Hybrid Online POMDP Planning and Deep Reinforcement Learning for Safer Self-Driving Cars

Florian Pusse[1] and Matthias Klusch[2]

***Abstract*— The problem of pedestrian collision-free navigation of self-driving cars modeled as a partially observable Markov decision process can be solved with either deep reinforcement learning or approximate POMDP planning. However, it is not known whether some hybrid approach that combines advantages of these fundamentally different solution categories could be superior to them in this context. This paper presents the first hybrid solution HyLEAP for collision-free navigation of self-driving cars together with a comparative experimental performance evaluation over the first benchmark OpenDS-CTS of simulated car-pedestrian accident scenarios based on the major German in-depth road accident study GIDAS. Our experiments revealed that HyLEAP can outperform each of its integrated state of the art methods for approximate POMDP planning and deep reinforcement learning in most GIDAS accident scenarios regarding safety, while they appear to be equally competitive regarding smoothness of driving and time to goal on average.**

I. INTRODUCTION

The accomplishment of pedestrian safety by autonomous or self-driving vehicles is considered paramount to their acceptance and economic success. In general, collision-free navigation refers to solving the constrained optimization problem where the time to reach a given goal is minimized subject to collision avoidance constraints for driving on computed path, i.e. the car does not crash into any pedestrian or other obstacle like a parking car on the lane. Since the environment of traffic scenes is usually considered to be not fully observable by the car, this problem is often modeled as a partially observable Markov decision process (POMDP) and solved, in general, by either approximate POMDP planning (APPL) or deep reinforcement learning (DRL).

In fact, there is a plethora of related work on collision-free navigation in robotics available since decades with recently renewed interest on the problem in the domain of autonomous driving [23], [20]. APPL-based solutions such as the online approximate POMDP planner IS-DESPOT [15] leverage their explicit world model to plan their actions into the future with an n-step lookahead, but do not learn from any experience. Experiments in [1] showed the applicability of APPL for real-time collision-free navigation on campus. On the other hand, DRL methods like NavA3C [17], [18], UNREAL [11], IntentionNet [9], SA-CADRL [6] and Fast-RDPG [25] are able to learn and generalize to unseen situations from experience offline but without explicit, task-oriented action planning into the future. Both types of approximate POMDP solving may show significantly different behaviors. For example, an APPL-based self-driving car could deduce future collisions with pedestrians based on its world model and start braking without having to see a similar traffic scene before, while DRL-based solutions will have to learn a situatively optimal policy offline and to generalize from situations they encountered in the past. Unlike APPL, though, they can learn from their mistakes and try to avoid repeating them in the future. However, it is not yet known, which of both types of approximate POMDP solving performs best for collision-free navigation of self-driving cars, and how to effectively and efficiently combine them for this purpose in general.

To this end, we developed the first hybrid solution, named HyLEAP, which combines advantages of selected state of the art methods for DRL and APPL, that are NavA3C[17], respectively, IS-DESPOT[15], for goal-directed collision-free navigation, and discuss the experimental evaluation results for simulated single and multiple pedestrian-car accident scenarios based on the German in-depth road accident study (GIDAS). All sources and benchmark of this work are publicly available [22].

The remainder of this paper is structured as follows. The considered problem of pedestrian collision avoidance by autonomous cars is defined in section 2, and our hybrid solution HyLEAP is described and discussed in section 3. This is followed by the comparative experimental performance evaluation in section 4, while related work is summarized in section 5 before we conclude in section 6.

II. PROBLEM DESCRIPTION

The pedestrian collision avoidance problem of a self-driving car is, in short, to minimize the time to a given goal subject to constraints with specific focus on avoiding near misses or even hits of pedestrians. This problem can be modeled as a discrete-time, partially observable Markov decision process (POMDP) to be solved by the car online in real-time. Before defining the POMDP, we shortly describe the underlying model of the car and pedestrian, and the considered critical traffic scenarios for the problem.

A. Prerequisites

1) Car and pedestrian model: Regarding car perception, we assume the car to have a 360° surround view of its environment, where pedestrians within a maximum viewing

[1] Florian Pusse is with the Computer Science Department, Saarland University, 66123 Saarbruecken, Germany

[3] Matthias Klusch is with the German Research Center for Artificial Intelligence (DFKI), 66123 Saarbruecken, Germany matthias.klusch@dfki.de

distance of 50 meters are observable, if no other obstacle occludes them. As in [1], only the exact positions of observable pedestrians are assumed to be known to the car. Moreover, we adopted the kinematic bicycle model in [12], which appeared appropriate for approximate modeling of the driving behavior of cars in our context [21]. The state of a car c at time t is defined as $[p, v, \theta]_t^c$, where $p_t^c = (p_x, p_y)_t^c \in \mathbb{R}^2$ denotes the position of c, $v_t^c \in \mathbb{R}$ its speed, and $\theta_t^c \in [0, 2\pi)$ its orientation at time t.

For goal-directed navigation learning and planning of the car in continuous space of POMDPs, we employ the hybrid A* path planner [24] as in [1], [9] to make sure the generated paths are driveable according to the car kinematics, though paths are not guaranteed to be optimal and complete. Using the path planner requires a costmap for which the 3D-model of the traffic scene in OpenDS is scanned for obstacles in a discretized grid into a graycale map and with standard obstacle costs of car states defined as maximum of 1, 50, and 100, if the car is on the road, partly on the sidewalk, with any part colliding with an obstacle, respectively, or infinite else. Car intentions are snippets (84x84x3 RGB images) of the cost map centered at the position of the car with its path in the past and to be driven in the future drawn. The idea of using car intentions for learning and planning is to guide the autonomous vehicle in the right direction using the information of the path planner as done in [9].

Similar to the car state, the pedestrian state $[p, v, \theta, g]_t^{ped}$ of pedestrian ped of a set $\mathcal{P}$ of pedestrians in the environment at time t is defined through its position $p_t^{ped} = (p_x, p_y)_t^{ped} \in \mathbb{R}^2$, speed $v_t^{ped} \in \mathbb{R}$ in m/s, orientation $\theta_t^{ped} \in [0, 2\pi)$, and goal $g_t^{ped} = (g_x, g_y)_t^{ped} \in \mathbb{R}^2$. Pedestrians only move into the direction of the goal in a straight line. Accident areas for collisions between car and pedestrian are defined via rectangles with safety margins (1.5m front, 0.5 back/side) for near-miss and hit (cf. Fig. 1) with respectively negative rewards for the car (cf. Sect. 2.B).

