Belief State Planning for Autonomously Navigating Urban Intersections

Maxime Bouton[1], Akansel Cosgun[2], and Mykel J. Kochenderfer[1]

Abstract— Urban intersections represent a complex environment for autonomous vehicles with many sources of uncertainty. The vehicle must plan in a stochastic environment with potentially rapid changes in driver behavior. Providing an efficient strategy to navigate through urban intersections is a difficult task. This paper frames the problem of navigating unsignalized intersections as a partially observable Markov decision process (POMDP) and solves it using a Monte Carlo sampling method. Empirical results in simulation show that the resulting policy outperforms a threshold-based heuristic strategy on several relevant metrics that measure both safety and efficiency.

I. INTRODUCTION

Intersections account for 40% of driving accidents and represent a major challenge for automated driving [1]. Handling intersections involves planning under uncertainty with respect to driver behavior. One must be able to infer the goals of the other agents and anticipate rapid maneuver changes. Another difficulty is ensuring the satisfaction of multiple conflicting objectives including the risk of accident, the time to cross the intersection, the comfort of the passengers, and the disturbance caused to other drivers. The relative importance of the objectives varies by driver. The variety of users present in urban environments as well as complicated traffic rules also make intersections difficult to handle.

Several approaches have been employed to address intersection crossing. One approach involves hand-engineering hierarchical state machines that attempt providing explicit strategies for all possible situations. These state machines are useful for solving simple driving problems, but rely heavily on the designer to anticipate how to best handle different situations in advance. Hierarchical state machines were used in the DARPA urban challenge [2] and almost lead to an accident at an intersection [3].

Learning-based methods can help reduce the burden on the designer for developing robust decision strategies. One type of learning approach known as behavioral cloning involves learning a policy from a human driver [4], [5]. Some behavioral cloning approaches attempt to directly map sensor readings (e.g., raw pixels from a camera [4]) to driving commands. These approaches rely on a large amount of data and are unlikely to perform better than the human driver used for training.

Another category for developing intersection crossing strategies involves planning with respect to a mathematical
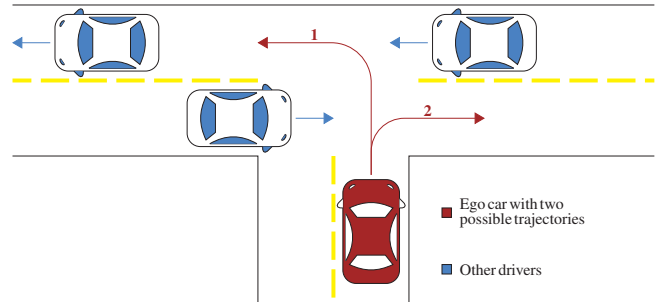
Fig. 1: The objective of the autonomous vehicle is to decide on the acceleration to apply along a given path. Two different scenarios are considered: right-turn and left-turn.

model of the problem. A partially observable Markov decision process (POMDP) is a standard model for sequential problems with stochastic state transitions and sensor uncertainty [6]–[8]. One of the challenges in this approach is in representing and modeling the problem in a way that allows the planning algorithm to be tractable [9]. Offline planners compute an approximately optimal strategy over the entire state space, prior to execution. Online planners, on the other hand, compute the best action to take at the current time step. A popular online algorithm is partially observable Monte Carlo Planning (POMCP), which relies on sampling from a generative model [10]. For the intersection problem in this paper, we augment POMCP with progressive widening [11] to accommodate the continuous state space.

The objective of this work is to develop an online decision making algorithm to cross an urban intersection autonomously. By modeling the problem as a POMDP, the autonomous system can dynamically change its decision to adapt to the behavior of other agents. As shown in Fig. 1, vehicles are at an unsignalized T-junction with traffic flowing in both directions. The autonomous vehicle is initially stopped at the intersection and tries to turn left or right. The nominal path is assumed to be generated by a high-level task planner and the proposed planner computes the acceleration profile along this path. Although we consider noisy position and velocity measurements, we do not consider sensor limitations such as occlusions. Finally, the behavior of the vehicles are represented by internal states that are not directly observable, but rather they are estimated using an interacting multiple model (IMM) filter [12].

