Generative Adversarial Imitation Learning for Quadrupedal Locomotion using Unstructured Expert Demonstrations

Siddhant Gangapurwala

***Abstract*—Robotic locomotion tasks heavily rely on control solutions for considerably simplified mechanical models of the robotic system in development. Despite being regarded as a highly complex problem, locomotion tasks performed by animals and humans can be considered to be near-optimal and highly efficient. For this reason, the work presented in this report proposes the use of expert demonstrations performed by humans to infer an underlying policy that directs the expert behaviour, and further proposes the use of the learned policy for execution of locomotion tasks. The report details the experimental results observed upon implementing an imitation learning algorithm for the purpose of footstep planning. The work also demonstrates the feasibility of using a learned policy over states that are not included in the set of expert training examples thereby making the algorithm suitable for execution in environments that are significantly different from the training environments.**

I. Introduction

Much of robotics research aims to develop control solutions that, depending on the environment of operation, exploit the machine's dynamics in order to achieve an extraordinarily agile behaviour. This, however, is limited by the use of traditional control techniques such as model predictive control (MPC) [1] and quadratic programming (QP) [2] which are often based on existing simplified mechanical models. A model-based optimization strategy employed over these simplified models eventually results in a mechanically constrained and inefficient response thereby limiting the agility of the robotic system in development.

Treating the control of robotic systems as a reinforcement learning (RL) problem enables the use of model-free algorithms that attempt to learn a policy which maximizes the expected future (discounted) reward without inferring the effects of an executed action on the environment. Authors of [3], [4] and [5] have successfully implemented these strategies for various robotics applications including control of robotic manipulators, helicopter aerobatics and even quadrupedal locomotion. However, despite the successful implementation of these RL algorithms for the mentioned tasks, one of the main challenges faced in solving an RL problem is defining a re function in order to learn an optimal policy resulting in a sensible robotic behaviour. Often, this reward function has to be tuned by a human expert.

As a solution to the reward function tuning required in RL problems, inverse reinforcement learning (IRL) [6] problem can be characterized as - given expert trajectories of an agent in a variety of circumstances, determine the reward function to be minimized. The recovered reward function is then used to generate a desired policy for a given environment. Authors of [7] and [8] have successfully implemented IRL techniques for perception and control tasks. However, the need for an extra step of solving an RL problem upon having recovered a reward function adds to the training time. Thus, instead of directly recovering a reward function, the problem can be designed as that of imitation learning [9], [10] in which an agent is trained to perform a task from demonstrations by learning a mapping between observations and actions.

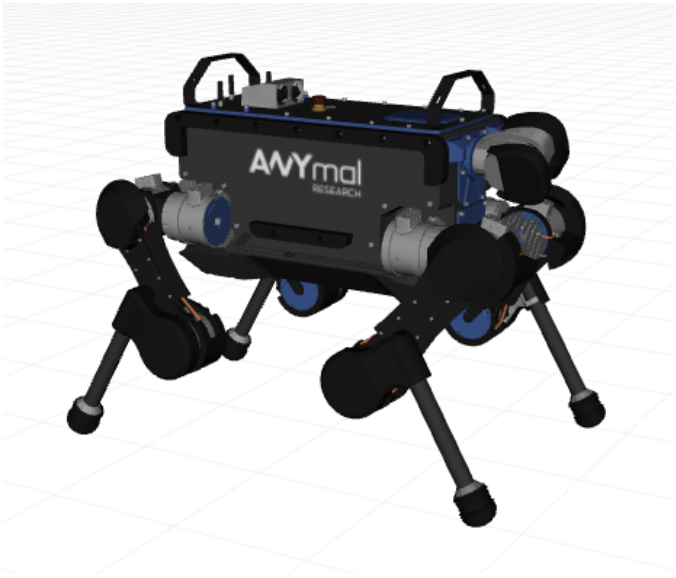

Figure 1. Model of the ANYmal robot used for this work. The hokuyo laser scanner is attached to the front of the robot, and the elevation map is generated using this sensor.

This work extends upon the technique used by authors of [11] on generative adversarial imitation learning (GAIL) to quadrupedal locomotion. GAIL bypasses the intermediate IRL step of recovering the reward function and directly generates a policy that maps observations to actions eliminating the need for model-based optimization strategies. Much of the work on GAIL so far has been done either using unrealistic simulations such as MuJoCo environments [12] or on robotic manipulators without significant research being carried out in the field of mobile robotics. This report details the use of GAIL strategy for quadrupedal footstep planning using the elevation map [13] obtained by the light detection and ranging (LIDAR) remote sensing method. The work was carried out using the simulation for the ANYmal robot [14] (Figure 1).

All of the work described in this report, unless otherwise stated, was performed at Oxford Robotics Institute, University of Oxford.

Supervisor: Dr. Ioannis Havoutis, Oxford Robotics Institute, University of Oxford.

