# Poster: Unraveling Reward Functions for Head-to-Head Autonomous Racing in AWS DeepRacer

Allen Tian
atian2005@uchicago.edu
University of Chicago
Chicago, Illinois, USA

Eddy Guerra John
guerrajohne1@gator.uhd.edu
University of Houston Downtown
Houston, Texas, USA

Kecheng Yang
yangk@txstate.edu
Texas State University
San Marcos, Texas, USA

## ABSTRACT

AWS DeepRacer is a fully autonomous 1/18th scale race car designed to help developers learn and practice reinforcement learning through cloud-based simulations and real-world racing. What drives the reinforcement learning model is the reward function, a way to provide positive or negative feedback to an agent, guiding its learning process in reinforcement learning by assigning numerical values. In the AWS training environment, there are multiple modes in which you can run training simulations and evaluations in. These modes include Time Trial, Object Avoidance, and a relatively new mode: Head to Bot. The research done in this project was primarily focused on testing reward functions in the Head to Bot mode as well as developing a research function that would be suited for training in this new Head to Bot mode. The developed algorithm outperformed the default Centerline reward function, as well as the Object Avoidance reward function.

## CCS CONCEPTS

• **Computing methodologies → Machine learning**; *Artificial intelligence.*

## KEYWORDS

machine learning, artificial intelligence, reinforcement learning, reward function, autonomous racing

**ACM Reference Format:**
Allen Tian, Eddy Guerra John, and Kecheng Yang. 2023. Poster: Unraveling Reward Functions for Head-to-Head Autonomous Racing in AWS DeepRacer. In *The Twenty-fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc '23), October 23–26, 2023, Washington, DC, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3565287.3617987

## 1 INTRODUCTION

The AWS DeepRacer is an autonomous 1/18th scale race car that has the ability to drive on a track or race other vehicles on a track. The DeepRacer is often used for developers to become familiar with the machine learning technique known as reinforcement learning. Reinforcement learning is a method of machine learning training that is based on rewarding positive behaviors and punishing negative behaviors through the reward function. As of the writing of this paper, there are three modes in the training environment; Time Trial, Object Avoidance, and a newly created Head to Bot. The Time Trial and Object Avoidance modes each have an example reward function tailored to the mode on the AWS website [1], but for the Head to Bot mode, there is no such thing.

The goal with this project was to test how the example reward functions would fare in the Head to Bot mode, as well as attempt to develop a reward function that outperforms the sample reward functions and more efficiently navigates the Head to Bot mode.

### 1.1 Reinforcement learning

The machine learning method used in the AWS DeepRacer is reinforcement learning. The reinforcement learning method trains the model through a trial and error process, rewarding desired behaviors and punishing undesired behaviors. The model begins randomized, but as more training iterations occur, its movements become more refined and begin to achieve optimization. The driving force in this method is the reward function, which assigns the reward that the agent receives from its actions. By factoring in environmental variables such as distance to the next object and position on the track into the way we calculate our reward, we can influence the way the model trains, allowing us to tailor our reward function to an environment to achieve better training results.

### 1.2 Different Modes and Reward Functions

In the DeepRacer training platform, there are 3 modes that you can train the vehicle in; Time Trial, Object Avoidance, and Head to Bot. Time Trial is the first and most basic mode, which involves an agent on an empty track. If the vehicle exits the track, it will have effectively crashed and be placed back on the track. The Centerline reward function, which is the reward function tailored to this mode, rewards the highest in the middle and calculates 3 markers that get increasingly further from the center line. As the agent hits each of these markers, the reward is decreased, with the lowest reward value assigned at the edge of the track.

The second mode in the training platform is Object Avoidance. Instead of just having an agent in an environment, there are now static objects placed on the track. The vehicle must now avoid these objects as well as stay between the two edges of the track. If the agent crashes or goes off track, it will be placed back on track. The Object Avoidance function takes these factors into account by creating two individual rewards, one for staying on the track, and the other for navigating around the objects. The reward function first assigns a reward if it is between the two edges of the track. Then the reward function will calculate whether the vehicle is on

*MobiHoc '23, October 23–26, 2023, Washington, DC, USA*
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9926-5/23/10.
https://doi.org/10.1145/3565287.3617987

track to collide with the next object. If so, then it will begin to reduce the reward as it crosses the three distance thresholds. If it is not on the same path, the highest reward will be assigned. In the end, both rewards will be weighted, added, and assigned.

The third and final mode that exists in the AWS DeepRacer Training platform is the Head to Bot racing mode. All other conditions hold true from the Object Avoidance mode, except now the static objects placed on the track are now dynamic. This is the mode where all the research in this project will be conducted. Unlike the previous two modes, this is a relatively new mode, with little research done on it. We aim to design a reward function that is suited for training in this mode.

