Probabilistic MDP-Behavior Planning for Cars

Conference Paper *in* Conference Record - IEEE Conference on Intelligent Transportation Systems · October 2011

DOI: 10.1109/ITSC.2011.6082928

Sebastian Brechtel
Karlsruhe Institute of Technology
12 PUBLICATIONS **1,118** CITATIONS
SEE PROFILE

Rüdiger Dillmann
Karlsruhe Institute of Technology and FZI
1,044 PUBLICATIONS **17,216** CITATIONS
SEE PROFILE

2011 14th International IEEE Conference on
Intelligent Transportation Systems
Washington, DC, USA. October 5-7, 2011

Probabilistic MDP-Behavior Planning for Cars

Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann
Institute for Anthropomatics
Karlsruhe Institute of Technology
D-76128 Karlsruhe, Germany
Email: {brechtel | gindele | dillmann}@kit.edu

***Abstract*— This paper presents a method for high-level decision making in traffic environments. In contrast to the usual approach of modeling decision policies by hand, a Markov Decision Process (MDP) is employed to plan the optimal policy by assessing the outcomes of actions. Using probability theory, decisions are deduced automatically from the knowledge about how road users behave over time. This approach does neither depend on an explicit situation recognition nor is it limited to only a variety of situations or types of descriptions. Hence it is versatile and powerful. The contribution of this paper is a mathematical framework to derive abstract symbolic states from complex continuous temporal models encoded as Dynamic Bayesian Networks (DBN). For this purpose discrete MDP states are interpreted by random variables. To make computation feasible this space grows adaptively during planning and according to the problem to be solved.**

I. Introduction

High-level behavior decision making for autonomous cars or advanced driver assistance systems (ADAS) is usually tackled by manually modeling behavior rules. Decision making in this case means to choose the appropriate high-level action, which is usually executed with some sort of low-level intelligence to react to small deviations of the standard situations (see Fig. 1). Due to the highly dynamic, continuous and stochastic nature of the traffic environment and the humongous number of situations, manually created decision rules are predestined to failure. Additionally, every situation has numerous relevant features and to completely perceive the environment is impossible. Even the best sensors and algorithms will never be invulnerable to occlusion. Wireless communication of traffic participants' states cannot transmit intentions. All these properties contribute to the difficulty of the generally formulated task *'drive a car'*. With growing problem complexity this procedure inevitably leads to crucial inconsistencies and loops in manually created sets of rules, even when using modern hierarchical and concurrent architectures to model the experts knowledge. These inconsistencies could be detected with formal verification methods, but the latter cannot find the correct solution. One *'emergency behavior'* does not fit every situation.

In this paper we propose a method, which concludes safe, goal-oriented, efficient and comfortable decisions from its knowledge about and understanding of the traffic environment including its participants. We formulate the driving task as MDP [1]. This enables the system to cope with uncertainties in the temporal evolution of the situation, like e.g. induced by unexpected sudden braking of other cars. The transition model of the MDP encapsulates such probabilities. It is derived from a hierarchical DBN, similar to the one described in [2], by discretizing continuous models. This allows for more detailed and general models and even using multiple abstraction levels in one MDP optimization.In combination with a reward model, rating how desirable situations are, this information is sufficient to formulate and solve the optimization problem.

After an overview on related work and MDPs, we show how a continuous space prediction can be used for discrete MDPs in general and in specific for the driving problem.

