Optimal Directional Drilling Control with Reinforcement Learning

Xiang Gu
Department of Computer Science
University of Texas at Austin
xiangu@cs.utexas.edu

Alexander Mathew Keller
Department of Mechanical Engineering
University of Texas at Austin
makeller1@utexas.edu

Abstract

We examined the possibility of tackling the directional drilling problem with a reinforcement learning approach. We formulated the problem as a Markov Decision Process (MDP) and applied reinforcement learning (RL) algorithms to it. We found Trust Region Policy Optimization (TRPO) algorithm gives the best performance over all RL-based algorithms that we tried. It closely matches the solution obtained by using model predictive control and yields a slightly better drilling trajectory. We further examined the robustness of our solution from two particular point of views – generalization to similar but different environments and generalization to different goal positions with different state representations. We showed that by training on a noise-corrupted environment it is helpful to aim generalization to new, unseen environments and we did not see expected improvements in goal-varying environments with several state representations we came up with. Further investigation is needed to thoroughly explain the results.

Keywords: Directional Drilling; Reinforcement Learning

Github URL: https://github.com/Xiang-Gu/directional_drilling

Youtube Video URL: https://youtu.be/ZG2LIbthpmE

1 Introduction

The drilling process is inherently a sequential problem where the process evolves over time according to actions taken throughout the process. The strategy will depend upon the current state and the action it takes will affect subsequent states. This gives rise to a reinforcement learning approach where sequential interaction is made between the learning agent and the environment, and those interactions are used to learn in a trial-and-error manner the best set of rules to follow.

In this paper, we examine the possibility of using reinforcement learning techniques to approach the directional drilling problem. We first model the problem as an episodic Markov Decision Process where we formally define the state space, action space, reward signal, and transition dynamics. Many thoughts from both empirical experience and limitations of physical dynamic/devices in the drilling business are drawn to accomplish the design of those components. Also, a simplified simulator of the drilling dynamics is developed to aim in carrying out experiments.

We then performed three experiments to examine our solution for different purposes – the first experiment tried to find the best RL learning algorithm to solve this problem and we found trust region policy optimization (TRPO) is the best option over all choices we considered. The second experiment tried to improve the robustness of the learned policy by adding noise to the simulator because, in practice, it is usually not possible to know the coefficients of the environment that determines the environment dynamics. We examined and justified this simple approach of adding noise to the training environment can improve performance on similar, yet different, environments. We compared four ways of adding noise and determined adding noise to the next state whenever a step is taken is the most effective approach. Finally, we also performed experiments on testing how to train the agent such that the learned policy is insensitive to the goal position, which is desired in real directional drilling business or it enables real-time position changing drilling with further effort. For this experiment, we tested one particular way of increasing robustness – state representation. We picked three state representations and trained and tested on a goal-varying environment with those three state representations. The results are not as good as expected and further investigation is needed to explain the results.

We also point out several deficiencies and possible improvements for our work and conclude our paper by reviewing the contents and potential future work.

2 Background and related work

Consider the problem of directional drilling in the oil and gas industry – determining a series of steering actions in order to drill a well horizontally into a reservoir. The trajectory taken to the reservoir is critical to the success of hydrocarbon resource extraction. An optimal well should intersect the reservoir in a purely horizontal direction and be as smooth as possible (require minimal curvature). In practice, there is a third requirement dictating the drilled path stays close to a predetermined path called a well plan. This problem, called trajectory tracking, has been approached with a variety of methods.

2.1 Reinforcement Learning

In [Zhang et al., 2018] a policy is developed with the Actor-Critic algorithm, a reinforcement learning method, to track a well plan with minimum curvature. Specifically, the policy (actor) and value function (critic) are represented with neural networks with radial basis activation functions. In [Martinsen and Lekkas, 2018] the authors learn a policy for curved trajectory tracking of ocean vessels experiencing disturbances from unknown currents. Their method is policy-based and represented by a neural network. Similar to this work, the method is model free. A form of transfer learning is utilized to speed up learning in which a simpler straight-line tracking policy is learned and then used to initialize a policy learning the curved trajectory tracking problem. Their reward function is relatively simple as the sum of two terms: one penalizing fast control changes and the other a Gaussian function that gives rewards dependent on the agent's distance from the path.