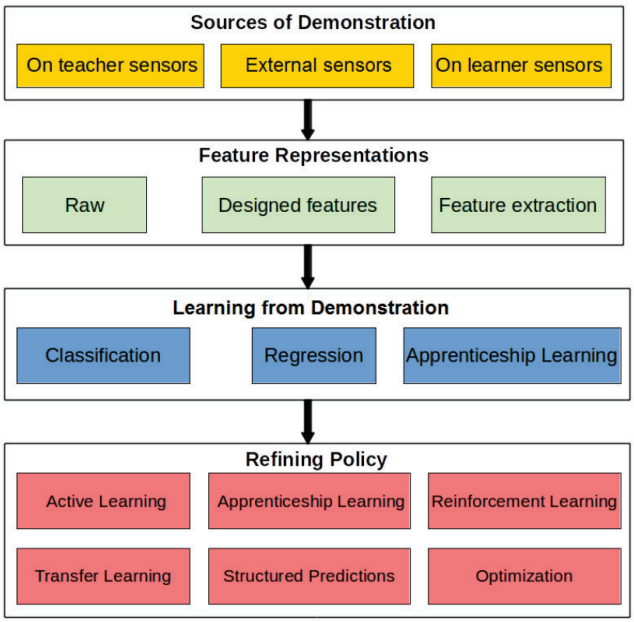

Figure 2. Imitation Learning Flowchart. Diagram by authors of [15].

II. Theory

The following subsections detail upon the techniques used in this work.

A. Learning from Demonstrations

As mentioned in [15], imitation learning draws its importance from its relevance to a variety of applications including flying vehicles, brain-computer interface, driverless cars, robotic manipulators, legged robots and computer games. It works by extracting information about the behaviour of an expert and the environment of operation and inferring a map between the state of the agent and the demonstrated actions. In a robotic system, the state may refer to the robot's position, joint angles, robot velocity, and, as in the case of this report, elevation map of the environment, whereas the actions may refer to the joint torque commands, acceleration of the robot, target velocity, or, as in the case of this report, displacement of each of the legs of a quadruped along the abscissa and ordinate.

The workflow of a typical imitation learning problem is as shown in Figure 2. Expert demonstrations are first acquired and are then encoded as state-action pairs. These state-action pairs are then used to train a policy. However, learning a direct mapping between state and action may not always result in a desired behaviour. This is often due to issues such as unstructured expert demonstrations, correspondence problem, or insufficient number of demonstrations.

Some of the definitions, taken from [15], required to understand the concept of imitation learning are introduced below.

***Definition** 1: The process of imitation learning is one by which an agent uses instances of performed actions to learn a policy that solves a given task.*

***Definition** 2: An agent is defined as an entity that autonomously interacts within an environment towards achieving or optimizing a goal [16]. An agent can be thought of as a software robot; it receives information from the environment by sensing or communication and acts upon the environment using a set of actuators.*

***Definition** 3: A policy is a function that maps a state (a description of the agent, such as pose, positions and velocities of various parts of the skeleton, and its relevant surrounding) to an action. It is what the agent uses to decide which action to execute when presented with a situation.*

***Definition** 4: A demonstration is presented as a pair of input and output (x, y). Where x is a vector of features describing the state at that instant and y is the action performed by the demonstrator*

***Definition** 5: An experience is presented as a tuple (s, a, r, s') where s is the state, a is the action taken at state s, r is the reward received for performing action a, and s' is the new state resulting from that action.*

***Definition** 6: A policy that uses t in learning the parameters of the policy is called a non-stationary policy (also known as non-autonomous policy [17]) i.e the policy takes into consideration at what stage of the task the agent is currently acting.*

***Definition** 7: A stationary policy (autonomous) neglects the time parameter and learns one policy for all steps in an action sequence.*

***Definition** 8: An action $u(t)$ can often represent a vector rather than a single value. This means that the action is comprised of more than one decision executed simultaneously; such as pressing multiple buttons on a controller or moving multiple joints in a robot.*

***Definition** 9: Low level actions are those that execute simple commands such as move forward and turn in navigation tasks, or jump and shoot in games.*

***Definition** 10: Motor primitives are simple building blocks that are executed in sequence to perform complex motions. An action is broken down into basic unit actions (often concerning one degree of freedom or actuator) that can be used to make up any action that needs to be performed in the given problem.*

***Definition** 11: High level macro actions are decisions that determine the immediate plan of the agent. It is then broken down to lower level action sequences. Examples of high level decisions are grasp object or perform forehand.*

The imitation learning problem can be introduced in the framework of Markov decision process (MDP) [18]. An MDP is a discrete time stochastic control process [19] in which, at each time step, the process is in a state s and an action a is chosen from the available actions in state s. The process then shifts to a new state s' with a probability given by the state transition function $\Pr_a(s, s')$ returning a corresponding reward $R_a(s, s')$. In an MDP, the next state s' is conditionally dependent on the current state s and the chosen action a, however, the current state s is independent of all previous actions and states.

To summarise, an MDP is a tuple (S, A, P_a, R_a, γ), where

- S is a set of states,
- A is a set of actions,
- $\Pr_a(s, s') = P(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that action a in state s at time t will lead to state s' at time $t+1$,
- $R_a(s, s')$ is the immediate reward received after transitioning from state s to s' as a result of action a,
- $\gamma \in [0, 1]$ is the discount factor.