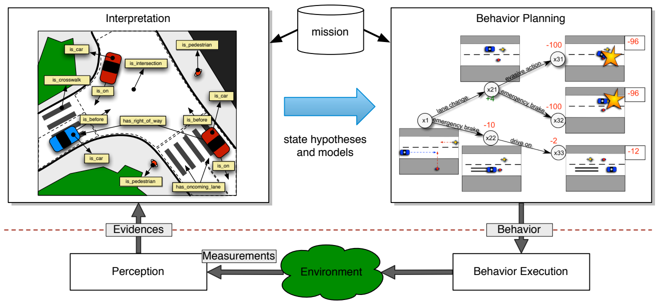


Fig. 1: System Overview

II. Related Work

Manual modeling of symbolic states and decisions is at least since the Urban Challenge in 2007 the most common approach for decision making of autonomous cars. Robotics knows several architectures and methods to efficiently represent knowledge about a robots environment and solutions to its task. In the domain of autonomous driving, among others (Petri-Nets, Behavior Nets [3], to name a few), hierarchical state machines have preferably been used to assess situations and decide in the same framework (e.g. in [4]). While those approaches rely on symbolic descriptions of the world, one could also abandon every human-understandable description. The purely reactive system ALVINN [5] used Neuronal Nets to deduce a control directly from camera images. Forbes *et al* [6] presented a fundamentally different, deliberative approach, which extensively uses stochastic theory to model the uncertain behavior of the traffic environment. They used supervised and model-free reinforcement learning techniques, combined with an MDP problem description and dynamic probabilistic networks for the transition probabilities. This approach showed more promising results compared to decision trees and other classical methods in highway scenarios.

MDPs are a mathematically sound formulation for decision and control problems encountering uncertain system behavior [1]. They can be extended with hidden states to Partially Observable MDPs (POMDP), able to cope with uncertain and incomplete perception. There are two main solutions for MDP optimization problems: on the one hand reinforcement learning [7] and on the other hand direct optimization techniques, like policy or value iteration [8]. In the following, the latter are denoted as *planners*. Dealing with partial observability, the problem becomes exponentially more complex. Hence, many extensions to find approximate solutions for POMDPs have been proposed. The main branch of extensions is based on point-based value iteration [9]. The key idea is to only consider a finite subset of points in the continuous belief space. These approaches usually need a fixed set of discrete states. In [10], methods to adaptively control the resolution of this discretization in MDPs are compared. Alternatively one could directly plan in the continuous space with parametric distributions, which suffer from the high calculation complexity [11], [12].

III. Markov Decision Process Preliminaries

While it is impossible to cover MDPs in detail in this paper, this section gives a short overview about the used terminology. A discounted MDP with infinite horizon is specified by a tuple $(S, A, T, R, p_0, \gamma)$ [1]. S denotes the set of discrete states $s \in S$ in the world with the initial probability $p_0(s)$. The transition model (also known as process or system model) T is the conditional probability function $T(s', a, s) = p(s'|a, s)$. It describes the probability that a system evolves from state s to s', when the agent chooses action $a \in A$. The real-valued reward function $R(s, a)$ expresses how desirable a state-action pair is for an agent.

The aim of the MDP optimization is to find an (approximately) optimal policy $\pi^* : s \to a$ that maximizes the expected reward (denoted value) over time:

$$\pi^* = \operatorname*{argmax}_{\pi} \sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \, .$$

Thanks to the discount $\gamma \in [0, 1)$, the value is finite and thus the problem well-defined. In this paper, value iteration is used to optimize the value. The calculation of the following Bellman equation will be referred to as *backup*:

$$V(s) = \max_{a \in A} \left(\sum_{s' \in S} T(s', a, s) \left(R(s', a) + \gamma V(s') \right) \right) .$$

It is repeated for every (known) state until convergence.

IV. MDP Decision Making for Cars

The presented decision making system concludes a policy by optimizing an MDP. The key paradigm of the approach is, besides an end-to-end probabilistic formulation, to minimize the manual encoding of expert knowledge. Fig. 1 shows an overview of the system. It decides for high-level behaviors on basis of probabilistic evidences. The objective function is deduced from the mission it has to accomplish, in addition to general driving aims such as safety. The two main difficulties of such an approach are to find a set of symbolic states that describes the world sufficiently and to give it a meaning in the real world by models. The latter can often be easily and precisely described in the continuous space, i. e. by a DBN. Thus, the following descriptions focus on a sound combination of the continuous and discrete descriptions of the world for planning.

A. State Space

Unfortunately, there exists no fixed optimal discretization of the continuous space for arbitrary problems. But in general (PO)MDP-solvers need a discrete description. In some situations usually negligible nuances make the important difference and one cannot know in advance which states of the world need to be considered or distinguished. The classical approach, first naming all possible states, then calculating the rewards and transition probabilities for all combinations of states and finally solving the MDP, does not work for a world with possibly infinite number of states.

