Comparison of Deep Reinforcement Learning Policies to Formal Methods for Moving Obstacle Avoidance

Arpit Garg[1], Hao-Tien Lewis Chiang[1,2], Satomi Sugaya[1], Aleksandra Faust[2] and Lydia Tapia[1]

***Abstract*— Deep Reinforcement Learning (RL) has recently emerged as a solution for moving obstacle avoidance. Deep RL learns to simultaneously predict obstacle motions and corresponding avoidance actions directly from robot sensors, even for obstacles with different dynamics models. However, deep RL methods typically cannot guarantee policy convergences, i.e., cannot provide probabilistic collision avoidance guarantees. In contrast, stochastic reachabilty (SR), a computationally expensive formal method that employs a known obstacle dynamics model, identifies the optimal avoidance policy and provides strict convergence guarantees. The availability of the optimal solution for versions of the moving obstacle problem provides a baseline to compare trained deep RL policies. In this paper, we compare the expected cumulative reward and actions of these policies to SR, and find the following. 1) The state-value function approximates the optimal collision probability well, thus explaining the high empirical performance. 2) RL policies deviate from the optimal significantly and thus negatively impacts collision avoidance in some cases. 3) Evidence suggests that the deviation is caused, at least partially, by the actor net failing to approximate the action that leads to the highest state-action value.**

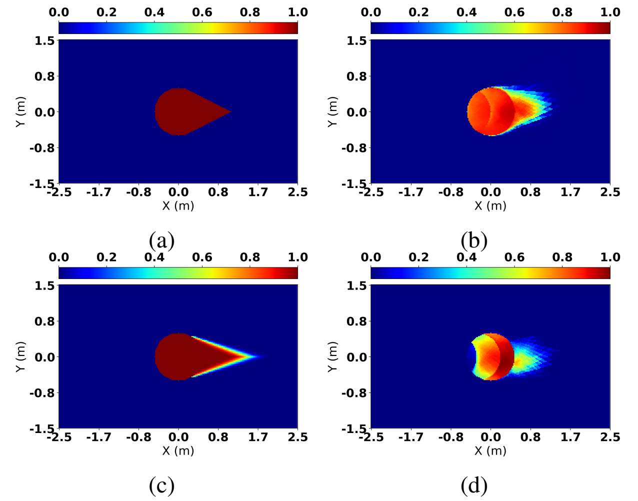

Fig. 1. V_{SR} (a, c) and normalized V_{RL} (b, d) in the relative coordinates for deterministic (a, b) and stochastic (c, d) obstacle motions.

I. INTRODUCTION

Moving obstacle avoidance is critical for many robotic applications such as self-driving cars [1], UAVs [2] and service robots [3]. However, it is a challenging problem to solve because even in the simplest case, where a 2D holonomic robot must avoid collision with polygonal obstacles moving at constant velocities, planning is NP-Hard [4] and in PSPACE [5]. Several motion planning algorithms exist for obstacle avoidance in dynamic environments [6], [7], [8]. In environments with stochastically moving obstacles, the most successful methods work by predicting the obstacle direction and velocity [6]. However, due to the problem complexity, no current solutions can guarantee collision-free navigation in crowded stochastic environments [6]. Formal methods, such as Stochastic Reachability (SR) analysis, assesses if the robot will, with a certain likelihood, remain within a desired subset of the robot state space. SR, through dynamic programming and the use of models of obstacle dynamics, can be formulated to provide probabilistic guarantees on avoiding stochastically moving obstacles.

Deep RL policies can map sensor observations, such as LiDAR information, directly to robot action, and have outperformed traditional approaches in tasks such as manipulation [9] and legged locomotion [10]. Deep RL has also been used for moving obstacle avoidance [11], [12], [13], where they learn to simultaneously predict obstacle motions and corresponding avoidance actions through trial-and-error. However, these methods have the following issues. 1) They use deep neural networks as nonlinear function approximators. Thus, in general case, there is no guarantee that the learned policy converges to an optimal or that collision avoidance is assured with a certain probability. 2) Deep RL methods are sensitive to hyperparameters and sometimes even random seeds [14], and 3) the learned robot behaviors are often difficult to interpret.

To better understand the performance of deep RL moving obstacle avoidance policies and gain insights into their behavior, we compare them with SR, which provides an optimal obstacle avoidance policy as well as collision probability. Since SR is limited in the dimensionality of the problems that it can address, we focus our comparison on avoiding a single obstacle in order to help us gain insights into the general obstacle avoidance problem. To do this, we first design a reward function that promotes obstacle avoidance and then train RL policies. Next, we interpret the corresponding state value function, an expected cumulative reward of the learned policy, as a proxy for collision probability. This interpretation enables the direct comparison of SR and RL. Specifically, we focus on A3C [15], a deep RL algorithm, and analyze the policy (actor) and state value function (critic) neural

[1]Department of Computer Science, University of New Mexico, MSC01 11301 University of New Mexico, Albuquerque, NM 87131, USA `lewispro@unm.edu`, `kiralobo@unm.edu`, `satomi@cs.unm.edu` and `tapia@cs.unm.edu`

[2]Google AI, Mountain View, CA 94043, USA `lewispro@google.com` and `faust@google.com`

networks, to gain insights into its behavior and how it differs from SR.

