

Motion Planning under Uncertainty for On-Road Autonomous Driving

Wenda Xu, Jia Pan, Junqing Wei, and John M. Dolan

Abstract—We present a motion planning framework for autonomous on-road driving considering both the uncertainty caused by an autonomous vehicle and other traffic participants. The future motion of traffic participants is predicted using a local planner, and the uncertainty along the predicted trajectory is computed based on Gaussian propagation. For the autonomous vehicle, the uncertainty from localization and control is estimated based on a Linear-Quadratic Gaussian (LQG) framework. Compared with other safety assessment methods, our framework allows the planner to avoid unsafe situations more efficiently, thanks to the direct uncertainty information feedback to the planner. We also demonstrate our planner’s ability to generate safer trajectories compared to planning only with a LQG framework.

I. INTRODUCTION

Autonomous vehicle technology has made considerable progress over the past few years [1], [2], [3]. Its main purpose is to improve driving safety by decreasing traffic accidents, which are mostly caused by human mistakes or distractions. However, autonomous vehicle technology also poses new challenges. One important problem is how to plan a safe trajectory for a vehicle, given uncertain knowledge about the surrounding environment and the vehicle’s own state.

Failure to consider such uncertainty may make autonomous vehicles unsafe. For example, if a vehicle uses a low-accuracy GPS for localization and the planning algorithm does not take that into account, then the vehicle may follow a dangerous path that may collide with obstacles. Besides the uncertainty related to the autonomous vehicle itself, including uncertainty in perception, localization and control, the autonomous vehicle system also needs to resolve the uncertainty caused by other traffic participants. For example, let us consider the scenario where a (non-autonomous) vehicle is changing lanes just three meters in front of an autonomous vehicle. If the planning algorithm can predict the movement of that merging vehicle, the autonomous vehicle is more likely to execute a safe and smooth trajectory rather than performing sudden braking or even crashing into the merging car. As a result, in order to improve the safety of the autonomous vehicle system, the vehicle planning algorithm needs to consider both kinds of uncertainties simultaneously.

This work was supported by NSF Grant CNS1035813

Wenda Xu and Junqing Wei are with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA {wendax, junqingw}@cmu.edu

Jia Pan is with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA 94709, USA jia.pan@berkeley.edu

John M. Dolan is with the Department of Electrical and Computer Engineering and Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA jmd@cs.cmu.edu

In this paper, we propose a planning framework that is able to consider both the uncertainty caused by autonomous vehicles and other traffic participants for better driving safety. We first generate candidate trajectories based on a spatio-temporal lattice [4] and then search for an optimal trajectory among them. When evaluating the quality of a trajectory candidate, we consider the uncertainties caused by both the autonomous vehicle itself and other traffic participants. For the autonomous vehicle itself, we use a Linear-Quadratic Gaussian (LQG) framework [5] to estimate the uncertainty when executing a given candidate trajectory, given the noise characteristics of the localization and control. Similar to [5], we incorporate the state measurements in the future as stochastic variables and the result is a distribution for the state of each candidate trajectory. For other traffic participants, we first estimate the control inputs they will choose using a local planner and then predict a distribution for their future states using a Kalman filter, where the estimated control inputs and future observations are used as inputs. In the search phase, the quality of a candidate trajectory is evaluated based on a cost function related to the state distributions of both the candidate trajectory and other traffic participants. Using this new trajectory quality evaluation framework, the planning algorithm can provide a safer trajectory for an autonomous vehicle.

In this work, we assume an autonomous vehicle is moving in a traffic scene in which all other vehicles are driven by humans. The autonomous vehicle knows its current state distribution from its localization system, and other vehicles’ current state, e.g. position and speed, from the perception system. A static map including all static obstacles is also generated by the perception system. We also assume that the controller on the autonomous vehicle is able to follow any planned trajectory. The objective of this paper is to find the optimal trajectory minimizing the risk from perception and motion uncertainty.

The rest of the paper is organized as follows. We survey related work on planning under uncertainty for autonomous vehicles in Section II. In Sections III and IV, we introduce the approach to estimate the uncertainty of traffic participants and the autonomous vehicle, respectively. The planning algorithm for both traffic participants and autonomous vehicle is presented in Section V. Results and conclusions are given in Sections VI and VII.