1) Formal Definition: Hence, the universal sample space $S_c \subseteq \mathbb{R}^n$ and the MDP state space $S \subseteq \mathbb{N}$ need to be related. In this approach the relation is defined by an S-valued random variable $I_S : S_c \to S$ with probability space (S_c, Σ, P) and measurable space (S, Σ'). We choose Σ as the Borel σ–algebra on S_c and Σ' as the power set of S. Considering only elementary events s, the preimage then yields the corresponding set of continuous states $I_s^{-1}(s) = \{s_c : I_s(s_c) \in s\}$ for any discrete state $s \in S$.

2) Specific Realization: For our practical example, we divide the space into equidistant, non-overlapping regions of the coordinates long- and latitudinal to the road and the velocity of object o in its local frame. The orientation is assumed to be always aligned with the lane:

$$(\mathrm{pos}_{\mathrm{lat},o}, \mathrm{pos}_{\mathrm{lon},o}, \mathrm{vel}_{x_{\mathrm{car}},o}) = s_{c,o} \in S_{c,o} \, .$$

The final combined continuous space S_c is constructed as the power set of the autonomous car's ($o = 1$) and all other objects' ($o = 2, \ldots, n$) continuous spaces. The agent will in the following be denoted as *ego* car. The system is thus capable of handling an arbitrary number and combination of objects.

$$S_c = \mathcal{P}(\prod_{o \in \{1,\ldots,n\}} S_{c,o}) \, .$$

Consequently, inside the region of a state $s \in S$, we assume the a priori probabilities to be approximately uniformly distributed. More sophisticated schemes are thinkable when using arbitrary densities like for example letting one sample influence many states or even update the state's density description on the fly.

B. Action Space

The action space is defined analogously to the state space by the random variable I_A. We assume a set of high-level actions $a \in A \subset \mathbb{N}$ to be given. MDP actions a are currently defined as behaviors $b_e \in B_e$ in the DBN and thus directly associated by $I_A : B_e \to A$. We can activate b_e deliberately by giving a virtual evidence. As modeled by the DBN in Fig. 2, the ego behaviors yield a distribution over likely trajectories T_e, which are represented as parameteric Bezier curves. During their planned or real execution, behaviors can react on other objects and changes in the environment due to a rudimentary intelligence (including e.g. emergency stopping) encoded manually in the DBN. Additionally, they underly kinematic and dynamic constraints. Their outcome is therefore not known before execution and might differ extremely from the targeted trajectory in some cases.

The set of behaviors currently consists of combinations of accelerations and lateral velocities to allow for lane changes and obstacle avoidance. The Bézier parameters of T_e are derived from the tuple b_e:

$$(\mathrm{acc}_{x_{\mathrm{car}}}, \mathrm{vel}_{\mathrm{lat}}, \mathrm{dur}) = b_e \in B_e\,.$$

Further, the amount of time a behavior uses is part of the action. This duration allows for balancing of comfort and safety. It is even more important for using actions with different time horizons and levels of abstraction in the same MDP. While an acceleration of only 10 ms might make sense, a lane change maneuver will never be finished in less than 1 s and therefore not give reasonable results after some milliseconds. The option to choose actions with different durations results in a Semi-MDP (SMDP) [13]. This has some consequences regarding the discounting, which are of minor interest in this paper and therefore not mentioned. We also refer to the problem as a MDP most of the time. While in MDP theory it is no problem to evaluate only a subset of all possible actions (like we do in the state space), we use a fixed number of reasonable actions.

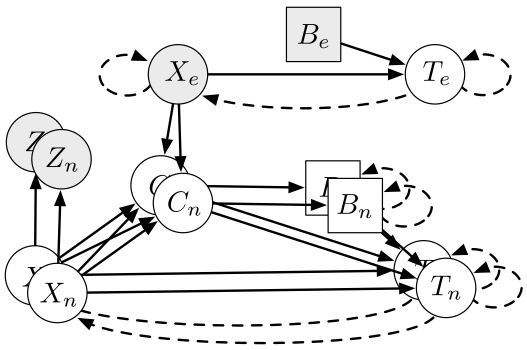


Fig. 2: DBN with ego and two other vehicles. Dashed lines indicate conditional dependencies over time.

