Zero-Shot Policy Transfer in Autonomous Racing: Reinforcement Learning vs Imitation Learning

Nathaniel Hamilton†
Department of Electrical and Computer Engineering
Vanderbilt University
Nashville, TN, USA
nathaniel.p.hamilton@vanderbilt.edu

Patrick Musau†
Department of Electrical and Computer Engineering
Vanderbilt University
Nashville, TN, USA
patrick.musau@vanderbilt.edu

Diego Manzanas Lopez
Department of Electrical and Computer Engineering
Vanderbilt University
Nashville, TN, USA
diego.manzanas.lopez@vanderbilt.edu

Taylor T. Johnson
Department of Computer Science
Vanderbilt University
Nashville, TN, USA
taylor.johnson@vanderbilt.edu

***Abstract*—There are few technologies that hold as much promise in achieving safe, accessible, and convenient transportation as autonomous vehicles. However, as recent years have demonstrated, safety and reliability remain the most obstinate challenges, especially in complex domains. Autonomous racing has demonstrated unique benefits in that researchers can conduct research in controlled environments, allowing for experimentation with approaches that are too risky to evaluate on public roads. In this work, we compare two leading methods for training neural network controllers, Reinforcement Learning and Imitation Learning, for the autonomous racing task. We compare their viability by analyzing their performance and safety when deployed in novel scenarios outside their training via zero-shot policy transfer. Our evaluation is made up of a large number of experiments in simulation and on our real-world hardware platform that analyze whether these algorithms remain effective when transferred to the real-world. Our results show reinforcement learning outperforms imitation learning in most scenarios. However, the increased performance comes at the cost of reduced safety. Thus, both methods are effective under different criteria.**

***Index Terms*—imitation learning, deep reinforcement learning, sim2real, autonomous racing, zero-shot policy transfer**

I. INTRODUCTION

Autonomous Racing is a growing topic of interest, ranging from small-scale academic competitions (e.g. F1/10 [1]) to full-scale competitions (e.g. Roborace, AWS DeepRacer [2] and the Indy Autonomous Challenge [3]). These racing competitions are integral to the development of *Autonomous Vehicles* (AVs) as they help promote general confidence and societal acceptance of a novel emerging technology. Moreover, they allow researchers to conduct explorations of possible solutions to difficult scenarios such as high-speed obstacle avoidance and other risky maneuvers that may be too dangerous to consider in urban settings [4].

Within this realm, one classical approach of constructing these systems involves a decomposition of tasks into four main areas: perception, planning, control, and system supervision [5]. Confining our focus to the control of these vehicles, many platforms favor classical or model predictive control techniques for their predictably safe performance. However, in recent years, many researchers have proposed the use of machine learning for control tasks, as these methods have shown significant potential in solving optimal control problems for highly nonlinear systems with varying degrees of uncertainty [5]. This prowess has made these types of regimes particularly attractive for autonomous vehicle development.

One of the most successful frameworks for solving machine learning control problems has been *Reinforcement Learning* (RL). RL is a branch of machine learning that focuses on software agents learning to maximize rewards in an environment through experience. The general idea is similar to training a dog to do tricks by giving it treats when it performs the desired task. Thus, an optimal controller can be synthesized using data evaluated by key performance criteria through trial and error [6]. Many RL approaches leverage neural networks due to their advantages in dealing with complex data. These approaches can be referred to as *Deep Reinforcement Learning* (also referred to as RL) techniques, and recent successes such as OpenAI's *OpenAI Five* outperforming pro-level players at Dota 2 [7], and Microsoft's *MuZero* [8] mastering Atari, Go, Chess and Shogi have helped bring RL to the forefront of AI discussion.

Despite their success in numerous realms, RL approaches, can be costly to train, especially as systems become more complex and dynamic. Additionally, RL allows agents to learn via trial and error, exploring *any behavior* during the learning process. In many realistic domains, this level of freedom is unacceptable, thus training in simulation is standard. Therefore, the challenge becomes how to minimize the inherent mismatches between real-world settings, and the simulation environments used to train RL agents [9].

† These authors contributed equally

Training agents in simulation and then deploying them on real-world hardware platforms, known as a *sim2real* transfer, is a challenging problem. In many cases, the agents do not perform as expected in the real world, sometimes resulting in unsafe or catastrophic behavior [10], [11]. Their performance can be improved with further training in the new environment, but that is only possible if the behavior policy is safe from the outset. Transferring a learned policy and evaluating before any additional training is done is referred to as a *zero-shot policy transfer*.

In this work we focus on zero-shot policy transfer since active learning, i.e. learning during evaluation, is impractical for real-time systems because updates to the neural network control policy are computationally expensive and time-consuming. Instead, we evaluate trained policy networks as they are. This is standard practice in industry, to deploy a trained model and release updates intermittently.[1]

One way to achieve high performance with a zero-shot policy transfer is by leveraging external or expert knowledge. *Imitation Learning* (IL) utilizes expert demonstrations to train an agent to mimic the behavior. Using IL, an agent can be trained to mimic a human or a complicated array of computationally intensive classical control methods that perform optimally in different scenarios. In this way, complicated algorithms and/or human experience can be boiled down to one neural network capable of replicating their behaviors.

While the last several years have witnessed a significant number of approaches for addressing these challenges, there have been few in-depth empirical studies comparing the efficacy of different learning frameworks for learning robust agent behavior [12]. In [12], Gros et al. note that RL approaches generally outperform IL. However, this performance comes at a cost of significant reward shaping. While this work provides an enlightening discussion, the authors consider only discrete environments and do not address *sim2real* challenges.

In light of the lack of empirical comparisons of IL and RL, in this work, we experiment with and compare *Neural Network Controllers* (NNCs) trained using these approaches for the control of a 1/10 scale autonomous vehicle. The performance of these trained NNCs are compared through a number of experiments, testing their ability to handle scenarios outside their training environment via zero-shot policy transfer. These experiments include changing the vehicle's constant speed, adding unknown obstacles to the track, and evaluating on different tracks. These experiments culminate in a *sim2real* transfer and evaluation of the controllers on our hardware platform.