In an MDP [20], an agent follows a policy $\pi : S \rightarrow A$ from a policy space Π that determines which action to take in the current state s. Upon taking action a, the agent receives a reward $R_a(s, s')$ bounded in $[0, 1]$ after moving to new state s' with a transition probability $P(s'|s, a)$. The state distribution at time t following a policy π from time 1 to $t-1$ is denoted as d_π^t, and the average state distribution of states over T steps is denoted as d_π. The T-step expected reward of policy π is thus given by

$$J(\pi) = \sum_{t=1}^{T} \mathbb{E}_{s \sim d_\pi^t} \left[R\left(s, \pi\left(s\right)\right) \right] = T\, \mathbb{E}_{s \sim d_\pi} \left[R\left(s, \pi\left(s\right)\right) \right].$$

A trajectory is a complete sequence of $\langle s, a, R\left(s, \pi\left(s\right)\right) \rangle$ tuples from $t = 1$ to $t = T$. The goal of the problem is to learn a policy $\pi \in \Pi$ that maximizes the task reward $J(\pi)$. In imitation learning, an expert which executes policy π^* and demonstrates actions $a_s^* = \arg\max_{a \in A} R(s, a)$ in state s is available. The learning algorithm only attempts to imitate the expert's behaviour without any notion of the task reward function. Thus maximizing the task reward is reduced to maximizing a *surrogate reward* with respect to the expert's policy.

One of the methods used for imitation learning is imitation by classification. In this approach, the expert's trajectories are used as supervised data to learn a multiclass classifier (policy) which predicts the expert action under distribution of states induced by running the expert's policy. At each step t, a training sample $\left(s_t, \pi^*\left(s_t\right)\right)$, where $\pi^*\left(s_t\right)$ is the expert's action (class label) in state s_t. Consider a surrogate reward function $r\left(s, \pi, \pi^*\left(s\right)\right)$. This could be any concave reward function used for training the classifier, for example, negative hinge loss in support vector machine (SVM). Thus, a supervised learning algorithm can be used to learn a policy such that

$$\hat{\pi} = \arg\max_{\pi \in \Pi} r\left(s, \pi, \pi^*\left(s\right)\right).$$

Supervised learning approach, however, ignores the fact that state distribution for the learner may be different than for the expert. Thus, when the learner attempts to mimic the expert in a situation where the state distribution is different from that of the expert, it cannot perform classification correctly resulting in the execution of an action that can potentially change the following state distribution thereby significantly increasing the negative reward. Eventually, the learner may reach a state where it performs random actions.

So as to correctly perform actions over the states for which the expert's demonstrations are not available, an inference scheme is necessary. The IRL problem tackles this by recovering the reward function for the agent through expert's demonstrations. An RL problem is then solved in an inner loop to generate policies. As an alternative to IRL techniques such as GCL [8], GAIL [11] draws a connection between imitation learning and generative adversarial networks (GAN) [21].

B. Generative Adversarial Imitation Learning

The concept of GAIL is largely based on IRL as opposed to behavioural cloning (BC). In case of BC the learner learns a policy as a supervised learning problem over state-action pairs from expert trajectories. Using IRL avoids the issues associated with BC such as that of covariant shifts. The authors of [11] adopted the technique of maximum causal entropy IRL [22], [23] in order to rationalize an expert policy. The technique fits a cost (negative reward) function c from a family of functions C with the optimization problem

$$\underset{c \in C}{\text{maximize}} \left(\min_{\pi \in \Pi} -H\left(\pi\right) + \mathbb{E}_\pi \left[c\left(s, a\right) \right] \right) - \mathbb{E}_{\pi^*} \left[c\left(s, a\right) \right]$$

where $H\left(\pi\right) \triangleq \mathbb{E}_\pi \left[-\log \pi\left(a|s\right) \right]$ is the γ-discounted causal entropy [24] of the policy π. Maximum causal entropy IRL looks for a cost function $c \in C$ that assigns low cost to the expert policy and high cost to other policies. The expert policy can be thus found using a certain RL procedure such that,

$$RL\left(c\right) = \arg\min_{\pi \in \Pi} -H\left(\pi\right) + \mathbb{E}_\pi \left[c\left(s, a\right) \right]$$

which maps a cost function to policies with high entropy that minimize the cumulative cost. The authors of [11] define an IRL primitive procedure $IRL_\psi(\pi^*)$ which finds a cost function such that the expert performs better than all other policies, with the convex cost function regularizer ψ by,

$$\arg\max_{c \in \mathbb{R}^{S \times A}} -\psi\left(c\right) + \left(\min_{\pi \in \Pi} -H\left(\pi\right) + \mathbb{E}_\pi \left[c\left(s, a\right) \right] \right) - \mathbb{E}_{\pi^*} \left[c\left(s, a\right) \right]$$

where $\mathbb{R}^{S \times A} = \left\{ c : S \times A \rightarrow \mathbb{R} \right\}$ are all the cost functions learned by the IRL strategy on largest possible set of cost functions C. The cost regularizer used in [11] is given by

$$\psi_{GA}\left(c\right) \triangleq \begin{cases} \mathbb{E}_{\pi^*} \left[g\left(c\left(s, a\right)\right) \right] & \text{if } c < 0 \\ +\infty & \text{otherwise} \end{cases}$$

where

$$\mathrm{g}\left(x\right) = \begin{cases} -x - \log\left(1 - \exp(x)\right) & \text{if } x < 0 \\ +\infty & \text{otherwise} \end{cases}$$

