1}_{lc_{k:k+1,OK}} = 0)$ directly. This is done by enclosing the regions of the state space where either indicator function is zero with a piecewise linear function and evaluating how much probability mass is on each side of the piecewise linear function. This amounts to evaluating one two dimensional cumulative density function for each linear segment, j, for which there exists efficient numerical implementations[3]. We thus approximate integrals of the type:

$$\int_{x^i_{k,r(j)} \in \boldsymbol{R}^2} (1 - \mathbf{1}_{\mathcal{I}_{OK}}) \alpha_j p(x^i_{k,r(j)}) dx^i_{k,r(j)}$$

where $\mathbf{1}_{\mathcal{I}_{OK}}$ is an indicator function as:

$$\alpha_j \sum_l \int_{y_l \in R_l} y_l \; dy_l$$

where $y_l = A_l x^i_{k,r(j)} + b_l$ and R_l is the region $s_{l,0} \leq y_{l,1} < s_{l,1}, y_{l,2} > 0$. $A_l x + b_l$ is a linear function that in its output keeps $s^i_{k,r(j)}$ as one dimension of y_l and calculates the signed perpendicular distance to linear boundary l as the other dimension. For a linear transform of a normally distributed variable, the resulting variances and correlations can be computed exactly, therefore the estimation error of $P(\mathbf{1}_{\mathcal{I}_{k:k+1,OK}} = 0)$ depends only on how well the linear regions l approximate the decision boundary and the numerical accuracy of the calculation of the cumulative density of a bi-variate normal. Figure 2 shows the linear regions for calculation of $P(lc_{k:k+1,OK}), P(c_{k:k+1,OK}), P(f_{k:k+1,OK})$ where $x^e_k = (s_e, v_e)^T = (0.0, 18.0)^T$. In the center we have the collision region, to the left the region where the required deceleration of a new follower is too large and to the right the region where the required deceleration of ego, if its leader brakes, is too large. In general high speeds of a follower is bad, if it is too close, and low speeds of a leader is bad, if it is too close.

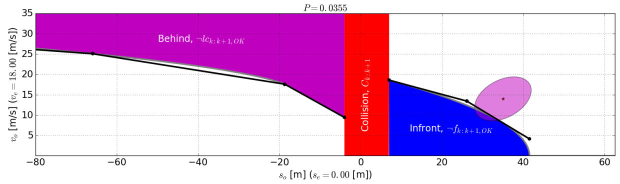

Fig. 2: Enclosing the region of invalid states for other vehicle o, colored areas, with linear regions, black lines. Distribution $p(x_o)$ is shown as a pink ellipse.

[3]https://github.com/brentonk/pbivnorm

The linear regions A_j, b_j are calculated as follows: we add four regions for the follower to close case, one region for the collision case, and two regions for the leader to close case:

- **Follower case** We first calculate the transition point (s^*_o, v^*_o) from the region where the initial time gap is too small: $s_e - s_o < T_{gap} v_o$, with linear decision boundary $s_e - s_o = T_{gap} v_o$, to the non-linear region $f(x_o, x_e) < a_{follow}$. Then we split the non-linear region into three equally spaced with respect to v_o assumed to lie in the interval $[v^*_o, v^{max}_o]$ and solve for s_o for each. Since the decision boundary of $f(x_o; x_e)$ is concave A_j, b_j will over-approximate the invalid set.
- **Collision case** This is where there is longitudinal position overlap between ego and the other vehicle: $R_j = \{s_e - r_e - f_o < s_o \leq s_e + f_e + r_o, -\infty < v_o < \infty\}$, where r_e is the distance from s_e to the rear bumper, f_e the distance to the front bumper and similarly for f_o and r_o with respect to s_o. A one-dimensional CDF is enough to calculate $\int_{y_j \in R_j} y_j dy_j$.
- **Leader case** We calculate the point where the decision boundary for $l(x_o, x_e) < a_{brake}$ intersects with $s_o = 0$ then linearize $l(x_o, x_e)$ at two points, one at the intersection of the collision region and front region, and one $4/5$ towards the root in s_o, where we solve for v_e in each case. The first region starts at the end of the collision region, the second region starts where the two gradient intersect and ends at the root. These two lines are gradients of a concave function and will always lie above it so that we over approximate the invalid region.