2.2 Control Theory

Controllers designed for steering a drill string have been detailed in literature since at least the late 1990s, however, they all depend on a well plan to track. Several trajectory tracking controllers are compared in [Liu et al., 2016] including classical PID, fuzzy logic, and a new trajectory control method called minimum-energy. However, the controller is designed for drilling tools with only first-order dynamics, whereas contemporary tools have at least second-order dynamics. [Demirer et al., 2019] formulate a model predictive controller that minimizes inclination and azimuth errors from a predetermined well plan every 30 feet. They model the drilling dynamics as a second-order differential equation in space. Though a path with minimal curvature is sought, the curvature is only implicitly minimized by constraining the

change in control input between control steps. The authors of [Kremers et al., 2015] model the drilling dynamics with a system of delay differential equations and develop a dynamic state-feedback controller to track a well plan.

Due to the uncertain nature of drilling deep underground and the financial risks associated with poor control, robustness is a critical feature of steering controllers. Sun et al. [2012] addresses the robustness problem with an $\mathcal{L}1$ adaptive controller which estimates model parameters and disturbances and ensures closed-loop stability. Similarly, [Inyang and Whidborne, 2019] guarantee stability with their bilinear proportional plus integral based controller.

This paper takes a new approach to the directional drilling steering problem by attempting to control the tool along an optimal path without a predetermined well plan. Following a well plan is not always the best thing to do; in practice, well plans are designed with first-order attitude dynamics, whereas modern drilling tools have more restricted second-order dynamics. This mismatch results in sub-optimal trajectories. Furthermore, every drilling tool has a limit of curvature generation capability. This property of the tool determines which points underground can be feasibly drilled to. If the dynamics of the tool change, such as when entering a new formation, the curvature generation capabilities may drop and render the reservoir entrance unreachable. By tracking a static well plan, these situations cannot be rectified.

This paper uses reinforcement learning to tackle this problem. Reinforcement learning provides a strong alternative to control theory formulations since this optimal control problem is non-convex.

3 Problem Formulation

3.1 MDP Structure

We model the problem as a Markov Decision Process (MDP) and, considering the audience of this paper, choose to omit the formal definition of MDP for brevity.

3.1.1 Transition Dynamics

The environment evolves according to a second-order differential equation similar to [Demirer et al., 2019],[Inyang and Whidborne, 2019], and [Martinsen and Lekkas, 2018], which has been field tested and shown to match well with the true drilling dynamics shown in [Cockburn et al., 2011]. The phenomenological model describing the 2 dimensional steering dynamics is described below:

$$\tau \frac{d^2\theta}{d\epsilon^2} = -\frac{d\theta}{d\epsilon} + K_u u + K_b$$
$$\frac{dx}{d\epsilon} = \cos\theta$$
$$\frac{dy}{d\epsilon} = \sin\theta$$

Where ϵ is the drilled depth, also called measured depth, θ is the inclination of drilling direction, τ is a depth constant, u is the control input $\in [-1, 1]$ K_u is the curvature generation capability of the tool (maximum turn rate), K_b is an uncontrolled disturbance coming from the environment, x is the vertical depth of the drill bit, and y is the lateral position of the drill bit. For clarity, we assume $K_b = 0$ for the experiments in the study, but it is trivial to include it in the dynamics.

3.1.2 State Space

The state space is designed to be a four-element tuple $(x, y, \theta, \dot{\theta})$, where $x \in$ [0, 1000] (unit: ft) represents the x coordinate of the drill bit along the vertical direction, $y \in$ [0, 500] (unit: ft) represents the y coordinate of the drill bit along the horizontal direction, $\theta \in [-10, 100]$ (unit: degree). represents the angle between the drilling direction and the vertical line, $\dot{\theta} \in [-25, 25]$ (unit: degree/ft) represents the angular velocity of the drill bit.

The x and y dimension can be arbitrarily large according to the real position of the oil reservoir but here we chose the goal position to be at $(500, 500)$ and hence decided the bounding box of the grid. You may notice that we allow double distance in the x-direction because we consider it a very common practice for the drilling bit to go down beyond the goal and then later need to drive up. Thus, we allow some room in the grid along the x-axis for the agent to learn that in case a direction reach is not possible. The θ and $\dot{\theta}$ ranges are determined by physical and practical drilling constraints.

3.1.3 Action Space

Though the actual action space is continuous between -1 and 1, directional drillers commonly set the steering force with a coarse resolution of about .1 in practice, so we have discretized the action space in this study. The action space is a discrete set of 21 actions that takes the value from -1 to 1 with 0.1 increments, i.e., A = {-1, -0.9, . . . , 0., . . . , 0.9, 1.0}. The

action represents the proportion of steering force applied to deviate the drill bit and subsequently the drilling direction. For example, an action of 1.0 is equivalent to applying 100% of the steering force to increase theta at the fastest rate.