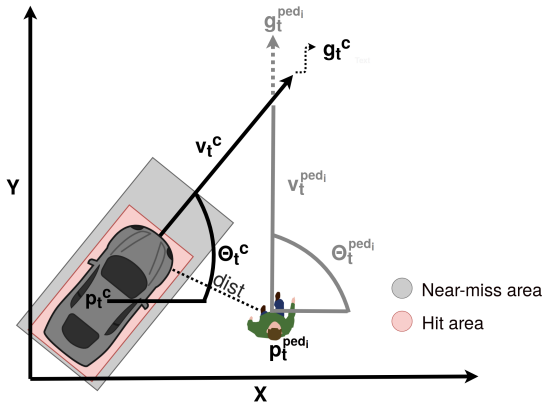


Fig. 1. Accident (near-miss or hit) areas for car-pedestrian collisions

2) Simulated car-pedestrian accident scenarios: For comparative evaluation of pedestrian collision avoidance methods of self-driving cars in accident scenarios with pedestrians (cf. Section 4), we created the first benchmark OpenDS-CTS 1.0 [22] based on the GIDAS (German In-Depth Accident Study) analysis of several thousands of such accidents that happened in Germany in the past [3]. In particular, OpenDS-CTS includes tens of thousands of scenes as instances of the most likely GIDAS car-pedestrian accident scenarios in which a pedestrian crosses the street in front of a car as shown in Fig. 2; all accident scenes are virtually simulated with the open-source 3D driving simulator OpenDS[1] on a test drive (of about 100 meters) in a virtual 3D version of Saarbruecken city. Each scene of a GIDAS accident scenario is denoted as $(scenario, p_{start}^c, \theta_{start}^c, p_{goal}^c, \mathcal{P})$.

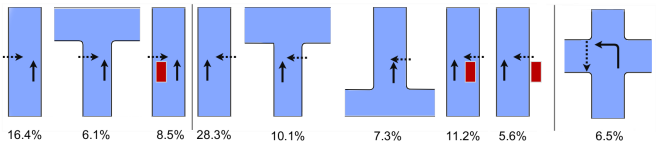


Fig. 2. Accident scenarios between car and pedestrian on German roads according to GIDAS analysis with percentages of their occurence. Dotted line denotes pedestrian movement, solid line that of the car, and red square represents a static obstacle which blocks the sight of car.

B. Collision-free navigation problem

We modeled the pedestrian collision avoidance problem of a car in a simulated GIDAS accident scene of the test drive as a POMDP $(S, A, T, R, \gamma, Z, O)$ as follows:

- S: Set $\{[c, ped_1, ..., ped_{|\mathcal{P}|}] \mid c \in \mathbb{R}^2 \times \mathbb{R} \times [0, 2\pi),\ ped_i \in \mathbb{R}^2 \times \mathbb{R} \times [0, 2\pi) \times \mathbb{R}^2, 1 \leq i \leq |\mathcal{P}|\}$ of states $s_t \in S$ at time t with car state c and pedestrian states $ped_i = [ped_i^o, ped_i^h] \in \mathcal{P}$, $1 \leq i \leq |\mathcal{P}|$, where ped_i^o denotes the observable position of pedestrian i
- A: Set $\{(\alpha, acc)\}$ of car driving actions $a \in A$ with steering angle $\alpha \in \mathbb{Z}, |\alpha| \leq \alpha_{max} \wedge \alpha \bmod 5 = 0$, $\alpha_{max} = 50^o$ in both directions, and step-wise acceleration $acc \in \{Accelerate, Maintain, Decelerate\}$ with $+5, 0, -5 km/h$.
- T: Transition probability $T(s_t, a_t, s_{t+1}) = p(s_{t+1}|s_t, a_t) \in [0, 1]$ of transitioning from state $s_t \in S$ into state $s_{t+1} \in S$ when executing action $a_t \in A$ at time t. State changes fully defined by car movement kinematics, pedestrian model.
- Z: Set $\{[c, ped_1^o, ..., ped_{|\mathcal{P}|}^o]\}$ of observations $o_t \in Z$ with car state and observable part of perceived pedestrians in the scene.
- O: Observation probability $O(s_{t+1}, a_t, o_{t+1}) = p(o_{t+1}|s_{t+1}, a_t) \in [0, 1]$ of observing $o_{t+1} \in Z$ when transitioning into s_{t+1} after executing a_t, and $O(s_{t+1}, a_t, o_{t+1}) = 1$, if $o_{t+1} = [c, ped'^o_1, ..., ped'^o_{|\mathcal{P}'|}]$ for pedestrians $\mathcal{P}' \subseteq \mathcal{P}$ currently perceived by car, else 0; states are not inferrable from single observations.
- R: Immediate reward $R(s_t, a_t) \in \mathbb{R}$ for executing action a_t in state s_t is:
  $r_{goal} = +1000$, if goal position reached; r_{near_miss} -500, if pedestrian in near-miss area of car (rectangular

[1]OpenDS: https://opends.dfki.de

area around car; includes smaller hit area) and $v_t^c > 0$; $r_{hit} = -1000$, if pedestrian in hit area and $v_t^c > 0$; $r_{obst} = -max\{obstCosts\}$, $obstCost$ defined in Sect. 2.A; $r_{acc} = -0.1$, if ac-/deceleration; $r_{steer} = -1$, if steering: $|\alpha_t| > 0$; $r_{notgoal} = -0.1$ else.

γ: Discount factor in $[0, 1]$, we set $\gamma = 0.98$ similar to [1].

The reward function R encourages safe, fast, and smooth driving. Its component rewards r_{goal} and $r_{notgoal}$ encourage the car to reach the goal, and r_{near_miss} to keep a safety distance to perceived pedestrians, while r_{hit}, r_{obst} penalize crashing into a pedestrian or other obstacles. Small penalties r_{acc} and r_{steer} intend to avoid unnecessary acceleration and steering which decrease the smoothness of driving.