Where $x_e = (s^e, v^e)$ and $x_o = (s^i_{r_e}, v^i_{r_e})$ for the time step.

V. Implementation

The planner is executed in a receding horizon fashion, with re-planning period Δt_r, where the execution module uses the previously calculated plan in order to calculate control inputs for the ego vehicle. In our simulations we used a PID controller, where we match the speed of the planned trajectory, for the throttle/brake of ego and a pure pursuit controller [18] for the steering angle.

We prune solutions where the ego speed oscillates too rapidly by limiting the ego acceleration to change sign, not counting the first change, at most every fourth search step. Also, only at most one lane change every planning is allowed. Only changes of acceleration where $|u_{1,k+1} - u_{1,k}| < \tau_{jerk}$ and lane changes with $u_{1,k} < \tau_{lc_a}$ are allowed. We have used $\tau_{jerk} = \tau_{lc_a} = 2.0 m/s^2$. Other parameters are listed in Tab. I.

TABLE I: Parameters for trajectory planning

$\mathcal{A} = \{-2.5, -1.25, 0.0, 1.25, 2.5\}$
$\mathcal{D} = \{-0.4, -0.2, 0.0, 0.2, 0.4\}$

N	$= 10$	Δt_p	$= 1.0$	Δt_r	$= 0.2$
w_v	$= 0.01$	w_d	$= 0.1$	w_a	$= 0.1$
$w_{\Delta d}$	$= 0.5$	c_{lc}	$= 0.5$	d_{min}	$= 2.0$
w_M	$= 5.0e3$	w_C	$= 1.0e4$		
$w_{f,0}$	$= 1.0e4$	$w_{f,1:N}$	$= 100$		

We achieved a mean run time of $18.6ms$ and a maximum run time of $152.0ms$ for the 102778 planning cycles in the simulated scenarios in Sec. VI using a C++ implementation running on an Intel i7-5930K. We see that the worst case run time is below the planning cycle time of 200 ms and that the mean run time is much lower.

VI. Simulations

A. Highway scenario

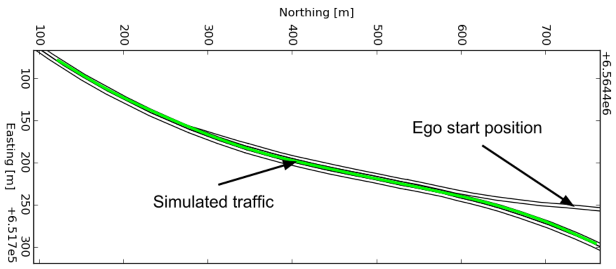

Fig. 3: Simulation scenario, a highway on-ramp near Stockholm.

We tested the trajectory planning algorithm in a simulated high way on-ramp scenario, shown in Fig. 3. The ego vehicle starts on the on-ramp and should adjust its speed and find a suitable gap to enter right lane of the highway. It can do so by either changing lane or by continuing on the on-ramp until this merges with the right lane. Traffic is simulated using the Intelligent Driver Model (IDM) for other vehicles, with random parameters, random initial speeds and random initial spacing as in [19]. We simulate two different traffic types, cars and trucks, with probability of occurrence 0.8 and 0.2, respectively. The mean reference speed for the IDM model for cars was set to $28m/s$, and for trucks to $20m/s$. Two different settings for traffic density was used, with different uniform distributions for initial spacing of vehicles, between $50-150m$ for traffic density A and between $50-90m$ for traffic density B. The reference speed for the ego vehicle was $22.2m/s$ $(80km/h)$.

