Generating Adversarial Driving Scenarios in High-Fidelity Simulators

Yasasa Abeysirigoonawardena[1], Florian Shkurti[2], and Gregory Dudek[1]

Abstract— **In recent years self-driving vehicles have become more commonplace on public roads, with the promise of bringing safety and efficiency to modern transportation systems. Increasing the reliability of these vehicles on the road requires an extensive suite of software tests, ideally performed on high-fidelity simulators, where multiple vehicles and pedestrians interact with the self-driving vehicle. It is therefore of critical importance to ensure that self-driving software is assessed against a wide range of challenging simulated driving scenarios. The state of the art in driving scenario generation, as adopted by some of the front-runners of the self-driving car industry, still relies on human input [1]. In this paper we propose to automate the process using Bayesian Optimization to generate adversarial self-driving scenarios that expose poorly-engineered or poorly-trained self-driving policies, and increase the risk of collision with simulated pedestrians and vehicles. We show that by incorporating the generated scenarios into the training set of the self-driving policy, and by fine-tuning the policy using vision-based imitation learning we obtain safer self-driving behavior.**

I. INTRODUCTION

How can we evaluate the performance of self-driving cars in terms of safety in the face of rare events? One approach is to measure miles per intervention from real driving data on the road. According to recent estimates [2], the number of miles that would have to be driven on the road in order to demonstrate the safety of a car in a statistically significant sense is in the range of hundreds of millions of miles, which corresponds to decades of driving, assuming current projections about the sizes of fleets of self-driving vehicles. Another, complementary, approach is to design rich simulation scenarios that will provide a preliminary assessment of the self-driving software stack. The question then becomes: what is the best way to design these driving scenarios in order to minimize the chances of accidents on public roads?

Current best practices for stress-testing self-driving navigation software involve a semi-autonomous process, in which a human operator specifies the number of cars and pedestrians in the scene, their initial locations and their approximate trajectories, while their speeds are automatically selected from a wide range of options [1]. Human involvement and specification renders this process time-consuming and difficult to scale to new environments. Most importantly, however, it could result in missing critical testing configurations.

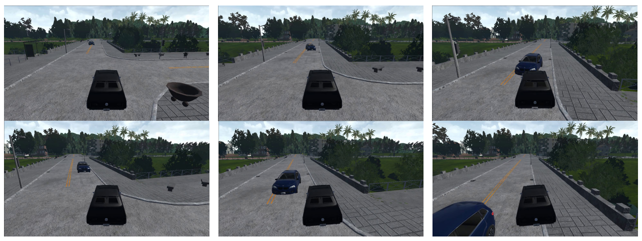

Fig. 1. Top row: generated adversarial scenario in which the self-driving policy cannot avoid the crash. Bottom row: policy avoids the crash by driving on the empty sidewalk after its training set has been augmented with expert reactions to adversarial scenarios. Videos available at `http://www.cim.mcgill.ca/~mrl/adversarial_driving_scenarios`

In this paper we show how to automate the generation of driving scenarios in high-fidelity simulators, by optimizing the behaviors of pedestrians and other vehicles on the road (Non-Player Characters, or NPCs for short), so that they navigate adversarially with respect to the self-driving vehicle that is being tested. Our proposed system automatically explores and finds parameters of NPC trajectories, in a way that reduces human involvement, and results in possible crashes and dangerous configurations.

We formulate the problem of searching for adversarial NPC behaviors in terms of Bayesian Optimization (BO), and we trade off exploration vs. exploitation, searching for the global minimum of a black box function. In our work this black-box function is the unknown cumulative performance (cost-to-go function) of a self-driving policy, given the trajectories of the NPC adversaries in a high-fidelity photorealistic simulator [3]. By keeping track of the uncertainty of the cost-to-go function using a Gaussian Process we perform informed exploration using the criterion of expected improvement. We also create an efficient hierarchical parametrization for the behavior of vehicles and pedestrians, and we optimize it to determine challenging scenarios for the self-driving policy.

The novelty of our proposed testing approach lies in the fact that we generate scenarios based on measurements of the self-driving performance of the given policy, whereas traditional testing based on human specification and predefined test cases does not do that.

We also show that when the generated adversarial scenarios are included in the training set they improve the self-driving policy, as long as expert demonstrations for how to avoid the accident are obtained. We obtain expert demonstrations from a path planning and control system,

*This work was supported by the Natural Sciences and Engineering Research Council (NSERC)

[1]Affiliated with the Mobile Robotics Lab at the Center for Intelligent Machines (CIM), McGill University, Montreal, Canada, {`yasasaa, dudek`} @ `cim.mcgill.ca`

[2]Affiliated with the Department of Computer Science, University of Toronto, Canada, `florian@cs.toronto.edu`

similar to [4], however, that is not a strict requirement and human input is also easy to integrate.

