Exploiting Continuity of Rewards: Efficient Sampling in POMDPs with Lipschitz Bandits

Ömer Şahin Taş, Felix Hauser, and Martin Lauer

Abstract— **Decision making under uncertainty can be framed as a partially observable Markov decision process (POMDP). Finding exact solutions of POMDPs is generally computationally intractable, but the solution can be approximated by sampling-based approaches. These sampling-based POMDP solvers rely on multi-armed bandit (MAB) heuristics, which assume the outcomes of different actions to be uncorrelated. In some applications, like motion planning in continuous spaces, similar actions yield similar outcomes. In this paper, we utilize variants of MAB heuristics that make Lipschitz continuity assumptions on the outcomes of actions to improve the efficiency of sampling-based planning approaches. We demonstrate the effectiveness of this approach in the context of motion planning for automated driving.**

Index Terms— POMDP, multi-armed bandits, Monte Carlo planning, POSLB, POSLB-V, motion planning.

I. Introduction

Sequential decision making problems in which the system dynamics are uncertain and the system state is unobservable can be framed as a POMDP. The POMDP framework encodes uncertain and incomplete knowledge not by single states, but by beliefs over all possible states. By optimizing over a sequence of actions and observations, it considers a very large number of possible future outcomes. This comes at the cost of high computational complexity.

A POMDP can typically be solved either by performing *value iteration*, or by *Monte Carlo tree search*. The latter are real-time capable and are more flexible since they do not require state discretization. Common approaches are the algorithms POMCP [1] and TAPIR [2]. They employ the Upper Confidence Bound (UCB) [3] bandit algorithm to explore promising actions and exploit good ones.

UCB does not make any assumption on the outcomes of similar actions. This is essential for certain decision making problems, e.g. playing board games. However, many real-world applications operate on compact, continuous spaces in which the profile of outcomes are continuously differentiable. Furthermore, the uncertainties present in the environment have a *smoothing effect* on any discontinuity, if they can be represented with a probability distribution whose density is continuous. Such applications can benefit from Lipschitz continuity assumptions on the outcomes of actions.

A POMDP solver can exploit the dependence between the expected rewards of different actions by utilizing a MAB that assumes Lipschitz continuity. Even though this would substantially improve the efficiency of the sampling, currently there is no POMDP solver that exploits this property. In this paper, we propose to utilize a Lipschitz MAB that can work within the POMDP framework. We further investigate whether the motion planning problem in automated driving has a continuous reward profile by analyzing the dependencies between actions in different scenarios, including edge cases like collisions.

In order to highlight the contribution of this work, we first provide background information on POMDPs in Section II. Next, in Section III we focus on existing works that deal with motion planning for automated vehicles by utilizing the POMDP framework and subsequently introduce our model, which shows several differences to existing works. Once the underlying settings are introduced, we provide an overview on MABs we benchmark in Section IV. We use a modified version of the POMCP algorithm to efficiently solve the POMDP problem, presenting those modifications in Section V, before analyzing the structure of reward profiles and benchmarking the MABs in the Evaluation section.

II. Background

A POMDP operates on spaces of states $\mathbb{S}$, observations $\mathbb{O}$ and actions $\mathbb{A}$. The state s cannot be observed exactly, and therefore, it is represented as a probability distribution over the state space, i.e. belief $b(s)$. Transitions from one state to another and their respective observations are given by a transition and an observation model. Every time the agent chooses an action $a \in \mathbb{A}$ and the environment transitions to the next state, the agent receives the scalar reward $r \in \mathbb{R}$, computed by the reward function $\mathcal{R}(s, a)$. The action-value $Q^\pi(b, a)$ represents the expected future reward when taking action a in belief b and following policy π afterwards.

The POMCP algorithm [1] applies Upper Confidence Bound for Trees (UCT) [4] to POMDPs by iteratively sampling sequences of states and observations. These sequences are kept track of in a tree data structure of nodes and edges, corresponding to states, actions and observations. Key to the POMCP algorithm is the equivalence of belief b and history h. A history is the sequence of actions and observations that have been selected and observed by the agent, $h_t = \{a_1, o_1, \dots, a_t, o_t\}$. When the initial belief b_0 is fixed and known, the belief is represented by the history.

One episode of the sampling procedure samples a single particle and involves four phases: In the `simulation` phase of POMCP, actions are selected by the MAB algorithm. Based on the generative observation model, the next state is determined. When simulation reaches a node that is not

Corresponding author: `tas@fzi.de`. The authors are with FZI Research Center for Information Technology and Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany.

in the tree yet, the tree is `expanded` by the node and the simulation stops. The initial value estimate of the new node is done by `rollout` policy, which is used as an heuristic in lieu of expanding the tree further. In the final phase, `backup`, the encountered nodes are updated from bottom-up in light of the newly gathered information.

III. Motion Planning in Automated Driving with the POMDP Framework

In automated driving, motion planning has to consider uncertain information such as the unknown intentions and future directions of other traffic participants, or objects in occluded areas. One way to consider these uncertainties is to use the POMDP framework.

A. Related Work

POMDPs have been used in previous works to solve the motion planning problem for automated driving. Bai et al. [5] use the POMCP algorithm for planning in an intersection scenario. They model the uncertain intentions of other drivers and driving style. Although the POMCP solver is capable of handling continuous state spaces, the authors used discrete states. Brechtel introduces a POMDP solver that is able to work with continuous spaces by learning a problem-specific representation of the state space [6], [7].

The importance of considering uncertainties in the planning of motion for automated driving is shown by Sunberg et al. [8]. They compare different planning frameworks based on MDPs and POMDPs, which are evaluated on a lane change scenario. The results clearly indicate the need to take uncertainties into consideration. For the POMDP variant the POMCP algorithm is enhanced with a technique called double progressive widening [9].