Results reveal the following. 1) End-to-end deep RL obstacle avoidance policies have up to 15% higher success than a state of the art multi-obstacle collision avoidance method, APF-SR [16]. 2) We observe evolving changes in behavior of RL policies during training. This was consistent across environments with deterministic and stochastic obstacle motions. 3) The state value function stored in the critic net approximates the optimal collision probability reasonably well. This explains why RL policies perform well empirically compared to the traditional methods. 4) However, the RL policy stored in the actor net deviates from the optimal policy significantly and thus negatively impacts the true policy collision probability. 5) We localize the regions of failures of RL policies. 6) Lastly, strong evidence suggests that the deviation from optimal policy is caused by the actor net failing to approximate the action that leads to the highest state action value stored in the critic net. The enclosed video demonstrates deep RL moving obstacle avoidance policies in environments with 50 obstacles compared to APF-SR.

II. RELATED WORK

A. RL in Moving Obstacle Avoidance

An approximate value iteration RL algorithm using a state value function approximated is a viable solution for moving obstacle avoidance [17], but it requires hand-engineered features such as the distance to obstacles. Recent advances in deep RL eliminate the need for hand-picked features, by relying on deep neural nets to approximate value functions [18]. As a result, deep RL policies can map raw sensor observations such as camera image or LiDAR directly to robot actions. This breakthrough inspired a new line of work using deep RL for moving obstacle avoidance. For example, an effective end-to-end (LiDAR to robot action) moving obstacle avoidance for point to point navigation and path following were achieved via Auto-RL, in which a large scale evolutionary strategy automatically tunes hyperparameters, network, and reward [11]. Proximal policy optimization [19] learns an end-to-end policy to navigate among dense crowds [12]. Another method, using a recurrent neural net to avoid collision with an arbitrary number of moving obstacles, obtains obstacle position information available through the use of clustering [13].

B. Moving Obstacle Avoidance with SR

SR assesses whether the state of the system will, with a certain likelihood, remain within and/or reach a desired subset of the state space in a finite time, or avoid an undesired subset of the state space [20]. SR can be formulated to avoid moving obstacles by setting the undesired set of states as states with a non-zero collision probability. To compute this set, methods such as [21] and [22] start with the set of states in collision and iterate backward in time using the Hamilton-Jacobi-Isaacs (HJI) equation [23]. The complement of this set of states assures collision avoidance. Unfortunately, the computational cost of dynamic programming-based SR increases exponentially with the number of obstacles and robot state space dimension [20]. As a result, SR cannot be used to avoid multiple moving obstacles directly in real-time. Several methods sacrifice the probabilistic guarantees provided by SR to work among many moving obstacles. These include using SR to bias roadmap edge weights [24] and artificial potential fields [16].

C. Evaluation of deep RL policies

Many RL methods rely on Bellman iteration to update policies [25]. RL policy converges to optimal in limited cases, e.g., when discrete or linear value function approximation and on-policy samples are used [26]. In the context of stochastic moving obstacle avoidance, optimal policies and corresponding value functions provide probabilistic guarantees of obstacle avoidance. Unfortunately, since most deep RL methods use nonlinear function approximators (neural networks) and stochastic gradient descent-based optimizers, it is very difficult to provide convergence guarantees [27]. As a result, previous work in deep RL empirically compares policy performance with traditional robotics approaches [12], [11], other RL algorithms [10], or humans [18]. These approaches may not be sufficient for moving obstacle avoidance since collisions often incur severe consequences. By quantitatively comparing deep RL policies to methods with theoretical guarantees, we can directly probe the safety and performance of deep RL for moving obstacle avoidance.

III. PRELIMINARIES

A. Robot and Obstacle Dynamics

Consider a holonomic circular robot and a circular obstacle in two dimensional workspace. The robot with radius R^r at location $\boldsymbol{x}_n^r$ is to avoid the obstacle with radius R^o at $\boldsymbol{x}_n^o$ at each discrete time step n. The robot may change its heading angle, θ_n^r, while moving with a constant speed, v^r. The obstacle moves in a straight line with a heading angle, θ^o, with velocity, $\boldsymbol{w}_n$, while its speed, $w_n = |\boldsymbol{w}_n|$, may change according to a probability mass function, $p(w_n)$. The spaces of stochastic obstacle speed and robot action are denoted by $\mathcal{W}$ and $\mathcal{U}$, respectively.