Our experiments demonstrate that as a result of fine-tuning the policy using visual imitation learning according to DAgger [5], the driving behavior becomes safer and accidents are reduced. We compare our method against baselines such as random parameter search and the Cross-Entropy Method and we show that Bayesian Optimization finds better adversarial scenarios faster compared to either of them.

II. RELATED WORK

A. Bayesian Optimization

Bayesian optimization (BO) [6], [7], [8] is an approach for finding global optima of a noisy black box function. The function in question is assumed to be costly to evaluate, so we would like to use as few evaluations as possible. In our case, photorealistic driving simulators, such as CARLA [3], which are based on game engines like the Unreal Engine 4, are currently not able to render faster than real time. This places a limit on the number of experiments that can be performed and the amount of experience than can be collected in a given time interval. We therefore need to judiciously select the most informative points at which to obtain noisy measurements from this simulation.

Bayesian Optimization has been used in many robotics applications, ranging from active sampling [9] to control and reinforcement learning [10]. BO is determined by two design choices: the prior over functions and the acquisition function, which formalizes the utility of querying the value of the unknown function at a given point. Finding the optimal value of the acquisition function is typically a much easier problem than finding the global optima of the unknown function. Examples of acquisition functions include the probability of improvement, expected improvement, upper confidence bound, and Thompson sampling [11]. Acquisition functions typically model the tradeoff between exploration (querying points in high uncertainty regions) and exploitation (querying points near the optima found so far). As such, there are many links between BO and the multi-armed bandit literature [12].

B. Reinforcement Learning and Informed Exploration

Our approach is also related to model-free reinforcement learning methods, particularly those involving exploration strategies that take into account the uncertainty in the value function induced by the behavior policy. Such approaches perform exploration by Thompson sampling [13], which is often more effective than the ϵ-greedy strategy.

Methods in this category have modeled the uncertainty in the value function using the bootstrap method [14], Bayesian linear regression [15], and variational inference [16]. Other methods, such as [17] have added noise in the policy parameters and have shown that it improves exploration. Finally, another approach has been inspired by the control-as-inference view [18], [19] and adds an explicit policy entropy term to the objective function to encourage exploration [20]. Although these approaches are very promising and scalable to high-dimensional states, they are typically not data-efficient, requiring in the order of millions of evaluations. As mentioned previously, photorealistic simulators such as the Unreal Engine 4 are currently not at the stage where they can be run faster than real-time so there is still a significant time cost to be paid for rollouts and evaluations.

Our method is also related to multi-agent reinforcement learning, and particularly adversarial behaviors from other agents, considered part of the environment, but whose actions we can observe and whose strategies we can model. Examples of methods in this category include minimax Q-learning [21], the decentralized Partially Observable Markov Decision Process (POMDP) [22], and more recent multi-agent reinforcement learning methods, such as [23], [24].

C. Adversarial Optimization

The representational capacity of neural networks has also brought increased and renewed interest in examining the conditions under which they generalize as well as a rich body of work in generating small adversarial perturbations on the input that drastically change the output. Most of this work on adversarial examples has focused on fooling classification networks [25], but recently they have been also applied to reinforcement learning [26]. Our work goes beyond this, in that we want to find *adversarial policies and trajectories* that will lead the self-driving policy to an unsafe state. Our paper is most related in purpose, but not in content or methodology, to [27], which does image synthesis and perturbations to account for example for weather changes that will make a CNN driving policy choose the wrong actions. Our paper is also related in purpose to [28] in that trajectories of pedestrians and vehicles are modelled in order to produce typical or atypical traffic behavior.

In addition to this research, there is data efficient imitation learning [29] that is inspired by contrastive divergence methods [30] and most notably Generative Adversarial Networks [31], where a discriminator network separates examples that have come from the expert as opposed to having been sampled from a generator network. This method, however, is not suitable for the exploration of possible trajectories that is required for our purposes.

D. Differentiable Rendering

The traditional computer graphics pipeline aims to efficiently create a 2D image rendering of a known 3D scene description, via ray tracing, shading, texture mapping, and other such methods. While these technologies have reached the stage where they can very efficiently perform realistic image synthesis, what they lack is a mapping between inputs and outputs in a way that correlations between rendering parameters and the resulting image can be computed analytically, as opposed to empirically. Although simple differentiable renderers have been proposed, for instance in [32], they are not yet part of typical game engines.