C. Transition Model

The transition model is in our system even more important than normally in MDPs, because it also directs the growth of the planning space. Due to the link I_S between the MDP states and the continuous states DBN states, the conditional probability distribution is defined by the transition probabilities in the continuous space. Thus, every possible discrete state is covered.

1) Formal Definition: Since I_S is (Σ, Σ') measurable, the probability measure P induces a probability measure P^{I_S} on the power set $\mathcal{P}(S) = \Sigma'$. This yields for $D \in \mathcal{P}(S)$:

$$P^{I_S}(D) = P(I_S^{-1}(D)) = \int_{\omega \in I_S^{-1}(D)} p(w)\,.$$

Accordingly the state space transition function $P^T(S'|S, A)$ can be derived from the sample space transition function:

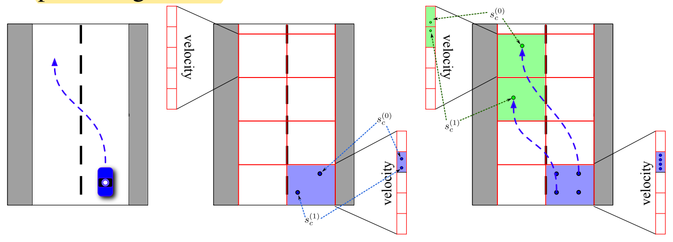
$$P^T(S'|S,A) = P(I_S^{-1}(S')|I_S^{-1}(S), I_A^{-1}(A)) = \int_{s'_c \in I_S^{-1}(S')} \int_{s_c \in I_S^{-1}(S)} \int_{a_c \in I_A^{-1}(A)} p(s'_c|s_c, a_c)\,. \quad (1)$$

2) Specific Realization: Our implementation uses Monte Carlo approximation for prediction. Therefore continuous probability density functions are approximated by finite dirac-mixtures. Eventually, a set of Q continuous particles and according weights represent an MDP state:

$$p_c = \{p_c^{(L)} \;:\; p_c^{(L)} = \langle w_c^{(L)}, s_c^{(L)} \rangle,\ L \in \{1, \ldots, Q\}\}\,.$$

The particles are predicted by the DBN. Note that any other continuous model representation could be used. Replacing the according integrals in Eq. 1 with sums, the discrete transition probabilities are finally approximated.

In Fig. 3 the simplified process of state sampling is visualized for a lane change maneuver and solely the ego vehicle in the state space. The practical procedure is very similar to the occupancy sampling in [14]. At the beginning of the solver's optimization, no specific discrete state is instantiated, except the currently observed. Important is that all of them are part of the reachable space and sophisticated solvers could utilize this further by calculating backups only for promising states.



(a) Scene and target trajectory. (b) Discretization and sampling (c) Transition and new follow-up states

Fig. 3: State space sampling and transition.

The SMDP action durations are currently between 1 to 3 s, while the DBN calculates with a fixed time step of 0.1 s. Hence, to calculate $p(s'|s,a)$ 10-30 DBN-steps are rolled out. The action a is fixed during execution, by setting virtual evidences for the corresponding b_e in every time step. During execution, among other information, the behavior of other traffic participants B_o is estimated, depending on their traffic context C_o (see Fig. 2 and [2] for details). If the autonomous car e.g. drives on the right lane and a much faster car approaches on the same lane from behind, a high probability for an overtaking maneuver is inferred and the other car will likely change the lane to the left. Therefore the result of the MDP optimization is to stay on the lane. A simple constant velocity model would result in a decision for a quick lane change to the left of the autonomous car, which would in reality most often implicate a crash. There are many other examples showing the mightiness and generality of that approach compared to planning with less sophisticated prediction algorithms.

