Local Feedback Control with Proximal Policy Optimization

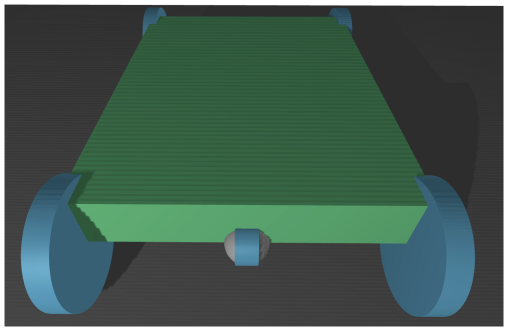
Tristan Shah

1 Introduction

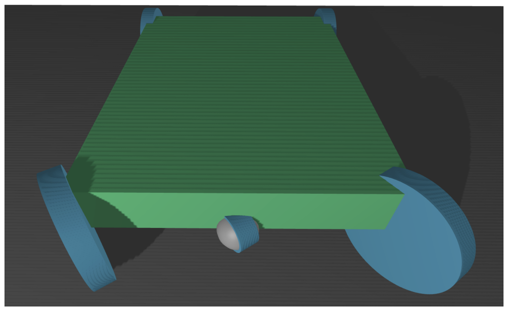
In this project a Local Feedback Controller (LFC) is implemented using a state of the art Reinforcement Learning (RL) algorithm. We test the algorithm's ability to control a robotic car in a physics simulator through actuation of back wheel torque and front wheel steering. Our scenario consists of a simple task of executing navigation between a randomized starting location and a fixed goal location to demonstrate the effectiveness of RL in this task. We show that after training the algorithm it is able to transport the car to the goal efficiently and stop.

1.1 Local Feedback Control

Local feedback control (LFC) is an important aspect in the pursuit of full self-driving technology. It is the most low level form of control in a hierarchy consisting of route planning, behavior planning, motion planning, and LFC. The LFC is the control algorithm responsible for executing a plan which is output by a higher level motion planner. The plan given to the LFC consists of a series of way-points which form a desirable path. Control outputs of the LFC will manipulate a self driving car such that the state of the car adheres to the motion plan while maintaining feasibility along with other constraints.



(a)



(b)

Figure 1: (a) Car with forward facing wheels. (b) Car with wheels turned left to the maximum allowable degree.

1.2 Proximal Policy Optimization

One of the state of the art policy gradient algorithms in reinforcement learning is Proximal Policy Optimization (PPO) [2]. Policy gradient algorithms in general work by finding the gradient of an objective function with respect to the parameters of a trainable policy network and using that gradient to update it. PPO is an enhancement on the standard policy gradient update to increase its stability and performance. Crucially, this algorithm adds importance sampling to reweight the policy gradient update due to changing action distributions. Another key feature of PPO is an entropy bonus which incentivizes the policy to be highly stochastic while still maximizing rewards.

1.3 Robotics Simulation

In order to train PPO as a LFC algorithm an accurate simulation of a robot must be used. One of the most popular robotics simulators used for training of reinforcement learning algorithms is MuJoCo [3]. However, MuJoCo and other simulators have often prioritized the appearance of physical realism rather than true physically realistic dynamics [1]. A new simulator known as Dojo [1] has been developed which is built from first principles as a "physics first" simulator. Dojo has three key features which distinguish it from others. 1.) Lower sample rate due to its variational integrator which allow for fast simulation. 2.) Accurate contact dynamics which prevent unrealistic artifacts such as interpenetration of objects through surfaces and velocity drift. 3.) Differentiability through discontinuities such as contact and friction events. Dojo represents a move towards realistic differentiable simulation which will translate to more robust machine learning algorithms used to control robots.

2 Current Work

In this section we describe the design and implementation of the simulated car. Additionally, how important quantities in RL such as states, actions, and rewards are defined which are crucial in training of the PPO algorithm.

2.1 Car Design

Within the Dojo simulator we construct a robotic car using primitive shapes such as cylinders, spheres, and a box. The design of the car consists of a body shown in green Fig 1a with four wheels attached at the corners. The body is unconstrained and has all six degrees of freedom. The back two wheels of the car have one rotational degree of freedom allowing them to spin and propel the car forward. To enable turning behavior the front two wheels are not only able to rotate but also turn on the z-axis $\pm 60°$. Turning behavior is demonstrated in Fig 1b where the wheels have been turned to their minimal position with respect to the body. The wheels are coordinated by a "steering wheel" shown as the knob in the front and center of the car. Both wheels have fixed orientation with respect to the steering wheel. A force applied to this component causes both wheels to turn to the same degree while also allowing circular rotation which moves the car forward.