Differentiable rendering [33], [34] uses neural networks to model this mapping, thus allowing for inverse rendering, where we want to find the rendering parameters that lead to desired properties in the resulting images or match properties of observed sensor data. Progress in this direction will enable our work to use model-based reinforcement learning end-to-end gradient-based optimization through the rendering process.

III. MODELING ADVERSARIAL BEHAVIORS

Let x_t be the state vector of the self-driving vehicle at time t. This vector includes the 2D position and orientation of the vehicle in a global reference frame. Let s_t be the vector containing the states of the NPCs, both pedestrians and other vehicles involved in a driving scenario. This vector involves the 2D positions and orientations $s_t^{(i)}$ of all agents in the simulation, except the self-driving vehicle. Since we are working with a simulator we know the dynamics model of the self-driving vehicle state $x_{t+1} \sim p(x|x_t, a_t)$ and of the NPC state $s_{t+1} \sim q(s|s_t, u_t)$, with a_t and u_t being the respective controls.

We determine the actions of NPCs by using a parametric policy $u_t \sim \pi_\phi(u|s_t)$. On the other hand, the self-driving vehicle's actions are determined by a policy $a_t \sim \pi_\theta(a|o_t)$, which is the policy that we want to stress-test. This policy depends on the observation o_t of the vehicle, which in our work includes only the current image, but can easily be extended to previous images, LiDAR, radar and other relevant sensors.

The observation model that characterizes the simulator's rendering process from the viewpoint of the self-driving car is denoted by $o_t \sim h(o|x_t, s_t)$. In our case o_t is an image and the process for generating it, the physics and rendering engine of UE4, is not differentiable. So, model-based reinforcement learning approaches are challenging to apply here. The time evolution of the joint state of the system is given by the following generative model:

$$o_t \sim h(o|x_t, s_t) \tag{1}$$
$$u_t \sim \pi_\phi(u|s_t) \tag{2}$$
$$a_t \sim \pi_\theta(a|o_t) \tag{3}$$
$$x_{t+1} \sim p(x|x_t, a_t) \tag{4}$$
$$s_{t+1} \sim q(s|s_t, u_t) \tag{5}$$

The finite-horizon cost-to-go function for our self-driving simulation scenarios, induced by the two policies and the observation model mentioned above, is the following:

$$J(\theta, \phi) = \sum_{t=0}^{T} \mathbb{E}_{x_t, s_t}\left[c(x_t, s_t) \mid x_0, s_0, \pi_\theta, \pi_\phi\right] \tag{6}$$

where $c(x_t, s_t)$ denotes the instantaneous cost which measures how close the vehicle x_t is to crashing with any of the NPCs in s_t. Setting this cost function to be an indicator variable for crashing events gives rise to a very sparse feedback, which makes optimization harder due to lack of gradients. Instead, we model it as the closest distance of the self-driving car x_t to any of the NPCs in s_t, and also include a penalty term for car collisions in order to encourage crashes as opposed to sideways contact or side-by-side driving at close distances. This extra term is not required for collisions with pedestrians. The final form of the instantaneous cost function is:

$$c(x_t, s_t) = \min_i\{||x_t - s_t^{(i)}||\} - \lambda \mathbb{1}_{[\text{car collision}]} \tag{7}$$

Minimizing the cost-to-go function corresponding to this cost means making the NPCs navigate in such a way that they are eventually attracted to the changing self-driving car location. In other words, we are interested in solving the following optimization problem:

$$\phi^* = \underset{\phi}{\operatorname{argmin}}\ J(\theta, \phi) \tag{8}$$

where the self-driving policy parameters θ are kept fixed during this optimization process. We integrate the sequence of NPC behaviors ϕ_i obtained by BO and we include the ones that are deemed the most dangerous according to their cost-to-go function in the training set of the self-driving policy π_θ. Our system is shown in Alg. 1.