The GAIL problem is constructed such that IRL is the dual of RL. The solution of the primal RL is considered optimal, and is also the solution as that recovered from the IRL cost function. The inspiration here is to characterize $RL \circ IRL_\psi\left(\pi^*\right)$. This is derived to be

$$RL \circ IRL_\psi\left(\pi^*\right) = \arg\min_{\pi \in \Pi} -H\left(\pi\right) + \psi_{\mathrm{GA}}^* \left(\rho_\pi - \rho_{\pi^*}\right)$$

where occupancy measure ρ_π is the distribution of state-action pair encountered while navigating the environment with

the policy π, and ψ^*_{GA}, the convex conjugate of ψ_{GA}. Here $\psi^*_{\text{GA}}(\rho_\pi - \rho_{\pi^*})$ is given by

$$\max_{D\in(0,1)^{S\times A}} \mathbb{E}_\pi\left[\log\left(D\left(s,a\right)\right)\right] + \mathbb{E}_{\pi^*}\left[\log\left(1-D\left(s,a\right)\right)\right].$$

where D refers to the discriminative classifiers.

Having characterized the problem as described above, the problem of imitation learning can be now considered as a procedure that tries to induce a policy which matches the expert's occupancy measure. The optimal loss up to a constant shift is the Jensen-Shannon divergence D_{JS}. Treating the causal entropy as a policy regularizer, controlled by $\lambda \geq 0$, we obtain an imitation learning algorithm given by

$$\underset{\pi}{\text{minimize}} -\lambda H\left(\pi\right) + \psi^*_{GA}\left(\rho_\pi - \rho_{\pi^*}\right) = D_{\text{JS}}\left(\rho_\pi, \rho_{\pi_E}\right) - \lambda H\left(\pi\right)$$

which finds a policy whose occupancy measure minimizes Jensen-Shannon divergence to the expert's.

In the the GAIL setting, the learner's occupancy measure is analogous to the data generated by G, and the expert's occupancy measure is analogous to the true data distribution. The discriminator function can then be considered as a local cost function providing learning signals to the policy. For the GAIL algorithm we aim to find the saddle point (π, D) of the expression

$$\mathbb{E}_\pi\left[\log\left(D\left(s,a\right)\right)\right] + \mathbb{E}_{\pi^*}\left[\log\left(1-D\left(s,a\right)\right)\right] - \lambda H\left(\pi\right).$$

To do so, function approximation for π and D are first introduced given by a parametrized policy π_θ with weights θ, and a discriminator network $D_\omega : S \times A \rightarrow (0,1)$ with weights ω. This is followed by alternating between an Adam [25] gradient step on ω to maximize the above equation with respect to D, and a policy gradient step using a method such as trust region policy optimization (TRPO) [26] or proximal policy optimization (PPO) [27] on θ to minimize the above equation with respect to π. This forms the basis for the GAIL algorithm (**Algorithm 1**) as described in [11].

Algorithm 1: Generative Adversarial Imitation Learning

Input: Expert trajectories $\tau^* \sim \pi^*$, initial policy and discriminator parameters $\theta_0,\ \omega_0$

for $i = 0,\ 1,\ 2,\ldots$ **do**

Sample trajectories $\tau_i \sim \pi_{\theta_i}$

Update the discriminator parameters from ω_i to ω_{i+1} with the gradient

$$\hat{\mathbb{E}}_{\tau_i}\left[\nabla_\omega \log\left(D\left(s,a\right)\right)\right] + \hat{\mathbb{E}}_{\tau^*}\left[\nabla_\omega \log\left(1-D\left(s,a\right)\right)\right]$$

Take a policy step from θ to θ_{i+1} using the TRPO rule with cost function $\log\left(D_{\omega_{i+1}(s,a)}\right)$. Specifically, take a KL-constrained natural gradient step with

$$\hat{\mathbb{E}}_{\tau_i}\left[\nabla_\theta \log \pi_\theta\left(a|s\right) Q\left(s,a\right)\right] - \lambda \nabla_\theta H\left(\pi_\theta\right)$$

where $Q\left(\overline{s},\overline{a}\right)$ is given by

$$\hat{\mathbb{E}}_{\tau_i}\left[\log\left(D_{\omega_{i+1}}\left(s,a\right)\right)|s_0=\overline{s}, a_0=\overline{a}\right]$$

end

III. Problem Statement

The motivation for the work detailed in this report was the fact that much of the work on imitation learning for robotics was performed on robotic systems such as robotic manipulators or on simulations in MuJoCo environments without significant work being done on mobile robots. Furthermore, much of the work that addresses tasks such as robotic locomotion heavily rely on using simplified mechanical models of the robotic systems in development, thus making the control of these systems inefficient. In order to address the challenges faced in developing solutions for robotic locomotion especially in case of legged robots, the strategy of learning from demonstrations, as in [11], [9], [28], seemed to provide a satisfactory solution for the experiments performed in the mentioned work, and was able to directly map raw feature space into actions. The GAIL algorithm was implemented in this work for its property of being able to directly generate a policy through IRL technique without the need for running an intermediate RL step. Moreover, GAIL provides a manner of inference which enables the use of the learned policy for mapping observations, which haven't been included in the demonstrations, to actions.