3.1.4 Reward Design

A non-zero reward will be received at the end of each episode, signaling by leaving the bounds on x and y. The reward function is a combination of error penalty in landing position, final inclination, and maximum curvature. Formally,

$$R = \begin{cases} -\sqrt{(x_t - G_x)^2 + (y_t - G_y)^2} - 2|\theta_t - \theta_G| - |\max \theta'| & t = T \\ 0 & t < T \end{cases}$$

The $-\sqrt{(x_t - G_x)^2 + (y_t - G_y)^2}$ term penalizes the final location when we move out of bounds and we can minimize this penalty term by moving as close to the goal position as possible. The $-2|\theta_t - \theta_G|$ term penalize the final inclination when we move out of bounds and it can be minimized if we move out of bounds with the drilling bit angle being horizontal. Finally, the $-|\max \theta'|$ term penalizes the max curvature of the trajectory form in the current episode. $|\max \theta'|$ here maximizes all θ''s at each step in current episode. This term can also be minimized by having a smooth trajectory where the max curvature at all points of this trajectory is reasonably small. All three factors play a role in conjunctively determining the reward and have conflicting effects. For example, a straight line that connects starting position and goal position has zero curvature and zero landing error but it has a high inclination, and more importantly, due to limitation of the physical device, such a straight-line trajectory is impossible. We hope those three penalty terms "fight and balance" each other so that by maximizing the return we end up with a policy that can produce accurate, smooth trajectory with good entry inclination, which is most valued in the drilling industry.

3.1.5 Episodic vs. Continuing Task

The directional drilling problem is naturally described as an episodic task. A goal position is defined (reservoir location) and when it is reached the drilling task is completed.

4 Experiments

We designed and performed experiments to answer the following three questions:

1. Which algorithm has the "best" performance?
2. How good will our learned policy still be when the dynamic of the environment is slighted changed?
3. How do different state representations affect performance on environments with a changed goal position?

From a high level, the first question is asking which RL algorithm, either a value-based or policy-based one, is most effective in solving this problem when the performance is measured with a certain metric. The second and third question is asking how good our learned policy can generalize to new environments in the forms of new transition dynamics or new goal positions, so, we want to take a closer look at the robustness or generalization of our solution.

4.1 Experiments 1: what is the best algorithm to pick?

We chose the following algorithms to run on the environment:[1]

- Tabular n-step Sarsa
- n-step Sarsa with Function approximation
- One-step Actor-Critic
- REINFORCE with Baseline
- Trust Region Policy Optimization (TRPO)
- Proximal Policy Optimization (PPO)

For Tabular n-step Sarsa, we simply discretize our state-space with $(100, 100, 91, 51)$ bins in each dimension of the state space, $n = 20$, $\epsilon = 0.05$, learning rate $\alpha = 0.3$.
For n-step Sarsa with Function Approximation, we used linear function approximation with tile coding with 16 tiling and tile width $= (50, 5, 5, 10)$ in each dimension of the state space, $n = 20$, $\epsilon = 0.05$, learning rate $\alpha = 0.006$.

[1] We used OpenAI Spinning Up implementation of REINFORCE with Baseline, TRPO, and PPO. [Achiam, 2018]