Algorithm 1 Data-Efficient Adversarial Driving Example Generation for Imitation Learning

```
 1: $\theta \leftarrow \theta_0$   self-driving policy parameters
 2: $D := \{(o_t, a_t)\}$ existing set of driving examples
 3: for i=1...K do
 4:     // $\phi$ are the NPC policy parameters
 5:     Initialize GP model for the cost-to-go $J$
 6:     $D' \leftarrow \{\}$ measurements obtained from simulator
 7:     for j=1...L do
 8:         $\phi_j = \underset{\phi}{\operatorname{argmax}}\ \alpha_{\text{EI}}(\phi; D')$
 9:         Measure $y = \hat{J}(\theta, \phi_j)$ from simulator rollouts
10:         Update GP model for cost-to-go $J$
11:         $D' \leftarrow D' \cup \{(\phi_j, y)\}$
12:     for $(\phi_j, y) \in D'$ do
13:         if $y < \delta$   (scenario is dangerous) then
14:             $o_{1:N}, a_{1:N} \leftarrow$ query expert for
                    reaction to scenario $\phi_j$
15:             $D \leftarrow D \cup \{o_{1:N}, a_{1:N}\}$
16:     Retrain driving policy $\pi_\theta$ on the updated set $D$
         using supervised learning
```

IV. BAYESIAN OPTIMIZATION

We model the cost-to-go function using a Gaussian Process (GP) [35] in order to capture the uncertainty around the performance of scenarios that we have not evaluated yet, in order to select informative examples. In particular, we fix the parameters θ of the driving policy and for notational convenience we view J solely as a function of ϕ. We consider $J \sim \mathcal{GP}(0, K_{\sigma,l}(\phi, \phi'))$, where the kernel is Matern 5/2 parameterized by variance σ^2 and length scale l, similarly to [7]. This allows modeling less smooth functions compared to radial-basis functions kernels. We model observations of J made through simulator rollouts as:

$$y = J(\phi) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \beta^2) \tag{9}$$

where β is the deviation of the measurement noise. The GP hyperparameters β, l, σ are found by maximizing the evidence lower bound of the log-marginal likelihood $\log p(y_i|\phi_i, \beta, l, \sigma)$ of the given data in $D' = \{\phi_i, y_i\}$. We do this using stochastic variational inference [36], as implemented in the deep probabilistic programming framework `pyro`[1].

We use Bayesian Optimization to find the optimal adversarial scenario $\phi^* = \underset{\phi}{\text{argmin}}\ J(\phi)$, but most importantly, due the sequential nature of the minima discovered, this method gives us a *sequence* of adversarial examples that make the cost-to-go function stay below some threshold.

The mean and variance estimates from the GP is used to create an acquisition function $\alpha(\phi)$ which formalizes the exploration vs. exploitation tradeoff, and whose optimization tells us which scenario to evaluate next. We use expected improvement as the acquisition function:

$$\alpha_{\text{EI}}(\phi) = \mathbb{E}_{p(y|\phi, D')}[I(\phi)] \tag{10}$$

where $I(\phi) = \max(J^* - y, 0)$ is the improvement function at the point ϕ, J^* is the current best measurement for the minimum value of the function, and expectation is taken with respect to the distribution over y given the dataset $D' = \{(\phi_i, y_i)\}_{i=1\ldots n}$ of measurements so far. We have analytical gradients for $\alpha(\phi)$ due to the Gaussian assumption of the GP posterior. The maximal value of the expected improvement function will correspond to the scenario ϕ that that will be evaluated next, given our current set of measurements D'.

Despite its benefits for modeling uncertainty, BO suffers from scalability issues and the curse of dimensionality: as the number of input dimensions increases, the required points to cover the feasible set increases exponentially. Furthermore, finding the global optimum of $\alpha(\phi)$ is significantly more difficult in higher dimensional settings. There have been attempts to perform BO in higher dimensions by exploiting the fact that a large class of problems have a low effective dimensionality [37], [38], however we do not rely on them in the current version of this work.

V. Hierarchical Policy Representation

Our key insight for making this optimization process data-efficient and for avoiding wasteful exploration of NPC behaviors that do not lead to crashes, is that we impose a hierarchy on the NPC behavior that makes use of the map of the simulator. We create a graph based high-level parametrization of major decision points that have to be made by NPC agents. Nodes represent the opportunity to make a low-level policy switch. Edges represent a space interval where a single low-level policy is executed. This is better illustrated in Fig. 2.

For NPC cars these decision points are spaced every few meters on the drivable road and they represent opportunities for: switching to another lane, making a turn at an intersection, switching to another speed distribution. For NPC pedestrians the graph includes nodes that are possible

[1]`http://www.pyro.ai`

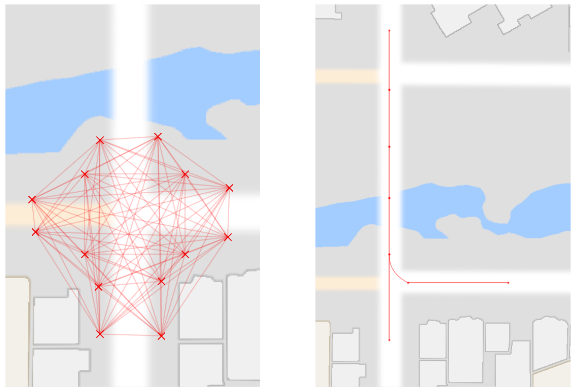

Fig. 2. (a) The nodes a pedestrian NPC can visit, and their edges, which enable both normal and jaywalking behavior. (b) The graph that determines the car NPC decision points.

waypoints on the sidewalk and across intersections, and edges are established between all possible pairs, thus being able to model both typical pedestrian behaviors as well as jaywalking behavior that would normally violate traffic laws.