II. PROPOSED APPROACH

A. POMDP Background

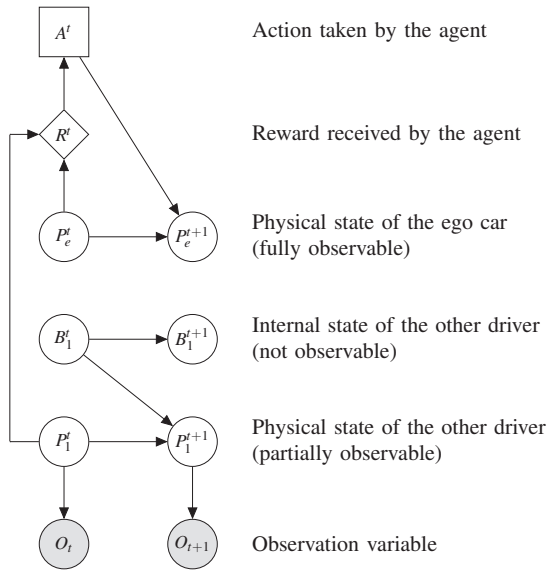A partially observable Markov decision process (POMDP) is a mathematical framework for sequential decision making

Fig. 2: Structure of the problem represented as a Bayesian network, the three components of the state variable are assumed independent.

Next to the figure nodes:
- Action taken by the agent
- Reward received by the agent
- Physical state of the ego car (fully observable)
- Internal state of the other driver (not observable)
- Physical state of the other driver (partially observable)
- Observation variable

under uncertainty. It is formally characterized by a tuple $(\mathscr{S}, \mathscr{A}, \mathscr{O}, T, O, R, \gamma)$, where $\mathscr{S}$ is the state space, $\mathscr{A}$ the action space, $\mathscr{O}$ is the observation space, T is a transition model, O is an observation model, R a reward model, and γ is a discount factor [13].

Uncertainty is represented by the transition model and the observation model. An important source of uncertainty in the context of autonomous driving is the behavior of the other drivers [8], [9]. From a state $s \in \mathscr{S}$ of the environment, the agent takes an action $a \in \mathscr{A}$ to maximize the expected accumulation of reward $r(s, a)$ over time. The state s will then transition to s' with probability $T(s', s, a) = \Pr(s' \mid s, a)$ under the Markov assumption that the state s' only depends on the previous state.

The agent has uncertain knowledge about the state of the environment and maintains a belief state $b \in \mathscr{B}$. The belief state is a probability distribution over all possible states, $b : \mathscr{S} \mapsto [0, 1]$, and $b(s)$ represents the probability of being in state s. Belief state planning involves finding a policy that maps belief states to actions in a way that maximizes the expected discounted accumulation of reward over time.

B. Intersection Navigation Problem

The structure of the problem can be represented by the Bayesian network in Fig. 2. The round nodes represent state variables that change over time. The diamond-shaped node corresponds to the reward received and the square node represents the action taken at a given time step.

1) State Space: The state in the POMDP includes both the physical state of the environment and the behavior of other drivers. The intersection geometry is assumed to be known to the planner. Three sets of variables are used to define the state space:

- $P_e^t = (x_e^t, y_e^t, \theta_e^t, v_e^t, a_e^t)$ describes the physical state of the ego car at time t.

- $P_i^t = (x_i^t, y_i^t, \theta_i^t, v_i^t, a_i^t)$, $i = 1, \ldots, n$ describes the physical state of the n other cars in the intersection at time t.
- B_i^t describes the internal state of the other drivers, characterizing their behavior at time t. In the experiments used in this paper, the internal state may correspond to one of two different models.