III. HYBRID SOLUTION HyLEAP

A. Overview

The HyLEAP method integrates the online POMDP planner IS-DESPOT [15] and the deep reinforcement learner NavA3C [17] for approximated solving of the collision-free navigation problem of self-driving cars as modeled in Sect. 2.B. In particular, HyLEAP combines the advantages of both approaches by online approximated POMDP planning that exploits the integrated DRL network as its critic. The overall HyLEAP architecture coupled with the OpenDS driving simulator is shown in Fig. 3. In each GIDAS accident scene simulation, HyLEAP gets trained in two consecutive phases. First, IS-DESPOT determines its (APPL) action policy using the coupled NavA3C network with fixed, initially random weights in a feed-forward manner only for evaluating its belief tree construction at each time step of scene simulation, and executes the selected actions in the scene. After simulation of the scene, the NavA3C network then gets trained to update its weights during its (DRL) action policy computation according to the received total discounted reward, such that its experience-based evaluation feedback to IS-DESPOT in the next scene simulations can improve. In this respect, the network acts as an experience-based critic of the acting online planner for simulated critical traffic scenes.

B. HyLEAP NavA3C Neural Network

The DRL network of HyLEAP uses the NavA3C architecture with only one instead of originally two chained LSTM layers for faster execution (cf. Fig. 4). The observation input from IS-DESPOT is the car intention RGB image (3x84x84 pixel), which is passed through two convolutional network layers conv1 (16x8x8 with stride 4x4) and conv2 (32x4x4 with stride 2x2). The output of conv2 is then fed into a fully connected layer, which result is concatenated with the last received reward $r_{t-1} \in \mathbb{R}$, current car velocity $v_t \in \mathbb{R}$, and last executed action $a_{t-1} \in \{0, 1\}^{|A|}$ by IS-DESPOT in the simulated scene, and then fed into the LSTM network layer with forget gates [10] together with the history h_{t-1}. The output of the LSTM layer is reduced to |A| outputs for the DRL action policy $\pi_t^{DRL} \in [0, 1]^{|A|}$ and to size one for the estimated value $V_t(h_{t-1}, o_t) \in \mathbb{R}$ based on its weights using a fully connected layer each. All layers use ReLUs except for the identity in the output layer.

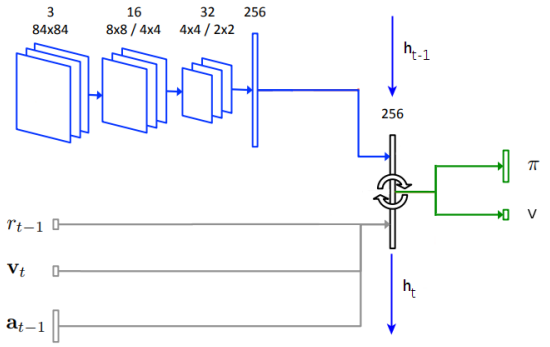


Fig. 4. HyLEAP NavA3C neural network architecture

The network $f_\theta(h_{t-1}, o_t) = (\pi_t^{DRL}, V_t)$ takes the current history h_{t-1} and the latest observation $o_t = (car\_intention_t \in [0, 1]^{84,84,3}, r_{t-1} \in \mathbb{R}, V_t \in \mathbb{R}, a_{t-1} \in \{0, 1\}^{|A|})$ at time t as an input, and then outputs the DRL action policy $\pi_t^{DRL} \in [0, 1]^{|A|}$ and the estimated value $V_t \in \mathbb{R}$ based on its weights θ. The computed DRL policy is not executed in the scene but used inside IS-DESPOT to guide its belief tree construction. As mentioned above, the network is trained to minimize (a) the error between the received total discounted reward R_t and the predicted value V_t, and (b) the error between the neural network policy π_t^{DRL} and the APPL policy of IS-DESPOT π_t^{APPL} for each scene simulation point in time $0 \leq t < T$ individually. The gradients of the loss function L_H with respect to the weights are calculated for each t sequentially and summed up, and after all gradients are accumulated, the network weights are updated using gradient descent. The loss function $L_H : \mathbb{R} \times \mathbb{R} \times [0, 1]^{|A|} \times [0, 1]^{|A|} \rightarrow \mathbb{R}_+$ is defined as $L_H(R_t, V_t, \pi_t^{APPL}, \pi_t^{DRL}) = (R_t - V_t)^2 - (\pi_t^{APPL})^\top log \pi_t^{DRL} + \lambda \parallel \theta \parallel^2$. This loss function aims at minimizing the mean squared error between the estimated value and the real value based on the total discounted reward R_t received, and the cross-entropy loss between the DRL policy and the APPL policy, with L2-regularization of the neural network weights using a regularization constant λ.

C. Integrated Planning and Learning

The main algorithm of HyLEAP training as outlined in Section 3.A is shown below in Alg. 1. Executing the trained HyLEAP car in a simulated GIDAS accident test scene is one run-through of its IS-DESPOT-p with integrated evaluation by the trained NavA3C network with input from the path planner and the driving simulator OpenDS. Training of HyLEAP is performed in two phases per traffic scene simulation in OpenDS as follows.

Phase 1: Network-guided action planning in traffic scene simulation. The online POMDP planner DESPOT [27] is an anytime heuristic search that uses upper and lower bounds on belief values to guide the construction of an approximated belief tree for computing an APPL action policy for POMDP