D. Reward Model

The autonomous car in our example is meant to make way as fast as possible, while at the same time avoiding to hit other cars or to leave the road. Additionally, we want it to drive as economical and comfortable as possible and to obey the rules. All these requirements are aggregated in the reward function $R(s, a)$, which is evaluated in the DBN by a continuous reward function $R_c(s_{DBN})$. Especially the latter is noteworthy, because it allows to subsume all uncertainty during the execution of a in the reward function. If, e.g., the chance for a crash during the prediction is 0.5, because 50% of the particles encountered a crash, the result of $R(s, a)$ will be about one half of the total cost of a crash. The input of the reward calculation is the state of the DBN s_{DBN}, which contains far more information than s_c. For that reason we are able to evaluate the reward every $\Delta t_{DBN} = 0.1\,\mathrm{s}$. This enables us to calculate e.g. whether a crash happened very precisely by linearly interpolating between the small substeps. If applied to the multiple seconds long MDP time step, linearization would be insufficient, Also comfort can be easily described by the sum of all accelerations encountered executing the chosen behavior. The concrete reward calculation is outlined as follows:

```
**if** ($\text{is}_{\text{crashed?}}$) **then**
    $R_c \leftarrow \text{rew}_{\text{crash}}$
**else**
    $R_c \leftarrow \Delta t_{DBN} \cdot (\text{rew}_{\text{comfort}} \cdot \text{acc} + \text{rew}_{\text{make\_way}} \cdot \text{vel}_{\text{lat}} + \text{rew}_{\text{not\_rightmost\_lane}} \cdot !\text{is}_{(\text{on\_rightmost\_lane?})})$
**end if**
```

In the examples weights $\text{rew}_{\text{crash}} = -1000$, $\text{rew}_{\text{comfort}} = -12$, $\text{rew}_{\text{make\_way}} = 3$ and $\text{rew}_{\text{not\_rightmost\_lane}} = -6$ are used. The results show that changes of these parameters only slightly influence the resulting policy, which speaks for the robustness of the system. For example one can double the crash-cost without noticeable changes in the policy. Only when quadrupling them the car behaves noticeably more timidly. The value results, however, change with small variations of the reward parameters.

E. Online-Offline MDP-Solver

The system currently implements a combination of offline and online planning. It is impossible to calculate the complete policy for every possible state due to the possibly infinite number of states. On the other hand starting the planning from the scratch in every state – like classical online planners do – would take far too much time for the real-time requirements imposed by the traffic environment. Hence, starting from a few states we calculate a policy *offline*. This policy is a good starting point for further computations, but does not offer a complete solution with decisions for any state. Especially states, which are not or with low probability reachable from the starting states are often not covered. Therefore we refine the planning *online*, while the car is driving. We add the currently encountered state to the state space, if it does not yet exist. A persisting solver instance is running permanently in the background and updating the policy. The anytime–nature always allows to extract a valid solution when needed. The idea behind this strategy is that the planning will cover most probable temporal evolutions and the solver hence plans usually close to the really encountered path. Even if it did not see it coming, the situation only changes slightly during two decisions. Consequently, in most of the predicted futures, planning ends up in a known state, for which the value calculation is already close to optimal. This idea of online-offline-planning can also be ported from the state space to the belief space for a POMDP formulation.

The optimization algorithm currently used is a standard (S)MDP-solver with discounting, repeatedly applying backups to all existing states. If a new state is discovered it will be pushed back into a pool of states. There are no optimizations, parallelization or heuristics to improve the efficiency. It only stops calculating backups, when they would not influence the policy any more because of the discount. This keeps the calculation finite. Due to the sparse state space description, the solver would tremendously benefit from a heuristic, telling it which states are worth a backup and which are not. However, the focus of our implementation is on maintaining the extendability to a POMDP with the heuristics described in [15]. Particularly the ability of SARSOP to sample in the optimally reachable belief-space by calculating lower and upper bounds for the value in combination with its sparse representation of beliefs fits very well to the adaptively growing space description in this paper.

V. Evaluation

For the evaluations of the prototype we use a two-lane highway scenario as shown in Fig. 4.

The autonomous MDP-controlled ego-vehicle veh_{ego} (*car 1* in Fig. 4) has to cope with two other vehicles veh_2 and veh_3 and thus has to manage tasks like overtaking with oncoming traffic. A detailed analysis of the resulting policy as well as tests in an 3D simulation environment have been done. The latter is able to realistically resemble situations by accurately simulating sensor data and driving physics. It allows to manually control traffic participants and thereby guarantee close to natural driving.