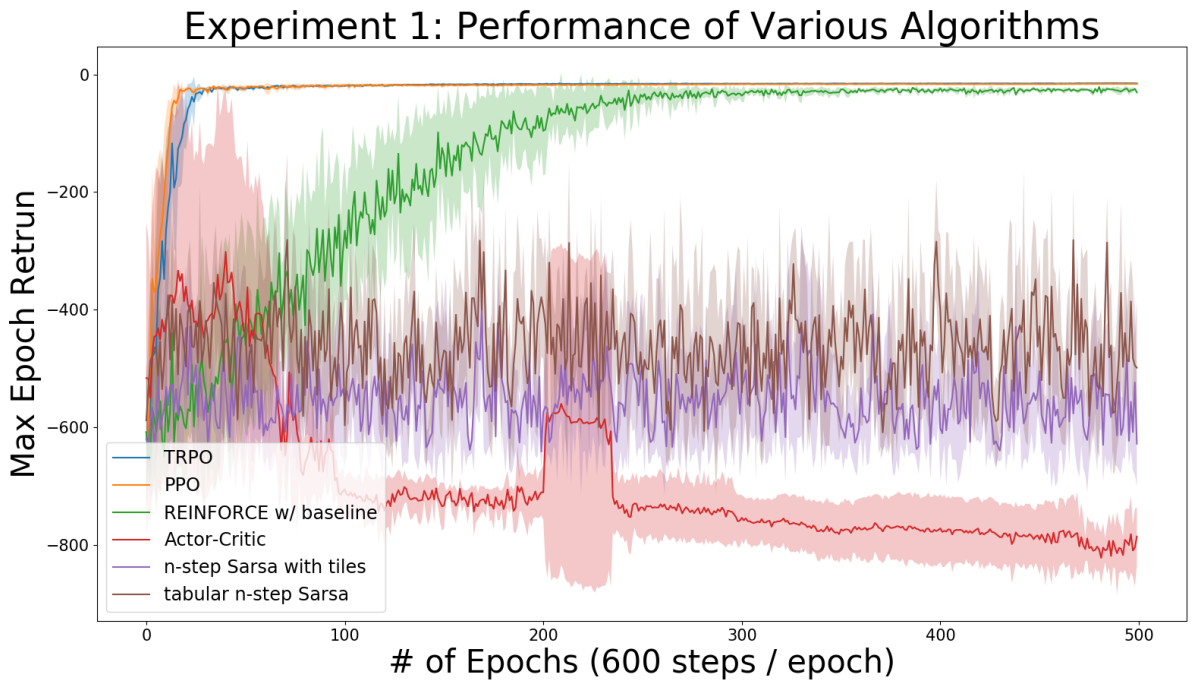

Figure 1: Performance comparison of various algorithms.

For One-step Actor-Critic, we used neural networks that have two layers with 32 units and the ReLU activation function and one fully connected layer without activation function. The learning rate for both the policy network (Actor) and the state-value approximation network (Critic) is set to 0.00001.
For REINFORCE with Baseline, TRPO, and PPO. We used neural networks that have one hidden layer with 32 units with a tanh activation function for both the policy network and the state-value approximation network. All the other parameters (e.g. policy network learning rate, state-value approximation network learning rate) are set to the default value.

For a fair comparison, we allow 500 epochs with 600 steps per epoch for each algorithm to interact with the simulator and make updates. We recorded the maximal episode return in each epoch during training. The same experiments are repeated 5 times for each algorithm and the result is shown in figure 1.

From the results, we can see that n-step Sarsa method does not have a good online performance because of the constant ϵ exploration and in this specific problem, one bad exploratory action could spoil the whole sequence and end up with a bad outcome (similar to the example of Acrobot where the author solved it with $Sarsa(\lambda)$ with $\epsilon = 0$. The exploration is solely driven by optimistic initialization [Sutton et al., 1998].) One-step Actor-Critic learned to approach the goal in the early stage but then suddenly the performance degenerate after about 50 epochs. We are still not completely sure why this happened but a similar behavior is observed from our coding assignment 4 where REINFORCE with Baseline has worse performance than REINFORCE. For the other three methods that we used from other people's implementation, they perform better and they converge to almost the same point. This is not surprising since those implementations utilize batch training, together with some tricks, to stablize learning. Among those three methods, TRPO and PPO have significantly better learning performance than REINFORCE with baseline both in terms of learning rate and final convergent performance. Finally, we can see that TRPO and PPO have very close learning performance. TRPO has a slightly better average learning rate (not statistically significant though) than PPO.

For experiments 2 and 3, we use solely TRPO as our learning algorithm because of its superior performance demonstrated in this section.

4.1.1 Comparison with Model Predictive Controller

To evaluate the performance of the policy, we compare the policy with a model predictive controller (MPC) from control theory. Model predictive controller determines actions by solving an optimization problem at every time step. Due to the difficulty of this optimization problem, a few approximations had to be made which reduced from the optimality of the MPC. The details of the formulation are included in the source code title 'mpc.py'. The resultant trajectories from