In summary, the contributions of this paper are:

1) We train a NNC using IL to imitate a path following algorithm that effectively balances efficiency and safety.
2) We train 2 NNCs using state-of-the-art RL algorithms, DDPG and SAC.
3) We compare their performance in a series of zero-shot policy transfer experiments in simulation.
4) We compare their performance in a *sim2real* zero-shot policy transfer experiment.

[1]The rate at which these updates occur depends highly on the application.

II. BACKGROUND

A. Imitation Learning

Imitation learning seeks to replicate the behavior of a human or other expert on a given task [13], [14]. These approaches fall within the field of *Expert Systems* in Artificial Intelligence, and in recent years the demand for these approaches has increased substantially. The surge in interest is spurred on by two main motivations. (1) The number of possible actions needed to execute a complex task is too large to cover by explicit programming. (2) Demonstrations show that having prior knowledge provided by an expert is more efficient than learning from scratch [13].

In this work, we employ one of the most common methods of IL, *Behavior Cloning*, which was first introduced to train a modified van to navigate paths at speeds up to 20 miles per hour [15], [16]. The work was later replicated with an updated convolutional neural network architecture in [17] with great success.

B. Reinforcement Learning

Reinforcement learning seeks to find the optimal behavior function for completing a given task through experimental trials. An agent converges on this optimal behavior function, or learned policy $a = \pi(s)$, by learning what results from executing action a when in state s. The result is the next state, s', and a reward, r, determined by a given reward function. This information is stored as a tuple, $\{s, a, r, s'\}$, often referred to as an *experience*. In this work we utilize two well-known, state-of-the-art off-policy deep reinforcement learning algorithms *Soft Actor-Critic* (SAC) [18], and its predecessor *Deep Deterministic Policy Gradient* (DDPG) [19].

C. F1/10

For our experiments, we utilize the F1/10 simulation and hardware platform [1]. The platform was designed to replicate the hardware and software capabilities of full scale autonomous vehicles. The hardware platform is equipped with a standard suite of sensors including stereo cameras, LiDAR (light detection and ranging), and inertial measurement units (IMU). The car is controlled by an NVIDIA Jetson TX2, and its software stack is built on the *Robot Operating System* (ROS) [20]. In the Gazebo simulation environment, all the sensors are replicated so the transition from simulation to the real-world and back is straightforward without hours worth of re-configuring.

III. EXPERIMENTAL SETUP

In order to make the comparisons as fair as possible, all the controllers we trained have the same neural network architecture and are trained on the *Porto* track shown in Fig. 2 unless otherwise specified. The trained NNCs selected for our experimental evaluations are the best performing of at least 3 NNCs trained the same way using different random

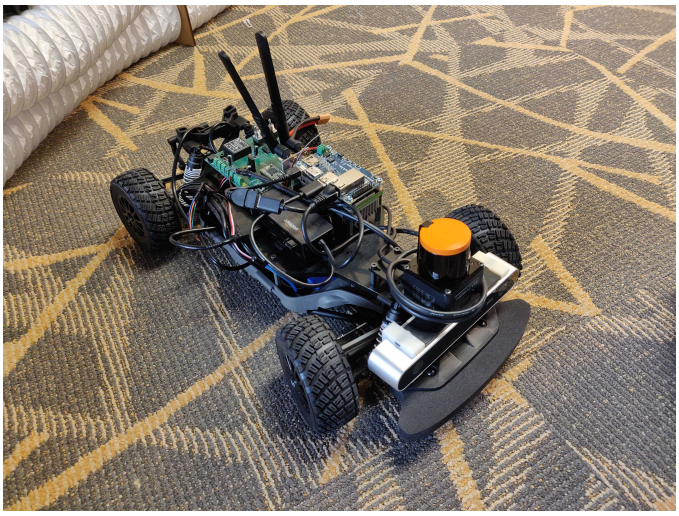

Fig. 1. Visualization of our experimental F1/10 hardware platform. This platform is a one-tenth scale RC car that has been altered to operate autonomously with the support of a sensor and compute architecture for autonomous decision-making [4].

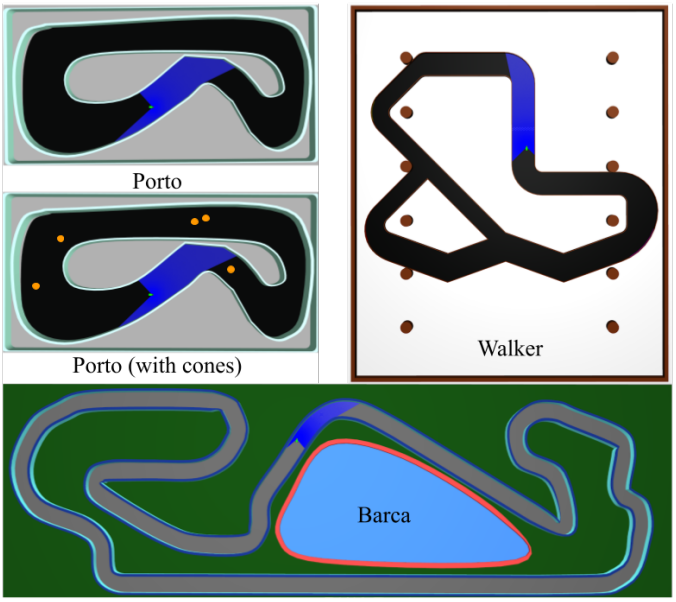

Fig. 2. The different tracks and the corresponding starting positions we used in our simulation experiments. The bright green rectangle is the simulated car, and the blue region around it represents the full set of range values collected by the LiDAR sensor.

seeds[2]. Additionally, the control output has been limited to only steering and the car travels at a constant speed of $1m/s$ during training.