### 1.3 Parameters of the Environment

There are also parameters of the environment that are modifiable by the user. The ones that were modified in this project are the number of boxes, the number of bot cars, as well as speed of bot cars. The track is also modifiable, but we used the default track in the platform for this project.

## 2 BACKGROUND

In the past, there have been documentations[2] of an algorithm for DeepRacer vehicle to allow multi-vehicle applications. However, this application was not created through reinforcement learning, and a specific algorithm was followed. Because of the novelty of dynamic object applications in AWS DeepRacer, as well as the lack of research on AWS DeepRacer as a whole, there are not many examples to compare results and methodologies with. However, among the existing examples, our methodology aligns similarly with a recent documentation[3] although altered to accommodate for the Head to Bot mode, but the results(Elapsed Time) are drastically different as our experiment changed many different environmental variables which would drastically alter the results, as well as was run in a completely separate mode with dynamic objects.

## 3 METHODOLOGY

Now that we understand how the model trains and the role of the reward function in the training of the DeepRacer model, we want to test the Centerline and Object Avoidance reward functions in the Head to Bot training mode. We want to see if we can gain any insight into what techniques are successful in this mode, as well as overall assess the performance of each function in this mode.

### 3.1 Testing of Centerline and Object Avoidance Reward Functions

Since we would like to assess the effectiveness of each reward function in this new mode, we must design an experiment that will allow us to compare the reward functions in a quantifiable manner. We will train each reward function(Centerline and Object Avoidance) for one hour in the Head to Bot racing mode. We chose one hour[3] because it gives the model enough time to begin maturing, but not enough for all the model's performances to begin converging.

### 3.2 Parameter Modifications

We set the number of bot cars to 3, as well as changed the speed of bot cars from 0.2 to 0.4. The reason we made these changes is that 3 bot cars allow the agent to have a good mix of navigation of dynamic objects as well as staying within the two borders. If the number of bot cars is too few, then the agent will have fewer chances to navigate dynamic objects, thus resulting in poorer navigation of the bot cars. If the number of bot cars is too many, then the agent will be constantly fixated on navigating the next bot car and the agent's performance in staying between the borders will suffer. With 3 bot cars equally spaced around the track, we have a good mix with stretches of no bot cars as well as sufficient opportunity for the agent to practice the navigation of a bot car. Additionally, we changed the speed of the bot cars from 0.2 to 0.4, as 0.2 was far too slow relative to the agent. With 0.4 speed, the agent has far more difficulty passing the bot car, but would still be able to maneuver around it.

### 3.3 Evaluation Mode and Metrics

Once each model was trained for one hour, we evaluated the model under evaluation mode. We set the evaluation to run for 10 laps, let the trained model undergo the evaluation, and store the evaluation metrics in a file. The metrics that we used for this experiment were crash count, off-track count, and elapsed time. We chose these metrics as they give a good indicator of overall performance, factoring in speed, staying between the edges, and navigating around the bot cars.

### 3.4 Prototype Reward Function

So after seeing the results of the Centerline and Object Avoidance reward functions, we realized that the object avoidance reward function was actually very good at avoiding dynamic objects as well as static objects. The only thing we wanted to change with it was how it assigns the maneuver reward. In the Object Avoidance reward function, it essentially rewards the agent for not going in the wrong direction. We wanted to add an additional reward for going in the right direction, to especially reinforce the correct path in passing the moving vehicle.

The original Object Avoidance function calculates whether the vehicle is on track for collision and then decreases the reward. We used that to our advantage and added an additional reward that first checks whether it is on track for collision. If not, then it checks if it's going to run off the track, if both of those are negative, then we assign the additional reward that gets higher as we get closer to the next object, as it would mean we are not going to collide with that object. Summed up, we are essentially rewarding good behavior as well as punishing bad behavior at the same time. The Prototype reward function will undergo the same experiment and the data will be compared alongside the other two reward functions.

## 4 RESULTS

In the first metric of average elapsed time per lap, we saw that the Prototype model was able to outperform the other two models by a significant margin, yielding average differences of 1586 and 7011 milliseconds with the Object Avoidance model and Centerline model respectively.

In the second metric of average crashes per lap, the Prototype model was again able to outperform the other two models, yielding average differences of 0.1 and 1.6 crashes per lap with the Object Avoidance model and Centerline model respectively

In the third metric of off-track count, the Centerline model was actually able to outperform the Prototype and Object avoidance model by a significant margin, averaging 0.5 off-track counts per lap in contrast with the other two models both averaging 0.8 off-track counts per lap.

Overall, the Prototype model yielded more favorable results in 2/3 tests(average elapsed time, average crashes per lap) that were performed and did not show any deterioration from the Object Avoidance function that it was based on in the average off-track count per lap.