II. RELATED WORK

Planning algorithms have been widely applied to car-like robots. A lattice planner [6] for an autonomous vehicle has been used in low-density, low-speed traffic. McNaughton et

al. [7] proposed a planner designed for highway driving. It extends path planning to trajectory planning based on a spatio-temporal lattice. Xu et al. [4] and Gu et al. [8] improved the performance of the spatio-temporal lattice planner by using posterior optimization and focused search, respectively. These traditional planning methods assume full knowledge of the perception and deterministic motion.

From the planning perspective, the uncertainty comes not only from the localization and control of an autonomous vehicle, but also from the sensing and prediction of obstacles in the environment. A Partially Observable Markov Decision Process (POMDP) [9] allows computing the control policy over the space of the belief state, which is a probability distribution over all the possible states. However, POMDP is known to be computationally expensive and scales poorly when the problem dimension increases. LQG-based motion planning algorithms [5], [10] estimate the vehicle's states based on a Kalman filter, and capture the uncertainty during execution using a Linear-Quadratic Regulator (LQR) controller, which is then combined with a sampling-based search algorithm to find the optimal trajectory. Compared to POMDP, LQG-based methods have lower computational complexity and also scale better with the number of states. However, these methods exclude the uncertainty due to the movement of dynamic obstacles.

A verification technique was proposed [11] to check safety for an autonomous vehicle based on the reachable set. The reachable set represents all the possible states that the vehicle may visit given a planned trajectory and the initial state. If the reachable set of an autonomous vehicle has no overlap with the reachable set of obstacles, the planned trajectory is marked as safe. In [12], the possible behaviors of traffic participants are also predicted and the reachable sets of traffic participants are computed based on a Markov chain. The limitation of this safety verification method is that it relies on the planned trajectory rather than being embedded in the planner. If a trajectory exported by the planner is marked as unsafe by the safety verification module, the planner must perform replanning. Moreover, the planner has no knowledge of why this trajectory is unsafe, and such indirect feedback may not help the planner to find a safe trajectory. A point-based Markov Decision Process (QMDP) framework is presented in [13] to consider all the uncertainty at the behavior level. It helps the autonomous vehicle to avoid potentially unsafe situations by making better decisions at the behavior level. However, this approach cannot be directly applied to vehicle planning due to its neglect of vehicle kinematics.

A Monte Carlo sampling-based probabilistic framework for reasoning about traffic participants' future motion is proposed in [14]. Given the initial states of all objects in the environment, this method simulates all the possible movements for each object, and outputs a probability density function for its future states. While the reachability analysis approach contains more complete future states than the Monte Carlo based-approach, the latter provides better results when computing crash probabilities [15]. The Monte Carlo-

based approach is suitable for predicting the movement of other vehicles, but is not suitable for computing the uncertainty of the autonomous vehicle itself. This is because it models the control inputs as stochastic variables, whereas for autonomous vehicles, the control is known beforehand.

In summary, the contribution of this paper is that we propose a planner for an autonomous vehicle that can consider not only the uncertainty from the autonomous vehicle itself, but also the uncertainty of traffic participants. Our results are safer than our previous deterministic planner and other planners that only consider the uncertainty from localization and control.

III. UNCERTAINTY PREDICTION OF TRAFFIC PARTICIPANT

A. Vehicle Model

In this paper, we only consider motor vehicles as traffic participant. The state of a car is denoted as $\mathbf{x} = [x \ y \ \theta]$, and the control input is defined as $\mathbf{u} = [v \ \kappa]$, where x and y are the position, θ is the orientation, κ is the curvature, and v is the speed.

Given discretization time step Δt , the vehicle dynamics are described by:

$$\mathbf{x}_t = A_t \mathbf{x}_{t-1} + B_t \mathbf{u}_{t-1} \quad (1)$$

$$A_t = \begin{bmatrix} 1 & 0 & -v_{t-1} \sin \theta_{t-1} \Delta t \\ 0 & 1 & v_{t-1} \cos \theta_{t-1} \Delta t \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$B_t = \begin{bmatrix} \cos \theta_{t-1} \Delta t & 0 \\ \sin \theta_{t-1} \Delta t & 0 \\ 0 & v_{t-1} \Delta t \end{bmatrix} \quad (3)$$

The future trajectories of traffic participants are determined by their current state and future control input. Suppose there are N_c vehicles in the scene, and their control inputs are denoted as \mathbf{u}^i , $i \in [1, N_c]$. For each car i , given its initial state \mathbf{x}_0^i , its future trajectory can be computed by applying control input \mathbf{u}_t^i ($t \in [0, T]$, where T represents the prediction time interval) to the vehicle dynamics (1).