A. Neural Network Architecture

In this work, we utilize a common architecture found in RL work. The simple multi-layer perceptron network consists of an input layer, 2 fully connected hidden layers of 64 nodes with ReLU activation functions, and a fully connected output layer with a hyperbolic tangent, $tanh$, activation function. The input layer accepts nine range values collected from the LiDAR at $-90°$, $-60°$, $-45°$, $-30°$, $0°$, $30°$, $45°$, $60°$, and $90°$ from forward. The range values are clipped between $[0m, 10m]$. The output layer provides a single value between $[-1, 1]$, which is scaled up linearly for the desired steering angle between $[-34°, 34°]$.

B. Training the Agents

1) Imitation Learning: We trained the imitation learning agent using a procedure that is a simplification of the seminal work by Dean Pomerleau, in which a neural network was trained to control an autonomous vehicle [16]. The agent in this work was trained on sensor-action pairs collected during experiments where the vehicle was controlled using a path following algorithm on the racetrack. The path we used for training lead around the middle of the track, ensuring safe operation. The path following algorithm we utilized, *Pure Pursuit* [23], does a quick search for a waypoint that it can safely reach governed by a specified look-ahead horizon, it then steers the car towards that waypoint. The pure pursuit algorithm has been used in numerous contexts and has been shown to be a robust method for efficiently and accurately following a path. This was our main motivation in using this controller.

The first imitation learning agent, IL, was trained only on data collected from the *Porto* track. This makes the training process more like what the RL agents will see, since they are also only trained on the *Porto* track. The second agent, IL-3, was trained using data collected from the *Porto* track as well as the two other tracks, *Walker* and *Barca* shown in Fig. 2. We include IL-3 to highlight one of the main advantages of using IL to train NNCs: any recorded data of the expert can be used for training.

2) Deep Reinforcement Learning: Both RL controllers, DDPG and SAC, were trained using common hyperparameters, which are provided in the Appendix. The agents optimize performance according to a dense reward function that assigns a positive reward for counterclockwise progress around the track. The reward is calculated using a reference path that runs through the middle of the track. The value of the reward is the positive arc length between the previous and current closest point along the path. This reward function encourages the agent to complete as many laps as possible as quickly as possible.

[2]The random seed used for training has a large impact on the training process and resulting policy, as demonstrated in [21], [22]

We trained the agents according to their respective algorithms. We halted the training process to evaluate performance after every 500 training steps. The performance is measured by how many laps the agent can complete within 100 seconds. This is more than enough time to complete 2 laps in the training track (*Porto*). We chose 2 laps because completing 1 lap is not enough to show the controller is capable of completing multiple laps. The car always starts in the same position, but may not return to the same position at the end of the first lap. However, the starting position of laps 2+ will be about the same. Thus, if the controller is able to complete 2 laps, it is likely capable of completing any number of laps.

The evaluation is repeated up to 10 times, and training stops when the agent is able to complete at least 2 laps 10 times in a row. Once the agent is able to complete at least 2 laps 10 times, the training process is halted and control policy is saved for our experiments.

C. Evaluating Performance

We evaluate the controllers through a variety of scenarios that test their ability to maintain optimal performance in scenarios outside their training environment. These scenarios include changing the constant speed value, adding obstacles to the track, evaluating on a different track, and a real-world evaluation on our hardware platform. We compare the performance of the controllers according to three metrics we refer to as *track distance*, *efficiency*, and *safety*.

Efficiency is calculated as the distance the car travels around the track divided by the amount of time it took to get there. Each test runs for a maximum of 60 seconds and cuts off sooner if the car collides with a wall or obstacle. We refer to this as a measure of efficiency because the distance is not measured by the direct distance the car traveled. Instead, the distance is measured in relation to the arc length of a path going through the center of the track, which we refer to as the *track distance*. The closer the car stays to following the center path, the closer the efficiency value will match the constant speed. However, if the car takes sharp turns around the corners, the efficiency will increase since the car covers the same track distance in less time.

Safety is a measure of how prone to collisions the controller is at a specific track. The safety value corresponds to the percentage of runs that ended with no collision regardless of the time or distance traveled, i.e. if safety = 100%, there were no collisions encountered in the experiments.

IV. EXPERIMENTS AND RESULTS

Our experiments were designed to test the performance of both the RL and IL controllers in challenging scenarios. The first experiment demonstrates the ideal test conditions, evaluating in the same environment the controllers were trained in. The following three experiments introduce changes to the environment that test the robustness of the learned control policies, building up towards the final experiment, deploying on the real-world hardware platform.[3]

All simulation experiments test each controller 30 times in the designated scenario. Each test lasts for a maximum of 60 seconds, stopping early in the event of a collision.[4]

A. Training Environment (Porto)

Our first experiment evaluates the performance of the controllers in the environment they were trained in. This provides a baseline that we can compare to as we test these controllers in scenarios outside their training. The results in Table I show all the controllers operate safely without any recorded collisions. Additionally, the results show both RL controllers operate more efficiently and travel further than the IL controllers.

B. Varying Speed

In our second experiment, we explore how changing the constant speed of the car impacts performance. This subtle change tests the robustness of the controllers with respect to a change in speed. The control policies were trained with the assumption the car moves at $1.0m/s$. Moving at different speeds, especially faster than expected, might reveal unsafe behaviors. Additionally, this experiment provides some insight into how well the controllers will handle a *sim2real* transfer. Unlike in simulation, the hardware platform can experience fluctuations in speed caused by a poorly-tuned speed regulator, wheel slippage, etc.

We tested the controllers on the *Porto* track with constant speeds $0.5m/s$ and $1.5m/s$. We expected the efficiency and track distance of the controllers to be cut in half when run at half speed. We also expected the controllers would remain safe at half speed. For the tests at a faster speed, we expected the efficiency to increase by a factor of 1.5, but experience more collisions.