Sefati et al. [10] also include uncertainties in the motion planning for an intersection scenario. They consider the unknown intentions of other traffic participants by inferring the state with a Bayesian network model. To solve the POMDP, the MCVI solver [11] is combined with ideas form the SARSOP algorithm [12]. Their approach uses several heuristics to guide the exploration through the action space, though sparse, containing only three actions.

Bouton et al. [13] study intersection and pedestrian crosswalk scenarios. They do not deal with intentions of other road users, but focus on the integration of occlusions. The action space includes only four or five actions, depending on the scenario. The POMDP problem with multiple participants is split into many problems involving only one other agent.

Hubmann et al. [14] use a real-time method for computing solutions to intersection scenarios with multiple vehicles with unclear intentions. They use the TAPIR algorithm and consider an action space of only four actions. In a later work they extend their approach to tackle occlusions [15].

B. Modeling the Motion Planning Problem for the POMDP Framework

The POMDP framework requires model-based representations on which to operate. We closely follow the model presented in [14] with some modifications to work better with a denser action space.

1) Map data: We refer to a path and its accompanying data as a *route* and denote it by ρ. Accompanying data consists of the curvature κ and the reference velocity v. In this way, we store every route ρ as a tuple of n points

$$\rho = (p_i)_{i=1,\dots,n} \quad \text{with} \quad p_i = (x_i, y_i, l_i, \kappa_i, v_i)^\top \in \mathbb{R}^5,$$

where x and y are Cartesian coordinates and l is the distance along the path.

2) States, Observations, Actions: We apply path-velocity decomposition [16] to reduce the complexity and, therefore, we plan acceleration along a path. The state s of a vehicle k is given by $s_k = (l_k, v_k, \rho_k)$. For the ego vehicle the route ρ is known, hence, we represent its state as $s_0 = (l_0, v_0)$. For k other vehicles in a traffic scene, we have the combined state $s = (s_0, s_1, s_2, \dots, s_k)$.

Observations o are defined in Cartesian coordinates and only contain information about k other traffic participants $o = (o_1, o_2, \dots, o_k)$ with $o_k = (x_k, y_k, v_k)^\top \in \mathbb{R}^3$.

Actions available to the planner represent different values of acceleration a. Throughout the rest of this work the available acceleration values are within the comfortable range $a \in [-3\,\mathrm{m\,s^{-2}}, 1\,\mathrm{m\,s^{-2}}]$ with equidistant spacing.

3) Transition model: The POMDP model is discretized in time and the transitions have the timestep $T_s = 1.0$s. The vehicles follow a constant acceleration model and the route does not change. For the ego vehicle the route is predefined and the acceleration input a_0 is simply the action chosen by the POMDP solver.

For predicting accelerations of other vehicles a_k, we use the Intelligent Driver Model (IDM) [17]. This model consists of a free driving term $a_{\text{ref},k}$ that allows the vehicle to smoothly adjust the velocity in case of deviations from its reference, and an interaction term $a_{\text{int},k}$ for avoiding collisions. We saturate the resulting acceleration value with the maximum deceleration a^- and add Gaussian noise $a_{\text{noise},k} \sim \mathcal{N}(0, \sigma_a^2)$ to cover for modeling errors and uncertainties in the behavior model.

$$a_k = \max(a_{\text{ref},k} + a_{\text{int},k},\ a^-) + a_{\text{noise},k}.$$

IDM may yield accelerations resulting in negative velocity due to time discretization. In such cases, we calculate the stop position from the motion kinematics.

4) Observation model: The observation model generates a possible observation from a given state-particle. After the transformation from path coordinates to Cartesian coordinates, we add observation noise sampled from independent Gaussian distributions.

5) Reward model: The terms in the reward model resemble those of an ordinary trajectory optimization problem. All terms are modeled as negative rewards

$$r = r_{\text{coll}} + r_v + r_{j,\text{lon}} + r_{a,\text{lat}}.$$

The collision term r_{coll} takes only collisions of the ego vehicle into account

$$r_{\text{coll}} = \begin{cases} 0 & \text{no collision} \\ \zeta_{\text{coll}} & \text{ego vehicle collides} \end{cases} .$$

The criterion from [18, p. 223] is applied for the collision check, which is performed during state transition. The corresponding reward therefore is a function of the current state and the previous one.

We punish the deviation of the ego vehicle velocity from a predefined reference with an asymmetric loss function

$$r_v = \begin{cases} \zeta_v \ (v_0 - v_{\text{ref}})^2 & \text{if } v_0 \geq v_{\text{ref}} \\ \zeta_v \ \log\Big(1 + (v_0 - v_{\text{ref}})^2\Big) & \text{otherwise} \end{cases} ,$$

with ζ_v being the cost scaling factor. We apply quadratic cost if v_{ref} is exceeded and *Cauchy loss* [19] otherwise. The asymmetric loss function is motivated by the assumption, that driving slow or standing still might be part of an ordinary solution to the motion planning problem, whereas driving with higher speed is only acceptable in extraordinary cases.

The third term in the reward model $r_{j,\text{lon}} = \zeta_{j,\text{lon}} {j_0}^2$ accounts for jerk and considers changes in the longitudinal acceleration. The last term is the lateral acceleration term $r_{a,\text{lat}} = \zeta_{a,\text{lat}} \left(\kappa \ v_0^2\right)^2$.

IV. Multi-armed bandits

The general MAB is a sequential decision problem [20], which proceeds in rounds denoted by $t \in \{1, \dots, T\}$. At every round, the agent picks one arm a from some set of arms $\mathcal{A} = \{a_1, a_2, \dots, a_K\}$, where K denotes the number of arms, with the goal of maximizing the rewards it collects over T rounds. In the stochastic setting, every arm corresponds to a reward distribution, whose properties are unknown to the agent. The distributions are assumed to be stationary, i.e. they do not change over time. The MAB algorithm balances between the exploration of unknown arms and exploitation of good arms.