The current states of other cars can be measured by the perception system of the autonomous vehicle. However, the future control inputs of traffic participants are unknown. In this paper, we propose a local planner that predicts the future motion of traffic participants. For each traffic participant, motion primitives are first designed based on typical human driving behavior, and then they are evaluated using a cost function. Finally, the optimal motion primitive is chosen as the best estimation of this traffic participant's future motion. The details are presented in Section V-B.

B. Gaussian Propagation

The vehicle dynamic model generally can be written or linearized as a time-varying linear model, and the state is generally partially observable. The discrete linear system is given as:

$$\mathbf{x}_t = A_t \mathbf{x}_{t-1} + B_t \mathbf{u}_{t-1} + \mathbf{w}_t, \quad \mathbf{w}_t \sim N(0, W_t) \quad (4)$$

$$\mathbf{z}_t = C_t \mathbf{x}_{t-1} + \mathbf{v}_t, \quad \mathbf{v}_t \sim N(0, V_t) \quad (5)$$

where \mathbf{x}_t is the state vector, \mathbf{u}_t is the control input vector, \mathbf{z}_t is the measurement vector, \mathbf{w}_t is the process noise, and \mathbf{v}_t is the measurement noise. The original Kalman filter equations are given as follows:

The process update is:

$$\hat{\mathbf{x}}_t^- = A_t \hat{\mathbf{x}}_{t-1} + B_t \mathbf{u}_{t-1} \quad (6)$$

$$\Sigma_t^- = A_t \Sigma_{t-1} A_t^T + W_t \quad (7)$$

The measurement update is:

$$L_t = \Sigma_t^- C_t^T (C_t \Sigma_t^- C_t^T + V_t)^{-1} \quad (8)$$

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^- + L_t (\mathbf{z}_t - C_t \hat{\mathbf{x}}_t^-) \quad (9)$$

$$\Sigma_t = (I - L_t C_t) \Sigma_t^- \quad (10)$$

where L_t is the Kalman gain, $\hat{\mathbf{x}}_t^-$ is the priori state estimate, $\hat{\mathbf{x}}_t$ is the posteriori state estimate, Σ_t is the posteriori estimate error covariance.

In our case, the control input \mathbf{u}_t and measurement \mathbf{z}_t are both unknown at the time of planning. \mathbf{u}_t is predicted as the best motion primitive based on the cost function, and \mathbf{z}_t is assumed as a Gaussian distribution. Given initial state \mathbf{x}_0 and control input \mathbf{u}_t , the state vector along the future trajectory is computed as $\tilde{\mathbf{x}}_t$. The final estimate of \mathbf{x}_t is :

$$P(\mathbf{x}_t) = N(\tilde{\mathbf{x}}_t, \Sigma_t) \quad (11)$$

IV. UNCERTAINTY PROPAGATION FOR THE AUTONOMOUS VEHICLE

The vehicle dynamic model for the autonomous vehicle is the same as that of the traffic participants, as defined in (4) and (5). Unlike the traffic participants, we can directly control the autonomous vehicle. For this time-varying linear system, a LQR controller can be designed to track the planned trajectory. Along the planned trajectory, the state vector is denoted as $\tilde{\mathbf{x}}_t$, the control input vector is denoted as $\tilde{\mathbf{u}}_t$, and the measurement vector is denoted as $\tilde{\mathbf{z}}_t$. The tracking problem can be converted to a regulator problem by defining:

$$\tilde{\mathbf{x}}_t = \mathbf{x}_t - \hat{\mathbf{x}}_t \quad (12)$$

$$\tilde{\mathbf{u}}_t = \mathbf{u}_t - \hat{\mathbf{u}}_t \quad (13)$$

$$\tilde{\mathbf{z}}_t = \mathbf{z}_t - \hat{\mathbf{z}}_t \quad (14)$$

where $\tilde{\mathbf{x}}_t$, $\tilde{\mathbf{u}}_t$, and $\tilde{\mathbf{z}}_t$ represent the deviation between the planned trajectory and the trajectory during execution.