A. Vehicle Parametrization

The low-level policies of NPC vehicles perform lane-following on the drivable roads of the map. Roads and intersections are modeled by a graph. When traversing an edge e in this graph, the low-level policy is guided by a target road boundary-following distance d_e. This parameter determines which lane the car should drive in, allowing atypical behaviors, such as turning on the wrong lane, driving between lanes, or even in the opposite direction of traffic.

If we overload our notation for the NPC policy so as to represent the behavior of a single NPC vehicle, we will have $\pi_\phi(u_t|s_t) = \sum_{e \in E} \pi_\phi(u_t|s_t, d_e) p(e|s_t) = \pi_\phi(u_t|s_t, d_{e_t})$ because we assume that $p(e|s_t)$ is 1 only for a single edge, e_t. The control vector for vehicles consists of linear velocity, angular velocity, and the identity of the next edge to be followed $u_t = (v_t, \omega_t, e_t)$.

Furthermore, at intersections NPC cars can decide whether they will make a turn or not, and if so, at which lane. At an $n-$way intersection the low-level policies will have sufficient weight parameters corresponding to the number directions in the intersection.

B. Pedestrian Parametrization

We model our pedestrians in similar fashion to the vehicles. Given a set of m nodes in the city the path of an NPC pedestrian is going to be parametrized as $[n_i, v_i]_{i \in \{1 \ldots k\}}$ $\ k \le m$ where v_i is a speed parameter for each edge and n_i are the index of a node in the graph, relaxed to a continuous variable that is rounded after optimization.

VI. Evaluation

We evaluate the quality of the generated adversarial examples in three ways. First, we compare our Bayesian Optimization method against two baselines: random search and the

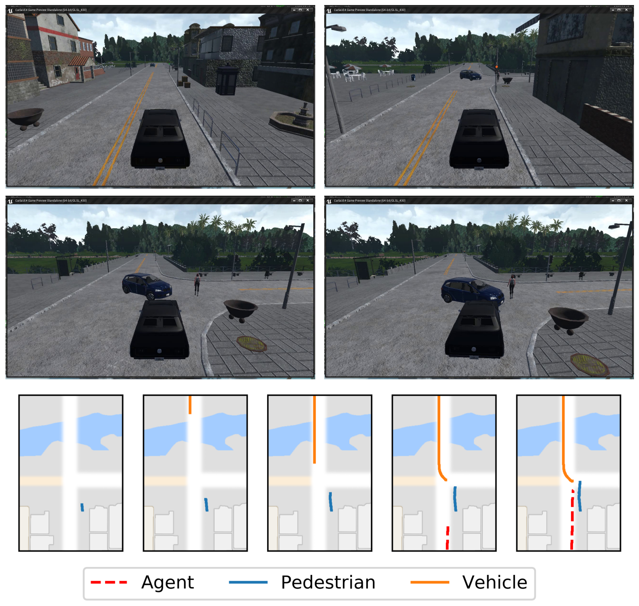

Fig. 3. Collision during scenario #3, in which a pedestrian crosses the road and prevents the incoming vehicle from completing its turn on time. The self-driving agent has not seen this scenario before, so its policy does not act correctly and crashes onto the NPC vehicle.

Cross-Entropy Method, with respect to the minimum cost-to-go that they achieve over time. We ran our experiments for 200 function evaluations, across three different types of scenarios that respectively involve: adversarial human NPCs, vehicle NPCs, and both. Each experiment was repeated across three random seeds.

Second, we measure the number of crashes and the magnitude of impact for each of these three methods. Measuring the impact during a collision is done in CARLA [3] based on the functionality of Unreal Engine 4. The measurement consists of the velocity norms of the agents involved at the exact time a collision event is issued. We found that our BO method produced crashes with about 1.4 times higher impact than the other methods, and it also discovered more accidents on average.

Third, we show qualitative scenarios for our method to give a sense of the complexity of the scenarios that it can generate. Finally, as illustrated in Fig. 1, we show that re-training the policy through imitation learning after including the generated sequence of adversarial scenarios, leads to a safer policy than just using typical driving scenarios.