The input to the planning algorithm is the current state of the ego vehicle without noise, a digital road map, and the output of a simulated Kalman filter that tracked the simulated vehicles using a constant speed model, assuming the path of each other vehicle is known. Predictions are generated using a constant velocity model, where we add prediction noise with variance ϵ m^2/s^2 to the speed. For a base line comparison we run the proposed method with deterministic predictions, denoted “det”, where only the mean is used, so cost terms in Eq. 1 are either 0 or the full cost.

We have evaluated the planner with different settings of the prediction noise ϵ for 100 simulated on-ramp merge scenarios for each of two different traffic densities, with statistics shown in Tab. II. Each of the 100 simulated scenarios for each traffic density has different random parameters for each other simulated vehicle, however when re-doing the experiment for different settings of ϵ or the “det” case we use the same seed in the simulation so that other simulated vehicle’s behavior parameters and initial states are identical. The risk associated with each merge attempt is evaluated by comparing the time gap of the new follower, $T_{g,f}$, of the ego vehicle at the time the ego vehicle enters the right lane of the highway and the time gap to its new leader, $T_{g,l}$. Median and minimum values are listed in Table III, along with the percentage of aborted merge attempts. In the “det” case there were two simulated collision for traffic density A and one for B. All of these happened when a vehicle parallel to the ego vehicle slowed down for a preceding vehicle just before on ramp ends. We can also see that an increase in the prediction noise parameter ϵ leads to more conservative behavior, with more aborted merge attempts. By modeling uncertainty in predictions we can compensate for a prediction model that does not always give good predictions, a constant velocity model for agents simulated using the IDM model in this case. However, this does lead to somewhat overly conservative behavior.

TABLE II: Traffic statistics

Traffic density	$\bar{v}$ [m/s]	σ_v [m/s]	$\bar{d}$ [m]	σ_d [m]
A	22.2	2.9	86.7	51.6
B	20.9	2.6	61.8	37.1

TABLE III: Simulation results, time gap in seconds

Traffic density A					
Exp.	abort %	med($T_{g,r}$)	min($T_{g,r}$)	med($T_{g,f}$)	min($T_{g,f}$)
$\epsilon = 0.2$	2.0	3.83	1.57	1.76	0.65
$\epsilon = 0.1$	3.0	3.68	1.57	1.81	0.65
det	1.0	2.92	1.15	1.56	0.31
Traffic density B					
Exp.	abort %	med($T_{g,r}$)	min($T_{g,r}$)	med($T_{g,f}$)	min($T_{g,f}$)
$\epsilon = 0.2$	12.0	4.47	1.74	1.76	0.90
$\epsilon = 0.1$	7.0	4.12	1.74	1.74	0.76
det	4.0	2.57	0.66	1.57	0.34

B. Comparison with POMDP

In [10] a scenario was proposed where the ego vehicle is approaching an intersection, where two other vehicles are present, one to the left and one to the right. The vehicle to the left was modeled as having two potential options for future routes: it could either continue straight or turn right, while the vehicle to the right would continue straight through the intersection. If the vehicle to the left makes its turn, the coast is clear for the ego vehicle to enter the intersection before the vehicle to the right, but if it does not the ego vehicle must slow down to let it pass and might therefore not be able to enter the intersection before the vehicles to the right. This scenario is shown in Fig. 4.

The ego vehicle has access to a classifier that will, at every planning cycle, output the probability that the left agent takes the turn or not, $P(turn)$. We have used the same classification model as in [10]. As in [10] the scenario was simulated with the vehicle to the left making the turn, however before this happens the future situation is ambiguous

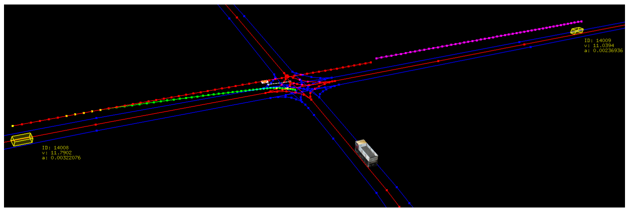

Fig. 4: Recreated simulated scenario from [10], the ego vehicle, bottom, should turn to the left in the intersection, avoiding the other two vehicles. The mean of future predicted positions of the other vehicles are shown as dotted lines, where the height corresponds to the mean predicted speed.

and the classifier will return non-zero probabilities for either case. We simulated the other vehicles using the intelligent driver model, taking into account curvature by modifying the reference speed.