The tracking system can be formulated as:

$$\tilde{\mathbf{x}}_t = A_t \tilde{\mathbf{x}}_{t-1} + B_t \tilde{\mathbf{u}}_{t-1} + \tilde{\mathbf{w}}_t, \quad \tilde{\mathbf{w}}_t \sim N(0, \tilde{W}_t) \quad (15)$$

$$\tilde{\mathbf{z}}_t = C_t \tilde{\mathbf{x}}_{t-1} + \tilde{\mathbf{v}}_t, \quad \tilde{\mathbf{v}}_t \sim N(0, \tilde{V}_t) \quad (16)$$

Given constant state and input weight matrices Q and R , a cost function is defined as:

$$J = \sum_{t=0}^T (\tilde{\mathbf{x}}_t^T Q \tilde{\mathbf{x}}_t + \tilde{\mathbf{u}}_t^T R \tilde{\mathbf{u}}_t) \quad (17)$$

where the time span T is obtained during trajectory generation (see Section V-A). The optimal control policy minimizing the cost function is obtained by:

$$\tilde{\mathbf{u}}_t = -K_{t+1} \tilde{\mathbf{x}}_t \quad (18)$$

where K_t is computed iteratively backwards based on the dynamic Riccati equation:

$$K_t = (R + B_t^T P_t B_t)^{-1} B_t^T P_t A_t \quad (19)$$

$$P_{t-1} = Q + A_t^T P_t A_t - A_t^T P_t B_t K_t \quad (20)$$

$$P_N = Q \quad (21)$$

The Kalman filter process equations for this trajectory tracking system are the same as (6) – (10). $\tilde{\Sigma}_t^-$ is the covariance after the process update, $\tilde{\Sigma}_t$ is the posteriori covariance after the measurement update, and \tilde{L}_t is the Kalman gain. According to [5], the final distribution for the state vector is:

$$P(\mathbf{x}_t) = N(\tilde{\mathbf{x}}_t, \tilde{\Sigma}_t + \Lambda_t) \quad (22)$$

where

$$\Lambda_t = (A_t - B_t K_t) \Lambda_{t-1} (A_t - B_t K_t)^T + \tilde{L}_t C_t \tilde{\Sigma}_t^- \quad (23)$$

V. TRAJECTORY PLANNING

A. Trajectory Generation

In this paper, the trajectory generation approach is adapted from the previous paper of the authors [4].

Given a reference path, which generally is the center line of a lane, the road coordinate is introduced as (ℓ, d) , where ℓ and d are the longitudinal and lateral offset of the reference path, respectively. A state (ℓ, d) with $d = 0$ denotes a point on the reference path. Endpoints are sampled in both the longitudinal and lateral directions. We sample N_ℓ layers along the reference path, and N_d points in each layer perpendicular to the reference path. The state of a sampled point (ℓ, d) is defined as:

$$x(\ell, d) = x(\ell, 0) + d \cos(\theta(\ell, 0) + \frac{\pi}{2}) \quad (24)$$

$$y(\ell, d) = y(\ell, 0) + d \sin(\theta(\ell, 0) + \frac{\pi}{2}) \quad (25)$$

$$\theta(\ell, d) = \theta(\ell, 0) \quad (26)$$

$$\kappa(\ell, d) = (\kappa(\ell, 0)^{-1} - d)^{-1} \quad (27)$$

Paths are generated by connecting endpoints in neighbouring layers using cubic curvature polynomials. The curvature of the path is a cubic polynomial of arc length.

$$\kappa(s) = \gamma_0 + \gamma_1 s + \gamma_2 s^2 + \gamma_3 s^3 \quad (28)$$

The details of solving the parameters for $\kappa(s)$ can be found in [4]. The length of the path is denoted as s_f . For

each path segment, the speed profile can be generated by discretizing the speed v_f and acceleration a_f at the endpoint. To reduce the search space, we assume $a_f = 0$. The number of discretized speeds is denoted as N_v .

The speed profile for each path is defined as a cubic polynomial of time:

$$v(t) = \rho_0 + \rho_1 t + \rho_2 t^2 + \rho_3 t^3 \quad (29)$$

which is different from [4], because speed as a function of time is easier for a controller to follow than speed as a function of arc length.