The results in Table II show that cutting the speed in half leads the efficiency and track distance to be reduced by about half for every controller. Since the efficiencies and recorded track distances of the controllers at $0.5m/s$ are slightly above the expected half, the controller's efficient behaviors are more impactful at slower speeds. Every controller except for SAC maintained their safe performance. In the one trial that the SAC controller collided with the wall, it was during the first left turn. The controller turned too early while driving close to the wall, resulting in a collision.

Furthermore, the results in Table II show that increasing the speed reduces the safety of all the controllers. In our experiments, none of the controllers were safe for all evaluated runs. In particular, DDPG was unable to complete any runs without colliding after the first curve. However, despite the increase in collisions, all the controllers operated more efficiently. IL, IL-3, DDPG, and SAC saw a $1.5x$, 1.45, $1.12x$, and $1.39x$ increase respectively. The IL controller was the only one able

[3]The hardware experiments are summarized at: **https://youtu.be/rgVb46RMMvE**

[4]A video summarizing the simulation experiments can be found at: **https://tinyurl.com/2bjwpcxs**

TABLE I
PERFORMANCE ON PORTO WITH AND WITHOUT OBSTACLES

	No Obstacles			Obstacles		
Algorithm	Track Distance	Efficiency	Safety	Track Distance	Efficiency	Safety
IL	121.44 ± 14.41	1.94 ± 0.25	100%	41.80 ± 0.88	1.93 ± 0.02	0%
IL-3	125.23 ± 0.31	2.01 ± 0.00	100%	40.79 ± 0.26	1.92 ± 0.02	0%
DDPG	142.74 ± 10.52	2.29 ± 0.16	100%	40.33 ± 0.37	2.23 ± 0.03	0%
SAC	144.04 ± 0.47	2.31 ± 0.01	100%	182.98 ± 2.92	2.94 ± 0.01	96.66%

TABLE II
PERFORMANCE ON PORTO VARYING CONSTANT SPEED

	0.5 m/s			1.0 m/s			1.5 m/s		
Algorithm	Track Distance	Efficiency	Safety	Track Distance	Efficiency	Safety	Track Distance	Efficiency	Safety
IL	64.73 ± 0.36	1.04 ± 0.01	100%	121.44 ± 14.41	1.94 ± 0.25	100%	171.24 ± 41.24	2.90 ± 0.06	93.33%
IL-3	64.60 ± 0.09	1.04 ± 0.00	100%	125.23 ± 0.31	2.01 ± 0.00	100%	178.54 ± 20.94	2.92 ± 0.03	96.67%
DDPG	73.60 ± 0.22	1.18 ± 0.00	100%	142.74 ± 10.52	2.29 ± 0.16	100%	18.09 ± 0.29	2.57 ± 0.08	0%
SAC	72.97 ± 10.05	1.20 ± 0.03	93.33%	144.04 ± 0.47	2.31 ± 0.01	100%	174.11 ± 62.58	3.21 ± 0.15	80.0%

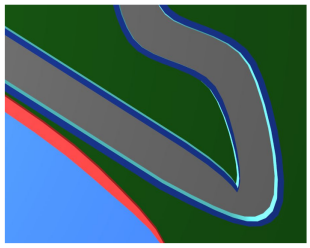

Fig. 3. Difficult sharp turn on Barca track.

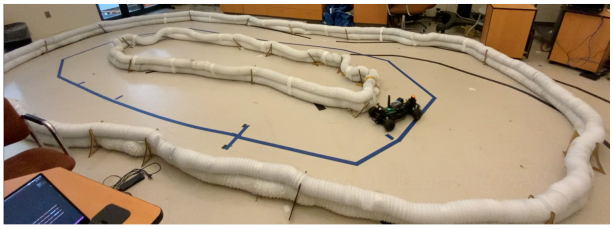

Fig. 4. Our real-world track with the reference path used for measuring the distance traveled marked in blue.

to meet the $1.5x$ increase we expected to match the speed increase.

C. Obstacles

Our third experiment introduces unknown obstacles, orange traffic cones, to the *Porto* track as shown in Fig. 2. This experiment tests the controllers beyond what they were trained to do. Not only does the controller have to steer the car along the optimal path while avoiding the walls, there are now additional obstacles to avoid. Thus, it provides a measure of each controller's ability to mimic the driving task, rather than robust pattern matching. The IL controllers failed to generalize to this scenario, and failed to complete a single lap without a collision failing around the last cone. DDPG was similar in nature, however, it maintained its higher level of efficiency over the IL controllers. SAC was the only controller able to handle obstacles and successfully navigated the cones in 96.66% of our evaluations. Interestingly, the obstacles improved SAC's performance. The last cone on the track was positioned just right to direct the controller to steer sooner, finding a more optimal path.

D. Alternate Race Tracks (Walker and Barca)

In our fourth experiment, we examined how well the controllers perform when used on two different, more complicated tracks, *Walker* and *Barca* shown in Fig. 2. *Walker* introduces a choice between two paths, which we anticipated would cause issues because none of the controllers, except IL-3, have experience with that scenario. We also anticipated that *Barca*'s long straightaways and sharp turns would cause more collisions for controllers trying to cut corners. The results for this experiment, and the lengths of the tracks for comparison, are shown in Table III

On the *Walker* track, both the IL and SAC controllers were unable to consistently navigate the junction. Instead of picking a direction to pursue, the IL controller drove the car directly into the corner of the junction in every test while the SAC controller managed to avoid that fatal mistake in some cases, but rarely passed it in the second lap. In contrast, the DDPG controller was able to successfully navigate the divergent track without prior experience. Additionally, both RL controllers navigated the track more efficiently than the IL-3 controller, which had prior experience on the track.

On the *Barca* track, only the IL controllers were able to safely navigate the sharp turn highlighted in Fig. 3. Both DDPG and SAC collide with the track wall at the sharp turn by either turning too soon or not turning at all.