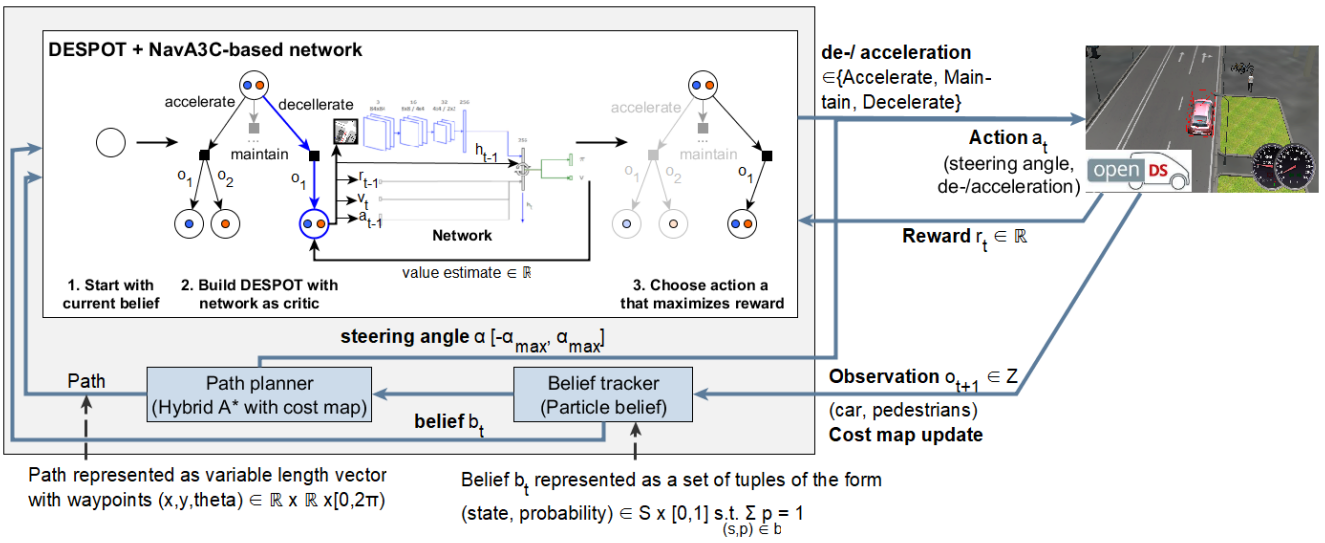
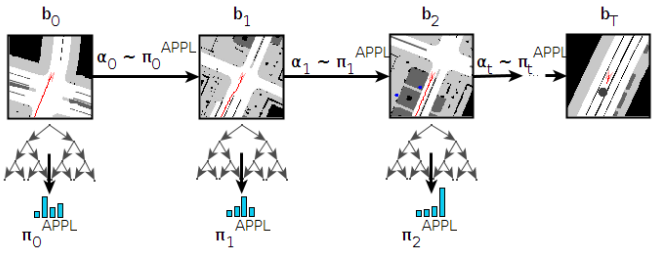


Fig. 3. Overview of HyLEAP architecture with OpenDS driving simulator

Algorithm 1 HyLEAP Training

episodes: $M \in \mathbb{N}_+$
steps per episode: $T_{max} \in \mathbb{N}_+$
particles for particle belief: $P \in \mathbb{N}_+$
Regularization parameter $\lambda \in [0, 1]$
Init neural network $(\pi^{DRL}, v) = f_\theta(h, o)$ with random weights θ
For episode $\leftarrow$ 1 to M **do**:
$t \leftarrow 1$
Init belief b_1 with P samples
Init history $h_0 = 0$
Get initial observation o_1
Repeat:
$angle_t \leftarrow PathPlanner(o_t)$
$D, h_t \leftarrow BuildDespot(b_t, h_{t-1})$
Sample $acceleration_t \sim \pi_i^{APPL}$
$a_t \leftarrow (angle_t, acceleration_t)$
$o_{t+1} \leftarrow OpenDS\_Step(a_t)$
$b_{t+1} \leftarrow UpdateBel(b_t, a_t, o_{t+1})$
$t \leftarrow t + 1$
Until $o_t == terminal$ or $t > T_{max}$
$-, v = f_\theta(h_{t-1}, o_t)$
R = $\begin{cases} 0, & \text{for terminal } o_t \\ v, & \text{for non-terminal } o_t \end{cases}$
Reset gradients $d\theta \leftarrow 0$
For $i \leftarrow t-1$ to 1 **do**:
$R \leftarrow r_i + \gamma R$
$\pi_i^{DRL}, v_i = f_\theta(h_{i-1}, o_i)$
Accumulate gradients wrt θ:
$d\theta \leftarrow d\theta + \nabla_\theta L_H$
End-For
Perform update of θ using $d\theta$
End-For

solving. One major problem of DESPOT's sampling-based approach is that outcomes that have a low probability of occuring, e.g. crashes, are not sampled, which may lead to decreased performance. In [15], this problem is addressed with importance sampling in the online approximate POMDP planner IS-DESPOT, which we adopted for HyLEAP. At each point in time t of one traffic scene simulation in OpenDS, IS-DESPOT constructs a belief tree with root b_t using the neural network as guidance (cf. Fig. 5), and then outputs a policy π_t^{APPL} from which an action is sampled for execution. This process is repeated until the scene simulation ends, i.e. a terminal state is reached at time $t = T$. During the whole scene execution, the NavA3C network remains unchanged but is used to guide the construction of the belief tree. This guidance is achieved by the network through its evaluation of the newly created belief nodes b' from expanding a selected leaf node with the estimated value V_t, which, in turn, is taken by IS-DESPOT as an estimation of the initial upper bound $U(b')$ of the belief value required for the belief node updates (backup) along the path back to the root of the tree. The additionally required initial lower bound is heuristically computed as in [15], i.e. the minimum number of steps until a car-pedestrian collision occurs as outcome. Finally, the new policy tree with π_t^{APPL} is derived from which the action (at root) with the highest lower bound is selected for execution in the scene.



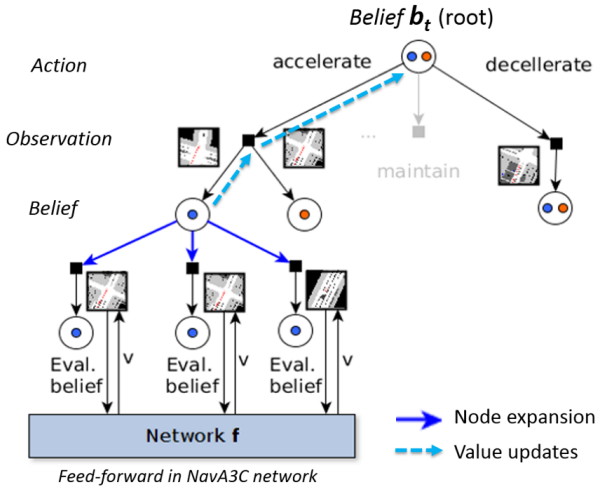


Fig. 5. Network-guided belief tree construction by IS-DESPOT at each simulation step t = 0..T of a traffic scene simulated in OpenDS

Phase 2: HyLEAP NavA3C network training. After simulation of the considered traffic scene ends, the NavA3C

network gets trained for this episode to improve on its evaluation of the APPL policy created by IS-DESPOT according to the total discounted rewards received. Ideally, the provided estimation of expected upper bound of the belief value that can be achieved by IS-DESPOT for potential outcomes according to the reward function R (cf. Sect. 2.B) are tighter than the heuristic in [1]. Our experiments revealed that this indeed is the case in most of the simulated GIDAS accident scenarios.