The discrete time dynamics of the robot and obstacle in relative coordinates ($\tilde{\boldsymbol{x}} = \boldsymbol{x}^r - \boldsymbol{x}^o \in \tilde{\mathcal{X}}$) is described by:

$$\tilde{\boldsymbol{x}}_{n+1} = \tilde{\boldsymbol{x}}_n + \Delta\left(f^r(\boldsymbol{u}_n, \theta_n^r) - f^o(\boldsymbol{w}_n)\right), \tag{1}$$

where Δ is the time step, $\boldsymbol{u}_n$ is the robot action, $f^r(\boldsymbol{u}, \theta^r)$ and $f^o(\boldsymbol{w})$ are the dynamics of the robot and obstacle, respectively. A collision occurs when

$$||\tilde{\boldsymbol{x}}_n||_2 \leq R^r + R^o. \tag{2}$$

B. SR Analysis

We briefly summarize SR formulation for stochastically moving obstacle avoidance. (See [28] for more details.) A value function, V, which corresponds to the collision probability over a finite time horizon, N, can be computed by formulating the SR problem in the following manner. First, we define an indicator function, $\mathbf{1}_K(\tilde{\boldsymbol{x}})$, with value one when the system is not in collision and zero otherwise. Next, we define the stochastic transition kernel, $\tau(\tilde{\boldsymbol{x}}_{n+1}|\tilde{\boldsymbol{x}}_n, \boldsymbol{u}_n)$, which gives the probability distribution of $\tilde{\boldsymbol{x}}_{n+1}$ given $\tilde{\boldsymbol{x}}_n$ and $\boldsymbol{u}_n$. The value function can be computed by the iterative relationship between time steps, starting from time step N [20]:

$$V_N(\tilde{\boldsymbol{x}}) = \mathbf{1}_K(\tilde{\boldsymbol{x}}) \tag{3}$$

$$V_n(\tilde{\boldsymbol{x}}) = \mathbf{1}_K(\tilde{\boldsymbol{x}}) \int_{\tilde{\mathcal{X}}} V_{n+1}(\tilde{\boldsymbol{x}}')\tau(\tilde{\boldsymbol{x}}'|\tilde{\boldsymbol{x}}_n, \boldsymbol{u}_n)\, d\tilde{\boldsymbol{x}}' \tag{4}$$

$$= \mathbf{1}_K(\tilde{\boldsymbol{x}}) \sum_{w\in\mathcal{W}} V^*_{n+1}\left(\tilde{\boldsymbol{x}} + \Delta\left(f^r_n - f^o_n\right)\right) p(w), \tag{5}$$

where $f^r_n = f^r(\boldsymbol{u}_n, \theta^r_n)$, $f^o_n = f^o(\boldsymbol{w}_n)$.

The optimal value function, V^*, can be computed by choosing the optimal action that maximizes the value function at each iteration:

$$V^*_n(\tilde{\boldsymbol{x}}) = \max_{\boldsymbol{u}\in\mathcal{U}} \left\{ \mathbf{1}_K(\tilde{\boldsymbol{x}}) \sum_{w\in\mathcal{W}} V^*_{n+1}(\tilde{\boldsymbol{x}} + \Delta(f^r_n - f^o_n))p(w) \right\}. \tag{6}$$

The optimal value function at $n = 0$, $V^*_0(\tilde{\boldsymbol{x}}_0)$, is the collision avoidance probability of state $\tilde{\boldsymbol{x}}_0$ given the best avoidance actions in the next N time steps. Therefore, the collision probability, V_{SR}, is simply $V_{\text{SR}} \equiv 1 - V^*_0(\tilde{\boldsymbol{x}}_0)$. Note that we can also use Eq. (6) to find optimal actions to avoid collisions from a given state $\tilde{\boldsymbol{x}}$, i.e., an optimal collision avoidance policy.

C. Deep RL

Deep RL methods typically formulate the problem as a Partially Observable Markov Decision Process (POMDP), which is a 5-tuple, $(\mathcal{O}, \mathcal{U}, \tau_{\boldsymbol{u}}, R, \gamma)$. The goal is to find an optimal policy, $\pi^*(\mathbf{o})$, that maps an observation, $\mathbf{o} \in \mathcal{O}$, to an action, $\boldsymbol{u} \in \mathcal{U}$, such that the expected discounted cumulative reward, $\mathcal{V}$, is maximized, where $\mathcal{V}$ is the sum of discounted rewards, R, by the factor, $\gamma \in [0, 1]$, along trajectories under system dynamics, $\tau_{\boldsymbol{u}}$.

A3C [15] approximates the optimal policy through the use of actor and critic neural nets. The actor net learns a policy $\pi(\mathbf{o})$ through policy gradient [29], which updates the actor net parameters towards the direction that increases the state action value function given by the critic net. Meanwhile, the critic net parameters are updated by Bellman's equation (same as Q-learning). To speed up learning, A3C employs multiple actor-learners to asynchronously collect experiences, i.e., observation, action and reward for each time step.

IV. Training of Deep RL Policies

a) Robot and environment: The 50 m by 50 m training environment (Fig. 2) has 50 moving obstacles (blue circles of radius 0.5 m) and the holonomic point robot (red dot) has a 1D LiDAR with 72 rays with a 5 m maximum range (green lines). When the obstacle reaches the boundary of the environment it teleports and reappears at the opposite boundary. We trained two policies, one with deterministic obstacle motion, a fixed velocity (2.5 m/s), and one with stochastic motion, a fixed heading but speed randomly sampled from $w = [1.5, 2.5, 3.5, 4.5]$ m/s with probability $P_w = [0.2, 0.3, 0.2, 0.3]$ at every time step (0.2 s). The robot action is either one of the 36 directions, spread evenly across 360°, or to remain stationary. The robot has a maximum speed of 1 m/s, which is up to 4.5 times slower than the maximum speed of stochastically moving obstacles.