Several approaches have been suggested in the past to deal with the MAB problem.

A. UCB

Introduced by Auer et al. [3], UCB is *optimistic in the face of uncertainty*. Its robustness and practicality have been proven in numerous applications. In each round t the algorithm calculates the index

$$b_t(a) = \hat{\mu}_t(a) + c\sqrt{\frac{2\log t}{n_t(a)}} \tag{1}$$

for every arm and then chooses the arm with the highest index (cf. Algorithm 1). The first term is the current average reward of an arm $\hat{\mu}_t(a)$. The second term is called the confidence radius and is proportional to the upper bound of the confidence interval of the average reward. The denominator $n_t(a)$ is the number of times arm a has been played and $c \in \mathbb{R}$ is the *exploration constant* trading-off exploitation and exploration.

Algorithm 1: Upper Confidence Bound (UCB)

```
if $t \leq K$ then
    Choose arm from $\{a : n_t(a) = 0\}$ at random
else
    Choose arm $a_t = \arg\max_{a \in \mathcal{A}} b_t(a)$
```

A more general case of UCB is the KL-UCB bandit algorithm. It allows distributions of canonical exponential family to be set for the reward. If rewards of the arms are assumed to be Gaussian distributed with equal variance, the KL-UCB index becomes equal to Equation 1, but with an exploration constant of $c = \sqrt{\sigma^2}$.

B. UCB-V

Audibert et al. enhanced UCB by including the estimated reward variances $\hat{\sigma}_t^2(a)$ [21]. In contrast to UCB, the variances of the rewards are not assumed to be equal. The index is given as

$$b_t(a) = \hat{\mu}_t(a) + \sqrt{\frac{2\hat{\sigma}_t^2(a)\log t}{n_t(a)}} + \frac{3c\log t}{n_t(a)}. \tag{2}$$

UCB-V still has the exploration constant in the third term, but with growing $n_t(a)$ the exploration constant loses influence and the estimated variances are more strongly considered. To be precise, the second term has $\sqrt{n_t(a)}$ times the weight of the third term.

C. POSLB

The Pareto Optimal Sampling for Lipschitz Bandits (POSLB) algorithm assumes the expected rewards to be Lipschitz continuous and thereby improves the efficiency of sampling by guiding the bandit faster to more rewarding arms [22, p. 21]. Although Lipschitz continuity in the reward function values is assumed, the set of arms is still discrete. The expected reward function over arms is assumed to obey

$$|\mu(a) - \mu(a')| \leq \mathcal{L}\,|a - a'|$$

for any pair of arms (a, a') and a Lipschitz constant $\mathcal{L}$. The POSLB algorithm for Kullback-Leibler divergence of Gaussian distributed rewards is given in Algorithm 2. The algorithm identifies the currently best arm a_t^* and calculates the KL-UCB index $b_t(a_t^*)$. The intermediate values $\big(\lambda_t(a, a_1), \dots, \lambda_t(a, a_K)\big)$ can be understood as the *most confusing* estimated reward vector, which would make the suboptimal arm a the optimal one. The Lipschitz assumption is integrated into the bandit via the confusing rewards and is adjusted by $\mathcal{L}$. POSLB looks at the differences between the most confusing rewards and the actual estimated rewards and favors arms where the sum of the differences, weighted by their visit counts, is small.

The algorithm runs with an increased complexity of $O(|\mathcal{A}|^2)$ compared to UCB.

Algorithm 2: POSLB

```
if $t \leq K$ then
    Choose arm from $\{a : n_t(a) = 0\}$ at random
else
    $a_t^* = \arg\max_{a \in \mathcal{A}} \hat{\mu}_t(a)$
    $f_t(a) = \begin{cases} \sum_{a' \in \mathcal{A}} n_t(a')\big(\hat{\mu}_t(a') - \lambda_t(a,a')\big)^2 (2\sigma^2)^{-1} & \text{if } a \neq a_t^* \\ n_t(a)\big(\hat{\mu}_t(a) - b_t(a_t^*)\big)^2 (2\sigma^2)^{-1} & \text{if } a = a_t^* \end{cases}$
    with
    $\lambda_t(a,a') = \max\big(b_t(a_t^*) - \mathcal{L}\,|a - a'|\,,\ \hat{\mu}_t(a')\big)$
    Choose arm $a_t = \arg\max_{a \in \mathcal{A}} \log t - f_t(a)$
```

D. POSLB-V

While POSLB is able to consider the Lipschitz continuity of the reward function, it does not make use of estimated variances. Simply replacing the variance parameter σ^2 in the POSLB algorithm by the estimated variances $\hat{\sigma}^2(a)$ leads to poor results. Instead, it is advisable to shift from exploration constant to estimated variances over time, like it is done in UCB-V.

Equating the KL-UCB and UCB-V indices and solving for the variance parameter leads to

$$\sigma_t^2(a) = \sigma^2 = \frac{n_t(a)}{2\log t}\left(\sqrt{\frac{2\hat{\sigma}_t^2(a)\log t}{n_t(a)}} + \frac{3c\log t}{n_t(a)}\right)^2.$$

We then use the estimated variances and Lipschitz assumption together and and call it POSLB-V.

The augmented variance $\sigma_t^2(a)$ is then used in POSLB to calculate $b_t(a_t^*)$ and $f_t(a)$, instead of the fixed σ^2. Otherwise, the algorithm stays unchanged.

