Autonomous Offroading : Terrain Aware Motion Planning for Ground Vehicles

Kshitij Kumbar*, Pranav Keni † and Varun Nayak ‡

Current efforts in autonomous vehicle planning and controls generally assume ideal road conditions. With the motivation of making controllers more adaptive to general terrain profiles, we propose a velocity trajectory planner for a bumpy road to provide a smooth ride for the passenger, using different reinforcement learning techniques. We pose the problem as an MDP with a finite look-ahead terrain profile and unknown vehicle dynamics.

I. Introduction

Passenger safety and comfort has long been an interesting problem to study due to the wide spread approaches spanning different areas of research. This includes approaches ranging from active suspension design using optimal control approaches such as LQR/LQG [1] to cost based planning algorithms such as A* and D* [2]. However, the active suspension approaches do not consider any lookahead at all, only helping in increasing comfort at each step. The latter algorithms help in planning paths around difficult terrains and incorporating the additional distance to travel over slopes but do not plan velocity profiles that maximize comfort. These algorithms also suffer from computational issues especially when the map is unknown.

In this paper, a velocity trajectory planner is proposed for autonomous vehicles based on the knowledge of the terrain upto a certain lookahead distance with some uncertainty. The problem is modelled as a discretized Markov Decision Process, where at each state the action taken is the desired velocity commanded at the next state such that it maximizes comfort while minimizing time taken to reach the goal. The initial attempts use model-free reinforcement learning algorithms (SARSA and Q-Learning) to build up the Q matrix using random paths and velocity profiles. An alternate approach to learn the model from this randomly generated data is also discussed.

II. Baseline Approach

To create a proof of concept for the proposed approach, a model-free approach involving a simplified state space was implemented and tested. First, we defined a path along which the vehicle would travel. This information encoded in this path was the change of the gradient of the path (as our reward model would depend on how fast the car would go over this change of gradient). This was implemented as a randomly generated array of positive values within $[0, 10]$:

$$P_{1:n} = [\nabla^2 h_1 \; \nabla^2 h_2,, ... \nabla^2 h_k, ... \nabla^2 h_n]$$

A fixed set of policies which involved using fixed velocity profiles for the path, were used. These velocities profiles were generated from all possible velocities that the vehicle could achieve. An implementation of the SARSA[3] learning algorithm was carried out to build our state-action value function $Q(s, a)$ and selected the optimal policy by extracting $\max_{a_k} Q(s_k, a_k)$.

A. Problem Formulation

The problem formulation involved defining states of the system as:

$$S = \begin{bmatrix} \nabla^2 h \\ d_{goal} \end{bmatrix} \qquad |S| = 10 \times 100$$

where $\nabla^2 h$ is the change of gradient and d_{goal} is the distance to the goal.

*Department of Mechanical Engineering, Stanford University, *kshitijk@stanford.edu*
†Department of Mechanical Engineering, Stanford University, *pranavk8@stanford.edu*
‡Department of Mechanical Engineering, Stanford University, *vunayak@stanford.edu*

Table 1 Baseline Approach: Results

Algorithm	Training Time	Advantage over Random Policy
SARSA	2-3 minutes (100 iterations)	22.7%

The action space is the possible velocities that the vehicle should be at for the next state i.e. the vehicle would achieve the commanded velocity at the next step. Note that instead of time steps, we index our states using the path index k.

$$A = a_k = v_{k+1} \qquad |A| = 10$$

A deterministic equation was used for the reward model to populate the rewards for each sample generated during the learning process. The objective is to penalize high velocities in the presence of large change of gradients of the road while rewarding high velocities when there is a large distance to goal. This represents greater comfort for the passenger in the presence of bumps while ensuring the time taken to reach the goal is minimized. The reward model, addressing the objective of reaching the goal as fast as possible (first term) while making sure to go slow over large bumps (second term) was formulated as follows:

$$R(s_k, a_k) = a_1 d_{goal_k} |v_k| + a_2 \frac{\nabla^2 h_k}{|v_k|} \tag{1}$$

Here, a_1 and a_2 are judiciously chosen coefficients. Thus, our problem was defined as an MDP.