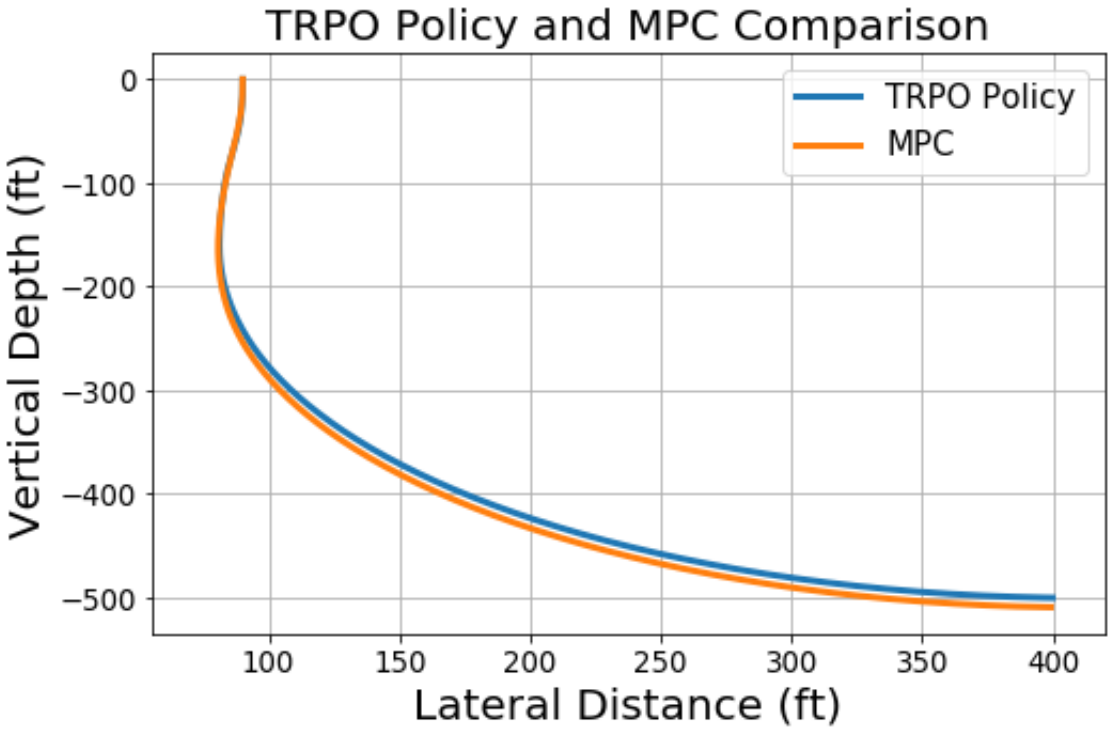

Figure 2: Comparison of drilled trajectory using TRPO policy and MPC from control theory.

the policy learned with TRPO and dictated by MPC are shown in 2. The solutions have nearly the same shape, however, the TRPO Policy performs better receiving a return of -22.4 while the MPC received a return of -31.36. We would expect the MPC to produce the globally optimal solution since it is model-based and matches the environment, however, the optimization problem could not be formulated with strict boundary conditions nor as a mixed-integer problem due to the limitations of the open source IPOPT solver, so the solution is suboptimal.

4.2 Experiment 2: how can we improve the robustness of learned policy?

In practice, the dynamics of the system can change while drilling due to multiple factors such as different rock strengths and fault lines. Since our simulator cannot perfectly match the dynamics of the real world we are again faced with the problem of how to ensure a policy that is learned from simulation can achieve good performance in actual drilling environments given the simulator is not perfect [2]. Many approaches have been proposed to address this problem including but not limited to [Jakobi et al., 1995], [Abbeel et al., 2006], [Hanna and Stone, 2017]. In our project, we use the approach of adding noise to our simulator, and we hope that the noise will expose more of the state space to the agent that will otherwise never be visited. The hope is the agent will know how to act in those "unusual" states, which the agent might actually run into in the testing phase with a different environment.

Namely, we set up a two-phrase experiment where in the first phase, we train our agent with TRPO on a bunch of candidate environments in the same way as we did in experiment 1. Those environments are different from each other and will be further explained later. Then, for each policy learned from phase 1, we deploy it on a particular environment (we called Stochastic Environment below) for 1000 episodes and record the episode returns. This two-phrase experiment is repeated 5 times for each candidate environment to average out stochasticity. The candidate environments we used in phrase include:

- **Average Parameter Environment**: the normal environment with constant environment dynamic coefficients. [3]
- **Noisy Next State Environment**: an environment where an Gaussian distributed noise with standard deviation of $\sigma_x = 1.5, \sigma_y = 1.5, \sigma_\theta = 0.5, \sigma_{\dot{\theta}} = 1$ is added to the next state whenever a step is taken.
- **Random Starting State Environment**: an environment where the starting state is $(x, y, 0, 0)$, where x is uniformly sampled from $[70, 110]$ and y is uniformly sampled from $[0, 20]$.

[2] Sometimes this problem is also known as sim2real.

[3] We used this one as our environment in experiment 1.