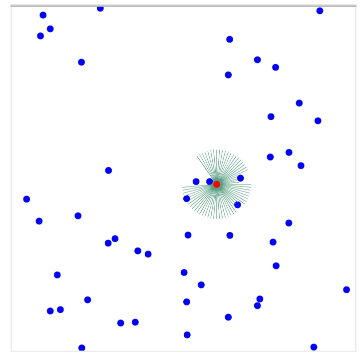

Fig. 2. Training environment. The holonomic point robot (red) makes LiDAR (green) observations of obstacles (blue).

b) RL setup: We train deep RL moving obstacle avoidance policies using A3C [15]. We chose A3C because, unlike policy optimization based methods, the critic net stores the state action value function which approximates the expected cumulative reward. The robot observes the 72 distances returned by LiDAR. To allow observation of obstacle velocity, the 5 most recent LiDAR measurements are used as the observation $\mathbf{o}$ by A3C. At every time step reward function, R, is evaluated, and it provides a value of 0.25 for a non-collision transition and -5 if the robot collides with a moving obstacle. We terminate the episode if collision occurs. We use 32 actor-learners, and the critic is updated every time 8 experiences were collected by each learner. A fully-connected network with two hidden layers, with [128, 32] neurons was used to approximate actor and critic.

V. Evaluation

In this Section, we compare deep RL value functions and policies to SR computation in order to directly probe learning outcomes. First, we begin with assessing the learning process, and after a policy is selected, we evaluate in depth the resulting value functions and actions selected by the two methods.

A. Policy selection and evaluation

During training, the learned deep RL policy constantly changes. The final policy used for obstacle avoidance is typically chosen by picking the policy with the highest cumulative reward or a performance metric [11]. Fig. 3 shows the survival rate, the percentage of 20 second collision-free runs of the robot in the training environment, as a function of training steps for the stochastic and deterministic obstacle motions. For comparison, we ran an SR based artificial potential field method, APF-SR [16], in the same environment. APF-SR uses the SR set as a repulsive potential

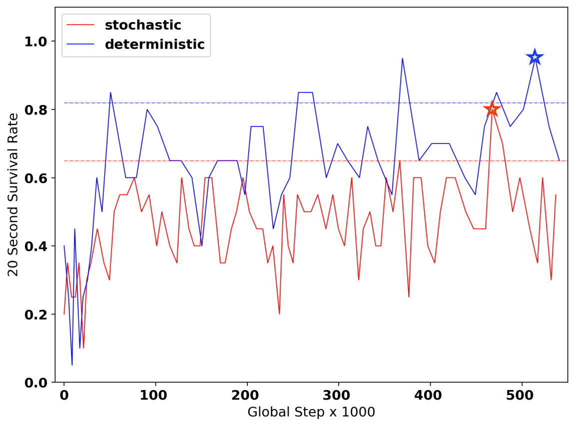

Fig. 3. 20 second survival rate as a function of global step in an environment with 50 obstacles. Deep RL policies (solid lines) and a non-learned comparison method APF-SR (dotted lines) for stochastic (red) and deterministic (blue) obstacle motions are shown. Stochastic and deterministic policies are picked at 0.80 (red star) and 0.95 (blue star) survival rates. APF-SR survival rates are 0.65 and 0.82 for stochastic and deterministic obstacle motions, respectively.

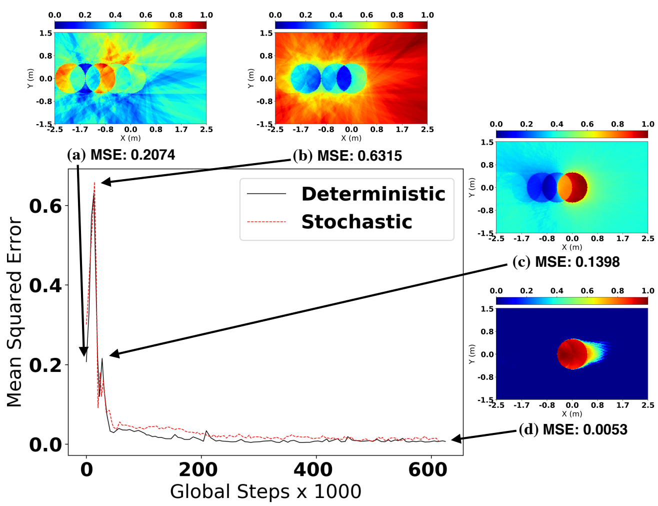

Fig. 4. MSE between V_{SR} and normalized V_{RL} as a function of global step (main figure) for deterministic (solid black) and stochastic (dotted red) obstacle motions. RL collision probabilities are shown at initial state (a), the first peak (b), the second peak (c) and convergence.

and goal as an attractive potential. Given the net potential, SR suggests an action to take. In our scenario the task is to survive without collision, thus there is no attractive potential. Fig. 3 shows the survival rates of APF-SR which is 15% and 13% lower than the survival rates of our RL policy in stochastic and deterministic obstacle motions, respectively.

For the remaining policy analysis, we empirically pick the RL policies with highest survival rate (95% for deterministic and 80% for stochastic).