2.2 States

Constructing the state vector for this problem is an important component of this project. We find a minimal amount of information needed by the agent to solve the given task. To navigate the car from a randomized starting position to the goal location the agent will need to be aware of the position, velocity, orientation, and angular velocity of the vehicle. Since the car in most normal situations will only move on a two dimensional plane we consider only the x and y components of the position $\mathbf{x}^{body} \in \mathbb{R}^3$ and velocity $\mathbf{v}^{body} \in \mathbb{R}^3$ vectors. Orientation and angular velocity of the car are denoted as quaternion $q^{body} \in \mathbb{H}$ and $\omega^{body} \in \mathbb{R}^3$ respectively. The front steering wheels of the car can turn on the z axis relative to the car body, therefore the orientation $q^{steer} \in \mathbb{H}$ and angular velocity $\omega^{steer} \in \mathbb{R}^3$ of the front wheels is also given. Lastly a scalar representing the percentage of time that has passed in the environment since the beginning of the episode is also given $t \in [0, 1]$. The full state can be represented by the concatenation of this information shown in Eq 1 where subscripts indicate vector indices.

$$\mathbf{s} = \begin{bmatrix} \mathbf{x}^{body}_{(1:2)} \\ \mathbf{v}^{body}_{(1:2)} \\ q^{body} \\ \omega^{body} \\ q^{steer} \\ \omega^{steer} \\ t \end{bmatrix} \in \mathbb{R}^{19} \tag{1}$$

2.3 Actions

Determining a physically realistic manner in which an agent can influence a robot is another step in the RL formulation process. Each component in the car can be actuated by a force in the Dojo simulator, however not all forces could be produced by a motor. In our work we consider that two control inputs are possible. Firstly, the agent can affect the torque on the back two wheels by a single scalar $\tau^{drive} \in [-0.2, 0.2]$. Secondly, the agent can apply torque to the steering wheel component $\tau^{steer} \in [-0.1, 0.1]$ which rotates both front wheels on their z axis. The control signal sent to the environment by the agent is a vector of both scalars shown in Eq 2.

$$\mathbf{u} = \begin{bmatrix} \tau^{drive} \\ \tau^{steer} \end{bmatrix} \in \mathbb{R}^2 \tag{2}$$

2.4 Rewards

The reward signal is arguably the most important component of RL. This signal is the supervision through which an agent learns to accomplish its task. In this work we design a reward which incentivises the agent to move the vehicle to the goal location while avoiding moving outside of the designated domain. The reward is given by the expression in Eq 3 where *body*$-$ and *body*$+$ represent the car before and after the action is applied to the environment. This reward is positive when the car moves towards the goal location and negative when it moves away. Function P is a penalty which is -1 if the agent moves the car outside of the domain and 0 otherwise. If the car moves out of the domain the episode terminates early.

$$r_t = \left\| \mathbf{g} - \mathbf{x}^{body-}_{(1:2)} \right\|_2 - \left\| \mathbf{g} - \mathbf{x}^{body+}_{(1:2)} \right\|_2 - P(\mathbf{x}^{body+}_{(1:2)}) \tag{3}$$

3 Experiments

The PPO algorithm is trained by collecting 20 episodes worth of data under the current policy then updating the actor and critic networks using this data. We used a batch size of 64 and 5 epochs per training cycle. This process repeats for 100 iterations before completion. The reward received per episode over the training period is shown in Fig 3. As the model trains the reward increases and eventually plateaus around 10.0. An interesting characteristic of this reward curve is the initial spike towards high rewards around episode 200 followed by a drop. The reason that this occurs is due to the agent learning to move the car forward which results in high reward. However, eventually this forward motion overshoots the goal location and results in the penalty for out being out of bounds. The agent has to learn a strategy to rapidly reduce it's velocity once it nears the goal location to prevent overshooting.

The trained policy is evaluated in the environment to visualize the trajectories it produces. A distribution of trajectories is shown in Fig 2 which shows the car starting at a random location on the x axis and moving towards the goal. All of the trajectories move close to the goal and stop before going out of bounds. Some of the trajectories exhibit the behavior of stopping and the reversing the car to obtain a final position located near the goal. Interestingly instead of only reversing the torque on the back wheels to slow the car down the agent instead develops a policy which makes a sharp turn

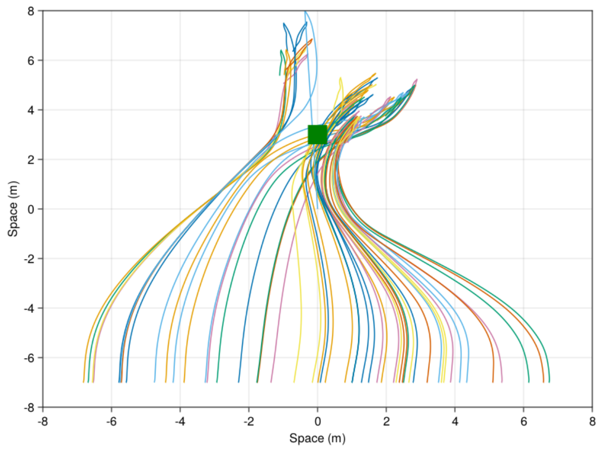


Figure 2: Runs of 50 trajectories in the simulator produced by the optimal trained PPO policy. Trajectories have randomized starting points on the x axis and navigate to the goal location (green square).