A. Adversarial Pedestrian NPC

In this experiment we setup a scenario with a single adversarial pedestrian and our self-driving agent, which traverses a straight road with one intersection in the middle. The pedestrian is initialized at the intersection, with available navigation waypoints as shown in Fig. 2. The results of this scenario are shown in Fig. 4, where BO clearly performs better than random search and cross-entropy method (CEM), given the same function evaluation budget. It quickly finds a crash scenario with pedestrian while the other methods fail

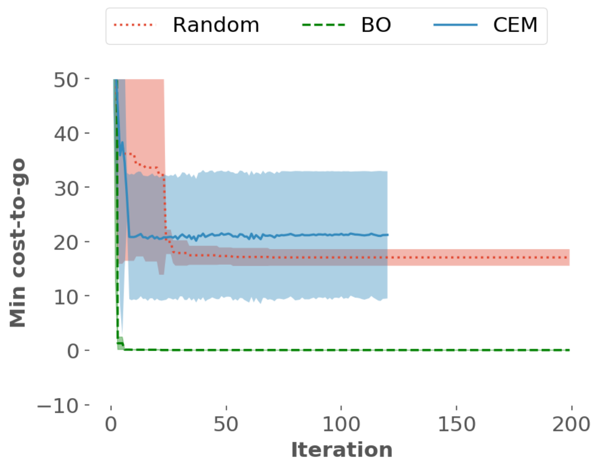

Fig. 4. Minimum cost-to-go per iteration for the case of adversarial scenarios involving pedestrian NPCs. The Bayesian Optimization approach identifies a crash scenario almost immediately, while random and the cross-entropy method (CEM) do not. Note: CEM was terminated after 120 iterations because its parameters converged and its exploration effectively stopped.

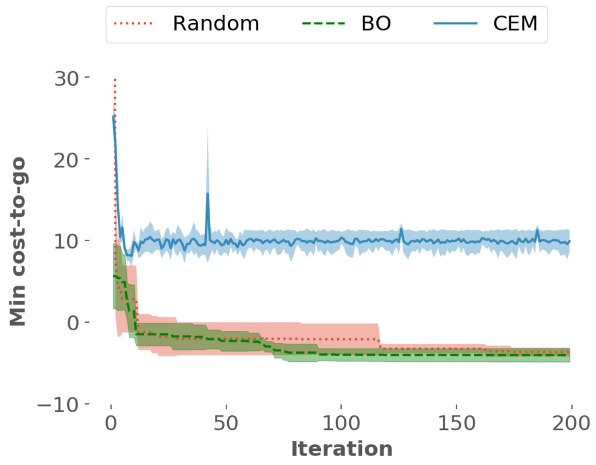

Fig. 5. Minimum cost-to-go per iteration for the case of adversarial scenarios involving vehicle NPCs. The Bayesian Optimization does better on average compared to random search, and outperforms the cross-entropy method.

to do so.

B. Adversarial Vehicle NPC

We setup this experiment with a single adversarial vehicle and our self-driving agent in the same map as in Fig. 2. The adversary is initialized on the opposite lane. Furthermore, we fix all other parameters of the adversarial car, including color, shape, model etc. Only parameters for the generated trajectory are varied. In this case too, Bayesian Optimization does better on average than CEM and random, but its difference from random is not as significant in this setup, as shown in Fig. 5.

We think this is because in general the search space for vehicles in the maps that we have access to (two lanes, straight road) limit the choice for adversarial behavior, so even random search ends up being a reasonable choice for NPC cars. CEM on the other hand seems particularly ill-suited to this problem because it quickly becomes overconfident in its solution and stops exploring, as shown in both Fig. 4 and 5.

C. Mixed Pedestrian and Vehicle Adversaries

We also attempt to identify scenarios based on potential interactions in multi agent settings. We initialize our scenario with a vehicle and a pedestrian. We fix our road-following distance parameters for the vehicle, and remove any waypoints that may cause the pedestrian to directly collide with the vehicle. Our optimization scheme was able to identify scenarios such as the one in Fig. 3, which causes the self-driving agent to collide with the other car, and which requires coordination between the NPC pedestrian and the vehicle.

D. Using Adversarial Scenarios for Imitation Learning

The purpose of generated adversarial scenarios is twofold: on one hand, we can use them for better testing and identification of problematic scenarios. On the other hand, we need to go beyond that and actually incorporate these scenarios in the training procedure of the self-driving policy.