For each trajectory, given the speed v_f and acceleration a_f at the endpoint, the time span T can be obtained:

$$T = \begin{cases} -\frac{c}{b}, & \text{if } a = 0 \\ \frac{-b + \sqrt{b^2 - 4ac}}{2a}, & \text{if } b^2 - 4ac \geq 0 \\ \emptyset, & \text{if } b^2 - 4ac < 0 \end{cases} \quad (30)$$

$$a = (a_0 - a_f)/12 \quad (31)$$

$$b = (v_0 + v_f)/2 \quad (32)$$

$$c = -s_f \quad (33)$$

The parameters for (29) are then computed as follows:

$$\rho_0 = v_0 \quad (34)$$

$$\rho_1 = a_0 \quad (35)$$

$$\rho_2 = \frac{1}{T^2} (3(v_f - v_0) - (a_f + 2a_0)T) \quad (36)$$

$$\rho_3 = \frac{1}{T^3} (-2(v_f - v_0) + (a_f + a_0)T) \quad (37)$$

We obtain state as a function of arc length $[x(s) \ y(s) \ \theta(s) \ \kappa(s)]$ from the path generation method, and obtain speed as a function of time $v(t)$ from the speed generation method. The arc length of time $s(t)$ can be computed by integration of $v(t)$. The state as a function of time $[x(t) \ y(t) \ \theta(t) \ \kappa(t)]$ follows from substituting $s(t)$ into state as a function of arc length $[x(s) \ y(s) \ \theta(s) \ \kappa(s)]$.

B. Prediction for Traffic Participants

As mentioned in Section III-A, traffic participants' motion is predicted by a local planner. For traffic participants, we choose $N_\ell = 1$, which means we only consider one layer in planning. The reason is that, the longer we predict, the higher probability the prediction is unreliable. Moreover, it helps to reduce the computation time.

When we predict the motion for a traffic participant, other vehicles' motion also needs to be considered beforehand, which leads to a chicken-and-egg problem. In this paper, we make a simplified assumption that when we predict the motion for a traffic participant, all the other vehicles (including the autonomous vehicle) will keep moving with current speeds and have no uncertainty on their positions.

We use the optimal trajectory from the local planner as the prediction of a traffic participant's future motion. Once the future trajectory is predicted, the distribution of the state along this trajectory can be computed using a Kalman filter, as described in Section III.

Algorithm 1: Dynamic Programming

```

1 foreach  $\ell_i, i = 0$  to  $N_\ell$  do
2   foreach  $d_j, j = 0$  to  $N_d$  do
3     foreach  $v_k, k = 0$  to  $N_v$  do
4       Form a state  $s_p \leftarrow (\ell_i, d_j, v_k)$ 
5       foreach  $d_m$  on  $\ell_{i+1}, m = 0$  to  $N_d$  do
6         foreach  $v_n, n = 0$  to  $N_v$  do
7           Form a state  $s_q \leftarrow (\ell_{i+1}, d_m, v_n)$ 
8           Form a trajectory  $\tau$  from  $s_p$  to  $s_q$ 
9           if  $c(s_q) > c(s_p) + c(\tau)$  then
10              $c(s_q) \leftarrow c(s_p) + c(\tau)$ 
11              $\text{Predecessor}(s_q) \leftarrow s_p$ 
12           end
13         end
14       end
15     end
16   end
17 end

```

C. Planning for Autonomous Vehicle

For the autonomous vehicle, we sample N_ℓ ($N_\ell \geq 3$) layers, N_d vertices in each layer, and N_v speeds at each vertex. The computational complexity for a straightforward exhaustive search is $O((N_d N_v)^{N_\ell})$. Instead of an exhaustive search, we use dynamic programming (Algorithm 1) to explore the search space, which reduces the computational order to $O(N_\ell (N_d N_v)^2)$.

In the search process, the state is defined as $s = (\ell, d, v)$. The cost of a state $c(s)$ represents the accumulated costs of the trajectories connecting the starting state with this state. The cost of a trajectory $c(\tau)$ is computed based on the cost functions defined in Section V-D. During the dynamic programming, the minimal cost at each state is obtained.