The work carried out as part of this project can be stated as: *planning sequence of footstep placements for ANYmal quadruped using elevation map and robot state as observations in order to ascend and descend a wide range of staircases using generative adversarial imitation learning approach by providing expert demonstrations for inference*. The problem statement was slightly modified from the original statement in which, instead of planning footstep placements, obtaining joint trajectories was proposed. However, the higher dimensionality associated with solving the problem for 12 joints in the quadruped resulted in trajectories which were not smooth (as described in the next section), and thus, not suitable for robotic control.

IV. Approach

The ANYmal development framework for the robot operating system (ROS) [29] was used as a platform for development and testing of the imitation learning strategy for this project. The overall flow of the project consisted of:

1) Task demonstration and trajectory data collection
2) Training the GAIL algorithm on the collected data
3) Deploying the learned model for task execution

A. Expert Demonstrations and Trajectory Data Collection

For the purpose of demonstrations, various world models were created in Gazebo [30] some of which are shown in Figure 3. The expert data collected was in the form of state-action pairs wherein the state consisted of a 200×200 elevation map matrix and an 8-dimensional vector consisting the robot end effector positions relative to the robot's base. Figure 4 shows the visual representation of the elevation map in Rviz [31]. The actions recorded were the displacements of the end-effector of each of the legs in x and z direction relative to the world frame. These are derived from the footstep

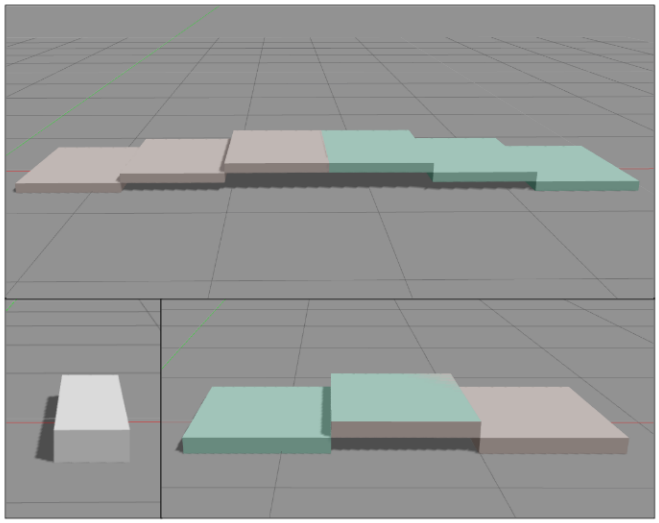

Figure 3. World models created using Gazebo.

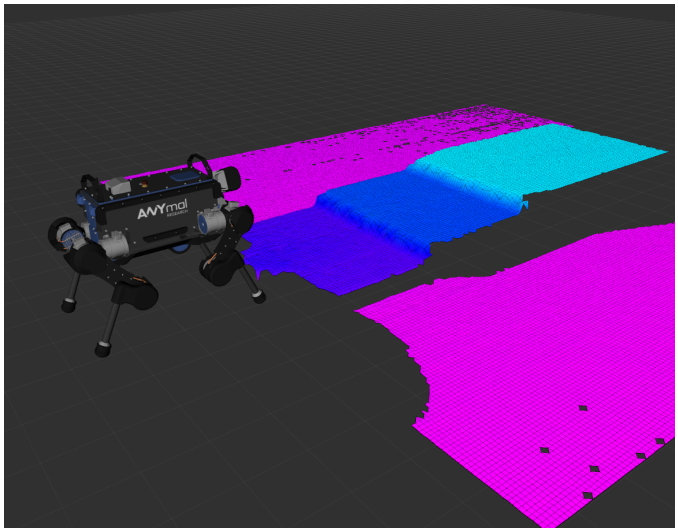

Figure 4. Visual representation of the elevation map in Rviz for a certain Gazebo world. The elevation map consists of 200×200 grid cells each of size 0.02×0.02 m^2

locations of the robot. The expert demonstrations were given by preselecting the positions of the each of the legs' end effector for each set of steps. The locations of these preplanned footsteps are as represented in Figure 5. The state-action pairs were recorded after execution of each of the steps. It was assumed that step execution effectively meant losing contact from ground and then re-establishing it. To realize this, the contact force sensors were used. A zero force implied the step was being executed. After the sensors returned a non-zero value, the execution was considered to be completed, and the state and action data was recorded.