To compare the proposed method with the POMDP method from [10] we ran both for exactly the same simulated scenario. Here the vehicle to the left travels at first at $11m/s$ towards the intersection, until it starts to decelerate at about $35m$ before the turn, the vehicle to the right travels at a constant speed of $9m/s$. The proposed method has access to the result of the classifier and the same motion model for the other agents, except that interactions are ignored, so agents roughly follow the curvature based reference speed for their path. This motion model is used to estimate $p(x^i_{0:N})$, by extracting the mean and covariance at each time step from a Monte Carlo simulation using 2000 samples.

We have used the same motion and observation model for our implementation of the Monte Carlo based POMDP approach as in [10] except for the calculation of the desired velocity based on curvature, v_{ref}^{POMDP}, which no longer includes an immediate jump up to the speed limit after a curve was passed. We solve the following linear program to calculate v_{ref}^{POMDP}, [20]:

$$\begin{array}{ll} \min & \mathbf{1}^T(\mathbf{w}_{max} - \mathbf{w}) \\ \text{s.t.} & 2a_{min}\mathbf{1} \leq D\mathbf{w} \leq 2a_{max}\mathbf{1} \\ & \qquad 0 \leq \mathbf{w} \quad \leq \mathbf{w}_{max} \end{array} \tag{5}$$

where $w_i = v_i^2$ for fixed position along a path at arc length, s_i, $w_{max,i} = v_{max,i}^2$ is the squared maximum allowed speed based on a maximum lateral acceleration from the curvature at this point and the speed limit and D a finite difference operator. We used $-a_{min} = a_{max} = 1.0ms^{-2}$ which corresponds well to the plots in [10] and $a_{lat,max} = 1.5ms^{-2}$. In Fig. 5, the reference speed is shown for the left turn that the ego vehicle makes. The actions for the ego vehicle are $a_i \in \{-1.0, 0, 1.0\}, ms^{-2}$, which corresponds to the plots in [10], and the reward function is the same. To sample new particles, in case there are not enough particles in the root of the belief tree at the start of a planning cycle we implemented a rejection sampling scheme. Here new particles are sampled around the current measurement with a randomly selected parent from the previous root and rejected with a probability corresponding the probability of the parents route.

Fig. 5 shows, in the top plot, the speed profile planned by the POMDP planner, the first speed in each speed profile planned by the proposed method, and the POMDP reference speed, v_{ref}^{POMDP}. The bottom plot shows the output of the classifier. If we run the simulation but let the classifier output the ground truth, both the proposed method and the POMDP planner will enter the intersection before the vehicle to the right, these speed profiles are described with the suffix "(GT)" in Fig 5. The proposed method has access to a larger number of acceleration options, and also employs these to go faster towards the intersection.

When the route choice of the vehicle to the left is ambiguous, we see that the POMDP method first accelerates to reach v_{ref}^{POMDP} then slows down as if waiting for the vehicle to the right, until it starts accelerating to go through

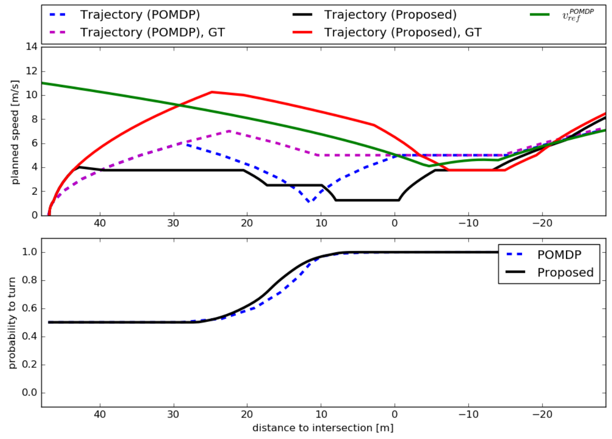

Fig. 5: Executed speed profiles, and estimated probability of the left vehicle turning.