For each trajectory during the search phase, the distribution of the state along the trajectory can be computed using a LQG framework, as described in Section IV. It also has to consider the uncertainty from traffic participants. An algorithm computing collision probability between the autonomous vehicle and a traffic participant is presented in Section V-D.3, considering uncertainty from both traffic participants and the autonomous vehicle.

D. Cost Function

The cost function in this work includes smoothness costs c_s , static obstacle costs c_{so} , and dynamic obstacle costs c_{do} .

$$c(\mathbf{u}) = \sum_{t=0}^T (c_s(\mathbf{u}_t) + c_{so}(\mathbf{u}_t) + c_{do}(\mathbf{u}_t)) \quad (38)$$

1) *Smoothness costs*: The smoothness costs c_s are composed of the sum of acceleration, sum of curvature, distance to the center of the lane, and difference from the desired speed (defined by the road speed limit).

2) *Static obstacle costs*: A static map including all static obstacles is produced by the perception system of the autonomous vehicle [2]. Generally, a static map is represented by occupancy grids, which indicate whether a grid is occupied by static obstacles. We compute the Euclidean Distance Transform (EDT) of the static map, and then we can query the static obstacle costs from the EDT map.

3) *Dynamic obstacle costs*: The dynamic obstacles considered in this paper are only vehicles, whose shapes are generally rectangles.

From the standpoint of a traffic participant, the position of every vehicle is deterministic. A collision between two vehicles is detected using a polygon intersection algorithm.

From the standpoint of the autonomous vehicle, both its state and other vehicles' states are stochastic, which are formalized using normal distributions. For a point robot with normal distribution, the ellipse defined by the covariance matrix can provide an upper bound of the probability δ that the robot will collide with deterministic obstacles. In other words, if the ellipse does not collide with any deterministic obstacle, the collision probability is at most δ . However, in our case, the shape of the robot is a rectangle. We compute the Minkowski sum of the rectangle and the ellipse, which is a rounded rectangle. We then choose the minimum bounding rectangle as a conservative approximation of the rounded rectangle. In this case, the new rectangle still has the property that the collision probability is at most δ if it has no overlap with obstacles. Finally, we apply a polygon intersection algorithm to the minimum bounding rectangles to detect collisions.

VI. EXPERIMENTAL RESULTS

We first test the algorithm in a simulated environment with two lanes in the same direction. We compare the results of planning with and without the prediction of traffic participants. The autonomous vehicle is driving in the left lane, another car is driving in the right lane, and a third car is parked by the roadside protruding into the right lane (see Fig. 1a). The autonomous vehicle and the traffic participant's initial state distributions are illustrated as grey ellipses in Fig. 1a. The dashed rectangle denotes the minimum bounding rectangle. The upper bound of the probability δ is chosen as 0.25. In Fig. 1b, the traffic participant's motion is predicted using the method proposed in this paper.

The result of planning only using the LQG framework and without prediction of traffic participants is shown in Fig. 1c. The autonomous vehicle has no prediction for the traffic participant, and only assumes that the traffic participant will move at the current speed. Therefore, the autonomous vehicle will plan to stay in the center of the lane. However, the traffic participant is likely to shift to the left to avoid the static car. In this case, the trajectory planned by the autonomous vehicle will collide with the traffic participant.

The result of a deterministic planner (i.e., the planner proposed in this paper but without considering uncertainty) with prediction of traffic participants is shown in Fig. 1d. There is no collision between vehicle bodies (the blue