B. Critic Comparison

Recall that the critic in A3C returns a scalar, V_{RL}, which approximates the expected cumulative reward. By training with the reward function described in Sec. IV, V_{RL} can be normalized to have values between zero and one to serve as a proxy for collision probability. This allows for a direct comparisons with the collision probability, V_{SR}, given the optimal avoidance policy from SR. Note that SR suffers from the curse of dimensionality and computing collision avoidance with multiple obstacles is infeasible [28]. Therefore, we compare V_{SR} and normalized V_{RL} in the presence of one moving obstacle.

Fig. 4 shows the Mean Squared Error (MSE) between V_{SR} and normalized V_{RL}, for the deterministic and stochastic obstacle motion as a function of global steps. Both curves follow the same trend, with two peaks of larger error before MSE converges. The four inset figures of normalized V_{RL} in Fig. 4 show distinct robot behaviors during training. At the initial stage of learning (inset a), V_{RL} is essentially random. Next, the robot learned to approach the obstacle, resulting in a high collision rate and MSE (inset b). This behavior helps the robot in the next stage of learning avoid the obstacle (inset c), but it does not consider the motion of obstacle. Lastly at convergence (inset d), the robot learns to consider obstacle motion thus resulting in a low MSE.

Figs. 1b and 1d show normalized V_{RL} for the best performing policies (as picked in Section V-A), given deterministic (top row) and stochastic (bottom row) obstacle motion. Also, for comparison, the V_{SR} is shown in Figs. 1a and 1c. The V_{SR} plots show that since the obstacle moves faster than the robot, the robot cannot avoid collision if it is positioned inside the obstacle or in a small region in front of the obstacle (collision probability of one). In addition to these regions, stochastically moving obstacles also have regions where the robot may only probabilistically avoid collision. Collision avoidance would only be achievable if the obstacle moves at slow speed. Lastly, there is a large area with zero collision probability for SR. The critic net captures these regions reasonably well as normalized V_{RL} shown in Figs. 1 (b) and 1 (d) is similar to V_{SR}, resulting in a low MSE.

C. Actor Comparison

After comparing the critic net with the collision probability given by SR, we compare the deep RL policy (actor net) with the optimal policy given by SR (Eq. (6)). Fig. 5 shows action as a function of robot position (white arrows) overlaid with V_{SR} on the left and normalized V_{RL} on the right, for obstacles with deterministic motion (top row) and stochastic motion (bottom row). A visual comparison of the arrows shows that RL actions are not the same as optimal actions given by SR. For example, there are actions in the RL deterministic obstacle case that suggest outrunning the obstacle. And, there are actions in the RL stochastic case that seemingly move toward the incoming obstacle. These suggest potential collisions. It should be noted that in V_{SR} arrows are not shown for those locations with zero collision probability because all actions are equally optimal. The deep RL policy, however, returns an action for each location.

Since the actions selected by the actor net deviates significantly from SR, we would like to answer the following questions: 1) How does this affect collision avoidance performance (Sec. V-D) and 2) what causes the discrepancy (Sec.