Here, x and y represent the position in a Cartesian frame, θ represents the vehicle orientation, v is the speed, and a is the magnitude of the acceleration. Variables with subscript e are associated with the ego car, and variables with subscript i correspond to the ith vehicle. The relationships between the variables are illustrated by Fig. 2. To simplify the structure of the problem we assume independence between the ego car physical state and the other vehicles' physical states.

2) Action space: The objective of the planner is to compute the acceleration profile along the desired path (left-turn or right-turn as illustrated in Fig. 1). Strategic maneuvers such as hard braking, moderate braking, maintaining constant speed and accelerating can be represented by a finite set of acceleration and deceleration action: $\{-4\,\mathrm{m\,s}^{-2}, -2\,\mathrm{m\,s}^{-2}, 0\,\mathrm{m\,s}^{-2}, 2\,\mathrm{m\,s}^{-2}\}$.

3) Process Model: The input to the problem is the ego path; either a left or right turn. The motion of the ego car along this path is only controlled by an acceleration input. Given the shape of the trajectory, it is more convenient to use polar coordinates. The kinematic equations used to update the ego car state are as follows:

$$\begin{cases} x_e^{t+1} = x_e^t + v_e^t \sin(\theta_e^t)\delta t + a_e^t \sin(\theta_e^t)\frac{\delta t^2}{2} \\ y_e^{t+1} = y_e^t + v_e^t \cos(\theta_e^t)\delta t + a_e^t \cos(\theta_e^t)\frac{\delta t^2}{2} \\ v^{t+1} = v_e^t + a_e^t \delta t \end{cases} \quad (1)$$

The orientation θ is updated according to the desired trajectory. In the equations, δt is the time step between decisions.

Two different kinematic models were used to model the behavior of the other drivers: constant velocity (CV) and constant acceleration (CA) [12]. These models can describe various behavior including braking (CA model with negative acceleration) or maintaining speed (CV model). The state transition function follows linear Gaussian dynamics:

$$\Pr(P_i^{t+1} \mid P_i^t) = \mathscr{N}(P_i^{t+1} \mid \mathbf{T}P_i^t, \mathbf{Q}) \quad (2)$$

where $\mathbf{T}$ is the state transition matrix and $\mathbf{Q}$ the process noise. These matrices are different for each kinematic model and follow the equations of Bar-Shalom, Li, and Kirubarajan [12]. Process noise matrices are characterized by a spectral density $\sigma_{\{CV, CA\}}$. The Gaussian dynamics provide a suitable representation of the world as it is describing continuous variables with a minimal amount of information (mean and covariance) so that the problem remains computationally feasible.

The internal state B_i^t can have one of two values corresponding to the two possible kinematic models. Given the value of B_i^t, the variable P_i^t is updated using Eq. (1). At each time step, we assume that the behavior can change according to a switching probability matrix p where p_{ij} is the probability from switching to model j from model i. No prior knowledge of the path of the other drivers is assumed.

4) Observation Model: The observation space describes what we can measure in the environment. The vehicle is assumed to have a perfect knowledge of its physical state. We assumed that position and velocity of other drivers are partially observable while their acceleration and their behavior are internal states that cannot be measured. Finally, the orientation of the other cars is assumed to be known. The observation space is defined as follows:

$$O_i^t = (z_{x_i}^t, z_{y_i}^t, \theta_i^t, z_{v_i}^t), \quad i = 1, \ldots, n \tag{3}$$

where the components are the measured position, the orientation, and the velocity of the ith vehicle at time t.

To have a simple representation of the observation distribution, we model the sensor measurements using a Gaussian distribution:

$$\Pr(O_i^t \mid P_i^t) = \mathcal{N}(O_i^t \mid \mathbf{H}P_i^t, \mathbf{R}) \tag{4}$$

where $\mathbf{H}$ is the observation model matrix and $\mathbf{R}$ is the observation noise matrix. We assume that the measurements are independent ($\mathbf{R}$ diagonal) and characterized by the standard deviations σ_p for the position measurement and σ_v for the velocity measurement.