For the purpose of the original problem statement for which change in joint positions was considered to be an action, the state consisted of the elevation map along with a 12 dimensional vector representing the joint states. The recorded trajectory was sampled every 100ms. Compared to footstep

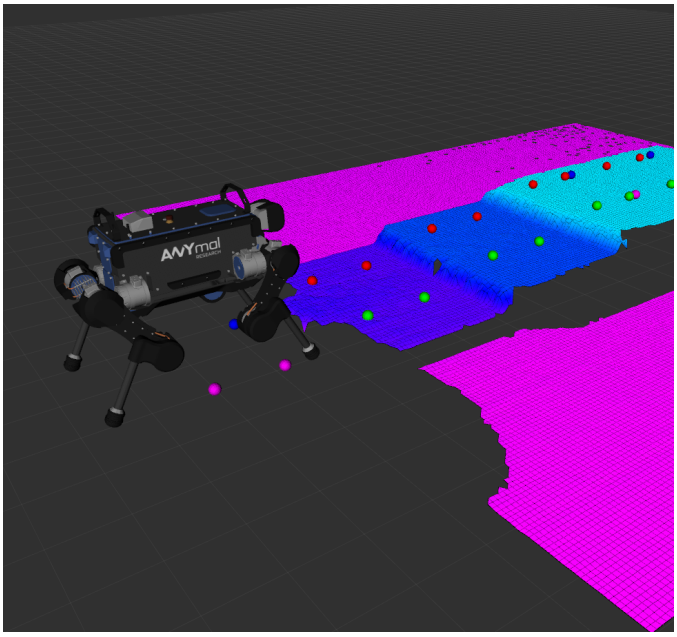

Figure 5. The coloured globs represent the expert footstep plan for each of the legs of the quadruped.

planning, this problem statement was more complex due to the high sampling rate and the higher dimensionality associated with the action space. The action was a 12 dimensional vector compared to 8 for the footstep planning case.

B. Learning a Policy from Expert Demonstrations

The GAIL algorithm was used to train a policy from the expert data collected in the previous step. The training was done in a different manner for each of the problem statements as previously mentioned. However, the general model for both the approaches is similar, and is represented in Figure 6.

1) Obtaining Joint Trajectory: As described in the previous subsection, the action space for obtaining joint trajectories using GAIL spanned 12 dimensions. Furthermore, the dataset for this problem statement consisted of a lot more sample points that for footstep planning. The training was done in a manner such that the previous four state vectors were used to form the state representation vector as in Figure 6. Having obtained the data, the GAIL algorithm with the losses defined in *Section II* were used to train the neural networks with a PPO step for generator policy optimization.

2) Planning Sequence of Footsteps: The training method used for this approach was similar to that used for obtaining joint trajectory with the exception that more training examples were required for this problem. The training was a lot slower especially because the simulations in Gazebo had to be run every time new samples were required. Since Gazebo only supports real-time execution of tasks, the time consumed in generating the expert trajectories was significantly high.

C. Executing the Learned Policy

Upon training the generator by the method shown in Figure 6, the policy was tested for its performance on Gazebo

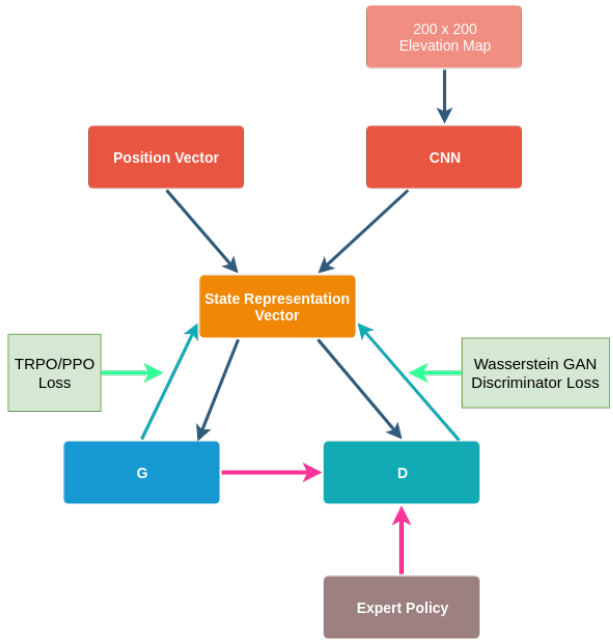

Figure 6. The flow of the algorithm implemented for imitation learning using GANs.

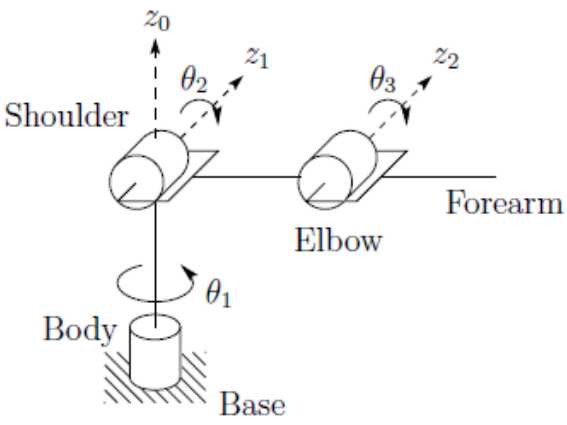

Figure 7. RRR robotic manipulator for which the FK equations were derived and then implemented for each of the legs of ANYmal to compute the end effector position.

worlds that had not been included in the training set. The trained policy was initially tested for the problem statement wherein the change in joint angles were obtained as actions. The 12-dimensional output from the generator network was then used to compute the position of the end-effector every 100 ms using forward kinematic (FK) [32] equations for an RRR manipulator as shown in Figure 7. Upon computing the end-effector positions, the Freegait controller provided as part of the ANYmal development framework was used to execute step actions.