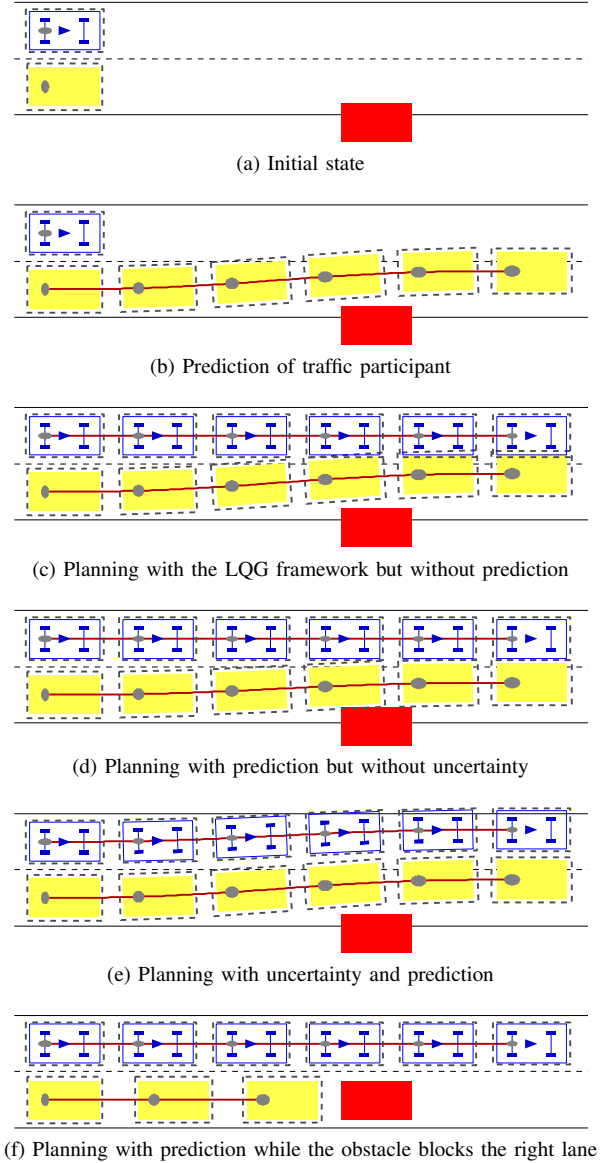


Fig. 1: Comparisons among: (e) the framework proposed in this paper, (c) planning only with the LQG framework, and (d) planning without uncertainty.

outlined rectangle and the yellow filled rectangle), but there are collisions between the minimum bounding rectangles. In other words, the trajectory planned by this deterministic planner is safe only if the perception and control of the autonomous vehicle are perfect. An alternative approach to make the deterministic planner safer is adding a constant buffer around the vehicle. However, compared to our method, which behaves like an adaptive buffer, a constant buffer may cause the autonomous vehicle's behavior to be either too aggressive or too conservative.

Fig. 1e shows the result of the full algorithm proposed in this paper. Our algorithm reasons about the most probable behavior the traffic participant will choose. Therefore, the autonomous vehicle plans a trajectory to avoid a potentially unsafe situation in the future. In Fig. 1f, the static car

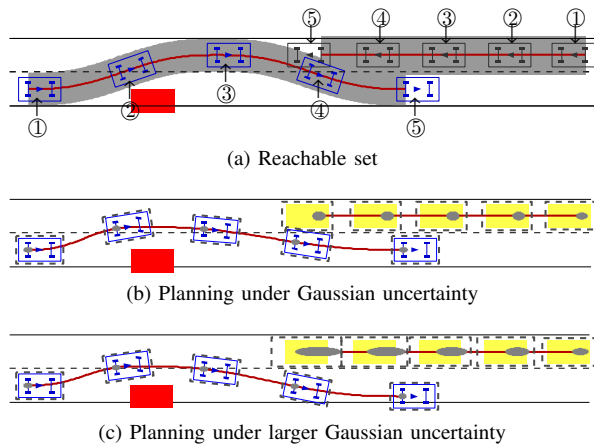


Fig. 2: Comparisons among: (a) reachable set based method, (b) the framework proposed in this paper, and (c) our method with larger uncertainty

is completely blocking the right lane. In this case, the autonomous vehicle forecasts that the traffic participant will stop behind the static car. Therefore, driving in the center of the lane will be a safe choice for the autonomous vehicle.

We also compared our method with the reachable set-based method [16]. As shown in Fig. 2, the autonomous vehicle (blue) is trying to pass a static car (red) on a two-lane roadway, while the oncoming traffic is approaching. In Fig. 2a, the best trajectory for the autonomous vehicle is first planned by a deterministic planner (i.e., the planner proposed in this paper but without considering uncertainty), and the reachable set is then computed along the trajectory. The trajectory is marked as unsafe since the reachable sets of the autonomous vehicle and the traffic participant have intersection at snapshot ④. In order to find a feasible trajectory, we have to replan a trajectory using the deterministic planner and evaluate it again, because the reachable set-based method provides no direct information to the planner. In addition, it will stop searching once a safe, but not necessary optimal, trajectory is found.