5) State Estimation: Since the state is factored into three independent variables, the belief is defined in a similar way:

- The ego car physical state, assumed perfectly known.
- A distribution over the vehicle physical state. From the measurements, we can maintain an estimate of the other vehicles' state. We assume that the physical states follow a Gaussian distribution $\mathcal{N}(\hat{P}_i^t, \hat{\Sigma}_i^t)$, where $\hat{P}_i^t$ and $\hat{\Sigma}_i^t$ are the estimated mean and covariance of the physical state of vehicle i at time t.
- A distribution over the two possible kinematic models $\{\mu_{1i}^t, \mu_{2i}^t\}$ representing the probability of car i following the CV model and the probability of car i following the CA model, respectively.

To infer which of the following models the cars are following, we used an Interacting Multiple Model (IMM). IMMs have been used in tracking applications and pedestrian intention prediction [14], [15], as well as for lane changing detection in autonomous driving [16]. The IMM mixes two Kalman filters for both kinematic models CV and CA and update both the state estimation and the model probability distribution at each time step given an observation.

The IMM algorithm consists of three steps: mixing, filtering and combining. The first step computes an estimate of the state with respect to the two transition models and from these estimates computes two mixed inputs (a linear combination of both). The two mixed inputs are then filtered using a classic Kalman filter [12].

In the POMDP context, we used the IMM as the belief updater. It takes as input a belief state and an observation and returns the updated belief state. It acts only on the partially observable part of the belief space, i.e. not on the ego car state.

One of the subtleties of the problem is that we cannot directly measure the intentions of other drivers. Since we assume that the drivers are following one among two possible Gaussian dynamics, the IMM is particularly well suited to estimate the states.

6) Reward Model: The agent is rewarded for reaching a final position in the intersection and receives a small penalty for each action and a large penalty for collision.

III. ONLINE BELIEF STATE PLANNING

Methods for computing optimal policies for a POMDP can be divided into two categories: offline and online [13]. Offline methods compute the policy over the entire state space prior to execution in the environment. Hence, they typically do not scale to high dimensional problems. In our problem formulation, we have five continuous variables for each vehicle in the intersection. Computing the policy over the entire belief state space is intractable. Moreover, it is likely that many states will never be encountered when interacting in the environment.

Online methods plan from the current belief state up to a certain horizon. As a consequence, online planning algorithms consider only the states reachable from the current belief and at each time step the solver computes the (approximately) optimal action. The best action is typically recomputed after each interaction with the environment.

A. POMCP

Since the state space is continuous, we use a sampling-based method known as Partially Observable Monte Carlo Planning (POMCP). POMCP is an extension of the Upper Confidence Tree (UCT) algorithm with partially observable state variables. The algorithm takes as input a belief state. From this belief state, it will build a tree where each node represents a history h, which is a sequence of actions and observations. Each node is sampled using the model described in Section II-B. The construction of the tree involves iterating through the following three steps many times:

- Expansion: If the node is not in the tree, we explore the outcome of all the possible actions and initialize $N(h,a)$ and $Q(h,a)$, which are the number of times we visited the node h taking action a and the associated value function.
- Rollout: We simulate up to a desired depth according to a rollout policy.
- Search: If the sampled state is already in the tree, we choose the action that maximizes $Q(h,a) + c\sqrt{\frac{N(h)}{N(h,a)}}$, where $N(h)$ is the number of times the history was visited and $N(h,a)$ is the number of times the action and history was visited. The parameter c controls the balance between exploration and exploitation.

After each iteration, the information is then propagated up to the root node. The POMCP algorithm converges to the optimal policy as the number of tree queries increases.

B. Planning in Continuous State Space

One of the drawbacks of POMCP is that it cannot handle continuous state spaces. When sampling a continuous variable from the initial belief, the probability of visiting the

same state twice is infinitesimally small, resulting in a very wide tree with a depth of one. One way to address this issue is to use progressive widening [11].