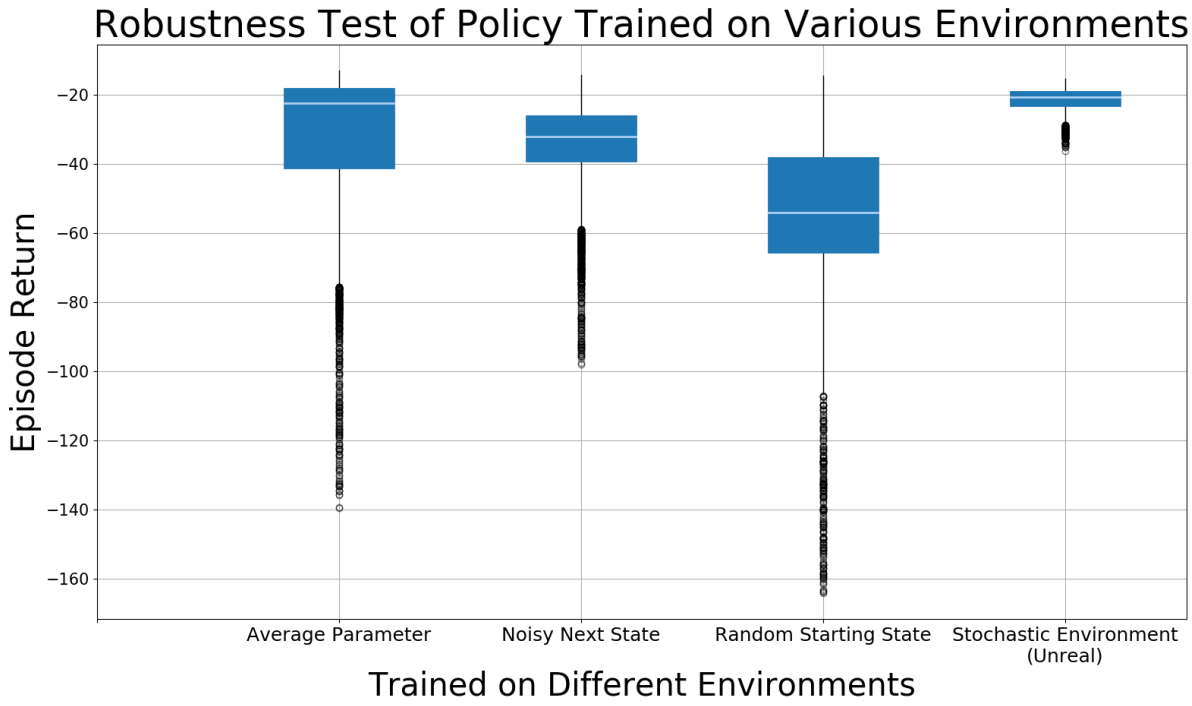

Figure 3: Robustness performance comparison of policy trained on various environments.

- **Stochastic Environment**: An environment that whenever reset() is evoked, two coefficients, τ and K that jointly determines the dynamics of the environment, are uniformly sampled from $[10, 20]$ and $[14, 22]$ respectively. This is also the policy that we use in phrase 2 to test the robustness of learned policies from phrase 1 [4].

The result is shown in figure 3. First of all, we are confirmed about our hypothesis that the unreal environment is able to give the best performance. For Random Starting State, it has the worst median performance as well as the worst-case performance. Although starting from different positions will be able to lead our agent (drilling bit) to a broader range of state space that will otherwise never be visited if it always starts from the origin, this is not directly related to our test rubric where the transition dynamic is changed. In other words, we are training the agent to do something that is not tested in the test phase, and hence a worse performance is expected. For Average Parameter Environment and Noisy Next State Environment, Average Parameter Environment has a higher median performance while Noisy Next State Environment has a better worse case performance which we believe is more important in real-world because it is really expensive to have real-world interactions so we are more conservative in the presence of the sim2real problem. Nevertheless, the Average Parameter Environment did a better job than we expected because results in this environment reflect what will happen when we put no effort to address the sim2real problem, and the result is not bad (or, for some people, even satisfactory).

4.3 Experiment 3: can we achieve goal position insensitive learning?

Finally, we also carried out an experiment that aims to test, again, robustness/generalization from the perspective of state representation. We want to see is there any performance difference when using different state representations (see below for details), that is, different state representations will enable the algorithm (TRPO) to learn different policies that generalize well to similar, yet different, environments.
The three state representations we are going to use are listed as follows:

- **state representation 0** $(x, y, \theta, \dot{\theta})$: the default state representation that we've been using so far.
- **state representation 1** $(\Delta x, \Delta y, \Delta\theta, \dot{\theta})$: the first three dimension in the state now becomes difference between current position and the goal – $\Delta x = G_x - x$, $\Delta y = G_y - y$, $\Delta\theta = \theta_{Goal} - \theta$, where G_x, G_y represents the x and

[4] Including this environment as the candidate for phrase 1 seems cheating and yes it is. It is better to think of the result trained in this environment as an (unreal) baseline for other candidate environments to compare against.