the intersection, anticipating the turn of the vehicle to the left. This behavior is qualitatively consistent with the results in [10]. The proposed method at first plans speed profiles that will enter the intersection just after the vehicle to the left, but before the vehicle to the right. But then, when the speed of the vehicle to the left has reduced to $10m/s$ it starts to plan trajectories that wait for the vehicle to the right to pass. These can be seen in the top plot for distance values $[20, 0]$. As there are no more vehicles coming from the left, the planner then plans a trajectory that will slowly enter the intersection after the vehicle to the right, starting to follow it at a distance of roughly $6m$.

To compare the level of risk for both methods we check if the ego vehicle can either brake such that the critical section is clear before it is reached or accelerate such that the ego vehicle goes through the intersection before the vehicle to the left. This is done using the maximum modeled deceleration and acceleration of each method. If the ego vehicle cannot avoid the critical section, by the assumption that our classifier outputs the correct probability we have a crash probability of $1.0 - P(turn)$. For the proposed method, which chooses a conservative plan where we wait for the vehicle to the left to clear the intersection, this probability is zero. For the POMDP method it is at most 0.2% at simulation time $14s$. There are two main contributing factors to this behavior: the cost of collision for the POMDP method is on the order of completing a crossing of the intersection [10] and for rare events we have particle deprivation. In Fig. 6 we can see that no particles survive in the root belief node which model the rare case (0.2%) of a vehicle continuing straight at time $14s$. Contributing factors to particle deprivation is the large branching factor of the belief tree and that the UCB1 heuristic mostly selects actions that cover promising regions (with high expected reward) of the belief space. With a small number of particles we can not hope to represent the belief distribution well. This problem becomes more severe at deeper levels of the tree, such that rare future situations may never affect the expected future rewards for different actions.

In the proposed method we don't try to model complex future belief distributions that depend on what sequence of

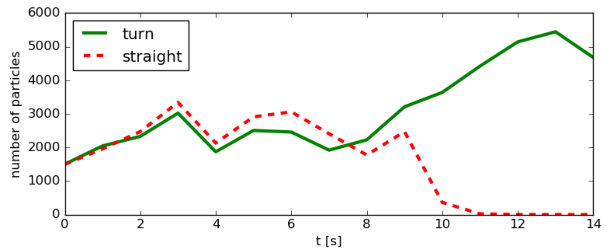

Fig. 6: Particle counts for the root belief, at different simulation times.

actions we choose and instead assume that future belief states are captured by $p(x_{t:t+T})$. The risk of a future trajectory is approximated with the risk assessment functions in Sec. IV-C rather than by a Monte Carlo procedure where the probability of collision is estimated. This allows us to achieve deterministic results (i.e. the same initial ego state and $p(x^i_{0:N})$ will give the same resulting plan) and manageable run times. This assumption does mean that many possible futures are not considered, and does tend towards conservative behavior, but because we re-plan often we can more easily adapt to a changing situation where our prediction models might be slightly wrong.

The proposed method can comfortably be run at $5Hz$ with a maximum run time for the scenario of $38ms$, and scales well when several other agents are included. The implementation by [10] of the POMDP method had a re-planning rate of $1Hz$ and could not maintain this run time if more vehicles were added to the simulation.

VII. Conclusions and future work

Our major contribution is a trajectory planning approach for on-road scenarios where we use a graph search approximation and uncertain predictions of other vehicles are accounted for by a novel inference technique that allows efficient calculation of the probability of dangerous outcomes for set of modeled situation types. We have also compared this trajectory planning approach with a Monte Carlo based POMDP solver and seen that while the POMDP solver can plan more complex behavior that is less conservative than the proposed approach, it suffers from particle deprivation that can lead to risky behavior. This is because the Monte Carlo method will sometimes not have sufficiently many samples for rare, but dangerous future situations. Our approach instead uses a parametric description of future belief states and does not suffer from this problem, and has significantly better run times.