Progressive Widening (PW) involves defining when to explore new states in the tree. It is controlled by two parameters α and k. The selection criteria is as follows:

- Compute $k' = kN(h,a)^{\alpha}$.
- If k' is greater than the number of children of the node (h,a), then we sample a new state. Otherwise, we choose a state that has already been visited.

The branching factor of the search tree can be affected by tuning k and α. When the noise in the generative model is large, one typically wants a large branching factor (which can be achieved by increasing α, for example).

IV. EXPERIMENTAL SETUP

A. Simulation and Parameters

We used the SUMO simulator [17] for our experiments, which relies on an Intelligent Driver Model (IDM) [18]. As a consequence, our generative model (relying on Gaussian dynamics) is different than the model used in the test environment. The motivation for this mismatch is to assess how the POMCP algorithm can handle model discrepancies that would exist in real-world applications.

The SUMO simulator takes into account the interaction between drivers, and it outputs the position, velocity, and orientation of the vehicles. To simulate the perception of the autonomous car, we added white noise to the position and velocity measurements. The simulation parameters are given in Table I. The traffic density is expressed as the probability of a vehicle going through the intersection every second. The noise parameters correspond to the standard deviations σ_p and σ_v in Section II-B.4.

TABLE I: Parameters of the simulation environment

Parameter	Value
Traffic density	0.2
Position sensor noise	$0.1\,\mathrm{m}$
Velocity sensor noise	$0.1\,\mathrm{m\,s^{-1}}$
Maximum Speed	$13.88\,\mathrm{m\,s^{-1}}$

The sequential decision making process proceeds as follows. We start with a prior belief b_t. From this belief, we compute an optimal action using the POMCP algorithm and then run a simulation step in SUMO where the environment evolves (including the ego car with respect to the action taken). After this step, the agent receives a reward and observes the environment, and updates its belief to b_{t+1} using the IMM algorithm. The decision and measurement are made every 0.25 s. In order to compute the action at each step, we used the POMCP algorithm with progressive widening with the parameters in Table II.

B. Performance Metrics

To evaluate performance, we used the following metrics:

- Average number of collisions
- Average time to cross the intersection

TABLE II: POMCP solver parameters used in the experiment

Parameter	Value
Depth	15
Exploration constant	20.0
Tree queries	2000
Rollout policy	TTC Policy
PW α	0.2
PW k	4.0

- Success rate at which the car crosses the intersection without crashing or a timeout.
- Average time when the traffic is braking
- Average time when a car is stopped.

The first three metrics account for safety and efficiency. The braking time and the waiting time captures the impact of the ego car on the current traffic.

C. Baseline Policy

We defined a simple heuristic policy to serve as a baseline that uses a time to collision (TTC) threshold to decide when to cross. The TTC is defined as follows. Consider an imaginary line starting from the ego car aligned with the y axis. The TTC with another vehicle i in the intersection will be the time it takes for the vehicle to reach that line. For vehicle i, it is estimated by dividing d_i by V_i, where d_i is the distance indicated in Fig. 3 and V_i is the speed of vehicle i relative to the ego car. If the TTC exceeds a threshold for two consecutive time steps of 0.1 s, the vehicle starts the crossing phase. The crossing phase follows the IDM.
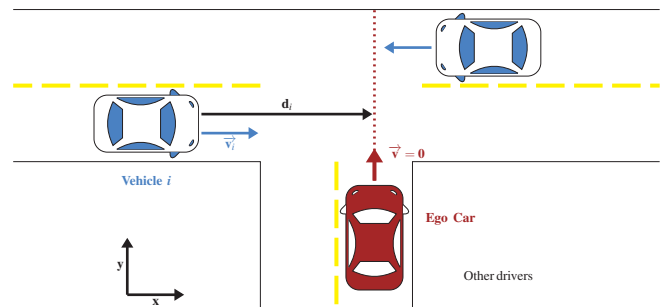

Fig. 3: Representation of the first phase of the TTC policy, the ego car measures V_i and d_i to compute the TTC and then decides to cross or not according to the threshold.