E. Real-World, Hardware Platform

In our final experiment, we test how well these controllers handle an actual *sim2real* transfer on our hardware platform. The experiments were conducted on our track, shown in Fig. 4, which has a middle-of-the-track path length of $13.08m$. Because we could not reliably record the time for runs that resulted in a collision, we do not compare the controllers' efficiency. Instead, we compare the distance traveled around

TABLE III
PERFORMANCE ACROSS DIFFERENT RACETRACKS

	Porto ($57.5m$)			Walker ($73.25m$)			Barca ($221.14m$)		
Algorithm	Track Distance	Efficiency	Safety	Track Distance	Efficiency	Safety	Track Distance	Efficiency	Safety
IL	121.44 ± 14.41	1.94 ± 0.25	100%	33.50 ± 1.55	1.89 ± 0.02	0%	108.54 ± 33.26	1.96 ± 0.03	83.33%
IL-3	125.23 ± 0.31	2.01 ± 0.00	100%	123.38 ± 0.33	1.98 ± 0.01	100%	124 ± 0.29	2.00 ± 0.00	100%
DDPG	142.74 ± 10.52	2.29 ± 0.16	100%	130.08 ± 0.34	2.09 ± 0.00	100%	31.54 ± 0.03	1.80 ± 0.01	0%
SAC	144.04 ± 0.47	2.31 ± 0.01	100%	62.64 ± 38.31	2.11 ± 0.25	0%	24.27 ± 4.09	1.76 ± 0.02	0%

TABLE IV
PERFORMANCE ON HARDWARE PLATFORM

Algorithm	Track Distance	Bumps	Collisions
IL	53.86 ± 0.47	0	0
IL-3	5.81 ± 0.00	0	10
DDPG	2.00 ± 0.00	0	10
SAC	61.83 ± 0.37	4.7 ± 0.67	0

the track in 60 seconds. We averaged the results across 10 runs and kept a count of how often the controllers drove the car along the side of the track, bumping into it (Bump), as well as how many times it drove directly into the side of the track (Collision). We halted the run in the event of a collision and recorded the final position as the total distance traveled. While bumps in our simulated results counted as collisions, we decided to allow them in the hardware experiments because they did not harm the track and would have been allowed in the F1/10 competition.

The results in Table IV show DDPG and IL-3 were unable to complete a lap, instead colliding with the side of the track before completing the first turn. However, both IL and SAC were able to complete over 4 laps in the allotted 60 seconds.

V. DISCUSSION

Our experiments highlight two main challenges to the *sim2real* problem, *model mismatch* and *domain mismatch*. Model mismatch centers around the output not having the expected outcome. We highlight this challenge in our experiments with varying speed. Domain mismatch centers around the input being out of scope, or not what is expected. In other words, the real inputs do not match the training inputs. We highlight this challenge in our experiments with the obstacles and alternate racetracks. In this section, we discuss which controllers handled each type of mismatch best and theorize why that might be the case.

A. Model Mismatch

Model mismatch is a result of the output not having the expected outcome. This could be the result of noisy actuators, inaccurate model dynamics, etc. In our experiments, we highlight this challenge by testing the controllers at varying speeds in Table II.

Our results show the IL controllers handled this challenge better than the RL controllers. We attribute this result to how the controllers were trained. The IL controllers were trained to imitate the behavior of our expert control that balanced safety and efficiency by trending towards the middle of the track. In contrast, the RL controllers were trained solely to optimize efficiency. As a result, the RL controllers cut corners sharply and drove close to the walls. Because of this, changes to the speed had a larger impact on safety. Turning close to the walls has a smaller margin for error than turning in the middle of the track. Despite this greater challenge, the SAC controller was almost as safe as the IL controllers, with much better performance. If we compare the track distance of only safe trials, the SAC controller traveled an average of $201.25m$ and the IL controllers traveled an average of $182.3m$. The difference between the two is about $1/3$ of a lap.

B. Domain Mismatch

Domain mismatch is a result of the input data not matching the training input. This could be the result of noisy sensors, unexpected obstacles, or a change in environment. In our experiments, we highlight this challenge in or experiments by introducing obstacles (Table I) and testing on alternate racetracks (Table III).

For this challenge, there is not a clear victor since the results were more varied. The IL-3 controller performed well across all the racetracks, but failed at obstacle avoidance. The SAC controller, on the other hand, successfully avoided colliding with obstacles in almost all our tests, but struggled when evaluated on alternate racetracks.

C. sim2real

The experiments on our hardware platform help emphasize why *sim2real* is such a challenging problem. While the IL-3 controller maintained performance across all three racetracks and was the safest controller when we varied the speed, it failed to complete a single lap in the real world. Meanwhile, the IL controller, which had similar results with varied speed but struggled more on the different racetracks, successfully navigated our real world track. Because the IL-3 controller failed despite the IL controller's success, we theorize IL-3's failure was a result of overfitting. Overfitting occurs when the learned policy too closely or exactly matches the training data, and fails to generalize well to new data reliably. Training the IL-3 controller across multiple racetracks helped it perform well on all three tracks and improved its performance on *Porto*. However, all the extra training data in simulation, across varied racetracks, caused the controller to overfit to the simulation domain where the car can safely maintain a $1m$ distance from the left wall without colliding.

The varied training data that negatively impacted the IL-3 controller is likely what caused the SAC controller to succeed.

While DDPG and SAC are similar RL approaches, they differ greatly in how they collect training data. In DDPG, new data is collected by adding random noise to the output of the learned policy. As the learned policy improves, the data collected starts to repeat. This repetition can cause undesirable effects on the learned policy, like *catastrophic forgetting* [24]. In contrast, SAC collects new data using an entropy maximizing function. This means that throughout the training process, new, unique, and varied data is prioritized. The result is a more robust learned policy with optimal performance.