For the highway merge scenario we used simplistic constant velocity predictions, we also do not consider sensor range and occlusions. Future work will consider more sophisticated prediction models and we believe that the geometric inference approach proposed in this paper could lend itself well to model sensor range and occlusion limitations. For instance, to model the risk posed by a possible vehicle beyond our sensor range we could integrate the constraint violation probability given a uniform probability of speeds and positions of the unseen vehicle.

References

[1] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[2] C. Hubmann, M. Aeberhard, and C. Stiller, "A generic driving strategy for urban environments," in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*. IEEE, 2016, pp. 1010–1016.

[3] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for berthaa local, continuous method," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 450–457.

[4] J. Nilsson, J. Silvlin, M. Brannstrom, E. Coelingh, and J. Fredriksson, "If, when, and how to perform lane change maneuvers on highways," *IEEE Intelligent Transportation Systems Magazine*, vol. 8, no. 4, pp. 68–78, 2016.

[5] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4889–4895.

[6] M. Bahram, A. Wolf, M. Aeberhard, and D. Wollherr, "A prediction-based reactive driving strategy for highly automated driving function on freeways," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 400–406.

[7] S. Ulbrich and M. Maurer, "Towards tactical lane change behavior planning for automated vehicles," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. IEEE, 2015, pp. 989–995.

[8] M. Althoff, O. Stursberg, and M. Buss, "Model-based probabilistic collision detection in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 299–310, 2009.

[9] W. Zhan, J. Chen, C.-Y. Chan, C. Liu, and M. Tomizuka, "Spatially-partitioned environmental representation and planning architecture for on-road autonomous driving," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017, pp. 632–639.

[10] C. Hubmann, M. Becker, D. Althoff, D. Lenz, and C. Stiller, "Decision making for autonomous driving considering interaction and uncertain prediction of surrounding vehicles," in *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017, pp. 1671–1678.

[11] W. Liu, S.-W. Kim, S. Pendleton, and M. H. Ang, "Situation-aware decision making for autonomous driving on urban road using online pomdp," in *Intelligent Vehicles Symposium (IV), 2015 IEEE*. IEEE, 2015, pp. 1126–1133.

[12] D. Klimenko, J. Song, and H. Kurniawati, "Tapir: A software toolkit for approximating and adapting pomdp solutions online," in *Proceedings of the Australasian Conference on Robotics and Automation, Melbourne, Australia*, vol. 24, 2014.

[13] N. E. Du Toit and J. W. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 101–115, 2012.

[14] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 987–993.

[15] Q. Tran and J. Firl, "Online maneuver recognition and multimodal trajectory prediction for intersection assistance using non-parametric regression," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*. IEEE, 2014, pp. 918–923.

[16] S. Ulbrich and M. Maurer, "Situation assessment in lane change behavior planning for automated vehicles," in *Proceedings of the 18th International IEEE Annual Conference on Intelligent Transportation Systems (ITSC 2015)*, 2015.

[17] M. Althoff and R. Lösch, "Can automated road vehicles harmonize with traffic flow while guaranteeing a safe distance?" in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*. IEEE, 2016, pp. 485–491.

[18] O. Amidi and C. E. Thorpe, "Integrated mobile robot control," in *Fibers' 91, Boston, MA*. International Society for Optics and Photonics, 1991, pp. 504–523.

[19] E. Ward, N. Evestedt, D. Axehill, and J. Folkesson, "Probabilistic model for interaction aware planning in merge scenarios," *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 2, pp. 133–146, 2017.

[20] P. F. Lima, M. Trincavelli, J. Mårtensson, and B. Wahlberg, "Clothoid-based speed profiler and control for autonomous driving," in *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*. IEEE, 2015, pp. 2194–2199.