E. Continuous bandits

All bandits above need to discretize the continuous action space. However, there are several bandit algorithms that can handle continuous action spaces [23]–[25]. Other MABs additionally assume that the returns of the arms are Lipschitz continuous [26]–[29]. An overview is provided in [20, p. 40]. In every round they choose a new arm from the action space, never sampling the same action twice. This prohibits the usage of these bandits within the POMCP algorithm, which needs to build a belief tree. The bandit in [30] can deal with a large amount of discrete actions, but cannot easily be utilized in a belief tree. None of these approaches is compatible with real-time POMCP and, therefore, are not investigated further.

V. Experimental Settings

We adapt the POMCP algorithm [1] by modifying the `simulation`, `rollout`, and `backup` phases.

In the `simulation` phase of POMCP, we compare the MABs presented in the previous sections for choosing actions.

In the `rollout` phase we perform constant velocity rollouts for their minimal computational time.

We modify the `backup` phase in several ways. We use incremental statistics [31] to calculate the mean and the variance of Q-values

$$Q_n = Q_{n-1} + \eta(n)\ \big(\widehat{Q} - Q_{n-1}\big), \tag{3}$$

$$\sigma_n^2 = \big(1 - \eta(n)\ \big)\Big(\sigma_{n-1}^2 + \eta(n)\big(\widehat{Q} - Q_{n-1}\big)^2\Big). \tag{4}$$

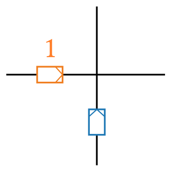
We alter the learning rate $\eta(n) = \frac{1}{n^\omega}$ by choosing a *polynomial learning rate* $\omega < 1$ instead of a *linear* rate where $\omega = 1$. As pointed out in [32], setting $\omega = 0.77$ has superior convergence properties. We use the maximum of Q-values as an estimate of the belief nodes value

$$V(h) = \max_{a \in \mathcal{A}} Q(h,a).$$

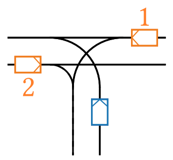
The belief tree of the POMDP solver works only with discrete observations. Therefore, we discretize the continuous observation space in a data stream clustering manner: A list of cluster centers is maintained and every new observation is compared to the entries in this list. If the Euclidean distance between observation and a cluster center is within a given threshold, the observation is assigned to the first matching cluster encountered in the ordered list. Otherwise, a new cluster center is inserted at the end of the list.

VI. Evaluation

We used two simple traffic scenarios "straight driving" and "traversing curves" for initial testing and development. For evaluation, we use two complex traffic scenes in which interaction with other participants are required (cf. Fig. 1). As the solution depends on the initial belief, we sample the state-particles of the initial belief from the same probability distribution. We assume that the distributions of position, velocity and route are independent.



(a) Collision scene.



(b) Intersection scene.

Fig. 1: Traffic scenes used in the evaluations.

Collisions pose a counter-example to the underlying assumption in this work: they introduce an abrupt change in the reward and hence pose the most challenging problem. In the collision scenario $\texttt{S}_{\texttt{Coll}}$ we simulate the case of an imminent collision. In the intersection scene, the ego vehicle has to identify whether the other vehicles are on collision paths and avoid collisions. We define two scenarios based on this scene: the $\texttt{S}_{\texttt{I-Lo}}$ scenario and the $\texttt{S}_{\texttt{I-Hi}}$ scenario, which pose low and high probabilities of collision, respectively. The parameters of the scenarios are provided in Appendix.

A. *Q-value Function Analysis*

If the transition model and reward function of the ego vehicle are Lipschitz continuous, the resulting Q-value function of an MDP is guaranteed to be Lipschitz continuous as well [33]. To expand the result to POMDPs, intuitively the observation function needs to be Lipschitz continuous as well. This is not the case in our model, as the reward includes the binary collision term, rendering the Q-value function discontinuous. We argue, however, that the noise, which is present in the transition and observation model, has a smoothing effect on these discontinuities.

In order to empirically analyze the continuity of Q-value function, we accurately evaluate it in the root node of the belief tree. We set up equidistant actions with $\Delta a = 0.05\,\mathrm{m\,s^{-2}}$ and thereby cover the action space densely. Then, the Q-values of these actions are evaluated by conducting a simulation run for each of the actions with 10^6 particles. During a single run, the action in the root node is kept fixed, whereas in the following belief nodes UCB is used to select among five available actions.

To highlight the discontinuity in the reward, we evaluate $\mathtt{S_{Coll}}$ without uncertainties (cf. Fig. 2). If the agent chooses an action between $-2.0\,\mathrm{m\,s^{-2}}$ and $-0.2\,\mathrm{m\,s^{-2}}$ a collision cannot be prevented. The step-like patterns emerge due to discretized values of braking actions in subsequent timesteps.

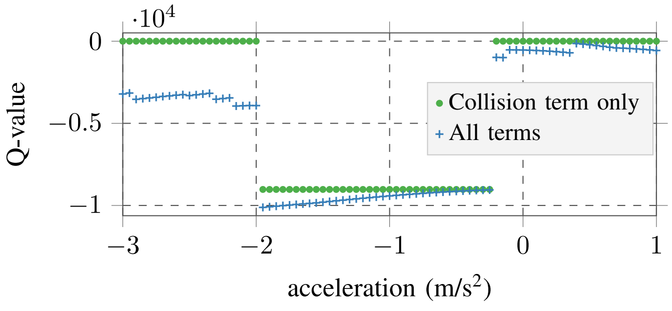


Fig. 2: Q-value profile of $\mathtt{S_{Coll}}$ without uncertainty.

Coarse and more accurate approximations of the Q-value profiles of $\mathtt{S_{Coll}}$ are presented in Fig. 3. The coarse one is approximated with 10^4 particles, whereas the more accurate one is obtained by sampling 10^7 particles with 17 actions, and doubled resolution of the observation discretization. The Q-value profile converges to a continuous function. From the more accurate simulation, it can be seen that the variance is reduced. However, as a result of reduced smoothing, the variance between successive actions tend to be higher than the rest at discontinuities. This points out that the Lipschitz assumption loses its validity for huge amount of samples under low uncertainty.