On the other hand, the planning framework proposed in this paper directly gives a safe trajectory, as shown in Fig. 2b, where grey ellipses indicate the magnitude of the uncertainty of state. However, the reachable set-based method has the advantage of providing a guarantee (i.e., 100% certainty) that a collision will not occur, while our Gaussian-based method gives a likelihood (e.g. 75%) bound that it will not. In Fig. 2c, we assume the autonomous vehicle receives no measurement on the traffic participant's x coordinate of its position. The autonomous vehicle turns back to the right lane earlier than its motion in Fig. 2b, since the traffic participant has larger uncertainty on its x coordinate.

VII. CONCLUSIONS

We have presented a framework of motion planning under uncertainty for an autonomous vehicle. The uncertainty considered in this paper comes from localization and control of an autonomous vehicle, and sensing and prediction of

traffic participants. The planner has shown the ability to deal with uncertainty from both the autonomous vehicle itself and traffic participants. It can produce safer trajectories when compared to our previous deterministic planner and planners that neglect the uncertainty caused by other traffic participants.

One possible improvement is to predict traffic participants' motion in a more stochastic way, e.g. Monte Carlo simulation [14]. In addition, a better prediction may be achieved based on intention, which is revealed in trajectory history and present situation. Another possible improvement is to estimate a more realistic distribution for GPS error than Gaussian distribution, which will help computing a better estimation of the localization uncertainty.

REFERENCES

- [1] J. Wei, J. M. Dolan, and B. Litkouhi, "A behavioral planning framework for autonomous driving," in *Intelligent Vehicles Symposium (IV)*, IEEE, 2014.
- [2] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi, "Towards a viable autonomous driving research platform," in *Intelligent Vehicles Symposium (IV)*, IEEE, 2013.
- [3] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, et al., "Towards fully autonomous driving: Systems and algorithms," in *Intelligent Vehicles Symposium (IV)*, 2011 IEEE, pp. 163–168, IEEE, 2011.
- [4] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, "A real-time motion planner with trajectory optimization for autonomous vehicles," in *Robotics and Automation (ICRA)*, 2012 IEEE International Conference on, pp. 2061–2067, IEEE, 2012.
- [5] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Robotics and Automation (ICRA)*, 2011 IEEE International Conference on, pp. 723–730, IEEE, 2011.
- [6] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al., "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.
- [7] M. McNaughton, C. Urmson, J. Dolan, and J. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *Robotics and Automation (ICRA)*, IEEE International Conference on, vol. 1, pp. 4889–4895, 2011.
- [8] T. Gu and J. M. Dolan, "On-road motion planning for autonomous vehicles," in *International Conference on Intelligent Robotics and Applications (ICIRA)*, pp. 588–597, 2012.
- [9] S. Thrun, W. Burgard, D. Fox, et al., *Probabilistic robotics*, vol. 1. MIT press Cambridge, 2005.
- [10] J. Van Den Berg, P. Abbeel, and K. Goldberg, "Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 895–913, 2011.
- [11] M. Althoff, O. Stursberg, and M. Buss, "Safety assessment of autonomous cars using verification techniques," in *American Control Conference, 2007. ACC'07*, pp. 4154–4159, IEEE, 2007.
- [12] M. Althoff, O. Stursberg, and M. Buss, "Model-based probabilistic collision detection in autonomous driving," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, no. 2, pp. 299–310, 2009.
- [13] J. Wei, J. M. Dolan, J. M. Snider, and B. Litkouhi, "A point-based mdp for robust single-lane autonomous driving behavior under uncertainties," in *Robotics and Automation (ICRA)*, 2011 IEEE International Conference on, pp. 2586–2592, IEEE, 2011.
- [14] A. Broadhurst, S. Baker, and T. Kanade, "Monte carlo road safety reasoning," in *Intelligent Vehicles Symposium (IV)*, pp. 319–324, IEEE, 2005.
- [15] M. Althoff and A. Mergel, "Comparison of markov chain abstraction and monte carlo simulation for the safety assessment of autonomous cars," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 12, no. 4, pp. 1237–1247, 2011.
- [16] M. Althoff and J. M. Dolan, "Set-based computation of vehicle behaviors for the online verification of autonomous vehicles," in *Intelligent Transportation Systems (ITSC)*, 2011 14th International IEEE Conference on, pp. 1162–1167, IEEE, 2011.