The polynomial runtime complexity of HyLEAP is $O(M * (T_{max} * (c_{pathPlanning} + c_{DESPOT} + c_{beliefUpdate}) + T_{max}c_{network}))$, where M denotes the number of episodes, T_{max} the maximum length of an episode, $c_{pathPlanning} = O(|V|^2)$ the cost to plan a path using hybrid A*, $c_{network}$ the cost of network updates, and $c_{DESPOT} = O(N(D+K)(D+ c_{network} + |A| + K))$ the cost of DESPOT construction.

IV. EVALUATION

A. Experimental Setting

The used benchmark OpenDS-CTS 1.0 consists of about 38000 accident scenes simulated in OpenDS with 9 different single pedestrian-car accident scenarios (about 4000 scenes for each) according to the GIDAS report. These accident scenarios are enumerated as follows (cf. Fig. 2): The pedestrian crosses the street in front of an approaching car from (1,2) the left without occlusion by obstacle; (4,5,6) the right without occlusion by obstacle; (3) the left with occlusion by parking vehicle on the road; (7) the right with occlusion by obstacle on sidewalk; (8) the right with occlusion by parking vehicle on the road; (9) the right after car turned left at an intersection. Each method is trained over all these GIDAS accident scenarios each of which instantiated with about 1200 scenes with varying pedestrian speeds (0.6-2m/s in 0.1 steps) and crossing distances (0.1-40m in 0.5m steps), and then tested in about 2000 previously unseen critical scenes of each scenario with different pedestrian speeds (0.25-2.85 m/s in 0.1 steps), crossing distances (4.25-49.75m in 1m steps) and unsual movement patterns on the sidewalk (ZigZag lines with varying angles of 30° - 160° in 10° steps, and 0.5-5m in 0.5m steps). For each scenario, the configuration of varying crossing distances (4.25m-49.75m) and pedestrian movement speeds (0.25m/s-2.85m/s) are tested 10 times each (10 runs per scene). Pedestrian position updates and car intentions are provided to the selected navigation learning and planning methods at each simulation time step $t = 0..T$ by OpenDS. The evaluation metrics are defined for pedestrian safety as the number of hits (crashes), time-to-goal (TTG) in seconds, and smoothness of driving as the number of acc-/decelerations on the used GIDAS scenario test drive in the simulated 3D version of Saarbruecken city by OpenDS. The basic reactive controller is implemented as in [1]. In each simulated scene, the car starts on the test drive with 0km/h and can maximally accelerate to the German city speed limit of 50km/h.

As state-of-the-art approaches for DRL-based and APPL-based navigation learning and planning methods, we selected NavA3C-p [18] and IS-DESPOT-p [1], respectively, each extended with path planner hybrid A* and coupled with OpenDS (cf. Sect. 3). NavA3C-p was trained on the supercomputer NVIDIA DGX-1 at DFKI in 1 million episodes taking 3 days in total with gradients computation intervals of 40 steps, decreasing learning rate (0.0003 to 0.0005) and entropy regularizer β (0.001 to 0.0008), weight optimization with RMSprop (decay 0.99, momentum 0, epsilon 0.1) and gradient clipping to size 40. The hybrid method HyLEAP was trained in 40000 episodes taking 7 days in total with learning rate 0.00015, L2 regularization (λ 0.0005) and optimization with RMSprop as for NavA3C-p. Testing of all methods was conducted on an Ubuntu 18 PC with a Nvidia GTX 1080Ti, an Intel i7-7820X CPU @ 3.60GHz, and 32GB RAM.

B. Results

As shown in Table I, the experimental results revealed that HyLEAP can outperform IS-DESPOT-p and NavA3C-p in most of the GIDAS accident scenarios on the simulated test drive regarding safety (GIDAS safe). Recorded crashes, impact speed in km/h, near-misses, smoothness, time to goal, and execution times are averaged over all test scenes.

In particular, HyLEAP was the safest self-driving method for simulated cars in accident scenarios 4, 5, 6, 8 and 9, while IS-DESPOT-p was safest in scenarios 1, 2, and 7, and NavA3C-p was superior to the other methods only in scenario 3. Averaged over all tested scenes, both HyLEAP and IS-Despot-p appear equally competitive regarding safety but with an advantage of IS-DESPOT-p in the avoidance of near-misses of pedestrians. Regarding the comfort or smoothness of driving, HyLEAP and NavA3C-p turned out to be similar, while IS-DESPOT-p performed only slightly worse in this respect. Though the basic reactive controller led to some safer driving than NavA3C-p and less near-misses than even HyLEAP in test scenes on average, it did not outperform the other methods in any GIDAS accident type with similarly competitive comfort or smoothness of driving and time to goal.

Behavior in GIDAS car-pedestrian accident scenarios. The behavior of tested cars differed depending on crossing distance and speed of a pedestrian (cf. Fig. 6). In general, the NavA3C-p car had severe problems in almost every tested GIDAS accident scenarios to properly generalize, hence could not react fast enough to previously unseen situations in which pedestrians were crossing either fast (2-3m/s) and close to (less than 10m), or slowly (0.2-0.5m/s) and far away (>30m) from the car. In the latter case, the car stopped correctly but often re-accelerated too early leading to crashes or near-misses. On the other hand, NavA3C-p was safer than the other methods in scenes with a parking vehicle on the road that occludes the pedestrian crossing from the left who it learned to evade during training.