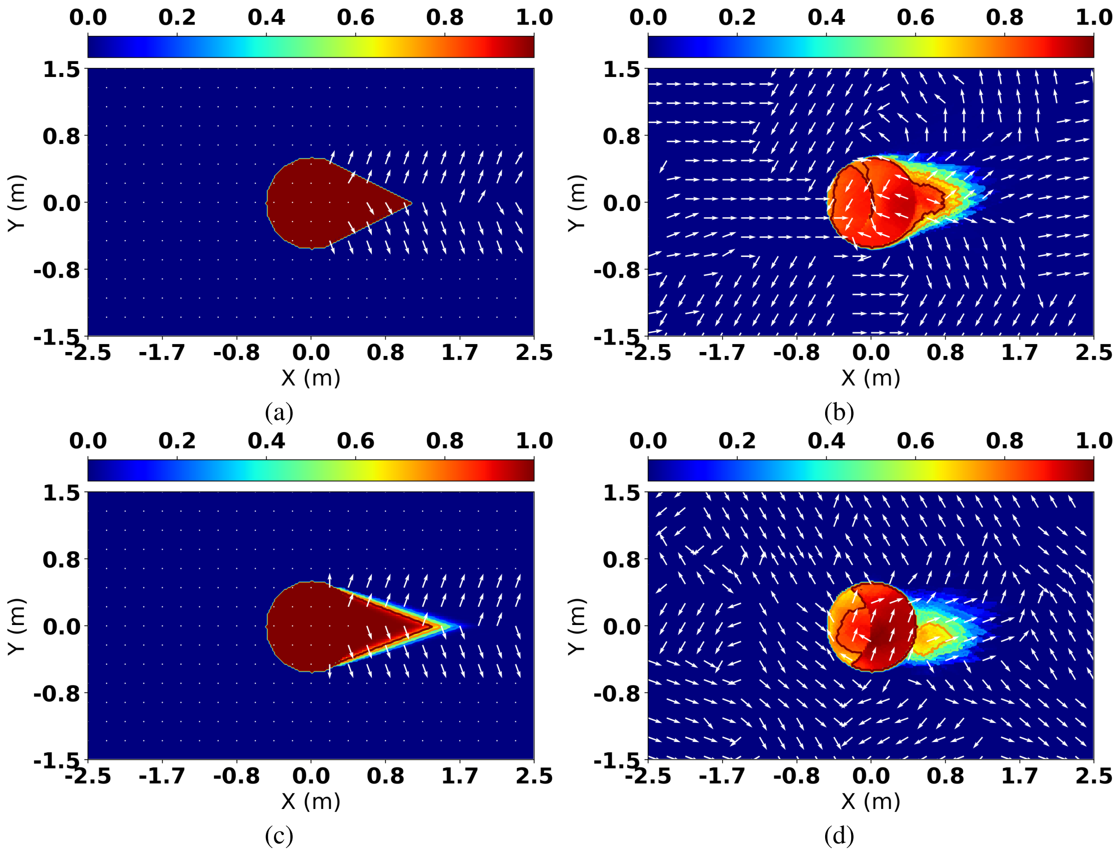

Fig. 5. V_{SR} (a, c) and normalized V_{RL} (b, d) contour overlayed with SR and RL action at each position (shown by arrows), for deterministic (a, b) and stochastic (c, d) obstacle motions.

V-D).

D. Collision Probability Comparison

In this section, we analyze how the deep RL and SR policy discrepancy affects collision avoidance. To do this, we compute the RL collision probability of a given position. This is achieved by starting the robot at every position and executing actions given by the actor net until either collision occurs or a horizon of six time steps is reached. This process is repeated 20 times for each position.

Fig. 6 (b, e) show the RL collision probability compared to the optimal collision probability given by SR, Fig. 6 (a, d), for deterministic (top row) and stochastic (bottom row) obstacle motions. It is clear that the RL collision probability differs significantly from the optimal. This means that the deep RL policy is not optimal and is more likely to collide with obstacles, particularly in regions where the RL policy deviates significantly with the optimal policy.

E. Causes for Sub-optimality

In this section, we aim to identify causes for the sub-optimality of the deep RL policy. We observe that normalized V_{RL} (Fig. 1 (b)) is close to the collision probability given by SR (Fig. 1 (a)) for deterministic obstacle motion. This is also reflected in the low MSE in Fig. 4. However, the RL policy deviates from the optimal given by SR significantly. This led to a hypothesis that *the actor net failed to approximate the action that leads to the highest state-action value stored in the critic net.* To support this hypothesis, we bypassed the actor net and devised a new RL policy from the critic net, $\pi_{\text{critic}}(\mathbf{o})$. This policy maps observation to action by evaluating the critic state-action value, $Q(\mathbf{o}, \mathbf{a})$, for each action and select the action $\mathbf{a}_{\text{critic}}$ with the highest Q:

$$\mathbf{a}_{\text{critic}} = \operatorname*{argmax}_{\mathbf{a}} Q(\mathbf{o}, \mathbf{a}) \tag{7}$$

We compute the collision probability of this "critic policy" with the procedure described in Sec. V-D. Figs. 6 (c, d) show this collision probability for deterministic and stochastic obstacle motions, respectively. Comparing to the collision probability of the original RL policy (actions obtained from the actor net, shown in Figs. 6 (b, e)), it is clear that the critic policy performs better for deterministic obstacle motion. This is a strong evidence supporting our hypothesis.

To further support our hypothesis, we compare the actions of the original deep RL policy (left column) and critic policies (right column) in Fig. 7 for deterministic (top row) and stochastic (bottom row) obstacle motions. It is clear that the actions are significantly different. This strongly suggests that the actor net failed to approximate the action that leads to the highest state-action value stored in the critic net. The same conclusion can be drawn for stochastic obstacle motion, as Figs. 7 (c, d) show that the two policies are significantly different.