B. Training and Results

State, action and reward samples for the learning algorithm are generated from exhaustive simulations using the reward model developed below. Based on these simulations, we populated the state action value function $Q(s, a)$ using SARSA and extracted the optimal policy from it during testing. The optimal policy learned using SARSA was tested against a random policy for a test path. The optimal policy consistently performed 20% better than the random policy on the test path. The advantage over the random policy was computed as follows:

$$adv = \frac{\sum r_{\pi*} - \sum r_{\pi^{rand}}}{\sum r_{\pi^{rand}}} \times 100\%$$

III. Finite Horizon Approach

In the initial approach, a proof of concept was developed which performed noticably better than the random policy for a given path. However, the finite horizon in terms of value of change of gradients was not considered. In this section a finite horizon is considered for policy generation with different model based and model free learning methods and a few methods are used to improve upon the observations from the baseline model.

A. Augmented Problem formulation

The previously formulated state vector is now augmented with current velocity and observed next change of gradient to get the following augmented state formulation:

$$S = \begin{bmatrix} V_k \\ \nabla^2 h_k \\ \nabla^2 h_{k+1} \\ d_{goal} \end{bmatrix} \qquad |S| = 10 \times 10 \times 10 \times 100$$

where $\nabla^2 h$ is the change of gradient and d_{goal} is the distance to the goal. It is noted that the state space has significantly expanded following this addition. Measures taken to account for this increase in state space are described in one of the later sections.

B. Reward Function

With the addition of current velocity values and observed change of gradients, the reward function is now augmented to account for these state values and generate the rewards accordingly

$$R(s_k, a_k) = a_1 d_{goal_k} |v_k| + a_2 \frac{\nabla^2 h_k}{|v_k|} + a_3 \frac{\nabla^2 h_{k+1}}{|a_k|} \quad (2)$$

C. State Space Discretization

Initial observations from the baseline formulation indicated that the fine discretization of d_{goal} had a negligible effect on the reward of the system. To counter this, a change in weights was proposed which did not lead to an improvement in scores. Thus, d_{goal} values were binned by a factor of 10 to observe the effect it had on the reward output. It was observed that this binning of the d_{goal} resulted in negligible difference in the optimal policy score. Thus implementing binning reduces the state space dimensionality by a factor of 10.

D. Transition Model

In the baseline approach, we assumed that the vehicle could achieve any desired velocity. However, from physics, we know that this is not true. Therefore, we introduced richer dynamics (with noise) that computed the next velocity as a function of the current velocity and the desired velocity.

$$v_{k+1} = v_k + \text{clamp}(v_{desired} - v_k, +2, -2) + v_{noise}$$

Where v_{noise} is a number sampled uniformly from the set $\{-1, 0, 1\}$, representing stochasticity in the vehicle dynamics. The clamp() function represents the acceleration limits of the vehicle.

IV. Training Methods

Three reinforcement learning techniques were implemented: SARSA, Q-Learning and Maximum Likelihood Model-Based Reinforcement Learning with Randomized Updates (Dyna), as described in [3].

Algorithm 1 SARSA

```
Require:
  Sates $\mathcal{S} = \{1, \ldots, n_x\}$
  Actions $\mathcal{A} = \{1, \ldots, n_a\}$,    $A : \mathcal{X} \Rightarrow \mathcal{A}$
  Reward function $R : \mathcal{X} \times \mathcal{A} \longrightarrow \mathbb{R}$
  Learning rate $\alpha \in [0, 1]$
  Discounting factor $\gamma \in [0, 1]$
  procedure SARSA($\mathcal{X}$, $A$, $R$, $\alpha$, $\gamma$, $\lambda$)
    $s_0 \longleftarrow$ initial state
    Initialize $Q$ arbitrarily
    while $k$ in $n$ do
      Select $a_k$ based on $Q$ and some exploration strategy
      while $s$ is not terminal do
        $r_k \longleftarrow R_{obs}(s_k, a_k)$                                  ▹ Observe the reward
        $s_{k+1} \longleftarrow T_{obs}(s_k, a_k)$                              ▹ Observe the new state
        Calculate $\pi$ based on $Q$ and some exploration strategy
        $a_{k+1} \longleftarrow \pi(s_{k+1})$
        $Q(s_k, a_k) \longleftarrow Q(s_k, a_k) + \alpha \cdot (r_k + \gamma Q(s_{k+1}, a_{k+1}) - Q(s_k, a_k))$
    return $Q$
```