The IS-DESPOT-p car can react online on any pedestrian speed using its model, though, similar to the results for the NavA3C-p car, fast moving pedestrians with short crossing distances caused crashes and near-misses due to its slow

	GIDAS safe	Crashes (%)	Impact speed	Near-misses (%)	# De-/Acc.	TTG (s)	Exec. (s)
HyLEAP	**5**	3.01	14.27	8.19	22.91	13.23	0.28
IS-DESPOT-p	3	2.95	16.44	6.22	24.90	13.66	0.27
NavA3C-p	1	3.88	19.18	8.27	22.59	13.56	0.01
React. contr.	0	3.29	5.35	6.86	27.59	15.17	0.001

TABLE I

OVERALL RESULTS OF SAFE DRIVING EVALUATION

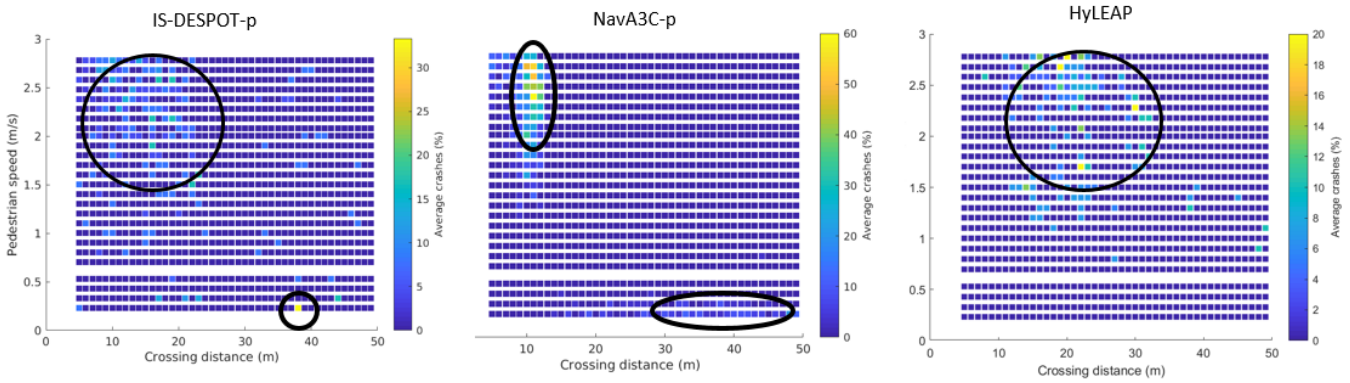


Fig. 6. Distribution of accident rates of IS-DESPOT-p, NavA3C-p for varying pedestrian speeds (m/s) [y-axis] and crossing distances (m) [x-axis] averaged over test scenes of all GIDAS accident scenarios.

shifting of beliefs with insufficient reaction time. If the belief is not shifted yet, the car can still slow down even if the planned path with hybrid A* does not keep enough distance from the pedestrian yet. In several scenes of slowly moving pedestrians (<0.8 m/s) with enough distance (about 35m) to brake in time, IS-DESPOT-p does not sample potential crash outcomes and the car maintains its speed until it is too late to brake. Besides, in scenes with highly frequent changes of pedestrian intentions, such as fast ZigZag movement, IS-DESPOT-p cannot make its belief updates fast enough to keep track with the situation, leading to indecisive behavior on whether to slow down, or to stop the car, or to evade the pedestrian. As IS-DESPOT-p does not learn from experience over time it is inherently prone to make the same failures again.

Like IS-DESPOT-p, HyLEAP, in general, suffered from the time taken by its planning component for the construction and update of belief nodes of the belief tree. But since its integratively trained NavA3C neural network learned to act as an experience-based critic of the action policy created by its planner IS-DESPOT, HyLEAP could benefit from this actor-critic-like interplay between online planning and learning in more GIDAS accident scenarios than all other methods. Compared to NavA3C-p and IS-DESPOT-p, HyLEAP avoided crashes with pedestrians walking at a lower speed and low crossing distances than seen during training, but in some scenes with slow pedestrians (<0.5m/s) and distances between 10 to 20m it was indecisive on whether to overtake or to brake, such that compared to IS-DESPOT-p its scene averaged crash rate reduced only slightly on average.

On the other hand, HyLEAP was safer than both NavA3C-p and IS-DESPOT-p in most GIDAS accident scenarios. For example, table II shows averaged results for the critical scenario 9 where the car turns left into a street at an intersection and a pedestrian crosses this street from the right with varying crossing distances ($4.25m - 49.75m$) and pedestrian speeds ($0.25m/s - 2.85m/s$). The IS-DESPOT-p slowed the car down while turning left at an intersection but then could not brake fast enough for the pedestrian due to its slow belief updating, hence killed the pedestrian. The car with NavA3C-p failed to generalize to the scene during training possibly due to the rotation of the car intention for and not recognizing the pedestrian moving in its direction after turning left, hence crashed into the pedestrian without slowing down. Only the HyLEAP-based car by combining experience-based learning and look-ahead planning properly slowed down when turning left and evaded the pedestrian in most scenes of this scenario (cf. Table II).

Unusual pedestrian movement patterns. Furthermore, we tested the behavior of the self-driving methods in situations where the pedestrian is walking in a ZigZag line on the sidewalk. Since the car with NavA3C-p did not encounter such pedestrian behavior during training and the observed directions frequently change, it becomes difficult for this method to estimate the pedestrian intention for decision-making. However, NavA3C-p learned to generalize to such new situations during training and, unlike IS-DESPOT-p, does not need to shift its belief online, hence reacted directly on perceived movement changes of and overtook the pedestrian fast and safely (Avg. De-/Accel. (#): 16, Avg. Time-to-goal (s): 12). The car with IS-DESPOT-p failed to overtake the erraticly moving pedestrian, since the required shift of probability mass by IS-DESPOT-p online is too slow; until the belief is adapted to the observed situation for possible

	Crashes (%)	Impact speed	Near-misses (%)	# De-/Accel.	TTG (s)
HyLEAP	**4.34**	20.58	8.29	14.69	**10.41**
IS-DESPOT-p	5.64	22.38	8.83	20.61	10.69
NavA3C-p	10.87	33.17	19.67	13.55	19.79

TABLE II

AVERAGED RESULTS FOR GIDAS ACCIDENT SCENARIO 9 (CAR TURNS LEFT AT INTERSECTION)

actions the pedestrian's direction changed again, such that the car just followed but did not overtake the pedestrian until he disappeared at the end ot the test drive (Avg. De-/Accel. (#): 20, Avg. Time-to-goal (s): 12.49). Although HyLEAP using IS-DESPOT also shifts the belief, it keeps a distance to the pedestrian and overtakes it straightforward, because the APPL policy evaluating neural network considered the approaching of a pedestrian as not critical enough to come to a stop or decelerate. In fact, HyLEAP executed an experience-based overtaking action by its online planner which was fastest and smoothest among all other methods (Avg. De-/Accel. (#): 14, Avg. Time-to-goal (s): 11.75).