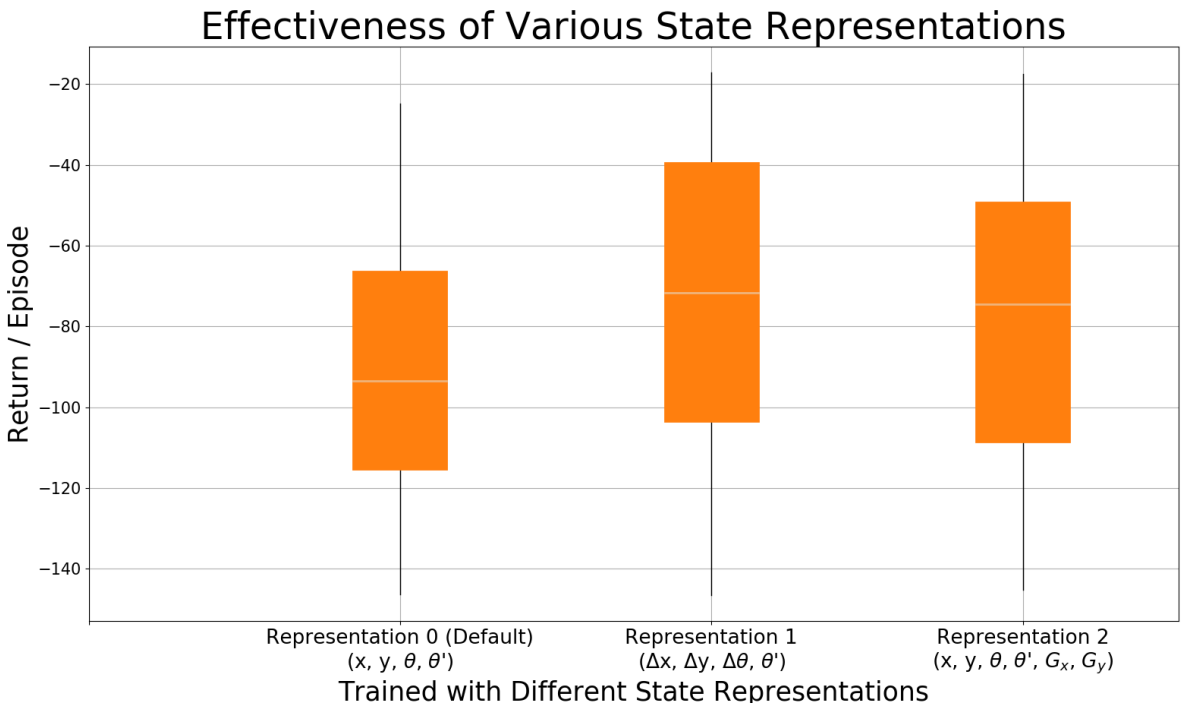

Figure 4: Generalization performance comparison of policy trained with various state representation.

y coordinate of the goal position, θ_{Goal} represents the angle between the line connecting the current drilling bit and the goal and the vertical line.

- **state representation 2** $(x, y, \theta, \dot{\theta}, G_x, G_y)$: we augment the state space now to include the current x and y coordinate of the goal position.

This time, however, we did not vary environment dynamics as the way to test robustness/generalization but instead, we used fixed environment dynamics (so Average Parameter Environment) with varying goal positions. Similarly to experiment 2, we set up a two-phase experiment – in phrase 1, we trained the agent with TRPO with those different state representations on an environment that has varying goal position – when the environment is reset, the goal position is determined by uniformly sampling a point from the square of $G_x \sim \text{UNIF}(475, 525)$ and $G_y \sim \text{UNIF}(425, 475)$. In phase 2, we deploy the learned policies from phrase 1 on a new environment with varying goals but the region where the goal is sampled from is different from that region used in phrase 1. The goal position is uniformly sampled from the region that is equal to the larger square $G_x \sim \text{UNIF}(450, 550)$ $G_y \sim \text{UNIF}(400, 500)$ exclude the small square $G_x \sim \text{UNIF}(475, 525)$ $G_y \sim \text{UNIF}(425, 475)$.

The result is shown in figure 4. Unsurprisingly, the default representation did not do very well since the state representation uses the "absolute" value of quantities and it will take the same action as long as it's in the same "absolute" state even though the goal position might be different. In contrast, representation 2 has a better performance since now it takes actions based on the difference between the current drilling bit and the goal position. In other words, the difference presented in the state representation stores the goal information implicitly. Then during test, the agent will take the same action as long as it is in a state that has the same position difference $(\Delta x, \Delta y)$, angular difference $(\Delta\theta)$, and angular velocity $(\dot{\theta})$, no matter where the true goal position is. As for representation 2, we expect it to be the best state representation because, after all, the goal position is explicitly coded in the representation. But the result does not support that. We do not know at this moment why – it might be that there are bugs in our code, or the choice for the varying goal region does not make sense. More thinking and experiments are needed to explain or verify this result.