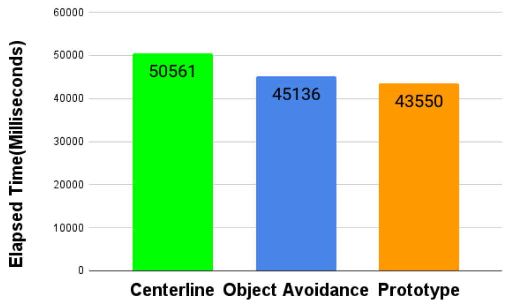

**Figure 1: Average Elapsed Time Per Lap Per Function**

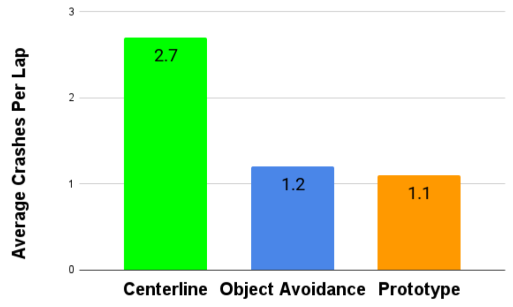

**Figure 2: Average Crashes Per Lap Per Function**

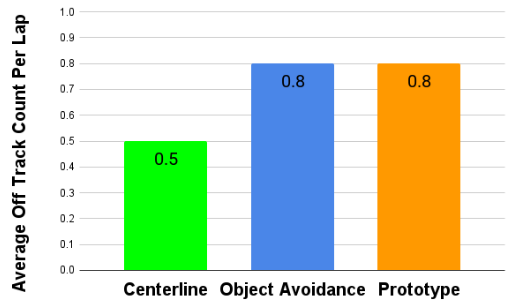

**Figure 3: Average Off-track Count Per Lap Per Function**

## 5 DISCUSSION

In the metric of average elapsed time per lap, the Centerline function performed poorly as expected as all it is programmed to do is to follow the Centerline. The Object Avoidance function fared well, but the Prototype function averaged 1586ms faster on average, showing that reinforcement of the optimal path is having some effect, decreasing the average lap time by a small margin.

In the metric of average crashes per lap, the Centerline function exhibited the most crashes. This makes sense as the Centerline function only takes the Centerline into account, disregarding objects. Again we see the Prototype function beating out the Object Avoidance function by a thin margin, averaging 0.1 fewer crashes per lap. While this margin isn't very substantial, it demonstrates that the idea of reinforcement of the best path has potential.

In the metric of average off-track per lap, we see a dramatic performance increase in the Centerline function, faring best in this discipline by a significant margin. This makes sense as the nature of the Centerline function is purely to follow the Centerline and not go near the edges of the track. We don't see any difference between the Object Avoidance and Prototype function, but they both still fared relatively well with less than one off-track per lap on average.

## 6 CONCLUSION

Overall, in this experiment, we were able to see how different reward functions affected the training outcome in the Head to Bot racing mode. The Centerline function suffered in performance for the majority of the metrics but performed well in the metric that aligns with its design. The Object Avoidance function performed well in the Head to Bot mode, adapting very well and producing good results, easily achieving higher results overall than the Centerline function. The Prototype Function performed better by a small margin in 2/3 of metrics and tied in 1/3 of metrics when compared to the Object Avoidance Function, showing that the reinforcement of the optimal path shows promise. The idea of rewarding good behavior while simultaneously punishing bad behavior, when applied to the Head to Bot racing mode, could be the key to creating a perfect balance of reward assignment in the complex dynamic environment of the mode. With tweaking of the distance values and reward values of the reinforcement of optimal path reward in the Prototype Function, the Prototype Function could exhibit even greater margins of performance.

## ACKNOWLEDGMENTS

This work is supported in part by NSF REU Site grant CNS-2149950.

## REFERENCES

[1] Amazon Web Services, Inc. 2023. AWS DeepRacer reward function examples. Online at https://docs.aws.amazon.com/deepracer/latest/developerguide/deepracer-reward-function-examples.html.

[2] Haikuo Du, Moyan Zhu, Wenjie Zhu, Yanbo Liu, Anbei Zhao, Wenchao Xu, Weiqi Sun, and Chunrun Du. 2022. A Dynamic Collaborative Planning Method for Multi-vehicles in the Autonomous Driving Platform of the DeepRacer. In *2022 41st Chinese Control Conference (CCC)*. 5524–5531. https://doi.org/10.23919/CCC55666.2022.9902120

[3] Jacob McCalip, Mandil Pradhan, and Kecheng Yang. 2023. Reinforcement Learning Approaches for Racing and Object Avoidance on AWS DeepRacer. In *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 958–961.