Multiple pedestrians. The navigation behavior of the different methods was also tested for scenarios (4800 scenes) with multiple, initially two to four, pedestrians crossing the street from both sides with walking speed (1.81m/s), varying distance (0.5-25m) to car, and delay of crossing the street (0-4s). The NavA3C-p and HyLEAP cars were trained on 1440 scenes with two pedestrians crossing from the right sidewalk only. In summary, with increasing number of pedestrians, the NavA3C-p car struggled to appropriately generalize, hence did hit the second pedestrian from the left most of the time, while the IS-DESPOT-p car was significantly safer, smoother in driving and faster in reaching the goal. The HyLEAP car outperformed each of them overall (cf. Table III).

V. RELATED WORK

As mentioned before, there are various approaches for goal-directed navigation of self-driving vehicles with approximate, near-optimal action policy finding in fully or partially observable environments. While some of them focus on pedestrian-collision avoidance, others do not consider pedestrians at all.

Approximate POMDP navigation planning. [2] present a mixed observable MDP that treats pedestrian intentions as hidden variables, while the state of the predestrians is assumed to be fully observable, and the offline policy for navigation actions is generated by SARSOP[13]. The approach suffered from performance problems with multiple pedestrians present in a scene, which is why the authors solved then a discrete POMDP for each pedestrian. [1] improved upon [2] by using DESPOT to solve the overall, single POMDP, keeping a joint belief over all pedestrians, which yielded comparatively better performance. [5] present a method for autonomous navigation in car merging scenarios using continuous-state POMDP for all road users, which are solved offline based on value iteration [4]. [14] propose a method for collision avoidance in an urban road context, where the car needs to avoid collisions with other (obstacle) vehicles on the same road and the environment is mapped with an empricially derived road context. Other cars are assigned motion intentions such as stopping, i.e. letting the ego car pass, or being aggressive, i.e. not letting anyone pass. The problem is modeled as a discrete POMDP and solved online using DESPOT [27].

DRL-based navigation learning. Applications of DRL for collision-free navigation range from navigation in simple grid worlds to driving autonomous robots in office environments. While some of the existing approaches do not explicitly consider other agents, e.g. pedestrians, in the environment some are explicitly created for that purpose. Similarly, some methods assume the environment to be wholly unknown and some assume to have perfect knowledge of the surroundings. For example, [17] and [11] improved the vanilla, asynchronous advantage actor-critic (A3C) DRL method in [18] in versions NavA3C and UNREAL, respectively, for learning to navigate in a completely unknown, virtual 3D maze with unknown goal locations directly from pixel values based on observations and rewards only. A recent study [8] argued that NavA3C in the maze is unable to generalize to new, unseen environments, meaning that the agent is not really able to learn to navigate but only to learn some simple wall-following strategy. The approach in [25] used Fast-RDPG for navigation of an unmanned aerial vehicle. Similar to NavA3C and UNREAL, Fast-RDPG does not use any sort of map for navigation, hence could potentially suffer from the same problems as described before. Though all previously mentioned approaches use LSTM layers to incorporate histories, there exist alternate approaches that utilize external memory, such as neural maps [19] or neural SLAM [28], to internally represent the map, which should force the agent to perform simultaneous localization and mapping (SLAM). IntentionNet [9] combines path planning with hybrid A* to generate the robot intention image and deep inverse reinforcement learning that can be used if at least a rough (street) map is available. It is then up to the neural network to decide on the control of the robot using the robot intention as a guide. Pedestrians are only considered implicitly by learning from examples. In [7] the collision-free navigation problem as a constrained optimization problem is turned into an MDP, which is then solved with a trained deep value network (DVN) but under the assumption, like in the improved version [6], that the maximum number of pedestrians for all possible scenes is known in prior in order to be able fix the input layer size of their neural network.

However, to the best of our knowledge, there is no integrated APPL- and DRL-based approach for approximate POMDP

	Crashes (%)	Impact speed	Near-misses (%)	# De-/Accel.	TTG (s)
HyLEAP	6.32	10.99	10.84	18.74	13.06
IS-DESPOT-p	6.58	11.27	12.01	20.10	13.32
NavA3C-p	15.17	14.84	24.40	26.32	14.59
React. Contr.	10.92	9.83	28.43	47.37	19.54

TABLE III

AVERAGED RESULTS FOR SCENARIO WITH FOUR PEDESTRIANS CROSSING FROM BOTH SIDES OF STREET

solving for collision-free navigation in road traffic. Besides, none of the current approaches were evaluated in synthesized car-pedestrian accident scenarios based on real-world accident studies like GIDAS.

VI. CONCLUSIONS

We presented the first hybrid solution for pedestrian collision-free navigation of self-driving cars in simulated critical traffic scenarios that combines selected methods of approximated POMDP planning and deep reinforcement learning. Our initial, comparative performance evaluation over the OpenDS-CTS benchmark based on the German in-depth road accident study GIDAS provided first, valuable insights into the behavior of each method applied to this problem and revealed that our hybrid solution HyLEAP is superior to its integrated individual methods regarding GIDAS pedestrian safety, though for each method there are scenarios in which it was superior to all others. Ongoing work is concerned with OpenDS-CTS benchmark extension and alternative hybrid combination of learning and planning in this context.

ACKNOWLEDGMENT

This research was supported by the German Federal Ministry for Education and Research (BMB+F) in the project REACT.