The Q-value profile of $\mathtt{S_{I\text{-}Lo}}$ and $\mathtt{S_{I\text{-}Hi}}$ are presented in Fig. 4. Both are very smooth and have the same underlying shape, whereas the overall level of Q-values is less and the variances are higher in $\mathtt{S_{I\text{-}Hi}}$, as a result of the higher collision probability.

The continuity evaluation illustrates that the uncertainties

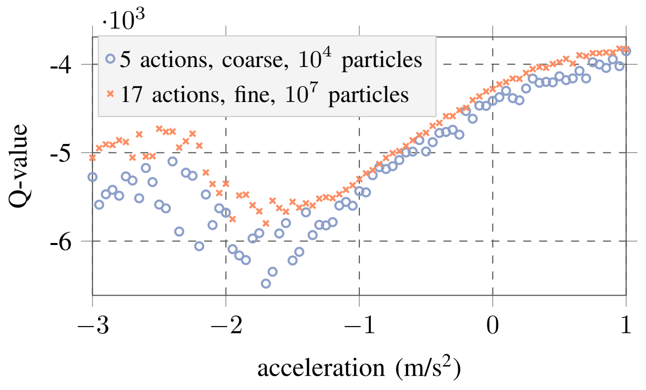


Fig. 3: Approximations of the Q-value profile of $\mathtt{S_{Coll}}$.

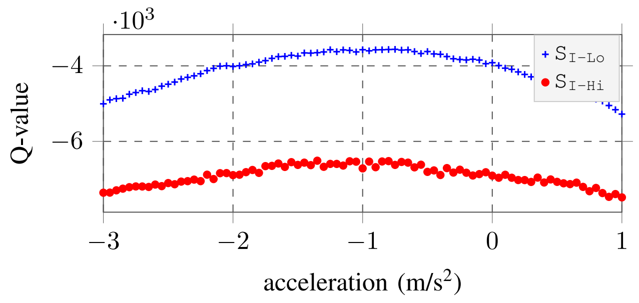


Fig. 4: Approximated Q-value profiles of the intersection scenarios sampled with 10^4 particles.

partially smooth the discontinuities. Increasing the number of available actions increases the smoothness of the profile, as well. Notice that we model uncertainties in the longitudinal direction only. Considering the lateral position uncertainty would further smooth the Q-values.

B. *Convergence Analysis*

A standard metric to benchmark convergence is to identify the optimal action after a predefined number of samples n. We use the *mean absolute error* (MAE) between the current best action a_n^* and the optimal action a^* as a performance measure.

We calculate a^* by sampling 10^7 particles while employing UCB bandit. The observed Q-values still have stochastic nature and hence, we perform Gaussian process regression to eliminate the noise of the Q-values and to recover the underlying function. As the profile of the Q-values is sufficiently smooth, we use a squared exponential kernel with length scale and noise level as hyperparameters. We estimate the optimal action for every scenario and number of available actions by evaluating the mean of the fitted Gaussian process at the locations of the available actions. The results are given in TABLE I in APPENDIX.

To perform a reliable analysis, we calculate the MAE over multiple simulation runs m

$$\mathrm{MAE}_n = \frac{1}{m}\sum_{i=0}^{m-1} \left|a_{i,n}^* - a^*\right|,$$

where $m = 100$ and $2 \cdot 10^4$ episodes. Given the ground truth, we determine a_n^* as

$$a_n^* = \arg\max_{a \in \mathcal{A}} Q_n(h_0, a).$$

The optimal Lipschitz constant required for POSLB is not known in advance. Overestimating and underestimating lead to suboptimal performance. We use the mean of the fitted Gaussian process and empirically select the Lipschitz constant $\mathcal{L} = 2000$ for these scenarios (cf. TABLE II in APPENDIX).

Fig. 5 presents the convergence results for $\texttt{S}_{\texttt{Coll}}$. From the figures, it is clear that for a low number of available actions all of the bandits have comparable convergence properties. However, as the number of actions increases, POSLB and POSLB-V show superior performance. In the case of 33 actions, the UCB bandits have twice the MAE compared to Lipschitz bandits after $2 \cdot 10^4$ episodes. The variants considering variances perform comparably. Another obvious result is that none of them can reach zero MAE. The value they reach is equal to the action discretization Δa, as expected. Even though not presented, the results for 17 actions lay between those for 9 and 33.

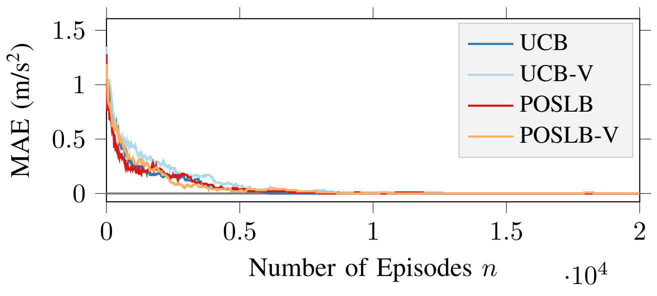
The results for 9 and 33 actions of $\texttt{S}_{\texttt{I-Lo}}$ are given in Figure 6. The results for $\texttt{S}_{\texttt{I-Hi}}$ are very similar to those of $\texttt{S}_{\texttt{Coll}}$ (cf. Fig. 5), as expected. Strikingly, the POSLB bandit shows the slowest convergence in the case of 33 actions, and POSLB-V performs best. The poor performance of POSLB is caused by misleading rollouts which point to a different area of the action space to be optimal. The Lipschitz assumption causes the bandit to select actions in that area. By selecting actions with higher variances more often, POSLB-V compensates the drawbacks resulting from such misleading rollouts. $\texttt{S}_{\texttt{I-Lo}}$ resembles such a narrow case in which the consideration of variances is advantageous.