As shown in Alg. 1 we currently do this by identifying dangerous scenarios and then asking an expert, either an optimal control system or a human, to demonstrate what the proper reaction is to each adversarial scenario. We experimented with both types of demonstrations, and in Fig. I we show the end result of visual imitation learning trained according to DAgger [5] on one of those scenarios, where the adversarial vehicle enters the opposite lane, and drives towards the self-driving vehicle. Without having seen a scenario like this the vehicle crashes. After having seen it, it is trained to drive on the empty sidewalk to avoid collision and minimize risk.

To avoid over fitting to given scenario and the expert's commands we also introduce temporally correlated noise to the expert's steering as in [39], [40]. We found that the multiple camera setup as recommended in [41] is not suitable for obstacle avoidance tasks, because automatically annotating the commands of the sideways-looking cameras is challenging. Therefore, we opt for using DAgger [5], where we can have the expert label states that are far from the driving model's training distribution. We found that this lead to successful retraining.

VII. FUTURE WORK

There are many interesting questions that remain open about making this work truly useful for self-driving car research, practical deployment, and testing. First, we need to address scalability to many pedestrians and many vehicles on a large-scale map. Second, we need to find probabilistic parameterizations of the NPC behaviors that encompass both normal scenarios and adversarial ones, and still enable efficient exploration of distinct behaviors. Third, we need to address the issue of exploration in high-dimensions, which in the last few years has seen a lot of progress under reinforcement learning research. Not only are all of these directions of research exciting, but there are many reasons to be optimistic given the current rate of progress in related problems.

VIII. CONCLUSION

In this paper we have shown a new connection between the evaluation of self-driving vehicles and Bayesian Optimization. Our method finds challenging scenarios for self-driving policies in high-fidelity photorealistic simulators; scenarios that would be considered rare events on the road, but nevertheless can cause serious accidents. We propose to automatically discover these adversarial scenarios by parameterizing the behavior of other users of the road, such as other cars and pedestrians. We do this based on the performance of a given self-driving policy, without making any assumptions about its structure being end-to-end trainable or modular and hand-engineered. We show that when these scenarios are incorporated in the training set of a driving policy through imitation learning, they increase its safety.

Automatic exploration and discovery of such challenging scenarios has the potential to drastically improve the reliability of the self-driving software stack, and surprisingly, it is a process that still requires human involvement, even among leading players in the industry. Our results represent a first step towards using active learning and exploration approaches to design better tests and to increase the safety of self-driving vehicles on public roads.

ACKNOWLEDGMENT

The authors would like to thank the Natural Sciences and Engineering Research Council (NSERC) of Canada for financial support.

REFERENCES

[1] A. C. Madrigal, "Inside waymo's secret world for training self-driving cars," *The Atlantic*, Aug 2017. [Online]. Available: https://www.theatlantic.com/technology/archive/2017/08/inside-waymos-secret-testing-and-simulation-facilities/537648/

[2] K. Nidhi and S. M. Paddock, "Driving to Safety: How Many Miles of Driving Would It Take to Demonstrate Autonomous Vehicle Reliability?" https://www.rand.org/pubs/research_reports/RR1478.html, 2016, [Online; accessed 19-July-2018].

[3] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[4] G. Kahn, T. Zhang, S. Levine, and P. Abbeel, "PLATO: policy learning using adaptive trajectory optimization," *CoRR*, vol. abs/1603.00622, 2016.

[5] S. Ross, G. J. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *International Conference on Artificial Intelligence and Statistics, AISTATS*, 2011, pp. 627–635.

[6] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[7] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[8] J. Mockus, "Bayesian approach to global optimization and application to multiobjective and constrained problems," *Journal of Optimization Theory and Applications*, vol. 70, no. 1, pp. 157–172, Jul 1991.

[9] S. Manjanna and G. Dudek, "Data-driven selective sampling for marine vehicles using multi-scale paths," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, Canada, September 2017, pp. 6111–6117.

[10] S. R. Kuindersma, R. A. Grupen, and A. G. Barto, “Variable risk control via stochastic optimization,” *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 806–825, 2013.

[11] E. Brochu, V. M. Cora, and N. de Freitas, “A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning,” *CoRR*, no. arXiv:1012.2599, December 2010.

[12] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, “Information-theoretic regret bounds for gaussian process optimization in the bandit setting,” *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3250–3265, May 2012.

[13] E. Kaufmann, N. Korda, and R. Munos, “Thompson sampling: An asymptotically optimal finite-time analysis,” in *23rd International Conference on Algorithmic Learning Theory*, ser. ALT’12. Springer-Verlag, 2012, pp. 199–213.