REFERENCES

[1] H. Bai, S. Cai, N. Ye, D. Hsu, and WS Lee, Intention-Aware Online POMDP Planning for Autonomous Driving in a Crowd. In: Proc. of Intl. Conference on Robotics and Automation (ICRA), IEEE, 2015.

[2] T. Bandyopadhyay, KS. Won, E. Frazzoli, D. Hsu, WS. Lee, and D. Rus, Intention-Aware Motion Planning. In: Algorithmic Foundations of Robotics X, Springer, 2013.

[3] B. Bartels, and H. Liers, Movement behaviour of pedestrians in road traffic - Part 2 (in German), In: FAT-Schriftenreihe, Vol. 268, Forschungsvereinigung Automobiltechnik e.V. (FAT), 2014.

[4] S. Brechtel, T. Gindele, and R. Dillmann, Solving Continuous POMDPs: Value Iteration with Incremental Learning of an Efficient Space Representation. In: Proc. of International Conference on Machine Learning, 28, JMLR, 2013.

[5] S. Brechtel, T. Gindele, and R. Dillmann, Probabilistic Decision-Making Under Uncertainty for Autonomous Driving Using Continuous POMDPs. In: Proc. of Intelligent Transportation Systems Conference (ITSC), IEEE, 2014.

[6] YF. Chen, M. Everett, M. Liu, and JP. How, Socially Aware Motion Planning with Deep Reinforcement Learning. In: Proc. of Intl. Conference on Intelligent Robots and Systems (IROS), IEEE, 2017.

[7] YF. Chen, M. Everett, M. Liu, and JP. How, Decentralized Non-Communicating Multiagent Collision Avoidance With Deep Reinforcement Learning. In: Proc. of Intl. Conference on Robotics and Automation (ICRA), 2017.

[8] V. Dhiman et al., A critical investigation of deep reinforcement learning for navigation. In: arXiv/1802.02274, Computing Research Repository (CoRR), 2018.

[9] W. Gao, D. Hsu, WS. Lee, S. Shen, and K. Subramanian, Intention-Net: Integrating Planning and Deep Learning for Goal-Directed Autonomous Navigation. In: Proc. of 1st Annual Conference on Robot Learning, PMLR 78, 2017

[10] FA. Gers, JA. Schmidhuber, and FA. Cummins, Learning to Forget: Continual Prediction with LSTM. Journal of Neural Computation, 12(10):24512471, MIT Press, 2000.

[11] M. Jaderberg et al., Reinforcement Learning with Unsupervised Auxiliary Tasks. arXiv/1611.05397, 2016.

[12] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, Kinematic and Dynamic Vehicle Models for Autonomous Driving Control Design. In: Proc. of Intelligent Vehicles Symposium, IEEE, 2015.

[13] H. Kurniawati, D. Hsu, and WS. Lee, SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In: Proc. of Robotics: Science and Systems, 4, MIT, 2008.

[14] W. Liu, S. Kim, S. Pendleton, and MH. Ang, Situation-Aware Decision Making for Autonomous Driving on Urban Road Using Online POMDP. In: Proc. Intl. Intelligent Vehicles Symposium, IEEE, 2015

[15] Y. Luo, H. Bai, D. Hsu, and W. Sun Lee, Importance Sampling for Online Planning Under Uncertainty. Journal of Robotics Research, SAGE, 2018.

[16] O. Madani, S. Hanks, and A. Condon, On the Undecidability of Probabilistic Planning and Infinite-horizon Partially Observable Markov Decision Problems. In: Proc. of Conference on Artificial Intelligence and Innovative Applications (AAAI/IAAI), 1999.

[17] P. Mirowski et al., Learning to Navigate in Complex Environments. In: arXiv/1611.03673, Computing Research Repository (CoRR), 2016.

[18] V. Mnih et al., Asynchronous Methods for Deep Reinforcement Learning. In: Proc. of Intl. Conf. on Machine Learning, 33(48), 2016.

[19] E. Parisotto, and R. Salakhutdinov, Neural Map: Structured Memory for Deep Reinforcement Learning. In: arXiv/1702.08360, Computing Research Repository (CoRR), 2017.

[20] SD. Pendleton et al., Perception, Planning, Control, and Coordination for Autonomous Vehicles. Journal of Machines 5(1), MDPI, 2017.

[21] P. Polack et al., The Kinematic Bicycle Model: A Consistent Model for Planning Feasible Trajectories for Autonomous Vehicles? In: Proc. of Intl. Intelligent Vehicles Symposium, IEEE, 2017.

[22] Pusse, F.: HyLEAP, IS-DESPOT-p, NavA3C-p sources with OpenDS-CTS 1.0 benchmark: https://github.com/FlorianPusse/OpenDS-CTS.

[23] W. Schwarting, J. Alonso-Mora, and D. Rus, Planning and decision-making for autonomous vehicles. Journal of Annual Review of Control, Robotics, and Autonomous Systems, 1, 2018.

[24] S. Thrun et al., Stanley: The Robot That Won the DARPA Grand Challenge. Journal of Field Robotics, 23(9), 2006

[25] C. Wang, J. Wang, X. Zhang, X. Zhang, Autonomous Navigation of UAV in Large-Scale Unknown Complex Environment with Deep Reinforcement Learning. In: Proc. of Global Conference on Signal and Information Processing (GlobalSIP), IEEE, 2017.

[26] S. Ulbrich and M. Maurer, Probabilistic Online POMDP Decision Making for Lane Changes in Fully Automated Driving. In: Proc. of Intelligent Transportation Systems Conference (ITSC), 2013,

[27] N. Ye, A. Somani, D. Hsu, and WS. Lee, DESPOT: Online POMDP Planning with Regularization. Journal of Artificial Intelligence Research, 58(1):231266, AI Access Foundation, 2017.

[28] J. Zhang et al., Neural SLAM. In: arXiv/1706.09520, Computing Research Repository (CoRR), 2017.

[29] P. Zhu, X. Li, and P. Poupart, On Improving Deep Reinforcement Learning for POMDPs. In: arXiv/1704.07978, Computing Research Repository (CoRR), 2017

[30] Y. Zhu et al., Target-Driven Visual Navigation in Indoor Scenes Using Deep Reinforcement Learning. In: Proc. of Intl. Conference on Robotics and Automation (ICRA), IEEE, 2017.