D. Lessons Learned

1) Reinforcement Learning vs Imitation Learning: From the data and observations we collected throughout the training and evaluation processes, we found that reinforcement learning has a greater potential to learn robust and optimal control policies. However, the potential is lost without a well-defined reward function. Since RL focuses solely on optimizing performance, when we changed the track, many of the optimal performance strategies backfired and lead the car into collisions. We expect that this problem could be mitigated if we defined a reward function that incorporated an additional aspect, like a punishment for moving away from the center of the track. The result would be a more robust control policy that avoids colliding with walls, even in new tracks.

On the other hand, IL is still a valuable method, particularly when creating a well-defined reward function is not possible. However, one of the main challenges with imitation learning lies in synthesizing a dataset that allows the agent to truly mimic the expert behavior. Although the training regime for these approaches resembles standard supervised learning regimes, the i.i.d assumption may no longer be valid [25]. Often, the current state of the system prompts the next state. Thus, if the agent makes a mistake in carrying out an action, it may eventually reach a state that the agent has never been trained on. For example, if the training data only contained state-action pairs where the agent was following a path in the center of the track, any deviation from this path could result in states outside the training data and suboptimal actions that lead the car straight into a wall. Therefore, while imitation learning is extremely effective in numerous applications, it can also fail spectacularly, like shown in our *sim2real* experiments.

2) Low Error is Not Necessarily a Good Indicator of Success: One commonly held principle within machine learning is that accuracy alone is generally a poor measure of evaluating a model's performance. In classification tasks, this can be addressed by using a metric such as an F1-Score, which balances the precision and recall of a model. However, it is not as straightforward for imitation learning tasks. In our experiments, we utilized mean-squared error to measure the effectiveness of our controllers. Curiously, some of the models that had very low error-rates, both on the test and validation set, could not complete a single lap. While other models, with a lower measured performance, did better on the driving task. This illustrates the need for better metrics for evaluating imitation learning tasks. There has been a large body of work towards this end over the last several years [26].

3) General Recommendations: We recognize that it is difficult to issue broad recommendations on a limited set of experiments. However, we believe the observations we made will translate to other platforms. Thus, we propose the following suggestions for those who wish to apply these techniques to other platforms:

- In general, we believe that RL approaches will fare better at *sim2real* tasks, since their inspiration is more conducive to exploring a wide range of state-action pairs than those considered in behavior cloning paradigms. However, reward shaping for these approaches is still extremely challenging. Therefore, one needs to weigh the cost of reward shaping against synthesizing expansive datasets for imitation learning models.
- Our experiments did not evaluate training or fine-tuning models in the real world. This choice was motivated by a desire to ensure fairness in the evaluation process between the two approaches. While it is straightforward to train imitation learning models on real-world data, training RL approaches in the real world remains a challenge within the machine learning literature [27]. Our future work would like to consider an analysis of training and/or fine-tuning RL- and IL-trained models in the real world.

While imitation learning and reinforcement learning approaches are not widely used within production-ready, state-of-the-art autonomous vehicles, they have enjoyed significant success within industrial robotics applications. One such example of this success, is the rise of robotics companies leveraging these approaches, such as Alphabet's Intrinsic AI, Veo Robotics, Symbio, and Covariant. Still, there are few works comparing the success of imitation learning versus reinforcement learning approaches within these contexts. This work serves to motivate these types of studies in the community at large.

VI. Related Work

There is a large body of work developing methods to improve reinforcement learning and overcome its shortcomings. These methods include ways to cut back on costly data collection and training time, reduce over-specialization, and improve the safety of the system during and after training is over. In this section, we highlight some of the promising methods we found in the literature. For an in-depth overview of how RL is being used in autonomous driving, we recommend Kiran et al.'s 2021 survey [28].

A. Offline Reinforcement Learning and Inverse Reinforcement Learning

Offline Reinforcement Learning, which is best described in [29], and Inverse Reinforcement Learning [30], are both similar to a combination of imitation learning and reinforcement learning.

Like IL, the training data is collected once and goes unaltered during the training process. Additionally, the agent does

not interact with the environment at all during the training process until it is deployed after training is complete. This method is very beneficial to settings where data collection is slow, expensive, and/or dangerous like in robotics, autonomous driving, or healthcare.

The key aspect that allows both of these approaches to perform better than the policy used to collect the data is the use of a reward function. The reward function allows the agent to better infer what should be done in unexplored states, guiding the agent to perform optimally. In Offline RL, the reward function is known, but in Inverse RL, the reward function is inferred from observing expert behavior.

B. Meta Reinforcement Learning

Meta Reinforcement Learning (Meta-RL), best represented by model-agnostic meta-learning [31] and RL^2 [32], seeks to reduce over-specialization by training across multiple environments. In addition to making the agent more robust, the agent is able to learn to solve new tasks quickly. Some promising works in the area include *Joint PPO* [33] and *POET* [34].

C. Safe Reinforcement Learning

Safe Reinforcement Learning is grouped into two main styles of approach, (1) modification of the optimality criterion and (2) modification of the exploration process [35].

The first, often referred to under the broader term *reward shaping*, uses cleverly designed reward functions that incorporate risk in order to discourage unsafe behavior during training and ensure they avoid those unsafe behaviors when deployed [36].

The second style, often referred to simply as *safe exploration*, leverages external knowledge in the form of a *safety monitor*, a *shield*, *control barrier functions* (CBF), or some other form of *runtime assurance* (RTA) to ensure the agent remains safe while training [37]–[41].

D. Runtime Assurance

The most effective way to ensure safety after training, no matter the learning algorithm, is with *Runtime Assurance* (RTA). Especially in safety critical settings like autonomous driving, it is imperative that system designers prevent catastrophic failures that can result from biased or limited training data [42]. In recent years, numerous RTA approaches have been proposed, ranging from approaches that are statistical in nature [43]–[47], to more rigorous formal proof regimes [48]–[54]. Formally demonstrating the correctness of modern machine learning models is a difficult task that often suffers from the well-known state explosion problem [55]. While there has been a recent influx of formal methods capable of being run in real-time, statistical methods are the current leaders at circumventing scalability issues, though without the formal guarantees.