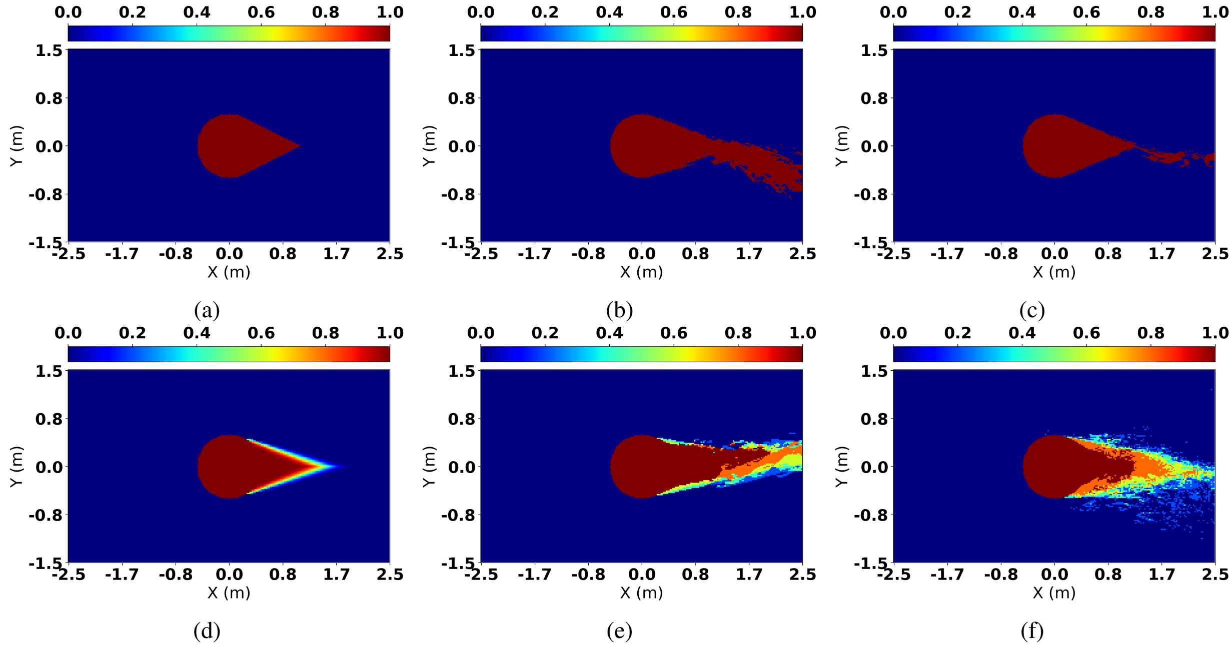

Fig. 6. V_{SR} (a, d), collision probability of the RL policy (b, e) and "critic policy" described in Sec. V-D (c, f) for deterministic (top) and stochastic (bottom) obstacle motions.

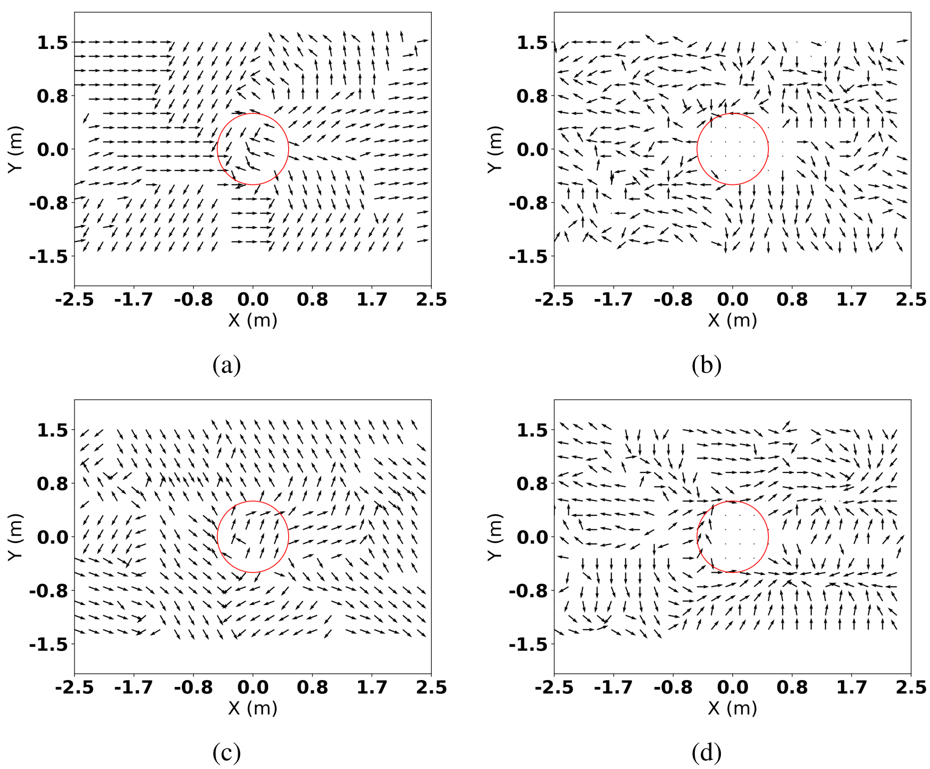

Fig. 7. RL actions as a function of position given by the actor net (a, c) and critic net ((b, d), actions that lead to the highest state-action value). The top and bottom rows show the obstacle (red circle) with deterministic and stochastic motion, respectively.

However, in contrast to deterministic obstacle motion, Fig. 6 (f) shows the critic policy for stochastic motion performs only slightly better than the original policy 6 (e). This indicates that the critic is not learning the optimal collision probability, as shown in Fig. 1 (c, d). This maybe due to the stochastic nature of obstacle speeds, which has a higher average speed (3.1 m/s compared to 2.5 m/s of deterministic obstacle motion) and randomly varies every time step.

We also ruled out some other sources for the discrepancy. We tried increasing the number of LiDAR beams to 288 (4 times as many as the) and increasing the network size and shape. These changes did not result in better empirical performance among many obstacles, and the true policy collision probability did not better approximate the optimal.

VI. CONCLUSION

In this paper we examined how the deep RL actor and critic compare to a traditional formal method, SR, for the task of avoiding moving obstacles. To that end, we performed a comparative analysis. In the presence of multiple moving stochastic obstacles RL performs empirically better than a state of the art planning method while having less information about obstacle dynamics and position. In the presence of a single obstacle, we uncover regions where RL policy under-performs an optimal SR computation. This is important because it gives a cue to practitioners where secondary safety policies might need to be added, to improve obstacle avoidance in physical systems. We also discover a consistent evolution of the RL agents during the training, where learning for both deterministic and stochastic obstacles passes through the same phases. This is important as it can provide further insight into both the performance of the agent and when deciding if more training is necessary.