A. Analysis of the Results

The plots in Fig. 5 show the results of the policy calculation when there is no oncoming traffic, just one vehicle veh_2 stopping at the right lane. The autonomous car veh_{ego}

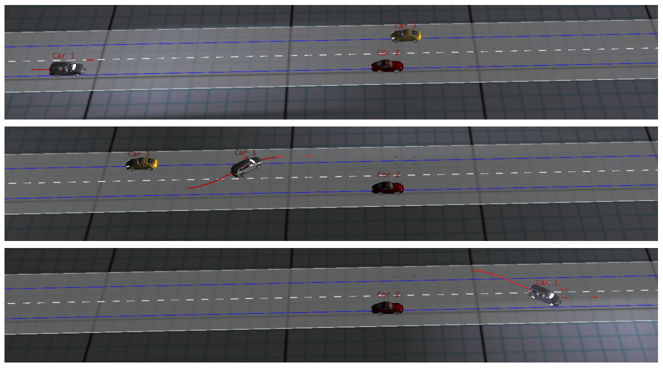
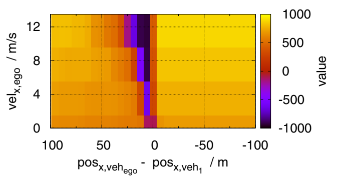
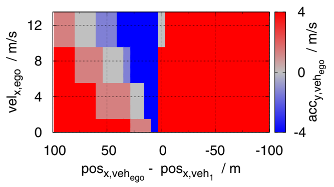


Fig. 4: Overtaking scenario in the simulation environment. Planned ego vehicle trajectory indicated by red line.

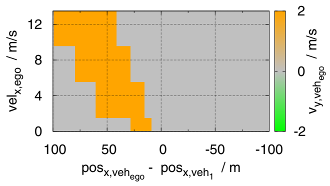
is also on the right lane. The x-axis of the plot shows the longitudinal distance between both cars. If this value is positive, veh_{ego} is situated behind veh_2. The current speed of the ego vehicle can be seen on the y-axis. For the discretization in this example we used regions of latitudinal length 6 m. In longitudinal direction only the two lanes with width 6 m were distinguished. The errors induced by this coarse discretization can be seen by the aliasing effect in all of the following plots.



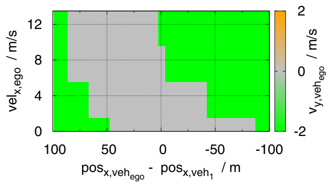
(a) Maximum value



(b) Acceleration

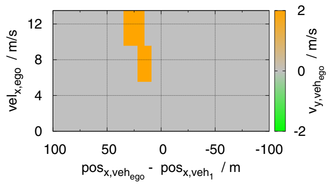


(c) Lane-change (ego on right lane)

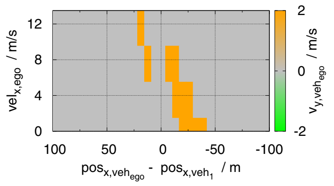


(d) Lane-change (ego on left lane)

Fig. 5: Results of policy calculation with one other car veh_1 at the right lane with speed $\text{vel}_{x_{\text{car}},o} = 0\,m/s$



(a) Complete set of behaviors



(b) Only *free ride* behavior

Fig. 6: Lane change policies with and without behavior estimation (veh_1 at right lane with speed $\text{vel}_{x_{\text{car}},o} = 7\,m/s$).

Fig. 5c shows when the ego car would overtake the other car. A lateral speed of 2 m/s of veh_{ego} is equivalent to a lane change to the left. The result for the acceleration part of the action is shown in Fig. 5b. It can be seen that the car changes the lane in a safe distance depending on its own speed. At the same time it reduces acceleration, because physical constraints limit accelerating and applying high longitudinal forces at the same time. If too close, it emergency brakes without changing the lane to be able to apply more braking forces. An evasive maneuver is already too risky at that distance. Once it has passed the other vehicle it just accelerates, because in this example no fuel costs are given. As can be seen in Fig. 5d, the car returns to the right lane, when it has reached a safe distance to avoid the cost of driving on the left lane. In Fig. 5a of the value, one can see how the system estimates situations. The closer the ego vehicle approaches the other car, the less it 'feels' safe, because there is the chance of the other car suddenly braking or changing the lane to the left. In the black areas with expected costs close to 1000, accidents are inevitable. It is noticeable that once the other car is passed, the value increases with the cars speed.