VII. FUTURE WORK AND CONCLUSIONS

In this work, we experimented with neural network controllers trained using imitation and reinforcement learning to compete in autonomous racing. We compared how the trained networks performed in new scenarios via zero-shot policy transfers. These scenarios tested the controllers' performance despite changes made to the operation of the vehicle and the track it was racing on. These changes were then combined by testing the controllers on our hardware platform.

The results show the RL controllers had more efficient performance even in new environments. SAC in particular was robust to the introduced static obstacles as well as the *sim2real* transfer. However, the RL controllers' more efficient performance led to more collisions. Therefore, unless work is done to train the RL controller to account for safety constraints, IL should be considered a competitive option for scenarios like this.

In future work, we would like to explore how the input space impacts performance by conducting the same experiments in this work with a new neural network architecture that utilizes convolutional layers. Cameras are a standard sensor on most autonomous vehicles due to their ability to sense color and other fine-grained details in the environment. This makes them particularly useful for tasks such as traffic light recognition, and identifying possible road work. Moreover, there are numerous, remarkable machine learning algorithms within the computer vision community that can deal with images at high levels of accuracy.

ACKNOWLEDGMENT

This material is based upon work supported by the Air Force Office of Scientific Research (AFOSR) under award number FA9550-22-1-0019, the National Science Foundation (NSF) under grant numbers 1918450, 1910017, and 2028001, the Department of Defense (DoD) through the National Defense Science & Engineering Graduate (NDSEG) Fellowship Program, and the Defense Advanced Research Projects Agency (DARPA) Assured Autonomy program through contract number FA8750-18-C-0089. Any opinions, finding, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force, DARPA, nor NSF.

APPENDIX

IL and IL-3 Hyperparameters:

- Network Architecture : (64, relu, 64, relu, tanh)
- Optimizer: Stochastic Gradient Descent, Nesterov Momentum
- Learning Rate (LR): 0.01
- Decay: 0.002
- Epochs: 100
- Loss: Mean Average Error

DDPG Hyperparameters:

- Policy Network (Actor): (64, relu, 64, relu, tanh)
- Q Network (Critic): (64, relu, 64, relu, linear)
- Actor LR: 0.0001
- Critic LR: 0.001
- Noise type: Ornstein-Uhlenbeck Process Noise $\sigma = 0.3$, $\theta = 0.15$

- Soft target update: $\tau = 0.001$
- $\gamma = 0.99$
- Critic L2 reg: 0.01
- buffer size: 10^6
- batch size: $B = 64$
- episode length: $T = 500$
- maximum number of steps: 45000

SAC Hyperparameters:

- Policy Network (Actor): (64, relu, 64, relu, tanh)
- Q Networks (Critics): (64, relu, 64, relu, relu)
- learning rate: 0.0001
- Soft target update: $\tau = 0.001$
- $\gamma = 0.99$
- $\alpha = 0.01$
- buffer size: 10^6
- batch size: $B = 64$
- episode length: $T = 500$
- maximum number of steps: 45000

REFERENCES

[1] M. O'Kelly, V. Sukhil, H. Abbas, J. Harkins, C. Kao, Y. V. Pant, R. Mangharam, D. Agarwal, M. Behl, P. Burgio, and M. Bertogna, "F1/10: an open-source autonomous cyber-physical platform," *CoRR*, vol. abs/1901.08567, 2019.

[2] B. Balaji, S. Mallya, S. Genc, S. Gupta, L. Dirac, V. Khare, G. Roy, T. Sun, Y. Tao, B. Townsend, E. Calleja, S. Muralidhara, and D. Karuppasamy, "Deepracer: Educational autonomous racing platform for experimentation with sim2real reinforcement learning," *CoRR*, vol. abs/1911.01562, 2019. [Online]. Available: http://arxiv.org/abs/1911.01562

[3] D. Murphy, "Tum autonomous motorsport wins the indy autonomous challenge powered by cisco at the indianapolis motor speedway and the $1 million grand prize," Oct 2021. [Online]. Available: https://www.indyautonomouschallenge.com/tum-autonomous-motorsport-wins-the-indy-autonomous-challenge-powered-by-cisco-at-the-indianapolis-motor-speedway-and-the-1-million-grand-prize

[4] M. O'Kelly, H. Zheng, D. Karthik, and R. Mangharam, "F1tenth: An open-source evaluation environment for continuous control and reinforcement learning," in *Post Proceedings of the NeurIPS 2019 Demonstration and Competition Track*, ser. Proceedings of Machine Learning Research, H. J. Escalante and R. Hadsell, Eds. PMLR, 2020.

[5] G. Velasco-Hernandez, D. J. Yeong, J. Barry, and J. Walsh, "Autonomous driving architectures, perception and data fusion: A review," in *2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2020, pp. 315–321.

[6] M. Han, Y. Tian, L. Zhang, J. Wang, and W. Pan, "Reinforcement learning control of constrained dynamic systems with uniformly ultimate boundedness stability guarantee," *Automatica*, vol. 129, p. 109689, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109821002090

[7] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[8] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, *et al.*, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, pp. 604–609, 2020.

[9] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020, pp. 737–744.

[10] K. Jang, E. Vinitsky, B. Chalaki, B. Remer, L. Beaver, A. A. Malikopoulos, and A. M. Bayen, "Simulation to scaled city: zero-shot policy transfer for traffic control via autonomous vehicles," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems, ICCPS 2019, Montreal, QC, Canada, April 16-18, 2019*, 2019, pp. 291–300.

[11] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra, "Are we making real progress in simulated environments? measuring the sim2real gap in embodied visual navigation," *arXiv preprint arXiv:1912.06321*, 2019.