C. *Discussion*

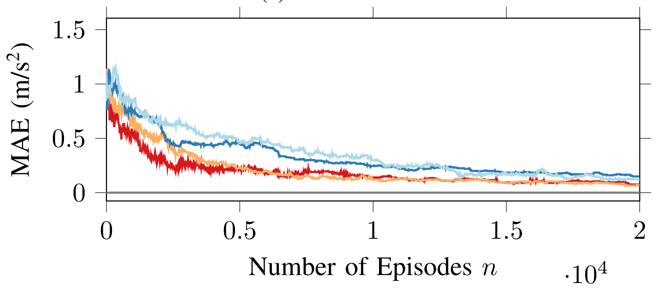
The results of the convergence analysis present the average over $m = 100$ runs. We analyze the standard deviation of MAE (σ_{MAE}) for individual runs of different bandits. The results indicate that the σ_{MAE} values are comparable, whereas POSLB bandits have slightly smaller σ_{MAE} then their UCB counterparts. The results for 9 actions in $\texttt{S}_{\texttt{Coll}}$ are presented in TABLE III in APPENDIX as an arbitrary example.

Tree depth of a solution is an important indicator of the quality: deeper trees consider longer horizons and are more accurate. In Fig. 7 we compare the tree depth for UCB and POSLB bandits for $\texttt{S}_{\texttt{Coll}}$ with the same number of particles. The bars in the figure represent the number of created nodes, and the color scales represent the number of visits for each level of the tree. It is clear that UCB has a greater branching factor compared to POSLB, resulting in shallower trees. In our experiments, other scenarios have verified this result.

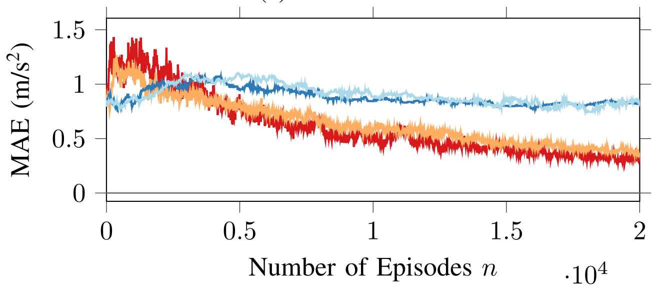
Although the Lipschitz bandits have an increased computational complexity over UCB, we observed in our experiments that their runtime is never longer than 10% longer. This applies to the most demanding case of 33 actions. On average, POSLB is 5% slower.



(a) 5 actions.



(b) 9 actions.



(c) 33 actions.

Fig. 5: Mean absolute error for $\texttt{S}_{\texttt{Coll}}$, for different numbers of actions.

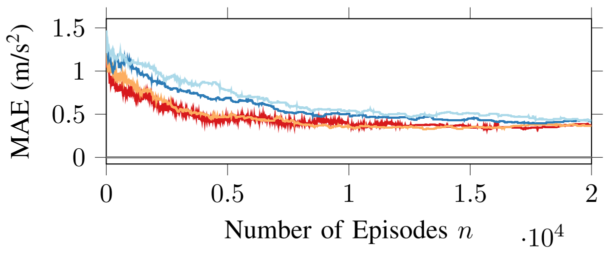
VII. CONCLUSIONS

In this paper, we present two important results. As a first result, we empirically show that the uncertainty in the transition and observation models of the POMDP formulation have a smoothing effect on the discontinuities in the Q-value function, eventually allowing for a Lipschitz continuity assumption.

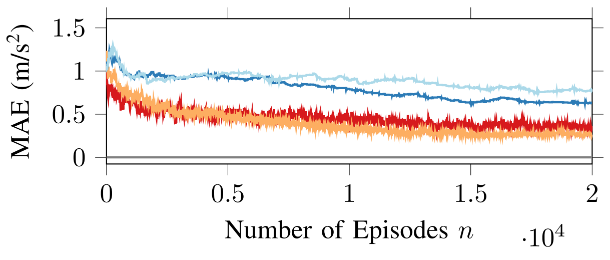
We further show that the planning problem can be solved with fewer samples by utilizing the continuity of Q-values. By replacing the standard multi-armed bandit (UCB) with one that assumes Lipschitz continuity (POSLB), considerable performance improvements are achieved for higher numbers of actions, especially in the early stages of sampling.

A further contribution is the POSLB-V bandit that is derived from the POSLB bandit. Motivated from UCB-V, it considers the variance of Q-values during the action selection. Experiments have shown that considering variances can be advantageous, in cases where the rollout policy might be misleading.

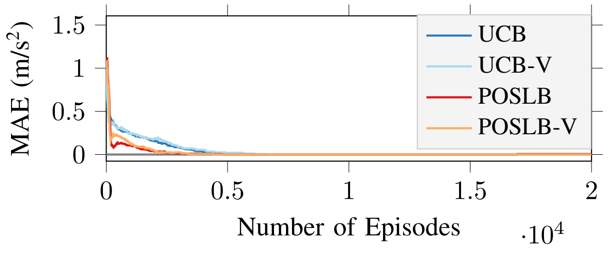
Real-time capability constraints have limited the use of



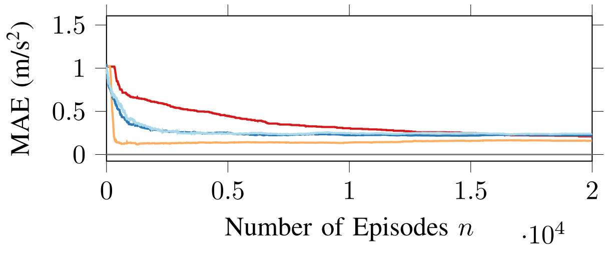
(a) $\mathtt{S}_{\mathtt{I\text{-}Hi}}$, 9 actions.