5 Deficiencies and Possible Improvements

5.1 From Reality to Simulator

The full directional drilling steering problem is a three-dimensional problem with many uncertainties and possible disturbances. With a 3D environment, the state space and action space are squared in size. To run these experiments on

consumer-grade computers, we simplified the problem to two dimensions. A practical policy to be used in the field would include the third dimension.

We modeled the problem as an MDP, but the real system is non-markovian because of spatial delays. The sections of pipe connected from the surface to the drill-bit exert a reaction bending force which influences the drilling direction. Since the bending force is dependent upon the shape of the drilled hole, the transition dynamics depend on previous states. Other works have modeled this phenomenon with the use of delay differential equations, which could be implemented in the simulator to improve the policy for use in real-world directional drilling.

There is also a time delay that has not been taken into account in our formulation. Due to the communication methods between the drill-bit and the surface, measurements can be delayed by up to 5 minutes. This effect is not considered in our study- which is partially justified in cases where drilling is slow enough.

6 Conclusion and Future Work

In the paper, we examined the possibility of tackling the directional drilling problem with a reinforcement learning approach. We formulated the problem as an MDP and applied reinforcement learning algorithms to it. TRPO gives the best performance over RL-based candidate algorithms that we tried. It closely matches the solution obtained by using model predictive control and yields a slightly better drilling trajectory. We further examined the robustness of our solution from two particular points of view – generalize to similar but different environments and generalize to different goal positions with different state representations. We showed that training in a noise-corrupted environment is helpful to aim generalization to a new environment and we did not see expected improvements in goal-varying environments with several state representations we came up with. Further investigation is needed to thoroughly explain the results.

There are a few things that we can improve on our current work:

- Perform a grid search on the hyper-parameters for all three experiments, including the step size, n in the n-step method, and algorithm-specific parameters.
- Search for the best neural network architecture in various algorithms that we used in experiment 1, including the number of layers, the number of hidden units, and the activation function.
- Move toward a 3D solution by first developing a simulator in the 3D space and then modify our problem formulation and algorithm implementations accordingly to enable learning in the 3D space.

References

Chi Zhang, Wei Zou, Ningbo Cheng, and Junshan Gao. Trajectory tracking control for rotary steerable systems using interval type-2 fuzzy logic and reinforcement learning. *Journal of the Franklin Institute*, 355(2):803–826, 2018.

Andreas B Martinsen and Anastasios M Lekkas. Curved path following with deep reinforcement learning: Results from three vessel models. In *OCEANS 2018 MTS/IEEE Charleston*, pages 1–8. IEEE, 2018.

Zhengchun Liu, Robello Samuel, et al. Wellbore-trajectory control by use of minimum well-profile-energy criterion for drilling automation. *SPE Journal*, 21(02):449–458, 2016.

Nazli Demirer, Umut Zalluhoglu, Julien Marck, Hossam Gharib, Robert Darbe, et al. A model predictive control method for autonomous directional drilling. In *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2019.

Niek Antonius Henricus Kremers, Emmanuel Detournay, and Nathan Van De Wouw. Model-based robust control of directional drilling systems. *IEEE Transactions on Control Systems Technology*, 24(1):226–239, 2015.

Hui Sun, Zhiyuan Li, Naira Hovakimyan, Tamer Başsar, and Geoff Downton. L1 adaptive control for directional drilling systems. *IFAC Proceedings Volumes*, 45(8):72–77, 2012.

Isonguyo J Inyang and James F Whidborne. Bilinear modelling, control and stability of directional drilling. *Control Engineering Practice*, 82:161–172, 2019.

Colin Bruce Cockburn, Justo Matheus, Khoa Le Pham Dang, et al. Automatic trajectory control in extended reach wells. In *SPE Middle East Oil and Gas Show and Conference*. Society of Petroleum Engineers, 2011.

Joshua Achiam. Openai spinning up. *GitHub, GitHub repository*, 2018.

Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 2. MIT press Cambridge, 1998.

Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *European Conference on Artificial Life*, pages 704–720. Springer, 1995.

Pieter Abbeel, Morgan Quigley, and Andrew Y Ng. Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 1–8. ACM, 2006.

Josiah P Hanna and Peter Stone. Grounded action transformation for robot learning in simulation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.