For the footstep planning problem, the Freegait controller was directly implemented for the outputs generated by the generator. No additional computations were required. Furthermore, since the actions were generated only after execution of

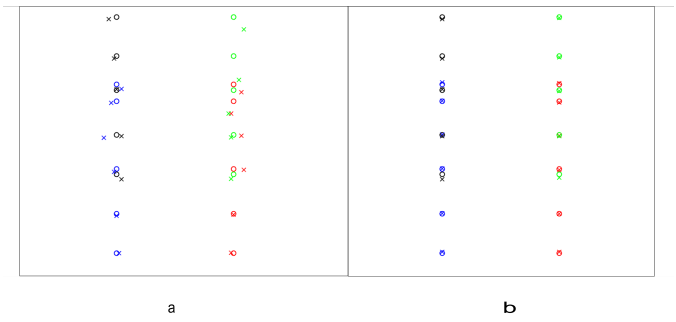

Figure 8. The figures represent the footstep placement for a certain test Gazebo world. The circular markers represent the expert's footstep placement, while the cross markers represent the output of the GAIL algorithm. a) Observed output for obtaining joint trajectories using GAIL. b) GAIL for footstep placement planning.

each step, which lasted for almost 1 s, there was no delay in the output generated by the network. Thus, the system could be executed real-time.

V. Observations and Results

The trained policy was used to generate actions for states that had been included in the training data. This was done in order to test the ability of the GAIL algorithm to learn a policy that matched the expert's.

It was observed that for the problem statement of obtaining joint trajectories, the GAIL algorithm failed to generate trajectories that were smooth. This added to the effect of drift in the y-direction which eventually caused the robot to become unstable. As can be seen in Figure 8, the learned policy did not closely match the expert's policy. However, for the problem of planning footsteps, imitation learning was observed to generate actions that closely matched the expert's. In this case, the GAIL strategy satisfactorily managed to learn the expert's policy.

Through the experiments conducted with GAIL for world models that had not been a part of the training set, it was observed that the algorithm performed well for new enviroments. It successfully managed to ascend and descend 4 out of 7 test staircases. Out of the other 3 staircases, the algorithm managed to ascend the staircase for 2 world models, however, failed to descend. This was largely due to the fact that the steps were not observable. The laser scanner failed to detect the steps that were right underneath the scanner. For the other 1 test staircase, the algorithm failed to plan footsteps that could result in a stable ascent. It was observed that along the edges of the steps, the algorithm generated foot sequences that were extremely close to the edges. In 3 test cases, the footstep plan was quite close to resulting in the quadruped flipping over.

VI. Conclusion and Future Work

The report reviewed the performance of the GAIL algorithm for its application in robotic locomotion. Having observed the behaviour for the joint trajectory problem statement, it was concluded that the high dimensional problem was not suitable for execution using the imitation strategy. Instead, the learned

policy performed a lot better for a low dimensional task of footstep planning. The performance of GAIL, however, could not be quantitatively defined. The learned policy performed well for the task of ascending and descending a staircase but the dynamic nature of the problem implies that the expert trajectories only cover a subset of possible states rendering the need for inference necessary. GAIL managed to execute action for states that weren't included in the demonstrations thereby being able to infer the expert's behaviour up to a satisfactory extent. To improve its performance however, a significantly bigger training set is necessary.

The task of ascending and descending a staircase is comparatively easier to imitate than a task which includes walking over a rough terrain. With this regard, the future work shall include environments where footstep placements heavily rely on the dynamic stability of the robot. Furthermore, the work shall include policy optimization techniques such as genetic distillation instead of PPO for its better sampling efficiency.

In conclusion, the GAIL algorithm was successfully implemented for the ANYmal robot for the task of ascending and descending a staircase.

REFERENCES

[1] D. Dimitrov, A. Sherikov, and P. B. Wieber. "A sparse model predictive control formulation for walking motion generation". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 2292–2299. DOI: 10.1109/IROS.2011.6095035.

[2] Scott Kuindersma, Frank Permenter, and Russ Tedrake. "An Efficiently Solvable Quadratic Program for Stabilizing Dynamic Locomotion". In: *CoRR* abs/1311.1839 (2013). arXiv: 1311.1839. URL: http://arxiv.org/abs/1311.1839.

[3] Shixiang Gu, Ethan Holly, Timothy P. Lillicrap, et al. "Deep Reinforcement Learning for Robotic Manipulation". In: *CoRR* abs/1610.00633 (2016). arXiv: 1610.00633. URL: http://arxiv.org/abs/1610.00633.