V. RESULTS

We analyzed the influence of the TTC threshold on different metrics and chose the threshold that results in zero collisions over a thousand simulations. Figure 4 shows how four metrics vary with respect to the choice of the TTC threshold. Due to measurement noise, we can see that even for high thresholds there are still small fluctuations in the collision rate. The chosen threshold for comparison with the POMCP policy is 4.5 s in order to guarantee success for both the right and the left turns under the experimental traffic condition in Table I, without being overly conservative.

The reward function in the POMDP formulation can be used to tune the behavior of the agent in favor one objective
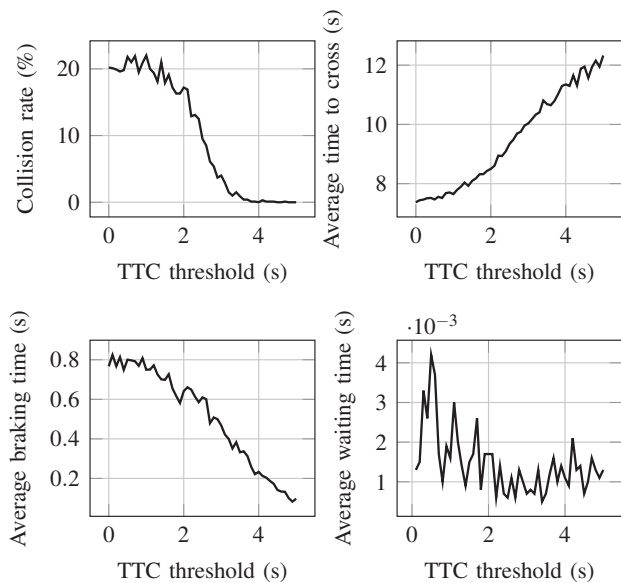
Fig. 4: Four metrics with different TTC thresholds for right-turns with a traffic density of 0.2.

Fig. 6: Metrics for varying traffic density for both the TTC and the POMCP policies in the right-turn scenario.

over another. By varying the reward parameters, such as the cost for each action, we can reach a point where the expected reward will decrease very fast with time. As a consequence, increasing this cost will favor a minimization of the time to cross at the expense of collision risk. By tuning the reward function, we can balance these two conflicting objectives. Figure 5 illustrates the trade-off between these two objectives for the right-turn scenario. The POMCP policy clearly dominates the threshold policy with respect to the collision rate and the time to cross.
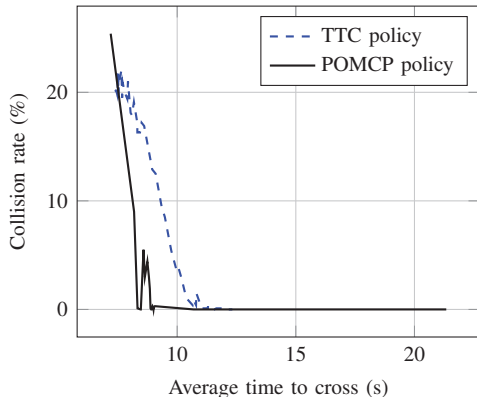

Fig. 5: Trade-off between collision rate and time to cross as key parameters are varied for the policies (action cost for POMCP policy and threshold for TTC policy) in the right-turn scenario with a traffic density of 0.2.