Algorithm 2 Q-Learning

```
Require:
  Sates $\mathcal{S} = \{1, \ldots, n_x\}$
  Actions $\mathcal{A} = \{1, \ldots, n_a\}$,        $A : \mathcal{X} \Rightarrow \mathcal{A}$
  Reward function $R : \mathcal{X} \times \mathcal{A} \longrightarrow \mathbb{R}$
  Learning rate $\alpha \in [0, 1]$
  Discounting factor $\gamma \in [0, 1]$
  procedure Q-LEARNING($\mathcal{X}$, $A$, $R$, $\alpha$, $\gamma$, $\lambda$)
      $s_0 \longleftarrow$ initial state
      Initialize $Q$ arbitrarily
      while $k$ in $n$ do
          Select $a_k$ based on $Q$ and some exploration strategy
          while $s$ is not terminal do
              $r_k \longleftarrow R_{obs}(s_k, a_k)$                                  ▷ Observe the reward
              $s_{k+1} \longleftarrow T_{obs}(s_k, a_k)$                              ▷ Observe the new state
              $Q(s_k, a_k) \longleftarrow Q(s_k, a_k) + \alpha \cdot (r_k + \gamma \max_{a'} Q(s_{k+1}, a') - Q(s_k, a_k))$
      return $Q$
```

Algorithm 3 Max Likelihood Model-Based RL using Dyna

```
Require:
  States $\mathcal{S} = \{1, \ldots, n_x\}$
  Actions $\mathcal{A} = \{1, \ldots, n_a\}$,        $A : \mathcal{X} \Rightarrow \mathcal{A}$
  Reward function $R : \mathcal{X} \times \mathcal{A} \longrightarrow \mathbb{R}$
  Learning rate $\alpha \in [0, 1]$
  Discounting factor $\gamma \in [0, 1]$
  procedure MAXLIKELIHOODMODELBASEDRL($\mathcal{X}$, $A$, $R$, $\alpha$, $\gamma$, $\lambda$)
      $s_0 \longleftarrow$ initial state
      Initialize $Q$ arbitrarily
      $N, \rho, T, R \longleftarrow \phi$
      for episode in episodes do                            ▷ Loop through all simulated episodes
          while k in samples do
              $r_k \sim R(s_k, a_k)$                                  ▷ Observe the reward
              $s_{k+1} \sim T_(s_k, a_k)$                             ▷ Observe the new state
              Increment $N(s_k, a_k, s_{k+1}$
              Increment $\rho(s_k, a_k)$
              Update $T, R$ (Max Likelihood)
              Update $Q$ using $T, R$ (Dyna)
      return $Q$
```

V. Results

Table 2 Policy Generation Approaches : Performance

Algorithm	#Episodes	Training Time	#Iterations to Converge	Advantage over Random Policy
SARSA	900 episodes	260 sec	>200	50.58%
Q-Learning	900 episodes	233 sec	160	62.88%
Model-Based RL	900 episodes	$\sim 38\,min$	N.A.	49.6%

As we can see from the plots for optimal policies, the vehicle attempts to slow down for significant bumps on the road i.e. large change of gradients. Model-based RL seems to perform worse than expected as we use randomized updates over a fraction of the state space. In future, prioritized sweeping could be used to address this lack of performance.

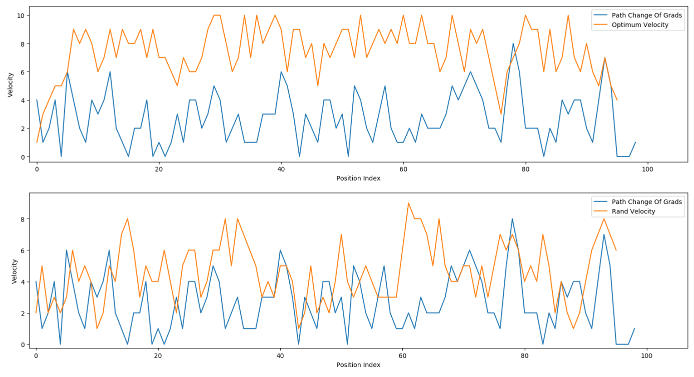

Fig. 1 SARSA vs Random Policy Velocity Profile.