[4] Andrew Y. Ng, Adam Coates, Mark Diel, et al. "Autonomous Inverted Helicopter Flight via Reinforcement Learning". In: *Experimental Robotics IX*. Ed. by Marcelo H. Ang and Oussama Khatib. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 363–372. ISBN: 978-3-540-33014-1.

[5] N. Kohl and P. Stone. "Policy gradient reinforcement learning for fast quadrupedal locomotion". In: *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*. Vol. 3. 2004, 2619–2624 Vol.3. DOI: 10.1109/ROBOT.2004.1307456.

[6] Andrew Y. Ng and Stuart J. Russell. "Algorithms for Inverse Reinforcement Learning". In: *Proceedings of the Seventeenth International Conference on Machine Learning*. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 663–670. ISBN: 1-55860-707-2. URL: http://dl.acm.org/citation.cfm?id=645529.657801.

[7] Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, et al. "Large-scale cost function learning for path planning using deep inverse reinforcement learning". In: *The International Journal of Robotics Research* (2017), p. 0278364917722396. DOI: 10.1177/0278364917722396. URL: http://dx.doi.org/10.1177/0278364917722396.

[8] Chelsea Finn, Sergey Levine, and Pieter Abbeel. "Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization". In: *CoRR* abs/1603.00448 (2016). arXiv: 1603.00448. URL: http://arxiv.org/abs/1603.00448.

[9] A. Billard and D. Grollman. "Robot learning by demonstration". In: *Scholarpedia* 8.12 (2013). revision #138061, p. 3824. DOI: 10.4249/scholarpedia.3824.

[10] Pieter Abbeel, Adam Coates, and Andrew Y Ng. "Autonomous helicopter aerobatics through apprenticeship learning". In: *The International Journal of Robotics Research* 29.13 (2010), pp. 1608–1639.

[11] Jonathan Ho and Stefano Ermon. "Generative adversarial imitation learning". In: *Advances in Neural Information Processing Systems*. 2016, pp. 4565–4573.

[12] E. Todorov, T. Erez, and Y. 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.

[13] Péter Fankhauser, Michael Bloesch, Christian Gehring, et al. "Robot-Centric Elevation Mapping with Uncertainty Estimates". In: *International Conference on Climbing and Walking Robots (CLAWAR)*. 2014.

[14] Marco Hutter, Christian Gehring, Dominic Jud, et al. "Anymal-a highly mobile and dynamic quadrupedal robot". In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 38–44.

[15] Ahmed Hussein, Mohamed Gaber, Eyad Elyan, et al. "Imitation Learning: A Survey of Learning Methods". In: 50 (Apr. 2017).

[16] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 2nd ed. Pearson Education, 2003. ISBN: 0137903952.

[17] Stefan Schaal, Auke Ijspeert, and Aude Billard. "Computational approaches to motor learning by imitation". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 358.1431 (2003), pp. 537–547.

[18] Alexandre Attia and Sharone Dayan. "Global overview of Imitation Learning". In: *arXiv preprint arXiv:1801.06503* (2018).

[19] Richard Bellman. "A Markovian Decision Process". In: *Indiana Univ. Math. J.* 6 (4 1957), pp. 679–684. ISSN: 0022-2518.

[20] He He, Jason Eisner, and Hal Daume. "Imitation learning by coaching". In: *Advances in Neural Information Processing Systems*. 2012, pp. 3149–3157.

[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[22] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, et al. “Maximum Entropy Inverse Reinforcement Learning.” In: *AAAI*. Vol. 8. Chicago, IL, USA. 2008, pp. 1433–1438.

[23] Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

[24] Michael Bloem and Nicholas Bambos. “Infinite time horizon maximum causal entropy inverse reinforcement learning”. In: *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE. 2014, pp. 4911–4916.

[25] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[26] John Schulman, Sergey Levine, Pieter Abbeel, et al. “Trust region policy optimization”. In: *International Conference on Machine Learning*. 2015, pp. 1889–1897.

[27] John Schulman, Filip Wolski, Prafulla Dhariwal, et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[28] Josh Merel, Yuval Tassa, Dhruva TB, et al. “Learning human behaviors from motion capture by adversarial imitation”. In: *CoRR* abs/1707.02201 (2017). arXiv: 1707.02201. URL: http://arxiv.org/abs/1707.02201.

[29] Morgan Quigley, Ken Conley, Brian Gerkey, et al. “ROS: an open-source Robot Operating System”. In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5.

[30] Nathan Koenig and Andrew Howard. “Design and use paradigms for gazebo, an open-source multi-robot simulator”. In: *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*. Vol. 3. IEEE. 2004, pp. 2149–2154.

[31] Hyeong Ryeol Kam, Sung-Ho Lee, Taejung Park, et al. “Rviz: a toolkit for real domain data visualization”. In: *Telecommunication Systems* 60.2 (2015), pp. 337–345.

[32] Mark W Spong, Seth Hutchinson, Mathukumalli Vidyasagar, et al. *Robot modeling and control*. Vol. 3. Wiley New York, 2006.