We selected a conservative set of POMCP policy parameters (Table III) to compare against the TTC policy with a threshold of 4.5 s. The penalties chosen for each action differ in order to favor forward motion. The numerical values were chosen to separate each outcome by several orders of magnitude. We also compared against a random policy for both scenarios in Table IV and Table V for fixed traffic
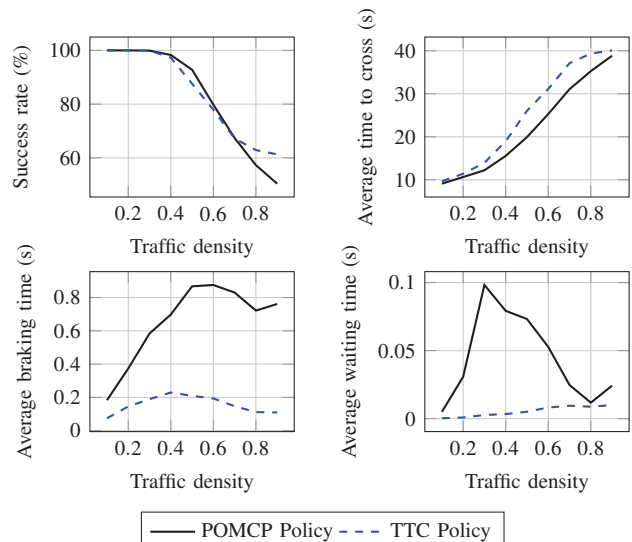
conditions (Table I). The metrics are averaged over one thousand simulations.

For the right-turn scenario, both the POMCP and TTC policies achieve 100% success rate (Table IV). However, the POMCP policy outperforms the TTC policy in average time to cross the intersection. The waiting time is higher for the POMCP policy, but it still does not exceed 10 ms on average.

Table V shows that for the left-turn scenario the TTC policy achieved zero collisions and 100% success rate, whereas the POMCP policy still has some collisions (0.2 %) and time-outs, leading to a success rate of (99.0 %). The braking time and waiting time are also higher.

In order to assess the scalability of the two policies, we analyzed the evolution of the metrics with an increasing traffic density for the right-turn scenario. Both policies achieved a collision rate of 0%, but they were subject to time-outs, which are reflected by a decrease in the success rate as the traffic density increases. Figure 6 shows that until a density of 0.7, the POMCP policy has a success rate at least as high as the TTC policy. For every tested traffic density, the POMCP policy takes on average less time to cross the intersection with a maximum difference of 6.12 s for a traffic density of 0.5. However, the braking time and the waiting time are higher than the TTC policy.

TABLE III: Reward function parameters

Parameter	Value
Collision penalty	−2000.0
Acceleration penalty	−4.98
Maintaining speed penalty	−4.99
Moderate braking penalty	−5.0
Strong braking penalty	−5.02
Crossing reward	100.0

The results show that both the POMCP policy and the TTC policy are safe for the right-turn scenario, even under some measurement noise. Moreover, the POMCP policy reaches

TABLE IV: Performance of the policies for the right-turn scenario

Policy	Time to cross (s)	Braking time (s)	Waiting time (s)	Collision rate (%)	Success rate (%)
POMCP	10.6465	0.3733	0.0308	0.0	100.0
TTC	10.7270	0.1170	0.0014	0.0	100.0
Random	55.5948	37.9805	20.4013	9.80	0.40

TABLE V: Performance of the policies for the left-turn scenario

Policy	Time to cross (s)	Braking time (s)	Waiting time (s)	Collision rate (%)	Success rate (%)
POMCP	10.3735	0.3763	0.1293	0.2	99.0
TTC	10.7704	0.1912	0.0010	0.0	100.0
Random	34.1028	17.0503	10.2940	75.70	0.10

the goal faster than the TTC policy but will cause the other users to brake and wait more often. Figure 6 shows that the POMCP policy manages to cross the intersection more often and faster than the TTC policy up to a certain traffic density at the expense of somewhat greater disruption to the traffic. The choice of the reward function penalizing the actions makes the ego car more eager to enter the intersection and is not penalizing for making the other drivers brake or wait. The problem formulation can explain the higher braking time and waiting time for the POMCP policy.

For left turns, similar conclusions can be drawn on the time to cross, the braking time, and the waiting time. However, the POMCP policy still has some collisions. One explanation is the difference between the generative model and the simulator model. In order to assess the discrepancies between the two, we measured the error in the position prediction as a function of the planning horizon. We found an average error of $2.15\,\mathrm{m}$ when predicting the motion of the other cars ten steps ahead (2.5 s). This difference in the predicted position prevents the algorithm from predicting some rapid maneuver changes. A more sophisticated generative model, combined with a good internal state estimator could improve performance, and we believe zero collision rate could be achieved for left turns.