(b) $\mathtt{S}_{\mathtt{I\text{-}Hi}}$, 33 actions.



(c) $\mathtt{S}_{\mathtt{I\text{-}Lo}}$, 9 actions.



(d) $\mathtt{S}_{\mathtt{I\text{-}Lo}}$, 33 actions.

Fig. 6: Mean absolute error for the intersection scenarios for different number of actions.

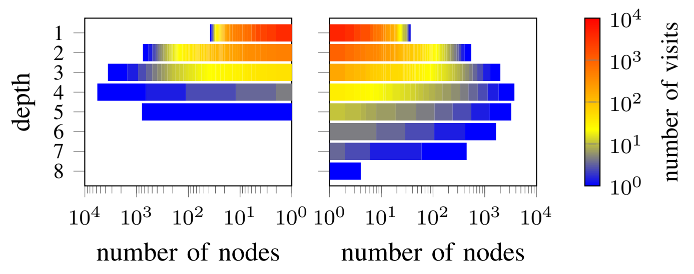


Fig. 7: Tree depth of UCB (left) and POSLB (right) bandits in $\mathtt{S}_{\mathtt{Coll}}$ for the same number of episodes.

existing POMDPs to decision making problems with fewer actions. With this work, we are able to accelerate the speed of POMDPs with a novel tree expansion technique that exploits the Q-value structure of our problem. This enables the use of POMDPs for problems where multiple actions need to be considered, such as in motion planning.

Appendix

TABLE I: Optimal action a^* $(\mathrm{m\,s^{-2}})$.

$\lvert\mathcal{A}\rvert$	Straight	Curve	$\mathtt{S}_{\mathtt{Coll}}$	$\mathtt{S}_{\mathtt{I\text{-}Lo}}$	$\mathtt{S}_{\mathtt{I\text{-}Hi}}$
5	0.0	-1.0	1.0	-1.0	-1.0
9	0.0	-1.5	1.0	-1.0	-1.5
17	0.0	-1.5	1.0	-1.0	-1.25
33	0.0	-1.0	0.875	-0.875	-1.125

TABLE II: Estimated Lipschitz constant $\mathcal{L}$ $(\mathrm{s^2\,m^{-1}})$.

$\lvert\mathcal{A}\rvert$	Straight	Curve	$\mathtt{S}_{\mathtt{Coll}}$	$\mathtt{S}_{\mathtt{I\text{-}Lo}}$	$\mathtt{S}_{\mathtt{I\text{-}Hi}}$
5	1247	1847	1003	1241	573
9	1271	2157	1981	1345	728
17	1336	2280	2742	1453	787
33	1370	2246	1260	1432	1033

Bandit	Number of Episodes						
	10^0	10^1	10^2	10^3	10^4	$2 \cdot 10^4$	10^5
UCB	0.57	0.57	0.81	0.55	0.29	0.23	0.07
UCB-V	0.52	0.52	0.66	0.53	0.32	0.21	0.09
POSLB	0.49	0.49	0.90	0.55	0.25	0.18	0.00
POSLB-V	0.65	0.65	0.77	0.57	0.21	0.17	0.07

TABLE III: σ_{MAE} for 9 actions of $\mathtt{S}_{\mathtt{Coll}}$.

TABLE IV: Times until the point-of-conflicts with different routes for the vehicles presented in the scenarios.

Scenario	Vehicle	Time-to-Intersection (s)			
$\mathtt{S}_{\mathtt{Coll}}$	ego	2.11			
	vehicle2	2.71			
$\mathtt{S}_{\mathtt{I\text{-}Lo}}$	ego	5.33	5.14	6.81	
	vehicle1	3.99	4.20	4.78	
	vehicle2	6.35	6.14	7.89	7.73
$\mathtt{S}_{\mathtt{I\text{-}Hi}}$	ego	2.66	2.28	5.63	
	vehicle1	2.78	3.23	4.52	
	vehicle2	3.42	3.05	6.21	5.92

TABLE V: Parameters used for evaluation.

Parameter	Value	Unit
ζ_{ref}	1	-
ζ_{lon}	35	-
ζ_{lat}	50	-
T_s	1	s
a^-	-3	$\mathrm{m\,s^{-2}}$
σ_a	3	$\mathrm{m\,s^{-2}}$
$t_{c,\min}$	1	s
$t_{c,\max}$	5	s
l_{goal}	15	m
l_{veh}	2	m
l_{idm}	2	m
T_{idm}	1.5	s
a_{cmf}	0.73	$\mathrm{m\,s^{-2}}$
b_{cmf}	1.67	$\mathrm{m\,s^{-2}}$
$\sigma_{o,\text{pos}}$	0.2	m
$\sigma_{o,\text{vel}}$	1.0	$\mathrm{m\,s^{-1}}$
d_{thresh}	1	m
ζ_{coll}	$-10\,000$	-
ζ_v	-100	-
$\zeta_{j,\text{lon}}$	-100	-
$\zeta_{a,\text{lat}}$	-100	-
ω	0.77	-
γ	0.95	-
c	$10\,000$	-
$\mathcal{L}$	2000	$\mathrm{s^2\,m^{-1}}$
$d_{\text{roll},\max}$	20	-

References

[1] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Advances in Neural Information Processing Systems*, 2010, pp. 2164–2172.

[2] 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.

[3] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time Analysis of the Multiarmed Bandit Problem," *Machine Learning*, vol. 47, no. 2, pp. 235–256, May 2002.