ACKNOWLEDGMENT

We would like to thank Meeko Oishi, University of New Mexico, and Kendra Lesser, Verus Research, for help computing the SR sets for this work. Tapia, Chiang, Garg, and Sugaya are partially supported by the National Science Foundation under Grant Numbers IIS-1528047 and IIS-1553266. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[2] C. Goerzen, Z. Kong, and B. Mettler, "A survey of motion planning algorithms from the perspective of autonomous uav guidance," *Journal of Intelligent & Robotic Systems*, vol. 57, no. 1, pp. 65–100, 2010.

[3] R. Triebel, K. Arras, R. Alami, L. Beyer, S. Breuers, R. Chatila, M. Chetouani, D. Cremers, V. Evers, M. Fiore *et al.*, "Spencer: A socially aware service robot for passenger guidance and help in busy airports," in *Field and Service Robotics*. Springer, 2016, pp. 607–622.

[4] J. Canny and J. Reif, "New lower bound techniques for robot motion planning problems," in *Found. of Comp. Sci., Annual Symp.* IEEE, 1987, pp. 49–60.

[5] J. Canny, "Some algebraic and geometric computations in pspace," in *Proc. of annual ACM symp. on Theory of computing*, 1988, pp. 460–467.

[6] H.-T. L. Chiang, B. HomChaudhuri, L. Smith, and L. Tapia, "Safety, challenges, and performance of motion planners in dynamic environments," in *The International Symposium on Robotics Research (ISRR)*, 2017, pp. 1–16.

[7] R. Benenson, S. Petti, T. Fraichard, and M. Parent, "Integrating perception and planning for autonomous navigation of urban vehicles," in *Proc. IEEE Int. Conf. on Intel. Robot. Sys. (IROS)*, 2006, pp. 98–104.

[8] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Robot. Res.*, vol. 17, no. 7, pp. 760–772, 1998.

[9] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke *et al.*, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," *arXiv preprint arXiv:1806.10293*, 2018.

[10] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke, "Sim-to-real: Learning agile locomotion for quadruped robots," *arXiv preprint arXiv:1804.10332*, 2018.

[11] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end to end with auto-rl," *Robot. and Automat. Lett.*, pp. 2007–2014, 2019.

[12] T. Fan, X. Cheng, J. Pan, P. Long, W. Liu, R. Yang, and D. Manocha, "Getting robots unfrozen and unlost in dense pedestrian crowds," *Robot. and Automat. Lett.*, pp. 1178–1185, 2019.

[13] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Proc. IEEE Int. Conf. on Intel. Robot. Sys. (IROS)*. IEEE, 2018, pp. 3052–3059.

[14] H. Mania, A. Guy, and B. Recht, "Simple random search of static linear policies is competitive for reinforcement learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 1805–1814.

[15] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. on Machine Learning (ICML)*, 2016, pp. 1928–1937.

[16] N. Malone, H.-T. Chiang, K. Lesser, M. Oishi, and L. Tapia, "Hybrid dynamic moving obstacle avoidance using a stochastic reachable set-based potential field," *IEEE Trans. Robot.*, pp. 1124–1138, 2017.

[17] A. Faust, H.-T. Chiang, N. Rackley, and L. Tapia, "Avoiding moving obstacles with stochastic hybrid dynamics using pearl: preference appraisal reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*. IEEE, 2016, pp. 484–490.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[19] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[20] A. Abate, M. Prandini, J. Lygeros, and S. Sastry, "Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems," *Automatica*, vol. 44, pp. 2724–2734, 2008.

[21] K. Margellos and J. Lygeros, "Hamilton-Jacobi formulation for reach-avoid problems with an application to air traffic management," *American Control Conference (ACC)*, pp. 3045–3050, 2010.

[22] J. H. Gillula, G. M. Hoffmann, H. Haomiao, M. P. Vitus, and C. J. Tomlin, "Applications of hybrid reachability analysis to robotic aerial vehicles," *Int. J. Robot. Res.*, vol. 30, pp. 335–354, 2011.

[23] I. Mitchell, A. Bayen, and C. Tomlin, "A time-dependent Hamilton-Jacobi formulation of reachable sets for continuous dynamic games," *Transaction on Automatic Control*, vol. 50, pp. 947–957, 2005.

[24] N. Malone, K. Lesser, M. Oishi, and L. Tapia, "Stochastic reachability based motion planning for multiple moving obstacle avoidance," in *Hybrid Systems: Computation and Control*. HSCC, 2014, pp. 51–60.

[25] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," *arXiv preprint arXiv:1708.05866*, 2017.

[26] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic programming*. Athena Scientific Belmont, MA, 1996, vol. 5.

[27] B. Dai, A. Shaw, L. Li, L. Xiao, N. He, Z. Liu, J. Chen, and L. Song, "Sbeed: Convergent reinforcement learning with nonlinear function approximation," *arXiv preprint arXiv:1712.10285*, 2017.

[28] H.-T. Chiang, N. Malone, K. Lesser, M. Oishi, and L. Tapia, "Aggressive moving obstacle avoidance using a stochastic reachable set based potential field," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pp. 73–89.

[29] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. on Machine Learning (ICML)*, 2014.