The simple example in Fig. 6 already shows the necessity of more sophisticated models than, e.g., constant velocity models, which are today mainly used for ADAS. The situation is basically the same as in the above example, except that veh_2 drives with 7 m/s on the right lane. The safety distance shrinks compared to the case with the standing obstacle, because there is more time to brake. The result in Fig. 6a was generated with the DBN's behavior estimation activated. In Fig. 6b the results for the lane change policy are given, when *free ride* is the only existing behavior. The missing uncertainty in the behavior results in a slightly smaller safety distance. More important the other car won't change lane during prediction, even when approaching the ego car quickly. Thus, the optimal action is an evasive maneuver to the left lane. In reality this is undoubtedly dangerous, because quickly approaching cars most certainly want to overtake. The behavior estimation of the DBN is able to predict this and consequently the ego car in Fig. 6a holds lane.

B. *Practical Driving Test*

The decisions of the system with oncoming traffic were practically evaluated in a 3D simulation environment. Several tests were repeated with the other cars being controlled manually with steering wheels. The system showed reasonable behavior, e.g., it follows the slower car in a decent distance until the oncoming traffic has passed, to be able to smoothly and quickly accomplish the subsequent overtaking maneuver. It also slows down when overtaking a car on the right to be able to react, if the other vehicle suddenly changes to the right lane.

C. *Performance*

The raw complexity analysis of MDPs or even POMDPs could cause doubts concerning the practical usability of the approach. We give some impressions of the runtime to eliminate those thoughts.

1) Offline Solving: The calculation times vary extremely depending on several factors, including the number of backups and states, the discount, the duration of the MDP actions as well as of the DBN, and last but certainly not least the amount of uncertainty in the model. The calculations with one other vehicle in Fig. 5 were done offline by sweeping over the whole space of positions from -100 to 100 m, from the left to the right lane and velocities 0 to 16 m/s for both veh_{ego} and veh_2, which in combination with the discount $\gamma = 0.95$ resulted in 3600 states in total. The system can choose between 9 actions with duration between 1 and 3 s. The DBN runs with fixed 64 particles and a time step 0.1 s and thus the DBN has to infer 10 to 30 time steps for one MDP prediction step. By repeating to backup every state in the space 7000 times, we got sure to obtain a close to optimal policy (usually a lot less backups are necessary). With these preliminaries the (offline) optimization needs about 22 min in total on a desktop computer. About half of the time is used for the DBN–inference, which could be sped up by using a longer time step. The other half of the time is used to look up states in the global state set and of course to calculate the backup itself. The lookup is implemented using a k-dimensional tree for the purpose of generality. Much quicker binary-trees or hash-maps can be used, when the rules of discretization are given a priori. A single lookup of an already optimized state for policy execution virtually needs no time at all (10 μs average) and could easily be done on a micro-controller.

2) Online-Offline Solving: For 3 cars pure offline solving is already impractically slow and memory consuming, due to the countless (in this case more than 20,000) possible combinations of states. The presented online-offline planning diminishes this drawback. For a realistic example we calculate the policy originating from one initial state. This takes ca. 1000 states and less than 10 min. While driving, new states can be instantiated online. The time until the value converges for that new state varies, but usually it takes less than 2 s and initializes only up to 10 new other states, if it is in the direct neighborhood of the initial state.

VI. Conclusion and Future Work

This paper presents a method for automatically inferring optimal behavior decisions based on sophisticated, probabilistic models describing the evolution of traffic situations. A theoretical approach of combining continuous world prediction by a DBN and discrete world MDP planning is given and evaluated analytically and practically in a simulation environment for the domain of autonomous driving. This combination is flexible and powerful at the same time and should be able to cover virtually any thinkable traffic scenario. The presented offline-online planning algorithm allows for decision making in before unknown scenarios with decent latency. Using offline planned policies for a set of standard situations, such as highway driving or merging into moving traffic, could be used for ADAS today. The results should be superior to established, for example, time-to-collision based methods and can be adapted to the drivers preferences by adjusting the reward parameters.