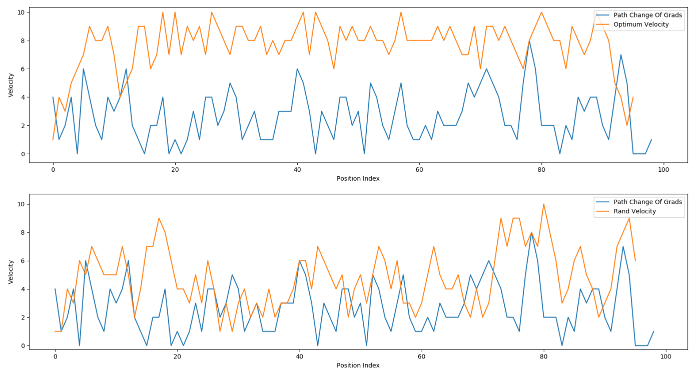

Fig. 2 Q-Learning vs Random Policy Velocity Profile.

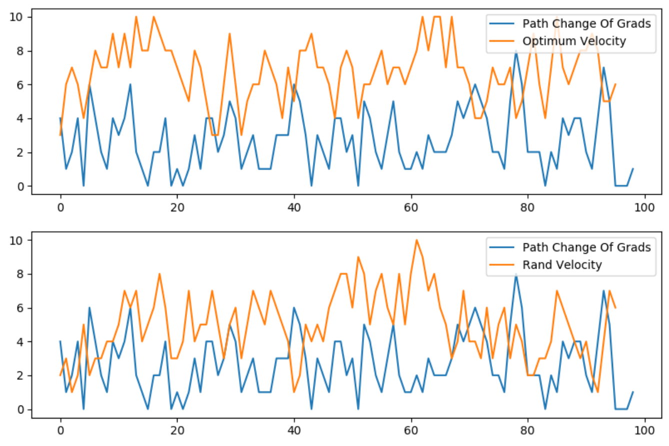

Fig. 3 Model-Based RL vs Random Policy Velocity Profile.

Augmented Horizon

Having observed a significant improvement in the optimal policy with the additon of a single observed change of gradient to the state space, we attemped to include a larger horizon of observed change of gradients. This attempt proved to be detrimental as the various iterations of this approach resulted in a maximum performance improvement of 71% with 10^4 episodes, apart from the added state space size and run time increase associated with it.

VI. Conclusion

In this project we formulated an MDP for vehicle motion planning problem based on a terrain profile, first by exploring the results of a no horizon based model-free reinforcement learning approach to generating a velocity profile over a given terrain length and profile and later using these observations to explore a possible alternative approach in the form of iterative Q-learning with finite terrain horizon information and introduce methods that reduce state space complexity and increase test performance with the additional state information in the form of observed terrain information. A combination of these methods resulted in a respectable improvement in overall optimum policy performance over a random policy.

Even with these improvements, it was evident that the performance could be improved using many different techniques. A possible approach explored was to increase the number of observed change of gradients to increase the finite horizon information. However, with an increase in state vector, the size of the state space increases by a factor of 10. This results in a sparse $Q(s, a)$ information table which is undesirable. A possible approach is to use Deep Reinforcement learning methods to approximate the Q function to over come the need to store and traverse the expanded state space for optimum policy generation.

VII. Contributions

- Kshitij: Developed environment simulator and sampler. Augmented and tuned reward function with additional state considerations. Testing and fine tuning QLearning Algorithm for additional states, path and velocity profiles.
- Pranav: Formulated function for the reward calculation. Implemented the function for clamping the set of possible velocities at each state and adding noise.
- Varun: Also contributed towards developing the simulator, learning and testing pipelines. Implemented the model-based RL approach.

Link to Github Repo: https://github.com/varununayak/aa228_project

References

[1] Taghirad, H. D., and Esmailzadeh, E., "Automobile passenger comfort assured through LQG/LQR active suspension," *Journal of vibration and control*, Vol. 4, No. 5, 1998, pp. 603–618.

[2] Saranya, C., Unnikrishnan, M., Ali, S. A., Sheela, D., and Lalithambika, V., "Terrain Based D Algorithm for Path Planning," *IFAC-PapersOnLine*, Vol. 49, No. 1, 2016, pp. 178–182.

[3] Kochenderfer, M. J., *Decision making under uncertainty: theory and application*, MIT press, 2015.