[14] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, “Deep exploration via bootstrapped dqn,” in *Advances in Neural Information Processing Systems 29*, 2016, pp. 4026–4034.

[15] K. Azizzadenesheli, E. Brunskill, and A. Anandkumar, “Efficient exploration through bayesian deep q-networks,” *CoRR*, vol. abs/1802.04412, 2018.

[16] Z. C. Lipton, J. Gao, L. Li, X. Li, F. Ahmed, and L. Deng, “Efficient exploration for dialog policy learning with deep BBQ networks,” *CoRR*, vol. abs/1608.05081, 2016.

[17] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg, “Noisy networks for exploration,” *CoRR*, vol. abs/1706.10295, 2017.

[18] A. Kumar, S. Zilberstein, and M. Toussaint, “Probabilistic inference techniques for scalable multiagent decision making,” *Journal of Artificial Intelligence Research*, vol. 53, no. 1, pp. 223–270, May 2015.

[19] K. Rawlik, M. Toussaint, and S. Vijayakumar, “On stochastic optimal control and reinforcement learning by approximate inference,” in *Proc. of Robotics: Science and Systems (R:SS 2012)*, 2012.

[20] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, “Reinforcement learning with deep energy-based policies,” *CoRR*, vol. abs/1702.08165, 2017.

[21] M. Littman, “Markov games as a framework for multi-agent reinforcement learning,” in *International Conference on Machine Learning*, 1994, pp. 157–163.

[22] F. A. Oliehoek, *Decentralized POMDPs*. Springer Berlin Heidelberg, 2012, pp. 471–503.

[23] J. Foerster, R. Y. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, and I. Mordatch, “Learning with opponent-learning awareness,” in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 122–130.

[24] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” *CoRR*, vol. abs/1706.02275, 2017.

[25] N. Papernot, P. D. McDaniel, I. J. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, “Practical black-box attacks against deep learning systems using adversarial examples,” *CoRR*, vol. abs/1602.02697, 2016.

[26] S. H. Huang, N. Papernot, I. J. Goodfellow, Y. Duan, and P. Abbeel, “Adversarial attacks on neural network policies,” *CoRR*, vol. abs/1702.02284, 2017.

[27] Y. Tian, K. Pei, S. Jana, and B. Ray, “Deeptest: Automated testing of deep-neural-network-driven autonomous cars,” in *Proceedings of the 40th International Conference on Software Engineering*. ACM, 2018, pp. 303–314.

[28] F. M. P. Behbahani, K. Shiarlis, X. Chen, V. Kurin, S. Kasewa, C. Stirbu, J. Gomes, S. Paul, F. A. Oliehoek, J. V. Messias, and S. Whiteson, “Learning from demonstration in the wild,” *CoRR*, vol. abs/1811.03516, 2018.

[29] J. Ho and S. Ermon, “Generative adversarial imitation learning,” *CoRR*, vol. abs/1606.03476, 2016.

[30] M. Á. Carreira-Perpiñán and G. E. Hinton, “On contrastive divergence learning,” in *AISTATS*, 2005.

[31] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems 27*, 2014, pp. 2672–2680.

[32] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen, “Differentiable monte carlo ray tracing through edge sampling,” *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, vol. 37, no. 6, pp. 222:1–222:11, 2018.

[33] M. M. Loper and M. J. Black, “Opendr: An approximate differentiable renderer,” in *European Conference on Computer Vision (ECCV)*, 2014, pp. 154–169.

[34] J. Wu, E. Lu, P. Kohli, W. T. Freeman, and J. B. Tenenbaum, “Learning to see physics via visual de-animation,” in *Advances in Neural Information Processing Systems*, 2017.

[35] C. E. Rasmussen, “Gaussian processes in machine learning,” in *Advanced lectures on machine learning*. Springer, 2004, pp. 63–71.

[36] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, “Variational inference: A review for statisticians,” *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[37] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, N. De Freitas *et al.*, “Bayesian optimization in high dimensions via random embeddings.” in *IJCAI*, 2013, pp. 1778–1784.

[38] C. Li, S. Gupta, S. Rana, V. Nguyen, S. Venkatesh, and A. Shilton, “High dimensional bayesian optimization using dropout,” in *International Joint Conference on Artificial Intelligence*, 2017, pp. 2096–2102.

[39] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, “End-to-end driving via conditional imitation learning,” in *International Conference on Robotics and Automation (ICRA)*, 2018.

[40] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg, “Dart: Noise injection for robust imitation learning,” in *Conference on Robot Learning*, vol. 78. PMLR, 13–15 Nov 2017, pp. 143–156.

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