VI. Conclusions

We demonstrated an online belief state planning approach based on a POMDP formulation to address a decision problem at an unsignalized intersection. The proposed approach performs better than a heuristic policy, even with a fairly simple transition model. It is robust enough to handle discrepancies between the assumed model and the simulator model. The resulting policy outperforms the baseline in most of the considered metrics. We showed that we can balance the different objectives of the problem by tuning the reward function of the POMDP. Further work will involve improving the state estimator by using a more accurate generative model than the linear Gaussian model used in this paper. We would also like to complement our current approach with an offline planner and increase the complexity of the scenarios and move toward more realistic models. An immediate consideration would involve pedestrians, as well as sensor occlusions.

References

[1] National Highway Traffic Safety Administration, "Traffic Safety Facts," Tech. Rep. DOT HS 0811 170, 2008, pp. 1–6.

[2] S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schöder, M. Thuy, M. Goebl, F. von Hundelshausen, *et al.*, "Team AnnieWays autonomous system for the DARPA urban challenge 2007," in *The DARPA Urban Challenge*, Springer, 2009, pp. 359–391.

[3] L. Fletcher, S. Teller, E. Olson, D. Moore, Y. Kuwata, J. How, J. Leonard, I. Miller, M. Campbell, D. Huttenlocher, *et al.*, "The MIT–Cornell collision and why it happened," *Journal of Field Robotics*, vol. 25, no. 10, pp. 775–807, 2008.

[4] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *ArXiv preprints*, vol. abs/1604.07316, 2016.

[5] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[6] H. Bai, D. Hsu, and W. S. Lee, "Integrated perception and planning in the continuous space: A POMDP approach," *Internatinal Journal of Robotics Research*, vol. 33, no. 9, pp. 1288–1302, 2014.

[7] D.-K. Kye, S.-W. Kim, and S.-W. Seo, "Decision making for automated driving at unsignalized intersection," in *International Conference on Control, Automation and Systems (ICCAS)*, 2015.

[8] W. Song, G. Xiong, and H. Chen, "Intention-aware autonomous driving decision-making in an uncontrolled intersection," *Mathematical Problems in Engineering*, vol. 2016, 2016.

[9] T. Bandyopadhyay, K. S. Won, E. Frazzoli, D. Hsu, W. S. Lee, and D. Rus, "Intention-aware motion planning," in *Algorithmic Foundations of Robotics X*, 2012.

[10] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Advances in Neural Information Processing Systems (NIPS)*, 2010.

[11] A. Couëtoux and H. Doghmen, "Adding double progressive widening to upper confidence trees to cope with uncertainty in planning problems," in *European Workshops on Reinforcement Learning (EWRL)*, 2011.

[12] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: Theory algorithms and software*. John Wiley & Sons, 2004.

[13] M. J. Kochenderfer, *Decision making under uncertainty: Theory and application*. MIT press, 2015.

[14] A. T. Schulz and R. Stiefelhagen, "A controlled interactive multiple model filter for combined pedestrian intention recognition and path prediction," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2015.

[15] A. F. Genovese, "The interacting multiple model algorithm for accurate state estimation of maneuvering targets," *Johns Hopkins APL Technical Digest*, vol. 22, no. 4, pp. 614–623, 2001.

[16] A. Goswami and C. S. G. Lee, "Design of an interactive multiple model based two-stage multi-vehicle tracking algorithm for autonomous navigation," in *IEEE Intelligent Vehicles Symposium (IV)*, 2015.

[17] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO–simulation of urban mobility," *International Journal on Advances in Systems and Measurements (IARIA)*, vol. 5, no. 3–4, 2012.

[18] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, no. 2, p. 1805, 2000.