The presented system is meant to be extended to a POMDP to manage incomplete knowledge in future work. Additionally to the environment in an ADAS the drivers' condition (e.g. attention) could be estimated as latent variables. Thus the system could give advices only when needed, similar to the assistance of persons with dementia shown in [16].

Nevertheless, there are open problems especially regarding the computational complexity with growing numbers of objects and after extending the problem to a POMDP. Formulating decision problems with a fixed discretization is inherently inefficient. The results for the example in Fig. 5 show that huge areas of the state space could be aggregated. On the other side the used regions are too coarse, which leads to aliasing. An optimal discretization is, however, not known a priori. A cyclist driving parallel to the car, e.g., has to be considered only when the car wants to turn in his direction. For those reasons, we plan to replace the current discretization rules with arbitrary probability functions adapting to the tasks requirements for precision during the process of solving the (PO)MDP. The planning would then construct the rules of its world representation online and enable the system to decide in arbitrary situations.

References

[1] R. Bellman, "A Markovian decision process," *Journal of Applied Mathematics and Mechanics*, vol. 6, pp. 679–684, 1957.

[2] T. Gindele, S. Brechtel, and R. Dillmann, "A Probabilistic Model for Estimating Driver Behaviors and Vehicle Trajectories in Traffic Environments," in *IEEE Int. Conf. on Intelligent Transportation Systems*, Madeira Island, Portugal, September 2010.

[3] J. Schröder, M. Hoffmann, M. Zöllner, and R. Dillmann, "Behavior Decision and Path Planning for Cognitive Vehicles using Behavior Networks," in *IEEE Int. Conf. on Intelligent Vehicles Symposium*, Istanbul, Turkey, 2007, pp. 710–715.

[4] T. Gindele, D. Jagszent, B. Pitzer, and R. Dillmann, "Design of the planner of Team AnnieWAY's autonomous vehicle used in the DARPA Urban Challenge 2007," in *IEEE Intelligent Vehicles Symposium*, Eindhoven, The Netherlands, 2008, pp. 1131–1136.

[5] D. Pomerleau, "Defense and Civilian Applications of the ALVINN Robot Driving System," in *Government Microcircuit Applications Conf.*, November 1994, pp. 358–362.

[6] J. Forbes, T. Huang, K. Kanazawa, and S. Russell, "The batmobile: Towards a bayesian automated taxi," in *Int. Joint Conf. on Artificial Intelligence*, Montreal, Quebec, Canada, 1995, pp. 1878–1885.

[7] L. Kaelbling, M. Littman, and A. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, no. 237-285, pp. 102–138, 1996.

[8] D. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific, 2007, vol. 2.

[9] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," *Int. Joint Conf. on Artificial Intelligence*, vol. 18, pp. 1025–1032, 2003.

[10] R. Munos and A. Moore, "Variable resolution discretization in optimal control," *Machine learning*, vol. 49, no. 2, pp. 291–323, 2002.

[11] A. Brooks, A. Makarenko, S. Williams, and H. Durrant-Whyte, "Parametric POMDPs for planning in continuous state spaces," *Robotics and Autonomous Systems*, vol. 54, no. 11, pp. 887–897, 2006.

[12] J. Porta, N. Vlassis, M. Spaan, and P. Poupart, "Point-based value iteration for continuous POMDPs," *The Journal of Machine Learning Research*, vol. 7, pp. 2329–2367, 2006.

[13] R. A. Howard, *Dynamic Probabilistic Systems: Semi-Markov and Decision Processes*. NY: Wiley, 1971.

[14] S. Brechtel, T. Gindele, and R. Dillmann, "Recursive Importance Sampling for Efficient Grid-Based Occupancy Filtering in Dynamic Environments," in *IEEE Int. Conf. on Robotics and Automation*, Anchorage, AL, USA, 2010.

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

[16] J. Hoey and P. Poupart, "Solving POMDPs with continuous or large discrete observation spaces," in *Int. Joint Conf. on Artificial Intelligence*, vol. 19, 2005, p. 1332.