[12] T. P. Gros, D. Höller, J. Hoffmann, and V. Wolf, "Tracking the race between deep reinforcement learning and imitation learning - extended version," *CoRR*, vol. abs/2008.00766, 2020. [Online]. Available: https://arxiv.org/abs/2008.00766

[13] A. Hussein, M. Gaber, E. Elyan, and C. Jayne, "Imitation learning," *ACM Computing Surveys (CSUR)*, vol. 50, pp. 1 – 35, 2017.

[14] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. A. Theodorou, and B. Boots, "Imitation learning for agile autonomous driving," *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 286–302, 2020. [Online]. Available: https://doi.org/10.1177/0278364919880273

[15] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural computation*, vol. 3, no. 1, pp. 88–97, 1991.

[16] ——, "Alvinn: An autonomous land vehicle in a neural network," Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, Tech. Rep., 1989.

[17] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[18] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1861–1870.

[19] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[20] J. Kerr and K. Nickels, "Robot operating systems: Bridging the gap between human and robot," in *Proceedings of the 2012 44th Southeastern Symposium on System Theory (SSST)*, 2012, pp. 99–104.

[21] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[22] H. Mania, A. Guy, and B. Recht, "Simple random search provides a competitive approach to reinforcement learning," *arXiv preprint arXiv:1803.07055*, 2018.

[23] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, Tech. Rep., 1992.

[24] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[25] K. Judah, A. P. Fern, T. G. Dietterich, and P. Tadepalli, "Active imitation learning: Formal and practical reductions to i.i.d. learning," *Journal of Machine Learning Research*, vol. 15, no. 120, pp. 4105–4143, 2014. [Online]. Available: http://jmlr.org/papers/v15/judah14a.html

[26] R. Memmesheimer, I. Kramer, V. Seib, and D. Paulus, "Simitate: A hybrid imitation learning benchmark," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 5243–5249.

[27] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *arXiv preprint arXiv:1904.12901*, 2019.

[28] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[29] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," 2020.

[30] A. Y. Ng, S. J. Russell, *et al.*, "Algorithms for inverse reinforcement learning." in *Icml*, vol. 1, 2000, p. 2.

[31] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135.

[32] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl^2: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[33] A. Nichol, V. Pfau, C. Hesse, O. Klimov, and J. Schulman, "Gotta learn fast: A new benchmark for generalization in rl," 2018.

[34] R. Wang, J. Lehman, J. Clune, and K. O. Stanley, "Paired open-ended trailblazer (poet): Endlessly generating increasingly complex

and diverse learning environments and their solutions," *arXiv preprint arXiv:1901.01753*, 2019.

[35] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 1, pp. 1437–1480, 2015.

[36] K. Jothimurugan, R. Alur, and O. Bastani, "A composable specification language for reinforcement learning tasks," *arXiv preprint arXiv:2008.09293*, 2020.

[37] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. H. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Transactions on Automatic Control*, 2018.

[38] N. Fulton and A. Platzer, "Safe reinforcement learning via formal methods," in *AAAI Conference on Artificial Intelligence*, 2018.

[39] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, "Safe reinforcement learning via shielding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[40] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3387–3395.

[41] H. Zhao, X. Zeng, T. Chen, Z. Liu, and J. Woodcock, "Learning safe neural network controllers with barrier certificates," in *International Symposium on Dependable Software Engineering: Theories, Tools, and Applications*. Springer, 2020, pp. 177–185.

[42] "Variational autoencoder for end-to-end control of autonomous driving with novelty detection and training de-biasing," 2018.

[43] O. Pettersson, "Execution monitoring in robotics: A survey," *Robotics and Autonomous Systems*, vol. 53, no. 2, pp. 73 – 88, 2005.

[44] A. Desai, T. Dreossi, and S. A. Seshia, "Combining model checking and runtime verification for safe robotics," in *Runtime Verification*, S. Lahiri and G. Reger, Eds. Cham: Springer International Publishing, 2017, pp. 172–189.

[45] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, "Reachability-based safe learning with gaussian processes," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 1424–1431.

[46] S. Mitsch and A. Platzer, "Modelplex: verified runtime validation of verified cyber-physical system models," *Formal Methods in System Design*, vol. 49, no. 1, pp. 33–74, 2016.

[47] D. T. Phan, R. Grosu, N. Jansen, N. Paoletti, S. A. Smolka, and S. D. Stoller, "Neural simplex architecture," in *NASA Formal Methods*, R. Lee, S. Jha, and A. Mavridou, Eds. Cham: Springer International Publishing, 2020, pp. 97–114.

[48] S. L. Herbert, S. Bansal, S. Ghosh, and C. J. Tomlin, "Reachability-based safety guarantees using efficient initializations," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 4810–4816.

[49] A. Bajcsy, S. Bansal, E. Bronstein, V. Tolani, and C. J. Tomlin, "An efficient reachability-based framework for provably safe autonomous navigation in unknown environments," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 1758–1765.

[50] S. Bansal, A. Bajcsy, E. Ratner, A. D. Dragan, and C. J. Tomlin, "A hamilton-jacobi reachability-based framework for predicting and analyzing human motion for safe planning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7149–7155.

[51] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin, "Combining optimal control and learning for visual navigation in novel environments," in *Conference on Robot Learning*. PMLR, 2020, pp. 420–429.

[52] A. Gattami, A. Al Alam, K. H. Johansson, and C. J. Tomlin, "Establishing safety for heavy duty vehicle platooning: A game theoretical approach," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 3818 – 3823, 2011, 18th IFAC World Congress.

[53] M. Chen, Q. Hu, J. F. Fisac, K. Akametalu, C. Mackin, and C. J. Tomlin, "Guaranteeing safety and liveness of unmanned aerial vehicle platoons on air highways," *CoRR*, vol. abs/1602.08150, 2016.

[54] A. Dhinakaran, M. Chen, G. Chou, J. C. Shih, and C. J. Tomlin, "A hybrid framework for multi-vehicle collision avoidance," 2017.

[55] A. Valmari, *The state explosion problem*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 429–528.