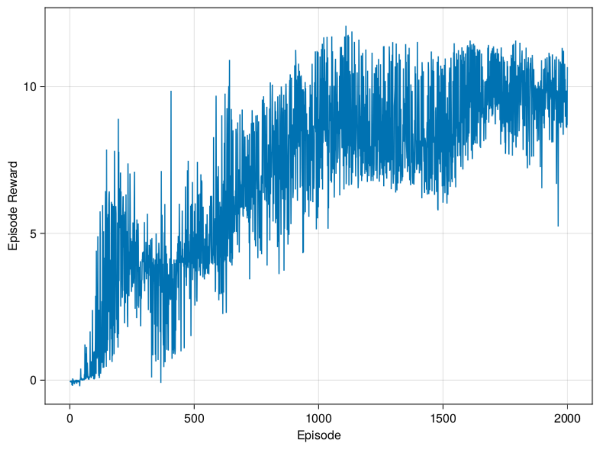


Figure 3: Reward received by the agent per episode.

4 Future Work

This work results in a RL agent which can successfully act as a local controller for a simple task of navigation between a randomized starting point and a fixed goal point. It serves as a proof of concept for control in a realistic robotics simulator with accurate friction dynamics. While this work is exciting it is also has limitations. For a more complete local planner the goal location must also be randomized. The agent should learn to navigate between arbitrary locations. Additionally, the current environment is quite simple, adding obstructions for the agent to overcome such as pedestrians or buildings would add a higher degree of complexity. Furthermore, combining this local control algorithm with a higher level planner such as a route planner could yield interesting results. Overall there are many exciting future directions to expand this project.

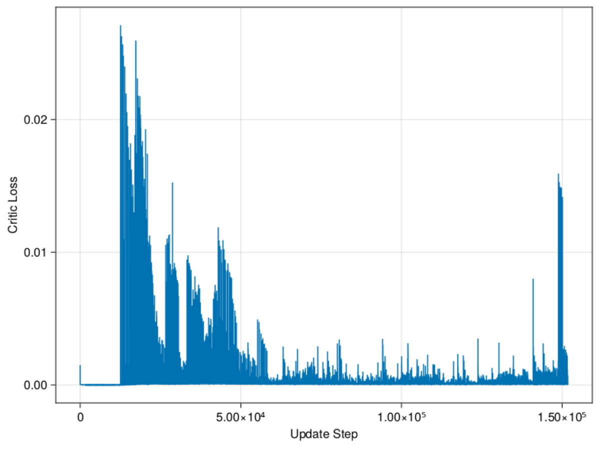


Figure 4: Loss of the value critic during training. This quantitiy represents the temporal difference between the predicted state value and the current reward plus the discounted next state value approximated by a target critic.

References

[1] Taylor A. Howell et al. *Dojo: A Differentiable Physics Engine for Robotics*. 2023. arXiv: `2203.00806 [cs.RO]`.

[2] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: `1707.06347 [cs.LG]`.

[3] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2012, pp. 5026–5033. DOI: `10.1109/IROS.2012.6386109`.

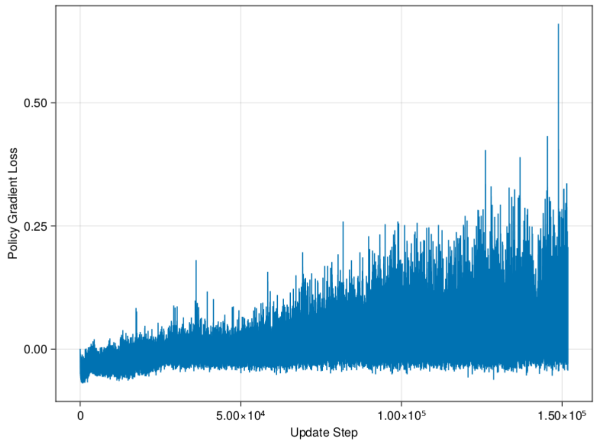


Figure 5: This loss is the clipped policy gradient loss as detailed in the original PPO paper [2].

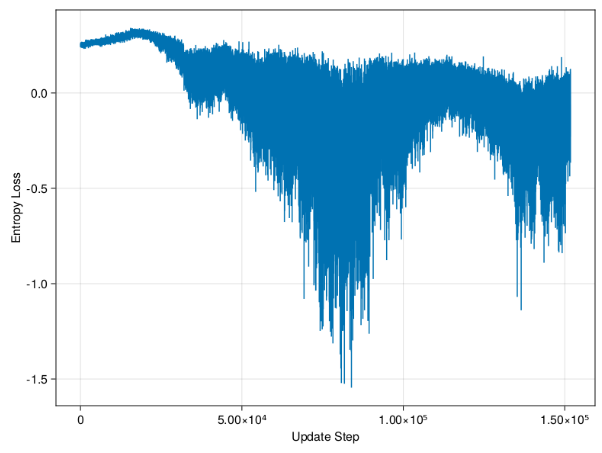


Figure 6: The entropy of the policy at each update.