[4] L. Kocsis and C. Szepesvári, "Bandit Based Monte-Carlo Planning," in *Machine Learning: ECML 2006*, ser. Lecture Notes in Computer Science, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds. Springer Berlin Heidelberg, 2006, pp. 282–293.

[5] Y. Bai, Z. J. Chong, M. H. Ang, and X. Gao, "An online approach for intersection navigation of autonomous vehicle," in *IEEE International Conference on Robotics and Biomimetics*, 2014, pp. 2127–2132.

[6] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs," in *International IEEE Conference on Intelligent Transportation Systems*, 2014, pp. 392–399.

[7] S. Brechtel, "Dynamic Decision-making in Continuous Partially Observable Domains: A Novel Method and its Application for Autonomous Driving." Ph.D. dissertation, Karlsruhe Institute of Technology, 2015.

[8] Z. N. Sunberg, C. J. Ho, and M. J. Kochenderfer, "The value of inferring the internal state of traffic participants for autonomous freeway driving," in *2017 American Control Conference (ACC)*. IEEE, 2017, pp. 3004–3010.

[9] M. Bouton, A. Cosgun, and M. J. Kochenderfer, "Belief state planning for autonomously navigating urban intersections," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 825–830.

[10] M. Sefati, J. Chandiramani, K. Kreisköther, A. Kampker, and S. Baldi, "Towards tactical behaviour planning under uncertainties for automated vehicles in urban scenarios," in *IEEE International Conference on Intelligent Transportation Systems*, 2017, pp. 1–7.

[11] H. Bai, D. Hsu, W. S. Lee, and V. A. Ngo, "Monte Carlo value iteration for continuous-state POMDPs," in *Algorithmic Foundations of Robotics IX*. Springer, 2010, pp. 175–191.

[12] H. Kurniawati, D. Hsu, and W. S. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces." in *Robotics: Science and Systems*, 2008.

[13] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer, "Scalable Decision Making with Sensor Occlusions for Autonomous Driving," in *IEEE International Conference on Robotics and Automation*, May 2018, pp. 2076–2081.

[14] C. Hubmann, J. Schulz, M. Becker, D. Althoff, and C. Stiller, "Automated Driving in Uncertain Environments: Planning With Interaction and Uncertain Maneuver Prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 1, pp. 5–17, Mar. 2018.

[15] C. Hubmann, N. Quetschlich, J. Schulz, J. Bernhard, D. Althoff, and C. Stiller, "A pomdp maneuver planner for occlusions in urban scenarios," in *Proc. IEEE Intell. Veh. Symp.*, 2019, pp. 2172–2179.

[16] K. Kant and S. W. Zucker, "Toward Efficient Trajectory Planning: The Path-Velocity Decomposition:," *The International Journal of Robotics Research*, Jul. 2016.

[17] M. Treiber, A. Hennecke, and D. Helbing, "Congested Traffic States in Empirical Observations and Microscopic Simulations," *Physical Review E*, vol. 62, no. 2, pp. 1805–1824, Aug. 2000.

[18] C. Ericson, *Real-Time Collision Detection*. CRC Press, 2004.

[19] J. T. Barron, "A General and Adaptive Robust Loss Function," *arXiv:1701.03077 [cs, stat]*, Jan. 2017.

[20] A. Slivkins, "Introduction to multi-armed bandits," *arXiv preprint arXiv:1904.07272*, 2019.

[21] J.-Y. Audibert, R. Munos, and C. Szepesvári, "Tuning Bandit Algorithms in Stochastic Environments," in *Algorithmic Learning Theory*, ser. Lecture Notes in Computer Science, M. Hutter, R. A. Servedio, and E. Takimoto, Eds. Springer Berlin Heidelberg, 2007, pp. 150–165.

[22] S. Magureanu, "Efficient Online Learning under Bandit Feedback," Ph.D. dissertation, KTH Royal Institute of Technology, 2018.

[23] C. Mansley, A. Weinstein, and M. Littman, "Sample-based planning for continuous action markov decision processes," in *Twenty-First International Conference on Automated Planning and Scheduling*, 2011.

[24] A. Weinstein and M. L. Littman, "Bandit-based planning and learning in continuous-action markov decision processes," in *International Conference on Automated Planning and Scheduling*, 2012.

[25] A. D. Bull *et al.*, "Adaptive-treed bandits," *Bernoulli*, vol. 21, no. 4, pp. 2289–2307, 2015.

[26] R. Kleinberg, A. Slivkins, and E. Upfal, "Multi-armed bandits in metric spaces," in *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM, 2008, pp. 681–690.

[27] S. Bubeck, G. Stoltz, C. Szepesvári, and R. Munos, "Online optimization in x-armed bandits," in *Advances in Neural Information Processing Systems*, 2009, pp. 201–208.

[28] O.-A. Maillard and R. Munos, "Online learning in adversarial lipschitz environments," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2010, pp. 305–320.

[29] S. Bubeck, G. Stoltz, and J. Y. Yu, "Lipschitz bandits without the lipschitz constant," in *International Conference on Algorithmic Learning Theory*. Springer, 2011, pp. 144–158.

[30] E. Wang, H. Kurniawati, and D. P. Kroese, "CEMAB: A cross-entropy-based method for large-scale multi-armed bandits," in *Australasian Conference on Artificial Life and Computational Intelligence*. Springer, 2017, pp. 353–365.

[31] T. Finch, "Incremental calculation of weighted mean and variance," *University of Cambridge*, vol. 4, no. 11-5, pp. 41–42, 2009.

[32] E. Even-Dar and Y. Mansour, "Learning Rates for Q-learning," *Journal of Machine Learning Research*, vol. 5, no. Dec, pp. 1–25, 2003.

[33] K. Asadi, D. Misra, and M. L. Littman, "Lipschitz continuity in model-based reinforcement learning," *arXiv preprint arXiv:1804.07193*, 2018.